
Amazon Elastic MapReduce

Developer Guide

API Version 2009-03-31



Amazon Elastic MapReduce: Developer Guide

Copyright © 2009 Amazon Web Services LLC or its affiliates. All rights reserved.

Table of Contents

Welcome	1
What's New	4
Introduction to Amazon Elastic MapReduce	6
Using the AWS Management Console	21
How to Create a Key Pair	21
How to Create a Job Flow	23
How to Create a Job Flow Using a Custom JAR	23
How to Create a Job Flow Using Streaming	27
How to Create a Job Flow Using Pig	32
How to Create a Job Flow Using Hive	36
How to Use Additional Files and Libraries With the Mapper or Reducer	40
How to View the Details of a Job Flow	40
How to Terminate a Job Flow	41
Additional Console Tutorials	42
Using the Command Line Interface	43
How to Download and Install Ruby and the Command Line Interface	43
How to Download, Install, and Configure the Elastic MapReduce CLI	44
How to List All Command Options	46
How to Create a Job Flow	47
How to Use Additional Files and Libraries With the Mapper or Reducer	51
How to List Job Flows	52
How to Get Information About a Job Flow	53
How to Add Steps to a Job Flow	53
How to Terminate a Job Flow	54
Additional CLI Tutorials	54
Making API Requests	55
AWS Library Support	55
Starting and Managing Job Flows Using the API	58
Amazon Elastic MapReduce Libraries	58
How to Start a Job Flow	59
Sample Applications	59
Job Flows with Multiple Steps	60
EC2 Instances	60
Availability Zones	60
Args Argument	61
How to Add Steps to a Job Flow	61
How to Use Additional Files and Libraries With the Mapper or Reducer	62
How to Get Information About a Job Flow	62
Job Flow States	63
Step States	63
How to Terminate a Job Flow	64
Monitoring and Troubleshooting Job Flows	65
Log Files	65
How to Monitor Job Flows	66
How to Monitor Job Flow Status Using SSH	67
How to Install FoxyProxy	68
How to Download Job Flow Logs from Amazon S3	71
How to Use the Hadoop User Interface	71
Troubleshooting	76
How to Debug Job Flows Starting With No Steps	76
How to Debug Job Flows That Have Steps	77
How to Debug Job Flows Using Log Files	77
Checking Hadoop Failures	79
Advanced Topics	81
EC2 Cluster Tuning	81

What We Expect in Your JAR	82
Distributed Cache	82
Hadoop Data Compression	88
How to Process Gzipped Files	88
hadoop.tmp.dir	88
How to Process Nested Directories of Files	89
Glossary	91
Document Conventions	94
Index	96

Welcome

Topics

- [Who Should Read this Guide \(p. 1\)](#)
- [How to Give Us Feedback \(p. 2\)](#)
- [How this Guide Is Organized \(p. 2\)](#)
- [Amazon Elastic MapReduce Resources \(p. 2\)](#)

This is the *Amazon Elastic MapReduce Developer Guide*. This section describes who should read this guide, how the guide is organized, and other resources related to this web service.

Amazon Elastic MapReduce, Amazon Elastic Compute Cloud, and Amazon Simple Storage Service are sometimes referred to within this guide as "Elastic MapReduce," "EC2," and "Amazon S3," respectively. All copyrights and legal protections still apply.

For a description of what's new in this release of the Amazon Elastic MapReduce service, see [What's New \(p. 4\)](#).

Who Should Read this Guide

This guide is for developers and for the community of researchers and data analysts that need to process vast amounts of data efficiently and cost-effectively.

Required Knowledge and Skills

Elastic MapReduce offers a console, a command line interface, and an API. Each interface requires a slightly different set of skills. The console and command line interface let you execute Elastic MapReduce functionality without having to write full applications. Both of these interfaces, however, require you to write your own [mapper](#) and [reducer](#) executables in the programming language of your choice.

Of the three interfaces, the API requires the greatest programming skills and knowledge of Hadoop. Even though Elastic MapReduce hides a lot of the Hadoop configuration details, knowing the basics of Hadoop is important.

Developers using the API should be familiar with the following:

- XML (for an overview, go to the [W3 Schools XML Tutorial](#))
- Basic understanding of web services (for an overview, go to the [W3 Schools Web Services Tutorial](#))
- A programming language for writing mapper and reducer executables and for consuming Elastic MapReduce responses
- Hadoop (for more information, go to <http://hadoop.apache.org/core/>)
- Familiarity with EC2 and Amazon S3 (for more information, see the [Amazon Elastic Compute Cloud Developer Guide](#) and the [Amazon Simple Storage Service Developer Guide](#), respectively)

How to Give Us Feedback

The online version of this guide provides a link at the top of each page that enables you to enter feedback about this guide. We strive to make our guides as complete, error free, and easy to read as possible. You can help by giving us feedback. Thank you in advance!



How this Guide Is Organized

This guide is organized into several major sections described in the following table.

Information	Relevant Sections
What's new in this release	What's New (p. 4)
Amazon Elastic MapReduce concepts	Introduction to Amazon Elastic MapReduce (p. 6)
Completing Elastic MapReduce tasks using a console	Using the AWS Management Console (p. 21)
Completing Elastic MapReduce tasks using a command line interface	Amazon Elastic MapReduce Command Line Interface (p. 43)
Completing Elastic MapReduce tasks programmatically using the API	Starting and Managing a Job Flows Using the API (p. 58)
Monitor and troubleshoot Amazon Elastic MapReduce job flows	Monitoring and Troubleshooting Job Flows (p. 65)

In addition, there is a glossary, an overview of our typographical conventions, and an index. Each section is written to stand on its own, so you should be able to look up the information you need and go back to work. However, you can also read through the major sections sequentially to get in-depth knowledge about the Amazon Elastic MapReduce.

Amazon Elastic MapReduce Resources

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

Resource	Description
Amazon Elastic MapReduce Getting Started Guide	The Getting Started Guide provides a quick tutorial of the service based on a simple use case. Examples and instructions for the console are included.
Amazon Elastic MapReduce API Reference	The API Reference describes Amazon Elastic MapReduce operations, errors, and data structures.
Amazon Elastic MapReduce Technical FAQ	The FAQ covers the top 20 questions developers have asked about this product.
Release notes	The release notes give a high-level overview of the current release. They specifically note any new features, corrections, and known issues.
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.
AWS Management Console	Location of the Amazon Elastic MapReduce console.
Discussion Forums	A community-based forum for developers to discuss technical questions related to Amazon Web Services.
AWS Support Center	The home page for AWS Technical Support, including access to our Developer Forums, Technical FAQs, Service Status page, and Premium Support .
Premium Support	The primary web page for information about AWS Premium Support, a one-on-one, fast-response support channel to help you build and run applications on AWS Infrastructure Services.
Amazon Elastic MapReduce product information	The primary web page for information about Amazon Elastic MapReduce.
Contact Us	A central contact point for inquiries concerning AWS billing, account, events, abuse etc.
Conditions of Use	Detailed information about the copyright and trademark usage at Amazon.com and other topics.

What's New

This What's New is associated with the 2009-03-31 version of the Amazon Elastic MapReduce web service. This guide was last updated on 2009-12-02.

Change	Description	Release Date
New Regions	Amazon Elastic MapReduce now supports the US-West (Northern California Region) and US-East (Northern Virginia). For more information, see Regions (p. 18).	2 December 2009
AWS SDK for .NET	AWS now provides libraries, sample code, tutorials, and other resources for software developers who prefer to build applications using .NET language-specific APIs instead of REST or SOAP. These libraries provide basic functions (not included in the REST or SOAP APIs), such as request authentication, request retries, and error handling so that it's easier to get started. For more information about language-specific libraries and resources, see AWS Library Support (p. 55).	2 December 2009
Support for Hive	Hive is an open source library that runs on top of Hadoop. The library takes SQL-like commands in a language called Hive QL and converts those commands into MapReduce jobs. Using Hive saves you the trouble of programming MapReduce applications in lower-level computing languages, such as Java. For more information, see Hive (p. 16).	1 October 2009
Technical documents reorganized	The API reference has been split out of the <i>Amazon Elastic MapReduce Developer Guide</i> . Now, on the documentation landing page , you can select the document you want to view. When viewing the documents online, the links in one document will take you, when appropriate, to one of the other guides.	1 October 2009
Support for Pig	Pig is an open source, Apache library that runs on top of Hadoop. The library takes SQL-like commands in a language called Pig Latin and converts those commands into a graph of MapReduce jobs, which is executed on data in an Amazon EC2 cluster. So now, in addition to running MapReduce jobs using a JAR or stream you can use Pig in programmatic or interactive modes. For more information, see Pig (p. 15).	1 August 2009

Change	Description	Release Date
Enhancement	Added a new endpoint for the European community. For the new WSDL location, see API Reference . For information about configuring your region in the Management Console, see How to Create a Job Flow (p. 23) . For information about configuring your region in the Command Line Interface, see the steps for editing the credentials.json file (p. 44) .	7 July 2009
New guide	This is the first publication of this guide.	2 April 2009

Introduction to Amazon Elastic MapReduce

Topics

- [Overview of Elastic MapReduce](#) (p. 6)
- [Elastic MapReduce Concepts](#) (p. 10)
- [Getting Help](#) (p. 20)

This introduction to Amazon Elastic MapReduce (Elastic MapReduce) provides a detailed summary of this web service. After reading this section, you should have a good idea what it offers and how it can fit in with your business.

Overview of Elastic MapReduce

Topics

- [Business Model](#) (p. 6)
- [Features](#) (p. 7)
- [Architectural Overview of Elastic MapReduce](#) (p. 9)

This overview describes the business model and the major features of Elastic MapReduce.

Business Model

Elastic MapReduce is a web service that makes it easy for researchers, data analysts, and developers to efficiently and cost-effectively process vast amounts of data using the Amazon Elastic Compute Cloud (EC2) and Amazon Simple Storage Service (Amazon S3). Whether you already have an end-to-end data processing [job flow](#) or only a data set, Elastic MapReduce lets you focus on analyzing

your data instead of the mechanics of the processing, including managing a cluster of computers in a complex distributed software development environment.

All Elastic MapReduce customers can use our simple web console or command line interface to execute most of the functionality available in the Elastic MapReduce API. Developers can also programmatically access the distributed processing power of the Elastic MapReduce API to process large data sets using Hadoop technology.

Pricing

Elastic MapReduce pricing is similar to all AWS services and is available on a pay-as-you-go basis. You pay only for what you use. There is no minimum fee, and there are no upfront contracts or commitments. Elastic MapReduce pricing is in addition to normal EC2 and Amazon S3 pricing. To estimate your bill, go to the [AWS Simple Monthly Calculator](#).

Billing commences when Elastic MapReduce determines that all EC2 instances are ready to process your job flow. Billing ends when the job flow terminates (normally or from job flow failure) or when you send a `TerminateJobFlow` request. For example, assume you launch a job flow with 100 instances. The instances begin booting immediately but they don't start at the same moment. Elastic MapReduce tracks each instance and checks it into the cluster when it's ready to process data.

Elastic MapReduce doesn't start charging until the cluster is ready to process data. After the job flow starts, Elastic MapReduce charges for its hours of execution at the beginning of each hour relative to the time the job flow started processing data. The minimum billing increment is one hour. So, one minute costs the same as fifty-nine minutes.

For billing information, refer to your account activity page. For the latest Elastic MapReduce prices, go to the [Elastic MapReduce detail page](#). For more information about EC2 pricing, go to [EC2 Pricing](#). For more information about Amazon S3 pricing, go to [Amazon S3 Pricing](#).

Features

Topics

- [JAR and Hadoop Streaming Support \(p. 7\)](#)
- [Multiple Sequential Steps Support \(p. 7\)](#)
- [Cluster Automation \(p. 8\)](#)
- [Reliability \(p. 8\)](#)
- [Data Security \(p. 8\)](#)
- [Processing Speed \(p. 8\)](#)
- [Data Storage \(p. 8\)](#)
- [Console, Command Line Interface, and API \(p. 9\)](#)

This section describes the major features of Elastic MapReduce.

JAR and Hadoop Streaming Support

Elastic MapReduce supports mapper and reducer executables written in all programming languages.

Multiple Sequential Steps Support

Elastic MapReduce supports job flows with multiple, sequential steps, and the ability to add steps while a job flow runs.

Cluster Automation

Elastic MapReduce handles the complexities of cluster computing, including dealing with networking, hardware, instance provisioning, scheduling, load balancing, Hadoop configuration, and health monitoring.

Reliability

Elastic MapReduce manages an EC2 cluster of server instances using Amazon's network infrastructure and data centers. Elastic MapReduce uses Hadoop's fault-tolerant software (version 0.18.3) as its data processing engine.

Hadoop splits the data into multiple subsets and assigns each subset to more than one EC2 instance. So, if an EC2 instance fails to process one subset of data, Elastic MapReduce uses the results of another EC2 instance.

Data Security

Elastic MapReduce provides [authentication](#) mechanisms to ensure that data stored in Amazon S3 is secured against unauthorized access. Unless you specify otherwise, only the AWS account owner can access the data he or she uploads to Amazon S3.

You can send data to Amazon S3 using the secure HTTPS protocol. Elastic MapReduce always uses a secure channel to send data between Amazon S3 and EC2. For added security, you may choose to encrypt your data before uploading it to Amazon S3. You can use any common data compression tool to do this. You must then add a decryption [step](#) to your job flow that runs before other data processing steps so that the job flow can work on unencrypted data.

Elastic MapReduce starts your instances in two security groups: one for the master and another for the slaves. The master security group has a port open for communication with our service. It also has the ssh port open to allow you to ssh as the Hadoop user into the instances, using the keypair specified in the `RunJobFlow` command (or the equivalent console action or command line command). The slaves start in a separate security group that only allows interaction with the master instance. Since these are security groups within your account, you can reconfigure them using the standard EC2 tools or console.

Processing Speed

The setup time for your job flow depends on the number of EC2 instances in the cluster and the amount of data to process. Elastic MapReduce loads all data onto the EC2 instances before processing begins.

The time to run your job flow depends on several factors, including the computational complexity of the job flow, the amount of input data, and the number and type of EC2 instances in the cluster. One of the advantages of using Elastic MapReduce with EC2 is that you pay only for what you use, which makes it convenient and inexpensive to test the performance of your job flow on different cluster types and sizes. One effective way to determine the most appropriate instance type is to launch small clusters and benchmark your job flow.

Data Storage

Elastic MapReduce stores all input and output data in the cluster while processing your job flow. Therefore, the maximum amount of data that you can process within EC2 by a twenty-instance cluster is 34 TB (20 Extra Large instances x 1.69 TB of hard disk per EC2 instance = 34 TB). The practical limit depends on the number and type of EC2 instances in your compute cluster, and on the size of

your intermediate and final data because the input, intermediate, and output data sets reside on the EC2 instances while your job flow runs.

For more information about EC2 instances, see [EC2 \(p. 18\)](#).

Console, Command Line Interface, and API

Elastic MapReduce offers a variety of interfaces, including a console, a command line interface (CLI), and an API. Each interface offers a different balance of ease and functionality. The Elastic MapReduce console is the easiest to use but offers the least functionality. The API offers the most functionality but is the most difficult to use. The CLI offers both ease of use and excellent functionality. The interface you choose depends on your expertise in Hadoop and programming, and the functionality you require.

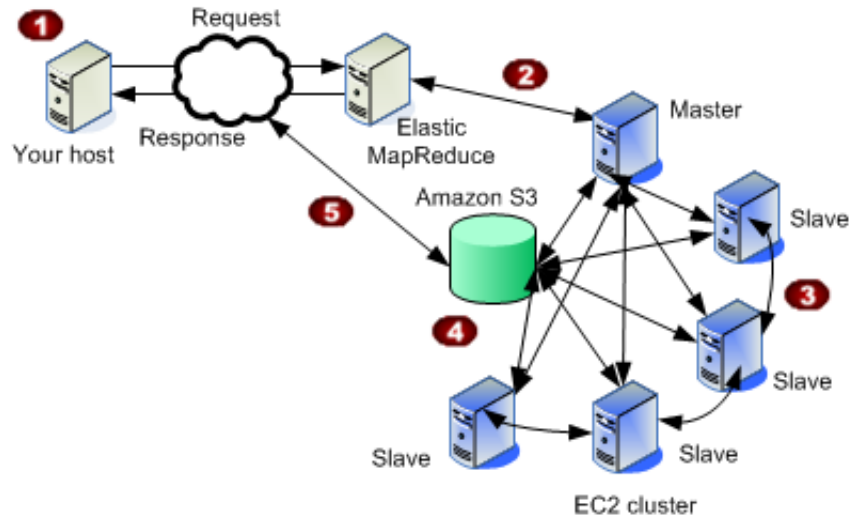
The following table compares the functionality of the interfaces.

Function	Console	CLI	API
Load data from Amazon S3 automatically	✓	✓	✓
Transfer processed data to Amazon S3 automatically	✓	✓	✓
Specify the number and type of EC2 instances that process the data	✓	✓	✓
Specify the MapReducer executable in multiple computer languages	✓	✓	✓
Provide verbose job flow details		✓	✓
Can run job flows with multiple steps		✓	✓
Add steps to job flows already running on Elastic MapReduce		✓	✓
Implement Hadoop data processing programmatically			✓

For more information about the console, see [Amazon Elastic MapReduce Console \(p. 21\)](#). For more information about the CLI, see [Elastic MapReduce Command Line Interface \(p. 43\)](#). For more information about the API, see the [Amazon Elastic MapReduce API Reference](#).

Architectural Overview of Elastic MapReduce

Elastic MapReduce works in conjunction with EC2 and Amazon S3, as shown in the following figure and table.



Elastic MapReduce Process

1	You upload into Amazon S3 the data you want to process along with the mapper and reducer executables that will process the data and then send a request to Elastic MapReduce to start a job flow.
2	Elastic MapReduce starts an EC2 cluster, which loads and runs Hadoop.
3	Hadoop executes a job flow by downloading data from Amazon S3 onto the cluster of slave instances.
4	Hadoop processes the data and then uploads the results from the cluster to Amazon S3.
5	You receive notification that the job flow is done and you download the processed data from Amazon S3.

Elastic MapReduce Concepts

Topics

- [Elastic MapReduce and Hadoop \(p. 11\)](#)
- [Master and Slave Nodes \(p. 12\)](#)
- [File Systems \(p. 13\)](#)
- [Amazon S3 Bucket Names \(p. 13\)](#)
- [Job Flow and Steps \(p. 14\)](#)
- [JAR, Streaming, Pig, and Hive \(p. 14\)](#)
- [Example Data, Mapper, and Reducer \(p. 18\)](#)
- [Regions \(p. 18\)](#)
- [EC2 \(p. 18\)](#)

This section describes the concepts and terminology you need to understand to use Elastic MapReduce effectively.

Elastic MapReduce and Hadoop

Topics

- [Comparing Hadoop and Elastic MapReduce \(p. 11\)](#)
- [MapReduce \(p. 11\)](#)

Elastic MapReduce uses Apache Hadoop (version 0.18.3 only), which is an open source Java software framework that supports massive data processing across a cluster of servers. Hadoop uses a programming model called *MapReduce* that divides a large data set into many small fragments. Hadoop distributes a data fragment and a copy of the MapReduce executable to each of the slave nodes in an EC2 cluster. Each slave node runs the MapReduce executable on its subset of the data. Hadoop then combines the results from all of the slave nodes into a finished output. Elastic MapReduce uploads that output into the Amazon S3 [bucket](#) you designate.

Typically, the processing involves performing relatively simple operations on very large amounts of data, for example, adding a watermark to 1,000,000 digital images. It is also typical to perform more than one process on the data. Each process is called a [step](#) and the sequence of steps is called a job flow. For example, if you encrypt your data on Amazon S3, the first step in the [job flow](#) would be to decrypt the data, the second step would be to process the data, and the final step might be to encrypt the data again so that it remains encrypted when uploaded to Amazon S3.

Without Elastic MapReduce, it is difficult for researchers, analysts, and developers to conduct data-intensive tasks requiring heavy processing power. You must provision hardware, invest time and resources to deploy, configure and maintain a data processing engine, and worry about data security and storage.

Elastic MapReduce removes this upfront complexity. Examples of job flows that can now be easily executed using Elastic MapReduce include web indexing, data mining, log file analysis, machine learning, and scientific simulation, among others. With Elastic MapReduce, you can use the console or command line interface to process large data sets easily using compute clusters of EC2 instances. As a software developer, you can use the API to programmatically run Elastic MapReduce job flows. The API enables you to implement more sophisticated data processing jobs and features. All three options, the console, command line interface, and API take the complexity out of using Hadoop.

For more information about Hadoop, go to <http://hadoop.apache.org/core/>.

Comparing Hadoop and Elastic MapReduce

Hadoop performs the MapReduce functions of dividing up the work among the slave instances in the cluster, tracking status, and combining the individual results into one output.

Elastic MapReduce takes care of provisioning an EC2 cluster, terminating it, moving the data between it and Amazon S3, and optimizing Hadoop. Elastic MapReduce removes most of the cumbersome details of setting up the hardware and networking required by the server cluster, such as monitoring that setup, configuring Hadoop, and executing the job flow. Together, Elastic MapReduce and Hadoop provide all of the power of Hadoop processing with the ease, low cost, scalability, and power afforded by Amazon S3 and EC2.

MapReduce

MapReduce is a combination of a mapper executable and a reducer executable that work together to process the data. The mapper executable processes the raw data into key/value pairs, which are called *intermediate results*. The reducer executable combines the intermediate results, applies additional algorithms, and produces the final output, as described in the following process.

MapReduce Process

1	Hadoop breaks a data set into multiple sets if the data set is too large to process quickly on a single EC2 instance.
2	Hadoop distributes the data files and the MapReduce executable to the EC2 instances in the cluster of slave nodes. Hadoop handles machine failures and manages network communication between the slave and master instances. In this way, developers do not need to know how to perform distributed programming or handle the details of data redundancy and failover.
3	The mapper uses an algorithm that you supply to parse the data into key/value pairs, which are passed to the reducer. Take, for example, a word counting job flow that counts the number of times a word appears in a document. The mapper might take each word in a document and assign it a value of 1. Each word is a key, in this case, and all values are 1.
4	The reducer collects the results from all of the mappers in the cluster, eliminates redundancy, and then performs an operation that you supply on all the values for one key, and outputs the results. Continuing with the previous example, the reducer would take all of the word counts from all of the slave instances in the cluster, add up the number of times each word was found in the document, and output that result.

You can write the executables in any programming language. Mapper and reducer applications written in Java are compiled into a JAR. Executables written in other programming languages use the Hadoop Streaming utility to implement the mapper and reducer algorithms.

The mapper executable should read the input from standard input and the reducer should output data through standard output. By default, each line of input/output represents a record and the first tab on each line of the output separates the key and value.

For more information about MapReduce, go to [How Map and Reduce operations are actually carried out](#) and [HadoopMapReduce](#).

Master and Slave Nodes

An EC2 cluster contains one [master node](#) and one or more [slave nodes](#), as shown in the following figure.



The master node helps coordinate the distribution of the MapReduce executable and subsets of the raw data to the slaves, tracks the status of each task the slaves perform, and monitors the health of the slaves. To monitor the progress of the job flow, you can SSH as the Hadoop user into the master node

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.

Each slave node has a copy of the MapReduce executable and receives a portion of the entire data set from Amazon S3. Each slave node processes the data, uploads it back to Amazon S3, and provides status [metadata](#) to the master node.

When you run a job flow, you must include in the request the number of slave instances you want to use as well as the computing size of each instance. Amazon EC2 instances vary in computing power and cache size. For more information, see [EC2 \(p. 18\)](#).

The default maximum number of instances you can specify is 20. If you need more instances, you can make a formal request. For more information, go to <http://aws.amazon.com/contact-us/ec2-request/>.

File Systems

Elastic MapReduce and Hadoop typically use two or more of the following file systems when processing a job flow:

- Hadoop Distributed File System (HDFS)
- Amazon S3 Native File System (S3N)
- Local file system
- Legacy Amazon S3 Block File System

Once the job flow completes, the HDFS file system ceases to exist. To make your output data persist, use the Amazon S3 Native File System. Hadoop uses HDFS to store temporary data and data passed from one step to another during the execution of a job flow.

The prefix in the input/output URI to Hadoop determines which file system to use. To use the Amazon S3 Native File System, use the following URI format:

```
s3n:[bucket-name]/[path-to-file-in-bucket]
```

For example, `s3n://my-bucket/path/to/my/file`. For more information about the Amazon S3 Native File System, go to <http://wiki.apache.org/hadoop/AmazonS3>.

To use the HDFS file system, use the following URI format:

```
hdfs:///path-to-data]
```

For example, `hdfs://home/hadoop/sampleInput2/`.

To directly access files on the local file system, use the following URI format:

```
file:[path-to-data]
```

Elastic MapReduce also supports the legacy Amazon S3 Block File System. Because this file system does not support saving files in their native format to Amazon S3, we recommend that you not use this file system except in special use cases.

To use the legacy S3 Block file system, use the following URI format:

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

Amazon S3 Bucket Names

In the course of using Elastic MapReduce you must create Amazon S3 buckets to hold the input and output data for your Hadoop processing. Amazon Elastic MapReduce uses the S3N Native File System

for Hadoop. This file system uses the “hostname” method for accessing data in Amazon S3, which places restrictions on bucket names used in Amazon Elastic MapReduce job flows.

To comply with Amazon S3 requirements, bucket names must:

- Contain lowercase letters, numbers, periods (.), underscores (_), and dashes (-)
- Start with a number or letter
- Be between 3 and 255 characters long
- Not be in an IP address style (e.g., "192.168.5.4")

To conform with DNS requirements, we recommend following these additional guidelines when creating buckets:

- Bucket names should not contain underscores (_)
- Bucket names should be between 3 and 63 characters long
- Bucket names should not end with a dash
- Two periods cannot be adjacent to one another
- Bucket names cannot contain dashes next to periods (e.g., "my-.bucket.com" and "my.-bucket" are invalid)

Job Flow and Steps

A [job flow](#) is a user-defined task that Elastic MapReduce performs. A job flow consists of one or more steps each of which must complete in sequence successfully for the job flow to finish. A step maps roughly to one algorithm that manipulates the data. A [step](#) is a MapReduce algorithm implemented as a Java JAR or a Hadoop streaming program written in Python, Ruby, Perl or C++. A job flow typically consists of multiple steps where the output of one step becomes the input of the next. For example, you might have a step that counts the number of times each word occurs in a the document and a second step that sorts the output from the first step based on the number of occurrences.



Tip

The AWS console only supports job flow with single steps.

JAR, Streaming, Pig, and Hive

Topics

- [JAR \(p. 15\)](#)
- [Streaming \(p. 15\)](#)
- [Pig \(p. 15\)](#)
- [Hive \(p. 16\)](#)

MapReduce is the algorithm you use to process your data using Hadoop. You can specify the algorithms in several ways:

- Using Java to create the algorithm and thereby creating a JAR to use for the Hadoop job flow
- Using some computer language other than Java to create the algorithm and thereby creating a Streaming job flow
- Using Pig, which enables you to use an SQL-like language called Pig Latin to create MapReduce algorithms for job flows

- Using Hive, which enables you to use an SQL-like language called QL to create MapReduce algorithms for job flows

JAR

A custom JAR job flow runs a compiled Java program that you have uploaded to Amazon S3. The program should be compiled against Hadoop 0.18.3 and should submit Hadoop jobs using the Hadoop `JobClient` interface.

Streaming

A streaming job flow runs a single Hadoop job consisting of map and reduce functions previously uploaded to Amazon S3. The functions may be implemented in one of the supported languages: Ruby, Perl, Python, PHP, R, Bash, C++.

Elastic MapReduce uses version 0.18.3 of Apache Hadoop streaming.



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/core/docs/r0.18.3/streaming.html>.

Pig

Topics

- [Interactive and Batch Modes \(p. 15\)](#)
- [Using JARs With Pig \(p. 16\)](#)
- [More Information \(p. 16\)](#)

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 job flows. Pig enables you to create database types of queries using familiar SQL-like commands and syntax thus avoiding the complexities of writing MapReduce algorithms using a lower level computer language, such as Java. While 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 task. Elastic MapReduce can use such scripts when you upload them to Amazon S3.

Interactive and Batch Modes

Elastic MapReduce enables you to run Pig scripts in two modes:

- Interactive
- Batch

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 load that script into Amazon S3. When you run a job flow using Pig, the first step is to download that script from Amazon S3 so that it can be used as the MapReduce job flow. When you use the AWS Management Console this download is done automatically for you. You use batch mode to run job flows in production.

The AWS Management Console and Elastic MapReduce command line enable you to set up a job flow to run in interactive or batch mode. Once you revise the Pig Latin script using the interactive mode, you should upload it to Amazon S3 and use the batch mode to run job flows. For more information about using ssh, go to [How to Monitor Job Flow Status Using SSH \(p. 67\)](#).

Using JARs With Pig

You can use custom JARs with Pig using the `REGISTER` command in your Pig script. The JAR can be on the local or a remote file system, such as Amazon S3. When the Pig script runs, Elastic MapReduce downloads the JAR automatically to the master node and then uploads the JAR to the Hadoop distributed cache. In this way, the JAR is automatically used as necessary by all instances in the cluster.

Using JARs With Pig

1	Upload your custom JAR into Amazon S3.
2	Use the <code>REGISTER</code> command in your Pig script to specify the location (in Amazon S3) of the custom JAR. <code>REGISTER s3://my-bucket/path/to/my/uploaded.jar;</code>

More Information

For more information about Pig and Pig Latin, go to:

- Pig tutorial—[Apache Log Analysis using Pig](#)
This tutorial shows you how to analyze Apache logs using Pig and Elastic MapReduce.
- Pig video tutorial—[Video that shows how to use a Pig script with the AWS Management Console and SSH](#)
This video tutorial shows you how to use Pig in batch and interactive modes with Elastic MapReduce.
- Sample Ppg script—[Parsing Logs with Apache Pig and Elastic MapReduce](#)
This document shows a sample Pig script.
- PiggyBank functions—[String Manipulation and DateTime Functions For Pig](#)
This is a list of five functions that AWS added to the Pig library.
- Pig Latin—[PigLatin](#)
This document explains the SQL-like language called Pig Latin. Pig converts Pig Latin into MapReduce job flows that you can then run using Elastic MapReduce.

Hive

Topics

- [Interactive and Batch Modes \(p. 17\)](#)
- [More Information \(p. 17\)](#)

A Hive program job flow is an easy way to start creating data processing applications. 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 that only supports primitive value types, such as dates, numbers,

and strings, values in Hive tables can be structured elements, such as JSON objects, any user-defined data type, or any function written in Java.

Executing Hive commands is different from executing SQL commands on a traditional database, such as an Oracle database. Hive and Hadoop are optimized to work on large data sets. Consequently, Hive is comparatively slow on small data sets.



Note

Whereas SQL queries running on traditional databases return in a matter of seconds, Hive queries return in a matter of minutes for small data sets or hours for very large data sets.

Interactive and Batch Modes

Elastic MapReduce enables you to run Hive scripts in two modes:

- Interactive
- Batch

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. The interactivity of this mode enables you to revise the Hive script quicker than you could using batch mode. Once you revise the Hive script in interactive mode, you should upload the script to Amazon S3 and use batch mode to run job flows in production.

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, or reference resources in Amazon S3, such as partitioned tables, JARs containing serialization or UDF functions, and map and reduce programs. Variables in Hive scripts use the dollar sign and curly braces, for example, `${VariableName}`. In the AWS Management Console, for example, you can pass parameter values into the Hive script using the **Extra Args** text box. For example, in the **Extra Args** text box you could put:

```
-d VariableName=Value
```

To use this value, your Hive script would include `${VariableName}`.



Note

AWS added the `-d` option to Hive so that you could pass values into the Hive script.

The AWS Management Console and the Elastic MapReduce command line support both interactive and batch modes. For more information about using Hive in the AWS Management Console, see [How to Create a Job Flow Using Hive \(p. 36\)](#). For more information about using Hive with the Elastic MapReduce command line, see [How to Create a Job Flow Using Hive \(p. 49\)](#).

More Information

For more information about Hive and QL, go to:

- Hive overview—<http://wiki.apache.org/hadoop/Hive>
- Hive video tutorial— <http://elasticmapreduce.s3.amazonaws.com/videos/hive-tutorial/HiveTutorial.html>
- Running Hive on Amazon ElasticMap Reduce—<http://developer.amazonwebservices.com/connect/entry.jspa?externalID=2857>
- Additional features of Hive in Amazon Elastic MapReduce—<http://developer.amazonwebservices.com/connect/entry.jspa?externalID=2856>

- Operating a data warehouse with Hive, Amazon Elastic MapReduce and Amazon SimpleDB —<http://developer.amazonwebservices.com/connect/entry.jspa?externalID=2854>
- Contextual advertising using Hive and Amazon Elastic MapReduce—<http://developer.amazonwebservices.com/connect/entry.jspa?externalID=2855>
- QL—[Hive Language Manual](#)
This document explains the SQL-like language called Hive QL. Hive converts QL into MapReduce algorithms for job flows that you can then run using Amazon Elastic MapReduce.

Example Data, Mapper, and Reducer

You can use the example mapper, reducer, and input data listed in the following table to try out and get a working knowledge of the Elastic MapReduce console, command line interface, and API.

Parameter	Value
Input location	s3://elasticmapreduce/samples/wordcount/input
Output location	s3://[<i>BucketYouCreate</i>]/output/[<i>Date</i>]
Mapper	s3://elasticmapreduce/samples/wordcount/wordSplitter.py \
Reducer	aggregate

The Aggregate library that comes with Hadoop includes the reducer, `aggregate`, which provides many basic reducer aggregations, such as sum, max, and min. For more information, go to [Working with the Hadoop Aggregate Package](#).

Before starting a job flow using these samples, you must create a bucket where Elastic MapReduce can upload the results. The format of the output URI is `s3://[BucketYouCreate]/output`. For more information, go to [Getting Set Up](#) in the *Amazon Elastic MapReduce Getting Started Guide*

Regions

You can choose the geographical Region where Amazon EC2 will process your data. You might choose a Region to optimize latency, minimize costs, or address regulatory requirements. Amazon Elastic MapReduce currently works with Amazon EC2 in the following Regions:

- **US-East (Northern Virginia)**—Uses Amazon S3 servers in northern Virginia
This Region-specific endpoint guarantees where your data will reside. To send an API request to this Region, use the endpoint `us-east-1.elasticmapreduce.amazonaws.com`.
- **US-West (Northern California)**—Uses Amazon S3 servers in northern California
This Region-specific endpoint guarantees where your data will reside. To send an API request to this Region, use the endpoint `us-west-1.elasticmapreduce.amazonaws.com`.
- **EU (Ireland)** —Uses Amazon S3 servers in Ireland
This Region-specific endpoint guarantees where your data will reside.

For more information about endpoints, see [Endpoints](#).

EC2

Elastic MapReduce enables you to choose the number and kind of EC2 instances that comprise the cluster that processes your job flow. EC2 offers two types of CPU instances:

- **Standard**—You can use EC2 standard instances for most applications

- **High-CPU**—These instances have proportionally more CPU resources than memory (RAM). You can use these instances for compute-intensive applications.

The following table describes all instance types offered by EC2.

Instance Type	RAM (GB)	Compute Units	Disk Drive (GB)	Platform (bits)	Name
Small (default)	1.7	1	160	32	m1.small
Large	7.5	4	850	64	m1.large
Extra Large	15	8	1690	64	m1.xlarge
High-CPU Medium	1.7	5	350	32	c1.medium
High-CPU Extra Large	7	20	1690	64	c1.xlarge
High-Memory Double Extra Large	34.2	13	850	64	m2.2xlarge
High-Memory Quadruple Extra Large	68.4	26	1690	64	m2.4xlarge

EC2 Key Pairs

When you start an EC2 instance, EC2 creates a 2048 bit RSA key-pair with a name you specify. EC2 stores the public key. Elastic MapReduce displays the private key on the AWS console and returns it in the response to a *DescribeJobFlows* request.

EC2 returns the private key as an unencrypted PEM-encoded PKCS#8 private key. The public/private key pair ensures that only you have access to instances your job flow launches. Whenever you launch an instance using the key-pair name, the public key becomes part of the instance metadata. This allows you to access the instance securely using the private key.

For more information, see [Creating a Key Pair](#) (p. 21).

Software Installed on Amazon Machine Images

An Amazon Machine Image (AMI) is the equivalent of a computer with a set of software products installed. AMI instances launched by Elastic MapReduce run Debian GNU/Linux version 5.0, code named Lenny. The instances include the following software:

- Hadoop v0.18.3
- Perl v5.10
- Ruby v1.8
- Python v2.5
- PHP v5.2
- Java v1.6
- R v2.7

Package Updates

Elastic MapReduce maintains a select set of the packages (including security patches) offered by Debian GNU/Linux. In particular, we update the package *r-recommended* as updates become

available. For more information about `r-recommended`, go to <http://packages.debian.org/search?keywords=r-recommended>.

Currently, users are not able to install Debian packages or package versions.

Location of Hadoop Software on Instances

The following table shows where important Hadoop files reside on EC2 instances started by Elastic MapReduce.

Software	Location
Hadoop 0.18.3	<code>/home/hadoop</code>
Hadoop configuration files including site configuration (hadoop-site.xml)	<code>/home/hadoop/conf</code>
Hadoop streaming JAR (hadoop-0.18-streaming.jar)	<code>/home/hadoop/contrib/streaming</code>



Note

The Hadoop configuration files set up by Elastic MapReduce should not be modified directly.

Getting Help

The [AWS Support Center](#) is the home page for AWS Technical Support. The page includes links to our [Discussion Forums](#) where you can ask questions of fellow developers and Amazon support personnel. On the same page, you'll find links to Elastic MapReduce Technical FAQ ([Amazon Elastic MapReduce Technical FAQ](#)), and the Service Status page. To get answers using the Elastic MapReduce documentation, go to the [Amazon Elastic MapReduce Developer Guide](#).

Using the AWS Management Console

Topics

- [How to Create a Key Pair \(p. 21\)](#)
- [How to Create a Job Flow \(p. 23\)](#)
- [How to View the Details of a Job Flow \(p. 40\)](#)
- [How to Terminate a Job Flow \(p. 41\)](#)
- [Additional Console Tutorials \(p. 42\)](#)

The AWS Management Console provides an easy way to perform Hadoop processing on data sets. The console does not provide all of the functionality of the command line interface or the API. For example, you cannot run a job flow that has multiple steps using the console. For a list of differences between the Elastic MapReduce interfaces, see [Console, Command Line Interface, and API \(p. 9\)](#).

Before you can run a job flow using the console, you need to create a key pair, which is a handle to an EC2 instance.

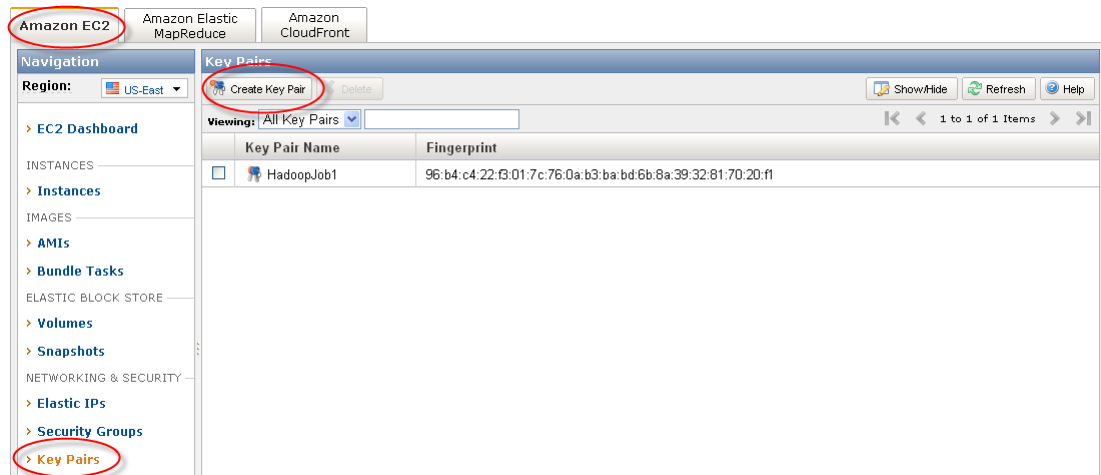
How to Create a Key Pair

A key pair is a handle you can use to log into an EC2 instance without having to know a password. The key pair has a public and private key. The private key is known only by you and guarantees that you are the only one who can log into the instance you created. In Elastic MapReduce you use the key pair to log into the master node in the EC2 cluster running your job flow. Logging into the master node enables you to monitor the status of the completion of your job flow by accessing either the log files or the user interface that Hadoop publishes on the master node.

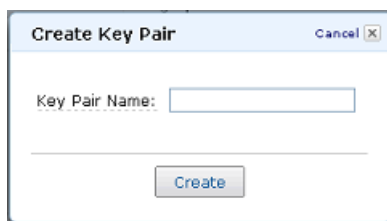
Elastic MapReduce does not provide the ability to create a key pair. You can, however, create one using the AWS Console.

To create a key pair

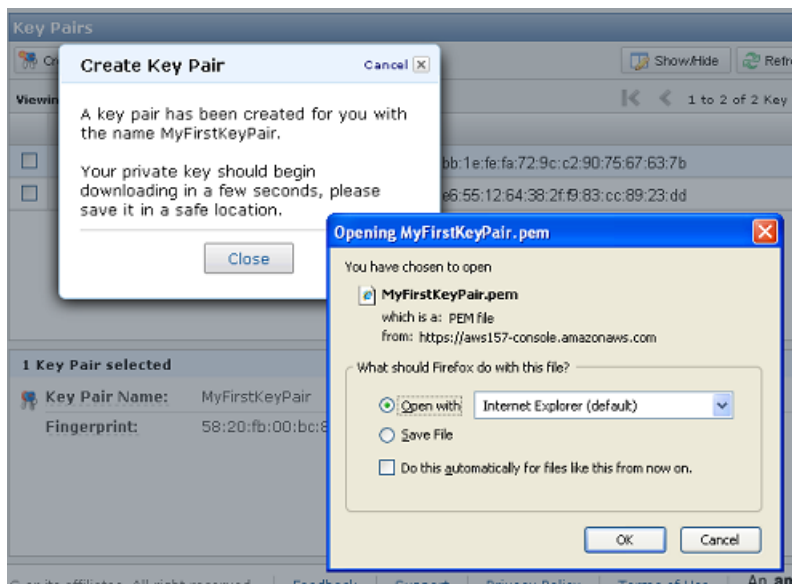
1. Open the AWS console by going to <https://console.aws.amazon.com/elasticmapreduce/home>. The console appears.



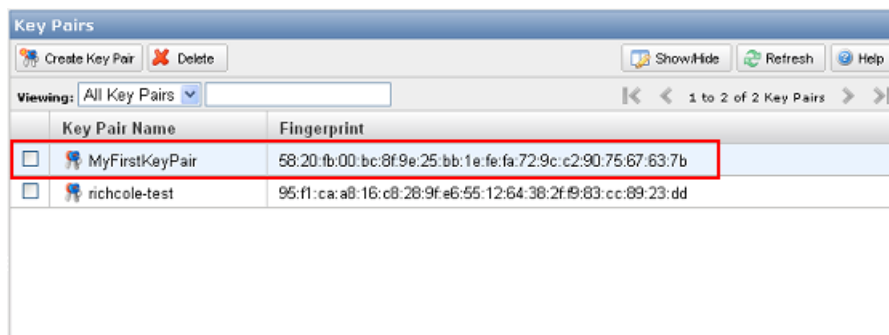
2. Click the **EC2** tab.
3. Click **Key Pairs** and then **Create Key Pair**. The **Create Key Pair** dialog box appears.



4. Enter a name and click **Create**. The name must be unique in your own set of key pairs. Make a note of the key pair. A confirmation message appears as does a dialog box that enables you to open the key pair file.



5. Click **Close** and optionally click **OK** to see the key pair file. The new key pair appears.



How to Create a Job Flow

Topics

- [How to Create a Job Flow Using a Custom JAR \(p. 23\)](#)
- [How to Create a Job Flow Using Streaming \(p. 27\)](#)
- [How to Create a Job Flow Using Pig \(p. 32\)](#)
- [How to Create a Job Flow Using Hive \(p. 36\)](#)
- [How to Use Additional Files and Libraries With the Mapper or Reducer \(p. 40\)](#)

This section explains how to use the AWS Management Console to start a job flow. You can specify the function that processes the data in a variety of ways, including using a JAR, Hive script, Pig script, or Apache's streaming utility. This section follows the creation of a job flow using each of those methods.



Note

We recommend that you restart your job flows periodically (for example, every 2 weeks) and to persist results in Amazon S3 periodically in case the EC2 cluster goes down.

How to Create a Job Flow Using a Custom JAR

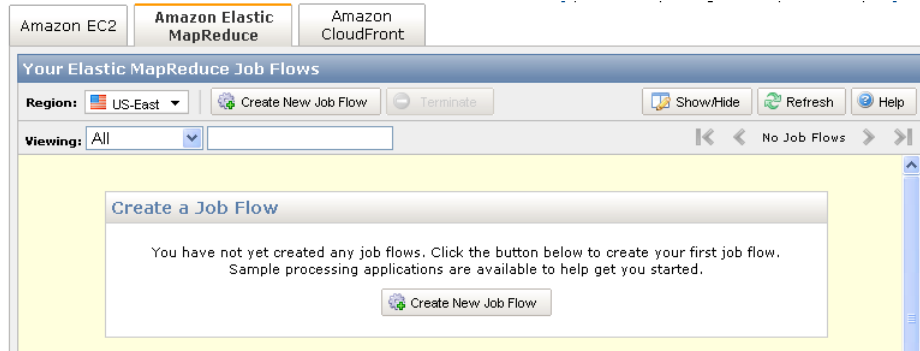
This section explains how to create a job flow using the AWS Management Console. It assumes that you already have a key pair. If you don't, see [How to Create a Key Pair \(p. 21\)](#).

The procedure for creating a JAR job flow is the same using the AWS Management Console.

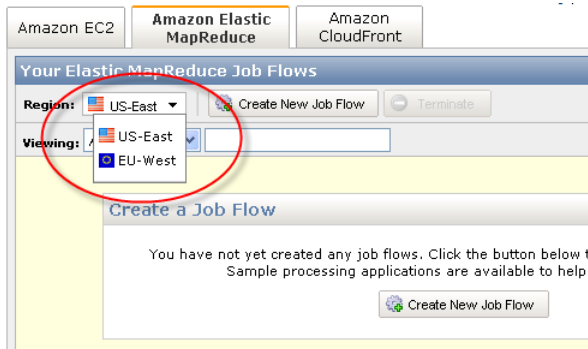
To create a job flow using a custom JAR

1. Go to <https://console.aws.amazon.com/elasticmapreduce/home> and click the **Amazon Elastic MapReduce** tab.

The AWS Management Console appears.



2. Select a **Region**.



For more information, see [Regions \(p. 18\)](#).

3. Click **Create a New Job Flow**.

The **Create a New Job Flow** page appears.

4. In the **Create a New Job Flow** page, enter the following information:

A. Enter a name in the **Job Flow Name** field.

We recommend that you use a descriptive name. It does not need to be unique.

To use the sample mapper, reducer, and data uploaded to Amazon S3, see the [Amazon Elastic MapReduce Getting Started Guide](#).

B. Select **Run your own application**.

C. Select **Custom Jar** in the drop down menu and click **Continue**.

Custom Jar means that you wrote your executables (mapper and reducer) in Java. If you select **Sample Applications**, choose one from the drop down menu.

Amazon Elastic MapReduce Developer Guide

How to Create a Job Flow Using a Custom JAR

The **Specify Parameters** page appears.

Create a New Job Flow Cancel

DEFINE JOB FLOW SPECIFY PARAMETERS CONFIGURE EC2 INSTANCES REVIEW

Specify the location in Amazon S3 of your jar. Hadoop executes the jar. You can specify its main class in its manifest. If you don't you must specify a class name as the first argument of the jar.

Jar Location*:

Jar Arguments*:

Back Continue * Required field

5. Use the following table as a guide to enter values in the boxes on the **Specify Parameters** page and then click **Continue**. The asterisks specify required parameters.

Field	Description
Jar Location*	Specify the URI where your JAR resides in Amazon S3. The value must be in the form: s3://[<i>bucketName</i>]/[<i>path</i>]. This JAR requires a manifest (.mf file) that specifies the main class to use.
Jar Arguments	Enter a list of arguments (space-separated strings) to supply values for variables in your JAR or to load additional files into the distributed cache.

The **Configure EC2 Instances** page appears.

Create a New Job Flow Cancel

DEFINE JOB FLOW SPECIFY PARAMETERS CONFIGURE EC2 INSTANCES REVIEW

Enter the number and type of EC2 instances you'd like to run your job flow on.

Number of Instances*: 4

The number of EC2 instances to run in your Hadoop cluster.
If you wish to run more than 20 instances, please complete the [limit request form](#).

Type of Instance*: Small (m1.small)

The type of EC2 instances to run in your Hadoop cluster ([learn more about instance types](#)).

[Hide Advanced Options](#)

Amazon S3 Log Path:

The log path is a location in Amazon S3 where Elastic MapReduce will upload the log files for each step in the job flow. It will take a few minutes after the step has completed for the logs to appear. If you do not specify a path, the log files will not be uploaded.

Amazon EC2 Key Pair: Select EC2 Key Pair

The Key Pair is the name of an Amazon EC2 Private Key that you have previously created when using Amazon EC2. It is a handle you can use to SSH into the master node of the Amazon EC2 cluster (without a password).

Back Continue * Required field

6. Use the following table as a guide to enter values in the boxes on the **Configure EC2 Instances** page and click **Continue**. The asterisks specify required parameters.

Field	Description
Number of Instances*	Specify the number (1 - 20) of slave nodes to use in the EC2 cluster. Twenty is the maximum number <i>per account</i> ; for example, if you have two job flows running, the total number of instances running for both job flows must be 20 or less. If you need more than 20 instances, you must submit a special request. For more information, go to the Request Form .

Amazon Elastic MapReduce Developer Guide

How to Create a Job Flow Using a Custom JAR

Field	Description
Type of Instance*	Specify the instance size you want to use. Valid values: m1.small (Default), m1.large, m1.xlarge, c1.medium, c1.xlarge. For more information, see Amazon EC2 (p. 18) .
Amazon S3 Log Path	Optionally, specify a path in Amazon S3 to store the log files. The value must be in the form: <code>[bucketName]/[path]</code> . If you do not supply a location, Elastic MapReduce does not log any files.
EC2 Key Pair	Optionally, specify a key-pair that you created previously (for more information, see Creating a Key Pair (p. 21)). This key pair becomes associated with the EC2 instances that will be created to process your job flow. The key pair name you enter in this field creates a handle you can use to access the master node in the EC2 cluster. With this name, you can log in to the master node without using a password. On the master node, you can retrieve detailed job flow processing status and statistics. If you don't enter a value in this field, you cannot SSH into the master node. For more information, see EC2 Key Pairs (p. 19) and Monitoring and Troubleshooting Job Flows (p. 65) .

The **Review** page appears.

Create a New Job Flow

Cancel

DEFINE JOB FLOW

SPECIFY PARAMETERS

CONFIGURE EC2 INSTANCES

REVIEW

Please review the details of your job flow and click "Create Job Flow" when you are ready to launch your Hadoop Cluster.

Job Flow Name:

My Job Flow

Type:

Custom Jar

Edit Job Flow Definition

Jar Location:

s3n://s3amazonaws.com/myBucket/myJAR

Jar Arguments:

-d key = Value

Edit Job Flow Parameters

Number of Instances:

4

Type of Instance:

m1.small

Amazon S3 Log Path:

Amazon EC2 Key Pair:

Edit EC2 Configs and Advanced Options

Back

Create Job Flow

Note: Once you click "Create Job Flow," instances will be launched and you will be charged accordingly.

- Review the information and click **Back** one or more times to revise any of the values, or click **Create Job Flow** if the information is correct.

When you click **Create Job Flow** and it succeeds, a success message appears.

Create a New Job Flow

Cancel

✔ Your Job Flow has been created.

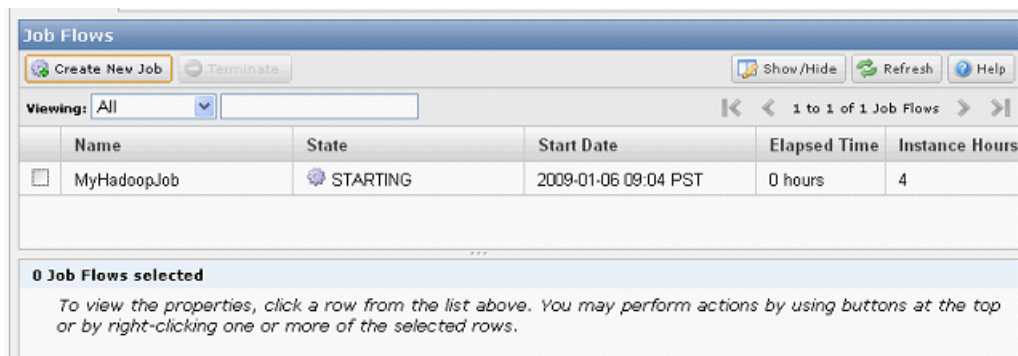
Note: Your job flow may take a few minutes to launch, depending on the type of processing job you are running.
View my job flows and check on job flow status

Close

* Required field

- Click **Close**.

The console reflects the new job flow.



After starting a job flow, you might like to view details about it. For more information, see [How to View the Details of a Job Flow](#) (p. 40).

How to Create a Job Flow Using Streaming

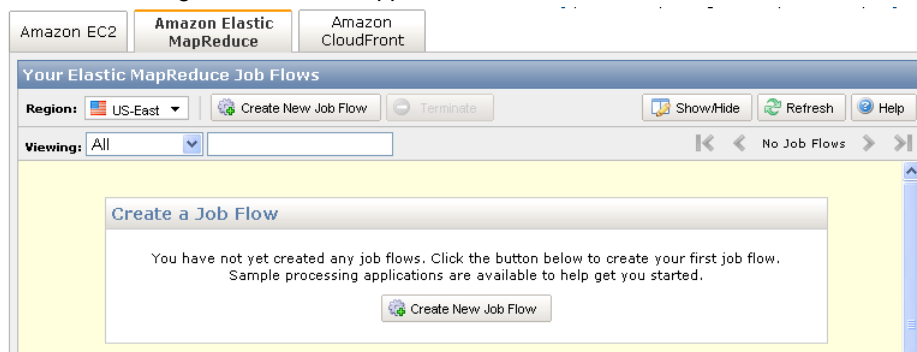
This section explains how to create a job flow using the AWS Management Console. It assumes that you already have a key pair. If you don't, see [How to Create a Key Pair](#) (p. 21).

The procedure for creating a JAR or Streaming job flow is the same using the console. To create a job flow using Pig, see [How to Create a Job Flow Using Pig](#) (p. 32).

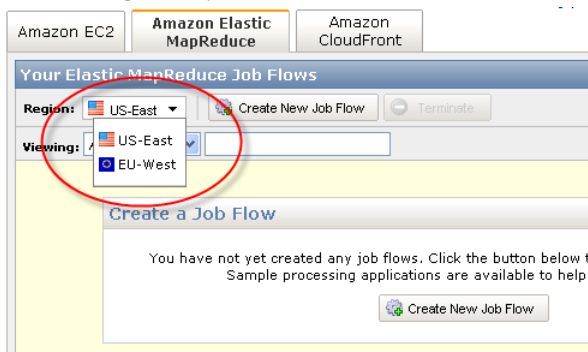
To create a job flow using Streaming or JAR

1. Go to <https://console.aws.amazon.com/elasticmapreduce/home>.

The AWS Management Console appears.



2. Set the **Region** to your locale:



3. Click **Create a New Job Flow**.

The **Create a New Job Flow** page appears.

4. In the **Create a New Job Flow** page, entering the following information:

A. Enter a name in the **Job Flow Name** field.

We recommend that you use a descriptive name. It does not need to be unique.

To use the sample mapper, reducer, and data uploaded to Amazon S3, see the [Amazon Elastic MapReduce Getting Started Guide](#).

B. Select **Run your own application**.

C. Select **Streaming** or **Custom Jar** in the drop down menu and click **Continue**.

Streaming is a Hadoop utility (preloaded on each EC2 instance) that enables you to write mapper and reducer executables in languages other than Java. **Pig Program** is for scripts written in Pig Latin that use the open source Apache Pig library. If you write your executables in Java, select **Custom Jar**. If you select **Sample Applications**, choose one from the drop down menu. For more information, see [Jar Streaming, and Pig \(p. 14\)](#).

When you select **Streaming**, the console automatically uses the streaming JAR located on each instance at `/home/hadoop/contrib/streaming/hadoop-0.18-streaming.jar`.

The **Specify Parameters** page appears.

Amazon Elastic MapReduce Developer Guide

How to Create a Job Flow Using Streaming

Create a New Job Flow

Cancel

DEFINE JOB FLOW

SPECIFY PARAMETERS

CONFIGURE EC2 INSTANCES

REVIEW

Specify Mapper and Reducer functions to run within the Job Flow. The mapper and reducers may be either (i) class names referring to a mapper or reducer class in Hadoop or (ii) locations in Amazon S3. (Click Here for a list of available tools to help you upload and download files from Amazon S3.) The format for specifying a location in Amazon S3 is bucket_name/path_name. The location should point to an executable program, for example a python program. Extra arguments are passed to the Hadoop streaming program and can specify things such as additional files to be loaded into the distributed cache.

Input Location*

elasticmapreduce/samples/wordcount/input

The URL of the Amazon S3 Bucket that contains the input files.

Output Location*

<yourbucket>/wordcount/output/2009-10-26

The URL of the Amazon S3 Bucket to store output files. Should be unique.

Mapper*

elasticmapreduce/samples/wordcount/wordSplitter.py

The mapper Amazon S3 location or streaming command to execute.

Reducer*

aggregate

The reducer Amazon S3 location or streaming command to execute.

Extra Args:

< Back

Continue

* Required field

5. Use the following table as a guide to enter values in the boxes on the **Specify Parameters** page and then click **Continue**. The asterisks specify required parameters.

Field	Description
Input Location*	Specify the URI where the input data resides in Amazon S3. The value must be in the form: s3://[bucketName]/[path].
Output Location*	Specify the URI where you want the output stored in Amazon S3. The value must be in the form: s3://[bucketName]/[path]
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 must be in the form: s3://[bucketName]/[path]
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 must be in the form: s3://[bucketName]/[path]. We support the special "aggregate" keyword. For more information, see the Aggregate library supplied by Hadoop.
Extra Args	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.

The **Configure EC2 Instances** page appears.

Amazon Elastic MapReduce Developer Guide

How to Create a Job Flow Using Streaming

Create a New Job Flow Cancel

DEFINE JOB FLOW SPECIFY PARAMETERS CONFIGURE EC2 INSTANCES REVIEW

Enter the number and type of EC2 instances you'd like to run your job flow on.

Number of Instances*:

The number of EC2 instances to run in your Hadoop cluster.
If you wish to run more than 20 instances, please complete the [limit request form](#).

Type of Instance*:

The type of EC2 instances to run in your Hadoop cluster ([learn more about instance types](#)).

[+ Hide Advanced Options](#)

Amazon S3 Log Path:

The log path is a location in Amazon S3 where Elastic MapReduce will upload the log files for each step in the job flow. It will take a few minutes after the step has completed for the logs to appear. If you do not specify a path, the log files will not be uploaded.

Amazon EC2 Key Pair:

The Key Pair is the name of an Amazon EC2 Private Key that you have previously created when using Amazon EC2. It is a handle you can use to SSH into the master node of the Amazon EC2 cluster (without a password).

[< Back](#) [Continue](#) * Required field

6. Use the following table as a guide to enter values in the boxes on the **Configure EC2 Instances** page and click **Continue**. The asterisks specify required parameters.

Field	Description
Number of Instances*	Specify the number (1 - 20) of slave nodes to use in the EC2 cluster. Twenty is the maximum number <i>per account</i> ; for example, if you have two job flows running, the total number of instances running for both job flows must be 20 or less. If you need more than 20 instances, you must submit a special request. For more information, go to the Request Form .
Type of Instance*	Specify the instance size you want to use. Valid values: m1.small (Default), m1.large, m1.xlarge, c1.medium, c1.xlarge. For more information, see Amazon EC2 (p. 18) .
Amazon S3 Log Path	Optionally, specify a path in Amazon S3 to store the log files. The value must be in the form: <code>[bucketName]/[path]</code> . If you do not supply a location, Elastic MapReduce does not log any files.
EC2 Key Pair	Optionally, specify a key-pair that you created previously (for more information, see Creating a Key Pair (p. 21)). This key pair becomes associated with the EC2 instances that will be created to process your job flow. The key pair name you enter in this field creates a handle you can use to access the master node in the EC2 cluster. With this name, you can log in to the master node without using a password. On the master node, you can retrieve detailed job flow processing status and statistics. If you don't enter a value in this field, you cannot SSH into the master node. For more information, see EC2 Key Pairs (p. 19) and Monitoring and Troubleshooting Job Flows (p. 65) .

The **Review** page appears.

Amazon Elastic MapReduce Developer Guide

How to Create a Job Flow Using Streaming

Your Elastic MapReduce Job Flows

Region: US-East Create New Job Flow Terminate Show/Hide Refresh Help

Viewing: All 1 to 14 of 14 Job Flows

Name	State	Creation Date	Elapsed Time	Normalized Instance Hours
My Job Flow	STARTING	2009-10-26 15:05 PST	0 hours 0 minutes	0

1 Job Flow selected

Id: j-4DJGF0855LIS **Creation Date:** 2009-10-26 15:05 PST

Name: My Job Flow **Start Date:** -

State: STARTING **End Date:** -

Last State Change Reason: Starting instances

Availability Zone: us-east-1c **Instance Count:** 1

Master Type: m1.small **Slave Type:** m1.small

Key Name: - **Log URI:** -

Master Public DNS Name: ec2-75-101-236-239.compute-1.amazonaws.com

Steps:

Step Name	State	Start Date	End Date	Jar	Main Class	Args
Streaming Job	PENDING	-	-	/home/hadoop/contrib/streaming/hadoop-0.18-streaming.jar		-input s3n://elasticmapreduce/samples/wordcount /input -output s3n://anhi-test-data/wordcount /output/2009-10-26 -mapper s3n://elasticmapreduce/samples/wordcount /wordSplitter.py -reducer aggregate

7. Review the information and click **Back** one or more times to revise any of the values, or click **Create Job Flow** if the information is correct.

When you click **Create Job Flow** and it succeeds, a success message appears.

Create a New Job Flow Cancel X

☒ Your Job Flow has been created.

Note: Your job flow may take a few minutes to launch, depending on the type of processing job you are running.

[View my job flows and check on job flow status](#)

Close

* Required field

8. Click **Close**.

The console reflects the new job flow. The maximum lifetime of a job flow is 2 weeks. Elastic MapReduce returns an error if processing doesn't complete within two weeks.

Job Flows

Create New Job Terminate Show/Hide Refresh Help

Viewing: All 1 to 1 of 1 Job Flows

Name	State	Start Date	Elapsed Time	Instance Hours
MyHadoopJob	STARTING	2009-01-06 09:04 PST	0 hours	4

0 Job Flows selected

To view the properties, click a row from the list above. You may perform actions by using buttons at the top or by right-clicking one or more of the selected rows.

After starting a job flow, you might like to view details about it. For more information, see [How to View the Details of a Job Flow](#) (p. 40).

How to Create a Job Flow Using Pig

This section explains how to create a job flow using the AWS Management Console and Pig using batch mode.



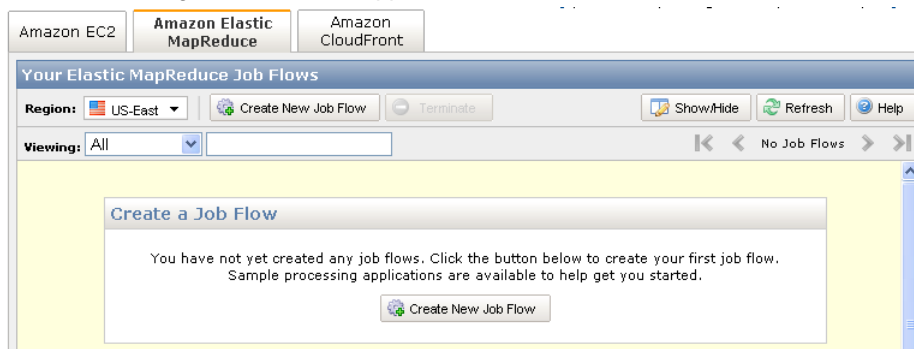
Note

For more information about running an interactive Pig session, go to the [Pig video tutorial](#).

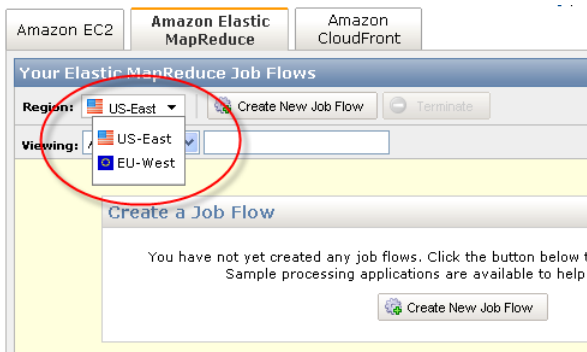
To create a job flow using Pig

1. Go to <https://console.aws.amazon.com/elasticmapreduce/home>.

The AWS Management Console appears.



2. Set the **Region** to your locale:



3. Click **Create a New Job Flow**.

The **Create a New Job Flow/ Define Job Flow** page appears.

Amazon Elastic MapReduce Developer Guide

How to Create a Job Flow Using Pig

Create a New Job Flow Cancel

DEFINE JOB FLOW | SPECIFY PARAMETERS | CONFIGURE EC2 INSTANCES | REVIEW

Creating a job flow to process your data using Amazon Elastic MapReduce is simple and quick. Let's begin by giving your job flow a name and selecting its type. If you don't already have an application you'd like to run on Amazon Elastic MapReduce, samples are available to help you get started.

Job Flow Name*:

Job Flow Name doesn't need to be unique. We suggest you give it a descriptive name.

Create a Job*: ☒ Run your own application
☐ Run a sample application

▼

- Choose a Job Type
- Streaming
- Custom Jar
- Pig Program**
- Hive Program

Pig is a SQL-like language built on top of Hadoop. This option allows you to define a job flow that runs a Pig script, or set up a job flow that can be used interactively via SSH to run Pig commands. [Learn More](#)

Continue * Required field

4. In the **Create a New Job Flow** page, define the job flow.

A. Enter a name in the **Job Flow Name** field.

We recommend that you use a descriptive name. It does not need to be unique.

B. Select **Run your own application**.

C. Select **Pig Program** in the drop down menu and click **Continue**.

Pig Program is for scripts written in Pig Latin that use the open source Apache Pig library. For more information, see [Jar Streaming, Pig, and Hive](#) (p. 14).

The **Specify Parameters** pane appears.

Create a New Job Flow Cancel

DEFINE JOB FLOW | **SPECIFY PARAMETERS** | CONFIGURE EC2 INSTANCES | REVIEW

Choose between either executing an existing Pig script or starting an interactive Pig session.

☒ **Execute a Pig Script**
Run a Pig script which has been uploaded to S3. With this option the job flow starts, automatically executes the script, then terminates the job flow automatically when the script has completed.

Script Location*:

The location of your Pig script in Amazon S3.

Input Location:

The URL of the Amazon S3 Bucket that contains the input files.

Output Location:

The URL of the Amazon S3 Bucket to store output files. Should be unique.

Extra Args:

☐ **Start an Interactive Pig Session**
Start a job flow with Pig setup for interactive use. Interactive use requires you to have an SSH client to access the master host. When you are finished your session, manually terminate the job flow from the list of running jobs.

< Back Continue * Required field

5. Click the **Execute a Pig Script** radio button, use the following table as a guide to enter values in the boxes on the **Specify Parameters** pane, and then click **Continue**. The asterisks specify required parameters.

Amazon Elastic MapReduce Developer Guide

How to Create a Job Flow Using Pig

Field	Description
Script Location*	Specify the URI where the Pig Latin script resides in Amazon S3. The value must be in the form: <code>[bucketName]/[path]</code> . If you're using the HDFS file system for the input data, use three slashes (<code>///</code>) to designate the path, for example, <code>hdfs:///aws-hadoop/CompanyName/SampleInput/</code> .
Input Location	Specify the URI where the input data resides in Amazon S3. The value must be in the form: <code>[bucketName]/[path]</code> . If you're using the HDFS file system for the input data, use three slashes (<code>///</code>) to designate the path, for example, <code>hdfs:///aws-hadoop/CompanyName/SampleInput/</code> .
Output Location	Specify the URI where you want the output stored in Amazon S3. The value must be in the form: <code>[bucketName]/[path]</code>
Extra Args	<p>Optionally, enter a list of arguments (space-separated strings) to pass to the Pig program. For example, you can specify additional files to load into the distributed cache. Elastic MapReduce supports the standard set of Pig arguments. For more information, see http://wiki.apache.org/pig/.</p> <p>In addition, AWS added an additional argument, <code>-d</code>, so that you can pass parameter values into the Hive script. The format is:</p> <pre>-d VariableName = Value</pre> <p>For example:</p> <pre>-d SomeLibrary = s3n://bucketName/LibraryName</pre> <p>In your script, you reference the variable using a dollar sign and curly braces, for example, <code>\${VariableName}</code></p>

The **Configure EC2 Instances** page appears.

Create a New Job Flow
Cancel X

▼
▼
○
▼

DEFINE JOB FLOW
SPECIFY PARAMETERS
CONFIGURE EC2 INSTANCES
REVIEW

Enter the number and type of EC2 instances you'd like to run your job flow on.

Number of Instances*:

The number of EC2 instances to run in your Hadoop cluster.
If you wish to run more than 20 instances, please complete the [limit request form](#).

Type of Instance*: Small (m1.small) ▼

The type of EC2 instances to run in your Hadoop cluster ([learn more about instance types](#)).

+ Hide Advanced Options

Amazon S3 Log Path:

The log path is a location in Amazon S3 where Elastic MapReduce will upload the log files for each step in the job flow. It will take a few minutes after the step has completed for the logs to appear. If you do not specify a path, the log files will not be uploaded.

Amazon EC2 Key Pair: - Select EC2 Key Pair - ▼

The Key Pair is the name of an Amazon EC2 Private Key that you have previously created when using Amazon EC2. It is a handle you can use to SSH into the master node of the Amazon EC2 cluster (without a password).

< Back
Continue >
* Required field

- Use the following table as a guide to enter values in the boxes on the **Configure EC2 Instances** page and click **Continue**. The asterisks specify required parameters.

Amazon Elastic MapReduce Developer Guide

How to Create a Job Flow Using Pig

Field	Description
Number of Instances*	Specify the number (1 - 20) of slave nodes to use in the EC2 cluster. Twenty is the maximum number <i>per account</i> ; for example, if you have two job flows running, the total number of instances running for both job flows must be 20 or less. If you need more than 20 instances, you must submit a special request. For more information, go to the Request Form .
Type of Instance*	Specify the instance size you want to use. Valid values: m1.small (Default), m1.large, m1.xlarge, c1.medium, c1.xlarge. For more information, see Amazon EC2 (p. 18) .
Amazon S3 Log Path	Optionally, specify a path in Amazon S3 to store the log files. The value must be in the form: <code>[bucketName]/[path]</code> . If you do not supply a location, Elastic MapReduce does not log any files.
EC2 Key Pair	This parameter is optional in batch mode but required in interactive mode. Use the parameter to specify a key-pair that you created previously (for more information, see Creating a Key Pair (p. 21)). This key pair becomes associated with the EC2 instances that will be created to process your job flow. The key pair name you enter in this field creates a handle you can use to access the master node in the EC2 cluster. With this name, you can log in to the master node without using a password. On the master node, you can retrieve detailed job flow processing status and statistics. If you don't enter a value in this field, you cannot SSH into the master node. For more information, see EC2 Key Pairs (p. 19) and Monitoring and Troubleshooting Job Flows (p. 65) .

The **Review** page appears.

Create a New Job Flow Cancel X

✓

✓

✓

○

DEFINE JOB FLOW SPECIFY PARAMETERS CONFIGURE EC2 INSTANCES REVIEW

Please review the details of your job flow and click "Create Job Flow" when you are ready to launch your Hadoop Cluster.

Job Flow Name:

My Job Flow

Type:

Pig Script

Edit Job Flow Definition

Pig Script Location:

s3n://MyBucket/MyPigScript

Input Location:

Output Location:

Extra Args:

Edit Job Flow Parameters

Number of Instances:

4

Type of Instance:

m1.small

Amazon S3 Log Path:

Amazon EC2 Key Pair:

Edit EC2 Configs and Advanced Options

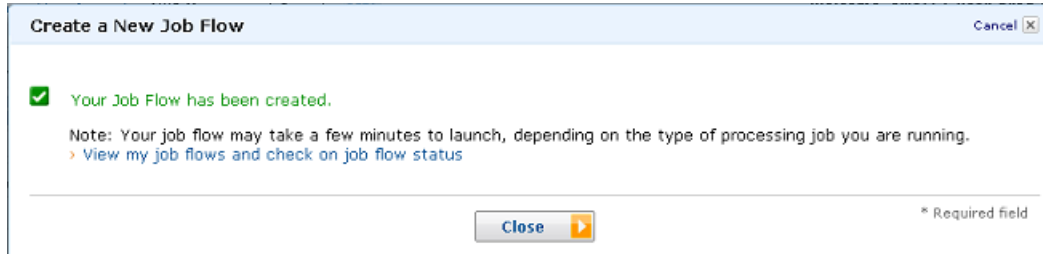
< Back

Create Job Flow

Note: Once you click "Create Job Flow," instances will be launched and you will be charged accordingly.

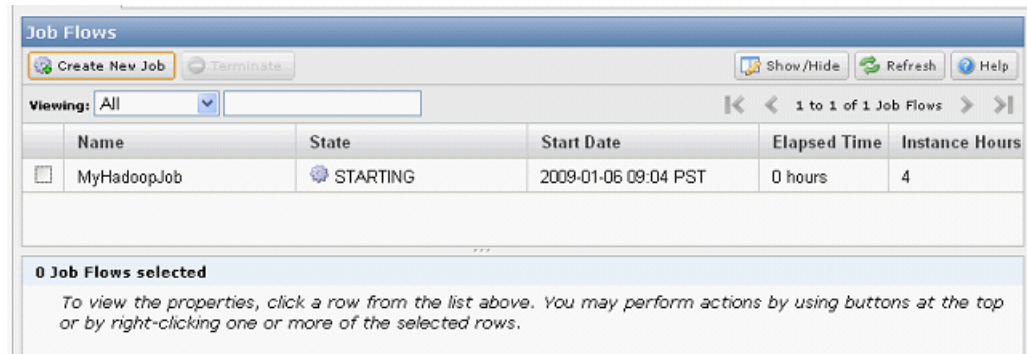
- Review the information and click **Back** one or more times to revise any of the values, or click **Create Job Flow** if the information is correct.

When you click **Create Job Flow** and it succeeds, a success message appears.



8. Click **Close**.

The console reflects the new job flow. The maximum lifetime of a job flow is 2 weeks. Elastic MapReduce returns an error if processing doesn't complete within two weeks.



After starting a job flow, you might like to view details about it. For more information, see [How to View the Details of a Job Flow](#) (p. 40).

How to Create a Job Flow Using Hive

This section explains how to create a job flow using the AWS Management Console, Hive, and batch mode.

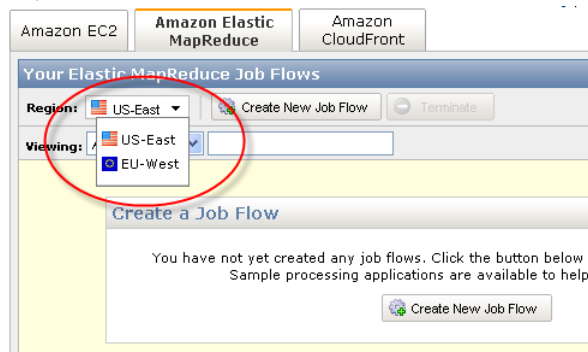


Note

For more information about running an interactive Hive session, go to the [Hive video tutorial](#).

To create a job flow using Hive

1. On the AWS Management Console, click the **Amazon Elastic MapReduce** tab and set the **Region** to your locale.



2. Click **Create a New Job Flow**.

Amazon Elastic MapReduce Developer Guide

How to Create a Job Flow Using Hive

The **Create a New Job Flow/ Define Job Flow** page appears.

Create a New Job Flow Cancel

DEFINE JOB FLOW SPECIFY PARAMETERS CONFIGURE EC2 INSTANCES REVIEW

Creating a job flow to process your data using Amazon Elastic MapReduce is simple and quick. Let's begin by giving your job flow a name and selecting its type. If you don't already have an application you'd like to run on Amazon Elastic MapReduce, samples are available to help you get started.

Job Flow Name*:

Job Flow Name doesn't need to be unique. We suggest you give it a descriptive name.

Create a Job*: ☒ Run your own application ☐ Run a sample application

Choose a Job Type

- Hive Program
- Streaming
- Custom Jar
- Pig Program
- Hive Program

Hive is an SQL-like language built on top of Hadoop. This option allows you to define a job flow that runs a Hive script, or set up a job flow that can be used interactively via SSH to run Hive commands. [Learn More](#)

Continue * Required field

3. In the **Create a New Job Flow** page, define the job flow.

A. Enter a name in the **Job Flow Name** field.

We recommend that you use a descriptive name. It does not need to be unique.

B. Select **Run your own application**.

C. Select **Hive Program** in the drop down menu.

Hive Program is for scripts written in Hive QL that use the open source Hive library.

4. Click **Continue**.

The **Specify Parameters** pane appears.

Create a New Job Flow Cancel

DEFINE JOB FLOW SPECIFY PARAMETERS CONFIGURE EC2 INSTANCES REVIEW

☒ **Execute a Hive Script**
Run a hive script which has been uploaded to S3. With this option the job flow starts, automatically executes the script, and when the script completes the job flow automatically terminates.

Script Location*:
Specify the location in Amazon S3 of your Hive script.

Input Location*:
The URL of the Amazon S3 Bucket that contains the input files.

Output Location*:
The URL of the Amazon S3 Bucket to store output files. Should be unique.

Extra Args:

☐ **Start an Interactive Hive Session**
Start a job flow with Hive setup for interactive use. Interactive use requires you to have a SSH client to be able to access the master host. It also requires that when you are finished your session, you need to manually terminate the job flow from the main grid.

Back Continue * Required field

5. Select the **Execute a Hive Script** radio button, use the following table as a guide to enter values in the boxes on the **Specify Parameters** pane, and click **Continue**. The asterisks specify required parameters.

Amazon Elastic MapReduce Developer Guide

How to Create a Job Flow Using Hive

Field	Description
Script Location*	Specify the URI where the Hive QL script resides in Amazon S3. The value must be in the form: s3://[<i>bucketName</i>]/[<i>Script</i>].
Input Location	Specify the URI where the input data resides in Amazon S3. The value must be in the form: s3://[<i>bucketName</i>]/[<i>Input</i>]. The value you supply is passed to the Hive script as <i>INPUT</i> .
Output Location	Specify the URI where you want the output stored in Amazon S3. The value must be in the form: s3://[<i>bucketName</i>]/[<i>Output</i>]. The value you supply is passed to the Hive script as <i>OUTPUT</i> .
Extra Args	<p>Optionally, enter a list of arguments (space-separated strings) to pass to the Hive program. For example, you can specify additional files to load into the distributed cache. . Elastic MapReduce supports the standard set of Hive arguments. For more information, go to http://wiki.apache.org/hadoop/Hive. In addition, AWS added an additional argument, <code>-d</code>, so that you can pass parameter values into the Hive script. The format is:</p> <pre>-d VariableName = Value</pre> <p>For example:</p> <pre>-d SomeLibrary = s3n://bucketName/LibraryName</pre> <p>In your script, you reference the variable using a dollar sign and curly braces, for example, <code>\${VariableName}</code></p>

The **Configure EC2 Instances** page appears.

Create a New Job Flow
Cancel

DEFINE JOB FLOW
SPECIFY PARAMETERS

CONFIGURE EC2 INSTANCES
REVIEW

Enter the number and type of EC2 instances you'd like to run your job flow on.

Number of Instances*:

The number of EC2 instances to run in your Hadoop cluster. If you wish to run more than 20 instances, please complete the [limit request form](#).

Type of Instance*: Small (m1.small) ▼

The type of EC2 instances to run in your Hadoop cluster ([learn more about instance types](#)).

[Hide Advanced Options](#)

Amazon S3 Log Path:

The log path is a location in Amazon S3 where Elastic MapReduce will upload the log files for each step in the job flow. It will take a few minutes after the step has completed for the logs to appear. If you do not specify a path, the log files will not be uploaded.

Amazon EC2 Key Pair: - Select EC2 Key Pair - ▼

The Key Pair is the name of an Amazon EC2 Private Key that you have previously created when using Amazon EC2. It is a handle you can use to SSH into the master node of the Amazon EC2 cluster (without a password).

< Back
Continue
* Required field

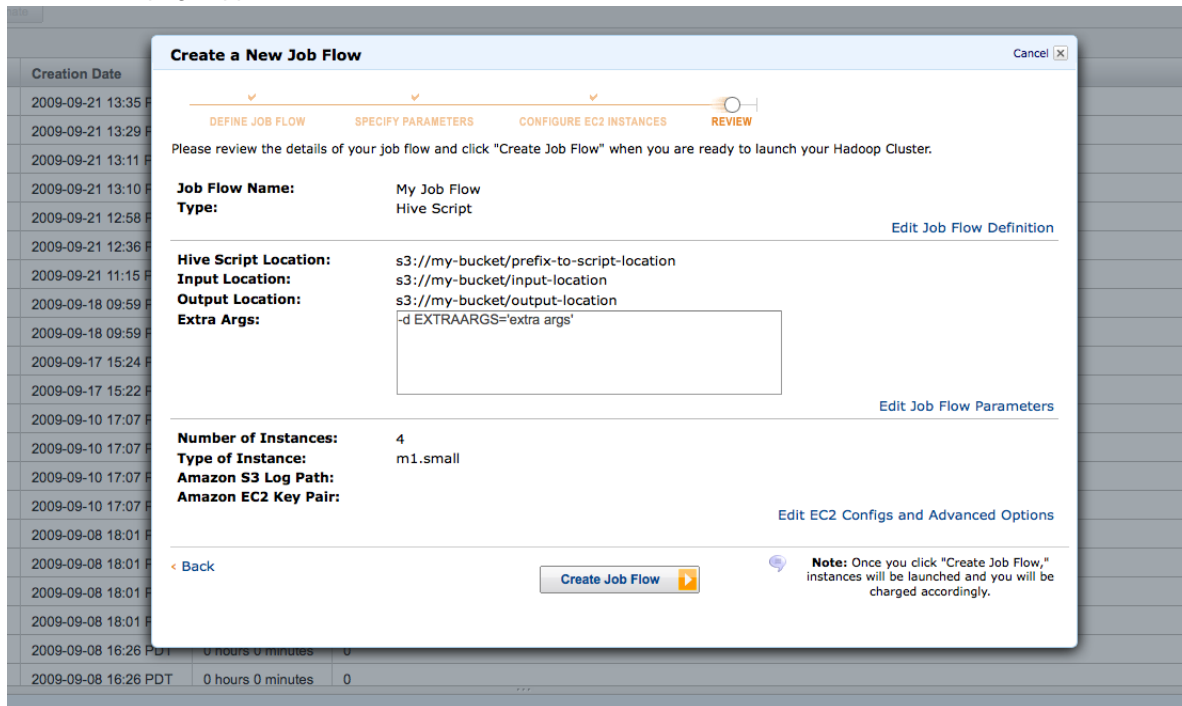
6. Use the following table as a guide to enter values in the boxes on the **Configure EC2 Instances** page and click **Continue**. The asterisks specify required parameters.

Amazon Elastic MapReduce Developer Guide

How to Create a Job Flow Using Hive

Field	Description
Number of Instances*	Specify the number (1 - 20) of slave nodes to use in the EC2 cluster. 20 is the maximum number <i>per account</i> ; for example, if you have two job flows running, the total number of instances running for both job flows must be 20 or less. If you need more than 20 instances, you must submit a special request. For more information, go to the Request Form .
Type of Instance*	Specify the instance size you want to use. Valid values: m1.small (Default), m1.large, m1.xlarge, c1.medium, c1.xlarge. For more information, see Amazon EC2 (p. 18) .
Amazon S3 Log Path	Optionally, specify a path in Amazon S3 to store the log files. The value must be in the form: s3://[<i>bucketName</i>]/[<i>path</i>]. If you do not supply a location, Elastic MapReduce does not log any files.
EC2 Key Pair	This parameter is optional in batch mode but required in interactive mode. Use the parameter to specify a key-pair that you created previously (for more information, see Creating a Key Pair (p. 21)). This key pair becomes associated with the EC2 instances that will be created to process your job flow. The key pair name you enter in this field creates a handle you can use to access the master node in the EC2 cluster. With this name, you can log in to the master node without using a password. On the master node, you can retrieve detailed job flow processing status and statistics. If you don't enter a value in this field, you cannot SSH into the master node. For more information, see EC2 Key Pairs (p. 19) and Monitoring and Troubleshooting Job Flows (p. 65) .

The **Review** page appears.



Create a New Job Flow [Cancel]

DEFINE JOB FLOW SPECIFY PARAMETERS CONFIGURE EC2 INSTANCES **REVIEW**

Please review the details of your job flow and click "Create Job Flow" when you are ready to launch your Hadoop Cluster.

Job Flow Name: My Job Flow
Type: Hive Script [Edit Job Flow Definition](#)

Hive Script Location: s3://my-bucket/prefix-to-script-location
Input Location: s3://my-bucket/input-location
Output Location: s3://my-bucket/output-location
Extra Args: -d EXTRAARGS="extra args" [Edit Job Flow Parameters](#)

Number of Instances: 4
Type of Instance: m1.small
Amazon S3 Log Path:
Amazon EC2 Key Pair: [Edit EC2 Configs and Advanced Options](#)

[Back](#) [Create Job Flow](#)

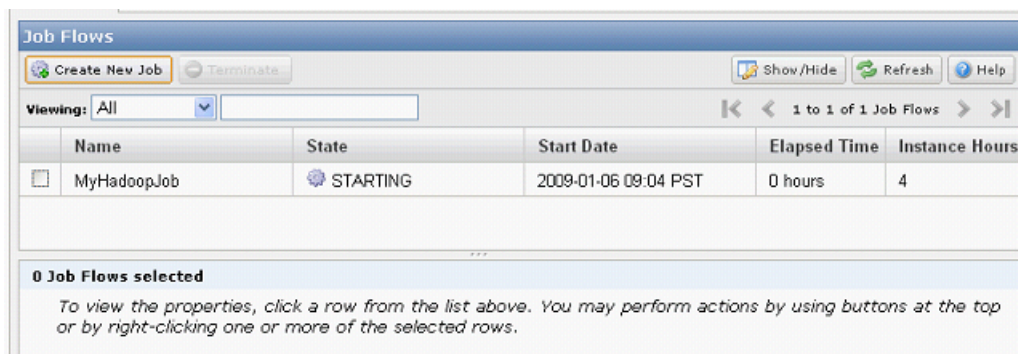
Note: Once you click "Create Job Flow," instances will be launched and you will be charged accordingly.

- Review the information and click **Back** one or more times to revise any of the values, or click **Create Job Flow** if the information is correct.

When you click **Create Job Flow** and it succeeds, a success message appears.

8. Click **Close**.

The console reflects the new job flow. The maximum lifetime of a job flow is two weeks. Elastic MapReduce returns an error if processing doesn't complete within that time.



After starting a job flow, you might like to view details about it. For more information, see [How to View the Details of a Job Flow](#) (p. 40).

How to Use Additional Files and Libraries With the Mapper or Reducer

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 into text.

To cache a file (or library), you run a streaming job flow and put the file in the working directory of the mapper and reducer. The file can be either a statically compiled binary or a script that your mapper or reducer application calls. For more information about caching a file or library, see [Distributed Cache](#) (p. 82).

To add a file or library for the mapper to use

- In the Extra args option, enter a command similar to the following:

```
-cacheFile s3://bucket/path_to_executable#local_path
```

The file, *local_path*, would be in the working directory of the mapper, which could reference the file.

How to View the Details of a Job Flow

After you start a job flow, you can monitor its status and get extended information about its execution. This section explains how to view the details of a job flow using the AWS Management Console.

To view the details of a job flow

1. Go to <https://console.aws.amazon.com/elasticmapreduce/home>.
The **Your Elastic MapReduce Job Flows** page appears.
2. Select the check box next to the job flow of interest.
The **Job Flow selected** pane appears providing detailed information about the selected job flow.

The screenshot shows the 'Your Elastic MapReduce Job Flows' page. At the top, there are buttons for 'Create New Job Flow' and 'Terminate'. Below these is a table of job flows. The second job flow, 'PS Test 1/30/2009 S_7', is selected. Below the table, the details for the selected job flow are shown, including its ID, name, state, creation date, start date, end date, last state change reason, availability zone, master type, key name, instance count, slave type, log URI, master public DNS name, and a list of steps.

Name	State	Start Date	Elapsed Time	Normalized Instance Hours
ps test 2/2/09	COMPLETED	2009-02-02 13:12 PST	0 hours 23 minutes	5
PS Test 1/30/2009 S_7	COMPLETED	2009-01-30 16:06 PST	1 hour 7 minutes	4

1 Job Flow selected

Id: j-83KFC1798FGK
Name: PS Test 1/30/2009 S_7
State: COMPLETED
Last State Change Reason: A step completed
Availability Zone: us-east-1c
Master Type: m1.small
Key Name: -
Creation Date: 2009-01-30 16:06 PST
Start Date: 2009-01-30 16:06 PST
End Date: 2009-01-30 17:12 PST
Instance Count: 2
Slave Type: m1.small
Log URI: s3n://aws157-user-prod/logs/pstest_2009_01_30_s_7/
Master Public DNS Name: ec2-174-129-170-32.compute-1.amazonaws.com

Steps:

Step Name	State	Start Date	End Date	Jar	Main Class	Args
Custom Jar	COMPLETED	2009-01-30 16:12 PST	2009-01-30 17:10 PST	s3n://aws157-user-prod/netflix/finch.jar		s3n://aws157-user-prod/netflix/input/s3n://aws157-user-prod/output/pstest_2009_01_30_s_7

How to Terminate a Job Flow

You can terminate a job flow while it is running. This section explains how to terminate a job flow using the AWS Management Console.

To terminate a job flow

1. Go to <https://console.aws.amazon.com/elasticmapreduce/home>. The **Your Elastic MapReduce Job Flows** page appears.
2. Select the check box next to the job flow you want to terminate. Elastic MapReduce displays the job flow details of the selected job flow.

The screenshot shows the 'Your Elastic MapReduce Job Flows' page. At the top, there are buttons for 'Create New Job Flow' and 'Terminate'. Below these is a table of job flows. The second job flow, 'PS Test 1/30/2009 S_7', is selected. Below the table, the details for the selected job flow are shown, including its ID, name, state, creation date, start date, end date, last state change reason, availability zone, master type, key name, instance count, slave type, log URI, master public DNS name, and a list of steps.

Name	State	Start Date	Elapsed Time	Normalized Instance Hours
ps test 2/2/09	COMPLETED	2009-02-02 13:12 PST	0 hours 23 minutes	5
PS Test 1/30/2009 S_7	COMPLETED	2009-01-30 16:06 PST	1 hour 7 minutes	4

1 Job Flow selected

Id: j-83KFC1798FGK
Name: PS Test 1/30/2009 S_7
State: COMPLETED
Last State Change Reason: A step completed
Availability Zone: us-east-1c
Master Type: m1.small
Key Name: -
Creation Date: 2009-01-30 16:06 PST
Start Date: 2009-01-30 16:06 PST
End Date: 2009-01-30 17:12 PST
Instance Count: 2
Slave Type: m1.small
Log URI: s3n://aws157-user-prod/logs/pstest_2009_01_30_s_7/
Master Public DNS Name: ec2-174-129-170-32.compute-1.amazonaws.com

Steps:

Step Name	State	Start Date	End Date	Jar	Main Class	Args
Custom Jar	COMPLETED	2009-01-30 16:12 PST	2009-01-30 17:10 PST	s3n://aws157-user-prod/netflix/job.jar		s3n://aws157-user-prod/netflix/input/s3n://aws157-user-prod/output/pstest_2009_01_30_s_7

3. Click **Terminate**.
The **Terminate Job Flow(s)** confirmation dialog box appears.
4. Click **Yes, Terminate**.
Elastic MapReduce releases the instances in the cluster and stops uploading data to Amazon S3.

Additional Console Tutorials

For additional console tutorials, go to [Introduction to Amazon Elastic MapReduce](#), [Finding Similar Items with Amazon Elastic MapReduce, Python, and Hadoop Streaming](#), and [Using a Pig Script with the Console Video Tutorial](#).

Using the Command Line Interface

Topics

- [How to Download and Install Ruby and the Command Line Interface \(p. 43\)](#)
- [How to List All Command Options \(p. 46\)](#)
- [How to Create a Job Flow \(p. 47\)](#)
- [How to Use Additional Files and Libraries With the Mapper or Reducer \(p. 51\)](#)
- [How to List Job Flows \(p. 52\)](#)
- [How to Get Information About a Job Flow \(p. 53\)](#)
- [How to Add Steps to a Job Flow \(p. 53\)](#)
- [How to Terminate a Job Flow \(p. 54\)](#)
- [Additional CLI Tutorials \(p. 54\)](#)

The Elastic MapReduce command line interface (CLI) provides a command line interface that makes it easy to use all of the functionality offered by the Elastic MapReduce API. This section discusses the third-party software required to use the CLI, how to download and install the CLI, and how to use it.

For information about downloading the Elastic MapReduce command line interface or Ruby, go to the *Amazon Elastic MapReduce Getting Started Guide*.

The Elastic MapReduce command line interface (CLI) provides all of the functionality of the Elastic MapReduce API and handles many tasks that have to be repeated, such as calculating the signature and including it in your requests.

The following sections explain how to use the CLI.

How to Download and Install Ruby and the Command Line Interface

This section discusses the third-party software required to use the CLI, how to download and install the CLI. For information about using the CLI, go to the [Command Line Interface section](#) in the *Amazon Elastic MapReduce Developer Guide*.

System Dependencies

To use the Elastic MapReduce CLI, you must install the following software:

- Ruby version 1.8 or later
To get Ruby, go to <http://rubyforge.org/>.
- Ruby json library version 0.4.2-1 or later
To get the ruby json library, go to <http://rubyforge.org/>.

How to Download, Install, and Configure the Elastic MapReduce CLI

This section explains how to download, install, and configure the Elastic MapReduce CLI.

You must have Ruby installed to use the Elastic MapReduce CLI.

Downloading and Installing Ruby

1. Use the appropriate download command for your operating system.

If you have this operating system	Use this command
Ubuntu and Debian	apt-get install ruby1.8
Redhat Linux	yum install ruby
Apple Macintosh	(Already installed)
Microsoft Windows	One-click windows installer for Ruby available from http://rubyforge.org/projects/rubyinstaller/

2. Verify that Ruby is running by typing the following at the command prompt:

```
$ ruby -v
```

If the version displays, you installed ruby correctly. If the version does not display, repeat the installation process.

To download, install and configure the Elastic MapReduce CLI

1. Download the Elastic MapReduce CLI, go to <http://developer.amazonwebservices.com/connect/entry.jspa?externalID=2264&categoryID=273> and click **Download**.
2. Unzip the compressed file.

```
$ unzip elastic-mapreduce-ruby.zip
```

The README file has more detailed instructions on how to run on how to install Ruby. It also contains specific instructions for running under Windows

3. Navigate to the `elastic-mapreduce-ruby-client` directory.
4. Optionally, create an Amazon EC2 keypair.
If you don't have an EC2 keypair, you can create one using the [EC2 tab of the AWS Management Console](#).

Specifying the keypair is optional. It enables you to log into the master node without using a password so that you can monitor the progress of your job flows. We strongly recommend that you use key pairs.

5. On a Linux/Unix system, set the permissions on the PEM file, which you download with the keypair. For example, if you saved the file as `mykey.pem`, run:>

```
chmod og-rwx mykey.pem
```

6. Create or edit `credentials.json` in a text editor, as follows.

With this operating system...	Do this...
Linux	<p>Edit <code>credentials.json</code> and replace the following:</p> <ul style="list-style-type: none"> • <code>access_id</code> with your real AWS Access Key ID • <code>private_key</code> with your Secret Access Key • <code>keypair</code> with the name of the Key Pair you would like to use to log into the EC2 master node If you don't have an EC2 keypair then you can create one using the EC2 tab of the AWS Management Console. • <code>log_uri</code> with the path to a bucket you own in Amazon S3, for example, <code>s3://mybucket/logs</code> • <code>region</code> with your region, if other than the default (us-east-1). From the EU locale, use eu-west-1.
Microsoft Windows	<p>Create a file named <code>credentials.json</code> using a text editor and enter the following:</p> <pre>{ "access_id": "[Your AWS Access Key ID]", "private_key": "[Your AWS Secret Access Key]", "keypair": "[KeyPair name created previously]" "log_uri": "[Path to a bucket you own in Amazon S3, for example, s3://mybucket/logs]" "region": "[The region where you want to launch your job flow, either us-east-1 or eu-west-1]" }</pre>

Your credentials are used to calculate the signature value for every request you make. Elastic MapReduce automatically looks for them in `credentials.json`. For that reason, it's most convenient to edit the `credentials.json` file. You can enter the name of another file on the command line where your credentials reside but you must do it for each request.

```
$ ./elastic-mapreduce -c AnotherCredentialsFile.json --list
```

Windows users would enter:

```
$ ruby elastic-mapreduce -c AnotherCredentialsFile.json --list
```

Credentials File

The credentials file provides information required for many commands. The file saves you from the trouble of entering the information repeatedly. You need to create or update your credentials file with the following fields:

```
{
```

```
"access-id": "181sAmPlE18ABA",  
"private-key": "ABA/A1SaMpLejla/AS1a",  
"key-pair": "my-key",  
"key-pair-file": "/home/user/keys/mykey.pem",  
"region": "us-east-1",  
"log-uri": "s3://mybucket/emr-logs"  
}
```

You must replace the values above with your own access key ID, private key, EC2 KeyPair, and so forth. You must create an EC2 KeyPair if you don't already have one. You can do that using the AWS Management Console (<http://console.aws.amazon.com>). For more information, go to [How to Create a KeyPair](#) (p. 21). The value for log-uri must be a bucket that you create. For more information about creating an Amazon S3 bucket, go to [Amazon S3 Bucket Creation](#) in the *Amazon Elastic MapReduce Getting Started Guide*.

How to List All Command Options

You can use the help option to list all of the commands available in the Elastic MapReduce CLI.

To list all of the command options

- Type the following at the command prompt:

```
$ ./elastic-mapreduce --help
```

For Microsoft Windows, type the following:

```
$ ruby elastic-mapreduce --help
```

The output appears as follows:

```
Usage: elastic-mapreduce [options]  
--create                Create a new job flow  
--name NAME             Name of the job flow  
--alive                Create a job flow that stays running  
                       even though it has executed all its steps  
--JAR JAR              Add a step that executes a jar  
--main-class MAIN_CLASS Specify main class for the JAR  
--arg ARG              Specify an argument to a jar step  
                       or a streaming step  
--jobconf JOB_CONF     Specify jobconf arguments to pass to streaming  
--stream              Add a step that performs Hadoop streaming  
--step_name STEP_NAME  Add a step to the work flow  
--input INPUT          Input to the steps, e.g. s3n://mybucket/input  
--output OUTPUT        The output to the steps,  
                       e.g. s3n://mybucket/output  
--mapper MAPPER        The mapper program or class  
--cache CACHE_FILE     A file to load into the cache,  
                       e.g. s3n://mybucket/sample.py#sample.py  
--reduce REDUCER       The reducer program or class  
--list, --describe     List all job flows created in the last 2 days  
--active              List running, starting or shutting down  
                       job flows  
--state STATE          List job flows in STATE  
--all                 List all job flows in the last 2 months  
--nosteps             Do not list steps when listing jobs  
-n, --max-results MAX_RESULTS  Maximum number of results to list
```

<code>--terminate</code>	Terminate the job flow
<code>--num-instances NUM_INSTANCES</code>	Number of instances in the job flow
<code>--key_pair KEYPAIR</code>	The type of the instances to launch
<code>--log_uri LOG_URI</code>	Location in S3 to store logs from the job flow, e.g. <code>s3n://mybucket/logs</code>
<code>--endpoint ENDPOINT</code>	Specify the web service endpoint to talk to
<code>-j, --jobflow JOB_FLOW_ID</code>	Job flow ID
<code>--instance-type INSTANCE_TYPE</code>	The type of the instances to launch
<code>-c CREDENTIALS_FILE</code>	File containing access_id and secret key
<code>--credentials</code>	
<code>-a, --access_id ACCESS_ID</code>	AWS Access Id
<code>-k, --secret_key SECRET_KEY</code>	AWS Secret Key
<code>-v, --verbose</code>	Turn on verbose logging of program interaction
<code>--debug</code>	Print stack traces when exceptions occur
<code>--version</code>	Print a version string
<code>-h, --help</code>	Show help message

How to Create a Job Flow

When developing steps for a job flow it is handy to keep a job flow | running and to add steps to it. This way if the step fails you can simply add another step without having to incur the startup cost of a job flow

The following command will start a job flow that will keep running and consuming resources until you terminate it.

To create a job flow

- From the command prompt, enter a command similar to the following:

```
$ ./elastic-mapreduce --create --alive --log-uri s3n://my-example-bucket/
logs
Created job flow j-36U2JMAE73054
```

By default this will launch a job flow running on a single m1.small instance. Later when you've got your steps running correctly on small sample data you'll want to launch job flows running on more instance. You can specify the number of instance and the type of instance to run with the `--num-instances` and `--instance-type` options.

The `--alive` option tells the job flow to keep running even when it has finished all its steps. The `log-uri` specifies a location in Amazon S3 for the log files from your job flow to be pushed.

It can be safely omitted if you haven't created a bucket yet in Amazon| S3. Log files are not pushed to Amazon S3 until 5 minutes after the step is complete, so for debug sessions you will most likely log onto the master node of your job flow. Specifying a `log-uri` is required if you want to be able to read log files from Amazon S3 after the job flow has terminated.

How to Create a Job Flow Using Pig

You can run Pig in interactive or batch modes. Typically, you use interactive mode to troubleshoot your job flow and batch mode in production.

Running Pig in Interactive Mode

1	<p>Use the <i>alive</i> option with the <i>create</i> command so that the job flow remains active until you terminate it.</p> <pre>\$ elastic-mapreduce --create --alive --name "Testing Pig -- \$USER" \ --num-instances 5 --instance-type c1.medium \ --pig-interactive</pre> <p>The return is similar to the following:</p> <pre>Created jobflow j-ABABABABABAB</pre>
2	<p>Wait for the job flow to start and reach a waiting state. Optionally, you can list running job flows using the following command:</p> <pre>\$ elastic-mapreduce --list --active</pre>
3	<p>Once the job flow is in the waiting state, SSH as the Hadoop user into to the master node and run Pig.</p> <pre>\$ elastic-mapreduce --jobflow j-ABABABABABAB --ssh \ ... \ hadoop@domU-12-12-12-12-12-12:~\$ pig \ Pig></pre> <p>You are now running Pig in interactive mode and can execute Pig queries.</p>

The following process shows how to run Pig in batch mode and assumes that you have stored the Pig script on Amazon S3. For more information about uploading files into Amazon S3, go to [Amazon S3 Bucket Creation](#) in the *Amazon Elastic MapReduce Getting Started Guide*.

Running Pig in Batch Mode

1	<p>Create a job flow with a step that executes a Pig script stored on Amazon S3.</p> <pre>\$ elastic-mapreduce --create \ --name "\$USER's Pig JobFlow" \ --pig-script \ --args s3://mybucket/myquery.q \ --args -d,INPUT=s3://mybucket/input,-d,OUTPUT=s3://mybucket/output</pre> <p>The <i>args</i> option provides arguments to the Pig-script. The first <i>args</i> option specifies the location of the Pig script in Amazon S3. In the second <i>args</i> option, the <i>-d</i> provides a way to pass values into the script (<i>INPUT</i>) and where to store results (<i>OUTPUT</i>). Within the Pig script these parameters are available as <i>\${variable}</i>. So, in this example Pig replaces <i>\${INPUT}</i> and <i>\${OUTPUT}</i> with the values passed in. These variables are substituted as a preprocessing step so may occur anywhere in a Pig script.</p> <p>The return is similar to the following:</p> <pre>Created jobflow j-ABABABABABAB</pre>
---	---

The following process shows how to add Pig as a new step in an existing job flow. Adding steps can help you test and develop Pig scripts. For example, if the script fails, you can add a new step to the job flow without having to wait for a new job flow to start.

Adding a Pig Script to an Existing Job Flow in Batch Mode

1	<p>Specify the location in Amazon S3 of a Pig script and associate it with an existing job flow.</p> <pre>\$ elastic-mapreduce --jobflow j-ABABABABABA \ --pig-script \ --args s3://mybucket/myquery.q \ --args -d,INPUT=s3://mybucket/input,-d,OUTPUT=s3://mybucket/output</pre>
---	---

How to Create a Job Flow Using Hive

You can run Hive in interactive or batch modes. Typically, you use interactive mode to troubleshoot your job flow and batch mode in production.

Running Hive in Interactive Mode

1	<p>Use the <code>alive</code> option with the <code>create</code> command so that the job flow remains active until you terminate it.</p> <pre>\$ elastic-mapreduce --create --alive --name "Testing Hive -- \$USER" \ --num-instances 5 --instance-type c1.medium \ --hive-interactive</pre> <p>The return is similar to the following:</p> <pre>Created jobflow j-ABABABABABAB</pre>
2	<p>Wait for the job flow to start and reach a waiting state. Optionally, you can list running job flows using the following command.</p> <pre>\$ elastic-mapreduce --list --active</pre>
3	<p>Once the job flow is in the waiting state, SSH as Hadoop user into to the master node and run Hive.</p> <pre>\$ elastic-mapreduce --jobflow j-ABABABABABAB --ssh \ ... \ hadoop@domU-12-12-12-12-12-12:~\$ hive \ Hive></pre> <p>You are now running Hive in interactive mode and can execute Hive queries.</p>

The following process shows how to run Hive in batch mode and assumes that you have stored the Hive script on Amazon S3. For more information about uploading files into Amazon S3, go to [Amazon S3 Bucket Creation](#) in the *Amazon Elastic MapReduce Getting Started Guide*.

Running Hive in Batch Mode

- 1 Create a job flow with a step that executes a Hive script stored on Amazon S3.

```
$ elastic-mapreduce --create \  
--name "$USER's Hive JobFlow" \  
--hive-script \  
--args s3://mybucket/myquery.q \  
--args -d,INPUT=s3://mybucket/input,-d,OUTPUT=s3://mybucket/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 passed in. These variables are substituted as a preprocessing step so it can occur anywhere in a Hive script.

The return is similar to the following:

```
Created jobflow j-ABABABABABAB
```

The following process shows how to add Hive as a new step in an existing job flow. Adding steps can help you test and develop Hive scripts. For example, if the script fails, you can add a new step to the job flow without having to wait for a new job flow to start.

Adding a Hive Script to an Existing Job Flow in Batch Mode

- 1 Specify the location in Amazon S3 of a Hive script and associate it with an existing job flow.

```
$ elastic-mapreduce --jobflow j-ABABABABABA \  
--hive-script \  
--args s3://mybucket/myquery.q \  
--args -d,INPUT=s3://mybucket/input,-d,OUTPUT=s3://mybucket/output
```

Storing a Table Schema Outside the EC2 Cluster

Hive stores schema information about the tables it manages in a MySQL database that runs, by default, on the master node. The schema information includes the location of data stored within Hive tables as well as the names and data types of the columns in each table. When a job flow terminates the associated EC2 instances shut down and the schema information, therefore, ceases to exist. (The data stored within tables located in Amazon S3 persists.) If you have multiple job flows that use the same relational database or run at the same time, you might like to make the schema information persist and thereby be available to multiple job flows.

There are two ways to share tables between job flows:

- Override the location of the MySQL database that stores the schema so that it resides somewhere persistent.
Use this option when you have a persistent Hive cluster in EC2 or if you have a persistent, backed up database instance.
- Insert `CREATE TABLE` statements at the top of your Hive scripts to recreate the schema information and issue `RECOVER PARTITION` statements for any of your partitioned tables.
Use this option when you don't have a persistent EC2 cluster or MySQL database instance.

Hive neither supports nor prevents concurrent write access to tables. If you share schema data between two job flows, you have to guarantee that you don't write to the same table concurrently from the job flows unless you're writing into different partitions of the same table.

You use the `hive-site` parameter to specify the location of the configuration file that overrides the default Hive values, for example,

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

The following process shows you how to override the default configuration values for Hive so that the schema information is stored outside of the job flow.

Creating a Job Flow Using a Schema Outside of the Cluster

1	<p>Set configuration values in <code>hive-site.xml</code>. If, for example, you have a MySQL instance running in EC2 and you adjusted your security groups to allow JDBC connections from instances running under your account and in the ElasticMapReduce-master security group, you would enter the following in your <code>hive-site.xml</code> configuration file that you store in Amazon S3.</p> <pre><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></pre> <p>The value for <code>javax.jdo.option.ConnectionURL</code> is the JDBC connection that provides access to the database where you want to store the schema information.</p>
2	<p>Create a job flow and specify a Hive configuration file on Amazon S3.</p> <pre>\$ elastic-mapreduce --create --alive \ --name "\$USER's Hive JobFlow" \ --hive-site=s3://mybucket/conf/hive-site.xml</pre> <p><code>--hive-site</code> installs the Hive configuration values stored in the specified location on Amazon S3. The <code>hive-site.xml</code> configuration values override the configuration values for Hive running on the job flow.</p>

How to Use Additional Files and Libraries With the Mapper or Reducer

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 into text.

To cache a file (or library), you run a streaming job flow and put the file in the working directory of the mapper and reducer. The file can be either a statically compiled binary or a script that your mapper or reducer application calls.

To add a file for the mapper or reducer to use

- In the **Extra Args** option, enter a command similar to the following:

```
-cacheFile s3n://bucket/path_to_executable#local_path
```

The file will be downloaded and appear in the working directory of your script as *local_path*.

To add an archive for the mapper or reducer to use

- In the **Extra Args** option, enter a command similar to the following:

```
-cacheArchive s3n://bucket/path_to_archive#local_directory
```

The archive will be downloaded and extracted into the working directory of your script as a directory named *local_directory/*.

For information about using DistributedCache to use data or application files in job flow, see [DistributedCache](#) (p. 82).

How to List Job Flows

Use the *list* option by itself or in combination with other options to list job flows in various states. This section presents some of those listings.

To list job flows created in the last two days

- Use the *list* option to list job flows:

```
$ ./elastic-mapreduce --list
```

The response is similar to the following:

			Example Job Flow
j-1YE2DN7RXJBWU	FAILED		
	CANCELLED	Custom Jar	
j-3GJ4FRRNKG97	COMPLETED	ec2-67-202-3-73.compute-1.amazonaws.com	
Example job flow			
j-5XXFIQS8PFNW	COMPLETED	ec2-67-202-51-30.compute-1.amazonaws.com	demo
3/24 s1			
	COMPLETED	Custom Jar	

This example shows there were three job flows created in the last two days. The indented lines are job flow steps. The columns for a job flow line are job flow ID, job flow state, master node DNS Name, and job flow name. The columns for a job flow step line are step state, and step name.

If you have not created any job flows in the last two days you will get no output from the command.

To list only active job flows

- Use the *list* and *active* options, as follows:

```
$ ./elastic-mapreduce --list --active
```

The response lists job flows that are starting, running, or shutting down.

To list only running or terminated job flows

- Use the *RUNNING* and *TERMINATED* options, as follows:

```
$ ./elastic-mapreduce --list --state RUNNING --state TERMINATED
```


The response lists job flows that are running, or have terminated down.

How to Get Information About a Job Flow

You can get information about a job flow using the *describe* option and a specified job flow ID.

To get information about a job flow

- Use the *describe* option with a valid job flow ID.

```
$ ./elastic-mapreduce --describe --jobflow[JobFlowId]
```

The response looks similar to the following:

```
{
  "JobFlows": [
    {
      "LogUri": null,
      "Name": "Development Job Flow",
      "ExecutionStatusDetail": {
        "EndDateTime": 1237948135.0,
        "CreationDateTime": 1237947852.0,
        "LastStateChangeReason": null,
        "State": "COMPLETED",
        "StartDateTime": 1237948085.0
      },
      "Steps": [],
      "Instances": {
        "Ec2KeyName": null,
        "InstanceCount": 1.0,
        "Placement": {
          "AvailabilityZone": "us-east-1a"
        },
        "KeepJobFlowAliveWhenNoSteps": false,
        "MasterInstanceType": "m1.small",
        "SlaveInstanceType": "m1.small",
        "MasterPublicDnsName": "ec2-67-202-3-73.compute-1.amazonaws.com",
        "MasterInstanceId": "i-39325750"
      },
      "JobFlowId": "j-3GJ4FRRNKG97"
    }
  ]
}
```

How to Add Steps to a Job Flow

You can add steps to a job flow only if you set the *RunJobFlow* parameter *KeepJobFlowAliveWhenNoSteps* to *True*. This value keeps the EC2 cluster engaged even after the successful completion of a job flow. If you already set the value to *False*, just revise the jar in the job flow and rerun it.

To add a step to a job flow using default parameter values

- Add the step using *j* option, as follows.

```
$ ./elastic-mapreduce -j j-36U2JMAE73054 --streaming
```

The `--streaming` argument adds a streaming step using default parameters. The default parameters are the word count example that is available in the Elastic MapReduce console.

You can see the step you just added using the console. In the console, refresh the job flow (j-36U2JMAE73054 in this example) you created, click it, and look at the detail pane in the lower half of the screen and you'll see the step you just added.

To add a step to a job flow using non-default parameter values

- Add the step using `j` option, as follows.

```
$ ./elastic-mapreduce -j j-36U2JMAE73054 \  
  --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
```

This command runs an example job flow step that downloads and runs the jar. The arguments are passed to the main function in the jar.

If your jar has a `manifest.mf` file, you do not need to specify the jar's main class using `--main-class`, as shown in the previous example.

How to Terminate a Job Flow

You use a job flow ID to specify the job flow you want to terminate.

To terminate a job flow

- Use the `terminate` command to terminate a job flow. This example uses job flow j-C019299B1X.

```
$ ./elastic-mapreduce --terminate j-C019299B1X
```

Additional CLI Tutorials

For additional CLI tutorials, go to <http://developer.amazonwebservices.com/connect/entry.jspa?externalID=2297&categoryID=269>.

Making API Requests

AWS Library Support

AWS provides libraries, sample code, tutorials, and other resources for software developers who prefer to build applications using language-specific APIs instead of Amazon Elastic MapReduce's REST and Query. These libraries provide basic functions (not included in Amazon Elastic MapReduce's REST and Query), such as request authentication, request retries, and error handling so that it's easier to get started. For more information about language-specific libraries and resources, go to:

- [Java](#)
- [PHP](#)
- [Ruby](#)
- [Windows and .NET](#)

For libraries and sample code in all languages, go to [Sample Code & Libraries](#).

REST and Query

Amazon Elastic MapReduce supports Query requests to call operations hosted by Amazon Elastic MapReduce servers. Query requests are simple HTTP requests using either the GET method with parameters in the URL or the POST method with parameters in the POST body. The response is an XML document that conforms to a WSDL.

A Query request is a URI. It is composed of the following parts.

- **Endpoint**—The URL of the web service where you send all Elastic MapReduce requests
For production requests, the endpoint is `elasticmapreduce.amazonaws.com`
- **Required and optional parameters**—These are specified by the operation
For more information about each operation's parameters, see the [API Reference](#).

The endpoint, which comes first, is separated by a question mark (?) from the parameters. The parameters are separated by ampersands (&) from one another. The following example shows a sample request formatted so that it is easy to read. The actual request would be one long string.

```
https://elasticmapreduce.amazonaws.com?
JobFlowIds.member.1=j-3UN6WX5RRO2AG&
Operation=DescribeJobFlows&
AWSAccessKeyId=[AWS Access Key ID]&
SignatureVersion=2&
SignatureMethod=HmacSHA256&
Timestamp=2009-01-28T21%3A49%3A59.000Z&
Signature=[calculated value]
```

Every request must contain the following parameters:

- **AWSAccessKeyId**—A value distributed by AWS when you sign up for an AWS account
To view your AWS Access Key ID, go to [Viewing Your AWS Account Identifiers](#) in the *Amazon Elastic MapReduce Getting Started Guide*.
- **Signature**—The signature enables Elastic MapReduce to guarantee that no one altered the request as it traveled across the Internet
- **Operation**—Specifies the operation
This value determines the parameters that can be used in the request. For descriptions of all Elastic MapReduce operations and their parameters, see the [API Reference](#).

For a list of parameters that must be included in every request, see [Common Request Parameters](#).

Because the request is a URI it cannot contain any spaces and all of the characters, such as ampersand (&) must be URL-encoded. So, for example, any spaces in a value are converted to %20.

Request Endpoints

An endpoint is a URL that is the entry point for a web service. Every web service request contains an endpoint. Amazon S3 REST requests use the following SSL secured or unsecured endpoints:

- <http://elasticmapreduce.amazonaws.com>
- <https://elasticmapreduce.amazonaws.com>,
- <http://us-east-1.elasticmapreduce.amazonaws.com>
- <https://us-east-1.elasticmapreduce.amazonaws.com>
- <http://us-west-1.elasticmapreduce.amazonaws.com>
- <https://us-west-1.elasticmapreduce.amazonaws.com>

Region-specific endpoints, such as <http://us-west-1.elasticmapreduce.amazonaws.com>, indicate the general location of the Amazon S3 servers.

How to Create Signatures

Web service requests are sent across the Internet and are subject to tampering. Amazon Elastic MapReduce uses the signature value to determine if any of the parameters or parameter values were changed in transit. For Query requests, you calculate this value or use a library that calculates this value for you and include the value in every request. The console and command line interface calculate and include a signature value in every request.

To create a signature

1. Create the canonicalized query string that you need later in this procedure:
 - a. Sort the UTF-8 query string components by parameter name with natural byte ordering.
 - b. URL encode the parameter name and values according to the following rules:
 - Do not URL encode any of the unreserved characters that RFC 3986 defines.

These unreserved characters are A-Z, a-z, 0-9, hyphen (-), underscore (_), period (.), and tilde (~).

- Percent encode all other characters with %XY, where X and Y are hex characters 0-9 and uppercase A-F.
- Percent encode extended UTF-8 characters in the form %XY%ZA....
- Percent encode the space character as %20 (and not +, as common encoding schemes do).



Note

Currently, all AWS service parameter names use unreserved characters, so you don't need to encode them. However, you might want to include code to handle parameter names that use reserved characters for possible future use.

- c. Separate the encoded parameter names from their encoded values with the equals sign (=) (ASCII character 61), even if the parameter value is empty.
 - d. Separate the name-value pairs with an ampersand (&) (ASCII code 38).
2. Create the string to sign according to the following pseudo-grammar (the "\n" represents an ASCII newline).

```
StringToSign = HTTPVerb + "\n" +  
               ValueOfHostHeaderInLowercase + "\n" +  
               HTTPRequestURI + "\n" +  
               CanonicalizedQueryString <from the preceding step>
```

The HTTPRequestURI component is the HTTP absolute path component of the URI up to, but not including, the query string. If the HTTPRequestURI is empty, use a forward slash (/).

3. Calculate an RFC 2104-compliant HMAC with the string you just created, your Secret Access Key as the key, and SHA256 or SHA1 as the hash algorithm.
4. Convert the resulting value to base64.
5. Use the resulting value as the value of the *Signature* request parameter.



Important

The final signature you send in the request must be URL encoded as specified in RFC 3986. If your toolkit URL encodes your final request, then it handles the required URL encoding of the signature. If your toolkit doesn't URL encode the final request, then make sure to URL encode the signature before you include it in the request. Most importantly, make sure the signature is URL encoded *only once*. A common mistake is to URL encode it manually during signature formation, and then again when the toolkit URL encodes the entire request.

Starting and Managing Job Flows Using the API

Topics

- [Amazon Elastic MapReduce Libraries](#) (p. 58)
- [How to Start a Job Flow](#) (p. 59)
- [How to Add Steps to a Job Flow](#) (p. 61)
- [How to Use Additional Files and Libraries With the Mapper or Reducer](#) (p. 62)
- [How to Get Information About a Job Flow](#) (p. 62)
- [How to Terminate a Job Flow](#) (p. 64)

This section describes the operations you use to run, modify, describe, and terminate a Hadoop job flow. The order of explanation is in the typical order that the operations would be applied.



Important

The maximum request rate for Elastic MapReduce is one request every ten seconds.

Amazon Elastic MapReduce Libraries

AWS has created libraries to implement Amazon Elastic MapReduce in the following computer languages:

- VB.NET
- PHP
- C#
- Java
- Perl

For more information, go to [Sample Code & Libraries](#).

How to Start a Job Flow

This section describes how to start a job flow.

To start a job flow using a jar

- Send a `RunJobFlow` request similar to the following:

```
https://elasticmapreduce.amazonaws.com?
Operation=RunJobFlow&
Name=MyJobFlowName&
LogUri=s3://mybucket/subdir&
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=s3://mybucket/MyJarFile&
Steps.member.1.HadoopJarStep.MainClass=MyMainClass&
Steps.member.1.HadoopJarStep.Args.member.1=arg1&
Steps.member.1.HadoopJarStep.Args.member.2=arg2&
AWSAccessKeyId=[AWS Access Key ID]&
SignatureVersion=2&
SignatureMethod=HmacSHA256&
Timestamp=2009-01-28T21%3A48%3A32.000Z&
Signature=[calculated value]
```

For an explanation of the parameters unique to `RunJobFlow`, see [RunJobFlow](#). For more information about the generic parameters in the request, see [Common Request Parameters](#).

The response includes a request ID, which you use in other Amazon Elastic MapReduce operations, for example, to describe or terminate the job flow. For this reason, it is important to store request IDs.

To find Elastic MapReduce code samples and sample applications, go to [Sample Data Processing Applications](#).

After starting a job flow, you might like to monitor its progress. For more information, see [How to Monitor Job Flows \(p. 66\)](#). Or, you might like to find job flows in a particular state, such as `COMPLETED`. For more information, see [How to Get Information About a Job Flow \(p. 62\)](#).

Sample Applications

AWS has tutorials that show you how to create complete applications, including:

- [Apache LogAnalysis using Pig](#)
- [Processing and Loading Data from Amazon S3 to the Vertica Analytic Database](#)
- [LogAnalyzer for Amazon CloudFront](#)
- [Cascading.Multitool](#)
- [FreeBase](#)
- [ItemSimilarity](#)
- [CloudBurst](#)
- [Word Count Example](#)

For more information, go to [Sample Data Processing Applications](#).

Job Flows with Multiple Steps

The *Steps* parameters define the location and input parameters for the Hadoop jar steps that perform the processing on the input data. The maximum number of steps allowed per job flow is 256. Each step is identified by a *member* number. In the previous example, the *member* number is 1. If there were two steps in this job flow, there would be a similar series of *Steps* definitions with a member number of 2, for example, `Steps.member.2.HadoopJarStep.Jar=MySecondJarFile&`. A *member* number is a means of associating related parameters, as follows.

```
...
Steps.member.1.Name=MyStepName&
Steps.member.1.ActionOnFailure=CONTINUE&
Steps.member.1.HadoopJarStep.Jar=s3://mybucket/MyJarFile&
Steps.member.1.HadoopJarStep.MainClass=MyMainClass&
Steps.member.1.HadoopJarStep.Args.member.1=arg1&
Steps.member.1.HadoopJarStep.Args.member.2=arg2&
Steps.member.2.Name=MyStepName2&
Steps.member.2.ActionOnFailure=CONTINUE&
Steps.member.2.HadoopJarStep.Jar=s3://mybucket/MyJarFile2&
Steps.member.2.HadoopJarStep.MainClass=MyMainClass2&
Steps.member.2.HadoopJarStep.Args.member.1=arg3&
Steps.member.2.HadoopJarStep.Args.member.2=arg4&
...
```

The *name* parameter helps you distinguish step results. So, make each name unique. Amazon Elastic MapReduce does not check for the uniqueness of step names.

The remainder of the *Steps* parameters specify the jar and the input parameters it uses to process the data.

EC2 Instances

The *Instances* parameters enable you to configure the kind and number of EC2 instances you would like to process the data. Hadoop spreads the processing of the data across multiple EC2 instances. The master instance is responsible for keeping track of the health of the slave instances and polling the slave instances for job result status. The slave instances do the actual processing of the data. If you have only one instance, it serves as both the master and the slave. If you have 2 or more instances, one is the master node and the rest are slave nodes.

The *KeepJobAlive* parameter in a *RunJobFlow* request determines whether to terminate the EC2 cluster when it runs out of steps to execute. Set this value to `False` when you know that the job flow is running as expected. When you are troubleshooting the job flow and adding steps while the job flow execution is suspended, set the value to `True` so that you don't spend time uploading the results to Amazon S3, only to turn around after modifying a step to start a new EC2 cluster, download the raw data from Amazon S3, and restart the job flow.

If *KeepJobAlive* is `true`, after successfully getting the job flow to complete its work, you must send a *TerminateJobFlows* request.

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

EC2 locations are composed of Availability Zones and Regions. Regions are dispersed and located in separate geographic areas (e.g., US-West and EU). Availability Zones are distinct locations within a Region 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. Currently, Elastic MapReduce is available in the following Regions: US Standard (the default), US-West, and EU. For more information, see [Regions \(p. 18\)](#).

The *AvailabilityZone* parameter specifies the general location of the EC2 cluster. This parameter is optional and, in general, we discourage its use. When *AvailabilityZone* is not specified we automatically pick the best *AvailabilityZone* for the job flow. You might find this parameter useful if you want to co-locate your instances with other existing running instances, and your job flow needs to read or write data from those instances. For more information, see the [Amazon Elastic Compute Cloud Developer Guide](#).

Args Argument

The *Args* argument contains location information for your input data, output data, mapper, reducer, and cache file, as shown in the following example.

```
"Name": "mapreduce phase 2",
  "ActionOnFailure": "CONTINUE",
  "HadoopJarStep": {
    "Jar": "/home/hadoop/contrib/streaming/hadoop-0.18.3-streaming.jar",
    "Args": [
      "-input",      "s3://home/hadoop/sampleInput2/",
      "-output",     "s3://aws-hadoop/MyCompany/SampleOutput2/",
      "-mapper",     "python item_item_similarity.py mapper2",
      "-reducer",    "python item_item_similarity.py reducer2",
      "-cacheFile",  "s3://aws-hadoop/MyCompany/
item_item_similarity.py#item_item_similarity.py"
    ]
  }
```

All paths are prefixed with their location. "s3://" refers to the s3n file system. If you use the HDFS file system, prepend the path with `hdfs:///` (make sure to use three slashes (///)), for example, `hdfs:///home/hadoop/sampleInput2/`.

How to Add Steps to a Job Flow

Typically, you specify all job flow steps in a *RunJobFlow* request. The value of *AddJobFlowSteps* is that you can add steps to a job flow while it is already loaded onto the EC2 instances. You typically add steps to modify the data processing or to aid in debugging a job flow when you are working interactively with the job flow, that is, you are adding steps to the job flow while the job flow execution has paused.

When you are debugging a job flow, you must set the *RunJobFlow* parameters, *KeepJobAliveWhenNoSteps*, to `True` and *ActionOnFailure* to `CANCEL_AND_WAIT`.

The maximum number of steps allowed in a job flow is 256.

To add steps to a job flow

- Send a request similar to the following.

```
https://elasticmapreduce.amazonaws.com?
JobFlowId=j-3UN6WX5RRO2AG&
Steps.member.1.Name=MyStep2&
Steps.member.1.ActionOnFailure=CONTINUE&
Steps.member.1.HadoopJarStep.Jar=s3://mybucket/MySecondJar&
Steps.member.1.HadoopJarStep.MainClass=MainClass&
Steps.member.1.HadoopJarStep.Args.member.1=arg1&
Operation=AddJobFlowSteps&
```

```
AWSAccessKeyId=[AWS Access Key ID]&  
SignatureVersion=2&  
SignatureMethod=HmacSHA256&  
Timestamp=2009-01-28T21%3A51%3A51.000Z&  
Signature=[calculated value]
```

For more information about the parameters unique to `AddJobFlowSteps`, see [AddJobFlowSteps](#).
For more information about the generic parameters in the request, see [Common Request Parameters](#).

The response contains the request ID.

After adding steps to a job flow, you might like to monitor its progress. For more information, see [How to Monitor Job Flows](#) (p. 66).

How to Use Additional Files and Libraries With the Mapper or Reducer

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 into text.

To cache a file (or library), you run a streaming job flow and put the file in the working directory of the mapper and reducer.

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`, would be in the working directory of the mapper, which could reference the file.

How to Get Information About a Job Flow

The `DescribeJobFlows` operation returns extensive details about specified job flows. You specify job flows by their ID, creation date, or state. Elastic MapReduce returns descriptions of job flows that are up to two months old. Specifying an older date returns an error. If you do not specify a `CreatedAfter` value, Elastic MapReduce uses the default of two months.

Each of the input parameters acts as a filter so that Elastic MapReduce returns information about a more precise set of job flows with each parameter you use in your request. When no parameters are used in the request Elastic MapReduce returns descriptions of all job flows that have:

- Not ended within the last two months
These jobs are in one of the following states: `RUNNING`, `WAITING`, `SHUTTING_DOWN`, `STARTING`.
- Ended and created in the last two weeks

For more information, see [Job Flow States](#) (p. 63).

To return information about a job flow identified by its job ID

- Issue a request similar to the following.

```
https://elasticmapreduce.amazonaws.com?
JobFlowIds.member.1=j-3UN6WX5RR02AG&
Operation=DescribeJobFlows&
AWSAccessKeyId=[AWS Access Key ID]&
SignatureVersion=2&
SignatureMethod=HmacSHA256&
Timestamp=2009-01-28T21%3A49%3A59.000Z&
Signature=[calculated value]
```

For more information about the input parameters unique to `DescribeJobFlows`, go to [DescribeJobFlows](#). For more information about the generic parameters in the request, see [Common Request Parameters](#).

To return information about a job flows in a COMPLETED state

- Issue a request similar to the following.

```
https://elasticmapreduce.amazonaws.com?
JobFlowStates=COMPLETED&
Operation=DescribeJobFlows&
AWSAccessKeyId=[AWS Access Key ID]&
SignatureVersion=2&
SignatureMethod=HmacSHA256&
Timestamp=2009-01-28T21%3A49%3A59.000Z&
Signature=[calculated value]
```

For more information about the input parameters unique to `DescribeJobFlows`, go to [DescribeJobFlows](#). For more information about the generic parameters in the request, see [Common Request Parameters](#).

The response contains information about the EC2 cluster, the status of the job and steps, and so forth. For more information, go to [DescribeJobFlows](#). For more information about step states, see [Step States](#) (p. 63).

Job Flow States

The job flow can be in one of the following states:

- **COMPLETED**—The job flow shut down after all steps completed successfully
- **FAILED**—The job flow shut down after a step failed or due to an internal error
- **RUNNING**—A step for the job flow is currently being run
- **SHUTTING_DOWN**—The job flow is in the process of shutting down
- **STARTING**—The job flow is provisioning and configuring its EC2 instances
- **TERMINATED**—The job terminated on request of the user
- **WAITING**—The job flow is currently active, but has no steps to run

Step States

Job flow steps can be in the following states:

- **CANCELLED**—The step was canceled before running – because an earlier step failed or a job flow was terminated before it could run
- **COMPLETED**—The step completed successfully

- **FAILED**—The step failed while running
- **PENDING**—The step is waiting to be run
- **RUNNING**—The step is currently running

How to Terminate a Job Flow

The `TerminateJobFlows` operation ends step processing, uploads any log data from EC2 to Amazon S3 (if so configured), and terminates the EC2 cluster. A job flow also terminates automatically if you set `KeepJobAliveWhenNoSteps` to `False` in a `RunJobFlows` request.

You can use this action to terminate either a single job flow or a list of job flows by their job flow IDs.

The following request shows how to terminate a job flow specified by `JobFlowIds`.

To terminate three job flows

- Issue a request similar to the following.

```
https://elasticmapreduce.amazonaws.com?
JobFlowIds.member.1=j-3UN6SOUERO2AG,j-3UN6WX5RR438r7,j-3UN6DUER23849&
Operation=TerminateJobFlows&
AWSAccessKeyId=[AWS Access Key ID]&
SignatureVersion=2&
SignatureMethod=HmacSHA256&
Timestamp=2009-01-28T21%3A53%3A50.000Z&
Signature=[calculated value]
```

For more information about the input parameters unique to `TerminateJobFlows`, go to [TerminateJobFlows](#). For more information about the generic parameters in the request, see [Common Request Parameters](#).

The response contains the request ID.

Monitoring and Troubleshooting Job Flows

Topics

- [Log Files](#) (p. 65)
- [How to Monitor Job Flows](#) (p. 66)
- [Troubleshooting](#) (p. 76)

This section describes how you can monitor and troubleshoot your job flows and the log files produced by Hadoop and Elastic MapReduce that provide information about the errors.

Log Files

The Elastic MapReduce job flow provides a jar (or streaming file) and initiates the Hadoop application on your EC2 instances. Both Elastic MapReduce and Hadoop produce log files, which describe the completion status of every step and task within a job flow. Elastic MapReduce groups the log files from all of the EC2 instances into one location that you specify in the *LogUri* parameter in the *RunJobFlow* operation.

Log File Directories

When you look in Amazon S3 at the bucket you specified with the *LogUri* parameter you find folders labeled with job IDs. Within each folder is a folder labeled *Steps*, and within that folder is a folder for each of the steps in the job flow. Each step folder contains a link to a variety of log files named *syslog*, *stdout*, *controller*, and *stderr*. Hadoop generates the files logged in *syslog* and Elastic MapReduce generates the files logged in *stdout* and *stderr*, as shown in the following example.

```
Task Logs: 'task_200807301447_0001_m_000000_0'
```

```
stdout logs
```

```
map: key = test
map: key = test2

stderr logs

syslog logs
2008-07-30 14:51:16,410 INFO org.apache.hadoop.metrics.jvm.JvmMetrics:
Initializing JVM Metrics with processName=MAP, sessionId=
2008-07-30 14:51:16,507 INFO org.apache.hadoop.mapred.MapTask:
numReduceTasks: 1
2008-07-30 14:51:17,120 INFO org.apache.hadoop.mapred.TaskRunner: Task
'task_200807301447_0001_m_000000_0' done.
```

How to Monitor Job Flows

Topics

- [Determining the DNS of the Master Node \(p. 66\)](#)
- [How to Monitor Job Flow Status Using SSH \(p. 67\)](#)
- [How to Install FoxyProxy \(p. 68\)](#)
- [How to Download Job Flow Logs from Amazon S3 \(p. 71\)](#)
- [How to Use the Hadoop User Interface \(p. 71\)](#)

There are two approaches to monitoring job flows.

- Set up an SSH tunnel between your host and the EC2 master node where you can look on the file system for log files (as described in [Log File Directories \(p. 65\)](#)) or at the job flow statistics published by the Hadoop web server
- Download log files from an Amazon S3 bucket after running a job flow

The master node in the cluster contains summary information of all of the work done by the slave nodes. You can, however, explore the working and error logs on each slave node in an effort to resolve problems occurring in the execution of the job flow.

Determining the DNS of the Master Node

You need the DNS of the master node to log in so that you can inspect the log files. This section explains how to discover the DNS of the master node.

To determine the DNS of a master node

- Use the `--list` option as follows:
For Linux/UNIX, enter:

```
$ ./elastic-mapreduce --list --jobflow [Your job flow ID]
```

For Microsoft Windows, enter:

```
$ ruby elastic-mapreduce --list --jobflow [Your job flow ID]
```

In the response, the third column lists the DNS name of the master node if that node is currently running.

If you don't know the job flow ID, use the `--list` option with the `--active` option (instead of the `--jobflow` option) to list all active job flows.

How to Monitor Job Flow Status Using SSH

You can use SSH port forwarding to set up a secure link between your computer and the master node in the EC2 cluster that is processing your job flow. To SSH as Hadoop user into the master node, your job flow status must be either `WAITING` or `RUNNING`. To make the job flow remain in a `WAITING` state even after successful completion, use the `--alive` option in the `CreateJobFlow` command. For more information, see [How to Create a Job Flow](#) (p. 47).

To view the Elastic MapReduce logs on the EC2 master node

1. Open an SSH shell and use an SSH command of the following form to set up an SSH connection as the Hadoop user between your host and the EC2 master node.

```
ssh -i [keyfile.pem] hadoop@[EC2_master_node_DNS]
```

Substitute the pem file from your own keypair and the public DNS name of the master node. The following is an example for `myKeyPairName.pem` at `ec2-67-202-49-73`.

```
ssh -i ~/ec2-keys/myKeyPairName.pem  
hadoop@ec2-67-202-49-73.compute-1.amazonaws.com
```

For `keyfile`, use the value you set for:

- **SSH Key Name** in the console
- `Ec2KeyName` in a CLI
- `key` in a `RunJobFlow` request

The key name provides a handle to the master node and enables you to log into it with account Hadoop without using a password. You cannot SSH into the master node if you did not set a value for **SSH Key Name** or `Ec2KeyName`. For more information, see [Creating a Job flow](#) (p. 23).

For the `EC2_master_node_DNS`, use the value returned for it in the console or from `DescribeJobFlows`. You always log in as Hadoop.



Note

As an alternative to SSH, you can use a utility, such as PuTTY.

If you get an error executing the `ssh` command, you might not have set the permissions on the `mykey.pem` file, the key file might be specified incorrectly, or you copied the DNS name incorrectly.

2. Navigate to `/mnt/var/log/hadoop/steps/1` to see the logs on the master node for the first step. The second step log files are in `/mnt/var/log/hadoop/steps/2` and so on. The log files are:
 - **controller**—Log file of the process that attempts to execute your step
 - **syslog**—Log file generated by Hadoop that describes the execution of your Hadoop job by the job flow step
 - **stderr**—A stderr log file generated by Hadoop when it attempts to execute your job flow
 - **stdout**—The stdout log file generated by Hadoop when it attempts to execute your job flow

These log files may not appear until the step has run for some time, or finished or failed. These logs contain counter and status information.


```
2009-01-20 07:59:44,146 INFO org.apache.hadoop.mapred.JobTracker: STARTUP_MSG:
/***** STARTUP_MSG: Starting
JobTracker STARTUP_MSG: host = domU-12-31-39-00-ED-78.compute-1.internal/10.254.242.134
STARTUP_MSG: args = [] STARTUP_MSG: version = 0.18 STARTUP_MSG: build = -r ; compiled by
'root' on Mon Jan 5 21:29:34 UTC 2009
*****/ 2009-01-20 07:59:44,548
INFO org.apache.hadoop.ipc.metrics.RpcMetrics: Initializing RPC Metrics with hostName=JobTracker,
port=9001 2009-01-20 07:59:44,554 INFO org.apache.hadoop.ipc.Server: IPC Server Responder: starting
2009-01-20 07:59:44,556 INFO org.apache.hadoop.ipc.Server: IPC Server listener on 9001: starting
2009-01-20 07:59:44,556 INFO org.apache.hadoop.ipc.Server: IPC Server handler 0 on 9001: starting
2009-01-20 07:59:44,557 INFO org.apache.hadoop.ipc.Server: IPC Server handler 1 on 9001: starting
2009-01-20 07:59:44,557 INFO org.apache.hadoop.ipc.Server: IPC Server handler 2 on 9001: starting
2009-01-20 07:59:44,557 INFO org.apache.hadoop.ipc.Server: IPC Server handler 3 on 9001: starting
2009-01-20 07:59:44,557 INFO org.apache.hadoop.ipc.Server: IPC Server handler 4 on 9001: starting
2009-01-20 07:59:44,557 INFO org.apache.hadoop.ipc.Server: IPC Server handler 5 on 9001: starting
2009-01-20 07:59:44,557 INFO org.apache.hadoop.ipc.Server: IPC Server handler 6 on 9001: starting
2009-01-20 07:59:44,558 INFO org.apache.hadoop.ipc.Server: IPC Server handler 7 on 9001: starting
2009-01-20 07:59:44,558 INFO org.apache.hadoop.ipc.Server: IPC Server handler 8 on 9001: starting
2009-01-20 07:59:44,558 INFO org.apache.hadoop.ipc.Server: IPC Server handler 9 on 9001: starting
2009-01-20 07:59:49,672 INFO org.mortbay.util.Credential: Checking Resource aliases 2009-01-20
07:59:49,909 INFO org.mortbay.http.HttpServer: Version Jetty/5.1.4 2009-01-20 07:59:49,909 INFO
org.mortbay.util.Container: Started HttpContext[/static,/static] 2009-01-20 07:59:49,910 INFO
org.mortbay.util.Container: Started HttpContext[/logs,/logs] 2009-01-20 07:59:50,535 INFO
org.mortbay.util.Container: Started org.mortbay.jetty.servlet.WebApplicationHandler@15d616e 2009-01-20
07:59:50,697 INFO org.mortbay.util.Container: Started WebApplicationContext[/,/] 2009-01-20
07:59:50,699 INFO org.mortbay.http.SocketListener: Started SocketListener on 0.0.0.0:50030 2009-01-20
07:59:50,699 INFO org.mortbay.util.Container: Started org.mortbay.jetty.Server@1a897a9 2009-01-20
07:59:50,701 INFO org.apache.hadoop.metrics.jvm.JvmMetrics: Initializing JVM Metrics with
processName=JobTracker, sessionId= 2009-01-20 07:59:50,703 INFO
```



Note

If you specified a log URI where Elastic MapReduce uploads log files onto Amazon S3, you can inspect the log files on Amazon S3. There is, however, a five minute delay between when the log files stop being written and when they're pushed into Amazon S3. So, it's a faster to look at the log files on the master node, especially if the step failed quickly

How to Install FoxyProxy

Hadoop provides a user interface called FoxyProxy that makes job flow related information, such as job flow counters and status, available. This user interface runs automatically on a web server running on the master node. To access that data, you set up a SOCKS server on your computer, and an SSH tunnel (as the Hadoop user) between your computer and the master node in the EC2 cluster that is processing your job flow. This tunnel is also known as *port forwarding*.

The following procedure explains how to install FoxyProxy so that you can access the Hadoop UI.

To install FoxyProxy

1. Start an SSH SOCKS server somewhere within your firewall (outside of EC2) using a command of the following form.

```
ssh -I [keyfile] -ND [port_number] hadoop@[EC2_master_node_DNS]
```

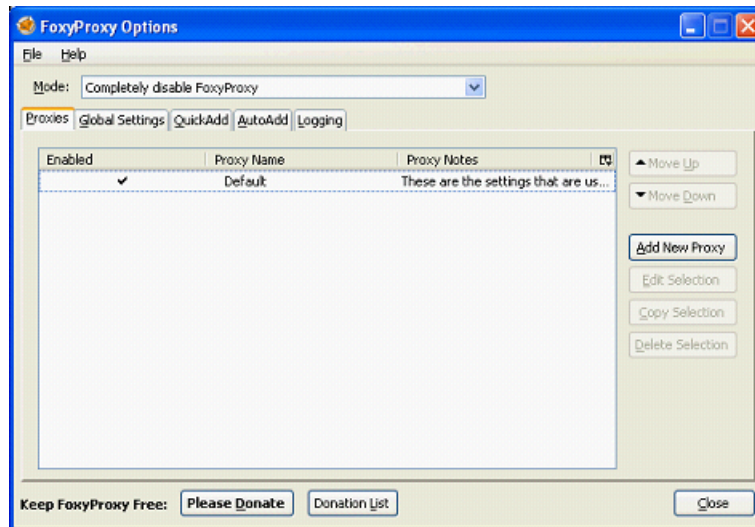
The following is an example for myKeyPairName at ec2-67-202-49-73.

```
ssh -i ~/ec2-keys/myKeyPairName -ND 8157
hadoop@ec2-67-202-49-73.compute-1.amazonaws.com
```

2. Download and install FoxyProxy from <http://foxyproxy.mozdev.org/downloads.html> and restart Firefox.

To configure FoxyProxy

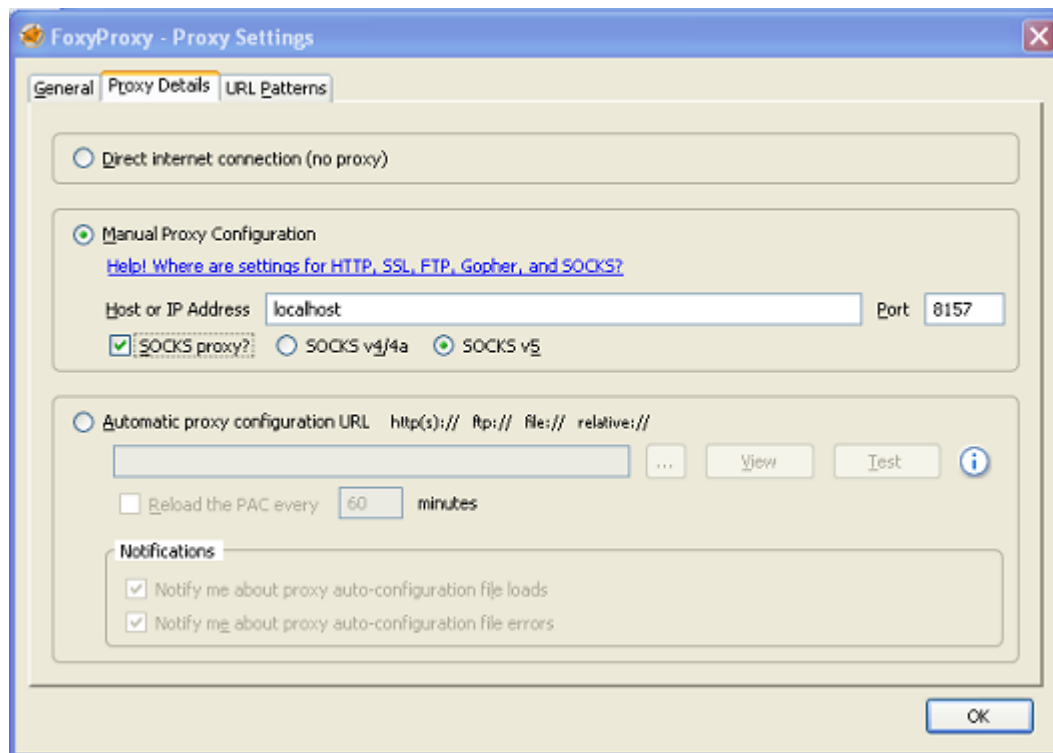
1. On the Firefox **Tools** menu, click **FoxyProxy**, and then select **Options**.
FoxyProxy displays the **FoxyProxy Options** window.



Note

This guide shows screen shots of FoxyProxy version 2.8.11.

2. On the **Proxies** tab click **Add New Proxy**.
The **FoxyProxy - Proxy Settings** dialog box opens.



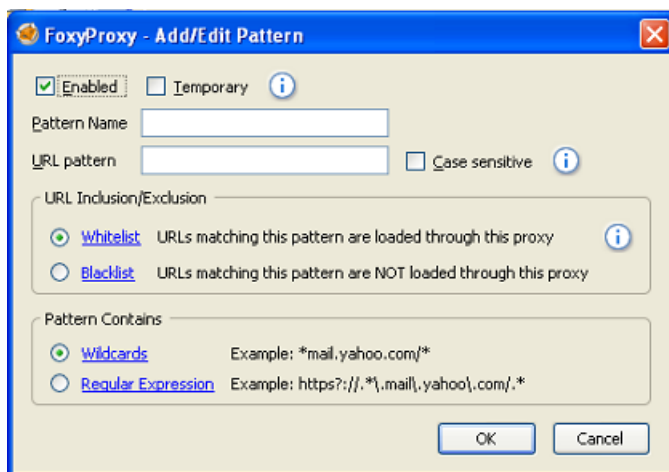
3. On the **General** tab enter a proxy name.

4. On the **Proxy Details** tab do the following.

A	Select Manual Proxy Configuration option and enter the host name and port number of the host you ran the SSH command as the Hadoop user from in step 1. In this case, we are running the proxy on our desktop so we enter localhost and port 8157.
B	Select the SOCKS proxy? check box.
C	Select SOCKS v5 .

5. On the **URL Patterns** tab click **Add New Pattern**.

The **FoxyProxy - Add/Edit Pattern** dialog box opens.



6. In the **FoxyProxy - Add/Edit Pattern** pane do the following.

A	Click the Enabled check box, click the Wildcards radio button, enter a name in the Pattern Name box, and enter the following URL pattern in the URL pattern box: <code>*ec2*.amazonaws.com*</code>
B	Click OK to close the FoxyProxy - Add/Edit Pattern pane.

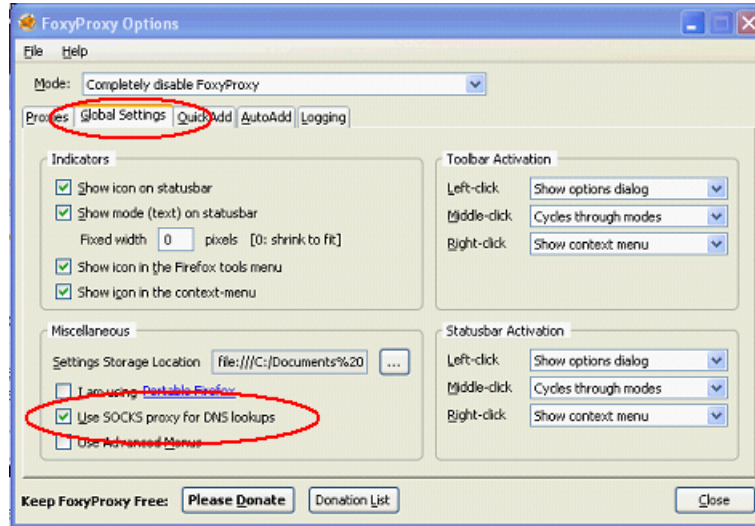
7. In the **FoxyProxy - Add/Edit Pattern** pane do the following.

A	Click the Enabled check box, click the Wildcards radio box, enter a name in the Pattern Name box, and enter the following URL pattern in the URL pattern box: <code>*ec2.internal*</code> Now when you browse to a URL that matches this pattern, the associated proxy is used to load that URL. This proxy establishes the secure port forwarding connection between your host and the master node.
B	Click OK to close the FoxyProxy - Add/Edit Pattern pane.

8. On the **FoxyProxy - Proxy Settings** pane, click **OK** to close it.

9. On the **FoxyProxy Options** pane, click the down arrow on the **Mode** pull down menu and select **Use proxies based on their predefined patterns and priorities**.

10. On the **Global Settings** tab, click the **Use SOCKS proxy for DNS lookups** check box, and then click **Close**.



11 Restart Firefox at the prompt.

Now, whenever you enter a URL that matches the pattern `*ec2*.amazonaws.com*` into the Firefox browser, Firefox sets up an SSH tunnel with the EC2 cluster's master node.

How to Download Job Flow Logs from Amazon S3

Instead of viewing logs on the EC2 master node, you can download the logs from Amazon S3. You can download the data in a bucket using the Amazon S3 `GetBucket` operation or using the Amazon S3 Organizer plugin for Firefox. To download the plugin, go to <https://addons.mozilla.org/en-US/firefox/addon/3247>.

To download log files from Amazon S3

1. Locate the value you supplied for `LogURI` in the console, CLI, or `RunJobFlow` request.
The `LogURI` is a path to a bucket on Amazon S3 of the form `s3n://[bucketName]/[path]`.
2. Download the logs in the bucket using the Amazon S3 Organizer plugin with Firefox, or the Amazon S3 `GET Bucket` operation.
Amazon S3 downloads the log files in the bucket.

How to Use the Hadoop User Interface

The Hadoop software publishes job flow status to an internally running web server on the master node of the EC2 cluster. You can view the job flow status by accessing this web server. The web user interfaces (UIs) are, by default, located as follows:

- `http://[master_dns_name]:9100/` - web UI for MapReduce job tracker(s)
- `http://[master_dns_name]:9101/` - web UI for HDFS name node(s)

To relocate these UIs, edit `conf/hadoop-default.xml`.

To view the Hadoop Distributed File System UI

- Go to `http://[master_dns_name]:9101/`.

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-B3	2	In Service	146.77	0	<div></div>	136.35	4
domU-12-31-38-00-52-03	2	In Service	146.77	0	<div></div>	136.35	6
domU-12-31-38-00-79-31	0	In Service	146.77	0	<div></div>	136.35	4
domU-12-31-38-00-79-58	2	In Service	146.77	0	<div></div>	136.35	6

Dead Datanodes : 0

To view the job flow status using the Hadoop UI

1. To access the Hadoop Job Tracker UI running on the master node, go to `http://[master_dns_name]:9100/`.

You can use the console to get the value for *master_dns_name*. The Hadoop UI opens.

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, r
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	<div>100.00%</div>	11	11	<div>100.00%</div>	1	1
job_200901200759_0002	hadoop	job.jar	<div>100.00%</div>	2	2	<div>100.00%</div>	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 job flows are 100% complete.

2. Click a job flow ID.

Hadoop displays information about the selected job flow.

Hadoop job_200901200759_0001 on domU-12-31-39-00-ED-78

User: hadoop
 Job Name: averaging pass
 Job File: hdfs://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 job flow counters. It also shows that zero tasks failed but two tasks were killed.

3. Choose one of the following actions:

To...

Do this...

Find out more about the killed tasks

Click on an entry in the **Failed/Killed Task Attempts** column.

Hadoop

[job_200901200759_0001](#)

failures on

[domU-12-31-39-00-ED-78](#)

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

To...	Do this...																																										
Get more information about the mapper tasks	<p>Click map. Hadoop displays all of the tasks completed and their status.</p> <div><h3>Hadoop map task list for job 200901200759_0001 on domU-12-31-39-00-ED-78</h3><table><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-ae:0+286875349</td><td>20-Jan-2009 08:01:57</td><td>20-Jan-2009 08:09:18 (7mins, 21sec)</td><td></td><td>8</td></tr><tr><td>task_200901200759_0001_m_000001</td><td>100.00%</td><td>s3n://anhi-test-data/netflix/input/part-aa:0+279753329</td><td>20-Jan-2009 08:01:57</td><td>20-Jan-2009 08:08:10 (6mins, 13sec)</td><td></td><td>8</td></tr><tr><td>task_200901200759_0001_m_000002</td><td>100.00%</td><td>s3n://anhi-test-data/netflix/input/part-ag:0+276353511</td><td>20-Jan-2009 08:01:58</td><td>20-Jan-2009 08:08:34 (6mins, 36sec)</td><td></td><td>8</td></tr><tr><td>task_200901200759_0001_m_000003</td><td>100.00%</td><td>s3n://anhi-test-data/netflix/input/part-af:0+276195815</td><td>20-Jan-2009 08:01:58</td><td>20-Jan-2009 08:08:33 (6mins, 35sec)</td><td></td><td>8</td></tr><tr><td>task_200901200759_0001_m_000004</td><td>100.00%</td><td>s3n://anhi-test-data/netflix/input/part-ab:0+276051944</td><td>20-Jan-2009 08:08:10</td><td>20-Jan-2009 08:14:58 (6mins, 47sec)</td><td></td><td>8</td></tr></tbody></table></div> <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-ae:0+286875349	20-Jan-2009 08:01:57	20-Jan-2009 08:09:18 (7mins, 21sec)		8	task_200901200759_0001_m_000001	100.00%	s3n://anhi-test-data/netflix/input/part-aa:0+279753329	20-Jan-2009 08:01:57	20-Jan-2009 08:08:10 (6mins, 13sec)		8	task_200901200759_0001_m_000002	100.00%	s3n://anhi-test-data/netflix/input/part-ag:0+276353511	20-Jan-2009 08:01:58	20-Jan-2009 08:08:34 (6mins, 36sec)		8	task_200901200759_0001_m_000003	100.00%	s3n://anhi-test-data/netflix/input/part-af:0+276195815	20-Jan-2009 08:01:58	20-Jan-2009 08:08:33 (6mins, 35sec)		8	task_200901200759_0001_m_000004	100.00%	s3n://anhi-test-data/netflix/input/part-ab:0+276051944	20-Jan-2009 08:08:10	20-Jan-2009 08:14:58 (6mins, 47sec)		8
Task	Complete	Status	Start Time	Finish Time	Errors	Counters																																					
task_200901200759_0001_m_000000	100.00%	s3n://anhi-test-data/netflix/input/part-ae:0+286875349	20-Jan-2009 08:01:57	20-Jan-2009 08:09:18 (7mins, 21sec)		8																																					
task_200901200759_0001_m_000001	100.00%	s3n://anhi-test-data/netflix/input/part-aa:0+279753329	20-Jan-2009 08:01:57	20-Jan-2009 08:08:10 (6mins, 13sec)		8																																					
task_200901200759_0001_m_000002	100.00%	s3n://anhi-test-data/netflix/input/part-ag:0+276353511	20-Jan-2009 08:01:58	20-Jan-2009 08:08:34 (6mins, 36sec)		8																																					
task_200901200759_0001_m_000003	100.00%	s3n://anhi-test-data/netflix/input/part-af:0+276195815	20-Jan-2009 08:01:58	20-Jan-2009 08:08:33 (6mins, 35sec)		8																																					
task_200901200759_0001_m_000004	100.00%	s3n://anhi-test-data/netflix/input/part-ab:0+276051944	20-Jan-2009 08:08:10	20-Jan-2009 08:14:58 (6mins, 47sec)		8																																					
Display task counters	<p>Click on an entry in the Counters column. Hadoop displays the task counter information.</p> <div><h3>Counters for task_200901200759_0001_m_000000</h3><table><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><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></div>	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	<div>Click a task. Hadoop displays task information.</div> <div>Job job_200901200759_0001</div> <div>All Task Attempts</div> <table><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_0</td><td>/default-rack/omU-12-31-39-00-A1-B6.compute-1.internal</td><td>SUCCEEDED</td><td><div><div>100.00%</div></div></td><td>20-Jan-2009 08:01:58</td><td>20-Jan-2009 08:09:18 (7mins, 20sec)</td><td></td><td>Last 4KB Last 8KB All</td><td>8</td><td></td></tr></tbody></table> <div>Input Split Locations</div> <div>/default-rack/localhost</div>	Task Attempts	Machine	Status	Progress	Start Time	Finish Time	Errors	Task Logs	Counters	Actions	attempt_200901200759_0001_m_000000_0	/default-rack/omU-12-31-39-00-A1-B6.compute-1.internal	SUCCEEDED	<div><div>100.00%</div></div>	20-Jan-2009 08:01:58	20-Jan-2009 08:09:18 (7mins, 20sec)		Last 4KB Last 8KB All	8	
Task Attempts	Machine	Status	Progress	Start Time	Finish Time	Errors	Task Logs	Counters	Actions												
attempt_200901200759_0001_m_000000_0	/default-rack/omU-12-31-39-00-A1-B6.compute-1.internal	SUCCEEDED	<div><div>100.00%</div></div>	20-Jan-2009 08:01:58	20-Jan-2009 08:09:18 (7mins, 20sec)		Last 4KB Last 8KB All	8													

4. On the **All Task Attempts** pane, choose one of the following actions:

To...	Do This...
Get information about the EC2 host that ran the task	<p>Click an entry in the Machine column. Hadoop displays host information.</p> <p>tracker_domU-12-31-39-00-A1-B6.compute-1.internal:localhost/127.0.0.1:43727 Task Tracker Status</p>  <p>Version: 0.18.1 Compiled: Mon Jan 5 21:29:34 UTC 2009 by root</p> <p>Running tasks</p> <div>Task Attempts Status Progress Errors</div> <p>Non-Running Tasks</p> <div>Task Attempts Status</div> <p>Tasks from Running Jobs</p> <div>Task Attempts Status Progress Errors</div> <p>Local Logs</p> <p>Log directory</p>

To...	Do This...
See the task logs	<p>Click an entry in the Task Logs column. Hadoop displays the logs.</p> <pre> Task Logs: 'attempt_200901200759_0001_m_000000_0' stdout logs stderr logs rvslog logs kvend = 491505; length = 491520 2009-01-20 08:06:24,907 INFO org.apache.hadoop.mapred.MapTask: Index: (0, 10770302, 10770302) 2009-01-20 08:06:24,907 INFO org.apache.hadoop.mapred.MapTask: Finished spill 19 2009-01-20 08:06:29,893 INFO org.apache.hadoop.mapred.MapTask: Spilling map output: buffer full = false and record full = true 2009-01-20 08:06:29,893 INFO org.apache.hadoop.mapred.MapTask: bufstart = 62118070; bufend = 74945608; bufvoid = 149422080 2009-01-20 08:06:29,893 INFO org.apache.hadoop.mapred.MapTask: kvstart = 491505; kvend = 393200; length = 491520 2009-01-20 08:06:40,162 INFO org.apache.hadoop.mapred.MapTask: Index: (0, 13613970, 13613970) 2009-01-20 08:06:40,162 INFO org.apache.hadoop.mapred.MapTask: Finished spill 20 2009-01-20 08:06:48,787 INFO org.apache.hadoop.mapred.MapTask: Spilling map output: buffer full = false and record full = true 2009-01-20 08:06:48,787 INFO org.apache.hadoop.mapred.MapTask: bufstart = 74945608; bufend = 85736781; bufvoid = 149422080 2009-01-20 08:06:48,787 INFO org.apache.hadoop.mapred.MapTask: kvstart = 393200; kvend = 294895; length = 491520 2009-01-20 08:06:52,909 INFO org.apache.hadoop.mapred.MapTask: Index: (0, 11577605, 11577605) 2009-01-20 08:06:52,909 INFO org.apache.hadoop.mapred.MapTask: Finished spill 21 2009-01-20 08:06:59,703 INFO org.apache.hadoop.mapred.MapTask: Spilling map output: buffer full = false and record full = true 2009-01-20 08:06:59,703 INFO org.apache.hadoop.mapred.MapTask: bufstart = 85736781; bufend = 97196611; bufvoid = 149422080 2009-01-20 08:06:59,703 INFO org.apache.hadoop.mapred.MapTask: kvstart = 294895; kvend = 196590; length = 491520 2009-01-20 08:07:02,870 INFO org.apache.hadoop.mapred.MapTask: Index: (0, 12248262, 12248262) 2009-01-20 08:07:02,870 INFO org.apache.hadoop.mapred.MapTask: Finished spill 22 2009-01-20 08:07:08,509 INFO org.apache.hadoop.mapred.MapTask: Spilling map output: buffer full = false and record full = true 2009-01-20 08:07:08,509 INFO org.apache.hadoop.mapred.MapTask: bufstart = 97196611; bufend = 105319918; bufvoid = 149422080 2009-01-20 08:07:08,509 INFO org.apache.hadoop.mapred.MapTask: kvstart = 196590; kvend = 98285; length = 491520 2009-01-20 08:07:15,837 INFO org.apache.hadoop.mapred.MapTask: Index: (0, 8907739, 8907739) 2009-01-20 08:07:15,837 INFO org.apache.hadoop.mapred.MapTask: Finished spill 23 2009-01-20 08:07:23,732 INFO org.apache.hadoop.mapred.MapTask: Spilling map output: buffer full = false and record full = true 2009-01-20 08:07:23,732 INFO org.apache.hadoop.mapred.MapTask: bufstart = 105319918; bufend = 115759223; bufvoid = 149422080 2009-01-20 08:07:23,732 INFO org.apache.hadoop.mapred.MapTask: kvstart = 98285; kvend = 491501; length = 491520 2009-01-20 08:07:28,240 INFO org.apache.hadoop.mapred.MapTask: Index: (0, 11225839, 11225839) </pre>

Troubleshooting

Topics

- [How to Debug Job Flows Starting With No Steps \(p. 76\)](#)
- [How to Debug Job Flows That Have Steps \(p. 77\)](#)
- [How to Debug Job Flows Using Log Files \(p. 77\)](#)

Because your Amazon Elastic MapReduce job flow generally entails processing a great amount of data, we recommend that you rigorously test your mapper and reducer executables locally on a small but representative amount of your data before uploading your jars to Amazon S3. When you do upload your jars, we recommend that you use one of the following debugging procedures for the first processing run.

There are two approaches for developing and troubleshooting your steps:

- Start a job flow that has no steps and add steps incrementally testing each time to make sure the newest step completes successfully
- Start with a job flow that has steps, which you revise each time a step completes with an error

This section describes both approaches.

How to Debug Job Flows Starting With No Steps

Elastic MapReduce allows you to create a job flow that has no steps in it. The effect is to create an EC2 cluster and stop. You can then add steps individually using `AddJobFlowSteps`. As soon as you issue that request, Elastic MapReduce runs the job flow and you can see whether or not the step completed successfully.

To develop and debug a job flow starting without steps

1. Create a jar file that does not include steps.
2. In a `RunJobFlow` request, set `KeepJobFlowAliveWhenNoSteps` to `true` and `ActionOnFailure` to `CANCEL_AND_WAIT`.
`CANCEL_AND_WAIT` stops job flow execution but does not terminate the EC2 cluster. The default value, `TERMINATE`, stops the job flow and terminates the cluster. `CANCEL_AND_WAIT` enables you to revise your jars or add steps and retry the job flow without incurring the expense of downloading the data from Amazon S3 to Amazon EC2.
3. Send the `RunJobFlow` request.
4. If you want to see the Hadoop system, SSH as Hadoop user into the master node.

```
ssh -i [keyfile] hadoop@[EC2_master_node_DNS]
```
5. In a `AddJobFlowSteps` request, set `ActionOnFailure` to `CANCEL_AND_WAIT`.
6. Send the `AddJobFlowSteps` request.
7. Inspect the log files using a tool like Amazon S3 Organizer to see if there were errors.

Using this procedure, you can work on a step to make sure it completes successfully before adding the next step. For more information about adding steps, go to [How to Add Steps to a Job Flow \(p. 61\)](#).

When you are ready for production, set `KeepJobFlowAliveWhenNoSteps` to `false` and `ActionOnFailure` to `TERMINATE_JOB_FLOW`.

This value automatically terminates the EC2 cluster after running the job flow.



Note

When you use the console to run a job flow, the value of `ActionOnFailure` is always `CONTINUE`.

How to Debug Job Flows That Have Steps

You might have a job flow that already has steps.

To develop and debug a job flow starting with steps

1. In a `RunJobFlow` request, set `ActionOnFailure` to `CANCEL_AND_WAIT`.
This value stops job flow execution but does not terminate the Amazon EC2 cluster. The default value, `TERMINATE`, stops the job flow and terminates the cluster. `CANCEL_AND_WAIT` enables you to revise your JARs or add steps and retry the job flow without incurring the expense of downloading the data from Amazon S3 to EC2.
2. Send the `RunJobFlow` request.
3. Inspect the log files using a tool like Amazon S3 Organizer to see if there were errors.
4. Change the step that caused the error and resubmit the step using `AddJobFlowStep` setting, in the request, `ActionOnFailure` to `CANCEL_AND_WAIT`.

How to Debug Job Flows Using Log Files

Topics

- [How to Check Step Log Files \(p. 78\)](#)
- [Task Attempts \(p. 79\)](#)

- [Checking Hadoop Failures \(p. 79\)](#)

Debugging errors in large, distributed applications can be difficult. Elastic MapReduce makes it easier by collecting the log files from the cluster and storing them in a location you specify on Amazon S3. If you do not specify a log URI in the `RunJobFlow` request, Elastic MapReduce does not collect logs.



Important

In this section, all relative Amazon S3 paths should be prefixed with your log URI and job flow ID to get the actual log locations.

How to Check Step Log Files

If you provided a custom JAR and there is a failure, the first things to check are the step log files. Elastic MapReduce uploads these log files to `steps/<step number>/` every few minutes. Each step creates the following four logs:

- **controller**—Contains files generated by Elastic MapReduce 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.
- **syslog**—Contains logs from non-Apache software, such as Apache and Hadoop
- **stdout**—Contains status generated by your mapper and reducer executables
- **stderr**—Contains your step's standard error messages

To debug a job flow using step log files

1. SSH as the Hadoop user into the master node using the pem file from the master node key pair to find the log files associated with the failed step.

```
$ ssh -i mykey.pem hadoop@ec2-67-202-20-49.compute-1.amazonaws.com
```

2. Use `cat` to view the log files.

The following example looks into the syslog files. You can use the same procedure with any of the other three logs, controller, stdout, and stderr.

```
$ cat /mnt/var/log/hadoop/steps/1/syslog
2009-03-25 18:43:27,145 WARN org.apache.hadoop.mapred.JobClient (main):
Use GenericOptionsParser for parsing
the arguments. Applications should implement Tool for the same.
2009-03-25 18:43:28,828 ERROR org.apache.hadoop.streaming.StreamJob
(main): Error Launching job : unknown
host: examples
$ exit
```

This error from Hadoop indicates that it was trying to look for a host called examples. If we look back at our request we see that the output path was set to `hdfs://examples/output`. This is actually incorrect since we want Hadoop to access the local HDFS system with the path `/examples/output`. We instead need to specify `hdfs:///examples/output`.

3. Specify the output of the streaming job on the command line and submit another step to the job flow.

```
./elastic-mapreduce --jobflow j-36U2JMAE73054 --stream --output hdfs:///
examples/output
```

4. List the job flows to see if it completes.

```
$ ./elastic-mapreduce --list -n 5
```

```
j-36U2JMAE73054      WAITING
ec2-67-202-20-49.compute-1.amazonaws.com      Example job flow
      FAILED          Example Streaming Step
      COMPLETED      Example Streaming Step
```

This time the job succeeded. We can run the job again but this time send the output to a bucket in Amazon S3.

5. Create a bucket in Amazon S3.

Buckets in Amazon S3 are unique so choose a unique name for your bucket. The following example uses `s3cmd`. For more information about creating buckets, see the *AWS Amazon Elastic MapReduce Getting Started Guide*.

```
$ s3cmd mb s3://my-example-bucket
Bucket 's3://my-example-bucket/' created
```

`s3cmd` requires you to specify Amazon S3 paths using the prefix `s3://`. Elastic MapReduce requires the prefix `s3n://` for files stored in Amazon S3.

6. Add a step to the job flow to send output to this bucket.

```
$ ./elastic-mapreduce -j j-36U2JMAE73054 --stream --output s3n://my-
example-bucket/output/1
Added steps to j-36U2JMAE73054
```

The protocol of the output URL is `s3n`. This tells Hadoop to use the Amazon S3 Native File System for the output location. The host part of the URL is the bucket and this is followed by the path.

7. Terminate the job flow.

```
$ ./elastic-mapreduce -j j-36U2JMAE73054 --terminate
```

8. Confirm that the job flow is shutting down.

```
$ ./elastic-mapreduce --list -n 5
```

There are other options that you can specify when creating and adding steps to job flows. Use the `--help` option to find out what they are.

Task Attempts

If your JAR successfully started or you created a streaming job, the next place to look for failures would be in the task attempts. The Map and Reduce functions you wrote execute in the context of a task. Tasks can execute multiple times as "task attempts" because of failures or speculative execution. Elastic MapReduce uploads task attempt logs into `task-attempts/`.

If one of the tasks failed, you can look at the task logs to determine what happened. These files are also available on the EC2 instances under `/mnt/var/log/hadoop/userlogs/`. Looking through log files on each instance in the cluster, however, makes this way of debugging harder.

Task-attempt log files are similar in format to the step log files.

Checking Hadoop Failures

In rare cases, Hadoop itself might fail. To see if that is the case, you must look at the Hadoop daemon logs.

To view the daemon log files

- Look under `/mnt/var/log/hadoop/` on each instance or under `daemons/<instance id>/` on Amazon S3.



Note

Not all EC2 instances run all daemons.

Advanced Topics

Topics

- [Application Tips](#) (p. 81)
- [EC2 Cluster Tuning](#) (p. 81)
- [What We Expect in Your JAR](#) (p. 82)
- [Distributed Cache](#) (p. 82)
- [Hadoop Data Compression](#) (p. 88)
- [hadoop.tmp.dir](#) (p. 88)
- [How to Process Nested Directories of Files](#) (p. 89)
- [How to Use the Ruby Client as a Library](#) (p. 89)

This section describes advanced subject matter related to Hadoop and Elastic MapReduce.

Application Tips

Use the following tips to improve your applications:

- When you set *ActionOnFailure* to CONTINUE, failed jobs will show as "Completed." We assume that by setting the value this way you will handle exceptions yourself. If that is not your intention, set the value to TERMINATE_JOB_FLOW.
- Although Elastic MapReduce can handle very large input and output files, for the best performance the input files should be between 64 MB and 128 MB. The output files can be larger but ideally should not be larger than 500MB each. You control the size of your input data by putting your data in multiple objects, each between 64 MB and 128 MB.

EC2 Cluster Tuning

Elastic MapReduce enables you to specify the number and kind of EC2 instances in the cluster, which are the primary means of affecting the speed with which your job flow completes. There are, however, a host of Hadoop parameter values that govern the operation of those instances at a much finer level of granularity.

Elastic MapReduce by default sets many Hadoop parameters. Some of these parameter values can be overridden by parameter values set in a *RunFlowJob* request. (For more information, see [RunJobFlow](#).) Hadoop parameters govern such things as the number of mapper and reducer tasks

assigned to each node in the cluster, the amount of memory allocated for these tasks, the number of threads, timeouts and other configuration parameters for the various Hadoop components.

Hadoop configuration parameters reside in Hadoop's `JobConf` file. You set the Hadoop configuration parameters by including them in your `JAR` file. For streaming jobs you can specify `JobConf` parameters using the `-jobconf` option. For details, goto http://hadoop.apache.org/core/docs/r0.18.3/mapred_tutorial.html. For a list of Hadoop parameters, go to <http://hadoop.apache.org/core/docs/r0.18.3/hadoop-default.html>.

While some `JobConf` parameters are straight forward to set, other parameters act in concert with related parameters or the entire framework and therefore are more difficult to set. For more information, go to [Job Configuration](#) on the Hadoop web site.

To assist with debugging and performance [tuning](#), Elastic MapReduce keeps a log of the Hadoop settings (from the Hadoop `JobConf`) that were used to execute each job flow. These XML files are under `jobs/` in Amazon S3 or at `/mnt/var/log/hadoop/history/` on the master node.

What We Expect in Your JAR

When you start a Elastic MapReduce job using `RunJobFlow` or `AddJobFlowSteps`, Elastic MapReduce starts Hadoop and then executes your JAR using the `bin/hadoop jar` command. For more information go to http://hadoop.apache.org/core/docs/r0.18.3/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, we assume the step failed. Likewise, when your JAR completes, we assume 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](#).

Distributed Cache

Topics

- [Archives \(p. 83\)](#)
- [Location of Cached Files \(p. 83\)](#)
- [How to Cache Files Using the Console \(p. 84\)](#)
- [How to Cache Files Using the Ruby Command Line Interface \(p. 85\)](#)
- [Accessing Cached Files From Mapper and Reducer Applications \(p. 87\)](#)
- [Streaming Example Using DistributedCache \(p. 87\)](#)

`DistributedCache` is a Hadoop mechanism that copies read-only files, such as text files, applications, and compressed files, onto every slave node in a cluster. The purpose of distributing these files or applications is to make them available for mappers and reducers to use during a Hadoop job. If your Hadoop application depends on applications or binaries that are not installed on Amazon EC2 instances, you can use `DistributedCache` to import these applications into the mapper's and reducer's current working directory. If your Hadoop application or Hadoop-run application requires data that it cannot receive using the Hadoop framework, you can use `DistributedCache` to import this data from Amazon S3 or HDFS into the mapper's and reducer's current working directory on each slave node.

Caching Files and Applications Using `DistributedCache`

1	Create hooks in your custom mapper or reducer application that import the files or call the applications that you intend to cache.
---	--

2	Upload your files, applications (binaries), or compressed files into Amazon S3.
3	Add the files (using Hadoop or Elastic MapReduce command line options) to the DistributedCache when creating a job flow.

You can cache files stored on any Hadoop-compatible file system (hdfs, s3n). The default size of the DistributedCache is 10GB. To change the size, use the Hadoop parameter, *local.cache.size*. Hadoop copies the files once per Hadoop job just before starting the Hadoop job and they remain cached for the duration of the job.



Note

DistributedCache caches files, applications, or compressed files only on the slave nodes and not the master node unless there are no slave nodes in the cluster. In that case, DistributedCache caches files on the master node.

Archives

You can either add to the DistributedCache single files or archives, which are one or more files that may or may not be compressed using a compression utility, such as gzip. DistributedCache passes the compressed files to each slave node (or to the master node if there are no slave nodes) and decompresses them there. DistributedCache supports the following compression formats:

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

DistributedCache decompresses the archives before starting the Hadoop job so that the files are available to the mapper and reducer.

Location of Cached Files

Hadoop makes the cached files available at the current working directory of the mapper and reducer through symlinks. The value of the Hadoop parameter, *mapred.local.dir*, specifies where Hadoop saves temporary files. Amazon Elastic MapReduce sets this parameter to `/mnt/var/lib/hadoop/mapred/`. Hadoop caches files in a subdirectory of that directory: `/mnt/var/lib/hadoop/mapred/taskTracker/archive/`. If you cache a single file, DistributedCache caches the file in the `archive` directory and gives it the filename of the single file. If you cache an archive of compressed files, DistributedCache decompresses the archive, creates a subdirectory in `/archive` using the name of the archive as the name of the directory, and caches the uncompressed files in that subdirectory. The unarchived files have the same name as their archived version.

Making Cached Files Available to Mapper and Reducer Applications

1	Hadoop copies the simple or archive files from Amazon S3 (or HDFS) to each of the slave nodes.
2	If the file is not compressed, Hadoop saves the file in <code>/mnt/var/lib/hadoop/mapred/taskTracker/archive/</code> on each slave instance. If the file is compressed, Hadoop decompresses the file in <code>/mnt/var/lib/hadoop/mapred/taskTracker/archive/[archiveName]/</code> on each slave instance, where <i>archiveName</i> is the name of the archive file (without the file extension).

3	Hadoop creates symlinks from the cached entities in <code>/mnt/var/lib/hadoop/mapred/taskTracker/archive/</code> to the current working directory of the mapper and reducer.
4	The mapper and reducer use the cached files as programmed.

How to Cache Files Using the Console

You can use the AWS Management Console to create job flows that use the DistributedCache. To cache a single file, you use the Hadoop parameter `-cacheFile` in the **Extra Args** field on the **Specify Parameters** page in the AWS Management Console. To cache a compressed file, you use `-cacheArchive`.

The screenshot shows the 'Create a New Job Flow' wizard in the AWS Management Console. The 'Specify Parameters' step is active. Under the 'Execute a Pig Script' section, the 'Extra Args' field is highlighted with a red circle. The text in the field is `-cacheFile s3n://bucket/sample_bin#sample_bin`. Other fields like 'Script Location', 'Input Location', and 'Output Location' are also visible but not highlighted.

For more information about using the console to create a job flow, see [How to Create a Job Flow Using Streaming and JAR \(p. 23\)](#).



Important

You can only use `-cacheFile` and `-cacheArchive` when creating Streaming job flows.

To add single files to the DistributedCache using the console

- In the creation of a job flow, on the **Specify Parameters** page, in the **Extra Args** text field, enter `-cacheFile` followed by the location of the files in Amazon S3, the pound (#) sign, and then the name you want to give the file when it's placed in the working directory of the mapper and reducer. For example, in the **Extra Args** text field you might add:

```
-cacheFile s3n://my_bucket/sample_binary.bin#sample_binary_cached.bin
```

In this example, the Hadoop takes the file, `sample_binary.bin`, located in `s3n://my_bucket/sample_binary.bin` and places the file in the current working directory of the mapper and reducer under the name `sample_binary_cached.bin`.

To add archive files to the DistributedCache using the console

- In the creation of a job flow, on the **Specify Parameters** page, in the **Extra Args** text field, enter `-cacheArchive` followed by the location of the files in Amazon S3, the pound (#) sign, and then the name you want to give the directory when it's placed in the working directory of the mapper and reducer. For example, in the **Extra Args** text field you might add:

```
-cacheArchive s3n://my_bucket/sample_dataset.tgz#sample_dataset_cached
```

In this example, the Hadoop takes the archive, `sample_dataset.tgz`, located in `s3n://my_bucket/sample_dataset.tgz`, creates a directory, `sample_dataset_cached`, in the current working directory of the mapper and reducer, and decompresses the archive files to `sample_dataset_cached`.

Instead of adding just one file or one archive file to the DistributedCache, you can use multiple `-cacheFile` and `-cacheArchive` options to add multiple files.

To add multiple files to the DistributedCache using the console

- In the creation of a job flow, on the **Specify Parameters** page, in the **Extra Args** text field, enter either `-cacheFile` if you're caching a single file or `-cacheArchive` if you're caching compressed files, followed by the location of the files in Amazon S3, the pound (#) sign, and then the name you want to give the file (if it's a file) or directory (if it's an archive) when it's placed in the working directory of the mapper and reducer. For example, in the **Extra Args** text field you might add:

```
-cacheFile s3n://my_bucket/sample_binaryA.bin#sample_binaryA_cached.bin
-cacheFile s3n://my_bucket/sample_binaryB.bin#sample_binaryB_cached.bin
-cacheArchive s3n://my_bucket/sample_datasetA.tgz#sample_datasetA_cached
-cacheArchive s3n://my_bucket/sample_datasetB.tgz#sample_datasetB_cached
```

In this example, the Hadoop takes the files, `sample_binaryA.bin` and `sample_binaryB.bin` and creates symlinks to `/mnt/var/lib/hadoop/mapred/taskTracker/archive/sample_binaryA_cached.bin` and `/mnt/var/lib/hadoop/mapred/taskTracker/archive/sample_binaryB_cached.bin`, respectively. Also, Hadoop takes the archives, `sample_datasetA.tgz` and `sample_datasetB.tgz`, creates directories under `/archive` named `sample_datasetA_cached` and `sample_datasetB_cached`, respectively, for the unarchived files. The unarchived files have the same name as they were in the archived version.

How to Cache Files Using the Ruby Command Line Interface

To add files or archives to the DistributedCache using the Ruby Command Line Interface (RubyCLI), you add the RubyCLI options `--cache` or `--cache-archive` on the RubyCLI command line. These options are the equivalent of the Hadoop options `-cacheFile` and `-cacheArchive`. `--cache` adds a single file to the DistributedCache. `--cache-archive` adds an archive of files to the DistributedCache. In both cases, you follow the option by the location of the files in Amazon S3 (or HDFS), the pound (#) sign, and then the name you want to give the file or directory (for archives) when placed in the working directory of the mapper and reducer.



Note

You use double dashes (--) in the RubyCLI and single dashes (-) in the console.



Important

You can only use `--cache` and `--cache-archive` in Streaming job flows.

To add a single file to the DistributedCache using the CLI

- On the RubyCLI command line, add the `--cache` option (to a `--create` command) followed by the location of the file in S3N or HDFS and the name of the file as you want it to be known in the current working directory of the mapper and reducer. For example:

```
--cache s3n://my_bucket/sample_dataset.dat#sample_dataset_cached.dat
```

In this example, the file `sample_dataset.dat`, which resides in `s3n://my_bucket/sample_dataset.dat` is symlinked into the current working directory of the mapper and reducer as a file named `sample_dataset_cached.dat`.

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

```
./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
```

`--cache-archive` adds one or more files to the DistributedCache. The syntax of the `--cache-archive` option is the same as it is for `--cache`. Unlike `--cache`, however, `--cache-archive` checks the extension of the file to see if it is a compressed file. If the file is compressed, DistributedCache creates a directory named the same as the compressed file, then decompresses the files in that directory, which are available to the working directory of the mapper and reducer.

To add an archive file to the DistributedCache using the CLI

- On the RubyCLI command line, add the `--cache-archive` option followed by the location of the archive file in S3N or HDFS and the name of the directory to unarchive the files as you want it to be known in the current working directory of the mapper and reducer. For example:

```
--cache-archive s3n://my_bucket/sample_dataset.tgz#sample_dataset_cached
```

In this example, the file `sample_dataset.tgz`, which resides in `s3n://my_bucket/sample_dataset.tgz` in the current working directory of the mapper and reducer as a file named `sample_dataset_cached`.

The following command shows the creation of a job flow and uses `--cache-archive` to add an archive, `sample_dataset.tgz`, to the DistributedCache.

```
./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
```

Instead of adding just one file or one archive file to the DistributedCache, you can use multiple `--cache` and `--cache-archive` options to add multiple files.

To add multiple files to the DistributedCache using the CLI

- In the RubyCLI, use the `--cache` or `--cache-archive` options for each file or archive to cache followed by its location in Amazon S3 (or HDFS), a pound (#) sign, and the name it will assume in the current working directory of the mapper and reducer. Use `--cache` to cache single files and `--cache-archive` to cache compressed files.

```
--cache s3n://my_bucket/sample_binary.bin#sample_binary_cached.bin
--cache s3n://my_other_bucket/sample_binary1.bin#sample_binary1_cached.bin
--cache-archive s3n://my_bucket/sample_dataset.tgz#sample_dataset_cached
```

The following command shows the creation of a job flow and adds multiple files to the DistributedCache.

```
./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
--cacheFile s3n://my_bucket/sample_binaryA.bin#sample_binaryA_cached.bin
--cacheFile s3n://my_bucket/sample_binaryB.bin#sample_binaryB_cached.bin
--cacheArchive s3n://my_bucket/sample_datasetA.tgz#sample_datasetA_cached
--cacheArchive s3n://my_bucket/sample_datasetB.tgz#sample_datasetB_cached
```

In this example, the Hadoop takes the files, `sample_binaryA.bin` and `sample_binaryB.bin` and creates `/mnt/var/lib/hadoop/mapred/taskTracker/archive/sample_binaryA_cached.bin` and `/mnt/var/lib/hadoop/mapred/taskTracker/archive/sample_binaryB_cached.bin`, respectively. Also, Hadoop takes the archives, `sample_datasetA.tgz` and `sample_datasetB.tgz`, creates directories under `/archive` named `sample_datasetA_cached` and `sample_datasetB_cached`, respectively, and the decompressed archive files.

Accessing 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 [DistributedCache](#).

Streaming Example Using DistributedCache

The following example of a mapper written in Python counts words in files and uses a list of words ignored in the counting. The list of ignored words resides in a file in the DistributedCache. Elastic MapReduce loads the file from S3 into the DistributedCache prior to running the Hadoop job.

```
#!/usr/bin/python

from __future__ import with_statement
import sys
import re

def main(argv):
    stop_word_list = set()
    with open("stop-word-list", "r") as f:
        for line in f:
            stop_word_list.add(line.rstrip())
    f.close()
    line = sys.stdin.readline()
    pattern = re.compile("[a-zA-Z][a-zA-Z0-9]*")
    try:
        while line:
            for word in pattern.findall(line):
                if word.lower() in stop_word_list:
                    continue
                print "LongValueSum:" + word.lower() + "\t" + "1"
                line = sys.stdin.readline()
            except "end of file":
                return None
    if __name__ == "__main__":
        main(sys.argv)
```

This script expects to find the file `stop-word-list` in its current working directory. To execute the mapper, you need to add the file to the DistributedCache, which you do in the console using `-cacheFile` in **ExtraArgs** or `--cache` from the RubyCLI

To run this job flow

1. Save the Python script to a bucket on Amazon S3.
Name it, for example, `wordSplitter.py`.
2. Save the file, `stop-word-list`, to a bucket on Amazon S3.
3. Use the following command line command to run the mapper, which uses `stop-word-list` in the DistributedCache.

```
./elastic-mapreduce --create --stream
--input s3n://your_bucket/wordcount/input
--output s3n://your_bucket/wordcount/output
--mapper s3n://your_bucket/wordcount/wordSplitter.py
--reducer aggregate
--cache s3n://your_bucket/wordcount/stop-word-list#stop-word-list
```



Note

This mapper requires Python 2.5.2 or above to run.

Hadoop Data Compression

Elastic MapReduce supports operating on compressed files using gzip and zlib. To save space in Amazon S3, you can compress your input files before uploading them. Hadoop decompresses the files as they are read and feeds them to your mapper applications. However, files compressed with gzip or zlib cannot be split into smaller chunks for processing. If you use these, you might not benefit from Hadoop's fine-grained load balancing. When using compressed data as your input, we recommend that you have at least one file per core. The file extension must be `.gz` or `.deflate` for Hadoop to recognize the files as compressed.

You can also compress your output data. We default to gzip when output compression is enabled. To enable compression while writing a custom JAR, follow the instructions in the Hadoop documentation. For more information, go to the [Hadoop documentation](#).

To enable output data compression using the console and a streaming job

- In the console's **Extra Args** field, enter the following:

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

How to Process Gzipped Files

Hadoop automatically detects the `.gz` extension on file names and extracts the contents. You do not need to take any action to extract gzipped files.

hadoop.tmp.dir

In `hadoop-site.xml`, we set `hadoop.tmp.dir` to `/mnt/var/lib/hadoop/tmp`. `/mnt` is where we mount the “extra” EC2 volumes, which can contain a lot more data than the default volume. (The exact amount depends on instance type.) Hadoop's `RunJar.java` (the module that unpacks the input JARs) interprets `hadoop.tmp.dir` as a Hadoop file system path rather than a local path, so it writes to the path in HDFS instead of a local path. HDFS is mounted under `/mnt` (specifically `/mnt/var/lib/hadoop/dfs/`). So, you can write lots of data to it.

/home/hadoop is the home directory for the “hadoop” user (which is what your jobs run as), so you can create directories there. This directory, however is on the small, default volume. For that reason, you are better off using /mnt/var/lib/hadoop as your base.

How to Process Nested Directories of Files

Hadoop supports global variables to define paths. Using global variables enables you to search sub directories. You can use the following special characters to create global variable path names:

- ?—matches any single character
- *—matches 0 or more characters (it does not recurse)
- {}—matches any of the comma-separated values between the braces, for example {ab,cd} matches ab or cd.

For example, to process all of the subdirectories of 20090519, specify the input file as `s3n://my-bucket/prod/logs/httpd/20090519/*`. Or, to process `http_log.gz`, which might occur in multiple directories, specify the input file as `s3n://my-bucket/prod/logs/httpd/20090519/*/http_log.gz`.

How to Use the Ruby Client as a Library

The program, *elastic-mapreduce*, is an example of how to use the Elastic MapReduce Ruby library. Almost all of the file is devoted to parsing command line arguments and translating them into web service calls.

To use the web service directly in your ruby programs

1. Use the following code:

```
config = {
  :endpoint          => "https://elasticmapreduce.amazonaws.com",
  :ca_file           => File.join(File.dirname(__FILE__), "cacert.pem"),
  :aws_access_key    => [my_access_id],
  :aws_secret_key    => [my_secret_key],
  :signature_algorithm => :V2
}
client = Amazon::Coral::ElasticMapReduceClient.new_aws_query(config)

puts client.DescribeJobFlows.inspect
puts client.DescribeJobFlows('JobFlowId' => 'j-ABAYAS1019012').inspect
```

2. Replace for *my_access_id* and *my_secret_key* your AWS Access Key ID and your AWS Secret Key ID, respectively.
3. If there is a connection failure, use the retry delegator to make your client retry.

```
$LOAD_PATH << File.dirname(__FILE__)
require 'amazon/coral/elasticmapreduceclient'
require 'amazon/retry_delegator'

config = {
  :endpoint          => "https://elasticmapreduce.amazonaws.com",
  :ca_file           => File.join(File.dirname(__FILE__),
  "cacert.pem"),
```

```
:aws_access_key      => my_access_id,
:aws_secret_key      => my_secret_key,
:signature_algorithm => :V2
}
client = Amazon::Coral::ElasticMapReduceClient.new_aws_query(config)

is_retryable_error_response = Proc.new do |response|
  if response == nil then
    false
  else
    ret = false
    if response['Error'] then
      ret ||= ['InternalFailure', 'Throttling', 'ServiceUnavailable',
'Timeout'].include?(response['Error']['Code'])
    end
    ret
  end
end

client = Amazon::RetryDelegator.new(client, :retry_if =>
is_retryable_error_response)

puts client.DescribeJobFlows.inspect
puts client.DescribeJobFlows('JobFlowId' => 'j-ABAYAS1019012').inspect
```

Glossary

AMI	<p>Amazon Machine Instance</p> <p>An AMI is the equivalent of one computer in EC2. AMIs come in a variety of configurations based on the number of processors and cache size. Elastic MapReduce automatically determines the right number of AMIs and the right type of AMIs for a job flow. You can, however, use the API to override that configuration. For more information, see the <i>Amazon Elastic Compute Cloud Developer Guide</i>.</p>
authentication	<p>The process of proving your identity to the system.</p>
AWS Access Key ID	<p>A string distributed by AWS that uniquely identifies an AWS developer.</p> <p>The value of this ID is included in every Amazon Elastic MapReduce request.</p>
block	<p>A data set.</p> <p>Amazon Elastic MapReduce breaks large amounts of data into subsets. Each subset is called a data block. Amazon Elastic MapReduce assigns an ID to each block and uses a hash table to keep track of block processing.</p>
bucket	<p>A container for objects stored in Amazon Elastic MapReduce. Every object is contained within a bucket. For example, if the object named <code>photos/puppy.jpg</code> is stored in the <code>johnsmith</code> bucket, then it is addressable using the URL <code>http://johnsmith/S3.amazonaws.com/photos/puppy.jpg</code></p>
endpoint	<p>The URI for Amazon Elastic MapReduce web service.</p> <p>The endpoint for all Amazon Elastic MapReduce requests is <code>https://elasticmapreduce.amazonaws.com</code>.</p>
HMAC	<p>Hash Message Authentication Code that is used to authenticate a message</p> <p>The HMAC is calculated using a standard, hash cryptographic algorithm, such as SHA-1. This algorithm uses a key value to perform the encryption. In Amazon Elastic MapReduce, that key is your Secret Key. For that reason, your Secret Key must remain a shared secret between you and Elastic MapReduce.</p>

instance	Once an AMI has been launched, the resulting running system is referred to as an instance. All instances based on the same AMI start out identical and any information on them is lost when the instances are terminated or fail.
job flow	Comprised of one or more steps, which specify all of the functions to be performed on the data. A job flow specifies the complete processing of the data.
key	The unique identifier for an object within a bucket. Every object in a bucket has exactly one key. Since a bucket and key together uniquely identify each object, Amazon Elastic MapReduce can be thought of as a basic data map between "bucket + key" and the object itself. Every object in Amazon Elastic MapReduce can be uniquely addressed through the combination of the Service endpoint, bucket name, and key.
mapper	An executable that splits the raw data into key/value pairs The output of the mapper, called <i>intermediate results</i> , is used by the reducer as its input.
master node	Keeps track of processing. The master node is a process running on an AMI that keeps track of work done by slave nodes. Each slave node represents the work done by one AMI, which is processing a block of data. The master node polls the slave nodes for status, including the completion of the processing. When a slave node completes its processing, the master node terminates the AMI.
metadata	The metadata is a set of name-value pairs that describe the object. These include default metadata such as the date last modified and standard HTTP metadata such as Content-Type. The developer can also specify custom metadata at the time the Object is stored.
object	The fundamental entity stored in Amazon S3. Objects consist of object data and metadata. The data portion is opaque to Amazon S3.
reducer	An executable that uses the intermediate results produced by the mapper and processes them into the final output
Secret Access Key	A string distributed by AWS that uniquely identifies an AWS developer. The Secret Access Key is a shared secret between a developer and AWS. The Secret Key is used as the key in the HMAC algorithm that encrypts the signature.
service endpoint	Same as endpoint.
signature	A URL-encoded string composed of request parameters and their values encrypted using an HMAC algorithm. Signatures are used to authenticate and safeguard requests.
slave node	Represents an AMI that is processing a data block Amazon Elastic MapReduce loads data blocks across multiple AMIs so the data can be processed in parallel. Each AMI that processes a data block is called a slave node. Slave nodes are polled by Master nodes for status. See master node.
step	A single function applied to the data in a job flow.

	<p>A step is a MapReduce algorithm implemented as a Java JAR or a Hadoop streaming program written in Python, Ruby, Perl or C++. Some steps have to be applied sequentially. The sum of all steps comprises a job flow.</p>
step type	<p>The type of work done in a step.</p> <p>There are a limited number of step types, including moving data from S3 to EC2 (Distcp) and moving data from EC2 to Amazon S3.</p>
streaming	<p>A utility that comes with Hadoop that enables you to develop MapReduce executables in languages other than Java</p>
tuning	<p>Selecting the number and type of AMIs to run the Hadoop job flow most efficiently.</p>

Document Conventions

This section lists the common typographical and symbol use conventions for AWS technical publications.

Typographical Conventions

This section describes common typographical use conventions.

Convention	Description/Example
Call-outs	<p>A call-out is a number in the body text to give you a visual reference. The reference point is for further discussion elsewhere.</p> <p>You can use this resource regularly. 1</p>
Code in text	<p>Inline code samples (including XML) and commands are identified with a special font.</p> <p>You can use the command <code>java -version</code>.</p>
Code blocks	<p>Blocks of sample code are set apart from the body and marked accordingly.</p> <pre># ls -l /var/www/html/index.html -rw-rw-r-- 1 root root 1872 Jun 21 09:33 /var/www/html/ index.html # date Wed Jun 21 09:33:42 EDT 2006</pre>
Emphasis	<p>Unusual or important words and phrases are marked with a special font.</p> <p>You <i>must</i> sign up for an account before you can use the service.</p>
Internal cross references	<p>References to a section in the same document are marked.</p> <p>See Document Conventions (p. 94).</p>
Logical values, constants, and regular expressions, abstracta	<p>A special font is used for expressions that are important to identify, but are not code.</p> <p>If the value is <code>null</code>, the returned response will be <code>false</code>.</p>

Convention	Description/Example
Product and feature names	Named AWS products and features are identified on first use. Create an <i>Amazon Machine Image</i> (AMI).
Operations	In-text references to operations. Use the <code>GetHITResponse</code> operation.
Parameters	In-text references to parameters. The operation accepts the parameter <i>AccountID</i> .
Response elements	In-text references to responses. A container for one <code>CollectionParent</code> and one or more <code>CollectionItems</code> .
Technical publication references	References to other AWS publications. If the reference is hyperlinked, it is also underscored. For detailed conceptual information, see the <i>Amazon Mechanical Turk Developer Guide</i> .
User entered values	A special font marks text that the user types. At the password prompt, type MyPassword .
User interface controls and labels	Denotes named items on the UI for easy identification. On the File menu, click Properties .
Variables	When you see this style, you must change the value of the content when you copy the text of a sample to a command line. % ec2-register <i><your-s3-bucket></i> /image.manifest See also the following symbol convention.

Symbol Conventions

This section describes the common use of symbols.

Convention	Symbol	Description/Example
Mutually exclusive parameters	(Parentheses and vertical bars)	Within a code description, bar separators denote options from which one must be chosen. % data = hdfread (start stride edge)
Optional parameters XML variable text	[square brackets]	Within a code description, square brackets denote completely optional commands or parameters. % sed [-n, -quiet] Use square brackets in XML examples to differentiate them from tags. <CustomerId>[ID]</CustomerId>
Variables	<arrow brackets>	Within a code sample, arrow brackets denote a variable that must be replaced with a valid value. % ec2-register <your-s3-bucket>/image.manifest

Index

gzip, 88
log files, nested, 89

A

add step to job flow
 using command line, 53
 using the API, 61
additional files, 40
additional libraries, 40, 62
Amazon EC2, 18
Amazon EC2 instance types, 19
Amazon Machine Image, 19
Amazon S3 buckets, 13
AMI, 19
architectural overview, 9
Args, 61
audience, 1
availability zones, 60

B

best practices, 81
buckets, 13
business model, 6

C

C#, 58
client, 43
cluster automation, 8
cluster tuning, 81
command
 add steps, 53
 all, 46
 create job flow, 47
 details, 53
 list job flows, 52
 terminate job flow, 54
command line
 as library, 89
 downloading, 44
 interface, 43
compression
 data, 88
concepts, 10
console, 21
console, create job flow JAR, 23
console, create job flow, Hive, 36
console, create job flow, Pig, 32
console, create job flow, streaming, 27
create a job flow
 using the API, 58
 using the command line, 47
create job flow, 23
create job flow, Hive, 36

create job flow, JAR
 using the console, 23
create job flow, Pig, 32
create job flow, streaming, 27
create requests, 55

D

data compression, 88
Debian, 19
debug, 77
debug using log files, 78
describe job flow, 62
download log files, 71

E

EC2, 18
 cluster tuning, 81
 instances, 60

F

failures, 79
Features, 7
file systems, 13
files, additional, 40
FoxyProxy, 68
FoxyProxy, configuring, 69

G

GNU, 19
guide organization, 2

H

Hadoop, 11, 19
 data compression, 88
 failures, 79
 location of, 20
 user interface, 71
hadoop.tmp.dir, 88
HDFS, 13
Hive, 14, 36

I

information about job flow, 53
installed software on nodes, 19
instance types, 19
instances, 60
interfaces: comparison, 9

J

jar, 7
JAR, 14, 82
 steps, 60
Java, 19, 58
job flow
 add steps
 using API, 61

- using command line, 53
- create
 - using command line, 47
 - using the API, 58
- debug, 76
 - job flow with steps, 77
 - job flow without steps, 76
- describe using API, 62
- details using command, 53
- download logs from Amazon S3, 71
- information, 40
- list, 52
- monitoring, 65, 66
- states, 63
- status using SSH, 67
- terminate
 - using API, 64
 - using command line, 54
 - using the console, 41
- job flow, create, streaming, 27
- job flow, creating, 23
- job flow, creating, Hive, 36, 49
- job flow, creating, JAR, 23
- job flow, creating, Pig, 32, 47

K

- key-pair, 19
 - creating a key-pair, 21

L

- libraries, additional, 40, 62
- library, code, 58
- Linux, 19
- list job flows, 52
- log files, 65, 88
 - directories, 65
 - download from Amazon S3, 71
 - step, 78
 - used to debug, 78

M

- MapReduce, 11
- MapReduce process, 12
- master node, 12
- monitor job flow, 67
- monitoring
 - job flows, 65, 66
- multiple steps, 60

N

- nested log files, 89
- new features, 4
- node, 12

O

- operating system, 19

- organization of guide, 2
- overview, 6

P

- Perl, 19, 58
- PHP, 19, 58
- Pig, 14
- pricing, 7
- processing speed, 8
- Python, 19

Q

- Query, 55

R

- reader feedback link, 2
- related resources, 2
- reliability, 8
- request, 55
- required knowledge, 1
- Ruby, 19, 43
- run job flow, 58

S

- S3N, 13
- security, 8
- signature, 56
- slave node, 12
- software
 - Hadoop, 19
 - installed on nodes, 19
 - Java, 19
 - Perl, 19
 - PHP, 19
 - Python, 19
 - Ruby, 19
- speed, 8
- SSH, 8, 66, 67, 68
 - interactive mode, 15, 17
 - into master node, 12
 - with Hive, 49
 - with Pig, 47
- start job flow, 58
- state
 - of job flow, 63
 - step, 63
- step, 60
 - add to job flow
 - using the API, 61
 - using the command line, 53
 - log files, 78
 - states, 63
- storage, 8
- streaming, 7, 14
- system dependencies, 44

T

- task, 75, 79
- terminate job flow, 41, 64
 - command, 54
- tips, 81, 81
- troubleshooting, 76

U

- updates, 4
- user interface
 - Hadoop, 71

V

- VB.NET, 58
- version, 4

W

- what's new, 4
- work flow, 14

Z

- zones, availability, 60