
Amazon Elastic MapReduce

Developer Guide

API Version 2009-03-31



Amazon Elastic MapReduce: Developer Guide

Copyright © 2015 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

The following are trademarks of Amazon Web Services, Inc.: Amazon, Amazon Web Services Design, AWS, Amazon CloudFront, AWS CloudTrail, AWS CodeDeploy, Amazon Cognito, Amazon DevPay, DynamoDB, ElastiCache, Amazon EC2, Amazon Elastic Compute Cloud, Amazon Glacier, Amazon Kinesis, Kindle, Kindle Fire, AWS Marketplace Design, Mechanical Turk, Amazon Redshift, Amazon Route 53, Amazon S3, Amazon VPC, and Amazon WorkDocs. In addition, Amazon.com graphics, logos, page headers, button icons, scripts, and service names are trademarks, or trade dress of Amazon in the U.S. and/or other countries. Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon.

All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What is Amazon EMR?	1
What Can You Do with Amazon EMR?	2
Hadoop Programming on Amazon EMR	2
Data Analysis and Processing on Amazon EMR	3
Data Storage on Amazon EMR	3
Move Data with Amazon EMR	3
Amazon EMR Features	3
Resizeable Clusters	4
Pay Only for What You Use	4
Easy to Use	4
Use Amazon S3 or HDFS	4
Parallel Clusters	4
Hadoop Application Support	4
Save Money with Spot Instances	4
AWS Integration	5
Instance Options	5
MapR Support	5
Business Intelligence Tools	5
User Control	5
Management Tools	5
Security	6
How Does Amazon EMR Work?	6
Hadoop	6
Nodes	7
Steps	8
Cluster	9
What Tools are Available for Amazon EMR?	10
Learn More About Hadoop and AWS Services Used with Amazon EMR:	12
Tutorial: Get Started with Amazon EMR	13
Tutorial Overview	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	22
Submit the Hive Script as a Step	23
View the Results	23
Step 5: Query Your Data Using Hue	24
Create an SSH Tunnel to the Master Node	24
Log into Hue and Submit an Interactive Hive Query	25
(Optional) Step 6: Explore Amazon EMR	26
(Optional) Step 7: Remove the Resources Used in the Tutorial	27
Next Steps: Learn more about Amazon EMR	28
Plan an Amazon EMR Cluster	30
Choose an AWS Region	31
Choose a Region using the Console	32
Specify a Region using the AWS CLI	32
Choose a Region Using an SDK or the API	32
Choose the Number and Type of Instances	32
Calculate the HDFS Capacity of a Cluster	33
Guidelines for the Number and Type of Instances	33
Instance Groups	34
Instance Configurations	35
Ensure Capacity with Reserved Instances (Optional)	37

(Optional) Lower Costs with Spot Instances	37
Configure the Software	48
Choose an Amazon Machine Image (AMI)	48
Choose a Version of Hadoop	115
(Optional) Create Bootstrap Actions to Install Additional Software	124
File Systems Compatible with Amazon EMR	138
Access File Systems	139
EMR File System (EMRFS) (Optional)	141
Choose the Cluster Lifecycle: Long-Running or Transient	163
Prepare Input Data (Optional)	164
Types of Input Amazon EMR Can Accept	164
How to Get Data Into Amazon EMR	165
Prepare an Output Location (Optional)	175
Create and Configure an Amazon S3 Bucket	175
What formats can Amazon EMR return?	176
How to write data to an Amazon S3 bucket you don't own	177
Compress the Output of your Cluster	179
Configure Access to the Cluster	180
Create SSH Credentials for the Master Node	180
Configure IAM User Permissions	181
Set Access Policies for IAM Users	183
Configure IAM Roles for Amazon EMR	185
Configure Security Groups for Amazon EMR	193
Setting Permissions on the System Directory	200
Configure Logging and Debugging (Optional)	200
Default Log Files	201
Archive Log Files to Amazon S3	201
Enable the Debugging Tool	204
Select an Amazon VPC Subnet for the Cluster (Optional)	205
Clusters in a VPC	206
Setting Up a VPC to Host Clusters	207
Launching Clusters into a VPC	209
Restricting Permissions to a VPC Using IAM	210
Tagging Amazon EMR Clusters	211
Tag Restrictions	212
Tagging Resources for Billing	213
Adding Tags to a New Cluster	213
Adding Tags to an Existing Cluster	214
Viewing Tags on a Cluster	215
Removing Tags from a Cluster	215
Use Third Party Applications With Amazon EMR (Optional)	216
Use Business Intelligence Tools with Amazon EMR	216
Use Hunk with Amazon EMR	216
Parse Data with HParser	217
Using the MapR Distribution for Hadoop	218
Run a Hadoop Application to Process Data	228
Build Binaries Using Amazon EMR	228
JAR Requirements	230
Run a Script in a Cluster	231
Submitting a Custom JAR Step Using the AWS CLI	231
Process Data with Streaming	232
Using the Hadoop Streaming Utility	232
Submit a Streaming Step	234
Process Data Using Cascading	235
Submit a Cascading Step	236
Process Data with a Custom JAR	237
Submit a Custom JAR Step	237
Hive and Amazon EMR	240

How Amazon EMR Hive Differs from Apache Hive	240
Combine Splits Input Format	241
Log files	241
Thrift Service Ports	242
Hive Authorization	242
Hive File Merge Behavior with Amazon S3	242
ACID Transactions and Amazon S3	242
Additional Features of Hive in Amazon EMR	243
Supported Hive Versions	251
Display the Hive Version	259
Share Data Between Hive Versions	259
Submit Hive Work	260
Submit Hive Work Using the Amazon EMR Console	260
Submit Hive Work Using the AWS CLI	260
Create a Hive Metastore Outside the Cluster	262
Use the Hive JDBC Driver	264
Apache Spark	268
Use Spark Interactively or in Batch Mode	269
Create a Cluster With Spark	269
Configure Spark	270
Manually adjusting executor settings	271
Access the Spark Shell	272
Write a Spark Application	273
Scala	273
Java	274
Python	275
Adding a Spark Step	275
Overriding Spark Default Configuration Settings	277
Impala	279
What Can I Do With Impala?	279
Differences from Traditional Relational Databases	280
Differences from Hive	280
Tutorial: Launching and Querying Impala Clusters on Amazon EMR	281
Sign up for the Service	281
Launch the Cluster	281
Generate Test Data	286
Create and Populate Impala Tables	286
Query Data in Impala	287
Impala Examples Included on the Amazon EMR AMI	288
TPCDS	289
Wikipedia	290
Supported Impala Versions	291
Updates for Impala 1.2.4	291
Impala Memory Considerations	292
Using Impala with JDBC	293
Accessing Impala Web User Interfaces	293
Impala-supported File and Compression Formats	293
Impala SQL Dialect	294
Impala User-Defined Functions	294
Impala Performance Testing and Query Optimization	294
Database Schema	294
Sample Data	295
Table Size	296
Queries	296
Performance Test Results	297
Optimizing Queries	300
Apache Pig	302
Supported Pig Versions	302

Pig Version Details	304
Additional Pig Functions	306
Interactive and Batch Pig Clusters	306
Submit Pig Work	306
Submit Pig Work Using the Amazon EMR Console	306
Submit Pig Work Using the AWS CLI	307
Call User Defined Functions from Pig	308
Call JAR files from Pig	308
Call Python/Jython Scripts from Pig	308
Apache HBase	310
What Can I Do with HBase?	310
Supported HBase Versions	311
HBase Cluster Prerequisites	311
Install HBase on an Amazon EMR Cluster	312
Connect to HBase Using the Command Line	317
Create a Table	318
Put a Value	318
Get a Value	318
Back Up and Restore HBase	318
Back Up and Restore HBase Using the Console	319
Back Up and Restore HBase Using the AWS CLI	321
Terminate an HBase Cluster	324
Configure HBase	324
Configure HBase Daemons	324
Configure HBase Site Settings	326
HBase Site Settings to Optimize	328
Access HBase Data with Hive	329
View the HBase User Interface	330
View HBase Log Files	330
Monitor HBase with CloudWatch	331
Monitor HBase with Ganglia	332
Configure Hue to View, Query, or Manipulate Data	334
What is Hue?	334
Create a Cluster with Hue Installed	335
Launch the Hue Web Interface	336
Use Hue with a Remote Database in Amazon RDS	336
Troubleshooting	339
Advanced Configurations for Hue	339
Configure Hue for LDAP Users	340
Metastore Manager Restrictions	343
Analyze Amazon Kinesis Data	344
What Can I Do With Amazon EMR and Amazon Kinesis Integration?	344
Checkpointed Analysis of Amazon Kinesis Streams	345
Provisioned IOPS Recommendations for Amazon DynamoDB Tables	345
Performance Considerations	346
Schedule Amazon Kinesis Analysis with Amazon EMR Clusters	346
Tutorial: Analyzing Amazon Kinesis Streams with Amazon EMR and Hive	346
Sign Up for the Service	347
Create an Amazon Kinesis Stream	347
Create an Amazon DynamoDB Table	347
Download Log4J Appender for Amazon Kinesis Sample Application, Sample Credentials File, and Sample Log File	348
Start Amazon Kinesis Publisher Sample Application	350
Launch the Cluster	352
Run the Ad-hoc Hive Query	356
Running Queries with Checkpoints	359
Scheduling Scripted Queries	360
Tutorial: Analyzing Amazon Kinesis Streams with Amazon EMR and Pig	361

Sign Up for the Service	361
Create an Amazon Kinesis Stream	362
Create an DynamoDB Table	362
Download Log4J Appender for Amazon Kinesis Sample Application, Sample Credentials File, and Sample Log File	363
Start Amazon Kinesis Publisher Sample Application	364
Launch the Cluster	366
Run the Pig Script	370
Scheduling Scripted Queries	374
Extract, Transform, and Load (ETL) Data with Amazon EMR	376
Distributed Copy Using S3DistCp	376
S3DistCp Options	377
Adding S3DistCp as a Step in a Cluster	381
S3DistCp Versions Supported in Amazon EMR	383
Export, Query, and Join Tables in DynamoDB	384
Prerequisites for Integrating Amazon EMR	385
Step 1: Create a Key Pair	386
Create a Cluster	386
Step 3: SSH into the Master Node	390
Set Up a Hive Table to Run Hive Commands	392
Hive Command Examples for Exporting, Importing, and Querying Data	396
Optimizing Performance	403
Store Avro Data in Amazon S3 Using Amazon EMR	406
Manage Clusters	408
View and Monitor a Cluster	408
View Cluster Details	409
View Log Files	413
View Cluster Instances in Amazon EC2	417
Monitor Metrics with CloudWatch	418
Logging Amazon Elastic MapReduce API Calls in AWS CloudTrail	431
Monitor Performance with Ganglia	433
Connect to the Cluster	441
Connect to the Master Node Using SSH	441
View Web Interfaces Hosted on Amazon EMR Clusters	446
Control Cluster Termination	458
Terminate a Cluster	459
Managing Cluster Termination	462
Resize a Running Cluster	465
Resize a Cluster Using the Console	466
Resize a Cluster Using the AWS CLI	467
Arrested State	468
Legacy Clusters	471
Cloning a Cluster Using the Console	472
Submit Work to a Cluster	473
Add Steps Using the CLI and Console	474
Submit Hadoop Jobs Interactively	475
Add More than 256 Steps to a Cluster	477
Automate Recurring Clusters with AWS Data Pipeline	477
Troubleshoot a Cluster	479
What Tools are Available for Troubleshooting?	479
Tools to Display Cluster Details	480
Tools to View Log Files	480
Tools to Monitor Cluster Performance	480
Known Issues with Amazon EMR AMIs	481
General Issues	481
Known Issues with Hadoop 2.4.0 AMIs	482
Known Issues with Hadoop 2.2.0 AMIs	483
Issues with Hadoop 1.0.3 AMIs	485

Troubleshoot a Failed Cluster	487
Step 1: Gather Data About the Issue	488
Step 2: Check the Environment	488
Step 3: Look at the Last State Change	489
Step 4: Examine the Log Files	490
Step 5: Test the Cluster Step by Step	491
Troubleshoot a Slow Cluster	491
Step 1: Gather Data About the Issue	492
Step 2: Check the Environment	492
Step 3: Examine the Log Files	493
Step 4: Check Cluster and Instance Health	495
Step 5: Check for Arrested Groups	496
Step 6: Review Configuration Settings	496
Step 7: Examine Input Data	498
Common Errors in Amazon EMR	498
Input and Output Errors	498
Permissions Errors	500
Memory Errors	501
Resource Errors	502
Streaming Cluster Errors	506
Custom JAR Cluster Errors	507
Hive Cluster Errors	507
VPC Errors	509
AWS GovCloud (US) Errors	511
Other Issues	511
Write Applications that Launch and Manage Clusters	512
End-to-End Amazon EMR Java Source Code Sample	512
Common Concepts for API Calls	516
Endpoints for Amazon EMR	516
Specifying Cluster Parameters in Amazon EMR	516
Availability Zones in Amazon EMR	517
How to Use Additional Files and Libraries in Amazon EMR Clusters	517
Amazon EMR Sample Applications	517
Use SDKs to Call Amazon EMR APIs	518
Using the AWS SDK for Java to Create an Amazon EMR Cluster	518
Using the AWS SDK for .Net to Create an Amazon EMR Cluster	519
Using the Java SDK to Sign an API Request	521
Hadoop Configuration Reference	522
JSON Configuration Files	522
Node Settings	523
Cluster Configuration	524
Configuration of <code>hadoop-user-env.sh</code>	526
Hadoop 2.2.0 and 2.4.0 Default Configuration	527
Hadoop Configuration (Hadoop 2.2.0, 2.4.0)	527
HDFS Configuration (Hadoop 2.2.0)	538
Task Configuration (Hadoop 2.2.0)	538
Intermediate Compression (Hadoop 2.2.0)	551
Hadoop 1.0.3 Default Configuration	553
Hadoop Configuration (Hadoop 1.0.3)	553
HDFS Configuration (Hadoop 1.0.3)	565
Task Configuration (Hadoop 1.0.3)	565
Intermediate Compression (Hadoop 1.0.3)	569
Hadoop 20.205 Default Configuration (Deprecated)	569
Hadoop Configuration (Hadoop 20.205)	569
HDFS Configuration (Hadoop 20.205)	573
Task Configuration (Hadoop 20.205)	573
Intermediate Compression (Hadoop 20.205)	576
Command Line Interface Reference for Amazon EMR	577

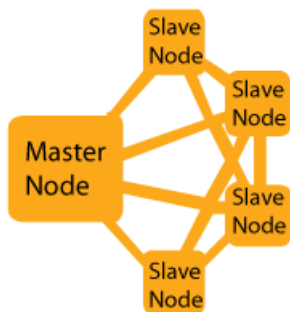
Specifying Parameter Values in AWS CLI for Amazon EMR	577
Setting Parameters with the Command Line	578
Displaying Parameter Values with the Command Line	578
Setting Parameters with the Configuration File	578
Install the Amazon EMR Command Line Interface (Deprecated)	579
Installing Ruby	579
Verifying the RubyGems package management framework	580
Installing the Amazon EMR Command Line Interface	580
Configuring Credentials	581
SSH Credentials	583
How to Call the Command Line Interface (Deprecated)	584
AWS EMR Command Line Interface Options (Deprecated)	585
Common Options	586
Uncommon Options	588
Options Common to All Step Types	588
Adding and Modifying Instance Groups	588
Adding JAR Steps to Job Flows	590
Adding JSON Steps to Job Flows	592
Adding Streaming Steps to Job Flows	592
Assigning an Elastic IP Address to the Master Node	595
Connecting to the Master Node	597
Creating Job Flows	598
Using HBase Options	605
Using Hive Options	615
Using Impala Options	619
Listing and Describing Job Flows	621
Passing Arguments to Steps	623
Using Pig Options	625
Specifying Step Actions	628
Specifying Bootstrap Actions	629
Tagging	635
Terminating Job Flows	637
Using S3DistCp	640
AWS EMR Command Line Interface Releases (Deprecated)	644
Document History	646

What is Amazon EMR?

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

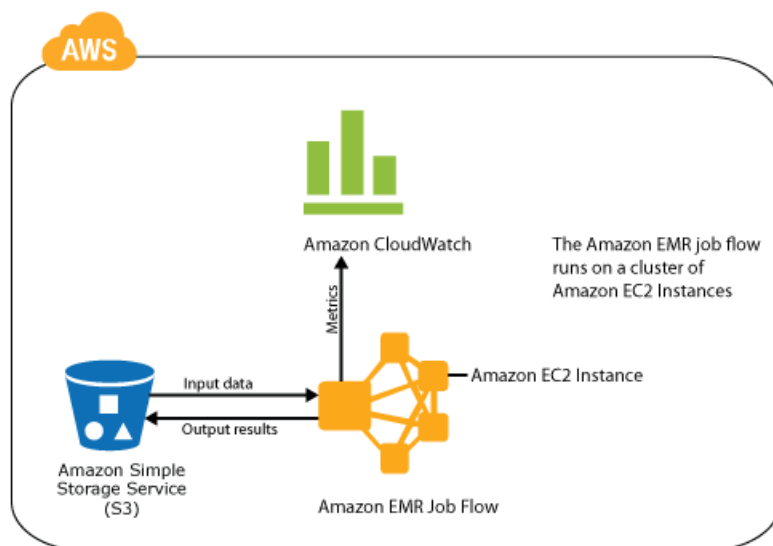
With Amazon Elastic MapReduce (Amazon EMR) you can analyze and process vast amounts of data. It does this by distributing the computational work across a cluster of virtual servers running in the Amazon cloud. The cluster is managed using an open-source framework called [Hadoop](#).

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.

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 \(p. 2\)](#)
- [Data Analysis and Processing on Amazon EMR \(p. 3\)](#)
- [Data Storage on Amazon EMR \(p. 3\)](#)
- [Move Data with Amazon EMR \(p. 3\)](#)

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 Elastic MapReduce (Amazon EMR) to run Hadoop on Amazon Web Services offers many advantages.

Topics

- [Resizable Clusters \(p. 4\)](#)
- [Pay Only for What You Use \(p. 4\)](#)
- [Easy to Use \(p. 4\)](#)
- [Use Amazon S3 or HDFS \(p. 4\)](#)
- [Parallel Clusters \(p. 4\)](#)
- [Hadoop Application Support \(p. 4\)](#)

- [Save Money with Spot Instances](#) (p. 4)
- [AWS Integration](#) (p. 5)
- [Instance Options](#) (p. 5)
- [MapR Support](#) (p. 5)
- [Business Intelligence Tools](#) (p. 5)
- [User Control](#) (p. 5)
- [Management Tools](#) (p. 5)
- [Security](#) (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](#) (p. 240), [Apache Pig](#) (p. 302), and [Apache HBase](#) (p. 310).

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 [\(Optional\) Lower Costs with Spot Instances \(p. 37\)](#).

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 \(p. 418\)](#) and [Export, Import, Query, and Join Tables in DynamoDB Using Amazon EMR \(p. 384\)](#).

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 [Choose the Number and Type of Instances \(p. 32\)](#).

MapR Support

Amazon EMR supports several MapR distributions. For more information, see [Using the MapR Distribution for Hadoop \(p. 218\)](#).

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 \(p. 216\)](#).

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 \(p. 124\)](#).

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? \(p. 10\)](#).

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 [Configure Access to the Cluster](#) (p. 180).

How Does Amazon EMR Work?

Amazon Elastic MapReduce (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. 3).

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

Topics

- [Hadoop](#) (p. 6)
- [Nodes](#) (p. 7)
- [Steps](#) (p. 8)
- [Cluster](#) (p. 9)

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. 6)
- [HDFS](#) (p. 7)
- [Jobs and Tasks](#) (p. 7)
- [Hadoop Applications](#) (p. 7)

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.

For more information, see <http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide.html>.

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 \(p. 9\)](#).

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 \(p. 240\)](#), [Apache Pig \(p. 302\)](#), and [Apache HBase \(p. 310\)](#).

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 [Instance Groups \(p. 34\)](#). For details on mapping legacy clusters to instance groups, see [Mapping Legacy Clusters to Instance Groups \(p. 471\)](#).

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 Elastic MapReduce (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 \(p. 473\)](#).

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.

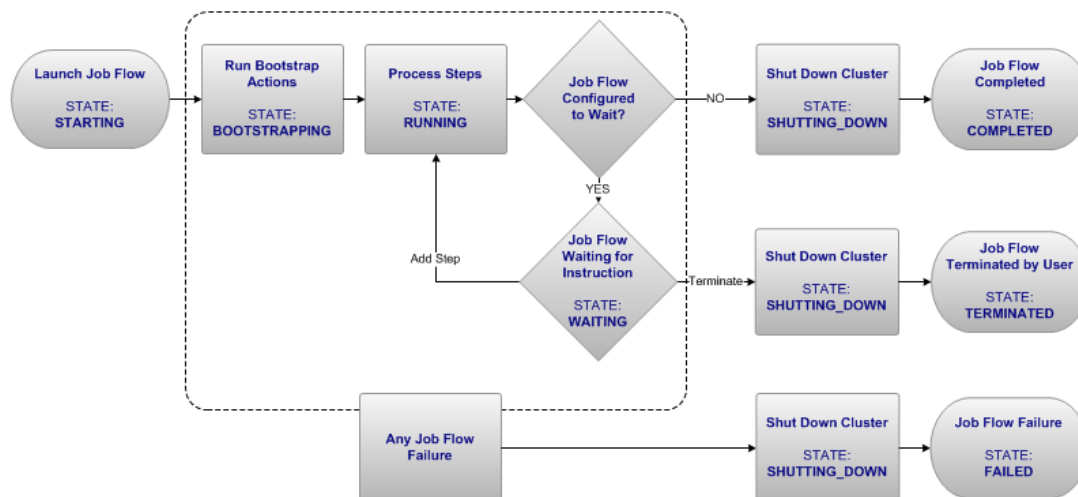
How to Send Work to a Cluster

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 \(p. 228\)](#).
- 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 \(p. 473\)](#).
- 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 \(p. 240\)](#), [Apache Pig \(p. 302\)](#), and [Apache HBase \(p. 310\)](#).
- 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 Elastic MapReduce (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`.

Note

Once the cluster reaches this phase, you are being billed for the EC2 instances provisioned.

After all bootstrap actions are completed, the cluster state is `RUNNING`. The job flow sequentially runs all cluster steps during this phase.

If you configured your cluster as a long-running cluster by enabling keep alive, the cluster will go into a `WAITING` state after processing is done and wait for the next set of instructions. For more information, see [How to Send Work to a Cluster \(p. 9\)](#) and [Choose the Cluster Lifecycle: Long-Running or Transient \(p. 163\)](#). You will have to manually terminate the cluster when you no longer require it.

If you configured your cluster as a transient cluster, it will automatically shut down after all of the steps complete.

When a cluster terminates without encountering an error, the state transitions to `SHUTTING_DOWN` and the cluster shuts down, terminating the virtual server instances. All data stored on the cluster is deleted. Information stored elsewhere, such as in your Amazon S3 bucket, persists. Finally, when all cluster activity is complete, the cluster state is marked as `COMPLETED`.

Unless termination protection is enabled, any failure during the cluster process terminates the cluster and all its virtual server instances. Any data stored on the cluster is deleted. The cluster state is marked as `FAILED`. For more information, see [Managing Cluster Termination \(p. 462\)](#).

For a complete list of cluster states, see the [JobFlowExecutionStatusDetail](#) data type in the *Amazon Elastic MapReduce (Amazon EMR) API Reference*.

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/>.

Amazon Elastic MapReduce Developer Guide

What Tools are Available for Amazon EMR?

- **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>.
- **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 \(p. 577\)](#).
- **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/>. 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 a custom SDK that calls Amazon EMR. For more information, see the [Amazon EMR API Reference](#)

The following table compares the functionality of the Amazon EMR interfaces.

Function	Console	AWS CLI	API, SDK, and Libraries
Create multiple clusters	✓	✓	✓
Define bootstrap actions in a cluster	✓	✓	✓
View logs for Hadoop jobs, tasks, and task attempts using a graphical interface	✓		
Implement Hadoop data processing programmatically			✓
Monitor clusters in real time	✓		
Provide verbose cluster details		✓	✓
Resize running clusters	✓	✓	✓
Run clusters with multiple steps	✓	✓	✓
Select version of Hadoop, Hive, and Pig	✓		✓
Specify the MapReduce executable in multiple computer languages	✓	✓	✓
Specify the number and type of EC2 instances that process the data	✓	✓	✓

Amazon Elastic MapReduce Developer Guide
Learn More About Hadoop and AWS Services Used with
Amazon EMR:

Function	Console	AWS CLI	API, SDK, and Libraries
Transfer data to and from Amazon S3 automatically	✓	✓	✓
Terminate clusters	✓	✓	✓

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

- Hadoop. For more information, go to <http://hadoop.apache.org/core/>.
- 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](#), the [Amazon Simple Storage Service Developer Guide](#), [Amazon SimpleDB Developer Guide](#) and the [Amazon CloudWatch Developer Guide](#), respectively.

Tutorial: Get Started with Amazon EMR

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

Amazon Elastic MapReduce (Amazon EMR) is a web service that makes it easy to process vast amounts of data quickly and cost-effectively. Amazon EMR uses Hadoop, an open source framework, to distribute your data and processing across a resizable cluster of Amazon EC2 instances. It can also run other distributed frameworks such as Spark and Presto. Amazon EMR is used in a variety of applications, including log analysis, web indexing, data warehousing, machine learning, financial analysis, scientific simulation, and bioinformatics.

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 Amazon Elastic Compute Cloud instances as virtual Linux servers for the master and slave nodes, Amazon Simple Storage Service for bulk storage of input and output data, and Amazon CloudWatch to monitor cluster performance and raise alarms. You can also move data into and out of Amazon DynamoDB using Amazon EMR and Hive. All of this is orchestrated by Amazon EMR control software that launches and manages the Hadoop cluster.

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 (a SQL-like scripting language for data warehousing and analysis), Pig (a scripting language for data analysis and transformation), HBase (a columnar, NoSQL data store), DistCp (a tool for copying large data sets), Ganglia (a monitoring framework), Impala (a distributed SQL-like query language), and Hue (a web interface for analyzing data), are already integrated with Amazon EMR. By running Hadoop on Amazon EMR you get the benefits of the cloud: the ability to inexpensively 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.

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 is performing 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, you launch a cluster and submit jobs interactively using the command line or you submit units of work called steps. From there you might interactively query the data, use the cluster as a data warehouse, or do periodic processing on a large data set. In this model, the cluster persists even when there are no steps or jobs queued for processing.

Tutorial Overview

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? \(p. 10\)](#).

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.

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.

This tutorial consists of the following.

Topics

- [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 \(p. 24\)](#)
- [\(Optional\) Step 6: Explore Amazon EMR \(p. 26\)](#)
- [\(Optional\) Step 7: Remove the Resources Used in the Tutorial \(p. 27\)](#)
- [Next Steps: Learn more about Amazon EMR \(p. 28\)](#)

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

1. Open <http://aws.amazon.com/> and click **Sign Up**.
2. Follow the on-screen 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. Click **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.
4. Click **Create**.
5. In the list, click your bucket name and click **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) Click **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

In this tutorial, you launch the 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.

Field	Action
Cluster name	When you create a cluster, the default name is "My cluster." You can also type a descriptive name for your cluster. The name is optional, and does not need to be unique.
Termination protection	<p>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.</p> <p>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.</p> <p>For more information, see Managing Cluster Termination (p. 462) .</p>
Logging	<p>By default, clusters created using the console have logging enabled. This option determines whether Amazon EMR writes detailed log data to Amazon S3.</p> <p>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.</p> <p>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.</p> <p>For more information, see View Log Files (p. 413) .</p>
Log folder S3 location	You can type or browse to your Amazon S3 bucket to store the Amazon EMR logs; for example, <code>s3://myemrbucket/logs</code> , 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.

Amazon Elastic MapReduce Developer Guide
Step 3: Launch an Amazon EMR Cluster

Field	Action
Debugging	<p>By default, when logging is enabled, debugging is also enabled. This option creates a debug log index in SimpleDB (additional charges apply) to enable detailed debugging in the Amazon EMR console. You can only enable debugging when the cluster is created.</p> <p>For more information about Amazon SimpleDB, go to the Amazon SimpleDB product description page.</p>

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	<p>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.</p> <p>For more information on MapR, see Using the MapR Distribution for Hadoop (p. 218).</p>
AMI version	<p>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.</p> <p>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.</p> <p>For more information on choosing an AMI, see Choose an Amazon Machine Image (AMI) (p. 48).</p>
Applications to be installed	<p>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 you select. You can remove pre-selected applications by choosing the Remove icon.</p>
Additional applications	<p>This option allows you to install additional applications such as Ganglia, Impala, HBase, and Hunk. When you choose an AMI, applications not available on the AMI do not appear in the list.</p>

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.

Amazon Elastic MapReduce Developer Guide
Step 3: Launch an Amazon EMR Cluster

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 must specify the EMRFS metadata store, the number of retries, and the retry period. For more information on EMRFS, see EMR File System (EMRFS) (Optional) (p. 141).

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](#).

Field	Action
Network	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 Your Default VPC and Subnets .
EC2 Subnet	No preference 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 on choosing a VPC subnet, see Select an Amazon VPC Subnet for the Cluster (Optional) (p. 205).

Amazon Elastic MapReduce Developer Guide
Step 3: Launch an Amazon EMR Cluster

Field	Action
Master	<p>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.</p> <p>For more information on Amazon EMR instance groups, see Instance Groups (p. 34).</p> <ul style="list-style-type: none">• EC2 instance type 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. <p>For more information on instance types supported by Amazon EMR, see Virtual Server Configurations.</p> <p>The default master instance type for Hadoop 2.x is m3.xlarge. This instance type is suitable for testing, development, and light workloads.</p> <ul style="list-style-type: none">• By default, Count is set to 1 for the master node. Currently, there is only one master node per cluster.• Request spot is unchecked by default. This option specifies whether to run master nodes on Spot Instances. <p>For more information on using Spot Instances, see (Optional) Lower Costs with Spot Instances (p. 37).</p>
Core	<p>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.</p> <p>For more information on Amazon EMR instance groups, see Instance Groups (p. 34).</p> <ul style="list-style-type: none">• 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. <p>For more information on instance types supported by Amazon EMR, see Virtual Server Configurations.</p> <p>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.</p> <ul style="list-style-type: none">• 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. <p>For more information on using Spot Instances, see (Optional) Lower Costs with Spot Instances (p. 37).</p>

Field	Action
Task	<p>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.</p> <p>For more information on Amazon EMR instance groups, see Instance Groups (p. 34).</p> <ul style="list-style-type: none"> • 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. <p>For more information on instance types supported by Amazon EMR, see Virtual Server Configurations.</p> <p>The default task instance type for Hadoop 2.x is m1.medium. This instance type is suitable for testing, development, and light workloads.</p> <ul style="list-style-type: none"> • 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. <p>For more information on using Spot Instances, see (Optional) Lower Costs with Spot Instances (p. 37).</p>

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.

Field	Action
EC2 key pair	<p>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.</p> <p>For more information on using SSH to connect to the master node, see Connect to the Master Node Using SSH (p. 441).</p>
IAM user access	<p>All other IAM users is selected by default. This option makes the cluster visible and accessible to all IAM users on the AWS account.</p> <p>If you choose No other IAM users, access to the cluster is restricted to the current IAM user.</p> <p>For more information on configuring cluster access, see Configure IAM User Permissions (p. 181).</p>

Field	Action
IAM roles	<p>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.</p> <p>If you select Custom you can specify your own EMR role and EC2 instance profile.</p> <p>For more information on using IAM roles with Amazon EMR, see Configure IAM Roles for Amazon EMR (p. 185).</p>

- 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 on using bootstrap actions, see [\(Optional\) Create Bootstrap Actions to Install Additional Software \(p. 124\)](#).

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

Field	Action
Add step	<p>By default, no user-defined steps are configured.</p> <p>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.</p>
Auto-terminate	<p>By default, auto-terminate is set to No. This keeps the cluster running until you terminate it.</p> <p>When set to Yes, the cluster is automatically terminated after the last step is completed.</p> <p>For more information on submitting work to a cluster, see Submit Work to a Cluster (p. 473).</p>

- Choose **Create cluster**.
- After clicking **Create cluster**, the **Cluster Details** page opens.
- 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.


```
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/>.
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
`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. 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. The step runs in approximately 1 minute.

View the Results

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

To view the output of the Hive script

1. Open the Amazon S3 console at <https://console.aws.amazon.com/s3/>.
2. In the Amazon S3 console, click the bucket that you used for the output data; for example,
`s3://myemrbucket/`.

3. Click the `output` folder.
4. The query writes results into a separate folder. Click `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

In order 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)

For more information on creating an SSH tunnel, see [Connect to the Master Node Using SSH \(p. 441\)](#).

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

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

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. In the console, on the **Cluster List** page, click the link for your cluster.
3. Note the **Master public DNS** value that appears at the top of the **Cluster Details** page. 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, click the link for your cluster.
3. Note the **Master public DNS** value that appears at the top of the **Cluster Details** page. This value is required to establish your SSH connection.
4. Double-click `putty.exe` to start PuTTY. You can also launch PuTTY from the Windows programs list.
5. If necessary, in the **Category** list, click **Session**.
6. In the **Host Name (or IP address)** field, type `hadoop@MasterPublicDNS`. For example:
`hadoop@ec2-###-##-###.compute-1.amazonaws.com`.
7. In the **Category** list, expand **Connection > SSH**, and then click **Auth**.
8. For **Private key file for authentication**, click **Browse** and select the .ppk file that you generated.
9. In the **Category** list, expand **Connection > SSH**, and then click **Tunnels**.
10. In the **Source port** field, type an unused local *port number*, for example 8157.
11. In the **Destination** field, 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. Click **Add**. You should see an entry in the **Forwarded ports** field similar to: L8157
`ec2-###-##-###.compute-1.amazonaws.com:8888`.
14. Click **Open** and then click **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, click **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. Click **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 Elastic MapReduce 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](#)

The following Amazon EMR tutorials are available to you in the *Amazon Elastic MapReduce 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](#)

The following Amazon EMR tutorials are available to you in *Getting Started with AWS: Analyzing Big Data*. These tutorials may require you to launch another cluster and incur additional costs.

- Conduct a sentiment analysis: [Tutorial: Sentiment Analysis](#)
- Analyze web server logs using Apache Hive: [Tutorial: Web Server Log Analysis](#)

Amazon Elastic MapReduce Developer Guide (Optional) Step 7: Remove the Resources Used in the Tutorial

The following Amazon EMR tutorials are available to you via the AWS Big Data Blog and AWS Official Blog. These tutorials may require you to launch another cluster and incur additional costs.

- Using Spark: [New — Apache Spark on Amazon EMR](#)
- Machine Learning and Spark: [Large-Scale Machine Learning with Spark on Amazon EMR](#)
- Using Node.js: [Node.js Streaming MapReduce with Amazon EMR](#)
- Building a recommendation engine: [Building and Running a Recommendation Engine at Any Scale](#)
- Using HBase: [Getting HBase Running on Amazon EMR and Connecting it to Amazon Kinesis](#)
- ETL processing: [ETL Processing Using AWS Data Pipeline and Amazon Elastic MapReduce](#)
- Using Spark: [Installing Apache Spark on an Amazon EMR Cluster](#)
- Using EMRFS: [Ensuring Consistency When Using Amazon S3 and Amazon Elastic MapReduce for ETL Workflows](#)
- Using R and RStudio: [Statistical Analysis with Open-Source R and RStudio on Amazon EMR](#)
- Using BI tools: [Using Amazon EMR with SQL Workbench and other BI Tools](#)
- Visualizing data: [Using Amazon EMR and Tableau to Analyze and Visualize Data](#)
- Using bootstrap actions: [Getting Started with Amazon EMR Bootstrap Actions](#)
- Building a recommender using Mahout: [Building a Recommender with Apache Mahout on Amazon Elastic MapReduce \(Amazon EMR\)](#)

The following Amazon EMR tutorials are available to you via the Articles & Tutorials page. These tutorials may require you to launch another cluster and incur additional costs.

- Using DynamoDB: [Using DynamoDB with Amazon Elastic MapReduce](#)
- Using Apache Accumulo: [Apache Accumulo and Amazon Elastic MapReduce](#)
- Using Informatica's HParser: [Parse Big Data with Informatica's HParser on Amazon EMR](#)
- Using Apache Hive for advertising: [Contextual Advertising using Apache Hive and Amazon EMR](#)
- Finding trending topics using Apache Hive: [Finding trending topics using Google Books n-grams data and Apache Hive on Amazon Elastic MapReduce](#)
- Finding similar items using Streaming: [Finding Similar Items with Amazon Elastic MapReduce, Python, and Hadoop Streaming](#)
- Operating a data warehouse with Hive: [Operating a Data Warehouse with Hive, Amazon Elastic MapReduce and Amazon SimpleDB](#)
- Analyzing CloudFront logs: [LogAnalyzer for Amazon CloudFront](#)
- Analyzing log data using Apache Hive: [Analyze Log Data with Apache Hive, Windows PowerShell, and Amazon EMR](#)

(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. Click 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. Right-click and choose **Delete** or click **Actions > Delete**.
5. Click **OK** to confirm the deletion when prompted.
6. Click **All Buckets** in the breadcrumb trail at the top of the window.
7. Right-click the bucket, choose **Delete**, and then click **OK** to confirm the deletion when prompted.

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 clicking the check box, and then click **Terminate**.
3. By default, clusters created using the console are launched with termination protection enabled. In the **Terminate clusters** dialog, for **Termination protection**, click **Change**.
4. Click **Off** and then click the check mark to confirm the change.
5. Click **Terminate**.

Next Steps: Learn more about Amazon EMR

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

To learn more about Amazon EMR, consult the resources in the following table.

Resource	Description
AWS Big Data Blog	The AWS big data blog contains technical articles designed to help you collect, store, clean, process, and visualize big data.
Amazon Elastic MapReduce Developer Guide	This document. Provides conceptual information about Amazon EMR and describes how to use Amazon EMR features.
Amazon Elastic MapReduce API Reference	Contains a technical description of all Amazon EMR APIs.
Getting Started Analyzing Big Data with AWS	This tutorial explains how to use Amazon EMR and Apache Hive to analyze web server log files and query them for information without writing any code at all.
Amazon EMR Technical FAQ	Covers the top questions developers have asked about this product.

Resource	Description
Amazon EMR Release Notes	Gives a high-level overview of the current release, and notes any new features, corrections, and known issues.
Amazon EMR Articles and Tutorials	A list of articles, tutorials, and videos about Amazon EMR. Topics include tutorials that walk you through using Amazon EMR to solve a specific business problem and using third-party applications with Amazon EMR.
AWS Developer Resource Center	A central starting point to find documentation, code samples, release notes, and other information to help you build innovative applications with AWS.
Amazon EMR console	Enables you to perform most of the functions of Amazon EMR and other AWS products without programming.
Amazon EMR Forum	A community-based forum for developers to discuss technical questions related to Amazon EMR.
AWS Support Center	The home page for AWS Support, including access to our Developer Forums, Technical FAQs, Service Status page, and AWS Premium Support (if you are subscribed to this program).
Amazon EMR Product Information	The primary web page for information about Amazon EMR.
Form for questions related to your AWS account: Contact Us	This form is <i>only</i> for account questions. For technical questions, use the Discussion Forums.

Plan an Amazon EMR Cluster

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

This section explains configuration options for launching Amazon Elastic MapReduce (Amazon EMR) clusters. Before you launch a cluster, review this information and make choices about the cluster options based on your data processing needs. The options that you choose depend on factors such as the following:

- The type of source data that you want to process
- The amount of source data and how you want to store it
- The acceptable duration and frequency of processing source data
- The network configuration and access control requirements for cluster connectivity
- The metrics for monitoring cluster activities, performance, and health
- The software that you choose to install in your cluster to process and analyze data
- The cost to run clusters based on the options that you choose

Although some configuration steps are optional, we recommend that you review them to make sure that you understand the options available to you and plan your cluster accordingly.

Topics

- [Choose an AWS Region \(p. 31\)](#)
- [Choose the Number and Type of Instances \(p. 32\)](#)
- [Configure the Software \(p. 48\)](#)
- [File Systems Compatible with Amazon EMR \(p. 138\)](#)
- [Choose the Cluster Lifecycle: Long-Running or Transient \(p. 163\)](#)
- [Prepare Input Data \(Optional\) \(p. 164\)](#)
- [Prepare an Output Location \(Optional\) \(p. 175\)](#)
- [Configure Access to the Cluster \(p. 180\)](#)
- [Configure Logging and Debugging \(Optional\) \(p. 200\)](#)
- [Select an Amazon VPC Subnet for the Cluster \(Optional\) \(p. 205\)](#)
- [Tagging Amazon EMR Clusters \(p. 211\)](#)
- [Use Third Party Applications With Amazon EMR \(Optional\) \(p. 216\)](#)

Choose an AWS Region

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

Amazon Web Services run on servers in data centers around the world. These 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, go to [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 October 10, 2012, the default region when you access a resource from the AWS Management Console is US West (Oregon) (us-west-2); for older accounts, the default region is 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.

Amazon EMR Region	Amazon EMR CLI and API Region	Amazon S3 Region	Amazon EC2 Region
US East (N. Virginia)	us-east-1	US Standard	US East (N. Virginia)
US West (Oregon)	us-west-2	Oregon	US West (Oregon)
US West (N. California)	us-west-1	Northern California	US West (N. California)
EU (Ireland)	eu-west-1	Ireland	EU (Ireland)
EU (Frankfurt)	eu-central-1	Frankfurt	EU (Frankfurt)
Asia Pacific (Singapore)	ap-southeast-1	Singapore	Asia Pacific (Singapore)
Asia Pacific (Tokyo)	ap-northeast-1	Tokyo	Asia Pacific (Tokyo)
Asia Pacific (Sydney)	ap-southeast-2	Sidney	Asia Pacific (Sydney)
South America (Sao Paulo)	sa-east-1	Sao Paulo	South America (Sao Paulo)
AWS GovCloud (US)	us-gov-west-1	GovCloud	AWS GovCloud (US)
China (Beijing)	cn-north-1	China (Beijing)	China (Beijing)

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, go to the [AWS GovCloud \(US\)](#) product page.

Choose a Region using the Console

To choose a region using the console

- Choose the region list to the right of your account information on the navigation bar, to switch regions. Your default region is displayed automatically.

Specify a Region using the AWS CLI

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

Choose a Region Using an SDK or the API

To choose a region using an SDK

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

```
client.setEndpoint("eu-west-1.elasticmapreduce.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.

Choose the Number and Type of Instances

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

One of the most important considerations when launching a cluster is the number and type of instances to launch in the cluster. This will determine the processing power and storage capacity of the cluster. Tuning the cluster to the data you need to process is important. Too little capacity can cause your cluster to run slowly, too much capacity results in unnecessary cost.

The servers you run in an Amazon EMR cluster are EC2 instances. These come in different configurations such as `m1.large` and `m1.xlarge`, with different CPU, input/output, and storage capacity. For more information, see [Instance Configurations \(p. 35\)](#). You can specify a different instance type for each server role in the Amazon EMR cluster.

Amazon EMR allows you to launch a cluster to dedicated hardware you manage within a virtual private cloud (VPC). To launch a cluster using dedicated instances, mark instances launched in a VPC with the `dedicated` tenancy attribute or mark the entire VPC at creation time. For more information, see the [Amazon VPC User Guide](#).

There are up to three types of server roles in a Amazon EMR cluster: master, core, and task. These are referred to as nodes and run on EC2 instances. A single master node manages the cluster. Core nodes both process data and store data in HDFS. Task nodes are optional, and only process data. For more information, see [Instance Groups \(p. 34\)](#). When selecting the number and type of instances for each role, keep in mind the different capacity needs of each role. The master node, which assigns tasks, doesn't require much processing power. Core nodes, which process tasks and store data in HDFS need both processing power and storage. Task nodes, which don't store data, need only processing power.

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 \(p. 408\)](#). Another way is to calculate the capacity of the instances you are considering and compare that value against the size of your data.

Calculate the HDFS Capacity of a Cluster

The amount of HDFS storage available to your cluster depends on three factors: the number of core nodes, the storage capacity of the type of EC2 instance you specify for the core nodes, and the replication factor. The replication factor is the number of times each data block is stored in HDFS for raid-like redundancy. By default, the replication factor is 3 for a cluster of 10 or more nodes, 2 for a cluster 4-9 nodes, and 1 for a cluster with 3 or fewer nodes.

To calculate the HDFS capacity of a cluster, multiply the storage capacity of the EC2 instance type you've selected by the number of nodes and divide the total by the replication factor for the cluster. For example, a cluster with 10 core nodes of type `m1.large` would have 2800 GB of space available to HDFS: $(10 \text{ nodes} \times 840 \text{ GB per node}) / \text{replication factor of } 3$. 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 by adding more core nodes, choosing an EC2 instance type with greater storage capacity, using data compression, or 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.

Guidelines for the Number and Type of Instances

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 a `m1.small` for Hadoop 1 clusters and `m1.large` for Hadoop 2 clusters. For clusters of more than 50 nodes, consider using an `m1.large` for Hadoop 1 clusters and `m1.xlarge` for Hadoop 2 clusters.
- The computational needs of the core and task nodes depend on the type of processing your application will perform. Many jobs can be run on `m1.large` instance types, which offer balanced performance in terms of CPU, disk space, and input/output. If your application has external dependencies that introduce delays (such as web crawling to collect data), you may be able to run the cluster on `m1.small` instances to reduce costs while the instances are waiting for dependencies to finish. For improved performance, consider running the cluster using `m1.xlarge` instances for the core and task nodes. If different phases of your job flow 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 `m1.large` and `m1.xlarge`. Computation-intensive clusters may benefit from running on High CPU instances, which have proportionally more CPU than RAM memory. 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. For more information about these instance types, see [Instance Configurations](#) (p. 35).

- The amount of data you can process depends on the capacity of your core nodes and the size of your data as input, during processing, and as output. The input, intermediate, and output data sets all reside on the cluster during processing.
- By default, the total number of EC2 instances you can run on a single AWS account is 20. This means that the total number of nodes you can have in a cluster is 20. For more information about how to request that this limit be increased for your account, see [AWS Limits](#).
- In Amazon EMR, m1.small and m1.medium instances are recommended only for testing purposes and m1.small is not supported on Hadoop 2 clusters.

Topics

- [Instance Groups](#) (p. 34)
- [Instance Configurations](#) (p. 35)
- [Ensure Capacity with Reserved Instances \(Optional\)](#) (p. 37)
- [\(Optional\) Lower Costs with Spot Instances](#) (p. 37)

Instance Groups

Amazon EMR runs a managed version of Apache Hadoop, handling the details of creating the cloud-server infrastructure to run the Hadoop cluster. Amazon EMR defines the concept of instance groups, which are collections of EC2 instances that perform a set of roles defined by the distributed applications that are installed on your cluster. Generally, these groups are organized as *master* and *slave* groups. There are three types of instance groups: master, core (slave), and task (slave).

Each Amazon EMR cluster can include up to 50 instance groups: one master instance group that contains one master node, a core instance group containing one or more core nodes, and up to 48 optional task instance groups, which can contain any number of task nodes.

If the cluster is run on a single node, then that instance is simultaneously a master and a core node. For clusters running on more than one node, one instance is the master node and the remaining are core or task nodes.

For more information about instance groups, see [Resize a Running Cluster](#) (p. 465).

Master Instance Group

The master instance group manages the cluster and typically runs master components of the distributed applications that are installed on your cluster. For example, it runs the YARN ResourceManager service to manage resources for applications and the HDFS NameNode service. It also tracks the status of jobs submitted to the cluster, and monitors the health of the instance groups. To monitor the progress of the cluster, you can SSH into the master node as the Hadoop user and either look at the Hadoop log files directly or access the user interface that Hadoop or Spark publishes to a web server running on the master node. For more information, see [View Log Files](#) (p. 413).

During a Hadoop MapReduce or Spark job, components on core and task nodes process the data, transfer the output to Amazon S3 or HDFS, and provide status metadata back to the master node. In the case of a single node cluster, all components are run on the master node.

Caution

The instance controller, an Amazon EMR cluster management component, on the master node uses MySQL. If MySQL becomes unavailable, the instance controller will be unable to launch and manage instances.

Core Instance Group

The core instance group contains all of the core nodes of a cluster. A core node is an EC2 instance in the cluster that runs tasks and stores data as part of the Hadoop Distributed File System (HDFS) by running the DataNode daemon. For example, a core node runs YARN NodeManager daemons and runs Hadoop MapReduce tasks and Spark executors.

Core nodes are managed by the master node. Because core nodes store data in HDFS, you can't remove them from a cluster. However, you can add more core nodes to a running cluster to increase HDFS storage capacity.

Caution

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

For more information about core instance groups, see [Resize a Running Cluster \(p. 465\)](#).

Task Instance Group

Task instance groups contain the task nodes in your cluster. Task instance groups are optional. You can add task groups when you start the cluster, or you can add task groups to a running cluster. Task nodes do not run the DataNode daemon or store data in HDFS. You can add and remove task nodes to adjust the number of EC2 instances in your cluster, increasing capacity to handle peak loads and decreasing it later.

Task nodes are managed by the master node. While a cluster is running you can increase and decrease the number of task nodes in your task groups, and you can add up to 48 additional task groups. When you add task groups to your cluster, you can create groups that leverage multiple instance types. For example, if you create a task group that leverages Spot Instances, and there are not enough Spot Instances available for a particular instance type, you can create an additional task group that leverages a different instance type with available Spot Instance capacity.

Creating multiple task groups that leverage Spot Instances can provide you with potential cost savings. For more information, see [\(Optional\) Lower Costs with Spot Instances \(p. 37\)](#).

For more information about task instance groups, see [Resize a Running Cluster \(p. 465\)](#).

Instance Configurations

Amazon EMR enables you to choose the number and kind of EC2 instances that comprise the cluster that processes your cluster. Amazon EC2 offers several basic 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 supported with Amazon EMR.

Instance Class	Instance Types
General purpose	m1.small (Hadoop 1 only) m1.medium m1.large m1.xlarge m3.xlarge m3.2xlarge
Compute optimized	c1.medium c1.xlarge c3.xlarge c3.2xlarge c3.4xlarge c3.8xlarge cc2.8xlarge
Memory optimized	m2.xlarge m2.2xlarge m2.4xlarge r3.xlarge r3.2xlarge r3.4xlarge r3.8xlarge cr1.8xlarge
Storage optimized	hi1.4xlarge hs1.2xlarge hs1.4xlarge hs1.8xlarge i2.xlarge i2.2xlarge i2.4xlarge i2.8xlarge d2.xlarge d2.2xlarge d2.4xlarge d2.8xlarge
GPU instances	g2.2xlarge cg1.4xlarge

Note

Amazon EMR does not support micro instances at this time.

Note

I2, G2, CC2, and R3 instance types are only supported on AMI 2.4.5 or later (Hadoop 1) and AMI 3.0.4 or later (Hadoop 2).

The following table shows the virtualization type for each instance family used in Amazon EMR:

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

For more information on these instance types, families, and virtualization types, see [Amazon EC2 Instances](#) and [Amazon Linux AMI Instance Type Matrix](#).

Ensure Capacity with Reserved Instances (Optional)

Reserved Instances provide reserved capacity and are an additional Amazon EC2 pricing option. You make a one-time payment for an instance to reserve capacity and reduce hourly usage charges. Reserved Instances complement existing Amazon EC2 On-Demand Instances and provide an option to reduce computing costs. As with On-Demand Instances, you pay only for the compute capacity that you actually consume, and if you don't use an instance, you don't pay usage charges for it.

Reserved Instances can also provide a considerable cost savings. For more information, see [Amazon EC2 Reserved Instances](#).

To use a Reserved Instance with Amazon EMR, launch your cluster in the same Availability Zone as your Reserved Instance. For example, let's say you purchase one m1.small Reserved Instance in US-East. If you launch a cluster that uses two m1.small instances in the same Availability Zone in Region US-East, one instance is billed at the Reserved Instance rate and the other is billed at the On-Demand rate. If you have a sufficient number of available Reserved Instances for the total number of instances you want to launch, you are provisioned the appropriate capacity. Your Reserved Instances are used before any On-Demand Instances are created.

You can use Reserved Instances by using either the Amazon EMR console, the command line interface (CLI), Amazon EMR API actions, or the AWS SDKs.

Related Topics

- [Amazon EC2 Reserved Instances](#)

(Optional) Lower Costs with Spot Instances

When Amazon EC2 has unused capacity, it offers Amazon EC2 instances at a reduced cost, called the *Spot Price*. This price fluctuates based on availability and demand. You can purchase Spot Instances by placing a request that includes the highest bid price you are willing to pay for those instances. When the Spot Price is below your bid price, your Spot Instances are launched and you are billed the Spot Price. If the Spot Price rises above your bid price, Amazon EC2 terminates your Spot Instances.

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

If your workload is flexible in terms of time of completion or required capacity, Spot Instances can significantly reduce the cost of running your clusters. Workloads that are ideal for using Spot Instances include: application testing, time-insensitive workloads, and long-running clusters with fluctuations in load.

Note

We do not recommend Spot Instances for master and core nodes unless the cluster is expected to be short-lived and the workload is non-critical. Also, Spot Instances are not recommended for clusters that are time-critical or that need guaranteed capacity. These clusters should be launched using on-demand instances.

Topics

- [When Should You Use Spot Instances? \(p. 38\)](#)
- [Choose What to Launch as Spot Instances \(p. 38\)](#)
- [Spot Instance Pricing in Amazon EMR \(p. 39\)](#)
- [Availability Zones and Regions \(p. 40\)](#)
- [Launch Spot Instances in a Cluster \(p. 40\)](#)
- [Change the Number of Spot Instances in a Cluster \(p. 44\)](#)

- [Troubleshoot Spot Instances \(p. 47\)](#)

When Should You Use Spot Instances?

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

Long-Running Clusters and Data Warehouses

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

Cost-Driven Workloads

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

Data-Critical Workloads

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

Application Testing

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

Choose What to Launch as Spot Instances

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

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

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

Project	Master Instance Group	Core Instance Group	Task Instance Group(s)
Long-running clusters	on-demand	on-demand	spot
Cost-driven workloads	spot	spot	spot
Data-critical workloads	on-demand	on-demand	spot
Application testing	spot	spot	spot

Master Instance Group as Spot Instances

The master node controls and directs the cluster. When it terminates, the cluster ends, so you should only launch the master node as a Spot Instance if you are running a cluster where sudden termination is acceptable. This might be the case if you are testing a new application, have a cluster that periodically persists data to an external store such as Amazon S3, or are running a cluster where cost is more important than ensuring the cluster's completion.

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

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

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

Core Instance Group as Spot Instances

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

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

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

Task Instance Groups as Spot Instances

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

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

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

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

Spot Instance Pricing in Amazon EMR

There are two components in Amazon EMR billing, the cost for the EC2 instances launched by the cluster and the charge Amazon EMR adds for managing the cluster. When you use Spot Instances, the Spot Price may change due to fluctuations in supply and demand, but the Amazon EMR rate remains fixed.

When you purchase Spot Instances, you can set the bid price only when you launch the instance group. It can't be changed later. This is something to consider when setting the bid price for an instance group in a long-running cluster.

You can launch different instance groups at different bid prices. For example, in a cluster running entirely on Spot Instances, you might choose to set the bid price for the master instance group at a higher price than the task instance group since if the master terminates, the cluster ends, but terminated task instances can be replaced.

If you manually start and stop instances in the cluster, partial hours are billed as full hours. If instances are terminated by AWS because the Spot Price rose above your bid price, you are not charged either the Amazon EC2 or Amazon EMR charges for the partial hour.

You can see the real time Spot Price in the console when you select **Request Spot** and mouse over the information tooltip next to **Bid Price**. The prices for each Availability Zone in the selected region are displayed and the lowest price accepted will be the green-colored row(s). For further information about the lowest and the on-demand price for instances, see [Amazon EC2 Pricing page](#).

Availability Zones and Regions

When you launch a cluster, you have the option to specify a region, Availability Zone (based on EC2 Subnet), and a Spot Price, which is displayed for each Availability Zone in the **Bid Price** tooltip in the console. Amazon EMR launches all of the instance groups in the Availability Zone chosen by Amazon EMR based on acceptance of a bid price specified. You can also select the Availability Zone into which your cluster launches by selecting an **EC2 Subnet**. If you select a subnet, the Bid Price tooltip will indicate this by underlining the corresponding row of Availability Zone with Spot Price.

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, you should study the history of Availability Zone pricing before choosing the Availability Zone for your cluster.

Note

Because Amazon EMR selects the Availability Zone based on free capacity of Amazon EC2 instance type you specified for the core instance group, your cluster may end up in an Availability Zone with less capacity in other EC2 instance types. For example, if you are launching your core instance group as Large and the master instance group as Extra Large, you may launch into an Availability Zone with insufficient unused Extra Large capacity to fulfill a Spot Instance request for your master node. If you run into this situation, you can launch the master instance group as on-demand, even if you are launching the core instance group as Spot Instances.

All instance groups in a cluster are launched into a single Availability Zone, regardless of whether they are on-demand or as Spot Instances. The reason for using a single Availability Zone is additional data transfer costs and performance overhead make running instance groups in multiple Availability Zones undesirable.

Launch Spot Instances in a Cluster

When you launch a new instance group you can launch the Amazon EC2 instances in that group either as on-demand or as Spot Instances. The procedure for launching Spot Instances is the same as launching on-demand instances, except that you specify the bid price.

Launch a Cluster with Spot Instances Using the Amazon EMR Console

To launch an entire cluster with Spot Instances

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Choose **Create cluster**.
3. In the **Hardware Configuration** section, to run the master node as a Spot Instance, click **Request Spot** in the **Master** row and enter the maximum hourly rate you are willing to pay per instance in the

Bid Price field. You can see the Spot Price for all of the current region's Availability Zones in the console by mousing over the information tooltip next to the **Bid Price** field. In most cases, you will want to enter a price higher than the current Spot Price.

4. To run the core nodes as Spot Instances, click the **Request Spot** check box in the **Core** row and enter the maximum hourly rate you are willing to pay per instance in the **Spot Bid Price** field.
5. To run the task nodes as Spot Instances, click the **Request Spot** check box in the **Task** row and enter the maximum hourly rate you are willing to pay per instance in the **Spot Bid Price** field.

Note

You can also create additional task instance groups (up to 48) using Spot Instances by clicking **Add task instance group**.

6. Proceed to create the cluster as described in [Plan an Amazon EMR Cluster \(p. 30\)](#).

Launch a Cluster with Spot Instances Using the AWS CLI

You can launch an entire cluster on Spot Instances using the AWS CLI, or you can launch specific instance groups on Spot Instances. With the CLI and API, you can choose the Availability Zone. If you do not select the Availability Zone, Amazon EMR will select it for you.

To launch an entire cluster with Spot Instances using the AWS CLI

To specify that an instance group should be launched with Spot Instances in the AWS CLI, type the `create-cluster` subcommand and use the `--instance-groups` parameter with the `BidPrice` argument.

- To create a cluster where the master, core, and task instance groups all use Spot Instances, 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 "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,Instance
Count=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 --in
stance-groups InstanceGroupType=MASTER,InstanceType=m3.xlarge,Instance
Count=1,BidPrice=0.25 InstanceGroupType=CORE,BidPrice=0.03,Instance
Type=m3.xlarge,InstanceCount=2 InstanceGroupType=TASK,BidPrice=0.10,Instance
Type=m3.xlarge,InstanceCount=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/>.

Launch a Cluster with Spot Instances Using the Java SDK

To launch an entire cluster with Spot Instances using the Java SDK

- To specify that an instance group should be launched as Spot Instances, set the `withBidPrice` and `withMarket` properties on the `InstanceGroupConfig` object that you instantiate for the instance group. The following code shows how to define master, core, and task instance groups that run as Spot Instances.

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

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

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

Launch Task Groups on Spot Instances Using the Amazon EMR Console

To launch task instance groups on Spot Instances

- Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
- Choose **Create cluster**.
- In the **Hardware Configuration** section, to run only the task nodes as Spot Instances, click **Request Spot** in the **Task** row and enter the maximum hourly rate you are willing to pay per instance in the **Bid Price** field. You can see the Spot Price for all of the current region's Availability Zones in the console by mousing over the information tooltip next to the **Bid Price** field. In most cases, you will want to enter a price higher than the current Spot Price.
- To add additional task instance groups that use Spot Instances, click **Add task instance group** and follow the previous steps. You can add up to 48 task instance groups using the console.
- Proceed to create the cluster as described in [Plan an Amazon EMR Cluster \(p. 30\)](#).

Launch Task Groups on Spot Instances Using the AWS CLI

Rather than launching all instance groups using Spot Instances, a more typical use case is to launch one or more task groups using Spot Instances. The following sections demonstrate launching task groups with Spot Instances using the console, the AWS CLI, the SDK, and the API.

To launch task instance groups on Spot Instances using the AWS CLI

To launch task instance groups on Spot Instances using the AWS CLI, specify the `BidPrice` for task instance groups.

- To launch a single task instance group on Spot Instances, 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 "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,Instance
Count=1 \
InstanceGroupType=CORE,InstanceType=m3.xlarge,InstanceCount=2 \
InstanceGroupType=TASK,BidPrice=0.03,InstanceType=m3.xlarge,InstanceCount=4
```

- 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 --in
stance-groups InstanceGroupType=MASTER,InstanceType=m3.xlarge,InstanceCount=1
  InstanceGroupType=CORE,InstanceType=m3.xlarge,InstanceCount=2 InstanceGroup
Type=TASK,BidPrice=0.03,InstanceType=m3.xlarge,InstanceCount=4
```

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.

- To launch multiple task instance groups on Spot Instances, type the following command and replace `myKey` with the name of your EC2 key pair. You can add up to 5 task instance groups in a single command.
- Linux, UNIX, and Mac OS X 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,Instance
Count=1 \
InstanceGroupType=CORE,InstanceType=m3.xlarge,InstanceCount=2 \
InstanceGroupType=TASK,BidPrice=0.03,InstanceType=m3.xlarge,InstanceCount=4
\
InstanceGroupType=TASK,BidPrice=0.04,InstanceType=m3.xlarge,InstanceCount=2
```

- 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 --in
stance-groups InstanceGroupType=MASTER,InstanceType=m3.xlarge,InstanceCount=1
InstanceGroupType=CORE,InstanceType=m3.xlarge,InstanceCount=2 InstanceGroup
Type=TASK,BidPrice=0.03,InstanceType=m3.xlarge,InstanceCount=4 InstanceGroup
Type=TASK,BidPrice=0.04,InstanceType=m3.xlarge,InstanceCount=2
```

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

Launch the Task Group on Spot Instances Using the Java SDK

To launch the task instance group on Spot Instances using the Java SDK

- To specify that an instance group should be launched as Spot Instances, set the *withBidPrice* and *withMarket* properties on the *InstanceGroupConfig* object that you instantiate for the instance group. The following code creates a task instance group of type *m1.large* with an instance count of 10. It specifies \$0.35 as the maximum bid price, which launches the task group on Spot Instances.

```
InstanceGroupConfig instanceGroupConfigMaster = new InstanceGroupConfig()
    .withInstanceCount(1)
    .withInstanceRole("MASTER")
    .withInstanceType("m3.xlarge")

InstanceGroupConfig instanceGroupConfigCore = new InstanceGroupConfig()
    .withInstanceCount(4)
    .withInstanceRole("CORE")
    .withInstanceType("m3.xlarge")

InstanceGroupConfig instanceGroupConfig = new InstanceGroupConfig()
    .withInstanceCount(10)
    .withInstanceRole("TASK")
    .withInstanceType("m3.xlarge")
    .withMarket("SPOT")
    .withBidPrice("0.35");
```

Change the Number of Spot Instances in a Cluster

With some restrictions, you can modify the number of Spot Instances in a cluster. For example, because the master instance group contains only one instance, you cannot modify the master group's instance count whether or not the instance group is using a Spot Instance.

If you are running a cluster that contains only a master node, you cannot add instance groups or instances to that cluster. A cluster must have one or more core instances for you to be able to add or modify instance groups.

You can only define an instance group using Spot Instances when it is created. For example, if you launch the core instance group using on-demand instances, you cannot change them to Spot Instances later.

The following examples show how to change the number of Spot Instances in an instance group using the console, CLI, SDK, and API.

Change the Number of Spot Instances Using the Amazon EMR Console

If you launch a cluster using Spot Instances for one or more instance groups, you can change the number of requested Spot Instances for those groups using the Amazon EMR console. Note that you can only increase the number of core instances in your cluster while you can increase or decrease the number of task instances. Setting the number of task instances to zero removes all Spot Instances but does not remove the instance group.

To change the number of Spot Instances in an instance group using the console

1. In the [Amazon EMR console](#), on the **Cluster List** page, click the link for your cluster.
2. Click **Resize**.
3. In the **Resize Cluster** dialog, in the **Count** column, click the **Resize** link for the instance groups to which you wish to add Spot Instances. You can also click **Add task instance group** to add additional task groups that use Spot Instances (up to 48).
4. To add core or task instances, type a higher number in the **Count** column and click the check mark to confirm the change. To reduce the number of task instances, type a lower number in the **Count** column and click the check mark to confirm the change. After changing the instance count, click **Close**.

Note

You cannot change the bid price when adding or removing Spot Instances from instance groups using the console. Also note, changing the instance count of the task group to 0 removes all Spot Instances but not the instance group.

Change the Number of Spot Instances Using the AWS CLI

Using the AWS CLI, you can: increase the number of Spot Instances in the core instance group, increase or decrease the number of Spot Instances in the task instance group, add one or more task instance groups containing Spot Instances, or terminate a Spot Instance in the core instance group.

To change the number of Spot Instances for instance groups using the AWS CLI

You can add Spot Instances to the core or task groups, and you can remove instances from task groups using the AWS CLI `modify-instance-groups` subcommand with the `InstanceCount` parameter. To add instances to the core or task groups, increase the `InstanceCount`. To reduce the number of instances in task groups, decrease the `InstanceCount`. Changing the instance count of a task group to 0 removes all Spot Instances but not the instance group.

1. Retrieve the cluster ID for the cluster you wish to alter using the console or by typing:

```
aws emr list-clusters
```

2. To change the instance count for an instance group, you need the `InstanceGroupId`. To retrieve the `InstanceGroupId`, use the `describe-cluster` subcommand with the cluster ID:

```
aws emr describe-cluster --cluster-id string
```

The output is a JSON object called `Cluster` that contains the ID of each instance group.

You can also pipe the JSON object to a program like `grep` or `jq`. To retrieve a cluster's instance group name and ID using `jq` on Unix-based systems, type the following:

```
aws emr describe-cluster --cluster-id string | jq ".Cluster.InstanceGroups[]  
| {name,Id}"
```

3. To increase the number of Spot Instances in a task instance group from 3 to 4, type:

```
aws emr modify-instance-groups --instance-groups InstanceGroupId=ig-  
31JXXXXXXBTO, InstanceCount=4
```

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

To add one or more task instance groups with Spot Instances to a cluster using the AWS CLI

Using the AWS CLI, you can add one or more task instance groups of Spot Instances to a cluster with the `add-instance-groups` subcommand. You can add up to 5 task instance groups in a single command.

1. To add a single task instance group of Spot Instances to a cluster, type:

```
aws emr add-instance-groups --cluster-id j-JXBXXXXXX37R --instance-groups  
InstanceCount=6,BidPrice=.25,InstanceGroupType=task,InstanceType=m3.xlarge
```

2. To add multiple task instance groups of Spot Instances to a cluster, type:

```
aws emr add-instance-groups --cluster-id j-JXBXXXXXX37R --instance-groups  
InstanceCount=6,BidPrice=.25,InstanceGroupType=task,InstanceType=m3.xlarge  
InstanceCount=4,BidPrice=.50,InstanceGroupType=task,InstanceType=m3.xlarge
```

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

To terminate a Spot Instance in the core instance group using the AWS CLI

Using the AWS CLI, you can terminate a Spot Instance in the core instance group with the `modify-instance-groups` subcommand. This should be done with caution. Terminating an instance in the core instance group risks data loss, and the instance is not automatically replaced. To terminate a specific instance you need the instance group ID (returned by the `aws emr describe-cluster --cluster-id` command) and the instance ID (returned by the `aws emr list-instances --cluster-id` command).

- Type the following command to terminate an instance in the core instance group.

```
aws emr modify-instance-groups --instance-groups InstanceGroupId=ig-  
6RXXXXXX07SA, EC2InstanceIdsToTerminate=i-f9XXXXf2
```

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

Add Spot Instances to Instance Groups Using the Java SDK

To add Spot Instances to a cluster using the Java SDK

- To specify that an instance group should be launched as Spot Instances, set the *withBidPrice* and *withMarket* properties on the *InstanceGroupConfig* object that you instantiate for the instance group. The following code creates a task instance group of type *m1.large* with an instance count of 10. It specifies \$0.35 as the maximum bid price, and runs as Spot Instances. When you make the call to modify the instance group, pass this object instance:

```
InstanceGroupConfig instanceGroupConfig = new InstanceGroupConfig()  
    .withInstanceCount(10)  
    .withInstanceRole("TASK")  
    .withInstanceType("m3.xlarge")  
    .withMarket("SPOT")  
    .withBidPrice("0.35");
```

Troubleshoot Spot Instances

The following topics address issues that may arise when you use Spot Instances. For additional information on how to debug cluster issues, see [Troubleshoot a Cluster \(p. 479\)](#).

Why haven't I received my Spot Instances?

Spot Instances are provisioned based on availability and bid price. If you haven't received the Spot Instances you requested, that means either your bid price is lower than the current Spot Price, or there is not enough supply at your bid price to fulfill your request.

Master and core instance groups are not fulfilled until all of the requested instances can be provisioned. Task nodes are fulfilled as they become available.

One way to address unfulfilled Spot Instance requests is to terminate the cluster and launch a new one, specifying a higher bid price. Reviewing the price history on Spot Instances tells you which bids have been successful in the recent past and can help you determine which bid is the best balance of cost savings and likelihood of being fulfilled. To review the Spot Instance price history, go to Spot Instances on the [Amazon EC2 Pricing page](#).

Another option is to change the type of instance you request. For example, if you requested four Extra Large instances and the request has not been filled after a period of time, you might consider relaunching the cluster and placing a request for four Large instances instead. Because the base rate is different for each instance type, you would want to adjust your bid price accordingly. For example, if you bid 80% of the on-demand rate for an Extra Large instance you might choose to adjust your bid price on the new Spot Instance request to reflect 80% of the on-demand rate for a Large instance.

The final variable in the fulfillment of a Spot Instance request is whether there is unused capacity in your region. You can try launching the Spot Instance request in a different region. Before selecting this option, however, consider the implications of data transfer across regions. For example, if the Amazon Simple Storage (Amazon S3) bucket housing your data is in region *us-east-1* and you launch a cluster as Spot Instances in *us-west-1*, the additional cross-region data transfer costs may outweigh any cost savings from using Spot Instances.

Why did my Spot Instances terminate?

By design, Spot Instances are terminated by Amazon EC2 when the Spot Instance price rises above your bid price.

If your bid price is equal to or lower than the Spot Instance price, the instances might have terminated normally at the end of the cluster, or they might have terminated because of an error. For more information about how to debug cluster errors, go to [Troubleshoot a Cluster \(p. 479\)](#).

How do I check the price history on Spot Instances?

To review the Spot Instance price history, go to Spot Instances on the [Amazon EC2 Pricing page](#). This pricing information is updated at regular intervals.

Configure the Software

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

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 \(AMI\) \(p. 48\)](#)
- [Choose a Version of Hadoop \(p. 115\)](#)
- [\(Optional\) Create Bootstrap Actions to Install Additional Software \(p. 124\)](#)

Choose an Amazon Machine Image (AMI)

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

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. Periodically, Amazon EMR updates these AMIs with new versions of 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 application 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 \(p. 53\)](#).

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 \(p. 49\)](#).

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 \(p. 49\)](#)
- [Default AMI and Hadoop Versions \(p. 50\)](#)
- [Specifying the AMI Version for a New Cluster \(p. 50\)](#)
- [View the AMI Version of a Running Cluster \(p. 51\)](#)
- [Amazon EMR AMIs and Hadoop Versions \(p. 53\)](#)
- [Amazon EMR AMI Deprecation \(p. 53\)](#)
- [AMI Versions Supported in Amazon EMR \(p. 53\)](#)

AMI Version Numbers

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.
- **No version specified**—In the Amazon EMR CLI, API, or SDK if you do not specify an AMI version, the cluster is launched with the default AMI for that tool. This option is not supported by the AWS CLI. For more information on the default AMI versions, see [Default AMI and Hadoop Versions \(p. 50\)](#).

Default AMI and Hadoop Versions

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

If you do not specify the AMI for the cluster (using the API or SDK), Amazon EMR launches your cluster with the default version. The default versions returned depend on the interface you use to launch the cluster. The AWS CLI requires you to specify the AMI version; it does not have a default AMI version.

Note

The default AMI is unavailable in the Asia Pacific (Sydney) region. Instead, fully specify the AMI or use the major-minor version.

Interface	Default AMI and Hadoop versions
API	AMI 1.0.1 with Hadoop 0.18
SDK	AMI 1.0.1 with Hadoop 0.18

Specifying the AMI Version for a New Cluster

You can specify which AMI version a new cluster should use when you launch it. For details about the default configuration and applications available on AMI versions, see [AMI Versions Supported in Amazon EMR \(p. 53\)](#).

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

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

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 3.2.3), 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 3.2.3 \  
--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 3.2.3 --applic  
ations 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 2.4.9, specifying `--ami-version 2.4` would launch a cluster using AMI 2.4.9.

```
aws emr create-cluster --name "Test cluster" --ami-version 2.4 --applications
  Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey -
-ininstance-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>.

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/>.
2. Click 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.

```
aws emr describe-cluster --cluster-id j-3QKHXXXXXXARD
```

The output shows the AMI version for the cluster:

```
{
  "Cluster": {
    "Status": {
      "Timeline": {
        "EndTime": 1412976409.738,
```

```
        "CreationDateTime": 1412976134.334
      },
      "State": "TERMINATED",
      "StateChangeReason": {
        "Message": "Terminated by user request",
        "Code": "USER_REQUEST"
      }
    },
    "Ec2InstanceAttributes": {
      "Ec2AvailabilityZone": "us-west-2c"
    },
    "Name": "Static AMI Version",
    "Tags": [],
    "TerminationProtected": false,
    "RunningAmiVersion": "2.4.8",
    "NormalizedInstanceHours": 0,
    "InstanceGroups": [
      {
        ...
      }
    ]
  }
}
```

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 Elastic MapReduce API Reference*.

```
<DescribeJobFlowsResponse xmlns="http://elasticmapreduce.amazonaws.com/doc/2009-03-31">
  <DescribeJobFlowsResult>
    <JobFlows>
      <member>
        ...
        <AmiVersion>
          2.1.3
        </AmiVersion>
        ...
      </member>
    </JobFlows>
  </DescribeJobFlowsResult>
  <ResponseMetadata>
    <RequestId>
      9cea3229-ed85-11dd-9877-6fad448a8419
    </RequestId>
  </ResponseMetadata>
</DescribeJobFlowsResponse>
```

Amazon EMR AMIs and Hadoop Versions

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

An AMI can contain multiple versions of Hadoop. If the AMI you specify has multiple versions of Hadoop available, you can select the version of Hadoop you want to run. You cannot specify a Hadoop version that is not available on the AMI. For a list of the versions of Hadoop supported on each AMI, go to [AMI Versions Supported in Amazon EMR \(p. 53\)](#).

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 deprecated, the cluster will not be affected. You will not, however, be able to create new clusters with the deprecated 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

Amazon EMR supports the AMI versions listed in the following tables. You must specify the AMI version to use when you create a cluster using the AWS CLI and the console. If you do not specify an AMI version in the SDK, API, or Amazon EMR CLI, Amazon EMR creates the cluster using the default AMI version. For information about default AMI configurations, see [Default AMI and Hadoop Versions \(p. 50\)](#).

Hadoop 2 AMI Versions

AMI Version	Includes	Notes	Release Date
3.9.0	<ul style="list-style-type: none">• AWS SDK for Java 1.9• AWS SDK for Ruby 1.9• Amazon Linux version 2015.03• Hadoop 2.4.0 (p. 117)• Hive 0.13.1 (p. 259)• Hue 3.7.1• Pig 0.12.0 (p. 304)• Hbase 0.94.18 (p. 311)• Impala 1.2.4 (p. 291)• Mahout 0.9 (p. 123)• 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		19 August 2015

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
		<p>This Amazon EMR AMI version provides the following major bug fixes:</p> <ul style="list-style-type: none">• 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 information, see HBase Shell Excessive Debug Log-	

AMI Version	Includes	Notes	Release Date
		ging (p. 481)	
3.3.0	<ul style="list-style-type: none"> • AWS SDK for Ruby 1.9 • Amazon Linux version 2014.03 • Hadoop 2.4.0 (p. 117) • Hive 0.13.1 (p. 259) • Hue 3.6 • Pig 0.12.0 (p. 304) • Hbase 0.94.18 (p. 311) • Impala 1.2.4 (p. 291) • Mahout 0.9 (p. 123) • 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 	<p>This Amazon EMR AMI version provides the following features and bug fixes:</p> <p>Major Feature Updates</p> <ul style="list-style-type: none"> • 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. 334). • Include Oozie as part of Hue release. 	6 November 2014

Hadoop 1 AMI Versions

AMI Version	Description	Release Date
2.4.11	This Amazon EMR AMI version provides the following: <ul style="list-style-type: none">• Minor performance-related bug fixes.	26 February 2015
2.4.2	Same as the previous AMI version, with the following additions: <ul style="list-style-type: none">• 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

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

AMI Version	Includes	Notes	Release Date
3.8.0	<ul style="list-style-type: none">• AWS SDK for Java 1.9• AWS SDK for Ruby 1.9• Amazon Linux version 2015.03• Hadoop 2.4.0 (p. 117)• Hive 0.13.1 (p. 259)• Hue 3.7• Pig 0.12.0 (p. 304)• Hbase 0.94.18 (p. 311)• Impala 1.2.4 (p. 291)• Mahout 0.9 (p. 123)• 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		10 June 2015

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
		<p>This Amazon EMR AMI version provides the following:</p> <p>Major Feature Release</p> <ul style="list-style-type: none"> • Support for Apache Spark. For more information, see Apache Spark (p.268). <p>Major Bug Fixes</p> <ul style="list-style-type: none"> • 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 table operation. • Includes the following patches: HIVE-8746, HIVE-8566, HIVE-8162, PIG-4496 	

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
		<p>Other Issues</p> <ul style="list-style-type: none">• Added srcPrefixes-File to S3DistCp. For more information, see S3DistCp Options (p. 377)• The Amazon EMR-DynamoDB connector now allows customers to load a custom AWS credentials provider for use with the connector.• Remove unused property, <code>skipScratch</code>, from <code>hive-default.xml</code>.	

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
3.7.0	<ul style="list-style-type: none">• AWS SDK for Java 1.9• AWS SDK for Ruby 1.9• Amazon Linux version 2015.03• Hadoop 2.4.0 (p. 117)• Hive 0.13.1 (p. 259)• Hue 3.7• Pig 0.12.0 (p. 304)• Hbase 0.94.18 (p. 311)• Impala 1.2.4 (p. 291)• Mahout 0.9 (p. 123)• 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		21 April 2015

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
		<p>This Amazon EMR AMI version provides the following feature release:</p> <p>Major Bug Fixes</p> <ul style="list-style-type: none"> • <code>/var/log</code> is moved to <code>/mnt/var/log</code>. Any files written to <code>/var/log</code> will be written to <code>/mnt/var/log</code> as there is now a symbolic link between the two paths. • Addresses an issue where some components would cause certain scripts running out of <code>/etc/init.d</code> 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 <code>easy_install</code>. • The default Ruby will now match 	

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
		<p>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.</p> <p>Other Issues</p> <ul style="list-style-type: none"> • 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 (50) 	

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
3.6.0	<ul style="list-style-type: none">• AWS SDK for Java 1.9• AWS SDK for Ruby 1.9• Amazon Linux version 2014.09• Hadoop 2.4.0 (p. 117)• Hive 0.13.1 (p. 259)• Hue 3.7• Pig 0.12.0 (p. 304)• Hbase 0.94.18 (p. 311)• Impala 1.2.4 (p. 291)• Mahout 0.9 (p. 123)• 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	<p>This Amazon EMR AMI version provides the following feature release:</p> <p>Major Feature Release</p> <ul style="list-style-type: none">• EMRFS supports Amazon S3 client-side encryption. For more information, see Using Amazon S3 Client-Side Encryption in EMRFS (p. 157).	24 March 2015

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
3.5.0	<ul style="list-style-type: none"> • AWS SDK for Java 1.9 • AWS SDK for Ruby 1.9 • Amazon Linux version 2014.09 • Hadoop 2.4.0 (p. 117) • Hive 0.13.1 (p. 259) • Hue 3.7 • Pig 0.12.0 (p. 304) • Hbase 0.94.18 (p. 311) • Impala 1.2.4 (p. 291) • Mahout 0.9 (p. 123) • 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 	<p>This Amazon EMR AMI version provides the following features and bug fixes:</p> <p>Major Bug Fixes</p> <ul style="list-style-type: none"> • Fixes a bug which prevented s3distcp from using temporary credentials (such as with Amazon STS tokens). • Fixes a bug that caused Amazon S3 multipart upload to hang on an individual upload. • Fixes performance issues with the Hive-DynamoDB connector. • Fixes an issue encountered in the Hue redirection middleware, which caused redirection to fail when redirected to SSL endpoints 	10 March 2015

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
3.4.0	<ul style="list-style-type: none">• AWS SDK for Java 1.9• AWS SDK for Ruby 1.9• Amazon Linux version 2014.09• Hadoop 2.4.0 (p. 117)• Hive 0.13.1 (p. 259)• Hue 3.7• Pig 0.12.0 (p. 304)• Hbase 0.94.18 (p. 311)• Impala 1.2.4 (p. 291)• Mahout 0.9 (p. 123)• 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		26 February 2015

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
		<p>This Amazon EMR AMI version provides the following features and bug fixes:</p> <p>Major Feature Updates</p> <ul style="list-style-type: none"> • 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 Amazon Kinesis connector which requires customers to co-locate the DynamoDB 	

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
		<p>table with their EMR cluster in the same region.</p> <p>Major Bug Fixes</p> <ul style="list-style-type: none">• Minor performance-related bug fixes.• Includes this patch: HIVE-7323	

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
3.3.2	<ul style="list-style-type: none">• AWS SDK for Ruby 1.9• Amazon Linux version 2014.03• Hadoop 2.4.0 (p. 117)• Hive 0.13.1 (p. 259)• Hue 3.6• Pig 0.12.0 (p. 304)• Hbase 0.94.18 (p. 311)• Impala 1.2.4 (p. 291)• Mahout 0.9 (p. 123)• 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		4 February 2015

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
		<p>This Amazon EMR AMI version provides the following features and bug fixes:</p> <p>Major Feature Updates</p> <ul style="list-style-type: none"> Fixes performance issues in DynamoDB connector <p>Major Bug Fixes</p> <ul style="list-style-type: none"> 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 Hue sample tables. 	

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
		<ul style="list-style-type: none">• Fixed an issue encountered in Hue when saving results from a multiple statement Hive query.• Fixes a mismatch between the installed default Python and pip.• Includes this patch: HIVE-7426	

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
3.3.1	<ul style="list-style-type: none">• AWS SDK for Ruby 1.9• Amazon Linux version 2014.03• Hadoop 2.4.0 (p. 117)• Hive 0.13.1 (p. 259)• Hue 3.6• Pig 0.12.0 (p. 304)• Hbase 0.94.18 (p. 311)• Impala 1.2.4 (p. 291)• Mahout 0.9 (p. 123)• 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		20 November 2014

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
		<p>This Amazon EMR AMI version provides the following features and bug fixes:</p> <p>Major Bug Fixes</p> <ul style="list-style-type: none"> • 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 <code>redirect_whitelist</code> Hue configuration option allowing whitelisting of domains 	

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
		<p>to which Hue can re-direct.</p> <ul style="list-style-type: none">• Fixed an issue with loading saved queries in Hive Editor when browsing Hue in Firefox.• Removed permission buttons in the Hue Amazon S3 browser as those operations are not supported.• Fixed an issue with the configure-hadoop script which caused a loss of file permissions when used.	

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
3.3.0	<ul style="list-style-type: none"> • AWS SDK for Ruby 1.9 • Amazon Linux version 2014.03 • Hadoop 2.4.0 (p. 117) • Hive 0.13.1 (p. 259) • Hue 3.6 • Pig 0.12.0 (p. 304) • Hbase 0.94.18 (p. 311) • Impala 1.2.4 (p. 291) • Mahout 0.9 (p. 123) • 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 	<p>This Amazon EMR AMI version provides the following features and bug fixes:</p> <p>Major Feature Updates</p> <ul style="list-style-type: none"> • 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. 334). • Include Oozie as part of Hue release. 	6 November 2014

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
3.2.3	<ul style="list-style-type: none">• AWS SDK for Ruby 1.9• Amazon Linux version 2014.03• Hadoop 2.4.0 (p. 117)• Hive 0.13.1 (p. 259)• Pig 0.12 (p. 304)• Hbase 0.94.18 (p. 311)• Impala 1.2.4 (p. 291)• Mahout 0.9 (p. 123)• 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		31 October 2014

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
		<p>This Amazon EMR AMI version provides the following features and bug fixes:</p> <p>Major Feature Updates</p> <ul style="list-style-type: none"> • EMRFS now supports Amazon S3 eventual consistency notifications for Amazon SQS and eventual consistency metrics for Cloud-Watch. • Performance optimizations for Amazon S3 multipart uploads with EMRFS. • The <code>configure-hadoop</code> 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. 	

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
		Other Feature Updates	

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
		<ul style="list-style-type: none"> • 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 is in <code>/mnt/var/log/hadoop</code>, the 	

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
		<p>compressed log is stored in /mnt/elasticmapreduce until the log retention period expires.</p> <ul style="list-style-type: none"> • Compressed log files larger than 4GB are not uploaded to Amazon S3. • A temporary directory cleaner is now included that cleans up temporary files in <code>/mnt/tmp</code>. <p>Major Bug Fixes</p>	

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
		<ul style="list-style-type: none"> • Fixes an issue where the Hive server does not start after installation. • Fixes an issue where the Hive web interface does not function properly. • Fixes an issue where the HBase restore procedure corrupts the source cluster if it is still running. • Adds support for reuse of file statuses in the Pig Zebra format split calculation logic to improve performance. • Adds multithreaded creation of Pig Zebra format indexes during MapReduce job closure to improve performance. • Adds support for the legacy location of <code>org.apache.pig.jar</code> to <code>org.apache.pig.jar</code>. • Fixes an issue where 	

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
		<p>Pig does not use automatic parallelism if the source file system is Amazon S3.</p> <ul style="list-style-type: none">• Fixes an issue where S3DistCp ignores all <code>-D</code> CLI parameters.• Includes the following patches: YARN-2008, YARN-1857, YARN-1198, YARN-1680, MAPREDUCE-6111, HDFS-7005, HIVE-7147, HIVE-8137, HIVE-4629, HIVE-6245	

AMI Version	Includes	Notes	Release Date
3.2.1	<ul style="list-style-type: none"> • Amazon Linux version 2014.03 • Hadoop 2.4.0 (p. 117) • Hive 0.13.1 (p. 259) • Pig 0.12 (p. 304) • Hbase 0.94.18 (p. 311) • Impala 1.2.4 (p. 291) • Mahout 0.9 (p. 123) • 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 	<p>This Amazon EMR AMI version provides the following fixes:</p> <p>Major Bug Fixes</p> <ul style="list-style-type: none"> • Added EM-RFS consistent view, which allows customers to track the consistent view of objects written by Amazon EMR to Amazon S3. For more information, see the section called “Consistent View” (p. 141) <p>Other Bug Fixes</p> <ul style="list-style-type: none"> • Fixes a port forwarding issue encountered with Hive. • Fixes an issue encountered with Hive-MetaStoreChecker. • Included a fix for: HIVE-7085. 	16 September 2014

AMI Version	Includes	Notes	Release Date
3.2.0	<ul style="list-style-type: none"> • Amazon Linux version 2014.03 • Hadoop 2.4.0 (p. 117) • Hive 0.13.1 (p. 259) • Pig 0.12 (p. 304) • Hbase 0.94.18 (p. 311) • Impala 1.2.4 (p. 291) • Mahout 0.9 (p. 123) • 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 	<p>This Amazon EMR AMI version provides the following features:</p> <p>Major Feature Updates</p> <ul style="list-style-type: none"> • Added Apache Hive 0.13.1. For more information, go to Hive version documentation (p. 259) and http://hive.apache.org/downloads.html. • Provided a change to the connector for Amazon Kinesis that takes a flag, <code>kinesis.iteration.timeout.ignore.failure</code>, to allow a job to continue checkpointing even if it has reached the timeout value. <p>Major Bug Fixes</p> <ul style="list-style-type: none"> • Included the following YARN patches: YARN-1718, YARN-1923, YARN-1889. 	3 September 2014

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
3.1.4	<ul style="list-style-type: none">• AWS SDK for Ruby 1.9• Amazon Linux version 2014.03• Hadoop 2.4.0 (p. 117)• Hive 0.11.0.2 (p. 259)• Pig 0.12 (p. 304)• Hbase 0.94.18 (p. 311)• Impala 1.2.4 (p. 291)• Mahout 0.9 (p. 123)• 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		31 October 2014

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
		<p>This Amazon EMR AMI version provides the following features and bug fixes:</p> <p>Major Feature Updates</p> <ul style="list-style-type: none"> • EMRFS now supports Amazon S3 eventual consistency notifications for Amazon SQS and eventual consistency metrics for Cloud-Watch. • Performance optimizations for Amazon S3 multipart uploads with EMRFS. • The <code>configure-hadoop</code> 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. 	

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
		Other Feature Updates	

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
		<ul style="list-style-type: none"> • 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 is in <code>/mnt/var/log/hadoop</code>, the 	

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
		<p>compressed log is stored in /mnt/elasticmapreduce until the log retention period expires.</p> <ul style="list-style-type: none"> • Compressed log files larger than 4GB are not uploaded to Amazon S3. • A temporary directory cleaner is now included that cleans up temporary files in <code>/mnt/tmp</code>. <p>Major Bug Fixes</p>	

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
		<ul style="list-style-type: none">• Fixes an issue where Pig does not use automatic parallelism if the source file system is Amazon S3.• Fixes an issue where the HBase restore procedure corrupts the source cluster if it is still running.• Fixes an issue where S3DistCp ignores all <code>-D</code> CLI parameters.• Includes the following patches: YARN-2008, YARN-1857, YARN-1198, YARN-1680, MAPREDUCE-6111, HDFS-7005, HIVE-2777	

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
3.1.2	<ul style="list-style-type: none"> • Amazon Linux version 2014.03 • Hadoop 2.4.0 (p. 117) • Hive 0.11.0.2 (p. 259) • Pig 0.12 (p. 304) • Hbase 0.94.18 (p. 311) • Impala 1.2.4 (p. 291) • Mahout 0.9 (p. 123) • 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 	<p>This Amazon EMR AMI version provides the following bug fixes:</p> <p>Major Bug Fixes</p> <ul style="list-style-type: none"> • Fixes an issue encountered when using AWS CLI on the image. • Fixes a port forwarding bug encountered with Hive. • Fixes an issue encountered with Hive-MetaStoreChecker. • Includes the following patches: MAPREDUCE-5956, YARN-2026, YARN-2187, YARN-2214, YARN-2111, YARN-2053, YARN-2074, YARN-2030, YARN-2155, YARN-2128, YARN-1913, YARN-596, YARN-2073, YARN-2017. 	16 September 2014

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
3.1.1	<ul style="list-style-type: none">• Amazon Linux version 2014.03• Hadoop 2.4.0 (p. 117)• Hive 0.11.0.2 (p. 259)• Pig 0.12 (p. 304)• Hbase 0.94.18 (p. 311)• Impala 1.2.4 (p. 291)• Mahout 0.9 (p. 123)• 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		15 August 2014

AMI Version	Includes	Notes	Release Date
		<p>This Amazon EMR AMI version provides the following bug fixes and enhancements:</p> <p>Major Feature Updates</p> <ul style="list-style-type: none"> • 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: the section called “Submit Work to a Cluster” (p.473). • Added Enhanced Networking for C3, R3, and I2 instance types. For more information, see the Enhanced Networking section in the AWS EC2 User Guide. • Updated Java version to 7u65. For more information, go to JDK 7 Update 65 Re- 	

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
		<p>lease.</p> <ul style="list-style-type: none">• Added support for AWS SDK 1.7.8 for all components except Impala.• Improved scalability of CloudWatch metrics support for Amazon EMR.• Enabled <code>fuse_dfs</code>. For more information, see the Mountable-HDFS website. <p>Major Bug Fixes</p>	

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
		<ul style="list-style-type: none"> • 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. • Fixed a bug that prevented users from launching Hive in local mode. • Fixed a bug that prevented NodeManager from using all available mount-points, which resulted in issues using ephemeral drives on certain instances. • Fixed an issue in ResourceManager, which prevented users from accessing user interfaces on loc- 	

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
		<p>alhost.</p> <ul style="list-style-type: none"> • Fixed a bug in JobHistory that may have prevented storage of JobHistory logs in Amazon S3 buckets. • Included the following Hadoop patches: HDFS-6701, HDFS-6460, HADOOP-10456, HD-FS-6268, MAPREDUCE-5900. • Backport of YARN-1864 to Hadoop 2. • Fixed a performance regression in Hive. • Hive is compiled against JDK 1.7 • Included the following Hive patches: HIVE-6938, HIVE-7429. • Fixed several Hbase bugs. • The connector for Amazon Kinesis for now supports all regions where Amazon Kinesis is available. 	

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
3.1.0	<ul style="list-style-type: none">• Amazon Linux version 2014.03• Hadoop 2.4.0 (p. 117)• Hive 0.11.0.2 (p. 259)• Pig 0.12 (p. 304)• Hbase 0.94.18 (p. 311)• Impala 1.2.4 (p. 291)• Mahout 0.9 (p. 123)• Java: Oracle/Sun jdk-7u60• Perl 5.16.3• PHP 5.3.28• Python 2.6.9• R 3.0.2• Ruby 2.0		15 May 2014

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
		<p>This Amazon EMR AMI version provides the following features:</p> <p>Major Feature Updates</p> <ul style="list-style-type: none"> • Significant updates and improvements to support new packages and Amazon EMR features. For more information, see the following: Hadoop 2.4.0 (p. 117), Pig 0.12 (p. 304), Impala 1.2.4 (p. 291), Hbase 0.94.18 (p. 311), Mahout 0.9 (p. 123), and Ruby 2.0. • Enabled Amazon S3 server side encryption with Hadoop. For more information, see Create a Cluster With Amazon S3 Server-Side Encryption Enabled (p. 156). • Updated Java version to 7u60 (early access release). For more inform- 	

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
		<p>ation, go to JDK 7 Update 60 Early Access Release.</p> <ul style="list-style-type: none">• Update Jetty to version 6.1.26.emr to fix Hadoop MapReduce issue MAPREDUCE-2980.• Changed the log level for Hive UDAFPercntile to DEBUG. <p>Major Bug Fixes</p>	

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
		<ul style="list-style-type: none"> • Fixed an issue encountered when no <code>log-uri</code> value is specified at cluster creation. • Fixed version utility to accurately display Amazon Hadoop Distribution version. • Fixed Hadoop to accept <code>HA-DOOP_NAMENODE_HEAPSIZE</code> and <code>HA-DOOP_DATANODE_HEAPSIZE</code> memory setting values. • Replaced <code>YARN_HEAPSIZE</code> with <code>YARN_RESOURCEMANAGER_HEAPSIZE</code>, <code>YARN_NODEMANAGER_HEAPSIZE</code>, and <code>YARN_RESOURCESERVER_HEAPSIZE</code> to allow more granularity when configuring. For more information, see Configuration of hadoop-user-env.sh (p.526). • Added memory setting, <code>HA-DOOP_JOB_HIS-</code> 	

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
		<p>TORYSERV- ER_HEAPSIZE.</p> <ul style="list-style-type: none">• Fixed an issue encountered with <code>hdfs -get</code> when used with an Amazon S3 path.• Fixed an issue with the HTTPFS service for Hadoop.• Fixed an issue that caused job failures after a previous job was killed.• Other improvements and bug fixes.	

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
3.0.4	<ul style="list-style-type: none"> • Amazon Linux version 2013.09 • Hadoop 2.2.0 (p. 118) • Hive 0.11.0.2 (p. 259) • Pig 0.11.1.1 (p. 304) • Hbase 0.94.7 (p. 310) • Impala 1.2.1 (p. 291) • Mahout 0.8 • Java: Oracle/Sun jdk-7u60 • Perl 5.10.1 • PHP 5.3.28 • Python 2.6.9 • R 3.0.1 • Ruby 1.8.7 	<p>This Amazon EMR AMI version provides the following features:</p> <ul style="list-style-type: none"> • Adds a connector for Amazon Kinesis, which allows users to process streaming data using standard Hadoop and ecosystem tools within Amazon EMR clusters. For more information, see Analyze Amazon Kinesis Data (p. 344). • Fixes an issue in the <code>yarn-site.xml</code> configuration file, which resulted in the JobHistory server not being fully configured. • Adds support for AWS SDK 1.7.0. 	19 February 2014

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
3.0.3	<ul style="list-style-type: none">• Amazon Linux version 2013.03• Hadoop 2.2.0 (p. 118)• Hive 0.11.0.2 (p. 259)• Pig 0.11.1.1 (p. 304)• Hbase 0.94.7 (p. 310)• Impala 1.2.1 (p. 291)• Mahout 0.8• Oracle/Sun jdk-7u45• Perl 5.10.1• PHP 5.3.28• Python 2.6.9• R 3.0.1• Ruby 1.8.7		11 February 2014

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
		<p>This Amazon EMR AMI version provides the following features:</p> <ul style="list-style-type: none"> • Adds support for AWS SDK 1.6.10. • Upgrades HttpClient to version 4.2 to be compatible with AWS SDK 1.6.10. • Fixes a problem related to orphaned Amazon EBS volumes. • Adds support for Hive 0.11.0.2. • Upgrades Protobuf to version 2.5. <p>Note The upgrade to Protobuf 2.5 requires you to regenerate and recompile any of your</p>	

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
		<p>Java code that was previously generated by the protocol tool.</p>	
3.0.2	<ul style="list-style-type: none"> • Amazon Linux version 2013.03 • Hadoop 2.2.0 (p. 118) • Hive 0.11.0.2 (p. 259) • Pig 0.11.1.1 (p. 304) • Hbase 0.94.7 (p. 310) • Impala 1.2.1 (p. 291) • Mahout 0.8 • Oracle/Sun jdk-7u45 • Perl 5.10.1 • PHP 5.3.28 • Python 2.6.9 • R 3.0.1 • Ruby 1.8.7 	<p>This Amazon EMR AMI version provides the following features:</p> <ul style="list-style-type: none"> • Adds support for Impala 1.2.1 with Hadoop 2. For more information, see Impala (p. 279). • Changes the uploadMultiParts function to use a retry policy. 	12 December 2013
3.0.1	<ul style="list-style-type: none"> • Amazon Linux version 2013.03 • Hadoop 2.2.0 (p. 118) • Hive 0.11.0.1 (p. 259) • Pig 0.11.1.1 (p. 304) • Hbase 0.94.7 (p. 310) • Impala 1.2.1 (p. 291) • Mahout 0.8 • Oracle/Sun jdk-7u45 • Perl 5.10.1 • PHP 5.3.28 • Python 2.6.9 • R 3.0.1 • Ruby 1.8.7 	<p>This Amazon EMR AMI version provides the following features:</p> <ul style="list-style-type: none"> • Adds support for viewing Hadoop 2 task attempt logs in the EMR console. • Fixes an issue with R 3.0.1. 	8 November 2013

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Includes	Notes	Release Date
3.0.0	<ul style="list-style-type: none"> • Amazon Linux version 2013.03 • Hadoop 2.2.0 (p. 118) • Hive 0.11.0.1 (p. 259) • Pig 0.11.1.1 (p. 304) • Hbase 0.94.7 (p. 310) • Impala 1.2.1 (p. 291) • Mahout 0.8 • Oracle/Sun jdk-7u45 • Perl 5.10.1 • PHP 5.3.28 • Python 2.6.9 • R 3.0.1 • Ruby 1.8.7 	<p>This new major Amazon EMR AMI version provides the following features:</p> <ul style="list-style-type: none"> • This Amazon EMR AMI is based on the Amazon Linux Release 2012.09. For more information, see Amazon Linux AMI 2012.09 Release Notes. • Adds support for Hadoop 2.2.0. For more information, see Supported Hadoop Versions (p. 115). • Adds support for HBase 0.94.7. For more information, go to the Apache web site. • Adds Java 7 support for Hadoop, HBase, and Pig. 	28 October 2013

Deprecated Hadoop 1 and Earlier AMIs

AMI Version	Description	Release Date
2.4.10	<p>This Amazon EMR AMI version provides the following bug fixes:</p> <ul style="list-style-type: none"> 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	<p>This Amazon EMR AMI version provides the following bug fixes:</p> <ul style="list-style-type: none"> 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	<p>This Amazon EMR AMI version provides the following feature and bug fixes:</p> <p>Major Feature Update</p> <ul style="list-style-type: none"> 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. 473). <p>Major Bug Fixes</p> <ul style="list-style-type: none"> Fixes an issue with <code>hbase-user-env.sh</code>, which resulted in Hbase ignoring any settings made by this script. Fixes an issue with <code>s3distcp</code> where using the <code>.*</code> regular expression causes an error. 	16 September 2014
2.4.7	<p>In addition to other enhancements and bug fixes, this version of Amazon EMR AMI corrects the following problems:</p> <ul style="list-style-type: none"> Fixes an issue with logs stored in <code>/mnt/var/log</code>, which may consume all of the volume's disk space. Fixes a deadlock issue encountered when adding steps. 	30 July 2014

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Description	Release Date
2.4.6	<p>This Amazon EMR AMI version provides the following features:</p> <ul style="list-style-type: none">• 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
2.4.5	<p>This Amazon EMR AMI version provides the following features:</p> <ul style="list-style-type: none">• 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 (Sao 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. <p>Note</p> <p>The upgrade to Protobuf 2.5 requires you to re-generate and recompile any of your Java code that was previously generated by the protoc tool.</p> <ul style="list-style-type: none">• 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 <code>log-uri</code> is specified at cluster creation.• Fixes version utility to accurately display Amazon Hadoop Distribution version.• Other improvements and bug fixes.	27 March 2014

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Description	Release Date
2.4.3	<p>This Amazon EMR AMI version provides the following features:</p> <ul style="list-style-type: none"> • 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. <p>Note The upgrade to Protobuf 2.5 requires you to re-generate and recompile any of your Java code that was previously generated by the protoc tool.</p>	3 January 2014
2.4.1	<p>Same as the previous AMI version, with the following additions:</p> <ul style="list-style-type: none"> • 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	<p>Same as the previous AMI version, with the following additions:</p> <ul style="list-style-type: none"> • 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

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Description	Release Date
2.3.6	<p>Same as 2.3.5, with the following additions:</p> <ul style="list-style-type: none"> 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
2.3.5	<p>Same as 2.3.3, with the following additions:</p> <ul style="list-style-type: none"> 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 <code>--s3ServerSideEncryption</code> option to the S3DistCp tool. For more information, see S3DistCp Options (p. 377). 	26 April 2013
2.3.4	<p>Note Because of an issue with AMI 2.3.4, this version is deprecated. We recommend that you use a different AMI version instead.</p>	16 April 2013
2.3.3	<p>Same as 2.3.2, with the following additions:</p> <ul style="list-style-type: none"> Improved CloudWatch LiveTaskTracker metric to take into account expired Hadoop TaskTrackers and minor improvements in Hadoop. 	01 March 2013
2.3.2	<p>Same as 2.3.1, with the following additions:</p> <ul style="list-style-type: none"> Fixes an issue which prevented customers from using the debugging feature in the Amazon EMR console. 	07 February 2013
2.3.1	<p>Same as 2.3.0, with the following additions:</p> <ul style="list-style-type: none"> Improves support for clusters running on hs1.8xlarge instances. 	24 December 2012
2.3.0	<p>Same as 2.2.4, with the following additions:</p> <ul style="list-style-type: none"> Adds support for IAM roles. For more information, see Configure IAM Roles for Amazon EMR (p. 185). 	20 December 2012

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Description	Release Date
2.2.4	<p>Same as 2.2.3, with the following additions:</p> <ul style="list-style-type: none"> 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
2.2.3	<p>Same as 2.2.1, with the following additions:</p> <ul style="list-style-type: none"> Improves HBase backup functionality. Updates the AWS SDK for Java to version 1.3.23. Resolves issues with the job tracker user interface. Improves Amazon S3 file system handling in Hadoop. Improves to NameNode functionality in Hadoop. 	30 November 2012
2.2.2	<p>Note Because of an issue with AMI 2.2.2, this version is deprecated. We recommend that you use a different AMI version instead.</p>	23 November 2012
2.2.1	<p>Same as 2.2.0, with the following additions:</p> <ul style="list-style-type: none"> 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. 167). 	30 August 2012
2.2.0	<p>Same as 2.1.3, with the following additions:</p> <ul style="list-style-type: none"> Adds support for Hadoop 1.0.3. No longer includes Hadoop 0.18 and Hadoop 0.20.205. <p>Operating system: Debian 6.0.5 (Squeeze) Applications: Hadoop 1.0.3, Hive 0.8.1.3, Pig 0.9.2.2, HBase 0.92.0 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</p>	6 August 2012
2.1.4	<p>Same as 2.1.3, with the following additions:</p> <ul style="list-style-type: none"> 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. 167). 	30 August 2012

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Description	Release Date
2.1.3	Same as 2.1.2, with the following additions: <ul style="list-style-type: none">• Fixes issues in HBase.	6 August 2012
2.1.2	Same as 2.1.1, with the following additions: <ul style="list-style-type: none">• Support for CloudWatch metrics when using MapR. Improve reliability of reporting metrics to CloudWatch.	6 August 2012
2.1.1	Same as 2.1.0, with the following additions: <ul style="list-style-type: none">• 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: <ul style="list-style-type: none">• Supports launching HBase clusters. For more information see Apache HBase (p. 310).• Supports running MapR Edition M3 and Edition M5. For more information, see Using the MapR Distribution for Hadoop (p. 218).• Enables HDFS append by default; <code>dfs.support.append</code> is set to <code>true</code> in <code>hdfs/hdfs-default.xml</code>. The default value in code is also set to <code>true</code>.• Fixes a race condition in instance controller.• Changes <code>mapreduce.user.classpath.first</code> to default to <code>true</code>. 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

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Description	Release Date
2.0.5	<p>Note Because of an issue with AMI 2.0.5, this version is deprecated. We recommend that you use a different AMI version instead.</p> <p>Same as AMI 2.0.4, with the following additions:</p> <ul style="list-style-type: none"> • 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 <code>mapreduce.jobtracker.system.dir.permission</code> that can be used to set permissions on the system directory. For more information, see Setting Permissions on the System Directory (p. 200). • 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/. • 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
2.0.4	<p>Same as AMI 2.0.3, with the following additions:</p> <ul style="list-style-type: none"> • Changes the default for <code>fs.s3n.blockSize</code> to 33554432 (32MiB). • Fixes a bug in reading zero-length files from Amazon S3. 	30 January 2012
2.0.3	<p>Same as AMI 2.0.2, with the following additions:</p> <ul style="list-style-type: none"> • Adds support for Amazon EMR metrics in CloudWatch. • Improves performance of seek operations in Amazon S3. 	24 January 2012

Amazon Elastic MapReduce Developer Guide
Choose an Amazon Machine Image (AMI)

AMI Version	Description	Release Date
2.0.2	<p>Same as AMI 2.0.1, with the following additions:</p> <ul style="list-style-type: none"> • Adds support for the Python API Dumbo. For more information about Dumbo, go to https://github.com/klbostee/dumbo/wiki/. • The AMI now runs the Network Time Protocol Daemon (NTPD) by default. For more information about NTPD, go to http://en.wikipedia.org/wiki/Ntpd. • 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 <code>fs.s3n.blockSize</code> parameter. You set this parameter by using the <code>configure-hadoop</code> bootstrap action. The default value is 9223372036854775807 (8 EiB). • Adds a <code>/dev/sd</code> symlink for each <code>/dev/xvd</code> device. For example, <code>/dev/xvdb</code> now has a symlink pointing to it called <code>/dev/sdb</code>. Now you can use the same device names for AMI 1.0 and 2.0. 	17 January 2012
2.0.1	<p>Same as AMI 2.0 except for the following bug fixes:</p> <ul style="list-style-type: none"> • Task attempt logs are pushed to Amazon S3. • Fixed <code>/mnt</code> mounting on 32-bit AMIs. • Uses Debian 6.0.3 (Squeeze) as the operating system. 	19 December 2011
2.0.0	<p>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</p> <p>Note: Added support for the Snappy compression/decompression library.</p>	11 December 2011
1.0.1	<p>Same as AMI 1.0 except for the following change:</p> <ul style="list-style-type: none"> • Updates <code>sources.list</code> to the new location of the Lenny distribution in <code>archive.debian.org</code>. 	3 April 2012

AMI Version	Description	Release Date
1.0.0	<p>Operating system: Debian 5.0 (Lenny)</p> <p>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)</p> <p>Languages: Perl 5.10.0, PHP 5.2.6, Python 2.5.2, R 2.7.1, Ruby 1.8.7</p> <p>File system: ext3 for root and ephemeral</p> <p>Kernel: Red Hat</p> <p>Note: <i>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.</i></p>	26 April 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 \(p. 218\)](#).

Topics

- [Supported Hadoop Versions \(p. 115\)](#)
- [How Does Amazon EMR Hadoop Differ from Apache Hadoop? \(p. 120\)](#)
- [Hadoop Patches Applied in Amazon EMR \(p. 120\)](#)
- [Supported Mahout Versions \(p. 123\)](#)

Supported Hadoop Versions

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

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.

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

For details about the default configuration and software available on AMIs used by Amazon Elastic MapReduce (Amazon EMR) see [Choose an Amazon Machine Image \(AMI\) \(p. 48\)](#).

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 \(p. 53\)](#).

- 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.
 - 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-count 5 --instance-type m3.xlarge
```

- 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-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 Logging and Debugging \(Optional\)](#) (p. 200).
- 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. 201).

Hadoop 2.2.0 New Features

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

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 \(p. 446\)](#).
- 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

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

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 \(p. 446\)](#).
- Updated Hadoop configuration files. For more information, see [JSON Configuration Files \(p. 522\)](#).
- Changes to bootstrap actions that configure Hadoop daemons. For more information, see [\(Optional\) Create Bootstrap Actions to Install Additional Software \(p. 124\)](#).

Considerations for Moving to Hadoop 2.2.0

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

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:

```
<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. 177).

For a list of the patches applied to the Amazon EMR version of Hadoop 1.0.3, see [Hadoop 1.0.3 Patches](#) (p. 121).

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.
- 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:

Parameter	Definition
<code>-files</code>	Specifies comma-separated files to copy to the map reduce cluster.
<code>-archives</code>	Specifies comma-separated archives to restore to the compute machines.
<code>-D</code>	Specifies a value for the key you enter, in the form of <code><key>=<value></code> .

For a list of the patches applied to the Amazon EMR version of Hadoop 0.20.205, see [Hadoop 0.20.205 Patches \(p. 121\)](#).

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 \(p. 120\)](#).

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 Elastic MapReduce (Amazon EMR) team has applied to the Hadoop versions loaded on Amazon EMR AMIs.

Topics

- [Hadoop 1.0.3 Patches \(p. 121\)](#)
- [Hadoop 0.20.205 Patches \(p. 121\)](#)

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.

Patch	Description
All of the patches applied to the Amazon EMR version of Hadoop 0.20.205.	See Hadoop 0.20.205 Patches (p. 121) for details.
HADOOP-5861	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 https://issues.apache.org/jira/browse/HADOOP-5861 . Status: Fixed Fixed in AWS Hadoop Version: 1.0.3 Fixed in Apache Hadoop Version: 0.21.0
HADOOP-6346	Supports specifying a pattern to RunJar.unJar that determines which files are unpacked. For more information, go to https://issues.apache.org/jira/browse/HADOOP-6346 . Status: Fixed Fixed in AWS Hadoop Version: 1.0.3 Fixed in Apache Hadoop Version: 0.21.0
MAPREDUCE-967	Changes the TaskTracker node so it does not fully unjar job jars into the job cache directory. For more information, go to https://issues.apache.org/jira/browse/MAPREDUCE-967 . Status: Fixed Fixed in AWS Hadoop Version: 1.0.3 Fixed in Apache Hadoop Version: 0.21.0
MAPREDUCE-2219	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 https://issues.apache.org/jira/browse/MAPREDUCE-2219 . Status: Fixed Fixed in AWS Hadoop Version: 1.0.3 Fixed in Apache Hadoop Version: 0.22.0

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.

Amazon Elastic MapReduce Developer Guide
Choose a Version of Hadoop

Patch	Description
Add hadoop-lzo	<p>Install the hadoop-lzo third-party package. For more information about hadoop-lzo, go to https://github.com/kevinweil/hadoop-lzo</p> <p>Status: Third-party Package Fixed in AWS Hadoop Version: 0.20.205 Fixed in Apache Hadoop Version: n/a</p>
Install the hadoop-snappy library	<p>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/.</p> <p>Status: Third-party Library Fixed in AWS Hadoop Version: 0.20.205 Fixed in Apache Hadoop Version: n/a</p>
MAPREDUCE-1597/2021/2046	<p>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-2021, and https://issues.apache.org/jira/browse/MAPREDUCE-2046.</p> <p>Status: Resolved, Fixed Fixed in AWS Hadoop Version: 0.20.205 Fixed in Apache Hadoop Version: 0.22.0</p>
HADOOP-6436	<p>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.</p> <p>Status: Closed, Fixed Fixed in AWS Hadoop Version: 0.20.205 Fixed in Apache Hadoop Version: 0.22.0,0.23.0</p>
MAPREDUCE-2185	<p>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.</p> <p>Status: Closed, Fixed Fixed in AWS Hadoop Version: 0.20.205 Fixed in Apache Hadoop Version: 0.23.0</p>
HADOOP-7082	<p>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.</p> <p>Status: Resolved, Fixed Fixed in AWS Hadoop Version: 0.20.205 Fixed in Apache Hadoop Version: 0.22.0</p>

Patch	Description
HADOOP-7015	<p>Update RawLocalFileSystem#listStatus 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 https://issues.apache.org/jira/browse/HADOOP-7015.</p> <p>Status: Closed, Fixed Fixed in AWS Hadoop Version: 0.20.205 Fixed in Apache Hadoop Version: 0.23.0</p>
HADOOP-4675	<p>Update the Ganglia metrics to be compatible with Ganglia 3.1. For more information about this patch go to https://issues.apache.org/jira/browse/HADOOP-4675.</p> <p>Status: Resolved, Fixed Fixed in AWS Hadoop Version: 0.20.205 Fixed in Apache Hadoop Version: 0.22.0</p>

Supported Mahout Versions

Amazon EMR currently supports the following Apache Mahout versions:

Mahout Version	AMI Version	Mahout Version Details
0.9	3.1.0 and later	<ul style="list-style-type: none"> • 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 Cluster-Dumper (MAHOUT-1343) • Enabled randomised testing for all Mahout modules using Carrot RandomizedRunner (MAHOUT-1345) • Online Algorithm for computing accurate Quantiles using 1-dimensional Clustering (MAHOUT-1361) • Upgrade to Lucene 4.6.1 (MAHOUT-1364)

Mahout Version	AMI Version	Mahout Version Details
0.8	3.0-3.0.4	
0.8	2.2 and later (with bootstrap action installation)	

For more information on Mahout releases, see: <https://mahout.apache.org>.

(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 \(p. 124\)](#)
- [Use Predefined Bootstrap Actions \(p. 125\)](#)
- [Use Custom Bootstrap Actions \(p. 132\)](#)

Bootstrap Action Basics

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

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

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

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

AWS CLI

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

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

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. 125)
- [Configure Hadoop Bootstrap Action](#) (p. 127)
- [Run If Bootstrap Action](#) (p. 131)
- [S3Get Bootstrap Action](#) (p. 131)
- [Shutdown Actions](#) (p. 132)

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\"`

Configuration Parameter	Description
<code>--<i>daemon</i>-heap-size</code>	Sets the heap size in megabytes for the specified daemon. Note <code>--client-heap-size</code> has no effect. Instead, change the client heap size using the <code>--client-opts=\"-Xmx#####\"</code> equivalent, where <code>#####</code> is numeric.
<code>--<i>daemon</i>-opts</code>	Sets additional Java options for the specified daemon.
<code>--replace</code>	Replaces the existing <code>hadoop-user-env.sh</code> file if it exists.

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:

Amazon Elastic MapReduce Developer Guide (Optional) Create Bootstrap Actions to Install Additional Software

```
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:GCTimeRa  
tio=19"]
```

- Windows users:

```
aws emr create-cluster --name "Test cluster" --ami-version 3.3 --use-de  
fault-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=["--na  
menode-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>.

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:

```
[  
  {  
    "Path": "s3://elasticmapreduce/bootstrap-actions/configure-daemons",  
    "Args": ["--namenode-heap-size=2048","--namenode-opts=-XX:GCTimeRa  
tio=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:

```
./elastic-mapreduce --create --alive \  
--bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-
```

Amazon Elastic MapReduce Developer Guide
(Optional) Create Bootstrap Actions to Install Additional Software

```
daemons \  
  --args --namenode-heap-size=2048,--namenode-opts=-XX:GCTimeRatio=19
```

- Windows:

```
ruby elastic-mapreduce --create --alive --bootstrap-action s3://elasticmapre  
duce/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.

Configuration File Name	Configuration File Keyword	File Name Short-cut	Key-value Pair Shortcut
log4j.properties	log4j	L	l
core-site.xml	core	C	c
hadoop-default.xml (deprecated)	default	D	d
hadoop-site.xml (deprecated)	site	S	s
hdfs-site.xml	hdfs	H	h
mapred-site.xml	mapred	M	m
yarn-site.xml	yarn	Y	y
httpsfs-site.xml	httpsfs	T	t
emrfs-site.xml	emrfs	E	e

Amazon Elastic MapReduce Developer Guide
(Optional) Create Bootstrap Actions to Install Additional Software

Configuration File Name	Configuration File Keyword	File Name Short-cut	Key
capacity-scheduler.xml	capacity	Z	z

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>.

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:

```
{
  "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.

```
{
  "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:

Amazon Elastic MapReduce Developer Guide (Optional) Create Bootstrap Actions to Install Additional Software

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<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:

```
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.tasktrack
er.map.tasks.maximum=2" ]
```

- Windows users:

```
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-ha
doop,Args=[ "-M", "s3://myawsbucket/config.xml", "-m", "mapred.tasktrack
er.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.

Amazon Elastic MapReduce Developer Guide (Optional) Create Bootstrap Actions to Install Additional Software

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:

```
[
  {
    "Path": "s3://elasticmapreduce/bootstrap-actions/configure-hadoop",
    "Args": [ "-M", "s3://myawsbucket/config.xml", "-m", "mapred.tasktrack
er.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>.

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.
- Linux, UNIX, and Mac OS X:

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

- Windows:

```
ruby elastic-mapreduce --create --bootstrap-action s3://elasticmapre  
duce/bootstrap-actions/configure-hadoop --args "-M,s3://myawsbucket/con  
fig.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:

```
aws emr create-cluster --name "Test cluster" --ami-version 3.3.2 \  
--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"]
```

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

```
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/run-  
if,Args=["instance.isMaster=true","echo running on master node"]
```

- Windows users:

```
aws emr create-cluster --name "Test cluster" --ami-version 3.3 --use-de  
fault-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/run-if,Args=["instance.is  
Master=true","echo running on master node"]
```

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

Note

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

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

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.

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 \(p. 132\)](#)
- [Add Custom Bootstrap Actions Using the Console \(p. 134\)](#)

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

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

The sample script looks like the following:

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

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

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

- To launch a cluster with a custom bootstrap action, type the following command, replace `myKey` with the name of your EC2 key pair.
- Linux, UNIX, and Mac OS X users:

```
aws emr create-cluster --name "Test cluster" --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:

```
aws emr create-cluster --name "Test cluster" --ami-version 3.3 --use-de  
fault-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:

```
./elastic-mapreduce --create --alive --bootstrap-action "s3://elasticmapre  
duce/bootstrap-actions/download.sh"
```

- Windows:

```
ruby elastic-mapreduce --create --alive --bootstrap-action "s3://elast  
icmapreduce/bootstrap-actions/download.sh"
```

Add Custom Bootstrap Actions Using the Console

The following procedure creates a predefined word count sample cluster with a bootstrap action script that downloads and extracts a compressed TAR archive from Amazon S3. The sample script is stored at <http://elasticmapreduce.s3.amazonaws.com/bootstrap-actions/download.sh>.

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. On the **Advanced cluster configuration** page, click **Configure sample application**.
5. In the **Configure Sample Application** page, in the **Select sample application** field, choose the **Word count** sample application from the list.
6. In the **Output location** field, type the path of an Amazon S3 bucket to store your output and then click **Ok**.
7. On the **Create Cluster** page, in the **Cluster Configuration** section, verify the fields according to the following table.

Field	Action
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. 462) . 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. 413) .
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. 413) .
Debugging	This option creates a debug log index in 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 about Amazon SimpleDB, go to the Amazon SimpleDB product description page.

Amazon Elastic MapReduce Developer Guide
(Optional) Create Bootstrap Actions to Install Additional Software

8. In the **Software Configuration** section, verify the fields according to the following table.

Field	Action
Hadoop distribution	<p>Choose Amazon.</p> <p>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. 218).</p>
AMI version	<p>Choose the latest Hadoop 2.x AMI or the latest Hadoop 1.x AMI from the list.</p> <p>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. 48).</p>

9. 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, go to the [Request to Increase Amazon EC2 Instance Limit Form](#).

Field	Action
Network	<p>Choose the default VPC. For more information about the default VPC, see Your Default VPC and Subnets in the <i>guide-vpc-user</i>.</p> <p>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 Select an Amazon VPC Subnet for the Cluster (Optional) (p. 205).</p>
EC2 Availability Zone	<p>Choose No preference.</p> <p>Optionally, you can launch the cluster in a specific EC2 Availability Zone.</p> <p>For more information, see Regions and Availability Zones in the <i>Amazon EC2 User Guide for Linux Instances</i>.</p>
Master	<p>Accept the default instance type.</p> <p>The master node assigns Hadoop tasks to core and task nodes, and monitors their status. There is always one master node in each cluster.</p> <p>This specifies the EC2 instance type to use for the master node.</p> <p>The default instance type is m1.medium for Hadoop 2.x. This instance type is suitable for testing, development, and light workloads.</p> <p>For more information on instance types supported by Amazon EMR, see Virtual Server Configurations. For more information on Amazon EMR instance groups, see Instance Groups (p. 34). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 471).</p>

Amazon Elastic MapReduce Developer Guide
(Optional) Create Bootstrap Actions to Install Additional Software

Field	Action
Request Spot Instances	<p>Leave this box unchecked.</p> <p>This specifies whether to run master nodes on Spot Instances. For more information, see (Optional) Lower Costs with Spot Instances (p. 37).</p>
Core	<p>Accept the default instance type.</p> <p>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.</p> <p>This specifies the EC2 instance types to use as core nodes.</p> <p>The default instance type is m1.medium for Hadoop 2.x. This instance type is suitable for testing, development, and light workloads.</p> <p>For more information on instance types supported by Amazon EMR, see Virtual Server Configurations. For more information on Amazon EMR instance groups, see Instance Groups (p. 34). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 471).</p>
Count	Choose 2 .
Request Spot Instances	<p>Leave this box unchecked.</p> <p>This specifies whether to run core nodes on Spot Instances. For more information, see (Optional) Lower Costs with Spot Instances (p. 37).</p>
Task	<p>Accept the default instance type.</p> <p>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.</p> <p>This specifies the EC2 instance types to use as task nodes.</p> <p>For more information on instance types supported by Amazon EMR, see Virtual Server Configurations. For more information on Amazon EMR instance groups, see Instance Groups (p. 34). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 471).</p>
Count	Choose 0 .
Request Spot Instances	<p>Leave this box unchecked.</p> <p>This specifies whether to run task nodes on Spot Instances. For more information, see (Optional) Lower Costs with Spot Instances (p. 37).</p>

10. In the **Security and Access** section, complete the fields according to the following table.

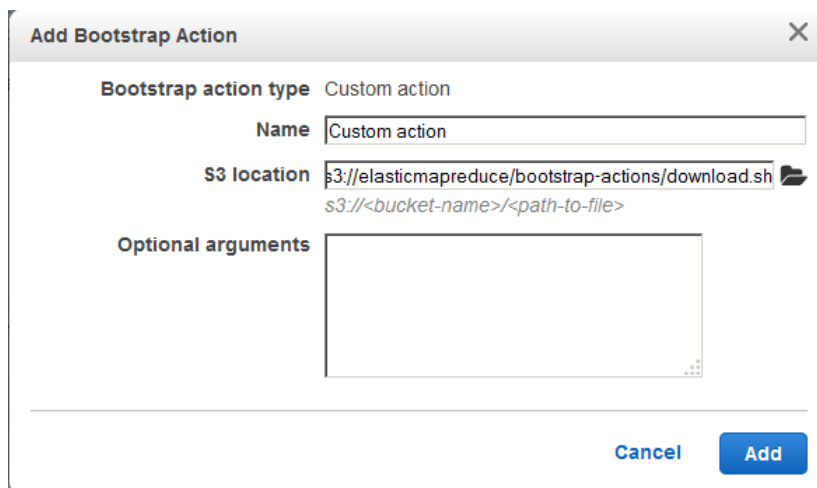
Amazon Elastic MapReduce Developer Guide
(Optional) Create Bootstrap Actions to Install Additional Software

Field	Action
EC2 key pair	<p>Choose your Amazon EC2 key pair private key from the list.</p> <p>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. 441) .</p>
IAM user access	<p>Choose All other IAM users to make the cluster visible and accessible to all IAM users on the AWS account. For more information, see Configure IAM User Permissions (p. 181) .</p> <p>Alternatively, choose No other IAM users to restrict access to the current IAM user.</p>
Roles configuration	<p>Choose Default to generate the default EMR role and 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 EMR role and EC2 instance profile are required when creating a cluster using the console.</p> <p>The EMR role allows Amazon EMR to access other AWS services on your behalf. The EC2 instance profile controls application access to the Amazon EC2 instances in the cluster.</p> <p>For more information, see Configure IAM Roles for Amazon EMR (p. 185) .</p>

11. In the **Bootstrap Actions** section, in the **Add bootstrap action** field, select **Custom action** and then click **Configure and add**.
12. In the **Add Bootstrap Action** dialog box, do the following:
 - a. Enter the following text in the **S3 location** field:

```
s3://elasticmapreduce/bootstrap-actions/download.sh
```

- b. (Optional) Enter any arguments in the **Optional arguments** field. Use spaces to separate the arguments.
 - c. Click **Add**.



The screenshot shows a dialog box titled "Add Bootstrap Action". It contains the following fields:

- Bootstrap action type:** Custom action
- Name:** Custom action
- S3 location:** s3://elasticmapreduce/bootstrap-actions/download.sh (with a file icon) and a placeholder `s3://<bucket-name>/<path-to-file>`
- Optional arguments:** An empty text area.

At the bottom right, there are "Cancel" and "Add" buttons.

13. In the **Bootstrap Actions** section, note the properties of the custom bootstrap action.
14. Review your configuration and if you are satisfied with the settings, choose **Create Cluster**.
15. When the cluster starts, the console displays the **Cluster Details** page.

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 \(p. 413\)](#)

File Systems Compatible with Amazon EMR

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.

File System	Prefix	Description
HDFS	<code>hdfs://</code> (or no prefix)	<p>HDFS is a distributed, scalable, and portable file system for Hadoop. An advantage of HDFS is data awareness between the Hadoop cluster nodes managing the clusters and the Hadoop cluster nodes managing the individual steps. For more information about how HDFS works, go to the Hadoop documentation.</p> <p>HDFS is used by the master and core nodes. One advantage is that it's fast; a disadvantage is that it's ephemeral storage which is reclaimed when the cluster ends. It's best used for caching the results produced by intermediate job-flow steps.</p>
EMRFS	<code>s3://</code>	<p>EMRFS is an implementation of HDFS used for reading and writing regular files from Amazon EMR directly to Amazon S3. EMRFS provides the convenience of storing persistent data in Amazon S3 for use with Hadoop while also providing features like Amazon S3 server-side encryption, read-after-write consistency, and list consistency.</p> <p>Note Previously, Amazon EMR used the S3 Native FileSystem with the URI scheme, <code>s3n</code>. While this still works, we recommend that you use the <code>s3</code> URI scheme for the best performance, security, and reliability.</p>
local file system		<p>The local file system refers to a locally connected disk. When a Hadoop cluster is created, each node is created from an EC2 instance that comes with a preconfigured block of preattached disk storage called an <i>instance store</i>. Data on instance store volumes persists only during the life of its EC2 instance. Instance store volumes are ideal for storing temporary data that is continually changing, such as buffers, caches, scratch data, and other temporary content. For more information, see Amazon EC2 Instance Storage.</p>
(Legacy) Amazon S3 block file system	<code>s3bfs://</code>	<p>The Amazon S3 block file system is a legacy file storage system. We strongly discourage the use of this system.</p> <p>Important We recommend that you do not use this file system because it can trigger a race condition that might cause your cluster to fail. However, it might be required by legacy applications.</p>

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.

Caution

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

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

EMR File System (EMRFS) (Optional)

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

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

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

Consistent View

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

Amazon S3 is designed for eventual consistency. For instance, buckets in the US East (N. Virginia) provide eventual consistency on read-after-write and read-after-rewrite requests. Amazon S3 buckets in the US West (Oregon), US West (N. California), EU (Ireland), EU (Frankfurt), Asia Pacific (Singapore), Asia Pacific (Tokyo), Asia Pacific (Sydney), and South America (Sao Paulo) regions provide read-after-write consistency for put requests of new objects and eventual consistency for overwrite put and delete requests. Therefore, if you are listing objects in an Amazon S3 bucket quickly after putting new objects, Amazon S3 does not provide a guarantee to return a consistent listing and it may be incomplete. This is more common in quick sequential MapReduce jobs which use Amazon S3 as a data store.

EMRFS includes a command line utility on the master node, `emrfs`, which allows administrator to perform operations on metadata such as import, delete, and sync. For more information about the EMRFS CLI, see [the section called "EMRFS CLI Reference" \(p. 149\)](#).

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

Note

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

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

Topics

- [How to Enable Consistent View](#) (p. 142)
- [Objects Tracked By EMRFS](#) (p. 143)
- [Retry Logic](#) (p. 143)
- [EMRFS Metadata](#) (p. 144)
- [Configuring Consistency Notifications for CloudWatch and Amazon SQS](#) (p. 146)
- [Configuring Consistent View](#) (p. 147)
- [EMRFS CLI Reference](#) (p. 149)

How to Enable Consistent View

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

Note

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

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

Note

Subsequent retries use an exponential backoff.

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

Note

You will need to install the current version of AWS CLI. To download the latest release, see <http://aws.amazon.com/cli/>.

Type the following command to launch an Amazon EMR cluster with consistent view enabled.

```
aws emr create-cluster --instance-type m1.large --instance-count 3 --emrfs  
Consistent=true --ami-version=3.2.1 --ec2-attributes KeyName=myKey
```

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

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

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

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

Objects Tracked By EMRFS

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

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

Objects read by EMRFS are not automatically added to the metadata. When a object is deleted by EMRFS, a listing still remains in the metadata with a deleted state until that listing is purged using the EMRFS CLI. To learn more about the CLI, see [the section called “EMRFS CLI Reference” \(p. 149\)](#). For more information about purging listings in the EMRFS metadata, see [the section called “EMRFS Metadata” \(p. 144\)](#).

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

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

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

Retry Logic

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

Setting retry period to a fixed amount

```
aws emr create-cluster --ami-version 3.2.1 --instance-type m1.large --instance-count 1 \  
--emrfs Consistent=true,Args=[fs.s3.consistent.throwExceptionOnInconsistency=false,\  
fs.s3.consistent.retryPolicyType=fixed,fs.s3.consistent.retryPeriodSeconds=10] \  
--ec2-attributes KeyName=myKey
```


For more information, see [the section called “Configuring Consistent View”](#) (p. ?).

EMRFS Metadata

Note

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

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

How to add entries to metadata

You can use the `sync` or `import` subcommands to add entries to metadata. `sync` will simply reflect the state of the Amazon S3 objects in a path while `import` is used strictly to add new entries to the metadata. For more information, see [the section called “EMRFS CLI Reference”](#) (p. 149).

How to check differences between metadata and objects in Amazon S3

To check for differences between the metadata and Amazon S3, use the `diff` subcommand of the EMRFS CLI. For more information, see [the section called “EMRFS CLI Reference”](#) (p. 149).

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 500 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 500 and 100 throughput capacity units, respectively. The following performance characteristics will give you an idea of what throughput is required for certain operations. These tests were performed using a single-node `m3.large` cluster. All operations were single threaded. Performance will differ greatly based on particular application characteristics and it may take experimentation to optimize file system operations.

Operation	Average read-per-second	Average write-per-second
<code>create</code> (object)	26.79	6.70
<code>delete</code> (object)	10.79	10.79

Operation	Average read-per-second	Average write-per-second
delete (directory containing 1000 objects)	21.79	338.40
getFileStatus (object)	34.70	0
getFileStatus (directory)	19.96	0
listStatus (directory containing 1 object)	43.31	0
listStatus (directory containing 10 objects)	44.34	0
listStatus (directory containing 100 objects)	84.44	0
listStatus (directory containing 1,000 objects)	308.81	0
listStatus (directory containing 10,000 objects)	416.05	0
listStatus (directory containing 100,000 objects)	823.56	0
listStatus (directory containing 1M objects)	882.36	0
mkdir (continuous for 120 seconds)	24.18	4.03
mkdir	12.59	0
rename (object)	19.53	4.88
rename (directory containing 1000 objects)	23.22	339.34

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

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

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

```
aws emr add-steps --cluster-id j-2AL4XXXXXX5T9 --steps 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.

Configuring Consistency Notifications for CloudWatch and Amazon SQS

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

CloudWatch

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

To view CloudWatch metrics for Amazon S3 eventual consistency issues

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

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

Amazon SQS

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

To manage Amazon SQS messages for Amazon S3 eventual consistency issues

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

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

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

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

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

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

Note

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

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

```
emrfs delete-sqs -queue-name queuename
```

For example, to delete the default queue, type:

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

Note

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

Configuring Consistent View

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

Changing default metadata read and write values at cluster launch

```
aws emr create-cluster --ami-version 3.2.1 --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 3.2.1 --instance-type m1.large \  
--bootstrap-actions Path=s3://us-east-1.elasticmapreduce/bootstrap-actions/con  
figure-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:

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

Property	Default value	Description
<code>fs.s3.consistent</code>	<code>false</code>	When set to <code>true</code> , this property configures EMRFS to use DynamoDB to provide consistency.
<code>fs.s3.consistent.retryPolicyType</code>	<code>exponential</code>	This property identifies the policy to use when retrying for consistency issues. Options include: <code>exponential</code> , <code>fixed</code> , or <code>none</code> .
<code>fs.s3.consistent.retryPeriodSeconds</code>	<code>10</code>	This property sets the length of time to wait between consistency retry attempts.
<code>fs.s3.consistent.retryCount</code>	<code>5</code>	This property sets the maximum number of retries when inconsistency is detected.

Amazon Elastic MapReduce Developer Guide
EMR File System (EMRFS) (Optional)

Property	Default value	Description
<code>fs.s3.consistent.throwExceptionOnInconsistency</code>	true	This property determines whether to throw or log a consistency exception. When set to true , a <code>ConsistencyException</code> is thrown.
<code>fs.s3.consistent.metadata.autoCreate</code>	true	When set to true , this property enables automatic creation of metadata tables.
<code>fs.s3.consistent.metadata.tableName</code>	EmrFS-Metadata	This property specifies the name of the metadata table in DynamoDB.
<code>fs.s3.consistent.metadata.read.capacity</code>	500	This property specifies the DynamoDB read capacity to provision when the metadata table is created.
<code>fs.s3.consistent.metadata.write.capacity</code>	250	This property specifies the DynamoDB write capacity to provision when the metadata table is created.
<code>fs.s3.consistent.fastList</code>	true	When set to true , this property uses multiple threads to list a directory (when necessary). Consistency must be enabled in order to use this property.
<code>fs.s3.consistent.fastList.prefetchMetadata</code>	false	When set to true , this property enables metadata prefetching for directories containing more than 20,000 items.
<code>fs.s3.consistent.notification.CloudWatch</code>	false	When set to true , CloudWatch metrics are enabled for FileSystem API calls that fail due to Amazon S3 eventual consistency issues.
<code>fs.s3.consistent.notification.SQS</code>	false	When set to true , eventual consistency notifications are pushed to an Amazon SQS queue.
<code>fs.s3.consistent.notification.SQS.queueName</code>	EMRFS-Inconsistency- <jobFlowId>	Changing this property allows you to specify your own SQS queue name for messages regarding Amazon S3 eventual consistency issues.
<code>fs.s3.consistent.notification.SQS.customMsg</code>	none	This property allows you to specify custom information included in SQS messages regarding Amazon S3 eventual consistency issues. If a value is not specified for this property, the corresponding field in the message is empty.

EMRFS CLI Reference

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

Note

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

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]]
```

emrfs command

The `emrfs` command accepts the following `[[options]]` (with or without arguments).

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

describe-metadata sub-command

The `describe-metadata` sub-command accepts the following `[[options]]` (with or without arguments).

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

describe-metadata example

The following example describes the default metadata table.

```
$ emrfs describe-metadata
EmrFSMetadata
```

```
read-capacity: 500
write-capacity: 100
status: ACTIVE
approximate-item-count (6 hour delay): 12
```

set-metadata-capacity sub-command

The **set-metadata-capacity** sub-command accepts the following [[options]] (with or without arguments).

Option	Description	Re-quired
-m <i>METADATA_NAME</i> - -metadata-name <i>METADATA_NAME</i>	<i>METADATA_NAME</i> is the name of the DynamoDB metadata table. If the <i>METADATA_NAME</i> argument is not supplied, the default value is <code>EmrFSMetadata</code> .	No
-r <i>READ_CAPACITY</i> --read- capacity <i>READ_CAPACITY</i>	The requested read throughput capacity for the metadata table. If the <i>READ_CAPACITY</i> argument is not supplied, the default value is 500.	No
-w <i>WRITE_CAPACITY</i> - -write-capacity <i>WRITE_CAPA- CITY</i>	The requested write throughput capacity for the metadata table. If the <i>WRITE_CAPACITY</i> argument is not supplied, the default value is 100.	No

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: 500
write-capacity: 100
status: UPDATING
approximate-item-count (6 hour delay): 0
```

delete-metadata sub-command

The **delete-metadata** sub-command accepts the following [[options]] (with or without arguments).

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

delete-metadata example

The following example deletes the default metadata table.

```
$ emrfs delete-metadata
[no output]
```

create-metadata sub-command

The **create-metadata** sub-command accepts the following [[options]] (with or without arguments).

Option	Description	Re-quired
-m <i>METADATA_NAME</i> - -metadata-name <i>METADATA_NAME</i>	<i>METADATA_NAME</i> is the name of the DynamoDB metadata table. If the <i>METADATA_NAME</i> argument is not supplied, the default value is EmrFSMetadata.	No
-r <i>READ_CAPACITY</i> --read- capacity <i>READ_CAPACITY</i>	The requested read throughput capacity for the metadata table. If the <i>READ_CAPACITY</i> argument is not supplied, the default value is 500.	No
-w <i>WRITE_CAPACITY</i> - -write-capacity <i>WRITE_CAPA- CITY</i>	The requested write throughput capacity for the metadata table. If the <i>WRITE_CAPACITY</i> argument is not supplied, the default value is 100.	No

create-metadata example

The following example creates a metadata table named EmrFSMetadataAlt.

```
$ emrfs create-metadata -m EmrFSMetadataAlt
Creating metadata: EmrFSMetadataAlt
EmrFSMetadataAlt
  read-capacity: 500
  write-capacity: 100
  status: ACTIVE
  approximate-item-count (6 hour delay): 0
```

list-metadata-stores sub-command

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

list-metadata-stores example

The following example lists your metadata tables.

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

diff sub-command

The **diff** sub-command accepts the following [[options]] (with or without arguments).

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

diff example

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

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

delete sub-command

The **delete** sub-command accepts the following [[options]] (with or without arguments).

Option	Description	Required
-m <i>METADATA_NAME</i> - -metadata-name <i>METADATA_NAME</i>	<i>METADATA_NAME</i> is the name of the DynamoDB metadata table. If the <i>METADATA_NAME</i> argument is not supplied, the default value is <code>EmrFSMetadata</code> .	No
[s3:// <i>s3Path</i>]	The path to the Amazon S3 bucket you are tracking for consistent view. Buckets sync recursively.	Yes
-t <i>TIME</i> --time <i>TIME</i>	The expiration time (interpreted using the time unit argument). All metadata entries older than the <i>TIME</i> argument are deleted for the specified bucket.	
-u <i>UNIT</i> --time-unit <i>UNIT</i>	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 <code>days</code> .	
--read-consumption <i>READ_CONSUMPTION</i>	The requested amount of available read throughput used for the delete operation. If the <i>READ_CONSUMPTION</i> argument is not specified, the default value is 500.	No
--write-consumption <i>WRITE_CONSUMPTION</i>	The requested amount of available write throughput used for the delete operation. If the <i>WRITE_CONSUMPTION</i> argument is not specified, the default value is 100.	No

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
```

import sub-command

The **import** sub-command accepts the following [[options]] (with or without arguments).

Option	Description	Re-quired
-m <i>METADATA_NAME</i> - -metadata-name <i>METADATA_NAME</i>	<i>METADATA_NAME</i> is the name of the DynamoDB metadata table. If the <i>METADATA_NAME</i> argument is not supplied, the default value is <code>EmrFSMetadata</code> .	No
[s3:// <i>s3Path</i>]	The path to the Amazon S3 bucket you are tracking for consistent view. Buckets sync recursively.	Yes
--read-consumption <i>READ_CONSUMPTION</i>	The requested amount of available read throughput used for the delete operation. If the <i>READ_CONSUMPTION</i> argument is not specified, the default value is 500.	No
--write-consumption <i>WRITE_CONSUMPTION</i>	The requested amount of available write throughput used for the delete operation. If the <i>WRITE_CONSUMPTION</i> argument is not specified, the default value is 100.	No

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
```

sync sub-command

The **sync** sub-command accepts the following [[options]] (with or without arguments).

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

Option	Description	Re-quired
<code>--read-consumption</code> <i>READ_CONSUMPTION</i>	The requested amount of available read throughput used for the delete operation. If the <i>READ_CONSUMPTION</i> argument is not specified, the default value is 500.	No
<code>--write-consumption</code> <i>WRITE_CONSUMPTION</i>	The requested amount of available write throughput used for the delete operation. If the <i>WRITE_CONSUMPTION</i> argument is not specified, the default value is 100.	No

sync example

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

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

read-sqs sub-command

The **read-sqs** sub-command accepts the following [[options]] (with or without arguments).

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

delete-sqs sub-command

The **delete-sqs** sub-command accepts the following [[options]] (with or without arguments).

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

Submitting EMRFS CLI Commands as Steps

To add an Amazon S3 bucket to the tracking metadata for consistent view (AWS SDK for Python)

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

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

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

myStep = JarStep(name='Boto EMRFS Sync',
                 jar='s3://elasticmapreduce/libs/script-runner/script-runner.jar',

                 action_on_failure="CONTINUE",
                 step_args=['/home/hadoop/bin/emrfs',
                             'sync',
                             's3://elasticmapreduce/samples/cloudfront'])

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

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

Creating an AWSCredentialsProvider for EMRFS

You can create a custom credentials provider which implements both the [AWSCredentialsProvider](#) and the Hadoop [Configurable](#) classes for use with EMRFS when it makes calls to Amazon S3. You must specify the full class name of the provider by setting `fs.s3.customAWSCredentialsProvider` in `/home/hadoop/conf/emrfs-site.xml`. You set this property at cluster creation time using the AWS CLI. For example, the following code sets `fs.s3.customAWSCredentialsProvider` to `MyAWSCredentialsProvider`.

```
aws emr create-cluster --ami-version 3.4 --instance-type m1.large \
--bootstrap-actions Path=s3://us-east-1.elasticmapreduce/bootstrap-actions/con
figure-hadoop, \
Args=[-e,fs.s3.customAWSCredentialsProvider=MyAWSCredentialsProvider] --ec2-
attributes KeyName=myKey
```

Use the configuration you created with the following syntax:

Additionally, you will need to place the JAR file of the `AWSCredentialsProvider` class in `/usr/share/aws/emr/auxlib`. An example implementation follows:

```
public class MyAWSCredentialsProvider implements AWSCredentialsProvider, Configurable {
    private Configuration conf;
    private String accessKey;
    private String secretKey;

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

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

    @Override
    public void refresh() {

    }

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

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

Encryption in EMRFS

You can use either server-side or client-side encryption to protect the data you store in Amazon S3. With server-side encryption, Amazon S3 encrypts your data after you upload it. With client-side encryption, you manage the encryption and keys. You can use AWS Key Management Service (AWS KMS) to manage your keys used for encryption. You enable both of these options at cluster creation time.

Note

Client-side encryption and server-side encryption are mutually exclusive global settings. When either option is enabled, all Amazon S3 write actions that happen through EMRFS use the form of encryption chosen.

EMRFS implements Amazon S3 encryption. If you write files locally on your cluster (in HDFS or local file systems volumes), those files are not encrypted.

Topics

- [Create a Cluster With Amazon S3 Server-Side Encryption Enabled \(p. 156\)](#)
- [Using Amazon S3 Client-Side Encryption in EMRFS \(p. 157\)](#)

Create a Cluster With Amazon S3 Server-Side Encryption Enabled

Amazon S3 server-side encryption (SSE) is supported with Amazon EMR on AMIs 3.2.1 or later. To launch a cluster with server-side encryption, you can use the AWS Management Console, AWS CLI, or the `configure-hadoop bootstrap` action to set `fs.s3.enableServerSideEncryption` to `"true"`. You

can specify the encryption key and/or algorithm used. However, if you provide your own key, you must have access to that same key to later access objects stored and encrypted.

To configure server-side encryption using the console

1. Choose **Create Cluster**.
2. Navigate to the **File System Configuration** section.
3. To use **Server-side encryption**, choose **Enabled**.
4. Choose **Create cluster**.

To launch a cluster with Amazon S3 server-side encryption enabled

Type the following command to launch an Amazon EMR cluster with Amazon S3 server-side encryption enabled.

```
aws emr create-cluster --ami-version 3.8 --instance-count 3 --instance-type m1.large --emrfs Encryption=ServerSide
```

To launch a cluster with Amazon S3 server side encryption enabled that specifies an encryption algorithm

Type the following command to launch an Amazon EMR cluster with Amazon S3 server-side encryption enabled that specifies AES256 as the encryption algorithm.

```
aws emr create-cluster --instance-type m3.xlarge --ami-version 3.8 --emrfs Encryption=ServerSide,Args=[fs.s3.serverSideEncryptionAlgorithm=AES256]
```

emrfs-site.xml properties for server-side encryption

Property	Default value	Description
<code>fs.s3.enableServerSideEncryption</code>	<code>false</code>	When set to <code>true</code> , objects stored in Amazon S3 are encrypted using server-side encryption.
<code>fs.s3.serverSideEncryptionAlgorithm</code>	<code>AES256</code>	When using server-side encryption, this property determines the algorithm used to encrypt data.

Using Amazon S3 Client-Side Encryption in EMRFS

EMRFS support for Amazon S3 client-side encryption enables your EMR cluster to work with S3 objects that were previously encrypted using an Amazon S3 encryption client. When Amazon S3 client-side encryption is enabled, EMRFS supports the decryption of objects encrypted using keys in AWS KMS or from your own key management system. Amazon S3 client-side encryption in EMRFS also supports re-encrypting the output from your EMR cluster using keys from either AWS KMS or your own key management system.

Note

EMRFS client-side encryption only ensures that output written from an enabled cluster to Amazon S3 will be encrypted. Data written to the local file systems and HDFS on the cluster are not encrypted. Furthermore, because Hue does not use EMRFS, objects written to Amazon S3 using the Hue S3 File Browser are not encrypted. For more information about security controls available for applications running on EC2 instances, see the [“Overview of Security Processes” whitepaper](#).

EMRFS support for Amazon S3 client-side encryption uses a process called *envelope encryption*, with keys stored in a location of your choosing, to encrypt and decrypt data stored in Amazon S3. In contrast to Amazon S3 server-side encryption, the decryption and encryption actions in Amazon S3 client-side encryption take place in the EMRFS client on your EMR cluster; the encrypted object streams from Amazon S3 to your EMR cluster in an encrypted form to be decrypted by the client on the cluster. Output from the cluster is then encrypted by the client before being written to Amazon S3.

The envelope encryption process uses a one-time symmetric data key generated by the encryption client, unique to each object, to encrypt data. The data key is then encrypted by your master key (stored in AWS KMS or your custom provider) and stored with the associated object in Amazon S3. When decrypting data on the client (e.g., an EMRFS client or your own Amazon S3 encryption client retrieving data for post-processing), the reverse process occurs: the encrypted data key is retrieved from the metadata of the object in Amazon S3. It is decrypted using the master key and then the client uses the data key to decrypt the object data. When Amazon S3 client-side encryption is enabled, the EMRFS client on the cluster can read either encrypted or unencrypted objects in Amazon S3.

When Amazon S3 client-side encryption in EMRFS is enabled, the behavior of the encryption client depends on the provider specified and the metadata of the object being decrypted or encrypted. When EMRFS encrypts an object before writing it to Amazon S3, the provider (e.g., AWS KMS or your custom provider) that you specified at cluster creation time is always used to supply the encryption key. When EMRFS reads an object from Amazon S3, it checks the object metadata for information about the master key used to encrypt the data key. If there is an AWS KMS key ID, EMRFS attempts to decrypt the object using AWS KMS. If there is metadata containing an `EncryptionMaterialsDescription` instance, EMRFS tries to fetch the key using the `EncryptionMaterialsProvider` instance. The provider uses this description to determine which key should be used and to retrieve it. If you do not have access to the required key, this raises an exception and causes an error. If there is no `EncryptionMaterialsDescription` instance in the Amazon S3 object metadata, EMRFS assumes that the object is unencrypted.

Amazon S3 client-side encryption in EMRFS provides two methods to supply the master keys for decryption when reading from Amazon S3 and encryption when writing to Amazon S3:

1. With a built-in AWS KMS provider, which can use a master key stored in AWS KMS. You specify the key to use for encryption, but EMRFS can use any AWS KMS key for decryption, assuming your cluster has permission to access it. AWS KMS charges apply for the storage and use of encryption keys.
2. With a custom Java class implementing both the Amazon S3 [EncryptionMaterialsProvider](#) and Hadoop [Configurable](#) classes. The `EncryptionMaterialsProvider` class is used to provide the materials description, detailing how and where to get the master keys.

For more information about Amazon S3 client-side encryption see, [Protecting Data Using Client-Side Encryption](#). For more information about how to use the AWS SDK for Java with Amazon S3 client-side encryption, see the article [Client-Side Data Encryption with the AWS SDK for Java and Amazon S3](#).

For information about how to create and manage keys in AWS KMS and associated pricing, see [AWS KMS Frequently Asked Questions](#) and the [AWS Key Management Service Developer Guide](#).

Topics

- [Enabling Amazon S3 Client-Side Encryption in the Console \(p. 158\)](#)
- [Selecting a Master Key Stored in AWS KMS using an SDK or CLI \(p. 159\)](#)
- [Configuring Amazon S3 Client-side Encryption Using a Custom Provider \(p. 159\)](#)
- [emrfs-site.xml Properties for Amazon S3 Client-side Encryption \(p. 162\)](#)

Enabling Amazon S3 Client-Side Encryption in the Console

To configure client-side encryption using the console

1. Choose Create Cluster.

2. Fill in the fields as appropriate for **Cluster Configuration** and **Tags**.
3. For the **Software Configuration** field, choose AMI 3.6.0 or later.
4. In the **File System Configuration** section, select a one of the following client-side encryption types for the **Encryption** field: **S3 client-side encryption with AWS Key Management Service (KMS)** or **S3 client-side encryption with custom encryption materials provider**.
 - a. If you chose **S3 client-side encryption with AWS Key Management Service (KMS)**, select the master key alias from the list of master keys that you have previously configured. Alternately, you can choose **Enter a Key ARN** and enter the ARN of a AWS KMS master key that belongs to a different account, provided that you have permissions to use that key. If you have assigned an instance profile to your EMR cluster, make sure that the role in that profile has permissions to use the key.
 - b. If you chose **S3 client-side encryption with custom encryption materials provider**, provide the full class name and Amazon S3 location of your EncryptionMaterialsProvider class. Amazon EMR automatically downloads your provider to each node in your cluster when it is created.
5. Fill in the fields as appropriate for **Hardware Configuration**, **Security and Access**, **Bootstrap Actions**, and **Steps**.
6. Choose **Create cluster**.

Selecting a Master Key Stored in AWS KMS using an SDK or CLI

When you enable Amazon S3 client-side encryption in EMRFS and specify keys stored in AWS KMS, you provide the KeyId value, key alias, or ARN of the key that Amazon EMR will use to encrypt objects written to Amazon S3. For decryption, EMRFS tries to access whichever key encrypted the object. You create the key using the IAM console, AWS CLI, or the AWS SDKs.

If you have assigned an instance profile to your EMR cluster, make sure that the role in that profile has permission to use the key. AWS KMS charges apply for API calls during each encryption or decryption activity and for storing your key. For more information, see the [AWS KMS pricing page](#).

To use an AWS KMS master key for encryption with EMRFS, provide the master key by reference using any of three possible identifiers:

- KeyId (a 32-character GUID)
- Alias mapped to the KeyId value (you must include the `alias/` prefix in this value)
- Full ARN of the key, which includes the region, account ID, and KeyId value

`MyKMSKeyId` in the example below can be any of the three values:

```
aws emr create-cluster --ami-version 3.8.0 --emrfs Encryption=ClientSide,ProviderType=KMS,KMSKeyId=MyKMSKeyId
```

Note

Note: You must use the ARN of the AWS KMS master key if you want to use a key owned by an account different than the one you are using to configure Amazon EMR.

Configuring Amazon S3 Client-side Encryption Using a Custom Provider

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


```
aws emr create-cluster --instance-type m3.xlarge --ami-version 3.8.0 --emrfs \
Encryption=ClientSide,ProviderType=Custom,CustomProviderLocation=s3://mybuck
et/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.

Custom EncryptionMaterialsProvider with Arguments

You may need to pass arguments directly to the provider, so you can use the `configure-hadoop` bootstrap action to supply arguments using `emrfs-site.xml`:

```
aws emr create-cluster --ami-version 3.8.0 --instance-type m3.xlarge --instance-
count 2 \
--emrfs Encryption=ClientSide,CustomProviderLocation=s3://mybucket/myfolder/mypro
vider.jar,CustomProviderClass=classname \
--bootstrap-action Path=s3://elasticmapreduce/bootstrap-actions/configure-ha
doop,Args=[-e,myProvider.arg1=value1,-e,myProvider.arg2=value2]
```

Then use the configuration with the CLI:

To use an SDK, you can first use the `s3get` bootstrap action to download the custom `EncryptionMaterialsProvider` class 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:

```
<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.encryptedMateri
alsProvider=full.class.name.of.EncryptionMaterialsProvider", "-e", "-e,myPro
vider.arg1=value1");

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

    RunJobFlowRequest request = new RunJobFlowRequest()
        .withName("Client-side enabled EMRFS")
        .withAmiVersion("3.8.0")
        .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>
```

For more information about a list of configuration key values to use to configure `emrfs-site.xml`, see [the section called “emrfs-site.xml Properties for Amazon S3 Client-side Encryption” \(p. ?\)](#).

Reference Implementation of Amazon S3 EncryptionMaterialsProvider

When fetching 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. You can enable this behavior by setting `fs.s3.cse.materialsDescription.enabled` to `true` in `emrfs-site.xml`. For example, the provider may return different keys for different Amazon S3 URI prefixes. Note that 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.

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

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

import java.util.Map;

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

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

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

    @Override
```

```

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

@Override
public void refresh() {

}

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

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

```

emrfs-site.xml Properties for Amazon S3 Client-side Encryption

Property	Default value	Description
fs.s3.cse.enabled	false	When set to true , objects stored in Amazon S3 are encrypted using client-side encryption.
fs.s3.cse.encryptionMaterialsProvider	N/A	The EncryptionMaterialsProvider class path used with client-side encryption.
fs.s3.cse.materialsDescription.enabled	false	Enabling will populate the materialsDescription of encrypted objects with the Amazon S3 URI for the object and the JobFlowId.
fs.s3.cse.kms.keyId	N/A	The value of the KeyId field for the AWS KMS encryption key that you are using with EMRFS encryption. Note This property also accepts the ARN and key alias associated with the key.
fs.s3.cse.cryptoStorageMode	Object-Metadata	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 Client-Side Data Encryption with the AWS SDK for Java and Amazon S3 .

Choose the Cluster Lifecycle: Long-Running or Transient

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 is performing 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.

If you want your cluster to be long-running, you must disable auto-termination when you launch the cluster. You can do this when you launch a cluster using the console, the AWS CLI, or programmatically. 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. 462).

To launch a long-running cluster using the console

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Choose **Create cluster**.
3. Choose **Go to advanced options**.
4. In the **Steps** section, in the **Auto-terminate** field, choose **No**, which runs the cluster until you terminate it.

Remember to terminate the cluster when it is done so you do not continue to accrue charges on an idle cluster.

5. Proceed with creating the cluster as described in [Plan an Amazon EMR Cluster](#) (p. 30).

To launch a long-running cluster using the AWS CLI

By default, clusters created using the AWS CLI are long-running. If you wish, you may optionally specify the `--no-auto-terminate` parameter when you use the `create-cluster` subcommand.

- To launch a long-running cluster using the `--no-auto-terminate` parameter, 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 3.8 --applications Name=Hue Name=Hive Name=Pig \  
--use-default-roles --ec2-attributes KeyName=myKey \  
--instance-type m3.xlarge --instance-count 3 --no-auto-terminate
```

- Windows users:

```
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 --no-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 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>.

Prepare Input Data (Optional)

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. 164)
- [How to Get Data Into Amazon EMR](#) (p. 165)

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. This is the input format most commonly used.

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

If you 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. 217).

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

How to Get Data Into Amazon EMR

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 (version 0.7.1.1 and later) includes functionality that you can use to import and export data between DynamoDB and an Amazon EMR cluster. If you have large amounts of on-premise data to process, you may find the AWS Direct Connect service useful.

Topics

- [Upload Data to Amazon S3](#) (p. 165)
- [Import files with Distributed Cache](#) (p. 168)
- [How to Process Compressed Files](#) (p. 174)
- [Import DynamoDB Data into Hive](#) (p. 174)
- [Connect to Data with AWS DirectConnect](#) (p. 174)
- [Upload Large Amounts of Data with AWS Import/Export](#) (p. 174)

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

Topics

- [Create and Configure an Amazon S3 Bucket](#) (p. 165)
- [Configure Multipart Upload for Amazon S3](#) (p. 167)

Create and Configure an Amazon S3 Bucket

Amazon Elastic MapReduce (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 \(p. 31\)](#) for guidance on choosing a Region.

5. Choose **Create**.

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

Note

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

Note

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

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

To set permissions on an Amazon S3 bucket using the console

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

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

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

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

Configure Multipart Upload for Amazon S3

Amazon Elastic MapReduce (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 S3 Developer Guide*.

Multipart upload allows you to upload a single file to Amazon S3 as a set of parts. Using the AWS Java SDK, you can upload these parts incrementally and in any order. Using the multipart upload method can result in faster uploads and shorter retries than when uploading a single large file.

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

Configuration Parameter Name	Default Value	Description
fs.s3n.multipart.uploads.enabled	True	A boolean type that indicates whether to enable multipart uploads.
fs.s3n.ssl.enabled	True	A boolean type that indicates whether to use http or https.

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. 124).

4. Proceed with creating the cluster as described in [Plan an Amazon EMR Cluster](#) (p. 30).

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:

```
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/configure-hadoop,Args=[ "-c", "fs.s3n.multipart.uploads.enabled=false" ]
```

- 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/configure-hadoop,Args=[ "-c", "fs.s3n.multipart.uploads.enabled=false" ]
```

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>.

Disable Multipart Upload Using the API

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

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

Import files with Distributed Cache

Topics

- [Supported File Types](#) (p. 169)
- [Location of Cached Files](#) (p. 169)
- [Access Cached Files From Streaming Applications](#) (p. 169)
- [Access Cached Files From Streaming Applications Using the Amazon EMR Console](#) (p. 169)
- [Access Cached Files From Streaming Applications Using the AWS CLI](#) (p. 170)

Distributed Cache is a Hadoop feature that can boost efficiency when a map or a reduce task needs access to common data. If your cluster depends on existing applications or binaries that are not installed

when the cluster is created, you can use Distributed Cache to import these files. This feature lets a cluster node read the imported files from its local file system, instead of retrieving the files from other cluster nodes.

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

You invoke Distributed Cache when you create the cluster. The files are cached just before starting the Hadoop job and the files remain cached for the duration of the job. You can cache files stored on any Hadoop-compatible file system, for example HDFS or Amazon S3. The default size of the file cache is 10GB. To change the size of the cache, reconfigure the Hadoop parameter, `local.cache.size` using the (Optional) [Create Bootstrap Actions to Install Additional Software](#) (p. 124) bootstrap action.

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 Hadoop parameter, `mapred.local.dir`, specifies the location of temporary files. Amazon Elastic MapReduce (Amazon EMR) sets this parameter to `/mnt/var/lib/hadoop/mapred/mnt/mapred` or some variation based on instance type. 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.

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

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.

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

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

Example 1

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

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:

```
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_redu
cer.py", "-mapper", "my_mapper.py", "-reducer", "my_reducer.py", "-in
put", "s3://my_bucket/my_input", "-output", "s3://my_bucket/my_output",
"-cacheFile", "s3://my_bucket/sample_dataset.dat#sample_dataset_cached.dat"]
```

- 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 --steps Type=STREAM
ING,Name="Streaming program",ActionOnFailure=CONTIN
UE,Args=["--files", "s3://my_bucket/my_mapper.py s3://my_bucket/my_redu
cer.py", "-mapper", "my_mapper.py", "-reducer", "my_reducer.py", "-in
put", "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:

```
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_buck
et/my_input", "-output", "s3://my_bucket/my_output",
"-cacheFile", "s3://my_bucket/sample_dataset.dat#sample_dataset_cached.dat"]
```

- Windows users:

```
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=STREAM
ING,Name="Streaming program",ActionOnFailure=CONTINUE,Args=["-mapper", "my_map
per.py", "-reducer", "my_reducer.py", "-input", "s3://my_bucket/my_input", "-out
```

```
put", "s3://my_bucket/my_output", "-cacheFile", "s3://my_bucket/sample_data  
set.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.

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.

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:

```
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_redu
cer.py", "-mapper", "my_mapper.py", "-reducer", "my_reducer.py", "-in
put", "s3://my_bucket/my_input", "-output", "s3://my_bucket/my_output", "-cac
heArchive", "s3://my_bucket/sample_dataset.tgz#sample_dataset_cached" ]
```

- 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 --steps Type=STREAM
  ING,Name="Streaming program",ActionOnFailure=CONTIN
  UE,Args=[ "--files", "s3://my_bucket/my_mapper.py s3://my_bucket/my_redu
  cer.py", "-mapper", "my_mapper.py", "-reducer", "my_reducer.py", "-in
  put", "s3://my_bucket/my_input", "-output", "s3://my_bucket/my_output", "-cac
  heArchive", "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:

```
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_buck
et/my_input", "-output", "s3://my_bucket/my_output", "-cac
heArchive", "s3://my_bucket/sample_dataset.tgz#sample_dataset_cached" ]
```

- Windows users:

```
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=STREAM
  ING,Name="Streaming program",ActionOnFailure=CONTINUE,Args=[ "-mapper", "my_map
  per.py", "-reducer", "my_reducer.py", "-input", "s3://my_bucket/my_input", "-out
```

```
put", "s3://my_bucket/my_output", "-cacheArchive", "s3://my_bucket/sample_data  
set.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.

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.

How to Process Compressed Files

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

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

Import DynamoDB Data into Hive

The implementation of Hive provided by Amazon EMR includes functionality that you can use to import and export data between DynamoDB and an Amazon EMR cluster. This is useful if your input data is stored in DynamoDB. For more information, see [Export, Import, Query, and Join Tables in DynamoDB Using Amazon EMR](#) (p. 384).

Connect to Data with AWS DirectConnect

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

Upload Large Amounts of Data with AWS Import/Export

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](#).

Prepare an Output Location (Optional)

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. 175)
- [What formats can Amazon EMR return?](#) (p. 176)
- [How to write data to an Amazon S3 bucket you don't own](#) (p. 177)
- [Compress the Output of your Cluster](#) (p. 179)

Create and Configure an Amazon S3 Bucket

Amazon Elastic MapReduce (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. 31) for guidance on choosing a Region.

5. Choose **Create**.

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

Note

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

Note

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

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

To set permissions on an Amazon S3 bucket using the console

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

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

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

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

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

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

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

- Have the other AWS user grant you write permissions for their Amazon S3 bucket. The cluster you launch runs under your AWS credentials, so any clusters you launch will also be able to write to that other AWS user's bucket.
- Set read permissions for the other AWS user on the files that you or the cluster write to the Amazon S3 bucket. The easiest way to set these read permissions is to use canned access control lists (ACLs), a set of pre-defined access policies defined by Amazon S3.

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

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

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

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

To write files using canned ACLs in Hive

- From the Hive command prompt, set the `fs.s3.canned.acl` configuration option to the canned ACL you want to have the cluster set on files it writes to Amazon S3. To access the Hive command prompt connect to the master node using SSH, and type Hive at the Hadoop command prompt. For more information, see [Connect to the Master Node Using SSH \(p. 441\)](#).

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

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

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

To write files using canned ACLs in Pig

- From the Pig command prompt, set the `fs.s3.canned.acl` configuration option to the canned ACL you want to have the cluster set on files it writes to Amazon S3. To access the Pig command prompt connect to the master node using SSH, and type Pig at the Hadoop command prompt. For more information, see [Connect to the Master Node Using SSH \(p. 441\)](#).

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 \(p. 179\)](#)
- [Intermediate Data Compression \(p. 179\)](#)
- [Using the Snappy Library with Amazon EMR \(p. 179\)](#)

Output Data Compression

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

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

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

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

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

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

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

Intermediate Data Compression

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

When writing a Custom Jar, use the following command:

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

Using the Snappy Library with Amazon EMR

Snappy is a compression and decompression library that is optimized for speed. It is available on Amazon EMR AMIs version 2.0 and later and is used as the default for intermediate compression. For more

information about Snappy, go to <http://code.google.com/p/snappy/>. For more information about Amazon EMR AMI versions, go to [Choose an Amazon Machine Image \(AMI\)](#) (p. 48).

Configure Access to the Cluster

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

Amazon EMR provides several ways to control access to the resources of your cluster. You can use AWS Identity and Access Management (IAM) to create user accounts and roles and configure permissions that control which AWS features those users and roles can access. Two IAM roles, a service role and an instance profile, are required for your cluster.

Note

If you created your first cluster before April 6, 2015, IAM roles are not mandatory. However, we recommend that you specify IAM roles for your cluster as a security best practice.

When you launch a cluster, you can associate an EC2 key pair with the cluster that you can use to connect to the cluster using SSH. You can also set permissions that allow users other than the default Hadoop user to submit jobs to your cluster.

Topics

- [Create SSH Credentials for the Master Node](#) (p. 180)
- [Configure IAM User Permissions](#) (p. 181)
- [Set Access Policies for IAM Users](#) (p. 183)
- [Configure IAM Roles for Amazon EMR](#) (p. 185)
- [Configure Security Groups for Amazon EMR](#) (p. 193)
- [Setting Permissions on the System Directory](#) (p. 200)

Create SSH Credentials for the Master Node

Create an Amazon EC2 Key Pair and PEM File

Amazon EMR uses an Amazon Elastic Compute Cloud (Amazon EC2) key pair to ensure that you alone have access to the instances that you launch. The PEM file associated with this key pair is required to `ssh` directly to the master node of the cluster.

To create an Amazon EC2 key pair

1. Sign in to the AWS Management Console and open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. From the Amazon EC2 console, select a **Region**.
3. In the **Navigation** pane, click **Key Pairs**.
4. On the **Key Pairs** page, click **Create Key Pair**.
5. In the **Create Key Pair** dialog box, enter a name for your key pair, such as, `mykeypair`.
6. Click **Create**.
7. Save the resulting PEM file in a safe location.

Your Amazon EC2 key pair and an associated PEM file are created.

Modify Your PEM File

Amazon Elastic MapReduce (Amazon EMR) enables you to work interactively with your cluster, allowing you to test cluster steps or troubleshoot your cluster environment. To log in directly to the master node of your running cluster, you can use `ssh` or PuTTY. You use your PEM file to authenticate to the master node. The PEM file requires a modification based on the tool you use that supports your operating system. You use the CLI to connect on Linux, UNIX, or Mac OS X computers. You use PuTTY to connect on Microsoft Windows computers. For more information about how to install the Amazon EMR CLI or how to install PuTTY, go to the [Amazon Elastic MapReduce Getting Started Guide](#).

To modify your credentials file

- Create a local permissions file:

If you are using...	Do this...
Linux, UNIX, or Mac OS X	<p>Set the permissions on the PEM file or your Amazon EC2 key pair. For example, if you saved the file as <code>mykeypair.pem</code>, the command looks like the following:</p> <pre>chmod og-rwx mykeypair.pem</pre>
Microsoft Windows	<ol style="list-style-type: none"> Download PuTTYgen.exe to your computer from http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html. Launch PuTTYgen. Click Load. Select the PEM file you created earlier. Click Open. Click OK on the PuTTYgen Notice telling you the key was successfully imported. Click Save private key to save the key in the PPK format. When PuTTYgen prompts you to save the key without a pass phrase, click Yes. Enter a name for your PuTTY private key, such as, <code>mykeypair.ppk</code>. Click Save. Exit the PuTTYgen application.

Your credentials file is modified to allow you to log in directly to the master node of your running cluster.

Configure IAM 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 user permissions. For more information on IAM, go to [Using IAM](#) in the *IAM User Guide* guide.

IAM enables you to create users under your AWS account. You can define policies that limit the actions those users can take with your AWS resources. For example, you can choose to give an IAM user the ability to view, but not to create or delete, Amazon S3 buckets in your AWS account. However, you cannot specify a specific Amazon EMR resource in a policy, such as a specific cluster. IAM is available at no charge to all AWS account holders; you do not 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.

Instead of giving permissions to individual users, it can be convenient to use IAM roles and group users with certain permissions. For more information, see [Configure IAM Roles for Amazon EMR \(p. 185\)](#).

Configuring Cluster Visibility

By default, clusters created using the console and AWS CLI are visible to all IAM users. You can change this setting when you launch a cluster or after the cluster is created. If an IAM user launches a cluster, and that cluster is hidden from other IAM users on the AWS account, only that user will see the cluster. For example, if an IAM user uses the AWS CLI to run the `list-clusters` command, clusters created by other IAM users with IAM user visibility set to "No other IAM users" are not listed. This filtering occurs on all Amazon EMR interfaces—the console, CLI, API, and SDKs—and prevents IAM users from inadvertently changing clusters created by other IAM users. It is useful for clusters that are intended to be viewed by only a single IAM user and the main AWS account.

Note

This filtering does not prevent IAM users from viewing the underlying resources of the cluster, such as EC2 instances, by using AWS interfaces outside of Amazon EMR.

The default option, launching a cluster with IAM user visibility set to "All other IAM users," makes a cluster visible and accessible to all IAM users under a single AWS account. Using this feature, all IAM users on your account have access to the cluster and, by configuring the policies of the IAM groups they belong to, you control how those users interact with the cluster. For example, Devlin, a developer, belongs to a group that has an IAM policy that grants full access to all Amazon EMR functionality. He could launch a cluster that is visible to all other IAM users on his company's AWS account. A second IAM user, Ann, a data analyst with the same company, could then run queries on that cluster. Because Ann does not launch or terminate clusters, the IAM policy for the group she is in would only contain the permissions necessary for her to run her queries.

To configure cluster access 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 **Security and Access** section, in the **IAM User Access** field, choose **All other IAM users** to make the cluster visible and accessible to all IAM users on the AWS account (the default option). Choose **No other IAM users** to restrict access. For more information, see [Configure IAM User Permissions \(p. 181\)](#).
4. Proceed with creating the cluster as described in [Plan an Amazon EMR Cluster \(p. 30\)](#).

To configure cluster access using the AWS CLI

By default, clusters created using the AWS CLI are visible to all users. You may optionally specify the `--visible-to-all-users` parameter to make a cluster visible to all IAM users. To restrict cluster access on a new cluster using the AWS CLI, type the `create-cluster` command with the `--no-visible-to-all-users` parameter.

- Linux, UNIX, and Mac OS X users:

```
aws emr create-cluster --name "Test cluster" --ami-version 3.3 --no-visible-to-all-users \
--applications Name=Hive Name=Pig Name=Hue \
--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 --no-visible-to-all-users --applications Name=Hive Name=Pig Name=Hue --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. If these roles do not exist in the AWS CLI configuration file, they will automatically be populated in that file and subsequent use of the CLI will not require the `--use-default-roles` option for AWS CLI EMR subcommands.

If you are configuring cluster access to an existing cluster, type the `modify-cluster-attributes` subcommand with the `--no-visible-to-all-users` parameter or the `--visible-to-all-users` parameter. The visibility of a running cluster can be changed only by the IAM user that created the cluster or the AWS account owner. To use this subcommand, you need the cluster identifier available via the console or the `list-clusters` subcommand.

To restrict access to a running cluster, type the following command.

```
aws emr modify-cluster-attributes --cluster-id j-1GMZXXXXXXXXYMZ --no-visible-to-all-users
```

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

Set Access Policies for IAM Users

The ability for IAM users to perform certain actions with Amazon EMR is controlled by IAM policies. IAM policies provide fine-grained control over the level of access and the criteria by which Amazon EMR grants access to IAM users.

Note

At a minimum, an IAM user needs the following permission set in their IAM policy to access the Amazon EMR console:

```
elasticmapreduce:ListClusters
```

In an IAM policy, to specify Amazon EMR actions, the action name must be prefixed with the lowercase string `elasticmapreduce`. You use wildcards to specify all actions related to Amazon EMR. The wildcard `"*"` matches zero or multiple characters.

For a complete list of Amazon EMR actions, see the API action names in the [Amazon EMR API Reference](#). For more information about permissions and policies, see [Permissions and Policies](#) in the *IAM User Guide*.

Full Access Policy

The following policy gives permissions for all actions required to use Amazon EMR. This policy includes actions for Amazon EC2, Amazon S3, and CloudWatch, as well as for all Amazon EMR actions. Amazon EMR relies on these additional services to perform such actions as launching instances, writing log files, or managing Hadoop jobs and tasks. You can also specify the AWS managed policy `AmazonElasticMapReduceFullAccess`, which is automatically updated with new permissions when they are required for Amazon EMR full access. For information about attaching managed policies to users, see [Managing Managed Policies](#) in the *IAM User Guide* guide.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "cloudwatch:*",
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:CancelSpotInstanceRequests",
        "ec2:CreateSecurityGroup",
        "ec2:CreateTags",
        "ec2>DeleteTags",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeAccountAttributes",
        "ec2:DescribeInstances",
        "ec2:DescribeKeyPairs",
        "ec2:DescribeRouteTables",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSpotInstanceRequests",
        "ec2:DescribeSpotPriceHistory",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcAttribute",
        "ec2:DescribeVpcs",
        "ec2:ModifyImageAttribute",
        "ec2:ModifyInstanceAttribute",
        "ec2:RequestSpotInstances",
        "ec2:RunInstances",
        "ec2:TerminateInstances",
        "elasticmapreduce:*",
        "iam:ListRoles",
        "iam:PassRole",
        "kms:List*",
        "s3:*",
        "sdb:*",
        "support:CreateCase",
        "support:DescribeServices",
        "support:DescribeSeverityLevels"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Note

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

Related Topics

- [Attaching Managed Policies](#) (*IAM User Guide* guide)

Read-Only Access Policy

The following managed policy, `AmazonElasticMapReduceReadOnlyAccess`, gives an IAM user read-only access to Amazon EMR. For information about attaching managed policies to users, see [Managing Managed Policies](#) in the *IAM User Guide* guide.

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

Related Topics

- [Attaching Managed Policies](#) (*IAM User Guide* guide)

Configure IAM Roles for Amazon EMR

Caution

IAM roles will be mandatory for all users after June 30, 2015.

AWS Identity and Access Management (IAM) roles provide a way for IAM users or AWS services to have certain specified permissions and access to resources. For example, this may allow users to access resources or other services to act on your behalf. You must specify two IAM roles for a cluster: a role for the Amazon EMR service (*service role*), and a role for the EC2 instances (*instance profile*) that Amazon EMR manages.

The service role defines the allowable actions for Amazon EMR based on the granted permissions. A user must specify the service role when a cluster is created. When the user accesses the cluster, Amazon EMR assumes the IAM role specified in the cluster definition, obtains the permissions of the assumed role, and then tries to execute requests using those permissions. The permissions determine which AWS resources a service can access, and what the service is allowed to do with those resources. Service role permissions are independent of the permissions granted to the IAM user who called the service, and they can therefore be managed separately by an AWS administrator.

Note

The user who sets up the roles for use with Amazon EMR should be an IAM user with administrative permissions. We recommend that all administrators use AWS MFA (multi-factor authentication).

The EC2 instance profile determines the permissions for applications that run on EC2 instances. For example, when Hive, an application on the cluster, needs to write output to an Amazon S3 bucket, the instance profile determines whether Hive has permissions to write to Amazon S3.

Using IAM roles with Amazon EMR allows a user to tailor a permissions policy that closely fits the usage patterns of the cluster. For more information about instance profiles, see [Granting Applications that Run on Amazon EC2 Instances Access to AWS Resources](#) in the *IAM User Guide* guide.

Topics

- [Required IAM Roles for Amazon EMR \(p. 186\)](#)
- [Default IAM Roles for Amazon EMR \(p. 186\)](#)
- [Create and Use IAM Roles for Amazon EMR \(p. 188\)](#)
- [Custom IAM Roles for Amazon EMR \(p. 190\)](#)
- [Launch a Cluster with IAM Roles \(p. 191\)](#)
- [Access AWS Resources Using IAM Roles \(p. 192\)](#)

Required IAM Roles for Amazon EMR

Existing AWS customers whose accounts were created prior to the launch of IAM roles for the Amazon EMR service are not immediately required to use a role for the Amazon EMR service in existing public regions. Existing customers are required to use roles in private regions (such as AWS GovCloud (US)) and all new public regions. Eventually, Amazon EMR roles and Amazon EC2 roles will be required for all users in all regions. For more information about service and Amazon EC2 roles, see [Use Cases: Roles for Users, Applications, and Services](#) and [Use roles for applications that run on Amazon EC2 instances](#).

Roles are required in the AWS GovCloud (US) region. If you are launching a cluster into the AWS GovCloud (US) region, for security purposes, you must launch the cluster in a VPC and specify an IAM role.

Default IAM Roles for Amazon EMR

To simplify using IAM roles, Amazon EMR can create a default EMR (service) role and EC2 role, which are attached to the AWS managed policies, **AmazonElasticMapReduceRole** and **AmazonElasticMapReduceforEC2Role**. AWS managed standalone policies are policies that are created and managed by AWS to attach to policies or roles required by services. For more information, see [Managed Policies and Inline Policies](#) in the *IAM User Guide* guide. You can view the most up-to-date managed policies for the Amazon EMR service role and EC2 role in the **Policies** tab in the AWS IAM Management Console. For more information, see [Create and Use IAM Roles for Amazon EMR \(p. 188\)](#).

The default roles allow Amazon EMR and EC2 instances access to the AWS services and resources suitable for most clusters. Your specific application may require other permissions. For more information, see [Custom IAM Roles for Amazon EMR \(p. 190\)](#).

If you are using an IAM user and creating default roles for a cluster, your IAM user must have the `iam:CreateRole`, `iam:PutRolePolicy`, `iam:CreateInstanceProfile`, `iam:AddRoleToInstanceProfile`, and `iam:PassRole` permissions. The `iam:PassRole` permission allows cluster creation. The remaining permissions allow creation of the default roles. If you are using custom IAM roles, your IAM user only needs the `iam:GetInstanceProfile`, `iam:GetRole`, and `iam:PassRole` permissions.

Note

You can view the most up-to-date managed policies for the Amazon EMR service role and EC2 role in the **Policies** tab in the AWS IAM Management Console.

Topics

- [Default Service Role for Amazon EMR \(p. 187\)](#)
- [Default EC2 Role for Amazon EMR \(p. 187\)](#)

Default Service Role for Amazon EMR

The default IAM role for the Amazon EMR service is `EMR_DefaultRole`, which is by default attached to **AmazonElasticMapReduceRole**. This role consists of a role policy and a trust policy. You can view the most up-to-date **AmazonElasticMapReduceRole** in the **Policies** tab in the [AWS IAM Management Console](#).

Attaching a Pre-existing `EMR_DefaultRole` to the Managed Policy Using the Console

If you previously created default roles before 6/11/2015, Amazon EMR did not attach managed policies to the role by default. To attach managed policies to your pre-existing role:

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. Click **Policies**.
3. Click **AmazonElasticMapReduceRole**.
4. Under **Attached Entities**, click **Attach**. Select `EMR_DefaultRole` and click **Attach**.

Attaching a Pre-existing `EMR_DefaultRole` to the Managed Policy Using the CLI

Use the following syntax to attach your pre-existing default role to the managed policy:

```
$ aws iam attach-role-policy --role-name EMR_DefaultRole\  
--policy-arn arn:aws:iam::aws:policy/service-role/AmazonElasticMapReduceRole  
$ aws iam delete-role-policy --role-name EMR_DefaultRole --policy-name EMR_De  
faultRole
```

Default EC2 Role for Amazon EMR

The default EC2 role is `EMR_EC2_DefaultRole`, which is by default attached to the **AmazonElasticMapReduceforEC2Role** managed policy. This role consists of a role policy and a trust policy. You can view the most up-to-date **AmazonElasticMapReduceforEC2Role** in the **Policies** tab in the [AWS IAM Management Console](#).

Attaching a Pre-existing `EMR_EC2_DefaultRole` to the Managed Policy Using the Console

If you previously created default roles before 06/11/2015, Amazon EMR did not attach managed policies to the role by default. To attach managed policies to your pre-existing roles:

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. Click **Policies**.
3. Click **AmazonElasticMapReduceforEC2Role**.
4. Under **Attached Entities**, click **Attach**. Select `EMR_EC2_DefaultRole` and click **Attach**.

Attaching a Pre-existing `EMR_EC2_DefaultRole` to the Managed Policy Using the CLI

Use the following AWS CLI syntax to attach your pre-existing default role to the managed policy:

```
$ aws iam attach-role-policy --role-name EMR_EC2_DefaultRole\  
--policy-arn arn:aws:iam::aws:policy/service-role/AmazonElasticMapReduce  
forEC2Role  
$ aws iam delete-role-policy --role-name EMR_EC2_DefaultRole --policy-name  
EMR_EC2_DefaultRole
```

Create and Use IAM Roles for Amazon EMR

Caution

IAM roles will be mandatory for all users after June 30, 2015.

There are three ways to create IAM roles:

- Use the Amazon EMR console to create the default roles. For more information, see [Create and Use IAM Roles with the Amazon EMR Console](#) (p. 188).
- Use the AWS CLI to create the default roles using the `create-default-roles` subcommand. For more information, see [Create and Use IAM Roles with the AWS CLI](#) (p. 188).
- Use the IAM console or API to create roles. For more information, see [Creating a Role for an AWS Service](#) in the *IAM User Guide* guide.

Note

When you use the IAM console to create a role where Amazon EC2 is the principal, IAM automatically creates an instance profile with the same name as the role, to contain the role and pass information to EC2 instances. For more information, see [Instance Profiles](#) in the *IAM User Guide* guide.

Create and Use IAM Roles with the Amazon EMR Console

AWS customers whose accounts were created after release of Amazon EMR roles are required to specify an EMR (service) role and an EC2 instance profile in all regions when using the console. You can create default roles at cluster launch using the console, or you can specify other roles you may already be using.

If you are using an IAM user and creating default roles for a cluster using the console, your IAM user must have the `iam:CreateRole`, `iam:PutRolePolicy`, `iam:CreateInstanceProfile`, `iam:AddRoleToInstanceProfile`, and `iam:PassRole` permissions. The `iam:PassRole` permission allows cluster creation. The remaining permissions allow creation of the default roles.

To create and use IAM roles with the console

1. Sign in to the AWS Management Console and open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Click **Create Cluster**.
3. In the **Security and Access** section, in the **IAM Roles** subsection, for **Roles configuration**, choose **Default**. If the default roles do not exist, they are created for you (assuming that you have appropriate permissions). If the roles exist, they are used for your cluster. After the roles are created, they are visible in the IAM console.

Note

To use custom roles with your cluster, click **Custom** and choose the existing roles from the list.

Create and Use IAM Roles with the AWS CLI

You can create the default Amazon EMR (service) role and EC2 instance profile using the CLI. Once the roles are created, they are visible in the IAM console. If not already present the roles are also auto-populated in the AWS CLI configuration file located at `~/.aws/config` on Unix, Linux, and OS X systems or at `C:\Users\USERNAME\.aws\config` on Windows systems. After creation, you can use the default roles when you launch a cluster.

To create and use IAM roles with the AWS CLI

To create default roles using the AWS CLI, type the `create-default-roles` subcommand. To use the default roles at cluster launch, type the `create-cluster` subcommand with the `--use-default-roles` parameter.

1. Type the following command to create default roles using the AWS CLI:

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

The output of the command lists the contents of the default EMR role, `EMR_DefaultRole`, and the default EC2 instance profile, `EMR_EC2_DefaultRole`. The AWS CLI configuration file will be populated with these role names for the `service_role` and `instance_profile` values. For example, after this command, the configuration file might look like:

```
[default]
output = json
region = us-east-1
aws_access_key_id = myAccessKeyID
aws_secret_access_key = mySecretAccessKey
emr =
  service_role = EMR_DefaultRole
  instance_profile = EMR_EC2_DefaultRole
```

You can also modify this file to use your own custom role and AWS CLI will use that role by default.

2. If the default roles already exist, you can use them when launching a cluster. Type the following command to use existing default roles when launching a cluster and replace `myKey` with the name of your EC2 key pair.

- Linux, UNIX, and Mac OS X users:

```
aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applica
tions Name=Hue Name=Hive Name=Pig \
--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 --applica
tions Name=Hue Name=Hive Name=Pig --use-default-roles --ec2-attributes
KeyName=myKey --instance-type m3.xlarge --instance-count 3
```

Note

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

We recommend that you begin by creating the default roles, then modify those roles as needed. For more information about default roles, see [Default IAM Roles for Amazon EMR \(p. 186\)](#).

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

Custom IAM Roles for Amazon EMR

If the default IAM roles provided by Amazon EMR do not meet your needs, you can create custom IAM roles instead. For example, if your application does not access Amazon DynamoDB, you can leave out DynamoDB permissions in your custom IAM role.

For more information about creating and managing IAM roles, see the following topics in the *IAM User Guide* guide.

- [Creating a Role](#)
- [Modifying a Role](#)
- [Deleting a Role](#)

We recommend that you use the permissions in the managed policies

AmazonElasticMapReduceforEC2Role and **AmazonElasticMapReduceRole** as a starting place when developing custom IAM roles to use with Amazon EMR. You can then copy the contents of these default roles, create new IAM roles, paste in the copied permissions, and modify the pasted permissions.

The following is an example of a custom instance profile for use with Amazon EMR. This example is for a cluster that does not use Amazon RDS, or DynamoDB.

The access to Amazon SimpleDB is included to permit debugging from the console. Access to CloudWatch is included so the cluster can report metrics. Amazon SNS and Amazon SQS permissions are included for messaging.

```
{
  "Statement": [
    {
      "Action": [
        "cloudwatch:*",
        "ec2:Describe*",
        "elasticmapreduce:Describe*",
        "s3:*",
        "sdb:*",
        "sns:*",
        "sqs:*"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Important

The IAM role name and the instance profile name must match exactly when you use either the Amazon EMR console or CLI. The console shows IAM role names in the **EC2 instance profile** list. The CLI interprets the name as instance profile names. For this reason, if you use the IAM CLI or API to create an IAM role and its associated instance profile, we recommend that you give the new IAM role the same name as its associated instance profile. By default, if you create an IAM role with the IAM console and do not specify an instance profile name, the instance profile name is the same as the IAM role name.

In some situations, you might need to work with an IAM role whose associated instance profile does not have the same name as the role. This can occur if you use AWS CloudFormation to manage IAM roles for you, because AWS CloudFormation adds a suffix to the role name to create the instance profile name. In this case, you can use the Amazon EMR API or CLI to specify the instance profile name. Unlike the console, the API and CLI does not interpret an

instance profile name as IAM role name. In the API, you can call the RunJobFlow action and pass the instance profile name for the JobFlowRole parameter. In the CLI, you can specify the instance profile name for the `--ec2-attributes InstanceProfile` option of `aws emr create-cluster` command.

Launch a Cluster with IAM Roles

The IAM user creating clusters needs permissions to retrieve and assign roles to Amazon EMR and EC2 instances. If the IAM user lacks these permissions, you get the error **User account is not authorized to call EC2**. You assign the correct role by creating the default roles as discussed in [Default IAM Roles for Amazon EMR \(p. 186\)](#).

To launch a cluster with IAM roles using the console

- On the create cluster page in the IAM Roles section, specify **Default** or **Custom**.
 - a. If you choose Default, you can optionally click **View policies for default roles**.
 - b. If you choose Custom, specify the IAM roles using the **EMR role** and **EC2 instance profile** fields. For more information, see [Create and Use IAM Roles with the Amazon EMR Console \(p. 188\)](#)

To launch a cluster with IAM roles using the AWS CLI

You can specify an EMR service role and EC2 instance profile using the AWS CLI. When launching a cluster, type the `create-cluster` subcommand with the `--service-role` and `--ec2-attributes InstanceProfile` parameters.

- Type the following command to specify an EMR role and EC2 instance profile when launching a cluster. This example uses the default EMR role, `EMR_DefaultRole`, and the default EC2 instance profile, `EMR_EC2_DefaultRole`.
 - 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 \  
--service-role EMR_DefaultRole --ec2-attributes InstanceProfile=EMR_EC2_De  
faultRole \  
--ec2-attributes KeyName=myKey \  
--instance-type m3.xlarge --instance-count 3
```

- Windows users:

```
aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applica  
tions Name=Hue Name=Hive Name=Pig --service-role EMR_DefaultRole --ec2-  
attributes InstanceProfile=EMR_EC2_DefaultRole --ec2-attributes Key  
Name=myKey --instance-type m3.xlarge --instance-count 3
```

Alternatively, you can use the `--use-default-roles` option, which assumes those roles have been created:

- 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
```

- Windows users:

```
aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applica  
tions Name=Hue Name=Hive Name=Pig --use-default-roles --ec2-attributes  
KeyName=myKey --instance-type m3.xlarge --instance-count 3
```

Another alternative is to set the service role and the instance profile located in the AWS CLI configuration file. For more information, see [Create and Use IAM Roles for Amazon EMR \(p. 188\)](#)

Note

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

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

Access AWS Resources Using IAM Roles

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.
 - [Using IAM Roles for Amazon EC2 Instances with the SDK for Java](#) in the *AWS SDK for Java Developer Guide*
 - [Using IAM Roles for Amazon EC2 Instances with the SDK for .NET](#) in the *AWS SDK for .NET Developer Guide*
 - [Using IAM Roles for Amazon EC2 Instances with the SDK for PHP](#) in the *AWS SDK for PHP Developer Guide*
 - [Using IAM Roles for Amazon EC2 Instances with the SDK for Ruby](#) in the *AWS SDK for Ruby Developer Guide*

To obtain temporary credentials from EC2 instance metadata

- Call the following URL from an EC2 instance that is running with the specified IAM role, which will return the associated temporary security credentials (AccessKeyId, SecretAccessKey, SessionToken, and Expiration). The example that follows uses the default instance profile for Amazon EMR, `EMR_EC2_DefaultRole`.

```
GET http://169.254.169.254/latest/meta-data/iam/security-credentials/EMR_EC2_DefaultRole
```

For more information about writing applications that use IAM roles, see [Granting Applications that Run on Amazon EC2 Instances Access to AWS Resources](#).

For more information about temporary security credentials, see [Using Temporary Security Credentials](#) in the *Using Temporary Security Credentials* guide.

Configure Security Groups for 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 and ElasticMapReduce-slave), or you can specify your own security groups for the master and core/task instances
- Additional security groups: you can specify security groups that apply to your Amazon EC2 instances in addition to the Amazon EMR–managed security groups

Security groups need to be specified at cluster launch using the console, CLI, API, or SDK. Instances in running clusters cannot be assigned new security groups, but you can edit or add rules to existing security groups. Edited rules and new rules added to security groups take effect when the rules are saved.

For more information, see [Amazon EC2 Security Groups](#) in the *Amazon EC2 User Guide for Linux Instances*. For more information about Amazon VPC security groups, see [Security Groups for Your VPC](#) in the *Amazon VPC User Guide*.

Topics

- [Amazon EMR–Managed Security Groups](#) (p. 193)
- [Amazon EMR Additional Security Groups](#) (p. 198)

Amazon EMR–Managed Security Groups

When you launch an Amazon EMR cluster, you must specify one Amazon EMR–managed security group for the master instance and one Amazon EMR–managed security group for the core/task (slave) instances. The core/task instances share the same security group.

You can use the default Amazon EMR–managed security groups for the master and core/task instances—ElasticMapReduce-master and ElasticMapReduce-slave—or you can choose your own Amazon EMR–managed 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`. 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 \(p. 196\)](#).

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 \(p. 196\)](#).

If you need to control instance membership in the Amazon EMR–managed security groups, you can create your own security groups for the master and core/task instances. For more information, see [Security Groups for Your VPC](#) in the *Amazon VPC User Guide*. When you create your own security groups, any inbound and outbound rules required for proper intra-cluster communication are written to the groups with some exceptions.

Typically, you would create your own Amazon EMR–managed security groups to isolate clusters so they cannot communicate with each other. This alleviates the need to create separate VPCs or AWS accounts for each of your clusters. For example, to isolate Amazon EMR clusters in a VPC, you create a security group for each cluster's master node, and you create a security group for each cluster's core/task nodes. When you launch a cluster, the rules in your security groups are applied only to the instances in that cluster, effectively preventing instances in separate clusters from communicating with each other.

If you need separate clusters to communicate, and you are using your own security groups, you can create an additional security group containing the rules necessary for the cluster instances to communicate with each other, or you can use the same security groups for both clusters. If you do not specify an additional security group when launching the clusters, and the clusters use different security groups, the clusters cannot communicate with each other unless you modify the rule sets in your master and core/task security groups. For more information about additional security groups, see [Amazon EMR Additional Security Groups \(p. 198\)](#).

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`. 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`. 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.
4. Proceed with creating the cluster as described in [Plan an Amazon EMR Cluster \(p. 30\)](#).

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, and replace *mybucket* with your bucket name.

- 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
```

- 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
```

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 *mybucket* with your bucket name, replace *masterSecurityGroupId* with the ID of the master security group, and replace *slaveSecurityGroupId* with the ID of the core/task security group.

- 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 \
--emr-managed-master-security-group masterSecurityGroupId --emr-managed-slave-security-group slaveSecurityGroupId \
--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 --emr-managed-master-security-group masterSecurityGroupId --emr-managed-slave-security-group slaveSecurityGroupId --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.

- 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 about using Amazon EMR commands in the AWS CLI, see the [AWS Command Line Interface Reference](#).

Default Options for Amazon EMR–Managed Security Groups

If you launch an Amazon EMR cluster using the default security groups, two groups are created: ElasticMapReduce-master and ElasticMapReduce-slave. 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 default security groups.

You can launch a cluster with the default security groups using the console, the API, the CLI, or the SDK. If you use the default security groups, there is no need to change your existing code or to add parameters to your CLI commands.

You can launch a cluster with the default security groups using the console. If the default security groups do not exist, they are created before your cluster is launched. If they do exist, they are automatically assigned.

Rules in Amazon EMR–Managed Security Groups

The tables in this section list the inbound and outbound access rules added to the Amazon EMR–managed security groups. IP address ranges and rules are automatically updated for Amazon EMR–managed security groups.

The following rules are added to Amazon EMR–managed master security groups:

Type	Pro- to- col	Port Range	Source	Details
<i>Inbound rules</i>				
All ICMP	ICMP	All	The default ElasticMapReduce-master security group ID or the master security group ID that you specify; for example, sg-88XXXXed	Allows inbound traffic from any instance in the ElasticMapReduce-master security group. By default, the master nodes in all Amazon EMR clusters in a single VPC can communicate with each other over any TCP, UDP, or ICMP port. If you choose your own master security group, only master instances in the group can communicate with each other over any TCP, UDP, or ICMP port.
All TCP	TCP	All		
All UDP	UDP	All		

Type	Protocol	Port Range	Source	Details
All ICMP	ICMP	All	The default ElasticMapReduce-slave security group ID or the core/task security group ID that you specify; for example, sg-8bXXXXee	Allows inbound traffic from any instance in the ElasticMapReduce-slave security group. By default, the master node accepts inbound communication from any core/task node in any Amazon EMR cluster in a single VPC over any TCP, UDP, or ICMP port. If you choose your own core/task security group, only core/task instances in this group can communicate with the master node over any TCP, UDP, or ICMP port.
All TCP	TCP	All		
All UDP	UDP	All		
HTTPS	TCP	8443	Various Amazon IP address ranges	Allows the cluster manager to communicate with the master nodes in each Amazon EMR cluster in a single VPC.
SSH	TCP	22	0.0.0.0/0	Allows inbound access to the master node via SSH from any IP address. This rule can be edited to limit access to individual IP addresses or address ranges. Note If you delete the SSH rule from the default master security group, Amazon EMR does not replace it. When you choose your own master security group, the inbound SSH rule is not added to the rule set.
<i>Outbound rules</i>				
All traffic	All	All	0.0.0.0/0	Provides outbound access to the Internet from any instance in the ElasticMapReduce-master security group or the group that you specify.

The following rules are added to Amazon EMR–managed core/task security groups:

Type	Protocol	Port Range	Source	Details
<i>Inbound rules</i>				
All ICMP	ICMP	All	The default ElasticMapReduce-master security group ID or the master security group ID that you specify; for example, sg-88XXXXed	Allows inbound traffic from any instance in the ElasticMapReduce-master security group or the group that you specify. By default, the core/task nodes in all Amazon EMR clusters in a single VPC accept inbound communication from master nodes over any TCP, UDP, or ICMP port. If you choose your own master security group, only master instances in this group can communicate with the core/task nodes over any TCP, UDP, or ICMP port.
All TCP	TCP	All		
All UDP	UDP	All		

Type	Protocol	Port Range	Source	Details
All ICMP	ICMP	All	The default ElasticMapReduce-slave security group ID or the core/task security group ID that you specify; for example, sg-8bXXXXee	Allows inbound traffic from any instance in the ElasticMapReduce-slave security group. By default, the core/task nodes accept inbound communication from any other core/task node in any Amazon EMR cluster in a single VPC over any TCP, UDP, or ICMP port. If you choose your own core/task security group, only core/task instances in this group can communicate with each other over any TCP, UDP, or ICMP port.
All TCP	TCP	All		
All UDP	UDP	All		
<i>Outbound rules</i>				
All traffic	All	All	0.0.0.0/0	Provides outbound access to the Internet from all instances in the ElasticMapReduce-slave security group or the group that you specify.

Amazon EMR Additional Security Groups

Whether you use the default managed security groups or your own custom managed security groups, you can assign additional security groups to the master and core/task instances in your cluster. Applying additional security groups gives you the ability to apply additional rules to your security groups so they do not have to be modified. Additional security groups are optional. They can be applied to the master group, core & task group, both groups, or neither group. You can also apply the same additional security groups to multiple clusters.

For example, if you are using your own managed security groups, and you want to allow inbound SSH access to the master group for a particular cluster, you can create an additional security group containing the rule and add it to the master security group for that cluster. Additional security groups are not modified by or maintained by Amazon EMR.

Typically, additional security groups are used to:

- Add access rules to instances in your cluster that are not present in the Amazon EMR–managed security groups
- Give particular clusters access to a specific resource such as a Amazon Redshift database

Security groups are restrictive by default. They reject all traffic. You can add a rule to allow traffic on a particular port to your custom or additional security groups. If there is more than one rule for a specific port in two security groups that apply to the same instances, the most permissive rule is applied. For example, if you have a rule that allows SSH access via TCP port 22 from IP address 203.0.113.1 and another rule that allows access to TCP port 22 from any IP address (0.0.0.0/0), the rule allowing access by any IP address takes precedence.

You can apply up to four additional security groups to both the master and core/task security groups. The number of additional groups allowed is dependent upon:

- The number of security groups allowed for your account
- The number of individual rules allowed for your account

For more information about rule limits in VPC security groups, see [Security Groups for Your VPC](#) in the *Amazon VPC User Guide*. You can assign the same additional security groups to both the master and core/task security groups.

Additional security groups can be applied using the console, the API, the CLI, or the SDK.

To specify additional security groups using the console

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Choose **Create cluster**.
3. 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, click 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, click 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 an Amazon EMR Cluster \(p. 30\)](#).

To specify additional security groups using the AWS CLI

Use the `create-cluster` command with the `--additional-master-security-groups` and `--additional-slave-security-groups` parameters. 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, replace `mybucket` with your bucket name, and replace `securityGroupId` with the ID of your master security group, core/task security group, and additional security groups.
 - Linux, UNIX, and Mac OS X users:

```
aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applica
tions Name=Hue Name=Hive Name=Pig \
--emr-managed-master-security-group securityGroupId --emr-managed-slave-
security-group securityGroupId \
--additional-master-security-groups securityGroupId --additional-slave-
security-groups securityGroupId \
--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 --applica
tions Name=Hue Name=Hive Name=Pig --emr-managed-master-security-group se
```



```
curityGroupId --emr-managed-slave-security-group securityGroupId --additional-master-security-groups securityGroupId --additional-slave-security-groups securityGroupId --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 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 about using Amazon EMR commands in the AWS CLI, see the [AWS Command Line Interface Reference](#).

Setting Permissions on the System Directory

You can set the permissions on the release directory by using a bootstrap action to modify the `mapreduce.jobtracker.system.dir.permission` configuration variable. This is useful if you are running a custom configuration in which users other than "hadoop user" submit jobs to Hadoop.

To set permissions on the system directory

- Specify a bootstrap action that sets the permissions for the directory using numerical permissions codes (octal notation). The following example bootstrap action, with a permissions code of `777`, gives full permissions to the user, group, and world. For more information about octal notation, go to http://en.wikipedia.org/wiki/File_system_permissions#Octal_notation.

```
s3://elasticmapreduce/bootstrap-actions/configure-hadoop \  
--args "-s,mapreduce.jobtracker.system.dir.permission=777"
```

Configure Logging and Debugging (Optional)

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.

Topics

- [Default Log Files \(p. 201\)](#)
- [Archive Log Files to Amazon S3 \(p. 201\)](#)
- [Enable the Debugging Tool \(p. 204\)](#)

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 \(p. 441\)](#). 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/>.
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 \(p. 413\)](#).

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 \(p. 413\)](#).

5. Proceed with creating the cluster as described in [Plan an Amazon EMR Cluster \(p. 30\)](#).

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:

```
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-attributes KeyName=myKey \
--instance-type m3.xlarge --instance-count 3
```

- Windows users:

```
aws emr create-cluster --name "Test cluster" --ami-version 3.8 --log-uri
s3://mybucket/logs/ --applications Name=Hue Name=Hive Name=Pig --use-de
fault-roles --ec2-attributes KeyName=myKey --instance-type m3.xlarge -
-instance-count 3
```

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

Note

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

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

To aggregate logs in Amazon S3 using the AWS CLI

Note

You cannot currently use log aggregation with the `yarn logs` utility. You can only use aggregation supported by this procedure.

Log aggregation (Hadoop 2.x) compiles logs from all containers for an individual application into a single file. To enable log aggregation to Amazon S3 using the AWS CLI, you use a bootstrap action at cluster launch to enable log aggregation and to specify the bucket to store the logs.

- 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 3.3 \  
--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-en  
able=true","-y","yarn.log-aggregation.retain-seconds=-1","-y","yarn.log-  
aggregation.retain-check-interval-seconds=3000","-y","yarn.nodemanager.re  
mote-app-log-dir=s3://mybucket/logs"] \  
--instance-type m3.xlarge --instance-count 3
```

- Windows users:

```
aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applica  
tions Name=Hue Name=Hive Name=Pig --use-default-roles --ec2-attributes  
KeyName=myKey --bootstrap-action Path=s3://elasticmapreduce/bootstrap-ac  
tions/configure-hadoop,Name="aggregate logs",Args=["-y","yarn.log-aggrega  
tion-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 is a graphical user interface that you can use to browse the log files from the console. When you enable debugging on a cluster, Amazon EMR archives the log files to Amazon S3 and then indexes those files. You can then use the graphical interface to browse the step, job, task, and task attempt logs for the cluster in an intuitive way.

To be able to use the graphical debugging tool, you must enable debugging when you launch the cluster. You can do this using the console, the CLI, or the API.

To enable the debugging tool using the Amazon EMR console

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Choose **Create cluster**.
3. 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 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, go to the [Amazon SimpleDB](#) product description page.

Note

Debugging is only supported in regions where Amazon SimpleDB is available.

6. Proceed with creating the cluster as described in [Plan an Amazon EMR Cluster](#) (p. 30).

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:

```
aws emr create-cluster --name "Test cluster" --ami-version 3.3 --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:

```
aws emr create-cluster --name "Test cluster" --ami-version 3.3 --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
```

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>.

Select an Amazon VPC Subnet for the Cluster (Optional)

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

Amazon Virtual Private Cloud (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.

The reasons to launch your cluster into a VPC include the following:

- **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.

- **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 to 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.

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

Topics

- [Clusters in a VPC \(p. 206\)](#)
- [Setting Up a VPC to Host Clusters \(p. 207\)](#)
- [Launching Clusters into a VPC \(p. 209\)](#)
- [Restricting Permissions to a VPC Using IAM \(p. 210\)](#)

Clusters in a VPC

There are two platforms on which you can launch the EC2 instances of your cluster: EC2-Classic and EC2-VPC. In EC2-Classic, your instances run in a single, flat network that you share with other customers. In EC2-VPC, your instances run in a virtual private cloud (VPC) that's logically isolated to your AWS account. Your AWS account is capable of launching clusters into either the EC2-Classic or EC2-VPC platform, or only into the EC2-VPC platform, on a region-by-region basis. For more information about EC2-VPC, see [Amazon Virtual Private Cloud \(Amazon VPC\)](#).

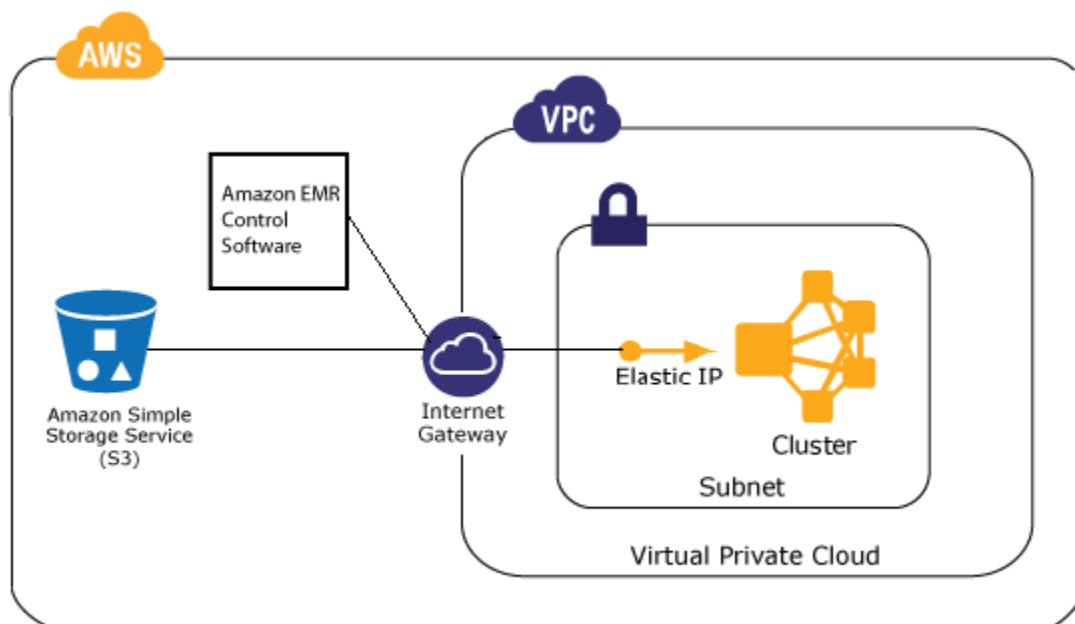
Because access to and from the AWS cloud is a requirement of the cluster, you must connect an Internet gateway to the subnet hosting the cluster. If your application has components you do not want connected to the Internet gateway, you can launch those components in a private subnet you create within your VPC.

Clusters running in a VPC uses two security groups, ElasticMapReduce-master and ElasticMapReduce-slave, which control access to the master and slave nodes. Both the slave and master nodes connect to Amazon S3 through the Internet gateway.

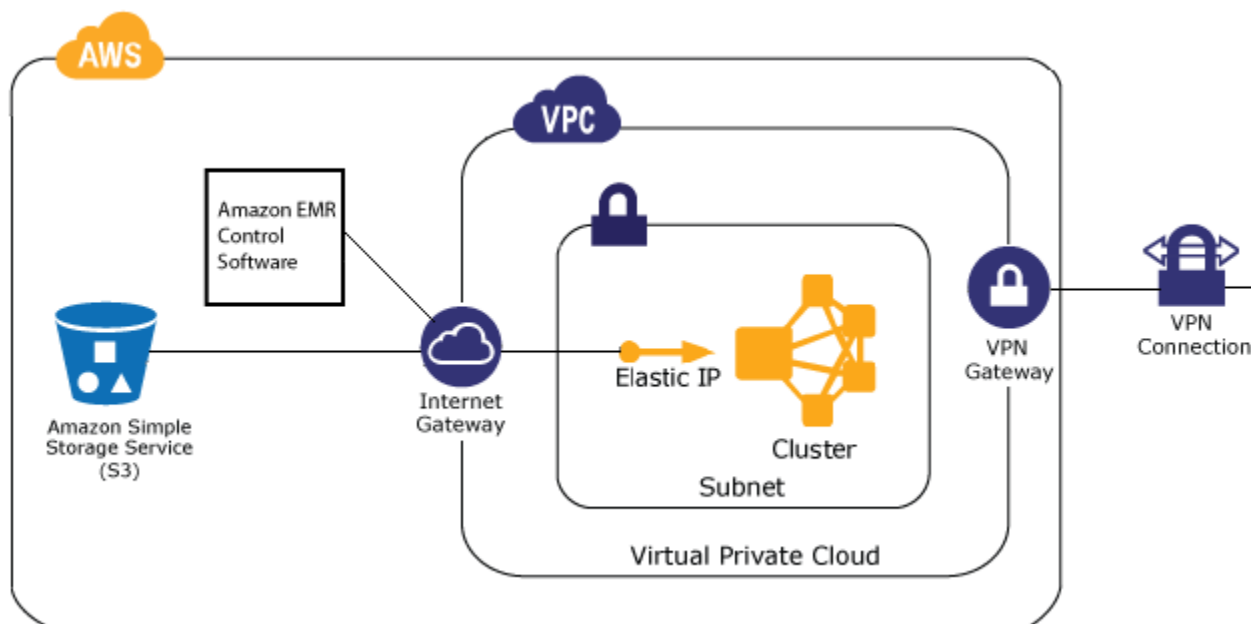
Note

When you launch instances in a VPC using Amazon EMR, several Elastic IP addresses are loaned to you by the system at no cost; however, these addresses are not visible because they are not created using your account.

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



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



Setting Up a VPC to Host Clusters

Before you can launch clusters on a VPC, you must create a VPC, a subnet, and an Internet gateway. The following instructions describe how to create a VPC capable of hosting Amazon EMR clusters using the Amazon EMR console.

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 where you'll be running your cluster.
3. Create a VPC by clicking **Start VPC Wizard**.
4. Choose the VPC configuration by selecting one of the following options:
 - **VPC with a Single Public Subnet** - Select this option if the data used in the cluster is available on the Internet (for example, in Amazon S3 or Amazon RDS).
 - **VPC with Public and Private subnets and Hardware VPN Access** - Select this option if the data used in the cluster is stored in your network (for example, an Oracle database).
5. Confirm the VPC settings.

Step 2: VPC with a Single Public Subnet

IP CIDR block:*	<input type="text" value="10.0.0.0/16"/>	(65531 IP addresses available)
VPC name:	<input type="text" value="My VPC"/>	
Public subnet:*	<input type="text" value="10.0.0.0/24"/>	(251 IP addresses available)
Availability Zone:*	<input type="text" value="No Preference"/>	
Subnet name:	<input type="text" value="Public subnet"/>	
You can add more subnets after AWS creates the VPC.		
Enable DNS hostnames:*	<input checked="" type="radio"/> Yes <input type="radio"/> No	
Hardware tenancy:*	<input type="text" value="Default"/>	
		Cancel and Exit Back Create VPC

- To work with Amazon EMR, the VPC 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
- 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 click **Edit** in the details pane. To create a DNS entry that does not include a domain name, you need to create a **DNS 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. Click **Create VPC**. A dialog box confirms that the VPC has been successfully created. Click **Close**.

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

Launching Clusters into a VPC

After you have a subnet that is configured to host Amazon EMR clusters, launching clusters on that subnet is as simple as specifying the subnet identifier during the cluster creation.

If the subnet does not have an Internet gateway, the cluster creation fails with the error: `Subnet not correctly configured, missing route to an Internet gateway`.

When the cluster is launched, Amazon EMR adds two security groups to the VPC: `ElasticMapReduce-slave` and `ElasticMapReduce-master`. By default, the `ElasticMapReduce-master` security group allows inbound SSH connections. The `ElasticMapReduce-slave` group does not. If you require SSH access for slave (core and task) nodes, you can add a rule to the `ElasticMapReduce-slave` security group. For more information about modifying security group rules, see [Adding Rules to a Security Group](#) in the *Amazon EC2 User Guide for Linux Instances*.

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 [DescribeInstances](#). If you disconnect this device, the cluster will fail.

After the cluster is created, it is able to access AWS services to connect to data stores, such as Amazon S3.

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, in the **Network** field, choose the ID of a VPC network you created previously. When you choose a VPC, the **EC2 Availability Zone** field becomes an **EC2 Subnet** field.

For more information, see [What is Amazon VPC?](#)

4. In the **EC2 Subnet** field, choose the ID of a subnet that you created previously.
5. Proceed with creating the cluster as described in [Plan an Amazon EMR Cluster \(p. 30\)](#).

To launch a cluster into a VPC using the AWS CLI

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

- To create a cluster in a specific subnet, type the following command, replace *myKey* with the name of your EC2 key pair, and replace *77XXXX03* with your subnet ID.
 - 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,SubnetId=subnet-77XXXX03 \
--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 --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 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/>.

Restricting 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 how IAM works with Amazon EMR, see [IAM User Guide](#).

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

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

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

To allow users to launch clusters in a VPC without the ability to modify the VPC

1. Create the VPC and launch Amazon EMR into a subnet of that VPC using an account with permissions to administer Amazon VPC and Amazon EMR.
2. Create a second user account with permissions to call the `RunJobFlow`, `DescribeJobFlows`, `TerminateJobFlows`, and `AddJobFlowStep` actions in the Amazon EMR API. You should also create an IAM policy that allows this user to launch EC2 instances. An example of this is shown below.

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Action": [  
        "ec2:AuthorizeSecurityGroupIngress",  
        "ec2:CancelSpotInstanceRequests",  
        "ec2:CreateSecurityGroup",  
        "ec2:CreateTags",  
        "ec2:DescribeAvailabilityZones",  
        "ec2:DescribeInstances",  
        "ec2:DescribeKeyPairs",  
        "ec2:DescribeSubnets",
```

```
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSpotInstanceRequests",
        "ec2:DescribeRouteTables",
        "ec2:ModifyImageAttribute",
        "ec2:ModifyInstanceAttribute",
        "ec2:RequestSpotInstances",
        "ec2:RunInstances",
        "ec2:TerminateInstances"
    ],
    "Effect": "Allow",
    "Resource": "*"
},
{
    "Action": [
        "elasticmapreduce:AddInstanceGroups",
        "elasticmapreduce:AddJobFlowSteps",
        "elasticmapreduce:DescribeJobFlows",
        "elasticmapreduce:ModifyInstanceGroups",
        "elasticmapreduce:RunJobFlow",
        "elasticmapreduce:TerminateJobFlows"
    ],
    "Effect": "Allow",
    "Resource": "*"
},
{
    "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:ListBucket"
    ],
    "Effect": "Allow",
    "Resource": "*"
}
]
```

Users with the IAM permissions set above are able to launch clusters within the VPC subnet, but are not able to change the VPC configuration.

Note

You should be cautious when granting `ec2:TerminateInstances` permissions because this action gives the recipient the ability to shut down any EC2 instance in the account, including those outside of Amazon EMR.

Tagging Amazon EMR Clusters

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

It can be convenient to categorize your AWS resources in different ways; for example, by purpose, owner, or environment. You can achieve this in Amazon EMR by assigning custom metadata to your Amazon EMR clusters using tags. A tag consists of a key and a value, both of which you define. For Amazon EMR, the cluster is the resource-level that you can tag. For example, you could define a set of tags for your account's clusters that helps you track each cluster's owner or identify a production cluster versus

a testing cluster. We recommend that you create a consistent set of tags to meet your organization requirements.

When you add a tag to an Amazon EMR cluster, the tag is also propagated to each active Amazon EC2 instance associated with the cluster. Similarly, when you remove a tag from an Amazon EMR cluster, that tag is removed from each associated active Amazon EC2 instance.

Important

Use the Amazon EMR console or CLI to manage tags on Amazon EC2 instances that are part of a cluster instead of the Amazon EC2 console or CLI, because changes that you make in Amazon EC2 do not synchronize back to the Amazon EMR tagging system.

You can identify an Amazon EC2 instance that is part of an Amazon EMR cluster by looking for the following system tags. In this example, *CORE* is the value for the instance group role and *j-12345678* is an example job flow (cluster) identifier value:

- `aws:elasticmapreduce:instance-group-role=CORE`
- `aws:elasticmapreduce:job-flow-id=j-12345678`

Note

Amazon EMR and Amazon EC2 interpret your tags as a string of characters with no semantic meaning.

You can work with tags using the AWS Management Console, the CLI, and the API.

You can add tags when creating a new Amazon EMR cluster and you can add, edit, or remove tags from a running Amazon EMR cluster. Editing a tag is a concept that applies to the Amazon EMR console, however using the CLI and API, to edit a tag you remove the old tag and add a new one. You can edit tag keys and values, and you can remove tags from a resource at any time a cluster is running. However, you cannot add, edit, or remove tags from a terminated cluster or terminated instances which were previously associated with a cluster that is still active. In addition, you can set a tag's value to the empty string, but you can't set a tag's value to null.

If you're using AWS Identity and Access Management (IAM) with your Amazon EC2 instances for resource-based permissions by tag, your IAM policies are applied to tags that Amazon EMR propagates to a cluster's Amazon EC2 instances. For Amazon EMR tags to propagate to your Amazon EC2 instances, your IAM policy for Amazon EC2 needs to allow permissions to call the Amazon EC2 `CreateTags` and `DeleteTags` APIs. Also, propagated tags can affect your Amazon EC2's resource-based permissions. Tags propagated to Amazon EC2 can be read as conditions in your IAM policy, just like other Amazon EC2 tags. Keep your IAM policy in mind when adding tags to your Amazon EMR clusters to avoid IAM users having incorrect permissions for a cluster. To avoid problems, make sure that your IAM policies do not include conditions on tags that you also plan to use on your Amazon EMR clusters. For more information, see [Controlling Access to Amazon EC2 Resources](#).

Tag Restrictions

The following basic restrictions apply to tags:

- The maximum number of tags per resource is 10.
- The maximum key length is 127 Unicode characters.
- The maximum value length is 255 Unicode characters.
- Tag keys and values are case sensitive.
- Do not use the `aws:` prefix in tag names and values because it is reserved for AWS use. In addition, you cannot edit or delete tag names or values with this prefix.
- You cannot change or edit tags on a terminated cluster.
- A tag value can be an empty string, but not null. In addition, a tag key cannot be an empty string.

For more information about tagging using the AWS Management Console, see [Working with Tags in the Console](#) in the *Amazon EC2 User Guide for Linux Instances*. For more information about tagging using the Amazon EC2API or command line, see [API and CLI Overview](#) in the *Amazon EC2 User Guide for Linux Instances*.

Tagging Resources for Billing

You can use tags for organizing your AWS bill to reflect your own cost structure. To do this, sign up to get your AWS account bill with tag key values included. You can then organize your billing information by tag key values, to see the cost of your combined resources. Although Amazon EMR and Amazon EC2 have different billing statements, the tags on each cluster are also placed on each associated instance so you can use tags to link related Amazon EMR and Amazon EC2 costs.

For example, you can tag several resources with a specific application name, and then organize your billing information to see the total cost of that application across several services. For more information, see [Cost Allocation and Tagging](#).

Adding Tags to a New Cluster

You can add tags to a cluster while you are creating it.

To add tags when creating a new cluster using the console

1. In the Amazon EMR console, select the **Cluster List** page and click **Create cluster**.
2. On the **Create Cluster** page, in the **Tags** section, click the empty field in the **Key** column and type the name of your key.

When you begin typing the new tag, another tag line automatically appears to provide space for the next new tag.

3. Optionally, click the empty field in the **Value** column and type the name of your value.
4. Repeat the previous steps for each tag key/value pair to add to the cluster. When the cluster launches, any tags you enter are automatically associated with the cluster.

To add tags when creating a new cluster using the AWS CLI

The following example demonstrates how to add a tag to a new cluster using the AWS CLI. To add tags when you create a cluster, type the `create-cluster` subcommand with the `--tags` parameter.

- To add a tag named `costCenter` with key value `marketing` when you create a cluster, type the following command and replace `myKey` with the name of your EC2 key pair.
 - 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 Key
Name=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-de
```

```
fault-roles --ec2-attributes KeyName=myKey --instance-type m3.xlarge -  
-instance-count 3
```

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

Note

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

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

Adding Tags to an Existing Cluster

You can also add tags to an existing cluster.

To add tags to an existing cluster using the console

1. In the Amazon EMR console, select the **Cluster List** page and click a cluster to which to add tags.
2. On the **Cluster Details** page, in the **Tags** field, click **View All/Edit**.
3. On the **View All/Edit** page, click **Add**.
4. Click the empty field in the **Key** column and type the name of your key.
5. Optionally, click the empty field in the **Value** column and type the name of your value.
6. With each new tag you begin, another empty tag line appears under the tag you are currently editing. Repeat the previous steps on the new tag line for each tag to add.

To add tags to a running cluster using the AWS CLI

The following example demonstrates how to add tags to a running cluster using the AWS CLI. Type the `add-tags` subcommand with the `--tag` parameter to assign tags to a resource identifier (cluster ID). The resource ID is the cluster identifier available via the console or the `list-clusters` command.

Note

The `add-tags` subcommand currently accepts only one resource ID.

- To add two tags to a running cluster (one with a key named `production` with no value and the other with a key named `costCenter` with a value of `marketing`) type the following command and replace `j-KT4XXXXXXXX1NM` with your cluster ID.

```
aws emr add-tags --resource-id j-KT4XXXXXXXX1NM --tag "costCenter=marketing"  
--tag "other=accounting"
```

Note

When tags are added using the AWS CLI, there is no output from the command.

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

Viewing Tags on a Cluster

If you would like to see all the tags associated with a cluster, you can view them in the console or at the CLI.

To view the tags on a cluster using the console

1. In the Amazon EMR console, select the **Cluster List** page and click a cluster to view tags.
2. On the **Cluster Details** page, in the **Tags** field, some tags are displayed here. Click **View All/Edit** to display all available tags on the cluster.

To view the tags on a cluster using the AWS CLI

To view the tags on a cluster using the AWS CLI, type the `describe-cluster` subcommand with the `--query` parameter.

- To view a cluster's tags, type the following command and replace `j-KT4XXXXXXXX1NM` with your cluster ID.

```
aws emr describe-cluster --cluster-id j-KT4XXXXXXXX1NM --query Cluster.Tags
```

The output displays all the tag information about the cluster similar to the following:

```
Value: accounting      Value: marketing
Key: other             Key: costCenter
```

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

Removing Tags from a Cluster

If you no longer need a tag, you can remove it from the cluster.

To remove tags from a cluster using the console

1. In the Amazon EMR console, select the **Cluster List** page and click a cluster from which to remove tags.
2. On the **Cluster Details** page, in the **Tags** field, click **View All/Edit**.
3. In the **View All/Edit** dialog box, click the **X** icon next to the tag to delete and click **Save**.
4. (Optional) Repeat the previous step for each tag key/value pair to remove from the cluster.

To remove tags from a cluster using the AWS CLI

To remove tags from a cluster using the AWS CLI, type the `remove-tags` subcommand with the `--tag-keys` parameter. When removing a tag, only the key name is required.

- To remove a tag from a cluster, type the following command and replace `j-KT4XXXXXXXX1NM` with your cluster ID.

```
aws emr remove-tags --resource-id j-KT4XXXXXXXX1NM --tag-keys "costCenter"
```


Note

You cannot currently remove multiple tags using a single command.

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

Use Third Party Applications With Amazon EMR (Optional)

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 \(p. 216\)](#)
- [Use Hunk with Amazon EMR \(p. 216\)](#)
- [Parse Data with HParser \(p. 217\)](#)
- [Using the MapR Distribution for Hadoop \(p. 218\)](#)

Use Business Intelligence Tools with Amazon EMR

You can use popular business intelligence tools like Microsoft Excel, MicroStrategy, QlikView, and Tableau with Amazon EMR to explore and visualize your data. Many of these tools require an ODBC (Open Database Connectivity) or JDBC (Java Database Connectivity) driver. You can download and install the necessary drivers from the links below:

- <http://amazon-odbc-jdbc-drivers.s3.amazonaws.com/public/HiveJDBC.zip>
- <http://amazon-odbc-jdbc-drivers.s3.amazonaws.com/public/HiveODBC.zip>
- <http://amazon-odbc-jdbc-drivers.s3.amazonaws.com/public/ImpalaJDBC.zip>
- <http://amazon-odbc-jdbc-drivers.s3.amazonaws.com/public/ImpalaODBC.zip>
- <http://amazon-odbc-jdbc-drivers.s3.amazonaws.com/public/HBaseODBC.zip>

For more information about how you would connect a business intelligence tool like Microsoft Excel to Hive, go to <http://www.simba.com/wp-content/uploads/2013/05/Simba-Apache-Hive-ODBC-Driver-Quickstart.pdf>.

Use Hunk with Amazon EMR

With [Hunk](#) you can interactively explore, analyze, and visualize data stored in Amazon EMR and Amazon S3. Hunk is a full-featured, integrated analytics platform that lets everyone in your organization interactively explore, analyze, and visualize big data regardless of where the data is stored. Simply point Hunk at your data to start exploring and visualizing patterns. Create dashboards and share reports—all from one integrated platform—in minutes. For more information on using Hunk and current pricing, see [Hunk documentation](#) and [Amazon EMR with Hunk](#).

To enable Hunk on an Amazon EMR cluster using the console

1. Open the Amazon Elastic MapReduce console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Click **Create cluster**.
3. In the **Software Configuration** section, for **Hadoop distribution**, click **Amazon** and for **AMI version**, choose **3.0.0** (or higher) from the drop-down list.
4. For **Additional applications**, choose **Hunk** from the drop-down list and in the pop-up dialog, click **Enable Hunk**.
5. Proceed with creating the cluster as described in [Plan an Amazon EMR Cluster \(p. 30\)](#).
6. Setup a Hunk Amazon EC2 instance following the instructions, [Install Hunk with Amazon EMR](#).

To enable Hunk on an Amazon EMR cluster using the AWS CLI

1. Type the following command to enable Hunk on an Amazon EMR cluster:

```
aws emr create-cluster --applications Name=Hunk Name=Hive Name=Pig --ami-  
version 3.2.1 \  
--instance-groups InstanceGroupType=MASTER, InstanceCount=1, Instance  
Type=m3.xlarge InstanceGroupType=CORE, InstanceCount=2, InstanceType=m3.xlarge  
\  
--no-auto-terminate
```

2. Setup a Hunk Amazon EC2 instance following the instructions, [Install Hunk with Amazon EMR](#).

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 Elastic MapReduce (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 a step-by-step tutorial of 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.

Important

If you launch a MapR job flow in a VPC, you must set `enableDnsHostnames` to true and all hosts in the cluster must have a fully qualified domain name. For more information, see [Using DNS with Your VPC](#) and [DHCP Options Sets](#).

Field	Action
Hadoop distribution	Choose MapR . This determines which version of Hadoop to run on your cluster. For more information about the MapR distribution for Hadoop, see Using the MapR Distribution for Hadoop (p. 218) .
Edition	Choose the MapR edition that meets your requirements. For more information, see Using the MapR Distribution for Hadoop (p. 218) .
Version	Choose the MapR version that meets your requirements. For more information, see Versions (p. 220) .
Arguments (Optional)	Specify any additional arguments to pass to MapR to meet your requirements. For more information, see Arguments (p. 220) .

- Click **Done** and proceed with creating the cluster as described in [Plan an Amazon EMR Cluster \(p. 30\)](#).

Launch a Cluster with MapR Using the AWS CLI

To launch an Amazon EMR cluster with MapR using the AWS CLI

Note

You cannot yet launch MapR clusters when specifying the `--release-label` parameter.

To launch a cluster with MapR using the AWS CLI, type the `create-cluster` command with the `--applications` parameter.

- To launch a cluster with MapR, m7 edition, type the following command and replace *myKey* with the name of your EC2 key pair.

- Linux, UNIX, and Mac OS X users:

```
aws emr create-cluster --name "MapR cluster" --applications Name=Hive
Name=Pig \
Name=MapR,Args=--edition,m7,--version,4.0.2 --ami-version 3.3.2 \
--use-default-roles --ec2-attributes KeyName=myKey \
--instance-type m1.xlarge --instance-count 3
```

- Windows users:

```
aws emr create-cluster --name "MapR cluster" --applications Name=Hive
Name=Pig Name=MapR,Args=--edition,m7,--version,3.1.1 --ami-version 2.4 -
--use-default-roles --ec2-attributes KeyName=myKey --instance-type m1.xlarge
--instance-count 3
```

Note

The version of MapR available in Amazon EMR that supports Hadoop 2.x is 4.0.2, and it is only available with Amazon EMR AMI 3.3.2

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

Note

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

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

MapR Editions, Versions, and Arguments

MapR supports several editions, versions, and arguments that you can pass to it using the Amazon EMR console or CLI. The following examples use the Amazon EMR CLI, however Amazon EMR provides an equivalent for each that you can use with the console.

For more information about launching clusters using the CLI, see the instructions for each cluster type in [Plan an Amazon EMR Cluster \(p. 30\)](#). For more information about launching clusters using the API, see [Use SDKs to Call Amazon EMR APIs \(p. 518\)](#).

Editions

Using the CLI, you can pass the `--edition` parameter to specify the following editions:

- m3
- m5
- m7

The default edition is **m3** if not specified.

Versions

You can use `--version` to specify the following versions:

- M3 Edition - 4.0.2, 3.1.1, 3.0.3, 3.0.2, 2.1.3
- M5 Edition - 4.0.2, 3.1.1, 3.0.3, 3.0.2, 2.1.3
- M7 Edition - 4.0.2, 3.1.1, 3.0.3, 3.0.2

Note

MapR version 4.0.2 is available with AMI version 3.3.2 only.

Arguments

The `--supported-product` option supports optional arguments. Amazon EMR accepts and forwards the argument list to the corresponding installation script as bootstrap action arguments. The following example shows the syntax for using the `--supported-product` option with or without optional arguments. The fields `key1` and `value1`, etc. represent custom key/value pairs that MapR recognizes and the key named `edition` is shown among them as an example:

```
--supported-product mapr-m3
--supported-product mapr-m3 --args "--key1,value1,--key2,value2"
--supported-product mapr --args "--edition,m3,--key1,value1,--key2,value2"
```

```
--supported-product mapr-m3 --args "--edition,m3,--key1,value1,--key2,value2"
```

The following table lists the parameters that MapR reads when users specify them with the `--args` option.

Parameter	Description
<code>--edition</code>	The MapR edition.
<code>--version</code>	The MapR version.
<code>--owner-password</code>	The password for the Hadoop owner. The default is <code>hadoop</code> .
<code>--num-mapr-masters</code>	Any additional master nodes for m5/m7 clusters.
<code>--mfs-percentage</code>	The proportion of space from the attached storage volumes to be used for MapR file system; the default is 100 (all available space); the minimum is 50. Users who are doing large copies in and out of Amazon S3 storage using <code>s3distcp</code> or another such mechanism may see faster performance by allowing some of the storage space to be used as regular Linux storage. For example: <code>--mfs-percentage,90</code> . The remainder is mounted to <code>S3_TMP_DIR</code> as RAID0 file system.
<code>--hive-version</code>	The version of the Amazon Hive package. The default is <code>latest</code> . To disable the installation of Amazon Hive, use <code>--hive-version,none</code> .
<code>--pig-version</code>	The version of the Amazon Pig package. The default is <code>latest</code> . To disable the installation of Amazon Pig, use <code>--pig-version,none</code> .

The following table shows examples of commands, including arguments with the `--args` parameter, and which command MapR executes after interpreting the input.

Options	Commands Processed by MapR
<code>--supported-product mapr</code>	<code>--edition m3</code>
<code>--supported-product mapr-m5</code>	<code>--edition m5</code>
<code>--supported-product mapr-m3</code>	<code>--edition m3</code>
<code>--with-supported-products mapr-m3 (deprecated)</code>	<code>--edition m3</code>
<code>--with-supported-products mapr-m5 (deprecated)</code>	<code>--edition m5</code>
<code>--supported-product mapr-m5 --args "--version,1.1"</code>	<code>--edition m5 --version 1.1</code>
<code>--supported-product mapr-m5 --args "--edition,m3"</code>	N/A - returns an error
<code>--supported-product mapr --args "--edition,m5"</code>	<code>--edition m5</code>
<code>--supported-product mapr --args "--version,1.1"</code>	<code>--edition m3 --version 1.1</code>

Options	Commands Processed by MapR
--supported-product mapr --args "--edition,m5,--key1 value1"	--edition m5 --key1 value1

Note

The `--with-supported-products` option is deprecated and replaced by the `--supported-product` option, which provides the same Hadoop and MapR versions with added support for parameters.

Enabling MCS access for your Amazon EMR Cluster

After your MapR cluster is running, you need to open port 8453 to enable access to the MapR Control System (MCS) from hosts other than the host that launched the cluster. Follow these steps to open the port.

To open a port for MCS access

1. Select your job from the list of jobs displayed in **Your Elastic MapReduce Job Flows** in the **Elastic MapReduce** tab of the AWS Management Console, then select the **Description** tab in the lower pane. Make a note of the **Master Public DNS Name** value.
2. Click the **Amazon EC2** tab in the AWS Management Console to open the Amazon EC2 console.
3. In the navigation pane, select **Security Groups** under the **Network and Security** group.
4. In the **Security Groups** list, select **Elastic MapReduce-master**.
5. In the lower pane, click the **Inbound** tab.
6. In **Port Range:**, type 8453. Leave the default value in the **Source:** field.

Note

The standard MapR port is 8443. Use port number 8453 instead of 8443 when you use the MapR REST API calls to MapR on an Amazon EMR cluster.

7. Click **Add Rule**, then click **Apply Rule Changes**.
8. You can now navigate to the master node's DNS address. Connect to port 8453 to log in to the MapR Control System. Use the string `hadoop` for both login and password at the MCS login screen.

Note

For M5 MapR clusters on Amazon EMR, the MCS web server runs on the primary and secondary CLDB nodes, giving you another entry point to the MCS if the primary fails.

Testing Your Cluster

This section explains how to test your cluster by performing a word count on a sample input file.

To create a file and run a test job

1. Connect to the master node with SSH as user `hadoop`. Pass your `.pem` credentials file to `ssh` with the `-i` flag, as in this example:

```
ssh -i /path_to_pemfile/credentials.pem hadoop@masterDNS.amazonaws.com
```

2. Create a simple text file:

```
cd /mapr/MapR_EMR.amazonaws.com
mkdir in
echo "the quick brown fox jumps over the lazy dog" > in/data.txt
```

3. Run the following command to perform a word count on the text file:

```
hadoop jar /opt/mapr/hadoop/hadoop-0.20.2/hadoop-0.20.2-dev-examples.jar
wordcount /mapr/MapR_EMR.amazonaws.com/in/ /mapr/MapR_EMR.amazonaws.com/out/
```

As the job runs, you should see terminal output similar to the following:

```
12/06/09 00:00:37 INFO fs.JobTrackerWatcher: Current running JobTracker is:
ip10118194139.ec2.internal/10.118.194.139:9001
12/06/09 00:00:37 INFO input.FileInputFormat: Total input paths to process
: 1
12/06/09 00:00:37 INFO mapred.JobClient: Running job: job_201206082332_0004
12/06/09 00:00:38 INFO mapred.JobClient: map 0% reduce 0%
12/06/09 00:00:50 INFO mapred.JobClient: map 100% reduce 0%
12/06/09 00:00:57 INFO mapred.JobClient: map 100% reduce 100%
12/06/09 00:00:58 INFO mapred.JobClient: Job complete: job_201206082332_0004
12/06/09 00:00:58 INFO mapred.JobClient: Counters: 25
12/06/09 00:00:58 INFO mapred.JobClient: Job Counters
12/06/09 00:00:58 INFO mapred.JobClient: Launched reduce tasks=1
12/06/09 00:00:58 INFO mapred.JobClient: Aggregate execution time of map
pers(ms)=6193
12/06/09 00:00:58 INFO mapred.JobClient: Total time spent by all reduces
waiting after reserving slots (ms)=0
12/06/09 00:00:58 INFO mapred.JobClient: Total time spent by all maps waiting
after reserving slots (ms)=0
12/06/09 00:00:58 INFO mapred.JobClient: Launched map tasks=1
12/06/09 00:00:58 INFO mapred.JobClient: Datalocalmap tasks=1
12/06/09 00:00:58 INFO mapred.JobClient: Aggregate execution time of redu
cers(ms)=4875
12/06/09 00:00:58 INFO mapred.JobClient: FileSystemCounters
12/06/09 00:00:58 INFO mapred.JobClient: MAPRFS_BYTES_READ=385
12/06/09 00:00:58 INFO mapred.JobClient: MAPRFS_BYTES_WRITTEN=276
12/06/09 00:00:58 INFO mapred.JobClient: FILE_BYTES_WRITTEN=94449
12/06/09 00:00:58 INFO mapred.JobClient: MapReduce Framework
12/06/09 00:00:58 INFO mapred.JobClient: Map input records=1
12/06/09 00:00:58 INFO mapred.JobClient: Reduce shuffle bytes=94
12/06/09 00:00:58 INFO mapred.JobClient: Spilled Records=16
12/06/09 00:00:58 INFO mapred.JobClient: Map output bytes=80
12/06/09 00:00:58 INFO mapred.JobClient: CPU_MILLISECONDS=1530
12/06/09 00:00:58 INFO mapred.JobClient: Combine input records=9
12/06/09 00:00:58 INFO mapred.JobClient: SPLIT_RAW_BYTES=125
12/06/09 00:00:58 INFO mapred.JobClient: Reduce input records=8
12/06/09 00:00:58 INFO mapred.JobClient: Reduce input groups=8
12/06/09 00:00:58 INFO mapred.JobClient: Combine output records=8
12/06/09 00:00:58 INFO mapred.JobClient: PHYSICAL_MEMORY_BYTES=329244672
12/06/09 00:00:58 INFO mapred.JobClient: Reduce output records=8
12/06/09 00:00:58 INFO mapred.JobClient: VIRTUAL_MEMORY_BYTES=3252969472
12/06/09 00:00:58 INFO mapred.JobClient: Map output records=9
12/06/09 00:00:58 INFO mapred.JobClient: GC time elapsed (ms)=1
```


4. Check the `/mapr/MapR_EMR.amazonaws.com/out` directory for a file named `part-r-00000` with the results of the job.

```
cat out/partr00000
brown 1
dog 1
fox 1
jumps 1
lazy 1
over 1
quick 1
the 2
```

Tutorial: Launch an Amazon EMR Cluster with MapR M7

This tutorial guides you through launching an Amazon EMR cluster featuring the M7 edition of the [MapR distribution for Hadoop](#). The MapR distribution for Hadoop is a complete Hadoop distribution that provides many unique capabilities, such as industry-standard NFS and ODBC interfaces, end-to-end management, high reliability, and automatic compression. You can manage a MapR cluster through the web-based [MapR Control System](#), the command line, or a REST API. M7 provides enterprise-grade capabilities such as high availability, [snapshot](#) and [mirror volumes](#), and native MapR table functionality on MapR-FS, enabling responsive HBase-style flat table databases compatible with snapshots and mirroring. It provides a single platform for storing and processing unstructured and structured data, integrated with existing infrastructure, applications, and tools.

Note

To use the commands in this tutorial, download and install the AWS CLI. For more information see [Installing the AWS CLI](#) in the *AWS Command Line Interface User Guide*.

1. To launch a cluster with MapR, m7 edition, type the following command and replace *myKey* with the name of your EC2 key pair.

- Linux, UNIX, and Mac OS X users:

```
aws emr create-cluster --name "MapR cluster" --applications Name=Hive
Name=Pig \
Name=MapR,Args=--edition,m7,--version,4.0.2 --ami-version 3.3.2 \
--use-default-roles --ec2-attributes KeyName=myKey \
--instance-type m1.xlarge --instance-count 3
```

- Windows users:

```
aws emr create-cluster --name "MapR cluster" --applications Name=Hive
Name=Pig Name=MapR,Args=--edition,m7,--version,3.1.1 --ami-version 2.4 -
--use-default-roles --ec2-attributes KeyName=myKey --instance-type m1.xlarge
--instance-count 3
```

Note

The versions of MapR available in Amazon EMR do not currently support Hadoop 2.x. When specifying the `--ami-version`, use a Hadoop 1.x AMI.

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

Note

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

After you run the command, the cluster takes between five and ten minutes to start. The `aws emr list-clusters` command shows your cluster in the STARTING and BOOTSTRAPPING states before entering the WAITING state.

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

2. Retrieve the cluster ID and then use SSH to connect to the cluster. In the following commands, replace `j-2AL4XXXXXX5T9` with your cluster ID, and replace `~/mykeypair.key` with the path to and filename of your key pair private key file.

```
aws emr list-clusters

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

Note

For more information about accessing a cluster with SSH, see [Connect to the Master Node Using SSH \(p. 441\)](#).

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. You can create [mappings](#) between the table names used in Apache HBase and M7 Edition's native tables.
 - a. Create a table with the following command:

```
$ echo "create '/tables/user_table', 'family' " | hbase shell
```

- b. List tables with the following command:

```
$ hadoop fs -ls /tables
Found 1 items
trwxr-xr-x 3 root root 2 2013-04-16 22:49 /tables/user_table
```

```
$ ls /mapr/MapR_EMR.amazonaws.com/tables
user_table
```

- c. Move or rename tables using the following command:

```
hadoop fs -mv /tables/user_table /tables/usertable
```

5. A snapshot is a read-only image of a volume at a specific point in time. Snapshots are useful any time you need to be able to roll back to a known good data set at a specific point in time.

- a. From the HBase shell, add a row to the newly-created table:

```
$ hbase shell
hbase(main):001:0> put '/tables/usertable', 'row_1' , 'family:child',
'username'
output: 0 row(s) in 0.0920 seconds
```

- b. Create the snapshot:

```
$ maprcli volume snapshot create -volume tables -snapshotname mysnap
```

- c. Change the table:

```
hbase(main):002:0> put '/tables/usertable', 'row_1' , 'family:location',
'San Jose'
```

- d. Snapshots are stored in a `.snapshot` directory. Scan the table from the snapshot and the current table to see the difference:

```
hbase shell >
scan '/tables/usertable'
ROW COLUMN+CELL
row_1 column=family:child, timestamp=1366154016042, value=username
row_1 column=family:home, timestamp=1366154055043, value=San Jose
1 row(s) in 0.2910 seconds
scan '/tables/.snapshot/mysnap/usertable'
ROW COLUMN+CELL
row_1 column=family:child, timestamp=1366154016042, value=username
1 row(s) in 0.2320 seconds
```

6. Test high availability:

- a. List the current regions on your system.

```
$ maprcli table region list -path /tables/usertable
secondarynodes scans primarynode puts
startkey gets lastheartbeat endkey
ip-10-191-5-21.ec2.internal, ip-10-68-37-140.ec2.internal ... ip-10-4-
```

```
74-175.ec2.internal ...  
-INFINITY ... 0 INFINITY
```

- b. Restart the primary node for one of the regions. Make sure that the primary node is not the access point to the cluster. Restarting your access point will result in loss of cluster access and terminate your YCSB client.

Connect to the cluster with SSH and restart the node with the following command:

```
$ ssh -i /Users/username/downloads/MyKey_Context.pem hadoop@ec2-23-20-  
100-174.compute-1.amazonaws.com  
$ sudo /sbin/reboot
```

Note

The restart will take 15 to 30 seconds to complete.

- c. After the restart is complete, list your new regions to see that the former primary node is now listed as secondary.

```
$ maprcli table region list -path /tables/usertable  
secondarynodes scans primarynode puts  
startkey gets lastheartbeat endkey  
ip-10-191-5-21.ec2.internal, ip-10-68-37-140.ec2.internal ... ip-10-4-  
74-175.ec2.internal ...  
-INFINITY ... 0 INFINITY
```

7. To open the MapR Control System page, navigate to the address <https://hostname.compute-1.amazonaws.com:8453>. The username and password for the default installation are both `hadoop`. The URL for your node's hostname appears in the message-of-the-day that displays when you first log in to the node over SSH.

The Nodes view displays the nodes in the cluster, by rack. The Nodes view contains two panes: the Topology pane and the Nodes pane. The Topology pane shows the racks in the cluster. Selecting a rack displays that rack's nodes in the Nodes pane to the right. Selecting **Cluster** displays all the nodes in the cluster. Clicking any column name sorts data in ascending or descending order by that column. If your YCSB job is still running, you can see the put streams continuing from the Nodes view.

Run a Hadoop Application to Process Data

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

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. 228)
- [JAR Requirements](#) (p. 230)
- [Run a Script in a Cluster](#) (p. 231)
- [Process Data with Streaming](#) (p. 232)
- [Process Data Using Cascading](#) (p. 235)
- [Process Data with a Custom JAR](#) (p. 237)

Build Binaries Using Amazon EMR

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

You can use Amazon Elastic MapReduce (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 [Instance Configurations \(p. 35\)](#). 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

1	Connect to the master node of your cluster.
2	Copy source files to the master node.
3	Build binaries with any necessary optimizations.
4	Copy binaries from the master node to Amazon S3.

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 \(p. 441\)](#).

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, go to 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:

```
mkdir SourceFiles
```

3. Copy your source files from Amazon S3 to the master node by typing a command similar to the following:

```
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:

```
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:

```
master$ uname -a
```

The output looks similar to the following.

```
Linux domU-12-31-39-17-29-39.compute-1.internal 2.6.21.7-2.fc8xen #1 SMP  
Fri Feb 15 12:34:28 EST 2008 x86_64 GNU/Linux
```

- c. To view the processor speed, enter the following command:

```
master$ cat /proc/cpuinfo
```

The output looks similar to the following.

```
processor : 0  
vendor_id : GenuineIntel  
model name : Intel(R) Xeon(R) CPU E5430 @ 2.66GHz  
flags : fpu tsc msr pae mce cx8 apic mca cmov pat pse36 clflush dts acpi  
mmx fxsr sse sse2 ss ht tm syscall nx lm constant_tsc pni monitor ds_cpl  
vmx est tm2 ssse3 cx16 xtpr cda lahf_lm  
...
```

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:

```
hadoop fs -put BinaryFiles s3://mybucket/BinaryDestination
```

JAR Requirements

When you launch an Amazon Elastic MapReduce (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 a success exit code, Amazon EMR assumes the step failed.

Likewise, when your JAR completes, Amazon EMR assumes that all Hadoop 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 Elastic MapReduce (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, go to [Submit Work to a Cluster \(p. 473\)](#).

If you want to run a script before step processing begins, use a bootstrap action. For more information on bootstrap actions, go to [\(Optional\) Create Bootstrap Actions to Install Additional Software \(p. 124\)](#).

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. `Script-runner.jar` is located at `s3://elasticmapreduce/libs/script-runner/script-runner.jar`.

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 Amazon 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 3.3 --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"]
```

- 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 --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 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>.

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 Elastic MapReduce (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 a complete description of Hadoop Streaming, go to <http://hadoop.apache.org/docs/stable/hadoop-streaming/HadoopStreaming.html>.

Using the Hadoop Streaming Utility

This section describes how use to 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.

Parameter	Description	Required
-input	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.	Yes
-output	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 <i>input</i> .	Yes
-mapper	Name of the mapper executable. Type: String Default: None	Yes
-reducer	Name of the reducer executable. Type: String Default: None	Yes
-cacheFile	An Amazon S3 location containing files for Hadoop to copy into your local working directory (primarily to improve performance). Type: String Default: None Constraints: [URI]#[symlink name to create in working directory]	No
-cacheArchive	JAR file to extract into the working directory Type: String Default: None Constraints: [URI]#[symlink directory name to create in working directory]	No
-combiner	Combines results Type: String Default: None Constraints: Java class name	No

The following code sample is a mapper executable written in Python. This script is part of the WordCount sample application.

```
#!/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 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. Once 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**, 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 **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 Amazon 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 Amazon 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. 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.

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.
- 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 \  
--steps Type=STREAMING,Name="Streaming Program",ActionOnFailure=CONTINUE,\  
Args=[--files, pathtoscripts, -mapper, mapperscript, -reducer, reducerscript, aggregate, -input, pathtoinputdata, -output, pathtooutputbucket]
```

- 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 --steps Type=STREAMING,Name="Streaming Program",ActionOnFailure=CONTINUE,Args=[--files, pathtoscripts, -mapper, mapperscript, -reducer, reducerscript, aggregate, -input, pathtoinputdata, -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 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>.

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 \(p. 236\)](#)

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/>.

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

```
aws emr create-cluster --name "Test cluster" --ami-version 2.4 --applica
tions 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=CONTIN
UE, Jar=pathtojarfile, \
```

```
Args=[ "-input", "pathtoinputdata", "-output", "pathtooutputbucket", "arg1", "arg2" ]
```

- 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-type m3.xlarge --instance-count 3 --bootstrap-actions Path=pathtobootstrapsript,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>.

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 on compiling a JAR file, see [Build Binaries Using Amazon EMR \(p. 228\)](#).

For more information about building a Hadoop MapReduce application, go to <http://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html>.

Submit a Custom JAR Step

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/>.
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 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. 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 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:

```
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=CUSTOM_JAR,Name="Custom JAR Step",ActionOnFailure=CONTINUE,Jar=pathtojarfile, \
Args=["pathtoinputdata", "pathtooutputbucket", "arg1", "arg2"]
```

- 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 --steps Type=CUSTOM_JAR,Name="Custom JAR Step",ActionOnFailure=CONTINUE,Jar=pathtojarfile,Args=["pathtoinputdata", "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>.

Hive and Amazon EMR

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 \(p. 240\)](#)
- [Supported Hive Versions \(p. 251\)](#)
- [Submit Hive Work \(p. 260\)](#)
- [Create a Hive Metastore Outside the Cluster \(p. 262\)](#)
- [Use the Hive JDBC Driver \(p. 264\)](#)

How Amazon EMR Hive Differs from Apache Hive

Topics

- [Combine Splits Input Format \(p. 241\)](#)
- [Log files \(p. 241\)](#)
- [Thrift Service Ports \(p. 242\)](#)
- [Hive Authorization \(p. 242\)](#)
- [Hive File Merge Behavior with Amazon S3 \(p. 242\)](#)
- [ACID Transactions and Amazon S3 \(p. 242\)](#)

- [Additional Features of Hive in Amazon EMR \(p. 243\)](#)

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.

Hive Version	Log File Name
0.13.1	<code>hive.log</code> Note Amazon EMR will now use an unversioned <code>hive.log</code> . Minor versions of will all share the same log location as the major version.
0.11.0	<code>hive_0110.log</code> Note Minor versions of Hive 0.11.0, such as 0.11.0.1, share the same log file location as Hive 0.11.0.
0.8.1	<code>hive_081.log</code> Note 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.

Hive Version	Log File Name
0.7.1	hive_07_1.log Note 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.
0.7	hive_07.log
0.5	hive_05.log
0.4	hive.log

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.

Hive Version	Port Number
Hive 0.13.1	10000
Hive 0.11.0	10004
Hive 0.8.1	10003
Hive 0.7.1	10002
Hive 0.7	10001
Hive 0.5	10000

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

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 \(p. 248\)](#).

Topics

- [Write Data Directly to Amazon S3 \(p. 243\)](#)
- [Use Hive to Access Resources in Amazon S3 \(p. 243\)](#)
- [Use Hive to Recover Partitions \(p. 244\)](#)
- [Variables in Hive \(p. 244\)](#)
- [Amazon EMR Hive queries to accommodate partial DynamoDB schemas \(p. 247\)](#)
- [Copy data between DynamoDB tables in different AWS regions \(p. 248\)](#)
- [Set DynamoDB throughput values per table \(p. 248\)](#)
- [Hive Patches \(p. 248\)](#)

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 Elastic MapReduce console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Click **Create cluster**.
3. In the **Steps** section, for **Add Step**, choose **Hive Program** from the list and click **Configure and add**.
4. In the **Add Step** dialog, specify the parameters using the following table as a guide, and then click **Add**.

Field	Action
Script S3 location*	Specify the URI where your script resides in Amazon S3. The value must be in the form <i>BucketName/path/ScriptName</i> . For example: <code>s3://elasticmapreduce/samples/hive-ads/libs/response-time-stats.q</code> .
Input S3 location	Optionally, specify the URI where your input files reside in Amazon S3. The value must be in the form <i>BucketName/path/</i> . 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> .
Output S3 location	Optionally, specify the URI where you want the output stored in Amazon S3. The value must be in the form <i>BucketName/path</i> . 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> .
Arguments	<p>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:</p> <pre>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';</pre> <p>To pass a value for the variable, type the following in the Arguments window:</p> <pre>-d SAMPLE=s3://elasticmapreduce/samples/hive-ads/.</pre>
Action on Failure	<p>This determines what the cluster does in response to any errors. The possible values for this setting are:</p> <ul style="list-style-type: none"> • 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}`.
- 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 \
--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 3.3 --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,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/]
```

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 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](#) in the *AWS SDK for Java API Reference*.

```
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

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 record
TimeStamp < "2012-01-01";
```

The following table shows the query syntax for selecting any combination of items from DynamoDB.

Query Example	Result Description
SELECT * FROM <i>table_name</i> ;	Selects all items (rows) from a given table and includes data from all columns available for those items.
SELECT * FROM <i>table_name</i> WHERE <i>field_name</i> = <i>value</i> ;	Selects some items (rows) from a given table and includes data from all columns available for those items.
SELECT <i>column1_name</i> , <i>column2_name</i> , <i>column3_name</i> FROM <i>table_name</i> ;	Selects all items (rows) from a given table and includes data from some columns available for those items.
SELECT <i>column1_name</i> , <i>column2_name</i> , <i>column3_name</i> FROM <i>table_name</i> WHERE <i>field_name</i> = <i>value</i> ;	Selects some items (rows) from a given table and includes data from some columns available for those items.

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,
map<String, String> fullColumn)
  STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'
  TBLPROPERTIES (
    "dynamodb.table.name" = "myTable",
    "dynamodb.region" = "eu-west-1",
    "dynamodb.throughput.read.percent" = ".1000",
    "dynamodb.column.mapping" = "hashKey:HashKey,recordTimeStamp:RangeKey" );
```

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](#).

```
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,recordTimeStamp:RangeKey" );
```

Hive Patches

The Amazon EMR team has created the following patches for Hive.

Patch	Description
Write to Amazon S3	<p>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.</p> <p>Status: Submitted Fixed in AWS Hive Version: 0.4 Fixed in Apache Hive Version: n/a (HIVE-2318)</p>

Patch	Description
Scripts in Amazon S3	<p>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.</p> <p>Status: Committed Fixed in AWS Hive Version: 0.4 Fixed in Apache Hive Version: 0.7.0 (HIVE-1624)</p>
Recover partitions	<p>Allows you to recover partitions from table data located in Amazon S3 and Hive table data in HDFS.</p> <p>Status: Not Submitted Fixed in AWS Hive Version: 0.4 Fixed in Apache Hive Version: n/a</p>
Variables in Hive	<p>Create a separate namespace (aside from HiveConf) for managing Hive variables. Adds support for setting variables on the command line using either '-define x=y' or 'set hivevar:x=y'. Adds support for referencing variables in statements using '\${var_name}'. Provides a means for differentiating between hiveconf, hivevar, system, and environment properties in the output of 'set -v'.</p> <p>Status: Committed Fixed in AWS Hive Version: 0.4 Fixed in Apache Hive Version: 0.8.0 (HIVE-2020)</p>
Report progress while writing to Amazon S3	<p>FileSinkOperator reports progress to Hadoop while writing large files, so that the task is not killed.</p> <p>Status: Not Submitted Fixed in AWS Hive Version: 0.5 Fixed in Apache Hive Version: n/a</p>
Fix compression arguments	<p>Corrects an issue where compression values were not set correctly in FileSinkOperator, which resulted in uncompressed files.</p> <p>Status: Submitted Fixed in AWS Hive Version: 0.5 Fixed in Apache Hive Version: n/a (HIVE-2266)</p>
Fix UDAFPercentile to tolerate null percentiles	<p>Fixes an issue where UDAFPercentile would throw a null pointer exception when passed null percentile list.</p> <p>Status: Committed Fixed in AWS Hive Version: 0.5 Fixed in Apache Hive Version: 0.8.0 (HIVE-2298)</p>
Fix hashCode method in DoubleWritable class	<p>Fixes the hashCode() method of DoubleWritable class of Hive and prevents the HashMap (of type DoubleWritable) from behaving as LinkedList.</p> <p>Status: Committed Fixed in AWS Hive Version: 0.7 Fixed in Apache Hive Version: 0.7.0 (HIVE-1629)</p>

Patch	Description
Recover partitions, version 2	<p>Improved version of Recover Partitions that uses less memory.</p> <p>Status: Not Submitted Fixed in AWS Hive Version: 0.7 Fixed in Apache Hive Version: n/a</p>
HAVING clause	<p>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.</p> <p>Status: Committed Fixed in AWS Hive Version: 0.7 Fixed in Apache Hive Version: 0.7.0 (HIVE-1790)</p>
Improve Hive query performance	<p>Reduces startup time for queries spanning a large number of partitions.</p> <p>Status: Committed Fixed in AWS Hive Version: 0.7.1 Fixed in Apache Hive Version: 0.8.0 (HIVE-2299)</p>
Improve Hive query performance for Amazon S3 queries	<p>Reduces startup time for Amazon S3 queries. Set <code>Hive.optimize.s3.query=true</code> to enable optimization.</p> <p>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.</p> <p>The partitions in an optimized query should have the same prefix, with HIVE_TABLE_ROOT as the common prefix.</p> <p>Status: Not Submitted Fixed in AWS Hive Version: 0.7.1 Fixed in Apache Hive Version: n/a</p>
Skip comments in Hive scripts	<p>Fixes an issue where Hive scripts would fail on a comment line; now Hive scripts skip commented lines.</p> <p>Status: Committed Fixed in AWS Hive Version: 0.7.1 Fixed in Apache Hive Version: 0.8.0 (HIVE-2259)</p>
Limit Recover Partitions	<p>Improves performance recovering partitions from Amazon S3 when there are many partitions to recover.</p> <p>Status: Not Submitted Fixed in AWS Hive Version: 0.8.1 Fixed in Apache Hive Version: n/a</p>

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:

Amazon Elastic MapReduce Developer Guide
Supported Hive Versions

Hive Version	Compatible Hadoop Versions	Hive Version Notes
0.13.1	2.4.0	

Hive Version	Compatible Hadoop Versions	Hive Version Notes
		<p>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.</p> <ul style="list-style-type: none"> • Vectorized query: processes thousand-row blocks instead of processing by row. • In-memory cache: hot data kept in-memory for quick reads. • Faster plan serialization • Support for DECIMAL and CHAR datatypes • Sub-query for IN, NOT IN, EXISTS and NOT EXISTS (correlated and uncorrelated) • JOIN conditions in the WHERE clause <p>Other feature contributed by Amazon EMR:</p> <ul style="list-style-type: none"> • Includes an optimization to Hive windowing functions that allows them to scale to large data sets. <p>Notable backward incompatibilities:</p> <ul style="list-style-type: none"> • Does not support -el flag for pushing error-logs to Amazon S3 bucket in case a query failed. • Does not support RECOVER PARTITION syntax. Instead use the native capability, MSCK REPAIR. • Round(sum(c),2) over w1 -> round(sum(c) over w1,2) (in several places). This syntax was changed in Hive 0.12. See HIVE-4214. • Default precision and scale was changed for DECIMAL. Compared to previous Hive versions, DECIMAL in Hive 13 is DECIMAL(10,0). • 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 <code>hive.default.rcfile.serde</code> is set to <code>org.apache.hadoop.hive.serde2.columnar.ColumnarSerDe</code>. See HIVE-4475. <p>Other notes and known issues:</p> <ul style="list-style-type: none"> • 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 <code>hive.metastore.warehouse.dir</code> instead of the database's properties. See HIVE-3486. • When a user loads data into a table using OVERWRITE with a different file it is not being overwritten. See HIVE-6209. • Since Amazon EMR uses HiveServer2, the username must be <code>hadoop</code> with no password. <p>The following patches Hive 0.14.0 patches were backported to this</p>

Amazon Elastic MapReduce Developer Guide
Supported Hive Versions

Hive Version	Compatible Hadoop Versions	Hive Version Notes
		<p>release:</p> <ul style="list-style-type: none"> • HIVE-6367 • HIVE-6394 • HIVE-6783 • HIVE-6785 • HIVE-6938
0.11.0.2	1.0.3 2.2.0	<p>Introduces the following features and improvements. For more information, see Apache Hive 0.11.0 Release Notes.</p> <ul style="list-style-type: none"> • Adds the Parquet library. • Fixes a problem related to the Avro serializer/deserializer accepting a schema URL in Amazon S3. • Fixes a problem with Hive returning incorrect results with indexing turned on. • Change Hive's log level from DEBUG to INFO. • Fixes a problem when tasks do not report progress while deleting files in Amazon S3 dynamic partitions. • This Hive version fixes the following issues: <ul style="list-style-type: none"> • HIVE-4618 • HIVE-4689 • HIVE-4757 • HIVE-4781 • HIVE-4932 • HIVE-4935 • HIVE-4990 • HIVE-5056 • HIVE-5149 • HIVE-5418
0.11.0.1	1.0.3 2.2.0	<ul style="list-style-type: none"> • Creates symlink <code>/home/hadoop/hive/lib/hive_contrib.jar</code> for backward compatibility. • Fixes a problem that prevents installation of Hive 0.11.0 with IAM roles.

Hive Version	Compatible Hadoop Versions	Hive Version Notes
0.11.0	1.0.3 2.2.0	<p>Introduces the following features and improvements. For more information, see Apache Hive 0.11.0 Release Notes.</p> <ul style="list-style-type: none"> • Simplifies <code>hive.metastore.uris</code> and the <code>hive.metastore.local</code> configuration settings. (HIVE-2585) • Changes the internal representation of binary type to <code>byte[]</code>. (HIVE-3246) • Allows <code>HiveStorageHandler.configureTableJobProperties()</code> 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 <code>hive.enforce.bucketing</code> and <code>hive.enforce.sorting insert</code>. (HIVE-4240)

Hive Version	Compatible Hadoop Versions	Hive Version Notes
0.8.1.8	1.0.3	<ul style="list-style-type: none">• 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. 248).• 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. 247).• Adds the ability to set the DynamoDB <code>readThroughputPercent</code> and <code>writeThroughputPercent</code> values per table at creation time. For more information, see Set DynamoDB throughput values per table (p. 248).• 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.

Amazon Elastic MapReduce Developer Guide
Supported Hive Versions

Hive Version	Compatible Hadoop Versions	Hive Version Notes
0.8.1.7	1.0.3	<ul style="list-style-type: none"> • Fixes <code>ColumnPruner</code> so that it works on <code>LateralView</code>. (HIVE-3226) • Fixes <code>utc_from_timestamp</code> and <code>utc_to_timestamp</code> to return correct results. (HIVE-2803) • Fixes a <code>NullPointerException</code> error on a join query with authorization enabled. (HIVE-3225) • Improves mapjoin filtering in the <code>ON</code> condition. (HIVE-2101) • Preserves the filter on a <code>OUTER JOIN</code> condition while merging the join tree. (HIVE-3070) • Fixes <code>ConcurrentModificationException</code> on a <code>lateral view</code> used with <code>explode</code>. (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 <code>NullPointerException</code> 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 <code>Timestamp</code> 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 <code>hive.exec.max.dynamic.partitions</code> 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 <code>hive.merge.mapfiles</code> parameter. If the output path is in Amazon S3, the <code>hive.merge.small-files.avgsize</code> setting is ignored. For more information, see Hive File Merge Behavior with Amazon S3 (p. 242) and Hive Configuration Variables. • Improves clean-up of junk files after an <code>INSERT OVERWRITE</code> command.
0.8.1.6	1.0.3	<ul style="list-style-type: none"> • Adds support for IAM roles. For more information, see Configure IAM Roles for Amazon EMR (p. 185)

Amazon Elastic MapReduce Developer Guide
Supported Hive Versions

Hive Version	Compatible Hadoop Versions	Hive Version Notes
0.8.1.5	1.0.3	<ul style="list-style-type: none"> • 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 <code>map<string, string></code> 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	<p>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.</p> <ul style="list-style-type: none"> • Support Binary DataType (HIVE-2380) • Support Timestamp DataType (HIVE-2272) • Provide a Plugin Developer Kit (HIVE-2244) • Support INSERT INTO append semantics (HIVE-306) • Support Per-Partition SerDe (HIVE-2484) • Support Import/Export facilities (HIVE-1918) • Support Bitmap Indexes (HIVE-1803) • Support RCFile Block Merge (HIVE-1950) • Incorporate Group By Optimization (HIVE-1694) • Enable HiveServer to accept -hiveconf option (HIVE-2139) • Support --auxpath option (HIVE-2355) • Add a new builtins subproject (HIVE-2523) • Insert overwrite table db.tname fails if partition already exists (HIVE-2617) • Add a new input format that passes multiple GZip files to each mapper, so fewer mappers are needed. (HIVE-2089) • Incorporate JDBC Driver improvements (HIVE-559, HIVE-1631, HIVE-2000, HIVE-2054, HIVE-2144, HIVE-2153, HIVE-2358, HIVE-2369, HIVE-2456)

Hive Version	Compatible Hadoop Versions	Hive Version Notes
0.7.1.4	0.20.205	Prevents the "SET" command in Hive from changing the current database of the current session.
0.7.1.3	0.20.205	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. 394) .
0.7.1.2	0.20.205	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 sparse data sets.
0.7.1.1	0.20.205	Introduces support for accessing DynamoDB, as detailed in Export, Import, Query, and Join Tables in DynamoDB Using Amazon EMR (p. 384) . 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.
0.7.1	0.20.205, 0.20, 0.18	Improves Hive query performance for a large number of partitions and for Amazon S3 queries. Changes Hive to skip commented lines.
0.7	0.20, 0.18	Improves Recover Partitions to use less memory, fixes the hashCode method, and introduces the ability to use the HAVING clause to filter on groups by expressions.
0.5	0.20, 0.18	Fixes issues with FileSinkOperator and modifies UDAFPercentile to tolerate null percentiles.
0.4	0.20, 0.18	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.

The AWS CLI does not support installing specific Hive versions. When using the AWS CLI, the latest version of Hive included on the AML 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

1	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.
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 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 \(p. 473\)](#).

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 [Prepare an Output Location \(Optional\) \(p. 175\)](#).
 - 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:

```
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:

```
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](#).

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:

```
aws emr add-steps --cluster-id j-2AXXXXXXGAPLF \  
--steps Type=Hive,Name="Hive Program",ActionOnFailure=CONTINUE,Args=[-f,pathtohivescript,-d,INPUT=pathtoinputdata,-d,OUTPUT=pathtooutputbucket]
```

- Windows users:

```
aws emr add-steps --cluster-id j-2AXXXXXXGAPLF --steps Type=Hive,Name="Hive Program",ActionOnFailure=CONTINUE,Args=[-f, pathtohivescript, -d, INPUT=pathtoinputdata, -d, OUTPUT=pathtooutputbucket]
```

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

Create a Hive Metastore Outside the Cluster

Hive records metastore information in a MySQL database that is located, by default, on the master node. The metastore contains a description of the input data, including the partition names and data types, contained in the input files.

When a cluster terminates, all associated cluster nodes shut down. All data stored on a cluster node, including the Hive metastore, is deleted. Information stored elsewhere, such as in your Amazon S3 bucket, persists.

If you have multiple clusters that share common data and update the metastore, you should locate the shared metastore on persistent storage.

To share the metastore between clusters, override the default location of the MySQL database to an external persistent storage location.

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 cluster

1. Create a MySQL database.

Relational Database Service (RDS) provides a cloud-based MySQL database. Instructions on how to create an Amazon RDS database are at <http://aws.amazon.com/rds/>.

2. Modify your security groups to allow JDBC connections between your MySQL database and the **ElasticMapReduce-Master** security group.

Instructions on how to modify your security groups for access are at <http://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:

```
<configuration>
  <property>
    <name>javax.jdo.option.ConnectionURL</name>
    <value>jdbc:mysql://hostname:3306/hive?createDatabaseIfNotEx
ist=true</value>
    <description>JDBC connect string for a JDBC metastore</description>
```

```
</property>
<property>
  <name>javax.jdo.option.ConnectionDriverName</name>
  <value>com.mysql.jdbc.Driver</value>
  <description>Driver class name for a JDBC metastore</description>
</property>
<property>
  <name>javax.jdo.option.ConnectionUserName</name>
  <value>username</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 MySQL. <username> and <password> are the credentials for your MySQL database.

The MySQL 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:

```
aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applica
tions 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://elast
icmapreduce/libs/hive/hive-script",\
Args=["--base-path","s3://elasticmapreduce/libs/hive","--install-hive-
site","--hive-site=s3://mybucket/hive-site.xml","--hive-versions","latest"]
```

- Windows users:

```
aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applica
tions 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://elasticmapre
duce/libs/hive/hive-script",Args=["--base-path","s3://elasticmapre
duce/libs/hive","--install-hive-site","--hive-site=s3://mybucket/hive-
site.xml","--hive-versions","latest"]
```

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>.

5. Connect to the master node of your cluster.

Instructions on how to connect to the master node are available at [Connect to the Master Node Using SSH \(p. 441\)](#).

6. Create your Hive tables specifying the location on Amazon S3 by entering a command similar to the following:

```
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. For more information, see [Choose an Amazon Machine Image \(AMI\) \(p. 48\)](#).

- Hive 0.13.1 JDBC drivers: <http://amazon-odbc-jdbc-drivers.s3.amazonaws.com/public/HiveJDBC.zip>
- Hive 0.11.0 JDBC drivers: <http://aws.amazon.com/developertools/1982901737448217>
- Hive 0.8.1 JDBC drivers: <http://aws.amazon.com/developertools/4897392426085727>

To install and configure SQL Workbench

1. Download the SQL Workbench/J client for your operating system from <http://www.sql-workbench.net/downloads.html>.
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 version	Command
0.13.1	<code>ssh -o ServerAliveInterval=10 -i <i>path-to-key-file</i> -N -L 10000:localhost:10000 hadoop@<i>master-public-dns-name</i></code>
0.11.0	<code>ssh -o ServerAliveInterval=10 -i <i>path-to-key-file</i> -N -L 10004:localhost:10004 hadoop@<i>master-public-dns-name</i></code>
0.8.1	<code>ssh -o ServerAliveInterval=10 -i <i>path-to-key-file</i> -N -L 10003:localhost:10003 hadoop@<i>master-public-dns-name</i></code>

- 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 \(p. 448\)](#).

Hive version	Tunnel settings
0.13.1	Source port: 10000 Destination: <i>master-public-dns-name</i> :10000
0.11.0	Source port: 10004 Destination: <i>master-public-dns-name</i> :10004
0.8.1	Source port: 10003 Destination: <i>master-public-dns-name</i> :10003

4. Add the JDBC driver to SQL Workbench/J.
 - 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**.

Hive driver version	JAR files to add
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**.

Hive version	Driver classname
0.13.1	com.amazon.hive.jdbc3.HS2Driver
0.11.0	org.apache.hadoop.hive.jdbc.HiveDriver.jar
0.8.1	org.apache.hadoop.hive.jdbc.HiveDriver.jar

- When you return to the **Manage Drivers** dialog box, verify that the **Classname** field is populated and click **OK**.
- 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.

Hive version	URL
0.13.1	<code>jdbc:hive2://localhost:10000/default</code>
0.11.0	<code>jdbc:hive://localhost:10004/default</code>
0.8.1	<code>jdbc:hive://localhost:10003/default</code>

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.

- 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, go to <http://wiki.apache.org/hadoop/Hive/HiveClient> and <http://wiki.apache.org/hadoop/Hive/HiveJDBCInterface>.

Apache Spark

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

[Apache Spark](#) is a cluster framework and programming model that helps you process data. 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.

Topics

- [Use Spark Interactively or in Batch Mode \(p. 269\)](#)
- [Create a Cluster With Spark \(p. 269\)](#)
- [Configure Spark \(p. 270\)](#)
- [Access the Spark Shell \(p. 272\)](#)
- [Write a Spark Application \(p. 273\)](#)
- [Adding a Spark Step \(p. 275\)](#)

Use Spark Interactively or in Batch Mode

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

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 \(p. 272\)](#).

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 \(p. 473\)](#). For an example of launching a long-running cluster and submitting a Spark step, see [Adding a Spark Step \(p. 275\)](#).

Create a Cluster With Spark

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

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 the **Software Configuration** field, choose **Amazon AMI Version 3.8.0** or later.
4. For the **Applications to be installed** field, 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 \(p. 270\)](#). 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:
 - Linux, UNIX, and Mac OS X users:

```
aws emr create-cluster --name "Spark cluster" --ami-version 3.8 --applications Name=Spark\
    --ec2-attributes KeyName=myKey --instance-type m3.xlarge --instance-count 3 --use-default-roles
```

- Windows users:

```
aws emr create-cluster --name "Spark cluster" --ami-version 3.8 --applications Name=Spark --ec2-attributes KeyName=myKey --instance-type m3.xlarge --instance-count 3 --use-default-roles
```

To launch a cluster with Spark installed using the AWS CLI

- Create the cluster with the following command:

```
aws emr create-cluster --name "Spark cluster" --ami-version 3.8 --applications Name=Spark --ec2-attributes KeyName=myKey --instance-type m3.xlarge --instance-count 3 --use-default-roles
```

To launch a cluster with Spark installed using the AWS 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:

```
AmazonElasticMapReduceClient emr = new AmazonElasticMapReduceClient(credentials);
SupportedProductConfig sparkConfig = new SupportedProductConfig()
    .withName("Spark");

RunJobFlowRequest request = new RunJobFlowRequest()
    .withName("Spark Cluster")
    .withAmiVersion("3.8.0")
    .withNewSupportedProducts(sparkConfig)
    .withInstances(new JobFlowInstancesConfig()
        .withEc2KeyName("myKeyName")
        .withInstanceCount(1)
        .withKeepJobFlowAliveWhenNoSteps(true)
        .withMasterInstanceType("m3.xlarge")
        .withSlaveInstanceType("m3.xlarge")
    );
RunJobFlowResult result = emr.runJobFlow(request);
```

Configure Spark

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.
- x
Sets the default Spark config for dictate Spark job utilization (e.g. 1 executor per node, all vcore and memory, all core nodes)

It is also possible to configure Spark dynamically for each application. For more information, see [Overriding Spark Default Configuration Settings \(p. 277\)](#).

Manually adjusting executor 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 1024m, using the following:

```
aws emr create-cluster --name "Spark cluster" --ami-version 3.8 --applications
  Name=Spark, \
  Args=[-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 3.8.0** 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 \(p. 441\)](#). 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.

```
scala> sc
res0: org.apache.spark.SparkContext = org.apache.spark.SparkContext@404721db

scala> val textFile = sc.textFile("s3://elasticmapreduce/samples/hive-ads/tables/impressions/dt=2009-04-13-08-05/ec2-0-51-75-39.amazon.com-2009-04-13-08-05.log")
```

Spark creates the `textFile` and associated [data structure](#). Next, the example counts the number of lines in the log file with the string "cartoonnetwork.com":

```
scala> val linesWithCartoonNetwork = textFile.filter(line => line.contains("cartoonnetwork.com")).count()
linesWithCartoonNetwork: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[2] at filter at <console>:23
<snip>
<Spark program runs>
scala> linesWithCartoonNetwork
res2: Long = 9
```

Example Using the Python-based Spark shell to count the occurrences of a string in a file stored in Amazon S3

Spark also includes a Python-based shell, `pyspark`, that you can use to prototype Spark programs written in Python. Just as with `spark-shell`, invoke `pyspark` on the master node; it also has the same `SparkContext` object.

```
>>> sc
<pyspark.context.SparkContext object at 0x7fe7e659fa50>
>>> textfile = sc.textFile("s3://elasticmapreduce/samples/hive-ads/tables/impressions/dt=2009-04-13-08-05/ec2-0-51-75-39.amazon.com-2009-04-13-08-05.log")
```

Spark creates the `textFile` and associated [data structure](#). Next, the example counts the number of lines in the log file with the string "cartoonnetwork.com".

```
>>> linesWithCartoonNetwork = textfile.filter(lambda line: "cartoonnetwork.com"
in line).count()
15/06/04 17:12:22 INFO lzo.GPLNativeCodeLoader: Loaded native gpl library from
the embedded binaries
15/06/04 17:12:22 INFO lzo.LzoCodec: Successfully loaded & initialized native-
lzo library [hadoop-lzo rev EXAMPLE]
15/06/04 17:12:23 INFO fs.EmrFileSystem: Consistency disabled, using
com.amazon.ws.emr.hadoop.fs.s3n.S3NativeFileSystem as filesystem implementation
<snip>
<Spark program continues>
>>> linesWithCartoonNetwork
9
```

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

```
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

```
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;
        int n = 100000 * slices;
        List<Integer> l = new ArrayList<Integer>(n);
        for (int i = 0; i < n; i++) {
            l.add(i);
        }

        JavaRDD<Integer> dataSet = jsc.parallelize(l, slices);

        int count = dataSet.map(new Function<Integer, Integer>() {
            @Override
            public Integer call(Integer integer) {
                double x = Math.random() * 2 - 1;
                double y = Math.random() * 2 - 1;
                return (x * x + y * y < 1) ? 1 : 0;
            }
        }).reduce(new Function2<Integer, Integer, Integer>() {
            @Override
            public Integer call(Integer integer, Integer integer2) {
                return integer + integer2;
            }
        });

        System.out.println("Pi is roughly " + 4.0 * count / n);

        jsc.stop();
    }
}
```

```
}  
}
```

Python

```
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 %f" % (4.0 * count / n)  
  
    sc.stop()
```

Adding a Spark Step

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

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 will run the `spark-submit` script as a step on your behalf. With the API, you use a step to invoke `spark-submit` using `script-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/>.
2. In the **Cluster List**, choose the name of your cluster.
3. Scroll to the **Steps** section and expand it, then choose **Add step**.

- 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**).
- Choose **Add**. The step appears in the console with a status of Pending.
- 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.

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.

- Use `create-cluster`.
 - Linux, UNIX, and Mac OS X users:

```
aws emr create-cluster --name "Add Spark Step Cluster" --ami-version 3.8
--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-ex
amples-1.3.1-hadoop2.4.0.jar,10]
```

- Windows users:

```
aws emr create-cluster --name "Add Spark Step Cluster" --ami-version 3.8
--applications Name=Spark --ec2-attributes KeyName=myKey --instance-type
m3.xlarge --instance-count 3 --steps Type=Spark,Name="Spark Program",Ac
tionOnFailure=CONTINUE,Args=[-class,org.apache.spark.ex
amples.SparkPi,/home/hadoop/spark/lib/spark-examples-1.3.1-ha
dooop2.4.0.jar,10]
```

- 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/ha
doop/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.ex
amples.SparkPi,/home/hadoop/spark/lib/spark-*.jar,10]
```

To submit work to Spark using the AWS SDK for Java

- To submit work to a cluster, use a step to run the `spark-submit` script on your EMR cluster. You add the step using the `addJobFlowSteps` method in [AmazonElasticMapReduceClient](#):

```
AWSCredentials credentials = new BasicAWSCredentials(accessKey, secretKey);
AmazonElasticMapReduce emr = new AmazonElasticMapReduceClient(credentials);

StepFactory stepFactory = new StepFactory();
AmazonElasticMapReduceClient emr = new AmazonElasticMapReduceClient(creden
tials);
AddJobFlowStepsRequest req = new AddJobFlowStepsRequest();
req.withJobFlowId("j-1K48XXXXXXHCB");

List<StepConfig> stepConfigs = new ArrayList<StepConfig>();

StepConfig sparkStep = new StepConfig()
    .withName("Spark Step")
    .withActionOnFailure("CONTINUE")
    .withHadoopJarStep(stepFactory.newScriptRunnerStep("/home/ha
doop/spark/bin/spark-submit", "--class", "org.apache.spark.ex
amples.SparkPi", "/home/hadoop/spark/lib/spark-examples-1.3.1-ha
doop2.4.0.jar", "10"));

stepConfigs.add(sparkStep);
req.withSteps(stepConfigs);
AddJobFlowStepsResult result = emr.addJobFlowSteps(req);
```

Overriding Spark Default Configuration Settings

It is probable that you will want to override Spark default configuration values on a per-application basis. You can do this when you submit applications using a step, which is 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/ha  
dooop/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.

Impala

Impala is an open source tool in the Hadoop ecosystem for interactive, ad hoc querying using SQL syntax. Instead of using MapReduce, it leverages a massively parallel processing (MPP) engine similar to that found in traditional relational database management systems (RDBMS). With this architecture, you can query your data in HDFS or HBase tables very quickly, and leverage Hadoop's ability to process diverse data types and provide schema at runtime. This lends Impala to interactive, low-latency analytics. In addition, Impala uses the Hive metastore to hold information about the input data, including the partition names and data types.

Note

Impala on Amazon EMR requires AMIs running Hadoop 2.x or greater.

Impala on Amazon EMR supports the following:

- Large subset of SQL and HiveQL commands
- Querying data in HDFS and HBase
- Use of ODBC and JDBC drivers
- Concurrent client requests for each Impala daemon
- Kerberos authentication
- Partitioned tables
- Appending and inserting data into tables using the INSERT statement
- Multiple HDFS file formats and compression codecs. For more information, see [Impala-supported File and Compression Formats \(p. 293\)](#).

For more information about Impala, go to http://en.wikipedia.org/wiki/Cloudera_Impala.

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. 294).

Tutorial: Launching and Querying Impala Clusters on Amazon EMR

This tutorial demonstrates how you can perform interactive queries with Impala on Amazon EMR. 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? \(p. 10\)](#).

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

1. Go to <http://aws.amazon.com> and click **Sign Up Now**.
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 both the Amazon EMR console and the 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**.
3. On the **Create Cluster** page, in the **Cluster Configuration** section, verify the fields according to the following table.

Field	Action
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.

Field	Action
Termination protection	<p>Leave the default option selected: Yes.</p> <p>Enabling termination protection ensures that the cluster does not shut down due to accident or error. For more information, see Managing Cluster Termination (p. 462) . 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.</p>
Logging	<p>This determines whether Amazon EMR captures detailed log data to Amazon S3.</p> <p>For more information, see View Log Files (p. 413) .</p>
Log folder S3 location	<p>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.</p> <p>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.</p> <p>For more information, see View Log Files (p. 413) .</p>
Debugging	<p>This option creates a debug log index in 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 about Amazon SimpleDB, go to the Amazon SimpleDB product description page.</p>

4. In the **Software Configuration** section, verify the fields according to the following table.

Field	Action
Hadoop distribution	<p>Choose Amazon.</p> <p>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. 218).</p>
AMI version	<p>Choose the latest Hadoop 2.4.0 AMI.</p> <p>This determines the version of Hadoop and other applications such as Hive or Pig to run on your cluster. Impala requires an Amazon EMR AMI that has Hadoop 2.x or newer. For more information, see Choose an Amazon Machine Image (AMI) (p. 48).</p>

5. Under the **Additional Applications** list, choose **Impala** and click **Configure and add**.

Note

If you do not see **Impala** in the list, ensure that you have chosen a Hadoop 2.4.0 AMI.

6. In the **Add Application** section, use the following table for guidance on making your selections.

Field	Action
Version	Choose the Impala version from the list, such as 1.2.4 .
Arguments	Optionally, specify command line arguments for Impala to execute. For examples of Impala command line arguments, see the <code>--impala-conf</code> section at AWS EMR Command Line Interface Options (Deprecated) (p. 585).

7. Click **Add**.
8. 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-Classical . Optionally, choose a VPC subnet identifier from the list to launch the cluster in an Amazon VPC. For more information, see Select an Amazon VPC Subnet for the Cluster (Optional) (p. 205).
EC2 Availability Zone	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.
Master	Choose m3.xlarge . This specifies the EC2 instance type to use for the master node. The master node assigns Hadoop tasks to core and task nodes, and monitors their status. There is always one master node in each cluster. For more information, see Instance Groups (p. 34). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 471).
Request Spot Instances	Leave this box unchecked. This specifies whether to run master nodes on Spot Instances. For more information, see (Optional) Lower Costs with Spot Instances (p. 37).

Field	Action
Core	<p>Choose m1.large. This specifies the EC2 instance type to use for the core nodes.</p> <p>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.</p> <p>For more information, see Instance Groups (p. 34). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 471).</p>
Count	Choose 2 .
Request Spot Instances	<p>Leave this box unchecked.</p> <p>This specifies whether to run core nodes on Spot Instances. For more information, see (Optional) Lower Costs with Spot Instances (p. 37).</p>
Task	<p>Accept the default option. This specifies the EC2 instance types to use as task nodes.</p> <p>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.</p> <p>For more information, see Instance Groups (p. 34). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 471).</p>
Count	Choose 0 . You do not use task nodes for this tutorial.
Request Spot Instances	<p>Leave this box unchecked.</p> <p>This specifies whether to run task nodes on Spot Instances. For more information, see (Optional) Lower Costs with Spot Instances (p. 37).</p>

9. In the **Security and Access** section, complete the fields according to the following table.

Field	Action
EC2 key pair	<p>Choose your Amazon EC2 key pair private key from the list.</p> <p>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. 441) .</p>
IAM user access	<p>Choose All other IAM users to make the cluster visible and accessible to all IAM users on the AWS account. For more information, see Configure IAM User Permissions (p. 181) .</p> <p>Alternatively, choose No other IAM users to restrict access to the current IAM user.</p>

Field	Action
Roles configuration	<p>Choose Default to generate the default EMR role and 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 EMR role and EC2 instance profile are required when creating a cluster using the console.</p> <p>The EMR role allows Amazon EMR to access other AWS services on your behalf. The EC2 instance profile controls application access to the Amazon EC2 instances in the cluster.</p> <p>For more information, see Configure IAM Roles for Amazon EMR (p. 185) .</p>

10. 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 \(p. 124\)](#).

11. In the **Steps** section, you do not need to change any of these settings.
12. Review your configuration and if you are satisfied with the settings, choose **Create Cluster**.
13. When the cluster starts, the console displays the **Cluster Details** page.

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:

```
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:

```
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.

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

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 \(p. 441\)](#).
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, pub  
lish_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.quant
ity) 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;
```

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.quant
ity) 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 \(p. 459\)](#).

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 \(p. 289\)](#)
- [Wikipedia \(p. 290\)](#)

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:

```
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:

```
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_demograph  
ics.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  
  
# Copy wikipedia data from s3 to hdfs  
hadoop jar /home/hadoop/lib/emr-s3distcp-1.0.jar -Dmapreduce.job.reduces=30  
--src s3://spark-data/ --dest hdfs://$HADOOP_NAMENODE_HOST:$HADOOP_NAMEN  
ODE_PORT/spark-data/ --outputCodec 'none'  
  
# Create hive tables  
hive -e "CREATE EXTERNAL TABLE wiki_small (id BIGINT, title STRING,  
last_modified STRING, xml STRING, text STRING) ROW FORMAT DELIMITED FIELDS  
TERMINATED BY '\t' LOCATION '/spark-data/wikipedia-sample/'"  
hive -e "CREATE EXTERNAL TABLE wiki_full (id BIGINT, title STRING,  
last_modified STRING, xml STRING, text STRING) ROW FORMAT DELIMITED FIELDS  
TERMINATED BY '\t' LOCATION '/spark-data/wikipedia-2010-09-12/'"  
hive -e "show tables"  
  
# Check client hostname  
client="127.0.0.1"  
echo "Checking client list..."  
nodelist=`curl -s http://$HADOOP_NAMENODE_HOST:9026/ws/v1/cluster/hostStatus`  
echo "Found client list: $nodelist"  
  
arr=$(echo $nodelist | tr "\"" "\n")  
for a in $arr  
do  
    if [[ $a == ip-* || $a == domU-* || $a =~ ^[0-9] ]]; then  
        client=$a  
    fi  
done
```

```
echo "Choose client $client"

# Show tables
impala-shell -r -i $client:21000 --query="show tables"

# Query wiki_small table
impala-shell -r -i $client:21000 --query="SELECT COUNT(1) FROM wiki_small
WHERE TEXT LIKE '%Berkeley%'"
impala-shell -r -i $client:21000 --query="SELECT title FROM wiki_small WHERE
TEXT LIKE '%Berkeley%'"

# Query wiki_full table
impala-shell -r -i $client:21000 --query="SELECT COUNT(1) FROM wiki_full
WHERE TEXT LIKE '%Berkeley%'"
impala-shell -r -i $client:21000 --query="SELECT title FROM wiki_full WHERE
TEXT LIKE '%Berkeley%'"
```

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 \(AMI\) \(p. 48\)](#). The Amazon EMR console does not support Impala versioning and always launches the latest version of Impala.

Impala Version	AMI Version	Impala Version Details
1.2.4	3.1.0 and later	Adds support for Impala 1.2.4.
1.2.1	3.0.2 3.0.3 3.0.4	Amazon EMR introduces support for Impala with this version.

Topics

- [Updates for Impala 1.2.4 \(p. 291\)](#)

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 JDBS and ODBC drivers.
- Numeric and conditional functions can return SMALLINT, FLOAT, and other smaller numerical types.

For more information, see: http://en.wikipedia.org/wiki/Cloudera_Impala.

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 \(p. 294\)](#).

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 \(p. 451\)](#).

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 \(p. 451\)](#) and [Option 2, Part 2: Configure Proxy Settings to View Websites Hosted on the Master Node \(p. 453\)](#).

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 \(p. 281\)](#).

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 \(p. 286\)](#).

```
$ 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|3-73425-809-9|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|TRAVEL|2013-05-26|Education and Media Group|73.99
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
4|Miles LOPEZ|1956-03-15|F|ND|miles.lopez.1956@hotmail.com|222-770-7004
5|Mackenzie PETERSON|1970-09-05|F|NJ|mackenzie.peterson.1970@outlook.com|114-521-5716
6|Maria SPENCER|2002-12-20|F|TX|maria.spencer.2002@yahoo.com|377-612-4105
7|Sienna HENDERSON|1982-11-04|F|MO|sienna.henderson.1982@gmail.com|199-288-5062
8|Samantha WALLACE|1998-03-06|F|TN|samantha.wallace.1998@hotmail.com|711-348-7410
9|Nevaeh PETERSON|1991-06-26|F|AL|nevaeh.peterson.1991@live.com|651-686-3436

$ head transactions/transactions
0|360677155|84060207|4|2010-03-24 10:24:22
1|228662770|136084430|5|2009-07-03 14:53:09
2|355529188|26348618|9|2009-09-13 11:53:26
3|1729168|20837134|5|2006-01-05 19:31:19
```



```
4|196166644|99142444|19|2007-01-02 15:07:38
5|43026573|479157832|17|2010-04-14 16:42:29
6|306402023|356688712|12|2010-05-24 22:15:54
7|359871959|312932516|31|2000-04-03 11:06:38
8|379787207|265709742|45|2013-09-09 06:01:06
9|144155611|137684093|11|2010-06-06 17:07:07
```

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.

Input Class (size of each table)	Books table (Million Rows)	Customers table (Million Rows)	Transactions table (Million Rows)
4GB	63	53	87
8GB	125	106	171
16GB	249	210	334
32GB	497	419	659
64GB	991	837	1304
128GB	1967	1664	2538
256GB	3919	3316	5000

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

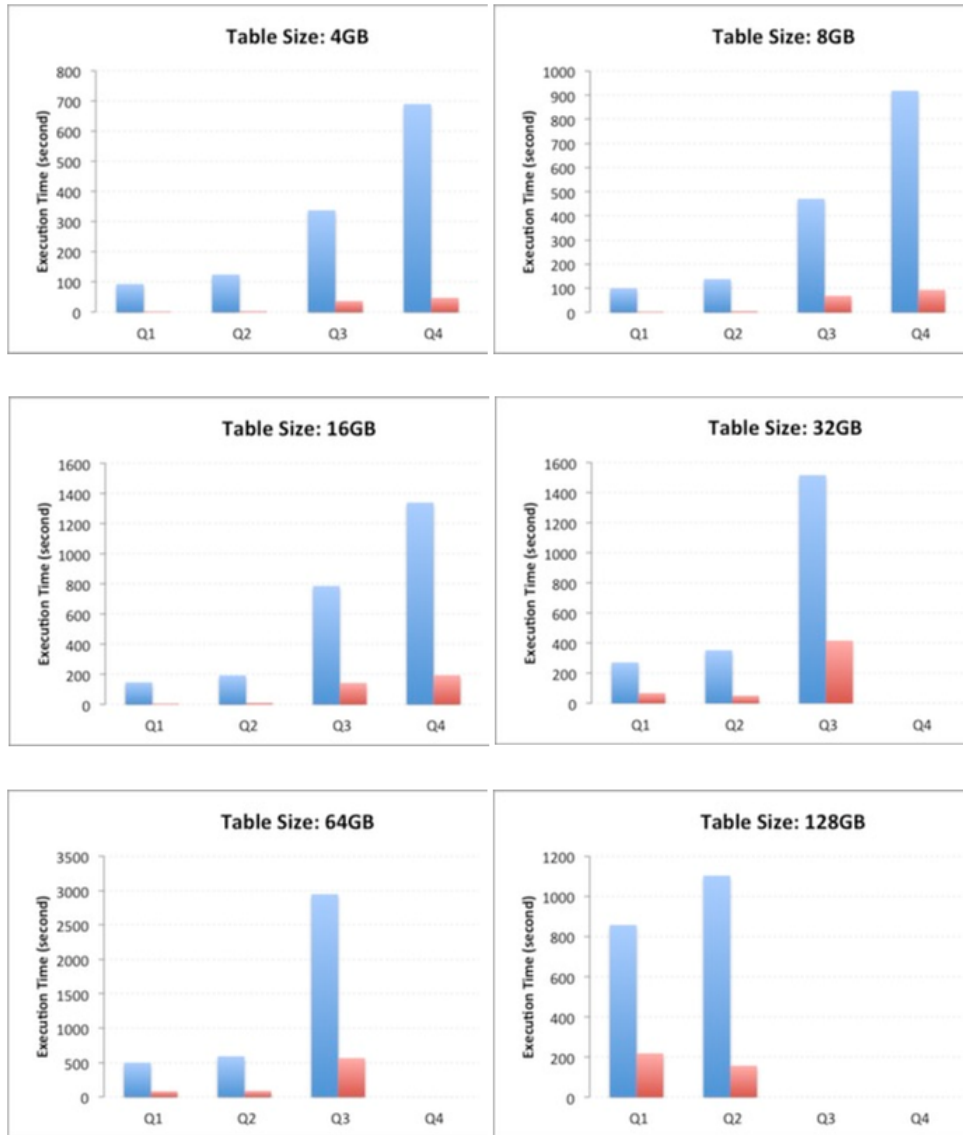
The first set of experimental results were obtained on a cluster of four m1.xlarge instances with Amazon EMR Hadoop 2.2.0 and Impala 1.1.1 installed. According to the hardware specification shown below, an aggregated memory of 60 GB is available on the cluster.

Instance Type	Processor Architecture	vCPUs	ECU	Memory (GiB)	Instance Storage (GB)
m1.xlarge	64-bit	4	8	15	4 x 420

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.

Amazon Elastic MapReduce Developer Guide Performance Test Results

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.

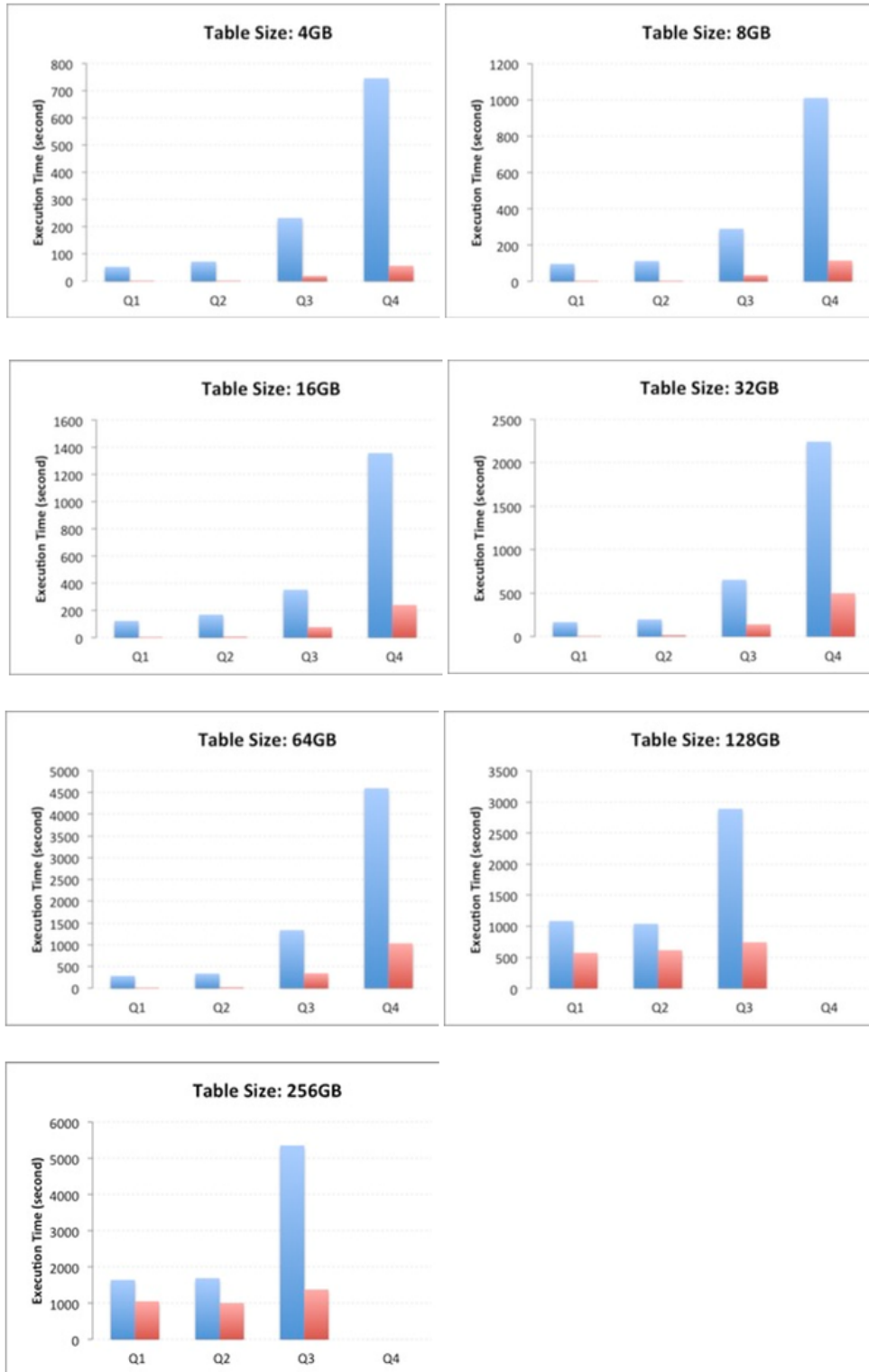


The second experimental environment was a cluster of 4 m2.4xlarge instances with an AMI with Hadoop 2.2.0 and Impala 1.1.1. The aggregated memory on this cluster is 274 GB. Detailed hardware specifications and experimental results are shown below. Like our first set of tests, missing data indicates Impala failures caused by out-of-memory issues, and we declined to run Hive tests for these queries.

Instance Type	Processor Architecture	vCPUs	ECU	Memory (GiB)	Instance Storage (GB)
m2.4xlarge	64-bit	8	26	68.4	2 x 840

Amazon Elastic MapReduce Developer Guide

Performance Test Results



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 table` 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.quant
ity) AS revenue
  FROM transactions JOIN books 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 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.quant
ity) 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 Elastic MapReduce (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 \(p. 473\)](#).

Topics

- [Supported Pig Versions \(p. 302\)](#)
- [Interactive and Batch Pig Clusters \(p. 306\)](#)
- [Submit Pig Work \(p. 306\)](#)
- [Call User Defined Functions from Pig \(p. 308\)](#)

Supported Pig Versions

The Pig version you can add to your cluster depends on the version of the Amazon Elastic MapReduce (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 \(AMI\) \(p. 48\)](#).

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.

Amazon Elastic MapReduce Developer Guide
Supported Pig Versions

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](#).

Pig Version	AMI Version	Configuration Parameters	Pig Version Details
0.12.0 Release Notes Documentation	3.1.0 and later	<code>--ami-version 3.1</code> <code>--ami-version 3.2</code> <code>--ami-version 3.3</code>	Adds support for the following: <ul style="list-style-type: none"> • Streaming UDFs without JVM implementations • ASSERT and IN operators • CASE expression • AvroStorage as a Pig built-in function. • ParquetLoader and ParquetStorer as built-in functions • BigInteger and Big-Decimal types
0.11.1.1 Release Notes Documentation	2.2 and later	<code>--pig-versions 0.11.1.1</code> <code>--ami-version 2.2</code>	Improves performance of LOAD command with PigStorage if input resides in Amazon S3.
0.11.1 Release Notes Documentation	2.2 and later	<code>--pig-versions 0.11.1</code> <code>--ami-version 2.2</code>	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 .
0.9.2.2 Release Notes Documentation	2.2 and later	<code>--pig-versions 0.9.2.2</code> <code>--ami-version 2.2</code>	Adds support for Hadoop 1.0.3.
0.9.2.1 Release Notes Documentation	2.2 and later	<code>--pig-versions 0.9.2.1</code> <code>--ami-version 2.2</code>	Adds support for MapR. For more information, see Using the MapR Distribution for Hadoop (p. 218).

Pig Version	AMI Version	Configuration Parameters	Pig Version Details
0.9.2 Release Notes Documentation	2.2 and later	<code>--pig-versions 0.9.2</code> <code>--ami-version 2.2</code>	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 .
0.9.1 Release Notes Documentation	2.0	<code>--pig-versions 0.9.1</code> <code>--ami-version 2.0</code>	
0.6 Release Notes	1.0	<code>--pig-versions 0.6</code> <code>--ami-version 1.0</code>	
0.3 Release Notes	1.0	<code>--pig-versions 0.3</code> <code>--ami-version 1.0</code>	

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 Elastic MapReduce (Amazon EMR) clusters. For more information about how to do this, see [Apache Pig \(p. 302\)](#). 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.

Patch	Description
PIG-1429	Add the Boolean data type to Pig as a first class data type. For more information, go to https://issues.apache.org/jira/browse/PIG-1429 . Status: Committed Fixed in Apache Pig Version: 0.10
PIG-1824	Support import modules in Jython UDF. For more information, go to https://issues.apache.org/jira/browse/PIG-1824 . Status: Committed Fixed in Apache Pig Version: 0.10
PIG-2010	Bundle registered JARs on the distributed cache. For more information, go to https://issues.apache.org/jira/browse/PIG-2010 . Status: Committed Fixed in Apache Pig Version: 0.11
PIG-2456	Add a ~/.pigbootup file where the user can specify default Pig statements. For more information, go to https://issues.apache.org/jira/browse/PIG-2456 . Status: Committed Fixed in Apache Pig Version: 0.11
PIG-2623	Support using Amazon S3 paths to register UDFs. For more information, go to https://issues.apache.org/jira/browse/PIG-2623 . Status: Committed Fixed in Apache Pig Version: 0.10, 0.11

Pig 0.9.1 Patches

The Amazon EMR team has applied the following patches to the Amazon EMR version of Pig 0.9.1.

Patch	Description
Support JAR files and Pig scripts in dfs	Add support for running scripts and registering JAR files stored in HDFS, Amazon S3, or other distributed file systems. For more information, go to https://issues.apache.org/jira/browse/PIG-1505 . Status: Committed Fixed in Apache Pig Version: 0.8.0
Support multiple file systems in Pig	Add support for Pig scripts to read data from one file system and write it to another. For more information, go to https://issues.apache.org/jira/browse/PIG-1564 . Status: Not Committed Fixed in Apache Pig Version: n/a

Patch	Description
Add Piggybank datetime and string UDFs	<p>Add datetime and string UDFs to support custom Pig scripts. For more information, go to https://issues.apache.org/jira/browse/PIG-1565.</p> <p>Status: Not Committed Fixed in Apache Pig Version: n/a</p>

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>.

Interactive and Batch Pig Clusters

Amazon Elastic MapReduce (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 \(p. 473\)](#).

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>.

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>.

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. 306).
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 my  
functions;
```

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 modeled after Google's BigTable. It was developed as part of Apache Software Foundation's Hadoop project and runs on top of Hadoop Distributed File System (HDFS) to provide BigTable-like capabilities for Hadoop. HBase provides you a fault-tolerant, efficient way of storing large quantities of sparse data using column-based compression and storage. In addition, HBase provides fast lookup of data because data is stored in-memory instead of on disk. HBase is optimized for sequential write operations, and is highly efficient for batch inserts, updates, and deletes.

HBase works seamlessly with Hadoop, sharing its file system and serving as a direct input and output to Hadoop jobs. 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).

Additionally, HBase on Amazon EMR provides the ability to back up your HBase data directly to Amazon Simple Storage Service (Amazon S3). You can also restore from a previously created backup when launching an HBase cluster.

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 for you 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.

Topics

- [Supported HBase Versions \(p. 311\)](#)
- [HBase Cluster Prerequisites \(p. 311\)](#)
- [Install HBase on an Amazon EMR Cluster \(p. 312\)](#)
- [Connect to HBase Using the Command Line \(p. 317\)](#)
- [Back Up and Restore HBase \(p. 318\)](#)
- [Terminate an HBase Cluster \(p. 324\)](#)
- [Configure HBase \(p. 324\)](#)
- [Access HBase Data with Hive \(p. 329\)](#)
- [View the HBase User Interface \(p. 330\)](#)
- [View HBase Log Files \(p. 330\)](#)
- [Monitor HBase with CloudWatch \(p. 331\)](#)
- [Monitor HBase with Ganglia \(p. 332\)](#)

Supported HBase Versions

HBase Version	AMI Version	AWS CLI Configuration Parameters	HBase Version Details
0.94.18	3.1.0 and later	<code>--ami-version 3.1</code> <code>--ami-version 3.2</code> <code>--ami-version 3.3</code> <code>--applications</code> <code>Name=HBase</code>	<ul style="list-style-type: none"> • Bux fixes and enhancements.
0.94.7	3.0-3.0.4	<code>--ami-version 3.0</code> <code>--applications</code> <code>Name=HBase</code>	
0.92	2.2 and later	<code>--ami-version 2.2</code> <code>or later</code> <code>--applications</code> <code>Name=HBase</code>	

HBase Cluster Prerequisites

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)**—If you want to monitor HBase performance metrics, you can 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 Amazon S3 bucket to receive log files when you create the cluster.

Install HBase on an Amazon EMR Cluster

When you launch HBase on Amazon EMR, you get the benefits of running in the Amazon Web Services (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 \(p. 310\)](#).

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 \(p. 324\)](#).

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.

Field	Action
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. 462) . 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. 413) .

Field	Action
Log folder S3 location	<p>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.</p> <p>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.</p> <p>For more information, see View Log Files (p. 413) .</p>
Debugging	<p>This option creates a debug log index in 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 about Amazon SimpleDB, go to the Amazon SimpleDB product description page.</p>

4. In the **Software Configuration** section, verify the fields according to the following table.

Field	Action
Hadoop distribution	<p>Choose Amazon.</p> <p>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. 218).</p>
AMI version	<p>Choose the latest Hadoop 2.x AMI or the latest Hadoop 1.x AMI from the list.</p> <p>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. 48).</p>

5. Under the **Additional Applications** list, choose **HBase** and click **Configure and add**.
6. In the **Add Application** section, indicate whether you want 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 click **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 \(p. 318\)](#).

Field	Action
Restore from backup	Specify whether to pre-load the HBase cluster with data stored in Amazon S3.
Backup location	Specify the URI where the backup to restore from resides in Amazon S3.
Backup version	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.
Schedule Regular Backups	Specify whether to schedule automatic incremental backups. The first backup will be a full backup to create a baseline for future incremental backups.

Field	Action
Consistent backup	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.
Backup frequency	The number of Days/Hours/Minutes between scheduled backups.
Backup location	The Amazon S3 URI where backups will be stored. The backup location for each HBase cluster should be different to ensure that differential backups stay correct.
Backup start time	Specify when the first backup should occur. You can set this to <code>now</code> , 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, <code>2012-06-15T20:00Z</code> , would set the start time to June 15, 2012 at 8pm UTC.

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, go to the [Request to Increase Amazon EC2 Instance Limit Form](#).

Field	Action
Network	Choose the default VPC. For more information about the default VPC, see Your Default VPC and Subnets in the <i>guide-vpc-user</i> . 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 Select an Amazon VPC Subnet for the Cluster (Optional) (p. 205).
EC2 Availability Zone	Choose No preference . Optionally, you can launch the cluster in a specific EC2 Availability Zone. For more information, see Regions and Availability Zones in the <i>Amazon EC2 User Guide for Linux Instances</i> .

Field	Action
Master	<p>Accept the default instance type.</p> <p>The master node assigns Hadoop tasks to core and task nodes, and monitors their status. There is always one master node in each cluster.</p> <p>This specifies the EC2 instance type to use for the master node.</p> <p>The default instance type is m1.medium for Hadoop 2.x. This instance type is suitable for testing, development, and light workloads.</p> <p>For more information on instance types supported by Amazon EMR, see Virtual Server Configurations. For more information on Amazon EMR instance groups, see Instance Groups (p. 34). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 471).</p>
Request Spot Instances	<p>Leave this box unchecked.</p> <p>This specifies whether to run master nodes on Spot Instances. For more information, see (Optional) Lower Costs with Spot Instances (p. 37).</p>
Core	<p>Accept the default instance type.</p> <p>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.</p> <p>This specifies the EC2 instance types to use as core nodes.</p> <p>The default instance type is m1.medium for Hadoop 2.x. This instance type is suitable for testing, development, and light workloads.</p> <p>For more information on instance types supported by Amazon EMR, see Virtual Server Configurations. For more information on Amazon EMR instance groups, see Instance Groups (p. 34). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 471).</p>
Count	Choose 2 .
Request Spot Instances	<p>Leave this box unchecked.</p> <p>This specifies whether to run core nodes on Spot Instances. For more information, see (Optional) Lower Costs with Spot Instances (p. 37).</p>
Task	<p>Accept the default instance type.</p> <p>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.</p> <p>This specifies the EC2 instance types to use as task nodes.</p> <p>For more information on instance types supported by Amazon EMR, see Virtual Server Configurations. For more information on Amazon EMR instance groups, see Instance Groups (p. 34). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 471).</p>
Count	Choose 0 .

Field	Action
Request Spot Instances	<p>Leave this box unchecked.</p> <p>This specifies whether to run task nodes on Spot Instances. For more information, see (Optional) Lower Costs with Spot Instances (p. 37).</p>

8. In the **Security and Access** section, complete the fields according to the following table.

Field	Action
EC2 key pair	<p>Choose your Amazon EC2 key pair private key from the list.</p> <p>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. 441) .</p>
IAM user access	<p>Choose All other IAM users to make the cluster visible and accessible to all IAM users on the AWS account. For more information, see Configure IAM User Permissions (p. 181) .</p> <p>Alternatively, choose No other IAM users to restrict access to the current IAM user.</p>
Roles configuration	<p>Choose Default to generate the default EMR role and 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 EMR role and EC2 instance profile are required when creating a cluster using the console.</p> <p>The EMR role allows Amazon EMR to access other AWS services on your behalf. The EC2 instance profile controls application access to the Amazon EC2 instances in the cluster.</p> <p>For more information, see Configure IAM Roles for Amazon EMR (p. 185) .</p>

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 \(p. 124\)](#).

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, you identify the application you want 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 EMR role and EC2 instance profile, type the following command to create them. Alternatively, you can specify your own roles. If you do not specify the EMR role and EC2 instance profile when creating your cluster, you may experience issues with HBase backup and restore functionality. For more information on roles, see [Configure IAM Roles for Amazon EMR \(p. 185\)](#).

```
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 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
```

- Windows users:

```
aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applica  
tions 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 Master node is launched, and the remaining instances are launched as core nodes. All nodes will use the instance type specified in the command.

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

Connect to HBase Using the Command Line

After you create an HBase cluster, the next step is to connect to HBase so you can begin reading and writing data.

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 \(p. 441\)](#).
2. Run `hbase shell`. The HBase shell will open with a prompt similar to the following example.

```
hbase(main):001:0>
```

You can issue HBase shell commands from the prompt. For a description of the shell commands and information on how to call them, type help at the HBase prompt and press **Enter**.

Create a Table

The following command will create 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 will put value 'v1' for row 'r1' in table 't1' and column 'f1'.

```
hbase(main):001:0>put 't1', 'r1', 'f1', 'v1'
```

Get a Value

The following command will get the values for row 'r1' in table 't1'.

```
hbase(main):001:0>get 't1', 'r1'
```

Back Up and Restore HBase

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. Once 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 prepopulated 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 will not be 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 will be queued and executed as soon as the synchronization completes. You can also schedule recurring backups, which will resolve any inconsistencies over time, as data that is missed on one backup pass will be 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-3AEXXXXXX16F2`. 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. Once 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 [Distributed Copy Using S3DistCp \(p. 376\)](#).

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 \(p. 321\)](#).

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, in the **Additional Applications** field, choose **HBase** and click **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 \(p. 312\)](#).
5. In **Backup Location** field, 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. In the **Backup Version** field, 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.

Add Application

Application name HBase

Restore from backup

Backup location

Backup version

Schedule regular backups

Cancel Add

7. Click **Add** and proceed to create the cluster as described in [Plan an Amazon EMR Cluster \(p. 30\)](#).

To schedule automated backups of HBase data using the console

1. In the **Software Configuration** section, in the **Additional Applications** field, choose **HBase** and click **Configure and add**.
2. Click **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 selecting **Days**, **Hours**, or **Minutes** from the drop-down box. The first automated backup that runs will be a full backup, after that, Amazon EMR will save incremental backups based on the schedule 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`, would set the start time to September 26, 2013 at 8pm UTC.

Update Application ✕

HBase

Restore From Backup

Schedule Regular Backups

Consistent Backup

Backup Frequency

Backup Location

Backup Start Time

[cancel](#)

7. Click **Add**.
8. Proceed with creating the cluster as described in [Plan an Amazon EMR Cluster \(p. 30\)](#).

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.

Back Up and Restore HBase Using the AWS CLI

Using the AWS CLI, you can create HBase backups, restore HBase data from backup when creating an Amazon 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`. 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.

- 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.

```
aws emr create-hbase-backup --cluster-id j-3AEXXXXXX16F2 --dir s3://mybucket/backups/j-3AEXXXXXX16F2
```

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-3AEXXXXXX16F2 --dir s3://mybucket/backups/j-3AEXXXXXX16F2 --consistent
```

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

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.

- Type the following command to schedule consistent HBase backups:

To create a consistent weekly full backup, with the first backup starting immediately, type the following command, replace `j-3AEXXXXXX16F2` with the cluster ID, and replace `mybucket` with your Amazon S3 bucket name.

```
aws emr schedule-hbase-backup --cluster-id j-3AEXXXXXX16F2 --type full --dir s3://mybucket/backups/j-3AEXXXXXX16F2 --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-3AEXXXXXX16F2 --type full --dir s3://mybucket/backups/j-3AEXXXXXX16F2 --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-3AEXXXXXX16F2 --type incremental --dir s3://mybucket/backups/j-3AEXXXXXX16F2 --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-3AEXXXXXX16F2 --type incremental --dir s3://mybucket/backups/j-3AEXXXXXX16F2 --interval 24 --unit hours --start-time 2014-06-15T20:00Z --consistent
```

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

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-3AEXXXXXX16F2` with your cluster ID.

```
aws emr disable-hbase-backups --cluster-id j-3AEXXXXXX16F2 --full
```

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

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-3AEXXXXXX16F2` with your cluster ID, and replace `mybucket` with your Amazon S3 bucket name.

```
aws emr restore-from-hbase-backup --cluster-id j-3AEXXXXXX16F2 --dir  
s3://mybucket/backups/j-3AEXXXXXX16F2 --backup-version 20120809T031314Z
```

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

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-3AEXXXXXX16F2`. 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 \  
--restore-from-hbase-backup Dir=s3://mybucket/backups/j-  
3AEXXXXXX16F2,BackupVersion=20120809T031314Z \  
--use-default-roles --ec2-attributes KeyName=myKey \  
--instance-type c1.xlarge --instance-count 3 --termination-protected
```

- Windows users:

```
aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applica  
tions Name=Hue Name=Hive Name=Pig Name=HBase --restore-from-hbase-backup  
Dir=s3://mybucket/backups/j-3AEXXXXXX16F2,BackupVersion=20120809T031314Z  
--use-default-roles --ec2-attributes KeyName=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 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>.

Terminate an HBase Cluster

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. For more information on terminating a cluster, see [Terminate a Cluster \(p. ?\)](#).

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, you 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 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 a list of regions supported by Amazon EMR see [Choose an AWS Region \(p. 31\)](#). 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 on configuring bootstrap actions, see [\(Optional\) Create Bootstrap Actions to Install Additional Software \(p. 124\)](#)

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.

Variable	Description
<code>hbase-master-opts</code>	Options that control how the JVM runs the master daemon. If set, these settings override the default <code>HBASE_MASTER_OPTS</code> variables.
<code>regionserver-opts</code>	Options that control how the JVM runs the region server daemon. If set, these settings override the default <code>HBASE_REGIONSERVER_OPTS</code> variables.
<code>zookeeper-opts</code>	Options that control how the JVM runs the zookeeper daemon. If set, these settings override the default <code>HBASE_ZOOKEEPER_OPTS</code> variables.

For more information about these options, go to <http://hbase.apache.org/configuration.html#hbase.env.sh>.

- 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 --applica  
tions 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/boot  
strap-actions/configure-hbase-daemons,Args=[ "--hbase-zookeeper-opts=-  
Xmx1024m -XX:GCTimeRatio=19" ,"--hbase-master-opts=-Xmx2048m" ,"--hbase-re  
gionserver-opts=-Xmx4096m" ]
```

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>.

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 a list of all the regions supported by Amazon EMR see [Choose an AWS Region \(p. 31\)](#).

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 on configuring bootstrap actions, see [\(Optional\) Create Bootstrap Actions to Install Additional Software \(p. 124\)](#)

For a complete list of the HBase site settings that you can configure, go to <http://hbase.apache.org/configuration.html#hbase.site>.

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 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,Args=[ "-s" ,"hbase.hregion.max.filesize=52428800" ]
```

- Windows users:

```
aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applica  
tions 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/boot  
strap-actions/configure-hbase,Args=[ "-s" ,"hbase.hregion.max.files  
ize=52428800" ]
```


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 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/my-hbase-site.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:

```
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:

```
aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applica  
tions 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/boot  
strap-actions/configure-hbase,Args=["--site-config-file","s3://mybucket/con  
fig.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 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>.

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. For more information on using the parameters in the Amazon EMR CLI, see the [Command Line Interface Reference for Amazon EMR \(p. 577\)](#).

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.

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. If you want the master server to 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 will reduce splitting, but it will take 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.

```
--bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hbase,Args=[ "-s", "hbase.hregion.memstore.flush.size=134217728" ]
```

Access HBase Data with Hive

To use Hive with HBase you'll typically want to launch two clusters, one to run HBase and the other to run Hive. Running HBase and Hive separately can improve performance because this allows HBase to fully utilize the cluster resources.

Although it is not recommended for most use cases, you can also run Hive and HBase on the same cluster.

A copy of HBase is installed on the AMI with Hive to provide connection infrastructure to access your HBase cluster. The following sections show how to use the client portion of the copy of HBase on your Hive cluster to connect to HBase on another cluster.

You can use Hive to connect to HBase and manipulate data, performing such actions as exporting data to Amazon S3, importing data from Amazon S3, and querying HBase data.

Note

You can only connect your Hive cluster to a single HBase cluster.

To connect Hive to HBase

1. Create a cluster with Hive installed using the steps in [Submit Hive Work \(p. 260\)](#) and create a cluster with HBase installed using the steps in [Install HBase on an Amazon EMR Cluster \(p. 312\)](#).
2. Use SSH to connect to the master node. For more information, see [Connect to the Master Node Using SSH \(p. 441\)](#).
3. Launch the Hive shell with the following command.

```
hive
```

4. 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`. For more information, see [To retrieve the public DNS name of the master node using the Amazon EMR console \(p. 442\)](#).

```
set hbase.zookeeper.quorum=public-DNS-name;
```

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,fam1:coll")
  tblproperties ("hbase.table.name" = "inputTable");

select count(*) from inputTable ;
```

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 information on 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 \(p. 451\)](#).
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 a tutorial on how to do this, see [Option 2, Part 2: Configure Proxy Settings to View Websites Hosted on the Master Node \(p. 453\)](#).
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. For information on how to locate the public DNS name of a master node, see [To retrieve the public DNS name of the master node using the Amazon EMR console \(p. 442\)](#).

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 for the latest log files to be 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 will not be available after the cluster ends unless you enable logging to Amazon S3 when the cluster is launched. For information about how to connect to the master node using SSH see, [Connect to the Master Node Using SSH \(p. 441\)](#). 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 on enabling logging for your cluster, see [Configure Logging and Debugging \(Optional\) \(p. 200\)](#).

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 \(p. 418\)](#).

Metric	Description
<code>HBaseBackupFailed</code>	<p>Whether the last backup failed. This is set to 0 by default and updated to 1 if the previous backup attempt failed. This metric is only reported for HBase clusters.</p> <p>Use Case: Monitor HBase backups</p> <p>Units: <i>Count</i></p>
<code>HBaseMostRecentBackupDuration</code>	<p>The amount of time it took the previous backup to complete. This metric is set regardless of whether the last completed backup succeeded or failed. While the backup is ongoing, this metric returns the number of minutes since the backup started. This metric is only reported for HBase clusters.</p> <p>Use Case: Monitor HBase Backups</p> <p>Units: <i>Minutes</i></p>

Metric	Description
HBaseTimeSinceLastSuccessful-Backup	The number of elapsed minutes since the last successful HBase backup started on your cluster. This metric is only reported for HBase clusters. Use Case: Monitor HBase backups Units: <i>Minutes</i>

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, go to <http://ganglia.info/>. For more information about using Ganglia with Amazon EMR clusters, see [Monitor Performance with Ganglia \(p. 433\)](#).

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.

Once the cluster is launched with Ganglia reporting 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 will be 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 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 a list of regions supported by Amazon EMR see [Choose an AWS Region \(p. 31\)](#).

- 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 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:

```
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
```

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 HBase metrics in the Ganglia web interface

1. Use SSH to tunnel into the master node and create a secure connection. For information on 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. 451).
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 a tutorial on how to do this, see [Option 2, Part 2: Configure Proxy Settings to View Websites Hosted on the Master Node](#) (p. 453).
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. For information on how to locate the public DNS name of a master node, see [To retrieve the public DNS name of the master node using the Amazon EMR console](#) (p. 442).

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 information about how to use SSH to connect to the master node, see [Connect to the Master Node Using SSH](#) (p. 441).

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 will be written there as well. Logs are written to Amazon S3 every five minutes, so there may be a slight delay for the latest log files to be available. For more information, see [View HBase Log Files](#) (p. 330).

Configure Hue to View, Query, or Manipulate Data

This section consists of the following topics.

Topics

- [What is Hue? \(p. 334\)](#)
- [Create a Cluster with Hue Installed \(p. 335\)](#)
- [Launch the Hue Web Interface \(p. 336\)](#)
- [Use Hue with a Remote Database in Amazon RDS \(p. 336\)](#)
- [Advanced Configurations for Hue \(p. 339\)](#)
- [Metastore Manager Restrictions \(p. 343\)](#)

What is Hue?

Hue is an open-source, web-based graphical user interface for use with Amazon Elastic MapReduce 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

To launch a cluster with Hue installed using the console

1. Navigate to **Software Configuration** and choose Amazon for **Hadoop distribution** and 3.3.0 (or later) for the **AMI version**.
2. In Software Configuration > Applications to be installed, Hue should appear in the list by default.
3. In the **Hardware Configuration** section, accept the default EC2 instance types: m3.xlarge for the master node and m1.large for the core nodes. 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.
4. 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.
5. 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 <http://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 \(p. 185\)](#).

```
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:


```
aws emr create-cluster --name "Hue 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
```

- Windows users:

```
aws emr create-cluster --name "Hue cluster" --ami-version 3.3 --applications  
Name=Hue Name=Hive Name=Pig --use-default-roles --ec2-attributes Key  
Name=myKey --instance-type m3.xlarge --instance-count 3
```

Note

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

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

Launch the Hue Web Interface

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 \(p. 446\)](#).

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 \(p. 451\)](#).
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

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 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 :

```
{
  "hue": {
    "database": {
      "name": "dbname",
      "user": "username",
      "password": "password",
      "host": "hueinstance.c3b8apyyjyzi.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:

```
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`:

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

```
{
  "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 \(p. 335\)](#).

- Linux, UNIX, and Mac OS X users:

```
aws emr create-cluster --name "Hue cluster" --ami-version 3.3 --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
```

- Windows users:

```
aws emr create-cluster --name "Hue cluster" --ami-version 3.3 --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
```

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>.

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 \(p. 340\)](#)

Configure Hue for LDAP Users

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.

```
{
  "hue": {
    "ldap": {
      "ldap_servers": {
        "yourcompany": {
          "base_dn": "DC=yourcompany,DC=hue,DC=com",
          "ldap_url": "ldap://ldapur1",
          "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": "hue",  
          "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=administrator,CN=users,DC=yourcompany,DC=hue,DC=com",  
          "bind_password": "password",  
          "ldap_username_pattern": "CN=username,OU=orgunit,DC=yourcom  
pany,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.

Type the following text if you have nested groups and would like to synchronize all users in the sub-groups when using direct bind for LDAP with `nt_domain`. This configuration uses the `subgroups` and `nested_members_search_depth` parameters.

```
{
  "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.

```
{
  "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=3.3 --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
```

- Windows users:

```
aws emr create-cluster --name "Hue cluster" --ami-version=3.3 --applications
Name=Hive Name=Pig Name=Hue,Args[--hue-config=s3://mybucket/huecon
fig.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 **Idap** section to view your settings.

Metastore Manager Restrictions

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 `iteration0` ends, it stores the ending sequence number in DynamoDB so that when the `iteration1` job begins, it can retrieve subsequent data from the stream. This mapping of iterations in stream data is called *checkpointing*. For more details, 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 Amazon 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:

Configuration Setting	Description
<code>kinesis.checkpoint.metastore.table.name</code>	DynamoDB table name where checkpoint information will be stored
<code>kinesis.checkpoint.metastore.hash.key.name</code>	Hash key name for the DynamoDB table
<code>kinesis.checkpoint.metastore.hash.range.name</code>	Range key name for the DynamoDB table
<code>kinesis.checkpoint.logical.name</code>	A logical name for current processing
<code>kinesis.checkpoint.iteration.no</code>	Iteration number for processing associated with the logical name
<code>kinesis.rerun.iteration.without.wait</code>	Boolean value that indicates if a failed iteration can be rerun without waiting for timeout; the default is <code>false</code>

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: $j*s/5$

For Write Capacity Units: j*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 Amazon 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 a 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 Amazon 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. 347)
- [Create an Amazon Kinesis Stream](#) (p. 347)
- [Create an Amazon DynamoDB Table](#) (p. 347)
- [Download Log4J Appender for Amazon Kinesis Sample Application, Sample Credentials File, and Sample Log File](#) (p. 348)
- [Start Amazon Kinesis Publisher Sample Application](#) (p. 350)
- [Launch the Cluster](#) (p. 352)

- [Run the Ad-hoc Hive Query](#) (p. 356)
- [Running Queries with Checkpoints](#) (p. 359)
- [Scheduling Scripted Queries](#) (p. 360)

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

1. Open <http://aws.amazon.com/> and click **Sign Up**.
2. Follow the on-screen 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 Amazon Kinesis 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 Amazon Kinesis, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.

To create a stream

1. Sign in to the AWS Management Console and go to the Amazon Kinesis 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 Amazon 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

Option	Default	Description
<code>log4j.appender.KINESIS.streamName</code>	AccessLogStream	Stream name to which data is to be published.
<code>log4j.appender.KINESIS.encoding</code>	UTF-8	Encoding used to convert log message strings into bytes before sending to Amazon Kinesis.
<code>log4j.appender.KINESIS.maxRetries</code>	3	Maximum number of retries when calling Kinesis APIs to publish a log message.
<code>log4j.appender.KINESIS.backoffInterval</code>	100ms	Milliseconds to wait before a retry attempt.
<code>log4j.appender.KINESIS.threadCount</code>	20	Number of parallel threads for publishing logs to configured Kinesis stream.

Amazon Elastic MapReduce Developer Guide
Download Log4J Appender for Amazon Kinesis Sample
Application, Sample Credentials File, and Sample Log
File

Option	Default	Description
<code>log4j.appender.KINESIS.bufferSize</code>	2000	Maximum number of outstanding log messages to keep in memory.
<code>log4j.appender.KINESIS.shutdown-Timeout</code>	30	Seconds to send buffered messages before application JVM quits normally.

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:

1. Download the Amazon Kinesis Log4j Appender JAR from <http://emr-kinesis.s3.amazonaws.com/publisher/kinesis-log4j-appender-1.0.0.jar>.
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?](#) 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 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.

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

Option	Default	Description
<code>log4j.appender.KINESIS.streamName</code>	AccessLogStream	Stream name to which data is to be published.
<code>log4j.appender.KINESIS.encoding</code>	UTF-8	Encoding used to convert log message strings into bytes before sending to Amazon Kinesis.

Option	Default	Description
<code>log4j.appender.KINESIS.maxRetries</code>	3	Maximum number of retries when calling Kinesis APIs to publish a log message.
<code>log4j.appender.KINESIS.backoffInterval</code>	100ms	Milliseconds to wait before a retry attempt.
<code>log4j.appender.KINESIS.threadCount</code>	20	Number of parallel threads for publishing logs to configured Kinesis stream.
<code>log4j.appender.KINESIS.bufferSize</code>	2000	Maximum number of outstanding log messages to keep in memory.
<code>log4j.appender.KINESIS.shutdownTimeout</code>	30	Seconds to send buffered messages before application JVM quits normally.

To download and configure the Amazon Kinesis Log4j Appender tool:

1. Download the Amazon Kinesis Log4j Appender JAR from <http://emr-kinesis.s3.amazonaws.com/publisher/kinesis-log4j-appender-1.0.0.jar>.
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?](#) in the *AWS General Reference*.

3. Download and save the sample access log file, `access_log_1`, from [samples/pig-apache/access_log_1.zip](#) 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.amazon  
aws.services.kinesis.log4j.FilePublisher access_log_1
```

- Windows users:

```
%JAVA_HOME%/bin/java -cp ./kinesis-log4j-appender-1.0.0.jar com.amazon  
aws.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:

1. Download the file, `publisher.sh` from <http://emr-kinesis.s3.amazonaws.com/publisher/publisher.sh>
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.

Field	Action
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. 462) . 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. 413) .

Field	Action
Log folder S3 location	<p>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.</p> <p>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.</p> <p>For more information, see View Log Files (p. 413) .</p>
Debugging	<p>This option creates a debug log index in 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 about Amazon SimpleDB, go to the Amazon SimpleDB product description page.</p>

4. In the **Software Configuration** section, verify the fields according to the following table.

Field	Action
Hadoop distribution	<p>Choose Amazon.</p> <p>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. 218) .</p>
AMI version	<p>Choose 3.0.4 (Hadoop 2.2.0).</p> <p>For more information, see Choose an Amazon Machine Image (AMI) (p. 48).</p>

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	<p>Choose Launch into EC2-Classical.</p> <p>Optionally, choose a VPC subnet identifier from the list to launch the cluster in an Amazon VPC. For more information, see Select an Amazon VPC Subnet for the Cluster (Optional) (p. 205) .</p>

Field	Action
EC2 Availability Zone	<p>Choose No preference.</p> <p>Optionally, you can launch the cluster in a specific Amazon EC2 Availability Zone.</p> <p>For more information, see Regions and Availability Zones in the Amazon EC2 User Guide.</p>
Master	<p>Choose m1.large.</p> <p>The master node assigns Hadoop tasks to core and task nodes, and monitors their status. There is always one master node in each cluster.</p> <p>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.</p> <p>This tutorial uses m1.large instances for all nodes.</p>
Request Spot Instances	<p>Leave this box unchecked.</p> <p>This specifies whether to run master nodes on Spot Instances. For more information, see (Optional) Lower Costs with Spot Instances (p. 37) .</p>
Core	<p>Choose m1.large.</p> <p>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.</p> <p>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.</p> <p>This tutorial uses m1.large instances for all nodes.</p>
Count	Choose 2 .
Request Spot Instances	<p>Leave this box unchecked.</p> <p>This specifies whether to run core nodes on Spot Instances. For more information, see (Optional) Lower Costs with Spot Instances (p. 37) .</p>
Task	<p>Choose m1.large.</p> <p>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.</p> <p>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.</p>
Count	Choose 0 .

Field	Action
Request Spot Instances	<p>Leave this box unchecked.</p> <p>This specifies whether to run task nodes on Spot Instances. For more information, see (Optional) Lower Costs with Spot Instances (p. 37) .</p>

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.

- In the **Security and Access** section, complete the fields according to the following table.

Field	Action
EC2 key pair	<p>Choose your Amazon EC2 key pair private key from the list.</p> <p>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. 441) .</p>
IAM user access	<p>Choose All other IAM users to make the cluster visible and accessible to all IAM users on the AWS account. For more information, see Configure IAM User Permissions (p. 181) .</p> <p>Alternatively, choose No other IAM users to restrict access to the current IAM user.</p>
Roles configuration	<p>Choose Default to generate the default EMR role and 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 EMR role and EC2 instance profile are required when creating a cluster using the console.</p> <p>The EMR role allows Amazon EMR to access other AWS services on your behalf. The EC2 instance profile controls application access to the Amazon EC2 instances in the cluster.</p> <p>For more information, see Configure IAM Roles for Amazon EMR (p. 185) .</p>

- In the **Bootstrap Actions** and **Steps** sections, you do not need to change any of these settings.
- Review your configuration and if you are satisfied with the settings, choose **Create Cluster**.
- 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:

```
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:

```
aws emr create-cluster --name "EmrKinesisTutorial" --ami-version 3.3 -  
-applications Name=Hue Name=Hive Name=Pig --use-default-roles --ec2-attri-  
butes 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 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 \(p. 441\)](#).
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 Amazon 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" = "([^ ]*) ([^ ]*) ([^ ]*) (-|\\|\\|\\|\\|\\|\\|)*\\|\\| ([^  
\\ ]*|\\\"[^\"]*\\\") ([0-9]*) ([0-9]*) ([^ \\\"]*|\\\"[^\"]*\\\") ([^ \\\"]*|\\\"[^\"]*\\\")"  
)
```

```

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 Amazon 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.records.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:

Configuration Setting	Default Value	Description
kinesis.stream.name		Amazon Kinesis stream name as the source of data.
kinesis.accessKey	Amazon S3 credentials set in the cluster or IAM-based role credentials if Amazon S3 credentials are absent.	AWS access key.
kinesis.secretKey	Amazon S3 credentials set in the cluster or IAM-based role credentials if Amazon S3 credentials are absent.	AWS secret key.

Configuration Setting	Default Value	Description
kinesis.nodata.timeout	5	Timeout, in minutes (integer), to finish this iteration if no data is received continuously for this duration.
kinesis.iteration.timeout	15	Maximum duration in minutes (integer) to run this iteration. The cluster would create a check-point at the end.
kinesis.records.batchsize	1000	Number of records to get from the Amazon Kinesis stream in a single GetRecords API call. Cannot be more than 10000 (limit enforced by the Amazon Kinesis API).
kinesis.endpoint.region	us-east-1	Amazon Kinesis endpoint region. You may experience better performance by choosing a Amazon Kinesis region closest to the Amazon EMR cluster region.
kinesis.retry.interval	500	Retry interval in msec (integer) for a failure when calling the Amazon KinesisAPI.
kinesis.retry.maxattempts	5	The maximum number of retries in case of a failure before giving up.

5. Run the following query to analyze the Hive table created in [Step 3 \(p. 356\)](#). This query counts number of visitors coming from Windows or Linux operating systems who got a 404 error:

```
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:

```
2014-01-23 22:37:13,773 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 30.86 sec
2014-01-23 22:37:14,807 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 30.86 sec
2014-01-23 22:37:15,839 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 30.86 sec
2014-01-23 22:37:16,870 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 30.86 sec
2014-01-23 22:37:17,907 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 30.86 sec
2014-01-23 22:37:18,942 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 30.86 sec
2014-01-23 22:37:19,978 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 32.77 sec
2014-01-23 22:37:21,013 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 32.77 sec
MapReduce Total cumulative CPU time: 32 seconds 770 msec
Ended Job = job_1390514680396_0001
Counters:
MapReduce Jobs Launched:
Job 0: Map: 2 Reduce: 1 Cumulative CPU: 32.77 sec HDFS Read: 526 HDFS Write: 21 SUCCESS
Total MapReduce CPU Time Spent: 32 seconds 770 msec
OK
Linux 20
Windows 136
Time taken: 369.149 seconds, Fetched: 2 row(s)
hive>
```

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 Amazon 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:kines
is.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 Amazon 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 \(p. 441\)](#).
2. Create a directory for all your scheduling-related resources called `/home/hadoop/crontab`:

```
% mkdir crontab
```

3. Download [executor.sh](#), [hive.config](#), [create_table.q](#), and [user_agents_count.q](#) in `/home/hadoop/crontab` using the `wget` command:

```
wget http://emr-kinesis.s3.amazonaws.com/crontab/executor.sh http://emr-
kinesis.s3.amazonaws.com/crontab/hive.config http://emr-kinesis.s3.amazon
aws.com/crontab/create_table.q http://emr-kinesis.s3.amazon
aws.com/crontab/user_agents_count.q
```

4. Edit `hive.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.

Note

If you have more than one script, you can specify a space-delimited list of pathnames. For example:

```
SCRIPTS="/home/hadoop/crontab/hivescript1 /home/ha
doop/crontab/hivescript2"
```

5. Edit `create_table.q`. At the end of the script, edit the `LOCATION`:

```
LOCATION 's3://<s3_output_path>/hive';
```

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:

```
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/hive.config  
1>>/tmp/cronlogs/hive.log 2>>/tmp/cronlogs/hive.log
```

Save and exit the editor; the crontab is updated upon exit. You can verify the crontab entries by executing `crontab -l`.

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. 361)
- [Create an Amazon Kinesis Stream](#) (p. 362)
- [Create an DynamoDB Table](#) (p. 362)
- [Download Log4J Appender for Amazon Kinesis Sample Application, Sample Credentials File, and Sample Log File](#) (p. 363)
- [Start Amazon Kinesis Publisher Sample Application](#) (p. 364)
- [Launch the Cluster](#) (p. 366)
- [Run the Pig Script](#) (p. 370)
- [Scheduling Scripted Queries](#) (p. 374)

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

1. Open <http://aws.amazon.com/> and click **Sign Up**.
2. Follow the on-screen 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 Amazon Kinesis 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 Amazon Kinesis, see [Regions and Endpoints](#) in the *Amazon Web Services General Reference*.

To create a stream

1. Sign in to the AWS Management Console and go to the Amazon Kinesis 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 Amazon 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

Option	Default	Description
<code>log4j.appender.KINESIS.streamName</code>	AccessLogStream	Stream name to which data is to be published.
<code>log4j.appender.KINESIS.encoding</code>	UTF-8	Encoding used to convert log message strings into bytes before sending to Amazon Kinesis.
<code>log4j.appender.KINESIS.maxRetries</code>	3	Maximum number of retries when calling Kinesis APIs to publish a log message.
<code>log4j.appender.KINESIS.backoffInterval</code>	100ms	Milliseconds to wait before a retry attempt.
<code>log4j.appender.KINESIS.threadCount</code>	20	Number of parallel threads for publishing logs to configured Kinesis stream.
<code>log4j.appender.KINESIS.bufferSize</code>	2000	Maximum number of outstanding log messages to keep in memory.
<code>log4j.appender.KINESIS.shutdownTimeout</code>	30	Seconds to send buffered messages before application JVM quits normally.

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:

1. Download the Amazon Kinesis Log4j Appender JAR from <http://emr-kinesis.s3.amazonaws.com/publisher/kinesis-log4j-appender-1.0.0.jar> .
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?](#) 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 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.amazon  
aws.services.kinesis.log4j.FilePublisher access_log_1
```

- Windows users:

```
%JAVA_HOME%/bin/java -cp .;kinesis-log4j-appender-1.0.0.jar com.amazon  
aws.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:

1. Download the file, `publisher.sh` from <http://emr-kinesis.s3.amazonaws.com/publisher/publisher.sh>
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.

Field	Action
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. 462) . 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. 413) .
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. 413) .
Debugging	This option creates a debug log index in 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 about Amazon SimpleDB, go to the Amazon SimpleDB product description page.

4. In the **Software Configuration** section, verify the fields according to the following table.

Field	Action
Hadoop distribution	<p>Choose Amazon.</p> <p>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.</p>
AMI version	<p>Release Label</p>

- 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	<p>Choose Launch into EC2-Classical.</p> <p>Optionally, choose a VPC subnet identifier from the list to launch the cluster in an Amazon VPC. For more information, see Select an Amazon VPC Subnet for the Cluster (Optional) (p. 205) .</p>
EC2 Availability Zone	<p>Choose No preference.</p> <p>Optionally, you can launch the cluster in a specific Amazon EC2 Availability Zone.</p> <p>For more information, see Regions and Availability Zones in the Amazon EC2 User Guide.</p>
Master	<p>Choose m1.large.</p> <p>The master node assigns Hadoop tasks to core and task nodes, and monitors their status. There is always one master node in each cluster.</p> <p>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.</p> <p>This tutorial uses m1.large instances for all nodes.</p> <p>For more information, see Instance Groups (p. 34)</p>
Request Spot Instances	<p>Leave this box unchecked.</p> <p>This specifies whether to run master nodes on Spot Instances. For more information, see (Optional) Lower Costs with Spot Instances (p. 37)</p>
Core	<p>Choose m1.large.</p> <p>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.</p> <p>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.</p> <p>This tutorial uses m1.large instances for all nodes.</p> <p>For more information, see Instance Groups (p. 34)</p>
Count	<p>Choose 2.</p>
Request Spot Instances	<p>Leave this box unchecked.</p> <p>This specifies whether to run core nodes on Spot Instances. For more information, see (Optional) Lower Costs with Spot Instances (p. 37)</p>

Field	Action
Task	<p>Choose m1.large.</p> <p>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.</p> <p>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.</p> <p>For more information, see Instance Groups (p. 34)</p>
Count	Choose 0 .
Request Spot Instances	<p>Leave this box unchecked.</p> <p>This specifies whether to run task nodes on Spot Instances. For more information, see (Optional) Lower Costs with Spot Instances (p. 37).</p>

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	<p>Choose your Amazon EC2 key pair private key from the list.</p> <p>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. 441) .</p>
IAM user access	<p>Choose All other IAM users to make the cluster visible and accessible to all IAM users on the AWS account. For more information, see Configure IAM User Permissions (p. 181) .</p> <p>Alternatively, choose No other IAM users to restrict access to the current IAM user.</p>
Roles configuration	<p>Choose Default to generate the default EMR role and 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 EMR role and EC2 instance profile are required when creating a cluster using the console.</p> <p>The EMR role allows Amazon EMR to access other AWS services on your behalf. The EC2 instance profile controls application access to the Amazon EC2 instances in the cluster.</p> <p>For more information, see Configure IAM Roles for Amazon EMR (p. 185) .</p>

7. In the **Bootstrap Actions** and **Steps** sections, you do not need to change any of these settings.

- Review your configuration and if you are satisfied with the settings, choose **Create Cluster**.
- 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:

```
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:

```
aws emr create-cluster --name "EmrKinesisTutorial" --ami-version 3.3 -
--applications Name=Hue Name=Hive Name=Pig --use-default-roles --ec2-attri
butes 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 Amazon Kinesis stream in the Grunt shell

- 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 \(p. 441\)](#).
- In the SSH window, from the home directory, start the Grunt shell by running the following command:

```
~/bin/pig
```

- Run the following script to process the data in the Amazon Kinesis stream, `AccessLogStream`, and print the agent count by operating system:

```
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.RegexEx
tract();

raw_logs = load 'AccessLogStream' using com.amazon.emr.kinesis.pig.Kin
```

```

esisStreamLoader('kinesis.iteration.timeout=1') 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+) "[^"]*" "[^"]*"')
      )
    )
  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:

Configuration Setting	Default Value	Description
kinesis.stream.name		Amazon Kinesis stream name as the source of data.
kinesis.accessKey	Amazon S3 credentials set in the cluster or IAM-based role credentials if Amazon S3 credentials are absent.	AWS access key.

Configuration Setting	Default Value	Description
kinesis.secretKey	Amazon S3 credentials set in the cluster or IAM-based role credentials if Amazon S3 credentials are absent.	AWS secret key.
kinesis.nodata.timeout	5	Timeout, in minutes (integer), to finish this iteration if no data is received continuously for this duration.
kinesis.iteration.timeout	15	Maximum duration in minutes (integer) to run this iteration. The cluster would create a checkpoint at the end.
kinesis.records.batchsize	1000	Number of records to get from the Amazon Kinesis stream in a single GetRecords API call. Cannot be more than 10000 (limit enforced by the Amazon Kinesis API).
kinesis.endpoint.region	us-east-1	Kinesis endpoint region. You may experience better performance by choosing a Amazon Kinesis region closest to the Amazon EMR cluster region.
kinesis.retry.interval	500	Retry interval in msec (integer) for a failure when calling the Amazon Kinesis API.
kinesis.retry.maxattempts	5	The maximum number of retries in case of a failure before giving up.

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 Amazon Kinesis stream and store the results in Amazon S3. Run the script as shown in the Grunt shell:

```
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+) "[^"]*" "[^"]*"')
      )
    )
  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 != '');

-- 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:

```
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:

```
Linux 137
Windows 2194
```

The file in iteration1 should contain values similar to the following:

```
Linux 73  
Windows 506
```

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 Amazon 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 \(p. 441\)](#).
2. Create a directory for all your scheduling-related resources called `/home/hadoop/crontab`:

```
% 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:

```
wget http://emr-kinesis.s3.amazonaws.com/crontab/executor.sh http://emr-kinesis.s3.amazonaws.com/crontab/pig.config http://emr-kinesis.s3.amazonaws.com/crontab/user_agents_count.pig
```

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.

Note

If you have more than one script, you can specify a space-delimited list of pathnames. For example:

```
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:

```
STORE by_agent_count into 's3://<s3_output_path>/pig/iteration_$(iteration  
Number)';
```

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:

```
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

- [Distributed Copy Using S3DistCp \(p. 376\)](#)
- [Export, Import, Query, and Join Tables in DynamoDB Using Amazon EMR \(p. 384\)](#)
- [Store Avro Data in Amazon S3 Using Amazon EMR \(p. 406\)](#)

Distributed Copy Using S3DistCp

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

Topics

- [S3DistCp Options \(p. 377\)](#)
- [Adding S3DistCp as a Step in a Cluster \(p. 381\)](#)
- [S3DistCp Versions Supported in Amazon EMR \(p. 383\)](#)

Apache DistCp is an open-source tool you can use to copy large amounts of data. DistCp 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, go to <http://hadoop.apache.org/docs/stable/hadoop-distcp/DistCp.html>.

S3DistCp is an extension of *DistCp* that is optimized to work with AWS, particularly Amazon S3. You use *S3DistCp* by adding it 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.

During a copy operation, *S3DistCp* stages a temporary copy of the output in HDFS on the cluster. There must be sufficient 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, therefore you must manually purge the temporary files. 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.

Option	Description	Re-quired
<code>--src, LOCATION</code>	Location of the data to copy. This can be either an HDFS or Amazon S3 location. Example: <code>--src, s3://myawsbucket/logs/j-3GYXXXXXX9IOJ/node</code> Important <i>S3DistCp</i> does not support Amazon S3 bucket names that contain the underscore character.	Yes
<code>--dest, LOCATION</code>	Destination for the data. This can be either an HDFS or Amazon S3 location. Example: <code>--dest, hdfs:///output</code> Important <i>S3DistCp</i> does not support Amazon S3 bucket names that contain the underscore character.	Yes

Option	Description	Re-quired
<code>--srcPattern, PATTERN</code>	<p>A regular expression that filters the copy operation to a subset of the data at <code>--src</code>. If neither <code>--srcPattern</code> nor <code>--groupBy</code> is specified, all data at <code>--src</code> is copied to <code>--dest</code>.</p> <p>If the regular expression argument contains special characters, such as an asterisk (*), either the regular expression or the entire <code>--args</code> string must be enclosed in single quotes (').</p> <p>Example: <code>--srcPattern, .*daemons.*-hadoop-.*</code></p>	No
<code>--groupBy, PATTERN</code>	<p>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.</p> <p>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 statement, the cluster fails on the S3DistCp step and return an error.</p> <p>If the regular expression argument contains special characters, such as an asterisk (*), either the regular expression or the entire <code>--args</code> string must be enclosed in single quotes (').</p> <p>When <code>--groupBy</code> is specified, only files that match the specified pattern are copied. You do not need to specify <code>--groupBy</code> and <code>--srcPattern</code> at the same time.</p> <p>Example: <code>--groupBy, .*subnetid.*([0-9]+-[0-9]+-[0-9]+-[0-9]+).*</code></p>	No
<code>--targetSize, SIZE</code>	<p>The size, in mebibytes (MiB), of the files to create based on the <code>--groupBy</code> option. This value must be an integer. When <code>--targetSize</code> is set, S3DistCp attempts to match this size; the actual size of the copied files may be larger or smaller than this value.</p> <p>If the files concatenated by <code>--groupBy</code> are larger than the value of <code>--targetSize</code>, they are broken up into part files, and named sequentially with a numeric value appended to the end. For example, a file concatenated into <code>myfile.gz</code> would be broken into parts as: <code>myfile0.gz</code>, <code>myfile1.gz</code>, etc.</p> <p>Example: <code>--targetSize, 2</code></p>	No

Option	Description	Re-quired
<code>--appendToLastFile</code>	Specifies the behavior of S3DistCp when copying to files already present. It appends new file data to existing files. If you use <code>--appendToLastFile</code> with <code>--groupBy</code> , new data is appended to files which match the same groups. This option also respects the <code>--targetSize</code> behavior when used with <code>--groupBy</code> .	No
<code>--outputCodec CODEC</code>	Specifies the compression codec to use for the copied files. This can take the values: <code>gzip</code> , <code>gz</code> , <code>lzo</code> , <code>snappy</code> , or <code>none</code> . 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 <code>gz</code> and <code>gzip</code> , the extension is <code>.gz</code>) If you do not specify a value for <code>--outputCodec</code> , the files are copied over with no change in their compression. Example: <code>--outputCodec lzo</code>	No
<code>--s3ServerSideEncryption</code>	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: <code>--s3ServerSideEncryption</code>	No
<code>--deleteOnSuccess</code>	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: <code>--deleteOnSuccess</code>	No
<code>--disableMultipartUpload</code>	Disables the use of multipart upload. Example: <code>--disableMultipartUpload</code>	No
<code>--multipartUploadChunkSize, SIZE</code>	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: <code>--multipartUploadChunkSize, 32</code>	No
<code>--numberFiles</code>	Prepends output files with sequential numbers. The count starts at 0 unless a different value is specified by <code>--startingIndex</code> . Example: <code>--numberFiles</code>	No

Amazon Elastic MapReduce Developer Guide
S3DistCp Options

Option	Description	Re-quired
<code>--startingIndex, INDEX</code>	Used with <code>--numberOfFiles</code> to specify the first number in the sequence. Example: <code>--startingIndex, 1</code>	No
<code>--outputManifest, FILENAME</code>	Creates a text file, compressed with Gzip, that contains a list of all the files copied by S3DistCp. Example: <code>--outputManifest, manifest-1.gz</code>	No
<code>--previousManifest, PATH</code>	Reads a manifest file that was created during a previous call to S3DistCp using the <code>--outputManifest</code> flag. When the <code>--previousManifest</code> flag is set, S3DistCp excludes the files listed in the manifest from the copy operation. If <code>--outputManifest</code> is specified along with <code>--previousManifest</code> , files listed in the previous manifest also appear in the new manifest file, although the files are not copied. Example: <code>--previousManifest, /usr/bin/manifest-1.gz</code>	No
<code>--requirePreviousManifest</code>	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.	No
<code>--copyFromManifest</code>	Reverses the behavior of <code>--previousManifest</code> 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: <code>--copyFromManifest --previousManifest, /usr/bin/manifest-1.gz</code>	No
<code>--s3Endpoint, ENDPOINT</code>	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 <code>s3.amazonaws.com</code> . For a list of the Amazon S3 endpoints, see Regions and Endpoints . Example: <code>--s3Endpoint, s3-eu-west-1.amazonaws.com</code>	No
<code>--storageClass, CLASS</code>	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: <code>--storageClass, STANDARD</code>	No

Option	Description	Re-quired
<code>--srcPrefixesFile,PATH</code>	<p>a text file in Amazon S3 (s3://), HDFS (hdfs://) or local file system (file:/) that contains a list of <code>src</code> prefixes, one prefix per line.</p> <p>If <code>srcPrefixesFile</code> is provided, S3DistCp will not list the <code>src</code> 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 <code>src</code> path, instead of these prefixes, will be used to generate the destination paths. If <code>srcPattern</code> is also specified, it will be applied to the combined list results of the source prefixes to further filter the input. If <code>copyFromManifest</code> is used, objects in the manifest will be copied and <code>srcPrefixesFile</code> will be ignored.</p> <p>Example: <code>--srcPrefixesFile,PATH</code></p>	No

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. 473) .

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. 377).

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:

```
aws emr add-steps --cluster-id j-3GYXXXXXX9IOK --steps Type=CUS
TOM_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:

```
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:

```
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=["--src,s3://mybucket/logs/j-3GYXXXXXX9IOJ/node/","--dest,hdfs:///output","--srcPattern,.*daemons.*-hadoop.*"]
```

- Windows users:

```
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=["--src,s3://mybucket/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 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 \(p. 174\)](#).

To load Amazon CloudFront logs into HDFS, 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:

```
aws emr add-steps --cluster-id j-3GYXXXXXX9IOK --steps Type=CUS
TOM_JAR,Name="S3DistCp step",Jar=/home/hadoop/lib/emr-s3distcp-1.0.jar,\
Args=[ "--src,s3://mybucket/cf", "--dest,hdfs:///local", "--groupBy,.*XAB
CD12345678.([0-9]+-[0-9]+-[0-9]+-[0-9]+).*", "--targetSize,128", "--outputCo
dec,lzo", "--deleteOnSuccess" ]
```

- Windows users:

```
aws emr add-steps --cluster-id j-3GYXXXXXX9IOK --steps Type=CUS
TOM_JAR,Name="S3DistCp step",Jar=/home/hadoop/lib/emr-s3distcp-
1.0.jar,Args=[ "--src,s3://mybucket/cf", "--dest,hdfs:///local", "--groupBy,.*XAB
CD12345678.([0-9]+-[0-9]+-[0-9]+-[0-9]+).*", "--targetSize,128", "--outputCo
dec,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
s3://myawsbucket/cf/XABCD12345678.2012-02-23-02.LpLfuShd.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
```

S3DistCp Versions Supported in Amazon EMR

Amazon EMR supports the following versions of S3DistCp.

Version	Description	Release Date
1.0.8	Adds the <code>--appendToLastFile</code> , <code>--requirePreviousManifest</code> , and <code>--storageClass</code> options.	3 January 2014
1.0.7	Adds the <code>--s3ServerSideEncryption</code> option.	2 May 2013
1.0.6	Adds the <code>--s3Endpoint</code> option.	6 August 2012
1.0.5	Improves the ability to specify which version of S3DistCp to run.	27 June 2012
1.0.4	Improves the <code>--deleteOnSuccess</code> option.	19 June 2012
1.0.3	Adds support for the <code>--numberFiles</code> and <code>--startingIndex</code> options.	12 June 2012
1.0.2	Improves file naming when using groups.	6 June 2012
1.0.1	Initial release of S3DistCp.	19 January 2012

Note

S3DistCp versions after 1.0.7 are found 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 \(p. 385\)](#)
- [Step 1: Create a Key Pair \(p. 386\)](#)
- [Create a Cluster \(p. 386\)](#)
- [Step 3: SSH into the Master Node \(p. 390\)](#)
- [Set Up a Hive Table to Run Hive Commands \(p. 392\)](#)
- [Hive Command Examples for Exporting, Importing, and Querying Data in DynamoDB \(p. 396\)](#)
- [Optimizing Performance for Amazon EMR Operations in DynamoDB \(p. 403\)](#)

In the following sections, you will learn how to use Amazon Elastic MapReduce (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 about DynamoDB go to the [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 \(p. 260\)](#).

Warning

Amazon EMR read or write operations on an 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 \(p. 404\)](#) or [Write Percent Setting \(p. 404\)](#) to a lower value to throttle the Amazon EMR operations. For information about DynamoDB throughput settings, see [Provisioned Throughput](#).

Prerequisites for Integrating Amazon EMR with DynamoDB

To use Amazon EMR (Amazon EMR) and Hive to manipulate data in DynamoDB, you need the following:

- An Amazon Web Services account. If you do not have one, you can get an account by going to <http://aws.amazon.com>, and clicking **Create an AWS Account**.
- A DynamoDB table that contains data on the same account used with Amazon EMR.
- A customized version of Hive that includes connectivity to DynamoDB. The latest version of Hive provided by Amazon EMR is available by default when you launch an Amazon EMR cluster from the AWS Management Console. For more information about Amazon EMR AMIs and Hive versioning, go to [Specifying the Amazon EMR AMI Version](#) and to [Configuring Hive](#) in the *Amazon EMR Developer Guide*.
- Support for DynamoDB connectivity. This is included in the Amazon EMR AMI version 2.0.2 or later.
- (Optional) An Amazon S3 bucket. For instructions about how to create a bucket, see [Get Started With Amazon Simple Storage Service](#). This bucket is used as a destination when exporting DynamoDB data to Amazon S3 or as a location to store a Hive script.
- (Optional) A Secure Shell (SSH) client application to connect to the master node of the Amazon EMR cluster and run HiveQL queries against the DynamoDB data. SSH is used to run Hive interactively. You can also save Hive commands in a text file and have Amazon EMR run the Hive commands from

the script. In this case an SSH client is not necessary, though the ability to SSH into the master node is useful even in non-interactive clusters, for debugging purposes.

An SSH client is available by default on most Linux, Unix, and Mac OS X installations. Windows users can install and use the [PuTTY](#) client, which has SSH support.

- (Optional) An Amazon EC2 key pair. This is only required for interactive clusters. The key pair provides the credentials the SSH client uses to connect to the master node. If you are running the Hive commands from a script in an Amazon S3 bucket, an EC2 key pair is optional.

Step 1: Create a Key Pair

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

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 \(p. 386\)](#)

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.



Field	Action
Cluster name	<p>Enter a descriptive name for your cluster.</p> <p>The name is optional, and does not need to be unique.</p>
Termination protection	<p>Choose Yes.</p> <p>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 <i>Amazon EMR Developer Guide</i>. Typically, set this value to Yes 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.</p>
Logging	<p>Choose Enabled.</p> <p>This determines whether Amazon EMR captures detailed log data to Amazon S3.</p> <p>For more information, see View Log Files in the <i>Amazon EMR Developer Guide</i>.</p>
Log folder S3 location	<p>Enter an Amazon S3 path to store your debug logs if you enabled logging in the previous field.</p> <p>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.</p> <p>For more information, see View Log Files in the <i>Amazon EMR Developer Guide</i>.</p>
Debugging	<p>Choose Enabled.</p> <p>This option creates a debug log index in 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 about Amazon SimpleDB, go to the Amazon SimpleDB product description page.</p>

4. In the **Software Configuration** section, verify the fields according to the following table.



Field	Action
Hadoop distribution	<p>Choose Amazon.</p> <p>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 <i>Amazon EMR Developer Guide</i>.</p>

Field	Action
AMI version	<p>Choose the latest AMI version in the list.</p> <p>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 <i>Amazon EMR Developer Guide</i>.</p>
Applications to be installed - Hive	<p>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.</p> <p>For more information, see Analyze Data with Hive in the <i>Amazon EMR Developer Guide</i>.</p>
Applications to be installed - Pig	<p>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.</p> <p>For more information, see Process Data with Pig in the <i>Amazon EMR Developer Guide</i>.</p>

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](#).



Field	Action
Network	<p>Choose Launch into EC2-Classical.</p> <p>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 <i>Amazon EMR Developer Guide</i>.</p>
EC2 Availability Zone	<p>Choose No preference.</p> <p>Optionally, you can launch the cluster in a specific EC2 Availability Zone.</p> <p>For more information, see Regions and Availability Zones in the Amazon EC2 User Guide.</p>
Master - Amazon EC2 Instance Type	<p>For this tutorial, use the default EC2 instance type that is shown in this field.</p> <p>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.</p> <p>For more information, see Instance Groups in the <i>Amazon EMR Developer Guide</i>.</p>

Field	Action
Request Spot Instances	<p>Leave this box unchecked.</p> <p>This specifies whether to run master nodes on Spot Instances. For more information, see Lower Costs with Spot Instances (Optional) in the <i>Amazon EMR Developer Guide</i>.</p>
Core - Amazon EC2 Instance Type	<p>For this tutorial, use the default EC2 instance type that is shown in this field.</p> <p>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.</p> <p>For more information, see Instance Groups in the <i>Amazon EMR Developer Guide</i>.</p>
Count	Choose 2 .
Request Spot Instances	<p>Leave this box unchecked.</p> <p>This specifies whether to run core nodes on Spot Instances. For more information, see Lower Costs with Spot Instances (Optional) in the <i>Amazon EMR Developer Guide</i>.</p>
Task - Amazon EC2 Instance Type	<p>For this tutorial, use the default EC2 instance type that is shown in this field.</p> <p>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.</p> <p>For more information, see Instance Groups in the <i>Amazon EMR Developer Guide</i>.</p>
Count	Choose 0 .
Request Spot Instances	<p>Leave this box unchecked.</p> <p>This specifies whether to run task nodes on Spot Instances. For more information, see Lower Costs with Spot Instances (Optional) in the <i>Amazon EMR Developer Guide</i>.</p>

6. In the **Security and Access** section, complete the fields according to the following table.

Field	Action
EC2 key pair	<p>Choose the key pair that you created in Step 1: Create a Key Pair (p. 386).</p> <p>For more information, see Create SSH Credentials for the Master Node in the <i>Amazon EMR Developer Guide</i>.</p> <p>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 <i>Amazon EMR Developer Guide</i>.</p>

Field	Action
IAM user access	Choose No other IAM users . Optionally, choose All other IAM users 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 <i>Amazon EMR Developer Guide</i> .
IAM role	Choose Proceed without roles . This controls application access to the EC2 instances in the cluster. For more information, see Configure IAM Roles for Amazon EMR in the <i>Amazon EMR Developer Guide</i> .

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.



Step 3: SSH into the Master Node

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

1. Download PuTTYgen.exe and PuTTY.exe to your computer from <http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>.
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

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/>.

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 \(p. 441\)](#).
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.

```
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_attri  
ute1_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 `dynamodhtable1`. The DynamoDB table `dynamodhtable1` 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 ar  
ray<string>)  
STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'  
TBLPROPERTIES ("dynamodb.table.name" = "dynamodhtable1",  
"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 *dynamoddbtable1* 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:

$$21,474,836,480 / 409,600 = 52,429 \text{ seconds} = 14.56 \text{ hours}$$

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.

- 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 *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 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.

Hive type	DynamoDB type
string	string (S)
bigint or double	number (N)
binary	binary (B)
array	number set (NS), string set (SS), or binary set (BS)

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.

Hive Options	Description
<code>dynamodb.throughput.read.percent</code>	<p>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.</p> <p>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. 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.</p> <p>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.</p>
<code>dynamodb.throughput.write.percent</code>	<p>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.</p> <p>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. 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</p> <p>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.</p>
<code>dynamodb.endpoint</code>	Specify the endpoint in case you have tables in different regions. For more information about the available DynamoDB endpoints, see Regions and Endpoints .
<code>dynamodb.max.map.tasks</code>	Specify the maximum number of map tasks when reading data from DynamoDB. This value must be equal to or greater than 1.
<code>dynamodb.retry.duration</code>	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.

These options are set using the `SET` command as shown in the following example.

```
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.

```
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

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 on creating and deleting tables in DynamoDB, go to [Working with Tables in DynamoDB](#).

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 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" = "dynamodhtable1",
"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" = "dynamodhtable1",
"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'
```

Amazon Elastic MapReduce Developer Guide

Hive Command Examples for Exporting, Importing, and Querying Data

```
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.

```
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 ar
ray<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.

```
CREATE EXTERNAL TABLE hiveTableName (col1 string, col2 bigint, col3 array<string>)
STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'
TBLPROPERTIES ("dynamodb.table.name" = "dynamodhtable1",
"dynamodb.column.mapping" = "col1:name,col2:year,col3:holidays");

SET dynamodb.throughput.read.percent=1.0;

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" = "dynamodhtable1",
"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" = "dynamodbttable1",
"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" = "dynamodbttable1");

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" = "dynamodbttable1",
"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'
TBLPROPERTIES ("dynamodb.table.name" = "Purchases",
"dynamodb.column.mapping" = "customerId:CustomerId,total_cost:Cost,items_purchased:Items");

SELECT max(total_cost) from hive_purchases where customerId = 717;
```

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.

```
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.

```
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_pur  
chased: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,custom  
erAddress: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.

```
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_pur  
chased: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

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:

$$21,474,836,480 / 409,600 = 52,429 \text{ seconds} = 14.56 \text{ hours}$$

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 re-launching 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, go to [Hadoop Configuration Reference](#). There, you will find a "Task Configuration" section for each of the supported configurations.

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 \(p. 306\)](#). 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:

```
{ "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 \(p. 441\)](#).
4. Launch the grunt shell:

```
$ pig
2014-03-21 16:50:29,565 [main] INFO org.apache.pig.Main - Apache Pig version
0.11.1.1-amzn (reexported) compiled Aug 03 2013, 22:52:20
2014-03-21 16:50:29,565 [main] INFO org.apache.pig.Main - Logging error
messages to: /home/hadoop/pig_1395420629558.log
```

```
2014-03-21 16:50:29,662 [main] INFO org.apache.pig.impl.util.Utils - Default
  bootstrap file /home/hadoop/.pigbootstrap not found
2014-03-21 16:50:29,933 [main] INFO org.apache.pig.backend.hadoop.execution
engine.HExecutionEngine - Connecting to hadoop file system at: hd
fs://172.31.17.132:9000
2014-03-21 16:50:30,696 [main] INFO org.apache.pig.backend.hadoop.execution
engine.HExecutionEngine - Connecting to map-reduce job tracker at:
172.31.17.132:9001
grunt>
```

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

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 \(p. 408\)](#)
- [Connect to the Cluster \(p. 441\)](#)
- [Control Cluster Termination \(p. 458\)](#)
- [Resize a Running Cluster \(p. 465\)](#)
- [Cloning a Cluster Using the Console \(p. 472\)](#)
- [Submit Work to a Cluster \(p. 473\)](#)
- [Automate Recurring Clusters with AWS Data Pipeline \(p. 477\)](#)

View and Monitor a Cluster

Amazon EMR provides several tools you can use to gather information about your cluster. You can access information about the cluster from the console, the CLI or programmatically. The standard Hadoop web interfaces and log files are available on the master node. You can also use monitoring services such as CloudWatch and Ganglia to track the performance of your cluster.

Topics

- [View Cluster Details \(p. 409\)](#)
- [View Log Files \(p. 413\)](#)
- [View Cluster Instances in Amazon EC2 \(p. 417\)](#)
- [Monitor Metrics with CloudWatch \(p. 418\)](#)
- [Logging Amazon Elastic MapReduce API Calls in AWS CloudTrail \(p. 431\)](#)
- [Monitor Performance with Ganglia \(p. 433\)](#)

View Cluster Details

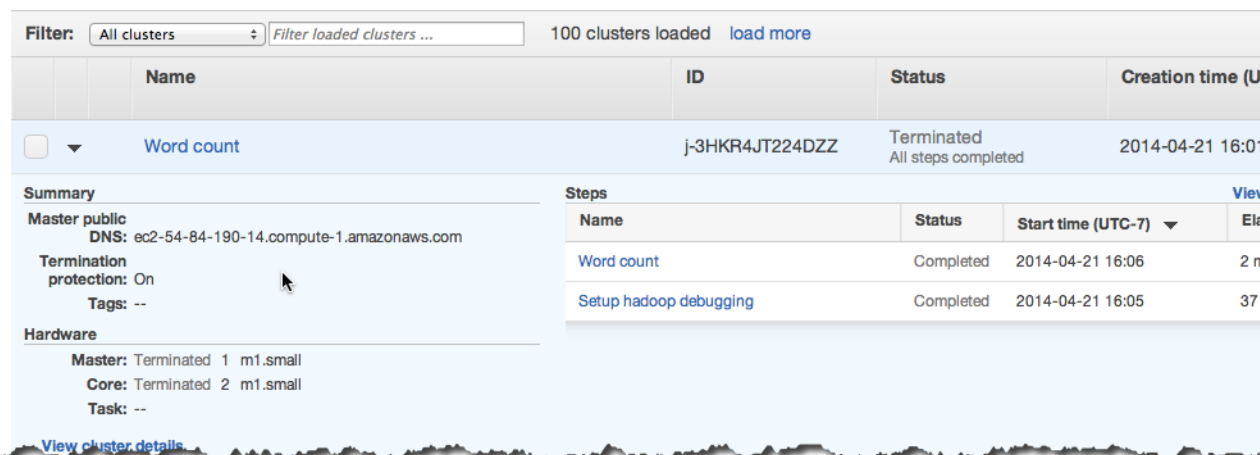
This documentation is for AMI versions 2.x and 3.x of Amazon EMR.

After you start a cluster, you can monitor its status and retrieve extended information about its execution. This section describes the methods used to view the details of Amazon EMR clusters. You can view clusters in any state.

This procedure explains how to view the details of a cluster using the Amazon EMR console.

To view cluster details using the console

1. Sign in to the AWS Management Console and open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. You have the option to view details in the **Cluster List** page. By selecting the arrow icon next to each cluster, the row view expands and gives some details and actions for the cluster you chose:



The screenshot shows the Amazon EMR console interface. At the top, there is a filter section with a dropdown menu set to 'All clusters', a search box containing 'Filter loaded clusters ...', and a status indicator '100 clusters loaded' with a 'load more' link. Below this is a table with columns for Name, ID, Status, and Creation time (UTC). The first row is expanded to show details for a cluster named 'Word count' with ID 'j-3HKR4JT224DZZ'. The status is 'Terminated' with the sub-status 'All steps completed'. The creation time is '2014-04-21 16:05'. The expanded view is divided into two main sections: 'Summary' and 'Steps'. The 'Summary' section includes 'Master public DNS: ec2-54-84-190-14.compute-1.amazonaws.com', 'Termination protection: On', 'Tags: --', and 'Hardware' information: 'Master: Terminated 1 m1.small', 'Core: Terminated 2 m1.small', and 'Task: --'. The 'Steps' section is a table with columns for Name, Status, Start time (UTC-7), and Elapsed time. It lists two steps: 'Word count' (Completed, 2014-04-21 16:06) and 'Setup hadoop debugging' (Completed, 2014-04-21 16:05). A 'View cluster details' link is visible at the bottom of the expanded view.

Name	ID	Status	Creation time (UTC)
Word count	j-3HKR4JT224DZZ	Terminated All steps completed	2014-04-21 16:05

Summary

Master public
DNS: ec2-54-84-190-14.compute-1.amazonaws.com

Termination protection: On
Tags: --

Hardware

Master: Terminated 1 m1.small
Core: Terminated 2 m1.small
Task: --

Steps

Name	Status	Start time (UTC-7)	Elapsed time
Word count	Completed	2014-04-21 16:06	2 m
Setup hadoop debugging	Completed	2014-04-21 16:05	37 s

3. For more details, choose the cluster link to open the **Cluster Details** page. The **Summary** section displays detailed information about the selected cluster.

Summary: Word count

Status:	Starting	Key name:	--
Name:	Word count	Subnet ID:	--
ID:	j-3HJGUSHI3YN5P	IAM role:	--
Auto-terminate:	No	Visible to all users:	None
Creation date:	2013-11-04 15:19:19 (local time - UTC-8)	AMI version:	2.4.2
End date:	--	Hadoop distribution:	Amazon
Master public DNS name:	ec2-54-205-127-15.compute-1.amazonaws.com	Termination protection:	On
Log URI:	s3n://examples-bucket/		
Applications:			

▶ Monitoring

▶ Hardware Configuration

The following examples demonstrate how to retrieve cluster details using the AWS CLI and the Amazon EMR CLI.

To view cluster details using the AWS CLI

Use the `describe-cluster` command to view cluster-level details including status, hardware and software configuration, VPC settings, bootstrap actions, instance groups and so on.

- To use the `describe-cluster` command, you need the cluster ID. The cluster ID can be retrieved using the `list-clusters` command. To view cluster details, type the following command and replace `j-1K48XXXXXXHCB` with your cluster ID.

```
aws emr describe-cluster --cluster-id j-1K48XXXXXXHCB
```

The output is similar to the following:

```
{
  "Cluster": {
    "Status": {
      "Timeline": {
        "ReadyDateTime": 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": "MASTER",
      "InstanceGroupType": "MASTER",
      "InstanceType": "m3.xlarge",
      "Id": "ig-26LXXXXXXFCXQ",
      "Market": "ON_DEMAND",
      "RunningInstanceCount": 0
    }
  ],
  "Applications": [
    {
      "Version": "2.4.0",
      "Name": "hadoop"
    }
  ],
  "MasterPublicDnsName": "ec2-XX-XX-XXX-XX.us-west-2.compute.amazon
```

```
aws.com",
  "VisibleToAllUsers": true,
  "BootstrapActions": [
    {
      "Args": [],
      "Name": "Install Ganglia",
      "ScriptPath": "s3://us-west-2.elasticmapreduce/bootstrap-
actions
/install-ganglia"
    }
  ],
  "RequestedAmiVersion": "3.1.0",
  "AutoTerminate": false,
  "Id": "j-Z2OXXXXXXI45"
}
```

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

To list clusters by creation date using the AWS CLI

To list clusters by creation date, type the `list-clusters` command with the `--created-after` and `--created-before` parameters.

- To list clusters created between 10-09-2014 and 10-12-2014, type the following command.

```
aws emr list-clusters --created-after 2014-10-09T00:12:00 --created-before
2014-10-12T00:12:00
```

The output lists all clusters created between the specified timestamps.

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

Note

Amazon EMR preserves metadata information about completed clusters for your reference, at no charge, for two months. The console does not provide a way to delete completed clusters from the console; these are automatically removed for you after two months.

To list clusters by state using the AWS CLI

To list clusters by state, type the `list-clusters` command with the `--cluster-states` parameter. Valid cluster states include: `STARTING`, `BOOTSTRAPPING`, `RUNNING`, `WAITING`, `TERMINATING`, `TERMINATED`, and `TERMINATED_WITH_ERRORS`.

You can also use several shortcut parameters to view clusters. The `--active` parameter filters clusters in the `STARTING`, `BOOTSTRAPPING`, `RUNNING`, `WAITING`, or `TERMINATING` states. The `--terminated` parameter filters clusters in the `TERMINATED` state. The `--failed` parameter filters clusters in the `TERMINATED_WITH_ERRORS` state.

- Type the following command to list all `TERMINATED` clusters.

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

Or:

```
aws emr list-clusters --terminated
```

The output lists all clusters in the state specified.

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

For more information about the input parameters unique to `DescribeJobFlows`, see [DescribeJobFlows](#).

View Log Files

Amazon EMR and Hadoop both produce log files that report status on the cluster. By default, these are written to the master node in the `/mnt/var/log/` directory. Depending on how you configured your cluster when you launched it, these logs may also be archived to Amazon S3 and may be viewable through the graphical debugging tool.

There are many types of logs written to the master node. Amazon EMR writes step, bootstrap action, and instance state logs. Apache Hadoop writes logs to report the processing of jobs, tasks, and task attempts. Hadoop also records logs of its daemons. For more information about the logs written by Hadoop, go to <http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/ClusterSetup.html>.

Topics

- [View Log Files on the Master Node \(p. 413\)](#)
- [View Log Files Archived to Amazon S3 \(p. 414\)](#)
- [View Log Files in the Debugging Tool \(p. 416\)](#)

View Log Files on the Master Node

The following table lists some of the log files you'll find on the master node.

Location	Description
<code>/mnt/var/log/bootstrap-actions</code>	Logs written during the processing of the bootstrap actions.
<code>/mnt/var/log/hadoop-state-pusher</code>	Logs written by the Hadoop state pusher process.
<code>/mnt/var/log/instance-controller</code>	Instance controller logs.
<code>/mnt/var/log/instance-state</code>	Instance state logs. These contain information about the CPU, memory state, and garbage collect-or threads of the node.
<code>/mnt/var/log/service-nanny</code>	Logs written by the service nanny process.
<code>/mnt/var/log/hadoop</code>	Hadoop logs, such as those written by the jobtracker and namenode processes.

Location	Description
<code>/mnt/var/log/hadoop/steps/<i>N</i></code>	<p>Step logs that contain information about the processing of the step. The value of <i>N</i> indicates the stepID assigned by Amazon EMR. For example, a cluster has two steps: <code>s-1234ABCDEFGH</code> and <code>s-5678IJKLMNOP</code>. The first step is located in <code>/mnt/var/log/hadoop/steps/s-1234ABCDEFGH/</code> and the second step in <code>/mnt/var/log/hadoop/steps/s-5678IJKLMNOP/</code>.</p> <p>The step logs written by Amazon EMR are as follows.</p> <ul style="list-style-type: none">• controller — Information about the processing of the step. If your step fails while loading, you can find the stack trace in this log.• syslog — Describes the execution of Hadoop jobs in the step.• stderr — The standard error channel of Hadoop while it processes the step.• stdout — The standard output channel of Hadoop while it processes the step.

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 \(p. 441\)](#).
2. Navigate to the directory that contains the log file information you wish to view. The preceding table gives a list of the types of log files that are available and where you will find them. The following example shows the command for navigating to the step log with an ID, `s-1234ABCDEFGH`.

```
cd /mnt/var/log/hadoop/steps/s-1234ABCDEFGH/
```

3. Use a text editor installed on the master node to view the contents of the log file. There are several you can choose from: `vi`, `nano`, and `emacs`. The following example shows how to open the controller step log using the `nano` text editor.

```
nano controller
```

View Log Files Archived to Amazon S3

By default, Amazon EMR clusters launched using the console automatically archive log files to Amazon S3. You can specify your own log path, or you can allow the console to automatically generate a log path for you. For clusters launched using the CLI or API, you must configure Amazon S3 log archiving manually.

When Amazon EMR is configured to archive log files to Amazon S3, it stores the files in the S3 location you specified, in the `/JobFlowId/` folder, where `JobFlowId` is the cluster identifier.

The following table lists some of the log files you'll find on Amazon S3.

Location	Description
<code>/JobFlowId/daemons/</code>	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.
<code>/JobFlowId/jobs/</code>	Job logs and the configuration XML file for each Hadoop job.
<code>/JobFlowId/node/</code>	Node logs, including bootstrap action, instance state, and application logs for the node. The logs for each node are stored in a folder labeled with the identifier of the EC2 instance of that node.
<code>/JobFlowId/steps/<i>N</i>/</code>	<p>Step logs that contain information about the processing of the step. The value of <i>N</i> indicates the stepId assigned by Amazon EMR. For example, a cluster has two steps: <code>s-1234ABCDEFGH</code> and <code>s-5678IJKLMNOP</code>. The first step is located in <code>/mnt/var/log/hadoop/steps/s-1234ABCDEFGH/</code> and the second step in <code>/mnt/var/log/hadoop/steps/s-5678IJKLMNOP/</code>.</p> <p>The step logs written by Amazon EMR are as follows.</p> <ul style="list-style-type: none"> • controller — Information about the processing of the step. If your step fails while loading, you can find the stack trace in this log. • syslog — Describes the execution of Hadoop jobs in the step. • stderr — The standard error channel of Hadoop while it processes the step. • stdout — The standard output channel of Hadoop while it processes the step.
<code>/JobFlowId/task-attempts/</code>	Task attempt logs. The logs for each task attempt are stored in a folder labeled with the identifier of the corresponding job.

To view log files archived to Amazon S3 using the console

1. Sign in to the AWS Management Console and open the Amazon S3 console at <https://console.aws.amazon.com/s3/>.
2. Open the S3 bucket specified when you configured the cluster to archive log files in Amazon S3.
3. Navigate to the log file containing the information to display. The preceding table gives a list of the types of log files that are available and where you will find them.
4. Double-click on a log file to view it in the browser.

If you don't want to view the log files in the Amazon S3 console, you can download the files from Amazon S3 to your local machine using a tool such as the Amazon S3 Organizer plug-in for the Firefox web browser, or by writing an application to retrieve the objects from Amazon S3. For more information, see [Getting Objects](#) in the *Amazon Simple Storage Service Developer Guide*.

View Log Files in the Debugging Tool

Amazon EMR does not automatically enable the debugging tool. You must configure this when you launch the cluster.

To view cluster logs using the console

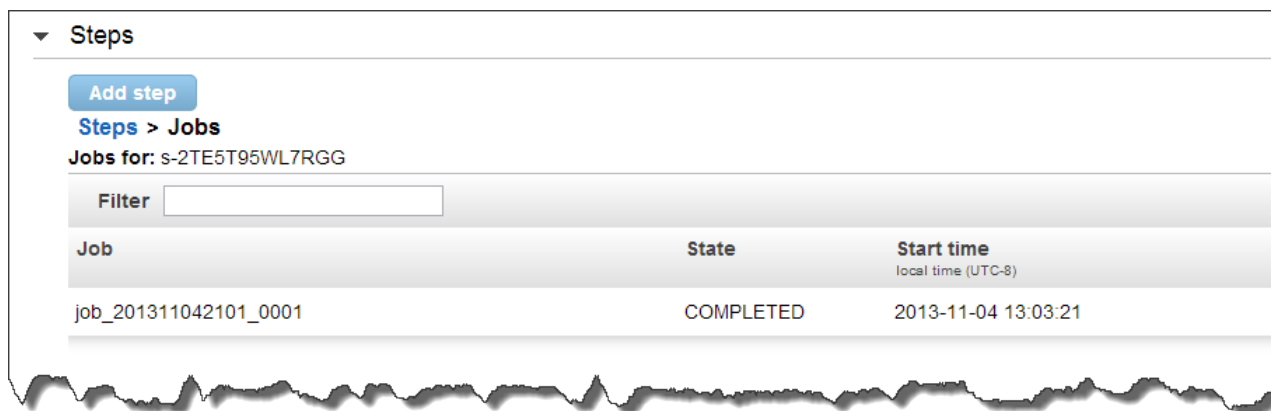
1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. From the **Cluster List** page, choose the details icon next to the cluster you want to view.

This brings up the **Cluster Details** page. In the **Steps** section, the links to the right of each step display the various types of logs available for the step. These logs are generated by Amazon EMR.

3. To view a list of the Hadoop jobs associated with a given step, choose the **View Jobs** link to the right of the step.



4. To view a list of the Hadoop tasks associated with a given job, choose the **View Tasks** link to the right of the job.



5. To view a list of the attempts a given task has run while trying to complete, choose the **View Attempts** link to the right of the task.

Steps

[Add step](#)

[Steps](#) > [Jobs](#) > [Tasks](#)

Tasks for: s-2TE5T95WL7RGG, Job 201311042101_0001

Task summary: 16 Total tasks - 16 Completed, 0 Running, 0 Failed, 0 Pending, 0 Cancelled.

Filter

Task	Type	State	Start time <small>local time (UTC-8)</small>
r_000002	reduce	COMPLETED	2013-11-04 13:05:26
r_000001	reduce	COMPLETED	2013-11-04 13:04:17
r_000000	reduce	COMPLETED	2013-11-04 13:04:15
m_000012	map	COMPLETED	2013-11-04 13:05:08

- To view the logs generated by a task attempt, choose the **stderr**, **stdout**, and **syslog** links to the right of the task attempt.

Steps

[Add step](#)

[Steps](#) > [Jobs](#) > [Tasks](#) > [Task attempts](#)

Attempts for: s-7NLV8B2TNWVE, Job 201311042021_0001, Task r_000001

Filter

Attempt	Type	State	Log fi
0	reduce	SUCCEEDED	contro

The debugging tool displays links to the log files after Amazon EMR uploads the log files to your bucket on Amazon S3. Because log files are uploaded to Amazon S3 every 5 minutes, it can take a few minutes for the log file uploads to complete after the step completes.

Amazon EMR periodically updates the status of Hadoop jobs, tasks, and task attempts in the debugging tool. You can click **Refresh List** in the debugging panes to get the most up-to-date status of these items.

View Cluster Instances in Amazon EC2

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:

Key	Value
aws:elasticmapreduce:job-flow-id	<job-flow-identifier>
aws:elasticmapreduce:instance-group-role	<group-role>

The values are further defined as follows:

- The <job-flow-identifier> is the ID of the cluster the instance is provisioned for. It appears in the format j-XXXXXXXXXXXXXXXX.
- 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.

Monitor Metrics with CloudWatch

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

When you're running a cluster, you often want to track its progress and health. Amazon EMR records metrics that can help you monitor your cluster. It makes these metrics available in the Amazon EMR console and in the CloudWatch console, where you can track them with your other AWS metrics. In CloudWatch, you can set alarms to warn you if a metric goes outside parameters you specify.

Metrics are updated every five minutes. This interval is not configurable. Metrics are archived for two weeks; after that period, the data is discarded.

These metrics are automatically collected and pushed to CloudWatch for every Amazon EMR cluster. There is no charge for the Amazon EMR metrics reported in CloudWatch; they are provided as part of the Amazon EMR service.

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. 48).

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 \(p. 422\)](#).

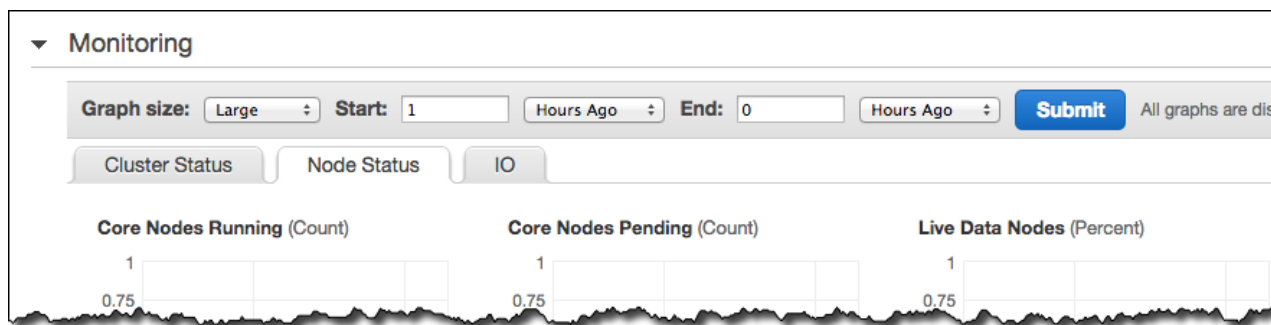
How do I?	Relevant Metrics
Track the progress of my cluster	Look at the <code>RunningMapTasks</code> , <code>RemainingMapTasks</code> , <code>RunningReduceTasks</code> , and <code>RemainingReduceTasks</code> metrics.
Detect clusters that are idle	The <code>IsIdle</code> metric tracks whether a cluster is live, but not currently running tasks. You can set an alarm to fire when the cluster has been idle for a given period of time, such as thirty minutes.
Detect when a node runs out of storage	The <code>HDFSUtilization</code> metric is the percentage of disk space currently used. If this rises above an acceptable level for your application, such as 80% of capacity used, you may need to resize your cluster and add more core nodes.

Access 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. Sign in to the AWS Management Console and open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. To view metrics for a cluster, click a cluster to display the **Summary** pane.
3. Select the **Monitoring** tab to view information about that cluster. Click any one of the tabs named **Cluster Status**, **Map/Reduce**, **Node Status**, **IO**, or **HBase** to load the reports about the progress and health of the cluster.
4. After you choose a metric to view, you can select a graph size. Edit **Start** and **End** fields to filter the metrics to a specific time frame.



To view metrics in the CloudWatch console

1. Sign in to the AWS Management Console and open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, click **EMR**.
3. Scroll down to the metric to graph. You can search on the cluster identifier of the cluster to monitor.

Dashboard
Alarms
ALARM 0
INSUFFICIENT 4
OK 0
Billing
Metrics
Selected Metrics
DynamoDB
EBS
EC2
EMR
RDS
Redshift

Browse Metrics Search Metrics X EMR Metrics

Showing the first 200 matching metrics. 4 additional metrics not listed for *EMR Metrics*.
Browse Metrics button above.

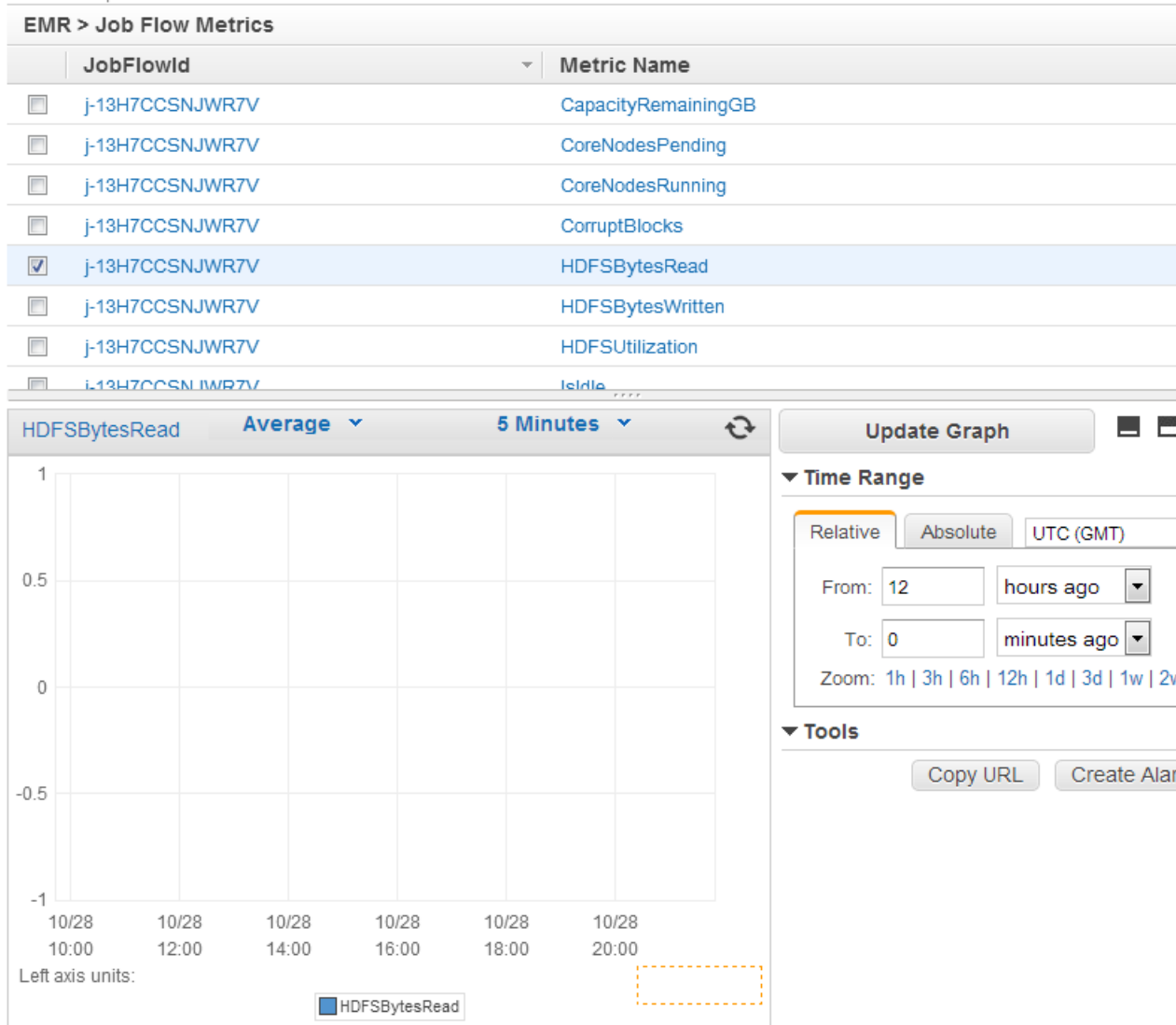
Select All | Clear

EMR > Job Flow Metrics

JobFlowId	Metric Name
j-13H7CCSNJWR7V	CapacityRemainingGB
j-13H7CCSNJWR7V	CoreNodesPending
j-13H7CCSNJWR7V	CoreNodesRunning
j-13H7CCSNJWR7V	CorruptBlocks
j-13H7CCSNJWR7V	HDFSBytesRead
j-13H7CCSNJWR7V	HDFSBytesWritten
j-13H7CCSNJWR7V	HDFSUtilization
j-13H7CCSNJWR7V	Idle
j-13H7CCSNJWR7V	JobsFailed
j-13H7CCSNJWR7V	JobsRunning
j-13H7CCSNJWR7V	LiveDataNodes
j-13H7CCSNJWR7V	LiveTaskTrackers
j-13H7CCSNJWR7V	MapSlotsOpen
i-13H7CCSNJWR7V	MissingBlocks

4. Click a metric to display the graph.

Select All | Clear



To access metrics from the CloudWatch CLI

- Call `mon-get-stats`. For more information, see the [Amazon CloudWatch Developer 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 [Using CloudWatch](#) in the *Amazon CloudWatch Developer Guide*.

Set alarms using the CloudWatch console

1. Sign in to the AWS Management Console and open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. Click the **Create Alarm** button. This launches the **Create Alarm Wizard**.
3. Click **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 click **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**. In the **Send notification to:** field, choose an existing SNS topic. If you select **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 you're about to create. Click **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 Developer Guide*.

To set an alarm using the CloudWatch API

- Call `mon-put-metric-alarm`. For more information, see [Amazon CloudWatch Developer 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 of the metrics that Amazon EMR reports in the console and pushes to CloudWatch.

Amazon EMR Metrics for Hadoop 1 AMIs

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.

Note

Amazon EMR pulls metrics from a cluster. If a cluster becomes unreachable, no metrics are reported until the cluster becomes available again.

Metric	Description
<i>Cluster Status</i>	

Metric	Description
Is Idle?	<p>Indicates that a cluster is no longer performing work, but is still alive and accruing charges. It is set to 1 if no tasks are running and no jobs are running, and set to 0 otherwise. This value is checked at five-minute intervals and a value of 1 indicates only that the cluster was idle when checked, not that it was idle for the entire five minutes. To avoid false positives, you should raise an alarm when this value has been 1 for more than one consecutive 5-minute check. For example, you might raise an alarm on this value if it has been 1 for thirty minutes or longer.</p> <p>Use case: Monitor cluster performance</p> <p>Units: <i>Boolean</i></p>
Jobs Running	<p>The number of jobs in the cluster that are currently running.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Count</i></p>
Jobs Failed	<p>The number of jobs in the cluster that have failed.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Count</i></p>
<i>Map/Reduce</i>	
Map Tasks Running	<p>The number of running map tasks for each job. If you have a scheduler installed and multiple jobs running, multiple graphs are generated.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>
Map Tasks Remaining	<p>The number of remaining map tasks for each job. If you have a scheduler installed and multiple jobs running, multiple graphs are generated. A remaining map task is one that is not in any of the following states: Running, Killed, or Completed.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>
Map Slots Open	<p>The unused map task capacity. This is calculated as the maximum number of map tasks for a given cluster, less the total number of map tasks currently running in that cluster.</p> <p>Use case: Analyze cluster performance</p> <p>Units: <i>Count</i></p>
Remaining Map Tasks Per Slot	<p>The ratio of the total map tasks remaining to the total map slots available in the cluster.</p> <p>Use case: Analyze cluster performance</p> <p>Units: <i>Ratio</i></p>

Metric	Description
Reduce Tasks Running	<p>The number of running reduce tasks for each job. If you have a scheduler installed and multiple jobs running, multiple graphs are generated.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>
Reduce Tasks Remaining	<p>The number of remaining reduce tasks for each job. If you have a scheduler installed and multiple jobs running, multiple graphs are generated.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>
Reduce Slots Open	<p>Unused reduce task capacity. This is calculated as the maximum reduce task capacity for a given cluster, less the number of reduce tasks currently running in that cluster.</p> <p>Use case: Analyze cluster performance</p> <p>Units: <i>Count</i></p>
<i>Node Status</i>	
Core Nodes Running	<p>The number of core nodes working. Data points for this metric are reported only when a corresponding instance group exists.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Count</i></p>
Core Nodes Pending	<p>The number of core nodes waiting to be assigned. All of the core nodes requested may not be immediately available; this metric reports the pending requests. Data points for this metric are reported only when a corresponding instance group exists.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Count</i></p>
Live Data Nodes	<p>The percentage of data nodes that are receiving work from Hadoop.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Percent</i></p>
Task Nodes Running	<p>The number of task nodes working. Data points for this metric are reported only when a corresponding instance group exists.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Count</i></p>

Amazon Elastic MapReduce Developer Guide
Monitor Metrics with CloudWatch

Metric	Description
Task Nodes Pending	<p>The number of core nodes waiting to be assigned. All of the task nodes requested may not be immediately available; this metric reports the pending requests. Data points for this metric are reported only when a corresponding instance group exists.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Count</i></p>
Live Task Trackers	<p>The percentage of task trackers that are functional.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Percent</i></p>
<i>IO</i>	
S3 Bytes Written	<p>The number of bytes written to Amazon S3.</p> <p>Use case: Analyze cluster performance, Monitor cluster progress</p> <p>Units: <i>Bytes</i></p>
S3 Bytes Read	<p>The number of bytes read from Amazon S3.</p> <p>Use case: Analyze cluster performance, Monitor cluster progress</p> <p>Units: <i>Bytes</i></p>
HDFS Utilization	<p>The percentage of HDFS storage currently used.</p> <p>Use case: Analyze cluster performance</p> <p>Units: <i>Percent</i></p>
HDFS Bytes Read	<p>The number of bytes read from HDFS.</p> <p>Use case: Analyze cluster performance, Monitor cluster progress</p> <p>Units: <i>Bytes</i></p>
HDFS Bytes Written	<p>The number of bytes written to HDFS.</p> <p>Use case: Analyze cluster performance, Monitor cluster progress</p> <p>Units: <i>Bytes</i></p>
Missing Blocks	<p>The number of blocks in which HDFS has no replicas. These might be corrupt blocks.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Count</i></p>
Total Load	<p>The total number of concurrent data transfers.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Count</i></p>
<i>HBase</i>	

Metric	Description
Backup Failed	<p>Whether the last backup failed. This is set to 0 by default and updated to 1 if the previous backup attempt failed. This metric is only reported for HBase clusters.</p> <p>Use case: Monitor HBase backups</p> <p>Units: <i>Count</i></p>
Most Recent Backup Duration	<p>The amount of time it took the previous backup to complete. This metric is set regardless of whether the last completed backup succeeded or failed. While the backup is ongoing, this metric returns the number of minutes after the backup started. This metric is only reported for HBase clusters.</p> <p>Use case: Monitor HBase Backups</p> <p>Units: <i>Minutes</i></p>
Time Since Last Successful Backup	<p>The number of elapsed minutes after the last successful HBase backup started on your cluster. This metric is only reported for HBase clusters.</p> <p>Use case: Monitor HBase backups</p> <p>Units: <i>Minutes</i></p>

Amazon EMR Metrics for Hadoop 2 AMIs

The following metrics are available for Hadoop 2 AMIs:

Metric	Description
<i>Cluster Status</i>	
Is Idle?	<p>Indicates that a cluster is no longer performing work, but is still alive and accruing charges. It is set to 1 if no tasks are running and no jobs are running, and set to 0 otherwise. This value is checked at five-minute intervals and a value of 1 indicates only that the cluster was idle when checked, not that it was idle for the entire five minutes. To avoid false positives, you should raise an alarm when this value has been 1 for more than one consecutive 5-minute check. For example, you might raise an alarm on this value if it has been 1 for thirty minutes or longer.</p> <p>Use case: Monitor cluster performance</p> <p>Units: <i>Boolean</i></p>
Container Allocated	<p>The number of resource containers allocated by the ResourceM-anager.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>

Metric	Description
Container Reserved	The number of containers reserved. Use case: Monitor cluster progress Units: <i>Count</i>
Container Pending	The number of containers in the queue that have not yet been allocated. Use case: Monitor cluster progress Units: <i>Count</i>
Apps Completed	The number of applications submitted to YARN that have completed. Use case: Monitor cluster progress Units: <i>Count</i>
Apps Failed	The number of applications submitted to YARN that have failed to complete. Use case: Monitor cluster progress, Monitor cluster health Units: <i>Count</i>
Apps Killed	The number of applications submitted to YARN that have been killed. Use case: Monitor cluster progress, Monitor cluster health Units: <i>Count</i>
Apps Pending	The number of applications submitted to YARN that are in a pending state. Use case: Monitor cluster progress Units: <i>Count</i>
Apps Running	The number of applications submitted to YARN that are running. Use case: Monitor cluster progress Units: <i>Count</i>
Apps Submitted	The number of applications submitted to YARN. Use case: Monitor cluster progress Units: <i>Count</i>
<i>Node Status</i>	

Amazon Elastic MapReduce Developer Guide
Monitor Metrics with CloudWatch

Metric	Description
Core Nodes Running	<p>The number of core nodes working. Data points for this metric are reported only when a corresponding instance group exists.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Count</i></p>
Core Nodes Pending	<p>The number of core nodes waiting to be assigned. All of the core nodes requested may not be immediately available; this metric reports the pending requests. Data points for this metric are reported only when a corresponding instance group exists.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Count</i></p>
Live Data Nodes	<p>The percentage of data nodes that are receiving work from Hadoop.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Percent</i></p>
MR Total Nodes	<p>The number of nodes presently available to MapReduce jobs.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>
MR Active Nodes	<p>The number of nodes presently running MapReduce tasks or jobs.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>
MR Lost Nodes	<p>The number of nodes allocated to MapReduce that have been marked in a LOST state.</p> <p>Use case: Monitor cluster health, Monitor cluster progress</p> <p>Units: <i>Count</i></p>
MR Unhealthy Nodes	<p>The number of nodes available to MapReduce jobs marked in an UNHEALTHY state.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>
MR Decommissioned Nodes	<p>The number of nodes allocated to MapReduce applications that have been marked in a DECOMMISSIONED state.</p> <p>Use case: Monitor cluster health, Monitor cluster progress</p> <p>Units: <i>Count</i></p>

Amazon Elastic MapReduce Developer Guide
Monitor Metrics with CloudWatch

Metric	Description
MR Rebooted Nodes	<p>The number of nodes available to MapReduce that have been rebooted and marked in a REBOOTED state.</p> <p>Use case: Monitor cluster health, Monitor cluster progress</p> <p>Units: <i>Count</i></p>
<i>IO</i>	
S3 Bytes Written	<p>The number of bytes written to Amazon S3.</p> <p>Use case: Analyze cluster performance, Monitor cluster progress</p> <p>Units: <i>Bytes</i></p>
S3 Bytes Read	<p>The number of bytes read from Amazon S3.</p> <p>Use case: Analyze cluster performance, Monitor cluster progress</p> <p>Units: <i>Bytes</i></p>
HDFS Utilization	<p>The percentage of HDFS storage currently used.</p> <p>Use case: Analyze cluster performance</p> <p>Units: <i>Percent</i></p>
HDFS Bytes Read	<p>The number of bytes read from HDFS.</p> <p>Use case: Analyze cluster performance, Monitor cluster progress</p> <p>Units: <i>Bytes</i></p>
HDFS Bytes Written	<p>The number of bytes written to HDFS.</p> <p>Use case: Analyze cluster performance, Monitor cluster progress</p> <p>Units: <i>Bytes</i></p>
Missing Blocks	<p>The number of blocks in which HDFS has no replicas. These might be corrupt blocks.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Count</i></p>
Total Load	<p>The total number of concurrent data transfers.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Count</i></p>
Memory Total MB	<p>The total amount of memory in the cluster.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Bytes</i></p>

Metric	Description
Memory Reserved MB	The amount of memory reserved. Use case: Monitor cluster progress Units: <i>Bytes</i>
Memory Available MB	The amount of memory available to be allocated. Use case: Monitor cluster progress Units: <i>Bytes</i>
Memory Allocated MB	The amount of memory allocated to the cluster. Use case: Monitor cluster progress Units: <i>Bytes</i>
Pending Deletion Blocks	The number of blocks marked for deletion. Use case: Monitor cluster progress, Monitor cluster health Units: <i>Count</i>
Under Replicated Blocks	The number of blocks that need to be replicated one or more times. Use case: Monitor cluster progress, Monitor cluster health Units: <i>Count</i>
Dfs Pending Replication Blocks	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: <i>Count</i>
Capacity Remaining GB	The amount of remaining HDFS disk capacity. Use case: Monitor cluster progress, Monitor cluster health Units: <i>Bytes</i>
<i>HBase</i>	
Backup Failed	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: <i>Count</i>

Metric	Description
Most Recent Backup Duration	<p>The amount of time it took the previous backup to complete. This metric is set regardless of whether the last completed backup succeeded or failed. While the backup is ongoing, this metric returns the number of minutes after the backup started. This metric is only reported for HBase clusters.</p> <p>Use case: Monitor HBase Backups</p> <p>Units: <i>Minutes</i></p>
Time Since Last Successful Backup	<p>The number of elapsed minutes after the last successful HBase backup started on your cluster. This metric is only reported for HBase clusters.</p> <p>Use case: Monitor HBase backups</p> <p>Units: <i>Minutes</i></p>

Dimensions for Amazon EMR Metrics

Amazon EMR data can be filtered using any of the dimensions in the following table.

Dimension	Description
ClusterId/JobFlowId	The identifier for a cluster. You can find this value by clicking on the cluster in the Amazon EMR console. It takes the form j-XXXXXXXXXXXXXX.
JobId	The identifier of a job within a cluster. You can use this to filter the metrics returned from a cluster down to those that apply to a single job within the cluster. JobId takes the form job_XXXXXXXXXXXXXX_XXXX.

Logging Amazon Elastic MapReduce API Calls in AWS CloudTrail

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

Amazon Elastic MapReduce 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 \(p. 432\)](#)
- [Understanding Amazon EMR Log File Entries \(p. 432\)](#)

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 Elastic MapReduce API Reference](#). For example, calls to the **ListClusters**, **DescribeCluster**, and **RunJobFlow** actions generate entries in CloudTrail log files.

Every log entry contains information about who generated the request. For example, if a request is made to create and run a new job flow (**RunJobFlow**), CloudTrail logs the user identity of the person or service that made the request. The user identity information helps you determine whether the request was made with root or IAM user credentials, with temporary security credentials for a role or federated user, or by another AWS service. For more information about CloudTrail fields, see [CloudTrail Event Reference](#) in the *AWS CloudTrail User Guide*.

You can store your log files in your bucket for as long as you want, but you can also define Amazon S3 lifecycle rules to archive or delete log files automatically. By default, your log files are encrypted by using Amazon S3 server-side encryption (SSE).

Understanding Amazon EMR Log File Entries

CloudTrail log files can contain one or more log entries composed of multiple JSON-formatted events. A log entry represents a single request from any source and includes information about the requested action, any input parameters, the date and time of the action, and so on. The log entries do not appear in any particular order. That is, they do not represent an ordered stack trace of the public API calls.

The following log file record shows that an IAM user called the **RunJobFlow** action by using the SDK.

```
{
  "Records": [
    {
      "eventVersion": "1.01",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:user/temporary-user-xx-7M",
        "accountId": "123456789012",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "userName": "temporary-user-xx-7M"
      },
      "eventTime": "2014-03-31T17:59:21Z",
      "eventSource": "elasticmapreduce.amazonaws.com",
      "eventName": "RunJobFlow",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "127.0.0.1",
      "userAgent": "aws-sdk-java/unknown-version Linux/xx Java_HotSpot(TM)_64-Bit_Server_VM/xx",
      "requestParameters": {
        "tags": [
          {
            "value": "prod",
            "key": "domain"
          },
          {
            "value": "us-east-1",
            "key": "realm"
          }
        ]
      }
    }
  ]
}
```

```
        {
            "value": "VERIFICATION",
            "key": "executionType"
        }
    ],
    "instances": {
        "slaveInstanceType": "m1.large",
        "ec2KeyName": "emr-integtest",
        "instanceCount": 1,
        "masterInstanceType": "m1.large",
        "keepJobFlowAliveWhenNoSteps": true,
        "terminationProtected": false
    },
    "visibleToAllUsers": false,
    "name": "Integ 1xm1large",
    "amiVersion": "3.0.4"
},
"responseElements": {
    "jobFlowId": "j-2WDJCGEG4E6AJ"
},
"requestID": "2f482daf-b8fe-11e3-89e7-75a3d0e071c5",
"eventID": "b348a38d-f744-4097-8b2a-e68c9b424698"
},
...additional entries
]
}
```

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 \(p. 433\)](#)
- [View Ganglia Metrics \(p. 434\)](#)
- [Ganglia Reports \(p. 436\)](#)
- [Hadoop Metrics in Ganglia \(p. 441\)](#)

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 click **Configure and add**.
4. Proceed with creating the cluster as described in [Plan an Amazon EMR Cluster \(p. 30\)](#).

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:

```
aws emr create-cluster --name "Test cluster" --ami-version 3.3 --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:

```
aws emr create-cluster --name "Test cluster" --ami-version 3.3 --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 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>.

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 [View Web Interfaces Hosted on Amazon EMR Clusters \(p. 446\)](#).

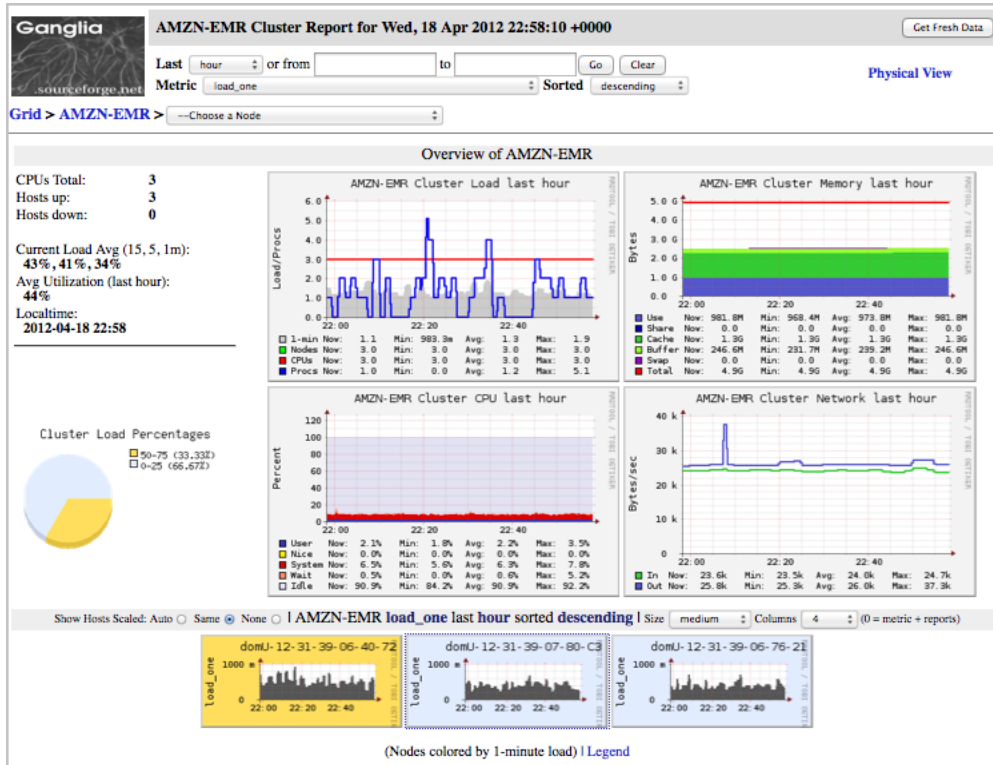
To view the Ganglia web interface

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 \(p. 451\)](#).
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 \(p. 453\)](#).
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 Amazon EMR cluster. For information about how to

locate the public DNS name of a master node, see [To retrieve the public DNS name of the master node using the Amazon EMR console \(p. 442\)](#).

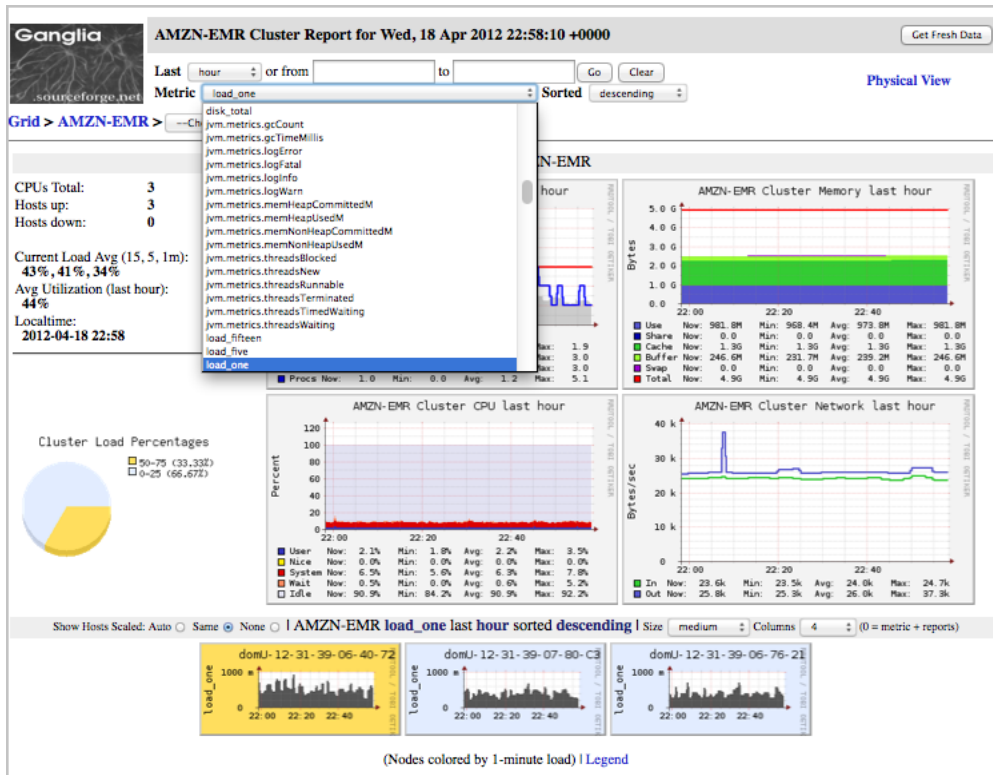
Ganglia Reports

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.



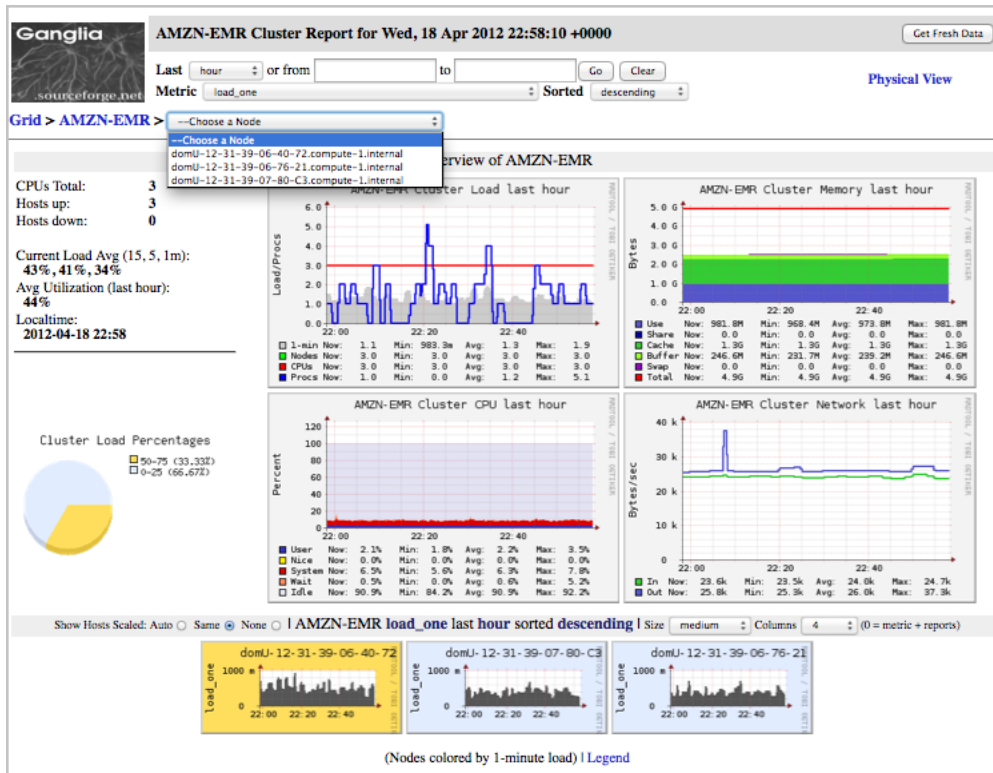
Amazon Elastic MapReduce Developer Guide Monitor Performance with Ganglia

The default graph for the node instances is Load, but you can use the **Metric** drop-down list to change the statistic displayed in the node-instance graphs.



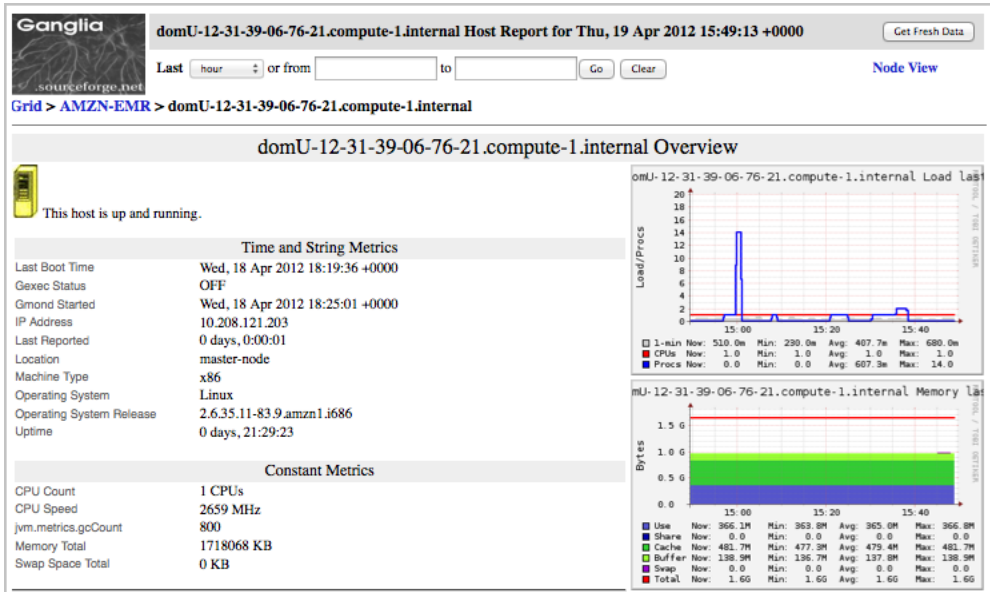
Amazon Elastic MapReduce Developer Guide Monitor Performance with Ganglia

You can drill down into the full set of statistics for a given instance by selecting the node from the drop-down list or by clicking the corresponding node-instance chart.



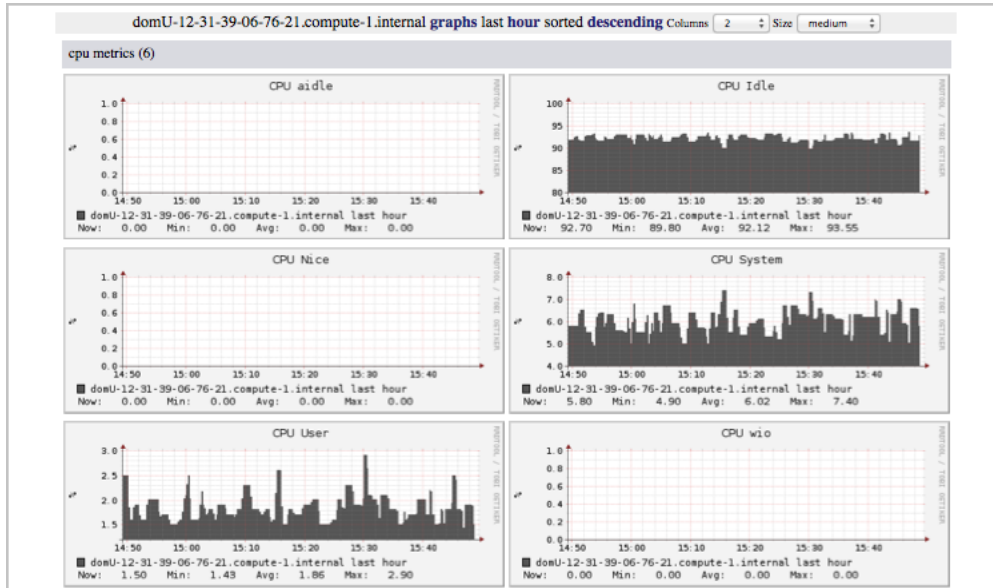
Amazon Elastic MapReduce Developer Guide Monitor Performance with Ganglia

This opens the Host Overview for the node.



Amazon Elastic MapReduce Developer Guide Monitor Performance with Ganglia

If you scroll down, you can view charts of the full range of statistics collected on the instance.



Hadoop Metrics in Ganglia

Ganglia reports Hadoop metrics for each node 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.*). You can view a complete list of these metrics by clicking the Gmetrics link, on the Host Overview page.

Connect to the Cluster

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

When you run an Amazon Elastic MapReduce (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 Amazon EMR cluster, the master node is an EC2 instance that coordinates the EC2 instances that are running as task and core nodes. The master node exposes a public DNS name that you can use to connect to it. By default, Amazon EMR creates security group rules for master and slave nodes that determine how you access the nodes. For example, the master node security group contains a rule that allows you to connect to the master node using an SSH client over TCP port 22.

Note

You can connect to the master node only while the cluster is running. When the cluster terminates, the EC2 instance acting as the master node is terminated and is no longer available. To connect to the master node, you must also specify an Amazon EC2 key pair private key when you launch the cluster. The key pair private key provides the credentials for the SSH connection to the master node. If you launch a cluster from the console, the Amazon EC2 key pair private key is specified in the **Security and Access** section on the **Create Cluster** page.

By default, the ElasticMapReduce-master security group permits inbound SSH access from CIDR range 0.0.0.0/0. This allows SSH connections over TCP port 22 from any IP address using the appropriate credentials. You can limit this rule by identifying a specific IP address or address range suitable for your environment. 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. 441)
- [View Web Interfaces Hosted on Amazon EMR Clusters](#) (p. 446)

Connect to the Master Node Using SSH

Secure Shell (SSH) is a network protocol you can use to create a secure connection to a remote computer. After you make a connection, the terminal on your local computer behaves as if it is running on the remote computer. Commands you issue locally run on the remote computer, and the command output from the remote computer appears in your terminal window.

When you use SSH with AWS, you are connecting to an EC2 instance, which is a virtual server running in the cloud. When working with Amazon EMR, the most common use of SSH is to connect to the EC2 instance that is acting as the master node of the cluster.

Using SSH to connect to the master node gives you the ability to monitor and interact with the cluster. You can issue Linux commands on the master node, run applications such as Hive and Pig interactively, browse directories, read log files, and so on. You can also create a tunnel in your SSH connection to view the web interfaces hosted on the master node. For more information, see [View Web Interfaces Hosted on Amazon EMR Clusters](#) (p. 446).

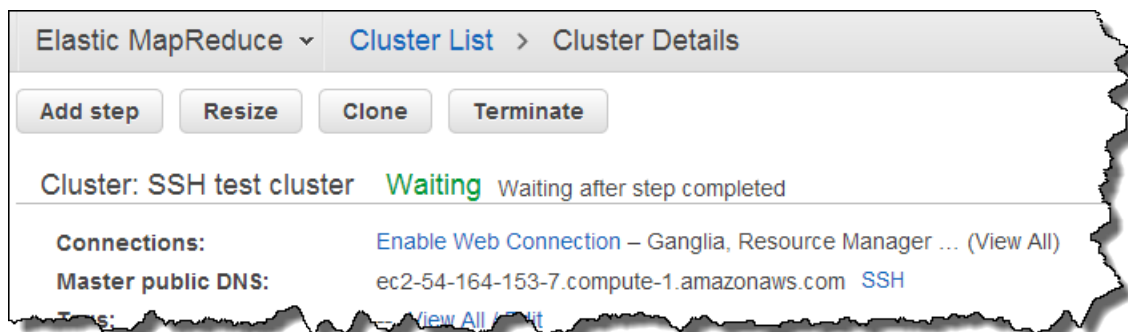
To connect to the master node using SSH, you need the public DNS name of the master node and your Amazon EC2 key pair private key. The Amazon EC2 key pair private key is specified when you launch the cluster. If you launch a cluster from the console, the Amazon EC2 key pair private key is specified in the **Security and Access** section on the **Create Cluster** page. For more information about accessing your key pair, see [Amazon EC2 Key Pairs](#) in the *Amazon EC2 User Guide for Linux Instances*.

Retrieve the Public DNS Name of the Master Node

You can retrieve the master public DNS name using the Amazon EMR console and the AWS CLI.

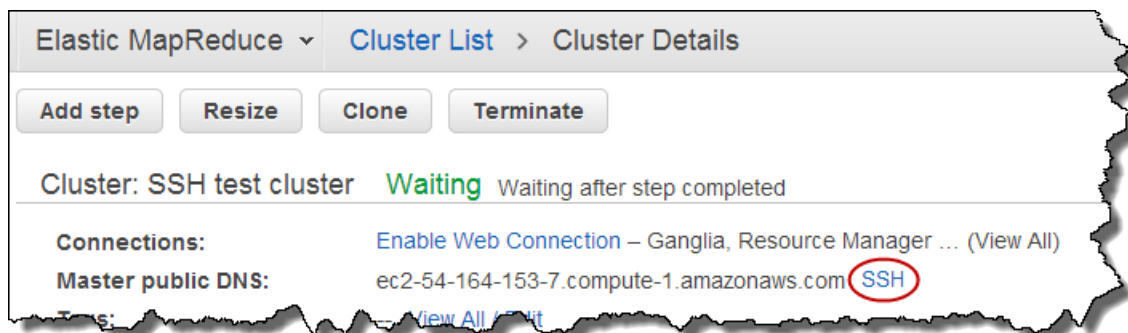
To retrieve the public DNS name of the master node using the Amazon EMR console

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. On the **Cluster List** page, click the link for your cluster.
3. Note the **Master public DNS** value that appears at the top of the **Cluster Details** page.



Note

You may also click the **SSH** link beside the master public DNS name for instructions on creating an SSH connection with the master node.



To retrieve the public DNS name of the master node using the AWS CLI

1. To retrieve the cluster identifier, type the following command.

```
aws emr list-clusters
```

The output lists your clusters including the cluster IDs. Note the cluster ID for the cluster to which you are connecting.

```
"Status": {
  "Timeline": {
    "ReadyDateTime": 1408040782.374,
    "CreationDateTime": 1408040501.213
  },
  "State": "WAITING",
  "StateChangeReason": {
    "Message": "Waiting after step completed"
  }
},
"NormalizedInstanceHours": 4,
"Id": "j-2AL4XXXXXX5T9",
"Name": "My cluster"
```

2. To list the cluster instances including the master public DNS name for the cluster, type one of the following commands. Replace `j-2AL4XXXXXX5T9` with the cluster ID returned by the previous command.

```
aws emr list-instances --cluster-id j-2AL4XXXXXX5T9
```

Or:

```
aws emr describe-clusters --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-12XXXXXXXXXXFMH",
"PrivateIpAddress": "###.##.##.###"
```

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

Connect to the Master Node Using SSH on Linux, Unix, and Mac OS X

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

The following instructions demonstrate opening an SSH connection to the Amazon EMR master node on Linux, Unix, and Mac OS X.

To configure the key pair private key file permissions

Before you can use your Amazon EC2 key pair private key to create an SSH connection, you must set permissions on the `.pem` file so that only the key owner has permission to access the file. This is required for creating an SSH connection using terminal or the AWS CLI.

1. Locate your `.pem` file. These instructions assume that the file is named `mykeypair.pem` and that it is stored in the current user's home directory.
2. Type the following command to set the permissions. Replace `~/mykeypair.pem` with the location and file name of your key pair private key file.

```
chmod 400 ~/mykeypair.pem
```

If you do not set permissions on the `.pem` file, you will receive an error indicating that your key file is unprotected and the key will be rejected. To connect, you only need to set permissions on the key pair private key file the first time you use it.

To connect to the master node using terminal

1. Open a terminal window. On Mac OS X, choose **Applications > Utilities > Terminal**. On other Linux distributions, terminal is typically found at **Applications > Accessories > Terminal**.
2. To establish a connection to the master node, type the following command. Replace `ec2-###-##-##-###.compute-1.amazonaws.com` with the master public DNS name of your cluster and replace `~/mykeypair.pem` with the location and file name of your `.pem` file.

```
ssh hadoop@ec2-###-##-##-###.compute-1.amazonaws.com -i ~/mykeypair.pem
```

Important

You must use the login name `hadoop` when you connect to the Amazon EMR master node, otherwise you may see an error similar to `Server refused our key`.

3. A warning states that the authenticity of the host you are connecting to cannot be verified. Type `yes` to continue.
4. When you are done working on the master node, type the following command to close the SSH connection.

```
exit
```

Connect to the Master Node Using the AWS CLI

You can create an SSH connection with the master node using the AWS CLI on Windows and on Linux, Unix, and Mac OS X. Regardless of the platform, you need the public DNS name of the master node and your Amazon EC2 key pair private key. If you are using the AWS CLI on Linux, Unix, or Mac OS X, you must also set permissions on the private key (`.pem` or `.ppk`) file as shown in [To configure the key pair private key file permissions](#) (p. 444).

To connect to the master node using the AWS CLI

1. To retrieve the cluster identifier, type:

```
aws emr list-clusters
```

The output lists your clusters including the cluster IDs. Note the cluster ID for the cluster to which you are connecting.

```
"Status": {
  "Timeline": {
    "ReadyDateTime": 1408040782.374,
    "CreationDateTime": 1408040501.213
  },
  "State": "WAITING",
  "StateChangeReason": {
    "Message": "Waiting after step completed"
  }
},
"NormalizedInstanceHours": 4,
"Id": "j-2AL4XXXXXX5T9",
"Name": "AWS CLI cluster"
```

2. Type the following command to open an SSH connection to the master node. In the following example, replace `j-2AL4XXXXXX5T9` with the cluster ID and replace `~/mykeypair.key` with the location and file name of your `.pem` file (for Linux, Unix, and Mac OS X) or `.ppk` file (for Windows).

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

3. When you are done working on the master node, close the AWS CLI window.

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

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. Double-click `putty.exe` to start PuTTY. You can also launch PuTTY from the Windows programs list.
2. If necessary, in the **Category** list, click **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 click **Auth**.
5. For **Private key file for authentication**, click **Browse** and select the `.ppk` file that you generated.
6. Click **Open**.
7. Click **Yes** to dismiss the PuTTY security alert.

Important

When logging into the master node, type `hadoop` if you are prompted for a user name .

8. When you are done working on the master node, you can close the SSH connection by closing PuTTY.

Note

To prevent the SSH connection from timing out, you can click **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 right-clicking the PuTTY title bar and choosing **Change Settings**.

View Web Interfaces Hosted on Amazon EMR Clusters

Hadoop and other applications you install on your Amazon EMR cluster, publish user interfaces as web sites hosted on the master node. For security reasons, these web sites are only available on the master node's local web server and are not publicly available over the Internet. Hadoop also publishes user interfaces as web sites hosted on the core and task (slave) nodes. These web sites are also only available on local web servers on the nodes.

The following table lists web interfaces 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. 442). 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. 451).

Name of Interface	URI
<i>Hadoop version 2.x</i>	
Hadoop ResourceManager	<code>http://master-public-dns-name:9026/http://master-public-dns-name:8088/</code>
Hadoop HDFS NameNode	<code>http://master-public-dns-name:9101/</code>
Ganglia Metrics Reports	<code>http://master-public-dns-name/ganglia/</code>
HBase Interface	<code>http://master-public-dns-name:60010/master-status</code>

Amazon Elastic MapReduce Developer Guide
View Web Interfaces Hosted on Amazon EMR Clusters

Name of Interface	URI
Hue Web Application	http:// <i>master-public-dns-name</i> :8888/
Impala Statestore	http:// <i>master-public-dns-name</i> :25000
Impalad	http:// <i>master-public-dns-name</i> :25010
Impala Catalog	http:// <i>master-public-dns-name</i> :25020
<i>Hadoop version 1.x</i>	
Hadoop MapReduce Job-Tracker	http:// <i>master-public-dns-name</i> :9100/
Hadoop HDFS NameNode	http:// <i>master-public-dns-name</i> :9101/
Ganglia Metrics Reports	http:// <i>master-public-dns-name</i> /ganglia/
HBase Interface	http:// <i>master-public-dns-name</i> :60010/master-status

For more information, see the following:

- Ganglia web interface: [Monitor Performance with Ganglia](#) (p. 433)
- Impala web interfaces: [Accessing Impala Web User Interfaces](#) (p. 293)
- Hue: [Configure Hue to View, Query, or Manipulate Data](#) (p. 334)

The following table lists web interfaces you can view on the core and task nodes. These Hadoop interfaces are available on all clusters. To access the following interfaces, replace *slave-public-dns-name* in the URI with the public DNS name of the node. For more information about retrieving the public DNS name of a core or task node instance, see [Connecting to Your Linux/Unix Instances Using SSH](#) in the *Amazon EC2 User Guide for Linux Instances*. In addition to retrieving the public DNS name of the core or task node, you must also edit the ElasticMapReduce-slave security group to allow SSH access over TCP port 22. For more information about modifying security group rules, see [Adding Rules to a Security Group](#) in the *Amazon EC2 User Guide for Linux Instances*.

Name of Interface	URI
<i>Hadoop version 2.x</i>	
Hadoop NodeManager	http:// <i>slave-public-dns-name</i> :9035/
Hadoop HDFS DataNode	http:// <i>slave-public-dns-name</i> :9102/
<i>Hadoop version 1.x</i>	
Hadoop HDFS DataNode (core nodes only)	http:// <i>slave-public-dns-name</i> :9102/
Hadoop MapReduce TaskTracker	http:// <i>slave-public-dns-name</i> :9103/

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 \(p. 453\)](#).

Topics

- [Option 1: Set Up an SSH Tunnel to the Master Node Using Local Port Forwarding \(p. 448\)](#)
- [Option 2, Part 1: Set Up an SSH Tunnel to the Master Node Using Dynamic Port Forwarding \(p. 451\)](#)
- [Option 2, Part 2: Configure Proxy Settings to View Websites Hosted on the Master Node \(p. 453\)](#)
- [Access the Web Interfaces on the Master Node Using the Console \(p. 457\)](#)

Option 1: Set Up an SSH Tunnel to the Master Node Using Local Port Forwarding

To connect to the local web server on the master node, you create a 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 (localhost: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 \(p. 442\)](#). 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 \(p. 446\)](#).

Set Up an SSH Tunnel to the Master Node Using Local Port Forwarding on Linux, Unix, and Mac OS X

To set up an SSH tunnel using local port forwarding in terminal

1. Open a terminal window. On Mac OS X, choose **Applications > Utilities > Terminal**. On other Linux distributions, terminal is typically found at **Applications > Accessories > Terminal**.
2. Type the following command to open an SSH tunnel on your local machine. This command accesses the ResourceManager web interface by forwarding traffic on local port 8157 (a randomly chosen, unused local port) to port 80 on the master node's local web server. In the command, replace `~/mykeypair.pem` with the location and file name of your `.pem` file and replace `ec2-###-##-##-###.compute-1.amazonaws.com` with the master public DNS name of your cluster.

```
ssh -i ~/mykeypair.pem -N -L 8157:ec2-###-##-##-###.compute-1.amazonaws.com: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.

3. To open the ResourceManager web interface in your browser, type: `http://localhost:8157/` in the address bar.
4. To access the NameNode web interface, launch a new terminal session, replace port 80 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 80 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, click **Session**.
3. In the **Host Name (or IP address)** field, type `hadoop@MasterPublicDNS`. For example: `hadoop@ec2-###-##-##-###.compute-1.amazonaws.com`.

- In the **Category** list, expand **Connection > SSH**, and then click **Auth**.
- For **Private key file for authentication**, click **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.

- In the **Category** list, expand **Connection > SSH**, and then click **Tunnels**.
- In the **Source port** field, type an unused local *port number*, for example `8157`.
- To access a web interface, in the **Destination** field, type *host name:port number*. For example, to access the Ganglia interface, type `localhost:80`.
- Leave the **Local** and **Auto** options selected.
- Click **Add**. You should see an entry in the **Forwarded ports** box similar to: `L8157 localhost:80`.
- Click **Open** and then click **Yes** to dismiss the PuTTY security alert.

Important

When logging into the master node, if you are prompted for a user name, type `hadoop`.

- To access the Ganglia interface on the master node, type `http://localhost:8157/ganglia` in your browser's address bar.
- 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**.
- Follow the previous steps to add additional tunnels for the remaining web interfaces using the following table as a guide.

Interface	Source port	Destination
Hadoop ResourceManager	8158	<i>master-public-dns-name:9026</i>
Hadoop NameNode	8159	<i>master-public-dns-name:9101</i>
Hue web application	8160	<i>master-public-dns-name:8888</i>

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 \(p. 446\)](#).

- After adding a new tunnel, click **Apply**.
- 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.

- 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

To connect to the local web server on the master node, you create a 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 \(p. 442\)](#). 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 \(p. 446\)](#).

Set Up an SSH Tunnel to the Master Node Using Dynamic Port Forwarding on Linux, Unix, and Mac OS X

To set up an SSH tunnel using dynamic port forwarding on Linux, Unix, and Mac OS X

1. Open a terminal window. On Mac OS X, choose **Applications > Utilities > Terminal**. On other Linux distributions, terminal is typically found at **Applications > Accessories > Terminal**.
2. Type the following command to open an SSH tunnel on your local machine. Replace `~/mykeypair.pem` with the location and file name of your `.pem` file, replace `8157` with an unused, local port number, and replace `c2-###-##-##-###.compute-1.amazonaws.com` with the master public DNS name of your cluster.

```
ssh -i ~/mykeypair.pem -N -D 8157 hadoop@ec2-###-##-##-###.compute-1.amazonaws.com
```

After you issue this command, the terminal remains open and does not return a response.

Note

`-D` signifies the use of dynamic port forwarding which allows you to specify a local port used to forward data to all remote ports on the master node's local web server. Dynamic port forwarding creates a local SOCKS proxy listening on the port specified in the command.

3. After the tunnel is active, configure a SOCKS proxy for your browser. For more information, see [Option 2, Part 2: Configure Proxy Settings to View Websites Hosted on the Master Node \(p. 453\)](#).
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 \(p. 444\)](#). 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 \(p. 445\)](#).
2. To retrieve the cluster identifier, type:

```
aws emr list-clusters
```

The output lists your clusters including the cluster IDs. Note the cluster ID for the cluster to which you are connecting.

```
"Status": {
  "Timeline": {
    "ReadyDateTime": 1408040782.374,
    "CreationDateTime": 1408040501.213
  },
  "State": "WAITING",
  "StateChangeReason": {
    "Message": "Waiting after step completed"
  }
},
"NormalizedInstanceHours": 4,
"Id": "j-2AL4XXXXXX5T9",
"Name": "AWS CLI cluster"
```

3. Type the following command to open an SSH tunnel to the master node using dynamic port forwarding. In the following example, replace `j-2AL4XXXXXX5T9` with the cluster ID and replace `~/mykeypair.key` with the location and file name of your `.pem` file (for Linux, Unix, and Mac OS X) or `.ppk` file (for Windows).

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

Note

The `socks` command automatically configures dynamic port forwarding on local port 8157. Currently, this setting cannot be modified.

4. After the tunnel is active, configure a SOCKS proxy for your browser. For more information, see [Option 2, Part 2: Configure Proxy Settings to View Websites Hosted on the Master Node \(p. 453\)](#).
5. When you are done working with the web interfaces on the master node, close the AWS CLI window.

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

Set Up an SSH Tunnel to the Master Node Using Dynamic Port Forwarding on Windows

Windows users can use an SSH client such as PuTTY to create an SSH tunnel to the master node. Before connecting to the Amazon EMR master node, you should download and install PuTTY and PuTTYgen. You can download these tools from the [PuTTY download page](#).

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

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

To set up an SSH tunnel using dynamic port forwarding on Windows

1. Double-click `putty.exe` to start PuTTY. You can also launch PuTTY from the Windows programs list.

Note

If you already have an active SSH session with the master node, you can add a tunnel by right-clicking the PuTTY title bar and choosing **Change Settings**.

2. If necessary, in the **Category** list, click **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 click **Auth**.
5. For **Private key file for authentication**, click **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 click **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. Click **Add** and then click **Open**.
11. Click **Yes** to dismiss the PuTTY security alert.

Important

When you log into 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 \(p. 453\)](#).
13. When you are done working with the web interfaces on the master node, close the PuTTY window.

Option 2, Part 2: Configure Proxy Settings to View Websites Hosted on the Master Node

If you use an SSH tunnel with dynamic port forwarding, you must use a SOCKS proxy management add-on to control the proxy settings in your browser. Using a SOCKS proxy management tool allows you to automatically filter URLs based on text patterns and to limit the proxy settings to domains that match the form of the master node's public DNS name. The browser add-on automatically handles turning the proxy on and off when you switch between viewing websites hosted on the master node, and those on the Internet. To manage your proxy settings, configure your browser to use an add-on such as FoxyProxy or SwitchySharp.

For more information about creating an SSH tunnel, see [Option 2, Part 1: Set Up an SSH Tunnel to the Master Node Using Dynamic Port Forwarding \(p. 451\)](#). For more information about the available web interfaces, see [View Web Interfaces Hosted on Amazon EMR Clusters \(p. 446\)](#).

Configure FoxyProxy for Firefox

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. Before configuring FoxyProxy, you must first create an SSH tunnel using dynamic port forwarding. For more information, see [Option 2, Part 1: Set Up an SSH Tunnel to the Master Node Using Dynamic Port Forwarding](#) (p. 451).

Note

The following tutorial uses FoxyProxy Standard version 4.2.4 and Firefox version 24.7.0.

To install and configure FoxyProxy in Firefox

1. Download and install the Standard version of FoxyProxy from <http://getfoxyproxy.org/downloads.html>.
2. Using a text editor, create a file named `foxyproxy-settings.xml` containing the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<foxyproxy>
  <proxies>
    <proxy name="emr-socks-proxy" id="2322596116" notes="" fromSubscription="false" enabled="true" mode="manual" selectedTabIndex="2" lastresort="false" animatedIcons="true" includeInCycle="true" color="#0055E5" proxyDNS="true" noInternalIPs="false" autoconfMode="pac" clearCacheBeforeUse="false" disableCache="false" clearCookiesBeforeUse="false" rejectCookies="false">
      <matches>
        <match enabled="true" name="*ec2*.amazonaws.com*" pattern="*ec2*.amazonaws.com*" isRegex="false" isBlackList="false" isMultiLine="false" caseSensitive="false" fromSubscription="false" />
        <match enabled="true" name="*ec2*.compute*" pattern="*ec2*.compute*" isRegex="false" isBlackList="false" isMultiLine="false" caseSensitive="false" fromSubscription="false" />
        <match enabled="true" name="10.*" pattern="http://10.*" isRegex="false" isBlackList="false" isMultiLine="false" caseSensitive="false" fromSubscription="false" />
        <match enabled="true" name="*10*.amazonaws.com*" pattern="*10*.amazonaws.com*" isRegex="false" isBlackList="false" isMultiLine="false" caseSensitive="false" fromSubscription="false" />
        <match enabled="true" name="*10*.compute*" pattern="*10*.compute*" isRegex="false" isBlackList="false" isMultiLine="false" caseSensitive="false" fromSubscription="false" />
      </matches>
      <manualconf host="localhost" port="8157" socksVersion="5" isSocks="true" username="" password="" domain="" />
    </proxy>
  </proxies>
</foxyproxy>
```

This file includes the following settings:

- Port 8157 is the local port number used to establish the SSH tunnel with the master node. This must match the port number you used in PuTTY or terminal.
- 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 1.x. Alter this filter if it conflicts with your network access plan.

3. Click **Firefox > Add-ons**.
4. On the **Add-ons** tab, to the right of **FoxyProxy Standard**, click **Options**.
5. In the **FoxyProxy Standard** dialog, click **File > Import Settings**.
6. Browse to the location of `foxyproxy-settings.xml`, select the file, and click **Open**.
7. Click **Yes** when prompted to overwrite the existing settings and then click **Yes** to restart Firefox.
8. When Firefox restarts, on the **Add-ons** tab, to the right of **FoxyProxy Standard**, click **Options**.
9. In the **FoxyProxy Standard** dialog, for **Select Mode**, choose **Use proxies based on their pre-defined patterns and priorities**.
10. Click **Close**.
11. 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.

Interface	URL
Ganglia Metrics Reports	<code>master-public-dns/ganglia/</code>
Hadoop ResourceManager	<code>master-public-dns-name:9026</code>
Hadoop NameNode	<code>master-public-dns-name:9101</code>
Hue web application	<code>master-public-dns-name:8888</code>

For a complete list of web interfaces on the master node, see [View Web Interfaces Hosted on Amazon EMR Clusters \(p. 446\)](#).

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. Before configuring FoxyProxy, you must first create an SSH tunnel using dynamic port forwarding. For more information, see [Option 2, Part 1: Set Up an SSH Tunnel to the Master Node Using Dynamic Port Forwarding \(p. 451\)](#).

Note

The following tutorial uses FoxyProxy Standard version 3.0.3 and Chrome version 24.7.0.

To install and configure FoxyProxy in Google Chrome

1. Download and install the Standard version of FoxyProxy from <http://getfoxyproxy.org/downloads.html>.
2. When prompted, click **FREE** to install the FoxyProxy extension and then click **Add**.
3. Using a text editor, create a file named `foxyproxy-settings.xml` containing the following:

```
<?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*" pat
```



```
tern="*ec2*.amazonaws.com" isRegex="false" isBlackList="false" isMultiLine="false" caseSensitive="false" fromSubscription="false" />
    <match enabled="true" name="*ec2*.compute*" pattern="*ec2*.compute*" isRegex="false" isBlackList="false" isMultiLine="false" caseSensitive="false" fromSubscription="false" />
    <match enabled="true" name="10.*" pattern="http://10.*" isRegex="false" isBlackList="false" isMultiLine="false" caseSensitive="false" fromSubscription="false" />
    <match enabled="true" name="*10*.amazonaws.com*" pattern="*10*.amazonaws.com*" isRegex="false" isBlackList="false" isMultiLine="false" caseSensitive="false" fromSubscription="false" />
    <match enabled="true" name="*10*.compute*" pattern="*10*.compute*" isRegex="false" isBlackList="false" isMultiLine="false" caseSensitive="false" fromSubscription="false" />
</matches>
<manualconf host="localhost" port="8157" socksVersion="5" isSocks="true" username="" password="" domain="" />
</proxy>
</proxies>
</foxyproxy>
```

This file includes the following settings:

- Port 8157 is the local port number used to establish the SSH tunnel with the master node. This must match the port number you used in PuTTY or terminal.
- 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 1.x. Alter this filter if it conflicts with your network access plan.

4. Click **Customize and Control Google Chrome > Tools > Extensions**.
5. On the **Extensions** tab, below **FoxyProxy Standard**, click **Options**.
6. On the **FoxyProxy Standard** page, click **Import/Export**.
7. On the **Import/Export** page, click **Choose File**, browse to the location of `foxyproxy-settings.xml`, select the file, and click **Open**.
8. Click **Replace** when prompted to overwrite the existing settings.
9. At the top of the page, for **Proxy mode**, choose **Use proxies based on their pre-defined patterns and priorities**.
10. 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.

Interface	URL
Ganglia Metrics Reports	<code>master-public-dns/ganglia/</code>
Hadoop ResourceManager	<code>master-public-dns-name:9026</code>
Hadoop NameNode	<code>master-public-dns-name:9101</code>
Hue web application	<code>master-public-dns-name::8888</code>

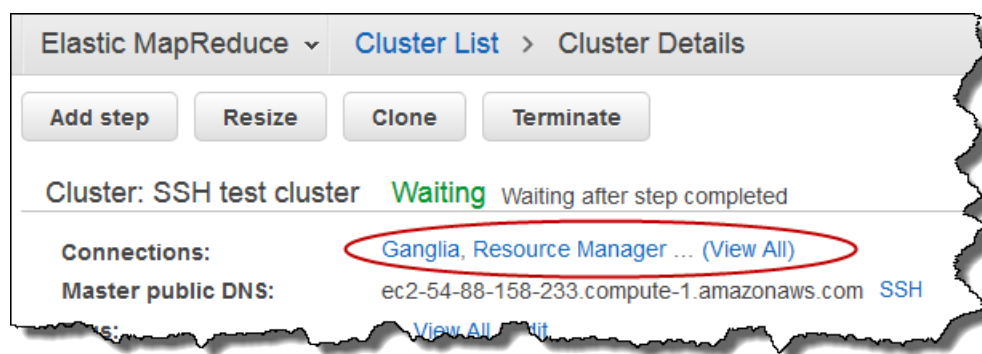
For a complete list of web interfaces on the master node, see [View Web Interfaces Hosted on Amazon EMR Clusters](#) (p. 446).

Access the Web Interfaces on the Master Node Using the Console

If you already have an SSH tunnel configured with the Amazon EMR master node using dynamic port forwarding, you can open the web interfaces using the console.

To open the web interfaces using the console

1. Verify that you have established an SSH tunnel with the master node and that you have a proxy management add-on configured for your browser.
2. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
3. On the **Cluster List** page, click the link for your cluster.
4. In the cluster details, for **Connections**, click the link for the web interface you wish to open in your browser.



5. Alternatively, click the **View All** link to display links to all of the available web interfaces on your cluster's master node. Clicking the links opens the interfaces in your browser.

Web Interfaces Hosted on this Cluster

Hadoop, Ganglia, and other applications publish user interfaces as websites hosted on the master node. For security reasons, these interfaces are only available on the master node's local webserver (`http://localhost:port`) and are not published on the Internet.

Note

For the below links to work properly an SSH tunnel must be open and your browser configured to use the proxy for Amazon EMR.

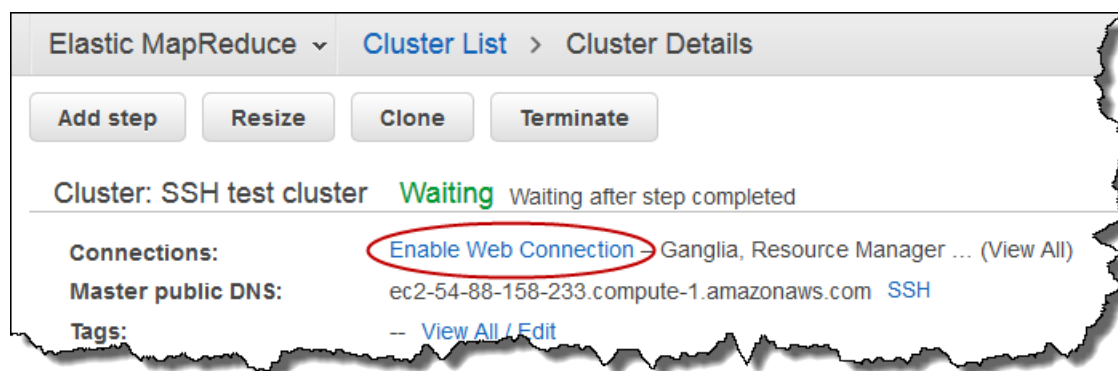
The following table lists web interfaces you can view on the master node:

Interface	URI
Resource Manager	http://ec2-54-164-54-29.compute-1.amazonaws.com:9026/
HDFS Name Node	http://ec2-54-164-54-29.compute-1.amazonaws.com:9101/

The following table lists web interfaces you can view on the slave nodes:

Interface	URI
Node Manager	http://ec2-000-000-000-000.compute-1.amazonaws.com:9035/
HDFS Data Node	http://ec2-000-000-000-000.compute-1.amazonaws.com:9102/

If you do not have an SSH tunnel open with the master node, click **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. 451).



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 \(p. 459\)](#)
- [Managing Cluster Termination \(p. 462\)](#)

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.

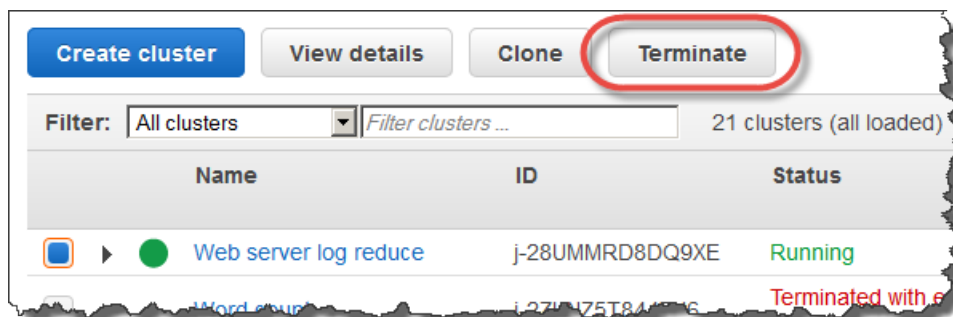
Depending on the configuration of the cluster, it may take up to 5-20 minutes for the cluster to completely terminate and release allocated resources, such as EC2 instances.

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. Note that you can select multiple clusters and terminate them at the same time.
3. Click **Terminate**.

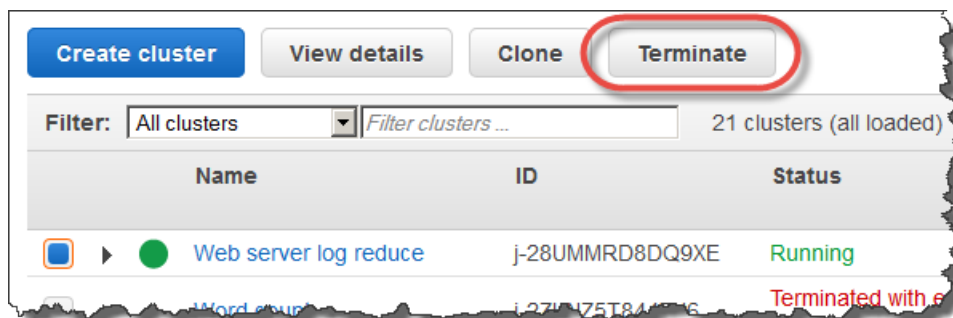


4. When prompted, click **Terminate**.

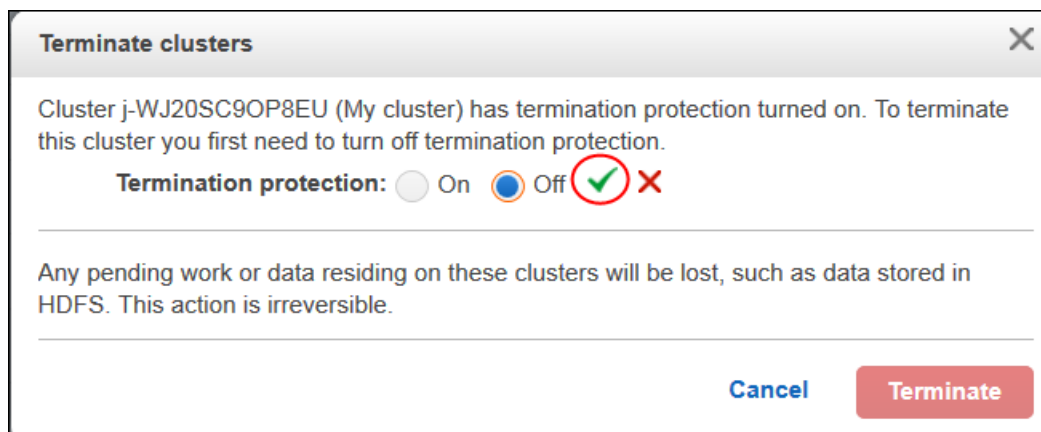
Amazon EMR terminates the instances in the cluster and stops saving log data.

To terminate a cluster with termination protection on

1. Sign in to the AWS Management Console and open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. On the **Cluster List** page, select the cluster to terminate. Note that you can select multiple clusters and terminate them at the same time.
3. Click **Terminate**.



4. When prompted, click **Change** to turn termination protection off. Note, if you selected multiple clusters, click **Turn off all** to disable termination protection for all the clusters at once.
5. In the **Terminate clusters** dialog, for **Termination Protection**, click **Off** and then click the check mark to confirm.



6. Click **Terminate**.

Amazon EMR terminates the instances in the cluster and stops saving log data.

Terminate a Cluster Using the AWS CLI

To terminate an unprotected cluster using the AWS CLI

To terminate an unprotected cluster using the AWS CLI, use the `terminate-clusters` subcommand with the `--cluster-ids` parameter.

- Type the following command to terminate a single cluster and replace `j-3KVXXXXXXXX7UG` with your cluster ID.

```
aws emr terminate-clusters --cluster-ids j-3KVXXXXXXXX7UG
```

To terminate multiple clusters, type the following command and replace `j-3KVXXXXXXXX7UG` and `j-WJ2XXXXXXXX8EU` with your cluster IDs.

```
aws emr terminate-clusters --cluster-ids j-3KVXXXXXXXX7UG j-WJ2XXXXXXXX8EU
```

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

To terminate a protected cluster using the AWS CLI

To terminate a protected cluster using the AWS CLI, first disable termination protection using the `modify-cluster-attributes` subcommand with the `--no-termination-protected` parameter. Then use the `terminate-clusters` subcommand with the `--cluster-ids` parameter to terminate it.

1. Type the following command to disable termination protection and replace `j-3KVTXXXXXXXX7UG` with your cluster ID.

```
aws emr modify-cluster-attributes --cluster-id j-3KVTXXXXXXXX7UG --no-termination-protected
```

2. To terminate the cluster, type the following command and replace `j-3KVXXXXXXXX7UG` with your cluster ID.

```
aws emr terminate-clusters --cluster-ids j-3KVXXXXXXXX7UG
```

To terminate multiple clusters, type the following command and replace `j-3KVXXXXXXXX7UG` and `j-WJ2XXXXXXXX8EU` with your cluster IDs.

```
aws emr terminate-clusters --cluster-ids j-3KVXXXXXXXX7UG j-WJ2XXXXXXXX8EU
```

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

Terminate a Cluster Using the API

The `TerminateJobFlows` operation ends step processing, uploads any log data from Amazon EC2 to Amazon S3 (if configured), and terminates the Hadoop cluster. A cluster also terminates automatically if you set `KeepJobAliveWhenNoSteps` to `False` in a `RunJobFlows` request.

You can use this action to terminate either a single cluster or a list of clusters by their cluster IDs.

For more information about the input parameters unique to `TerminateJobFlows`, see [TerminateJobFlows](#). For more information about the generic parameters in the request, see [Common Request Parameters](#).

Managing Cluster Termination

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

Termination protection ensures that the EC2 instances in your job flow are not shut down by an accident or error. This protection is especially useful if your cluster contains data in instance storage that you need to recover before those instances are terminated.

When termination protection is not enabled, you can terminate clusters either through calls to the `TerminateJobFlows` API, through the Amazon EMR console, or by using the command line interface. In addition, the master node may terminate a task node that has become unresponsive or has returned an error.

By default, termination protection is enabled when you launch a cluster using the console. Termination protection is disabled by default when you launch a cluster using the CLI or API. When termination protection is enabled, you must explicitly remove termination protection from the cluster before you can terminate it. With termination protection enabled, `TerminateJobFlows` cannot terminate the cluster and users cannot terminate the cluster using the CLI. Users terminating the cluster using the Amazon EMR console receive an extra confirmation box asking if they want to remove termination protection before terminating the cluster.

If you attempt to terminate a protected cluster with the API or CLI, the API returns an error, and the CLI exits with a non-zero return code.

When you submit steps to a cluster, the `ActionOnFailure` setting determines what the cluster does in response to any errors. The possible values for this setting are:

- `TERMINATE_JOB_FLOW`: If the step fails, terminate the cluster. If the cluster has termination protection enabled AND auto-terminate disabled, it will not terminate.

- `CANCEL_AND_WAIT`: If the step fails, cancel the remaining steps. If the cluster has auto-terminate disabled, the cluster will not terminate.
- `CONTINUE`: If the step fails, continue to the next step.

Termination Protection in Amazon EMR and Amazon EC2

Termination protection of clusters in Amazon EMR is analogous to setting the `disableAPITermination` flag on an EC2 instance. In the event of a conflict between the termination protection set in Amazon EC2 and that set in Amazon EMR, the Amazon EMR cluster protection status overrides that set by Amazon EC2 on the given instance. For example, if you use the Amazon EC2 console to *enable* termination protection on an EC2 instance in an Amazon EMR cluster that has termination protection *disabled*, Amazon EMR turns off termination protection on that EC2 instance and shuts down the instance when the rest of the cluster terminates.

Termination Protection and Spot Instances

Amazon EMR termination protection does not prevent an Amazon EC2 Spot Instance from terminating when the Spot Price rises above the maximum bid price.

Termination Protection and Auto-terminate

Enabling auto-terminate creates a transient cluster. The cluster automatically terminates when the last step is successfully completed even if termination protection is enabled.

Disabling auto-terminate causes instances in a cluster to persist after steps have successfully completed, but still allows the cluster to be terminated by user action, by errors, and by calls to `TerminateJobFlows` (if termination protection is disabled).

Note

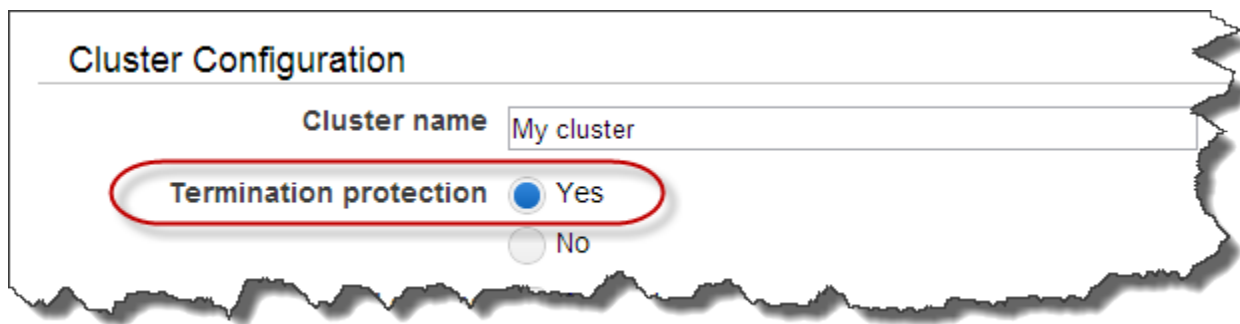
By default, auto-terminate is disabled for clusters launched using the console and the CLI. Clusters launched using the API have auto-terminate enabled.

Configuring Termination Protection for New Clusters

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

To configure termination protection for a new cluster using the console

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Choose **Create cluster**.
3. 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.



4. Proceed with creating the cluster.

To configure termination protection for a new cluster using the AWS CLI

Using the AWS CLI, you can launch a cluster with termination protection enabled by typing the `create-cluster` command with the `--termination-protected` parameter. By default, termination protection is disabled when you launch a cluster using the AWS CLI. You can also use the `--no-termination-protected` parameter to disable termination protection.

- To launch a protected cluster, type the following command and replace *myKey* with the name of your EC2 key pair.

- Linux, UNIX, and Mac OS X users:

```
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:

```
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
```

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

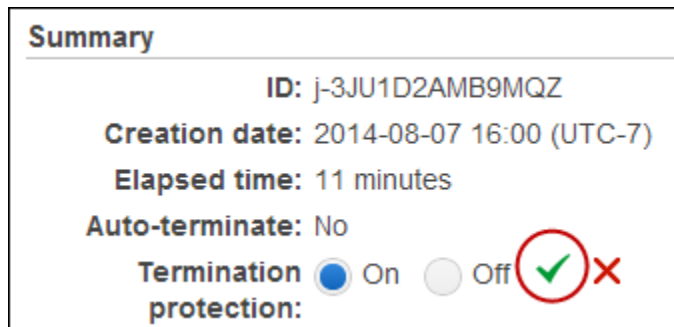
Configuring Termination Protection for Running Clusters

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

To configure termination protection for a running cluster using the console

1. Open the Amazon EMR console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. On the **Cluster List** page, click the link for your cluster.
3. On the **Cluster Details** page, in the **Summary** section, for **Termination protection**, click **Change**.

- Click **On** and then click the check mark icon to enable termination protection. Alternatively, click **Off** to disable it.



To configure termination protection for a running cluster using the AWS CLI

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

- Type the following command to enable termination protection on a running cluster.

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

To disable termination protection, type:

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

Resize a Running Cluster

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

Topics

- [Resize a Cluster Using the Console \(p. 466\)](#)
- [Resize a Cluster Using the AWS CLI \(p. 467\)](#)
- [Arrested State \(p. 468\)](#)
- [Legacy Clusters \(p. 471\)](#)

A cluster contains a single master node. The master node controls any slave nodes that are present. There are two types of slave nodes: core nodes, which store data in the Hadoop Distributed File System (HDFS), and task nodes, which do not use HDFS.

Nodes within a cluster are managed as instance groups. All clusters require a master instance group containing a single master node. Clusters using slave nodes require a core instance group that contains at least one core node. Additionally, if a cluster has a core instance group, it can also have one or more

task instance groups containing one or more task nodes. A cluster can contain up to 50 instance groups. This allows you to create up to 48 task instance groups in addition to the core and master groups.

Note

You must have at least one core node at cluster creation in order to resize the cluster. In other words, single node clusters cannot be resized.

You can resize the core instance group in a running cluster by adding nodes using the console, CLI, or 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 console, CLI, or 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 Amazon 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 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 cluster wide. 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.

Resize a Cluster Using the Console

You can use the Amazon EMR console to resize a running cluster.

To resize a running cluster using the console

1. From the **Cluster List** page, click a cluster to resize.
2. On the **Cluster Details** page, click **Resize**. Alternatively, you can expand the **Hardware Configuration** section, click the **Resize** button adjacent to the core or task groups.
3. To add nodes to the core group or to one or more task groups, click the **Resize** link in the **Count** column, change the number of instances, and click the green check mark.

To add additional task instance groups, click **Add task instance group**, choose the task node type, the number of task nodes, and whether the task nodes are spot instances. If you attempt to add more than 48 task groups, you will receive an error message. If your cluster was launched without a task group, click **Add task nodes** to add one.

Note

You can only increase the number of core nodes using the console, but you can both increase and decrease the number of task nodes.

When you make a change to the number of nodes, the Amazon EMR console updates the status of the instance group through the **Provisioning** and **Resizing** states until they are ready and indicate in brackets the newly requested number of nodes. When the change to the node count finishes, the instance groups return to the **Running** state.

Resize a Cluster Using the AWS CLI

You can use the AWS CLI to resize a running cluster. You can increase or decrease the number of task nodes, and you can increase the number of core nodes in a running cluster. It is also possible to terminate an instance in the core instance group using the AWS CLI or the API. This should be done with caution. Terminating an instance in the core instance group risks data loss, and the instance is not automatically replaced.

In addition to resizing the core and task groups, you can also add one or more task instance groups to a running cluster using the AWS CLI.

To resize a cluster by changing the instance count using the AWS CLI

You can add instances to the core group or task group, and you can remove instances from the task group using the AWS CLI `modify-instance-groups` subcommand with the `InstanceCount` parameter. To add instances to the core or task groups, increase the `InstanceCount`. To reduce the number of instances in the task group, decrease the `InstanceCount`. Changing the instance count of the task group to 0 removes all instances but not the instance group.

- To increase the number of instances in the task instance group from 3 to 4, type the following command and replace `ig-31JXXXXXXBTO` with the instance group ID.

```
aws emr modify-instance-groups --instance-groups InstanceGroupId=ig-31JXXXXXXBTO, InstanceCount=4
```

To retrieve the `InstanceGroupId`, use the `describe-cluster` subcommand. The output is a JSON object called `Cluster` that contains the ID of each instance group. To use this command, you need the cluster ID (which you can retrieve using the `aws emr list-clusters` command or the console). To retrieve the instance group ID, type the following command and replace `j-2AXXXXXXXGAPLF` with the cluster ID.

```
aws emr describe-cluster --cluster-id j-2AXXXXXXXGAPLF
```

Using the AWS CLI, you can also terminate an instance in the core instance group with the `--modify-instance-groups` subcommand. This should be done with caution. Terminating an instance in the core instance group risks data loss, and the instance is not automatically replaced. To terminate a specific instance you need the instance group ID (returned by the `aws emr describe-cluster --cluster-id` subcommand) and the instance ID (returned by the `aws emr list-instances --cluster-id` subcommand), type the following command, replace `ig-6RXXXXXX07SA` with the instance group ID and replace `i-f9XXXXf2` with the instance ID.

```
aws emr modify-instance-groups --instance-groups InstanceGroupId=ig-6RXXXXXX07SA,EC2InstanceIdsToTerminate=i-f9XXXXXf2
```

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

To resize a cluster by adding task instance groups using the AWS CLI

Using the AWS CLI, you can add between one and 48 task instance groups to a cluster with the `--add-instance-groups` subcommand. Task instances groups can only be added to a cluster containing a master instance group and a core instance group. When using the AWS CLI, you can add up to 5 task instance groups each time you use the `--add-instance-groups` subcommand.

1. To add a single task instance group to a cluster, type the following command and replace *j-JXBXXXXXX37R* with the cluster ID.

```
aws emr add-instance-groups --cluster-id j-JXBXXXXXX37R --instance-groups InstanceCount=6, InstanceGroupType=task, InstanceType=m1.large
```

2. To add multiple task instance groups to a cluster, type the following command and replace *j-JXBXXXXXX37R* with the cluster ID. You can add up to 5 task instance groups in a single command.

```
aws emr add-instance-groups --cluster-id j-JXBXXXXXX37R --instance-groups InstanceCount=6, InstanceGroupType=task, InstanceType=m1.large InstanceCount=10, InstanceGroupType=task, InstanceType=m3.xlarge
```

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

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-3KVXXXXXXY7UG* with the cluster ID.

```
aws emr describe-cluster --cluster-id j-3KVXXXXXXY7UG
```

The output will display information about your instance groups and the state of the instances:

```
{
  "Cluster": {
    "Status": {
      "Timeline": {
        "ReadyDateTime": 1413187781.245,
        "CreationDateTime": 1413187405.356
      },
      "State": "WAITING",
      "StateChangeReason": {
        "Message": "Waiting after step completed"
      }
    },
    "Ec2InstanceAttributes": {
      "Ec2AvailabilityZone": "us-west-2b"
    },
    "Name": "Development Cluster",
    "Tags": [],
    "TerminationProtected": false,
    "RunningAmiVersion": "3.2.1",
    "NormalizedInstanceHours": 16,
    "InstanceGroups": [
      {
        "RequestedInstanceCount": 1,
        "Status": {
          "Timeline": {
            "ReadyDateTime": 1413187775.749,
            "CreationDateTime": 1413187405.357
          },
          "State": "RUNNING",
          "StateChangeReason": {
            "Message": ""
          }
        },
        "Name": "MASTER",
        "InstanceGroupType": "MASTER",
        "InstanceType": "m1.large",
        "Id": "ig-3ETXXXXXXFYV8",
        "Market": "ON_DEMAND",
        "RunningInstanceCount": 1
      },
      {
        "RequestedInstanceCount": 1,
        "Status": {
          "Timeline": {
            "ReadyDateTime": 1413187781.301,
            "CreationDateTime": 1413187405.357
          },
          "State": "RUNNING",
          "StateChangeReason": {
            "Message": ""
          }
        },
        "Name": "CORE",
        "InstanceGroupType": "CORE",
        "InstanceType": "m1.large",
        "Id": "ig-3SUXXXXXXQ9ZM",
        "Market": "ON_DEMAND",
        "RunningInstanceCount": 1
      }
    ]
  }
}
```

```
    }  
    ...  
}
```

To view information on a particular instance group, type the `list-instances` subcommand with the `--cluster-id` and `--instance-group-types` parameters. You can view information for the MASTER, CORE, or TASK groups.

```
aws emr list-instances --cluster-id j-3KVXXXXXXXXY7UG --instance-group-types  
"CORE"
```

Use the `modify-instance-groups` subcommand with the `--instance-groups` parameter to reset a cluster in the ARRESTED state. The instance group id is returned by the `describe-cluster` subcommand.

```
aws emr modify-instance-groups --instance-groups InstanceGroupId=ig-  
3SUXXXXXXQ9ZM, InstanceCount=3
```

Legacy Clusters

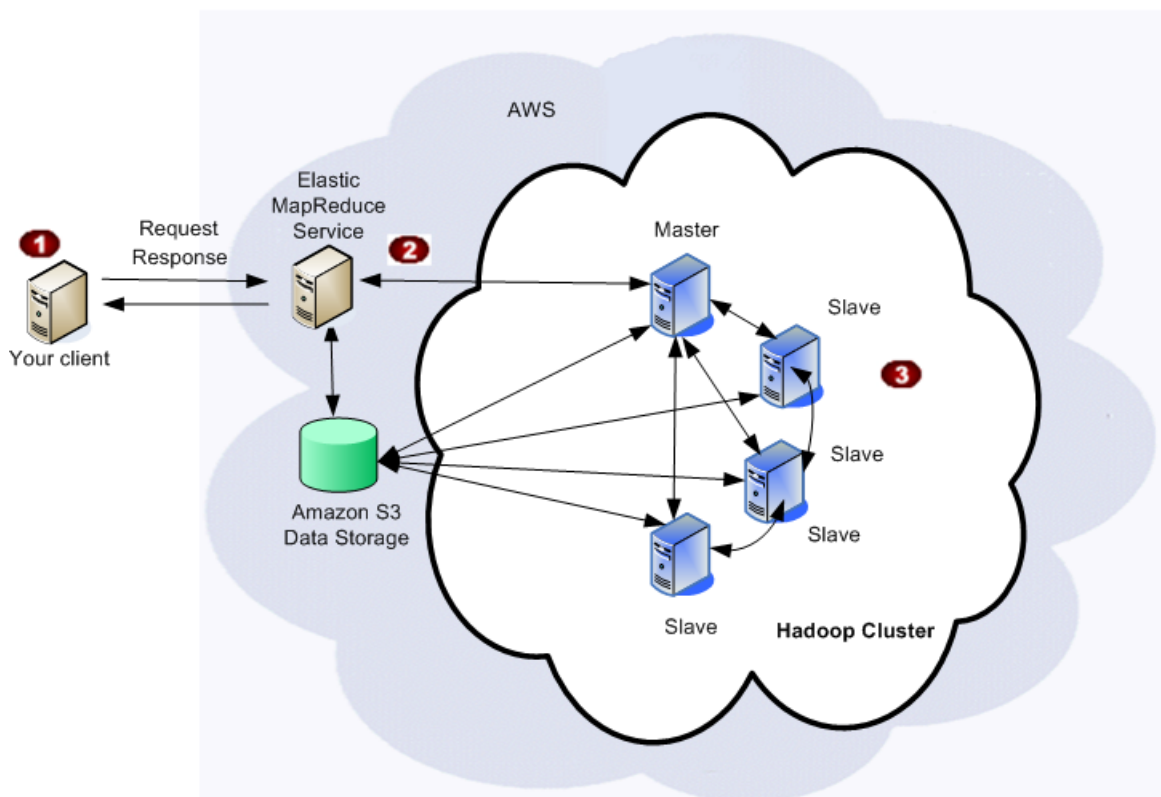
Before October 2010, Amazon EMR did not have the concept of *instance groups*. Clusters developed for Amazon EMR that were built before the option to resize running clusters was available are considered *legacy* clusters. Previously, the Amazon EMR architecture did not use instance groups to manage nodes 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

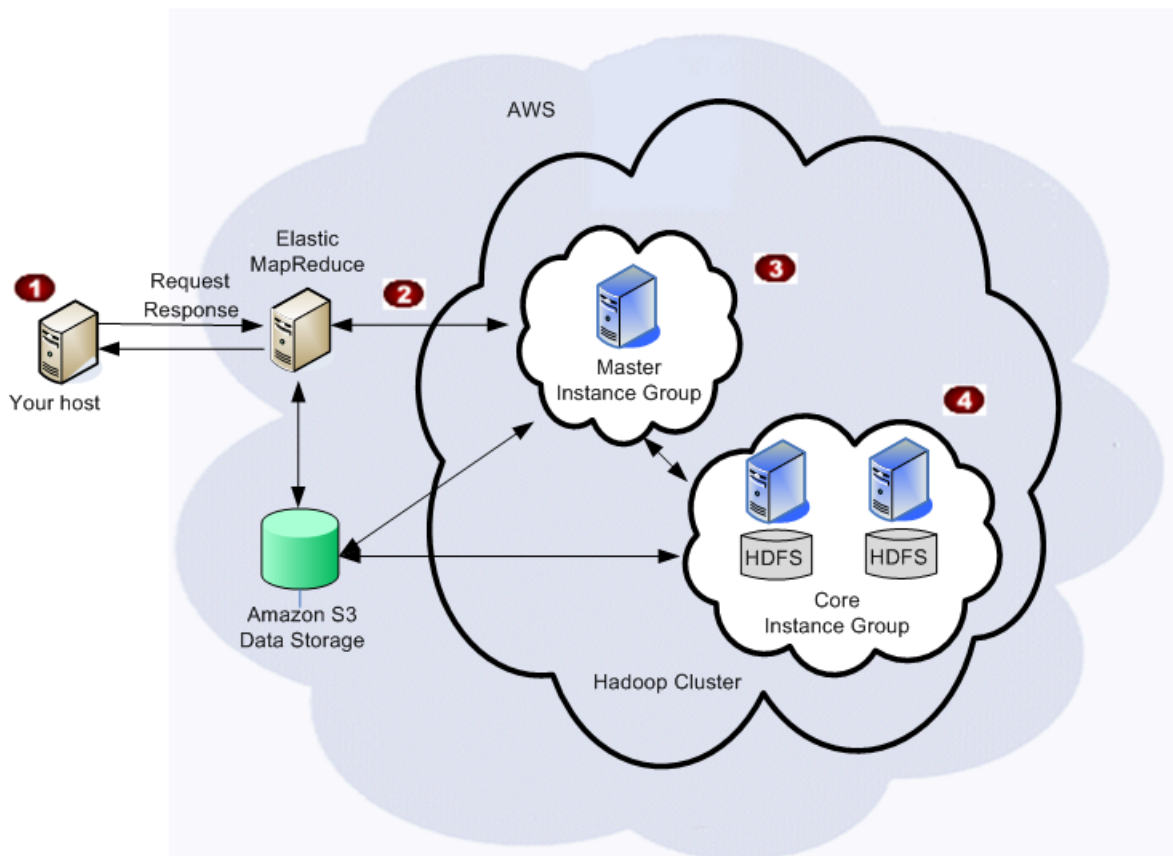
1	A legacy cluster launches and a request is sent to Amazon EMR to start the cluster.
2	Amazon EMR creates a Hadoop cluster.
3	The legacy cluster runs on a cluster consisting of a single master node and the specified number of slave nodes.

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.

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 \(p. 473\)](#).

3. The **Create Cluster** page appears with a copy of the original cluster's configuration. Review the configuration, make any necessary changes, and then click **Create Cluster**.

Submit Work to a Cluster

This section describes the methods for submitting work to an Amazon EMR cluster. You can submit work to a cluster by adding steps or by interactively submitting Hadoop jobs to the master node. You can add steps to a cluster using the AWS CLI, the Amazon EMR API, or the console. 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.

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.

The total number of step records you can view (regardless of status) is 1,000. This total includes both user-submitted and system steps. As the status of user-submitted steps changes to COMPLETED or FAILED, additional user-submitted steps can be added to the cluster until the 1,000 step limit is reached. After 1,000 steps have been added to a cluster, the submission of additional steps causes the removal of older, user-submitted step records. These records are not removed from the log files. They are removed from the console display, and they do not appear when you use the CLI or API to retrieve cluster information. System step records are never removed.

The step information you can view depends on the mechanism used to retrieve cluster information. The following tables indicate the step information returned by each of the available options.

Option	DescribeJobFlow or --describe -- jobflow	ListSteps or list-steps
SDK	256 steps	1,000 steps
Amazon EMR CLI	256 steps	NA
AWS CLI	NA	1,000 steps
API	256 steps	1,000 steps

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 \(p. 477\)](#).

Topics

- [Add Steps Using the CLI and Console \(p. 474\)](#)
- [Submit Hadoop Jobs Interactively \(p. 475\)](#)
- [Add More than 256 Steps to a Cluster \(p. 477\)](#)

Add Steps Using the CLI and Console

You can add steps to a cluster using the AWS CLI, the Amazon EMR CLI, or the console. Using the Amazon EMR console, you can add steps to a cluster when the cluster is created. You can also use the console to add steps to a running cluster if the cluster was created with the auto-terminate option disabled. Disabling the auto-terminate option creates a long-running cluster.

Add steps using the console

Whether you add steps during cluster creation or while the cluster is running, the process is similar to the following procedure.

To add a step to a running cluster using the Amazon EMR 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 --applica
tions Name=Hive Name=Pig \
--use-default-roles --ec2-attributes KeyName=myKey \
--instance-groups InstanceGroupType=MASTER,InstanceCount=1,Instance
Type=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/input
data/,-p,OUTPUT=s3://mybucket/outputdata/, $INPUT=s3://mybucket/input
data/, $OUTPUT=s3://mybucket/outputdata/]
```

- Windows users:

```
aws emr create-cluster --name "Test cluster" --ami-version 2.4 --applications Name=Hive Name=Pig --use-default-roles --ec2-attributes KeyName=myKey --instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m3.xlarge InstanceGroupType=CORE,InstanceCount=2,InstanceType=m3.xlarge --steps Type=PIG,Name="Pig Program",ActionOnFailure=CONTINUE,Args=[-f,s3://mybucket/scripts/pigscript.pig,-p,INPUT=s3://mybucket/inputdata/,-p,OUTPUT=s3://mybucket/outputdata/,$INPUT=s3://mybucket/inputdata/,$OUTPUT=s3://mybucket/outputdata/]
```

Note

The list of arguments changes depending on the type of step.

The output is a cluster identifier similar to the following:

```
{
  "ClusterId": "j-2AXXXXXXGAPLF"
}
```

To add a step to a running cluster

- Type the following command to add a step to a running cluster. Replace `j-2AXXXXXXGAPLF` with your cluster ID and replace `mybucket` with your Amazon S3 bucket name.

```
aws emr add-steps --cluster-id j-2AXXXXXXGAPLF --steps Type=PIG,Name="Pig Program",Args=[-f,s3://mybucket/scripts/pigscript.pig,-p,INPUT=s3://mybucket/inputdata/,-p,OUTPUT=s3://mybucket/outputdata/,$INPUT=s3://mybucket/inputdata/,$OUTPUT=s3://mybucket/outputdata/]
```

The output is a step identifier similar to the following:

```
{
  "StepIds": [
    "s-Y9XXXXXXAPMD"
  ]
}
```

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

Submit Hadoop Jobs Interactively

In addition to adding steps to a cluster, you can connect to the master node using an SSH client or the AWS CLI and interactively submit Hadoop jobs. For example, you can use PuTTY to establish an SSH connection with the master node and submit interactive Hive queries which are compiled into one or more Hadoop jobs.

You can submit 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. 413).

The following examples demonstrate interactively submitting Hadoop jobs and Hive jobs to the master node. The process for submitting jobs for other programming frameworks (such as Pig) is similar to these examples.

To submit Hadoop jobs interactively using the AWS CLI

- You can submit Hadoop jobs interactively using the AWS CLI by establishing an SSH connection in the CLI command (using the `ssh` subcommand).

To copy a JAR file from your local Windows machine to the master node's file system, type the following command. Replace `j-2A6HXXXXXXL7J` with your cluster ID, replace `mykey.ppk` with the name of your key pair file, and replace `myjar.jar` with the name of your JAR file.

```
aws emr put --cluster-id j-2A6HXXXXXXL7J --key-pair-file "C:\Users\user
name\Desktop\Keys\mykey.ppk" --src "C:\Users\username\myjar.jar"
```

To create an SSH connection and submit the Hadoop job `myjar.jar`, type the following command.

```
aws emr ssh --cluster-id j-2A6HXXXXXXL7J --key-pair-file "C:\Users\user
name\Desktop\Keys\mykey.ppk" --command "hadoop jar myjar.jar"
```

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

To interactively submit Hive jobs using the AWS CLI

In addition to submitting jobs to the master node via JAR files, you can submit jobs by interacting with one of the Hadoop programming frameworks running on the master node. For example, you can interactively submit Hive queries or Pig transformations at the command line, or you can submit scripts to the cluster for processing. Your commands or scripts are then compiled into one or more Hadoop jobs.

The following procedure demonstrates running a Hive script using the AWS CLI.

- If Hive is not installed on the cluster, type the following command to install it. Replace `j-2A6HXXXXXXL7J` with your cluster ID.

```
aws emr install-applications --cluster-id j-2A6HXXXXXXL7J --apps Name=Hive
```

- Create a Hive script file containing the queries or commands to run. The following example script named `my-hive.q` creates two tables, `aTable` and `anotherTable`, and copies the contents of `aTable` to `anotherTable`, replacing all data.

```
---- sample Hive script file: my-hive.q ----
create table aTable (aColumn string) ;
create table anotherTable like aTable;
insert overwrite table anotherTable select * from aTable
```

- Type the following commands to run the script from the command line using the `ssh` subcommand.

To copy `my-hive.g` from a Windows machine to your cluster, type the following command. Replace `j-2A6HXXXXXXXXL7J` with your cluster ID and replace `mykey.ppk` with the name of your key pair file.

```
aws emr put --cluster-id j-2A6HXXXXXXXXL7J --key-pair-file "C:\Users\user\name\Desktop\Keys\mykey.ppk" --src "C:\Users\username\my-hive.g"
```

To create an SSH connection and submit the Hive script `my-hive.g`, type the following command.

```
aws emr ssh --cluster-id j-2A6HXXXXXXXXL7J --key-pair-file "C:\Users\user\name\Desktop\Keys\mykey.ppk" --command "hive -f my-hive.g"
```

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

Add More than 256 Steps to a Cluster

Beginning with AMI 3.1.1 (Hadoop 2.x) and AMI 2.4.8 (Hadoop 1.x), you can submit an unlimited number of steps over the lifetime of a long-running cluster, but only 256 can be active or pending at any given time. For earlier AMI versions, the total number of steps that can be processed by a cluster is limited to 256 (including system steps such as install Hive and install Pig). For more information, see [Submit Work to a Cluster \(p. 473\)](#).

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 \(p. 441\)](#). For more information about interactively submitting Hadoop jobs, see [Submit Hadoop Jobs Interactively \(p. 475\)](#).

Automate Recurring Clusters with AWS Data Pipeline

AWS Data Pipeline is a service that automates the movement and transformation of data. You can use it to schedule moving input data into Amazon S3 and to schedule launching clusters to process that data. For example, consider the case where you have a web server recording traffic logs. If you want to run a weekly cluster to analyze the traffic data, you can use AWS Data Pipeline to schedule those clusters.

AWS Data Pipeline is a data-driven workflow, so that one task (launching the cluster) can be dependent on another task (moving the input data to Amazon S3). It also has robust retry functionality.

For more information about AWS Data Pipeline, see the [AWS Data Pipeline Developer Guide](#), especially the tutorials regarding Amazon EMR:

- [Tutorial: Launch an Amazon EMR Job Flow](#)
- [Getting Started: Process Web Logs with AWS Data Pipeline, Amazon EMR, and Hive](#)
- [Tutorial: Amazon DynamoDB Import and Export Using AWS Data Pipeline](#)

Troubleshoot a Cluster

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

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. 479)
- [Known Issues with Amazon EMR AMIs](#) (p. 481)
- [Troubleshoot a Failed Cluster](#) (p. 487)
- [Troubleshoot a Slow Cluster](#) (p. 491)
- [Common Errors in Amazon EMR](#) (p. 498)

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 Logging and Debugging \(Optional\)](#) (p. 200) and [Step 5: Test the Cluster Step by Step](#) (p. 491).

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. 480)
- [Tools to View Log Files](#) (p. 480)
- [Tools to Monitor Cluster Performance](#) (p. 480)

Tools to Display Cluster Details

You can use any of the Amazon EMR interfaces (console, CLI or API) to retrieve detailed information about a cluster. For more information, see [View Cluster Details \(p. 409\)](#).

Amazon EMR Console Details Pane

The console displays all of the clusters you've launched in the past two weeks, regardless of whether they are active or terminated. If you click on a cluster, the console displays a details pane with information about that cluster.

Amazon EMR Command Line Interface

You can locate details about a cluster from the CLI using the `--describe` argument.

Amazon EMR API

You can locate details about a cluster from the API using the `DescribeJobFlows` action.

Tools to View Log Files

Amazon EMR and Hadoop both generate log files as the cluster runs. You can access these log files from several different tools, depending on the configuration you specified when you launched the cluster. For more information, see [Configure Logging and Debugging \(Optional\) \(p. 200\)](#).

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 a suddenly terminated cluster may not be available.

Amazon EMR Console Debugging Tool

The console has a **Debug** button that you can push to open a GUI interface to browse an archived copy of the log files stored in Amazon S3. For this functionality to be available, you must have specified an Amazon S3 log path and enabled debugging when you launched the cluster and launched the cluster.

When you launch a cluster with debugging enabled, Amazon EMR creates an index of those log files. When you click **Debug**, it provides a graphical interface that you can use to browse the indexed log files.

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 \(p. 446\)](#).

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 \(p. 418\)](#).

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 \(p. 433\)](#).

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 \(p. 53\)](#).

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 \(p. 49\)](#).

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

Issue	Resolution
EMRFS DynamoDB metadata table is created in region us-east-1	Resolved in AMI 3.2.3
Hive server does not start after cluster launch	Resolved in AMI 3.2.3

Issues with AMI 3.2.0

Issue	Resolution
<p>Root volume reaches 100% capacity on instance types with fewer than 3 instance store volumes causing jobs and/or clusters to fail</p> <p>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</p>	Resolved in AMI 3.2.1

Issues with AMI 3.1.0 (deprecated)

Issue	Resolution
<p>Unable to access ResourceManager and NameNode interfaces via Lynx browser on master node using URL indicated in SSH connection details: <code>lynx://localhost:9026</code> and <code>lynx://localhost:9101</code></p>	<p>Replace localhost with the master private DNS name: <code>lynx://<i>master private DNS</i>:9026</code></p> <p>SSH connection details are correct in AMI 3.1.2</p>
<p>Restarting the instance controller causes communication failures that result in unexpected behavior:</p> <ul style="list-style-type: none"> Resizing core nodes is unsuccessful Console reports an incorrect number of core nodes and may indicate 1 or more nodes terminated due to scheduled retirement Jobs or clusters fail 	Resolved in AMI 3.1.3 and 3.2.3

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)

Issue	Resolution
Amazon Linux kernel upgrade causes Ganglia installation failure	Resolved in the Ganglia bootstrap action
ERROR: java.lang.UnsupportedOperationException: This is supposed to be overridden by subclasses... when creating an HBase snapshot	Resolved in AMI 3.1.0
Received IOException while reading '...', attempting to reopen... received intermittently while reading S3 input	Resolved in AMI 3.1.0
Hadoop NameNode and DataNode heap size incorrectly default to 1GB and are not configurable using the <code>configure-daemons</code> bootstrap action	Resolved in AMI 3.1.0

Issues with AMI 3.0.3 (deprecated)

Issue	Resolution
ERROR: java.lang.UnsupportedOperationException: This is supposed to be overridden by subclasses... when creating an HBase snapshot	Resolved in AMI 3.1.0
Received IOException while reading '...', attempting to reopen... received intermittently while reading S3 input	Resolved in AMI 3.1.0
Hadoop NameNode and DataNode heap size incorrectly default to 1GB and are not configurable using the <code>configure-daemons</code> bootstrap action	Resolved in AMI 3.1.0
yum update command produces the following error:Transaction check error: file [...] from install of [...] conflicts with file from package [...]	Resolved in AMI 3.0.4

Issues with AMI 3.0.2 (deprecated)

Issue	Resolution
Ephemeral volumes on HVM (cc.*) instances are not attached or mounted (instead a 30 GB EBS volume is attached)	Resolved in AMI 3.0.3
Received IOException while reading '...', attempting to reopen... received intermittently while reading S3 input	Resolved in AMI 3.1.0
Hadoop NameNode and DataNode heap size incorrectly default to 1GB and are not configurable using the <code>configure-daemons</code> bootstrap action	Resolved in AMI 3.1.0
<code>yum update</code> command produces the following error: Transaction check error: file [...] from install of [...] conflicts with file from package [...]	Resolved in AMI 3.0.4

Issues with AMI 3.0.1 (deprecated)

Issue	Resolution
Ephemeral volumes on HVM (cc.*) instances are not attached or mounted (instead a 30 GB EBS volume is attached)	Resolved in AMI 3.0.3
Received IOException while reading '...', attempting to reopen... received intermittently while reading S3 input	Resolved in AMI 3.1.0
Hadoop NameNode and DataNode heap size incorrectly default to 1GB and are not configurable using the <code>configure-daemons</code> bootstrap action	Resolved in AMI 3.1.0
<code>yum update</code> command produces the following error: Transaction check error: file [...] from install of [...] conflicts with file from package [...]	Resolved in AMI 3.0.4

Issues with AMI 3.0.0 (deprecated)

Issue	Resolution
Ephemeral volumes on HVM (cc.*) instances are not attached or mounted (instead a 30 GB EBS volume is attached)	Resolved in AMI 3.0.3
Received IOException while reading '...', attempting to reopen... received intermittently while reading S3 input	Resolved in AMI 3.1.0

Issue	Resolution
Hadoop NameNode and DataNode heap size incorrectly default to 1GB and are not configurable using the <code>configure-daemons</code> bootstrap action	Resolved in AMI 3.1.0
<code>yum update</code> command produces the following error: <code>Transaction check error: file [...] from install of [...] conflicts with file from package [...]</code>	Resolved in AMI 3.0.4

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

Issue	Resolution
When using the <code>distcp</code> command with LZO output, some output files are not compressed	Resolved in AMI 2.4.8

Issues with AMI 2.4.3

Issue	Resolution
When using the AWS Java SDK, the following exception occurs performing content length validation on Amazon S3 files larger than 2GB: <code>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.</code>	Resolved in AMI 2.4.4
Amazon EMR logs show numerous messages containing <code>Too many fetch-failures</code> or <code>Error reading task output</code>	Resolved in AMI 2.4.5

Issues with AMI 2.4.2

Issue	Resolution
Amazon EMR logs show numerous messages containing <code>Too many fetch-failures</code> or <code>Error reading task output</code>	Resolved in AMI 2.4.5

Issues with AMI 2.4.1

Issue	Resolution
S3DistCp spawns too many threads, causing other processes to fail when trying to create new threads	Resolved in AMI 2.4.2
Amazon EMR logs show numerous messages containing <code>Too many fetch-failures</code> or <code>Error reading task output</code>	Resolved in AMI 2.4.5

Issues with AMI 2.4.0

Issue	Resolution
Running Java 7 produces the following exception: <code>Communication exception: java.util.ConcurrentModificationException...</code>	Resolved in AMI 2.4.1
HBase shell does not function properly	Resolved in AMI 2.4.1
Amazon EMR logs show numerous messages containing <code>Too many fetch-failures</code> or <code>Error reading task output</code>	Resolved in AMI 2.4.5

Issues with AMI 2.3.6

Issue	Resolution
Amazon EMR logs show numerous messages containing <code>Too many fetch-failures</code> or <code>Error reading task output</code>	Resolved in AMI 2.4.5
Clusters with nodes using <code>m3.xlarge</code> or <code>c3.4xlarge</code> instance types fail to launch	Resolved in AMI 2.4.0

Issues with AMI 2.3.5

Issue	Resolution
Clusters with nodes using <code>m3.xlarge</code> or <code>c3.4xlarge</code> instance types fail to launch	Resolved in AMI 2.4.0

Issues with AMI 2.3.4 (deprecated)

Issue	Resolution
Clusters with nodes using <code>m3.xlarge</code> or <code>c3.4xlarge</code> instance types fail to launch	Resolved in AMI 2.4.0

Issues with AMI 2.3.3

Issue	Resolution
Streaming jobs using an input format other than <code>TextInputFormat</code> insert a byte offset on the first line of an input split causing mappers to fail	Resolved in AMI 2.3.4
Clusters with nodes using m3.xlarge or c3.4xlarge instance types fail to launch	Resolved in AMI 2.4.0

Issues with AMI 2.3.2

Issue	Resolution
Clusters with nodes using m3.xlarge or c3.4xlarge instance types fail to launch	Resolved in AMI 2.4.0

Issues with AMI 2.3.1

Issue	Resolution
Clusters with nodes using m3.xlarge or c3.4xlarge instance types fail to launch	Resolved in AMI 2.4.0

Issues with AMI 2.3.0

Issue	Resolution
Clusters with nodes using m3.xlarge or c3.4xlarge instance types fail to launch	Resolved in AMI 2.4.0

Issues with AMI 2.2.4

Issue	Resolution
Clusters with nodes using m3.xlarge or c3.4xlarge instance types fail to launch	Resolved in AMI 2.4.0

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 \(p. 491\)](#) instead.

Topics

- [Step 1: Gather Data About the Issue](#) (p. 488)
- [Step 2: Check the Environment](#) (p. 488)
- [Step 3: Look at the Last State Change](#) (p. 489)
- [Step 4: Examine the Log Files](#) (p. 490)
- [Step 5: Test the Cluster Step by Step](#) (p. 491)

Step 1: Gather Data About the Issue

The first step in troubleshooting an cluster is to gather information about what went wrong and the current status and configuration of the cluster. This information will be used in the following steps to confirm or rule out possible causes of the issue.

Define the Problem

A clear definition of the problem in the first place to begin. Some questions to ask yourself:

- What did I expect to happen? What happened instead?
- When did this problem first occur? How often has it happened since?
- Has anything changed in how I configure or run my cluster?

Cluster Details

The following cluster details are useful in helping track down issues. For more information on how to gather this information, see [View Cluster Details](#) (p. 409).

- 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](#) (p. 53).
- Type and number of EC2 instances specified for the master, core, and task nodes.

Step 2: Check the Environment

Amazon EMR operates as part of an ecosystem of web services and open-source software. Things that affect those dependencies can impact the performance of Amazon EMR.

Topics

- [Check for Service Outages](#) (p. 488)
- [Check Usage Limits](#) (p. 489)
- [Check the AMI Version](#) (p. 489)
- [Check the Amazon VPC Subnet Configuration](#) (p. 489)

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 \(p. 53\)](#).

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 [Select an Amazon VPC Subnet for the Cluster \(Optional\) \(p. 205\)](#). In addition, check that the subnet you launch the cluster into has enough free elastic IP addresses to assign one to each node in the cluster.

Step 3: Look at the Last State Change

The last state change provides information about what occurred the last time the cluster changed state. This often has information that can tell you what went wrong as a cluster changes state to `FAILED`. For example, if you launch a streaming cluster and specify an output location that already exists in Amazon S3, the cluster will fail with a last state change of "Streaming output directory already exists".

You can locate the last state change value from the console by viewing the details pane for the cluster, from the CLI using the `--describe` argument, or from the API using the `DescribeJobFlows` action. For more information, see [View Cluster Details \(p. 409\)](#).

Step 4: Examine the Log Files

The next step is to examine the log files in order to locate an error code or other indication of the issue that your cluster experienced. For information on the log files available, where to find them, and how to view them, see [View Log Files](#) (p. 413).

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 Elastic MapReduce (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 daemon logs. They are located at `/mnt/var/log/hadoop/` on each node or under `daemons/` in log files archived to Amazon S3. Note that not all cluster nodes run all daemons.

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 5: Test the Cluster Step by Step

A useful technique when you are trying to track down the source of an error is to restart the cluster and submit the steps to it one by one. This lets you check the results of each step before processing the next one, and gives you the opportunity to correct and re-run a step that has failed. This also has the advantage that you only load your input data once.

To test a cluster step by step

1. Launch a new cluster, with both keep alive and termination protection enabled. Keep alive keeps the cluster running after it has processed all of its pending steps. Termination protection prevents a cluster from shutting down in the event of an error. For more information, see [Choose the Cluster Lifecycle: Long-Running or Transient \(p. 163\)](#) and [Managing Cluster Termination \(p. 462\)](#).
2. Submit a step to the cluster. For more information, see [Submit Work to a Cluster \(p. 473\)](#).
3. When the step completes processing, check for errors in the step log files. For more information, see [Step 4: Examine the Log Files \(p. 490\)](#). 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 \(p. 462\)](#).

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 \(p. 487\)](#)

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 [Choose the Number and Type of Instances \(p. 32\)](#).

The following topics walk you through the process of identifying alternative causes of a slow cluster.

Topics

- [Step 1: Gather Data About the Issue \(p. 492\)](#)

- [Step 2: Check the Environment \(p. 492\)](#)
- [Step 3: Examine the Log Files \(p. 493\)](#)
- [Step 4: Check Cluster and Instance Health \(p. 495\)](#)
- [Step 5: Check for Arrested Groups \(p. 496\)](#)
- [Step 6: Review Configuration Settings \(p. 496\)](#)
- [Step 7: Examine Input Data \(p. 498\)](#)

Step 1: Gather Data About the Issue

The first step in troubleshooting an cluster is to gather information about what went wrong and the current status and configuration of the cluster. This information will be used in the following steps to confirm or rule out possible causes of the issue.

Define the Problem

A clear definition of the problem in the first place to begin. Some questions to ask yourself:

- What did I expect to happen? What happened instead?
- When did this problem first occur? How often has it happened since?
- Has anything changed in how I configure or run my cluster?

Cluster Details

The following cluster details are useful in helping track down issues. For more information on how to gather this information, see [View Cluster Details \(p. 409\)](#).

- 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 \(p. 53\)](#).
- Type and number of EC2 instances specified for the master, core, and task nodes.

Step 2: Check the Environment

Topics

- [Check for Service Outages \(p. 492\)](#)
- [Check Usage Limits \(p. 493\)](#)
- [Check the AMI Version \(p. 493\)](#)
- [Check the Amazon VPC Subnet Configuration \(p. 493\)](#)
- [Restart the Cluster \(p. 493\)](#)

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](#) (p. 53).

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 [Select an Amazon VPC Subnet for the Cluster \(Optional\)](#) (p. 205). In addition, check that the subnet you launch the cluster into has enough free elastic IP addresses to assign one to each node in the cluster.

Restart the Cluster

The slow down in processing may be caused by a transient condition. Consider terminating and restarting the cluster to see if performance improves.

Step 3: Examine the Log Files

The next step is to examine the log files in order to locate an error code or other indication of the issue that your cluster experienced. For information on the log files available, where to find them, and how to view them, see [View Log Files](#) (p. 413).

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 Elastic MapReduce (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 daemon logs. They are located at `/mnt/var/log/hadoop/` on each node or under `daemons/` in log files archived to Amazon S3. Note that not all cluster nodes run all daemons.

You can use the JobTracker logs to map a failed task attempt to the node it was run on. Once you know the node associated with the task attempt, you can check the health of the EC2 instance hosting that node to see if there were any issues such as running out of CPU or memory.

Step 4: Check Cluster and Instance Health

An Amazon EMR cluster is made up of nodes running on Amazon EC2 instances. If those instances become resource-bound (such as running out of CPU or memory), experience network connectivity issues, or are terminated, the speed of cluster processing suffers.

There are up to three types of nodes in a cluster:

- **master node** — manages the cluster. If it experiences a performance issue, the entire cluster is affected.
- **core nodes** — process map-reduce tasks and maintain the Hadoop Distributed Filesystem (HDFS). If one of these nodes experiences a performance issue, it can slow down HDFS operations as well as map-reduce processing. You can add additional core nodes to a cluster to improve performance, but cannot remove core nodes. For more information, see [Resize a Running Cluster \(p. 465\)](#).
- **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 [Resize a Running Cluster \(p. 465\)](#).

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 \(p. 418\)](#).

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 \(p. 433\)](#).

Check Job and HDFS Health with Hadoop Web Interfaces

Hadoop provides a series of web interfaces you can use to view information. For more information about how to access these web interfaces, see [View Web Interfaces Hosted on Amazon EMR Clusters \(p. 446\)](#).

- **JobTracker** — provides information about the progress of job being processed by the cluster. You can use this interface to identify when a job has become stuck.
- **HDFS NameNode** — provides information about the percentage of HDFS utilization and available space on each node. You can use this interface to identify when HDFS is becoming resource bound and requires additional capacity.
- **TaskTracker** — provides information about the tasks of the job being processed by the cluster. You can use this interface to identify when a task has become stuck.

Check Instance Health with Amazon EC2

Another way to look for information about the status of the instances in your cluster is to use the Amazon EC2 console. Because each node in the cluster runs on an EC2 instance, you can use tools provided by Amazon EC2 to check their status. For more information, see [View Cluster Instances in Amazon EC2](#) (p. 417).

Step 5: Check for Arrested Groups

An instance group becomes arrested when it encounters too many errors while trying to launch nodes. For example, if new nodes repeatedly fail while performing bootstrap actions, the instance group will — after some time — go into the `ARRESTED` state rather than continuously attempt to provision new nodes.

A node could fail to come up if:

- Hadoop or the cluster is somehow broken and does not accept a new node into the cluster
- A bootstrap action fails on the new node
- The node is not functioning correctly and fails to check in with Hadoop

If an instance group is in the `ARRESTED` state, and the cluster is in a `WAITING` state, you can add a cluster step to reset the desired number of slave nodes. Adding the step resumes processing of the cluster and put the instance group back into a `RUNNING` state.

For more information about how to reset a cluster in an arrested state, see [Arrested State](#) (p. 468).

Step 6: Review Configuration Settings

Configuration settings specify details about how a cluster runs, such as how many times to retry a task and how much memory is available for sorting. When you launch a cluster using Amazon EMR, there are Amazon EMR-specific settings in addition to the standard Hadoop configuration settings. The configuration settings are stored on the master node of the cluster. You can check the configuration settings to ensure that your cluster has the resources it requires to run efficiently.

Amazon EMR defines default Hadoop configuration settings that it uses to launch a cluster. The values are based on the AMI and the instance type you specify for the cluster. For more information, see [Hadoop Configuration Reference](#) (p. 522). 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. 124) and [Configuration of `hadoop-user-env.sh`](#) (p. 526). 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/>.

Amazon Elastic MapReduce Developer Guide
Step 6: Review Configuration Settings

Configuration Setting	Description
dfs.replication	The number of HDFS nodes to which a single block (like the hard drive block) is copied to in order to produce a RAID-like environment. Determines the number of HDFS nodes which contain a copy of the block.
io.sort.mb	Total memory available for sorting. This value should be 10x io.sort.factor. This setting can also be used to calculate total memory used by task node by figuring io.sort.mb multiplied by mapred.tasktracker.ap.tasks.maximum.
io.sort.spill.percent	Used during sort, at which point the disk will start to be used because the allotted sort memory is getting full.
mapred.child.java.opts	Deprecated. Use mapred.map.child.java.opts and mapred.reduce.child.java.opts instead. The Java options TaskTracker uses when launching a JVM for a task to execute within. A common parameter is "-Xmx" for setting max memory size.
mapred.map.child.java.opts	The Java options TaskTracker uses when launching a JVM for a map task to execute within. A common parameter is "-Xmx" for setting max memory heap size.
mapred.map.tasks.speculative.execution	Determines whether map task attempts of the same task may be launched in parallel.
mapred.reduce.tasks.speculative.execution	Determines whether reduce task attempts of the same task may be launched in parallel.
mapred.map.max.attempts	The maximum number of times a map task can be attempted. If all fail, then the map task is marked as failed.
mapred.reduce.child.java.opts	The Java options TaskTracker uses when launching a JVM for a reduce task to execute within. A common parameter is "-Xmx" for setting max memory heap size.
mapred.reduce.max.attempts	The maximum number of times a reduce task can be attempted. If all fail, then the map task is marked as failed.
mapred.reduce.slowstart.completed.maps	The amount of maps tasks that should complete before reduce tasks are attempted. Not waiting long enough may cause "Too many fetch-failure" errors in attempts.
mapred.reuse.jvm.num.tasks	A task runs within a single JVM. Specifies how many tasks may reuse the same JVM.
mapred.tasktracker.map.tasks.maximum	The max amount of tasks that can execute in parallel per task node during mapping.
mapred.tasktracker.reduce.tasks.maximum	The max amount of tasks that can execute in parallel per task node during reducing.

If your cluster tasks are memory-intensive, you can enhance performance by using fewer tasks per core node and reducing your job tracker heap size.

Step 7: Examine Input Data

Look at your input data. Is it distributed evenly among your key values? If your data is heavily skewed towards one or few key values, the processing load may be mapped to a small number of nodes, while other nodes idle. This imbalanced distribution of work can result in slower processing times.

An example of an imbalanced data set would be running a cluster to alphabetize words, but having a data set that contained only words beginning with the letter "a". When the work was mapped out, the node processing values beginning with "a" would be overwhelmed, while nodes processing words beginning with other letters would go idle.

Common Errors in Amazon EMR

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 \(p. 498\)](#)
- [Permissions Errors \(p. 500\)](#)
- [Memory Errors \(p. 501\)](#)
- [Resource Errors \(p. 502\)](#)
- [Streaming Cluster Errors \(p. 506\)](#)
- [Custom JAR Cluster Errors \(p. 507\)](#)
- [Hive Cluster Errors \(p. 507\)](#)
- [VPC Errors \(p. 509\)](#)
- [AWS GovCloud \(US\) Errors \(p. 511\)](#)
- [Other Issues \(p. 511\)](#)

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? \(p. 499\)](#)
- [Are you trying to recursively traverse input directories? \(p. 499\)](#)
- [Does your output directory already exist? \(p. 499\)](#)
- [Are you trying to specify a resource using an HTTP URL? \(p. 499\)](#)
- [Are you referencing an Amazon S3 bucket using an invalid name format? \(p. 499\)](#)
- [Are you experiencing trouble loading data to or from Amazon S3? \(p. 499\)](#)

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 [File Systems Compatible with Amazon EMR](#) (p. 138).

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.

- Data changes in Amazon S3 may not be available to all clients immediately, depending on the consistency model. Amazon S3 uses two consistency models: eventual and read-after-write. Eventual consistency means that data changes may take a moment to be available to all clients. Read-after-write consistency means that PUTS of new objects are available immediately, but other operations still use eventual consistency. The consistency model that Amazon S3 uses depends on the region you specified

when you created the bucket. For more information, see [Amazon S3 Concepts](#). Your application needs to be able to handle the delays associated with eventual consistency.

- 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 \(AMI\) \(p. 48\)](#).
- 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? \(p. 508\)](#).

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? \(p. 500\)](#)
- [If you are using IAM, do you have the proper Amazon EC2 policies set? \(p. 501\)](#)

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 \(p. 441\)](#).

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 [Set Access Policies for IAM Users \(p. 183\)](#).

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 -s,mapred.tasktracker.map.tasks.maximum=maximum number of simultaneous map tasks  
--args -s,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 DataNode 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? \(p. 503\)](#)
- [Are you seeing "EC2 Quota Exceeded" errors? \(p. 503\)](#)
- [Are you seeing "Too many fetch-failures" errors? \(p. 503\)](#)
- [Are you seeing "File could only be replicated to 0 nodes instead of 1" errors? \(p. 504\)](#)

- [Are your TaskTracker nodes being blacklisted?](#) (p. 505)

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 \(p. 124\)](#).
- 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 [Choose the Number and Type of Instances \(p. 32\)](#).
- 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 [Resize a Running Cluster \(p. 465\)](#) and [Monitor Metrics with CloudWatch \(p. 418\)](#).

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 \(p. 413\)](#).

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 \(p. 124\)](#).
- If the TaskTracker node has more than 4 failed tasks. You can change this to a higher value using the `mapred.max.tracker.failures` configuration parameter. Other configuration settings you might want to change are the settings that control how many times to attempt a task before marking it as failed: `mapred.map.max.attempts` for map tasks and `mapreduce.reduce.maxattempts` for reduce tasks. For more information about changing Hadoop configuration settings, see [\(Optional\) Create Bootstrap Actions to Install Additional Software \(p. 124\)](#).

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 [Amazon CloudWatch Command Line Interface Reference](#).

```
mon-get-stats NoOfBlackListedTaskTrackers --dimensions JobFlowId=JobFlowID -  
-statistics Maximum --namespace AWS/ElasticMapReduce
```

The following example shows how to launch a cluster and use a bootstrap action to set the value of `mapred.max.tracker.failures` to 7, instead of the default 4.

Type the following command using the AWS CLI and replace *myKey* with the name of your EC2 key pair.

```
aws emr create-cluster --name "Test cluster" --ami-version 2.4 --applications  
Name=Hive Name=Pig \  
--use-default-roles --ec2-attributes KeyName=myKey \  
--instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=m3.xlarge \  
InstanceGroupType=CORE,InstanceCount=2,InstanceType=m3.xlarge \  
--bootstrap-action Path=s3://elasticmapreduce/bootstrap-actions/configure-ha  
doop,Name="Modified mapred.max.tracker.failures",Args=[ "-m", "mapred.max.track  
er.failures=7" ]
```

Note

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

When you launch a cluster using the preceding example, you can connect to the master node and see the changed configuration setting in `/home/hadoop/conf/mapred-site.xml`. The modified line will appear as shown in the following example.

```
<property><name>mapred.max.tracker.failures</name><value>7</value></property>
```

Streaming Cluster Errors

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? \(p. 506\)](#)
- [Is your script timing out? \(p. 506\)](#)
- [Are you passing in invalid streaming arguments? \(p. 506\)](#)
- [Did your script exit with an error? \(p. 507\)](#)

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 \(p. 413\)](#).

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 \(p. 413\)](#).

You can change the time limit by setting a new value for the `mapred.task.timeout` configuration setting. This setting specifies the number of milliseconds after which Amazon EMR will terminate a task that has not read input, written output, or updated its status string. You can update this value by passing an additional streaming argument `-jobconf mapred.task.timeout=800000`.

Are you passing in invalid streaming arguments?

Hadoop streaming supports only the following arguments. If you pass in arguments other than those listed below, the cluster will fail.

```
-blockAutoGenerateCacheFiles  
-cacheArchive  
-cacheFile
```

```
-cmdenv  
-combiner  
-debug  
-input  
-inputformat  
-inputreader  
-jobconf  
-mapper  
-numReduceTasks  
-output  
-outputformat  
-partitioner  
-reducer  
-verbose
```

In addition, Hadoop streaming only recognizes arguments passed in using Java syntax; that is, preceded by a single hyphen. If you pass in arguments preceded by a double hyphen, the cluster will fail.

Did your script exit with an error?

If your mapper or reducer script exits with an error, you can locate the error in the `stderr` file of task attempt logs of the failed task attempt. For more information, see [View Log Files \(p. 413\)](#).

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? \(p. 507\)](#)
- [Is your JAR throwing an error inside a map task? \(p. 507\)](#)

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 \(p. 413\)](#).

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 \(p. 413\)](#).

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? \(p. 508\)](#)
- [Did you encounter a syntax error in the Hive script? \(p. 508\)](#)
- [Did a job fail when running interactively? \(p. 508\)](#)
- [Are you having trouble loading data to or from Amazon S3 into Hive? \(p. 508\)](#)

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 \(p. 251\)](#).

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 \(p. 413\)](#).

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 \(p. 413\)](#).

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? \(p. 499\)](#). If none of those issues are the cause, consider the following options specific to Hive.

- Make sure you are 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 \(p. 251\)](#).
- Using `INSERT OVERWRITE` requires listing the contents of the Amazon S3 bucket or folder. This is an expensive operation. If possible, manually prune the path instead of having Hive list and delete the existing objects.
- Pre-cache the results of an Amazon S3 list operation locally on the cluster. You do this in HiveQL with the following command: `set hive.optimize.s3.query=true;`
- Use static partitions where possible.
- In some versions of Hive and Amazon EMR, it is possible that using `ALTER TABLES` will fail because the table is stored in a different location than expected by Hive. The solution is to add or update following in `/home/hadoop/conf/core-site.xml`:

```
<property>
  <name>fs.s3n.endpoint</name>
  <value>s3.amazonaws.com</value>
</property>
```

VPC Errors

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 \(p. 509\)](#)
- [Missing DHCP Options Set \(p. 509\)](#)
- [Permissions Errors \(p. 510\)](#)
- [Errors That Result in START_FAILED \(p. 510\)](#)

Invalid Subnet Configuration

On the **Cluster Details** page, in the **Status** field, you see an error similar to the following:

```
The subnet configuration was invalid: Cannot find route to InternetGateway in main RouteTable rtb-id for vpc vpc-id.
```

To solve this problem, you must create an Internet Gateway and attach it to your VPC. For more information, see [Adding an Internet Gateway to Your VPC](#).

Alternatively, verify that you have configured your VPC with **Enable DNS resolution** and **Enable DNS hostname support** enabled. For more information, see [Using DNS with Your VPC](#).

Missing DHCP Options Set

You see a step failure in the cluster system log (syslog) with an error similar to the following:

```
ERROR org.apache.hadoop.security.UserGroupInformation (main):  
PrivilegedActionException as:hadoop (auth:SIMPLE) cause:java.io.IOException:  
org.apache.hadoop.yarn.exceptions.ApplicationNotFoundException: Application  
with id 'application_id' doesn't exist in RM.
```

or

```
ERROR org.apache.hadoop.streaming.StreamJob (main): Error Launching job :  
org.apache.hadoop.yarn.exceptions.ApplicationNotFoundException: Application  
with id 'application_id' doesn't exist in RM.
```

To solve this problem, you must configure a VPC that includes a DHCP Options Set whose parameters are set to the following values:

Note

If you use the AWS GovCloud (US) region, set domain-name to `us-gov-west-1.compute.internal` instead of the value used in the following example.

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

Use `ec2.internal` if your region is US East (N. Virginia). For other regions, use `region-name.compute.internal`. For example in us-west-2, use `domain-name=us-west-2.compute.internal`.

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

For more information, see [DHCP Options Sets](#).

Permissions Errors

A failure in the `stderr` log for a step indicates that an Amazon S3 resource does not have the appropriate permissions. This is a 403 error and the error looks like:

```
Exception in thread "main" com.amazonaws.services.s3.model.AmazonS3Exception:  
Access Denied (Service: Amazon S3; Status Code: 403; Error Code: AccessDenied;  
Request ID: REQUEST_ID)
```

If the `ActionOnFailure` is set to `TERMINATE_JOB_FLOW`, then this would result in the cluster terminating with the state, `SHUTDOWN_COMPLETED_WITH_ERRORS`.

A few ways to troubleshoot this problem include:

- If you are using an Amazon S3 bucket policy within a VPC, make sure to give access to all buckets by creating a VPC endpoint and selecting **Allow all** under the Policy option when creating the endpoint.
- Make sure that any policies associated with S3 resources include the VPC in which you launch the cluster.
- Try running the following command from your cluster to verify you can access the bucket

```
hadoop fs -copyToLocal s3://path-to-bucket /tmp/
```

- You can get more specific debugging information by setting the `log4j.logger.org.apache.http.wire` parameter to `DEBUG` in `/home/hadoop/conf/log4j.properties` file on the cluster. You can check the `stderr` log file after trying to access the bucket from the cluster. The log file will provide more detailed information:

```
Access denied for getting the prefix for bucket - us-west-2.elasticmapreduce  
with path samples/wordcount/input/  
15/03/25 23:46:20 DEBUG http.wire: >> "GET /?prefix=samples%2Fwordcount%2Fin  
put%2F&delimiter=%2F&max-keys=1 HTTP/1.1[\r][\n]"  
15/03/25 23:46:20 DEBUG http.wire: >> "Host: us-west-2.elasticmapre  
duce.s3.amazonaws.com[\r][\n]"
```

Errors That Result in `START_FAILED`

Before AMI 3.7.0, for VPCs where a hostname is specified Amazon EC2 instances, Amazon EMR Amazon EMR maps the internal hostnames of the subnet with custom domain addresses as follows:

`ip-X.X.X.X.customdomain.com.tld`. For example, if the hostname was `ip-10.0.0.10` and the VPC has the domain name option set to `customdomain.com`, the resulting hostname mapped by Amazon EMR would be `ip-10.0.1.0.customdomain.com`. An entry is added in `/etc/hosts` to resolve the hostname to `10.0.0.10`. This behavior is changed with AMI 3.7.0 and now Amazon EMR honors the DHCP configuration of the VPC entirely. Previously, customers could also use a bootstrap action to specify a hostname mapping.

If you would like to preserve this behavior, you must provide the DNS and forward resolution setup you require for the custom domain.

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 \(p. 185\)](#).
- 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 \(p. 509\)](#).

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 \(p. 498\)](#).

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.

Topics

- [End-to-End Amazon EMR Java Source Code Sample \(p. 512\)](#)
- [Common Concepts for API Calls \(p. 516\)](#)
- [Use SDKs to Call Amazon EMR APIs \(p. 518\)](#)

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 \(p. 518\)](#).

Important

The maximum request rate for Amazon EMR is one request every ten seconds.

End-to-End Amazon EMR Java Source Code Sample

Developers can call the Amazon EMR API using custom Java code to do the same things possible with the Amazon EMR console or CLI. This section provides the end-to-end steps necessary to install the AWS Toolkit for Eclipse and run a fully-functional Java source code sample that adds steps to an Amazon EMR cluster.

Note

This example focuses on Java, but Amazon EMR also supports several programming languages with a collection of Amazon EMR SDKs. For more information, see [Use SDKs to Call Amazon EMR APIs \(p. 518\)](#).

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. Click **Help** and click **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, click **Database Development** and click **Finish**.
5. Restart Eclipse when prompted.

Next, install the AWS Toolkit for Eclipse to make the helpful, pre-configured source code project templates available.

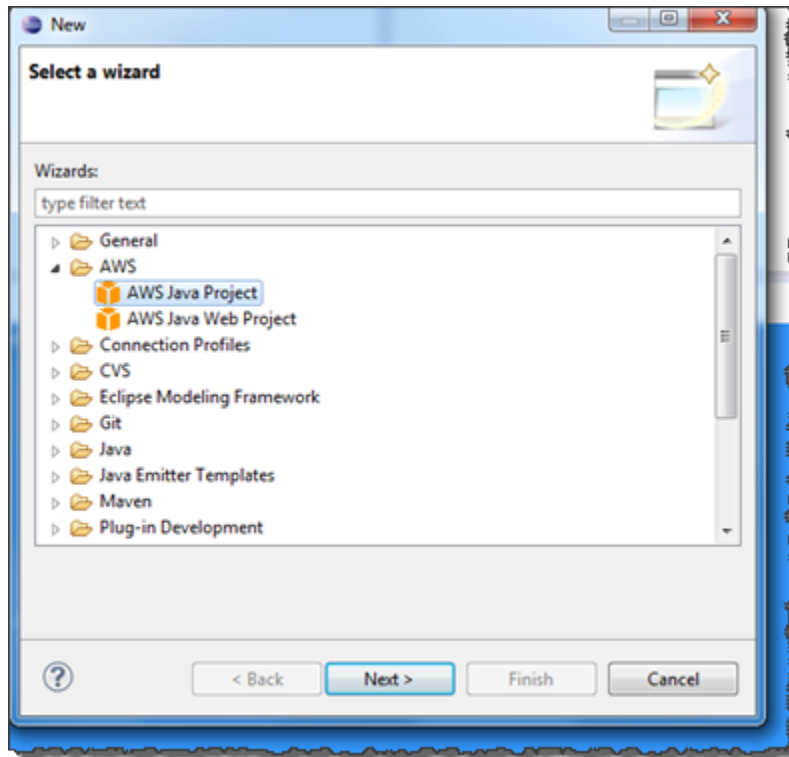
To install the AWS Toolkit for Eclipse

1. Open the Eclipse IDE.
2. Click **Help** and click **Install New Software**.
3. In the **Work with:** field, type `http://aws.amazon.com/eclipse`.
4. In the items list, click **AWS Toolkit for Eclipse** and click **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. Click **File**, click **New**, and click **Other**.
3. In the **Select a wizard** dialog, choose **AWS Java Project**, and click **Next**.

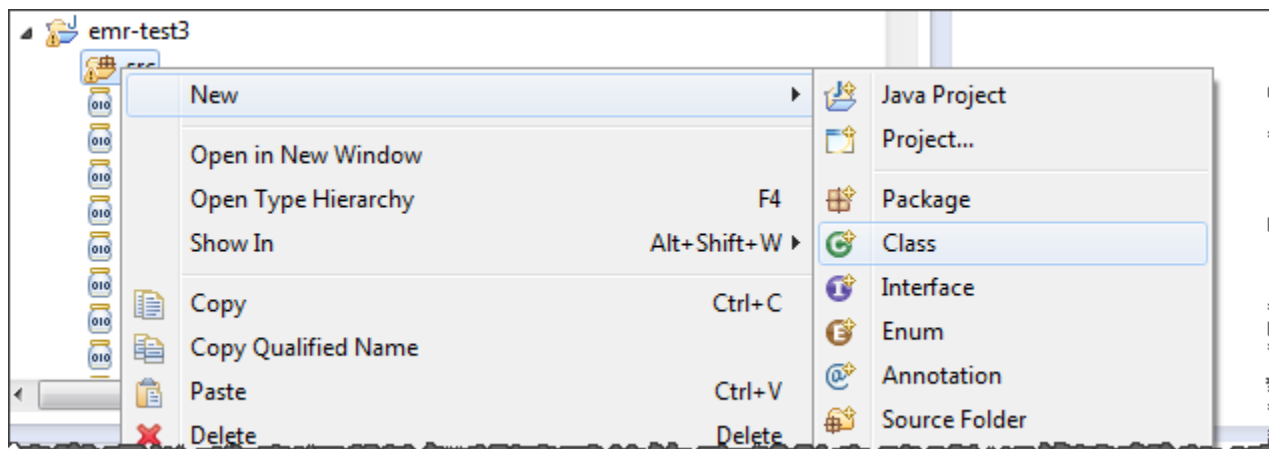


4. In the **New AWS Java Project** dialog, in the **Project name:** field, enter the name of your new project, for example `EMR-sample-code`.
5. Click **Configure AWS accounts...**, enter your public and private access keys, and click **Finish**. For more information about creating access keys, see [How 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, click **New**, and click **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 click **Finish**.
9. Enter the Java source code inside your new class and add the appropriate **import** statements for the classes and methods in the sample. For your convenience, the full source code listing is shown below:

Note

In the following sample code, replace the example cluster ID (`j-1HTE8WKS7SODR`) with a valid cluster ID in your account. In addition, replace the example Amazon S3 path (`s3://mybucket/my-jar-location1`) with the valid path to your JAR. Lastly, replace the example class name (`com.my.Main1`) with the correct name of the class in your JAR, if applicable.

```
import java.io.IOException;
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.PropertiesCredentials;
import com.amazonaws.services.elasticmapreduce.*;
import com.amazonaws.services.elasticmapreduce.model.AddJobFlowStepsRequest;
import com.amazonaws.services.elasticmapreduce.model.AddJobFlowStepsResult;
import com.amazonaws.services.elasticmapreduce.model.HadoopJarStepConfig;
import com.amazonaws.services.elasticmapreduce.model.StepConfig;
import com.amazonaws.services.elasticmapreduce.util.StepFactory;

public class main {

    public static void main(String[] args) {

        AWSCredentials credentials = null;
        try {
            credentials = new PropertiesCredentials(
                main.class.getResourceAsStream("AwsCredentials.properties"));
        } catch (IOException e1) {
            System.out.println("Credentials were not properly entered into
            AwsCredentials.properties.");
            System.out.println(e1.getMessage());
            System.exit(-1);
        }

        AmazonElasticMapReduce client = new AmazonElasticMapReduceClient(credentials);

        // predefined steps. See StepFactory for list of predefined steps
        StepConfig hive = new StepConfig("Hive", new StepFactory().newInstall
        HiveStep());

        // A custom step
        HadoopJarStepConfig hadoopConfig1 = new HadoopJarStepConfig()
            .withJar("s3://mybucket/my-jar-location1")
            .withMainClass("com.my.Main1") // optional main class, this can be
            omitted if jar above has a manifest
            .withArgs("--verbose"); // optional list of arguments
        StepConfig customStep = new StepConfig("Step1", hadoopConfig1);

        AddJobFlowStepsResult result = client.addJobFlowSteps(new AddJob
        FlowStepsRequest()
            .withJobFlowId("j-1HTE8WKS7SODR")
            .withJobFlowRole("jobflow_role")
            .withServiceRole("service_role")
```

```
.withSteps(hive, customStep));  
System.out.println(result.getStepIds());  
  
}  
}
```

10. Click **Run**, **Run As**, and click **Java Application**.
11. If the sample runs correctly, a list of IDs for the new steps appears in the Eclipse IDE console window. The correct output is similar to the following:

```
[s-39BLQZRJB2E5E, s-1L6A4ZU2SAURC]
```

Common Concepts for API Calls

Topics

- [Endpoints for Amazon EMR \(p. 516\)](#)
- [Specifying Cluster Parameters in Amazon EMR \(p. 516\)](#)
- [Availability Zones in Amazon EMR \(p. 517\)](#)
- [How to Use Additional Files and Libraries in Amazon EMR Clusters \(p. 517\)](#)
- [Amazon EMR Sample Applications \(p. 517\)](#)

When you write an application that calls the Amazon Elastic MapReduce (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 s3n://bucket/path_to_executable#local_path
```

The file, `local_path`, is in the working directory of the mapper, which could reference the file.

Amazon EMR Sample Applications

AWS provides tutorials that show you how to create complete applications, including:

- [Contextual Advertising using Apache Hive and Amazon EMR with High Performance Computing instances](#)
- [Parsing Logs with Apache Pig and Elastic MapReduce](#)
- [Finding Similar Items with Amazon EMR, Python, and Hadoop Streaming](#)
- [ItemSimilarity](#)
- [Word Count Example](#)

For more Amazon EMR code examples, go to [Sample Code & Libraries](#).

Use SDKs to Call Amazon EMR APIs

Topics

- [Using the AWS SDK for Java to Create an Amazon EMR Cluster \(p. 518\)](#)
- [Using the AWS SDK for .Net to Create an Amazon EMR Cluster \(p. 519\)](#)
- [Using the Java SDK to Sign an API Request \(p. 521\)](#)

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, go to [Sample Code & Libraries](#). SDKs are currently available for the following languages/platforms:

- [Java](#)
- [Node.js](#)
- [PHP](#)
- [Python](#)
- [Ruby](#)
- [Windows and .NET](#)

Using the AWS SDK for Java to Create an Amazon EMR Cluster

The AWS SDK for Java provides three packages with Amazon Elastic MapReduce (Amazon EMR) functionality:

- [com.amazonaws.services.elasticmapreduce](#)
- [com.amazonaws.services.elasticmapreduce.model](#)
- [com.amazonaws.services.elasticmapreduce.util](#)

For more information about these packages, go to the [AWS SDK for Java API Reference](#).

The following example illustrates how the SDKs can simplify programming with Amazon EMR. The code sample below uses the `StepFactory` object, a helper class for creating common Amazon EMR step types, to create an interactive Hive cluster with debugging enabled.

Note

If you are adding IAM user visibility to a new cluster, call `RunJobFlow` and set `VisibleToAllUsers=true`, otherwise IAM users cannot view the cluster.

```
AWSCredentials credentials = new BasicAWSCredentials(accessKey, secretKey);

AmazonElasticMapReduceClient emr = new AmazonElasticMapReduceClient(credentials);

StepFactory stepFactory = new StepFactory();

StepConfig enableddebugging = new StepConfig()
```

Amazon Elastic MapReduce Developer Guide Using the AWS SDK for .Net to Create an Amazon EMR Cluster

```
.withName("Enable debugging")
.withActionOnFailure("TERMINATE_JOB_FLOW")
.withHadoopJarStep(stepFactory.newEnableDebuggingStep());

StepConfig installHive = new StepConfig()
    .withName("Install Hive")
    .withActionOnFailure("TERMINATE_JOB_FLOW")
    .withHadoopJarStep(stepFactory.newInstallHiveStep());

RunJobFlowRequest request = new RunJobFlowRequest()
    .withName("Hive Interactive")
    .withAmiVersion("3.8")
    .withSteps(enableddebugging, 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:::role/EMR_DefaultRole"
  "RoleName": "EMR_EC2_DefaultRole",
  "Arn": "arn:aws:iam:::role/EMR_EC2_DefaultRole"
```

If the default roles do not exist, you can use the following AWS CLI command to create them:

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

Using the AWS SDK for .Net to Create an Amazon EMR Cluster

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.

Amazon Elastic MapReduce Developer Guide Using the AWS SDK for .Net to Create an Amazon EMR Cluster

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 enableddebugging = 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 = {enableddebugging, 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:

```
aws iam list-roles | grep EMR
```

Both the instance profile (`EMR_EC2_DefaultRole`) and the service role (`EMR_DefaultRole`) will be displayed if they exist:

```
"RoleName": "EMR_DefaultRole",
"Arn": "arn:aws:iam::AccountID:role/EMR_DefaultRole"
"RoleName": "EMR_EC2_DefaultRole",
"Arn": "arn:aws:iam::AccountID:role/EMR_EC2_DefaultRole"
```

If the default roles do not exist, you can use the following AWS CLI command to create them:

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

Using the Java SDK to Sign an API Request

Amazon EMR uses the Signature Version 4 signing process to construct signed requests to AWS. For more information, see [Signature Version 4 Signing Process](#).

Hadoop Configuration Reference

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

Apache Hadoop runs on the EC2 instances launched in a cluster. Amazon Elastic MapReduce (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. 48).

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. 522)
- [Configuration of hadoop-user-env.sh](#) (p. 526)
- [Hadoop 2.2.0 and 2.4.0 Default Configuration](#) (p. 527)
- [Hadoop 1.0.3 Default Configuration](#) (p. 553)
- [Hadoop 20.205 Default Configuration \(Deprecated\)](#) (p. 569)

JSON Configuration Files

Topics

- [Node Settings](#) (p. 523)
- [Cluster Configuration](#) (p. 524)

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.

Parameter	Description
<code>isMaster</code>	Indicates that is the master node. Type: Boolean
<code>isRunningNameNode</code>	Indicates that this node is running the Hadoop name node daemon. Type: Boolean
<code>isRunningDataNode</code>	Indicates that this node is running the Hadoop data node daemon. Type: Boolean
<code>isRunningJobTracker</code>	Indicates that this node is running the Hadoop job tracker daemon. Type: Boolean
<code>isRunningTaskTracker</code>	Indicates that this node is running the Hadoop task tracker daemon. Type: Boolean

Hadoop 2.2.0 adds the following parameters to the `instance.json` file.

Parameter	Description
<code>isRunningResourceManager</code>	Indicates that this node is running the Hadoop resource manager daemon. Type: Boolean
<code>isRunningNodeManager</code>	Indicates that this node is running the Hadoop node manager daemon. Type: Boolean

The following example shows the contents of an `instance.json` file:

```
{
  "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.

- Linux, UNIX, and Mac OS X users:

```
aws emr create-cluster --name "Test cluster" --ami-version 3.3 --applica-
tions 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 --applica-
tions 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>.

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.

Parameter	Description
JobFlowID	Contains the ID for the cluster. Type: String
jobFlowCreationInstant	Contains the time that the cluster was created. Type: Long
instanceCount	Contains the number of nodes in an instance group. Type: Integer
masterInstanceID	Contains the ID for the master node. Type: String
masterPrivateDnsName	Contains the private DNS name of the master node. Type: String

Parameter	Description
masterInstanceType	Contains the EC2 instance type of the master node. Type: String
slaveInstanceType	Contains the EC2 instance type of the slave nodes. Type: String
HadoopVersion	Contains the version of Hadoop running on the cluster. Type: String
instanceGroups	A list of objects specifying each instance group in the cluster instanceGroupId—unique identifier for this instance group. Type: String instanceGroupName—User defined name of the instance group. Type: String instanceRole—one of Master, Core, or Task. Type: String instanceType—the Amazon EC2 type of the node, such as "m1.small". Type: String requestedInstanceCount—the target number of nodes for this instance group. Type: Long

The following example shows the contents of an `job-flow.json` file.

```
{
  "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",
      "instanceType": "AmazonEC2InstanceType",
      "requestedInstanceCount": Count
    }
  ]
}
```

```
{
  "instanceGroupId": "InstanceGroupID",
  "instanceGroupName": "Name",
  "instanceRole": "Task",
  "marketType": "Type",
  "instanceType": "AmazonEC2InstanceType",
  "requestedInstanceCount": Count
}
```

Configuration of `hadoop-user-env.sh`

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. 124).

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. 527)
- [HDFS Configuration \(Hadoop 2.2.0\)](#) (p. 538)
- [Task Configuration \(Hadoop 2.2.0\)](#) (p. 538)
- [Intermediate Compression \(Hadoop 2.2.0\)](#) (p. 551)

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. 48).

Hadoop Configuration (Hadoop 2.2.0, 2.4.0)

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. 48).

m1.medium

Parameter	Value
YARN_RESOURCEMANAGER_HEAPSIZE	384
YARN_PROXYSERVER_HEAPSIZE	192
YARN_NODEMANAGER_HEAPSIZE	256
HADOOP_JOB_HISTORYSERVER_HEAPSIZE	256
HADOOP_NAMENODE_HEAPSIZE	384

Parameter	Value
HADOOP_DATANODE_HEAPSIZE	192

m1.large

Parameter	Value
YARN_RESOURCEMANAGER_HEAPSIZE	768
YARN_PROXYSERVER_HEAPSIZE	384
YARN_NODEMANAGER_HEAPSIZE	512
HADOOP_JOB_HISTORYSERVER_HEAPSIZE	512
HADOOP_NAMENODE_HEAPSIZE	768
HADOOP_DATANODE_HEAPSIZE	384

m1.xlarge

Parameter	Value
YARN_RESOURCEMANAGER_HEAPSIZE	1024
YARN_PROXYSERVER_HEAPSIZE	512
YARN_NODEMANAGER_HEAPSIZE	768
HADOOP_JOB_HISTORYSERVER_HEAPSIZE	1024
HADOOP_NAMENODE_HEAPSIZE	2304
HADOOP_DATANODE_HEAPSIZE	384

m2.xlarge

Parameter	Value
YARN_RESOURCEMANAGER_HEAPSIZE	1536
YARN_PROXYSERVER_HEAPSIZE	1024
YARN_NODEMANAGER_HEAPSIZE	1024
HADOOP_JOB_HISTORYSERVER_HEAPSIZE	1024
HADOOP_NAMENODE_HEAPSIZE	3072
HADOOP_DATANODE_HEAPSIZE	384

m2.2xlarge

Parameter	Value
YARN_RESOURCEMANAGER_HEAPSIZE	1536
YARN_PROXYSERVER_HEAPSIZE	1024
YARN_NODEMANAGER_HEAPSIZE	1024
HADOOP_JOB_HISTORYSERVER_HEAPSIZE	1536
HADOOP_NAMENODE_HEAPSIZE	6144
HADOOP_DATANODE_HEAPSIZE	384

m2.4xlarge

Parameter	Value
YARN_RESOURCEMANAGER_HEAPSIZE	2048
YARN_PROXYSERVER_HEAPSIZE	1024
YARN_NODEMANAGER_HEAPSIZE	1536
HADOOP_JOB_HISTORYSERVER_HEAPSIZE	1536
HADOOP_NAMENODE_HEAPSIZE	12288
HADOOP_DATANODE_HEAPSIZE	384

m3.xlarge

Parameter	Value
YARN_RESOURCEMANAGER_HEAPSIZE	2396
YARN_PROXYSERVER_HEAPSIZE	2396
YARN_NODEMANAGER_HEAPSIZE	2048
HADOOP_JOB_HISTORYSERVER_HEAPSIZE	2396
HADOOP_NAMENODE_HEAPSIZE	1740
HADOOP_DATANODE_HEAPSIZE	757

m3.2xlarge

Parameter	Value
YARN_RESOURCEMANAGER_HEAPSIZE	2703
YARN_PROXYSERVER_HEAPSIZE	2703
YARN_NODEMANAGER_HEAPSIZE	2048
HADOOP_JOB_HISTORYSERVER_HEAPSIZE	2703
HADOOP_NAMENODE_HEAPSIZE	3276
HADOOP_DATANODE_HEAPSIZE	1064

c1.medium

Parameter	Value
YARN_RESOURCEMANAGER_HEAPSIZE	192
YARN_PROXYSERVER_HEAPSIZE	96
YARN_NODEMANAGER_HEAPSIZE	128
HADOOP_JOB_HISTORYSERVER_HEAPSIZE	128
HADOOP_NAMENODE_HEAPSIZE	192
HADOOP_DATANODE_HEAPSIZE	96

c1.xlarge

Parameter	Value
YARN_RESOURCEMANAGER_HEAPSIZE	768
YARN_PROXYSERVER_HEAPSIZE	384
YARN_NODEMANAGER_HEAPSIZE	512
HADOOP_JOB_HISTORYSERVER_HEAPSIZE	512
HADOOP_NAMENODE_HEAPSIZE	768
HADOOP_DATANODE_HEAPSIZE	384

c3.xlarge

Parameter	Value
YARN_RESOURCEMANAGER_HEAPSIZE	2124
YARN_PROXYSERVER_HEAPSIZE	2124
YARN_NODEMANAGER_HEAPSIZE	2124
HADOOP_JOB_HISTORYSERVER_HEAPSIZE	2124
HADOOP_NAMENODE_HEAPSIZE	972
HADOOP_DATANODE_HEAPSIZE	588

c3.2xlarge

Parameter	Value
YARN_RESOURCEMANAGER_HEAPSIZE	2396
YARN_PROXYSERVER_HEAPSIZE	2396
YARN_NODEMANAGER_HEAPSIZE	2048
HADOOP_JOB_HISTORYSERVER_HEAPSIZE	2396
HADOOP_NAMENODE_HEAPSIZE	1740
HADOOP_DATANODE_HEAPSIZE	757

c3.4xlarge

Parameter	Value
YARN_RESOURCEMANAGER_HEAPSIZE	2703
YARN_PROXYSERVER_HEAPSIZE	2703
YARN_NODEMANAGER_HEAPSIZE	2048
HADOOP_JOB_HISTORYSERVER_HEAPSIZE	2703
HADOOP_NAMENODE_HEAPSIZE	3276
HADOOP_DATANODE_HEAPSIZE	1064

c3.8xlarge

Parameter	Value
YARN_RESOURCEMANAGER_HEAPSIZE	3317
YARN_PROXYSERVER_HEAPSIZE	3317
YARN_NODEMANAGER_HEAPSIZE	2048
HADOOP_JOB_HISTORYSERVER_HEAPSIZE	3317
HADOOP_NAMENODE_HEAPSIZE	6348
HADOOP_DATANODE_HEAPSIZE	1679

cc2.8xlarge

Parameter	Value
YARN_RESOURCEMANAGER_HEAPSIZE	2048
YARN_PROXYSERVER_HEAPSIZE	1024
YARN_NODEMANAGER_HEAPSIZE	1536
HADOOP_JOB_HISTORYSERVER_HEAPSIZE	1536
HADOOP_NAMENODE_HEAPSIZE	12288
HADOOP_DATANODE_HEAPSIZE	384

cg1.4xlarge

Parameter	Value
YARN_RESOURCEMANAGER_HEAPSIZE	2048
YARN_PROXYSERVER_HEAPSIZE	1024
YARN_NODEMANAGER_HEAPSIZE	1536
HADOOP_JOB_HISTORYSERVER_HEAPSIZE	1536
HADOOP_NAMENODE_HEAPSIZE	3840
HADOOP_DATANODE_HEAPSIZE	384

cr1.8xlarge

Parameter	Value
YARN_RESOURCEMANAGER_HEAPSIZE	7086
YARN_PROXYSERVER_HEAPSIZE	7086
YARN_NODEMANAGER_HEAPSIZE	2048
HADOOP_JOB_HISTORYSERVER_HEAPSIZE	7086
HADOOP_NAMENODE_HEAPSIZE	25190
HADOOP_DATANODE_HEAPSIZE	4096

d2.xlarge

Parameter	Value
YARN_RESOURCEMANAGER_HEAPSIZE	2713
YARN_PROXYSERVER_HEAPSIZE	2713
YARN_NODEMANAGER_HEAPSIZE	2048
HADOOP_JOB_HISTORYSERVER_HEAPSIZE	2713
HADOOP_NAMENODE_HEAPSIZE	3328
HADOOP_DATANODE_HEAPSIZE	1075

d2.2xlarge

Parameter	Value
YARN_RESOURCEMANAGER_HEAPSIZE	3338
YARN_PROXYSERVER_HEAPSIZE	3338
YARN_NODEMANAGER_HEAPSIZE	2048
HADOOP_JOB_HISTORYSERVER_HEAPSIZE	3338
HADOOP_NAMENODE_HEAPSIZE	6451
HADOOP_DATANODE_HEAPSIZE	1699

d2.4xlarge

Parameter	Value
YARN_RESOURCEMANAGER_HEAPSIZE	4587
YARN_PROXYSERVER_HEAPSIZE	4587
YARN_NODEMANAGER_HEAPSIZE	2048
HADOOP_JOB_HISTORYSERVER_HEAPSIZE	4587
HADOOP_NAMENODE_HEAPSIZE	12697
HADOOP_DATANODE_HEAPSIZE	2949

d2.8xlarge

Parameter	Value
YARN_RESOURCEMANAGER_HEAPSIZE	7089
YARN_PROXYSERVER_HEAPSIZE	7086
YARN_NODEMANAGER_HEAPSIZE	2048
HADOOP_JOB_HISTORYSERVER_HEAPSIZE	7086
HADOOP_NAMENODE_HEAPSIZE	25190
HADOOP_DATANODE_HEAPSIZE	4096

hi1.4xlarge

Parameter	Value
YARN_RESOURCEMANAGER_HEAPSIZE	3328
YARN_PROXYSERVER_HEAPSIZE	3328
YARN_NODEMANAGER_HEAPSIZE	2048
HADOOP_JOB_HISTORYSERVER_HEAPSIZE	3328
HADOOP_NAMENODE_HEAPSIZE	6400
HADOOP_DATANODE_HEAPSIZE	1689

hs1.8xlarge

Parameter	Value
YARN_RESOURCEMANAGER_HEAPSIZE	2048
YARN_PROXYSERVER_HEAPSIZE	1024
YARN_NODEMANAGER_HEAPSIZE	1536
HADOOP_JOB_HISTORYSERVER_HEAPSIZE	1536
HADOOP_NAMENODE_HEAPSIZE	12288
HADOOP_DATANODE_HEAPSIZE	384

i2.xlarge

Parameter	Value
YARN_RESOURCEMANAGER_HEAPSIZE	2713
YARN_PROXYSERVER_HEAPSIZE	2713
YARN_NODEMANAGER_HEAPSIZE	2048
HADOOP_JOB_HISTORYSERVER_HEAPSIZE	2713
HADOOP_NAMENODE_HEAPSIZE	3328
HADOOP_DATANODE_HEAPSIZE	1075

i2.2xlarge

Parameter	Value
YARN_RESOURCEMANAGER_HEAPSIZE	3338
YARN_PROXYSERVER_HEAPSIZE	3338
YARN_NODEMANAGER_HEAPSIZE	2048
HADOOP_JOB_HISTORYSERVER_HEAPSIZE	3338
HADOOP_NAMENODE_HEAPSIZE	6451
HADOOP_DATANODE_HEAPSIZE	1699

i2.4xlarge

Parameter	Value
YARN_RESOURCEMANAGER_HEAPSIZE	4587
YARN_PROXYSERVER_HEAPSIZE	4587
YARN_NODEMANAGER_HEAPSIZE	2048
HADOOP_JOB_HISTORYSERVER_HEAPSIZE	4587
HADOOP_NAMENODE_HEAPSIZE	12697
HADOOP_DATANODE_HEAPSIZE	2949

i2.8xlarge

Parameter	Value
YARN_RESOURCEMANAGER_HEAPSIZE	7086
YARN_PROXYSERVER_HEAPSIZE	7086
YARN_NODEMANAGER_HEAPSIZE	2048
HADOOP_JOB_HISTORYSERVER_HEAPSIZE	7086
HADOOP_NAMENODE_HEAPSIZE	25190
HADOOP_DATANODE_HEAPSIZE	4096

g2.2xlarge

Parameter	Value
YARN_RESOURCEMANAGER_HEAPSIZE	1536
YARN_PROXYSERVER_HEAPSIZE	1024
YARN_NODEMANAGER_HEAPSIZE	1024
HADOOP_JOB_HISTORYSERVER_HEAPSIZE	1024
HADOOP_NAMENODE_HEAPSIZE	2304
HADOOP_DATANODE_HEAPSIZE	384

r3.xlarge

Parameter	Value
YARN_RESOURCEMANAGER_HEAPSIZE	2713
YARN_PROXYSERVER_HEAPSIZE	2713
YARN_NODEMANAGER_HEAPSIZE	2048
HADOOP_JOB_HISTORYSERVER_HEAPSIZE	2713
HADOOP_NAMENODE_HEAPSIZE	3328
HADOOP_DATANODE_HEAPSIZE	1075

r3.2xlarge

Parameter	Value
YARN_RESOURCEMANAGER_HEAPSIZE	3338
YARN_PROXYSERVER_HEAPSIZE	3338
YARN_NODEMANAGER_HEAPSIZE	2048
HADOOP_JOB_HISTORYSERVER_HEAPSIZE	3338
HADOOP_NAMENODE_HEAPSIZE	6451
HADOOP_DATANODE_HEAPSIZE	1699

r3.4xlarge

Parameter	Value
YARN_RESOURCEMANAGER_HEAPSIZE	4587
YARN_PROXYSERVER_HEAPSIZE	4587
YARN_NODEMANAGER_HEAPSIZE	2048
HADOOP_JOB_HISTORYSERVER_HEAPSIZE	4587
HADOOP_NAMENODE_HEAPSIZE	12697
HADOOP_DATANODE_HEAPSIZE	2949

r3.8xlarge

Parameter	Value
YARN_RESOURCEMANAGER_HEAPSIZE	7086
YARN_PROXYSERVER_HEAPSIZE	7086
YARN_NODEMANAGER_HEAPSIZE	2048
HADOOP_JOB_HISTORYSERVER_HEAPSIZE	7086
HADOOP_NAMENODE_HEAPSIZE	25190
HADOOP_DATANODE_HEAPSIZE	4096

HDFS Configuration (Hadoop 2.2.0)

The following table describes the default Hadoop Distributed File System (HDFS) parameters and their settings.

Parameter	Definition	Default Value
dfs.block.size	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 <code>NameNode</code> .	134217728 (128 MB)
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 <code>configure-hadoop</code> bootstrap action.	1 for clusters < four nodes 2 for clusters < ten nodes 3 for all other clusters

Task Configuration (Hadoop 2.2.0)

Topics

- [Task JVM Memory Settings \(AMI 3.0.0\) \(p. 538\)](#)
- [Avoiding Cluster Slowdowns \(AMI 3.0.0\) \(p. 550\)](#)

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.

Task JVM Memory Settings (AMI 3.0.0)

Hadoop 2.2.0 uses two parameters to configure memory for map and reduce: `mapreduce.map.java.opts` and `mapreduce.reduce.java.opts`, respectively. These replace the single configuration option from previous Hadoop versions: `mapreduce.map.java.opts`.

The defaults for these settings per instance type are shown in the following tables.

m1.medium

Configuration Option	Default Value
mapreduce.map.java.opts	-Xmx512m
mapreduce.reduce.java.opts	-Xmx768m
mapreduce.map.memory.mb	768
mapreduce.reduce.memory.mb	1024
yarn.app.mapreduce.am.resource.mb	1024
yarn.scheduler.minimum-allocation-mb	256
yarn.scheduler.maximum-allocation-mb	2048
yarn.nodemanager.resource.memory-mb	2048

m1.large

Configuration Option	Default Value
mapreduce.map.java.opts	-Xmx512m
mapreduce.reduce.java.opts	-Xmx1024m
mapreduce.map.memory.mb	768
mapreduce.reduce.memory.mb	1536
yarn.app.mapreduce.am.resource.mb	1536
yarn.scheduler.minimum-allocation-mb	256
yarn.scheduler.maximum-allocation-mb	3072
yarn.nodemanager.resource.memory-mb	5120

m1.xlarge

Configuration Option	Default Value
mapreduce.map.java.opts	-Xmx512m
mapreduce.reduce.java.opts	-Xmx1536m
mapreduce.map.memory.mb	768
mapreduce.reduce.memory.mb	2048
yarn.app.mapreduce.am.resource.mb	2048
yarn.scheduler.minimum-allocation-mb	256
yarn.scheduler.maximum-allocation-mb	8192
yarn.nodemanager.resource.memory-mb	12288

m2.xlarge

Configuration Option	Default Value
mapreduce.map.java.opts	-Xmx864m
mapreduce.reduce.java.opts	-Xmx1536m
mapreduce.map.memory.mb	1024
mapreduce.reduce.memory.mb	2048
yarn.app.mapreduce.am.resource.mb	2048
yarn.scheduler.minimum-allocation-mb	256
yarn.scheduler.maximum-allocation-mb	7168
yarn.nodemanager.resource.memory-mb	14336

m2.2xlarge

Configuration Option	Default Value
mapreduce.map.java.opts	-Xmx1280m
mapreduce.reduce.java.opts	-Xmx2304m
mapreduce.map.memory.mb	1536
mapreduce.reduce.memory.mb	2560
yarn.app.mapreduce.am.resource.mb	2560
yarn.scheduler.minimum-allocation-mb	256
yarn.scheduler.maximum-allocation-mb	8192
yarn.nodemanager.resource.memory-mb	30720

m3.xlarge

Configuration Option	Default Value
mapreduce.map.java.opts	-Xmx1152m
mapreduce.reduce.java.opts	-Xmx2304m
mapreduce.map.memory.mb	1440
mapreduce.reduce.memory.mb	2880
yarn.scheduler.minimum-allocation-mb	1440
yarn.scheduler.maximum-allocation-mb	11520
yarn.nodemanager.resource.memory-mb	11520

m3.2xlarge

Configuration Option	Default Value
mapreduce.map.java.opts	-Xmx1152m
mapreduce.reduce.java.opts	-Xmx2304m
mapreduce.map.memory.mb	1440
mapreduce.reduce.memory.mb	2880
yarn.scheduler.minimum-allocation-mb	1440
yarn.scheduler.maximum-allocation-mb	23040
yarn.nodemanager.resource.memory-mb	23040

m2.4xlarge

Configuration Option	Default Value
mapreduce.map.java.opts	-Xmx1280m
mapreduce.reduce.java.opts	-Xmx2304m
mapreduce.map.memory.mb	1536
mapreduce.reduce.memory.mb	2560
yarn.scheduler.minimum-allocation-mb	256
yarn.scheduler.maximum-allocation-mb	8192
yarn.nodemanager.resource.memory-mb	61440

c1.medium

Configuration Option	Default Value
io.sort.mb	100
mapreduce.map.java.opts	-Xmx288m
mapreduce.reduce.java.opts	-Xmx288m
mapreduce.map.memory.mb	512
mapreduce.reduce.memory.mb	512
yarn.scheduler.minimum-allocation-mb	256
yarn.scheduler.maximum-allocation-mb	512
yarn.nodemanager.resource.memory-mb	1024

c1.xlarge

Configuration Option	Default Value
io.sort.mb	150
mapreduce.map.java.opts	-Xmx864m
mapreduce.reduce.java.opts	-Xmx1536m
mapreduce.map.memory.mb	1024
mapreduce.reduce.memory.mb	2048
yarn.scheduler.minimum-allocation-mb	256
yarn.scheduler.maximum-allocation-mb	2048
yarn.nodemanager.resource.memory-mb	5120

c3.xlarge

Configuration Option	Default Value
mapreduce.map.java.opts	-Xmx1126m
mapreduce.reduce.java.opts	-Xmx2252m
mapreduce.map.memory.mb	1408
mapreduce.reduce.memory.mb	2816
yarn.scheduler.minimum-allocation-mb	1408
yarn.scheduler.maximum-allocation-mb	5632
yarn.nodemanager.resource.memory-mb	5632

c3.2xlarge

Configuration Option	Default Value
mapreduce.map.java.opts	-Xmx1152m
mapreduce.reduce.java.opts	-Xmx2304m
mapreduce.map.memory.mb	1440
mapreduce.reduce.memory.mb	2880
yarn.scheduler.minimum-allocation-mb	1440
yarn.scheduler.maximum-allocation-mb	11520
yarn.nodemanager.resource.memory-mb	11520

c3.4xlarge

Configuration Option	Default Value
mapreduce.map.java.opts	-Xmx1152m
mapreduce.reduce.java.opts	-Xmx2304m
mapreduce.map.memory.mb	1440
mapreduce.reduce.memory.mb	2880
yarn.scheduler.minimum-allocation-mb	1440
yarn.scheduler.maximum-allocation-mb	23040
yarn.nodemanager.resource.memory-mb	23040

c3.8xlarge

Configuration Option	Default Value
mapreduce.map.java.opts	-Xmx1331m
mapreduce.reduce.java.opts	-Xmx2662m
mapreduce.map.memory.mb	1664
mapreduce.reduce.memory.mb	3328
yarn.scheduler.minimum-allocation-mb	1664
yarn.scheduler.maximum-allocation-mb	53248
yarn.nodemanager.resource.memory-mb	53248

cg1.4xlarge

Configuration Option	Default Value
mapreduce.map.java.opts	-Xmx1280m
mapreduce.reduce.java.opts	-Xmx2304m
mapreduce.map.memory.mb	1536
mapreduce.reduce.memory.mb	2560
yarn.app.mapreduce.am.resource.mb	2560
yarn.scheduler.minimum-allocation-mb	256
yarn.scheduler.maximum-allocation-mb	5120
yarn.nodemanager.resource.memory-mb	20480

cc2.8xlarge

Configuration Option	Default Value
mapreduce.map.java.opts	-Xmx1280m
mapreduce.reduce.java.opts	-Xmx2304m
mapreduce.map.memory.mb	1536
mapreduce.reduce.memory.mb	2560
yarn.app.mapreduce.am.resource.mb	2560
yarn.scheduler.minimum-allocation-mb	256
yarn.scheduler.maximum-allocation-mb	8192
yarn.nodemanager.resource.memory-mb	56320

cr1.8xlarge

Configuration Option	Default Value
mapreduce.map.java.opts	-Xmx6042m
mapreduce.reduce.java.opts	-Xmx12084m
mapreduce.map.memory.mb	7552
mapreduce.reduce.memory.mb	15104
yarn.scheduler.minimum-allocation-mb	7552
yarn.scheduler.maximum-allocation-mb	241664
yarn.nodemanager.resource.memory-mb	241664

d2.xlarge

Configuration Option	Default Value
mapreduce.map.java.opts	-Xmx2342m
mapreduce.reduce.java.opts	-Xmx4684m
mapreduce.map.memory.mb	2928
mapreduce.reduce.memory.mb	5856
yarn.scheduler.minimum-allocation-mb	256
yarn.scheduler.maximum-allocation-mb	23424
yarn.nodemanager.resource.memory-mb	23424

d2.2xlarge

Configuration Option	Default Value
mapreduce.map.java.opts	-Xmx2714m
mapreduce.reduce.java.opts	-Xmx5428m
mapreduce.map.memory.mb	3392
mapreduce.reduce.memory.mb	6784
yarn.scheduler.minimum-allocation-mb	256
yarn.scheduler.maximum-allocation-mb	54272
yarn.nodemanager.resource.memory-mb	54272

d2.4xlarge

Configuration Option	Default Value
mapreduce.map.java.opts	-Xmx2918m
mapreduce.reduce.java.opts	-Xmx5836m
mapreduce.map.memory.mb	3648
mapreduce.reduce.memory.mb	7296
yarn.scheduler.minimum-allocation-mb	256
yarn.scheduler.maximum-allocation-mb	116736
yarn.nodemanager.resource.memory-mb	116736

d2.8xlarge

Configuration Option	Default Value
mapreduce.map.java.opts	-Xmx2685m
mapreduce.reduce.java.opts	-Xmx5370m
mapreduce.map.memory.mb	3356
mapreduce.reduce.memory.mb	6712
yarn.scheduler.minimum-allocation-mb	256
yarn.scheduler.maximum-allocation-mb	241664
yarn.nodemanager.resource.memory-mb	241664

g2.2xlarge

Configuration Option	Default Value
mapreduce.map.java.opts	-Xmx512m

Configuration Option	Default Value
mapreduce.reduce.java.opts	-Xmx1536m
mapreduce.map.memory.mb	768
mapreduce.reduce.memory.mb	2048
yarn.app.mapreduce.am.resource.mb	2048
yarn.scheduler.minimum-allocation-mb	256
yarn.scheduler.maximum-allocation-mb	8192
yarn.nodemanager.resource.memory-mb	12288

hi1.4xlarge

Configuration Option	Default Value
mapreduce.map.java.opts	-Xmx2688m
mapreduce.reduce.java.opts	-Xmx5376m
mapreduce.map.memory.mb	3360
mapreduce.reduce.memory.mb	6720
yarn.scheduler.minimum-allocation-mb	3360
yarn.scheduler.maximum-allocation-mb	53760
yarn.nodemanager.resource.memory-mb	53760

hs1.8xlarge

Configuration Option	Default Value
mapreduce.map.java.opts	-Xmx1280m
mapreduce.reduce.java.opts	-Xmx2304m
mapreduce.map.memory.mb	1536
mapreduce.reduce.memory.mb	2560
yarn.app.mapreduce.am.resource.mb	2560
yarn.scheduler.minimum-allocation-mb	256
yarn.scheduler.maximum-allocation-mb	8192
yarn.nodemanager.resource.memory-mb	56320

i2.xlarge

Configuration Option	Default Value
mapreduce.map.java.opts	-Xmx2342m

Configuration Option	Default Value
mapreduce.reduce.java.opts	-Xmx4684m
mapreduce.map.memory.mb	2928
mapreduce.reduce.memory.mb	5856
yarn.scheduler.minimum-allocation-mb	2928
yarn.scheduler.maximum-allocation-mb	23424
yarn.nodemanager.resource.memory-mb	23424

i2.2xlarge

Configuration Option	Default Value
mapreduce.map.java.opts	-Xmx2714m
mapreduce.reduce.java.opts	-Xmx5428m
mapreduce.map.memory.mb	3392
mapreduce.reduce.memory.mb	6784
yarn.scheduler.minimum-allocation-mb	3392
yarn.scheduler.maximum-allocation-mb	54272
yarn.nodemanager.resource.memory-mb	54272

i2.4xlarge

Configuration Option	Default Value
mapreduce.map.java.opts	-Xmx2918m
mapreduce.reduce.java.opts	-Xmx5836m
mapreduce.map.memory.mb	3648
mapreduce.reduce.memory.mb	7296
yarn.scheduler.minimum-allocation-mb	3648
yarn.scheduler.maximum-allocation-mb	116736
yarn.nodemanager.resource.memory-mb	116736

i2.8xlarge

Configuration Option	Default Value
mapreduce.map.java.opts	-Xmx3021m
mapreduce.reduce.java.opts	-Xmx6042m
mapreduce.map.memory.mb	15974

Configuration Option	Default Value
mapreduce.reduce.memory.mb	16220
yarn.scheduler.minimum-allocation-mb	3776
yarn.scheduler.maximum-allocation-mb	241664
yarn.nodemanager.resource.memory-mb	241664

r3.xlarge

Configuration Option	Default Value
mapreduce.map.java.opts	-Xmx2342m
mapreduce.reduce.java.opts	-Xmx4684m
mapreduce.map.memory.mb	2982
mapreduce.reduce.memory.mb	5856
yarn.scheduler.minimum-allocation-mb	2928
yarn.scheduler.maximum-allocation-mb	23424
yarn.nodemanager.resource.memory-mb	23424

r3.2xlarge

Configuration Option	Default Value
mapreduce.map.java.opts	-Xmx2714m
mapreduce.reduce.java.opts	-Xmx5428m
mapreduce.map.memory.mb	3392
mapreduce.reduce.memory.mb	6784
yarn.scheduler.minimum-allocation-mb	3392
yarn.scheduler.maximum-allocation-mb	54272
yarn.nodemanager.resource.memory-mb	54272

r3.4xlarge

Configuration Option	Default Value
mapreduce.map.java.opts	-Xmx5837m
mapreduce.reduce.java.opts	-Xmx11674m
mapreduce.map.memory.mb	7296
mapreduce.reduce.memory.mb	14592
yarn.scheduler.minimum-allocation-mb	7296

Configuration Option	Default Value
yarn.scheduler.maximum-allocation-mb	116736
yarn.nodemanager.resource.memory-mb	116736

r3.8xlarge

Configuration Option	Default Value
mapreduce.map.java.opts	-Xmx6042m
mapreduce.reduce.java.opts	-Xmx12084m
mapreduce.map.memory.mb	7552
mapreduce.reduce.memory.mb	15104
yarn.scheduler.minimum-allocation-mb	7552
yarn.scheduler.maximum-allocation-mb	241664
yarn.nodemanager.resource.memory-mb	241664

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 that all memory is freed for subsequent tasks.

Use the `mapred.job.reuse.jvm.num.tasks` option to configure the JVM reuse settings.

To modify JVM settings via a bootstrap action using the AWS CLI

To modify JVM settings using the AWS CLI, type the `--bootstrap-action` parameter and specify the settings in the arguments list.

- To configure infinite JVM reuse, 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 \
--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="Configuring infinite JVM reuse",Args=["-m","mapred.job.reuse.jvm.num.tasks=-1"]
```

- 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-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="Configuring infinite JVM reuse",Args=[ "-m", "mapred.job.reuse.jvm.num.tasks=-1" ]
```

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.

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.

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

Avoiding Cluster Slowdowns (AMI 3.0.0)

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. 124).

Speculative Execution Parameters

Parameter	Default Setting
<code>mapred.map.tasks.speculative.execution</code>	true
<code>mapred.reduce.tasks.speculative.execution</code>	true

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.

- Type the following command to disable reducer speculative execution 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 \  
--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:

```
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>.

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 \(p. 124\)](#).

The following table presents the default values for the parameters that affect intermediate compression.

Parameter	Value
<code>mapred.compress.map.output</code>	true
<code>mapred.map.output.compression.codec</code>	org.apache.hadoop.io.compress.SnappyCodec

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:

```
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 compression",Args=["-m","mapred.compress.map.output=false"]
```

- 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-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 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:

```
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="Change compression codec",Args=["-m","mapred.map.output.compression.codec=org.apache.hadoop.io.compress.GzipCodec"]
```

- 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-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="Change compression co
```

```
dec", 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>.

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 \(AMI\) \(p. 48\)](#).

Topics

- [Hadoop Configuration \(Hadoop 1.0.3\) \(p. 553\)](#)
- [HDFS Configuration \(Hadoop 1.0.3\) \(p. 565\)](#)
- [Task Configuration \(Hadoop 1.0.3\) \(p. 565\)](#)
- [Intermediate Compression \(Hadoop 1.0.3\) \(p. 569\)](#)

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 \(AMI\) \(p. 48\)](#).

m1.small

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	576
HADOOP_NAMENODE_HEAPSIZE	192
HADOOP_TASKTRACKER_HEAPSIZE	192
HADOOP_DATANODE_HEAPSIZE	96
mapred.child.java.opts	-Xmx288m
mapred.tasktracker.map.tasks.maximum	2

Parameter	Value
mapred.tasktracker.reduce.tasks.maximum	1

m1.medium

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	1152
HADOOP_NAMENODE_HEAPSIZE	384
HADOOP_TASKTRACKER_HEAPSIZE	192
HADOOP_DATANODE_HEAPSIZE	192
mapred.child.java.opts	-Xmx576m
mapred.tasktracker.map.tasks.maximum	2
mapred.tasktracker.reduce.tasks.maximum	1

m1.large

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	2304
HADOOP_NAMENODE_HEAPSIZE	768
HADOOP_TASKTRACKER_HEAPSIZE	384
HADOOP_DATANODE_HEAPSIZE	384
mapred.child.java.opts	-Xmx864m
mapred.tasktracker.map.tasks.maximum	3
mapred.tasktracker.reduce.tasks.maximum	1

m1.xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	6912
HADOOP_NAMENODE_HEAPSIZE	2304
HADOOP_TASKTRACKER_HEAPSIZE	384
HADOOP_DATANODE_HEAPSIZE	384
mapred.child.java.opts	-Xmx768m
mapred.tasktracker.map.tasks.maximum	8

Parameter	Value
mapred.tasktracker.reduce.tasks.maximum	3

m2.xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	9216
HADOOP_NAMENODE_HEAPSIZE	3072
HADOOP_TASKTRACKER_HEAPSIZE	384
HADOOP_DATANODE_HEAPSIZE	384
mapred.child.java.opts	-Xmx2304m
mapred.tasktracker.map.tasks.maximum	3
mapred.tasktracker.reduce.tasks.maximum	1

m2.2xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	18432
HADOOP_NAMENODE_HEAPSIZE	6144
HADOOP_TASKTRACKER_HEAPSIZE	384
HADOOP_DATANODE_HEAPSIZE	384
mapred.child.java.opts	-Xmx2688m
mapred.tasktracker.map.tasks.maximum	6
mapred.tasktracker.reduce.tasks.maximum	2

m2.4xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	36864
HADOOP_NAMENODE_HEAPSIZE	12288
HADOOP_TASKTRACKER_HEAPSIZE	384
HADOOP_DATANODE_HEAPSIZE	384
mapred.child.java.opts	-Xmx2304m
mapred.tasktracker.map.tasks.maximum	14

Parameter	Value
mapred.tasktracker.reduce.tasks.maximum	4

m3.xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	3686
HADOOP_NAMENODE_HEAPSIZE	1740
HADOOP_TASKTRACKER_HEAPSIZE	686
HADOOP_DATANODE_HEAPSIZE	757
mapred.child.java.opts	-Xmx1440m
mapred.tasktracker.map.tasks.maximum	6
mapred.tasktracker.reduce.tasks.maximum	2

m3.2xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	6758
HADOOP_NAMENODE_HEAPSIZE	3276
HADOOP_TASKTRACKER_HEAPSIZE	839
HADOOP_DATANODE_HEAPSIZE	1064
mapred.child.java.opts	-Xmx1440m
mapred.tasktracker.map.tasks.maximum	12
mapred.tasktracker.reduce.tasks.maximum	4

c1.medium

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	576
HADOOP_NAMENODE_HEAPSIZE	192
HADOOP_TASKTRACKER_HEAPSIZE	192
HADOOP_DATANODE_HEAPSIZE	96
mapred.child.java.opts	-Xmx288m
mapred.tasktracker.map.tasks.maximum	2

Parameter	Value
mapred.tasktracker.reduce.tasks.maximum	1

c1.xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	2304
HADOOP_NAMENODE_HEAPSIZE	768
HADOOP_TASKTRACKER_HEAPSIZE	384
HADOOP_DATANODE_HEAPSIZE	384
mapred.child.java.opts	-Xmx384m
mapred.tasktracker.map.tasks.maximum	7
mapred.tasktracker.reduce.tasks.maximum	2

c3.xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	2124
HADOOP_NAMENODE_HEAPSIZE	972
HADOOP_TASKTRACKER_HEAPSIZE	588
HADOOP_DATANODE_HEAPSIZE	588
mapred.child.java.opts	-Xmx1408m
mapred.tasktracker.map.tasks.maximum	3
mapred.tasktracker.reduce.tasks.maximum	1

c3.2xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	3686
HADOOP_NAMENODE_HEAPSIZE	1740
HADOOP_TASKTRACKER_HEAPSIZE	686
HADOOP_DATANODE_HEAPSIZE	757
mapred.child.java.opts	-Xmx1440m
mapred.tasktracker.map.tasks.maximum	6

Parameter	Value
mapred.tasktracker.reduce.tasks.maximum	2

c3.4xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	6758
HADOOP_NAMENODE_HEAPSIZE	3276
HADOOP_TASKTRACKER_HEAPSIZE	839
HADOOP_DATANODE_HEAPSIZE	1064
mapred.child.java.opts	-Xmx1440m
mapred.tasktracker.map.tasks.maximum	12
mapred.tasktracker.reduce.tasks.maximum	4

c3.8xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	12902
HADOOP_NAMENODE_HEAPSIZE	6348
HADOOP_TASKTRACKER_HEAPSIZE	1146
HADOOP_DATANODE_HEAPSIZE	1679
mapred.child.java.opts	-Xmx1664m
mapred.tasktracker.map.tasks.maximum	24
mapred.tasktracker.reduce.tasks.maximum	8

cc2.8xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	30114
HADOOP_NAMENODE_HEAPSIZE	12288
HADOOP_TASKTRACKER_HEAPSIZE	384
HADOOP_DATANODE_HEAPSIZE	384
mapred.child.java.opts	-Xmx1536m
mapred.tasktracker.map.tasks.maximum	24

Parameter	Value
mapred.tasktracker.reduce.tasks.maximum	6

cg1.4xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	7680
HADOOP_NAMENODE_HEAPSIZE	3840
HADOOP_TASKTRACKER_HEAPSIZE	384
HADOOP_DATANODE_HEAPSIZE	384
mapred.child.java.opts	-Xmx864m
mapred.tasktracker.map.tasks.maximum	12
mapred.tasktracker.reduce.tasks.maximum	3

cr1.8xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	50585
HADOOP_NAMENODE_HEAPSIZE	25190
HADOOP_TASKTRACKER_HEAPSIZE	2048
HADOOP_DATANODE_HEAPSIZE	4096
mapred.child.java.opts	-Xmx7552m
mapred.tasktracker.map.tasks.maximum	24
mapred.tasktracker.reduce.tasks.maximum	8

hi1.4xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	30114
HADOOP_NAMENODE_HEAPSIZE	12288
HADOOP_TASKTRACKER_HEAPSIZE	384
HADOOP_DATANODE_HEAPSIZE	384
mapred.child.java.opts	-Xmx1536m
mapred.tasktracker.map.tasks.maximum	24

Parameter	Value
mapred.tasktracker.reduce.tasks.maximum	6

hs1.8xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	30114
HADOOP_NAMENODE_HEAPSIZE	12288
HADOOP_TASKTRACKER_HEAPSIZE	384
HADOOP_DATANODE_HEAPSIZE	384
mapred.child.java.opts	-Xmx1536m
mapred.tasktracker.map.tasks.maximum	24
mapred.tasktracker.reduce.tasks.maximum	6

cg1.4xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	7680
HADOOP_NAMENODE_HEAPSIZE	3840
HADOOP_TASKTRACKER_HEAPSIZE	384
HADOOP_DATANODE_HEAPSIZE	384
mapred.child.java.opts	-Xmx864m
mapred.tasktracker.map.tasks.maximum	12
mapred.tasktracker.reduce.tasks.maximum	3

d2.xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	6860
HADOOP_NAMENODE_HEAPSIZE	3328
HADOOP_TASKTRACKER_HEAPSIZE	844
HADOOP_DATANODE_HEAPSIZE	1075
mapred.child.java.opts	-Xmx2928m
mapred.tasktracker.map.tasks.maximum	6

Parameter	Value
mapred.tasktracker.reduce.tasks.maximum	2

d2.2xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	13107
HADOOP_NAMENODE_HEAPSIZE	6451
HADOOP_TASKTRACKER_HEAPSIZE	1157
HADOOP_DATANODE_HEAPSIZE	1699
mapred.child.java.opts	-Xmx3392m
mapred.tasktracker.map.tasks.maximum	12
mapred.tasktracker.reduce.tasks.maximum	4

d2.4xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	25600
HADOOP_NAMENODE_HEAPSIZE	12697
HADOOP_TASKTRACKER_HEAPSIZE	1781
HADOOP_DATANODE_HEAPSIZE	2949
mapred.child.java.opts	-Xmx3648m
mapred.tasktracker.map.tasks.maximum	24
mapred.tasktracker.reduce.tasks.maximum	8

d2.8xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	50585
HADOOP_NAMENODE_HEAPSIZE	25190
HADOOP_TASKTRACKER_HEAPSIZE	2048
HADOOP_DATANODE_HEAPSIZE	4096
mapred.child.java.opts	-Xmx3356m
mapred.tasktracker.map.tasks.maximum	54

Parameter	Value
mapred.tasktracker.reduce.tasks.maximum	18

g2.2xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	6912
HADOOP_NAMENODE_HEAPSIZE	2304
HADOOP_TASKTRACKER_HEAPSIZE	384
HADOOP_DATANODE_HEAPSIZE	384
mapred.child.java.opts	-Xmx768m
mapred.tasktracker.map.tasks.maximum	8
mapred.tasktracker.reduce.tasks.maximum	3

i2.xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	6860
HADOOP_NAMENODE_HEAPSIZE	3328
HADOOP_TASKTRACKER_HEAPSIZE	844
HADOOP_DATANODE_HEAPSIZE	1075
mapred.child.java.opts	-Xmx2928m
mapred.tasktracker.map.tasks.maximum	6
mapred.tasktracker.reduce.tasks.maximum	2

i2.2xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	13107
HADOOP_NAMENODE_HEAPSIZE	6451
HADOOP_TASKTRACKER_HEAPSIZE	1157
HADOOP_DATANODE_HEAPSIZE	1699
mapred.child.java.opts	-Xmx3392m
mapred.tasktracker.map.tasks.maximum	12

Parameter	Value
mapred.tasktracker.reduce.tasks.maximum	4

i2.4xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	25600
HADOOP_NAMENODE_HEAPSIZE	12697
HADOOP_TASKTRACKER_HEAPSIZE	1781
HADOOP_DATANODE_HEAPSIZE	2949
mapred.child.java.opts	-Xmx3648m
mapred.tasktracker.map.tasks.maximum	24
mapred.tasktracker.reduce.tasks.maximum	8

i2.8xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	50585
HADOOP_NAMENODE_HEAPSIZE	25190
HADOOP_TASKTRACKER_HEAPSIZE	2048
HADOOP_DATANODE_HEAPSIZE	4096
mapred.child.java.opts	-Xmx3776m
mapred.tasktracker.map.tasks.maximum	48
mapred.tasktracker.reduce.tasks.maximum	16

r3.xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	6860
HADOOP_NAMENODE_HEAPSIZE	3328
HADOOP_TASKTRACKER_HEAPSIZE	844
HADOOP_DATANODE_HEAPSIZE	1075
mapred.child.java.opts	-Xmx2928m
mapred.tasktracker.map.tasks.maximum	6

Parameter	Value
mapred.tasktracker.reduce.tasks.maximum	2

r3.2xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	13107
HADOOP_NAMENODE_HEAPSIZE	6451
HADOOP_TASKTRACKER_HEAPSIZE	1157
HADOOP_DATANODE_HEAPSIZE	1699
mapred.child.java.opts	-Xmx3392m
mapred.tasktracker.map.tasks.maximum	12
mapred.tasktracker.reduce.tasks.maximum	4

r3.4xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	25600
HADOOP_NAMENODE_HEAPSIZE	12697
HADOOP_TASKTRACKER_HEAPSIZE	1781
HADOOP_DATANODE_HEAPSIZE	2949
mapred.child.java.opts	-Xmx7296m
mapred.tasktracker.map.tasks.maximum	12
mapred.tasktracker.reduce.tasks.maximum	4

r3.8xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	50585
HADOOP_NAMENODE_HEAPSIZE	25190
HADOOP_TASKTRACKER_HEAPSIZE	2048
HADOOP_DATANODE_HEAPSIZE	4096
mapred.child.java.opts	-Xmx7552m
mapred.tasktracker.map.tasks.maximum	24

Parameter	Value
mapred.tasktracker.reduce.tasks.maximum	8

HDFS Configuration (Hadoop 1.0.3)

The following table describes the default Hadoop Distributed File System (HDFS) parameters and their settings.

Parameter	Definition	Default Value
dfs.block.size	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 <code>NameNode</code> .	134217728 (128 MB)
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 <code>configure-hadoop</code> bootstrap action.	1 for clusters < four nodes 2 for clusters < ten nodes 3 for all other clusters

Task Configuration (Hadoop 1.0.3)

Topics

- [Tasks per Machine \(p. 565\)](#)
- [Tasks per Job \(AMI 2.3\) \(p. 567\)](#)
- [Task JVM Settings \(AMI 2.3\) \(p. 567\)](#)
- [Avoiding Cluster Slowdowns \(AMI 2.3\) \(p. 568\)](#)

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.

EC2 Instance Name	Mappers	Reducers
m1.small	2	1

Amazon Elastic MapReduce Developer Guide
Task Configuration (Hadoop 1.0.3)

EC2 Instance Name	Mappers	Reducers
m1.medium	2	1
m1.large	3	1
m1.xlarge	8	3
m2.xlarge	3	1
m2.2xlarge	6	2
m2.4xlarge	14	4
m3.xlarge	6	2
m3.2xlarge	12	4
c1.medium	2	1
c1.xlarge	7	2
c3.xlarge	3	1
c3.2xlarge	6	2
c3.4xlarge	12	4
c3.8xlarge	24	8
cc2.8xlarge	24	6
d2.xlarge	6	2
d2.2xlarge	12	4
d2.4xlarge	24	8
d2.8xlarge	54	18
hi1.4xlarge	24	6
hs1.8xlarge	24	6
cg1.4xlarge	12	3
g2.2xlarge	8	3
i2.xlarge	6	2
i2.2xlarge	12	4
i2.4xlarge	24	8
i2.8xlarge	48	16
r3.xlarge	6	2
r3.2xlarge	12	4
r3.4xlarge	12	4
r3.8xlarge	24	8

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.

Parameter	Description
<code>mapred.map.tasks</code>	Target number of map tasks to run. The actual number of tasks created is sometimes different than this number.
<code>mapred.map.tasksperslot</code>	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.
<code>mapred.reduce.tasks</code>	Number of reduce tasks to run.
<code>mapred.reduce.tasksperslot</code>	Number of reduce tasks to run as a ratio of the number of reduce slots in the cluster.

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.

Amazon EC2 Instance Name	Default JVM value
m1.small	-Xmx288m
m1.medium	-Xmx576m
m1.large	-Xmx864m
m1.xlarge	-Xmx768m
c1.medium	-Xmx288m
c1.xlarge	-Xmx384m

Amazon EC2 Instance Name	Default JVM value
m2.xlarge	-Xmx2304m
m2.2xlarge	-Xmx2688m
m2.4xlarge	-Xmx2304m
cc2.8xlarge	-Xmx1536m
hi1.4xlarge	-Xmx2048m
hs1.8xlarge	-Xmx1536m
cg1.4xlarge	-Xmx864m

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 \(p. 124\)](#).

Speculative Execution Parameters

Parameter	Default Setting
<code>mapred.map.tasks.speculative.execution</code>	true
<code>mapred.reduce.tasks.speculative.execution</code>	true

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. 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. 124).

The following table presents the default values for the parameters that affect intermediate compression.

Parameter	Value
mapred.compress.map.output	true
mapred.map.output.compression.codec	org.apache.hadoop.io.compress.SnappyCodec

Hadoop 20.205 Default Configuration (Deprecated)

Topics

- [Hadoop Configuration \(Hadoop 20.205\)](#) (p. 569)
- [HDFS Configuration \(Hadoop 20.205\)](#) (p. 573)
- [Task Configuration \(Hadoop 20.205\)](#) (p. 573)
- [Intermediate Compression \(Hadoop 20.205\)](#) (p. 576)

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. 48).

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. 48).

m1.small

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	768
HADOOP_NAMENODE_HEAPSIZE	256
HADOOP_TASKTRACKER_HEAPSIZE	256

Parameter	Value
HADOOP_DATANODE_HEAPSIZE	128
mapred.child.java.opts	-Xmx384m
mapred.tasktracker.map.tasks.maximum	2
mapred.tasktracker.reduce.tasks.maximum	1

m1.medium

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	1536
HADOOP_NAMENODE_HEAPSIZE	512
HADOOP_TASKTRACKER_HEAPSIZE	256
HADOOP_DATANODE_HEAPSIZE	256
mapred.child.java.opts	-Xmx768m
mapred.tasktracker.map.tasks.maximum	2
mapred.tasktracker.reduce.tasks.maximum	1

m1.large

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	3072
HADOOP_NAMENODE_HEAPSIZE	1024
HADOOP_TASKTRACKER_HEAPSIZE	512
HADOOP_DATANODE_HEAPSIZE	512
mapred.child.java.opts	-Xmx1152m
mapred.tasktracker.map.tasks.maximum	3
mapred.tasktracker.reduce.tasks.maximum	1

m1.xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	9216
HADOOP_NAMENODE_HEAPSIZE	3072
HADOOP_TASKTRACKER_HEAPSIZE	512

Parameter	Value
HADOOP_DATANODE_HEAPSIZE	512
mapred.child.java.opts	-Xmx1024m
mapred.tasktracker.map.tasks.maximum	8
mapred.tasktracker.reduce.tasks.maximum	3

c1.medium

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	768
HADOOP_NAMENODE_HEAPSIZE	256
HADOOP_TASKTRACKER_HEAPSIZE	256
HADOOP_DATANODE_HEAPSIZE	128
mapred.child.java.opts	-Xmx384m
mapred.tasktracker.map.tasks.maximum	2
mapred.tasktracker.reduce.tasks.maximum	1

c1.xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	3072
HADOOP_NAMENODE_HEAPSIZE	1024
HADOOP_TASKTRACKER_HEAPSIZE	512
HADOOP_DATANODE_HEAPSIZE	512
mapred.child.java.opts	-Xmx512m
mapred.tasktracker.map.tasks.maximum	7
mapred.tasktracker.reduce.tasks.maximum	2

m2.xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	12288
HADOOP_NAMENODE_HEAPSIZE	4096
HADOOP_TASKTRACKER_HEAPSIZE	512

Parameter	Value
HADOOP_DATANODE_HEAPSIZE	512
mapred.child.java.opts	-Xmx3072m
mapred.tasktracker.map.tasks.maximum	3
mapred.tasktracker.reduce.tasks.maximum	1

m2.2xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	24576
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

Parameter	Value
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
HADOOP_DATANODE_HEAPSIZE	512
mapred.child.java.opts	-Xmx1152m
mapred.tasktracker.map.tasks.maximum	12
mapred.tasktracker.reduce.tasks.maximum	3

HDFS Configuration (Hadoop 20.205)

The following table describes the default Hadoop Distributed File System (HDFS) parameters and their settings.

Parameter	Definition	Default Value
dfs.block.size	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 <code>NameNode</code> .	134217728 (128 MB)
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 <code>configure-hadoop</code> bootstrap action.	1 for clusters < four nodes 2 for clusters < ten nodes 3 for all other clusters

Task Configuration (Hadoop 20.205)

Topics

- [Tasks per Machine \(p. 574\)](#)
- [Tasks per Job \(AMI 2.0 and 2.1\) \(p. 574\)](#)
- [Task JVM Settings \(AMI 2.0 and 2.1\) \(p. 575\)](#)

- [Avoiding Cluster Slowdowns \(AMI 2.0 and 2.1\) \(p. 576\)](#)

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.

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.

Parameter	Description
mapred.map.tasks	Target number of map tasks to run. The actual number of tasks created is sometimes different than this number.
mapred.map.tasksperslot	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.
mapred.reduce.tasks	Number of reduce tasks to run.
mapred.reduce.tasksperslot	Number of reduce tasks to run as a ratio of the number of reduce slots in the cluster.

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 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.

Amazon EC2 Instance Name	Default JVM value
m1.small	-Xmx384m
m1.medium	-Xmx768m
m1.large	-Xmx1152m
m1.xlarge	-Xmx1024m
c1.medium	-Xmx384m
c1.xlarge	-Xmx512m
m2.xlarge	-Xmx3072m
m2.2xlarge	-Xmx3584m
m2.4xlarge	-Xmx3072m
cc2.8xlarge	-Xmx2048m
cg1.4xlarge	-Xmx1152m

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. 124).

Speculative Execution Parameters

Parameter	Default Setting
<code>mapred.map.tasks.speculative.execution</code>	true
<code>mapred.reduce.tasks.speculative.execution</code>	true

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. 124).

The following table presents the default values for the parameters that affect intermediate compression.

Parameter	Value
<code>mapred.compress.map.output</code>	true
<code>mapred.map.output.compression.codec</code>	<code>org.apache.hadoop.io.compress.SnappyCodec</code>

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/>.

Topics

- [Specifying Parameter Values in AWS CLI for Amazon EMR](#) (p. 577)
- [Install the Amazon EMR Command Line Interface \(Deprecated\)](#) (p. 579)
- [How to Call the Command Line Interface \(Deprecated\)](#) (p. 584)
- [AWS EMR Command Line Interface Options \(Deprecated\)](#) (p. 585)
- [AWS EMR Command Line Interface Releases \(Deprecated\)](#) (p. 644)

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

Parameter	Description
instance_profile	The instance profile you want Amazon EMR to use to run application on the cluster's Amazon EC2 instances.
service_role	The service role you want the Amazon EMR service to use.
log_uri	The Amazon S3 URI you want Amazon EMR to place cluster logs.
key_name	The EC2 key pair name you want to use the access the EMR cluster.
enable_debugging	The Boolean value that indicates if you want to enable debugging when creating a cluster.

ssh, get, put, socks

Parameter	Description
key_pair_file	The path to the private key pair file you use to connect to the EMR cluster.

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:

```
[default]
region = us-east-1
emr =
  service_role = EMR_DefaultRole
  instance_profile = EMR_EC2_DefaultRole
  log_uri = s3://myBucket/logs
  enable_debugging = True
  key_name = myKeyName
  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 \(p. 579\)](#)
- [Verifying the RubyGems package management framework \(p. 580\)](#)
- [Installing the Amazon EMR Command Line Interface \(p. 580\)](#)
- [Configuring Credentials \(p. 581\)](#)
- [SSH Credentials \(p. 583\)](#)

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:
 - Linux and Unix users can download Ruby 1.8.7 from <http://www.ruby-lang.org/en/news/2010/06/23/ruby-1-8-7-p299-released/>, Ruby 1.9.3 from <https://www.ruby-lang.org/en/news/2014/02/24/ruby-1-9-3-p545-is-released/>, and Ruby 2.0 from <https://www.ruby-lang.org/en/news/2014/02/24/ruby-2-0-0-p451-is-released/>.
 - 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:
 - a. Go to <http://aws.amazon.com/developertools/2264>.
 - 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:

```
{
  "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 root account 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 <http://docs.aws.amazon.com/AmazonS3/latest/dev/BucketRestrictions.html>.

4. For **Region**, choose the bucket region.

Amazon EMR region	Amazon S3 region
US East (N. Virginia)	US Standard
US West (Oregon)	Oregon
US West (N. California)	Northern California

Amazon EMR region	Amazon S3 region
EU (Ireland)	Ireland
EU (Frankfurt)	Frankfurt
Asia Pacific (Tokyo)	Japan
Asia Pacific (Singapore)	Singapore
Asia Pacific (Sydney)	Sydney
South America (Sao Paulo)	Sao Paulo
AWS GovCloud (US)	GovCloud

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.

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/>.
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:

```
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>.
 - 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:

```
./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
```

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 \(p. 586\)](#)
- [Uncommon Options \(p. 588\)](#)
- [Options Common to All Step Types \(p. 588\)](#)
- [Adding and Modifying Instance Groups \(p. 588\)](#)
- [Adding JAR Steps to Job Flows \(p. 590\)](#)
- [Adding JSON Steps to Job Flows \(p. 592\)](#)
- [Adding Streaming Steps to Job Flows \(p. 592\)](#)
- [Assigning an Elastic IP Address to the Master Node \(p. 595\)](#)
- [Connecting to the Master Node \(p. 597\)](#)
- [Creating Job Flows \(p. 598\)](#)
- [Using HBase Options \(p. 605\)](#)
- [Using Hive Options \(p. 615\)](#)
- [Using Impala Options \(p. 619\)](#)
- [Listing and Describing Job Flows \(p. 621\)](#)

- [Passing Arguments to Steps](#) (p. 623)
- [Using Pig Options](#) (p. 625)
- [Specifying Step Actions](#) (p. 628)
- [Specifying Bootstrap Actions](#) (p. 629)
- [Tagging](#) (p. 635)
- [Terminating Job Flows](#) (p. 637)
- [Using S3DistCp](#) (p. 640)

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 root account 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 \(Deprecated\) \(p. 579\)](#). 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:

```
./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 c1.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 instances in your cluster while you can increase or decrease the number of task instances. Setting the number of task instances to zero removes all Spot Instances (but not the instance group).

- 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 \  
--modify-instance-group task --instance-count 5
```

- Windows users:

```
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 Amazon EMR CLI, type the following command.
 - Linux, UNIX, and Mac OS X users:

```
./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 3 --arg 0 --arg 1 --arg 240 --arg 48 --arg 24 \  
--arg 24 --arg 128 --arg 16
```

- Windows users:

```
ruby elastic-mapreduce --create --name "Test custom JAR" --jar
s3://elasticmapreduce/samples/cloudburst/cloudburst.jar --arg s3://elast
icmapreduce/samples/cloudburst/input/s_suis.br --arg s3://elasticmapre
duce/samples/cloudburst/input/100k.br --arg s3://mybucket/cloudburst/output
--arg 36 --arg 3 --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,-edgeLoca  
tionReport"
```

- Windows users:

```
ruby elastic-mapreduce --create --name "Test Cascading" --bootstrap-action  
s3://files.cascading.org/sdk/2.1/install-cascading-sdk.sh --JAR elast  
icmapreduce/samples/cloudfront/logprocessor.jar --args "-input,s3://elast  
icmapreduce/samples/cloudfront/input,-start,any,-end,2010-12-27-02 300,-  
output,s3://mybucket/cloudfront/output/2010-12-27-02,-overallVolumeReport,-  
objectPopularityReport,-clientIPReport,-edgeLocationReport"
```

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/multi  
tool/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 m1.large --arg "-files" --arg "s3://elasticmapre  
duce/samples/wordcount/wordSplitter.py" \  
--input s3://elasticmapreduce/samples/wordcount/input --mapper wordSplit  
ter.py --reducer aggregate \  
--output s3://mybucket/output/2014-01-16
```

- Windows users:

```
ruby elastic-mapreduce --create --stream --ami-version 3.3 --instance-type  
m1.large --arg "-files" --arg "s3://elasticmapreduce/samples/word  
Splitter.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://elasticmapre  
duce/samples/wordcount/input --mapper s3://elasticmapreduce/samples/word  
count/wordSplitter.py --reducer aggregate --output s3://mybucket/out  
put/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.

Action	Parameter to add
Add an individual file to the Distributed Cache	<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
Add an archive file to the Distributed Cache	<code>--cache-archive</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

Your cluster copies the files to the cache location before processing any job flow steps.

Example 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:

```
./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:

```
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:

```
./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:

```
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:

```
./elastic-mapreduce --list
```

- Windows users:

```
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 `j-3L7WXXXXXHO4H` with your cluster identifier.

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce -j j-3L7WXXXXXHO4H --ssh
```

- Windows users:

```
ruby elastic-mapreduce -j j-3L7WXXXXXXHO4H --ssh
```

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:

```
./elastic-mapreduce -j j-3L7WXXXXXXHO4H --socks
```

- Windows users:

```
ruby elastic-mapreduce -j j-3L7WXXXXXXHO4H --socks
```

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 `elastic-mapreduce --version` at the command line. You can download the latest version of the CLI from <http://aws.amazon.com/code/Elastic-MapReduce/2264>.

Creating Job Flows

`--alive`

Used with `--create` to launch a cluster that continues running even after completing all its steps. Interactive clusters require this option.

`--ami-version AMI_VERSION`

Used with `--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 `--hadoop-version` parameter is no longer supported.

In the Amazon EMR CLI, if you use the keyword `latest` instead of a version number for the AMI (for example `--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 `--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.

`--name` "*JOB_FLOW_NAME*"
Specifies a name for the cluster. This can only be set when the jobflow is created.

`--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 [Default IAM Roles for Amazon EMR \(p. 186\)](#).

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:

```
./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_De  
faultRole --jobflow-role EMR_EC2_DefaultRole
```

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 --num-instances 3 \  
--instance-type m1.large \  
--name "myJobFlowName" \  
--hive-interactive --hive-versions 0.8.1.6 \  
--ami-version 2.3.0 \  
--jobflow-role EMR_EC2_DefaultRole \  
--service-role EMR_DefaultRole
```

- Windows users:

```
ruby elastic-mapreduce --create --alive --num-instances 3 --instance-type  
m1.small --name "myJobFlowName" --hive-interactive --hive-versions 0.8.1.6  
--ami-version 2.3.0 --jobflow-role EMR_EC2_DefaultRole --service-role  
EMR_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 \(p. 581\)](#).

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`.

```
{
  "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 \(p. 581\)](#).

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,--ver  
sion,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-in  
stances 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" --boot  
strap-action s3://elasticmapreduce/bootstrap-actions/configure-hbase-daemons  
--args "--hbase-zookeeper-opts=-Xmx1024m -XX:GCTimeRatio=19,--hbase-master-  
opts=-Xmx2048m,--hbase-regionserver-opts=-Xmx4096m"
```

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:

```
./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:

```
ruby elastic-mapreduce --create --hbase --name "My HBase Cluster" --boot  
strap-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:

```
./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:

```
ruby elastic-mapreduce --create --hbase --name "My HBase Cluster" --boot-   
strap-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 \(p. 31\)](#).

- 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/con   
figure-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 \ --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 \ --backup-dir s3://myawsbucket/backups/j-ABABABABABA \ --consistent
```

- Windows users:

```
ruby elastic-mapreduce --jobflow j-ABABABABABA --hbase-backup --backup-dir s3://myawsbucket/backups/j-ABABABABABA --consistent
```

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.

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 \  
--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
```

- Windows users:

```
ruby elastic-mapreduce --jobflow j-ABABABABABA --hbase-schedule-backup --full-  
backup-time-interval 7 --full-backup-time-unit days --backup-dir s3://mybuck  
et/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 \  
--hbase-schedule-backup \  
--incremental-backup-time-interval 24 \  
--incremental-backup-time-unit hours \  
--backup-dir s3://mybucket/backups/j-ABABABABABA
```

- Windows users:

```
ruby elastic-mapreduce --jobflow j-ABABABABABA --hbase-schedule-backup --in-  
cremental-backup-time-interval 24 --incremental-backup-time-unit hours -  
-backup-dir s3://mybucket/backups/j-ABABABABABA
```

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:

```
./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:

```
ruby elastic-mapreduce --jobflow j-ABABABABABA --hbase-schedule-backup --in-  
cremental-backup-time-interval 24 --incremental-backup-time-unit hours -  
-backup-dir s3://mybucket/backups/j-ABABABABABA --start-time 2012-06-15T20:00Z
```

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:

```
./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:

```
ruby 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://mybuck  
et/backups/j-ABABABABABA
```

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:

```
./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
```

- Windows users:

```
ruby 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://mybuck  
et/backups/j-ABABABABABA --start-time 2012-06-15T20:00Z
```

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.

- 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:

```
ruby 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://mybuck  
et/backups/j-ABABABABABA --start-time 2012-06-15T20:00Z --consistent
```

- 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 \  
--full-backup-time-interval 7 --full-backup-time-unit days \  
--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
```

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.

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --jobflow j-ABABABABABA \  
--hbase-schedule-backup --disable-full-backups \  
--disable-incremental-backups
```

- Windows users:

```
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:

```
./elastic-mapreduce --jobflow j-ABABABABABA --hbase-restore \  
--backup-dir s3://myawsbucket/backups/j-ABABABABABA
```

- Windows users:

```
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:

```
./elastic-mapreduce --jobflow j-ABABABABABA --hbase-restore \  
--backup-dir s3://myawsbucket/backups/j-ABABABABABA \  
--backup-version 20120809T031314Z
```

- Windows users:

```
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:

```
./elastic-mapreduce --create --name "My HBase Restored" \  
--hbase --hbase-restore \  
--backup-dir s3://myawsbucket/backups/j-ABABABABABA
```

- Windows users:

```
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:

```
./elastic-mapreduce --create --name "My HBase Restored" \  
--hbase --hbase-restore \  
--backup-dir s3://myawsbucket/backups/j-ABABABABABA
```

```
--backup-dir s3://myawsbucket/backups/j-ABABABABABA \  
--backup-version 20120809T031314Z
```

- Windows users:

```
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:

```
./elastic-mapreduce --hive-script --arg s3://mybucket/script.q \  
--args -d,LIB=s3://elasticmapreduce/samples/hive-ads/lib
```

- Windows users:

```
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:


```
./elastic-mapreduce --create --alive --name "Test Hive" \  
--num-instances 5 --instance-type m1.large \  
--hive-interactive \  
--hive-versions latest
```

- Windows users:

```
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:

```
./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:

```
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:

```
./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:

```
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:

```
./elastic-mapreduce --jobflow JobFlowID --print-hive-version
```

- Windows users:

```
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:

```
./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
```

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:

```
./elastic-mapreduce --create --name "Test Hive" --ami-version 3.3 --hive-script \s3://elasticmapreduce/samples/hive-ads/libs/model-build.g \--args -d,LIBS=s3://elasticmapreduce/samples/hive-ads/libs, \-d,INPUT=s3://elasticmapreduce/samples/hive-ads/tables, \-d,OUTPUT=s3://mybucket/hive-ads/output/
```

- Windows users:

```
ruby elastic-mapreduce --create --name "Test Hive" --ami-version 3.3 --hive-script s3://elasticmapreduce/samples/hive-ads/libs/model-build.g --args -d,LIBS=s3://elasticmapreduce/samples/hive-ads/libs, -d,INPUT=s3://elasticmapreduce/samples/hive-ads/tables, -d,OUTPUT=s3://mybucket/hive-ads/output/
```

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:

```
./elastic-mapreduce --create --alive \--name "Hive cluster" \--hive-interactive \--hive-site=s3://mybucket/hive-site.xml
```

- Windows users:

```
ruby elastic-mapreduce --create --alive --name "Hive cluster" --hive-interactive --hive-site=s3://mybucket/hive-site.xml
```

The `--hive-site` parameter 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`.

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.

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --jobflow JobFlowID --hive-interactive
```

- Windows users:

```
ruby elastic-mapreduce --jobflow JobFlowID --hive-interactive
```

2. Create a Hive script file containing the queries or commands to run. The following example script named `my-hive.q` creates two tables, `aTable` and `anotherTable`, and copies the contents of `aTable` to `anotherTable`, replacing all data.

```
---- sample Hive script file: my-hive.q ----  
create table aTable (aColumn string) ;  
create table anotherTable like aTable;  
insert overwrite table anotherTable select * from aTable
```

3. Type the following command, using the `--scp` parameter to copy the script from your local machine to the master node and the `--ssh` parameter to create an SSH connection and submit the Hive script for processing.

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --jobflow JobFlowID --scp my-hive.q \  
--ssh "hive -f my-hive.q"
```

- Windows users:

```
ruby elastic-mapreduce --jobflow JobFlowID --scp my-hive.q --ssh "hive -f  
my-hive.q"
```

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-script s3-path`. For example:

```
./elastic-mapreduce --create --alive --instance-type m1.large --instance-count 3 --ami-version 3.0.2 --impala-script s3://my-bucket/script-name.sql --impala-output s3://my-bucket/ --impala-conf "IMPALA_MEM_LIMIT=50%"
```

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---.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=DATETTIME`
Lists all clusters created after the specified time and date in XML date-time format.
- `--created-before=DATETTIME`
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:

```
./elastic-mapreduce --list
```

- Windows users:

```
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:

```
./elastic-mapreduce --list
```

- Windows users:

```
ruby elastic-mapreduce --list
```

The response is similar to the following:

```
j-1YE2DN7RXJBWU    FAILED    Example Job Flow
                  CANCELLED Custom Jar
j-3GJ4FRRNKG97    COMPLETED ec2-67-202-3-73.compute-1.amazonaws.com Example
cluster
j-5XXFIQS8PFNW    COMPLETED ec2-67-202-51-30.compute-1.amazonaws.com demo
3/24 s1
                  COMPLETED Custom Jar
```

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, 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:

```
./elastic-mapreduce --list --active
```

- Windows users:

```
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:

```
./elastic-mapreduce --list --state RUNNING --state TERMINATED
```

- Windows users:

```
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:

```
./elastic-mapreduce --describe --jobflow JobFlowID
```

- Windows users:

```
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.

- 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"
```

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 -j JobFlowID \  
  --jar s3n://elasticmapreduce/samples/cloudburst/cloudburst.jar \  
  --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:

```
ruby elastic-mapreduce -j JobFlowID --jar s3n://elasticmapre  
duce/samples/cloudburst/cloudburst.jar --arg s3n://elasticmapre  
duce/samples/cloudburst/input/s_suis.br --arg s3n://elasticmapre  
duce/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
```

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.

To add a step to run a script

- 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 "My Development Jobflow" \  
--jar s3://elasticmapreduce/libs/script-runner/script-runner.jar \  
--args "s3://mybucket/script-path/my_script.sh"
```

- Windows users:

```
ruby elastic-mapreduce --create --alive --name "My Development Jobflow" -  
-jar s3://elasticmapreduce/libs/script-runner/script-runner.jar --args  
"s3://mybucket/script-path/my_script.sh"
```

This cluster runs the script `my_script.sh` on the master node when the step is processed.

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 \  
--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 m1.large \  
--pig-interactive
```

- Windows users:

```
ruby elastic-mapreduce --create --alive --name "Testing Pig" --num-instances  
5 --instance-type m1.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,OUT  
PUT=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.

- Linux, UNIX, and Mac OS X users:

```
./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  
  
./elastic-mapreduce -j JobFlowID --wait-for-steps
```

- Windows users:

```
ruby elastic-mapreduce -j JobFlowID --jar s3n://elasticmapre  
duce/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

ruby elastic-mapreduce -j JobFlowID --wait-for-steps
```

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. 579). 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 --enable-debugging \  
--log-uri s3://mybucket
```

- Windows users:

```
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:

```
--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:

- 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://elasticmapre
duce/samples/wordcount/wordSplitter.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://elasticmapre
duce/samples/wordcount/wordSplitter.py" --bootstrap-action s3://elasticmapre
duce/bootstrap-actions/install-ganglia --input s3://elasticmapre
duce/samples/wordcount/input --output s3://mybucket/output/2014-01-16 --mapper
wordSplitter.py --reducer aggregate
```

For Hadoop 1.x, use the following command:

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --alive --instance-type m1.xlarge --num-instances
3 \
--bootstrap-action s3://elasticmapreduce/bootstrap-actions/install-ganglia -
-stream \
--input s3://elasticmapreduce/samples/wordcount/input \
--output s3://mybucket/output/2014-01-16 \
--mapper s3://elasticmapreduce/samples/wordcount/wordSplitter.py --reducer
aggregate
```

- Windows users:

```
ruby elastic-mapreduce --create --alive --instance-type m1.xlarge --num-in
stances 3 --bootstrap-action s3://elasticmapreduce/bootstrap-actions/install-
ganglia --stream --input s3://elasticmapreduce/samples/wordcount/input --output
s3://mybucket/output/2014-01-16 --mapper s3://elasticmapreduce/samples/word
count/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 \  
  --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://elasticmapre  
duce/bootstrap-actions/configure-daemons --args --namenode-heap-  
size=2048,--namenode-opts=-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 \  
--bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hadoop  
\  
--args "-M,s3://mybucket/config.xml,-m,mapred.tasktracker.map.tasks.maxim  
um=2"
```

- Windows:

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

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" \  
--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 execu  
tion" --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. Use `mapred.compress.map.output=false` to disable intermediate compression. Use `mapred.map.output.compression.codec=org.apache.hadoop.io.compress.GzipCodec` to change the compression codec to Gzip. Both arguments are presented below.
- Linux, UNIX, and Mac OS X users:

```
./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.com  
press.GzipCodec"
```

- Windows users:

```
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.out  
put=false" --args "-m,mapred.map.output.compression.codec=org.apache.ha  
doop.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:

```
./elastic-mapreduce --create --alive --name "Modified mapred.max.track  
er.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:

```
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/boot  
strap-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:

```
./elastic-mapreduce --create --alive \  
--bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hadoop \  
\  
--bootstrap-name "enable multipart upload" \  
--args "-c,fs.s3n.multipart.uploads.enabled=false"
```

- Windows users:

```
ruby elastic-mapreduce --create --alive --bootstrap-action s3://elasticmapre  
duce/bootstrap-actions/configure-hadoop --bootstrap-name "enable multipart  
upload" --args "-c,fs.s3n.multipart.uploads.enabled=false"
```

This cluster remains in the `WAITING` state until it is terminated.

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:

```
TAG cluster j-1234567890123 production  
TAG cluster j-1234567890123 costCenter marketing
```

In addition, you can apply the same tags to multiple clusters by specifying more than one cluster identifier separated by a space, for example:

```
./elastic-mapreduce --add-tags j-1234567890123 j-9876543210987 --tag produc  
tion --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:

```
./elastic-mapreduce --jobflow "j-1234567890123" --list-tags
```

- Windows users:

```
ruby elastic-mapreduce --jobflow "j-1234567890123" --list-tags
```

The output displays all the tag information about the cluster similar to the following:

```
Key: id Value: 2785  
Key: costCenter Value: marketing
```

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:

```
./elastic-mapreduce --remove-tags j-1234567890123 --tag "costCenter=marketing"
```

- Windows users:

```
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:

```
./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:

```
./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/word  
Splitter.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/out  
put/2014-01-16 --mapper wordSplitter.py --reducer aggregate --set-termina  
tion-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:

```
ruby 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
```

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:

```
./elastic-mapreduce --set-termination-protection true --jobflow JobFlowID
```

- Windows users:

```
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:

```
./elastic-mapreduce --terminate JobFlowID
```

- Windows users:

```
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.

```
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:

```
./elastic-mapreduce --terminate JobFlowID
```

- Windows users:

```
ruby elastic-mapreduce --terminate JobFlowID
```


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 \(p. 377\)](#).

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 \(p. 377\)](#). 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:

```
./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'
```

- Windows users:

```
ruby elastic-mapreduce --jobflow JobFlowID --jar /home/hadoop/lib/emr-  
s3distcp-1.0.jar --args "S3DistCp-OptionName1,S3DistCp-OptionValue1,S3Dist  
Cp-OptionName2,S3DistCp-OptionValue2,S3DistCp-OptionName3,S3DistCp-Option  
Value3"
```

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:

```
./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:

```
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:

```
./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:

```
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:

```
./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:

```
ruby elastic-mapreduce --jobflow j-3GYXXXXXX9IOJ --jar /home/hadoop/lib/emr-  
s3distcp-1.0.jar --args "--src,s3://myawsbucket/logs/j-3GY8JC4179IOJ/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 \(p. 174\)](#).

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-3GYXXXXXX9IOK --jar \  
/home/hadoop/lib/emr-s3distcp-1.0.jar \  
--args '--src,s3://mybucket/cf,--dest,hdfs:///local,--groupBy,.*XAB  
CD12345678.([0-9]+-[0-9]+-[0-9]+-[0-9]+).*,--targetSize,128,--outputCodec,lzo,-  
-deleteOnSuccess'
```

- Windows users:

```
ruby elastic-mapreduce --jobflow j-3GYXXXXXX9IOK --jar /home/hadoop/lib/emr-  
s3distcp-1.0.jar --args "--src,s3://myawsbucket/cf,--dest,hdfs:///local,--  
groupBy,.*XABCD12345678.([0-9]+-[0-9]+-[0-9]+-[0-9]+).*,--targetSize,128,--  
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  
s3://myawsbucket/cf/XABCD12345678.2012-02-23-02.LpLfuShd.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.

Release Date	Description
2014-05-15	Adds <code>--ec2-instance-ids-to-terminate</code> option. Adds support for Signature Version 4 signing with the CLI. Fixes a security issue.
2013-12-10	Adds support for Impala on Amazon EMR. For more information, see Impala (p. 279) .
2013-12-02	Adds support for Amazon EMR tags. For more information, see Tagging Amazon EMR Clusters (p. 211) .
2013-10-07	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.
2013-07-08	Fixes a bug that ignores any hard-coded Pig version number and incorrectly uses the latest Pig version.
2013-03-19	Improved support for launching clusters on third-party applications with a new <code>--supported-product</code> parameter that accepts custom user arguments.
2012-12-17	Adds support for IAM roles.
2012-09-18	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.
2012-08-22	Improved SSL certificate verification.
2012-07-30	Adds support for Hadoop 1.0.3.
2012-07-09	Adds support for specifying the major and minor AMI version and automatically getting the AMI that matches those specifications and contains the latest patches.
2012-06-12	Adds support for HBase and MapR.
2012-04-09	Adds support for Pig 0.9.1, Pig versioning, and Hive 0.7.1.4.
2012-03-13	Adds support for Hive 0.7.1.3.
2012-02-28	Adds support for Hive 0.7.1.2.
2011-12-08	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.

Amazon Elastic MapReduce Developer Guide
AWS EMR Command Line Interface Releases
(Deprecated)

Release Date	Description
2011-11-30	Fixes support for Elastic IP addresses.
2011-08-08	Adds support for running a cluster on Spot Instances.
2011-01-24	Fixes bugs in the <code>--json</code> command processing and the list option.
2011-12-08	Adds support for Hive 0.7.
2011-11-11	Fixes issues in the processing of pig and hive arguments and the <code>--main-class</code> argument to the custom jar step.
2011-10-19	Adds support for resizing running clusters. Substantially re-worked processing arguments to be more consistent and unit testable.
2011-09-16	Adds support for fetching files from Amazon EMR.
2011-06-02	Adds support for Hadoop 0.20, Hive 0.5 and Pig 0.6.
2011-04-07	Adds support for bootstrap actions.

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 documentation since the last release of Amazon Elastic MapReduce (Amazon EMR).

API version: 2009-03-31

Latest documentation update: June 16, 2015

Change	Description	Release Date
Apache Spark support	Amazon EMR natively supports Apache Spark. For more information, see Apache Spark (p. 268) .	June 16, 2015
AMI 3.8.0	Amazon EMR supports AMI 3.8.0. For more information, see AMI Versions Supported in Amazon EMR (p. 53) .	June 10, 2015
AMI 3.7.0	Amazon EMR supports AMI 3.7.0. For more information, see AMI Versions Supported in Amazon EMR (p. 53) .	April 21, 2015
D2 Instances	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 Instance Configurations (p. 35)	April 2, 2015
AWS CLI Parameter Values for Amazon EMR	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. 577)	April 2, 2015
EMRFS support for Amazon S3 client-side encryption	EMRFS natively supports Amazon S3 client-side encryption. For more information, see Using Amazon S3 Client-Side Encryption in EMRFS (p. 157) .	March 25, 2015
AMI 3.6.0	Amazon EMR supports AMI 3.6.0. For more information, see AMI Versions Supported in Amazon EMR (p. 53) .	March 24, 2015
AMI 3.5.0	Amazon EMR supports AMI 3.5.0. For more information, see AMI Versions Supported in Amazon EMR (p. 53) .	March 10, 2015
AMI 2.4.11 and 3.4.0	Amazon EMR supports AMI 3.4.0. For more information, see AMI Versions Supported in Amazon EMR (p. 53) .	February 26, 2015

Change	Description	Release Date
AMI 2.4.10	Amazon EMR supports AMI 2.4.10. For more information, see AMI Versions Supported in Amazon EMR (p. 53) .	February 13, 2015
AMI 3.3.2	Amazon EMR supports AMI 3.3.1. For more information, see AMI Versions Supported in Amazon EMR (p. 53) .	February 4, 2015
AMI 3.3.1	Amazon EMR supports AMI 3.3.1. For more information, see AMI Versions Supported in Amazon EMR (p. 53) .	November 20, 2014
Hue support	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. 334) .	November 6, 2014
Consistent view	Amazon EMR supports EMRFS consistent view. For more information, see the section called "Consistent View" (p. ?) .	September 17, 2014
AMIs 2.4.8, 3.1.2, and 3.2.1	Amazon EMR supports these new images. For more information, see AMI Versions Supported in Amazon EMR (p. 53) .	September 16, 2014
AMI 3.1.1	Amazon EMR supports AMI 3.1.1. For more information, see AMI Versions Supported in Amazon EMR (p. 53) .	August 15, 2014
AMI 2.4.7	Amazon EMR supports AMI 2.4.7. For more information, see AMI Versions Supported in Amazon EMR (p. 53) .	July 30, 2014
AMI 2.4.6	Amazon EMR supports AMI 2.4.6. For more information, see AMI Versions Supported in Amazon EMR (p. 53) .	May 15, 2014
AMI 3.1.0	Amazon EMR supports AMI 3.1.0. For more information, see AMI Versions Supported in Amazon EMR (p. 53) .	May 15, 2014
AMI 2.4.5	Amazon EMR supports AMI 2.4.5. For more information, see AMI Versions Supported in Amazon EMR (p. 53) .	March 27, 2014
AMI 3.0.4	Amazon EMR supports AMI 3.0.4 and a connector for Amazon Kinesis . For more information, see AMI Versions Supported in Amazon EMR (p. 53) .	February 20, 2014
AMI 3.0.3	Amazon EMR supports AMI 3.0.3. For more information, see AMI Versions Supported in Amazon EMR (p. 53) .	February 11, 2014
Hive 0.11.0.2	Amazon EMR supports Hive 0.11.0.2. For more information, see Supported Hive Versions (p. 251) .	February 11, 2014
Impala 1.2.1	Amazon EMR supports Impala 1.2.1 with Hadoop 2. For more information, see Impala (p. 279) .	December 12, 2013
AMI 3.0.2	Amazon EMR supports AMI 3.0.2. For more information, see AMI Versions Supported in Amazon EMR (p. 53) .	December 12, 2013
Amazon EMR tags	Amazon EMR supports tagging on Amazon EMR clusters. For more information, see Tagging Amazon EMR Clusters (p. 211) .	December 5, 2013
CLI version 2013-12-02	Adds support for Amazon EMR tags. For more information, see AWS EMR Command Line Interface Releases (Deprecated) (p. 644) .	December 5, 2013
AMI 3.0.1	Amazon EMR supports AMI 3.0.1. For more information, see AMI Versions Supported in Amazon EMR (p. 53) .	November 8, 2013

Change	Description	Release Date
New Amazon EMR console	<p>A new management console is available for Amazon EMR. The new console is much faster and has powerful new features, including:</p> <ul style="list-style-type: none"> Resizing a running cluster (that is, adding or removing instances) Cloning the launch configurations for running or terminated clusters Hadoop 2 support, including custom Amazon CloudWatch metrics Targeting specific Availability Zones Creating clusters with IAM roles Submitting multiple steps (before and after cluster creation) New console help portal with integrated documentation search 	November 6, 2013
MapR 3.0.2	Amazon EMR supports MapR 3.0.2. For more information, see Using the MapR Distribution for Hadoop (p. 218) .	November 6, 2013
Hadoop 2.2.0	Amazon EMR supports Hadoop 2.2.0. For more information, see Hadoop 2.2.0 New Features (p. 118) .	October 29, 2013
AMI 3.0.0	Amazon EMR supports AMI 3.0.0. For more information, see AMI Versions Supported in Amazon EMR (p. 53) .	October 29, 2013
CLI version 2013-10-07	Maintenance update for the Amazon EMR CLI. For more information, see AWS EMR Command Line Interface Releases (Deprecated) (p. 644) .	October 7, 2013
AMI 2.4.2	Amazon EMR supports AMI 2.4.2 For more information, see AMI Versions Supported in Amazon EMR (p. 53) .	October 7, 2013
AMI 2.4.1	Amazon EMR supports AMI 2.4.1 For more information, see AMI Versions Supported in Amazon EMR (p. 53) .	August 20, 2013
Hive 0.11.0.1	Amazon EMR supports Hive 0.11.0.1. For more information, see Supported Hive Versions (p. 251) .	August 2, 2013
Hive 0.11.0	Amazon EMR supports Hive 0.11.0. For more information, see Supported Hive Versions (p. 251) .	August 1, 2013
Pig 0.11.1.1	Amazon EMR supports Pig 0.11.1.1. For more information, see Supported Pig Versions (p. 302) .	August 1, 2013
AMI 2.4	Amazon EMR supports AMI 2.4. For more information, see AMI Versions Supported in Amazon EMR (p. 53) .	August 1, 2013
MapR 2.1.3	Amazon EMR supports MapR 2.1.3. For more information, see Using the MapR Distribution for Hadoop (p. 218) .	August 1, 2013
MapR M7 Edition	Amazon EMR supports MapR M7 Edition. For more information, see Using the MapR Distribution for Hadoop (p. 218) .	July 11, 2013

Change	Description	Release Date
CLI version 2013-07-08	Maintenance update to the Amazon EMR CLI version 2013-07-08. For more information, see AWS EMR Command Line Interface Releases (Deprecated) (p. 644).	July 11, 2013
Pig 0.11.1	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. 302).	July 1, 2013
Hive 0.8.1.8	Amazon EMR supports Hive 0.8.1.8. For more information, see Supported Hive Versions (p. 251).	June 18, 2013
AMI 2.3.6	Amazon EMR supports AMI 2.3.6. For more information, see AMI Versions Supported in Amazon EMR (p. 53).	May 17, 2013
Hive 0.8.1.7	Amazon EMR supports Hive 0.8.1.7. For more information, see Supported Hive Versions (p. 251).	May 2, 2013
Improved documentation organization, new table of contents, and new topics	Updated documentation organization with a restructured table of contents and many new topics for better ease of use and to accommodate customer feedback.	April 29, 2013
AMI 2.3.5	Amazon EMR supports AMI 2.3.5. For more information, see AMI Versions Supported in Amazon EMR .	April 26, 2013
M1 Medium Amazon EC2 Instances	Amazon EMR supports m1.medium instances. For more information, see Hadoop 2.2.0 and 2.4.0 Default Configuration (p. 527).	April 18, 2013
MapR 2.1.2	Amazon Elastic MapReduce supports MapR 2.1.2. For more information, see Using the MapR Distribution for Hadoop (p. 218).	April 18, 2013
AMI 2.3.4	Deprecated.	April 16, 2013
AWS GovCloud (US)	Adds support for AWS GovCloud (US). For more information, see AWS GovCloud (US) .	April 9, 2013
Supported Product User Arguments	Improved support for launching job flows on third-party applications with a new <code>--supported-product</code> CLI option that accepts custom user arguments. For more information, see Launch an Amazon EMR cluster with MapR using the console (p. 218).	March 19, 2013
Amazon VPC	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 .	March 11, 2013
AMI 2.3.3	Amazon Elastic MapReduce supports AMI 2.3.3. For more information, see AMI Versions Supported in Amazon EMR .	March 1, 2013
High I/O Instances	Amazon Elastic MapReduce supports hi1.4xlarge instances. For more information, see Hadoop 2.2.0 and 2.4.0 Default Configuration (p. 527).	February 14, 2013
AMI 2.3.2	Amazon Elastic MapReduce supports AMI 2.3.2. For more information, see AMI Versions Supported in Amazon EMR .	February 7, 2013

Change	Description	Release Date
New introduction and tutorial	Added sections that describe Amazon EMR and a tutorial that walks you through your first streaming cluster.	January 9, 2013
CLI Reference	Added CLI reference. For more information, see Command Line Interface Reference for Amazon EMR (p. 577) .	January 8, 2013
AMI 2.3.1	Amazon Elastic MapReduce supports AMI 2.3.1. For more information, see AMI Versions Supported in Amazon EMR .	December 24, 2012
High Storage Instances	Amazon Elastic MapReduce supports hs1.8xlarge instances. For more information, see Hadoop 2.2.0 and 2.4.0 Default Configuration (p. 527) .	December 20, 2012
IAM Roles	Amazon Elastic MapReduce supports IAM Roles. For more information, see Configure IAM Roles for Amazon EMR (p. 185) .	December 20, 2012
Hive 0.8.1.6	Amazon Elastic MapReduce supports Hive 0.8.1.6. For more information, see Supported Hive Versions (p. 251) .	December 20, 2012
AMI 2.3.0	Amazon Elastic MapReduce supports AMI 2.3.0. For more information, see AMI Versions Supported in Amazon EMR .	December 20, 2012
AMI 2.2.4	Amazon Elastic MapReduce supports AMI 2.2.4. For more information, see AMI Versions Supported in Amazon EMR .	December 6, 2012
AMI 2.2.3	Amazon Elastic MapReduce supports AMI 2.2.3. For more information, see AMI Versions Supported in Amazon EMR .	November 30, 2012
Hive 0.8.1.5	Amazon Elastic MapReduce supports Hive 0.8.1.5. For more information, see Hive and Amazon EMR (p. 240) .	November 30, 2012
Asia Pacific (Sydney)	Adds support for Amazon EMR in the Asia Pacific (Sydney) region.	November 12, 2012
Visible To All IAM Users	Added support making a cluster visible to all IAM users on an AWS account. For more information, see Configure IAM User Permissions (p. 181) .	October 1, 2012
Hive 0.8.1.4	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.	September 17, 2012
AMI 2.2.1	<ul style="list-style-type: none"> Fixes an issue with HBase backup functionality. Enables multipart upload by default for files larger than the Amazon S3 block size specified by <code>fs.s3n.block-size</code>. For more information, see Configure Multipart Upload for Amazon S3 (p. 167). 	August 30, 2012
AMI 2.1.4	<ul style="list-style-type: none"> 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. 167). 	August 30, 2012

Change	Description	Release Date
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. 115) .	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. 48) .	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. 310) .	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. 218) .	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. 441) .	June 12, 2012
Hive 0.8.1	Amazon Elastic MapReduce supports Hive 0.8.1. For more information, see Hive and Amazon EMR (p. 240) .	May 30, 2012
HParser	Added information about running Informatica HParser on Amazon EMR. For more information, see Parse Data with HParser (p. 217) .	April 30, 2012
AMI 2.0.5	Enhancements to performance and other updates. For more information, see AMI Versions Supported in Amazon EMR (p. 53) .	April 19, 2012
Pig 0.9.2	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. 304) .	April 9, 2012
Pig versioning	Amazon Elastic MapReduce supports the ability to specify the Pig version when launching a cluster. For more information, see Apache Pig (p. 302) .	April 9, 2012
Hive 0.7.1.4	Amazon Elastic MapReduce supports Hive 0.7.1.4. For more information, see Hive and Amazon EMR (p. 240) .	April 9, 2012

Change	Description	Release Date
AMI 1.0.1	Updates sources.list to the new location of the Lenny distribution in archive.debian.org.	April 3, 2012
Hive 0.7.1.3	Support for new version of Hive, version 0.7.1.3. This version adds the <code>dynamodb.retry.duration</code> 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.	March 13, 2012
Support for IAM in the console	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.	February 28, 2012
Support for Cloud-Watch Metrics	Support for monitoring cluster metrics and setting alarms on metrics.	January 31, 2012
Support for S3DistCp	Support for distributed copy using S3DistCp.	January 19, 2012
Support for DynamoDB	Support for exporting and querying data stored in DynamoDB.	January 18, 2012
AMI 2.0.2 and Hive 0.7.1.1	Support for Amazon EMR AMI 2.0.2 and Hive 0.7.1.1.	January 17, 2012
Cluster Compute Eight Extra Large (cc2.8xlarge)	Support for Cluster Compute Eight Extra Large (cc2.8xlarge) instances in clusters.	December 21, 2011
Hadoop 0.20.205	Support for Hadoop 0.20.205. For more information, see Supported Hadoop Versions (p. 115) .	December 11, 2011
Pig 0.9.1	Support for Pig 0.9.1. For more information see Supported Pig Versions (p. 302) .	December 11, 2011
AMI versioning	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. 48) .	December 11, 2011
Amazon EMR clusters on Amazon VPC	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 Select an Amazon VPC Subnet for the Cluster (Optional) (p. 205) .	December 11, 2011
Spot Instances	Support for launching cluster instance groups as Spot Instances added. For more information, see (Optional) Lower Costs with Spot Instances (p. 37) .	August 19, 2011
Hive 0.7.1	Support for Hive 0.7.1 added. For more information, see Supported Hive Versions (p. 251) .	July 25, 2011
Termination Protection	Support for a new Termination Protection feature. For more information, see Managing Cluster Termination (p. 462) .	April 14, 2011
Tagging	Support for Amazon EC2 tagging. For more information, see View Cluster Instances in Amazon EC2 (p. 417) .	March 9, 2011

Change	Description	Release Date
IAM Integration	Support for AWS Identity and Access Management. For more information, see Configure IAM User Permissions (p. 181) and Configure IAM User Permissions (p. 181) .	February 21, 2011
Elastic IP Support	Support for Elastic IP addresses.	February 21, 2011
Environment Configuration	Expanded sections on Environment Configuration and Performance Tuning. For more information, see (Optional) Create Bootstrap Actions to Install Additional Software (p. 124) .	February 21, 2011
Distributed Cache	For more information about using DistributedCache to upload files and libraries, see Import files with Distributed Cache (p. 168) .	February 21, 2011
How to build modules using Amazon EMR	For more information, see Build Binaries Using Amazon EMR (p. 228) .	February 21, 2011
Amazon S3 multipart upload	Support of Amazon S3 multipart upload through the AWS SDK for Java. For more information, see Configure Multipart Upload for Amazon S3 (p. 167) .	January 6, 2010
Hive 0.70	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 (p. 240) .	December 8, 2010
JDBC Drivers for Hive	Support for JDBC with Hive 0.5 and Hive 0.7. For more information, see Use the Hive JDBC Driver (p. 264) .	December 8, 2010
Support HPC	Support for cluster compute instances. For more information, see Instance Configurations (p. 35) .	November 14, 2010
Bootstrap Actions	Expanded content and samples for bootstrap actions. For more information, see (Optional) Create Bootstrap Actions to Install Additional Software (p. 124) .	November 14, 2010
Cascading clusters	Description of Cascading cluster support. For more information, see Submit a Cascading Step (p. 236) and Process Data Using Cascading (p. 235) .	November 14, 2010
Resize Running Cluster	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) , Resize a Running Cluster (p. 465) , and Resize a Running Cluster (p. 465) .	October 19, 2010
Appendix: Configuration Options	Expanded information on configuration options available in Amazon EMR. For more information, see Hadoop Configuration Reference (p. 522) .	October 19, 2010
Guide revision	This release features a reorganization of the <i>Amazon Elastic MapReduce Developer Guide</i> .	October 19, 2010