Amazon EMR
Developer Guide
# Table of Contents

What is Amazon EMR? ............................................................................................................. 1  
Resources ................................................................................................................................ 2  
What Can You Do with Amazon EMR? .................................................................................... 3  
  Hadoop Programming on Amazon EMR ............................................................................. 3  
  Data Analysis and Processing on Amazon EMR ............................................................... 4  
  Data Storage on Amazon EMR .......................................................................................... 4  
  Move Data with Amazon EMR ......................................................................................... 4  
Amazon EMR Features .......................................................................................................... 4  
  Resizeable Clusters ........................................................................................................... 5  
  Pay Only for What You Use ............................................................................................. 5  
  Easy to Use ....................................................................................................................... 5  
  Use Amazon S3 or HDFS ................................................................................................. 5  
  Parallel Clusters ............................................................................................................... 5  
  Hadoop Application Support ............................................................................................ 5  
  Save Money with Spot Instances ...................................................................................... 5  
  AWS Integration ............................................................................................................... 6  
  Instance Options .............................................................................................................. 6  
  MapR Support ................................................................................................................... 6  
  Business Intelligence Tools ............................................................................................... 6  
  User Control ....................................................................................................................... 6  
  Management Tools ............................................................................................................ 6  
  Security ............................................................................................................................ 6  
How Does Amazon EMR Work? ............................................................................................ 7  
  Hadoop ............................................................................................................................. 7  
  Nodes ................................................................................................................................ 8  
  Steps ................................................................................................................................ 8  
  Cluster .............................................................................................................................. 10  
What Tools are Available for Amazon EMR? .......................................................................... 11  
Learn More About Hadoop and AWS Services Used with Amazon EMR: .......................... 12  
Getting Started .................................................................................................................... 14  
  Tutorial Costs .................................................................................................................... 14  
  Step 1: Create an AWS Account ....................................................................................... 14  
  Step 2: Create an Amazon S3 Bucket for Your Cluster Logs and Output Data .............. 15  
  Step 3: Launch an Amazon EMR Cluster ........................................................................ 16  
  Step 4: Run the Hive Script as a Step ............................................................................. 21  
    Hive Script Overview ...................................................................................................... 21  
    Submit the Hive Script as a Step ................................................................................... 22  
    View the Results ............................................................................................................ 22  
  Step 5: Query Your Data Using Hue ............................................................................... 23  
    Create an SSH Tunnel to the Master Node ................................................................... 23  
    Log into Hue and Submit an Interactive Hive Query .................................................... 24  
  (Optional) Step 6: Explore Amazon EMR ........................................................................ 25  
  (Optional) Step 7: Remove the Resources Used in the Tutorial ...................................... 25  
Plan and Configure Clusters ................................................................................................. 27  
  Configure Cluster Location and Data Storage .................................................................. 27  
    Choose an AWS Region .................................................................................................. 27  
    Work with Storage and File Systems .......................................................................... 29  
    Prepare Input Data ......................................................................................................... 31  
    Configure an Output Location ....................................................................................... 40  
Using EMR File System (EMRFS) .......................................................................................... 45  
  Consistent View .............................................................................................................. 46  
  Authorizing Access to EMRFS Data in Amazon S3 .......................................................... 60  
  Specifying Amazon S3 Encryption Using EMRFS Properties .......................................... 61  
  Configure a Cluster to be Transient or Long-Running .................................................... 67
Configure Cluster Software ................................................................. 68
Choose an Amazon Machine Image (AMI) .................................................. 69
Choose a Version of Hadoop ........................................................................ 120
(Optional) Create Bootstrap Actions to Install Additional Software .......... 129
Configure Cluster Hardware and Networking ............................................ 139
Master Node .................................................................................. 139
Core Nodes .................................................................................. 139
Task Nodes .................................................................................. 140
Instance Fleets .............................................................................. 140
Uniform Instance Groups .................................................................... 140
Plan and Configure EC2 Instances ...................................................... 140
Plan and Configure Networking ........................................................ 145
Create a Cluster with Instance Fleets or Uniform Instance Groups .......... 153
Configure Cluster Logging and Debugging ............................................. 167
Default Log Files ........................................................................ 167
Archive Log Files to Amazon S3 .......................................................... 168
Enable the Debugging Tool ................................................................ 170
Debugging Option Information ......................................................... 171
Tag Clusters .................................................................................. 171
Tag Restrictions ........................................................................... 172
Tag Resources for Billing .................................................................. 172
Add Tags to a New Cluster ............................................................... 173
Adding Tags to an Existing Cluster ..................................................... 174
View Tags on a Cluster ................................................................... 174
Remove Tags from a Cluster ............................................................... 175
Drivers and Third-Party Application Integration ..................................... 176
Use Business Intelligence Tools with Amazon EMR ............................... 176
Parse Data with HParser ................................................................ 176
Using the MapR Distribution for Hadoop .............................................. 177
Security ...................................................................................... 187
Use IAM Policies to Allow and Deny User Permissions ......................... 187
Amazon EMR Actions in User-Based IAM Policies .............................. 188
Use Managed Policies for User Access .............................................. 188
Use Cluster Tagging with IAM Policies for Cluster-Specific Control ...... 190
Use Kerberos Authentication ............................................................. 194
Supported Applications ..................................................................... 195
Configure Kerberos ......................................................................... 195
Configure a Cluster-Dedicated KDC ................................................... 200
Configure a Cross-Realm Trust .......................................................... 202
Use an Amazon EC2 Key Pair for SSH Credentials ............................... 206
Encrypt Data in Transit and at Rest ...................................................... 207
Understanding Encryption Options ...................................................... 207
Creating Keys and Certificates for Data Encryption ............................ 211
EMRFS Authorization for Data in Amazon S3 ........................................ 214
How EMRFS Authorization Works ..................................................... 214
Set Up EMRFS Authorization ........................................................... 214
Control Network Traffic with Security Groups ........................................ 216
Use Amazon EMR–Managed Security Groups ..................................... 217
Configure Additional Security Groups ................................................ 221
Use Security Configurations to Set Up Cluster Security ......................... 223
Create a Security Configuration ....................................................... 223
Specify a Security Configuration for a Cluster ................................... 234
Configure IAM Roles for Amazon EMR Permissions to AWS Services .... 234
Use Default IAM Roles and Managed Policies .................................... 235
Allow Users and Groups to Create and Modify Roles ............................ 241
Customize IAM Roles .................................................................... 241
Specify Custom IAM Roles When You Create a Cluster ...................... 242
Sign up for the Service ................................................................. 302
Launch the Cluster ................................................................. 302
Generate Test Data ................................................................. 303
Create and Populate Impala Tables ....................................... 303
Query Data in Impala ............................................................. 304
Impala Examples Included on the Amazon EMR AMI .......... 305
TPCDS ......................................................................................... 305
Wikipedia ..................................................................................... 306
Supported Impala Versions ...................................................... 307
Updates for Impala 1.2.4 .......................................................... 308
Impala Memory Considerations ............................................... 308
Using Impala with JDBC .......................................................... 309
Accessing Impala Web User Interfaces ................................. 309
Impala-supported File and Compression Formats .................. 310
Impala SQL Dialect ................................................................. 310
Impala User-Defined Functions .............................................. 310
Impala Performance Testing and Query Optimization .......... 310
Database Schema ................................................................. 311
Sample Data ............................................................................. 311
Table Size .................................................................................. 312
Queries ...................................................................................... 312
Performance Test Results ...................................................... 313
Optimizing Queries ................................................................. 316
Apache Pig ................................................................. 318
Supported Pig Versions ............................................................ 318
Pig Version Details ................................................................. 320
Additional Pig Functions .......................................................... 321
Interactive and Batch Pig Clusters .......................................... 322
Submit Pig Work ......................................................................... 322
Submit Pig Work Using the Amazon EMR Console .............. 322
Submit Pig Work Using the AWS CLI ...................................... 323
Call User Defined Functions from Pig ................................. 323
Call JAR files from Pig ............................................................ 323
Call Python/Jython Scripts from Pig ...................................... 324
HBase .......................................................................................... 325
HBase Release Information for This Release of Amazon EMR ... 325
What Can I Do with HBase? .................................................... 325
Reference data for Hadoop analytics .................................... 325
Real-time log ingestion and batch log analytics .................... 325
Store for high frequency counters and summary data .......... 325
Supported HBase Versions ....................................................... 326
HBase Cluster Prerequisites ..................................................... 326
Install HBase on an Amazon EMR Cluster .............................. 327
Using the HBase Shell ............................................................ 332
Create a Table ........................................................................... 332
Put a Value .............................................................................. 332
Get a Value .............................................................................. 332
Access HBase Tables with Hive .............................................. 333
Back Up and Restore HBase .................................................... 334
Back Up and Restore HBase Using the Console ................. 335
Back Up and Restore HBase Using the AWS CLI ................. 336
Terminate an HBase Cluster ..................................................... 339
Configure HBase ....................................................................... 339
Configure HBase Daemons ...................................................... 340
Configure HBase Site Settings ................................................ 341
HBase Site Settings to Optimize ............................................. 343
View the HBase User Interface ................................................. 344
What is Amazon EMR?

With Amazon EMR (Amazon EMR) you can analyze and process vast amounts of data. It does this by distributing the computational work across a cluster of virtual servers running in the Amazon cloud. The cluster is managed using an open-source framework called Hadoop.

Hadoop uses a distributed processing architecture called MapReduce in which a task is mapped to a set of servers for processing. The results of the computation performed by those servers is then reduced down to a single output set. One node, designated as the master node, controls the distribution of tasks. The following diagram shows a Hadoop cluster with the master node directing a group of slave nodes which process the data.

Amazon EMR has made enhancements to Hadoop and other open-source applications to work seamlessly with AWS. For example, Hadoop clusters running on Amazon EMR use EC2 instances as virtual Linux servers for the master and slave nodes, Amazon S3 for bulk storage of input and output data, and CloudWatch to monitor cluster performance and raise alarms. You can also move data into and out of DynamoDB using Amazon EMR and Hive. All of this is orchestrated by Amazon EMR control software that launches and manages the Hadoop cluster. This process is called an Amazon EMR cluster.

The following diagram illustrates how Amazon EMR interacts with other AWS services.
Open-source projects that run on top of the Hadoop architecture can also be run on Amazon EMR. The most popular applications, such as Hive, Pig, HBase, DistCp, and Ganglia, are already integrated with Amazon EMR.

By running Hadoop on Amazon EMR you get the benefits of the cloud:

- The ability to provision clusters of virtual servers within minutes.
- You can scale the number of virtual servers in your cluster to manage your computation needs, and only pay for what you use.
- Integration with other AWS services.

**Resources**

The following related resources can help you as you work with this service.

- **Classes & Workshops** – Links to role-based and specialty courses as well as self-paced labs to help sharpen your AWS skills and gain practical experience.
- **AWS Developer Tools** – Links to developer tools, SDKs, IDE toolkits, and command line tools for developing and managing AWS applications.
- **AWS Whitepapers** – Links to a comprehensive list of technical AWS whitepapers, covering topics such as architecture, security, and economics and authored by AWS Solutions Architects or other technical experts.
- **AWS Support Center** – The hub for creating and managing your AWS Support cases. Also includes links to other helpful resources, such as forums, technical FAQs, service health status, and AWS Trusted Advisor.
What Can You Do with Amazon EMR?

Amazon EMR simplifies running Hadoop and related big-data applications on AWS. You can use it to manage and analyze vast amounts of data. For example, a cluster can be configured to process petabytes of data.

Topics
- Hadoop Programming on Amazon EMR (p. 3)
- Data Analysis and Processing on Amazon EMR (p. 4)
- Data Storage on Amazon EMR (p. 4)
- Move Data with Amazon EMR (p. 4)

Hadoop Programming on Amazon EMR

In order to develop and deploy custom Hadoop applications, you used to need access to a lot of hardware for your Hadoop programs. Amazon EMR makes it easy to spin up a set of EC2 instances as virtual servers to run your Hadoop cluster. You can run various server configurations, such as fully-loaded production servers and temporary testing servers, without having to purchase or reconfigure hardware. Amazon EMR makes it easy to configure and deploy your always-on production clusters, but also to easily terminate unused testing clusters after your development and testing phase is complete.

Amazon EMR provides several methods of running Hadoop applications, depending on the type of program you are developing and the libraries you intend to use.

Custom JAR

Run your custom MapReduce program written in Java. Running a custom JAR gives you low-level access to the MapReduce API. You have the responsibility of defining and implementing the MapReduce tasks in your Java application.
Cascading

Run your application using the Cascading Java library, which provides features such as splitting and joining data streams. Using the Cascading Java library can simplify application development. With Cascading you can still access the low-level MapReduce APIs as you can with a Custom JAR application.

Streaming

Run a Hadoop job based on Map and Reduce functions you upload to Amazon S3. The functions can be implemented in any of the following supported languages: Ruby, Perl, Python, PHP, R, Bash, C++.

Data Analysis and Processing on Amazon EMR

You can also use Amazon EMR to analyze and process data without writing a line of code. Several open-source applications run on top of Hadoop and make it possible to run MapReduce jobs and manipulate data using either a SQL-like syntax or a specialized language called Pig Latin. Amazon EMR is integrated with Apache Hive and Apache Pig.

Data Storage on Amazon EMR

Distributed storage is a way to store large amounts of data over a distributed network of computers with redundancy to protect against data loss. Amazon EMR is integrated with the Hadoop Distributed File System (HDFS) and Apache HBase.

Move Data with Amazon EMR

You can use Amazon EMR to move large amounts of data in and out of databases and data stores. By distributing the work, the data can be moved quickly. Amazon EMR provides custom libraries to move data in and out of Amazon Simple Storage Service (Amazon S3), DynamoDB, and Apache HBase.

Amazon EMR Features

Using Amazon EMR to run Hadoop on Amazon Web Services offers many advantages.

Topics

- Resizeable Clusters (p. 5)
- Pay Only for What You Use (p. 5)
- Easy to Use (p. 5)
- Use Amazon S3 or HDFS (p. 5)
- Parallel Clusters (p. 5)
- Hadoop Application Support (p. 5)
- Save Money with Spot Instances (p. 5)
- AWS Integration (p. 6)
- Instance Options (p. 6)
- MapR Support (p. 6)
- Business Intelligence Tools (p. 6)
- User Control (p. 6)
- Management Tools (p. 6)
Resizeable Clusters

When you run your Hadoop cluster on Amazon EMR, you can easily expand or shrink the number of virtual servers in your cluster depending on your processing needs. Adding or removing servers takes minutes, which is much faster than making similar changes in clusters running on physical servers.

Pay Only for What You Use

By running your cluster on Amazon EMR, you only pay for the computational resources you use. You do not pay ongoing overhead costs for hardware maintenance and upgrades and you do not have to pre-purchase extra capacity to meet peak needs. For example, if the amount of data you process in a daily cluster peaks on Monday, you can increase the number of servers to 50 in the cluster that day, and then scale back to 10 servers in the clusters that run on other days of the week. You won’t have to pay to maintain those additional 40 servers during the rest of the week as you would with physical servers. For more information, see Amazon Elastic MapReduce Pricing.

Easy to Use

When you launch a cluster on Amazon EMR, the web service allocates the virtual server instances and configures them with the needed software for you. Within minutes you can have a cluster configured and ready to run your Hadoop application.

Use Amazon S3 or HDFS

The version of Hadoop installed on Amazon EMR clusters is integrated with Amazon S3, which means that you can store your input and output data in Amazon S3, on the cluster in HDFS, or a mix of both. Amazon S3 can be accessed like a file system from applications running on your Amazon EMR cluster.

Parallel Clusters

If your input data is stored in Amazon S3 you can have multiple clusters accessing the same data simultaneously.

Hadoop Application Support

You can use popular Hadoop applications such as Hive, Pig, and HBase with Amazon EMR. For more information, see Hive and Amazon EMR (EMR 3.x Releases) (p. 260), Apache Pig (p. 318), and Apache HBase (p. 325).

Save Money with Spot Instances

Spot Instances are a way to purchase virtual servers for your cluster at a discount. Excess capacity in Amazon Web Services is offered at a fluctuating price, based on supply and demand. You set a maximum bid price that you wish pay for a certain configuration of virtual server. While the price of Spot Instances for that type of server are below your bid price, the servers are added to your cluster and you are billed the Spot Price rate. When the Spot Price rises above your bid price, the servers are terminated.

For more information about how use Spot Instances effectively in your cluster, see When Should You Use Spot Instances? (p. 164).
AWS Integration

Amazon EMR is integrated with other Amazon Web Services such as Amazon EC2, Amazon S3, DynamoDB, Amazon RDS, CloudWatch, and AWS Data Pipeline. This means that you can easily access data stored in AWS from your cluster and you can make use of the functionality offered by other Amazon Web Services to manage your cluster and store the output of your cluster.

For example, you can use Amazon EMR to analyze data stored in Amazon S3 and output the results to Amazon RDS or DynamoDB. Using CloudWatch, you can monitor the performance of your cluster and you can automate recurring clusters with AWS Data Pipeline. As new services are added, you'll be able to make use of those new technologies as well. For more information, see Monitor Metrics with CloudWatch (p. 434) and Export, Import, Query, and Join Tables in DynamoDB Using Amazon EMR (p. 393).

Instance Options

When you launch a cluster on Amazon EMR, you specify the size and capabilities of the virtual servers used in the cluster. This way you can match the virtualized servers to the processing needs of the cluster. You can choose virtual server instances to improve cost, speed up performance, or store large amounts of data.

For example, you might launch one cluster with high storage virtual servers to host a data warehouse, and launch a second cluster on virtual servers with high memory to improve performance. Because you are not locked into a given hardware configuration as you are with physical servers, you can adjust each cluster to your requirements. For more information about the server configurations available using Amazon EMR, see Plan and Configure EC2 Instances (p. 140).

MapR Support

Amazon EMR supports several MapR distributions. For more information, see Using the MapR Distribution for Hadoop (p. 177).

Business Intelligence Tools

Amazon EMR integrates with popular business intelligence (BI) tools such as Tableau, MicroStrategy, and Datameer. For more information, see Use Business Intelligence Tools with Amazon EMR (p. 176).

User Control

When you launch a cluster using Amazon EMR, you have root access to the cluster and can install software and configure the cluster before Hadoop starts. For more information, see (Optional) Create Bootstrap Actions to Install Additional Software (p. 129).

Management Tools

You can manage your clusters using the Amazon EMR console (a web-based user interface), a command line interface, web service APIs, and a variety of SDKs. For more information, see What Tools are Available for Amazon EMR? (p. 11).

Security

You can run Amazon EMR in a Amazon VPC in which you configure networking and security rules. Amazon EMR also supports IAM users and roles which you can use to control access to your cluster and permissions that restrict what others can do on the cluster. For more information, see Use IAM Policies to Allow and Deny User Permissions (p. 187).
How Does Amazon EMR Work?

Amazon EMR is a service you can use to run managed Hadoop clusters on Amazon Web Services. A Hadoop cluster is a set of servers that work together to perform computational tasks by distributing the work and data among the servers. The task might be to analyze data, store data, or to move and transform data. By using several computers linked together in a cluster, you can run tasks that process or store vast amounts (petabytes) of data.

When Amazon EMR launches a Hadoop cluster, it runs the cluster on virtual servers provided by Amazon EC2. Amazon EMR has made enhancements to the version of Hadoop it installs on the servers to work seamlessly with AWS. This provides several advantages, as described in Amazon EMR Features (p. 4).

In addition to integrating Hadoop with AWS, Amazon EMR adds some new concepts to distributed processing such as nodes and steps.

Topics
- Hadoop (p. 7)
- Nodes (p. 8)
- Steps (p. 8)
- Cluster (p. 10)

Hadoop

Apache Hadoop is an open-source Java software framework that supports massive data processing across a cluster of servers. It can run on a single server, or thousands of servers. Hadoop uses a programming model called MapReduce to distribute processing across multiple servers. It also implements a distributed file system called HDFS that stores data across multiple servers. Hadoop monitors the health of servers in the cluster, and can recover from the failure of one or more nodes. In this way, Hadoop provides not only increased processing and storage capacity, but also high availability.

For more information, see http://hadoop.apache.org.

Topics
- MapReduce (p. 7)
- HDFS (p. 7)
- Jobs and Tasks (p. 8)
- Hadoop Applications (p. 8)

MapReduce

MapReduce is a programming model for distributed computing. It simplifies the process of writing parallel distributed applications by handling all of the logic except 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.

For more information, see http://wiki.apache.org/hadoop/HadoopMapReduce.

HDFS

Hadoop Distributed File System (HDFS) is a distributed, scalable, and portable file system for Hadoop. HDFS distributes the data it stores across servers in the cluster, storing multiple copies of data on different servers to ensure that no data is lost if an individual server fails. HDFS is ephemeral storage
that is reclaimed when you terminate the cluster. HDFS is useful for caching intermediate results during MapReduce processing or as the basis of a data warehouse for long-running clusters.


Amazon EMR extends Hadoop to add the ability to reference data stored in Amazon S3 as if it was a file system like HDFS. You can use either HDFS or Amazon S3 as the file system in your cluster. If you store intermediate results in Amazon S3, however, be aware that data will stream between every slave node in the cluster and Amazon S3. This could potentially overrun the limit of 200 transactions per second to Amazon S3. Most often, Amazon S3 is used to store input and output data and intermediate results are stored in HDFS.

**Jobs and Tasks**

In Hadoop, a job is a unit of work. Each job may consist of one or more tasks, and each task may be attempted one or more times until it succeeds. Amazon EMR adds a new unit of work to Hadoop, the step, which may contain one or more Hadoop jobs. For more information, see [Steps](p. 8).

You can submit work to your cluster in a variety of ways. For more information, see [How to Send Work to a Cluster](p. 10).

**Hadoop Applications**

Hadoop is a popular open-source distributed computing architecture. Other open-source applications such as Hive, Pig, and HBase run on top of Hadoop and extend its functionality by adding features such as queries of data stored on a cluster and data warehouse functionality.

For more information, see [Hive and Amazon EMR (EMR 3.x Releases)](p. 260), [Apache Pig](p. 318), and [Apache HBase](p. 325).

**Nodes**

Amazon EMR defines three roles for the servers in a cluster. These different roles are referred to as node types. The Amazon EMR node types map to the master and slave roles defined in Hadoop.

- **Master node** — Manages the cluster: coordinating the distribution of the MapReduce executable and subsets of the raw data, to the core and task instance groups. It also tracks the status of each task performed, and monitors the health of the instance groups. There is only one master node in a cluster. This maps to the Hadoop master node.
- **Core nodes** — Runs tasks and stores data using the Hadoop Distributed File System (HDFS). This maps to a Hadoop slave node.
- **Task nodes (optional)** — Run tasks. This maps to a Hadoop slave node.

For more information, see [Create a Cluster with Instance Fleets or Uniform Instance Groups](p. 153). For details on mapping legacy clusters to instance groups, see [Mapping Legacy Clusters to Instance Groups](p. 490).

**Steps**

Amazon EMR defines a unit of work called a step, which can contain one or more Hadoop jobs. A step is an instruction that manipulates the data. For example, a cluster that processes encrypted data might contain the following steps:

- **Step 1: Decrypt data**
- **Step 2: Process data**
• Step 3: Encrypt data
• Step 4: Save data

You can track the progress of steps by checking their state. The following diagram shows the processing of a series of steps.

A cluster contains one or more steps. Steps are processed in the order in which they are listed in the cluster. Steps are run following this sequence: all steps have their state set to PENDING. The first step is run and the step's state is set to RUNNING. When the step is completed, the step's state changes to COMPLETED. The next step in the queue is run, and the step's state is set to RUNNING. After each step completes, the step's state is set to COMPLETED and the next step in the queue is run. Steps are run until there are no more steps. Processing flow returns to the cluster.

If a step fails, the step state is FAILED and all remaining steps with a PENDING state are marked as CANCELLED. No further steps are run and processing returns to the cluster.

Data is normally communicated from one step to the next using files stored on the cluster's Hadoop Distributed File System (HDFS). Data stored on HDFS exists only as long as the cluster is running. When the cluster is shut down, all data is deleted. The final step in a cluster typically stores the processing results in an Amazon S3 bucket.

For a complete list of step states, see the StepExecutionStatusDetail data type in the Amazon EMR API Reference.

Beginning with AMI 3.1.1 (Hadoop 2.x) and AMI 2.4.8 (Hadoop 1.x), the maximum number of PENDING and ACTIVE steps allowed in a cluster is 256 (this includes system steps such as install Pig, install Hive, install HBase, and configure debugging). You can submit an unlimited number of steps over
the lifetime of a long-running cluster created using these AMIs, but only 256 steps can be ACTIVE or PENDING at any given time. For more information about adding steps to a cluster, see Submit Work to a Cluster (p. 493).

Cluster

A cluster is a set of servers that perform the work. In Amazon EMR the cluster is a set of virtual servers running as EC2 instances.

How to Send Work to a Cluster

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

- Provide the entire definition of the work to be done in the Map and Reduce functions. This is typically done for clusters that process a set amount of data and then terminate when processing is complete. For more information, see Run a Hadoop Application to Process Data (p. 249).
- Create a long-running cluster and use the console, the Amazon EMR API, the AWS CLI or the Amazon EMR CLI to submit steps, which may contain one or more Hadoop jobs. For more information, see Submit Work to a Cluster (p. 493).
- Create a cluster with a Hadoop application such as Hive, Pig, or HBase installed, and use the interface provided by the application to submit queries, either scripted or interactively. For more information, see Hive and Amazon EMR (EMR 3.x Releases) (p. 260), Apache Pig (p. 318), and Apache HBase (p. 325).
- Create a long-running cluster, connect to it, and submit Hadoop jobs using the Hadoop API. For more information, see http://hadoop.apache.org/docs/current/api/org/apache/hadoop/mapred/JobClient.html.

Life Cycle of a Cluster

The following diagram shows the life cycle of a cluster and how each stage maps to a particular cluster state.

A successful Amazon EMR cluster follows this process: Amazon EMR first provisions a Hadoop cluster. During this phase, the cluster state is STARTING. Next, any user-defined bootstrap actions are run. During this phase, the cluster state is BOOTSTRAPPING.
What Tools are Available for Amazon EMR?

There are several ways you can interact with Amazon EMR:

- **Console** — a graphical 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/](https://console.aws.amazon.com/elasticmapreduce/).

- **AWS CLI (Command Line Interface)** — a client application you run on your local machine to connect to Amazon EMR and create and manage clusters. The AWS CLI contains a feature-rich set of commands specific to Amazon EMR. With it, you can write scripts that automate the process of launching and managing clusters. Using the AWS CLI is the best option if you prefer working from a command line. For more information on using the AWS CLI, see [http://docs.aws.amazon.com/cli/latest/reference/emr](http://docs.aws.amazon.com/cli/latest/reference/emr).

- **Amazon EMR CLI** — a legacy client application you run on your local machine to connect to Amazon EMR and create and manage clusters. With it, you can write scripts that automate the process of launching and managing clusters. The Amazon EMR CLI is no longer under feature development. Customers using the Amazon EMR CLI are encouraged to migrate to the AWS CLI. New users should download the AWS CLI, not the Amazon EMR CLI. For more information on using the Amazon EMR CLI, see [Command Line Interface Reference for Amazon EMR](http://aws.amazon.com/code/Elastic-MapReduce/).

- **Software Development Kit (SDK)** — AWS provides an SDK with functions that call Amazon EMR to create and manage clusters. With it, you can write applications that automate the process of creating and managing clusters. Using the SDK is the best option if you want to extend or customize the functionality of Amazon EMR. You can download the AWS SDK for Java from [http://aws.amazon.com/sdkforjava/](http://aws.amazon.com/sdkforjava/). For more information about the AWS SDKs, refer to the list of current AWS SDKs. Libraries are available for Java, C#, VB.NET, and PHP. For more information, see [Sample Code & Libraries](http://aws.amazon.com/code/Elastic-MapReduce/).

- **Web Service API** — AWS provides a low-level interface that you can use to call the web service directly using JSON. Using the API is the best option if you want to create an custom SDK that calls Amazon EMR. For more information, see the [Amazon EMR API Reference](http://aws.amazon.com/code/Elastic-MapReduce/).
The following table compares the functionality of the Amazon EMR interfaces.

<table>
<thead>
<tr>
<th>Function</th>
<th>Console</th>
<th>AWS CLI</th>
<th>API, SDK, and Libraries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create multiple clusters</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Define bootstrap actions in a cluster</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>View logs for Hadoop jobs, tasks, and task attempts using a graphical interface</td>
<td>✔️</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Implement Hadoop data processing programmatically</td>
<td></td>
<td></td>
<td>✔️</td>
</tr>
<tr>
<td>Monitor clusters in real time</td>
<td></td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td>Provide verbose cluster details</td>
<td></td>
<td></td>
<td>✔️</td>
</tr>
<tr>
<td>Resize running clusters</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Run clusters with multiple steps</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Select version of Hadoop, Hive, and Pig</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Specify the MapReduce executable in multiple computer languages</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Specify the number and type of EC2 instances that process the data</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Transfer data to and from Amazon S3 automatically</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Terminate clusters</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>

Learn More About Hadoop and AWS Services Used with Amazon EMR:

- Amazon Elastic Compute Cloud (Amazon EC2), Amazon Simple Storage Service (Amazon S3), and CloudWatch. For more information, see the [Amazon EC2 User Guide for Linux Instances](http://docs.aws.amazon.com/AmazonEC2/latest/UserGuide/), the [Amazon EC2 User Guide for Windows Instances](http://docs.aws.amazon.com/AmazonEC2/latest/UserGuide/) for Windows users.
Services Used with Amazon EMR:

Getting Started

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

In this tutorial, you launch a long-running Amazon EMR cluster using the console. In addition to the console used in this tutorial, Amazon EMR provides a command-line client, a REST-like API, and several SDKs that you can use to launch and manage clusters. For more information about these interfaces, see What Tools are Available for Amazon EMR? (p. 11).

After launching the cluster, you run a Hive script to analyze a series of CloudFront web distribution log files. After running the script, you query your data using the Hue web interface.

Topics
- Tutorial Costs (p. 14)
- Step 1: Create an AWS Account (p. 14)
- Step 2: Create an Amazon S3 Bucket for Your Cluster Logs and Output Data (p. 15)
- Step 3: Launch an Amazon EMR Cluster (p. 16)
- Step 4: Run the Hive Script as a Step (p. 21)
- Step 5: Query Your Data Using Hue (p. 23)
- (Optional) Step 6: Explore Amazon EMR (p. 25)
- (Optional) Step 7: Remove the Resources Used in the Tutorial (p. 25)

Tutorial Costs

The AWS service charges incurred by completing this tutorial include the cost of running an Amazon EMR cluster containing 3 m3.xlarge instances for one hour and the cost of storing log and output data in Amazon S3. The total cost of this tutorial is approximately $1.05 (depending on your region). Your actual costs may differ slightly from this estimate.

Service charges vary by region. If you are a new customer, within your first year of using AWS, the Amazon S3 storage charges are potentially waived, given you have not used the capacity allowed in the Free Usage Tier. Amazon EC2 and Amazon EMR charges resulting from this tutorial are not included in the Free Usage Tier, but they are minimal.

AWS service pricing is subject to change. For current pricing information, see the AWS Service Pricing Overview and use the AWS Simple Monthly Calculator to estimate your bill.

Step 1: Create an AWS Account

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Before you begin, you must have an AWS account. If you have an account, proceed to the next step. If you do not have an AWS account, use the following procedure to create one.

To sign up for AWS

2. Follow the online instructions.

AWS notifies you by email when your account is active and available for you to use. Your AWS account gives you access to all services, but you are charged only for the resources that you use.

For console access, use your IAM user name and password to sign in to the AWS Management Console using the IAM sign-in page. IAM lets you securely control access to AWS services and resources in your AWS account. For more information about creating access keys, see How Do I Get Security Credentials? in the AWS General Reference.

Step 2: Create an Amazon S3 Bucket for Your Cluster Logs and Output Data

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Amazon EMR can use Amazon S3 to store input data, log files, and output data. Amazon S3 refers to these storage locations as buckets. This section shows you how to use the Amazon S3 console to create a bucket that stores your cluster logs and your output data. For more information about using Amazon S3 with Hadoop, go to http://wiki.apache.org/hadoop/AmazonS3.

Creating a path for your cluster logs is optional. When you launch a cluster using the console, if you do not specify an Amazon S3 log location, one is generated for you automatically.

To create an Amazon S3 bucket using the console

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose Create Bucket.
3. In the Create a Bucket dialog box:
   - Type a bucket name, such as myemrbucket. The bucket name should be globally unique. If the name you type is in use by another bucket, type a different name.
     
     **Note**
     Because of Hadoop requirements, Amazon S3 bucket names used with Amazon EMR must contain only lowercase letters, numbers, periods (.), and hyphens (-). Also, bucket names cannot end in numbers.
   - For Region, choose a region for your bucket. To avoid paying cross-region bandwidth charges, create the Amazon S3 bucket in the same region as your cluster.
4. Choose Create.
5. In the list, select your bucket name and choose Create Folder.
6. For Name, type output and then press Enter. This creates the following path for your output data: s3://myemrbucket/output.
7. (Optional) Choose Create Folder again.
8. For Name, type logs and then press Enter. This creates the following path for your cluster logs: s3://myemrbucket/logs.

**Note**
This step is optional. If you do not create the /logs folder before launching the cluster, the console generates it for you based on what you type in the Log folder S3 location field. You can also allow the console to generate a log path for you automatically.
Step 3: Launch an Amazon EMR Cluster

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

The next step is to launch the Amazon EMR cluster. When you launch a cluster, Amazon EMR provisions Amazon EC2 instances (virtual servers) to perform the computation. These instances are created using an Amazon Machine Image (AMI) customized for Amazon EMR. The AMI has Hadoop and other big data applications preloaded.

To launch an Amazon EMR cluster using the console

If you do not need to retain the cluster logs, you may disable the logging option.

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. Choose Create cluster.
3. On the Create Cluster page, in the Cluster Configuration section, accept the default options. These options are defined in the following table.

<table>
<thead>
<tr>
<th>Field</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster name</td>
<td>When you create a cluster, the default name is &quot;My cluster.&quot; You can also type a descriptive name for your cluster. The name is optional, and does not need to be unique.</td>
</tr>
<tr>
<td>Termination protection</td>
<td>By default, clusters created using the console have termination protection enabled (set to Yes). Enabling termination protection ensures that the cluster does not shut down due to accident or error. Typically, you enable termination protection when developing an application (so you can debug errors that would have otherwise terminated the cluster), to protect long-running clusters, or to preserve data. For more information, see Managing Cluster Termination (p. 474).</td>
</tr>
<tr>
<td>Logging</td>
<td>By default, clusters created using the console have logging enabled. This option determines whether Amazon EMR writes detailed log data to Amazon S3. When this value is set, Amazon EMR copies the log files from the EC2 instances in the cluster to Amazon S3. Logging to Amazon S3 can only be enabled when the cluster is created. Logging to Amazon S3 prevents the log files from being lost when the cluster ends and the EC2 instances hosting the cluster are terminated. These logs are useful for troubleshooting purposes. For more information, see View Log Files (p. 422).</td>
</tr>
<tr>
<td>Log folder S3 location</td>
<td>You can type or browse to your Amazon S3 bucket to store the Amazon EMR logs; for example, s3://myemrbucket/logs, or you can allow Amazon EMR to generate an Amazon S3 path for you. If you type the name of a folder that does not exist in the bucket, it is created for you.</td>
</tr>
</tbody>
</table>
Step 3: Launch an Amazon EMR Cluster

### Field | Action
--- | ---
**Debugging** | By default, when logging is enabled, debugging is also enabled. This option creates an Amazon SQS exchange to process debugging messages. For more information about SQS, see the Amazon SQS product description page. For more information about debugging, see Debugging Option Information (p. 171).

4. In the **Tags** section, leave the options blank. You do not use tags in this tutorial. Tagging allows you to categorize resources using key-value pairs. Tags on Amazon EMR clusters are propagated to the underlying Amazon EC2 instances.

5. In the **Software Configuration** section, accept the default options. These options are defined in the following table.

### Field | Action
--- | ---
**Hadoop distribution** | This option determines which distribution of Hadoop to run on your cluster. By default the Amazon distribution of Hadoop is selected, but you can choose to run one of several MapR distributions instead.

For more information about MapR, see Using the MapR Distribution for Hadoop (p. 177).

**AMI version** | Amazon Elastic MapReduce (Amazon EMR) uses Amazon Machine Images (AMIs) to initialize the EC2 instances it launches to run a cluster. The AMIs contain the Linux operating system, Hadoop, and other software used to run the cluster. These AMIs are specific to Amazon EMR and can be used only in the context of running a cluster. By default, the latest Hadoop 2.x AMI is selected. You can also choose a particular Hadoop 2.x AMI or a particular Hadoop 1.x AMI from the list.

The AMI you choose determines the specific version of Hadoop and other applications such as Hive or Pig to run on your cluster. When you use the console to choose an AMI, deprecated AMIs are not shown in the list.

For more information on choosing an AMI, see Choose an Amazon Machine Image (AMI) (p. 69).

**Applications to be installed** | When you choose the latest Hadoop 2.x AMI, Hive, Pig, and Hue are installed by default. The applications installed and the application versions change depending on the AMI that you select. You can remove pre-selected applications by choosing the **Remove** icon.

**Additional applications** | This option allows you to install additional applications. When you choose an AMI, applications not available on the AMI do not appear in the list.

6. In the **File System Configuration** section, accept the default options for EMRFS. EMRFS is an implementation of HDFS which allows Amazon EMR clusters to store data on Amazon S3. The default options for EMRFS are defined in the following table.

### Field | Action
--- | ---
**Server-side encryption** | When using the console to create a cluster, server-side encryption is deselected by default. This option enables Amazon S3 server-side encryption for EMRFS.

**Consistent view** | When using the console to create a cluster, consistent view is deselected by default. This option enables consistent view for EMRFS. When enabled, you
Step 3: Launch an Amazon EMR Cluster

7. In the **Hardware Configuration** section, for the Core EC2 instance type, choose m3.xlarge and accept the remaining default options. These options are defined in the following table.

**Note**
Twenty is the default maximum number of nodes per AWS account. For example, if you have two clusters, the total number of nodes running for both clusters must be 20 or less. Exceeding this limit results in cluster failures. If you need more than 20 nodes, you must submit a request to increase your Amazon EC2 instance limit. Ensure that your requested limit increase includes sufficient capacity for any temporary, unplanned increases in your needs. For more information, go to the [Request to Increase Amazon EC2 Instance Limit Form](#).

<table>
<thead>
<tr>
<th>Field</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network</td>
<td>When using the console to create a cluster, the default VPC is selected automatically. If you have additional VPCs, you may choose an alternate VPC from the list. For more information about the default VPC, see <a href="#">Your Default VPC and Subnets</a>.</td>
</tr>
<tr>
<td>EC2 Subnet</td>
<td><strong>No preference</strong> is selected by default which allows Amazon EMR to choose a random subnet. Alternatively, you can choose a particular VPC subnet identifier from the list. For more information about choosing a VPC subnet, see <a href="#">Plan and Configure Networking</a>.</td>
</tr>
<tr>
<td>Master</td>
<td>The master instance assigns Hadoop tasks to core and task nodes and monitors their status. Amazon EMR clusters must contain 1 master node. The master node is contained in a master instance group. For more information about Amazon EMR instance groups, see <a href="#">Create a Cluster with Instance Fleets or Uniform Instance Groups</a>.</td>
</tr>
<tr>
<td></td>
<td>• <strong>EC2 instance type</strong> determines the type of virtual server used to launch the Amazon EMR master node. The instance type you choose determines the virtual computing environment for the node: processing power, storage capacity, memory, and so on. For more information about instance types supported by Amazon EMR, see <a href="#">Virtual Server Configurations</a>.</td>
</tr>
<tr>
<td></td>
<td>§ The default master instance type for Hadoop 2.x is m3.xlarge. This instance type is suitable for testing, development, and light workloads.</td>
</tr>
<tr>
<td></td>
<td>§ By default, <strong>Count</strong> is set to 1 for the master node. Currently, there is only one master node per cluster.</td>
</tr>
<tr>
<td></td>
<td>§ <strong>Request spot</strong> is unchecked by default. This option specifies whether to run master nodes on Spot Instances.</td>
</tr>
</tbody>
</table>
## Step 3: Launch an Amazon EMR Cluster

### Field

<table>
<thead>
<tr>
<th>Field</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>For more information about using Spot Instances, see When Should You Use Spot Instances? (p. 164).</td>
<td></td>
</tr>
</tbody>
</table>

### Core

- The core instances run Hadoop tasks and store data using the Hadoop Distributed File System (HDFS). Your cluster must contain at least 1 core node. Core nodes are contained in a core instance group.

For more information about Amazon EMR instance groups, see Create a Cluster with Instance Fleets or Uniform Instance Groups (p. 153).

- **EC2 instance type** determines the type of virtual server used to launch the Amazon EMR core nodes. The instance type you choose determines the virtual computing environment for the node: processing power, storage capacity, memory, and so on.

For more information about instance types supported by Amazon EMR, see Virtual Server Configurations.

- The default core instance type for Hadoop 2.x is m1.large. Be sure to change this to m3.xlarge. The m1.large instance type is not available in every region. The m3.xlarge instance type is suitable for testing, development, and light workloads.

- By default, **Count** is set to 2 for the core nodes.

- **Request spot** is unchecked by default. This option specifies whether to run core nodes on Spot Instances.

For more information about using Spot Instances, see When Should You Use Spot Instances? (p. 164).

### Task

- The task instances run Hadoop tasks. Task instances do not store data using HDFS. When task nodes are used, they are contained in a task instance group.

For more information about Amazon EMR instance groups, see Create a Cluster with Instance Fleets or Uniform Instance Groups (p. 153).

- **EC2 instance type** determines the type of virtual server used to launch the Amazon EMR task nodes. The instance type you choose determines the virtual computing environment for the node: processing power, storage capacity, memory, and so on.

For more information about instance types supported by Amazon EMR, see Virtual Server Configurations.

- The default task instance type for Hadoop 2.x is m1.medium. This instance type is suitable for testing, development, and light workloads.

- By default, **Count** is set to 0 for the task nodes. Using task nodes with Amazon EMR is optional. When the instance count for the task nodes is 0, a task instance group is not created.

- **Request spot** is unchecked by default. This option specifies whether to run task nodes on Spot Instances.

For more information about using Spot Instances, see When Should You Use Spot Instances? (p. 164).

8. In the **Security and Access** section, for **EC2 key pair**, choose your key pair from the list and accept the remaining default options. These options are defined in the following table.
Step 3: Launch an Amazon EMR Cluster

<table>
<thead>
<tr>
<th>Field</th>
<th>Action</th>
</tr>
</thead>
</table>
| EC2 key pair     | By default, the key pair option is set to **Proceed without an EC2 key pair**. This option prevents you from using SSH to connect to the master, core, and task nodes. You should choose your Amazon EC2 key pair from the list.  

For more information about using SSH to connect to the master node, see [Connect to the Master Node Using SSH](p. 457). |
| IAM user access  | **All other IAM users** is selected by default. This option makes the cluster visible and accessible to all IAM users on the AWS account.  

If you choose **No other IAM users**, access to the cluster is restricted to the current IAM user.  

For more information about configuring cluster access, see [Use IAM Policies to Allow and Deny User Permissions](p. 187). |
| IAM roles        | **Default** is selected automatically. This option generates the default EMR role and default EC2 instance profile. The EMR role and EC2 instance profile are required when creating a cluster using the console.  

If you choose **Custom** you can specify your own EMR role and EC2 instance profile.  

For more information about using IAM roles with Amazon EMR, see [Configure IAM Roles for Amazon EMR Permissions to AWS Services](p. 234). |

9. In the **Bootstrap Actions** section, accept the default option (none). Bootstrap actions are scripts that are executed during setup before Hadoop starts on every cluster node. You can use them to install additional software and customize your applications. This tutorial does not use bootstrap actions.  

For more information about using bootstrap actions, see [(Optional) Create Bootstrap Actions to Install Additional Software](p. 129).  

10. In the **Steps** section, accept the default options. These options are defined in the following table.  

<table>
<thead>
<tr>
<th>Field</th>
<th>Action</th>
</tr>
</thead>
</table>
| Add step| By default, no user-defined steps are configured.  

A step is a unit of work you submit to the cluster. A step might contain one or more Hadoop jobs, or contain instructions to install or configure an application. |
| Auto-terminate| By default, auto-terminate is set to **No**. This keeps the cluster running until you terminate it.  

When set to **Yes**, the cluster is automatically terminated after the last step is completed.  

For more information about submitting work to a cluster, see [Submit Work to a Cluster](p. 493). |

11. Choose **Create cluster**.  

12. On the **Cluster Details** page, proceed to the next step to run the Hive script as a cluster step and to use the Hue web interface to query your data.
Step 4: Run the Hive Script as a Step

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Before you query your data in the Hue web interface, you run a Hive script that loads the sample data into a Hive table and submits a query that writes output to Amazon S3. When the script is complete, you examine the output data.

You use the Amazon EMR console to run the Hive script as a step. A step is a unit of work containing one or more Hadoop jobs. The script reads and queries a series of sample CloudFront logs stored in Amazon S3.

You can submit steps to a cluster when the cluster is created or while it is running (if it is a long-running cluster). In this tutorial, you use the console to submit your Hive step to a running cluster. You can also submit steps using the AWS CLI, API, or SDK.

This model—submitting work programmatically to a cluster using steps—is useful in situations where you have automated batch processes or where you need to process data for later analysis. For example, you may use a Hive step to load data into a data warehouse for later analysis and visualization.

### Hive Script Overview

The Hive script used in this tutorial reads a series of CloudFront web distribution log files. Each entry in the log files gives details about a single user request in the following format:

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

For more information about CloudFront logs, see the Amazon CloudFront Developer Guide.

The Hive script you run in this tutorial uses the regular expression serializer/deserializer (or RegEx SerDe) to parse the CloudFront log input and write it to an external Hive table named `cloudfront_logs`. The Hive code that creates the table looks like the following:

```sql
CREATE EXTERNAL TABLE IF NOT EXISTS cloudfront_logs (  
  Date Date,  
  Time STRING,  
  Location STRING,  
  Bytes INT,  
  RequestIP STRING,  
  Method STRING,  
  Host STRING,  
  Uri STRING,  
  Status INT,  
  Referrer STRING,  
  OS String,  
  Browser String,  
  BrowserVersion String )
```

The Hive code that parses the log files using the RegEx SerDe looks like the following:

```sql
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'  
```
After creating the table, the Hive script submits a HiveQL query against the data to retrieve the total requests per operating system for a given time frame:

```
SELECT os, COUNT(*) count FROM cloudfront_logs WHERE date BETWEEN '2014-07-05' AND '2014-08-05' GROUP BY os;
```

The query results are written to the Amazon S3 bucket that you previously created.

## Submit the Hive Script as a Step

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

**Note**

You must complete **Step 2: Create an Amazon S3 Bucket for Your Cluster Logs and Output Data** (p. 15) before you run the script.

**To submit your Hive script as a step**

1. Open the Amazon EMR console at [https://console.aws.amazon.com/elasticmapreduce/](https://console.aws.amazon.com/elasticmapreduce/).
2. For **Cluster List**, select the name of your cluster.
3. Scroll to the **Steps** section and expand it, then choose **Add step**.
4. In the **Add Step** dialog box:
   - For **Step type**, choose **Hive program**
   - For **Name**, accept the default name (Hive program) or type a new name
   - For **Script S3 location**, type `s3://[yourregion].elasticmapreduce.samples/cloudfront/code/Hive_CloudFront.q`
     Replace `[yourregion]` (including the brackets) with your region, for example, `s3://us-west-2.elasticmapreduce.samples/cloudfront/code/Hive_CloudFront.q`
   - For **Input S3 location**, type `s3://[yourregion].elasticmapreduce.samples`
     Replace `[yourregion]` (including the brackets) with your region, for example, `s3://us-west-2.elasticmapreduce.samples`
   - For **Output S3 location**, type or browse to `s3://myemrbucket/output` (or the name of the bucket you created previously)
   - For **Arguments**, leave the field blank
   - For **Action on failure**, accept the default option (Continue)
5. Choose **Add**. The step appears in the console with a status of Pending.
6. The status of the step changes from Pending to Running to Completed as the step runs. To update the status, choose **Refresh** above the **Actions** column. The step runs in approximately 1 minute.

## View the Results

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

**To view the output of the Hive script**

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

Step 5: Query Your Data Using Hue

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

After you run the Hive script that creates your table and loads the data, log into the Hue web interface and submit an interactive query against the data. Hue is an open source web user interface for Hadoop that allows technical and non-technical users to take advantage of Hive, Pig, and many of the other tools that are part of the Hadoop and Amazon EMR ecosystem. Using Hue gives data analysts and data scientists a simple way to query data and create scripts interactively.

Before logging into Hue, create an SSH tunnel to the Amazon EMR master node.

Create an SSH Tunnel to the Master Node

To connect to Hue and run the script, you must connect to the master node via SSH and establish a tunnel to the Hue interface running on port 8888. Creating the SSH connection requires:

- An SSH client such as PuTTY (Windows) or OpenSSH (Linux, Mac OS X)
- An Amazon EC2 key pair private key file (.ppk for Windows or .pem for Linux and Mac OS X)

Your Linux computer most likely includes an SSH client by default. For example, OpenSSH is installed on most Linux, Unix, and Mac OS X operating systems. You can check for an SSH client by typing ssh at a shell command line. If your computer doesn't recognize the command, you must install an SSH client to connect to the master node. The OpenSSH project provides a free implementation of the full suite of SSH tools. For more information, go to http://www.openssh.org.

For more information about creating an SSH tunnel, see Connect to the Master Node Using SSH (p. 457).

To create an SSH tunnel to the master node on Linux and Mac OS X using OpenSSH

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. In the console, on the Cluster List page, select a cluster.
3. On the Cluster Details page, note the Master public DNS value that appears at the top. This value is required to establish your SSH connection.
4. In a terminal window, type the following command to open an SSH tunnel on your local machine. This command accesses the Hue web interface by forwarding traffic on local port 8157 (a randomly chosen, unused local port) to port 8888 on the master node. In the command, replace ~/mykeypair.pem with the location and file name of your .pem file and replace ec2-###-##-##-###.compute-1.amazonaws.com with the master public DNS name of your cluster.

```
ssh -i ~/mykeypair.pem -N -L 8157:ec2-###-##-##-###.compute-1.amazonaws.com:8888
hadoop@ec2-###-##-##-###.compute-1.amazonaws.com
```
After you issue this command, the terminal remains open and does not return a response.

**Note**

-L signifies the use of local port forwarding which allows you to specify a local port used to forward data to the identified remote port on the master node's local web server.

**To set up an SSH tunnel to the master node on Windows using PuTTY**

Windows users can use an SSH client such as PuTTY to connect to the master node. Before connecting to the Amazon EMR master node, you should download and install PuTTY and PuTTYgen. You can download these tools from the PuTTY download page.

PuTTY does not natively support the key pair private key file format (.pem) generated by Amazon EC2. You use PuTTYgen to convert your key file to the required PuTTY format (.ppk). You must convert your key into this format (.ppk) before attempting to connect to the master node using PuTTY.

For more information about converting your key, see Converting Your Private Key Using PuTTYgen in the Amazon EC2 User Guide for Linux Instances.

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. In the console, on the Cluster List page, select a cluster.
3. On the Cluster Details page, note the Master public DNS value that appears. This value is required to establish your SSH connection.
4. Open putty.exe to start PuTTY. You can also launch PuTTY from the Windows programs list.
5. If necessary, in the Category list, choose Session.
6. For Host Name (or IP address), type hadoop@MasterPublicDNS. For example: hadoop@ec2-###-##-##-###.compute-1.amazonaws.com.
7. In the Category list, expand Connection, SSH and then choose Auth.
8. For Private key file for authentication, choose Browse and select the .ppk file that you generated.
9. For Category, choose Connection, SSH, and Tunnels.
10. For Source port, type an unused local port number, for example 8157.
11. For Destination, type MasterPublicDNS:8888 to access the Hue interface, for example ec2-###-##-##-##-###.compute-1.amazonaws.com:8888.
12. Leave the Local and Auto options selected.
13. Choose Add. You should see an entry in the Forwarded ports field similar to: L8157 ec2-###-##-##-##-###.compute-1.amazonaws.com:8888.
14. Choose Open and Yes to dismiss the PuTTY security alert.

**Important**

When you log into the master node and are prompted for a user name, type hadoop.

**Log into Hue and Submit an Interactive Hive Query**

After configuring your SSH tunnel to the Amazon EMR master node, log into Hue and run the Hive script.

**To run the Hive script in Hue**

1. Type the following URL in your browser: http://localhost:8157.
2. At the Hue welcome page, type a Username and Password. The name and password used the first time you log into Hue become the Hue superuser credentials.
Note
The password must be at least 8 characters long, and must contain both uppercase and lowercase letters, at least one number, and at least one special character.

3. At the Did you know? dialog, choose Got it, prof! When the My Documents page opens, the sample projects are displayed.
4. From the menu options, choose Query Editors > Hive.
5. Delete the sample text and type:

```
SELECT browser, COUNT(*) count FROM cloudfront_logs WHERE date BETWEEN '2014-07-05' AND '2014-08-05' GROUP BY browser;
```

This HiveQL query retrieves the total requests per browser for a given time frame.
6. Choose Execute. As the query runs, log entries are displayed on the Log tab in the window below. When the query completes, the Results tab is displayed.
7. Review the output data from the query.
8. After examining the output, close your browser, or as time permits, continue to explore Hue.

(Optional) Step 6: Explore Amazon EMR

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

If you finish this tutorial in less than an hour (or if you choose to incur extra charges by going beyond an hour), consider completing the following self-directed activities in the Amazon EMR Developer Guide:

- Examine the CloudWatch metrics for your cluster – Monitor Metrics with CloudWatch
- Turn off termination protection for your cluster – Managing Cluster Termination
- Resize your cluster by adding one or more core nodes (note that adding nodes to your cluster will increase the cost of this tutorial) – Resize a Running Cluster
- Examine your cluster logs in Amazon S3 – View Log Files

If you decide to explore at another time, terminate your cluster to avoid extra charges. For more information, see (Optional) Step 7: Remove the Resources Used in the Tutorial (p. 25).

The following Amazon EMR tutorials are available to you in the Amazon EMR Developer Guide. These tutorials may require you to launch another cluster and incur additional costs.

- Query data using Impala: Tutorial: Launching and Querying Impala Clusters on Amazon EMR
- Analyze Kinesis streams: Tutorial: Analyzing Amazon Kinesis Streams with Amazon EMR and Hive and Tutorial: Analyzing Amazon Kinesis Streams with Amazon EMR and Pig
- Analyze Elastic Load Balancing log data: Tutorial: Query Elastic Load Balancing Access Logs with Amazon Elastic MapReduce

(Optional) Step 7: Remove the Resources Used in the Tutorial

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.
After you have finished the tutorial, you should remove the resources you created to avoid additional charges: your Amazon S3 bucket and your long-running Amazon EMR cluster. Terminating your cluster terminates the associated EC2 instances and stops the accrual of Amazon EMR charges. Amazon EMR preserves metadata information about completed clusters for your reference, at no charge, for two months. The console does not provide a way to delete completed clusters from the console; these are automatically removed for you after two months.

**To delete your Amazon S3 bucket**

If you used an existing bucket or did not create a bucket, this step is optional.

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. To delete a bucket, you must first delete all of the objects in it. Select the bucket that stores your logs and output data.
3. Use the **SHIFT** or **CRTL** keys to select all the objects in the bucket.
   
   **Tip**
   
   You can use the **SHIFT** and **CRTL** keys to select multiple objects and perform the same action on them simultaneously.
4. Open the context menu (right-click) for the objects and choose **Delete** or choose **Actions, Delete**.
5. Choose **OK** to confirm the deletion when prompted.
6. Choose **All Buckets** in the breadcrumb trail at the top of the window.
7. Open the context menu (right-click) for the bucket and choose **Delete, OK**.

**To terminate your Amazon EMR cluster**

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. On the **Cluster List** page, select your cluster by selecting the check box and choose **Terminate**.
3. By default, clusters created using the console are launched with termination protection enabled. In the **Terminate clusters** dialog box, for **Termination protection**, choose **Change**.
4. Choose **Off** and then select the check mark to confirm the change.
5. Choose **Terminate**.
Plan and Configure Clusters

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

- What region to run a cluster in, where and how to store data, and how to output results. See Configure Cluster Location and Data Storage (p. 27).
- Whether a cluster is long-running or transient, and what software it runs. See Configure a Cluster to be Transient or Long-Running (p. 67) and Configure Cluster Software (p. 68).
- The hardware and networking options that optimize cost, performance, and availability for your application. See Configure Cluster Hardware and Networking (p. 139).
- How to set up clusters so you can manage them more easily, and monitor activity, performance, and health. See Configure Cluster Logging and Debugging (p. 167) and Tag Clusters (p. 171).
- How to authenticate and authorize access to cluster resources, and how to encrypt data. See Security (p. 187).
- How to integrate with other software and services. See Drivers and Third-Party Application Integration (p. 176).

Configure Cluster Location and Data Storage

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

Topics
- Choose an AWS Region (p. 27)
- Work with Storage and File Systems (p. 29)
- Prepare Input Data (p. 31)
- Configure an Output Location (p. 40)
Choose an AWS Region

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

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

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

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

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

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

Choose a Region Using the Console

Your default region is displayed automatically.

To change regions using the console

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

Specify a Region Using the AWS CLI

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

Choose a Region Using an SDK or the API

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

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

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

**Work with Storage and File Systems**

This documentation is for AMI versions 2.x and 3.x of 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://myawsbucket/path` references an Amazon S3 bucket using EMRFS. The following table lists the available file systems, with recommendations about when it's best to use each one.

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

<table>
<thead>
<tr>
<th>File System</th>
<th>Prefix</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDFS</td>
<td>hdfs:// (or no prefix)</td>
<td>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.</td>
</tr>
<tr>
<td>EMRFS</td>
<td>s3://</td>
<td>EMRFS is an implementation of HDFS used for reading and writing regular files from Amazon EMR directly to Amazon S3. EMRFS provides the convenience of storing persistent data in Amazon S3 for use with Hadoop while also providing features like Amazon S3 server-side encryption, read-after-write consistency, and list consistency.</td>
</tr>
<tr>
<td>local file system</td>
<td></td>
<td>The local file system refers to a locally connected disk. When a Hadoop cluster is created, each node is created from an EC2 instance that comes with a preconfigured block of preattached disk storage called an instance store. Data on instance store volumes persists only during the life of its EC2 instance. Instance store volumes are ideal for storing temporary data that is continually changing, such as buffers, caches, scratch data, and other temporary content. For more information, see Amazon EC2 Instance Storage.</td>
</tr>
</tbody>
</table>

Note

Previously, Amazon EMR used the S3 Native FileSystem with the URI scheme, s3n. While this still works, we recommend that you use the s3 URI scheme for the best performance, security, and reliability.
<table>
<thead>
<tr>
<th>File System</th>
<th>Prefix</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Legacy) Amazon S3 block file system</td>
<td>s3bfs://</td>
<td>The Amazon S3 block file system is a legacy file storage system. We strongly discourage the use of this system.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>

**Access File Systems**

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

**To access a local HDFS**

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

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

**To access a remote HDFS**

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

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

**To access Amazon S3**

- Use the `s3://` prefix.

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

**To access the Amazon S3 block file system**

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

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

**Warning**

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

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

---

### Prepare Input Data

This documentation is for AMI versions 2.x and 3.x of 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 (p. 31)
- How to Get Data Into Amazon EMR (p. 31)

---

### Types of Input Amazon EMR Can Accept

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

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](http://hadoop.apache.org/docs/current/api/org/apache/hadoop/mapred/InputFormat.html).

If you need to analyze data stored in a legacy format, such as PDF and Word files, you can use Informatica's HParser to convert the data to text or XML format. For more information, see [Parse Data with HParser](p. 176).

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](https://cwiki.apache.org/confluence/display/Hive/SerDe).

---

### How to Get Data Into Amazon EMR

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.
Amazon EMR provides several ways to get data onto a cluster. The most common way is to upload the data to Amazon S3 and use the built-in features of Amazon EMR to load the data onto your cluster. You can also use the Distributed Cache feature of Hadoop to transfer files from a distributed file system to the local file system. The implementation of Hive provided by Amazon EMR (Hive version 0.7.1.1 and later) includes functionality that you can use to import and export data between DynamoDB and an Amazon EMR cluster. If you have large amounts of on-premises data to process, you may find the AWS Direct Connect service useful.

**Topics**
- Upload Data to Amazon S3 (p. 32)
- Import files with Distributed Cache (p. 35)
- How to Process Compressed Files (p. 39)
- Import DynamoDB Data into Hive (p. 40)
- Connect to Data with AWS DirectConnect (p. 40)
- Upload Large Amounts of Data with AWS Import/Export (p. 40)

### Upload Data to Amazon S3

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

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

**Topics**
- Create and Configure an Amazon S3 Bucket (p. 32)
- Best Practices (p. 33)
- Configure Multipart Upload for Amazon S3 (p. 33)

### Create and Configure an Amazon S3 Bucket

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

This section shows you how to use the Amazon S3 AWS Management Console to create and then set permissions for an Amazon S3 bucket. You can also create and set permissions for an Amazon S3 bucket using the Amazon S3 API or AWS CLI. You can also use Curl along with a modification to pass the appropriate authentication parameters for Amazon S3.

See the following resources:

- To create a bucket using the console, see Create a Bucket in the Amazon Simple Storage Service Console User Guide.
- To create and work with buckets using the AWS CLI, see Using High-Level S3 Commands with the AWS Command Line Interface in the Amazon Simple Storage Service Console User Guide.
- To create a bucket using an SDK, see Examples of Creating a Bucket in the Amazon Simple Storage Service Developer Guide.
- To work with buckets using Curl, see Amazon S3 Authentication Tool for Curl.
- For more information on specifying Region-specific buckets, see Accessing a Bucket in the Amazon Simple Storage Service Developer Guide.
**Note**
If you enable logging for a bucket, it enables only bucket access logs, not Amazon EMR cluster logs.

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

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

**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, go to Using Versioning in the Amazon Simple Storage Service Developer Guide.

**Consider Lifecycle Management**

Lifecycle Management in Amazon S3 allows you to create rules which control the storage class and lifetime of your objects. For more information, see Object Lifecycle Management.

**Clean Up Failed Multipart Uploads and Version Markers**

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 this configuration, see Configure Multipart Upload for Amazon S3 (p. 33). Amazon EMR does not automatically manage incomplete multipart uploads. Sometimes the upload of a large file can result in an incomplete Amazon S3 multipart upload. When a multipart upload is unable to complete successfully, the in-progress multipart upload continues to occupy your bucket and incurs storage charges. We recommend that for buckets you use with Amazon EMR, you use a lifecycle rule to remove incomplete multipart uploads three days after the upload initiation date. For more information, see Aborting Incomplete Multipart Uploads Using a Bucket Lifecycle Policy.

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

**Performance best practices**

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

**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 on Amazon S3 multipart uploads, go to Uploading Objects Using Multipart Upload in the Amazon Simple Storage Service Developer Guide.
The Amazon EMR configuration parameters for multipart upload are described in the following table.

<table>
<thead>
<tr>
<th>Configuration Parameter Name</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fs.[s3</td>
<td>s3n].multipart.uploads.enabled</td>
<td>True</td>
</tr>
<tr>
<td>fs.s3n.ssl.enabled</td>
<td>True</td>
<td>A boolean type that indicates whether to use http or https.</td>
</tr>
<tr>
<td>fs.s3.buckets.create.enabled</td>
<td>True</td>
<td>This setting automatically creates a bucket if it doesn't exist. Setting to False will cause an exception on CreateBucket operations for that case.</td>
</tr>
</tbody>
</table>

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

**Disable Multipart Upload Using the Amazon EMR Console**

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

**To disable multipart uploads with a bootstrap action using the console**

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. Choose **Create cluster**.
3. In the **Bootstrap Actions** section, in the **Add bootstrap action** field, select **Configure Hadoop** and click **Configure and add**.

   Enter the following information:

   a. In **Optional arguments**, replace the default value with the following:

      ```
      -c fs.s3n.multipart.uploads.enabled=false
      ```

   b. Click **Add**.

   For more information, see (Optional) Create Bootstrap Actions to Install Additional Software (p. 129).
4. Proceed with creating the cluster as described in Plan and Configure Clusters (p. 27).

**Disable Multipart Upload Using the AWS CLI**

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

**To disable multipart upload using the AWS CLI**

- To disable multipart upload, type the following command and replace `myKey` with the name of your EC2 key pair.

- Linux, UNIX, and Mac OS X users:

  ```bash
  aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applications Name=Hue Name=Hive Name=Pig \ 
  --use-default-roles --ec2-attributes KeyName=myKey \ 
  --instance-type m3.xlarge --instance-count 3 \
  ```
When you specify the instance count without using the --instance-groups parameter, a single Master node is launched, and the remaining instances are launched as core nodes. All nodes will use the instance type specified in the command.

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

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

### Disable Multipart Upload Using the API

For information on using Amazon S3 multipart uploads programmatically, go to [Using the AWS SDK for Java for Multipart Upload](http://docs.aws.amazon.com/s3/latest/userguide/uploading-multipart-java-sdk.html) in the *Amazon Simple Storage Service Developer Guide*.

For more information about the AWS SDK for Java, go to the [AWS SDK for Java](https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/index.html) detail page.

### Import files with Distributed Cache

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

**Topics**

- Supported File Types (p. 36)
- Location of Cached Files (p. 36)
- Access Cached Files From Streaming Applications (p. 36)
- Access Cached Files From Streaming Applications Using the Amazon EMR Console (p. 36)
- Access Cached Files From Streaming Applications Using the AWS CLI (p. 37)

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


You invoke Distributed Cache when you create the cluster. The files are cached just before starting the Hadoop job and the files remain cached for the duration of the job. You can cache files stored on any Hadoop-compatible file system, for example HDFS or Amazon S3. The default size of the file cache is 10GB. To change the size of the cache, reconfigure the Hadoop parameter, `local.cache.size` using...
the bootstrap action. For more information, see (Optional) Create Bootstrap Actions to Install Additional Software (p. 129).

**Supported File Types**

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

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

- zip
- tgz
- tar.gz
- tar
- jar

**Location of Cached Files**

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

Distributed Cache associates the cache files to the current working directory of the mapper and reducer using symlinks. A symlink is an alias to a file location, not the actual file location. The value of the parameter, mapred.local.dir, specifies the location of temporary files. Amazon EMR sets this parameter to /mnt/var/lib/hadoop/mapred/, /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/var/lib/hadoop/mapred/taskTracker/archive/.

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

You can use Distributed Cache only when using Streaming.

**Access Cached Files From Streaming Applications**

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

**Access Cached Files From Streaming Applications Using the Amazon EMR Console**

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

**To specify Distributed Cache files using the console**

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

   The size of the file (or total size of the files in an archive file) must be less than the allocated cache size.
<table>
<thead>
<tr>
<th>If you want to ...</th>
<th>Action</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add an individual file to the Distributed Cache</td>
<td>Specify <code>-cacheFile</code> followed by the name and location of the file, the pound (#) sign, and then the name you want to give the file when it’s placed in the local cache.</td>
<td><code>-cacheFile \s3://bucket_name/file_name#cache_file_name</code></td>
</tr>
<tr>
<td>Add an archive file to the Distributed Cache</td>
<td>Enter <code>-cacheArchive</code> followed by the location of the files in Amazon S3, the pound (#) sign, and then the name you want to give the collection of files in the local cache.</td>
<td><code>-cacheArchive \s3://bucket_name/archive_name#cache_archive_name</code></td>
</tr>
</tbody>
</table>

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

**Access Cached Files From Streaming Applications Using the AWS CLI**

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

**To specify Distributed Cache files using the AWS CLI**

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

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

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

**Example 1**

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

The Hadoop streaming syntax is different between Hadoop 1.x and Hadoop 2.x.
For Hadoop 2.x, type the following command and replace myKey with the name of your EC2 key pair.

- Linux, UNIX, and Mac OS X users:

```bash
aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applications Name=Hue Name=Hive Name=Pig \
--use-default-roles --ec2-attributes KeyName=myKey \
--instance-type m3.xlarge --instance-count 3 \
--steps Type=STREAMING,Name="Streaming program",ActionOnFailure=CONTINUE,\ 
Args="--files","s3://my_bucket/my_mapper.py s3://my_bucket/my_reducer.py","-mapper","my_mapper.py","-reducer","my_reducer.py","-input","s3://my_bucket/my_input","-output","s3://my_bucket/my_output", "-cacheFile","s3://my_bucket/sample_dataset.dat#sample_dataset_cached.dat"
```

- Windows users:

```bash
aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applications Name=Hue Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey --instance-type m3.xlarge --instance-count 3 --steps Type=STREAMING,Name="Streaming program",ActionOnFailure=CONTINUE,Args="--files","s3://my_bucket/my_mapper.py s3://my_bucket/my_reducer.py","-mapper","my_mapper.py","-reducer","my_reducer.py","-input","s3://my_bucket/my_input","-output","s3://my_bucket/my_output", "-cacheFile","s3://my_bucket/sample_dataset.dat#sample_dataset_cached.dat"
```

For Hadoop 1.x, use the following command and replace myKey with the name of your EC2 key pair.

- Linux, UNIX, and Mac OS X users:

```bash
aws emr create-cluster --name "Test cluster" --ami-version 2.4 --applications Name=Hue Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey --instance-type m3.xlarge --instance-count 3 --steps Type=STREAMING,Name="Streaming program",ActionOnFailure=CONTINUE,Args="-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"
```

- Windows users:

```bash
aws emr create-cluster --name "Test cluster" --ami-version 2.4 --applications Name=Hue Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey --instance-type m3.xlarge --instance-count 3 --steps Type=STREAMING,Name="Streaming program",ActionOnFailure=CONTINUE,Args="-mapper","my_mapper.py","-reducer","my_reducer.py","-input","s3://my_bucket/my_input","-output","s3://my_bucket/my_output", "-cacheFile","s3://my_bucket/sample_dataset.dat#sample_dataset_cached.dat"
```

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

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

**Example 2**

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

The Hadoop streaming syntax is different between Hadoop 1.x and Hadoop 2.x.
For Hadoop 2.x, use the following command and replace `myKey` with the name of your EC2 key pair.

- **Linux, UNIX, and Mac OS X users:**

```bash
aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applications Name=Hue Name=Hive Name=Pig \ --use-default-roles --ec2-attributes KeyName=myKey \ --instance-type m3.xlarge --instance-count 3 \ --steps Type=STREAMING,Name="Streaming program",ActionOnFailure=CONTINUE,\ Args="--files","s3://my_bucket/my_mapper.py s3://my_bucket/my_reducer.py","-mapper","my_mapper.py","-reducer","my_reducer.py","-input","s3://my_bucket/my_input","-output","s3://my_bucket/my_output","-cacheArchive","s3://my_bucket/sample_dataset.tgz#sample_dataset_cached"
```

- **Windows users:**

```bash
aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applications Name=Hue Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey -- instance-type m3.xlarge --instance-count 3 --steps Type=STREAMING,Name="Streaming program",ActionOnFailure=CONTINUE,Args="--files","s3://my_bucket/my_mapper.py s3://my_bucket/my_reducer.py","-mapper","my_mapper.py","-reducer","my_reducer.py","-input","s3://my_bucket/my_input","-output","s3://my_bucket/my_output","-cacheArchive","s3://my_bucket/sample_dataset.tgz#sample_dataset_cached"
```

For Hadoop 1.x, use the following command and replace `myKey` with the name of your EC2 key pair.

- **Linux, UNIX, and Mac OS X users:**

```bash
aws emr create-cluster --name "Test cluster" --ami-version 2.4 --applications Name=Hue Name=Hive Name=Pig \ --use-default-roles --ec2-attributes KeyName=myKey \ --instance-type m3.xlarge --instance-count 3 \ --steps Type=STREAMING,Name="Streaming program",ActionOnFailure=CONTINUE,\ Args="-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"
```

- **Windows users:**

```bash
aws emr create-cluster --name "Test cluster" --ami-version 2.4 --applications Name=Hue Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey -- instance-type m3.xlarge --instance-count 3 --steps Type=STREAMING,Name="Streaming program",ActionOnFailure=CONTINUE,Args="-mapper","my_mapper.py","-reducer","my_reducer.py","-input","s3://my_bucket/my_input","-output","s3://my_bucket/my_output","-cacheArchive","s3://my_bucket/sample_dataset.tgz#sample_dataset_cached"
```

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

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

**How to Process Compressed Files**

This documentation is for AMI versions 2.x and 3.x of 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**

This documentation is for AMI versions 2.x and 3.x of 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 (p. 393).

**Connect to Data with AWS DirectConnect**

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

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

**Upload Large Amounts of Data with AWS Import/Export**

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

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

**Configure an Output Location**

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

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

For more information, see the following topics:

**Topics**
- Create and Configure an Amazon S3 Bucket (p. 41)
Create and Configure an Amazon S3 Bucket

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

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

**To create an Amazon S3 bucket using the console**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https:// console.aws.amazon.com/s3/.
2. Choose **Create Bucket**.
   
   The **Create a Bucket** dialog box opens.
3. Enter a bucket name, such as **myawsbucket**.
   
   This name should be globally unique, and cannot be the same name used by another bucket.
4. Select the **Region** for your bucket. To avoid paying cross-region bandwidth charges, create the Amazon S3 bucket in the same region as your cluster.
   
   Refer to **Choose an AWS Region (p. 27)** for guidance on choosing a Region.
5. Choose **Create**.

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

**Note**

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

**Note**

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

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

**To set permissions on an Amazon S3 bucket using the console**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https:// console.aws.amazon.com/s3/.
2. In the **Buckets** pane, open (right-click) the bucket you just created.
3. Select **Properties**.
4. In the **Properties** pane, select the **Permissions** tab.
5. Choose **Add more permissions**.
6. Select **Authenticated Users** in the **Grantee** field.
Configure an Output Location

7. To the right of the Grantee drop-down list, select List.
8. Choose Save.

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

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

<table>
<thead>
<tr>
<th>Information</th>
<th>Example Location on Amazon S3</th>
</tr>
</thead>
<tbody>
<tr>
<td>script or program</td>
<td>s3://myawsbucket/script/MapperScript.py</td>
</tr>
<tr>
<td>log files</td>
<td>s3://myawsbucket/logs</td>
</tr>
<tr>
<td>input data</td>
<td>s3://myawsbucket/input</td>
</tr>
<tr>
<td>output data</td>
<td>s3://myawsbucket/output</td>
</tr>
</tbody>
</table>

**What formats can Amazon EMR return?**

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

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**

This documentation is for AMI versions 2.x and 3.x of 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 Console User Guide.

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

<table>
<thead>
<tr>
<th>Canned ACL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AuthenticatedRead</td>
<td>Specifies that the owner is granted Permission.FullControl and the GroupGrantee.AuthenticatedUsers group grantee is granted Permission.Read access.</td>
</tr>
<tr>
<td>BucketOwnerFullControl</td>
<td>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.</td>
</tr>
<tr>
<td>BucketOwnerRead</td>
<td>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.</td>
</tr>
<tr>
<td>LogDeliveryWrite</td>
<td>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.</td>
</tr>
<tr>
<td>Private</td>
<td>Specifies that the owner is granted Permission.FullControl.</td>
</tr>
<tr>
<td>PublicRead</td>
<td>Specifies that the owner is granted Permission.FullControl and the GroupGrantee.AllUsers group grantee is granted Permission.Read access.</td>
</tr>
<tr>
<td>PublicReadWrite</td>
<td>Specifies that the owner is granted Permission.FullControl and the GroupGrantee.AllUsers group grantee is granted Permission.Read and Permission.Write access.</td>
</tr>
</tbody>
</table>

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

**To write files using canned ACLs in Hive**

- From the Hive command prompt, set the `fs.s3.canned.acl` configuration option to the canned ACL you want to have the cluster set on files it writes to Amazon S3. To access the Hive command prompt connect to the master node using SSH, and type Hive at the Hadoop command prompt. For more information, see Connect to the Master Node Using SSH (p. 457).

  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.
To write files using canned ACLs in Pig

- From the Pig command prompt, set the `fs.s3.canned.acl` configuration option to the canned ACL you want to have the cluster set on files it writes to Amazon S3. To access the Pig command prompt connect to the master node using SSH, and type Pig at the Hadoop command prompt. For more information, see Connect to the Master Node Using SSH (p. 457).

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

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

To write files using canned ACLs in a custom JAR

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

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

Compress the Output of your Cluster

Topics

- Output Data Compression (p. 44)
- Intermediate Data Compression (p. 45)
- Using the Snappy Library with Amazon EMR (p. 45)

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

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

When writing a Custom Jar, use the following command:

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

**Using the Snappy Library with Amazon EMR**

Snappy is a compression and decompression library that is optimized for speed. It is available on Amazon EMR AMIs version 2.0 and later and is used as the default for intermediate compression. For more information about Snappy, go to [http://code.google.com/p/snappy/](http://code.google.com/p/snappy/). For more information about Amazon EMR AMI versions, go to [Choose an Amazon Machine Image (AMI)](p. 69).

**Using EMR File System (EMRFS)**

The EMR File System (EMRFS) is an implementation of HDFS that all Amazon EMR clusters use for reading and writing regular files from Amazon EMR directly to Amazon S3. EMRFS provides the convenience of storing persistent data in Amazon S3 for use with Hadoop while also providing features like consistent view and data encryption.

Consistent view provides consistency checking for list and read-after-write (for new put requests) for objects in Amazon S3. Data encryption allows you to encrypt objects that EMRFS writes to Amazon S3, and enables EMRFS to work with encrypted objects in Amazon S3. If you are using Amazon EMR release version 4.8.0 or later, you can use security configurations to set up encryption for EMRFS objects in Amazon S3, along with other encryption settings. For more information, see [Understanding Encryption Options](p. 207). If you use an earlier release version of Amazon EMR, you can manually configure encryption settings. For more information, see [Specifying Amazon S3 Encryption Using EMRFS Properties](p. 61).

When using Amazon EMR release version 5.10.0 or later, you can use EMRFS authorization for Amazon S3 to control access to EMRFS objects in Amazon S3 based on user, group, or the location of EMRFS data in Amazon S3. For more information, see [EMRFS Authorization for Data in Amazon S3](p. 214).
Consistent View

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

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

With consistent view enabled, EMRFS returns the set of objects listed in an EMRFS metadata store and those returned directly by Amazon S3 for a given path. Because Amazon S3 is still the “source of truth” for the objects in a path, EMRFS ensures that everything in a specified Amazon S3 path is being processed regardless of whether it is tracked in the metadata. However, EMRFS consistent view only ensures that the objects in the folders that you track are checked for consistency.

You can use the EMRFS utility (emrfs) from the command line of the master node to perform operations on Amazon S3 objects that are tracked by consistent view. For example, you can import, delete, and sync Amazon S3 objects with the EMRFS metadata store. For more information about the EMRFS CLI utility, see EMRFS CLI Reference (p. 54).

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

Topics

• Enable Consistent View (p. 46)
• Understanding How EMRFS Consistent View Tracks Objects in Amazon S3 (p. 47)
• Retry Logic (p. 48)
• EMRFS Consistent View Metadata (p. 49)
• Configure Consistency Notifications for CloudWatch and Amazon SQS (p. 51)
• Configure Consistent View (p. 52)
• EMRFS CLI Reference (p. 54)

Enable Consistent View

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.
You can enable Amazon S3 server-side encryption or consistent view for EMRFS using the AWS Management Console, AWS CLI, or you can use a bootstrap action to configure additional settings for EMRFS.

To configure consistent view using the console:

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. Choose Create cluster, Go to advanced options.
4. For Step 3: General Cluster Settings, under Additional Options, choose EMRFS consistent view.
5. For EMRFS Metadata store, type the name of your metadata store. The default value is EmrFSMetadata. If the EmrFSMetadata table does not exist, it is created for you in DynamoDB.
   
   Note
   Amazon EMR does not automatically remove the EMRFS metadata from DynamoDB when the cluster is terminated.
6. For Number of retries, type an integer value. EMRFS tries to call Amazon S3 this number of times if an inconsistency is detected. The default value is 5.
7. For Retry period (in seconds), type an integer value. This is the amount of time that EMRFS waits between retry attempts. The default value is 10.
   
   Note
   Subsequent retries use an exponential backoff.

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

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

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

```bash
aws emr create-cluster --instance-type m1.large --instance-count 3 --emrfs Consistent=true --ami-version=3.9 --ec2-attributes KeyName=myKey
```

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

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

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

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

Understanding How EMRFS Consistent View Tracks Objects in Amazon S3:

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.
EMRFS creates a consistent view of objects in Amazon S3 by adding information about those objects to the EMRFS metadata. EMRFS adds these listings to its metadata when:

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

Objects read by EMRFS are not automatically added to the metadata. When EMRFS deletes an object, a listing still remains in the metadata with a deleted state until that listing is purged using the EMRFS CLI. To learn more about the CLI, see EMRFS CLI Reference (p. 54). For more information about purging listings in the EMRFS metadata, see EMRFS Consistent View Metadata (p. 49).

For every Amazon S3 operation, EMRFS checks the metadata for information about the set of objects in consistent view. If EMRFS finds that Amazon S3 is inconsistent during one of these operations, it retries the operation according to parameters defined in emrfs-site configuration properties. After EMRFS exhausts the retries, it either throws a ConsistencyException or logs the exception and continue the workflow. For more information about retry logic, see Retry Logic (p. 48). You can find ConsistencyExceptions in your logs, for example:

- listStatus: No s3 object for metadata item /S3_bucket/dir/object
- getFileStatus: Key dir/file is present in metadata but not s3

If you delete an object directly from Amazon S3 that EMRFS consistent view tracks, EMRFS treats that object as inconsistent because it is still listed in the metadata as present in Amazon S3. If your metadata becomes out of sync with the objects EMRFS tracks in Amazon S3, you can use the sync sub-command of the EMRFS CLI to reset metadata so that it reflects Amazon S3. To discover discrepancies between metadata and Amazon S3, use the diff. Finally, EMRFS only has a consistent view of the objects referenced in the metadata; there can be other objects in the same Amazon S3 path that are not being tracked. When EMRFS lists the objects in an Amazon S3 path, it will return the superset of the objects being tracked in the metadata and those in that Amazon S3 path.

**Retry Logic**

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

### Example Setting retry period to a fixed amount

```bash
aws emr create-cluster --ami-version 3.10.0 --instance-type m1.large --instance-count 1 \ --emrfs Consistent=true,Args=[fs.s3.contristent.throwExceptionOnInconsistency=false,\ fs.s3.contristent.retryPolicyType=fixed,fs.s3.contristent.retryPeriodSeconds=10] --ec2-attributes KeyName=myKey
```
**Note**

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

For more information, see Consistent View (p. 46).

**EMRFS Consistent View Metadata**

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

EMRFS consistent view tracks consistency using a DynamoDB table to track objects in Amazon S3 that have been synced with or created by EMRFS. The metadata is used to track all operations (read, write, update, and copy), and no actual content is stored in it. This metadata is used to validate whether the objects or metadata received from Amazon S3 matches what is expected. This confirmation gives EMRFS the ability to check list consistency and read-after-write consistency for new objects EMRFS writes to Amazon S3 or objects synced with EMRFS. Multiple clusters can share the same metadata.

**How to add entries to metadata**

You can use the `sync` or `import` subcommands to add entries to metadata. `sync` will simply reflect the state of the Amazon S3 objects in a path while `import` is used strictly to add new entries to the metadata. For more information, see EMRFS CLI Reference (p. 54).

**How to check differences between metadata and objects in Amazon S3**

To check for differences between the metadata and Amazon S3, use the `diff` subcommand of the EMRFS CLI. For more information, see EMRFS CLI Reference (p. 54).

**How to know if metadata operations are being throttled**

EMRFS sets default throughput capacity limits on the metadata for its read and write operations at 400 and 100 units, respectively. Large numbers of objects or buckets may cause operations to exceed this capacity, at which point they will be throttled by DynamoDB. For example, an application may cause EMRFS to throw a `ProvisionedThroughputExceededException` if you are performing an operation that exceeds these capacity limits. Upon throttling the EMRFS CLI tool will attempt to retry writing to the DynamoDB table using exponential backoff until the operation finishes or when it reaches the maximum retry value for writing objects from EMR to Amazon S3.

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

**Throughput characteristics for notable EMRFS operations**

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

<table>
<thead>
<tr>
<th>Operation</th>
<th>Average read-per-second</th>
<th>Average write-per-second</th>
</tr>
</thead>
<tbody>
<tr>
<td>create (object)</td>
<td>26.79</td>
<td>6.70</td>
</tr>
<tr>
<td>Operation</td>
<td>Average read-per-second</td>
<td>Average write-per-second</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>delete (object)</td>
<td>10.79</td>
<td>10.79</td>
</tr>
<tr>
<td>delete (directory containing 1000 objects)</td>
<td>21.79</td>
<td>338.40</td>
</tr>
<tr>
<td>getFileStatus (object)</td>
<td>34.70</td>
<td>0</td>
</tr>
<tr>
<td>getFileStatus (directory)</td>
<td>19.96</td>
<td>0</td>
</tr>
<tr>
<td>listStatus (directory containing 1 object)</td>
<td>43.31</td>
<td>0</td>
</tr>
<tr>
<td>listStatus (directory containing 10 objects)</td>
<td>44.34</td>
<td>0</td>
</tr>
<tr>
<td>listStatus (directory containing 100 objects)</td>
<td>84.44</td>
<td>0</td>
</tr>
<tr>
<td>listStatus (directory containing 1,000 objects)</td>
<td>308.81</td>
<td>0</td>
</tr>
<tr>
<td>listStatus (directory containing 10,000 objects)</td>
<td>416.05</td>
<td>0</td>
</tr>
<tr>
<td>listStatus (directory containing 100,000 objects)</td>
<td>823.56</td>
<td>0</td>
</tr>
<tr>
<td>listStatus (directory containing 1M objects)</td>
<td>882.36</td>
<td>0</td>
</tr>
<tr>
<td>mkdir (continuous for 120 seconds)</td>
<td>24.18</td>
<td>4.03</td>
</tr>
<tr>
<td>mkdir</td>
<td>12.59</td>
<td>0</td>
</tr>
<tr>
<td>rename (object)</td>
<td>19.53</td>
<td>4.88</td>
</tr>
<tr>
<td>rename (directory containing 1000 objects)</td>
<td>23.22</td>
<td>339.34</td>
</tr>
</tbody>
</table>

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

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

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

```bash
aws emr add-steps --cluster-id j-2AL4XXXXXX5T9 --steps
  Type=CUSTOM_JAR,ActionOnFailure=CONTINUE,Jar=s3://elasticmapreduce/libs/script-runner/script-runner.jar,
  Args=[/home/hadoop/bin/emrfs,delete,--time,2,--time-unit,days]
{
  "StepIds": [
    "s-B12345678902"
}
Use the StepId value returned to check the logs for the result of the operation.

**Configure Consistency Notifications for CloudWatch and Amazon SQS**

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

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

**CloudWatch**

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

**To view CloudWatch metrics for Amazon S3 eventual consistency issues**

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

2. In the Dashboard, in the Metrics section, choose EMRFS.
3. In the Job Flow Metrics pane, select one or more JobFlowId/Metric Name pairs. A graphical representation of the metrics appears in the window below.

**Amazon SQS**

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

**To manage Amazon SQS messages for Amazon S3 eventual consistency issues**

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

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

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

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

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

   ```bash
   emrfs read-sqs -queue-name EMRFS-Inconsistency-j-162XXXXXXM2CU -output-file /path/filename
   ```
You can also use the -q and -o shortcuts instead of -queue-name and -output-file respectively.

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

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

For example, to delete the default queue, type:

```
emrfs delete-sqs -queue-name EMRFS-Inconsistency-j-162XXXXXXXm2CU
```

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

**Configure Consistent View**

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

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

**Example Changing default metadata read and write values at cluster launch**

```
aws emr create-cluster --ami-version 3.10.0 --instance-type m1.large --emrfs Consistent=true,Args=[fs.s3.consistent.metadata.read.capacity=600, fs.s3.consistent.metadata.write.capacity=300] --ec2-attributes KeyName=myKey
```

```
aws emr create-cluster --ami-version 3.10.0 --instance-type m1.large --bootstrap-actions Path=s3://us-east-1.elasticmapreduce/bootstrap-actions/configure-hadoop, Args=[-e,fs.s3.consistent=true,-e,fs.s3.consistent.metadata.read.capacity=600, -e,fs.s3.consistent.metadata.write.capacity=300] --ec2-attributes KeyName=myKey
```

Use the configuration you created with the following syntax:

**Note**

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

The following options can be set using bootstrap action or AWS CLI `--emrfs` arguments. For information about those arguments, see the AWS Command Line Interface Reference.

**emrfs-site.xml properties for consistent view**

<table>
<thead>
<tr>
<th>Property</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fs.s3.consistent</td>
<td>false</td>
<td>When set to true, this property configures EMRFS to use DynamoDB to provide consistency.</td>
</tr>
<tr>
<td>Property</td>
<td>Default value</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>fs.s3.consistent.retryPolicyType</code></td>
<td><code>exponential</code></td>
<td>This property identifies the policy to use when retrying for consistency issues. Options include: exponential, fixed, or none.</td>
</tr>
<tr>
<td><code>fs.s3.consistent.retryPeriodSeconds</code></td>
<td><code>10</code></td>
<td>This property sets the length of time to wait between consistency retry attempts.</td>
</tr>
<tr>
<td><code>fs.s3.consistent.retryCount</code></td>
<td><code>5</code></td>
<td>This property sets the maximum number of retries when inconsistency is detected.</td>
</tr>
<tr>
<td><code>fs.s3.consistent.throwExceptionOnInconsistency</code></td>
<td><code>true</code></td>
<td>This property determines whether to throw or log a consistency exception. When set to <code>true</code>, a ConsistencyException is thrown.</td>
</tr>
<tr>
<td><code>fs.s3.consistent.metadata.autoCreate</code></td>
<td><code>true</code></td>
<td>When set to <code>true</code>, this property enables automatic creation of metadata tables.</td>
</tr>
<tr>
<td><code>fs.s3.consistent.metadata.tableName</code></td>
<td><code>EmrFSMetadata</code></td>
<td>This property specifies the name of the metadata table in DynamoDB.</td>
</tr>
<tr>
<td><code>fs.s3.consistent.metadata.read.capacity</code></td>
<td><code>400</code></td>
<td>This property specifies the DynamoDB read capacity to provision when the metadata table is created.</td>
</tr>
<tr>
<td><code>fs.s3.consistent.metadata.write.capacity</code></td>
<td><code>100</code></td>
<td>This property specifies the DynamoDB write capacity to provision when the metadata table is created.</td>
</tr>
<tr>
<td><code>fs.s3.consistent.fastList</code></td>
<td><code>true</code></td>
<td>When set to <code>true</code>, this property uses multiple threads to list a directory (when necessary). Consistency must be enabled in order to use this property.</td>
</tr>
<tr>
<td><code>fs.s3.consistent.fastList.prefetchMetadata</code></td>
<td><code>false</code></td>
<td>When set to <code>true</code>, this property enables metadata prefetching for directories containing more than 20,000 items.</td>
</tr>
<tr>
<td><code>fs.s3.consistent.notification.CloudWatch</code></td>
<td><code>false</code></td>
<td>When set to <code>true</code>, CloudWatch metrics are enabled for FileSystem API calls that fail due to Amazon S3 eventual consistency issues.</td>
</tr>
<tr>
<td><code>fs.s3.consistent.notification.SQS</code></td>
<td><code>false</code></td>
<td>When set to <code>true</code>, eventual consistency notifications are pushed to an Amazon SQS queue.</td>
</tr>
</tbody>
</table>
### Property | Default value | Description
--- | --- | ---
`fs.s3.consistent.notification.SQS.queueName` | `EMRFS-Inconsistency<jobFlowId>` | Changing this property allows you to specify your own SQS queue name for messages regarding Amazon S3 eventual consistency issues.
`fs.s3.consistent.notification.SQS.customMsg` | `none` | This property allows you to specify custom information included in SQS messages regarding Amazon S3 eventual consistency issues. If a value is not specified for this property, the corresponding field in the message is empty.
`fs.s3.consistent.dynamodb.endpoint` | `none` | This property allows you to specify a custom DynamoDB endpoint for your consistent view metadata.

### EMRFS CLI Reference

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

The EMRFS CLI is installed by default on all cluster master nodes created using Amazon EMR release version 3.2.1 or later. You can use the EMRFS CLI to manage the metadata for consistent view.

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

### emrfs top-level command

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

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

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

#### [options] for emrfs

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-a AWS_ACCESS_KEY_ID</code></td>
<td>The AWS access key you use to write objects to Amazon S3 and to create or access a metadata store in DynamoDB. By default, <code>AWS_ACCESS_KEY_ID</code> is set to the access key used to create the cluster.</td>
<td>No</td>
</tr>
<tr>
<td><code>--access-key AWS_ACCESS_KEY_ID</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

**emrfs describe-metadata sub-command**

**[options] for emrfs describe-metadata**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>-m METADATA_NAME</td>
<td>--metadata-name METADATA_NAME</td>
<td>METADATA_NAME is the name of the DynamoDB metadata table. If the METADATA_NAME argument is not supplied, the default value is EmrFSMetadata.</td>
</tr>
</tbody>
</table>

**emrfs describe-metadata example**

The following example describes the default metadata table.

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

**emrfs set-metadata-capacity sub-command**

**[options] for emrfs set-metadata-capacity**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>-m METADATA_NAME</td>
<td>--metadata-name METADATA_NAME</td>
<td>METADATA_NAME is the name of the DynamoDB metadata table. If the METADATA_NAME argument is not supplied, the default value is EmrFSMetadata.</td>
</tr>
<tr>
<td>-r READ_CAPACITY</td>
<td>--read-capacity READ_CAPACITY</td>
<td>The requested read throughput capacity for the metadata table. If the READ_CAPACITY argument is not supplied, the default value is 400.</td>
</tr>
<tr>
<td>-w WRITE_CAPACITY</td>
<td>--write-capacity WRITE_CAPACITY</td>
<td>The requested write throughput capacity for the metadata table. If the WRITE_CAPACITY argument is not supplied, the default value is 100.</td>
</tr>
</tbody>
</table>

**emrfs set-metadata-capacity example**

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

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

### emrfs delete-metadata sub-command

**[options] for emrfs delete-metadata**

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

### emrfs delete-metadata example

The following example deletes the default metadata table.

```
$ emrfs delete-metadata
```

### emrfs create-metadata sub-command

**[options] for emrfs create-metadata**

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

### emrfs create-metadata example

The following example creates a metadata table named EmrFSMetadataAlt.

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

### emrfs list-metadata-stores sub-command

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

The following example lists your metadata tables.

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

emrfs diff sub-command

[options] for emrfs diff

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>`-m METADATA_NAME</td>
<td>--metadata-name METADATA_NAME</td>
<td>METADATA_NAME is the name of the DynamoDB metadata table. If the METADATA_NAME argument is not supplied, the default value is EmrFSMetadata.</td>
</tr>
<tr>
<td><code>s3://s3Path</code></td>
<td>The path to the Amazon S3 bucket you are tracking for consistent view that you wish to compare to the metadata table. Buckets sync recursively.</td>
<td>Yes</td>
</tr>
</tbody>
</table>

emrfs diff example

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

```bash
$ emrfs diff s3://elasticmapreduce/samples/cloudfront
```

emrfs delete sub-command

[options] for emrfs delete

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>`-m METADATA_NAME</td>
<td>--metadata-name METADATA_NAME</td>
<td>METADATA_NAME is the name of the DynamoDB metadata table. If the METADATA_NAME argument is not supplied, the default value is EmrFSMetadata.</td>
</tr>
<tr>
<td><code>s3://s3Path</code></td>
<td>The path to the Amazon S3 bucket you are tracking for consistent view. Buckets sync recursively.</td>
<td>Yes</td>
</tr>
<tr>
<td>`-t TIME</td>
<td>--time TIME`</td>
<td>The expiration time (interpreted using the time unit argument). All metadata entries older than the TIME argument are deleted for the specified bucket.</td>
</tr>
</tbody>
</table>
Consistent View

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>-u UNIT</td>
<td>--time-unit UNIT</td>
<td>The measure used to interpret the time argument (nanoseconds, microseconds, milliseconds, seconds, minutes, hours, or days). If no argument is specified, the default value is days.</td>
</tr>
<tr>
<td>--read-consumption READ_CONSUMPTION</td>
<td>The requested amount of available read throughput used for the delete operation. If the READ_CONSUMPTION argument is not specified, the default value is 400.</td>
<td>No</td>
</tr>
<tr>
<td>--write-consumption WRITE_CONSUMPTION</td>
<td>The requested amount of available write throughput used for the delete operation. If the WRITE_CONSUMPTION argument is not specified, the default value is 100.</td>
<td>No</td>
</tr>
</tbody>
</table>

emrfs delete example

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

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

emrfs import sub-command

[options] for emrfs import

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

emrfs import example

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

```
$ emrfs import s3://elasticmapreduce/samples/cloudfront
```
emrfs sync sub-command

[options] for emrfs sync

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>-m METADATA_NAME</td>
<td>--metadata-name METADATA_NAME</td>
<td>METADATA_NAME is the name of the DynamoDB metadata table. If the METADATA_NAME argument is not supplied, the default value is EmrFSMetadata.</td>
</tr>
<tr>
<td>s3://s3Path</td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>--read-consumption</td>
<td>READ_CONSUMPTION</td>
<td>No</td>
</tr>
<tr>
<td>--write-consumption</td>
<td>WRITE_CONSUMPTION</td>
<td>No</td>
</tr>
</tbody>
</table>

emrfs sync command example

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

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

emrfs read-sqs sub-command

[options] for emrfs read-sqs

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>-q QUEUE_NAME</td>
<td>--queue-name QUEUE_NAME</td>
<td>QUEUE_NAME is the name of the Amazon SQS queue configured in emrfs-site.xml. The default value is EMRFS-Inconsistency-jobFlowId.</td>
</tr>
<tr>
<td>-o OUTPUT_FILE</td>
<td>--output-file OUTPUT_FILE</td>
<td>OUTPUT_FILE is the path to the output file on the master node's local file system. Messages read from the queue are written to this file.</td>
</tr>
</tbody>
</table>
emrfs delete-sqs sub-command

[options] for emrfs delete-sqs

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>-q QUEUE_NAME</td>
<td>--queue-name QUEUE_NAME</td>
<td>QUEUE_NAME is the name of the Amazon SQS queue configured in emrfs-site.xml. The default value is EMRFS-Inconsistency-&lt;jobFlowId&gt;.</td>
</tr>
</tbody>
</table>

Submitting EMRFS CLI Commands as Steps

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

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

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

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

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

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

Authorizing Access to EMRFS Data in Amazon S3

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

By default, the role that you specify as the EC2 instance profile for Amazon EMR determines the permissions for accessing EMRFS data in Amazon S3. The IAM policies that are attached to this role apply regardless of the user or group making the request through EMRFS. The default is EMR_EC2_DefaultRole.

Beginning with Amazon EMR release version 5.10.0, you can use a security configuration to set up EMRFS authorization for a cluster, which allows you to implement fine-grained access control to EMRFS data in Amazon S3 for clusters that have multiple users. You can specify different IAM roles for different users and groups. The EC2 instance profile can assume a role based on the user or group that makes the request. In addition, you can specify different IAM roles based on the location of EMRFS data in Amazon S3. For more information, see EMRFS Authorization for Data in Amazon S3 (p. 214).

Alternatively, you can define a custom credentials provider class, which allows you to customize access to EMRFS data in Amazon S3 if your Amazon EMR solution prevents you from using EMRFS authorization.
Creating a Custom Credentials Provider for EMRFS Data in Amazon S3

To create a custom credentials provider, you implement the AWS CredentialsProvider and the Hadoop Configurable classes.

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

The basic steps are as follows:

To specify a custom credentials provider

1. Create a custom credentials provider class compiled as a JAR file.
2. Run a script as a bootstrap action to copy the custom credentials provider JAR file to the /usr/share/aws/emr/emrfs/auxlib location on the cluster's master node. For more information about bootstrap actions, see (Optional) Create Bootstrap Actions to Install Additional Software.
3. Use a bootstrap action to modify the emrfs-site.xml file when you create the cluster so that it specifies the class that you defined in the JAR file.

Specifying Amazon S3 Encryption Using EMRFS Properties

**Important**

Beginning with Amazon EMR release version 4.8.0, you can use security configurations to apply encryption settings more easily and with more options. We recommend using security configurations. For information, see Configure Data Encryption (p. 224). The console instructions described in this section are available for release versions earlier than 4.8.0. If you use the AWS CLI to configure Amazon S3 encryption both in the cluster configuration and in a security configuration in subsequent versions, the security configuration overrides the cluster configuration.

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

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

To specify EMRFS encryption options using the AWS Management Console

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

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

- If you choose an option that uses AWS Key Management, choose an AWS KMS Key ID. For more information, see Using AWS KMS Customer Master Keys (CMKs) for EMRFS Encryption (p. 62).
- If you choose S3 client-side encryption with custom materials provider, provide the Class name and the JAR location. For more information, see Amazon S3 Client-Side Encryption (p. 63).

7. Choose other options as appropriate for your application and then choose Create Cluster.

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

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

The role for the Amazon EC2 instance profile must have permission to use the CMK you specify. The default role for the instance profile in Amazon EMR is EMR_EC2_DefaultRole. If you use a different role for the instance profile, or you use EMRFS authorization to assume different roles based on the call to Amazon S3, make sure that each role is added as a key user as appropriate. This gives the role permission to use the CMK. For more information, see Using Key Policies in the AWS Key Management Service Developer Guide and Use Default IAM Roles and Managed Policies (p. 235).

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

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

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

1. Sign in to the AWS Management Console and open the AWS Identity and Access Management (IAM) console at https://console.aws.amazon.com/iam/.
2. In the left navigation pane, choose Encryption keys.
3. For Region, choose the appropriate AWS Region. Do not use the region selector in the navigation bar (top right corner).
4. Select the alias of the CMK to modify.
5. On the key details page under Key Users, choose Add.
6. In the Attach dialog box, select the appropriate role. The name of the default role is EMR_EC2_DefaultRole.
7. Choose Attach.

Amazon S3 Server-Side Encryption

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

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

- **SSE-S3**: Amazon S3 manages keys for you.
Specifying Amazon S3 Encryption Using EMRFS Properties

- **SSE-KMS**: You use an AWS KMS customer master key (CMK) set up with policies suitable for Amazon EMR. For more information about key requirements for Amazon EMR, see Using AWS KMS Customer Master Keys (CMKs) for Encryption (p. 211). When you use AWS KMS, charges apply for the storage and use of encryption keys. For more information, see AWS KMS Pricing.

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

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

- Type the following command:

  ```bash
  aws emr create-cluster --release-label emr-4.7.2 or earlier \ 
  --instance-count 3 --instance-type m1.large --emrfs Encryption=ServerSide
  ``

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

**Amazon S3 Client-Side Encryption**

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

**Note**

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

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

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

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

**Creating a Custom Key Provider**

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

The `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.

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

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

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

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

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

    @Override
    public void refresh() {
    }

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

    @Override
    public EncryptionMaterials getEncryptionMaterials() {
        return this.encryptionMaterials;
    }
}
```
Specifying a Custom Materials Provider Using the AWS CLI

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

```bash
aws emr create-cluster --instance-type m3.xlarge --ami-version 3.11.0 --emrfs
    Encryption=ClientSide,ProviderType=Custom,CustomProviderLocation=s3://mybucket/myfolder/provider.jar,CustomProviderClass=classname
```

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

Specifying a Custom Materials Provider Using an SDK

To use an SDK, you can first use the s3get bootstrap action to download the custom EncryptionMaterialsProvider class that you store in Amazon S3 to each node in your cluster. You can then use a second bootstrap action, configure-hadoop, to configure the emrfs-site.xml file with CSE enabled and the proper location of the custom provider.

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

```java
<snip>
    ScriptBootstrapActionConfig s3getConfig = new ScriptBootstrapActionConfig()
    .withPath("file:/usr/share/aws/emr/scripts/s3get")
    .withArgs("-s","s3://mybucket/MyCustomEncryptionMaterialsProvider.jar","-d","/usr/share/aws/emr/auxlib/");
    ScriptBootstrapActionConfig emrfsConfig = new ScriptBootstrapActionConfig()
    .withPath("file:/usr/share/aws/emr/scripts/configure-hadoop")
    .withArgs("-e","fs.s3.cse.enabled=true","-e","fs.s3.cse.encryptionMaterialsProvider=full.class.name.of.EncryptionMaterialsProvider","-e","-e,myProvider.arg1=value1");

    BootstrapActionConfig s3get = new BootstrapActionConfig().withScriptBootstrapAction(s3getConfig);
    BootstrapActionConfig emrfs = new BootstrapActionConfig().withScriptBootstrapAction(emrfsConfig);
    s3get.setName("s3getBA");
    emrfs.setName("emrfsBA");

    RunJobFlowRequest request = new RunJobFlowRequest()
    .withName("Client-side enabled EMRFS")
    .withAmiVersion("3.11.0")
    .withLogUri("s3://myLogUri")
    .withBootstrapActions(s3get,emrfs)
    .withInstances(new JobFlowInstancesConfig()
    .withEc2KeyName("myEc2Key")
    .withInstanceCount(1)
    .withKeepJobFlowAliveWhenNoSteps(true)
    .withMasterInstanceType("m3.xlarge")
    .withSlaveInstanceType("m3.xlarge")
    );

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

Specifying a Custom Materials Provider Using an SDK

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

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

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

Configuration myEmrfsConfig = new Configuration()
                          .withClassification("emrfs-site")
                          .withProperties(emrfsProperties);

RunJobFlowRequest request = new RunJobFlowRequest()
                           .withName("Custom EncryptionMaterialsProvider")
                           .withReleaseLabel(""")
                           .withApplications(myApp)
                           .withConfigurations(myEmrfsConfig)
                           .withServiceRole("EMR_DefaultRole")
                           .withLogUri("s3://myLogUri/")
                           .withInstances(new JobFlowInstancesConfig()
                             .withEc2KeyName("myEc2Key")
                             .withInstanceCount(2)
                             .withKeepJobFlowAliveWhenNoSteps(true)
                             .withMasterInstanceType("m3.xlarge")
                             .withSlaveInstanceType("m3.xlarge")
                           );

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

Custom EncryptionMaterialsProvider with Arguments

You may need to pass arguments directly to the provider. From the AWS CLI, you can do this using the `configure-hadoop` bootstrap action to supply arguments to `emrfs-site.xml`:

```bash
aws emr create-cluster --ami-version 3.10.0 --instance-type m3.xlarge --instance-count 2 --ec2-availability-zone us-west-2a emrfs Encryption=ClientSide,CustomProviderLocation=s3://mybucket/myfolder/myprovider.jar,CustomProviderClass=classname --bootstrap-action Path=s3://elasticmapreduce/bootstrap-actions/configure-hadoop,Args=[-e,myProvider.arg1=value1,-e,myProvider.arg2=value2]
```
### emrfs-site.xml Properties for Amazon S3 Client-Side Encryption

<table>
<thead>
<tr>
<th>Property</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fs.s3.cse.enabled</td>
<td>false</td>
<td>When set to <code>true</code>, EMRFS objects stored in Amazon S3 are encrypted using client-side encryption.</td>
</tr>
<tr>
<td>fs.s3.cse.encryptionMaterialsProvider.uri</td>
<td>N/A</td>
<td>Applies when using custom encryption materials. The Amazon S3 URI where the JAR with the EncryptionMaterialsProvider is located. When you provide this URI, Amazon EMR automatically downloads the JAR to all nodes in the cluster.</td>
</tr>
<tr>
<td>fs.s3.cse.encryptionMaterialsProvider</td>
<td>N/A</td>
<td>The EncryptionMaterialsProvider class path used with client-side encryption. When using CSE-KMS, specify <code>com.amazon.ws.emr.hadoop.fs.cse.KMSEncryptionMaterialsProvider</code>.</td>
</tr>
<tr>
<td>fs.s3.cse.materialsDescription.enabled</td>
<td>false</td>
<td>When set to <code>true</code>, populates the materialsDescription of encrypted objects with the Amazon S3 URI for the object and the JobFlowId. Set to <code>true</code> when using custom encryption materials.</td>
</tr>
<tr>
<td>fs.s3.cse.kms.keyId</td>
<td>N/A</td>
<td>Applies when using CSE-KMS. The value of the KeyId, ARN, or alias of the AWS KMS CMK used for encryption.</td>
</tr>
<tr>
<td>fs.s3.cse.cryptoStorageMode</td>
<td>ObjectMetadata</td>
<td>The Amazon S3 storage mode. By default, the description of the encryption information is stored in the object metadata. You can also store the description in an instruction file. Valid values are ObjectMetadata and InstructionFile. For more information, see <a href="https://docs.aws.amazon.com/emr/latest/developerguide/emrfs-s3-client-side-cse.html">Client-Side Data Encryption with the AWS SDK for Java and Amazon S3</a>.</td>
</tr>
</tbody>
</table>

### Configure a Cluster to be Transient or Long-Running

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.
You can run your cluster as a transient process: one that launches the cluster, loads the input data, processes the data, stores the output results, and then automatically shuts down. This is the standard model for a cluster that performs a periodic processing task. Shutting down the cluster automatically ensures that you are only billed for the time required to process your data.

The other model for running a cluster is as a long-running cluster. In this model, the cluster launches and loads the input data. From there, you might interactively query the data, use the cluster as a data warehouse, or do periodic processing on a data set so large that it would be inefficient to load the data into new clusters each time. In this model, the cluster persists even when there are no tasks queued for processing. Another option you might want to enable on a long-running cluster is termination protection. This protects your cluster from being terminated accidentally or in the event that an error occurs. For more information, see Managing Cluster Termination (p. 474).

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

To launch a transient cluster using the console

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. Choose Create cluster.
3. Choose Go to advanced options.
4. Under Add steps (optional) select Auto-terminate cluster after the last step is completed.
5. Proceed with creating the cluster as described in Plan and Configure Clusters (p. 27).

To launch a transient cluster using the AWS CLI

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

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

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

  aws emr create-cluster --name "Test cluster" --ami-version 3.8 --applications Name=Hue Name=Hive Name=Pig \  --use-default-roles --ec2-attributes KeyName=myKey \  --instance-type m3.xlarge --instance-count 3 --auto-terminate

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

  For more information on using Amazon EMR commands in the AWS CLI, see AWS CLI Reference.
Amazon EMR uses an Amazon Machine Image (AMI) to install Linux, Hadoop, and other software on the instances that it launches in the cluster. New versions of the Amazon EMR AMI are released on a regular basis, adding new features and fixing issues. We recommend that you use the latest AMI to launch your cluster whenever possible. The latest version of the AMI is the default when you launch a cluster from the console.

The AWS version of Hadoop installed by Amazon EMR is based on Apache Hadoop, with patches and improvements added that make it work efficiently with AWS. Each Amazon EMR AMI has a default version of Hadoop associated with it. If your application requires a different version of Hadoop than the default, specify that Hadoop version when you launch the cluster.

In addition to the standard software and applications that are available for installation on the cluster, you can use bootstrap actions to install custom software and to change the configuration of applications on the cluster. Bootstrap actions are scripts that are run on the instances when Amazon EMR launches the cluster. You can write custom bootstrap actions, or use predefined bootstrap actions provided by Amazon EMR. A common use of bootstrap actions is to change the Hadoop configuration settings.

For more information, see the following topics:

Topics
- Choose an Amazon Machine Image (AMI) (p. 69)
- Choose a Version of Hadoop (p. 120)
- (Optional) Create Bootstrap Actions to Install Additional Software (p. 129)

Choose an Amazon Machine Image (AMI)

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Amazon EMR uses Amazon Machine Images (AMIs) to initialize the EC2 instances it launches to run a cluster. The AMIs contain the Linux operating system and other software used to run the cluster. These AMIs are specific to Amazon EMR and can be used only in the context of running a cluster. Periodically, Amazon EMR updates these AMIs with new versions of applications such as Hadoop and other software, so users can take advantage of improvements and new features. If you create a new cluster using an updated AMI, you must ensure that your custom applications will work with it.

For general information about AMIs, see Amazon Machine Images in the Amazon EC2 User Guide for Linux Instances. For more information about the software versions included in the Amazon EMR AMIs, see AMI Versions Supported in Amazon EMR Versions 2.x and 3.x (p. 73).

AMI versioning gives you the option to choose the specific AMI your cluster uses to launch EC2 instances. If your application depends on a specific version or configuration of Hadoop, you might want delay upgrading to a new AMI until you have tested your application on it.

Specifying the AMI version during cluster creation is required when you use the console or the AWS CLI and is optional in the API, and SDK. If you specify an AMI version when you create a cluster, your instances will be created using that AMI. This provides stability for long-running or mission-critical applications. The trade-off is that your application will not have access to new features on more up-to-date AMI versions unless you launch a new cluster using a newer AMI. For more information on specifying the AMI version, see AMI Version Numbers (Versions 2.x, 3.x) (p. 70).

In the API and SDK, the AMI version is optional; if you do not provide an AMI version parameter, and you are using the API or SDK, your clusters will run on the default AMI version for the tool you are using.

Topics
- AMI Version Numbers (Versions 2.x, 3.x) (p. 70)
Choose an Amazon Machine Image (AMI)

- Specify the AMI Version for a New Cluster (p. 70)
- View the AMI Version of a Running Cluster (p. 71)
- Amazon EMR AMI Deprecation (p. 72)
- AMI Versions Supported in Amazon EMR Versions 2.x and 3.x (p. 73)

AMI Version Numbers (Versions 2.x, 3.x)

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

AMI version numbers are composed of three parts major-version.minor-version.patch. There are several ways to specify which version of the AMI to use to launch your cluster, depending on the tool you use to launch the cluster: the SDK, API, or AWS CLI.

- **Fully specified**—If you specify the AMI version using all three parts (for example --ami-version 2.0.1) your cluster will be launched on exactly that version. This is useful if you are running an application that depends on a specific AMI version. All tools support this option.

- **Major-minor version specified**—If you specify just the major and minor version for the AMI (for example --ami-version 2.0), your cluster will be launched on the AMI that matches those specifications and has the latest patches. If the latest patch for the 2.0 AMI series is .4, the preceding example would launch a cluster using AMI 2.0.4. This option ensures that you receive the benefits of the latest patches for the AMI series specified. All tools support this option.

Specify the AMI Version for a New Cluster

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

You must specify which AMI version a new cluster should use when you launch it. For more information about the default configuration and applications available on AMI versions, see AMI Versions Supported in Amazon EMR Versions 2.x and 3.x (p. 73).

**To specify an AMI version during cluster launch using the AWS CLI**

When you create a cluster using the AWS CLI, add the --ami-version parameter to the create-cluster subcommand. The --ami-version parameter is required when you create a cluster using the AWS CLI.

- To launch a cluster and fully specify the AMI (using AMI version 3.11.0), type the following command and replace myKey with the name of your EC2 key pair.

  ```
  aws emr create-cluster --name "Test cluster" --ami-version 3.11.0 \\n  --applications Name=Hue Name=Hive Name=Pig \\n  --use-default-roles --ec2-attributes KeyName=myKey \\n  --instance-count 5 --instance-type m3.xlarge
  ```

- Linux, UNIX, and Mac OS X users:

  ```
  aws emr create-cluster --name "Test cluster" --ami-version 3.11.0 --applications Name=Hue Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey --instance-count 5 --instance-type m3.xlarge
  ```

- Windows users:

  ```
  aws emr create-cluster --name "Test cluster" --ami-version 3.11.0 --applications Name=Hue Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey --instance-count 5 --instance-type m3.xlarge
  ```
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 `--instance-type`.

**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.

The following example specifies the AMI using the major and minor version. The cluster is launched on the AMI that matches those specifications and has the latest patches. For example, if the most recent AMI version is 3.11.0, specifying `--ami-version 3` would launch a cluster using AMI 3.11.0.

```bash
aws emr create-cluster --name "Test cluster" --ami-version 2.4 --applications Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey --instance-count 5 --instance-type m3.xlarge
```

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

## View the AMI Version of a Running Cluster

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

If you need to find out which AMI version a cluster is running, you can retrieve this information using the console, the CLI, or the API.

**To view the current AMI version using the console**

1. Sign in to the AWS Management Console and open the Amazon EMR console at [https://console.aws.amazon.com/elasticmapreduce/](https://console.aws.amazon.com/elasticmapreduce/).
2. Select a cluster link on the **Cluster List** page. The **AMI Version** and other details about the cluster are displayed on the **Cluster Details** page.

**To view the current AMI version using the AWS CLI**

Type the `describe-cluster` subcommand with the `cluster-id` parameter to retrieve information about a cluster including the AMI version. The cluster identifier is required to use the `describe-cluster` subcommand. You can retrieve the cluster ID using the console or the `list-clusters` subcommand.

- Type the following command to view cluster information.

```bash
aws emr describe-cluster --cluster-id j-3QKHXXXXXXXARD
```

The output shows the AMI version for the cluster:

```json
{
    "Cluster": {
        "Status": {
            "Timeline": {
```

71
Choose an Amazon Machine Image (AMI)

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

To view the current AMI version using the API

- Call DescribeJobFlows to check which AMI version a cluster is using. The version will be returned as part of the response data, as shown in the following example. For the complete response syntax, go to DescribeJobFlows in the Amazon EMR API Reference.

```
  <DescribeJobFlowsResult>
    <JobFlows>
      <member>
        ...
        <AmiVersion>
          2.1.3
        </AmiVersion>
        ...
      </member>
    </JobFlows>
    <ResponseMetadata>
      <RequestId>9cea3229-ed85-11dd-9877-6fada48a8419</RequestId>
    </ResponseMetadata>
  </DescribeJobFlowsResult>
</DescribeJobFlowsResponse>
```

Amazon EMR AMI Deprecation

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Eighteen months after an AMI version is released, the Amazon EMR team might choose to deprecate that AMI version and no longer support it. In addition, the Amazon EMR team might deprecate an AMI before
eighteen months has elapsed if a security risk or other issue is identified in the software or operating system of the AMI. If a cluster is running when its AMI is depreciated, the cluster will not be affected. You will not, however, be able to create new clusters with the depreciated AMI version. The best practice is to plan for AMI obsolescence and move to new AMI versions as soon as is practical for your application.

**AMI Versions Supported in Amazon EMR Versions 2.x and 3.x**

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Amazon EMR supports the 3.x AMI versions listed in the following tables. The 2.x release series is now deprecated. You must specify the AMI version to use when you create a cluster using the AWS CLI and the console.

*Important*

- We recommend you migrate to an Amazon EMR 4.x release. For more information, see the [Amazon EMR Release Guide](#).

### Hadoop 2 AMI Versions

<table>
<thead>
<tr>
<th>AMI Version</th>
<th>Includes</th>
<th>Notes</th>
<th>Release Date</th>
</tr>
</thead>
</table>
| 3.11.0      | - AWS SDK for Java 1.10.41  
             - AWS SDK for Ruby 2.2.8  
             - Amazon Linux version 2015.09  
             - Hadoop 2.4.0  
             - Hive 0.13.1  
             - Hue 3.7.1  
             - Pig 0.12.0  
             - HBase 0.94.18  
             - Impala 1.2.4  
             - Mahout 0.9  
             - Java: Oracle/Sun jdk-7u76  
             - Perl 5.16.3  
             - PHP 5.6.14  
             - Python 2.6.9  
             - R 3.2.2  
             - Ruby 1.8.7  
             - Scala 2.11.1  
             - Spark 1.3.1 | This Amazon EMR AMI version provides the following bug fixes and changes:  
             - Fixed a bug that prevented MapReduce JobHistory logs from pushing to Amazon S3.  
             - Fixed bugs that prevented YARN container logs from being pushed to Amazon S3. | 4 January 2016 |
| 3.10.0      | - AWS SDK for Java 1.10.0  
             - AWS SDK for Ruby 1.9  
             - Amazon Linux version 2015.03  
             - Hadoop 2.4.0  
             - Hive 0.13.1  
             - Hue 3.7.1  
             - Pig 0.12.0 | This Amazon EMR AMI version provides the following bug fixes and changes: | 2 October 2015 |
<table>
<thead>
<tr>
<th>AMI Version</th>
<th>Includes</th>
<th>Notes</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• HBase 0.94.18</td>
<td>• Fixed a bug which was a regression to YARN-90 that caused issues</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Impala 1.2.4</td>
<td>• with log aggregation to Amazon S3.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Mahout 0.9</td>
<td>• Patched YARN-3241 (YARN-3241.002.patch).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Java: Oracle/Sun jdk-7u76</td>
<td>• Set HBase logging level to WARN.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Perl 5.16.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• PHP 5.6.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Python 2.6.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• R 3.1.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Ruby 2.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Scala 2.11.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Spark 1.3.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AMI Version</td>
<td>Includes</td>
<td>Notes</td>
<td>Release Date</td>
</tr>
<tr>
<td>-------------</td>
<td>----------</td>
<td>-------</td>
<td>--------------</td>
</tr>
</tbody>
</table>
| 3.9.0       | • AWS SDK for Java 1.10  
• AWS SDK for Ruby 1.9  
• Amazon Linux version 2015.03  
• Hadoop 2.4.0  
• Hive 0.13.1  
• Hue 3.7.1  
• Pig 0.12.0  
• HBase 0.94.18  
• Impala 1.2.4  
• Mahout 0.9  
• Java: Oracle/Sun jdk-7u76  
• Perl 5.16.3  
• PHP 5.6.9  
• Python 2.6.9  
• R 3.1.1  
• Ruby 2.0  
• Scala 2.11.1  
• Spark 1.3.1 | This Amazon EMR AMI version provides the following major bug fixes:  
• Fixed an issue where multiple service daemon processes resulted in job failures.  
• Fixed an issue with orphaned daemon processes not being killed before a new one starts.  
• Fixed an issue that resulted in missing job history logs.  
• Fixed an issue in EMRFS client-side encryption where deleting without an instruction file present caused an uncaught exception and failure.  
• Provided a bootstrap action that fixes excessive logging encountered in HBase shell. For more | 19 August 2015 |
<table>
<thead>
<tr>
<th>AMI Version</th>
<th>Includes</th>
<th>Notes</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>information, see HBase Shell Excessive Debug Logging (p. 503)</td>
<td></td>
</tr>
<tr>
<td>AMI Version</td>
<td>Includes</td>
<td>Notes</td>
<td>Release Date</td>
</tr>
<tr>
<td>-------------</td>
<td>----------</td>
<td>-------</td>
<td>--------------</td>
</tr>
</tbody>
</table>
| 3.3.0       | - AWS SDK for Ruby 1.9  
- Amazon Linux version 2014.03  
- Hadoop 2.4.0  
- Hive 0.13.1  
- Hue 3.6  
- Pig 0.12.0  
- HBase 0.94.18  
- Impala 1.2.4  
- Mahout 0.9  
- Java: Oracle/Sun jdk-7u71  
- Perl 5.16.3  
- PHP 5.3.28  
- Python 2.6.9  
- R 3.0.2  
- Ruby 2.0  
- Scala 2.11.1 | This Amazon EMR AMI version provides the following features and bug fixes:  
**Major Feature Updates**  
- Amazon EMR now supports HUE, an open-source user interface for Hadoop that makes it easier to interact with your cluster. With Hue, you can run and develop Hive queries, manage files in HDFS, run and develop Pig scripts, and manage tables. Hue also enables you to browse and use files in Amazon S3. For more information, see *Configure Hue to View, Query, or Manipulate Data* (p. 348).  
- Include Oozie as part of Hue release. | 6 November 2014 |
Legacy AMIs

The following AMIs are deprecated. Although you may be able to launch clusters with these AMIs, they may have defects or older software we no longer support. However, we have kept the documentation here for your reference.

Deprecated Hadoop 2 AMI Versions

<table>
<thead>
<tr>
<th>AMI Version</th>
<th>Includes</th>
<th>Notes</th>
<th>Release Date</th>
</tr>
</thead>
</table>
| 3.8.0       | • AWS SDK for Java 1.9  
  • AWS SDK for Ruby 1.9  
  • Amazon Linux version 2015.03  
  • Hadoop 2.4.0  
  • Hive 0.13.1  
  • Hue 3.7  
  • Pig 0.12.0  
  • HBase 0.94.18  
  • Impala 1.2.4  
  • Mahout 0.9  
  • Java: Oracle/Sun jdk-7u76  
  • Perl 5.16.3  
  • PHP 5.6.9  
  • Python 2.6.9  
  • R 3.1.1  
  • Ruby 2.0  
  • Scala 2.11.1  
  • Spark 1.3.1 | This Amazon EMR AMI version provides the following:  
  **Major Feature Release**  
  • Support for Apache Spark. For more information, see Apache Spark (p. 288).  
  **Major Bug Fixes**  
  • Fixed a bug in the Amazon EMR-DynamoDB connector which resulted in the errant generation of zero map tasks if the master and core instance types were different.  
  • Fixed an issue in Hive that resulted in the deletion of the target S3 folder of an `explain insert overwrite` query. | 10 June 2015 |
Choose an Amazon Machine Image (AMI)

<table>
<thead>
<tr>
<th>AMI Version</th>
<th>Includes</th>
<th>Notes</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Includes the following patches: HIVE-8746, HIVE-8566, HIVE-8162, PIG-4496

Other Issues

- Added srcPrefixesFile to S3DistCp. For more information, see [S3DistCp Options](#) (p. 387)
- The Amazon EMR-DynamoDB connector now allows customers to load a custom AWS credentials provider for use with the connector.
- Remove unused property, skips3scratch, from hive-default.xml.
<table>
<thead>
<tr>
<th>AMI Version</th>
<th>Includes</th>
<th>Notes</th>
<th>Release Date</th>
</tr>
</thead>
</table>
| 3.7.0       | - AWS SDK for Java 1.9  
- AWS SDK for Ruby 1.9  
- Amazon Linux version 2015.03  
- Hadoop 2.4.0  
- Hive 0.13.1  
- Hue 3.7  
- Pig 0.12.0  
- HBase 0.94.18  
- Impala 1.2.4  
- Mahout 0.9  
- Java: Oracle/Sun jdk-7u71  
- Perl 5.16.3  
- PHP 5.3.28  
- Python 2.6.9  
- R 3.0.2  
- Ruby 2.0  
- Scala 2.11.1 | This Amazon EMR AMI version provides the following feature release:  
**Major Bug Fixes**  
- `/var/log` is moved to `/mnt/var/log`. Any files written to `/var/log` will be written to `/mnt/var/log` as there is now a symbolic link between the two paths.  
- Addresses an issue where some components would cause certain scripts running out of `/etc/init.d` to not work, causing issues with the yum installer.  
- Hue no longer starts the HBase Thrift server when installed.  
- Python's base and package installations now use pip instead of easy_install.  
- The default Ruby will | 21 April 2015 |
<table>
<thead>
<tr>
<th>AMI Version</th>
<th>Includes</th>
<th>Notes</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>now match the latest released Amazon Linux. Previously, Amazon EMR would use a previous version of Ruby. While these interpreters are still available, the default will now reflect the latest version of Amazon Linux.</td>
<td></td>
</tr>
</tbody>
</table>

**Other Issues**
- Amazon EMR now honors the DHCP configuration of its VPC entirely. Some commands that used to return a fully-qualified domain name will now only return the Amazon EC2 name of the host. Scripts that expect this behavior will fail. For more information, see Errors That Result in START_FAILED (p. 532)
<table>
<thead>
<tr>
<th>AMI Version</th>
<th>Includes</th>
<th>Notes</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.6.0</td>
<td>• AWS SDK for Java 1.9</td>
<td>This Amazon EMR AMI version provides the following feature release:</td>
<td>24 March 2015</td>
</tr>
<tr>
<td></td>
<td>• AWS SDK for Ruby 1.9</td>
<td><strong>Major Feature Release</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Amazon Linux version 2014.09</td>
<td>• EMRFS supports Amazon S3 client-side encryption. For more information, see Amazon S3 Client-Side Encryption (p. 63).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Hadoop 2.4.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Hive 0.13.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Hue 3.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Pig 0.12.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• HBase 0.94.18</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Impala 1.2.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Mahout 0.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Java: Oracle/Sun jdk-7u71</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Perl 5.16.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• PHP 5.3.28</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Python 2.6.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• R 3.0.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Ruby 2.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Scala 2.11.1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Choose an Amazon Machine Image (AMI)

<table>
<thead>
<tr>
<th>AMI Version</th>
<th>Includes</th>
<th>Notes</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.5.0</td>
<td>• AWS SDK for Java 1.9</td>
<td>This Amazon EMR AMI version provides the following features and bug fixes:</td>
<td>10 March 2015</td>
</tr>
<tr>
<td></td>
<td>• AWS SDK for Ruby 1.9</td>
<td>Major Bug Fixes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Amazon Linux version 2014.09</td>
<td>• Fixes a bug which prevented s3distcp from using temporary credentials (such as with Amazon STS tokens).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Hadoop 2.4.0</td>
<td>• Fixes a bug that caused Amazon S3 multipart upload to hang on an individual upload.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Hive 0.13.1</td>
<td>• Fixes performance issues with the Hive-DynamoDB connector.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Hue 3.7</td>
<td>• Fixes an issue encountered in the Hue redirection middleware, which caused redirection to fail when redirected to SSL endpoints.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Pig 0.12.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• HBase 0.94.18</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Impala 1.2.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Mahout 0.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Java: Oracle/Sun jdk-7u71</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Perl 5.16.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• PHP 5.3.28</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Python 2.6.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• R 3.0.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Ruby 2.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Scala 2.11.1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Choose an Amazon Machine Image (AMI)

<table>
<thead>
<tr>
<th>AMI Version</th>
<th>Includes</th>
<th>Notes</th>
<th>Release Date</th>
</tr>
</thead>
</table>
| 3.4.0       | • AWS SDK for Java 1.9  
• AWS SDK for Ruby 1.9  
• Amazon Linux version 2014.09  
• Hadoop 2.4.0  
• Hive 0.13.1  
• Hue 3.7  
• Pig 0.12.0  
• HBase 0.94.18  
• Impala 1.2.4  
• Mahout 0.9  
• Java: Oracle/Sun jdk-7u71  
• Perl 5.16.3  
• PHP 5.3.28  
• Python 2.6.9  
• R 3.0.2  
• Ruby 2.0  
• Scala 2.11.1 | This Amazon EMR AMI version provides the following features and bug fixes:  
**Major Feature Updates**  
• Updated Hue version from 3.6 to 3.7.1.  
• Added support for EBS-backed HVM instances in all regions.  
• Added General Purpose EBS root volume for all instance types that use EBS-backed HVM in all regions except for US East (N. Virginia), US West (N. California), and Asia Pacific (Tokyo) where instances in those regions use Standard EBS volumes.  
• Added a modification to the Kinesis connector which requires customers to co-locate the | 26 February 2015 |
<table>
<thead>
<tr>
<th>AMI Version</th>
<th>Includes</th>
<th>Notes</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>DynamoDB table with their EMR cluster in the same region.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Major Bug Fixes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Minor performance-related bug fixes.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Includes this patch: HIVE-7323</td>
<td></td>
</tr>
<tr>
<td>AMI Version</td>
<td>Includes</td>
<td>Notes</td>
<td>Release Date</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------</td>
<td>------------------</td>
</tr>
</tbody>
</table>
| 3.3.2       | • AWS SDK for Ruby 1.9  
• Amazon Linux version 2014.03  
• Hadoop 2.4.0  
• Hive 0.13.1  
• Hue 3.6  
• Pig 0.12.0  
• HBase 0.94.18  
• Impala 1.2.4  
• Mahout 0.9  
• Java: Oracle/Sun jdk-7u71  
• Perl 5.16.3  
• PHP 5.3.28  
• Python 2.6.9  
• R 3.0.2  
• Ruby 2.0  
• Scala 2.11.1 | This Amazon EMR AMI version provides the following features and bug fixes:  
**Major Feature Updates**  
• Fixes performance issues in DynamoDB connector  
**Major Bug Fixes**  
• Fixes an issue in Hue encountered when copying files greater than 64MB from HDFS to Amazon S3.  
• Fixes an issue in the Hue S3 Browser which incorrectly handled non-ASCII key names in the EU (Frankfurt) and China (Beijing) regions.  
• Fixes an issue encountered in Hue when using a remote Hive Metastore database that has existing | 4 February 2015 |
<table>
<thead>
<tr>
<th>AMI Version</th>
<th>Includes</th>
<th>Notes</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Hue sample tables.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fixed an issue encountered in Hue when saving results from a multiple statement Hive query.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fixes a mismatch between the installed default Python and pip.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Includes this patch: HIVE-7426</td>
<td></td>
</tr>
<tr>
<td>AMI Version</td>
<td>Includes</td>
<td>Notes</td>
<td>Release Date</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
<td>----------------------------</td>
</tr>
</tbody>
</table>
| 3.3.1       | • AWS SDK for Ruby 1.9  
• Amazon Linux version 2014.03  
• Hadoop 2.4.0  
• Hive 0.13.1  
• Hue 3.6  
• Pig 0.12.0  
• HBase 0.94.18  
• Impala 1.2.4  
• Mahout 0.9  
• Java: Oracle/Sun jdk-7u71  
• Perl 5.16.3  
• PHP 5.3.28  
• Python 2.6.9  
• R 3.0.2  
• Ruby 2.0  
• Scala 2.11.1 | This Amazon EMR AMI version provides the following features and bug fixes:  
**Major Bug Fixes**  
• Fixes an issue where installing Impala caused certain Hive commands to fail.  
• Fixed multiple issues in Hue Amazon S3 browser when used in the EU (Frankfurt) region.  
• User home directories are now created if not found, fixing an issue where re-using an external Hue database resulted in some services to fail, notably Oozie.  
• Fixed a security issue in the Hue Pig editor.  
• Fixed the redirect_whitelist Hue configuration option allowing | 20 November 2014 |
## Choose an Amazon Machine Image (AMI)

<table>
<thead>
<tr>
<th>AMI Version</th>
<th>Includes</th>
<th>Notes</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>whitelisting of domains to which Hue can redirect.</td>
<td>89</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fixed an issue with loading saved queries in Hive Editor when browsing Hue in Firefox.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Removed permission buttons in the Hue Amazon S3 browser as those operations are not supported.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fixed an issue with the configure-hadoop script which caused a loss of file permissions when used.</td>
<td></td>
</tr>
<tr>
<td>AMI Version</td>
<td>Includes</td>
<td>Notes</td>
<td>Release Date</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>3.3.0</td>
<td>• AWS SDK for Ruby 1.9</td>
<td>This Amazon EMR AMI version provides the following features and bug fixes:</td>
<td>6 November 2014</td>
</tr>
<tr>
<td></td>
<td>• Amazon Linux version 2014.03</td>
<td>Major Feature Updates</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Hadoop 2.4.0</td>
<td>• Amazon EMR now supports HUE, an open-source user interface for Hadoop that makes it easier to interact with your cluster. With Hue, you can run and develop Hive queries, manage files in HDFS, run and develop Pig scripts, and manage tables. Hue also enables you to browse and use files in Amazon S3. For more information, see Configure Hue to View, Query, or Manipulate Data (p. 348).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Hive 0.13.1</td>
<td>• Include Oozie as part of Hue release.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Hue 3.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Pig 0.12.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• HBase 0.94.18</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Impala 1.2.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Mahout 0.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Java: Oracle/Sun jdk-7u71</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Perl 5.16.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• PHP 5.3.28</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Python 2.6.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• R 3.0.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Ruby 2.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Scala 2.11.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AMI Version</td>
<td>Includes</td>
<td>Notes</td>
<td>Release Date</td>
</tr>
<tr>
<td>-------------</td>
<td>----------</td>
<td>-------</td>
<td>--------------</td>
</tr>
</tbody>
</table>
| 3.2.3       | - AWS SDK for Ruby 1.9  
- Amazon Linux version 2014.03  
- Hadoop 2.4.0  
- Hive 0.13.1  
- Pig 0.12  
- HBase 0.94.18  
- Impala 1.2.4  
- Mahout 0.9  
- Java: Oracle/Sun jdk-7u65  
- Perl 5.16.3  
- PHP 5.3.28  
- Python 2.6.9  
- R 3.0.2  
- Ruby 2.0  
- Scala 2.11.1 | This Amazon EMR AMI version provides the following features and bug fixes:  
**Major Feature Updates**  
- EMRFS now supports Amazon S3 eventual consistency notifications for Amazon SQS and eventual consistency metrics for CloudWatch.  
- Performance optimizations for Amazon S3 multipart uploads with EMRFS.  
- The configure-hadoop bootstrap action now supports configuring log levels for different Hadoop daemons. You can now configure separate appenders for each daemon, and the following new identifiers are provided: HADOOP, MAPRED, JHS, and YARN. | 31 October 2014 |
### Other Feature Updates

- At least 3 hours of log files are kept uncompressed on disk for better debugging. Older logs are removed with the exception of active application logs. These logs continue to remain uncompressed, on-disk unless they are rotated by log4j.
- Once they are compressed, log files are uploaded to Amazon S3 with special headers to allow browsing (if the raw log file size is less than 500MB). If the size is greater than 500MB, you are prompted to download the file.
- Compressed log files are now kept in a temporary directory in the same directory as the original log. If the log file size is less than 500MB, a special header is included in the compressed log files to allow browsing.
<table>
<thead>
<tr>
<th>AMI Version</th>
<th>Includes</th>
<th>Notes</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>log is in /mnt/ var/log/ hadoop, the compressed log is stored in /mnt/ tmp/mnt/ var/log/ hadoop until the log retention period expires.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Compressed log files larger than 4GB are not uploaded to Amazon S3.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A temporary directory cleaner is now included that cleans up temporary files in /mnt/tmp.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Major Bug Fixes</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fixes an issue where the Hive server does not start after installation.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fixes an issue where the Hive web interface does not function properly.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fixes an issue where the HBase restore procedure corruptions</td>
<td></td>
</tr>
<tr>
<td>AMI Version</td>
<td>Includes</td>
<td>Notes</td>
<td>Release Date</td>
</tr>
<tr>
<td>-------------</td>
<td>----------</td>
<td>-------</td>
<td>--------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the source cluster if it is still running.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Adds support for reuse of file statuses in the Pig Zebra format split calculation logic to improve performance.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Adds multithreaded creation of Pig Zebra format indexes during MapReduce job closure to improve performance.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Adds support for the legacy location of piggybank.jar: /home/hadoop/lib/pig/.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fixes an issue where Pig does not use automatic parallelism if the source file system is Amazon S3.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fixes an issue where S3DistCp ignores all -D CLI parameters.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Includes the following patches: YARN-2008, YARN-1857, YARN-1198, YARN-1680,</td>
<td></td>
</tr>
<tr>
<td>AMI Version</td>
<td>Includes</td>
<td>Notes</td>
<td>Release Date</td>
</tr>
<tr>
<td>-------------</td>
<td>----------</td>
<td>-------</td>
<td>--------------</td>
</tr>
</tbody>
</table>
| 3.2.1       | • Amazon Linux version 2014.03  
              • Hadoop 2.4.0  
              • Hive 0.13.1  
              • Pig 0.12  
              • HBase 0.94.18  
              • Impala 1.2.4  
              • Mahout 0.9  
              • Java: Oracle/Sun jdk-7u65  
              • Perl 5.16.3  
              • PHP 5.3.28  
              • Python 2.6.9  
              • R 3.0.2  
              • Ruby 2.0  
              • Scala 2.11.1 | This Amazon EMR AMI version provides the following fixes:  
**Major Bug Fixes**  
• Added EMRFS consistent view, which allows customers to track the consistent view of objects written by Amazon EMR to Amazon S3. For more information, see Consistent View (p. 46)  
**Other Bug Fixes**  
• Fixes a port forwarding issue encountered with Hive.  
• Fixes an issue encountered with HiveMetaStoreChecker.  
• Included a fix for: HIVE-7085. | 16 September 2014 |
## Choose an Amazon Machine Image (AMI)

<table>
<thead>
<tr>
<th>AMI Version</th>
<th>Includes</th>
<th>Notes</th>
<th>Release Date</th>
</tr>
</thead>
</table>
| 3.2.0       | - Amazon Linux version 2014.03  
- Hadoop 2.4.0  
- Hive 0.13.1  
- Pig 0.12  
- HBase 0.94.18  
- Impala 1.2.4  
- Mahout 0.9  
- Java: Oracle/Sun jdk-7u65  
- Perl 5.16.3  
- PHP 5.3.28  
- Python 2.6.9  
- R 3.0.2  
- Ruby 2.0  
- Scala 2.11.1 | This Amazon EMR AMI version provides the following features:  
- **Major Feature Updates**  
  - Added Apache Hive 0.13.1.  
  - Provided a change to the connector for Kinesis that takes a flag, `kinesis.iteration.timeout.ignore.failure`, to allow a job to continue checkpointing even if it has reached the timeout value.  
- **Major Bug Fixes**  
  - Included the following YARN patches: YARN-1718, YARN-1923, YARN-1889. | 3 September 2014 |
### Choose an Amazon Machine Image (AMI)

<table>
<thead>
<tr>
<th>AMI Version</th>
<th>Includes</th>
<th>Notes</th>
<th>Release Date</th>
</tr>
</thead>
</table>
| 3.1.4       | • AWS SDK for Ruby 1.9  
               • Amazon Linux version 2014.03  
               • Hadoop 2.4.0  
               • Hive 0.11.0.2  
               • Pig 0.12  
               • HBase 0.94.18  
               • Impala 1.2.4  
               • Mahout 0.9  
               • Java: Oracle/Sun jdk-7u65  
               • Perl 5.16.3  
               • PHP 5.3.28  
               • Python 2.6.9  
               • R 3.0.2  
               • Ruby 2.0  
               • Scala 2.11.1 | This Amazon EMR AMI version provides the following features and bug fixes:  
**Major Feature Updates**  
• EMRFS now supports Amazon S3 eventual consistency notifications for Amazon SQS and eventual consistency metrics for CloudWatch.  
• Performance optimizations for Amazon S3 multipart uploads with EMRFS.  
• The configure-hadoop bootstrap action now supports configuring log levels for different Hadoop daemons. You can now configure separate appenders for each daemon, and the following new identifiers are provided: HADOOP, MAPRED, JHS, and YARN. | 31 October 2014 |
## Choose an Amazon Machine Image (AMI)

<table>
<thead>
<tr>
<th>AMI Version</th>
<th>Includes</th>
<th>Notes</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><strong>Other Feature Updates</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• At least 3 hours of log files are kept uncompressed on disk for better debugging. Older logs are removed with the exception of active application logs. These logs continue to remain uncompressed, on-disk unless they are rotated by log4j.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Once they are compressed, log files are uploaded to Amazon S3 with special headers to allow browsing (if the raw log file size is less than 500MB). If the size is greater than 500MB, you are prompted to download the file.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Compressed log files are now kept in a temporary directory in the same directory as the original log. If the</td>
<td></td>
</tr>
</tbody>
</table>

---

98
<table>
<thead>
<tr>
<th>AMI Version</th>
<th>Includes</th>
<th>Notes</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>log is in /mnt/ var/log/hadoop, the compressed log is stored in /mnt/tmp/mnt/var/log/hadoop until the log retention period expires.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Compressed log files larger than 4GB are not uploaded to Amazon S3.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A temporary directory cleaner is now included that cleans up temporary files in /mnt/tmp.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Major Bug Fixes</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fixes an issue where Pig does not use automatic parallelism if the source file system is Amazon S3.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fixes an issue where the HBase restore procedure corrupts the source cluster if it is still running.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fixes an issue where</td>
<td></td>
</tr>
<tr>
<td>AMI Version</td>
<td>Includes</td>
<td>Notes</td>
<td>Release Date</td>
</tr>
<tr>
<td>-------------</td>
<td>----------</td>
<td>------------------------------------------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>S3DistCp ignores all <code>-D</code> CLI parameters.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Includes the following patches: YARN-2008, YARN-1857, YARN-1198,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>YARN-1680, MAPREDUCE-6111, HDFS-7005, HIVE-2777</td>
<td></td>
</tr>
<tr>
<td>AMI Version</td>
<td>Includes</td>
<td>Notes</td>
<td>Release Date</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>3.1.2</td>
<td>• Amazon Linux version 2014.03</td>
<td>This Amazon EMR AMI version provides the following bug fixes:</td>
<td>16 September 2014</td>
</tr>
<tr>
<td></td>
<td>• Hadoop 2.4.0</td>
<td><strong>Major Bug fixes</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Hive 0.11.0.2</td>
<td>• Fixes an issue encountered when using AWS CLI on the image.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Pig 0.12</td>
<td>• Fixes a port forwarding bug encountered with Hive.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• HBase 0.94.18</td>
<td>• Fixes an issue encountered with HiveMetaStoreChecker.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Mahout 0.9</td>
<td>• Includes the following patches:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Java: Oracle/Sun jdk-7u65</td>
<td>MAPREDUCE-5956, YARN-2026, YARN-2187, YARN-2214, YARN-2111, YARN-2053,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Perl 5.16.3</td>
<td>YARN-2074, YARN-2030, YARN-2155, YARN-2128, YARN-1913, YARN-596,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• PHP 5.3.28</td>
<td>YARN-2073, YARN-2017.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Python 2.6.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• R 3.0.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Ruby 2.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Scala 2.11.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AMI Version</td>
<td>Includes</td>
<td>Notes</td>
<td>Release Date</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>3.1.1</td>
<td>• Amazon Linux version 2014.03</td>
<td>This Amazon EMR AMI version provides the following bug fixes and enhancements:</td>
<td>15 August 2014</td>
</tr>
<tr>
<td></td>
<td>• Hadoop 2.4.0</td>
<td><strong>Major Feature Updates</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Hive 0.11.0.2</td>
<td>• Allows unlimited steps over the lifetime of the cluster with up to 256 ACTIVE or PENDING steps at a given time and display of up to 1,000 step records (including system steps). For more information: Submit Work to a Cluster (p. 493).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Pig 0.12</td>
<td>• Added Enhanced Networking for C3, R3, and I2 instance types. For more information, see the Enhanced Networking section in the AWS EC2 User Guide.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• HBase 0.94.18</td>
<td>• Updated Java version to 7u65. For more information, go to JDK 7.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Impala 1.2.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Mahout 0.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Java: Oracle/Sun jdk-7u65</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Perl 5.16.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• PHP 5.3.28</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Python 2.6.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• R 3.0.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Ruby 2.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Scala 2.11.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AMI Version</td>
<td>Includes</td>
<td>Notes</td>
<td>Release Date</td>
</tr>
<tr>
<td>-------------</td>
<td>----------</td>
<td>-------</td>
<td>--------------</td>
</tr>
<tr>
<td>Update 65 Release.</td>
<td>• Added support for AWS SDK 1.7.8 for all components except Impala. • Improved scalability of CloudWatch metrics support for Amazon EMR. • Enabled fuse_dfs. For more information, see the MountableHDFS website.</td>
<td>Major Bug Fixes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Fixed a bug that placed Application Master services on instances in the Task group, which may have resulted in the undesired termination of certain Application Master daemons. • Fixed a bug that prevented clusters from moving or copying files larger than 5 GB.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AMI Version</td>
<td>Includes</td>
<td>Notes</td>
<td>Release Date</td>
</tr>
<tr>
<td>-------------</td>
<td>----------</td>
<td>-------</td>
<td>--------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fixed a bug that prevented users from launching Hive in local mode.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fixed a bug that prevented NodeManager from using all available mountpoints, which resulted in issues using ephemeral drives on certain instances.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fixed an issue in ResourceManager, which prevented users from accessing user interfaces on localhost.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fixed a bug in JobHistory that may have prevented storage of JobHistory logs in Amazon S3 buckets.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Included the following Hadoop patches: HDFS-6701, HDFS-6460, HADOOP-10456, HDFS-6268, MAPREDUCE-5900.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Backport of YARN-1864 to Hadoop 2.</td>
<td></td>
</tr>
<tr>
<td>AMI Version</td>
<td>Includes</td>
<td>Notes</td>
<td>Release Date</td>
</tr>
<tr>
<td>-------------</td>
<td>----------</td>
<td>-------</td>
<td>--------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fixed a performance regression in Hive.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Hive is compiled against JDK 1.7</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Included the following Hive patches: HIVE-6938, HIVE-7429.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fixed several HBase bugs.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The connector for Kinesis for now supports all regions where Kinesis is available.</td>
<td></td>
</tr>
<tr>
<td>AMI Version</td>
<td>Includes</td>
<td>Notes</td>
<td>Release Date</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>3.1.0</td>
<td>• Amazon Linux version 2014.03</td>
<td>This Amazon EMR AMI version provides the following features:</td>
<td>15 May 2014</td>
</tr>
<tr>
<td></td>
<td>• Hadoop 2.4.0</td>
<td>Major Feature Updates</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Hive 0.11.0.2</td>
<td>- Significant updates and improvements to support new packages and Amazon EMR features: Hadoop 2.4.0, Pig 0.12, Impala 1.2.4, HBase 0.94.18, Mahout 0.9, and Ruby 2.0.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Pig 0.12</td>
<td>- Enabled Amazon S3 server side encryption with Hadoop.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• HBase 0.94.18</td>
<td>- For more information, see ??? (p. 62).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Hive 0.11.0.2</td>
<td>- Updated Java version to 7u60 (early access release).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Impala 1.2.4</td>
<td>- For more information, go to JDK 7 Update 60 Early Access Release.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Mahout 0.9</td>
<td>- Update Jetty to version 6.1.26.emr to fix Hadoop MapReduce</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Java: Oracle/Sun jdk-7u60</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Perl 5.16.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• PHP 5.3.28</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Python 2.6.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• R 3.0.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Ruby 2.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AMI Version</td>
<td>Includes</td>
<td>Notes</td>
<td>Release Date</td>
</tr>
<tr>
<td>-------------</td>
<td>----------</td>
<td>-------</td>
<td>--------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>issue MAPREDUCE-2980.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Changed the log level for Hive UDAFPercentile to DEBUG.</td>
<td></td>
</tr>
<tr>
<td>Major Bug Fixes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fixed an issue encountered when no log-uri value is specified at cluster creation.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fixed version utility to accurately display Amazon Hadoop Distribution version.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fixed Hadoop to accept HADOOP_NAMENODE_HEAPSIZE and HADOOP_DATANODE_HEAPSIZE memory setting values.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Replaced YARN_HEAPSIZE with YARNResourceManager_HEAPSIZE, YARN_Nodemanager_HEAPSIZE, and YARN_PROXYSERVER_HEAPSIZE to allow more granularity when configuring. For more information, see Configuration of hadoop-</td>
<td></td>
</tr>
<tr>
<td>AMI Version</td>
<td>Includes</td>
<td>Notes</td>
<td>Release Date</td>
</tr>
<tr>
<td>-------------</td>
<td>----------</td>
<td>-------</td>
<td>--------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>user-env.sh (p. 547),</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>memory</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>setting,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>HADOOP_JOB_HISTORYSERVER_HEAPSIZE.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fixed</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>an issue</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>encountered</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>with hdfs</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>-get when</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>used with an</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Amazon S3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>path.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fixed an</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>issue with</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>the HTTPFS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>service for</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Hadoop.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fixed an</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>issue that</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>caused job</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>failures after</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>a previous</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>job was killed.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Other</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>improvements</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>and bug</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>fixes.</td>
<td></td>
</tr>
<tr>
<td>AMI Version</td>
<td>Includes</td>
<td>Notes</td>
<td>Release Date</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>3.0.4</td>
<td>• Amazon Linux version 2013.09</td>
<td>This Amazon EMR AMI version provides the following features:</td>
<td>19 February 2014</td>
</tr>
<tr>
<td></td>
<td>• Hadoop 2.2.0</td>
<td>• Adds a connector for Amazon Kinesis, which allows users to process</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Hive 0.11.0.2</td>
<td>streaming data using standard Hadoop and ecosystem tools within</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Pig 0.11.1.1</td>
<td>Amazon EMR clusters. For more information, see Analyze Amazon</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• HBase 0.94.7</td>
<td>Kinesis Data (p. 358).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Impala 1.2.1</td>
<td>• Fixes an issue in the yarn-site.xml configuration file, which</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Mahout 0.8</td>
<td>resulted in the JobHistory server not being fully configured.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Java: Oracle/Sun jdk-7u60</td>
<td>• Adds support for AWS SDK 1.7.0.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Perl 5.10.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• PHP 5.3.28</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Python 2.6.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• R 3.0.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Ruby 1.8.7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Choose an Amazon Machine Image (AMI)

<table>
<thead>
<tr>
<th>AMI Version</th>
<th>Includes</th>
<th>Notes</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.0.3</td>
<td>• Amazon Linux version 2013.03</td>
<td>This Amazon EMR AMI version provides the following features:</td>
<td>11 February 2014</td>
</tr>
<tr>
<td></td>
<td>• Hadoop 2.2.0</td>
<td>• Adds support for AWS SDK 1.6.10.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Hive 0.11.0.2</td>
<td>• Upgrades HttpClient to version 4.2 to be compatible with AWS SDK 1.6.10.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Pig 0.11.1.1</td>
<td>• Fixes a problem related to orphaned Amazon EBS volumes.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• HBase 0.94.7</td>
<td>• Adds support for Hive 0.11.0.2.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Impala 1.2.1</td>
<td>• Upgrades Protobuf to version 2.5.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Mahout 0.8</td>
<td>Note: The upgrade to Protobuf 2.5 requires you to regenerate and recompile any of your Java code that was previously generated by</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Oracle/Sun jdk-7u45</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Perl 5.10.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• PHP 5.3.28</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Python 2.6.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• R 3.0.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Ruby 1.8.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AMI Version</td>
<td>Includes</td>
<td>Notes</td>
<td>Release Date</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>3.0.2</td>
<td>• Amazon Linux version 2013.03</td>
<td>Adds support for Impala 1.2.1 with Hadoop 2. For more information, see Impala (p. 300).</td>
<td>12 December 2013</td>
</tr>
<tr>
<td></td>
<td>• Hadoop 2.2.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Hive 0.11.0.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Pig 0.11.1.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• HBase 0.94.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Impala 1.2.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Mahout 0.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Oracle/Sun jdk-7u45</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Perl 5.10.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• PHP 5.3.28</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Python 2.6.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• R 3.0.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Ruby 1.8.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.0.1</td>
<td>• Amazon Linux version 2013.03</td>
<td>Adds support for viewing Hadoop 2 task attempt logs in the EMR console.</td>
<td>8 November 2013</td>
</tr>
<tr>
<td></td>
<td>• Hadoop 2.2.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Hive 0.11.0.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Pig 0.11.1.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• HBase 0.94.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Impala 1.2.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Mahout 0.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Oracle/Sun jdk-7u45</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Perl 5.10.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• PHP 5.3.28</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Python 2.6.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• R 3.0.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Ruby 1.8.7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Choose an Amazon Machine Image (AMI)

<table>
<thead>
<tr>
<th>AMI Version</th>
<th>Includes</th>
<th>Notes</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.0.0</td>
<td>• Amazon Linux version 2013.03</td>
<td></td>
<td>This new major Amazon EMR AMI version provides the following features:</td>
</tr>
<tr>
<td></td>
<td>• Hadoop 2.2.0</td>
<td></td>
<td>• This Amazon EMR AMI is based on the Amazon Linux Release 2012.09. For more information, see Amazon Linux AMI 2012.09 Release Notes.</td>
</tr>
<tr>
<td></td>
<td>• Hive 0.11.0.1</td>
<td></td>
<td>• Adds support for Hadoop 2.2.0. For more information, see Supported Hadoop Versions (p. 120).</td>
</tr>
<tr>
<td></td>
<td>• Pig 0.11.1.1</td>
<td></td>
<td>• Adds support for HBase 0.94.7. For more information, go to the Apache web site.</td>
</tr>
<tr>
<td></td>
<td>• HBase 0.94.7</td>
<td></td>
<td>• Adds Java 7 support for Hadoop, HBase, and Pig.</td>
</tr>
<tr>
<td></td>
<td>• Impala 1.2.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Mahout 0.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Oracle/Sun jdk-7u45</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Perl 5.10.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• PHP 5.3.28</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Python 2.6.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• R 3.0.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Ruby 1.8.7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Deprecated Hadoop 1 and Earlier AMIs

<table>
<thead>
<tr>
<th>AMI Version</th>
<th>Description</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.4.11</td>
<td>This Amazon EMR AMI version provides the following:</td>
<td>26 February 2015</td>
</tr>
<tr>
<td></td>
<td>• Minor performance-related bug fixes.</td>
<td></td>
</tr>
</tbody>
</table>
## Choose an Amazon Machine Image (AMI)

<table>
<thead>
<tr>
<th>AMI Version</th>
<th>Description</th>
<th>Release Date</th>
</tr>
</thead>
</table>
| 2.4.10      | This Amazon EMR AMI version provides the following bug fixes:  
- Fixes an issue that prevented logs being properly handled due to corrupted files or improper permissions.  
- Fixes an issue that may have caused a step to not complete properly. | 13 February 2015 |
| 2.4.9       | This Amazon EMR AMI version provides the following bug fixes:  
- Includes the patch for bash issues: CVE-2014-6271 and CVE-2014-7169.  
- Backports Hadoop patch MAPREDUCE-5877.  
- Fixes an issue with JobTracker where a successful fetch from a local reducer prevents a bad TaskTracker node from being excluded. | 31 October 2014 |
| 2.4.8       | This Amazon EMR AMI version provides the following feature and bug fixes:  
**Major Feature Update**  
- Allows unlimited steps over the lifetime of the cluster with up to 256 ACTIVE or PENDING steps at a given time and display of up to 1,000 step records (including system steps). For more information, see: Submit Work to a Cluster (p. 493).  
**Major Bug Fixes**  
- Fixes an issue with `hbase-user-env.sh`, which resulted in HBase ignoring any settings made by this script.  
- Fixes an issue with `s3distcp` where using the `.*` regular expression causes an error. | 16 September 2014 |
| 2.4.7       | In addition to other enhancements and bug fixes, this version of Amazon EMR AMI corrects the following problems:  
- Fixes an issue with logs stored in `/mnt/var/log`, which may consume all of the volume's disk space.  
- Fixes a deadlock issue encountered when adding steps. | 30 July 2014 |
| 2.4.6       | This Amazon EMR AMI version provides the following features:  
- Adds support for Cascading 2.5.  
- Adds support for new instance types.  
- Fixed a permissions issue with Hadoop.  
- Various other bug fixes and enhancements. | 15 May 2014 |
<table>
<thead>
<tr>
<th>AMI Version</th>
<th>Description</th>
<th>Release Date</th>
</tr>
</thead>
</table>
| 2.4.5       | This Amazon EMR AMI version provides the following features:  
  - Adds support for HVM AMIs in US East (N. Virginia), US West (Oregon), US West (N. California), EU (Ireland), Asia Pacific (Singapore), Asia Pacific (Sydney), Asia Pacific (Tokyo), and South America (São Paulo) regions.  
  - Adds support for AWS SDK 1.7.0  
  - Adds support for Python 2.7.  
  - Adds support for Hive 0.11.0.2.  
  - Upgrades Protobuf to version 2.5.  
  
  **Note**  
  The upgrade to Protobuf 2.5 requires you to regenerate and recompile any of your Java code that was previously generated by the protoc tool.  
  - Updates to Java version to 7u60 (early access release).  
    For more information, go to [JDK 7 Update 60 Early Access Release](#).  
  - Updates Jetty to version 6.1.26.emr.1 that fixes Hadoop MapReduce issue [MAPREDUCE-2980](#).  
  - Fixes an issue encountered when no log-uri is specified at cluster creation.  
  - Fixes version utility to accurately display Amazon Hadoop Distribution version.  
  - Other improvements and bug fixes. | 27 March 2014 |
| 2.4.3       | This Amazon EMR AMI version provides the following features:  
  - Adds support for Python 2.7.  
  - Updates Jetty to version 6.1.26.emr.1 that fixes the Hadoop MapReduce issue [MAPREDUCE-2980](#).  
  - Updates to Java version to 7u60 (early access release).  
    For more information, go to [JDK 7 Update 60 Early Access Release](#).  
  - Adds support for Hive 0.11.0.2.  
  - Upgrades Protobuf to version 2.5.  
  
  **Note**  
  The upgrade to Protobuf 2.5 requires you to regenerate and recompile any of your Java code that was previously generated by the protoc tool. | 3 January 2014 |
<table>
<thead>
<tr>
<th>AMI Version</th>
<th>Description</th>
<th>Release Date</th>
</tr>
</thead>
</table>
| 2.4.2       | Same as the previous AMI version, with the following additions:  
- Fixed a bug in host resolution that limited map-side local data optimization. Customers who use Fair Scheduler may observe a change in job execution due to the emphasis the system puts on data locality. The schedule may now hold back tasks to run them locally.  
- Includes Hadoop 1.0.3, Java 1.7, Perl 5.10.1, Python 2.6.6, and R 2.11 | 7 October 2013 |
| 2.4.1       | Same as the previous AMI version, with the following additions:  
- Fixes a bug that causes the HBase shell not to work properly.  
- Fixes a bug that causes some clusters to fail with the error 'concurrent modifications exception'.  
- Adds new logic in the instance controller to detect and reboot instances that have been blacklisted by Hadoop for an extended period of time.  
- Includes Hadoop 1.0.3, Java 1.7, Perl 5.10.1, Python 2.6.6, and R 2.11 | 20 August 2013 |
| 2.4         | Same as the previous AMI version, with the following additions:  
- Adds support for Java 7 with Hadoop and HBase. Other Amazon EMR features, such as Hive and Pig, continue to require Java 6.  
- Improved JobTracker detection and response time when reducers become stuck due to a problematic mapper.  
- Fixes a problem that some Hadoop reducers are unable to fetch map output data due to a bad mapper, causing job delays.  
- Adds FetchStatusMap to keep track of all fetch errors and success along with their time stamp.  
- Fixes a problem with "Text File Busy" errors when launching tasks. For more information, go to MAPREDUCE-2374. | 1 August 2013 |
| 2.3.6       | Same as 2.3.5, with the following additions:  
- Fixes a problem in the Debian sources.lst and preferences files that caused certain bootstrap actions to fail, including Ganglia. Customers using AMI versions 2.0.0 to 2.3.5 may notice an additional bootstrap action in their list named EMR Debian Patch. | 17 May 2013 |
<table>
<thead>
<tr>
<th>AMI Version</th>
<th>Description</th>
<th>Release Date</th>
</tr>
</thead>
</table>
| 2.3.5       | Same as 2.3.3, with the following additions:  
- Fixes an S3DistCp bug which created invalid manifest file entries for certain URL encoded file names.  
- Improves log pushing functionality and adds a 7 day retention policy for on-cluster log files. Log files not modified for 7 or more days are deleted from the cluster.  
- Adds a streaming configuration option for not emitting the mapper key. For more information, go to MAPREDUCE-1785.  
- Adds the `--s3ServerSideEncryption` option to the S3DistCp tool. For more information, see S3DistCp Options (p. 387). | 26 April 2013 |
| 2.3.4       | **Note**  
Because of an issue with AMI 2.3.4, this version is deprecated. We recommend that you use a different AMI version instead. | 16 April 2013 |
| 2.3.3       | Same as 2.3.2, with the following additions:  
- Improved CloudWatch LiveTaskTracker metric to take into account expired Hadoop TaskTrackers and minor improvements in Hadoop. | 01 March 2013 |
| 2.3.2       | Same as 2.3.1, with the following additions:  
- Fixes an issue which prevented customers from using the debugging feature in the Amazon EMR console. | 07 February 2013 |
| 2.3.1       | Same as 2.3.0, with the following additions:  
- Improves support for clusters running on hs1.8xlarge instances. | 24 December 2012 |
| 2.3.0       | Same as 2.2.4, with the following additions:  
- Adds support for IAM roles. For more information, see Configure IAM Roles for Amazon EMR Permissions to AWS Services (p. 234). | 20 December 2012 |
| 2.2.4       | Same as 2.2.3, with the following additions:  
- Improves error handling in the Snappy decompressor. For more information, go to HADOOP-8151.  
- Fixes an issue with MapFile.Reader reading LZO or Snappy compressed files. For more information, go to HADOOP-8423.  
- Updates the kernel to the AWS version of 3.2.30-49.59. | 6 December 2012 |
<table>
<thead>
<tr>
<th>AMI Version</th>
<th>Description</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2.3</td>
<td>Same as 2.2.1, with the following additions:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Improves HBase backup functionality.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Updates the AWS SDK for Java to version 1.3.23.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Resolves issues with the job tracker user interface.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Improves Amazon S3 file system handling in Hadoop.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Improves NameNode functionality in Hadoop.</td>
<td>30 November 2012</td>
</tr>
<tr>
<td>2.2.2</td>
<td><strong>Note</strong> Because of an issue with AMI 2.2.2, this version is deprecated. We recommend that you use a different AMI version instead.</td>
<td>23 November 2012</td>
</tr>
<tr>
<td>2.2.1</td>
<td>Same as 2.2.0, with the following additions:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Fixes an issue with HBase backup functionality.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Enables multipart upload by default for files larger than the Amazon S3 block size specified by <code>fs.s3n.blockSize</code>. For more information, see [Configure Multipart Upload for Amazon S3](p. 33).</td>
<td>30 August 2012</td>
</tr>
<tr>
<td>2.2.0</td>
<td>Same as 2.1.3, with the following additions:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Adds support for Hadoop 1.0.3.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• No longer includes Hadoop 0.18 and Hadoop 0.20.205.</td>
<td>6 August 2012</td>
</tr>
<tr>
<td></td>
<td><strong>Operating system:</strong> Debian 6.0.5 (Squeeze)</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Applications:</strong> Hadoop 1.0.3, Hive 0.8.1.3, Pig 0.9.2.2, HBase 0.92.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Languages:</strong> Perl 5.10.1, PHP 5.3.3, Python 2.6.6, R 2.11.1, Ruby 1.8.7</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>File system:</strong> ext3 for root, xfs for ephemeral</td>
<td></td>
</tr>
<tr>
<td>2.1.4</td>
<td>Same as 2.1.3, with the following additions:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Fixes issues in the Native Amazon S3 file system.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Enables multipart upload by default. For more information, see [Configure Multipart Upload for Amazon S3](p. 33).</td>
<td>30 August 2012</td>
</tr>
<tr>
<td>2.1.3</td>
<td>Same as 2.1.2, with the following additions:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Fixes issues in HBase.</td>
<td>6 August 2012</td>
</tr>
<tr>
<td>2.1.2</td>
<td>Same as 2.1.1, with the following additions:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Support for CloudWatch metrics when using MapR.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Improve reliability of reporting metrics to CloudWatch.</td>
<td>6 August 2012</td>
</tr>
<tr>
<td>AMI Version</td>
<td>Description</td>
<td>Release Date</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
<td>--------------</td>
</tr>
</tbody>
</table>
| 2.1.1       | Same as 2.1.0, with the following additions:  
- Improves the reliability of log pushing.  
- Adds support for HBase in Amazon VPC.  
- Improves DNS retry functionality. | 3 July 2012 |
| 2.1.0       | Same as AMI 2.0.5, with the following additions:  
- Supports launching HBase clusters. For more information see [Apache HBase](p. 325).  
- Supports running MapR Edition M3 and Edition M5. For more information, see [Using the MapR Distribution for Hadoop](p. 177).  
- Enables HDFS append by default; `dfs.support.append` is set to `true` in `hdfs/hdfs-default.xml`. The default value in code is also set to `true`.  
- Fixes a race condition in instance controller.  
- Changes `mapreduce.user.classpath.first` to default to `true`. This configuration setting indicates whether to load classes first from the cluster's JAR file or the Hadoop system lib directory. This change was made to provide a way for you to easily override classes in Hadoop.  
- Uses Debian 6.0.5 (Squeeze) as the operating system. | 12 June 2012 |
| 2.0.5       | **Note**  
Because of an issue with AMI 2.0.5, this version is deprecated. We recommend that you use a different AMI version instead.  
Same as AMI 2.0.4, with the following additions:  
- Improves Hadoop performance by reinitializing the recycled compressor object for mappers only if they are configured to use the GZip compression codec for output.  
- Adds a configuration variable to Hadoop called `mapreduce.jobtracker.system.dir.permission` that can be used to set permissions on the system directory.  
- Changes InstanceController to use an embedded database rather than the MySQL instance running on the box. MySQL remains installed and running by default.  
- Improves the collectd configuration. For more information about collectd, go to [http://collectd.org/](http://collectd.org/).  
- Fixes a rare race condition in InstanceController.  
- Changes the default shell from dash to bash.  
- Uses Debian 6.0.4 (Squeeze) as the operating system. | 19 April 2012 |
### AMI Version

<table>
<thead>
<tr>
<th>AMI Version</th>
<th>Description</th>
<th>Release Date</th>
</tr>
</thead>
</table>
| 2.0.4       | Same as AMI 2.0.3, with the following additions:  
- Changes the default for `fs.s3n.blockSize` to 33554432 (32MiB).  
- Fixes a bug in reading zero-length files from Amazon S3. | 30 January 2012 |
| 2.0.3       | Same as AMI 2.0.2, with the following additions:  
- Adds support for Amazon EMR metrics in CloudWatch.  
- Improves performance of seek operations in Amazon S3. | 24 January 2012 |
| 2.0.2       | Same as AMI 2.0.1, with the following additions:  
- Adds support for the Python API Dumbo. For more information about Dumbo, go to [https://github.com/klbostee/dumbo/wiki/](https://github.com/klbostee/dumbo/wiki/).  
- Updates the Amazon Web Services SDK to version 1.2.16.  
- Improves the way Amazon S3 file system initialization checks for the existence of Amazon S3 buckets.  
- Adds support for configuring the Amazon S3 block size to facilitate splitting files in Amazon S3. You set this in the `fs.s3n.blockSize` parameter. You set this parameter by using the configure-hadoop bootstrap action. The default value is 9223372036854775807 (8 EiB).  
- Adds a `/dev/sd` symlink for each `/dev/xvd` device. For example, `/dev/xvdb` now has a symlink pointing to it called `/dev/sdb`. Now you can use the same device names for AMI 1.0 and 2.0. | 17 January 2012 |
| 2.0.1       | Same as AMI 2.0 except for the following bug fixes:  
- Task attempt logs are pushed to Amazon S3.  
- Fixed `/mnt` mounting on 32-bit AMIs.  
- Uses Debian 6.0.3 (Squeeze) as the operating system. | 19 December 2011 |
| 2.0.0       | **Operating system**: Debian 6.0.2 (Squeeze)  
**Applications**: Hadoop 0.20.205, Hive 0.7.1, Pig 0.9.1  
**Languages**: Perl 5.10.1, PHP 5.3.3, Python 2.6.6, R 2.11.1, Ruby 1.8.7  
**File system**: ext3 for root, xfs for ephemeral  
**Note**: Added support for the Snappy compression/decompression library. | 11 December 2011 |
Choose a Version of Hadoop

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

The AWS version of Hadoop installed by Amazon EMR is based on Apache Hadoop, with patches and improvements added that make it work efficiently with AWS. Each Amazon EMR AMI has a default version of Hadoop associated with it. We recommend that you launch a cluster with the latest AMI version, running the default version of Hadoop whenever possible, as this gives you access to the most recent features and latest bug fixes.

If your application requires a different version of Hadoop than the default, you can specify that version of Hadoop when you launch the cluster. You can also choose to launch an Amazon EMR cluster using a MapR distribution of Hadoop. For more information, see Using the MapR Distribution for Hadoop (p. 177).

Topics

- Supported Hadoop Versions (p. 120)
- How Does Amazon EMR Hadoop Differ from Apache Hadoop? (p. 125)
- Hadoop Patches Applied in Amazon EMR (p. 125)
- Supported Mahout Versions (p. 128)

Supported Hadoop Versions

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

<table>
<thead>
<tr>
<th>AMI Version</th>
<th>Description</th>
<th>Release Date</th>
</tr>
</thead>
</table>
| 1.0.1       | Same as AMI 1.0 except for the following change:  
• Updates sources.list to the new location of the Lenny distribution in archive.debian.org. | 3 April 2012 |
| 1.0.0       | **Operating system:** Debian 5.0 (Lenny)  
**Applications:** Hadoop 0.20 and 0.18 (default); Hive 0.5, 0.7 (default), 0.7.1; Pig 0.3 (on Hadoop 0.18), 0.6 (on Hadoop 0.20)  
**Languages:** Perl 5.10.0, PHP 5.2.6, Python 2.5.2, R 2.7.1, Ruby 1.8.7  
**File system:** ext3 for root and ephemeral  
**Kernel:** Red Hat  
**Note:** This was the last AMI released before the CLI was updated to support AMI versioning. For backward compatibility, job flows launched with versions of the CLI downloaded before 11 December 2011 use this version. | 26 April 2011 |
Amazon EMR allows you to choose which version of Hadoop to run. You do this using the CLI and setting the `--ami-version` as shown in the following table. We recommend using the latest version of Hadoop to take advantage of performance enhancements and new functionality.

**Note**
The AMI version determines the Hadoop version and the `--hadoop-version` parameter is no longer supported.

<table>
<thead>
<tr>
<th>Hadoop Version</th>
<th>AMI Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.4.0</td>
<td><code>--ami-version 3.1.X and higher (all 3.1 and 3.2 versions)</code></td>
</tr>
<tr>
<td>2.2.0</td>
<td><code>--ami-version 3.0.X (all 3.0 versions)</code></td>
</tr>
<tr>
<td>1.0.3</td>
<td><code>--ami-version 2.2.X, 2.3.X, 2.4.X (all 2.2, 2.3, and 2.4 versions)</code></td>
</tr>
<tr>
<td>0.20.205</td>
<td><code>--ami-version 2.1.4</code></td>
</tr>
<tr>
<td>0.20</td>
<td><code>--ami-version 1.0</code></td>
</tr>
</tbody>
</table>

For details about the default configuration and software available on AMIs used by Amazon EMR see [Choose an Amazon Machine Image (AMI)](p. 69).

**Note**
The Asia Pacific (Sydney) region and AWS GovCloud (US) regions support only Hadoop 1.0.3 and later. AWS GovCloud (US) additionally requires AMI 2.3.0 or later.

**To specify the Hadoop version using the AWS CLI**

To specify the Hadoop version using the AWS CLI, type the `create-cluster` subcommand with the `--ami-version` parameter. The AMI version determines the version of Hadoop for Amazon EMR to use. For details about the version of Hadoop available on an AMI, see [AMI Versions Supported in Amazon EMR Versions 2.x and 3.x](p. 73).

- To launch a cluster running Hadoop 2.4.0 using AMI version 3.3, type the following command and replace `myKey` with the name of your EC2 key pair.

  ```bash
  aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applications Name=Hue Name=Hive Name=Pig \ --use-default-roles --ec2-attributes KeyName=myKey \ --instance-count 5 --instance-type m3.xlarge
  ```

- Linux, UNIX, and Mac OS X users:

  ```bash
  aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applications Name=Hue Name=Hive Name=Pig \ --use-default-roles --ec2-attributes KeyName=myKey \ --instance-count 5 --instance-type m3.xlarge
  ```

- Windows users:

  ```bash
  aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applications Name=Hue Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey -- instance-count 5 --instance-type m3.xlarge
  ```

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

**Note**
If you have not previously created the default EMR service role and EC2 instance profile, type `aws emr create-default-roles` to create them before typing the `create-cluster` subcommand.
For more information on using Amazon EMR commands in the AWS CLI, see http://docs.aws.amazon.com/cli/latest/reference/emr.

Hadoop 2.4.0 New Features

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Hadoop 2.4.0 enhancements were primarily focused on HDFS and YARN. Please see below for the release highlights:

**HDFS**

- Full HTTPS support for HDFS (HDFS-5305)
- Support for Access Control Lists (ACL) in HDFS (HDFS-4685)
- Usage of protocol-buffers for HDFS FSImage for smooth operational upgrades (HDFS-5698)

**YARN**

- Enhanced support for new applications on YARN with Application History Server (YARN-321) and Application Timeline Server (YARN-1530)
- Support for strong SLAs in YARN CapacityScheduler via Preemption (YARN-185)

For a full list of new features and fixes available in Hadoop 2.4.0, see the Hadoop 2.4.0 Release Notes.

The following change included in Hadoop 2.3.0 are relevant to Amazon EMR customers:

- Heterogeneous Storage for HDFS (HDFS-2832)
- In-memory Cache for data resident in HDFS via DataNode (HDFS-4949)

**Note**

While Amazon EMR generally supports the features listed in Hadoop Common Releases, the following features are not supported in this Amazon release of Hadoop 2.4.0:

- Native support for Rolling Upgrades in HDFS Rolling upgrades
- HDFS Federation and HDFS NameNode High Availability (HA)
- Support for Automatic Failover of the YARN ResourceManager (YARN-149)

**Amazon EMR Enhancements**

The following enhancements are available with AMI 3.1.0 and later:

- Customers can now see their job logs in “Log folder S3 location” under subfolder “jobs” when logging and debugging are enabled. For more information about logging, see Configure Cluster Logging and Debugging (p. 167).
- Today, customers can specify an Amazon S3 bucket where task attempt logs are redirected. However, these logs are stored at the individual attempt level, which means a job might produce thousands of log files. Customers can now aggregate all of their task attempt logs to a smaller number of files to make it easy to view and analyze these logs. For more information about how to enable log aggregation, see the section called “Archive Log Files to Amazon S3” (p. 168).
Hadoop 2.2.0 New Features

Hadoop 2.2.0 supports the following new features:

- MapReduce NextGen (YARN) resource management system as a general big data processing platform. YARN is a new architecture that divides the two major functions of the JobTracker (resource management and job life-cycle management) into separate components and introduces a new ResourceManager. For more information, go to Apache Hadoop NextGen MapReduce (YARN) and View Web Interfaces Hosted on Amazon EMR Clusters (p. 461).
- Pluggable Shuffle and Pluggable Sort. For more information, go to Pluggable Shuffle and Pluggable Sort.
- Capacity Scheduler and Fair Scheduler. For more information, go to Capacity Scheduler and Fair Scheduler.
- Hadoop Distributed File System (HDFS) snapshots. For more information, go to HDFS Snapshots.
- Performance-related enhancements such as short-circuit read. For more information, go to HDFS Short-Circuit Local Reads.
- Distributed job life cycle management by the application master
- Various security improvements

For a full list of new features and fixes available in Hadoop 2.2.0, go to Hadoop 2.2.0 Release Notes.

Note
While Amazon EMR generally supports the features listed in Hadoop Common Releases, HDFS Federation and HDFS name node High Availability (HA) are not supported in this Amazon release of Hadoop 2.2.0. In addition, Hadoop 2.2.0 is not supported on m1.small instances.

Major Changes from Hadoop 1 to Hadoop 2

Current Hadoop 1 users should take notice of several major changes introduced in Hadoop 2:

- Updated Hadoop user interfaces with new URLs, including a new ResourceManager. For more information, see View Web Interfaces Hosted on Amazon EMR Clusters (p. 461).
- Updated Hadoop configuration files. For more information, see JSON Configuration Files (p. 543).
- Changes to bootstrap actions that configure Hadoop daemons. For more information, see (Optional) Create Bootstrap Actions to Install Additional Software (p. 129).

Considerations for Moving to Hadoop 2.2.0

General availability (GA) for Hadoop 2.2 was announced on October 16, 2013. According to the Apache Software Foundation, Hadoop 2.2 "has achieved the level of stability and enterprise-readiness to earn the General Availability designation." Amazon recommends that customers continue running mission-
critical applications on Hadoop 1.0.3 and make the switch only after carefully testing their applications on Hadoop 2.2. In general, the Hadoop community is moving from Hadoop 1 to Hadoop 2.

Hadoop 1.0 New Features

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Hadoop 1.0.3 support in Amazon EMR includes the features listed in Hadoop Common Releases, including:

- A RESTful API to HDFS, providing a complete FileSystem implementation for accessing HDFS over HTTP.
- Support for executing new writes in HBase while an hflush/sync is in progress.
- Performance-enhanced access to local files for HBase.
- The ability to run Hadoop, Hive, and Pig jobs as another user, similar to the following:

  ```
  $ export HADOOP_USER_NAME=usernamehere
  
  By exporting the `HADOOP_USER_NAME` environment variable the job would then be executed by the specified username.
  
  **Note**
  If HDFS is used then you need to either change the permissions on HDFS to allow READ and WRITE access to the specified username or you can disable permission checks on HDFS. This is done by setting the configuration variable `dfs.permissions` to `false` in the `mapred-site.xml` file and then restarting the namenodes, similar to the following:
  ```xml
  <property>
    <name>dfs.permissions</name>
    <value>false</value>
  </property>
  
  S3 file split size variable renamed from `fs.s3.blockSize` to `fs.s3.block.size`, and the default is set to 64 MB. This is for consistency with the variable name added in patch HADOOP-5861.

Setting access permissions on files written to Amazon S3 is also supported in Hadoop 1.0.3 with Amazon EMR. For more information see How to write data to an Amazon S3 bucket you don't own (p. 42).

For a list of the patches applied to the Amazon EMR version of Hadoop 1.0.3, see Hadoop 1.0.3 Patches (p. 126).

Hadoop 0.20 New Features

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Hadoop 0.18 was not designed to efficiently handle multiple small files. The following enhancements in Hadoop 0.20 and later improve the performance of processing small files:

- Hadoop 0.20 and later assigns multiple tasks per heartbeat. A heartbeat is a method that periodically checks to see if the client is still alive. By assigning multiple tasks, Hadoop can distribute tasks to slave nodes faster, thereby improving performance. The time taken to distribute tasks is an important part of the processing time usage.
Choose a Version of Hadoop

• Historically, Hadoop processes each task in its own Java Virtual Machine (JVM). If you have many small files that take only a second to process, the overhead is great when you start a JVM for each task. Hadoop 0.20 and later can share one JVM for multiple tasks, thus significantly improving your processing time.

• Hadoop 0.20 and later allows you to process multiple files in a single map task, which reduces the overhead associated with setting up a task. A single task can now process multiple small files.

Hadoop 0.20 and later also supports the following features:

• A new command line option, \-libjars, enables you to include a specified JAR file in the class path of every task.

• The ability to skip individual records rather than entire files. In previous versions of Hadoop, failures in record processing caused the entire file containing the bad record to skip. Jobs that previously failed can now return partial results.

In addition to the Hadoop 0.18 streaming parameters, Hadoop 0.20 and later introduces the three new streaming parameters listed in the following table:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>-files</td>
<td>Specifies comma-separated files to copy to the map reduce cluster.</td>
</tr>
<tr>
<td>-archives</td>
<td>Specifies comma-separated archives to restore to the compute machines.</td>
</tr>
<tr>
<td>-D</td>
<td>Specifies a value for the key you enter, in the form of &lt;key&gt;=&lt;value&gt;.</td>
</tr>
</tbody>
</table>

For a list of the patches applied to the Amazon EMR version of Hadoop 0.20.205, see Hadoop 0.20.205 Patches (p. 126).

How Does Amazon EMR Hadoop Differ from Apache Hadoop?

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

The AWS version of Hadoop installed when you launch an Amazon EMR cluster is based on Apache Hadoop, but has had several patches and improvements added to make it work efficiently on AWS. Where appropriate, improvements written by the Amazon EMR team have been submitted to the Apache Hadoop code base. For more information about the patches applied to AWS Hadoop, see Hadoop Patches Applied in Amazon EMR (p. 125).

Hadoop Patches Applied in Amazon EMR

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

The following sections detail the patches the Amazon EMR team has applied to the Hadoop versions loaded on Amazon EMR AMIs.

Topics

• Hadoop 1.0.3 Patches (p. 126)
• Hadoop 0.20.205 Patches (p. 126)
Hadoop 1.0.3 Patches

The Amazon EMR team has applied the following patches to Hadoop 1.0.3 on the Amazon EMR AMI version 2.2.

<table>
<thead>
<tr>
<th>Patch</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>All of the patches applied to the Amazon EMR version of Hadoop 0.20.205.</td>
<td>See Hadoop 0.20.205 Patches (p. 126) for details.</td>
</tr>
<tr>
<td>HADOOP-5861</td>
<td>Files stored on the native Amazon S3 file system, those with URLs of the form s3n://, now report a block size determined by fs.s3n.block.size. For more information, go to <a href="https://issues.apache.org/jira/browse/HADOOP-5861">https://issues.apache.org/jira/browse/HADOOP-5861</a>.</td>
</tr>
<tr>
<td>Status: Fixed</td>
<td></td>
</tr>
<tr>
<td>Fixed in AWS Hadoop Version: 1.0.3</td>
<td></td>
</tr>
<tr>
<td>Fixed in Apache Hadoop Version: 0.21.0</td>
<td></td>
</tr>
<tr>
<td>HADOOP-6346</td>
<td>Supports specifying a pattern to RunJar.unJar that determines which files are unpacked. For more information, go to <a href="https://issues.apache.org/jira/browse/HADOOP-6346">https://issues.apache.org/jira/browse/HADOOP-6346</a>.</td>
</tr>
<tr>
<td>Status: Fixed</td>
<td></td>
</tr>
<tr>
<td>Fixed in AWS Hadoop Version: 1.0.3</td>
<td></td>
</tr>
<tr>
<td>Fixed in Apache Hadoop Version: 0.21.0</td>
<td></td>
</tr>
<tr>
<td>MAPREDUCE-967</td>
<td>Changes the TaskTracker node so it does not fully unjar job jars into the job cache directory. For more information, go to <a href="https://issues.apache.org/jira/browse/MAPREDUCE-967">https://issues.apache.org/jira/browse/MAPREDUCE-967</a>.</td>
</tr>
<tr>
<td>Status: Fixed</td>
<td></td>
</tr>
<tr>
<td>Fixed in AWS Hadoop Version: 1.0.3</td>
<td></td>
</tr>
<tr>
<td>Fixed in Apache Hadoop Version: 0.21.0</td>
<td></td>
</tr>
<tr>
<td>MAPREDUCE-2219</td>
<td>Changes the JobTracker service to remove the contents of mapred.system.dir during startup instead of removing the directory itself. For more information, go to <a href="https://issues.apache.org/jira/browse/MAPREDUCE-2219">https://issues.apache.org/jira/browse/MAPREDUCE-2219</a>.</td>
</tr>
<tr>
<td>Status: Fixed</td>
<td></td>
</tr>
<tr>
<td>Fixed in AWS Hadoop Version: 1.0.3</td>
<td></td>
</tr>
<tr>
<td>Fixed in Apache Hadoop Version: 0.22.0</td>
<td></td>
</tr>
</tbody>
</table>

Hadoop 0.20.205 Patches

The Amazon EMR team has applied the following patches to Hadoop 0.20.205 on the Amazon EMR AMI version 2.0.
<table>
<thead>
<tr>
<th>Patch</th>
<th>Description</th>
</tr>
</thead>
</table>
| Add hadoop-lzo | Install the hadoop-lzo third-party package. For more information about hadoop-lzo, go to [https://github.com/kevinweil/hadoop-lzo](https://github.com/kevinweil/hadoop-lzo)  
**Status:** Third-party Package  
**Fixed in AWS Hadoop Version:** 0.20.205  
**Fixed in Apache Hadoop Version:** n/a |
| Install the hadoop-snappy library | Add the hadoop-snappy library to provide access to the snappy compression. For more information about this library, go to [http://code.google.com/p/hadoop-snappy/](http://code.google.com/p/hadoop-snappy/).  
**Status:** Third-party Library  
**Fixed in AWS Hadoop Version:** 0.20.205  
**Fixed in Apache Hadoop Version:** n/a |
| MAPREDUCE-1597/2021/2046 | Fixes to how CombineFileInputFormat handles split locations and files that can be split. For more information about these patches, go to [https://issues.apache.org/jira/browse/MAPREDUCE-1597](https://issues.apache.org/jira/browse/MAPREDUCE-1597), [https://issues.apache.org/jira/browse/MAPREDUCE-2021](https://issues.apache.org/jira/browse/MAPREDUCE-2021), and [https://issues.apache.org/jira/browse/MAPREDUCE-2046](https://issues.apache.org/jira/browse/MAPREDUCE-2046).  
**Status:** Resolved, Fixed  
**Fixed in AWS Hadoop Version:** 0.20.205  
**Fixed in Apache Hadoop Version:** 0.22.0 |
| HADOOP-6436 | Remove the files generated by automake and autoconf of the native build and use the host's automake and autoconf to generate the files instead. For more information about this patch, go to [https://issues.apache.org/jira/browse/HADOOP-6436](https://issues.apache.org/jira/browse/HADOOP-6436).  
**Status:** Closed, Fixed  
**Fixed in AWS Hadoop Version:** 0.20.205  
**Fixed in Apache Hadoop Version:** 0.22.0, 0.23.0 |
| MAPREDUCE-2185 | Prevent an infinite loop from occurring when creating splits using CombineFileInputFormat. For more information about this patch, go to [https://issues.apache.org/jira/browse/MAPREDUCE-2185](https://issues.apache.org/jira/browse/MAPREDUCE-2185).  
**Status:** Closed, Fixed  
**Fixed in AWS Hadoop Version:** 0.20.205  
**Fixed in Apache Hadoop Version:** 0.23.0 |
| HADOOP-7082 | Change Configuration.writeXML to not hold a lock while outputting. For more information about this patch, go to [https://issues.apache.org/jira/browse/HADOOP-7082](https://issues.apache.org/jira/browse/HADOOP-7082).  
**Status:** Resolved, Fixed |
<table>
<thead>
<tr>
<th>Patch</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP-7015</td>
<td>Update <code>RawLocalFileSystem#listStatus</code> to deal with a directory that has changing entries, as in a multi-threaded or multi-process environment. For more information about this patch, go to <a href="https://issues.apache.org/jira/browse/HADOOP-7015">https://issues.apache.org/jira/browse/HADOOP-7015</a>.</td>
</tr>
<tr>
<td></td>
<td><strong>Status:</strong> Closed, Fixed</td>
</tr>
<tr>
<td></td>
<td><strong>Fixed in AWS Hadoop Version:</strong> 0.20.205</td>
</tr>
<tr>
<td></td>
<td><strong>Fixed in Apache Hadoop Version:</strong> 0.22.0</td>
</tr>
<tr>
<td>HADOOP-4675</td>
<td>Update the Ganglia metrics to be compatible with Ganglia 3.1. For more information about this patch go to <a href="https://issues.apache.org/jira/browse/HADOOP-4675">https://issues.apache.org/jira/browse/HADOOP-4675</a>.</td>
</tr>
<tr>
<td></td>
<td><strong>Status:</strong> Resolved, Fixed</td>
</tr>
<tr>
<td></td>
<td><strong>Fixed in AWS Hadoop Version:</strong> 0.20.205</td>
</tr>
<tr>
<td></td>
<td><strong>Fixed in Apache Hadoop Version:</strong> 0.22.0</td>
</tr>
</tbody>
</table>

### Supported Mahout Versions

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Amazon EMR currently supports the following Apache Mahout versions:

<table>
<thead>
<tr>
<th>Mahout Version</th>
<th>AMI Version</th>
<th>Mahout Version Details</th>
</tr>
</thead>
</table>
| 0.9            | 3.1.0 and later | • New and improved Mahout website based on Apache CMS (MAHOUT-1245)  
• Early implementation of a Multi-Layer Perceptron (MLP) classifier (MAHOUT-1265)  
• Scala DSL Bindings for Mahout Math Linear Algebra (MAHOUT-1297)  
• Recommenders as Search (MAHOUT-1288)  
• Support for easy functional Matrix views and derivatives (MAHOUT-1300)  
• JSON output format for ClusterDumper (MAHOUT-1343)  
• Enabled randomised testing for all Mahout modules using  

(Optional) Create Bootstrap Actions to Install Additional Software

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

You can use a bootstrap action to install additional software and to change the configuration of applications on the cluster. Bootstrap actions are scripts that are run on the cluster nodes when Amazon EMR launches the cluster. They run before Hadoop starts and before the node begins processing data. You can create custom bootstrap actions, or use predefined bootstrap actions provided by Amazon EMR. A common use of bootstrap actions is to change the Hadoop configuration settings.

Contents
- Bootstrap Action Basics (p. 129)
- Use Predefined Bootstrap Actions (p. 130)
- Shutdown Actions (p. 136)
- Use Custom Bootstrap Actions (p. 137)

Bootstrap Action Basics

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

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

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

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

AWS CLI

<table>
<thead>
<tr>
<th>Mahout Version</th>
<th>AMI Version</th>
<th>Mahout Version Details</th>
</tr>
</thead>
</table>
| 0.8            | 3.0-3.0.4   | Carrot RandomizedRunner (MAHOUT-1345)  
  • Online Algorithm for computing accurate Quantiles using 1-dimensional Clustering (MAHOUT-1361)  
  • Upgrade to Lucene 4.6.1 (MAHOUT-1364)  |
| 0.8            | 2.2 and later (with bootstrap action installation) |
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.

Use Predefined Bootstrap Actions

Amazon EMR provides predefined bootstrap action scripts that you can use to customize Hadoop settings. This section describes the available predefined bootstrap actions. References to predefined bootstrap action scripts are passed to Amazon EMR by using the bootstrap-action parameter.

Contents
- Configure Daemons Bootstrap Action (p. 130)
- Configure Hadoop Bootstrap Action (p. 132)
- Run If Bootstrap Action (p. 135)
- S3Get Bootstrap Action (p. 136)

Configure Daemons Bootstrap Action

Use this predefined bootstrap action to specify the heap size or other Java Virtual Machine (JVM) options for the Hadoop daemons. You can configure Hadoop for large jobs that require more memory than Hadoop allocates by default. You can also use this bootstrap action to modify advanced JVM options, such as garbage collector (GC) behavior.

The location of the script is s3://elasticmapreduce/bootstrap-actions/configure-daemons.

The following table describes the valid parameters for the script. In the table, daemon can be namenode, datanode, jobtracker, tasktracker, or client (Hadoop 1.x) or namenode, datanode, resourcemanager, nodemanager, or client (Hadoop 2.x). For example, --namenode-heap-size=2048,--namenode-opts="-XX:GCTimeRatio=19"

<table>
<thead>
<tr>
<th>Configuration Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--daemon-heap-size</td>
<td>Sets the heap size in megabytes for the specified daemon. Note --client-heap-size has no effect. Instead, change the client heap size using the --client-opts=&quot;-Xmx#####M&quot; equivalent, where ##### is numeric.</td>
</tr>
<tr>
<td>--daemon-opts</td>
<td>Sets additional Java options for the specified daemon.</td>
</tr>
<tr>
<td>--replace</td>
<td>Replaces the existing hadoop-user-env.sh file if it exists.</td>
</tr>
</tbody>
</table>

The configure-daemons bootstrap action supports Hadoop 2.x with a configuration file, yarn-site.xml. Its configuration file keyword is yarn.

The following examples set the NameNode JVM heap size to 2048 MB and configures a JVM GC option for the NameNode.

To set the NameNode heap size using the AWS CLI

When using the AWS CLI to include a bootstrap action, specify the Path and Args as a comma-separated list.
• To create a cluster and run a bootstrap action to configure the Hadoop NameNode daemon’s heap size, type the following command and replace `myKey` with the name of your Amazon EC2 key pair.

- **Linux, UNIX, and Mac OS X users:**

```bash
aws emr create-cluster --name "Test cluster" --ami-version 3.3 \  --use-default-roles --ec2-attributes KeyName=myKey \  --applications Name=Hue Name=Hive Name=Pig \  --instance-count 5 --instance-type m3.xlarge \  --bootstrap-action Path=s3://elasticmapreduce/bootstrap-actions/configure-daemons,Args=["--namenode-heap-size=2048","--namenode-opts=-XX:GCTimeRatio=19"]
```

- **Windows users:**

```bash
aws emr create-cluster --name "Test cluster" --ami-version 3.3 --use-default-roles --ec2-attributes KeyName=myKey --applications Name=Hue Name=Hive Name=Pig --instance-count 5 --instance-type m3.xlarge --bootstrap-action Path=s3://elasticmapreduce/bootstrap-actions/configure-daemons,Args=["--namenode-heap-size=2048","--namenode-opts=-XX:GCTimeRatio=19"]
```

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

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

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

Alternatively, you can supply a JSON syntax in a file if you have a long list of arguments or multiple bootstrap actions. For example, the JSON file `configuredaemons.json` would look like the following:

```json
[
  {
    "Path": "s3://elasticmapreduce/bootstrap-actions/configure-daemons",
    "Args": ["--namenode-heap-size=2048","--namenode-opts=-XX:GCTimeRatio=19"],
    "Name": "Configure Daemons"
  }
]
```

### To set the NameNode heap size using the Amazon EMR CLI

**Note**
The Amazon EMR CLI is no longer under feature development. Customers are encouraged to use the Amazon EMR commands in the AWS CLI instead.

- In the directory where you installed the Amazon EMR CLI, type the following command.

- **Linux, UNIX, and Mac OS X:**

```bash
./elastic-mapreduce --create --alive \  --bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-daemons \  --args --namenode-heap-size=2048,--namenode-opts=-XX:GCTimeRatio=19
```
• Windows:

```
ruby elastic-mapreduce --create --alive --bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-daemons --args --namenode-heap-size=2048,--namenode-opts=-XX:GCTimeRatio=19
```

**Configure Hadoop Bootstrap Action**

You can use this bootstrap action to set cluster-wide Hadoop settings. The location of the script is `s3://elasticmapreduce/bootstrap-actions/configure-hadoop`. This script provides the following command line options:

- **--keyword-config-file**—Merges the existing Hadoop configuration with a user-specified XML configuration file that you upload to Amazon S3 or the local filesystem. The user-specified file can be named anything.

- **--keyword-key-value**—Overrides specific key-value pairs in the Hadoop configuration files.

With both options, replace **--keyword** with a keyword (or use the single character shortcut instead) that represents one of the five Hadoop configuration files described in the following table. Because the single-character shortcuts can be used together in the same command, an uppercase character indicates that the shortcut refers to a configuration file and a lowercase character indicates that the shortcut refers to a key-value pair. If you specify multiple options, the later options override the earlier ones.

<table>
<thead>
<tr>
<th>Configuration File Name</th>
<th>Configuration File Keyword</th>
<th>File Name Shortcut</th>
<th>Key-Value Pair Shortcut</th>
</tr>
</thead>
<tbody>
<tr>
<td>log4j.properties</td>
<td>log4j</td>
<td>L</td>
<td>l</td>
</tr>
<tr>
<td>core-site.xml</td>
<td>core</td>
<td>C</td>
<td>c</td>
</tr>
<tr>
<td>hadoop-default.xml (deprecated)</td>
<td>default</td>
<td>D</td>
<td>d</td>
</tr>
<tr>
<td>hadoop-site.xml (deprecated)</td>
<td>site</td>
<td>S</td>
<td>s</td>
</tr>
<tr>
<td>hdfs-site.xml</td>
<td>hdfs</td>
<td>H</td>
<td>h</td>
</tr>
<tr>
<td>mapred-site.xml</td>
<td>mapred</td>
<td>M</td>
<td>m</td>
</tr>
<tr>
<td>yarn-site.xml</td>
<td>yarn</td>
<td>Y</td>
<td>y</td>
</tr>
<tr>
<td>httpsfs-site.xml</td>
<td>httpfs</td>
<td>T</td>
<td>t</td>
</tr>
<tr>
<td>emrfs-site.xml</td>
<td>emrfs</td>
<td>E</td>
<td>e</td>
</tr>
<tr>
<td>capacity-scheduler.xml</td>
<td>capacity</td>
<td>Z</td>
<td>z</td>
</tr>
</tbody>
</table>

For Hadoop configuration variables you can look at Hadoop documentation, for example, at [http://hadoop.apache.org/docs/r2.2.0/hadoop-project-dist/hadoop-common/core-default.xml](http://hadoop.apache.org/docs/r2.2.0/hadoop-project-dist/hadoop-common/core-default.xml).

You can provide multiple configurations for multiple instance types. For example, you may have a task group that consists of different instance types than your core group. The `configure-hadoop bootstrap`
action provides an option, `instance-type-config`, which accepts the Amazon S3 URI or local path to a JSON file that specifies configurations for each instance type. That file would look something like:

```json
{
    "m1.small":{
        "log4j":{
            "key1":"value1"
        },
        "site":{
            "key3":"value3"
        }
    },
    "m1.xlarge":{
        "yarn":{
            "lkey1":"lvalue1",
            "lkey11":"lvalue12"
        },
        "emrfs":{
            "lkey2":"lvalue2"
        },
        "site":{
            "lkey3":"lvalue3"
        }
    }
}
```

**Note**

If you do not want to set any values for a particular instance type, you should still provide a blank entry in the JSON list, e.g.

```json
{
    "m1.small":{
    }
}
```

The following example shows how to use the configuration file keywords (`mapred` in this example) to merge a user-specified configuration file (`config.xml`) with Hadoop's `mapred-site.xml` file and set the maximum map tasks value to 2 in the `mapred-site.xml` file. The configuration file that you provide in the Amazon S3 bucket must be a valid Hadoop configuration file; for example:

```xml
<?xml version="1.0"?>
<configuration>
    <property>
        <name>mapred.userlog.retain.hours</name>
        <value>4</value>
    </property>
</configuration>
```

The configuration file for Hadoop 0.18 is `hadoop-site.xml`. In Hadoop 0.20 and later, the old configuration file is replaced with three new files: `core-site.xml`, `mapred-site.xml`, and `hdfs-site.xml`.

For Hadoop 0.18, the name and location of the configuration file is `/conf/hadoop-site.xml`.

The configuration options are applied in the order described in the bootstrap action script. Settings specified later in the sequence override those specified earlier.
To change the maximum number of map tasks using the AWS CLI

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

- To launch a cluster with a bootstrap action that configures the maximum number of map tasks, type the following command and replace `myKey` with the name of your EC2 key pair.
  - Linux, UNIX, and Mac OS X users:
    ```bash
    aws emr create-cluster --name "Test cluster" --ami-version 3.3.0 --use-default-roles --ec2-attributes KeyName=myKey --applications Name=Hue Name=Hive Name=Pig --instance-count 5 --instance-type m3.xlarge --bootstrap-action Path=s3://elasticmapreduce/bootstrap-actions/configure-hadoop,Args=["-M", "s3://myawsbucket/config.xml","-m","mapred.tasktracker.map.tasks.maximum=2"]
    ```
  - Windows users:
    ```bash
    aws emr create-cluster --name "Test cluster" --ami-version 3.3.0 --use-default-roles --ec2-attributes KeyName=myKey --applications Name=Hue Name=Hive Name=Pig --instance-count 5 --instance-type m3.xlarge --bootstrap-action Path=s3://elasticmapreduce/bootstrap-actions/configure-hadoop,Args=["-M", "s3://myawsbucket/config.xml","-m","mapred.tasktracker.map.tasks.maximum=2"]
    ```

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

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

  Alternatively, you can provide a JSON file if you have a long list of arguments or multiple bootstrap actions. For example, the JSON file `configuredaemons.json` would look like this:

  ```json
  [
    {
      "Path": "s3://elasticmapreduce/bootstrap-actions/configure-hadoop",
      "Args": ["-M","s3://myawsbucket/config.xml","-m","mapred.tasktracker.map.tasks.maximum=2"],
      "Name": "Configure Hadoop"
    }
  ]
  ```

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

To change the maximum number of map tasks using the Amazon EMR CLI

  **Note**
  
  The Amazon EMR CLI is no longer under feature development. Customers are encouraged to use the Amazon EMR commands in the AWS CLI instead.

  - In the directory where you installed the Amazon EMR CLI, type the following command.
(Optional) Create Bootstrap Actions to Install Additional Software

- Linux, UNIX, and Mac OS X:

```bash
./elastic-mapreduce --create --bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hadoop --args "-M,s3://mybucket/config.xml,-m,mapred.tasktracker.map.tasks.maximum=2"
```

- Windows:

```bash
ruby elastic-mapreduce --create --bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hadoop --args "-M,s3://myawsbucket/config.xml,-m,mapred.tasktracker.map.tasks.maximum=2"
```

To provide multiple configurations using the AWS CLI

- To launch a cluster with different instance type configurations using the AWS CLI and configure-hadoop bootstrap action, supply the `instance-type-config` option with the URI or path to the JSON configuration file:

```bash
aws emr create-cluster --name "Test cluster" --ami-version 3.3.2 --use-default-roles --ec2-attributes KeyName=myKey --instance-count 5 --instance-type m3.xlarge --bootstrap-action Path=s3://elasticmapreduce/bootstrap-actions/configure-hadoop,Args="["instance-type-config","s3://myBucket/myInstanceConfigfile.json"]"
```

Run If Bootstrap Action

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

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

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

To run a command conditionally using the AWS CLI

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

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

```bash
aws emr create-cluster --name "Test cluster" --ami-version 3.3 --use-default-roles --ec2-attributes KeyName=myKey --applications Name=Hue Name=Hive Name=Pig --instance-count 5 --instance-type m3.xlarge --bootstrap-action Path=s3://elasticmapreduce/bootstrap-actions/configure-hadoop,Args="["instance-type-config","s3://myBucket/myInstanceConfigfile.json"]"
```

- Windows users:

```bash
aws emr create-cluster --name "Test cluster" --ami-version 3.3 --use-default-roles --ec2-attributes KeyName=myKey --applications Name=Hue Name=Hive Name=Pig --instance-count 5 --instance-type m3.xlarge --bootstrap-action Path=s3://elasticmapreduce/bootstrap-actions/configure-hadoop,Args="["instance-type-config","s3://myBucket/myInstanceConfigfile.json"]"
```
When you specify the instance count without using the `--instance-groups` parameter, a single Master node is launched, and the remaining instances are launched as core nodes. All nodes will use the instance type specified in the command.

**Note**

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

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

### S3Get Bootstrap Action

**Note**

This bootstrap action is only available with AMIs greater than 3.4.0.

Use this predefined bootstrap action to retrieve files from Amazon S3 and place them in a path on each node in the cluster. `s3get` is local to the cluster and the location of the script is `file:/usr/share/aws/emr/scripts/s3get`.

This script is useful if you must use artifacts located in Amazon S3 that must be placed on each node in the Amazon EMR cluster. For example, in EMRFS client-side encryption, you may need to provide a custom EncryptionMaterialsProvider class JAR. You use `s3get` to retrieve the JAR from your S3 bucket and place it in a target path on every node in the cluster.

The options for `s3get` are:

- `-s path | --src=path`
  
  The Amazon S3 source path.

- `-d path | --dst=path`
  
  The EMR cluster destination path.

- `-f | --force`
  
  Overwrite the destination path if a file already exists at that location.

### Shutdown Actions

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

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

**Note**

When using Amazon EMR versions 4.0 and later, you must manually create the `/mnt/var/lib/instance-controller/public/shutdown-actions/` directory on the master node. It doesn't exist by default; however, after being created, scripts in this directory nevertheless
run before shutdown. For more information about connecting to the Master node to create directories, see Connect to the Master Node Using SSH (p. 457).

Use Custom Bootstrap Actions

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

Contents

- Add Custom Bootstrap Actions Using the AWS CLI or the Amazon EMR CLI (p. 137)
- Add Custom Bootstrap Actions Using the Console (p. 138)

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

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

The sample script looks like the following:

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

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

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

- To launch a cluster with a custom bootstrap action, type the following command, replace myKey with the name of your EC2 key pair.
  - Linux, UNIX, and Mac OS X users:
    ```bash
    aws emr create-cluster --name "Test cluster" --ami-version 3.3 \
    --use-default-roles --ec2-attributes KeyName=myKey \
    --applications Name=Hue Name=Hive Name=Pig \
    --instance-count 5 --instance-type m3.xlarge \
    --bootstrap-action Path="s3://elasticmapreduce/bootstrap-actions/download.sh"
    ```
  - Windows users:
    ```bash
    aws emr create-cluster --name "Test cluster" --ami-version 3.10 --use-default-roles --ec2-attributes KeyName=myKey --applications Name=Hue Name=Hive Name=Pig \
    --instance-count 5 --instance-type m3.xlarge --bootstrap-action Path="s3://elasticmapreduce/bootstrap-actions/download.sh"
    ```

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

Note

If you have not previously created the default Amazon EMR service role and EC2 instance profile, type aws emr create-default-roles to create them before typing the create-cluster subcommand.
For more information on using Amazon EMR commands in the AWS CLI, see http://docs.aws.amazon.com/cli/latest/reference/emr.

**To create a cluster with a custom bootstrap action using the Amazon EMR CLI**

*Note*
The Amazon EMR CLI is no longer under feature development. Customers are encouraged to use the Amazon EMR commands in the AWS CLI instead.

- In the directory where you installed the Amazon EMR CLI, type the following command.
  - Linux, UNIX, and Mac OS X:
    ```bash
    ./elastic-mapreduce --create --alive --bootstrap-action "s3://elasticmapreduce/bootstrap-actions/download.sh"
    ```
  - Windows:
    ```ruby
    ruby elastic-mapreduce --create --alive --bootstrap-action "s3://elasticmapreduce/bootstrap-actions/download.sh"
    ```

**Add Custom Bootstrap Actions Using the Console**

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

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

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. Choose **Create cluster**.
3. Click **Go to advanced options**.
4. In Create Cluster - Advanced Options, Steps 1 and 2 choose the options as desired and proceed to **Step 3: General Cluster Settings**.
5. Under **Bootstrap Actions** select **Configure and add** to specify the Name, JAR location, and arguments for your bootstrap action. Choose **Add**.
6. Optionally add more bootstrap actions as desired.
7. Proceed to create the cluster. Your bootstrap action(s) will be performed after the cluster has been provisioned and initialized.

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

**Related Topics**

- [View Log Files](p. 422)

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

```bash
aws emr create-cluster --name "Test cluster" --release-label \
--use-default-roles --ec2-attributes KeyName=myKey \
--applications Name=Hive Name=Pig \
--instance-count 3 --instance-type m3.xlarge \
```
Configure Cluster Hardware and Networking

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

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

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

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

Master Node

The master node manages the cluster and typically runs master components of distributed applications. For example, the master node runs the YARN ResourceManager service to manage resources for applications, as well as the HDFS NameNode service. It also tracks the status of jobs submitted to the cluster and monitors the health of the instance groups. Because there is only one master node, the instance group or instance fleet consists of a single EC2 instance.

To monitor the progress of a cluster and interact directly with applications, you can connect to the master node over SSH as the Hadoop user. For more information, see Connect to the Master Node Using SSH (p. 457). Connecting to the master node allows you to access directories and files, such as Hadoop log files, directly. For more information, see View Log Files (p. 422). You can also view user interfaces that applications publish as websites running on the master node. For more information, see View Web Interfaces Hosted on Amazon EMR Clusters (p. 461).

Core Nodes

Core nodes are managed by the master node. Core nodes run the Data Node daemon to coordinate data storage as part of the Hadoop Distributed File System (HDFS). They also run the Task Tracker daemon and perform other parallel computation tasks on data that installed applications require. For example, a core node runs YARN NodeManager daemons, Hadoop MapReduce tasks, and Spark executors. Like the master node, at least one core node is required per cluster. However, unlike the master node, there can be multiple core nodes—and therefore multiple EC2 instances—in the instance group or instance fleet. There is only one core instance group or instance fleet. With instance groups, you can add and remove EC2 instances while the cluster is running or set up automatic scaling. For more information about adding and removing EC2 instances with the instance groups configuration, see Scaling Cluster Resources (p. 477). With instance fleets, you can effectively add and remove instances by modifying the instance fleet’s target capacities for On-Demand and Spot accordingly. For more information about target capacities, see Instance Fleet Options (p. 154).
Warning
Removing HDFS daemons from a running node runs the risk of losing data.

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

Instance Fleets
The instance fleets configuration offers the widest variety of provisioning options for EC2 instances. Each node type has a single instance fleet, and the task instance fleet is optional. For each instance fleet, you specify up to five instance types, which can be provisioned as On-Demand and Spot Instances. For the core and task instance fleets, you assign a target capacity for On-Demand Instances, and another for Spot Instances. Amazon EMR chooses any mix of the five instance types to fulfill the target capacities, provisioning both On-Demand and Spot Instances. For the master node type, Amazon EMR chooses a single instance type from your list of up to five, and you specify whether it's provisioned as an On-Demand or Spot Instance. Instance fleets also provide additional options for Spot Instance purchases, which include a defined duration (also known as a spot block) and a timeout that specifies an action to take if Spot capacity can't be provisioned. For more information, see Configure Instance Fleets (p. 153).

Uniform Instance Groups
Uniform instance groups offer a simplified setup. Each Amazon EMR cluster can include up to 50 instance groups: one master instance group that contains one EC2 instance, a core instance group that contains one or more EC2 instances, and up to 48 optional task instance groups. Each core and task instance group can contain any number of EC2 instances. You can scale each instance group by adding and removing EC2 instances manually, or you can set up automatic scaling. For more information about configuring uniform instance groups, see Configure Uniform Instance Groups (p. 161). For information about adding and removing instances, see Scaling Cluster Resources (p. 477).

Plan and Configure EC2 Instances
EC2 instances come in different configurations, which are known as instance types. Each instance type has a different CPU, input/output, and storage capacity. In addition to the instance type, you can choose different purchasing options for EC2 instances. You can specify different instance types and purchasing options within instance groups or instance fleets. For more information about instance fleets, see Configure Instance Fleets (p. 153). For more information about instance groups, see Create a Cluster with Instance Fleets or Uniform Instance Groups (p. 153). For guidance about choosing the right options, see Cluster Configuration Guidelines (p. 163).

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.
Supported Instance Types

Amazon EC2 offers several basic instance types.

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

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

**Note**
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 instances and upgrade paths, see Previous Generation Instances.

<table>
<thead>
<tr>
<th>Instance Class</th>
<th>Instance Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>General purpose</td>
<td>m1.medium</td>
</tr>
<tr>
<td>Compute optimized</td>
<td>c1.medium</td>
</tr>
<tr>
<td>Memory optimized</td>
<td>r3.xlarge</td>
</tr>
<tr>
<td>Storage optimized</td>
<td>hs1.8xlarge</td>
</tr>
<tr>
<td></td>
<td>Note: i3-series instances available when using Amazon EMR version 5.9.0 and later.</td>
</tr>
<tr>
<td>GPU instances</td>
<td>g2.2xlarge</td>
</tr>
<tr>
<td></td>
<td>Note: NVIDIA and CUDA drivers are installed on P2 and P3 instance types by default.</td>
</tr>
</tbody>
</table>
Note
M4 and C4 instance types are only supported on Amazon EMR releases 3.10.0 and 4.x or greater, and all new releases launch clusters with EBS-backed storage for root volumes on all instance types.

Virtualization Types

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

<table>
<thead>
<tr>
<th>Instance Family</th>
<th>Virtualization Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>PVM</td>
</tr>
<tr>
<td>c3</td>
<td>PVM</td>
</tr>
<tr>
<td>c4</td>
<td>HVM</td>
</tr>
<tr>
<td>cc2</td>
<td>HVM</td>
</tr>
<tr>
<td>cg1</td>
<td>HVM</td>
</tr>
<tr>
<td>cr1</td>
<td>HVM</td>
</tr>
<tr>
<td>d2</td>
<td>HVM</td>
</tr>
<tr>
<td>g2</td>
<td>HVM</td>
</tr>
<tr>
<td>hs1</td>
<td>PVM</td>
</tr>
<tr>
<td>i2</td>
<td>HVM</td>
</tr>
<tr>
<td>i3</td>
<td>HVM</td>
</tr>
<tr>
<td>m1</td>
<td>PVM</td>
</tr>
<tr>
<td>m2</td>
<td>PVM</td>
</tr>
<tr>
<td>m3</td>
<td>PVM</td>
</tr>
<tr>
<td>m4</td>
<td>HVM</td>
</tr>
<tr>
<td>p2</td>
<td>HVM</td>
</tr>
<tr>
<td>p3</td>
<td>HVM</td>
</tr>
<tr>
<td>r3</td>
<td>HVM</td>
</tr>
<tr>
<td>r4</td>
<td>HVM</td>
</tr>
</tbody>
</table>

Enhanced Networking

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

- Linux AMI Virtualization Types
- Enhanced Networking on Linux
- Placement Groups
Instance Purchasing Options

In addition to the instance type, you choose a purchasing option for instances. You can choose to use On-Demand Instances, Spot Instances, or both. If you choose to use the uniform instance group configuration, the instance type and purchasing option apply to all EC2 instances in each instance group, and you can only specify these options for an instance group when you create it. If you choose to use the instance fleet configuration, you can change purchasing options after an instance group is created, and you can mix purchasing options to fulfill a target capacity that you specify. For more information about these configurations, see Create a Cluster with Instance Fleets or Uniform Instance Groups (p. 153).

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

On-Demand Instances

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

Using Reserved Instances

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

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

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

Using Dedicated Instances

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

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

Spot Instances

Spot Instances in Amazon EMR provide an option for you to purchase Amazon EC2 instance capacity at a reduced cost as compared to On-Demand purchasing. The disadvantage of using Spot Instances is that
instances may terminate unpredictably as prices fluctuate. When you create a cluster with instance fleets, you have the option to use a defined duration (also known as a Spot block) which provides a greater degree of predictability. Spot Instances terminate at the end of the duration, but are not interrupted before the duration expires. This topic describes how Spot Instances work with Amazon EMR. For further details about Spot Instances, see Spot Instances in the Amazon EC2 User Guide for Linux Instances.

When Amazon EC2 has unused capacity, it offers EC2 instances at a reduced cost, called the Spot price. This price fluctuates based on availability and demand, and is established by region and Availability Zone. For current pricing, you can see Amazon EC2 Spot Instances Pricing. When you create and configure a cluster, you specify network options that ultimately determine the Availability Zone where your cluster launches. For more information, see Plan and Configure Networking (p. 145). When you configure a cluster with instance groups or instance fleets, you can designate the Spot purchasing option for each node type, along with the maximum 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, instances launch. While instances run, you're charged at the current Spot price (not your maximum Spot price). If you manually start and terminate Spot Instances, partial hours are billed as full hours. On the other hand, if instances are terminated because the current Spot price rises above your maximum Spot price, you are not charged either the Amazon EC2 or Amazon EMR charges for the partial hour.

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

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

**Spot Instances in Uniform Instance Groups**

When you use Spot Instances in a uniform instance group, all instances in an instance group must be Spot Instances. You specify a single subnet or Availability Zone for the cluster. For each instance group, you specify a single Spot Instance type and a maximum Spot price. Spot Instances of that type launch if the Spot price in the cluster's region and Availability Zone is below the maximum Spot price. Instances terminate if the Spot price is above your maximum Spot price. You set the maximum Spot price only when you configure an instance group. It can't be changed later. For more information, see Create a Cluster with Instance Fleets or Uniform Instance Groups (p. 153).

**Spot Instances in Instance Fleets**

When you use the instance fleets configuration, additional options give you more control over how Spot Instances launch and terminate. Fundamentally, instance fleets use a different method than uniform instance groups to launch instances. The way it works is you establish a target capacity for Spot Instances (and On-Demand Instances) and up to five instance types. You can also specify a weighted capacity for each instance type or use the vCPU (YARN vcores) of the instance type as weighted capacity. This weighted capacity counts toward your target capacity when an instance of that type is provisioned. Amazon EMR provisions instances with both purchasing options until the target capacity for each target is fulfilled. In addition, you can define a range of Availability Zones for Amazon EMR to choose from when launching instances. You also provide additional Spot options for each fleet, including a provisioning timeout and, optionally, a defined duration. For more information, see Configure Instance Fleets (p. 153).
Instance Store and Amazon EBS

There are two types of storage volumes available for EC2 instances: Amazon EBS volumes and the instance store. Amazon EBS volumes are available only in Amazon EMR version 4.0 and later. Whether the root device volume uses the instance store or an Amazon EBS volume depends on the AMI. Some AMIs are backed by Amazon EC2 instance store, and some are backed by Amazon EBS. For more information, see Amazon EC2 Root Device Volume in the Amazon EC2 User Guide for Linux Instances.

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

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

Amazon EMR automatically attaches an Amazon EBS General Purpose SSD (gp2) 10-GB volume as the root device for its AMIs to enhance performance. The EBS costs are pro-rated by the hour based on the monthly Amazon EBS charges for gp2 volumes in the region where the cluster runs. For example, the EBS cost per hour for the root volume on each cluster node in a region that charges $0.10/GB/month would be approximately $0.00139 per hour ($0.10/GB/month divided by 30 days divided by 24h times 10 GB).

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

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

Other caveats for using Amazon EBS with EMR clusters are:

- You can't snapshot an EBS volume and then restore it within Amazon EMR. To create reusable custom configurations, use a custom AMI (available in Amazon EMR version 5.7.0 and later).
- An encrypted EBS root storage volume is supported only when using a custom AMI. Encrypted EBS storage volumes are not supported.
- If you apply tags using the Amazon EMR API, those operations are applied to EBS volumes.
- There is a limit of 25 volumes per instance.

Plan and Configure Networking

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.
There may be two network platform options you can choose for your cluster: **EC2-Classic** or **EC2-VPC**. In EC2-Classic, your instances run in a single, flat network that you share with other customers. EC2-Classic is available only with certain accounts in certain regions. For more information, see Amazon EC2 and Amazon VPC in the Amazon EC2 User Guide for Linux Instances. In EC2-VPC, your cluster uses Amazon Virtual Private Cloud (Amazon VPC), and EC2 instances run in a VPC that’s logically isolated within your AWS account. Amazon VPC enables you to provision a virtual private cloud (VPC), an isolated area within AWS where you can configure a virtual network, controlling aspects such as private IP address ranges, subnets, routing tables, and network gateways.

VPC offers the following capabilities:

- **Processing sensitive data**

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

- **Accessing resources on an internal network**

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

**Public and Private Subnets**

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

**Important**

Amazon EMR only supports launching clusters in private subnets in releases 4.2 or greater. For further information about private subnets in Amazon EMR, see Amazon EMR Management Guide.

For more information about Amazon VPC, see the Amazon VPC User Guide.

**Topics**

- Amazon VPC Options (p. 146)
- Set up a VPC to Host Clusters (p. 148)
- Launch Clusters into a VPC (p. 150)
- Restrict Permissions to a VPC Using IAM (p. 151)
- Minimum Amazon S3 Policy for Private Subnet (p. 152)
- More Resources for Learning About VPCs (p. 152)

**Amazon VPC Options**

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

EMR clusters in a public subnet require a connected internet gateway. This is because Amazon EMR clusters must access AWS services and Amazon EMR. If a service, such as Amazon S3, provides the ability to create a VPC endpoint, you can access those services using the endpoint instead of accessing a public endpoint through an internet gateway. Additionally, Amazon EMR cannot communicate with clusters in public subnets through a network address translation (NAT) device. An internet gateway is required for this purpose but you can still use a NAT instance or gateway for other traffic in more complex scenarios.

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

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

Public Subnet Security Groups

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

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

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

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

To create a subnet to run Amazon EMR clusters

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation bar, select the region in which to run your cluster.
3. Choose Start VPC Wizard.
4. Choose the VPC configuration by selecting one of the following options:
   
   - **VPC with a Single Public Subnet**—Select this option if the data used in the cluster is available on the internet (for example, in Amazon S3 or Amazon RDS).

5. Confirm the VPC settings. The images show both single public and private and public scenarios.

   - To work with Amazon EMR, the VPC with a public subnet must have both an internet gateway and a subnet.
   - Use a private IP address space for your VPC to ensure proper DNS hostname resolution; otherwise, you may experience Amazon EMR cluster failures. This includes the following IP address ranges:
     - 10.0.0.0 - 10.255.255.255
     - 172.16.0.0 - 172.31.255.255
     - 192.168.0.0 - 192.168.255.255
   - Choose **Use a NAT instance instead** and select options as appropriate.
   - Optionally choose to **Add endpoints for S3 to your subnets**.
   - Verify that **Enable DNS hostnames** is checked. You have the option to enable DNS hostnames when you create the VPC. To change the setting of DNS hostnames, select your VPC in the VPC list, then choose **Edit** in the details pane. To create a DNS entry that does not include a domain name, create a value for **DHCP Options Set**, and then associate it with your VPC. You cannot edit the domain name using the console after the DNS option set has been created.

   For more information, see Using DNS with Your VPC.

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

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

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

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

   For more information, see DHCP Options Sets in the Amazon VPC User Guide.

6. Choose **Create VPC**. If you are creating a NAT instance, it may take a few minutes for this to complete.
After the VPC is created, go to the **Subnets** page and note the identifier of one of the subnets of your VPC. You use this information when you launch the EMR cluster into the VPC.

### Launch Clusters into a VPC

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

**Note**

Amazon EMR supports private subnets in release versions 4.2 and above. For further information about private subnets in EMR, see Amazon EMR Management Guide.

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

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

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

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

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

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

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

**Important**

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

**Note**

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

5. Proceed with creating the cluster.

### To launch a cluster into a VPC using the AWS CLI

**Note**

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

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

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

  - **Linux, UNIX, and macOS users:**

    ```
    aws emr create-cluster --name "Test cluster" --ami-version 3.10 --applications Name=Hue Name=Hive Name=Pig \ 
    --use-default-roles --ec2-attributes KeyName=myKey,SubnetId=subnet-77XXXX03 \ 
    --instance-type m3.xlarge --instance-count 3
    ```

  - **Windows users:**

    ```
    aws emr create-cluster --name "Test cluster" --ami-version 3.10 --applications Name=Hue Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey,SubnetId=subnet-77XXXX03 --instance-type m3.xlarge --instance-count 3
    ```

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

**Note**

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

For more information about using Amazon EMR commands in the AWS CLI, see the [AWS CLI](https://docs.aws.amazon.com/cli/latest/reference/emr.html).

### Restrict Permissions to a VPC Using IAM

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

You can also use IAM to control who can create and administer subnets. For more information about administering policies and actions in Amazon EC2 and Amazon VPC, see [IAM Policies for Amazon EC2](https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/iam-policies-for-ec2.html) in the [Amazon EC2 User Guide for Linux Instances](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/).
By default, all IAM users can see all of the subnets for the account, and any user can launch a cluster in any subnet.

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

**Minimum Amazon S3 Policy for Private Subnet**

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

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

**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
Create a Cluster with Instance Fleets or Uniform Instance Groups

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

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

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

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

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

Topics
• Configure Instance Fleets (p. 153)
• Configure Uniform Instance Groups (p. 161)
• Cluster Configuration Guidelines (p. 163)

Configure Instance Fleets

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

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

### Instance Fleet Options

#### Target Capacities and Weighted Capacity

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

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

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

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

### Spot Instance Options

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

For the fleet as a whole, when you specify a *Defined duration*, Spot Instances are not interrupted during the defined duration and terminate after it expires. Defined duration pricing applies when you select this option. For more information, see Amazon EC2 Spot Instances Pricing. If you don't specify a defined duration, instances terminate as soon as the Spot price exceeds the maximum Spot price.

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

For more information about Spot Instances, see Spot Instances in the Amazon EC2 User Guide for Linux Instances.

### Multiple Subnet (Availability Zones) Options

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

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

Summary of Key Features

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

Use the Console to Configure Instance Fleets

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

To create a cluster with instance fleets using the console

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. Choose Create cluster.
3. Choose Go to advanced options, enter Software Configuration options, and then choose Next.
4. Choose Instance fleets.
5. For Network, enter a value. If you choose a VPC for Network, choose a single EC2 Subnet or CTRL + click to choose multiple EC2 subnets. The subnets you select must be the same type (public or private). If you choose only one, your cluster launches in that subnet. If you choose a group, the subnet with the best fit is selected from the group when the cluster launches. **Note**
   
   Your account and region may give you the option to choose Launch into EC2-Classic for Network. If you choose that option, choose one or more from EC2 Availability Zones rather than EC2 Subnets. For more information, see Amazon EC2 and Amazon VPC in the Amazon EC2 User Guide for Linux Instances.

6. Within each node type row, under Node type, if you want to change the default name of an instance fleet, click the pencil icon and then enter a friendly name. If want to remove the Task instance fleet, click the X icon.
7. Under Target capacity, choose options according to the following guidelines:
• Choose how you want to define the **Target capacity.** If you choose **vCPU**, the number of YARN vcores for each **Fleet instance type** is used as its weighted capacity. If you choose **Generic units**, you assign a custom number for each target capacity, and then assign a custom weighted capacity to each instance type. A field for this purpose appears for each instance you add under **Fleet instance type.**

• For the **Master** node, select whether the instance is **On-demand** or **Spot**.

• For the **Core** and **Task** nodes, enter target capacities for **On-demand** and **Spot**. Amazon EMR provisions the **Fleet instance types** that you specify until these capacities are fulfilled.

8. Under **Fleet instance types** for each **Node type**, choose options according to the following guidelines:

• Choose **Add/remove instance types to fleet**, and then choose up to five instance types from the list. Amazon EMR may choose to provision any mix of these instance types when it launches the cluster.

• If a Node type is configured with a **Target capacity** for **Spot**, choose **Maximum Spot price** options. You can enter your maximum Spot price as a % of **On-Demand** pricing, or you can enter a **Dollars ($)** amount in USD.

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

• If you chose **Default units** for **Target capacity**, enter the weighted capacity you want to assign to each instance type in the **Each instance counts as** box.

• To have EBS volumes attached to the instance type when it's provisioned, click the pencil next to **EBS Storage** and then enter EBS configuration options.

9. If you established a **Target capacity** for **Spot**, choose **Advanced Spot options** according to the following guidelines:

• **Defined duration**—if left to the default, **Not set**, Spot Instances terminate as soon as the Spot price rises above the Maximum Spot price, or when the cluster terminates. If you set a value, Spot Instances don't terminate until the duration has expired.

  **Important**
  If you set a **Defined duration**, special defined duration pricing applies. For pricing details, see Amazon EC2 Spot Instances Pricing.

• **Provisioning timeout**—Use these settings to control what Amazon EMR does when it can't provision Spot Instances from among the **Fleet instance types** you specify. You enter a timeout period in minutes, and then choose whether to **Terminate the cluster** or **Switch to provisioning On-Demand Instances**. If you choose to switch to On-Demand Instances, the weighted capacity of On-Demand Instances counts toward the target capacity for Spot Instances, and Amazon EMR provisions On-Demand Instances until the target capacity for Spot Instances is fulfilled.

10. Choose **Next**, modify other cluster settings, and then launch the cluster.

**Use the CLI to Configure Instance Fleets**

• To create and launch a cluster with instance fleets, use the `create-cluster` command along with `--instance-fleet` parameters.

• To get configuration details of the instance fleets in a cluster, use the `list-instance-fleets` command.

• To make changes to the target capacity for an instance fleet, use the `modify-instance-fleet` command.

• To add a task instance fleet to a cluster that doesn't already have one, use the `add-instance-fleet` command.
Create a Cluster with the Instance Fleets Configuration

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Example Use a JSON Configuration File

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

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

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

```
[
  {
    "Name": "Masterfleet",
    "InstanceFleetType": "MASTER",
    "TargetSpotCapacity": 1,
    "LaunchSpecifications": {
      "SpotSpecification": {
        "TimeoutDurationMinutes": 120,
        "TimeoutAction": "SWITCH_TO_ON_DEMAND"
      }
    }]
```
Get Configuration Details of Instance Fleets in a Cluster

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

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

```
{
  "InstanceFleets": [
    {
      "Status": {
        "Timeline": {
          "159": 
```
```
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.

```bash
aws emr modify-instance-fleet --cluster-id 'j-12ABCDEFGHI34JK' /
```
Create a Cluster with Instance Fleets or Uniform Instance Groups

--instance-fleet
InstanceFleetId='if-2ABC4DEFGHIJ4',TargetOnDemandCapacity=1,TargetSpotCapacity=1

Add a Task Instance Fleet to a Cluster

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

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

Configure Uniform Instance Groups

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

To add different instance types after a cluster is created, you can add additional task instance groups, specifying different instance types and purchasing options for each instance group. For more information, see Scaling Cluster Resources (p. 477).

This section covers creating a cluster with uniform instance groups. For more information about modifying an existing instance group by adding or removing instances manually or with automatic scaling, see Manage Clusters (p. 416).

Use the Console to Configure Uniform Instance Groups

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

To create a cluster with uniform instance groups using the console

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. Choose Create cluster.
3. Choose Go to advanced options, enter Software Configuration options, and then choose Next.
4. In the Hardware Configuration screen, leave Uniform instance groups selected.
5. Choose the Network, and then choose the EC2 Subnet in which you want your cluster to run. For more information about VPCs and subnets, see Plan and Configure Networking (p. 145).

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

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

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

• Under **Instance count**, enter the number of instances on which to run that node type. There is only one instance in the **Master** node type.

• Under **Purchasing option** choose **On-demand**, or choose **Spot** and enter the **Maximum Spot price** you are willing to pay per instance. Instances in this instance group launch when the Spot price in the Availability Zone of the **EC2 Subnet** you chose is below the **Maximum Spot price**.

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

7. To add another task instance group to the cluster, click and configure settings for the instance group as described in the previous step.

8. Choose **Next**, modify other cluster settings, and then launch the cluster.

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

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

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

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

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

Linux, UNIX, and macOS users:

```bash
aws emr create-cluster --name "Spot cluster" --ami-version 3.3 --applications Name=Hive Name=Pig \
  --use-default-roles --ec2-attributes KeyName=myKey \
  --instance-groups
  InstanceGroupType=MASTER,InstanceType=m3.xlarge,InstanceCount=1,BidPrice=0.25 \
  InstanceGroupType=CORE,BidPrice=0.03,InstanceType=m3.xlarge,InstanceCount=2 \
  InstanceGroupType=TASK,BidPrice=0.10,InstanceType=m3.xlarge,InstanceCount=3
```

Windows users:
aws emr create-cluster --name "Spot cluster" --ami-version 3.3 --applications Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey --instance-groups InstanceGroupType=MASTER,InstanceType=m3.xlarge,InstanceCount=1,BidPrice=0.25 InstanceGroupType=CORE,BidPrice=0.03,InstanceType=m3.xlarge,InstanceCount=2 InstanceGroupType=TASK,BidPrice=0.10,InstanceType=m3.xlarge,InstanceCount=3

Use the Java SDK to Create an Instance Group

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

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

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

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

Cluster Configuration Guidelines

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

What Instance Type Should You Use?

There are several ways to add EC2 instances to a cluster, which depend on whether you use the instance groups configuration or the instance fleets configuration for the cluster.

- **Instance Groups**
  - Manually add instances of the same type to existing core and task instance groups.
  - Manually add a task instance group, which can use a different instance type.
  - Set up automatic scaling in Amazon EMR for an instance group, adding and removing instances automatically based on the value of an Amazon CloudWatch metric that you specify. For more information, see Scaling Cluster Resources (p. 477).

- **Instance Fleets**
  - Add a single task instance fleet.
  - Change the target capacity for On-Demand and Spot Instances for existing core and task instance fleets. For more information, see Configure Instance Fleets (p. 153).
One way to plan the instances of your cluster is to run a test cluster with a representative sample set of data and monitor the utilization of the nodes in the cluster. For more information, see View and Monitor a Cluster (p. 416). Another way is to calculate the capacity of the instances you are considering and compare that value against the size of your data.

In general, the master node type, which assigns tasks, doesn’t require an EC2 instance with much processing power; EC2 instances for the core node type, which process tasks and store data in HDFS, need both processing power and storage capacity; EC2 instances for the task node type, which don’t store data, need only processing power. For guidelines about available EC2 instances and their configuration, see Plan and Configure EC2 Instances (p. 140).

The following guidelines apply to most Amazon EMR clusters.

• The master node does not have large computational requirements. For most clusters of 50 or fewer nodes, consider using an m3.xlarge instance. For clusters of more than 50 nodes, consider using an m3.2xlarge.

• The computational needs of the core and task nodes depend on the type of processing your application performs. Many jobs can be run on m3.xlarge instance types, which offer balanced performance in terms of CPU, disk space, and input/output. If your application has external dependencies that introduce delays (such as web crawling to collect data), you may be able to run the cluster on m3.large instances to reduce costs while the instances are waiting for dependencies to finish. For improved performance, consider running the cluster using m3.2xlarge instances for the core and task nodes. If different phases of your cluster have different capacity needs, you can start with a small number of core nodes and increase or decrease the number of task nodes to meet your job flow’s varying capacity requirements.

• Most Amazon EMR clusters can run on standard EC2 instance types such as m3.xlarge and m3.2xlarge. Computation-intensive clusters may benefit from running on High CPU instances, which have proportionally more CPU than RAM. Database and memory-caching applications may benefit from running on High Memory instances. Network-intensive and CPU-intensive applications like parsing, NLP, and machine learning may benefit from running on Cluster Compute instances, which provide proportionally high CPU resources and increased network performance.

• The amount of data you can process depends on the capacity of your core nodes and the size of your data as input, during processing, and as output. The input, intermediate, and output datasets all reside on the cluster during processing.

• By default, the total number of EC2 instances you can run on a single AWS account is 20. This means that the total number of nodes you can have in a cluster is 20. For more information about how to request that this limit be increased for your account, see AWS Limits.

• In Amazon EMR, m1.small and m1.medium instances are recommended only for testing purposes and m1.small is not supported on Hadoop 2 clusters.

When Should You Use Spot Instances?

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

Long-Running Clusters and Data Warehouses

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

Cost-Driven Workloads

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

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

Application Testing

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

Choose What to Launch as Spot Instances

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

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

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

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

<table>
<thead>
<tr>
<th>Project</th>
<th>Master Instance Group</th>
<th>Core Instance Group</th>
<th>Task Instance Groups</th>
</tr>
</thead>
<tbody>
<tr>
<td>Long-running clusters</td>
<td>On-Demand</td>
<td>On-Demand or instance-fleet mix</td>
<td>Spot or instance-fleet mix</td>
</tr>
<tr>
<td>Cost-driven workloads</td>
<td>Spot</td>
<td>Spot</td>
<td>Spot</td>
</tr>
<tr>
<td>Data-critical workloads</td>
<td>On-Demand</td>
<td>On-Demand</td>
<td>Spot or instance-fleet mix</td>
</tr>
<tr>
<td>Application testing</td>
<td>Spot</td>
<td>Spot</td>
<td>Spot</td>
</tr>
</tbody>
</table>

Master Node as Spot Instance

The master node controls and directs the cluster. When it terminates, the cluster ends, so you should only launch the master node as a Spot Instance if you are running a cluster where sudden termination is acceptable. This might be the case if you are testing a new application, have a cluster that periodically persists data to an external store such as Amazon S3, or are running a cluster where cost is more important than ensuring the cluster's completion.

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

You can only add a Spot Instance master node when you launch the cluster. Master nodes cannot be added or removed from a running cluster.
Typically, you would only run the master node as a Spot Instance if you are running the entire cluster (all instance groups) as Spot Instances.

Core Instance Group as Spot Instances

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

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

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

Task Instance Groups as Spot Instances

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

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

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

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

Calculating the Required HDFS Capacity of a Cluster

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

- The number of EC2 instances in the core instance group.
- The storage capacity of the EC2 instances.
- The number and size of EBS volumes attached to core nodes.
- A replication factor, which accounts for how each data block is stored in HDFS for RAID-like redundancy. By default, the replication factor is three for a cluster of 10 or more core nodes, 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, add the capacity of instance store volumes the EC2 instance types you've selected to the total volume storage you have attached with EBS and multiply the result by the number of nodes in each instance group. Divide the total by the replication factor for the cluster. For example, a cluster with 10 core nodes of type m1.large would have 850 GB of space per-instance available to HDFS: (10 nodes x 850 GB per node) / replication factor of 3. For more information on instance store volumes, see Amazon EC2 Instance Store in the Amazon EC2 User Guide for Linux Instances.
If the calculated HDFS capacity value is smaller than your data, you can increase the amount of HDFS storage in the following ways:

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

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

Configure Cluster Logging and Debugging

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

One of the things to decide as you plan your cluster is how much debugging support you want to make available. When you are first developing your data processing application, we recommend testing the application on a cluster processing a small, but representative, subset of your data. When you do this, you will likely want to take advantage of all the debugging tools that Amazon EMR offers, such as archiving log files to Amazon S3.

When you've finished development and put your data processing application into full production, you may choose to scale back debugging. Doing so can save you the cost of storing log file archives in Amazon S3 and reduce processing load on the cluster as it no longer needs to write state to Amazon S3. The trade off, of course, is that if something goes wrong, you'll have fewer tools available to investigate the issue.

Default Log Files

By default, each cluster writes log files on the master node. These are written to the `/mnt/var/log/` directory. You can access them by using SSH to connect to the master node as described in Connect to the Master Node Using SSH (p. 457). Because these logs exist on the master node, when the node terminates—either because the cluster was shut down or because an error occurred—these log files are no longer available.

You do not need to enable anything to have log files written on the master node. This is the default behavior of Amazon EMR and Hadoop.

A cluster generates several types of log files, including:

- **Step logs** — These logs are generated by the Amazon EMR service and contain information about the cluster and the results of each step. The log files are stored in `/mnt/var/log/hadoop/steps/` directory on the master node. Each step logs its results in a separate numbered subdirectory: `/mnt/var/log/hadoop/steps/s-stepId1/` for the first step, `/mnt/var/log/hadoop/steps/s-stepId2/`, for the second step, and so on. The 13-character step identifiers (e.g. stepId1, stepId2) are unique to a cluster.

- **Hadoop logs** — These are the standard log files generated by Apache Hadoop. They contain information about Hadoop jobs, tasks, and task attempts. The log files are stored in `/mnt/var/log/hadoop/` on the master node.
• **Bootstrap action logs** — If your job uses bootstrap actions, the results of those actions are logged. The log files are stored in `/mnt/var/log/bootstrap-actions/` on the master node. Each bootstrap action logs its results in a separate numbered subdirectory: `/mnt/var/log/bootstrap-actions/1/` for the first bootstrap action, `/mnt/var/log/bootstrap-actions/2/`, for the second bootstrap action, and so on.

• **Instance state logs** — These logs provide information about the CPU, memory state, and garbage collector threads of the node. The log files are stored in `/mnt/var/log/instance-state/` on the master node.

### Archive Log Files to Amazon S3

**Note**

You cannot currently use log aggregation to Amazon S3 with the `yarn logs` utility.

You can configure a cluster to periodically archive the log files stored on the master node to Amazon S3. This ensures that the log files are available after the cluster terminates, whether this is through normal shut down or due to an error. Amazon EMR archives the log files to Amazon S3 at 5 minute intervals.

To have the log files archived to Amazon S3, you must enable this feature when you launch the cluster. You can do this using the console, the CLI, or the API. By default, clusters launched using the console have log archiving enabled. For clusters launched using the CLI or API, logging to Amazon S3 must be manually enabled.

#### To archive log files to Amazon S3 using the console

1. Open the Amazon EMR console at [https://console.aws.amazon.com/elasticmapreduce/](https://console.aws.amazon.com/elasticmapreduce/).
2. Choose *Create cluster*.
3. In the *Cluster Configuration* section, in the *Logging* field, accept the default option: **Enabled**.
   
   This determines whether Amazon EMR captures detailed log data to Amazon S3. You can only set this when the cluster is created. For more information, see View Log Files (p. 422).
4. In the *Log folder S3 location* field, type (or browse to) an Amazon S3 path to store your logs. You may also allow the console to generate an Amazon S3 path for you. If you type the name of a folder that does not exist in the bucket, it is created.

   When this value is set, Amazon EMR copies the log files from the EC2 instances in the cluster to Amazon S3. This prevents the log files from being lost when the cluster ends and the EC2 instances hosting the cluster are terminated. These logs are useful for troubleshooting purposes.

   For more information, see View Log Files (p. 422).
5. Proceed with creating the cluster as described in Plan and Configure Clusters (p. 27).

#### To archive log files to Amazon S3 using the AWS CLI

To archive log files to Amazon S3 using the AWS CLI, type the `create-cluster` command and specify the Amazon S3 log path using the `--log-uri` parameter.

- To log files to Amazon S3 type the following command and replace `myKey` with the name of your EC2 key pair.
  
  - Linux, UNIX, and Mac OS X users:

    ```bash
    aws emr create-cluster --name "Test cluster" --ami-version 3.8 --log-uri
    s3://mybucket/logs/ \
    --applications Name=Hue Name=Hive Name=Pig \
    --use-default-roles --ec2-attributesKeyName=myKey \
    ```
When you specify the instance count without using the `--instance-groups` parameter, a single Master node is launched, and the remaining instances are launched as core nodes. All nodes will use the instance type specified in the command.

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

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

**To aggregate logs in Amazon S3 using the AWS CLI**

**Note**
You cannot currently use log aggregation with the `yarn logs` utility. You can only use aggregation supported by this procedure.

Log aggregation (Hadoop 2.x) compiles logs from all containers for an individual application into a single file. To enable log aggregation to Amazon S3 using the AWS CLI, you use a bootstrap action at cluster launch to enable log aggregation and to specify the bucket to store the logs.

- To enable log aggregation, type the following command and replace `myKey` with the name of your EC2 key pair.

- **Linux, UNIX, and Mac OS X users:**

  ```
  aws emr create-cluster --name "Test cluster" --ami-version 3.9 --applications Name=Hue Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey --bootstrap-action Path=s3://elasticmapreduce/bootstrap-actions/configure-hadoop,Name="aggregate logs",Args=["-y","yarn.log-aggregation-enable=true","-y","yarn.log-aggregation.retain-seconds=-1","-y","yarn.log-aggregation.retain-check-interval-seconds=3000","-y","yarn.nodemanager.remote-app-log-dir=s3://mybucket/logs"] --instance-type m3.xlarge --instance-count 3
  ```

- **Windows users:**

  ```
  aws emr create-cluster --name "Test cluster" --ami-version 3.9 --applications Name=Hue Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey --bootstrap-action Path=s3://elasticmapreduce/bootstrap-actions/configure-hadoop,Name="aggregate logs",Args=["-y","yarn.log-aggregation-enable=true","-y","yarn.log-aggregation.retain-seconds=-1","-y","yarn.log-aggregation.retain-check-interval-seconds=3000","-y","yarn.nodemanager.remote-app-log-dir=s3://mybucket/logs"] --instance-type m3.xlarge --instance-count 3
  ```

When you specify the instance count without using the `--instance-groups` parameter, a single Master node is launched, and the remaining instances are launched as core nodes. All nodes will use the instance type specified in the command.
Note
If you have not previously created the default EMR service role and EC2 instance profile, type `aws emr create-default-roles` to create them before typing the `create-cluster` subcommand.

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

Enable the Debugging Tool

The debugging tool allows you to more easily browse log files from the EMR console. For more information, see View Log Files in the Debugging Tool (p. 425). When you enable debugging on a cluster, Amazon EMR archives the log files to Amazon S3 and then indexes those files. You can then use the console to browse the step, job, task, and task-attempt logs for the cluster in an intuitive way.

To use the debugging tool in the EMR console, you must enable debugging when you launch the cluster using the console, the CLI, or the API.

To enable the debugging tool using the Amazon EMR console

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. Choose Create cluster.
3. In the Cluster Configuration section, in the Logging field, choose Enabled. You cannot enable debugging without enabling logging.
4. In the Log folder S3 location field, type an Amazon S3 path to store your logs.
5. In the Debugging field, choose Enabled.

The debugging option creates an Amazon SQS exchange to publish debugging messages to the Amazon EMR service backend. Charges for publishing messages to the exchange may apply. For more information, see https://aws.amazon.com/sqs.
6. Proceed with creating the cluster as described in Plan and Configure Clusters (p. 27).

To enable the debugging tool using the AWS CLI

To enable debugging using the AWS CLI, type the `create-cluster` subcommand with the `--enable-debugging` parameter. You must also specify the `--log-uri` parameter when enabling debugging.

- To enable debugging using the AWS CLI, type the following command and replace `myKey` with the name of your EC2 key pair.
- Linux, UNIX, and Mac OS X users:

  ```bash
  aws emr create-cluster --name "Test cluster" --ami-version 3.9 --log-uri s3://mybucket/logs/ \n  --enable-debugging --applications Name=Hue Name=Hive Name=Pig \n  --use-default-roles --ec2-attributes KeyName=myKey \n  --instance-type m3.xlarge --instance-count 3
  
  aws emr create-cluster --name "Test cluster" --ami-version 3.9 --log-uri s3://mybucket/logs/ --enable-debugging --applications Name=Hue Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey --instance-type m3.xlarge --instance-count 3
  ```

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

**Note**

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

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

### Debugging Option Information

The following AMIs begin support for debugging in all regions:

- AMI 3.9
- AMI 2.4.11

The Amazon EMR creates an Amazon SQS queue to process debugging data. Message charges may apply. However, Amazon SQS does have Free Tier of up to 1,000,000 requests available. For more information, see the [Amazon SQS detail page](http://docs.aws.amazon.com/sqs/latest/index.html).

Debugging requires the use of roles; your service role and instance profile must allow you to use all Amazon SQS API operations. If your roles are attached to Amazon EMR managed policies, you do not need to do anything to modify your roles. If you have custom roles, you need to add `sqs:*` permissions. For more information, see [Configure IAM Roles for Amazon EMR Permissions to AWS Services](http://docs.aws.amazon.com/emr/latest/UG/iam Roles.html).

### Tag Clusters

It can be convenient to categorize your AWS resources in different ways; for example, by purpose, owner, or environment. You can achieve this in Amazon EMR by assigning custom metadata to your Amazon EMR clusters using tags. A tag consists of a key and a value, both of which you define. For Amazon EMR, the cluster is the resource-level that you can tag. For example, you could define a set of tags for your account's clusters that helps you track each cluster's owner or identify a production cluster versus a testing cluster. We recommend that you create a consistent set of tags to meet your organization requirements.

When you add a tag to an Amazon EMR cluster, the tag is also propagated to each active Amazon EC2 instance associated with the cluster. Similarly, when you remove a tag from an Amazon EMR cluster, that tag is removed from each associated active Amazon EC2 instance.

**Important**

Use the Amazon EMR console or CLI to manage tags on Amazon EC2 instances that are part of a cluster instead of the Amazon EC2 console or CLI, because changes that you make in Amazon EC2 do not synchronize back to the Amazon EMR tagging system.

You can identify an Amazon EC2 instance that is part of an Amazon EMR cluster by looking for the following system tags. In this example, `CORE` is the value for the instance group role and `j-12345678` is an example job flow (cluster) identifier value:
Tag Restrictions

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

The following basic restrictions apply to tags:

- 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](http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMAZON_EC2_USING_TAGS.html) in the Amazon EC2 User Guide for Linux Instances. For more information about tagging using the Amazon EC2 API or command line, see [API and CLI Overview](http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMAZON_EC2_USING_TAGS.html) in the Amazon EC2 User Guide for Linux Instances.

Tag Resources for Billing

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.
You can use tags for organizing your AWS bill to reflect your own cost structure. To do this, sign up to get your AWS account bill with tag key values included. You can then organize your billing information by tag key values, to see the cost of your combined resources. Although Amazon EMR and Amazon EC2 have different billing statements, the tags on each cluster are also placed on each associated instance so you can use tags to link related Amazon EMR and Amazon EC2 costs.

For example, you can tag several resources with a specific application name, and then organize your billing information to see the total cost of that application across several services. For more information, see Cost Allocation and Tagging in the AWS Billing and Cost Management User Guide.

Add Tags to a New Cluster

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

You can add tags to a cluster while you are creating it.

To add tags when creating a new cluster using the console

1. In the Amazon EMR console, select the Cluster List page and click Create cluster.
2. On the Create Cluster page, in the Tags section, click the empty field in the Key column and type the name of your key.
   
   When you begin typing the new tag, another tag line automatically appears to provide space for the next new tag.
3. Optionally, click the empty field in the Value column and type the name of your value.
4. Repeat the previous steps for each tag key/value pair to add to the cluster. When the cluster launches, any tags you enter are automatically associated with the cluster.

To add tags when creating a new cluster using the AWS CLI

The following example demonstrates how to add a tag to a new cluster using the AWS CLI. To add tags when you create a cluster, type the create-cluster subcommand with the --tags parameter.

- To add a tag named costCenter with key value marketing when you create a cluster, type the following command and replace myKey with the name of your EC2 key pair.

  • Linux, UNIX, and Mac OS X users:

    ```
    aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applications Name=Hue Name=Hive Name=Pig " --tags "costCenter=marketing" --use-default-roles --ec2-attributes KeyName=myKey " --instance-type m3.xlarge --instance-count 3
    ```

  • Windows users:

    ```
    aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applications Name=Hue Name=Hive Name=Pig " --tags "costCenter=marketing" --use-default-roles --ec2-attributes KeyName=myKey " --instance-type m3.xlarge --instance-count 3
    ```

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

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

You can also add tags to an existing cluster.

To add tags to an existing cluster using the console

1. In the Amazon EMR console, select the Cluster List page and click a cluster to which to add tags.
2. On the Cluster Details page, in the Tags field, click View All/Edit.
3. On the View All/Edit page, click Add.
4. Click the empty field in the Key column and type the name of your key.
5. Optionally, click the empty field in the Value column and type the name of your value.
6. With each new tag you begin, another empty tag line appears under the tag you are currently editing. Repeat the previous steps on the new tag line for each tag to add.

To add tags to a running cluster using the AWS CLI

The following example demonstrates how to add tags to a running cluster using the AWS CLI. Type the add-tags subcommand with the --tag parameter to assign tags to a resource identifier (cluster ID). The resource ID is the cluster identifier available via the console or the list-clusters command.

```
aws emr add-tags --resource-id j-KT4XXXXXXXX1NM --tag "costCenter=marketing" --tag "other=accounting"
```

Note
The add-tags subcommand currently accepts only one resource ID.

• To add two tags to a running cluster (one with a key named production with no value and the other with a key named costCenter with a value of marketing) type the following command and replace j-KT4XXXXXXXX1NM with your cluster ID.

```
aws emr add-tags --resource-id j-KT4XXXXXXXX1NM --tag "production" --tag "costCenter=marketing"
```

Note
When tags are added using the AWS CLI, there is no output from the command.

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

View Tags on a Cluster

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.
If you would like to see all the tags associated with a cluster, you can view them in the console or at the CLI.

**To view the tags on a cluster using the console**

1. In the Amazon EMR console, select the **Cluster List** page and click a cluster to view tags.
2. On the **Cluster Details** page, in the **Tags** field, some tags are displayed here. Click **View All/Edit** to display all available tags on the cluster.

**To view the tags on a cluster using the AWS CLI**

To view the tags on a cluster using the AWS CLI, type the `describe-cluster` subcommand with the `--query` parameter.

- To view a cluster's tags, type the following command and replace `j-KT4XXXXXXXX1NM` with your cluster ID.

  ```bash
  aws emr describe-cluster --cluster-id j-KT4XXXXXXXX1NM --query Cluster.Tags
  ```

  The output displays all the tag information about the cluster similar to the following:

<table>
<thead>
<tr>
<th>Value: accounting</th>
<th>Value: marketing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key: other</td>
<td>Key: costCenter</td>
</tr>
</tbody>
</table>

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

---

**Remove Tags from a Cluster**

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

If you no longer need a tag, you can remove it from the cluster.

**To remove tags from a cluster using the console**

1. In the Amazon EMR console, select the **Cluster List** page and click a cluster from which to remove tags.
2. On the **Cluster Details** page, in the **Tags** field, click **View All/Edit**.
3. In the **View All/Edit** dialog box, click the **X** icon next to the tag to delete and click **Save**.
4. (Optional) Repeat the previous step for each tag key/value pair to remove from the cluster.

**To remove tags from a cluster using the AWS CLI**

To remove tags from a cluster using the AWS CLI, type the `remove-tags` subcommand with the `--tag-keys` parameter. When removing a tag, only the key name is required.

- To remove a tag from a cluster, type the following command and replace `j-KT4XXXXXXXX1NM` with your cluster ID.

  ```bash
  aws emr remove-tags --resource-id j-KT4XXXXXXXX1NM --tag-keys "costCenter"
  ```
Drivers and Third-Party Application Integration

This documentation is for AMI versions 2.x and 3.x of 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 (p. 176)
- Parse Data with HParser (p. 176)
- Using the MapR Distribution for Hadoop (p. 177)

Use Business Intelligence Tools with Amazon EMR

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

You can use popular business intelligence tools like Microsoft Excel, MicroStrategy, QlikView, and Tableau with Amazon EMR to explore and visualize your data. Many of these tools require an ODBC (Open Database Connectivity) or JDBC (Java Database Connectivity) driver. You can download and install the necessary drivers from the links below:

- https://s3.amazonaws.com/amazon-odbc-jdbc-drivers/public/AmazonHiveJDBC_1.0.4.1004.zip

For more information about how you would connect a business intelligence tool like Microsoft Excel to Hive, go to http://cdn.simba.com/products/Hive/doc/Simba_Hive_ODBC_Quickstart.pdf.

Parse Data with HParser

Informatica's HParser is a tool you can use to extract data stored in heterogeneous formats and convert it into a form that is easy to process and analyze. For example, if your company has legacy stock trading information stored in custom-formatted text files, you could use HParser to read the text files and extract the relevant data as XML. In addition to text and XML, HParser can extract and convert data stored in proprietary formats such as PDF and Word files.

HParser is designed to run on top of the Hadoop architecture, which means you can distribute operations across many computers in a cluster to efficiently parse vast amounts of data. Amazon EMR makes it easy...
to run Hadoop in the Amazon Web Services (AWS) cloud. With Amazon EMR you can set up a Hadoop cluster in minutes and automatically terminate the resources when the processing is complete.

In our stock trade information example, you could use HParser running on top of Amazon EMR to efficiently parse the data across a cluster of machines. The cluster will automatically shut down when all of your files have been converted, ensuring you are only charged for the resources used. This makes your legacy data available for analysis, without incurring ongoing IT infrastructure expenses.

The following tutorial walks you through the process of using HParser hosted on Amazon EMR to process custom text files into an easy-to-analyze XML format. The parsing logic for this sample has been defined for you using HParser, and is stored in the transformation services file (services_basic.tar.gz). This file, along with other content needed to run this tutorial, has been preloaded onto Amazon Simple Storage Service (Amazon S3) at s3n://elasticmapreduce/samples/informatica/. You will reference these files when you run the HParser job.

For information about how to run HParser on Amazon EMR, see Parse Data with HParser on Amazon EMR.

For more information about HParser and how to use it, go to http://www.informatica.com/us/products/b2b-data-exchange/hparser/.

Using the MapR Distribution for Hadoop

MapR is a third-party application offering an open, enterprise-grade distribution that makes Hadoop easier to use and more dependable. For ease of use, MapR provides network file system (NFS) and open database connectivity (ODBC) interfaces, a comprehensive management suite, and automatic compression. For dependability, MapR provides high availability with a self-healing no-NameNode architecture, and data protection with snapshots, disaster recovery, and with cross-cluster mirroring. For more information about MapR, go to http://www.mapr.com/.

There are several editions of MapR available on Amazon EMR:

• **M3 Edition** (versions 4.0.2, 3.1.1, 3.0.3, 3.0.2, 2.1.3)—The free version of a complete distribution for Hadoop. M3 delivers a fully random, read-write–capable platform that supports industry-standard interfaces (e.g., NFS, ODBC), and provides management, compression, and performance advantages.

• **M5 Edition** (versions 4.0.2, 3.1.1, 3.0.3, 3.0.2, 2.1.3)—The complete distribution for Apache Hadoop that delivers enterprise-grade features for all file operations on Hadoop. M5 features include mirroring, snapshots, NFS HA, and data placement control. For more information, including pricing, see MapR on Amazon EMR Detail Page.

• **M7 Edition** (versions 4.0.2, 3.1.1, 3.0.3, 3.0.2)—The complete distribution for Apache Hadoop that delivers ease of use, dependability, and performance advantages for NoSQL and Hadoop applications. M7 provides scale, strong consistency, reliability, and continuous low latency with an architecture that does not require compactions or background consistency checks. For more information, including pricing, see MapR on Amazon EMR Detail Page.

**Note**

For enterprise-grade reliability and consistent performance for Apache HBase applications, use the MapR M7 edition.

In addition, MapR does not support Ganglia and debugging and the MapR M3 and M5 editions on Amazon EMR do not support Apache HBase.

The version of MapR available in Amazon EMR that supports Hadoop 2.x is 4.0.2, and it is only available with Amazon EMR AMI 3.3.2

Launch an Amazon EMR cluster with MapR using the console

You can launch any standard cluster on a MapR distribution by specifying MapR when you set the Hadoop version.
To launch an Amazon EMR cluster with MapR using the console

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. Choose Create cluster.
3. In the Software Configuration section, verify the fields according to the following table.

<table>
<thead>
<tr>
<th>Field</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hadoop distribution</td>
<td>Choose MapR. This determines which version of Hadoop to run on your</td>
</tr>
<tr>
<td></td>
<td>cluster. For more information about the MapR distribution for Hadoop,</td>
</tr>
<tr>
<td></td>
<td>see Using the MapR Distribution for Hadoop (p. 177).</td>
</tr>
<tr>
<td>Edition</td>
<td>Choose the MapR edition that meets your requirements. For more</td>
</tr>
<tr>
<td></td>
<td>information, see Using the MapR Distribution for Hadoop (p. 177).</td>
</tr>
<tr>
<td>Version</td>
<td>Choose the MapR version that meets your requirements. For more</td>
</tr>
<tr>
<td></td>
<td>information, see Versions (p. 179).</td>
</tr>
<tr>
<td>Arguments (Optional)</td>
<td>Specify any additional arguments to pass to MapR to meet your</td>
</tr>
<tr>
<td></td>
<td>requirements. For more information, see Arguments (p. 180).</td>
</tr>
</tbody>
</table>

4. Click Done and proceed with creating the cluster as described in Plan and Configure Clusters (p. 27).

Launch a Cluster with MapR Using the AWS CLI

To launch an Amazon EMR cluster with MapR using the AWS CLI

Note
You cannot yet launch MapR clusters when specifying the --release-label parameter.

To launch a cluster with MapR using the AWS CLI, type the `create-cluster` command with the --applications parameter.

- To launch a cluster with MapR, m7 edition, type the following command and replace `myKey` with the name of your EC2 key pair.

  - Linux, UNIX, and Mac OS X users:

    ```
    aws emr create-cluster --name "MapR cluster" --applications Name=Hive Name=Pig \
    Name=MapR,Args=-edtion,m7,-version,4.0.2 --ami-version 3.3.2 \
    --use-default-roles --ec2-attributes KeyName=myKey \
    --instance-type m1.xlarge --instance-count 3
    ```

  - Windows users:
aws emr create-cluster --name "MapR cluster" --applications Name=Hive Name=Pig Name=MapR,Args=--edition,m7,--version,3.1.1 --ami-version 2.4 --use-default-roles --ec2-attributesKeyName=myKey --instance-type m1.xlarge --instance-count 3

**Note**
The version of MapR available in Amazon EMR that supports Hadoop 2.x is 4.0.2, and it is only available with Amazon EMR AMI 3.3.2

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

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

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

**MapR Editions, Versions, and Arguments**

MapR supports several editions, versions, and arguments that you can pass to it using the Amazon EMR console or CLI. The following examples use the Amazon EMR CLI, however Amazon EMR provides an equivalent for each that you can use with the console.

For more information about launching clusters using the CLI, see the instructions for each cluster type in Plan and Configure Clusters (p. 27). For more information about launching clusters using the API, see Use SDKs to Call Amazon EMR APIs (p. 539).

**Editions**
Using the CLI, you can pass the `--edition` parameter to specify the following editions:

- m3
- m5
- m7

The default edition is **m3** if not specified.

**Versions**
You can use `--version` to specify the following versions:

- M3 Edition - 4.0.2, 3.1.1, 3.0.3, 3.0.2, 2.1.3
- M5 Edition - 4.0.2, 3.1.1, 3.0.3, 3.0.2, 2.1.3
- M7 Edition - 4.0.2, 3.1.1, 3.0.3, 3.0.2

**Note**
MapR version 4.0.2 is available with AMI version 3.3.2 only.
Arguments

The `--supported-product` option supports optional arguments. Amazon EMR accepts and forwards the argument list to the corresponding installation script as bootstrap action arguments. The following example shows the syntax for using the `--supported-product` option with or without optional arguments. The fields `key1` and `value1`, etc. represent custom key/value pairs that MapR recognizes and the key named `edition` is shown among them as an example:

```
--supported-product mapr-m3
--supported-product mapr-m3 --args "--key1,value1,--key2,value2"
--supported-product mapr --args "--edition,m3,--key1,value1,--key2,value2"
--supported-product mapr-m3 --args "--edition,m3,--key1,value1,--key2,value2"
```

The following table lists the parameters that MapR reads when users specify them with the `--args` option.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--version</td>
<td>The MapR version.</td>
</tr>
<tr>
<td>--owner-password</td>
<td>The password for the Hadoop owner. The default is hadoop.</td>
</tr>
<tr>
<td>--num-mapr-masters</td>
<td>Any additional master nodes for m5/m7 clusters.</td>
</tr>
<tr>
<td>--mfs-percentage</td>
<td>The proportion of space from the attached storage volumes to be used for MapR file system; the default is 100 (all available space); the minimum is 50. Users who are doing large copies in and out of Amazon S3 storage using s3distcp or another such mechanism may see faster performance by allowing some of the storage space to be used as regular Linux storage. For example: <code>--mfs-percentage,90</code>. The remainder is mounted to S3_TMP_DIR as RAID0 file system.</td>
</tr>
<tr>
<td>--hive-version</td>
<td>The version of the Amazon Hive package. The default is latest. To disable the installation of Amazon Hive, use <code>--hive-version,none</code>.</td>
</tr>
<tr>
<td>--pig-version</td>
<td>The version of the Amazon Pig package. The default is latest. To disable the installation of Amazon Pig, use <code>--pig-version,none</code>.</td>
</tr>
</tbody>
</table>

The following table shows examples of commands, including arguments with the `--args` parameter, and which command MapR executes after interpreting the input.

<table>
<thead>
<tr>
<th>Options</th>
<th>Commands Processed by MapR</th>
</tr>
</thead>
<tbody>
<tr>
<td>--supported-product mapr</td>
<td>--edition m3</td>
</tr>
<tr>
<td>--supported-product mapr-m5</td>
<td>--edition m5</td>
</tr>
<tr>
<td>--supported-product mapr-m3</td>
<td>--edition m3</td>
</tr>
</tbody>
</table>
### Options

<table>
<thead>
<tr>
<th>Options</th>
<th>Commands Processed by MapR</th>
</tr>
</thead>
<tbody>
<tr>
<td>--with-supported-products mapr-m3 (deprecated)</td>
<td>--edition m3</td>
</tr>
<tr>
<td>--with-supported-products mapr-m5 (deprecated)</td>
<td>--edition m5</td>
</tr>
<tr>
<td>--supported-product mapr-m5 --args &quot;--version,1.1&quot;</td>
<td>--edition m5 --version 1.1</td>
</tr>
<tr>
<td>--supported-product mapr-m5 --args &quot;--edition,m3&quot;</td>
<td>N/A - returns an error</td>
</tr>
<tr>
<td>--supported-product mapr --args &quot;--edition,m5&quot;</td>
<td>--edition m5</td>
</tr>
<tr>
<td>--supported-product mapr --args &quot;--version,1.1&quot;</td>
<td>--edition m3 --version 1.1</td>
</tr>
<tr>
<td>--supported-product mapr --args &quot;--edition,m5,--key1 value1&quot;</td>
<td>--edition m5 --key1 value1</td>
</tr>
</tbody>
</table>

**Note**
The `--with-supported-products` option is deprecated and replaced by the `--supported-product` option, which provides the same Hadoop and MapR versions with added support for parameters.

### Enabling MCS access for your Amazon EMR Cluster

After your MapR cluster is running, you need to open port 8453 to enable access to the MapR Control System (MCS) from hosts other than the host that launched the cluster. Follow these steps to open the port.

**To open a port for MCS access**

1. Select your job from the list of jobs displayed in Your Elastic MapReduce Job Flows in the Elastic MapReduce tab of the AWS Management Console, then select the Description tab in the lower pane. Make a note of the Master Public DNS Name value.
2. Click the Amazon EC2 tab in the AWS Management Console to open the Amazon EC2 console.
3. In the navigation pane, select Security Groups under the Network and Security group.
4. In the Security Groups list, select Elastic MapReduce-master.
5. In the lower pane, click the Inbound tab.

   **Note**
The standard MapR port is 8443. Use port number 8453 instead of 8443 when you use the MapR REST API calls to MapR on an Amazon EMR cluster.

7. Click Add Rule, then click Apply Rule Changes.
8. You can now navigate to the master node's DNS address. Connect to port 8453 to log in to the MapR Control System. Use the string hadoop for both login and password at the MCS login screen.

   **Note**
For M5 MapR clusters on Amazon EMR, the MCS web server runs on the primary and secondary CLDB nodes, giving you another entry point to the MCS if the primary fails.

### Testing Your Cluster

This section explains how to test your cluster by performing a word count on a sample input file.
To create a file and run a test job

1. Connect to the master node with SSH as user hadoop. Pass your .pem credentials file to ssh with the -i flag, as in this example:

   ```
   ssh -i /path_to_pemfile/credentials.pem hadoop@masterDNS.amazonaws.com
   ```

2. Create a simple text file:

   ```
   cd /mapr/MapR_EMR.amazonaws.com
   mkdir in
   echo "the quick brown fox jumps over the lazy dog" > in/data.txt
   ```

3. Run the following command to perform a word count on the text file:

   ```
   hadoop jar /opt/mapr/hadoop/hadoop-0.20.2/hadoop-0.20.2-dev-examples.jar wordcount /mapr/MapR_EMR.amazonaws.com/in/ /mapr/MapR_EMR.amazonaws.com/out/
   ```

As the job runs, you should see terminal output similar to the following:

```
12/06/09 00:00:37 INFO fs.JobTrackerWatcher: Current running JobTracker is:
ip10118194139.ec2.internal/10.118.194.139:9001
12/06/09 00:00:37 INFO input.FileInputFormat: Total input paths to process : 1
12/06/09 00:00:37 INFO mapred.JobClient: Running job: job_201206082332_0004
12/06/09 00:00:38 INFO mapred.JobClient: map 0% reduce 0%
12/06/09 00:00:50 INFO mapred.JobClient: map 100% reduce 0%
12/06/09 00:00:57 INFO mapred.JobClient: map 100% reduce 100%
12/06/09 00:00:58 INFO mapred.JobClient: Job complete: job_201206082332_0004
12/06/09 00:00:58 INFO mapred.JobClient: Counters: 25
12/06/09 00:00:58 INFO mapred.JobClient: Job Counters
12/06/09 00:00:58 INFO mapred.JobClient: Launched reduce tasks=1
12/06/09 00:00:58 INFO mapred.JobClient: Aggregate execution time of mappers(ms)=6193
12/06/09 00:00:58 INFO mapred.JobClient: Total time spent by all reduces waiting after
reserving slots (ms)=0
12/06/09 00:00:58 INFO mapred.JobClient: Total time spent by all maps waiting after
reserving slots (ms)=0
12/06/09 00:00:58 INFO mapred.JobClient: Launched map tasks=1
12/06/09 00:00:58 INFO mapred.JobClient: Aggregate execution time of reducers(ms)=4875
12/06/09 00:00:58 INFO mapred.JobClient: FileSystemCounters
12/06/09 00:00:58 INFO mapred.JobClient: MAPRFS_BYTES_READ=385
12/06/09 00:00:58 INFO mapred.JobClient: MAPRFS_BYTES_WRITTEN=276
12/06/09 00:00:58 INFO mapred.JobClient: FILE_BYTES_WRITTEN=94449
12/06/09 00:00:58 INFO mapred.JobClient: Map input records=1
12/06/09 00:00:58 INFO mapred.JobClient: Reduce shuffle bytes=94
12/06/09 00:00:58 INFO mapred.JobClient: Spilled Records=16
12/06/09 00:00:58 INFO mapred.JobClient: Reduce output records=8
12/06/09 00:00:58 INFO mapred.JobClient: CPU_MILLISECONDS=1530
12/06/09 00:00:58 INFO mapred.JobClient: Combine input records=9
12/06/09 00:00:58 INFO mapred.JobClient: SPLIT_RAW_BYTES=125
12/06/09 00:00:58 INFO mapred.JobClient: Reduce input records=8
12/06/09 00:00:58 INFO mapred.JobClient: Combine output records=8
12/06/09 00:00:58 INFO mapred.JobClient: PHYSICAL_MEMORY_BYTES=329244672
12/06/09 00:00:58 INFO mapred.JobClient: Reduce output records=8
12/06/09 00:00:58 INFO mapred.JobClient: VIRTUAL_MEMORY_BYTES=3252969472
12/06/09 00:00:58 INFO mapred.JobClient: Map output records=9
12/06/09 00:00:58 INFO mapred.JobClient: GC time elapsed (ms)=1
```

4. Check the /mapr/MapR_EMR.amazonaws.com/out directory for a file named `part-r-00000` with the results of the job.

182
Tutorial: Launch an Amazon EMR Cluster with MapR M7

This tutorial guides you through launching an Amazon EMR cluster featuring the M7 edition of the MapR distribution for Hadoop. The MapR distribution for Hadoop is a complete Hadoop distribution that provides many unique capabilities, such as industry-standard NFS and ODBC interfaces, end-to-end management, high reliability, and automatic compression. You can manage a MapR cluster through the web-based MapR Control System, the command line, or a REST API. M7 provides enterprise-grade capabilities such as high availability, snapshot and mirror volumes, and native MapR table functionality on MapR-FS, enabling responsive HBase-style flat table databases compatible with snapshots and mirroring. It provides a single platform for storing and processing unstructured and structured data, integrated with existing infrastructure, applications, and tools.

Note
To use the commands in this tutorial, download and install the AWS CLI. For more information see Installing the AWS CLI in the AWS Command Line Interface User Guide.

1. To launch a cluster with MapR, m7 edition, type the following command and replace myKey with the name of your EC2 key pair.

   - Linux, UNIX, and Mac OS X users:

     ```
     aws emr create-cluster --name "MapR cluster" --applications Name=Hive Name=Pig \
     Name=MapR,Args=--edition,m7,--version,4.0.2 --ami-version 3.3.2 \
     --use-default-roles --ec2-attributes KeyName=myKey \
     --instance-type m1.xlarge --instance-count 3
     ```

   - Windows users:

     ```
     aws emr create-cluster --name "MapR cluster" --applications Name=Hive Name=Pig \
     Name=MapR,Args=--edition,m7,--version,3.1.1 --ami-version 2.4 --use-default-roles --ec2-attributes KeyName=myKey --instance-type m1.xlarge --instance-count 3
     ```

Note
The versions of MapR available in Amazon EMR do not currently support Hadoop 2.x. When specifying the --ami-version, use a Hadoop 1.x AMI.

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

Note
If you have not previously created the default EMR service role and EC2 instance profile, type aws emr create-default-roles to create them before typing the create-cluster subcommand.
After you run the command, the cluster takes between five and ten minutes to start. The `aws emr list-clusters` command shows your cluster in the STARTING and BOOTSTRAPPING states before entering the WAITING state.

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

2. Retrieve the cluster ID and then use SSH to connect to the cluster. In the following commands, replace `j-2AL4XXXXXX5T9` with your cluster ID, and replace `~/mykeypair.key` with the path to and filename of your key pair private key file.

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

**Note**
For more information about accessing a cluster with SSH, see *Connect to the Master Node Using SSH* (p. 457).

3. MapR provides volumes as a way to organize data and manage cluster performance. A volume is a logical unit that allows you to apply policies to a set of files, directories, and sub-volumes. You can use volumes to enforce disk usage limits, set replication levels, establish ownership and accountability, and measure the cost generated by different projects or departments. Create a volume for each user, department, or project. You can mount volumes under other volumes to build a structure that reflects the needs of your organization. The volume structure defines how data is distributed across the nodes in your cluster.

Run the following command after connecting to your cluster over SSH to create a volume:

```
# maprcli volume create -name tables -replicationtype low_latency -path /tables
```

4. The M7 Edition of the MapR distribution for Hadoop enables you to create and manipulate tables in many of the same ways that you create and manipulate files in a standard UNIX file system. In the M7 Edition, tables share the same namespace as files and are specified with full path names.

a. Create a table with the following command:

```
$ echo "create '/tables/user_table', 'family' " | hbase shell
```

b. List tables with the following command:

```
$ hadoop fs -ls /tables
Found 1 items
trwxr-xr-x 3 root root 2 2013-04-16 22:49 /tables/user_table
$ ls /mapr/MapR_EMR.amazonaws.com/tables
user_table
```

c. Move or rename tables using the following command:

```
hadoop fs -mv /tables/user_table /tables/usertable
```

5. A snapshot is a read-only image of a volume at a specific point in time. Snapshots are useful any time you need to be able to roll back to a known good data set at a specific point in time.

a. From the HBase shell, add a row to the newly-created table:

```
$ hbase shell
```
b. **Create the snapshot:**

```
$ maprcli volume snapshot create -volume tables -snapshotname mysnap
```

c. **Change the table:**

```
hbase(main):002:0> put '/tables/usertable', 'row_1', 'family:location', 'San Jose'
```

d. **Snapshots are stored in a .snapshot directory. Scan the table from the snapshot and the current table to see the difference:**

```
hbase shell >
scan '/tables/usertable'
ROW COLUMN+CELL
row_1 column=family:child, timestamp=1366154016042, value=username
row_1 column=family:home, timestamp=1366154055043, value=San Jose
1 row(s) in 0.2910 seconds
scan '/tables/.snapshot/mysnap/usertable'
ROW COLUMN+CELL
row_1 column=family:child, timestamp=1366154016042, value=username
1 row(s) in 0.2320 seconds
```

6. **Test high availability:**

a. **List the current regions on your system.**

```
$ maprcli table region list -path /tables/usertable
secondarynodes scans primarynode puts
startkey gets lastheartbeat endkey
ip-10-191-5-21.ec2.internal, ip-10-68-37-140.ec2.internal ...
ip-10-4-74-175.ec2.internal ...
+INFINITY ... 0 INFINITY
```

b. **Restart the primary node for one of the regions. Make sure that the primary node is not the access point to the cluster. Restarting your access point will result in loss of cluster access and terminate your YCSB client.**

Connect to the cluster with SSH and restart the node with the following command:

```
$ ssh -i /Users/username/downloads/MyKey_Context.pem
hadoop@ec2-23-20-100-174.compute-1.amazonaws.com
$ sudo /sbin/reboot
```

**Note**

The restart will take 15 to 30 seconds to complete.

c. **After the restart is complete, list your new regions to see that the former primary node is now listed as secondary.**

```
$ maprcli table region list -path /tables/usertable
secondarynodes scans primarynode puts
startkey gets lastheartbeat endkey
ip-10-191-5-21.ec2.internal, ip-10-68-37-140.ec2.internal ...
ip-10-4-74-175.ec2.internal ...
+INFINITY ... 0 INFINITY
```
7. To open the MapR Control System page, navigate to the address https://hostname.compute-1.amazonaws.com:8453. The username and password for the default installation are both hadoop. The URL for your node's hostname appears in the message-of-the-day that displays when you first log in to the node over SSH.

The Nodes view displays the nodes in the cluster, by rack. The Nodes view contains two panes: the Topology pane and the Nodes pane. The Topology pane shows the racks in the cluster. Selecting a rack displays that rack's nodes in the Nodes pane to the right. Selecting Cluster displays all the nodes in the cluster. Clicking any column name sorts data in ascending or descending order by that column. If your YCSB job is still running, you can see the put streams continuing from the Nodes view.
Security

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

- **AWS Identity and Access Management (IAM) policies** allow or deny permissions for IAM users and groups to perform actions. Policies can be combined with tagging to control access on a cluster-by-cluster basis. For more information, see Use IAM Policies to Allow and Deny User Permissions (p. 187).
- **Kerberos** can be set up to provide strong authentication through secret-key cryptography. For more information, see Use Kerberos Authentication (p. 194).
- **Secure Socket Shell (SSH)** provides a secure way for users to connect to the command line on cluster instances. It also provides tunneling to view Web interfaces that applications run on the master node. Clients can authenticate using Kerberos or an Amazon EC2 key pair. For more information, see Use an Amazon EC2 Key Pair for SSH Credentials (p. 206) and Connect to the Cluster (p. 456).
- **Data encryption** helps protect data at rest and in transit. For more information, see Encrypt Data in Transit and at Rest (p. 207).
- **EMRFS Authorization for Data in Amazon S3** allows you to control whether S3 files can be accessed from within EMR based on user, group, or the location of EMRFS data in Amazon S3. For more information, see EMRFS Authorization for Data in Amazon S3 (p. 214).
- **Security groups** act as a virtual firewall for Amazon EMR cluster instances, limiting inbound and outbound network traffic. For more information, see Control Network Traffic with Security Groups (p. 216).
- **Security configurations** are templates for security configurations so that you can conveniently re-use a security setup whenever you create a cluster. For more information, see Use Security Configurations to Set Up Cluster Security (p. 223).
- **The Amazon EMR service role, instance profile, and service-based role** control how Amazon EMR is able to access other AWS services. For more information, see Configure IAM Roles for Amazon EMR Permissions to AWS Services (p. 234).

Use IAM Policies to Allow and Deny User Permissions

Amazon EMR supports AWS Identity and Access Management (IAM) policies. IAM is a web service that enables AWS customers to manage users and their permissions. You can use IAM to create policies and attach them to principals, such as users and groups. The policies grant or deny permissions and determine what actions a user can perform with Amazon EMR and other AWS resources. For example, you can allow a user to view EMR clusters in an AWS account but not create or delete them. In addition,
you can tag EMR clusters and then use the tags to apply fine-grained permissions to users on individual clusters or a group of clusters that share the same tag.

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

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

Topics
- Amazon EMR Actions in User-Based IAM Policies (p. 188)
- Use Managed Policies for User Access (p. 188)
- Use Cluster Tagging with IAM Policies for Cluster-Specific Control (p. 190)

Amazon EMR Actions in User-Based IAM Policies

In IAM user-based policies for Amazon EMR, all Amazon EMR actions are prefixed with the lowercase elasticmapreduce element. You can specify the "elasticmapreduce:*" key, using the wildcard character (*), to specify all actions related to Amazon EMR, or you can allow a subset of actions, for example, "elasticmapreduce:Describe*". You can also explicitly specify individual Amazon EMR actions, for example "elasticmapreduce:DescribeCluster". For a complete list of Amazon EMR actions, see the API action names in the Amazon EMR API Reference. Because Amazon EMR relies on other services such as Amazon EC2 and Amazon S3, users need to be allowed a subset of permissions for these services as well. For more information, see IAM Managed Policy for Full Access (p. 189).

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

elasticmapreduce:ListClusters

For more information about permissions and policies, see Access Management in the IAM User Guide.

Amazon EMR does not support resource-based and resource-level policies, but you can use the Condition element (also called the Condition block) to specify fine-grained access control based on cluster tags. For more information, see Use Cluster Tagging with IAM Policies for Cluster-Specific Control (p. 190). Because Amazon EMR does not support resource-based or resource-level policies, the Resource element always has a wildcard value.

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

Use Managed Policies for User Access

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

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

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

IAM Managed Policy for Full Access

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

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

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

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

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

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

---

**Note**

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

**IAM Managed Policy for Read-Only Access**

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

**Note**

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

```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": "*"
      }
   ]
}
```
Important
The condition context keys apply only to those Amazon EMR API actions that accept `ClusterID` as a request parameter.

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

Example Amazon EMR Policy Statements

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

Allow Actions Only on Clusters with Specific Tag Values

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

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

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

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt14793345241244",
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:DescribeCluster",
        "elasticmapreduce:ListSteps",
        "elasticmapreduce:TerminateJobFlows",
        "elasticmapreduce:ListBootstrapActions",
        "elasticmapreduce:DescribeStep"
      ],
      "Resource": ["*"],
      "Condition": {
        "StringEquals": {
          "elasticmapreduce:ResourceTag/department": "dev"
        }
      }
    }
  ]
}
```
You can also specify multiple tag values using a condition operator. For example, to allow all actions on clusters where the `department` tag contains the value `dev` or `test`, you could replace the condition block in the earlier example with the following.

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

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

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1479334524000",
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:RunJobFlow",
        "iam:PassRole"
      ],
      "Resource": ["*"]
    },
    "condition": {
      "StringEquals": {
        "elasticmapreduce:RequestTag/department": "dev"
      }
    }
  ]
}
```

In the following example, the EMR actions that allow the addition and removal of tags is combined with a `StringNotEquals` operator specifying the `dev` tag we’ve seen in earlier examples. The effect of this policy is to deny a user the permission to add or remove any tags on EMR clusters that are tagged with a `department` tag that contains the `dev` value.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "elasticmapreduce:AddTags",
        "elasticmapreduce:RemoveTags"
      ]
    }
  ]
}
```
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.

```json
"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.

```json
{
"Version": "2012-10-17",
"Statement": [
  {
    "Action": [
      "elasticmapreduce:RunJobFlow",
      "iam:PassRole"
    ],
    "Condition": {
      "Null": {
        "elasticmapreduce:RequestTag/department": "false"
      }
    },
    "Effect": "Allow",
    "Resource": [
      "*
    ]
  }
}
```

Require Users to Add Tags When Creating a Cluster

The following policy statement allows a user to create an EMR cluster only if the cluster will have a `department` tag that contains the value `dev` when it is created.
Use Kerberos Authentication

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

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

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

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

Topics

- Supported Applications (p. 195)
- Configure Kerberos (p. 195)
- Configure a Cluster-Dedicated KDC (p. 200)
- Configure a Cross-Realm Trust (p. 202)
Supported Applications

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

No web user interfaces hosted on the cluster are Kerberized.

- HDFS
- YARN
- Tez
- Hadoop MapReduce
- Hive
- Hive JDBC is not Kerberized. Establish JDBC authentication using cluster settings.
- Do not enable Hive with LDAP authentication. This may cause issues communicating with Kerberized YARN.
- HCatalog
- Spark
- Oozie
- 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.
- Hbase
- Zeppelin
- Zeppelin is only configured to use Kerberos with the Spark interpreter. It is not configured for other interpreters.
- Zeppelin impersonation with Kerberos is not supported. All users logged in to Zeppelin use the same Zeppelin user principal to run Spark jobs and authenticate to YARN.
- Zookeeper
- Zookeeper client is not supported.

Configure Kerberos

Set up Kerberos on Amazon EMR by following these steps.

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

You can create a security configuration that specifies Kerberos attributes using the EMR console, the AWS CLI, or the EMR API. The security configuration can also contain other security options, such as encryption. For more information, see Create a Security Configuration (p. 223). When you create a Kerberized cluster, you specify the security configuration together with Kerberos attributes that are specific to the cluster. You can't specify one set without the other or an error occurs.
The following Kerberos parameters are set using the security configuration:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable Kerberos</td>
<td>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.</td>
</tr>
<tr>
<td>Ticket Lifetime</td>
<td>The period for which a Kerberos ticket issued by the cluster-dedicated KDC is valid. Ticket lifetimes are limited for security reasons. Cluster applications and services auto-renew tickets after they expire. Users who connect to the cluster over SSH using Kerberos credentials need to run <code>kinit</code> from the master node command line to renew after a ticket expires.</td>
</tr>
<tr>
<td>Cross-realm trust</td>
<td>If you provide a cross-realm trust configuration, principals (typically users) from another realm are authenticated to clusters that use this configuration. Additional configuration in the other Kerberos realm is also required. For more information, see Configure a Cross-Realm Trust (p. 202).</td>
</tr>
<tr>
<td>Admin server</td>
<td>The fully qualified domain name (FQDN) of the other Kerberos admin server in the trust relationship. The admin server and KDC typically run on the same server. Optionally, you can specify the port used to communicate with Kerberos admin server. If not specified, port 749 is used, which is the Kerberos default.</td>
</tr>
<tr>
<td>KDC server</td>
<td>The fully qualified domain name (FQDN) of the KDC in the other realm of the trust relationship. Optionally, you can specify the port used to communicate with the KDC server. If not specified, port 88 is used, which is the Kerberos default.</td>
</tr>
<tr>
<td>Domain name</td>
<td>The domain name of the other realm in the trust relationship.</td>
</tr>
</tbody>
</table>

The following examples demonstrate the same configurations specified in the EMR console and using a JSON structure for the `create-security-configuration` command from the AWS CLI. The KDC and admin services in the cross-trust realm are hosted on the same server, `ad.domain.com` and the default Kerberos ports are used: 749 for the KDC, and 88 for administrative services. If your application uses customized ports, use the form `ad.domain.com:portnumber`. 
Step 2: Configure Kerberos Attributes for a Cluster

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

The following Kerberos attributes are specified using the cluster configuration:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Realm</td>
<td>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.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>KDC admin password</td>
<td>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.</td>
</tr>
<tr>
<td>Cross-realm trust principal password (optional)</td>
<td>Required when establishing a cross-realm trust. The cross-realm principal password, which must be identical across realms. Use a very strong password.</td>
</tr>
<tr>
<td>AD domain join user (optional)</td>
<td>Required when establishing a cross-realm trust with an Active Directory (AD) domain. This is User logon name of an AD account with sufficient privileges to join computers to the domain. Amazon EMR uses this identity to join the cluster to the domain. For more information see the section called “Step 3: Add User Accounts to the Domain for the EMR Cluster” (p. 203).</td>
</tr>
<tr>
<td>AD domain join password (optional)</td>
<td>The password for the AD user that has sufficient privileges to join the cluster to the AD domain. For more information see the section called &quot;Step 3: Add User Accounts to the Domain for the EMR Cluster&quot; (p. 203).</td>
</tr>
</tbody>
</table>

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

Create Cluster - Advanced Options

Note

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

Example Example JSON Snippet For create-security-configuration

```json
aws emr create-cluster --name "MyKerberosCluster" \ 
--release-label emr-5.10.0 \ 
--instance-type m3.xlarge \ 
--instance-count 3 \ 
--use-default-roles \ 
--ec2-attributes KeyName=MyEC2KeyPair \ 
--security-configuration KerberosSecurityConfig.json \ 
--applications Name=Hadoop Name=Hive Name=Oozie Name=Hue Name=HCatalog Name=Spark \ 
--kerberos-attributes Realm=EC2.INTERNAL,KdcAdminPassword=MyVeryStrongPassword,\ 
CrossRealmTrustPrincipalPassword=MyVeryStrongMatchingPassword,\ 
ADDomainJoinUser=ADUser,ADDomainJoinPassword=MyADUserPassword
```

Step 3: Add Kerberos-Authenticated Users

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

You can add users in either of the following ways:

- Configure a cross-realm trust to authenticate users from a different Kerberos realm, such as an AD directory. For more information, see Configure a Cross-Realm Trust (p. 202).
- Add Linux accounts on the local cluster and add principals to the cluster-dedicated KDC for those accounts. For more information, see Configure a Cluster-Dedicated KDC (p. 200).
Important
The KDC, along with the database of principals, is lost when the master node terminates because the master node uses ephemeral storage. If you create users for SSH connections, we recommend that you establish a cross-realm trust with an external KDC configured for high-availability. Alternatively, if you create users for SSH connections using Linux user accounts, automate the account creation process using bootstrap actions and scripts so that it can be repeated when you create a new cluster.

Configure a Cluster-Dedicated KDC

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

Step 1: Create the Kerberized Cluster

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

```bash
aws emr create-security-configuration --name MyKerberosConfig
    --security-configuration '{"AuthenticationConfiguration": "KerberosConfiguration": "ClusterDedicatedKdc", "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 prior to creating the cluster.

```bash
aws emr create-cluster --name "MyKerberosCluster" 
    --release-label emr-5.10.0 
    --instance-type m3.xlarge 
    --instance-count 3 
    --ec2-attributes InstanceProfile=EMR_EC2_DefaultRole,KeyName=MyEC2KeyPair 
    --service-role EMR_DefaultRole 
    --security-configuration MyKerberosConfig 
    --applications Name=Nadoop Name=Hive Name=Oozie Name=Hue Name=HCatalog Name=Spark 
    --kerberos-attributes Realm=EC2.INTERNAL, KdcAdminPassword=MyClusterKDCAdminPwd 
    --bootstrap-actions Path=s3://mybucket/createlinuxusers.sh
```

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

```bash
#!/bin/bash
sudo adduser user1
sudo adduser user2
sudo adduser user3
```
Step 2: Add Principals to the KDC, Create HDFS User Directories, and Configure SSH

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

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

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

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

```
#!/bin/bash
#Add a principal to the KDC for the master node, using the master node's returned host name
sudo kadmin.local -q "ktadd -k /etc/krb5.keytab host/'`hostname -f`'"
#Declare an associative array of user names and passwords to add
declare -A arr
arr=( [user1]=pwd1 [user2]=pwd2 [user3]=pwd3)
for i in ${!arr[@]}; do
  #Assign plain language variables for clarity
  name=${i}
  password=${arr[$name]}
  # Create principal for sshuser in the master node and require a new password on first logon
  sudo kadmin.local -q "addprinc -pw +needchange $password $name"
  #Add user hdfs directory
  hdfs dfs -mkdir /user/$name
  #Change owner of user's hdfs directory to user
  hdfs dfs -chown $name:$name /user/$name
done
# Enable GSSAPI authentication for SSH and restart SSH service
sudo sed -i '/GSSAPIAuthentication.*$/d' /etc/ssh/sshd_config
sudo sed -i '/GSSAPICleanupCredentials.*$/d' /etc/ssh/sshd_config
/etc/init.d/sshd restart
```

Step 3: Connect Using SSH

The users that you created in the previous steps can now log in to the cluster with the user names and passwords that you specified in the steps above.

Your Linux computer most likely includes an SSH client by default. For example, OpenSSH is installed on most Linux, Unix, and Mac OS X operating systems. You can check for an SSH client by typing `ssh` at the command line. If your computer does not recognize the command, install an SSH client to connect to the master node. The OpenSSH project provides a free implementation of the full suite of SSH tools. For more information, see the OpenSSH website.
For more information about SSH connections, see Connect to the Cluster (p. 456).

To connect to a cluster using Kerberos principals from the cluster-dedicated KDC

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

- Linux users

  Ensure the krb5.conf file on the client matches the cluster-dedicated KDC settings.

  ```
  kinit username
  ssh -K username@MasterPublicDNS
  ```

Configure a Cross-Realm Trust

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

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

Setting Up a Cross-Realm Trust with an AD Domain Controller

Step 1: Set Up the VPC and Subnet (p. 202)
Step 2: Launch and Install the AD Domain Controller (p. 203)
Step 3: Add User Accounts to the Domain for the EMR Cluster (p. 203)
Step 4: Configure an Incoming Trust on the AD Domain Controller (p. 204)
Step 5: Use a DHCP Option Set to Specify the AD Domain Controller as a VPC DNS Server (p. 204)
Step 6: Launch a Kerberized EMR Cluster (p. 204)
Step 7: Create HDFS Users and Set Permissions on the Cluster for AD User Accounts (p. 205)
Step 8: Use SSH to log in to the cluster (p. 206)

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 AD domain controller and resolve its domain name. In these steps, domain-name resolution is provided by referencing the AD 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 AD Domain Controller as a VPC DNS Server (p. 204).

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

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

### Step 2: Launch and Install the AD Domain Controller

1. Launch an EC2 instance based on the Microsoft Windows Server 2016 Base AMI. We recommend an m4.xlarge or better instance type. For more information, see Launching an AWS Marketplace Instance in the Amazon EC2 User Guide for Windows Instances.

2. Connect to the EC2 instance using RDP. For more information, see Connecting to Your Windows Instance in the Amazon EC2 User Guide for Windows Instances.

3. Start Server Manager to install and configure the Active Directory Domain Services role on the server. Promote the sever 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 AD, you can follow the instructions in How to Set Up Active Directory (AD) in Windows Server 2016.

   The instance restarts when you finish.

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

RDP to the AD domain controller to create user accounts in Active Directory Users and Computers for each cluster user. For instructions, see Create a User Account in Active Directory Users and Computers. Make a note of each user's **User logon name**. You need these later when you configure the cluster.
In addition, create a user account with sufficient privileges to join computers to the domain. You specify this account when you create a cluster. Amazon EMR uses it to join cluster instances to the domain. You specify this account and its password in Step 6: Launch a Kerberized EMR Cluster (p. 204). To delegate computer join privileges to the user account, we recommend that you create a group with join privileges and then assign the user to the group. For instructions, see Delegating Directory Join Privileges in the AWS Directory Service Administration Guide.

Step 4: Configure an Incoming Trust on the AD Domain Controller

The example commands below create a trust in AD, which is a one-way, incoming, non-transitive, realm trust with the cluster-dedicated KDC. The example we use for the cluster’s realm is `EC2.INTERNAL`. The `passwordt` parameter specifies the `cross-realm principal password`, which you specify along with the cluster `realm` when you create a cluster. The realm name is derived from the default domain name in `us-east-1` for the cluster.

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

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

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

Now that the AD 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.amazon.aws` for other regions. For Domain name servers, you must specify the IP address of the AD 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 AD 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 AD Domain Controller (p. 203).

```
aws emr create-security-configuration --name MyKerberosConfig --security-configuration '{
  "AuthenticationConfiguration": {
    "KerberosConfiguration": {
      "Provider": "ClusterDedicatedKdc",
      "ClusterDedicatedKdcConfiguration": {
        "TicketLifetimeInHours": 24,
        "CrossRealmTrustConfiguration": {
          "Realm": "AD.DOMAIN.COM",
          "Domain": "ad.domain.com",
          "AdminServer": "ad.domain.com",
          "KdcServer": "ad.domain.com"
        }
      }
    }
  }
}
```
2. Create the cluster, specifying the security configuration (in this example, MyKerberosConfig) and the same subnet you created in Step 1: Set Up the VPC and Subnet (p. 202).

Also specify the following cluster-specific kerberos-attributes:

- The realm for the cluster that you specified when you set up the AD domain controller
- The cross-realm trust principal password that you specified as passwordt in Step 4: Configure an Incoming Trust on the AD Domain Controller (p. 204).
- A KdcAdminPassword, which you can use to administer the cluster-dedicated KDC.
- The user logon name and password of the AD account with computer join privileges that you created in Step 4: Configure an Incoming Trust on the AD Domain Controller (p. 204).

The following example launches a kerberized cluster.

```
aws emr create-cluster --name "MyKerberosCluster" \
--release-label emr-5.10.0 \
--instance-type m3.xlarge \
--instance-count 3 \
--ec2-attributes InstanceProfile=EMR_EC2_DefaultRole,KeyId=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,\ 
ADDomainJoinUser=ADUserLogonName,ADDomainJoinPassword=ADUserPassword,\ 
CrossRealmTrustPrincipalPassword=MatchADTrustPwd
```

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

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

To allow your users to login to the cluster to run Hadoop jobs, you must add HDFS user directories for their Linux user accounts, and grant each user ownership of their directory. To do this, we recommend that you run a script saved to Amazon S3 as a cluster step. Alternatively, you can run the commands in the script below from the command line on the master node. Use the EC2 key pair that you specified when you created the cluster to connect to the master node over SSH as the Hadoop user. For more information, see Use an Amazon EC2 Key Pair for SSH Credentials (p. 206).

Run the following command to add a step to the cluster that runs a script, **AddHDFSUsers.sh**.

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

The contents of the file **AddHDFSUsers.sh** is as follows.

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

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

# For each user listed, create an HDFS user directory
# and change ownership to the user
```
for username in ${ADUSERS[@]}; do
dfs df -mkdir /user/$username
dfs df -chown $username:$username /user/$username
done

AD Groups Mapped to Hadoop Groups

Amazon EMR uses System Security Services Daemon (SSD) to map AD groups to Hadoop groups. To confirm group mappings, after you log in to the master node as described in the following step, you can use the dfs groups command to confirm that AD groups to which your AD 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:
dfs groups lijuan. For more information, see groups in the Apache HDFS Commands Guide.

Step 8: Use SSH to log in to the cluster

Users in the AD domain should now be able to log on to the cluster with their domain credentials. Linux users can connect using ssh as shown in the example below, replacing myusername with the user logon name from AD and replacing ec2-xx-xxx-xx-xx.compute-1.amazonaws.com with the Master public DNS value listed on the cluster’s Summary page.

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

Your Linux computer most likely includes an SSH client by default. For example, OpenSSH is installed on most Linux, Unix, and Mac OS X operating systems. You can check for an SSH client by typing ssh at the command line. If your computer does not recognize the command, install an SSH client to connect to the master node. The OpenSSH project provides a free implementation of the full suite of SSH tools. For more information, see the OpenSSH website.

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

For more information about SSH connections, see Connect to the Cluster (p. 456).

Use an Amazon EC2 Key Pair for SSH Credentials

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Amazon EMR can use an Amazon EC2 key pair to authorize SSH client connections to cluster instances. Usually, an SSH connection is used to connect directly to the master node to interact directly with system files and logs, or to view web interfaces that the master instance hosts. Alternatively, with Amazon EMR release version 5.10.0 or later, you can configure Kerberos to authenticate users and SSH connections to the master node. For more information, see Use Kerberos Authentication (p. 194).

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

- For more information about creating an Amazon EC2 key pair, see Amazon EC2 Key Pairs in the Amazon EC2 User Guide for Linux Instances.
Encrypt Data in Transit and at Rest

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

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

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

Topics

- Understanding Encryption Options (p. 207)
- Creating Keys and Certificates for Data Encryption (p. 211)

Understanding Encryption Options

Beginning with Amazon EMR release version 4.8.0, you can use a security configuration to specify settings for Amazon S3 encryption with EMR File System (EMRFS), local disk encryption, and in-transit encryption. You can use a security configuration to encrypt data at-rest, data in-transit, or both. Each security configuration is stored in Amazon EMR rather than in the cluster configuration, so you can easily reuse a configuration to specify data encryption settings whenever a cluster is created. For more information, see Create a Security Configuration (p. 223).

The following diagram shows the different data encryption options available with security configurations. Using a security configuration does not encrypt the EBS root device volume. To do this, use Amazon EMR version 5.7.0 or later and specify a custom AMI with an encrypted root device volume. For more information, see Using a Custom AMI in the Amazon EMR Management Guide.
The following encryption options are also available and are not configured using a security configuration:

- Beginning with Amazon EMR release version 5.7.0, you can specify a custom AMI with an encrypted EBS root device volume. For more information, see Using a Custom AMI.
- If you are using a release version of Amazon EMR that does not support security configurations, you can configure encryption for EMRFS data in Amazon S3 manually. For more information, see Specifying Amazon S3 Encryption Using EMRFS Properties (p. 61).
- Optionally, beginning with Amazon EMR release version 4.1.0 and later, you can choose to configure transparent encryption in HDFS. For more information, see Transparent Encryption in HDFS on Amazon EMR in the Amazon EMR Release Guide.

Data encryption requires keys and certificates. A security configuration gives you the flexibility to choose from several options, including keys managed by AWS Key Management Service, keys managed by Amazon S3, and keys and certificates from custom providers that you supply. When using AWS KMS as your key provider, charges apply for the storage and use of encryption keys. For more information, see AWS KMS Pricing.

Before you specify encryption options, decide on the key and certificate management systems you want to use, so you can first create the keys and certificates or the custom providers that you specify as part of encryption settings.
Amazon S3 encryption and local disk encryption options are specified together when you configure at-rest encryption. You can choose to enable only at-rest encryption, only in-transit encryption, or both.

**At-Rest Encryption for EMRFS data in Amazon S3**

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

**Amazon S3 Server-Side Encryption**

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

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

- **SSE-S3**: Amazon S3 manages keys for you.
- **SSE-KMS**: You use an AWS KMS customer master key (CMK) set up with policies suitable for Amazon EMR. For more information about key requirements for Amazon EMR, see Using AWS KMS Customer Master Keys (CMKs) for Encryption (p. 211). When you use AWS KMS, charges apply for the storage and use of encryption keys. For more information, see AWS KMS Pricing.

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

**Amazon S3 Client-Side Encryption**

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

**Note**

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

**At-rest Encryption for Local Disks**

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

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

- **Open-source HDFS Encryption**: HDFS exchanges data between cluster instances during distributed processing, and also reads from and writes data to instance store volumes and the EBS volumes.
attached to instances. The following open-source Hadoop encryption options are activated when you enable local-disk encryption:

- **Secure Hadoop RPC** is set to "Privacy", which uses Simple Authentication Security Layer (SASL).
- **Data encryption on HDFS block data transfer** is set to true and is configured to use AES 256 encryption.

**Note**
You can activate additional Apache Hadoop encryption by enabling in-transit encryption (see In-Transit Data Encryption (p. 210)). These encryption settings do not activate HDFS transparent encryption, which you can configure manually. For more information, see Transparent Encryption in HDFS on Amazon EMR in the Amazon EMR Release Guide.

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

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

**In-Transit Data Encryption**

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

- **Hadoop** (for more information, see Hadoop in Secure Mode in Apache Hadoop documentation):
  - Hadoop MapReduce Encrypted Shuffle uses TLS.
  - Secure Hadoop RPC is set to "Privacy" and uses SASL (activated in Amazon EMR when at-rest encryption is enabled).
  - Data encryption on HDFS block data transfer uses AES 256 (activated in Amazon EMR when at-rest encryption is enabled in the security configuration).
- **Presto**:
  - Internal communication between Presto nodes uses SSL/TLS (Amazon EMR version 5.6.0 and later only).
- **Tez**:
  - Tez Shuffle Handler uses TLS (tez.runtime.ssl.enable).
- **Spark** (for more information, see Spark security settings):
  - Block transfer service uses SASL and 3DES.
  - External shuffle service uses SASL. Applications that are not set up to use SASL encryption fail to connect to the shuffle service.

You specify the encryption artifacts used for in-transit encryption in one of two ways: either by providing a zipped file of certificates that you upload to Amazon S3, or by referencing a custom Java class that provides encryption artifacts. For more information, see Providing Certificates for In-Transit Data Encryption with Amazon EMR Encryption (p. 213).
Amazon EMR Developer Guide
Creating Keys and Certificates for Data Encryption

Creating Keys and Certificates for Data Encryption

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

Providing Keys for At-Rest Data Encryption with Amazon EMR

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

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

Using AWS KMS Customer Master Keys (CMKs) for Encryption

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

The role for the Amazon EC2 instance profile must have permission to use the CMK you specify. The default role for the instance profile in Amazon EMR is EMR_EC2_DefaultRole. If you use a different role for the instance profile, or you use EMRFS authorization to assume different roles based on the call to Amazon S3, make sure that each role is added as a key user as appropriate. This gives the role permission to use the CMK. For more information, see Using Key Policies in the AWS Key Management Service Developer Guide and Use Default IAM Roles and Managed Policies (p. 235).

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

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

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

1. Sign in to the AWS Management Console and open the AWS Identity and Access Management (IAM) console at https://console.aws.amazon.com/iam/.
2. In the left navigation pane, choose Encryption keys.
3. For Region, choose the appropriate AWS Region. Do not use the region selector in the navigation bar (top right corner).
4. Select the alias of the CMK to modify.
5. On the key details page under Key Users, choose Add.
6. In the Attach dialog box, select the appropriate role. The name of the default role is EMR_EC2_DefaultRole.
7. Choose Attach.

Creating a Custom Key Provider

When using a security configuration, you must specify a different provider class name for local disk encryption and Amazon S3 encryption.
When you create a custom key provider, the application is expected to implement the `EncryptionMaterialsProvider` interface, which is available in the AWS SDK for Java version 1.11.0 and later. The implementation can use any strategy to provide encryption materials. You may, for example, choose to provide static encryption materials or integrate with a more complex key management system.

The `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.

```java
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() {
    }
}
```
Providing Certificates for In-Transit Data Encryption with Amazon EMR Encryption

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

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

Amazon EMR automatically downloads artifacts to each node in the cluster and later uses them to implement the open-source, in-transit encryption features. For more information about available options, see In-Transit Data Encryption (p. 210).

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**

<table>
<thead>
<tr>
<th>File name</th>
<th>Required/optional</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>privateKey.pem</td>
<td>Required</td>
<td>Private key</td>
</tr>
<tr>
<td>certificateChain.pem</td>
<td>Required</td>
<td>Certificate chain</td>
</tr>
<tr>
<td>trustedCertificates.pem</td>
<td>Optional</td>
<td>Required if the provided certificate is not signed by either the Java default trusted root certification authority (CA) or an intermediate CA that can link to the Java default trusted root CA. The Java default trusted root CAs can be found in jre/lib/security/cacerts.</td>
</tr>
</tbody>
</table>

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, you may choose to specify a common name in the certificate configuration that allows access to the cluster by specifying CN=*.ec2.internal in the certificate subject definition. For more information
about Amazon EMR cluster configuration within Amazon VPC, see Select an Amazon VPC Subnet for the Cluster.

The following example demonstrates how to use OpenSSL to generate a self-signed X.509 certificate with a 1024-bit RSA private key that allows access to the issuer's Amazon EMR cluster instances in the US East (N. Virginia) region. This is identified by the *.ec2.internal domain name as the common name. Other optional subject items—such as country (C), state (S), Locale (L), etc.—are specified. Because a self-signed certificate is generated, the example then copies the certificateChain.pem file to the trustedCertificates.pem file. The zip command is then used to create the my-certs.zip file that contains the certificates.

```
# openssl req -x509 -newkey rsa:1024 -keyout privateKey.pem -out certificateChain.pem -days 365 -nodes -subj '/C=US/S=Washington/L=Seattle/O=MyOrg/OU=MyDept/CN=*.ec2.internal'
# cp certificateChain.pem trustedCertificates.pem
# zip -r -X my-certs.zip certificateChain.pem privateKey.pem trustedCertificates.pem
```

EMRFS Authorization for Data in Amazon S3

If you have clusters with multiple users who need different levels of access to EMRFS data in Amazon S3, you can use EMRFS authorization within a security configuration to implement fine-grained access control. EMRFS authorization lets you apply different permissions for access based on the user or group making the request, or the location of EMRFS data in Amazon S3. This lets you allow or deny access permissions more easily, even for IAM users in different AWS accounts. EMRFS authorization is available in Amazon EMR release version 5.10.0 and later. If you use an earlier version of Amazon EMR or have additional authorization requirements, you can create a custom credentials provider instead. For more information, see Creating a Custom Credentials Provider for EMRFS Data in Amazon S3 (p. 61)

How EMRFS Authorization Works

When EMRFS issues a request to Amazon S3, the permissions to access Amazon S3 data are determined by the permissions policies attached to the IAM role for the EC2 instance profile (the default is EMR_EC2_DefaultRole). If a cluster doesn't use EMRFS authorization, the IAM policies attached to this role apply regardless of the user or group making the request through EMRFS, or the location of data in Amazon S3. When you set up EMRFS authorization, you create a security configuration and specify role mappings. Each role mapping specifies an IAM role along with identifiers. The identifiers can be users, groups, or Amazon S3 prefixes that indicate a data location. 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). When EMRFS makes a request to Amazon S3 from a cluster that uses the security configuration, if the request contains an identifier specified in a role mapping, the Amazon EC2 instance profile assumes the IAM role for that request, and the IAM permissions attached to that role apply.

- For more information about creating and specifying a security configuration for a cluster, see Use Security Configurations to Set Up Cluster Security (p. 223).
- For more information about EMRFS, see Using EMR File System (EMRFS) (p. 45).

Set Up EMRFS Authorization

Before you set up EMRFS authorization, 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 role for the instance profile, which is AmazonElasticMapReduceforEC2Role, and edit this policy according to your requirements. For more information, see Use Default IAM Roles and Managed Policies (p. 235). If a role allows access to a location in Amazon S3 that is encrypted using a AWS Key Management Service customer master key (CMK), make sure that the role is specified as a key user. This gives the role permission to use the CMK. For more information, see Using Key Policies in the AWS Key Management Service Developer Guide.

You can create and use multiple roles, and you can specify multiple identifiers and multiple role mappings within a security configuration. You can also specify multiple identifiers within a single role mapping, but they must all be of the same type. When EMRFS makes a request to Amazon S3, role mappings are evaluated in the top-down order that they appear in the security configuration. The EC2 instance profile assumes the role that is specified for the first identifier that matches the request. If an identifier isn't found in any role mapping, the EC2 instance profile assumes the role that you specified when you created the cluster. For this reason, we recommend that the policies you attach to this role limit permissions to Amazon S3.

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

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;EmrFsConfiguration&quot;:</td>
<td>Required for EMRFS authorization. Contains EMRFS authorization configurations.</td>
</tr>
<tr>
<td>&quot;RoleMappings&quot;:</td>
<td>Required for EMRFS authorization. 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. Role mappings consist of the following required parameters:</td>
</tr>
<tr>
<td>&quot;Role&quot;:</td>
<td>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</td>
</tr>
</tbody>
</table>
### Parameter Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;IdentifierType&quot;:</td>
<td>Can be one of the following:</td>
</tr>
<tr>
<td></td>
<td>• &quot;User&quot; 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.</td>
</tr>
<tr>
<td></td>
<td>• &quot;Prefix&quot; specifies that the identifier is an Amazon S3 location. The IAM role is assumed for calls to the location or locations with the specified prefixes. For example, the prefix s3://mybucket/ matches s3://mybucket/mydir and s3://mybucket/yetanotherdir.</td>
</tr>
<tr>
<td></td>
<td>• &quot;Group&quot; 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.</td>
</tr>
<tr>
<td>&quot;Identifiers&quot;:</td>
<td>Specifies one or more identifiers of the appropriate identifier type. Separate multiple identifiers by commas with no spaces.</td>
</tr>
</tbody>
</table>

## Control Network Traffic with Security Groups

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

A security group acts as a virtual firewall for your EC2 instances to control inbound and outbound traffic. For each security group, one set of rules controls the inbound traffic to instances, and a separate set of rules controls outbound traffic. In Amazon EMR, there are two types of security groups for your EC2 instances:

- Amazon EMR–managed security groups for the master and core/task instances: you can choose the default master and core/task security groups (ElasticMapReduce-master, ElasticMapReduce-slave, ElasticMapReduce-Master-Private, ElasticMapReduce-Slave-Private, and ElasticMapReduce-ServiceAccess), or you can specify your own security groups for the master and core/task instances
- Additional security groups: you can specify security groups that apply to your Amazon EC2 instances in addition to the Amazon EMR–managed security groups

Security groups need to be specified at cluster launch using the console, CLI, API, or SDK. Instances in running clusters cannot be assigned new security groups, but you can edit or add rules to existing security groups. Edited rules and new rules added to security groups take effect when the rules are saved.

For more information, see Amazon EC2 Security Groups in the Amazon EC2 User Guide for Linux Instances. For more information about Amazon VPC security groups, see Security Groups for Your VPC in the Amazon VPC User Guide.
Use Amazon EMR–Managed Security Groups

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

When you launch an Amazon EMR cluster, you must specify one Amazon EMR–managed security group for the master instance, one for Amazon EMR–managed security group for the core/task (slave) instances, and optionally, one for the Amazon EMR resources used to manage clusters in private subnets. The core/task instances share the same security group.

You can use the default Amazon EMR–managed security groups for cluster resources or you can choose your own security groups for the master and core/task instances. You cannot combine your own security groups with the default groups.

When you launch a cluster using the console, if the default security groups do not exist, the Amazon EMR–managed security groups fields are populated with Create ElasticMapReduce-master and Create ElasticMapReduce-slave or Create ElasticMapReduce-Master-Private, Create ElasticMapReduce-Slave-Private, and Create ElasticMapReduce-ServiceAccess. If the default security groups exist, the fields are populated with the default security group IDs; for example, Default: sg-01XXXX6a (ElasticMapReduce-master) and Default: sg-07XXXX6c (ElasticMapReduce-slave). To use your own security groups, choose them from the lists. For more information about the default Amazon EMR–managed security group options, see Default Options for Amazon EMR–Managed Security Groups (p. 219).

Amazon EMR-specific inbound and outbound access rules are written to and maintained in the Amazon EMR–managed security groups. Whether you choose the default security groups or your own, Amazon EMR modifies the security group rules to ensure proper communication between instances in a cluster. For more information about the rules written to the Amazon EMR–managed security groups, see Rules in Amazon EMR–Managed Security Groups (p. 219).

If you need to control instance membership in the Amazon EMR–managed security groups, you can create your own security groups for the master and core/task instances. For more information, see Security Groups for Your VPC in the Amazon VPC User Guide. When you create your own security groups, any inbound and outbound rules required for proper intra-cluster communication are written to the groups with some exceptions.

Typically, you would create your own Amazon EMR–managed security groups to isolate clusters so they cannot communicate with each other. This alleviates the need to create separate VPCs or AWS accounts for each of your clusters. For example, to isolate Amazon EMR clusters in a VPC, you create a security group for each cluster's master node, and you create a security group for each cluster's core/task nodes. When you launch a cluster, the rules in your security groups are applied only to the instances in that cluster, effectively preventing instances in separate clusters from communicating with each other.

If you need separate clusters to communicate, and you are using your own security groups, you can create an additional security group containing the rules necessary for the cluster instances to communicate with each other, or you can use the same security groups for both clusters. If you do not specify an additional security group when launching the clusters, and the clusters use different security groups, the clusters cannot communicate with each other unless you modify the rule sets in your master and core/task security groups. For more information about additional security groups, see Configure Additional Security Groups (p. 221).
To specify Amazon EMR–managed security groups using the console

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. Choose Create cluster.
3. In the Security and Access section, in the EC2 Security Groups subsection:
   - For Master, if the default security groups do not exist, the field is populated with Create ElasticMapReduce-master or Create ElasticMapReduce-Master-Private. If the default security groups exist, the field is populated with the default security group ID; for example, Default: sg-01XXXX6a (ElasticMapReduce-master). To specify your own master security group, choose it from the list.
   - For Core & Task, if the default security groups do not exist, the field is populated with Create ElasticMapReduce-slave or Create ElasticMapReduce-Slave-Private. If the default security groups exist, the field is populated with the default security group ID; for example, Default: sg-07XXXX6c (ElasticMapReduce-slave). To specify your own core/task security group, choose it from the list.
   - (Optional) For Service Access to clusters in private subnets, if the default security groups do not exist, the field is populated with Create ElasticMapReduce-ServiceAccess. If the default security groups exist, the field is populated with the default security group ID; for example, Default: sg-4dXXXX34 (ElasticMapReduce-ServiceAccess). To specify your own service access security group, choose it from the list.
4. Proceed with creating the cluster as described in Use an Amazon EC2 Key Pair for SSH Credentials (p. 206).

To specify Amazon EMR–managed security groups using the AWS CLI

Use the create-cluster command with the --emr-managed-master-security-group and --emr-managed-slave-security-group parameters.

If you are using the default options for Amazon EMR–managed security groups, no additional parameters are required. Use the create-cluster command as you normally would using the CLI. If the default security groups do not exist, they are created before your cluster is launched. If they do exist, they are automatically assigned.

**Note**

Amazon EMR–managed security groups are not supported in the Amazon EMR CLI.

1. To launch a cluster using the default security group options, type the following command, replace myKey with the name of your Amazon EC2 key pair.

   ```
   aws emr create-cluster --name "Test cluster" --ami-version 3.10 --applications Name=Hue Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey --instance-type m3.xlarge --instance-count 3
   ```

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

2. To launch a cluster using your own security groups, type the following command, replace myKey with the name of your Amazon EC2 key pair, replace masterSecurityGroupId with the ID of the master security group, and replace slaveSecurityGroupId with the ID of the core/task security group.

   ```
   aws emr create-cluster --name "Test cluster" --ami-version 3.10 --applications Name=Hive Name=Pig --ec2-attributes KeyName=myKey,EmrManagedMasterSecurityGroup=sg-
   ```
Use Amazon EMR–Managed Security Groups

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

3. To retrieve security group information for your cluster, type the following command and replace j-1K48XXXXXXHCB with your cluster ID:

```
aws emr describe-cluster --cluster-id j-1K48XXXXXXHCB
```

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

Default Options for Amazon EMR–Managed Security Groups

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

If you launch an Amazon EMR cluster using the default security groups, two groups are created for public subnets: ElasticMapReduce-master and ElasticMapReduce-slave. For private subnets, three groups are created:

- Create ElasticMapReduce-Master-Private
- Create ElasticMapReduce-Slave-Private
- Create ElasticMapReduce-ServiceAccess

The inbound and outbound access rules written to these groups ensure that the master and core/task instances in a cluster can communicate properly.

In addition, if you launch other Amazon EMR clusters in the same VPC using the default security groups, the instances in those clusters can communicate with the instances in any other Amazon EMR cluster within that VPC whose instances also belong to the same security groups.

You can launch a cluster with the default security groups using the console, the API, the CLI, or the SDK. If you use the default security groups, there is no need to change your existing code or to add parameters to your CLI commands.

You can launch a cluster with the default security groups using the console. If the default security groups do not exist, they are created before your cluster is launched. If they do exist, they are automatically assigned.

Rules in Amazon EMR–Managed Security Groups

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

The tables in this section list the inbound and outbound access rules added to the Amazon EMR–managed security groups. IP address ranges and rules are automatically updated for Amazon EMR–managed security groups.
**Warning**  
We do not recommend creating an inbound rule for any protocol or port that allows inbound traffic from all IP addresses (0.0.0.0/0). This opens access to everyone and creates a security vulnerability.

The following rules are added to Amazon EMR–managed master security groups:

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol</th>
<th>Port Range</th>
<th>Source</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inbound rules</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All ICMP</td>
<td>ICMP</td>
<td>All</td>
<td>The default ElasticMapReduce-master security group ID or the master security group ID that you specify; for example, sg-88XXXXed</td>
<td>Allows inbound traffic from any instance in the ElasticMapReduce-master security group. By default, the master nodes in all Amazon EMR clusters in a single VPC can communicate with each other over any TCP, UDP, or ICMP port. If you choose your own master security group, only master instances in the group can communicate with each other over any TCP, UDP, or ICMP port.</td>
</tr>
<tr>
<td>All TCP</td>
<td>TCP</td>
<td>All</td>
<td></td>
<td></td>
</tr>
<tr>
<td>All UDP</td>
<td>UDP</td>
<td>All</td>
<td></td>
<td></td>
</tr>
<tr>
<td>All ICMP</td>
<td>ICMP</td>
<td>All</td>
<td>The default ElasticMapReduce-slave security group ID or the core/task security group ID that you specify; for example, sg-8bXXXee</td>
<td>Allows inbound traffic from any instance in the ElasticMapReduce-slave security group. By default, the master node accepts inbound communication from any core/task node in any Amazon EMR cluster in a single VPC over any TCP, UDP, or ICMP port. If you choose your own core/task security group, only core/task instances in this group can communicate with the master node over any TCP, UDP, or ICMP port.</td>
</tr>
<tr>
<td>All TCP</td>
<td>TCP</td>
<td>All</td>
<td></td>
<td></td>
</tr>
<tr>
<td>All UDP</td>
<td>UDP</td>
<td>All</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HTTPS</td>
<td>TCP</td>
<td>8443</td>
<td>Various Amazon IP address ranges</td>
<td>Allows the cluster manager to communicate with the master nodes in each Amazon EMR cluster in a single VPC.</td>
</tr>
</tbody>
</table>

**Outbound rules**

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol</th>
<th>Port Range</th>
<th>Source</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>All traffic</td>
<td>All</td>
<td>All</td>
<td>0.0.0.0/0</td>
<td>Provides outbound access to the Internet from any instance in the ElasticMapReduce-master security group or the group that you specify.</td>
</tr>
</tbody>
</table>

The following rules are added to Amazon EMR–managed core/task security groups:

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol</th>
<th>Port Range</th>
<th>Source</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inbound rules</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All ICMP</td>
<td>ICMP</td>
<td>All</td>
<td>The default ElasticMapReduce-master security group ID or the master security</td>
<td>Allows inbound traffic from any instance in the ElasticMapReduce-master security group or the group that you specify. By default, the core/task nodes in all Amazon EMR clusters in a single VPC</td>
</tr>
</tbody>
</table>
### Configure Additional Security Groups

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Whether you use the default managed security groups or your own custom managed security groups, you can assign additional security groups to the master and core/task instances in your cluster. Applying additional security groups gives you the ability to apply additional rules to your security groups so they do not have to be modified. Additional security groups are optional. They can be applied to the master group, core/task group, both groups, or neither group. You can also apply the same additional security groups to multiple clusters.

For example, if you are using your own managed security groups, and you want to allow inbound SSH access to the master group for a particular cluster, you can create an additional security group containing the rule and add it to the master security group for that cluster. Additional security groups are not modified by or maintained by Amazon EMR.

Typically, additional security groups are used to:

- Add access rules to instances in your cluster that are not present in the Amazon EMR-managed security groups
- Give particular clusters access to a specific resource such as a Amazon Redshift database

---

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol</th>
<th>Port Range</th>
<th>Source</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Custom TCP</td>
<td>TCP</td>
<td>8443</td>
<td></td>
<td>Provides outbound access to the Internet from all instances in the ElasticMapReduce-slave security group or the group that you specify.</td>
</tr>
<tr>
<td>All ICMP</td>
<td>ICMP</td>
<td>All</td>
<td>The default ElasticMapReduce-slave security group ID or the core/task security group ID that you specify; for example, sg-8bXXXXee</td>
<td>Allows inbound traffic from any instance in the ElasticMapReduce-slave security group. By default, the core/task nodes accept inbound communication from any other core/task node in any Amazon EMR cluster in a single VPC over any TCP, UDP, or ICMP port. If you choose your own core/task security group, only core/task instances in this group can communicate with each other over any TCP, UDP, or ICMP port.</td>
</tr>
<tr>
<td>All TCP</td>
<td>TCP</td>
<td>All</td>
<td>The default ElasticMapReduce-slave security group ID or the core/task security group ID that you specify; for example, sg-8bXXXXee</td>
<td>Allows inbound traffic from any instance in the ElasticMapReduce-slave security group. By default, the core/task nodes accept inbound communication from any other core/task node in any Amazon EMR cluster in a single VPC over any TCP, UDP, or ICMP port. If you choose your own core/task security group, only core/task instances in this group can communicate with each other over any TCP, UDP, or ICMP port.</td>
</tr>
<tr>
<td>All UDP</td>
<td>UDP</td>
<td>All</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

Outbound rules

| All traffic | All | All | 0.0.0.0/0 | Provides outbound access to the Internet from all instances in the ElasticMapReduce-slave security group or the group that you specify. |
Security groups are restrictive by default. They reject all traffic. You can add a rule to allow traffic on a particular port to your custom or additional security groups. If there is more than one rule for a specific port in two security groups that apply to the same instances, the most permissive rule is applied. For example, if you have a rule that allows SSH access via TCP port 22 from IP address 203.0.113.1 and another rule that allows access to TCP port 22 from any IP address (0.0.0.0/0), the rule allowing access by any IP address takes precedence.

You can apply up to four additional security groups to both the master and core/task security groups. The number of additional groups allowed is dependent upon:

- The number of security groups allowed for your account
- The number of individual rules allowed for your account

For more information about rule limits in VPC security groups, see Security Groups for Your VPC in the Amazon VPC User Guide. You can assign the same additional security groups to both the master and core/task security groups.

Additional security groups can be applied using the console, the API, the CLI, or the SDK.

**To specify additional security groups using the console**

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. Choose **Create cluster**.
3. In the **Security and Access** section, in the **EC2 Security Groups** subsection:
   - For **Master**, choose either the default security group or a custom security group from the list.
   - In the **Additional security groups** column, choose the icon to add up to four additional groups to the master security group.
   - For **Core & Task**, choose either the default security group or a custom security group from the list.
   - In the **Additional security groups** column, choose the icon to add up to four additional groups to the core/task security group.

   **Note**
   You cannot combine custom and default security groups.

4. Proceed with creating the cluster as described in Plan and Configure Clusters (p. 27).

**To specify additional security groups using the AWS CLI**

Use the `create-cluster` command with the `--ec2-attributes` parameter, specifying security group IDs for the `AdditionalSlaveSecurityGroups` and `AdditionalMasterSecurityGroups` variables. Additional security groups are optional. They can be applied to the master group, core/task group, both groups, or neither group.

**Note**
Amazon EMR-managed security groups and additional security groups are not supported in the Amazon EMR CLI.

1. To launch a cluster using additional security groups, type the following command. Replace `myKey` with the name of your Amazon EC2 key pair, and replace `securityGroupId` with the ID of your master security group, core/task security group, and additional security groups.

   ```bash
   aws emr create-cluster --name "Test cluster" --ami-version 3.10 --applications Name=Hue Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey,EmrManagedMasterSecurityGroup=sg-masterId,EmrManagedSlaveSecurityGroup=sg-slaveId,
   ```
Use Security Configurations to Set Up Cluster Security

With Amazon EMR release version 4.8.0 or later, you can use security configurations to configure data encryption, Kerberos authentication (available in release version 5.10.0 and later), and Amazon S3 authorization for EMRFS (available in release version 5.10.0 or later). You can use the console, the AWS Command Line Interface (AWS CLI), or the AWS SDKs to create security configurations. You can also use an AWS CloudFormation template to create a security configuration. For more information, see AWS CloudFormation User Guide and the template reference for AWS::EMR::SecurityConfiguration.

After you create a security configuration, you specify it when you create a cluster, and you can re-use it for any number of clusters.

Topics

- Create a Security Configuration (p. 223)
- Specify a Security Configuration for a Cluster (p. 234)

Create a Security Configuration

To create a security configuration using the console

1. Sign in to the AWS Management Console and open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. 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.
Create a Security Configuration

To create a security configuration using the AWS CLI

1. Use the `create-security-configuration` command with the following syntax:

   ```bash
   aws emr create-security-configuration --name "SecConfigName" --security-configuration SecConfigDef
   ```

   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.

2. For `SecConfigDef`, specify an inline JSON structure or the path to a JSON file in Amazon S3, such as `s3://mybucket/MySecConfig.json` or a local file, such as `file:///MySecConfig.json`. The JSON parameters define options for Encryption and Authentication as described in the sections below.

Configure Data Encryption

Before you configure encryption in a security configuration, create the keys and certificates that are used for encryption. For more information, see Providing Keys for At-Rest Data Encryption with Amazon EMR (p. 211) and Providing Certificates for In-Transit Data Encryption with Amazon EMR Encryption (p. 213).

When you create a security configuration, you specify two sets of encryption options: at-rest data encryption and in-transit data encryption. Options for at-rest data encryption include both Amazon S3 with EMRFS and local-disk encryption. In-transit encryption options enable the open-source encryption features for certain applications that support Transport Layer Security (TLS). At-rest options and in-transit options can be enabled together or separately. For more information, see Encrypt Data in Transit and at Rest (p. 207).

Specifying Encryption Options Using the Console

Choose options under Encryption according to the following guidelines.

- Choose **At rest encryption** to encrypt data stored within the file system. This also enables Hadoop Distributed File System (HDFS) block-transfer encryption and RPC encryption, which need no further configuration.
- Under **S3 data encryption**, for **Encryption mode**, choose a value to determines how Amazon EMR encrypts Amazon S3 data with EMRFS.

What you do next depends on the encryption mode you chose:

- **SSE-S3**
  Specifies **Server-side encryption with Amazon S3-managed encryption keys**. You don't need to do anything more because Amazon S3 handles keys for you.
- **SSE-KMS or CSE-KMS**
  Specifies **server-side encryption with AWS KMS-managed keys (SSE-KMS) or client-side encryption with AWS KMS-managed keys (CSE-KMS)**. For **AWS KMS Key**, select a key. The key must exist in the same region as your Amazon EMR cluster. For key requirements, see Using AWS KMS Customer Master Keys (CMKs) for Encryption (p. 211).
- **CSE-Custom**
  Specifies **client-side encryption using a custom client-side master key (CSE-Custom)**. For **S3 object**, enter the location in Amazon S3, or the Amazon S3 ARN, of your custom key-provider JAR file. Then, for **Key provider class** field, enter the full class name of a class declared in your application that implements the EncryptionMaterialsProvider interface.
Under **Local disk encryption**, choose a value for **Key provider type**. Amazon EMR uses this key for Linux Unified Key System (LUKS) encryption for the local volumes (except boot volumes) attached to your cluster nodes.

- **AWS KMS**

Select this option to specify an AWS KMS customer master key (CMK). For **AWS KMS Key**, select a key. The key must exist in the same region as your Amazon EMR cluster. For more information about key requirements, see Using AWS KMS Customer Master Keys (CMKs) for Encryption (p. 211).

- **Custom**

Select this option to specify a custom key provider. For **S3 object**, enter the location in Amazon S3, or the Amazon S3 ARN, of your custom key-provider JAR file. For **Key provider class**, enter the full class name of a class declared in your application that implements the EncryptionMaterialsProvider interface. The class name you provide here must be different from the class name provided for CSE-Custom.

- Choose **In-transit encryption** to enable the open-source TLS encryption features for in-transit data. Choose a **Certificate provider type** according to the following guidelines:

  - **PEM**

Select this option to use PEM files that you provide within a zip file. Two artifacts are required within the zip file: privateKey.pem and certificateChain.pem. A third file, trustedCertificates.pem, is optional. See Providing Certificates for In-Transit Data Encryption with Amazon EMR Encryption (p. 213) 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 of JSON parameters and appropriate values.

#### Example In-Transit Data Encryption Options

The example below illustrates the following scenario:

- In-transit data encryption is enabled and at-rest data encryption is disabled.
- A zip file with certificates in Amazon S3 is used as the key provider (see Providing Certificates for In-Transit Data Encryption with Amazon EMR Encryption (p. 213) for certificate requirements.

```bash
aws emr create-security-configuration --name "MySecConfig" --security-configuration '{
    "EncryptionConfiguration": {
        "EnableInTransitEncryption" : true,
        "DisableAtRestEncryption" : false,
        "InTransitEncryptionConfiguration" : {
            "TLSCertificateConfiguration" : {
                "CertificateProviderType" : "PEM",
                "S3Object" : "s3://MyConfigStore/artifacts/MyCerts.zip"
            }
        }
    }
}'
```
Create a Security Configuration

The example below illustrates the following scenario:

- In-transit data encryption is enabled and at-rest data encryption is disabled.
- A custom key provider is used (see Providing Certificates for In-Transit Data Encryption with Amazon EMR Encryption (p. 213) for certificate requirements).

AWS CLI Example

```bash
aws emr create-security-configuration --name "MySecConfig" --security-configuration '{
  "EncryptionConfiguration": {
    "EnableInTransitEncryption": true,
    "EnableAtRestEncryption": false,
    "InTransitEncryptionConfiguration": {
      "TLSConfiguration": {
        "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 CLI Example

```bash
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" : "Custom",
    "S3Object" : "arn:aws:s3:::artifacts/MyKeyProvider.jar",
    "EncryptionKeyProviderClass" : "com.mycompany.MyKeyProvider.jar"
}
}'}

The example below illustrates the following scenario:

- In-transit data encryption is enabled 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" : "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" : "Custom",
    "S3Object" : "arn:aws:s3:::artifacts/MyKeyProvider.jar",
    "EncryptionKeyProviderClass" : "com.mycompany.MyKeyProvider.jar"
}
}'}

The example below illustrates the following scenario:

- In-transit data encryption is enabled with a custom key provider
• CSE-Custom is used for Amazon S3 data
• Local disk encryption uses a custom key provider

```bash
aws emr create-security-configuration --name "MySecConfig" --security-configuration '{
  "EncryptionConfiguration": {
    "EnableInTransitEncryption" : "true",
    "EnableAtRestEncryption" : "true",
    "InTransitEncryptionConfiguration" : {
      "TLSCertificateConfiguration" : {
        "CertificateProviderType" : "Custom",
        "S3Object" : "s3://MyConfig/artifacts/MyCerts.jar",
        "CertificateProviderClass" : "com.mycompany.MyCertProvider"
      }
    },
    "AtRestEncryptionConfiguration" : {
      "S3EncryptionConfiguration" : {
        "EncryptionMode" : "CSE-Custom",
        "S3Object" : "s3://MyConfig/artifacts/MyCerts.jar",
        "EncryptionKeyProviderClass" : "com.mycompany.MyKeyProvider"
      },
      "LocalDiskEncryptionConfiguration" : {
        "EncryptionKeyProviderType" : "Custom",
        "S3Object" : "s3://MyConfig/artifacts/MyCerts.jar",
        "EncryptionKeyProviderClass" : "com.mycompany.MyKeyProvider"
      }
    }
  }
}
```

### JSON Reference for Encryption Settings

The following table lists the JSON parameters for encryption settings and provides a description of acceptable values for each parameter.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;EnableInTransitEncryption&quot; : true</td>
<td>Specify true to enable in-transit encryption and false to disable it. If omitted, false is assumed, and in-transit encryption is disabled.</td>
</tr>
<tr>
<td>&quot;EnableAtRestEncryption&quot; : true</td>
<td>Specify true to enable at-rest encryption and false to disable it. If omitted, false is assumed and at-rest encryption is disabled.</td>
</tr>
</tbody>
</table>

#### In-transit encryption parameters

<p>| &quot;InTransitEncryptionConfiguration&quot; : | Specifies a collection of values used to configure in-transit encryption when EnableInTransitEncryption is true. |
| &quot;CertificateProviderType&quot; : &quot;PEM&quot; | Specifies whether to use PEM certificates referenced with a zip 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. |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;S3Object&quot; : &quot;ZipLocation&quot;</td>
<td>Provides the location in Amazon S3 to a zip file when PEM is specified, or to a JAR file when Custom is specified. The format can be a path (for example, s3://MyConfig/artifacts/CertFiles.zip) or an ARN (for example, arn:aws:s3:::Code/MyCertProvider.jar). If a zip file is specified, it must contain files named exactly privateKey.pem and certificateChain.pem. A file named trustedCertificates.pem is optional.</td>
</tr>
<tr>
<td>&quot;CertificateProviderClass&quot; : &quot;MyClassID&quot;</td>
<td>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.</td>
</tr>
</tbody>
</table>

**At-rest encryption parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;AtRestEncryptionConfiguration&quot; :</td>
<td>Specifies a collection of values for at-rest encryption when EnableAtRestEncryption is true, including Amazon S3 encryption and local disk encryption.</td>
</tr>
</tbody>
</table>

**Amazon S3 encryption parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;S3EncryptionConfiguration&quot; :</td>
<td>Specifies a collection of values used for Amazon S3 encryption with the EMR File System (EMRFS).</td>
</tr>
<tr>
<td>&quot;EncryptionMode&quot; : &quot;SSE-S3&quot;</td>
<td>Specifies the type of Amazon S3 encryption to use. If SSE-S3 is specified, no further Amazon S3 encryption values are required. If either SSE-KMS or CSE-KMS is specified, an AWS KMS customer master key (CMK) ARN must be specified as the AwsKmsKey value. If CSE-Custom is specified, S3Object and EncryptionKeyProviderClass values must be specified.</td>
</tr>
<tr>
<td>&quot;AwsKmsKey&quot; : &quot;MyKeyARN&quot;</td>
<td>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).</td>
</tr>
<tr>
<td>&quot;S3Object&quot; : &quot;JarLocation&quot;</td>
<td>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).</td>
</tr>
</tbody>
</table>
Create a Security Configuration

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;EncryptionKeyProviderClass&quot; : &quot;MyS3KeyClassID&quot;</td>
<td>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.</td>
</tr>
</tbody>
</table>

Local disk encryption parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;LocalDiskEncryptionKeyProvider&quot;</td>
<td>Specifies the key provider and corresponding values to be used for local disk encryption.</td>
</tr>
<tr>
<td>&quot;Type&quot; : &quot;AwsKms&quot;</td>
<td>&quot;Custom&quot;</td>
</tr>
<tr>
<td>&quot;AwsKmsKey&quot; : &quot;MyKeyARN&quot;</td>
<td>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).</td>
</tr>
<tr>
<td>&quot;S3Object&quot; : &quot;JarLocation&quot;</td>
<td>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).</td>
</tr>
<tr>
<td>&quot;EncryptionKeyProviderClass&quot; : &quot;MyLocalDiskKeyClassID&quot;</td>
<td>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.</td>
</tr>
</tbody>
</table>

Configure Kerberos Authentication

A security configuration with Kerberos settings can only be used by a cluster that is created with Kerberos attributes or an error occurs. For more information, see Configure Kerberos (p. 195). 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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable Kerberos</td>
<td>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.</td>
</tr>
</tbody>
</table>
### Parameter | Description
--- | ---
**Ticket Lifetime** | The period for which a Kerberos ticket issued by the cluster-dedicated KDC is valid. Ticket lifetimes are limited for security reasons. Cluster applications and services auto-renew tickets after they expire. Users who connect to the cluster over SSH using Kerberos credentials need to run `kinit` from the master node command line to renew after a ticket expires.

**Cross-realm trust** | If you provide a cross-realm trust configuration, principals (typically users) from another realm are authenticated to clusters that use this configuration. Additional configuration in the other Kerberos realm is also required. For more information, see [Configure a Cross-Realm Trust](p. 202).

**Admin server** | The fully qualified domain name (FQDN) of the other Kerberos admin server in the trust relationship. The admin server and KDC typically run on the same server. Optionally, you can specify the port used to communicate with Kerberos admin server. If not specified, port 749 is used, which is the Kerberos default.

**KDC server** | The fully qualified domain name (FQDN) of the KDC in the other realm of the trust relationship. Optionally, you can specify the port used to communicate with the KDC server. If not specified, port 88 is used, which is the Kerberos default.

**Domain name** | The domain name of the other realm in the trust relationship.

---

### JSON Reference for Kerberos Settings

The following example shows JSON parameters for a Kerberos configuration, followed by a reference of parameters and values.

```json
{
  "AuthenticationConfiguration": {
    "KerberosConfiguration": {
      "Provider": "ClusterDedicatedKdc",
      "TicketLifeTimeInHours": number,
      "ClusterDedicatedKdcConfiguration": {
        "CrossRealmTrustConfiguration": {
          "TrustingAdminServer": "domain.example.com",
          "TrustingDomainName": "domain.example.com",
          "TrustingKdcServer": "domain.example.com"
        }
      }
    }
  }
}
```
Parameter | Description
----------|--------------------------------------------------
"AuthenticationConfiguration" : | Required. Contains Kerberos configuration parameters.
"KerberosConfiguration" : | Required. Contains Kerberos configuration parameters.
"Provider": "ClusterDedicatedKdc" | Required. Specifies that a cluster-dedicated KDC is created on the master node.
"TicketLifeTimeInHours": number | Optional. Specifies the period for which a Kerberos ticket issued by the cluster-dedicated KDC is valid. Ticket lifetimes are limited for security reasons. Cluster applications and services auto-renew tickets after they expire. Users who connect to the cluster over SSH using Kerberos credentials need to run `kinit` from the master node command line to renew after a ticket expires. If omitted, defaults to 24.
"CrossRealmTrustConfiguration": | Optional. Contains parameters that define a cross-realm trust configuration.
"TrustingAdminServer": "domain.example.com" | Specifies the fully qualified domain name (FQDN) of the other Kerberos admin server in the trust relationship. The admin server and KDC typically run on the same server. Optionally, you can specify the port used to communicate with Kerberos admin server (for example, domain.example.com:749). If not specified, port 749 is used, which is the Kerberos default.
"TrustingDomainName": "domain.example.com" | Specifies the domain name of the other realm in the trust relationship.
"TrustingKdcServer": "domain.example.com" | Specifies the fully qualified domain name (FQDN) of the KDC in the other realm of the trust relationship. Optionally, you can specify the port used to communicate with the KDC server (for example, domain.example.com:88). If not specified, port 88 is used, which is the Kerberos default.

**Configure S3 Authorization for EMRFS**

EMRFS S3 authorization provides a way for you to provide dynamic credentials for users to access EMRFS data in Amazon S3. You create role mappings that specify an IAM role that is assumed when the 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 EMRFS Authorization for Data in Amazon S3 (p. 214).

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

```json
{
    "AuthorizationConfiguration": {
        "EmrFsConfiguration": {
```
"RoleMappings": [{
    "Role": "arn:aws:iam::123456789101:role/allow EMRFS_access_for_user1",
    "IdentifierType": "User",
    "Identifiers": [ "user1" ]
},
    "Role": "arn:aws:iam::123456789101:role/allow EMRFS_access_to_MyBuckets",
    "IdentifierType": "Prefix",
    "Identifiers": [ "s3://MyBucket/","s3://MyOtherBucket/" ]
},
    "Role": "arn:aws:iam::123456789101:role/allow EMRFS_access_for_AdminGroup",
    "IdentifierType": "Group",
    "Identifiers": [ "AdminGroup" ]
}]

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;EmrFsConfiguration&quot;</td>
<td>Required for EMRFS authorization. Contains EMRFS authorization configurations.</td>
</tr>
<tr>
<td>&quot;RoleMappings&quot;</td>
<td>Required for EMRFS authorization. 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. Role mappings consist of the following required parameters:</td>
</tr>
<tr>
<td>&quot;Role&quot;</td>
<td>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.</td>
</tr>
<tr>
<td>&quot;IdentifierType&quot;</td>
<td>Can be one of the following:</td>
</tr>
<tr>
<td></td>
<td>- &quot;User&quot; 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.</td>
</tr>
<tr>
<td></td>
<td>- &quot;Prefix&quot; specifies that the identifier is an Amazon S3 location. The IAM role is assumed for calls to the location or locations with the specified prefixes. For example, the prefix s3://mybucket/ matches s3://mybucket/mydir and s3://mybucket/yetanotherdir.</td>
</tr>
<tr>
<td></td>
<td>- &quot;Group&quot; 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.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>&quot;Identifiers&quot;:</td>
<td>Specifies one or more identifiers of the appropriate identifier type. Separate multiple identifiers by commas with no spaces.</td>
</tr>
</tbody>
</table>

Specify a Security Configuration for a Cluster

You can specify encryption settings when you create a cluster by specifying the security configuration. You can use the AWS Management Console or the AWS CLI.

Specifying a Security Configuration Using the Console

When using the AWS console to create an Amazon EMR cluster, you choose the security configuration during Step 4: Security of the advanced options creation process.

1. Sign in to the AWS Management Console and open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. Choose Create cluster, Go to advanced options.
3. On the Step 1: Software and Steps screen, from the Release list, choose emr-4.8.0 or a more recent release. Choose the settings you want and choose Next.
4. On the Step 2: Hardware screen, choose the settings you want and choose Next. Do the same for Step 3: General Cluster Settings.
6. Configure other security options as desired and choose Create cluster.

Specifying a Security Configuration Using the CLI

When you use `aws emr create-cluster`, you can optionally apply a security configuration using `--security-configuration MySecConfig`, where `MySecConfig` is the name of the security configuration, as shown in the following example. The `--release-label` specified must be 4.8.0 or later and the `--instance-type` can be any available.

```
aws emr create-cluster --instance-type m3.xlarge --release-label emr-5.0.0 --security-configuration mySecConfig
```

Configure IAM Roles for Amazon EMR Permissions to AWS Services

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Amazon EMR and applications such as Hadoop and Spark need permissions to access other AWS resources and perform actions when they run. Each cluster in Amazon EMR must have a service role and a role for the Amazon EC2 instance profile. The IAM policies attached to these roles provide permissions for the cluster to interoperate with other AWS services. All clusters in all regions require a service role and a
role for the EC2 instance profile. An additional role, the **autoscaling role**, is required if your cluster uses automatic scaling. For more information, see IAM Roles and Using Instance Profiles in the IAM User Guide.

If you are creating a cluster for the first time in an account, roles for Amazon EMR do not yet exist. You can specify default roles for Amazon EMR to create, or you can create custom roles and specify them when you create a cluster. For more information, see Use Default IAM Roles and Managed Policies (p. 235), and Customize IAM Roles (p. 241).

**Summary of Amazon EMR Roles and Their Purposes**

- **The service role** defines the allowable actions for Amazon EMR when provisioning resources and performing other service-level tasks that are not performed in the context of a particular EC2 instance.

- **The role for the Amazon EC2 instance profile** is assumed by EC2 instances within the cluster. The permissions associated with this role apply to processes that run on cluster instances. As long as an application process runs on top of the Hadoop ecosystem, the application assumes the role. If you have application code that calls AWS services directly, you may need to use the SDK to specify roles. For more information, see Use IAM Roles with Applications That Call AWS Services Directly (p. 243).

- Clusters typically read and write data to Amazon S3 using EMRFS. When you have multiple cluster users and multiple data stores, you may want users to have different permissions to EMRFS data in Amazon S3. Use EMRFS authorization to do this, which allows EMRFS to assume a different, customized role based on the user or group making the request or the location of EMRFS data in Amazon S3. For more information, see EMRFS Authorization for Data in Amazon S3 (p. 214).

- **The autoscaling role** performs a similar function as the service role, but allows additional actions for dynamically scaling environments.

- **A service linked role** is created automatically by Amazon EMR and allows Amazon EMR to clean up Amazon EC2 resources if the service for Amazon EMR has lost that ability. For more information, see Using the Service-Linked Role for Amazon EMR (p. 244). Amazon EMR must also be allowed to create a service-linked role for Spot instances.

**Topics**

- Use Default IAM Roles and Managed Policies (p. 235)
- Allow Users and Groups to Create and Modify Roles (p. 241)
- Customize IAM Roles (p. 241)
- Specify Custom IAM Roles When You Create a Cluster (p. 242)
- Use IAM Roles with Applications That Call AWS Services Directly (p. 243)
- Using the Service-Linked Role for Amazon EMR (p. 244)

### Use Default IAM Roles and Managed Policies

Amazon EMR provides the default roles listed below. The managed policies listed are created and attached to them by default. These roles and policies do not exist in your account until you create them. After you create them, you can view the roles, the policies attached to them, and the permissions allowed or denied by the policies in the IAM console (https://console.aws.amazon.com/iam/). Managed policies are created and maintained by AWS, so they are updated automatically if service requirements change.

<table>
<thead>
<tr>
<th>Default Role</th>
<th>Description</th>
<th>Default Managed Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMR_DefaultRole</td>
<td>The service role for Amazon EMR, which allows Amazon EMR to call other AWS services such as Amazon EC2 on your behalf.</td>
<td>AmazonElasticMapReduceRole.</td>
</tr>
</tbody>
</table>
### Default Role

<table>
<thead>
<tr>
<th>Default Role</th>
<th>Description</th>
<th>Default Managed Policy</th>
</tr>
</thead>
</table>
| Default Role                  | Description                                                                                       | Important
|                               | Requesting spot instances requires a service-linked role. If this role doesn't exist, the Amazon EMR service role must have permission to create it or a permission error occurs. The managed policy includes a statement to allow this action. If you customize this role or policy, be sure to include a statement that allows creation of this service linked role. For more information, see Default Contents of AmazonElasticMapReduceRole (p. 238) and Service-Linked Role for Spot Instance Requests in the Amazon EC2 User Guide for Linux Instances. |
| EMR_EC2_DefaultRole           | The service role for EC2 instances within a cluster. This role is assumed by processes that run on cluster instances when they call other AWS services. For accessing EMRFS data in Amazon S3, you can specify different roles to be assumed based on the user or group making the request, or on the location of data in Amazon S3. For more information, see EMRFS Authorization for Data in Amazon S3 (p. 214). | AmazonElasticMapReduceforEC2Role. For more information, see Default Contents of AmazonElasticMapReduceforEC2Role (p. 240). |
| EMR_AutoScalingDefaultRole    | Allows additional actions for dynamically scaling environments. Required only for clusters that use autoscaling. For more information, see the section called “Using Automatic Scaling in Amazon EMR” (p. 478). | AmazonElasticMapReduceforAutoScalingRole. For more information, see Default Contents of AmazonMapReduceforAutoScalingRole (p. 241). |
To create default IAM roles for Amazon EMR in your account for the first time

1. If you are logged in as an IAM user, make sure that you are allowed the appropriate IAM actions to create roles. For more information, see Allow Users and Groups to Create and Modify Roles (p. 241).

2. Use the Amazon EMR console to create a cluster and leave the default roles specified. You can do this either using Quick options or using Advanced options. Amazon EMR automatically creates the roles when it launches the cluster. They are available to any clusters you launch later.

—OR—

Use the emr create-default-roles command from the AWS CLI.

The output of the command lists the contents of the roles. The CLI configuration file (config) is populated with these role names for the service_role and instance_profile values. For example:

```
[default]
output = json
region = us-west-1
aws_access_key_id = myAccessKeyId
aws_secret_access_key = mySecretAccessKeyId
emr =
  service_role = EMR_DefaultRole
  instance_profile = EMR_EC2_DefaultRole
```

Managed Policy Contents

The contents of managed policies for Amazon EMR default roles are shown below. Managed policies are automatically updated, so the policies shown here may be out-of-date. Use the procedure below to view the most recent policy. You can also check the version listed below against the version listed for each managed policy.

To view current default roles and managed policies for Amazon EMR

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

2. Choose Create cluster, Go to advanced options.

3. Choose Next until you reach Security options.

4. Choose the service role you want to view.
5. Choose the permissions policy to view the policy JSON.

6. Choose Policy versions to see the current policy version number, a history of policy updates, and the policy JSON for past versions.

Default Contents of AmazonElasticMapReduceRole

The contents of version 9 of this policy are shown below.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Resource": "*",
      "Action": [
        "ec2:AuthorizeSecurityGroupEgress",
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:CancelSpotInstanceRequests",
        "ec2:CreateNetworkInterface",
        "ec2:CreateSecurityGroup",
      ]
    }
  ]
}
```
"ec2:CreateTags",
"ec2:DeleteNetworkInterface",
"ec2:DeleteSecurityGroup",
"ec2:DeleteTags",
"ec2:DescribeAvailabilityZones",
"ec2:DescribeAccountAttributes",
"ec2:DescribeDhcpOptions",
"ec2:DescribeImages",
"ec2:DescribeInstanceStatus",
"ec2:DescribeInstances",
"ec2:DescribeKeyPairs",
"ec2:DescribeNetworkAcls",
"ec2:DescribeNetworkInterfaces",
"ec2:DescribePrefixLists",
"ec2:DescribeRouteTables",
"ec2:DescribeSecurityGroups",
"ec2:DescribeSpotInstanceRequests",
"ec2:DescribeSpotPriceHistory",
"ec2:DescribeSubnets",
"ec2:DescribeTags",
"ec2:DescribeVpcAttribute",
"ec2:DescribeVpcEndpoints",
"ec2:DescribeVpcEndpointServices",
"ec2:DescribeVpcs",
"ec2:DetachNetworkInterface",
"ec2:ModifyImageAttribute",
"ec2:ModifyInstanceAttribute",
"ec2:RequestSpotInstances",
"ec2:RevokeSecurityGroupEgress",
"ec2:RunInstances",
"ec2:TerminateInstances",
"ec2:DeleteVolume",
"ec2:DescribeVolumeStatus",
"ec2:DescribeVolumes",
"ec2:DetachVolume",
"iam:GetRole",
"iam:GetRolePolicy",
"iam:ListInstanceProfiles",
"iam:ListRolePolicies",
"iam:PassRole",
"s3:CreateBucket",
"s3:Get*",
"s3:List*",
"sdb:BatchPutAttributes",
"sdb:Select",
"sqs:CreateQueue",
"sqs:Delete*",
"sqs:GetQueue*",
"sqs:PurgeQueue",
"sqs:ReceiveMessage",
"cloudwatch:PutMetricAlarm",
"cloudwatch:DescribeAlarms",
"cloudwatch:DeleteAlarms",
"application-autoscaling:RegisterScalableTarget",
"application-autoscaling:DeregisterScalableTarget",
"application-autoscaling:PutScalingPolicy",
"application-autoscaling:DeleteScalingPolicy",
"application-autoscaling:Describe**
]
},
{
"Effect": "Allow",
"Action": "iam:CreateServiceLinkedRole",
"Resource": "arn:aws:iam::*:role/aws-service-role/spot.amazonaws.com/AWSServiceRoleForEC2Spot**",
"Condition": {
}}
Use Default IAM Roles and Managed Policies

The contents of version 3 of this policy are shown below.

```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",
```
Default Contents of AmazonMapReduceforAutoScalingRole

The contents of version 1 of this policy are shown below.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "cloudwatch:DescribeAlarms",
                "elasticmapreduce:ListInstanceGroups",
                "elasticmapreduce:ModifyInstanceGroups"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

Allow Users and Groups to Create and Modify Roles

IAM principals (users and groups) who create, modify, and specify roles for a cluster, including default roles, must be allowed to perform the following actions. For details about each action, see Actions in the IAM API Reference.

- `iam:CreateRole`
- `iam:PutRolePolicy`
- `iam:CreateInstanceProfile`
- `iam:AddRoleToInstanceProfile`
- `iam:ListRoles`
- `iam:GetPolicy`
- `iam:GetInstanceProfile`
- `iam:GetPolicyVersion`
- `iam:AttachRolePolicy`
- `iam:PassRole`

The `iam:PassRole` permission allows cluster creation. The remaining permissions allow creation of the default roles.

Customize IAM Roles

You may want to customize IAM roles and permissions for your requirements. For example, if your application does not use EMRFS consistent view, you may not want to allow Amazon EMR to access Amazon DynamoDB. To customize permissions, we recommend that you create new roles and policies. Begin with the permissions in the managed policies for the default roles (for example, `AmazonElasticMapReduceforEC2Role` and `AmazonElasticMapReduceRole`), copy and paste the contents to new policy statements, modify the permissions as appropriate, and attach the modified permissions policies to the roles that you create. You must have the appropriate IAM permissions to
work with roles and policies. For more information, see Allow Users and Groups to Create and Modify Roles (p. 241).

**Important**
Inline policies are not automatically updated when service requirements change. If you create inline policies and attach them to IAM roles for Amazon EMR, be aware that service updates could occur that suddenly cause permission errors. For more information, see Managed Policies and Inline Policies in the IAM User Guide and Specify Custom IAM Roles When You Create a Cluster (p. 242).

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 creating clusters needs permissions to retrieve and assign roles to Amazon EMR and EC2 instances. Otherwise, a **User account is not authorized to call EC2** error occurs. For more information, see Allow Users and Groups to Create and Modify Roles (p. 241).

**Use the Console to Specify Custom Roles**

You can specify a custom service role for Amazon EMR, a custom role for the EC2 instance profile, and a custom autoscaling role using Advanced options when you create a cluster. When you use Quick options, the default service role and the default role for the EC2 instance profile are specified. For more information, see Use Default IAM Roles and Managed Policies (p. 235).

**To specify custom IAM roles using the console**

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. Choose Create cluster, Go to advanced options.
3. Choose the cluster settings appropriate for your application until you reach Security Options.

   Under Permissions, the Default roles for Amazon EMR are selected.

4. Choose Custom.
5. For each role type, select a role from the list. Only roles within your account that have the appropriate trust policy for that role type are listed.
6. Choose other options as appropriate for your cluster and then choose **Create Cluster**.

### Use the AWS CLI to Specify Custom Roles

You can specify a service role and a role for the EC2 instance profile using options with the `create-cluster` command from the AWS CLI. Use the `--service-role` option to specify the service role. Use the `InstanceProfile` argument of the `--ec2-attributes` option to specify the role for the EC2 instance profile.

You can use these options to specify default roles explicitly rather than using the `--use-default-roles` option. The `--use-default-roles` option will specify the service role and the role for the EC2 instance profile defined in the CLI configuration file, which you can update to reflect your custom roles. For more information, see [Use Default IAM Roles and Managed Policies](#) (p. 235).

The autoscaling role is specified using a separate option, `--auto-scaling-role`. For more information, see [Using Automatic Scaling in Amazon EMR](#) (p. 478).

### To specify custom IAM roles using the AWS CLI

- The following command specifies the custom service role, `MyEMRServiceRole`, and a custom role for the EC2 instance profile, `MyEC2RoleForEMR`, when launching a cluster. This example uses the default Amazon EMR role.

  ```
  aws emr create-cluster --name "Test cluster" --release-label\n  --applications Name=Hive Name=Pig --service-role MyEMRServiceRole\n  --ec2-attributes InstanceProfile=MyEC2RoleForEMR,\n  KeyName=myKey --instance-type m3.xlarge --instance-count 3
  ```

### Use IAM Roles with Applications That Call AWS Services Directly

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Applications running on the EC2 instances of a cluster can use the instance profile to obtain temporary security credentials when calling AWS services.

The version of Hadoop available on AMI 2.3.0 and later has already been updated to make use of IAM roles. If your application runs strictly on top of the Hadoop architecture, and does not directly call any service in AWS, it should work with IAM roles with no modification.

If your application calls services in AWS directly, you need to update it to take advantage of IAM roles. This means that instead of obtaining account credentials from `/home/hadoop/conf/core-site.xml` on the EC2 instances in the cluster, your application now uses an SDK to access the resources using IAM roles, or calls the EC2 instance metadata to obtain the temporary credentials.

### To access AWS resources with IAM roles using an SDK

- The following topics show how to use several of the AWS SDKs to access temporary credentials using IAM roles. Each topic starts with a version of an application that does not use IAM roles and then walks you through the process of converting that application to use IAM roles.
To obtain temporary credentials from EC2 instance metadata

- Call the following URL from an EC2 instance that is running with the specified IAM role, which returns the associated temporary security credentials (AccessKeyId, SecretAccessKey, SessionToken, and Expiration). The example that follows uses the default instance profile for Amazon EMR, EMR_EC2_DefaultRole.

```
```

For more information about writing applications that use IAM roles, see Granting Applications that Run on Amazon EC2 Instances Access to AWS Resources.

For more information about temporary security credentials, see Using Temporary Security Credentials in the Using Temporary Security Credentials guide.

# Using the Service-Linked Role for Amazon EMR

Amazon EMR uses AWS Identity and Access Management (IAM) service-linked roles. A service-linked role is a unique type of IAM role that is linked directly to Amazon EMR. The service-linked role is predefined by Amazon EMR and includes the permissions that Amazon EMR requires to call Amazon EC2 on your behalf to clean up cluster resources after they are no longer in use. The service-linked role works together with the Amazon EMR service role and Amazon EC2 instance profile for Amazon EMR. For more information about the service role and instance profile, see Configure IAM Roles for Amazon EMR Permissions to AWS Services (p. 234).

Amazon EMR defines the permissions of this service-linked role, and unless defined otherwise, only Amazon EMR can assume the role. The defined permissions include the trust policy and the permissions policy, and that permissions policy cannot be attached to any other IAM entity. You can delete the role only after you terminate all EMR clusters in the account.

For information about other services that support service-linked roles, see AWS Services That Work with IAM and look for the services that have Yes in the Service-Linked Role column. Choose a Yes with a link to view the service-linked role documentation for that service.

## Service-Linked Role Permissions for Amazon EMR

Amazon EMR uses the AWSServiceRoleForEMRCleanup role, which is a service-based role that allows Amazon EMR to terminate and delete Amazon EC2 resources on your behalf if the Amazon EMR service role has lost that ability. Amazon EMR creates the role automatically during cluster creation if it does not already exist.

The AWSServiceRoleForEMRCleanup service-linked role trusts the following services to assume the role:
The AWSServiceRoleForEMRCleanup service-linked role permissions policy allows Amazon EMR to complete the following actions on the specified resources:

- Action: DescribeInstances on ec2
- Action: DescribeSpotInstanceRequests on ec2
- Action: ModifyInstanceAttribute on ec2
- Action: TerminateInstances on ec2
- Action: CancelSpotInstanceRequests on ec2
- Action: DeleteNetworkInterface on ec2
- Action: DescribeInstanceAttribute on ec2
- Action: DescribeVolumeStatus on ec2
- Action: DescribeVolumes on ec2
- Action: DetachVolume on ec2
- Action: DeleteVolume on ec2

You must configure permissions to allow an IAM entity (such as a user, group, or role) to create, edit, or delete a service-linked role.

To allow an IAM entity to create the AWSServiceRoleForEMRCleanup service-linked role

Add the following statement to the permissions policy for the IAM entity that needs to create the service-linked role:

```json
{
  "Effect": "Allow",
  "Action": [
    "iam:CreateServiceLinkedRole",
    "iam:PutRolePolicy"
  ],
  "Resource": "arn:aws:iam::*:role/aws-service-role/elasticmapreduce.amazonaws.com/AWSServiceRoleForEMRCleanup*",
  "Condition": {"StringLike": {"iam:AWSServiceName": "elasticmapreduce.amazonaws.com"}}
}
```

To allow an IAM entity to edit the description of the AWSServiceRoleForEMRCleanup service-linked role

Add the following statement to the permissions policy for the IAM entity that needs to edit the description of a service-linked role:

```json
{
  "Effect": "Allow",
  "Action": [
    "iam:UpdateRoleDescription"
  ],
  "Resource": "arn:aws:iam::*:role/aws-service-role/elasticmapreduce.amazonaws.com/AWSServiceRoleForEMRCleanup*",
  "Condition": {"StringLike": {"iam:AWSServiceName": "elasticmapreduce.amazonaws.com"}}
}
```

To allow an IAM entity to delete the AWSServiceRoleForEMRCleanup service-linked role

Add the following statement to the permissions policy for the IAM entity that needs to delete a service-linked role:

```json
{
  "Effect": "Allow",
  "Action": [
    "iam:CreateServiceLinkedRole",
    "iam:PutRolePolicy"
  ],
  "Resource": "arn:aws:iam::*:role/aws-service-role/elasticmapreduce.amazonaws.com/AWSServiceRoleForEMRCleanup*",
  "Condition": {"StringLike": {"iam:AWSServiceName": "elasticmapreduce.amazonaws.com"}}
}
```
Creating a Service-Linked Role for Amazon EMR

You don't need to manually create the AWSServiceRoleForEMRCleanup role. When you launch a cluster, either for the first time or when a service-linked role is not present, Amazon EMR creates the service-linked role for you. You must have permissions to create the service linked role. For an example statement that adds this capability to the permissions policy of an IAM entity (such as a user, group, or role), see Service-Linked Role Permissions for Amazon EMR (p. 244).

**Important**
If you were using Amazon EMR before October 24, 2017, when service-linked roles were not supported, then Amazon EMR created the AWSServiceRoleForEMRCleanup role in your account. For more information, see A New Role Appeared in My IAM Account.

Editing a Service-Linked Role for Amazon EMR

Amazon EMR does not allow you to edit the AWSServiceRoleForEMRCleanup service-linked role. After you create a service-linked role, you cannot change the name of the role because various entities might reference the role. However, you can edit the description of the role using IAM.

**Editing a Service-Linked Role Description (IAM Console)**

You can use the IAM console to edit the description of a service-linked role.

**To edit the description of a service-linked role (console)**

1. In the navigation pane of the IAM console, choose **Roles**.
2. Choose the name of the role to modify.
3. To the far right of **Role description**, choose **Edit**.
4. Type a new description in the box and choose **Save**.

**Editing a Service-Linked Role Description (IAM CLI)**

You can use IAM commands from the AWS Command Line Interface to edit the description of a service-linked role.

**To change the description of a service-linked role (CLI)**

1. (Optional) To view the current description for a role, use the following commands:

   ```bash
   $ 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`.  

---

246
To update a service-linked role's description, use one of the following commands:

```
$ aws iam update-role-description --role-name role-name --description description
```

**Editing a Service-Linked Role Description (IAM API)**

You can use the IAM API to edit the description of a service-linked role.

**To change the description of a service-linked role (API)**

1. (Optional) To view the current description for a role, use the following command:
   
   IAM API: GetRole

2. To update a role's description, use the following command:
   
   IAM API: UpdateRoleDescription

**Deleting a Service-Linked Role for Amazon EMR**

If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. That way you don't have an unused entity that is not actively monitored or maintained. However, you must clean up your service-linked role before you can delete it.

**Cleaning Up a Service-Linked Role**

Before you can use IAM to delete a service-linked role, you must first confirm that the role has no active sessions and remove any resources used by the role.

**To check whether the service-linked role has an active session in the IAM console**

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose Roles. Select the name (not the check box) of the AWSServiceRoleForEMRCleanup role.
3. On the Summary page for the selected role, choose Access Advisor.
4. On the Access Advisor tab, review recent activity for the service-linked role.

**Note**

If you are unsure whether Amazon EMR is using the AWSServiceRoleForEMRCleanup role, you can try to delete the role. If the service is using the role, then the deletion fails and you can view the regions where the role is being used. If the role is being used, then you must wait for the session to end before you can delete the role. You cannot revoke the session for a service-linked role.

**To remove Amazon EMR resources used by the AWSServiceRoleForEMRCleanup**

- Terminate all clusters in your account. For more information, see Terminate a Cluster (p. 472).

**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/.

---

247
2. In the navigation pane, choose Roles. Select the check box next to AWSServiceRoleForEMRCleanup, not the name or row itself.
3. For Role actions at the top of the page, choose Delete role.
4. In the confirmation dialog box, review the service last accessed data, which shows when each of the selected roles last accessed an AWS service. This helps you to confirm whether the role is currently active. To proceed, choose Yes, Delete.
5. Watch the IAM console notifications to monitor the progress of the service-linked role deletion. Because the IAM service-linked role deletion is asynchronous, after you submit the role for deletion, the deletion task can succeed or fail. If the task fails, you can choose View details or View Resources from the notifications to learn why the deletion failed. If the deletion fails because there are resources in the service that are being used by the role, then the reason for the failure includes a list of resources.

Deleting a Service-Linked Role (IAM CLI)

You can use IAM commands from the AWS Command Line Interface to delete a service-linked role.

To delete a service-linked role (CLI)

1. 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. You must capture the deletion-task-id from the response to check the status of the deletion task. Type the following command to submit a service-linked role deletion request:

   ```bash
   $ aws iam delete-service-linked-role --role-name AWSServiceRoleForEMRCleanup
   ```

2. Type the following command to check the status of the deletion task:

   ```bash
   $ 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.

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.

   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. You must capture the DeletionTaskId from the response to check the status of the deletion task.

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.
Run a Hadoop Application to Process Data

Amazon EMR provides several models for creating custom Hadoop applications to process data:

- **Custom JAR or Cascading** — write a Java application, which may or may not make use of the Cascading Java libraries, generate a JAR file, and upload the JAR file to Amazon S3 where it will be imported into the cluster and used to process data. When you do this, your JAR file must contain an implementation for both the Map and Reduce functionality.

- **Streaming** — write separate Map and Reduce scripts using one of several scripting languages, upload the scripts to Amazon S3, where the scripts are imported into the cluster and used to process data. You can also use built-in Hadoop classes, such as `aggregate`, instead of providing a script.

Regardless of which type of custom application you create, the application must provide both Map and Reduce functionality, and should adhere to Hadoop programming best practices.

Topics

- Build Binaries Using Amazon EMR (p. 249)
- JAR Requirements (p. 251)
- Run a Script in a Cluster (p. 251)
- Process Data with Streaming (p. 252)
- Process Data Using Cascading (p. 256)
- Process Data with a Custom JAR (p. 258)

Build Binaries Using Amazon EMR

You can use Amazon EMR as a build environment to compile programs for use in your cluster. Programs that you use with Amazon EMR must be compiled on a system running the same version of Linux used by Amazon EMR. For a 32-bit version, you should have compiled on a 32-bit machine or with 32-bit cross compilation options turned on. For a 64-bit version, you need to have compiled on a 64-bit machine or with 64-bit cross compilation options turned. For more information about EC2 instance versions, go to Plan and Configure EC2 Instances (p. 140). Supported programming languages include C++, Cython, and C#.

The following table outlines the steps involved to build and test your application using Amazon EMR.
Process for Building a Module

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Connect to the master node of your cluster.</td>
</tr>
<tr>
<td>2</td>
<td>Copy source files to the master node.</td>
</tr>
<tr>
<td>3</td>
<td>Build binaries with any necessary optimizations.</td>
</tr>
<tr>
<td>4</td>
<td>Copy binaries from the master node to Amazon S3.</td>
</tr>
</tbody>
</table>

The details for each of these steps are covered in the sections that follow.

To connect to the master node of the cluster

- Follow these instructions to connect to the master node: Connect to the Master Node Using SSH (p. 457).

To copy source files to the master node

1. Put your source files in an Amazon S3 bucket. To learn how to create buckets and how to move data into Amazon S3, see the Amazon Simple Storage Service Getting Started Guide.
2. Create a folder on your Hadoop cluster for your source files by entering a command similar to the following:

   ```bash
   mkdir SourceFiles
   ```
3. Copy your source files from Amazon S3 to the master node by typing a command similar to the following:

   ```bash
   hadoop fs -get s3://mybucket/SourceFiles SourceFiles
   ```

Build binaries with any necessary optimizations

How you build your binaries depends on many factors. Follow the instructions for your specific build tools to setup and configure your environment. You can use Hadoop system specification commands to obtain cluster information to determine how to install your build environment.

To identify system specifications

- Use the following commands to verify the architecture you are using to build your binaries.
  a. To view the version of Debian, enter the following command:

     ```bash
     master$ cat /etc/issue
     ```

     The output looks similar to the following.

     Debian GNU/Linux 5.0

  b. To view the public DNS name and processor size, enter the following command:

     ```bash
     master$ uname -a
     ```

     The output looks similar to the following.
Once your binaries are built, you can copy the files to Amazon S3.

**To copy binaries from the master node to Amazon S3**

- Type the following command to copy the binaries to your Amazon S3 bucket:

```plaintext
hadoop fs -put BinaryFiles s3://mybucket/BinaryDestination
```

### JAR Requirements

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

When you launch an Amazon EMR cluster and add steps that and specify a JAR, Amazon EMR starts Hadoop and then executes your JAR using the `bin/hadoop jar` command.

Your JAR should not exit until your step is complete and then it should have an exit code to indicate successful completion (0). If there isn't an exit code indicating success, Amazon EMR assumes the step failed. Likewise, when your JAR completes, Amazon EMR assumes that all activity related to that step is complete.

### Run a Script in a Cluster

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Amazon EMR enables you to run a script at any time during step processing in your cluster. You specify a step that runs a script either when you create your cluster or you can add a step if your cluster is in the WAITING state. For more information about adding steps, see **Submit Work to a Cluster** (p. 493).

To run a script before step processing begins, use a bootstrap action. For more information about bootstrap actions, see **(Optional) Create Bootstrap Actions to Install Additional Software** (p. 129).
Submitting a Custom JAR Step Using the AWS CLI

This section describes how to add a step to run a script. The `script-runner.jar` takes arguments to the path to a script and any additional arguments for the script. The JAR file runs the script with the passed arguments.

**Important**

`script-runner.jar` is located at `s3://region.elasticmapreduce/libs/script-runner/script-runner.jar` where `region` is the region in which your EMR cluster resides.

The cluster containing a step that runs a script looks similar to the following examples.

**To add a step to run a script using the AWS CLI**

- To run a script using the AWS CLI, type the following command, replace `myKey` with the name of your EC2 key pair and replace `mybucket` with your S3 bucket. This cluster runs the script `my_script.sh` on the master node when the step is processed.

  - Linux, UNIX, and Mac OS X users:

    ```
    aws emr create-cluster --name "Test cluster" --ami-version 3.11 --applications Name=Hue Name=Hive Name=Pig \ 
    --use-default-roles --ec2-attributes KeyName=myKey \ 
    --instance-type m3.xlarge --instance-count 3 \ 
    --steps Type=CUSTOM_JAR,Name=CustomJAR,ActionOnFailure=CONTINUE,Jar=s3://region.elasticmapreduce/libs/script-runner/script-runner.jar,Args=["s3://mybucket/script-path/my_script.sh"]
    ```

  - Windows users:

    ```
    aws emr create-cluster --name "Test cluster" --ami-version 3.10 --applications Name=Hue Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey --instance-type m3.xlarge --instance-count 3 --steps Type=CUSTOM_JAR,Name=CustomJAR,ActionOnFailure=CONTINUE,Jar=s3://elasticmapreduce/libs/script-runner/script-runner.jar,Args=["s3://mybucket/script-path/my_script.sh"]
    ```

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

  **Note**

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

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

Process Data with Streaming

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.
Hadoop Streaming is a utility that comes with Hadoop that enables you to develop MapReduce executables in languages other than Java. Streaming is implemented in the form of a JAR file, so you can run it from the Amazon EMR API or command line just like a standard JAR file.

This section describes how to use Streaming with Amazon EMR.

**Note**
Apache Hadoop Streaming is an independent tool. As such, all of its functions and parameters are not described here. For more information about Hadoop Streaming, go to http://hadoop.apache.org/docs/stable/hadoop-streaming/HadoopStreaming.html.

**Using the Hadoop Streaming Utility**

This section describes how to use Hadoop's Streaming utility.

**Hadoop Process**

1. Write your mapper and reducer executable in the programming language of your choice. Follow the directions in Hadoop's documentation to write your streaming executables. The programs should read their input from standard input and output data through standard output. By default, each line of input/output represents a record and the first tab on each line is used as a separator between the key and value.

2. Test your executables locally and upload them to Amazon S3.

3. Use the Amazon EMR command line interface or Amazon EMR console to run your application.

Each mapper script launches as a separate process in the cluster. Each reducer executable turns the output of the mapper executable into the data output by the job flow.

The **input**, **output**, **mapper**, and **reducer** parameters are required by most Streaming applications. The following table describes these and other, optional parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>-input</td>
<td>Location on Amazon S3 of the input data. Type: String Default: None Constraint: URI. If no protocol is specified then it uses the cluster's default file system.</td>
<td>Yes</td>
</tr>
<tr>
<td>-output</td>
<td>Location on Amazon S3 where Amazon EMR uploads the processed data. Type: String Default: None Constraint: URI Default: If a location is not specified, Amazon EMR uploads the data to the location specified by input.</td>
<td>Yes</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Required</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>----------</td>
</tr>
<tr>
<td>-mapper</td>
<td>Name of the mapper executable.</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Type: String</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default: None</td>
<td></td>
</tr>
<tr>
<td>-reducer</td>
<td>Name of the reducer executable.</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Type: String</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default: None</td>
<td></td>
</tr>
<tr>
<td>-cacheFile</td>
<td>An Amazon S3 location containing files for Hadoop to copy into your local working directory (primarily to improve performance).</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Type: String</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default: None</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Constraints: [URI]#[symlink name to create in working directory]</td>
<td></td>
</tr>
<tr>
<td>-cacheArchive</td>
<td>JAR file to extract into the working directory</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Type: String</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default: None</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Constraints: [URI]#[symlink directory name to create in working directory]</td>
<td></td>
</tr>
<tr>
<td>-combiner</td>
<td>Combines results</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Type: String</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default: None</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Constraints: Java class name</td>
<td></td>
</tr>
</tbody>
</table>

The following code sample is a mapper executable written in Python. This script is part of the WordCount sample application.

```python
#!/usr/bin/python
import sys

def main(argv):
    line = sys.stdin.readline()
    try:
        while line:
            line = line.rstrip()
            words = line.split()
            for word in words:
                print "LongValueSum:" + word + "\t" + "1"
            line = sys.stdin.readline()
    except "end of file":
        return None
    if __name__ == "__main__":
        main(sys.argv)
```
Submit a Streaming Step

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

This section covers the basics of submitting a Streaming step to a cluster. A Streaming application reads input from standard input and then runs a script or executable (called a mapper) against each input. The result from each of the inputs is saved locally, typically on a Hadoop Distributed File System (HDFS) partition. After all the input is processed by the mapper, a second script or executable (called a reducer) processes the mapper results. The results from the reducer are sent to standard output. You can chain together a series of Streaming steps, where the output of one step becomes the input of another step.

The mapper and the reducer can each be referenced as a file or you can supply a Java class. You can implement the mapper and reducer in any of the supported languages, including Ruby, Perl, Python, PHP, or Bash.

Submit a Streaming Step Using the Console

This example describes how to use the Amazon EMR console to submit a Streaming step to a running cluster.

To submit a Streaming step

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. In the Cluster List, select the name of your cluster.
3. Scroll to the Steps section and expand it, then choose Add step.
4. In the Add Step dialog box:
   - For Step type, choose Streaming program.
   - For Name, accept the default name (Streaming program) or type a new name.
   - For Mapper, type or browse to the location of your mapper class in Hadoop, or an S3 bucket where the mapper executable, such as a Python program, resides. The path value must be in the form BucketName/path/MapperExecutable.
   - For Reducer, type or browse to the location of your reducer class in Hadoop, or an S3 bucket where the reducer executable, such as a Python program, resides. The path value must be in the form BucketName/path/MapperExecutable. Amazon EMR supports the special aggregate keyword. For more information, go to the Aggregate library supplied by Hadoop.
   - For Input S3 location, type or browse to the location of your input data.
   - For Output S3 location, type or browse to the name of your Amazon S3 output bucket.
   - For Arguments, leave the field blank.
   - For Action on failure, accept the default option (Continue).
5. Choose Add. The step appears in the console with a status of Pending.
6. The status of the step changes from Pending to Running to Completed as the step runs. To update the status, choose the Refresh icon above the Actions column.

AWS CLI

These examples demonstrate how to use the AWS CLI to create a cluster and submit a Streaming step.

To create a cluster and submit a Streaming step using the AWS CLI

- To create a cluster and submit a Streaming step using the AWS CLI, type the following command and replace myKey with the name of your EC2 key pair.
Process Data Using Cascading

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Cascading is an open-source Java library that provides a query API, a query planner, and a job scheduler for creating and running Hadoop MapReduce applications. Applications developed with Cascading are compiled and packaged into standard Hadoop-compatible JAR files similar to other native Hadoop applications. A Cascading step is submitted as a custom JAR in the Amazon EMR console. For more information about Cascading, go to http://www.cascading.org.

Topics

• Submit a Cascading Step (p. 256)

Submit a Cascading Step

This section covers the basics of submitting a Cascading step.
Submit a Cascading Step Using the Console

This example describes how to use the Amazon EMR console to submit a Cascading step to a running cluster as a custom JAR file.

**To submit a Cascading step using the console**

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. In the **Cluster List**, click the name of your cluster.
3. Scroll to the **Steps** section and expand it, then click Add step.
4. In the **Add Step** dialog:
   - For **Step type**, choose Custom jar.
   - For **Name**, accept the default name (Custom jar) or type a new name.
   - For **JAR location**, type or browse to the location of your script. The value must be in the form `BucketName/path/ScriptName`.
   - For **Arguments**, leave the field blank.
   - For **Action on failure**, accept the default option (Continue).
5. Click Add. The step appears in the console with a status of Pending.
6. The status of the step changes from Pending to Running to Completed as the step runs. To update the status, click the Refresh icon above the Actions column.

Launching a Cluster and Submitting a Cascading Step Using the AWS CLI

This example describes how to use the AWS CLI to create a cluster and submit a Cascading step. The Cascading SDK includes Cascading and Cascading-based tools such as Multitool and Load. For more information, go to [http://www.cascading.org/sdk/](http://www.cascading.org/sdk/).

**To create a cluster and submit a Cascading step using the AWS CLI**

- Type the following command to launch your cluster and submit a Cascading step. Replace `myKey` with the name of your EC2 key pair and replace `mybucket` with the name of your Amazon S3 bucket.

  ```bash
  aws emr create-cluster --name "Test cluster" --ami-version 2.4 --applications Name=Hive Name=Pig \
  --use-default-roles --ec2-attributes KeyName=myKey \ 
  --instance-type m3.xlarge --instance-count 3 \ 
  --bootstrap-actions Path=pathtobootstrapscript,Name="CascadingSDK" \ 
  --steps Type="CUSTOM_JAR",Name="Cascading Step",ActionOnFailure=CONTINUE,Jar=pathtojarfile,\ 
  Args=['-input','pathtoinputdata','-output','pathtooutputbucket','arg1','arg2']
  ```

- Windows users:

  ```bash
  aws emr create-cluster --name "Test cluster" --ami-version 2.4 --applications Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey --instance-type m3.xlarge --instance-count 3 --bootstrap-actions Path=pathtobootstrapscript,Name="CascadingSDK" --steps Type="CUSTOM_JAR",Name="Cascading Step",ActionOnFailure=CONTINUE,Jar=pathtojarfile,Args=['-input','pathtoinputdata','-output','pathtooutputbucket','arg1','arg2']
  ```
When you specify the instance count without using the --instance-groups parameter, a single Master node is launched, and the remaining instances are launched as core nodes. All nodes will use the instance type specified in the command.

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

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

---

### Process Data with a Custom JAR

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

A custom JAR runs a compiled Java program that you upload to Amazon S3. Compile the program against the version of Hadoop you want to launch and submit a CUSTOM_JAR step to your Amazon EMR cluster. For more information about compiling a JAR file, see [Build Binaries Using Amazon EMR (p. 249)](#).


### Submit a Custom JAR Step

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

This section covers the basics of submitting a custom JAR step in Amazon EMR. Submitting a custom JAR step enables you to write a script to process your data using the Java programming language.

### Submit a Custom JAR Step Using the Console

This example describes how to use the Amazon EMR console to submit a custom JAR step to a running cluster.

**To submit a custom JAR step using the console**

1. Open the Amazon EMR console at [https://console.aws.amazon.com/elasticmapreduce/](https://console.aws.amazon.com/elasticmapreduce/).
2. In the **Cluster List**, select the name of your cluster.
3. Scroll to the **Steps** section and expand it, then choose **Add step**.
4. In the **Add Step** dialog:
   - For **Step type**, choose **Custom JAR**.
   - For **Name**, accept the default name (Custom JAR) or type a new name.
   - For **JAR S3 location**, type or browse to the location of your JAR file. The value must be in the form `s3://BucketName/path/JARfile`.
   - For **Arguments**, type any required arguments as space-separated strings or leave the field blank.
   - For **Action on failure**, accept the default option (Continue).
5. Choose Add. The step appears in the console with a status of Pending.
6. The status of the step changes from Pending to Running to Completed as the step runs. To update the status, choose the Refresh icon above the Actions column.

Launching a cluster and submitting a custom JAR step using the AWS CLI

To launch a cluster and submit a custom JAR step using the AWS CLI

To launch a cluster and submit a custom JAR step using the AWS CLI, type the create-cluster subcommand with the --steps parameter.

- To launch a cluster and submit a custom JAR step, type the following command, replace myKey with the name of your EC2 key pair, and replace mybucket with your bucket name.

- Linux, UNIX, and Mac OS X users:

```bash
aws emr create-cluster --name "Test cluster" --ami-version 3.11.0 --applications Name=Hue Name=Hive Name=Pig \
--use-default-roles --ec2-attributes KeyName=myKey \
--instance-type m3.xlarge --instance-count 3 \
--steps Type=CUSTOM_JAR,Name="Custom JAR Step",ActionOnFailure=CONTINUE, Jar=pathtojarfile, \
Args=["pathToFile", "pathToOutput", "arg1", "arg2"]
```

- Windows users:

```bash
aws emr create-cluster --name "Test cluster" --ami-version 3.11.0 --applications Name=Hue Name=Hive Name=Pig \
--use-default-roles --ec2-attributes KeyName=myKey --instance-type m3.xlarge --instance-count 3 --steps Type=CUSTOM_JAR, Name="Custom JAR Step", ActionOnFailure=CONTINUE, Jar=pathtojarfile, Args=["pathToFile", "pathToOutput", "arg1", "arg2"]
```

**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 (^).

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

**Note**

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

For more information on using Amazon EMR commands in the AWS CLI, see [http://docs.aws.amazon.com/cli/latest/reference/emr](http://docs.aws.amazon.com/cli/latest/reference/emr).
Hive and Amazon EMR (EMR 3.x Releases)

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Hive is an open-source, data warehouse and analytic package that runs on top of Hadoop. Hive scripts use an SQL-like language called Hive QL (query language) that abstracts the MapReduce programming model and supports typical data warehouse interactions. Hive enables you to avoid the complexities of writing MapReduce programs in a lower level computer language, such as Java.

Hive extends the SQL paradigm by including serialization formats and the ability to invoke mapper and reducer scripts. In contrast to SQL, which only supports primitive value types (such as dates, numbers, and strings), values in Hive tables are structured elements, such as JSON objects, any user-defined data type, or any function written in Java.

For a more information on Hive, go to http://hive.apache.org/.

Topics
- How Amazon EMR Hive Differs from Apache Hive (p. 260)
- Supported Hive Versions (p. 271)
- Submit Hive Work (p. 278)
- Configuring an External Metastore for Hive (p. 280)
- Use the Hive JDBC Driver (p. 285)

How Amazon EMR Hive Differs from Apache Hive

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Topics
- Combine Splits Input Format (p. 261)
- Log files (p. 261)
- Thrift Service Ports (p. 262)
- Hive Authorization (p. 262)
- Hive File Merge Behavior with Amazon S3 (p. 262)
- ACID Transactions and Amazon S3 (p. 262)
- Additional Features of Hive in Amazon EMR (p. 262)

This section describes the differences between Amazon EMR Hive installations and the default versions of Hive available at http://svn.apache.org/viewvc/hive/branches/.
Note
With Hive 0.13.1 on Amazon EMR, certain options introduced in previous versions of Hive on
Amazon EMR have been removed in favor of greater parity with Apache Hive. For example, the -x option was removed.

Combine Splits Input Format

If you have many GZip files in your Hive cluster, you can optimize performance by passing multiple files
to each mapper. This reduces the number of mappers needed in your cluster and can help your clusters
complete faster. You do this by specifying that Hive use the HiveCombineSplitsInputFormat input
format and setting the split size, in bytes. This is shown in the following example

```
hive> set hive.input.format=org.apache.hadoop.hive.ql.io.HiveCombineSplitsInputFormat;
hive> set mapred.min.split.size=10000000;
```

Note
This functionality was deprecated with Hive 0.13.1. To get the same split input format
functionality, use the following:

```
set hive.hadoop.supports.splittable.combineinputformat=true;
```

Log files

Apache Hive saves Hive log files to /tmp/{user.name}/ in a file named hive.log. Amazon EMR saves
Hive logs to /mnt/var/log/apps/. In order to support concurrent versions of Hive, the version of Hive
that you run determines the log file name, as shown in the following table.

<table>
<thead>
<tr>
<th>Hive Version</th>
<th>Log File Name</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.13.1</td>
<td>hive.log</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Amazon EMR will now use an unversioned hive.log. Minor versions of will all share the same log location as the major version.</td>
</tr>
<tr>
<td>0.11.0</td>
<td>hive_0110.log</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Minor versions of Hive 0.11.0, such as 0.11.0.1, share the same log file location as Hive 0.11.0.</td>
</tr>
<tr>
<td>0.8.1</td>
<td>hive_081.log</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Minor versions of Hive 0.8.1, such as Hive 0.8.1.1, share the same log file location as Hive 0.8.1.</td>
</tr>
<tr>
<td>0.7.1</td>
<td>hive_07_1.log</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Minor versions of Hive 0.7.1, such as Hive 0.7.1.3 and Hive 0.7.1.4, share the same log file location as Hive 0.7.1.</td>
</tr>
<tr>
<td>0.7</td>
<td>hive_07.log</td>
<td></td>
</tr>
</tbody>
</table>
Thrift Service Ports

Thrift is an RPC framework that defines a compact binary serialization format used to persist data structures for later analysis. Normally, Hive configures the server to operate on the following ports.

<table>
<thead>
<tr>
<th>Hive Version</th>
<th>Log File Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>hive_05.log</td>
</tr>
<tr>
<td>0.4</td>
<td>hive.log</td>
</tr>
</tbody>
</table>

For more information about thrift services, go to http://wiki.apache.org/thrift/.

Hive Authorization

Amazon EMR supports Hive Authorization for HDFS but not for EMRFS and Amazon S3. Amazon EMR clusters run with authorization disabled by default.

Hive File Merge Behavior with Amazon S3

Apache Hive merges small files at the end of a map-only job if hive.merge.mapfiles is true and the merge is triggered only if the average output size of the job is less than the hive.merge.smallfiles.avgsize setting. Amazon EMR Hive has exactly the same behavior if the final output path is in HDFS; however, if the output path is in Amazon S3, the hive.merge.smallfiles.avgsize parameter is ignored. In that situation, the merge task is always triggered if hive.merge.mapfiles is set to true.

ACID Transactions and Amazon S3

ACID (Atomicity, Consistency, Isolation, Durability) transactions are not supported with Hive data stored in Amazon S3. If you attempt to create a transactional table in Amazon S3, this will cause an exception.

Additional Features of Hive in Amazon EMR

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.
Amazon EMR extends Hive with new features that support Hive integration with other AWS services, such as the ability to read from and write to Amazon Simple Storage Service (Amazon S3) and DynamoDB. For information about which versions of Hive support these additional features, see Hive Patches (p. 268).

Topics
- Write Data Directly to Amazon S3 (p. 263)
- Use Hive to Access Resources in Amazon S3 (p. 263)
- Use Hive to Recover Partitions (p. 264)
- Variables in Hive (p. 264)
- Amazon EMR Hive Queries to Accommodate Partial DynamoDB Schemas (p. 266)
- Copy Data Between DynamoDB Tables in Different AWS Regions (p. 267)
- Set DynamoDB Throughput Values Per Table (p. 268)
- Hive Patches (p. 268)

Write Data Directly to Amazon S3

The Hadoop Distributed File System (HDFS) and Amazon S3 are handled differently within Amazon EMR and Hive. The version of Hive installed with Amazon EMR is extended with the ability to write directly to Amazon S3 without the use of temporary files. This produces a significant performance improvement but it means that HDFS and Amazon S3 behave differently within Hive.

A consequence of Hive writing directly to Amazon S3 is that you cannot read and write to the same table within the same Hive statement if that table is located in Amazon S3. The following example shows how to use multiple Hive statements to update a table in Amazon S3.

To update a table in Amazon S3 using Hive

1. From a Hive prompt or script, create a temporary table in the cluster’s local HDFS filesystem.
2. Write the results of a Hive query to the temporary table.
3. Copy the contents of the temporary table to Amazon S3. This is shown in the following example.

```
CREATE TEMPORARY TABLE tmp LIKE my_s3_table;
INSERT OVERWRITE TABLE tmp SELECT ....;
INSERT OVERWRITE TABLE my_s3_table SELECT * FROM tmp;
```

Use Hive to Access Resources in Amazon S3

The version of Hive installed in Amazon EMR enables you to reference resources, such as JAR files, located in Amazon S3.

```
add jar s3://elasticmapreduce/samples/hive-ads/libs/jsonserde.jar
```

You can also reference scripts located in Amazon S3 to execute custom map and reduce operations. This is shown in the following example.

```
from logs select transform (line)
```
using 's3://mybucket/scripts/parse-logs.pl' as
(time string, exception_type string, exception_details string)

Use Hive to Recover Partitions

We added a statement to the Hive query language that recovers the partitions of a table from table data located in Amazon S3. The following example shows this.

```
CREATE EXTERNAL TABLE (json string) raw_impression
PARTITIONED BY (dt string)
LOCATION 's3://elastic-mapreduce/samples/hive-ads/tables/impressions';
ALTER TABLE logs RECOVER PARTITIONS;
```

The partition directories and data must be at the location specified in the table definition and must be named according to the Hive convention: for example, dt=2009-01-01.

**Note**
After Hive 0.13.1 this capability is supported natively using `msck repair table` and therefore `recover partitions` is not supported. For more information, see https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL.

Variables in Hive

You can include variables in your scripts by using the dollar sign and curly braces.

```
add jar ${LIB}/jsonserde.jar
```

You pass the values of these variables to Hive on the command line using the `-d` parameter, as in the following example:

```
-d LIB=s3://elasticmapreduce/samples/hive-ads/lib
```

You can also pass the values into steps that execute Hive scripts.

**To pass variable values into Hive steps using the console**

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. Choose **Create cluster**.
3. In the **Steps** section, for **Add Step**, choose **Hive Program** from the list and **Configure and add**.
4. In the **Add Step** dialog, specify the parameters using the following table as a guide, and then choose **Add**.

<table>
<thead>
<tr>
<th>Field</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Script S3 location*</td>
<td>Specify the URI where your script resides in Amazon S3. The value must be in the form <code>BucketName/path/ScriptName</code>. For example: s3://elasticmapreduce/samples/hive-ads/libs/response-time-stats.q.</td>
</tr>
</tbody>
</table>
### Additional Features of Hive in Amazon EMR

<table>
<thead>
<tr>
<th>Field</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input S3 location</td>
<td>Optionally, specify the URI where your input files reside in Amazon S3. The value must be in the form <code>BucketName/path</code>. If specified, this will be passed to the Hive script as a parameter named <code>INPUT</code>. For example: <code>s3://elasticmapreduce/samples/hive-ads/tables/</code>.</td>
</tr>
<tr>
<td>Output S3 location</td>
<td>Optionally, specify the URI where you want the output stored in Amazon S3. The value must be in the form <code>BucketName/path</code>. If specified, this will be passed to the Hive script as a parameter named <code>OUTPUT</code>. For example: <code>s3://mybucket/hive-ads/output/</code>.</td>
</tr>
<tr>
<td>Arguments</td>
<td>Optionally, enter a list of arguments (space-separated strings) to pass to Hive. If you defined a path variable in your Hive script named <code>${SAMPLE}</code>, for example: <code>CREATE EXTERNAL TABLE logs (requestBeginTime STRING, requestEndTime STRING, hostname STRING) PARTITIONED BY (dt STRING) ROW FORMAT serde 'com.amazon.elasticmapreduce.JsonSerde' WITH SERDEPROPERTIES ( 'paths'='requestBeginTime, requestEndTime, hostname' ) LOCATION '${SAMPLE}/tables/impressions';</code>. To pass a value for the variable, type the following in the Arguments window: <code>-d SAMPLE=s3://elasticmapreduce/samples/hive-ads/</code>.</td>
</tr>
</tbody>
</table>
| Action on Failure      | This determines what the cluster does in response to any errors. The possible values for this setting are:  
  - **Terminate cluster**: If the step fails, terminate the cluster. If the cluster has termination protection enabled AND keep alive enabled, it will not terminate.  
  - **Cancel and wait**: If the step fails, cancel the remaining steps. If the cluster has keep alive enabled, the cluster will not terminate.  
  - **Continue**: If the step fails, continue to the next step. |

5. Select values as necessary and choose Create cluster.

### To pass variable values into Hive steps using the AWS CLI

To pass variable values into Hive steps using the AWS CLI, use the `--steps` parameter and include an arguments list:

- To pass a variable into a Hive step using the AWS CLI, type the following command, replace `myKey` with the name of your EC2 key pair, and replace `mybucket` with your bucket name. In this example, `SAMPLE` is a variable value preceded by the `-d` switch. This variable is defined in the Hive script as: `${SAMPLE}`.

  ```bash
  aws emr create-cluster --name "Test cluster" --ami-version 3.9 --applications Name=Hue Name=Hive Name=Pig 
  --use-default-roles --ec2-attributes KeyName=myKey 
  --instance-type m3.xlarge --instance-count 3 
  --steps Type=Hive,Name="Hive Program",ActionOnFailure=CONTINUE,Args=[-f,s3://elasticmapreduce/samples/hive-ads/libs/response-time-stats.q,-d,INPUT=s3://elasticmapreduce/samples/hive-ads/tables,-d,OUTPUT=s3://mybucket/hive-ads/output/,-d,SAMPLE=s3://elasticmapreduce/samples/hive-ads/]
  ```
• Windows users:

```
aws emr create-cluster --name "Test cluster" --ami-version 3.9 --applications Name=Hue Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey --instance-type m3.xlarge --instance-count 3 --steps Type=Hive,Name="Hive Program",ActionOnFailure=CONTINUE,Args=[-f,s3://elasticmapreduce/samples/hive-ads/libs/response-time-stats.q,-d,INPUT=s3://elasticmapreduce/samples/hive-ads/tables,-d,OUTPUT=s3://mybucket/hive-ads/output/,-d,SAMPLE=s3://elasticmapreduce/samples/hive-ads/]
```

**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 (^).

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

**Note**

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

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

**To pass variable values into Hive steps using the Java SDK**

• The following example demonstrates how to pass variables into steps using the SDK. For more information, see [Class StepFactory](https://docs.aws.amazon.com/AWSJavaSDK/latest/api/com/amazonaws/emr/StepFactory.html) in the AWS SDK for Java API Reference.

```java
StepFactory stepFactory = new StepFactory();

StepConfig runHive = new StepConfig()
   .withName("Run Hive Script")
   .withActionOnFailure("TERMINATE_JOB_FLOW")
   .withHadoopJarStep(stepFactory.newRunHiveScriptStep("s3://mybucket/script.q",
   Lists.newArrayList("-d","LIB= s3://elasticmapreduce/samples/hive-ads/lib"));
```

**Amazon EMR Hive Queries to Accommodate Partial DynamoDB Schemas**

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Amazon EMR Hive provides maximum flexibility when querying DynamoDB tables by allowing you to specify a subset of columns on which you can filter data, rather than requiring your query to include all columns. This partial schema query technique is effective when you have a sparse database schema and want to filter records based on a few columns, such as filtering on time stamps.

The following example shows how to use a Hive query to:

• Create a DynamoDB table.
• Select a subset of items (rows) in DynamoDB and further narrow the data to certain columns.
• Copy the resulting data to Amazon S3.

```
DROP TABLE dynamodb;
DROP TABLE s3;

CREATE EXTERNAL TABLE dynamodb(hashKey STRING, recordTimeStamp BIGINT, fullColumn
map<String, String>)
STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'
TBLPROPERTIES (  
"dynamodb.table.name" = "myTable",
"dynamodb.throughput.read.percent" = ".1000",
"dynamodb.column.mapping" = "hashKey:HashKey,recordTimeStamp:RangeKey");

CREATE EXTERNAL TABLE s3(map<String, String>)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',
LOCATION 's3://bucketname/path/subpath/';

INSERT OVERWRITE TABLE s3 SELECT item fullColumn FROM dynamodb WHERE recordTimeStamp < "2012-01-01";
```

The following table shows the query syntax for selecting any combination of items from DynamoDB.

<table>
<thead>
<tr>
<th>Query example</th>
<th>Result description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT * FROM <code>table_name</code>;</td>
<td>Selects all items (rows) from a given table and includes data from all columns available for those items.</td>
</tr>
<tr>
<td>SELECT * FROM <code>table_name</code> WHERE <code>field_name</code> = <code>value</code>;</td>
<td>Selects some items (rows) from a given table and includes data from all columns available for those items.</td>
</tr>
<tr>
<td>SELECT <code>column1_name</code>, <code>column2_name</code>, <code>column3_name</code> FROM <code>table_name</code>;</td>
<td>Selects all items (rows) from a given table and includes data from some columns available for those items.</td>
</tr>
<tr>
<td>SELECT <code>column1_name</code>, <code>column2_name</code>, <code>column3_name</code> FROM <code>table_name</code> WHERE <code>field_name</code> = <code>value</code>;</td>
<td>Selects some items (rows) from a given table and includes data from some columns available for those items.</td>
</tr>
</tbody>
</table>

**Copy Data Between DynamoDB Tables in Different AWS Regions**

Amazon EMR Hive provides a `dynamodb.region` property you can set per DynamoDB table. When `dynamodb.region` is set differently on two tables, any data you copy between the tables automatically occurs between the specified regions.

The following example shows you how to create a DynamoDB table with a Hive script that sets the `dynamodb.region` property:

**Note**
Per-table region properties override the global Hive properties.

```
CREATE EXTERNAL TABLE dynamodb(hashKey STRING, recordTimeStamp BIGINT, fullColumn
map<String, String>)
STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'
TBLPROPERTIES (  
"dynamodb.table.name" = "myTable",
"dynamodb.region" = "eu-west-1",
```

267
Set DynamoDB Throughput Values Per Table

Amazon EMR Hive enables you to set the DynamoDB readThroughputPercent and writeThroughputPercent settings on a per table basis in the table definition. The following Amazon EMR Hive script shows how to set the throughput values. For more information about DynamoDB throughput values, see Specifying Read and Write Requirements for Tables.

```sql
CREATE EXTERNAL TABLE dynamodb(hashKey STRING, recordTimeStamp BIGINT, map<String, String> fullColumn)
STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'
TBLPROPERTIES (
  "dynamodb.table.name" = "myTable",
  "dynamodb.throughput.read.percent" = ".4",
  "dynamodb.throughput.write.percent" = "1.0",
  "dynamodb.column.mapping" = "hashKey:HashKey,recordTimeStamps:RangeKey");
```

Hive Patches

The Amazon EMR team has created the following patches for Hive.

<table>
<thead>
<tr>
<th>Patch</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write to Amazon S3</td>
<td>Supports moving data between different file systems, such as HDFS and Amazon S3. Adds support for file systems (such as Amazon S3) that do not provide a “move” operation. Removes redundant operations like moving data to and from the same location.</td>
</tr>
<tr>
<td><strong>Status:</strong> Submitted</td>
<td></td>
</tr>
<tr>
<td><strong>Fixed in AWS Hive Version:</strong> 0.4</td>
<td></td>
</tr>
<tr>
<td><strong>Fixed in Apache Hive Version:</strong> n/a (HIVE-2318)</td>
<td></td>
</tr>
<tr>
<td>Scripts in Amazon S3</td>
<td>Enables Hive to download the Hive scripts in Amazon S3 buckets and run them. Saves you the step of copying scripts to HDFS before running them.</td>
</tr>
<tr>
<td><strong>Status:</strong> Committed</td>
<td></td>
</tr>
<tr>
<td><strong>Fixed in AWS Hive Version:</strong> 0.4</td>
<td></td>
</tr>
<tr>
<td><strong>Fixed in Apache Hive Version:</strong> 0.7.0 (HIVE-1624)</td>
<td></td>
</tr>
<tr>
<td>Recover partitions</td>
<td>Allows you to recover partitions from table data located in Amazon S3 and Hive table data in HDFS.</td>
</tr>
<tr>
<td><strong>Status:</strong> Not Submitted</td>
<td></td>
</tr>
<tr>
<td><strong>Fixed in AWS Hive Version:</strong> 0.4</td>
<td></td>
</tr>
<tr>
<td><strong>Fixed in Apache Hive Version:</strong> n/a</td>
<td></td>
</tr>
<tr>
<td>Variables in Hive</td>
<td>Create a separate namespace (aside from HiveConf) for managing Hive variables. Adds support for setting variables on the command line using either <code>-define x=y</code> or <code>set hivevar:x=y</code>. Adds support for</td>
</tr>
<tr>
<td>Patch</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Patch</td>
<td>referencing variables in statements using <code>${var_name}</code>. Provides a means for differentiating between hiveconf, hivevar, system, and environment properties in the output of <code>set -v</code>.</td>
</tr>
<tr>
<td></td>
<td><strong>Status:</strong> Committed</td>
</tr>
<tr>
<td></td>
<td><strong>Fixed in AWS Hive Version:</strong> 0.4</td>
</tr>
<tr>
<td></td>
<td><strong>Fixed in Apache Hive Version:</strong> 0.8.0 (HIVE-2020)</td>
</tr>
<tr>
<td>Report progress while writing to Amazon S3</td>
<td>FileSinkOperator reports progress to Hadoop while writing large files, so that the task is not killed.</td>
</tr>
<tr>
<td></td>
<td><strong>Status:</strong> Not Submitted</td>
</tr>
<tr>
<td></td>
<td><strong>Fixed in AWS Hive Version:</strong> 0.5</td>
</tr>
<tr>
<td></td>
<td><strong>Fixed in Apache Hive Version:</strong> n/a</td>
</tr>
<tr>
<td>Fix compression arguments</td>
<td>Corrects an issue where compression values were not set correctly in FileSinkOperator, which resulted in uncompressed files.</td>
</tr>
<tr>
<td></td>
<td><strong>Status:</strong> Submitted</td>
</tr>
<tr>
<td></td>
<td><strong>Fixed in AWS Hive Version:</strong> 0.5</td>
</tr>
<tr>
<td></td>
<td><strong>Fixed in Apache Hive Version:</strong> n/a (HIVE-2266)</td>
</tr>
<tr>
<td>Fix UDAFPercentile to tolerate null percentiles</td>
<td>Fixes an issue where UDAFPercentile would throw a null pointer exception when passed null percentile list.</td>
</tr>
<tr>
<td></td>
<td><strong>Status:</strong> Committed</td>
</tr>
<tr>
<td></td>
<td><strong>Fixed in AWS Hive Version:</strong> 0.5</td>
</tr>
<tr>
<td></td>
<td><strong>Fixed in Apache Hive Version:</strong> 0.8.0 (HIVE-2298)</td>
</tr>
<tr>
<td>Fix hashCode method in DoubleWritable class</td>
<td>Fixes the hashCode() method of DoubleWritable class of Hive and prevents the HashMap (of type DoubleWritable) from behaving as LinkedList.</td>
</tr>
<tr>
<td></td>
<td><strong>Status:</strong> Committed</td>
</tr>
<tr>
<td></td>
<td><strong>Fixed in AWS Hive Version:</strong> 0.7</td>
</tr>
<tr>
<td></td>
<td><strong>Fixed in Apache Hive Version:</strong> 0.7.0 (HIVE-1629)</td>
</tr>
<tr>
<td>Recover partitions, version 2</td>
<td>Improved version of Recover Partitions that uses less memory.</td>
</tr>
<tr>
<td></td>
<td><strong>Status:</strong> Not Submitted</td>
</tr>
<tr>
<td></td>
<td><strong>Fixed in AWS Hive Version:</strong> 0.7</td>
</tr>
<tr>
<td></td>
<td><strong>Fixed in Apache Hive Version:</strong> n/a</td>
</tr>
<tr>
<td>Patch</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| HAVING clause | Use the HAVING clause to directly filter on groups by expressions (instead of using nested queries). Integrates Hive with other data analysis tools that rely on the HAVING expression.  
**Status:** Committed  
**Fixed in AWS Hive Version:** 0.7  
**Fixed in Apache Hive Version:** 0.7.0 (HIVE-1790) |
| Improve Hive query performance | Reduces startup time for queries spanning a large number of partitions.  
**Status:** Committed  
**Fixed in AWS Hive Version:** 0.7.1  
**Fixed in Apache Hive Version:** 0.8.0 (HIVE-2299) |
| Improve Hive query performance for Amazon S3 queries | Reduces startup time for Amazon S3 queries. Set Hive.optimize.s3.query=true to enable optimization.  
The optimization flag assumes that the partitions are stored in standard Hive partitioning format: "HIVE_TABLE_ROOT/partition1=value1/partition2=value2". This is the format used by Hive to create partitions when you do not specify a custom location.  
The partitions in an optimized query should have the same prefix, with HIVE_TABLE_ROOT as the common prefix.  
**Status:** Not Submitted  
**Fixed in AWS Hive Version:** 0.7.1  
**Fixed in Apache Hive Version:** n/a |
| Skip comments in Hive scripts | Fixes an issue where Hive scripts would fail on a comment line; now Hive scripts skip commented lines.  
**Status:** Committed  
**Fixed in AWS Hive Version:** 0.7.1  
**Fixed in Apache Hive Version:** 0.8.0 (HIVE-2259) |
| Limit Recover Partitions | Improves performance recovering partitions from Amazon S3 when there are many partitions to recover.  
**Status:** Not Submitted  
**Fixed in AWS Hive Version:** 0.8.1  
**Fixed in Apache Hive Version:** n/a |
## Supported Hive Versions

The default configuration for Amazon EMR is the latest version of Hive running on the latest AMI version. The following versions of Hive are available:

<table>
<thead>
<tr>
<th>Hive Version</th>
<th>Compatible Hadoop Versions</th>
<th>Hive Version Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.13.1</td>
<td>2.4.0</td>
<td>Introduces the following features, improvements, and backwards incompatibilities. For more information, see Apache Hive 0.13.1 Release Notes and Apache Hive 0.13.0 Release Notes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Vectorized query: processes thousand-row blocks instead of processing by row.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• In-memory cache: hot data kept in-memory for quick reads.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Faster plan serialization</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Support for DECIMAL and CHAR datatypes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Sub-query for IN, NOT IN, EXISTS and NOT EXISTS (correlated and uncorrelated)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• JOIN conditions in the WHERE clause</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Other feature contributed by Amazon EMR:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Includes an optimization to Hive windowing functions that allows them to scale to large data sets.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Notable backward incompatibilities:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Does not support -el flag for pushing error-logs to Amazon S3 bucket in case a query failed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Does not support RECOVER PARTITION syntax. Instead use the native capability, MSCK REPAIR.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Round(sum(c),2) over w1 -&gt; round(sum(c) over w1),2) (in several places). This syntax was changed in Hive 0.12. See HIVE-4214.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Default precision and scale was changed for DECIMAL. Compared to previous Hive versions, DECIMAL in Hive 13 is DECIMAL(10,0).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The default SerDe for RCFile-backed tables is LazyBinaryColumnarSerDe in Apache Hive 0.12 and above. This means tables that were created with Hive versions 0.12 or greater will not be able to read data files which were generated with Hive 0.11 correctly unless hive.default.rcfile.serde is set to org.apache.hadoop.hive.serde2.columnar.ColumnarSerDe. See HIVE-4475.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Other notes and known issues:</td>
</tr>
</tbody>
</table>
|              |                             | • When a Hive database is created with a custom location, the CREATE TABLE AS SELECT (CTAS) operation ignores it. It takes the location from parameter
### Supported Hive Versions

<table>
<thead>
<tr>
<th>Hive Version</th>
<th>Compatible Hadoop Versions</th>
<th>Hive Version Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>hive.metastore.warehouse.dir instead of the database's properties. See HIVE-3486.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• When a user loads data into a table using OVERWRITE with a different file it is not being overwritten. See HIVE-6209.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Since Amazon EMR uses HiveServer2, the username must be hadoop with no password.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The following patches Hive 0.14.0 patches were backported to this release:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• HIVE-6367</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• HIVE-6394</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• HIVE-6783</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• HIVE-6785</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• HIVE-6938</td>
</tr>
<tr>
<td>0.11.0.2</td>
<td>1.0.3 2.2.0</td>
<td>Introduces the following features and improvements. For more information, see Apache Hive 0.11.0 Release Notes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Adds the Parquet library.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fixes a problem related to the Avro serializer/deserializer accepting a schema URL in Amazon S3.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fixes a problem with Hive returning incorrect results with indexing turned on.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Change Hive's log level from DEBUG to INFO.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fixes a problem when tasks do not report progress while deleting files in Amazon S3 dynamic partitions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• This Hive version fixes the following issues:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• HIVE-4618</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• HIVE-4689</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• HIVE-4757</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• HIVE-4781</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• HIVE-4932</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• HIVE-4935</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• HIVE-4990</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• HIVE-5056</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• HIVE-5149</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• HIVE-5418</td>
</tr>
<tr>
<td>0.11.0.1</td>
<td>1.0.3 2.2.0</td>
<td>Creates symlink /home/hadoop/hive/lib/hive_contrib.jar for backward compatibility.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fixes a problem that prevents installation of Hive 0.11.0 with IAM roles.</td>
</tr>
<tr>
<td>Hive Version</td>
<td>Compatible Hadoop Versions</td>
<td>Hive Version Notes</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------</td>
<td>--------------------</td>
</tr>
</tbody>
</table>
| 0.11.0       | 1.0.3 2.2.0                | Introduces the following features and improvements. For more information, see Apache Hive 0.11.0 Release Notes.  
• Simplifies hive.metastore.uris and the hive.metastore.local configuration settings. (HIVE-2585)  
• Changes the internal representation of binary type to byte[]. (HIVE-3246)  
• Allows HiveStorageHandler.configureTableJobProperties() to signal to its handler whether the configuration is input or output. (HIVE-2773)  
• Add environment context to metastore Thrift calls. (HIVE-3252)  
• Adds a new, optimized row columnar file format. (HIVE-3874)  
• Implements TRUNCATE. (HIVE-446)  
• Adds LEAD/LAG/FIRST/LAST analytical windowing functions. (HIVE-896)  
• Adds DECIMAL data type. (HIVE-2693)  
• Supports Hive list bucketing/DML. (HIVE-3073)  
• Supports custom separator for file output. (HIVE-3682)  
• Supports ALTER VIEW AS SELECT. (HIVE-3834)  
• Adds method to retrieve uncompressed/compressed sizes of columns from RC files. (HIVE-3897)  
• Allows updating bucketing/sorting metadata of a partition through the CLI. (HIVE-3903)  
• Allows PARTITION BY/ORDER BY in OVER clause and partition function. (HIVE-4048)  
• Improves GROUP BY syntax. (HIVE-581)  
• Adds more query plan optimization rules. (HIVE-948)  
• Allows CREATE TABLE LIKE command to accept TBLPROPERTIES. (HIVE-3527)  
• Fixes sort-merge join with sub-queries. (HIVE-3633)  
• Supports altering partition column type. (HIVE-3672)  
• De-emphasizes mapjoin hint. (HIVE-3784)  
• Changes object inspectors to initialize based on partition metadata. (HIVE-3833)  
• Adds merge map-job followed by map-reduce job. (HIVE-3952)  
• Optimizes hive.enforce.bucketing and hive.enforce.sorting insert. (HIVE-4240)  

273
<table>
<thead>
<tr>
<th>Hive Version</th>
<th>Compatible Hadoop Versions</th>
<th>Hive Version Notes</th>
</tr>
</thead>
</table>
| 0.8.1.8      | 1.0.3                       | • Adds support for copying data between DynamoDB tables in different regions. For more information, see Copy Data Between DynamoDB Tables in Different AWS Regions (p. 267).
• Adds support in Amazon EMR Hive for queries that can specify a subset of columns on which to filter data, rather than requiring queries to include all columns. For more information, see Amazon EMR Hive Queries to Accommodate Partial DynamoDB Schemas (p. 266).
• Adds the ability to set the DynamoDB readThroughputPercent and writeThroughputPercent values per table at creation time. For more information, see Set DynamoDB Throughput Values Per Table (p. 268).
• Fixes an issue where a query on an Amazon S3 input path gets stuck if there are a large number of paths to list before the input path. |
<table>
<thead>
<tr>
<th>Hive Version</th>
<th>Compatible Hadoop Versions</th>
<th>Hive Version Notes</th>
</tr>
</thead>
</table>
| 0.8.1.7      | 1.0.3                      | • Fixes `ColumnPruner` so that it works on `LateralView`. (HIVE-3226)  
• Fixes `utc_from_timestamp` and `utc_to_timestamp` to return correct results. (HIVE-2803)  
• Fixes a `NullPointerException` error on a join query with authorization enabled. (HIVE-3225)  
• Improves `mapjoin` filtering in the `ON` condition. (HIVE-2101)  
• Preserves the filter on a `OUTER JOIN` condition while merging the join tree. (HIVE-3070)  
• Fixes ConcurrentModificationException on a lateral view used with `explode`. (HIVE-2540)  
• Fixes an issue where an insert into a table overwrites the existing table, if the table name contains an uppercase character. (HIVE-3062)  
• Fixes an issue where jobs fail when there are multiple aggregates in a query. (HIVE-3732)  
• Fixes a `NullPointerException` error in nested user-defined aggregation functions (UDAFs). (HIVE-1399)  
• Provides an error message when using a user-defined aggregation function (UDAF) in the place of a user-defined function (UDF). (HIVE-2956)  
• Fixes an issue where Timestamp values without a nano-second part break the following columns in a row. (HIVE-3090)  
• Fixes an issue where the move task is not picking up changes to `hive.exec.max.dynamic.partitions` set in the Hive CLI. (HIVE-2918)  
• Adds the ability to atomically add drop partitions from the metastore. (HIVE-2777)  
• Adds partition pruning pushdown to the database for non-string partitions. (HIVE-2702)  
• Adds support for merging small files in Amazon S3 at the end of a map-only job using the `hive.merge.mapfiles` parameter. If the output path is in Amazon S3, the `hive.merge.smallfiles.avgsize` setting is ignored. For more information, see Hive File Merge Behavior with Amazon S3 (p. 262) and Hive Configuration Variables.  
• Improves clean-up of junk files after an `INSERT OVERWRITE` command. |
<p>| 0.8.1.6      | 1.0.3                      | • Adds support for IAM roles. For more information, see Configure IAM Roles for Amazon EMR Permissions to AWS Services (p. 234) |</p>
<table>
<thead>
<tr>
<th>Hive Version</th>
<th>Compatible Hadoop Versions</th>
<th>Hive Version Notes</th>
</tr>
</thead>
</table>
| 0.8.1.5      | 1.0.3                     | - Adds support for the new DynamoDB binary data type.  
- Adds the patch Hive-2955, which fixes an issue where queries consisting only of metadata always return an empty value.  
- Adds the patch Hive-1376, which fixes an issue where Hive would crash on an empty result set generated by "where false" clause queries.  
- Fixes the RCFile interaction with Amazon Simple Storage Service (Amazon S3).  
- Replaces JetS3t with the AWS SDK for Java.  
- Uses BatchWriteItem for puts to DynamoDB.  
- Adds schemaless mapping of DynamoDB tables into a Hive table using a Hive `map<string, string>` column. |
| 0.8.1.4      | 1.0.3                     | Updates the HBase client on Hive clusters to version 0.92.0 to match the version of HBase used on HBase clusters. This fixes issues that occurred when connecting to an HBase cluster from a Hive cluster. |
| 0.8.1.3      | 1.0.3                     | Adds support for Hadoop 1.0.3. |
| 0.8.1.2      | 1.0.3, 0.20.205           | Fixes an issue with duplicate data in large clusters. |
| 0.8.1.1      | 1.0.3, 0.20.205           | Adds support for MapR and HBase. |
| 0.8.1        | 1.0.3, 0.20.205           | Introduces new features and improvements. The most significant of these are as follows. For more information about the changes in Hive 0.8.1, go to [Apache Hive 0.8.1 Release Notes](https://hadoop.apache.org/docs/r3.2.1/hive/faq.html#version-0.8.1).  
- Support Binary DataType ([HIVE-2380](https://issues.apache.org/jira/browse/HIVE-2380))  
- Support Timestamp DataType ([HIVE-2272](https://issues.apache.org/jira/browse/HIVE-2272))  
- Provide a Plugin Developer Kit ([HIVE-2244](https://issues.apache.org/jira/browse/HIVE-2244))  
- Support INSERT INTO append semantics ([HIVE-306](https://issues.apache.org/jira/browse/HIVE-306))  
- Support Per-Partition SerDe ([HIVE-2484](https://issues.apache.org/jira/browse/HIVE-2484))  
- Support Import/Export facilities ([HIVE-1918](https://issues.apache.org/jira/browse/HIVE-1918))  
- Support Bitmap Indexes ([HIVE-1803](https://issues.apache.org/jira/browse/HIVE-1803))  
- Support RCFile Block Merge ([HIVE-1950](https://issues.apache.org/jira/browse/HIVE-1950))  
- Incorporate Group By Optimization ([HIVE-1694](https://issues.apache.org/jira/browse/HIVE-1694))  
- Enable HiveServer to accept -hiveconf option ([HIVE-2139](https://issues.apache.org/jira/browse/HIVE-2139))  
- Support --auxpath option ([HIVE-2355](https://issues.apache.org/jira/browse/HIVE-2355))  
- Add a new builtns subproject ([HIVE-2523](https://issues.apache.org/jira/browse/HIVE-2523))  
- Insert overwrite table db.tname fails if partition already exists ([HIVE-2617](https://issues.apache.org/jira/browse/HIVE-2617))  
- Add a new input format that passes multiple GZip files to each mapper, so fewer mappers are needed. ([HIVE-2089](https://issues.apache.org/jira/browse/HIVE-2089))  
<table>
<thead>
<tr>
<th>Hive Version</th>
<th>Compatible Hadoop Versions</th>
<th>Hive Version Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.7.1.4</td>
<td>0.20.205</td>
<td>Prevents the &quot;SET&quot; command in Hive from changing the current database of the current session.</td>
</tr>
<tr>
<td>0.7.1.3</td>
<td>0.20.205</td>
<td>Adds the <code>dynamodb.retry.duration</code> option, which you can use to configure the timeout duration for retrying Hive queries against tables in Amazon DynamoDB. This version of Hive also supports the <code>dynamodb.endpoint</code> option, which you can use to specify the Amazon DynamoDB endpoint to use for a Hive table. For more information about these options, see Hive Options (p. 403).</td>
</tr>
<tr>
<td>0.7.1.2</td>
<td>0.20.205</td>
<td>Modifies the way files are named in Amazon S3 for dynamic partitions. It prepends file names in Amazon S3 for dynamic partitions with a unique identifier. Using Hive 0.7.1.2 you can run queries in parallel with <code>set hive.exec.parallel=true</code>. It also fixes an issue with filter pushdown when accessing DynamoDB with spare data sets.</td>
</tr>
<tr>
<td>0.7.1.1</td>
<td>0.20.205</td>
<td>Introduces support for accessing DynamoDB, as detailed in Export, Import, Query, and Join Tables in DynamoDB Using Amazon EMR (p. 393). It is a minor version of 0.7.1 developed by the Amazon EMR team. When specified as the Hive version, Hive 0.7.1.1 overwrites the Hive 0.7.1 directory structure and configuration with its own values. Specifically, Hive 0.7.1.1 matches Apache Hive 0.7.1 and uses the Hive server port, database, and log location of 0.7.1 on the cluster.</td>
</tr>
<tr>
<td>0.7.1</td>
<td>0.20.205, 0.20, 0.18</td>
<td>Improves Hive query performance for a large number of partitions and for Amazon S3 queries. Changes Hive to skip commented lines.</td>
</tr>
<tr>
<td>0.7</td>
<td>0.20, 0.18</td>
<td>Improves Recover Partitions to use less memory, fixes the <code>hashCode</code> method, and introduces the ability to use the HAVING clause to filter on groups by expressions.</td>
</tr>
<tr>
<td>0.5</td>
<td>0.20, 0.18</td>
<td>Fixes issues with <code>FileSinkOperator</code> and modifies <code>UDAFPercentile</code> to tolerate null percentiles.</td>
</tr>
<tr>
<td>0.4</td>
<td>0.20, 0.18</td>
<td>Introduces the ability to write to Amazon S3, run Hive scripts from Amazon S3, and recover partitions from table data stored in Amazon S3. Also creates a separate namespace for managing Hive variables.</td>
</tr>
</tbody>
</table>

The AWS CLI does not support installing specific Hive versions. When using the AWS CLI, the latest version of Hive included on the AMI is installed by default.

**Display the Hive Version**

You can view the version of Hive installed on your cluster using the console. In the console, the Hive version is displayed on the **Cluster Details** page. In the **Configuration Details** column, the **Applications** field displays the Hive version.
Share Data Between Hive Versions

You can take advantage of Hive bug fixes and performance improvements on your existing Hive clusters by upgrading your version of Hive. Different versions of Hive, however, have different schemas. To share data between two versions of Hive, you can create an external table in each version of Hive with the same LOCATION parameter.

To share data between Hive versions

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Start a cluster with the new version of Hive. This procedure assumes that you already have a cluster with the old version of Hive running.</td>
</tr>
</tbody>
</table>
| 2 | Configure the two clusters to allow communication:  
   | On the cluster with the old version of Hive, configure the insert overwrite directory to the location of the HDFS of the cluster with the new version of Hive. |
| 3 | Export and reimport the data. |

Submit Hive Work

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

This section demonstrates submitting Hive work to an Amazon EMR cluster. You can submit Hive work to your cluster interactively, or you can submit work as a cluster step using the console, CLI, or API. You can submit steps when the cluster is launched, or you can submit steps to a running cluster. For more information, see Submit Work to a Cluster (p. 493).

Submit Hive Work Using the Amazon EMR Console

This example describes how to use the Amazon EMR console to submit a Hive step to a running cluster. Whether you submit steps when the cluster is launched, or to a running cluster, the process for adding steps in the console is identical.

To submit a Hive step to a cluster using the console

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. In the Cluster List, click the name of your cluster.
3. Scroll to the Steps section and expand it, then click Add step.
4. In the Add Step dialog:
   - For Step type, choose Hive program.
   - For Name, accept the default name (Hive program) or type a new name.
   - For Script S3 location, type or browse to the location of your Hive script.
   - For Input S3 location, type or browse to the location of your input data.
   - For Output S3 location, type or browse to the name of your Amazon S3 output bucket. For more information about creating an output location, see Configure an Output Location (p. 40).
   - For Arguments, type your arguments as space-separated strings or leave the field blank.
   - For Action on failure, accept the default option (Continue).
5. Click Add. The step appears in the console with a status of Pending.
6. The status of the step changes from Pending to Running to Completed as the step runs. To update the status, click the **Refresh** icon above the Actions column.

## Submit Hive Work Using the AWS CLI

These examples describe how to use the AWS CLI to submit Hive work to a cluster. Using the CLI, you can submit steps when a cluster is launched, or you can submit steps to a long-running cluster.

### To launch a cluster and submit a Hive step using the AWS CLI

To submit a Hive step when the cluster is launched, type the `--steps` parameter, indicate the step type using the `Type` argument, and provide the necessary argument string.

- To launch a cluster and submit a Hive step, type the following command. Replace the `parameters` as appropriate.
  
  **Linux, UNIX, and Mac OS X users:**
  ```bash
  aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applications Name=Hue Name=Hive Name=Pig \
  --use-default-roles --ec2-attributes KeyName=myKey \ 
  --instance-type m3.xlarge --instance-count 3 \ 
  --steps Type=Hive,Name="Hive Program",ActionOnFailure=CONTINUE,Args=[-f,pathtohivescript,-d,INPUT=pathtoinputdata,-d,OUTPUT=pathtooutputbucket]
  ```
  
  **Windows users:**
  ```bash
  aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applications Name=Hue Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey --instance-type m3.xlarge --instance-count 3 --steps Type=Hive,Name="Hive Program",ActionOnFailure=CONTINUE,Args=[-f,pathtohivescript,-d,INPUT=pathtoinputdata,-d,OUTPUT=pathtooutputbucket]
  ```
  
  When you specify the instance count without using the `--instance-groups` parameter, a single master node is launched, and the remaining instances are launched as core nodes. All nodes use the instance type specified in the command.

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

  For more information about using Amazon EMR commands in the AWS CLI, see [AWS Command Line Interface Reference](https://docs.aws.amazon.com/cli/latest/reference/emr/).

### To submit a Hive step to a running cluster using the AWS CLI

To add a Hive step to a running cluster, type the `add-steps` subcommand with the `--steps` parameter, indicate the step type using the `Type` argument, and provide the necessary argument string.

- To submit a Hive step to a running cluster, type the following command. Replace `j-2AXXXXXXGAPLF` with the cluster ID, replace `[yourregion]` with the name of your region, and replace `mybucket` with the name of your Amazon S3 output location.
  
  **Linux, UNIX, and Mac OS X users:**
  ```bash
  aws emr add-steps --cluster-id j-2AXXXXXXGAPLF \ 
  ```
  
  **Windows users:**
  ```bash
  aws emr add-steps --cluster-id j-2AXXXXXXGAPLF \ 
  ```
Configuring an External Metastore for Hive

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

By default, Hive records metastore information in a MySQL database on the master node's file system. The metastore contains a description of the table and the underlying data on which it is built, including the partition names, data types, and so on. When a cluster terminates, all cluster nodes shut down, including the master node. When this happens, local data is lost because node file systems use ephemeral storage. If you need the metastore to persist, you must create an external metastore that exists outside the cluster.

You have two options for an external metastore:

- AWS Glue Data Catalog (Amazon EMR version 5.8.0 or later only).
  
  For more information, see Using the AWS Glue Data Catalog as the Metastore for Hive (p. 280).

- Amazon RDS or Amazon Aurora.
  
  For more information, see Using an External MySQL Database or Amazon Aurora (p. 282).

Using the AWS Glue Data Catalog as the Metastore for Hive

Using Amazon EMR version 5.8.0 or later, you can configure Hive to use the AWS Glue Data Catalog as its metastore. We recommend this configuration when you require a persistent metastore or a metastore shared by different clusters, services, and applications.

AWS Glue is a fully managed extract, transform, and load (ETL) service that makes it simple and cost-effective to categorize your data, clean it, enrich it, and move it reliably between various data stores. The AWS Glue Data Catalog provides a unified metadata repository across a variety of data sources and data formats, integrating with Amazon EMR as well as Amazon RDS, Amazon Redshift, Redshift Spectrum, Athena, and any application compatible with the Apache Hive metastore. AWS Glue crawlers can automatically infer schema from source data in Amazon S3 and store the associated metadata in the Data Catalog. For more information about the Data Catalog, see Populating the AWS Glue Data Catalog in the AWS Glue Developer Guide.

Separate charges apply for AWS Glue. There is a monthly rate for storing and accessing the metadata in the Data Catalog, an hourly rate billed per minute for AWS Glue ETL jobs and crawler runtime, and an hourly rate billed per minute for each provisioned development endpoint. The Data Catalog allows you
to store up to a million objects at no charge. If you store more than a million objects, you are charged USD$1 for each 100,000 objects over a million. An object in the Data Catalog is a table, partition, or database. For more information, see Glue Pricing.

**Important**

If you created tables using Amazon Athena or Amazon Redshift Spectrum before August 14, 2017, databases and tables are stored in an Athena-managed catalog, which is separate from the AWS Glue Data Catalog. To integrate Amazon EMR with these tables, you must upgrade to the AWS Glue Data Catalog. For more information, see Upgrading to the AWS Glue Data Catalog in the Amazon Athena User Guide.

### Specifying AWS Glue Data Catalog as the Metastore

You can specify the AWS Glue Data Catalog as the metastore using the AWS Management Console, AWS CLI, or Amazon EMR API. When you create a cluster using the CLI or API, you use the *hive-site* configuration classification to specify the Data Catalog. When you create a cluster using the console, you can specify the Data Catalog using Advanced Options or Quick Options.

**Note**

The option to use the Data Catalog is also available with HCatalog because Hive is installed with HCatalog.

#### To specify AWS Glue Data Catalog as the metastore using the console

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. Choose Create cluster, Go to advanced options.
3. For Release, choose emr-5.8.0 or later.
4. Under Release, select Hive or HCatalog.
5. Under AWS Glue Data Catalog settings select Use for Hive table metadata.
6. Choose other options for your cluster as appropriate, choose Next, and then configure other cluster options as appropriate for your application.

#### To specify the AWS Glue Data Catalog as the metastore using the AWS CLI or Amazon EMR API

- Specify the value for `hive.metastore.client.factory.class` using the *hive-site* classification as shown in the following example. For more information, see Configuring Applications.

Example Configuration JSON for Using the AWS Glue Data Catalog:

```json
[
{
"Classification": "hive-site",
"Properties": {
"hive.metastore.client.factory.class": "com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory",
"com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory",
"com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory"
},
},
]
```

### IAM Permissions

The EMR_EC2_DefaultRole must be allowed IAM permissions for AWS Glue actions. This is only a concern if you don't use the default AmazonElasticMapReduceforEC2Role managed policy and you
attach a customer-managed policy to the role. In this case, you need to configure the policy to allow permission to perform AWS Glue actions. Open the IAM console (https://console.aws.amazon.com/iam/) and view the contents of the AmazonElasticMapReduceforEC2Role managed policy to see the required AWS Glue actions to allow.

Unsupported Configurations, Functions, and Known Issues

The limitations listed below apply when using the AWS Glue Data Catalog as a metastore:

- Adding auxiliary JARs using the Hive shell is not supported. As a workaround, add auxiliary JARs into the Hive classpath (specified using hive.aux.jars.path).
- The VALUES keyword is not supported.
- Hive transactions are not supported.
- Renaming tables from within AWS Glue is not supported.
- Partition values containing quotes and apostrophes are not supported (for example, PARTITION (owner="Doe's").
- Table and partition statistics are not supported.
- Using Hive authorization is not supported.
- Hive constraints are not supported.
- Setting hive.metastore.partition.inherit.table.properties is not supported.
- Using the following metastore constants is not supported: BUCKET_COUNT, BUCKET_FIELD_NAME, DDL_TIME, FIELD_TO_DIMENSION, FILE_INPUT_FORMAT, FILE_OUTPUT_FORMAT, HIVE_FILTER_FIELD_LAST_ACCESS, HIVE_FILTER_FIELD.Owner, HIVE_FILTER_FIELD_PARAMS, IS_ARCHIVED, META_TABLE_COLUMNS, META_TABLE_COLUMN_TYPES, META_TABLE_DB, META_TABLE_LOCATION, META_TABLE_NAME, META_TABLE_PARTITION_COLUMNS, META_TABLE_SERDE, META_TABLE_STORAGE, ORIGINAL_LOCATION.
- When you use a predicate expression, explicit values must be on the right side of the comparison operator, or queries might fail.
  - Correct: SELECT * FROM mytable WHERE time > 11
  - Incorrect: SELECT * FROM mytable WHERE 11 > time
- We do not recommend using user-defined functions (UDFs) in predicate expressions. Queries may fail because of the way Hive tries to optimize query execution.
- Temporary tables are not supported.

Using an External MySQL Database or Amazon Aurora

To use an external MySQL database or Amazon Aurora as your Hive metastore, you override the default configuration values for the metastore in Hive to specify the external database location, either on an Amazon RDS MySQL instance or an Amazon Aurora instance.

Note
Hive neither supports nor prevents concurrent write access to metastore tables. If you share metastore information between two clusters, you must ensure that you do not write to the same metastore table concurrently, unless you are writing to different partitions of the same metastore table.

The following procedure shows you how to override the default configuration values for the Hive metastore location and start a cluster using the reconfigured metastore location.
To create a metastore located outside of the EMR cluster

1. Create a MySQL or Aurora database.

   For information about how to create an Amazon RDS database, see https://aws.amazon.com/rds/.

2. Modify your security groups to allow JDBC connections between your database and the ElasticMapReduce-Master security group.

   For information about how to modify your security groups for access, see https://aws.amazon.com/rds/faqs/#security.

3. Set the JDBC configuration values in hive-site.xml:
   a. Create a hive-site.xml configuration file containing the following information:

   ```xml
   <configuration>
   <property>
     <name>javax.jdo.option.ConnectionURL</name>
     <value>jdbc:mariadb://hostname:3306/hive?createDatabaseIfNotExist=true</value>
     <description>JDBC connect string for a JDBC metastore</description>
   </property>
   <property>
     <name>javax.jdo.option.ConnectionUserName</name>
     <value>hive</value>
     <description>Username to use against metastore database</description>
   </property>
   <property>
     <name>javax.jdo.option.ConnectionPassword</name>
     <value>password</value>
     <description>Password to use against metastore database</description>
   </property>
   </configuration>
   
   hostname is the DNS address of the Amazon RDS instance running the database. username and password are the credentials for your database. For more information about connecting to MySQL and Aurora database instances, see Connecting to a DB Instance Running the MySQL Database Engine and Connecting to an Aurora DB Cluster in the Amazon Relational Database Service User Guide.

   The JDBC drivers are installed by Amazon EMR.

   **Note**
   The value property should not contain any spaces or carriage returns. It should appear all on one line.

   b. Save your hive-site.xml file to a location on Amazon S3, such as s3://mybucket/hive-site.xml.

4. Create a cluster and specify the Amazon S3 location of the new Hive configuration file.

**AWS CLI**

To specify the location of the configuration file using the AWS CLI, type the following command, replace myKey with the name of your EC2 key pair, and replace mybucket with your Amazon S3 bucket name:

- Linux, UNIX, and Mac OS X users:

  ```bash
  aws emr create-cluster --name "Test cluster" --ami-version 3.10 --applications
  Name=Hue Name=Hive Name=Pig \
  --use-default-roles --ec2-attributes KeyName=myKey \
  --instance-type m3.xlarge --instance-count 3 \
  ```
--bootstrap-actions
  Name="Install Hive Site Configuration",
  Path="s3://region.elasticmapreduce/libs/hive/hive-script",
  Args=["--base-path", "s3://elasticmapreduce/libs/hive",
  "--install-hive-site",
  "--hive-site=s3://mybucket/hive-site.xml",
  "--hive-versions", "latest"]

Where \textit{region} is the region where you are launching your cluster.

- Windows users:
  
  ```
  aws emr create-cluster --name "Test cluster" --ami-version 3.10 --applications
  Name="Hive" Name="Pig" --use-default-roles --ec2-attributes
  KeyName=myKey --instance-type m3.xlarge --instance-count 3
  --bootstrap-actions
  Name="Install Hive Site Configuration",
  Path="s3://elasticmapreduce/libs/hive/hive-script",
  Args=["--base-path", "s3://region.elasticmapreduce/libs/hive",
  "--install-hive-site",
  "--hive-site=s3://mybucket/hive-site.xml",
  "--hive-versions", "latest"]
  ```

Where \textit{region} is the region where you are launching your cluster.

The us-east-1 endpoint used above, https://s3.amazonaws.com/, should be replaced with the Amazon S3 endpoint for the region in which your cluster will be located.

\textbf{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 (^).

When you specify the instance count without using the \textit{--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.

\textbf{Note}

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

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

5. Connect to the master node of your cluster.

Instructions on how to connect to the master node are available at \url{Connect to the Master Node Using SSH (p. 457)}.

6. Create your Hive tables specifying the location on Amazon S3 by entering a command similar to the following:

```sql
CREATE EXTERNAL TABLE IF NOT EXISTS table_name
(
  key int,
  value int
)
LOCATION s3://mybucket/hdfs/
```

7. Add your Hive script to the running cluster.

Your Hive cluster runs using the metastore located in Amazon RDS. Launch all additional Hive clusters that share this metastore by specifying the metastore location.
Use the Hive JDBC Driver

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. Amazon EMR supports both JDBC and ODBC connectivity.

To connect to Hive via JDBC requires you to download the JDBC driver and install a SQL client. The following example demonstrates using SQL Workbench/J to connect to Hive using JDBC.

**To download JDBC drivers**

- Download and extract the drivers appropriate to the versions of Hive that you want to access. The Hive version differs depending on the AMI that you choose when you create an Amazon EMR cluster.
  - Hive 0.13.1 JDBC drivers: https://amazon-odbc-jdbc-drivers.s3.amazonaws.com/public/AmazonHiveJDBC_1.0.4.1004.zip
  - Hive 0.11.0 JDBC drivers: https://aws.amazon.com/developertools/1982901737448217
  - Hive 0.8.1 JDBC drivers: https://aws.amazon.com/developertools/4897392426085727

**To install and configure SQL Workbench**

2. Go to the Installing and starting SQL Workbench/J page and follow the instructions for installing SQL Workbench/J on your system.
3. • Linux, Unix, Mac OS X users: In a terminal session, create an SSH tunnel to the master node of your cluster using the following command. Replace `master-public-dns-name` with the public DNS name of the master node and `path-to-key-file` with the location and file name of your Amazon EC2 private key (.pem) file.
   - Hive 0.13.1
     ```bash
     ssh -o ServerAliveInterval=10 -i path-to-key-file -N -L 10000:localhost:10000 hadoop@master-public-dns-name
     ```
   - Hive 0.11.0
     ```bash
     ssh -o ServerAliveInterval=10 -i path-to-key-file -N -L 10004:localhost:10004 hadoop@master-public-dns-name
     ```
   - Hive 0.8.1
     ```bash
     ssh -o ServerAliveInterval=10 -i path-to-key-file -N -L 10003:localhost:10003 hadoop@master-public-dns-name
     ```

   • Windows users: In a PuTTY session, create an SSH tunnel to the master node of your cluster (using local port forwarding) with the following settings. Replace `master-public-dns-name` with the public DNS name of the master node. For more information about creating an SSH tunnel to the master node, see Option 1: Set Up an SSH Tunnel to the Master Node Using Local Port Forwarding (p. 463).
4. Add the JDBC driver to SQL Workbench.
   a. In the Select Connection Profile dialog box, click Manage Drivers.
   b. Click the Create a new entry (blank page) icon.
   c. In the Name field, type Hive JDBC.
   d. For Library, click the Select the JAR file(s) icon.
   e. Browse to the location containing the extracted drivers, select the following JAR files and click Open.

<table>
<thead>
<tr>
<th>Hive driver version</th>
<th>JAR files to add</th>
</tr>
</thead>
</table>
| 0.13.1              | hive_metastore.jar
|                     | hive_service.jar
|                     | HiveJDBC3.jar
|                     | libfb303-0.9.0.jar
|                     | libthrift-0.9.0.jar
|                     | log4j-1.2.14.jar
|                     | ql.jar
|                     | slf4j-api-1.5.8.jar
|                     | slf4j-log4j12-1.5.8.jar
|                     | TCLIServiceClient.jar
| 0.11.0              | hadoop-core-1.0.3.jar
|                     | hive-exec-0.11.0.jar
|                     | hive-jdbc-0.11.0.jar
|                     | hive-metastore-0.11.0.jar
|                     | hive-service-0.11.0.jar
|                     | libfb303-0.9.0.jar
|                     | commons-logging-1.0.4.jar
|                     | slf4j-api-1.6.1.jar
| 0.8.1               | hadoop-core-0.20.205.jar
|                     | hive-exec-0.8.1.jar
|                     | hive-jdbc-0.8.1.jar
|                     | hive-metastore-0.8.1.jar
|                     | hive-service-0.8.1.jar
|                     | libfb303-0.7.0.jar
|                     | libthrift-0.7.0.jar
|                     | log4j-1.2.15.jar
|                     | slf4j-api-1.6.1.jar
|                     | slf4j-log4j12-1.6.1.jar

f. In the Please select one driver dialog box, select one of the following and click OK.
Use the Hive JDBC Driver

<table>
<thead>
<tr>
<th>Hive version</th>
<th>Driver classname</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.13.1</td>
<td>com.amazon.hive.jdbc3.HS2Driver</td>
</tr>
<tr>
<td>0.11.0</td>
<td>org.apache.hadoop.hive.jdbc.HiveDriver.jar</td>
</tr>
<tr>
<td>0.8.1</td>
<td>org.apache.hadoop.hive.jdbc.HiveDriver.jar</td>
</tr>
</tbody>
</table>

5. When you return to the Manage Drivers dialog box, verify that the Classname field is populated and click OK.

6. When you return to the Select Connection Profile dialog box, verify that the Driver field is set to Hive JDBC and provide the JDBC connection string in the URL field.

<table>
<thead>
<tr>
<th>Hive version</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.13.1</td>
<td>jdbc:hive2://localhost:10000/default</td>
</tr>
<tr>
<td>0.11.0</td>
<td>jdbc:hive://localhost:10004/default</td>
</tr>
<tr>
<td>0.8.1</td>
<td>jdbc:hive://localhost:10003/default</td>
</tr>
</tbody>
</table>

**Note**
If your cluster uses AMI version 3.3.1 or later, in the Select Connection Profile dialog box, type hadoop in the Username field.

7. Click OK to connect. After the connection is complete, connection details appear at the top of the SQL Workbench/J window.

For more information about using Hive and the JDBC interface, see HiveClient and HiveJDBCInterface in Apache Hive documentation.
Apache Spark

Apache Spark is a cluster framework and programming model that helps you do machine learning, stream processing, or graph analytics using Amazon EMR clusters. Similar to Apache Hadoop, Spark is an open-source, distributed processing system commonly used for big data workloads. However, Spark has several notable differences from Hadoop MapReduce. Spark has an optimized directed acyclic graph (DAG) execution engine and actively caches data in-memory, which can boost performance especially for certain algorithms and interactive queries.

Spark natively supports applications written in Scala, Python, and Java and includes several tightly integrated libraries for SQL (Spark SQL), machine learning (MLlib), stream processing (Spark Streaming), and graph processing (GraphX). These tools make it easier to leverage the Spark framework for a wide variety of use cases.

Spark can be installed alongside the other Hadoop applications available in Amazon EMR, and it can also leverage the EMR file system (EMRFS) to directly access data in Amazon S3. Hive is also integrated with Spark. So you can use a HiveContext object to run Hive scripts using Spark. A Hive context is included in the spark-shell as sqlContext.

To view an end-to-end example using Spark on Amazon EMR, see the New — Apache Spark on Amazon EMR post on the AWS official blog.

To view a machine learning example using Spark on Amazon EMR, see the Large-Scale Machine Learning with Spark on Amazon EMR post on the AWS Big Data blog.

Spark Release Information for This Release of Amazon EMR

Use Spark Interactively or in Batch Mode

Amazon EMR enables you to run Spark applications in two modes:

- Interactive
- Batch

When you launch a long-running cluster using the console or the AWS CLI, you can connect using SSH into the master node as the Hadoop user and use the Spark shell to develop and run your Spark applications interactively. Using Spark interactively enables you to prototype or test Spark applications...
more easily than in a batch environment. After you successfully revise the Spark application in interactive mode, you can put that application JAR or Python program in the file system local to the master node of the cluster on Amazon S3. You can then submit the application as a batch workflow. For more information about working with the shell, see Access the Spark Shell (p. 293).

In batch mode, upload your Spark script to Amazon S3 or the local master node file system, and then submit the work to the cluster as a step. Spark steps can be submitted to a long-running cluster or a transient cluster. For more information about submitting work to a cluster, see Submit Work to a Cluster (p. 493). For an example of launching a long-running cluster and submitting a Spark step, see Adding a Spark Step (p. 296).

Create a Cluster With Spark

To launch a cluster with Spark installed using the console

The following procedure creates a cluster with Spark installed. For more information about launching clusters with the console, see Step 3: Launch an Amazon EMR Cluster (p. 16).

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. Choose Create cluster.
3. For Software Configuration, choose the latest release.
4. For Applications to be installed, choose Spark from the list, then choose Configure and add.
5. You can add arguments to change the Spark configuration. For more information, see Configure Spark (EMR 3.x Releases) (p. 292). Choose Add.
6. Select other options as necessary and then choose Create cluster.

To launch a cluster with Spark installed using the AWS CLI

- Create the cluster with the following command:

```bash
aws emr create-cluster --name "Spark cluster" --ami-version 3.10.0 --applications Name=Spark,Name=Spark 
   --ec2-attributes KeyName=myKey --instance-type m1.xlarge --instance-count 3 --use-default-roles
```

**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 (^).

**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 (^).

To launch a cluster with Spark installed using the SDK for Java

Specify Spark as an application with SupportedProductConfig used in RunJobFlowRequest.

- The following Java program excerpt shows how to create a cluster with Spark:

```java
AmazonElasticMapReduceClient emr = new AmazonElasticMapReduceClient(credentials);
```
SupportedProductConfig sparkConfig = new SupportedProductConfig()
    .withName("Spark");

RunJobFlowRequest request = new RunJobFlowRequest()
    .withName("Spark Cluster")
    .withAmiVersion("3.11.0")
    .withNewSupportedProducts(sparkConfig)
    .withInstances(new JobFlowInstancesConfig()
        .withEc2KeyName("myKeyName")
        .withInstanceCount(1)
        .withKeepJobFlowAliveWhenNoSteps(true)
        .withMasterInstanceType("m3.xlarge")
        .withSlaveInstanceType("m3.xlarge")
    );

RunJobFlowResult result = emr.runJobFlow(request);

Using the AWS Glue Data Catalog as the Metastore for Spark SQL

Using Amazon EMR version 5.8.0 or later, you can configure Spark SQL to use the AWS Glue Data Catalog as its metastore. We recommend this configuration when you require a persistent metastore or a metastore shared by different clusters, services, and applications.

AWS Glue is a fully managed extract, transform, and load (ETL) service that makes it simple and cost-effective to categorize your data, clean it, enrich it, and move it reliably between various data stores. The AWS Glue Data Catalog provides a unified metadata repository across a variety of data sources and data formats, integrating with Amazon EMR as well as Amazon RDS, Amazon Redshift, Redshift Spectrum, Athena, and any application compatible with the Apache Hive metastore. AWS Glue crawlers can automatically infer schema from source data in Amazon S3 and store the associated metadata in the Data Catalog. For more information about the Data Catalog, see Populating the AWS Glue Data Catalog in the AWS Glue Developer Guide.

Separate charges apply for AWS Glue. There is a monthly rate for storing and accessing the metadata in the Data Catalog, an hourly rate billed per minute for AWS Glue ETL jobs and crawler runtime, and an hourly rate billed per minute for each provisioned development endpoint. The Data Catalog allows you to store up to a million objects at no charge. If you store more than a million objects, you are charged USD$1 for each 100,000 objects over a million. An object in the Data Catalog is a table, partition, or database. For more information, see Glue Pricing.

Important
If you created tables using Amazon Athena or Amazon Redshift Spectrum before August 14, 2017, databases and tables are stored in an Athena-managed catalog, which is separate from the AWS Glue Data Catalog. To integrate Amazon EMR with these tables, you must upgrade to the AWS Glue Data Catalog. For more information, see Upgrading to the AWS Glue Data Catalog in the Amazon Athena User Guide.

Specifying AWS Glue Data Catalog as the Metastore

You can specify the AWS Glue Data Catalog as the metastore using the AWS Management Console, AWS CLI, or Amazon EMR API. When you create a cluster using the CLI or API, you use the spark-hive-site configuration classification to specify the Data Catalog. When you create a cluster using the console, you can specify the Data Catalog using Advanced Options or Quick Options.

Note
The option to use AWS Glue Data Catalog is also available with Zeppelin because Zeppelin is installed with Spark SQL components.
To specify the AWS Glue Data Catalog as the metastore for Spark SQL using the console

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. Choose Create cluster, Go to advanced options.
3. For Release, choose emr-5.8.0 or later.
5. Under AWS Glue Data Catalog settings, select Use for Hive table metadata.
6. Choose other options for your cluster as appropriate, choose Next, and then configure other cluster options as appropriate for your application.

To specify the AWS Glue Data Catalog as the metastore using the AWS CLI or Amazon EMR API

• Specify the value for hive.metastore.client.factory.class using the spark-hive-site classification as shown in the following example. For more information, see Configuring Applications.

Example Example Configuration JSON for Using the AWS Glue Data Catalog

```json
[
  {
    "Classification": "spark-hive-site",
    "Properties": {
      "hive.metastore.client.factory.class": "com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory"
    }
  }
]
```

IAM Permissions

The EMR_EC2_DefaultRole must be allowed IAM permissions for AWS Glue actions. This is only a concern if you don't use the default AmazonElasticMapReduceforEC2Role managed policy and you attach a customer-managed policy to the role. In this case, you need to configure the policy to allow permission to perform AWS Glue actions. Open the IAM console (https://console.aws.amazon.com/iam/) and view the contents of the AmazonElasticMapReduceforEC2Role managed policy to see the required AWS Glue actions to allow.

Unsupported Configurations, Functions, and Known Issues

The limitations listed below apply when using the AWS Glue Data Catalog as a metastore:

• Having a default database without a location URI causes failures when you create a table. As a workaround, use the LOCATION clause to specify a bucket location, such as s3://mybucket, when you use CREATE TABLE. Alternatively create tables within a database other than the default database.
• Renaming tables from within AWS Glue is not supported.
• Partition values containing quotes and apostrophes are not supported (for example, PARTITION (owner="Doe's")).
• Table and partition statistics are not supported.
• Using Hive authorization is not supported.
• **Hive constraints** are not supported.
• Setting `hive.metastore.partition.inherit.table.properties` is not supported.
• Using the following metastore constants is not supported: `BUCKET_COUNT`, `BUCKET_FIELD_NAME`, `DDL_TIME`, `FIELD_TO_DIMENSION`, `FILE_INPUT_FORMAT`, `FILE_OUTPUT_FORMAT`, `HIVE_FILTER_FIELD_LAST_ACCESS`, `HIVE_FILTER_FIELD_OWNER`, `HIVE_FILTER_FIELD_PARAMS`, `IS_ARCHIVED`, `META_TABLE_COLUMNS`, `META_TABLE_COLUMN_TYPES`, `META_TABLE_DB`, `META_TABLE_LOCATION`, `META_TABLE_NAME`, `META_TABLE_PARTITION_COLUMNS`, `META_TABLE_SERDE`, `META_TABLE_STORAGE`, `ORIGINAL_LOCATION`.
• When you use a predicate expression, explicit values must be on the right side of the comparison operator, or queries might fail.
  • **Correct:** `SELECT * FROM mytable WHERE time > 11`
  • **Incorrect:** `SELECT * FROM mytable WHERE 11 > time`
• We do not recommend using user-defined functions (UDFs) in predicate expressions. Queries may fail because of the way Hive tries to optimize query execution.
• **Temporary tables** are not supported.

## Configure Spark (EMR 3.x Releases)

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

To configure Spark, you need to do so when you create your cluster by providing arguments. Spark on Amazon EMR is configured according to the bootstrap action located at GitHub. CLI and console will accept as arguments the options documented there. You can configure any of the options listed in the Spark Configuration topic in the Apache Spark documentation. You can view these configurations for existing clusters in the `$SPARK_CONF_DIR/spark-defaults.conf` configuration file.

You can supply the following arguments when creating a cluster:

- `-d key=value`
  
  Provide SparkConf settings to override `spark-defaults.conf`. To specify multiple options, prefix each key-value pair with `-d`.

- `-c S3_Path`

  The location of a `spark-install` configuration file stored on Amazon S3.

- `-g`

  Installs a Ganglia metrics configuration for Spark.

- `-a`

  Place `spark-assembly-*-jar` ahead of all system JARs on the Spark classpath.

- `-u S3_Path`

  Add the JARs at the given S3 URI to the Spark classpath. This option takes precedence over the `-a` option so, if specified together with `-a`, the JARs added will precede the `spark-assembly-*-jar`.

- `-l logging_level`

  Sets the logging level for `log4j.logger.org.apache.spark` of the driver. Options are: OFF, ERROR, WARN, INFO, DEBUG, or ALL. Default is INFO.

- `-h`

  Use Amazon EMR custom Hive JARs instead of those provided with Spark.
Changing Spark Default Settings

The following procedures show how to set executor settings using the CLI or console.

**To create a cluster with spark.executor.memory set to 2G using the CLI**

- Create a cluster with Spark installed and spark.executor.memory set to 2G, using the following:

```bash
aws emr create-cluster --name "Spark cluster" --ami-version 3.11.0 --applications Name=Spark,\nArgs=[-d,spark.executor.memory=2G] --ec2-attributes KeyName=myKey --instance-type m3.xlarge --instance-count 3 --use-default-roles
```

**To create a cluster with spark.executor.memory set to 2G using the console**

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. Choose **Create cluster**.
3. For the **Software Configuration** field, choose **Amazon AMI Version 3.9.0** or later.
4. For the **Applications to be installed** field, choose **Spark** from the list, then choose **Configure and add**. Then you put the argument, `spark.executor.memory=2G` into the arguments box and select **Add**.
5. Select other options as necessary and then choose **Create cluster**.

Access the Spark Shell

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

The Spark shell is based on the Scala REPL (Read-Eval-Print-Loop). It allows you to create Spark programs interactively and submit work to the framework. You can access the Spark shell by connecting to the master node with SSH and invoking `spark-shell`. For more information about connecting to the master node, see [Connect to the Master Node Using SSH](p. 457)
. The following examples use Apache HTTP Server access logs stored in Amazon S3.

**Note**

The bucket used in these examples is available to clients that can access US East (N. Virginia).

By default, the Spark shell creates its own `SparkContext` object called `sc`. You can use this context if it is required within the REPL. `sqlContext` is also available in the shell and it is a `HiveContext`.

**Example Using the Spark shell to count the occurrences of a string in a file stored in Amazon S3**

This example uses `sc` to read a `textFile` in Amazon S3.
When using Amazon EMR release version 5.11.0 and later, the `aws-sagemaker-spark-sdk` component is installed along with Spark. This component installs Amazon SageMaker Spark and associated dependencies for Spark integration with Amazon SageMaker. You can use Amazon SageMaker Spark to construct Spark machine learning (ML) pipelines using Amazon SageMaker stages. For more information, see the Amazon SageMaker Spark Readme on GitHub and Using Apache Spark with Amazon SageMaker in the Amazon SageMaker Developer Guide.
Write a Spark Application

Spark applications can be written in Scala, Java, or Python. There are several examples of Spark applications located on Spark Examples topic in the Apache Spark documentation. The Estimating Pi example is shown below in the three natively supported applications. You can also view complete examples in $SPARK_HOME/examples and at GitHub. For more information about how to build JARs for Spark, see the Quick Start topic in the Apache Spark documentation.

Scala

```scala
package org.apache.spark.examples
import scala.math.random
import org.apache.spark._

/** Computes an approximation to pi */
object SparkPi {
  def main(args: Array[String]) {
    val conf = new SparkConf().setAppName("Spark Pi")
    val spark = new SparkContext(conf)
    val slices = if (args.length > 0) args(0).toInt else 2
    val n = math.min(100000L * slices, Int.MaxValue).toInt // avoid overflow
    val count = spark.parallelize(1 until n, slices).map { i =>
      val x = random * 2 - 1
      val y = random * 2 - 1
      if (x*x + y*y < 1) 1 else 0
    }.reduce(_ + _)
    println("Pi is roughly " + 4.0 * count / n)
    spark.stop()
  }
}
```

Java

```java
package org.apache.spark.examples;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.api.java.JavaSparkContext;
import org.apache.spark.api.java.function.Function;
import org.apache.spark.api.java.function.Function2;
import java.util.ArrayList;
import java.util.List;
/**
 * Computes an approximation to pi
 * Usage: JavaSparkPi [slices]
 */
public final class JavaSparkPi {
  public static void main(String[] args) throws Exception {
    SparkConf sparkConf = new SparkConf().setAppName("JavaSparkPi");
    JavaSparkContext jsc = new JavaSparkContext(sparkConf);
    int slices = (args.length == 1) ? Integer.parseInt(args[0]) : 2;
```

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.
```
import sys
from random import random
from operator import add
from pyspark import SparkContext

if __name__ == "__main__":
    
    Usage: pi [partitions]
    
    sc = SparkContext(appName="PythonPi")
    partitions = int(sys.argv[1]) if len(sys.argv) > 1 else 2
    n = 100000 * partitions

    def f(_):
        x = random() * 2 - 1
        y = random() * 2 - 1
        return 1 if x ** 2 + y ** 2 < 1 else 0

    count = sc.parallelize(xrange(1, n + 1), partitions).map(f).reduce(add)
    print "Pi is roughly %.4f" % (4.0 * count / n)
    sc.stop()
```

```
296
```
You can use Amazon EMR Steps (p. 8) to submit work to the Spark framework installed on an EMR cluster. In the console and CLI, you do this using a Spark application step, which runs the `spark-submit` script as a step on your behalf. With the API, you use a step to invoke `spark-submit` using `command-runner.jar`.

For more information about submitting applications to Spark, see the Submitting Applications topic in the Apache Spark documentation.

**Note**
If you choose to deploy work to Spark using the client deploy mode, your application files must be in a local path on the EMR cluster. You cannot currently use S3 URIs for this location in client mode. However, you can use S3 URIs with cluster deploy mode.

### To submit a Spark step using the console

1. Open the Amazon EMR console at [https://console.aws.amazon.com/elasticmapreduce/](https://console.aws.amazon.com/elasticmapreduce/).
2. In the Cluster List, choose the name of your cluster.
3. Scroll to the Steps section and expand it, then choose Add step.
4. In the Add Step dialog box:
   - For Step type, choose Spark application.
   - For Name, accept the default name (Spark application) or type a new name.
   - For Deploy mode, choose Cluster or Client mode. Cluster mode launches your driver program on the cluster (for JVM-based programs, this is `main()`), while client mode launches the driver program locally. For more information, see Cluster Mode Overview in the Apache Spark documentation.
     **Note**
     Cluster mode allows you to submit work using S3 URIs. Client mode requires that you put the application in the local file system on the cluster master node.
   - Specify the desired Spark-submit options. For more information about `spark-submit` options, see Launching Applications with `spark-submit`.
   - For Application location, specify the local or S3 URI path of the application.
   - For Arguments, leave the field blank.
   - For Action on failure, accept the default option (Continue).
5. Choose Add. The step appears in the console with a status of Pending.
6. The status of the step changes from Pending to Running to Completed as the step runs. To update the status, choose the Refresh icon above the Actions column.
7. The results of the step are located in the Amazon EMR console Cluster Details page next to your step under Log Files if you have logging configured. You can optionally find step information in the log bucket you configured when you launched the cluster.

### To submit work to Spark using the AWS CLI

Submit a step when you create the cluster or use the `aws emr add-steps` subcommand in an existing cluster.

1. Use create-cluster.
   - Linux, UNIX, and Mac OS X users:
     ```shell
     aws emr create-cluster --name "Add Spark Step Cluster" --ami-version 3.11.0 --applications Name=Spark\ --ec2-attributes KeyName=myKey --instance-type m3.xlarge --instance-count 3 \   ```
Adding a Spark Step

```
--steps Type=Spark,Name="Spark Program",ActionOnFailure=CONTINUE,Args=[--class,org.apache.spark.examples.SparkPi,/home/hadoop/spark/lib/spark-examples-1.3.1-hadoop2.4.0.jar,10]
```

- **Windows users:**

```
aws emr create-cluster --name "Add Spark Step Cluster" --ami-version 3.11.0 --applications Name=Spark --ec2-attributes KeyName=myKey --instance-type m3.xlarge --instance-count 3 --steps Type=Spark,Name="Spark Program",ActionOnFailure=CONTINUE,Args=[--class,org.apache.spark.examples.SparkPi,/home/hadoop/spark/lib/spark-examples-1.3.1-hadoop2.4.0.jar,10]
```

**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 (^).

2. Alternatively, add steps to a cluster already running. Use `add-steps`.

- **Linux, UNIX, and Mac OS X users:**

```
aws emr add-steps --cluster-id j-2AXXXXXXGAPLF --steps Type=Spark,Name="Spark Program",\ 
    Args=[--class,org.apache.spark.examples.SparkPi,/home/hadoop/spark/lib/spark-examples-*.jar,10]
```

- **Windows users:**

```
aws emr add-steps --cluster-id j-2AXXXXXXGAPLF --steps Type=Spark,Name="Spark Program", \ 
    Args=[--class,org.apache.spark.examples.SparkPi,/home/hadoop/spark/lib/spark-*.jar,10]
```

To submit work to Spark using the SDK for Java

- To submit work to a cluster, use a step to run the `spark-submit` script on your EMR cluster. Add the step using the `addJobFlowSteps` method in `AmazonElasticMapReduceClient`:

```
AWSCredentials credentials = new BasicAWSCredentials(accessKey, secretKey);
AmazonElasticMapReduceClient emr = new AmazonElasticMapReduceClient(creds);
StepFactory stepFactory = new StepFactory();
AddJobFlowStepsRequest req = new AddJobFlowStepsRequest();
req.withJobFlowId("j-1X48XXXXXHCB");
List<StepConfig> stepConfigs = new ArrayList<StepConfig>();
StepConfig sparkStep = new StepConfig() .withName("Spark Step") .withActionOnFailure("CONTINUE") .withHadoopJarStep(stepFactory.newScriptRunnerStep("/home/hadoop/spark/bin/spark-submit", "--class", "org.apache.spark.examples.SparkPi", "/home/hadoop/spark/lib/spark-examples-1.3.1-hadoop2.4.0.jar", "10"));
stepConfigs.add(sparkStep);
req.withSteps(stepConfigs);
AddJobFlowStepsResult result = emr.addJobFlowSteps(req);
```
View the results of the step by examining the logs for the step. You can do this in the AWS Management Console if you have enabled logging by choosing Steps, selecting your step, and then, for Log files, choosing either stdout or stderr. To see the logs available, choose View Logs.

**Overriding Spark Default Configuration Settings**

You may want to override Spark default configuration values on a per-application basis. You can do this when you submit applications using a step, which essentially passes options to `spark-submit`. For example, you may wish to change the memory allocated to an executor process by changing `spark.executor.memory`. You would supply the `--executor-memory` switch with an argument like the following:

```
/home/hadoop/spark/bin/spark-submit --executor-memory 1g --class org.apache.spark.examples.SparkPi /home/hadoop/spark/lib/spark-examples*.jar 10
```

Similarly, you can tune `--executor-cores` and `--driver-memory`. In a step, you would provide the following arguments to the step:

```
--executor-memory 1g --class org.apache.spark.examples.SparkPi /home/hadoop/spark/lib/spark-examples*.jar 10
```

You can also tune settings that may not have a built-in switch using the `--conf` option. For more information about other settings that are tunable, see the Dynamically Loading Spark Properties topic in the Apache Spark documentation.

**View Spark Application History**

Amazon EMR collects information from YARN applications on your cluster and keeps historical information for up to seven days after applications have completed. Detailed information is available for Spark applications. This feature enables you to view data that is otherwise available in the Spark History Server’s web interface, but without having to establish a potentially complicated SSH tunnel for connecting to the cluster’s master node. You can also identify and open relevant log files more easily than you would by connecting to the master node and searching the file structure.

**To view Spark application history**

1. From the Clusters list, select the Name of a cluster and then select Application history.

   Each application is listed by Application ID.

   ![Application List]

2. Expand a row to see basic diagnostic information for the application or select the Application ID to view additional details.

   ![Application Details]

For more information, see View Application History in the Amazon EMR Management Guide.
What Can I Do With Impala?

Similar to using Hive with Amazon EMR, leveraging Impala with Amazon EMR can implement sophisticated data-processing applications with SQL syntax. However, Impala is built to perform faster in certain use cases (see below). With Amazon EMR, you can use Impala as a reliable data warehouse to execute tasks such as data analytics, monitoring, and business intelligence. Here are three use cases:

- **Use Impala instead of Hive on long-running clusters to perform ad hoc queries.** Impala reduces interactive queries to seconds, making it an excellent tool for fast investigation. You could run Impala on the same cluster as your batch MapReduce work flows, use Impala on a long-running analytics cluster with Hive and Pig, or create a cluster specifically tuned for Impala queries.

- **Use Impala instead of Hive for batch ETL jobs on transient Amazon EMR clusters.** Impala is faster than Hive for many queries, which provides better performance for these workloads. Like Hive, Impala uses SQL, so queries can easily be modified from Hive to Impala.

- **Use Impala in conjunction with a third-party business intelligence tool.** Connect a client ODBC or JDBC driver with your cluster to use Impala as an engine for powerful visualization tools and dashboards.

Both batch and interactive Impala clusters can be created in Amazon EMR. For instance, you can have a long-running Amazon EMR cluster running Impala for ad hoc, interactive querying or use transient Impala clusters for quick ETL workflows.

Differences from Traditional Relational Databases

Traditional relational database systems provide transaction semantics and database atomicity, consistency, isolation, and durability (ACID) properties. They also allow tables to be indexed and cached
so that small amounts of data can be retrieved very quickly and provide for fast update of small amounts of data and for enforcement of referential integrity constraints. Typically, they run on a single large machine and do not provide support for acting over complex user defined data types.

Impala uses a similar distributed query system to that found in RDBMSs, but queries data stored in HDFS and uses the Hive metastore to hold information about the input data. As with Hive, the schema for a query is provided at runtime, allowing for easier schema changes. Also, Impala can query a variety of complex data types and execute user defined functions. However, because Impala processes data in-memory, it is important to understand the hardware limitations of your cluster and optimize your queries for the best performance.

Differences from Hive

Impala executes SQL queries using a massively parallel processing (MPP) engine, while Hive executes SQL queries using MapReduce. Impala avoids Hive's overhead from creating MapReduce jobs, giving it faster query times than Hive. However, Impala uses significant memory resources and the cluster's available memory places a constraint on how much memory any query can consume. Hive is not limited in the same way, and can successfully process larger data sets with the same hardware.

Generally, you should use Impala for fast, interactive queries, while Hive is better for ETL workloads on large datasets. Impala is built for speed and is great for ad hoc investigation, but requires a significant amount of memory to execute expensive queries or process very large datasets. Because of these limitations, Hive is recommended for workloads where speed is not as crucial as completion.

Note
With Impala, you may experience performance gains over Hive, even when using standard instance types. For more information, see Impala Performance Testing and Query Optimization (p. 310).

Tutorial: Launching and Querying Impala Clusters on Amazon EMR

Important
Amazon EMR installing Impala at cluster creation time is not supported in current release versions of Amazon EMR. The examples and tutorial in this section require Amazon EMR release versions 3.11.0, 3.10.0, and 3.9.0 and an older version of Impala (1.2.4) is installed.

The instructions in this tutorial include how to:

- Sign up for Amazon EMR
- Launch a long-running cluster with Impala installed
- Connect to the cluster using SSH
- Generate a test data set
- Create Impala tables and populate them with data
- Perform interactive queries on Impala tables

Amazon EMR provides several tools you can use to launch and manage clusters: the console, a CLI, an API, and several SDKs. For more information about these tools, see What Tools are Available for Amazon EMR? (p. 11).
Sign up for the Service

If you don’t already have an AWS account, you’ll need to get one. Your AWS account gives you access to all services, but you are charged only for the resources that you use. For this example walk-through, the charges will be minimal.

To sign up for AWS

2. Follow the on-screen instructions.

AWS notifies you by email when your account is active and available for you to use.

Launch the Cluster

The next step is to launch the cluster. This tutorial provides the steps to launch a long-running cluster using the Amazon EMR console and CLI. Choose the method that best meets your needs. When you launch the cluster, Amazon EMR provisions EC2 instances (virtual servers) to perform the computation. These EC2 instances are preloaded with an Amazon Machine Image (AMI) that has been customized for Amazon EMR and which has Hadoop and other big data applications preloaded.

To add Impala to a cluster using the console

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. Choose Create cluster, Go to advanced options.
3. Choose a Release of 3.11.0, 3.10.0, or 3.9.0.
4. Choose Impala 1.2.4.
5. Specify Arguments for Impala to execute.
6. Choose other applications to install and specify steps as appropriate for your application. Choose Next.

To add Impala to a cluster using the AWS CLI

To add Impala to a cluster using the AWS CLI, type the create-cluster subcommand with the --applications parameter.

• To install Impala on a cluster, type the following command and replace myKey with the name of your EC2 key pair.
  - Linux, UNIX, and Mac OS X users:
    ```bash
    aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applications Name=Hue Name=Hive Name=Pig Name=Impala \ 
    --use-default-roles --ec2-attributes KeyName=myKey \ 
    --instance-type m3.xlarge --instance-count 3
    ```
  - Windows users:
    ```bash
    aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applications Name=Hue Name=Hive Name=Pig Name=Impala --use-default-roles --ec2-attributes KeyName=myKey --instance-type m3.xlarge --instance-count 3
    ```
When you specify the instance count without using the --instance-groups parameter, a single Master node is launched, and the remaining instances are launched as core nodes. All nodes will use the instance type specified in the command.

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

## Generate Test Data

### To generate the test data on the master node

1. Connect to the master node of the cluster using SSH and run the commands shown in the following steps. Your client operating system determines which steps to use to connect to the cluster. For more information, see *Connect to the Cluster* (p. 456).
2. In the SSH window, from the home directory, create and navigate to the directory that will contain the test data using the following commands:

   ```
   mkdir test
cd test
   ```

3. Download the JAR containing a program that automatically creates the test data using the following command:

   ```
   wget http://elasticmapreduce.s3.amazonaws.com/samples/impala/dbgen-1.0-jar-with-dependencies.jar
   ```

4. Launch the program to create the test data using the following command. In this example, the command-line parameters specify an output path of /mnt/dbgen, and the size for the books, customers, and transactions tables to be 1 GB each.

   ```
   java -cp dbgen-1.0-jar-with-dependencies.jar DBGen -p /mnt/dbgen -b 1 -c 1 -t 1
   ```

5. Create a new folder in the cluster's HDFS file system and copy the test data from the master node's local file system to HDFS using the following commands:

   ```
   hadoop fs -mkdir /data/
hadoop fs -put /mnt/dbgen/* /data/
hadoop fs -ls -h -R /data/
   ```

## Create and Populate Impala Tables

In this section, you create the Impala tables and fill them with test data.

### To create and populate the Impala tables with the test data

1. In the SSH window, launch the Impala shell prompt using the following command:

   ```
   impala-shell
   ```

2. Create and populate the books table with the test data by running the following command at the Impala shell prompt:
create EXTERNAL TABLE books( id BIGINT, isbn STRING, category STRING, publish_date TIMESTAMP, publisher STRING, price FLOAT ) ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' LOCATION '/data/books/';

3. Create and populate the customers table with the test data by running the following command at the Impala shell prompt:

create EXTERNAL TABLE customers( id BIGINT, name STRING, date_of_birth TIMESTAMP, gender STRING, state STRING, email STRING, phone STRING ) ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' LOCATION '/data/customers/';

4. Create and populate the transactions table with the test data by running the following command at the Impala shell prompt:

create EXTERNAL TABLE transactions( id BIGINT, customer_id BIGINT, book_id BIGINT, quantity INT, transaction_date TIMESTAMP ) ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' LOCATION '/data/transactions/';

Query Data in Impala

In this section, you perform queries on the data that you loaded in the Impala tables in the previous steps.

To perform various queries on the test data in the Impala tables

1. Perform a table scan through the entire customers table by running the following query at the Impala shell prompt:

```
SELECT COUNT(*)
FROM customers
WHERE name = 'Harrison SMITH';
```

2. Perform an aggregation query that scans a single table, groups the rows, and calculates the size of each group by running the following query at the Impala shell prompt:

```
SELECT category, count(*) cnt
FROM books
GROUP BY category
ORDER BY cnt DESC LIMIT 10;
```

3. Perform a query that joins the books table with the transactions table to determine the top ten book categories that have the maximum total revenue during a certain period of time by running the following query at the Impala shell prompt:

```
SELECT tmp.book_category, ROUND(tmp.revenue, 2) AS revenue
FROM ( 
    SELECT books.category AS book_category, SUM(books.price * transactions.quantity) AS revenue
    FROM books JOIN [SHUFFLE] transactions ON ( 
        transactions.book_id = books.id
        AND YEAR(transactions.transaction_date) BETWEEN 2008 AND 2010
    )
    GROUP BY books.category
) tmp
```
4. Perform a memory-intensive query that joins three tables by running the following query at the Impala shell prompt:

```
SELECT tmp.book_category, ROUND(tmp.revenue, 2) AS revenue
FROM (SELECT books.category AS book_category, SUM(books.price * transactions.quantity) AS revenue
FROM books
JOIN [SHUFFLE] transactions ON (transactions.book_id = books.id)
JOIN [SHUFFLE] customers ON (transactions.customer_id = customers.id
AND customers.state IN ('WA', 'CA', 'NY'))
GROUP BY books.category)
tmp
ORDER BY revenue DESC LIMIT 10;
```

**Important**
Now that you've completed the tutorial, you should terminate the cluster to ensure that your account does not accrue additional charges. For more information, see Terminate a Cluster (p. 472).

## Impala Examples Included on the Amazon EMR AMI

There are example data sets and queries included with the Impala installation on the Amazon EMR AMI.

**Topics**
- TPCDS (p. 305)
- Wikipedia (p. 306)

### TPCDS

The TPCDS example is derived from Cloudera's Impala demo virtual machine.

**To run the TPCDS example**

1. On the master node of the cluster, navigate to the examples directory and run the following scripts:

   ```shell
   cd ~/impala/examples/tpcds/
   ./tpcds-setup.sh
   ./tpcds-samplequery.sh
   ```

   The `tpcds-setup.sh` script loads data into HDFS and creates Hive tables. The `tpcds-samplequery.sh` script uses the following query to demonstrate how to use Impala to query data:

   ```sql
   select i_item_id,
           s_state,
   ```
avg(ss_quantity) agg1,
avg(ss_list_price) agg2,
avg(ss_coupon_amt) agg3,
avg(ss_sales_price) agg4
FROM store_sales
JOIN date_dim on (store_sales.ss_sold_date_sk = date_dim.d_date_sk)
JOIN item on (store_sales.ss_item_sk = item.i_item_sk)
JOIN customer_demographics on (store_sales.ss_cdemo_sk =
customer_demographics.cd_demo_sk)
JOIN store on (store_sales.ss_store_sk = store.s_store_sk)
where
cd_gender = 'M' and
cd_marital_status = 'S' and
cd_education_status = 'College' and
d_year = 2002 and
s_state in ('TN','SD', 'SD', 'SD', 'SD', 'SD')
group by
i_item_id,
s_state
order by
i_item_id,
s_state
limit 100;

2. Impala can create and manage Parquet tables. Parquet is a column-oriented binary file format intended to be highly efficient for scanning particular columns within a table. For more information, go to http://parquet.io/. After running the query, test the Parquet format by running the following script:

./tpcds-samplequery-parquet.sh

Wikipedia

The Wikipedia example uses the data and sample queries from the Shark example in GitHub. For more information, go to https://github.com/amplab/shark/wiki/Running-Shark-on-EC2.

To run the Wikipedia example

- On the master node of the cluster, navigate to the examples directory and run the following script:

  cd ~/impala/examples/wikipedia/
  ./wikipedia.sh

Alternatively, you can use this script instead:

  ./wikipedia-with-s3distcp.sh

The wikipedia.sh and wikipedia-with-s3distcp.sh scripts copy 42 GB of data from Amazon S3 to HDFS, create Hive tables, and use Impala to select data from the Hive tables. The difference between wikipedia.sh and wikipedia-with-s3distcp.sh is that wikipedia.sh uses Hadoop distcp to copy data from Amazon S3 to HDFS, but wikipedia-with-s3distcp.sh uses Amazon EMR S3DistCp for the same purpose.

The wikipedia-with-s3distcp.sh script contains the following code:

#!/bin/bash

  . /home/hadoop/impala/conf/impala.conf
Supported Impala Versions

The Impala version you can run depends on the version of the Amazon EMR AMI and the version of Hadoop that you are using. The table below shows the AMI versions that are compatible with different versions of Impala. We recommend using the latest available version of Impala to take advantage of performance enhancements and new functionality. For more information about the Amazon EMR AMIs and AMI versioning, see Choose an Amazon Machine Image (AMI) (p. 69). The Amazon EMR console does not support Impala versioning and always launches the latest version of Impala.

<table>
<thead>
<tr>
<th>Impala Version</th>
<th>AMI Version</th>
<th>Impala Version Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2.4</td>
<td>3.1.0 and later</td>
<td>Adds support for Impala 1.2.4.</td>
</tr>
<tr>
<td>1.2.1</td>
<td>3.0.2</td>
<td>3.0.3</td>
</tr>
</tbody>
</table>
Updates for Impala 1.2.4

The following updates are relevant for Impala 1.2.4:

- Performance improvements on metadata loading and synchronization on Impala startup. You can run queries before the loading is finished (and the query will wait until the metadata for that table is loaded if it's necessary).
- INVALIDATE METADATA was modified to accept argument, `table_name`, to load metadata for a specific table created by Hive. Conversely, if the table has been dropped by Hive, Impala will know that the table is gone.
- You can set the parallelism of catalogd's metadata loading using `--load_catalog_in_background` or `--num_metadata_loading_threads`.

**Note**
The following features were added with Impala 1.2.3 and 1.2.2 and are available with this release.

**Update for Impala 1.2.3**

- Notable bug fix: compatibility with Parquet files generated outside of Impala.

**Updates for Impala 1.2.2**

- Changes to Join order. Impala will automatically optimize a join query to minimize disk I/O and network traffic, instead of making a user order the join in a specific fashion.
- Use the STRAIGHT_JOIN operator to bypass Impala's automatic optimization of table order in a join query.
- Find table and column information with COMPUTE STATS.
- Use the CROSS JOIN operator in a SELECT statement for queries needing Cartesian products.
- New clauses for ALTER TABLE.
- LDAP authentication for JDBC and ODBC drivers.
- Numeric and conditional functions can return SMALLINT, FLOAT, and other smaller numerical types.


**Impala Memory Considerations**

Impala's memory requirements are decided by the type of query. There are no simple rules to determine the correlation between the maximum data size that a cluster can process and the aggregated memory size. The compression type, partitions, and the actual query (number of joins, result size, etc.) all play a role in the memory required. For example, your cluster may have only 60 GB of memory, but you can perform a single table scan and process 128 GB tables and larger. In contrast, when performing join operations, Impala may quickly use up the memory even though the aggregated table size is smaller than what's available. Therefore, to make full use of the available resources, it is extremely important to optimize the queries. You can optimize an Impala query for performance and to minimize resource consumption, and you can use the EXPLAIN statement to estimate the memory and other resources.
needed your query. In addition, for the best experience with Impala, we recommend using memory-optimized instances for your cluster. For more information, see Impala Performance Testing and Query Optimization (p. 310).

You can run multiple queries at one time on an Impala cluster on Amazon EMR. However, as each query is completed in-memory, ensure that you have the proper resources on your cluster to handle the number of concurrent queries you anticipate. In addition, you can set up a multi-tenant cluster with both Impala and MapReduce installed. You should be sure to allocate resources (memory, disk, and CPU) to each application using YARN on Hadoop 2.x. The resources allocated should be dependent on the needs for the jobs you plan to run on each application. For more information, go to http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html.

If you run out of memory, queries fail and the Impala daemon installed on the affected node shuts down. Amazon EMR then restarts the daemon on that node so that Impala will be ready to run another query. Your data in HDFS on the node remains available, because only the daemon running on the node shuts down, rather than the entire node itself. For ad hoc analysis with Impala, the query time can often be measured in seconds; therefore, if a query fails, you can discover the problem quickly and be able to submit a new query in quick succession.

### Using Impala with JDBC

While you can use ODBC drivers, Impala is also a great engine for third-party tools connected through JDBC. You can download and install the Impala client JDBC driver from http://elasticmapreduce.s3.amazonaws.com/libs/impala/1.2.1/impala-jdbc-1.2.1.zip.

From the client computer where you have your business intelligence tool installed, connect the JDBC driver to the master node of an Impala cluster using SSH or a VPN on port 21050. For more information, see Option 2, Part 1: Set Up an SSH Tunnel to the Master Node Using Dynamic Port Forwarding (p. 466).

### Accessing Impala Web User Interfaces

Impala 1.2.x and newer provides Web user interfaces for the statestore, impalad, and catalog daemons. These Web UIs are accessible through the following URLs and ports, respectively:

```
http://master-node-public-dns-name:25000
http://master-node-public-dns-name:25010
http://master-node-public-dns-name:25020
```

You can set up an SSH tunnel to the master node in Amazon EC2 to view the Web UIs on your local machine using the following example commands:

```
# ssh -i PERM_FILE -nTxNf -L 127.0.0.1:25000:master-node-public-dns-name:25000 \ 
hadoop@master-node-public-dns-name
# ssh -i PERM_FILE -nTxNf -L 127.0.0.1:25010:master-node-public-dns-name:25010 \ 
hadoop@master-node-public-dns-name
# ssh -i PERM_FILE -nTxNf -L 127.0.0.1:25020:master-node-public-dns-name:25020 \ 
hadoop@master-node-public-dns-name
```

For more information, see Option 2, Part 1: Set Up an SSH Tunnel to the Master Node Using Dynamic Port Forwarding (p. 466) and Option 2, Part 2: Configure Proxy Settings to View Websites Hosted on the Master Node (p. 468).
Impala-supported File and Compression Formats

Choosing the correct file type and compression is key for optimizing the performance of your Impala cluster. With Impala, you can query the following data types:

- Parquet
- Avro
- RCFile
- SequenceFile
- Unstructured text

In addition, Impala supports the following compression types:

- Snappy
- GZIP
- LZO (for text files only)
- Deflate (except Parquet and text)
- BZIP2 (except Parquet and text)

Depending on the file type and compression, you may need to use Hive to load data or create a table.

Impala SQL Dialect

Impala supports a subset of the standard SQL syntax, similar to HiveQL. For more information on the Impala SQL language, go to impala SQL Language Reference.

Impala User-Defined Functions

Impala supports user defined functions (UDFs) written in Java or C++. In addition, you can modify UDFs or user-defined aggregate functions created for Hive for use with Impala. For more information about Hive UDFs, go to https://cwiki.apache.org/confluence/display/Hive/LanguageManual+UDF.

Impala Performance Testing and Query Optimization

When using Impala, it is important to understand how your cluster’s memory resources limit the query types and dataset sizes you can process with them. Inspired by TPCDS and Berkeley's Big Data Benchmark, we implemented a workload generator which generates table files of specified sizes in text file format. We designed a spectrum of relational queries to test Impala's performance of whole table scans, aggregations and joins across different number of tables. We executed these queries against different input classes on clusters of different instance types. The performance data is compared against that of Hive's to help assess Impala's strength and limitations. Also, the methods used in these tests are the basis for the Launching and Querying Impala Clusters on Amazon EMR tutorial. For more information, see Tutorial: Launching and Querying Impala Clusters on Amazon EMR (p. 301).
Database Schema

The input data set consists of three tables as shown with the following table creation statements in Impala SQL dialect.

```
CREATE EXTERNAL TABLE books(
    id BIGINT,
    isbn STRING,
    category STRING,
    publish_date TIMESTAMP,
    publisher STRING,
    price FLOAT
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' LOCATION '/data/books/';

CREATE EXTERNAL TABLE customers(
    id BIGINT,
    name STRING,
    date_of_birth TIMESTAMP,
    gender STRING,
    state STRING,
    email STRING,
    phone STRING
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' LOCATION '/data/customers/';

CREATE EXTERNAL TABLE transactions(
    id BIGINT,
    customer_id BIGINT,
    book_id BIGINT,
    quantity INT,
    transaction_date TIMESTAMP
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' LOCATION '/data/transactions/';
```

Sample Data

All tables are populated with randomly generated data that resembles real-world values. You can generate data in the same method outlined in the Generate Sample Data section of the tutorial. For more information, see Generate Test Data (p. 303).

```
# head books/books
0|1-45812-668-3|EDUCATION|1986-06-14|Shinchosha|50.99
1|9-69091-140-1|BODY-MIND-SPIRIT|1983-07-29|Lefebvre-Sarrut|91.99
2|7-37425-809-9|TRANSPORTATION|1996-07-08|Mondadori|54.99
3|8-23483-356-2|FAMILY-RELATIONSHIPS|2002-08-20|Lefebvre-Sarrut|172.99
4|3-58984-308-3|POETRY|1974-06-13|EKSMO|155.99
5|2-34120-729-8|TRAVEL|2004-06-30|Cengage|190.99
6|0-38870-277-1|HISTORY|2012-05-01|Olma Media Group|82.99
7|8-74275-772-8|LAW|2012-05-01|Holtzbrinck|112.99
8|4-41109-927-4|LITERARY-CRITICISM|1986-04-06|Olma Media Group|82.99
9|8-45276-479-4|TRAVEL|1998-07-04|Lefebvre-Sarrut|180.99

# head customers/customers
0|Bailey RUIZ|1947-12-19|M|CT|bailey.ruiz.1947@hotmail.com|114-925-4866
1|Taylor BUTLER|1938-07-30|M|IL|taylor.butler.1938@yahoo.com|517-158-1597
2|Henry BROOKS|1956-12-27|M|IN|henry.brooks.1956@yahoo.com|221-653-3887
3|Kaitlyn WRIGHT|1988-11-20|F|NE|kaitlyn.wright.1988@hotmail.com|645-726-8901
```

311
Table Size

The following table shows the row count for each table (in millions of rows). The GB value indicates the size of the text file of each table. Within an input class, the books, customers, and transactions tables always have the same size.

<table>
<thead>
<tr>
<th>Input Class (size of each table)</th>
<th>Books table (Million Rows)</th>
<th>Customers table (Million Rows)</th>
<th>Transactions table (Million Rows)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4GB</td>
<td>63</td>
<td>53</td>
<td>87</td>
</tr>
<tr>
<td>8GB</td>
<td>125</td>
<td>106</td>
<td>171</td>
</tr>
<tr>
<td>16GB</td>
<td>249</td>
<td>210</td>
<td>334</td>
</tr>
<tr>
<td>32GB</td>
<td>497</td>
<td>419</td>
<td>659</td>
</tr>
<tr>
<td>64GB</td>
<td>991</td>
<td>837</td>
<td>1304</td>
</tr>
<tr>
<td>128GB</td>
<td>1967</td>
<td>1664</td>
<td>2538</td>
</tr>
<tr>
<td>256GB</td>
<td>3919</td>
<td>3316</td>
<td>5000</td>
</tr>
</tbody>
</table>

Queries

We used four different query types in our performance testing:

Q1: Scan Query

```
SELECT COUNT(*)
FROM customers
WHERE name = 'Harrison SMITH';
```

This query performs a table scan through the entire table. With this query, we mainly test:

- Impala's read throughput compared to that of Hive.
- With a given aggregated memory size, is there a limit on input size when performing a table scan, and if yes, what is the maximum input size that Impala can handle?
Q2: Aggregation Query

```
SELECT category, count(*) cnt
FROM books
GROUP BY category
ORDER BY cnt DESC LIMIT 10;
```

The aggregation query scans a single table, groups the rows, and calculates the size of each group.

Q3: Join Query between Two Tables

```
SELECT tmp.book_category, ROUND(tmp.revenue, 2) AS revenue
FROM (  
  SELECT books.category AS book_category, SUM(books.price * transactions.quantity) AS revenue
  FROM books JOIN [SHUFFLE] transactions ON (  
    transactions.book_id = books.id
    AND YEAR(transactions.transaction_date) BETWEEN 2008 AND 2010
  )
  GROUP BY books.category
) tmp
ORDER BY revenue DESC LIMIT 10;
```

This query joins the books table with the transactions table to find the top 10 book categories with the maximum total revenue during a certain period of time. In this experiment, we test Impala’s performance on a join operation and compare these results with Hive.

Q4: Join Query between Three Tables

```
SELECT tmp.book_category, ROUND(tmp.revenue, 2) AS revenue
FROM (  
  SELECT books.category AS book_category, SUM(books.price * transactions.quantity) AS revenue
  FROM books  
  JOIN [SHUFFLE] transactions ON (  
    transactions.book_id = books.id
  )  
  JOIN [SHUFFLE] customers ON (  
    transactions.customer_id = customers.id
    AND customers.state IN ('WA', 'CA', 'NY')
  )
  GROUP BY books.category
) tmp
ORDER BY revenue DESC LIMIT 10;
```

This fourth query joins three tables, and is the most memory intensive query in our performance testing.

Performance Test Results

The first set of experimental results were obtained on a cluster of four m1.xlarge instances with Amazon EMR Hadoop 2.2.0 and Impala 1.1.1 installed. According to the hardware specification shown below, an aggregated memory of 60 GB is available on the cluster.

<table>
<thead>
<tr>
<th>Instance Type</th>
<th>Processor Architecture</th>
<th>vCPUs</th>
<th>ECU</th>
<th>Memory (GiB)</th>
<th>Instance Storage (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1.xlarge</td>
<td>64-bit</td>
<td>4</td>
<td>8</td>
<td>15</td>
<td>4 x 420</td>
</tr>
</tbody>
</table>
We compared the query performance of Impala and Hive in terms of the query execution time and show the results in the charts below. In these charts, the y axis shows the average execution time measured using the time command from four trials. The missing data indicates Impala failed due to out-of-memory issue, and we did not test Hive against these failed Impala queries.

From these figures, we observed that at smaller scales (in this experiment, 16 GB and lower), Impala is much faster than Hive due to the absence of the MapReduce framework overhead. Nonetheless, when the input data set is large enough such that the framework overhead is negligible compared to overall query time, Impala is only about 3 to 10 times faster.

The second experimental environment was a cluster of 4 m2.4xlarge instances with an AMI with Hadoop 2.2.0 and Impala 1.1.1. The aggregated memory on this cluster is 274 GB. Detailed hardware
specifications and experimental results are shown below. Like our first set of tests, missing data indicates Impala failures caused by out-of-memory issues, and we declined to run Hive tests for these queries.

<table>
<thead>
<tr>
<th>Instance Type</th>
<th>Processor Architecture</th>
<th>vCPUs</th>
<th>ECU</th>
<th>Memory (GiB)</th>
<th>Instance Storage (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>m2.4xlarge</td>
<td>64-bit</td>
<td>8</td>
<td>26</td>
<td>68.4</td>
<td>2 x 840</td>
</tr>
</tbody>
</table>
Optimizing Queries

Impala's memory requirement is determined by query type. There are no simple and generic rules to determine the correlation between the maximum data size that a cluster can process with its aggregated memory size.

Impala does not load entire tables into memory, so the amount of available memory doesn't limit the table size that it can handle. Impala builds hash tables in memory, such as the right-hand side table of a join or the result set of an aggregation. In addition, Impala uses memory as I/O buffers, where the number of processor cores on the cluster and the speed of the scanners determine the amount of buffering that is necessary in order to keep all cores busy. For example, a simple `SELECT count(*)` from statement only uses I/O buffer memory.

For example, our m1.xlarge cluster in part 1 of our experiment only had 60 GB of memory, but when we performed single table scan, we were able to process tables of 128 GB and above. Because Impala didn't need to cache the entire result set of the query, it streamed the result set back to the client. In contrast, when performing a join operation, Impala may quickly use up a cluster's memory even if the aggregated table size is smaller than the aggregated amount of memory. To make full use of the available resources, it is extremely important to optimize your queries. In this section, we take Q3 as an example to illustrate some of the optimization techniques you can try when an out-of-memory error happens.

Shown below is the typical error message you receive when the impalad process on a particular data node crashed due to a memory issue. To confirm the out-of-memory issue, you can simply log on to the data nodes and use the top command to monitor the memory usage (%MEM). Note that even for the same query, the out-of-memory error may not always happen on the same node. Also, no action is needed to recover from an out-of-memory error, because impalad is restarted automatically.

```
Backend 6: Couldn't open transport for ip-10-139-0-87.ec2.internal:22000 (connect() failed: Connection refused)
```

Simple query optimization techniques might be effective in allowing your queries to use less memory, allowing you to sidestep an out-of-memory error. For example, the first version of Q3 (pre-optimization) is shown below, where the transactions table is on the left side of JOIN while the books table is on the right:

```
SELECT tmp.book_category, ROUND(tmp.revenue, 2) AS revenue
FROM (SELECT books.category AS book_category, SUM(books.price * transactions.quantity) AS revenue
```

---

**Table Size: 256GB**

<table>
<thead>
<tr>
<th></th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Time (second)</td>
<td>1000</td>
<td>2000</td>
<td>3000</td>
<td>4000</td>
</tr>
</tbody>
</table>

---

316
FROM transactions JOIN books ON (transactions.book_id = books.id AND YEAR(transactions.transaction_date) BETWEEN 2008 AND 2010) GROUP BY books.category;}

ORDER BY revenue DESC LIMIT 10;

This query only worked for the 4 GB input class and failed for 8 GB and above due to the out-of-memory error. To understand why, you must consider how Impala executes queries. In preparation for the join, Impala builds a hash table from the books table that contains only the columns category, price, and id. Nothing of the transactions table is cached in memory. However, because Impala broadcasts the right-side table in this example, the books table is replicated to all the nodes that require the books table for joining. In versions of Impala newer than 1.2.1, Impala makes a cost-based decision between broadcast and partitioned join based on table statistics. We simply swapped these two tables in the JOIN statement to get the second version of Q3 shown below:

SELECT tmp.book_category, ROUND(tmp.revenue, 2) AS revenue
FROM (SELECT books.category AS book_category, SUM(books.price * transactions.quantity) AS revenue
FROM books JOIN transactions ON (transactions.book_id = books.id AND YEAR(transactions.transaction_date) BETWEEN 2008 AND 2010)
GROUP BY books.category) tmp
ORDER BY revenue DESC LIMIT 10;

The second version of Q3 is more efficient, because only a part of the transactions table is broadcast instead of the entire books table. Nonetheless, when we scaled up to the 32 GB input class, even the second version of Q3 started to fail due to memory constraints. To further optimize the query, we added a hint to force Impala to use the “partitioned join,” which creates the final version of Q3 as shown above in the Queries section. With all the optimizations, we eventually managed to execute Q3 successfully for input classes up to 64 GB, giving us a 16x more memory-efficient query than the first version. There are many other ways to optimize queries for Impala, and we see these methods as a good way to get the best performance from your hardware and avoid out-of-memory errors.
Apache Pig

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Amazon EMR supports Apache Pig, a programming framework you can use to analyze and transform large data sets. For more information about Pig, go to http://pig.apache.org/. Amazon EMR supports several versions of Pig.

Pig is an open-source, Apache library that runs on top of Hadoop. The library takes SQL-like commands written in a language called Pig Latin and converts those commands into MapReduce jobs. You do not have to write complex MapReduce code using a lower level computer language, such as Java.

You can execute Pig commands interactively or in batch mode. To use Pig interactively, create an SSH connection to the master node and submit commands using the Grunt shell. To use Pig in batch mode, write your Pig scripts, upload them to Amazon S3, and submit them as cluster steps. For more information on submitting work to a cluster, see Submit Work to a Cluster (p. 493).

Topics
- Supported Pig Versions (p. 318)
- Interactive and Batch Pig Clusters (p. 322)
- Submit Pig Work (p. 322)
- Call User Defined Functions from Pig (p. 323)

Supported Pig Versions

The Pig version you can add to your cluster depends on the version of the Amazon EMR (Amazon EMR) AMI and the version of Hadoop you are using. The table below shows which AMI versions and versions of Hadoop are compatible with the different versions of Pig. We recommend using the latest available version of Pig to take advantage of performance enhancements and new functionality. For more information about the Amazon EMR AMIs and AMI versioning, see Choose an Amazon Machine Image (AMI) (p. 69).

If you choose to install Pig on your cluster using the console or the AWS CLI, the AMI you specify determines the version of Pig installed. By default, Pig is installed on your cluster when you use the console, but you can remove it during cluster creation. Pig is also installed by default when you use the AWS CLI unless you use the --applications parameter to identify which applications you want on your cluster. The AWS CLI does not support Pig versioning.

When you use the API to install Pig, the default version is used unless you specify --pig-versions as an argument to the step that loads Pig onto the cluster during the call to RunJobFlow.

<table>
<thead>
<tr>
<th>Pig Version</th>
<th>AMI Version</th>
<th>Configuration Parameters</th>
<th>Pig Version Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.12.0</td>
<td>3.1.0 and later</td>
<td>--ami-version 3.1</td>
<td>Adds support for the following:</td>
</tr>
<tr>
<td>Release Notes</td>
<td>3.1.0 and later</td>
<td>--ami-version 3.2</td>
<td>• Streaming UDFs without JVM implementations</td>
</tr>
<tr>
<td>Documentation</td>
<td>3.1.0 and later</td>
<td>--ami-version 3.3</td>
<td></td>
</tr>
</tbody>
</table>

318
<table>
<thead>
<tr>
<th>Pig Version</th>
<th>AMI Version</th>
<th>Configuration Parameters</th>
<th>Pig Version Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.11.1.1</td>
<td>2.2 and later</td>
<td>--pig-versions 0.11.1.1  --ami-version 2.2</td>
<td>Improves performance of LOAD command with PigStorage if input resides in Amazon S3.</td>
</tr>
<tr>
<td>0.11.1</td>
<td>2.2 and later</td>
<td>--pig-versions 0.11.1  --ami-version 2.2</td>
<td>Adds support for JDK 7, Hadoop 2, Groovy User Defined Functions, SchemaTuple optimization, new operators, and more. For more information, see Pig 0.11.1 Change Log.</td>
</tr>
<tr>
<td>0.9.2.2</td>
<td>2.2 and later</td>
<td>--pig-versions 0.9.2.2  --ami-version 2.2</td>
<td>Adds support for Hadoop 1.0.3.</td>
</tr>
<tr>
<td>0.9.2.1</td>
<td>2.2 and later</td>
<td>--pig-versions 0.9.2.1  --ami-version 2.2</td>
<td>Adds support for MapR. For more information, see Using the MapR Distribution for Hadoop (p. 177).</td>
</tr>
<tr>
<td>0.9.2</td>
<td>2.2 and later</td>
<td>--pig-versions 0.9.2  --ami-version 2.2</td>
<td>Includes several performance improvements and bug fixes. For complete information about the changes for Pig 0.9.2, go to the Pig 0.9.2 Change Log.</td>
</tr>
<tr>
<td>0.9.1</td>
<td>2.0</td>
<td>--pig-versions 0.9.1  --ami-version 2.0</td>
<td></td>
</tr>
</tbody>
</table>
Pig Version Details

Amazon EMR supports certain Pig releases that might have additional Amazon EMR patches applied. You can configure which version of Pig to run on Amazon EMR clusters. For more information about how to do this, see Apache Pig (p. 318). The following sections describe different Pig versions and the patches applied to the versions loaded on Amazon EMR.

Pig Patches

This section describes the custom patches applied to Pig versions available with Amazon EMR.

Pig 0.11.1.1 Patches

The Amazon EMR version of Pig 0.11.1.1 is a maintenance release that improves performance of LOAD command with PigStorage if the input resides in Amazon S3.

Pig 0.11.1 Patches

The Amazon EMR version of Pig 0.11.1 contains all the updates provided by the Apache Software Foundation and the cumulative Amazon EMR patches from Pig version 0.9.2.2. However, there are no new Amazon EMR-specific patches in Pig 0.11.1.

Pig 0.9.2 Patches

Apache Pig 0.9.2 is a maintenance release of Pig. The Amazon EMR team has applied the following patches to the Amazon EMR version of Pig 0.9.2.

<table>
<thead>
<tr>
<th>Patch</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIG-1429</td>
<td>Add the Boolean data type to Pig as a first class data type. For more information, go to <a href="https://issues.apache.org/jira/browse/PIG-1429">https://issues.apache.org/jira/browse/PIG-1429</a>. Status: Committed Fixed in Apache Pig Version: 0.10</td>
</tr>
<tr>
<td>PIG-1824</td>
<td>Support import modules in Jython UDF. For more information, go to <a href="https://issues.apache.org/jira/browse/PIG-1824">https://issues.apache.org/jira/browse/PIG-1824</a>. Status: Committed Fixed in Apache Pig Version: 0.10</td>
</tr>
<tr>
<td>PIG-2010</td>
<td>Bundle registered JARs on the distributed cache. For more information, go to <a href="https://issues.apache.org/jira/browse/PIG-2010">https://issues.apache.org/jira/browse/PIG-2010</a>.</td>
</tr>
</tbody>
</table>
### Pig 0.9.1 Patches

The Amazon EMR team has applied the following patches to the Amazon EMR version of Pig 0.9.1.

<table>
<thead>
<tr>
<th>Patch</th>
<th>Description</th>
<th>Status</th>
<th>Fixed in Apache Pig Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support JAR files and Pig scripts in dfs</td>
<td>Add support for running scripts and registering JAR files stored in HDFS, Amazon S3, or other distributed file systems. For more information, go to <a href="https://issues.apache.org/jira/browse/PIG-1505">https://issues.apache.org/jira/browse/PIG-1505</a>.</td>
<td>Committed</td>
<td>0.8.0</td>
</tr>
<tr>
<td>Support multiple file systems in Pig</td>
<td>Add support for Pig scripts to read data from one file system and write it to another. For more information, go to <a href="https://issues.apache.org/jira/browse/PIG-1564">https://issues.apache.org/jira/browse/PIG-1564</a>.</td>
<td>Not Committed</td>
<td>n/a</td>
</tr>
<tr>
<td>Add Piggybank datetime and string UDFs</td>
<td>Add datetime and string UDFs to support custom Pig scripts. For more information, go to <a href="https://issues.apache.org/jira/browse/PIG-1565">https://issues.apache.org/jira/browse/PIG-1565</a>.</td>
<td>Not Committed</td>
<td>n/a</td>
</tr>
</tbody>
</table>

### Additional Pig Functions

The Amazon EMR development team has created additional Pig functions that simplify string manipulation and make it easier to format date-time information. These are available at [http://aws.amazon.com/code/2730](http://aws.amazon.com/code/2730).
Interactive and Batch Pig Clusters

Amazon EMR enables you to run Pig scripts in two modes:

- Interactive
- Batch

When you launch a long-running cluster using the console or the AWS CLI, you can ssh into the master node as the Hadoop user and use the Grunt shell to develop and run your Pig scripts interactively. Using Pig interactively enables you to revise the Pig script more easily than batch mode. After you successfully revise the Pig script in interactive mode, you can upload the script to Amazon S3 and use batch mode to run the script in production. You can also submit Pig commands interactively on a running cluster to analyze and transform data as needed.

In batch mode, you upload your Pig script to Amazon S3, and then submit the work to the cluster as a step. Pig steps can be submitted to a long-running cluster or a transient cluster. For more information on submitting work to a cluster, see Submit Work to a Cluster (p. 493).

Submit Pig Work

This section demonstrates submitting Pig work to an Amazon EMR cluster. The examples that follow are based on the Amazon EMR sample: Apache Log Analysis using Pig. The sample evaluates Apache log files and then generates a report containing the total bytes transferred, a list of the top 50 IP addresses, a list of the top 50 external referrers, and the top 50 search terms using Bing and Google. The Pig script is located in the Amazon S3 bucket s3://elasticmapreduce/samples/pig-apache/do-reports2.pig. Input data is located in the Amazon S3 bucket s3://elasticmapreduce/samples/pig-apache/input. The output is saved to an Amazon S3 bucket.

Submit Pig Work Using the Amazon EMR Console

This example describes how to use the Amazon EMR console to add a Pig step to a cluster.

To submit a Pig step

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. In the Cluster List, click the name of your cluster.
3. Scroll to the Steps section and expand it, then click Add step.
4. In the Add Step dialog:
   - For Step type, choose Pig program.
   - For Name, accept the default name (Pig program) or type a new name.
   - For Script S3 location, type the location of the Pig script. For example: s3://elasticmapreduce/samples/pig-apache/do-reports2.pig.
   - For Input S3 location, type the location of the input data. For example: s3://elasticmapreduce/samples/pig-apache/input.
   - For Output S3 location, type or browse to the name of your Amazon S3 output bucket.
   - For Arguments, leave the field blank.
   - For Action on failure, accept the default option (Continue).
5. Click Add. The step appears in the console with a status of Pending.
6. The status of the step changes from Pending to Running to Completed as the step runs. To update the status, click the Refresh icon above the Actions column.
Submit Pig Work Using the AWS CLI

To submit a Pig step using the AWS CLI

When you launch a cluster using the AWS CLI, use the --applications parameter to install Pig. To submit a Pig step, use the --steps parameter.

- To launch a cluster with Pig installed and to submit a Pig step, type the following command, replace myKey with the name of your EC2 key pair, and replace mybucket with the name of your Amazon S3 bucket.

```
aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applications Name=Hue Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey --instance-type m3.xlarge --instance-count 3 --steps Type=PIG,Name="Pig Program",ActionOnFailure=CONTINUE,Args=[-f,s3://elasticmapreduce/samples/pig-apache/do-reports2.pig,-p,INPUT=s3://elasticmapreduce/samples/pig-apache/input,-p,OUTPUT=s3://mybucket/pig-apache/output]
```

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

**Note**

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

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

Call User Defined Functions from Pig

Pig provides the ability to call user defined functions (UDFs) from within Pig scripts. You can do this to implement custom processing to use in your Pig scripts. The languages currently supported are Java, Python/Jython, and JavaScript. (Though JavaScript support is still experimental.)

The following sections describe how to register your functions with Pig so you can call them either from the Pig shell or from within Pig scripts. For more information about using UDFs with Pig, go to [http://pig.apache.org/docs/r0.14.0/udf.html](http://pig.apache.org/docs/r0.14.0/udf.html).

Call JAR files from Pig

You can use custom JAR files with Pig using the REGISTER command in your Pig script. The JAR file is local or a remote file system such as Amazon S3. When the Pig script runs, Amazon EMR downloads the JAR file automatically to the master node and then uploads the JAR file to the Hadoop distributed cache. In this way, the JAR file is automatically used as necessary by all instances in the cluster.

**To use JAR files with Pig**

1. Upload your custom JAR file into Amazon S3.
2. Use the REGISTER command in your Pig script to specify the bucket on Amazon S3 of the custom JAR file.

```
REGISTER s3://mybucket/path/mycustomjar.jar;
```
Call Python/Jython Scripts from Pig

You can register Python scripts with Pig and then call functions in those scripts from the Pig shell or in a Pig script. You do this by specifying the location of the script with the register keyword.

Because Pig is written in Java, it uses the Jython script engine to parse Python scripts. For more information about Jython, go to http://www.jython.org/.

To call a Python/Jython script from Pig

1. Write a Python script and upload the script to a location in Amazon S3. This should be a bucket owned by the same account that creates the Pig cluster, or that has permissions set so the account that created the cluster can access it. In this example, the script is uploaded to s3://mybucket/pig/python.
2. Start a pig cluster. If you'll be accessing Pig from the Grunt shell, run an interactive cluster. If you're running Pig commands from a script, start a scripted Pig cluster. In this example, we'll start an interactive cluster. For more information about how to create a Pig cluster, see Submit Pig Work (p. 322).
3. Because we've launched an interactive cluster, we'll now SSH into the master node where we can run the Grunt shell. For more information about how to SSH into the master node, see SSH into the Master Node.
4. Run the Grunt shell for Pig by typing pig at the command line.

   
   pig

5. Register the Jython library and your Python script with Pig using the register keyword at the Grunt command prompt, as shown in the following, where you would specify the location of your script in Amazon S3.

   grunt> register 'lib/jython.jar';
   grunt> register 's3://mybucket/pig/python/myscript.py' using jython as myfunctions;

6. Load the input data. The following example loads input from an Amazon S3 location.

   grunt> input = load 's3://mybucket/input/data.txt' using TextLoader as (line:chararray);

7. You can now call functions in your script from within Pig by referencing them using myfunctions.

   grunt> output=foreach input generate myfunctions.myfunction($1);
Apache HBase

HBase is an open source, non-relational, distributed database developed as part of Apache Software Foundation's Hadoop project. HBase runs on top of Hadoop Distributed File System (HDFS) to provide non-relational database capabilities for the Hadoop ecosystem. HBase works seamlessly with Hadoop, sharing its file system and serving as a direct input and output to the MapReduce framework and execution engine. HBase also integrates with Apache Hive, enabling SQL-like queries over HBase tables, joins with Hive-based tables, and support for Java Database Connectivity (JDBC). For more information about HBase, see Apache HBase and HBase documentation on the Apache website.

With HBase on Amazon EMR, you can also back up your HBase data directly to Amazon S3, and restore from a previously created backup when launching an HBase cluster.

HBase Release Information for This Release of Amazon EMR

What Can I Do with HBase?

You can use HBase for random, repeated access to and modification of large volumes of data. HBase provides low-latency lookups and range scans, along with efficient updates and deletions of individual records.

Here are several HBase use cases to consider:

Reference data for Hadoop analytics

With its direct integration with Hadoop and Hive and rapid access to stored data, HBase can be used to store reference data used by multiple Hadoop tasks or across multiple Hadoop clusters. This data can be stored directly on the cluster running Hadoop tasks or on a separate cluster. Types of analytics include analytics requiring fast access to demographic data, IP address geolocation lookup tables, and product dimensional data.

Real-time log ingestion and batch log analytics

HBase's high write throughput, optimization for sequential data, and efficient storage of sparse data make it a great solution for real-time ingestion of log data. At the same time, its integration with Hadoop and optimization for sequential reads and scans makes it equally suited for batch analysis of that log data after ingestion. Common use cases include ingestion and analysis of application logs, clickstream data, and in game usage data.

Store for high frequency counters and summary data

Counter increments aren't just database writes, they're read-modify-writes, so they're a very expensive operation for a relational database. However, because HBase is a nonrelational, distributed database, it supports very high update rates and, given its consistent reads and writes, provides immediate access to
that updated data. In addition, if you want to run more complex aggregations on the data (such as max-mins, averages, and group-bys), you can run Hadoop jobs directly and feed the aggregated results back into HBase.

**Supported HBase Versions**

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

<table>
<thead>
<tr>
<th>HBase Version</th>
<th>AMI Version</th>
<th>AWS CLI Configuration Parameters</th>
<th>HBase Version Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.94.18</td>
<td>3.1.0 and later</td>
<td>--ami-version 3.1</td>
<td>• Bug fixes and enhancements.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>--ami-version 3.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>--ami-version 3.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>--applications Name=HBase</td>
<td></td>
</tr>
<tr>
<td>0.94.7</td>
<td>3.0-3.0.4</td>
<td>--ami-version 3.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>--applications Name=HBase</td>
<td></td>
</tr>
<tr>
<td>0.92</td>
<td>2.2 and later</td>
<td>--ami-version 2.2 or later</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>--applications Name=HBase</td>
<td></td>
</tr>
</tbody>
</table>

**HBase Cluster Prerequisites**

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

An Amazon EMR cluster should meet the following requirements in order to run HBase:

- The AWS CLI (optional)—To interact with HBase using the command line, download and install the latest version of the AWS CLI. For more information, see Installing the AWS Command Line Interface in the AWS Command Line Interface User Guide.
- At least two instances (optional)—The cluster's master node runs the HBase master server and Zookeeper, and slave nodes run the HBase region servers. For best performance, HBase clusters should run on at least two EC2 instances, but you can run HBase on a single node for evaluation purposes.
- Long-running cluster—HBase only runs on long-running clusters. By default, the CLI and Amazon EMR console create long-running clusters.
- An Amazon EC2 key pair set (recommended)—To use the Secure Shell (SSH) network protocol to connect with the master node and run HBase shell commands, you must use an Amazon EC2 key pair when you create the cluster.
- The correct AMI and Hadoop versions—HBase clusters are currently supported only on Hadoop 20.205 or later.
• Ganglia (optional)—To monitor HBase performance metrics, install Ganglia when you create the cluster.

• An Amazon S3 bucket for logs (optional)—The logs for HBase are available on the master node. If you’d like these logs copied to Amazon S3, specify an S3 bucket to receive log files when you create the cluster.

Install HBase on an Amazon EMR Cluster

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

When you launch HBase on Amazon EMR, you get the benefits of running in the AWS cloud—easy scaling, low cost, pay only for what you use, and ease of use. The Amazon EMR team has tuned HBase to run optimally on AWS. For more information about HBase and running it on Amazon EMR, see Apache HBase (p. 325).

The following procedure shows how to launch an HBase cluster with the default settings. If your application needs custom settings, you can configure HBase as described in Configure HBase (p. 339).

**Note**

HBase configuration can only be done at launch time.

**To launch a cluster and install HBase using the console**

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. Choose Create cluster.
3. On the Create Cluster page, in the Cluster Configuration section, verify the fields according to the following table.

<table>
<thead>
<tr>
<th>Field</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster name</td>
<td>Enter a descriptive name for your cluster or leave the default name &quot;My cluster.&quot;</td>
</tr>
<tr>
<td></td>
<td>The name is optional, and does not need to be unique.</td>
</tr>
<tr>
<td>Termination protection</td>
<td>Leave the default option selected: Yes.</td>
</tr>
<tr>
<td></td>
<td>Enabling termination protection ensures that the cluster does not shut down due to accident or error. For more information, see Managing Cluster Termination (p. 474). Typically, you set this value to Yes when developing an application (so you can debug errors that would have otherwise terminated the cluster), to protect long-running clusters, or to preserve data.</td>
</tr>
<tr>
<td>Logging</td>
<td>This determines whether Amazon EMR captures detailed log data to Amazon S3.</td>
</tr>
<tr>
<td></td>
<td>For more information, see View Log Files (p. 422).</td>
</tr>
<tr>
<td>Log folder S3 location</td>
<td>Type or browse to an Amazon S3 path to store your debug logs if you enabled logging in the previous field. You may also allow the console to generate an Amazon S3 path for you. If the log folder does not exist, the Amazon EMR console creates it.</td>
</tr>
</tbody>
</table>
When Amazon S3 log archiving is enabled, Amazon EMR copies the log files from the EC2 instances in the cluster to Amazon S3. This prevents the log files from being lost when the cluster ends and the EC2 instances hosting the cluster are terminated. These logs are useful for troubleshooting purposes.

For more information, see View Log Files (p. 422).

4. In the **Software Configuration** section, verify the fields according to the following table.

<table>
<thead>
<tr>
<th>Field</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hadoop distribution</td>
<td>Choose Amazon. This determines which distribution of Hadoop to run on your cluster. You can choose to run the Amazon distribution of Hadoop or one of several MapR distributions. For more information, see Using the MapR Distribution for Hadoop (p. 177).</td>
</tr>
<tr>
<td>AMI version</td>
<td>Choose the latest Hadoop 2.x AMI or the latest Hadoop 1.x AMI from the list. The AMI you choose determines the specific version of Hadoop and other applications such as Hive or Pig to run on your cluster. For more information, see Choose an Amazon Machine Image (AMI) (p. 69).</td>
</tr>
</tbody>
</table>

5. For **Additional Applications**, choose HBase and **Configure and add**.

6. In the **Add Application** section, indicate whether to pre-load the HBase cluster with data stored in Amazon S3 and whether you want to schedule regular backups of your HBase cluster, and then choose **Add**. Use the following table for guidance on making your selections. For more information about backing up and restoring HBase data, see Back Up and Restore HBase (p. 334).

<table>
<thead>
<tr>
<th>Field</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restore from backup</td>
<td>Specify whether to pre-load the HBase cluster with data stored in Amazon S3.</td>
</tr>
<tr>
<td>Backup location</td>
<td>Specify the URI where the backup from which to restore resides in Amazon S3.</td>
</tr>
<tr>
<td>Backup version</td>
<td>Optionally, specify the version name of the backup at Backup Location to use. If you leave this field blank, Amazon EMR uses the latest backup at Backup Location to populate the new HBase cluster.</td>
</tr>
<tr>
<td>Schedule Regular Backups</td>
<td>Specify whether to schedule automatic incremental backups. The first backup is a full backup to create a baseline for future incremental backups.</td>
</tr>
<tr>
<td>Consistent backup</td>
<td>Specify whether the backups should be consistent. A consistent backup is one which pauses write operations during the initial backup stage, synchronization across nodes. Any write operations thus paused are placed in a queue and resume when synchronization completes.</td>
</tr>
<tr>
<td>Backup frequency</td>
<td>The number of days/hours/minutes between scheduled backups.</td>
</tr>
</tbody>
</table>
7. In the **Hardware Configuration** section, verify the fields according to the following table.

**Note**

Twenty is the default maximum number of nodes per AWS account. For example, if you have two clusters, the total number of nodes running for both clusters must be 20 or less. Exceeding this limit results in cluster failures. If you need more than 20 nodes, you must submit a request to increase your Amazon EC2 instance limit. Ensure that your requested limit increase includes sufficient capacity for any temporary, unplanned increases in your needs. For more information, see the [Request to Increase Amazon EC2 Instance Limit Form](#).

<table>
<thead>
<tr>
<th>Field</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Backup location</strong></td>
<td>The Amazon S3 URI where backups are stored. The backup location for each HBase cluster should be different to ensure that differential backups stay correct.</td>
</tr>
<tr>
<td><strong>Backup start time</strong></td>
<td>Specify when the first backup should occur. You can set this to <em>now</em>, which causes the first backup to start as soon as the cluster is running, or enter a date and time in <strong>ISO format</strong>. For example, 2012-06-15T20:00Z would set the start time to June 15, 2012 at 8pm UTC.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Network</strong></td>
<td>Choose the default VPC. For more information, see <a href="#">Your Default VPC and Subnets</a> in the <em>Amazon VPC User Guide</em>. Optionally, if you have created additional VPCs, you can choose your preferred VPC subnet identifier from the list to launch the cluster in that Amazon VPC. For more information, see <a href="#">Plan and Configure Networking</a>.</td>
</tr>
<tr>
<td><strong>EC2 Availability Zone</strong></td>
<td>Choose <strong>No preference</strong>. Optionally, you can launch the cluster in a specific Availability Zone. For more information, see <a href="#">Regions and Availability Zones</a> in the <em>Amazon EC2 User Guide for Linux Instances</em>.</td>
</tr>
<tr>
<td><strong>Master</strong></td>
<td>Accept the default instance type. The master node assigns Hadoop tasks to core and task nodes, and monitors their status. There is always one master node in each cluster. This specifies the EC2 instance type to use for the master node. The default instance type is m1.medium for Hadoop 2.x. This instance type is suitable for testing, development, and light workloads. For more information about instance types supported by Amazon EMR, see <a href="#">Virtual Server Configurations</a>. For more information about Amazon EMR instance groups, see <a href="#">Create a Cluster with Instance Fleets or Uniform Instance Groups</a>. For information about mapping legacy clusters to instance groups, see <a href="#">Mapping Legacy Clusters to Instance Groups</a>.</td>
</tr>
<tr>
<td><strong>Request Spot Instances</strong></td>
<td>Leave this box unchecked. This specifies whether to run master nodes on Spot Instances. For more information, see <a href="#">When Should You Use Spot Instances?</a>.</td>
</tr>
<tr>
<td><strong>Core</strong></td>
<td>Accept the default instance type.</td>
</tr>
</tbody>
</table>
## Install HBase on an Amazon EMR Cluster

A core node is an EC2 instance that runs Hadoop map and reduce tasks and stores data using the Hadoop Distributed File System (HDFS). Core nodes are managed by the master node.

This specifies the EC2 instance types to use as core nodes.

The default instance type is m1.medium for Hadoop 2.x. This instance type is suitable for testing, development, and light workloads.

For more information about instance types supported by Amazon EMR, see [Virtual Server Configurations](#). For more information about Amazon EMR instance groups, see [Create a Cluster with Instance Fleets or Uniform Instance Groups](#). For information about mapping legacy clusters to instance groups, see [Mapping Legacy Clusters to Instance Groups](#).

<table>
<thead>
<tr>
<th>Field</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Count</strong></td>
<td>Choose 2.</td>
</tr>
<tr>
<td>Request Spot Instances</td>
<td>Leave this box unchecked.</td>
</tr>
<tr>
<td>Task</td>
<td>Accept the default instance type.</td>
</tr>
<tr>
<td></td>
<td>Task nodes only process Hadoop tasks and don't store data. You can add and remove them from a cluster to manage the EC2 instance capacity your cluster uses, increasing capacity to handle peak loads and decreasing it later. Task nodes only run a TaskTracker Hadoop daemon.</td>
</tr>
<tr>
<td>Count</td>
<td>Choose 0.</td>
</tr>
<tr>
<td>Request Spot Instances</td>
<td>Leave this box unchecked.</td>
</tr>
</tbody>
</table>

3. In the **Security and Access** section, complete the fields according to the following table.

<table>
<thead>
<tr>
<th>Field</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EC2 key pair</strong></td>
<td>Choose your Amazon EC2 key pair private key from the list.</td>
</tr>
<tr>
<td></td>
<td>Optionally, choose <strong>Proceed without an EC2 key pair</strong>. If you do not enter a value in this field, you cannot use SSH to connect to the master node. For more information, see <a href="#">Connect to the Master Node Using SSH</a>.</td>
</tr>
<tr>
<td><strong>IAM user access</strong></td>
<td>Choose <strong>All other IAM users</strong> to make the cluster visible and accessible to all IAM users on the AWS account. For more information, see <a href="#">Use IAM Policies to Allow and Deny User Permissions</a>.</td>
</tr>
</tbody>
</table>
### Install HBase on an Amazon EMR Cluster

<table>
<thead>
<tr>
<th>Field</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Alternatively, choose No other IAM users to restrict access to the current IAM user.</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Roles configuration</strong></td>
<td>Choose <strong>Default</strong> to generate the default Amazon EMR role and Amazon EC2 instance profile. If the default roles exist, they are used for your cluster. If they do not exist, they are created (assuming you have proper permissions). You may also choose <strong>View policies for default roles</strong> to view the default role properties. Alternatively, if you have custom roles, you can choose <strong>Custom</strong> and choose your roles. An Amazon EMR role and Amazon EC2 instance profile are required when creating a cluster using the console. The role allows Amazon EMR to access other AWS services on your behalf. The Amazon EC2 instance profile controls application access to the Amazon EC2 instances in the cluster. For more information, see [Configure IAM Roles for Amazon EMR Permissions to AWS Services](p. 234).</td>
</tr>
</tbody>
</table>

9. In the **Bootstrap Actions** section, there are no bootstrap actions necessary for this sample configuration.

   Optionally, you can use bootstrap actions, which are scripts that can install additional software and change the configuration of applications on the cluster before Hadoop starts. For more information, see [(Optional) Create Bootstrap Actions to Install Additional Software](p. 129).

10. In the **Steps** section, you do not need to change any of these settings.

11. Review your configuration and if you are satisfied with the settings, choose **Create Cluster**.

12. When the cluster starts, the console displays the **Cluster Details** page.

---

### To launch a cluster and install HBase using the AWS CLI

You can install HBase on a cluster using the AWS CLI by typing the `create-cluster` subcommand with the **--applications** parameter. When using the **--applications** parameter, identify the application to install via the **Name** argument.

By default, clusters launched with the AWS CLI have termination protection disabled. To prevent the cluster from being terminated inadvertently or in the case of an error, add the **--termination-protected** parameter to the `create-cluster` subcommand.

1. If you have not previously created the default Amazon EMR role and Amazon EC2 instance profile, type the following command to create them. Alternatively, you can specify your own roles. If you do not specify the Amazon EMR role and Amazon EC2 instance profile when creating your cluster, you may experience issues with HBase backup and restore functionality. For more information about roles, see [Configure IAM Roles for Amazon EMR Permissions to AWS Services](p. 234).

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

2. To install HBase when a cluster is launched, type the following command and replace **myKey** with the name of your Amazon EC2 key pair.
   
   ```bash
   aws emr create-cluster --name "Test cluster" --ami-version 3.3 \
   --applications Name=Hue Name=Hive Name= Pig Name=HBase \
   --use-default-roles --ec2-attributes KeyName=myKey \
   --instance-type c1.xlarge --instance-count 3 --termination-protected
   ```

---

331
Windows users:

```java
aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applications Name=Hue Name=Hive Name=Pig Name=HBase --use-default-roles --ec2-attributes KeyName=myKey --instance-type c1.xlarge --instance-count 3 --termination-protected
```

**Note**

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

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

---

**Using the HBase Shell**

After you create an HBase cluster, the next step is to connect to HBase so you can begin reading and writing data (data writes are not supported on a read-replica cluster). You can use the HBase shell to test commands.

**To open the HBase shell**

1. Use SSH to connect to the master server in the HBase cluster. For information about how to connect to the master node using SSH, see [Connect to the Master Node Using SSH](#).
2. Run `hbase shell`. The HBase shell opens with a prompt similar to the following example:

```
hbase(main):001:0>
```

You can issue HBase shell commands from the prompt. For more information about the shell commands and how to call them, type `help` at the HBase prompt and press Enter.

**Create a Table**

The following command creates a table named ‘t1’ that has a single column family named ‘f1’:

```
hbase(main):001:0>create 't1', 'f1'
```

**Put a Value**

The following command puts value ‘v1’ for row ‘r1’ in table ‘t1’ and column ‘f1’:

```
hbase(main):001:0>put 't1', 'r1', 'f1:col1', 'v1'
```

**Get a Value**

The following command gets the values for row ‘r1’ in table ‘t1’:
Access HBase Tables with Hive

HBase and Hive and Amazon EMR (EMR 3.x Releases) (p. 260) are tightly integrated, allowing you run massively parallel processing workloads directly on data stored in HBase. To use Hive with HBase, you can usually launch them on the same cluster. You can, however, launch Hive and HBase on separate clusters. Running HBase and Hive separately on different clusters can improve performance because this allows each application to use cluster resources more efficiently.

The following procedures show how to connect to HBase on a cluster using Hive.

**Note**
You can only connect a Hive cluster to a single HBase cluster.

**To connect Hive to HBase**

1. Create separate clusters with Hive and HBase installed or create a single cluster with both HBase and Hive installed.
2. If you are using separate clusters, modify your security groups so that HBase and Hive ports are open between these two master nodes.
3. Use SSH to connect to the master node for the cluster with Hive installed. For more information, see Connect to the Master Node Using SSH (p. 457).
4. Launch the Hive shell with the following command.

   ```
   hive
   ```

5. (Optional) You do not need to do this if HBase and Hive are located on the same cluster. Connect the HBase client on your Hive cluster to the HBase cluster that contains your data. In the following example, `public-DNS-name` is replaced by the public DNS name of the master node of the HBase cluster, for example: `ec2-50-19-76-67.compute-1.amazonaws.com`.

   ```
   set hbase.zookeeper.quorum=public-DNS-name;
   ```

6. Proceed to run Hive queries on your HBase data as desired or see the next procedure.

**To access HBase data from Hive**

- After the connection between the Hive and HBase clusters has been made (as shown in the previous procedure), you can access the data stored on the HBase cluster by creating an external table in Hive.

The following example, when run from the Hive prompt, creates an external table that references data stored in an HBase table called `inputTable`. You can then reference `inputTable` in Hive statements to query and modify data stored in the HBase cluster.

**Note**

The following example uses `protobuf-java-2.4.0a.jar` in AMI 2.3.3, but you should modify the example to match your version. To check which version of the Protocol Buffers JAR you have, run the command at the Hive command prompt: `ls /home/hadoop/lib;`
add jar lib/emr-metrics-1.0.jar;
add jar lib/protobuf-java-2.4.0a.jar;
set hbase.zookeeper.quorum=ec2-107-21-163-157.compute-1.amazonaws.com;
create external table inputTable (key string, value string)
   stored by 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
   with serdeproperties ("hbase.columns.mapping" = "key,f1:col1")
   tblproperties ("hbase.table.name" = "t1");
select count(*) from inputTable;

For a more advanced use case and example combining HBase and Hive, see the AWS Big Data Blog post, Combine NoSQL and Massively Parallel Analytics Using Apache HBase and Apache Hive on Amazon EMR.

Back Up and Restore HBase

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Amazon EMR provides the ability to back up your HBase data to Amazon S3, either manually or on an automated schedule. You can perform both full and incremental backups. After you have a backed-up version of HBase data, you can restore that version to an HBase cluster. You can restore to an HBase cluster that is currently running, or launch a new cluster pre-populated with backed-up data.

During the backup process, HBase continues to execute write commands. Although this ensures that your cluster remains available throughout the backup, there is the risk of inconsistency between the data being backed up and any write operations being executed in parallel. To understand the inconsistencies that might arise, you have to consider that HBase distributes write operations across the nodes in its cluster. If a write operation happens after a particular node is polled, that data is not included in the backup archive. You may even find that earlier writes to the HBase cluster (sent to a node that has already been polled) might not be in the backup archive, whereas later writes (sent to a node before it was polled) are included.

If a consistent backup is required, you must pause writes to HBase during the initial portion of the backup process, synchronization across nodes. You can do this by specifying the --consistent parameter when requesting a backup. With this parameter, writes during this period are queued and executed as soon as the synchronization completes. You can also schedule recurring backups, which resolves any inconsistencies over time, as data that is missed on one backup pass is backed up on the following pass.

When you back up HBase data, you should specify a different backup directory for each cluster. An easy way to do this is to use the cluster identifier as part of the path specified for the backup directory. For example, s3://mybucket/backups/j-3AEXXXXXXX16F2. This ensures that any future incremental backups reference the correct HBase cluster.

When you are ready to delete old backup files that are no longer needed, we recommend that you first do a full backup of your HBase data. This ensures that all data is preserved and provides a baseline for future incremental backups. After the full backup is done, you can navigate to the backup location and manually delete the old backup files.

The HBase backup process uses S3DistCp for the copy operation, which has certain limitations regarding temporary file storage space. For more information, see S3DistCp (s3-dist-cp) (p. 386).
Back Up and Restore HBase Using the Console

The console provides the ability to launch a new cluster and populate it with data from a previous HBase backup. It also gives you the ability to schedule periodic incremental backups of HBase data. Additional backup and restore functionality, such as the ability to restore data to an already running cluster, do manual backups, and schedule automated full backups, is available using the CLI. For more information, see Back Up and Restore HBase Using the AWS CLI (p. 336).

To populate a new cluster with archived HBase data using the console

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. Choose Create cluster.
3. In the Software Configuration section, for Additional Applications, choose HBase and Configure and add.
4. On the Add Application dialog box, check Restore From Backup. For more information, see Install HBase on an Amazon EMR Cluster (p. 327).
5. For Backup Location, specify the location of the backup you wish to load into the new HBase cluster. This should be an Amazon S3 URL of the form s3://myawsbucket/backups/.
6. For Backup Version, you have the option to specify the name of a backup version to load by setting a value. If you do not set a value for Backup Version, Amazon EMR loads the latest backup in the specified location.

7. Choose Add and proceed to create the cluster with other options as desired.

To schedule automated backups of HBase data using the console

1. In the Software Configuration section, for Additional Applications, choose HBase and Configure and add.
2. Choose Schedule Regular Backups.
3. Specify whether the backups should be consistent. A consistent backup is one which pauses write operations during the initial backup stage, synchronization across nodes. Any write operations thus paused are placed in a queue and resume when synchronization completes.
4. Set how often backups should occur by entering a number for Backup Frequency and choosing Days, Hours, or Minutes. The first automated backup that runs is a full backup; after that, Amazon EMR saves incremental backups based on the schedule that you specify.
5. Specify the location in Amazon S3 where the backups should be stored. Each HBase cluster should be backed up to a separate location in Amazon S3 to ensure that incremental backups are calculated correctly.
6. Specify when the first backup should occur by setting a value for Backup Start Time. You can set this to now, which causes the first backup to start as soon as the cluster is running, or enter a date
and time in **ISO format**. For example, `2013-09-26T20:00Z`, sets the start time to September 26, 2013 at 8pm UTC.

7. Choose **Add**.
8. Proceed with creating the cluster with other options as desired.

## Back Up and Restore HBase Using the AWS CLI

Running HBase on Amazon EMR provides many ways to back up your data, you can create full or incremental backups, run backups manually, and schedule automatic backups.

Using the AWS CLI, you can create HBase backups, restore HBase data from backup when creating an EMR cluster, schedule HBase backups, restore HBase from backup data in Amazon S3, and disable HBase backups.

### To manually create an HBase backup using the AWS CLI

To create an HBase backup, type the `create-hbase-backup` subcommand with the `--dir` parameter to identify the backup location in Amazon S3. Amazon EMR tags the backup with a name derived from the time the backup was launched. This is in the format `YYYYMMDDTHHMMSSZ`, for example: `20120809T031314Z`. To label your backups with another name, you can create a location in Amazon S3 (such as `backups` in the example below) and use the location name as a way to tag the backup files.

- Type the following command to back up HBase data to `s3://mybucket/backups`, with the timestamp as the version name. Replace `j-3AEXXXXXX16F2` with the cluster ID and replace `mybucket` with your Amazon S3 bucket name. This backup does not pause writes to HBase and as such, may be inconsistent.
Type the following command to back up data and use the --consistent parameter to enforce backup consistency. This flag pauses all writes to HBase during the backup:

```
aws emr create-hbase-backup --cluster-id j-3AEXXXXX16F2 --dir s3://mybucket/backups/j-3AEXXXXX16F2 --consistent
```

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

**To schedule automated backups of HBase data using the AWS CLI**

To schedule HBase backups, type the `schedule-hbase-backup` subcommand with the --interval and --unit parameters. If you do not specify a start time, the first backup starts immediately. Use the --consistent parameter to pause all write operations to HBase during the backup process.

- Use the following command examples to schedule consistent HBase backups:

  To create a consistent weekly full backup, with the first backup starting immediately, type the following command, replace `j-3AEXXXXX16F2` with the cluster ID, and replace `mybucket` with your Amazon S3 bucket name:

  ```
  aws emr schedule-hbase-backup --cluster-id j-3AEXXXXX16F2 --type full --dir s3://mybucket/backups/j-3AEXXXXX16F2 --interval 7 --unit days --consistent
  ```

  To create a consistent weekly full backup, with the first backup starting on 15 June 2014, 8 p.m. UTC time, type:

  ```
  aws emr schedule-hbase-backup --cluster-id j-3AEXXXXX16F2 --type full --dir s3://mybucket/backups/j-3AEXXXXX16F2 --interval 7 --unit days --start-time 2014-06-15T20:00Z --consistent
  ```

  To create a consistent daily incremental backup with the first backup beginning immediately, type:

  ```
  aws emr schedule-hbase-backup --cluster-id j-3AEXXXXX16F2 --type incremental --dir s3://mybucket/backups/j-3AEXXXXX16F2 --interval 24 --unit hours --consistent
  ```

  To create a consistent daily incremental backup, with the first backup starting on 15 June 2014, 8 p.m. UTC time, type:

  ```
  aws emr schedule-hbase-backup --cluster-id j-3AEXXXXX16F2 --type incremental --dir s3://mybucket/backups/j-3AEXXXXX16F2 --interval 24 --unit hours --start-time 2014-06-15T20:00Z --consistent
  ```

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

**To disable HBase backups using the AWS CLI**

To disable HBase backups, type the `disable-hbase-backups` subcommand with the --cluster-id parameter. The cluster ID can be retrieved using the console or the `list-clusters` subcommand. When disabling backups, identify the backup type: --full or --incremental.
• Type the following command to disable full backups and replace `j-3AEXXXXXXX16F2` with your cluster ID.

```bash
aws emr disable-hbase-backups --cluster-id j-3AEXXXXXXX16F2 --full
```

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

**To restore HBase backup data to a running cluster using the AWS CLI**

To restore HBase backup data to a running cluster, type the `restore-from-hbase-backup` subcommand with the `--cluster-id` parameter. To restore from backup, you must provide the backup directory and (optionally) the backup version. The backup version specifies the version number of an existing backup to restore. If the backup version is not specified, Amazon EMR uses the latest backup, as determined by lexicographical order. This is in the format `YYYYMMDDTHHMMSSZ`, for example: `20120809T031314Z`.

• To restore HBase backup data to a running cluster, type the following command, replace `j-3AEXXXXXXX16F2` with your cluster ID, and replace `mybucket` with your Amazon S3 bucket name.

```bash
aws emr restore-from-hbase-backup --cluster-id j-3AEXXXXXXX16F2 --dir s3://mybucket/backups/j-3AEXXXXXXX16F2 --backup-version 20120809T031314Z
```

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

**To populate a new cluster with HBase backup data using the AWS CLI**

To populate a new cluster with HBase backup data, type the `create-cluster` subcommand with the `--restore-from-hbase-backup` parameter. To restore from backup, you must provide the backup directory and (optionally) the backup version. The backup version specifies the version number of an existing backup to restore. If the backup version is not specified, Amazon EMR uses the latest backup, as determined by lexicographical order. This is in the format `YYYYMMDDTHHMMSSZ`, for example: `20120809T031314Z`.

• Type the following command to create a cluster with HBase installed and to load HBase with the backup data in `s3://mybucket/backups/j-3AEXXXXXXX16F2`. Replace `myKey` with the name of your Amazon EC2 key pair.

  - Linux, UNIX, and Mac OS X users:

  ```bash
  aws emr create-cluster --name "Test cluster" --ami-version 3.3
  --applications Name=Hue Name=Hive Name=Pig Name=HBase
  --restore-from-hbase-backup Dir=s3://mybucket/backups/j-3AEXXXXXXX16F2,BakcupVersion=20120809T031314Z
  --instance-type c1.xlarge --instance-count 3 --termination-protected
  --use-default-roles --ec2-attributesKeyName=myKey
  ```

  - Windows users:

  ```bash
  aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applications Name=Hue Name=Hive Name=Pig Name=HBase --restore-from-hbase-backup Dir=s3://mybucket/backups/j-3AEXXXXXXX16F2,BakcupVersion=20120809T031314Z --use-default-roles --ec2-attributesKeyName=myKey --instance-type c1.xlarge --instance-count 3 --termination-protected
  ```
When you specify the instance count without using the `--instance-groups` parameter, a single master node is launched, and the remaining instances are launched as core nodes. All nodes use the instance type specified in the command.

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

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

## Terminate an HBase Cluster

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

By default, Amazon EMR launches clusters created using the console with termination protection turned on. This prevents the cluster from being terminated inadvertently or in the case of an error. Before you terminate the cluster, you must first disable termination protection.

Clusters created using the AWS CLI have termination protection disabled. To prevent the cluster from being terminated inadvertently or in the case of an error, add the `--termination-protected` parameter to the `create-cluster` subcommand.

## Configure HBase

Although the default settings should work for most applications, you have the flexibility to modify your HBase configuration settings. To do this, run one of two bootstrap action scripts:

- **configure-hbase-daemons**—Configures properties of the master, regionserver, and zookeeper daemons. These properties include heap size and options to pass to the Java Virtual Machine (JVM) when the HBase daemon starts. You set these properties as arguments in the bootstrap action. This bootstrap action modifies the `/home/hadoop/conf/hbase-user-env.sh` configuration file on the HBase cluster.

- **configure-hbase**—Configures HBase site-specific settings such as the port the HBase master should bind to and the maximum number of times the client CLI client should retry an action. You can set these one-by-one, as arguments in the bootstrap action, or you can specify the location of an XML configuration file in Amazon S3. This bootstrap action modifies the `/home/hadoop/conf/hbase-site.xml` configuration file on the HBase cluster.

**Note**
These scripts, like other bootstrap actions, can only be run when the cluster is created; you cannot use them to change the configuration of an HBase cluster that is currently running.

When you run the `configure-hbase` or `configure-hbase-daemons` bootstrap actions, the values you specify override the default values. Any values that you don't explicitly set receive the default values.

Configuring HBase with these bootstrap actions is analogous to using bootstrap actions in Amazon EMR to configure Hadoop settings and Hadoop daemon properties. The difference is that HBase does not have per-process memory options. Instead, memory options are set using the `--daemon-opts` argument, where `daemon` is replaced by the name of the daemon to configure.
Configure HBase Daemons

Amazon EMR provides a bootstrap action, s3://region.elasticmapreduce/bootstrap-actions/configure-hbase-daemons, that you can use to change the configuration of HBase daemons, where region is the region into which you're launching your HBase cluster.

For more information about regions supported by Amazon EMR see Choose an AWS Region (p. 27). The bootstrap action can only be run when the HBase cluster is launched.

You can configure a bootstrap action using the console, the AWS CLI, or the API. For more information about configuring bootstrap actions, see (Optional) Create Bootstrap Actions to Install Additional Software (p. 129).

To configure HBase daemons using the AWS CLI

Add the bootstrap action, configure-hbase-daemons, when you launch the cluster to configure one or more HBase daemons. You can set the following properties with the configure-hbase-daemons bootstrap action.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hbase-master-opts</td>
<td>Options that control how the JVM runs the master daemon. If set, these override the default HBASE_MASTER_OPTS variables.</td>
</tr>
<tr>
<td>regionserver-opts</td>
<td>Options that control how the JVM runs the region server daemon. If set, these override the default HBASE_REGIONSERVER_OPTS variables.</td>
</tr>
<tr>
<td>zookeeper-opts</td>
<td>Options that control how the JVM runs the zookeeper daemon. If set, these override the default HBASE_ZOOKEEPER_OPTS variables.</td>
</tr>
</tbody>
</table>

For more information about these options, see hbase-env.sh in the HBase documentation.

• To use a bootstrap action to configure values for zookeeper-opts and hbase-master-opts, type the following command and replace myKey with the name of your EC2 key pair.

  • Linux, UNIX, and Mac OS X users:

    ```
    aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applications Name=Hue Name=Hive Name=Pig Name=HBase --use-default-roles --ec2-attributes KeyName=myKey --instance-type c1.xlarge --instance-count 3 --termination-protected --bootstrap-action Path=s3://elasticmapreduce/bootstrap-actions/configure-hbase-daemons,Args=["--hbase-zookeeper-opts=-Xmx1024m -XX:GCTimeRatio=19","--hbase-master-opts=-Xmx2048m","--hbase-regionserver-opts=-Xmx4096m"]
    ```

  • Windows users:

    ```
    aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applications Name=Hue Name=Hive Name=Pig Name=HBase --use-default-roles --ec2-attributes KeyName=myKey --instance-type c1.xlarge --instance-count 3 --termination-protected --bootstrap-action Path=s3://elasticmapreduce/bootstrap-actions/configure-hbase-daemons,Args=["--hbase-zookeeper-opts=-Xmx1024m -XX:GCTimeRatio=19","--hbase-master-opts=-Xmx2048m","--hbase-regionserver-opts=-Xmx4096m"]
    ```

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

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

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

### Configure HBase Site Settings

Amazon EMR provides a bootstrap action, `s3://elasticmapreduce/bootstrap-actions/configure-hbase`, that you can use to change the configuration of HBase. You can set configuration values one-by-one, as arguments in the bootstrap action, or you can specify the location of an XML configuration file in Amazon S3. Setting configuration values one-by-one is useful if you only need to set a few configuration settings. Setting them using an XML file is useful if you have many changes to make, or if you want to save your configuration settings for reuse.

**Note**
You can prefix the Amazon S3 bucket name with a region prefix, such as `s3://region.elasticmapreduce/bootstrap-actions/configure-hbase`, where `region` is the region into which you're launching your HBase cluster. For more information about the regions supported by Amazon EMR, see Choose an AWS Region (p. 27).

This bootstrap action modifies the `/home/hadoop/conf/hbase-site.xml` configuration file on the HBase cluster. The bootstrap action can only be run when the HBase cluster is launched. For more information about configuring bootstrap actions, see (Optional) Create Bootstrap Actions to Install Additional Software (p. 129)

For more information about the HBase site settings that you can configure, see Default Configuration in the HBase documentation.

**To specify individual HBase site settings using the AWS CLI**

Set the `configure-hbase` bootstrap action when you launch the HBase cluster and specify the values in `hbase-site.xml` to change.

- To change the `hbase.hregion.max.filesize` setting, type the following command and replace `myKey` with the name of your Amazon EC2 key pair.

  Linux, UNIX, and Mac OS X users:

  ```bash
  aws emr create-cluster --name "Test cluster" --ami-version 3.3
  --applications Name=Hue Name=Hive Name=Pig Name=HBase
  --use-default-roles --ec2-attributes KeyName=myKey
  --instance-type c1.xlarge --instance-count 3 --termination-protected
  --bootstrap-action Path=s3://elasticmapreduce/bootstrap-actions/configure-hbase,Args=["-s","hbase.hregion.max.filesize=52428800"]
  ```

  Windows users:

  ```bash
  aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applications Name=Hue Name=Hive Name=Pig Name=HBase
  --use-default-roles --ec2-attributes
  ```

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

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

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

### To specify HBase site settings with an XML file using the AWS CLI

1. Create a custom version of `hbase-site.xml`. Your custom file must be valid XML. To reduce the chance of introducing errors, start with the default copy of `hbase-site.xml`, located on the Amazon EMR HBase master node at `/home/hadoop/conf/hbase-site.xml`, and edit a copy of that file instead of creating a file from scratch. You can give your new file a new name, or leave it as `hbase-site.xml`.

2. Upload your custom `hbase-site.xml` file to an Amazon S3 bucket. It should have permissions set so the AWS account that launches the cluster can access the file. If the AWS account launching the cluster also owns the Amazon S3 bucket, it will have access.

3. Set the `configure-hbase` bootstrap action when you launch the HBase cluster, and include the location of your custom `hbase-site.xml` file. The following example sets the HBase site configuration values to those specified in the file `s3://mybucket/config.xml`. Type the following command, replace `myKey` with the name of your EC2 key pair, and replace `mybucket` with the name of your Amazon S3 bucket.

   - **Linux, UNIX, and Mac OS X users:**
     ```bash
     aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applications Name=Hue Name=Hive Name=Pig Name=HBase --use-default-roles --ec2-attributes KeyName=myKey --instance-type c1.xlarge --instance-count 3 --termination-protected --bootstrap-action Path=s3://elasticmapreduce/bootstrap-actions/configure-hbase,Args=["--site-config-file","s3://mybucket/config.xml"]
     ```

   - **Windows users:**
     ```bash
     aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applications Name=Hue Name=Hive Name=Pig Name=HBase --use-default-roles --ec2-attributes --instance-type c1.xlarge --instance-count 3 --termination-protected --bootstrap-action Path=s3://elasticmapreduce/bootstrap-actions/configure-hbase,Args=["--site-config-file","s3://mybucket/config.xml"]
     ```

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

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

You can set any or all of the HBase site settings to optimize the HBase cluster for your application's workload. We recommend the following settings as a starting point in your investigation. If you specify more than one option, you must prepend each key-value pair with a `-s` option switch. All options below are for the AWS CLI.

**zookeeper.session.timeout**

The default timeout is three minutes (180000 ms). If a region server crashes, this is how long it takes the master server to notice the absence of the region server and start recovery. To help the master server recover faster, you can reduce this value to a shorter time period. The following example uses one minute, or 60000 ms:

```
--bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hbase,Args=["-s","zookeeper.session.timeout=60000"]
```

**hbase.regionserver.handler.count**

This defines the number of threads the region server keeps open to serve requests to tables. The default of 10 is low, in order to prevent users from killing their region servers when using large write buffers with a high number of concurrent clients. The rule of thumb is to keep this number low when the payload per request approaches the MB range (big puts, scans using a large cache) and high when the payload is small (gets, small puts, ICVs, deletes). The following example raises the number of open threads to 30:

```
--bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hbase,Args=["-s","hbase.regionserver.handler.count=30"]
```

**hbase.hregion.max.filesize**

This parameter governs the size, in bytes, of the individual regions. By default, it is set to 256 MB. If you are writing a lot of data into your HBase cluster and it's causing frequent splitting, you can increase this size to make individual regions bigger. It reduces splitting but takes more time to load balance regions from one server to another.

```
--bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hbase,Args=["-s","hbase.hregion.max.filesize=1073741824"]
```

**hbase.hregion.memstore.flush.size**

This parameter governs the maximum size of memstore, in bytes, before it is flushed to disk. By default, it is 64 MB. If your workload consists of short bursts of write operations, you might want to increase this limit so all writes stay in memory during the burst and get flushed to disk later. This can boost performance during bursts.
View the HBase User Interface

HBase provides a web-based user interface that you can use to monitor your HBase cluster. When you run HBase on Amazon EMR, the web interface runs on the master node and can be viewed using port forwarding, also known as creating an SSH tunnel.

To view the HBase User Interface

1. Use SSH to tunnel into the master node and create a secure connection. For more information, see Option 2, Part 1: Set Up an SSH Tunnel to the Master Node Using Dynamic Port Forwarding (p. 466).
2. Install a web browser with a proxy tool, such as the FoxyProxy plug-in for Firefox, to create a SOCKS proxy for AWS domains. For more information, see Option 2, Part 2: Configure Proxy Settings to View Websites Hosted on the Master Node (p. 468).
3. With the proxy set and the SSH connection open, you can view the HBase UI by opening a browser window with http://master-public-dns-name:60010/master-status, where master-public-dns-name is the public DNS address of the master server in the HBase cluster.

View HBase Log Files

As part of its operation, HBase writes log files with details about configuration settings, daemon actions, and exceptions. These log files can be useful for debugging issues with HBase as well as for tracking performance.

If you configure your cluster to persist log files to Amazon S3, you should know that logs are written to Amazon S3 every five minutes, so there may be a slight delay before the latest log files are available.

To view HBase logs on the master node

- You can view the current HBase logs by using SSH to connect to the master node, and navigating to the mnt/var/log/hbase directory. These logs are not available after the cluster is terminated unless you enable logging to Amazon S3 when the cluster is launched. For more information, see Connect to the Master Node Using SSH (p. 457). After you have connected to the master node using SSH, you can navigate to the log directory using a command like the following:

```
cd mnt/var/log/hbase
```

To view HBase logs on Amazon S3

- To access HBase logs and other cluster logs on Amazon S3, and to have them available after the cluster ends, you must specify an Amazon S3 bucket to receive these logs when you create the cluster. This is done using the --log-uri option. For more information about enabling logging for your cluster, see Configure Cluster Logging and Debugging (p. 167).
Monitor HBase with CloudWatch

Amazon EMR reports three metrics to CloudWatch that you can use to monitor your HBase backups. These metrics are pushed to CloudWatch at five-minute intervals, and are provided without charge. For more information about using CloudWatch to monitor Amazon EMR metrics, see Monitor Metrics with CloudWatch (p. 434).

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HBaseBackupFailed</strong></td>
<td>Whether the last backup failed. This is set to 0 by default and updated to 1 if the previous backup attempt failed. This metric is only reported for HBase clusters. Use case: Monitor HBase backups Units: Count</td>
</tr>
<tr>
<td><strong>HBaseMostRecentBackupDuration</strong></td>
<td>The amount of time it took the previous backup to complete. This metric is set regardless of whether the last completed backup succeeded or failed. While the backup is ongoing, this metric returns the number of minutes after the backup started. This metric is only reported for HBase clusters. Use case: Monitor HBase Backups Units: Minutes</td>
</tr>
<tr>
<td><strong>HBaseTimeSinceLastSuccessfulBackup</strong></td>
<td>The number of elapsed minutes after the last successful HBase backup started on your cluster. This metric is only reported for HBase clusters. Use case: Monitor HBase backups Units: Minutes</td>
</tr>
</tbody>
</table>

Monitor HBase with Ganglia

The Ganglia open-source project 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. For more information about the Ganglia open-source project, see http://ganglia.info/. For more information about using Ganglia with Amazon EMR clusters, see Monitor Performance with Ganglia (p. 450).

You configure Ganglia for HBase using the configure-hbase-for-ganglia bootstrap action. This bootstrap action configures HBase to publish metrics to Ganglia.

**Note**

You must configure HBase and Ganglia when you launch the cluster; Ganglia reporting cannot be added to a running cluster.
After the cluster is launched with Ganglia configured, you can access the Ganglia graphs and reports using the graphical interface running on the master node.

Ganglia also stores log files on the server at /mnt/var/log/ganglia/rrds. If you configured your cluster to persist log files to an Amazon S3 bucket, the Ganglia log files are persisted there as well.

To configure a cluster for Ganglia and HBase using the AWS CLI

- To launch a cluster and specify the `configure-hbase-for-ganglia` bootstrap action, type the following command and replace `myKey` with the name of your Amazon EC2 key pair.

  **Note**
  You can prefix the Amazon S3 bucket path with the region where your HBase cluster was launched, for example, `s3://region.elasticmapreduce/bootstrap-actions/configure-hbase-for-ganglia`. For more information about regions supported by Amazon EMR, see Choose an AWS Region (p. 27).

- Linux, UNIX, and Mac OS X users:

  ```bash
  aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applications Name=Hue Name=Hive Name=Pig Name=HBase Name=Ganglia --use-default-roles --ec2-attributes KeyName=myKey --instance-type c1.xlarge --instance-count 3 --termination-protected --bootstrap-action Path=s3://elasticmapreduce/bootstrap-actions/configure-hbase-for-ganglia
  ```

- Windows users:

  ```bash
  aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applications Name=Hue Name=Hive Name=Pig Name=HBase Name=Ganglia --use-default-roles --ec2-attributes KeyName=myKey --instance-type c1.xlarge --instance-count 3 --termination-protected --bootstrap-action Path=s3://elasticmapreduce/bootstrap-actions/configure-hbase-for-ganglia
  ```

To view HBase metrics in the Ganglia web interface

1. Use SSH to tunnel into the master node and create a secure connection. For more information, see Option 2, Part 1: Set Up an SSH Tunnel to the Master Node Using Dynamic Port Forwarding (p. 466).

2. Install a web browser with a proxy tool, such as the FoxyProxy plug-in for Firefox, to create a SOCKS proxy for AWS domains. For more information, see Option 2, Part 2: Configure Proxy Settings to View Websites Hosted on the Master Node (p. 468).

3. With the proxy set and the SSH connection open, you can view the Ganglia metrics by opening a browser window with `http://master-public-dns-name/ganglia/`, where `master-public-dns-name` is the public DNS address of the master server in the HBase cluster.

To view Ganglia log files on the master node

- If the cluster is still running, you can access the log files by using SSH to connect to the master node and navigating to the `/mnt/var/log/ganglia/rrds` directory. For more information, see Connect to the Master Node Using SSH (p. 457).

To view Ganglia log files on Amazon S3

- If you configured the cluster to persist log files to Amazon S3 when you launched it, the Ganglia log files are written there as well. Logs are written to Amazon S3 every five minutes, so there may
be a slight delay before the latest log files are available. For more information, see View HBase Log Files (p. 344).

Migrating from Previous HBase Versions

To migrate data from a previous HBase version, see Upgrading and HBase version number and compatibility in the Apache HBase Reference Guide. You may need to pay special attention to the requirements for upgrading from pre-1.0 versions of HBase.
Configure Hue to View, Query, or Manipulate Data

This chapter consists of the following topics.

Topics
- What is Hue? (p. 348)
- Create a Cluster with Hue Installed (p. 349)
- Launch the Hue Web Interface (p. 350)
- Use Hue with a Remote Database in Amazon RDS (p. 350)
- Advanced Configurations for Hue (p. 354)
- Metastore Manager Restrictions (p. 357)

What is Hue?

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Hue is an open-source, web-based graphical user interface for use with Amazon EMR and Apache Hadoop. Hue groups together several different Hadoop ecosystem projects into a configurable interface for your Amazon EMR cluster. Amazon has also added customizations specific to Hue on Amazon EMR. You launch your cluster using the Amazon EMR console and you can interact with Hadoop and related components on your cluster using Hue. For more information about Hue, go to http://gethue.com.

What are the major features of Hue on Amazon EMR?

Hue on Amazon EMR supports the following:

- Amazon S3 and Hadoop File System (HDFS) Browser—if you have appropriate permissions, you can browse and move data between the ephemeral HDFS storage and Amazon S3 buckets belonging to your account.
- Hive—you use the Hive editor to run interactive queries on your data. This is also a useful way to prototype programmatic or batched querying.
- Pig—you use the Pig editor to run scripts on your data or to issue interactive commands.
- Metastore Manager—allows you to view and manipulate the contents of the Hive metastore (import/create, drop, and so on).
- Job browser—use the job browser to see the status of your submitted Hadoop jobs.
- User management—allows you to manage Hue user accounts and to integrate LDAP users with Hue.
- AWS Samples—there are several “ready-to-run” examples, which process sample data from various AWS services using applications in Hue. When you login to Hue, you are taken to the Hue Home application where the samples are pre-installed.

Hue versus the AWS Management Console
Cluster administrators use the AWS Management Console to launch and administer clusters. This is also the case when you want to launch a cluster with Hue installed. On the other hand, end-users may interact entirely with their Amazon EMR cluster through an application such as Hue. Hue acts as a front-end for the applications on the cluster and it allows users to interact with their cluster with a more user-friendly interface. The applications in Hue, such as the Hive and Pig editors, replace the need to login to the cluster to interactively run scripts with their respective shell applications.

**Supported Hue versions**

The initial version of Hue supported with Amazon EMR is **Hue 3.6**.

### Create a Cluster with Hue Installed

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

**To launch a cluster with Hue installed using the console**

1. Choose **Go to Advanced Options**.
2. Navigate to **Software Configuration** and choose Amazon for **Hadoop distribution** and 3.3.0 (or later) for the **AMI version**.
3. In **Software Configuration > Applications to be installed**, Hue should appear in the list by default.
4. In the **Hardware Configuration** section, accept the default EC2 instance types: m3.xlarge for instance types. You can change the instance types to suit your needs. If you will have more than 20 concurrent users accessing Hue, we recommend an instance type of m3.2xlarge or greater for the master node. We also recommend that you have a minimum of two core nodes for clusters running Hue.
5. In **Security and Access**, select a key pair for connecting to your cluster. You will need to use a key pair to open an SSH tunnel to connect to the Hue Web interface on the master node.
6. Click **Create cluster**.

**To launch a cluster with Hue installed using the AWS CLI**

To launch a cluster with Hue installed using the AWS CLI, type the create-cluster subcommand with the **--applications** parameter.

**Note**

You will need to install the current version of the AWS CLI. To download the latest release, see [https://aws.amazon.com/cli/](https://aws.amazon.com/cli/).

1. If you have not previously created the default EMR role and EC2 instance profile, type the following command to create them. Alternatively, you can specify your own roles. For more information on using your own roles, see [Configure IAM Roles for Amazon EMR Permissions to AWS Services (p. 234)](p. 234).

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

2. To launch an Amazon EMR cluster with Hue installed using the default roles, type the following command and replace `myKey` with the name of your EC2 key pair.

- Linux, UNIX, and Mac OS X users:

```bash
aws emr create-cluster --name "Hue cluster" --ami-version 3.11.0 --applications Name=Hue Name=Hive Name=Pig
```
Launch the Hue Web Interface

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Launching Hue is the same as connecting to any HTTP interface hosted on the master node of a cluster. The following procedure describes how to access the Hue interface. For more information on accessing web interfaces hosted on the master node, see: View Web Interfaces Hosted on Amazon EMR Clusters (p. 461).

To launch the Hue web interface

1. Follow these instructions to create an SSH tunnel to the master node and to configure an HTTP proxy add-in for your browser: Option 2, Part 1: Set Up an SSH Tunnel to the Master Node Using Dynamic Port Forwarding (p. 466).
2. Type the following address in your browser to open the Hue web interface: http://master public DNS:8888.
3. At the Hue login screen, if you are the administrator logging in for the first time, enter a username and password to create your Hue superuser account and then click Create account. Otherwise, type your username and password and click Create account or enter the credentials provided by your administrator.

Use Hue with a Remote Database in Amazon RDS

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

By default, Hue user information and query histories are stored in a local MySQL database on the master node. However, you can create one or more Hue-enabled clusters using a configuration stored in Amazon RDS.
S3 and a MySQL database in Amazon RDS. This allows you to persist user information and query history created by Hue without keeping your Amazon EMR cluster running. We recommend using Amazon S3 server-side encryption to store the configuration file.

First create the remote database for Hue.

**To create the external MySQL database**

1. Open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. Click **Launch a DB Instance**.
3. Choose MySQL and click **Select**.
4. Leave the default selection of **Multi-AZ Deployment and Provisioned IOPS Storage** and click **Next**.
5. Leave the Instance Specifications at their defaults, specify Settings, and click **Next**.
6. On the Configure Advanced Settings page, choose a proper security group and database name. The security group you use must at least allow ingress TCP access for port 3306 from the master node of your cluster. If you have not created your cluster at this point, you can allow all hosts to connect to port 3306 and adjust the security group after you have launched the cluster. Click **Launch DB Instance**.
7. Create your JSON configuration file:
   a. From the RDS Dashboard, click on **Instances** and select the instance you have just created. When your database is available, you can open a text editor and copy the following information into a file, `hueconfig.json`. The comments in the following JSON block provide corresponding names used in the RDS console:

   ```json
   {
     "hue": {
       "database": {
         "name": "dbname",
         "user": "username",
         "password": "password",
         "host": "hueinstance.c3b8qyyjyzi.us-east-1.rds.amazonaws.com",
         "port": "3306",
         "engine": "mysql"
       }
     }
   }
   ```
   b. Upload the Hue database configuration file to the administrator's Amazon S3 bucket using AWS CLI:

   ```sh
   aws s3 cp ./hueconfig.json s3://mybucket
   ```

**Creating a Configuration Role for Hue**

*Important*

Since Hue uses **EMR_EC2_DefaultRole** (e.g. InstanceProfile) by default to access Amazon S3 for both fetching the Hue database configuration and file browser functionality, your configuration file is exposed to all of your Hue users.

To avoid this, the administrator can configure an additional role to assume for accessing that configuration file. This way you can restrict **EMR_EC2_DefaultRole** so that users cannot access the configuration.

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. Create a role, called HueConfigRole.
   a. Click **Roles** and then **Create New Role**.
   b. In **Role Name**, enter HueConfigRole and click **Next Step**.
   c. Under **Role for Cross-Account Access** select **Provide access between AWS accounts you own** and click **Next Step**.
   d. Enter the **Account ID** for the Hue users and click **Next Step**.
   e. Choose Custom Policy and click **Select**.

   Choose a name for the following policy you are attaching to HueConfigRole:

   ```json
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Action": [
           "s3:Get*",
           "s3:List*"
         ],
         "Resource": "arn:aws:s3:::bucketName/path"
       }
     ]
   }
   ``

   where the `bucketName` and `path` are the location of your Hue configuration file. Click **Next Step**.
   f. Click **Create Role**.

3. Attach the following custom trust policy to **EMR_EC2_DefaultRole** by selecting the role in Dashboard and clicking **Attach Role Policy**:

   ```json
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Action": ["sts:AssumeRole"],
         "Resource": "arn:aws:iam::AccountID:role/HueConfigRole"
       },
       {
         "Effect": "Deny",
         "Action": ["s3:*"],
         "Resource": "arn:aws:s3:::bucketName/path"
       }
     ]
   }
   ``

   The `AccountID` is the account of the owner and is synonymous with root. For more information, see AWS Account Identifiers. This has the effect of denying bucket access to the normal **EMR_EC2_DefaultRole** instance profile where users are not administrators.

You can then launch a cluster with the configuration file you created.

**To specify an external MySQL database for Hue when launching a cluster using the console**

1. In the **Applications to be installed** section, click the **Edit** icon for Hue.
2. In the **Update Application** dialog, for **Use external database**, click **Yes**, type or browse to the location of your Hue configuration file in Amazon S3, and then click **Save**.

3. Proceed with creating your cluster using the steps in **Create a Cluster with Hue Installed** (p. 349).

**To specify an external MySQL database for Hue when launching a cluster using the AWS CLI**

To specify an external MySQL database for Hue when launching a cluster using the AWS CLI, type the `--applications` parameter and provide the path to the Hue configuration file in an arguments list.

*Note*
You can create multiple clusters that use the same external database, but each cluster will share query history and user information.

- To specify an external MySQL database for Hue when launching a cluster, type the following command, replace `myKey` with the name of your EC2 key pair, and replace `path-to-config.json` with the path to and filename of your `.json` configuration file.

  * Linux, UNIX, and Mac OS X users:

    ```
    aws emr create-cluster --name "Hue cluster" --ami-version 3.11.0 --applications
    Name=Hive Name=Pig \n    Name=Hue,Args=[--hue-config=s3://path-to-config.json,--hue-config-role=HueConfigRole]
    --use-default-roles --ec2-attributes KeyName=myKey \n    --instance-type m3.xlarge --instance-count 3
    ```

  * Windows users:

    ```
    aws emr create-cluster --name "Hue cluster" --ami-version 3.11.0 --applications
    Name=Hive Name=Pig Name=Hue,Args=[--hue-config=s3://path-to-config.json] --use-
    default-roles --ec2-attributes KeyName=myKey --instance-type m3.xlarge --instance-
    count 3
    ```

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

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

**Troubleshooting**

**In the event of Amazon RDS failover**

It is possible users may encounter delays when running a query because the Hue database instance is non-responsive or is in the process of failover. The following are some facts and guidelines for this issue:

- If you login to the Amazon RDS console, you can search for failover events. For example, to see if a failover is in process or has occurred, look for events such as "Multi-AZ instance failover started" and "Multi-AZ instance failover completed."
- It takes about 30 seconds for an RDS instance to complete a failover.
- If you are experiencing longer-than-normal responses for queries in Hue, try to re-execute the query.
Advanced Configurations for Hue

This section includes the following topics.

Topics

- Configure Hue for LDAP Users (p. 354)

Configure Hue for LDAP Users

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Integration with LDAP allows users to log into Hue using existing credentials stored in an LDAP directory. When you integrate Hue with LDAP, you do not need to independently manage user information in Hue. The information below demonstrates Hue integration with Microsoft Active Directory, but the configuration options are analogous to any LDAP directory.

LDAP authentication first binds to the server and establishes the connection. Then, the established connection is used for any subsequent queries to search for LDAP user information. Unless your Active Directory server allows anonymous connections, a connection needs to be established using a bind distinguished name and password. The bind distinguished name (or DN) is defined by the `bind_dn` configuration setting. The bind password is defined by the `bind_password` configuration setting. Hue has two ways to bind LDAP requests: search bind and direct bind. The preferred method for using Hue with Amazon EMR is search bind.

When search bind is used with Active Directory, Hue uses the user name attribute (defined by `user_name_attr config`) to find the attribute that needs to be retrieved from the base distinguished name (or DN). Search bind is useful when the full DN is not known for the Hue user.

For example, you may have `user_name_attr config` set to use the common name (or CN). In that case, the Active Directory server uses the Hue username provided during login to search the directory tree for a common name that matches, starting at the base distinguished name. If the common name for the Hue user is found, the user's distinguished name is returned by the server. Hue then constructs a distinguished name used to authenticate the user by performing a bind operation.

**Note**

Search bind searches usernames in all directory subtrees beginning at the base distinguished name. The base distinguished name specified in the Hue LDAP configuration should be the closest parent of the username, or your LDAP authentication performance may suffer.

When direct bind is used with Active Directory, the exact `nt_domain` or `ldap_username_pattern` must be used to authenticate. When direct bind is used, if the `nt_domain` (defined by the `nt_domain` configuration setting) attribute is defined, a user distinguished name template is created using the form: `<login username>@nt_domain`. This template is used to search all directory subtrees beginning at the base distinguished name. If the `nt_domain` is not configured, Hue searches for an exact distinguished name pattern for the user (defined by the `ldap_username_pattern` configuration setting). In this instance, the server searches for a matching `ldap_username_pattern` value in all directory subtrees beginning at the base distinguished name.

**To Launch an Amazon EMR cluster Using a Hue Configuration File for LDAP integration**

1. Type the following text in your `.json` configuration file to enable search bind for LDAP.

```json
{
    "hue": {
        "ldap": {
```

354
"ldap_servers": {
    "yourcompany": {
        "base_dn": "DC=yourcompany,DC=hue,DC=com",
        "ldap_url": "ldap://ldapurl",
        "search_bind_authentication": "true",
        "bind_dn": "CN=hue,CN=users,DC=yourcompany,DC=hue,DC=com",
        "bind_password": "password"
    }
}

Type the following text in your .json configuration file to enable direct bind for LDAP using nt_domain.

{
    "hue": {
        "ldap": {
            "ldap_servers": {
                "yourcompany": {
                    "base_dn": "DC=yourcompany,DC=hue,DC=com",
                    "ldap_url": "ldap://ldapurl",
                    "search_bind_authentication": "false",
                    "bind_dn": "CN=administrator,CN=users,DC=yourcompany,DC=hue,DC=com",
                    "bind_password": "password",
                    "nt_domain": "yourcompany.hue.com"
                }
            }
        }
    }
}

Type the following text in your .json configuration file to enable direct bind for LDAP using ldap_username_pattern.

{
    "hue": {
        "ldap": {
            "ldap_servers": {
                "yourcompany": {
                    "base_dn": "DC=yourcompany,DC=hue,DC=com",
                    "ldap_url": "ldap://ldapurl",
                    "search_bind_authentication": "false",
                    "bind_dn": "CN=admin,CN=users,DC=yourcompany,DC=hue,DC=com",
                    "bind_password": "password",
                    "ldap_username_pattern": "CN=username,OU=orgunit,DC=yourcompany,DC=hue,DC=com"
                }
            }
        }
    }
}

**Note**

If you are configuring LDAP direct bind with Hue, using ldap_username_pattern, and the distinguished name includes the common name attribute for the username, Hue requires you to log in using the common name (or full name in Active Directory). In this case, you cannot use sAMAccountName to log in; but when Hue is synchronizing groups and users, sAMAccountName is used to create users in Hue. This renders group synchronization ineffective.
Configure Hue for LDAP Users

Type the following text if you have nested groups and would like to synchronize all users in the subgroups when using direct bind for LDAP with `nt_domain`. This configuration uses the `subgroups` and `nested_members_search_depth` parameters.

```json
{
    "hue": {
        "ldap": {
            "ldap_servers": {
                "yourcompany": {
                    "base_dn": "DC=yourcompany,DC=hue,DC=com",
                    "ldap_url": "ldap://ldapurl",
                    "search_bind_authentication": "false",
                    "bind_dn": "hue",
                    "bind_password": "password",
                    "nt_domain": "yourcompany.hue.com"
                }
            },
            "subgroups": "nested",
            "nested_members_search_depth": 3
        }
    }
}
```

Type the following text to use search bind and to apply user and group filters when managing LDAP entities. This configuration uses the `groups` and `users` parameters.

```json
{
    "hue": {
        "ldap": {
            "ldap_servers": {
                "yourcompany": {
                    "base_dn": "DC=yourcompany,DC=hue,DC=com",
                    "ldap_url": "ldap://ldapurl",
                    "search_bind_authentication": "true",
                    "bind_dn": "CN=hue,CN=users,DC=yourcompany,DC=hue,DC=com",
                    "bind_password": "password",
                    "groups": {
                        "group_name_attr": "cn",
                        "group_filter": "objectclass=group"
                    },
                    "users": {
                        "user_filter": "objectclass=user",
                        "user_name_attr": "sAMAccountName"
                    }
                }
            }
        }
    }
}
```

2. Upload your Hue configuration (.json) file to your Amazon S3 bucket.

   **Important**

   Your .json configuration file should be stored in a secure Amazon S3 bucket to protect the configuration settings.

3. Type the following command in the AWS CLI to create an Amazon EMR cluster and apply the LDAP configuration settings in your .json configuration file. Replace `mybucket` with the name of your Amazon S3 bucket, replace `myKey` with the name of your EC2 key pair, and replace `hueconfig.json` with the path to and filename of your .json file.

   - Linux, UNIX, and Mac OS X users:
aws emr create-cluster --name "Hue cluster" --ami-version=3.11.0 --applications
Name=Hive \nName=Pig Name=Hue,Args=[--hue-config=s3://mybucket/hueconfig.json] \n--use-default-roles --ec2-attributes KeyName=myKey \n--instance-type m3.xlarge --instance-count 3

Windows users:

aws emr create-cluster --name "Hue cluster" --ami-version=3.11.0 --applications
Name=Hive Name=Pig Name=Hue,Args=[--hue-config=s3://mybucket/hueconfig.json] --use-default-roles --ec2-attributes KeyName=myKey --instance-type m3.xlarge --instance-count 3

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

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

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

To View LDAP Settings in Hue
1. Verify you have an active VPN connection or SSH tunnel to the Amazon EMR cluster's master node. Then, in your browser, type master-public-dns:8888 to open the Hue web interface.
2. Log in using your Hue administrator credentials. If the Did you know? window opens, click Got it, prof! to close it.
3. Click the Hue icon in the toolbar.
4. On the About Hue page, click Configuration.
5. In the Configuration Sections and Variables section, click Desktop.
6. Scroll to the ldap section to view your settings.

Metastore Manager Restrictions

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

The Metastore Manager default database exists in HDFS. You cannot import a table from Amazon S3 using a database stored in HDFS. To import a table from Amazon S3, create a new database, change the default location of the database to Amazon S3, create a table in the database, and then import your table from Amazon S3.
Analyze Amazon Kinesis Data

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

**Note**

Amazon Kinesis functionality in Amazon EMR is available to all public regions except China (Beijing).

Amazon EMR clusters can read and process Amazon Kinesis streams directly, using familiar tools in the Hadoop ecosystem such as Hive, Pig, MapReduce, the Hadoop Streaming API, and Cascading. You can also join real-time data from Amazon Kinesis with existing data on Amazon S3, Amazon DynamoDB, and HDFS in a running cluster. You can directly load the data from Amazon EMR to Amazon S3 or DynamoDB for post-processing activities. For information about Amazon Kinesis service highlights and pricing, see Amazon Kinesis.

What Can I Do With Amazon EMR and Amazon Kinesis Integration?

Integration between Amazon EMR and Amazon Kinesis makes certain scenarios much easier; for example:

- **Streaming log analysis**—You can analyze streaming web logs to generate a list of top 10 error types every few minutes by region, browser, and access domain.
- **Customer engagement**—You can write queries that join clickstream data from Amazon Kinesis with advertising campaign information stored in a DynamoDB table to identify the most effective categories of ads that are displayed on particular websites.
- **Ad-hoc interactive queries**—You can periodically load data from Amazon Kinesis streams into HDFS and make it available as a local Impala table for fast, interactive, analytic queries.

Checkpointed Analysis of Amazon Kinesis Streams

Users can run periodic, batched analysis of Amazon Kinesis streams in what are called *iterations*. Because Amazon Kinesis stream data records are retrieved by using a sequence number, iteration boundaries are defined by starting and ending sequence numbers that Amazon EMR stores in a DynamoDB table. For example, when iteration 0 ends, it stores the ending sequence number in DynamoDB so that when the iteration 1 job begins, it can retrieve subsequent data from the stream. This mapping of iterations in stream data is called *checkpointing*. For more information, see Kinesis Connector.

If an iteration was checkpointed and the job failed processing an iteration, Amazon EMR attempts to reprocess the records in that iteration, provided that the data records have not reached the 24-hour limit for Amazon Kinesis streams.

Checkpointing is a feature that allows you to:

- Start data processing after a sequence number processed by a previous query that ran on same stream and logical name
- Re-process the same batch of data from Kinesis that was processed by an earlier query
To enable checkpointing, set the `kinesis.checkpoint.enabled` parameter to `true` in your scripts. Also, configure the following parameters:

<table>
<thead>
<tr>
<th>Configuration Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>kinesis.checkpoint.metastore.table.name</code></td>
<td>DynamoDB table name where checkpoint information will be stored</td>
</tr>
<tr>
<td><code>kinesis.checkpoint.metastore.hash.key.name</code></td>
<td>Hash key name for the DynamoDB table</td>
</tr>
<tr>
<td><code>kinesis.checkpoint.metastore.hash.range.name</code></td>
<td>Range key name for the DynamoDB table</td>
</tr>
<tr>
<td><code>kinesis.checkpoint.logical.name</code></td>
<td>A logical name for current processing</td>
</tr>
<tr>
<td><code>kinesis.checkpoint.iteration.no</code></td>
<td>Iteration number for processing associated with the logical name</td>
</tr>
<tr>
<td><code>kinesis.rerun.iteration.without.wait</code></td>
<td>Boolean value that indicates if a failed iteration can be rerun without waiting for timeout; the default is false</td>
</tr>
</tbody>
</table>

### Provisioned IOPS Recommendations for Amazon DynamoDB Tables

The Amazon EMR connector for Amazon Kinesis uses the DynamoDB database as its backing for checkpointing metadata. You must create a table in DynamoDB before consuming data in an Amazon Kinesis stream with an Amazon EMR cluster in checkpointed intervals. The table must be in the same region as your Amazon EMR cluster. The following are general recommendations for the number of IOPS you should provision for your DynamoDB tables; let \( j \) be the maximum number of Hadoop jobs (with different logical name+iteration number combination) that can run concurrently and \( s \) be the maximum number of shards that any job will process:

- **For Read Capacity Units**: \( j \times s / 5 \)
- **For Write Capacity Units**: \( j \times s \)

### Performance Considerations

Amazon Kinesis shard throughput is directly proportional to the instance size of nodes in Amazon EMR clusters and record size in the stream. We recommend that you use m1.xlarge or larger instances on master and core nodes for production workloads.

### Schedule Amazon Kinesis Analysis with Amazon EMR Clusters

When you are analyzing data on an active Amazon Kinesis stream, limited by timeouts and a maximum duration for any iteration, it is important that you run the analysis frequently to gather periodic details from the stream. There are multiple ways to execute such scripts and queries at periodic intervals; we recommend using AWS Data Pipeline for recurrent tasks like these. For more information, see AWS Data Pipeline PigActivity and AWS Data Pipeline HiveActivity in the AWS Data Pipeline Developer Guide.
Tutorial: Analyzing Kinesis Streams with Amazon EMR and Hive

Note
Amazon Kinesis functionality in Amazon EMR is available to all public regions except China (Beijing).

This tutorial demonstrates how to use Amazon EMR to query and analyze incoming data from an Amazon Kinesis stream using Hive. The instructions in this tutorial include how to:

- Sign up for an AWS account
- Create an Amazon Kinesis stream
- Use the Kinesis publisher sample application to populate the stream with sample Apache web log data
- Create an interactive Amazon EMR cluster for use with Hive
- Connect to the cluster and perform operations on stream data using Hive

Note
The Log4J Appender for Amazon Kinesis currently only works with streams created in US East (N. Virginia) region.

Topics
- Sign Up for the Service (p. 360)
- Create an Amazon Kinesis Stream (p. 360)
- Create an Amazon DynamoDB Table (p. 361)
- Download Log4J Appender for Amazon Kinesis Sample Application, Sample Credentials File, and Sample Log File (p. 361)
- Start Amazon Kinesis Publisher Sample Application (p. 363)
- Launch the Cluster (p. 364)
- Run the Ad-hoc Hive Query (p. 368)
- Running Queries with Checkpoints (p. 370)
- Scheduling Scripted Queries (p. 371)

Sign Up for the Service

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

To sign up for AWS

2. Follow the online instructions.

AWS notifies you by email when your account is active and available for you to use. Your AWS account gives you access to all services, but you are charged only for the resources that you use. For this example walk-through, the charges will be minimal.

Create an Amazon Kinesis Stream

Before you create a Kinesis Data Streams stream, you must determine the size that you need the stream to be. For information about determining stream size, see How Do I Size an Amazon Kinesis Stream? in the Amazon Kinesis Developer Guide.
For more information about the endpoints available for Kinesis Data Streams, see Regions and Endpoints in the Amazon Web Services General Reference.

To create a stream

1. Open the Kinesis Data Streams console at https://console.aws.amazon.com/kinesis/.
   If you haven't yet signed up for the Amazon Kinesis service, you'll be prompted to sign up when you go to the console.
2. Select the US East (N. Virginia) Region in the region selector.
3. Choose Create Stream.
4. On the Create Stream page, provide a name for your stream (for example, AccessLogStream), specify 2 shards, and then choose Create.
   On the Stream List page, your stream's Status value is CREATING while the stream is being created. When the stream is ready to use, the Status value changes to ACTIVE.
5. Choose the name of your stream. The Stream Details page displays a summary of your stream configuration, along with monitoring information.

For information about how to create an Amazon Kinesis stream programmatically, see Using the Amazon Kinesis Service API in the Amazon Kinesis Developer Guide.

Create an Amazon DynamoDB Table

The Amazon EMR connector for Kinesis uses the DynamoDB database as its backing database for checkpointing. You must create a table in DynamoDB before consuming data in a Amazon Kinesis stream with an Amazon EMR cluster in checkpointed intervals.

   Note
   If you have completed any other tutorials that use the same DynamoDB table, you do not need to create the table again. However, you must clear that table's data before you use it for the checkpointing scripts in this tutorial.

To Create a Amazon DynamoDB Database for Use By the Amazon EMR Connector for Amazon Kinesis

1. Using the DynamoDB console in the same region as your Amazon EMR cluster, create a table with the name MyEMRKinesisTable.
2. For the Primary Key Type, choose Hash and Range.
3. For the Hash Attribute Name, use HashKey.
4. For the Range Attribute Name, use RangeKey.
5. Choose Continue.
6. You do not need to add any indexes for this tutorial. Choose Continue.
7. For Read Capacity Units and Write Capacity Units, use 10 IOPS for each.

Download Log4J Appender for Amazon Kinesis Sample Application, Sample Credentials File, and Sample Log File

The Amazon Kinesis Log4J Appender is an implementation of the Apache Log4J Appender Interface that will push Log4J output directly to a user specified Amazon Kinesis stream without requiring any custom code. The implementation uses the AWS SDK for Java APIs for Amazon Kinesis and is configurable
using the `log4j.properties` file. Users who would like to utilize the Amazon Kinesis Log4j Appender independent of this sample can download the jar file [here]. To simplify the steps in this tutorial, the sample app referenced below incorporates a JAR file and provides a default configuration for the appender. Users who would like to experiment with the full functionality of the publisher sample application can modify the `log4j.properties`. The configurable options are:

**log4j.properties Config Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>log4j.appender.KINESIS.streamName</code></td>
<td>AccessLogStream</td>
<td>Stream name to which data is to be published.</td>
</tr>
<tr>
<td><code>log4j.appender.KINESIS.encoding</code></td>
<td>UTF-8</td>
<td>Encoding used to convert log message strings into bytes before sending to Amazon Kinesis.</td>
</tr>
<tr>
<td><code>log4j.appender.KINESIS.maxRetries</code></td>
<td>3</td>
<td>Maximum number of retries when calling Kinesis APIs to publish a log message.</td>
</tr>
<tr>
<td><code>log4j.appender.KINESIS.backoffInterval</code></td>
<td>100ms</td>
<td>Milliseconds to wait before a retry attempt.</td>
</tr>
<tr>
<td><code>log4j.appender.KINESIS.threadCount</code></td>
<td>20</td>
<td>Number of parallel threads for publishing logs to configured Kinesis stream.</td>
</tr>
<tr>
<td><code>log4j.appender.KINESIS.bufferSize</code></td>
<td>2000</td>
<td>Maximum number of outstanding log messages to keep in memory.</td>
</tr>
<tr>
<td><code>log4j.appender.KINESIS.shutdownTimeout</code></td>
<td>30</td>
<td>Seconds to send buffered messages before application JVM quits normally.</td>
</tr>
</tbody>
</table>

The Amazon Kinesis publisher sample application is `kinesis-log4j-appender-1.0.0.jar` and requires Java 1.7 or later.

**To download and configure the Amazon Kinesis Log4j Appender tool:**

2. Create a file in the same folder where you downloaded `kinesis-log4j-appender-1.0.0.jar` called `AwsCredentials.properties`, and edit it with your credentials:

   ```
   accessKey=<your_access_key>
   secretKey=<your_secret_key>
   ```

   Replace `<your_access_key>` and `<your_secret_key>` with your `accessKey` and `secretKey` from your AWS account. For more information about access keys, see [How Do I Get Security Credentials?](https://aws.amazon.com/about-aws/whats-new/docs/general-reference/) in the AWS General Reference.

3. Download and save the sample access log file, `access_log_1`, from [http://elasticmapreduce.s3.amazonaws.com/samples/pig-apache/input/access_log_1](http://elasticmapreduce.s3.amazonaws.com/samples/pig-apache/input/access_log_1) in the same directory where you saved the credentials and JAR file.
4. **(Optional)** In the same directory, download log4j.properties from http://emr-kinesis.s3.amazonaws.com/publisher/log4j.properties and modify the settings according to your own applications needs.

## Start Amazon Kinesis Publisher Sample Application

The next step is to start the Amazon Kinesis publisher tool.

### To Start Amazon Kinesis Publisher for One-Time Publishing

1. In the same directory path where you have the JAR file, credentials, and log file, run the following from the command line:

   - Linux, UNIX, and Mac OS X users:
     ```
     $(JAVA_HOME)/bin/java -cp .:kinesis-log4j-appender-1.0.0.jar
     com.amazonaws.services.kinesis.log4j.FilePublisher access_log_1
     ```
   - Windows users:
     ```
     %JAVA_HOME%/bin/java -cp .;kinesis-log4j-appender-1.0.0.jar
     com.amazonaws.services.kinesis.log4j.FilePublisher access_log_1
     ```

2. Amazon Kinesis Publisher will upload each row of the log file to Amazon Kinesis until there are no rows remaining.

   ```
   [...]
   DEBUG [main] (FilePublisher.java:62) - 39100 records written
   DEBUG [main] (FilePublisher.java:62) - 39200 records written
   DEBUG [main] (FilePublisher.java:62) - 39300 records written
   INFO [main] (FilePublisher.java:66) - Finished publishing 39344 log events from access_log_1, took 229 secs to publish
   INFO [main] (FilePublisher.java:68) - DO NOT kill this process, publisher threads will keep on sending buffered logs to Amazon Kinesis
   ```

   **Note**
   The message "INFO [main] (FilePublisher.java:68) - DO NOT kill this process, publisher threads will keep on sending buffered logs to Kinesis" may appear after have published your records to the Amazon Kinesis stream. For the purposes of this tutorial, it is alright to kill this process once you have reached this message.

### To Start Amazon Kinesis Publisher for Continuous Publishing on Linux

If you want to simulate continuous publishing to your Amazon Kinesis stream for use with checkpointing scripts, follow these steps on Linux systems to run this shell script that loads the sample logs to your Amazon Kinesis stream every 400 seconds:

2. Make publisher.sh executable:

   ```
   % chmod +x publisher.sh
   ```
3. Create a log directory to which publisher.sh output redirects, /tmp/cronlogs:

   ```bash
   mkdir /tmp/cronlogs
   ```

4. Run publisher.sh with the following `nohup` command:

   ```bash
   nohup ./publisher.sh 1>>&2 /tmp/cronlogs/publisher.log 2>&1 &
   ```

5. **Important**
   This script will run indefinitely. Terminate the script to avoid further charges upon completion of the tutorial.

### To Start Amazon Kinesis Publisher for Continuous Publishing on Windows

If you want to simulate continuous publishing to your Amazon Kinesis stream for use with checkpointing scripts, follow these steps on Windows systems to run this batch script that loads the sample logs to your Amazon Kinesis stream every 400 seconds:

1. Download the file, `publisher.bat`, from http://emr-kinesis.s3.amazonaws.com/publisher/publisher.bat:
2. Run `publisher.bat` on the command prompt by typing it and pressing return. You can optionally open the file in Windows Explorer.
3. **Important**
   This script will run indefinitely. Terminate the script to avoid further charges upon completion of the tutorial.

### Launch the Cluster

The next step is to launch the cluster. This tutorial provides the steps to launch the cluster using both the Amazon EMR console and the Amazon EMR CLI. Choose the method that best meets your needs.

When you launch the cluster, Amazon EMR provisions EC2 instances (virtual servers) to perform the computation. These EC2 instances are preloaded with an Amazon Machine Image (AMI) that has been customized for Amazon EMR and which has Hadoop and other big data applications preloaded.

#### To launch a cluster for use with Kinesis using the console

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. Choose **Create cluster**.
3. On the **Create Cluster** page, in the **Cluster Configuration** section, verify the fields according to the following table.

<table>
<thead>
<tr>
<th>Field</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cluster name</strong></td>
<td>Enter a descriptive name for your cluster or leave the default name &quot;My cluster.&quot;</td>
</tr>
<tr>
<td></td>
<td>The name is optional, and does not need to be unique.</td>
</tr>
<tr>
<td><strong>Termination protection</strong></td>
<td>Leave the default option selected: <strong>Yes</strong>.</td>
</tr>
<tr>
<td></td>
<td>Enabling termination protection ensures that the cluster does not shut down due to accident or error. For more information, see Managing Cluster Termination (p. 474). Typically, you set this value to <strong>Yes</strong> when developing an application (so you can debug errors that would have</td>
</tr>
</tbody>
</table>
### Field | Action
--- | ---
otherwise terminated the cluster), to protect long-running clusters, or to preserve data. | Logging
This determines whether Amazon EMR captures detailed log data to Amazon S3. For more information, see View Log Files (p. 422).

Log folder S3 location
Type or browse to an Amazon S3 path to store your debug logs if you enabled logging in the previous field. You may also allow the console to generate an Amazon S3 path for you. If the log folder does not exist, the Amazon EMR console creates it. When Amazon S3 log archiving is enabled, Amazon EMR copies the log files from the EC2 instances in the cluster to Amazon S3. This prevents the log files from being lost when the cluster ends and the EC2 instances hosting the cluster are terminated. These logs are useful for troubleshooting purposes. For more information, see View Log Files (p. 422).

4. In the **Software Configuration** section, verify the fields according to the following table.

| Field | Action |
--- | --- |
Hadoop distribution | Choose Amazon. This determines which distribution of Hadoop to run on your cluster. You can choose to run the Amazon distribution of Hadoop or one of several MapR distributions. For more information, see Using the MapR Distribution for Hadoop (p. 177). |
AMI version | Choose 3.0.4 (Hadoop 2.2.0). For more information, see Choose an Amazon Machine Image (AMI) (p. 69). |

5. In the **Hardware Configuration** section, verify the fields according to the following table.

**Note**
Twenty is the default maximum number of nodes per AWS account. For example, if you have two clusters running, the total number of nodes running for both clusters must be 20 or less. Exceeding this limit results in cluster failures. If you need more than 20 nodes, you must submit a request to increase your Amazon EC2 instance limit. Ensure that your requested limit increase includes sufficient capacity for any temporary, unplanned increases in your needs. For more information, go to the Request to Increase Amazon EC2 Instance Limit Form.

| Field | Action |
--- | --- |
Network | Choose **Launch into EC2-Classic**. Optionally, choose a VPC subnet identifier from the list to launch the cluster in an Amazon VPC. For more information, see Plan and Configure Networking (p. 145). |
EC2 Availability Zone | Choose **No preference**. |
### Launch the Cluster

<table>
<thead>
<tr>
<th>Field</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optionally, you can launch the cluster in a specific Amazon EC2 Availability Zone.  For more information, see <a href="https://docs.aws.amazon.com/AmazonEC2/latest/UserGuide/using-regions-availability-zones.html">Regions and Availability Zones</a> in the <em>Amazon EC2 User Guide for Linux Instances</em>.</td>
<td></td>
</tr>
</tbody>
</table>
| **Master** | Choose **m1.large**.  
> The master node assigns Hadoop tasks to core and task nodes, and monitors their status. There is always one master node in each cluster.  
> This specifies the EC2 instance types to use as master nodes. Valid types are m1.large (default), m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cg1.4xlarge.  
> This tutorial uses m1.large instances for all nodes. |
| **Core** | Choose **m1.large**.  
> A core node is an EC2 instance that runs Hadoop map and reduce tasks and stores data using the Hadoop Distributed File System (HDFS). Core nodes are managed by the master node.  
> This specifies the EC2 instance types to use as core nodes. Valid types: m1.large (default), m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cg1.4xlarge.  
> This tutorial uses m1.large instances for all nodes. |
| **Task** | Choose **m1.large**.  
> Task nodes only process Hadoop tasks and don’t store data. You can add and remove them from a cluster to manage the EC2 instance capacity that your cluster uses, increasing capacity to handle peak loads and decreasing it later.  
> Task nodes only run a TaskTracker Hadoop daemon.  
> This specifies the EC2 instance types to use as task nodes. Valid types: m1.large (default), m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cg1.4xlarge. |
| **Count** | Choose **2**.  
| **Request Spot Instances** | Leave this box unchecked.  
> This specifies whether to run core nodes on Spot Instances. For more information, see [When Should You Use Spot Instances?](https://docs.aws.amazon.com/AmazonEC2/latest/UserGuide/using-spot-instances.html) (p. 164). |
| **Task** | Choose **m1.large**.  
> Task nodes only process Hadoop tasks and don’t store data. You can add and remove them from a cluster to manage the EC2 instance capacity that your cluster uses, increasing capacity to handle peak loads and decreasing it later.  
> Task nodes only run a TaskTracker Hadoop daemon.  
> This specifies the EC2 instance types to use as task nodes. Valid types: m1.large (default), m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cg1.4xlarge. |
| **Count** | Choose **0**.  
| **Request Spot Instances** | Leave this box unchecked.  
> This specifies whether to run task nodes on Spot Instances. For more information, see [When Should You Use Spot Instances?](https://docs.aws.amazon.com/AmazonEC2/latest/UserGuide/using-spot-instances.html) (p. 164). |
To create a cluster using the AWS CLI

- To launch your cluster, type the following command and replace myKey with the name of your EC2 key pair.

  ```bash
  aws emr create-cluster --name "EmrKinesisTutorial" --ami-version 3.3 --applications Name=Hue Name=Hive Name=Pig \  --use-default-roles --ec2-attributes KeyName=myKey \  --instance-type m3.xlarge --instance-count 3
  ```

- Windows users:
Run the Ad-hoc Hive Query

To run an ad-hoc Hive query

1. Connect to the master node of the cluster using SSH and run the commands shown in the following steps. Your client operating system determines which steps to use to connect to the cluster. For more information, see Connect to the Master Node Using SSH (p. 457).

2. In the SSH window, from the home directory, start the Hive shell by running the following command:

   ```
   ~/bin/hive
   ```

3. Run the following query to create a table `apachelog` by parsing the records in the Kinesis stream `AccessLogStream`:

   ```
   DROP TABLE apachelog;
   CREATE TABLE apachelog (host STRING, identity STRING, user STRING, time STRING, request STRING, status STRING, size STRING, referrer STRING, agent STRING)
   ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
   WITH SERDEPROPERTIES ("input.regex" = "((\^[^\s]*) ([^\s]*) ([^\s]*) (-|\\[^[\s]*\\]) ([^[\s]*|"^[\s]*") ([0-9]*) ([0-9]*) ([^[\s]*|"^[\s]*") ([^[\s]*|"^[\s]*")")
   STORED BY 'com.amazon.emr.kinesis.hive.KinesisStorageHandler'
   TBLPROPERTIES("kinesis.stream.name"="AccessLogStream");
   ```

   This query uses RegexSerde to parse the Apache web log format into individual columns. Note how this query specifies the Kinesis stream name.

4. Optional additional configurations can be specified as part of the table definition using the following additional lines, for example:
... STORED BY 'com.amazon.emr.kinesis.hive.KinesisStorageHandler'
TBLPROPERTIES( "kinesis.stream.name"="AccessLogStream",
"kinesis.accessKey"="AwsAccessKey",
"kinesis.secretKey"="AwsSecretKey",
"kinesis.nodata.timeout"="1",
"kinesis.iteration.timeout"="5",
"kinesis.iteration.batchsize"="1000",
"kinesis.endpoint.region"="us-east-1",
"kinesis.retry.interval"="1000",
"kinesis.retry.maxattempts"="3"
);

In addition, these optional properties can alternatively be set using global variables before firing the actual query:

... hive> SET kinesis.stream.name=AccessLogStream;
hive> SET kinesis.nodata.timeout=1;
hive> ...

**Note**
Values in these table properties always override the global configuration values.

The following table provides information about other configuration properties that you can set in the table definition, as well as global variables:

<table>
<thead>
<tr>
<th>Configuration Setting</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kinesis.stream.name</td>
<td></td>
<td>Kinesis stream name as the source of data.</td>
</tr>
<tr>
<td>kinesis.accessKey</td>
<td></td>
<td>Amazon S3 credentials set in the cluster or IAM-based role credentials if Amazon S3 credentials are absent.</td>
</tr>
<tr>
<td>kinesis.secretKey</td>
<td></td>
<td>Amazon S3 credentials set in the cluster or IAM-based role credentials if Amazon S3 credentials are absent.</td>
</tr>
<tr>
<td>kinesis.nodata.timeout</td>
<td>5</td>
<td>Timeout, in minutes (integer), to finish this iteration if no data is received continuously for this duration.</td>
</tr>
<tr>
<td>kinesis.iteration.timeout</td>
<td>15</td>
<td>Maximum duration in minutes (integer) to run this iteration. The cluster would create a checkpoint at the end.</td>
</tr>
</tbody>
</table>
### Configuration Settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kinesis.iteration.batchsize</td>
<td>1000</td>
<td>Number of records to get from the Kinesis stream in a single GetRecords API call. Cannot be more than 10000 (limit enforced by the Kinesis API).</td>
</tr>
<tr>
<td>kinesis.endpoint.region</td>
<td>us-east-1</td>
<td>Kinesis endpoint region. You may experience better performance by choosing a Kinesis region closest to the Amazon EMR cluster region.</td>
</tr>
<tr>
<td>kinesis.retry.interval</td>
<td>500</td>
<td>Retry interval in msec (integer) for a failure when calling the Kinesis API.</td>
</tr>
<tr>
<td>kinesis.retry.maxattempts</td>
<td>5</td>
<td>The maximum number of retries in case of a failure before giving up.</td>
</tr>
</tbody>
</table>

5. Run the following query to analyze the Hive table created in Step 3 (p. 368). This query counts number of visitors coming from Windows or Linux operating systems who got a 404 error:

```sql
SELECT OS, COUNT(*) AS COUNT
FROM (SELECT regexp_extract(agent,'.*(Windows|Linux).*',1) AS OS
      FROM apachelog WHERE STATUS=404
      ) X
WHERE OS IN ('Windows','Linux')
GROUP BY OS;
```

6. Check the output of the analysis query, which should look similar to the following:

7. **Important**
   Remember to terminate your cluster to avoid additional charges.

---

## Running Queries with Checkpoints

### Note

If you have completed any other tutorials that use the same DynamoDB table, you must clear that table data before you execute these commands.
You can process data in a running Kinesis stream and store the results in Amazon S3 using Hive's dynamic partitions and the previously-created table `apachelog`, as shown in the following example:

```
CREATE TABLE apachelog_s3 (os string, error_count int)
PARTITIONED BY(iteration_no int)
LOCATION 'my s3 location';

set kinesis.checkpoint.enabled=true;
set kinesis.checkpoint.metastore.table.name=MyEMRKinesisTable;
set kinesis.checkpoint.metastore.hash.key.name=HashKey;
set kinesis.checkpoint.metastore.range.key.name=RangeKey;
set kinesis.checkpoint.logical.name=TestLogicalName;
set kinesis.checkpoint.iteration.no=0;

--The following query will create OS-ERROR_COUNT result under dynamic partition for
--iteration no 0
INSERT OVERWRITE TABLE apachelog_s3 partition (iteration_no=
#{hiveconf:kinesis.checkpoint.iteration.no}) SELECT OS, COUNT(*) AS COUNT
FROM (
    SELECT regexp_extract(agent,'.*(Windows|Linux).*',1) AS OS
    FROM apachelog WHERE STATUS=404
) X
WHERE OS IN ('Windows','Linux')
GROUP BY OS;

set kinesis.rerun.iteration.without.wait=true;
set kinesis.checkpoint.iteration.no=1;

--The following query will create OS-ERROR_COUNT result under dynamic partition for
--iteration no 1
INSERT OVERWRITE TABLE apachelog_s3 partition (iteration_no=
#{hiveconf:kinesis.checkpoint.iteration.no}) SELECT OS, COUNT(*) AS COUNT
FROM (
    SELECT regexp_extract(agent,'.*(Windows|Linux).*',1) AS OS
    FROM apachelog WHERE STATUS=404
) X
WHERE OS IN ('Windows','Linux')
GROUP BY OS;
```

### Scheduling Scripted Queries

You can schedule scripts to run on your Hadoop cluster using the Linux **cron** system daemon on the master node. This is especially useful when processing Kinesis stream data at regular intervals.

#### To set up a cronjob for scheduled runs

1. Connect to your cluster's master node using SSH. For more information about connecting to your cluster, see [Connect to the Master Node Using SSH (p. 457)](https://docs.aws.amazon.com/emr/latest/UG/install-clients-ssh.html).
2. Create a directory for all your scheduling-related resources called `/home/hadoop/crontab`:

   ```bash
   % mkdir crontab
   ```

3. Download `executor.sh`, `hive.config`, `create_table.q`, and `user_agents_count.q` in `/home/hadoop/crontab` using the `wget` command:

   ```bash
   ```

371
Tutorial: Analyzing Amazon Kinesis Streams with Amazon EMR and Pig

Note
Amazon Kinesis functionality in Amazon EMR is available to all public regions except China (Beijing).

This tutorial demonstrates how to use Amazon EMR to query and analyze incoming data from a Amazon Kinesis stream using Pig. The instructions in this tutorial include how to:

- Sign up for an AWS account
- Create an Amazon Kinesis stream
- Use the Amazon Kinesis publisher sample application to populate the stream with sample Apache web log data
- Create an interactive Amazon EMR cluster for use with Pig
- Connect to the cluster and perform operations on Amazon Kinesis stream data using Pig
Topics

- Sign Up for the Service (p. 373)
- Create an Amazon Kinesis Stream (p. 373)
- Create an DynamoDB Table (p. 374)
- Download Log4J Appender for Amazon Kinesis Sample Application, Sample Credentials File, and Sample Log File (p. 374)
- Start Amazon Kinesis Publisher Sample Application (p. 375)
- Launch the Cluster (p. 377)
- Run the Pig Script (p. 381)
- Scheduling Scripted Queries (p. 384)

Sign Up for the Service

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

To sign up for AWS

2. Follow the online instructions.

AWS notifies you by email when your account is active and available for you to use. Your AWS account gives you access to all services, but you are charged only for the resources that you use. For this example walk-through, the charges will be minimal.

Create an Amazon Kinesis Stream

Before you create an Kinesis Data Streams stream, you must determine the size that you need the stream to be. For information about determining stream size, see How Do I Size an Amazon Kinesis Stream? in the Amazon Kinesis Developer Guide.

For more information about the endpoints available for Kinesis Data Streams, see Regions and Endpoints in the Amazon Web Services General Reference.

To create a stream

1. Open the Kinesis Data Streams console at https://console.aws.amazon.com/kinesis/.
   
   If you haven't yet signed up for the Amazon Kinesis service, you'll be prompted to sign up when you go to the console.
2. Select the US East (N. Virginia) Region in the region selector.
3. Choose Create Stream.
4. On the Create Stream page, provide a name for your stream (for example, AccessLogStream), specify 2 shards, and then choose Create.
   
   On the Stream List page, your stream's Status value is CREATING while the stream is being created. When the stream is ready to use, the Status value changes to ACTIVE.
5. Choose the name of your stream. The Stream Details page displays a summary of your stream configuration, along with monitoring information.

For information about how to create an Amazon Kinesis stream programmatically, see Using the Amazon Kinesis Service API in the Amazon Kinesis Developer Guide.
Create an DynamoDB Table

The Amazon EMR connector for Kinesis uses the DynamoDB database as its backing database for checkpointing. You must create a table in DynamoDB before consuming data in a Amazon Kinesis stream with an Amazon EMR cluster in checkpointed intervals.

**Note**
If you have completed any other tutorials that use the same DynamoDB table, you do not need to create the table again. However, you must clear that table's data before you use it for the checkpointing scripts in this tutorial.

**To Create a Amazon DynamoDB Database for Use By the Amazon EMR Connector for Amazon Kinesis**

1. Using the DynamoDB console in the same region as your Amazon EMR cluster, create a table with the name *MyEMRKinesisTable*.
2. For the **Primary Key Type**, choose **Hash and Range**.
3. For the **Hash Attribute Name**, use **HashKey**.
4. For the **Range Attribute Name**, use **RangeKey**.
5. Choose **Continue**.
6. You do not need to add any indexes for this tutorial. Choose **Continue**.
7. For **Read Capacity Units** and **Write Capacity Units**, use 10 IOPS for each.

Download Log4J Appender for Amazon Kinesis Sample Application, Sample Credentials File, and Sample Log File

The Amazon Kinesis Log4J Appender is an implementation of the Apache Log4J Appender Interface that will push Log4J output directly to a user specified Amazon Kinesis stream without requiring any custom code. The implementation uses the AWS SDK for Java APIs for Amazon Kinesis and is configurable using the `log4j.properties` file. Users who would like to utilize the Amazon Kinesis Log4J Appender independent of this sample can download the jar file [here](#). To simplify the steps in this tutorial, the sample app referenced below incorporates a JAR file and provides a default configuration for the appender. Users who would like to experiment with the full functionality of the publisher sample application can modify the `log4j.properties`. The configurable options are:

**log4j.properties Config Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>log4j.appender.KINESIS.streamName</td>
<td>AccessLogStream</td>
<td>Stream name to which data is to be published.</td>
</tr>
<tr>
<td>log4j.appender.KINESIS.encoding</td>
<td>UTF-8</td>
<td>Encoding used to convert log message strings into bytes before sending to Amazon Kinesis.</td>
</tr>
<tr>
<td>log4j.appender.KINESIS.maxRetries</td>
<td>3</td>
<td>Maximum number of retries when calling Kinesis APIs to publish a log message.</td>
</tr>
</tbody>
</table>
The Amazon Kinesis publisher sample application is `kinesis-log4j-appender-1.0.0.jar` and requires Java 1.7 or later.

**To download and configure the Amazon Kinesis Log4j Appender tool:**

2. Create a file in the same folder where you downloaded `kinesis-log4j-appender-1.0.0.jar` called `AwsCredentials.properties`, and edit it with your credentials:

   ```
   accessKey=<your_access_key>
   secretKey=<your_secret_key>
   ```

   Replace `<your_access_key>` and `<your_secret_key>` with your accessKey and secretKey from your AWS account. For more information about access keys, see [How Do I Get Security Credentials?](http://aws.amazon.com/general-reference/) in the AWS General Reference.
3. Download and save the sample access log file, `access_log_1`, from [http://elasticmapreduce.s3.amazonaws.com/samples/pig-apache/input/access_log_1](http://elasticmapreduce.s3.amazonaws.com/samples/pig-apache/input/access_log_1) in the same directory where you saved the credentials and JAR file.
4. *(Optional)* In the same directory, download `log4j.properties` from [http://emr-kinesis.s3.amazonaws.com/publisher/log4j.properties](http://emr-kinesis.s3.amazonaws.com/publisher/log4j.properties) and modify the settings according to your own applications needs.

**Start Amazon Kinesis Publisher Sample Application**

The next step is to start the Amazon Kinesis publisher tool.

**To Start Amazon Kinesis Publisher for One-Time Publishing**

1. In the same directory path where you have the JAR file, credentials, and log file, run the following from the command line:

   - Linux, UNIX, and Mac OS X users:

   ```
   $(JAVA_HOME)/bin/java -cp ./kinesis-log4j-appender-1.0.0.jar com.amazonaws.services.kinesis.log4j.FilePublisher access_log_1
   ```
To Start Amazon Kinesis Publisher Sample Application

- **Windows users:**

  ```
  %JAVA_HOME%/bin/java -cp .;kinesis-log4j-appender-1.0.0.jar
  com.amazonaws.services.kinesis.log4j.FilePublisher access_log_1
  ```

2. Amazon Kinesis Publisher will upload each row of the log file to Amazon Kinesis until there are no rows remaining.

   ```
   [...
   DEBUG [main] (FilePublisher.java:62) - 39100 records written
   DEBUG [main] (FilePublisher.java:62) - 39200 records written
   DEBUG [main] (FilePublisher.java:62) - 39300 records written
   INFO [main] (FilePublisher.java:66) - Finished publishing 39344 log events from access_log_1, took 229 secs to publish
   INFO [main] (FilePublisher.java:68) - DO NOT kill this process, publisher threads will keep on sending buffered logs to Amazon Kinesis
   ```

**Note**

The message "INFO [main] (FilePublisher.java:68) - DO NOT kill this process, publisher threads will keep on sending buffered logs to Kinesis" may appear after you have published your records to the Amazon Kinesis stream. For the purposes of this tutorial, it is alright to kill this process once you have reached this message.

**To Start Amazon Kinesis Publisher for Continuous Publishing on Linux**

If you want to simulate continuous publishing to your Amazon Kinesis stream for use with checkpointing scripts, follow these steps on Linux systems to run this shell script that loads the sample logs to your Amazon Kinesis stream every 400 seconds:


2. Make `publisher.sh` executable:

   ```
   % chmod +x publisher.sh
   ```

3. Create a log directory to which `publisher.sh` output redirects, `/tmp/cronlogs`:

   ```
   % mkdir /tmp/cronlogs
   ```

4. Run `publisher.sh` with the following `nohup` command:

   ```
   % nohup ./publisher.sh 1>>/tmp/cronlogs/publisher.log 2>>/tmp/cronlogs/publisher.log &
   ```

5. **Important**

   This script will run indefinitely. Terminate the script to avoid further charges upon completion of the tutorial.

**To Start Amazon Kinesis Publisher for Continuous Publishing on Windows**

If you want to simulate continuous publishing to your Amazon Kinesis stream for use with checkpointing scripts, follow these steps on Windows systems to run this batch script that loads the sample logs to your Amazon Kinesis stream every 400 seconds:
1. Download the file, `publisher.bat`, from http://emr-kinesis.s3.amazonaws.com/publisher/publisher.bat:

2. Run `publisher.bat` on the command prompt by typing it and pressing return. You can optionally open the file in Windows Explorer.

3. **Important**
   This script will run indefinitely. Terminate the script to avoid further charges upon completion of the tutorial.

# Launch the Cluster

The next step is to launch the cluster. This tutorial provides the steps to launch the cluster using both the Amazon EMR console and the Amazon EMR CLI. Choose the method that best meets your needs. When you launch the cluster, Amazon EMR provisions EC2 instances (virtual servers) to perform the computation. These EC2 instances are preloaded with an Amazon Machine Image (AMI) that has been customized for Amazon EMR and which has Hadoop and other big data applications preloaded.

## To launch a cluster for use with Amazon Kinesis using the console

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. Choose Create cluster.
3. On the Create Cluster page, in the Cluster Configuration section, verify the fields according to the following table.

<table>
<thead>
<tr>
<th>Field</th>
<th>Action</th>
</tr>
</thead>
</table>
| **Cluster name**              | Enter a descriptive name for your cluster or leave the default name "My cluster."  
                                 | The name is optional, and does not need to be unique.                                                                                   |
| **Termination protection**    | Leave the default option selected: Yes.  
                                 | Enabling termination protection ensures that the cluster does not shut down due to accident or error. For more information, see Managing Cluster Termination (p. 474) . Typically, you set this value to Yes when developing an application (so you can debug errors that would have otherwise terminated the cluster), to protect long-running clusters, or to preserve data. |
| **Logging**                   | This determines whether Amazon EMR captures detailed log data to Amazon S3.  
                                 | For more information, see View Log Files (p. 422) .                                                                                     |
| **Log folder S3 location**    | Type or browse to an Amazon S3 path to store your debug logs if you enabled logging in the previous field. You may also allow the console to generate an Amazon S3 path for you. If the log folder does not exist, the Amazon EMR console creates it.  
                                 | When Amazon S3 log archiving is enabled, Amazon EMR copies the log files from the EC2 instances in the cluster to Amazon S3. This prevents the log files from being lost when the cluster ends and the EC2 instances hosting the cluster are terminated. These logs are useful for troubleshooting purposes. |
4. In the **Software Configuration** section, verify the fields according to the following table.

<table>
<thead>
<tr>
<th>Field</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hadoop distribution</td>
<td>Choose Amazon. This determines which distribution of Hadoop to run on your cluster. You can choose to run the Amazon distribution of Hadoop or one of several MapR distributions.</td>
</tr>
<tr>
<td>AMI version</td>
<td>Choose emr-4.0.0. The connector for Kinesis comes with this Amazon EMR AMI version or newer. For more information, see Choose an Amazon Machine Image (AMI) (p. 69).</td>
</tr>
</tbody>
</table>

5. In the **Hardware Configuration** section, verify the fields according to the following table.

**Note**
Twenty is the default maximum number of nodes per AWS account. For example, if you have two clusters running, the total number of nodes running for both clusters must be 20 or less. Exceeding this limit results in cluster failures. If you need more than 20 nodes, you must submit a request to increase your Amazon EC2 instance limit. Ensure that your requested limit increase includes sufficient capacity for any temporary, unplanned increases in your needs. For more information, go to the Request to Increase Amazon EC2 Instance Limit Form.

<table>
<thead>
<tr>
<th>Field</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network</td>
<td>Choose Launch into EC2-Classic. Optionally, choose a VPC subnet identifier from the list to launch the cluster in an Amazon VPC. For more information, see Plan and Configure Networking (p. 145).</td>
</tr>
<tr>
<td>EC2 Availability Zone</td>
<td>Choose No preference. Optionally, you can launch the cluster in a specific Amazon EC2 Availability Zone. For more information, see Regions and Availability Zones in the Amazon EC2 User Guide for Linux Instances.</td>
</tr>
<tr>
<td>Master</td>
<td>Choose m1.large. The master node assigns Hadoop tasks to core and task nodes, and monitors their status. There is always one master node in each cluster. This specifies the EC2 instance types to use as master nodes. Valid types are m1.large (default), m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cg1.4xlarge. This tutorial uses m1.large instances for all nodes.</td>
</tr>
</tbody>
</table>
### Field | Action
--- | ---
| **Core** | **Choose m1.large.**  
A core node is an EC2 instance that runs Hadoop map and reduce tasks and stores data using the Hadoop Distributed File System (HDFS). Core nodes are managed by the master node.  
This specifies the EC2 instance types to use as core nodes. Valid types: m1.large (default), m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cg1.4xlarge.  
This tutorial uses m1.large instances for all nodes.  
For more information, see *Create a Cluster with Instance Fleets or Uniform Instance Groups* (p. 153)
| Count | Choose **2**.
| **Task** | **Choose m1.large.**  
Task nodes only process Hadoop tasks and don’t store data. You can add and remove them from a cluster to manage the EC2 instance capacity that your cluster uses, increasing capacity to handle peak loads and decreasing it later.  
Task nodes only run a TaskTracker Hadoop daemon.  
This specifies the EC2 instance types to use as task nodes. Valid types: m1.large (default), m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cg1.4xlarge.  
For more information, see *Create a Cluster with Instance Fleets or Uniform Instance Groups* (p. 153)
| Count | Choose **0**.
| Request Spot Instances | Leave this box unchecked.  
This specifies whether to run core nodes on Spot Instances. For more information, see *When Should You Use Spot Instances?* (p. 164)
| Request Spot Instances | Leave this box unchecked.  
This specifies whether to run master nodes on Spot Instances. For more information, see *Create a Cluster with Instance Fleets or Uniform Instance Groups* (p. 153)

**Note**  
To save costs, we recommend using **m1.large** instance types for this tutorial. For production workloads, we recommend at least **m1.xlarge** instance types.

6. In the **Security and Access** section, complete the fields according to the following table.
### Field | Action
--- | ---
EC2 key pair | Choose your Amazon EC2 key pair private key from the list. Optionally, choose **Proceed without an EC2 key pair**. If you do not enter a value in this field, you cannot use SSH to connect to the master node. For more information, see [Connect to the Master Node Using SSH](p. 457).

IAM user access | Choose **All other IAM users** to make the cluster visible and accessible to all IAM users on the AWS account. For more information, see [Use IAM Policies to Allow and Deny User Permissions](p. 187).
Alternatively, choose **No other IAM users** to restrict access to the current IAM user.

Roles configuration | Choose **Default** to generate the default Amazon EMR role and Amazon EC2 instance profile. If the default roles exist, they are used for your cluster. If they do not exist, they are created (assuming you have proper permissions). You may also choose **View policies for default roles** to view the default role properties. Alternatively, if you have custom roles, you can choose **Custom** and choose your roles. An Amazon EMR role and Amazon EC2 instance profile are required when creating a cluster using the console.

The role allows Amazon EMR to access other AWS services on your behalf. The Amazon EC2 instance profile controls application access to the Amazon EC2 instances in the cluster.

For more information, see [Configure IAM Roles for Amazon EMR Permissions to AWS Services](p. 234).

7. In the **Bootstrap Actions** and **Steps** sections, you do not need to change any of these settings.
8. Review your configuration and if you are satisfied with the settings, choose **Create Cluster**.
9. When the cluster starts, the console displays the **Cluster Details** page.

### To create a cluster using the AWS CLI

- **To launch your cluster**, type the following command and replace `myKey` with the name of your EC2 key pair.

  - Linux, UNIX, and Mac OS X users:

    ```bash
    aws emr create-cluster --name "EmrKinesisTutorial" --ami-version 3.3 --applications Name=Hue Name=Hive Name=Pig \ --use-default-roles --ec2-attributes KeyName=myKey \ --instance-type m3.xlarge --instance-count 3
    ```

  - Windows users:

    ```bash
    aws emr create-cluster --name "EmrKinesisTutorial" --ami-version 3.3 --applications Name=Hue Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey --instance-type m3.xlarge --instance-count 3
    ```

When you specify the instance count without using the `--instance-groups` parameter, a single Master node is launched, and the remaining instances are launched as core nodes. All nodes will use the instance type specified in the command.
Note
If you have not previously created the default EMR service role and EC2 instance profile, type `aws emr create-default-roles` to create them before typing the `create-cluster` subcommand.

Run the Pig Script

To run a Pig script over a Kinesis stream in the Grunt shell

1. Connect to the master node of the cluster using SSH and run the commands shown in the following steps. Your client operating system determines which steps to use to connect to the cluster. For more information, see Connect to the Master Node Using SSH (p. 457).

2. In the SSH window, from the home directory, start the Grunt shell by running the following command:

   ```bash
   ~/bin/pig
   ```

3. Run the following script to process the data in the Kinesis stream, `AccessLogStream`, and print the agent count by operating system:

   ```pig
   REGISTER file:/home/hadoop/pig/lib/piggybank.jar;
   DEFINE EXTRACT org.apache.pig.piggybank.evaluation.string.EXTRACT();
   DEFINE REGEX_EXTRACT org.apache.pig.piggybank.evaluation.string.RegexExtract();

   raw_logs = load 'AccessLogStream' using
   com.amazon.emr.kinesis.pig.KinesisStreamLoader('kinesis.iteration.timeout=1') as
   (line:chararray);
   logs_base =
   -- for each weblog string convert the weblong string into a
   -- structure with named fields
   FOREACH
   raw_logs
   GENERATE
   FLATTEN(
     EXTRACT(
       line,
       "^((\S+) (\S+) (\S+) \[(\[\w:/\]+\s[+\-]\d{4})\]\) "(\S+) (\S+)"
     )
   )
   AS (host: chararray, identity: chararray, user: chararray, time: chararray,
   request: chararray, status: int, size: chararray, referrer: chararray,
   agent: chararray)
   )
   by_agent_count_raw =
   -- group by the referrer URL and count the number of requests
   FOREACH
   (GROUP logs_base BY REGEX_EXTRACT(agent,'.*(Windows|Linux).*',1))
   GENERATE
   FLATTEN($0),
   COUNT($1) AS agent_count
   ;

   by_agent_count = FILTER by_agent_count_raw by $0 IS NOT null OR ($0='');
   dump by_agent_count;
   ```
A list of agent operating systems and associated counts are printed. Results should contain values similar to the following:

(Linux, 707)
(Windows, 8328)

4. The following table provides information about other configuration properties that you can set in the table definition, as well as global variables:

<table>
<thead>
<tr>
<th>Configuration Setting</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kinesis.stream.name</td>
<td></td>
<td>Kinesis stream name as the source of data.</td>
</tr>
<tr>
<td>kinesis.accessKey</td>
<td></td>
<td>AWS access key.</td>
</tr>
<tr>
<td>kinesis.secretKey</td>
<td></td>
<td>AWS secret key.</td>
</tr>
<tr>
<td>kinesis.nodata.timeout</td>
<td>5</td>
<td>Timeout, in minutes (integer), to finish this iteration if no data is received continuously for this duration.</td>
</tr>
<tr>
<td>kinesis.iteration.timeout</td>
<td>15</td>
<td>Maximum duration in minutes (integer) to run this iteration. The cluster would create a checkpoint at the end.</td>
</tr>
<tr>
<td>kinesis.iteration.batchsize</td>
<td>1000</td>
<td>Number of records to get from the Kinesis stream in a single GetRecords API call. Cannot be more than 10000 (limit enforced by the Kinesis API).</td>
</tr>
<tr>
<td>kinesis.endpoint.region</td>
<td>us-east-1</td>
<td>Kinesis endpoint region. You may experience better performance by choosing a Kinesis region closest to the Amazon EMR cluster region.</td>
</tr>
<tr>
<td>kinesis.retry.interval</td>
<td>500</td>
<td>Retry interval in msec (integer) for a failure when calling the Kinesis API.</td>
</tr>
<tr>
<td>kinesis.retry.maxattempts</td>
<td>5</td>
<td>The maximum number of retries in case of a failure before giving up.</td>
</tr>
</tbody>
</table>

You can set these values in the script using the set command; for example, set kinesis.nodata.timeout 5;.
To run queries with checkpoints

1. **Note**
   If you have completed any other tutorials that use the same DynamoDB metastore, you must clear that table data before you execute these commands.

You can process data in a running Kinesis stream and store the results in Amazon S3. Run the script as shown in the Grunt shell:

```pig
REGISTER file:/home/hadoop/pig/lib/piggybank.jar;
DEFINE EXTRACT org.apache.pig.piggybank.evaluation.string.EXTRACT();
DEFINE REGEX_EXTRACT org.apache.pig.piggybank.evaluation.string.RegexExtract();
raw_logs = LOAD 'AccessLogStream' USING com.amazon.emr.kinesis.pig.KinesisStreamLoader() AS (line:chararray);
logs_base = -- for each weblog string convert the weblog string into a
-- structure with named fields
FOREACH raw_logs
  GENERATE
  FLATTEN (EXTRACT(
    line,
    '^(\S+) (\S+) (\S+) \[(\[\w:/\]+\s[+-]\d{4})\] "(.+?)" (\S+) (\S+)"
  ))
by_agent_count_raw = -- group by the referrer URL and count the number of requests
FOREACH (GROUP logs_base BY REGEX_EXTRACT(agent,'.*(Windows|Linux).*',1))
  GENERATE
  FLATTEN($0),
  COUNT($1) AS agent_count;
by_agent_count = FILTER by_agent_count_raw BY $0 IS NOT null OR ($0 != '');
-- Set checkpointing related parameters
set kinesis.checkpoint.enabled true;
set kinesis.checkpoint.metastore.table.name MyEMRKinesisTable;
set kinesis.checkpoint.metastore.hash.key.name HashKey;
set kinesis.checkpoint.metastore.range.key.name RangeKey;
set kinesis.checkpoint.logical.name TestLogicalName;
set kinesis.checkpoint.iteration.no 0;
STORE by_agent_count into 's3://my_s3_path/iteration_0';
```

2. Wait until the first iteration completes, then enter the following command:

```pig
set kinesis.rerun.iteration.without.wait true;
set kinesis.checkpoint.iteration.no 1;
STORE by_agent_count into 's3://my_s3_path/iteration_1';
```

3. Check the files in Amazon S3. The file in `iteration0` should contain values similar to the following:
The file in iteration1 should contain values similar to the following:

<table>
<thead>
<tr>
<th>Platform</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td>73</td>
</tr>
<tr>
<td>Windows</td>
<td>506</td>
</tr>
</tbody>
</table>

**Important**

Remember to terminate your cluster to avoid additional charges.

**Scheduling Scripted Queries**

You can schedule scripts to run on your Hadoop cluster using the Linux `cron` system daemon on the master node. This is especially useful when processing Kinesis stream data at regular intervals.

**To set up a cronjob for scheduled runs**

1. Connect to your cluster's master node using SSH. For more information about connecting to your cluster, see Connect to the Master Node Using SSH (p. 457).
2. Create a directory for all your scheduling-related resources called `/home/hadoop/crontab`:
   ```bash
   % mkdir crontab
   ```
3. Download executor.sh, pig.config, and user_agent_counts.pig from http://emr-kinesis.s3.amazonaws.com. Put the following in `/home/hadoop/crontab` using the `wget` command:
   ```bash
   ```
4. Edit `pig.config` to replace the value of the `SCRIPTS` variable with the full path of the script. If the directory you created in Step 2 is `/home/hadoop/crontab`, you do not need to do anything.
   ```bash
   SCRIPTS="/home/hadoop/crontab/pigscript1 /home/hadoop/crontab/pigscript2"
   ```
5. Edit `user_agents_count.pig`. At the end of the script, there is a `STORE` operator:
   ```pig
   STORE by_agent_count into 's3://<s3_output_path>/pig/iteration_$iterationNumber';
   ```
   Replace `<s3_output_path>` with your Amazon S3 bucket. Save and exit the editor.
6. Create a temporary directory, `/tmp/cronlogs`, for storing the log output generated by the cronjobs:
   ```bash
   mkdir /tmp/cronlogs
   ```
7. Make `executor.sh` executable:
% chmod +x executor.sh

8. Edit your crontab by executing `crontab -e` and inserting the following line in the editor:

```
*/15 * * * * /home/hadoop/crontab/executor.sh /home/hadoop/crontab/pig.config 1>>/tmp/cronlogs/pig.log 2>>/tmp/cronlogs/pig.log
```

Save and exit the editor; the crontab is updated upon exit. You can verify the crontab entries by executing `crontab -l`. 
Extract, Transform, and Load (ETL) Data with Amazon EMR

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Amazon EMR provides tools you can use to move data and to transform the data from one format to another. S3DistCp is a custom implementation of Apache DistCp that is optimized to work with Amazon S3. Using S3DistCp, you can efficiently copy a large amount of data from Amazon S3 into the HDFS datastore of your cluster. The implementation of Hive provided by Amazon EMR (version 0.7.1.1 and later) includes libraries you can use to import data from DynamoDB or move data from Amazon S3 to DynamoDB.

**Topics**

- S3DistCp (s3-dist-cp) (p. 386)
- Export, Import, Query, and Join Tables in DynamoDB Using Amazon EMR (p. 393)
- Store Avro Data in Amazon S3 Using Amazon EMR (p. 414)

S3DistCp (s3-dist-cp)

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Apache DistCp is an open-source tool you can use to copy large amounts of data. S3DistCp is an extension of DistCp that is optimized to work with AWS, particularly Amazon S3. The command for S3DistCp in Amazon EMR version 4.0 and later is `s3-dist-cp`, which you add as a step in a cluster or at the command line. Using S3DistCp, you can efficiently copy large amounts of data from Amazon S3 into HDFS where it can be processed by subsequent steps in your Amazon EMR cluster. You can also use S3DistCp to copy data between Amazon S3 buckets or from HDFS to Amazon S3. S3DistCp is more scalable and efficient for parallel copying large numbers of objects across buckets and across AWS accounts.

For specific commands that demonstrate the flexibility of S3DistCp in real-world scenarios, see [Seven Tips for Using S3DistCp](https://aws.amazon.com/blogs/big-data/seven-tips-for-using-s3distcp/) on the AWS Big Data blog.

Like DistCp, S3DistCp uses MapReduce to copy in a distributed manner, sharing the copy, error handling, recovery, and reporting tasks across several servers. For more information about the Apache DistCp open source project, see the [DistCp Guide](https://hadoop.apache.org/docs) in Apache Hadoop documentation.

During a copy operation, S3DistCp stages a temporary copy of the output in HDFS on the cluster. There must be enough free space in HDFS to stage the data, otherwise the copy operation fails. In addition, if S3DistCp fails, it does not clean the temporary HDFS directory, so you must purge the temporary files manually. For example, if you copy 500 GB of data from HDFS to S3, S3DistCp copies the entire 500 GB into a temporary directory in HDFS, then uploads the data to Amazon S3 from the temporary directory.
When the copy is complete, S3DistCp removes the files from the temporary directory. If you only have 250 GB of space remaining in HDFS prior to the copy, the copy operation fails.

If S3DistCp is unable to copy some or all of the specified files, the cluster step fails and returns a non-zero error code. If this occurs, S3DistCp does not clean up partially copied files.

Important
S3DistCp does not support Amazon S3 bucket names that contain the underscore character.

S3DistCp Options

When you call S3DistCp, you can specify options that change how it copies and compresses data. These are described in the following table. The options are added to the step using the arguments list. Examples of the S3DistCp arguments are shown in the following table.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>--src=LOCATION</td>
<td>Location of the data to copy. This can be either an HDFS or Amazon S3 location.</td>
<td>Yes</td>
</tr>
<tr>
<td>Example: --src=s3://myawsbucket/logs/j-3GYXXXXX9IOJ/node</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Important</td>
<td>S3DistCp does not support Amazon S3 bucket names that contain the underscore character.</td>
<td></td>
</tr>
<tr>
<td>--dest=LOCATION</td>
<td>Destination for the data. This can be either an HDFS or Amazon S3 location.</td>
<td>Yes</td>
</tr>
<tr>
<td>Example: --dest=hdfs:///output</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Important</td>
<td>S3DistCp does not support Amazon S3 bucket names that contain the underscore character.</td>
<td></td>
</tr>
<tr>
<td>--srcPattern=PATTERN</td>
<td>A regular expression that filters the copy operation to a subset of the data at --src. If neither --srcPattern nor --groupBy is specified, all data at --src is copied to --dest.</td>
<td>No</td>
</tr>
<tr>
<td>Example: --srcPattern=.<em>daemons.</em>-hadoop.*</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>If the regular expression argument contains special characters, such as an asterisk (*), either the regular expression or the entire --args string must be enclosed in single quotes (').</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example: --srcPattern=.<em>daemons.</em>-hadoop-*</td>
<td></td>
</tr>
<tr>
<td>--groupBy=_PATTERN</td>
<td>A regular expression that causes S3DistCp to concatenate files that match the expression. For example, you could use this option to combine all of the log files written in one hour into a single file. The concatenated filename is the value matched by the regular expression for the grouping.</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Parentheses indicate how files should be grouped, with all of the items that match the parenthetical statement being combined into a single output file. If the regular expression does not include a parenthetical</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
<td>Required</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td></td>
<td>statement, the cluster fails on the S3DistCp step and return an error. If the regular expression argument contains special characters, such as an asterisk (<em>), either the regular expression or the entire --args string must be enclosed in single quotes ('). When --groupBy is specified, only files that match the specified pattern are copied. You do not need to specify --groupBy and --srcPattern at the same time. Example: --groupBy=.<em>subnetid.</em>([0-9]+-[0-9]+-[0-9]+-[0-9]+).</em></td>
<td></td>
</tr>
<tr>
<td>--targetSize=SIZE</td>
<td>The size, in mebibytes (MiB), of the files to create based on the --groupBy option. This value must be an integer. When --targetSize is set, S3DistCp attempts to match this size; the actual size of the copied files may be larger or smaller than this value. Jobs are aggregated based on the size of the data file, thus it is possible that the target file size will match the source data file size. If the files concatenated by --groupBy are larger than the value of --targetSize, they are broken up into part files, and named sequentially with a numeric value appended to the end. For example, a file concatenated into myfile.gz would be broken into parts as myfile0.gz, myfile1.gz, etc. Example: --targetSize=2</td>
<td>No</td>
</tr>
<tr>
<td>--appendToLastFile</td>
<td>Specifies the behavior of S3DistCp when copying to files from Amazon S3 to HDFS which are already present. It appends new file data to existing files. If you use --appendToLastFile with --groupBy, new data is appended to files which match the same groups. This option also respects the --targetSize behavior when used with --groupBy.</td>
<td>No</td>
</tr>
<tr>
<td>--outputCodec=CODEC</td>
<td>Specifies the compression codec to use for the copied files. This can take the values: gzip, gz, lzo, snappy, or none. You can use this option, for example, to convert input files compressed with Gzip into output files with LZO compression, or to uncompress the files as part of the copy operation. If you choose an output codec, the filename will be appended with the appropriate extension (e.g. for gz and gzip, the extension is .gz) If you do not specify a value for --outputCodec, the files are copied over with no change in their compression. Example: --outputCodec=lzo</td>
<td>No</td>
</tr>
</tbody>
</table>
## S3DistCp Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>--s3ServerSideEncryption</td>
<td>Ensures that the target data is transferred using SSL and automatically encrypted in Amazon S3 using an AWS service-side key. When retrieving data using S3DistCp, the objects are automatically unencrypted. If you attempt to copy an unencrypted object to an encryption-required Amazon S3 bucket, the operation fails. For more information, see Using Data Encryption. Example: --s3ServerSideEncryption</td>
<td>No</td>
</tr>
<tr>
<td>--deleteOnSuccess</td>
<td>If the copy operation is successful, this option causes S3DistCp to delete the copied files from the source location. This is useful if you are copying output files, such as log files, from one location to another as a scheduled task, and you don't want to copy the same files twice. Example: --deleteOnSuccess</td>
<td>No</td>
</tr>
<tr>
<td>--disableMultipartUpload</td>
<td>Disables the use of multipart upload. Example: --disableMultipartUpload</td>
<td>No</td>
</tr>
<tr>
<td>--multipartUploadChunkSize=SIZE</td>
<td>The size, in MiB, of the multipart upload part size. By default, it uses multipart upload when writing to Amazon S3. The default chunk size is 16 MiB. Example: --multipartUploadChunkSize=32</td>
<td>No</td>
</tr>
<tr>
<td>--numberFiles</td>
<td>Prepends output files with sequential numbers. The count starts at 0 unless a different value is specified by --startingIndex. Example: --numberFiles</td>
<td>No</td>
</tr>
<tr>
<td>--startingIndex=INDEX</td>
<td>Used with --numberFiles to specify the first number in the sequence. Example: --startingIndex=1</td>
<td>No</td>
</tr>
<tr>
<td>--outputManifest=FILENAME</td>
<td>Creates a text file, compressed with Gzip, that contains a list of all the files copied by S3DistCp. Example: --outputManifest=manifest-1.gz</td>
<td>No</td>
</tr>
<tr>
<td>--previousManifest=PATH</td>
<td>Reads a manifest file that was created during a previous call to S3DistCp using the --outputManifest flag. When the --previousManifest flag is set, S3DistCp excludes the files listed in the manifest from the copy operation. If --outputManifest is specified along with --previousManifest, files listed in the previous manifest also appear in the new manifest file, although the files are not copied. Example: --previousManifest=/usr/bin/manifest-1.gz</td>
<td>No</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
<td>Required</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>--requirePreviousManifest</td>
<td>Requires a previous manifest created during a previous call to S3DistCp. If this is set to false, no error is generated when a previous manifest is not specified. The default is true.</td>
<td>No</td>
</tr>
<tr>
<td>--copyFromManifest</td>
<td>Reverses the behavior of --previousManifest to cause S3DistCp to use the specified manifest file as a list of files to copy, instead of a list of files to exclude from copying. Example: --copyFromManifest --previousManifest=/usr/bin/manifest-1.gz</td>
<td>No</td>
</tr>
<tr>
<td>--s3Endpoint=ENDPOINT</td>
<td>Specifies the Amazon S3 endpoint to use when uploading a file. This option sets the endpoint for both the source and destination. If not set, the default endpoint is s3.amazonaws.com. For a list of the Amazon S3 endpoints, see Regions and Endpoints. Example: --s3Endpoint=s3-eu-west-1.amazonaws.com</td>
<td>No</td>
</tr>
<tr>
<td>--storageClass=CLASS</td>
<td>The storage class to use when the destination is Amazon S3. Valid values are STANDARD and REDUCED_REDUNDANCY. If this option is not specified, S3DistCp tries to preserve the storage class. Example: --storageClass=STANDARD</td>
<td>No</td>
</tr>
<tr>
<td>--srcPrefixesFile=PATH</td>
<td>a text file in Amazon S3 (s3://), HDFS (hdfs:///) or local file system (file://) that contains a list of src prefixes, one prefix per line. If srcPrefixesFile is provided, S3DistCp will not list the src path. Instead, it generates a source list as the combined result of listing all prefixes specified in this file. The relative path as compared to src path, instead of these prefixes, will be used to generate the destination paths. If srcPattern is also specified, it will be applied to the combined list results of the source prefixes to further filter the input. If copyFromManifest is used, objects in the manifest will be copied and srcPrefixesFile will be ignored. Example: --srcPrefixesFile=PATH</td>
<td>No</td>
</tr>
</tbody>
</table>

In addition to the options above, S3DistCp implements the Tool interface which means that it supports the generic options.

**Adding S3DistCp as a Step in a Cluster**

You can call S3DistCp by adding it as a step in your cluster. Steps can be added to a cluster at launch or to a running cluster using the console, CLI, or API. The following examples demonstrate adding an S3DistCp step to a running cluster. For more information on adding steps to a cluster, see Submit Work to a Cluster (p. 493).
To add an S3DistCp step to a running cluster using the AWS CLI

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

- To add a step to a cluster that calls S3DistCp, pass the parameters that specify how S3DistCp should perform the copy operation as arguments.

The following example copies daemon logs from Amazon S3 to hdfs:///output. In the following command:

- --cluster-id specifies the cluster
- Jar is the location of the S3DistCp JAR file
- Args is a comma-separated list of the option name-value pairs to pass in to S3DistCp. For a complete list of the available options, see S3DistCp Options (p. 387).

To add an S3DistCp copy step to a running cluster, type the following command, replace j-3GYXXXXXX9IOK with your cluster ID, and replace mybucket with your Amazon S3 bucket name.

- Linux, UNIX, and Mac OS X users:

```bash
aws emr add-steps --cluster-id j-3GYXXXXXX9IOK --steps Type=CUSTOM_JAR,Name="S3DistCp step",Jar=/home/hadoop/lib/emr-s3distcp-1.0.jar,\ Args=["--s3Endpoint, s3-eu-west-1.amazonaws.com", "--src, s3://mybucket/logs/j-3GYXXXXXX9IOJ/node/", "--dest, hdfs:///output", "--srcPattern, .*[a-zA-Z,]+"]
```

- Windows users:

```bash
aws emr add-steps --cluster-id j-3GYXXXXXX9IOK --steps Type=CUSTOM_JAR,Name="S3DistCp step",Jar=/home/hadoop/lib/emr-s3distcp-1.0.jar,Args=["--s3Endpoint, s3-eu-west-1.amazonaws.com", "--src, s3://mybucket/logs/j-3GYXXXXXX9IOJ/node/", "--dest, hdfs:///output", "--srcPattern, .*[a-zA-Z,]+"]
```

```
[  
    {
      "Name": "S3DistCp step",  
      "Args": ["s3-dist-cp", "--s3Endpoint=s3.amazonaws.com", "--src=s3://mybucket/logs/j-3GYXXXXXX9IOJ/node/", "--dest=hdfs:///output", "--srcPattern=.*[a-zA-Z,]+"],  
      "ActionOnFailure": "CONTINUE",  
      "Type": "CUSTOM_JAR",  
      "Jar": "command-runner.jar"
    }
]
```

Example Copy log files from Amazon S3 to HDFS

This example also illustrates how to copy log files stored in an Amazon S3 bucket into HDFS by adding a step to a running cluster. In this example the --srcPattern option is used to limit the data copied to the daemon logs.

To copy log files from Amazon S3 to HDFS using the --srcPattern option, type the following command, replace j-3GYXXXXXX9IOK with your cluster ID, and replace mybucket with your Amazon S3 bucket name.

- Linux, UNIX, and Mac OS X users:
Adding S3DistCp as a Step in a Cluster

aws emr add-steps --cluster-id j-3GYXXXXXX91O9K --steps Type=CUSTOM_JAR,Name="S3DistCp step",Jar=/home/hadoop/lib/emr-s3distcp-1.0.jar,\Args=["--src,s3://mybucket/logs/j-3GYXXXXXX91O9J/node/","--dest,hdfs:///output","--srcPattern,.daemons.*-hadoop-.*"]

Windows users:

aws emr add-steps --cluster-id j-3GYXXXXXX91O9K --steps Type=CUSTOM_JAR,Name="S3DistCp step",Jar=/home/hadoop/lib/emr-s3distcp-1.0.jar,\Args=["--src,s3://mybucket/logs/j-3GYXXXXXX91O9J/node/","--dest,hdfs:///output","--srcPattern,.daemons.*-hadoop-.*"]

Example Load Amazon CloudFront logs into HDFS

This example loads Amazon CloudFront logs into HDFS by adding a step to a running cluster. In the process it changes the compression format from Gzip (the CloudFront default) to LZO. This is useful because data compressed using LZO can be split into multiple maps as it is decompressed, so you don't have to wait until the compression is complete, as you do with Gzip. This provides better performance when you analyze the data using Amazon EMR. This example also improves performance by using the regular expression specified in the --groupBy option to combine all of the logs for a given hour into a single file. Amazon EMR clusters are more efficient when processing a few, large, LZO-compressed files than when processing many, small, Gzip-compressed files. To split LZO files, you must index them and use the hadoop-lzo third party library. For more information, see How to Process Compressed Files (p. 39).

To load Amazon CloudFront logs into HDFS, type the following command, replace j-3GYXXXXXX91O9K with your cluster ID, and replace mybucket with your Amazon S3 bucket name.

- Linux, UNIX, and Mac OS X users:


Windows users:


Consider the case in which the preceding example is run over the following CloudFront log files.

s3://myawsbucket/cf/XABCD12345678.2012-02-23-01.HLUS3JKx.gz
s3://myawsbucket/cf/XABCD12345678.2012-02-23-01.I9CNAZrg.gz
s3://myawsbucket/cf/XABCD12345678.2012-02-23-02.YRRwERSA.gz
s3://myawsbucket/cf/XABCD12345678.2012-02-23-02.dshVLXFE.gz

S3DistCp copies, concatenates, and compresses the files into the following two files, where the file name is determined by the match made by the regular expression.
S3DistCp Versions Supported in Amazon EMR

Amazon EMR supports the following versions of S3DistCp.

<table>
<thead>
<tr>
<th>Version</th>
<th>Description</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0.8</td>
<td>Adds the <code>--appendToLastFile</code>, <code>--requirePreviousManifest</code>, and <code>--storageClass</code> options.</td>
<td>3 January 2014</td>
</tr>
<tr>
<td>1.0.7</td>
<td>Adds the <code>--s3ServerSideEncryption</code> option.</td>
<td>2 May 2013</td>
</tr>
<tr>
<td>1.0.6</td>
<td>Adds the <code>--s3Endpoint</code> option.</td>
<td>6 August 2012</td>
</tr>
<tr>
<td>1.0.5</td>
<td>Improves the ability to specify which version of S3DistCp to run.</td>
<td>27 June 2012</td>
</tr>
<tr>
<td>1.0.4</td>
<td>Improves the <code>--deleteOnSuccess</code> option.</td>
<td>19 June 2012</td>
</tr>
<tr>
<td>1.0.3</td>
<td>Adds support for the <code>--numberFiles</code> and <code>--startingIndex</code> options.</td>
<td>12 June 2012</td>
</tr>
<tr>
<td>1.0.2</td>
<td>Improves file naming when using groups.</td>
<td>6 June 2012</td>
</tr>
<tr>
<td>1.0.1</td>
<td>Initial release of S3DistCp.</td>
<td>19 January 2012</td>
</tr>
</tbody>
</table>

Note
S3DistCp versions after 1.0.7 are found on directly on the clusters. Users should use the JAR in `/home/hadoop/lib` for the latest features.

Export, Import, Query, and Join Tables in DynamoDB Using Amazon EMR

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Topics
- Prerequisites for Integrating Amazon EMR with DynamoDB (p. 394)
- Step 1: Create a Key Pair (p. 395)
- Create a Cluster (p. 396)
- SSH into the Master Node (p. 399)
- Set Up a Hive Table to Run Hive Commands (p. 401)
- Hive Command Examples for Exporting, Importing, and Querying Data in DynamoDB (p. 405)
- Optimizing Performance for Amazon EMR Operations in DynamoDB (p. 411)
In the following sections, you will learn how to use Amazon EMR with a customized version of Hive that includes connectivity to DynamoDB to perform operations on data stored in DynamoDB, such as:

- Loading DynamoDB data into the Hadoop Distributed File System (HDFS) and using it as input into an Amazon EMR cluster.
- Querying live DynamoDB data using SQL-like statements (HiveQL).
- Joining data stored in DynamoDB and exporting it or querying against the joined data.
- Exporting data stored in DynamoDB to Amazon S3.
- Importing data stored in Amazon S3 to DynamoDB.

To perform each of the tasks above, you'll launch an Amazon EMR cluster, specify the location of the data in DynamoDB, and issue Hive commands to manipulate the data in DynamoDB.

DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. Developers can create a database table and grow its request traffic or storage without limit. DynamoDB automatically spreads the data and traffic for the table over a sufficient number of servers to handle the request capacity specified by the customer and the amount of data stored, while maintaining consistent, fast performance. Using Amazon EMR and Hive you can quickly and efficiently process large amounts of data, such as data stored in DynamoDB. For more information, see the Amazon DynamoDB Developer Guide.

Apache Hive is a software layer that you can use to query map reduce clusters using a simplified, SQL-like query language called HiveQL. It runs on top of the Hadoop architecture. For more information about Hive and HiveQL, go to the HiveQL Language Manual.

There are several ways to launch an Amazon EMR cluster: you can use the Amazon EMR console, the command line interface (CLI), or you can program your cluster using the AWS SDK or the API. You can also choose whether to run a Hive cluster interactively or from a script. In this section, we will show you how to launch an interactive Hive cluster from the Amazon EMR console and the CLI.

Using Hive interactively is a great way to test query performance and tune your application. Once you have established a set of Hive commands that will run on a regular basis, consider creating a Hive script that Amazon EMR can run for you. For more information about how to run Hive from a script, go to Submit Hive Work (p. 278).

**Warning**

Amazon EMR read or write operations on a DynamoDB table count against your established provisioned throughput, potentially increasing the frequency of provisioned throughput exceptions. For large requests, Amazon EMR implements retries with exponential backoff to manage the request load on the DynamoDB table. Running Amazon EMR jobs concurrently with other traffic may cause you to exceed the allocated provisioned throughput level. You can monitor this by checking the ThrottleRequests metric in Amazon CloudWatch. If the request load is too high, you can relaunch the cluster and set the Read Percent Setting (p. 412) or Write Percent Setting (p. 412) to a lower value to throttle the Amazon EMR operations. For information about DynamoDB throughput settings, see Provisioned Throughput in the Amazon DynamoDB Developer Guide.

**Prerequisites for Integrating Amazon EMR with DynamoDB**

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

To use Amazon EMR (Amazon EMR) and Hive to manipulate data in DynamoDB, you need the following:
Step 1: Create a Key Pair

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

To run Hive interactively to manage data in DynamoDB, you will need a key pair to connect to the Amazon EC2 instances launched by Amazon EMR (Amazon EMR). You will use this key pair to connect to the master node of the Amazon EMR job flow to run a HiveQL script (a language similar to SQL).

To generate a key pair

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the upper right hand corner of the console, select a Region from the Region drop-down menu. This should be the same region that your DynamoDB database is in.
3. Click Key Pairs in the Navigation pane.

The console displays a list of key pairs associated with your account.
4. Click Create Key Pair.
5. Enter a name for the key pair, such as mykeypair, for the new key pair in the Key Pair Name field and click Create.
6. Download the private key file. The file name will end with .pem, (such as mykeypair.pem). Keep this private key file in a safe place. You will need it to access any instances that you launch with this key pair.

Important
If you lose the key pair, you cannot connect to your Amazon EC2 instances.

For more information about key pairs, see Amazon Elastic Compute Cloud Key Pairs in the Amazon EC2 User Guide for Linux Instances.
Create a Cluster

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

In order for Hive to run on Amazon EMR, you must create a cluster with Hive enabled. This sets up the necessary applications and infrastructure for Hive to connect to DynamoDB. The following procedures explain how to create an interactive Hive cluster from the AWS Management Console and the CLI.

**Topics**
- To start a cluster using the AWS Management Console (p. 396)

### To start a cluster using the AWS Management Console

1. Sign in to the AWS Management Console and open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. Click **Create Cluster**.
3. In the **Create Cluster** page, in the **Cluster Configuration** section, verify the fields according to the following table.

<table>
<thead>
<tr>
<th>Field</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster name</td>
<td>Enter a descriptive name for your cluster.</td>
</tr>
<tr>
<td></td>
<td>The name is optional, and does not need to be unique.</td>
</tr>
<tr>
<td>Termination protection</td>
<td>Choose <strong>Yes</strong>.</td>
</tr>
<tr>
<td></td>
<td>Enabling termination protection ensures that the cluster does not shut down due to accident or error. For more information, see Protect a Cluster from Termination in the Amazon EMR Developer Guide. Typically, set this value to <strong>Yes</strong> only when developing an application (so you can debug errors that would have otherwise terminated the cluster) and to protect long-running clusters or clusters that contain data.</td>
</tr>
<tr>
<td>Logging</td>
<td>Choose <strong>Enabled</strong>.</td>
</tr>
<tr>
<td></td>
<td>This determines whether Amazon EMR captures detailed log data to Amazon S3.</td>
</tr>
<tr>
<td></td>
<td>For more information, see View Log Files in the Amazon EMR Developer Guide.</td>
</tr>
<tr>
<td>Log folder S3 location</td>
<td>Enter an Amazon S3 path to store your debug logs if you enabled logging in the previous field.</td>
</tr>
<tr>
<td></td>
<td>When this value is set, Amazon EMR copies the log files from the EC2 instances in the cluster to Amazon S3. This prevents the log files from being lost when the cluster ends and the EC2 instances hosting the cluster are terminated. These logs are useful for troubleshooting purposes.</td>
</tr>
<tr>
<td></td>
<td>For more information, see View Log Files in the Amazon EMR Developer Guide.</td>
</tr>
</tbody>
</table>
### Create a Cluster

<table>
<thead>
<tr>
<th>Field</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debugging</td>
<td>Choose <strong>Enabled</strong>. This option creates a debug log index in Amazon SimpleDB (additional charges apply) to enable detailed debugging in the Amazon EMR console. You can only set this when the cluster is created. For more information, see the Amazon SimpleDB product description page.</td>
</tr>
</tbody>
</table>

4. In the **Software Configuration** section, verify the fields according to the following table.

<table>
<thead>
<tr>
<th>Field</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hadoop distribution</td>
<td>Choose <strong>Amazon</strong>. This determines which distribution of Hadoop to run on your cluster. You can choose to run the Amazon distribution of Hadoop or one of several MapR distributions. For more information, see Using the MapR Distribution for Hadoop in the Amazon EMR Developer Guide.</td>
</tr>
<tr>
<td>AMI version</td>
<td>Choose the latest AMI version in the list. This determines the version of Hadoop and other applications such as Hive or Pig to run on your cluster. For more information, see Choose a Machine Image in the Amazon EMR Developer Guide.</td>
</tr>
<tr>
<td>Applications to be installed - Hive</td>
<td>A default Hive version should already be selected and displayed in the list. If it does not appear, choose it from the Additional applications list. For more information, see Analyze Data with Hive in the Amazon EMR Developer Guide.</td>
</tr>
<tr>
<td>Applications to be installed - Pig</td>
<td>A default Pig version should already be selected and displayed in the list. If it does not appear, choose it from the Additional applications list. For more information, see Process Data with Pig in the Amazon EMR Developer Guide.</td>
</tr>
</tbody>
</table>

5. In the **Hardware Configuration** section, verify the fields according to the following table.

**Note**
The default maximum number of nodes per AWS account is twenty. For example, if you have two clusters running, the total number of nodes running for both clusters must be 20 or less. Exceeding this limit will result in cluster failures. If you need more than 20 nodes, you must submit a request to increase your Amazon EC2 instance limit. Ensure that your requested limit increase includes sufficient capacity for any temporary, unplanned increases in your needs. For more information, go to the Request to Increase Amazon EC2 Instance Limit Form.

<table>
<thead>
<tr>
<th>Field</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network</td>
<td>Choose <strong>Launch into EC2-Classic</strong>. Optionally, choose a VPC subnet identifier from the list to launch the cluster in an Amazon VPC. For more information, see Select a Amazon VPC Subnet for the Cluster (Optional) in the Amazon EMR Developer Guide.</td>
</tr>
<tr>
<td>Field</td>
<td>Action</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>EC2 Availability Zone</td>
<td>Choose <strong>No preference</strong>.</td>
</tr>
<tr>
<td></td>
<td>Optionally, you can launch the cluster in a specific EC2 Availability Zone.</td>
</tr>
<tr>
<td></td>
<td>For more information, see <a href="https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html">Regions and Availability Zones</a> in the <em>Amazon EC2 User Guide for Linux Instances</em>.</td>
</tr>
<tr>
<td>Master - Amazon EC2 Instance Type</td>
<td>For this tutorial, use the default <strong>EC2 instance type</strong> that is shown in this field. This specifies the EC2 instance types to use as master nodes. The master node assigns Hadoop tasks to core and task nodes, and monitors their status. There is always one master node in each cluster.</td>
</tr>
<tr>
<td></td>
<td>For more information, see <a href="https://docs.aws.amazon.com/emr/latest/developerguide/instance-groups.html">Instance Groups</a> in the <em>Amazon EMR Developer Guide</em>.</td>
</tr>
<tr>
<td>Request Spot Instances</td>
<td>Leave this box unchecked.</td>
</tr>
<tr>
<td></td>
<td>This specifies whether to run master nodes on Spot Instances. For more information, see <a href="https://docs.aws.amazon.com/emr/latest/developerguide/lower-cost-spot-instances.html">Lower Costs with Spot Instances (Optional)</a> in the <em>Amazon EMR Developer Guide</em>.</td>
</tr>
<tr>
<td>Core - Amazon EC2 Instance Type</td>
<td>For this tutorial, use the default <strong>EC2 instance type</strong> that is shown in this field. This specifies the EC2 instance types to use as core nodes. A core node is an EC2 instance that runs Hadoop map and reduce tasks and stores data using the Hadoop Distributed File System (HDFS). Core nodes are managed by the master node.</td>
</tr>
<tr>
<td></td>
<td>For more information, see <a href="https://docs.aws.amazon.com/emr/latest/developerguide/instance-groups.html">Instance Groups</a> in the <em>Amazon EMR Developer Guide</em>.</td>
</tr>
<tr>
<td>Count</td>
<td>Choose <strong>2</strong>.</td>
</tr>
<tr>
<td>Request Spot Instances</td>
<td>Leave this box unchecked.</td>
</tr>
<tr>
<td></td>
<td>This specifies whether to run core nodes on Spot Instances. For more information, see <a href="https://docs.aws.amazon.com/emr/latest/developerguide/lower-cost-spot-instances.html">Lower Costs with Spot Instances (Optional)</a> in the <em>Amazon EMR Developer Guide</em>.</td>
</tr>
<tr>
<td>Task - Amazon EC2 Instance Type</td>
<td>For this tutorial, use the default <strong>EC2 instance type</strong> that is shown in this field. This specifies the EC2 instance types to use as task nodes. A task node only processes Hadoop tasks and don't store data. You can add and remove them from a cluster to manage the EC2 instance capacity your cluster uses, increasing capacity to handle peak loads and decreasing it later. Task nodes only run a TaskTracker Hadoop daemon.</td>
</tr>
<tr>
<td></td>
<td>For more information, see <a href="https://docs.aws.amazon.com/emr/latest/developerguide/instance-groups.html">Instance Groups</a> in the <em>Amazon EMR Developer Guide</em>.</td>
</tr>
<tr>
<td>Count</td>
<td>Choose <strong>0</strong>.</td>
</tr>
<tr>
<td>Request Spot Instances</td>
<td>Leave this box unchecked.</td>
</tr>
<tr>
<td></td>
<td>This specifies whether to run task nodes on Spot Instances. For more information, see <a href="https://docs.aws.amazon.com/emr/latest/developerguide/lower-cost-spot-instances.html">Lower Costs with Spot Instances (Optional)</a> in the <em>Amazon EMR Developer Guide</em>.</td>
</tr>
</tbody>
</table>
6. In the **Security and Access** section, complete the fields according to the following table.

<table>
<thead>
<tr>
<th>Field</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>EC2 key pair</td>
<td>Choose the key pair that you created in Step 1: Create a Key Pair (p. 395). For more information, see Create SSH Credentials for the Master Node in the Amazon EMR Developer Guide. If you do not enter a value in this field, you will not be able to connect to the master node using SSH. For more information, see Connect to the Cluster in the Amazon EMR Developer Guide.</td>
</tr>
<tr>
<td>IAM user access</td>
<td>Choose <strong>No other IAM users</strong>. Optionally, choose <strong>All other IAM users</strong> to make the cluster visible and accessible to all IAM users on the AWS account. For more information, see Configure IAM User Permissions in the Amazon EMR Developer Guide.</td>
</tr>
<tr>
<td>IAM role</td>
<td>Choose <strong>Proceed without roles</strong>. This controls application access to the EC2 instances in the cluster. For more information, see Configure IAM Roles for Amazon EMR in the Amazon EMR Developer Guide.</td>
</tr>
</tbody>
</table>

7. Review the **Bootstrap Actions** section, but note that you do not need to make any changes. There are no bootstrap actions necessary for this sample configuration.

   Optionally, you can use bootstrap actions, which are scripts that can install additional software and change the configuration of applications on the cluster before Hadoop starts. For more information, see Create Bootstrap Actions to Install Additional Software (Optional) in the Amazon EMR Developer Guide.

8. Review your configuration and if you are satisfied with the settings, click **Create Cluster**.

9. When the cluster starts, you see the **Summary** pane.

---

**SSH into the Master Node**

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

When the cluster’s status is **WAITING**, the master node is ready for you to connect to it. With an active SSH session into the master node, you can execute command line operations.

**To locate the public DNS name of the master node**

- In the Amazon EMR console, select the cluster from the list of running clusters in the **WAITING** state.

   The DNS name you use to connect to the instance is listed as **Master Public DNS Name**.
To connect to the master node using Mac OS X/Linux/UNIX

1. Go to the command prompt on your system. (On Mac OS X, use the Terminal program in /Applications/Utilities/Terminal.)

2. Set the permissions on the .pem file for your Amazon EC2 key pair so that only the key owner has permissions to access the key. For example, if you saved the file as mykeypair.pem in the user’s home directory, the command is:

   chmod og-rwx ~/mykeypair.pem

   If you do not perform this step, SSH returns an error saying that your private key file is unprotected and rejects the key. You only need to perform this step the first time you use the private key to connect.

3. To establish the connection to the master node, enter the following command line, which assumes the .pem file is in the user’s home directory. Replace master-public-dns-name with the Master Public DNS Name of your cluster and replace ~/mykeypair.pem with the location and filename of your .pem file.

   ssh hadoop@master-public-dns-name -i ~/mykeypair.pem

   A warning states that the authenticity of the host you are connecting to can't be verified.

4. Type yes to continue.

   **Note**
   If you are asked to log in, enter hadoop.

To install and configure PuTTY on Windows


2. Launch PuTTYgen.

3. Click **Load**.

4. Select the PEM file you created earlier. Note that you may have to change the search parameters from file of type "PuTTY Private Key Files (*.ppk) to "All Files (*.*)".

5. Click **Open**.

6. Click **OK** on the PuTTYgen notice telling you the key was successfully imported.

7. Click **Save private key** to save the key in the PPK format.

8. When PuTTYgen prompts you to save the key without a pass phrase, click **Yes**.

9. Enter a name for your PuTTY private key, such as mykeypair.ppk.

10. Click **Save**.

11. Close PuTTYgen.

To connect to the master node using PuTTY on Windows

1. Start PuTTY.

2. Select **Session** in the Category list. Enter hadoop@DNS in the Host Name field. The input looks similar to hadoop@ec2-184-72-128-177.compute-1.amazonaws.com.

3. In the Category list, expand **Connection**, expand **SSH**, and then select **Auth**. The Options controlling the SSH authentication pane appears.
4. For **Private key file for authentication**, click **Browse** and select the private key file you generated earlier. If you are following this guide, the file name is `mykeypair.ppk`.

5. Click **Open**.

   A PuTTY Security Alert pops up.

6. Click **Yes** for the PuTTY Security Alert.

   **Note**
   
   If you are asked to log in, enter `hadoop`.

After you connect to the master node using either SSH or PuTTY, you should see a Hadoop command prompt and you are ready to start a Hive interactive session.

**Set Up a Hive Table to Run Hive Commands**

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Apache Hive is a data warehouse application you can use to query data contained in Amazon EMR clusters using a SQL-like language. For more information about Hive, go to [http://hive.apache.org/](http://hive.apache.org/).

The following procedure assumes you have already created a cluster and specified an Amazon EC2 key pair. To learn how to get started creating clusters, see Step 3: Launch an Amazon EMR Cluster (p. 16).

**To run Hive commands interactively**

1. Connect to the master node. For more information, see Connect to the Master Node Using SSH (p. 457).

2. At the command prompt for the current master node, type `hive`.

   You should see a hive prompt: `hive>`

3. Enter a Hive command that maps a table in the Hive application to the data in DynamoDB. This table acts as a reference to the data stored in Amazon DynamoDB; the data is not stored locally in Hive and any queries using this table run against the live data in DynamoDB, consuming the table’s read or write capacity every time a command is run. If you expect to run multiple Hive commands against the same dataset, consider exporting it first.

   The following shows the syntax for mapping a Hive table to a DynamoDB table.

   ```sql
   CREATE EXTERNAL TABLE hive_tablename
   (hive_column1_name column1_datatype, hive_column2_name column2_datatype...)
   STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'
   TBLPROPERTIES ("dynamodb.table.name" = "dynamodb_tablename",
   "dynamodb.column.mapping" =
   "hive_column1_name:dynamodb_attribute1_name,hive_column2_name:dynamodb_attribute2_name...");
   ```

   When you create a table in Hive from DynamoDB, you must create it as an external table using the keyword **EXTERNAL**. The difference between external and internal tables is that the data in internal tables is deleted when an internal table is dropped. This is not the desired behavior when connected to Amazon DynamoDB, and thus only external tables are supported.

   For example, the following Hive command creates a table named `hivetable1` in Hive that references the DynamoDB table named `dynamodbtable1`. The DynamoDB table `dynamodbtable1` has a hash-
and-range primary key schema. The hash key element is `name` (string type), the range key element is `year` (numeric type), and each item has an attribute value for `holidays` (string set type).

```
CREATE EXTERNAL TABLE hivetable1 (col1 string, col2 bigint, col3 array<string>) 
STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'
TBLPROPERTIES ("dynamodb.table.name" = "dynamodbtable1",
"dynamodb.column.mapping" = "col1:name,col2:year,col3:holidays");
```

Line 1 uses the HiveQL `CREATE EXTERNAL TABLE` statement. For `hivetable1`, you need to establish a column for each attribute name-value pair in the DynamoDB table, and provide the data type. These values are not case-sensitive, and you can give the columns any name (except reserved words).

Line 2 uses the `STORED BY` statement. The value of `STORED BY` is the name of the class that handles the connection between Hive and DynamoDB. It should be set to 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'.

Line 3 uses the `TBLPROPERTIES` statement to associate "hivetable1" with the correct table and schema in DynamoDB. Provide `TBLPROPERTIES` with values for the `dynamodb.table.name` parameter and `dynamodb.column.mapping` parameter. These values are case-sensitive.

**Note**
All DynamoDB attribute names for the table must have corresponding columns in the Hive table; otherwise, the Hive table won’t contain the name-value pair from DynamoDB. If you do not map the DynamoDB primary key attributes, Hive generates an error. If you do not map a non-primary key attribute, no error is generated, but you won’t see the data in the Hive table. If the data types do not match, the value is null.

Then you can start running Hive operations on `hivetable1`. Queries run against `hivetable1` are internally run against the DynamoDB table `dynamodbtable1` of your DynamoDB account, consuming read or write units with each execution.

When you run Hive queries against a DynamoDB table, you need to ensure that you have provisioned a sufficient amount of read capacity units.

For example, suppose that you have provisioned 100 units of read capacity for your DynamoDB table. This will let you perform 100 reads, or 409,600 bytes, per second. If that table contains 20GB of data (21,474,836,480 bytes), and your Hive query performs a full table scan, you can estimate how long the query will take to run:

\[
\frac{21,474,836,480}{409,600} = 52,429 \text{ seconds} = 14.56 \text{ hours}
\]

The only way to decrease the time required would be to adjust the read capacity units on the source DynamoDB table. Adding more Amazon EMR nodes will not help.

In the Hive output, the completion percentage is updated when one or more mapper processes are finished. For a large DynamoDB table with a low provisioned read capacity setting, the completion percentage output might not be updated for a long time; in the case above, the job will appear to be 0% complete for several hours. For more detailed status on your job’s progress, go to the Amazon EMR console; you will be able to view the individual mapper task status, and statistics for data reads. You can also log on to Hadoop interface on the master node and see the Hadoop statistics. This will show you the individual map task status and some data read statistics. For more information, see the following topics:

- Web Interfaces Hosted on the Master Node
- View the Hadoop Web Interfaces
For more information about sample HiveQL statements to perform tasks such as exporting or importing data from DynamoDB and joining tables, see Hive Command Examples for Exporting, Importing, and Querying Data in Amazon DynamoDB in the Amazon EMR Developer Guide.

To cancel a Hive request

When you execute a Hive query, the initial response from the server includes the command to cancel the request. To cancel the request at any time in the process, use the Kill Command from the server response.

1. Enter Ctrl+C to exit the command line client.
2. At the shell prompt, enter the Kill Command from the initial server response to your request.

Alternatively, you can run the following command from the command line of the master node to kill the Hadoop job, where \( \text{job-id} \) is the identifier of the Hadoop job and can be retrieved from the Hadoop user interface. For more information about the Hadoop user interface, see How to Use the Hadoop User Interface in the Amazon EMR Developer Guide.

```
hadoop job -kill \text{job-id}
```

Data Types for Hive and DynamoDB

The following table shows the available Hive data types and how they map to the corresponding DynamoDB data types.

<table>
<thead>
<tr>
<th>Hive type</th>
<th>DynamoDB type</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>string (S)</td>
</tr>
<tr>
<td>bigint or double</td>
<td>number (N)</td>
</tr>
<tr>
<td>binary</td>
<td>binary (B)</td>
</tr>
<tr>
<td>array</td>
<td>number set (NS), string set (SS), or binary set (BS)</td>
</tr>
</tbody>
</table>

The bigint type in Hive is the same as the Java long type, and the Hive double type is the same as the Java double type in terms of precision. This means that if you have numeric data stored in DynamoDB that has precision higher than is available in the Hive datatypes, using Hive to export, import, or reference the DynamoDB data could lead to a loss in precision or a failure of the Hive query.

Exports of the binary type from DynamoDB to Amazon Simple Storage Service (Amazon S3) or HDFS are stored as a Base64-encoded string. If you are importing data from Amazon S3 or HDFS into the DynamoDB binary type, it should be encoded as a Base64 string.

Hive Options

You can set the following Hive options to manage the transfer of data out of Amazon DynamoDB. These options only persist for the current Hive session. If you close the Hive command prompt and reopen it later on the cluster, these settings will have returned to the default values.
<table>
<thead>
<tr>
<th>Hive Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dynamodb.throughput.read.percent</td>
<td>Set the rate of read operations to keep your DynamoDB provisioned throughput rate in the allocated range for your table. The value is between $0.1$ and $1.5$, inclusively.</td>
</tr>
<tr>
<td></td>
<td>The value of $0.5$ is the default read rate, which means that Hive will attempt to consume half of the read provisioned throughout resources in the table. Increasing this value above $0.5$ increases the read request rate.</td>
</tr>
<tr>
<td></td>
<td>Decreasing it below $0.5$ decreases the read request rate. This read rate is approximate. The actual read rate will depend on factors such as whether there is a uniform distribution of keys in DynamoDB.</td>
</tr>
<tr>
<td></td>
<td>If you find your provisioned throughput is frequently exceeded by the Hive operation, or if live read traffic is being throttled too much, then reduce this value below $0.5$. If you have enough capacity and want a faster Hive operation, set this value above $0.5$. You can also oversubscribe by setting it up to $1.5$ if you believe there are unused input/output operations available.</td>
</tr>
<tr>
<td>dynamodb.throughput.write.percent</td>
<td>Set the rate of write operations to keep your DynamoDB provisioned throughput rate in the allocated range for your table. The value is between $0.1$ and $1.5$, inclusively.</td>
</tr>
<tr>
<td></td>
<td>The value of $0.5$ is the default write rate, which means that Hive will attempt to consume half of the write provisioned throughout resources in the table. Increasing this value above $0.5$ increases the write request rate.</td>
</tr>
<tr>
<td></td>
<td>Decreasing it below $0.5$ decreases the write request rate. This write rate is approximate. The actual write rate will depend on factors such as whether there is a uniform distribution of keys in DynamoDB.</td>
</tr>
<tr>
<td></td>
<td>If you find your provisioned throughput is frequently exceeded by the Hive operation, or if live write traffic is being throttled too much, then reduce this value below $0.5$. If you have enough capacity and want a faster Hive operation, set this value above $0.5$. You can also oversubscribe by setting it up to $1.5$ if you believe there are unused input/output operations available or this is the initial data upload to the table and there is no live traffic yet.</td>
</tr>
<tr>
<td>dynamodb.endpoint</td>
<td>Specify the endpoint in case you have tables in different regions. For more information about the available DynamoDB endpoints, see Regions and Endpoints.</td>
</tr>
<tr>
<td>dynamodb.max.map.tasks</td>
<td>Specify the maximum number of map tasks when reading data from DynamoDB. This value must be equal to or greater than $1$.</td>
</tr>
<tr>
<td>dynamodb.retry.duration</td>
<td>Specify the number of minutes to use as the timeout duration for retrying Hive commands. This value must be an integer equal to or greater than $0$. The default timeout duration is two minutes.</td>
</tr>
</tbody>
</table>
These options are set using the `SET` command as shown in the following example.

```sql
SET dynamodb.throughput.read.percent=1.0;

INSERT OVERWRITE TABLE s3_export SELECT * FROM hiveTableName;
```

If you are using the AWS SDK for Java, you can use the `-e` option of Hive to pass in the command directly, as shown in the last line of the following example.

```java
steps.add(new StepConfig()
    .withName("Run Hive Script")
    .withHadoopJarStep(new HadoopJarStepConfig()
        .withJar("s3://us-west-2.elasticmapreduce/libs/script-runner/script-runner.jar")
        .withArgs("s3://us-west-2.elasticmapreduce/libs/hive/hive-script",
            "--base-path","s3://us-west-2.elasticmapreduce/libs/hive/",
            "--run-hive-script",
            "--args","-e","SET dynamodb.throughput.read.percent=1.0;")));
```

### Hive Command Examples for Exporting, Importing, and Querying Data in DynamoDB

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

The following examples use Hive commands to perform operations such as exporting data to Amazon S3 or HDFS, importing data to DynamoDB, joining tables, querying tables, and more.

Operations on a Hive table reference data stored in DynamoDB. Hive commands are subject to the DynamoDB table's provisioned throughput settings, and the data retrieved includes the data written to the DynamoDB table at the time the Hive operation request is processed by DynamoDB. If the data retrieval process takes a long time, some data returned by the Hive command may have been updated in DynamoDB since the Hive command began.

Hive commands `DROP TABLE` and `CREATE TABLE` only act on the local tables in Hive and do not create or drop tables in DynamoDB. If your Hive query references a table in DynamoDB, that table must already exist before you run the query. For more information about creating and deleting tables in DynamoDB, see [Working with Tables in DynamoDB in the Amazon DynamoDB Developer Guide](https://docs.amazonwebservices.com/AmazonDynamoDB/latest/DeveloperGuide/WorkingWithTables.html).

**Note**

When you map a Hive table to a location in Amazon S3, do not map it to the root path of the bucket, `s3://mybucket`, as this may cause errors when Hive writes the data to Amazon S3. Instead map the table to a subpath of the bucket, `s3://mybucket/mypath`.

### Exporting Data from DynamoDB

You can use Hive to export data from DynamoDB.

**To export a DynamoDB table to an Amazon S3 bucket**

- Create a Hive table that references data stored in DynamoDB. Then you can call the `INSERT OVERWRITE` command to write the data to an external directory. In the following example, `s3://bucketname/path/subpath/` is a valid path in Amazon S3. Adjust the columns and datatypes in the `SELECT` statement to match the schema of the target table.
the CREATE command to match the values in your DynamoDB. You can use this to create an archive of your DynamoDB data in Amazon S3.

```
CREATE EXTERNAL TABLE hiveTableName (col1 string, col2 bigint, col3 array<string>)
STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'
TBLPROPERTIES ("dynamodb.table.name" = "dynamodbtable1",
"dynamodb.column.mapping" = "col1:name,col2:year,col3:holidays");

INSERT OVERWRITE DIRECTORY 's3://bucketname/path/subpath/' SELECT *
FROM hiveTableName;
```

To export a DynamoDB table to an Amazon S3 bucket using formatting

- Create an external table that references a location in Amazon S3. This is shown below as s3_export. During the CREATE call, specify row formatting for the table. Then, when you use INSERT OVERWRITE to export data from DynamoDB to s3_export, the data is written out in the specified format. In the following example, the data is written out as comma-separated values (CSV).

```
CREATE EXTERNAL TABLE hiveTableName (col1 string, col2 bigint, col3 array<string>)
STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'
TBLPROPERTIES ("dynamodb.table.name" = "dynamodbtable1",
"dynamodb.column.mapping" = "col1:name,col2:year,col3:holidays");

CREATE EXTERNAL TABLE s3_export (a_col string, b_col bigint, c_col array<string>)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LOCATION 's3://bucketname/path/subpath/';

INSERT OVERWRITE TABLE s3_export SELECT *
FROM hiveTableName;
```

To export a DynamoDB table to an Amazon S3 bucket without specifying a column mapping

- Create a Hive table that references data stored in DynamoDB. This is similar to the preceding example, except that you are not specifying a column mapping. The table must have exactly one column of type map<string, string>. If you then create an EXTERNAL table in Amazon S3 you can call the INSERT OVERWRITE command to write the data from DynamoDB to Amazon S3. You can use this to create an archive of your DynamoDB data in Amazon S3. Because there is no column mapping, you cannot query tables that are exported this way. Exporting data without specifying a column mapping is available in Hive 0.8.1.5 or later, which is supported on Amazon EMR AMI 2.2.x and later.

```
CREATE EXTERNAL TABLE hiveTableName (item map<string,string>)
STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'
TBLPROPERTIES ("dynamodb.table.name" = "dynamodbtable1");

CREATE EXTERNAL TABLE s3TableName (item map<string, string>)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n'
LOCATION 's3://bucketname/path/subpath/';

INSERT OVERWRITE TABLE s3TableName SELECT *
FROM hiveTableName;
```
To export a DynamoDB table to an Amazon S3 bucket using data compression

- Hive provides several compression codecs you can set during your Hive session. Doing so causes the exported data to be compressed in the specified format. The following example compresses the exported files using the Lempel-Ziv-Oberhumer (LZO) algorithm.

```sql
SET hive.exec.compress.output=true;
SET io.seqfile.compression.type=BLOCK;
SET mapred.output.compression.codec = com.hadoop.compression.lzo.LzopCodec;

CREATE EXTERNAL TABLE hiveTableName (col1 string, col2 bigint, col3 array<string>) STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'
TBLPROPERTIES ("dynamodb.table.name" = "dynamodbtable1",
"dynamodb.column.mapping" = "col1:name,col2:year,col3:holidays");

CREATE EXTERNAL TABLE lzo_compression_table (line STRING)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n'
LOCATION 's3://bucketname/path/subpath/';

INSERT OVERWRITE TABLE lzo_compression_table SELECT * FROM hiveTableName;
```

The available compression codecs are:
- org.apache.hadoop.io.compress.GzipCodec
- org.apache.hadoop.io.compress.DefaultCodec
- com.hadoop.compression.lzo.LzoCodec
- com.hadoop.compression.lzo.LzopCodec
- org.apache.hadoop.io.compress.BZip2Codec
- org.apache.hadoop.io.compress.SnappyCodec

To export a DynamoDB table to HDFS

- Use the following Hive command, where hdfs:///directoryName is a valid HDFS path and hiveTableName is a table in Hive that references DynamoDB. This export operation is faster than exporting a DynamoDB table to Amazon S3 because Hive 0.7.1.1 uses HDFS as an intermediate step when exporting data to Amazon S3. The following example also shows how to set dynamodb.throughput.read.percent to 1.0 in order to increase the read request rate.

```sql
SET dynamodb.throughput.read.percent=1.0;

CREATE EXTERNAL TABLE hiveTableName (col1 string, col2 bigint, col3 array<string>) STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'
TBLPROPERTIES ("dynamodb.table.name" = "dynamodbtable1",
"dynamodb.column.mapping" = "col1:name,col2:year,col3:holidays");

INSERT OVERWRITE DIRECTORY 'hdfs:///directoryName' SELECT * FROM hiveTableName;
```

You can also export data to HDFS using formatting and compression as shown above for the export to Amazon S3. To do so, simply replace the Amazon S3 directory in the examples above with an HDFS directory.
To read non-printable UTF-8 character data in Hive

- You can read and write non-printable UTF-8 character data with Hive by using the `STORED AS` `SEQUENCEFILE` clause when you create the table. A SequenceFile is Hadoop binary file format; you need to use Hadoop to read this file. The following example shows how to export data from DynamoDB into Amazon S3. You can use this functionality to handle non-printable UTF-8 encoded characters.

```
CREATE EXTERNAL TABLE hiveTableName (col1 string, col2 bigint, col3 array<string>)
STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'
TBLPROPERTIES ("dynamodb.table.name" = "dynamodbtable1",
"dynamodb.column.mapping" = "col1:name,col2:year,col3:holidays");

CREATE EXTERNAL TABLE s3_export(a_col string, b_col bigint, c_col array<string>)
STORED AS SEQUENCEFILE
LOCATION 's3://bucketname/path/subpath/';

INSERT OVERWRITE TABLE s3_export SELECT * FROM hiveTableName;
```

Importing Data to DynamoDB

When you write data to DynamoDB using Hive you should ensure that the number of write capacity units is greater than the number of mappers in the cluster. For example, clusters that run on m1.xlarge EC2 instances produce 8 mappers per instance. In the case of a cluster that has 10 instances, that would mean a total of 80 mappers. If your write capacity units are not greater than the number of mappers in the cluster, the Hive write operation may consume all of the write throughput, or attempt to consume more throughput than is provisioned. For more information about the number of mappers produced by each EC2 instance type, go to Hadoop Configuration Reference. There, you will find a "Task Configuration" section for each of the supported configurations.

The number of mappers in Hadoop are controlled by the input splits. If there are too few splits, your write command might not be able to consume all the write throughput available.

If an item with the same key exists in the target DynamoDB table, it will be overwritten. If no item with the key exists in the target DynamoDB table, the item is inserted.

To import a table from Amazon S3 to DynamoDB

- You can use Amazon EMR (Amazon EMR) and Hive to write data from Amazon S3 to DynamoDB.

```
CREATE EXTERNAL TABLE s3_import(a_col string, b_col bigint, c_col array<string>)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LOCATION 's3://bucketname/path/subpath/';

CREATE EXTERNAL TABLE hiveTableName (col1 string, col2 bigint, col3 array<string>)
STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'
TBLPROPERTIES ("dynamodb.table.name" = "dynamodbtable1",
"dynamodb.column.mapping" = "col1:name,col2:year,col3:holidays");

INSERT OVERWRITE TABLE hiveTableName SELECT * FROM s3_import;
```
**To import a table from an Amazon S3 bucket to DynamoDB without specifying a column mapping**

- Create an `EXTERNAL` table that references data stored in Amazon S3 that was previously exported from DynamoDB. Before importing, ensure that the table exists in DynamoDB and that it has the same key schema as the previously exported DynamoDB table. In addition, the table must have exactly one column of type `map<string, string>`. If you then create a Hive table that is linked to DynamoDB, you can call the `INSERT OVERWRITE` command to write the data from Amazon S3 to DynamoDB. Because there is no column mapping, you cannot query tables that are imported this way. Importing data without specifying a column mapping is available in Hive 0.8.1.5 or later, which is supported on Amazon EMR AMI 2.2.3 and later.

```
CREATE EXTERNAL TABLE s3TableName (item map<string, string>)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n'
LOCATION 's3://bucketname/path/subpath/';

CREATE EXTERNAL TABLE hiveTableName (item map<string,string>)
STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'
TBLPROPERTIES ("dynamodb.table.name" = "dynamodbtable1");

INSERT OVERWRITE TABLE hiveTableName SELECT * FROM s3TableName;
```

**To import a table from HDFS to DynamoDB**

- You can use Amazon EMR and Hive to write data from HDFS to DynamoDB.

```
CREATE EXTERNAL TABLE hdfs_import(a_col string, b_col bigint, c_col array<string>)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LOCATION 'hdfs:///directoryName';

CREATE EXTERNAL TABLE hiveTableName (col1 string, col2 bigint, col3 array<string>)
STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'
TBLPROPERTIES ("dynamodb.table.name" = "dynamodbtable1",
"dynamodb.column.mapping" = "col1:name,col2:year,col3:holidays");

INSERT OVERWRITE TABLE hiveTableName SELECT * FROM hdfs_import;
```

**Querying Data in DynamoDB**

The following examples show the various ways you can use Amazon EMR to query data stored in DynamoDB.

**To find the largest value for a mapped column (max)**

- Use Hive commands like the following. In the first command, the CREATE statement creates a Hive table that references data stored in DynamoDB. The SELECT statement then uses that table to query data stored in DynamoDB. The following example finds the largest order placed by a given customer.

```
CREATE EXTERNAL TABLE hive_purchases(customerId bigint, total_cost double, items_purchased array<String>)
STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'
```
To aggregate data using the `GROUP BY` clause

- You can use the `GROUP BY` clause to collect data across multiple records. This is often used with an aggregate function such as sum, count, min, or max. The following example returns a list of the largest orders from customers who have placed more than three orders.

```sql
CREATE EXTERNAL TABLE hive_purchases(
customerId bigint, total_cost double,
items_purchased array<String>)
STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'
TBLPROPERTIES ("dynamodb.table.name" = "Purchases",
"dynamodb.column.mapping" =
"customerId:CustomerId,total_cost:Cost,items_purchased:Items");
SELECT customerId, max(total_cost) from hive_purchases GROUP BY customerId HAVING count(*) > 3;
```

To join two DynamoDB tables

- The following example maps two Hive tables to data stored in DynamoDB. It then calls a join across those two tables. The join is computed on the cluster and returned. The join does not take place in DynamoDB. This example returns a list of customers and their purchases for customers that have placed more than two orders.

```sql
CREATE EXTERNAL TABLE hive_purchases(
customerId bigint, total_cost double,
items_purchased array<String>)
STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'
TBLPROPERTIES ("dynamodb.table.name" = "Purchases",
"dynamodb.column.mapping" =
"customerId:CustomerId,total_cost:Cost,items_purchased:Items");

CREATE EXTERNAL TABLE hive_customers(
customerId bigint, customerName string,
customerAddress array<String>)
STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'
TBLPROPERTIES ("dynamodb.table.name" = "Customers",
"dynamodb.column.mapping" =
"customerId:CustomerId, customerName:Name, customerAddress:Address");

Select c.customerId, c.customerName, count(*) as count from hive_customers c
JOIN hive_purchases p ON c.customerId=p.customerId
GROUP BY c.customerId, c.customerName HAVING count > 2;
```

To join two tables from different sources

- In the following example, Customer_S3 is a Hive table that loads a CSV file stored in Amazon S3 and hive_purchases is a table that references data in DynamoDB. The following example joins together
customer data stored as a CSV file in Amazon S3 with order data stored in DynamoDB to return a set of data that represents orders placed by customers who have "Miller" in their name.

```sql
CREATE EXTERNAL TABLE hive_purchases(
customerId bigint, total_cost double,
items_purchased array<String>
)
STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'
TBLPROPERTIES ("dynamodb.table.name" = "Purchases",
"dynamodb.column.mapping" = 
"customerId:CustomerId,total_cost:Cost,items_purchased:Items");

CREATE EXTERNAL TABLE Customer_S3(customerId bigint, customerName string,
customerAddress array<String>)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',
LOCATION 's3://bucketname/path/subpath/';

Select c.customerId, c.customerName, c.customerAddress from Customer_S3 c
JOIN hive_purchases p
ON c.customerid=p.customerid
where c.customerName like '%Miller%';
```

**Note**

In the preceding examples, the CREATE TABLE statements were included in each example for clarity and completeness. When running multiple queries or export operations against a given Hive table, you only need to create the table one time, at the beginning of the Hive session.

## Optimizing Performance for Amazon EMR Operations in DynamoDB

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Amazon EMR operations on a DynamoDB table count as read operations, and are subject to the table's provisioned throughput settings. Amazon EMR implements its own logic to try to balance the load on your DynamoDB table to minimize the possibility of exceeding your provisioned throughput. At the end of each Hive query, Amazon EMR returns information about the cluster used to process the query, including how many times your provisioned throughput was exceeded. You can use this information, as well as CloudWatch metrics about your DynamoDB throughput, to better manage the load on your DynamoDB table in subsequent requests.

The following factors influence Hive query performance when working with DynamoDB tables.

### Provisioned Read Capacity Units

When you run Hive queries against a DynamoDB table, you need to ensure that you have provisioned a sufficient amount of read capacity units.

For example, suppose that you have provisioned 100 units of Read Capacity for your DynamoDB table. This will let you perform 100 reads, or 409,600 bytes, per second. If that table contains 20GB of data (21,474,836,480 bytes), and your Hive query performs a full table scan, you can estimate how long the query will take to run:

\[
\text{time (seconds)} = \frac{\text{data size}}{\text{read throughput}} = \frac{21,474,836,480}{409,600} = 52,429 \text{ seconds} = 14.56 \text{ hours}
\]
The only way to decrease the time required would be to adjust the read capacity units on the source DynamoDB table. Adding more nodes to the Amazon EMR cluster will not help.

In the Hive output, the completion percentage is updated when one or more mapper processes are finished. For a large DynamoDB table with a low provisioned Read Capacity setting, the completion percentage output might not be updated for a long time; in the case above, the job will appear to be 0% complete for several hours. For more detailed status on your job’s progress, go to the Amazon EMR console; you will be able to view the individual mapper task status, and statistics for data reads.

You can also log on to Hadoop interface on the master node and see the Hadoop statistics. This will show you the individual map task status and some data read statistics. For more information, see the following topics:

- Web Interfaces Hosted on the Master Node
- View the Hadoop Web Interfaces

Read Percent Setting

By default, Amazon EMR manages the request load against your DynamoDB table according to your current provisioned throughput. However, when Amazon EMR returns information about your job that includes a high number of provisioned throughput exceeded responses, you can adjust the default read rate using the dynamodb.throughput.read.percent parameter when you set up the Hive table. For more information about setting the read percent parameter, see Hive Options in the Amazon EMR Developer Guide.

Write Percent Setting

By default, Amazon EMR manages the request load against your DynamoDB table according to your current provisioned throughput. However, when Amazon EMR returns information about your job that includes a high number of provisioned throughput exceeded responses, you can adjust the default write rate using the dynamodb.throughput.write.percent parameter when you set up the Hive table. For more information about setting the write percent parameter, see Hive Options in the Amazon EMR Developer Guide.

Retry Duration Setting

By default, Amazon EMR re-runs a Hive query if it has not returned a result within two minutes, the default retry interval. You can adjust this interval by setting the dynamodb.retry.duration parameter when you run a Hive query. For more information about setting the write percent parameter, see Hive Options in the Amazon EMR Developer Guide.

Number of Map Tasks

The mapper daemons that Hadoop launches to process your requests to export and query data stored in DynamoDB are capped at a maximum read rate of 1 MiB per second to limit the read capacity used. If you have additional provisioned throughput available on DynamoDB, you can improve the performance of Hive export and query operations by increasing the number of mapper daemons. To do this, you can either increase the number of EC2 instances in your cluster or increase the number of mapper daemons running on each EC2 instance.

You can increase the number of EC2 instances in a cluster by stopping the current cluster and relaunching it with a larger number of EC2 instances. You specify the number of EC2 instances in the Configure EC2 Instances dialog box if you’re launching the cluster from the Amazon EMR console, or with the --num-instances option if you’re launching the cluster from the CLI.

The number of map tasks run on an instance depends on the EC2 instance type. For more information about the supported EC2 instance types and the number of mappers each one provides, see Hadoop.
Another way to increase the number of mapper daemons is to change the `mapred.tasktracker.map.tasks.maximum` configuration parameter of Hadoop to a higher value. This has the advantage of giving you more mappers without increasing either the number or the size of EC2 instances, which saves you money. A disadvantage is that setting this value too high can cause the EC2 instances in your cluster to run out of memory. To set `mapred.tasktracker.map.tasks.maximum`, launch the cluster and specify the Configure Hadoop bootstrap action, passing in a value for `mapred.tasktracker.map.tasks.maximum` as one of the arguments of the bootstrap action. This is shown in the following example.

```
--bootstrap-action s3n://elasticmapreduce/bootstrap-actions/configure-hadoop \
--args -m,mapred.tasktracker.map.tasks.maximum=10
```

Another way to increase the number of mapper daemons is to change the `mapreduce.tasktracker.map.tasks.maximum` configuration parameter of Hadoop to a higher value. This has the advantage of giving you more mappers without increasing either the number or the size of EC2 instances, which saves you money. A disadvantage is that setting this value too high can cause the EC2 instances in your cluster to run out of memory. To set `mapreduce.tasktracker.map.tasks.maximum`, launch the cluster and specify the a configuration for Hadoop, passing in a value for `mapreduce.tasktracker.map.tasks.maximum` as one of the arguments. This is shown in the following example.

**Parallel Data Requests**

Multiple data requests, either from more than one user or more than one application to a single table may drain read provisioned throughput and slow performance.

**Process Duration**

Data consistency in DynamoDB depends on the order of read and write operations on each node. While a Hive query is in progress, another application might load new data into the DynamoDB table or modify or delete existing data. In this case, the results of the Hive query might not reflect changes made to the data while the query was running.

**Avoid Exceeding Throughput**

When running Hive queries against DynamoDB, take care not to exceed your provisioned throughput, because this will deplete capacity needed for your application's calls to `DynamoDB::Get`. To ensure that this is not occurring, you should regularly monitor the read volume and throttling on application calls to `DynamoDB::Get` by checking logs and monitoring metrics in Amazon CloudWatch.

**Request Time**

Scheduling Hive queries that access a DynamoDB table when there is lower demand on the DynamoDB table improves performance. For example, if most of your application's users live in San Francisco, you might choose to export daily data at 4 a.m. PST, when the majority of users are asleep, and not updating records in your DynamoDB database.

**Time-Based Tables**

If the data is organized as a series of time-based DynamoDB tables, such as one table per day, you can export the data when the table becomes no longer active. You can use this technique to back up data to Amazon S3 on an ongoing fashion.
Archived Data

If you plan to run many Hive queries against the data stored in DynamoDB and your application can tolerate archived data, you may want to export the data to HDFS or Amazon S3 and run the Hive queries against a copy of the data instead of DynamoDB. This conserves your read operations and provisioned throughput.

Store Avro Data in Amazon S3 Using Amazon EMR

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Avro is a data serialization system, which relies on schemas stored in JSON format to store and load data. The following procedure shows you how to take data from a flat file and store that data using Amazon S3. The procedure assumes that you have already launched a cluster with Pig installed. For more information about how to launch a cluster with Pig installed, see Submit Pig Work (p. 322). For more information about Avro, go to https://cwiki.apache.org/confluence/display/PIG/AvroStorage

To store and load data using Avro

1. Create a text file, top_nhl_scorers.txt, with this information taken from Wikipedia article, http://en.wikipedia.org/wiki/List_of_NHL_players_with_1000_points#1000-point_scorers:

   Gordie Howe Detroit Red Wings 1767 1850
   Jean Beliveau Montreal Canadiens 1125 1219
   Alex Delvecchio Detroit Red Wings 1969 1281
   Bobby Hull Chicago Black Hawks 1063 1170
   Norm Ullman Toronto Maple Leafs 1410 1229
   Stan Mikita Chicago Black Hawks 1394 1467
   Johnny Bucyk Boston Bruins 556 1369
   Frank Mahovlich Montreal Canadiens 1973 1103
   Henri Richard Montreal Canadiens 1256 1046
   Phil Esposito Boston Bruins 717 1590

   Upload the file to a bucket in Amazon S3.

2. Create an Avro schema file, top_nhl_scorers.avro, with the following structure:

   ```json
   {"namespace": "top_nhl_scorers.avro",
    "type": "record",
    "name": "Names",
    "fields": [
      {"name": "name", "type": "string"},
      {"name": "team", "type": "string"},
      {"name": "games_played", "type": "int"},
      {"name": "points", "type": "int"}
    ]
   }
   
   Upload this file to the same bucket in Amazon S3.

3. Connect to the master node of your cluster. For more information, see Connect to the Master Node Using SSH (p. 457).

4. Launch the grunt shell:

   ```
   $ pig
   ```
5. Register the JARs required to invoke the necessary storage handlers:

```
REGISTER /home/hadoop/lib/avro-1.7.4.jar;
REGISTER /home/hadoop/lib/pig/piggybank.jar;
REGISTER /home/hadoop/lib/jackson-mapper-asl-1.9.9.jar;
REGISTER /home/hadoop/lib/jackson-core-asl-1.9.9.jar;
REGISTER /home/hadoop/lib/json-simple-1.1.1.jar;
```

6. Load the source data you previously stored in your bucket:

```
data = LOAD 's3://your-bucket/hockey_stats/input/*.txt' USING PigStorage('\t') AS
    (name:chararray,team:chararray,games_played:int,points:int);
```

7. Store the data into your bucket using the AvroStorage handler:

```
STORE data INTO 's3://your-bucket/avro/output/' USING
    org.apache.pig.piggybank.storage.avro.AvroStorage('schema_file','s3://your-
        bucket/hockey_stats/schemas/top_nhl_scorers.avro');
```

8. To read Avro data, you can use the same AvroStorage handler:

```
avro_data = LOAD 's3://your-bucket/avro/output/' USING
    org.apache.pig.piggybank.storage.avro.AvroStorage();
```
Manage Clusters

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

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

After you've launched your cluster, you can monitor and manage it. Amazon EMR provides several tools you can use to connect to and control your cluster.

Topics
- View and Monitor a Cluster (p. 416)
- Connect to the Cluster (p. 456)
- Control Cluster Termination (p. 472)
- Scaling Cluster Resources (p. 477)
- Cloning a Cluster Using the Console (p. 493)
- Submit Work to a Cluster (p. 493)
- Automate Recurring Clusters with AWS Data Pipeline (p. 498)

View and Monitor a Cluster

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Amazon EMR provides several tools you can use to gather information about your cluster. You can access information about the cluster from the console, the CLI or programmatically. The standard Hadoop web interfaces and log files are available on the master node. You can also use monitoring services such as CloudWatch to track the performance of your cluster.

Topics
- View Cluster Status and Details (p. 416)
- View Application History (p. 421)
- View Log Files (p. 422)
- View Cluster Instances in Amazon EC2 (p. 426)
- CloudWatch Events and Metrics (p. 427)
- Logging Amazon EMR API Calls in AWS CloudTrail (p. 448)
- Monitor Performance with Ganglia (p. 450)

View Cluster Status and Details

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.
After you create a cluster, you can monitor its status and get detailed information about its execution and errors that may have occurred, even after it has terminated. Amazon EMR saves metadata about terminated clusters for your reference for two months, after which the metadata is deleted. Application history is saved for one week from the time it is recorded, regardless of whether the cluster is running or terminated. You can’t delete clusters from the cluster history, but using the AWS Management Console, you can use the filter, and using the AWS CLI, you can use options with the list-clusters command to focus on the clusters that you care about.

**View Cluster Status Using the AWS Management Console**

The Clusters List in the Amazon EMR console lists all the clusters in your account and region, including terminated clusters. The list shows the Name and ID of each cluster, the Status, the cluster Creation time, the Elapsed time that the cluster was running, and the Normalized instance hours that has accrued for all EC2 instances in the cluster. This list is the starting point, and it’s designed so that you can drill down into each cluster's details for analysis and troubleshooting.

**To view an abridged summary of cluster information**

- Select the down arrow next to the link for the cluster under Name.
  
  The cluster's row expands to provide more information about the cluster, hardware, steps, and bootstrap actions. Use the links in this section to drill into specifics. For example, click a link under Steps to access step log files, see the JAR associated with the step, drill into the step's jobs and tasks, and access log files.

**To view cluster status in depth**

- Choose the cluster link under Name to open a cluster details page for the cluster. Use each tab to view information as described in the following section.

  Use each tab for the following information:

<table>
<thead>
<tr>
<th>Tab</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>Use this tab to view basics of your cluster configuration, such as the URL to use for SSH connections to the master node, what open-source applications Amazon EMR installed when</td>
</tr>
</tbody>
</table>
Tab | Information
--- | ---
 | the cluster was created, where logs are stored in Amazon S3, and what version of Amazon EMR was used to create the cluster.
Application history | Use this tab to view YARN application details. For Spark jobs, you can drill down into available information about jobs, stages, and executors. For more information, see View Application History (p. 421).
Monitoring | Use this tab to view graphs depicting key indicators of cluster operation over a time period that you specify. You can view cluster-level data, node-level data, and information about I/O and data storage.
Hardware | Use this tab to view information about nodes in your cluster, including EC2 instance IDs, DNS names, and IP addresses, and more.
Events | Use this tab to view the event log for your cluster. For more information, see Monitor CloudWatch Events (p. 427).
Steps | Use this tab to see the status and access log files for steps that you submitted. For more information about steps, see Work with Steps Using the CLI and Console (p. 494).
Configurations | Use this tab to view any customized configuration objects applied to the cluster. For more information about configuration classifications, see Configuring Applications in the Amazon EMR Release Guide.
Bootstrap actions | Use this tab to view the status of any bootstrap actions the cluster runs when it launches. Bootstrap actions are used for custom software installations and advanced configuration. For more information, see (Optional) Create Bootstrap Actions to Install Additional Software (p. 129).

View Cluster Status Using the AWS CLI

The following examples demonstrate how to retrieve cluster details using the AWS CLI. For more information about available commands, see the AWS CLI Command Reference for Amazon EMR. You can use the `describe-cluster` command to view cluster-level details including status, hardware and software configuration, VPC settings, bootstrap actions, instance groups, and so on. The following example demonstrates using the `describe-cluster` command, followed by examples of the `list-clusters` command.
Example Viewing Cluster Status

To use the `describe-cluster` command, you need the cluster ID. This example demonstrates using to get a list of clusters created within a certain date range, and then using one of the cluster IDs returned to list more information about an individual cluster’s status.

The following command describes cluster `j-1K48XXXXXXHCB`, which you replace with your cluster ID.

```bash
aws emr describe-cluster --cluster-id j-1K48XXXXXXHCB
```

The output of your command is similar to the following:

```
{
  "Cluster": {
    "Status": {
      "Timeline": {
        "ReadyDateTime": 1413102659.072,
        "EndDateTime": 1413103872.89,
        "CreationDateTime": 1413102441.707
      },
      "State": "TERMINATED",
      "StateChangeReason": {
        "Message": "Terminated by user request",
        "Code": "USER_REQUEST"
      }
    },
    "Ec2InstanceAttributes": {
      "Ec2AvailabilityZone": "us-west-2a"
    },
    "Name": "Development Cluster",
    "Tags": [],
    "TerminationProtected": false,
    "RunningAmiVersion": "3.1.0",
    "NormalizedInstanceHours": 24,
    "InstanceGroups": [
      {
        "RequestedInstanceCount": 2,
        "Status": {
          "Timeline": {
            "ReadyDateTime": 1413102659.09,
            "EndDateTime": 1413103872.779,
            "CreationDateTime": 1413102441.708
          },
          "State": "TERMINATED",
          "StateChangeReason": {
            "Message": "Job flow terminated",
            "Code": "CLUSTER_TERMINATED"
          }
        },
        "Name": "CORE",
        "InstanceGroupType": "CORE",
        "InstanceType": "m3.xlarge",
        "Id": "ig-115XXXXXX52SX",
        "Market": "ON_DEMAND",
        "RunningInstanceCount": 0
      },
      {
        "RequestedInstanceCount": 1,
        "Status": {
          "Timeline": {
            "ReadyDateTime": 1413102655.968,
            "EndDateTime": 1413103872.779,
            "CreationDateTime": 1413102441.708
          },
          "State": "TERMINATED",
          "StateChangeReason": {
            "Message": "Job flow terminated",
            "Code": "CLUSTER_TERMINATED"
          }
        },
        "Name": "CORE",
        "InstanceGroupType": "CORE",
        "InstanceType": "m3.xlarge",
        "Id": "ig-115XXXXXX52SX",
        "Market": "ON_DEMAND",
        "RunningInstanceCount": 0
      }
    ]
  }
}
```
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, 2014 and October 12, 2014.

```
aws emr list-clusters --created-after 2014-10-09T00:12:00 --created-before 2014-10-12T00:12:00
```

Example Listing Clusters by State

To list clusters by state, use the `list-clusters` command with the `--cluster-states` parameter. Valid cluster states include: STARTING, BOOTSTRAPPING, RUNNING, WAITING, TERMINATING, TERMINATED, and TERMINATED_WITH_ERRORS.

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

You can also use the following shortcut parameters to list all clusters in the states specified:

- `--active` filters clusters in the STARTING, BOOTSTRAPPING, RUNNING, WAITING, or TERMINATING states.
- `--terminated` filters clusters in the TERMINATED state.
View Application History

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Using Amazon EMR version 5.8.0 or later, you can view YARN application details using the Application history tab of a cluster's detail page in the console. Using Amazon EMR application history makes it easier for you to troubleshoot and analyze active jobs and job history. Instead of setting up and connecting to the master node to view open-source troubleshooting UIs or sift through log files, you can quickly view application metrics and access relevant log files.

Application history is enabled automatically for all clusters that run YARN applications. No special setup is required. Amazon EMR keeps historical information for up to seven days after an application has completed. Detailed application history is available only for Spark. There are additional minor limitations in Amazon EMR version 5.8.0. For more information, see Known Issues for version 5.8.0 in the Amazon EMR Release Guide.

Example: View Job Details for a Spark Application

The following sequence demonstrates a drill-down through a Spark application into job detail to evaluate stages, tasks, and executors using the Application history on a cluster details page (to view cluster details, from the Clusters list, select a cluster Name).

On the Application history tab, two rows are expanded to show the diagnostic summaries for two different Spark applications, and then an Application ID is selected to view further application detail:

On the Jobs tab of YARN application details, the Description of Job 0 is selected to see Job 0 details:
On the **Job 0** details page, information about individual job stages is expanded, and then the **Description** for Stage 1 is selected to see Stage 1 details:

![Job Details](image1)

On the **Stage 1** details page, key metrics for stage tasks and executors can be seen, and task and executor logs can be viewed using links:

![Stage 1 Details](image2)

**View Log Files**

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Amazon EMR and Hadoop both produce log files that report status on the cluster. By default, these are written to the master node in the `/mnt/var/log/` directory. Depending on how you configured your cluster when you launched it, these logs may also be archived to Amazon S3 and may be viewable through the graphical debugging tool.
There are many types of logs written to the master node. Amazon EMR writes step, bootstrap action, and instance state logs. Apache Hadoop writes logs to report the processing of jobs, tasks, and task attempts. Hadoop also records logs of its daemons. For more information about the logs written by Hadoop, go to http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/ClusterSetup.html.

Topics
- View Log Files on the Master Node (p. 423)
- View Log Files Archived to Amazon S3 (p. 424)
- View Log Files in the Debugging Tool (p. 425)

View Log Files on the Master Node

The following table lists some of the log files you’ll find on the master node.

<table>
<thead>
<tr>
<th>Location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/mnt/var/log/bootstrap-actions</td>
<td>Logs written during the processing of the bootstrap actions.</td>
</tr>
<tr>
<td>/mnt/var/log/hadoop-state-pusher</td>
<td>Logs written by the Hadoop state pusher process.</td>
</tr>
<tr>
<td>/mnt/var/log/bootstrap-actions</td>
<td>Logs written during the processing of the bootstrap actions.</td>
</tr>
<tr>
<td>/mnt/var/log/hadoop-state-pusher</td>
<td>Logs written by the Hadoop state pusher process.</td>
</tr>
<tr>
<td>/mnt/var/log/hadoop-state-pusher</td>
<td>Logs written by the Hadoop state pusher process.</td>
</tr>
<tr>
<td>/mnt/var/log/instance-controller (Amazon EMR 4.6.0 and earlier)</td>
<td>Instance controller logs.</td>
</tr>
<tr>
<td>/emr/instance-controller (Amazon EMR 4.7.0 and later)</td>
<td>Instance controller logs.</td>
</tr>
<tr>
<td>/mnt/var/log/instance-state</td>
<td>Instance state logs. These contain information about the CPU, memory state, and garbage collector threads of the node.</td>
</tr>
<tr>
<td>/mnt/var/log/service-nanny (Amazon EMR 4.6.0 and earlier)</td>
<td>Logs written by the service nanny process.</td>
</tr>
<tr>
<td>/emr/service-nanny (Amazon EMR 4.7.0 and later)</td>
<td>Logs written by the service nanny process.</td>
</tr>
<tr>
<td>/mnt/var/log/hadoop</td>
<td>Hadoop logs, such as those written by the jobtracker and namenode processes.</td>
</tr>
<tr>
<td>/mnt/var/log/hadoop/steps/*</td>
<td>Step logs that contain information about the processing of the step. The value of * 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/.</td>
</tr>
</tbody>
</table>

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.
To view log files on the master node.

1. Use SSH to connect to the master node as described in Connect to the Master Node Using SSH (p. 457).
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.

   ```bash
cd /mnt/var/log/hadoop/steps/s-1234ABCDEFGH/
   ``
3. Use a text editor installed on the master node to view the contents of the log file. There are several you can choose from: vi, nano, and emacs. The following example shows how to open the controller step log using the nano text editor.

   ```bash
   nano controller
   ```

View Log Files Archived to Amazon S3

By default, Amazon EMR clusters launched using the console automatically archive log files to Amazon S3. You can specify your own log path, or you can allow the console to automatically generate a log path for you. For clusters launched using the CLI or API, you must configure Amazon S3 log archiving manually.

When Amazon EMR is configured to archive log files to Amazon S3, it stores the files in the S3 location you specified, in the `/JobFlowId/` folder, where `JobFlowId` is the cluster identifier.

The following table lists some of the log files you'll find on Amazon S3.

<table>
<thead>
<tr>
<th>Location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>/JobFlowId/daemons/</code></td>
<td>Logs written by Hadoop daemons, such as datanode and tasktracker. The logs for each node are stored in a folder labeled with the identifier of the EC2 instance of that node.</td>
</tr>
<tr>
<td><code>/JobFlowId/jobs/</code></td>
<td>Job logs and the configuration XML file for each Hadoop job.</td>
</tr>
<tr>
<td><code>/JobFlowId/node/</code></td>
<td>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.</td>
</tr>
<tr>
<td><code>/JobFlowId/steps/N/</code></td>
<td>Step logs that contain information about the processing of the step. The value of <code>N</code> indicates the stepId assigned by Amazon EMR. For example, a cluster has two steps: s-1234ABCDEFGH and s-5678IJKLmnop. The first step is</td>
</tr>
<tr>
<td>Location</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>located in /mnt/var/log/hadoop/steps/</td>
<td>The step logs written by Amazon EMR are as follows.</td>
</tr>
<tr>
<td>s-1234ABCDEFGH/</td>
<td>- controller — Information about the processing of the step. If your step fails while loading, you can find the stack trace in this log.</td>
</tr>
<tr>
<td>and the second step in /mnt/var/log/hadoop/</td>
<td>- syslog — Describes the execution of Hadoop jobs in the step.</td>
</tr>
<tr>
<td>steps/s-5678IJKLMNOP/</td>
<td>- stderr — The standard error channel of Hadoop while it processes the step.</td>
</tr>
<tr>
<td>/JobFlowId/task-attempts/</td>
<td>- stdout — The standard output channel of Hadoop while it processes the step.</td>
</tr>
</tbody>
</table>

To view log files archived to Amazon S3 using the console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Open the S3 bucket specified when you configured the cluster to archive log files in Amazon S3.
3. Navigate to the log file containing the information to display. The preceding table gives a list of the types of log files that are available and where you will find them.
4. Double-click on a log file to view it in the browser.

If you don’t want to view the log files in the Amazon S3 console, you can download the files from Amazon S3 to your local machine using a tool such as the Amazon S3 Organizer plug-in for the Firefox web browser, or by writing an application to retrieve the objects from Amazon S3. For more information, see Getting Objects in the Amazon Simple Storage Service Developer Guide.

View Log Files in the Debugging Tool

Amazon EMR does not automatically enable the debugging tool. You must configure this when you launch the cluster.

To view cluster logs using the console

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. From the Cluster List page, choose the details icon next to the cluster you want to view.
   This brings up the Cluster Details page. In the Steps section, the links to the right of each step display the various types of logs available for the step. These logs are generated by Amazon EMR.
3. To view a list of the Hadoop jobs associated with a given step, choose the View Jobs link to the right of the step.
4. To view a list of the Hadoop tasks associated with a given job, choose the View Tasks link to the right of the job.
5. To view a list of the attempts a given task has run while trying to complete, choose the View Attempts link to the right of the task.

6. To view the logs generated by a task attempt, choose the stderr, stdout, and syslog links to the right of the task attempt.

The debugging tool displays links to the log files after Amazon EMR uploads the log files to your bucket on Amazon S3. Because log files are uploaded to Amazon S3 every 5 minutes, it can take a few minutes for the log file uploads to complete after the step completes.

Amazon EMR periodically updates the status of Hadoop jobs, tasks, and task attempts in the debugging tool. You can click Refresh List in the debugging panes to get the most up-to-date status of these items.

View Cluster Instances in Amazon EC2

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

To help you manage your resources, Amazon EC2 allows you to assign metadata to resources in the form of tags. Each Amazon EC2 tag consists of a key and a value. Tags allow you to categorize your Amazon EC2 resources in different ways: for example, by purpose, owner, or environment.

You can search and filter resources based on the tags. The tags assigned using your AWS account are available only to you. Other accounts sharing the resource cannot view your tags.
Amazon EMR automatically tags each EC2 instance it launches with key-value pairs that identify the cluster and the instance group to which the instance belongs. This makes it easy to filter your EC2 instances to show, for example, only those instances belonging to a particular cluster or to show all of the currently running instances in the task-instance group. This is especially useful if you are running several clusters concurrently or managing large numbers of EC2 instances.

These are the predefined key-value pairs that Amazon EMR assigns:

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>aws:elasticmapreduce:job-flow-id</td>
<td>&lt;job-flow-identifier&gt;</td>
</tr>
<tr>
<td>aws:elasticmapreduce:instance-group-role</td>
<td>&lt;group-role&gt;</td>
</tr>
</tbody>
</table>

The values are further defined as follows:

- The `<job-flow-identifier>` is the ID of the cluster the instance is provisioned for. It appears in the format j-XXXXXXXXXXXXXXX.
- The `<group-role>` is one of the following values: master, core, or task. These values correspond to the master instance group, core instance group, and task instance group.

You can view and filter on the tags that Amazon EMR adds. For more information, see Using Tags in the Amazon EC2 User Guide for Linux Instances. Because the tags set by Amazon EMR are system tags and cannot be edited or deleted, the sections on displaying and filtering tags are the most relevant.

**Note**

Amazon EMR adds tags to the EC2 instance when its status is updated to running. If there's a latency period between the time the EC2 instance is provisioned and the time its status is set to running, the tags set by Amazon EMR do not appear until the instance starts. If you don't see the tags, wait for a few minutes and refresh the view.

CloudWatch Events and Metrics

You can use events and metrics to track the activity and health of an Amazon EMR cluster, viewing events and metrics quickly in the Amazon EMR console for a single cluster, and viewing events for all clusters in a region. You can use CloudWatch Events to define an action to take when Amazon EMR generates an event that matches a pattern that you specify, and you can also use CloudWatch to monitor metrics.

Events are useful for monitoring a specific occurrence within a cluster—for example, when a cluster changes state from starting to running. Metrics are useful for monitoring a specific value—for example, the percentage of available disk space that HDFS is using within a cluster.

For more information about CloudWatch Events, see the Amazon CloudWatch Events User Guide. For more information about CloudWatch metrics, see Using Amazon CloudWatch Metrics and Creating Amazon CloudWatch Alarms in the Amazon CloudWatch User Guide.

**Topics**

- Monitor CloudWatch Events (p. 427)
- Monitor Metrics with CloudWatch (p. 434)

Monitor CloudWatch Events

Amazon EMR tracks events and keeps information about them for up to seven days. Changes in the state of clusters, instance groups, automatic scaling policies, and steps cause an event to be recorded. Each
event has information such as the date and time the event occurred, along with further detail about the event, such as the cluster or instance group affected.

The following table lists Amazon EMR events, along with the state or state change that the event indicates, the severity of the event, and event messages. Each event is represented as a JSON object that is sent automatically to an event stream. The JSON object includes further detail about the event. The JSON object is particularly important when you set up rules for event processing using CloudWatch Events because rules seek to match patterns in the JSON object. For more information, see Events and Event Patterns and Amazon EMR Events in the Amazon CloudWatch Events User Guide.

## Cluster Events

<table>
<thead>
<tr>
<th>State or State Change</th>
<th>Severity</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>STARTING</td>
<td>INFO</td>
<td>Amazon EMR cluster <strong>ClusterId (ClusterName)</strong> was requested at <strong>Time</strong> and is being created.</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td></td>
<td>Applies only to clusters with the instance fleets configuration and multiple subnets selected within a VPC.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Amazon EMR cluster <strong>ClusterId (ClusterName)</strong> is being created in subnet (<strong>SubnetName</strong>) in VPC (<strong>VPCName</strong>) in availability zone (<strong>AvailabilityZoneID</strong>), which was chosen from the specified VPC options.</td>
</tr>
<tr>
<td>STARTING</td>
<td>INFO</td>
<td>Amazon EMR cluster <strong>ClusterId (ClusterName)</strong> is being created in availability zone (<strong>AvailabilityZoneID</strong>), which was chosen from the specified availability zone options.</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td></td>
<td>Applies only to clusters with the instance fleets configuration and multiple Availability Zones selected within EC2-Classic.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Amazon EMR cluster <strong>ClusterId (ClusterName)</strong> is being created in availability zone (<strong>AvailabilityZoneID</strong>), which was chosen from the specified availability zone options.</td>
</tr>
<tr>
<td>RUNNING</td>
<td>INFO</td>
<td>Amazon EMR cluster <strong>ClusterId (ClusterName)</strong> began running steps at <strong>Time</strong>.</td>
</tr>
<tr>
<td>WAITING</td>
<td>INFO</td>
<td>Amazon EMR cluster <strong>ClusterId (ClusterName)</strong> was created at <strong>Time</strong> and is ready for use.</td>
</tr>
</tbody>
</table>

—or—
### State or State Change | Severity | Message
--- | --- | ---
| | | Amazon EMR cluster **ClusterId (ClusterName)** finished running all pending steps at **Time**.
| **Note** | | A cluster in the **WAITING** state may nevertheless be processing jobs.

#### TERMINATED
The severity depends on the reason for the state change, as shown in the following:
- **CRITICAL** if the cluster terminated with any of the following state change reasons: **INTERNAL_ERROR**, **VALIDATION_ERROR**, **INSTANCE_FAILURE**, **BOOTSTRAP_FAILURE**, or **STEP_FAILURE**.
- **INFO** if the cluster terminated with any of the following state change reasons: **USER_REQUEST** or **ALL_STEPS_COMPLETED**.

#### TERMINATED_WITH_ERRORS
**CRITICAL**

Amazon EMR Cluster **ClusterId (ClusterName)** has terminated with errors at **Time** with a reason of **StateChangeReason:Code**.

### Instance Fleet Events
**Note**
The instance fleets configuration is available only in Amazon EMR versions 4.8.0 and later, excluding 5.0.x versions.

#### State or State Change | Severity | Message
--- | --- | ---
| From **PROVISIONING** to **WAITING** | **INFO** | Provisioning for instance fleet **InstanceFleetID** in Amazon EMR cluster **ClusterId (ClusterName)** is complete. Provisioning started at **Time** and took **Num** minutes. The instance fleet now has On-Demand capacity of **Num** and Spot capacity of **Num**. Target On-Demand capacity was **Num**, and target Spot capacity was **Num**.
<table>
<thead>
<tr>
<th>State or State Change</th>
<th>Severity</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>From WAITING to RESIZING</td>
<td>INFO</td>
<td>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.</td>
</tr>
<tr>
<td>From RESIZING to WAITING</td>
<td>INFO</td>
<td>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.</td>
</tr>
<tr>
<td>From RESIZING to WAITING</td>
<td>WARN</td>
<td>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.</td>
</tr>
<tr>
<td>ARRESTED</td>
<td>ERROR</td>
<td>Instance fleet InstanceFleetID in Amazon EMR cluster ClusterId (ClusterName) was arrested at Time for the following reason: ReasonDesc.</td>
</tr>
<tr>
<td>RESIZING</td>
<td>WARNING</td>
<td>The resizing operation for instance fleet InstanceFleetID in Amazon EMR cluster ClusterId (ClusterName) is stuck for the following reason: ReasonDesc.</td>
</tr>
<tr>
<td>WAITING or RUNNING</td>
<td>INFO</td>
<td>A resize for instance fleet InstanceFleetID in Amazon EMR cluster ClusterId (ClusterName) was initiated by Entity at Time.</td>
</tr>
</tbody>
</table>
### Instance Group Events

<table>
<thead>
<tr>
<th>State or State Change</th>
<th>Severity</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>From RESIZING to RUNNING</td>
<td>INFO</td>
<td>The resizing operation for instance group <code>InstanceGroupId</code> in Amazon EMR cluster <code>ClusterId (ClusterName)</code> is complete. It now has an instance count of <code>Num</code>. The resize started at <code>Time</code> and took <code>Num</code> minutes to complete.</td>
</tr>
<tr>
<td>From RUNNING to RESIZING</td>
<td>INFO</td>
<td>A resize for instance group <code>InstanceGroupId</code> in Amazon EMR cluster <code>ClusterId (ClusterName)</code> started at <code>Time</code>. It is resizing from an instance count of <code>Num</code> to <code>Num</code>.</td>
</tr>
<tr>
<td>ARRESTED</td>
<td>ERROR</td>
<td>Instance group <code>InstanceGroupId</code> in Amazon EMR cluster <code>ClusterId (ClusterName)</code> was arrested at <code>Time</code> for the following reason: <code>ReasonDesc</code>.</td>
</tr>
<tr>
<td>RESIZING</td>
<td>WARNING</td>
<td>The resizing operation for instance group <code>InstanceGroupId</code> in Amazon EMR cluster <code>ClusterId (ClusterName)</code> is stuck for the following reason: <code>ReasonDesc</code>.</td>
</tr>
<tr>
<td>WAITING or RUNNING</td>
<td>INFO</td>
<td>A resize for instance group <code>InstanceGroupId</code> in Amazon EMR cluster <code>ClusterId (ClusterName)</code> was initiated by <code>Entity</code> at <code>Time</code>.</td>
</tr>
</tbody>
</table>

### Automatic Scaling Policy Events

<table>
<thead>
<tr>
<th>State or State Change</th>
<th>Severity</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>PENDING</td>
<td>INFO</td>
<td>An Auto Scaling policy was added to instance group <code>InstanceGroupId</code> in Amazon EMR cluster <code>ClusterId (ClusterName)</code> at <code>Time</code>. The policy is pending attachment.</td>
</tr>
<tr>
<td></td>
<td>—or—</td>
<td>The Auto Scaling policy for instance group</td>
</tr>
</tbody>
</table>

431
### State or State Change

<table>
<thead>
<tr>
<th>State or State Change</th>
<th>Severity</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATTACHED</td>
<td>INFO</td>
<td>The Auto Scaling policy for instance group InstanceGroupId in Amazon EMR cluster ClusterId (ClusterName) was attached at Time.</td>
</tr>
<tr>
<td>DETACHED</td>
<td>INFO</td>
<td>The Auto Scaling policy for instance group InstanceGroupId in Amazon EMR cluster ClusterId (ClusterName) was detached at Time.</td>
</tr>
</tbody>
</table>
| 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

<table>
<thead>
<tr>
<th>State or State Change</th>
<th>Severity</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>PENDING</td>
<td>INFO</td>
<td>Step StepId (StepName) was added to Amazon EMR cluster ClusterId (ClusterName) at Time and is pending execution.</td>
</tr>
<tr>
<td>CANCEL_PENDING</td>
<td>WARN</td>
<td>Step StepId (StepName) in Amazon EMR cluster ClusterId (ClusterName) was cancelled at Time and is pending cancellation.</td>
</tr>
<tr>
<td>RUNNING</td>
<td>INFO</td>
<td>Step StepId (StepName) in Amazon EMR cluster ClusterId</td>
</tr>
<tr>
<td>State or State Change</td>
<td>Severity</td>
<td>Message</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------</td>
<td>---------</td>
</tr>
<tr>
<td>COMPLETED</td>
<td>INFO</td>
<td>(ClusterName) started running at Time.</td>
</tr>
<tr>
<td>CANCELLED</td>
<td>WARN</td>
<td>Cancellation request has succeeded for cluster step StepID (StepName) in Amazon EMR cluster ClusterId (ClusterName) at Time, and the step is now cancelled.</td>
</tr>
<tr>
<td>FAILED</td>
<td>ERROR</td>
<td>Step StepID (StepName) in Amazon EMR cluster ClusterId (ClusterName) failed at Time.</td>
</tr>
</tbody>
</table>

### Viewing Events Using the Amazon EMR Console

For each cluster, you can view a simple list of events in the details pane, which lists events in descending order of occurrence. You can also view all events for all clusters in a region in descending order of occurrence.

**Note**

If you don’t want a user to see all cluster events for a region, add a statement that denies permission ("Effect": "Deny") for the elasticmapreduce:ViewEventsFromAllClustersInConsole action to a policy that is attached to the user.

**To view events for all clusters in a region**

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

**To view events for a particular cluster**

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. Choose Cluster List, select a cluster, and then choose View details.
3. Choose Events in the cluster details pane.
Creating Rules for Amazon EMR Events Using CloudWatch

Amazon EMR automatically sends events to a CloudWatch event stream. You can create rules that match events according to a specified pattern, and route the events to targets to take action, such as sending an email notification. Patterns are matched against the event JSON object. For more information about Amazon EMR event details, see Amazon EMR Events in the Amazon CloudWatch Events User Guide.

To create a rule for an Amazon EMR event using the CloudWatch console

2. In the navigation pane, choose Rules, Create rule.
3. For Event source, choose Amazon EMR.
4. Choose event states and other details according to your requirements for event handling. To create a rule by modifying the JSON according to the guidelines in Events and Event Patterns, choose Show advanced options, edit.
5. Select a target and add additional targets according to your requirements for event handling.
6. Choose Configure details, provide rule definition details, and then choose Create rule.

Monitor Metrics with CloudWatch

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Metrics are updated every five minutes and automatically collected and pushed to CloudWatch for every EMR cluster. This interval is not configurable. There is no charge for the Amazon EMR metrics reported in CloudWatch. Metrics are archived for two weeks, after which the data is discarded.

Note

Viewing Amazon EMR metrics in CloudWatch is supported only for clusters launched with AMI 2.0.3 or later and running Hadoop 0.20.205 or later. For more information about selecting the AMI version for your cluster, see Choose an Amazon Machine Image (AMI) (p. 69).
How Do I Use Amazon EMR Metrics?

The metrics reported by Amazon EMR provide information that you can analyze in different ways. The table below shows some common uses for the metrics. These are suggestions to get you started, not a comprehensive list. For the complete list of metrics reported by Amazon EMR, see Metrics Reported by Amazon EMR in CloudWatch (p. 438).

<table>
<thead>
<tr>
<th>How do I?</th>
<th>Relevant Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Track the progress of my cluster</td>
<td>Look at the RunningMapTasks, RemainingMapTasks, RunningReduceTasks, and RemainingReduceTasks metrics.</td>
</tr>
<tr>
<td>Detect clusters that are idle</td>
<td>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.</td>
</tr>
<tr>
<td>Detect when a node runs out of storage</td>
<td>The HDFSUtilization metric is the percentage of disk space currently used. If this rises above an acceptable level for your application, such as 80% of capacity used, you may need to resize your cluster and add more core nodes.</td>
</tr>
</tbody>
</table>

Accessing CloudWatch Metrics

There are many ways to access the metrics that Amazon EMR pushes to CloudWatch. You can view them through either the Amazon EMR console or CloudWatch console, or you can retrieve them using the CloudWatch CLI or the CloudWatch API. The following procedures show you how to access the metrics using these various tools.

To view metrics in the Amazon EMR console

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. To view metrics for a cluster, select a cluster to display the Summary pane.
3. Choose Monitoring to view information about that cluster. Choose any one of the tabs named Cluster Status, Map/Reduce, Node Status, IO, or HBase to load the reports about the progress and health of the cluster.
4. After you choose a metric to view, you can select a graph size. Edit Start and End fields to filter the metrics to a specific time frame.

To view metrics in the CloudWatch console

2. In the navigation pane, choose EMR.
3. Scroll down to the metric to graph. You can search on the cluster identifier of the cluster to monitor.
4. Open a metric to display the graph.
To access metrics from the CloudWatch CLI

- Call `mon-get-stats`. For more information, see the Amazon CloudWatch User Guide.

To access metrics from the CloudWatch API

- Call `GetMetricStatistics`. For more information, see Amazon CloudWatch API Reference.

### Setting Alarms on Metrics

Amazon EMR pushes metrics to CloudWatch, which means you can use CloudWatch to set alarms on your Amazon EMR metrics. You can, for example, configure an alarm in CloudWatch to send you an email any time the HDFS utilization rises above 80%.

The following topics give you a high-level overview of how to set alarms using CloudWatch. For detailed instructions, see Create or Edit a CloudWatch Alarm in the Amazon CloudWatch User Guide.

#### Set alarms using the CloudWatch console

2. Choose Create Alarm. This launches the Create Alarm Wizard.
3. Choose **EMR Metrics** and scroll through the Amazon EMR metrics to locate the metric you want to place an alarm on. An easy way to display just the Amazon EMR metrics in this dialog box is to search on the cluster identifier of your cluster. Select the metric to create an alarm on and choose **Next**.

4. Fill in the **Name**, **Description**, **Threshold**, and **Time** values for the metric.

5. If you want CloudWatch to send you an email when the alarm state is reached, in the **Whenever this alarm:** field, choose **State is ALARM**. For **Send notification to:** select an existing SNS topic. If you choose **Create topic**, you can set the name and email addresses for a new email subscription list. This list is saved and appears in the field for future alarms.

   **Note**
   If you use **Create topic** to create a new Amazon SNS topic, the email addresses must be verified before they receive notifications. Emails are only sent when the alarm enters an alarm state. If this alarm state change happens before the email addresses are verified, they do not receive a notification.

6. At this point, the **Define Alarm** screen gives you a chance to review the alarm that you’re about to create. Choose **Create Alarm**.

   **Note**
   For more information about how to set alarms using the CloudWatch console, see **Create an Alarm that Sends Email** in the **Amazon CloudWatch User Guide**.

**To set an alarm using the CloudWatch API**

- Call `mon-put-metric-alarm`. For more information, see **Amazon CloudWatch User Guide**.

**To set an alarm using the CloudWatch API**

- Call `PutMetricAlarm`. For more information, see **Amazon CloudWatch API Reference**

**Metrics Reported by Amazon EMR in CloudWatch**

The following table lists all the metrics that Amazon EMR reports in the console and pushes to CloudWatch.

**Amazon EMR Metrics**

Amazon EMR sends data for several metrics to CloudWatch. All Amazon EMR clusters automatically send metrics in five-minute intervals. Metrics are archived for two weeks; after that period, the data is discarded.

The **AWS/ElasticMapReduce** namespace includes the following metrics.

   **Note**
   Amazon EMR pulls metrics from a cluster. If a cluster becomes unreachable, no metrics are reported until the cluster becomes available again.

The following are Hadoop 1 metrics:

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster Status</td>
<td>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</td>
</tr>
<tr>
<td>Metric</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>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</td>
</tr>
<tr>
<td>JobsRunning</td>
<td>The number of jobs in the cluster that are currently running. Use case: Monitor cluster health Units: Count</td>
</tr>
<tr>
<td>JobsFailed</td>
<td>The number of jobs in the cluster that have failed. Use case: Monitor cluster health Units: Count</td>
</tr>
<tr>
<td>Map/Reduce</td>
<td></td>
</tr>
<tr>
<td>MapTasksRunning</td>
<td>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</td>
</tr>
<tr>
<td>MapTasksRemaining</td>
<td>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</td>
</tr>
<tr>
<td>MapSlotsOpen</td>
<td>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</td>
</tr>
<tr>
<td>RemainingMapTasksPerSlot</td>
<td>The ratio of the total map tasks remaining to the total map slots available in the cluster. Use case: Analyze cluster performance Units: Ratio</td>
</tr>
</tbody>
</table>

439
<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReduceTasksRunning</td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>Units: Count</td>
</tr>
<tr>
<td>ReduceTasksRemaining</td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>Units: Count</td>
</tr>
<tr>
<td>ReduceSlotsOpen</td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>Units: Count</td>
</tr>
<tr>
<td><strong>Node Status</strong></td>
<td></td>
</tr>
<tr>
<td>CoreNodesRunning</td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>Units: Count</td>
</tr>
<tr>
<td>CoreNodesPending</td>
<td>The number of core nodes waiting to be assigned. All of the core nodes requested may not be immediately available; this metric reports the pending requests. Data points for this metric are reported only when a corresponding instance group exists. Use case: Monitor cluster health</td>
</tr>
<tr>
<td></td>
<td>Units: Count</td>
</tr>
<tr>
<td>LiveDataNodes</td>
<td>The percentage of data nodes that are receiving work from Hadoop. Use case: Monitor cluster health</td>
</tr>
<tr>
<td></td>
<td>Units: Percent</td>
</tr>
<tr>
<td>TaskNodesRunning</td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>Units: Count</td>
</tr>
<tr>
<td>Metric</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TaskNodesPending</td>
<td>The number of core nodes waiting to be assigned. All of the task nodes requested may not be immediately available; this metric reports the pending requests. Data points for this metric are reported only when a corresponding instance group exists. Use case: Monitor cluster health Units: Count</td>
</tr>
<tr>
<td>LiveTaskTrackers</td>
<td>The percentage of task trackers that are functional. Use case: Monitor cluster health Units: Percent</td>
</tr>
<tr>
<td>IO</td>
<td></td>
</tr>
<tr>
<td>S3BytesWritten</td>
<td>The number of bytes written to Amazon S3. This metric aggregates MapReduce jobs only, and does not apply for other workloads on EMR. Use case: Analyze cluster performance, Monitor cluster progress Units: Bytes</td>
</tr>
<tr>
<td>S3BytesRead</td>
<td>The number of bytes read from Amazon S3. This metric aggregates MapReduce jobs only, and does not apply for other workloads on EMR. Use case: Analyze cluster performance, Monitor cluster progress Units: Bytes</td>
</tr>
<tr>
<td>HDFSUtilization</td>
<td>The percentage of HDFS storage currently used. Use case: Analyze cluster performance Units: Percent</td>
</tr>
<tr>
<td>HDFSBytesRead</td>
<td>The number of bytes read from HDFS. Use case: Analyze cluster performance, Monitor cluster progress Units: Bytes</td>
</tr>
<tr>
<td>HDFSBytesWritten</td>
<td>The number of bytes written to HDFS. Use case: Analyze cluster performance, Monitor cluster progress Units: Bytes</td>
</tr>
</tbody>
</table>
### CloudWatch Events and Metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MissingBlocks</td>
<td>The number of blocks in which HDFS has no replicas. These might be corrupt blocks. Use case: Monitor cluster health. Units: Count</td>
</tr>
<tr>
<td>TotalLoad</td>
<td>The current, total number of readers and writers reported by all DataNodes in a cluster. Use case: Diagnose the degree to which high I/O might be contributing to poor job execution performance. Worker nodes running the DataNode daemon must also perform map and reduce tasks. Persistently high TotalLoad values over time can indicate that high I/O might be a contributing factor to poor performance. Occasional spikes in this value are typical and do not usually indicate a problem. Units: Count</td>
</tr>
</tbody>
</table>

**HBase**

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BackupFailed</td>
<td>Whether the last backup failed. This is set to 0 by default and updated to 1 if the previous backup attempt failed. This metric is only reported for HBase clusters. Use case: Monitor HBase backups Units: Count</td>
</tr>
<tr>
<td>MostRecentBackupDuration</td>
<td>The amount of time it took the previous backup to complete. This metric is set regardless of whether the last completed backup succeeded or failed. While the backup is ongoing, this metric returns the number of minutes after the backup started. This metric is only reported for HBase clusters. Use case: Monitor HBase Backups Units: Minutes</td>
</tr>
<tr>
<td>TimeSinceLastSuccessfulBackup</td>
<td>The number of elapsed minutes after the last successful HBase backup started on your cluster. This metric is only reported for HBase clusters. Use case: Monitor HBase backups Units: Minutes</td>
</tr>
</tbody>
</table>

The following metrics are available for Hadoop 2 AMIs:

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster Status</td>
<td></td>
</tr>
<tr>
<td>IsIdle</td>
<td>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</td>
</tr>
<tr>
<td>Metric</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>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</td>
<td></td>
</tr>
<tr>
<td>ContainerAllocated</td>
<td>The number of resource containers allocated by the ResourceManager. Use case: Monitor cluster progress Units: Count</td>
</tr>
<tr>
<td>ContainerReserved</td>
<td>The number of containers reserved. Use case: Monitor cluster progress Units: Count</td>
</tr>
<tr>
<td>ContainerPending</td>
<td>The number of containers in the queue that have not yet been allocated. Use case: Monitor cluster progress Units: Count</td>
</tr>
<tr>
<td>ContainerPendingRatio</td>
<td>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.</td>
</tr>
<tr>
<td>AppsCompleted</td>
<td>The number of applications submitted to YARN that have completed. Use case: Monitor cluster progress Units: Count</td>
</tr>
<tr>
<td>AppsFailed</td>
<td>The number of applications submitted to YARN that have failed to complete. Use case: Monitor cluster progress, Monitor cluster health Units: Count</td>
</tr>
<tr>
<td>Metric</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AppsKilled</td>
<td>The number of applications submitted to YARN that have been killed.</td>
</tr>
<tr>
<td></td>
<td>Use case: Monitor cluster progress, Monitor cluster health</td>
</tr>
<tr>
<td></td>
<td>Units: <em>Count</em></td>
</tr>
<tr>
<td>AppsPending</td>
<td>The number of applications submitted to YARN that are in a pending state.</td>
</tr>
<tr>
<td></td>
<td>Use case: Monitor cluster progress</td>
</tr>
<tr>
<td></td>
<td>Units: <em>Count</em></td>
</tr>
<tr>
<td>AppsRunning</td>
<td>The number of applications submitted to YARN that are running.</td>
</tr>
<tr>
<td></td>
<td>Use case: Monitor cluster progress</td>
</tr>
<tr>
<td></td>
<td>Units: <em>Count</em></td>
</tr>
<tr>
<td>AppsSubmitted</td>
<td>The number of applications submitted to YARN.</td>
</tr>
<tr>
<td></td>
<td>Use case: Monitor cluster progress</td>
</tr>
<tr>
<td></td>
<td>Units: <em>Count</em></td>
</tr>
<tr>
<td><strong>Node Status</strong></td>
<td><em>CoreNodesRunning</em></td>
</tr>
<tr>
<td>CoreNodesRunning</td>
<td>The number of core nodes working. Data points for this metric are reported</td>
</tr>
<tr>
<td></td>
<td>only when a corresponding instance group exists.</td>
</tr>
<tr>
<td></td>
<td>Use case: Monitor cluster health</td>
</tr>
<tr>
<td></td>
<td>Units: <em>Count</em></td>
</tr>
<tr>
<td>CoreNodesPending</td>
<td>The number of core nodes waiting to be assigned. All of the core nodes</td>
</tr>
<tr>
<td></td>
<td>requested may not be immediately available; this metric reports the</td>
</tr>
<tr>
<td></td>
<td>pending requests. Data points for this metric are reported only when a</td>
</tr>
<tr>
<td></td>
<td>corresponding instance group exists.</td>
</tr>
<tr>
<td></td>
<td>Use case: Monitor cluster health</td>
</tr>
<tr>
<td></td>
<td>Units: <em>Count</em></td>
</tr>
<tr>
<td>LiveDataNodes</td>
<td>The percentage of data nodes that are receiving work from Hadoop.</td>
</tr>
<tr>
<td></td>
<td>Use case: Monitor cluster health</td>
</tr>
<tr>
<td></td>
<td>Units: <em>Percent</em></td>
</tr>
<tr>
<td>MRTotalNodes</td>
<td>The number of nodes presently available to MapReduce jobs. Equivalent to</td>
</tr>
<tr>
<td></td>
<td>YARN metric mapred.resourcemanager.TotalNodes.</td>
</tr>
<tr>
<td></td>
<td>Use case: Monitor cluster progress</td>
</tr>
<tr>
<td></td>
<td>Units: <em>Count</em></td>
</tr>
<tr>
<td>Metric</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MRActiveNodes</td>
<td>The number of nodes presently running MapReduce tasks or jobs. Equivalent to YARN metric mapred.resourceManager.NoOfActiveNodes. Use case: Monitor cluster progress. Units: Count.</td>
</tr>
<tr>
<td>MRLostNodes</td>
<td>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.</td>
</tr>
<tr>
<td>MRUnhealthyNodes</td>
<td>The number of nodes available to MapReduce jobs marked in an UNHEALTHY state. Equivalent to YARN metric mapred.resourceManager.NoOfUnhealthyNodes. Use case: Monitor cluster progress. Units: Count.</td>
</tr>
<tr>
<td>MRDecommissionedNodes</td>
<td>The number of nodes allocated to MapReduce applications that have been marked in a DECOMMISSIONED state. Equivalent to YARN metric mapred.resourceManager.NoOfDecommissionedNodes. Use case: Monitor cluster health, Monitor cluster progress. Units: Count.</td>
</tr>
<tr>
<td>MRRestartedNodes</td>
<td>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.</td>
</tr>
<tr>
<td>IO</td>
<td></td>
</tr>
<tr>
<td>S3BytesWritten</td>
<td>The number of bytes written to Amazon S3. Use case: Analyze cluster performance, Monitor cluster progress. Units: Bytes.</td>
</tr>
<tr>
<td>S3BytesRead</td>
<td>The number of bytes read from Amazon S3. Use case: Analyze cluster performance, Monitor cluster progress. Units: Bytes.</td>
</tr>
<tr>
<td>Metric</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>HDFSUtilization</td>
<td>The percentage of HDFS storage currently used. Use case: Analyze cluster performance</td>
</tr>
<tr>
<td></td>
<td>Units: Percent</td>
</tr>
<tr>
<td>HDFSBytesRead</td>
<td>The number of bytes read from HDFS. This metric aggregates MapReduce jobs only, and does not apply for other workloads on EMR. Use case: Analyze cluster performance, Monitor cluster progress</td>
</tr>
<tr>
<td></td>
<td>Units: Bytes</td>
</tr>
<tr>
<td>HDFSBytesWritten</td>
<td>The number of bytes written to HDFS. This metric aggregates MapReduce jobs only, and does not apply for other workloads on EMR. Use case: Analyze cluster performance, Monitor cluster progress</td>
</tr>
<tr>
<td></td>
<td>Units: Bytes</td>
</tr>
<tr>
<td>MissingBlocks</td>
<td>The number of blocks in which HDFS has no replicas. These might be corrupt blocks. Use case: Monitor cluster health</td>
</tr>
<tr>
<td></td>
<td>Units: Count</td>
</tr>
<tr>
<td>CorruptBlocks</td>
<td>The number of blocks that HDFS reports as corrupted. Use case: Monitor cluster health</td>
</tr>
<tr>
<td></td>
<td>Units: Count</td>
</tr>
<tr>
<td>TotalLoad</td>
<td>The total number of concurrent data transfers. Use case: Monitor cluster health</td>
</tr>
<tr>
<td></td>
<td>Units: Count</td>
</tr>
<tr>
<td>MemoryTotalMB</td>
<td>The total amount of memory in the cluster. Use case: Monitor cluster progress</td>
</tr>
<tr>
<td></td>
<td>Units: Bytes</td>
</tr>
<tr>
<td>MemoryReservedMB</td>
<td>The amount of memory reserved. Use case: Monitor cluster progress</td>
</tr>
<tr>
<td></td>
<td>Units: Bytes</td>
</tr>
</tbody>
</table>

446
<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MemoryAvailableMB</td>
<td>The amount of memory available to be allocated. Use case: Monitor cluster progress Units: Bytes</td>
</tr>
<tr>
<td>YARNMemoryAvailablePercentage</td>
<td>The percentage of remaining memory available to YARN (YARNMemoryAvailablePercentage = MemoryAvailableMB / MemoryTotalMB). This value is useful for scaling cluster resources based on YARN memory usage.</td>
</tr>
<tr>
<td>MemoryAllocatedMB</td>
<td>The amount of memory allocated to the cluster. Use case: Monitor cluster progress Units: Bytes</td>
</tr>
<tr>
<td>PendingDeletionBlocks</td>
<td>The number of blocks marked for deletion. Use case: Monitor cluster progress, Monitor cluster health Units: Count</td>
</tr>
<tr>
<td>UnderReplicatedBlocks</td>
<td>The number of blocks that need to be replicated one or more times. Use case: Monitor cluster progress, Monitor cluster health Units: Count</td>
</tr>
<tr>
<td>DfsPendingReplicationBlocks</td>
<td>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</td>
</tr>
<tr>
<td>CapacityRemainingGB</td>
<td>The amount of remaining HDFS disk capacity. Use case: Monitor cluster progress, Monitor cluster health Units: Bytes</td>
</tr>
</tbody>
</table>

**HBase**

| HbaseBackupFailed            | Whether the last backup failed. This is set to 0 by default and updated to 1 if the previous backup attempt failed. This metric is only reported for HBase clusters. Use case: Monitor HBase backups Units: Count |

---

447
### Metric

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
</table>
| **MostRecentBackupDuration** | The amount of time it took the previous backup to complete. This metric is set regardless of whether the last completed backup succeeded or failed. While the backup is ongoing, this metric returns the number of minutes after the backup started. This metric is only reported for HBase clusters.  
Use case: Monitor HBase Backups  
Units: Minutes |
| **TimeSinceLastSuccessfulBackup** | The number of elapsed minutes after the last successful HBase backup started on your cluster. This metric is only reported for HBase clusters.  
Use case: Monitor HBase backups  
Units: Minutes |

### Dimensions for Amazon EMR Metrics

Amazon EMR data can be filtered using any of the dimensions in the following table.

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>JobFlowId</strong></td>
<td>The same as cluster ID, which is the unique identifier of a cluster in the form <code>j-XXXXXXXXXXXXX</code>. Find this value by clicking on the cluster in the Amazon EMR console.</td>
</tr>
<tr>
<td><strong>JobId</strong></td>
<td>The identifier of a job within a cluster. You can use this to filter the metrics returned from a cluster down to those that apply to a single job within the cluster. JobId takes the form <code>job_XXXXXXXXXXXX_XXXX</code>.</td>
</tr>
</tbody>
</table>

### Logging Amazon EMR API Calls in AWS CloudTrail

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Amazon EMR is integrated with AWS CloudTrail, a service that captures API calls made by or on behalf of your AWS account. This information is collected and written to log files that are stored in an Amazon S3 bucket that you specify. API calls are logged when you use the Amazon EMR API, the Amazon EMR console, a back-end console, or the AWS CLI. Using the information collected by CloudTrail, you can determine what request was made to Amazon EMR, the source IP address the request was made from, who made the request, when it was made, and so on.

To learn more about CloudTrail, including how to configure and enable it, see the [AWS CloudTrail User Guide](#).

**Topics**

- [Amazon EMR Information in CloudTrail](#) (p. 449)
- [Understanding Amazon EMR Log File Entries](#) (p. 449)
Amazon EMR Information in CloudTrail

If CloudTrail logging is turned on, calls made to all Amazon EMR actions are captured in log files. All of the Amazon EMR actions are documented in the Amazon EMR API Reference. For example, calls to the ListClusters, DescribeCluster, and RunJobFlow actions generate entries in CloudTrail log files.

Every log entry contains information about who generated the request. For example, if a request is made to create and run a new job flow (RunJobFlow), CloudTrail logs the user identity of the person or service that made the request. The user identity information helps you determine whether the request was made with root or IAM user credentials, with temporary security credentials for a role or federated user, or by another AWS service. For more information about CloudTrail fields, see CloudTrail Event Reference in the AWS CloudTrail User Guide.

You can store your log files in your bucket for as long as you want, but you can also define Amazon S3 lifecycle rules to archive or delete log files automatically. By default, your log files are encrypted by using Amazon S3 server-side encryption (SSE).

Understanding Amazon EMR Log File Entries

CloudTrail log files can contain one or more log entries composed of multiple JSON-formatted events. A log entry represents a single request from any source and includes information about the requested action, any input parameters, the date and time of the action, and so on. The log entries do not appear in any particular order. That is, they do not represent an ordered stack trace of the public API calls.

The following log file record shows that an IAM user called the RunJobFlow action by using the SDK.

```json
{
    "Records": [
    
    
    "eventVersion": "1.01",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:user/temporary-user-xx-7M",
        "accountId": "123456789012",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "temporary-user-xx-7M"
    },
    "eventTime": "2014-03-31T17:59:21Z",
    "eventSource": "elasticmapreduce.amazonaws.com",
    "eventName": "RunJobFlow",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "127.0.0.1",
    "userAgent": "aws-sdk-java/unknown-version Linux/xx Java_HotSpot(TM)_64-Bit_Server_VM/xx",
    "requestParameters": {
        "tags": [
            { "value": "prod",
              "key": "domain"
            },
            { "value": "us-east-1",
              "key": "realm"
            },
            { "value": "VERIFICATION",
              "key": "executionType"
            }
        ]
    }
    ]
}
```
Monitor Performance with Ganglia

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

The Ganglia open source project 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. For more information about the Ganglia open-source project, go to http://ganglia.info/.

**Topics**
- Add Ganglia to a Cluster (p. 450)
- View Ganglia Metrics (p. 451)
- Ganglia Reports (p. 451)
- Hadoop and Spark Metrics in Ganglia (p. 456)

### Add Ganglia to a Cluster

**To add Ganglia to a cluster using the console**

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. Choose **Create cluster**.
3. Under the **Additional Applications** list, choose **Ganglia** and **Configure and add**.
4. Proceed with creating the cluster with configurations as appropriate.

**To add Ganglia to a cluster using the AWS CLI**

In the AWS CLI, you can add Ganglia to a cluster by using `create-cluster` subcommand with the `--applications` parameter. This installs Ganglia using a bootstrap action, making the `--bootstrap-
action parameter unnecessary. If you specify only Ganglia using the --applications parameter, Ganglia is the only application installed.

- Type the following command to add Ganglia when you create a cluster and replace myKey with the name of your EC2 key pair.

- Linux, UNIX, and Mac OS X users:

```bash
aws emr create-cluster --name "Test cluster" --ami-version 3.11.0 --applications Name=Hue Name=Ganglia Name=Hive Name=Pig \ --use-default-roles --ec2-attributes KeyName=myKey \ --instance-type m3.xlarge --instance-count 3
```

- Windows users:

```bash
aws emr create-cluster --name "Test cluster" --ami-version 3.11.0 --applications Name=Hue Name=Ganglia Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey --instance-type m3.xlarge --instance-count 3
```

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

**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.


### View Ganglia Metrics

Ganglia provides a web-based user interface that you can use to view the metrics Ganglia collects. When you run Ganglia on Amazon EMR, the web interface runs on the master node and can be viewed using port forwarding, also known as creating an SSH tunnel. For more information about viewing web interfaces on Amazon EMR, see .

**To view the Ganglia web interface**

1. Use SSH to tunnel into the master node and create a secure connection. For information about how to create an SSH tunnel to the master node, see Option 2, Part 1: Set Up an SSH Tunnel to the Master Node Using Dynamic Port Forwarding (p. 466).
2. Install a web browser with a proxy tool, such as the FoxyProxy plug-in for Firefox, to create a SOCKS proxy for domains of the type "ec2.amazonaws.com". For more information, see Option 2, Part 2: Configure Proxy Settings to View Websites Hosted on the Master Node (p. 468).
3. With the proxy set and the SSH connection open, you can view the Ganglia UI by opening a browser window with http://master-public-dns-name/ganglia/, where master-public-dns-name is the public DNS address of the master server in the EMR cluster.

### Ganglia Reports

When you open the Ganglia web reports in a browser, you see an overview of the cluster's performance, with graphs detailing the load, memory usage, CPU utilization, and network traffic of the cluster. Below the cluster statistics are graphs for each individual server in the cluster. In the preceding cluster creation
example, we launched three instances, so in the following reports there are three instance charts showing the cluster data.
The default graph for the node instances is Load, but you can use the **Metric** list to change the statistic displayed in the node-instance graphs.
You can drill down into the full set of statistics for a given instance by selecting the node from the list or by choosing the corresponding node-instance chart.
This opens the Host Overview for the node.
If you scroll down, you can view charts of the full range of statistics collected on the instance.

Hadoop and Spark Metrics in Ganglia

Ganglia reports Hadoop metrics for each instance. The various types of metrics are prefixed by category: distributed file system (dfs.*), Java virtual machine (jvm.*), MapReduce (mapred.*), and remote procedure calls (rpc.*).

Ganglia metrics for Spark generally have prefixes for YARN application ID and Spark DAGScheduler. So prefixes follow this form:

- DAGScheduler.*
- application_xxxxxxxxx_xxxx.driver.*
- application_xxxxxxxxx_xxxx.executor.*

Connect to the Cluster

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

When you run an Amazon EMR cluster, often all you need to do is run an application to analyze your data and then collect the output from an Amazon S3 bucket. At other times, you may want to interact with the master node while the cluster is running. For example, you may want to connect to the master node to run interactive queries, check log files, debug a problem with the cluster, and so on. The following sections describe techniques that you can use to connect to the master node.

In an EMR cluster, the master node is an Amazon EC2 instance that coordinates the EC2 instances that are running as task and core nodes. The master node exposes a public DNS name that you can use to connect to it. By default, Amazon EMR creates security group rules for master and slave nodes that
determine how you access the nodes. For example, the master node security group contains a rule that allows you to connect to the master node using an SSH client over TCP port 22.

**Note**
You can connect to the master node only while the cluster is running. When the cluster terminates, the EC2 instance acting as the master node is terminated and is no longer available. To connect to the master node, you must also authenticate to the cluster. You can either use Kerberos for authentication, or specify an Amazon EC2 key pair private key when you launch the cluster. For more information about configuring Kerberos, and then connecting, see Use Kerberos Authentication (p. 194). When you launch a cluster from the console, the Amazon EC2 key pair private key is specified in the **Security and Access** section on the **Create Cluster** page.

By default, the ElasticMapReduce-master security group does not permit inbound SSH access. You may need to add an inbound rule that allows SSH access (TCP port 22) from the sources you want to have access. For more information about modifying security group rules, see Adding Rules to a Security Group in the Amazon EC2 User Guide for Linux Instances.

**Important**
Do not modify the remaining rules in the ElasticMapReduce-master security group. Modifying these rules may interfere with the operation of the cluster.

**Topics**
- Connect to the Master Node Using SSH (p. 457)
- View Web Interfaces Hosted on Amazon EMR Clusters (p. 461)

**Connect to the Master Node Using SSH**

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Secure Shell (SSH) is a network protocol you can use to create a secure connection to a remote computer. After you make a connection, the terminal on your local computer behaves as if it is running on the remote computer. Commands you issue locally run on the remote computer, and the command output from the remote computer appears in your terminal window.

When you use SSH with AWS, you are connecting to an EC2 instance, which is a virtual server running in the cloud. When working with Amazon EMR, the most common use of SSH is to connect to the EC2 instance that is acting as the master node of the cluster.

Using SSH to connect to the master node gives you the ability to monitor and interact with the cluster. You can issue Linux commands on the master node, run applications such as Hive and Pig interactively, browse directories, read log files, and so on. You can also create a tunnel in your SSH connection to view the web interfaces hosted on the master node. For more information, see View Web Interfaces Hosted on Amazon EMR Clusters (p. 461).

To connect to the master node using SSH, you need the public DNS name of the master node.

**Retrieve the Public DNS Name of the Master Node**

You can retrieve the master public DNS name using the Amazon EMR console and the AWS CLI.

**To retrieve the public DNS name of the master node using the Amazon EMR console**

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. On the **Cluster List** page, select the link for your cluster.

3. Note the **Master public DNS** value that appears at the top of the **Cluster Details** page.

   ![Cluster Details](image)

   **Note**
   
   You may also choose the **SSH** link beside the master public DNS name for instructions on creating an **SSH** connection with the master node.

   ![Cluster Details](image)

   **To retrieve the public DNS name of the master node using the AWS CLI**

   1. To retrieve the cluster identifier, type the following command.

      ```
      aws emr list-clusters
      ```

      The output lists your clusters including the cluster IDs. Note the cluster ID for the cluster to which you are connecting.

      ```json
      "Status": {
        "Timeline": {
          "ReadyDateTime": 1408040782.374,
          "CreationDateTime": 1408040501.213
        },
        "State": "WAITING",
        "StateChangeReason": {
          "Message": "Waiting after step completed"
        }
      },
      "NormalizedInstanceHours": 4,
      "Id": "j-2AL4XXXXXX5T9",
      "Name": "My cluster"
      ```

   2. To list the cluster instances including the master public DNS name for the cluster, type one of the following commands. Replace **j-2AL4XXXXXX5T9** with the cluster ID returned by the previous command.

   ```
   aws emr list-instances --cluster-id j-2AL4XXXXXX5T9
   ```
aws emr list-instances --cluster-id j-2AL4XXXXXX5T9

Or:

aws emr describe-cluster --cluster-id j-2AL4XXXXXX5T9

The output lists the cluster instances including DNS names and IP addresses. Note the value for PublicDnsName.

"Status": {
  "Timeline": {
    "ReadyDateTime": 1408040779.263,
    "CreationDateTime": 1408040515.535
  },
  "State": "RUNNING",
  "StateChangeReason": {}
},
"Ec2InstanceId": "i-e89b45e7",
"PublicDnsName": "ec2-###-##-##-###.us-west-2.compute.amazonaws.com",
"PrivateDnsName": "ip-###-##-##-###.us-west-2.compute.internal",
"PublicIpAddress": "##.###.###.##",
"Id": "ci-12XXXXXXXXFMH",
"PrivateIpAddress": "###.##.#.###"

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

Connect to the Master Node Using SSH and an Amazon EC2 Private Key on Linux, Unix, and Mac OS X

To create an SSH connection authenticated with a private key file, you need to specify the Amazon EC2 key pair private key when you launch a cluster. If you launch a cluster from the console, the Amazon EC2 key pair private key is specified in the Security and Access section on the Create Cluster page. For more information about accessing your key pair, see Amazon EC2 Key Pairs in the Amazon EC2 User Guide for Linux Instances.

Your Linux computer most likely includes an SSH client by default. For example, OpenSSH is installed on most Linux, Unix, and Mac OS X operating systems. You can check for an SSH client by typing ssh at the command line. If your computer does not recognize the command, install an SSH client to connect to the master node. The OpenSSH project provides a free implementation of the full suite of SSH tools. For more information, see the OpenSSH website.

The following instructions demonstrate opening an SSH connection to the Amazon EMR master node on Linux, Unix, and Mac OS X.

To configure the key pair private key file permissions

Before you can use your Amazon EC2 key pair private key to create an SSH connection, you must set permissions on the .pem file so that only the key owner has permission to access the file. This is required for creating an SSH connection using terminal or the AWS CLI.

1. Locate your .pem file. These instructions assume that the file is named mykeypair.pem and that it is stored in the current user's home directory.
2. Type the following command to set the permissions. Replace ~/mykeypair.pem with the location and file name of your key pair private key file.
If you do not set permissions on the .pem file, you will receive an error indicating that your key file is unprotected and the key will be rejected. To connect, you only need to set permissions on the key pair private key file the first time you use it.

**To connect to the master node using the terminal**

1. Open a terminal window. On Mac OS X, choose **Applications > Utilities > Terminal**. On other Linux distributions, terminal is typically found at **Applications > Accessories > Terminal**.
2. To establish a connection to the master node, type the following command. Replace `ec2-###-##-##-###.compute-1.amazonaws.com` with the master public DNS name of your cluster and replace `~/mykeypair.pem` with the location and file name of your .pem file.

   ```bash
   ssh hadoop@ec2-###-##-##-###.compute-1.amazonaws.com -i ~/mykeypair.pem
   ```

   **Important**
   You must use the login name `hadoop` when you connect to the Amazon EMR master node; otherwise, you may see an error similar to `Server refused our key`.

3. A warning states that the authenticity of the host you are connecting to cannot be verified. Type `yes` to continue.

4. When you are done working on the master node, type the following command to close the SSH connection.

   ```bash
   exit
   ```

**Connect to the Master Node Using the AWS CLI**

You can create an SSH connection with the master node using the AWS CLI on Windows and on Linux, Unix, and Mac OS X. Regardless of the platform, you need the public DNS name of the master node and your Amazon EC2 key pair private key. If you are using the AWS CLI on Linux, Unix, or Mac OS X, you must also set permissions on the private key (.pem or .ppk) file as shown in To configure the key pair private key file permissions (p. 459).

**To connect to the master node using the AWS CLI**

1. To retrieve the cluster identifier, type:

   ```bash
   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.

   ```json
   "Status": {
   "Timeline": {
   "ReadyDateTime": 1408040782.374,
   "CreationDateTime": 1408040501.213
   },
   "State": "WAITING",
   "StateChangeReason": {
   "Message": "Waiting after step completed"
   }
   ```
2. Type the following command to open an SSH connection to the master node. In the following example, replace j-2AL4XXXXXX5T9 with the cluster ID and replace ~/mykeypair.key with the location and file name of your .pem file (for Linux, Unix, and Mac OS X) or .ppk file (for Windows).

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

3. When you are done working on the master node, close the AWS CLI window.

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

**Connect to the Master Node Using SSH on Windows**

Windows users can use an SSH client such as PuTTY to connect to the master node. Before connecting to the Amazon EMR master node, you should download and install PuTTY and PuTTYgen. You can download these tools from the PuTTY download page.

PuTTY does not natively support the key pair private key file format (.pem) generated by Amazon EC2. You use PuTTYgen to convert your key file to the required PuTTY format (.ppk). You must convert your key into this format (.ppk) before attempting to connect to the master node using PuTTY.

For more information about converting your key, see Converting Your Private Key Using PuTTYgen in the Amazon EC2 User Guide for Linux Instances.

**To connect to the master node using PuTTY**

1. Open putty.exe. You can also launch PuTTY from the Windows programs list.
2. If necessary, in the Category list, choose Session.
3. For Host Name (or IP address), type hadoop@MasterPublicDNS. For example: hadoop@ec2-###-###-##-##-##.compute-1.amazonaws.com.
4. In the Category list, choose Connection > SSH, Auth.
5. For Private key file for authentication, choose Browse and select the .ppk file that you generated.
6. Choose Open and then Yes to dismiss the PuTTY security alert.

**Important**

When logging into the master node, type hadoop if you are prompted for a user name.

7. When you are done working on the master node, you can close the SSH connection by closing PuTTY.

**Note**

To prevent the SSH connection from timing out, you can choose Connection in the Category list and select the option Enable TCP_keepalives. If you have an active SSH session in PuTTY, you can change your settings by opening the context (right-click) for the PuTTY title bar and choosing Change Settings.

**View Web Interfaces Hosted on Amazon EMR Clusters**

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.
Hadoop and other applications you install on your Amazon EMR cluster, publish user interfaces as web sites hosted on the master node. For security reasons, when using EMR-Managed Security Groups, these web sites are only available on the master node's local web server, so you need to connect to the master node to view them. For more information, see Connect to the Master Node Using SSH (p. 457). Hadoop also publishes user interfaces as web sites hosted on the core and task (slave) nodes. These web sites are also only available on local web servers on the nodes.

**Warning**

It is possible to configure a custom security group to allow inbound access to these web interfaces. Keep in mind that any port on which you allow inbound traffic represents a potential security vulnerability. Carefully review custom security groups to ensure that you minimize vulnerabilities. For more information, see Control Network Traffic with Security Groups (p. 216).

The following table lists web interfaces you can view on the master node. The Hadoop interfaces are available on all clusters. Other web interfaces such as Ganglia and HBase are only available if you install additional applications on your cluster. To access the following interfaces, replace `master-public-dns-name` in the URI with the DNS name of the master node after creating an SSH tunnel. For more information about retrieving the master public DNS name, see Retrieve the Public DNS Name of the Master Node (p. 457). For more information about creating an SSH tunnel, see Option 2, Part 1: Set Up an SSH Tunnel to the Master Node Using Dynamic Port Forwarding (p. 466).

<table>
<thead>
<tr>
<th>Name of interface</th>
<th>URI</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hadoop version 2.x</strong></td>
<td></td>
</tr>
<tr>
<td>Hadoop ResourceManager</td>
<td><code>http://master-public-dns-name:9026/</code></td>
</tr>
<tr>
<td>Hadoop HDFS NameNode</td>
<td><code>http://master-public-dns-name:9101/</code></td>
</tr>
<tr>
<td>Ganglia Metrics Reports</td>
<td><code>http://master-public-dns-name/ganglia/</code></td>
</tr>
<tr>
<td>HBase Interface</td>
<td><code>http://master-public-dns-name:60010/master-status</code></td>
</tr>
<tr>
<td>Hue Web Application</td>
<td><code>http://master-public-dns-name:8888/</code></td>
</tr>
<tr>
<td>Impala Statestore</td>
<td><code>http://master-public-dns-name:25000</code></td>
</tr>
<tr>
<td>Impalad</td>
<td><code>http://master-public-dns-name:25010</code></td>
</tr>
<tr>
<td>Impala Catalog</td>
<td><code>http://master-public-dns-name:25020</code></td>
</tr>
<tr>
<td><strong>Hadoop version 1.x</strong></td>
<td></td>
</tr>
<tr>
<td>Hadoop MapReduce JobTracker</td>
<td><code>http://master-public-dns-name:9100/</code></td>
</tr>
<tr>
<td>Hadoop HDFS NameNode</td>
<td><code>http://master-public-dns-name:9101/</code></td>
</tr>
<tr>
<td>Ganglia Metrics Reports</td>
<td><code>http://master-public-dns-name/ganglia/</code></td>
</tr>
<tr>
<td>HBase Interface</td>
<td><code>http://master-public-dns-name:60010/master-status</code></td>
</tr>
<tr>
<td>HBase Thrift</td>
<td><code>http://master-public-dns-name:9000</code></td>
</tr>
</tbody>
</table>

**Note**

You can change the configuration of the Hadoop version 2.x web interfaces by editing the `conf/hdfs-site.xml` file. You can change the configuration of the Hadoop version 1.x web interfaces by editing the `conf/hadoop-default.xml` file.
Because there are several application-specific interfaces available on the master node that are not available on the core and task nodes, the instructions in this document are specific to the Amazon EMR master node. Accessing the web interfaces on the core and task nodes can be done in the same manner as you would access the web interfaces on the master node.

There are several ways you can access the web interfaces on the master node. The easiest and quickest method is to use SSH to connect to the master node and use the text-based browser, Lynx, to view the web sites in your SSH client. However, Lynx is a text-based browser with a limited user interface that cannot display graphics. The following example shows how to open the Hadoop ResourceManager interface using Lynx (Lynx URLs are also provided when you log into the master node using SSH).

```
lynx http://ip-###-##-##-###.us-west-2.compute.internal:9026/
```

There are two remaining options for accessing web interfaces on the master node that provide full browser functionality. Choose one of the following:

- **Option 1** (recommended for more technical users): Use an SSH client to connect to the master node, configure SSH tunneling with local port forwarding, and use an Internet browser to open web interfaces hosted on the master node. This method allows you to configure web interface access without using a SOCKS proxy.

- **Option 2** (recommended for new users): Use an SSH client to connect to the master node, configure SSH tunneling with dynamic port forwarding, and configure your Internet browser to use an add-on such as FoxyProxy or SwitchySharp to manage your SOCKS proxy settings. This method allows you to automatically filter URLs based on text patterns and to limit the proxy settings to domains that match the form of the master node's DNS name. The browser add-on automatically handles turning the proxy on and off when you switch between viewing websites hosted on the master node, and those on the Internet. For more information about how to configure FoxyProxy for Firefox and Google Chrome, see **Option 2, Part 2: Configure Proxy Settings to View Websites Hosted on the Master Node** (p. 468).

**Topics**
- **Option 1: Set Up an SSH Tunnel to the Master Node Using Local Port Forwarding** (p. 463)
- **Option 2, Part 1: Set Up an SSH Tunnel to the Master Node Using Dynamic Port Forwarding** (p. 466)
- **Option 2, Part 2: Configure Proxy Settings to View Websites Hosted on the Master Node** (p. 468)
- **Access the Web Interfaces on the Master Node Using the Console** (p. 470)

**Option 1: Set Up an SSH Tunnel to the Master Node Using Local Port Forwarding**

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

To connect to the local web server on the master node, you create an SSH tunnel between your computer and the master node. This is also known as port forwarding. If you do not wish to use a SOCKS proxy, you can set up an SSH tunnel to the master node using local port forwarding. With local port forwarding, you specify unused local ports that are used to forward traffic to specific remote ports on the master node's local web server. For example, you could configure an unused local port (such as 8157) to forward traffic to the Ganglia web interface on the master node (hostname:80). Beginning with AMI 3.1.1, the Hadoop NameNode and ResourceManager web interfaces on the master node are no longer bound to localhost. To set up an SSH tunnel using local port forwarding for these interfaces, you use the master public DNS name instead of localhost.
Setting up an SSH tunnel using local port forwarding requires the public DNS name of the master node and your key pair private key file. For information about how to locate the master public DNS name, see To retrieve the public DNS name of the master node using the Amazon EMR console (p. 457). For more information about accessing your key pair, see Amazon EC2 Key Pairs in the Amazon EC2 User Guide for Linux Instances. For more information about the sites you might want to view on the master node, see View Web Interfaces Hosted on Amazon EMR Clusters (p. 461).

Set Up an SSH Tunnel to the Master Node Using Local Port Forwarding on Linux, Unix, and Mac OS X

To set up an SSH tunnel using local port forwarding in terminal

1. Open a terminal window. On Mac OS X, choose Applications > Utilities > Terminal. On other Linux distributions, terminal is typically found at Applications > Accessories > Terminal.

2. Type the following command to open an SSH tunnel on your local machine. This command accesses the ResourceManager web interface by forwarding traffic on local port 8157 (a randomly chosen, unused local port) to port 8088 on the master node's local web server. In the command, replace ~/mykeypair.pem with the location and file name of your .pem file and replace ec2-###-###-###.compute-1.amazonaws.com with the master public DNS name of your cluster.

   ssh -i ~/mykeypair.pem -N -L 8157:ec2-###-###-###.compute-1.amazonaws.com:9026 hadoop@ec2-###-###-###.compute-1.amazonaws.com

   After you issue this command, the terminal remains open and does not return a response.

   **Note**
   
   -L signifies the use of local port forwarding which allows you to specify a local port used to forward data to the identified remote port on the master node's local web server.


4. To access the NameNode web interface, launch a new terminal session, replace port 8088 in the previous command with port 9101 and replace port 8157 with port 8159 (a third unused local port). For example:

   ssh -i ~/mykeypair.pem -N -L 8159:ec2-###-###-###.compute-1.amazonaws.com:9101 hadoop@ec2-###-###-###.compute-1.amazonaws.com

   Then, type the following address in your browser: http://localhost:8159/.

5. To access the Hue web interface, launch a new terminal session, replace port 8088 in the previous command with port 8888 and replace port 8157 with port 8160 (a fourth unused local port). For example:

   ssh -i ~/mykeypair.pem -N -L 8160:ec2-###-###-###.compute-1.amazonaws.com:8888 hadoop@ec2-###-###-###.compute-1.amazonaws.com

   Then, type the following address in your browser: http://localhost:8160/.

6. When you are done working with the web interfaces on the master node, close the terminal windows.
Set Up an SSH Tunnel to the Master Node Using Local Port Forwarding on Windows

To set up an SSH tunnel using local port forwarding in PuTTY

1. Double-click putty.exe to start PuTTY. You can also launch PuTTY from the Windows programs list.
2. If necessary, in the Category list, choose Session.
3. In the Host Name (or IP address) field, type hadoop@MasterPublicDNS. For example: hadoop@ec2-###-##-##-###.compute-1.amazonaws.com.
4. In the Category list, expand Connection > SSH, and then choose Auth.
5. For Private key file for authentication, choose Browse and select the .ppk file that you generated.
   Note
   PuTTY does not natively support the key pair private key file format (.pem) generated by Amazon EC2. You use PuTTYgen to convert your key file to the required PuTTY format (.ppk). You must convert your key into this format (.ppk) before attempting to connect to the master node using PuTTY.
6. In the Category list, expand Connection > SSH, and then choose Tunnels.
7. In the Source port field, type an unused local port number, for example 8157.
8. To access a web interface, in the Destination field, type host name:port number. For example, to access the Ganglia interface, type localhost:80.
9. Leave the Local and Auto options selected.
10. Choose Add. You should see an entry in the Forwarded ports box similar to: L8157 localhost:80.
11. Choose Open and Yes to dismiss the PuTTY security alert.
   Important
   When logging in to the master node, if you are prompted for a user name, type hadoop.
12. To access the Ganglia interface on the master node, type http://localhost:8157/ganglia in your browser's address bar.
13. To access other interfaces on the master node, you must add additional tunnels for each port. Right-click the PuTTY title bar and choose Change Settings.
14. Follow the previous steps to add additional tunnels for the remaining web interfaces using the following table as a guide.

<table>
<thead>
<tr>
<th>Interface</th>
<th>Source port</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hadoop ResourceManager</td>
<td>8158</td>
<td>master-public-dns-name:9026</td>
</tr>
<tr>
<td>Hadoop NameNode</td>
<td>8159</td>
<td>master-public-dns-name:9101</td>
</tr>
<tr>
<td>Hue web application</td>
<td>8160</td>
<td>master-public-dns-name:8888</td>
</tr>
</tbody>
</table>

Note
For AMI 3.1.0 and earlier, you may use localhost for the destination instead of the master public DNS name.

For a complete list of web interfaces on the master node, see View Web Interfaces Hosted on Amazon EMR Clusters (p. 461).
15. After adding a new tunnel, choose Apply.
16. To open the interfaces, type localhost:port number in your browser's address bar. For example, to open the ResourceManager interface, type http://localhost:8158/.
Note
Setting up a SOCKS proxy and dynamic port forwarding eliminates the need to create multiple tunnels. Also note that you can save your PuTTY session settings for later reuse.

17. When you are done working with the web interfaces on the master node, close the PuTTY window.

Option 2, Part 1: Set Up an SSH Tunnel to the Master Node Using Dynamic Port Forwarding

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

To connect to the local web server on the master node, you create an SSH tunnel between your computer and the master node. This is also known as port forwarding. If you create your SSH tunnel using dynamic port forwarding, all traffic routed to a specified unused local port is forwarded to the local web server on the master node. This creates a SOCKS proxy. You can then configure your Internet browser to use an add-on such as FoxyProxy or SwitchySharp to manage your SOCKS proxy settings. Using a proxy management add-on allows you to automatically filter URLs based on text patterns and to limit the proxy settings to domains that match the form of the master node's public DNS name. The browser add-on automatically handles turning the proxy on and off when you switch between viewing websites hosted on the master node, and those on the Internet.

Before you begin, you need the public DNS name of the master node and your key pair private key file. For information about how to locate the master public DNS name, see To retrieve the public DNS name of the master node using the Amazon EMR console (p. 457). For more information about accessing your key pair, see Amazon EC2 Key Pairs in the Amazon EC2 User Guide for Linux Instances. For more information about the sites you might want to view on the master node, see View Web Interfaces Hosted on Amazon EMR Clusters (p. 461).

Set Up an SSH Tunnel to the Master Node Using Dynamic Port Forwarding on Linux, Unix, and Mac OS X

To set up an SSH tunnel using dynamic port forwarding on Linux, Unix, and Mac OS X

1. Open a terminal window. On Mac OS X, choose Applications > Utilities > Terminal. On other Linux distributions, terminal is typically found at Applications > Accessories > Terminal.
2. Type the following command to open an SSH tunnel on your local machine. Replace ~/mykeypair.pem with the location and file name of your .pem file, replace 8157 with an unused, local port number, and replace c2-###-##-##-###.compute-1.amazonaws.com with the master public DNS name of your cluster.

   ssh -i ~/mykeypair.pem -N -D 8157 hadoop@c2-###-##-##-###.compute-1.amazonaws.com

   After you issue this command, the terminal remains open and does not return a response.

   Note
   -D signifies the use of dynamic port forwarding which allows you to specify a local port used to forward data to all remote ports on the master node's local web server. Dynamic port forwarding creates a local SOCKS proxy listening on the port specified in the command.
3. After the tunnel is active, configure a SOCKS proxy for your browser. For more information, see Option 2, Part 2: Configure Proxy Settings to View Websites Hosted on the Master Node (p. 468).
4. When you are done working with the web interfaces on the master node, close the terminal window.
Set Up an SSH tunnel Using Dynamic Port Forwarding with the AWS CLI

You can create an SSH connection with the master node using the AWS CLI on Windows and on Linux, Unix, and Mac OS X. If you are using the AWS CLI on Linux, Unix, or Mac OS X, you must set permissions on the .pem file as shown in To configure the key pair private key file permissions (p. 459). If you are using the AWS CLI on Windows, PuTTY must appear in the path environment variable or you may receive an error such as OpenSSH or PuTTY not available.

To set up an SSH tunnel using dynamic port forwarding with the AWS CLI

1. Create an SSH connection with the master node as shown in Connect to the Master Node Using the AWS CLI (p. 460).
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-2AL4XXX5T9",
   "Name": "AWS CLI cluster"
   ```

3. Type the following command to open an SSH tunnel to the master node using dynamic port forwarding. In the following example, replace `j-2AL4XXX5T9` 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-2AL4XXX5T9 --key-pair-file ~/mykeypair.key
   ```

   **Note**
   The socks command automatically configures dynamic port forwarding on local port 8157. Currently, this setting cannot be modified.

4. After the tunnel is active, configure a SOCKS proxy for your browser. For more information, see Option 2, Part 2: Configure Proxy Settings to View Websites Hosted on the Master Node (p. 468).
5. When you are done working with the web interfaces on the master node, close the AWS CLI window.

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

Set Up an SSH Tunnel to the Master Node Using Dynamic Port Forwarding on Windows

Windows users can use an SSH client such as PuTTY to create an SSH tunnel to the master node. Before connecting to the Amazon EMR master node, you should download and install PuTTY and PuTTYgen. You can download these tools from the [PuTTY download page](http://www.putty.org/).
PuTTY does not natively support the key pair private key file format (.pem) generated by Amazon EC2. You use PuTTYgen to convert your key file to the required PuTTY format (.ppk). You must convert your key into this format (.ppk) before attempting to connect to the master node using PuTTY.

For more information about converting your key, see Converting Your Private Key Using PuTTYgen in the Amazon EC2 User Guide for Linux Instances.

**To set up an SSH tunnel using dynamic port forwarding on Windows**

1. Double-click `putty.exe` to start PuTTY. You can also launch PuTTY from the Windows programs list.

   **Note**
   If you already have an active SSH session with the master node, you can add a tunnel by right-clicking the PuTTY title bar and choosing Change Settings.

2. If necessary, in the **Category** list, choose **Session**.

3. In the **Host Name** field, type `hadoop@MasterPublicDNS`. For example: `hadoop@ec2-###-##-##-###.compute-1.amazonaws.com`.

4. In the **Category** list, expand **Connection > SSH**, and then choose **Auth**.

5. For **Private key file for authentication**, choose **Browse** and select the .ppk file that you generated.

   **Note**
   PuTTY does not natively support the key pair private key file format (.pem) generated by Amazon EC2. You use PuTTYgen to convert your key file to the required PuTTY format (.ppk). You must convert your key into this format (.ppk) before attempting to connect to the master node using PuTTY.

6. In the **Category** list, expand **Connection > SSH**, and then choose **Tunnels**.

7. In the **Source port** field, type 8157 (an unused local port).

8. Leave the **Destination** field blank.

9. Select the **Dynamic** and **Auto** options.

10. Choose **Add** and **Open**.

11. Choose **Yes** to dismiss the PuTTY security alert.

   **Important**
   When you log in to the master node, type `hadoop` if you are prompted for a user name.

12. After the tunnel is active, configure a SOCKS proxy for your browser. For more information, see Option 2, Part 2: Configure Proxy Settings to View Websites Hosted on the Master Node (p. 468).

13. When you are done working with the web interfaces on the master node, close the PuTTY window.

**Option 2, Part 2: Configure Proxy Settings to View Websites Hosted on the Master Node**

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

If you use an SSH tunnel with dynamic port forwarding, you must use a SOCKS proxy management add-on to control the proxy settings in your browser. Using a SOCKS proxy management tool allows you to automatically filter URLs based on text patterns and to limit the proxy settings to domains that match the form of the master node’s public DNS name. The browser add-on automatically handles turning the proxy on and off when you switch between viewing websites hosted on the master node and those on the Internet. To manage your proxy settings, configure your browser to use an add-on such as FoxyProxy or SwitchySharp.
For more information about creating an SSH tunnel, see Option 2, Part 1: Set Up an SSH Tunnel to the Master Node Using Dynamic Port Forwarding (p. 466). For more information about the available web interfaces, see View Web Interfaces Hosted on Amazon EMR Clusters (p. 461).

The following example demonstrates a FoxyProxy configuration using Google Chrome. The relevant settings that are loaded from the configuration file in the example are as follows:

- **Host or IP Address**—This is set to `localhost` with the Port set to `8157` in the example. You should set this port to the local port number that you used to establish the SSH tunnel with the master node in Option 2, Part 1: Set Up an SSH Tunnel to the Master Node Using Dynamic Port Forwarding (p. 466). This port must also match the port number you use in PuTTY or other terminal emulator you use to connect.
- **SOCKS v5** configuration is specified.
- **Login credentials** are not specified.
- **URL Patterns**
  - The `*ec2*.amazonaws.com*` and `*10*.amazonaws.com*` patterns match the public DNS name of clusters in US regions.
  - The `*ec2*.compute*` and `*10*.compute*` patterns match the public DNS name of clusters in all other regions.
  - The `10.*` pattern provides access to the JobTracker log files in Hadoop. Alter this filter if it conflicts with your network access plan.

Configure FoxyProxy for Google Chrome

You can configure FoxyProxy for Google Chrome, Mozilla Firefox, and Microsoft Internet Explorer. FoxyProxy provides a set of proxy management tools that allow you to use a proxy server for URLs that match patterns corresponding to the domains used by the Amazon EC2 instances in your Amazon EMR cluster.

To install and configure FoxyProxy using Google Chrome

1. See https://chrome.google.com/webstore/search/foxy%20proxy and follow the links and instructions to add FoxyProxy to Chrome.
2. Using a text editor, create a file named `foxyproxy-settings.xml` with the following contents:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<foxyproxy>
  <proxies>
    <proxy name="emr-socks-proxy" id="2322596116" notes="" fromSubscription="false" enabled="true" mode="manual" selectedTabIndex="2" lastresort="false" animatedIcons="true" includeInCycle="true" color="#0055E5" proxyDNS="true" noInternalIPs="false" autoconfMode="pac" clearCacheBeforeUse="false" disableCache="false" clearCookiesBeforeUse="false" rejectCookies="false">
      <matches>
        <match enabled="true" name="*ec2*.amazonaws.com*" pattern="*ec2*.amazonaws.com*" isRegEx="false" isBlackList="false" isMultiLine="false" caseSensitive="false" fromSubscription="false" />
        <match enabled="true" name="*ec2*.compute*" pattern="*ec2*.compute*" isRegEx="false" isBlackList="false" isMultiLine="false" caseSensitive="false" fromSubscription="false" />
        <match enabled="true" name="10.*" pattern="http://10.*" isRegEx="false" isBlackList="false" isMultiLine="false" caseSensitive="false" fromSubscription="false" />
        <match enabled="true" name="*10*.amazonaws.com*" pattern="*10*.amazonaws.com*" isRegEx="false" isBlackList="false" isMultiLine="false" caseSensitive="false" fromSubscription="false" />
      </matches>
    </proxy>
  </proxies>
</foxyproxy>
```
3. Manage extensions in Chrome (go to chrome://extensions).
5. On the FoxyProxy page, choose Import/Export.
6. On the Import/Export page, choose Choose File, browse to the location of the foxyproxy-settings.xml file you created, select the file, and choose Open.
7. Choose Replace when prompted to overwrite the existing settings.
8. For Proxy mode, choose Use proxies based on their predefined patterns and priorities.
9. To open the web interfaces, in your browser's address bar, type master-public-dns followed by the port number or URL. Use the following table as a guide.

<table>
<thead>
<tr>
<th>Interface</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ganglia Metrics Reports</td>
<td>master-public-dns/ganglia/</td>
</tr>
<tr>
<td>Hadoop ResourceManager</td>
<td>master-public-dns-name:9026</td>
</tr>
<tr>
<td>Hadoop NameNode</td>
<td>master-public-dns-name:9101</td>
</tr>
<tr>
<td>Hue web application</td>
<td>master-public-dns-name:8888</td>
</tr>
</tbody>
</table>

For a complete list of web interfaces on the master node, see View Web Interfaces Hosted on Amazon EMR Clusters (p. 461).

**Access the Web Interfaces on the Master Node Using the Console**

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

If you already have an SSH tunnel configured with the Amazon EMR master node using dynamic port forwarding, you can open the web interfaces using the console.

**To open the web interfaces using the console**

1. Verify that you have established an SSH tunnel with the master node and that you have a proxy management add-on configured for your browser.
2. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
3. On the Cluster List page, choose the link for your cluster.
4. In the cluster details, for **Connections**, choose the link for the web interface you wish to open in your browser.

![Cluster Details](image)

5. Alternatively, choose the **View All** link to display links to all of the available web interfaces on your cluster's master node. Choosing the links opens the interfaces in your browser.

![Web Interfaces](image)

If you do not have an SSH tunnel open with the master node, choose **Enable Web Connection** for instructions on creating a tunnel, or see Option 2, Part 1: Set Up an SSH Tunnel to the Master Node Using Dynamic Port Forwarding (p. 466).

![Enable Web Connection](image)

**Note**
If you have an SSH tunnel configured using local port forwarding, the Amazon EMR console does not detect the connection.
Control Cluster Termination

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Control over cluster termination is determined by two options: termination protection and auto-termination. By default, when you launch a cluster using the console, termination protection is turned on. This prevents accidental termination of the cluster. When you launch a cluster using the CLI or API, termination protection is turned off.

Auto-termination determines whether the cluster should automatically terminate when all steps are complete. When you launch a cluster using the console, the default behavior is for the cluster to remain active after all steps are complete. In other words, the cluster is long-running. A long-running cluster must be manually terminated. When you launch a cluster using the CLI or API, the default behavior is for the cluster to terminate when data processing is complete; that is, when no more steps are left to run. This creates a transient cluster.

Topics

- Terminate a Cluster (p. 472)
- Managing Cluster Termination (p. 474)

Terminate a Cluster

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

This section describes the methods of terminating a cluster. You can terminate clusters in the STARTING, RUNNING, or WAITING states. A cluster in the WAITING state must be terminated or it runs indefinitely, generating charges to your account. You can terminate a cluster that fails to leave the STARTING state or is unable to complete a step.

If you are terminating a cluster that has termination protection set on it, you must first disable termination protection before you can terminate the cluster. After termination protection is disabled, you can terminate the cluster. Clusters can be terminated using the console, the AWS CLI, or programmatically using the TerminateJobFlows API.

Depending on the configuration of the cluster, it may take up to 5-20 minutes for the cluster to completely terminate and release allocated resources, such as EC2 instances.

Terminate a Cluster Using the Console

You can terminate one or more clusters using the Amazon EMR console. The steps to terminate a cluster in the console vary depending on whether termination protection is on or off. To terminate a protected cluster, you must first disable termination protection.

To terminate a cluster with termination protection off

1. Sign in to the AWS Management Console and open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. Select the cluster to terminate. You can select multiple clusters and terminate them at the same time.
3. Choose Terminate.
4. When prompted, choose **Terminate**.

Amazon EMR terminates the instances in the cluster and stops saving log data.

**To terminate a cluster with termination protection on**

1. Sign in to the AWS Management Console and open the Amazon EMR console at [https://console.aws.amazon.com/elasticmapreduce/](https://console.aws.amazon.com/elasticmapreduce/).
2. On the **Cluster List** page, select the cluster to terminate. You can select multiple clusters and terminate them at the same time.
3. Choose **Terminate**.
4. When prompted, choose **Change** to turn termination protection off. If you selected multiple clusters, choose **Turn off all** to disable termination protection for all the clusters at once.
5. In the **Terminate clusters** dialog, for **Termination Protection**, choose **Off** and then click the check mark to confirm.
6. Click **Terminate**.

Amazon EMR terminates the instances in the cluster and stops saving log data.

**Terminate a Cluster Using the AWS CLI**

**To terminate an unprotected cluster using the AWS CLI**

To terminate an unprotected cluster using the AWS CLI, use the `terminate-clusters` subcommand with the `--cluster-ids` parameter.

- Type the following command to terminate a single cluster and replace `j-3KVXXXXXXX7UG` with your cluster ID.

```
aws emr terminate-clusters --cluster-ids j-3KVXXXXXXX7UG
```

To terminate multiple clusters, type the following command and replace `j-3KVXXXXXXX7UG` and `j-WJ2XXXXXX8EU` with your cluster IDs.

```
aws emr terminate-clusters --cluster-ids j-3KVXXXXXXX7UG j-WJ2XXXXXX8EU
```

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

**To terminate a protected cluster using the AWS CLI**

To terminate a protected cluster using the AWS CLI, first disable termination protection using the `modify-cluster-attributes` subcommand with the `--no-termination-protected` parameter. Then use the `terminate-clusters` subcommand with the `--cluster-ids` parameter to terminate it.

1. Type the following command to disable termination protection and replace `j-3KVTXXXXXXX7UG` with your cluster ID.

```
aws emr modify-cluster-attributes --cluster-id j-3KVTXXXXXXX7UG --no-termination-protected
```

2. To terminate the cluster, type the following command and replace `j-3KVXXXXXXX7UG` with your cluster ID.
Terminate a Cluster Using the API

The TerminateJobFlows operation ends step processing, uploads any log data from Amazon EC2 to Amazon S3 (if configured), and terminates the Hadoop cluster. A cluster also terminates automatically if you set KeepJobAliveWhenNoSteps to False in a RunJobFlows request.

You can use this action to terminate either a single cluster or a list of clusters by their cluster IDs.

For more information about the input parameters unique to TerminateJobFlows, see TerminateJobFlows. For more information about the generic parameters in the request, see Common Request Parameters.

Managing Cluster Termination

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Termination protection helps ensure that the EC2 instances in your job flow are not shut down by an accident or error. This protection is especially useful if your cluster contains data in instance storage that you need to recover before those instances are terminated. When termination protection is not enabled, you can terminate clusters either through calls to the TerminateJobFlows API, through the Amazon EMR console, or by using the command line interface. In addition, the master node may terminate a node that has become unresponsive or has returned an error.

Warning
Termination protection helps protect cluster instances from accidental shutdown, but it does not guarantee that data is retained in the event of a human error—for example, if a reboot command is issued from the command line while connected to the instance, or if termination protection is disabled and a shutdown or reboot call is made through the API, the AWS CLI, or the AWS Management Console. When an instance shuts down, data saved to ephemeral storage on the instance, such as HDFS data, is lost and cannot be recovered. Back up critical data to Amazon S3 and as appropriate for your business continuity requirements.

By default, termination protection is enabled when you launch a cluster using the console. Termination protection is disabled by default when you launch a cluster using the CLI or API. When termination protection is enabled, you must explicitly remove termination protection from the cluster before you can terminate it. With termination protection enabled, TerminateJobFlows cannot terminate the cluster and users cannot terminate the cluster using the CLI. Users terminating the cluster using the Amazon EMR console are prompted with an extra step to turn termination protection off before terminating the cluster.

If you attempt to terminate a protected cluster with the API or CLI, the API returns an error, and the CLI exits with a non-zero return code.
When you submit steps to a cluster, the ActionOnFailure setting determines what the cluster does in response to any errors. The possible values for this setting are:

- TERMINATE_JOB_FLOW: If the step fails, terminate the cluster. If the cluster has termination protection enabled AND auto-terminate disabled, it will not terminate.
- CANCEL_AND_WAIT: If the step fails, cancel the remaining steps. If the cluster has auto-terminate disabled, the cluster will not terminate.
- CONTINUE: If the step fails, continue to the next step.

Termination Protection in Amazon EMR and Amazon EC2

An EMR cluster with termination protection enabled has the disableAPITermination attribute set for all EC2 instances in the cluster. If there is a conflict between the termination protection setting in Amazon EC2 and the setting in Amazon EMR for an instance, the Amazon EMR setting overrides the Amazon EC2 setting on the given instance. For example, if you use the Amazon EC2 console to enable termination protection on an EC2 instance in an Amazon EMR cluster that has termination protection disabled, Amazon EMR turns off termination protection on that EC2 instance and shuts down the instance when the rest of the cluster terminates.

Termination Protection and Spot Instances

Amazon EMR termination protection does not prevent an Amazon EC2 Spot Instance from terminating when the Spot Price rises above the maximum Spot price.

Termination Protection and Auto-Terminate

Enabling auto-terminate creates a transient cluster. The cluster automatically terminates when the last step is successfully completed even if termination protection is enabled.

Disabling auto-terminate causes instances in a cluster to persist after steps have successfully completed, but still allows the cluster to be terminated by user action, by errors, and by calls to TerminateJobFlows (if termination protection is disabled).

Note
By default, auto-terminate is disabled for clusters launched using the console and the CLI. Clusters launched using the API have auto-terminate enabled.

Configuring Termination Protection for New Clusters

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

To configure termination protection for a new cluster using the console

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. Choose Create cluster.
3. In the Cluster Configuration section, set the Termination protection field to Yes to enable protection, or set the field to No to disable it. By default, termination protection is enabled.
To configure termination protection for a new cluster using the AWS CLI

Using the AWS CLI, you can launch a cluster with termination protection enabled by typing the `create-cluster` command with the `--termination-protected` parameter. By default, termination protection is disabled when you launch a cluster using the AWS CLI. You can also use the `--no-termination-protected` parameter to disable termination protection.

- To launch a protected cluster, type the following command and replace `myKey` with the name of your EC2 key pair.
  - Linux, UNIX, and macOS users:
    ```bash
    aws emr create-cluster --name "Test cluster" --ami-version 3.8 --applications Name=Hue Name=Hive Name=Pig \ 
    --use-default-roles --ec2-attributes KeyName=myKey \ 
    --instance-type m3.xlarge --instance-count 3 --termination-protected
    ```
  - Windows users:
    ```bash
    aws emr create-cluster --name "Test cluster" --ami-version 3.8 --applications Name=Hue Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey --instance-type m3.xlarge --instance-count 3 --termination-protected
    ```


Configuring Termination Protection for Running Clusters

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

To configure termination protection for a running cluster using the console

1. Open the Amazon EMR console at [https://console.aws.amazon.com/elasticmapreduce/](https://console.aws.amazon.com/elasticmapreduce/).
2. On the Cluster List page, choose the link for your cluster.
3. On the Cluster Details page, in the Summary section, for Termination protection, choose Change.
4. To enable termination protection, choose On and select the check mark icon. Alternatively, choose Off to disable it.
To configure termination protection for a running cluster using the AWS CLI

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

- Type the following command to enable termination protection on a running cluster.

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

  To disable termination protection, type:

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

Scaling Cluster Resources

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

You can adjust the number of Amazon EC2 instances available to an EMR cluster automatically or manually in response to workloads that have varying demands. The following options are available:

- Using Amazon EMR versions 4.x and later, you can configure automatic scaling for the core instance group and task instance groups when you first create them or after the cluster is running. Amazon EMR automatically configures Auto Scaling parameters according to rules you specify, and then adds and removes instances based on a CloudWatch metric.
- You can manually resize the core instance group and task instance groups by manually adding or removing Amazon EC2 instances.
- You can add a new task instance group to the cluster.

The option to specify the Amazon EC2 instance type is only available during initial configuration of an instance group, so you can change the Amazon EC2 instance type only by adding a new task. When using Amazon EMR version 5.1.0 or later, a cluster-wide configuration allows you to specify whether Amazon EC2 instances removed from a cluster are terminated at the instance-hour boundary, or when tasks on the Amazon EC2 instance are complete.

Before you choose one of the methods for scaling described in this section, you should be familiar with some important concepts. First, you should understand the role of node types in an EMR cluster and how
Instance groups are used to manage them. For more information about the function of node types, see What is Amazon EMR?, and for more information about instance groups, see Instance Groups. You should also develop a strategy for right-sizing cluster resources based on the nature of your workload. For more information, see Cluster Configuration Guidelines.

**Note**
The master instance group in an EMR cluster always consists of a single node running on a single Amazon EC2 instance, so it can't scale after you initially configure it. You work with the core instance groups and task instance groups to scale out and scale in a cluster. It's possible to have a cluster with only a master node, and no core or task nodes. You must have at least one core node at cluster creation in order to scale the cluster. In other words, single node clusters cannot be resized.

**Topics**
- Using Automatic Scaling in Amazon EMR (p. 478)
- Manually Resizing a Running Cluster (p. 486)
- Cluster Scale-Down (p. 491)

## Using Automatic Scaling in Amazon EMR

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Automatic scaling in Amazon EMR release versions 4.0 and later allows you to programmatically scale out and scale in core nodes and task nodes based on a CloudWatch metric and other parameters that you specify in a scaling policy. The scaling policy is part of an instance group configuration. You can specify a policy during initial configuration of an instance group, or by modifying an instance group in an existing cluster, even when that instance group is active. Each instance group in a cluster, except the master instance group, can have its own scaling policy, which consists of scale-out and scale-in rules. Scale-out and scale-in rules can be configured independently, with different parameters for each rule.

You can configure scaling policies using the AWS Management Console, the AWS CLI, or the Amazon EMR API. When you use the AWS CLI or Amazon EMR API, you specify the scaling policy in JSON format. In addition, when using the AWS CLI or the Amazon EMR API, you can specify custom CloudWatch metrics. Custom metrics are not available for selection using the AWS Management Console. When you initially create a scaling policy using the console, a default policy suitable for many applications is pre-configured to help you get started. You can delete or modify the default rules.

Even though automatic scaling allows you to adjust EMR cluster capacity on-the-fly, you should still consider baseline workload requirements and plan your node and instance group configurations. For more information, see Cluster Configuration Guidelines.

**Note**
For most workloads, setting up both scale-in and scale-out rules is desirable to optimize resource utilization. Setting either rule without the other means that you need to manually resize the instance count after a scaling activity. In other words, this sets up a “one-way” automatic scale-out or scale-in policy with a manual reset.

## Creating the IAM Role for Automatic Scaling

Automatic scaling in Amazon EMR requires an IAM role with permissions to add and terminate instances when scaling activities are triggered. A default role, EMR_AutoScaling_DefaultRole, configured with the appropriate role policy and trust policy, is available for this purpose. When you create a cluster with a scaling policy using the AWS Management Console for the first time, Amazon EMR creates the default
role and attaches the default managed policy, AmazonElasticMapReduceForAutoScalingRole, for permissions.

When you create a cluster with an automatic scaling policy using the AWS CLI, you must first ensure that either the default IAM role exists, or that you have a custom IAM role with a policy attached that provides the appropriate permissions. To create the default role, you can run the create-default-roles command before you create a cluster. You can then specify --auto-scaling-role EMR_AutoScaling_DefaultRole option when you create a cluster. Alternatively, you can create a custom autoscaling role and then specify it when you create a cluster, for example --auto-scaling-role MyEMRAutoScalingRole. If you create a customized autoscaling role for Amazon EMR, we recommend that you base permissions policies for your custom role based on the managed policy. For more information, see Configure IAM Roles for Amazon EMR Permissions to AWS Services (p. 234).

Understanding Automatic Scaling Rules

When a scale-out rule triggers a scaling activity for an instance group, Amazon EC2 instances are added to the instance group according to your rules. New nodes can be used by applications such as Apache Spark and Apache Hive as soon as the Amazon EC2 instance enters the InService state. You can also set up a scale-in rule that terminates instances and removes nodes. For more information about the lifecycle of Amazon EC2 instances that scale automatically, see Auto Scaling Lifecycle in the Auto Scaling User Guide.

You can configure how a cluster terminates Amazon EC2 instances. You can choose to either terminate at the Amazon EC2 instance-hour boundary for billing, or upon task completion. This setting applies both to automatic scaling and to manual resizing operations. For more information about this configuration see Cluster Scale-Down (p. 491).

The following parameters for each rule in a policy determine automatic scaling behavior.

**Note**
The parameters listed here are based on the AWS Management Console for Amazon EMR. When you use the AWS CLI or Amazon EMR API, additional advanced configuration options are available. For more information about advanced options, see SimpleScalingPolicyConfiguration in the Amazon EMR API Reference.

- Maximum instances and minimum instances. The **Maximum instances** constraint specifies the maximum number of Amazon EC2 instances that can be in the instance group, and applies to all scale-out rules. Similarly, the **Minimum instances** constraint specifies the minimum number of Amazon EC2 instances and applies to all scale-in rules.
- The **Rule name**, which must be unique within the policy.
- The **scaling adjustment**, which determines the number of EC2 instances to add (for scale-out rules) or terminate (for scale-in rules) during the scaling activity triggered by the rule.
- The **CloudWatch metric**, which is watched for an alarm condition.
- A **comparison operator**, which is used to compare the CloudWatch metric to the **Threshold** value and determine a trigger condition.
- An **evaluation period**, in five-minute increments, for which the CloudWatch metric must be in a trigger condition before scaling activity is triggered.
- A **Cooldown period**, which determines the amount of time that must elapse between a scaling activity started by a rule and the start of the next scaling activity, regardless of the rule that triggers it. When an instance group has finished a scaling activity and reached its post-scale state, the cooldown period provides an opportunity for the CloudWatch metrics that might trigger subsequent scaling activities to stabilize. For more information, see Auto ScalingCooldownsin the Auto Scaling User Guide.
Using the AWS Management Console to Configure Automatic Scaling

When you create a cluster, you configure a scaling policy for instance groups using the advanced cluster configuration options. You can also create or modify a scaling policy for an instance group in-service by modifying instance groups in the Hardware settings of an existing cluster.

1. If you are creating a cluster, in the Amazon EMR console, select Create Cluster, select Go to advanced options, choose options for Step 1: Software and Steps, and then go to Step 2: Hardware Configuration.

—or—

If you are modifying an instance group in a running cluster, select your cluster from the cluster list, and then expand the Hardware section.

2. Click the pencil icon that appears in the Auto Scaling column for the instance group you want to configure. If an automatic scaling policy is already configured for the instance group, the number of Maximum instances and Minimum instances appear in this column; otherwise, Not enabled appears.

The Auto Scaling rules screen opens. Scale out and Scale in are selected by default, and default rules are pre-configured with settings suitable for many applications.

3. Type the Maximum instances you want the instance group to contain after it scales out, and type the Minimum instances you want the instance group to contain after it scales in.

4. Click the pencil to edit rule parameters, click the X to remove a rule from the policy, and click Add rule to add additional rules.

5. Choose rule parameters as described earlier in this topic. For descriptions of available CloudWatch metrics for Amazon EMR, see Amazon EMR Metrics and Dimensions in the Amazon CloudWatch User Guide.

Using the AWS CLI to Configure Automatic Scaling

You can use AWS CLI commands for Amazon EMR to configure automatic scaling when you create a cluster and when you create an instance group. You can use a shorthand syntax, specifying the JSON configuration inline within the relevant commands, or you can reference a file containing the
configuration JSON. You can also apply an automatic scaling policy to an existing instance group and remove an automatic scaling policy that was previously applied. In addition, you can retrieve details of a scaling policy configuration from a running cluster.

**Important**
When you create a cluster that has an automatic scaling policy, you must use the `--auto-scaling-role` `MyAutoScalingRole` command to specify the IAM role for autoscaling. The default role is `EMR_AutoScaling_DefaultRole` and can be created with the `create-default-roles` command. The role can only be added when the cluster is created, and cannot be added to an existing cluster.

For a detailed description of the parameters available when configuring an automatic scaling policy, see `PutAutoScalingPolicy` in Amazon EMR API Reference.

**Creating a Cluster with an Automatic Scaling Policy Applied to an Instance Group**

You can specify an automatic scaling configuration within the `--instance-groups` option of the `aws emr create-cluster` command. The following example illustrates a create-cluster command where an automatic scaling policy for the core instance group is provided inline. The command creates a scaling configuration equivalent to the default scale-out policy that appears when you create an auto scaling policy using the AWS Management Console for Amazon EMR. For brevity, a scale-in policy is not shown.

```bash
aws emr create-cluster --release-label emr-5.2.0 --service-role EMR_DefaultRole --ec2-attributes InstanceProfile=EMR_EC2_DefaultRole --auto-scaling-role EMR_AutoScaling_DefaultRole --instance-groups
    Name=MyMasterIG,InstanceGroupType=MASTER,InstanceType=m3.xlarge,InstanceCount=1
    'Name=MyCoreIG,InstanceGroupType=CORE,InstanceType=m3.xlarge,InstanceCount=2,AutoScalingPolicy={Constraints={MinCapacity=2,MaxCapacity=10},Rules=[{Name=Default-scale-out,Description=Replicates the default scale-out rule in the console.,Action={SimpleScalingPolicyConfiguration={AdjustmentType=CHANGE_IN_CAPACITY,ScalingAdjustment=1,CoolDown=300}},Trigger={CloudWatchAlarmDefinition={ComparisonOperator=LESS_THAN,EvaluationPeriods=1,MetricName=YARNMemoryAvailablePercentage,Namespace=AWS/ElasticMapReduce,Period=300,Statistic=AVERAGE,Threshold=15,Unit=PERCENT,Dimensions=[{Key=JobFlowId,Value="${emr.clusterId}"}]}}]}
```

The following command illustrates using the command line to provide the automatic scaling policy definition as part of an instance group configuration file named `instancegroupconfig.json`.

```bash
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:

```json
[
    {
        "InstanceCount": 1,
        "Name": "MyMasterIG",
        "InstanceGroupType": "MASTER",
        "InstanceType": "m3.xlarge"
    },
    {
        "InstanceCount": 2,
        "Name": "MyCoreIG",
        "InstanceGroupType": "CORE",
        "InstanceType": "m3.xlarge",
        "AutoScalingPolicy": {}
    }
]"}
Using Automatic Scaling in Amazon EMR

"Constraints":
{
  "MinCapacity": 2,
  "MaxCapacity": 10
},
"Rules":
[
{
  "Name": "Default-scale-out",
  "Description": "Replicates the default scale-out rule in the console for YARN memory.",
  "Action":{
    "SimpleScalingPolicyConfiguration":{
      "AdjustmentType": "CHANGE_IN_CAPACITY",
      "ScalingAdjustment": 1,
      "CoolDown": 300
    }
  },
  "Trigger":{
    "CloudWatchAlarmDefinition":{
      "ComparisonOperator": "LESS_THAN",
      "EvaluationPeriods": 1,
      "MetricName": "YARNMemoryAvailablePercentage",
      "Namespace": "AWS/ElasticMapReduce",
      "Period": 300,
      "Threshold": 15,
      "Statistic": "AVERAGE",
      "Unit": "PERCENT",
      "Dimensions":[
        {
          "Key": "JobFlowId",
          "Value": "${emr.clusterId}"
        }
      ]
    }
  }
}
]

Adding an Instance Group with an Automatic Scaling Policy to a Cluster

You can specify a scaling policy configuration using the --instance-groups option with the add-instance-groups command in the same way you can when you use create-cluster. The following example uses a reference to a JSON file, instancegroupconfig.json, with the instance group configuration.

```
aws emr add-instance-groups --cluster-id j-1EKZ3TYEVF1S2 --instance-groups file://your/path/to/instancegroupconfig.json
```

Applying an Automatic Scaling Policy to an Existing Instance Group or Modifying an Applied Policy

Use the aws emr put-auto-scaling-policy command to apply an automatic scaling policy to an existing instance group. The instance group must be part of a cluster that uses the automatic scaling IAM role. The following example uses a reference to a JSON file, autoscaleconfig.json, that specifies the automatic scaling policy configuration.
aws emr put-auto-scaling-policy --cluster-id j-1EKZ3TYEVF1S2 --instance-group-id ig-3PLUZBA6WLS07 --auto-scaling-policy file://your/path/to/autoscaleconfig.json

The contents of the autoscaleconfig.json file, which defines the same scale-out rule as shown in the previous example, is shown below.

"AutoScalingPolicy":

   "Constraints":
     {
       "MinCapacity": 2,
       "MaxCapacity": 10
     },

   "Rules":
     [
       {
         "Name": "Default-scale-out",
         "Description": "Replicates the default scale-out rule in the console for YARN memory."
       },

       "Action":{
         "SimpleScalingPolicyConfiguration":{
           "AdjustmentType": "CHANGE_IN_CAPACITY",
           "ScalingAdjustment": 1,
           "CoolDown": 300
         }
       },

       "Trigger":{
         "CloudWatchAlarmDefinition":{
           "ComparisonOperator": "LESS_THAN",
           "EvaluationPeriods": 1,
           "MetricName": "YARNMemoryAvailablePercentage",
           "Namespace": "AWS/ElasticMapReduce",
           "Period": 300,
           "Threshold": 15,
           "Statistic": "AVERAGE",
           "Unit": "PERCENT",
           "Dimensions":[
             {
               "Key": "JobFlowId",
               "Value": "${emr.clusterId}"
             }
           ]
         }
       }
     ]

Removing an Automatic Scaling Policy from an Instance Group

aws emr remove-auto-scaling-policy --cluster-id j-1EKZ3TYEVF1S2 --instance-group-id ig-3PLUZBA6WLS07

Retrieving an Automatic Scaling Policy Configuration

The describe-cluster command retrieves the policy configuration in the InstanceGroup block. For example, the following command retrieves the configuration for the cluster with a cluster ID of j-1CWOHP4PI3OVJ.
aws emr describe-cluster --cluster-id j-1CWOHP4P130VJ

The command produces the following example output.

```
{
   "Cluster": {
      "Id": "j-1CWOHP4P130VJ",
      "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": {
               "State": "ATTACHED",
               "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-1CWOHP4P130VJ"
                           }
                        ],
                        "EvaluationPeriods": 1,
                        "Period": 300,
```
"ComparisonOperator": "LESS_THAN",
"Statistic": "AVERAGE"
}
}
"Description": "",
"Action": {
"SimpleScalingPolicyConfiguration": {
"CoolDown": 300,
"AdjustmentType": "CHANGE_IN_CAPACITY",
"ScalingAdjustment": 1
}
}
",
"Name": "Default-scale-in",
"Trigger": {
"CloudWatchAlarmDefinition": {
"MetricName": "YARNMemoryAvailablePercentage",
"Unit": "PERCENT",
"Namespace": "AWS/ElasticMapReduce",
"Threshold": 0.75,
"Dimensions": [
{"Key": "JobFlowId",
"Value": "j-1CWOHP4PI3O7VJ"}
],
"EvaluationPeriods": 1,
"Period": 300,
"ComparisonOperator": "GREATER_THAN",
"Statistic": "AVERAGE"
}
},
"Description": "",
"Action": {
"SimpleScalingPolicyConfiguration": {
"CoolDown": 300,
"AdjustmentType": "CHANGE_IN_CAPACITY",
"ScalingAdjustment": -1
}
}
]
],
"Configurations": [],
"InstanceType": "m3.xlarge",
"Market": "ON_DEMAND",
"Name": "Core - 2",
"ShrinkPolicy": {},
"Status": {
"Timeline": {
"CreationDateTime": 1479413437.342,
"ReadyDateTime": 1479413864.615
},
"State": "RUNNING",
"StateChangeReason": {
"Message": ""
}
}
,"RunningInstanceCount": 2,
"Id": "ig-3M168X8E8C3PH1",
"InstanceGroupType": "CORE",
"RequestedInstanceCount": 2,
"EbsBlockDevices": []}
Manually Resizing a Running Cluster

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

You can resize the core instance group in a running cluster by adding nodes using the AWS Management Console, AWS CLI, or the Amazon EMR API. You cannot shrink the size of the core instance group in a running cluster by reducing the instance count. However, it is possible to terminate an instance in the core instance group using the AWS CLI or the API. This should be done with caution. Terminating an instance in the core instance group risks data loss, and the instance is not automatically replaced.

Task nodes also run your Hadoop jobs. After a cluster is running, you can increase or decrease the number of task nodes, and you can add additional task instance groups using the AWS Management Console, AWS CLI, or the Amazon EMR API.

When your cluster runs, Hadoop determines the number of mapper and reducer tasks needed to process the data. Larger clusters should have more tasks for better resource use and shorter processing time. Typically, an EMR cluster remains the same size during the entire cluster; you set the number of tasks when you create the cluster. When you resize a running cluster, you can vary the processing during the


Manually Resizing a Running Cluster

cluster execution. Therefore, instead of using a fixed number of tasks, you can vary the number of tasks during the life of the cluster. There are two configuration options to help set the ideal number of tasks:

- mapred.map.tasksperslot
- mapred.reduce.tasksperslot

You can set both options in the mapred-conf.xml file. When you submit a job to the cluster, the job client checks the current total number of map and reduce slots available clusterwide. The job client then uses the following equations to set the number of tasks:

- mapred.map.tasks = mapred.map.tasksperslot * map slots in cluster
- mapred.reduce.tasks = mapred.reduce.tasksperslot * reduce slots in cluster

The job client only reads the tasksperslot parameter if the number of tasks is not configured. You can override the number of tasks at any time, either for all clusters via a bootstrap action or individually per job by adding a step to change the configuration.

Amazon EMR withstands slave node failures and continues cluster execution even if a slave node becomes unavailable. Amazon EMR automatically provisions additional slave nodes to replace those that fail.

You can have a different number of slave nodes for each cluster step. You can also add a step to a running cluster to modify the number of slave nodes. Because all steps are guaranteed to run sequentially by default, you can specify the number of running slave nodes for any step.

Arrested State

An instance group goes into arrested state if it encounters too many errors while trying to start the new cluster nodes. For example, if new nodes fail while performing bootstrap actions, the instance group goes into an ARRESTED state, rather than continuously provisioning new nodes. After you resolve the underlying issue, reset the desired number of nodes on the cluster's instance group, and then the instance group resumes allocating nodes. Modifying an instance group instructs Amazon EMR to attempt to provision nodes again. No running nodes are restarted or terminated.

In the AWS CLI, the list-instances subcommand returns all instances and their states as does the describe-cluster subcommand. If Amazon EMR detects a fault with an instance group, it changes the group's state to ARRESTED.

To reset a cluster in an ARRESTED state using the AWS CLI

Type the describe-cluster subcommand with the --cluster-id parameter to view the state of the instances in your cluster.

- To view information on all instances and instance groups in a cluster, type the following command and replace j-3KVXXXXXXXY7UG with the cluster ID.

```bash
aws emr describe-cluster --cluster-id j-3KVXXXXXXXY7UG
```

The output displays information about your instance groups and the state of the instances:

```json
{
    "Cluster": {
        "Status": {
            "Timeline": {
                "ReadyDateTime": 1413187781.245,
                "CreationDateTime": 1413187405.356
            }
        }
    }
}
```
To view information on a particular instance group, type the `list-instances` subcommand with the `--cluster-id` and `--instance-group-types` parameters. You can view information for the MASTER, CORE, or TASK groups.

```
aws emr list-instances --cluster-id j-3KVXXXXXY7UG --instance-group-types "CORE"
```
Use the `modify-instance-groups` subcommand with the `--instance-groups` parameter to reset a cluster in the ARRESTED state. The instance group id is returned by the `describe-cluster` subcommand.

```
aws emr modify-instance-groups --instance-groups
  InstanceGroupId=ig-3SUXXXXXXXXQ92M,InstanceCount=3
```
Legacy Clusters

Before October 2010, Amazon EMR did not have the concept of instance groups. Clusters developed for Amazon EMR that were built before the option to resize running clusters was available are considered legacy clusters. Previously, the Amazon EMR architecture did not use instance groups to manage nodes and only one type of slave node existed. Legacy clusters reference slaveInstanceType and other now deprecated fields. Amazon EMR continues to support the legacy clusters; you do not need to modify them to run them correctly.

Cluster Behavior

If you run a legacy cluster and only configure master and slave nodes, you observe a slaveInstanceType and other deprecated fields associated with your clusters.

Mapping Legacy Clusters to Instance Groups

Before October 2010, all cluster nodes were either master nodes or slave nodes. An Amazon EMR configuration could typically be represented like the following diagram.

**Old Amazon EMR Model**

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A legacy cluster launches and a request is sent to Amazon EMR to start the cluster.</td>
</tr>
<tr>
<td>2</td>
<td>Amazon EMR creates a Hadoop cluster.</td>
</tr>
<tr>
<td>3</td>
<td>The legacy cluster runs on a cluster consisting of a single master node and the specified number of slave nodes.</td>
</tr>
</tbody>
</table>

Clusters created using the older model are fully supported and function as originally designed. The Amazon EMR API and commands map directly to the new model. Master nodes remain master nodes and become part of the master instance group. Slave nodes still run HDFS and become core nodes and join the core instance group.

**Note**

No task instance group or task nodes are created as part of a legacy cluster, however you can add them to a running cluster at any time.

The following diagram illustrates how a legacy cluster now maps to master and core instance groups.
Old Amazon EMR Model Remapped to Current Architecture

1 A request is sent to Amazon EMR to start a cluster.
2 Amazon EMR creates an Hadoop cluster with a master instance group and core instance group.
3 The master node is added to the master instance group.
4 The slave nodes are added to the core instance group.

Cluster Scale-Down

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

With Amazon EMR release version 5.1.0 and later, there are two options for scale-down behavior: terminate at the instance-hour boundary for Amazon EC2 billing, or terminate at task completion. Starting with Amazon EMR release version 5.10.0, the terminate at instance-hour boundary setting is deprecated because of the introduction of per-second billing in Amazon EC2. We do not recommend specifying termination at the instance-hour boundary in versions where the option is available.

Warning

If you use the AWS CLI to issue a `modify-instance-groups` with `EC2InstanceIdsToTerminate`, these instances are terminated immediately, without consideration for these settings, and regardless of the status of applications running on them. Terminating an instance in this way risks data loss and unpredictable cluster behavior.
When terminate at task completion is specified, Amazon EMR blacklists and drains tasks from nodes before terminating the Amazon EC2 instances. With either behavior specified, Amazon EMR does not terminate Amazon EC2 instances in core instance groups if it could lead to HDFS corruption.

**Terminate at Task Completion**

Amazon EMR allows you to scale down your cluster without affecting your workload. Amazon EMR gracefully decommissions YARN, HDFS, and other daemons on core and task nodes during a resize down operation without losing data or interrupting jobs. Amazon EMR only shrinks instance groups if the work assigned to the groups has completed and they are idle. For YARN NodeManager decommissioning, you can manually adjust the time a node waits for decommissioning by setting `yarn.resourcemanager.decommissioning.timeout` inside `/etc/hadoop/conf/yarn-site.xml`; this setting is dynamically propagated. If there are still running containers or YARN applications when the decommissioning timeout passes, the node is forced to be decommissioned and YARN reschedules affected containers on other nodes. The default value is 3600s (one hour), meaning a YARN node under that has been issued a resize down request will become decommissioned within one hour or less. You can set this timeout to be an arbitrarily high value to force graceful shrink to wait longer.

**Task Node Groups**

Amazon EMR will intelligently select instances which are not running tasks and remove them from a cluster first. If all instances in the cluster are being utilized, Amazon EMR will wait for tasks to complete on a given instance before removing it from the cluster. The default wait time is 1 hour and this value can be changed by setting `yarn.resourcemanager.decommissioning.timeout`. Amazon EMR will dynamically use the new setting. You can set this to an arbitrarily large number to ensure that no tasks are killed while shrinking the cluster.

**Core Node Groups**

On core nodes, both YARN NodeManager and HDFS DataNode daemons must be decommissioned in order for the instance group to shrink. For YARN, graceful shrink ensures that a node marked for decommissioning is only transitioned to the `DECOMMISSIONED` state if there are no pending or incomplete containers or applications. The decommissioning finishes immediately if there are no running containers on the node at the beginning of decommissioning.

For HDFS, graceful shrink ensures that the target capacity of HDFS is large enough to fit all existing blocks. If the target capacity is not large enough, only a partial amount of core instances are decommissioned such that the remaining nodes can handle the current data residing in HDFS. You should ensure additional HDFS capacity to allow further decommissioning. You should also try to minimize write I/O before attempting to shrink instance groups as that may delay the completion of the resize operation.

Another limit is the default replication factor, `dfs.replication` inside `/etc/hadoop/conf/hdfs-site`. Amazon EMR configures the value based on the number of instances in the cluster: 1 with 1-3 instances, 2 for clusters with 4-9 instances, and 3 for clusters with 10+ instances. Graceful shrink does not allow you to shrink core nodes below the HDFS replication factor; this is to prevent HDFS from being unable to close files due insufficient replicas. To circumvent this limit, you must lower the replication factor and restart the NameNode daemon.

**Configuring Amazon EMR Scale-Down Behavior**

*Note*

This configuration feature is only available for Amazon EMR releases 5.1.0 or later.

You can use the AWS Management Console, the AWS CLI, or the Amazon EMR API to configure scale-down behavior when you create a cluster. Configuring scale-down using the AWS Management Console is done in the **Step 3: General Cluster Settings** screen when you create a cluster using **Advanced options**.
When you create a cluster using the AWS CLI, use the `--ScaleDownBehavior` option to specify either `TERMINATE_AT_INSTANCE_HOUR` or `TERMINATE_AT_TASK_COMPLETION`.

**Cloning a Cluster Using the Console**

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

You can use the Amazon EMR console to clone a cluster, which makes a copy of the configuration of the original cluster to use as the basis for a new cluster.

**To clone a cluster using the console**

1. From the **Cluster List** page, click a cluster to clone.
2. At the top of the **Cluster Details** page, click **Clone**.

   In the dialog box, choose **Yes** to include the steps from the original cluster in the cloned cluster. Choose **No** to clone the original cluster's configuration without including any of the steps.

   **Note**

   For clusters created using AMI 3.1.1 and later (Hadoop 2.x) or AMI 2.4.8 and later (Hadoop 1.x), if you clone a cluster and include steps, all system steps (such as configuring Hive) are cloned along with user-submitted steps, up to 1,000 total. Any older steps that no longer appear in the console's step history cannot be cloned. For earlier AMIs, only 256 steps can be cloned (including system steps). For more information, see **Submit Work to a Cluster** (p. 493).

3. The **Create Cluster** page appears with a copy of the original cluster's configuration. Review the configuration, make any necessary changes, and then click **Create Cluster**.

**Submit Work to a Cluster**

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

This section describes the methods for submitting work to an Amazon EMR cluster. You can submit work to a cluster by adding steps or by interactively submitting Hadoop jobs to the master node. The maximum number of PENDING and ACTIVE steps allowed in a cluster is 256. You can submit jobs interactively to the master node even if you have 256 active steps running on the cluster. You can submit...
an unlimited number of steps over the lifetime of a long-running cluster, but only 256 steps can be
ACTIVE or PENDING at any given time.

For clusters created using AMI version 3.1.0 and earlier (Hadoop 2.x) or AMI version 2.4.7 and earlier
(Hadoop 1.x), the total number of steps available over the lifetime of a cluster is limited to 256. For more
information about how to overcome this limitation, see Add More than 256 Steps to a Cluster (p. 498).

Topics
• Work with Steps Using the CLI and Console (p. 494)
• Submit Hadoop Jobs Interactively (p. 496)
• Add More than 256 Steps to a Cluster (p. 498)

Work with Steps Using the CLI and Console

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

You can add steps to a cluster using the AWS Management Console, the AWS CLI, or the Amazon EMR
API. The maximum number of PENDING and ACTIVE steps allowed in a cluster is 256, which includes
system steps such as install Pig, install Hive, install HBase, and configure debugging. You can submit an
unlimited number of steps over the lifetime of a long-running cluster, but only 256 steps can be ACTIVE
or PENDING at any given time. With EMR version 4.8.0 and later, except version 5.0.0, you can cancel
steps that are PENDING using the AWS Management Console, the AWS CLI, or the Amazon EMR API.

Adding Steps to a Cluster

You can add steps to a cluster using the AWS CLI, the Amazon EMR SDK, or the AWS Management
Console. Using the AWS Management Console, you can add steps to a cluster when the cluster is created.
You can also add steps to a long-running cluster—that is, a cluster with the auto-terminate option
disabled.

Add Steps Using the Console

Whether you add steps during cluster creation or to a cluster, the procedure is similar to the following.

To add a step to a running cluster using the AWS Management Console

1. In the Amazon EMR console, on the Cluster List page, click the link for your cluster.
2. On the Cluster Details page, expand the Steps section, and then click Add step.
3. Type appropriate values in the fields in the Add Step dialog, and then click Add. Depending on the
   step type, the options are different.

Add Steps Using the AWS CLI

The following procedures demonstrate adding steps to a newly-created cluster and to a running cluster
using the AWS CLI. In both examples, the --steps subcommand is used to add steps to the cluster.

To add a step during cluster creation

• Type the following command to create a cluster and add a Pig step. Replace myKey with the name of
  your EC2 key pair and replace mybucket with the name of your Amazon S3 bucket.

  Linux, UNIX, and Mac OS X users:
aws emr create-cluster --name "Test cluster" --ami-version 2.4 --applications
   Name=Hive Name=Pig \
   --use-default-roles --ec2-attributes KeyName=myKey \
   --instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m3.xlarge
   InstanceGroupType=CORE,InstanceCount=2,InstanceType=m3.xlarge \
   --steps Type=PIG,Name="Pig Program",ActionOnFailure=CONTINUE,Args=[-f,s3://mybucket/
   scripts/pigscript.pig,-p,INPUT=s3://mybucket/inputdata/,-p,OUTPUT=s3://mybucket/
   outputdata/,#INPUT=s3://mybucket/inputdata/,#OUTPUT=s3://mybucket/outputdata/]

Windows users:
aws emr create-cluster --name "Test cluster" --ami-version 2.4 --applications
   Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey --
   instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m3.xlarge
   InstanceGroupType=CORE,InstanceCount=2,InstanceType=m3.xlarge --steps
   Type=PIG,Name="Pig Program",ActionOnFailure=CONTINUE,Args=[-f,s3://mybucket/
   scripts/pigscript.pig,-p,INPUT=s3://mybucket/inputdata/,-p,OUTPUT=s3://mybucket/
   outputdata/,#INPUT=s3://mybucket/inputdata/,#OUTPUT=s3://mybucket/outputdata/]

Note
The list of arguments changes depending on the type of step.

The output is a cluster identifier similar to the following:

```json
{
   "ClusterId": "j-2AXXXXXXGAPLF"
}
```

To add a step to a running cluster

- Type the following command to add a step to a running cluster. Replace `j-2AXXXXXXGAPLF` with your cluster ID and replace `mybucket` with your Amazon S3 bucket name.

```bash
aws emr add-steps --cluster-id j-2AXXXXXXGAPLF --steps Type=PIG,Name="Pig Program",Args=[-f,s3://mybucket/scripts/pigscript.pig,-p,INPUT=s3://mybucket/inputdata/,-p,OUTPUT=s3://mybucket/outputdata/,#INPUT=s3://mybucket/inputdata/,#OUTPUT=s3://mybucket/outputdata/]
```

The output is a step identifier similar to the following:

```json
{
   "StepIds": [
      "s-Y9XXXXXXAPMD"
   ]
}
```

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

**Viewing Steps**

The total number of step records you can view (regardless of status) is 1,000. This total includes both user-submitted and system steps. As the status of user-submitted steps changes to COMPLETED or FAILED, additional user-submitted steps can be added to the cluster until the 1,000 step limit is reached.
After 1,000 steps have been added to a cluster, the submission of additional steps causes the removal of older, user-submitted step records. These records are not removed from the log files. They are removed from the console display, and they do not appear when you use the CLI or API to retrieve cluster information. System step records are never removed.

The step information you can view depends on the mechanism used to retrieve cluster information. The following tables indicate the step information returned by each of the available options.

<table>
<thead>
<tr>
<th>Option</th>
<th>DescribeJobFlow or --describe --jobflow</th>
<th>ListSteps or list-steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDK</td>
<td>256 steps</td>
<td>1,000 steps</td>
</tr>
<tr>
<td>Amazon EMR CLI</td>
<td>256 steps</td>
<td>NA</td>
</tr>
<tr>
<td>AWS CLI</td>
<td>NA</td>
<td>1,000 steps</td>
</tr>
<tr>
<td>API</td>
<td>256 steps</td>
<td>1,000 steps</td>
</tr>
</tbody>
</table>

**Cancel Pending Steps**

You can cancel steps using the the AWS Management Console, the AWS CLI, or the Amazon EMR API. Only steps that are PENDING can be canceled.

**To cancel steps using the AWS Management Console**

1. In the Amazon EMR console, on the Cluster List page, choose the link for the cluster.
2. On the Cluster Details page, expand the Steps section.
3. For each step you want to cancel, select the step from the list of Steps, select Cancel step, and then confirm you want to cancel the step.

**To cancel steps using the AWS CLI**

- Use the `aws emr cancel-steps` command, specifying the cluster and steps to cancel. The following example demonstrates an AWS CLI command to cancel two steps.

```
aws emr cancel-steps --ClusterID j-2QUAJ7T3OTEI8 --StepIds s-3M8DKCZYY1QW,s-3M8DKCZYY1QW
```

**Submit Hadoop Jobs Interactively**

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

In addition to adding steps to a cluster, you can connect to the master node using an SSH client or the AWS CLI and interactively submit Hadoop jobs. For example, you can use PuTTY to establish an SSH connection with the master node and submit interactive Hive queries which are compiled into one or more Hadoop jobs.

You can submit Hadoop jobs interactively by establishing an SSH connection to the master node (using an SSH client such as PuTTY or OpenSSH) or by using the ssh subcommand in the AWS CLI. You can submit jobs interactively to the master node even if you have 256 active steps running on the cluster. Note however that log records associated with interactively submitted jobs are included in the “step
created jobs" section of the currently running step's controller log. For more information about step logs, see View Log Files (p. 422).

The following examples demonstrate interactively submitting Hadoop jobs and Hive jobs to the master node. The process for submitting jobs for other programming frameworks (such as Pig) is similar to these examples.

**To submit Hadoop jobs interactively using the AWS CLI**

- You can submit Hadoop jobs interactively using the AWS CLI by establishing an SSH connection in the CLI command (using the `ssh` subcommand). To copy a JAR file from your local Windows machine to the master node's file system, type the following command. Replace `j-2A6HXXXXXXL7J` with your cluster ID, replace `mykey.ppk` with the name of your key pair file, and replace `myjar.jar` with the name of your JAR file.

```bash
aws emr put --cluster-id j-2A6HXXXXXXL7J --key-pair-file "C:\Users\username\Desktop\Keys\mykey.ppk" --src "C:\Users\username\myjar.jar"
```

To create an SSH connection and submit the Hadoop job `myjar.jar`, type the following command.

```bash
aws emr ssh --cluster-id j-2A6HXXXXXXL7J --key-pair-file "C:\Users\username\Desktop\Keys\mykey.ppk" --command "hadoop jar myjar.jar"
```

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

**To interactively submit Hive jobs using the AWS CLI**

In addition to submitting jobs to the master node via JAR files, you can submit jobs by interacting with one of the Hadoop programming frameworks running on the master node. For example, you can interactively submit Hive queries or Pig transformations at the command line, or you can submit scripts to the cluster for processing. Your commands or scripts are then compiled into one or more Hadoop jobs.

The following procedure demonstrates running a Hive script using the AWS CLI.

1. If Hive is not installed on the cluster, type the following command to install it. Replace `j-2A6HXXXXXXL7J` with your cluster ID.

   ```bash
   aws emr install-applications --cluster-id j-2A6HXXXXXXL7J --apps Name=Hive
   ```

2. Create a Hive script file containing the queries or commands to run. The following example script named `my-hive.q` creates two tables, `aTable` and `anotherTable`, and copies the contents of `aTable` to `anotherTable`, replacing all data.

   ```sql
   ---- sample Hive script file: my-hive.q ----
   create table aTable (aColumn string);
   create table anotherTable like aTable;
   insert overwrite table anotherTable select * from aTable
   ```

3. Type the following commands to run the script from the command line using the `ssh` subcommand.

   To copy `my-hive.q` from a Windows machine to your cluster, type the following command. Replace `j-2A6HXXXXXXL7J` with your cluster ID and replace `mykey.ppk` with the name of your key pair file.

   ```bash
   aws emr put --cluster-id j-2A6HXXXXXXL7J --key-pair-file "C:\Users\username\Desktop\Keys\mykey.ppk" --src "C:\Users\username\my-hive.q"
   ```
Add More than 256 Steps to a Cluster

To create an SSH connection and submit the Hive script `my-hive.q`, type the following command.

```
aws emr ssh --cluster-id j-2A6HXXXXXTJ7J --key-pair-file "C:\Users\username\Desktop\Keys\mykey.ppk" --command "hive -f my-hive.q"
```

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

Add More than 256 Steps to a Cluster

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Beginning with AMI 3.1.1 (Hadoop 2.x) and AMI 2.4.8 (Hadoop 1.x), you can submit an unlimited number of steps over the lifetime of a long-running cluster, but only 256 can be active or pending at any given time. For earlier AMI versions, the total number of steps that can be processed by a cluster is limited to 256 (including system steps such as install Hive and install Pig). For more information, see Submit Work to a Cluster (p. 493).

You can use one of several methods to overcome the 256 step limit in pre-3.1.1 and pre-2.4.8 AMIs:

1. Have each step submit several jobs to Hadoop. This does not allow you unlimited steps in pre-3.1.1 and pre-2.4.8 AMIs, but it is the easiest solution if you need a fixed number of steps greater than 256.
2. Write a workflow program that runs in a step on a long-running cluster and submits jobs to Hadoop.
   You could have the workflow program either:
   • Listen to an Amazon SQS queue to receive information about new steps to run.
   • Check an Amazon S3 bucket on a regular schedule for files containing information about the new steps to run.
3. Write a workflow program that runs on an EC2 instance outside of Amazon EMR and submits jobs to your long-running clusters using SSH.
4. Connect to your long-running cluster via SSH and submit Hadoop jobs using the Hadoop API. For more information, see http://hadoop.apache.org/docs/current/api/org/apache/hadoop/mapred/JobClient.html.
5. Connect to the master node using an SSH client (such as PuTTY or OpenSSH) and manually submit jobs to the cluster or use the `ssh` subcommand in the AWS CLI to both connect and submit jobs. For more information about establishing an SSH connection with the master node, see Connect to the Master Node Using SSH (p. 457). For more information about interactively submitting Hadoop jobs, see Submit Hadoop Jobs Interactively (p. 496).

Automate Recurring Clusters with AWS Data Pipeline

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

AWS Data Pipeline is a service that automates the movement and transformation of data. You can use it to schedule moving input data into Amazon S3 and to schedule launching clusters to process that data. For example, consider the case where you have a web server recording traffic logs. If you want to run a
weekly cluster to analyze the traffic data, you can use AWS Data Pipeline to schedule those clusters. AWS Data Pipeline is a data-driven workflow, so that one task (launching the cluster) can be dependent on another task (moving the input data to Amazon S3). It also has robust retry functionality.

For more information about AWS Data Pipeline, see the AWS Data Pipeline Developer Guide, especially the tutorials regarding Amazon EMR:

- Tutorial: Launch an Amazon EMR Job Flow
- Getting Started: Process Web Logs with AWS Data Pipeline, Amazon EMR, and Hive
- Tutorial: Amazon DynamoDB Import and Export Using AWS Data Pipeline
Troubleshoot a Cluster

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

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

A cluster hosted by Amazon EMR runs in a complex ecosystem made up of several types of open-source software, custom application code, and Amazon Web Services. An issue in any of these parts can cause the cluster to fail or take longer than expected to complete. The following topics will help you figure out what has gone wrong in your cluster and give you suggestions on how to fix it.

Topics
- What Tools are Available for Troubleshooting? (p. 500)
- Viewing and Restarting Amazon EMR and Application Processes (Daemons) (p. 502)
- Known Issues with Amazon EMR AMIs (p. 503)
- Troubleshoot a Failed Cluster (p. 509)
- Troubleshoot a Slow Cluster (p. 513)
- Common Errors in Amazon EMR (p. 520)

When you are developing a new Hadoop application, we recommend that you enable debugging and process a small but representative subset of your data to test the application. You may also want to run the application step-by-step to test each step separately. For more information, see Configure Cluster Logging and Debugging (p. 167) and Step 5: Test the Cluster Step by Step (p. 513).

What Tools are Available for Troubleshooting?

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

There are several tools you can use to gather information about your cluster to help determine what went wrong. Some require that you initialize them when you launch the cluster; others are available for every cluster.

Topics
- Tools to Display Cluster Details (p. 500)
- Tools to View Log Files (p. 501)
- Tools to Monitor Cluster Performance (p. 501)

Tools to Display Cluster Details

You can use the AWS Management Console, AWS CLI, or EMR API to retrieve detailed information about an EMR cluster and job execution. For more information about using the AWS Management Console and AWS CLI, see View Cluster Status and Details (p. 416).
Amazon EMR Console Details Pane

In the Clusters list on the Amazon EMR console you can see high-level information about the status of each cluster in your account and region. The list displays all clusters that you have launched in the past two months, regardless of whether they are active or terminated. From the Clusters list, you can select a cluster Name to view cluster details. This information is organized in different categories to make it easy to navigate.

The Application history available in the cluster details page can be particularly useful for troubleshooting. It provides status of YARN applications, and for some, such as Spark applications you can drill into different metrics and facets, such as jobs, stages, and executors. For more information, see View Application History (p. 421). This feature is available only in Amazon EMR version 5.8.0 and later.

Amazon EMR Command Line Interface

You can locate details about a cluster from the CLI using the --describe argument.

Amazon EMR API

You can locate details about a cluster from the API using the DescribeJobFlows action.

Tools to View Log Files

Amazon EMR and Hadoop both generate log files as the cluster runs. You can access these log files from several different tools, depending on the configuration you specified when you launched the cluster. For more information, see Configure Cluster Logging and Debugging (p. 167).

Log Files on the Master Node

Every cluster publishes logs files to the /mnt/var/log/ directory on the master node. These log files are only available while the cluster is running.

Log Files Archived to Amazon S3

If you launch the cluster and specify an Amazon S3 log path, the cluster copies the log files stored in /mnt/var/log/ on the master node to Amazon S3 in 5-minute intervals. This ensures that you have access to the log files even after the cluster is terminated. Because the files are archived in 5-minute intervals, the last few minutes of an suddenly terminated cluster may not be available.

Tools to Monitor Cluster Performance

Amazon EMR provides several tools to monitor the performance of your cluster.

Hadoop Web Interfaces

Every cluster publishes a set of web interfaces on the master node that contain information about the cluster. You can access these web pages by using an SSH tunnel to connect them on the master node. For more information, see View Web Interfaces Hosted on Amazon EMR Clusters (p. 461).

CloudWatch Metrics

Every cluster reports metrics to CloudWatch. CloudWatch is a web service that tracks metrics, and which you can use to set alarms on those metrics. For more information, see Monitor Metrics with CloudWatch (p. 434).
Ganglia

Ganglia is a cluster monitoring tool. To have this available, you have to install Ganglia on the cluster when you launch it. After you’ve done so, you can monitor the cluster as it runs by using an SSH tunnel to connect to the Ganglia UI running on the master node. For more information, see Monitor Performance with Ganglia (p. 450).

Viewing and Restarting Amazon EMR and Application Processes (Daemons)

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

When you troubleshoot a cluster, you may want to list running processes. You may also find it useful to stop or restart processes in some circumstances—for example, after you change a configuration or notice a problem with a particular process after you analyze log files and error messages.

There are two types of processes that run on a cluster: Amazon EMR processes (for example, instance-controller and Log Pusher), and processes associated with the applications installed on the cluster (for example, hadoop-hdfs-namenode, and hadoop-yarn-resource-manager).

To work with processes directly on a cluster, you connect to the master node. For more information, see Connect to the Cluster (p. 456).

Viewing Running Processes

If you are using Amazon EMR versions earlier than 4.x, both Amazon EMR processes and processes associated with applications you install run as part of the SysV init system via init.d scripts.

To view a list of running processes

- Type the following command (without the $, which indicates the Linux command prompt):

```
# ls /etc/init.d/
```

The command returns a list of running processes similar to the following example:

```
acpid        hadoop-hdfs-namenode        instance-controller
```

Restarting Processes

After you determine which processes are running, you can stop and then restart them if necessary.

To restart a process

1. Type the following command to stop the process, replacing `processname` with the process name returned by the `ls` command in the procedure above:

```
# sudo /etc/init.d/processname stop
```
Known Issues with Amazon EMR AMIs

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

This section presents known issues with Amazon Machine Images (AMIs) supported by Amazon EMR. AMIs with significant issues are deprecated and should not be used. For a complete list of supported and deprecated AMIs, see: AMI Versions Supported in Amazon EMR Versions 2.x and 3.x (p. 73).

In many cases, known issues are corrected in newer AMIs. Customers are encouraged to use the latest AMI available. To ensure you are using the latest AMI in a series using the AWS CLI, specify the major and minor version number using the `--ami-version` parameter; for example, `--ami-version 3.2`. For more information on specifying AMI versions, see: AMI Version Numbers (Versions 2.x, 3.x) (p. 70).

**General Issues**

**HBase Shell Excessive Debug Logging**

When using the HBase shell, the logging may be so excessive that you can tell what is happening in the shell, which makes it unusable. This logging verbosity is the same for any other process that uses Zookeeper, so there is also a possibility that `/mnt` will become full. Use this bootstrap action invocation example as a guide to work around this issue:

```
aws emr create-cluster --ami-version 3.9.0 --instance-type m3.xlarge --instance-count 2 --ec2-attributes KeyName=myKey --use-default-roles --applications Name=HBase --bootstrap-actions --bootstrap-actions Path=s3://support.elasticmapreduce/bootstrap-actions/misc/run-patch.bash,Args="["s3://support.elasticmapreduce/patch/ami-3.x-fix-excessive-zookeeper-logging-all.patch"]"
```

**Known Issues with Hadoop 2.4.0 AMIs**

The following known issues impact the Hadoop 2.4.0 AMIs. Where appropriate, the AMI that resolves the issue is indicated. For workarounds to known issues or to get assistance migrating to a newer AMI, contact your AWS support representative.

**Issues with AMI 3.2.1**

<table>
<thead>
<tr>
<th>Issue</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMRFS DynamoDB metadata table is created in region us-east-1</td>
<td>Resolved in AMI 3.2.3</td>
</tr>
<tr>
<td>Hive server does not start after cluster launch</td>
<td>Resolved in AMI 3.2.3</td>
</tr>
</tbody>
</table>
Issues with AMI 3.2.0

<table>
<thead>
<tr>
<th>Issue</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root volume reaches 100% capacity on instance types with fewer than 3 instance store volumes causing jobs and/or clusters to fail</td>
<td>Resolved in AMI 3.2.1</td>
</tr>
<tr>
<td>Impacts these instance types: c3.xlarge, c3.2xlarge, c3.4xlarge, c3.8xlarge, c1.medium, m3.xlarge, m3.2xlarge, m2.xlarge, m2.2xlarge, m2.4xlarge, m1.medium, m1.large, hi1.4xlarge</td>
<td></td>
</tr>
</tbody>
</table>

Issues with AMI 3.1.0 (deprecated)

<table>
<thead>
<tr>
<th>Issue</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unable to access ResourceManager and NameNode interfaces via Lynx browser on master node using URL indicated in SSH connection details: lynx://localhost:9026 and lynx://localhost:9101</td>
<td>Replace localhost with the master private DNS name: lynx://master_private_DNS:9026</td>
</tr>
<tr>
<td></td>
<td>SSH connection details are correct in AMI 3.1.2</td>
</tr>
<tr>
<td>Repeating the instance controller causes communication failures that result in unexpected behavior:</td>
<td>Resolved in AMI 3.1.3 and 3.2.3</td>
</tr>
<tr>
<td>• Resizing core nodes is unsuccessful</td>
<td></td>
</tr>
<tr>
<td>• Console reports an incorrect number of core nodes and may indicate 1 or more nodes terminated due to scheduled retirement</td>
<td></td>
</tr>
<tr>
<td>• Jobs or clusters fail</td>
<td></td>
</tr>
</tbody>
</table>

Known Issues with Hadoop 2.2.0 AMIs

The following known issues impact the Hadoop 2.2.0 AMIs. Where appropriate, the AMI that resolves the issue is indicated. For workarounds to known issues or to get assistance migrating to a newer AMI, contact your AWS support representative.

Issues with AMI 3.0.4 (deprecated)

<table>
<thead>
<tr>
<th>Issue</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon Linux kernel upgrade causes Ganglia installation failure</td>
<td>Resolved in the Ganglia bootstrap action</td>
</tr>
<tr>
<td>ERROR: java.lang.UnsupportedOperationException: This is supposed to be overridden by subclasses... when creating an HBase snapshot</td>
<td>Resolved in AMI 3.1.0</td>
</tr>
</tbody>
</table>
## Known Issues with Hadoop 2.2.0 AMIs

<table>
<thead>
<tr>
<th>Issue</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Received IOException while reading '...', attempting to reopen...received intermittently while reading S3 input</td>
<td>Resolved in AMI 3.1.0</td>
</tr>
<tr>
<td>Hadoop NameNode and DataNode heap size incorrectly default to 1GB and are not configurable using the configure-daemons bootstrap action</td>
<td>Resolved in AMI 3.1.0</td>
</tr>
</tbody>
</table>

### Issues with AMI 3.0.3 (deprecated)

<table>
<thead>
<tr>
<th>Issue</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR: java.lang.UnsupportedOperationException: This is supposed to be overridden by subclasses... when creating an HBase snapshot</td>
<td>Resolved in AMI 3.1.0</td>
</tr>
<tr>
<td>Received IOException while reading '...', attempting to reopen...received intermittently while reading S3 input</td>
<td>Resolved in AMI 3.1.0</td>
</tr>
<tr>
<td>Hadoop NameNode and DataNode heap size incorrectly default to 1GB and are not configurable using the configure-daemons bootstrap action</td>
<td>Resolved in AMI 3.1.0</td>
</tr>
<tr>
<td>yum update command produces the following error: Transaction check error: file [...] from install of [...] conflicts with file from package [...]</td>
<td>Resolved in AMI 3.0.4</td>
</tr>
</tbody>
</table>

### Issues with AMI 3.0.2 (deprecated)

<table>
<thead>
<tr>
<th>Issue</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ephemeral volumes on HVM (cc.*) instances are not attached or mounted (instead a 30 GB EBS volume is attached)</td>
<td>Resolved in AMI 3.0.3</td>
</tr>
<tr>
<td>Received IOException while reading '...', attempting to reopen...received intermittently while reading S3 input</td>
<td>Resolved in AMI 3.1.0</td>
</tr>
<tr>
<td>Hadoop NameNode and DataNode heap size incorrectly default to 1GB and are not configurable using the configure-daemons bootstrap action</td>
<td>Resolved in AMI 3.1.0</td>
</tr>
<tr>
<td>yum update command produces the following error: Transaction check error: file [...] from install of [...] conflicts with file from package [...]</td>
<td>Resolved in AMI 3.0.4</td>
</tr>
</tbody>
</table>
### Known Issues with Hadoop 2.2.0 AMIs

<table>
<thead>
<tr>
<th>Issue</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>[... from install of [...] conflicts with file from package [...]</td>
<td></td>
</tr>
</tbody>
</table>

### Issues with AMI 3.0.1 (deprecated)

<table>
<thead>
<tr>
<th>Issue</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ephemeral volumes on HVM (cc.*) instances are not attached or mounted (instead a 30 GB EBS volume is attached)</td>
<td>Resolved in AMI 3.0.3</td>
</tr>
<tr>
<td>Received IOException while reading '...', attempting to reopen... received intermittently while reading S3 input</td>
<td>Resolved in AMI 3.1.0</td>
</tr>
<tr>
<td>Hadoop NameNode and DataNode heap size incorrectly default to 1GB and are not configurable using the configure-daemons bootstrap action</td>
<td>Resolved in AMI 3.1.0</td>
</tr>
<tr>
<td>yum update command produces the following error: Transaction check error: file [...] from install of [...] conflicts with file from package [...]</td>
<td>Resolved in AMI 3.0.4</td>
</tr>
</tbody>
</table>

### Issues with AMI 3.0.0 (deprecated)

<table>
<thead>
<tr>
<th>Issue</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ephemeral volumes on HVM (cc.*) instances are not attached or mounted (instead a 30 GB EBS volume is attached)</td>
<td>Resolved in AMI 3.0.3</td>
</tr>
<tr>
<td>Received IOException while reading '...', attempting to reopen... received intermittently while reading S3 input</td>
<td>Resolved in AMI 3.1.0</td>
</tr>
<tr>
<td>Hadoop NameNode and DataNode heap size incorrectly default to 1GB and are not configurable using the configure-daemons bootstrap action</td>
<td>Resolved in AMI 3.1.0</td>
</tr>
<tr>
<td>yum update command produces the following error: Transaction check error: file [...] from install of [...] conflicts with file from package [...]</td>
<td>Resolved in AMI 3.0.4</td>
</tr>
</tbody>
</table>
Issues with Hadoop 1.0.3 AMIs

The following known issues impact the Hadoop 1.0.3 AMIs. Where appropriate, the AMI that resolves the issue is indicated. For workarounds to known issues or to get assistance migrating to a newer AMI, contact your AWS support representative.

Issues with AMI 2.4.7

<table>
<thead>
<tr>
<th>Issue</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>When using the distcp command with LZO output, some output files are not compressed</td>
<td>Resolved in AMI 2.4.8</td>
</tr>
</tbody>
</table>

Issues with AMI 2.4.3

<table>
<thead>
<tr>
<th>Issue</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>When using the AWS Java SDK, the following exception occurs performing content length validation on Amazon S3 files larger than 2GB: Unable to verify integrity of data download. Client calculated content length didn't match content length received from Amazon S3. The data may be corrupt.</td>
<td>Resolved in AMI 2.4.4</td>
</tr>
<tr>
<td>Amazon EMR logs show numerous messages containing Too many fetch-failures or Error reading task output</td>
<td>Resolved in AMI 2.4.5</td>
</tr>
</tbody>
</table>

Issues with AMI 2.4.2

<table>
<thead>
<tr>
<th>Issue</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon EMR logs show numerous messages containing Too many fetch-failures or Error reading task output</td>
<td>Resolved in AMI 2.4.5</td>
</tr>
</tbody>
</table>

Issues with AMI 2.4.1

<table>
<thead>
<tr>
<th>Issue</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>S3DistCp spawns too many threads, causing other processes to fail when trying to create new threads</td>
<td>Resolved in AMI 2.4.2</td>
</tr>
<tr>
<td>Amazon EMR logs show numerous messages containing Too many fetch-failures or Error reading task output</td>
<td>Resolved in AMI 2.4.5</td>
</tr>
</tbody>
</table>
# Issues with AMI 2.4.0

<table>
<thead>
<tr>
<th>Issue</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Running Java 7 produces the following exception: Communication exception: java.util.ConcurrentModificationException...</td>
<td>Resolved in AMI 2.4.1</td>
</tr>
<tr>
<td>HBase shell does not function properly</td>
<td>Resolved in AMI 2.4.1</td>
</tr>
<tr>
<td>Amazon EMR logs show numerous messages containing Too many fetch-failures or Error reading task output</td>
<td>Resolved in AMI 2.4.5</td>
</tr>
</tbody>
</table>

# Issues with AMI 2.3.6

<table>
<thead>
<tr>
<th>Issue</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon EMR logs show numerous messages containing Too many fetch-failures or Error reading task output</td>
<td>Resolved in AMI 2.4.5</td>
</tr>
<tr>
<td>Clusters with nodes using m3.xlarge or c3.4xlarge instance types fail to launch</td>
<td>Resolved in AMI 2.4.0</td>
</tr>
</tbody>
</table>

# Issues with AMI 2.3.5

<table>
<thead>
<tr>
<th>Issue</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clusters with nodes using m3.xlarge or c3.4xlarge instance types fail to launch</td>
<td>Resolved in AMI 2.4.0</td>
</tr>
</tbody>
</table>

# Issues with AMI 2.3.4 (deprecated)

<table>
<thead>
<tr>
<th>Issue</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clusters with nodes using m3.xlarge or c3.4xlarge instance types fail to launch</td>
<td>Resolved in AMI 2.4.0</td>
</tr>
</tbody>
</table>

# Issues with AMI 2.3.3

<table>
<thead>
<tr>
<th>Issue</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Streaming jobs using an input format other than TextInputFormat insert a byte offset on the first line of an input split causing mappers to fail</td>
<td>Resolved in AMI 2.3.4</td>
</tr>
<tr>
<td>Clusters with nodes using m3.xlarge or c3.4xlarge instance types fail to launch</td>
<td>Resolved in AMI 2.4.0</td>
</tr>
</tbody>
</table>
## Issues with AMI 2.3.2

<table>
<thead>
<tr>
<th>Issue</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clusters with nodes using m3.xlarge or c3.4xlarge instance types fail to launch</td>
<td>Resolved in AMI 2.4.0</td>
</tr>
</tbody>
</table>

## Issues with AMI 2.3.1

<table>
<thead>
<tr>
<th>Issue</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clusters with nodes using m3.xlarge or c3.4xlarge instance types fail to launch</td>
<td>Resolved in AMI 2.4.0</td>
</tr>
</tbody>
</table>

## Issues with AMI 2.3.0

<table>
<thead>
<tr>
<th>Issue</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clusters with nodes using m3.xlarge or c3.4xlarge instance types fail to launch</td>
<td>Resolved in AMI 2.4.0</td>
</tr>
</tbody>
</table>

## Issues with AMI 2.2.4

<table>
<thead>
<tr>
<th>Issue</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clusters with nodes using m3.xlarge or c3.4xlarge instance types fail to launch</td>
<td>Resolved in AMI 2.4.0</td>
</tr>
</tbody>
</table>

## Troubleshoot a Failed Cluster

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

This section walks you through the process of troubleshooting a cluster that has failed. This means that the cluster terminated with an error code. If the cluster is still running, but is taking a long time to return results, see Troubleshoot a Slow Cluster (p. 513) instead.

### Topics
- Step 1: Gather Data About the Issue (p. 510)
- Step 2: Check the Environment (p. 510)
- Step 3: Look at the Last State Change (p. 511)
- Step 4: Examine the Log Files (p. 512)
- Step 5: Test the Cluster Step by Step (p. 513)
Step 1: Gather Data About the Issue

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

The first step in troubleshooting an cluster is to gather information about what went wrong and the current status and configuration of the cluster. This information will be used in the following steps to confirm or rule out possible causes of the issue.

Define the Problem

A clear definition of the problem is the first place to begin. Some questions to ask yourself:

- What did I expect to happen? What happened instead?
- When did this problem first occur? How often has it happened since?
- Has anything changed in how I configure or run my cluster?

Cluster Details

The following cluster details are useful in helping track down issues. For more information on how to gather this information, see View Cluster Status and Details (p. 416).

- Identifier of the cluster. (Also called a job flow identifier.)
- Region and availability zone the cluster was launched into.
- State of the cluster, including details of the last state change.
- AMI version used to launch the cluster. If the version is listed as "latest", you can map this to a version number at AMI Versions Supported in Amazon EMR Versions 2.x and 3.x (p. 73).
- Type and number of EC2 instances specified for the master, core, and task nodes.

Step 2: Check the Environment

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Amazon EMR operates as part of an ecosystem of web services and open-source software. Things that affect those dependencies can impact the performance of Amazon EMR.

Topics

- Check for Service Outages (p. 510)
- Check Usage Limits (p. 511)
- Check the Release Version (p. 511)
- Check the Amazon VPC Subnet Configuration (p. 511)

Check for Service Outages

Amazon EMR uses several Amazon Web Services internally. It runs virtual servers on Amazon EC2, stores data and scripts on Amazon S3, indexes log files in Amazon SimpleDB, and reports metrics to CloudWatch. Events that disrupt these services are rare — but when they occur — can cause issues in Amazon EMR.
Before you go further, check the Service Health Dashboard. Check the region where you launched your cluster to see whether there are disruption events in any of these services.

**Check Usage Limits**

If you are launching a large cluster, have launched many clusters simultaneously, or you are an IAM user sharing an AWS account with other users, the cluster may have failed because you exceeded an AWS service limit.

Amazon EC2 limits the number of virtual server instances running on a single AWS region to 20 on-demand or reserved instances. If you launch a cluster with more than 20 nodes, or launch a cluster that causes the total number of EC2 instances active on your AWS account to exceed 20, the cluster will not be able to launch all of the EC2 instances it requires and may fail. When this happens, Amazon EMR returns an EC2 QUOTA EXCEEDED error. You can request that AWS increase the number of EC2 instances that you can run on your account by submitting a Request to Increase Amazon EC2 Instance Limit application.

Another thing that may cause you to exceed your usage limits is the delay between when a cluster is terminated and when it releases all of its resources. Depending on its configuration, it may take up to 5-20 minutes for a cluster to fully terminate and release allocated resources. If you are getting an EC2 QUOTA EXCEEDED error when you attempt to launch a cluster, it may be because resources from a recently terminated cluster may not yet have been released. In this case, you can either request that your Amazon EC2 quota be increased, or you can wait twenty minutes and re-launch the cluster.

Amazon S3 limits the number of buckets created on an account to 100. If your cluster creates a new bucket that exceeds this limit, the bucket creation will fail and may cause the cluster to fail.

**Check the Release Version**

Compare the Amazon machine image (AMI) that you used to launch the cluster with the latest Amazon EMR AMI version. Each release of the Amazon EMR AMI includes improvements such as new features, patches, and bug fixes. The issue that is affecting your cluster may have already been fixed in the latest AMI version. If possible, re-run your cluster using the latest AMI version. For more information about the AMI versions supported by Amazon EMR and the changes made in each version, see AMI Versions Supported in Amazon EMR Versions 2.x and 3.x (p. 73).

**Check the Amazon VPC Subnet Configuration**

If your cluster was launched in a Amazon VPC subnet, the subnet needs to be configured as described in Plan and Configure Networking (p. 145). 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**

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

The last state change provides information about what occurred the last time the cluster changed state. This often has information that can tell you what went wrong as a cluster changes state to FAILED. For example, if you launch a streaming cluster and specify an output location that already exists in Amazon S3, the cluster will fail with a last state change of “Streaming output directory already exists”.

You can locate the last state change value from the console by viewing the details pane for the cluster, from the CLI using the list-steps or describe-cluster arguments, or from the API using the DescribeCluster and ListSteps actions. For more information, see View Cluster Status and Details (p. 416).
Step 4: Examine the Log Files

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

The next step is to examine the log files in order to locate an error code or other indication of the issue that your cluster experienced. For information on the log files available, where to find them, and how to view them, see View Log Files (p. 422).

It may take some investigative work to determine what happened. Hadoop runs the work of the jobs in task attempts on various nodes in the cluster. Amazon EMR can initiate speculative task attempts, terminating the other task attempts that do not complete first. This generates significant activity that is logged to the controller, stderr and syslog log files as it happens. In addition, multiple tasks attempts are running simultaneously, but a log file can only display results linearly.

Start by checking the bootstrap action logs for errors or unexpected configuration changes during the launch of the cluster. From there, look in the step logs to identify Hadoop jobs launched as part of a step with errors. Examine the Hadoop job logs to identify the failed task attempts. The task attempt log will contain details about what caused a task attempt to fail.

The following sections describe how to use the various log files to identify error in your cluster.

Check the Bootstrap Action Logs

Bootstrap actions run scripts on the cluster as it is launched. They are commonly used to install additional software on the cluster or to alter configuration settings from the default values. Checking these logs may provide insight into errors that occurred during set up of the cluster as well as configuration settings changes that could affect performance.

Check the Step Logs

There are four types of step logs.

- **controller**—Contains files generated by Amazon EMR (Amazon EMR) that arise from errors encountered while trying to run your step. If your step fails while loading, you can find the stack trace in this log. Errors loading or accessing your application are often described here, as are missing mapper file errors.

- **stderr**—Contains error messages that occurred while processing the step. Application loading errors are often described here. This log sometimes contains a stack trace.

- **stdout**—Contains status generated by your mapper and reducer executables. Application loading errors are often described here. This log sometimes contains application error messages.

- **syslog**—Contains logs from non-Amazon software, such as Apache and Hadoop. Streaming errors are often described here.

Check stderr for obvious errors. If stderr displays a short list of errors, the step came to a quick stop with an error thrown. This is most often caused by an error in the mapper and reducer applications being run in the cluster.

Examine the last lines of controller and syslog for notices of errors or failures. Follow any notices about failed tasks, particularly if it says "Job Failed".

Check the Task Attempt Logs

If the previous analysis of the step logs turned up one or more failed tasks, investigate the logs of the corresponding task attempts for more detailed error information.
Step 5: Test the Cluster Step by Step

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

A useful technique when you are trying to track down the source of an error is to restart the cluster and submit the steps to it one by one. This lets you check the results of each step before processing the next one, and gives you the opportunity to correct and re-run a step that has failed. This also has the advantage that you only load your input data once.

To test a cluster step by step

1. Launch a new cluster, with both keep alive and termination protection enabled. Keep alive keeps the cluster running after it has processed all of its pending steps. Termination protection prevents a cluster from shutting down in the event of an error. For more information, see Configure a Cluster to be Transient or Long-Running (p. 67) and Managing Cluster Termination (p. 474).
2. Submit a step to the cluster. For more information, see Submit Work to a Cluster (p. 493).
3. When the step completes processing, check for errors in the step log files. For more information, see Step 4: Examine the Log Files (p. 512). The fastest way to locate these log files is by connecting to the master node and viewing the log files there. The step log files do not appear until the step runs for some time, finishes, or fails.
4. If the step succeeded without error, run the next step. If there were errors, investigate the error in the log files. If it was an error in your code, make the correction and re-run the step. Continue until all steps run without error.
5. When you are done debugging the cluster, and want to terminate it, you will have to manually terminate it. This is necessary because the cluster was launched with termination protection enabled. For more information, see Managing Cluster Termination (p. 474).

Troubleshoot a Slow Cluster

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

This section walks you through the process of troubleshooting a cluster that is still running, but is taking a long time to return results. For more information about what to do if the cluster has terminated with an error code, see Troubleshoot a Failed Cluster (p. 509).

Amazon EMR enables you to specify the number and kind of instances in the cluster. These specifications are the primary means of affecting the speed with which your data processing completes. One thing you might consider is re-running the cluster, this time specifying EC2 instances with greater resources, or specifying a larger number of instances in the cluster. For more information, see Configure Cluster Hardware and Networking (p. 139).

The following topics walk you through the process of identifying alternative causes of a slow cluster.

Topics
- Step 1: Gather Data About the Issue (p. 514)
- Step 2: Check the Environment (p. 514)
- Step 3: Examine the Log Files (p. 515)
- Step 4: Check Cluster and Instance Health (p. 517)
- Step 5: Check for Arrested Groups (p. 518)
Step 1: Gather Data About the Issue

The first step in troubleshooting an Amazon EMR cluster is to gather information about what went wrong and the current status and configuration of the cluster. This information will be used in the following steps to confirm or rule out possible causes of the issue.

Define the Problem

A clear definition of the problem is the first place to begin. Some questions to ask yourself:

- What did I expect to happen? What happened instead?
- When did this problem first occur? How often has it happened since?
- Has anything changed in how I configure or run my cluster?

Cluster Details

The following cluster details are useful in helping track down issues. For more information on how to gather this information, see View Cluster Status and Details (p. 416).

- Identifier of the cluster. (Also called a job flow identifier.)
- Region and availability zone the cluster was launched into.
- State of the cluster, including details of the last state change.
- AMI version used to launch the cluster. If the version is listed as "latest", you can map this to a version number at AMI Versions Supported in Amazon EMR Versions 2.x and 3.x (p. 73).
- Type and number of EC2 instances specified for the master, core, and task nodes.

Step 2: Check the Environment

Check for Service Outages

Amazon EMR uses several Amazon Web Services internally. It runs virtual servers on Amazon EC2, stores data and scripts on Amazon S3, indexes log files in Amazon SimpleDB, and reports metrics to
CloudWatch. Events that disrupt these services are rare — but when they occur — can cause issues in Amazon EMR.

Before you go further, check the Service Health Dashboard. Check the region where you launched your cluster to see whether there are disruption events in any of these services.

**Check Usage Limits**

If you are launching a large cluster, have launched many clusters simultaneously, or you are an IAM user sharing an AWS account with other users, the cluster may have failed because you exceeded an AWS service limit.

Amazon EC2 limits the number of virtual server instances running on a single AWS region to 20 on-demand or reserved instances. If you launch a cluster with more than 20 nodes, or launch a cluster that causes the total number of EC2 instances active on your AWS account to exceed 20, the cluster will not be able to launch all of the EC2 instances it requires and may fail. When this happens, Amazon EMR returns an **EC2 QUOTA EXCEEDED** error. You can request that AWS increase the number of EC2 instances that you can run on your account by submitting a Request to Increase Amazon EC2 Instance Limit application.

Another thing that may cause you to exceed your usage limits is the delay between when a cluster is terminated and when it releases all of its resources. Depending on its configuration, it may take up to 5-20 minutes for a cluster to fully terminate and release allocated resources. If you are getting an **EC2 QUOTA EXCEEDED** error when you attempt to launch a cluster, it may be because resources from a recently terminated cluster may not yet have been released. In this case, you can either request that your Amazon EC2 quota be increased, or you can wait twenty minutes and re-launch the cluster.

Amazon S3 limits the number of buckets created on an account to 100. If your cluster creates a new bucket that exceeds this limit, the bucket creation will fail and may cause the cluster to fail.

**Check the AMI Version**

Compare the Amazon machine image (AMI) that you used to launch the cluster with the latest Amazon EMR AMI version. Each release of the Amazon EMR AMI includes improvements such as new features, patches, and bug fixes. The issue that is affecting your cluster may have already been fixed in the latest AMI version. If possible, re-run your cluster using the latest AMI version. For more information about the AMI versions supported by Amazon EMR and the changes made in each version, see AMI Versions Supported in Amazon EMR Versions 2.x and 3.x (p. 73).

**Check the Amazon VPC Subnet Configuration**

If your cluster was launched in a Amazon VPC subnet, the subnet needs to be configured as described in Plan and Configure Networking (p. 145). 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**

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.
The next step is to examine the log files in order to locate an error code or other indication of the issue that your cluster experienced. For information on the log files available, where to find them, and how to view them, see View Log Files (p. 422).

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 /mnt/var/log/hadoop/ on each node or under daemons/ in log files archived to Amazon S3.
You can use the JobTracker logs to map a failed task attempt to the node it was run on. Once you know the node associated with the task attempt, you can check the health of the EC2 instance hosting that node to see if there were any issues such as running out of CPU or memory.

**Step 4: Check Cluster and Instance Health**

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

An Amazon EMR cluster is made up of nodes running on Amazon EC2 instances. If those instances become resource-bound (such as running out of CPU or memory), experience network connectivity issues, or are terminated, the speed of cluster processing suffers.

There are up to three types of nodes in a cluster:

- **master node** — manages the cluster. If it experiences a performance issue, the entire cluster is affected.
- **core nodes** — process map-reduce tasks and maintain the Hadoop Distributed Filesystem (HDFS). If one of these nodes experiences a performance issue, it can slow down HDFS operations as well as map-reduce processing. You can add additional core nodes to a cluster to improve performance, but cannot remove core nodes. For more information, see Manually Resizing a Running Cluster (p. 486).
- **task nodes** — process map-reduce tasks. These are purely computational resources and do not store data. You can add task nodes to a cluster to speed up performance, or remove task nodes that are not needed. For more information, see Manually Resizing a Running Cluster (p. 486).

When you look at the health of a cluster, you should look at both the performance of the cluster overall, as well as the performance of individual instances. There are several tools you can use:

**Check Cluster Health with CloudWatch**

Every Amazon EMR cluster reports metrics to CloudWatch. These metrics provide summary performance information about the cluster, such as the total load, HDFS utilization, running tasks, remaining tasks, corrupt blocks, and more. Looking at the CloudWatch metrics gives you the big picture about what is going on with your cluster and can provide insight into what is causing the slow down in processing. In addition to using CloudWatch to analyze an existing performance issue, you can set alarms that cause CloudWatch to alert if a future performance issue occurs. For more information, see Monitor Metrics with CloudWatch (p. 434).

**Check Cluster and Instance Health with Ganglia**

The Ganglia open source project 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 nodes in the cluster. For more information about the Ganglia open-source project, go to http://ganglia.info/. To have Ganglia available on your cluster, you have to install it using a bootstrap action when you launch the cluster. For more information about using Ganglia with Amazon EMR, see Monitor Performance with Ganglia (p. 450).

**Check Job Status and HDFS Health**

Use the Application history on the cluster details page to view YARN application details. For certain applications, you can drill into further detail and access logs directly. This is particularly useful for Spark applications. For more information, see View Application History (p. 421).

Hadoop provides a series of web interfaces you can use to view information. For more information about how to access these web interfaces, see View Web Interfaces Hosted on Amazon EMR Clusters (p. 461).
Step 5: Check for Arrested Groups

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

An instance group becomes arrested when it encounters too many errors while trying to launch nodes. For example, if new nodes repeatedly fail while performing bootstrap actions, the instance group will — after some time — go into the ARRESTED state rather than continuously attempt to provision new nodes.

A node could fail to come up if:

- Hadoop or the cluster is somehow broken and does not accept a new node into the cluster
- A bootstrap action fails on the new node
- The node is not functioning correctly and fails to check in with Hadoop

If an instance group is in the ARRESTED state, and the cluster is in a WAITING state, you can add a cluster step to reset the desired number of slave nodes. Adding the step resumes processing of the cluster and puts the instance group back into a RUNNING state.

For more information about how to reset a cluster in an arrested state, see Arrested State (p. 487).

Step 6: Review Configuration Settings

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

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. For more information, see Hadoop Configuration Reference (p. 543). You can modify the configuration settings from the
default values using a bootstrap action or by specifying new values in job execution parameters. For more information, see (Optional) Create Bootstrap Actions to Install Additional Software (p. 129) and Configuration of hadoop-user-env.sh (p. 547). 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/.

<table>
<thead>
<tr>
<th>Configuration Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dfs.replication</td>
<td>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.</td>
</tr>
<tr>
<td>io.sort.mb</td>
<td>Total memory available for sorting. This value should be 10x io.sort.factor. This setting can also be used to calculate total memory used by task node by figuring io.sort.mb multiplied by mapred.tasktracker.ap.tasks.maximum.</td>
</tr>
<tr>
<td>io.sort.spill.percent</td>
<td>Used during sort, at which point the disk will start to be used because the allotted sort memory is getting full.</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>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.</td>
</tr>
<tr>
<td>mapred.map.child.java.opts</td>
<td>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.</td>
</tr>
<tr>
<td>mapred.map.tasks.speculative.execution</td>
<td>Determines whether map task attempts of the same task may be launched in parallel.</td>
</tr>
<tr>
<td>mapred.reduce.tasks.speculative.execution</td>
<td>Determines whether reduce task attempts of the same task may be launched in parallel.</td>
</tr>
<tr>
<td>mapred.map.max.attempts</td>
<td>The maximum number of times a map task can be attempted. If all fail, then the map task is marked as failed.</td>
</tr>
<tr>
<td>mapred.reduce.child.java.opts</td>
<td>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.</td>
</tr>
<tr>
<td>mapred.reduce.max.attempts</td>
<td>The maximum number of times a reduce task can be attempted. If all fail, then the map task is marked as failed.</td>
</tr>
<tr>
<td>mapred.reduce.slowstart.completed.maps</td>
<td>The amount of map tasks that should complete before reduce tasks are attempted. Not waiting long enough may cause “Too many fetch-failure” errors in attempts.</td>
</tr>
</tbody>
</table>
### Step 7: Examine Input Data

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Look at your input data. Is it distributed evenly among your key values? If your data is heavily skewed towards one or few key values, the processing load may be mapped to a small number of nodes, while other nodes idle. This imbalanced distribution of work can result in slower processing times.

An example of an imbalanced data set would be running a cluster to alphabetize words, but having a data set that contained only words beginning with the letter "a". When the work was mapped out, the node processing values beginning with "a" would be overwhelmed, while nodes processing words beginning with other letters would go idle.

### Common Errors in Amazon EMR

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

There are many reasons why a cluster might fail or be slow in processing data. The following sections list the most common issues and suggestions for fixing them.

**Topics**

- Input and Output Errors (p. 521)
- Permissions Errors (p. 522)
- Memory Errors (p. 523)
- Resource Errors (p. 524)
- Streaming Cluster Errors (p. 528)
- Custom JAR Cluster Errors (p. 529)
- Hive Cluster Errors (p. 529)
- VPC Errors (p. 531)
- AWS GovCloud (US) Errors (p. 532)
- Other Issues (p. 533)
Input and Output Errors

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

The following errors are common in cluster input and output operations.

Topics
- Does your path to Amazon Simple Storage Service (Amazon S3) have at least three slashes? (p. 521)
- Are you trying to recursively traverse input directories? (p. 521)
- Does your output directory already exist? (p. 521)
- Are you trying to specify a resource using an HTTP URL? (p. 521)
- Are you referencing an Amazon S3 bucket using an invalid name format? (p. 521)
- Are you experiencing trouble loading data to or from Amazon S3? (p. 522)

Does your path to Amazon Simple Storage Service (Amazon S3) have at least three slashes?

When you specify an Amazon S3 bucket, you must include a terminating slash on the end of the URL. For example, instead of referencing a bucket as “s3n://myawsbucket”, you should use “s3n://myawsbucket/”, otherwise Hadoop fails your cluster in most cases.

Are you trying to recursively traverse input directories?

Hadoop does not recursively search input directories for files. If you have a directory structure such as /corpus/01/01.txt, /corpus/01/02.txt, /corpus/02/01.txt, etc. and you specify /corpus/ as the input parameter to your cluster, Hadoop does not find any input files because the /corpus/ directory is empty and Hadoop does not check the contents of the subdirectories. Similarly, Hadoop does not recursively check the subdirectories of Amazon S3 buckets.

The input files must be directly in the input directory or Amazon S3 bucket that you specify, not in subdirectories.

Does your output directory already exist?

If you specify an output path that already exists, Hadoop will fail the cluster in most cases. This means that if you run a cluster one time and then run it again with exactly the same parameters, it will likely work the first time and then never again; after the first run, the output path exists and thus causes all successive runs to fail.

Are you trying to specify a resource using an HTTP URL?

Hadoop does not accept resource locations specified using the http:// prefix. You cannot reference a resource using an HTTP URL. For example, passing in http://mysite/myjar.jar as the JAR parameter causes the cluster to fail. For more information about how to reference files in Amazon EMR, see Work with Storage and File Systems (p. 29).

Are you referencing an Amazon S3 bucket using an invalid name format?

If you attempt to use a bucket name such as “myawsbucket.1” with Amazon EMR, your cluster will fail because Amazon EMR requires that bucket names be valid RFC 2396 host names; the name cannot end
with a number. In addition, because of the requirements of Hadoop, Amazon S3 bucket names used with Amazon EMR must contain only lowercase letters, numbers, periods (.), and hyphens (-). For more information about how to format Amazon S3 bucket names, see Bucket Restrictions and Limitations in the Amazon Simple Storage Service Developer Guide.

Are you experiencing trouble loading data to or from Amazon S3?

Amazon S3 is the most popular input and output source for Amazon EMR. A common mistake is to treat Amazon S3 as you would a typical file system. There are differences between Amazon S3 and a file system that you need to take into account when running your cluster.

- If an internal error occurs in Amazon S3, your application needs to handle this gracefully and re-try the operation.
- If calls to Amazon S3 take too long to return, your application may need to reduce the frequency at which it calls Amazon S3.
- Listing all the objects in an Amazon S3 bucket is an expensive call. Your application should minimize the number of times it does this.

There are several ways you can improve how your cluster interacts with Amazon S3.

- Launch your cluster using the latest AMI. This version has the latest improvements to how Amazon EMR accesses Amazon S3. For more information, see Choose an Amazon Machine Image (AMI) (p. 69).
- Use S3DistCp to move objects in and out of Amazon S3. S3DistCp implements error handling, retries and back-offs to match the requirements of Amazon S3. For more information, see Distributed Copy Using S3DistCp.
- Design your application with eventual consistency in mind. Use HDFS for intermediate data storage while the cluster is running and Amazon S3 only to input the initial data and output the final results.
- If your clusters will commit 200 or more transactions per second to Amazon S3, contact support to prepare your bucket for greater transactions per second and consider using the key partition strategies described in Amazon S3 Performance Tips & Tricks.
- Set the Hadoop configuration setting io.file.buffer.size to 65536. This causes Hadoop to spend less time seeking through Amazon S3 objects.
- Consider disabling Hadoop's speculative execution feature if your cluster is experiencing Amazon S3 concurrency issues. You do this through the mapred.map.tasks.speculative.execution and mapred.reduce.tasks.speculative.execution configuration settings. This is also useful when you are troubleshooting a slow cluster.
- If you are running a Hive cluster, see Are you having trouble loading data to or from Amazon S3 into Hive? (p. 530).

For additional information, see Amazon S3 Error Best Practices in the Amazon Simple Storage Service Developer Guide.

Permissions Errors

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

The following errors are common when using permissions or credentials.

Topics

- Are you passing the correct credentials into SSH? (p. 523)
• If you are using IAM, do you have the proper Amazon EC2 policies set? (p. 523)

### Are you passing the correct credentials into SSH?

If you are unable to use SSH to connect to the master node, it is most likely an issue with your security credentials.

First, check that the .pem file containing your SSH key has the proper permissions. You can use chmod to change the permissions on your .pem file as is shown in the following example, where you would replace mykey.pem with the name of your own .pem file.

```
chmod og-rwx mykey.pem
```

The second possibility is that you are not using the keypair you specified when you created the cluster. This is easy to do if you have created multiple key pairs. Check the cluster details in the Amazon EMR console (or use the `--describe` option in the CLI) for the name of the keypair that was specified when the cluster was created.

After you have verified that you are using the correct key pair and that permissions are set correctly on the .pem file, you can use the following command to use SSH to connect to the master node, where you would replace mykey.pem with the name of your .pem file and hadoop@ec2-01-001-001-1.compute-1.amazonaws.com with the public DNS name of the master node (available through the `--describe` option in the CLI or through the Amazon EMR console.)

**Important**

You must use the login name `hadoop` when you connect to an Amazon EMR cluster node, otherwise an error similar to `Server refused our key` error may occur.

```
ssh -i mykey.pem hadoop@ec2-01-001-001-1.compute-1.amazonaws.com
```

For more information, see [Connect to the Master Node Using SSH](#) (p. 457).

### If you are using IAM, do you have the proper Amazon EC2 policies set?

Because Amazon EMR uses EC2 instances as nodes, IAM users of Amazon EMR also need to have certain Amazon EC2 policies set in order for Amazon EMR to be able to manage those instances on the IAM user's behalf. If you do not have the required permissions set, Amazon EMR returns the error: "User account is not authorized to call EC2."

For more information about the Amazon EC2 policies your IAM account needs to set to run Amazon EMR, see [Amazon EMR Actions in User-Based IAM Policies](#) (p. 188).

### Memory Errors

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Memory tuning issues may appear as one or more of the following symptoms, depending on the circumstances:
• The job gets stuck in mapper or reducer phase.
• The job hangs for long periods of time or finishes much later than expected.
• The job completely fails and or terminates.

These symptoms are typical when the master instance type or slave nodes run out of memory.

Limit the number of simultaneous map/reduce tasks

• In Amazon EMR, a JVM runs for each slot and thus a high number of map slots require a large amount of memory. The following bootstrap actions control the number of maps/reduces that run simultaneously on a TaskTracker and as a result, control the overall amount of memory consumed:

```
--args -m,mapred.tasktracker.map.tasks.maximum=maximum number of simultaneous map tasks
--args -m,mapred.tasktracker.reduce.tasks.maximum=maximum number of simultaneous reduce tasks
```

Increase the JVM heap size and task timeout

• Avoid memory issues by tuning the JVM heap size for Hadoop processes, because even large instance size such as m1.xlarge can run out of memory using the default settings. For example, an instance type of m1.xlarge assigns 1GB of memory per task by default. However, if you decrease tasks to one per node, increase the memory allocated to the heap with the following bootstrap action:

```
s3://elasticmapreduce/bootstrap-actions/configure-hadoop --args -m,mapred.child.java.opts=-Xmxamount of memory in MB
```

Even after tuning the heap size, it can be useful to increase the `mapred.task.timeout` setting to make Hadoop wait longer before it begins terminating processes.

Increase NameNode heap size

• There are times when the NameNode can run out of memory as well. The NameNode keeps all of the metadata for the HDFS file system in memory. The larger the HDFS file system is, blocks and files, the more memory that will be needed by the NameNode. Also, if DataNodes availability is not consistent the NameNode will begin to queue up blocks that are in invalid states. It is rare that NameNodes will get out of memory errors but is something to watch for if you have a very large HDFS file system, or have an HDFS directory with a large number of entries. The NameNode JVM heap size can be increased with the following Bootstrap Action:

```
--bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-daemons --args --namenode-heap-size=amount of memory in MB
```

Resource Errors

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

The following errors are commonly caused by constrained resources on the cluster.

Topics
Do you have enough HDFS space for your cluster?

If you do not, Amazon EMR returns the following error: "Cannot replicate block, only managed to replicate to zero nodes." This error occurs when you generate more data in your cluster than can be stored in HDFS. You will see this error only while the cluster is running, because when the cluster ends it releases the HDFS space it was using.

The amount of HDFS space available to a cluster depends on the number and type of Amazon EC2 instances that are used as core nodes. All of the disk space on each Amazon EC2 instance is available to be used by HDFS. For more information about the amount of local storage for each EC2 instance type, see Instance Types and Families in the Amazon EC2 User Guide for Linux Instances.

The other factor that can affect the amount of HDFS space available is the replication factor, which is the number of copies of each data block that are stored in HDFS for redundancy. The replication factor increases with the number of nodes in the cluster: there are 3 copies of each data block for a cluster with 10 or more nodes, 2 copies of each block for a cluster with 4 to 9 nodes, and 1 copy (no redundancy) for clusters with 3 or fewer nodes. The total HDFS space available is divided by the replication factor. In some cases, such as increasing the number of nodes from 9 to 10, the increase in replication factor can actually cause the amount of available HDFS space to decrease.

For example, a cluster with ten core nodes of type m1.large would have 2833 GB of space available to HDFS ((10 nodes X 850 GB per node)/replication factor of 3).

If your cluster exceeds the amount of space available to HDFS, you can add additional core nodes to your cluster or use data compression to create more HDFS space. If your cluster is one that can be stopped and restarted, you may consider using core nodes of a larger Amazon EC2 instance type. You might also consider adjusting the replication factor. Be aware, though, that decreasing the replication factor reduces the redundancy of HDFS data and your cluster's ability to recover from lost or corrupted HDFS blocks.

Are you seeing "EC2 Quota Exceeded" errors?

If you are getting messages that you are exceeding your Amazon EC2 instance quota, this may be for one of several reasons. Depending on configuration differences, it may take up to 5-20 minutes for previous clusters to terminate and release allocated resources. If you are getting an EC2 QUOTA EXCEEDED error when you attempt to launch a cluster, it may be because resources from a recently terminated cluster have not yet been released. Furthermore, if you attempt to resize an instance group, you may also encounter this error when your new target size is greater than the current instance quota for the account. In these cases, you can either terminate and launch the cluster with a smaller target size. In all cases, you can terminate unused cluster resources or EC2 instances, request that your Amazon EC2 quota be increased, or wait to re-launch a cluster.

Note
You cannot issue more than one resize request to the same cluster. Therefore, if your first request fails, you will have to potentially terminate your current cluster and launch another cluster to with the number of instances desired.

Are you seeing "Too many fetch-failures" errors?

The presence of "Too many fetch-failures" or "Error reading task output" error messages in step or task attempt logs indicates the running task is dependent on the output of another task. This often occurs
when a reduce task is queued to execute and requires the output of one or more map tasks and the output is not yet available.

There are several reasons the output may not be available:

- The prerequisite task is still processing. This is often a map task.
- The data may be unavailable due to poor network connectivity if the data is located on a different instance.
- If HDFS is used to retrieve the output, there may be an issue with HDFS.

The most common cause of this error is that the previous task is still processing. This is especially likely if the errors are occurring when the reduce tasks are first trying to run. You can check whether this is the case by reviewing the syslog log for the cluster step that is returning the error. If the syslog shows both map and reduce tasks making progress, this indicates that the reduce phase has started while there are map tasks that have not yet completed.

One thing to look for in the logs is a map progress percentage that goes to 100% and then drops back to a lower value. When the map percentage is at 100%, this does not mean that all map tasks are completed. It simply means that Hadoop is executing all the map tasks. If this value drops back below 100%, it means that a map task has failed and, depending on the configuration, Hadoop may try to reschedule the task. If the map percentage stays at 100% in the logs, look at the CloudWatch metrics, specifically RunningMapTasks, to check whether the map task is still processing. You can also find this information using the Hadoop web interface on the master node.

If you are seeing this issue, there are several things you can try:

- Instruct the reduce phase to wait longer before starting. You can do this by altering the Hadoop configuration setting mapred.reduce.slowstart.completed.maps to a longer time. For more information, see (Optional) Create Bootstrap Actions to Install Additional Software (p. 129).
- Match the reducer count to the total reducer capability of the cluster. You do this by adjusting the Hadoop configuration setting mapred.reduce.tasks for the job.
- Use a combiner class code to minimize the amount of outputs that need to be fetched.
- Check that there are no issues with the Amazon EC2 service that are affecting the network performance of the cluster. You can do this using the Service Health Dashboard.
- Review the CPU and memory resources of the instances in your cluster to make sure that your data processing is not overwhelming the resources of your nodes. For more information, see Configure Cluster Hardware and Networking (p. 139).
- Check the version of the Amazon Machine Image (AMI) used in your Amazon EMR cluster. If the version is 2.3.0 through 2.4.4 inclusive, update to a later version. AMI versions in the specified range use a version of Jetty that may fail to deliver output from the map phase. The fetch error occurs when the reducers cannot obtain output from the map phase.

Jetty is an open-source HTTP server that is used for machine to machine communications within a Hadoop cluster.

Are you seeing "File could only be replicated to 0 nodes instead of 1" errors?

When a file is written to HDFS, it is replicated to multiple core nodes. When you see this error, it means that the NameNode daemon does not have any available DataNode instances to write data to in HDFS. In other words, block replication is not taking place. This error can be caused by a number of issues:

- The HDFS filesystem may have run out of space. This is the most likely cause.
- DataNode instances may not have been available when the job was run.
• DataNode instances may have been blocked from communication with the master node.
• Instances in the core instance group might not be available.
• Permissions may be missing. For example, the JobTracker daemon may not have permissions to create job tracker information.
• The reserved space setting for a DataNode instance may be insufficient. Check whether this is the case by checking the dfs.datanode.du.reserved configuration setting.

To check whether this issue is caused by HDFS running out of disk space, look at the HDFSUtilization metric in CloudWatch. If this value is too high, you can add additional core nodes to the cluster. Keep in mind that you can only add core nodes to a cluster, you cannot remove them. If you have a cluster that you think might run out of HDFS disk space, you can set an alarm in CloudWatch to alert you when the value of HDFSUtilization rises above a certain level. For more information, see Manually Resizing a Running Cluster (p. 486) and Monitor Metrics with CloudWatch (p. 434).

If HDFS running out of space was not the issue, check the DataNode logs, the NameNode logs and network connectivity for other issues that could have prevented HDFS from replicating data. For more information, see View Log Files (p. 422).

Are your TaskTracker nodes being blacklisted?

A TaskTracker node is a node in the cluster that accepts map and reduce tasks. These are assigned by a JobTracker daemon. The JobTracker monitors the TaskTracker node through a heartbeat.

There are a couple of situations in which the JobTracker daemon blacklists a TaskTracker node, removing it from the pool of nodes available to process tasks:

• If the TaskTracker node has not sent a heartbeat to the JobTracker daemon in the past 10 minutes (60000 milliseconds). This time period can be configured using the mapred.tasktracker.expiry.interval configuration setting. For more information about changing Hadoop configuration settings, see (Optional) Create Bootstrap Actions to Install Additional Software (p. 129).

• If the TaskTracker node has more than 4 failed tasks. You can change this to a higher value using the modify mapred.max.tracker.failures configuration parameter. Other configuration settings you might want to change are the settings that control how many times to attempt a task before marking it as failed: mapred.map.max.attempts for map tasks and mapreduce.reduce.maxattempts for reduce tasks. For more information about changing Hadoop configuration settings, see (Optional) Create Bootstrap Actions to Install Additional Software (p. 129).

You can use the CloudWatch CLI to view the number of blacklisted TaskTracker nodes. The command for doing so is shown in the following example. For more information, see the Amazon CloudWatch CLI Reference.

```
mon-get-stats NoOfBlackListedTaskTrackers --dimensions JobFlowId=JobFlowID --statistics Maximum --namespace AWS/ElasticMapReduce
```

The following example shows how to launch a cluster and use a bootstrap action to set the value of mapred.max.tracker.failures to 7, instead of the default 4.

Type the following command using the AWS CLI and replace myKey with the name of your EC2 key pair.

```
aws emr create-cluster --name "Test cluster" --ami-version 2.4 --applications Name=Hive Name=Pig 
--use-default-roles --ec2-attributes KeyName=myKey 
```
Note
If you have not previously created the default EMR service role and EC2 instance profile, type
`aws emr create-default-roles` to create them before typing the `create-cluster` subcommand.

When you launch a cluster using the preceding example, you can connect to the master node and see
the changed configuration setting in `/home/hadoop/conf/mapred-site.xml`. The modified line will
appear as shown in the following example.

```xml
<property>
  <name>mapred.max.tracker.failures</name>
  <value>7</value>
</property>
```

Streaming Cluster Errors

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

You can usually find the cause of a streaming error in a `syslog` file. Link to it on the **Steps** pane.

The following errors are common to streaming clusters.

**Topics**

- Is data being sent to the mapper in the wrong format? (p. 528)
- Is your script timing out? (p. 528)
- Are you passing in invalid streaming arguments? (p. 528)
- Did your script exit with an error? (p. 529)

**Is data being sent to the mapper in the wrong format?**

To check if this is the case, look for an error message in the `syslog` file of a failed task attempt in the
task attempt logs. For more information, see View Log Files (p. 422).

**Is your script timing out?**

The default timeout for a mapper or reducer script is 600 seconds. If your script takes longer than this,
the task attempt will fail. You can verify this is the case by checking the `syslog` file of a failed task
attempt in the task attempt logs. For more information, see View Log Files (p. 422).

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.
Custom JAR Cluster Errors

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

The following errors are common to custom JAR clusters.

Topics

- Is your JAR throwing an exception before creating a job? (p. 529)
- Is your JAR throwing an error inside a map task? (p. 529)

Is your JAR throwing an exception before creating a job?

If the main program of your custom JAR throws an exception while creating the Hadoop job, the best place to look is the syslog file of the step logs. For more information, see View Log Files (p. 422).

Is your JAR throwing an error inside a map task?

If your custom JAR and mapper throw an exception while processing input data, the best place to look is the syslog file of the task attempt logs. For more information, see View Log Files (p. 422).

Hive Cluster Errors

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.
You can usually find the cause of a Hive error in the syslog file, which you link to from the Steps pane. If you can't determine the problem there, check in the Hadoop task attempt error message. Link to it on the Task Attempts pane.

The following errors are common to Hive clusters.

Topics

- Are you using the latest version of Hive? (p. 530)
- Did you encounter a syntax error in the Hive script? (p. 530)
- Did a job fail when running interactively? (p. 530)
- Are you having trouble loading data to or from Amazon S3 into Hive? (p. 530)

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. For more information, see Supported Hive Versions (p. 271).

Did you encounter a syntax error in the Hive script?

If a step fails, look at the stdout file of the logs for the step that ran the Hive script. If the error is not there, look at the syslog file of the task attempt logs for the task attempt that failed. For more information, see View Log Files (p. 422).

Did a job fail when running interactively?

If you are running Hive interactively on the master node and the cluster failed, look at the syslog entries in the task attempt log for the failed task attempt. For more information, see View Log Files (p. 422).

Are you having trouble loading data to or from Amazon S3 into Hive?

If you are having trouble accessing data in Amazon S3, first check the possible causes listed in Are you experiencing trouble loading data to or from Amazon S3? (p. 522). If none of those issues is the cause, consider the following options specific to Hive.

- Make sure you are using the latest version of Hive, which has all the current patches and bug fixes that may resolve your issue. For more information, see Apache Hive.
- Using INSERT OVERWRITE requires listing the contents of the Amazon S3 bucket or folder. This is an expensive operation. If possible, manually prune the path instead of having Hive list and delete the existing objects.
- Pre-cache the results of an Amazon S3 list operation locally on the cluster. You do this in HiveQL with the following command: set hive.optimize.s3.query=true;
- Use static partitions where possible.
- In some versions of Hive and Amazon EMR, it is possible that using ALTER TABLES will fail because the table is stored in a different location than expected by Hive. The solution is to add or update following in /home/hadoop/conf/core-site.xml:

```xml
<property>
  <name>fs.s3n.endpoint</name>
  <value>s3.amazonaws.com</value>
</property>
```
VPC Errors

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

The following errors are common to VPC configuration in Amazon EMR.

Topics

- Invalid Subnet Configuration (p. 531)
- Missing DHCP Options Set (p. 531)
- Permissions Errors (p. 532)
- Errors That Result in START_FAILED (p. 532)

Invalid Subnet Configuration

On the Cluster Details page, in the Status field, you see an error similar to the following:

The subnet configuration was invalid: Cannot find route to InternetGateway in main RouteTable rtb-id for vpc vpc-id.

To solve this problem, you must create an Internet Gateway and attach it to your VPC. For more information, see Adding an Internet Gateway to Your VPC.

Alternatively, verify that you have configured your VPC with Enable DNS resolution and Enable DNS hostname support enabled. For more information, see Using DNS with Your VPC.

Missing DHCP Options Set

You see a step failure in the cluster system log (syslog) with an error similar to the following:

ERROR org.apache.hadoop.security.UserGroupInformation (main):
PrivilegedActionException as:hadoop (auth:SIMPLE) cause:java.io.IOException: org.apache.hadoop.yarn.exceptions.ApplicationNotFoundException: Application with id 'application_id' doesn't exist in RM.

or


To solve this problem, you must configure a VPC that includes a DHCP Options Set whose parameters are set to the following values:

Note
If you use the AWS GovCloud (US) region, set domain-name to us-gov-west-1.compute.internal instead of the value used in the following example.

- domain-name = ec2.internal

Use ec2.internal if your region is US East (N. Virginia). For other regions, use region-name.compute.internal. For example in us-west-2, use domain-name=us-west-2.compute.internal.
- domain-name-servers = AmazonProvidedDNS
For more information, see DHCP Options Sets.

Permissions Errors

A failure in the stderr log for a step indicates that an Amazon S3 resource does not have the appropriate permissions. This is a 403 error and the error looks like:

```
Exception in thread "main" com.amazonaws.services.s3.model.AmazonS3Exception: Access Denied
```

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[
```

Errors That Result in START_FAILED

Before AMI 3.7.0, for VPCs where a hostname is specified Amazon EC2 instances, Amazon EMR Amazon EMR maps the internal hostnames of the subnet with custom domain addresses as follows: ip-X.X.X.X.customdomain.com.tld. For example, if the hostname was ip-10.0.0.10 and the VPC has the domain name option set to customdomain.com, the resulting hostname mapped by Amazon EMR would be ip-10.0.1.0.customdomain.com. An entry is added in /etc/hosts to resolve the hostname to 10.0.0.10. This behavior is changed with AMI 3.7.0 and now Amazon EMR honors the DHCP configuration of the VPC entirely. Previously, customers could also use a bootstrap action to specify a hostname mapping.

If you would like to preserve this behavior, you must provide the DNS and forward resolution setup you require for the custom domain.

AWS GovCloud (US) Errors

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.
The AWS GovCloud (US) region differs from other regions in its security, configuration, and default settings. As a result, use the following checklist to troubleshoot Amazon EMR errors that are specific to the AWS GovCloud (US) region before using more general troubleshooting recommendations.

- Verify that your IAM roles are correctly configured. For more information, see Configure IAM Roles for Amazon EMR Permissions to AWS Services (p. 234).
- Ensure that your VPC configuration has correctly configured DNS resolution/hostname support, Internet Gateway, and DHCP Option Set parameters. For more information, see VPC Errors (p. 531).

If these steps do not solve the problem, continue with the steps for troubleshooting common Amazon EMR errors. For more information, see Common Errors in Amazon EMR (p. 520).

Other Issues

Do you not see the cluster you expect in the Cluster List page or in results returned from ListClusters API?

Check the following:

- The cluster age is less than two months. Amazon EMR preserves metadata information about completed clusters for your reference, at no charge, for two months. The console does not provide a way to delete completed clusters from the console; these are automatically removed for you after two months.
- You have permissions to view the cluster. If the VisibleToAllUsers property is set to false, other users in the same IAM account will not be able to view a cluster.
- You are viewing the correct region.
Write Applications that Launch and Manage Clusters

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

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

Topics

- End-to-End Amazon EMR Java Source Code Sample (p. 534)
- Common Concepts for API Calls (p. 538)
- Use SDKs to Call Amazon EMR APIs (p. 539)

You can access the functionality provided by the Amazon EMR API by calling wrapper functions in one of the AWS SDKs. The AWS SDKs provide language-specific functions that wrap the web service's API and simplify connecting to the web service, handling many of the connection details for you. For more information about calling Amazon EMR using one of the SDKs, see Use SDKs to Call Amazon EMR APIs (p. 539).

Important

The maximum request rate for Amazon EMR is one request every ten seconds.

End-to-End Amazon EMR Java Source Code Sample

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Developers can call the Amazon EMR API using custom Java code to do the same things possible with the Amazon EMR console or CLI. This section provides the end-to-end steps necessary to install the AWS Toolkit for Eclipse and run a fully-functional Java source code sample that adds steps to an Amazon EMR cluster.

Note

This example focuses on Java, but Amazon EMR also supports several programming languages with a collection of Amazon EMR SDKs. For more information, see Use SDKs to Call Amazon EMR APIs (p. 539).

This Java source code example demonstrates how to perform the following tasks using the Amazon EMR API:
• Retrieve AWS credentials and send them to Amazon EMR to make API calls
• Configure a new custom step and a new predefined step
• Add new steps to an existing Amazon EMR cluster
• Retrieve cluster step IDs from a running cluster

**Note**
This sample demonstrates how to add steps to an existing cluster and thus requires that you have an active cluster on your account.

Before you begin, install a version of the **Eclipse IDE for Java EE Developers** that matches your computer platform. For more information, go to **Eclipse Downloads**.

Next, install the Database Development plug-in for Eclipse.

**To install the Database Development Eclipse plug-in**

1. Open the Eclipse IDE.
2. Choose **Help** and **Install New Software**.
3. In the **Work with**: field, type `http://download.eclipse.org/releases/kepler` or the path that matches the version number of your Eclipse IDE.
4. In the items list, choose **Database Development** and **Finish**.
5. Restart Eclipse when prompted.

Next, install the Toolkit for Eclipse to make the helpful, pre-configured source code project templates available.

**To install the Toolkit for Eclipse**

1. Open the Eclipse IDE.
2. Choose **Help** and **Install New Software**.
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.

---

536
Note
In the following sample code, replace the example cluster ID (j-1HTE8WKS7SODR) with a valid cluster ID in your account found either in the AWS Management Console or by using the following AWS CLI command:

```
aws emr list-clusters --active | grep "Id"
```

In addition, replace the example Amazon S3 path (s3://mybucket/my-jar-location1) with the valid path to your JAR. Lastly, replace the example class name (com.my.Main1) with the correct name of the class in your JAR, if applicable.

```java
import java.io.IOException;
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.PropertiesCredentials;
import com.amazonaws.services.elasticmapreduce.*;
import com.amazonaws.services.elasticmapreduce.model.AddJobFlowStepsRequest;
import com.amazonaws.services.elasticmapreduce.model.AddJobFlowStepsResult;
import com.amazonaws.services.elasticmapreduce.model.HadoopJarStepConfig;
import com.amazonaws.services.elasticmapreduce.util.StepFactory;
public class main {
    public static void main(String[] args) {
        AWSCredentials credentials = null;
        try {
            credentials = new PropertiesCredentials(
                    main.class.getResourceAsStream("AwsCredentials.properties"));
        } catch (IOException e1) {
            System.out.println("Credentials were not properly entered into
                    AwsCredentials.properties.");
            System.out.println(e1.getMessage());
            System.exit(-1);
        }
        AmazonElasticMapReduce client = new AmazonElasticMapReduceClient(credentials);
        // predefined steps. See StepFactory for list of predefined steps
        StepConfig hive = new StepConfig("Hive", new StepFactory().newInstallHiveStep());
        // A custom step
        HadoopJarStepConfig hadoopConfig1 = new HadoopJarStepConfig()
                .withJar("s3://mybucket/my-jar-location1")
                .withMainClass("com.my.Main1") // optional main class, this can be omitted if
                .withArgs("--verbose"); // optional list of arguments
        StepConfig customStep = new StepConfig("Step1", hadoopConfig1);
        AddJobFlowStepsResult result = client.addJobFlowSteps(new AddJobFlowStepsRequest()
                .withJobFlowId("j-1HTE8WKS7SODR")
                .withSteps(hive, customStep));
        System.out.println(result.getStepIds());
    }
}
```

10. Choose Run, Run As, and Java Application.
11. If the sample runs correctly, a list of IDs for the new steps appears in the Eclipse IDE console window. The correct output is similar to the following:
Common Concepts for API Calls

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Topics

- Endpoints for Amazon EMR (p. 538)
- Specifying Cluster Parameters in Amazon EMR (p. 538)
- Availability Zones in Amazon EMR (p. 539)
- How to Use Additional Files and Libraries in Amazon EMR Clusters (p. 539)

When you write an application that calls the Amazon EMR API, there are several concepts that apply when calling one of the wrapper functions of an SDK.

Endpoints for Amazon EMR

An endpoint is a URL that is the entry point for a web service. Every web service request must contain an endpoint. The endpoint specifies the AWS region where clusters are created, described, or terminated. It has the form `elasticmapreduce.regionname.amazonaws.com`. If you specify the general endpoint (`elasticmapreduce.amazonaws.com`), Amazon EMR directs your request to an endpoint in the default region. For accounts created on or after March 8, 2013, the default region is us-west-2; for older accounts, the default region is us-east-1.

For more information about the endpoints for Amazon EMR, see Regions and Endpoints in the Amazon Web Services General Reference.

Specifying Cluster Parameters in Amazon EMR

The `Instances` parameters enable you to configure the type and number of EC2 instances to create nodes to process the data. Hadoop spreads the processing of the data across multiple cluster nodes. The master node is responsible for keeping track of the health of the core and task nodes and polling the nodes for job result status. The core and task nodes do the actual processing of the data. If you have a single-node cluster, the node serves as both the master and a core node.

The `KeepJobAlive` parameter in a `RunJobFlow` request determines whether to terminate the cluster when it runs out of cluster steps to execute. Set this value to `False` when you know that the cluster is running as expected. When you are troubleshooting the job flow and adding steps while the cluster execution is suspended, set the value to `True`. This reduces the amount of time and expense of uploading the results to Amazon Simple Storage Service (Amazon S3), only to repeat the process after modifying a step to restart the cluster.

If `KeepJobAlive` is `true`, after successfully getting the cluster to complete its work, you must send a `TerminateJobFlows` request or the cluster continues to run and generate AWS charges.

For more information about parameters that are unique to `RunJobFlow`, see `RunJobFlow`. For more information about the generic parameters in the request, see Common Request Parameters.
Availability Zones in Amazon EMR

Amazon EMR uses EC2 instances as nodes to process clusters. These EC2 instances have locations composed of Availability Zones and regions. Regions are dispersed and located in separate geographic areas. Availability Zones are distinct locations within a region insulated from failures in other Availability Zones. Each Availability Zone provides inexpensive, low-latency network connectivity to other Availability Zones in the same region. For a list of the regions and endpoints for Amazon EMR, see Regions and Endpoints in the Amazon Web Services General Reference.

The AvailabilityZone parameter specifies the general location of the cluster. This parameter is optional and, in general, we discourage its use. When AvailabilityZone is not specified Amazon EMR automatically picks the best AvailabilityZone value for the cluster. You might find this parameter useful if you want to colocate your instances with other existing running instances, and your cluster needs to read or write data from those instances. For more information, see the Amazon EC2 User Guide for Linux Instances.

How to Use Additional Files and Libraries in Amazon EMR Clusters

There are times when you might like to use additional files or custom libraries with your mapper or reducer applications. For example, you might like to use a library that converts a PDF file into plain text.

To cache a file for the mapper or reducer to use when using Hadoop streaming

- In the JAR args field, add the following argument:

  -cacheFile s3://bucket/path_to_executable#local_path

  The file, local_path, is in the working directory of the mapper, which could reference the file.

Use SDKs to Call Amazon EMR APIs

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Topics

- Using the AWS SDK for Java to Create an Amazon EMR Cluster (p. 540)
- Using the AWS SDK for .Net to Create an Amazon EMR Cluster (p. 541)
- Using the Java SDK to Sign an API Request (p. 542)

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

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

The AWS SDK for Java provides three packages with Amazon EMR functionality:

- com.amazonaws.services.elasticmapreduce
- com.amazonaws.services.elasticmapreduce.model
- com.amazonaws.services.elasticmapreduce.util

For more information about these packages, see the AWS SDK for Java API Reference.

The following example illustrates how the SDKs can simplify programming with Amazon EMR. The code sample below uses the StepFactory object, a helper class for creating common Amazon EMR step types, to create an interactive Hive cluster with debugging enabled.

**Note**
If you are adding IAM user visibility to a new cluster, call RunJobFlow and set VisibleToAllUsers=true, otherwise IAM users cannot view the cluster.

```java
AWSCredentials credentials = new BasicAWSCredentials(accessKey, secretKey);
AmazonElasticMapReduceClient emr = new AmazonElasticMapReduceClient(credentials);
String COMMAND_RUNNER = "command-runner.jar";
String DEBUGGING_COMMAND = "state-pusher-script";
String DEBUGGING_NAME = "Setup Hadoop Debugging";
StepFactory stepFactory = new StepFactory();
StepConfig enabledebugging = new StepConfig()
    .withName(DEBUGGING_NAME)
    .withActionOnFailure(ActionOnFailure.TERMINATE_CLUSTER)
    .withHadoopJarStep(new HadoopJarStepConfig()
        .withJar(COMMAND_RUNNER)
        .withArgs(DEBUGGING_COMMAND));
RunJobFlowRequest request = new RunJobFlowRequest()
    .withName("Hive Interactive")
    .withAmiVersion("3.8")
    .withSteps(enabledebugging, installHive)
    .withLogUri("s3://myawsbucket/")
    .withServiceRole("service_role")
    .withJobFlowRole("jobflow_role")
    .withInstances(new JobFlowInstancesConfig()
        .withEc2KeyName("keypair")
        .withInstanceCount(5)
        .withKeepJobFlowAliveWhenNoSteps(true)
        .withMasterInstanceType("m3.xlarge")
        .withSlaveInstanceType("m1.large"));
RunJobFlowResult result = emr.runJobFlow(request);
```
At minimum, you must pass a service role and jobflow role corresponding to EMR_DefaultRole and EMR_EC2_DefaultRole, respectively. You can do this by invoking this AWS CLI command for the same account. First, look to see if the roles already exist:

```
aws iam list-roles | grep EMR
```

Both the instance profile (EMR_EC2_DefaultRole) and the service role (EMR_DefaultRole) will be displayed if they exist:

```
"RoleName": "EMR_DefaultRole",
"Arn": "arn:aws:iam::AccountID:role/EMR_DefaultRole"
"RoleName": "EMR_EC2_DefaultRole",
"Arn": "arn:aws:iam::AccountID:role/EMR_EC2_DefaultRole"
```

If the default roles do not exist, you can use the following AWS CLI command to create them:

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

**Using the AWS SDK for .Net to Create an Amazon EMR Cluster**

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

The following example illustrates how the SDKs can simplify programming with Amazon EMR. The code sample below uses the StepFactory object, a helper class for creating common Amazon EMR step types, to create an interactive Hive cluster with debugging enabled.

**Note**

If you are adding IAM user visibility to a new cluster, call `RunJobFlow` and set `VisibleToAllUsers=true`, otherwise IAM users cannot view the cluster.

```
var emrClient = AWSClientFactory.CreateAmazonElasticMapReduceClient(RegionEndpoint.USWest2);
var stepFactory = new StepFactory();

var enabled debugging = new StepConfig{
    Name = "Enable debugging",
    ActionOnFailure = "TERMINATE_JOB_FLOW",
    HadoopJarStep = stepFactory.NewEnableDebuggingStep()
};

var installHive = new StepConfig{
    Name = "Install Hive",
    ActionOnFailure = "TERMINATE_JOB_FLOW",
    HadoopJarStep = stepFactory.NewInstallHiveStep()
};

var instanceConfig = new JobFlowInstancesConfig{
    Ec2KeyName = "keypair",
    InstanceCount = 5,
    KeepJobFlowAliveWhenNoSteps = true,
    MasterInstanceType = "m1.small",
    SlaveInstanceType = "m1.small"
};
```
var request = new RunJobFlowRequest{
    Name = "Hive Interactive",
    Steps = {enabledebugging, installHive},
    AmiVersion = "3.8.0",
    LogUri = "s3://myawsbucket",
    Instances = instanceConfig,
    ServiceRole = "service_role",
    JobFlowRole = "jobflow_role"
};

var result = emrClient.RunJobFlow(request);

At minimum, you must pass a service role and jobflow role corresponding to EMR_DefaultRole and EMR_EC2_DefaultRole, respectively. You can do this by invoking this AWS CLI command for the same account. First, look to see if the roles already exist:

```bash
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:

```json
"RoleName": "EMR_DefaultRole",
"Arn": "arn:aws:iam::AccountID:role/EMR_DefaultRole"
"RoleName": "EMR_EC2_DefaultRole",
"Arn": "arn:aws:iam::AccountID:role/EMR_EC2_DefaultRole"
```

If the default roles do not exist, you can use the following AWS CLI command to create them:

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

### Using the Java SDK to Sign an API Request

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Amazon EMR uses the Signature Version 4 signing process to construct signed requests to AWS. For more information, see Signature Version 4 Signing Process in the Amazon Web Services General Reference.
Hadoop Configuration Reference

Apache Hadoop runs on the EC2 instances launched in a cluster. Amazon EMR defines a default configuration for Hadoop, depending on the Amazon Machine Image (AMI) that you specify when you launch the cluster. For more information about the supported AMI versions, see Choose an Amazon Machine Image (AMI) (p. 69).

**Note**
Although Amazon EMR makes an effort to provide optimal configuration settings for each instance type for the broadest range of use cases, it is possible that you may need to manually adjust these settings for the needs of your application.

The following sections describe the various configuration settings and mechanisms available in Amazon EMR.

**Topics**
- JSON Configuration Files (p. 543)
- Configuration of hadoop-user-env.sh (p. 547)
- Hadoop 2.2.0 and 2.4.0 Default Configuration (p. 548)
- Hadoop 1.0.3 Default Configuration (p. 564)
- Hadoop 20.205 Default Configuration (Deprecated) (p. 579)

JSON Configuration Files

When Amazon EMR creates a Hadoop cluster, each node contains a pair of JSON files containing configuration information about the node and the currently running cluster. These files are in the `/mnt/var/lib/info` directory, and accessible by scripts running on the node.

**Node Settings**
Settings for an Amazon EMR cluster node are contained in the `instance.json` file.

The following table describes the contents of the `instance.json` file.
### Node Settings

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>isMaster</td>
<td>Indicates that is the master node.</td>
</tr>
<tr>
<td></td>
<td>Type: Boolean</td>
</tr>
<tr>
<td>isRunningNameNode</td>
<td>Indicates that this node is running the Hadoop name node daemon.</td>
</tr>
<tr>
<td></td>
<td>Type: Boolean</td>
</tr>
<tr>
<td>isRunningDataNode</td>
<td>Indicates that this node is running the Hadoop data node daemon.</td>
</tr>
<tr>
<td></td>
<td>Type: Boolean</td>
</tr>
<tr>
<td>isRunningJobTracker</td>
<td>Indicates that this node is running the Hadoop job tracker daemon.</td>
</tr>
<tr>
<td></td>
<td>Type: Boolean</td>
</tr>
<tr>
<td>isRunningTaskTracker</td>
<td>Indicates that this node is running the Hadoop task tracker daemon.</td>
</tr>
<tr>
<td></td>
<td>Type: Boolean</td>
</tr>
</tbody>
</table>

Hadoop 2.2.0 adds the following parameters to the `instance.json` file.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>isRunningResourceManager</td>
<td>Indicates that this node is running the Hadoop resource manager daemon.</td>
</tr>
<tr>
<td></td>
<td>Type: Boolean</td>
</tr>
<tr>
<td>isRunningNodeManager</td>
<td>Indicates that this node is running the Hadoop node manager daemon.</td>
</tr>
<tr>
<td></td>
<td>Type: Boolean</td>
</tr>
</tbody>
</table>

The following example shows the contents of an `instance.json` file:

```json
{
    "instanceGroupId": "Instance_Group_ID",
    "isMaster": Boolean,
    "isRunningNameNode": Boolean,
    "isRunningDataNode": Boolean,
    "isRunningJobTracker": Boolean,
    "isRunningTaskTracker": Boolean
}
```

### To read settings in `instance.json` with a bootstrap action using the AWS CLI

This procedure uses a run-if bootstrap action to demonstrate how to execute the command line function `echo` to display the string `running on master node` by evaluating the JSON file parameter `instance.isMaster` in the `instance.json` file.

- To read settings in `instance.json`, type the following command and replace `myKey` with the name of your EC2 key pair.

```bash
  clang -isYSTEM-EC2-keypair-key-pair-key-username > instance-permissions-policy.json
```

- Linux, UNIX, and Mac OS X users:
aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applications Name=Hue Name=Hive Name=Pig \
--use-default-roles --ec2-attributes KeyName=myKey \
--instance-type m3.xlarge --instance-count 3 \
--bootstrap-action Path=s3://elasticmapreduce/bootstrap-actions/run-if,Name="Run-if Bootstrap",Args=["instance.isMaster=true", "echo 'Running on master node'"]

Windows users:

aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applications Name=Hue Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey --instance-type m3.xlarge --instance-count 3 --bootstrap-action Path=s3://elasticmapreduce/bootstrap-actions/run-if,Name="Run-if Bootstrap",Args=["instance.isMaster=true", "echo 'Running on master node'"]

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

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

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

## Cluster Configuration

Information about the currently running cluster is contained in the `job-flow.json` file.

The following table describes the contents of the `job-flow.json` file.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JobFlowID</td>
<td>Contains the ID for the cluster. Type: String</td>
</tr>
<tr>
<td>jobFlowCreationInstant</td>
<td>Contains the time that the cluster was created. Type: Long</td>
</tr>
<tr>
<td>instanceCount</td>
<td>Contains the number of nodes in an instance group. Type: Integer</td>
</tr>
<tr>
<td>masterInstanceID</td>
<td>Contains the ID for the master node. Type: String</td>
</tr>
<tr>
<td>masterPrivateDnsName</td>
<td>Contains the private DNS name of the master node. Type: String</td>
</tr>
<tr>
<td>masterInstanceType</td>
<td>Contains the EC2 instance type of the master node.</td>
</tr>
</tbody>
</table>
### Cluster Configuration

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>slaveInstanceType</td>
<td>Contains the EC2 instance type of the slave nodes.</td>
</tr>
<tr>
<td>HadoopVersion</td>
<td>Contains the version of Hadoop running on the cluster.</td>
</tr>
<tr>
<td>instanceGroups</td>
<td>A list of objects specifying each instance group in the cluster</td>
</tr>
<tr>
<td>instanceGroupId</td>
<td>Unique identifier for this instance group.</td>
</tr>
<tr>
<td>instanceGroupName</td>
<td>User defined name of the instance group.</td>
</tr>
<tr>
<td>instanceRole</td>
<td>One of Master, Core, or Task.</td>
</tr>
<tr>
<td>instanceType</td>
<td>The Amazon EC2 type of the node, such as &quot;m1.small&quot;.</td>
</tr>
<tr>
<td>requestedInstanceCount</td>
<td>The target number of nodes for this instance group.</td>
</tr>
</tbody>
</table>

The following example shows the contents of an `job-flow.json` file.

```json
{
  "jobFlowId":"JobFlowID",
  "jobFlowCreationInstant": CreationInstanceID,
  "instanceCount": Count,
  "masterInstanceId":"MasterInstanceID",
  "masterPrivateDnsName":"Name",
  "masterInstanceType":"Amazon_EC2_Instance_Type",
  "slaveInstanceType":"Amazon_EC2_Instance_Type",
  "hadoopVersion":"Version",
  "instanceGroups": [
    {
      "instanceGroupId":"InstanceGroupID",
      "instanceGroupName":"Name",
      "instanceRole":"Master",
      "marketType":"Type",
      "instanceType":"AmazonEC2InstanceType",
      "requestedInstanceCount": Count
    },
    {
      "instanceGroupId":"InstanceGroupID",
      "instanceGroupName":"Name",
      "instanceRole":"Core",
      "marketType":"Type",
```
Configuration of hadoop-user-env.sh

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

When you run a Hadoop daemon or job, a number of scripts are executed as part of the initialization process. The executable hadoop is actually the alias for a Bash script called /home/hadoop/bin/hadoop. This script is responsible for setting up the Java classpath, configuring the Java memory settings, determining which main class to run, and executing the actual Java process.

As part of the Hadoop configuration, the hadoop script executes a file called conf/hadoop-env.sh. The hadoop-env.sh script can set various environment variables. The conf/hadoop-env.sh script is used so that the main bin/hadoop script remains unmodified. Amazon EMR creates a hadoop-env.sh script on every node in a cluster in order to configure the amount of memory for every Hadoop daemon launched.

You can create a script, conf/hadoop-user-env.sh, to allow you to override the default Hadoop settings that Amazon EMR configures.

You should put your custom overrides for the Hadoop environment variables in conf/hadoop-user-env.sh. Custom overrides could include items such as changes to Java memory or naming additional JAR files in the classpath. The script is also where Amazon EMR writes data when you use a bootstrap action to configure memory or specifying additional Java args.

Note

If you want your custom classpath to override the original class path, you should set the environment variable, HADOOP_USER_CLASSPATH_FIRST to true so that the HADOOP_CLASSPATH value specified in hadoop-user-env.sh is first.

Examples of environment variables that you can specify in hadoop-user-env.sh include:

- export HADOOP_DATANODE_HEAPSIZE="128"
- export HADOOP_JOBTRACKER_HEAPSIZE="768"
- export HADOOP_NAMENODE_HEAPSIZE="256"
- export HADOOP_OPTS="-server"
- export HADOOP_TASKTRACKER_HEAPSIZE="512"

In addition, Hadoop 2.2.0 adds the following new environment variables that you can specify in hadoop-user-env.sh:

- YARN_RESOURCEMANAGER_HEAPSIZE="128"
• YARN_NODEMANAGER_HEAPSIZE="768"

For more information, go to the Hadoop MapReduce Next Generation - Cluster Setup topic on the Hadoop Apache website.

A bootstrap action runs before Hadoop starts and before any steps are run. In some cases, it is necessary to configure the Hadoop environment variables referenced in the Hadoop launch script.

If the script /home/hadoop/conf/hadoop-user-env.sh exists when Hadoop launches, Amazon EMR executes this script and any options are inherited by bin/hadoop.

For example, to add a JAR file to the beginning of the Hadoop daemon classpath, you can use a bootstrap action such as:

```
echo 'export HADOOP_USER_CLASSPATH_FIRST=true' >> /home/hadoop/conf/hadoop-user-env.sh
echo 'export HADOOP_CLASSPATH=/path/to/my.jar:$HADOOP_CLASSPATH' >> /home/hadoop/conf/hadoop-user-env.sh
```

For more information about using bootstrap actions, see (Optional) Create Bootstrap Actions to Install Additional Software (p. 129).

Hadoop 2.2.0 and 2.4.0 Default Configuration

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

Topics
- Hadoop Configuration (Hadoop 2.2.0, 2.4.0) (p. 548)
- HDFS Configuration (Hadoop 2.2.0) (p. 560)
- Task Configuration (Hadoop 2.2.0) (p. 561)
- Intermediate Compression (Hadoop 2.2.0) (p. 562)

This section describes the default configuration settings that Amazon EMR uses to configure a Hadoop cluster launched with Hadoop 2.2.0. For more information about the AMI versions supported by Amazon EMR, see Choose an Amazon Machine Image (AMI) (p. 69).

Hadoop Configuration (Hadoop 2.2.0, 2.4.0)

This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

The following tables list the default configuration settings for each EC2 instance type in clusters launched with the Amazon EMR Hadoop 2.2.0. For more information about the AMI versions supported by Amazon EMR, see Choose an Amazon Machine Image (AMI) (p. 69).

**m1.medium**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARN_RESourcemanager_HEAPSIZE</td>
<td>384</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>YARN_PROXYSERVER_HEAPSIZE</td>
<td>192</td>
</tr>
<tr>
<td>YARN_NODEMANAGER_HEAPSIZE</td>
<td>256</td>
</tr>
<tr>
<td>HADOOP_JOB_HISTORYSERVER_HEAPSIZE</td>
<td>256</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>384</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>192</td>
</tr>
</tbody>
</table>

**m1.large**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARN_RESOURCEMANAGER_HEAPSIZE</td>
<td>768</td>
</tr>
<tr>
<td>YARN_PROXYSERVER_HEAPSIZE</td>
<td>384</td>
</tr>
<tr>
<td>YARN_NODEMANAGER_HEAPSIZE</td>
<td>512</td>
</tr>
<tr>
<td>HADOOP_JOB_HISTORYSERVER_HEAPSIZE</td>
<td>512</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>768</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>384</td>
</tr>
</tbody>
</table>

**m1.xlarge**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARN_RESOURCEMANAGER_HEAPSIZE</td>
<td>1024</td>
</tr>
<tr>
<td>YARN_PROXYSERVER_HEAPSIZE</td>
<td>512</td>
</tr>
<tr>
<td>YARN_NODEMANAGER_HEAPSIZE</td>
<td>768</td>
</tr>
<tr>
<td>HADOOP_JOB_HISTORYSERVER_HEAPSIZE</td>
<td>1024</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>2304</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>384</td>
</tr>
</tbody>
</table>

**m2.xlarge**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARN_RESOURCEMANAGER_HEAPSIZE</td>
<td>1536</td>
</tr>
<tr>
<td>YARN_PROXYSERVER_HEAPSIZE</td>
<td>1024</td>
</tr>
<tr>
<td>YARN_NODEMANAGER_HEAPSIZE</td>
<td>1024</td>
</tr>
<tr>
<td>HADOOP_JOB_HISTORYSERVER_HEAPSIZE</td>
<td>1024</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>3072</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>384</td>
</tr>
</tbody>
</table>
### m2.2xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARN_RESOURCEMANAGER_HEAPSIZE</td>
<td>1536</td>
</tr>
<tr>
<td>YARN_PROXYSERVER_HEAPSIZE</td>
<td>1024</td>
</tr>
<tr>
<td>YARN_NODEMANAGER_HEAPSIZE</td>
<td>1024</td>
</tr>
<tr>
<td>HADOOP_JOB_HISTORYSERVER_HEAPSIZE</td>
<td>1536</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>6144</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>384</td>
</tr>
</tbody>
</table>

### m2.4xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARN_RESOURCEMANAGER_HEAPSIZE</td>
<td>2048</td>
</tr>
<tr>
<td>YARN_PROXYSERVER_HEAPSIZE</td>
<td>1024</td>
</tr>
<tr>
<td>YARN_NODEMANAGER_HEAPSIZE</td>
<td>1536</td>
</tr>
<tr>
<td>HADOOP_JOB_HISTORYSERVER_HEAPSIZE</td>
<td>1536</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>12288</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>384</td>
</tr>
</tbody>
</table>

### m3.xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARN_RESOURCEMANAGER_HEAPSIZE</td>
<td>2396</td>
</tr>
<tr>
<td>YARN_PROXYSERVER_HEAPSIZE</td>
<td>2396</td>
</tr>
<tr>
<td>YARN_NODEMANAGER_HEAPSIZE</td>
<td>2048</td>
</tr>
<tr>
<td>HADOOP_JOB_HISTORYSERVER_HEAPSIZE</td>
<td>2396</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>1740</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>757</td>
</tr>
</tbody>
</table>

### m4.large

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARN_RESOURCEMANAGER_HEAPSIZE</td>
<td>2252</td>
</tr>
<tr>
<td>YARN_PROXYSERVER_HEAPSIZE</td>
<td>2252</td>
</tr>
<tr>
<td>YARN_NODEMANAGER_HEAPSIZE</td>
<td>2048</td>
</tr>
<tr>
<td>HADOOP_JOB_HISTORYSERVER_HEAPSIZE</td>
<td>2252</td>
</tr>
</tbody>
</table>
### m4.xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>1024</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>614</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARN_RESourcemANAGER_HEAPSIZE</td>
<td>2416</td>
</tr>
<tr>
<td>YARN_PROXYSERVER_HEAPSIZE</td>
<td>2416</td>
</tr>
<tr>
<td>YARN_NODEMANAGER_HEAPSIZE</td>
<td>2048</td>
</tr>
<tr>
<td>HADOOP_JOB_HISTORYSERVER_HEAPSIZE</td>
<td>2416</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>2048</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>778</td>
</tr>
</tbody>
</table>

### m4.2xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARN_RESourcemANAGER_HEAPSIZE</td>
<td>2744</td>
</tr>
<tr>
<td>YARN_PROXYSERVER_HEAPSIZE</td>
<td>2744</td>
</tr>
<tr>
<td>YARN_NODEMANAGER_HEAPSIZE</td>
<td>2048</td>
</tr>
<tr>
<td>HADOOP_JOB_HISTORYSERVER_HEAPSIZE</td>
<td>2744</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>3481</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>1105</td>
</tr>
</tbody>
</table>

### m4.4xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARN_RESourcemANAGER_HEAPSIZE</td>
<td>3399</td>
</tr>
<tr>
<td>YARN_PROXYSERVER_HEAPSIZE</td>
<td>3399</td>
</tr>
<tr>
<td>YARN_NODEMANAGER_HEAPSIZE</td>
<td>2048</td>
</tr>
<tr>
<td>HADOOP_JOB_HISTORYSERVER_HEAPSIZE</td>
<td>3399</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>6758</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>1761</td>
</tr>
</tbody>
</table>
### m4.10xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARN_RESOURCEMANAGER_HEAPSIZE</td>
<td>5365</td>
</tr>
<tr>
<td>YARN_PROXYSERVER_HEAPSIZE</td>
<td>5365</td>
</tr>
<tr>
<td>YARN_NODEMANAGER_HEAPSIZE</td>
<td>2048</td>
</tr>
<tr>
<td>HADOOP_JOB_HISTORYSERVER_HEAPSIZE</td>
<td>5365</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>16588</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>3727</td>
</tr>
</tbody>
</table>

### m3.2xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARN_RESOURCEMANAGER_HEAPSIZE</td>
<td>2703</td>
</tr>
<tr>
<td>YARN_PROXYSERVER_HEAPSIZE</td>
<td>2703</td>
</tr>
<tr>
<td>YARN_NODEMANAGER_HEAPSIZE</td>
<td>2048</td>
</tr>
<tr>
<td>HADOOP_JOB_HISTORYSERVER_HEAPSIZE</td>
<td>2703</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>3276</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>1064</td>
</tr>
</tbody>
</table>

### c1.medium

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARN_RESOURCEMANAGER_HEAPSIZE</td>
<td>192</td>
</tr>
<tr>
<td>YARN_PROXYSERVER_HEAPSIZE</td>
<td>96</td>
</tr>
<tr>
<td>YARN_NODEMANAGER_HEAPSIZE</td>
<td>128</td>
</tr>
<tr>
<td>HADOOP_JOB_HISTORYSERVER_HEAPSIZE</td>
<td>128</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>192</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>96</td>
</tr>
</tbody>
</table>

### c1.xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARN_RESOURCEMANAGER_HEAPSIZE</td>
<td>768</td>
</tr>
<tr>
<td>YARN_PROXYSERVER_HEAPSIZE</td>
<td>384</td>
</tr>
<tr>
<td>YARN_NODEMANAGER_HEAPSIZE</td>
<td>512</td>
</tr>
<tr>
<td>HADOOP_JOB_HISTORYSERVER_HEAPSIZE</td>
<td>512</td>
</tr>
</tbody>
</table>
### Hadoop Configuration (Hadoop 2.2.0, 2.4.0)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>768</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>384</td>
</tr>
</tbody>
</table>

#### c3.xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARN_RESOURCEMANAGER_HEAPSIZE</td>
<td>2124</td>
</tr>
<tr>
<td>YARN_PROXYSERVER_HEAPSIZE</td>
<td>2124</td>
</tr>
<tr>
<td>YARN_NODEMANAGER_HEAPSIZE</td>
<td>2124</td>
</tr>
<tr>
<td>HADOOP_JOB_HISTORYSERVER_HEAPSIZE</td>
<td>2124</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>972</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>588</td>
</tr>
</tbody>
</table>

#### c3.2xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARN_RESOURCEMANAGER_HEAPSIZE</td>
<td>2396</td>
</tr>
<tr>
<td>YARN_PROXYSERVER_HEAPSIZE</td>
<td>2396</td>
</tr>
<tr>
<td>YARN_NODEMANAGER_HEAPSIZE</td>
<td>2048</td>
</tr>
<tr>
<td>HADOOP_JOB_HISTORYSERVER_HEAPSIZE</td>
<td>2396</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>1740</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>757</td>
</tr>
</tbody>
</table>

#### c3.4xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARN_RESOURCEMANAGER_HEAPSIZE</td>
<td>2703</td>
</tr>
<tr>
<td>YARN_PROXYSERVER_HEAPSIZE</td>
<td>2703</td>
</tr>
<tr>
<td>YARN_NODEMANAGER_HEAPSIZE</td>
<td>2048</td>
</tr>
<tr>
<td>HADOOP_JOB_HISTORYSERVER_HEAPSIZE</td>
<td>2703</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>3276</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>1064</td>
</tr>
</tbody>
</table>
### c3.8xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARN_RESCOURCEMANAGER_HEAPSIZE</td>
<td>3317</td>
</tr>
<tr>
<td>YARN_PROXYSERVER_HEAPSIZE</td>
<td>3317</td>
</tr>
<tr>
<td>YARN_NODEMANAGER_HEAPSIZE</td>
<td>2048</td>
</tr>
<tr>
<td>HADOOP_JOB_HISTORYSERVER_HEAPSIZE</td>
<td>3317</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>6348</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>1679</td>
</tr>
</tbody>
</table>

### c4.large

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARN_RESCOURCEMANAGER_HEAPSIZE</td>
<td>1152</td>
</tr>
<tr>
<td>YARN_PROXYSERVER_HEAPSIZE</td>
<td>1152</td>
</tr>
<tr>
<td>YARN_NODEMANAGER_HEAPSIZE</td>
<td>1152</td>
</tr>
<tr>
<td>HADOOP_JOB_HISTORYSERVER_HEAPSIZE</td>
<td>1152</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>576</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>384</td>
</tr>
</tbody>
</table>

### c4.xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARN_RESCOURCEMANAGER_HEAPSIZE</td>
<td>2124</td>
</tr>
<tr>
<td>YARN_PROXYSERVER_HEAPSIZE</td>
<td>2124</td>
</tr>
<tr>
<td>YARN_NODEMANAGER_HEAPSIZE</td>
<td>2048</td>
</tr>
<tr>
<td>HADOOP_JOB_HISTORYSERVER_HEAPSIZE</td>
<td>2124</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>972</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>588</td>
</tr>
</tbody>
</table>

### c4.2xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARN_RESCOURCEMANAGER_HEAPSIZE</td>
<td>2396</td>
</tr>
<tr>
<td>YARN_PROXYSERVER_HEAPSIZE</td>
<td>2396</td>
</tr>
<tr>
<td>YARN_NODEMANAGER_HEAPSIZE</td>
<td>2048</td>
</tr>
<tr>
<td>HADOOP_JOB_HISTORYSERVER_HEAPSIZE</td>
<td>2396</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>1740</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>757</td>
</tr>
</tbody>
</table>

**c4.4xlarge**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARN_RESOURCEMANAGER_HEAPSIZE</td>
<td>2703</td>
</tr>
<tr>
<td>YARN_PROXYSERVER_HEAPSIZE</td>
<td>2703</td>
</tr>
<tr>
<td>YARN_NODEMANAGER_HEAPSIZE</td>
<td>2048</td>
</tr>
<tr>
<td>HADOOP_JOB_HISTORYSERVER_HEAPSIZE</td>
<td>2703</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>3276</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>1064</td>
</tr>
</tbody>
</table>

**c4.8xlarge**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARN_RESOURCEMANAGER_HEAPSIZE</td>
<td>3317</td>
</tr>
<tr>
<td>YARN_PROXYSERVER_HEAPSIZE</td>
<td>3317</td>
</tr>
<tr>
<td>YARN_NODEMANAGER_HEAPSIZE</td>
<td>2048</td>
</tr>
<tr>
<td>HADOOP_JOB_HISTORYSERVER_HEAPSIZE</td>
<td>3317</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>6348</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>1679</td>
</tr>
</tbody>
</table>

**cc2.8xlarge**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARN_RESOURCEMANAGER_HEAPSIZE</td>
<td>2048</td>
</tr>
<tr>
<td>YARN_PROXYSERVER_HEAPSIZE</td>
<td>1024</td>
</tr>
<tr>
<td>YARN_NODEMANAGER_HEAPSIZE</td>
<td>1536</td>
</tr>
<tr>
<td>HADOOP_JOB_HISTORYSERVER_HEAPSIZE</td>
<td>1536</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>12288</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>384</td>
</tr>
</tbody>
</table>
### cg1.4xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARN_RESourcemanager_heapSize</td>
<td>2048</td>
</tr>
<tr>
<td>YARN_Proxyserver_heapSize</td>
<td>1024</td>
</tr>
<tr>
<td>YARN_nodeManager_heapSize</td>
<td>1536</td>
</tr>
<tr>
<td>Hadoop_jobHistoryserver_heapSize</td>
<td>1536</td>
</tr>
<tr>
<td>Hadoop_NameNode_heapSize</td>
<td>3840</td>
</tr>
<tr>
<td>Hadoop_Datanode_heapSize</td>
<td>384</td>
</tr>
</tbody>
</table>

### cr1.8xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARN_RESourcemanager_heapSize</td>
<td>7086</td>
</tr>
<tr>
<td>YARN_Proxyserver_heapSize</td>
<td>7086</td>
</tr>
<tr>
<td>YARN_nodeManager_heapSize</td>
<td>2048</td>
</tr>
<tr>
<td>Hadoop_jobHistoryserver_heapSize</td>
<td>7086</td>
</tr>
<tr>
<td>Hadoop_NameNode_heapSize</td>
<td>25190</td>
</tr>
<tr>
<td>Hadoop_Datanode_heapSize</td>
<td>4096</td>
</tr>
</tbody>
</table>

### d2.xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARN_RESourcemanager_heapSize</td>
<td>2713</td>
</tr>
<tr>
<td>YARN_Proxyserver_heapSize</td>
<td>2713</td>
</tr>
<tr>
<td>YARN_nodeManager_heapSize</td>
<td>2048</td>
</tr>
<tr>
<td>Hadoop_jobHistoryserver_heapSize</td>
<td>2713</td>
</tr>
<tr>
<td>Hadoop_NameNode_heapSize</td>
<td>3328</td>
</tr>
<tr>
<td>Hadoop_Datanode_heapSize</td>
<td>1075</td>
</tr>
</tbody>
</table>

### d2.2xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARN_RESourcemanager_heapSize</td>
<td>3338</td>
</tr>
<tr>
<td>YARN_Proxyserver_heapSize</td>
<td>3338</td>
</tr>
<tr>
<td>YARN_nodeManager_heapSize</td>
<td>2048</td>
</tr>
<tr>
<td>Hadoop_jobHistoryserver_heapSize</td>
<td>3338</td>
</tr>
</tbody>
</table>
### Hadoop Configuration (Hadoop 2.2.0, 2.4.0)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>6451</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>1699</td>
</tr>
</tbody>
</table>

**d2.4xlarge**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARN_RESOURCEMANAGER_HEAPSIZE</td>
<td>4587</td>
</tr>
<tr>
<td>YARN_PROXYSERVER_HEAPSIZE</td>
<td>4587</td>
</tr>
<tr>
<td>YARN_NODEMANAGER_HEAPSIZE</td>
<td>2048</td>
</tr>
<tr>
<td>HADOOP_JOB_HISTORYSERVER_HEAPSIZE</td>
<td>4587</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>12697</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>2949</td>
</tr>
</tbody>
</table>

**d2.8xlarge**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARN_RESOURCEMANAGER_HEAPSIZE</td>
<td>7089</td>
</tr>
<tr>
<td>YARN_PROXYSERVER_HEAPSIZE</td>
<td>7086</td>
</tr>
<tr>
<td>YARN_NODEMANAGER_HEAPSIZE</td>
<td>2048</td>
</tr>
<tr>
<td>HADOOP_JOB_HISTORYSERVER_HEAPSIZE</td>
<td>7086</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>25190</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>4096</td>
</tr>
</tbody>
</table>

**hi1.4xlarge**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARN_RESOURCEMANAGER_HEAPSIZE</td>
<td>3328</td>
</tr>
<tr>
<td>YARN_PROXYSERVER_HEAPSIZE</td>
<td>3328</td>
</tr>
<tr>
<td>YARN_NODEMANAGER_HEAPSIZE</td>
<td>2048</td>
</tr>
<tr>
<td>HADOOP_JOB_HISTORYSERVER_HEAPSIZE</td>
<td>3328</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>6400</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>1689</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>YARN_RESOURCEMANAGER_HEAPSIZE</td>
<td>2048</td>
</tr>
<tr>
<td>YARN_PROXYSERVER_HEAPSIZE</td>
<td>1024</td>
</tr>
<tr>
<td>YARN_NODEMANAGER_HEAPSIZE</td>
<td>1536</td>
</tr>
<tr>
<td>HADOOP_JOB_HISTORYSERVER_HEAPSIZE</td>
<td>1536</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>12288</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>384</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARN_RESOURCEMANAGER_HEAPSIZE</td>
<td>2713</td>
</tr>
<tr>
<td>YARN_PROXYSERVER_HEAPSIZE</td>
<td>2713</td>
</tr>
<tr>
<td>YARN_NODEMANAGER_HEAPSIZE</td>
<td>2048</td>
</tr>
<tr>
<td>HADOOP_JOB_HISTORYSERVER_HEAPSIZE</td>
<td>2713</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>3328</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>1075</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARN_RESOURCEMANAGER_HEAPSIZE</td>
<td>3338</td>
</tr>
<tr>
<td>YARN_PROXYSERVER_HEAPSIZE</td>
<td>3338</td>
</tr>
<tr>
<td>YARN_NODEMANAGER_HEAPSIZE</td>
<td>2048</td>
</tr>
<tr>
<td>HADOOP_JOB_HISTORYSERVER_HEAPSIZE</td>
<td>3338</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>6451</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>1699</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARN_RESOURCEMANAGER_HEAPSIZE</td>
<td>4587</td>
</tr>
<tr>
<td>YARN_PROXYSERVER_HEAPSIZE</td>
<td>4587</td>
</tr>
<tr>
<td>YARN_NODEMANAGER_HEAPSIZE</td>
<td>2048</td>
</tr>
<tr>
<td>HADOOP_JOB_HISTORYSERVER_HEAPSIZE</td>
<td>4587</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>12697</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>2949</td>
</tr>
</tbody>
</table>

**i2.8xlarge**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARN_RESourcemanager_HEAPSIZE</td>
<td>7086</td>
</tr>
<tr>
<td>YARN_PROXYserver_HEAPSIZE</td>
<td>7086</td>
</tr>
<tr>
<td>YARN_NODEMANAGER_HEAPSIZE</td>
<td>2048</td>
</tr>
<tr>
<td>HADOOP_JOB_HISTORYSERVER_HEAPSIZE</td>
<td>7086</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>25190</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>4096</td>
</tr>
</tbody>
</table>

**g2.2xlarge**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARN_RESourcemanager_HEAPSIZE</td>
<td>1536</td>
</tr>
<tr>
<td>YARN_PROXYserver_HEAPSIZE</td>
<td>1024</td>
</tr>
<tr>
<td>YARN_NODEMANAGER_HEAPSIZE</td>
<td>1024</td>
</tr>
<tr>
<td>HADOOP_JOB_HISTORYSERVER_HEAPSIZE</td>
<td>1024</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>2304</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>384</td>
</tr>
</tbody>
</table>

**r3.xlarge**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>YARN_RESourcemanager_HEAPSIZE</td>
<td>2713</td>
</tr>
<tr>
<td>YARN_PROXYserver_HEAPSIZE</td>
<td>2713</td>
</tr>
<tr>
<td>YARN_NODEMANAGER_HEAPSIZE</td>
<td>2048</td>
</tr>
<tr>
<td>HADOOP_JOB_HISTORYSERVER_HEAPSIZE</td>
<td>2713</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>3328</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>1075</td>
</tr>
</tbody>
</table>
HDFS Configuration (Hadoop 2.2.0)

The following table describes the default Hadoop Distributed File System (HDFS) parameters and their settings.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>dfs.block.size</td>
<td>The size of HDFS blocks. When operating on data stored in HDFS, the split size is generally the size of an HDFS block.</td>
<td>134217728 (128 MB)</td>
</tr>
</tbody>
</table>
### Task Configuration (Hadoop 2.2.0)

#### Topics
- Avoiding Cluster Slowdowns (AMI 3.0.0) (p. 561)

There are a number of configuration variables for tuning the performance of your MapReduce jobs. This section describes some of the important task-related settings.

**Avoiding Cluster Slowdowns (AMI 3.0.0)**

**Hadoop Speculative Execution**

In a distributed environment, you are going to experience random delays, slow hardware, failing hardware, and other problems that collectively slow down your cluster. This is known as the *stragglers* problem. Hadoop has a feature called *speculative execution* that can help mitigate this issue. As the cluster progresses, some machines complete their tasks. Hadoop schedules tasks on nodes that are free. Whichever task finishes first is the successful one, and the other tasks are killed. This feature can substantially cut down on the run time of jobs. The general design of a mapreduce algorithm is such that the processing of map tasks is meant to be idempotent. However, if you are running a job where the task execution has side effects (for example, a zero reducer job that calls an external resource), it is important to disable speculative execution.

You can enable speculative execution for mappers and reducers independently. By default, Amazon EMR enables it for mappers and reducers in AMI 2.3. You can override these settings with a bootstrap action. For more information about using bootstrap actions, see (Optional) Create Bootstrap Actions to Install Additional Software (p. 129).

**Speculative Execution Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>mapred.map.tasks.speculative.execution</td>
<td>true</td>
</tr>
<tr>
<td>mapred.reduce.tasks.speculative.execution</td>
<td>true</td>
</tr>
</tbody>
</table>

To disable reducer speculative execution via a bootstrap action using the AWS CLI

To disable reduce speculative execution using the AWS CLI, type the `--bootstrap-action` parameter and specify the arguments.

Larger numbers provide less task granularity, but also put less strain on the cluster NameNode.

This determines how many copies of each block to store for durability. For small clusters, we set this to 2 because the cluster is small and easy to restart in case of data loss. You can change the setting to 1, 2, or 3 as your needs dictate. Amazon EMR automatically calculates the replication factor based on cluster size. To overwrite the default value, use a configure-hadoop bootstrap action.

### Parameter | Definition | Default Value
---|---|---
| dfs.replication | This determines how many copies of each block to store for durability. For small clusters, we set this to 2 because the cluster is small and easy to restart in case of data loss. You can change the setting to 1, 2, or 3 as your needs dictate. Amazon EMR automatically calculates the replication factor based on cluster size. To overwrite the default value, use a configure-hadoop bootstrap action. | 1 for clusters < four nodes
2 for clusters < ten nodes
3 for all other clusters |
• Type the following command to disable reducer speculative execution and replace `myKey` with the name of your EC2 key pair.

  ```bash
  aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applications Name=Hue Name=Hive Name=Pig
  --use-default-roles --ec2-attributes KeyName=myKey
  --instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m3.xlarge
  InstanceGroupType=CORE,InstanceCount=2,InstanceType=m3.xlarge
  --bootstrap-action Path=s3://elasticmapreduce/bootstrap-actions/configure-hadoop,Name="Disable reducer speculative execution",Args=["-m","mapred.reduce.tasks.speculative.execution=false"]
  ```

• Windows users:

  ```bash
  aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applications Name=Hue Name=Hive Name=Pig
  --use-default-roles --ec2-attributes KeyName=myKey
  --instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m3.xlarge
  InstanceGroupType=CORE,InstanceCount=2,InstanceType=m3.xlarge
  --bootstrap-action Path=s3://elasticmapreduce/bootstrap-actions/configure-hadoop,Name="Disable reducer speculative execution",Args=["-m","mapred.reduce.tasks.speculative.execution=false"]
  ```

**Note**

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

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

**Intermediate Compression (Hadoop 2.2.0)**

Hadoop sends data between the mappers and reducers in its shuffle process. This network operation is a bottleneck for many clusters. To reduce this bottleneck, Amazon EMR enables intermediate data compression by default. Because it provides a reasonable amount of compression with only a small CPU impact, we use the Snappy codec.

You can modify the default compression settings with a bootstrap action. For more information about using bootstrap actions, see *(Optional) Create Bootstrap Actions to Install Additional Software (p. 129).*

The following table presents the default values for the parameters that affect intermediate compression.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>mapred.compress.map.output</td>
<td>true</td>
</tr>
<tr>
<td>mapred.map.output.compression.codec</td>
<td>org.apache.hadoop.io.compress.SnappyCodec</td>
</tr>
</tbody>
</table>

**To disable intermediate compression or change the compression codec via a bootstrap action using the AWS CLI**

To disable intermediate compression or to change the intermediate compression codec using the AWS CLI, type the `--bootstrap-action` parameter and specify the arguments.
1. To disable compression, type the following command and replace myKey with the name of your EC2 key pair.

   - Linux, UNIX, and Mac OS X users:

     ```bash
     aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applications Name=Hue Name=Hive Name=Pig \
     --use-default-roles --ec2-attributes KeyName=myKey \n     --instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m3.xlarge \n     InstanceGroupType=CORE,InstanceCount=2,InstanceType=m3.xlarge \n     --bootstrap-action Path=s3://elasticmapreduce/bootstrap-actions/configure-hadoop,Name="Disable compression",Args="-m","mapred.compress.map.output=false"
     ```

   - Windows users:

     ```bash
     aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applications Name=Hue Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey \n     --instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m3.xlarge \n     InstanceGroupType=CORE,InstanceCount=2,InstanceType=m3.xlarge --bootstrap-action Path=s3://elasticmapreduce/bootstrap-actions/configure-hadoop,Name="Disable compression",Args="-m","mapred.compress.map.output=false"
     ```

   **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.

2. To change the intermediate compression codec from Snappy to Gzip, type the following command and replace myKey with the name of your EC2 key pair.

   - Linux, UNIX, and Mac OS X users:

     ```bash
     aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applications Name=Hue Name=Hive Name=Pig \
     --use-default-roles --ec2-attributes KeyName=myKey \n     --instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m3.xlarge \n     InstanceGroupType=CORE,InstanceCount=2,InstanceType=m3.xlarge \n     --bootstrap-action Path=s3://elasticmapreduce/bootstrap-actions/configure-hadoop,Name="Change compression codec",Args="-m","mapred.map.output.compression.codec=org.apache.hadoop.io.compress.GzipCodec"
     ```

   - Windows users:

     ```bash
     aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applications Name=Hue Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey \n     --instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m3.xlarge \n     InstanceGroupType=CORE,InstanceCount=2,InstanceType=m3.xlarge --bootstrap-action Path=s3://elasticmapreduce/bootstrap-actions/configure-hadoop,Name="Change compression codec",Args="-m","mapred.map.output.compression.codec=org.apache.hadoop.io.compress.GzipCodec"
     ```

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

For more information on using Amazon EMR commands in the AWS CLI, see [http://docs.aws.amazon.com/cli/latest/reference/emr](http://docs.aws.amazon.com/cli/latest/reference/emr).
Hadoop 1.0.3 Default Configuration

This section describes the default configuration settings that Amazon EMR uses to configure a Hadoop cluster launched with Hadoop 1.0.3. For more information about the AMI versions supported by Amazon EMR, see Choose an Amazon Machine Image (AMI) (p. 69).

Topics
- Hadoop Configuration (Hadoop 1.0.3) (p. 564)
- HDFS Configuration (Hadoop 1.0.3) (p. 574)
- Task Configuration (Hadoop 1.0.3) (p. 575)
- Intermediate Compression (Hadoop 1.0.3) (p. 578)

Hadoop Configuration (Hadoop 1.0.3)

The following Amazon EMR default configuration settings for clusters launched with Amazon EMR AMI 2.3 are appropriate for most workloads.

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.

The following tables list the default configuration settings for each EC2 instance type in clusters launched with the Amazon EMR AMI version 2.3. For more information about the AMI versions supported by Amazon EMR, see Choose an Amazon Machine Image (AMI) (p. 69).

m1.small

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>576</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>192</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>192</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>96</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx288m</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>2</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>1</td>
</tr>
</tbody>
</table>

m1.medium

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>1152</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>384</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>192</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>192</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx576m</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>2</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>1</td>
</tr>
</tbody>
</table>

### m1.large

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>2304</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>768</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>384</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>384</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx864m</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>3</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>1</td>
</tr>
</tbody>
</table>

### m1.xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>6912</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>2304</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>384</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>384</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx768m</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>8</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>3</td>
</tr>
</tbody>
</table>

### m2.xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>9216</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>3072</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>384</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>384</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx2304m</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>3</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>1</td>
</tr>
</tbody>
</table>
### m2.2xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>18432</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>6144</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>384</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>384</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx2688m</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>6</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>2</td>
</tr>
</tbody>
</table>

### m2.4xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>36864</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>12288</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>384</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>384</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx2304m</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>14</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>4</td>
</tr>
</tbody>
</table>

### m3.xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>3686</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>1740</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>686</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>757</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx1440m</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>6</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>2</td>
</tr>
</tbody>
</table>

### m3.2xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>6758</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>3276</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>839</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>1064</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx1440m</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>12</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>4</td>
</tr>
</tbody>
</table>

### c1.medium

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>576</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>192</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>192</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>96</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx288m</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>2</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>1</td>
</tr>
</tbody>
</table>

### c1.xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>2304</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>768</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>384</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>384</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx384m</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>7</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>2</td>
</tr>
</tbody>
</table>

### c3.xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>2124</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>972</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>588</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>588</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx1408m</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>3</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>1</td>
</tr>
</tbody>
</table>

**c3.2xlarge**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>3686</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>1740</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>686</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>757</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx1440m</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>6</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>2</td>
</tr>
</tbody>
</table>

**c3.4xlarge**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>6758</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>3276</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>839</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>1064</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx1440m</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>12</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>4</td>
</tr>
</tbody>
</table>

**c3.8xlarge**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>12902</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>6348</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>1146</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>1679</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx1664m</td>
</tr>
</tbody>
</table>
### Parameter Values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>24</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>8</td>
</tr>
</tbody>
</table>

### cc2.8xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>30114</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>12288</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>384</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>384</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx1536m</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>24</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>6</td>
</tr>
</tbody>
</table>

### cg1.4xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>7680</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>3840</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>384</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>384</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx864m</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>12</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>3</td>
</tr>
</tbody>
</table>

### cr1.8xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>50585</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>25190</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>2048</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>4096</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx7552m</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>24</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>8</td>
</tr>
</tbody>
</table>
### hi1.4xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>30114</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>12288</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>384</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>384</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx1536m</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>24</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>6</td>
</tr>
</tbody>
</table>

### hs1.8xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>30114</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>12288</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>384</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>384</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx1536m</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>24</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>6</td>
</tr>
</tbody>
</table>

### cg1.4xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>7680</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>3840</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>384</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>384</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx864m</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>12</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>3</td>
</tr>
</tbody>
</table>

### d2.xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>6860</td>
</tr>
</tbody>
</table>
### Parameter Value

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>3328</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>844</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>1075</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx2928m</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>6</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>2</td>
</tr>
</tbody>
</table>

#### d2.2xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>13107</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>6451</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>1157</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>1699</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx3392m</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>12</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>4</td>
</tr>
</tbody>
</table>

#### d2.4xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>25600</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>12697</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>1781</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>2949</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx3648m</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>24</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>8</td>
</tr>
</tbody>
</table>

#### d2.8xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>50585</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>25190</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>2048</td>
</tr>
</tbody>
</table>
### Hadoop Configuration (Hadoop 1.0.3)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>4096</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx3356m</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>54</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>18</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>6912</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>2304</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>384</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>384</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx768m</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>8</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>13107</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>6451</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>1157</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>1699</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx3392m</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>12</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>4</td>
</tr>
</tbody>
</table>

### i2.4xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>25600</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>12697</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>1781</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>2949</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx3648m</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>24</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>8</td>
</tr>
</tbody>
</table>

### i2.8xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>50585</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>25190</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>2048</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>4096</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx3776m</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>48</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>16</td>
</tr>
</tbody>
</table>

### r3.xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>6860</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>3328</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>844</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>1075</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx2928m</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>6</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>2</td>
</tr>
</tbody>
</table>
### HDFS Configuration (Hadoop 1.0.3)

The following table describes the default Hadoop Distributed File System (HDFS) parameters and their settings.

#### r3.2xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>13107</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>6451</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>1157</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>1699</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx3392m</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>12</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>4</td>
</tr>
</tbody>
</table>

#### r3.4xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>25600</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>12697</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>1781</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>2949</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx7296m</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>12</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>4</td>
</tr>
</tbody>
</table>

#### r3.8xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>50585</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>25190</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>2048</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>4096</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx7552m</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>24</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>8</td>
</tr>
<tr>
<td>Parameter</td>
<td>Definition</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>dfs.block.size</td>
<td>The size of HDFS blocks. When operating on data stored in HDFS, the split size is generally the size of an HDFS block. Larger numbers provide less task granularity, but also put less strain on the cluster NameNode.</td>
</tr>
<tr>
<td>dfs.replication</td>
<td>This determines how many copies of each block to store for durability. For small clusters, we set this to 2 because the cluster is small and easy to restart in case of data loss. You can change the setting to 1, 2, or 3 as your needs dictate. Amazon EMR automatically calculates the replication factor based on cluster size. To overwrite the default value, use a configure-hadoop bootstrap action.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Task Configuration (Hadoop 1.0.3)

Topics

- Tasks per Machine (p. 575)
- Tasks per Job (AMI 2.3) (p. 577)
- Task JVM Settings (AMI 2.3) (p. 577)
- Avoiding Cluster Slowdowns (AMI 2.3) (p. 578)

There are a number of configuration variables for tuning the performance of your MapReduce jobs. This section describes some of the important task-related settings.

Tasks per Machine

Two configuration options determine how many tasks are run per node, one for mappers and the other for reducers. They are:

- mapred.tasktracker.map.tasks.maximum
- mapred.tasktracker.reduce.tasks.maximum

Amazon EMR provides defaults that are entirely dependent on the EC2 instance type. The following table shows the default settings for clusters launched with AMIs after 2.4.6.

<table>
<thead>
<tr>
<th>EC2 Instance Name</th>
<th>Mappers</th>
<th>Reducers</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1.small</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>m1.medium</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>m1.large</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>m1.xlarge</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>m2.xlarge</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>m2.2xlarge</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>m2.4xlarge</td>
<td>14</td>
<td>4</td>
</tr>
<tr>
<td>EC2 Instance Name</td>
<td>Mappers</td>
<td>Reducers</td>
</tr>
<tr>
<td>------------------</td>
<td>---------</td>
<td>----------</td>
</tr>
<tr>
<td>m3.xlarge</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>m3.2xlarge</td>
<td>12</td>
<td>4</td>
</tr>
<tr>
<td>c1.medium</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>c1.xlarge</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>c3.xlarge</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>c3.2xlarge</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>c3.4xlarge</td>
<td>12</td>
<td>4</td>
</tr>
<tr>
<td>c3.8xlarge</td>
<td>24</td>
<td>8</td>
</tr>
<tr>
<td>cc2.8xlarge</td>
<td>24</td>
<td>6</td>
</tr>
<tr>
<td>d2.xlarge</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>d2.2xlarge</td>
<td>12</td>
<td>4</td>
</tr>
<tr>
<td>d2.4xlarge</td>
<td>24</td>
<td>8</td>
</tr>
<tr>
<td>d2.8xlarge</td>
<td>54</td>
<td>18</td>
</tr>
<tr>
<td>hi1.4xlarge</td>
<td>24</td>
<td>6</td>
</tr>
<tr>
<td>hs1.8xlarge</td>
<td>24</td>
<td>6</td>
</tr>
<tr>
<td>cg1.4xlarge</td>
<td>12</td>
<td>3</td>
</tr>
<tr>
<td>g2.2xlarge</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>i2.xlarge</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>i2.2xlarge</td>
<td>12</td>
<td>4</td>
</tr>
<tr>
<td>i2.4xlarge</td>
<td>24</td>
<td>8</td>
</tr>
<tr>
<td>i2.8xlarge</td>
<td>48</td>
<td>16</td>
</tr>
<tr>
<td>r3.xlarge</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>r3.2xlarge</td>
<td>12</td>
<td>4</td>
</tr>
<tr>
<td>r3.4xlarge</td>
<td>12</td>
<td>4</td>
</tr>
<tr>
<td>r3.8xlarge</td>
<td>24</td>
<td>8</td>
</tr>
</tbody>
</table>

**Note**
The number of default mappers is based on the memory available on each EC2 instance type. If you increase the default number of mappers, you also need to modify the task JVM settings to decrease the amount of memory allocated to each task. Failure to modify the JVM settings appropriately could result in out of memory errors.
Tasks per Job (AMI 2.3)

When your cluster runs, Hadoop creates a number of map and reduce tasks. These determine the number of tasks that can run simultaneously during your cluster. Run too few tasks and you have nodes sitting idle; run too many and there is significant framework overhead.

Amazon EMR determines the number of map tasks from the size and number of files of your input data. You configure the reducer setting. There are four settings you can modify to adjust the reducer setting.

The parameters for configuring the reducer setting are described in the following table.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mapred.map.tasks</td>
<td>Target number of map tasks to run. The actual number of tasks created is sometimes different than this number.</td>
</tr>
<tr>
<td>mapred.map.tasksperslot</td>
<td>Target number of map tasks to run as a ratio to the number of map slots in the cluster. This is used if <code>mapred.map.tasks</code> is not set.</td>
</tr>
<tr>
<td>mapred.reduce.tasks</td>
<td>Number of reduce tasks to run.</td>
</tr>
<tr>
<td>mapred.reduce.tasksperslot</td>
<td>Number of reduce tasks to run as a ratio of the number of reduce slots in the cluster.</td>
</tr>
</tbody>
</table>

The two `tasksperslot` parameters are unique to Amazon EMR. They only take effect if `mapred.*.tasks` is not defined. The order of precedence is:

1. `mapred.map.tasks` set by the Hadoop job
2. `mapred.map.tasks` set in `mapred-conf.xml` on the master node
3. `mapred.map.tasksperslot` if neither of those are defined

Task JVM Settings (AMI 2.3)

You can configure the amount of heap space for tasks as well as other JVM options with the `mapred.child.java.opts` setting. Amazon EMR provides a default `-Xmx` value in this location, with the defaults per instance type shown in the following table.

<table>
<thead>
<tr>
<th>Amazon EC2 Instance Name</th>
<th>Default JVM value</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1.small</td>
<td>-Xmx288m</td>
</tr>
<tr>
<td>m1.medium</td>
<td>-Xmx576m</td>
</tr>
<tr>
<td>m1.large</td>
<td>-Xmx864m</td>
</tr>
<tr>
<td>m1.xlarge</td>
<td>-Xmx768m</td>
</tr>
<tr>
<td>c1.medium</td>
<td>-Xmx288m</td>
</tr>
<tr>
<td>c1.xlarge</td>
<td>-Xmx384m</td>
</tr>
<tr>
<td>m2.xlarge</td>
<td>-Xmx2304m</td>
</tr>
<tr>
<td>m2.2xlarge</td>
<td>-Xmx2688m</td>
</tr>
</tbody>
</table>
You can start a new JVM for every task, which provides better task isolation, or you can share JVMs between tasks, providing lower framework overhead. If you are processing many small files, it makes sense to reuse the JVM many times to amortize the cost of start-up. However, if each task takes a long time or processes a large amount of data, then you might choose to not reuse the JVM to ensure all memory is freed for subsequent tasks.

Use the `mapred.job.reuse.jvm.num.tasks` option to configure the JVM reuse settings.

**Note**
Amazon EMR sets the value of `mapred.job.reuse.jvm.num.tasks` to 20, but you can override it with a bootstrap action. A value of -1 means infinite reuse within a single job, and 1 means do not reuse tasks.

### Avoiding Cluster Slowdowns (AMI 2.3)

In a distributed environment, you are going to experience random delays, slow hardware, failing hardware, and other problems that collectively slow down your cluster. This is known as the *stragglers* problem. Hadoop has a feature called *speculative execution* that can help mitigate this issue. As the cluster progresses, some machines complete their tasks. Hadoop schedules tasks on nodes that are free. Whichever task finishes first is the successful one, and the other tasks are killed. This feature can substantially cut down on the run time of jobs. The general design of a mapreduce algorithm is such that the processing of map tasks is meant to be idempotent. However, if you are running a job where the task execution has side effects (for example, a zero reducer job that calls an external resource), it is important to disable speculative execution.

You can enable speculative execution for mappers and reducers independently. By default, Amazon EMR enables it for mappers and reducers in AMI 2.3. You can override these settings with a bootstrap action. For more information about using bootstrap actions, see *(Optional) Create Bootstrap Actions to Install Additional Software* (p. 129).

### Speculative Execution Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>mapred.map.tasks.speculative.execution</td>
<td>true</td>
</tr>
<tr>
<td>mapred.reduce.tasks.speculative.execution</td>
<td>true</td>
</tr>
</tbody>
</table>

### Intermediate Compression (Hadoop 1.0.3)

Hadoop sends data between the mappers and reducers in its shuffle process. This network operation is a bottleneck for many clusters. To reduce this bottleneck, Amazon EMR enables intermediate data compression by default on intermediate blocks. This enables you to increase the maximum memory for intermediate blocks by increasing the value of the `mapred.reduce.shuffle.filename.size` configuration parameter.
compression by default. Because it provides a reasonable amount of compression with only a small CPU impact, we use the Snappy codec.

You can modify the default compression settings with a bootstrap action. For more information about using bootstrap actions, see (Optional) Create Bootstrap Actions to Install Additional Software (p. 129).

The following table presents the default values for the parameters that affect intermediate compression.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>mapred.compress.map.output</td>
<td>true</td>
</tr>
<tr>
<td>mapred.map.output.compression.codec</td>
<td>org.apache.hadoop.io.compress.SnappyCodec</td>
</tr>
</tbody>
</table>

### Hadoop 20.205 Default Configuration (Deprecated)

This section describes the default configuration settings that Amazon EMR uses to configure a Hadoop cluster launched with Hadoop 20.205. For more information about the AMI versions supported by Amazon EMR, see Choose an Amazon Machine Image (AMI) (p. 69).

#### Hadoop Configuration (Hadoop 20.205)

The following Amazon EMR default configuration settings for clusters launched with Amazon EMR AMI 2.0 or 2.1 are appropriate for most workloads.

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.

The following tables list the default configuration settings for each EC2 instance type in clusters launched with the Amazon EMR AMI version 2.0 or 2.1. For more information about the AMI versions supported by Amazon EMR, see Choose an Amazon Machine Image (AMI) (p. 69).

#### m1.small

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>768</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>256</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>256</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>128</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx384m</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>2</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>1</td>
</tr>
</tbody>
</table>

**m1.medium**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>1536</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>512</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>256</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>256</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx768m</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>2</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>1</td>
</tr>
</tbody>
</table>

**m1.large**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>3072</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>1024</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>512</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>512</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx1152m</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>3</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>1</td>
</tr>
</tbody>
</table>

**m1.xlarge**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>9216</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>3072</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>512</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>512</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx1024m</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>8</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>3</td>
</tr>
</tbody>
</table>
### c1.medium

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>768</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>256</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>256</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>128</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx384m</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>2</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>1</td>
</tr>
</tbody>
</table>

### c1.xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>3072</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>1024</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>512</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>512</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx512m</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>7</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>2</td>
</tr>
</tbody>
</table>

### m2.xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>12288</td>
</tr>
<tr>
<td>HADOOP_NAMENODE_HEAPSIZE</td>
<td>4096</td>
</tr>
<tr>
<td>HADOOP_TASKTRACKER_HEAPSIZE</td>
<td>512</td>
</tr>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>512</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx3072m</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>3</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>1</td>
</tr>
</tbody>
</table>

### m2.2xlarge

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_JOBTRACKER_HEAPSIZE</td>
<td>24576</td>
</tr>
</tbody>
</table>

Parameter | Value
--- | ---
HADOOP\_NAMENODE\_HEAPSIZE | 8192
HADOOP\_TASKTRACKER\_HEAPSIZE | 512
HADOOP\_DATANODE\_HEAPSIZE | 512
mapred.child.java.opts | -Xmx3584m
mapred.tasktracker.map.tasks.maximum | 6
mapred.tasktracker.reduce.tasks.maximum | 2

**m2.4xlarge**

Parameter | Value
--- | ---
HADOOP\_JOBTRACKER\_HEAPSIZE | 49152
HADOOP\_NAMENODE\_HEAPSIZE | 16384
HADOOP\_TASKTRACKER\_HEAPSIZE | 512
HADOOP\_DATANODE\_HEAPSIZE | 512
mapred.child.java.opts | -Xmx3072m
mapred.tasktracker.map.tasks.maximum | 14
mapred.tasktracker.reduce.tasks.maximum | 4

**cc2.8xlarge**

Parameter | Value
--- | ---
HADOOP\_JOBTRACKER\_HEAPSIZE | 40152
HADOOP\_NAMENODE\_HEAPSIZE | 16384
HADOOP\_TASKTRACKER\_HEAPSIZE | 512
HADOOP\_DATANODE\_HEAPSIZE | 512
mapred.child.java.opts | -Xmx2048m
mapred.tasktracker.map.tasks.maximum | 24
mapred.tasktracker.reduce.tasks.maximum | 6

**cg1.4xlarge**

Parameter | Value
--- | ---
HADOOP\_JOBTRACKER\_HEAPSIZE | 10240
HADOOP\_NAMENODE\_HEAPSIZE | 5120
HADOOP\_TASKTRACKER\_HEAPSIZE | 512
### HDFS Configuration (Hadoop 20.205)

The following table describes the default Hadoop Distributed File System (HDFS) parameters and their settings.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADOOP_DATANODE_HEAPSIZE</td>
<td>512</td>
</tr>
<tr>
<td>mapred.child.java.opts</td>
<td>-Xmx1152m</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>12</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>3</td>
</tr>
</tbody>
</table>

### Task Configuration (Hadoop 20.205)

#### Topics
- Tasks per Machine (p. 583)
- Tasks per Job (AMI 2.0 and 2.1) (p. 584)
- Task JVM Settings (AMI 2.0 and 2.1) (p. 585)
- Avoiding Cluster Slowdowns (AMI 2.0 and 2.1) (p. 585)

There are a number of configuration variables for tuning the performance of your MapReduce jobs. This section describes some of the important task-related settings.

#### Tasks per Machine

Two configuration options determine how many tasks are run per node, one for mappers and the other for reducers. They are:

- mapred.tasktracker.map.tasks.maximum
- mapred.tasktracker.reduce.tasks.maximum

Amazon EMR provides defaults that are entirely dependent on the EC2 instance type. The following table shows the default settings for clusters launched with AMI 2.0 or 2.1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>dfs.block.size</td>
<td>The size of HDFS blocks. When operating on data stored in HDFS, the split size is generally the size of an HDFS block. Larger numbers provide less task granularity, but also put less strain on the cluster NameNode.</td>
<td>134217728 (128 MB)</td>
</tr>
<tr>
<td>dfs.replication</td>
<td>This determines how many copies of each block to store for durability. For small clusters, we set this to 2 because the cluster is small and easy to restart in case of data loss. You can change the setting to 1, 2, or 3 as your needs dictate. Amazon EMR automatically calculates the replication factor based on cluster size. To overwrite the default value, use a configure-hadoop bootstrap action.</td>
<td>1 for clusters &lt; four nodes&lt;br&gt;2 for clusters &lt; ten nodes&lt;br&gt;3 for all other clusters</td>
</tr>
</tbody>
</table>
Amazon EC2 Instance Name | Mappers | Reducers
--- | --- | ---
m1.small | 2 | 1
m1.medium | 2 | 1
m1.large | 3 | 1
m1.xlarge | 8 | 3
c1.medium | 2 | 1
c1.xlarge | 7 | 2
m2.xlarge | 3 | 1
m2.2xlarge | 6 | 2
m2.4xlarge | 14 | 4
cc2.8xlarge | 24 | 6
cg1.4xlarge | 12 | 3

**Note**
The number of default mappers is based on the memory available on each EC2 instance type. If you increase the default number of mappers, you also need to modify the task JVM settings to decrease the amount of memory allocated to each task. Failure to modify the JVM settings appropriately could result in *out of memory* errors.

**Tasks per Job (AMI 2.0 and 2.1)**

When your cluster runs, Hadoop creates a number of map and reduce tasks. These determine the number of tasks that can run simultaneously during your cluster. Run too few tasks and you have nodes sitting idle; run too many and there is significant framework overhead.

Amazon EMR determines the number of map tasks from the size and number of files of your input data. You configure the reducer setting. There are four settings you can modify to adjust the reducer setting.

The parameters for configuring the reducer setting are described in the following table.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mapred.map.tasks</td>
<td>Target number of map tasks to run. The actual number of tasks created is sometimes different than this number.</td>
</tr>
<tr>
<td>mapred.map.tasksperslot</td>
<td>Target number of map tasks to run as a ratio to the number of map slots in the cluster. This is used if mapred.map.tasks is not set.</td>
</tr>
<tr>
<td>mapred.reduce.tasks</td>
<td>Number of reduce tasks to run.</td>
</tr>
<tr>
<td>mapred.reduce.tasksperslot</td>
<td>Number of reduce tasks to run as a ratio of the number of reduce slots in the cluster.</td>
</tr>
</tbody>
</table>

The two tasksperslot parameters are unique to Amazon EMR. They only take effect if mapred.*.tasks is not defined. The order of precedence is:
1. `mapred.map.tasks` set by the Hadoop job
2. `mapred.map.tasks` set in `mapred-conf.xml` on the master node
3. `mapred.map.tasks` set per slot if neither of the above are defined

## Task JVM Settings (AMI 2.0 and 2.1)

You can configure the amount of heap space for tasks as well as other JVM options with the `mapred.child.java.opts` setting. Amazon EMR provides a default `-Xmx` value in this location, with the defaults per instance type shown in the following table.

<table>
<thead>
<tr>
<th>Amazon EC2 Instance Name</th>
<th>Default JVM value</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1.small</td>
<td>-Xmx384m</td>
</tr>
<tr>
<td>m1.medium</td>
<td>-Xmx768m</td>
</tr>
<tr>
<td>m1.large</td>
<td>-Xmx1152m</td>
</tr>
<tr>
<td>m1.xlarge</td>
<td>-Xmx1024m</td>
</tr>
<tr>
<td>c1.medium</td>
<td>-Xmx384m</td>
</tr>
<tr>
<td>c1.xlarge</td>
<td>-Xmx512m</td>
</tr>
<tr>
<td>m2.xlarge</td>
<td>-Xmx3072m</td>
</tr>
<tr>
<td>m2.2xlarge</td>
<td>-Xmx3584m</td>
</tr>
<tr>
<td>m2.4xlarge</td>
<td>-Xmx3072m</td>
</tr>
<tr>
<td>cc2.8xlarge</td>
<td>-Xmx2048m</td>
</tr>
<tr>
<td>cg1.4xlarge</td>
<td>-Xmx1152m</td>
</tr>
</tbody>
</table>

You can start a new JVM for every task, which provides better task isolation, or you can share JVMs between tasks, providing lower framework overhead. If you are processing many small files, it makes sense to reuse the JVM many times to amortize the cost of start-up. However, if each task takes a long time or processes a large amount of data, then you might choose to not reuse the JVM to ensure all memory is freed for subsequent tasks.

Use the `mapred.job.reuse.jvm.num.tasks` option to configure the JVM reuse settings.

**Note**

Amazon EMR sets the value of `mapred.job.reuse.jvm.num.tasks` to 20, but you can override it with a bootstrap action. A value of -1 means infinite reuse within a single job, and 1 means do not reuse tasks.

## Avoiding Cluster Slowdowns (AMI 2.0 and 2.1)

In a distributed environment, you are going to experience random delays, slow hardware, failing hardware, and other problems that collectively slow down your cluster. This is known as the *stragglers* problem. Hadoop has a feature called *speculative execution* that can help mitigate this issue. As the cluster progresses, some machines complete their tasks. Hadoop schedules tasks on nodes that are free. Whichever task finishes first is the successful one, and the other tasks are killed. This feature can substantially cut down on the run time of jobs. The general design of a mapreduce algorithm is such that the processing of map tasks is meant to be idempotent. However, if you are running a job where the task
execution has side effects (for example, a zero reducer job that calls an external resource), it is important to disable speculative execution.

You can enable speculative execution for mappers and reducers independently. By default, Amazon EMR enables it for mappers and reducers in AMI 2.0 or 2.1. You can override these settings with a bootstrap action. For more information about using bootstrap actions, see (Optional) Create Bootstrap Actions to Install Additional Software (p. 129).

### Speculative Execution Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>mapred.map.tasks.speculative.execution</td>
<td>true</td>
</tr>
<tr>
<td>mapred.reduce.tasks.speculative.execution</td>
<td>true</td>
</tr>
</tbody>
</table>

### Intermediate Compression (Hadoop 20.205)

Hadoop sends data between the mappers and reducers in its shuffle process. This network operation is a bottleneck for many clusters. To reduce this bottleneck, Amazon EMR enables intermediate data compression by default. Because it provides a reasonable amount of compression with only a small CPU impact, we use the Snappy codec.

You can modify the default compression settings with a bootstrap action. For more information about using bootstrap actions, see (Optional) Create Bootstrap Actions to Install Additional Software (p. 129).

The following table presents the default values for the parameters that affect intermediate compression.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>mapred.compress.map.output</td>
<td>true</td>
</tr>
<tr>
<td>mapred.map.output.compression.codec</td>
<td>org.apache.hadoop.io.compress.SnappyCodec</td>
</tr>
</tbody>
</table>
Command Line Interface Reference for Amazon EMR

The Amazon EMR command line interface (CLI) is a tool you can use to launch and manage clusters from the command line.

**Note**
The Amazon EMR CLI is no longer under feature development. Customers are encouraged to use the Amazon EMR commands in the AWS CLI instead.

The AWS Command Line Interface version 1.4 provides support for Amazon EMR. We recommend you download and install the AWS CLI instead of using the Amazon EMR CLI. For more information, see [http://aws.amazon.com/cli/](http://aws.amazon.com/cli/).

**Topics**
- Specifying Parameter Values in AWS CLI for Amazon EMR (p. 587)
- Install the Amazon EMR Command Line Interface (Deprecated) (p. 588)
- How to Call the Command Line Interface (Deprecated) (p. 594)
- AWS EMR Command Line Interface Options (Deprecated) (p. 594)
- AWS EMR Command Line Interface Releases (Deprecated) (p. 645)

Specifying Parameter Values in AWS CLI for Amazon EMR

You can specify values for parameters supplied with the Amazon EMR subcommands create-cluster, ssh, get, put, and socks. You can set the value of the parameter by using `aws configure` or by setting the values in your `~/.aws/config` or `C:\Users\USERNAME\.aws\config` files. The following tables show the subcommands and parameters that can be set.

### create-cluster

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>instance_profile</td>
<td>The instance profile you want Amazon EMR to use to run application on the cluster's Amazon EC2 instances.</td>
</tr>
<tr>
<td>service_role</td>
<td>The service role you want the Amazon EMR service to use.</td>
</tr>
<tr>
<td>log_uri</td>
<td>The Amazon S3 URI you want Amazon EMR to place cluster logs.</td>
</tr>
<tr>
<td>key_name</td>
<td>The EC2 key pair name you want to use to access the EMR cluster.</td>
</tr>
<tr>
<td>enable_debugging</td>
<td>The Boolean value that indicates if you want to enable debugging when creating a cluster.</td>
</tr>
</tbody>
</table>
### ssh, get, put, socks

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key_pair_file</td>
<td>The path to the private key pair file you use to connect to the EMR cluster.</td>
</tr>
</tbody>
</table>

### Setting Parameters with the Command Line

To set a parameter, you can use the command `aws configure set emr.parameter_name value`. For example, set the value of the `key_name` to `myKeyName` use the following:

```
% aws configure set emr.key_name myKeyName
```

### Displaying Parameter Values with the Command Line

You can also display a value for a given parameter using `aws configure get emr.parameter_name value`. For example, to get the value of the `key_name` you just set, use the following command and `myKeyName` will be displayed:

```
% aws configure get emr.key_name
myKeyName
```

### Setting Parameters with the Configuration File

To set parameters using the configuration file, you specify the service and then key-value assignments in the AWS CLI configuration file. For example, on Linux, Mac OS X, Unix systems this is located at `~/.aws/config` or at `C:\Users\USERNAME\.aws\config` on Windows systems. A sample configuration looks like:

```
[dynamodb]
region = us-east-1
dynamodb =
    service_role = MY_DEFAULT_DYNAMODB_ROLE
    instance_profile = MY_EC2_DYNAMODB_ROLE
    log_uri = s3://myBucket/logs
enable_debugging = True
dynamodb_key_name = myKeyName
dynamodb_key_pair_file = /home/myUser/myKeyName.pem
```

**Note**

If you create roles for Amazon EMR using `aws emr create-default-roles` they will be automatically be populated in the configuration file.

---

### Install the Amazon EMR Command Line Interface (Deprecated)

**Note**

The Amazon EMR CLI is no longer under feature development. Customers are encouraged to use the Amazon EMR commands in the AWS CLI instead.
The AWS Command Line Interface version 1.4 provides support for Amazon EMR. We recommend you download and install the AWS CLI instead of using the Amazon EMR CLI. For more information, see http://aws.amazon.com/cli/.

To install the Amazon EMR command line interface, complete the following tasks:

Topics
- Installing Ruby (p. 589)
- Verifying the RubyGems package management framework (p. 589)
- Installing the Amazon EMR Command Line Interface (p. 590)
- Configuring Credentials (p. 590)
- SSH Credentials (p. 593)

Installing Ruby

The Amazon EMR CLI works with versions 1.8.7, 1.9.3, and 2.0. If your machine does not have Ruby installed, download one of those versions for use with the CLI.

To install Ruby

1. Download and install Ruby:
   - Windows users can install the Ruby versions from http://rubyinstaller.org/downloads/. During the installation process, select the check boxes to add Ruby executables to your PATH environment variable and to associate .rb files with this Ruby installation.
   - Mac OS X comes with Ruby installed. You can check the version as shown in the following step.

2. Verify that Ruby is running by typing the following at the command prompt:

   ruby -v

   The Ruby version is shown, confirming that you installed Ruby. The output should be similar to the following:

   ruby 1.8.7 (2012-02-08 patchlevel 358) [universal-darwin11.0]

Verifying the RubyGems package management framework

The Amazon EMR CLI requires RubyGems version 1.8 or later.

To verify the RubyGems installation and version

- To check whether RubyGems is installed, run the following command from a terminal window. If RubyGems is installed, this command displays its version information.

   gem -v
If you don’t have RubyGems installed, download and install RubyGems before you can install the Amazon EMR CLI.

**To install RubyGems on Linux/Unix/Mac OS**

1. Download and extract RubyGems version 1.8 or later from RubyGems.org.
2. Install RubyGems using the following command.

```
sudo ruby setup.rb
```

**Installing the Amazon EMR Command Line Interface**

**To download the Amazon EMR CLI**

1. Create a new directory to install the Amazon EMR CLI into. From the command-line prompt, enter the following:

```
mkdir elastic-mapreduce-cli
```

2. Download the Amazon EMR files:
   b. Click Download.
   c. Save the file in your newly created directory.

**To install the Amazon EMR CLI**

1. Navigate to your `elastic-mapreduce-cli` directory.
2. Unzip the compressed file:
   - Linux, UNIX, and Mac OS X users, from the command-line prompt, enter the following:
     
     ```
     unzip elastic-mapreduce-ruby.zip
     ```
   - Windows users, from Windows Explorer, open the `elastic-mapreduce-ruby.zip` file and select **Extract all files**.

**Configuring Credentials**

The Amazon EMR credentials file can provide information required for many commands. You can also store command parameters in the file so you don’t have to repeatedly enter that information at the command line each time you create a cluster.

Your credentials are used to calculate the signature value for every request you make. Amazon EMR automatically looks for your credentials in the file `credentials.json`. It is convenient to edit the `credentials.json` file and include your AWS credentials. An AWS key pair is a security credential similar to a password, which you use to securely connect to your instance when it is running. We recommend that you create a new key pair to use with this guide.

**To create your credentials file**

1. Create a file named `credentials.json` in the directory where you unzipped the Amazon EMR CLI.
2. Add the following lines to your credentials file:
Configuring Credentials

```json
{
  "access_id": "Your AWS Access Key ID",
  "private_key": "Your AWS Secret Access Key",
  "key-pair": "Your key pair name",
  "key-pair-file": "The path and name of your PEM file",
  "log_uri": "A path to a bucket you own on Amazon S3, such as, s3n://mylog-uri/",
  "region": "The region of your cluster, either us-east-1, us-west-2, us-west-1, eu-west-1, eu-central-1, ap-northeast-1, ap-southeast-1, ap-southeast-2, or sa-east-1"
}
```

Note the name of the region. You use this region to create your Amazon EC2 key pair and your Amazon S3 bucket. For more information about regions supported by Amazon EMR, see Regions and Endpoints in the Amazon Web Services General Reference.

The next sections explain how to create and find your credentials.

**AWS Security Credentials**

AWS uses security credentials to help protect your data. This section shows you how to view your security credentials so you can add them to your credentials.json file.

For CLI access, you need an access key ID and secret access key. Use IAM user access keys instead of AWS account root user access keys. IAM lets you securely control access to AWS services and resources in your AWS account. For more information about creating access keys, see How Do I Get Security Credentials? in the AWS General Reference.

Set your access_id parameter to the value of your access key ID and set your private_key parameter to the value of your secret access key.

**To create an Amazon EC2 key pair**

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. From the EC2 Dashboard, select the region you used in your credentials.json file, then click Key Pair.
3. On the Key Pairs page, click Create Key Pair.
4. Enter a name for your key pair, such as, mykeypair.
5. Click Create.
6. Save the resulting PEM file in a safe location.

In your credentials.json file, change the key-pair parameter to your Amazon EC2 key pair name and change the key-pair-file parameter to the location and name of your PEM file. This PEM file is what the CLI uses as the default for the Amazon EC2 key pair for the EC2 instances it creates when it launches a cluster.

**Amazon S3 Bucket**

The log-uri parameter specifies a location in Amazon S3 for the Amazon EMR results and log files from your cluster. The value of the log-uri parameter is an Amazon S3 bucket that you create for this purpose.

**To create an Amazon S3 bucket**

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Click **Create Bucket**.
3. In the **Create a Bucket** dialog box, enter a bucket name, such as `mylog-uri`.
   
   This name should be globally unique, and cannot be the same name used by another bucket. For more information about valid bucket names, see **Bucket Restrictions** in the *Amazon Simple Storage Service Developer Guide*.
4. For **Region**, choose the bucket region.

<table>
<thead>
<tr>
<th>Amazon EMR region</th>
<th>Amazon S3 region</th>
</tr>
</thead>
<tbody>
<tr>
<td>US East (N. Virginia)</td>
<td>US Standard</td>
</tr>
<tr>
<td>US West (Oregon)</td>
<td>Oregon</td>
</tr>
<tr>
<td>US West (N. California)</td>
<td>Northern California</td>
</tr>
<tr>
<td>EU (Ireland)</td>
<td>Ireland</td>
</tr>
<tr>
<td>EU (Frankfurt)</td>
<td>Frankfurt</td>
</tr>
<tr>
<td>Asia Pacific (Tokyo)</td>
<td>Japan</td>
</tr>
<tr>
<td>Asia Pacific (Singapore)</td>
<td>Singapore</td>
</tr>
<tr>
<td>Asia Pacific (Sydney)</td>
<td>Sydney</td>
</tr>
<tr>
<td>South America (São Paulo)</td>
<td>Sao Paulo</td>
</tr>
<tr>
<td>AWS GovCloud (US)</td>
<td>GovCloud</td>
</tr>
</tbody>
</table>

**Note**

To use the AWS GovCloud (US) region, contact your AWS business representative. You can't create an AWS GovCloud (US) account on the AWS website. You must engage directly with AWS and sign an AWS GovCloud (US) Enterprise Agreement. For more information, see the [AWS GovCloud (US) product page](https://aws.amazon.com/govcloud-us/).  

5. Click **Create**.

**Note**

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

You have created a bucket with the URI `s3://mylog-uri/`.

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

**To set permissions on an Amazon S3 bucket**

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

You have now created a bucket and assigned it permissions. Set your log-uri parameter to this bucket’s URI as the location for Amazon EMR to upload your logs and results.

**SSH Credentials**

Configure your SSH credentials for use with either SSH or PuTTY. This step is required.

**To configure your SSH credentials**

- Configure your computer to use SSH:
  - Linux, UNIX, and Mac OS X users, set the permissions on the PEM file for your Amazon EC2 key pair. For example, if you saved the file as `mykeypair.pem`, the command looks like:
    ```bash
    chmod og-rwx mykeypair.pem
    ```
  - Windows users
    a. Windows users use PuTTY to connect to the master node. Download PuTTYgen.exe to your computer from [http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html](http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html).
    b. Launch PuTTYgen.
    c. Click Load. Select the PEM file you created earlier.
    d. Click Open.
    e. Click OK on the PuTTYgen Notice telling you the key was successfully imported.
    f. Click Save private key to save the key in the PPK format.
    g. When PuTTYgen prompts you to save the key without a pass phrase, click Yes.
    h. Enter a name for your PuTTY private key, such as, `mykeypair.ppk`.
    i. Click Save.
    j. Exit the PuTTYgen application.

**Verify installation of the Amazon EMR CLI**

- In the directory where you installed the Amazon EMR CLI, type the following command.
  - Linux, UNIX, and Mac OS X users:
    ```bash
    ./elastic-mapreduce --version
    ```
  - Windows users:
    ```bash
    ruby elastic-mapreduce --version
    ```

If the CLI is correctly installed and the credentials properly configured, the CLI should display its version number represented as a date. The output should look similar to the following:

```
Version 2012-12-17
```
How to Call the Command Line Interface (Deprecated)

Note
The Amazon EMR CLI is no longer under feature development. Customers are encouraged to use the Amazon EMR commands in the AWS CLI instead.

The syntax that you use to run the command line interface (CLI) differs slightly depending on the operating system you use. In the following examples, commands are issued in either a terminal (Linux, UNIX, and Mac OS X) or a command (Windows) interface. Both examples assume that you are running the commands from the directory where you unzipped the Amazon EMR CLI.

In the Linux/UNIX/Mac OS X version of the CLI call, you use a period and slash (./) to indicate that the script is located in the current directory. The operating system automatically detects that the script is a Ruby script and uses the correct libraries to interpret the script. In the Windows version of the call, using the current directory is implied, but you have to explicitly specify which scripting engine to use by prefixing the call with "ruby".

Aside from the preceding operating-system–specific differences in how you call the CLI Ruby script, the way you pass options to the CLI is the same. In the directory where you installed the Amazon EMR CLI, issue commands in one of the following formats, depending on your operating system.

- Linux, UNIX, and Mac OS X users:

  ./elastic-mapreduce Options

- Windows users:

  ruby elastic-mapreduce Options

AWS EMR Command Line Interface Options (Deprecated)

Note
The Amazon EMR CLI is no longer under feature development. Customers are encouraged to use the Amazon EMR commands in the AWS CLI instead.

The Amazon EMR command line interface (CLI) supports the following options, arranged according to function. Options that fit into more than one category are listed multiple times.

Topics
- Common Options (p. 595)
- Uncommon Options (p. 597)
- Options Common to All Step Types (p. 597)
- Adding and Modifying Instance Groups (p. 597)
- Adding JAR Steps to Job Flows (p. 599)
- Adding JSON Steps to Job Flows (p. 601)
- Adding Streaming Steps to Job Flows (p. 601)
- Assigning an Elastic IP Address to the Master Node (p. 604)
- Connecting to the Master Node (p. 605)
- Creating Job Flows (p. 606)
Common Options

--access-id ACCESS_ID

Sets the AWS access identifier.

Shortcut: -a ACCESS_ID

--credentials CREDENTIALS_FILE

Specifies the credentials file that contains the AWS access identifier and the AWS private key to use when contacting Amazon EMR.

Shortcut: -c CREDENTIALS_FILE

For CLI access, you need an access key ID and secret access key. Use IAM user access keys instead of AWS account root user access keys. IAM lets you securely control access to AWS services and resources in your AWS account. For more information about creating access keys, see How Do I Get Security Credentials? in the AWS General Reference.

--help

Displays help information from the CLI.

Shortcut: -h

--http-proxy HTTP_PROXY

HTTP proxy server address host[:port].

--http-proxy-user USER

The username supplied to the HTTP proxy.

--http-proxy-pass PASS

The password supplied to the HTTP proxy.

--jobflow JOB_FLOW_IDENTIFIER

Specifies the cluster with the given cluster identifier.

Shortcut: -j JOB_FLOW_IDENTIFIER

--log-uri

Specifies the Amazon S3 bucket to receive log files. Used with --create.

--private-key PRIVATE_KEY

Specifies the AWS private key to use when contacting Amazon EMR.
Shortcut: -p PRIVATE_KEY

--trace
Traces commands made to the web service.

--verbose
Turns on verbose logging of program interaction.

--version
Displays the version of the CLI.

Shortcut: -v

To archive log files to Amazon S3

- Set the --log-uri argument when you launch the cluster and specify a location in Amazon S3. Alternatively, you can set this value in the credentials.json file that you configured for the CLI. This causes all of the clusters you launch with the CLI to archive log files to the specified Amazon S3 bucket. For more information about credentials.json, see "Configuring Credentials" in Install the Amazon EMR Command Line Interface (Deprecated) (p. 588). The following example illustrates creating a cluster that archives log files to Amazon S3. Replace mybucket with the name of your bucket.

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --log-uri s3://mybucket
```

- Windows users:

```
ruby elastic-mapreduce --create --log-uri s3://mybucket
```

To aggregate logs in Amazon S3

- Log aggregation in Hadoop 2.x compiles logs from all containers for an individual application into a single file. This option is only available on Hadoop 2.x AMIs. To enable log aggregation to Amazon S3 using the Amazon EMR CLI, you use a bootstrap action at cluster launch to enable log aggregation and to specify the bucket to store the logs.

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --alive --master-instance-type m1.xlarge --slave-instance-type m1.xlarge
  --num-instances 1 --ami-version 3.3 --bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hadoop --args
  "-y,yarn.log-aggregation-enable=true,-y,yarn.log-aggregation.retain-seconds=-1,-y,yarn.log-aggregation.retain-check-interval-seconds=3000,-y,yarn.nodemanager.remote-app-log-dir=s3://mybucket/logs" 
  --ssh --name "log aggregation sub-bucket name"
```

- Windows users:

```
ruby elastic-mapreduce --create --alive --master-instance-type m1.xlarge --slave-instance-type m1.xlarge --num-instances 1 --ami-version 3.3 --bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hadoop --args "-y,yarn.log-aggregation-enable=true,-y,yarn.log-aggregation.retain-seconds=-1,-y,yarn.log-aggregation.retain-check-interval-seconds=3000,-y,yarn.nodemanager.remote-app-log-dir=s3://mybucket/logs" --ssh --name "log aggregation sub-bucket name"
```
Uncommon Options

--apps-path APPLICATION_PATH

Specifies the Amazon S3 path to the base of the Amazon EMR bucket to use, for example: s3://elasticmapreduce.

--endpoint ENDPOINT

Specifies the Amazon EMR endpoint to connect to.

--debug

Prints stack traces when exceptions occur.

Options Common to All Step Types

--no-wait

Don't wait for the master node to start before executing SCP or SSH, or assigning an elastic IP address.

--key-pair-file FILE_PATH

The path to the local PEM file of the Amazon EC2 key pair to set as the connection credential when you launch the cluster.

Adding and Modifying Instance Groups

--add-instance-group INSTANCE_ROLE

Adds an instance group to an existing cluster. The role may be task only.

--modify-instance-group INSTANCE_GROUP_ID

Modifies an existing instance group.

--add-instance-group INSTANCE_ROLE

Adds an instance group to an existing cluster. The role may be task only.

To launch an entire cluster with Spot Instances using the Amazon EMR CLI

To specify that an instance group should be launched as Spot Instances, use the --bid-price parameter. The following example shows how to create a cluster where the master, core, and task instance groups are all running as Spot Instances. The following code launches a cluster only after until the requests for the master and core instances have been completely fulfilled.

- In the directory where you installed the Amazon EMR CLI, type the following command.
- Linux, UNIX, and Mac OS X users:

```bash
./elastic-mapreduce --create --alive --name "Spot Cluster" \
--instance-group master --instance-type m1.large --instance-count 1 --bid-price 0.25 \
--instance-group core --instance-type m1.large --instance-count 4 --bid-price 0.03 \
--instance-group task --instance-type c1.medium --instance-count 2 --bid-price 0.10
```
• Windows users:

```
ruby elastic-mapreduce --create --alive --name "Spot Cluster" --instance-group master
 --instance-type m1.large --instance-count 1 --bid-price 0.25 --instance-group core
 --instance-type m1.large --instance-count 4 --bid-price 0.03 --instance-group task
 --instance-type m1.medium --instance-count 2 --bid-price 0.10
```

To launch a task instance group on Spot Instances

You can launch a task instance group on Spot Instances using the `--bid-price` parameter, but multiple task groups are not supported. The following example shows how to create a cluster where only the task instance group uses Spot Instances. The command launches a cluster even if the request for Spot Instances cannot be fulfilled. In that case, Amazon EMR adds task nodes to the cluster if it is still running when the Spot Price falls below the bid price.

• In the directory where you installed the Amazon EMR CLI, type the following command.

• Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --alive --name "Spot Task Group" \
 --instance-group master --instance-type m1.large \
 --instance-count 1 \
 --instance-group core --instance-type m1.large \
 --instance-count 2 \
 --instance-group task --instance-type m1.large \
 --instance-count 4 --bid-price 0.03
```

• Windows users:

```
ruby elastic-mapreduce --create --alive --name "Spot Task Group" --instance-group master
 --instance-type m1.large --instance-count 1 --instance-group core
 --instance-type m1.large --instance-count 2 --instance-group task --instance-type m1.small
 --instance-count 4 --bid-price 0.03
```

To add a task instance group with Spot Instances to a cluster

Using the Amazon EMR CLI, you can add a task instance group with Spot Instances, but you cannot add multiple task groups.

• In the directory where you installed the Amazon EMR CLI, type the following command.

• Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --jobflow JobFlowId \
 --add-instance-group task --instance-type m1.small \
 --instance-count 5 --bid-price 0.05
```

• Windows users:

```
ruby elastic-mapreduce --jobflow JobFlowId --add-instance-group task --instance-type m1.small
 --instance-count 5 --bid-price 0.05
```

To change the number of Spot Instances in instance groups

You can change the number of requested Spot Instances in a cluster using the `--modify-instance-group` and `--instance-count` commands. Note that you can only increase the number of core
In the directory where you installed the Amazon EMR CLI, type the following command:

- **Linux, UNIX, and Mac OS X users:**

  ```bash
  ./elastic-mapreduce --jobflow JobFlowId --modify-instance-group task --instance-count 5
  ```

- **Windows users:**

  ```bash
  ruby elastic-mapreduce --jobflow JobFlowId --modify-instance-group task --instance-count 5
  ```

### Adding JAR Steps to Job Flows

--jar JAR_FILE_LOCATION

Specifies the location of a Java archive (JAR) file. Typically, the JAR file is stored in an Amazon S3 bucket.

--main-class

Specifies the JAR file's main class. This parameter is not needed if your JAR file has a manifest.

--args "arg1,arg2"

Specifies the arguments for the step.

### To create a cluster and submit a custom JAR step

- In the directory where you installed the custom JAR CLI, type the following command:

  - **Linux, UNIX, and Mac OS X users:**

    ```bash
    ./elastic-mapreduce --create --name "Test custom JAR" --jar s3://elasticmapreduce/samples/cloudburst/cloudburst.jar --arg s3://elasticmapreduce/samples/cloudburst/input/s_suis.br --arg s3://elasticmapreduce/samples/cloudburst/input/100k.br --arg s3://mybucket/cloudburst --arg 36 --arg 9 --arg 0 --arg 1 --arg 240 --arg 48 --arg 24 --arg 24 --arg 128 --arg 16
    ```

  - **Windows users:**

    ```bash
    ruby elastic-mapreduce --create --name "Test custom JAR" --jar s3://elasticmapreduce/samples/cloudburst/cloudburst.jar --arg s3://elasticmapreduce/samples/cloudburst/input/s_suis.br --arg s3://elasticmapreduce/samples/cloudburst/input/100k.br --arg s3://mybucket/cloudburst/output --arg 36 --arg 9 --arg 0 --arg 1 --arg 240 --arg 48 --arg 24 --arg 24 --arg 128 --arg 16
    ```

**Note**

The individual --arg values above could also be represented as --args followed by a comma-separated list.
By default, this command launches a cluster to run on a single-node cluster using an Amazon EC2 m1.small instance. Later, when your steps are running correctly on a small set of sample data, you can launch clusters to run on multiple nodes. You can specify the number of nodes and the type of instance to run with the --num-instances and --instance-type parameters, respectively.

To create a cluster and submit a Cascading step

- In the directory where you installed the Amazon EMR CLI, type the following command.

  - Linux, UNIX, and Mac OS X users:

    ```
    ./elastic-mapreduce --create --name "Test Cascading" \
    --bootstrap-action s3://files.cascading.org/sdk/2.1/install-cascading-sdk.sh \
    --JAR elasticmapreduce/samples/cloudfront/logprocessor.jar \
    --args "-input,s3://elasticmapreduce/samples/cloudfront/input,-start,any,-end,2010-12-27-02 300,-output,s3://mybucket/cloudfront/output/2010-12-27-02,-overallVolumeReport,-objectPopularityReport,-clientIPReport,-edgeLocationReport"
    ```

  - Windows users:

    ```
    ```

  Note
  The bootstrap action pre-installs the Cascading Software Development Kit on Amazon EMR. The Cascading SDK includes Cascading and Cascading-based tools such as Multitool and Load. The bootstrap action extracts the SDK and adds the available tools to the default PATH. For more information, go to http://www.cascading.org/sdk/.

To create a cluster with the Cascading Multitool

- Create a cluster referencing the Cascading Multitool JAR file and supply the appropriate Multitool arguments as follows.

  In the directory where you installed the Amazon EMR CLI, type the following command.

  - Linux, UNIX, and Mac OS X users:

    ```
    ./elastic-mapreduce --create \
    --jar s3://elasticmapreduce/samples/multitool/multitool-aws-03-31-09.jar \
    --args [args]
    ```

  - Windows users:

    ```
    ruby elastic-mapreduce --create --jar s3://elasticmapreduce/samples/multitool/multitool-aws-03-31-09.jar --args [args]
    ```
Adding JSON Steps to Job Flows

--json JSON_FILE

Adds a sequence of steps stored in the specified JSON file to the cluster.

--param VARIABLE=VALUE ARGS

Substitutes the string VARIABLE with the string VALUE in the JSON file.

Adding Streaming Steps to Job Flows

--cache FILE_LOCATION#NAME_OF_FILE_IN_CACHE

Adds an individual file to the distributed cache.

--cache-archive LOCATION#NAME_OF_ARCHIVE

Adds an archive file to the distributed cache

--ec2-instance-ids-to-terminate INSTANCE_ID

Use with --terminate and --modify-instance-group to specify the instances in the core and task instance groups to terminate. This allows you to shrink the number of core instances by terminating specific instances of your choice rather than those chosen by Amazon EMR.

--input LOCATION_OF_INPUT_DATA

Specifies the input location for the cluster.

--instance-count INSTANCE_COUNT

Sets the count of nodes for an instance group.

--instance-type INSTANCE_TYPE

Sets the type of EC2 instance to create nodes for an instance group.

--jobconf KEY=VALUE

Specifies jobconf arguments to pass to a streaming cluster, for example mapred.task.timeout=800000.

--mapper LOCATION_OF_MAPPER_CODE

The name of a Hadoop built-in class or the location of a mapper script.

--output LOCATION_OF_JOB_FLOW_OUTPUT

Specifies the output location for the cluster.

--reducer REDUCER

The name of a Hadoop built-in class or the location of a reducer script.

--stream

Used with --create and --arg to launch a streaming cluster.

Note
The --arg option must immediately follow the --stream option.

To create a cluster and submit a streaming step

• In the directory where you installed the Amazon EMR CLI, type one of the following commands.
Note
The Hadoop streaming syntax is different between Hadoop 1.x and Hadoop 2.x when using the Amazon EMR CLI.

For Hadoop 2.x, type the following command:

• Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --stream --ami-version 3.3
--instance-type ml.large --arg "-files" --arg "s3://elasticmapreduce/samples/wordcount/wordSplitter.py"
--input s3://elasticmapreduce/samples/wordcount/input --mapper wordSplitter.py --reducer aggregate
--output s3://mybucket/output/2014-01-16
```

• Windows users:

```
ruby elastic-mapreduce --create --stream --ami-version 3.3 --instance-type ml.large
--arg "-files" --arg "s3://elasticmapreduce/samples/wordcount/wordSplitter.py"
--input s3://elasticmapreduce/samples/wordcount/input --mapper wordSplitter.py --reducer aggregate --output s3://mybucket/output/2014-01-16
```

For Hadoop 1.x, type the following command:

• Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --stream
--input s3://elasticmapreduce/samples/wordcount/input
--mapper s3://elasticmapreduce/samples/wordcount/wordSplitter.py
--reducer aggregate
--output s3://mybucket/output/2014-01-16
```

• Windows users:

```
ruby elastic-mapreduce --create --stream --input s3://elasticmapreduce/samples/wordcount/input
--mapper s3://elasticmapreduce/samples/wordcount/wordSplitter.py
--reducer aggregate --output s3://mybucket/output/2014-01-16
```

By default, this command launches a cluster to run on a single-node cluster. Later, when your steps are running correctly on a small set of sample data, you can launch clusters to run on multiple nodes. You can specify the number of nodes and the type of instance to run with the `--num-instances` and `--instance-type` parameters, respectively.

To specify Distributed Cache files
Specify the options `--cache` or `--cache-archive` at the command line.

• Create a cluster and add the following parameters. The size of the file (or total size of the files in an archive file) must be less than the allocated cache size.

<table>
<thead>
<tr>
<th>Action</th>
<th>Parameter to add</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add an individual file to the</td>
<td><code>--cache</code> followed by the name and location of the file, the pound (#{}) sign, and then the name you want to give the file when it's placed in the local cache</td>
</tr>
<tr>
<td>Distributed Cache</td>
<td></td>
</tr>
</tbody>
</table>
### Adding Streaming Steps to Job Flows

<table>
<thead>
<tr>
<th>Action</th>
<th>Parameter to add</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add an archive file to the Distributed Cache</td>
<td>--cache-archive 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</td>
</tr>
</tbody>
</table>

Your cluster copies the files to the cache location before processing any job flow steps.

#### Example

The following command shows the creation of a streaming cluster and uses --cache to add one file, `sample_dataset_cached.dat`, to the cache. The Hadoop streaming syntax is different between Hadoop 1.x and Hadoop 2.x.

For Hadoop 2.x, use the following command:

- **Linux, UNIX, and Mac OS X users:**

  ```bash
  ./elastic-mapreduce --create --stream --arg "--files" --arg "s3://my_bucket/my_mapper.py,s3://my_bucket/my_reducer.py" 
  --input s3://my_bucket/my_input 
  --output s3://my_bucket/my_output 
  --mapper my_mapper.py 
  --reducer my_reducer.py 
  --cache s3://my_bucket/sample_dataset.dat#sample_dataset_cached.dat
  ```

- **Windows users:**

  ```bash
  ruby elastic-mapreduce --create --stream --arg "-files" --arg "s3://my_bucket/my_mapper.py,s3://my_bucket/my_reducer.py" --input s3://my_bucket/my_input --output s3://my_bucket/my_output --mapper my_mapper.py --reducer my_reducer.py --cache s3://my_bucket/sample_dataset.dat#sample_dataset_cached.dat
  ```

For Hadoop 1.x, use the following command:

- **Linux, UNIX, and Mac OS X users:**

  ```bash
  ./elastic-mapreduce --create --stream 
  --input s3://my_bucket/my_input 
  --output s3://my_bucket/my_output 
  --mapper s3://my_bucket/my_mapper.py 
  --reducer s3://my_bucket/my_reducer.py 
  --cache s3://my_bucket/sample_dataset.dat#sample_dataset_cached.dat
  ```

- **Windows users:**

  ```bash
  ruby elastic-mapreduce --create --stream --input s3://my_bucket/my_input --output s3://my_bucket/my_output --mapper s3://my_bucket/my_mapper.py --reducer s3://my_bucket/my_reducer.py --cache s3://my_bucket/sample_dataset.dat#sample_dataset Cached.dat
  ```
Assigning an Elastic IP Address to the Master Node

--eip ELASTIC_IP

Associates an Elastic IP address to the master node. If no Elastic IP address is specified, allocate a new Elastic IP address and associate it to the master node.

You can allocate an Elastic IP address and assign it to either a new or running cluster. After you assign an Elastic IP address to a cluster, it may take one or two minutes before the instance is available from the assigned address.

To assign an Elastic IP address to a new cluster

- Create a cluster and add the --eip parameter. The CLI allocates an Elastic IP address and waits until the Elastic IP address is successfully assigned to the cluster. This assignment can take up to two minutes to complete.

  **Note**
  If you want to use a previously allocated Elastic IP address, use the --eip parameter followed by your allocated Elastic IP address. If the allocated Elastic IP address is in use by another cluster, the other cluster loses the Elastic IP address and is assigned a new dynamic IP address.

To assign an Elastic IP address to a running cluster

1. If you do not currently have a running cluster, create a cluster.
2. Identify your cluster:

   Your cluster must have a public DNS name before you can assign an Elastic IP address. Typically, a cluster is assigned a public DNS name one or two minutes after launching the cluster.

   In the directory where you installed the Amazon EMR CLI, type the following command.

   - Linux, UNIX, and Mac OS X users:

     ```bash
     ./elastic-mapreduce --list
     ```

   - Windows users:

     ```bash
     ruby elastic-mapreduce --list
     ```

   The output looks similar to the following.

   ```
   j-SLRI9SCLK7UC    STARTING    ec2-75-101-168-82.compute-1.amazonaws.com
   New Job Flow    PENDING     Streaming Job
   ```

   The response includes the cluster ID and the public DNS name. You need the cluster ID to perform the next step.

3. Allocate and assign an Elastic IP address to the cluster:

   In the directory where you installed the Amazon EMR CLI, type the following command. If you assign an Elastic IP address that is currently associated with another cluster, the other cluster is assigned a new dynamic IP address.
• Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce JobFlowId --eip
```

• Windows users:

```
ruby elastic-mapreduce JobFlowId --eip
```

This allocates an Elastic IP address and associates it with the named cluster.

**Note**
If you want to use a previously allocated Elastic IP address, include your Elastic IP address, `Elastic_IP`, as follows.

• Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce JobFlowId --eip Elastic_IP
```

• Windows users:

```
ruby elastic-mapreduce JobFlowId --eip Elastic_IP
```

## Connecting to the Master Node

```
--get SOURCE
```
Copies the specified file from the master node using SCP.

```
--logs
```
Displays the step logs for the step most recently executed.

```
--put SOURCE
```
Copies a file to the master node using SCP.

```
--scp FILE_TO_COPY
```
Copies a file from your local directory to the master node of the cluster.

```
--socks
```
Uses SSH to create a tunnel to the master node of the specified cluster. You can then use this as a SOCKS proxy to view web interfaces hosted on the master node.

```
--ssh COMMAND
```
Uses SSH to connect to the master node of the specified cluster and, optionally, run a command. This option requires that you have an SSH client, such as OpenSSH, installed on your desktop.

```
--to DESTINATION
```
Specifies the destination location when copying files to and from the master node using SCP.

### To connect to the master node

To connect to the master node, you must: configure your credentials.json file so the keypair value is set to the name of the keypair you used to launch the cluster, set the key-pair-file value to the full path to your private key file, set appropriate permissions on the .pem file, and install an SSH client.
on your machine (such as OpenSSH). You can open an SSH connection to the master node by issuing the following command. This is a handy shortcut for frequent CLI users. Replace \textit{j-3L7WXXXXXHO4H} with your cluster identifier.

- Linux, UNIX, and Mac OS X users:

\begin{verbatim}
./elastic-mapreduce -j j-3L7WXXXXXHO4H --ssh
\end{verbatim}

- Windows users:

\begin{verbatim}
ruby elastic-mapreduce -j j-3L7WXXXXXHO4H --ssh
\end{verbatim}

\textbf{To create an SSH tunnel to the master node}

- In the directory where you installed the Amazon EMR CLI, type the following command.

  - Linux, UNIX, and Mac OS X users:

\begin{verbatim}
./elastic-mapreduce -j j-3L7WXXXXXHO4H --socks
\end{verbatim}

  - Windows users:

\begin{verbatim}
ruby elastic-mapreduce -j j-3L7WXXXXXHO4H --socks
\end{verbatim}

\textbf{Note}
The --socks feature is available only on the CLI version 2012-06-12 and later. To find out what version of the CLI you have, run \texttt{elastic-mapreduce --version} at the command line. You can download the latest version of the CLI from \url{http://aws.amazon.com/code/Elastic-MapReduce/2264}.

\section*{Creating Job Flows}

\textbf{--alive}

Used with \texttt{--create} to launch a cluster that continues running even after completing all its steps. Interactive clusters require this option.

\textbf{--ami-version AMI_VERSION}

Used with \texttt{--create} to specify the version of the AMI to use when launching the cluster. This setting also determines the version of Hadoop to install, because the \texttt{--hadoop-version} parameter is no longer supported.

In the Amazon EMR CLI, if you use the keyword \texttt{latest} instead of a version number for the AMI (for example \texttt{--ami-version latest}), the cluster is launched with the AMI listed as the "latest" AMI version—currently AMI version 2.4.2. This configuration is suitable for prototyping and testing, and is not recommended for production environments. This option is not supported by the AWS CLI, SDK, or API.

For Amazon EMR CLI version 2012-07-30 and later, the latest AMI is 2.4.2 with Hadoop 1.0.3. For Amazon EMR CLI versions 2011-12-08 to 2012-07-09, the latest AMI is 2.1.3 with Hadoop 0.20.205. For Amazon EMR CLI version 2011-12-11 and earlier, the latest AMI is 1.0.1 with Hadoop 0.18.

The default AMI is unavailable in the Asia Pacific (Sydney) region. Instead, use \texttt{--ami-version latest} (in the Amazon EMR CLI), fully specify the AMI, or use the major-minor version.
--availability-zone AVAILABILITY_ZONE

    The Availability Zone to launch the cluster in. For more information about Availability Zones
supported by Amazon EMR, see Regions and Endpoints in the Amazon Web Services General
Reference.

--bid-price BID_PRICE

    The bid price, in U.S. dollars, for a group of Spot Instances.

--create

    Launches a new cluster.

--hadoop-version VERSION

    Specify the version of Hadoop to install.

--info INFO

    Specifies additional information during cluster creation.

--instance-group INSTANCE_GROUP_TYPE

    Sets the instance group type. A type is MASTER, CORE, or TASK.

--jobflow-role IAM_ROLE_NAME

    Launches the EC2 instances of a cluster with the specified IAM role.

--service-role IAM_ROLE_NAME

    Launches the Amazon EMR service with the specified IAM role.

--key-pair KEY_PAIR_PEM_FILE

    The name of the Amazon EC2 key pair to set as the connection credential when you launch the
cluster.

--master-instance-type INSTANCE_TYPE

    The type of EC2 instances to launch as the master nodes in the cluster.

--num-instances NUMBER_OF_INSTANCES

    Used with --create and --modify-instance-group to specify the number of EC2 instances in
the cluster.

    You can increase or decrease the number of task instances in a running cluster, and you can add a
single task instance group to a running cluster. You can also increase but not decrease the number of
core instances.

--plain-output

    Returns the cluster identifier from the create step as simple text.

--region REGION

    Specifies the region in which to launch the cluster.

--slave-instance-type

    The type of EC2 instances to launch as the slave nodes in the cluster.

--subnet EC2-SUBNET_ID

    Launches a cluster in an Amazon VPC subnet.
--visible-to-all-users BOOLEAN

   Makes the instances in an existing cluster visible to all IAM users of the AWS account that launched the cluster.

--with-supported-products PRODUCT

   Installs third-party software on an Amazon EMR cluster; for example, installing a third-party distribution of Hadoop. It accepts optional arguments for the third-party software to read and act on. It is used with --create to launch the cluster that can use the specified third-party applications. The 2013-03-19 and newer versions of the Amazon EMR CLI accepts optional arguments using the --args parameter.

--with-termination-protection

   Used with --create to launch the cluster with termination protection enabled.

To launch a cluster into a VPC

After your VPC is configured, you can launch Amazon EMR clusters in it by using the --subnet argument with the subnet address.

- In the directory where you installed the Amazon EMR CLI, type the following command.
  - Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --alive --subnet subnet-77XXXX03
```
  
  - Windows users:

```
ruby elastic-mapreduce --create --alive --subnet subnet-77XXXX03
```

To create a long-running cluster

- In the directory where you installed the Amazon EMR CLI, type the following command.
  - Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --alive --name "Interactive Cluster" --num-instances=1 --master-instance-type=m1.large --hive-interactive
```
  
  - Windows users:

```
ruby elastic-mapreduce --create --alive --name "Interactive Cluster" --num-instances=1 --master-instance-type=m1.large --hive-interactive
```

To specify the AMI version when creating a cluster

When creating a cluster using the CLI, add the --ami-version parameter. If you do not specify this parameter, or if you specify --ami-version latest, the most recent version of AMI will be used.

- In the directory where you installed the Amazon EMR CLI, type the following command.
  - Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --alive --name "Static AMI Version" --ami-version 2.4.8
```

--num-instances 5 --instance-type m1.large

- Windows users:

ruby elastic-mapreduce --create --alive --name "Static AMI Version" --ami-version 2.4.8 --num-instances 5 --instance-type m1.large

The following example specifies the AMI using just the major and minor version. It will launch the cluster on the AMI that matches those specifications and has the latest patches. For example, if the most recent AMI version is 2.4.8, specifying --ami-version 2.4 would launch a cluster using AMI 2.4.8.

- Linux, UNIX, and Mac OS X users:

./elastic-mapreduce --create --alive --name "Major-Minor AMI Version" \
--ami-version 2.4 \
--num-instances 5 --instance-type m1.large

- Windows users:

ruby elastic-mapreduce --create --alive --name "Major-Minor AMI Version" --ami-version 2.4 --num-instances 5 --instance-type m1.large

The following example specifies that the cluster should be launched with the latest AMI.

- Linux, UNIX, and Mac OS X users:

./elastic-mapreduce --create --alive --name "Latest AMI Version" \
--ami-version latest \
--num-instances 5 --instance-type m1.large

- Windows users:

ruby elastic-mapreduce --create --alive --name "Latest AMI Version" --ami-version latest --num-instances 5 --instance-type m1.large

**To view the current AMI version of a cluster**

Use the --describe parameter to retrieve the AMI version on a cluster. The AMI version will be returned along with other information about the cluster.

- In the directory where you installed the Amazon EMR CLI, type the following command:

- Linux, UNIX, and Mac OS X users:

./elastic-mapreduce --describe --jobflow JobFlowID

- Windows users:

ruby elastic-mapreduce --describe --jobflow JobFlowID
To configure cluster visibility

By default, clusters created using the Amazon EMR CLI are not visible to all users. If you are adding IAM user visibility to a new cluster using the Amazon EMR CLI, add the --visible-to-all-users flag to the cluster call as shown in the following example.

- In the directory where you installed the Amazon EMR CLI, type the following command.
  - Linux, UNIX, and Mac OS X users:
    ```
    ./elastic-mapreduce --create --alive / 
    --instance-type m1.xlarge --num-instances 2 / 
    --visible-to-all-users
    ```
  - Windows users:
    ```
    ruby elastic-mapreduce --create --alive --instance-type m1.xlarge --num-instances 2 
    --visible-to-all-users
    ```

If you are adding IAM user visibility to an existing cluster, you can use the --set-visible-to-all-users option, and specify the identifier of the cluster to modify. The visibility of a running cluster can be changed only by the IAM user that created the cluster or the AWS account that owns the cluster.

In the directory where you installed the Amazon EMR CLI, type the following command.

- Linux, UNIX, and Mac OS X users:
  ```
  ./elastic-mapreduce --set-visible-to-all-users true --jobflow JobFlowId
  ```
- Windows users:
  ```
  ruby elastic-mapreduce --set-visible-to-all-users true --jobflow JobFlowId
  ```

To create and use IAM roles

If the default roles already exist, no output is returned. We recommend that you begin by creating the default roles, then modify those roles as needed. For more information about default roles, see Use Default IAM Roles and Managed Policies (p. 235).

1. In the directory where you installed the Amazon EMR CLI, type the following command:

   - Linux, UNIX, and Mac OS X users:
     ```
     ./elastic-mapreduce --create-default-roles
     ```
   - Windows users:
     ```
     ruby elastic-mapreduce --create-default-roles
     ```

2. To specify the default roles, type the following command. This command can also be used to specify custom roles.

   - Linux, UNIX, and Mac OS X users:
To launch a cluster with IAM roles

Add the `--service-role` and `--jobflow-role` parameters to the command that creates the cluster and specify the name of the IAM roles to apply to Amazon EMR and EC2 instances in the cluster.

- In the directory where you installed the Amazon EMR CLI, type the following command.
  - Linux, UNIX, and Mac OS X users:
    ```
    ./elastic-mapreduce --create --alive --name "Test cluster" --ami-version 2.4 --num-instances 5 --instance-type m1.large --service-role EMR_DefaultRole --jobflow-role EMR_EC2_DefaultRole
    ```
  - Windows users:
    ```
    ruby elastic-mapreduce --create --alive --name "Test cluster" --ami-version 2.4 --num-instances 5 --instance-type m1.large --service-role EMR_DefaultRole --jobflow-role EMR_EC2_DefaultRole
    ```

To set a default IAM role

If you launch most or all of your clusters with a specific IAM role, you can set that IAM role as the default for the Amazon EMR CLI, so you don’t need to specify it at the command line. You can override the IAM role specified in `credentials.json` at any time by specifying a different IAM role at the command line as shown in the preceding procedure.

- Add a `jobflow-role` field in the `credentials.json` file that you created when you installed the CLI. For more information about `credentials.json`, see Configuring Credentials (p. 590).

The following example shows the contents of a `credentials.json` file that causes the CLI to always launch clusters with the user-defined IAM roles, `MyCustomEC2Role` and `MyCustomEMRRole`.

```json
{
    "access-id": "AccessKeyID",
    "private-key": "PrivateKey",
    "key-pair": "KeyName",
    "jobflow-role": "MyCustomEC2Role",
    "service-role": "MyCustomEMRRole",
    "key-pair-file": "location of key pair file",
    "region": "Region",
    "log-uri": "location of bucket on Amazon S3"
}
```
To specify a region

- In the directory where you installed the Amazon EMR CLI, type the following command.
  - Linux, UNIX, and Mac OS X users:

    ```
    ./elastic-mapreduce --create --region eu-west-1
    ```
  - Windows users:

    ```
    ruby elastic-mapreduce --create --region eu-west-1
    ```

Tip
To reduce the number of parameters required each time you issue a command from the CLI, you can store information such as region in your credentials.json file. For more information about creating a credentials.json file, go to the Configuring Credentials (p. 590).

To launch a cluster with MapR

- In the directory where you installed the Amazon EMR CLI, specify the MapR edition and version by passing arguments with the --args option.
  - Linux, UNIX, and Mac OS X users:

    ```
    ./elastic-mapreduce --create --alive --instance-type m1.large --num-instances 3 --supported-product mapr --name m5 --args "--edition,m5,--version,3.1.1"
    ```
  - Windows users:

    ```
    ruby elastic-mapreduce --create --alive --instance-type m1.large --num-instances 3 --supported-product mapr --name m5 --args "--edition,m5,--version,3.1.1"
    ```

To reset a cluster in an ARRESTED state

Use the --modify-instance-group command to reset a cluster in the ARRESTED state. Enter the --modify-instance-group command as follows:

- In the directory where you installed the Amazon EMR CLI, type the following command.
  - Linux, UNIX, and Mac OS X users:

    ```
    ./elastic-mapreduce --modify-instance-group InstanceGroupID --instance-count COUNT
    ```
  - Windows users:

    ```
    ruby elastic-mapreduce --modify-instance-group InstanceGroupID --instance-count COUNT
    ```
The `<InstanceGroupId>` is the ID of the arrested instance group and `<COUNT>` is the number of nodes you want in the instance group.

**Tip**
You do not need to change the number of nodes from the original configuration to free a running cluster. Set `--instance-count` to the same count as the original setting.

### Using HBase Options

`--backup-dir BACKUP_LOCATION`
The directory where an HBase backup exists or should be created.

`--backup-version VERSION_NUMBER`
Specifies the version number of an existing HBase backup to restore.

`--consistent`
Pauses all write operations to the HBase cluster during the backup process, to ensure a consistent backup.

`--full-backup-time-interval INTERVAL`
An integer that specifies the period of time units to elapse between automated full backups of the HBase cluster.

`--full-backup-time-unit TIME_UNIT`
The unit of time to use with `--full-backup-time-interval` to specify how often automatically scheduled HBase backups should run. This can take any one of the following values: minutes, hours, days.

`--hbase`
Used to launch an HBase cluster.

`--hbase-backup`
Creates a one-time backup of HBase data to the location specified by `--backup-dir`.

`--hbase-restore`
Restores a backup from the location specified by `--backup-dir` and (optionally) the version specified by `--backup-version`.

`--hbase-schedule-backup`
Schedules an automated backup of HBase data.

`--incremental-backup-time-interval TIME_INTERVAL`
An integer that specifies the period of time units to elapse between automated incremental backups of the HBase cluster. Used with `--hbase-schedule-backup` this parameter creates regularly scheduled incremental backups. If this period schedules a full backup at the same time as an incremental backup is scheduled, only the full backup is created. Used with `--incremental-backup-time-unit`.

`--incremental-backup-time-unit TIME_UNIT`
The unit of time to use with `--incremental-backup-time-interval` to specify how often automatically scheduled incremental HBase backups should run. This can take any one of the following values: minutes, hours, days.
--disable-full-backups

Turns off scheduled full HBase backups by passing this flag into a call with --hbase-schedule-backup.

--disable-incremental-backups

Turns off scheduled incremental HBase backups by passing this flag into a call with --hbase-schedule-backup.

--start-time START_TIME

Specifies the time that a HBase backup schedule should start. If this is not set, the first backup begins immediately. This should be in ISO date-time format. You can use this to ensure your first data load process has completed before performing the initial backup or to have the backup occur at a specific time each day.

To launch a cluster and install HBase

Specify the --hbase parameter when you launch a cluster using the CLI.

The following example shows how to launch a cluster running HBase from the CLI. We recommend that you run at least two instances in the HBase cluster.

The CLI implicitly launches the HBase cluster with keep alive and termination protection set.

- In the directory where you installed the Amazon EMR CLI, type the following command.
  - Linux, UNIX, and Mac OS X users:
    ```
    ./elastic-mapreduce --create --hbase --name "HBase Cluster" \
    --num-instances 3 \
    --instance-type c1.xlarge
    ```
  - Windows users:
    ```
    ruby elastic-mapreduce --create --hbase --name "HBase Cluster" --num-instances 3 --instance-type c1.xlarge
    ```

To configure HBase daemons

Add a bootstrap action, configure-hbase-daemons, when you launch the HBase cluster. You can use this bootstrap action to configure one or more daemons and set values for zookeeper-opts and hbase-master-opts which configure the options used by the zookeeper and master node components of the HBase cluster.

- In the directory where you installed the Amazon EMR CLI, type the following command.
  - Linux, UNIX, and Mac OS X users:
    ```
    ./elastic-mapreduce --create --hbase --name "My HBase Cluster" \
    --bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hbase-daemons \
    --args "--hbase-zookeeper-opts=-Xmx1024m -XX:GCTimeRatio=19,--hbase-master-opts=-Xmx2048m,--hbase-regionserver-opts=-Xmx4096m"
    ```
  - Windows users:
    ```
    ruby elastic-mapreduce --create --hbase --name "My HBase Cluster" --bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hbase-daemons --args "--
Note
When you specify the arguments for this bootstrap action, you must put quotes around the --args parameter value to keep the shell from breaking the arguments up. You must also include a space character between JVM arguments; in the example above, there is a space between -Xmx1000M and -XX:GCTimeRatio=19.

To specify individual HBase site settings
Set the configure-hbase bootstrap action when you launch the HBase cluster, and specify the values within hbase-site.xml to change. The following example illustrates how to change the hbase.hregion.max.filesize settings.

- In the directory where you installed the Amazon EMR CLI, type the following command.
  - Linux, UNIX, and Mac OS X users:
    ```bash
    ./elastic-mapreduce --create --hbase --name "My HBase Cluster" \
    --bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hbase \
    --args -s,hbase.hregion.max.filesize=52428800
    ```
  - Windows users:
    ```bash
    ruby elastic-mapreduce --create --hbase --name "My HBase Cluster" --bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hbase --args -s,hbase.hregion.max.filesize=52428800
    ```

To specify HBase site settings with an XML file
1. Create a custom version of hbase-site.xml. Your custom file must be valid XML. To reduce the chance of introducing errors, start with the default copy of hbase-site.xml, located on the Amazon EMR HBase master node at /home/hadoop/conf/hbase-site.xml, and edit a copy of that file instead of creating a file from scratch. You can give your new file a new name, or leave it as hbase-site.xml.
2. Upload your custom hbase-site.xml file to an Amazon S3 bucket. It should have permissions set so the AWS account that launches the cluster can access the file. If the AWS account launching the cluster also owns the Amazon S3 bucket, it will have access.
3. Set the configure-hbase bootstrap action when you launch the HBase cluster, and pass in the location of your custom hbase-site.xml file.

- In the directory where you installed the Amazon EMR CLI, type the following command.
  - Linux, UNIX, and Mac OS X users:
    ```bash
    ./elastic-mapreduce --create --hbase --name "My HBase Cluster" \
    --bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hbase \
    --args --site-config-file s3://bucket/config.xml
    ```
  - Windows users:
    ```bash
    ruby elastic-mapreduce --create --hbase --name "My HBase Cluster" --bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hbase --args --site-config-file s3://bucket/config.xml
    ```
To configure an HBase cluster for Ganglia

Launch the cluster and specify both the `install-ganglia` and `configure-hbase-for-ganglia` bootstrap actions.

**Note**
You can prefix the Amazon S3 bucket path with the region where your HBase cluster was launched, for example `s3://region.elasticmapreduce/bootstrap-actions/configure-hbase-for-ganglia`. For a list of regions supported by Amazon EMR see [Choose an AWS Region](p. 27).

- In the directory where you installed the Amazon EMR CLI, type the following command.
  - Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --hbase --name "My HBase Cluster" \
--bootstrap-action s3://elasticmapreduce/bootstrap-actions/install-ganglia \
--bootstrap-action s3://region.elasticmapreduce/bootstrap-actions/configure-hbase-for-ganglia
```
  - Windows users:

```
ruby elastic-mapreduce --create --hbase --name "My HBase Cluster" --bootstrap-action s3://elasticmapreduce/bootstrap-actions/install-ganglia --bootstrap-action s3://region.elasticmapreduce/bootstrap-actions/configure-hbase-for-ganglia
```

To manually back up HBase data

Run `--hbase-backup` in the CLI and specify the cluster and the backup location in Amazon S3. Amazon EMR tags the backup with a name derived from the time the backup was launched. This is in the format YYYYMMDDTHHMMSSZ, for example: 20120809T031314Z. If you want to label your backups with another name, you can create a location in Amazon S3 (such as backups in the example below) and use the location name as a way to tag the backup files.

- In the directory where you installed the Amazon EMR CLI, type the following command.
  - Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --jobflow j-ABABABABABA --hbase-backup \\n--backup-dir s3://myawsbucket/backups/j-ABABABABABA
```
  - Windows users:

```
ruby elastic-mapreduce --jobflow j-ABABABABABA --hbase-backup --backup-dir s3://myawsbucket/backups/j-ABABABABABA
```

This example backs up data, and uses the `--consistent` flag to enforce backup consistency. This flag causes all writes to the HBase cluster to pause during the backup.

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --jobflow j-ABABABABABA --hbase-backup \\n--backup-dir s3://myawsbucket/backups/j-ABABABABABA \\n--consistent
```
  - Windows users:
To schedule automated backups of HBase data

Call `--hbase-schedule-backup` on the HBase cluster and specify the backup time interval and units. If you do not specify a start time, the first backup starts immediately. The following example creates a weekly full backup, with the first backup starting immediately.

- In the directory where you installed the Amazon EMR CLI, type the following command.
  - Linux, UNIX, and Mac OS X users:
    ```
    ./elastic-mapreduce --jobflow j-ABABABABABA --hbase-schedule-backup \n    --full-backup-time-interval 7 --full-backup-time-unit days \n    --backup-dir s3://mybucket/backups/j-ABABABABABA
    ```
  - Windows users:
    ```
    ruby elastic-mapreduce --jobflow j-ABABABABABA --hbase-schedule-backup --full-backup-time-interval 7 --full-backup-time-unit days --backup-dir s3://mybucket/backups/j-ABABABABABA
    ```

The following example creates a weekly full backup, with the first backup starting on 15 June 2012, 8 p.m. UTC time.

- Linux, UNIX, and Mac OS X users:
  ```
  ./elastic-mapreduce --jobflow j-ABABABABABA \n  --hbase-schedule-backup \n  --full-backup-time-interval 7 --full-backup-time-unit days \n  --backup-dir s3://mybucket/backups/j-ABABABABABA \n  --start-time 2012-06-15T20:00Z
  ```
- Windows users:
  ```
  ruby elastic-mapreduce --jobflow j-ABABABABABA --hbase-schedule-backup --full-backup-time-interval 7 --full-backup-time-unit days --backup-dir s3://mybucket/backups/j-ABABABABABA --start-time 2012-06-15T20:00Z
  ```

The following example creates a daily incremental backup. The first incremental backup will begin immediately.

- Linux, UNIX, and Mac OS X users:
  ```
  ./elastic-mapreduce --jobflow j-ABABABABABA \n  --hbase-schedule-backup \n  --incremental-backup-time-interval 24 \n  --incremental-backup-time-unit hours \n  --backup-dir s3://mybucket/backups/j-ABABABABABA
  ```
- Windows users:
The following example creates a daily incremental backup, with the first backup starting on 15 June 2012, 8 p.m. UTC time.

- **Linux, UNIX, and Mac OS X users:**

  ```bash
  ./elastic-mapreduce --jobflow j-ABABABABABA 
  --hbase-schedule-backup 
  --incremental-backup-time-interval 24 
  --incremental-backup-time-unit hours 
  --backup-dir s3://mybucket/backups/j-ABABABABABA 
  --start-time 2012-06-15T20:00Z
  ```

- **Windows users:**

  ```bash
  ```

The following example creates both a weekly full backup and a daily incremental backup, with the first full backup starting immediately. Each time the schedule has the full backup and the incremental backup scheduled for the same time, only the full backup will run.

- **Linux, UNIX, and Mac OS X users:**

  ```bash
  ./elastic-mapreduce --jobflow j-ABABABABABA 
  --hbase-schedule-backup 
  --full-backup-time-interval 7 
  --full-backup-time-unit days 
  --incremental-backup-time-interval 24 
  --incremental-backup-time-unit hours 
  --backup-dir s3://mybucket/backups/j-ABABABABABA
  ```

- **Windows users:**

  ```bash
  ```

The following example creates both a weekly full backup and a daily incremental backup, with the first full backup starting on June 15, 2012. Each time the schedule has the full backup and the incremental backup scheduled for the same time, only the full backup will run.

- **Linux, UNIX, and Mac OS X users:**

  ```bash
  ./elastic-mapreduce --jobflow j-ABABABABABA 
  --hbase-schedule-backup 
  --full-backup-time-interval 7 
  --full-backup-time-unit days 
  --incremental-backup-time-interval 24 
  --incremental-backup-time-unit hours 
  --backup-dir s3://mybucket/backups/j-ABABABABABA
  ```
Use the following command to create both a weekly full backup and a daily incremental backup, with the first full backup starting on June 15, 2012. Each time the schedule has the full backup and the incremental backup scheduled for the same time, only the full backup will run. The --consistent flag is set, so both the incremental and full backups will pause write operations during the initial portion of the backup process to ensure data consistency.

- Windows users:

```
```

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --jobflow j-ABABABABABA \  
--hbase-schedule-backup \  
--full-backup-time-interval 7 \  
--full-backup-time-unit days \  
--incremental-backup-time-interval 24 \  
--incremental-backup-time-unit hours \  
--backup-dir s3://mybucket/backups/j-ABABABABABA \  
--start-time 2012-06-15T20:00Z \  
--consistent
```

- Windows users:

```
```

### To turn off automated HBase backups

Call the cluster with the --hbase-schedule-backup parameter and set the --disable-full-backups or --disable-incremental-backups flag, or both flags.

1. In the directory where you installed the Amazon EMR CLI, type the following command.

   - Linux, UNIX, and Mac OS X users:

   ```
   ./elastic-mapreduce --jobflow j-ABABABABABA \  
   --hbase-schedule-backup --disable-full-backups
   ```

   - Windows users:

   ```
ruby elastic-mapreduce --jobflow j-ABABABABABA --hbase-schedule-backup --disable-full-backups
```

2. Use the following command to turn off incremental backups.

   - Linux, UNIX, and Mac OS X users:

   ```
   ./elastic-mapreduce --jobflow j-ABABABABABA \  
   --hbase-schedule-backup --disable-incremental-backups
   ```

   - Windows users:

   ```
ruby elastic-mapreduce --jobflow j-ABABABABABA --hbase-schedule-backup --disable-incremental-backups
```
3. Use the following command to turn off both full and incremental backups.

- Windows users:
  ```bash
  ruby elastic-mapreduce --jobflow j-ABABABABABA --hbase-schedule-backup --disable-full-backups --disable-incremental-backups
  ```

- Linux, UNIX, and Mac OS X users:
  ```bash
  ./elastic-mapreduce --jobflow j-ABABABABABA --hbase-schedule-backup --disable-full-backups --disable-incremental-backups
  ```

- Windows users:
  ```bash
  ruby elastic-mapreduce --jobflow j-ABABABABABA --hbase-schedule-backup --disable-full-backups --disable-incremental-backups
  ```

To restore HBase backup data to a running cluster

Run an `--hbase-restore` step and specify the jobflow, the backup location in Amazon S3, and (optionally) the name of the backup version. If you do not specify a value for `--backup-version`, Amazon EMR loads the last version in the backup directory. This is the version with the name that is lexicographically greatest.

- In the directory where you installed the Amazon EMR CLI, type the following command.

- Linux, UNIX, and Mac OS X users:
  ```bash
  ./elastic-mapreduce --jobflow j-ABABABABABA --hbase-restore --backup-dir s3://myawsbucket/backups/j-ABABABABABA
  ```

- Windows users:
  ```bash
  ruby elastic-mapreduce --jobflow j-ABABABABABA --hbase-restore --backup-dir s3://myawsbucket/backups/j-ABABABABABA
  ```

This example restored the HBase cluster to the specified version of backup data stored in `s3://myawsbucket/backups`, overwriting any data stored in the HBase cluster.

- Linux, UNIX, and Mac OS X users:
  ```bash
  ./elastic-mapreduce --jobflow j-ABABABABABA --hbase-restore --backup-dir s3://myawsbucket/backups/j-ABABABABABA --backup-version 20120809T031314Z
  ```

- Windows users:
  ```bash
  ruby elastic-mapreduce --jobflow j-ABABABABABA --hbase-restore --backup-dir s3://myawsbucket/backups/j-ABABABABABA --backup-version 20120809T031314Z
  ```

To populate a new cluster with HBase backup data

When you add `--hbase-restore` and `--backup-directory` to the `--create` step in the CLI, you can optionally specify `--backup-version` to indicate which version in the backup directory to load. If
you do not specify a value for --backup-version, Amazon EMR loads the last version in the backup directory. This will either be the version with the name that is lexicographically last or, if the version names are based on timestamps, the latest version.

- In the directory where you installed the Amazon EMR CLI, type the following command line.

- Linux, UNIX, and Mac OS X users:

```bash
./elastic-mapreduce --create --name "My HBase Restored" \  --hbase --hbase-restore \  --backup-dir s3://myawsbucket/backups/j-ABABABABABA
```

- Windows users:

```bash
ruby elastic-mapreduce --create --name "My HBase Restored" --hbase --hbase-restore --backup-dir s3://myawsbucket/backups/j-ABABABABABA
```

This example creates a new HBase cluster and loads it with the specified version of data in s3://myawsbucket/backups/j-ABABABABABA.

- Linux, UNIX, and Mac OS X users:

```bash
./elastic-mapreduce --create --name "My HBase Restored" \  --hbase --hbase-restore \  --backup-dir s3://myawsbucket/backups/j-ABABABABABA \  --backup-version 20120809T031314Z
```

- Windows users:

```bash
ruby elastic-mapreduce --create --name "My HBase Restored" --hbase --hbase-restore --backup-dir s3://myawsbucket/backups/j-ABABABABABA --backup-version 20120809T031314Z
```

### Using Hive Options

--hive-interactive

Used with --create to launch a cluster with Hive installed.

--hive-script HIVE_SCRIPT_LOCATION

The Hive script to run in the cluster.

--hive-site HIVE_SITE_LOCATION

Installs the configuration values in hive-site.xml in the specified location. The --hive-site parameter overrides only the values defined in hive-site.xml.

--hive-versions HIVE_VERSIONS

The Hive version or versions to load. This can be a Hive version number or "latest" to load the latest version. When you specify more than one Hive version, separate the versions with a comma.

### To pass variable values into Hive steps

To pass a Hive variable value into a step using the Amazon EMR CLI, type the --args parameter with the -d flag.

- In the directory where you installed the Amazon EMR CLI, type the following command.
• Linux, UNIX, and Mac OS X users:

```bash
./elastic-mapreduce --hive-script --arg s3://mybucket/script.q \ --args -d,LIB=s3://elasticmapreduce/samples/hive-ads/lib
```

• Windows users:

```bash
ruby elastic-mapreduce --hive-script --arg s3://mybucket/script.q --args -d,LIB=s3://elasticmapreduce/samples/hive-ads/lib
```

To specify the latest Hive version when creating a cluster

Use the `--hive-versions` option with the `latest` keyword.

• In the directory where you installed the Amazon EMR CLI, type the following command line.

• Linux, UNIX, and Mac OS X users:

```bash
./elastic-mapreduce --create --alive --name "Test Hive" \ --num-instances 5 --instance-type m1.large \ --hive-interactive \ --hive-versions latest
```

• Windows users:

```bash
ruby elastic-mapreduce --create --alive --name "Test Hive" --num-instances 5 --instance-type m1.large --hive-interactive --hive-versions latest
```

To specify the Hive version for a cluster that is interactive and uses a Hive script

If you have a cluster that uses Hive both interactively and from a script, you must set the Hive version for each type of use. The following example illustrates setting both the interactive and the script version of Hive to use 0.7.1.

• In the directory where you installed the Amazon EMR CLI, type the following command line.

• Linux, UNIX, and Mac OS X users:

```bash
./elastic-mapreduce --create --debug --log-uri s3://mybucket/logs/ \ --name "Testing m1.large AMI 1" \ --ami-version latest \ --instance-type m1.large --num-instances 5 \ --hive-interactive --hive-versions 0.7.1.2 \ --hive-script s3://mybucket/hive-script.hql --hive-versions 0.7.1.2
```

• Windows users:

```bash
ruby elastic-mapreduce --create --debug --log-uri s3://mybucket/logs/ --name "Testing m1.large AMI" --ami-version latest --instance-type m1.large --num-instances 5 --hive-interactive --hive-versions 0.7.1.2 --hive-script s3://mybucket/hive-script.hql --hive-versions 0.7.1.2
```

To load multiple versions of Hive for a cluster

With this configuration, you can use any of the installed versions of Hive on the cluster.
• In the directory where you installed the Amazon EMR CLI, type the following command line.
  
  • Linux, UNIX, and Mac OS X users:
    
    ```bash
    ./elastic-mapreduce --create --alive --name "Test Hive" \ 
    --num-instances 5 --instance-type m1.large \ 
    --hive-interactive \ 
    --hive-versions 0.5,0.7.1
    ```
  
  • Windows users:
    
    ```bash
    ruby elastic-mapreduce --create --alive --name "Test Hive" --num-instances 5 --instance-type m1.large --hive-interactive --hive-versions 0.5,0.7.1
    ```

To call a specific version of Hive

• Add the version number to the call. For example, hive-0.5 or hive-0.7.1.

  **Note**
  If you have multiple versions of Hive loaded on a cluster, calling `hive` accesses the default version of Hive or the version loaded last if there are multiple `--hive-versions` options specified in the cluster creation call. When the comma-separated syntax is used with `--hive-versions` to load multiple versions, `hive` accesses the default version of Hive.

  **Note**
  When running multiple versions of Hive concurrently, all versions of Hive can read the same data. They cannot, however, share metadata. Use an external metastore if you want multiple versions of Hive to read and write to the same location.

To display the Hive version

This is a useful command to call after you have upgraded to a new version of Hive to confirm that the upgrade succeeded, or when you are using multiple versions of Hive and need to confirm which version is currently running.

• In the directory where you installed the Amazon EMR CLI, type the following command.

  • Linux, UNIX, and Mac OS X users:
    
    ```bash
    ./elastic-mapreduce --jobflow JobFlowID --print-hive-version
    ```
  
  • Windows users:
    
    ```bash
    ruby elastic-mapreduce --jobflow JobFlowID --print-hive-version
    ```

To launch a Hive cluster in interactive mode

• In the directory where you installed the Amazon EMR CLI, type the following command line.

  • Linux, UNIX, and Mac OS X users:
    
    ```bash
    ./elastic-mapreduce --create --alive --name "Hive cluster" \ 
    --num-instances 5 --instance-type m1.large \ 
    --hive-interactive
    ```
  
  • Windows users:
To launch a cluster and submit a Hive step

- In the directory where you installed the Amazon EMR CLI, type the following command.
- Linux, UNIX, and Mac OS X users:

  ruby elastic-mapreduce --create --alive --name "Hive cluster" --num-instances 5 --instance-type m1.large --hive-interactive

- Windows users:

  ruby elastic-mapreduce --create --alive --name "Hive cluster" --num-instances 5 --instance-type m1.large --hive-interactive

  By default, this command launches a cluster to run on a two-node cluster. Later, when your steps are running correctly on a small set of sample data, you can launch clusters to run on multiple nodes. You can specify the number of nodes and the type of instance to run with the --num-instances and --instance-type parameters, respectively.

To create an external Hive metastore using the Amazon EMR CLI

- To specify the location of the configuration file using the Amazon EMR CLI, in the directory where you installed the Amazon EMR CLI, type the following command.
- Linux, UNIX, and Mac OS X users:

  ruby elastic-mapreduce --create --alive --name "Hive cluster" --num-instances 5 --instance-type m1.large --hive-interactive

  By default, this command launches a cluster to run on a two-node cluster. Later, when your steps are running correctly on a small set of sample data, you can launch clusters to run on multiple nodes. You can specify the number of nodes and the type of instance to run with the --num-instances and --instance-type parameters, respectively.

To interactively submit Hive jobs

In the directory where you installed the Amazon EMR CLI, type the following commands.

1. If Hive is not already installed, type the following command to install it.
Using Impala Options

--impala-conf OPTIONS

Use with the --create and --impala-interactive options to provide command-line parameters for Impala to parse.

The parameters are key/value pairs in the format "key1=value1,key2=value2,...". For example to set the Impala start-up options IMPALA_BACKEND_PORT and IMPALA_MEM_LIMIT, use the following command:

```
./elastic-mapreduce --create --alive --instance-type m1.large --
instance-count 3 --ami-version 3.0.2 --impala-interactive --impala-conf
"IMPALA_BACKEND_PORT=22001,IMPALA_MEM_LIMIT=70%"
```

--impala-interactive

Use with the --create option to launch an Amazon EMR cluster with Impala installed.

--impala-output PATH

Use with the --impala-script option to store Impala script output to an Amazon S3 bucket using the syntax --impala-output s3-path.

--impala-script [SCRIPT]

Use with the --create option to add a step to a cluster to run an Impala query file stored in Amazon S3 using the syntax --impala-scripts s3-path. For example:
When using `--impala-script` with `--create`, the `--impala-version` and `--impala-conf` options will also function. It is acceptable, but unnecessary, to use `--impala-interactive` and `--impala-script` in the same command when creating a cluster. The effect is equivalent to using `--impala-script` alone.

Alternatively, you can add a step to an existing cluster, but you must already have installed Impala on the cluster. For example:

```
./elastic-mapreduce -j cluster-id --impala-script s3://my-bucket/script-name.sql --impala-output s3://my-bucket/
```

If you try to use `--impala-script` to add a step to a cluster where Impala is not installed, you will get an error message similar to **Error: Impala is not installed**.

`--impala-version IMPALA_VERSION`

The version of Impala to be installed.

### To add Impala to a cluster

- In the directory where you installed the Amazon EMR CLI, type the following command.
- Linux, UNIX, and Mac OS X users:

  ```
  ./elastic-mapreduce --create --alive --instance-type m1.large --instance-count 3 --ami-version 3.3 --impala-interactive --key-pair keypair-name
  ```

- Windows users:

  ```
  ruby elastic-mapreduce --create --alive --instance-type m1.large --instance-count 3 --ami-version 3.3 --impala-interactive --key-pair keypair-name
  ```

### Listing and Describing Job Flows

- `--active`
  Modifies a command to apply only to clusters in the RUNNING, STARTING or WAITING states. Used with `--list`.

- `--all`
  Modifies a command to apply only to all clusters, regardless of status. Used with `--list`, it lists all the clusters created in the last two weeks.

- `--created-after=DATETIME`
  Lists all clusters created after the specified time and date in XML date-time format.

- `--created-before=DATETIME`
  Lists all clusters created before the specified time and date in XML date-time format.

- `--describe`
  Returns information about the specified cluster or clusters.
--list

Lists clusters created in the last two days.

--no-steps

Prevents the CLI from listing steps when listing clusters.

--print-hive-version

Prints the version of Hive that is currently active on the cluster.

--state JOB_FLOW_STATE

Specifies the state of the cluster. The cluster state will be one of the following values: STARTING, RUNNING, WAITING, TERMINATED.

To retrieve the public DNS name of the master node

You can retrieve the master public DNS using the Amazon EMR CLI.

- In the directory where you installed the Amazon EMR CLI, type the following command.
  - Linux, UNIX, and Mac OS X users:
    ```bash
    ./elastic-mapreduce --list
    ```
  - Windows users:
    ```bash
    ruby elastic-mapreduce --list
    ```

To list clusters created in the last two days

- Use the `--list` parameter with no additional arguments to display clusters created during the last two days as follows:

In the directory where you installed the Amazon EMR CLI, type the following command.

- Linux, UNIX, and Mac OS X users:
  ```bash
  ./elastic-mapreduce --list
  ```
- Windows users:
  ```bash
  ruby elastic-mapreduce --list
  ```

The response is similar to the following:

<table>
<thead>
<tr>
<th>Cluster ID</th>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>j-1YE2DN7RXJBWU</td>
<td>FAILED</td>
<td>Example Job Flow</td>
</tr>
<tr>
<td></td>
<td>CANCELLED</td>
<td>Custom Jar</td>
</tr>
<tr>
<td>j-3GJ4FRNKGY97</td>
<td>COMPLETED</td>
<td>ec2-67-202-3-73.compute-1.amazonaws.com</td>
</tr>
<tr>
<td>j-5XFIQ58FFNW</td>
<td>COMPLETED</td>
<td>ec2-67-202-51-30.compute-1.amazonaws.com</td>
</tr>
<tr>
<td></td>
<td>COMPLETED</td>
<td>Custom Jar</td>
</tr>
</tbody>
</table>

The example response shows that three clusters were created in the last two days. The indented lines are the steps of the cluster. The information for a cluster is in the following order: the cluster ID, the cluster state, and the master node's public DNS.
state, the DNS name of the master node, and the cluster name. The information for a cluster step is in the following order: step state, and step name.

If no clusters were created in the previous two days, this command produces no output.

To list active clusters

- Use the `--list` and `--active` parameters as follows:
  - Linux, UNIX, and Mac OS X users:
    
    ```bash
    ./elastic-mapreduce --list --active
    ```
  - Windows users:
    
    ```bash
    ruby elastic-mapreduce --list --active
    ```

  The response lists clusters that are in the state of STARTING, RUNNING, or SHUTTING_DOWN.

To list only running or terminated clusters

- Use the `--state` parameter as follows:
  - Linux, UNIX, and Mac OS X users:
    
    ```bash
    ./elastic-mapreduce --list --state RUNNING --state TERMINATED
    ```
  - Windows users:
    
    ```bash
    ruby elastic-mapreduce --list --state RUNNING --state TERMINATED
    ```

  The response lists clusters that are running or terminated.

To view information about a cluster

You can view information about a cluster using the `--describe` parameter with the cluster ID.

- Use the `--describe` parameter with a valid cluster ID.
  - Linux, UNIX, and Mac OS X users:
    
    ```bash
    ./elastic-mapreduce --describe --jobflow JobFlowID
    ```
  - Windows users:
    
    ```bash
    ruby elastic-mapreduce --describe --jobflow JobFlowID
    ```

To interactively submit Hadoop jobs

- To interactively submit Hadoop jobs using the Amazon EMR CLI, use the `--ssh` parameter to create an SSH connection to the master node and set the value to the command you want to run. In the directory where you installed the Amazon EMR CLI, type the following command. This command uses the `--scp` parameter to copy the JAR file `myjar.jar` from your local machine to the master node of cluster `JobFlowID` and runs the command using an SSH connection.
Passing Arguments to Steps

--arg ARG

Passes in a single argument value to a script or application running on the cluster.

Note
When used in a Hadoop streaming cluster, if you use the --arg options, they must immediately follow the --stream option.

--args ARG1,ARG2,ARG3,...

Passes in multiple arguments, separated by commas, to a script or application running on the cluster. This is a shorthand for specifying multiple --arg options. The --args option does not support escaping for the comma character (,). To pass arguments containing the comma character (,) use the --arg option which does not consider commas as a separator. The argument string may be surrounded with double-quotes. In addition, you can use double quotes when passing arguments containing whitespace characters.

Note
When used in a Hadoop streaming cluster, if you use the --args option, it must immediately follow the --stream option.

--step-action

Specifies the action the cluster should take when the step finishes. This can be one of CANCEL_AND_WAIT, TERMINATE_JOB_FLOW, or CONTINUE.

--step-name

Specifies a name for a cluster step.

This section describes the methods for adding steps to a cluster using the Amazon EMR CLI. You can add steps to a running cluster only if you use the --alive parameter to when you create the cluster. This parameter creates a long-running cluster by keeping the cluster active even after the completion of your steps.

To add a custom JAR step to a running cluster

In the directory where you installed the Amazon EMR CLI, type the following command.

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --jobflow JobFlowID --scp myjar.jar --ssh "hadoop jar myjar.jar"
```

- Windows users:

```
ruby elastic-mapreduce --jobflow JobFlowID --scp myjar.jar --ssh "hadoop jar myjar.jar"
```
Using Pig Options

--pig-interactive
Used with --create to launch a cluster with Pig installed.
--pig-script PIG_SCRIPT_LOCATION
The Pig script to run in the cluster.
--pig-versions VERSION
Specifies the version or versions of Pig to install on the cluster. If specifying more than one version of Pig, separate the versions with commas.

To add a specific Pig version to a cluster

• Use the --pig-versions parameter. The following command-line example creates an interactive Pig cluster running Hadoop 1.0.3 and Pig 0.11.1.

In the directory where you installed the Amazon EMR CLI, type the following command.

• Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --alive --name "Test Pig" \
--ami-version 2.3.6 \
```

This command adds a step that downloads and runs a JAR file. The arguments are passed to the main function in the JAR file. If your JAR file does not have a manifest, specify the JAR file's main class using the --main-class option.
Using Pig Options

--num-instances 5 --instance-type m1.large
--pig-interactive
--pig-versions 0.11.1

• Windows users:

  ruby elastic-mapreduce --create --alive --name "Test Pig" --ami-version 2.3.6 --num-instances 5 --instance-type m1.large --pig-interactive --pig-versions 0.11.1

To add the latest version of Pig to a cluster

• Use the --pig-versions parameter with the latest keyword. The following command-line example creates an interactive Pig cluster running the latest version of Pig.

  In the directory where you installed the Amazon EMR CLI, type the following command.

  • Linux, UNIX, and Mac OS X users:

    ./elastic-mapreduce --create --alive --name "Test Latest Pig" --ami-version 2.2 --num-instances 5 --instance-type m1.large --pig-interactive --pig-versions latest

  • Windows users:

    ruby elastic-mapreduce --create --alive --name "Test Latest Pig" --ami-version 2.2 --num-instances 5 --instance-type m1.large --pig-interactive --pig-versions latest

To add multiple versions of Pig to a cluster

• Use the --pig-versions parameter and separate the version numbers by commas. The following command-line example creates an interactive Pig job flow running Hadoop 0.20.205 and Pig 0.9.1 and Pig 0.9.2. With this configuration, you can use either version of Pig on the cluster.

  In the directory where you installed the Amazon EMR CLI, type the following command.

  • Linux, UNIX, and Mac OS X users:

    ./elastic-mapreduce --create --alive --name "Test Pig" --ami-version 2.0 --num-instances 5 --instance-type m1.large --pig-interactive --pig-versions 0.9.1,0.9.2

  • Windows users:

    ruby elastic-mapreduce --create --alive --name "Test Pig" --ami-version 2.0 --num-instances 5 --instance-type m1.large --pig-interactive --pig-versions 0.9.1,0.9.2

If you have multiple versions of Pig loaded on a cluster, calling Pig accesses the default version of Pig, or the version loaded last if there are multiple --pig-versions parameters specified in the cluster creation call. When the comma-separated syntax is used with --pig-versions to load multiple versions, Pig accesses the default version.
To run a specific version of Pig on a cluster

- Add the version number to the call. For example, pig-0.11.1 or pig-0.9.2. You would do this, for example, in an interactive Pig cluster by using SSH to connect to the master node and then running a command like the following from the terminal.

    pig-0.9.2

To run Pig in interactive mode

To run Pig in interactive mode use the --alive parameter to create a long-running cluster with the --pig-interactive parameter.

- In the directory where you installed the Amazon EMR CLI, type the following command.
  
  Linux, UNIX, and Mac OS X users:

  ```
  ./elastic-mapreduce --create --alive --name "Testing Pig" \ 
  --num-instances 5 --instance-type ml.large \ 
  --pig-interactive
  ```

  Windows users:

  ```
  ruby elastic-mapreduce --create --alive --name "Testing Pig" --num-instances 5 --instance-type ml.large --pig-interactive
  ```

To add Pig to a cluster and submit a Pig step

- In the directory where you installed the Amazon EMR CLI, type the following command.

  Linux, UNIX, and Mac OS X users:

  ```
  ./elastic-mapreduce --create --name "Test Pig" \ 
  --pig-script s3://elasticmapreduce/samples/pig-apache/do-reports2.pig \ 
  --ami-version 2.0 \ 
  --args "-p,INPUT=s3://elasticmapreduce/samples/pig-apache/input, \ 
  -p,OUTPUT=s3://mybucket/pig-apache/output"
  ```

  Windows users:

  ```
  ruby elastic-mapreduce --create --name "Test Pig" --pig-script s3://elasticmapreduce/ 
  samples/pig-apache/do-reports2.pig --ami-version 2.0 --args "-p,INPUT=s3:// 
  elasticmapreduce/samples/pig-apache/input, -p,OUTPUT=s3://mybucket/pig-apache/output"
  ```

By default, this command launches a single-node cluster. Later, when your steps are running correctly on a small set of sample data, you can launch clusters to run on multiple nodes. You can specify the number of nodes and the type of instance to run with the --num-instances and --instance-type parameters, respectively.
### Specifying Step Actions

**--enable-debugging**

Used with **--create** to launch a cluster with debugging enabled.

**--script SCRIPT_LOCATION**

Specifies the location of a script. Typically, the script is stored in an Amazon S3 bucket.

**--wait-for-steps**

Causes the cluster to wait until a step has completed.

When you submit steps to a cluster using the Amazon EMR CLI, you can specify that the CLI should wait until the cluster has completed all pending steps before accepting additional commands. This can be useful, for example, if you are using a step to copy data from Amazon S3 into HDFS and need to be sure that the copy operation is complete before you run the next step in the cluster. You do this by specifying the **--wait-for-steps** parameter after you submit the copy step.

**Note**

The AWS CLI does not have an option comparable to the **--wait-for-steps** parameter.

The **--wait-for-steps** parameter does not ensure that the step completes successfully, just that it has finished running. If, as in the earlier example, you need to ensure the step was successful before submitting the next step, check the cluster status. If the step failed, the cluster is in the FAILED status.

Although you can add the **--wait-for-steps** parameter in the same CLI command that adds a step to the cluster, it is best to add it in a separate CLI command. This ensures that the **--wait-for-steps** argument is parsed and applied after the step is created.

**To wait until a step completes**

- Add the **--wait-for-steps** parameter to the cluster. This is illustrated in the following example, where **JobFlowID** is the cluster identifier that Amazon EMR returned when you created the cluster. The JAR, main class, and arguments specified in the first CLI command are from the Word Count sample application; this command adds a step to the cluster. The second CLI command causes the cluster to wait until all of the currently pending steps have completed before accepting additional commands. In the directory where you installed the Amazon EMR CLI, type the following command.

\[
/elastic-mapreduce -j JobFlowID \n--jar s3n://elasticmapreduce/samples/cloudburst/cloudburst.jar \n--main-class org.myorg.WordCount \n--arg s3n://elasticmapreduce/samples/cloudburst/input/s_suis.br \n--arg s3n://elasticmapreduce/samples/cloudburst/input/100k.br \n--arg hdfs:///cloudburst/output/1 \n--arg 36 --arg 3 --arg 0 --arg 1 --arg 240 --arg 48 --arg 24 --arg 24 --arg 128 --arg 16
/elastic-mapreduce -j JobFlowID --wait-for-steps
\]

- Linux, UNIX, and Mac OS X users:

\[
ruby elastic-mapreduce -j JobFlowID --jar s3n://elasticmapreduce/samples/cloudburst/cloudburst.jar --main-class org.myorg.WordCount --arg s3n://elasticmapreduce/samples/cloudburst/input/s_suis.br --arg s3n://elasticmapreduce/samples/cloudburst/input/100k.br --arg hdfs:///cloudburst/output/1 --arg 36 --arg 3 --arg 0 --arg 1 --arg 240 --arg 48 --arg 24 --arg 24 --arg 128 --arg 16
\]

- Windows users:
To enable the debugging tool

- Use the `--enable-debugging` argument when you create the cluster. You must also set the `--log-uri` argument and specify a location in Amazon S3 because archiving the log files to Amazon S3 is a prerequisite of the debugging tool. Alternately, you can set the `--log-uri` value in the `credentials.json` file that you configured for the CLI. For more information about `credentials.json`, see "Configuring Credentials" in Install the Amazon EMR Command Line Interface (Deprecated) (p. 588). The following example illustrates creating a cluster that archives log files to Amazon S3. Replace `mybucket` with the name of your bucket.

- Linux, UNIX, and Mac OS X users:

  ```bash
  ./elastic-mapreduce --create --enable-debugging --log-uri s3://mybucket
  ```

- Windows users:

  ```bash
  ruby elastic-mapreduce --create --enable-debugging --log-uri s3://mybucket
  ```

Specifying Bootstrap Actions

`--bootstrap-action LOCATION_OF_bootstrap_ACTION_SCRIPT`

Used with `--create` to specify a bootstrap action to run when the cluster launches. The location of the bootstrap action script is typically a location in Amazon S3. You can add more than one bootstrap action to a cluster.

`--bootstrap-name bootstrap_NAME`

Sets the name of the bootstrap action.

`--args "arg1,arg2"`

Specifies arguments for the bootstrap action.

To add Ganglia to a cluster using a bootstrap action

- When you create a new cluster using the Amazon EMR CLI, specify the Ganglia bootstrap action by adding the following parameter to your cluster call:

  ```bash
  --bootstrap-action s3://elasticmapreduce/bootstrap-actions/install-ganglia
  ```

The following command illustrates the use of the `bootstrap-action` parameter when starting a new cluster. In this example, you start the Word Count sample cluster provided by Amazon EMR and launch three instances.

In the directory where you installed the Amazon EMR CLI, type the following command.

**Note**

The Hadoop streaming syntax is different between Hadoop 1.x and Hadoop 2.x.

For Hadoop 2.x, use the following command:
For Hadoop 1.x, use the following command:

- **Linux, UNIX, and Mac OS X users:**

```
./elastic-mapreduce --create --alive --ami-version 3.0.3 --instance-type m1.xlarge \
--num-instances 3 --stream --arg "-files" --arg "s3://elasticmapreduce/samples/wordcount/\nwordSplitter.py" --bootstrap-action s3://elasticmapreduce/bootstrap-actions/install-ganglia \
--input s3://elasticmapreduce/samples/wordcount/input \
--output s3://mybucket/output/2014-01-16 --mapper wordSplitter.py --reducer aggregate
```

- **Windows users:**

```
ruby elastic-mapreduce --create --alive --ami-version 3.0.3 --instance-type m1.xlarge --num-instances 3 --stream --arg "-files" --arg "s3://elasticmapreduce/samples/wordcount/\nwordSplitter.py" --bootstrap-action s3://elasticmapreduce/bootstrap-actions/install-ganglia \
--input s3://elasticmapreduce/samples/wordcount/input --output s3://mybucket/output/2014-01-16 \
--mapper s3://elasticmapreduce/samples/wordcount/wordSplitter.py --reducer aggregate
```

To set the NameNode heap size using a bootstrap action

- **In the directory where you installed the Amazon EMR CLI, type the following command.**

  - **Linux, UNIX, and Mac OS X:**

    ```
    ./elastic-mapreduce --create --alive --ami-version 3.0.3 --instance-type m1.xlarge --num-instances 3 \
    --bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-daemons \
    --args --namenode-heap-size=2048,--namenode-opts=-XX:GCTimeRatio=19
    ```

  - **Windows:**

    ```
ruby elastic-mapreduce --create --alive --bootstrap-action s3://elasticmapreduce/\nbootstrap-actions/configure-daemons --args --namenode-heap-size=2048,--namenode-\nopts=-XX:GCTimeRatio=19
```

To change the maximum number of map tasks using a bootstrap action

- **In the directory where you installed the Amazon EMR CLI, type the following command.**

  - **Linux, UNIX, and Mac OS X:**

    ```
    ./elastic-mapreduce --create --alive --bootstrap-action s3://elasticmapreduce/\nbootstrap-actions/configure-daemons --args --namenode-heap-size=2048,--namenode-\nopts=-XX:GCTimeRatio=19
```
To run a command conditionally using a bootstrap action

- In the directory where you installed the Amazon EMR CLI, type the following command. Notice that the optional arguments for the --args parameter are separated with commas.
  - Linux, Unix, and Mac OS X:

        ./elastic-mapreduce --create --alive --bootstrap-action s3://elasticmapreduce/bootstrap-actions/run-if --args "instance.isMaster=true,echo running on master node"

  - Windows:

        ruby elastic-mapreduce --create --alive --bootstrap-action s3://elasticmapreduce/bootstrap-actions/run-if --args "instance.isMaster=true,echo running on master node"

To create a cluster with a custom bootstrap action

- In the directory where you installed the Amazon EMR CLI, type the following command.
  - Linux, UNIX, and Mac OS X:

        ./elastic-mapreduce --create --alive --bootstrap-action s3://elasticmapreduce/bootstrap-actions/download.sh

  - Windows:

        ruby elastic-mapreduce --create --alive --bootstrap-action s3://elasticmapreduce/bootstrap-actions/download.sh

To read settings in instance.json with a bootstrap action

This procedure uses a run-if bootstrap action to demonstrate how to execute the command line function echo to display the string running on master node by evaluating the JSON file parameter instance.isMaster in the instance.json file.

- In the directory where you installed the Amazon EMR CLI, type the following command.
  - Linux, UNIX, and Mac OS X users:

        ./elastic-mapreduce --create --alive --name "RunIf" --bootstrap-action s3://elasticmapreduce/bootstrap-actions/run-if --bootstrap-name "Run only on master"
To specify bootstrap actions, you can use the --bootstrap-action option in the create operation. The format is:

```
--bootstrap-action s3://<bucket>/<path> --bootstrap-name <name> --args "<args>
```

For example:

```
ruby elastic-mapreduce --create --alive --bootstrap-action s3://<bucket>/<path> --bootstrap-name "RunIf" --args "instance.isMaster=true,echo,‘Running on master node’"
```

### Windows users:

```
ruby elastic-mapreduce --create --alive --name "RunIf" --bootstrap-action s3://elasticmapreduce/bootstrap-actions/run-if --bootstrap-name "Run only on master" --args "instance.isMaster=true,echo,‘Running on master node’"
```

### To modify JVM settings using a bootstrap action

- In the directory where you installed the Amazon EMR CLI, type the following command.

  **Linux, UNIX, and Mac OS X users:**

  ```
  ./elastic-mapreduce --create --alive --name "JVM infinite reuse" \
  --bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hadoop \
  --bootstrap-name "Configuring infinite JVM reuse" \
  --args "-m,mapred.job.reuse.jvm.num.tasks=-1"
  ```

  **Windows users:**

  ```
  ruby elastic-mapreduce --create --alive --name "JVM infinite reuse" --bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hadoop --bootstrap-name "Configuring infinite JVM reuse" --args "-m,mapred.job.reuse.jvm.num.tasks=-1"
  ```

### Note

Amazon EMR sets the value of `mapred.job.reuse.jvm.num.tasks` to 20, but you can override it with a bootstrap action. A value of -1 means infinite reuse within a single job, and 1 means do not reuse tasks.

### To disable reducer speculative execution using a bootstrap action

- In the directory where you installed the Amazon EMR CLI, type the following command.

  **Linux, UNIX, and Mac OS X users:**

  ```
  ./elastic-mapreduce --create --alive --name "Reducer speculative execution" \
  --bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hadoop \
  --bootstrap-name "Disable reducer speculative execution" \
  --args "-m,mapred.reduce.tasks.speculative.execution=false"
  ```

  **Windows users:**

  ```
  ruby elastic-mapreduce --create --alive --name "Reducer speculative execution" \
  --bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hadoop \
  --bootstrap-name "Disable reducer speculative execution" --args "-m,mapred.reduce.tasks.speculative.execution=false"
  ```

### To disable intermediate compression or change the compression codec using a bootstrap action

- In the directory where you installed the Amazon EMR CLI, type the following command.

Specifying Bootstrap Actions

- **Linux, UNIX, and Mac OS X users:**

  ```bash
  ./elastic-mapreduce --create --alive --name "Disable compression" \
  --bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hadoop \
  --bootstrap-name "Disable compression" \
  --args "-m,mapred.compress.map.output=false" \
  --args "-m,mapred.map.output.compression.codec=org.apache.hadoop.io.compress.GzipCodec"
  ```

- **Windows users:**

  ```bash
  ruby elastic-mapreduce --create --alive --name "Disable compression" 
  --bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hadoop 
  --bootstrap-name "Disable compression" 
  --args "-m,mapred.compress.map.output=false" 
  --args "-m,mapred.map.output.compression.codec=org.apache.hadoop.io.compress.GzipCodec"
  ```

To increase the `mapred.max.tracker.failures` parameter using a bootstrap action

The following example shows how to launch a cluster and use a bootstrap action to set the value of `mapred.max.tracker.failures` to 7, instead of the default 4. This allows you to troubleshoot issues where TaskTracker nodes are being blacklisted.

- **In the directory where you installed the Amazon EMR CLI, type the following command.**

- **Linux, UNIX, and Mac OS X users:**

  ```bash
  ./elastic-mapreduce --create --alive --name "Modified mapred.max.tracker.failures" \
  --num-instances 2 --slave-instance-type m1.large --master-instance-type m1.large \
  --key-pair mykeypair --debug --log-uri s3://mybucket/logs \
  --bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hadoop \
  --bootstrap-name "Modified mapred.max.tracker.failures" \
  --args "-m,mapred.max.tracker.failures=7"
  ```

- **Windows users:**

  ```bash
  ruby elastic-mapreduce --create --alive --name "Modified mapred.max.tracker.failures" 
  --num-instances 2 --slave-instance-type m1.large 
  --master-instance-type m1.large 
  --key-pair mykeypair --debug --log-uri s3://mybucket/logs 
  --bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hadoop 
  --bootstrap-name "Modified mapred.max.tracker.failures" 
  --args "-m,mapred.max.tracker.failures=7"
  ```

To disable S3 multipart upload using a bootstrap action

This procedure explains how to disable multipart upload using the Amazon EMR CLI. The command creates a cluster in a waiting state with multipart upload disabled.

- **In the directory where you installed the Amazon EMR CLI, type the following command.**

- **Linux, UNIX, and Mac OS X users:**

  ```bash
  ./elastic-mapreduce --create --alive \
  --bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hadoop \
  --bootstrap-name "enable multipart upload" \
  --args "--efs.s3n.multipart.uploads.enabled=false"
  ```

- **Windows users:**

  ```bash
  ruby elastic-mapreduce --create --alive 
  --bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hadoop 
  --bootstrap-name "enable multipart upload" 
  --args "--efs.s3n.multipart.uploads.enabled=false"
  ```
Tagging

--tag

Manages tags associated with Amazon EMR resources.

To add tags when creating a new cluster

The following example demonstrates how to add a tag to a new cluster using the Amazon EMR CLI.

- In the directory where you installed the Amazon EMR CLI, type the following command.
  
  - Linux, UNIX, and Mac OS X users:
    ```
    ./elastic-mapreduce --create --tag "costCenter=marketing"
    ```
  
  - Windows users:
    ```
    ruby elastic-mapreduce --create --tag "costCenter=marketing"
    ```

To add tags to a running cluster

The following example demonstrates how to add two tags to a running cluster using the Amazon EMR CLI. One tag has a key named `production` with no value, and the other tag has a key named `costCenter` with a value of `marketing`.

- In the directory where you installed the Amazon EMR CLI, type the following command.
  
  - Linux, UNIX, and Mac OS X users:
    ```
    ./elastic-mapreduce --add-tags j-1234567890123 --tag production --tag "costCenter=marketing"
    ```
  
  - Windows users:
    ```
    ruby elastic-mapreduce --add-tags j-1234567890123 --tag production --tag "costCenter=marketing"
    ```

Note

Quotes are unnecessary when your tag has only a key.

If the command completes successfully, the output is similar to the following:

<table>
<thead>
<tr>
<th>TAG</th>
<th>cluster j-1234567890123</th>
<th>production</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAG</td>
<td>cluster j-1234567890123</td>
<td>costCenter</td>
</tr>
</tbody>
</table>
In addition, you can apply the same tags to multiple clusters by specifying more than one cluster identifier separated by a space, for example:

```bash
./elastic-mapreduce --add-tags j-1234567890123 j-9876543210987 --tag production --tag "costCenter=marketing"
```

**To view the tags on a cluster**

- In the directory where you installed the Amazon EMR CLI, type the following command.
  - Linux, UNIX, and Mac OS X users:
    ```bash
    ./elastic-mapreduce --jobflow "j-1234567890123" --list-tags
    ```
  - Windows users:
    ```bash
    ruby elastic-mapreduce --jobflow "j-1234567890123" --list-tags
    ```

  The output displays all the tag information about the cluster similar to the following:

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>2785</td>
</tr>
<tr>
<td>costCenter</td>
<td>marketing</td>
</tr>
</tbody>
</table>

**To remove tags from a cluster**

The following example demonstrates how to remove one tag from a cluster using the Amazon EMR CLI.

- In the directory where you installed the Amazon EMR CLI, type the following command.
  - Linux, UNIX, and Mac OS X users:
    ```bash
    ./elastic-mapreduce --remove-tags j-1234567890123 --tag "costCenter=marketing"
    ```
  - Windows users:
    ```bash
    ruby elastic-mapreduce --remove-tags j-1234567890123 --tag "costCenter=marketing"
    ```

  In addition, you can remove all tags from a cluster by specifying only the cluster identifier, as shown in the following example:

  ```bash
  ./elastic-mapreduce --remove-tags j-1234567890123
  ```

  Also, you can remove a tag from a cluster using only its key name, without quotes, when the value does not matter, as shown in the following example:

  ```bash
  ./elastic-mapreduce --remove-tags j-1234567890123 --tag costCenter
  ```
Terminating Job Flows

--set-termination-protection TERMINATION_PROTECTION_STATE

Enables or disables termination protection on the specified cluster or clusters. To enable termination protection, set this value to true. To disable termination protection, set this value to false.

--terminate

Terminates the specified cluster or clusters.

To configure termination protection for a new cluster

- To enable termination protection using the Amazon EMR CLI, specify --set-termination-protection true during the cluster creation call. If the parameter is not used, termination protection is disabled. You can also type --set-termination-protection false to disable protection. The following example shows setting termination protection on a cluster running the WordCount sample application.

In the directory where you installed the Amazon EMR CLI, type the following command.

Note
The Hadoop streaming syntax shown in the following examples is different between Hadoop 1.x and Hadoop 2.x.

For Hadoop 2.x, type the following command:

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --alive --ami-version 3.0.3 \
--instance-type m1.xlarge --num-instances 2 \
--stream --arg "-files" --arg "s3://elasticmapreduce/samples/wordcount/wordSplitter.py" \
--input s3://elasticmapreduce/samples/wordcount/input \
--output s3://mybucket/output/2014-01-16 --mapper wordSplitter.py --reducer aggregate \
--set-termination-protection true
```

- Windows users:

```
ruby elastic-mapreduce --create --alive --ami-version 3.0.3 --instance-type m1.xlarge \
--num-instances 2 --stream --arg "-files" --arg "s3://elasticmapreduce/samples/wordcount/wordSplitter.py" --input s3://elasticmapreduce/samples/wordcount/input --output s3://mybucket/output/2014-01-16 --mapper wordSplitter.py --reducer aggregate \
--set-termination-protection true
```

For Hadoop 1.x, type the following command:

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --alive \
--instance-type m1.xlarge --num-instances 2 --stream \
--input s3://elasticmapreduce/samples/wordcount/input \
--output s3://myawsbucket/wordcount/output/2011-03-25 \
--mapper s3://elasticmapreduce/samples/wordcount/wordSplitter.py --reducer aggregate \
--set-termination-protection true
```

- Windows users:
To configure termination protection for a running cluster

- Set the `--set-termination-protection` flag to `true`. This is shown in the following example, where `JobFlowID` is the identifier of the cluster on which to enable termination protection.

In the directory where you installed the Amazon EMR CLI, type the following command.

- Linux, UNIX, and Mac OS X users:
  ```sh
  ./elastic-mapreduce --set-termination-protection true --jobflow JobFlowID
  ```
- Windows users:
  ```sh
  ruby elastic-mapreduce --set-termination-protection true --jobflow JobFlowID
  ```

To terminate an unprotected cluster

To terminate an unprotected cluster using the Amazon EMR CLI, type the `--terminate` parameter and specify the cluster to terminate.

- In the directory where you installed the Amazon EMR CLI, type the following from command.

  - Linux, UNIX, and Mac OS X users:
    ```sh
    ./elastic-mapreduce --terminate JobFlowID
    ```
  - Windows users:
    ```sh
    ruby elastic-mapreduce --terminate JobFlowID
    ```

To terminate a protected cluster

1. Disable termination protection by setting the `--set-termination-protection` parameter to `false`. This is shown in the following example, where `JobFlowID` is the identifier of the cluster on which to disable termination protection.

   ```sh
   elastic-mapreduce --set-termination-protection false --jobflow JobFlowID
   ```

2. Terminate the cluster using the `--terminate` parameter and the cluster identifier of the cluster to terminate.

   In the directory where you installed the Amazon EMR CLI, type the following command.

   - Linux, UNIX, and Mac OS X users:
     ```sh
     ./elastic-mapreduce --terminate JobFlowID
     ```
   - Windows users:
Using S3DistCp

When you call S3DistCp, you can specify options that change how it copies and compresses data. For more information about the options available for S3DistCp, see S3DistCp Options (p. 387).

To add a S3DistCp step to a cluster

- Add a step to the cluster that calls S3DistCp, passing in the parameters that specify how S3DistCp should perform the copy operation.

The following example copies daemon logs from Amazon S3 to hdfs:///output.

In this CLI command:

- `--jobflow` specifies the cluster to add the copy step to.
- `--jar` is the location of the S3DistCp JAR file.
- `--args` is a comma-separated list of the option name-value pairs to pass in to S3DistCp. For a complete list of the available options, see S3DistCp Options (p. 387). You can also specify the options singly, using multiple `--arg` parameters. Both forms are shown in examples below.

You can use either the `--args` or `--arg` syntax to pass options into the cluster step. The `--args` parameter is a convenient way to pass in several `--arg` parameters at one time. It splits the string passed in on comma (,) characters to parse them into arguments. This syntax is shown in the following example. Note that the value passed in by `--args` is enclosed in single quotes ('). This prevents asterisks (*) and any other special characters in any regular expressions from being expanded by the Linux shell.

In the directory where you installed the Amazon EMR CLI, type the following command.

- **Linux, UNIX, and Mac OS X users:**

  ```
  ruby elastic-mapreduce --terminate JobFlowID
  ```

- **Windows users:**

  ```
  ruby elastic-mapreduce --terminate JobFlowID
  ```

If the value of a S3DistCp option contains a comma, you cannot use `--args`, and must use instead individual `--arg` parameters to pass in the S3DistCp option names and values. Only the `--src` and `--dest` arguments are required. Note that the option values are enclosed in single quotes ('). This prevents asterisks (*) and any other special characters in any regular expressions from being expanded by the Linux shell.

- **Linux, UNIX, and Mac OS X users:**

  ```
  ruby elastic-mapreduce --jobflow JobFlowID --jar /home/hadoop/lib/emr-s3distcp-1.0.jar
  --arg "S3DistCp-OptionName1=S3DistCp-OptionValue1, S3DistCp-OptionName2=S3DistCp-OptionValue2, S3DistCp-OptionName3=S3DistCp-OptionValue3"
  ```

- **Windows users:**

  ```
  ruby elastic-mapreduce --jobflow JobFlowID --jar /home/hadoop/lib/emr-s3distcp-1.0.jar --args "S3DistCp-OptionName1=S3DistCp-OptionValue1,S3DistCp-OptionName2=S3DistCp-OptionValue2,S3DistCp-OptionName3=S3DistCp-OptionValue3"
  ```
Using S3DistCp

```bash
./elastic-mapreduce --jobflow JobFlowID --jar \
/home/hadoop/lib/emr-s3distcp-1.0.jar \
--arg S3DistCp-OptionName1 --arg 'S3DistCp-OptionValue1' \
--arg S3DistCp-OptionName2 --arg 'S3DistCp-OptionValue2' \
--arg S3DistCp-OptionName3 --arg 'S3DistCp-OptionValue3'
```

- Windows users:

```bash
ruby elastic-mapreduce --jobflow JobFlowID --jar /home/hadoop/lib/emr-s3distcp-1.0.jar \
--arg "S3DistCp-OptionName1" --arg "S3DistCp-OptionValue1" --arg "S3DistCp-OptionName2" --arg "S3DistCp-OptionValue2" --arg "S3DistCp-OptionName3" --arg "S3DistCp-OptionValue3"
```

**Example Specify an option value that contains a comma**

In this example, `--srcPattern` is set to `.*[a-zA-Z,]+`. The inclusion of a comma in the `--srcPattern` regular expression requires the use of individual `--arg` parameters.

- Linux, UNIX, and Mac OS X users:

```bash
./elastic-mapreduce --jobflow j-3GYXXXXXX9IOJ --jar \
/home/hadoop/lib/emr-s3distcp-1.0.jar \
--arg --s3Endpoint --arg 's3-eu-west-1.amazonaws.com' \
--arg --src --arg 's3://myawsbucket/logs/j-3GYXXXXXX9IOJ/node/' \
--arg --dest --arg 'hdfs:///output' \
--arg --srcPattern --arg '.*[a-zA-Z,]+'`}

- Windows users:

```bash
ruby elastic-mapreduce --jobflow j-3GYXXXXXX9IOJ --jar /home/hadoop/lib/emr-s3distcp-1.0.jar \
--arg --s3Endpoint --arg "s3-eu-west-1.amazonaws.com" --arg --src --arg "s3://myawsbucket/logs/j-3GYXXXXXX9IOJ/node/" --arg --dest --arg "hdfs:///output" --arg --srcPattern --arg ".*[a-zA-Z,]+"
```

**Example Copy log files from Amazon S3 to HDFS**

This example illustrates how to copy log files stored in an Amazon S3 bucket into HDFS. In this example the `--srcPattern` option is used to limit the data copied to the daemon logs.

- Linux, UNIX, and Mac OS X users:

```bash
./elastic-mapreduce --jobflow j-3GYXXXXXX9IOJ --jar \
/home/hadoop/lib/emr-s3distcp-1.0.jar \
--args "--src,s3://mybucket/logs/j-3GYXXXXXX9IOJ/node/,--dest,hdfs:///output,--srcPattern,.*daemons.*-hadoop-.*"`}

- Windows users:

```bash
ruby elastic-mapreduce --jobflow j-3GYXXXXXX9IOJ --jar /home/hadoop/lib/emr-s3distcp-1.0.jar \
--args "--src,s3://myawsbucket/logs/j-3GYXXXXXX9IOJ/node/,--dest,hdfs:///output,--srcPattern,.*daemons.*-hadoop-.*"`}
Example Load Amazon CloudFront logs into HDFS

This example loads Amazon CloudFront logs into HDFS. In the process it changes the compression format from Gzip (the CloudFront default) to LZO. This is useful because data compressed using LZO can be split into multiple maps as it is decompressed, so you don't have to wait until the compression is complete, as you do with Gzip. This provides better performance when you analyze the data using Amazon EMR. This example also improves performance by using the regular expression specified in the --groupBy option to combine all of the logs for a given hour into a single file. Amazon EMR clusters are more efficient when processing a few, large, LZO-compressed files than when processing many, small, Gzip-compressed files. To split LZO files, you must index them and use the hadoop-lzo third party library. For more information, see How to Process Compressed Files (p. 39).

In the directory where you installed the Amazon EMR CLI, type the following command.

- Linux, UNIX, and Mac OS X users:
  ```bash
  ./elastic-mapreduce --jobflow j-3GYXXXXXX9IOK --jar 
  /home/hadoop/lib/emr-s3distcp-1.0.jar 
  ```

- Windows users:
  ```bash
  ruby elastic-mapreduce --jobflow j-3GYXXXXXX9IOK --jar /home/hadoop/lib/emr-
  s3distcp-1.0.jar --args "--src,s3://myawsbucket/cf,--dest,hdfs:///local,--
  outputCodec,lzo,--deleteOnSuccess"
  ```

Consider the case in which the preceding example is run over the following CloudFront log files.

- s3://myawsbucket/cf/XABCD12345678.2012-02-23-01.HLUS3JKx.gz
- s3://myawsbucket/cf/XABCD12345678.2012-02-23-01.I9CNAZrg.gz
- s3://myawsbucket/cf/XABCD12345678.2012-02-23-02.YRRwERSA.gz
- s3://myawsbucket/cf/XABCD12345678.2012-02-23-02.dshVLXFE.gz

S3DistCp copies, concatenates, and compresses the files into the following two files, where the file name is determined by the match made by the regular expression.

- hdfs:///local/2012-02-23-01.lzo
- hdfs:///local/2012-02-23-02.lzo

AWS EMR Command Line Interface Releases (Deprecated)

**Note**

The Amazon EMR CLI is no longer under feature development. Customers are encouraged to use the Amazon EMR commands in the AWS CLI instead.

The following table lists the releases and changes in Amazon EMR CLI versions. The Amazon EMR CLI uses the release date as its version number.
<table>
<thead>
<tr>
<th>Release Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2013-12-10</td>
<td>Adds support for Impala on Amazon EMR. For more information, see Impala (p. 300).</td>
</tr>
<tr>
<td>2013-12-02</td>
<td>Adds support for Amazon EMR tags. For more information, see Tag Clusters (p. 171).</td>
</tr>
<tr>
<td>2013-10-07</td>
<td>Replaces the versioned HBase path with a version-less symlink so that HBase can be installed and used for both Hadoop 1.x and Hadoop 2.x.</td>
</tr>
<tr>
<td>2013-07-08</td>
<td>Fixes a bug that ignores any hard-coded Pig version number and incorrectly uses the latest Pig version.</td>
</tr>
<tr>
<td>2013-03-19</td>
<td>Improved support for launching clusters on third-party applications with a new <code>--supported-product</code> parameter that accepts custom user arguments.</td>
</tr>
<tr>
<td>2012-12-17</td>
<td>Adds support for IAM roles.</td>
</tr>
<tr>
<td>2012-09-18</td>
<td>Adds support for setting the visibility of clusters for IAM users with the <code>--visible-to-all-users</code> and <code>--set-visible-to-all-users</code> flags.</td>
</tr>
<tr>
<td>2012-08-22</td>
<td>Improved SSL certificate verification.</td>
</tr>
<tr>
<td>2012-07-30</td>
<td>Adds support for Hadoop 1.0.3.</td>
</tr>
<tr>
<td>2012-07-09</td>
<td>Adds support for specifying the major and minor AMI version and automatically getting the AMI that matches those specifications and contains the latest patches.</td>
</tr>
<tr>
<td>2012-06-12</td>
<td>Adds support for HBase and MapR.</td>
</tr>
<tr>
<td>2012-04-09</td>
<td>Adds support for Pig 0.9.1, Pig versioning, and Hive 0.7.1.4.</td>
</tr>
<tr>
<td>2012-03-13</td>
<td>Adds support for Hive 0.7.1.3.</td>
</tr>
<tr>
<td>2012-02-28</td>
<td>Adds support for Hive 0.7.1.2.</td>
</tr>
<tr>
<td>2011-12-08</td>
<td>Adds support for Amazon Machine Image (AMI) versioning, Hadoop 0.20.205, Hive 0.7.1, and Pig 0.9.1. The default AMI version is the latest AMI version available.</td>
</tr>
<tr>
<td>2011-11-30</td>
<td>Fixes support for Elastic IP addresses.</td>
</tr>
<tr>
<td>2011-08-08</td>
<td>Adds support for running a cluster on Spot Instances.</td>
</tr>
<tr>
<td>2011-01-24</td>
<td>Fixes bugs in the <code>--json</code> command processing and the list option.</td>
</tr>
<tr>
<td>2011-12-08</td>
<td>Adds support for Hive 0.7.</td>
</tr>
<tr>
<td>2011-11-11</td>
<td>Fixes issues in the processing of pig and hive arguments and the <code>--main-class</code> argument to the custom jar step.</td>
</tr>
<tr>
<td>Release Date</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>2011-10-19</td>
<td>Adds support for resizing running clusters. Substantially reworked processing arguments to be more consistent and unit testable.</td>
</tr>
<tr>
<td>2011-09-16</td>
<td>Adds support for fetching files from Amazon EMR.</td>
</tr>
<tr>
<td>2011-06-02</td>
<td>Adds support for Hadoop 0.20, Hive 0.5 and Pig 0.6.</td>
</tr>
<tr>
<td>2011-04-07</td>
<td>Adds support for bootstrap actions.</td>
</tr>
</tbody>
</table>

To display the version of the Amazon EMR CLI currently installed

- In the directory where you installed the Amazon EMR CLI, type the following command.
  - Linux, UNIX, and Mac OS X users:
    ```
    ./elastic-mapreduce --version
    ```
  - Windows users:
    ```
    ruby elastic-mapreduce --version
    ```

If the CLI is correctly installed and the credentials properly configured, the CLI should display its version number represented as a date. The output should look similar to the following:

```
Version 2012-12-17
```
## Document History

The following table describes the important changes to the Amazon EMR Developer Guide since the last 3.x release version of Amazon EMR.

**API version:** 2009-03-31

**Latest documentation update:** June 16, 2015

<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache Spark support</td>
<td>Amazon EMR natively supports Apache Spark. For more information, see Apache Spark (p. 288).</td>
<td>June 16, 2015</td>
</tr>
<tr>
<td>AMI 3.8.0</td>
<td>Amazon EMR supports AMI 3.8.0. For more information, see AMI Versions Supported in Amazon EMR Versions 2.x and 3.x (p. 73).</td>
<td>June 10, 2015</td>
</tr>
<tr>
<td>AMI 3.7.0</td>
<td>Amazon EMR supports AMI 3.7.0. For more information, see AMI Versions Supported in Amazon EMR Versions 2.x and 3.x (p. 73).</td>
<td>April 21, 2015</td>
</tr>
<tr>
<td>D2 Instances</td>
<td>Support for next generation Amazon EC2 dense-storage instances. For more information, see D2 Instances in the Amazon EC2 User Guide for Linux Instances and Plan and Configure EC2 Instances (p. 140)</td>
<td>April 2, 2015</td>
</tr>
<tr>
<td>AWS CLI Parameter Values for Amazon EMR</td>
<td>You can now set parameter values for certain EMR subcommands using the CLI or the configuration file. For more information, see Specifying Parameter Values in AWS CLI for Amazon EMR (p. 587)</td>
<td>April 2, 2015</td>
</tr>
<tr>
<td>EMRFS support for Amazon S3 client-side encryption</td>
<td>EMRFS natively supports Amazon S3 client-side encryption. For more information, see Amazon S3 Client-Side Encryption (p. 63).</td>
<td>March 25, 2015</td>
</tr>
<tr>
<td>AMI 3.6.0</td>
<td>Amazon EMR supports AMI 3.6.0. For more information, see AMI Versions Supported in Amazon EMR Versions 2.x and 3.x (p. 73).</td>
<td>March 24, 2015</td>
</tr>
<tr>
<td>AMI 3.5.0</td>
<td>Amazon EMR supports AMI 3.5.0. For more information, see AMI Versions Supported in Amazon EMR Versions 2.x and 3.x (p. 73).</td>
<td>March 10, 2015</td>
</tr>
<tr>
<td>AMI 2.4.11 and 3.4.0</td>
<td>Amazon EMR supports AMI 3.4.0. For more information, see AMI Versions Supported in Amazon EMR Versions 2.x and 3.x (p. 73).</td>
<td>February 26, 2015</td>
</tr>
<tr>
<td>AMI 2.4.10</td>
<td>Amazon EMR supports AMI 2.4.10. For more information, see AMI Versions Supported in Amazon EMR Versions 2.x and 3.x (p. 73).</td>
<td>February 13, 2015</td>
</tr>
<tr>
<td>AMI 3.3.2</td>
<td>Amazon EMR supports AMI 3.3.1. For more information, see AMI Versions Supported in Amazon EMR Versions 2.x and 3.x (p. 73).</td>
<td>February 4, 2015</td>
</tr>
<tr>
<td>Change</td>
<td>Description</td>
<td>Release Date</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>--------------</td>
</tr>
<tr>
<td>AMI 3.3.1</td>
<td>Amazon EMR supports AMI 3.3.1. For more information, see AMI Versions Supported in Amazon EMR Versions 2.x and 3.x (p. 73).</td>
<td>November 20, 2014</td>
</tr>
<tr>
<td>Hue support</td>
<td>Amazon EMR supports Hue, an open-source application for interacting with clusters. For more information, see Configure Hue to View, Query, or Manipulate Data (p. 348).</td>
<td>November 6, 2014</td>
</tr>
<tr>
<td>Consistent view</td>
<td>Amazon EMR supports EMRFS consistent view. For more information, see Consistent View (p. 46).</td>
<td>September 17, 2014</td>
</tr>
<tr>
<td>AMIs 2.4.8, 3.1.2, and 3.2.1</td>
<td>Amazon EMR supports these new images. For more information, see AMI Versions Supported in Amazon EMR Versions 2.x and 3.x (p. 73).</td>
<td>September 16, 2014</td>
</tr>
<tr>
<td>AMI 3.1.1</td>
<td>Amazon EMR supports AMI 3.1.1. For more information, see AMI Versions Supported in Amazon EMR Versions 2.x and 3.x (p. 73).</td>
<td>August 15, 2014</td>
</tr>
<tr>
<td>AMI 2.4.7</td>
<td>Amazon EMR supports AMI 2.4.7. For more information, see AMI Versions Supported in Amazon EMR Versions 2.x and 3.x (p. 73).</td>
<td>July 30, 2014</td>
</tr>
<tr>
<td>AMI 2.4.6</td>
<td>Amazon EMR supports AMI 2.4.6. For more information, see AMI Versions Supported in Amazon EMR Versions 2.x and 3.x (p. 73).</td>
<td>May 15, 2014</td>
</tr>
<tr>
<td>AMI 3.1.0</td>
<td>Amazon EMR supports AMI 3.1.0. For more information, see AMI Versions Supported in Amazon EMR Versions 2.x and 3.x (p. 73).</td>
<td>May 15, 2014</td>
</tr>
<tr>
<td>AMI 2.4.5</td>
<td>Amazon EMR supports AMI 2.4.5. For more information, see AMI Versions Supported in Amazon EMR Versions 2.x and 3.x (p. 73).</td>
<td>March 27, 2014</td>
</tr>
<tr>
<td>AMI 3.0.4</td>
<td>Amazon EMR supports AMI 3.0.4 and a connector for Amazon Kinesis. For more information, see AMI Versions Supported in Amazon EMR Versions 2.x and 3.x (p. 73).</td>
<td>February 20, 2014</td>
</tr>
<tr>
<td>AMI 3.0.3</td>
<td>Amazon EMR supports AMI 3.0.3. For more information, see AMI Versions Supported in Amazon EMR Versions 2.x and 3.x (p. 73).</td>
<td>February 11, 2014</td>
</tr>
<tr>
<td>Hive 0.11.0.2</td>
<td>Amazon EMR supports Hive 0.11.0.2. For more information, see Supported Hive Versions (p. 271).</td>
<td>February 11, 2014</td>
</tr>
<tr>
<td>Impala 1.2.1</td>
<td>Amazon EMR supports Impala 1.2.1 with Hadoop 2. For more information, see Impala (p. 300).</td>
<td>December 12, 2013</td>
</tr>
<tr>
<td>AMI 3.0.2</td>
<td>Amazon EMR supports AMI 3.0.2. For more information, see AMI Versions Supported in Amazon EMR Versions 2.x and 3.x (p. 73).</td>
<td>December 12, 2013</td>
</tr>
<tr>
<td>Amazon EMR tags</td>
<td>Amazon EMR supports tagging on Amazon EMR clusters. For more information, see Tag Clusters (p. 171).</td>
<td>December 5, 2013</td>
</tr>
<tr>
<td>Change</td>
<td>Description</td>
<td>Release Date</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>CLI version 2013-12-02</td>
<td>Adds support for Amazon EMR tags. For more information, see AWS EMR Command Line Interface Releases (Deprecated) (p. 645).</td>
<td>December 5, 2013</td>
</tr>
<tr>
<td>AMI 3.0.1</td>
<td>Amazon EMR supports AMI 3.0.1. For more information, see AMI Versions Supported in Amazon EMR Versions 2.x and 3.x (p. 73).</td>
<td>November 8, 2013</td>
</tr>
<tr>
<td>New Amazon EMR console</td>
<td>A new management console is available for Amazon EMR. The new console is much faster and has powerful new features, including:</td>
<td>November 6, 2013</td>
</tr>
<tr>
<td></td>
<td>• Resizing a running cluster (that is, adding or removing instances)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Cloning the launch configurations for running or terminated clusters</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Hadoop 2 support, including custom Amazon CloudWatch metrics</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Targeting specific Availability Zones</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Creating clusters with IAM roles</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Submitting multiple steps (before and after cluster creation)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• New console help portal with integrated documentation search</td>
<td></td>
</tr>
<tr>
<td>MapR 3.0.2</td>
<td>Amazon EMR supports MapR 3.0.2. For more information, see Using the MapR Distribution for Hadoop (p. 177).</td>
<td>November 6, 2013</td>
</tr>
<tr>
<td>Hadoop 2.2.0</td>
<td>Amazon EMR supports Hadoop 2.2.0. For more information, see Hadoop 2.2.0 New Features (p. 123).</td>
<td>October 29, 2013</td>
</tr>
<tr>
<td>AMI 3.0.0</td>
<td>Amazon EMR supports AMI 3.0.0. For more information, see AMI Versions Supported in Amazon EMR Versions 2.x and 3.x (p. 73).</td>
<td>October 29, 2013</td>
</tr>
<tr>
<td>CLI version 2013-10-07</td>
<td>Maintenance update for the Amazon EMR CLI. For more information, see AWS EMR Command Line Interface Releases (Deprecated) (p. 645).</td>
<td>October 7, 2013</td>
</tr>
<tr>
<td>AMI 2.4.2</td>
<td>Amazon EMR supports AMI 2.4.2 For more information, see AMI Versions Supported in Amazon EMR Versions 2.x and 3.x (p. 73).</td>
<td>October 7, 2013</td>
</tr>
<tr>
<td>AMI 2.4.1</td>
<td>Amazon EMR supports AMI 2.4.1 For more information, see AMI Versions Supported in Amazon EMR Versions 2.x and 3.x (p. 73).</td>
<td>August 20, 2013</td>
</tr>
<tr>
<td>Hive 0.11.0.1</td>
<td>Amazon EMR supports Hive 0.11.0.1. For more information, see Supported Hive Versions (p. 271).</td>
<td>August 2, 2013</td>
</tr>
<tr>
<td>Hive 0.11.0</td>
<td>Amazon EMR supports Hive 0.11.0. For more information, see Supported Hive Versions (p. 271).</td>
<td>August 1, 2013</td>
</tr>
<tr>
<td>Pig 0.11.1.1</td>
<td>Amazon EMR supports Pig 0.11.1.1. For more information, see Supported Pig Versions (p. 318).</td>
<td>August 1, 2013</td>
</tr>
<tr>
<td>Change</td>
<td>Description</td>
<td>Release Date</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>AMI 2.4</td>
<td>Amazon EMR supports AMI 2.4. For more information, see AMI Versions Supported in Amazon EMR Versions 2.x and 3.x (p. 73).</td>
<td>August 1, 2013</td>
</tr>
<tr>
<td>MapR 2.1.3</td>
<td>Amazon EMR supports MapR 2.1.3. For more information, see Using the MapR Distribution for Hadoop (p. 177).</td>
<td>August 1, 2013</td>
</tr>
<tr>
<td>MapR M7 Edition</td>
<td>Amazon EMR supports MapR M7 Edition. For more information, see Using the MapR Distribution for Hadoop (p. 177).</td>
<td>July 11, 2013</td>
</tr>
<tr>
<td>CLI version 2013-07-08</td>
<td>Maintenance update to the Amazon EMR CLI version 2013-07-08. For more information, see AWS EMR Command Line Interface Releases (Deprecated) (p. 645).</td>
<td>July 11, 2013</td>
</tr>
<tr>
<td>Pig 0.11.1</td>
<td>Amazon EMR supports Pig 0.11.1. Pig 0.11.1 adds support for JDK 7, Hadoop 2, and more. For more information, see Supported Pig Versions (p. 318).</td>
<td>July 1, 2013</td>
</tr>
<tr>
<td>Hive 0.8.1.8</td>
<td>Amazon EMR supports Hive 0.8.1.8. For more information, see Supported Hive Versions (p. 271).</td>
<td>June 18, 2013</td>
</tr>
<tr>
<td>AMI 2.3.6</td>
<td>Amazon EMR supports AMI 2.3.6. For more information, see AMI Versions Supported in Amazon EMR Versions 2.x and 3.x (p. 73).</td>
<td>May 17, 2013</td>
</tr>
<tr>
<td>Hive 0.8.1.7</td>
<td>Amazon EMR supports Hive 0.8.1.7. For more information, see Supported Hive Versions (p. 271).</td>
<td>May 2, 2013</td>
</tr>
<tr>
<td>Improved documentation organization, new table of contents, and new topics</td>
<td>Updated documentation organization with a restructured table of contents and many new topics for better ease of use and to accommodate customer feedback.</td>
<td>April 29, 2013</td>
</tr>
<tr>
<td>AMI 2.3.5</td>
<td>Amazon EMR supports AMI 2.3.5. For more information, see AMI Versions Supported in Amazon EMR.</td>
<td>April 26, 2013</td>
</tr>
<tr>
<td>M1 Medium Amazon EC2 Instances</td>
<td>Amazon EMR supports m1.medium instances. For more information, see Hadoop 2.2.0 and 2.4.0 Default Configuration (p. 548).</td>
<td>April 18, 2013</td>
</tr>
<tr>
<td>MapR 2.1.2</td>
<td>Amazon Elastic MapReduce supports MapR 2.1.2. For more information, see Using the MapR Distribution for Hadoop (p. 177).</td>
<td>April 18, 2013</td>
</tr>
<tr>
<td>AMI 2.3.4</td>
<td>Deprecated.</td>
<td>April 16, 2013</td>
</tr>
<tr>
<td>AWS GovCloud (US)</td>
<td>Adds support for AWS GovCloud (US). For more information, see AWS GovCloud (US).</td>
<td>April 9, 2013</td>
</tr>
<tr>
<td>Supported Product User Arguments</td>
<td>Improved support for launching job flows on third-party applications with a new --supported-product CLI option that accepts custom user arguments. For more information, see Launch an Amazon EMR cluster with MapR using the console (p. 177).</td>
<td>March 19, 2013</td>
</tr>
<tr>
<td>Change</td>
<td>Description</td>
<td>Release Date</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Amazon VPC</td>
<td>Amazon Elastic MapReduce supports two platforms on which you can launch the EC2 instances of your job flow: EC2-Classic and EC2-VPC. For more information, see Amazon VPC.</td>
<td>March 11, 2013</td>
</tr>
<tr>
<td>AMI 2.3.3</td>
<td>Amazon Elastic MapReduce supports AMI 2.3.3. For more information, see AMI Versions Supported in Amazon EMR.</td>
<td>March 1, 2013</td>
</tr>
<tr>
<td>High I/O Instances</td>
<td>Amazon Elastic MapReduce supports hi1.4xlarge instances. For more information, see Hadoop 2.2.0 and 2.4.0 Default Configuration (p. 548).</td>
<td>February 14, 2013</td>
</tr>
<tr>
<td>AMI 2.3.2</td>
<td>Amazon Elastic MapReduce supports AMI 2.3.2. For more information, see AMI Versions Supported in Amazon EMR.</td>
<td>February 7, 2013</td>
</tr>
<tr>
<td>New introduction and tutorial</td>
<td>Added sections that describe Amazon EMR and a tutorial that walks you through your first streaming cluster.</td>
<td>January 9, 2013</td>
</tr>
<tr>
<td>CLI Reference</td>
<td>Added CLI reference. For more information, see Command Line Interface Reference for Amazon EMR (p. 587).</td>
<td>January 8, 2013</td>
</tr>
<tr>
<td>AMI 2.3.1</td>
<td>Amazon Elastic MapReduce supports AMI 2.3.1. For more information, see AMI Versions Supported in Amazon EMR.</td>
<td>December 24, 2012</td>
</tr>
<tr>
<td>High Storage Instances</td>
<td>Amazon Elastic MapReduce supports hs1.8xlarge instances. For more information, see Hadoop 2.2.0 and 2.4.0 Default Configuration (p. 548).</td>
<td>December 20, 2012</td>
</tr>
<tr>
<td>IAM Roles</td>
<td>Amazon Elastic MapReduce supports IAM Roles. For more information, see Configure IAM Roles for Amazon EMR Permissions to AWS Services (p. 234).</td>
<td>December 20, 2012</td>
</tr>
<tr>
<td>Hive 0.8.1.6</td>
<td>Amazon Elastic MapReduce supports Hive 0.8.1.6. For more information, see Supported Hive Versions (p. 271).</td>
<td>December 20, 2012</td>
</tr>
<tr>
<td>AMI 2.3.0</td>
<td>Amazon Elastic MapReduce supports AMI 2.3.0. For more information, see AMI Versions Supported in Amazon EMR.</td>
<td>December 20, 2012</td>
</tr>
<tr>
<td>AMI 2.2.4</td>
<td>Amazon Elastic MapReduce supports AMI 2.2.4. For more information, see AMI Versions Supported in Amazon EMR.</td>
<td>December 6, 2012</td>
</tr>
<tr>
<td>AMI 2.2.3</td>
<td>Amazon Elastic MapReduce supports AMI 2.2.3. For more information, see AMI Versions Supported in Amazon EMR.</td>
<td>November 30, 2012</td>
</tr>
<tr>
<td>Hive 0.8.1.5</td>
<td>Amazon Elastic MapReduce supports Hive 0.8.1.5. For more information, see Hive and Amazon EMR (EMR 3.x Releases) (p. 260).</td>
<td>November 30, 2012</td>
</tr>
<tr>
<td>Asia Pacific (Sydney)</td>
<td>Adds support for Amazon EMR in the Asia Pacific (Sydney) region.</td>
<td>November 12, 2012</td>
</tr>
<tr>
<td>Visible To All IAM Users</td>
<td>Added support making a cluster visible to all IAM users on an AWS account. For more information, see Use IAM Policies to Allow and Deny User Permissions (p. 187).</td>
<td>October 1, 2012</td>
</tr>
<tr>
<td>Change</td>
<td>Description</td>
<td>Release Date</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Hive 0.8.1.4</td>
<td>Updates the HBase client on Hive clusters to version 0.92.0 to match the version of HBase used on HBase clusters. This fixes issues that occurred when connecting to an HBase cluster from a Hive cluster.</td>
<td>September 17, 2012</td>
</tr>
</tbody>
</table>
| AMI 2.2.1 | • Fixes an issue with HBase backup functionality.  
• Enables multipart upload by default for files larger than the Amazon S3 block size specified by `fs.s3n.blockSize`. For more information, see Configure Multipart Upload for Amazon S3 (p. 33). | August 30, 2012 |
| AMI 2.1.4 | • Fixes issues in the native Amazon S3 file system.  
• Enables multipart upload by default. For more information, see Configure Multipart Upload for Amazon S3 (p. 33). | August 30, 2012 |
<p>| Hadoop 1.0.3, AMI 2.2.0, Hive 0.8.1.3, Pig 0.9.2.2 | Support for Hadoop 1.0.3. For more information, see Supported Hadoop Versions (p. 120). | August 6, 2012 |
| AMI 2.1.3 | Fixes issues with HBase. | August 6, 2012 |
| AMI 2.1.2 | Support for Amazon CloudWatch metrics when using MapR. | August 6, 2012 |
| AMI 2.1.1 | Improves the reliability of log pushing, adds support for HBase in Amazon VPC, and improves DNS retry functionality. | July 9, 2012 |
| Major-Minor AMI Versioning | Improves AMI versioning by adding support for major-minor releases. Now you can specify the major-minor version for the AMI and always have the latest patches applied. For more information, see Choose an Amazon Machine Image (AMI) (p. 69). | July 9, 2012 |
| Hive 0.8.1.2 | Fixes an issue with duplicate data in large clusters. | July 9, 2012 |
| S3DistCp 1.0.5 | Provides better support for specifying the version of S3DistCp to use. | June 27, 2012 |
| Store Data with HBase | Amazon EMR supports HBase, an open source, non-relational, distributed database modeled after Google's BigTable. For more information, see Apache HBase (p. 325). | June 12, 2012 |
| Launch a Cluster on the MapR Distribution for Hadoop | Amazon EMR supports MapR, an open, enterprise-grade distribution that makes Hadoop easier and more dependable. For more information, see Using the MapR Distribution for Hadoop (p. 177). | June 12, 2012 |
| Connect to the Master Node in an Amazon EMR Cluster | Added information about how to connect to the master node using both SSH and a SOCKS proxy. For more information, see Connect to the Cluster (p. 456). | June 12, 2012 |
| Hive 0.8.1 | Amazon Elastic MapReduce supports Hive 0.8.1. For more information, see Hive and Amazon EMR (EMR 3.x Releases) (p. 260). | May 30, 2012 |</p>
<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Release Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>HParser</td>
<td>Added information about running Informatica HParser on Amazon EMR. For more information, see Parse Data with HParser (p. 176).</td>
<td>April 30, 2012</td>
</tr>
<tr>
<td>AMI 2.0.5</td>
<td>Enhancements to performance and other updates. For more information, see AMI Versions Supported in Amazon EMR Versions 2.x and 3.x (p. 73).</td>
<td>April 19, 2012</td>
</tr>
<tr>
<td>Pig 0.9.2</td>
<td>Amazon Elastic MapReduce supports Pig 0.9.2. Pig 0.9.2 adds support for user-defined functions written in Python and other improvements. For more information, see Pig Version Details (p. 320).</td>
<td>April 9, 2012</td>
</tr>
<tr>
<td>Pig versioning</td>
<td>Amazon Elastic MapReduce supports the ability to specify the Pig version when launching a cluster. For more information, see Apache Pig (p. 318).</td>
<td>April 9, 2012</td>
</tr>
<tr>
<td>Hive 0.7.1.4</td>
<td>Amazon Elastic MapReduce supports Hive 0.7.1.4. For more information, see Hive and Amazon EMR (EMR 3.x Releases) (p. 260).</td>
<td>April 9, 2012</td>
</tr>
<tr>
<td>AMI 1.0.1</td>
<td>Updates sources.list to the new location of the Lenny distribution in archive.debian.org.</td>
<td>April 3, 2012</td>
</tr>
<tr>
<td>Hive 0.7.1.3</td>
<td>Support for new version of Hive, version 0.7.1.3. This version adds the dynamodb.retry.duration variable which you can use to configure the timeout duration for retrying Hive queries. This version of Hive also supports setting the DynamoDB endpoint from within the Hive command-line application.</td>
<td>March 13, 2012</td>
</tr>
<tr>
<td>Support for IAM in the console</td>
<td>Support for AWS Identity and Access Management (IAM) in the Amazon EMR console. Improvements for S3DistCp and support for Hive 0.7.1.2 are also included.</td>
<td>February 28, 2012</td>
</tr>
<tr>
<td>Support for DynamoDB</td>
<td>Support for exporting and querying data stored in DynamoDB.</td>
<td>January 18, 2012</td>
</tr>
<tr>
<td>AMI 2.0.2 and Hive 0.7.1.1</td>
<td>Support for Amazon EMR AMI 2.0.2 and Hive 0.7.1.1.</td>
<td>January 17, 2012</td>
</tr>
<tr>
<td>Cluster Compute Eight Extra Large (cc2.8xlarge)</td>
<td>Support for Cluster Compute Eight Extra Large (cc2.8xlarge) instances in clusters.</td>
<td>December 21, 2011</td>
</tr>
<tr>
<td>Hadoop 0.20.205</td>
<td>Support for Hadoop 0.20.205. For more information, see Supported Hadoop Versions (p. 120).</td>
<td>December 11, 2011</td>
</tr>
<tr>
<td>Pig 0.9.1</td>
<td>Support for Pig 0.9.1. For more information see Supported Pig Versions (p. 318).</td>
<td>December 11, 2011</td>
</tr>
<tr>
<td>Change</td>
<td>Description</td>
<td>Release Date</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>AMI versioning</td>
<td>You can now specify which version of the Amazon EMR AMI to use to launch your cluster. All EC2 instances in the cluster will be initialized with the AMI version that you specify. For more information, see Choose an Amazon Machine Image (AMI) (p. 69).</td>
<td>December 11, 2011</td>
</tr>
<tr>
<td>Amazon EMR clusters on Amazon VPC</td>
<td>You can now launch Amazon EMR clusters inside of your Amazon Virtual Private Cloud (Amazon VPC) for greater control over network configuration and access. For more information, see Plan and Configure Networking (p. 145).</td>
<td>December 11, 2011</td>
</tr>
<tr>
<td>Spot Instances</td>
<td>Support for launching cluster instance groups as Spot Instances added. For more information, see When Should You Use Spot Instances? (p. 164).</td>
<td>August 19, 2011</td>
</tr>
<tr>
<td>Hive 0.7.1</td>
<td>Support for Hive 0.7.1 added. For more information, see Supported Hive Versions (p. 271).</td>
<td>July 25, 2011</td>
</tr>
<tr>
<td>Termination Protection</td>
<td>Support for a new Termination Protection feature. For more information, see Managing Cluster Termination (p. 474).</td>
<td>April 14, 2011</td>
</tr>
<tr>
<td>Tagging</td>
<td>Support for Amazon EC2 tagging. For more information, see View Cluster Instances in Amazon EC2 (p. 426).</td>
<td>March 9, 2011</td>
</tr>
<tr>
<td>IAM Integration</td>
<td>Support for AWS Identity and Access Management. For more information, see Use IAM Policies to Allow and Deny User Permissions (p. 187) and Use IAM Policies to Allow and Deny User Permissions (p. 187).</td>
<td>February 21, 2011</td>
</tr>
<tr>
<td>Environment Configuration</td>
<td>Expanded sections on Environment Configuration and Performance Tuning. For more information, see (Optional) Create Bootstrap Actions to Install Additional Software (p. 129).</td>
<td>February 21, 2011</td>
</tr>
<tr>
<td>Distributed Cache</td>
<td>For more information about using DistributedCache to upload files and libraries, see Import files with Distributed Cache (p. 35).</td>
<td>February 21, 2011</td>
</tr>
<tr>
<td>How to build modules using Amazon EMR</td>
<td>For more information, see Build Binaries Using Amazon EMR (p. 249).</td>
<td>February 21, 2011</td>
</tr>
<tr>
<td>Amazon S3 multipart upload</td>
<td>Support of Amazon S3 multipart upload through the AWS SDK for Java. For more information, see Configure Multipart Upload for Amazon S3 (p. 33).</td>
<td>January 6, 2010</td>
</tr>
<tr>
<td>Hive 0.70</td>
<td>Support for Hive 0.70 and concurrent versions of Hive 0.5 and Hive 0.7 on same cluster. Note: You need to update the Amazon EMR command line interface to resize running job flows and modify instance groups. For more information, see Hive and Amazon EMR (EMR 3.x Releases) (p. 260).</td>
<td>December 8, 2010</td>
</tr>
<tr>
<td>Change</td>
<td>Description</td>
<td>Release Date</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>JDBC Drivers for Hive</td>
<td>Support for JDBC with Hive 0.5 and Hive 0.7. For more information, see Use the Hive JDBC Driver (p. 285).</td>
<td>December 8, 2010</td>
</tr>
<tr>
<td>Support HPC</td>
<td>Support for cluster compute instances. For more information, see Plan and Configure EC2 Instances (p. 140).</td>
<td>November 14, 2010</td>
</tr>
<tr>
<td>Bootstrap Actions</td>
<td>Expanded content and samples for bootstrap actions. For more information, see (Optional) Create Bootstrap Actions to Install Additional Software (p. 129).</td>
<td>November 14, 2010</td>
</tr>
<tr>
<td>Cascading clusters</td>
<td>Description of Cascading cluster support. For more information, see Submit a Cascading Step (p. 256) and Process Data Using Cascading (p. 256).</td>
<td>November 14, 2010</td>
</tr>
<tr>
<td>Resize Running Cluster</td>
<td>Support for resizing a running cluster. New node types task and core replace slave node. For more information, see What is Amazon EMR? (p. 1), Manually Resizing a Running Cluster (p. 486), and Manually Resizing a Running Cluster (p. 486).</td>
<td>October 19, 2010</td>
</tr>
<tr>
<td>Appendix: Configuration Options</td>
<td>Expanded information on configuration options available in Amazon EMR. For more information, see Hadoop Configuration Reference (p. 543).</td>
<td>October 19, 2010</td>
</tr>
<tr>
<td>Guide revision</td>
<td>This release features a reorganization of the Amazon EMR Developer Guide.</td>
<td>October 19, 2010</td>
</tr>
</tbody>
</table>