

Introduction to Amazon Web Services
and MapReduce Jobs

CS341 Spring 2013

Sébastien Robaszkiewicz

April 4, 2013

Contents

1	Introduction	3
1.1	What for?	3
1.1.1	Hadoop Programming on Amazon Elastic MapReduce . . .	3
1.1.2	Data Analysis and Processing on Amazon EMR	4
1.2	Tools	4
2	Subscription to Amazon Web Services	4
3	Amazon Elastic MapReduce using the Ruby Command Line Interface: Configuration	5
4	Amazon Simple Storage Service	6
4.1	Buckets, folders and files	6
4.2	How to access Amazon S3	6
5	Your first MapReduce job flow using Hadoop Streaming	7
5.1	Getting started: uploading your files to Amazon S3	7
5.2	Launching the Job Flow with Amazon Console	8
5.3	Launching the Job Flow with the CLI	8
5.4	Monitoring the Job Flow	10
5.5	Viewing the results	10
6	Your first MapReduce job flow using Hadoop Custom JAR	10
6.1	Creating the JAR file	11
6.2	Launching the Job Flow with Amazon Console	11
6.3	Launching the Job Flow with the CLI	12
6.4	Monitoring the Job Flow and seeing the results	12
7	Conclusion	12
8	Some useful links	14
8.1	Documentation	14
8.2	Pricing	14

1 Introduction

The goal of this document is to get you started as quickly as possible with Amazon Web Services (AWS). After reading this, **you should be able to run your own MapReduce jobs on Amazon Elastic MapReduce (EMR)**. I strongly recommend you to also have a look at [the official AWS documentation](#) after you finish this tutorial. The documentation is very rich and has a lot of information in it, but they are sometimes hard to find.

This tutorial is a concise version of the [Getting Started with Elastic MapReduce](#), complemented with several other useful resources. Instead of getting lost surfing across many different pages, this tutorial should give you everything to comfortably get started, in one place.

But first of all, let's have a very narrow overview of Amazon Web Services — in particular Amazon Elastic MapReduce — and what you can do with that.

1.1 What for?

1.1.1 Hadoop Programming on Amazon Elastic MapReduce

Amazon EMR makes it easy to spin up a set of Amazon EC2 instances as virtual servers to run a Hadoop cluster. When you're done developing and testing your application, you can terminate your cluster, only paying for the computational time you used.

Amazon EMR provides several types of clusters that you can launch to run custom Hadoop map-reduce code, depending on the type of program you're developing and the libraries you intend to use. We will focus on two of them: Streaming and Custom JAR.

Streaming. Hadoop streaming is the built-in utility provided with Hadoop. Streaming supports any scripting language, such as Python or Ruby. It is easy to read and debug, numerous libraries and data are available, and it is fast and simple. You can script your data analysis process and avoid writing code by using the existing libraries. Streaming is a low level interface. You are responsible for converting your problem definition into specific Map and Reduce tasks and then implementing those tasks via scripts.

Custom JAR. The Custom JAR job flow type supports MapReduce programs written in Java. You can leverage your existing knowledge of Java using this method. While you have the most flexibility of any job flow type in designing your job flow, you must know Java and the MapReduce API. Custom JAR is a low level interface. You are responsible for converting your problem definition into specific Map and Reduce tasks and then implementing those tasks in your JAR.

1.1.2 Data Analysis and Processing on Amazon EMR

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

1.2 Tools

There are several ways to interact with Amazon Web Services. In this tutorial, will focus on 2 of them¹:

- **The Amazon Console:** it is a graphical interface that you can use to launch and manage job flows, which is the easiest way to get started;
- **The Command Line Interface (CLI):** it is an application you run on your local machine to connect to Amazon EMR, and create and manage job flows.

2 Subscription to Amazon Web Services

First of all, you will need an Amazon Web Services account. Although you should already have performed this step, here i a quick recap. You will need an email account, your cellphone, and your credit card with you to begin. Begin by going to <http://aws.amazon.com> and then do the following:

1. Click **Sign up** at the top
2. Fill in your login credentials
3. Enter your contact information
4. In the **Payment Method** step, enter your credit card information
5. In the **Identity Verification** step, provide a phone number
6. Enter the pin code that is on your computer screen on your phone, and press *
7. Your account verification should be starting. It usually takes less than 5 minutes but it may take longer.
8. Once your account has been activated, you should receive a confirmation email.

¹Other tools exists: the SDK (that can allow you to write applications to automate the process of creating and managing job flows — for more information, please refer to <http://aws.amazon.com/sdkforjava/>), and the Web Services API (that you can use to call the web service directly using JSON).

9. Finally, you can go back to <http://aws.amazon.com> and click **My Account/Console** to log in.

After that, you can enter your redeem code (providing \$3,000 to your team) at the following address: <http://aws.amazon.com/awscredits/>. You can also grant access to your account by other people with [AWS Identity and Access Management \(IAM\)](#).

3 Amazon Elastic MapReduce using the Ruby Command Line Interface: Configuration

Before we actually start launching our first MapReduce job on Amazon EMR, we will install the Ruby Command Line Interface (CLI). The Amazon EMR CLI requires Ruby 1.8.7².

1. Create a directory called `elastic-mapreduce-cli` on your computer. Download the [Amazon Elastic MapReduce Ruby Client](#) to this directory and unzip it.
2. In the same directory, create a file `credentials.json` with the following content:

```
{
  "access_id": "[Your AWS Access Key ID]",
  "private_key": "[Your AWS Secret Access Key]",
  "keypair": "[Your key pair name]",
  "key-pair-file": "[The path and name of your PEM file]",
  "region": "[The region of your job flow, either us-east-1,
             us-west-2, us-west-1, eu-west-1, ap-northeast-1, ap-
             southeast-1, ap-southeast-2, or sa-east-1. Choose for
             instance us-west-1]"
}
```

In order to get your AWS Access Key ID and AWS Secret Access Key, connect to <http://aws.amazon.com>, and click on **My account/Console** → **Security Credentials** in the top right menu.

- `access_id`: your Access Key ID is displayed in the **Access Credentials** section.
- `private_key`: to display your Secret Access Key, click **Show** in the **Your Secret Access Key** area.

In order to get your key pair information, go to **My account/Console** → **Amazon Management Console** → **EC2**. In the top-left drop down menu, select the region you entered in the `region` field (so you will

²if you have troubles installing Ruby 1.8.7, please refer to [the detailed steps here](#).

probably select **US West (N. California)** and click **Key pair**. Then click on **Create Key Pair**. Save the resulting PEM file in a safe location. Now, back to the `credentials.json` file:

- `keypair`: should contain the your key pair name.
- `key-pair-file`: should contain the location of your `*.pem` file.

Then finally³:

- `region`: choose for instance `us-west-1`. It has to be the same as the region you selected in the previous step while generating your key pair.
3. To verify that the CLI has been installed correctly, go to the directory where you installed the CLI with your Terminal, and type the command `./elastic-mapreduce --version` in your terminal⁴. You should see something like `Version 2012-12-17`.

4 Amazon Simple Storage Service

4.1 Buckets, folders and files

In order to run an Elastic MapReduce job, we need to create a bucket on Amazon Simple Storage Service (S3), the AWS storage service, which will allow us to store our input files, our algorithms, and to save the outputs. This can be achieved by navigating to **Amazon Management Console** → **S3** → **Create Bucket**. Enter your bucket name⁵, and click **Create**. Say we just created the bucket `mybucket` (which is very unlikely since all bucket names have to be unique across all AWS users).

4.2 How to access Amazon S3

The **AWS Management Console** → **S3** tab gives users filesystem like access to their buckets. Users can create folders in their buckets, upload files to these folders and set their permissions, and delete them.

There are other ways to access the buckets and files on S3. You can use either one, but keep in mind that there may be some problems if you upload files whose size is greater than 5 GB. In that case, you have to use a utility that supports *multipart uploads*, such as Cyberduck or s3afe.

³You could also fill in a `log-uri` field in this JSON. If you decide to do so, this should be the path of a bucket you own on Amazon S3, such as `s3n://mys3bucket/mylog-uri`. This directory would contain all the log files of your job flows. However, you can specify a specific path when you are launching a job flow. We'll get to that soon.

⁴For Windows users, you would use Cygwin and type `ruby elastic-mapreduce --version`. More generally, this document will contain the commands for Linux/Mac OS X operating systems. If you are a Windows user, just replace `./elastic-mapreduce` by `ruby elastic-mapreduce` to start all your commands.

⁵Your bucket name must contain only lowercase letters, numbers, periods (`.`), and dashes (`-`), and has to be *unique* ([more information on buckets here](#)).

- The command line utility [s3cmd](#) allows you to do pretty much everything.
- There is also a Firefox extension called [S3Fox](#).
- You can use a FTP client such as [Cyberduck](#), which would be very useful for file sizes greater 5 GB.
- Finally, you can use [s3afe](#), a command line Python script (based on the [boto library](#)) which should also be able to handle big files.

5 Your first MapReduce job flow using Hadoop Streaming

A streaming job flow reads input from standard input and then runs a script or executable (called a *mapper*) against each input. The result from each of the inputs is saved locally, typically on a Hadoop Distributed File System (HDFS) partition. Once all the input is processed by the mapper, a second script or executable (called a *reducer*) processes the mapper results. The results from the reducer are sent to standard output. You can chain together a series of streaming job flows, where the output of one streaming job flow becomes the input of another job flow. The functions can be implemented in any of the following supported languages: Ruby, Perl, Python, PHP, R, Bash, C++.

5.1 Getting started: uploading your files to Amazon S3

We show an example of using Elastic MapReduce in the streaming format where it is possible to use unix executables as mapper and reducers. In this word count example, we use the `mapper.py` as the mapper and `reducer.py` as the reducer, which you can [download from the class website](#).

Note that in streaming, any executable which writes its output to `stdout` can be used as a mapper or a reducer. By default, the prefix of a line up to the first tab character is the key and the the rest of the line (excluding the tab character) will be the value. For further details, please refer to the [Hadoop documentation on streaming](#).

We need to copy the above code into files called `mapper.py` and `reducer.py` and save them to the local filesystem. Next, for instance using the **AWS Management Console** → **S3** tab, we create a folder `streamingcode` in our bucket `mybucket` and upload the files `mapper.py` and `reducer.py` to it.

We also have to upload our input file, the [Alice's Adventures in Wonderland](#) text (from the Gutenberg Project) from the class website. Upload the `alice.txt` file to a folder called `input` in the bucket `mybucket`.

And now, we are ready to go! We will see two different methods to launch your first job flow: one that uses the Amazon Console (a Graphical User Interface), and one that uses the Command Line Interface we installed in Section 3.

5.2 Launching the Job Flow with Amazon Console

With its Graphical User Interface, the Amazon Console is the easiest way to get started.

Go to **Amazon Management Console** → **Elastic MapReduce** → **Create New Job Flow**.

1. **Define Job Flow.** Enter a name for your job flow (pick something specific), and select *Streaming* just like in Figure 1.
2. **Specify Parameters.** This is where you tell Amazon where your input files, output files, mapper and reducer are. Enter information consistently with what we did before, just like in Figure 2.
3. **Configure EC2 Instances.** Keep default parameters.
4. **Advanced Options.** Keep default parameters⁶.
5. **Bootstrap Actions.** Keep default parameters.
6. **Review.** This is a summary of what we did. After reviewing that everything is correct, click **Create Job Flow**.

After you click **Create Job Flow**, your request is processed, and you should receive a success message. You can now jump directly to Section 5.4, unless you want to see directly another way to launch a job flow, from the command line interface.

5.3 Launching the Job Flow with the CLI

Another solution to launch your job flow is to use the Ruby Command Line Interface that we installed earlier. This is probably what you will end up doing during the quarter as you are getting more and more familiar with AWS.

In order to launch your job flow from the CLI, you can run the following command⁷ from the `elastic-mapreduce-cli` directory (where you should replace `mybucket` with the actual name of your S3 bucket):

```
./elastic-mapreduce --create --stream \  
  --input s3n://mybucket/input \  
  --output s3n://mybucket/output-streaming-cli \  
  --mapper s3n://mybucket/streamingcode/mapper.py \  
  --reducer s3n://mybucket/streamingcode/reducer.py
```

⁶If you want, you can specify a bucket/folder for your log files on Amazon S3, such as `s3n://mybucket/log-streaming-console`, and enable debugging.

⁷Note that the backslashes \ just represent the fact that everything should be on the same line.

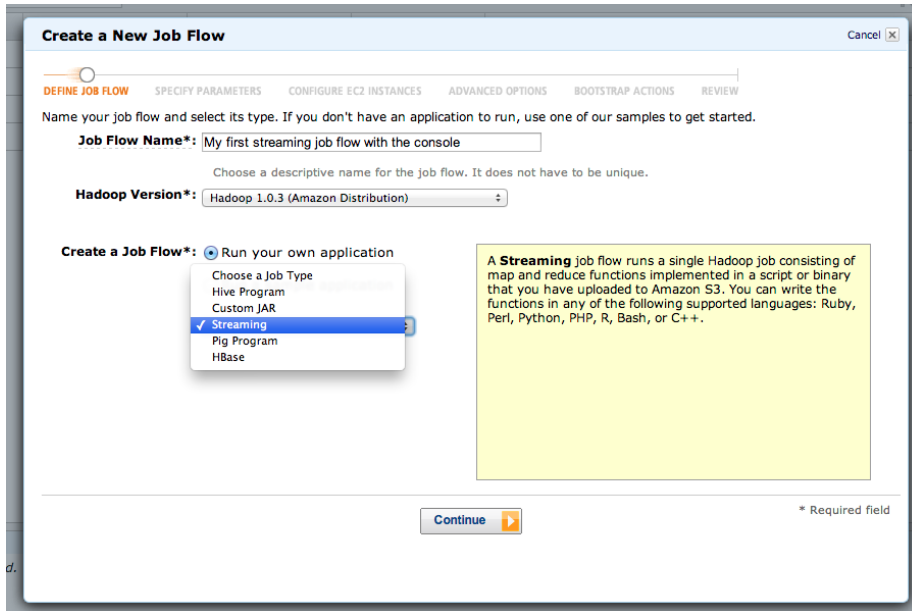


Figure 1: First step to create a streaming job flow from the Amazon Console.

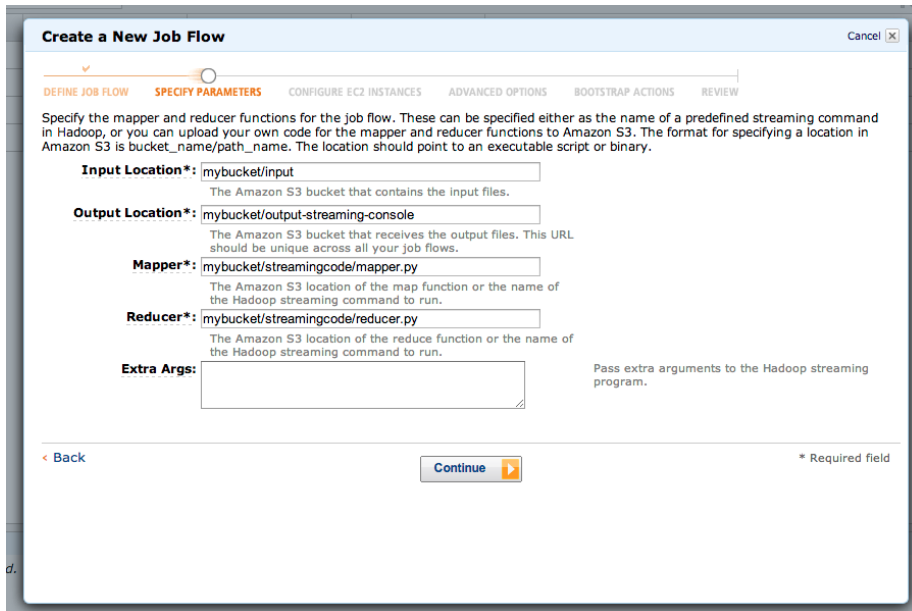


Figure 2: Second step to create a streaming job flow from the Amazon Console.

Note that we are basically entering the same parameters as in the Amazon Console, except that we are doing it with a command line interface. More options are available (as you can see [here](#)) but for now, these options are more than enough⁸.

5.4 Monitoring the Job Flow

Go to **AWS Management Console** → **Elastic MapReduce**: you should now see your job flow in the list. Completing a job flow, even on small inputs, can take a while (expect 3 or 4 minutes for this example with Alice). Clicking on the job flow will allow you to access more information in the tabs below⁹.

5.5 Viewing the results

Once your job flow is marked as completed, go to **AWS Management Console** → **S3**, and click on the output folder you specified (which should be `output-streaming-console` or `output-streaming-cli`). You can see the files `part-0000*`, which contain the results of your job flow.

Important note. The commands we run to launch our job flow in this example are set to automatically terminate the EC2 instance that is running the jobs. However, as you get into more custom commands, this will not necessarily be the case. In those situations, you have to **terminate the instance**. Otherwise, you will get credited for the time they are running (even if the job flow completed). To terminate an instance, you can either:

- Go on **Amazon Management Console** → **Elastic MapReduce**, select your Job Flow and click on **Terminate**;
- Or use the Command Line Interface and type `./elastic-mapreduce --terminate <your-job-id>`. You can find the ID of your job flow by clicking on the job flow in the Amazon Console. The ID should look like `j-6RSCDZSF3VHZ`.

6 Your first MapReduce job flow using Hadoop Custom JAR

A job flow using a custom JAR file enables you to write a script to process your data using the Java programming language. This provides low-level access to

⁸By default, this command launches a job flow to run on a single-node cluster using an Amazon EC2 `m1.small` instance. Later, when your steps are running correctly on a small set of sample data, you can launch job flows to run on multiple nodes. You can specify the number of nodes and the type of instance to run with the `--num-instances` and `--instance-type` parameters, respectively. You can also add an argument `--log-uri` if you want to get a log file, for instance: `--log-uri s3n://mybucket/log-streaming-cli`

⁹For more information about what these tabs mean, please refer to [Step 6: Monitor the Job Flow](#) in the Amazon documentation.

the MapReduce API. You have the responsibility of defining and implementing the map-reduce tasks in your Java application.

6.1 Creating the JAR file

First of all, we have to create the custom JAR file that contains our map-reduce tasks. Before we start, download the [Word Count Java files](#) from the class website. Also download the version 1.0.3 of Hadoop (which is the version used by Amazon) from the [Cloudera Repository](#): click on `hadoop-core-1.0.3.jar`.

1. Create a new Java project in Eclipse.
2. Right click on your project, and then **Build Path** → **Add External Archives...**, and select the `hadoop-core-1.0.3.jar` file you downloaded.
3. Create a new class named *Map*, and copy and paste the code from `Map.java` that you downloaded.
4. Create a new class named *Reduce*, and copy and paste the code from `Reduce.java` that you downloaded.
5. Create a new class named *WordCount*, and copy and paste the code from `WordCount.java` that you downloaded.
6. Right click on your project, and then **Export...**
 - Select **Java** → **JAR file** and click **Next**;
 - On the screen *JAR File Specification*, select the destination of your JAR file and click **Next**;
 - On the screen *JAR Packaging Options*, keep default values and click **Next**;
 - On the last screen *JAR Manifest Specifications*, keep default values and enter the name of your main class in the last text box. Here, it is `WordCount`. Click **Finish**.

Now you can upload your JAR file to your Amazon S3 bucket, for instance in the folder `code`.

6.2 Launching the Job Flow with Amazon Console

Go to **Amazon Management Console** → **Elastic MapReduce** → **Create New Job Flow**.

1. **Define Job Flow**. Enter a name for your job flow (pick something specific), and select *Custom JAR* just like in Figure 3.

2. **Specify Parameters.** This is where you tell clusters where your input files, output files, mapper and reducer are. In *JAR Location*, enter the location of the JAR file you uploaded to Amazon S3; in *JAR Arguments*, enter the arguments of your Java program just like you would do in Eclipse's *Run Configurations*. So in our case, it is the S3 path of the input followed by the S3 path of where we want our output to be. Refer to Figure 4 to see it in picture.
3. **Configure EC2 Instances.** Keep default parameters.
4. **Advanced Options.** Keep default parameters.
5. **Bootstrap Actions.** Keep default parameters.
6. **Review.** This is a summary of what we did. After reviewing that everything is correct, click **Create Job Flow**.

After you click **Create Job Flow**, your request is processed, and you should receive a success message.

6.3 Launching the Job Flow with the CLI

Just like we did for the Streaming part, if we want to launch the job flow from the Command Line Interface instead of Amazon Console, we just transcribe the parameters into lines of code:

```
./elastic-mapreduce --create --name "My first Custom JAR job flow
  from the CLI" \
  --jar s3n://mybucket/code/WordCount.jar \
  --arg s3n://mybucket/input/ \
  --arg s3n://mybucket/output-customjar-cli
```

6.4 Monitoring the Job Flow and seeing the results

It is exactly the same as in Sections 5.4 and 5.5.

7 Conclusion

You should now be able to launch your own MapReduce jobs on Amazon Web Services! Getting familiar with AWS requires time and practice, but hopefully this tutorial helped you dive into it and get a first grasp of how things work.

Amazon Web Services is a very flexible platform and you can do pretty much everything you want with it. For instance, instead of using Amazon Elastic MapReduce, which has all Hadoop preconfigured, you could choose to use Amazon EC2 and configure yourself everything you want. But this is a more advanced topic which is not suitable for this tutorial.

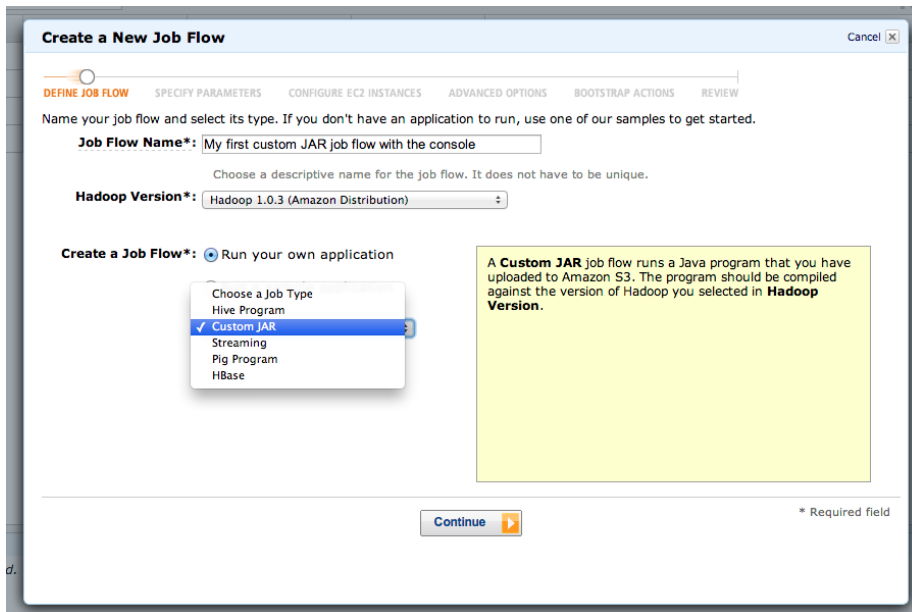


Figure 3: First step to create a Custom JAR job flow from the Amazon Console.

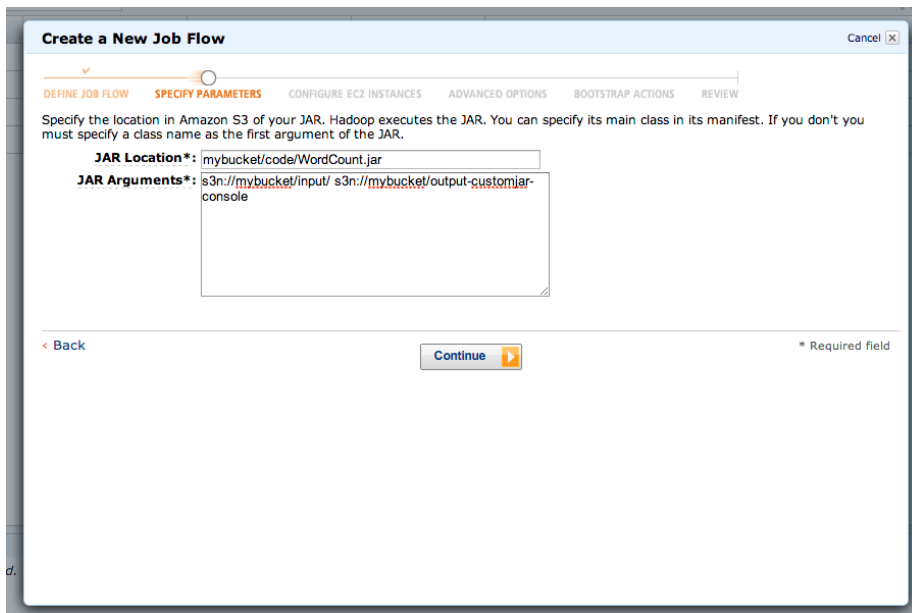


Figure 4: Second step to create a Custom JAR job flow from the Amazon Console.

8 Some useful links

8.1 Documentation

- To get started, the [Amazon Elastic MapReduce Getting Started Guide](#).
- For more in-depth topics, the [Amazon Elastic MapReduce Developer Guide](#).
- More generally, you can find the documentation of Amazon on:
<http://aws.amazon.com/documentation/>

8.2 Pricing

Even if pricing should not be an issue with your \$3,000 credited account (**unless you forget to shut down an instance** — we will never stress that enough), it is a good thing to have an idea of the costs.

- Amazon Elastic MapReduce Pricing:
<http://aws.amazon.com/elasticmapreduce/pricing/>
- Amazon EC2 Pricing: <http://aws.amazon.com/ec2/pricing/>
- Amazon S3 Pricing: <http://aws.amazon.com/s3/pricing/>