Getting Started with Amazon Web Services (AWS) and Elastic MapReduce (EMR) Jobs

1. Introduction

This document describes how you can get started using AWS to run MapReduce jobs on Amazon Elastic MapReduce (EMR). If you want to try this out you should also have a look at the official AWS documentation. This documentation summarizes the Getting Started with Elastic MapReduce guide.

1.1 What?

1.1.1 Hadoop Programming on Amazon Elastic MapReduce

Amazon EMR makes it easy to spin up a set of Amazon Elastic Computing Cloud (EC2) instances as virtual servers to run a Hadoop cluster. When you're through developing and testing your application, you can terminate your cluster, only paying for the computation 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 a 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 approach. 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, we focus on two of them:

• **The Amazon Console**: a graphical interface that you can use to launch and manage job flows, which is the easiest way to get started. This approach will be demonstrated in class.

• **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. The standard CLI interface is built on Ruby. This approach will not be described further in this document.

2. Subscription to Amazon Web Services

In order to work with AWS you need an account. To set up an account you need an email account, your cellphone, and your credit card. 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.
8. Once your account has been activated, you should receive a confirmation email.
9. Finally, you can go back to http://aws.amazon.com and click **My Account/Console** to log in.

3. Amazon Simple Storage Service

3.1 Buckets, folders and files

In order to run an Elastic MapReduce job, you need to create a *bucket* on Amazon Simple Storage Service (S3), the AWS storage service, which will allow you to store input files, code, and outputs. This can be achieved by navigating to **Amazon Management Console → S3 → Create Bucket**. Enter your bucket name, and click **Create**. Let's say we named our bucket *mybucket* (which is very unlikely since all bucket names have to be unique across all AWS users).

3.2 How to access Amazon S3

The **AWS Management Console → S3** tab gives users files system 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.

4. 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++.

4.1 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 `mapper.py` as the mapper and `reducer.py` as the reducer. Files available on Sakai.

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 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 file system. 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 DonQuixote text (from the Gutenberg Project). Upload the DonQuixote.txt file to a folder called `input` in the bucket mybucket. And now, we are ready to go! We will walk through the approach of using the Amazon Console GUI but a command line interface (CLI) can be used to accomplish the same goal.

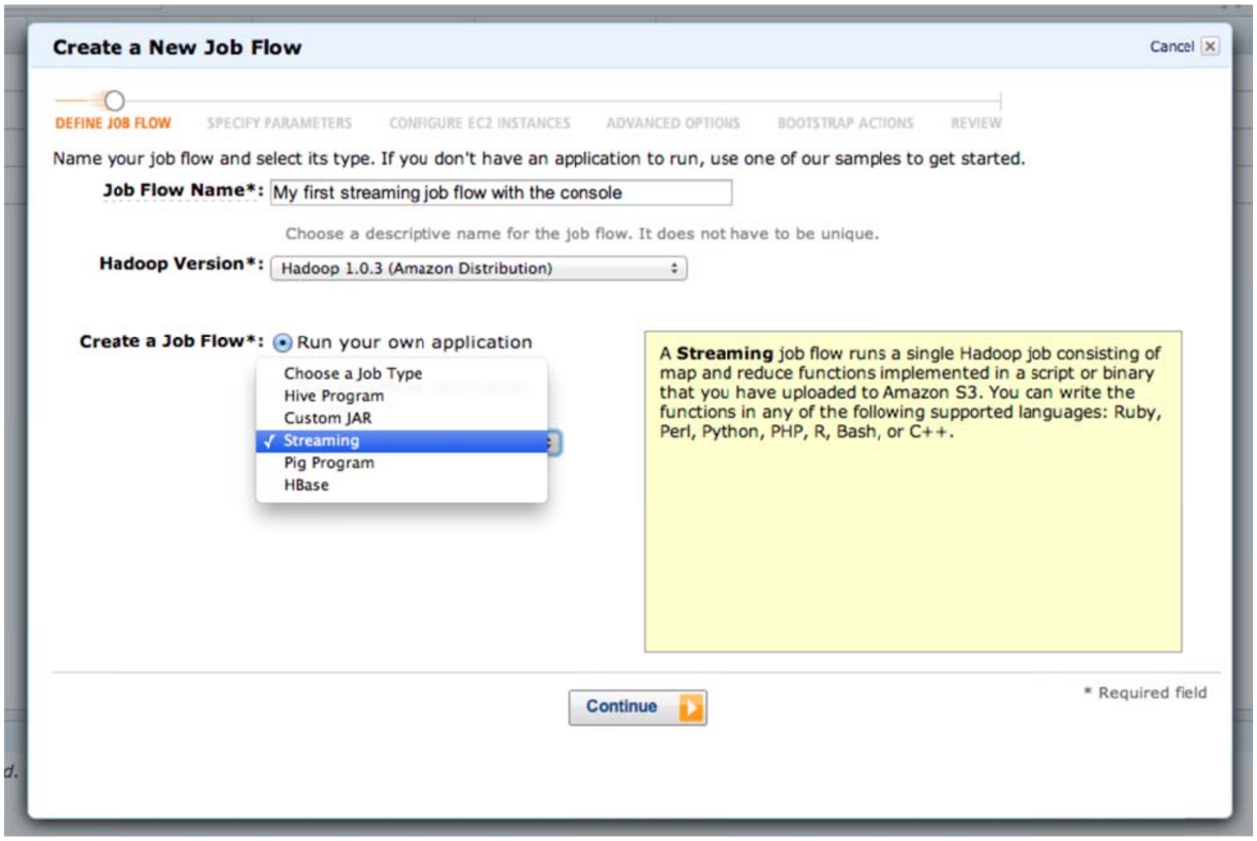
4.2 Launching the Job Flow with Amazon Console

With a GUI 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 below.
2. **Specify Parameters.** This is where you tell Amazon where your input files, output files, mapper and reducer are. Enter information consistently as in Figure 2.
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.



Name your job flow and select its type. If you don't have an application to run, use one of our samples to get started.

Job Flow Name*: My first streaming job flow with the console

Choose a descriptive name for the job flow. It does not have to be unique.

Hadoop Version*: Hadoop 1.0.3 (Amazon Distribution)

Create a Job Flow*: Run your own application

A **Streaming** job flow runs a single Hadoop job consisting of map and reduce functions implemented in a script or binary that you have uploaded to Amazon S3. You can write the functions in any of the following supported languages: Ruby, Perl, Python, PHP, R, Bash, or C++.

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

4.3 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 5-8 minutes for this example with DonQuixote). Clicking on the job flow will allow you to access more information in the tabs below.

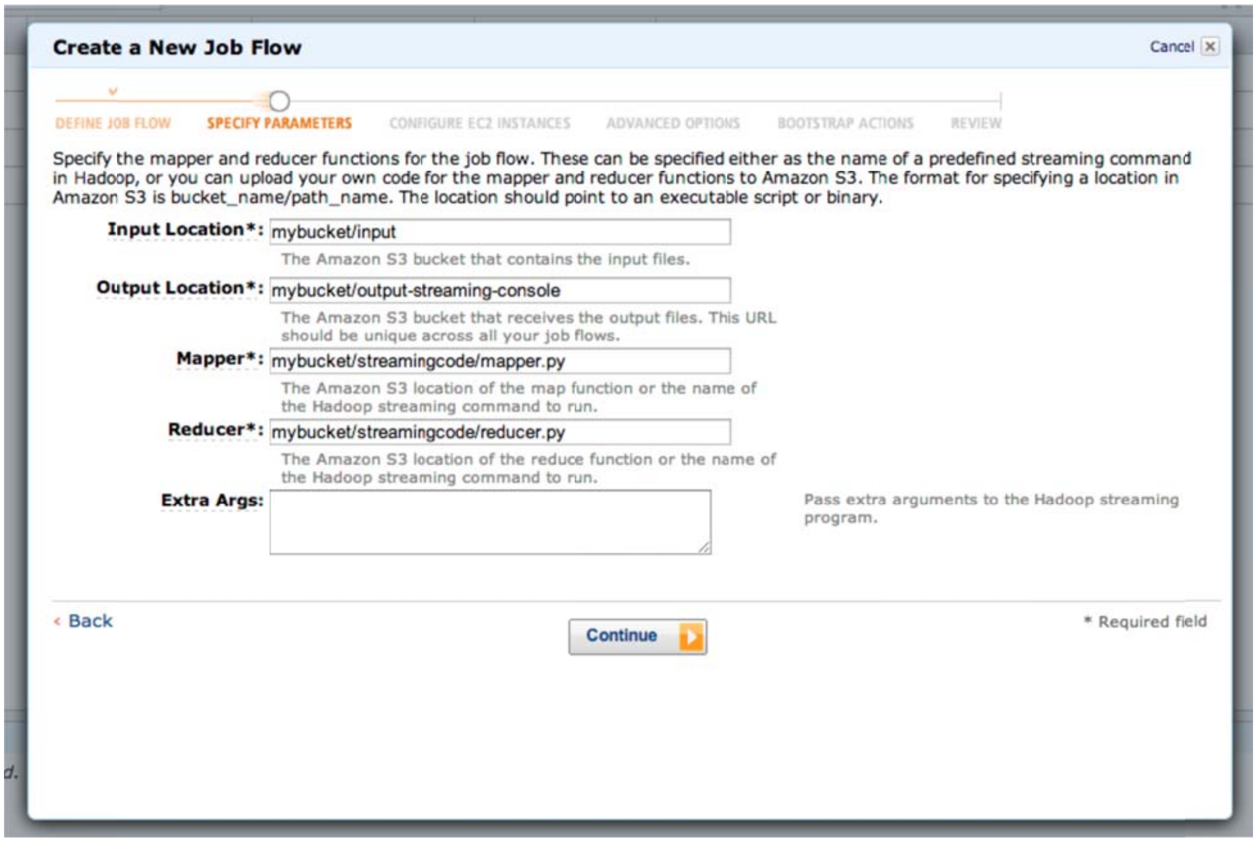
4.4 Viewing the results

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 or similar). 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.



Specify the mapper and reducer functions for the job flow. These can be specified either as the name of a predefined streaming command in Hadoop, or you can upload your own code for the mapper and reducer functions to Amazon S3. The format for specifying a location in Amazon S3 is bucket_name/path_name. The location should point to an executable script or binary.

Input Location*:	mybucket/input — The Amazon S3 bucket that contains the input files.
Output Location*:	mybucket/output-streaming-console — The Amazon S3 bucket that receives the output files. This URL should be unique across all your job flows.
Mapper*:	mybucket/streamingcode/mapper.py — The Amazon S3 location of the map function or the name of the Hadoop streaming command to run.
Reducer*:	mybucket/streamingcode/reducer.py — The Amazon S3 location of the reduce function or the name of the Hadoop streaming command to run.
Extra Args:	Pass extra arguments to the Hadoop streaming program.

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

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

5.1 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 wordcount.java 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 WordCount, and copy and paste the code from wordcount.java that you downloaded.
4. 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 named code.

5.2 Launching the Job Flow with Amazon Console

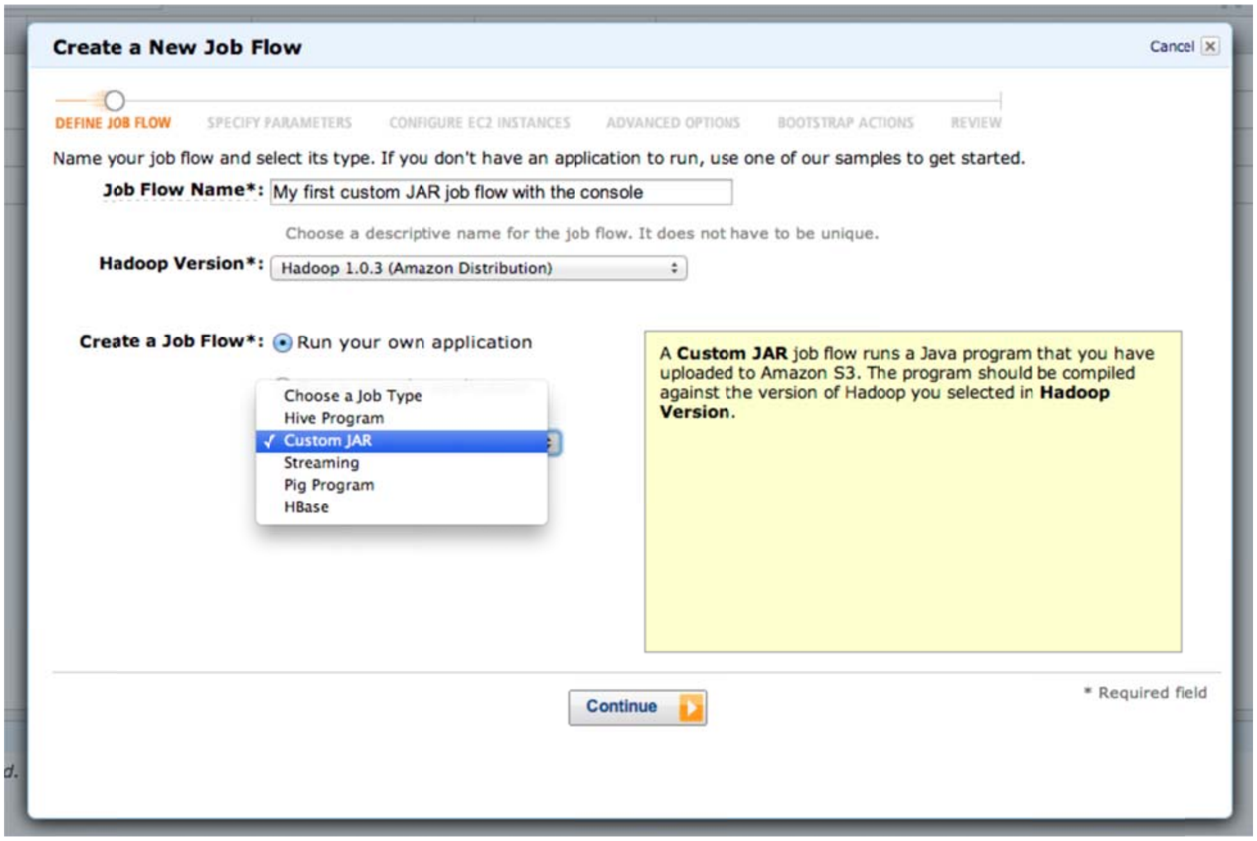
Go to **Amazon Management Console → Elastic MapReduce → Create New Job Flow.**

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 indicate where your input files, output files, and .jar file. 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.
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.

5.3 Monitoring the Job Flow and seeing the results

Same as in Sections 4.3 and 4.4.



Create a New Job Flow

Name your job flow and select its type. If you don't have an application to run, use one of our samples to get started.

Job Flow Name*: My first custom JAR job flow with the console

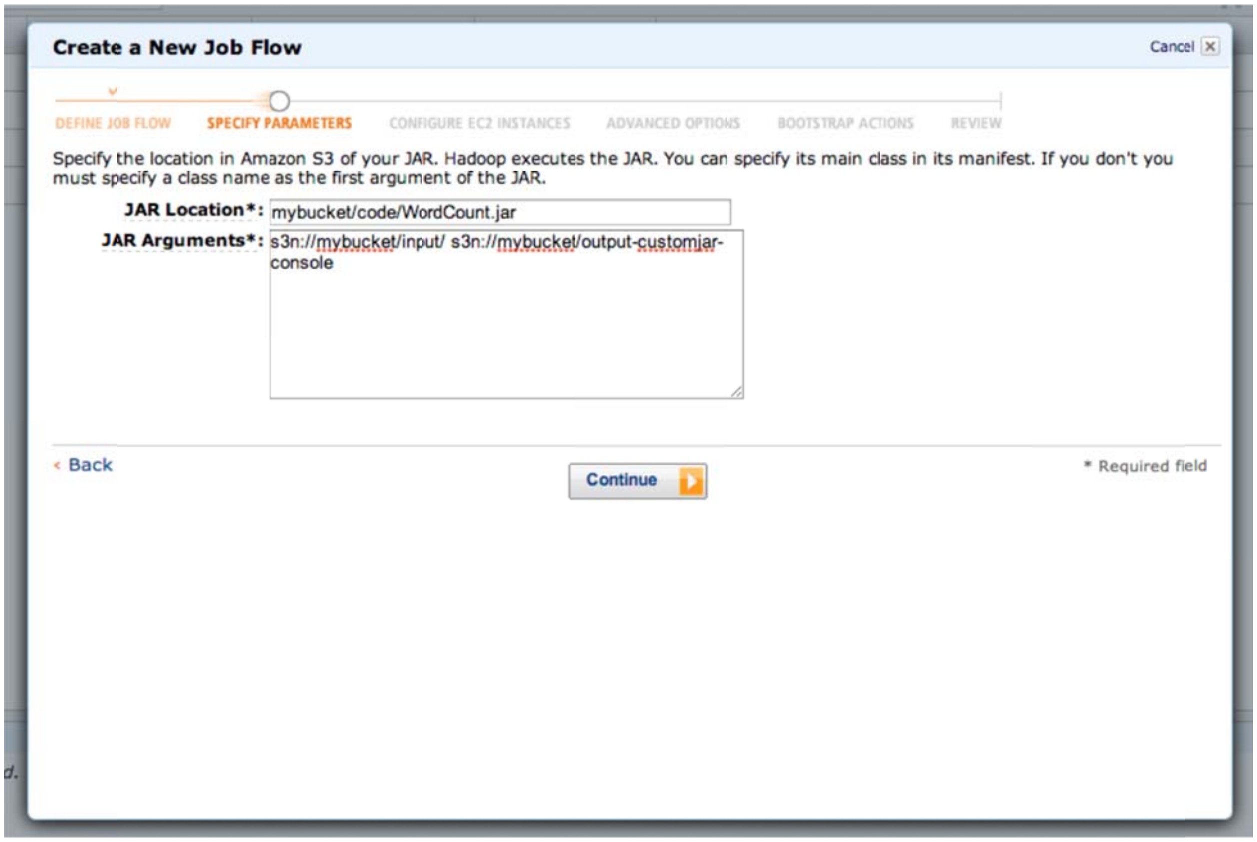
Choose a descriptive name for the job flow. It does not have to be unique.

Hadoop Version*: Hadoop 1.0.3 (Amazon Distribution)

Create a Job Flow*: ● Run your own application

A **Custom JAR** job flow runs a Java program that you have uploaded to Amazon S3. The program should be compiled against the version of Hadoop you selected in **Hadoop Version**.

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



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*: mybucket/code/WordCount.jar

JAR Arguments*: s3n://mybucket/input/ s3n://mybucket/output-customjar-console

* Required field

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

6. Conclusions

You now have the basic information needed to launch your own MapReduce jobs on Amazon Web Services! Getting familiar with AWS requires time and practice, but this tutorial should help you dive into it and get a 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 Hadoop preconfigured, you could choose to use Amazon EC2 and configure yourself everything you want.