
Amazon Elastic MapReduce

Developer Guide

API Version 2009-03-31



Amazon Elastic MapReduce: Developer Guide

Copyright © 2014 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, Cloudfront, Amazon DevPay, DynamoDB, ElastiCache, Amazon EC2, Amazon Elastic Compute Cloud, Amazon Glacier, Kindle, Kindle Fire, AWS Marketplace Design, Mechanical Turk, Amazon Redshift, Amazon Route 53, Amazon S3, Amazon VPC. 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	5
AWS Integration	5
Instance Options	5
MapR Support	5
Business Intelligence Tools	5
User Control	5
Management Tools	6
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 Amazon EMR	11
Learn More About Hadoop and AWS Services Used with Amazon EMR:	12
Get Started: Count Words with Amazon EMR	13
How Much Does it Cost to Run this Tutorial?	13
Sign up for the Service	13
Launch the Cluster	14
Monitor the Cluster	21
View the Results	24
View the Debug Logs (Optional)	24
Clean Up	26
Where Do I Go From Here?	28
Next Steps	28
Plan an Amazon EMR Cluster	29
Choose an AWS Region	29
Choose a Region using the Console	30
Choose a Region using the CLI	31
Choose a Region Using an SDK or the API	31
Choose the Type of Cluster to Run	32
Hive Cluster	32
Custom JAR Cluster	32
Streaming Cluster	32
Pig Cluster	33
HBase Cluster	33
Choose the Number and Type of Virtual Servers	33
Calculate the HDFS Capacity of a Cluster	33
Guidelines for the Number and Type of Virtual Servers	34
Instance Groups	35
Virtual Server Configurations	36

Ensure Capacity with Reserved Instances (Optional)	36
Lower Costs with Spot Instances (Optional)	37
Configure the Virtual Server Software	50
Choose a Machine Image	51
Choose a Version of Hadoop	79
Create Bootstrap Actions to Install Additional Software (Optional)	87
Choose the Cluster Lifecycle: Long-Running or Transient	97
Prepare Input Data (Optional)	98
Types of Input Amazon EMR Can Accept	99
File Systems compatible with Amazon EMR	99
How to Get Data Into Amazon EMR	102
Prepare an Output Location (Optional)	112
Create and Configure an Amazon S3 Bucket	112
What formats can Amazon EMR return?	114
How to write data to an Amazon S3 bucket you don't own	114
Compress the Output of your Cluster	116
Configure Access to the Cluster	117
Create SSH Credentials for the Master Node	117
Configure IAM User Permissions	119
Set Access Policies for IAM Users	122
Configure IAM Roles for Amazon EMR	125
Setting Permissions on the System Directory	133
Configure Logging and Debugging (Optional)	133
Default Log Files	133
Archive Log Files to Amazon S3	134
Enable the Debugging Tool	136
Select a Amazon VPC Subnet for the Cluster (Optional)	137
Clusters in a VPC	138
Setting Up a VPC to Host Clusters	139
Launching Clusters into a VPC	140
Restricting Permissions to a VPC Using IAM	141
Tagging Amazon EMR Clusters	143
Tag Restrictions	143
Tagging Resources for Billing	144
Adding Tags to a New Cluster	144
Adding Tags to an Existing Cluster	145
Viewing Tags on a Cluster	146
Removing Tags from a Cluster	147
Use Third Party Applications With Amazon EMR (Optional)	148
Use Business Intelligence Tools with Amazon EMR	148
Parse Data with HParser	148
Using the MapR Distribution for Hadoop	149
Run a Hadoop Application to Process Data	161
Build Binaries Using Amazon EMR	161
JAR Requirements	166
Run a Script in a Cluster	166
CLI	166
Process Data with a Streaming Cluster	167
Using the Hadoop Stream Utility	167
Launch a Streaming Cluster	169
Process Data with a Cascading Cluster	177
Launch a Cascading Cluster	177
Multitool Cascading Application	185
Process Data with a Custom JAR Cluster	192
Launch a Custom JAR Cluster	194
Analyze Data with Hive	202
How Amazon EMR Hive Differs from Apache Hive	202
Input Format	203

Combine Splits Input Format	203
Log files	203
Thrift Service Ports	204
Hive Authorization	204
Hive File Merge Behavior with Amazon S3	204
Additional Features of Hive in Amazon EMR	205
Supported Hive Versions	213
Display the Hive Version	221
Upgrade to Hive 0.11.0	222
Share Data Between Hive Versions	230
Interactive and Batch Hive Clusters	231
Running Hive in Interactive Mode	232
Running Hive in Batch Mode	233
Launch a Hive Cluster	234
Amazon EMR Console	234
CLI	241
Create a Metastore Outside the Hadoop Cluster	241
Use the Hive JDBC Driver	243
Analyze Data with Impala	247
What Can I Do With Impala?	247
Differences from Traditional Relational Databases	248
Differences from Hive	248
Tutorial: Launching and Querying Impala Clusters on Amazon EMR	249
Sign up for the Service	249
Launch the Cluster	249
Generate Test Data	255
Create and Populate Impala Tables	256
Query Data in Impala	256
Impala Examples Included on the Amazon EMR AMI	257
TPCDS	258
Wikipedia	259
Supported Impala Versions	260
Updates for Impala 1.2.4	260
Impala Memory Considerations	261
Using Impala with JDBC	262
Accessing Impala Web User Interfaces	262
Impala-supported File and Compression Formats	262
Impala SQL Dialect	263
Statements	263
Functions	263
Data Types	264
Operators	264
Clauses	264
Impala User-Defined Functions	265
Impala Performance Testing and Query Optimization	265
Database Schema	265
Sample Data	266
Table Size	266
Queries	267
Performance Test Results	268
Optimizing Queries	271
Process Data with Pig	273
Supported Pig Versions	273
Pig Version Details	277
Additional Pig Functions	279
Interactive and Batch Pig Clusters	279
Run Pig in Interactive Mode	279
Run Pig in Batch Mode	280

Launch a Pig Cluster	281
Amazon EMR Console	281
Call User Defined Functions from Pig	287
Call JAR files from Pig	287
Call Python/Jython Scripts from Pig	288
Store Data with HBase	289
What Can I Do with HBase?	289
Supported HBase Versions	290
HBase Cluster Prerequisites	290
Launch an HBase Cluster on Amazon EMR	291
Connect to HBase Using the Command Line	298
Create a Table	298
Put a Value	298
Get a Value	298
Back Up and Restore HBase	298
Back Up and Restore HBase Using the Console	299
Back Up and Restore HBase Using the CLI	301
Terminate an HBase Cluster	309
Configure HBase	309
Configure HBase Daemons	309
Configure HBase Site Settings	310
HBase Site Settings to Optimize	312
Access HBase Data with Hive	313
View the HBase User Interface	315
View HBase Log Files	315
Monitor HBase with CloudWatch	316
Monitor HBase with Ganglia	317
Analyze Amazon Kinesis Data	319
What Can I Do With Amazon EMR and Amazon Kinesis Integration?	319
Checkpointed Analysis of Amazon Kinesis Streams	319
Provisioned IOPS Recommendations for Amazon DynamoDB Tables	320
Performance Considerations	321
Tutorial: Analyzing Amazon Kinesis Streams with Amazon EMR and Hive	321
Sign Up for the Service	321
Create an Amazon Kinesis Stream	322
Create an DynamoDB Table	324
Download Log4J Appender for Amazon Kinesis Sample Application, Sample Credentials File, and Sample Log File	325
Start Amazon Kinesis Publisher Sample Application	326
Launch the Cluster	327
Run the Ad-hoc Hive Query	333
Running Queries with Checkpoints	336
Scheduling Scripted Queries	336
Tutorial: Analyzing Amazon Kinesis Streams with Amazon EMR and Pig	338
Sign Up for the Service	338
Create an Amazon Kinesis Stream	339
Create an DynamoDB Table	340
Download Log4J Appender for Amazon Kinesis Sample Application, Sample Credentials File, and Sample Log File	341
Start Amazon Kinesis Publisher Sample Application	342
Launch the Cluster	344
Run the Pig Script	349
Scheduling Scripted Queries	352
Schedule Amazon Kinesis Analysis with Amazon EMR Clusters	353
Extract, Transform, and Load (ETL) Data with Amazon EMR	355
Distributed Copy Using S3DistCp	355
S3DistCp Options	356
Adding S3DistCp as a Step in a Cluster	360

S3DistCp Versions Supported in Amazon EMR	364
Export, Query, and Join Tables in DynamoDB	364
Prerequisites for Integrating Amazon EMR	365
Step 1: Create a Key Pair	366
Step 2: Create a Cluster	366
Step 3: SSH into the Master Node	374
Step 4: Set Up a Hive Table to Run Hive Commands	376
Hive Command Examples for Exporting, Importing, and Querying Data	381
Optimizing Performance	388
Store Avro Data in Amazon S3 Using Amazon EMR	390
Analyze Elastic Load Balancing Log Data	393
Tutorial: Query Elastic Load Balancing Access Logs with Amazon Elastic MapReduce	393
Sign Up for the Service	394
Launch the Cluster and Run the Script	395
Interactively Query the Data in Hive	403
Using AWS Data Pipeline to Schedule Access Log Processing	405
Manage Clusters	406
View and Monitor a Cluster	406
View Cluster Details	407
View Web Interfaces on the Cluster (Hadoop 2.x)	411
View Log Files	418
View Cluster Instances in Amazon EC2	422
Monitor Metrics with CloudWatch	423
Logging Amazon Elastic MapReduce API Calls in AWS CloudTrail	436
Monitor Performance with Ganglia	438
Connect to the Cluster	446
Connect to the Master Node Using SSH	446
Web Interfaces Hosted on the Master Node	450
Open an SSH Tunnel to the Master Node	452
Configure FoxyProxy to View Websites Hosted on the Master Node	453
Control Cluster Termination	458
Terminate a Cluster	458
Protect a Cluster from Termination	459
Resize a Running Cluster	464
Resize a Cluster Using the Console	465
Parameters for Resizing Clusters	466
Arrested State	469
Legacy Clusters	470
Library Files	471
Cloning a Cluster Using the Console	472
Add Steps to a Cluster	473
Wait for Steps to Complete	474
Add More than 256 Steps to a Cluster	475
Associate an Elastic IP Address with a Cluster	477
Assign an Elastic IP Address to a Cluster	478
View Allocated Elastic IP Addresses using Amazon EC2	480
Manage Elastic IP Addresses using Amazon EC2	480
Automate Recurring Clusters with AWS Data Pipeline	480
Troubleshoot a Cluster	481
What Tools are Available for Troubleshooting?	481
Tools to Display Cluster Details	481
Tools to View Log Files	482
Tools to Monitor Cluster Performance	482
Troubleshoot a Failed Cluster	483
Step 1: Gather Data About the Issue	483
Step 2: Check the Environment	484
Step 3: Look at the Last State Change	485
Step 4: Examine the Log Files	485

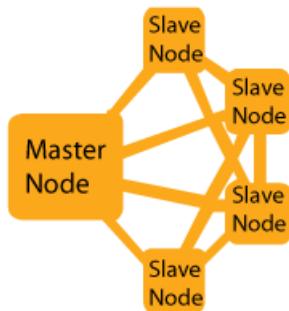
Step 5: Test the Cluster Step by Step	486
Troubleshoot a Slow Cluster	487
Step 1: Gather Data About the Issue	487
Step 2: Check the Environment	488
Step 3: Examine the Log Files	489
Step 4: Check Cluster and Instance Health	490
Step 5: Check for Arrested Groups	491
Step 6: Review Configuration Settings	491
Step 7: Examine Input Data	493
Common Errors in Amazon EMR	493
Input and Output Errors	493
Permissions Errors	496
Memory Errors	497
Resource Errors	498
Streaming Cluster Errors	501
Custom JAR Cluster Errors	502
Hive Cluster Errors	503
VPC Errors	504
GovCloud-related Errors	505
Write Applications that Launch and Manage Clusters	506
End-to-End Amazon EMR Java Source Code Sample	506
Common Concepts for API Calls	510
Endpoints for Amazon EMR	510
Specifying Cluster Parameters in Amazon EMR	510
Availability Zones in Amazon EMR	511
How to Use Additional Files and Libraries in Amazon EMR Clusters	511
Amazon EMR Sample Applications	511
Use SDKs to Call Amazon EMR APIs	512
Using the AWS SDK for Java to Create an Amazon EMR Cluster	512
Using the AWS SDK for .Net to Create an Amazon EMR Cluster	513
Using the Java SDK to Sign an API Request	514
Hadoop Configuration Reference	515
JSON Configuration Files	515
Node Settings	515
Cluster Configuration	517
Configuration of hadoop-user-env.sh	519
Hadoop 2.2.0 and 2.4.0 Default Configuration	520
Hadoop Configuration (Hadoop 2.2.0, 2.4.0)	520
HDFS Configuration (Hadoop 2.2.0)	528
Task Configuration (Hadoop 2.2.0)	529
Intermediate Compression (Hadoop 2.2.0)	540
Hadoop 1.0.3 Default Configuration	541
Hadoop Configuration (Hadoop 1.0.3)	541
HDFS Configuration (Hadoop 1.0.3)	551
Task Configuration (Hadoop 1.0.3)	551
Intermediate Compression (Hadoop 1.0.3)	555
Hadoop 20.205 Default Configuration	556
Hadoop Configuration (Hadoop 20.205)	557
HDFS Configuration (Hadoop 20.205)	560
Task Configuration (Hadoop 20.205)	561
Intermediate Compression (Hadoop 20.205)	565
Hadoop Memory-Intensive Configuration Settings (Legacy AMI 1.0.1 and earlier)	565
Hadoop Default Configuration (AMI 1.0)	569
Hadoop Configuration (AMI 1.0)	569
HDFS Configuration (AMI 1.0)	572
Task Configuration (AMI 1.0)	573
Intermediate Compression (AMI 1.0)	577
Hadoop 0.20 Streaming Configuration	577

Command Line Interface Reference for Amazon EMR	579
Install the Amazon EMR Command Line Interface	579
Installing Ruby	579
Verifying the RubyGems package management framework	580
Installing the Command Line Interface	581
Configuring Credentials	581
SSH Setup and Configuration	584
How to Call the Command Line Interface	585
Command Line Interface Options	585
Common Options	586
Uncommon Options	587
Options Common to All Step Types	587
Short Options	587
Adding and Modifying Instance Groups	587
Adding JAR Steps to Job Flows	588
Adding JSON Steps to Job Flows	588
Adding Streaming Steps to Job Flows	588
Assigning an Elastic IP Address to the Master Node	589
Contacting the Master Node	589
Creating Job Flows	590
HBase Options	591
Hive Options	593
Impala Options	593
Listing and Describing Job Flows	594
Passing Arguments to Steps	594
Pig Options	595
Specific Steps	595
Specifying Bootstrap Actions	596
Tagging	596
Terminating job flows	596
Command Line Interface Releases	596
Document History	599

What is Amazon EMR?

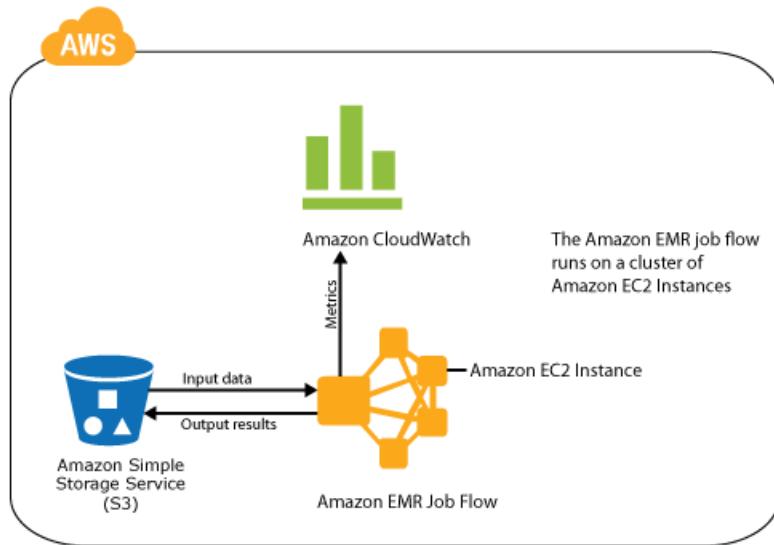
With Amazon Elastic MapReduce (Amazon EMR) you can analyze and process vast amounts of data. It does this by distributing the computational work across a cluster of virtual servers running in the Amazon cloud. The cluster is managed using an open-source framework called [Hadoop](#).

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 types of clusters that you can launch to run custom Hadoop map-reduce code, depending on the type of program you're developing and the libraries you intend to use.

Custom JAR

Run your custom map-reduce program written in Java. This cluster provides low-level access to the MapReduce API. You have the responsibility of defining and implementing the map reduce tasks in your Java application.

Cascading

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

Streaming

Run a single 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 map-reduce 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 \(p. 5\)](#)
- [AWS Integration \(p. 5\)](#)
- [Instance Options \(p. 5\)](#)
- [MapR Support \(p. 5\)](#)
- [Business Intelligence Tools \(p. 5\)](#)
- [User Control \(p. 5\)](#)
- [Management Tools \(p. 6\)](#)
- [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 don't pay ongoing overhead costs for hardware maintenance and upgrades and you don't 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 [Analyze Data with Hive \(p. 202\)](#), [Process Data with Pig \(p. 273\)](#), and [Store Data with HBase \(p. 289\)](#).

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

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 \(p. 423\)](#) and [Export, Import, Query, and Join Tables in DynamoDB Using Amazon EMR \(p. 364\)](#).

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 Virtual Servers \(p. 33\)](#).

MapR Support

Amazon EMR supports several MapR distributions. For more information, see [Using the MapR Distribution for Hadoop \(p. 149\)](#).

Business Intelligence Tools

Amazon EMR integrates with popular business intelligence (BI) tools such as [Tableau](#), [MicroStrategy](#), and [Datameer](#).

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 [Create Bootstrap Actions to Install Additional Software \(Optional\) \(p. 87\)](#).

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 \(p. 117\)](#).

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: nodes, steps, and job flows.

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 \(p. 7\)](#)
- [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 [Analyze Data with Hive \(p. 202\)](#), [Process Data with Pig \(p. 273\)](#), and [Store Data with HBase \(p. 289\)](#).

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 \(p. 35\)](#). For details on mapping legacy clusters to instance groups, see [Mapping Legacy Clusters to Instance Groups \(p. 470\)](#).

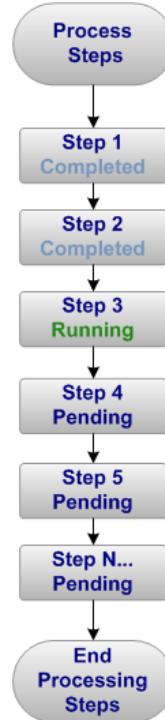
Steps

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

The maximum number of steps an Amazon EMR cluster can process is 256, which includes the additional steps that Amazon EMR adds when you enable debugging. If you need to run more steps than this, such as on a long-running cluster, you can connect to the cluster and submit jobs directly to Hadoop. For more information about how to define the steps run on a cluster, see [How to Send Work to a Cluster \(p. 9\)](#).

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*.

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.

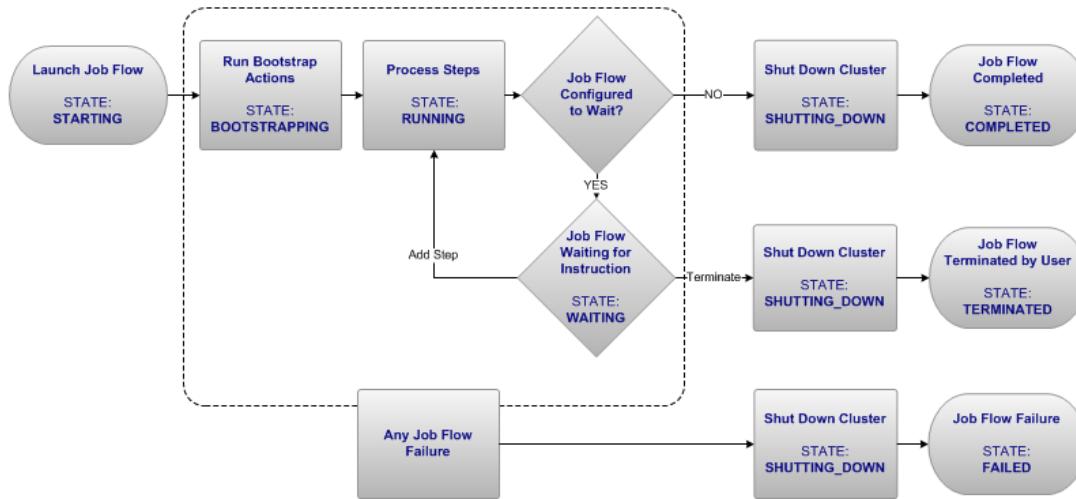
How to Send Work to a Cluster

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 \(p. 161\)](#).
- Create a long-running cluster and use the Amazon EMR API to submit steps, which may contain one or more Hadoop jobs. For more information, see [Add Steps to a Cluster \(p. 473\)](#).
- Create a cluster using a Hadoop application such as Hive, Pig, or HBase and use the interface provided by the application to submit queries, either scripted or interactively. For more information, see [Analyze Data with Hive \(p. 202\)](#), [Process Data with Pig \(p. 273\)](#), and [Store Data with HBase \(p. 289\)](#).
- 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`. 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 \(p. 97\)](#). 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 [Protect a Cluster from Termination \(p. 459\)](#).

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/>.
- **Command Line Interface (CLI)** — an 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. Using the CLI is the best option if you prefer working from a command line. For more information, see [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- **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	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	✓	✓	✓
Transfer data to and from Amazon S3 automatically	✓	✓	✓
Terminate clusters in real time	✓	✓	

Learn more about Amazon EMR

The following table lists related resources that you'll find useful as you work with Amazon EMR.

Resource	Description
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.
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.

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 Elastic Compute Cloud User Guide](#), the [Amazon Simple Storage Service Developer Guide](#), [Amazon SimpleDB Developer Guide](#) and the [Amazon CloudWatch Developer Guide](#), respectively.

Get Started: Count Words with Amazon EMR

Now that you know what Amazon EMR can do, let's walk through a tutorial using mapper and reducer functions to analyze data in a streaming cluster. In this example, you'll use Amazon EMR to count the frequency of words in a text file. The mapper logic is written as a Python script and you'll use the built-in `aggregator` function provided by Hadoop as the reducer. Using the Amazon EMR console, you'll launch a cluster of virtual servers into a cluster to process the data in a distributed fashion, according to the logic in the Python script and the `aggregator` function.

In addition to the console used in this tutorial, Amazon EMR provides a command-line client, a REST-like API set, 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? \(p. 10\)](#).

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](#).

How Much Does it Cost to Run this Tutorial?

The AWS service charges incurred by working through this tutorial are the cost of running an Amazon EMR cluster containing three m1.small instances for one hour. These prices vary by region and storage used. 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.

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.

Topics

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. Go to <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.

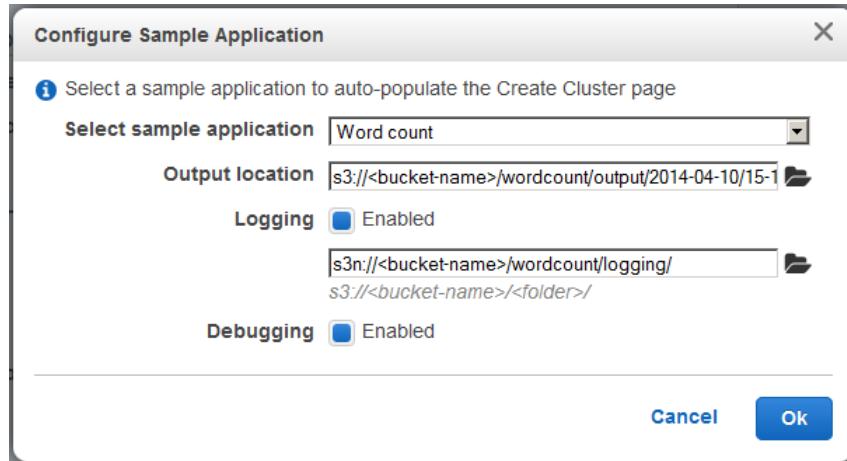
For console access, use your IAM user name and password to sign in to the [AWS Management Console](#) using the [IAM sign-in page](#). IAM lets you securely control access to AWS services and resources in your AWS account. For more information about creating access keys, see [How Do I Get Security Credentials?](#)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

Launch the Cluster

The next step is to launch the cluster. When you do, 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 the Amazon EMR cluster

1. Open the Amazon Elastic MapReduce console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Click **Create cluster**.
3. In the **Create Cluster** page, click **Configure sample application**.



Field	Action
Select sample application	Select Word count .
Output location	Type the path of an Amazon S3 bucket to store your output. If the bucket does not exist, the Amazon EMR console creates it for you.

Field	Action
Logging	<p>Choose Enabled.</p> <p>This determines whether Amazon EMR captures detailed log data to Amazon S3. When logging is enabled, you need to enter the output location.</p> <p>For more information, see View Log Files (p. 418).</p>
Debugging	<p>Check the box to enable debugging.</p> <p>This option creates a debug log index in Amazon SimpleDB (additional charges apply) to enable detailed debugging in the Amazon EMR console. You can only set this when the cluster is created. For more information about Amazon SimpleDB, go to the Amazon SimpleDB product description page.</p>

When you have finished configuring the sample Word Count application, click **OK**.

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

Software Configuration

Hadoop distribution

Amazon

Use Amazon's Hadoop distribution. [Learn more](#)

AMI version

2.4.2 (hadoop 1.0.3) - latest

Determines the base configuration of the instances in your cluster, including the Hadoop version. [Learn more](#)

MapR

Use MapR's Hadoop distribution. [Learn more](#)

Applications to be installed	Version	
Hive	0.11.0.1	
Pig	0.11.1.1	
Additional applications		<input style="width: 150px; border: 1px solid #ccc; height: 20px; margin-bottom: 5px;" type="text" value="Select an application"/> <input style="border: 1px solid #ccc; padding: 2px 10px; font-size: 0.9em; width: 150px;" type="button" value="Configure and add"/>

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. 149).</p>
AMI version	<p>Choose 2.4.2 (Hadoop 1.0.3).</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 (p. 51).</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, 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](#).

Hardware Configuration

Info Specify the [networking](#) and [hardware](#) configuration for your cluster. If you need more than 20 EC2 [Request Spot instances](#) (unused EC2 capacity) to save money.

Network	<input type="text" value="vpc-c1a3b3a3 (172.31.0.0/16) (default)"/>	Use a Virtual data or connect												
EC2 Subnet	<input type="text" value="No preference (random subnet)"/>	Create a Subnet												
<table style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 30%;">EC2 instance type</th> <th style="width: 20%;">Count</th> <th style="width: 50%; text-align: right;">Request spot</th> </tr> </thead> <tbody> <tr> <td>Master</td> <td><input type="text" value="1"/></td> <td style="text-align: right;"><input type="checkbox"/></td> </tr> <tr> <td>Core</td> <td><input type="text" value="2"/></td> <td style="text-align: right;"><input type="checkbox"/></td> </tr> <tr> <td>Task</td> <td><input type="text" value="0"/></td> <td style="text-align: right;"><input type="checkbox"/></td> </tr> </tbody> </table>		EC2 instance type	Count	Request spot	Master	<input type="text" value="1"/>	<input type="checkbox"/>	Core	<input type="text" value="2"/>	<input type="checkbox"/>	Task	<input type="text" value="0"/>	<input type="checkbox"/>	<p>The Master instances are the primary task nodes, and handle the work in the Hadoop Distributed File System (HDFS).</p> <p>Core instances are used to store the Hadoop Distributed File System (HDFS) data.</p> <p>Task instances are used to process the data in the HDFS.</p>
EC2 instance type	Count	Request spot												
Master	<input type="text" value="1"/>	<input type="checkbox"/>												
Core	<input type="text" value="2"/>	<input type="checkbox"/>												
Task	<input type="text" value="0"/>	<input type="checkbox"/>												

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 a Amazon VPC Subnet for the Cluster (Optional) (p. 137).</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 Elastic Compute Cloud User Guide</i>.</p>

Field	Action
Master	<p>Choose m1.small.</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.small (default), m1.large, m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cc1.4xlarge, cg1.4xlarge.</p> <p>This tutorial uses small instances for all nodes due to the light workload and to keep your costs low.</p> <p>For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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 Lower Costs with Spot Instances (Optional) (p. 37).</p>
Core	<p>Choose m1.small.</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.small (default), m1.large, m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cc1.4xlarge, cg1.4xlarge.</p> <p>This tutorial uses small instances for all nodes due to the light workload and to keep your costs low.</p> <p>For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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 Lower Costs with Spot Instances (Optional) (p. 37).</p>

Field	Action
Task	<p>Choose m1.small.</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. Valid types: m1.small (default), m1.large, m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cc1.4xlarge, cg1.4xlarge.</p> <p>For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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) (p. 37).</p>

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

Security and Access

EC2 key pair Use an existing key pair to SSH into the master node of the Amazon EC2 cluster as the user "hadoop". [Learn more](#)

IAM user access All other IAM users No other IAM users Control the visibility of this cluster to other IAM users. [Learn more](#)

IAM Roles

EC2 instance profile Allows EC2 instances in an EMR cluster to access other AWS services such as S3. [Learn more](#)

Field	Action
EC2 key pair	<p>Choose an Amazon EC2 key pair from the list.</p> <p>For more information, see Create an Amazon EC2 Key Pair and PEM File (p. 117).</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 Cluster (p. 446).</p>
IAM user access	<p>Choose No other IAM users.</p> <p> 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 (p. 119).</p>

Field	Action
EC2 instance profile	<p>You can proceed without choosing an instance profile. If you create a cluster without selecting a specific instance profile, one will be created for you.</p> <p>This controls application access to the Amazon EC2 instances in the cluster.</p> <p>For more information, see Configure IAM Roles for Amazon EMR (p. 125).</p>
EMR role	<p>Choose Proceed without role.</p> <p>Allows Amazon EMR to access other AWS services on your behalf.</p> <p>For more information, see Configure IAM Roles for Amazon EMR (p. 125).</p>

7. In the **Bootstrap Actions** section, there are no bootstrap actions necessary for this sample configuration.

Optional, 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 \(Optional\) \(p. 87\)](#).

8. In the **Steps** section, note the step that Amazon EMR configured for you by choosing the sample application. You can modify these settings to meet your needs. Complete the fields according to the following table.

Steps

i 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. You can submit up to 256 steps to a cluster. [Learn more](#)

Name	Action on failure	JAR S3 location	Arguments
Word count	Terminate cluster	/home/hadoop/contrib/streaming/hadoop-streaming.jar	-input s3n://elasticmapreduce/sample s/wordcount/input -output s3n://examples-bucket/wordcount/output/2013-11-04/12-32-33 -mapper s3n://elasticmapreduce/sample s/wordcount/wordSplitter.py -reducer aggregate

Add step

Auto-terminate Yes Automatically terminate cluster after the last step is completed.
 No Keep cluster running until you terminate it.

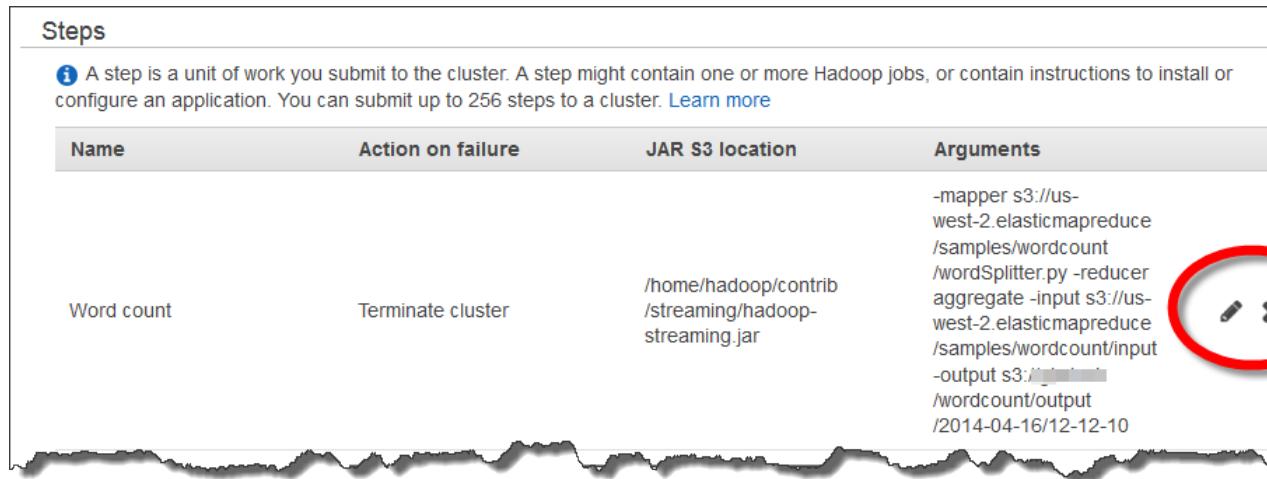
Field	Action
Add step	Leave this option set to Select a step . For more information, see Steps (p. 8) .

Field	Action
Auto-terminate	<p>Choose No.</p> <p>This determines what the cluster does after its last step. Yes means the cluster auto-terminates after the last step completes. No means the cluster runs until you manually terminate it.</p> <p>Remember to terminate the cluster when it is done so you do not continue to accrue charges on an idle cluster.</p>

Steps

i 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. You can submit up to 256 steps to a cluster. [Learn more](#)

Name	Action on failure	JAR S3 location	Arguments
Word count	Terminate cluster	/home/hadoop/contrib/streaming/hadoop-streaming.jar	-mapper s3://us-west-2.elasticmapreduce/samples/wordcount/wordSplitter.py -reducer aggregate -input s3://us-west-2.elasticmapreduce/samples/wordcount/input -output s3://[REDACTED]/wordcount/output/2014-04-16/12-12-10



The preceding image shows the step details section with a red circle around the edit step (pencil) and delete step (X) buttons. If you click the edit button, you can edit the following settings.

Field	Action
Mapper	Set this field to <code>s3n://elasticmapreduce/samples/wordcount/wordSplitter.py</code> .
Reducer	Set this field to <code>aggregate</code> .
Input S3 location	Set this field to <code>s3://elasticmapreduce/samples/wordcount/input</code> .
Output S3 location	Set this field to <code>s3://example-bucket/wordcount/output/2013-11-11/11-07-05</code> .
Arguments	Leave this field blank.
Action on failure	Set this field to <code>Terminate cluster</code> .

9. Review your configuration and if you are satisfied with the settings, click **Create Cluster**.
10. When the cluster starts, the console displays the **Cluster Details** page.

Next, Amazon EMR begins to count the words in the text of the CIA World Factbook, which is pre-configured in an Amazon S3 bucket as the input data for demonstration purposes. When the cluster is finished processing the data, Amazon EMR copies the word count results into the output Amazon S3 bucket that you chose in the previous steps.

Monitor the Cluster

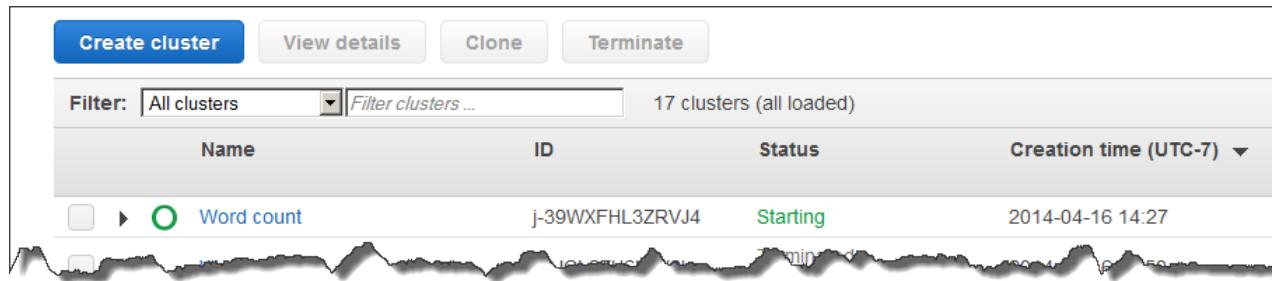
There are several ways to gain information about your cluster while it is running.

- Query Amazon EMR using the console, command-line interface (CLI), or programmatically.
- Amazon EMR automatically reports metrics about the cluster to CloudWatch. These metrics are provided free of charge. You can access them either through the CloudWatch interface or in the Amazon EMR console. For more information, see [Monitor Metrics with CloudWatch \(p. 423\)](#).
- Create an SSH tunnel to the master node and view the Hadoop web interfaces. Creating an SSH tunnel requires that you specify a value for **Amazon EC2 Key Pair** when you launch the cluster. For more information, see [Web Interfaces Hosted on the Master Node \(p. 450\)](#).
- Run a bootstrap action when you launch the cluster to install the Ganglia monitoring application. You can then create an SSH tunnel to view the Ganglia web interfaces. Creating an SSH tunnel requires that you specify a value for **Amazon EC2 Key Pair** when you launch the cluster. For more information, see [Monitor Performance with Ganglia \(p. 438\)](#).
- Use SSH to connect to the master node and browse the log files. Creating an SSH connection requires that you specify a value for **Amazon EC2 Key Pair** when you launch the cluster.
- View the archived log files on Amazon S3. This requires that you specify a value for **Amazon S3 Log Path** when you create the cluster. For more information, see [View Log Files \(p. 418\)](#).

In this tutorial, you'll monitor the cluster using the Amazon EMR console.

To monitor the cluster using the Amazon EMR console

1. Click **Cluster List** in the Amazon EMR console. This shows a list of clusters to which your account has access and the status of each. In this example, see a cluster in the **Starting** status. There are other possible status messages, for example **Running**, **Waiting**, **Terminated (All steps completed)**, **Terminated (User request)**, **Terminated with errors (Validation error)**, etc.



2. Click the details icon next to your cluster to see the cluster details page.

In this example, the cluster is in **Starting** status, provisioning the compute resources needed for the Word Count application. When the cluster finishes, it will sit idle in the **Waiting** status because we did not configure the cluster to terminate automatically. Remember to terminate your cluster to avoid additional charges.

The screenshot shows the 'Cluster Details' page for a cluster named 'Word count'. The cluster status is 'Starting'. The 'Summary' section includes fields for Master public DNS (empty), Tags (empty), ID (j-39WXFHL3ZRVJ4), Creation date (2014-04-16 14:27 (UTC-7)), Elapsed time (empty), Auto-terminate (Yes), and Termination protection (On Change). The 'Configuration Details' section shows AMI version (2.4.2), Hadoop distribution (Amazon 1.0.3), Applications (empty), and Log URI (empty). The 'Security/Network' section includes Availability zone (empty), Subnet ID (subnet-c39989a1), Key name (redacted), EC2 role (empty), and Visible to all users (None). The 'Hardware' section shows Master (Provisioning 1 m1.small), Core (Provisioning 2 m1.small), and Task (empty). A 'Monitoring' section is visible at the bottom.

3. The **Monitoring** section displays metrics about the cluster. These metrics are also reported to CloudWatch, and can also be viewed from the CloudWatch console. The charts track various cluster statistics over time, for example:

- Number of jobs the cluster is running
- Status of each node in the cluster
- Number of remaining map and reduce tasks
- Number of Amazon S3 and HDFS bytes read/written

Note

The statistics in the **Monitoring** section may take several minutes to populate. In addition, the Word Count sample application runs very quickly and may not generate highly detailed runtime information.

For more information about these metrics and how to interpret them, see [Monitor Metrics with CloudWatch \(p. 423\)](#).

4. In the **Software Configuration** section, you can see details about the software configuration of the cluster; for example:
- The AMI version of the nodes in the cluster
 - The Hadoop Distribution
 - The Log URI to store output logs
5. In the **Hardware Configuration** section, you can see details about the hardware configuration of the cluster, for example:
- The Availability Zone the cluster runs within

- The number of master, core, and task nodes including their instance sizes and status

In addition, you can control Termination Protection and resize the cluster.

6. In the **Steps** section, you can see details about each step in the cluster. In addition, you can add steps to the cluster.

Steps

Add step

Steps

Filter All steps 2 steps (all loaded)

ID	Name	Status	Creation time	Start time	Log files
s-2TE5T95WL7RGG	Word count	Completed	2013-11-04 12:58:25	2013-11-04 13:03:02	View logs
s-3RPPS1QMV2BCP	Setup hadoop debugging	Completed	2013-11-04 12:58:25	2013-11-04 13:02:31	View logs

In this example, you can see that the cluster had two steps: the Word count step (streaming step) and the Setup hadoop debugging step (script-runner step). If you enable debugging, Amazon EMR automatically adds the Setup hadoop debugging step to copy logs from the cluster to Amazon S3.

For more information about how steps are used in a cluster, see [Life Cycle of a Cluster \(p. 9\)](#).

- Click the arrow next to the Word Count step to see more information about the step.

Add step

Steps

Filter All steps 2 steps (all loaded)

ID	Name	Status	Creation time	Start time	Log files
s-2TE5T95WL7RGG	Word count	Completed	2013-11-04 12:58:25	2013-11-04 13:03:02	View logs

Jar location: /home/hadoop/contrib/streaming/hadoop-streaming.jar
Arguments: -input, s3n://elasticmapreduce/samples/wordcount/input, -output, s3n://examples-bucket/wordcount/output/2013-11-04
Action on failure: Terminate cluster

In this example, you can determine the following:

- The step uses a streaming JAR located on the cluster
- The input are files in an Amazon S3 location
- The output writes to an Amazon S3 location
- The mapper is a Python script named wordSplitter.py
- The final output compiles using the aggregate reducer
- The cluster will terminate if it encounters an error

7. Lastly, the **Bootstrap Actions** section lists the bootstrap actions run by the cluster, if any. In this example, the cluster has not run any bootstrap actions to initialize the cluster. For more information about how to use bootstrap actions in a cluster, see [Create Bootstrap Actions to Install Additional Software \(Optional\) \(p. 87\)](#).

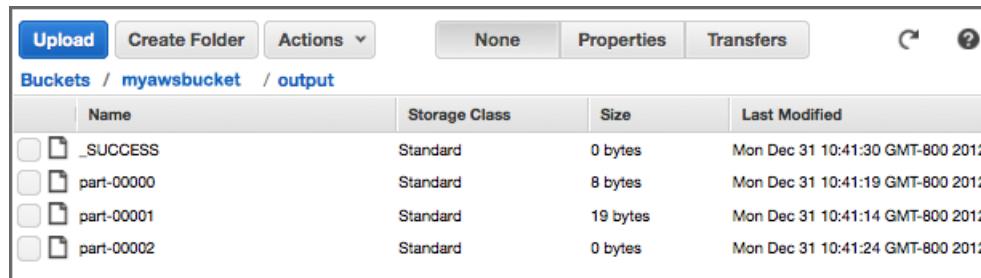
View the Results

After the cluster is complete, the results of the word frequency count are stored in the folder you specified on Amazon S3 when you launched the cluster.

To view the output of the cluster

1. From the Amazon S3 console, select the bucket you used for the output location when you configured the sample application.
2. Select the `output` folder, click **Actions**, and then select **Open**.

The results of running the cluster are stored in text files. The first file in the listing is an empty file titled according to the result of the cluster. In this case, it is titled "`_SUCCESS`" to indicate that the cluster succeeded.



Name	Storage Class	Size	Last Modified
<code>_SUCCESS</code>	Standard	0 bytes	Mon Dec 31 10:41:30 GMT-800 2012
<code>part-00000</code>	Standard	8 bytes	Mon Dec 31 10:41:19 GMT-800 2012
<code>part-00001</code>	Standard	19 bytes	Mon Dec 31 10:41:14 GMT-800 2012
<code>part-00002</code>	Standard	0 bytes	Mon Dec 31 10:41:24 GMT-800 2012

3. To download each file, right-click on it and select **Download**.
4. Open the text files using a text editor such as Notepad (Windows),TextEdit (Mac OS), or gEdit (Linux). In the output files, you should see a column that displays each word found in the source text followed by a column that displays the number of times that word was found.

The other output generated by the cluster are log files which detail the progress of the cluster. Viewing the log files can provide insight into the workings of the cluster, and can help you troubleshoot any problems that arise.

View the Debug Logs (Optional)

If you encounter any errors, you can use the debug logs to gather more information and troubleshoot the problem.

To view cluster logs using the console

1. Open the Amazon Elastic MapReduce console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. From the **Cluster List** page, click 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, click the **View Jobs** link to the right of the step.

Steps

Add step

Steps

Filter All steps Filter steps ... 2 steps (all loaded)

ID	Name	Status	Creation time	Start time	Log files
s-2TE5T95WL7RGG	Word count	Completed	2013-11-04 12:58:25	2013-11-04 13:03:02	View logs
s-3RPPS1QMV2BCP	Setup hadoop debugging	Completed	2013-11-04 12:58:25	2013-11-04 13:02:31	View logs

4. To view a list of the Hadoop tasks associated with a given job, click the **View Tasks** link to the right of the job.

Steps

Add step

Steps > Jobs

Jobs for: s-2TE5T95WL7RGG

Filter

Job	State	Start time
job_201311042101_0001	COMPLETED	2013-11-04 13:03:21

5. To view a list of the attempts a given task has run while trying to complete, click the **View Attempts** link to the right of the task.

Task	Type	State	Start time local time (UTC-8)
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

6. To view the logs generated by a task attempt, click the **stderr**, **stdout**, and **syslog** links to the right of the task attempt.

Attempt	Type	State	Log file
0	reduce	SUCCEEDED	control

Clean Up

Now that you've completed the tutorial, you should delete the Amazon S3 bucket that you created to ensure that your account does not accrue additional storage charges.

You do not need to delete the completed cluster. After a cluster ends, it terminates the associated EC2 instances and no longer accrues Amazon EMR maintenance 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.

Buckets with objects in them cannot be deleted. Before deleting a bucket, all objects within the bucket must be deleted.

You should also disable logging for your Amazon S3 bucket. Otherwise, logs might be written to your bucket immediately after you delete your bucket's objects.

To disable logging

1. Open the Amazon S3 console at <https://console.aws.amazon.com/s3/>.
2. Right-click your bucket and select **Properties**.
3. Click the **Logging** tab.
4. Deselect the **Enabled** check box to disable logging.

To delete an object

1. Open the Amazon S3 console at <https://console.aws.amazon.com/s3/>.
2. Click the bucket where the objects are stored.
3. Right-click the object to delete.

Tip

You can use the **SHIFT** and **CTRL** keys to select multiple objects and perform the same action on them simultaneously.

4. Click **Delete**.
5. Confirm the deletion when the console prompts you.

To delete a bucket, you must first delete all of the objects in it.

To delete a bucket

1. Right-click the bucket to delete.
2. Click **Delete**.
3. Confirm the deletion when the console prompts you.

You have now deleted your bucket and all its contents.

The next step is optional. It deletes two security groups created for you by Amazon EMR when you launched the cluster. You are not charged for security groups. If you are planning to explore Amazon EMR further, you should retain them.

To delete Amazon EMR security groups

1. In the Amazon EC2 console **Navigation** pane, click **Security Groups**.
2. In the **Security Groups** pane, click **ElasticMapReduce-slave**.
3. In the details pane for the ElasticMapReduce-slave security group, delete all rules that reference ElasticMapReduce. Click **Apply Rule Changes**.
4. In the right pane, select **ElasticMapReduce-master**.
5. In the details pane for the ElasticMapReduce-master security group, delete all rules that reference Amazon EMR. Click **Apply Rule Changes**.
6. With ElasticMapReduce-master security group still selected in the **Security Groups** pane, click **Delete**. Click **Yes, Delete** to confirm.
7. In the **Security Groups** pane, click **ElasticMapReduce-slave**, and then click **Delete**. Click **Yes, Delete** to confirm.

Where Do I Go From Here?

The following provides several possible next steps to learn more about Amazon EMR.

Next Steps

- Figure out how Amazon EMR can meet your computational needs. For some ideas of how to use Amazon EMR to solve business problems, see [What Can You Do with Amazon EMR? \(p. 2\)](#).
- Decide what type of cluster you'll use to implement your solution. For more information, see [Choose the Type of Cluster to Run \(p. 32\)](#).
- Choose an interface for working with Amazon EMR. The main interfaces are the Amazon EMR console, the command line interface (CLI), the REST APIs, and the SDKs published by Amazon Web Services.
- Start working with Amazon EMR. For more information, see [Manage Clusters \(p. 406\)](#).

Plan an Amazon EMR Cluster

Before you launch an Amazon EMR cluster, you need to figure out how you want the cluster configured. You'll make choices based on the type of data processing you want to do, the amount of data, speed, cost, and the software you want available on the cluster. The following topics walk you through the process of planning a cluster.

Some steps are optional. If you are new to Amazon EMR, we recommend that you also read through the optional steps to familiarize yourself with all that Amazon EMR can do.

Topics

- [Choose an AWS Region \(p. 29\)](#)
- [Choose the Type of Cluster to Run \(p. 32\)](#)
- [Choose the Number and Type of Virtual Servers \(p. 33\)](#)
- [Configure the Virtual Server Software \(p. 50\)](#)
- [Choose the Cluster Lifecycle: Long-Running or Transient \(p. 97\)](#)
- [Prepare Input Data \(Optional\) \(p. 98\)](#)
- [Prepare an Output Location \(Optional\) \(p. 112\)](#)
- [Configure Access to the Cluster \(p. 117\)](#)
- [Configure Logging and Debugging \(Optional\) \(p. 133\)](#)
- [Select a Amazon VPC Subnet for the Cluster \(Optional\) \(p. 137\)](#)
- [Tagging Amazon EMR Clusters \(p. 143\)](#)
- [Use Third Party Applications With Amazon EMR \(Optional\) \(p. 148\)](#)

Choose an AWS Region

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 which region to launch it into. 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 (Virginia) (us-east-1). If you access a resource from the command line interface or the application programming interface, the default region is always us-east-1. For more information, go to [Regions and Endpoints](#).

Some AWS features are available only in limited regions. For example, Cluster Compute instances are available only in the US-East (Northern 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 (Virginia)	us-east-1	US Standard	US East (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 West (Ireland)	eu-west-1	Ireland	EU West (Ireland)
Asia Pacific (Singapore)	ap-southeast-1	Singapore	Asia Pacific (Singapore)
Asia Pacific (Sydney)	ap-southeast-2	Sydney	Asia Pacific (Sydney)
Asia Pacific (Tokyo)	ap-northeast-1	Tokyo	Asia Pacific (Tokyo)
South America (Sao Paulo)	sa-east-1	Sao Paulo	South America (Sao Paulo)
AWS GovCloud (US)	us-gov-west-1	GovCloud	AWS GovCloud (US)

Note

To use the AWS GovCloud Region, contact your AWS business representative. You can't create an AWS GovCloud 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

- On the Amazon EMR console, expand the **Region Selector** on the navigation bar, and select a region.



Choose a Region using the CLI

To choose a region using the CLI

- Specify the region with the `--region` parameter, as in the following example. If the `--region` parameter is not specified, the cluster is created in the `us-east-1` region.

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- 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 \(p. 581\)](#).

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 Type of Cluster to Run

Amazon EMR offers several different types of clusters, each configured for a particular type of data processing. They vary in terms of the software installed on the cluster and the technologies available. You should select the one that best supports your intended use. The following sections describe the types of clusters you can launch in Amazon EMR.

If you are new to Amazon EMR and want to learn more about the different cluster types and how they work, you can run the sample applications available in the Amazon EMR console. These provide examples of a Hive cluster, a Pig cluster, two Custom JAR clusters and a Streaming cluster. The tutorial, [Get Started: Count Words with Amazon EMR \(p. 13\)](#), provides a detailed walkthrough of running a Streaming cluster.

Hive Cluster

Hive clusters are preloaded with Apache Hive, an open-source data warehouse and analytic package that runs on top of Hadoop. With this type of cluster you can load data onto your cluster and then query it using HiveQL, a SQL-like query language. Hive handles the process of converting your queries into distributed map-reduce applications. This cluster type is useful if you want to use the power of distributed processing to store and query your data without having to write a map-reduce application. This is also the cluster type you should select if you want to use Amazon EMR to move data in or out of DynamoDB. For more information, see [Analyze Data with Hive \(p. 202\)](#) and [Export, Import, Query, and Join Tables in DynamoDB Using Amazon EMR \(p. 364\)](#).

Custom JAR Cluster

A custom JAR cluster runs a Java map-reduce application that you have previously compiled into a JAR file and uploaded to Amazon S3. To create the JAR file, compile your Java code against the version of Hadoop you plan to launch in the cluster and submit Hadoop jobs using the Hadoop `JobClient` interface. Custom JAR clusters are preloaded with the Cascading open-source Java library. This library provides a query API, a query planner, and a job scheduler for creating and running Hadoop MapReduce applications. This type of cluster is best if you are a Hadoop developer and want the flexibility to create a custom map-reduce application. For more information, see [Process Data with a Custom JAR Cluster \(p. 192\)](#).

Streaming Cluster

A Streaming cluster runs mapper and reducer scripts that you have previously uploaded to Amazon S3. The scripts can be written using any of the following supported languages: Ruby, Perl, Python, PHP, R, Bash, or C++. Streaming clusters are preloaded with the Apache Streaming open-source library, and you can reference the functions in this library in your scripts. This type of cluster is best if you are a Hadoop developer and want to use the Streaming library to quickly develop a map-reduce application. For more information, see [Process Data with a Streaming Cluster \(p. 167\)](#).

Pig Cluster

Pig clusters are preloaded with Apache Pig, an open-source Apache library that runs on top of Hadoop. With this type of cluster, you can load data onto your cluster and then query it using Pig Latin, a SQL-like query language. Hive handles the process of converting your queries into distributed map-reduce applications. This cluster type is useful if you want to use the power of distributed processing to store and query your data without having to write a map-reduce application. For more information, see [Process Data with Pig \(p. 273\)](#).

HBase Cluster

HBase clusters are preloaded with Apache HBase, an open source, non-relational, distributed database modeled after Google's BigTable. 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 also integrates with Apache Hive, enabling SQL-like queries over HBase tables, joins with Hive-based tables, and support for Java Database Connectivity (JDBC). This cluster type is useful for creating a data warehouse. For more information, see [Store Data with HBase \(p. 289\)](#)

Choose the Number and Type of Virtual Servers

One of the most important considerations when launching a cluster is the number and type of virtual servers 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 virtual server 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 [Virtual Server Configurations \(p. 36\)](#). 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 virtual server 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 \(p. 35\)](#). 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 \(p. 406\)](#). 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, multiple 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 2833 GB of space available to HDFS: (10 nodes x 850 GB per node) / replication factor of 3.

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 Virtual Servers

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 [Virtual Server Configurations \(p. 36\)](#)
- 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 \(p. 35\)](#)
- [Virtual Server Configurations \(p. 36\)](#)
- [Ensure Capacity with Reserved Instances \(Optional\) \(p. 36\)](#)
- [Lower Costs with Spot Instances \(Optional\) \(p. 37\)](#)

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 roles analogous to the master and slave nodes of Hadoop. There are three types of instance groups: master, core, and task.

Each Amazon EMR cluster includes one master instance group that contains one master node, a core instance group containing one or more core nodes, and an optional task instance group, 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 \(p. 464\)](#).

Master Instance Group

The master instance group 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. 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 publishes to the web server running on the master node. For more information, see [View Log Files \(p. 418\)](#).

As the cluster progresses, each core and task node processes its data, transfers the data back to Amazon S3, and provides status metadata to the master node. The master node runs the NameNode and Job-Tracker daemons. In the case of a single-node cluster, the master node also runs the TaskTracker and DataNode daemons as well.

Caution

The instance controller 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 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.

The EC2 instances you assign as core nodes are capacity that must be allotted for the entire cluster run. Because core nodes store data, you can't remove them from a cluster. However, you can add more core nodes to a running cluster. Core nodes run both the DataNodes and TaskTracker Hadoop daemons.

Caution

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

For more information about core instance groups, see [Resize a Running Cluster \(p. 464\)](#).

Task Instance Group

The task instance group contains all of the task nodes in a cluster. The task instance group is optional. You can add it when you start the cluster or add a task instance group to a cluster in progress.

Task nodes are managed by the master node. While a cluster is running you can increase and decrease the number of task nodes. Because they don't store data and can be added and removed from a cluster,

you can use task nodes 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.

It can be beneficial to configure task nodes as Spot Instances because of the potential cost savings. For more information, see [Lower Costs with Spot Instances \(Optional\) \(p. 37\)](#).

For more information about task instance groups, see [Resize a Running Cluster \(p. 464\)](#).

Virtual Server 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 Family	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 cc1.4xlarge 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
GPU instances	g2.2xlarge cg1.4xlarge

For more information on these instance types and families see [Instance Type Details](#).

Note

Amazon EMR does not support micro instances at this time.

Note

I2, G2, CC1, 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).

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](#)

Lower Costs with Spot Instances (Optional)

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 Elastic Compute Cloud User Guide*.

The following video describes how Spot Instances work in Amazon Elastic MapReduce (Amazon EMR) and walks you through the process of launching a cluster on Spot Instances from the Amazon EMR console: [Using Spot Instances with Amazon ElasticMapReduce](#).

Additional video instruction includes:

- [Amazon EC2 - Deciding on your Spot Bidding Strategy](#), describes strategies to use when setting a bid price for Spot Instances.
- [Amazon EC2 - Managing Interruptions for Spot Instance Workloads](#), describes ways to handle Spot Instance termination.

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 you expect the cluster to be short-lived and not critical.

Note

Spot Instances are not recommended for clusters that are time-critical or which need guaranteed capacity. These clusters should be launched using on-demand instance groups.

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 a task instance group 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 group 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.

For more information, see the following topics:

Topics

- [Choose What to Launch as Spot Instances \(p. 38\)](#)
- [Spot Instance Pricing in Amazon EMR \(p. 40\)](#)
- [Availability Zones and Regions \(p. 40\)](#)
- [Launch Spot Instances in a Cluster \(p. 41\)](#)
- [Change the Number of Spot Instances in a Cluster \(p. 47\)](#)
- [Troubleshoot Spot Instances \(p. 49\)](#)

Choose What to Launch as Spot Instances

When you launch a cluster in Amazon Elastic MapReduce (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
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 will 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. Because termination of core nodes can result in data loss and possible termination of the cluster, you would 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 will not launch if there are only five nodes available at or below your bid price. In this case, Amazon EMR will continue to wait until all six core nodes are available at your Spot Price until it is successful or 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 remove core nodes from a running cluster.

Task Instance Group 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 the task instance group as Spot Instances, Amazon EMR will provision 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 will launch the instance group with five nodes, adding the sixth later if it can.

Launching the task instance group 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 the instance group as needed to handle peak traffic or to speed up data processing.

You can add and remove Spot Instance task nodes from a running cluster.

Spot Instance Pricing in Amazon EMR

There are two components in Amazon Elastic MapReduce (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 look up the current Spot Price and the on-demand price for instances on the [Amazon EC2 Pricing page](#).

Availability Zones and Regions

When you launch a cluster, you have the option to specify a region and an Availability Zone within that region.

If you do not specify an Availability Zone when you launch a cluster, Amazon Elastic MapReduce (Amazon EMR) selects the Availability Zone with lowest Spot Instance pricing and the largest available capacity of EC2 instance types specified for your core instance group, and then launches the master, core, and task instance groups in that Availability Zone.

Because of fluctuating Spot Prices between Availability Zones, selecting the Availability Zone with the lowest initial price (or allowing Amazon EMR to select it for you) 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 Amazon 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.

If you specify an Availability Zone for the cluster, Amazon EMR launches all of the instance groups in that Availability Zone.

All instance groups in a cluster are launched into a single Availability Zone, regardless of whether they are on-demand or 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.

Note

Selecting the Availability Zone is currently not available in the Amazon EMR console. Amazon EMR assigns an Availability Zone to clusters launched from the Amazon EMR console as described above.

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 additional information such as the market type and the bid price.

Amazon EMR Console

To launch an entire cluster on Spot Instances

	EC2 instance type	Count	Request spot	Bid price
Master	m1.xlarge	1	<input checked="" type="checkbox"/>	.35
Core	m1.small	2	<input checked="" type="checkbox"/>	.20
Task	m1.small	3	<input checked="" type="checkbox"/>	.20

1. Open the Amazon Elastic MapReduce console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Click **Create cluster**.
3. In the **Hardware Configuration** section, to run the master node as a Spot Instance, select **Request Spot** in the **Master** row and enter the maximum hourly rate you are willing to pay per instance in the **Bid price** field that appears. You can look up the current Spot Price for instances on the [Amazon EC2 Pricing page](#). 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, select the **Request Spot Instance** check box in the **Core** row and enter the maximum hourly rate you are willing to pay per instance in the **Spot Bid Price** text box that appears.
5. To run the task nodes as Spot Instances, select the **Request Spot Instance** check box in the **Task** row and enter the maximum hourly rate you are willing to pay per instance in the **Spot Bid Price** text box that appears.
6. Proceed to create the cluster as described in [Plan an Amazon EMR Cluster \(p. 29\)](#).

CLI

To launch an entire cluster on Spot Instances using the Ruby 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, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- 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.small --instance-count 4 --bid-
price 0.03 \
--instance-group task --instance-type cl.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.small --instance-count 4 --bid-
price 0.03 --instance-group task --instance-type cl.medium --instance-count
2 --bid-price 0.10
```

Java SDK

To launch an entire cluster on Spot Instances

- 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("m1.large")
    .withMarket("SPOT")
    .withBidPrice("0.25");

InstanceGroupConfig instanceGroupConfigCore = new InstanceGroupConfig()
    .withInstanceCount(4)
    .withInstanceRole("CORE")
    .withInstanceType("m1.small")
    .withMarket("SPOT")
    .withBidPrice("0.03");

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

API

To launch an entire cluster on Spot Instances

- To specify that an instance group should be launched as Spot Instances, set the *BidPrice* and *Market* properties of the `InstanceGroupDetail` members of the `InstanceGroupDetailList`. The code below shows how to define master, core, and task instance groups that run as Spot Instances.

Example Sample Request

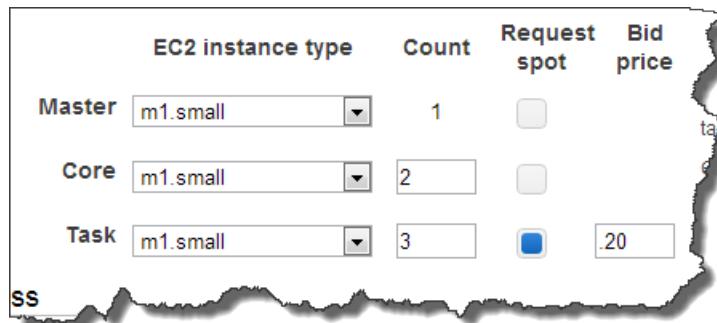
```
https://elasticmapreduce.amazonaws.com?Operation=RunJobFlow
&Name=MyJobFlowName
&LogUri=s3n%3A%2Fmybucket%2Fsubdir
&Instances.MasterInstanceType=m1.large
&Instances.SlaveInstanceType=m1.small
&Instances.InstanceCount=4
&Instances.Ec2KeyName=myec2keyname
&Instances.Placement.AvailabilityZone=us-east-1a
&Instances.KeepJobFlowAliveWhenNoSteps=true
&Instances.TerminationProtected=true
&Instances.InstanceGroups.member.1.InstanceRole=MASTER
&Instances.InstanceGroups.member.1.Market=SPOT
&Instances.InstanceGroups.member.1.BidPrice=.25
&Instances.InstanceGroups.member.2.InstanceRole=CORE
&Instances.InstanceGroups.member.2.Market=SPOT
&Instances.InstanceGroups.member.2.BidPrice=.03
&Instances.InstanceGroups.member.3.InstanceRole=TASK
&Instances.InstanceGroups.member.3.Market=SPOT
&Instances.InstanceGroups.member.3.BidPrice=.03
&Steps.member.1.Name=MyStepName
&Steps.member.1.ActionOnFailure=CONTINUE
&Steps.member.1.HadoopJarStep.Jar=MyJarFile
&Steps.member.1.HadoopJarStep.MainClass=MyMainClass
&Steps.member.1.HadoopJarStep.Args.member.1=arg1
&Steps.member.1.HadoopJarStep.Args.member.2=arg2
&AuthParams
```

Example Sample Response

```
<RunJobFlowResponse xmlns="http://elasticmapreduce.amazonaws.com/doc/2009-03-31">
  <RunJobFlowResult>
    <JobFlowId>
      j-3UN6WX5RRO2AG
    </JobFlowId>
  </RunJobFlowResult>
  <ResponseMetadata>
    <RequestId>
      8296d8b8-ed85-11dd-9877-6fad448a8419
    </RequestId>
  </ResponseMetadata>
</RunJobFlowResponse>
```

Amazon EMR Console

To launch only the task instance group on Spot Instances



1. Open the Amazon Elastic MapReduce console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Click **Create cluster**.
3. In the **Hardware Configuration** section, to run only the task nodes as Spot Instances, select **Request Spot Instance** in the **Task** row and enter the maximum hourly rate you are willing to pay per instance in the **Bid price** field that appears. You can look up the current Spot Price for instances on the [Amazon EC2 Pricing page](#). In most cases, you will want to enter a price higher than the current Spot Price.
4. Proceed to create the cluster as described in [Plan an Amazon EMR Cluster \(p. 29\)](#).

CLI

To launch only the task instance group on Spot Instances using the 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 only the task instance group is running as Spot Instances. The code below will launch a cluster even if the request for Spot Instances can't be fulfilled. In that case, Amazon EMR will add 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, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- 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.small \
--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
```

Java SDK

To launch only the task instance group on Spot Instances

- 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 will run as Spot Instances.

```
InstanceGroupConfig instanceGroupConfigMaster = new InstanceGroupConfig()
.withInstanceCount(1)
.withInstanceRole("MASTER")
.withInstanceType("m1.large")

InstanceGroupConfig instanceGroupConfigCore = new InstanceGroupConfig()
.withInstanceCount(4)
.withInstanceRole("CORE")
.withInstanceType("m1.small")

InstanceGroupConfig instanceGroupConfig = new InstanceGroupConfig()
.withInstanceCount(10)
.withInstanceRole("TASK")
.withInstanceType("m1.large")
.withMarket("SPOT")
.withBidPrice("0.35");
```

API

To launch only the task instance group on Spot Instances

- To specify that an instance group should be launched as Spot Instances, set the *BidPrice* and *Market* properties of the `TASK` `InstanceGroupDetail` member of the `InstanceGroupDetailList`. The following code shows how to define only the task instance group to run as Spot Instances.

The following example shows how to format the request.

Example Sample Request

```
https://elasticmapreduce.amazonaws.com?Operation=RunJobFlow
&Name=MyJobFlowName
&LogUri=s3n%3A%2F%2Fmybucket%2Fsubdir
&Instances.MasterInstanceType=m1.large
&Instances.SlaveInstanceType=m1.small
&Instances.InstanceCount=4
&Instances.Ec2KeyName=myec2keyname
&Instances.Placement.AvailabilityZone=us-east-1a
&Instances.KeepJobFlowAliveWhenNoSteps=true
&Instances.TerminationProtected=true
  &Instances.InstanceGroups.member.1.InstanceRole=MASTER
&Instances.InstanceGroups.member.2.InstanceRole=CORE
&Instances.InstanceGroups.member.3.InstanceRole=TASK
&Instances.InstanceGroups.member.3.Market=SPOT
  &Instances.InstanceGroups.member.3.BidPrice=.03
&Steps.member.1.Name=MyStepName
&Steps.member.1.ActionOnFailure=CONTINUE
&Steps.member.1.HadoopJarStep.Jar=MyJarFile
&Steps.member.1.HadoopJarStep.MainClass=MyMainClass
&Steps.member.1.HadoopJarStep.Args.member.1=arg1
&Steps.member.1.HadoopJarStep.Args.member.2=arg2
&AuthParams
```

Example Sample Response

```
<RunJobFlowResponse xmlns="http://elasticmapreduce.amazonaws.com/doc/2009-03-31">
  <RunJobFlowResult>
    <JobFlowId>
      j-3UN6WX5RRO2AG
    </JobFlowId>
  </RunJobFlowResult>
  <ResponseMetadata>
    <RequestId>
      8296d8b8-ed85-11dd-9877-6fad448a8419
    </RequestId>
  </ResponseMetadata>
</RunJobFlowResponse>
```

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, you cannot change the number of instances in the master instance group, it always has one instance. To prevent data loss, you can add, but not remove core nodes from an instance group. Task nodes do not store state, and so they can be added or removed from a running cluster.

You can only define an instance group as running as Spot Instances when it is created, so for example, if you launched the core instance group as on-demand, you would not be able to change its market type to Spot Instances later.

Note

It's possible to automatically modify the number of slave nodes in a cluster between cluster steps. You just include a predefined step in your workflow that changes the number of requested Spot Instances.

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --jobflow JobFlowId \
--jar s3://elasticmapreduce/libs/resize-job-flow/0.1/resize-job-flow.jar \
\
--args --modify-instance-group,task,--instance-count,4
```

- Windows users:

```
ruby elastic-mapreduce --jobflow JobFlowId --jar s3://elasticmapre
duce/libs/resize-job-flow/0.1/resize-job-flow.jar --args --modify-in
stance-group,task,--instance-count,4
```

The following examples show how to add task and core instance groups to a running cluster by increasing the number of instances in an instance group. For task nodes, you can also decrease the number of instances. The procedure is the same as shown in the examples that follow, but instead of specifying a larger number of instances than is currently running, you would specify a smaller number.

Note

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

Note

Adding nodes to a running cluster is not currently supported in the Amazon EMR console.

CLI

To add Spot Instances to a running cluster

- You can use `--add-instance-group` to add a task instance group of Spot Instances to a running cluster.

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- 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
```

CLI

To increase or decrease the number of Spot Instances in a running cluster

- You can change the number of requested Spot Instances in a running cluster by calling the `--modify-instance-group` and `--instance-count` on the command line. 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 will remove all Spot Instances.

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- 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
```

This will change the number of requested `TASK InstanceGroup` instances to 5.

Java SDK

To add Spot Instances to a running cluster

- 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 will run as Spot Instances.

When you make the call to modify the instance group, pass this object instance in.

```
InstanceGroupConfig instanceGroupConfig = new InstanceGroupConfig()  
.withInstanceCount(10)
```

```
.withInstanceRole("TASK")
.withInstanceType("m1.large")
.withMarket("SPOT")
.withBidPrice("0.35");
```

API

To add Spot Instances to a running cluster

- The following sample request increases the number of task nodes in the task instance group to eight and requests that they be launched as Spot Instances with an hourly bid price of .35.

The following is an example of the request you would send in to Amazon EMR.

Sample Request

```
https://elasticmapreduce.amazonaws.com?Operation=AddInstanceGroups
&InstanceGroups.member.1.InstanceGroupId=i-3UN6WX5RRO2AG
&InstanceGroups.member.1.InstanceRequestCount=8
&InstanceGroups.member.1.InstanceRole=TASK
&InstanceGroups.member.1.Market=SPOT
&InstanceGroups.member.1.BidPrice=.35
&AuthParams
```

Sample Response

```
<ModifyInstanceGroupsResponse xmlns="http://elasticmapreduce.amazonaws.com/doc/2009-03-31">
  <ResponseMetadata>
    <RequestId>
      2690d7eb-ed86-11dd-9877-6fad448a8419
    </RequestId>
  </ResponseMetadata>
</ModifyInstanceGroupsResponse>
```

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, go to [Troubleshoot a Cluster \(p. 481\)](#).

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 will not be 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 will tell 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 will likely 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 \(p. 481\)](#).

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 Virtual Server Software

Amazon EMR uses an Amazon Machine Image (AMI) to install Linux, Hadoop, and other software on the virtual servers 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 installed on the cluster, you can use bootstrap actions to install additional software and to change the configuration of applications on the cluster. Bootstrap actions are scripts that are run on the virtual servers 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 a Machine Image \(p. 51\)](#)

- [Choose a Version of Hadoop \(p. 79\)](#)
- [Create Bootstrap Actions to Install Additional Software \(Optional\) \(p. 87\)](#)

Choose a Machine Image

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.

For general information about AMIs, go to [Amazon Machine Images](#) in the *Amazon Elastic Compute Cloud User Guide*. For more information about the software versions included in the Amazon EMR AMIs, see [AMI Versions Supported in Amazon EMR \(p. 56\)](#).

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

Specifying the AMI version during cluster creation is optional; if you do not provide an AMI-version parameter, and you are using the CLI, your clusters will run on the most recent AMI version. This means you always have the latest software running on your clusters, but you must ensure that your application will work with new changes as they are released.

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.

Topics

- [AMI Version Numbers \(p. 51\)](#)
- [Default AMI and Hadoop Versions \(p. 52\)](#)
- [Specifying the AMI Version for a New Cluster \(p. 52\)](#)
- [Check the AMI Version of a Running Cluster \(p. 54\)](#)
- [Amazon EMR AMIs and Hadoop Versions \(p. 56\)](#)
- [Amazon EMR AMI Deprecation \(p. 56\)](#)
- [AMI Versions Supported in Amazon EMR \(p. 56\)](#)

AMI Version Numbers

AMI version numbers are composed of three parts `major-version.minor-version.patch`. The [current version of the Amazon EMR CLI](#) provides three ways to specify which version of the AMI to use to launch your cluster.

- **Fully specified**—If you specify the AMI version using all three parts (e.g. `--ami-version 2.0.1`) your cluster will be launched on exactly that version. The preceding example would launch a cluster using AMI 2.0.1. This is useful if you are running an application that depends on a specific AMI version and you want to ensure that AMI version is the one used to launch your clusters. The downside is you will not benefit from new features and improvements that are released on subsequent AMIs.
- **Major-minor version specified**—If you specify just the major and minor version for the AMI (e.g. `--ami-version 2.0`), your cluster will be launched on the AMI that matches those specifications and which has the latest patches. The preceding example would launch a cluster using AMI 2.0.4, since .4 is the latest patch for the 2.0 AMI series that is not deprecated. This scenario ensures a measure of stability in the AMI version, while ensuring that you receive the benefits of new patches and bug releases.

- **Latest version specified**—If you use the keyword `latest` instead of a version number for the AMI (e.g. `--ami-version latest`), the cluster is launched with the latest version available. This is the most dynamic way to run your clusters, as AMIs are updated regularly. This configuration is best for prototyping and testing, and is not recommended for production environments.

Default AMI and Hadoop Versions

If you don't specify the AMI for the cluster, Amazon EMR launches your cluster with the default version. The default versions returned depend on the interface you use to launch the cluster.

Note

The default AMI is unavailable in the Asia Pacific (Sydney) Region. Instead, use the `--ami-version latest` keyword to specify the latest AMI for that region instead.

Interface	Default AMI and Hadoop versions
Amazon EMR console	latest AMI and Hadoop versions
API	AMI 1.0, Hadoop 0.18
SDK	AMI 1.0, Hadoop 0.18
CLI (version 2012-07-30) and later	latest AMI and Hadoop versions
CLI (versions 2011-12-08 to 2012-07-09)	AMI 2.1.3, Hadoop 0.20.205
CLI (version 2011-12-11 and earlier)	AMI 1.0, Hadoop 0.18

To determine which version of the Amazon EMR CLI you have installed

- In the directory where you installed the CLI, run the following from the command line:
 - Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --version
```

- Windows users:

```
ruby elastic-mapreduce --version
```

Specifying the AMI Version for a New Cluster

You can specify which AMI version a new cluster should use when you create it. For details about the default configuration and applications available on AMI versions, see [AMI Versions Supported in Amazon EMR \(p. 56\)](#).

To specify an AMI version using the CLI

- 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.

The following example specifies the AMI completely and will launch a cluster on AMI 2.4.2.

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --alive --name "Static AMI Version" \  
--ami-version 2.4.2 \  
--num-instances 5 --instance-type m1.large
```

- Windows users:

```
ruby elastic-mapreduce --create --alive --name "Static AMI Version" --ami-  
version 2.4.2 --num-instances 5 --instance-type m1.small
```

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 which has the latest patches. This example would launch a cluster using AMI 2.4.2, since .2 is the latest patch for the 2.4 AMI series.

- 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.small
```

- Windows users:

```
ruby elastic-mapreduce --create --alive --name "Major-Minor AMI Version" \  
--ami-version 2.4 --num-instances 5 --instance-type m1.small
```

The following example specifies that the cluster should be launched with the most current version available.

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --alive --name "Latest AMI Version" \  
--ami-version latest \  
--num-instances 5 --instance-type m1.small
```

- Windows users:

```
ruby elastic-mapreduce --create --alive --name "Latest AMI Version" --ami-  
version latest --num-instances 5 --instance-type m1.small
```

To specify an AMI version using the API

- When creating a cluster using the API, add the `AmiVersion` and the `HadoopVersion` parameters to the request string, as shown in the following example. If you do not specify these parameters, Amazon EMR will create the cluster using the version 1.0 AMI and Hadoop 0.20. For more information, go to [RunJobFlow](#) in the *Amazon Elastic MapReduce API Reference*.

```
https://elasticmapreduce.amazonaws.com?Operation=RunJobFlow
&Name=MyJobFlowName
&LogUri=s3n%3A%2F%2Fmybucket%2Fsubdir
&AmiVersion=1.0
&HadoopVersion=0.20
&Instances.MasterInstanceType=m1.small
&Instances.SlaveInstanceType=m1.small
&Instances.InstanceCount=4
&Instances.Ec2KeyName=myec2keyname
&Instances.Placement.AvailabilityZone=us-east-1a
&Instances.KeepJobFlowAliveWhenNoSteps=true
&Steps.member.1.Name=MyStepName
&Steps.member.1.ActionOnFailure=CONTINUE
&Steps.member.1.HadoopJarStep.Jar=MyJarFile
&Steps.member.1.HadoopJarStep.MainClass=MyMainClass
&Steps.member.1.HadoopJarStep.Args.member.1=arg1
&Steps.member.1.HadoopJarStep.Args.member.2=arg2
&AuthParams
```

Check the AMI Version of a Running Cluster

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 check the current AMI version using the console

- Sign in to the AWS Management Console and open the Amazon Elastic MapReduce console at <https://console.aws.amazon.com/elasticmapreduce/>.
- Click on a cluster. The **Ami Version** and other details about the cluster are displayed in the **Summary** pane.

To check the current AMI version using the CLI

- Use the `--describe` parameter to retrieve the AMI version on a cluster. In the following example `JobFlowID` is the identifier of the cluster. The AMI version will be returned along with other information about the cluster.

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --describe --jobflow JobFlowID
```

- Windows users:

```
ruby elastic-mapreduce --describe --jobflow JobFlowID
```

To check 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

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 as described in [Hadoop Configuration Reference \(p. 515\)](#). 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 \(p. 56\)](#).

Amazon EMR AMI Deprecation

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.

Before an AMI is deprecated, the Amazon EMR team will send out an announcement specifying the date on which the AMI version will no longer be supported.

AMI Versions Supported in Amazon EMR

Amazon EMR supports the AMI versions listed in the following tables. You can specify the AMI version to use when you create a cluster. If you do not specify an AMI version, Amazon EMR creates the cluster using the default AMI version. For information about default AMI configurations, see [Default AMI and Hadoop Versions \(p. 52\)](#).

Hadoop 2 AMI Versions

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. 80)• Hive 0.11.0.2 (p. 219)• Pig 0.12 (p. 275)• Hbase 0.94.18 (p. 290)• Impala 1.2.4 (p. 260)• Mahout 0.9 (p. 86)• 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"> Added Enhanced Networking for C3, R3, and I2 instance types. For more information, see the Enhanced Networking section in the AWS EC2 User Guide. Updates to Java version to 7u65. For more information, go to JDK 7 Update 65 Release. Adds support for AWS SDK 1.7.8 for all components except Impala. Scalability of Cloudwatch metrics support for EMR has been improved. Enabled <code>fuse_efs</code>. For more information, see the AWS FSx for Lustre User Guide. 	

AMI Version	Includes	Notes	Release Date
		<p>MountableHDFS website.</p> <ul style="list-style-type: none"> • Step log location: previously steps were assigned a number based on their execution sequence. Now, step locations will be provided under their <code>stepId</code>. For example, the new path for the <code>stepId</code> LogPathStep is <p>Log paths can either be local to the cluster or Amazon S3 paths. To get the <code>stepId</code>, you can use the <code>ListSteps</code>.</p> <p>Major Bug Fixes</p>	

AMI Version	Includes	Notes	Release Date
		<ul style="list-style-type: none"> • Fixes 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 mountpoints, which resulted in issues using ephemeral drives on certain instances. • Fixed an issue in ResourceManager, which prevented users from accessing user 	

AMI Version	Includes	Notes	Release Date
		<p>interfaces on localhost.</p> <ul style="list-style-type: none"> • Fixed a bug in JobHistory that may have prevented storage of JobHistory logs in S3 buckets. • Includes the following Hadoop patches: HDFS-6701, HDFS-6460, HADOOP-10456, HDFS-6268, MAPREDUCE-310 • Backport of YARN-1864 to Hadoop 2. • Fixed a performance regression in Hive. • Hive is compiled against JDK 1.7 • Includes the following Hive patches: HIVE-6938, HIVE-7429. • Fixed several Hbase bugs. • The connector for Amazon Kinesis for EMR now supports all regions where Kinesis is available. 	

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. 80)• Hive 0.11.0.2 (p. 219)• Pig 0.12 (p. 275)• Hbase 0.94.18 (p. 290)• Impala 1.2.4 (p. 260)• Mahout 0.9 (p. 86)• 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

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. Click the respective links for more information: Hadoop 2.4.0 (p. 80), Pig 0.12 (p. 275), Impala 1.2.4 (p. 260), Hbase 0.94.18 (p. 290), Mahout 0.9 (p. 86), and Ruby 2.0. • Enables Amazon S3 Server Side Encryption with Hadoop. For more information, see Create a cluster with Amazon S3 Server Side Encryption Enabled (p. 106). • Updates to Java version to 7u60 (early 	

AMI Version	Includes	Notes	Release Date
		<p>access release). For more information, go to JDK 7 Update 60 Early Access Release.</p> <ul style="list-style-type: none">• Updates Jetty to version 6.1.26.emr that fixes Hadoop MapReduce issue MAPREDUCE-298.• Changed the log level for Hive UDAFPercentile to DEBUG. <p>Major Bug Fixes</p>	

AMI Version	Includes	Notes	Release Date
		<ul style="list-style-type: none"> Fixes an issue encountered when no log-uri is specified at cluster creation. Fixes version utility to accurately display Amazon Hadoop Distribution version. Fixes Hadoop to accept HADOOP_HEAPSIZE and HADOOP_HEAPSIZE memory setting values. <code>YARN_HEAPSIZE</code> is replaced with YARN_HEAPSIZE, YARN_HEAPSIZE, and YARN_HEAPSIZE to allow more granularity when configuring. For more information, see Configuration of hadoopconf(59) Added memory setting, HDFS_HEAPSIZE Fixes an issue encountered with hdfs -get when used with an 	

AMI Version	Includes	Notes	Release Date
		<p>Amazon S3 path.</p> <ul style="list-style-type: none">• 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.	

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. 81) • Hive 0.11.0.2 (p. 219) • Pig 0.11.1.1 (p. 275) • Hbase 0.94.7 (p. 289) • Impala 1.2.1 (p. 260) • 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.319). • Fixes an issue in the <code>yaml-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

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. 81)• Hive 0.11.0.2 (p. 219)• Pig 0.11.1.1 (p. 275)• Hbase 0.94.7 (p. 289)• Impala 1.2.1 (p. 260)• 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

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 update to Redshift 2.5 requires you to update and redistribute any of your Java code that was previously generated by</p>	

AMI Version	Includes	Notes	Release Date
		the <code>pdc</code> tool.	
3.0.2	<ul style="list-style-type: none"> • Amazon Linux version 2013.03 • Hadoop 2.2.0 (p. 81) • Hive 0.11.0.2 (p. 219) • Pig 0.11.1.1 (p. 275) • Hbase 0.94.7 (p. 289) • Impala 1.2.1 (p. 260) • 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 Analyze Data with Impala (p.247). • Changes the <code>uploadMultiParts</code> 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. 81) • Hive 0.11.0.1 (p. 219) • Pig 0.11.1.1 (p. 275) • Hbase 0.94.7 (p. 289) • Impala 1.2.1 (p. 260) • 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

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. 81) • Hive 0.11.0.1 (p. 219) • Pig 0.11.1.1 (p. 275) • Hbase 0.94.7 (p. 289) • Impala 1.2.1 (p. 260) • 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. 79). • 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

Hadoop 1 and Earlier AMIs

AMI Version	Description	Release Date
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
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-1, us-west-2, us-west-1, eu-west-1, ap-southeast-1, ap-southeast-2, ap-northeast-1, and sa-east-1 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 The upgrade to Protobuf 2.5 requires you to regenerate and recompile any of your Java code that was previously generated by the protoc tool.</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

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 regenerate and recompile any of your Java code that was previously generated by the protoc tool.</p>	3 January 2014
2.4.2	<p>Same as the previous AMI version, with the following additions:</p> <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
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

AMI Version	Description	Release Date
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
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 --s3ServerSideEncryption option to the S3DistCp tool. For more information, see S3DistCp Options (p. 356). 	26 April 2013
2.3.4	Deprecated	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

AMI Version	Description	Release Date
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. 125). 	20 December 2012
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	Deprecated	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 <code>fs.s3n.blockSize</code>. For more information, see Configure Multipart Upload for Amazon S3 (p. 104). 	30 August 2012

AMI Version	Description	Release Date
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. 104). 	30 August 2012
2.1.3	<p>Same as 2.1.2, with the following additions:</p> <ul style="list-style-type: none"> • Fixes issues in HBase. 	6 August 2012
2.1.2	<p>Same as 2.1.1, with the following additions:</p> <ul style="list-style-type: none"> • Support for CloudWatch metrics when using MapR. <p>Improve reliability of reporting metrics to CloudWatch.</p>	6 August 2012
2.1.1	<p>Same as 2.1.0, with the following additions:</p> <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

AMI Version	Description	Release Date
2.1.0	<p>Same as AMI 2.0.5, with the following additions:</p> <ul style="list-style-type: none"> Supports launching HBase clusters. For more information see Store Data with HBase (p. 289). Supports running MapR Edition M3 and Edition M5. For more information, see Using the MapR Distribution for Hadoop (p. 149). 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 true. 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
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. 133). 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

AMI Version	Description	Release Date
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
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/kbostee/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 <i>fs.s3n.blockSize</i> parameter. You set this parameter by using the configure-hadoop bootstrap action. The default value is 9223372036854775807 (8 EiB). • Adds a /dev/sd symlink for each /dev/xvd device. For example, /dev/xvdb now has a symlink pointing to it called /dev/sdb. Now you can use the same device names for AMI 1.0 and 2.0. 	17 January 2012
2.0.1	<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 /mnt 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 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 sources.list to the new location of the Lenny distribution in archive.debian.org. 	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

Note

The cc2.8xlarge instance type is supported only on AMI 2.0.0 or later. The hi1.4xlarge and hs1.8xlarge instance types are supported only on AMI 2.3 or later.

Choose a Version of Hadoop

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 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 \(p. 149\)](#).

Topics

- [Supported Hadoop Versions \(p. 79\)](#)
- [How Does Amazon EMR Hadoop Differ from Apache Hadoop? \(p. 83\)](#)
- [Hadoop Patches Applied in Amazon EMR \(p. 84\)](#)
- [Supported Mahout Versions \(p. 86\)](#)

Supported Hadoop Versions

Amazon Elastic MapReduce (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	Configuration Parameters
2.4.0	<code>--ami-version 3.1.0</code>
2.2.0	<code>--ami-version 3.0.4 3.0.3 3.0.2 3.0.1</code>

Hadoop Version	Configuration Parameters
1.0.3	--ami-version 2.4.5 2.4.3 2.4.2
0.20.205	--ami-version 2.1.4
0.20	--ami-version 1.0

For details about the default configuration and software available on AMIs used by Amazon Elastic MapReduce (Amazon EMR) see [Choose a Machine Image \(p. 51\)](#).

Note

The Asia Pacific (Sydney) Region and AWS GovCloud (US) support only Hadoop 1.0.3 and later. AWS GovCloud (US) additionally requires AMI 2.3.0 and later.

To specify the Hadoop version when creating a cluster with the CLI

- Add the --ami-version option and specify the version number. The AMI version determines the version of Hadoop for Amazon EMR to use. The following example creates a waiting cluster running Hadoop 2.4.0. Amazon EMR then launches the appropriate AMI for that version of Hadoop. For details about the version of Hadoop available on an AMI, see [AMI Versions Supported in Amazon EMR \(p. 56\)](#).

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --alive --name "Test Hadoop" \
--ami-version 3.1.0 \
--num-instances 5 --instance-type m1.large
```

- Windows users:

```
ruby elastic-mapreduce --create --alive --name "Test Hadoop" --ami-version
3.0.1 --num-instances 5 --instance-type m1.large
```

Hadoop 2.4.0 New Features

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 changes 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 AMIs 3.1.0 or 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. 133).
- 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. 134).

Hadoop 2.2.0 New Features

Hadoop 2.2.0 supports the following new features:

- MapReduce NextGen (YARN) resource management system as a general big data processing platform. YARN is a new architecture that divides the two major functions of the JobTracker (resource management and job life-cycle management) into separate components and introduces a new ResourceManager. For more information, go to [Apache Hadoop NextGen MapReduce \(YARN\)](#) and [View Web Interfaces on the Cluster \(Hadoop 2.x\)](#) (p. 411).
- Pluggable Shuffle and Pluggable Sort. For more information, go to [Pluggable Shuffle and Pluggable Sort](#).
- Capacity Scheduler and Fair Scheduler. For more information, go to [Capacity Scheduler](#) and [Fair Scheduler](#).
- Hadoop Distributed File System (HDFS) snapshots. For more information, go to [HDFS Snapshots](#).
- Performance-related enhancements such as short-circuit read. For more information, go to [HDFS Short-Circuit Local Reads](#).
- Distributed job life cycle management by the application master
- Various security improvements

For a full list of new features and fixes available in Hadoop 2.2.0, go to [Hadoop 2.2.0 Release Notes](#).

Note

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

Major Changes from Hadoop 1 to Hadoop 2

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

- Updated Hadoop user interfaces with new URLs, including a new ResourceManager. For more information, see [View Web Interfaces on the Cluster \(Hadoop 2.x\) \(p. 411\)](#).
- Updated Hadoop configuration files. For more information, see [JSON Configuration Files \(p. 515\)](#).
- Changes to bootstrap actions that configure Hadoop daemons. For more information, see [Create Bootstrap Actions to Install Additional Software \(Optional\) \(p. 87\)](#).

Considerations for Moving to Hadoop 2.2.0

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

Hadoop 1.0 New Features

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 \(p. 114\)](#).

For a list of the patches applied to the Amazon EMR version of Hadoop 1.0.3, see [Hadoop 1.0.3 Patches \(p. 84\)](#).

Hadoop 0.20 New Features

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 \(p. 85\)](#).

How Does Amazon EMR Hadoop Differ from Apache Hadoop?

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 \(p. 84\)](#).

Hadoop Patches Applied in 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 \(p. 84\)](#)
- [Hadoop 0.20.205 Patches \(p. 85\)](#)

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. 85) 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.

Patch	Description
Add hadoop-lzo	Install the hadoop-lzo third-party package. For more information about hadoop-lzo, go to https://github.com/kevinweil/hadoop-lzo <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	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>Status: Third-party Library Fixed in AWS Hadoop Version: 0.20.205 Fixed in Apache Hadoop Version: n/a</p>
MAPREDUCE-1597/2021/2046	Fixes to how CombineFileInputFormat handles split locations and files that can be split. For more information about these patches, go to https://issues.apache.org/jira/browse/MAPREDUCE-1597 , https://issues.apache.org/jira/browse/MAPREDUCE-2021 , and https://issues.apache.org/jira/browse/MAPREDUCE-2046 . <p>Status: Resolved, Fixed Fixed in AWS Hadoop Version: 0.20.205 Fixed in Apache Hadoop Version: 0.22.0</p>
HADOOP-6436	Remove the files generated by automake and autoconf of the native build and use the host's automake and autoconf to generate the files instead. For more information about this patch, go to https://issues.apache.org/jira/browse/HADOOP-6436 . <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	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>Status: Closed, Fixed Fixed in AWS Hadoop Version: 0.20.205 Fixed in Apache Hadoop Version: 0.23.0</p>
HADOOP-7082	Change Configuration.writeXML to not hold a lock while outputting. For more information about this patch, go to https://issues.apache.org/jira/browse/HADOOP-7082 . <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 ClusterDumper (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>.

Create Bootstrap Actions to Install Additional Software (Optional)

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 \(p. 87\)](#)
- [Using Predefined Bootstrap Actions \(p. 87\)](#)
- [Using Custom Bootstrap Actions \(p. 91\)](#)

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 interfaces support bootstrap actions. You can specify up to 16 bootstrap actions per cluster by providing multiple `bootstrap-action` parameters from the 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. The syntax for a `--bootstrap-action` parameter is as follows:

```
--bootstrap-action s3://myawsbucket/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.

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

- [Configure Hadoop Settings with a Bootstrap Action \(p. 89\)](#)
- [Run If \(p. 90\)](#)
- [Shutdown Actions \(p. 90\)](#)

Configure Daemons

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`, `resource-manager`, `nodemanager`, or `client` (Hadoop 2.x). For example, `--namenode-heap-size=2048,--namenode-opts=-XX:GCTimeRatio=19`

Configuration Parameter	Description
<code>--daemon-heap-size</code>	Sets the heap size in megabytes for the specified daemon.
<code>--daemon-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.

In Hadoop 1.x, `--client-heap-size` has no effect. Instead, change the client heap size using the `--client-opts=-Xmx#####` equivalent, where ##### is numeric.

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 Amazon EMR CLI

In the directory where you installed the Amazon EMR CLI, run 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://elasticmapreduce/bootstrap-actions/configure-daemons --args --namenode-heap-size=2048,--namenode-opts=-XX:GCTimeRatio=19
```

Configure Hadoop Settings with a 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 Shortcut	example
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

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

```
<?xml 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 Amazon EMR CLI

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

- Linux, UNIX, and Mac OS X:

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

- Windows:

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

Run If

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 Amazon EMR CLI

In the directory where you installed the Amazon EMR CLI, run 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"
```

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.

Using Custom Bootstrap Actions

In addition to predefined 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

- [Running Custom Bootstrap Actions Using the CLI \(p. 91\)](#)
- [Running Custom Bootstrap Actions Using the Console \(p. 91\)](#)

Running Custom Bootstrap Actions Using the CLI

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

The sample script looks like the following:

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

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

In the directory where you installed the Amazon EMR CLI, run 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"
```

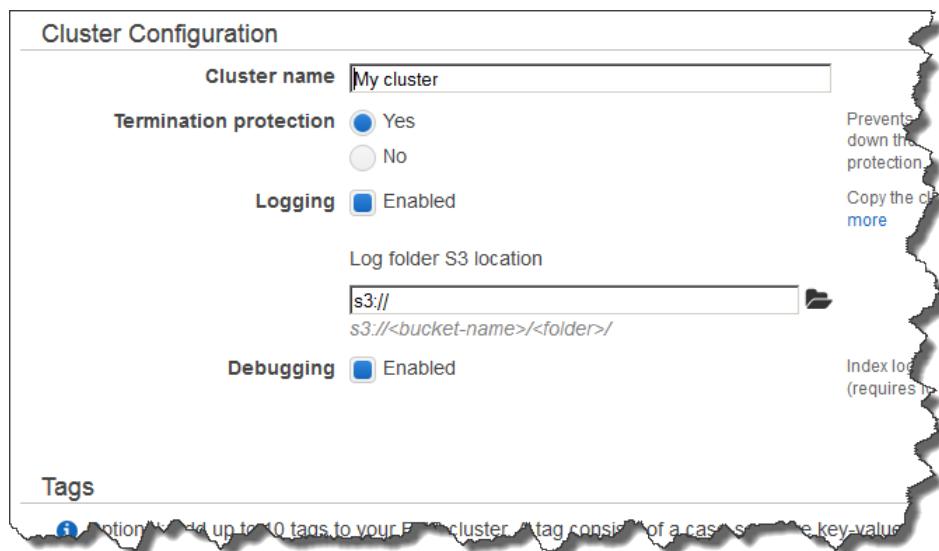
Running Custom Bootstrap Actions Using the Console

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 Elastic MapReduce console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Click **Create cluster**.
3. In the **Create Cluster** page, click **Configure sample application**.

4. In the **Configure Sample Application** page, in the **Select sample application** field, choose the **Word count** sample application from the list.
5. In the **Output location** field, type the path of an Amazon S3 bucket to store your output and then click **Ok**.
6. 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>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 (p. 459). 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>This determines whether Amazon EMR captures detailed log data to Amazon S3.</p> <p>For more information, see View Log Files (p. 418).</p>
Log folder S3 location	<p>Enter an Amazon S3 path to store your debug logs if you enabled logging in the previous field. If the log folder does not exist, the Amazon EMR console creates it for you.</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 (p. 418).</p>

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

Field	Action
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.

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

Software Configuration

Hadoop distribution **Amazon** Use Amazon's Hadoop distribution. [Learn more](#)

AMI version Determines the base configuration of the instances in your cluster, including the Hadoop version. [Learn more](#)

MapR Use MapR's Hadoop distribution. [Learn more](#)

Applications to be installed	Version	Edit
Hive	0.11.0.1	
Pig	0.11.1.1	
Additional applications	<input type="text" value="Select an application"/>	Configure and add

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. 149).</p>
AMI version	<p>Choose 2.4.2 (Hadoop 1.0.3).</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 (p. 51).</p>

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, 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](#).

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

Hardware Configuration

Specify the networking and hardware configuration for your cluster. If you need more than 20 EC2 Request Spot instances (unused EC2 capacity) to save money.

Network	<input type="text" value="vpc-c1a3b3a3 (172.31.0.0/16) (default)"/>	<input type="checkbox"/> Use a Virtual Private Cloud (VPC) to connect to your cluster	
EC2 Subnet	<input type="text" value="No preference (random subnet)"/>	Create a Subnet	
EC2 instance type	Count	Request spot	
Master	<input type="text" value="m1.small"/>	1 <input type="checkbox"/>	The Master instance is the primary node in the cluster, and is responsible for assigning tasks to core and task nodes, and monitoring their status. There is always one master node in each cluster.
Core	<input type="text" value="m1.small"/>	2 <input type="checkbox"/>	Core instances are used to store data and run Hadoop MapReduce tasks.
Task	<input type="text" value="m1.small"/>	0 <input type="checkbox"/>	Task instances are used to run Hadoop MapReduce tasks.

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 a Amazon VPC Subnet for the Cluster (Optional) (p. 137).</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 Elastic Compute Cloud User Guide</i>.</p>
Master	<p>Choose m1.small.</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.small (default), m1.large, m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cc1.4xlarge, cg1.4xlarge.</p> <p> This tutorial uses small instances for all nodes due to the light workload and to keep your costs low.</p> <p> For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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 Lower Costs with Spot Instances (Optional) (p. 37).</p>

Field	Action
Core	<p>Choose m1.small.</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.small (default), m1.large, m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cc1.4xlarge, cg1.4xlarge.</p> <p>This tutorial uses small instances for all nodes due to the light workload and to keep your costs low.</p> <p>For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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) (p. 37).</p>
Task	<p>Choose m1.small.</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. Valid types: m1.small (default), m1.large, m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cc1.4xlarge, cg1.4xlarge.</p> <p>For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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) (p. 37).</p>

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

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

Security and Access

EC2 key pair [Proceed without an EC2 key pair](#) Use an existing key pair to SSH into the master node of the Amazon EC2 cluster as the user "hadoop". [Learn more](#)

IAM user access All other IAM users Control the visibility of this cluster to other IAM users. [Learn more](#)
 No other IAM users

IAM Roles

EMR role [No roles found](#) Allows EMR to access other AWS Services such as EC2 on your behalf. [Learn more](#)
[Create Default Role](#)

EC2 instance profile [Proceed without role](#) Allows EC2 instances in an EMR cluster to access other AWS services such as S3. [Learn more](#)
[Create Default Role](#)

Field	Action
EC2 key pair	<p>Choose an Amazon EC2 key pair from the list.</p> <p>For more information, see Create an Amazon EC2 Key Pair and PEM File (p. 117).</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 Cluster (p. 446).</p>
IAM user access	<p>Choose No other IAM users.</p> <p> 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 (p. 119).</p>
EC2 instance profile	<p>You can proceed without choosing an instance profile. If you create a cluster without selecting a specific instance profile, one will be created for you.</p> <p>This controls application access to the Amazon EC2 instances in the cluster.</p> <p>For more information, see Configure IAM Roles for Amazon EMR (p. 125).</p>
EMR role	<p>Choose Proceed without role.</p> <p>Allows Amazon EMR to access other AWS services on your behalf.</p> <p>For more information, see Configure IAM Roles for Amazon EMR (p. 125).</p>

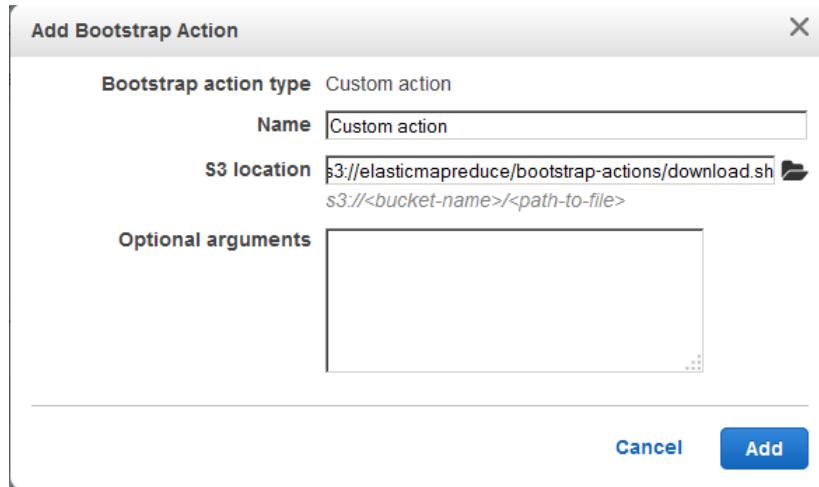
10. In the **Bootstrap Actions** section, in the **Add bootstrap action** field, select **Custom action** and click **Configure and add**.

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**.



11. In the **Steps** section, note the step that Amazon EMR configured for you by choosing the sample application.

You do not need to change any of the settings in this section.

12. Review your configuration and if you are satisfied with the settings, click **Create Cluster**.
13. When the cluster starts, the console displays the **Cluster Details** page.

While the cluster 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 \(p. 418\)](#)

Choose the Cluster Lifecycle: Long-Running or Transient

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 enable Keep Alive when you launch the cluster. You can do this when you launch a cluster using the console, the 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 [Protect a Cluster from Termination \(p. 459\)](#).

To launch a long-running 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 **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.

The screenshot shows a 'Steps' configuration page. At the top, there is a note: 'A step is a unit of work you submit to the cluster. A step might contain one or more Hadoop jobs to configure an application. You can submit up to 256 steps to a cluster. [Learn more](#)'.

Name	Action on failure	JAR S3 location
Add step	Streaming program	<input type="button" value="Configure and add"/>

Below the table, there is a section for 'Auto-terminate' with two radio buttons: 'Yes' and 'No'. The 'No' radio button is selected and highlighted with a red box.

4. Click **Done** and proceed to create the cluster as described in [Plan an Amazon EMR Cluster \(p. 29\)](#).

To launch a long-running cluster using the CLI

- In the directory where you installed the Amazon EMR CLI, launch a cluster and specify the `--alive` argument. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).
 - Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --alive
```

- Windows users:

```
ruby elastic-mapreduce --create --alive
```

Prepare Input Data (Optional)

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 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 \(p. 99\)](#)
- [File Systems compatible with Amazon EMR \(p. 99\)](#)
- [How to Get Data Into Amazon EMR \(p. 102\)](#)

Types of Input Amazon EMR Can Accept

The default input format for a cluster is text files with each line separated by a newline (`\n`) character. 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 `HParse` to convert the data to text or XML format. For more information, see [Parse Data with HParse \(p. 148\)](#).

If you are launching a Hive cluster, 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>.

File Systems compatible with Amazon EMR

Amazon Elastic MapReduce (Amazon EMR) and Hadoop provide a variety of file systems 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 the S3 native file system. The following table lists the available file systems, with recommendations on when it's best to use them.

Amazon EMR and Hadoop typically use two or more of the following file systems when processing a cluster. HDFS and S3N 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, see the Hadoop documentation.</p> <p>HDFS is used by the master and core nodes. Its advantage is that it's fast; its 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>

File System	Prefix	Description
Amazon S3 native	s3n://	<p>The Amazon S3 Native File System (S3N) is a file system for reading and writing regular files on Amazon S3. The advantage of this file system is that you can access files on Amazon S3 that were written using other tools.</p> <p>Important On Amazon EMR, s3n:// and s3:// both map to the Amazon S3 native file system. In the default configuration of Apache Hadoop, s3:// is mapped to the Amazon S3 block storage system. For more information about how Amazon S3 and Hadoop work together, see Amazon S3 on the Hadoop Wiki.</p> <p>S3N is a persistent and fault-tolerant file system. It continues to exist after the cluster ends. The disadvantage is that it's slower than HDFS because of the round-trip to Amazon S3. S3N is best used for storing the input to a cluster, the output of the cluster, and the results of intermediate cluster steps where re-computing the step would be onerous.</p> <p>Note that paths that specify a bucket name only must end with a terminating slash. For example, specify s3n://myawsbucket/ instead of s3n://myawsbucket. Ending with a folder, for example s3n://myawsbucket/myfolder, is also fine.</p>
(Legacy) Amazon S3 block	s3bfs://	<p>The Amazon S3 Block File System Files is a legacy file storage system. We strongly discourage the use of this system.</p> <p>Important We do not recommend that you 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>
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 to store 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>

Upload Large Files with the S3 Native File System

The S3 native file system imposes a 5 GiB file-size limit. You might need to upload or store files larger than 5 GiB with Amazon S3. Amazon EMR makes this possible by extending the S3 file system through the AWS Java SDK to support multipart uploads. Using this feature of Amazon EMR you can upload files of up to 5 TiB in size. Multipart upload is disabled by default; to learn how to enable it for your cluster, see [Configure Multipart Upload for Amazon S3 \(p. 104\)](#).

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 the Amazon S3 native file system

- Use the `s3n://` or `s3://` prefix. Amazon EMR resolves both of the URIs below to the same location.

```
s3n://bucket-name/path-to-file-in-bucket  
s3://bucket-name/path-to-file-in-bucket
```

Note

Because of the file syntax difference between Hadoop running on Amazon EMR and standard Apache Hadoop, it is recommended that you use the `s3n://` prefix to highlight the fact that you are using the S3 native file system.

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 `s3 bfs://` 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 GiB in size. With the multipart upload functionality Amazon EMR provides through the AWS Java SDK, you can upload files of up to 5 TiB 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 the Amazon S3 native file system instead.

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

How to Get Data Into Amazon EMR

Amazon EMR provides several ways to get data onto a cluster. The most common way is to upload the data to Amazon S3, and use the built-in features of Amazon EMR to load the data onto your cluster. You can also use the Distributed Cache feature of Hadoop to transfer files from a distributed file system to the local file system. The implementation of Hive provided by Amazon EMR (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 \(p. 102\)](#)
- [Import files using Distributed Cache \(p. 106\)](#)
- [How to Process Compressed Files \(p. 111\)](#)
- [Import DynamoDB Data into Hive \(p. 112\)](#)
- [Connect to Data with AWS DirectConnect \(p. 112\)](#)
- [Upload Large Amounts of Data with AWS Import/Export \(p. 112\)](#)

Upload Data to Amazon S3

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 \(p. 102\)](#)
- [Configure Multipart Upload for Amazon S3 \(p. 104\)](#)
- [Create a cluster with Amazon S3 Server Side Encryption Enabled \(p. 106\)](#)

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. Click **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 \(p. 29\)](#) for guidance on choosing a Region.
5. Click **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, 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** drop-down list, select **List**.
8. Click **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>

Information	Example Location on Amazon S3
output data	s3://myawsbucket/output

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.

Amazon EMR Console

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

To disable multipart uploads with a predefined bootstrap action

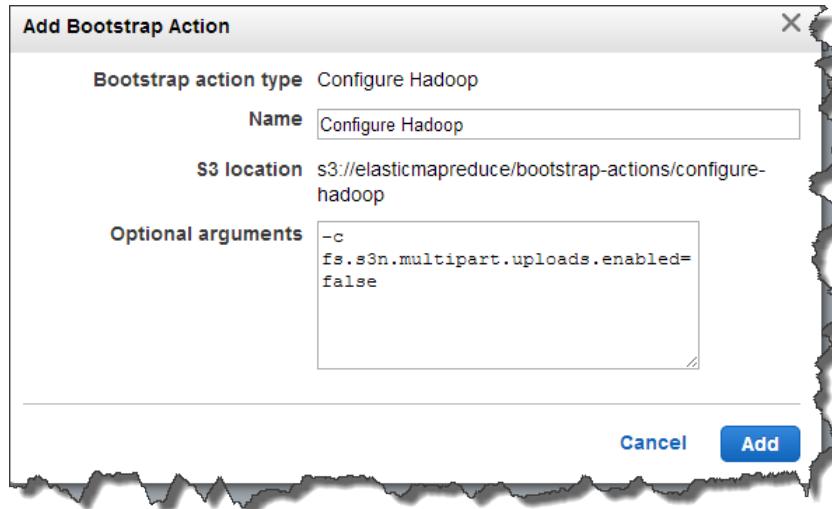
1. Open the Amazon Elastic MapReduce console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Click **Create cluster**.
3. In the **Bootstrap Actions** section, in the **Add bootstrap action** field, select **Configure Hadoop** and click **Configure and add**.

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 [Create Bootstrap Actions to Install Additional Software \(Optional\) \(p. 87\)](#).

4. Click **Done** and proceed to create the cluster as described in [Plan an Amazon EMR Cluster \(p. 29\)](#).

CLI

This procedure explains how to disable multipart upload using the CLI. The command creates a cluster in a waiting state with multipart upload disabled.

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- 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://elasticmapreduce/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.

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.

Create a cluster with Amazon S3 Server Side Encryption Enabled

Amazon S3 Server Side Encryption (SSE) is supported with Amazon EMR on AMIs 3.1.0 or later. This functionally allows customers to write directly through to Amazon S3 locations that have policies which enforce encryption. To launch a cluster with server-side encryption use the `configure-hadoop` bootstrap action to set `fs.s3.enableServerSideEncryption` to "true":

```
elastic-mapreduce --create --alive --ami-version 3.1.0 --instance-type m1.large
 \
--bootstrap-action s3:/us-east-1.elasticmapreduce/bootstrap-actions/configure-
hadoop --args "-e,fs.s3.enableServerSideEncryption=true"
```

Note

This is a global setting. When enabled, all Amazon S3 write actions that happen through Hadoop will use SSE.

Import files using Distributed Cache

Topics

- [Supported File Types \(p. 106\)](#)
- [Location of Cached Files \(p. 107\)](#)
- [Access Cached Files From Mapper and Reducer Applications \(p. 107\)](#)
- [Amazon EMR Console \(p. 107\)](#)
- [CLI \(p. 108\)](#)

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 S3 native. 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 [Create Bootstrap Actions to Install Additional Software \(Optional\) \(p. 87\)](#) 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/`. 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 creating streaming clusters.

Access Cached Files From Mapper and Reducer 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.

For more information, go to http://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html#DistributedCache.

Amazon EMR Console

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

To specify Distributed Cache files

1. Open the Amazon Elastic MapReduce console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Click **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 \s3n://bucket_name/file_name#cache_file_name</code>

If you want to ...	Action	Example
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 \s3n://bucket_name/archive_name#cache_archive_name</code>

Add Step

Step type: Streaming program

Name*:

Mapper*: S3 location of the map function or the name of the Hadoop streaming command to run.

Reducer*: S3 location of the reduce function or the name of the Hadoop streaming command to run.

Input S3 location*: S3 location of the input data.

Output S3 location*: S3 location of the output data.

Arguments:

Action on failure: What to do if the step fails.

[Cancel](#) [Add](#)

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

CLI

You can use the Amazon EMR console to create clusters that use Distributed Cache. To add files or archives to the Distributed Cache using the CLI, you specify the options `--cache` or `--cache-archive` to the CLI command line.

To specify Distributed Cache files

- Create a streaming cluster and add the following parameters:

For information on how to create a streaming cluster using the CLI, go to [Launch a Streaming Cluster \(p. 169\)](#).

The size of the file (or total size of the files in an archive file) must be less than the allocated cache size.

If you want to ...	Add the following parameter to the cluster ...
add an individual file to the Distributed Cache	specify <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	enter <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.

The output looks similar to the following.

```
Created jobflow JobFlowID
```

Your cluster copies the files to the cache location before processing any job flow steps.

Example 1

The following command shows the creation of a streaming cluster and uses `--cache` to add one file, `sample_dataset_cached.dat`, to the cache.

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

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 "s3n://my_bucket/my_mapper.py,s3n://my_bucket/my_reducer.py" \
--input s3n://my_bucket/my_input \
--output s3n://my_bucket/my_output \
--mapper my_mapper.py \
--reducer my_reducer.py \
--cache s3n://my_bucket/sample_dataset.dat#sample_dataset_cached.dat
```

- Windows users:

```
ruby elastic-mapreduce --create --stream --arg "-files" --arg "s3n://my_bucket/my_mapper.py,s3n://my_bucket/my_reducer.py" --input s3n://my_bucket/my_input \
--output s3n://my_bucket/my_output --mapper my_mapper.py --reducer my_reducer.py --cache s3n://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 s3n://my_bucket/my_input \
--output s3n://my_bucket/my_output \
--mapper s3n://my_bucket/my_mapper.py \
--reducer s3n://my_bucket/my_reducer.py \
--cache s3n://my_bucket/sample_dataset.dat#sample_dataset_cached.dat
```

- Windows users:

```
ruby elastic-mapreduce --create --stream --input s3n://my_bucket/my_input --output s3n://my_bucket/my_output --mapper s3n://my_bucket/my_mapper.py --reducer s3n://my_bucket/my_reducer.py --cache s3n://my_bucket/sample_dataset.dat#sample_dataset_cached.dat
```

Example 2

The following command shows the creation of a streaming cluster and uses `--cache-archive` 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:

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --stream \
--arg "-files" --arg "s3n://my_bucket/my_mapper.py,s3n://my_bucket/my_redu
cer.py \
--input s3n://my_bucket/my_input \
--output s3n://my_bucket/my_output \
--mapper my_mapper.py \
--reducer my_reducer.py \
--cache-archive s3n://my_bucket/sample_dataset.tgz#sample_dataset_cached
```

- Windows users:

```
ruby elastic-mapreduce --create --stream --arg "-files" --arg "s3n://my_buck
et/my_mapper.py,s3n://my_bucket/my_reducer.py" --input s3n://my_bucket/my_input
--output s3n://my_bucket/my_output --mapper my_mapper.py --reducer my_redu
cer.py --cache-archive s3n://my_bucket/sample_dataset.tgz#sample_dataset_cached
```

For Hadoop 1.x, use the following command:

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --stream \
--input s3n://my_bucket/my_input \
--output s3n://my_bucket/my_output \
--mapper s3n://my_bucket/my_mapper.py \
--reducer s3n://my_bucket/my_reducer.py \
--cache-archive s3n://my_bucket/sample_dataset.tgz#sample_dataset_cached
```

- Windows users:

```
ruby elastic-mapreduce --create --stream --input s3n://my_bucket/my_input --o
utput s3n://my_bucket/my_output --mapper s3n://my_bucket/my_mapper.py --redu
cer s3n://my_bucket/my_reducer.py --cache-archive s3n://my_bucket/sample_data
set.tgz#sample_dataset_cached
```

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 (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. 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 \(p. 364\)](#).

Connect to Data with AWS DirectConnect

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

Upload Large Amounts of Data with AWS Import/Export

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

Prepare an Output Location (Optional)

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 \(p. 112\)](#)
- [What formats can Amazon EMR return? \(p. 114\)](#)
- [How to write data to an Amazon S3 bucket you don't own \(p. 114\)](#)
- [Compress the Output of your Cluster \(p. 116\)](#)

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. Click **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 \(p. 29\)](#) for guidance on choosing a Region.
5. Click **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, 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** drop-down list, select **List**.
8. Click **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>

Information	Example Location on Amazon S3
log files	s3://myawsbucket/logs
input data	s3://myawsbucket/input
output data	s3://myawsbucket/output

What formats can Amazon EMR return?

The default output format for a cluster is text with key, value pairs written to individual lines of the text files. This is the output format most commonly used.

If your output data needs to be written in a format other than the default text files, you can use the Hadoop interface `OutputFormat` to specify other output types. You can even create a subclass of the `FileOutputFormat` class to handle custom data types. For more information, see <http://hadoop.apache.org/docs/current/api/org/apache/hadoop/mapred/OutputFormat.html>.

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

How to write data to an Amazon S3 bucket you don't own

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

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

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

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

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

Canned ACL	Description
AuthenticatedRead	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.

Canned ACL	Description
BucketOwnerFullControl	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.
BucketOwnerRead	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.
LogDeliveryWrite	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.
Private	Specifies that the owner is granted <code>Permission.FullControl</code> .
PublicRead	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.
PublicReadWrite	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 \(p. 446\)](#).

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

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 somedata 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 \(p. 116\)](#)
- [Intermediate Data Compression \(p. 117\)](#)
- [Using the Snappy Library with Amazon EMR \(p. 117\)](#)

Output Data Compression

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

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

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

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

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

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

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

Intermediate Data Compression

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

When writing a Custom Jar, use the following command:

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

Using the Snappy Library with Amazon EMR

Snappy is a compression and decompression library that is optimized for speed. It is available on Amazon EMR AMIs version 2.0 and later and is used as the default for intermediate compression. For more information about Snappy, go to <http://code.google.com/p/snappy/>. For more information about Amazon EMR AMI versions, go to [Choose a Machine Image \(p. 51\)](#)

Configure Access to the Cluster

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. When you launch a cluster, you can associate an Amazon 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 \(p. 117\)](#)
- [Configure IAM User Permissions \(p. 119\)](#)
- [Set Access Policies for IAM Users \(p. 122\)](#)
- [Configure IAM Roles for Amazon EMR \(p. 125\)](#)
- [Setting Permissions on the System Directory \(p. 133\)](#)

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 <code>PEM</code> 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">a. 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, <code>mykeypair.ppk</code>.i. Click Save.j. 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 *Using IAM* 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 terminate, Amazon S3 buckets in your AWS account. 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 Amazon EMR 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 \(p. 125\)](#).

Hidden Clusters

By default, if an IAM user launches a cluster, that cluster is hidden from other IAM users on the AWS account. For example, if an IAM user uses the CLI to run the `--list` command, the CLI only lists hidden clusters launched by that IAM user, not hidden clusters launched by other IAM users on the AWS account. This filtering occurs on all Amazon EMR interfaces—the console, CLI, API, and SDKs—and prevents IAM users from accessing and 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

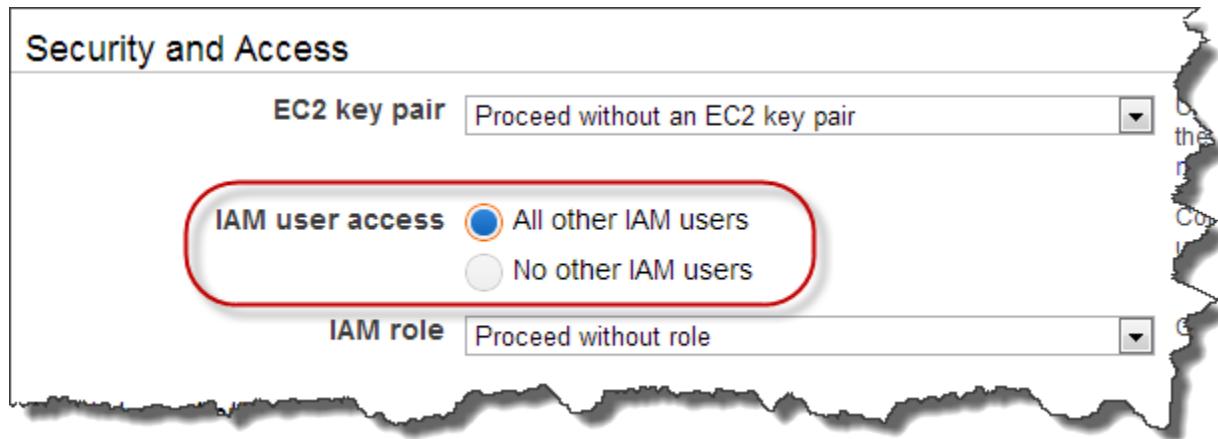
This filtering does not prevent IAM users from viewing the underlying resources of the cluster, such as EC2 instances, by using AWS interfaces outside of Amazon EMR.

Visible Clusters

You also have the option to make a cluster visible and accessible to all IAM users under a single AWS account. This visibility can be set when you launch the cluster, or it can be added to a cluster that is already running.

Using this feature, you can make it possible for all IAM users on your account to access the cluster and, by configuring the policies of the IAM groups they belong to, 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 make a cluster visible to all IAM users using the Amazon EMR console



1. Open the Amazon Elastic MapReduce console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Click **Create cluster**.
3. In the **Security and Access** section, in the **IAM User Access** field, choose **All other IAM Users**.

This makes the cluster visible and accessible to all IAM users on the AWS account. For more information, see [Configure IAM User Permissions \(p. 119\)](#).

4. Click **Done** and proceed to create the cluster as described in [Plan an Amazon EMR Cluster \(p. 29\)](#).

To make a cluster visible to all IAM users using the Amazon EMR CLI

- If you are adding IAM user visibility to a new cluster, 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, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- 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 of the Amazon EMR CLI, and specify identifier of the cluster to modify. This is shown in the following example, where `job-flow-identifier` would be replaced by the cluster identifier of your cluster. 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, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --set-visible-to-all-users true --jobflow job-flow-identifier
```

- Windows users:

```
ruby elastic-mapreduce --set-visible-to-all-users true --jobflow job-flow-identifier
```

To make a cluster visible to all IAM users using the Amazon EMR API

- If you are adding IAM user visibility to a new cluster, call [RunJobFlow](#) and set `VisibleToAllUsers=true`, as shown in the following example.

```
https://elasticmapreduce.amazonaws.com?Operation=RunJobFlow
&Name=MyJobFlowName
&VisibleToAllUsers=true
&LogUri=s3n%3A%2Fmybucket%2Fsubdir
&Instances.MasterInstanceType=m1.small
&Instances.SlaveInstanceType=m1.small
&Instances.InstanceCount=4
&Instances.Ec2KeyName=myec2keyname
&Instances.Placement.AvailabilityZone=us-east-1a
&Instances.KeepJobFlowAliveWhenNoSteps=true
&Instances.TerminationProtected=true
&Steps.member.1.Name=MyStepName
&Steps.member.1.ActionOnFailure=CONTINUE
&Steps.member.1.HadoopJarStep.Jar=MyJarFile
&Steps.member.1.HadoopJarStep.MainClass=MyMainClass
&Steps.member.1.HadoopJarStep.Args.member.1=arg1
&Steps.member.1.HadoopJarStep.Args.member.2=arg2
&AuthParams
```

If you are adding IAM user visibility to an existing cluster, call [SetVisibleToAllUsers](#) and set `VisibleToAllUsers` to `true`, as shown in the following example. 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.

```
https://elasticmapreduce.amazonaws.com?Operation=SetVisibleToAllUsers
&VisibleToAllUsers=true
&JobFlowIds.member.1=j-3UN6WX5RRO2AG
&AuthParams
```

Set Access Policies for IAM Users

The ability for 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

For more information, see [Creating and Listing Groups](#) in *Using IAM* guide.

To add a permission to a user or group, write a policy that contains the permission and attach the policy to the user or group. You cannot specify a specific Amazon EMR resource in a policy, such as a specific cluster. You can only specify `Allow` or `Deny` access to Amazon EMR API actions.

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 *Using IAM* guide.

Users with permission to use Amazon EMR API actions can create and manage clusters as described elsewhere in this guide. Users must use their own AWS access ID and secret key to authenticate Amazon EMR commands. For more information about creating clusters, see [Manage Clusters \(p. 406\)](#).

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. For information about attaching policies to users, see [Managing IAM Policies](#) in the *Using IAM* guide.

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Action": [  
        "iam>ListRoles",  
        "iam>PassRole",  
        "elasticmapreduce:*",  
        "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:DescribeVpcAttributes",
    "ec2:DescribeVpcs",
    "ec2:ModifyImageAttribute",
    "ec2:ModifyInstanceStateAttribute",
    "ec2:RequestSpotInstances",
    "ec2:RunInstances",
    "ec2:TerminateInstances",
    "cloudwatch:/*",
    "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

- [How to Write a Policy](#) (*Using IAM* guide)

Read-Only Access Policy

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

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

Related Topics

- [How to Write a Policy](#) (*Using IAM guide*)

IP-Restricted Access Policy

The following policy denies any traffic using the AWS account that does not come from the named IP address ranges. For information about attaching policies to users, see [Managing IAM Policies](#) in the *Using IAM* guide.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Deny",  
            "Action": "*",  
            "Resource": "*",  
            "Condition": {  
                "NotIpAddress": {  
                    "aws:SourceIp": ["10.1.2.0/24", "10.1.3.0/24"]  
                }  
            }  
        }  
    ]  
}
```

This example policy uses the AWS-wide key called `aws:SourceIp` to specify the range of valid IP addresses. For information about AWS-wide policy keys, see [Element Descriptions](#) in the *Using IAM* guide.

Related Topics

- [How to Write a Policy](#) (*Using IAM guide*)

Amazon EC2 Limited Access Policy

This example policy allows people in certain job functions, such as data analysts, to develop and run Amazon EMR clusters, but not create, modify, or terminate Amazon EC2 instances outside of Amazon EMR.

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

```
        "cloudwatch: *",
        "sdb: *"
    ],
    "Effect": "Allow",
    "Resource": "*"
},
{
    "Action": [
        "ec2:RunInstances",
        "ec2:ModifyImageAttribute",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeInstances",
        "ec2:DescribeKeyPairs",
        "ec2:DescribeRouteTables",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSpotInstanceRequests",
        "ec2:DescribeSubnets"
    ],
    "Effect": "Allow",
    "Resource": "*"
}
]
```

Configure IAM Roles for Amazon EMR

AWS Identity and Access Management (IAM) roles provide a way to delegate access so that IAM users or AWS services have the specified permissions and can act on your AWS resources. Amazon EMR lets you 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 can 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.

The 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 *Using IAM* guide.

Topics

- [Required IAM Roles for Amazon EMR \(p. 126\)](#)
- [Default IAM Roles for Amazon EMR \(p. 126\)](#)
- [Custom IAM Roles for Amazon EMR \(p. 129\)](#)
- [Launch a Cluster with IAM Roles \(p. 130\)](#)
- [Access AWS Resources Using IAM Roles \(p. 132\)](#)

Required IAM Roles for Amazon EMR

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. If you are not launching the cluster in the AWS GovCloud (US) region, roles are optional.

If you select a service role, you must also select an instance profile.

If you do not specify the name of a role when you launch a cluster, the cluster is launched without roles enabled, and any applications on the cluster that need to access AWS resources must use pre-role authentication methods. This method of authentication will be phased out in the future.

To use federation, you must select service role and instance profile.

Default IAM Roles for Amazon EMR

To simplify using IAM roles, Amazon EMR can create default service and instance profiles. For more information, see [Create Default IAM Roles for Amazon EMR \(p. 128\)](#).

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 \(p. 129\)](#).

If you are using an IAM user with the CLI and creating default roles for a cluster, your IAM user must have the `iam:CreateRole`, `iam:PutRolePolicy`, `iam:CreateInstanceProfile`, `iam:AddRoleToInstanceProfile`, `iam:PassRole`, and `iam>ListInstanceProfiles` permissions. However, to use a pre-existing IAM role, a user only needs the `iam:GetInstanceProfile`, `iam:GetRole`, `iam:PassRole`, and `iam>ListInstanceProfiles` permissions.

Topics

- Default Service Roles for Amazon EMR (p. 126)
 - Default instance profiles for Amazon EMR (p. 127)
 - Create Default IAM Roles for Amazon EMR (p. 128)

Default Service Roles for Amazon EMR

The default IAM role for Amazon EMR is `EMR_DefaultRole`. This role is defined as follows:

```
        "iam>ListRolePolicies",
        "iam:GetRole",
        "iam:GetRolePolicy",
        "iam>ListInstanceProfiles",
        "s3:Get*",
        "s3>List*",
        "s3>CreateBucket",
        "sdb:BatchPutAttributes",
        "sdb>Select"
    ],
    "Resource": "*",
    "Effect": "Allow"
}
],
{
    "Version": "2008-10-17",
    "Statement": [
        {
            "Action": "sts:AssumeRole",
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "elasticmapreduce.amazonaws.com"
            }
        }
    ]
}
```

Default instance profiles for Amazon EMR

The default instance profile is `EMR_EC2_DefaultRole`. This role is defined as follows:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "cloudwatch: *",
                "dynamodb: *",
                "ec2:Describe*",
                "elasticmapreduce:Describe*",
                "rds:Describe*",
                "s3: *",
                "sdb: *",
                "sns: *",
                "sqs: *"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        }
    ],
    {
        "Version": "2008-10-17",
```

```
    "Statement": [
        {
            "Action": "sts:AssumeRole",
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "ec2.amazonaws.com"
            }
        }
    ]
```

Create Default IAM Roles for Amazon EMR

There are three ways to create IAM roles:

- Use the Amazon EMR console to create the default roles. For more information, see [Create Default Roles with the Console \(p. 128\)](#).
- Use the CLI to create the default roles, using the `--create-default-roles` option. For more information, see [Create Default Roles with the CLI \(p. 128\)](#).
- Use the IAM console or API. For more information, see [Creating a Role for an AWS Service](#) in the [Using IAM](#) 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 [Using IAM](#) guide.

Create Default Roles with the Console

You can create default roles at cluster creation. You can also specify other roles you may already be using.

To create default roles using the console

1. Sign in to the AWS Management Console and open the Amazon Elastic MapReduce console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Click **Create Cluster**.
3. In the **Security and Access** section, in the **IAM User Access** field, choose a role from the list for **EC2 role** or create the default role by clicking **Create Default Role**.
4. In the **Service role** field, choose a role from the list for **Service role** or create the default role by clicking **Create Default Role**.

Create Default Roles with the CLI

To create default roles using the CLI

- Enter the following at the command line to create default roles:
 - Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create-default-roles
```

- Windows Users:

```
ruby elastic-mapreduce --create-default-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 \(p. 126\)](#).

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 *Using IAM* guide.

- [Creating a Role](#)
- [Modifying a Role](#)
- [Deleting a Role](#)

We recommend that you use the permissions in `EMR_DefaultRole` and `EMR_EC2_DefaultRole` as a starting place when developing custom IAM roles to use with Amazon EMR. To ensure that you always have access to the original version of the default IAM roles, generate the default roles using the Amazon EMR CLI, copy the contents of the 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.

```
{
  "Version": "2012-10-17",
  "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 Role** list. The CLI interprets the name passed to the `--jobflow-role` option as an IAM role name and does not recognize instance profile names. For this reason, if you use the IAM CLI or API to create a IAM role and its associated instance profile, we recommend that you give a 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 to specify the instance profile name. Unlike the console or CLI, the API does not interpret an instance profile name as a IAM role name. You can call the `RunJobFlow` action and pass the instance profile name for the `JobFlowRole` parameter.

Launch a Cluster with IAM Roles

To use IAM roles, the following versions of Amazon EMR components are required:

- AMI version 2.3.0 or later.
- If you are using Hive, version 0.8.1.6 or later.
- If you are using the CLI, version 2012-12-17 or later.
- If you are using s3DistCP, use the version at `s3://elasticmapreduce/libs/s3distcp/role/s3distcp.jar`.

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**. The following policy allows the IAM user to create a cluster:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Action": [  
                "elasticmapreduce:*",  
                "ec2:*",  
                "cloudwatch:*",  
                "s3:*",  
                "sdb:*",  
                "iam:AddRoleToInstanceProfile",  
                "iam:PassRole",  
                "iam:GetInstanceProfile",  
                "iam:GetRole"  
            ],  
            "Effect": "Allow",  
            "Resource": "*"  
        }  
    ]  
}
```

```
    ]  
}
```

To launch a cluster with IAM roles using the console

- Create a cluster and specify the IAM roles using the **IAM User Access** and **Service role** fields. For more information, see [Create Default Roles with the Console \(p. 128\)](#)

To launch a cluster with IAM roles using the CLI

- 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. The following example shows how to create an interactive Hive cluster that uses the default IAM roles provided by Amazon EMR.

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- Linux, UNIX, and Mac OS X users:

```
./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
```

- 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 for the CLI

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 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"
}
```

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.

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.

Amazon Elastic MapReduce (Amazon EMR) added this configuration variable to Hadoop in AMI 2.0.5, and it is available in AMI versions 2.0.5 and later. For more information about AMI versions, see [Choose a Machine Image \(p. 51\)](#).

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)

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 \(p. 133\)](#)
- [Archive Log Files to Amazon S3 \(p. 134\)](#)
- [Enable the Debugging Tool \(p. 136\)](#)

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 \(p. 446\)](#). 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/1/ for the first step, /mnt/var/log/hadoop/steps/2/, for the second step, and so on.
- **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

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. How to do this is shown in the following procedures.

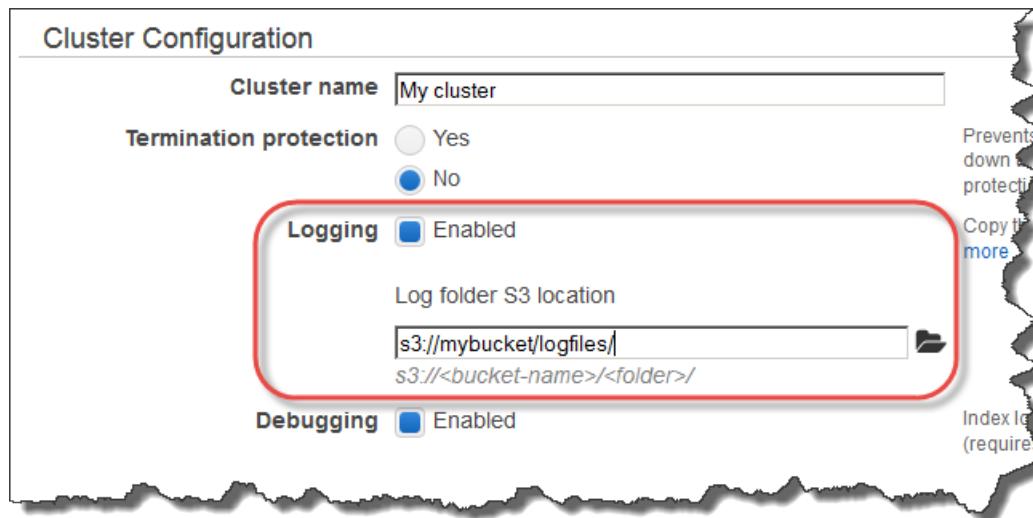
To archive log files to Amazon S3 using the console

1. Open the Amazon Elastic MapReduce console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Click **Create cluster**.
3. In the **Cluster Configuration** section, in the **Logging** field, choose **Enabled**.

This determines whether Amazon EMR will capture detailed log data to Amazon S3.

For more information, see [View Log Files \(p. 418\)](#).

4. In the **Log folder S3 location** field, enter an Amazon S3 path to store your debug logs.



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 \(p. 418\)](#).

5. Click **Done** and proceed to create the cluster as described in [Plan an Amazon EMR Cluster \(p. 29\)](#).

To archive log files to Amazon S3 using the CLI

- Set the `--log-uri` argument when you launch the cluster and specify a location in Amazon S3. Alternately, you can set this value in the `credentials.json` file that you configured for the CLI. This will cause all of the clusters you launch with the CLI to archive log files to the specified S3 bucket. For more information about `credentials.json`, see "Configuring Credentials" in [Install the Amazon EMR Command Line Interface \(p. 579\)](#).

The following example illustrates creating a cluster that archives log files to Amazon S3. You would replace the value `myawsbucket` with the name of S3 bucket that you own.

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --log-uri s3://myawsbucket
```

- Windows users:

```
ruby elastic-mapreduce --create --log-uri s3://myawsbucket
```

To aggregate logs using the CLI

To enable log aggregation to Amazon S3 using the CLI, you use a bootstrap action at cluster launch and to enable log aggregation and to specify a bucket name:

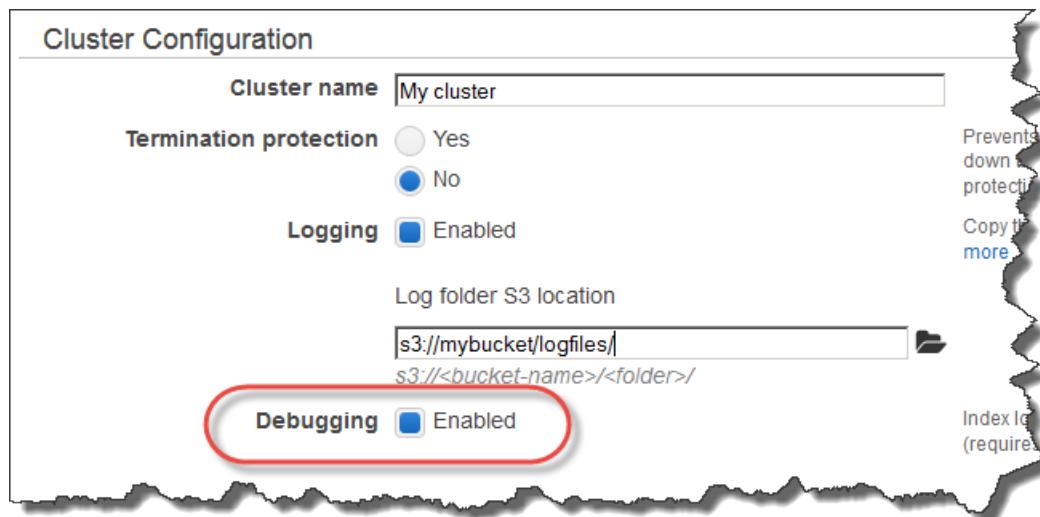
```
./elastic-mapreduce --create --alive --master-instance-type m1.xlarge --slave-instance-type m1.xlarge \
--num-instances 1 --ami-version 3.1.0 --bootstrap-action \
s3://elasticmapreduce/bootstrap-actions/configure-hadoop --args \
"-y,yarn.log-aggregation-enable=true,-y,yarn.log-aggregation.retain-seconds=-1,-y,yarn.log-aggregation.retain-check-interval-seconds=3000, \
-y,yarn.nodemanager.remote-app-log-dir=s3://mys3bucket/logs" \
--ssh --name "log aggregation sub-bucket name"
```

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. An example of using the debugging tool to browse log files is shown in [View the Results \(p. 24\)](#).

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 Elastic MapReduce console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Click **Create cluster**.
3. In the **Cluster Configuration** section, in the **Debugging** field, choose **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 set this when the cluster is created. For more information about Amazon SimpleDB, go to the [Amazon SimpleDB product description page](#).

4. Click **Done** and proceed to create the cluster as described in [Plan an Amazon EMR Cluster \(p. 29\)](#).

To enable the debugging tool using the CLI

- 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 \(p. 579\)](#).

The following example illustrates creating a cluster that archives log files to Amazon S3. You would replace the value `myawsbucket` with the name of S3 bucket that you own.

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --enable-debugging \
--log-uri s3://myawsbucket
```

- Windows users:

```
ruby elastic-mapreduce --create --enable-debugging --log-uri s3://myaws
bucket
```

Select a Amazon VPC Subnet for the Cluster (Optional)

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 Virtual Private Cloud User Guide](#).

Topics

- [Clusters in a VPC \(p. 138\)](#)
- [Setting Up a VPC to Host Clusters \(p. 139\)](#)
- [Launching Clusters into a VPC \(p. 140\)](#)
- [Restricting Permissions to a VPC Using IAM \(p. 141\)](#)

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 \(Amazon VPC\)](#).

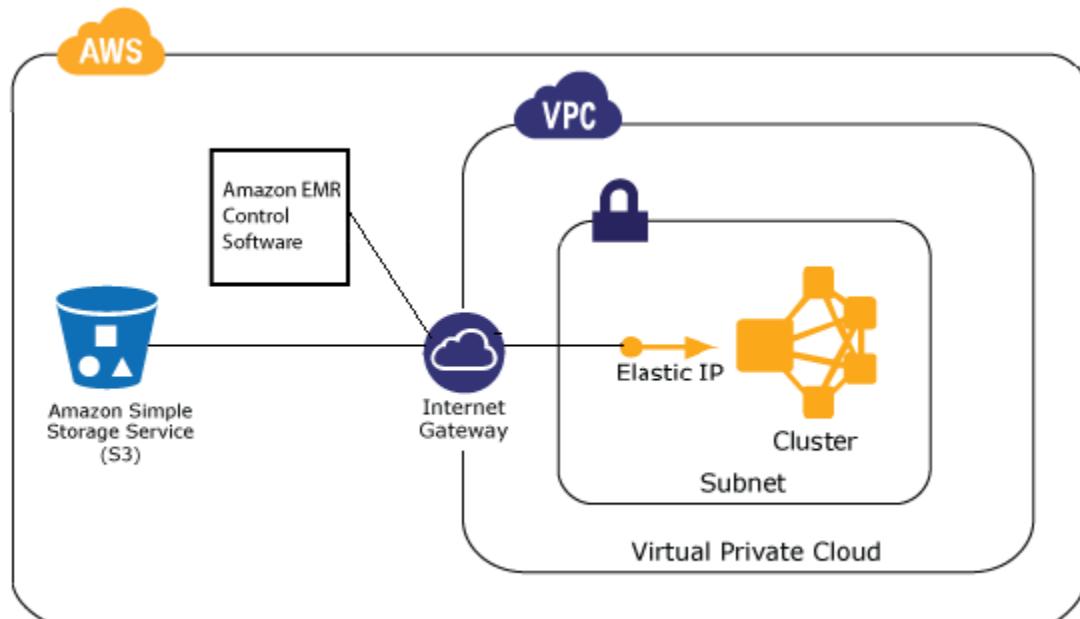
Because access to and from the AWS cloud is a requirement of the cluster, you must connect an Internet gateway to the subnet hosting the cluster. If your application has components you do not want connected to the Internet gateway, you can launch those components in a private subnet you create within 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.

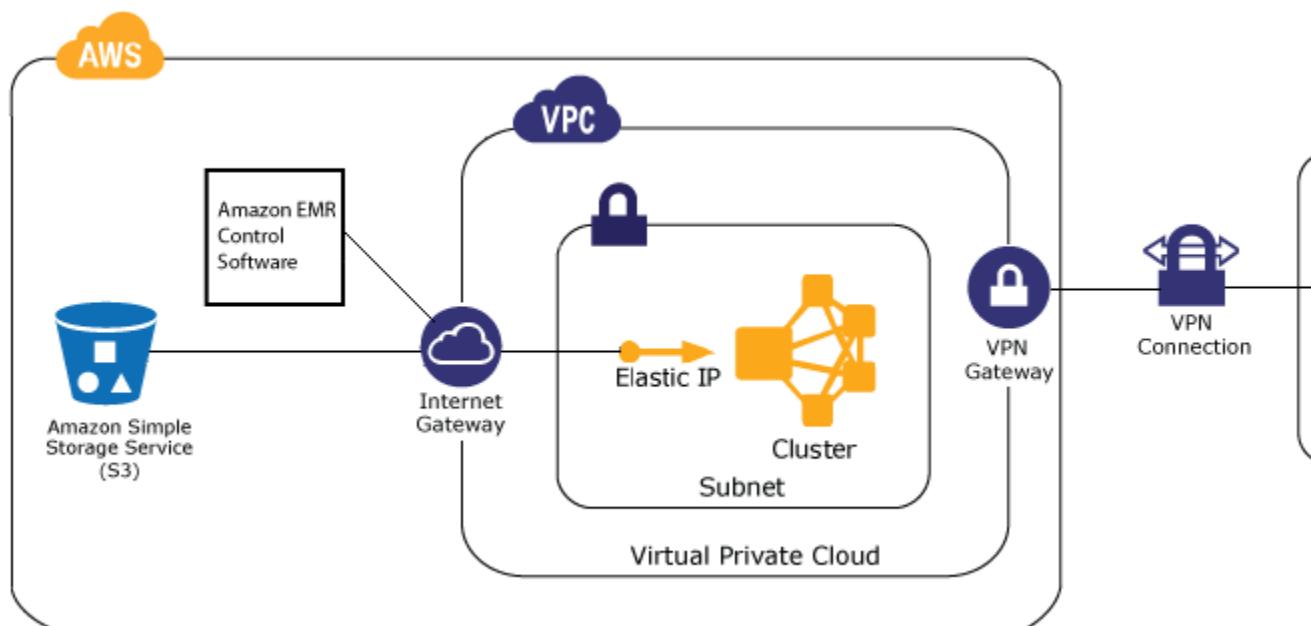
Note

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*	10.0.0/16	(65531 IP addresses available)
VPC name:	My VPC	
Public subnet*	10.0.0/24	(251 IP addresses available)
Availability Zone*	No Preference	
Subnet name:	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*	Default	
<input type="button" value="Cancel and Exit"/> <input type="button" value="Back"/> <input type="button" value="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-1**. For other regions, use **region-name.compute.internal**. For example in **us-west-2**, use **us-west-2.compute.internal**. For 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 Virtual Private Cloud 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 does not allow inbound SSH connections. If you require this functionality, you can add it to the security group.

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 Elastic MapReduce console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Click **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 to create the cluster as described in [Plan an Amazon EMR Cluster \(p. 29\)](#).

To launch a cluster into a VPC using the Amazon EMR CLI

After your VPC is configured, you can launch Amazon EMR clusters on it by using the `--subnet` argument and specifying the subnet address. This is illustrated in the following example, which creates a long-running cluster on the specified subnet. In the directory where you installed the Amazon EMR CLI, run the following from the command line.

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

```
./elastic-mapreduce --create --alive --subnet subnet-identifier
```

- Windows users:

```
ruby elastic-mapreduce --create --alive --subnet subnet-identifier
```

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 [Using IAM](#).

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 Elastic Compute Cloud User Guide*.

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

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 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 Amazon EMR CLI, and the Amazon EMR 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 Elastic Compute Cloud User Guide*. For more information about tagging using the Amazon EC2 API or command line, see [API and CLI Overview](#) in the *Amazon Elastic Compute Cloud User Guide*.

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. For more information on how to create a cluster, see [Choose the Type of Cluster to Run \(p. 32\)](#).

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 as shown in the following image. When the cluster launches, any tags you enter are automatically associated with the cluster.

Key	Value (optional)
Revision	2.0.8
Production	No
CostCenter	30811
<i>Add a key to create a tag</i>	

To add tags when creating a new cluster using the CLI

The following example demonstrates how to add a tag to a new cluster at creation time.

- In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).
 - Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --tag "costCenter=marketing"
```

- Windows users:

```
ruby elastic-mapreduce --create --tag "costCenter=marketing"
```

Adding Tags to an Existing Cluster

You can also add new 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 an existing cluster using the CLI

The following example demonstrates how to add two tags to a cluster. One tag has a key named `production` with no value, and the other tag has a key named `costCenter` with a value of `marketing`.

Note

Quotes are unnecessary when your tag has only a key.

- In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).
 - 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"
```

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 production --tag "costCenter=marketing"
```

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 CLI

- In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).
 - Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --jobflow "j-1234567890123" --list-tags
```

- Windows users:

```
ruby elastic-mapreduce --jobflow "j-1234567890123" --list-tags
```

If the command completes successfully, the output displays all the tag information about the cluster similar to the following:

Key: id	Value: 2785
Key: costCenter	Value: marketing

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 CLI

The following example demonstrates how to remove one tag from a cluster.

- In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).
 - 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
```

Use Third Party Applications With Amazon EMR (Optional)

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.

Product	Description
HParse	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. In addition to text and XML, HParse can extract and convert data stored in proprietary formats such as PDF and Word files. For more information about running HParsers with Amazon EMR, see Parse Data with HParse (p. 148) .
MapR Distribution for Hadoop	An open, enterprise-grade distribution that makes Hadoop easier 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, data protection with snapshots and disaster recovery and with cross-cluster mirroring. For more information about using MapR with Amazon EMR, see Using the MapR Distribution for Hadoop (p. 149) .

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-Quickstart1.pdf>.

Parse Data with HParse

Informatica's HParse 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 HParse to read the text files and extract the relevant data as XML. In addition to text and XML, HParse 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** (v2.1.3)—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** (v2.1.3)—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** (v3.0.2)—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.

MapR does not support Hadoop 2.x.

Launch an Amazon EMR cluster with MapR using the console

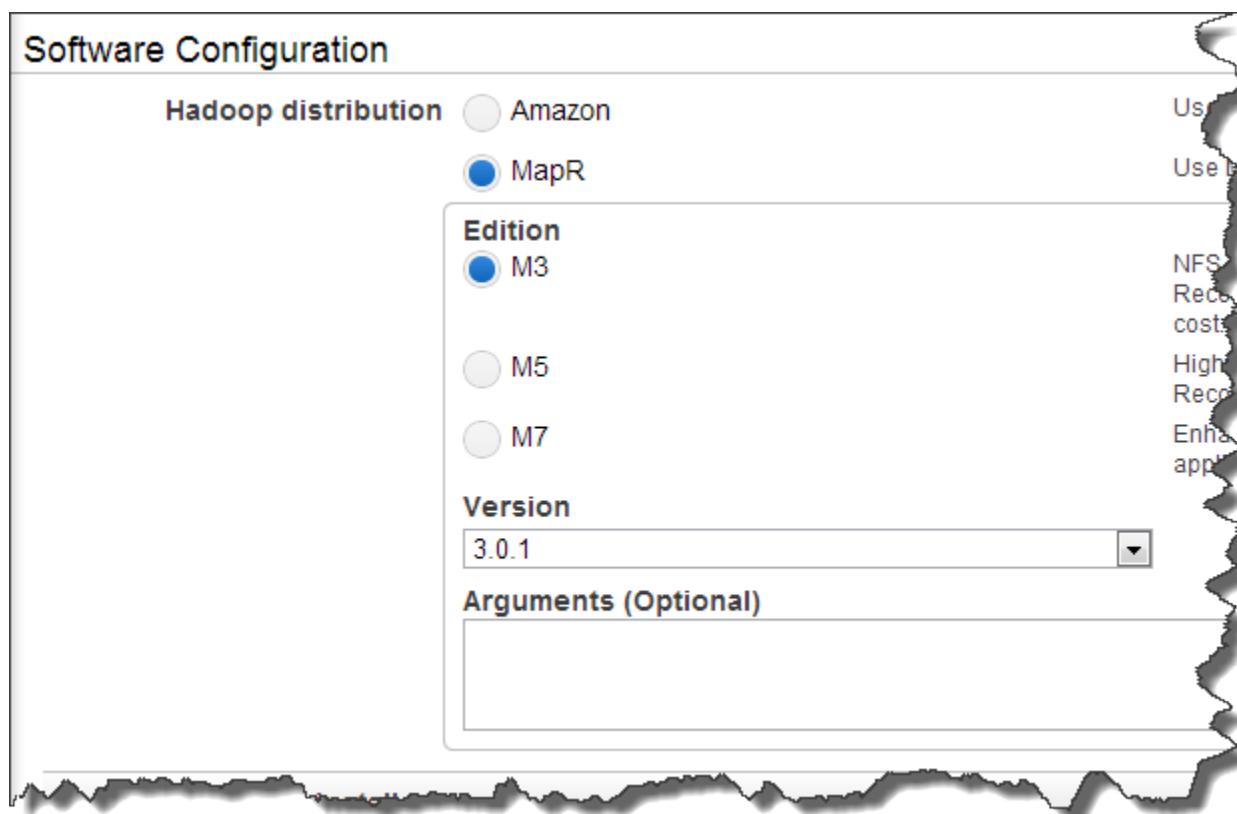
You can launch any standard cluster on a MapR distribution by specifying MapR when you set the Hadoop version.

To launch an Amazon EMR cluster with MapR using the console

1. Open the Amazon Elastic MapReduce console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Click **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. 149) .
Edition	Choose the MapR edition that meets your requirements. For more information, see Using the MapR Distribution for Hadoop (p. 149) .

Field	Action
Version	Choose the MapR version that meets your requirements. For more information, see Versions (p. 152) .
Arguments (Optional)	Specify any additional arguments to pass to MapR to meet your requirements. For more information, see Arguments (p. 152) .

4. Click **Done** and proceed to create the cluster as described in [Plan an Amazon EMR Cluster \(p. 29\)](#).

Launch an Amazon EMR cluster with MapR using the CLI

To launch an Amazon EMR cluster with MapR using the CLI

- In the directory where you installed the Amazon EMR CLI, specify the MapR edition and version by passing arguments with the `--args` option. For more information, see [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).
 - Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --alive \
--instance-type [instance type] --num-instances [number] \
--supported-product [mapr-edition] --args "--edition,<edition label>,-version,<version number>"
```

- Windows users:

```
ruby elastic-mapreduce --create --alive --instance-type [instance type]
--num-instances [number] --supported-product [mapr-edition] --args "--edition,<edition label>,-version,<version number>"
```

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 \(p. 29\)](#). For more information about launching clusters using the API, see [Use SDKs to Call Amazon EMR APIs \(p. 512\)](#).

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:

- 1.2.8
- 2.1.3
- 3.0.2

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, e.g. <code>m3</code> , <code>m5</code> , <code>m7</code> .
<code>--version</code>	The MapR version, e.g. <code>1.2.8</code> , <code>2.1.3</code> , <code>3.0.2</code> .
<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 <code>m5/m7</code> 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</code> (deprecated)	<code>--edition m3</code>
<code>--with-supported-products mapr-m5</code> (deprecated)	<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>
<code>--supported-product mapr --args "--edition,m5 --key1 value1"</code>	<code>--edition m5 --key1 value1</code>

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/hadoop0.20.2/hadoop0.20.2devexamples.jar wordcount
/mapr
```

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
```

Note

The ability to use standard Linux tools such as `echo` and `cat` in this example are made possible by MapR's ability to mount the cluster on NFS at `/mapr/MapR_EMR.amazonaws.com`.

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 [Amazon EMR CLI](#) version 2013-03-19 or newer.

1. Use the Amazon EMR CLI to launch a cluster from the command line using the following command:

```
$ ./elastic-mapreduce --create --alive \
--instance-type m1.xlarge \
--num-instances 5 \
--supported-product mapr \
--name m7 \
--args "--edition,m7"
```

2. List the job flows to find the access point, then use SSH to connect.

```
$ ./elastic-mapreduce --list --active
j-XXXXXXXXXXXXX WAITING hostname.amazonaws.com m7

$ ssh -i KeyName.pem hadoop@hostname.amazonaws.com
```

After you run the command, the cluster takes between five and ten minutes to start. The `elastic-mapreduce --list` command shows your cluster in the STARTING and BOOTSTRAPPING states before entering the WAITING state.

Note

For more information about accessing a cluster with SSH, see [Getting a Key Pair](#).

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. To load data into your cluster, you can use the Yahoo! Cloud Serving Benchmark (YCSB) tool, which you must download and install. For more information, see [Getting Started with YCSB](#). The YCSB tool can run without modification because MapR tables are binary-compatible with Apache HBase tables. This procedure doesn't use YCSB to provide a proper benchmark, only for data-loading purposes, and assumes that the `usertable` table and the `family` column family created in a previous procedure exist on the cluster.

- a. Add or update the following entry to the `core-site.xml` file to map the table requests to the `/tables` directory:

```
<property>
<name>hbase.table.namespace.mappings</name>
<value>*: /tables</value>
</property>
```

- b. Download and open YCSB:

```
$ curl -O http://cloud.github.com/downloads/brianfrankcooper/YCSB/ ycsb-0.1.4.tar.gz
$ tar xzvf ycsb-0.1.4.tar.gz
$ cd ycsb-0.1.4
```

- c. Use the following command to load data:

Note

This command performs ten million inserts, which can take up to ten minutes to complete. Using a smaller value for the `recordcount` parameter shortens the run time for this command.

```
$ HBASE_CLASSPATH=core/lib/core-0.1.4.jar:hbase-binding/lib/hbase-binding-0.1.4.jar \
hbase com.yahoo.ycsb.Client \
-db com.yahoo.ycsb.db.HBaseClient \
-P workloads/workloada \
-p columnfamily=family \
-p recordcount=10000000 \
-load -s > ycsb_output.txt&
```

The `-s` option displays status messages to standard output every 10 seconds, allowing you to track the process. The `ycsb_output.txt` file contains only the final output for the command.

7. 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
```

8. 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.

Nodes							
Overview		<input type="checkbox"/> Filter	Properties	Remove	Manage Services	Change Topology	
Topology		HIdx	Hostname	Physical IP(s)	FS HB	TT HB	Physical Topology
Cluster			qa-node101.qa.prv	10.10.100.101	0s ago	2s ago	/rack1/qa-node101.qa.prv
	rack1		qa-node102.qa.prv	10.10.100.102	0s ago	2s ago	/rack2/qa-node102.qa.prv
	rack2		qa-node103.qa.prv	10.10.100.103	0s ago	2s ago	/rack3/qa-node103.qa.prv
	rack3		qa-node104.qa.prv	10.10.100.104	0s ago	2s ago	/rack4/qa-node104.qa.prv
	rack4		qa-node105.qa.prv	10.10.100.105	0s ago	2s ago	/rack5/qa-node105.qa.prv
	rack5		qa-node111.qa.prv	10.10.100.111	0s ago	2s ago	/rack1/qa-node111.qa.prv
			qa-node112.qa.prv	10.10.100.112	0s ago	2s ago	/rack2/qa-node112.qa.prv
			qa-node113.qa.prv	10.10.100.113	0s ago	2s ago	/rack3/qa-node113.qa.prv
			qa-node114.qa.prv	10.10.100.114	0s ago	2s ago	/rack4/qa-node114.qa.prv
			qa-node115.qa.prv	10.10.100.115	0s ago	2s ago	/rack5/qa-node115.qa.prv
			qa-node121.qa.prv	10.10.100.121	0s ago	2s ago	/rack1/qa-node121.qa.prv
			qa-node122.qa.prv	10.10.100.122	0s ago	2s ago	/rack2/qa-node122.qa.prv
			qa-node123.qa.prv	10.10.100.123	0s ago	2s ago	/rack3/qa-node123.qa.prv
			qa-node124.qa.prv	10.10.100.124	0s ago	2s ago	/rack4/qa-node124.qa.prv
			qa-node125.qa.prv	10.10.100.125	0s ago	2s ago	/rack5/qa-node125.qa.prv

Also, you can see the following from the Volumes view:

Volumes										
<input type="checkbox"/>	Mnt	Vol Name	Mount Path	Creator	Quota	Vol Size	Snap Size	Total Size	Replication Factor	Physical Topology
<input checked="" type="checkbox"/>		ATS-Run-2011-03	/ATS-Run-2011-03	root	none	1.9TB	0	1.9TB	3	/
<input checked="" type="checkbox"/>		mapr.clldb.internal	/root	root	none	1.0MB	0	1.0MB	3	/
<input checked="" type="checkbox"/>		mapr.cluster.root	/	root	none	0	0	0	3	/
<input checked="" type="checkbox"/>		mapr.hbase	/hbase	root	none	0	0	0	3	/
<input checked="" type="checkbox"/>		mapr.jobtracker.vol	/var/mapr/cluster/mr	root	none	10.0MB	0	10.0MB	3	/
<input checked="" type="checkbox"/>		mapr.qa-node101.c	/var/mapr/local/qa-n	root	none	0	0	0	1	/rack1/qa-node101.qa.prv
<input checked="" type="checkbox"/>		mapr.qa-node101.c	/var/mapr/local/qa-n	root	none	23.0MB	0	23.0MB	1	/rack1/qa-node101.qa.prv
<input checked="" type="checkbox"/>		mapr.qa-node102.c	/var/mapr/local/qa-n	root	none	0	0	0	1	/rack2/qa-node102.qa.prv
<input checked="" type="checkbox"/>		mapr.qa-node102.c	/var/mapr/local/qa-n	root	none	23.0MB	0	23.0MB	1	/rack2/qa-node102.qa.prv
<input checked="" type="checkbox"/>		mapr.qa-node103.c	/var/mapr/local/qa-n	root	none	0	0	0	1	/rack3/qa-node103.qa.prv
<input checked="" type="checkbox"/>		mapr.qa-node103.c	/var/mapr/local/qa-n	root	none	23.0MB	0	23.0MB	1	/rack3/qa-node103.qa.prv
<input checked="" type="checkbox"/>		mapr.qa-node104.c	/var/mapr/local/qa-n	root	none	0	0	0	1	/rack4/qa-node104.qa.prv
<input checked="" type="checkbox"/>		mapr.qa-node104.c	/var/mapr/local/qa-n	root	none	25.0MB	0	25.0MB	1	/rack4/qa-node104.qa.prv
<input checked="" type="checkbox"/>		mapr.qa-node105.c	/var/mapr/local/qa-n	root	none	0	0	0	1	/rack5/qa-node105.qa.prv
<input checked="" type="checkbox"/>		mapr.qa-node105.c	/var/mapr/local/qa-n	root	none	18.0MB	0	18.0MB	1	/rack5/qa-node105.qa.prv
<input checked="" type="checkbox"/>		mapr.qa-node111.c	/var/mapr/local/qa-n	root	none	0	0	0	1	/rack1/qa-node111.qa.prv
<input checked="" type="checkbox"/>		mapr.qa-node111.c	/var/mapr/local/qa-n	root	none	21.0MB	0	21.0MB	1	/rack1/qa-node111.qa.prv
<input checked="" type="checkbox"/>		mapr.qa-node112.c	/var/mapr/local/qa-n	root	none	0	0	0	1	/rack2/qa-node112.qa.prv
<input checked="" type="checkbox"/>		mapr.qa-node112.c	/var/mapr/local/qa-n	root	none	25.0MB	0	25.0MB	1	/rack2/qa-node112.qa.prv
<input checked="" type="checkbox"/>		mapr.qa-node113.c	/var/mapr/local/qa-n	root	none	0	0	0	1	/rack3/qa-node113.qa.prv
<input checked="" type="checkbox"/>		mapr.qa-node113.c	/var/mapr/local/qa-n	root	none	18.0MB	0	18.0MB	1	/rack3/qa-node113.qa.prv
<input checked="" type="checkbox"/>		mapr.qa-node114.c	/var/mapr/local/qa-n	root	none	0	0	0	1	/rack4/qa-node114.qa.prv
<input checked="" type="checkbox"/>		mapr.qa-node114.c	/var/mapr/local/qa-n	root	none	19.0MB	0	19.0MB	1	/rack4/qa-node114.qa.prv
<input checked="" type="checkbox"/>		mapr.qa-node121.c	/var/mapr/local/qa-n	root	none	0	0	0	1	/rack1/qa-node121.qa.prv
<input checked="" type="checkbox"/>		mapr.qa-node121.c	/var/mapr/local/qa-n	root	none	22.0MB	0	22.0MB	1	/rack1/qa-node121.qa.prv
<input checked="" type="checkbox"/>		mapr.qa-node122.c	/var/mapr/local/qa-n	root	none	0	0	0	1	/rack2/qa-node122.qa.prv
<input checked="" type="checkbox"/>		mapr.qa-node122.c	/var/mapr/local/qa-n	root	none	26.0MB	0	26.0MB	1	/rack2/qa-node122.qa.prv
<input checked="" type="checkbox"/>		mapr.qa-node123.c	/var/mapr/local/qa-n	root	none	0	0	0	1	/rack3/qa-node123.qa.prv
<input checked="" type="checkbox"/>		mapr.qa-node123.c	/var/mapr/local/qa-n	root	none	19.0MB	0	19.0MB	1	/rack3/qa-node123.qa.prv

Select: Visible All None **0 of 129 items selected**

1 - 50 of 129

Run a Hadoop Application to Process Data

Amazon EMR provides two models for creating custom Hadoop applications to process data:

- **Custom JAR or Cascading cluster** — 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 cluster** — 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 \(p. 161\)](#)
- [JAR Requirements \(p. 166\)](#)
- [Run a Script in a Cluster \(p. 166\)](#)
- [Process Data with a Streaming Cluster \(p. 167\)](#)
- [Process Data with a Cascading Cluster \(p. 177\)](#)
- [Process Data with a Custom JAR Cluster \(p. 192\)](#)

Build Binaries Using 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 Debian used by Amazon EMR. For a 32-bit version, (m1.small and m1.medium) 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 [Virtual Server Configurations \(p. 36\)](#). 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	Create an interactive cluster.
2	Identify the cluster ID and Public DNS name of the master node.
3	SSH as the Hadoop user to the master node of your Hadoop cluster.
4	Copy source files to the master node.
5	Build binaries with any necessary optimizations.
6	Copy binaries from the master node to Amazon S3.
7	Close the SSH connection.
8	Terminate the cluster.

The details for each of these steps are covered in the sections that follow.

To create an interactive cluster

- Create an interactive cluster with a single node Hadoop cluster using the desired instance type:

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- 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
```

The output looks similar to:

```
Created jobflow JobFlowID
```

To identify the cluster ID and Public DNS name of the master node

- Identify your cluster:
 - 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
Interactive Cluster	PENDING	Hive Job

The response includes the cluster ID and the public DNS name. You use this information to connect to the master node.

Typically you need to wait one or two minutes after launching the cluster before the public DNS name is assigned.

To SSH as the Hadoop user to the master node

- Use your credentials created for your Amazon EC2 key pair to log in to the master node: Instructions for creating credentials are located at [Configuring Credentials \(p. 581\)](#).
- In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).
 - Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --ssh --jobflow JobFlowID
```

- Windows users:
 - a. Start PuTTY.
 - b. Select **Session** in the **Category** list. Enter **hadoop@DNS** in the **Host Name** field. In this example, the input looks similar to **hadoop@ec2-75-101-168-82.compute-1.amazonaws.com**.
 - c. In the **Category** list, expand **Connection**, expand **SSH**, and then select **Auth**. The **Options controlling SSH authentication** pane appears.
 - d. Click **Browse** for **Private key file for authentication**, and select the private key file you generated earlier. If you are following this guide, the file name is **mykeypair.ppk**.
 - e. Click **OK**.
 - f. Click **Open** to connect to your master node.
 - g. A **PuTTY Security Alert** pops up. Click **Yes**.

When you successfully connect to the master node, the output looks similar to the following:

```
Using username "hadoop".
Authenticating with public key "imported-openssh-key"
Linux domU-12-31-39-01-5C-F8 2.6.21.7-2.fc8xen #1 SMP Fri Feb 15 12:39:36
EST 2008 i686
-----
```

```
-----  
Welcome to Amazon EMR running Hadoop and Amazon Linux.  
Hadoop is installed in /home/hadoop. Log files are in /mnt/var/log/hadoop.  
Check  
/mnt/var/log/hadoop/steps for diagnosing step failures.  
The Hadoop UI can be accessed via the following commands:  
ResourceManager      lynx http://localhost:9026/  
NameNode              lynx http://localhost:9101/  
-----  
-----
```

To copy source files to the master node

- Copy your source files to the master node:
 - a. Put your source files on Amazon S3. To learn how to create buckets and move files with Amazon S3, go to the [Amazon Simple Storage Service Getting Started Guide](#).
 - b. Create a folder on your Hadoop cluster for your source files by entering a command similar to the following:

```
mkdir Source Destination
```

You now have a destination folder for your source files.

- Copy your sources files from Amazon S3 to the Hadoop cluster by entering a command similar to the following:

```
hadoop fs -get s3://myawsbucket/SourceFiles SourceDestination
```

Your source files are now located in your destination folder on the master node of your Hadoop cluster.

Build binaries with any necessary optimizations

How you build your binaries code 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 dca 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

- Copy the binaries to Amazon S3 by entering the following command:

```
hadoop fs -put BinaryFiles s3://myawsbucket/BinaryDestination
```

Your binaries are now stored in your Amazon S3 bucket.

To close the SSH connection

- Enter the following command from the Hadoop command-line prompt:

```
• exit
```

You are no longer connected to your cluster via SSH.

To terminate the cluster

- In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).
 - Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --terminate JobFlowID
```

- Windows users:

```
ruby elastic-mapreduce --terminate JobFlowID
```

Your cluster is terminated.

Important

Terminating a cluster delete all files and executables saved to the cluster. Remember to save all required files before terminating a cluster.

JAR Requirements

When you launch an Amazon Elastic MapReduce (Amazon EMR) cluster or add steps to a cluster and specify a JAR, Amazon EMR starts Hadoop and then executes your JAR using the `bin/hadoop jar` command. For more information go to http://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html.

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; and that if it is the final step, it is safe to shut down Hadoop.

For more information about how to create a MapReduce executable for your JAR, go to [Hadoop Map/Reduce Tutorial](#).

Run a Script in a Cluster

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 [Add Steps to a Cluster \(p. 473\)](#). For more information about running an interactive cluster, go to [Interactive and Batch Hive Clusters \(p. 231\)](#).

If you want to run a script before step processing begins, use a bootstrap action. For more information on bootstrap actions, go to [Create Bootstrap Actions to Install Additional Software \(Optional\) \(p. 87\)](#).

If you want to run a script immediately before cluster shutdown, use a shutdown action. For more information about shutdown actions, go to [Shutdown Actions \(p. 90\)](#).

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:

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- 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://myawsbucket/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://my
awsbucket/script-path/my_script.sh"
```

This cluster runs the script `my_script.sh` on the master node when the step is processed.

Process Data with a Streaming Cluster

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 stream Hadoop.

Note

Apache Hadoop Streaming is an independent tool. As such, we do not describe all of its functions and parameters. For a complete description of Apache Hadoop Streaming, go to <http://hadoop.apache.org/docs/r1.1.2/streaming.html>.

Using the Hadoop Stream Utility

This section describes how to use the 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 program.

Each mapper script launches as a separate process in the Hadoop 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 Hadoop stream clusters. The following table describes these and other, optional parameters.

Parameter	Description	Required
<code>-input</code>	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
<code>-output</code>	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
<code>-mapper</code>	Name of the mapper executable. Type: String Default: None	Yes
<code>-reducer</code>	Name of the reducer executable. Type: String Default: None	Yes
<code>-cacheFile</code>	An Amazon S3 location containing files for Hadoop to copy into your job flow's local working directory, primarily to improve performance. Type: String Default: None Constraints: [URI]#[symlink name to create in working directory]	No
<code>-cacheArchive</code>	JAR file to extract into the working directory Type: String Default: None Constraints: [URI]#[symlink directory name to create in working directory]	No
<code>-combiner</code>	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. For a complete description of this sample application, see [Get Started: Count Words with Amazon EMR \(p. 13\)](#).

```
#!/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)
```

Launch a Streaming Cluster

This section covers the basics of creating and launching a streaming cluster using Amazon Elastic MapReduce (Amazon EMR). You'll step through how to create a streaming cluster using either the Amazon EMR console, the CLI, or the Query API. Before you create your cluster you'll need to create objects and permissions; for more information see [Prepare Input Data \(Optional\) \(p. 98\)](#).

A streaming cluster 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 clusters, where the output of one streaming cluster becomes the input of another cluster.

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.

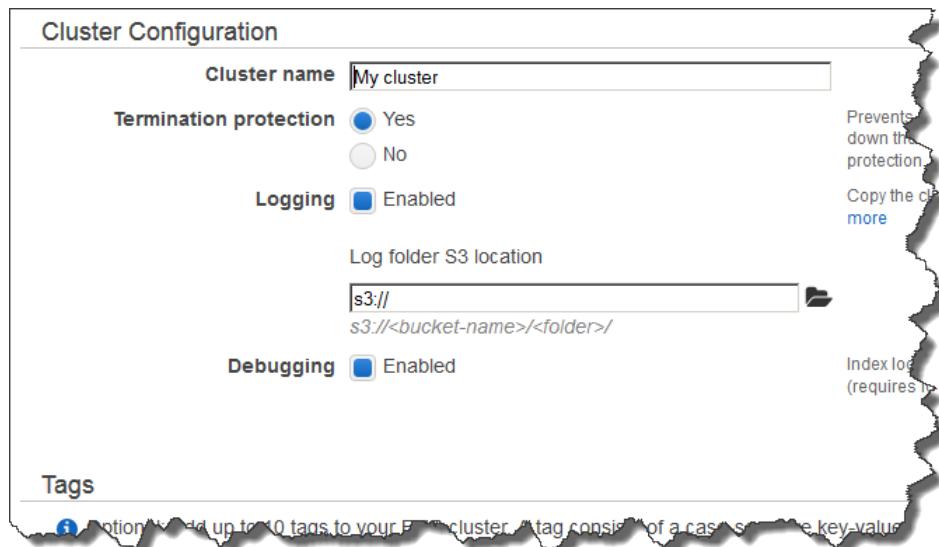
The example that follows is based on the Amazon EMR [Word Count Example](#). This example shows how to use Hadoop streaming to count the number of times each word occurs within a text file. In this example, the input is located in the Amazon S3 bucket `s3n://elasticmapreduce/samples/wordcount/input`. The mapper is a Python script that counts the number of times a word occurs in each input string and is located at `s3://elasticmapreduce/samples/wordcount/wordSplitter.py`. The reducer references a standard Hadoop library package called `aggregate`. Aggregate provides a special Java class and a list of simple aggregators that perform aggregations such as sum, max, and min over a sequence of values. The output is saved to an Amazon S3 bucket you created in [Prepare an Output Location \(Optional\) \(p. 112\)](#).

Amazon EMR Console

This example describes how to use the Amazon EMR console to create a streaming cluster.

To create a streaming cluster

1. Open the Amazon Elastic MapReduce 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>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 (p. 459). 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>This determines whether Amazon EMR captures detailed log data to Amazon S3.</p> <p>For more information, see View Log Files (p. 418).</p>
Log folder S3 location	<p>Enter an Amazon S3 path to store your debug logs if you enabled logging in the previous field. If the log folder does not exist, the Amazon EMR console creates it for you.</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 (p. 418).</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.

Software Configuration

Hadoop distribution Amazon Use Amazon's Hadoop distribution. [Learn more](#)

AMI version Determines the base configuration of the instances in your cluster, including the Hadoop version. [Learn more](#)

MapR Use MapR's Hadoop distribution. [Learn more](#)

Applications to be installed	Version	Actions
Hive	0.11.0.1	  
Pig	0.11.1.1	  
Additional applications	<input type="text" value="Select an application"/> <input type="button" value="Configure and add"/>	

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. 149).</p>
AMI version	<p>Choose 2.4.2 (Hadoop 1.0.3).</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 (p. 51).</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, 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](#).

Hardware Configuration

Specify the networking and hardware configuration for your cluster. If you need more than 20 EC2 Request Spot instances (unused EC2 capacity) to save money.

Network: vpc-c1a3b3a3 (172.31.0.0/16) (default) Use a Virtual Private Subnet

EC2 Subnet: No preference (random subnet) Create a Subnet

EC2 instance type	Count	Request spot		
Master	m1.small	1	<input type="checkbox"/>	The Master instance is the task nodes, and
Core	m1.small	2	<input type="checkbox"/>	Core instances are the Hadoop DataNodes
Task	m1.small	0	<input type="checkbox"/>	Task instances are the Hadoop MapReduce

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 a Amazon VPC Subnet for the Cluster (Optional) (p. 137).</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 Elastic Compute Cloud User Guide</i>.</p>
Master	<p>Choose m1.small.</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.small (default), m1.large, m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cc1.4xlarge, cg1.4xlarge.</p> <p> This tutorial uses small instances for all nodes due to the light workload and to keep your costs low.</p> <p> For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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 Lower Costs with Spot Instances (Optional) (p. 37).</p>

Field	Action
Core	<p>Choose m1.small.</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.small (default), m1.large, m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cc1.4xlarge, cg1.4xlarge.</p> <p>This tutorial uses small instances for all nodes due to the light workload and to keep your costs low.</p> <p>For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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) (p. 37).</p>
Task	<p>Choose m1.small.</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. Valid types: m1.small (default), m1.large, m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cc1.4xlarge, cg1.4xlarge.</p> <p>For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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) (p. 37).</p>

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

Security and Access

EC2 key pair	<input type="button" value="Proceed without an EC2 key pair"/>	Use an existing key pair to SSH into the master node of the Amazon EC2 cluster as the user "hadoop". Learn more
IAM user access		<input type="radio"/> All other IAM users <input checked="" type="radio"/> No other IAM users
Control the visibility of this cluster to other IAM users. Learn more		
IAM Roles		
<p>EMR role <input type="button" value="No roles found"/> Create Default Role Allows EMR to access other AWS Services such as EC2 on your behalf. Learn more</p> <p>EC2 instance profile <input type="button" value="Proceed without role"/> Create Default Role Allows EC2 instances in an EMR cluster to access other AWS services such as S3. Learn more</p>		

Field	Action
EC2 key pair	<p>Choose an Amazon EC2 key pair from the list.</p> <p>For more information, see Create an Amazon EC2 Key Pair and PEM File (p. 117).</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 Cluster (p. 446).</p>
IAM user access	<p>Choose No other IAM users.</p> <p> 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 (p. 119).</p>
EC2 instance profile	<p>You can proceed without choosing an instance profile. If you create a cluster without selecting a specific instance profile, one will be created for you.</p> <p>This controls application access to the Amazon EC2 instances in the cluster.</p> <p>For more information, see Configure IAM Roles for Amazon EMR (p. 125).</p>
EMR role	<p>Choose Proceed without role.</p> <p>Allows Amazon EMR to access other AWS services on your behalf.</p> <p>For more information, see Configure IAM Roles for Amazon EMR (p. 125).</p>

7. 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 [Create Bootstrap Actions to Install Additional Software \(Optional\) \(p. 87\)](#).

8. In the **Steps** section, choose **Streaming Program** from the list and click **Configure and add**.

In the **Add Step** dialog, specify the cluster parameters using the following table as a guide, and then click **Add**.

Add Step

<p>Step Type Streaming Program</p> <p>Name* <input type="text" value="Streaming Program"/></p> <p>Mapper S3 Location* <input type="text" value="s3://elasticmapreduce/samples/wordcount/wordSplitter.py"/></p> <p>Reducer S3 Location* <input type="text" value="aggregate"/></p> <p>Input S3 Location* <input type="text" value="s3://elasticmapreduce/samples/wordcount/input"/></p> <p>Output S3 Location* <input type="text" value="s3://myawsbucket/wordcount/output/2013-09-26"/></p> <p>Arguments <input type="text" value=""/></p> <p>Action on Failure <input type="text" value="Continue"/> <input type="button" value="▼"/></p>
--

Field	Action
Mapper*	Specify either a class name that refers to a mapper class in Hadoop, or a path on Amazon S3 where the mapper executable, such as a Python program, resides. The path value must be in the form <i>BucketName/path/MapperExecutable</i> .
Reducer*	Specify either a class name that refers to a reducer class in Hadoop, or a path on Amazon S3 where the reducer executable, such as a Python program, resides. The path value must be in the form <i>BucketName/path/ReducerExecutable</i> . Amazon EMR supports the special <i>aggregate</i> keyword. For more information, go to the Aggregate library supplied by Hadoop.
Input S3 location*	Specify the URI where the input data resides in Amazon S3. The value must be in the form <i>BucketName/path</i> .
Output S3 location*	Specify the URI where you want the output stored in Amazon S3. The value must be in the form <i>BucketName/path</i> .
Arguments	Optionally, enter a list of arguments (space-separated strings) to pass to the Hadoop streaming utility. For example, you can specify additional files to load into the distributed cache.

Field	Action
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.

* Required parameter

9. Review your configuration and if you are satisfied with the settings, click **Create Cluster**.
10. When the cluster starts, the console displays the **Cluster Details** page.

CLI

This example describes how to use the CLI to create a streaming cluster. Replace the red text with your Amazon S3 bucket information.

To create a cluster

- In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

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 --stream --ami-version 3.0.3 \
--instance-type m1.large --arg "-files" --arg "s3://elasticmapre
duce/samples/wordcount/wordSplitter.py" \
--input s3n://elasticmapreduce/samples/wordcount/input --mapper wordSplit
ter.py --reducer aggregate \
--output s3n://myawsbucket/output/2014-01-16
```

- Windows users:

```
ruby elastic-mapreduce --create --stream --ami-version 3.0.3 --instance-
type m1.large --arg "-files" --arg "s3://elasticmapreduce/samples/word
count/wordSplitter.py" --input s3n://elasticmapreduce/samples/wordcount/in
put --mapper wordSplitter.py --reducer aggregate --output s3n://myawsbuck
et/output/2014-01-16
```

For Hadoop 1.x, use the following command:

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --stream \
--input s3n://elasticmapreduce/samples/wordcount/input \
--mapper s3://elasticmapreduce/samples/wordcount/wordSplitter.py \
--reducer aggregate \
--output s3n://myawsbucket/output/2014-01-16
```

- Windows users:

```
ruby elastic-mapreduce --create --stream --input s3n://elasticmapreduce/samples/wordcount/input --mapper s3://elasticmapreduce/samples/wordcount/wordSplitter.py --reducer aggregate --output s3n://myawsbucket/output/2014-01-16
```

The output looks similar to the following.

```
Created jobflow JobFlowID
```

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.

Process Data with a Cascading Cluster

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 cluster is treated as a custom JAR in the Amazon EMR console. For more information about Cascading, go to <http://www.cascading.org>.

Topics

- [Launch a Cascading Cluster \(p. 177\)](#)
- [Multitool Cascading Application \(p. 185\)](#)

Launch a Cascading Cluster

This section covers the basics of creating a Cascading cluster using a custom JAR file in Amazon Elastic MapReduce (Amazon EMR). You'll step through how to create a cluster with either the Amazon EMR console, the CLI, or the Query API. Before you create your cluster you'll need to create objects and permissions; for more information see [Prepare Input Data \(Optional\) \(p. 98\)](#).

The examples that follow are based on the Amazon EMR sample: [LogAnalyzer for Amazon CloudFront](#). LogAnalyzer is implemented using [Cascading](#). This sample generates usage reports containing total traffic volume, object popularity, a breakdown of traffic by client IP address, and edge location. Reports are formatted as tab-delimited text files, and saved to the Amazon S3 bucket that you specify.

In this example, the Java JAR is located in an Amazon S3 bucket at `elasticmapreduce/samples/cloudfront/logprocessor.jar`. The input data is located in the Amazon S3 bucket `s3n://elasticmapre-`

duce/samples/cloudfront/input. The output is saved to an Amazon S3 bucket you created as part of [Prepare an Output Location \(Optional\) \(p. 112\)](#).

Amazon EMR Console

This example describes how to use the Amazon EMR console to create a cluster using a custom JAR file.

To create a cluster using Cascading

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/>.

Enter the following information:

Add Bootstrap Action

Action Type	Custom Action
Name	Custom Action
S3 Location	s3://files.cascading.org/sdk/2.1/install-cascading-sdk.sh
Optional Arguments	

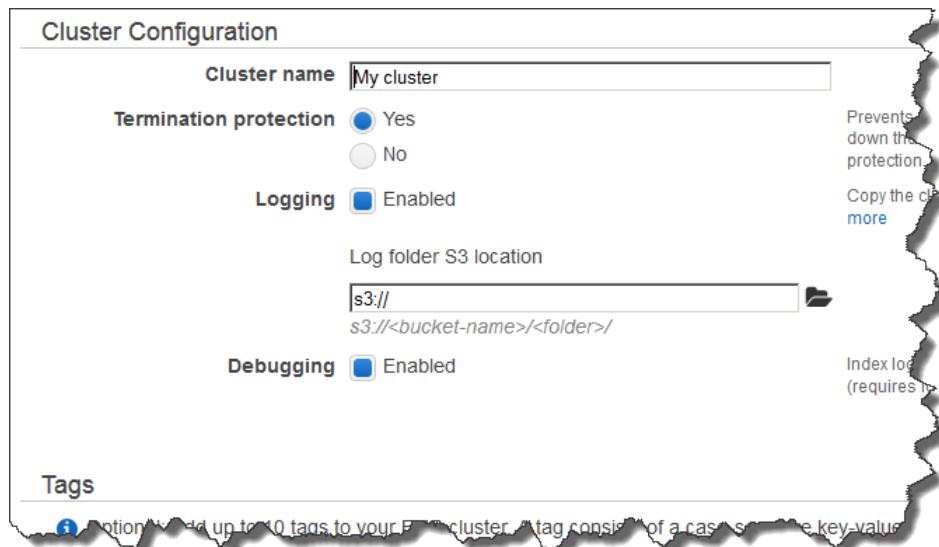
- a. Enter the following text in the **S3 location** field:

```
s3://files.cascading.org/sdk/2.1/install-cascading-sdk.sh
```

- b. Click **Add**.

For more information, see [Create Bootstrap Actions to Install Additional Software \(Optional\) \(p. 87\)](#).

1. Open the Amazon Elastic MapReduce 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>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 (p. 459). 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>This determines whether Amazon EMR captures detailed log data to Amazon S3.</p> <p>For more information, see View Log Files (p. 418).</p>
Log folder S3 location	<p>Enter an Amazon S3 path to store your debug logs if you enabled logging in the previous field. If the log folder does not exist, the Amazon EMR console creates it for you.</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 (p. 418).</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.

Software Configuration

Hadoop distribution Amazon Use Amazon's Hadoop distribution. [Learn more](#)

AMI version Determines the base configuration of the instances in your cluster, including the Hadoop version. [Learn more](#)

MapR Use MapR's Hadoop distribution. [Learn more](#)

Applications to be installed	Version	Actions
Hive	0.11.0.1	  
Pig	0.11.1.1	  
Additional applications	<input type="text" value="Select an application"/> <input type="button" value="Configure and add"/>	

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. 149).</p>
AMI version	<p>Choose 2.4.2 (Hadoop 1.0.3).</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 (p. 51).</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, 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](#).

Hardware Configuration

Specify the networking and hardware configuration for your cluster. If you need more than 20 EC2 Request Spot instances (unused EC2 capacity) to save money.

Network: vpc-c1a3b3a3 (172.31.0.0/16) (default) Use a Virtual Private Subnet

EC2 Subnet: No preference (random subnet) Create a Subnet

EC2 instance type	Count	Request spot		
Master	m1.small	1	<input type="checkbox"/>	The Master instance is the task nodes, and
Core	m1.small	2	<input type="checkbox"/>	Core instances are the Hadoop DataNodes
Task	m1.small	0	<input type="checkbox"/>	Task instances are the Hadoop MapReduce

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 a Amazon VPC Subnet for the Cluster (Optional) (p. 137).</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 Elastic Compute Cloud User Guide</i>.</p>
Master	<p>Choose m1.small.</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.small (default), m1.large, m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cc1.4xlarge, cg1.4xlarge.</p> <p> This tutorial uses small instances for all nodes due to the light workload and to keep your costs low.</p> <p> For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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 Lower Costs with Spot Instances (Optional) (p. 37).</p>

Field	Action
Core	<p>Choose m1.small.</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.small (default), m1.large, m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cc1.4xlarge, cg1.4xlarge.</p> <p>This tutorial uses small instances for all nodes due to the light workload and to keep your costs low.</p> <p>For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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) (p. 37).</p>
Task	<p>Choose m1.small.</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. Valid types: m1.small (default), m1.large, m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cc1.4xlarge, cg1.4xlarge.</p> <p>For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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) (p. 37).</p>

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

Security and Access

EC2 key pair	<input type="button" value="Proceed without an EC2 key pair"/>	Use an existing key pair to SSH into the master node of the Amazon EC2 cluster as the user "hadoop". Learn more
IAM user access		<input type="radio"/> All other IAM users <input checked="" type="radio"/> No other IAM users
IAM Roles		
<p>EMR role <input type="button" value="No roles found"/> Create Default Role</p> <p>EC2 instance profile <input type="button" value="Proceed without role"/> Create Default Role</p>		

Field	Action
EC2 key pair	<p>Choose an Amazon EC2 key pair from the list.</p> <p>For more information, see Create an Amazon EC2 Key Pair and PEM File (p. 117).</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 Cluster (p. 446).</p>
IAM user access	<p>Choose No other IAM users.</p> <p> 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 (p. 119).</p>
EC2 instance profile	<p>You can proceed without choosing an instance profile. If you create a cluster without selecting a specific instance profile, one will be created for you.</p> <p>This controls application access to the Amazon EC2 instances in the cluster.</p> <p>For more information, see Configure IAM Roles for Amazon EMR (p. 125).</p>
EMR role	<p>Choose Proceed without role.</p> <p>Allows Amazon EMR to access other AWS services on your behalf.</p> <p>For more information, see Configure IAM Roles for Amazon EMR (p. 125).</p>

7. In the **Bootstrap Actions** section, in the **Add bootstrap action** field, select **Custom Action** and click **Configure and add**.
8. In the **Steps** section, in the **Add step** field, choose **Custom JAR** from the list and click **Configure and add**.

In the **Add Step** dialog, enter values in the boxes using the following table as a guide, and then click **Add**.

Add Step

Step Type Custom Jar

Name* Custom Jar

JAR S3 Location* `s3://elasticmapreduce/samples/cloudfront/logprocessor.jar`

Arguments

```
-input s3n://elasticmapreduce/samples/cloudfront/input
-output s3n://myawsbucket/cloudfront/output/2012-06-10-00
-start any
-end 2012-06-10-00
-timeBucket 300
-overallVolumeReport
-objectPopularityReport
-clientIPReport
-edgeLocationReport
```

These are the JAR arguments you can pass to the JAR file you created.

Action on Failure Continue

Field	Action
JAR S3 location	Specify the URI where your script resides in Amazon S3. The value must be in the form <i>BucketName/path/ScriptName</i> .
Arguments	Enter a list of arguments (space-separated strings) to pass to the JAR file.
Action on Failure	This determines what the cluster does in response to any errors. The possible values for this setting are: <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.

9. Review your configuration and if you are satisfied with the settings, click **Create Cluster**.
10. When the cluster starts, the console displays the **Cluster Details** page.

CLI

This example describes how to use the CLI to create a cluster using Cascading.

To create a cluster using Cascading

- In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).
 - 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,s3n://elasticmapreduce/samples/cloudfront/input, \
-start,any,-end,2010-12-27-02 300,-output, \
s3n://myawsbucket/cloudfront/output/2010-12-27-02, \
-overallVolumeReport,-objectPopularityReport,-clientIPReport, \
-edgeLocationReport"
```

- 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,s3n://elast
icmapreduce/samples/cloudfront/input,-start,any,-end,2010-12-27-02 300,-
output,s3n://myawsbucket/cloudfront/output/2010-12-27-02,-overallVolumeRe
port,-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/>.

The output looks similar to the following.

```
Created cluster JobFlowID
```

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.

Multitool Cascading Application

Multitool is a Cascading application that provides a simple command line interface for managing large datasets. For example, you can filter records matching a Java regular expression from data stored in Amazon S3 and copy the results to the Hadoop file system.

You can run the Cascading Multitool application on Amazon Elastic MapReduce (Amazon EMR) using either the Amazon EMR command line interface or the Amazon EMR console. Amazon EMR supports all Multitool arguments.

The Multitool JAR file is at `s3n://elasticmapreduce/samples/multitool/multitool-aws-03-31-09.jar`. The Multitool source code, along with a number of other tools, is available for download

from the project website at <http://www.cascading.org/modules.html>. For additional samples and tips for using Multitool, go to [Cascading.Multitool - Tips on using the Multitool](#) and [Generate usage reports](#).

To create a Cascading cluster with Multitool using the CLI

- 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, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create \
--jar s3n://elasticmapreduce/samples/multitool/multitool-aws-03-31-09.jar
\
--args [args]
```

- Windows users:

```
ruby elastic-mapreduce --create --jar s3n://elasticmapreduce/samples/mul
titool/multitool-aws-03-31-09.jar --args [args]
```

To create a Cascading cluster with Multitool using the Amazon EMR console

1. Open the Amazon Elastic MapReduce 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.

Setting	Value	Description
Cluster name	My cluster	
Termination protection	<input checked="" type="radio"/> Yes	Prevents cluster from being terminated
Logging	<input checked="" type="checkbox"/> Enabled	Copy the cluster logs to S3
Log folder S3 location	s3://<bucket-name>/<folder>/	
Debugging	<input checked="" type="checkbox"/> Enabled	Index logs (requires Java 6)

Tags
You can add up to 10 tags to your EMR cluster. A tag consists of a case-sensitive key-value pair.

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>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 (p. 459). 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>This determines whether Amazon EMR captures detailed log data to Amazon S3.</p> <p>For more information, see View Log Files (p. 418).</p>
Log folder S3 location	<p>Enter an Amazon S3 path to store your debug logs if you enabled logging in the previous field. If the log folder does not exist, the Amazon EMR console creates it for you.</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 (p. 418).</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.

Software Configuration

Hadoop distribution	<input checked="" type="radio"/> Amazon	Use Amazon's Hadoop distribution. Learn more						
AMI version		Determines the base configuration of the instances in your cluster, including the Hadoop version. Learn more						
<input checked="" type="radio"/> 2.4.2 (hadoop 1.0.3) - latest		<input type="radio"/> MapR						
Applications to be installed <table style="width: 100%; border-collapse: collapse;"> <tr> <th style="width: 60%;">Applications to be installed</th> <th style="width: 40%;">Version</th> </tr> <tr> <td>Hive</td> <td>0.11.0.1</td> </tr> <tr> <td>Pig</td> <td>0.11.1.1</td> </tr> </table>			Applications to be installed	Version	Hive	0.11.0.1	Pig	0.11.1.1
Applications to be installed	Version							
Hive	0.11.0.1							
Pig	0.11.1.1							
Additional applications <div style="border: 1px solid #ccc; padding: 2px; width: 150px; margin-bottom: 5px;"> <input type="text" value="Select an application"/> </div> <div style="border: 1px solid #ccc; padding: 2px; width: 150px; text-align: center;"> <input type="button" value="Configure and add"/> </div>								

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. 149).</p>
AMI version	<p>Choose 2.4.2 (Hadoop 1.0.3).</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 (p. 51).</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, 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](#).

Hardware Configuration

Specify the networking and hardware configuration for your cluster. If you need more than 20 EC2 Request Spot instances (unused EC2 capacity) to save money.

Network	<input type="text" value="vpc-c1a3b3a3 (172.31.0.0/16) (default)"/>	<input type="checkbox"/> Use a Virtual data or conn												
EC2 Subnet	<input type="text" value="No preference (random subnet)"/>	Create a Sub												
<table style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 30%;">EC2 instance type</th> <th style="width: 20%;">Count</th> <th style="width: 50%;">Request spot</th> </tr> </thead> <tbody> <tr> <td>Master</td> <td><input type="text" value="1"/></td> <td><input type="checkbox"/></td> </tr> <tr> <td>Core</td> <td><input type="text" value="2"/></td> <td><input type="checkbox"/></td> </tr> <tr> <td>Task</td> <td><input type="text" value="0"/></td> <td><input type="checkbox"/></td> </tr> </tbody> </table>			EC2 instance type	Count	Request spot	Master	<input type="text" value="1"/>	<input type="checkbox"/>	Core	<input type="text" value="2"/>	<input type="checkbox"/>	Task	<input type="text" value="0"/>	<input type="checkbox"/>
EC2 instance type	Count	Request spot												
Master	<input type="text" value="1"/>	<input type="checkbox"/>												
Core	<input type="text" value="2"/>	<input type="checkbox"/>												
Task	<input type="text" value="0"/>	<input type="checkbox"/>												

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 a Amazon VPC Subnet for the Cluster (Optional) (p. 137).</p>

Field	Action
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 Elastic Compute Cloud User Guide</i>.</p>
Master	<p>Choose m1.small.</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.small (default), m1.large, m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cc1.4xlarge, cg1.4xlarge.</p> <p>This tutorial uses small instances for all nodes due to the light workload and to keep your costs low.</p> <p>For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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 Lower Costs with Spot Instances (Optional) (p. 37).</p>
Core	<p>Choose m1.small.</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.small (default), m1.large, m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cc1.4xlarge, cg1.4xlarge.</p> <p>This tutorial uses small instances for all nodes due to the light workload and to keep your costs low.</p> <p>For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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) (p. 37).</p>

Field	Action
Task	<p>Choose m1.small.</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. Valid types: m1.small (default), m1.large, m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cc1.4xlarge, cg1.4xlarge.</p> <p>For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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) (p. 37).</p>

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

Security and Access

EC2 key pair Use an existing key pair to SSH into the master node of the Amazon EC2 cluster as the user "hadoop". [Learn more](#)

IAM user access All other IAM users No other IAM users Control the visibility of this cluster to other IAM users. [Learn more](#)

IAM Roles

EC2 instance profile Allows EC2 instances in an EMR cluster to access other AWS services such as S3. [Learn more](#)

Field	Action
EC2 key pair	<p>Choose an Amazon EC2 key pair from the list.</p> <p>For more information, see Create an Amazon EC2 Key Pair and PEM File (p. 117).</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 Cluster (p. 446).</p>
IAM user access	<p>Choose No other IAM users.</p> <p> 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 (p. 119).</p>

Field	Action
EC2 instance profile	<p>You can proceed without choosing an instance profile. If you create a cluster without selecting a specific instance profile, one will be created for you.</p> <p>This controls application access to the Amazon EC2 instances in the cluster.</p> <p>For more information, see Configure IAM Roles for Amazon EMR (p. 125).</p>
EMR role	<p>Choose Proceed without role.</p> <p>Allows Amazon EMR to access other AWS services on your behalf.</p> <p>For more information, see Configure IAM Roles for Amazon EMR (p. 125).</p>

7. In the **Bootstrap Actions** section, there are no bootstrap actions necessary for this sample configuration.

Optional, 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 \(Optional\) \(p. 87\)](#).

8. In the **Steps** section, in the **Add step** field, choose **Custom JAR** from the list and click **Configure and add**.

In the **Add Step** dialog, specify the cluster parameters:

- a. In the **JAR S3 location** field, specify the path and file name for the Multitool JAR file, for example:

s3n://elasticmapreduce/samples/multitool/multitool-aws-03-31-09.jar

Add Step

Step Type Custom Jar

Name* Custom Jar

JAR S3 Location* s3n://elasticmapreduce/samples/multitool/multitool-aws-03-

Arguments

These are passed to the main function of the JAR. If the JAR does not specify a main class, you can specify another class or argument.

Action on Failure

Skip step and continue



What to do if the Step fails

- b. Specify any arguments for the cluster.

All Multitool arguments are supported, including those listed in the following table.

Parameter	Description
<code>-input</code>	Location of input file.
<code>-output</code>	Location of output files.
<code>-start</code>	Use data created after the start time.
<code>-end</code>	Use data created by the end time.

9. Review your configuration and if you are satisfied with the settings, click **Create Cluster**.
10. When the cluster starts, the console displays the **Cluster Details** page.

Process Data with a Custom JAR Cluster

A custom JAR cluster runs a compiled Java program that you uploaded to Amazon S3. Compile the program against the version of Hadoop you want to launch and submit Hadoop jobs using the Hadoop JobClient interface.

For more information about building a Hadoop main function, go to http://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html and look at the word count example.

Build a JAR file that contains a main function and is linked against Hadoop.

This example cluster uses a python script that counts the words in a file. Before copying the following sample cluster code, replace the variables, `MY_LOG_BUCKET` and `MY_BUCKET` with actual Amazon S3 bucket names and `MY_KEY_NAME` with an Amazon EC2 key name. Save the following example cluster in a file called `custom_jar_jobflow.json`.

```
{
  "Name": "Execute custom jar step",
  "LogUri": "MY_LOG_BUCKET/log",
  "Instances": {
    "SlaveInstanceType": "m1.small",
    "MasterInstanceType": "m1.small",
    "InstanceCount": "1",
    "Ec2KeyName": "MY_KEY_NAME",
    "KeepJobFlowAliveWhenNoSteps": "false"
  },
  "Steps": [
    {
      "Name": "Custom Jar Grep Example",
      "ActionOnFailure": "CONTINUE",
      "HadoopJarStep": {
        "MainClass": "grep",
        "Jar": "MY_BUCKET/demo/custom.jar",
        "Args": [
          "MY_BUCKET/demo/input",
          "MY_BUCKET/output",
          "the"
        ]
      }
    }
  ]
}
```

```
    ]  
}
```

This example cluster specifies what type of hardware to run the cluster on including the number of machines (*InstanceCount*), the place for the cluster to upload logs to (*LogUri*), whether the cluster should remain running after all steps have finished (*KeepJobFlowAliveWhenNoSteps*), and what happens if there is an error (*ActionOnError*). The *Steps* element is an array that enables you to specify 0 or more steps. Amazon Elastic MapReduce (Amazon EMR) executes multiple steps in the order listed in the cluster.

Note

Zero steps means that data is not processed. You might specify zero steps to set up the Hadoop cluster so that you can subsequently use `AddJobFlowSteps` to add steps in a debugging scenario.

Run the cluster.

```
./elastic-mapreduce --json streaming_jobflow.json --jobflow <job_flow_id>
```

The maximum lifetime of a cluster is 2 weeks.

To create a custom JAR cluster

- Bundle a JAR file as follows, presuming your source code is in the `src` directory and you unpacked Hadoop into a directory stored in `$HADOOP_HOME` in one of the following ways:

If you are using...	Do the following...
Linux, UNIX, or Mac OS X	<p>From the command line, enter the following:</p> <pre>elasticmapreduce/samples/cloudburst/cloudburst.jar \ s3n://elasticmapreduce/samples/cloudburst/input/s_suis.br \ s3n://elasticmapreduce/samples/cloudburst/input/100k.br \ s3n://<myawsbucket>/cloudburst/output/ \ 36 3 0 1 240 48 24 24 128 16</pre>
Microsoft Windows	

After you are finished terminate the cluster

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --jobflow JobFlowID --terminate
```

- Windows users:

```
ruby elastic-mapreduce --jobflow JobFlowID --terminate
```

Launch a Custom JAR Cluster

This section covers the basics of creating a cluster using a custom JAR file in Amazon Elastic MapReduce (Amazon EMR). You'll step through how to create a cluster using a Custom JAR with either the Amazon EMR console, the CLI, or the Query API. Before you create your job flow you'll need to create objects and permissions; for more information see [Prepare Input Data \(Optional\) \(p. 98\)](#).

A cluster using a custom JAR file enables you to write a script to process your data using the Java programming language. The example that follows is based on the Amazon EMR sample: [CloudBurst](#).

In this example, the JAR file is located in an Amazon S3 bucket at `s3n://elasticmapreduce/samples/cloudburst/cloudburst.jar`. All of the data processing instructions are located in the JAR file and the script is referenced by the main class `org.myorg.WordCount`. The input data is located in the Amazon S3 bucket `s3n://elasticmapreduce/samples/cloudburst/input`. The output is saved to an Amazon S3 bucket you created as part of [Prepare an Output Location \(Optional\) \(p. 112\)](#).

Amazon EMR Console

This example describes how to use the Amazon EMR console to create a cluster using a custom JAR file.

To create a cluster using a custom JAR file

1. Open the Amazon Elastic MapReduce 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.

Cluster Configuration

Cluster name	My cluster
Termination protection	<input checked="" type="radio"/> Yes
	<input type="radio"/> No
Logging	<input checked="" type="checkbox"/> Enabled
Log folder S3 location	<code>s3://<bucket-name>/<folder>/</code>
Debugging	<input checked="" type="checkbox"/> Enabled

Tags

Information: You can add up to 10 tags to your EMR cluster. A tag consists of a case-sensitive key-value pair.

Field	Action
Cluster name	Enter a descriptive name for your cluster. The name is optional, and does not need to be unique.

Field	Action
Termination protection	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 (p. 459) . 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.
Logging	This determines whether Amazon EMR captures detailed log data to Amazon S3. For more information, see View Log Files (p. 418) .
Log folder S3 location	Enter an Amazon S3 path to store your debug logs if you enabled logging in the previous field. If the log folder does not exist, the Amazon EMR console creates it for you. 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. 418) .
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.

Software Configuration

Hadoop distribution	<input checked="" type="radio"/> Amazon	Use Amazon's Hadoop distribution. Learn more						
AMI version	<input type="text" value="2.4.2 (hadoop 1.0.3) - latest"/>	Determines the base configuration of the instances in your cluster, including the Hadoop version. Learn more						
<input type="radio"/> MapR		Use MapR's Hadoop distribution. Learn more						
<table border="1"> <thead> <tr> <th>Applications to be installed</th> <th>Version</th> </tr> </thead> <tbody> <tr> <td>Hive</td> <td>0.11.0.1</td> </tr> <tr> <td>Pig</td> <td>0.11.1.1</td> </tr> </tbody> </table>			Applications to be installed	Version	Hive	0.11.0.1	Pig	0.11.1.1
Applications to be installed	Version							
Hive	0.11.0.1							
Pig	0.11.1.1							
Additional applications	<input type="text" value="Select an application"/>	Configure and add						

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. 149).</p>
AMI version	<p>Choose 2.4.2 (Hadoop 1.0.3).</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 (p. 51).</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, 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](#).

Hardware Configuration

Specify the networking and hardware configuration for your cluster. If you need more than 20 EC2 Request Spot instances (unused EC2 capacity) to save money.

Network	<input type="text" value="vpc-c1a3b3a3 (172.31.0.0/16) (default)"/>	<input type="checkbox"/> Use a Virtual data or conn												
EC2 Subnet	<input type="text" value="No preference (random subnet)"/>	Create a Sub												
<table style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 30%;">EC2 instance type</th> <th style="width: 20%;">Count</th> <th style="width: 50%;">Request spot</th> </tr> </thead> <tbody> <tr> <td>Master</td> <td><input type="text" value="1"/></td> <td><input type="checkbox"/></td> </tr> <tr> <td>Core</td> <td><input type="text" value="2"/></td> <td><input type="checkbox"/></td> </tr> <tr> <td>Task</td> <td><input type="text" value="0"/></td> <td><input type="checkbox"/></td> </tr> </tbody> </table>			EC2 instance type	Count	Request spot	Master	<input type="text" value="1"/>	<input type="checkbox"/>	Core	<input type="text" value="2"/>	<input type="checkbox"/>	Task	<input type="text" value="0"/>	<input type="checkbox"/>
EC2 instance type	Count	Request spot												
Master	<input type="text" value="1"/>	<input type="checkbox"/>												
Core	<input type="text" value="2"/>	<input type="checkbox"/>												
Task	<input type="text" value="0"/>	<input type="checkbox"/>												

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 a Amazon VPC Subnet for the Cluster (Optional) (p. 137).</p>

Field	Action
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 Elastic Compute Cloud User Guide</i>.</p>
Master	<p>Choose m1.small.</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.small (default), m1.large, m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cc1.4xlarge, cg1.4xlarge.</p> <p>This tutorial uses small instances for all nodes due to the light workload and to keep your costs low.</p> <p>For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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 Lower Costs with Spot Instances (Optional) (p. 37).</p>
Core	<p>Choose m1.small.</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.small (default), m1.large, m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cc1.4xlarge, cg1.4xlarge.</p> <p>This tutorial uses small instances for all nodes due to the light workload and to keep your costs low.</p> <p>For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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) (p. 37).</p>

Field	Action
Task	<p>Choose m1.small.</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. Valid types: m1.small (default), m1.large, m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cc1.4xlarge, cg1.4xlarge.</p> <p>For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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) (p. 37).</p>

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

Security and Access

EC2 key pair

Use an existing key pair to SSH into the master node of the Amazon EC2 cluster as the user "hadoop". [Learn more](#)

IAM user access All other IAM users No other IAM users

Control the visibility of this cluster to other IAM users. [Learn more](#)

IAM Roles

1 An IAM role for the EMR service and an EC2 instance profile for instances in an EMR cluster are recommended. You can create and assign these roles to limit the permissions of the EMR service and applications running on a cluster.

EMR role

Allows EMR to access other AWS Services such as EC2 on your behalf. [Learn more](#)

[Create Default Role](#)

EC2 instance profile

Allows EC2 Instances in an EMR cluster to access other AWS services such as S3. [Learn more](#)

[Create Default Role](#)

Field	Action
EC2 key pair	<p>Choose an Amazon EC2 key pair from the list.</p> <p>For more information, see Create an Amazon EC2 Key Pair and PEM File (p. 117).</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 Cluster (p. 446).</p>
IAM user access	<p>Choose No other IAM users.</p> <p> 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 (p. 119).</p>

Field	Action
EC2 instance profile	<p>You can proceed without choosing an instance profile. If you create a cluster without selecting a specific instance profile, one will be created for you.</p> <p>This controls application access to the Amazon EC2 instances in the cluster.</p> <p>For more information, see Configure IAM Roles for Amazon EMR (p. 125).</p>
EMR role	<p>Choose Proceed without role.</p> <p>Allows Amazon EMR to access other AWS services on your behalf.</p> <p>For more information, see Configure IAM Roles for Amazon EMR (p. 125).</p>

7. 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 [Create Bootstrap Actions to Install Additional Software \(Optional\) \(p. 87\)](#).

8. In the **Steps** section, choose **Custom Jar** from the list and click **Add**.

In the **Add Step** dialog, enter values in the boxes using the following table as a guide, and then click **Add**.

Add Step

Step type Custom JAR

Name*

JAR S3 location*

Arguments

Action on failure

Field	Action
JAR S3 location*	Specify the URI where your script resides in Amazon S3. The value must be in the form <code>s3://BucketName/path/ScriptName</code> .
Arguments*	Enter a list of arguments (space-separated strings) to pass to the JAR file.
Action on Failure	This determines what the cluster does in response to any errors. The possible values for this setting are: <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.

* Required parameter

9. Review your configuration and if you are satisfied with the settings, click **Create Cluster**.
10. When the cluster starts, the console displays the **Cluster Details** page.

CLI

This section explains how to run a cluster that uses a custom JAR file.

To create a cluster using a Custom JAR

- In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).
 - Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --name "Test custom JAR" \
--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 s3n://myawsbucket/cloud \
--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
s3n://elasticmapreduce/samples/cloudburst/cloudburst.jar --arg
s3n://elasticmapreduce/samples/cloudburst/input/s_suis.br --arg
s3n://elasticmapreduce/samples/cloudburst/input/100k.br --arg s3n://myaws
bucket/cloud --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, as shown in the preceding examples.

The output looks similar to the following.

```
Created cluster JobFlowID
```

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.

Analyze Data with Hive

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/>.

Amazon Elastic MapReduce (Amazon EMR) provides support for Apache Hive. Amazon EMR supports several versions of Hive, which you can install on any running cluster. Amazon EMR also allows you to run multiple versions concurrently, allowing you to control your Hive version upgrade. The following sections describe the Hive configurations using Amazon EMR.

Topics

- [How Amazon EMR Hive Differs from Apache Hive \(p. 202\)](#)
- [Supported Hive Versions \(p. 213\)](#)
- [Interactive and Batch Hive Clusters \(p. 231\)](#)
- [Launch a Hive Cluster \(p. 234\)](#)
- [Create a Metastore Outside the Hadoop Cluster \(p. 241\)](#)
- [Use the Hive JDBC Driver \(p. 243\)](#)

How Amazon EMR Hive Differs from Apache Hive

Topics

- [Input Format \(p. 203\)](#)
- [Combine Splits Input Format \(p. 203\)](#)
- [Log files \(p. 203\)](#)
- [Thrift Service Ports \(p. 204\)](#)

- [Hive Authorization \(p. 204\)](#)
- [Hive File Merge Behavior with Amazon S3 \(p. 204\)](#)
- [Additional Features of Hive in Amazon EMR \(p. 205\)](#)

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/>.

Input Format

The Apache Hive default input format is text. The Amazon EMR default input format for Hive is `org.apache.hadoop.hive.ql.io.CombineHiveInputFormat`. You can specify the `hive.base.inputformat` option in Hive to select a different file format, for example:

```
hive>set hive.base.inputformat=org.apache.hadoop.hive.ql.io.HiveInputFormat;
```

To switch back to the default Amazon EMR input format, you would enter the following:

```
hive>set hive.base.inputformat=default;
```

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=1000000000;
```

Note

This input format is available only in clusters running Hive 0.8.1 or later.

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 you run determines the log file name, as shown in the following table.

Hive Version	Log File Name
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.

Hive Version	Log File Name
0.8.1	hive_081.log 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.
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 0.11.0	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 does not support [Hive Authorization](#). Amazon EMR clusters run with authorization disabled. You cannot use Hive authorization in your Amazon EMR cluster.

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 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.

Additional Features of Hive in Amazon EMR

We have extended Hive with new features that integrate Hive with Amazon Web Services (AWS), such as the ability to read and write from Amazon Simple Storage Service (Amazon S3). For information about which versions of Hive support these additional features, see [Hive Patches \(p. 211\)](#).

Topics

- [Write Data Directly to Amazon S3 \(p. 205\)](#)
- [Use Hive to Access Resources in Amazon S3 \(p. 205\)](#)
- [Use Hive to Recover Partitions \(p. 206\)](#)
- [Variables in Hive \(p. 206\)](#)
- [Make JDBC Connections in Hive \(p. 207\)](#)
- [Persist Hive Schema \(p. 208\)](#)
- [Amazon EMR Hive Steps \(p. 209\)](#)
- [Amazon EMR Hive queries to accommodate partial DynamoDB schemas \(p. 209\)](#)
- [Copy data between DynamoDB tables in different AWS regions \(p. 210\)](#)
- [Set DynamoDB throughput values per table \(p. 211\)](#)
- [Hive Patches \(p. 211\)](#)

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 tmp select .... ;
insert overwrite 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)
```

The ability to initialize Hive from a file stored in Amazon S3 was introduced with Hive 0.8.1. Versions of Hive prior to 0.8.1 do not support initializing Hive from Amazon S3. For example, in the following Hive command, `-i s3n://myawsbucket/hive_init.sql` succeeds if run with Hive 0.8.1 or later, and fails if run with an earlier version of Hive.

```
hive -i s3n://myawsbucket/hive_init.sql -f s3n://myawsbucket/hive_example.sql
```

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`.

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 the following example shows.

```
-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 steps with the CLI

- In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).
 - 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 pass variable values into steps with the 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"));
```

Make JDBC Connections in Hive

When you start an interactive Hive session using either the EMR console or the CLI, a Hive server starts on the master node and installs Hive in a cluster. The Hive server accepts JDBC connections from the Hive JDBC driver on port 10000.

To establish a connection from a remote machine

1. Start an SSH tunnel.

```
ssh -i my_private_key.pem hadoop@<master_node> -N -L 1234:localhost:10000
```

Replace <master_node> with the DNS name of the master node of your cluster. Alternatively, you can establish an SSH tunnel using Java secure channel (JSch).

2. Connect to the Hive server using the JDBC connection string.

```
jdbc:hive://localhost:1234/default
```

Alternatively, you can connect from a machine running in Amazon EC2 that is either in the ElasticMapReduce-master or ElasticMapReduce-slave security group.

Persist Hive Schema

By default, Hive keeps its schema information on the master node and that information ceases to exist when the cluster terminates. You can use the `hive-site.xml` feature to override the default location of the metadata store and replace it with a location that persists. In the following example, the default location is replaced by a MySQL instance that is already running in Amazon EC2.

To override the default location of the metadata store

1. Create a Hive site configuration file and store it in Amazon S3 so that it can override the location of the metadata store.

```
<configuration>
<property>
  <name>javax.jdo.option.ConnectionURL</name>
  <value>jdbc:mysql://ec2-72-44-33-189.compute-1.amazonaws.com:3306/hive?user=user12&password=abababa7&create=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>
</configuration>
```

In this example, the Hive site configuration file is in the following Amazon S3 location:

```
s3://mybucket/config/hive-site.conf
```

2. In the directory where you installed the Amazon EMR CLI, use `hive-site` to install the configuration file in a cluster by running the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --jobflow $JOBFLOW \
--hive-site=s3://mybucket/conf/hive-site.xml
```

- Windows users:

```
ruby elastic-mapreduce --jobflow $JOBFLOW --hive-site=s3://mybucket/conf/hive-site.xml
```

Amazon EMR Hive Steps

The Amazon EMR CLI provides a convenient way to access Hive steps. You can also access Hive steps from programs that call directly into the Amazon EMR web service.

To access Hive steps from the CLI

- Run the --describe command of the install step on a cluster that has Hive installed on it.

```
"StepConfig": {
    "ActionOnFailure": "TERMINATE_JOB_FLOW",
    "Name": "Setup Hive",
    "HadoopJarStep": {
        "MainClass": null,
        "Jar": "s3://elasticmapreduce/libs/script-runner/script-runner.jar",
        "Args": [
            "s3://elasticmapreduce/libs/hive/0.4/install-hive"
        ],
        "Properties": []
    }
}
```

In this example, you can see that a custom JAR called script-runner executes a script called install-hive, which resides in Amazon S3.

Notice that the install scripts are region-specific. If you're launching a cluster in eu-west-1 for example, you should include the installed script in the bucket eu-west-1.elasticmapreduce rather than the bucket us-east-1.elasticmapreduce.

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,
map<String, String> fullColumn)
    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 s3 SELECT item fullColumn FROM dynamodb WHERE recordTimeStamp
< "2012-01-01";

```

The following table shows the query syntax for selecting any combination of items from DynamoDB.

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. For more information, see [Hive Options \(p. 379\)](#).

```
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](#).

Note

These per table throughput properties will override the global Hive properties as mentioned in this section: [Hive Options \(p. 379\)](#).

```
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>
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>

Patch	Description
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>
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>

Patch	Description
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 Hive.optimize.s3.query=true 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

You can choose to run Hive in several different configurations. You set the `--hive-versions`, and `--ami-version` options in the job creation call as shown in the following table.

The default configuration for Amazon EMR is the latest version of Hive running on the latest AMI version.

The Amazon EMR console does not support Hive versioning and always loads the latest version of Hive.

Versions of the Amazon EMR CLI released on 9 April 2012 and later load the latest version of Hive by default. To use a version of Hive other than the latest, specify the `--hive-versions` option when you create the cluster. Versions of the Amazon EMR CLI released prior to 9 April 2012 load the default configuration of Hive.

Calls to the API launch the default configuration of Hive, unless you specify the `--hive-versions` option for the step that loads Hive onto the cluster during the call to [RunJobFlow](#).

Hive Version	Compatible Hadoop Versions	Hive Version Notes
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 <code>TBLPROPERTIES</code>. (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. 210).• 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. 209).• 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. 211).• 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.

Hive Version	Compatible Hadoop Versions	Hive Version Notes
0.8.1.7	1.0.3	<ul style="list-style-type: none"> Fixes ColumnPruner 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 ConcurrentModificationException on a lateral view 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.smallfiles.avgsize</code> setting is ignored. For more information, see Hive File Merge Behavior with Amazon S3 (p. 204) 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. 125)

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 dynamodb.retry.duration 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 dynamodb.endpoint 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. 379) .
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 spare 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. 364) . 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.

For more information about the changes in a version of Hive, see [Supported Hive Versions \(p. 213\)](#). For information about Hive patches and functionality developed by the Amazon EMR team, see [Additional Features of Hive in Amazon EMR \(p. 205\)](#).

To specify the Hive version when creating the cluster

- Use the `--hive-versions` option. The following command-line example creates an interactive Hive cluster running Hadoop 0.20 and Hive 0.7.1.

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- 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 0.7.1
```

- Windows users:

```
ruby elastic-mapreduce --create --alive --name "Test Hive" --num-instances
5 --instance-type m1.large --hive-interactive --hive-versions 0.7.1
```

The `--hive-versions` option must come after any reference to the options `--hive-interactive`, `--hive-script`, or `--hive-site`.

To specify the latest Hive version when creating the cluster

- Use the `--hive-versions` option with the `latest` keyword. The following command-line example creates an interactive Hive cluster running the latest version of Hive.
- 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 command-line example illustrates setting both the interactive and the script version of Hive to use 0.7.1.
- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --debug --log-uri s3://myawsbucket/perft
est/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://myawsbucket/perftest/hive-script.hql --hive-versions
0.7.1.2
```

- Windows users:

```
ruby elastic-mapreduce --create --debug --log-uri s3://myawsbucket/perftest/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://myawsbucket/perftest/hive-script.hql --hive-versions 0.7.1.2
```

To load multiple versions of Hive for a given cluster

- Use the `--hive-versions` option and separate the version numbers by comma. The following command-line example creates an interactive cluster running Hadoop 0.20 and multiple versions of Hive. With this configuration, you can use any of the installed versions of Hive on the cluster.
 - 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` will access 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` will access 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.

Display the Hive Version

You can use the `--print-hive-version` command to display the version of the Hive currently in use for a given cluster. 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. The syntax for this is as follows, where `JobFlowID` is the identifier of the cluster to check the Hive version on.

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --jobflow JobFlowID --print-hive-version
```

- Windows users:

```
ruby elastic-mapreduce --jobflow JobFlowID --print-hive-version
```

Upgrade to Hive 0.11.0

Hive 0.11.0 is an important upgrade that provides significant performance improvements, new features, and bug fixes. Notable updates in this version are:

- Support for the [ORC file format](#), a more efficient and high-performance way to store data in Hive.
- Better Hive SQL flexibility with new [windowing and analytics functions](#) such as LEAD/LAG/FIRST/LAST and improvements to PARTITION BY and ORDER BY.
- Better SQL compatibility with the introduction of decimal data type and TRUNCATE command.
- JOIN optimizations are enhanced and the use of hints in JOINs is de-emphasized. ([HIVE-3784](#))
- Improvements to query plan optimization rules by removing redundant operators. ([HIVE-948](#))

For a detailed list of new features, see [Supported Hive Versions \(p. 213\)](#).

To use these improvements, you can either launch a new cluster on an AMI that supports Hive 0.11.0 or upgrade an older Hive cluster to Hive 0.11.0.

Note

Hadoop 1.0.3 (AMI 2.2.0 or later) does not support Hive 0.7 or Hive 0.5 and you must use Hive 0.8 or later.

Topics

- [Upgrade the Configuration Files \(p. 222\)](#)
- [Upgrade the Metastore \(p. 222\)](#)

Upgrade the Configuration Files

Apache deprecated the `hive-default.xml` file for Hive 0.8 and newer. However, Amazon EMR still uses `hive-default.xml` to maintain its default configuration settings. You can add a new configuration by appending it to either `hive-default.xml` or `hive-site.xml`. In addition, you can override an existing configuration in `hive-default.xml` either by changing its value in `hive-default.xml` or adding it to `hive-site.xml`.

Upgrade the Metastore

The Hive metastore stores the metadata for Hive tables and partitions. Amazon EMR uses a MySQL database to contain the metastore.

By default, Amazon EMR creates the MySQL database on the master node. In this scenario, the metastore is deleted when the cluster terminates. For the metastore to persist between clusters, you can specify that the Hive cluster use a remote metastore, such as a MySQL database hosted on Amazon RDS. For

more information about how to create a remote metastore, see [Create a Metastore Outside the Hadoop Cluster \(p. 241\)](#).

If you create a new cluster using Hive 0.11 and let it create a new metastore on the master node (the default behavior), it will have the new schema. No updates are required.

If you have an existing metastore, created with Hive 0.8 or earlier, that you want to reuse, you must update the schema to the Hive 0.11 format. Apache Hive provides scripts you can use to update metastore schemas from one version to another; their use is explained in the following procedures.

If you are updating a metastore created with a version of Hive prior to 0.8, this may require multiple steps. For example, a metastore created with Hive 0.6 would first need to be updated to the Hive 0.7 schema, then the Hive 0.8 schema, before it could be updated to the Hive 0.11 schema. There were no schema changes from version 0.8 to 0.9 and 0.10 to 0.11. So, if you are upgrading from 0.8 to 0.11, running the upgrade script from 0.9 to 0.10 is sufficient. For a list of the Apache update scripts for previous versions of Hive, go to <http://svn.apache.org/viewvc/hive/branches/branch-0.8/metastore/scripts/upgrade/mysql/>.

Note

The transformation scripts only work in one direction. After you've converted your Hive metastore to the Hive 0.11 format, you cannot use the scripts to convert the metastore back to the Hive 0.8 format. It is recommended that you back up your metastore before you begin the upgrade process.

Upgrade to Hive 0.11 (MySQL on the Master Node)

The following procedures explain how to upgrade a Hive metastore stored in a MySQL database hosted on the master node of a cluster (the default behavior).

If your metastore is stored remotely, as described in [Create a Metastore Outside the Hadoop Cluster \(p. 241\)](#), please use the instructions at [Upgrade to Hive 0.11 \(MySQL on Amazon RDS\) \(p. 227\)](#) instead.

To upgrade the metastore from 0.8 to 0.11 (MySQL on the master node)

1. Stop running Hive processes during the upgrade procedure. This ensures that the database is not altered while the upgrade scripts are running.
2. Use SSH to connect to the master node of the Hive cluster to upgrade. For more information about how to use SSH to connect to the master node, see [Connect to the Master Node Using SSH \(p. 446\)](#).
3. Copy the upgrade scripts from Apache to the master node. You can do this by running the wget utility on the master node. (The view=co request variable in the following URLs ensures that you get the plain-text version of the scripts, not the HTML-encoded version.)

```
wget -O 010-HIVE-3072.mysql.sql ht
tp://svn.apache.org/viewvc/hive/branches/branch-0.11/metastore/scripts/upgrade/mysql/010-HIVE-3072.mysql.sql?view=co
wget -O 011-HIVE-3649.mysql.sql ht
tp://svn.apache.org/viewvc/hive/branches/branch-0.11/metastore/scripts/upgrade/mysql/011-HIVE-3649.mysql.sql?view=co
wget -O 012-HIVE-1362.mysql.sql ht
tp://svn.apache.org/viewvc/hive/branches/branch-0.11/metastore/scripts/upgrade/mysql/012-HIVE-1362.mysql.sql?view=co
wget -O upgrade-0.9.0-to-0.10.0.mysql.sql ht
tp://svn.apache.org/viewvc/hive/branches/branch-0.11/metastore/scripts/upgrade/mysql/upgrade-0.9.0-to-0.10.0.mysql.sql?view=co
```

4. Launch the MySQL Monitor using the following command.

```
mysql -u root
```

5. From the MySQL Monitor command prompt, find the name of the database that contains your Hive metastore. You can do this by running the following command.

```
mysql> show databases;
```

This returns results such as the following. The database that starts with "hive_" is the one that contains your Hive MetaStore. In the example below, this is "hive_081".

```
+-----+  
| Database |  
+-----+  
| information_schema |  
| InstanceController |  
| hive_081 |  
| mysql |  
+-----+  
4 rows in set (0.00 sec)
```

6. Exit MySQL Monitor.

```
mysql> exit
```

7. Back up your MySQL metastore database. You can do this using the mysqldump utility, as shown in the example below, where hive_081 is the name of the database containing the metastore.

```
mysqldump --opt hive_081 > metastore_backup.sql -u root
```

8. Export the current metastore schema to a file. The following mysqldump command extracts only the schema content.

```
mysqldump --skip-add-drop-table --no-data \  
hive_081 > my-schema-0.8.1.mysql.sql -u root
```

9. Compare your current schema against the official Apache Hive schema listed at <http://svn.apache.org/viewvc/hive/branches/branch-0.11/metastore/scripts/upgrade/mysql/>. If you are upgrading from 0.8 to 0.11, compare the schema you exported against the version in `hive-schema-0.8.0.mysql.sql`. The schemas should match. If you have made custom changes to the schema, you may need to roll those back in order for the upgrade scripts to work properly.

Differences you may find:

- **Missing tables.** By default, Hive only creates schema elements when they are used. If you have not created a certain type of Hive catalog object, the corresponding table will not exist in your schema. You must create these missing tables for the upgrade script to succeed. You can do this by hand or by running the official schema DDL script (located at <http://svn.apache.org/viewvc/hive/branches/branch-0.11/metastore/scripts/upgrade/mysql/hive-schema-0.8.0.mysql.sql?view=co>) against your metastore. This script ignores tables that already exist, and will only create those that are missing.
- **Extra tables.** If your schema contains tables named NUCLEUS_TABLES or SEQUENCE_TABLE, these will not affect the upgrade script. You do not need to remove them.
- **Reversed Column Constraint Names.** If a table has multiple constraints, the names may be reversed between your schema and the canonical schema. For example, if a table contains PARTITIONS_FK1 and PARTITIONS_FK2 which reference SDS.SD_ID and TBLS.TBL_ID, your schema may instead connect PARTITIONS_FK1 to TBLS.TBL_ID and PARTITIONS_FK2 to SDS.SD_ID. This will not affect the upgrade script and can be ignored.
- **Changes in Column and Constraint Names.** If your schema contains tables with unique keys named "UNIQUE<tab_name>" or columns named "IDX" you will need to rename these to "UNIQUE_<tab_name>" and "INTEGER_IDX" before you run the upgrade script. The reason for this is explained in HIVE-1435.

10. Launch the MySQL Monitor using the following command.

```
mysql -u root
```

11. Run upgrade-0.9.0-to-0.10.0.mysql.sql script using the source command in the MySQL command line. This is shown in the following example, where hive_081 is the database containing the metastore.

```
mysql> use hive_081;
mysql> source upgrade-0.9.0-to-0.10.0.mysql.sql;
```

The script should complete without error.

12. Exit MySQL Monitor.

```
mysql> exit
```

13. Export the upgraded metastore schema to a file. The following mysqldump command extracts only the schema content.

```
sudo mysqldump --skip-add-drop-table --no-data \
hive_081 > my-schema-0.11.0.mysql.sql
```

14. Compare this schema to the official Apache Hive 0.11 schema: <http://svn.apache.org/viewvc/hive/branches/branch-0.11/metastore/scripts/upgrade/mysql/hive-schema-0.8.0.mysql.sql>. They should match.
15. Back up the upgraded metastore.

```
mysqldump --opt hive_081 > metastore_upgraded_backup.sql \
-u root
```

To move your upgraded configuration files and metastore to a Hive 0.11 cluster (MySQL on the master node)

1. Create a new cluster on Hadoop 1.0.3 (AMI version 2.2 or later).
2. Use SSH to connect to the master node. For more information about how to do this, see [Connect to the Master Node Using SSH \(p. 446\)](#).
3. Use the scp utility to copy your upgraded configuration files and upgraded metastore backup file to the new cluster.
4. Copy custom configuration settings from `hive-default.xml` and `hive-site.xml` to the `hive-site.xml` on the new cluster (found on the master node at `hive/conf/`.) Take care not to overwrite any settings used by Amazon EMR.
5. Import the upgraded metastore to the MySQL database (whether locally on the master, or on a remote server.)

```
mysql -u root
mysql>show databases;
```

This returns results such as the following. The database that starts with "hive_" is the one that contains your Hive MetaStore. In the example below, this is "hive_0110".

```
+-----+
| Database      |
+-----+
| information_schema |
| InstanceController |
| hive_0110        |
| mysql           |
+-----+
4 rows in set (0.00 sec)
```

6. Replace this empty metastore, `hive_0110` in the preceding example, with your upgraded metastore. You can use the following commands to move your database to the new location.

```
mysql> drop database hive_0110;
mysql> create database hive_0110;
```

```
mysql> use database hive_0110;
mysql> source metastore_upgraded_backup.sql;
```

Upgrade to Hive 0.11 (MySQL on Amazon RDS)

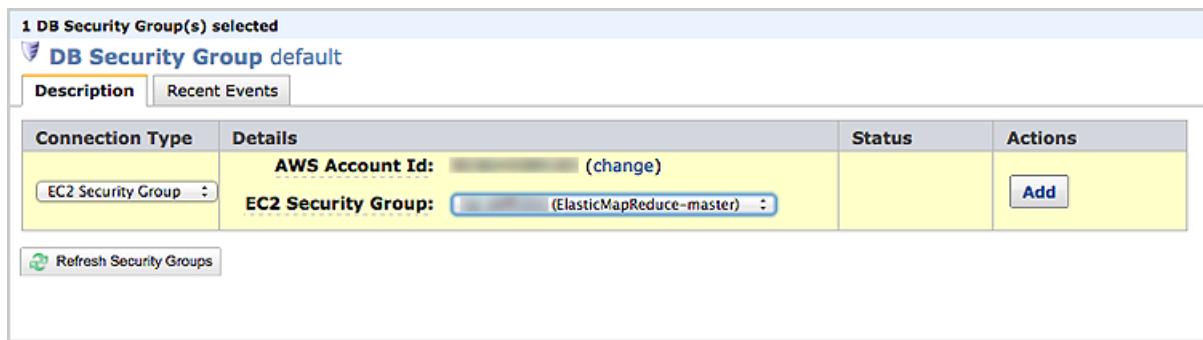
The following procedures explain how to upgrade a Hive metastore stored outside of the cluster, as described in [Create a Metastore Outside the Hadoop Cluster \(p. 241\)](#).

If your metastore is stored on the master node (the default behavior) please use the procedures in [Upgrade to Hive 0.11 \(MySQL on the Master Node\) \(p. 223\)](#) instead.

In order to connect to the Amazon RDS database from the master node, you need to add the EC2 Security Group elasticmapreduce-master to the DB Security Groups. You can do this as described in the following procedure.

To configure security groups to enable connections to Amazon RDS from the master node

1. From the Amazon RDS Console, click **DB Security Groups** in the left pane.
2. Select the security group to modify in the center pane. This should be the security group that was used to launch the MySQL database hosting the Hive metastore. The default security group is **default**.
3. In the information pane at the bottom, select **EC2 Security Group** for **Connection Type** and select **ElasticMapReduce-master** for **EC2 Security Group**.



4. Click **Add**.

To upgrade the metastore from 0.8 to 0.11 (MySQL on Amazon RDS)

1. Stop running Hive processes during the upgrade procedure. This ensures that the database is not altered while the upgrade scripts are running.
2. Back up the Amazon RDS database as described in [Creating a DB Snapshot](#) in the *Amazon Relational Database Service User Guide*.
3. Use SSH to connect to the master node of the Hive cluster to upgrade. For more information, see [Connect to the Master Node Using SSH \(p. 446\)](#).
4. Copy the upgrade scripts from Apache to the master node. You can do this by running the wget utility on the master node. (The view=co request variable in the following URLs ensures that you get the plain-text version of the scripts, not the HTML-encoded version.)

```
wget -O 010-HIVE-3072.mysql.sql ht
tp://svn.apache.org/viewvc/hive/branches/branch-0.11/metastore/scripts/up
```

```
grade/mysql/010-HIVE-3072.mysql.sql?view=co
wget -O 011-HIVE-3649.mysql.sql ht
tp://svn.apache.org/viewvc/hive/branches/branch-0.11/metastore/scripts/upgrade/mysql/011-HIVE-3649.mysql.sql?view=co
wget -O 012-HIVE-1362.mysql.sql ht
tp://svn.apache.org/viewvc/hive/branches/branch-0.11/metastore/scripts/upgrade/mysql/012-HIVE-1362.mysql.sql?view=co
wget -O upgrade-0.9.0-to-0.10.0.mysql.sql ht
tp://svn.apache.org/viewvc/hive/branches/branch-0.11/metastore/scripts/upgrade/mysql/upgrade-0.9.0-to-0.10.0.mysql.sql?view=co
```

5. Use the MySQL monitor installed on the master node to connect to the Amazon RDS database. Before you can do this, you must have completed the steps in the preceding procedure, **To configure security groups to enable connections to Amazon RDS from the master node**.

From the master node, run the following command to connect to the Amazon RDS database, where `myinstance` is the name of the database, `mydnsnameexample` is the custom portion of the DNS name assigned to the database and `mymasteruser` is the master user on the database. For more information about how to connect to an Amazon RDS instance, go to [Connecting to a DB Instance](#) [Running the MySQL Database Engine in the Amazon Relational Database Service User Guide](#).

```
mysql -h myinstance.mydnsnameexample.rds.amazonaws.com -P 3306 -u mymasteruser
-p
```

6. From the MySQL Monitor command prompt, find the name of the database that contains your Hive metastore. You can do this by running the following command.

```
mysql> show databases;
```

This returns results such as the following. The database that starts with "hive_" is the one that contains your Hive MetaStore.

```
+-----+
| Database      |
+-----+
| information_schema |
| InstanceController |
| hive_081          |
| mysql            |
+-----+
4 rows in set (0.00 sec)
```

7. Exit MySQL Monitor.

```
mysql> exit
```

8. Export the current metastore schema from Amazon RDS. The following mysqldump command extracts only the schema content.

Note

If your Amazon RDS instance runs mysql 5.6, you will not be able to execute following mysqldump command. This is because the EMR master node's mysqldump version is 5.1 and running following command results in version mismatch errors. A workaround is to run the dump command on a machine that has compatible mysqldump utility installed.

```
mysqldump --skip-add-drop-table --no-data hive_081 > my-schema-0.8.1.mysql.sql \
 \
-u root -h myinstance.mydnsnameexample.rds.amazonaws.com \
-p 3306 -u mymasteruser -p
```

9. Compare your current schema against the official Apache Hive schema listed at <http://svn.apache.org/viewvc/hive/branches/branch-0.11/metastore/scripts/upgrade/mysql/>. If you are upgrading from 0.8 to 0.11, compare the schema you exported against the version in hive-schema-0.8.0.mysql.sql. The schemas should match. If you have made custom changes to the schema, you may need to roll those back in order for the upgrade scripts to work properly.

Differences you may find:

- **Missing tables.** By default, Hive only creates schema elements when they are used. If you have not created a certain type of Hive catalog object, the corresponding table will not exist in your schema. You must create these missing tables for the upgrade script to succeed. You can do this by hand or by running the official schema DDL script (located at <http://svn.apache.org/viewvc/hive/branches/branch-0.11/metastore/scripts/upgrade/mysql/hive-schema-0.8.0.mysql.sql?view=co>) against your metastore. This script ignores tables that already exist, and will only create those that are missing.
- **Extra tables.** If your schema contains tables named NUCLEUS_TABLES or SEQUENCE_TABLE, these will not affect the upgrade script. You do not need to remove them.
- **Reversed Column Constraint Names.** If a table has multiple constraints, the names may be reversed between your schema and the canonical schema. For example, if a table contains PARTITIONS_FK1 and PARTITIONS_FK2 which reference SDS.SD_ID and TBLS.TBL_ID, your schema may instead connect PARTITIONS_FK1 to TBLS.TBL_ID and PARTITIONS_FK2 to SDS.SD_ID. This will not affect the upgrade script and can be ignored.
- **Changes in Column and Constraint Names.** If your schema contains tables with unique keys named "UNIQUE<tab_name>" or columns named "IDX" you will need to rename these to "UNIQUE_<tab_name>" and "INTEGER_IDX" before you run the upgrade script. The reason for this is explained in HIVE-1435.

10. Use the MySQL monitor installed on the master node to connect to the Amazon RDS database.

```
mysql -h myinstance.mydnsnameexample.rds.amazonaws.com \
-p 3306 -u mymasteruser -p
```

11. Run the upgrade-0.9.0-to-0.10.0.mysql.sql script using the source command in the MySQL command line. This is shown in the following example, where hive_081 is the database containing the metastore.

```
mysql> use hive_081;
mysql> source upgrade-0.9.0-to-0.10.0.mysql.sql;
```

The script should complete without error.

12. Exit MySQL Monitor.

```
mysql> exit
```

13. Export the upgraded metastore schema from Amazon RDS. The following mysqldump command extracts only the schema content.

Note

The following command may generate a version mismatch error if your mysql version is incorrect. In that case, refer to the workaround mentioned in the previous steps.

```
mysqldump --skip-add-drop-table --no-data hive_081 > metastore_up
graded_backup.sql \
-u root -h myinstance.mydnsnameexample.rds.amazonaws.com \
-P 3306 -u mymasteruser -p
```

14. Compare this schema to the official Apache Hive 0.11 schema: <http://svn.apache.org/viewvc/hive/branches/branch-0.11/metastore/scripts/upgrade/mysql/hive-schema-0.10.0.mysql.sql>. They should match.
15. Backup the upgraded metastore by backing up the Amazon RDS database as described in [Creating a DB Snapshot](#) in the *Amazon Relational Database Service User Guide*.

To move your upgraded configuration files and metastore to a Hive 0.11 cluster (MySQL on Amazon RDS)

1. Create a new cluster on Hadoop 1.0.3 (AMI version 2.2 or later). Follow the instructions at [Create a Metastore Outside the Hadoop Cluster \(p. 241\)](#) to have the Hive cluster use the metastore you upgraded to Hive 0.11 in the previous procedure.
2. Use SSH to connect to the master node. For more information about how to do this, see [Connect to the Master Node Using SSH \(p. 446\)](#).
3. Use scp to copy your upgraded configuration files to the new cluster.
4. Copy custom configuration settings from `hive-default.xml` and `hive-site.xml` to the `hive-site.xml` on the new cluster (found on the master node at `hive/conf/`). Take care not to overwrite any settings used by Amazon EMR.

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.

Interactive and Batch Hive Clusters

Amazon EMR enables you to run Hive scripts in two modes:

- Interactive
- Batch

Typically, you use interactive mode to troubleshoot your cluster and use batch mode in production.

In interactive mode, you `ssh` as the Hadoop user into the master node in the Hadoop cluster and use the Hive Command Line Interface to develop and run your Hive script. Interactive mode enables you to revise the Hive script more easily than batch mode. After you successfully revise the Hive script in interactive mode, you can upload the script to Amazon S3 and use batch mode to run production clusters.

In batch mode, you upload your Hive script to Amazon S3, and then execute it using a job flow. You can pass parameter values into your Hive script and reference resources in Amazon S3. Variables in Hive scripts use the dollar sign and curly braces, for example:

`${VariableName}`

In the Amazon EMR CLI, use the `-d` parameter to pass values into the Hive script as in the following example.

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create \
--name "Hive Cluster" \
--hive-script \
--args s3://myawsbucket/myquery.q \
--args -d,INPUT=s3://myawsbucket/input,-d,OUTPUT=s3://myawsbucket/output
```

- Windows users:

```
ruby elastic-mapreduce --create --name "Hive Cluster" --hive-script --args
s3://myawsbucket/myquery.q --args -d,INPUT=s3://myawsbucket/input,-d,OUT
PUT=s3://myawsbucket/output
```

Using batch mode, you can pass parameter values into a Hive script from the **Specify Parameters** page of the **Create a New Job Flow** wizard found in the Amazon EMR console. The values go into the **Extra Args** field. For example, you could enter:

```
-d VariableName=Value
```

The Amazon EMR console and Amazon EMR command line interface (CLI) support both interactive and batch modes.

Running Hive in Interactive Mode

You can run Hive in interactive mode from both the CLI and Amazon EMR console.

- To start an interactive cluster from the command line, use the `--alive` option with the `--create` parameter so that the cluster remains active until you terminate it.

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- 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
```

The return output is similar to the following:

```
Created jobflow JobFlowID
```

Add additional steps from the Amazon Elastic MapReduce (Amazon EMR) CLI or `ssh` directly to the master node following the instructions in the [Amazon Elastic MapReduce \(Amazon EMR\) Getting Started Guide](#).

You start an interactive cluster from the Amazon EMR console in the **Create a New Job Flow** wizard.

The cluster begins. When the cluster is in the `WAITING` state, you can add steps to your cluster from the Amazon EMR CLI or `ssh` directly to the master node following the instructions in the [Amazon Elastic MapReduce \(Amazon EMR\) Getting Started Guide](#).

Adding steps can help you test and develop Hive scripts. For example, if the script fails, you can add a new step to the cluster without having to wait for a new cluster to start. The following procedure shows you how to use the command line to add Hive as a new step to an existing cluster.

To add Hive to an existing cluster

- Enter the following command, replacing the location with an Amazon S3 bucket containing a Hive script and the `<JobFlowID>` from your job:

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --jobflow JobFlowID \
--hive-script \
--args s3://location/myquery.q \
--args -d,INPUT=s3://location/input,-d,OUTPUT=s3://location/output
```

- Windows users:

```
ruby elastic-mapreduce --jobflow JobFlowID --hive-script --args s3://location/myquery.q --args -d,INPUT=s3://location/input,-d,OUTPUT=s3://location/output
```

Running Hive in Batch Mode

The following procedure shows how to run Hive in batch mode from the command line. The procedure assumes that you stored the Hive script in a bucket on Amazon S3. For more information about uploading files into Amazon S3, go to the [Amazon S3 Getting Started Guide](#).

To create a cluster with a step that executes a Hive script

- Enter the following command, substituting the replaceable parameters with the actual values from your job:

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create \
--name "Hive cluster" \
--hive-script \
--args s3://myawsbucket/myquery.q \
--args -d,INPUT=s3://myawsbucket/input,-d,OUTPUT=s3://myawsbucket/output
```

- Windows users:

```
ruby elastic-mapreduce --create --name "Hive cluster" --hive-script --args
s3://myawsbucket/myquery.q --args -d,INPUT=s3://myawsbucket/input,-d,OUT
PUT=s3://myawsbucket/output
```

The `--args` option provides arguments to the Hive-script. The first `--args` option here specifies the location of the Hive script in Amazon S3. In the second `--args` option, the `-d` provides a way to pass values (`INPUT`, `OUTPUT`) into the script. Within the Hive script, these parameters are available as `${variable}`. In this example, Hive replaces `${INPUT}` and `${OUTPUT}` with the values you passed in. These variables are substituted during a preprocessing step, so the variables can occur anywhere in the Hive script.

The return output is similar to the following:

```
Created jobflow JobFlowID
```

Launch a Hive Cluster

This section covers the basics of creating a cluster using Hive in Amazon Elastic MapReduce (Amazon EMR). You'll step through how to create a cluster using Hive with either the Amazon EMR console, the CLI, or the Query API. Before you create your cluster you'll need to create objects and permissions; for more information see [Prepare Input Data \(Optional\) \(p. 98\)](#).

For advanced information on Hive configuration options, see [Analyze Data with Hive \(p. 202\)](#).

A cluster using Hive enables you to create a data analysis application using a SQL-like language. The example that follows is based on the Amazon EMR sample: [Contextual Advertising using Apache Hive and Amazon EMR with High Performance Computing instances](#). This sample describes how to correlate customer click data to specific advertisements.

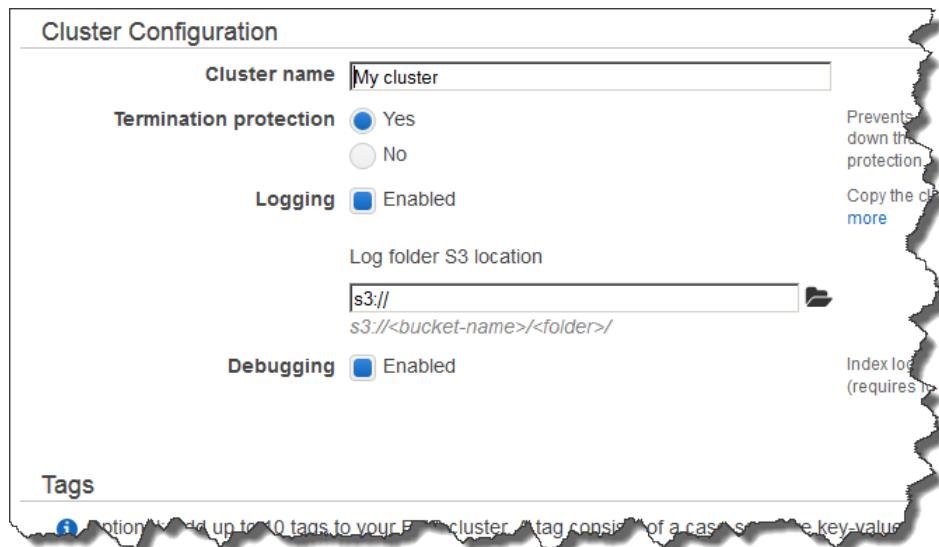
In this example, the Hive script is located in an Amazon S3 bucket at `s3n://elasticmapreduce/samples/hive-ads/libs/model-build`. All of the data processing instructions are located in the Hive script. The script requires additional libraries that are located in an Amazon S3 bucket at `s3n://elasticmapreduce/samples/hive-ads/libs`. The input data is located in the Amazon S3 bucket `s3n://elasticmapreduce/samples/hive-ads/tables`. The output is saved to an Amazon S3 bucket you created as part of [Prepare an Output Location \(Optional\) \(p. 112\)](#).

Amazon EMR Console

This example describes how to use the Amazon EMR console to create a cluster using Hive.

To create a cluster using Hive

1. Open the Amazon Elastic MapReduce 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>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 (p. 459). 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>This determines whether Amazon EMR captures detailed log data to Amazon S3.</p> <p>For more information, see View Log Files (p. 418).</p>
Log folder S3 location	<p>Enter an Amazon S3 path to store your debug logs if you enabled logging in the previous field. If the log folder does not exist, the Amazon EMR console creates it for you.</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 (p. 418).</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.

Software Configuration

Hadoop distribution Amazon Use Amazon's Hadoop distribution. [Learn more](#)

AMI version
2.4.2 (hadoop 1.0.3) - latest Determines the base configuration of the instances in your cluster, including the Hadoop version. [Learn more](#)

MapR Use MapR's Hadoop distribution. [Learn more](#)

Applications to be installed	Version	
Hive	0.11.0.1	  
Pig	0.11.1.1	  
Additional applications	Select an application <input type="button" value="▼"/>	<input type="button" value="Configure and add"/>

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. 149).</p>
AMI version	<p>Choose 2.4.2 (Hadoop 1.0.3).</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 (p. 51).</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, 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](#).

Hardware Configuration

Specify the networking and hardware configuration for your cluster. If you need more than 20 EC2 Request Spot instances (unused EC2 capacity) to save money.

Network: vpc-c1a3b3a3 (172.31.0.0/16) (default) Use a Virtual Private Cloud to connect to your data or connect to the Internet.

EC2 Subnet: No preference (random subnet) Create a Subnet

EC2 instance type	Count	Request spot		
Master	m1.small	1	<input type="checkbox"/>	The Master instance is the primary node in the cluster, and is responsible for assigning tasks to core and task nodes, and monitoring their status. There is always one master node in each cluster.
Core	m1.small	2	<input type="checkbox"/>	Core instances are the worker nodes in the Hadoop Distributed File System (HDFS).
Task	m1.small	0	<input type="checkbox"/>	Task instances are the worker nodes in the MapReduce framework.

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 a Amazon VPC Subnet for the Cluster (Optional) (p. 137).</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 Elastic Compute Cloud User Guide</i>.</p>
Master	<p>Choose m1.small.</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.small (default), m1.large, m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cc1.4xlarge, cg1.4xlarge.</p> <p> This tutorial uses small instances for all nodes due to the light workload and to keep your costs low.</p> <p> For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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 Lower Costs with Spot Instances (Optional) (p. 37).</p>

Field	Action
Core	<p>Choose m1.small.</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.small (default), m1.large, m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cc1.4xlarge, cg1.4xlarge.</p> <p>This tutorial uses small instances for all nodes due to the light workload and to keep your costs low.</p> <p>For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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) (p. 37).</p>
Task	<p>Choose m1.small.</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. Valid types: m1.small (default), m1.large, m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cc1.4xlarge, cg1.4xlarge.</p> <p>For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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) (p. 37).</p>

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

Security and Access

EC2 key pair	<input type="button" value="Proceed without an EC2 key pair"/>	Use an existing key pair to SSH into the master node of the Amazon EC2 cluster as the user "hadoop". Learn more
IAM user access		<input type="radio"/> All other IAM users <input checked="" type="radio"/> No other IAM users
Control the visibility of this cluster to other IAM users. Learn more		
IAM Roles		
<p>EMR role <input type="button" value="No roles found"/> Allows EMR to access other AWS Services such as EC2 on your behalf. Learn more</p> <p>Create Default Role</p> <p>EC2 instance profile <input type="button" value="Proceed without role"/> Allows EC2 instances in an EMR cluster to access other AWS services such as S3. Learn more</p> <p>Create Default Role</p>		

Field	Action
EC2 key pair	<p>Choose an Amazon EC2 key pair from the list.</p> <p>For more information, see Create an Amazon EC2 Key Pair and PEM File (p. 117).</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 Cluster (p. 446).</p>
IAM user access	<p>Choose No other IAM users.</p> <p> 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 (p. 119).</p>
EC2 instance profile	<p>You can proceed without choosing an instance profile. If you create a cluster without selecting a specific instance profile, one will be created for you.</p> <p>This controls application access to the Amazon EC2 instances in the cluster.</p> <p>For more information, see Configure IAM Roles for Amazon EMR (p. 125).</p>
EMR role	<p>Choose Proceed without role.</p> <p>Allows Amazon EMR to access other AWS services on your behalf.</p> <p>For more information, see Configure IAM Roles for Amazon EMR (p. 125).</p>

7. 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 [Create Bootstrap Actions to Install Additional Software \(Optional\) \(p. 87\)](#).

8. In the **Steps** section, choose **Hive Program** from the list and click **Configure and add**.

In the **Add Step** dialog, specify the cluster parameters using the following table as a guide, and then click **Add**.

Add Step

Step Type	Hive Program
Name	Hive Program
Script S3 Location*	s3://elasticmapreduce/samples/hive-ads/libs/model-build.q
Input S3 Location	s3://elasticmapreduce/samples/hive-ads/table
Output S3 Location	s3://myawsbucket/hive-ads/output/2013-09-26
Arguments	<pre>-d LIBS=s3n://elasticmapreduce/samples/hive-ads/libs</pre>
Action on Failure	<input type="button" value="Continue"/>

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> .
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> .
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> .
Arguments	Optionally, enter a list of arguments (space-separated strings) to pass to Hive.
Action on Failure	This determines what the cluster does in response to any errors. The possible values for this setting are: <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.

* Required parameter

9. Review your configuration and if you are satisfied with the settings, click **Create Cluster**.
10. When the cluster starts, the console displays the **Cluster Details** page.

CLI

This example describes how to use the CLI to create a cluster using Hive.

To create a cluster using Hive

- In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).
 - Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --name "Test Hive" --hive-script \
s3n://elasticmapreduce/samples/hive-ads/libs/model-build.q \
--args -d,LIBS=s3n://elasticmapreduce/samples/hive-ads/libs, \
-d,INPUT=s3n://elasticmapreduce/samples/hive-ads/tables, \
-d,OUTPUT=s3n://myawsbucket/hive-ads/output/
```

- Windows users:

```
ruby elastic-mapreduce --create --name "Test Hive" --hive-script
s3n://elasticmapreduce/samples/hive-ads/libs/model-build.q --args -
-d,LIBS=s3n://elasticmapreduce/samples/hive-ads/libs,-d,INPUT=s3n://elast
icmapreduce/samples/hive-ads/tables,-d,OUTPUT=s3n://myawsbucket/hive-
ads/output/
```

The output looks similar to the following.

```
Created cluster JobFlowID
```

By default, this command launches a cluster to run on a two-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.

Create a Metastore Outside the Hadoop 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/#31>.
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?createDatabaseIfNotExist=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://myawsbucket/conf/hive-site.xml`.
4. Create a cluster and specify the Amazon S3 location of the new Hive configuration file, for example:

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --alive \
--name "Hive cluster" \
--hive-interactive \
--hive-site=s3://myawsbucket/conf/hive-site.xml
```

- Windows users:

```
ruby elastic-mapreduce --create --alive --name "Hive cluster" --hive-interactive \
--hive-site=s3://myawsbucket/conf/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`.

5. Connect to the master node of your cluster.
Instructions on how to connect to the master node are available in the [Amazon Elastic MapReduce \(Amazon EMR\) Getting Started Guide](#).
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://myawsbucket/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

The Hive JDBC driver provides a mechanism to move data from one database format to another. Installing a JDBC client requires you to download the JDBC driver and install the client software correctly. You can use the Hive JDBC driver to connect to a SQL client. An example of connecting to the SQuirrel SQL client follows.

To download JDBC drivers

- Download Hive 0.11.0 JDBC drivers from <http://aws.amazon.com/developertools/1982901737448217> and save the files locally.
- Download Hive 0.8.1 JDBC drivers from <http://aws.amazon.com/developertools/4897392426085727> and save the files locally.
- Download Hive 0.7.1 JDBC drivers from <http://aws.amazon.com/developertools/Elastic-MapReduce/8084613472207189> and save the files locally.
- Download Hive 0.7 JDBC drivers from <http://aws.amazon.com/developertools/Elastic-MapReduce/1818074809286277> and save the files locally.
- Download Hive 0.5 JDBC drivers from <http://aws.amazon.com/developertools/Elastic-MapReduce/0196055244487017> and save the files locally.

You need only download the drivers appropriate to the version(s) of Hive you want to access.

To install SQuirrel SQL client

- Download SQuirrel SQL client from <http://squirrel-sql.sourceforge.net/>.
- Open the self extracting JAR file, and follow the wizard instructions to install the software.
- From the command line, create an SSH tunnel to the master node of your Hive job flow as follows:

If you are installing...	Enter the following...
Hive 0.11.0 drivers	ssh -o ServerAliveInterval=10 -L 10004:localhost:10004 hadoop@ <i>MasterNodeDNS</i> -i \$HOME/ <i>mysecretkey.pem</i>
Hive 0.8.1 drivers	ssh -o ServerAliveInterval=10 -L 10003:localhost:10003 hadoop@ <i>MasterNodeDNS</i> -i \$HOME/ <i>mysecretkey.pem</i>
Hive 0.7.1 drivers	ssh -o ServerAliveInterval=10 -L 10002:localhost:10002 hadoop@ <i>MasterNodeDNS</i> -i \$HOME/ <i>mysecretkey.pem</i>
Hive 0.7 drivers	ssh -o ServerAliveInterval=10 -L 10001:localhost:10001 hadoop@ <i>MasterNodeDNS</i> -i \$HOME/ <i>mysecretkey.pem</i>
Hive 0.5 drivers	ssh -o ServerAliveInterval=10 -L 10000:localhost:10000 hadoop@ <i>MasterNodeDNS</i> -i \$HOME/ <i>mysecretkey.pem</i>

The *MasterNodeDNS* is the public DNS name of the master node of the Hadoop cluster and *mysecretkey.pem* is the name of your Amazon EC2 key file.

- Add the JDBC driver to SQuirrel SQL:
 - Open SQuirrel SQL and click the **Drivers** tab.
 - Double-click **JDBC ODBC Bridge** to add attributes.
 - Type `org.apache.hadoop.hive.jdbc.HiveDriver` in the **Class Name** field, and then click **Add**.
 - Navigate to the location of your JDBC drivers.
 - Add the following JAR files:

If you are installing...	Add the following...
Hive 0.11.0 drivers	<pre>hadoop-core-1.0.3.jar hive/lib/hive-exec-0.11.0.jar hive/lib/hive-jdbc-0.11.0.jar hive/lib/hive-metastore-0.11.0.jar hive/lib/hive-service-0.11.0.jar hive/lib/libfb303-0.9.0.jar lib/commons-logging-1.0.4.jar slf4j-api-1.6.1.jar</pre>
Hive 0.8.1 drivers	<pre>hadoop-core-1.0.3.jar hive/lib/hive-exec-0.8.1.jar hive/lib/hive-jdbc-0.8.1.jar hive/lib/hive-metastore-0.8.1.jar hive/lib/hive-service-0.8.1.jar hive/lib/libfb303.jar lib/commons-logging-1.0.4.jar slf4j-api-1.6.1.jar slf4j-log4j12-1.6.1.jar</pre>
Hive 0.7.1 drivers	<pre>hadoop-0.20-core.jar hive/lib/hive-exec-0.7.1.jar hive/lib/hive-jdbc-0.7.1.jar hive/lib/hive-metastore-0.7.1.jar hive/lib/hive-service-0.7.1.jar hive/lib/libfb303.jar lib/commons-logging-1.0.4.jar slf4j-api-1.6.1.jar slf4j-log4j12-1.6.1.jar</pre>
Hive 0.7 drivers	<pre>hadoop-0.20-core.jar hive/lib/hive-exec-0.7.0.jar hive/lib/hive-jdbc-0.7.0.jar hive/lib/hive-metastore-0.7.0.jar hive/lib/hive-service-0.7.0.jar hive/lib/libfb303.jar lib/commons-logging-1.0.4.jar slf4j-api-1.5.6.jar slf4j-log4j12-1.5.6.jar</pre>
Hive 0.5 drivers	<pre>hadoop-0.20-core.jar hive/lib/hive-exec-0.5.0.jar hive/lib/hive-jdbc-0.5.0.jar hive/lib/hive-metastore-0.5.0.jar hive/lib/hive-service-0.5.0.jar hive/lib/libfb303.jar hive/lib/log4j-1.2.15.jar lib/commons-logging-1.0.4.jar</pre>

- f. Click **OK**.
5. Add a new alias:
 - a. Click the **Alias** tab, and then click **+** to add a new alias.
 - b. Enter the following information in the **Add Alias** dialog:

Field	Description
Name	Enter the name of the alias.
Driver	Select the JDBC driver from the list.
User Name	Enter your local machine login.
Password	Enter your local machine password.

- c. Enter the **URL** information in the **Add Alias** dialog based on the version of Hive:

If you are installing...	Enter the following...
Hive 0.11.0 drivers	<code>jdbc:hive://localhost:10004/default</code>
Hive 0.8.1 drivers	<code>jdbc:hive://localhost:10003/default</code>
Hive 0.7.1 drivers	<code>jdbc:hive://localhost:10002/default</code>
Hive 0.7 drivers	<code>jdbc:hive://localhost:10001/default</code>
Hive 0.5 drivers	<code>jdbc:hive://localhost:10000/default</code>

- d. Click **OK**.

The SQuirrel SQL client is ready to use.

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>.

Analyze Data with 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 \(p. 262\)](#).

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 Data-bases

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 \(p. 265\)](#).

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 cluster with Impala installed
- Connect to the cluster using SSH
- Generate a test data set
- Create and Impala tables and populate them with data
- Perform interactive queries on Impala tables

In addition to the console used in this tutorial, Amazon EMR provides a command-line client, a REST-like API set, 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? \(p. 10\)](#).

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.

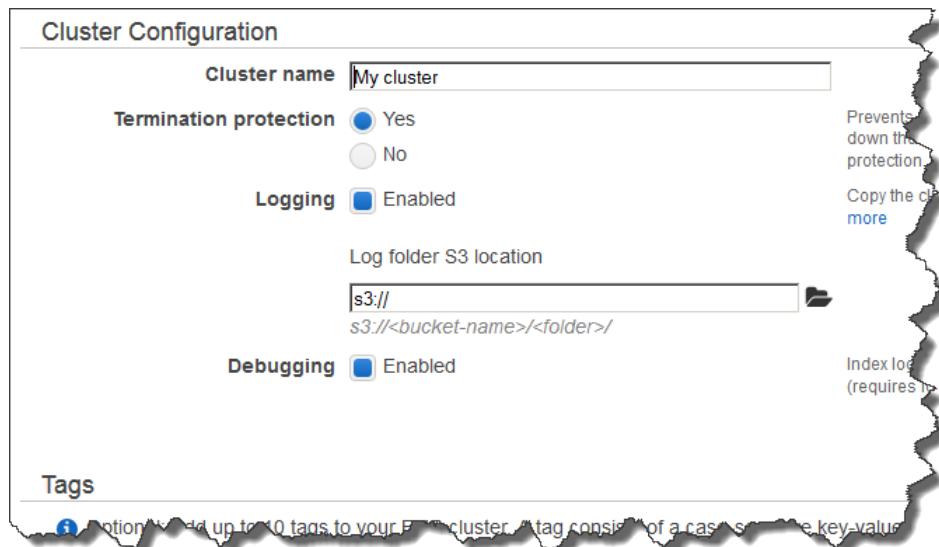
AWS notifies you by email when your account is active and available for you to use.

Launch the Cluster

The next step is to launch the cluster. This tutorial provides the steps to launch 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.

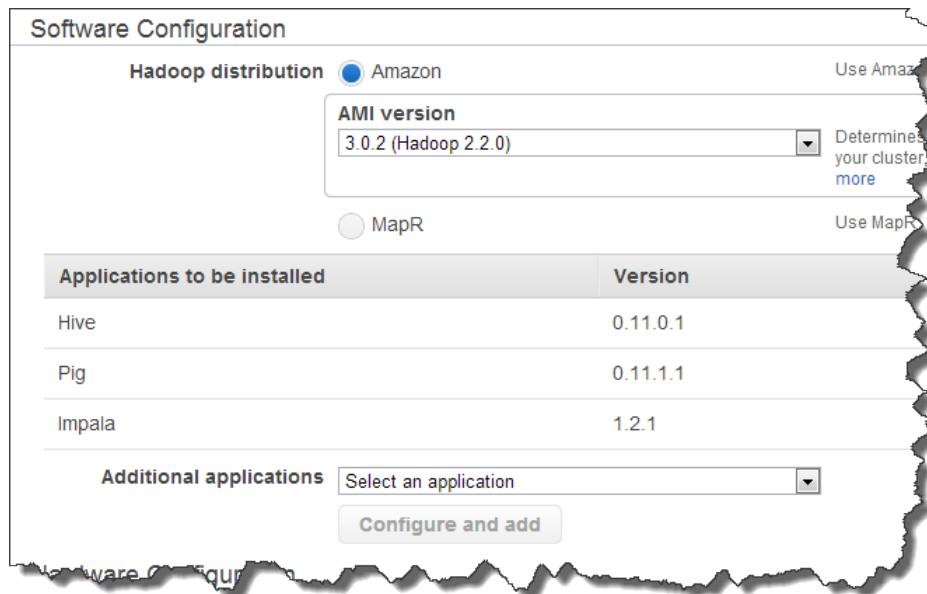
To launch an Impala cluster using the console

1. Open the Amazon Elastic MapReduce console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Click **Create cluster**.
3. In the **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>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 (p. 459). 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>This determines whether Amazon EMR captures detailed log data to Amazon S3.</p> <p>For more information, see View Log Files (p. 418).</p>
Log folder S3 location	<p>Enter an Amazon S3 path to store your debug logs if you enabled logging in the previous field. If the log folder does not exist, the Amazon EMR console creates it for you.</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 (p. 418).</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. 149).</p>
AMI version	<p>Choose 3.0.2 (Hadoop 2.2.0).</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 a Machine Image (p. 51).</p>

- 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 **AMI 3.0.1 (Hadoop 2.2.0)** or newer in the **AMI version** list mentioned in the previous steps.
- 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 1.2.1 - latest .
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 Command Line Interface Options (p. 585) .

- Click **Add**.
- 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](#).

Hardware Configuration

Tip Specify the [networking](#) and [hardware](#) configuration for your cluster. If you need more than 20 nodes, [Request Spot instances](#) (unused EC2 capacity) to save money.

Network	Launch into EC2-Classic	Use data	
EC2 availability zone	No preference	Launch	
EC2 instance type	Count	Request spot	
Master	m1.large	1	<input type="checkbox"/>
Core	m1.large	2	<input type="checkbox"/>
Task	m1.small	0	<input type="checkbox"/>

Security and Access

Field	Action
Network	<p>Choose Launch into EC2-Classic.</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) (p. 137).</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.small (default), m1.large, m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cc1.4xlarge, cg1.4xlarge.</p> <p> This tutorial uses small instances for all nodes due to the light workload and to keep your costs low.</p> <p> For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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) (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.small (default), m1.large, m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cc1.4xlarge, cg1.4xlarge.</p> <p>This tutorial uses small instances for all nodes due to the light workload and to keep your costs low.</p> <p>For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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) (p. 37).</p>
Task	<p>Choose m1.small.</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. Valid types: m1.small (default), m1.large, m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cc1.4xlarge, cg1.4xlarge.</p> <p>For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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) (p. 37).</p>

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

Security and Access

EC2 key pair [Proceed without an EC2 key pair](#) Use an existing key pair to SSH into the master node of the Amazon EC2 cluster as the user "hadoop". [Learn more](#)

IAM user access All other IAM users Control the visibility of this cluster to other IAM users. [Learn more](#)
 No other IAM users

IAM Roles

EMR role [No roles found](#) Allows EMR to access other AWS Services such as EC2 on your behalf. [Learn more](#)
[Create Default Role](#)

EC2 instance profile [Proceed without role](#) Allows EC2 instances in an EMR cluster to access other AWS services such as S3. [Learn more](#)
[Create Default Role](#)

Field	Action
EC2 key pair	<p>Choose an Amazon EC2 key pair from the list.</p> <p>For more information, see Create an Amazon EC2 Key Pair and PEM File (p. 117).</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 Cluster (p. 446).</p>
IAM user access	<p>Choose No other IAM users.</p> <p> 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 (p. 119).</p>
EC2 instance profile	<p>You can proceed without choosing an instance profile. If you create a cluster without selecting a specific instance profile, one will be created for you.</p> <p>This controls application access to the Amazon EC2 instances in the cluster.</p> <p>For more information, see Configure IAM Roles for Amazon EMR (p. 125).</p>
EMR role	<p>Choose Proceed without role.</p> <p>Allows Amazon EMR to access other AWS services on your behalf.</p> <p>For more information, see Configure IAM Roles for Amazon EMR (p. 125).</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 [Create Bootstrap Actions to Install Additional Software \(Optional\) \(p. 87\)](#).

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, click **Create Cluster**.
13. When the cluster starts, the console displays the **Cluster Details** page.

To create an Impala cluster using the Amazon EMR CLI

- In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).
 - Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --alive --instance-type m1.large --instance-count 3 --ami-version 3.0.2 --impala-interactive --key-pair keypair-name
```

- Windows users:

```
ruby elastic-mapreduce --create --alive --instance-type m1.large --instance-count 3 --ami-version 3.0.2 --impala-interactive --key-pair keypair-name
```

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 \(p. 446\)](#).
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 /tmp/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 /tmp/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 /tmp/dbgen/* /data/  
hadoop fs -ls -h -R /data/
```

Create and Populate Impala Tables

In this section, you create the Impala tables and fill them with test data.

To create and populate the Impala tables with the test data

1. In the SSH window, launch the Impala shell prompt using the following command:

```
impala-shell
```

2. Create and populate the `books` table with the test data by running the following command at the Impala shell prompt:

```
create EXTERNAL TABLE books( id BIGINT, isbn STRING, category STRING, publish_date TIMESTAMP, publisher STRING, price FLOAT )
    ROW FORMAT DELIMITED FIELDS TERMINATED BY '|'
    LOCATION '/data/books/' ;
```

3. Create and populate the `customers` table with the test data by running the following command at the Impala shell prompt:

```
create EXTERNAL TABLE customers( id BIGINT, name STRING, date_of_birth
    TIMESTAMP, gender STRING, state STRING, email STRING, phone STRING )
    ROW FORMAT DELIMITED FIELDS TERMINATED BY '|'
    LOCATION '/data/customers/' ;
```

4. Create and populate the `transactions` table with the test data by running the following command at the Impala shell prompt:

```
create EXTERNAL TABLE transactions( id BIGINT, customer_id BIGINT, book_id
    BIGINT, quantity INT, transaction_date TIMESTAMP )
    ROW FORMAT DELIMITED FIELDS TERMINATED BY '|'
    LOCATION '/data/transactions/' ;
```

Query Data in Impala

In this section, you perform queries on the data that you loaded in the Impala tables in the previous steps.

To perform various queries on the test data in the Impala tables

1. Perform a table scan through the entire `customers` table by running the following query at the Impala shell prompt:

```
SELECT COUNT(*)
FROM customers
WHERE name = 'Harrison SMITH' ;
```

2. Perform an aggregation query that scans a single table, groups the rows, and calculates the size of each group by running the following query at the Impala shell prompt:

```
SELECT category, count(*) cnt
FROM books
GROUP BY category
ORDER BY cnt DESC LIMIT 10;
```

3. Perform a query that joins the books table with the transactions table to determine the top ten book categories that have the maximum total revenue during a certain period of time by running the following query at the Impala shell prompt:

```
SELECT tmp.book_category, ROUND(tmp.revenue, 2) AS revenue
FROM (
    SELECT books.category AS book_category, SUM(books.price * transactions.quantity) AS revenue
    FROM books JOIN [SHUFFLE] transactions ON (
        transactions.book_id = books.id
        AND YEAR(transactions.transaction_date) BETWEEN 2008 AND 2010
    )
    GROUP BY books.category
) tmp
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.quantity) AS revenue
    FROM books
    JOIN [SHUFFLE] transactions ON (
        transactions.book_id = books.id
    )
    JOIN [SHUFFLE] customers ON (
        transactions.customer_id = customers.id
        AND customers.state IN ('WA', 'CA', 'NY')
    )
    GROUP BY books.category
) tmp
ORDER BY revenue DESC LIMIT 10;
```

Important

Now that you've completed the tutorial, you should terminate the cluster to ensure that your account does not accrue additional charges. For more information, see [Terminate a Cluster \(p. 458\)](#).

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 \(p. 258\)](#)
- [Wikipedia \(p. 259\)](#)

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_demographics.cd_demo_sk)  
JOIN store on (store_sales.ss_store_sk = store.s_store_sk)  
where  
    cd_gender = 'M' and  
    cd_marital_status = 'S' and  
    cd_education_status = 'College' and  
    d_year = 2002 and  
    s_state in ('TN', 'SD', 'SD', 'SD', 'SD', 'SD')  
group by  
    i_item_id,  
    s_state  
order by  
    i_item_id,  
    s_state  
limit 100;
```

2. Impala can create and manage Parquet tables. Parquet is a column-oriented binary file format intended to be highly efficient for scanning particular columns within a table. For more information, go to <http://parquet.io/>. After running the query, test the Parquet format by running the following script:

```
./tpcds-samplequery-parquet.sh
```

Wikipedia

The Wikipedia example uses the data and sample queries from the Shark example in GitHub. For more information, go to <https://github.com/amplab/shark/wiki/Running-Shark-on-EC2>.

To run the Wikipedia example

- On the master node of the cluster, navigate to the examples directory and run the following script:

```
cd ~/impala/examples/wikipedia/  
./wikipedia.sh
```

Alternatively, you can use this script instead:

```
./wikipedia-with-s3distcp.sh
```

The `wikipedia.sh` and `wikipedia-with-s3distcp.sh` scripts copy 42 GB of data from Amazon S3 to HDFS, create Hive tables, and use Impala to select data from the Hive tables. The difference between `wikipedia.sh` and `wikipedia-with-s3distcp.sh` is that `wikipedia.sh` uses Hadoop `distcp` to copy data from Amazon S3 to HDFS, but `wikipedia-with-s3distcp.sh` uses Amazon EMR S3DistCp for the same purpose.

The `wikipedia-with-s3distcp.sh` script contains the following code:

```
#!/bin/bash  
  
. /home/hadoop/impala/conf/impala.conf  
  
# 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_NAMENODE_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" "\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 a Machine Image \(p. 51\)](#). 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 \(p. 260\)](#)

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 \(p. 265\)](#).

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 [Open an SSH Tunnel to the Master Node \(p. 452\)](#).

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 [Open an SSH Tunnel to the Master Node \(p. 452\)](#) and [Configure FoxyProxy 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.

Statements

Impala supports the following SQL statements:

- ALTER TABLE
- ALTER VIEW
- [BROADCAST]
- CREATE DATABASE
- CREATE EXTERNAL TABLE
- CREATE FUNCTION
- CREATE TABLE
- CREATE VIEW
- DESCRIBE
- DROP DATABASE
- DROP FUNCTION
- DROP TABLE
- DROP VIEW
- EXPLAIN
- INSERT
- INVALIDATE METADATA
- JOIN
- LOAD DATA
- NULL
- REFRESH
- SELECT
- SHOW
- [SHUFFLE]
- USE

Functions

Impala supports the following SQL functions:

- AVG
- COUNT
- MAX
- MIN
- NDV

- SUM

Data Types

Impala supports the following SQL data types:

- BIGINT
- BOOLEAN
- DOUBLE
- FLOAT
- INT
- SMALLINT
- STRING
- TIMESTAMP
- TINYINT

Operators

Impala supports the following SQL operators:

- =, !=, <, <=, >, >=, <>
- /*, */ (Comments)
- BETWEEN
- DISTINCT
- IS NULL
- IS NOT NULL
- LIKE
- REGEXP
- RLIKE

Clauses

Impala supports the following SQL clauses:

- GROUP BY
- HAVING
- LIMIT
- OFFSET
- ORDER BY
- UNION
- VALUES
- WITH

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 \(p. 249\)](#).

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 \(p. 255\)](#).

```
$ 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.

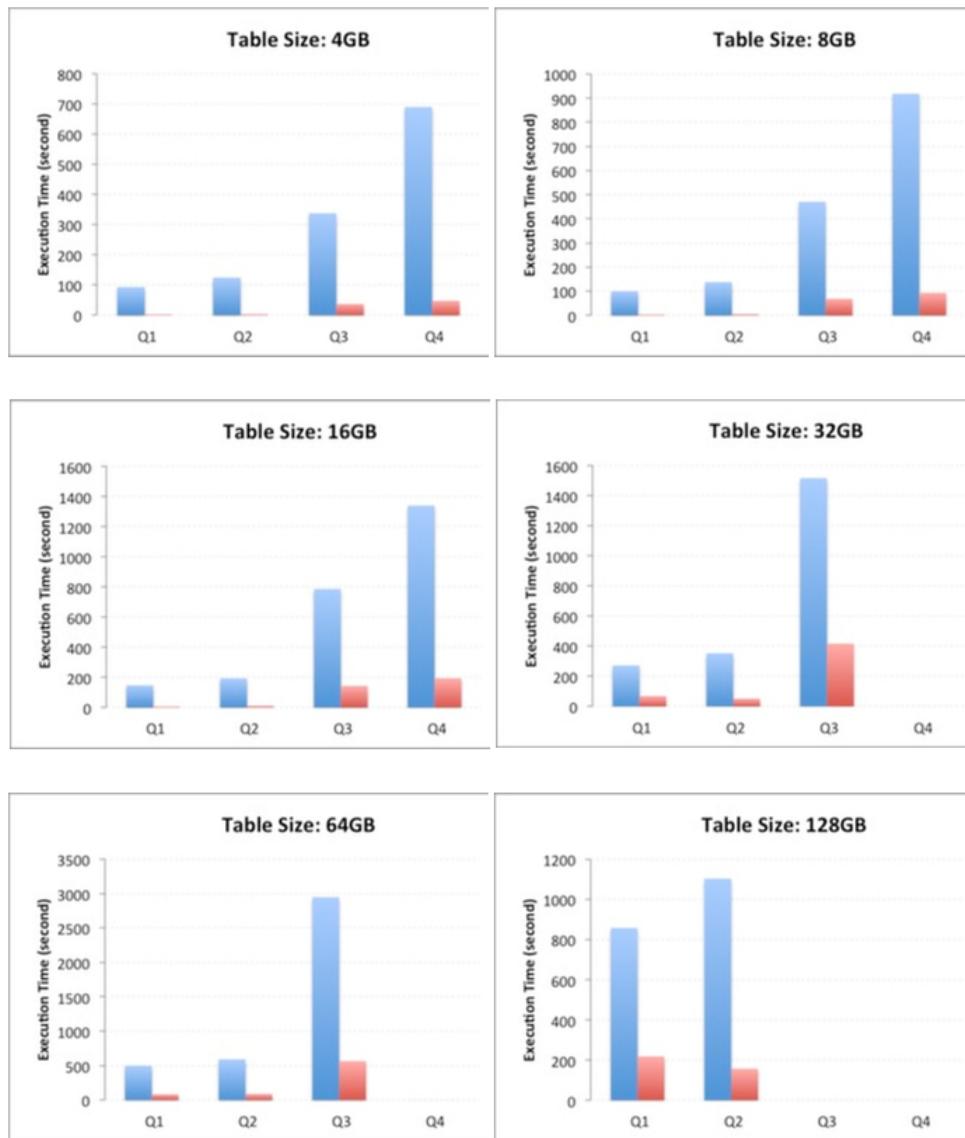
Performance Test Results

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.

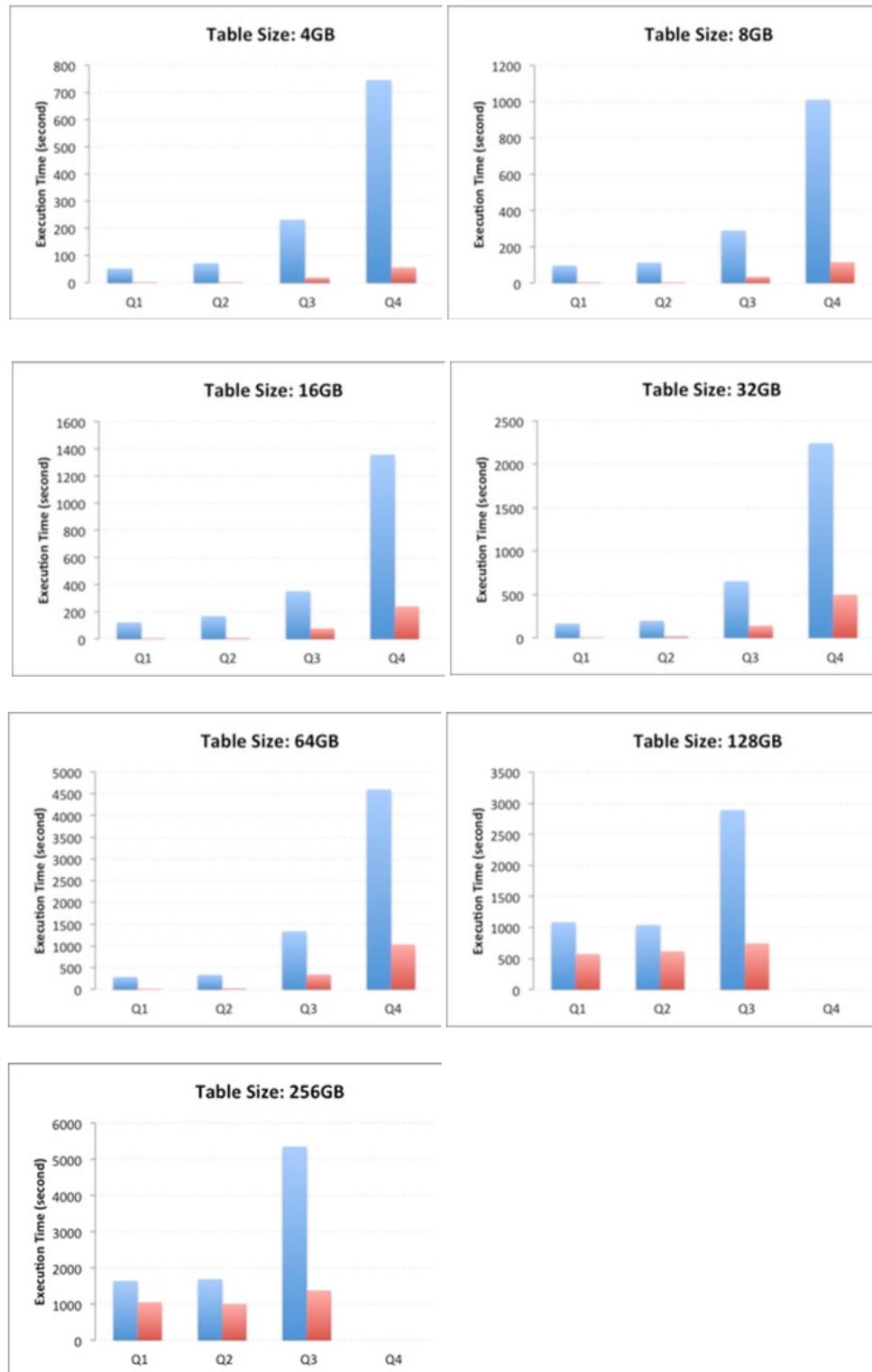
From these figures, we observed that at smaller scales (in this experiment, 16 GB and lower), Impala is much faster than Hive due to the absence of the MapReduce framework overhead. Nonetheless, when the input data set is large enough such that the framework overhead is negligible compared to overall 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.quantity) 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.quantity) AS revenue
  FROM books JOIN transactions ON (
    transactions.book_id = books.id
    AND YEAR(transactions.transaction_date) BETWEEN 2008 AND 2010
  )
  GROUP BY books.category
) tmp
ORDER BY revenue DESC LIMIT 10;
```

The second version of Q3 is more efficient, because only a part of the transactions table is broadcast instead of the entire books table. Nonetheless, when we scaled up to the 32 GB input class, even the second version of Q3 started to fail due to memory constraints. To further optimize the query, we added a hint to force Impala to use the "partitioned join," which creates the final version of Q3 as shown above in the Queries section. With all the optimizations, we eventually managed to execute Q3 successfully for input classes up to 64 GB, giving us a 16x more memory-efficient query than the first version. There are many other ways to optimize queries for Impala, and we see these methods as a good way to get the best performance from your hardware and avoid out-of-memory errors.

Process Data with Pig

Amazon Elastic MapReduce (Amazon EMR) supports Apache Pig, a platform you can use to analyze large data sets. For more information about Pig, go to <http://pig.apache.org/>. Amazon EMR supports several versions of Pig. The following sections describe how to configure Pig on Amazon EMR.

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 clusters. Pig enables you to create database types of queries using familiar SQL-like commands and syntax, so you do not have to write complex MapReduce algorithms using a lower level computer language, such as Java. Although you can execute one Pig Latin command at a time, it is far more common to write a script of Pig Latin commands that accomplish a complete task. Amazon EMR can use these scripts when you upload them to Amazon S3.

Topics

- [Supported Pig Versions \(p. 273\)](#)
- [Interactive and Batch Pig Clusters \(p. 279\)](#)
- [Launch a Pig Cluster \(p. 281\)](#)
- [Call User Defined Functions from Pig \(p. 287\)](#)

Supported Pig Versions

The Pig version you can run 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. To select the configuration, use the `--ami-version` and `--pig-versions` parameters in the cluster creation call. For more information about the Amazon EMR AMIs and AMI versioning, see [Choose a Machine Image \(p. 51\)](#).

Amazon EMR attempts to use the latest version of Pig if you do not specify a version number.

The Amazon EMR console does not support Pig versioning and always launches the latest version of Pig.

You can manually specify a Pig version (using the `--pig-versions` parameter) with the Amazon EMR CLI version 2013-07-08 or newer, available from <http://aws.amazon.com/code/Elastic-MapReduce/2264>.

When you call the API, you will launch the default configuration of Pig 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 Release Notes Documentation	3.1.0 and later	<code>--ami-version 3.1.0</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 BigDecimal 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. 149) .

Pig Version	AMI Version	Configuration Parameters	Pig Version Details
0.9.2 Release Notes Documentation	2.2 and later	--pig-versions 0.9.2 --ami-version 2.2	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	--pig-versions 0.9.1 --ami-version 2.0	
0.6 Release Notes	1.0	--pig-versions 0.6 --ami-version 1.0	
0.3 Release Notes	1.0	--pig-versions 0.3 --ami-version 1.0	

To specify the Pig version when creating the 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 following, `instanceType` would be replaced by an EC2 instance type such as `m1.small`.

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --alive --name "Test Pig" \
--ami-version 2.3.6 \
--num-instances 5 --instance-type instanceType \
--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 instanceType --pig-interactive --
pig-versions 0.11.1
```

To specify the latest Pig version when creating the 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 following, `instanceType` would be replaced by an EC2 instance type such as `m1.small`.

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --alive --name "Test Latest Pig" \
--ami-version 2.2 \
--num-instances 5 --instance-type instanceType \
--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 instanceType --pig-interactive
--pig-versions latest
```

To load multiple versions of Pig for a given 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 following, `instanceType` would be replaced by an EC2 instance type such as `m1.small`.

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --alive --name "Test Pig" \
--ami-version 2.0 \
--num-instances 5 --instance-type instanceType \
--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 instanceType --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 call a specific version of Pig

- 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
```

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 [Process Data with Pig \(p. 273\)](#). 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	<p>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.</p> <p>Status: Committed Fixed in Apache Pig Version: 0.10</p>

Patch	Description
PIG-1824	<p>Support import modules in Jython UDF. For more information, go to https://issues.apache.org/jira/browse/PIG-1824.</p> <p>Status: Committed Fixed in Apache Pig Version: 0.10</p>
PIG-2010	<p>Bundle registered JARs on the distributed cache. For more information, go to https://issues.apache.org/jira/browse/PIG-2010.</p> <p>Status: Committed Fixed in Apache Pig Version: 0.11</p>
PIG-2456	<p>Add a <code>~/.pigbootup</code> file where the user can specify default Pig statements. For more information, go to https://issues.apache.org/jira/browse/PIG-2456.</p> <p>Status: Committed Fixed in Apache Pig Version: 0.11</p>
PIG-2623	<p>Support using Amazon S3 paths to register UDFs. For more information, go to https://issues.apache.org/jira/browse/PIG-2623.</p> <p>Status: Committed Fixed in Apache Pig Version: 0.10, 0.11</p>

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	<p>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.</p> <p>Status: Committed Fixed in Apache Pig Version: 0.8.0</p>
Support multiple file systems in Pig	<p>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.</p> <p>Status: Not Committed Fixed in Apache Pig Version: n/a</p>
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

Typically, you use interactive mode to troubleshoot your cluster and batch mode in production. After you revise the Pig Latin script using the interactive mode, you should upload it to Amazon S3 and use the batch mode to run clusters. For more information about using SSH, go to [View Log Files \(p. 418\)](#).

In interactive mode, you SSH as the Hadoop user into the master node in the Hadoop cluster and run the Pig Latin script on it so that you can debug it. The interactivity of this mode enables you to revise the Pig Latin script quicker than you could in batch mode.

In batch mode, you create a Pig script using Pig Latin and then load that script into Amazon S3. When you run a cluster using Pig, the first step is to download that script from Amazon S3 so that it is used in the Amazon EMR cluster. When you use the Amazon EMR console this download is done automatically for you. You use batch mode to run clusters in production.

Run Pig in Interactive Mode

To run Pig in interactive mode use the `alive` option with the `create` command so that the cluster remains active until you terminate it.

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --alive --name "Testing Pig -- $USER" \
--num-instances 5 --instance-type instanceType \
--pig-interactive
```

- Windows users:

```
ruby elastic-mapreduce --create --alive --name "Testing Pig -- $USER" --num-
instances 5 --instance-type instanceType --pig-interactive
```

The return is similar to the following:

```
Created jobflow JobFlowID
```

You are now running Pig in interactive mode and can execute Pig queries.

Run Pig in Batch Mode

The following process shows how to run Pig in batch mode and assumes that you stored the Pig script in a bucket on Amazon S3. For more information about uploading files into Amazon S3, see the [Amazon S3 Getting Started Guide](#).

To run Pig in batch mode, create a cluster with a step that executes a Pig script stored on Amazon S3.

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create \
--name "$USER's Pig JobFlow" \
--pig-script \
--args s3://myawsbucket/myquery.q \
--args -p,INPUT=s3://myawsbucket/input,-p,OUTPUT=s3://myawsbucket/output
```

- Windows users:

```
ruby elastic-mapreduce --create --name "$USER's Pig JobFlow" --pig-script --args s3://myawsbucket/myquery.q --args -p,INPUT=s3://myawsbucket/input,-p,OUTPUT=s3://myawsbucket/output
```

The `args` option provides arguments to the Pig script. The first `args` option specifies the location of the Pig script in Amazon S3. In the second `args` option, the `-p` provides a way to pass values into the script (`INPUT`) and where to store results (`OUTPUT`). Within the Pig script these parameters are available as `$(variable)`. So, in this example Pig replaces `$(INPUT)` and `$(OUTPUT)` with the values passed in. These variables are substituted as a preprocessing step, so they can occur anywhere in a Pig script. The return is similar to the following:

```
Created jobflow JobFlowID
```

You might need to add Pig as a new step in an existing cluster. Adding steps can help you test and develop Pig scripts. For example, if the script fails, you can add a new step to the cluster without having to wait for a new cluster to start.

To add a Pig script to an existing cluster in batch mode, specify the location in Amazon S3 of a Pig script and associate it with an existing cluster.

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --jobflow JobFlowID \
--pig-script \
--args s3://myawsbucket/myquery.q \
--args -p,INPUT=s3://myawsbucket/input,-p,OUTPUT=s3://myawsbucket/output
```

- Windows users:

```
ruby elastic-mapreduce --jobflow JobFlowID --pig-script --args s3://myawsbucket  
et/myquery.q --args -p,INPUT=s3://myawsbucket/input,-p,OUTPUT=s3://myawsbucket  
et/output
```

Launch a Pig Cluster

This section covers the basics of creating a cluster using Pig in Amazon Elastic MapReduce (Amazon EMR). You'll step through how to create a cluster using Pig with either the Amazon EMR console, the CLI, or the Query API. Before you create your cluster you'll need to create objects and permissions; for more information see [Prepare Input Data \(Optional\) \(p. 98\)](#).

A cluster using Pig takes SQL-like commands written in Pig Latin, and converts those commands into Hadoop MapReduce algorithms. 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 `s3n://elasticmapreduce/samples/pig-apache/do-reports2.pig`. Input data is located in the Amazon S3 bucket `s3n://elasticmapreduce/samples/pig-apache/input`. The output is saved to an Amazon S3 bucket you created as part of [Prepare an Output Location \(Optional\) \(p. 112\)](#).

Amazon EMR Console

This example describes how to use the Amazon EMR console to create a cluster using Pig.

To create a Pig cluster using the console

Next, Amazon EMR runs the cluster to perform the work you specified.

When the cluster is finished processing the data, Amazon EMR copies the results into the output Amazon S3 bucket that you chose in the previous steps. Verify that your Amazon S3 output bucket contains a set of folders that contain aggregates derived from the input data, similar to the following:

All Buckets / examples-bucket / pig-apache / output / 2013-09-26	
	Name
<input type="checkbox"/>	 top_50_external_referrers
<input type="checkbox"/>	 top_50_ips
<input type="checkbox"/>	 top_50_search_terms_from_bing_google
<input type="checkbox"/>	 total_requests_bytes_per_hour

1. Open the Amazon Elastic MapReduce console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Click **Create cluster**.
3. In the **Create Cluster** page, click **Configure sample application**.
4. In the **Configure Sample Application** page, choose the **Apache log reports (Pig script)** sample application from the list.

5. In the **Output location** field, type the path of an Amazon S3 bucket to store your output and click **Ok**.
6. In the **Create Cluster** page, in the **Cluster Configuration** section, verify the fields according to the following table.

Cluster Configuration

Cluster name	<input type="text" value="My cluster"/>	Prevents cluster from being terminated
Termination protection	<input checked="" type="radio"/> Yes <input type="radio"/> No	Copy the cluster configuration to another cluster
Logging	<input checked="" type="checkbox"/> Enabled	Index logs (requires Hadoop 2.0.0 or later)
Log folder S3 location	<input type="text" value="s3://<bucket-name>/<folder>"/> <small>s3://<bucket-name>/<folder>/</small>	
Debugging	<input checked="" type="checkbox"/> Enabled	
Tags <small>Add up to 10 tags to your EMR cluster. A tag consists of a case-sensitive key-value pair.</small>		

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>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 (p. 459). 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>This determines whether Amazon EMR captures detailed log data to Amazon S3.</p> <p>For more information, see View Log Files (p. 418).</p>
Log folder S3 location	<p>Enter an Amazon S3 path to store your debug logs if you enabled logging in the previous field. If the log folder does not exist, the Amazon EMR console creates it for you.</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 (p. 418).</p>

Field	Action
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.

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

Software Configuration

Hadoop distribution Amazon Use Amazon's Hadoop distribution. [Learn more](#)

AMI version Determines the base configuration of the instances in your cluster, including the Hadoop version. [Learn more](#)

MapR Use MapR's Hadoop distribution. [Learn more](#)

Applications to be installed	Version	Actions
Hive	0.11.0.1	  
Pig	0.11.1.1	  
Additional applications	Select an application	<input type="button" value="Configure and add"/>

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. 149).</p>
AMI version	<p>Choose 2.4.2 (Hadoop 1.0.3).</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 (p. 51).</p>

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, 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](#).

Hardware Configuration

Specify the networking and hardware configuration for your cluster. If you need more than 20 EC2 Request Spot instances (unused EC2 capacity) to save money.

Network	<input type="text" value="vpc-c1a3b3a3 (172.31.0.0/16) (default)"/>	<input type="checkbox"/> Use a Virtual Private Cloud	
EC2 Subnet	<input type="text" value="No preference (random subnet)"/>	Create a Subnet	
EC2 instance type	Count	Request spot	
Master	<input type="text" value="m1.small"/>	1 <input type="checkbox"/>	The Master instance runs the task nodes, and monitors their status.
Core	<input type="text" value="m1.small"/>	2 <input type="checkbox"/>	Core instances run the Hadoop Distributed File System.
Task	<input type="text" value="m1.small"/>	0 <input type="checkbox"/>	Task instances run Hadoop MapReduce jobs.

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 a Amazon VPC Subnet for the Cluster (Optional) (p. 137).</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 Elastic Compute Cloud User Guide</i>.</p>
Master	<p>Choose m1.small.</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.small (default), m1.large, m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cc1.4xlarge, cg1.4xlarge.</p> <p> This tutorial uses small instances for all nodes due to the light workload and to keep your costs low.</p> <p> For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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 Lower Costs with Spot Instances (Optional) (p. 37).</p>

Field	Action
Core	<p>Choose m1.small.</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.small (default), m1.large, m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cc1.4xlarge, cg1.4xlarge.</p> <p>This tutorial uses small instances for all nodes due to the light workload and to keep your costs low.</p> <p>For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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) (p. 37).</p>
Task	<p>Choose m1.small.</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. Valid types: m1.small (default), m1.large, m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cc1.4xlarge, cg1.4xlarge.</p> <p>For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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) (p. 37).</p>

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

Security and Access

EC2 key pair	<input type="button" value="Proceed without an EC2 key pair"/>	<small>Use an existing key pair to SSH into the master node of the Amazon EC2 cluster as the user "hadoop". Learn more</small>
IAM user access		
<input type="radio"/> All other IAM users		
<input checked="" type="radio"/> No other IAM users		
IAM Roles		
<small>EMR role <input type="button" value="No roles found"/></small>		
<small>Create Default Role</small>		
<small>EC2 instance profile <input type="button" value="Proceed without role"/></small>		
<small>Create Default Role</small>		

Field	Action
EC2 key pair	<p>Choose an Amazon EC2 key pair from the list.</p> <p>For more information, see Create an Amazon EC2 Key Pair and PEM File (p. 117).</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 Cluster (p. 446).</p>
IAM user access	<p>Choose No other IAM users.</p> <p> 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 (p. 119).</p>
EC2 instance profile	<p>You can proceed without choosing an instance profile. If you create a cluster without selecting a specific instance profile, one will be created for you.</p> <p>This controls application access to the Amazon EC2 instances in the cluster.</p> <p>For more information, see Configure IAM Roles for Amazon EMR (p. 125).</p>
EMR role	<p>Choose Proceed without role.</p> <p>Allows Amazon EMR to access other AWS services on your behalf.</p> <p>For more information, see Configure IAM Roles for Amazon EMR (p. 125).</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 [Create Bootstrap Actions to Install Additional Software \(Optional\) \(p. 87\)](#).

11. In the **Steps** section, note the step that Amazon EMR configured for you by choosing the sample application.

You do not need to change any of the settings in this section.

12. Review your configuration and if you are satisfied with the settings, click **Create Cluster**.
13. When the cluster starts, the console displays the **Cluster Details** page.

To create a Pig cluster using the CLI

- In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).
 - Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --name "Test Pig" \
--pig-script s3n://elasticmapreduce/samples/pig-apache/do-reports2.pig \
--ami-version 2.0 \
--args "-p,INPUT=s3n://elasticmapreduce/samples/pig-apache/input, \
-p,OUTPUT=s3n://myawsbucket/pig-apache/output"
```

- Windows users:

```
ruby elastic-mapreduce --create --name "Test Pig" --pig-script
s3n://elasticmapreduce/samples/pig-apache/do-reports2.pig --ami-version
2.0 --args "-p,INPUT=s3n://elasticmapreduce/samples/pig-apache/input, -
p,OUTPUT=s3n://myawsbucket/pig-apache/output"
```

The output looks similar to the following.

```
Created cluster JobFlowID
```

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.

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/r0.9.2/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://myawsbucket/path/to/my/uploaded.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://myawsbucket/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 [Launch a Pig Cluster \(p. 281\)](#).
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://myawsbucket/pig/python/myscript.py' using jython as
myfunctions;
```

6. Load the input data. The following example loads input from an Amazon S3 location.

```
grunt> input = load 's3://myawsbucket/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);
```

Store Data with 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 \(p. 290\)](#)
- [HBase Cluster Prerequisites \(p. 290\)](#)
- [Launch an HBase Cluster on Amazon EMR \(p. 291\)](#)
- [Connect to HBase Using the Command Line \(p. 298\)](#)
- [Back Up and Restore HBase \(p. 298\)](#)
- [Terminate an HBase Cluster \(p. 309\)](#)
- [Configure HBase \(p. 309\)](#)
- [Access HBase Data with Hive \(p. 313\)](#)
- [View the HBase User Interface \(p. 315\)](#)
- [View HBase Log Files \(p. 315\)](#)
- [Monitor HBase with CloudWatch \(p. 316\)](#)
- [Monitor HBase with Ganglia \(p. 317\)](#)

Supported HBase Versions

HBase Version	AMI Version	Configuration Parameters	HBase Version Details
0.94.18	3.1.0 and later	--ami-version 3.1.0 --hbase	• Bux fixes and enhancements.
0.94.7	3.0-3.0.4	--ami-version 3.0 / 3.0.1 / 3.0.2 / 3.0.3 / 3.0.4 --hbase	
0.92	2.2 and later	--ami-version 2.2 or later --hbase	

HBase Cluster Prerequisites

An Amazon EMR cluster should meet the following requirements in order to run HBase.

- **A version of the Amazon EMR command line interface (CLI) that supports HBase (Optional)**—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. For more information about the Amazon EMR CLI and how to install it, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#). If you do not have the latest version of the CLI installed, you can use the Amazon EMR console to launch HBase clusters.
- **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.

- **Persistent cluster**—HBase only runs on persistent clusters. The CLI and Amazon EMR console automatically create HBase clusters with the `--alive` flag set.
- **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 set an Amazon EC2 key pair when you create the cluster.
- **The correct instance type**—HBase is only supported on the following instance types: m1.large, m1.xlarge, c1.xlarge, m2.2xlarge, m2.4xlarge, cc1.4xlarge, cc2.8xlarge, hi1.4xlarge, or hs1.8xlarge.

The cc2.8xlarge instance type is only supported in the US East (Northern Virginia), US West (Oregon), and EU (Ireland) regions. The cc1.4xlarge and hs1.8xlarge instance types are only supported in the US East (Northern Virginia) region. The hi1.4xlarge instance type is only supported in the US East (Northern Virginia) and EU (Ireland) regions.

- **The correct AMI and Hadoop versions**—HBase clusters are currently supported only on Hadoop 2.0.205 or later. The CLI and Amazon EMR console automatically set the correct AMI on HBase clusters.
- **Ganglia (Optional)**—If you want to monitor HBase performance metrics, you can use a bootstrap action to 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.

Launch an HBase Cluster on Amazon EMR

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 EMR team has tuned HBase to run optimally on AWS. For more information about HBase and running it on Amazon EMR, see [Store Data with HBase \(p. 289\)](#).

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 \(p. 309\)](#).

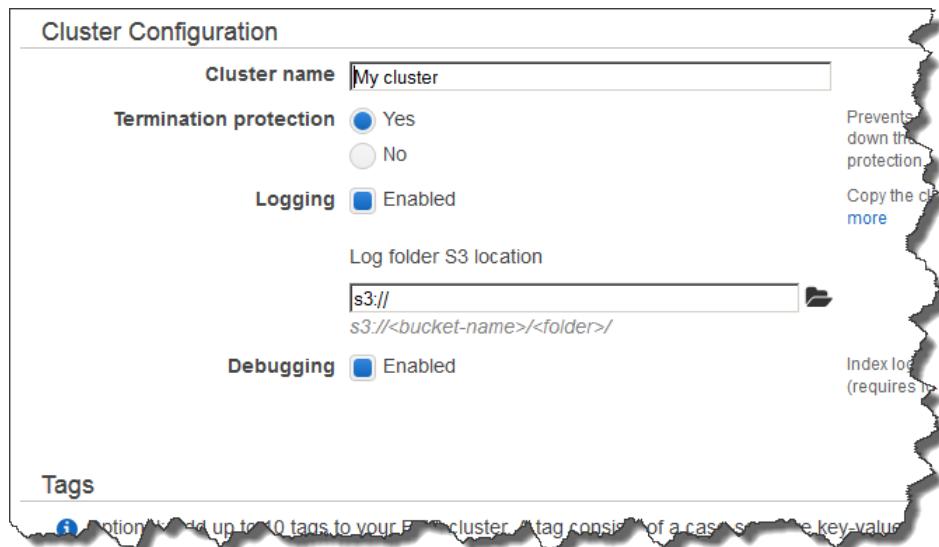
Note

HBase configuration can only be done at launch time.

For production environments, we recommend that you launch HBase on one cluster and launch any analysis tools, such as Hive, on a separate cluster. This ensures that HBase has ready access to the CPU and memory resources it requires.

To launch an HBase 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 **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>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 (p. 459). 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>This determines whether Amazon EMR captures detailed log data to Amazon S3.</p> <p>For more information, see View Log Files (p. 418).</p>
Log folder S3 location	<p>Enter an Amazon S3 path to store your debug logs if you enabled logging in the previous field. If the log folder does not exist, the Amazon EMR console creates it for you.</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 (p. 418).</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.

Software Configuration

Hadoop distribution Amazon Use Amazon's Hadoop distribution. [Learn more](#)

AMI version
2.4.2 (hadoop 1.0.3) - latest Determines the base configuration of the instances in your cluster, including the Hadoop version. [Learn more](#)

MapR Use MapR's Hadoop distribution. [Learn more](#)

Applications to be installed	Version	Actions
Hive	0.11.0.1	  
Pig	0.11.1.1	  
Additional applications	Select an application <input type="button" value="▼"/>	<input type="button" value="Configure and add"/>

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. 149).</p>
AMI version	<p>Choose 2.4.2 (Hadoop 1.0.3).</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 (p. 51).</p>

- Under the **Additional Applications** list, choose **HBase** and click **Configure and add**.
- 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. 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 \(p. 298\)](#).

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](#).

Hardware Configuration

Note: Specify the networking and hardware configuration for your cluster. If you need more than 20 EC2 Request Spot Instances (unused EC2 capacity) to save money.

EC2 instance type	Count	Request spot		
Master	m1.small	1	<input type="checkbox"/>	The Master instance is the primary node for the cluster, and is responsible for managing task nodes, and the HDFS NameNode.
Core	m1.small	2	<input type="checkbox"/>	Core instances are used for the Hadoop Distributed File System (HDFS) DataNodes.
Task	m1.small	0	<input type="checkbox"/>	Task instances are used for Map and Reduce tasks.

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 a Amazon VPC Subnet for the Cluster (Optional) (p. 137).</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 Elastic Compute Cloud User Guide</i>.</p>
Master	<p>Choose m1.small.</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.small (default), m1.large, m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cc1.4xlarge, cg1.4xlarge.</p> <p>This tutorial uses small instances for all nodes due to the light workload and to keep your costs low.</p> <p>For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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 Lower Costs with Spot Instances (Optional) (p. 37).</p>
Core	<p>Choose m1.small.</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.small (default), m1.large, m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cc1.4xlarge, cg1.4xlarge.</p> <p>This tutorial uses small instances for all nodes due to the light workload and to keep your costs low.</p> <p>For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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) (p. 37).</p>

Field	Action
Task	<p>Choose m1.small.</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. Valid types: m1.small (default), m1.large, m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cc1.4xlarge, cg1.4xlarge.</p> <p>For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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) (p. 37).</p>

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

Security and Access

EC2 key pair Use an existing key pair to SSH into the master node of the Amazon EC2 cluster as the user "hadoop". [Learn more](#)

IAM user access All other IAM users No other IAM users Control the visibility of this cluster to other IAM users. [Learn more](#)

IAM Roles

EC2 instance profile Allows EC2 instances in an EMR cluster to access other AWS services such as S3. [Learn more](#)

Field	Action
EC2 key pair	<p>Choose an Amazon EC2 key pair from the list.</p> <p>For more information, see Create an Amazon EC2 Key Pair and PEM File (p. 117).</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 Cluster (p. 446).</p>
IAM user access	<p>Choose No other IAM users.</p> <p> 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 (p. 119).</p>

Field	Action
EC2 instance profile	<p>You can proceed without choosing an instance profile. If you create a cluster without selecting a specific instance profile, one will be created for you.</p> <p>This controls application access to the Amazon EC2 instances in the cluster.</p> <p>For more information, see Configure IAM Roles for Amazon EMR (p. 125).</p>
EMR role	<p>Choose Proceed without role.</p> <p>Allows Amazon EMR to access other AWS services on your behalf.</p> <p>For more information, see Configure IAM Roles for Amazon EMR (p. 125).</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 [Create Bootstrap Actions to Install Additional Software \(Optional\) \(p. 87\)](#).

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, click **Create Cluster**.
12. When the cluster starts, the console displays the **Cluster Details** page.

To launch an HBase cluster using the CLI

- Specify `--hbase` 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 `--instance-type` parameter must be one of the following: `m1.large`, `m1.xlarge`, `c1.xlarge`, `m2.2xlarge`, `m2.4xlarge`, `cc1.4xlarge`, `hi1.4xlarge`, `hs1.8xlarge`, or `cc2.8xlarge`. The `cc2.8xlarge` instance type is only available in the US East (Northern Virginia), US West (Oregon), and EU (Ireland) regions. The `cc1.4xlarge` and `hs1.8xlarge` instance types are only supported in the US East (Northern Virginia) region. The `hi1.4xlarge` instance type is only supported in the US East (Northern Virginia) and EU (Ireland) regions.

The CLI implicitly launches the HBase cluster with keep alive and termination protection set.

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --hbase --name "$USER HBase Cluster" \
--num-instances 2 \
--instance-type cc1.4xlarge
```

- Windows users:

```
ruby elastic-mapreduce --create --hbase --name "$USER HBase Cluster" \
--num-instances 2 --instance-type cc1.4xlarge
```

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 \(p. 446\)](#).
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` flag when requesting a backup. With this flag, 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-ABABABABAB`. This ensures that any future incremental backups reference the correct HBase cluster.

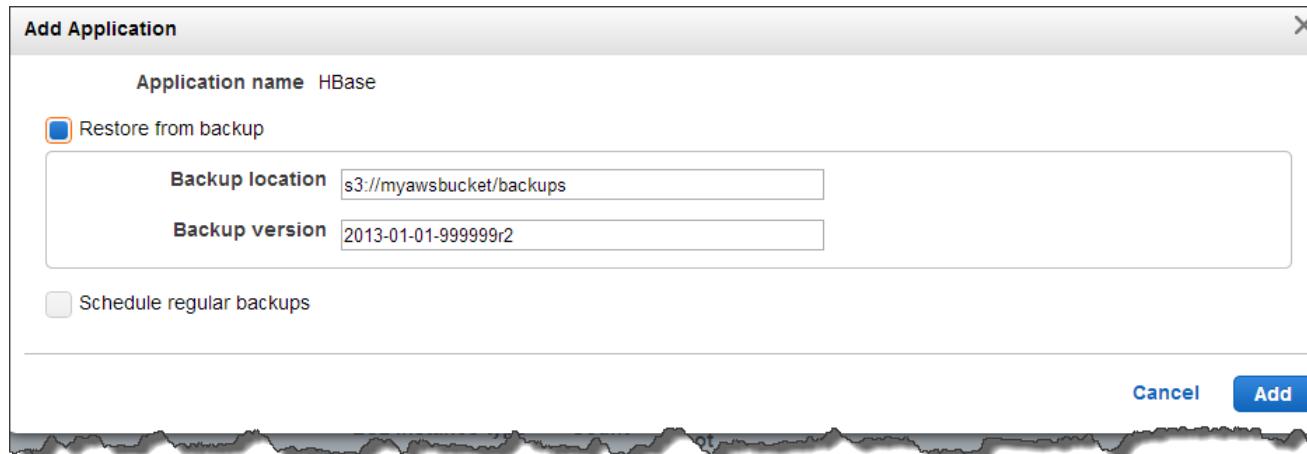
When you are ready to delete old backup files that are no longer needed, we recommend that you first do a full backup of your HBase data. This ensures that all data is preserved and provides a baseline for future incremental backups. Once the full backup is done, you can navigate to the backup location 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 \(p. 355\)](#).

Back Up and Restore HBase Using the Console

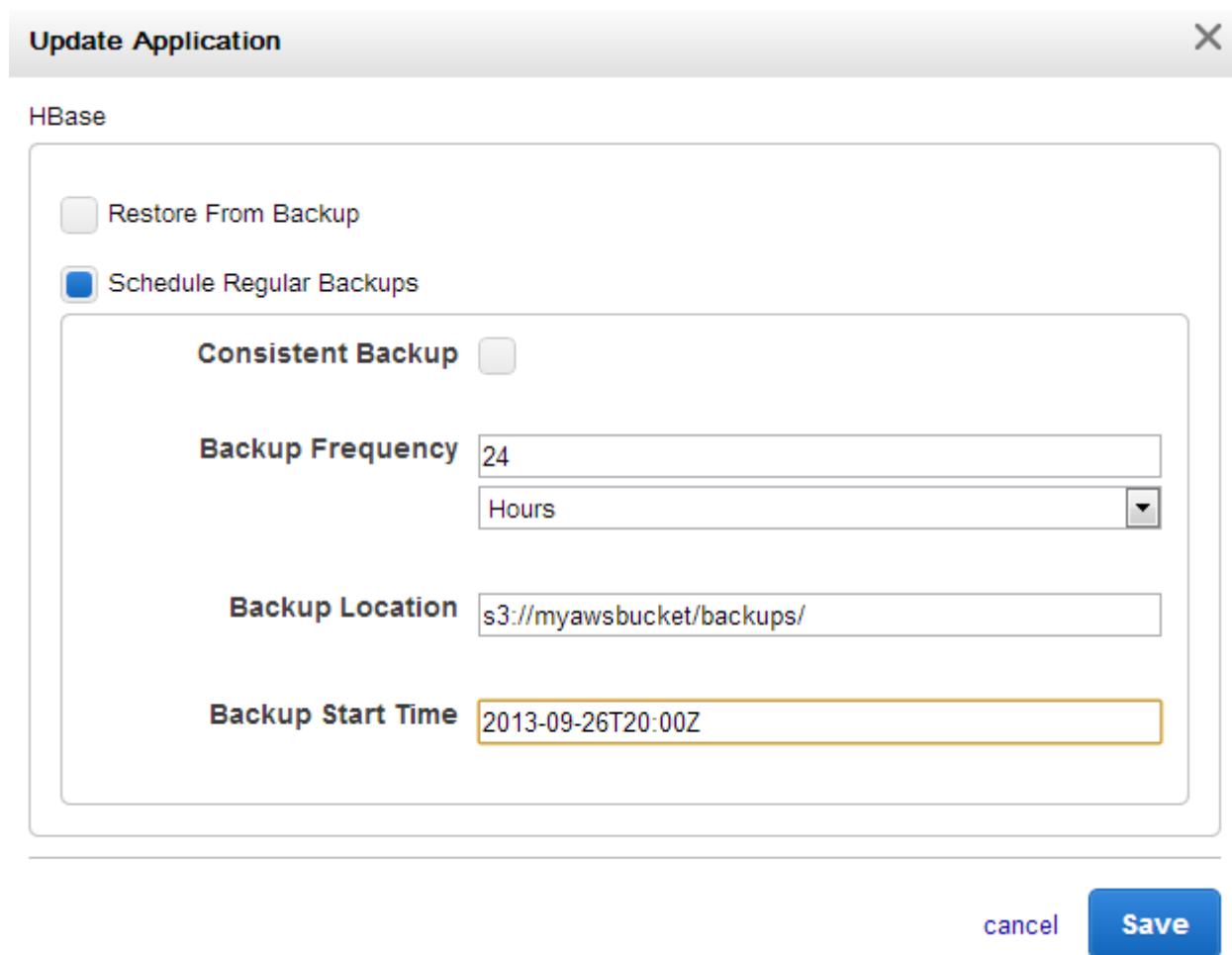
The console provides the ability to launch a new HBase cluster and populate it with data from a previous backup of an HBase cluster. It also gives you the ability to schedule periodic incremental backups of a new HBase cluster. Additional backup and restore functionality, such as the ability to restore data to an already running cluster, do manual backups, and schedule automated full backups is available using the Amazon EMR CLI. For more information, see [Back Up and Restore HBase Using the CLI \(p. 301\)](#)

To populate a new HBase cluster with archived data 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, 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 [Launch an HBase Cluster on Amazon EMR \(p. 291\)](#).
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.
7. Click **Add** and proceed to create the cluster as described in [Plan an Amazon EMR Cluster \(p. 29\)](#).

To schedule automated backups of HBase data using the console



Update Application X

HBase

Restore From Backup

Schedule Regular Backups

Consistent Backup

Backup Frequency 24
Hours

Backup Location

Backup Start Time

cancel **Save**

1. On the **Create Cluster** page, in the **Software Configuration** section, click **Configure**.
2. 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.

3. 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.
4. 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.
5. 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, 2012-06-15T20:00Z, would set the start time to June 15, 2012 at 8pm UTC.
6. Click **Add Application**.
7. Click **Done** and proceed to create the cluster as described in [Plan an Amazon EMR Cluster \(p. 29\)](#).

Back Up and Restore HBase Using the 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. The following table lists all the flags and parameters you can set in order to backup HBase data. Following the table are examples of commands that use these flags and parameters to back up data in various ways.

Parameter	Description
<code>--backup-dir</code>	The directory where a backup exists or should be created.
<code>--backup-version</code>	(Optional) Specifies the version number of an existing backup to restore. If the backup version is not specified in a restore operation, Amazon EMR uses the latest backup, as determined by lexicographical order. This is in the format YYYYMMDDTHHMMSSZ, for example: 20120809T031314Z.
<code>--consistent</code>	(Optional) Pauses all write operations to the HBase cluster during the backup process, to ensure a consistent backup.
<code>--disable-full-backups</code>	Turn off scheduled full backups by passing this flag into a call with <code>--hbase-schedule-backup</code>
<code>--disable-incremental-backups</code>	Turn off scheduled incremental backups by passing this flag into a call with <code>--hbase-schedule-backup</code>
<code>--full-backup-time-interval</code>	An integer that specifies the period of time units to elapse between automated full backups of the HBase cluster. Used with <code>--hbase-schedule-backup</code> this parameter creates regularly scheduled full 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 <code>--full-backup-time-unit</code> .
<code>--full-backup-time-unit</code>	The unit of time to use with <code>--full-backup-time-interval</code> to specify how often automatically scheduled backups should run. This can take any one of the following values: minutes, hours, days.
<code>--hbase-backup</code>	Create a one-time backup of HBase data to the location specified by <code>--backup-dir</code> .

Parameter	Description
--hbase-restore	Restore a backup from the location specified by --backup-dir and (optionally) the version specified by --backup-version.
--hbase-schedule-backup	Schedule an automated backup of HBase data. This can set an incremental backup, a full backup, or both, depending on the flags used to set the intervals and time units. The first backup in the schedule begins immediately unless a value is specified by --start-time.
--incremental-backup-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	The unit of time to use with --incremental-backup-time-interval to specify how often automatically scheduled incremental backups should run. This can take any one of the following values: minutes, hours, days.
--start-time	(Optional) Specifies the time that a 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 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.

The following example backs up the HBase data to s3://myawsbucket/backups, with the timestamp as the version name. This backup does not pause writes to the HBase cluster and as such, may be inconsistent.

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- 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

1. Call `--hbase-schedule-backup` on the HBase cluster and specify the backup time interval and units. If you do not specify a start time, the first backup starts immediately. The following example creates a weekly full backup, with the first backup starting immediately.

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- 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
```

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 \  
--start-time 2012-06-15T20:00:00Z
```

```
--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://mybucket/backups/j-ABABABABABA --start-time 2012-06-15T20:00Z
```

The following example creates a daily incremental backup. The first incremental backup will begin immediately.

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --jobflow j-ABABABABABA \
--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 --incremental-backup-time-interval 24 --incremental- \
--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 -- \
incremental-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://mybucket/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://mybucket/backups/j-ABABABABABA --start-time 2012-06-15T20:00Z
```

2. 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. 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://mybucket/backups/j-ABABABABABA --start-time 2012-06-15T20:00Z --
consistent
```

To turn off automated backups

- Call the cluster with the `--hbase-schedule-backup` parameter and set the `--disable-full-backups` or `--disable-incremental-backups` flag, or both flags. The following example turns off full backups.

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- 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
```

The following example turns 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
```

The following example turns 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 data to a running HBase 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.

The following example restores the HBase cluster to the latest version of backup data stored in s3://myawsbucket/backups, overwriting any data stored in the HBase cluster.

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- 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 HBase cluster with archived data

- 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.

The following example creates a new HBase cluster and loads it with the latest 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
```

- 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-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
```

Terminate an HBase Cluster

Amazon EMR launches HBase clusters 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, see [Terminating a Protected Cluster \(p. 463\)](#).

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 example `elasticmapreduce/hbase-beta/configure-hbase-daemons`.

For a list of regions supported by Amazon EMR see [Choose an AWS Region \(p. 29\)](#). The bootstrap action can only be run when the HBase cluster is launched.

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.

The following example creates a new HBase cluster and uses the `configure-hbase-daemons` bootstrap action to 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, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- 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 single quotes around the value of the `--args` property, 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`.

You can set the following properties with the `configure-hbase-daemons` bootstrap action.

Variable	Description
hbase-master-opts	Options that control how the JVM runs the master daemon. If set, these settings override the default <code>HBASE_MASTER_OPTS</code> variables.
regionserver-opts	Options that control how the JVM runs the region server daemon. If set, these settings override the default <code>HBASE_REGIONSERVER_OPTS</code> variables.
zookeeper-opts	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>.

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 \(p. 29\)](#).

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 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 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, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- 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 the site settings with an XML file

- 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`.

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.

Set the **configure-hbase** bootstrap action when you launch the HBase cluster, and pass in 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://myawsbucket/my-hbase-site.xml`.

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --hbase --name "My HBase Cluster" \
--bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hbase
```

```
\n--args --site-config-file s3://bucket/config.xml
```

- Windows users:

```
ruby elastic-mapreduce --create --hbase --name "My HBase Cluster" --bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hbase --args --site-config-file s3://bucket/config.xml
```

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.

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.

The connection between the Hive and HBase clusters is structured as shown in the following diagram.



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 an interactive Hive cluster. Use Hive version 0.7 or later and AMI version 2.0.4 or later. The following example shows how to launch such a cluster using the Amazon EMR CLI.

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --alive --instance-type m2.4xlarge --num-instances 4 \
--hive-interactive --hive-versions latest
```

- Windows users:

```
ruby elastic-mapreduce --create --alive --instance-type m2.4xlarge --num-instances 4 --hive-interactive --hive-versions latest
```

2. Use SSH to connect to the master node. For more information, see [Connect to the Master Node Using SSH \(p. 446\)](#).
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 locate the public DNS name of the master node using the Amazon EMR console \(p. 447\)](#).

```
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:col1")
  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 [Open an SSH Tunnel to the Master Node \(p. 452\)](#).
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 a tutorial on how to do this, see [Configure FoxyProxy 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 locate the public DNS name of the master node using the Amazon EMR console \(p. 447\)](#).

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. For information about how to connect to the master node using SSH see, [Connect to the Master Node Using SSH \(p. 446\)](#). 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 job-flow 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, as shown in the following example.

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --alive --name "$USER's HBase Test" \
--num-instances 2 \
```

```
--instance-type m1.xlarge \
--bootstrap-action s3://beta.elasticmapreduce/hbase-beta/install-hbase-
stage-1 \
--args s3://beta.elasticmapreduce/hbase-beta \
--bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-
hadoop \
--args -m,dfs.support.append=true \
--log-uri s3://myawsbucket/logfiles
```

- Windows users:

```
ruby elastic-mapreduce --create --alive --name "$USER's HBase Test" --num-
instances 2 --instance-type m1.xlarge --bootstrap-action s3://beta.elast
icmapreduce/hbase-beta/install-hbase-stage-1 --args s3://beta.elasticmapre
duce/hbase-beta --bootstrap-action s3://elasticmapreduce/bootstrap-ac
tions/configure-hadoop --args -m,dfs.support.append=true --log-uri
s3://myawsbucket/logfiles
```

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 \(p. 423\)](#).

Metric	Description
HBaseBackupFailed	Whether the last backup failed. This is set to 0 by default and updated to 1 if the previous backup attempt failed. This metric is only reported for HBase clusters. Use Case: Monitor HBase backups Units: <i>Count</i>
HBaseMostRecentBackupDuration	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. Use Case: Monitor HBase Backups Units: <i>Minutes</i>
HBaseTimeSinceLastSuccessfulBackup	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 \(p. 438\)](#).

You can install Ganglia on an Amazon EMR cluster by calling two bootstrap actions. The first, **install-ganglia**, installs Ganglia. The second, **configure-hbase-for-ganglia**, configures HBase to publish metrics to Ganglia.

Note

You must specify these bootstrap actions when you launch the HBase cluster; Ganglia reporting cannot be added to an HBase cluster that is already running.

Once the HBase cluster has been launched with Ganglia reporting configured, you can use port forwarding to access the Ganglia graphs and reports.

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 an HBase cluster for Ganglia

- Launch the cluster and specify both the **install-ganglia** and **configure-hbase-for-ganglia** bootstrap actions. This is shown in the following example.

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 \(p. 29\)](#).

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --hbase --name "My HBase Cluster" \
--bootstrap-action s3://elasticmapreduce/bootstrap-actions/install-ganglia \
--bootstrap-action s3://region.elasticmapreduce/bootstrap-actions/configure-hbase-for-ganglia
```

- Windows users:

```
ruby elastic-mapreduce --create --hbase --name "My HBase Cluster" --bootstrap-action s3://elasticmapreduce/bootstrap-actions/install-ganglia \
--bootstrap-action s3://region.elasticmapreduce/bootstrap-actions/configure-hbase-for-ganglia
```

To 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 [Open an SSH Tunnel to the Master Node \(p. 452\)](#).
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 a tutorial on how to do this, see [Configure FoxyProxy 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 locate the public DNS name of the master node using the Amazon EMR console \(p. 447\)](#).

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

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 \(p. 315\)](#).

Analyze Amazon Kinesis Data

Starting with AMI 3.0.4, 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](#).

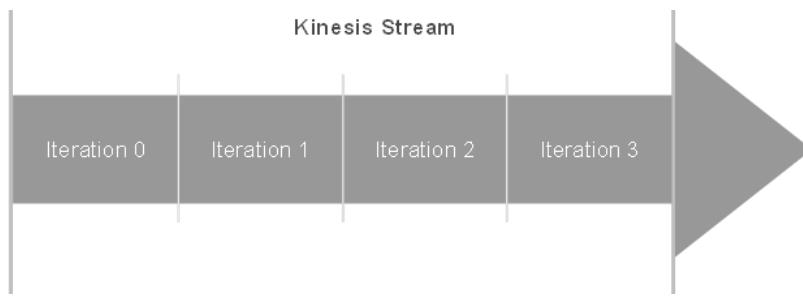
What Can I Do With Amazon EMR and Amazon Kinesis Integration?

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 re-process 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 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.

Tutorial: Analyzing Amazon Kinesis Streams with Amazon EMR and Hive

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

In addition to the console used in this tutorial, Amazon EMR provides a command-line client, a REST-like API set, 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? \(p. 10\)](#).

Note

The Log4J Appender for Amazon Kinesis will currently only work with streams created in US East (Northern Virginia) Region.

Topics

- [Sign Up for the Service \(p. 321\)](#)
- [Create an Amazon Kinesis Stream \(p. 322\)](#)
- [Create an DynamoDB Table \(p. 324\)](#)
- [Download Log4J Appender for Amazon Kinesis Sample Application, Sample Credentials File, and Sample Log File \(p. 325\)](#)
- [Start Amazon Kinesis Publisher Sample Application \(p. 326\)](#)
- [Launch the Cluster \(p. 327\)](#)
- [Run the Ad-hoc Hive Query \(p. 333\)](#)
- [Running Queries with Checkpoints \(p. 336\)](#)
- [Scheduling Scripted Queries \(p. 336\)](#)

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. Go to <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 (Northern Virginia) Region in the region selector.



3. Click **Create Stream**.
4. On the **Create Stream** page, provide a name for your stream (for example, `AccessLogStream`), specify 2 shards, and then click **Create**.

Amazon Kinesis Create Stream

A stream is composed of multiple shards, each of which provides a fixed unit of capacity. The total capacity of the stream is the sum of the capacities of its shards. Each shard corresponds to 1 MB/s of write capacity and 2 MB/s of read capacity. See the [Amazon Kinesis Developer Guide](#) for more information on estimating number of shards needed for your stream. Note that the cost of the stream is also a function of the number of shards. To learn more about the stream, see the [Amazon Kinesis Pricing Page](#)

Stream Name* The Stream Name identifies the stream and is used to access the data written to the stream

Help me decide how many shards I need Use the shard calculator to estimate the number of shards needed for the stream

Number of Shards* You can change the number of shards in the stream without re-creating the stream

Values calculated based on the number of shards entered above:

Read:	Write:
Total Stream Capacity: 4 MB/s	2 MB/s
Max Transactions/second: 10	2000

* Required information

[Cancel](#) [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**.

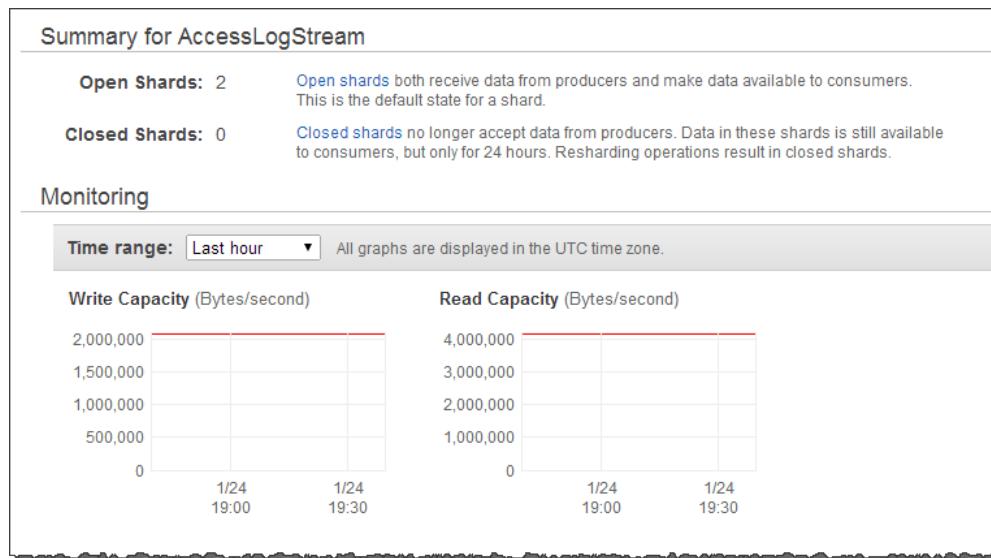
Amazon Kinesis Stream List

[Create Stream](#) [Delete Stream](#)

Filter:

Stream Name	Number of Shards	Status
<input type="checkbox"/> AccessLogStream	2	ACTIVE

5. Click 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 *us-east-1*, 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. Click Continue.
6. You do not need to add any indexes for this tutorial. Click Continue.
7. For Read Capacity Units and Write Capacity Units, use 10 IOPS for each. For more advice on provisioned IOPS, see [Provisioned IOPS Recommendations for Amazon DynamoDB Tables \(p. 320\)](#).

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.appenders.KINESIS.streamName</code>	<code>AccessLogStream</code>	Stream name to which data is to be published.
<code>log4j.appenders.KINESIS.encoding</code>	<code>UTF-8</code>	Encoding used to convert log message strings into bytes before sending to Amazon Kinesis.
<code>log4j.appenders.KINESIS.maxRetries</code>	<code>3</code>	Maximum number of retries when calling Kinesis APIs to publish a log message.
<code>log4j.appenders.KINESIS.backoffInterval</code>	<code>100ms</code>	Milliseconds to wait before a retry attempt.
<code>log4j.appenders.KINESIS.threadCount</code>	<code>20</code>	Number of parallel threads for publishing logs to configured Kinesis stream.
<code>log4j.appenders.KINESIS.bufferSize</code>	<code>2000</code>	Maximum number of outstanding log messages to keep in memory.
<code>log4j.appenders.KINESIS.shutdownTimeout</code>	<code>30</code>	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 double click on 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 Elastic MapReduce 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.

The screenshot shows the 'Cluster Configuration' section of the Amazon EMR 'Create Cluster' page. It includes the following fields:

- Cluster name:** My cluster
- Termination protection:** Yes (radio button selected)
- Logging:** Enabled
- Log folder S3 location:** s3://s3://<bucket-name>/<folder>/
- Debugging:** Enabled

Below these fields is a 'Tags' section with a note: "You can add up to 10 tags to your EMR cluster. A tag consists of a case-sensitive key-value pair." A tooltip for 'Log folder S3 location' explains: "Prevents data loss when the cluster ends. Copy the cluster more easily." A tooltip for 'Debugging' explains: "Index logs (requires more memory)."

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>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 (p. 459). 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>This determines whether Amazon EMR captures detailed log data to Amazon S3.</p> <p>For more information, see View Log Files (p. 418).</p>
Log folder S3 location	<p>Enter an Amazon S3 path to store your debug logs if you enabled logging in the previous field. If the log folder does not exist, the Amazon EMR console creates it for you.</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 (p. 418).</p>

Field	Action
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.

Software Configuration

Hadoop distribution **Amazon** **MapR**

AMI version
3.0.4 (Hadoop 2.2.0)

Applications to be installed

	Version
Hive	0.11.0.2
Pig	0.11.1.1

Additional applications

Field	Action
Hadoop distribution	Choose Amazon . This determines which distribution of Hadoop to run on your cluster. You can choose to run the Amazon distribution of Hadoop or one of several MapR distributions. For more information, see Using the MapR Distribution for Hadoop (p. 149) .
AMI version	Choose 3.0.4 (Hadoop 2.2.0) . The connector for Amazon Kinesis comes with this Amazon EMR AMI version or newer. For more information, see Choose a Machine Image (p. 51) .

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](#).

Hardware Configuration

Specify the networking and hardware configuration for your cluster. If you need more than 2 master nodes, consider Request Spot Instances (unused EC2 capacity) to save money.

EC2 instance type	Count	Request spot
Master	1	<input type="checkbox"/>
Core	2	<input type="checkbox"/>
Task	0	<input type="checkbox"/>

Security and Access

Field	Action
Network	<p>Choose Launch into EC2-Classic.</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) (p. 137).</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, cc1.4xlarge, cg1.4xlarge.</p> <p> This tutorial uses m1.large instances for all nodes.</p> <p> For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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 Lower Costs with Spot Instances (Optional) (p. 37).</p>

Field	Action
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, cc1.4xlarge, cg1.4xlarge.</p> <p>This tutorial uses m1.large instances for all nodes.</p> <p>For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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) (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, cc1.4xlarge, cg1.4xlarge.</p> <p>For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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) (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.

Security and Access

EC2 key pair [Proceed without an EC2 key pair](#) Use an existing key pair to SSH into the master node of the Amazon EC2 cluster as the user "hadoop". [Learn more](#)

IAM user access All other IAM users Control the visibility of this cluster to other IAM users. [Learn more](#)
 No other IAM users

IAM Roles

EMR role [No roles found](#) Allows EMR to access other AWS Services such as EC2 on your behalf. [Learn more](#)
[Create Default Role](#)

EC2 instance profile [Proceed without role](#) Allows EC2 instances in an EMR cluster to access other AWS services such as S3. [Learn more](#)
[Create Default Role](#)

Field	Action
EC2 key pair	<p>Choose an Amazon EC2 key pair from the list.</p> <p>For more information, see Create an Amazon EC2 Key Pair and PEM File (p. 117).</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 Cluster (p. 446).</p>
IAM user access	<p>Choose No other IAM users.</p> <p> 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 (p. 119).</p>
EC2 instance profile	<p>You can proceed without choosing an instance profile. If you create a cluster without selecting a specific instance profile, one will be created for you.</p> <p>This controls application access to the Amazon EC2 instances in the cluster.</p> <p>For more information, see Configure IAM Roles for Amazon EMR (p. 125).</p>
EMR role	<p>Choose Proceed without role.</p> <p>Allows Amazon EMR to access other AWS services on your behalf.</p> <p>For more information, see Configure IAM Roles for Amazon EMR (p. 125).</p>

7. In the **Bootstrap Actions** and **Steps** sections, you do not need to change any of these settings.
8. Review your configuration and if you are satisfied with the settings, click **Create Cluster**.
9. When the cluster starts, the console displays the **Cluster Details** page.

To create a cluster using the Amazon EMR CLI

- In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).
 - Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --alive --name EmrKinesisTutorial --ami-version 3.0.4 --instance-type m1.xlarge \
--hive-interactive --pig-interactive --num-instances 3 --ssh --key-pair example-keypair
```

- Windows users:

```
ruby elastic-mapreduce --create --alive --name EmrKinesisTutorial --ami-version 3.0.4 --instance-type m1.xlarge \
    --hive-interactive --pig-interactive --num-instances 3 --ssh --key-pair example-keypair
```

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 Cluster \(p. 446\)](#).
 2. In the SSH window, from the home directory, start the Hive shell by running the following command:

```
~/bin/hive
```

- 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" = "([^\s]*) ([^\s]*) ([^\s]*) (-|\\"[^\\\"]*\\") ([^\s]*|\\"[^\\\"]*\\") ([0-9]*) ([0-9]*) ([^\s\"]*|\\"[^\\\"]*\\") ([^\s\"]*|\\"[^\\\"]*\\")"
)
STORED BY
'com.amazon.emr.kinesis.hive.KinesisStorageHandler'
TBLPROPERTIES("kinesis.stream.name"="AccessLogStream");
```

This query uses `RegexSerde` to parse the Apache web log format into individual columns. Note how this query specifies the 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.
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.

Configuration Setting	Default Value	Description
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 Kinesis API.
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 \(p. 333\)](#). 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: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;

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 [the section called “Connect to the Master Node Using SSH” \(p. 446\)](#).
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.amazonaws.com/crontab/create_table.q http://emr-kinesis.s3.amazonaws.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/hadoop/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

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

In addition to the console used in this tutorial, Amazon EMR provides a command-line client, a REST-like API set, 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? \(p. 10\)](#).

Note

The Log4J Appender for Amazon Kinesis will currently only work with streams created in US East (Northern Virginia) Region.

Topics

- [Sign Up for the Service \(p. 338\)](#)
- [Create an Amazon Kinesis Stream \(p. 339\)](#)
- [Create an DynamoDB Table \(p. 340\)](#)
- [Download Log4J Appender for Amazon Kinesis Sample Application, Sample Credentials File, and Sample Log File \(p. 341\)](#)
- [Start Amazon Kinesis Publisher Sample Application \(p. 342\)](#)
- [Launch the Cluster \(p. 344\)](#)
- [Run the Pig Script \(p. 349\)](#)
- [Scheduling Scripted Queries \(p. 352\)](#)

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. Go to <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 (Northern Virginia) Region in the region selector.



3. Click **Create Stream**.
4. On the **Create Stream** page, provide a name for your stream (for example, `AccessLogStream`), specify 2 shards, and then click **Create**.

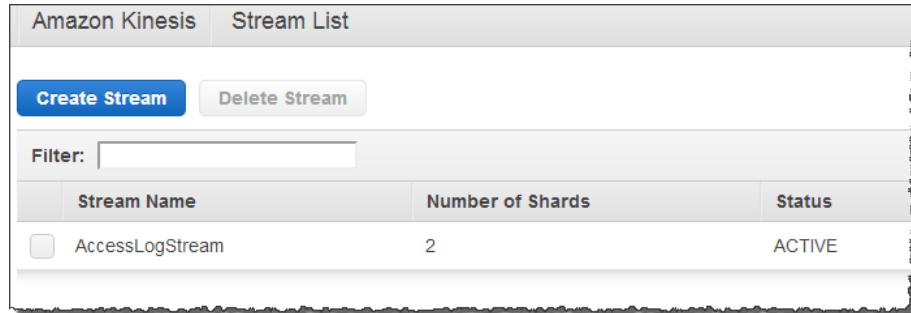
A screenshot of the "Create Stream" page in the Amazon Kinesis console. The page has a header "Amazon Kinesis" and "Create Stream".

A stream is composed of multiple shards, each of which provides a fixed unit of capacity. The total capacity of the stream is the sum of the capacities of its shards. Each shard corresponds to 1 MB/s of write capacity and 2 MB/s of read capacity. See the [Amazon Kinesis Developer Guide](#) for more information on estimating number of shards needed for your stream. Note that the cost of the stream is also a function of number of shards. To learn more about the stream, see the [Amazon Kinesis Pricing Page](#)

Stream Name*	<input type="text" value="AccessLogStream"/>	The Stream Name identifies the stream and is used to access the data written to the stream
<input type="checkbox"/> Help me decide how many shards I need		Use the shard calculator to estimate the number of shards needed for the stream
Number of Shards*	<input type="text" value="2"/>	You can change the number of shards in the stream without re-creating the stream
Values calculated based on the number of shards entered above:		
Read: Total Stream Capacity: 4 MB/s	Write: 2 MB/s	
Max Transactions/second: 10		2000

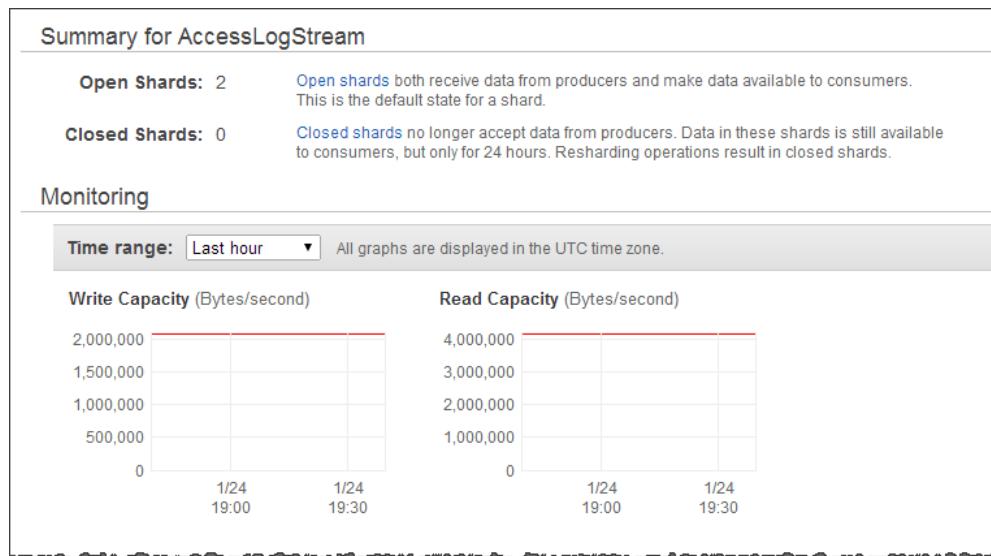
* Required information [Cancel](#) [Create](#)

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.



Stream Name	Number of Shards	Status
AccessLogStream	2	ACTIVE

5. Click 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 *us-east-1*, 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. Click Continue.
6. You do not need to add any indexes for this tutorial. Click Continue.
7. For Read Capacity Units and Write Capacity Units, use 10 IOPS for each. For more advice on provisioned IOPS, see [Provisioned IOPS Recommendations for Amazon DynamoDB Tables \(p. 320\)](#).

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.appenders.KINESIS.streamName</code>	<code>AccessLogStream</code>	Stream name to which data is to be published.
<code>log4j.appenders.KINESIS.encoding</code>	<code>UTF-8</code>	Encoding used to convert log message strings into bytes before sending to Amazon Kinesis.
<code>log4j.appenders.KINESIS.maxRetries</code>	<code>3</code>	Maximum number of retries when calling Kinesis APIs to publish a log message.
<code>log4j.appenders.KINESIS.backoffInterval</code>	<code>100ms</code>	Milliseconds to wait before a retry attempt.
<code>log4j.appenders.KINESIS.threadCount</code>	<code>20</code>	Number of parallel threads for publishing logs to configured Kinesis stream.
<code>log4j.appenders.KINESIS.bufferSize</code>	<code>2000</code>	Maximum number of outstanding log messages to keep in memory.
<code>log4j.appenders.KINESIS.shutdownTimeout</code>	<code>30</code>	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.amazonaws.services.kinesis.log4j.FilePublisher access_log_1
```

- Windows users:

```
 %JAVA_HOME%/bin/java -cp .;kinesis-log4j-appender-1.0.0.jar com.amazonaws.services.kinesis.log4j.FilePublisher access_log_1
```

2. Amazon Kinesis Publisher will upload each row of the log file to Amazon Kinesis until there are no rows remaining.

```
[...]
DEBUG [main] (FilePublisher.java:62) - 39100 records written
DEBUG [main] (FilePublisher.java:62) - 39200 records written
DEBUG [main] (FilePublisher.java:62) - 39300 records written
INFO [main] (FilePublisher.java:66) - Finished publishing 39344 log events
```

```
from access_log_1, took 229 secs to publish
INFO [main] (FilePublisher.java:68) - DO NOT kill this process, publisher
threads will keep on sending buffered logs to Amazon Kinesis
```

Note

The message "INFO [main] (FilePublisher.java:68) - DO NOT kill this process, publisher threads will keep on sending buffered logs to Kinesis" may appear after have published your records to the Amazon Kinesis stream. For the purposes of this tutorial, it is alright to kill this process once you have reached this message.

To Start Amazon Kinesis Publisher for Continuous Publishing on Linux

If you want to simulate continuous publishing to your Amazon Kinesis stream for use with checkpointing scripts, follow these steps on Linux systems to run this shell script that loads the sample logs to your Amazon Kinesis stream every 400 seconds:

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:

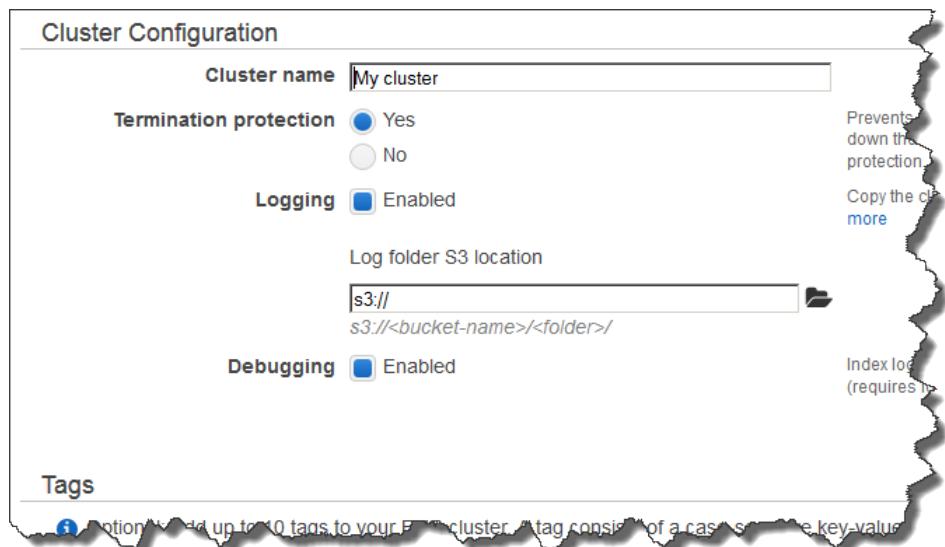
1. Download the file, `publisher.bat`, from <http://emr-kinesis.s3.amazonaws.com/publisher/publisher.bat>:
2. Run `publisher.bat` on the command prompt by typing it and pressing return. You can optionally double click on the file in Windows Explorer.
3. **Important**
This script will run indefinitely. 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 Elastic MapReduce 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	Enter a descriptive name for your cluster. The name is optional, and does not need to be unique.
Termination protection	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 (p. 459) . 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.
Logging	This determines whether Amazon EMR captures detailed log data to Amazon S3. For more information, see View Log Files (p. 418) .

Field	Action
Log folder S3 location	<p>Enter an Amazon S3 path to store your debug logs if you enabled logging in the previous field. If the log folder does not exist, the Amazon EMR console creates it for you.</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 (p. 418).</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.

Software Configuration

Hadoop distribution **Amazon** **MapR** Use Amazon's Hadoop

AMI version **3.0.4 (Hadoop 2.2.0)** Determines the basic software components for your cluster, including Hadoop, Java, and the Amazon Kinesis connector.

Use MapR's Hadoop

Applications to be installed	Version
Hive	0.11.0.2
Pig	0.11.1.1

Additional applications **Select an application** **Configure and add**

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. 149).</p>
AMI version	<p>Choose 3.0.4 (Hadoop 2.2.0).</p> <p>The connector for Amazon Kinesis comes with this Amazon EMR AMI version or newer. For more information, see Choose a Machine Image (p. 51).</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](#).

Hardware Configuration

Specify the networking and hardware configuration for your cluster. 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](#).

EC2 instance type	Count	Request spot	
Master	m1.large	1	<input type="checkbox"/>
Core	m1.large	2	<input type="checkbox"/>
Task	m1.small	0	<input type="checkbox"/>

Field	Action
Network	<p>Choose Launch into EC2-Classic.</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) (p. 137).</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, cc1.4xlarge, cg1.4xlarge.</p> <p> This tutorial uses m1.large instances for all nodes.</p> <p> For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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 Lower Costs with Spot Instances (Optional) (p. 37).</p>

Field	Action
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, cc1.4xlarge, cg1.4xlarge.</p> <p>This tutorial uses m1.large instances for all nodes.</p> <p>For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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) (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, cc1.4xlarge, cg1.4xlarge.</p> <p>For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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) (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.

Security and Access

EC2 key pair [Proceed without an EC2 key pair](#) Use an existing key pair to SSH into the master node of the Amazon EC2 cluster as the user "hadoop". [Learn more](#)

IAM user access All other IAM users Control the visibility of this cluster to other IAM users. [Learn more](#)
 No other IAM users

IAM Roles

EMR role [No roles found](#) Allows EMR to access other AWS Services such as EC2 on your behalf. [Learn more](#)
[Create Default Role](#)

EC2 instance profile [Proceed without role](#) Allows EC2 instances in an EMR cluster to access other AWS services such as S3. [Learn more](#)
[Create Default Role](#)

Field	Action
EC2 key pair	<p>Choose an Amazon EC2 key pair from the list.</p> <p>For more information, see Create an Amazon EC2 Key Pair and PEM File (p. 117).</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 Cluster (p. 446).</p>
IAM user access	<p>Choose No other IAM users.</p> <p> 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 (p. 119).</p>
EC2 instance profile	<p>You can proceed without choosing an instance profile. If you create a cluster without selecting a specific instance profile, one will be created for you.</p> <p>This controls application access to the Amazon EC2 instances in the cluster.</p> <p>For more information, see Configure IAM Roles for Amazon EMR (p. 125).</p>
EMR role	<p>Choose Proceed without role.</p> <p>Allows Amazon EMR to access other AWS services on your behalf.</p> <p>For more information, see Configure IAM Roles for Amazon EMR (p. 125).</p>

7. In the **Bootstrap Actions** and **Steps** sections, you do not need to change any of these settings.
8. Review your configuration and if you are satisfied with the settings, click **Create Cluster**.
9. When the cluster starts, the console displays the **Cluster Details** page.

To create a cluster using the Amazon EMR CLI

- In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).
 - Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --alive --name EmrKinesisTutorial --ami-version 3.0.4 --instance-type m1.xlarge \
--hive-interactive --pig-interactive --num-instances 3 --ssh --key-pair example-keypair
```

- Windows users:

```
ruby elastic-mapreduce --create --alive --name EmrKinesisTutorial --ami-version 3.0.4 --instance-type m1.xlarge \
--hive-interactive --pig-interactive --num-instances 3 --ssh --key-pair example-keypair
```

Run the Pig Script

To run a Pig script over a Amazon Kinesis stream in the Grunt shell

1. Connect to the master node of the cluster using SSH and run the commands shown in the following steps. Your client operating system determines which steps to use to connect to the cluster. For more information, see [Connect to the Cluster \(p. 446\)](#).
2. In the SSH window, from the home directory, start the Grunt shell by running the following command:

```
~/bin/pig
```

3. 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.RegexExtract();

raw_logs = load 'AccessLogStream' using com.amazon.emr.kinesis.pig.KinesisStreamLoader('kinesis.iteration.timeout=1') as (line:chararray);
logs_base =
  -- for each weblog string convert the weblong string into a
  -- structure with named fields
  FOREACH
    raw_logs
  GENERATE
    FLATTEN (
      EXTRACT(
        line,
        '^(\s+)(\s+)(\s+) \\\[(\w:/]+\s[+\-]\d{4})\\] "(.*)" (\s+)
        "([^\"]*)" "([^\"]*)"
      )
    )
  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.
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).

Configuration Setting	Default Value	Description
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 weblong string into a
    -- structure with named fields
FOREACH
    raw_logs
GENERATE
    FLATTEN (
        EXTRACT(
            line,
            '^(\S+) (\S+) (\S+) \\\[(\w:/+\s[+\-]\d{4})\\] "(.+?)" (\S+)
            (\S+) "([^"]*)" "([^"]*)"'
        )
    )
AS (
    host: chararray, IDENTITY: chararray, USER: chararray, TIME: chararray,
    request: chararray, STATUS: INT, SIZE: chararray, referrer: chararray,
    agent: chararray
)

```

```
;  
by_agent_count_raw =  
    -- group by the referrer URL and count the number of requests  
FOREACH  
    (GROUP logs_base BY REGEX_EXTRACT(agent,'.*(Windows|Linux).*',1))  
GENERATE  
    FLATTEN($0),  
    COUNT($1) AS agent_count  
;  
  
by_agent_count = FILTER by_agent_count_raw BY $0 IS NOT null OR ($0 != '');  
  
-- 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 \(p. 446\)](#).
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](#) 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_$iterationNumber' ;
```

Replace `<s3_output_path>` with your Amazon S3 bucket. Save and exit the editor.

6. Create a temporary directory, `/tmp/cronlogs`, for storing the log output generated by the cronjobs:

```
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`.

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 recom-

mend 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*.

Extract, Transform, and Load (ETL) Data with 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 \(p. 355\)](#)
- [Export, Import, Query, and Join Tables in DynamoDB Using Amazon EMR \(p. 364\)](#)
- [Store Avro Data in Amazon S3 Using Amazon EMR \(p. 390\)](#)

Distributed Copy Using S3DistCp

Topics

- [S3DistCp Options \(p. 356\)](#)
- [Adding S3DistCp as a Step in a Cluster \(p. 360\)](#)
- [S3DistCp Versions Supported in Amazon EMR \(p. 364\)](#)

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://ha-doop.apache.org/docs/r1.2.1/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. 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.

If you use AMI version 2.3.5 or newer, you should use the version of S3DistCp located on the cluster node at `/home/hadoop/lib/emr-s3distcp-1.0.jar`. Older AMI versions can use S3DistCp directly from Amazon S3 without copying it to the cluster by replacing the `/home/hadoop/lib/emr-s3distcp-1.0.jar` path with `s3://elasticmapreduce/libs/s3distcp/1.latest/s3distcp.jar`. If you are using an AMI version older than 2.3.5 and IAM roles in your cluster, use the version of S3DistCp at `s3://elasticmapreduce/libs/s3distcp/role/s3distcp.jar` instead of the locations mentioned above. For more information, see [Configure IAM Roles for Amazon EMR \(p. 125\)](#).

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 a file already exists in the destination location, S3DistCp overwrites it. This is true for destinations in both Amazon S3 and HDFS.

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 either the `--arg` or `--args` syntax, examples of which are shown following the table.

Option	Description	Required
<code>--src, LOCATION</code>	<p>Location of the data to copy. This can be either an HDFS or Amazon S3 location.</p> <p>Example: <code>--src,s3://myawsbucket/logs/j-3GY8JC4179IOJ/node</code></p> <p>Important S3DistCp does not support Amazon S3 bucket names that contain the underscore character.</p>	Yes
<code>--dest, LOCATION</code>	<p>Destination for the data. This can be either an HDFS or Amazon S3 location.</p> <p>Example: <code>--dest,hdfs:///output</code></p> <p>Important S3DistCp does not support Amazon S3 bucket names that contain the underscore character.</p>	Yes

Option	Description	Required
--srcPattern, PATTERN	<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
--groupBy, PATTERN	<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
--targetSize, SIZE	<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	Required
--appendToFile	Specifies the behavior of S3DistCp when copying to files already present. It appends new file data to existing files. If you use --appendToFile with --groupBy, new data is appended to files which match the same groups. This option also respects the --targetSize behavior when used with --groupBy.	No
--outputCodec,_CODEC	Specifies the compression codec to use for the copied files. This can take the values: gzip, lzo, snappy, or none. You can use this option, for example, to convert input files compressed with Gzip into output files with LZO compression, or to uncompress the files as part of the copy operation. If you do not specify a value for --outputCodec, the files are copied over with no change in their compression. Example: --outputCodec, lzo	No
--s3ServerSideEncryption	Ensures that the target data is transferred using SSL and automatically encrypted in Amazon S3 using an AWS service-side key. When retrieving data using S3DistCp, the objects are automatically unencrypted. If you attempt to copy an unencrypted object to an encryption-required Amazon S3 bucket, the operation fails. For more information, see Using Data Encryption . Example: --s3ServerSideEncryption	No
--deleteOnSuccess	If the copy operation is successful, this option causes S3DistCp to delete the copied files from the source location. This is useful if you are copying output files, such as log files, from one location to another as a scheduled task, and you don't want to copy the same files twice. Example: --deleteOnSuccess	No
--disableMultipartUpload	Disables the use of multipart upload. For more information about multipart upload, see Configure Multipart Upload for Amazon S3 (p. 104) . Example: --disableMultipartUpload	No
--multipartUploadChunkSize,SIZE	The size, in MiB, of the multipart upload part size. By default, it uses multipart upload when writing to Amazon S3. The default chunk size is 16 MiB. Example: --multipartUploadChunkSize,32	No
--numberFiles	Prepends output files with sequential numbers. The count starts at 0 unless a different value is specified by --startingIndex. Example: --numberFiles	No

Option	Description	Required
--startingIndex, INDEX	Used with --numberFiles to specify the first number in the sequence. Example: --startingIndex,1	No
--outputManifest, FILENAME	Creates a text file, compressed with Gzip, that contains a list of all the files copied by S3DistCp. Example: --outputManifest,manifest-1.gz	No
--previousManifest, PATH	Reads a manifest file that was created during a previous call to S3DistCp using the --outputManifest flag. When the --previousManifest flag is set, S3DistCp excludes the files listed in the manifest from the copy operation. If --outputManifest is specified along with --previousManifest, files listed in the previous manifest also appear in the new manifest file, although the files are not copied. Example: --previousManifest,/usr/bin/manifest-1.gz	No
--requirePreviousManifest	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
--copyFromManifest	Reverses the behavior of --previousManifest to cause S3DistCp to use the specified manifest file as a list of files to copy, instead of a list of files to exclude from copying. Example: --copyFromManifest --previousManifest,/usr/bin/manifest-1.gz	No
--s3Endpoint ENDPOINT	Specifies the Amazon S3 endpoint to use when uploading a file. This option sets the endpoint for both the source and destination. If not set, the default endpoint is s3.amazonaws.com. For a list of the Amazon S3 endpoints, see Regions and Endpoints . Example: --s3Endpoint s3-eu-west-1.amazonaws.com	No
--storageClass CLASS	The storage class to use when the destination is Amazon S3. Valid values are STANDARD and REDUCED_REDUNDANCY. If this option is not specified, S3DistCp tries to preserve the storage class. Example: --storageClass STANDARD	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.

To add a S3DistCp step to a cluster using the CLI

- Add a step to the cluster that calls S3DistCp, passing in the parameters that specify how S3DistCp should perform the copy operation. For more information about adding steps to a cluster, see [Add Steps to a Cluster \(p. 473\)](#).

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 \(p. 356\)](#). 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, run the following from the command line. For more information about the Amazon EMR CLI, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- 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,
S3DistCp-OptionName2,S3DistCp-OptionValue2,
S3DistCp-OptionName3,S3DistCp-OptionValue3'
```

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-3GY8JC4179IOJ --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-3GY8JC4179IOJ/node/' \  
--arg --dest --arg 'hdfs:///output' \  
--arg --srcPattern --arg '.*[a-zA-Z, ]+'
```

- Windows users:

```
ruby elastic-mapreduce --jobflow j-3GY8JC4179IOJ --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-3GY8JC4179IOJ/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-3GY8JC4179IOJ --jar \
/home/hadoop/lib/emr-s3distcp-1.0.jar \
--args '--src,s3://myawsbucket/logs/j-3GY8JC4179IOJ/node/, \
--dest,hdfs:///output, \
--srcPattern,.*daemons.*-hadoop-.*'
```

- Windows users:

```
ruby elastic-mapreduce --jobflow j-3GY8JC4179IOJ --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 \(p. 111\)](#).

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information about the Amazon EMR CLI, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --jobflow j-3GY8JC4179I0K --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'
```

- Windows users:

```
ruby elastic-mapreduce --jobflow j-3GY8JC4179I0K --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
```

S3DistCp Versions Supported in Amazon EMR

Amazon EMR supports the following versions of S3DistCp.

Version	Description	Release Date
1.0.8	Adds the <code>--appendToFile</code> , <code>--requirePreviousManifest</code> , and <code>--storageClass</code> options.	10 April 2013
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 on directly on the clusters. Users should use the JAR in `/home/hadoop/lib` for the latest features.

Export, Import, Query, and Join Tables in DynamoDB Using Amazon EMR

Topics

- [Prerequisites for Integrating Amazon EMR with DynamoDB \(p. 365\)](#)
- [Step 1: Create a Key Pair \(p. 366\)](#)
- [Step 2: Create a Cluster \(p. 366\)](#)
- [Step 3: SSH into the Master Node \(p. 374\)](#)
- [Step 4: Set Up a Hive Table to Run Hive Commands \(p. 376\)](#)
- [Hive Command Examples for Exporting, Importing, and Querying Data in DynamoDB \(p. 381\)](#)
- [Optimizing Performance for Amazon EMR Operations in DynamoDB \(p. 388\)](#)

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 Amazon EMR 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 [Interactive and Batch Hive Clusters \(p. 231\)](#).

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 \(p. 388\)](#) or [Write Percent Setting \(p. 389\)](#) 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](#).
- An DynamoDB table that contains data on the same account used with Amazon EMR.
- A customized version of Hive that includes connectivity to DynamoDB (Hive 0.7.1.3 or later or—if you are using the binary data type—Hive 0.8.1.5 or later). These versions of Hive require the Amazon EMR AMI version 2.0 or later and Hadoop 0.20.205. 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 or from a version of the [Command Line Interface for Amazon EMR](#) downloaded after 11 December 2011. If you launch a cluster using the AWS SDK or the API, you must explicitly set the AMI version to latest and the Hive version to 0.7.1.3 or later. 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 loaded on 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 an SSH client called [PuTTY](#).

- (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 Navigation pane, 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**.

You are prompted to download the key file.

6. Download the private key file and keep it 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 [Getting an SSH Key Pair](#) in the *Amazon Elastic Compute Cloud User Guide*.

Step 2: Create a Cluster

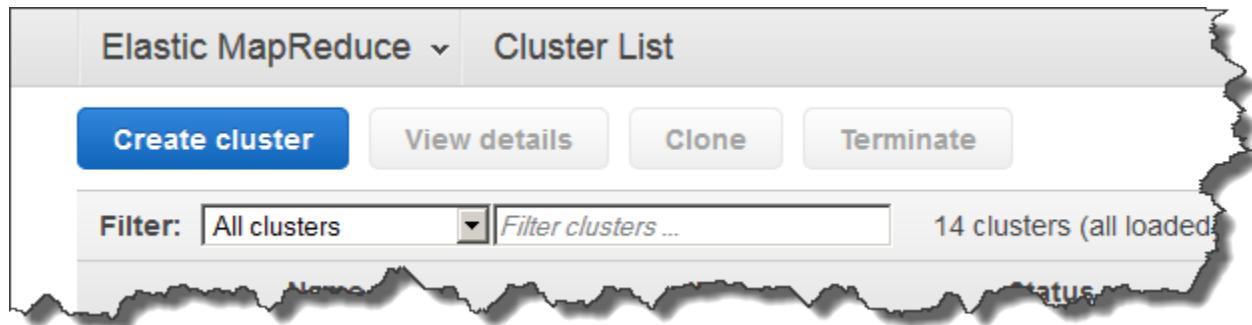
For Hive to run on Amazon EMR, you must create a cluster with Hive enabled. This sets up the necessary applications and infrastructure for Hive to connect to DynamoDB. The following procedures explain how to create an interactive Hive cluster from the AWS Management Console and the CLI.

Topics

- To start a cluster using the AWS Management Console (p. 367)
- To start a cluster using a command line client (p. 373)

To start a cluster using the AWS Management Console

1. Sign in to the AWS Management Console and open the Amazon Elastic MapReduce 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.

Cluster Configuration	
Cluster name	<input type="text" value="My cluster"/>
Termination protection	<input checked="" type="radio"/> Yes <input type="radio"/> No
Logging	<input checked="" type="checkbox"/> Enabled
Log folder S3 location	<input type="text" value="s3://"/> s3://<bucket-name>/<folder>/
Debugging	<input checked="" type="checkbox"/> Enabled
Tags	<p>Info: You can add up to 10 tags to your cluster. A tag consists of a case-sensitive key-value pair.</p>

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.

Software Configuration

Hadoop distribution Amazon Use Amazon

AMI version Determine your cluster's version

MapR Use MapR

Applications to be installed	Version
Hive	0.11.0.1
Pig	0.11.1.1

Additional applications ▼

Configure and add

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>
AMI version	<p>Choose 2.4.2 (Hadoop 1.0.3).</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

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 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](#).

Hardware Configuration

1 Specify the [networking](#) and [hardware](#) configuration for your cluster. If you need more than [Request Spot instances](#) (unused EC2 capacity) to save money.

EC2 instance type	Count	Request spot
Master	1	<input type="checkbox"/>
Core	2	<input type="checkbox"/>
Task	0	<input type="checkbox"/>

Field	Action
Network	Choose Launch into EC2-Classic . Optionally, choose a VPC subnet identifier from the list to launch the cluster in an Amazon VPC. For more information, see Select a Amazon VPC Subnet for the Cluster (Optional) in the <i>Amazon EMR Developer Guide</i> .
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</i> .

Field	Action
Master	<p>Choose m1.small.</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.small (default), m1.large, m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cc1.4xlarge, cg1.4xlarge.</p> <p>This tutorial uses small instances for all nodes due to the light workload and to keep your costs low.</p> <p>For more information, see Instance Groups in the <i>Amazon EMR Developer Guide</i>.</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 Lower Costs with Spot Instances (Optional) in the <i>Amazon EMR Developer Guide</i>.</p>
Core	<p>Choose m1.small.</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.small (default), m1.large, m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cc1.4xlarge, cg1.4xlarge.</p> <p>This tutorial uses small instances for all nodes due to the light workload and to keep your costs low.</p> <p>For more information, see Instance Groups in the <i>Amazon EMR Developer Guide</i>.</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 Lower Costs with Spot Instances (Optional) in the <i>Amazon EMR Developer Guide</i>.</p>
Task	<p>Choose m1.small.</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. Valid types: m1.small (default), m1.large, m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cc1.4xlarge, cg1.4xlarge.</p> <p>For more information, see Instance Groups in the <i>Amazon EMR Developer Guide</i>.</p>

Field	Action
Count	Choose 0 .
Request Spot Instances	Leave this box unchecked. 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> .

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

Security and Access

EC2 key pair Use an existing key pair to SSH into Amazon EC2 cluster as the user "hadoop".

IAM user access All other IAM users No other IAM users Control the visibility of this cluster users. [Learn more](#)

IAM Roles

EMR role Allows EMR to access other AWS services on your behalf. [Learn more](#)

[Create Default Role](#)

EC2 instance profile Allows EC2 instances in an EMR cluster to access AWS services such as S3. [Learn more](#)

[Create Default Role](#)

Field	Action
EC2 key pair	Choose Proceed without an EC2 key pair . Optionally, specify a key pair that you created previously. For more information, see Create SSH Credentials for the Master Node in the <i>Amazon EMR Developer Guide</i> . 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 Cluster in the <i>Amazon EMR Developer Guide</i> .
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 role . 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. 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 [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.

The screenshot shows the 'Cluster Details' page for an Amazon Elastic MapReduce cluster. The cluster is named 'Word count' and is currently 'Starting'. The 'Summary' section shows the cluster's ID (j-3G8ZX3Q9PT433), creation date (2014-04-10 11:12 UTC-7), elapsed time (30 seconds), and auto-terminate setting (Yes). The 'Configuration Details' section shows the AMI version (2.4.2), Hadoop distribution (Amazon 1.0.3), and log URI (s3://.../wordcount/logging/). A sidebar on the left provides links for Monitoring, Steps, and Bootstrap Actions.

To start a cluster using a command line client

1. Download the [Amazon EMR Ruby command line client \(CLI\)](#). If you downloaded the Amazon EMR CLI before 11 December 2011, you will need to download and install the latest version to get support for AMI versioning, Amazon EMR AMI version 2.0, and Hadoop 0.20.205.
2. Install the command line client and set up your credentials. For information about how to do this, go to [Command Line Interface Reference for Amazon EMR](#) in the *Amazon EMR Developer Guide*.
3. Use the following syntax to start a new cluster, specifying your own values for the instance size and your own cluster name for "myJobFlowName":

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information about the Amazon EMR CLI, see [Command Line Interface for Amazon EMR](#) in the *Amazon EMR Developer Guide*.

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --alive --num-instances 3 \
--instance-type m1.small \
--name "myJobFlowName" \
--hive-interactive --hive-versions 0.7.1.1 \
--ami-version latest \
--hadoop-version 0.20.205
```

- Windows users:

```
ruby elastic-mapreduce --create --alive --num-instances 3 --instance-type
m1.small --name "myJobFlowName" --hive-interactive --hive-versions 0.7.1.1
--ami-version latest --hadoop-version 0.20.205
```

You must use the same account to create the Amazon EMR cluster that you used to store data in DynamoDB. This ensures that the credentials passed in by the CLI will match those required by DynamoDB.

Note

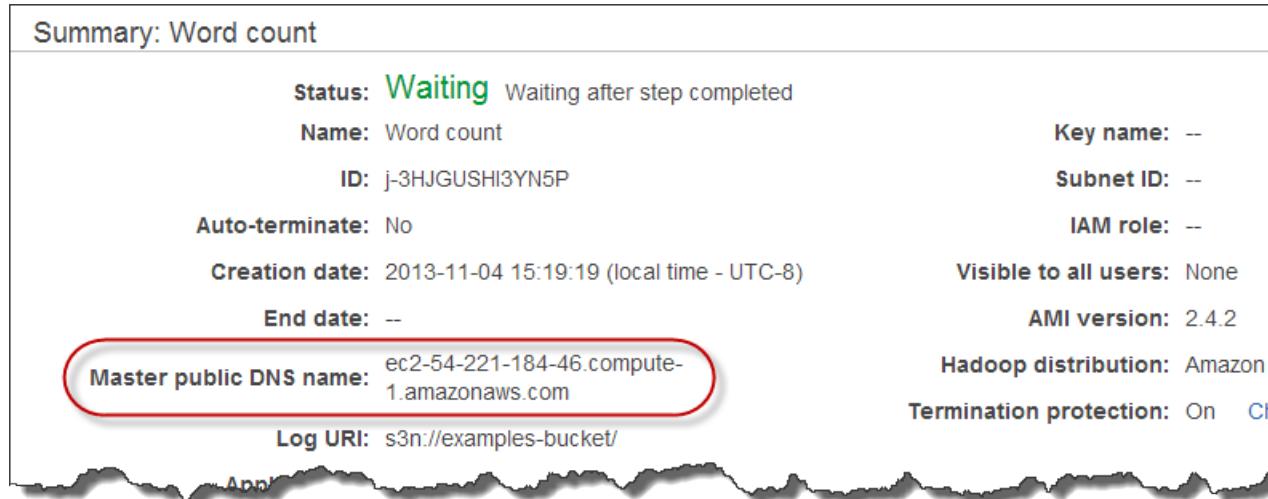
After you create the cluster, you should wait until its status is WAITING before continuing to the next step.

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. Details about the cluster appear in the lower pane.



Summary: Word count

Status:	Waiting Waiting after step completed
Name:	Word count
ID:	j-3HJGUSHI3YN5P
Auto-terminate:	No
Creation date:	2013-11-04 15:19:19 (local time - UTC-8)
End date:	--
Master public DNS name:	ec2-54-221-184-46.compute-1.amazonaws.com
Log URI:	s3n://examples-bucket/
App:	Apache Hadoop
Visible to all users:	None
AMI version:	2.4.2
Hadoop distribution:	Amazon
Termination protection:	On

The DNS name you use to connect to the instance is listed on the Description tab as **Master Public DNS Name**.

To connect to the master node using Linux/Unix/Mac OS X

1. Open a terminal window. This is found at Applications/Utilities/Terminal on Mac OS X and at Applications/Accessories/Terminal on many Linux distributions.
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

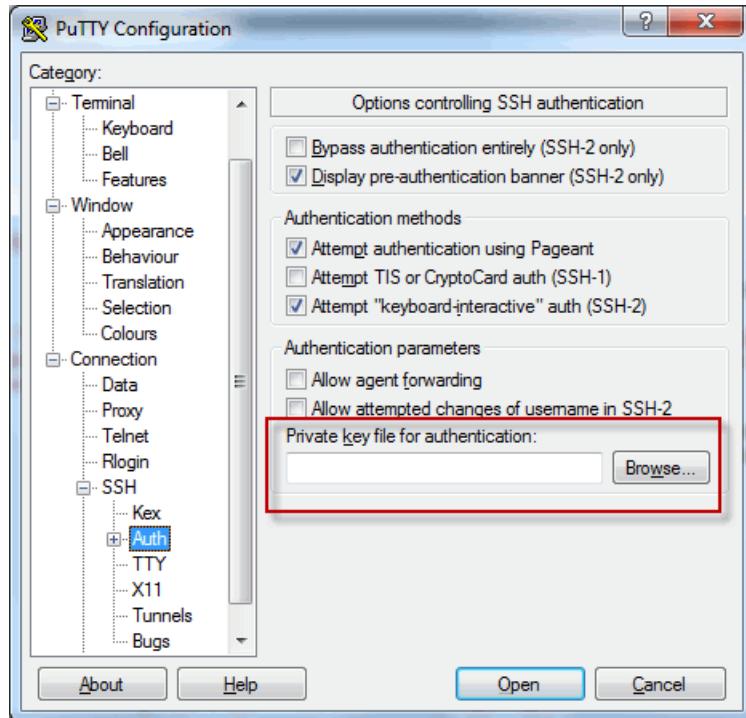
If you are asked to log in, enter `hadoop`.

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.

Step 4: 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. Because we launched the cluster as a Hive application, Amazon EMR installs Hive on the EC2 instances it launches to process the cluster. For more information about Hive, go to <http://hive.apache.org/>.

If you've followed the previous instructions to set up a cluster and use SSH to connect to the master node, you are ready to use Hive interactively.

To run Hive commands interactively

1. At the `hadoop` command prompt for the current master node, type `hive`.
You should see a `hive` prompt: `hive>`
2. 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 an 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_attribute1_name,hive_column2_name:dynamodb_attribute2_name...");
```

When you create a table in Hive from DynamoDB, you must create it as an external table using the keyword `EXTERNAL`. The difference between external and internal tables is that the data in internal tables is deleted when an internal table is dropped. This is not the desired behavior when connected to Amazon DynamoDB, and thus only external tables are supported.

For example, the following Hive command creates a table named `hivetab1` in Hive that references the DynamoDB table named `dynamodbt1`. The DynamoDB table `dynamodbt1` 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 hivetab1 (col1 string, col2 bigint, col3 array<string>)  
STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'  
TBLPROPERTIES ("dynamodb.table.name" = "dynamodbt1",  
"dynamodb.column.mapping" = "col1:name,col2:year,col3:holidays");
```

Line 1 uses the HiveQL `CREATE EXTERNAL TABLE` statement. For `hivetab1`, 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 "hivetab1" 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 `hivetab1`. Queries run against `hivetab1` are internally run against the DynamoDB table `dynamodbt1` of your DynamoDB account, consuming read or write units with each execution.

When you run Hive queries against an 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*.

You can also create a file that contains a series of commands, launch a cluster, and reference that file to perform the operations. For more information, see [Interactive and Batch Modes](#) in the *Amazon EMR Developer Guide*.

To cancel a Hive request

When you execute a Hive query, the initial response from the server includes the command to cancel the request. To cancel the request at any time in the process, use the **Kill Command** from the server response.

1. Enter **Ctrl+C** to exit the command line client.
2. At the shell prompt, enter the **Kill Command** from the initial server response to your request.

Alternatively, you can run the following command from the command line of the master node to kill the Hadoop job, where *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.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>

Hive Options	Description
<i>dynamodb.throughput.write.percent</i>	<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>
<i>dynamodb.endpoint</i>	Specify the endpoint in case you have tables in different regions. For more information about the available DynamoDB endpoints, see Regions and Endpoints .
<i>dynamodb.max.map.tasks</i>	Specify the maximum number of map tasks when reading data from DynamoDB. This value must be equal to or greater than 1.
<i>dynamodb.retry.duration</i>	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-east-1.elasticmapreduce/libs/script-runner/script-runner.jar")
.withArgs("s3://us-east-1.elasticmapreduce/libs/hive/hive-script",
```

```
--base-path", "s3://us-east-1.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 an 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 ar
ray<string>)
STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'
TBLPROPERTIES ("dynamodb.table.name" = "dynamodbt1",
" dynamodb.column.mapping" = "col1:name,col2:year,col3:holidays" );

INSERT OVERWRITE DIRECTORY 's3://bucketname/path/subpath/' SELECT *
FROM hiveTableName;
```

To export an DynamoDB table to an Amazon S3 bucket using formatting

- Create an external table that references a location in Amazon S3. This is shown below as s3_export. During the CREATE call, specify row formatting for the table. Then, when you use INSERT OVERWRITE to export data from DynamoDB to s3_export, the data is written out in the specified format. In the following example, the data is written out as comma-separated values (CSV).

```
CREATE EXTERNAL TABLE hiveTableName (col1 string, col2 bigint, col3 array<string>)
STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'
TBLPROPERTIES ("dynamodb.table.name" = "dynamodbtable1",
"dynamodb.column.mapping" = "col1:name,col2:year,col3:holidays" );

CREATE EXTERNAL TABLE s3_export(a_col string, b_col bigint, c_col array<string>)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LOCATION 's3://bucketname/path/subpath/';

INSERT OVERWRITE TABLE s3_export SELECT *
FROM hiveTableName;
```

To export an 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 2.2.3 and later.

```
CREATE EXTERNAL TABLE hiveTableName (item map<string,string>)
STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'
TBLPROPERTIES ("dynamodb.table.name" = "dynamodbtable1");

CREATE EXTERNAL TABLE s3TableName (item map<string, string>)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n'
LOCATION 's3://bucketname/path/subpath/';

INSERT OVERWRITE TABLE s3TableName SELECT *
FROM hiveTableName;
```

To export an 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 array<string>)
STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'
TBLPROPERTIES ("dynamodb.table.name" = "dynamodtable1",
"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 an 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" = "dynamodtable1",
"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" = "dynamodtable1",
" 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](#) in the *Amazon EMR Developer Guide*. 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" = "dynamodtable1",
"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" = "dynamodtable1");

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" = "dynamodbtb1",  
"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_pur  
chased: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_pur  
chased: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_purchased:Items");  
  
CREATE EXTERNAL TABLE hive_customers(customerId bigint, customerName string,  
    customerAddress array<String>)  
STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'  
TBLPROPERTIES ("dynamodb.table.name" = "Customers",  
    "dynamodb.column.mapping" = "customerId:CustomerId,customerName:Name,customerAddress:Address");  
  
Select c.customerId, c.customerName, count(*) as count from hive_customers  
    c  
JOIN hive_purchases p ON c.customerId=p.customerId  
GROUP BY c.customerId, c.customerName HAVING count > 2;
```

To join two tables from different sources

- In the following example, Customer_S3 is a Hive table that loads a CSV file stored in Amazon S3 and *hive_purchases* is a table that references data in DynamoDB. The following example joins together customer data stored as a CSV file in Amazon S3 with order data stored in DynamoDB to return a set of data that represents orders placed by customers who have "Miller" in their name.

```
CREATE EXTERNAL TABLE hive_purchases(customerId bigint, total_cost double,  
    items_purchased array<String>)  
STORED BY 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'  
TBLPROPERTIES ("dynamodb.table.name" = "Purchases",  
    "dynamodb.column.mapping" = "customerId:CustomerId,total_cost:Cost,items_purchased:Items");  
  
CREATE EXTERNAL TABLE Customer_S3(customerId bigint, customerName string,  
    customerAddress array<String>)  
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','  
LOCATION 's3://bucketname/path/subpath/';  
  
Select c.customerId, c.customerName, c.customerAddress from  
Customer_S3 c  
JOIN hive_purchases p  
ON c.customerId=p.customerId  
where c.customerName like '%Miller%';
```

Note

In the preceding examples, the CREATE TABLE statements were included in each example for clarity and completeness. When running multiple queries or export operations against a given Hive table, you only need to create the table one time, at the beginning of the Hive session.

Optimizing Performance for Amazon EMR Operations in DynamoDB

Amazon EMR operations on an 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 an 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](#) in the *Amazon EMR Developer Guide*. 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
```

For more information about bootstrap actions, see [Using Custom Bootstrap Actions](#) in the *Amazon EMR Developer Guide*.

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.

Viewing Hadoop Logs

If you run into an error, you can investigate what went wrong by viewing the Hadoop logs and user interface. For more information, see [How to Monitor Hadoop on a Master Node](#) and [How to Use the Hadoop User Interface](#) in the *Amazon EMR Developer Guide*.

Store Avro Data in Amazon S3 Using 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 inform-

ation about how to launch a cluster with Pig installed, see [Launch a Pig Cluster \(p. 281\)](#). 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 \(p. 446\)](#).
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 (rexported) 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
bootup file /home/hadoop/.pigbootup 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();
```

Analyze Elastic Load Balancing Log Data

Amazon EMR is ideal for processing large volumes of log files. It is especially useful for services such as Elastic Load Balancing, which can potentially produce large amounts of data in log files stored in Amazon S3. Because you can leverage the built-in connectivity between Amazon EMR and Amazon S3, you can load log data directly into Hadoop jobs running in an Amazon EMR cluster. Furthermore, your cluster can scale with your data; the code and queries used to process 100 GB of log files continue to work as your data grows to 100 TBs, by adding more capacity to your cluster. To learn more about processing Elastic Load Balancing logs in Amazon EMR, proceed to [Tutorial: Query Elastic Load Balancing Access Logs with Amazon Elastic MapReduce \(p. 393\)](#).

Tutorial: Query Elastic Load Balancing Access Logs with Amazon Elastic MapReduce

This tutorial demonstrates how to use Amazon EMR to query and analyze access logs generated by Elastic Load Balancing using Hive. These logs deliver detailed information about all requests made through the Elastic Load Balancing service. For example, access logs contain information such as request IP address, name of the load balancer, time the request was received, and so on. For more information, see [Access Logs](#) in the *Elastic Load Balancing Developer Guide*. Sample log files are provided for HTTP requests. Both log file formats look like the following examples.

HTTP:

```
2014-03-04T02:20:21.212932Z my-elb 192.22.123.169:21029 172.16.93.0:80 0.000066  
0.000651 0.000044 404 404 0 315 "GET http://example.com:80/index2.html HTTP/1.1"  
  
2014-03-04T02:20:21.273241Z my-elb 192.22.127.225:28376 172.16.93.1:80 0.000047  
0.000701 0.000032 200 200 0 1085 "GET http://example.com:80/index.html HTTP/1.1"  
  
2014-03-04T02:20:21.444392Z my-elb 192.22.165.43:13503 172.16.93.1:80 0.000067  
0.00067 0.000053 200 200 0 1085 "GET http://example.com:80/index.html HTTP/1.1"  
  
2014-03-04T02:20:21.977025Z my-elb 192.22.121.162:30815 172.16.93.1:80 0.00007
```

```
0.000867 0.000069 200 200 0 1097 "GET http://example.com:80/sample.html HT  
TP/1.1"
```

TCP:

```
2014-03-04T02:15:43.959433Z my-elb 192.22.81.240:24599 172.16.93.1:80 0.000497  
0.000015 0.000017 - - 200 1362 "-" "  
2014-03-04T02:15:44.001637Z my-elb 192.22.152.221:27889 172.16.93.0:80 0.000574  
0.000015 0.000017 - - 200 1362 "-" "  
2014-03-04T02:15:44.196433Z my-elb 192.22.165.251:49539 172.16.93.1:80 0.00053  
0.000012 0.000017 - - 200 1362 "-" "  
2014-03-04T02:15:44.634838Z my-elb 192.22.32.224:57443 172.16.93.0:80 0.000587  
0.000015 0.000017 - - 200 1362 "-" "  
2014-03-04T02:15:44.667070Z my-elb 192.22.68.165:33606 172.16.93.0:80 0.000506  
0.000015 0.000017 - - 200 1350 "-" "  
2014-03-04T02:15:44.764904Z my-elb 192.22.250.2:32954 172.16.93.0:80 0.000483  
0.000011 0.000018 - - 200 1350 "-" "
```

A sample HiveQL (Hive Query Language) script is provided for you to use with this tutorial, or to download and modify for your own experimentation. For more information about Hive, go to: <https://cwiki.apache.org/confluence/display/Hive/LanguageManual>.

The instructions in this tutorial include how to:

- Sign up for an AWS account
- Create an Amazon EMR cluster, submit the Hive script and process sample log data
- Perform queries on the table created from sample data.

In addition to the console used in this tutorial, Amazon EMR provides a command-line client, a REST-like API set, 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? \(p. 10\)](#). For more information about the Amazon EMR service, see [What is Amazon EMR? \(p. 1\)](#).

Topics

- [Sign Up for the Service \(p. 394\)](#)
- [Launch the Cluster and Run the Script \(p. 395\)](#)
- [Interactively Query the Data in Hive \(p. 403\)](#)
- [Using AWS Data Pipeline to Schedule Access Log Processing \(p. 405\)](#)

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. Go to <http://aws.amazon.com> and click **Sign Up**.
2. Follow the on-screen instructions.

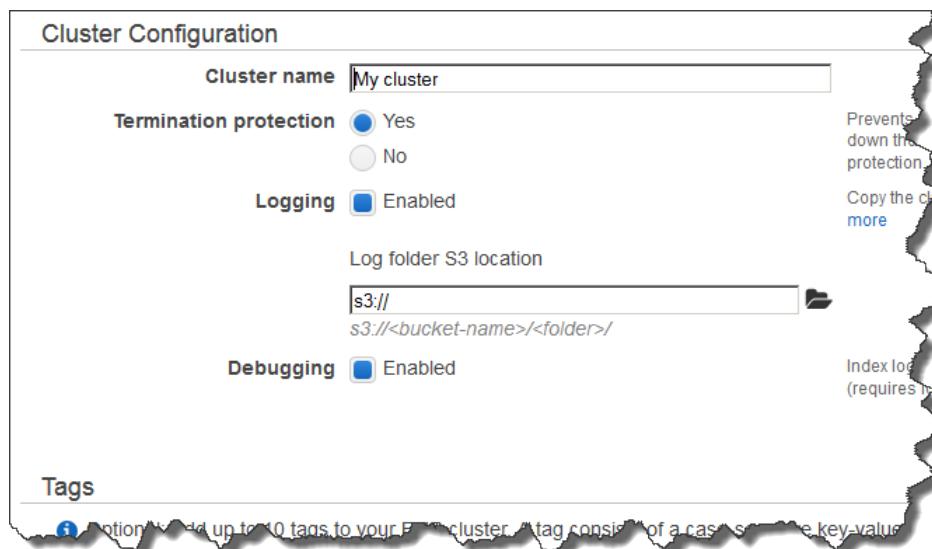
AWS notifies you by email when your account is active and available for you to use. Your AWS account gives you access to all services, but you are charged only for the resources that you use. For this example walk-through, the charges will be minimal.

Launch the Cluster and Run the Script

The next step is to launch the cluster. This tutorial provides the steps to launch the cluster using the Amazon EMR console. 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. You can also launch an Amazon EMR cluster using the CLI. For more information about launching clusters with the CLI, see: [How to Call the Command Line Interface \(p. 585\)](#).

To launch a 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 **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. The name is optional, and does not need to be unique.
Termination protection	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 (p. 459) . 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.
Logging	This determines whether Amazon EMR captures detailed log data to Amazon S3. For more information, see View Log Files (p. 418) .

Field	Action
Log folder S3 location	<p>Enter an Amazon S3 path to store your debug logs if you enabled logging in the previous field. If the log folder does not exist, the Amazon EMR console creates it for you.</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 (p. 418).</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.

Software Configuration

Hadoop distribution **Amazon** **MapR** Use Amazon's Hadoop

AMI version **3.0.4 (Hadoop 2.2.0)** Determines the basic software stack for your cluster, including Hadoop, Java, and Python.

Applications to be installed	Version
Hive	0.11.0.2
Pig	0.11.1.1

Additional applications **Select an application** Configure and add

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. 149).</p>
AMI version	<p>Choose 3.0.4 (Hadoop 2.2.0).</p> <p>For more information, see Choose a Machine Image (p. 51).</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 region. For example, if you have two clusters running in the same region, 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](#).

Hardware Configuration

Specify the networking and hardware configuration for your cluster. If you need more than 25 nodes, Request Spot Instances (unused EC2 capacity) to save money.

EC2 instance type	Count	Request spot
Master	1	<input type="checkbox"/>
Core	2	<input type="checkbox"/>
Task	0	<input type="checkbox"/>

Security and Access

Field	Action
Network	<p>Choose Launch into EC2-Classic.</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) (p. 137).</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: m1.large (default), m1.xlarge, c1.medium, c1.xlarge, m2.xlarge, m2.2xlarge, and m2.4xlarge, cc1.4xlarge, cg1.4xlarge.</p> <p> This tutorial uses m1.large instances for all nodes.</p> <p> For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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 Lower Costs with Spot Instances (Optional) (p. 37).</p>

Field	Action
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, cc1.4xlarge, cg1.4xlarge.</p> <p>This tutorial uses m1.large instances for all nodes.</p> <p>For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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) (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, cc1.4xlarge, cg1.4xlarge.</p> <p>For more information, see Instance Groups (p. 35). For information about mapping legacy clusters to instance groups, see Mapping Legacy Clusters to Instance Groups (p. 470).</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) (p. 37).</p>

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

Security and Access

EC2 key pair Use an existing key pair to SSH into the master node of the Amazon EC2 cluster as the user "hadoop". [Learn more](#)

IAM user access All other IAM users No other IAM users Control the visibility of this cluster to other IAM users. [Learn more](#)

IAM Roles

EMR role Allows EMR to access other AWS Services such as EC2 on your behalf. [Learn more](#)
[Create Default Role](#)

EC2 instance profile Allows EC2 instances in an EMR cluster to access other AWS services such as S3. [Learn more](#)
[Create Default Role](#)

Field	Action
EC2 key pair	<p>Choose an Amazon EC2 key pair from the list.</p> <p>For more information, see Create an Amazon EC2 Key Pair and PEM File (p. 117).</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 Cluster (p. 446).</p>
IAM user access	<p>Choose No other IAM users.</p> <p> 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 (p. 119).</p>
EC2 instance profile	<p>You can proceed without choosing an instance profile. If you create a cluster without selecting a specific instance profile, one will be created for you.</p> <p>This controls application access to the Amazon EC2 instances in the cluster.</p> <p>For more information, see Configure IAM Roles for Amazon EMR (p. 125).</p>
EMR role	<p>Choose Proceed without role.</p> <p>Allows Amazon EMR to access other AWS services on your behalf.</p> <p>For more information, see Configure IAM Roles for Amazon EMR (p. 125).</p>

7. In the **Bootstrap Actions** section, you do not need to change any of these settings. In the **Steps** section, add the step for executing the Hive script. Under **Add Step**, select **Hive program** and click **Configure and add**.
8. In the **Add Step** window, complete the fields according to the following table and click **Add**.

Add Step

Step type: Hive program

Name: S3 location of your Hive script.

Script S3 location*: S3 location of your Hive script.

Input S3 location: S3 location of your Hive input files.

Output S3 location: S3 location of your Hive output files.

Arguments: Specify optional arguments for your script.

Action on failure: What to do if the step fails.

Field	Action
S3 Script location	Put in the location for the sample script: s3://elasticmapreduce/samples/elb-access-logs/elb_access_log.q
Input S3 location	Choose the input location. In this case, you can use s3://elasticmapreduce/samples/elb-access-logs/data.
Output S3 location	Choose the output location to be an Amazon S3 bucket under your account. For example, s3:// yourbucket/folder . If the path that you specified does not exist, the bucket and path are created for you.
Arguments	Do not enter anything.
Action on failure	Choose Continue .

You can also download the script from https://elasticmapreduce.s3.amazonaws.com/samples/elb-access-logs/elb_access_log.q. You can download a [Sample Log File](#)

The `elb_access_log.q` script looks like the following:

```

DROP TABLE elb_raw_access_logs;
CREATE EXTERNAL TABLE elb_raw_access_logs (
  Timestamp STRING,
  ELBName STRING,
  RequestIP STRING,
  RequestPort INT,
  BackendIP STRING,
  BackendPort INT,
  RequestProcessingTime DOUBLE,
  BackendProcessingTime DOUBLE,
  ClientResponseTime DOUBLE,
  ELBResponseCode STRING,
  BackendResponseCode STRING,
  ReceivedBytes BIGINT,
  SentBytes BIGINT,
  RequestVerb STRING,
  
```

```

URL      STRING,
Protocol  STRING
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
    "input.regex" = "([^\n]*)([^\n]*)([^\n]*):([0-9]*)([^\n]*):([0-9]*)
([.0-9]*)([.0-9]*)([.0-9]*)(-[|][0-9]*)(-[|][0-9]*)([-0-9]*)([-0-9]*)
\"([^\n]*)([^\n]*)(-|[^\n]*\")$"
)
LOCATION '${INPUT}';

select * from elb_raw_access_logs;
-- list of requests that resulted into error from backend
select RequestIP, count(RequestIP) from elb_raw_access_logs where
BackendResponseCode<>200 group by RequestIP;
-- list of requests that were made on TCP instead of HTTP
select RequestIP, count(RequestIP) from elb_raw_access_logs where Request
Verb='-' and URL='-' and Protocol='-' group by RequestIP;
-- list of ports on the backend that are being accessed by requests along
with number of times they were requested
select BackendPort, count(BackendPort) from elb_raw_access_logs group by
BackendPort;
-- list of ports other than 80 on the backend that were accessed by requests
over tcp along with number of times they were requested
select BackendPort, count(BackendPort) from elb_raw_access_logs where Re
questVerb='-' and URL='-' and Protocol='-' and BackendPort<>80 group by
BackendPort;

DROP TABLE elb_raw_access_logs_s3;
CREATE TABLE elb_raw_access_logs_s3 (
    Timestamp      STRING,
    ELBName        STRING,
    RequestIP      STRING,
    RequestPort    INT,
    BackendIP      STRING,
    BackendPort    INT,
    RequestProcessingTime  DOUBLE,
    BackendProcessingTime  DOUBLE,
    ClientResponseTime  DOUBLE,
    ELBResponseCode  STRING,
    BackendResponseCode  STRING,
    ReceivedBytes   BIGINT,
    SentBytes       BIGINT,
    RequestVerb     STRING,
    URL            STRING,
    Protocol        STRING
)
PARTITIONED BY(yyyy STRING, mm STRING, dd STRING)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LOCATION '${OUTPUT}/hive-partitioned-logs';

SET YYYY=2014;
SET MM=03;
SET DD=04;

INSERT OVERWRITE TABLE elb_raw_access_logs_s3 partition(yyyy=${hive
conf:YYYY},mm=${hiveconf:MM},dd=${hiveconf:DD})
SELECT * FROM elb_raw_access_logs;

```

This statement creates a table by executing the following `CREATE EXTERNAL TABLE` Hive statement:

```
CREATE EXTERNAL TABLE elb_raw_access_logs (
    Timestamp      STRING,
    ELBName        STRING,
    RequestIP      STRING,
    RequestPort    INT,
    BackendIP      STRING,
    BackendPort    INT,
    RequestProcessingTime  DOUBLE,
    BackendProcessingTime  DOUBLE,
    ClientResponseTime  DOUBLE,
    ELBResponseCode  STRING,
    BackendResponseCode  STRING,
    ReceivedBytes   BIGINT,
    SentBytes       BIGINT,
    RequestVerb     STRING,
    URL            STRING,
    Protocol        STRING
)
```

This statement creates a table using log data stored in Amazon S3. It assigns schema by explicitly delineating column names and their corresponding data types (e.g., `Timestamp STRING`).

```
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
    "input.regex" = "([^\n]*)([^\n]*)([^\n]*):([0-9]*)([^\n]*):([0-9]*)
([.0-9]*)([.0-9]*)([.0-9]*)(-[0-9]*)(-[0-9]*)([-0-9]*)([-0-9]*)
\"([^\n]*)([^\n]*)(-[^\n]*)\"$"
)
```

In this statement, when Hive parses the data, it can read either HTTP or TCP access logs provided by Elastic Load Balancing using a regular expression serializer-deserializer (SERDE) built into Hive. Each field in the regex corresponds to the columns above.

```
LOCATION '${INPUT}';
```

This statement is specifying the location from which Hive reads and parses the table data. Hive and other tools within Amazon EMR provide full interoperability with Amazon S3 and the `${INPUT}` parameter is replaced with the input path you specified in the "Input S3 location" field above.

The following statement demonstrates how you would create a table whose partitions are defined by an Amazon S3 path by selecting data in an existing table (for example, `elb_raw_access_logs`):

```
CREATE TABLE elb_raw_access_logs_s3 (
    Timestamp      STRING,
    ELBName        STRING,
    RequestIP      STRING,
    RequestPort    INT,
    BackendIP      STRING,
    BackendPort    INT,
    RequestProcessingTime  DOUBLE,
    BackendProcessingTime  DOUBLE,
    ClientResponseTime  DOUBLE,
    ELBResponseCode  STRING,
```

```
BackendResponseCode      STRING,  
ReceivedBytes           BIGINT,  
SentBytes               BIGINT,  
RequestVerb             STRING,  
URL                     STRING,  
Protocol                STRING  
)  
  
PARTITIONED BY(yyyy STRING, mm STRING, dd STRING)  
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','  
LOCATION '${OUTPUT}/hive-partitioned-logs';  
  
SET YYYY=2014;  
SET MM=03;  
SET DD=04;  
  
INSERT OVERWRITE TABLE elb_raw_access_logs_s3 partition(yyyy=${hive  
conf:YYYY},mm=${hiveconf:MM},dd=${hiveconf:DD})  
SELECT * FROM elb_raw_access_logs;
```

For more information on the queries included in the script, see [Interactively Query the Data in Hive \(p. 403\)](#).

9. In the **Auto-terminate** section, choose **No** to keep the cluster running when the script step has successfully completed.
10. Review your configuration and if you are satisfied with the settings, click **Create Cluster**.
11. When the cluster starts, the console displays the **Cluster Details** page.
12. When the Hive program step completes successfully, you can check the output of the script by examining the contents of the Amazon S3 output bucket that you specified in [Step 8 \(p. 399\)](#). The contents of the `elb_raw_access_logs_s3` table should be stored in that bucket following the schema provided.
13. Terminate your cluster or you can optionally proceed to [Interactively Query the Data in Hive \(p. 403\)](#).

Interactively Query the Data in Hive

You may want to read in data and prototype Hive queries interactively before committing them to a script. The following tutorial shows you how to query the table that you already created by running the `elb_access_log.q` script.

To use the Hive shell for interactive queries

1. Log in to your cluster by using SSH to connect to the master node. For more information, see [Connect to the Master Node Using SSH \(p. 446\)](#).

Note

To log in to the master node, you must have selected an Amazon EC2 key pair when creating the cluster.

2. Type `hive` at the master node shell prompt to launch the Hive shell.
3. Use the following command to verify that the `elb_raw_access_logs_s3` and `elb_raw_access_logs` tables exist:

```
hive> SHOW TABLES;  
OK
```

```
elb_raw_access_logs
elb_raw_access_logs_s3
Time taken: 4.151 seconds, Fetched: 2 row(s)
```

4. Try some interactive sample queries.

This statement selects the first ten elements from the table `elb_raw_access_logs`.

```
hive> SELECT * FROM elb_raw_access_logs LIMIT 10;
OK
2014-03-04T02:20:28.947447Z my-elb 192.22.198.254 56657 172.16.93.1 80 5.0E-
5 7.43E-4 3.3E-5 200 200 0 1080 GET http://example.com:80/index.html HTTP/1.1
2014-03-04T02:20:29.090278Z my-elb 192.22.70.68 40378 172.16.93.0 80 3.4E-
5 6.85E-4 3.4E-5 200 200 0 1092 GET http://example.com:80/sample.html HTTP/1.1
2014-03-04T02:20:29.390016Z my-elb 192.22.208.34 6430 172.16.93.1 80 5.1E-
5 7.02E-4 3.3E-5 200 200 0 1092 GET http://example.com:80/sample.html HTTP/1.1
2014-03-04T02:20:29.866223Z my-elb 192.22.143.211 42402 172.16.93.1 80 6.9E-
5 9.9E-4 5.1E-5 200 200 0 1080 GET http://example.com:80/index.html HTTP/1.1
2014-03-04T02:20:30.192116Z my-elb 192.22.126.198 57548 172.16.93.0 80 5.5E-
5 8.12E-4 3.5E-5 200 200 0 1092 GET http://example.com:80/sample.html HTTP/1.1
2014-03-04T02:20:30.202754Z my-elb 192.22.77.50 11990 172.16.93.0 80 3.8E-
5 6.58E-4 3.3E-5 200 200 0 1080 GET http://example.com:80/index.html HTTP/1.1
2014-03-04T02:20:30.260543Z my-elb 192.22.42.185 24558 172.16.93.1 80 7.9E-
5 7.39E-4 8.3E-5 200 200 0 1092 GET http://example.com:80/sample.html HTTP/1.1
2014-03-04T02:20:30.356423Z my-elb 192.22.200.216 26638 172.16.93.1 80 5.9E-
5 6.75E-4 3.5E-5 200 200 0 1092 GET http://example.com:80/sample.html HTTP/1.1
2014-03-04T02:20:30.748524Z my-elb 192.22.214.99 49139 172.16.93.1 80 6.1E-
5 0.001032 3.6E-5 200 200 0 1092 GET http://example.com:80/sample.html HT
TP/1.1
2014-03-04T02:20:30.839575Z my-elb 192.22.167.125 49832 172.16.93.0 80 5.0E-
5 9.27E-4 3.1E-5 200 200 0 1080 GET http://example.com:80/index.html HTTP/1.1
Time taken: 0.855 seconds, Fetched: 10 row(s)
```

This statement lists the first ten IP addresses whose requests returned an error along with an error count.

```
hive> SELECT RequestIP, COUNT(RequestIP) FROM elb_raw_access_logs WHERE
BackendResponseCode<>200 GROUP BY RequestIP LIMIT 10;
<snip>
OK
192.22.0.171 1
192.22.0.251 1
192.22.100.114 1
192.22.100.255 1
192.22.102.26 1
192.22.103.7 1
192.22.104.196 1
192.22.106.163 1
192.22.108.18 1
192.22.109.199 1
Time taken: 32.991 seconds, Fetched: 10 row(s)
```

This statement lists the backend ports that received requests and counts for each. Because the sample data is only HTTP, the only port is 80.

```
hive> SELECT BackendPort, COUNT(BackendPort) FROM elb_raw_access_logs GROUP
    BY BackendPort LIMIT 10;
<snip>
OK
80 5122
Time taken: 32.1 seconds, Fetched: 1 row(s)
```

This statement lists the URLs that gave a response status other than 200, the response code, the backend IP, and the count for each.

```
hive> SELECT COUNT(*) AS CountOfNon200 , BackendIP, BackendResponseCode ,
    URL FROM elb_raw_access_logs WHERE BackendResponseCode <> 200 GROUP BY
    BackendIP , BackendResponseCode, URL ORDER BY CountOfNon200 DESC;
<snip>
OK
58 172.16.93.0 404 http://example.com:80/index2.html
52 172.16.93.1 404 http://example.com:80/index2.html
10 172.16.93.0 304 http://example.com:80/sample.html
10 172.16.93.0 304 http://example.com:80/index.html
5 172.16.93.1 304 http://example.com:80/sample.html
2 172.16.93.1 304 http://example.com:80/index.html
Time taken: 93.57 seconds, Fetched: 6 row(s)
```

5. Exit the Hive shell and the master node shell by pressing Control+D twice.
6. **Important**
Remember to terminate your cluster to avoid additional charges.

Using AWS Data Pipeline to Schedule Access Log Processing

If you are routinely running a Hive script or performing other processing with Amazon EMR, it is advantageous to schedule this process. This type of ETL (Extract Transfer and Load) work is ideally handled by AWS Data Pipeline, which can orchestrate the processing and movement of data between many different AWS services.

The tutorial, [Getting Started: Processing Access Logs with Amazon EMR and Hive](#), demonstrates how you would process web server access logs using AWS Data Pipeline. However, you can adapt that pipeline in the tutorial for use with Elastic Load Balancing access logs by making a few modifications. Use the Hive [script](#), `elb_access_log.q`, and input data provided in this tutorial.

For more information about how to use AWS Data Pipeline, see the [Getting Started: Processing Access Logs with Amazon EMR and Hive](#) tutorial and the [AWS Data Pipeline Developer Guide](#).

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 \(p. 406\)](#)
- [Connect to the Cluster \(p. 446\)](#)
- [Control Cluster Termination \(p. 458\)](#)
- [Resize a Running Cluster \(p. 464\)](#)
- [Cloning a Cluster Using the Console \(p. 472\)](#)
- [Add Steps to a Cluster \(p. 473\)](#)
- [Associate an Elastic IP Address with a Cluster \(p. 477\)](#)
- [Automate Recurring Clusters with AWS Data Pipeline \(p. 480\)](#)

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 \(p. 407\)](#)
- [View Web Interfaces on the Cluster \(Hadoop 2.x\) \(p. 411\)](#)
- [View Log Files \(p. 418\)](#)
- [View Cluster Instances in Amazon EC2 \(p. 422\)](#)
- [Monitor Metrics with CloudWatch \(p. 423\)](#)
- [Logging Amazon Elastic MapReduce API Calls in AWS CloudTrail \(p. 436\)](#)
- [Monitor Performance with Ganglia \(p. 438\)](#)

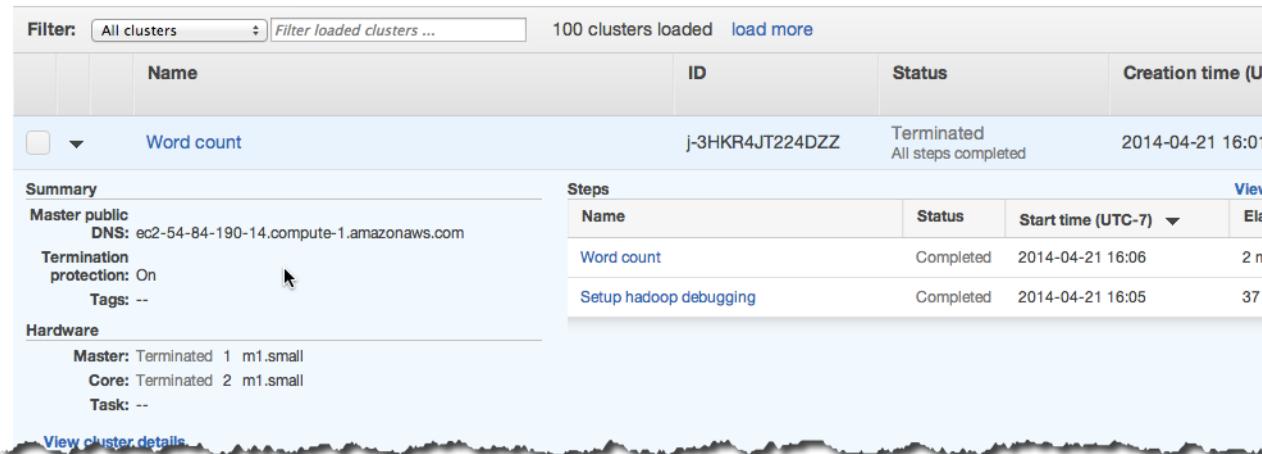
View Cluster Details

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 the details of a cluster using the console

1. Sign in to the AWS Management Console and open the Amazon Elastic MapReduce 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 Cluster List page. At the top, there is a filter bar with 'All clusters' selected and a 'Filter loaded clusters ...' dropdown. Below the filter is a message '100 clusters loaded' and a 'load more' button. The main table has columns for Name, ID, Status, and Creation time (U). A cluster named 'Word count' is selected, indicated by a dropdown arrow icon. The expanded row shows the following details:

Summary		Steps	
Name	ID	Name	Status
Word count	j-3HKR4JT224DZZ	Word count	Completed
Master public		Setup hadoop debugging	Completed
DNS: ec2-54-84-190-14.compute-1.amazonaws.com		Start time (UTC-7)	2014-04-21 16:05
Termination protection: On			
Tags: --			
Hardware			
Master: Terminated 1 m1.small			
Core: Terminated 2 m1.small			
Task: --			

At the bottom of the expanded row, there is a link 'View cluster details'.

3. For more details, select the cluster to view by clicking its Name. The **Summary** pane appears, providing 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 EMR
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			

To view cluster details from the CLI, use the `--list` parameter to list clusters. This section presents some of these variations.

To list clusters created in the last two days using the CLI

- 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, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- 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-3GJ4FRRNKGY97	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.

You can get information about a cluster using the `--describe` parameter and specifying a cluster ID.

To retrieve information about a cluster

- 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
```

The response looks similar to the following:

```
{  
  "JobFlows": [  
    {  
      "Name": "Development Job Flow (requires manual termination)",  
      "LogUri": "s3n://AKIAIOSFODNN7EXAMPLE/FileName//",  
      "ExecutionStatusDetail": {  
        "StartTime": null,  
        "EndTime": null,  
        "LastStateChangeReason": "Starting instances",  
        "CreationDateTime": DateTimeStamp,  
        "State": "STARTING",  
        "ReadyDateTime": null  
      },  
      "Steps": [],  
      "Instances": {  
        "MasterInstanceId": null,  
        "Ec2KeyName": "KeyName",  
        "NormalizedInstanceHours": 0,  
        "InstanceCount": 5,  
        "Placement": {  
          "AvailabilityZone": "us-east-1a"  
        },  
        "SlaveInstanceType": "m1.small",  
        "HadoopVersion": "0.20",  
        "MasterPublicDnsName": null,  
        "KeepJobFlowAliveWhenNoSteps": true,  
        "InstanceGroups": [  
          {  
            "StartDateTime": null,  
            "SpotPrice": null,  
            "Name": "Master Instance Group",  
            "InstanceRole": "MASTER",  
            "EndDateTime": null,  
            "LastStateChangeReason": "",  
            "CreationDateTime": DateTimeStamp,  
            "LaunchGroup": null,  
            "InstanceGroupId": "InstanceGroupID",  
            "State": "PROVISIONING",  
            "Market": "ON_DEMAND",  
            "ReadyDateTime": null,  
            "InstanceType": "m1.small",  
            "InstanceRunningCount": 0,  
            "InstanceRequestCount": 1  
          },  
          {  
            "StartDateTime": null,  
            "SpotPrice": null,  
            "Name": "Slave Instance Group",  
            "InstanceRole": "SLAVE",  
            "EndDateTime": null,  
            "LastStateChangeReason": "",  
            "CreationDateTime": DateTimeStamp,  
            "LaunchGroup": null,  
            "InstanceGroupId": "InstanceGroupID",  
            "State": "PROVISIONING",  
            "Market": "ON_DEMAND",  
            "ReadyDateTime": null,  
            "InstanceType": "m1.small",  
            "InstanceRunningCount": 0,  
            "InstanceRequestCount": 1  
          }  
        ]  
      }  
    }  
  ]  
}
```

```

        "Name": "Task Instance Group",
        "InstanceRole": "TASK",
        "EndDateTime": null,
        "LastStateChangeReason": "",
        "CreationDateTime": DateTimeStamp,
        "LaunchGroup": null,
        "InstanceGroupId": "InstanceGroupID",
        "State": "PROVISIONING",
        "Market": "ON_DEMAND",
        "ReadyDateTime": null,
        "InstanceType": "m1.small",
        "InstanceRunningCount": 0,
        "InstanceRequestCount": 2
    },
    {
        "StartDateTime": null,
        "SpotPrice": null,
        "Name": "Core Instance Group",
        "InstanceRole": "CORE",
        "EndDateTime": null,
        "LastStateChangeReason": "",
        "CreationDateTime": DateTimeStamp,
        "LaunchGroup": null,
        "InstanceGroupId": "InstanceGroupID",
        "State": "PROVISIONING",
        "Market": "ON_DEMAND",
        "ReadyDateTime": null,
        "InstanceType": "m1.small",
        "InstanceRunningCount": 0,
        "InstanceRequestCount": 2
    }
],
"MasterInstanceType": "m1.small"
},
"bootstrapActions": [],
"JobFlowId": "JobFlowID"
}
]
}

```

For more information about the input parameters unique to `DescribeJobFlows`, see [DescribeJobFlows](#).

View Web Interfaces on the Cluster (Hadoop 2.x)

Hadoop publishes web interfaces that display status about the cluster. These web interfaces are hosted on the master node of the cluster. You can view these web interfaces by using SSH to create a tunnel to the master node and configuring a SOCKS proxy so your browser can view websites hosted on the master node using the SSH tunnel. For more information, see [Connect to the Cluster \(p. 446\)](#).

The web user interfaces for Hadoop are located at the URLs in the following table, where `master-public-dns-name` is the public DNS name of the master node and `slave-public-dns-name` is the public DNS name of a core or task node in the cluster. For information about how to locate the public DNS name of the master node for a cluster, see [Connect to the Master Node Using SSH \(p. 446\)](#).

Name of Interface	URL
ResourceManager	http://<i>master-public-dns-name</i>:9026

Name of Interface	URL
NameNode	http://<i>master-public-dns-name</i>:9101/
NodeManager	http://<i>slave-public-dns-name</i>:9035/

To relocate these web interfaces, edit `conf/hdfs-site.xml`.

To view the ResourceManager web interface

- To access the ResourceManager web interface running on the master node, open [http://*master-public-dns-name*:9026/](http://<i>master-public-dns-name</i>:9026/). This web interface is hosted on the master node of the cluster. You can access the web interface by using SSH to create a tunnel to the master node and configuring a SOCKS proxy so your browser can view websites hosted on the master node. For more information, see [Connect to the Cluster \(p. 446\)](#).

To view the HDFS NameNode web interface

- To access the HDFS NameNode web interface running on the master node, open [http://*master-public-dns-name*:9101/](http://<i>master-public-dns-name</i>:9101/). This web interface is hosted on the master node of the cluster. You can access the web interface by using SSH to create a tunnel to the master node and configuring a SOCKS proxy so your browser can view websites hosted on the master node. For more information, see [Connect to the Cluster \(p. 446\)](#).

NameNode 'ip-XXXXXX.ec2.internal:9000' (active)

Started:	Tue Oct 29 20:38:10 UTC 2013		
Version:	2.2.0, 2f86911a07c659a94f573e641ad324a98824f4f2		
Compiled:	2013-10-25T20:21Z by Elastic MapReduce from (detached from 2f86911)		
Cluster ID:	CID-b84a7a22-70b7-4279-a119-c9a0fc9f6fa9		
Block Pool ID:	BP-884063330-10.29.185.209-1383079074264		

[Browse the filesystem](#)
[NameNode Logs](#)

Cluster Summary

Security is **OFF**
7 files and directories, 0 blocks = 7 total.
Heap Memory used 125.74 MB is 59% of Committed Heap Memory 212.50 MB. Max Heap Memory is 889 MB.
Non Heap Memory used 30.31 MB is 96% of Committed Non Heap Memory 31.44 MB. Max Non Heap Memory is 130 MB.

Configured Capacity	1.64 TB
DFS Used	16 KB
Non DFS Used	0 B
DFS Remaining	1.64 TB
DFS Used%	0.00%
DFS Remaining%	100.00%
Block Pool Used	16 KB
Block Pool Used%	0.00%
DataNodes usages	Min % Median % Max % stdev %
	0.00% 0.00% 0.00% 0.00%
Live Nodes	2 (Decommissioned: 0)
Dead Nodes	0 (Decommissioned: 0)
Decommissioning Nodes	0
Number of Under-Replicated Blocks	0

View Web Interfaces on the Cluster (Hadoop 1.x)

Hadoop publishes web interfaces that display status about the cluster. These web interfaces are hosted on the master node of the cluster. You can view these web interfaces by using SSH to create a tunnel to the master node and configuring a SOCKS proxy so your browser can view websites hosted on the master node using the SSH tunnel. For more information, see [Connect to the Cluster \(p. 446\)](#).

The web user interfaces for Hadoop are located at the URLs in the following table, where *master-public-dns-name* is the public DNS name of the master node of the cluster. For information about how to locate the public DNS name of the master node for a cluster, see [Connect to the Master Node Using SSH \(p. 446\)](#).

Name of Interface	URL
JobTracker	http://<i>master-public-dns-name</i>:9100/
HDFS NameNode	http://<i>master-public-dns-name</i>:9101/
TaskTracker	http://<i>slave-public-dns-name</i>:9103/

To relocate these web interfaces, edit `conf/hadoop-default.xml`.

To view the JobTracker web interface

1. To access the JobTracker web interface running on the master node, open <http://master-public-dns-name:9100/>. This web interface is hosted on the master node of the cluster. You can access the web interface by using SSH to create a tunnel to the master node and configuring a SOCKS proxy so your browser can view websites hosted on the master node. For more information, see [Connect to the Cluster \(p. 446\)](#).

domU-12-31-39-00-ED-78 Hadoop Map/Reduce Administration

State: RUNNING
Started: Tue Jan 20 07:59:50 UTC 2009
Version: 0.18.1
Compiled: Mon Jan 5 21:29:34 UTC 2009 by root
Identifier: 200901200759

Cluster Summary

Maps	Reduces	Total Submissions	Nodes	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node
0	0	2	2	4	4	4.00

Running Jobs

Running Jobs	
none	

Completed Jobs

Completed Jobs								
Jobid	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Reduces Completed
job_200901200759_0001	hadoop	averaging pass	100.00%	11	11	100.00%	1	1
job_200901200759_0002	hadoop	job jar	100.00%	2	2	100.00%	1	1

Failed Jobs

The **Cluster Summary** shows that there were two slave nodes in the cluster and that each performed four tasks. The **Completed Jobs** section shows that the map and reduce clusters are 100% complete.

2. Click a cluster ID.

Hadoop displays information about the selected cluster.

Hadoop job_200901200759_0001 on domU-12-31-39-00-ED-78

User: hadoop
Job Name: averaging pass
Job File: http://domU-12-31-39-00-ED-78.compute-1.internal:9000/mnt/hadoop/tmp/mapred/system/job_200901200759_0001/job.xml
Status: Succeeded
Started at: Tue Jan 20 08:01:56 UTC 2009
Finished at: Tue Jan 20 08:25:00 UTC 2009
Finished in: 23mins, 4sec

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	100.00%	11	0	0	11	0	0/2
reduce	100.00%	1	0	0	1	0	0/0

	Counter	Map	Reduce	Total
File Systems	HDFS bytes written	0	671,779	671,779
	Local bytes read	2,829,910,200	2,829,897,738	5,659,807,938
	Local bytes written	5,659,808,334	2,829,897,738	8,489,706,072
Job Counters	Launched reduce tasks	0	0	1
	Rack-local map tasks	0	0	11
	Launched map tasks	0	0	13
	Reduce input groups	0	17,359	17,359
	Combine output records	0	0	0
	Map input records	100,480,507	0	100,480,507

This display shows a variety of file system and cluster counters.

3. Choose one of the following actions:

To...	Do this...																																										
Find out more about the killed tasks	<p>Click on an entry in the Failed/Killed Task Attempts column.</p> <p>Hadoop job_200901200759_0001 failures on domU-12-31-39-00-ED-78</p> <table border="1"> <thead> <tr> <th>Attempt</th><th>Task</th><th>Machine</th><th>State</th><th>Error</th><th>Logs</th></tr> </thead> <tbody> <tr> <td>attempt_200901200759_0001_m_000008_0</td><td>task_200901200759_0001_m_000008</td><td>domU-12-31-39-00-C0-94 compute-1 internal</td><td>KILLED</td><td></td><td>Last 4KB Last 8KB All</td></tr> <tr> <td>attempt_200901200759_0001_m_000009_1</td><td>task_200901200759_0001_m_000009</td><td>domU-12-31-39-00-A1-B6 compute-1 internal</td><td>KILLED</td><td></td><td>Last 4KB Last 8KB All</td></tr> </tbody> </table>	Attempt	Task	Machine	State	Error	Logs	attempt_200901200759_0001_m_000008_0	task_200901200759_0001_m_000008	domU-12-31-39-00-C0-94 compute-1 internal	KILLED		Last 4KB Last 8KB All	attempt_200901200759_0001_m_000009_1	task_200901200759_0001_m_000009	domU-12-31-39-00-A1-B6 compute-1 internal	KILLED		Last 4KB Last 8KB All																								
Attempt	Task	Machine	State	Error	Logs																																						
attempt_200901200759_0001_m_000008_0	task_200901200759_0001_m_000008	domU-12-31-39-00-C0-94 compute-1 internal	KILLED		Last 4KB Last 8KB All																																						
attempt_200901200759_0001_m_000009_1	task_200901200759_0001_m_000009	domU-12-31-39-00-A1-B6 compute-1 internal	KILLED		Last 4KB Last 8KB All																																						
Get more information about the mapper tasks	<p>Click map.</p> <p>Hadoop displays all of the tasks completed and their status.</p> <p>Hadoop map task list for job_200901200759_0001 on domU-12-31-39-00-ED-78</p> <p>All Tasks</p> <table border="1"> <thead> <tr> <th>Task</th><th>Complete</th><th>Status</th><th>Start Time</th><th>Finish Time</th><th>Errors</th><th>Counters</th></tr> </thead> <tbody> <tr> <td>task_200901200759_0001_m_000000</td><td>100.00%</td><td>s3n://anhi-test-data/netflix/input/part-aa0+286875349</td><td>20-Jan-2009 08:01:57</td><td>20-Jan-2009 08:09:18 (7mins, 21sec)</td><td></td><td></td></tr> <tr> <td>task_200901200759_0001_m_000001</td><td>100.00%</td><td>s3n://anhi-test-data/netflix/input/part-aa0+279753239</td><td>20-Jan-2009 08:01:57</td><td>20-Jan-2009 08:08:10 (6mins, 13sec)</td><td></td><td></td></tr> <tr> <td>task_200901200759_0001_m_000002</td><td>100.00%</td><td>s3n://anhi-test-data/netflix/input/part-ag0+276353511</td><td>20-Jan-2009 08:01:58</td><td>20-Jan-2009 08:08:34 (6mins, 38sec)</td><td></td><td></td></tr> <tr> <td>task_200901200759_0001_m_000003</td><td>100.00%</td><td>s3n://anhi-test-data/netflix/input/part-af0+276195815</td><td>20-Jan-2009 08:01:58</td><td>20-Jan-2009 08:08:33 (6mins, 35sec)</td><td></td><td></td></tr> <tr> <td>task_200901200759_0001_m_000004</td><td>100.00%</td><td>s3n://anhi-test-data/netflix/input/part-ab0+276051944</td><td>20-Jan-2009 08:08:10</td><td>20-Jan-2009 08:14:58 (6mins, 47sec)</td><td></td><td></td></tr> </tbody> </table> <p>All of the mapper tasks completed successfully.</p>	Task	Complete	Status	Start Time	Finish Time	Errors	Counters	task_200901200759_0001_m_000000	100.00%	s3n://anhi-test-data/netflix/input/part-aa0+286875349	20-Jan-2009 08:01:57	20-Jan-2009 08:09:18 (7mins, 21sec)			task_200901200759_0001_m_000001	100.00%	s3n://anhi-test-data/netflix/input/part-aa0+279753239	20-Jan-2009 08:01:57	20-Jan-2009 08:08:10 (6mins, 13sec)			task_200901200759_0001_m_000002	100.00%	s3n://anhi-test-data/netflix/input/part-ag0+276353511	20-Jan-2009 08:01:58	20-Jan-2009 08:08:34 (6mins, 38sec)			task_200901200759_0001_m_000003	100.00%	s3n://anhi-test-data/netflix/input/part-af0+276195815	20-Jan-2009 08:01:58	20-Jan-2009 08:08:33 (6mins, 35sec)			task_200901200759_0001_m_000004	100.00%	s3n://anhi-test-data/netflix/input/part-ab0+276051944	20-Jan-2009 08:08:10	20-Jan-2009 08:14:58 (6mins, 47sec)		
Task	Complete	Status	Start Time	Finish Time	Errors	Counters																																					
task_200901200759_0001_m_000000	100.00%	s3n://anhi-test-data/netflix/input/part-aa0+286875349	20-Jan-2009 08:01:57	20-Jan-2009 08:09:18 (7mins, 21sec)																																							
task_200901200759_0001_m_000001	100.00%	s3n://anhi-test-data/netflix/input/part-aa0+279753239	20-Jan-2009 08:01:57	20-Jan-2009 08:08:10 (6mins, 13sec)																																							
task_200901200759_0001_m_000002	100.00%	s3n://anhi-test-data/netflix/input/part-ag0+276353511	20-Jan-2009 08:01:58	20-Jan-2009 08:08:34 (6mins, 38sec)																																							
task_200901200759_0001_m_000003	100.00%	s3n://anhi-test-data/netflix/input/part-af0+276195815	20-Jan-2009 08:01:58	20-Jan-2009 08:08:33 (6mins, 35sec)																																							
task_200901200759_0001_m_000004	100.00%	s3n://anhi-test-data/netflix/input/part-ab0+276051944	20-Jan-2009 08:08:10	20-Jan-2009 08:14:58 (6mins, 47sec)																																							
Display task counters	<p>Click on an entry in the Counters column.</p> <p>Hadoop displays the task counter information.</p> <p>Counters for task_200901200759_0001_m_000000</p> <table border="1"> <thead> <tr> <th colspan="2">File Systems</th> </tr> </thead> <tbody> <tr> <td>Local bytes read</td><td>293,355,815</td></tr> <tr> <td>Local bytes written</td><td>586,710,428</td></tr> </tbody> </table> <table border="1"> <thead> <tr> <th colspan="2">Map-Reduce Framework</th> </tr> </thead> <tbody> <tr> <td>Combine output records</td><td>0</td></tr> <tr> <td>Map input records</td><td>10,000,000</td></tr> <tr> <td>Map output bytes</td><td>271,080,499</td></tr> <tr> <td>Map input bytes</td><td>286,875,349</td></tr> <tr> <td>Combine input records</td><td>0</td></tr> <tr> <td>Map output records</td><td>10,000,000</td></tr> </tbody> </table>	File Systems		Local bytes read	293,355,815	Local bytes written	586,710,428	Map-Reduce Framework		Combine output records	0	Map input records	10,000,000	Map output bytes	271,080,499	Map input bytes	286,875,349	Combine input records	0	Map output records	10,000,000																						
File Systems																																											
Local bytes read	293,355,815																																										
Local bytes written	586,710,428																																										
Map-Reduce Framework																																											
Combine output records	0																																										
Map input records	10,000,000																																										
Map output bytes	271,080,499																																										
Map input bytes	286,875,349																																										
Combine input records	0																																										
Map output records	10,000,000																																										

To...	Do this...																				
Get information about tasks	<p>Click a task. Hadoop displays task information.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <p>Job job_200901200759_0001</p> <p>All Task Attempts</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Task Attempts</th> <th>Machine</th> <th>Status</th> <th>Progress</th> <th>Start Time</th> <th>Finish Time</th> <th>Errors</th> <th>Task Logs</th> <th>Counters</th> <th>Actions</th> </tr> </thead> <tbody> <tr> <td>attempt_200901200759_0001_m_000000</td> <td>0.1.39.0-A1-B6.compute-1.internal</td> <td>SUCCEEDED</td> <td>100.00%</td> <td>20-Jan-2009 08:01:58</td> <td>20-Jan-2009 08:01:58 (7mins, 20sec)</td> <td>0</td> <td>Last Log Last Status All</td> <td>8</td> <td></td> </tr> </tbody> </table> <p>Input Split Locations</p> <p>/a/b/c/d/e/localhost</p> </div>	Task Attempts	Machine	Status	Progress	Start Time	Finish Time	Errors	Task Logs	Counters	Actions	attempt_200901200759_0001_m_000000	0.1.39.0-A1-B6.compute-1.internal	SUCCEEDED	100.00%	20-Jan-2009 08:01:58	20-Jan-2009 08:01:58 (7mins, 20sec)	0	Last Log Last Status All	8	
Task Attempts	Machine	Status	Progress	Start Time	Finish Time	Errors	Task Logs	Counters	Actions												
attempt_200901200759_0001_m_000000	0.1.39.0-A1-B6.compute-1.internal	SUCCEEDED	100.00%	20-Jan-2009 08:01:58	20-Jan-2009 08:01:58 (7mins, 20sec)	0	Last Log Last Status All	8													

4. On the **All Task Attempts** pane, choose one of the following actions:

To...	Do This...										
Get information about the node that ran the task	<p>Click an entry in the Machine column. Hadoop displays host information.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <p>tracker_domU-12-31-39-00-A1-B6.compute-1.internal:localhost/127.0.0.1:43727 Task Tracker Status</p> <p> hadoop Version: 0.18. r Compiled: Mon Jan 5 21:29:34 UTC 2009 by root</p> <p>Running tasks</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Task Attempts</th> <th>Status</th> <th>Progress</th> <th>Errors</th> </tr> </thead> </table> <p>Non-Running Tasks</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Task Attempts</th> <th>Status</th> </tr> </thead> </table> <p>Tasks from Running Jobs</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Task Attempts</th> <th>Status</th> <th>Progress</th> <th>Errors</th> </tr> </thead> </table> <p>Local Logs</p> <p>Log directory</p> </div>	Task Attempts	Status	Progress	Errors	Task Attempts	Status	Task Attempts	Status	Progress	Errors
Task Attempts	Status	Progress	Errors								
Task Attempts	Status										
Task Attempts	Status	Progress	Errors								

To...	Do This...
See the task logs	<p>Click an entry in the Task Logs column.</p> <p>Hadoop displays the logs.</p> <div style="border: 1px solid blue; padding: 5px; background-color: #f0f0f0;"> <p>Task Logs: 'attempt_200901200759_0001_m_000000_0'</p> <p><u>student log</u></p> <p><u>student log</u></p> <p><u>error log</u></p> <pre> kvenid = 491505; length = 491520 2009-01-20 08:01:01,497 INFO org.apache.hadoop.mapred.MapTask: Index: (0, 10770031, 10770032) 2009-01-20 08:01:01,497 INFO org.apache.hadoop.mapred.MapTask: Finishing map output: buffer full = false and record full = true 2009-01-20 08:01:01,499 INFO org.apache.hadoop.mapred.MapTask: Spilling map output: buffer full = false and record full = true 2009-01-20 08:01:01,499 INFO org.apache.hadoop.mapred.MapTask: bufferstart = 61118070; bufend = 794945608; bufvoid = 149442080 2009-01-20 08:01:01,499 INFO org.apache.hadoop.mapred.MapTask: length = 9837600; length = 491520 2009-01-20 08:01:01,499 INFO org.apache.hadoop.mapred.MapTask: Index: (0, 15613970, 15613970) 2009-01-20 08:01:01,499 INFO org.apache.hadoop.mapred.MapTask: buffer full = false and record full = true 2009-01-20 08:01:01,499 INFO org.apache.hadoop.mapred.MapTask: bufferstart = 74945600; bufend = 85716701; bufvoid = 149442080 2009-01-20 08:01:01,499 INFO org.apache.hadoop.mapred.MapTask: length = 9837600; length = 491520 2009-01-20 08:01:01,499 INFO org.apache.hadoop.mapred.MapTask: buffer full = false and record full = true 2009-01-20 08:01:01,499 INFO org.apache.hadoop.mapred.MapTask: bufferstart = 98495600; bufend = 294895; bufvoid = 149442080 2009-01-20 08:01:01,499 INFO org.apache.hadoop.mapred.MapTask: length = 9837600; length = 491520 2009-01-20 08:01:01,499 INFO org.apache.hadoop.mapred.MapTask: buffer full = false and record full = true 2009-01-20 08:01:01,499 INFO org.apache.hadoop.mapred.MapTask: bufferstart = 11577600; bufend = 11577600; bufvoid = 149442080 2009-01-20 08:01:01,499 INFO org.apache.hadoop.mapred.MapTask: length = 9837600; length = 491520 2009-01-20 08:01:01,499 INFO org.apache.hadoop.mapred.MapTask: buffer full = false and record full = true 2009-01-20 08:01:01,499 INFO org.apache.hadoop.mapred.MapTask: bufferstart = 17198611; bufend = 27198611; bufvoid = 149442080 2009-01-20 08:01:01,499 INFO org.apache.hadoop.mapred.MapTask: length = 9837600; length = 491520 2009-01-20 08:01:01,499 INFO org.apache.hadoop.mapred.MapTask: buffer full = false and record full = true 2009-01-20 08:01:01,499 INFO org.apache.hadoop.mapred.MapTask: bufferstart = 284895; bufend = 19489501; bufvoid = 149442080 2009-01-20 08:01:01,499 INFO org.apache.hadoop.mapred.MapTask: length = 9837600; length = 491520 2009-01-20 08:01:01,499 INFO org.apache.hadoop.mapred.MapTask: buffer full = false and record full = true 2009-01-20 08:01:01,499 INFO org.apache.hadoop.mapred.MapTask: bufferstart = 97194511; bufend = 105319918; bufvoid = 149442080 2009-01-20 08:01:01,499 INFO org.apache.hadoop.mapred.MapTask: length = 9837600; length = 491520 2009-01-20 08:01:01,499 INFO org.apache.hadoop.mapred.MapTask: buffer full = false and record full = true 2009-01-20 08:01:01,499 INFO org.apache.hadoop.mapred.MapTask: bufferstart = 0; bufend = 98285; bufvoid = 149442080 2009-01-20 08:01:01,499 INFO org.apache.hadoop.mapred.MapTask: length = 9837600; length = 491520 2009-01-20 08:01:01,499 INFO org.apache.hadoop.mapred.MapTask: buffer full = false and record full = true 2009-01-20 08:01:01,499 INFO org.apache.hadoop.mapred.MapTask: bufferstart = 105319918; bufend = 115759323; bufvoid = 149442080 2009-01-20 08:01:01,499 INFO org.apache.hadoop.mapred.MapTask: length = 9837600; length = 491520 2009-01-20 08:01:01,499 INFO org.apache.hadoop.mapred.MapTask: buffer full = false and record full = true 2009-01-20 08:01:01,499 INFO org.apache.hadoop.mapred.MapTask: bufferstart = 98285; bufend = 4915101; length = 491520 2009-01-20 08:01:01,499 INFO org.apache.hadoop.mapred.MapTask: length = 9837600; length = 491520 </pre> </div>

To view the HDFS NameNode web interface

- Open <http://master-public-dns-name:9101/>. The Hadoop Distributed File System (HDFS) web interface is hosted on the master node of the cluster. You can access the web interface by using SSH to create a tunnel to the master node and configuring a SOCKS proxy so your browser can view websites hosted on the master node. For more information, see [View Web Interfaces on the Cluster \(Hadoop 2.x\) \(p. 411\)](#).

NameNode 'domU-12-31-38-00-68-47.compute-1.internal:9000'

Started: Fri Mar 20 00:31:09 UTC 2009
 Version: 0.18, r
 Compiled: Thu Mar 12 18:17:37 UTC 2009 by root
 Upgrades: There are no upgrades in progress.

[Browse the filesystem](#)

Cluster Summary

8 files and directories, 6 blocks = 14 total. Heap Size is 4.94 MB / 992.31 MB (0%)

Capacity	: 587.08 GB
DFS Remaining	: 545.39 GB
DFS Used	: 96 KB
DFS Used%	: 0 %
Live Nodes	: 4
Dead Nodes	: 0

Live Datanodes : 4

Node	Last Contact	Admin State	Size (GB)	Used (%)	Used (%)	Remaining (GB)	Blocks
domU-12-31-38-00-51-83	2	In Service	146.77	0		136.35	4
domU-12-31-38-00-52-03	2	In Service	146.77	0		136.35	6
domU-12-31-38-00-79-31	0	In Service	146.77	0		136.35	4
domU-12-31-38-00-79-58	2	In Service	146.77	0		136.35	6

Dead Datanodes : 0

To view the TaskTracker web interface

- Open <http://master-public-dns-name:9103/>. The TaskTracker web interface is hosted on the master node of the cluster. You can access the web interface by using SSH to create a tunnel

to the master node and configuring a SOCKS proxy so your browser can view websites hosted on the master node. For more information, see [View Web Interfaces on the Cluster \(Hadoop 2.x\) \(p. 411\)](#).

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. For more information, see [Configure Logging and Debugging \(Optional\) \(p. 133\)](#).

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 \(p. 418\)](#)
- [View Log Files Archived to Amazon S3 \(p. 419\)](#)
- [View Log Files in the Debugging Tool \(p. 421\)](#)

View Log Files on the Master Node

The following table lists some of the log files you'll find on the master node.

Location	Description
/mnt/var/log/bootstrap-actions	Logs written during the processing of the bootstrap actions.
/mnt/var/log/hadoop-state-pusher	Logs written by the Hadoop state pusher process.
/mnt/var/log/instance-controller	Instance controller logs.
/mnt/var/log/instance-state	Instance state logs. These contain information about the CPU, memory state, and garbage collector threads of the node.
/mnt/var/log/service-nanny	Logs written by the service nanny process.
/mnt/var/log/hadoop	Hadoop logs, such as those written by the jobtracker and namenode processes.

Location	Description
/mnt/var/log/hadoop/steps/ <i>N</i>	<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: s-1234ABCDEFGH and s-5678IJKLMNOP. The first step is located in /mnt/var/log/hadoop/steps/s-1234ABCDEFGH/ and the second step in /mnt/var/log/hadoop/steps/s-5678IJKLMNOP/.</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 \(p. 446\)](#).
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

Amazon EMR does not automatically archive log files to Amazon S3. You must configure this when you launch the cluster. For more information, see [Configure Logging and Debugging \(Optional\) \(p. 133\)](#).

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
<i>/JobFlowId/daemons/</i>	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.
<i>/JobFlowId/jobs/</i>	Job logs and the configuration XML file for each Hadoop job.
<i>/JobFlowId/node/</i>	Node logs, including bootstrap action 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.
<i>/JobFlowId/steps/<i>N</i>/</i>	<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: s-1234ABCDEFGH and s-5678IJKLMNOP. The first step is located in <i>/mnt/var/log/hadoop/steps/s-1234ABCDEFGH/</i> and the second step in <i>/mnt/var/log/hadoop/steps/s-5678IJKLMNOP/</i>.</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.
<i>/JobFlowId/task-attempts/</i>	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 you 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

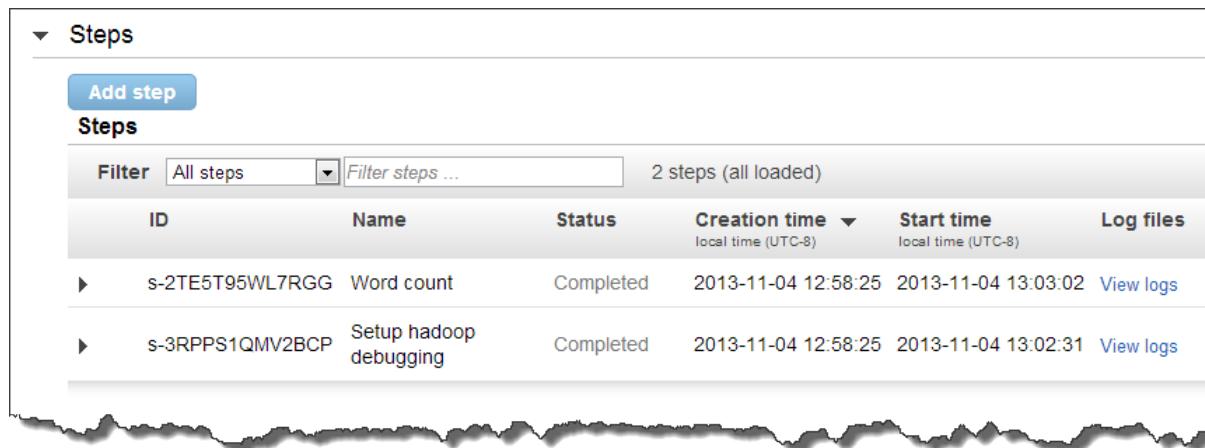
Amazon EMR does not automatically enable the debugging tool. You must configure this when you launch the cluster. For more information, see [Configure Logging and Debugging \(Optional\) \(p. 133\)](#).

To view cluster logs using the console

1. Open the Amazon Elastic MapReduce console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. From the **Cluster List** page, click the details icon next to the cluster you want to view.

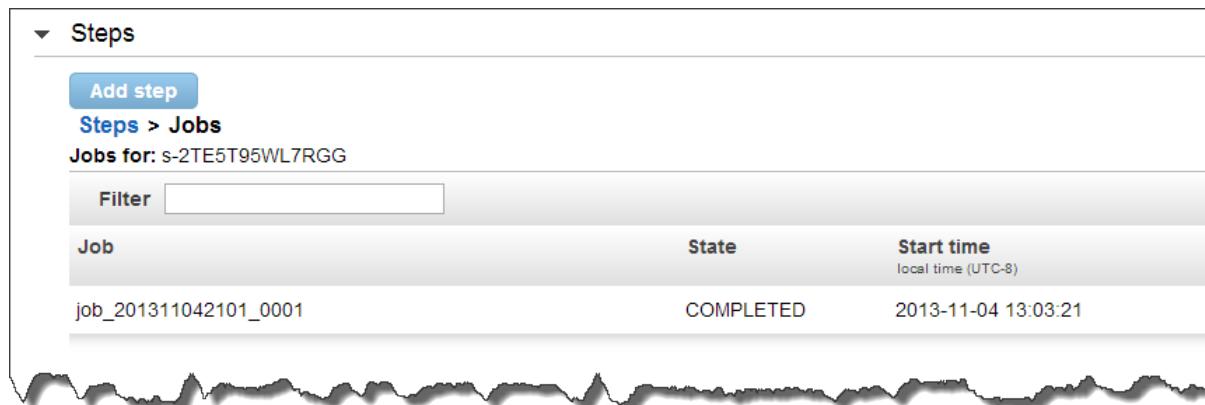
This brings up the **Cluster Details** page. In the **Steps** section, the links to the right of each step display the various types of logs available for the step. These logs are generated by Amazon EMR.

3. To view a list of the Hadoop jobs associated with a given step, click the **View Jobs** link to the right of the step.



ID	Name	Status	Creation time	Start time	Log files
s-2TE5T95WL7RGG	Word count	Completed	2013-11-04 12:58:25	2013-11-04 13:03:02	View logs
s-3RPPS1QMV2BCP	Setup hadoop debugging	Completed	2013-11-04 12:58:25	2013-11-04 13:02:31	View logs

4. To view a list of the Hadoop tasks associated with a given job, click the **View Tasks** link to the right of the job.



Job	State	Start time
job_201311042101_0001	COMPLETED	2013-11-04 13:03:21

5. To view a list of the attempts a given task has run while trying to complete, click the **View Attempts** link to the right of the task.

The screenshot shows the 'Steps' section of the Amazon EMR developer guide. It displays a table of tasks for a specific job and step. The table columns are: Task, Type, State, and Start time (local time (UTC-8)). The tasks listed are: 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), and m_000012 (map, COMPLETED, 2013-11-04 13:05:08). A 'Filter' input field is present above the table.

Task	Type	State	Start time local time (UTC-8)
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

6. To view the logs generated by a task attempt, click the **stderr**, **stdout**, and **syslog** links to the right of the task attempt.

The screenshot shows the 'Task attempts' section of the Amazon EMR developer guide. It displays a table of attempts for a specific task. The table columns are: Attempt, Type, State, and Log file. The attempt listed is: 0 (reduce, SUCCEEDED, control). A 'Filter' input field is present above the table.

Attempt	Type	State	Log file
0	reduce	SUCCEEDED	control

The debugging tool displays links to the log files after Amazon EMR uploads the log files to your bucket on Amazon S3. Because log files are uploaded to Amazon S3 every 5 minutes, it can take a few minutes for the log file uploads to complete after the step completes.

Amazon EMR periodically updates the status of Hadoop jobs, tasks, and task attempts in the debugging tool. You can click **Refresh List** in the debugging panes to get the most up-to-date status of these items.

View Cluster Instances in Amazon EC2

To help you manage your resources, Amazon EC2 allows you to assign metadata to resources in the form of tags. Each Amazon EC2 tag consists of a key and a value. Tags allow you to categorize your Amazon EC2 resources in different ways: for example, by purpose, owner, or environment.

You can search and filter resources based on the tags. The tags assigned using your AWS account are available only to you. Other accounts sharing the resource cannot view your tags.

Amazon EMR automatically tags each EC2 instance it launches with key-value pairs that identify the cluster and the instance group to which the instance belongs. This makes it easy to filter your EC2 instances to show, for example, only those instances belonging to a particular cluster or to show all of the currently running instances in the task-instance group. This is especially useful if you are running several clusters concurrently or managing large numbers of EC2 instances.

These are the predefined key-value pairs that Amazon EMR assigns:

Key	Value
aws:elasticmapreduce:job-flow-id	<job-flow-identifier>
aws:elasticmapreduce:instance-group-role	<group-role>

The values are further defined as follows:

- The <job-flow-identifier> is the ID of the cluster the instance is provisioned for. It appears in the format j-XXXXXXXXXXXXXX.
- The <group-role> is one of the following values: master, core, or task. These values correspond to the master instance group, core instance group, and task instance group.

You can view and filter on the tags that Amazon EMR adds. For more information, see [Using Tags](#) in the *Amazon Elastic Compute Cloud User Guide*. Because the tags set by Amazon EMR are system tags and cannot be edited or deleted, the sections on displaying and filtering tags are the most relevant.

Note

Amazon EMR adds tags to the EC2 instance when its status 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

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 a Machine Image \(p. 51\)](#).

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 \(p. 430\)](#).

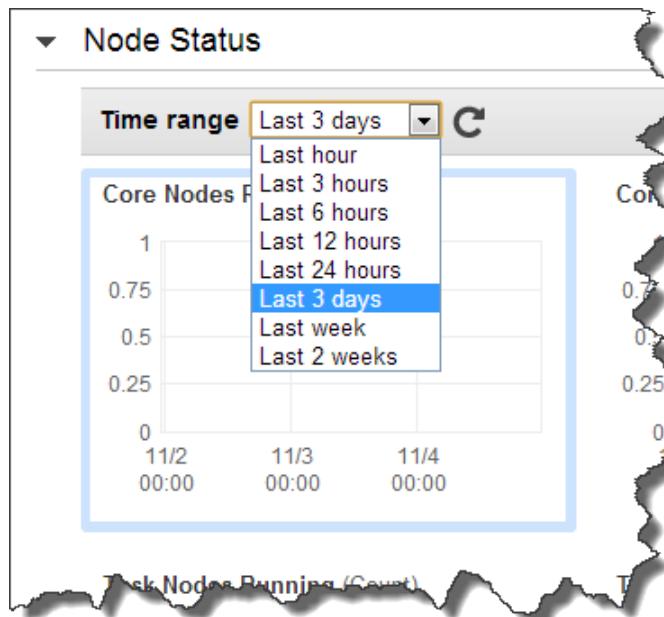
How do I?	Relevant Metrics
Track the progress of my cluster	Look at the <code>RunningMapTasks</code> , <code>RemainingMapTasks</code> , <code>RunningReduceTasks</code> , and <code>RemainingReduceTasks</code> metrics.
Detect clusters that are idle	The <code>IsIdle</code> metric tracks whether a cluster 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 Elastic MapReduce 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, click the **Time range** field 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 Amazon 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 X EMR Metrics

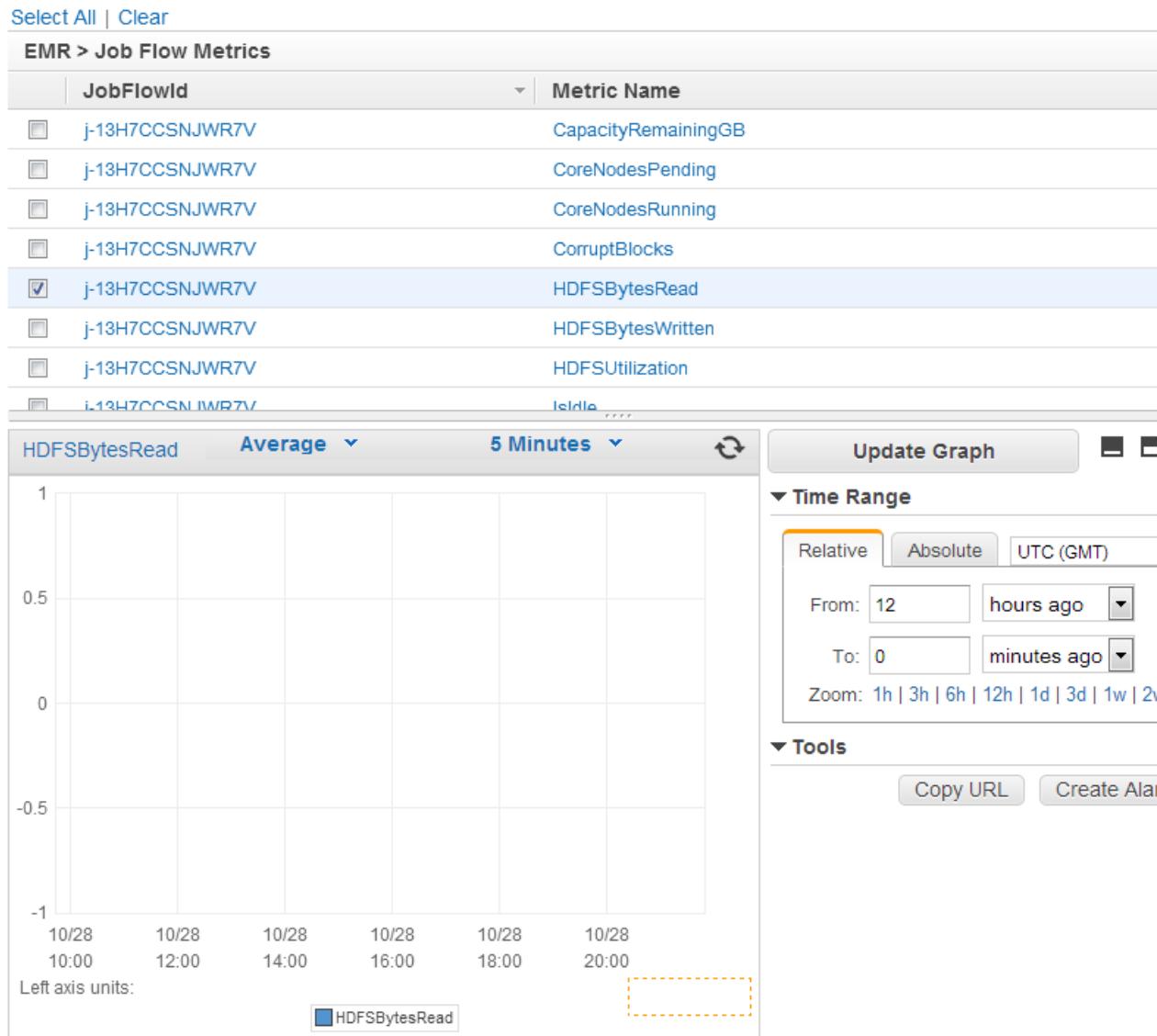
Showing the first 200 matching metrics. 4 additional metrics not listed for EMR Metrics
Browse Metrics button above.

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	IsIdle
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.



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 Amazon CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. Click the **Create Alarm** button. This launches the **Create Alarm Wizard**.

Alarm Summary

You do not have any alarms created in the EU West (Ireland) region. Alarms allow you to send notifications or execute Auto Scaling actions in response to any CloudWatch metric.

You can now use Amazon CloudWatch alarms to monitor the estimated charges on your AWS bill. Receive email alerts whenever charges exceed a threshold you define. Visit the CloudWatch US East (Virginia) region to manage your billing alarms.

[Go to CloudWatch US East region](#)

Service Health



Current Status	Details
Amazon CloudWatch Service (Ireland)	Service is operating normally
View complete service health details	

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.

1. Select Metric

2. Define Alarm

[Back](#) [Next](#)

[Cancel](#)

Please set the alarm threshold, actions and click **Create Alarm** below.

[Create Alarm](#)

Alarm Threshold

Provide the details and threshold for your alarm. Use the graph on the right to help set the appropriate threshold.

Name:

Description:

Whenever: CapacityRemainingGB

is: \geq

for: consecutive period(s)

Actions

Define what actions are taken when your alarm changes state.

Notification	Delete
Whenever this alarm: <input type="text" value="State is ALARM"/> Send notification to: <input type="text" value="Please select an SNS topic"/> Create topic	+ Notification + AutoScaling Action + EC2 Action

CapacityRemainingGB ≥ 0

This alarm will trigger when the blue line goes up to or above the red line for a duration of 5 minutes

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.

Actions

Define what actions are taken when your alarm changes state.

Notification	Delete
Whenever this alarm: <input type="text" value="State is ALARM"/> Send notification to: <input type="text" value="MySNSTopicName"/> Email list: <input type="text" value="user@example.com"/>	+ Notification + AutoScaling Action + EC2 Action

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 Amazon EMR console and pushes to CloudWatch.

Amazon EMR Metrics for Hadoop 1 AMIs

Amazon EMR sends data for several metrics to Amazon 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 will be reported until the cluster becomes available again.

Metric	Description
CoreNodesPending	<p>The number of core nodes waiting to be assigned. All of the core nodes requested may not be immediately available; this metric reports the pending requests. Data points for this metric are reported only when a corresponding instance group exists.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Count</i></p>
CoreNodesRunning	<p>The number of core nodes working. Data points for this metric are reported only when a corresponding instance group exists.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Count</i></p>
HBaseBackupFailed	<p>Whether the last backup failed. This is set to 0 by default and updated to 1 if the previous backup attempt failed. This metric is only reported for HBase clusters.</p> <p>Use case: Monitor HBase backups</p> <p>Units: <i>Count</i></p>
HBaseMostRecentBackupDuration	<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
HBaseTimeSinceLastSuccessfulBackup	<p>The number of elapsed minutes since 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>
HDFSBytesRead	<p>The number of bytes read from HDFS.</p> <p>Use case: Analyze cluster performance, Monitor cluster progress</p> <p>Units: <i>Count</i></p>
HDFSBytesWritten	<p>The number of bytes written to HDFS.</p> <p>Use case: Analyze cluster performance, Monitor cluster progress</p> <p>Units: <i>Count</i></p>
HDFSUtilization	<p>The percentage of HDFS storage currently used.</p> <p>Use case: Analyze cluster performance</p> <p>Units: <i>Percent</i></p>
IsIdle	<p>Indicates that a cluster is no longer performing work, but is still alive and accruing charges. It is set to 1 if no tasks are running and no jobs are running, and set to 0 otherwise. This value is checked at five-minute intervals and a value of 1 indicates only that the cluster was idle when checked, not that it was idle for the entire five minutes. To avoid false positives, you should raise an alarm when this value has been 1 for more than one consecutive 5-minute check. For example, you might raise an alarm on this value if it has been 1 for thirty minutes or longer.</p> <p>Use case: Monitor cluster performance</p> <p>Units: <i>Count</i></p>
JobsFailed	<p>The number of jobs in the cluster that have failed.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Count</i></p>
JobsRunning	<p>The number of jobs in the cluster that are currently running.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Count</i></p>
LiveDataNodes	<p>The percentage of data nodes that are receiving work from Hadoop.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Percent</i></p>

Metric	Description
LiveTaskTrackers	<p>The percentage of task trackers that are functional.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Percent</i></p>
MapSlotsOpen	<p>The unused map task capacity. This is calculated as the maximum number of map tasks for a given cluster, less the total number of map tasks currently running in that cluster.</p> <p>Use case: Analyze cluster performance</p> <p>Units: <i>Count</i></p>
MissingBlocks	<p>The number of blocks in which HDFS has no replicas. These might be corrupt blocks.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Count</i></p>
ReduceSlotsOpen	<p>Unused reduce task capacity. This is calculated as the maximum reduce task capacity for a given cluster, less the number of reduce tasks currently running in that cluster.</p> <p>Use case: Analyze cluster performance</p> <p>Units: <i>Count</i></p>
RemainingMapTasks	<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>
RemainingMapTasksPerSlot	<p>The ratio of the total map tasks remaining to the total map slots available in the cluster.</p> <p>Use case: Analyze cluster performance</p> <p>Units: <i>Ratio</i></p>
RemainingReduceTasks	<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>

Metric	Description
RunningMapTasks	<p>The number of running map tasks for each job. If you have a scheduler installed and multiple jobs running, multiple graphs will be generated.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>
RunningReduceTasks	<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>
S3BytesRead	<p>The number of bytes read from Amazon S3.</p> <p>Use case: Analyze cluster performance, Monitor cluster progress</p> <p>Units: <i>Count</i></p>
S3BytesWritten	<p>The number of bytes written to Amazon S3.</p> <p>Use case: Analyze cluster performance, Monitor cluster progress</p> <p>Units: <i>Count</i></p>
TaskNodesPending	<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>
TaskNodesRunning	<p>The number of task nodes working. Data points for this metric are reported only when a corresponding instance group exists.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Count</i></p>
TotalLoad	<p>The total number of concurrent data transfers.</p> <p>Use case: Monitor cluster health</p> <p>Units: <i>Count</i></p>

Amazon EMR Metrics for Hadoop 2 AMIs

In addition to the metrics above, the following metrics were added for Hadoop 2 AMIs:

Metric	Description
AppsCompleted	<p>The number of applications submitted to YARN that have completed.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>
AppsFailed	<p>The number of applications submitted to YARN that have failed to complete.</p> <p>Use case: Monitor cluster progress, Monitor cluster health</p> <p>Units: <i>Count</i></p>
AppsKilled	<p>The number of applications submitted to YARN that have been killed.</p> <p>Use case: Monitor cluster progress, Monitor cluster health</p> <p>Units: <i>Count</i></p>
AppsPending	<p>The number of applications submitted to YARN that are in a pending state.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>
AppsRunning	<p>The number of applications submitted to YARN that are running.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>
AppsSubmitted	<p>The number of applications submitted to YARN.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>
ContainerAllocated	<p>The number of resource containers allocated by the ResourceManager.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>
ContainerPending	<p>The number of containers in the queue that have not yet been allocated.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>
ContainerReserved	<p>The number of containers reserved.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>

Metric	Description
MRActiveNodes	<p>The number of nodes presently running MapReduce tasks or jobs.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>
MRDecommissionedNodes	<p>The number of nodes allocated to MapReduce applications that have been marked in a DECOMMISSIONED state.</p> <p>Use case: Monitor cluster health, Monitor cluster progress</p> <p>Units: <i>Count</i></p>
MRLostNodes	<p>The number of nodes allocated to MapReduce that have been marked in a LOST state.</p> <p>Use case: Monitor cluster health, Monitor cluster progress</p> <p>Units: <i>Count</i></p>
MRRebootedNodes	<p>The number of nodes available to MapReduce that have been rebooted and marked in a REBOOTED state.</p> <p>Use case: Monitor cluster health, Monitor cluster progress</p> <p>Units: <i>Count</i></p>
MRTotalNodes	<p>The number of nodes presently available to MapReduce jobs.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>
MRUnhealthyNodes	<p>The number of nodes available to MapReduce jobs marked in an UNHEALTHY state.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>
MemoryAllocatedMB	<p>The amount of memory allocated to the cluster.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>
MemoryAvailableMB	<p>The amount of memory available to be allocated.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>
MemoryReservedMB	<p>The amount of memory reserved.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>

Metric	Description
MemoryTotalMB	<p>The total amount of memory in the cluster.</p> <p>Use case: Monitor cluster progress</p> <p>Units: <i>Count</i></p>

Dimensions for Amazon EMR Metrics

Amazon EMR data can be filtered using any of the dimensions in the following table.

Dimension	Description
JobFlowId	The identifier for a cluster. You can find this value by clicking on the cluster in the Amazon EMR console. It takes the form <code>j-XXXXXXXXXXXXXX</code> .
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 <code>job_XXXXXXXXXXXX_XXXX</code> .

Logging Amazon Elastic MapReduce API Calls in AWS CloudTrail

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 \(p. 436\)](#)
- [Understanding Amazon EMR Log File Entries \(p. 437\)](#)

Amazon EMR Information in CloudTrail

If CloudTrail logging is turned on, calls made to all Amazon EMR actions are captured in log files. All of the Amazon EMR actions are documented in the [Amazon EMR API Reference](#). For example, calls to the **ListClusters**, **DescribeCluster**, and **RunJobFlow** actions generate entries in CloudTrail log files.

Every log entry contains information about who generated the request. For example, if a request is made to create and run a new job flow (**RunJobFlow**), CloudTrail logs the user identity of the person or service that made the request. The user identity information helps you determine whether the request was made with root or IAM user credentials, with temporary security credentials for a role or federated user, or by another AWS service. For more information about CloudTrail fields, see [CloudTrail Event Reference](#) in the AWS CloudTrail User Guide.

You can store your log files in your bucket for as long as you want, but you can also define Amazon S3 lifecycle rules to archive or delete log files automatically. By default, your log files are encrypted by using Amazon S3 server-side encryption (SSE).

Understanding Amazon EMR Log File Entries

CloudTrail log files can contain one or more log entries composed of multiple JSON-formatted events. A log entry represents a single request from any source and includes information about the requested action, any input parameters, the date and time of the action, and so on. The log entries do not appear in any particular order. That is, they do not represent an ordered stack trace of the public API calls.

The following log file record shows that an IAM user called the **RunJobFlow** action by using the SDK.

```
        "name": "Integ 1xmllarge",
        "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

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 \(p. 438\)](#)
- [View Ganglia Metrics \(p. 440\)](#)
- [Ganglia Reports \(p. 441\)](#)
- [Hadoop Metrics in Ganglia \(p. 446\)](#)

Add Ganglia to a Cluster

To add Ganglia to a cluster using the console

1. Open the Amazon Elastic MapReduce console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Click **Create cluster**.
3. Under the **Additional Applications** list, choose **Ganglia** and click **Configure and add**.
4. Proceed to create the cluster as described in [Plan an Amazon EMR Cluster \(p. 29\)](#).

To add a Ganglia bootstrap action using the CLI

- When you create a new cluster using the 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, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

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://myawsbucket/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://myawsbucket/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://myawsbucket/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://myawsbucket/output/2014-01-16 --mapper s3://elasticmapreduce/samples/word
count/wordSplitter.py --reducer aggregate
```

View Ganglia Metrics

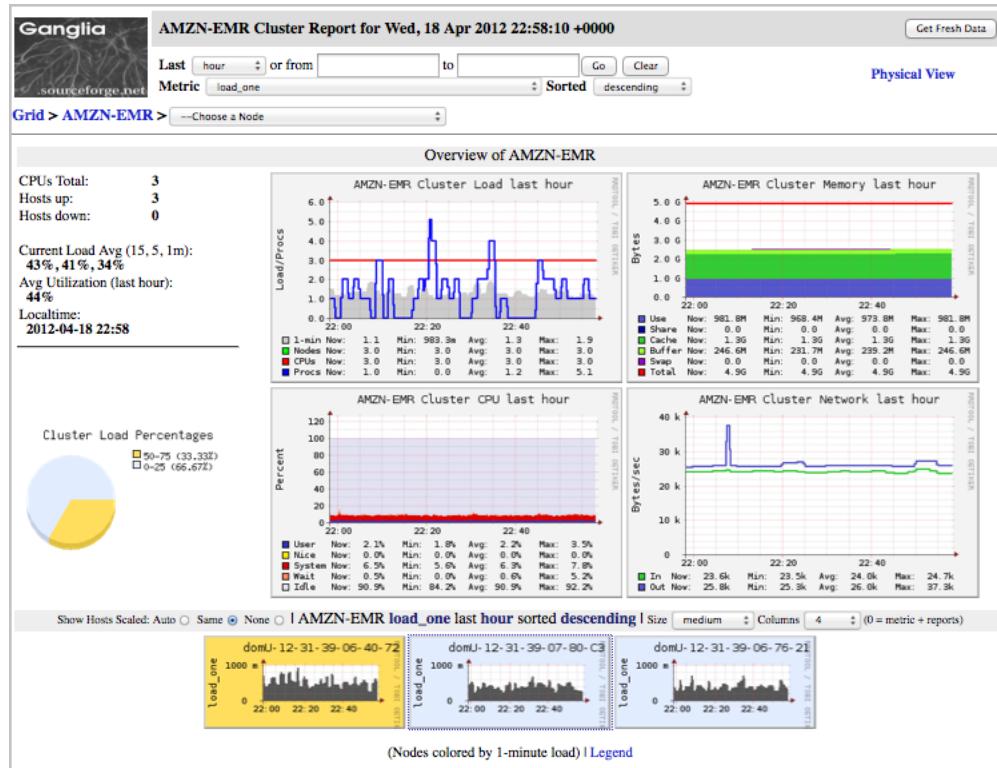
Ganglia provides a web-based user interface that you can use to view the metrics Ganglia collects. When you run Ganglia on Amazon EMR, the web interface runs on the master node and can be viewed using port forwarding, also known as creating an SSH tunnel. For more information about viewing web interfaces on Amazon EMR, see [Web Interfaces Hosted on the Master Node \(p. 450\)](#).

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 [Open an SSH Tunnel to the Master Node \(p. 452\)](#).
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 [Configure FoxyProxy 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 locate the public DNS name of the master node using the Amazon EMR console \(p. 447\)](#).

Ganglia Reports

When you open the Ganglia web reports in a browser, you see an overview of the cluster's performance, with graphs detailing the load, memory usage, CPU utilization, and network traffic of the cluster. Below the cluster statistics are graphs for each individual server in the cluster. In the preceding cluster creation example, we launched three instances, so in the following reports there are three instance charts showing the cluster data.



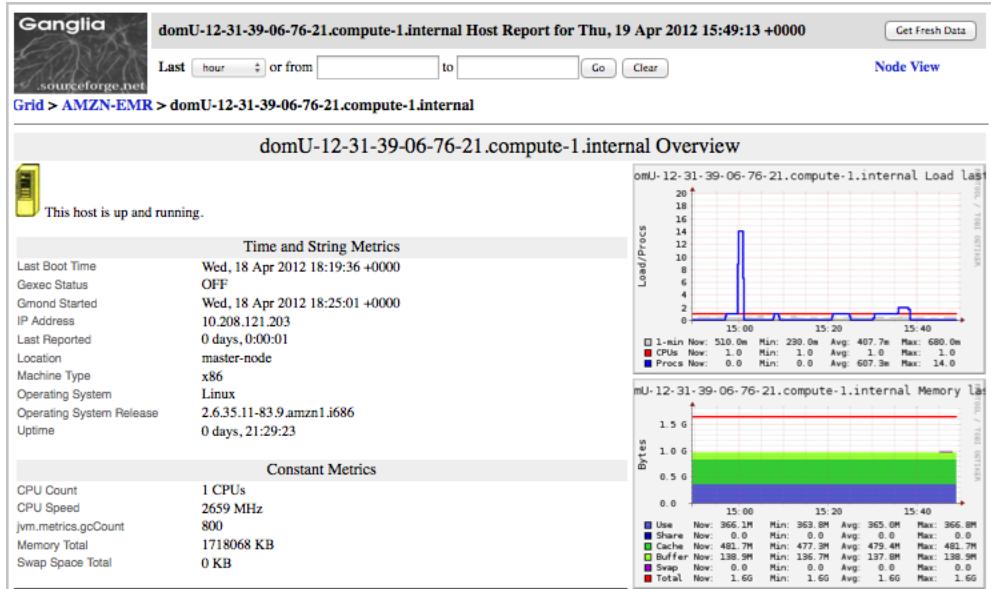
The default graph for the node instances is Load, but you can use the **Metric** drop-down list to change the statistic displayed in the node-instance graphs.



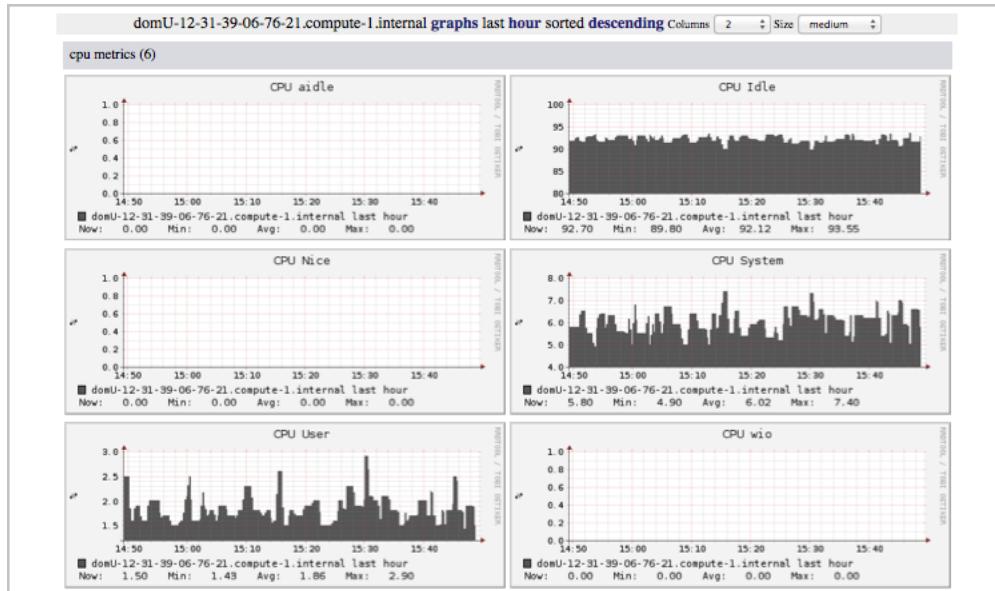
You can drill down into the full set of statistics for a given instance by selecting the node from the drop-down list or by clicking the corresponding node-instance chart.



This opens the Host Overview for the node.



If you scroll down, you can view charts of the full range of statistics collected on the instance.



Hadoop 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

Often when you run an Amazon Elastic MapReduce (Amazon EMR) cluster, all you need to do is launch the analysis and then collect the output from an Amazon S3 bucket. There are other times, however, when you'll 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, monitor performance using an application such as Ganglia that runs on the master node, debug a problem with the cluster, and more. The following sections describe techniques 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.

Note

You can connect to the master node only while the cluster is running. After the cluster terminates, the EC2 instance acting as the master node is terminated and no longer available. You also must specify an Amazon EC2 key pair when you launch the cluster, as you use the key pair as the credentials for the SSH connection. If you are launching the cluster from the console, the Amazon EC2 key pair is specified on the **ADVANCED OPTIONS** pane of the **Create a New Job Flow** wizard.

By default, Amazon EMR creates security group rules for master and slave nodes. For example, TCP port 22 is open by default to allow you to connect to the master node using SSH. Also, port 8443 and certain IP ranges are opened to allow the cluster control plane to operate.

Note

You should not modify the default control plane security group rules as that may interfere with the operation of your cluster.

Topics

- [Connect to the Master Node Using SSH \(p. 446\)](#)
- [Web Interfaces Hosted on the Master Node \(p. 450\)](#)
- [Open an SSH Tunnel to the Master Node \(p. 452\)](#)
- [Configure FoxyProxy to View Websites Hosted on the Master Node \(p. 453\)](#)

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. Once you've made this connection, it's as if the terminal on your local computer is running on the remote computer. Commands you issue locally run on the remote computer and the output of those commands 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 HBase, Hive, and Pig interactively, browse directories, read log files, and more.

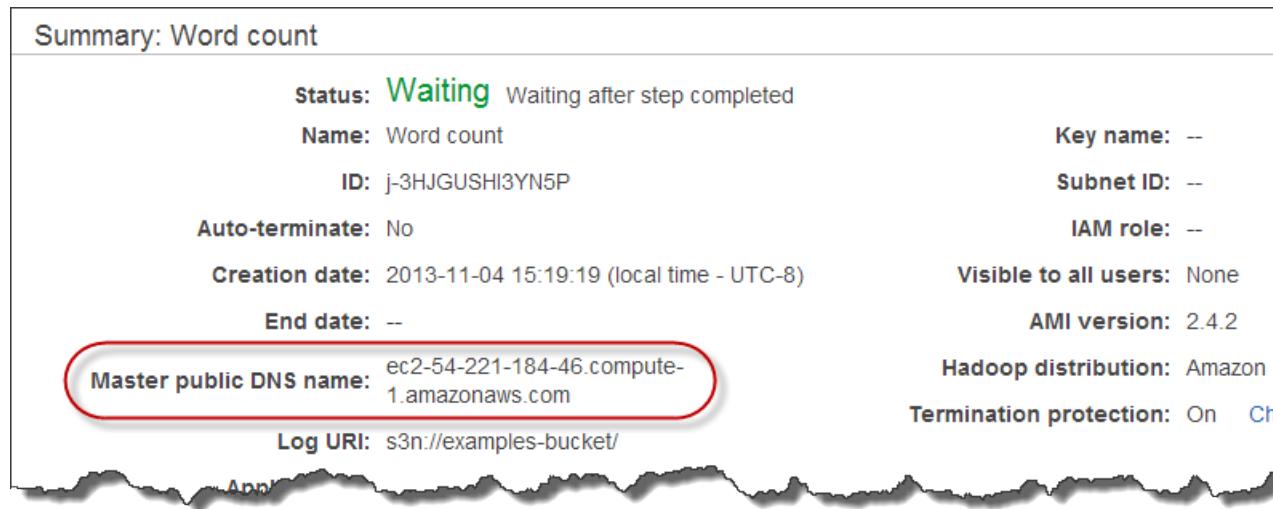
To connect to the master node using SSH, you need the public DNS name of the master node. You also must specify an Amazon EC2 key pair when you launch the cluster, as you use the key pair as the credentials for the SSH connection. If you are launching the cluster from the console, the Amazon EC2 key pair is specified on the **ADVANCED OPTIONS** pane of the **Create a New Job Flow** wizard.

Note

To permit SSH access to a master node, you must add your external source IP for TCP Port 22 to the ingress rules on the master node security group. For more information, see [Adding a Security Group Rule](#) in the *Amazon Elastic Compute Cloud User Guide*.

To locate the public DNS name of the master node using the Amazon EMR console

- In the [Amazon EMR console](#), select the job from the list of running clusters in the WAITING or RUNNING state. Details about the cluster appear in the lower pane.



The DNS name you used to connect to the instance is listed on the Description tab as **Master public DNS name**.

To locate the public DNS name of the master node using the CLI

- If you have the Amazon EMR CLI installed, you can retrieve the public DNS name of the master by running the following command.

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --list
```

- Windows users:

```
ruby elastic-mapreduce --list
```

This returns a list of all the currently active clusters in the following format. In the example below, `ec2-204-236-242-218.compute-1.amazonaws.com`, is the public DNS name of the master node for the cluster `j-3L7WK3E07HO4H`.

```
j-3L7WK3E07HO4H      WAITING      ec2-204-236-242-218.compute-1.amazonaws.com
My Job Flow
```

OpenSSH is installed on most Linux, Unix, and Mac OS X operating systems. Windows users can use an application called PuTTY to connect to the master node. Following are platform-specific instructions for opening an SSH connection.

To configure the permissions of the keypair file using Linux/Unix/Mac OS X

- Before you can use the keypair file to create an SSH connection, you must set 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 do this, SSH returns an error saying that your private key file is unprotected and will reject the key. You only need to configure these permissions the first time you use the private key to connect.

To connect to the master node using Linux/Unix/Mac OS X

- Open a terminal window. This is found at Applications/Utilities/Terminal on Mac OS X and at Applications/Accessories/Terminal on many Linux distributions.
- Check that SSH is installed by running the following command. If SSH is installed, this command returns the SSH version number. If SSH is not installed, you'll need to install the OpenSSH package from a repository.

```
ssh -v
```

- 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 `ec2-107-22-74-202.compute-1.amazonaws.com` with the Master public DNS name of your cluster and replace `~/mykeypair.pem` with the location and filename of your PEM file.

```
ssh hadoop@ec2-107-22-74-202.compute-1.amazonaws.com -i ~/mykeypair.pem
```

A warning states that the authenticity of the host you are connecting to can't be verified.

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.

- Type `yes` to continue.

To connect to the master node using the CLI on Linux/Unix/Mac OS X

- If you have the Amazon EMR CLI installed and have configured your credentials.json file so the "keypair" value is set to the name of the keypair you used to launch the cluster and "key-pair-file" value is set to the full path to your keypair .pem file, and the permissions on the .pem file are set to `0644` as shown in [To configure the permissions of the keypair file using Linux/Unix/Mac OS X \(p. 448\)](#), and you have OpenSSH installed on your machine, you can open an SSH connection to the master node by issuing the following command. This is a handy shortcut for frequent CLI users. In the example below you would replace the red text with the cluster identifier of the cluster to connect to.

```
./elastic-mapreduce -j j-3L7WK3E07HO4H --ssh
```

To close an SSH connection using Linux/Unix/Mac OS X

- When you are done working on the master node, you can close the SSH connection using the `exit` command.

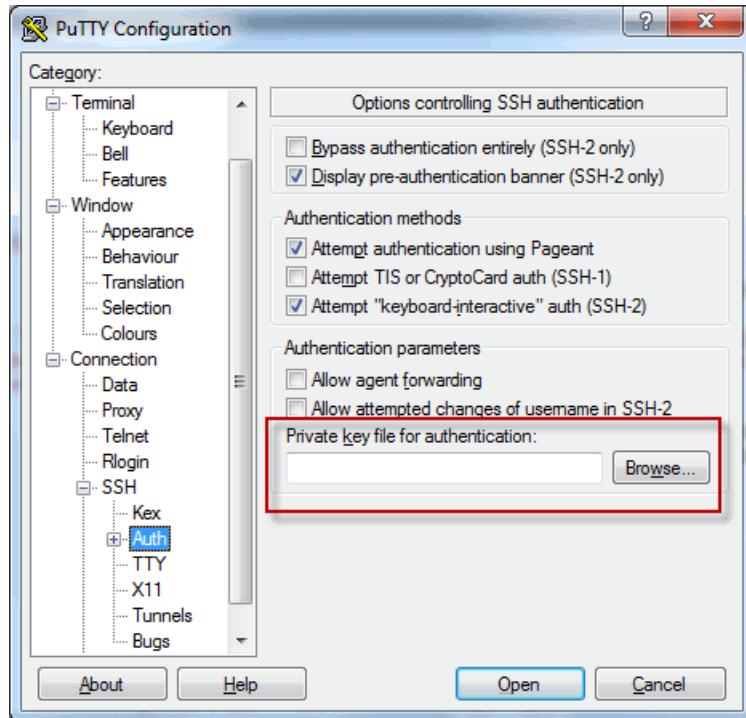
```
exit
```

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. For the option **Type of key to generate**, choose **SSH2-RSA**.
8. Click **Save private key** to save the key in the PPK format.
9. When PuTTYgen prompts you to save the key without a pass phrase, click **Yes**.
10. Enter a name for your PuTTY private key, such as `mykeypair.ppk`.
11. Click **Save**.
12. 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.

Important

If you are asked to log in, enter `hadoop`.

Web Interfaces Hosted on the Master Node

Hadoop, Ganglia, and other applications publish user interfaces as websites hosted on the master node. For security reasons, these websites are only available on the master node's local webserver (`http://localhost:port`) and are not published on the Internet. There are several ways you can access these web interfaces:

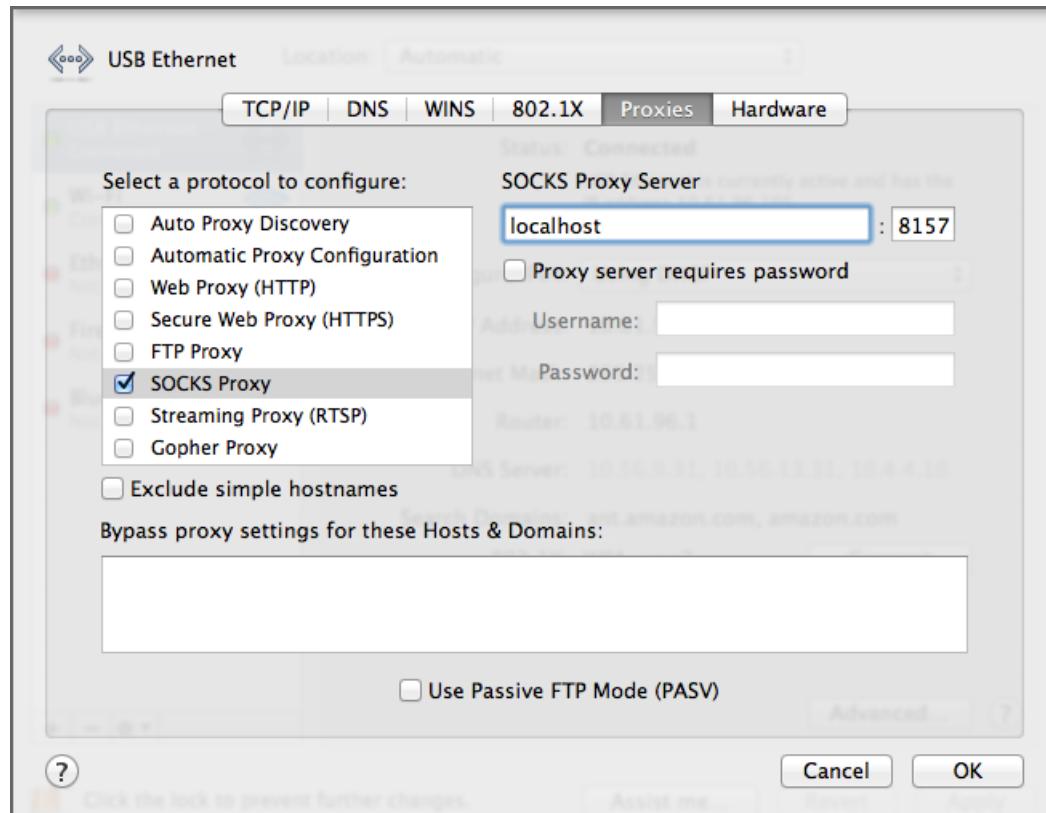
- Use SSH to connect to the master node and use the text-based browser, Lynx, to view the websites from the SSH terminal. The following example shows how to open the Hadoop JobTracker user interface using Lynx. This is the easiest and quickest way to access these web interfaces. The disadvantage is that Lynx is a text-based browser with a limited user interface that cannot display graphics.

```
lynx http://localhost:9100/
```

Note

To permit SSH access to a master node, you must add your external source IP for TCP Port 22 to the ingress rules on the master node security group. For more information, see [Adding a Security Group Rule](#) in the *Amazon Elastic Compute Cloud User Guide*.

- Create an SSH tunnel to the master node and manually configure your browser to use the SOCKS proxy you've just created for all URLs. This has the advantage of being relatively easy to configure (see your web browser's documentation for details). The disadvantage is you must then manually disable the proxy in your browser to resume normal web browsing. The following screenshot shows the settings you'd use to manually configure Safari to view the web interfaces over a SOCKS proxy.



- Create an SSH tunnel to the master node and use a browser add-on, such as FoxyProxy (an add-on for the FireFox browser), to automatically filter URLs based on text patterns and use the SOCKS proxy you've created only for domains that match the form of an EC2 instance's public DNS name. This requires that you install an add-on and configure the appropriate patterns in it, but once done, 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, see [Configure FoxyProxy to View Websites Hosted on the Master Node \(p. 453\)](#).

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, are only available if additional features have been added to the cluster.

Name of Interface	URI
Hadoop MapReduce job tracker	http://master-public-dns-name:9100/
Hadoop HDFS name node	http://master-public-dns-name:9101/
Hadoop MapReduce task tracker	http://master-public-dns-name:9103/
Ganglia Metrics Reports	http://master-public-dns-name/ganglia/

Name of Interface	URI
HBase Interface	http:// <i>master-public-dns-name</i> :60010/master-status

For more information about the Hadoop web interfaces, see [View Web Interfaces on the Cluster \(Hadoop 2.x\) \(p. 411\)](#). For more information about the Ganglia web interface, see [Monitor Performance with Ganglia \(p. 438\)](#).

Open an SSH Tunnel to the Master Node

Hadoop, Ganglia, and other applications publish user interfaces as websites hosted on the master node. For security reasons, these websites are only available on the master node's local webserver (http://localhost:port) and are not published on the Internet. To connect to the local webserver on the master node you can create a an SSH tunnel between your computer and the master node. This is also known as *port forwarding* and creates a SOCKS proxy server. For more information about the sites you might want to view on the master node, see [Web Interfaces Hosted on the Master Node \(p. 450\)](#).

Before you begin, you'll need the public DNS name of the master node. For information about how to locate this value, see [To locate the public DNS name of the master node using the Amazon EMR console \(p. 447\)](#). You also must specify an Amazon EC2 key pair when you launch the cluster, as you use the key pair as the credentials for the SSH connection. If you are launching the cluster from the console, the Amazon EC2 key pair is specified on the **ADVANCED OPTIONS** pane of the **Create a New Job Flow** wizard.

Note

To permit SSH access to a master node, you must add your external source IP for TCP Port 22 to the ingress rules on the master node security group. For more information, see [Adding a Security Group Rule](#) in the *Amazon Elastic Compute Cloud User Guide*.

To create an SSH tunnel to the master node using Linux/Unix/Mac OS X

- Open an SSH tunnel on your local machine using the following command:

```
ssh -i path-to-keyfile -ND port_number hadoop@master-public-DNS-name
```

The following shows the command with example values filled in.

```
ssh -i ~/ec2-keys/myKeyName -ND 8157 hadoop@ec2-107-22-74-202.compute-1.amazonaws.com
```

After you issue this command, the terminal remains open and not return a command prompt. It is now acting as a SOCKS server.

To create an SSH tunnel to the master node using the CLI on Linux/Unix/Mac OS X

- If you have the Amazon EMR CLI installed and have configured your credentials.json file so the "keypair" value is set to the name of the keypair you used to launch the cluster and "key-pair-file" value is set to the full path to your keypair .pem file, and the permissions on the .pem file are set to og-rwx as shown in [To configure the permissions of the keypair file using Linux/Unix/Mac OS X \(p. 448\)](#), and you have OpenSSH installed on your machine, you can open an SSH connection to the master node by issuing the following command. This is a handy shortcut for frequent CLI users. In the example below, replace the red text with the cluster identifier of the cluster to open an SSH tunnel and use as a SOCKS server.

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

```
./elastic-mapreduce -j j-3L7WK3E07HO4H --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>.

After you've created an SSH tunnel to the master node, you can browse the websites hosted there using the text-based browser Lynx, or set up proxies in Firefox using the FoxyProxy add-on. This latter technique gives you full access to the graphical version of the web pages hosted locally on the master node. For more information, see [Configure FoxyProxy to View Websites Hosted on the Master Node \(p. 453\)](#).

If you are using a browser with a SOCKS proxy configured, as described in [Configure FoxyProxy to View Websites Hosted on the Master Node \(p. 453\)](#), you can use the browser to access any cluster launched in the same region as the one you used to create the SSH tunnel. This works because all of the master nodes you launch in a region share the same security group and thus are able to access each other.

To close an SSH tunnel using Linux/Unix/Mac OS X

- In the terminal, press `Ctrl+C`.

Configure FoxyProxy to View Websites Hosted on the Master Node

FoxyProxy is an add-on for Google Chrome, Firefox, and Internet Explorer that provides a set of proxy management tools. You can configure it to use a proxy server on URLs that match patterns corresponding to the domains used by the Amazon EC2 instances in your Amazon EMR cluster if you have created an SSH tunnel to the master node. For more information, see [Open an SSH Tunnel to the Master Node \(p. 452\)](#). You can configure FoxyProxy to use the SOCKS proxy you have created to connect to EC2 instances. This enables you to view the web interfaces available on the master node. For more information about the available web interfaces, see [Web Interfaces Hosted on the Master Node \(p. 450\)](#).

Note

The following tutorial uses FoxyProxy Standard version 2.9 in Chrome and FoxyProxy Standard version 3.6.2 in Firefox.

To install FoxyProxy in Firefox

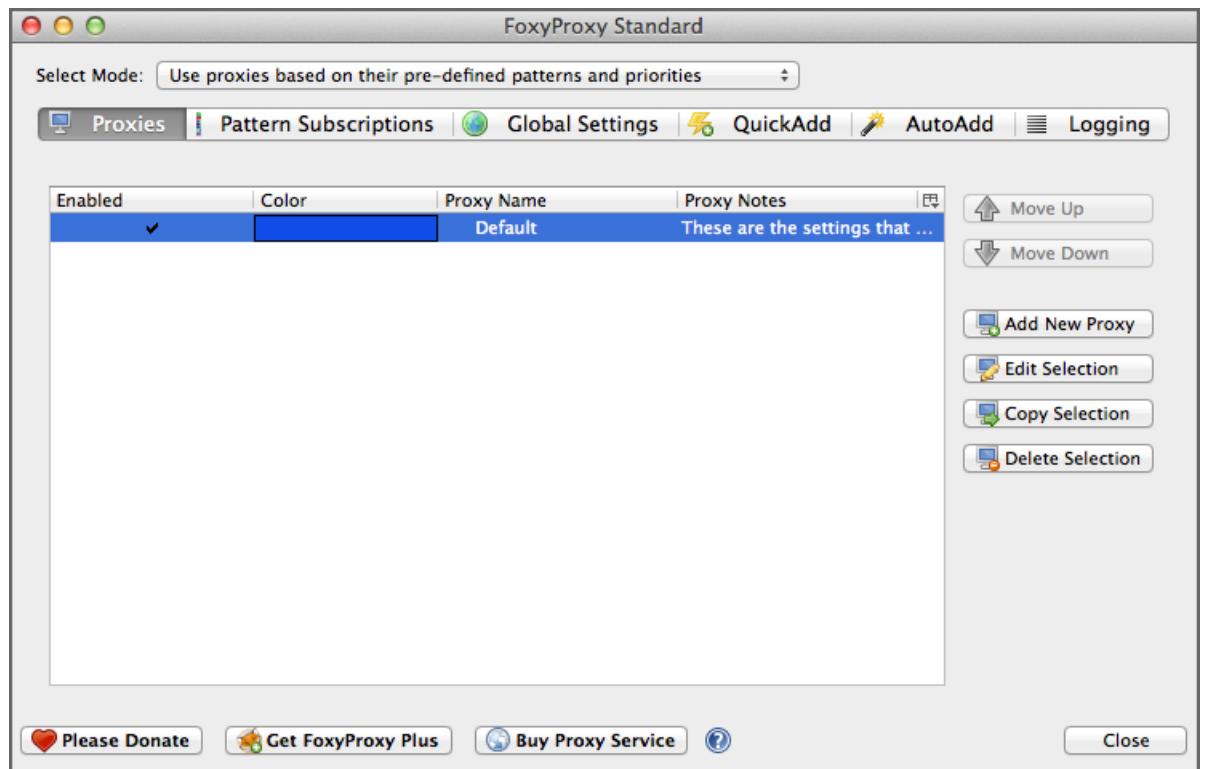
1. Download and install the standard version of FoxyProxy from <http://foxyproxy.mozdev.org/downloads.html>.
2. Restart your browser after installing FoxyProxy.

To configure FoxyProxy with Firefox

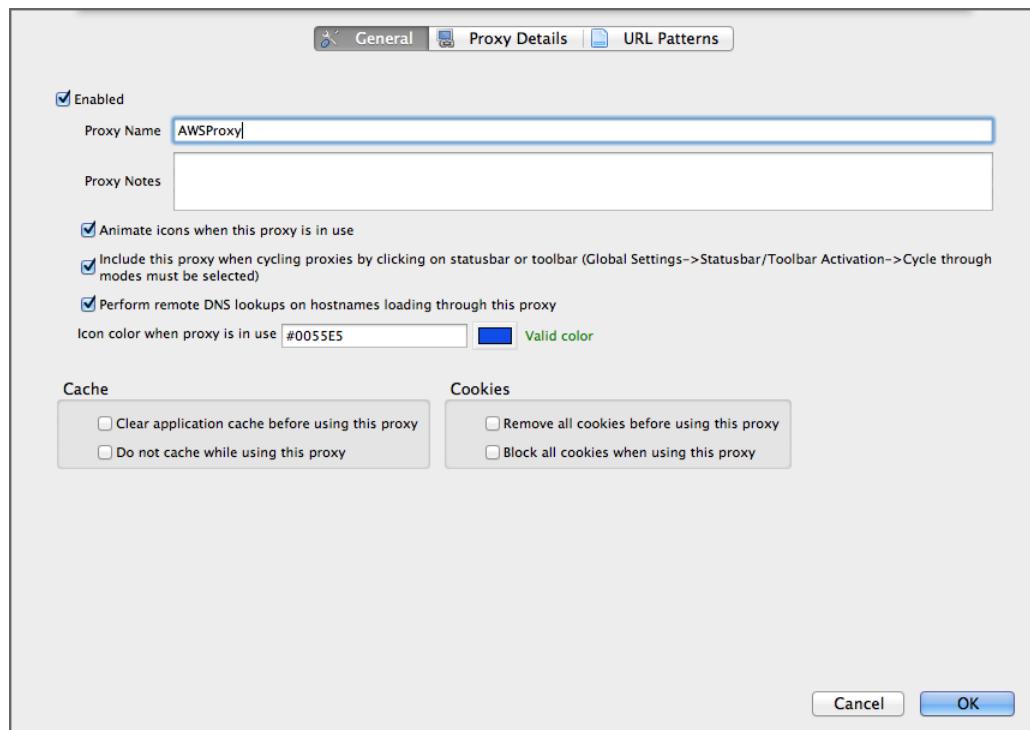
1. On the Firefox **Tools** menu, click **FoxyProxy Standard**, and then select **Options**.

FoxyProxy displays the **FoxyProxy Standard** window.

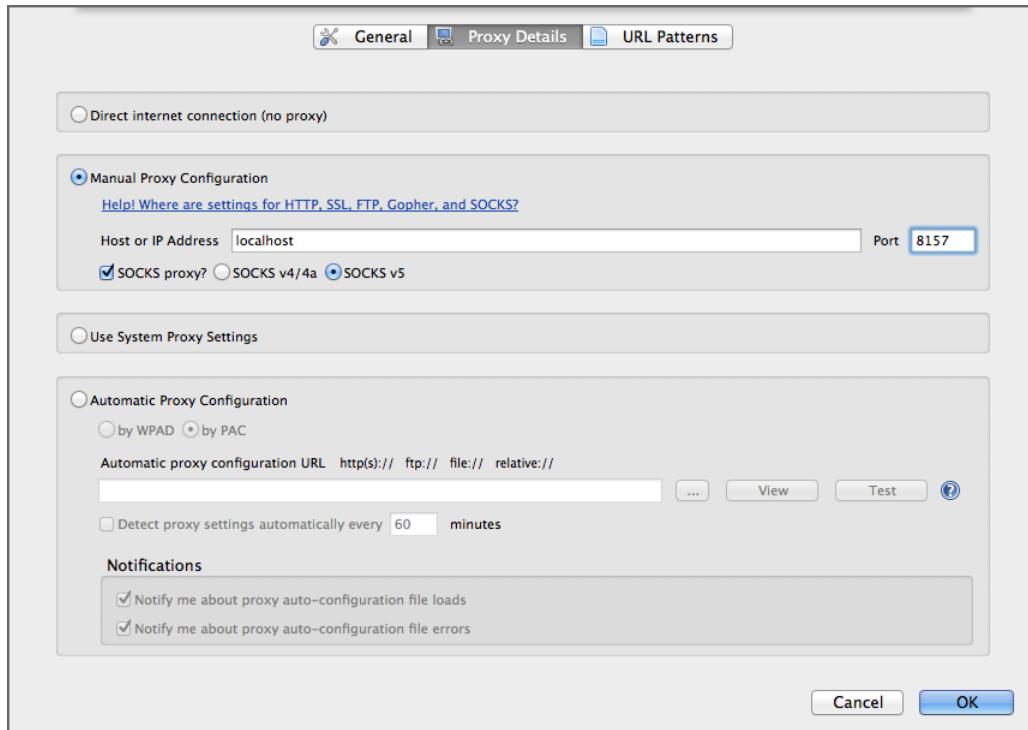
Amazon Elastic MapReduce Developer Guide
Configure FoxyProxy to View Websites Hosted on the
Master Node



2. Click **Add New Proxy**.
3. Click **General**.
4. Enter a proxy name and verify that **Perform remote DNS lookups on hostnames loading through this proxy** is selected.



5. Click **Proxy Details**.
6. a. Select **Manual Proxy Configuration** and enter the host name and port number of the host you ran the ssh command as the Hadoop user in step 1.
The SOCKS proxy (or SSH tunnel) is running on your desktop so enter `localhost` and port 8157.
b. Select the **SOCKS proxy?** check box.
c. Select **SOCKS v5**.

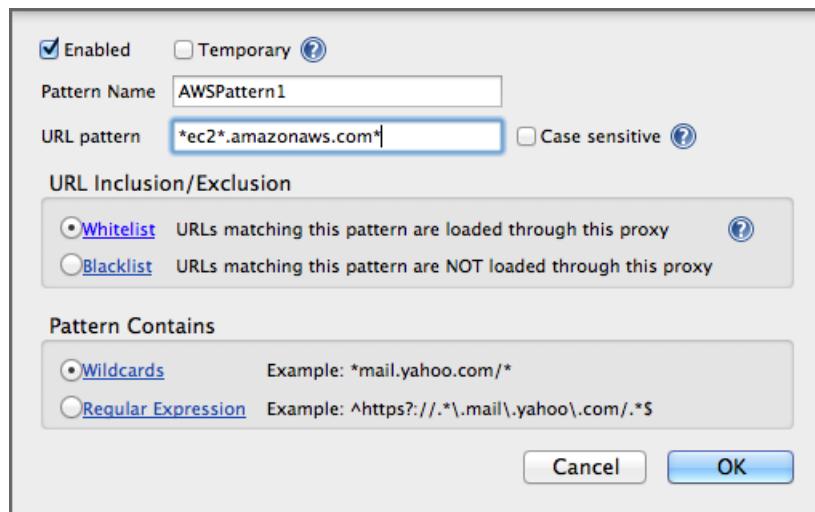


7. Click **URL Patterns**. Next you'll add URL patterns that cause URLs of the form `*ec2*.amazonaws.com`, `*ec2.internal`, and `*.compute.internal` to use the proxy.

Note

The `ec2` and `ec2.internal` patterns should not be enclosed with dots.

8. Click **Add New Pattern**.
9. a. Select the **Enabled** check box.
b. Enter a name in the **Pattern Name** box.
c. Enter the following URL pattern in the **URL pattern** box: `*ec2*.amazonaws.com`
d. Select the **Wildcards** option.
e. Click **OK**.

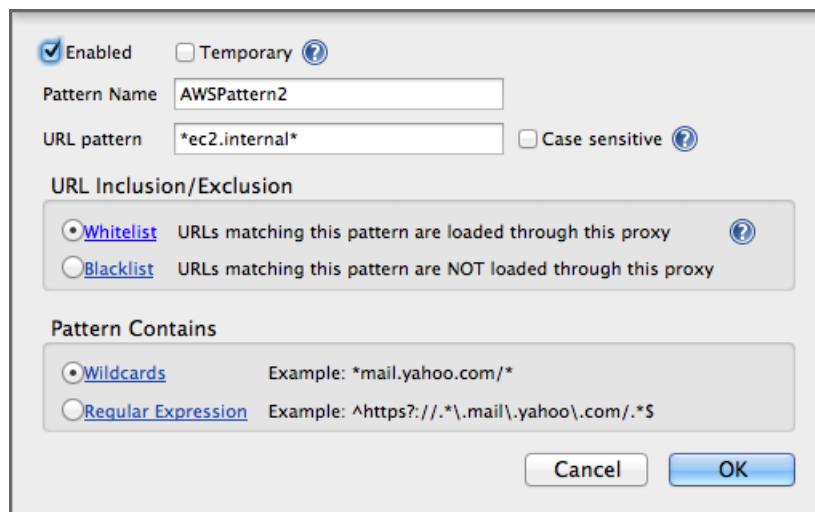


10. Select **Add New Pattern** again to add the second pattern.

Note

This step is for clusters in the `us-east-1` region, which have the machine name suffix `ec2.internal`.

11. a. Select the **Enabled** check box.
b. Enter a name in the **Pattern Name** box.
c. Enter the following URL pattern in the **URL pattern** box: `*ec2.internal*`
d. Select the **Wildcards** option.
e. Click **OK**.



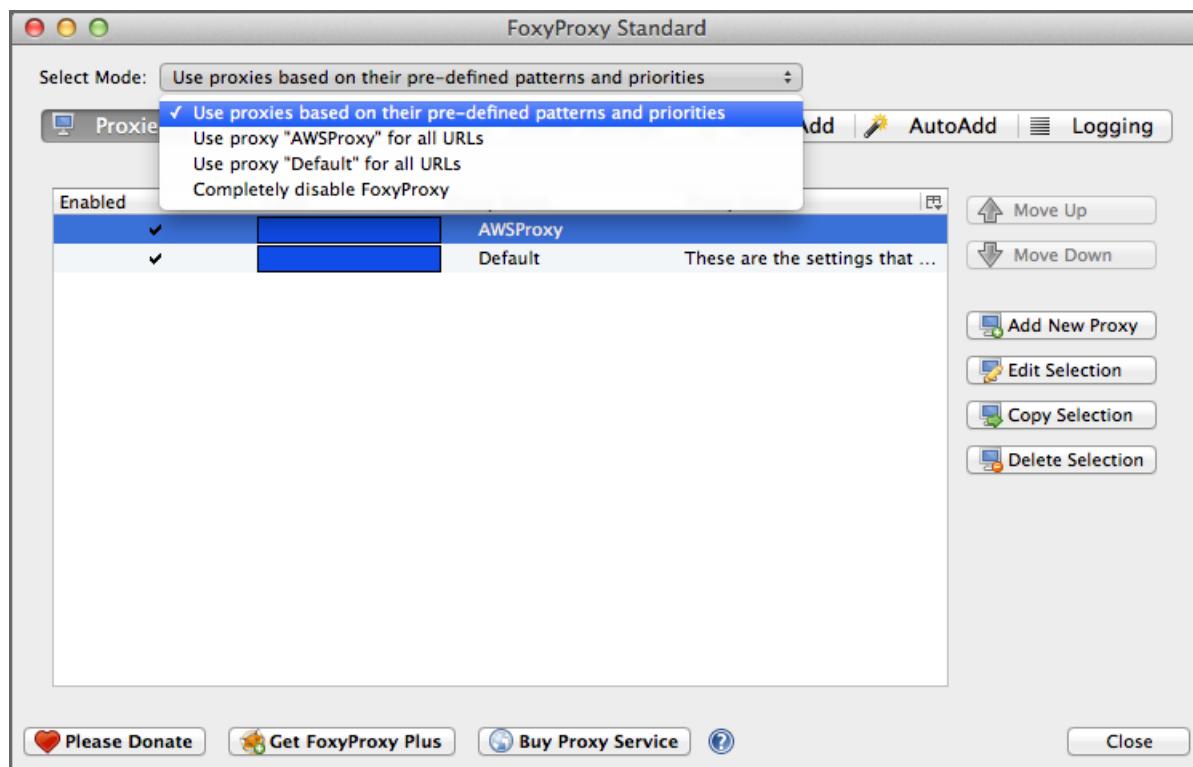
12. Select **Add New Pattern** again to add the third pattern.

Note

This step is for clusters in regions other than `us-east-1`, which have the machine name suffix `.compute.internal`.

13. a. Select the **Enabled** check box.
b. Enter a name in the **Pattern Name** box.

- c. Enter the following URL pattern in the **URL pattern** box: `*.compute.internal*`
 - d. Select the **Wildcards** option.
 - e. Click **OK**.
14. [Optional] Select **Add New Pattern** again to provide access to JobTracker log files.
15. a. Select the **Enabled** check box.
- b. Enter a name in the **Pattern Name** box.
- c. Enter the following URL pattern in the **URL pattern** box: `http://10.*`
- Note**
Use an alternate IP address filter if the `10.*` filter conflicts with your existing network address plan.
- d. Select the **Wildcards** option.
 - e. Click **OK**.
16. On the **FoxyProxy Options** pane, expand the **Select Mode** drop-down menu, select **Use proxies based on their predefined patterns and priorities**.



17. Click **Close**.

Now that you've configured FoxyProxy when you enter a URL that matches the pattern `*ec2*.amazon-aws.com*` into the Firefox browser, Firefox uses the proxy to connect to the master node of the cluster. You can now load the web pages listed in [Web Interfaces Hosted on the Master Node \(p. 450\)](#) in your browser using URLs such as the following, where the text in red is replaced by the public DNS name of the master node of your cluster.

<http://ec2-107-22-74-202.compute-1.amazonaws.com:9100>

Control Cluster Termination

The default behavior of an Amazon EMR cluster is to terminate when data processing is complete; that is, when no more steps are left to run. You can change this behavior when you launch the cluster by choosing a long-running cluster. For more information, see [Choose the Cluster Lifecycle: Long-Running or Transient \(p. 97\)](#). When you do so, you'll need to manually terminate the cluster when you're done.

Topics

- [Terminate a Cluster \(p. 458\)](#)
- [Protect a Cluster from Termination \(p. 459\)](#)

Terminate a Cluster

This section describes the methods to terminate 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. For more information, see [Terminating a Protected Cluster \(p. 463\)](#).

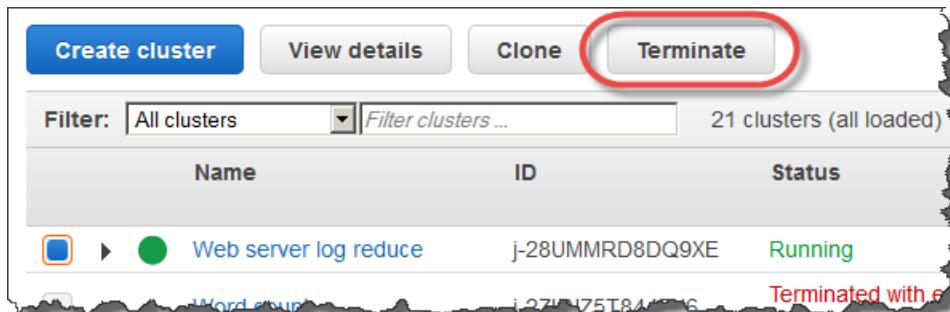
Depending on the configuration of the cluster, it may take up to 5-20 minutes for the cluster to completely terminate and release allocated resources, such as EC2 instances.

Amazon EMR Console

You can terminate a cluster using the Amazon EMR console.

To terminate a cluster

1. Sign in to the AWS Management Console and open the Amazon Elastic MapReduce console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Select the cluster to terminate.



3. Click **Terminate**.

4. Click **Yes, Terminate**.

Amazon EMR terminates the instances in the cluster and stops saving log data.

Using the CLI

To terminate a cluster, use the `--terminate` parameter and specify the cluster to terminate. The example that follows uses cluster `j-C019299B1X`.

To terminate a cluster

- In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).
 - Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --terminate JobFlowID
```

- Windows users:

```
ruby elastic-mapreduce --terminate JobFlowID
```

The response is similar to the following:

```
Terminated cluster JobFlowID
```

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](#).

Protect a Cluster from Termination

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.

By default, termination protection is disabled on clusters. 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.

When termination protection is enabled, you must explicitly remove termination protection from the cluster before you can terminate the cluster. With termination protection enabled, `TerminateJobFlows` can't

terminate the cluster and users can't 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.

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 job flow. If the job flow 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.

Note

Use cluster termination protection judiciously because it can lead to additional charges for the persistent EC2 instances.

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 EC2 Spot Instance from terminating when the Spot price rises above the maximum bid price. For more information about the behavior of EC2 Spot Instances in Amazon EMR, see [Lower Costs with Spot Instances \(Optional\) \(p. 37\)](#).

Termination Protection and Keep Alive

Enabling termination protection on a cluster is similar to enabling keep alive on a cluster (using the `--alive` argument in the CLI), but the protections each offers are different. Keep alive causes instances in a cluster to persist after the cluster has successfully completed, but still allows the cluster to be terminated by calls to `TerminateJobFlows` and errors. Termination protection allows the job to terminate after successful completion, but keeps it persistent in the case of user actions, errors, and `TerminateJobFlow` calls.

The following table compares the protections offered by termination protection and keep alive.

Protects against termination from...	Termination Protection	Keep Alive
Successful completion		✓
User actions	✓	

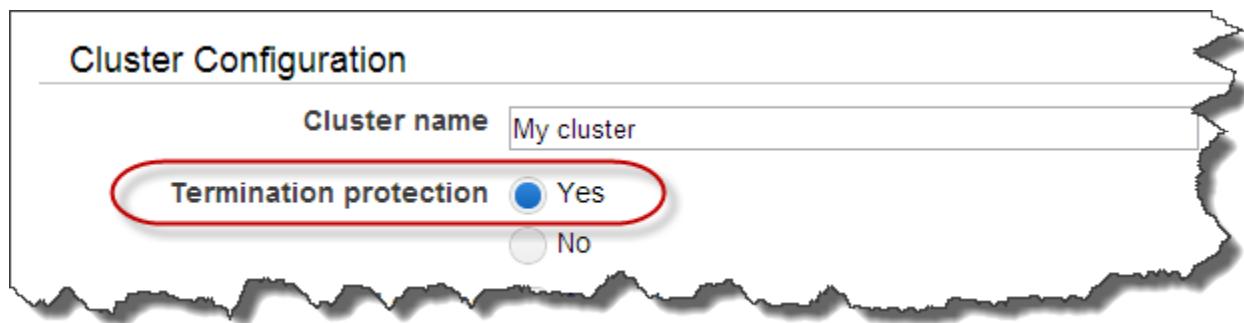
Protects against termination from...	Termination Protection	Keep Alive
TerminateJobFlows API	✓	
Errors	✓	

Protecting a New Cluster

You can specify that a new cluster be protected from termination during the cluster creation.

Launch a cluster with termination protection using the Amazon EMR console

1. Open the Amazon Elastic MapReduce console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Click **Create cluster**.
3. In the **Cluster Configuration** section, set the **Termination protection** switch to **Yes**.



4. Continue through the configuration sections, following the directions for the type of cluster you are launching. For more information, see [Plan an Amazon EMR Cluster \(p. 29\)](#).

Launch a cluster with termination protection using the CLI

- Specify `--with-termination-protection` during the cluster creation call. The following example shows setting termination protection on the WordCount sample application.

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

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 2 \
    --stream --arg "-files" --arg "s3://elasticmapreduce/samples/word \
    count/wordSplitter.py" \
    --input s3://elasticmapreduce/samples/wordcount/input \
    --output s3://myawsbucket/output/2014-01-16 --mapper wordSplitter.py
```

```
--reducer aggregate \
--with-termination-protection
```

- 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://myawsbucket/output/2014-01-16 --mapper wordSplitter.py --reducer aggregate --with-termination-protection
```

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 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 /
--with-termination-protection
```

- 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 --with-termination-protection
```

For more information about launching clusters using the CLI, see [Plan an Amazon EMR Cluster \(p. 29\)](#).

Protecting an Existing Cluster

You can add termination protection to an already running cluster using either the CLI or the API.

Note

You cannot currently add termination protection to a running cluster using the Amazon EMR console.

To enable termination protection for an existing cluster using the CLI

- 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, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- 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
```

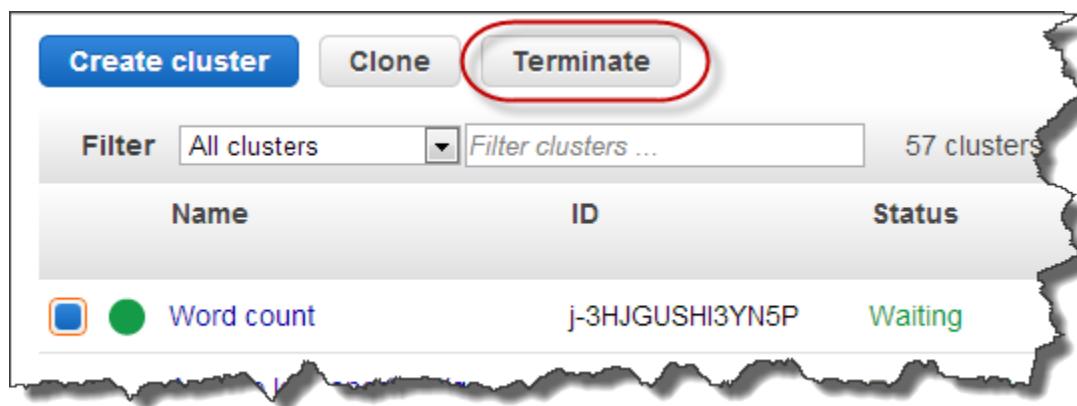
\

Terminating a Protected Cluster

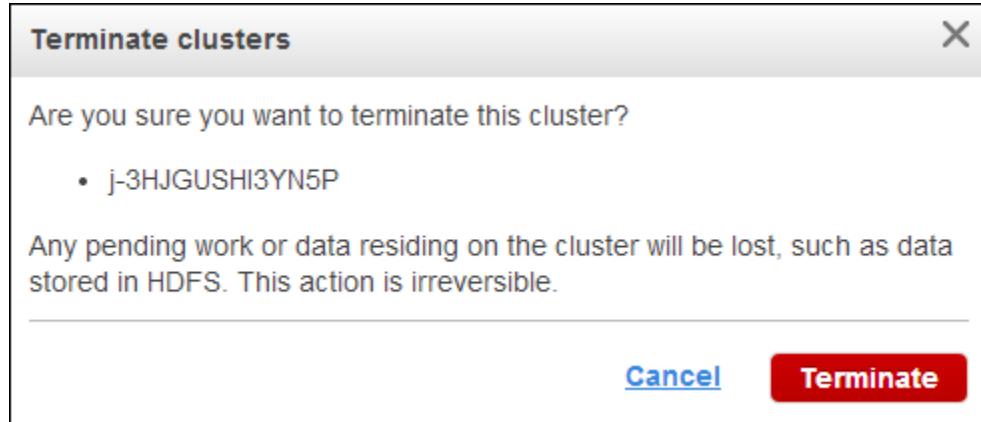
To terminate a protected cluster, you must first disable termination protection. After termination protection is disabled, you can terminate the cluster from the Amazon EMR console, CLI, or programmatically using the `TerminateJobFlows` API.

To terminate a cluster with termination protection set using the Amazon EMR console.

1. Sign in to the AWS Management Console and open the Amazon Elastic MapReduce console at <https://console.aws.amazon.com/elasticmapreduce/>.
2. Select the cluster to terminate.
3. Click **Terminate**.



4. Click **Terminate** on the confirmation dialog box, to confirm that you wish to disable termination protection and terminate the cluster.



To terminate a cluster with termination protection set using the CLI

1. Disable termination protection by setting the `--set-termination-protection` 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 specifying the cluster identifier of the cluster to terminate.

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --terminate JobFlowID
```

- Windows users:

```
ruby elastic-mapreduce --terminate JobFlowID
```

Resize a Running Cluster

Topics

- [Resize a Cluster Using the Console \(p. 465\)](#)
- [Parameters for Resizing Clusters \(p. 466\)](#)
- [Arrested State \(p. 469\)](#)
- [Legacy Clusters \(p. 470\)](#)
- [Library Files \(p. 471\)](#)

You can increase or decrease the number of nodes in a running cluster. 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 hold data to process in the Hadoop Distributed File System (HDFS), and task nodes, which do not contain HDFS. After a cluster is running, you can increase, but not decrease, the number of core nodes. Task nodes also run your Hadoop jobs. After a cluster is running, you can both increase or decrease the number of task nodes.

You can modify the size of a running cluster using either the console, CLI, or API.

Nodes within a cluster are managed by 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 a task instance group containing one or more task nodes.

Note

You must have at least one core node at cluster creation in order to resize. In other words, single node clusters cannot be resized.

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 flow 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 its slave nodes. Because all steps are guaranteed to run sequentially, you can specify the number of running slave nodes for any job flow step.

Resize a Cluster Using the Console

You can use the Amazon EMR console to resize a cluster while it is running.

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 nodes, and increase or decrease the number of instances for the instance group.

3. To add task nodes to a cluster that has none, click **Add Task Nodes** to choose the task node type, the number of task nodes, and whether the task nodes are spot instances.

Note

You can only increase the number of core nodes 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.

Parameters for Resizing Clusters

The Amazon EMR CLI provides parameters so you can control how you resize a running cluster.

Parameters to Increase or Decrease Nodes

You can increase or decrease the number of nodes in a running cluster. The parameters are listed in the following table.

Parameter	Description
--modify-instance-group <i>INSTANCE_GROUP_ID</i>	Modify an existing instance group.
--instance-count <i>INSTANCE_COUNT</i>	Set the count of nodes for an instance group. Note You are only allowed to increase the number of nodes in a core instance group. You can increase or decrease the number of nodes in a task instance group. Master instance groups can not be modified.

Parameters to Add an Instance Group to a Running Job Flow

You can add an instance group to your running cluster. The parameters are listed in the following table.

Parameter	Description
--add-instance-group <i>ROLE</i>	Add an instance group to an existing cluster. The role may be TASK only. Currently, Amazon Elastic MapReduce (Amazon EMR) does not permit adding core or master instance groups to a running cluster.
--instance-count <i>INSTANCE_COUNT</i>	Set the count of nodes for an instance group.
--instance-type <i>INSTANCE_TYPE</i>	Set the type of EC2 instance to create nodes for an instance group.

Parameters to Specify an Instance Group when Creating a Cluster

You can specify instance groups when you create a cluster. The parameters are listed in the following table.

Parameter	Description
--instance-group <i>TYPE</i>	Set the instance group type. A type is MASTER, CORE, or TASK
--instance-count <i>INSTANCE_COUNT</i>	Set the count of nodes for an instance group.
--instance-type <i>INSTANCE_TYPE</i>	Set the type of EC2 instance for nodes in an instance group.

The `--describe` command describes all instance groups and node types. If you run `elastic-mapreduce --jobflow JobFlowID --describe`, you see a section called `InstanceGroups`. You can see that your cluster contains a master instance group and, potentially, core and task instance groups.

The following CLI commands show how to display information about the instance groups of a running cluster. In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --jobflow JobFlowID --describe
```

- Windows users:

```
ruby elastic-mapreduce --jobflow JobFlowID --describe
```

This returns information about the cluster similar to the following:

```
"InstanceCount": 5,
"Placement": {
    "AvailabilityZone": "us-east-1a"
},
"SlaveInstanceType": "m1.small",
"HadoopVersion": "0.20",
"MasterPublicDnsName": null,
"KeepJobFlowAliveWhenNoSteps": true,
"InstanceGroups": [
    {
        "StartDate": null,
        "SpotPrice": null,
        "Name": "Master Instance Group",
        "InstanceRole": "MASTER",
        "EndDateTime": null,
        "LastStateChangeReason": "",
        "CreationDateTime": DateTimeStamp,
        "LaunchGroup": null,
        "InstanceGroupId": "InstanceGroupID",
        "State": "PROVISIONING",
        "Market": "ON_DEMAND",
        "ReadyDateTime": null,
        "InstanceType": "m1.small",
        "InstanceRunningCount": 0,
        "InstanceRequestCount": 1
    },
    {
        "StartDate": null,
        "SpotPrice": null,
        "Name": "Task Instance Group",
        "InstanceRole": "TASK",
        "EndDateTime": null,
        "LastStateChangeReason": "",
        "CreationDateTime": DateTimeStamp,
        "LaunchGroup": null,
        "InstanceGroupId": "InstanceGroupID",
        "State": "PROVISIONING",
        "Market": "ON_DEMAND",
        "ReadyDateTime": null,
        "InstanceType": "m1.small",
        "InstanceRunningCount": 0,
        "InstanceRequestCount": 2
    },
    {
        "StartDate": null,
        "SpotPrice": null,
        "Name": "Core Instance Group",
        "InstanceRole": "CORE",
        "EndDateTime": null,
        "LastStateChangeReason": "",
        "CreationDateTime": DateTimeStamp,
        "LaunchGroup": null,
        "InstanceGroupId": "InstanceGroupID",
        "State": "PROVISIONING",
        "Market": "ON_DEMAND",
        "ReadyDateTime": null,
        "InstanceType": "m1.small",
        "InstanceRunningCount": 0,
```

```
        "InstanceRequestCount": 2
    }
],
"MasterInstanceType": "m1.small"
},
"bootstrapActions": [],
"JobFlowId": "JobFlowID"
}
]
```

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 provision 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.

The command `--describe` returns all instance groups and node types, and so you can see the state of the instance groups for the cluster. If Amazon EMR detects any kind of fault with an instance group, it changes the group's state to `ARRESTED`.

Use the `--modify-instance-group` command to reset a cluster in the `ARRESTED` state.

Modifying the instance group instructs Amazon EMR to attempt to provision nodes again. No running nodes are restarted or terminated.

To reset a cluster in an arrested state

- Enter the `--modify-instance-group` command as follows:

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- 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>/<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.

Legacy Clusters

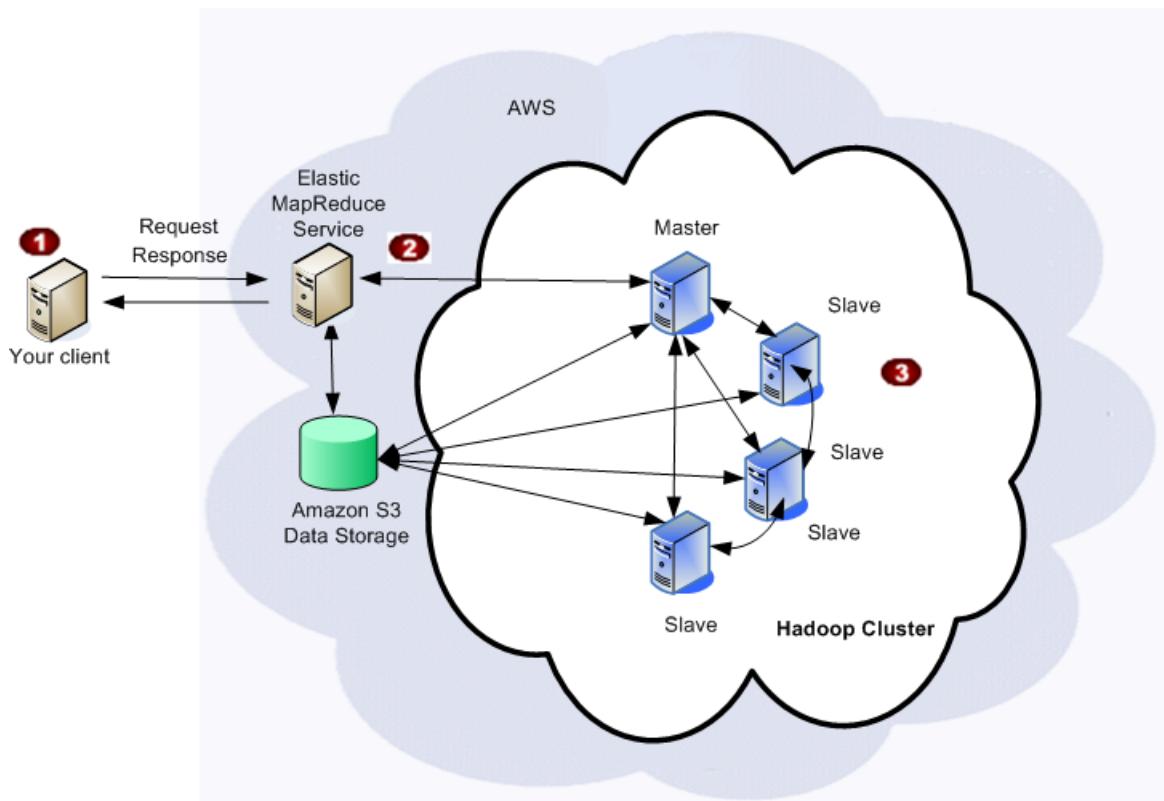
Before October 2010, Amazon EMR did not have the concept of *instance groups*. Clusters developed for Amazon EMR that were built before the option to resize running clusters was available are considered *legacy clusters*. Previously, the Amazon EMR architecture did not use instance groups to manage nodes 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

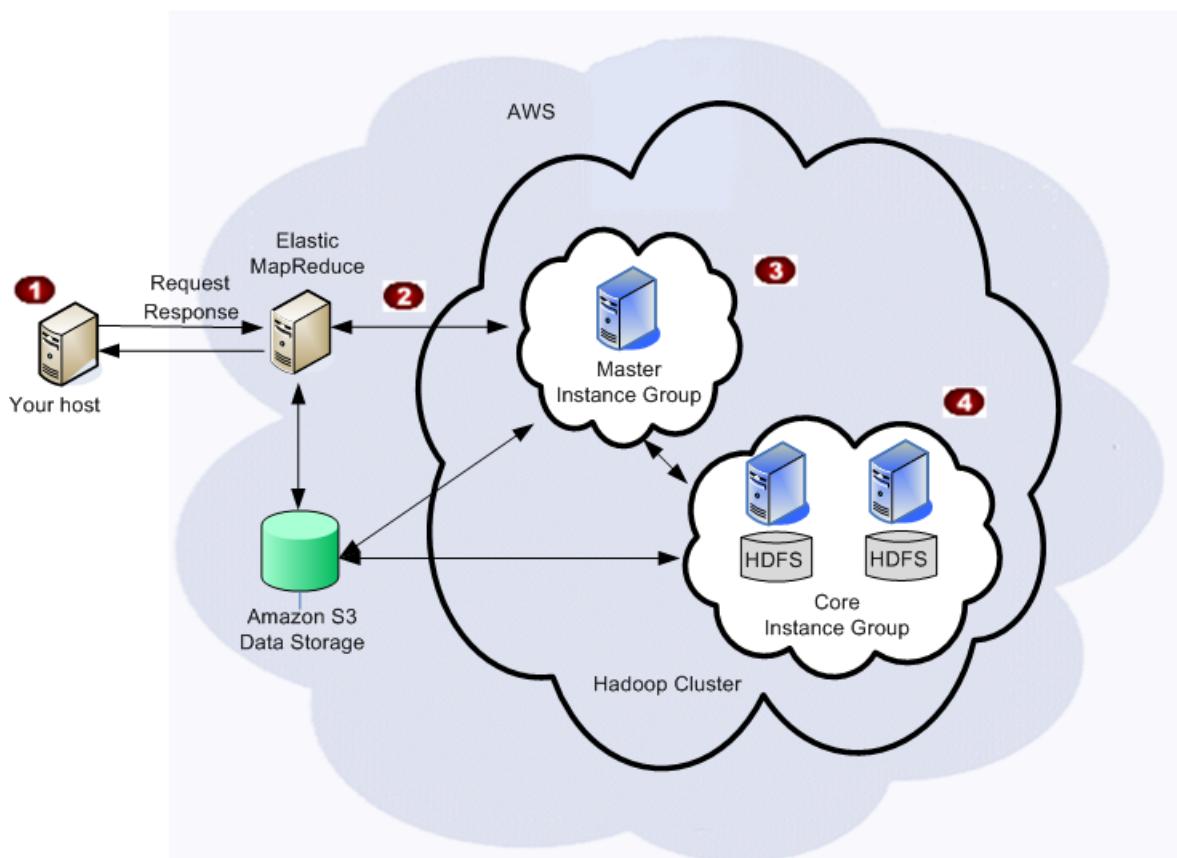
1	A legacy cluster launches and a request is sent to Amazon EMR to start the cluster.
2	Amazon EMR creates a Hadoop cluster.
3	The legacy cluster runs on a cluster consisting of a single master node and the specified number of slave nodes.

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.

Library Files

Amazon EMR provides a library file containing a JAR file to create a cluster step programmatically instead of directly through the CLI.

The JAR file to programmatically resize a running cluster is available at `s3://elasticmapreduce/libs/resize-job-flow/0.1/resize-job-flow.jar` and supports the optional arguments described in the following table.

Option	Description
<code>--help</code>	List all help information.
<code>--modify-instance-group <i>ROLE/InstanceGroupID</i></code>	Apply changes to the named instance group, specified by either role or Instance Group ID. Instance group roles: MASTER, CORE, or TASK.
<code>--set-instance-count <COUNT></code>	Change the number of nodes of the named instance group.
<code>--add-instance-group <ROLE></code>	Apply operations to the named instance group. Instance group roles: TASK. Currently, Amazon EMR does not permit adding core or master instance groups to a running cluster.
<code>--instance-count <COUNT></code>	Specify the number of nodes for the named instance group.
<code>--instance-type <TYPE></code>	Specify the type of EC2 instances used to create nodes in the new instance group.
<code>--no-wait</code>	The cluster continues in the RUNNING state after the step makes a request to create or resize an instance group.
<code>--on-failure <i>STATE</i></code>	Step state if one of the resizing actions fails: FAIL or CONTINUE.
<code>--on-arrested <STATE></code>	Cluster state if an instance group enters the ARRESTED state: FAIL, WAIT, or CONTINUE.

The JAR file is configured to write to `stderr`. Only error and fatal messages are reported. The JAR file includes the source code.

The cluster step looks similar to:

```
s3://elasticmapreduce/libs/resize-job-flow/0.1/resize-job-flow.jar \
--add-instance-group task --instance-type InstanceType --instance-count 10
```

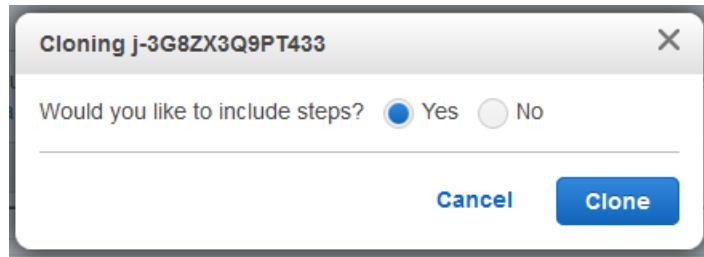
For more information about how to add a cluster step, see [Add Steps to a Cluster \(p. 473\)](#).

Cloning a Cluster Using the Console

You can use the Amazon EMR console to clone a cluster, which makes a copy of the configuration of the original cluster to use as the basis for a new cluster.

To clone a cluster using the console

1. From the **Cluster List** page, click a cluster to clone.
2. At the top of the **Cluster Details** page, click **Clone**. The following dialog box appears:



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 process steps.

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**.

Add Steps to a Cluster

Topics

- [Wait for Steps to Complete \(p. 474\)](#)
- [Add More than 256 Steps to a Cluster \(p. 475\)](#)

This section describes the methods for adding steps to a cluster.

You can add steps to a running cluster only if you set the `KeepJobFlowAliveWhenNoSteps` parameter to `True` when you create the cluster. This value keeps the Hadoop cluster engaged even after the completion of a cluster.

The following procedure creates a simple cluster and then adds a step to the cluster.

To add a step to a cluster using the CLI

1. Create a cluster:

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --alive
```

- Windows users:

```
ruby elastic-mapreduce --create --alive
```

The `--alive` parameter keeps the cluster running even when all steps have been completed, unless you explicitly terminate it.

The output looks similar to the following.

```
Created cluster JobFlowID
```

2. Add a step:

- 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://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
```

This command runs an example cluster step that downloads and runs the 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 `--main-class` option.

Note

The maximum number of steps allowed in a cluster is 256. The debugging option uses additional steps to function, so it can exceed your step limit quickly. For more information about how to overcome this limitation, see [Add More than 256 Steps to a Cluster \(p. 475\)](#).

Wait for Steps to Complete

When you submit steps to a cluster using the command line interface (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.

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 will be 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. This is illustrated in the example that follows.

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, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- 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://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

ruby elastic-mapreduce -j JobFlowID --wait-for-steps
```

Add More than 256 Steps to a Cluster

Amazon EMR currently limits the number of steps in a cluster to 256. If your cluster is long-running (such as a Hive data warehouse) or complex, you may require more than 256 steps to process your data. The debugging option uses additional steps to function, so it can exceed your step limit quickly.

You can employ several methods to get around this limitation:

1. Have each step submit several jobs to Hadoop. This does not allow you unlimited steps, 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 clusters using SSH.
4. Manually connect to the master node using SSH and submit clusters.

You can add more steps to a cluster by using the SSH shell to connect to the master node and submitting queries directly to the software running on the master node, such as Hive and Hadoop.

You can SSH directly into the master node using a conventional SSH connection, as outlined in [View Log Files \(p. 418\)](#). You can also use the `--ssh` command line argument to pass queries in and save yourself the process of establishing a new SSH connection.

To manually submit steps to Hadoop on the master node

- From a terminal or command-line window, call the CLI client, specifying the `--ssh` parameter, and set its value to the command you want to run on the master node. The CLI uses its connection to the master node to run the command.

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- 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"
```

The preceding example uses the `--scp` parameter to copy the JAR file `myjar.jar` from your local directory to the master node of cluster `JobFlowID`. The example uses the `--ssh` parameter to command the copy of Hadoop running on the master node to run `myjar.jar`.

To manually submit queries to Hive on the master node

- If Hive is not already installed, use the following command to install it.

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --jobflow JobFlowID --hive-interactive
```

- Windows users:

```
ruby elastic-mapreduce --jobflow JobFlowID --hive-interactive
```

- Create a Hive script file containing the query or command to run. The following example script creates two tables, `aTable` and `anotherTable`, and copies the contents of one table to another, 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. Call the CLI client, specifying the `--ssh` parameter, and set its value to a Hive script containing the command you want to run on the master node. The CLI uses its connection to the master node and your `.pem` credentials file to run the command.

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- 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"
```

The preceding example connects to Hive on the master node of the `JobFlowID` cluster and runs the query contained in the script file `my-hive.q`.

To manually submit tasks based on Python files to Hadoop while connected using SSH

- Use the Hadoop streaming jar, as shown in the example below.

```
hadoop jar /home/hadoop/contrib/streaming/hadoop-streaming.jar
  -input s3n://elasticmapreduce/samples/wordcount/input \
  -output hdfs:///rubbish/1 \
  -mapper s3n://elasticmapreduce/samples/wordcount/wordSplitter.py \
  -reducer aggregate
```

Associate an Elastic IP Address with a Cluster

Topics

- [Assign an Elastic IP Address to a Cluster \(p. 478\)](#)
- [View Allocated Elastic IP Addresses using Amazon EC2 \(p. 480\)](#)
- [Manage Elastic IP Addresses using Amazon EC2 \(p. 480\)](#)

Elastic IP addresses are static IP addresses designed for dynamic cloud computing. An Elastic IP address is associated with your account, not a particular instance. You control the addresses associated with your account until you choose to explicitly release them.

You can associate one Elastic IP address with only one cluster at a time. To ensure that our customers are efficiently using Elastic IP addresses, we impose a small hourly charge when IP addresses associated with your account are not mapped to a cluster or EC2 instance. When Elastic IP addresses are mapped to an instance, they are free of charge.

For more information about using IP addresses in AWS, see [Using Elastic IP Addresses](#) section in the *Amazon Elastic Compute Cloud User Guide*.

To help you manage your resources, you can change the dynamically assigned IP address of the master node of your running cluster to a static Elastic IP address. Elastic IP addresses enable you to quickly remap the dynamically assigned IP address of the cluster's master node to a static IP address. An Elastic IP address is associated with your AWS account, not with a particular cluster. You control your Elastic IP address until you choose to explicitly release it. For more information about Elastic IP addresses, see [Using Instance IP Addresses](#) in the *Amazon Elastic Compute Cloud User Guide*.

By default, the master node of your running cluster is assigned a dynamic IP address that is reachable from the Internet. The dynamic IP address is associated with the master node of your running cluster until it is stopped, terminated, or replaced with an Elastic IP address. When a cluster with an Elastic IP address is stopped or terminated, the Elastic IP address is not released and remains associated with your AWS account.

Assign an Elastic IP Address to a Cluster

Using the Amazon EMR CLI, you can allocate an Elastic IP address and assign it to either a new or running cluster. Amazon EMR does not support assignment of Elastic IP addresses from the Amazon EMR console or through the Amazon EMR API.

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.

For information about how to create a cluster using the CLI, see [Plan an Amazon EMR Cluster \(p. 29\)](#).

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.

For information about creating a cluster, see [Plan an Amazon EMR Cluster \(p. 29\)](#).

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, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- 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, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#). 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 job_flow_ID --eip
```

- Windows users:

```
ruby elastic-mapreduce job_flow_ID --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:

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce job_flow_ID --eip
Elastic_IP
```

- Windows users:

```
ruby elastic-mapreduce job_flow_ID --eip  
      Elastic_IP
```

You have successfully assigned an Elastic IP address to your cluster.

View Allocated Elastic IP Addresses using Amazon EC2

After you have allocated an Elastic IP address, you can reuse it on other clusters. For more information about how to identify your currently allocated IP addresses, see [Using Elastic IP Addresses](#) in the *Amazon Elastic Compute Cloud User Guide*.

Note

By default, each AWS customer has a limit of five Elastic IP addresses that can be associated with their account. If you would like to increase this limit, please submit a [Request to Increase Elastic IP Address Limit](#) (https://aws.amazon.com/support/createCase?type=service_limit_increase&serviceLimitIncreaseType=elastic-ips) to increase your maximum number of Elastic IP addresses.

Manage Elastic IP Addresses using Amazon EC2

Amazon EC2 allows you to manage your Elastic IP addresses from the Amazon EC2 console, the Amazon EC2 command line interface, and the Amazon EC2 API.

For more information about using Amazon EC2 to create and manage your Elastic IP addresses, see [Using Elastic IP Addresses](#) in the *Amazon Elastic Compute Cloud User Guide*.

Automate Recurring Clusters with AWS Data Pipeline

AWS Data Pipeline is a service that automates the movement and transformation of data. You can use it to schedule moving input data into Amazon S3 and to schedule launching clusters to process that data. For example, consider the case where you have a web server recording traffic logs. If you want to run a weekly cluster to analyze the traffic data, you can use AWS Data Pipeline to schedule those clusters. AWS Data Pipeline is a data-driven workflow, so that one task (launching the cluster) can be dependent on another task (moving the input data to Amazon S3). It also has robust retry functionality.

For more information about AWS Data Pipeline, see the [AWS Data Pipeline Developer Guide](#), especially the tutorials regarding Amazon EMR:

- [Tutorial: Launch an Amazon EMR Job Flow](#)
- [Getting Started: Process Web Logs with AWS Data Pipeline, Amazon EMR, and Hive](#)
- [Tutorial: Amazon DynamoDB Import and Export Using AWS Data Pipeline](#)

Troubleshoot a Cluster

A cluster hosted by Amazon EMR runs in a complex ecosystem made up of several types of open-source software, custom application code, and Amazon Web Services. An issue in any of these parts can cause the cluster to fail or take longer than expected to complete. The following topics will help you figure out what has gone wrong in your cluster and give you suggestions on how to fix it.

Topics

- [What Tools are Available for Troubleshooting? \(p. 481\)](#)
- [Troubleshoot a Failed Cluster \(p. 483\)](#)
- [Troubleshoot a Slow Cluster \(p. 487\)](#)
- [Common Errors in Amazon EMR \(p. 493\)](#)

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 \(Optional\) \(p. 133\)](#) and [Step 5: Test the Cluster Step by Step \(p. 486\)](#).

What Tools are Available for Troubleshooting?

There are several tools you can use to gather information about your cluster to help determine what went wrong. Some require that you initialize them when you launch the cluster; others are available for every cluster.

Topics

- [Tools to Display Cluster Details \(p. 481\)](#)
- [Tools to View Log Files \(p. 482\)](#)
- [Tools to Monitor Cluster Performance \(p. 482\)](#)

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 \(p. 407\)](#).

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 \(Optional\) \(p. 133\)](#).

Log Files on the Master Node

Every cluster publishes logs files to the `/mnt/var/log/` directory on the master node. These log files are only available while the cluster is running.

Log Files Archived to Amazon S3

If you launch the cluster and specify an Amazon S3 log path, the cluster copies the log files stored in `/mnt/var/log/` on the master node to Amazon S3 in 5-minute intervals. This ensures that you have access to the log files even after the cluster is terminated. Because the files are archived in 5-minute intervals, the last few minutes of an suddenly terminated cluster may not be available.

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 on the Cluster \(Hadoop 2.x\) \(p. 411\)](#).

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 \(p. 423\)](#).

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 \(p. 438\)](#).

Troubleshoot a Failed Cluster

This section walks you through the process of troubleshooting a cluster that has failed. This means that the cluster terminated with an error code. If the cluster is still running, but is taking a long time to return results, see [Troubleshoot a Slow Cluster \(p. 487\)](#) instead.

Topics

- [Step 1: Gather Data About the Issue \(p. 483\)](#)
- [Step 2: Check the Environment \(p. 484\)](#)
- [Step 3: Look at the Last State Change \(p. 485\)](#)
- [Step 4: Examine the Log Files \(p. 485\)](#)
- [Step 5: Test the Cluster Step by Step \(p. 486\)](#)

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 \(p. 407\)](#).

- 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 \(p. 56\)](#).

- 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 \(p. 484\)](#)
- [Check Usage Limits \(p. 484\)](#)
- [Check the AMI Version \(p. 484\)](#)
- [Check the Amazon VPC Subnet Configuration \(p. 485\)](#)

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 \(p. 56\)](#).

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 a Amazon VPC Subnet for the Cluster \(Optional\) \(p. 137\)](#). 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 \(p. 407\)](#).

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 \(p. 418\)](#).

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 \(p. 97\)](#) and [Protect a Cluster from Termination \(p. 459\)](#).
2. Submit a step to the cluster. For more information, see [Add Steps 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 \(p. 485\)](#). 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 [Protect a Cluster from Termination \(p. 459\)](#).

Troubleshoot a Slow Cluster

This section walks you through the process of troubleshooting a cluster that is still running, but is taking a long time to return results. For more information about what to do if the cluster has terminated with an error code, see [Troubleshoot a Failed Cluster \(p. 483\)](#)

Amazon EMR enables you to specify the number and kind of virtual server 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 virtual servers in the cluster. For more information, see [Choose the Number and Type of Virtual Servers \(p. 33\)](#).

The following topics walk you through the process of identifying alternative causes of a slow cluster.

Topics

- [Step 1: Gather Data About the Issue \(p. 487\)](#)
- [Step 2: Check the Environment \(p. 488\)](#)
- [Step 3: Examine the Log Files \(p. 489\)](#)
- [Step 4: Check Cluster and Instance Health \(p. 490\)](#)
- [Step 5: Check for Arrested Groups \(p. 491\)](#)
- [Step 6: Review Configuration Settings \(p. 491\)](#)
- [Step 7: Examine Input Data \(p. 493\)](#)

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 \(p. 407\)](#).

- 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 \(p. 56\)](#).
- Type and number of EC2 instances specified for the master, core, and task nodes.

Step 2: Check the Environment

Topics

- [Check for Service Outages \(p. 488\)](#)
- [Check Usage Limits \(p. 488\)](#)
- [Check the AMI Version \(p. 488\)](#)
- [Check the Amazon VPC Subnet Configuration \(p. 489\)](#)
- [Restart the Cluster \(p. 489\)](#)

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 \(p. 56\)](#).

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 a Amazon VPC Subnet for the Cluster \(Optional\) \(p. 137\)](#). 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 \(p. 418\)](#).

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 virtual server 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 \(p. 464\)](#).
- **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 \(p. 464\)](#).

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 virtual server 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 \(p. 423\)](#).

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 \(p. 438\)](#).

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 on the Cluster \(Hadoop 2.x\) \(p. 411\)](#).

- 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 virtual server 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 \(p. 422\)](#).

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 \(p. 469\)](#).

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 virtual server instance type you specify for the cluster. For more information, see [Hadoop Configuration Reference \(p. 515\)](#). You can modify the configuration settings from the default values using a bootstrap action or by specifying new values in job execution parameters. For more information, see [Create Bootstrap Actions to Install Additional Software \(Optional\) \(p. 87\)](#) and [Configuration of](#)

[hadoop-user-env.sh \(p. 519\)](#). To determine whether a bootstrap action changed the configuration settings, check the bootstrap action logs.

Amazon EMR logs the Hadoop settings used to execute each job. The log data is stored in a file named `job_job-id_conf.xml` under the `/mnt/var/log/hadoop/history/` directory of the master node, where `job-id` is replaced by the identifier of the job. If you've enabled log archiving, this data is copied to Amazon S3 in the `logs/date/jobflow-id/jobs` folder, where `date` is the date the job ran, and `jobflow-id` is the identifier of the cluster.

The following Hadoop job configuration settings are especially useful for investigating performance issues. For more information about the Hadoop configuration settings and how they affect the behavior of Hadoop, go to <http://hadoop.apache.org/docs/>.

Configuration Setting	Description
<code>dfs.replication</code>	The number of HDFS nodes to which a single block (like the hard drive block) is copied to in order to produce a RAID-like environment. Determines the number of HDFS nodes which contain a copy of the block.
<code>io.sort.mb</code>	Total memory available for sorting. This value should be 10x <code>io.sort.factor</code> . This setting can also be used to calculate total memory used by task node by figuring <code>io.sort.mb</code> multiplied by <code>mapred.tasktracker.ap.tasks.maximum</code> .
<code>io.sort.spill.percent</code>	Used during sort, at which point the disk will start to be used because the allotted sort memory is getting full.
<code>mapred.child.java.opts</code>	Deprecated. Use <code>mapred.map.child.java.opts</code> and <code>mapred.reduce.child.java.opts</code> instead. The Java options TaskTracker uses when launching a JVM for a task to execute within. A common parameter is “ <code>-Xmx</code> ” for setting max memory size.
<code>mapred.map.child.java.opts</code>	The Java options TaskTracker uses when launching a JVM for a map task to execute within. A common parameter is “ <code>-Xmx</code> ” for setting max memory heap size.
<code>mapred.map.tasks.speculative.execution</code>	Determines whether map task attempts of the same task may be launched in parallel.
<code>mapred.reduce.tasks.speculative.execution</code>	Determines whether reduce task attempts of the same task may be launched in parallel.
<code>mapred.map.max.attempts</code>	The maximum number of times a map task can be attempted. If all fail, then the map task is marked as failed.
<code>mapred.reduce.child.java.opts</code>	The Java options TaskTracker uses when launching a JVM for a reduce task to execute within. A common parameter is “ <code>-Xmx</code> ” for setting max memory heap size.
<code>mapred.reduce.max.attempts</code>	The maximum number of times a reduce task can be attempted. If all fail, then the map task is marked as failed.
<code>mapred.reduce.slowstart.completed.maps</code>	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.
<code>mapred.reuse.jvm.num.tasks</code>	A task runs within a single JVM. Specifies how many tasks may reuse the same JVM.

Configuration Setting	Description
mapred.tasktracker.map.tasks.maximum	The max amount of tasks that can execute in parallel per task node during mapping.
mapred.tasktracker.reduce.tasks.maximum	The max amount of tasks that can execute in parallel per task node during reducing.

If your cluster tasks are memory-intensive, you can enhance performance by using fewer tasks per core node and reducing your job tracker heap size.

Step 7: Examine Input Data

Look at your input data. Is it distributed evenly among your key values? If your data is heavily skewed towards one or few key values, the processing load may be mapped to a small number of nodes, while other nodes idle. This imbalanced distribution of work can result in slower processing times.

An example of an imbalanced data set would be running a cluster to alphabetize words, but having a data set that contained only words beginning with the letter "a". When the work was mapped out, the node processing values beginning with "a" would be overwhelmed, while nodes processing words beginning with other letters would go idle.

Common Errors in Amazon EMR

There are many reasons why a cluster might fail or be slow in processing data. The following sections list the most common issues and suggestions for fixing them.

Topics

- [Input and Output Errors \(p. 493\)](#)
- [Permissions Errors \(p. 496\)](#)
- [Memory Errors \(p. 497\)](#)
- [Resource Errors \(p. 498\)](#)
- [Streaming Cluster Errors \(p. 501\)](#)
- [Custom JAR Cluster Errors \(p. 502\)](#)
- [Hive Cluster Errors \(p. 503\)](#)
- [VPC Errors \(p. 504\)](#)
- [GovCloud-related Errors \(p. 505\)](#)

Input and Output Errors

The following errors are common in cluster input and output operations.

Topics

- [Does your path to Amazon Simple Storage Service \(Amazon S3\) have at least three slashes? \(p. 494\)](#)
- [Are you trying to recursively traverse input directories? \(p. 494\)](#)
- [Does your output directory already exist? \(p. 494\)](#)
- [Are you trying to specify a resource using an HTTP URL? \(p. 494\)](#)
- [Are you referencing an Amazon S3 bucket using an invalid name format? \(p. 494\)](#)
- [Are you using a custom jar when running DistCp? \(p. 494\)](#)

- Are you experiencing trouble loading data to or from Amazon S3? (p. 495)

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 \(p. 99\)](#).

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 using a custom jar when running DistCp?

You cannot run DistCp by specifying a JAR residing on the AMI. Instead, you should use the samples/distcp/distcp.jar file in the elasticmapreduce Amazon S3 bucket. The following example shows how to call the Amazon EMR version of DistCp. Replace j-ABABABABABAB with the identifier of your cluster.

Note

DistCp is deprecated on Amazon EMR, we recommend that you use S3DistCp instead. For more information about S3DistCp, see [Distributed Copy Using S3DistCp \(p. 355\)](#).

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --jobflow j-ABABABABABAB \
--jar s3n://elasticmapreduce/samples/distcp/distcp.jar \
--arg s3n://elasticmapreduce/samples/wordcount/input \
--arg hdfs:///samples/wordcount/input
```

- Windows users:

```
ruby elastic-mapreduce --jobflow j-ABABABABABAB --jar s3n://elasticmapre
duce/samples/distcp/distcp.jar --arg s3n://elasticmapreduce/samples/word
count/input --arg hdfs:///samples/wordcount/input
```

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 a Machine Image \(p. 51\)](#).
- 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 \(p. 355\)](#).
- 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? \(p. 503\)](#).

For additional information, see [Amazon S3 Error Best Practices](#) in the *Amazon Simple Storage Service Developer Guide*.

Permissions Errors

The following errors are common when using permissions or credentials.

Topics

- [Are you passing the correct credentials into SSH? \(p. 496\)](#)
- [If you are using IAM, do you have the proper Amazon EC2 policies set? \(p. 497\)](#)

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

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 \(p. 122\)](#).

Memory Errors

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 Na-

meNodes 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

The following errors are commonly caused by constrained resources on the cluster.

Topics

- [Do you have enough HDFS space for your cluster? \(p. 498\)](#)
- [Have your previous clusters finished terminating? \(p. 498\)](#)
- [Are you seeing "Too many fetch-failures" errors? \(p. 499\)](#)
- [Are you seeing "File could only be replicated to 0 nodes instead of 1" errors? \(p. 500\)](#)
- [Are your TaskTracker nodes being blacklisted? \(p. 500\)](#)

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 Elastic Compute Cloud User Guide*.

The other factor that can affect the amount of HDFS space available is the replication factor, which is the number of copies of each data block that are stored in HDFS for redundancy. The replication factor increases with the number of nodes in the cluster: there are 3 copies of each data block for a cluster with 10 or more nodes, 2 copies of each block for a cluster with 4 to 9 nodes, and 1 copy (no redundancy) for clusters with 3 or fewer nodes. The total HDFS space available is divided by the replication factor. In some cases, such as increasing the number of nodes from 9 to 10, the increase in replication factor can actually cause the amount of available HDFS space to decrease.

For example, a cluster with ten core nodes of type m1.large would have 2833 GB of space available to HDFS ((10 nodes X 850 GB per node)/replication factor of 3).

If your cluster exceeds the amount of space available to HDFS, you can add additional core nodes to your cluster or use data compression to create more HDFS space. If your cluster is one that can be stopped and restarted, you may consider using core nodes of a larger Amazon EC2 instance type. You might also consider adjusting the replication factor. Be aware, though, that decreasing the replication factor reduces the redundancy of HDFS data and your cluster's ability to recover from lost or corrupted HDFS blocks.

Have your previous clusters finished terminating?

Depending on the configuration of the cluster, it may take up to 5-20 minutes for the cluster to terminate and release allocated resources, such as Amazon EC2 instances. If you are getting a EC2 QUOTA EX-

CEEDED 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.

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 virtual server.
- If HDFS is used to retrieve the output, there may be an issue with HDFS.

The most common cause of this error is that the previous task is still processing. This is especially likely if the errors are occurring when the reduce tasks are first trying to run. You can check whether this is the case by reviewing the syslog log for the cluster step that is returning the error. If the syslog shows both map and reduce tasks making progress, this indicates that the reduce phase has started while there are map tasks that have not yet completed.

One thing to look for in the logs is a map progress percentage that goes to 100% and then drops back to a lower value. When the map percentage is at 100%, this does not mean that all map tasks are completed. It simply means that Hadoop is executing all the map tasks. If this value drops back below 100%, it means that a map task has failed and, depending on the configuration, Hadoop may try to reschedule the task. If the map percentage stays at 100% in the logs, look at the CloudWatch metrics, specifically `RunningMapTasks`, to check whether the map task is still processing. You can also find this information using the Hadoop web interface on the master node.

If you are seeing this issue, there are several things you can try:

- Instruct the reduce phase to wait longer before starting. You can do this by altering the Hadoop configuration setting `mapred.reduce.slowstart.completed.maps` to a longer time. For more information, see [Create Bootstrap Actions to Install Additional Software \(Optional\) \(p. 87\)](#).
- 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 virtual servers 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 Virtual Servers \(p. 33\)](#).
- 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 \(p. 464\)](#) and [Monitor Metrics with CloudWatch \(p. 423\)](#).

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 \(p. 418\)](#).

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 [Create Bootstrap Actions to Install Additional Software \(Optional\) \(p. 87\)](#).
- If the TaskTracker node has more than 4 failed tasks. You can change this to a higher value using the `modify mapred.max.tracker.failures` configuration parameter. Other configuration settings you might want to change are the settings that control how many times to attempt a task before marking it as failed: `mapred.map.max.attempts` for map tasks and `mapreduce.reduce.maxattempts` for reduce tasks. For more information about changing Hadoop configuration settings, see [Create Bootstrap Actions to Install Additional Software \(Optional\) \(p. 87\)](#).

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.

In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).

- Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --alive --name "Modified mapred.max.tracker.failures" \
--num-instances 2 --slave-instance-type m1.small --master-instance-type m1.small \
--key-pair key-pair --debug --log-uri s3://bucket-name \
--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.small --master-instance-type m1.small --key-pair key-pair --debug --log-uri s3://bucket-name \
--bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hadoop \
--bootstrap-name "Modified mapred.max.tracker.failures" --args "-m, mapred.max.tracker.failures=7"
```

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

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? \(p. 501\)](#)
- [Is your script timing out? \(p. 502\)](#)
- [Are you passing in invalid streaming arguments? \(p. 502\)](#)
- [Did your script exit with an error? \(p. 502\)](#)

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 \(p. 418\)](#).

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 \(p. 418\)](#).

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 \(p. 418\)](#).

Custom JAR Cluster Errors

The following errors are common to custom JAR clusters.

Topics

- [Is your JAR throwing an exception before creating a job? \(p. 503\)](#)
- [Is your JAR throwing an error inside a map task? \(p. 503\)](#)

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 \(p. 418\)](#).

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 \(p. 418\)](#).

Hive Cluster Errors

You can usually find the cause of a Hive error in the `syslog` file, which you link to from the **Steps** pane. If you can't determine the problem there, check in the Hadoop task attempt error message. Link to it on the **Task Attempts** pane.

The following errors are common to Hive clusters.

Topics

- [Are you using the latest version of Hive? \(p. 503\)](#)
- [Did you encounter a syntax error in the Hive script? \(p. 503\)](#)
- [Did a job fail when running interactively? \(p. 503\)](#)
- [Are you having trouble loading data to or from Amazon S3 into Hive? \(p. 503\)](#)

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 \(p. 213\)](#).

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 \(p. 418\)](#).

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 \(p. 418\)](#).

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? \(p. 495\)](#). 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 \(p. 213\)](#).
- 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.

VPC Errors

The following errors are common to VPC configuration in Amazon EMR.

Topics

- [Invalid Subnet Configuration \(p. 504\)](#)
- [Missing DHCP Options Set \(p. 504\)](#)

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-1. 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](#).

GovCloud-related Errors

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 \(p. 125\)](#).
- 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 \(p. 504\)](#).

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 \(p. 493\)](#).

Write Applications that Launch and Manage Clusters

Topics

- [End-to-End Amazon EMR Java Source Code Sample \(p. 506\)](#)
- [Common Concepts for API Calls \(p. 510\)](#)
- [Use SDKs to Call Amazon EMR APIs \(p. 512\)](#)

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 \(p. 512\)](#).

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 \(p. 512\)](#).

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. To create a simple cluster to run the sample on, see [Launch the Cluster \(p. 14\)](#).

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.

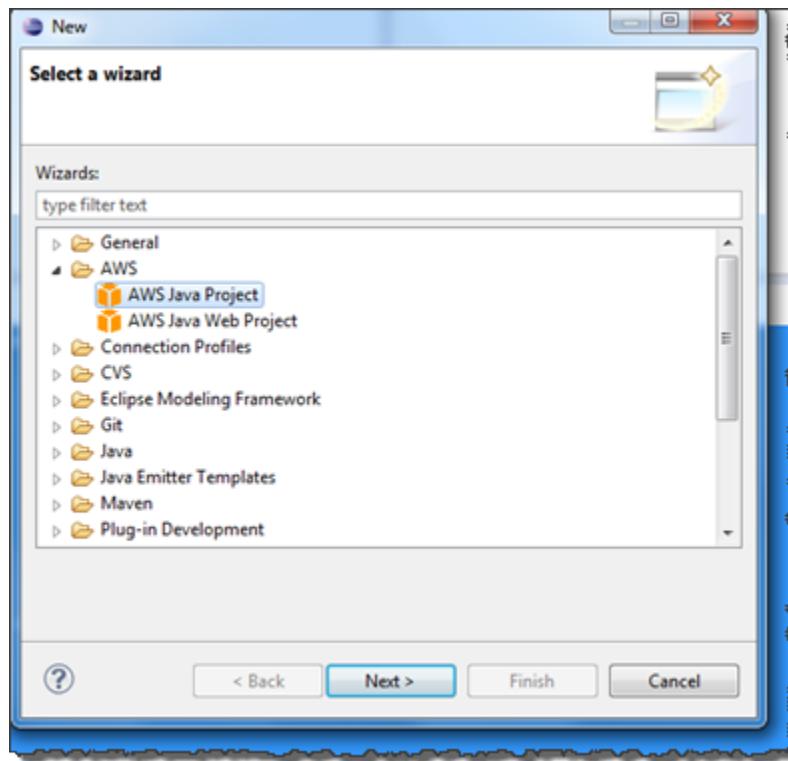
To install the AWS Toolkit for Eclipse

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**.

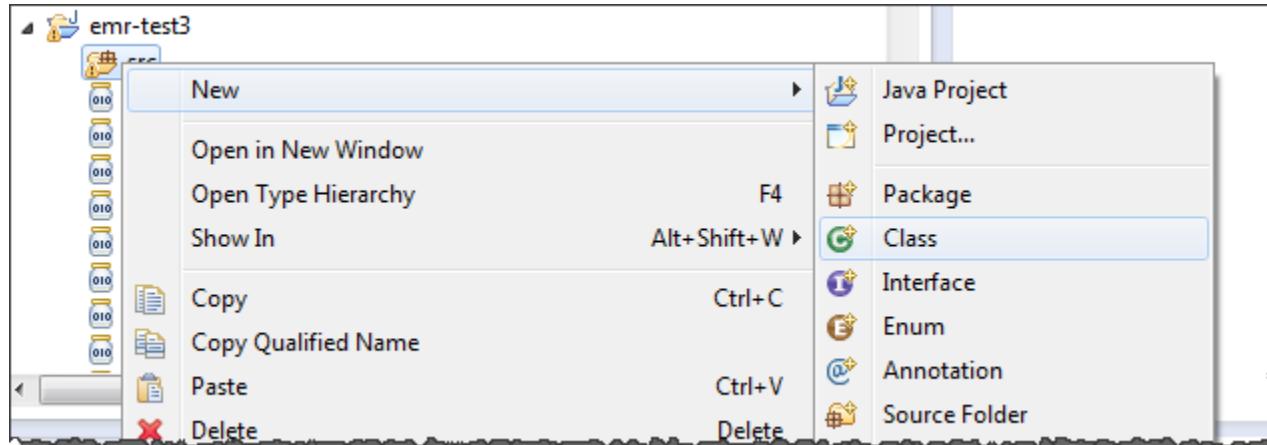


4. In the **New AWS Java Project** dialog, in the **Project name:** field, enter the name of your new project, for example `EMR-sample-code`.
5. Click **Configure AWS accounts...**, enter your public and private access keys, and click **Finish**. For more information about creating access keys, see [How 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.AWS Credentials;
import com.amazonaws.auth.PropertiesCredentials;
import com.amazonaws.services.elasticmapreduce.*;
import com.amazonaws.services.elasticmapreduce.model.AddJobFlowStepsRequest;
import com.amazonaws.services.elasticmapreduce.model.AddJobFlowStepsResult;
import com.amazonaws.services.elasticmapreduce.model.HadoopJarStepConfig;
import com.amazonaws.services.elasticmapreduce.model.StepConfig;
import com.amazonaws.services.elasticmapreduce.util.StepFactory;

public class main {

    public static void main(String[] args) {

        AWS Credentials credentials = null;
        try {
            credentials = new PropertiesCredentials(
                main.class.getResourceAsStream("AwsCredentials.properties"));
        } catch (IOException e1) {
            System.out.println("Credentials were not properly entered into
AwsCredentials.properties.");
            System.out.println(e1.getMessage());
            System.exit(-1);
        }

        AmazonElasticMapReduce client = new AmazonElasticMapReduceClient(credentials);

        // predefined steps. See StepFactory for list of predefined steps
        StepConfig hive = new StepConfig("Hive", new StepFactory().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")
            .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 \(p. 510\)](#)
- [Specifying Cluster Parameters in Amazon EMR \(p. 510\)](#)
- [Availability Zones in Amazon EMR \(p. 511\)](#)
- [How to Use Additional Files and Libraries in Amazon EMR Clusters \(p. 511\)](#)
- [Amazon EMR Sample Applications \(p. 511\)](#)

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 Elastic Compute Cloud Developer Guide](#).

How to Use Additional Files and Libraries in Amazon EMR Clusters

There are times when you might like to use additional files or custom libraries with your mapper or reducer applications. For example, you might like to use a library that converts a PDF file into plain text.

To cache a file for the mapper or reducer to use when using Hadoop streaming

- In the JAR `args` field:, add the following argument:

```
-cacheFile 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 \(p. 512\)](#)
- [Using the AWS SDK for .Net to Create an Amazon EMR Cluster \(p. 513\)](#)
- [Using the Java SDK to Sign an API Request \(p. 514\)](#)

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 enabledebugging = new StepConfig()
```

```
.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")
    .withSteps(enabledebugging, installHive)
    .withLogUri("s3://myawsbucket/")
    .withInstances(new JobFlowInstancesConfig()
        .withEc2KeyName("keypair")
        .withHadoopVersion("0.20")
        .withInstanceCount(5)
        .withKeepJobFlowAliveWhenNoSteps(true)
        .withMasterInstanceType("m1.small")
        .withSlaveInstanceType("m1.small")));
    .withSlaveInstanceType("m1.small"));

RunJobFlowResult result = emr.runJobFlow(request);
```

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.

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 enabledebugging = 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",
```

```
HadoopVersion = "0.20",
InstanceCount = 5,
KeepJobFlowAliveWhenNoSteps = true,
MasterInstanceType = "m1.small",
SlaveInstanceType = "m1.small"
};

var request = new RunJobFlowRequest{
    Name = "Hive Interactive",
    Steps = {enabledebugging, installHive},
    LogUri = "s3://myawsbucket",
    Instances = instanceConfig
};

var result = emrClient.RunJobFlow(request);
```

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

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 a Machine Image \(p. 51\)](#).

The following sections describe the various configuration settings and mechanisms available in Amazon EMR.

Topics

- [JSON Configuration Files \(p. 515\)](#)
- [Configuration of hadoop-user-env.sh \(p. 519\)](#)
- [Hadoop 2.2.0 and 2.4.0 Default Configuration \(p. 520\)](#)
- [Hadoop 1.0.3 Default Configuration \(p. 541\)](#)
- [Hadoop 20.205 Default Configuration \(p. 556\)](#)
- [Hadoop Memory-Intensive Configuration Settings \(Legacy AMI 1.0.1 and earlier\) \(p. 565\)](#)
- [Hadoop Default Configuration \(AMI 1.0\) \(p. 569\)](#)
- [Hadoop 0.20 Streaming Configuration \(p. 577\)](#)

JSON Configuration Files

Topics

- [Node Settings \(p. 515\)](#)
- [Cluster Configuration \(p. 517\)](#)

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
isMaster	Indicates that this node is the master node. Type: Boolean
isRunningNameNode	Indicates that this node is running the Hadoop name node daemon. Type: Boolean
isRunningDataNode	Indicates that this node is running the Hadoop data node daemon. Type: Boolean
isRunningJobTracker	Indicates that this node is running the Hadoop job tracker daemon. Type: Boolean
isRunningTaskTracker	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
isRunningResourceManager	Indicates that this node is running the Hadoop resource manager daemon. Type: Boolean
isRunningNodeManager	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 identify settings in JSON file using a bootstrap action

This procedure demonstrates how to execute the command line function `echo` to display the string `running on master node` on a master node by evaluating the JSON file parameter `instance.isMaster`.

- In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).
 - 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'"
```

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
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

Parameter	Description
instanceGroups	<p>A list of objects specifying each instance group in the cluster</p> <p>instanceGroupId—unique identifier for this instance group. Type: String</p> <p>instanceGroupName—uUser defined name of the instance group. Type: String</p> <p>instanceRole—one of Master, Core, or Task. Type: String</p> <p>instanceType—the Amazon EC2 type of the node, such as "m1.small". Type: String</p> <p>requestedInstanceCount—the target number of nodes for this instance group. Type: Long</p>

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:

```
#!/bin/bash export HADOOP_USER_CLASSPATH_FIRST=true; echo "HA  
DOOP_CLASSPATH=/path/to/my.jar" >> /home/hadoop/conf/hadoop-user-env.sh
```

For more information about using bootstrap actions, see [Create Bootstrap Actions to Install Additional Software \(Optional\) \(p. 87\)](#).

Hadoop 2.2.0 and 2.4.0 Default Configuration

Topics

- [Hadoop Configuration \(Hadoop 2.2.0, 2.4.0\) \(p. 520\)](#)
- [HDFS Configuration \(Hadoop 2.2.0\) \(p. 528\)](#)
- [Task Configuration \(Hadoop 2.2.0\) \(p. 529\)](#)
- [Intermediate Compression \(Hadoop 2.2.0\) \(p. 540\)](#)

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 a Machine Image \(p. 51\)](#).

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 a Machine Image \(p. 51\)](#).

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
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

Parameter	Value
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

Parameter	Value
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

Parameter	Value
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

cc1.4xlarge

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

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

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

Parameter	Value
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

Parameter	Value
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 NameNode.	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 configure-hadoop 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 \(AMI 3.0.0\) \(p. 529\)](#)
- [Avoiding Cluster Slowdowns \(AMI 3.0.0\) \(p. 539\)](#)

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
<code>mapreduce.map.java.opts</code>	<code>-Xmx512m</code>
<code>mapreduce.reduce.java.opts</code>	<code>-Xmx768m</code>
<code>mapreduce.map.memory.mb</code>	768
<code>mapreduce.reduce.memory.mb</code>	1024
<code>yarn.app.mapreduce.am.resource.mb</code>	1024
<code>yarn.scheduler.minimum-allocation-mb</code>	256
<code>yarn.scheduler.maximum-allocation-mb</code>	2048
<code>yarn.nodemanager.resource.memory-mb</code>	2048

m1.large

Configuration Option	Default Value
<code>mapreduce.map.java.opts</code>	<code>-Xmx512m</code>
<code>mapreduce.reduce.java.opts</code>	<code>-Xmx1024m</code>
<code>mapreduce.map.memory.mb</code>	768
<code>mapreduce.reduce.memory.mb</code>	1536
<code>yarn.app.mapreduce.am.resource.mb</code>	1536
<code>yarn.scheduler.minimum-allocation-mb</code>	256
<code>yarn.scheduler.maximum-allocation-mb</code>	3072
<code>yarn.nodemanager.resource.memory-mb</code>	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

Configuration Option	Default Value
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

Configuration Option	Default Value
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

cc1.4xlarge

Configuration Option	Default Value
mapreduce.map.java.opts	-Xmx1280m
mapreduce.reduce.java.opts	-Xmx2304m
mapreduce.map.memory.mb	1536

Configuration Option	Default Value
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

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

Configuration Option	Default Value
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

g2.2xlarge

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

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

Configuration Option	Default Value
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
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

Configuration Option	Default Value
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
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

Configuration Option	Default Value
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
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 using a bootstrap action

- In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).
 - Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --alive --name "JVM infinite reuse" \
--bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hadoop
```

```
\n--bootstrap-name "Configuring infinite JVM reuse" \n--args "-m,mapred.job.reuse.jvm.num.tasks=-1"
```

- Windows users:

```
ruby elastic-mapreduce --create --alive --name "JVM infinite reuse" --\nbootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hadoop\n--bootstrap-name "Configuring infinite JVM reuse" --args "-\n-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.

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 [Create Bootstrap Actions to Install Additional Software \(Optional\) \(p. 87\)](#).

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 using a bootstrap action

- In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).
 - Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --alive --name "Reducer speculative execution" \n--bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hadoop
```

```

\>
--bootstrap-name "Disable reducer speculative execution" \
--args "-m,mapred.reduce.tasks.speculative.execution=false"

```

- Windows users:

```

ruby elastic-mapreduce --create --alive --name "Reducer speculative execution" \
--bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hadoop \
--bootstrap-name "Disable reducer speculative execution" --args "-m,mapred.reduce.tasks.speculative.execution=false"

```

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 [Create Bootstrap Actions to Install Additional Software \(Optional\) \(p. 87\)](#).

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

To enable or disable compression using a bootstrap action

- In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).
 - 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 compression" \
--args "mapred.compress.map.output=false" \
--args "mapred.map.output.compression.codec=org.apache.hadoop.io.compress.GzipCodec"

```

- Windows users:

```

ruby elastic-mapreduce --create --alive --name "Reducer speculative execution" \
--bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hadoop \
--bootstrap-name "Disable compression" --args "mapred.com"

```

```
press.map.output=false" --args "mapred.map.output.compression.co  
dec=org.apache.hadoop.io.compress.GzipCodec"
```

Hadoop 1.0.3 Default Configuration

Topics

- [Hadoop Configuration \(Hadoop 1.0.3\) \(p. 541\)](#)
- [HDFS Configuration \(Hadoop 1.0.3\) \(p. 551\)](#)
- [Task Configuration \(Hadoop 1.0.3\) \(p. 551\)](#)
- [Intermediate Compression \(Hadoop 1.0.3\) \(p. 555\)](#)

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 a Machine Image \(p. 51\)](#).

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 a Machine Image \(p. 51\)](#).

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
mapred.tasktracker.reduce.tasks.maximum	1

m1.medium

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	1152

Parameter	Value
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
mapred.tasktracker.reduce.tasks.maximum	3

m2.xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	9216
HADOOP_NAMENODE_HEAPSIZE	3072
HADOOP_TASKTRACKER_HEAPSIZE	384

Parameter	Value
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
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

Parameter	Value
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
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
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

Parameter	Value
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

cc1.4xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	7680
HADOOP_NAMENODE_HEAPSIZE	3840
HADOOP_TASKTRACKER_HEAPSIZE	384
HADOOP_DATANODE_HEAPSIZE	384
mapred.child.java.opts	-Xmx912m
mapred.tasktracker.map.tasks.maximum	12
mapred.tasktracker.reduce.tasks.maximum	3

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
mapred.tasktracker.reduce.tasks.maximum	6

cg1.4xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	7680
HADOOP_NAMENODE_HEAPSIZE	3840
HADOOP_TASKTRACKER_HEAPSIZE	384

Parameter	Value
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
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

Parameter	Value
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

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
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

Parameter	Value
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

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

Parameter	Value
HADOOP_DATANODE_HEAPSIZE	4096
mapred.child.java.opts	-Xmx7552m
mapred.tasktracker.map.tasks.maximum	24
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 NameNode.	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 configure-hadoop 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 \(p. 551\)](#)
- [Tasks per Job \(AMI 2.3\) \(p. 553\)](#)
- [Task JVM Settings \(AMI 2.3\) \(p. 553\)](#)
- [Avoiding Cluster Slowdowns \(AMI 2.3\) \(p. 555\)](#)

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
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	1
m3.2xlarge	12	3
c1.medium	2	1
c1.xlarge	7	2
c3.xlarge	6	1
c3.2xlarge	12	3
c3.4xlarge	24	6
c3.8xlarge	48	12
cc1.4xlarge	12	3
cc2.8xlarge	24	6
hi1.4xlarge	24	6
hs1.8xlarge	24	6
cg1.4xlarge	12	3
g2.2xlarge	8	4
i2.xlarge	12	3
i2.2xlarge	12	3
i2.4xlarge	24	6
i2.8xlarge	48	12
r3.xlarge	6	1
r3.2xlarge	12	3
r3.4xlarge	24	6
r3.8xlarge	48	12

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
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 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	<code>-Xmx288m</code>
m1.medium	<code>-Xmx576m</code>
m1.large	<code>-Xmx864m</code>
m1.xlarge	<code>-Xmx768m</code>
c1.medium	<code>-Xmx288m</code>
c1.xlarge	<code>-Xmx384m</code>
m2.xlarge	<code>-Xmx2304m</code>

Amazon EC2 Instance Name	Default JVM value
m2.2xlarge	-Xmx2688m
m2.4xlarge	-Xmx2304m
cc1.4xlarge	-Xmx912m
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.

To modify JVM using a bootstrap action

- In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).
 - 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.

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 [Create Bootstrap Actions to Install Additional Software \(Optional\) \(p. 87\)](#).

Speculative Execution Parameters

Parameter	Default Setting
mapred.map.tasks.speculative.execution	true
mapred.reduce.tasks.speculative.execution	true

To disable reducer speculative execution using a bootstrap action

- In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).
 - Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --alive --name "Reducer speculative execution"
  \
  --bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hadoop
  \
  --bootstrap-name "Disable reducer speculative execution" \
  --args "-m,mapred.reduce.tasks.speculative.execution=false"
```

- Windows users:

```
ruby elastic-mapreduce --create --alive --name "Reducer speculative execution" --bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hadoop --bootstrap-name "Disable reducer speculative execution" --args "-m,mapred.reduce.tasks.speculative.execution=false"
```

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 [Create Bootstrap Actions to Install Additional Software \(Optional\) \(p. 87\)](#).

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

To enable or disable compression using a bootstrap action

- In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).
 - 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 compression" \
--args "mapred.compress.map.output=false" \
--args "mapred.map.output.compression.codec=org.apache.hadoop.io.compress.GzipCodec"
```

- Windows users:

```
ruby elastic-mapreduce --create --alive --name "Reducer speculative execution" --bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hadoop --bootstrap-name "Disable compression" --args "mapred.compress.map.output=false" --args "mapred.map.output.compression.codec=org.apache.hadoop.io.compress.GzipCodec"
```

Hadoop 20.205 Default Configuration

Topics

- [Hadoop Configuration \(Hadoop 20.205\) \(p. 557\)](#)
- [HDFS Configuration \(Hadoop 20.205\) \(p. 560\)](#)
- [Task Configuration \(Hadoop 20.205\) \(p. 561\)](#)
- [Intermediate Compression \(Hadoop 20.205\) \(p. 565\)](#)

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 a Machine Image \(p. 51\)](#).

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 a Machine Image \(p. 51\)](#).

m1.small

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

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

Parameter	Value
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
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

Parameter	Value
mapred.tasktracker.reduce.tasks.maximum	2

m2.xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	12288
HADOOP_NAMENODE_HEAPSIZE	4096
HADOOP_TASKTRACKER_HEAPSIZE	512
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

cc1.4xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	10240
HADOOP_NAMENODE_HEAPSIZE	5120
HADOOP_TASKTRACKER_HEAPSIZE	512
HADOOP_DATANODE_HEAPSIZE	512
mapred.child.java.opts	-Xmx1216m
mapred.tasktracker.map.tasks.maximum	12
mapred.tasktracker.reduce.tasks.maximum	3

cc2.8xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	40152
HADOOP_NAMENODE_HEAPSIZE	16384
HADOOP_TASKTRACKER_HEAPSIZE	512
HADOOP_DATANODE_HEAPSIZE	512
mapred.child.java.opts	-Xmx2048m
mapred.tasktracker.map.tasks.maximum	24
mapred.tasktracker.reduce.tasks.maximum	6

cg1.4xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	10240
HADOOP_NAMENODE_HEAPSIZE	5120
HADOOP_TASKTRACKER_HEAPSIZE	512
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 NameNode.	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 configure-hadoop 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 \(p. 561\)](#)
- [Tasks per Job \(AMI 2.0 and 2.1\) \(p. 562\)](#)
- [Task JVM Settings \(AMI 2.0 and 2.1\) \(p. 562\)](#)
- [Avoiding Cluster Slowdowns \(AMI 2.0 and 2.1\) \(p. 564\)](#)

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

Amazon EC2 Instance Name	Mappers	Reducers
m2.4xlarge	14	4
cc1.4xlarge	12	3
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
cc1.4xlarge	-Xmx1216m
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.

To modify JVM using a bootstrap action

- In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).
 - 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.

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 [Create Bootstrap Actions to Install Additional Software \(Optional\) \(p. 87\)](#).

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 using a bootstrap action

- In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).
 - Linux, UNIX, and Mac OS X users:

```
./elastic-mapreduce --create --alive --name "Reducer speculative execution"
 \
--bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hadoop
 \
--bootstrap-name "Disable reducer speculative execution" \
--args "-m,mapred.reduce.tasks.speculative.execution=false"
```

- Windows users:

```
ruby elastic-mapreduce --create --alive --name "Reducer speculative execution"
 --bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hadoop
 --bootstrap-name "Disable reducer speculative execution" --args "-m,mapred.reduce.tasks.speculative.execution=false"
```

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 [Create Bootstrap Actions to Install Additional Software \(Optional\) \(p. 87\)](#).

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

To enable or disable compression using a bootstrap action

- In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).
 - 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 compression" \
--args "mapred.compress.map.output=false" \
--args "mapred.map.output.compression.codec=org.apache.hadoop.io.compress.GzipCodec"
```

- Windows users:

```
ruby elastic-mapreduce --create --alive --name "Reducer speculative execution" --bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hadoop --bootstrap-name "Disable compression" --args "mapred.compress.map.output=false" --args "mapred.map.output.compression.codec=org.apache.hadoop.io.compress.GzipCodec"
```

Hadoop Memory-Intensive Configuration Settings (Legacy AMI 1.0.1 and earlier)

Note

The memory-intensive settings are set by default in AMI 2.0.0 and later. You should only need to adjust these settings for AMI versions 1.0.1 and earlier.

The Amazon EMR default configuration settings are appropriate for most workloads. However, based on your cluster's specific memory and processing requirements, you might want to modify the configuration settings.

For example, if your cluster tasks are memory-intensive, you can use fewer tasks per core node and reduce your job tracker heap size. A predefined bootstrap action is available to configure your cluster on startup.

The following tables list the recommended configuration settings for each EC2 instance type. The default configurations for the cc1.4xlarge, cc2.8xlarge, hi1.4xlarge, hs1.8xlarge, and cg1.4xlarge instances are sufficient for memory-intensive workloads; therefore, the recommended configuration settings for these instances are not listed.

m1.small

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	512
HADOOP_NAMENODE_HEAPSIZE	512
HADOOP_TASKTRACKER_HEAPSIZE	256
HADOOP_DATANODE_HEAPSIZE	128
mapred.child.java.opts	-Xmx512m
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

Parameter	Value
mapred.child.java.opts	-Xmx1024m
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
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	512
HADOOP_TASKTRACKER_HEAPSIZE	256
HADOOP_DATANODE_HEAPSIZE	128
mapred.child.java.opts	-Xmx512m
mapred.tasktracker.map.tasks.maximum	2
mapred.tasktracker.reduce.tasks.maximum	1

c1.xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	2048
HADOOP_NAMENODE_HEAPSIZE	1024
HADOOP_TASKTRACKER_HEAPSIZE	512
HADOOP_DATANODE_HEAPSIZE	512
mapred.child.java.opts	-Xmx512m
mapred.tasktracker.map.tasks.maximum	7

Parameter	Value
mapred.tasktracker.reduce.tasks.maximum	2

m2.xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	4096
HADOOP_NAMENODE_HEAPSIZE	2048
HADOOP_TASKTRACKER_HEAPSIZE	512
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	8192
HADOOP_NAMENODE_HEAPSIZE	4096
HADOOP_TASKTRACKER_HEAPSIZE	1024
HADOOP_DATANODE_HEAPSIZE	1024
mapred.child.java.opts	-Xmx4096m
mapred.tasktracker.map.tasks.maximum	6
mapred.tasktracker.reduce.tasks.maximum	2

m2.4xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	8192
HADOOP_NAMENODE_HEAPSIZE	8192
HADOOP_TASKTRACKER_HEAPSIZE	1024
HADOOP_DATANODE_HEAPSIZE	1024
mapred.child.java.opts	-Xmx4096m
mapred.tasktracker.map.tasks.maximum	14
mapred.tasktracker.reduce.tasks.maximum	4

Hadoop Default Configuration (AMI 1.0)

Topics

- [Hadoop Configuration \(AMI 1.0\) \(p. 569\)](#)
- [HDFS Configuration \(AMI 1.0\) \(p. 572\)](#)
- [Task Configuration \(AMI 1.0\) \(p. 573\)](#)
- [Intermediate Compression \(AMI 1.0\) \(p. 577\)](#)

This section describes the default configuration settings that Amazon EMR uses to configure a Hadoop cluster launched with Amazon Machine Image (AMI) version 1.0. For more information about the AMI versions supported by Amazon EMR, see [Choose a Machine Image \(p. 51\)](#).

Hadoop Configuration (AMI 1.0)

The following Amazon EMR default configuration settings 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. These and other memory-intensive configuration settings are described in [Hadoop Memory-Intensive Configuration Settings \(Legacy AMI 1.0.1 and earlier\) \(p. 565\)](#).

The following tables list the default configuration settings for each EC2 instance type in clusters launched with Amazon EMR AMI version 1.0. For more information about the AMI versions supported by Amazon EMR, see [Choose a Machine Image \(p. 51\)](#).

m1.small

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	768
HADOOP_NAMENODE_HEAPSIZE	256
HADOOP_TASKTRACKER_HEAPSIZE	512
HADOOP_DATANODE_HEAPSIZE	128
mapred.child.java.opts	-Xmx725m
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	-Xmx1152m

Parameter	Value
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	1536
HADOOP_DATANODE_HEAPSIZE	256
mapred.child.java.opts	-Xmx1600m
mapred.tasktracker.map.tasks.maximum	4
mapred.tasktracker.reduce.tasks.maximum	2

m1.xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	9216
HADOOP_NAMENODE_HEAPSIZE	3072
HADOOP_TASKTRACKER_HEAPSIZE	3072
HADOOP_DATANODE_HEAPSIZE	512
mapred.child.java.opts	-Xmx1600m
mapred.tasktracker.map.tasks.maximum	8
mapred.tasktracker.reduce.tasks.maximum	4

c1.medium

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	768
HADOOP_NAMENODE_HEAPSIZE	256
HADOOP_TASKTRACKER_HEAPSIZE	512
HADOOP_DATANODE_HEAPSIZE	256
mapred.child.java.opts	-Xmx362m
mapred.tasktracker.map.tasks.maximum	4
mapred.tasktracker.reduce.tasks.maximum	2

c1.xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	3072
HADOOP_NAMENODE_HEAPSIZE	1024
HADOOP_TASKTRACKER_HEAPSIZE	1536
HADOOP_DATANODE_HEAPSIZE	512
mapred.child.java.opts	-Xmx747m
mapred.tasktracker.map.tasks.maximum	8
mapred.tasktracker.reduce.tasks.maximum	4

m2.xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	12288
HADOOP_NAMENODE_HEAPSIZE	4096
HADOOP_TASKTRACKER_HEAPSIZE	3072
HADOOP_DATANODE_HEAPSIZE	512
mapred.child.java.opts	-Xmx2048m
mapred.tasktracker.map.tasks.maximum	4
mapred.tasktracker.reduce.tasks.maximum	2

m2.2xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	24576
HADOOP_NAMENODE_HEAPSIZE	8192
HADOOP_TASKTRACKER_HEAPSIZE	3072
HADOOP_DATANODE_HEAPSIZE	1024
mapred.child.java.opts	-Xmx3200m
mapred.tasktracker.map.tasks.maximum	8
mapred.tasktracker.reduce.tasks.maximum	4

m2.4xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	49152

Parameter	Value
HADOOP_NAMENODE_HEAPSIZE	16384
HADOOP_TASKTRACKER_HEAPSIZE	3072
HADOOP_DATANODE_HEAPSIZE	2048
mapred.child.java.opts	-Xmx3733m
mapred.tasktracker.map.tasks.maximum	16
mapred.tasktracker.reduce.tasks.maximum	8

cc1.4xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	10240
HADOOP_NAMENODE_HEAPSIZE	3840
HADOOP_TASKTRACKER_HEAPSIZE	512
HADOOP_DATANODE_HEAPSIZE	384
mapred.child.java.opts	-Xmx1024m
mapred.tasktracker.map.tasks.maximum	12
mapred.tasktracker.reduce.tasks.maximum	3

cg1.4xlarge

Parameter	Value
HADOOP_JOBTRACKER_HEAPSIZE	10240
HADOOP_NAMENODE_HEAPSIZE	5120
HADOOP_TASKTRACKER_HEAPSIZE	512
HADOOP_DATANODE_HEAPSIZE	512
mapred.child.java.opts	-Xmx1024m
mapred.tasktracker.map.tasks.maximum	12
mapred.tasktracker.reduce.tasks.maximum	3

HDFS Configuration (AMI 1.0)

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 NameNode.	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 configure-hadoop bootstrap action.	1 for clusters < four nodes 2 for clusters < ten nodes 3 for all other clusters

Task Configuration (AMI 1.0)

Topics

- [Tasks per Machine \(p. 573\)](#)
- [Tasks per Job \(AMI 1.0\) \(p. 574\)](#)
- [Task JVM Settings \(AMI 1.0\) \(p. 574\)](#)
- [Avoiding Cluster Slowdowns \(AMI 1.0\) \(p. 576\)](#)

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.

Amazon EC2 Instance Name	Mappers	Reducers
m1.small	2	1
m1.medium	2	1
m1.large	4	2
m1.xlarge	8	4
c1.medium	4	2
c1.xlarge	8	4
m2.xlarge	4	2
m2.2xlarge	8	4

Amazon EC2 Instance Name	Mappers	Reducers
m2.4xlarge	16	8
cc1.4xlarge	12	3
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 1.0)

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 <i>mapred.map.tasks</i> 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 1.0)

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	<i>-Xmx725m</i>

Amazon EC2 Instance Name	Default JVM value
m1.medium	-Xmx1152m
m1.large	-Xmx1600m
m1.xlarge	-Xmx1600m
c1.medium	-Xmx362m
c1.xlarge	-Xmx747m
m2.xlarge	-Xmx2048m
m2.2xlarge	-Xmx3200m
m2.4xlarge	-Xmx3733m
cc1.4xlarge	-Xmx1024m
cg1.4xlarge	-Xmx1024m

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.

To modify JVM using a bootstrap action

- In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).
 - 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.

Avoiding Cluster Slowdowns (AMI 1.0)

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 disables it for reducers in AMI 1.0. You can override these settings with a bootstrap action. For more information about using bootstrap actions, see [Create Bootstrap Actions to Install Additional Software \(Optional\) \(p. 87\)](#).

Speculative Execution Parameters

Parameter	Default Setting
<code>mapred.map.tasks.speculative.execution</code>	true
<code>mapred.reduce.tasks.speculative.execution</code>	false

To enable reducer speculative execution using a bootstrap action

- In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).
 - 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 "Enable reducer speculative execution" \
--args "-m, mapred.reduce.tasks.speculative.execution=true"
```

- Windows users:

```
ruby elastic-mapreduce --create --alive --name "Reducer speculative execution"
 --bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hadoop
 --bootstrap-name "Enable reducer speculative execution" --args "-m, mapred.reduce.tasks.speculative.execution=true"
```

Intermediate Compression (AMI 1.0)

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 LZO codec.

You can modify the default compression settings with a bootstrap action. For more information about using bootstrap actions, see [Create Bootstrap Actions to Install Additional Software \(Optional\) \(p. 87\)](#).

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	com.hadoop.compression.lzo.LzoCodec

To enable or disable compression using a bootstrap action

- In the directory where you installed the Amazon EMR CLI, run the following from the command line. For more information, see the [Command Line Interface Reference for Amazon EMR \(p. 579\)](#).
 - 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 compression" \
--args "mapred.compress.map.output=false" \
--args "mapred.map.output.compression.codec=org.apache.hadoop.io.compress.GzipCodec
```

- Windows users:

```
ruby elastic-mapreduce --create --alive --name "Reducer speculative execution" --bootstrap-action s3://elasticmapreduce/bootstrap-actions/configure-hadoop --bootstrap-name "Disable compression" --args "mapred.compress.map.output=false" --args "mapred.map.output.compression.codec=org.apache.hadoop.io.compress.GzipCodec
```

Hadoop 0.20 Streaming Configuration

Hadoop 0.20 and later supports the three streaming parameters described in the following table, in addition to the version 0.18 parameters.

Parameter	Description
-files	Specifies comma-separated files to copy to the MapReduce cluster.

Parameter	Description
<code>-archives</code>	Specifies comma-separated archives to restore to the compute machines.
<code>-D <i>KEY=VALUE</i></code>	Sets a Hadoop configuration variable. <i>KEY</i> is a Hadoop configuration, such as <code>mapred.map.tasks</code> , and <i>VALUE</i> is the new value.

The `--files` and `--archives` parameters are similar to `--cacheFile` and `--cacheArchive` of Hadoop 0.18, except that they accept comma-separated values.

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 AWS Command Line Interface version 1.4 provides support for Amazon EMR. We recommend you use the AWS CLI by downloading and installing the latest version. For more information, see <http://aws.amazon.com/cli/>.

Topics

- [Install the Amazon EMR Command Line Interface \(p. 579\)](#)
- [How to Call the Command Line Interface \(p. 585\)](#)
- [Command Line Interface Options \(p. 585\)](#)
- [Command Line Interface Releases \(p. 596\)](#)

Install the Amazon EMR Command Line Interface

To install the command line interface, complete the following tasks:

Topics

- [Installing Ruby \(p. 579\)](#)
- [Verifying the RubyGems package management framework \(p. 580\)](#)
- [Installing the Command Line Interface \(p. 581\)](#)
- [Configuring Credentials \(p. 581\)](#)
- [SSH Setup and Configuration \(p. 584\)](#)

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 http://rubyforge.org/frs/?group_id=126.
2. Install RubyGems using the following command.

```
sudo ruby setup.rb
```

Installing the 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, 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.

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. Select the **Region** for your bucket.

| If your Amazon EMR region is... | Select the Amazon S3 region... |
|---------------------------------|--------------------------------|
| us-east-1                       | <b>US Standard</b>             |
| us-west-2                       | <b>Oregon</b>                  |
| us-west-1                       | <b>Northern California</b>     |
| eu-west-1                       | <b>Ireland</b>                 |
| ap-northeast-1                  | <b>Japan</b>                   |
| ap-southeast-1                  | <b>Singapore</b>               |
| ap-southeast-2                  | <b>Sydney</b>                  |
| sa-east-1                       | <b>Sao Paulo</b>               |
| us-gov-west-1                   | <b>GovCloud</b>                |

**Note**

To use the AWS GovCloud region, contact your AWS business representative. You can't create an AWS GovCloud 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 `s3n://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 Setup and Configuration

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
  - 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>.
  - 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, `mykeypair.ppk`.
  - Click **Save**.
  - Exit the PuTTYgen application.

### Verify installation of the Amazon EMR CLI

- In the directory where you installed the Amazon EMR CLI, run the following commands from the command line:
  - 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

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. For more information about the CLI options, see [Command Line Interface Options \(p. 585\)](#).

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
```

# Command Line Interface Options

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 \(p. 587\)](#)
- [Options Common to All Step Types \(p. 587\)](#)
- [Short Options \(p. 587\)](#)
- [Adding and Modifying Instance Groups \(p. 587\)](#)
- [Adding JAR Steps to Job Flows \(p. 588\)](#)
- [Adding JSON Steps to Job Flows \(p. 588\)](#)
- [Adding Streaming Steps to Job Flows \(p. 588\)](#)
- [Assigning an Elastic IP Address to the Master Node \(p. 589\)](#)
- [Contacting the Master Node \(p. 589\)](#)
- [Creating Job Flows \(p. 590\)](#)
- [HBase Options \(p. 591\)](#)
- [Hive Options \(p. 593\)](#)
- [Impala Options \(p. 593\)](#)
- [Listing and Describing Job Flows \(p. 594\)](#)

- [Passing Arguments to Steps \(p. 594\)](#)
- [Pig Options \(p. 595\)](#)
- [Specific Steps \(p. 595\)](#)
- [Specifying Bootstrap Actions \(p. 596\)](#)
- [Tagging \(p. 596\)](#)
- [Terminating job flows \(p. 596\)](#)

## Common Options

`-a ACCESS_ID`  
Sets the AWS access identifier.

`--access-id ACCESS_ID`  
Sets the AWS access identifier.

`--credentials CREDENTIALS_FILE`  
Specifies the credentials file that contains the AWS access identifier and the AWS private key to use when contacting Amazon EMR.

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.

`--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.

Usage: [View Cluster Details \(p. 407\)](#), [Add Steps to a Cluster \(p. 473\)](#), [Resize a Running Cluster \(p. 464\)](#), [Change the Number of Spot Instances in a Cluster \(p. 47\)](#)

`-j JOB_FLOW_IDENTIFIER`  
Specifies the cluster with the given cluster identifier.

Usage: [View Cluster Details \(p. 407\)](#), [Add Steps to a Cluster \(p. 473\)](#), [Resize a Running Cluster \(p. 464\)](#), [Change the Number of Spot Instances in a Cluster \(p. 47\)](#)

`--log-uri`  
Specifies the Amazon S3 bucket to receive log files. Used with `--create`.

Usage: [View HBase Log Files \(p. 315\)](#)

`--private-key PRIVATE_KEY`  
Specifies the AWS private key to use when contacting Amazon EMR.

`--trace`  
Traces commands made to the web service.

`--verbose`  
Turns on verbose logging of program interaction.

`--version`  
Displays the version of the CLI.

Usage: Command Line Interface Releases (p. 596)

## 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.
```

## Short Options

```
-aACCESS_ID
 Sets the AWS access identifier.

-cCREDENTIALS_FILE
 Specifies the credentials file that contains the AWS access identifier and the AWS private key to use when contacting Amazon EMR.

 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.

-h
 Displays help information from the CLI.

-jJOB_FLOW_IDENTIFIER
 Specifies the cluster with the given cluster identifier.

 Usage: View Cluster Details \(p. 407\), Add Steps to a Cluster \(p. 473\), Resize a Running Cluster \(p. 464\), Change the Number of Spot Instances in a Cluster \(p. 47\)

-pPRIVATE_KEY
 Specifies the AWS private key to use when contacting Amazon EMR.

-v
 Turns on verbose logging of program interaction.
```

## Adding and Modifying Instance Groups

```
--add-instance-group INSTANCE_ROLE
 Adds an instance group to an existing cluster. The role may be task only.
```

Usage: [Resize a Running Cluster \(p. 464\)](#), [Change the Number of Spot Instances in a Cluster \(p. 47\)](#)

--modify-instance-group *INSTANCE\_GROUP\_ID*  
Modifies an existing instance group.

Usage: [Resize a Running Cluster \(p. 464\)](#), [Change the Number of Spot Instances in a Cluster \(p. 47\)](#)

--add-instance-group *INSTANCE\_ROLE*  
Adds an instance group to an existing cluster. The role may be `task` only.

Usage: [Resize a Running Cluster \(p. 464\)](#), [Change the Number of Spot Instances in a Cluster \(p. 47\)](#)

## 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.

Usage: [Resize a Running Cluster \(p. 464\)](#), [Distributed Copy Using S3DistCp \(p. 355\)](#)

--main-class  
Specifies the JAR file's main class. This parameter is not needed if your JAR file has a manifest.

Usage: [Add Steps to a Cluster \(p. 473\)](#)

## 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.

Usage: [Import files using Distributed Cache \(p. 106\)](#)

--cache-archive *LOCATION#NAME\_OF\_ARCHIVE*  
Adds an archive file to the distributed cache

Usage: [Import files using Distributed Cache \(p. 106\)](#)

--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.

Usage: [Launch a Streaming Cluster \(p. 169\)](#)

--instance-count *INSTANCE\_COUNT*  
Sets the count of nodes for an instance group.

Usage: [Resize a Running Cluster \(p. 464\)](#), [Change the Number of Spot Instances in a Cluster \(p. 41\)](#)

```
--instance-type INSTANCE_TYPE
 Sets the type of EC2 instance to create nodes for an instance group.

 Usage: Resize a Running Cluster \(p. 464\), Launch Spot Instances in a Cluster \(p. 41\)

--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.

 Usage: Launch a Streaming Cluster \(p. 169\)

--output LOCATION_OF_JOB_FLOW_OUTPUT
 Specifies the output location for the cluster.

 Usage: Launch a Streaming Cluster \(p. 169\)

--reducer REDUCER
 The name of a Hadoop built-in class or the location of a reducer script.

 Usage: Launch a Streaming Cluster \(p. 169\)

--stream
 Used with --create and --arg to launch a streaming cluster.

Note
 The --arg option must immediately follow the --stream option.

 Usage: Launch a Streaming Cluster \(p. 169\)
```

## 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. For more information, see Associate an
 Elastic IP Address with a Cluster \(p. 477\).
```

## Contacting 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.

 Usage: Add Steps to a Cluster \(p. 473\)

--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.

 Usage: Open an SSH Tunnel to the Master Node \(p. 452\), Configure FoxyProxy to View Websites
 Hosted on the Master Node \(p. 453\)
```

--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.

Usage: [Connect to the Master Node Using SSH \(p. 446\)](#)

--to *DESTINATION*  
Specifies the destination location when copying files to and from the master node using SCP.

## 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.

Usage: [Add Steps to a Cluster \(p. 473\)](#)

--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.

Usage: [Choose a Machine Image \(p. 51\)](#)

--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.

Usage: [Launch Spot Instances in a Cluster \(p. 41\)](#)

--create  
Launches a new cluster.

Usage: [Launch a Streaming Cluster \(p. 169\)](#), [Launch a Hive Cluster \(p. 234\)](#), [Launch a Pig Cluster \(p. 281\)](#), [Launch a Cascading Cluster \(p. 177\)](#), [Launch an HBase Cluster on Amazon EMR \(p. 291\)](#)

--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.

Usage: [Resize a Running Cluster \(p. 464\)](#)

--jobflow-role *IAM\_ROLE\_NAME*  
Launches the EC2 instances of a cluster with the specified IAM role.

[Configure IAM Roles for Amazon EMR \(p. 125\)](#)

--service-role *IAM\_ROLE\_NAME*  
Launches the Amazon EMR service with the specified IAM role.

[Configure IAM Roles for Amazon EMR \(p. 125\)](#)

--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.

Usage: [Build Binaries Using Amazon EMR \(p. 161\)](#)

--name "*JOB\_FLOW\_NAME*"

Specifies a name for the cluster. This can only be set when the jobflow is created.

Usage: [Launch a Streaming Cluster \(p. 169\)](#), [Launch a Hive Cluster \(p. 234\)](#), [Launch a Pig Cluster \(p. 281\)](#), [Launch a Cascading Cluster \(p. 177\)](#), [Launch an HBase Cluster on Amazon EMR \(p. 291\)](#)

--num-instances *NUMBER\_OF\_INSTANCES*

Used with --create and --modify-instance-group to specify the number of EC2 instances in the cluster.

Usage: [Launch a Streaming Cluster \(p. 169\)](#), [Launch a Hive Cluster \(p. 234\)](#), [Launch a Pig Cluster \(p. 281\)](#), [Launch a Cascading Cluster \(p. 177\)](#), [Launch an HBase Cluster on Amazon EMR \(p. 291\)](#), [Change the Number of Spot Instances in a Cluster \(p. 47\)](#)

--plain-output

Returns the cluster identifier from the create step as simple text.

--region *REGION*

Specifies the region in which to launch the cluster.

Usage: [Choose an AWS Region \(p. 29\)](#)

--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.

Usage: [Select a Amazon VPC Subnet for the Cluster \(Optional\) \(p. 137\)](#)

--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.

Usage: [Configure IAM User Permissions \(p. 119\)](#)

--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 --argsparameter.

--with-termination-protection

Used with --create to launch the cluster with termination protection enabled.

Usage: [Protect a Cluster from Termination \(p. 459\)](#)

## HBase Options

--backup-dir *BACKUP\_LOCATION*

The directory where an Hbase backup exists or should be created.

Usage: [Back Up and Restore HBase \(p. 298\)](#)

--backup-version *VERSION\_NUMBER*

Specifies the version number of an existing Hbase backup to restore.

Usage: [Back Up and Restore HBase \(p. 298\)](#)

--consistent

Pauses all write operations to the HBase cluster during the backup process, to ensure a consistent backup.

[Usage: Back Up and Restore HBase \(p. 298\)](#)

--full-backup-time-interval *INTERVAL*  
An integer that specifies the period of time units to elapse between automated full backups of the HBase cluster.

[Usage: Back Up and Restore HBase \(p. 298\)](#)

--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.

[Usage: Back Up and Restore HBase \(p. 298\)](#)

--hbase  
Used to launch an Hbase cluster.

[Usage: Launch an HBase Cluster on Amazon EMR \(p. 291\)](#)

--hbase-backup  
Creates a one-time backup of HBase data to the location specified by --backup-dir.

[Usage: Back Up and Restore HBase \(p. 298\)](#)

--hbase-restore  
Restores a backup from the location specified by --backup-dir and (optionally) the version specified by --backup-version.

[Usage: Back Up and Restore HBase \(p. 298\)](#)

--hbase-schedule-backup  
Schedules an automated backup of HBase data.

[Usage: Back Up and Restore HBase \(p. 298\)](#)

--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.

[Usage: Back Up and Restore HBase \(p. 298\)](#)

--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.

[Usage: Back Up and Restore HBase \(p. 298\)](#)

--disable-full-backups  
Turns off scheduled full Hbase backups by passing this flag into a call with --hbase-schedule-backup.

[Usage: Back Up and Restore HBase \(p. 298\)](#)

--disable-incremental-backups  
Turns off scheduled incremental Hbase backups by passing this flag into a call with --hbase-schedule-backup.

[Usage: Back Up and Restore HBase \(p. 298\)](#)

--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.

Usage: [Back Up and Restore HBase \(p. 298\)](#)

## Hive Options

--hive-interactive

Used with --create to launch a cluster with Hive installed.

Usage: [Interactive and Batch Hive Clusters \(p. 231\)](#)

--hive-script *HIVE\_SCRIPT\_LOCATION*

The Hive script to run in the cluster.

Usage: [Interactive and Batch Hive Clusters \(p. 231\)](#)

--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*.

Usage: [Create a Metastore Outside the Hadoop Cluster \(p. 241\)](#), [Additional Features of Hive in Amazon EMR \(p. 205\)](#)

--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.

Usage: [Supported Hive Versions \(p. 213\)](#)

## 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.

Usage: [Launch the Cluster \(p. 249\)](#)

--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.
```

## Listing and Describing Job Flows

`--active`

Modifies a command to apply only to clusters in the RUNNING, STARTING or WAITING states. Used with `--list`.

Usage: [View Cluster Details \(p. 407\)](#)

`--all`

Modifies a command to apply only to all clusters, regardless of status. Used with `--list`, it lists all the clusters created in the last two weeks.

`--created-after=DATETIME`

Lists all clusters created after the specified time and date in XML date-time format.

`--created-before=DATETIME`

Lists all clusters created before the specified time and date in XML date-time format.

`--describe`

Returns information about the specified cluster or clusters.

Usage: [View Cluster Details \(p. 407\)](#)

`--list`

Lists clusters created in the last two days.

Usage: [View Cluster Details \(p. 407\)](#)

`--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.

Usage: [Supported Hive Versions \(p. 213\)](#)

`--state JOB_FLOW_STATE`

Specifies the state of the cluster. The cluster state will be one of the following values: STARTING, RUNNING, WAITING, TERMINATED.

Usage: [View Cluster Details \(p. 407\)](#)

## 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.

Usage: [Launch a Hive Cluster \(p. 234\)](#), [Launch a Pig Cluster \(p. 281\)](#), [Launch a Cascading Cluster \(p. 177\)](#), [Add Steps to a Cluster \(p. 473\)](#), [Create Bootstrap Actions to Install Additional Software \(Optional\) \(p. 87\)](#)

`--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 comma 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.

Usage: [Launch a Hive Cluster \(p. 234\)](#), [Launch a Pig Cluster \(p. 281\)](#), [Launch a Cascading Cluster \(p. 177\)](#), [Add Steps to a Cluster \(p. 473\)](#), [Create Bootstrap Actions to Install Additional Software \(Optional\) \(p. 87\)](#)

`--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.

## Pig Options

`--pig-interactive`

Used with `--create` to launch a cluster with Pig installed.

Usage: [Launch a Pig Cluster \(p. 281\)](#)

`--pig-script PIG_SCRIPT_LOCATION`

The Pig script to run in the cluster.

Usage: [Launch a Pig Cluster \(p. 281\)](#)

`--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.

Usage: [Supported Pig Versions \(p. 273\)](#)

## Specific Steps

`--enable-debugging`

Used with `--create` to launch a cluster with debugging enabled.

Usage: [Configure Logging and Debugging \(Optional\) \(p. 133\)](#)

`--resize-jobflow`

Adds a step to resize the cluster.

`--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.

Usage: [Add Steps to a Cluster \(p. 473\)](#)

## 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.

Usage: [Create Bootstrap Actions to Install Additional Software \(Optional\) \(p. 87\)](#)

--bootstrap-name *bootstrap\_NAME*  
Sets the name of the bootstrap action.

Usage: [Create Bootstrap Actions to Install Additional Software \(Optional\) \(p. 87\)](#)

## Tagging

--tag  
Manages tags associated with Amazon EMR resources.

Usage: [Tagging Amazon EMR Clusters \(p. 143\)](#)

## 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.

Usage: [Protect a Cluster from Termination \(p. 459\)](#)

--terminate  
Terminates the specified cluster or clusters.

Usage: [Terminate a Cluster \(p. 458\)](#)

## Command Line Interface Releases

The following table lists the releases and changes in CLI versions. The Amazon EMR CLI uses the release date as its version number.

| Release Date | Description                                                                                                                 |
|--------------|-----------------------------------------------------------------------------------------------------------------------------|
| 2014-05-15   | Adds --ec2-instance-ids-to-terminate option. Adds support for Signature Version 4 signing with CLI. Fixes a security issue. |
| 2013-12-10   | Adds support for Impala on Amazon EMR. For more information, see <a href="#">Analyze Data with Impala (p. 247)</a> .        |

| <b>Release Date</b> | <b>Description</b>                                                                                                                                                 |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2013-12-02          | Adds support for Amazon EMR tags. For more information, see <a href="#">Tagging Amazon EMR Clusters (p. 143)</a> .                                                 |
| 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.   |
| 2011-11-30          | Fixes support for Amazon 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 <code>list</code> option.                                                                         |
| 2011-12-08          | Adds support for Hive 0.7.                                                                                                                                         |
| 2011-11-11          | Fixes issues in the processing of <code>pig</code> and <code>hive</code> arguments and the <code>--main-class</code> argument to the <code>custom jar</code> step. |
| 2011-10-19          | Adds support for resizing running clusters. Substantially reworked 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, run the following commands from the command line:
  - 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:** August 15, 2014

| Change                                             | Description                                                                                                                                                                  | Release Date      |
|----------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
| AMI 3.1.1                                          | Amazon EMR supports AMI 3.1.1. For more information, see <a href="#">AMI Versions Supported in Amazon EMR (p. 56)</a> .                                                      | August 15, 2014   |
| AMI 2.4.7                                          | Amazon EMR supports AMI 2.4.7. For more information, see <a href="#">AMI Versions Supported in Amazon EMR (p. 56)</a> .                                                      | July 30, 2014     |
| AMI 2.4.6                                          | Amazon EMR supports AMI 2.4.6. For more information, see <a href="#">AMI Versions Supported in Amazon EMR (p. 56)</a> .                                                      | May 15, 2014      |
| AMI 3.1.0                                          | Amazon EMR supports AMI 3.1.0. For more information, see <a href="#">AMI Versions Supported in Amazon EMR (p. 56)</a> .                                                      | May 15, 2014      |
| AMI 2.4.5                                          | Amazon EMR supports AMI 2.4.5. For more information, see <a href="#">AMI Versions Supported in Amazon EMR (p. 56)</a> .                                                      | March 27, 2014    |
| Elastic Load Balancing Access Logs with Amazon EMR | Added a tutorial for processing access logs produced by Elastic Load Balancing. For more information, see <a href="#">Analyze Elastic Load Balancing Log Data (p. 393)</a> . | March 6, 2014     |
| AMI 3.0.4                                          | Amazon EMR supports AMI 3.0.4 and a connector for Amazon Kinesis. For more information, see <a href="#">AMI Versions Supported in Amazon EMR (p. 56)</a> .                   | February 20, 2014 |
| AMI 3.0.3                                          | Amazon EMR supports AMI 3.0.3. For more information, see <a href="#">AMI Versions Supported in Amazon EMR (p. 56)</a> .                                                      | February 11, 2014 |
| Hive 0.11.0.2                                      | Amazon EMR supports Hive 0.11.0.2. For more information, see <a href="#">Supported Hive Versions (p. 213)</a> .                                                              | February 11, 2014 |
| Impala 1.2.1                                       | Amazon EMR supports Impala 1.2.1 with Hadoop 2. For more information, see <a href="#">Analyze Data with Impala (p. 247)</a> .                                                | December 12, 2013 |

| Change                 | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | Release Date      |
|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
| AMI 3.0.2              | Amazon EMR supports AMI 3.0.2. For more information, see <a href="#">AMI Versions Supported in Amazon EMR (p. 56)</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | December 12, 2013 |
| Amazon EMR tags        | Amazon EMR supports tagging on Amazon EMR clusters. For more information, see <a href="#">Tagging Amazon EMR Clusters (p. 143)</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | December 5, 2013  |
| CLI version 2013-12-02 | Adds support for Amazon EMR tags. For more information, see <a href="#">Command Line Interface Releases (p. 596)</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | December 5, 2013  |
| AMI 3.0.1              | Amazon EMR supports AMI 3.0.1. For more information, see <a href="#">AMI Versions Supported in Amazon EMR (p. 56)</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | November 8, 2013  |
| 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"> <li>• Resizing a running cluster (that is, adding or removing instances)</li> <li>• Cloning the launch configurations for running or terminated clusters</li> <li>• Hadoop 2 support, including custom Amazon CloudWatch metrics</li> <li>• Targeting specific Availability Zones</li> <li>• Creating clusters with IAM roles</li> <li>• Submitting multiple steps (before and after cluster creation)</li> <li>• New console help portal with integrated documentation search</li> </ul> | November 6, 2013  |
| MapR 3.0.2             | Amazon EMR supports MapR 3.0.2. For more information, see <a href="#">Using the MapR Distribution for Hadoop (p. 149)</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | November 6, 2013  |
| Hadoop 2.2.0           | Amazon EMR supports Hadoop 2.2.0. For more information, see <a href="#">Hadoop 2.2.0 New Features (p. 81)</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | October 29, 2013  |
| AMI 3.0.0              | Amazon EMR supports AMI 3.0.0. For more information, see <a href="#">AMI Versions Supported in Amazon EMR (p. 56)</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | October 29, 2013  |
| CLI version 2013-10-07 | Maintenance update for the Amazon EMR CLI. For more information, see <a href="#">Command Line Interface Releases (p. 596)</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | October 7, 2013   |
| AMI 2.4.2              | Amazon EMR supports AMI 2.4.2 For more information, see <a href="#">AMI Versions Supported in Amazon EMR (p. 56)</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | October 7, 2013   |
| AMI 2.4.1              | Amazon EMR supports AMI 2.4.1 For more information, see <a href="#">AMI Versions Supported in Amazon EMR (p. 56)</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | August 20, 2013   |
| Hive 0.11.0.1          | Amazon EMR supports Hive 0.11.0.1. For more information, see <a href="#">Supported Hive Versions (p. 213)</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | August 2, 2013    |
| Hive 0.11.0            | Amazon EMR supports Hive 0.11.0. For more information, see <a href="#">Supported Hive Versions (p. 213)</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | August 1, 2013    |

| Change                                                                     | Description                                                                                                                                                                                                                                                                   | Release Date   |
|----------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|
| Pig 0.11.1.1                                                               | Amazon EMR supports Pig 0.11.1.1. For more information, see <a href="#">Supported Pig Versions (p. 273)</a> .                                                                                                                                                                 | August 1, 2013 |
| AMI 2.4                                                                    | Amazon EMR supports AMI 2.4. For more information, see <a href="#">AMI Versions Supported in Amazon EMR (p. 56)</a> .                                                                                                                                                         | August 1, 2013 |
| MapR 2.1.3                                                                 | Amazon EMR supports MapR 2.1.3. For more information, see <a href="#">Using the MapR Distribution for Hadoop (p. 149)</a> .                                                                                                                                                   | August 1, 2013 |
| MapR M7 Edition                                                            | Amazon EMR supports MapR M7 Edition. For more information, see <a href="#">Using the MapR Distribution for Hadoop (p. 149)</a> .                                                                                                                                              | July 11, 2013  |
| CLI version 2013-07-08                                                     | Maintenance update to the Amazon EMR CLI version 2013-07-08. For more information, see <a href="#">Command Line Interface Releases (p. 596)</a> .                                                                                                                             | 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 <a href="#">Supported Pig Versions (p. 273)</a> .                                                                                                            | July 1, 2013   |
| Hive 0.8.1.8                                                               | Amazon EMR supports Hive 0.8.1.8. For more information, see <a href="#">Supported Hive Versions (p. 213)</a> .                                                                                                                                                                | June 18, 2013  |
| AMI 2.3.6                                                                  | Amazon EMR supports AMI 2.3.6. For more information, see <a href="#">AMI Versions Supported in Amazon EMR (p. 56)</a> .                                                                                                                                                       | May 17, 2013   |
| Hive 0.8.1.7                                                               | Amazon EMR supports Hive 0.8.1.7. For more information, see <a href="#">Supported Hive Versions (p. 213)</a> .                                                                                                                                                                | 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 <a href="#">AMI Versions Supported in Amazon EMR</a> .                                                                                                                                                               | April 26, 2013 |
| M1 Medium Amazon EC2 Instances                                             | Amazon EMR supports m1.medium instances. For more information, see <a href="#">Hadoop 2.2.0 and 2.4.0 Default Configuration (p. 520)</a> .                                                                                                                                    | April 18, 2013 |
| MapR 2.1.2                                                                 | Amazon Elastic MapReduce supports MapR 2.1.2. For more information, see <a href="#">Using the MapR Distribution for Hadoop (p. 149)</a> .                                                                                                                                     | April 18, 2013 |
| AMI 2.3.4                                                                  | Deprecated.                                                                                                                                                                                                                                                                   | April 16, 2013 |
| AWS GovCloud (US)                                                          | Adds support for AWS GovCloud (US). For more information, see <a href="#">AWS GovCloud (US)</a> .                                                                                                                                                                             | 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 <a href="#">Launch an Amazon EMR cluster with MapR using the console (p. 150)</a> . | March 19, 2013 |

| Change                        | Description                                                                                                                                                                                                                                             | Release Date      |
|-------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
| 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 <a href="#">Amazon VPC</a> .                                                             | March 11, 2013    |
| AMI 2.3.3                     | Amazon Elastic MapReduce supports AMI 2.3.3. For more information, see <a href="#">AMI Versions Supported in Amazon EMR</a> .                                                                                                                           | March 1, 2013     |
| High I/O Instances            | Amazon Elastic MapReduce supports hi1.4xlarge instances. For more information, see <a href="#">Hadoop 2.2.0 and 2.4.0 Default Configuration (p. 520)</a> .                                                                                              | February 14, 2013 |
| AMI 2.3.2                     | Amazon Elastic MapReduce supports AMI 2.3.2. For more information, see <a href="#">AMI Versions Supported in Amazon EMR</a> .                                                                                                                           | February 7, 2013  |
| New introduction and tutorial | Added sections that describe Amazon EMR and a tutorial that walks you through your first streaming cluster. For more information, see <a href="#">What is Amazon EMR? (p. 1)</a> and <a href="#">Get Started: Count Words with Amazon EMR (p. 13)</a> . | January 9, 2013   |
| CLI Reference                 | Added CLI reference. For more information, see <a href="#">Command Line Interface Reference for Amazon EMR (p. 579)</a> .                                                                                                                               | January 8, 2013   |
| AMI 2.3.1                     | Amazon Elastic MapReduce supports AMI 2.3.1. For more information, see <a href="#">AMI Versions Supported in Amazon EMR</a> .                                                                                                                           | December 24, 2012 |
| High Storage Instances        | Amazon Elastic MapReduce supports hs1.8xlarge instances. For more information, see <a href="#">Hadoop 2.2.0 and 2.4.0 Default Configuration (p. 520)</a> .                                                                                              | December 20, 2012 |
| IAM Roles                     | Amazon Elastic MapReduce supports IAM Roles. For more information, see <a href="#">Configure IAM Roles for Amazon EMR (p. 125)</a> .                                                                                                                    | December 20, 2012 |
| Hive 0.8.1.6                  | Amazon Elastic MapReduce supports Hive 0.8.1.6. For more information, see <a href="#">Supported Hive Versions (p. 213)</a> .                                                                                                                            | December 20, 2012 |
| AMI 2.3.0                     | Amazon Elastic MapReduce supports AMI 2.3.0. For more information, see <a href="#">AMI Versions Supported in Amazon EMR</a> .                                                                                                                           | December 20, 2012 |
| AMI 2.2.4                     | Amazon Elastic MapReduce supports AMI 2.2.4. For more information, see <a href="#">AMI Versions Supported in Amazon EMR</a> .                                                                                                                           | December 6, 2012  |
| AMI 2.2.3                     | Amazon Elastic MapReduce supports AMI 2.2.3. For more information, see <a href="#">AMI Versions Supported in Amazon EMR</a> .                                                                                                                           | November 30, 2012 |
| Hive 0.8.1.5                  | Amazon Elastic MapReduce supports Hive 0.8.1.5. For more information, see <a href="#">Analyze Data with Hive (p. 202)</a> .                                                                                                                             | November 30, 2012 |
| Asia Pacific (Sydney) Region  | 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 <a href="#">Configure IAM User Permissions (p. 119)</a> .                                                                                          | October 1, 2012   |

| Change                                               | Description                                                                                                                                                                                                                                                                                                                            | Release Date       |
|------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------|
| 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"> <li>Fixes an issue with HBase backup functionality.</li> <li>Enables multipart upload by default for files larger than the Amazon S3 block size specified by <code>fs.s3n.blockSize</code>. For more information, see <a href="#">Configure Multipart Upload for Amazon S3 (p. 104)</a>.</li> </ul> | August 30, 2012    |
| AMI 2.1.4                                            | <ul style="list-style-type: none"> <li>Fixes issues in the native Amazon S3 file system.</li> <li>Enables multipart upload by default. For more information, see <a href="#">Configure Multipart Upload for Amazon S3 (p. 104)</a>.</li> </ul>                                                                                         | August 30, 2012    |
| 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 <a href="#">Supported Hadoop Versions (p. 79)</a> .                                                                                                                                                                                                                                | 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 <a href="#">Choose a Machine Image (p. 51)</a> .                                                                                     | 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 <a href="#">Store Data with HBase (p. 289)</a> .                                                                                                                                            | 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 <a href="#">Using the MapR Distribution for Hadoop (p. 149)</a> .                                                                                                                             | 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 <a href="#">Connect to the Cluster (p. 446)</a> .                                                                                                                                                                | June 12, 2012      |
| Hive 0.8.1                                           | Amazon Elastic MapReduce supports Hive 0.8.1. For more information, see <a href="#">Analyze Data with Hive (p. 202)</a> .                                                                                                                                                                                                              | May 30, 2012       |

| Change                                          | Description                                                                                                                                                                                                                                                                                             | Release Date      |
|-------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
| HParser                                         | Added information about running Informatica HParser on Amazon EMR. For more information, see <a href="#">Parse Data with HParser (p. 148)</a> .                                                                                                                                                         | April 30, 2012    |
| AMI 2.0.5                                       | Enhancements to performance and other updates. For more information, see <a href="#">AMI Versions Supported in Amazon EMR (p. 56)</a> .                                                                                                                                                                 | 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 <a href="#">Pig Version Details (p. 277)</a> .                                                                                       | April 9, 2012     |
| Pig versioning                                  | Amazon Elastic MapReduce supports the ability to specify the Pig version when launching a cluster. For more information, see <a href="#">Process Data with Pig (p. 273)</a> .                                                                                                                           | April 9, 2012     |
| Hive 0.7.1.4                                    | Amazon Elastic MapReduce supports Hive 0.7.1.4. For more information, see <a href="#">Analyze Data with Hive (p. 202)</a> .                                                                                                                                                                             | April 9, 2012     |
| 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 dynamodb.retry.duration variable which you can use to configure the timeout duration for retrying Hive queries. This version of Hive also supports setting the DynamoDB endpoint from within the Hive command-line application. | 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 CloudWatch 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 <a href="#">Supported Hadoop Versions (p. 79)</a> .                                                                                                                                                                                              | December 11, 2011 |
| Pig 0.9.1                                       | Support for Pig 0.9.1. For more information see <a href="#">Supported Pig Versions (p. 273)</a> .                                                                                                                                                                                                       | December 11, 2011 |

| Change                                | Description                                                                                                                                                                                                                                                                                | Release Date      |
|---------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
| 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 <a href="#">Choose a Machine Image (p. 51)</a> .                              | 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 <a href="#">Select a Amazon VPC Subnet for the Cluster (Optional) (p. 137)</a> .                      | December 11, 2011 |
| Spot Instances                        | Support for launching cluster instance groups as Spot Instances added. For more information, see <a href="#">Lower Costs with Spot Instances (Optional) (p. 37)</a> .                                                                                                                      | August 19, 2011   |
| Hive 0.7.1                            | Support for Hive 0.7.1 added. For more information, see <a href="#">Supported Hive Versions (p. 213)</a> .                                                                                                                                                                                 | July 25, 2011     |
| Termination Protection                | Support for a new Termination Protection feature. For more information, see <a href="#">Protect a Cluster from Termination (p. 459)</a> .                                                                                                                                                  | April 14, 2011    |
| Tagging                               | Support for Amazon EC2 tagging. For more information, see <a href="#">View Cluster Instances in Amazon EC2 (p. 422)</a> .                                                                                                                                                                  | March 9, 2011     |
| IAM Integration                       | Support for AWS Identity and Access Management. For more information, see <a href="#">Configure IAM User Permissions (p. 119)</a> and <a href="#">Configure IAM User Permissions (p. 119)</a> .                                                                                            | February 21, 2011 |
| Elastic IP Support                    | Support for Elastic IP addresses. For more information, see <a href="#">Associate an Elastic IP Address with a Cluster (p. 477)</a> and <a href="#">Associate an Elastic IP Address with a Cluster (p. 477)</a> .                                                                          | February 21, 2011 |
| Environment Configuration             | Expanded sections on Environment Configuration and Performance Tuning. For more information, see <a href="#">Create Bootstrap Actions to Install Additional Software (Optional) (p. 87)</a> .                                                                                              | February 21, 2011 |
| Distributed Cache                     | For more information about using DistributedCache to upload files and libraries, see <a href="#">Import files using Distributed Cache (p. 106)</a> .                                                                                                                                       | February 21, 2011 |
| How to build modules using Amazon EMR | For more information, see <a href="#">Build Binaries Using Amazon EMR (p. 161)</a> .                                                                                                                                                                                                       | February 21, 2011 |
| Comparison of cluster types           | For more information, see <a href="#">Choose the Type of Cluster to Run (p. 32)</a> .                                                                                                                                                                                                      | February 21, 2011 |
| Amazon S3 multipart upload            | Support of Amazon S3 multipart upload through the AWS SDK for Java. For more information, see <a href="#">Configure Multipart Upload for Amazon S3 (p. 104)</a> .                                                                                                                          | 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 <a href="#">Analyze Data with Hive (p. 202)</a> . | December 8, 2010  |

| Change                          | Description                                                                                                                                                                                                                                                                 | Release Date      |
|---------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
| JDBC Drivers for Hive           | Support for JDBC with Hive 0.5 and Hive 0.7. For more information, see <a href="#">Use the Hive JDBC Driver (p. 243)</a> .                                                                                                                                                  | December 8, 2010  |
| Support HPC                     | Support for cluster compute instances. For more information, see <a href="#">Virtual Server Configurations (p. 36)</a> .                                                                                                                                                    | November 14, 2010 |
| Bootstrap Actions               | Expanded content and samples for bootstrap actions. For more information, see <a href="#">Create Bootstrap Actions to Install Additional Software (Optional) (p. 87)</a> .                                                                                                  | November 14, 2010 |
| Cascading clusters              | Description of Cascading cluster support. For more information, see <a href="#">Launch a Cascading Cluster (p. 177)</a> and <a href="#">Process Data with a Cascading Cluster (p. 177)</a> .                                                                                | 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 <a href="#">What is Amazon EMR? (p. 1)</a> , <a href="#">Resize a Running Cluster (p. 464)</a> , and <a href="#">Resize a Running Cluster (p. 464)</a> . | October 19, 2010  |
| Appendix: Configuration Options | Expanded information on configuration options available in Amazon EMR. For more information, see <a href="#">Hadoop Configuration Reference (p. 515)</a> .                                                                                                                  | October 19, 2010  |
| Guide revision                  | This release features a reorganization of the <i>Amazon Elastic MapReduce Developer Guide</i> .                                                                                                                                                                             | October 19, 2010  |