Felipe Guth – COMP41110 – Student Number: 14210231

Project-3 (Hadoop/Map/Reduce- log analysis) 1- Student

In this project, you will gain hands-on experience with the Map/Reduce programming model building and running an application using Hadoop. A common task in digital forensics is the analysis of log files. Each line (entry) of a web server's log file normally contains important information such as: IP address, data and time of request, request line, HTTP status, etc. Write a Map Reduce program that takes a log file as an input and then extracts the following information from the log files:

a)Total number of connections to the server (i.e. total numbers of entries).

b)List of distinct IPs.

c)Number of entries for each IP.

d)List of distinct IPs, number of entries for each IP in a given periods (e.g.from 01/02/2015to 28/02/2015)

You can use any Hadoop platforms to test your program (AWSMap/Reduce/Hadoop, Google Hadoop Cloud, etc.).

Model Design

The next figure shows how the solution for log analysis was designed.

Log File: Is the input file of a server that follow a determine pattern of logs of web servers. For this application it was used a sample log downloaded from the Apache HTTP server project. It contains a list of request to a server from different IPs and shows data as IP, date time, link, type of operation, etc.

Map Reducer: Map/Reduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key. Many real world tasks are expressible in this model, as shown in the paper.

Map tasks:

Map Input: The input data of the map tasks are a number of n lines of the log file that contains data about the requests to a web server.

Map Output: The output of the map tasks are the emission of {i,k} pairs of key, value. Where the keys are the IPs gathered in each line of the log file and the value 1 that is used for counting.

Reducer tasks:

Reducer Input: The {i,k} key, values pairs produces by the mappers are the input for the reducer tasks. Each reduce task processes a determine number of keys and produce a counting.

Reducer Output: The output of each reducer tasks is an output file that contain the keys (IPs) and values (count) processed by each reducer.

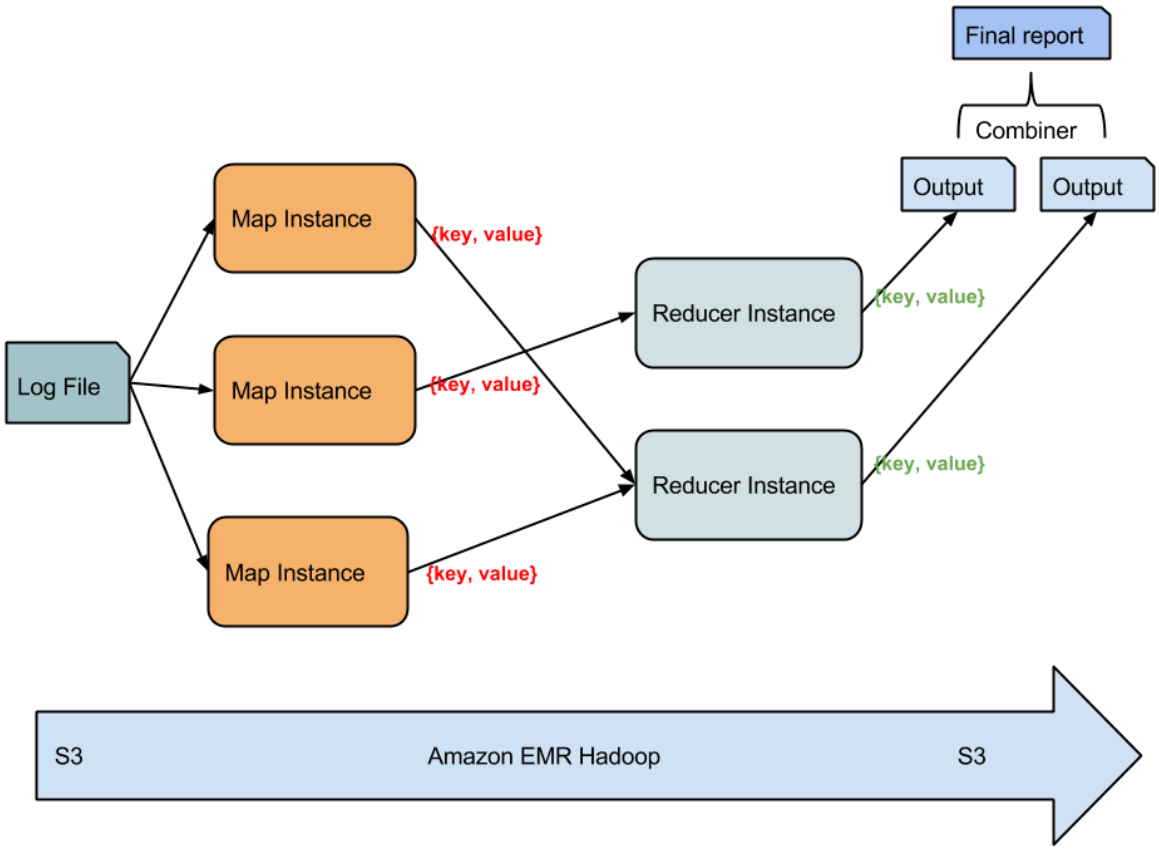
Combiner:

In order to generate a final report accordingly to the project propose, a combiner was implemented to read the output of each reducer and count the number of access from each IP to produce the total number of connections to the server.

Filtering:

With the purpose of filter the data given a data range it was implemented in the map function a routine that takes into account the dates specified by the user. As in each Map Reduce application on Hadoop, the user must specify the archives for the mapper and the reducer, it turns out that the most time effective solution is to include the range dates directly in the map script. In that way, many different Map Reduce applications can be configured with different copies of the map script filtering the data in distinct date ranges.

In local tests it is possible to develop a script to receive the dates as parameters by the system call, i.e., **./map.py 05/03.2004 06/03/2004**. However in the map reduce Hadoop environment this seems not possible until now. A question was created in the Stack Overflow regarding the passage of parameters in Hadoop, but there is no answer until this date (http://stackoverflow.com/questions/29549740/how-to-pass-arguments-to-streaming-job-on-amazon-emr). For this reason, the date parameters were fixed in the code of the map function.



System Architecture

In order to test the Map Reduce application for log analysis the system was deployed in the Amazon Infrastructure. Amazon provides different version of Hadoop that use server and storage integrated resources, also provided by Amazon.

Hadoop:

Hadoop Map/Reduce is a software framework for easily writing applications which process vast amounts of data (multi-terabyte data-sets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner.

A Map/Reduce job usually splits the input data-set into independent chunks which are processed by the map tasks in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the reduce tasks. Typically both the input and the output of the job are stored in a file-system. The framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks.

Typically the compute nodes and the storage nodes are the same, that is, the Map/Reduce framework and the Hadoop Distributed File System (see HDFS Architecture Guide) are running on the same set of nodes. This configuration allows the framework to effectively schedule tasks on the nodes where data is already present, resulting in very high aggregate bandwidth across the cluster.

Amazon Elastic Map Reduce (EMR):

Amazon Elastic Map/Reduce (Amazon EMR) is a web service that makes it easy to quickly and cost-effectively process vast amounts of data. Amazon EMR is based on Hadoop, a Java-based programming framework that supports the processing of large data sets in a distributed computing environment. Amazon EMR processes data across a Hadoop cluster of virtual servers on the Amazon Elastic Compute Cloud (EC2). The elastic in EMR's name refers to its dynamic resizing ability, which allows it to ramp up or reduce resource use depending on the demand at any given time.

Amazon EMR is used for data analysis in log analysis, web indexing, data warehousing, machine learning, financial analysis, scientific simulation, bioinformatics and more. (Amazon.com)

Amazon Simple Storage Service (S3):

Amazon Simple Storage Service (Amazon S3), provides developers and IT teams with secure, durable, highly-scalable object storage. Amazon S3 is easy to use, with a simple web services interface to store and retrieve any amount of data from anywhere on the web. With Amazon S3, you pay only for the storage you actually use. (Amazon.com).

The S3 is integrated with the Amazon solution (EMR), where the scripts of the map and reducer and the input log are linked in the Map Reduce application. The EMR also generates the reducer output files in a pre-defined S3 bucket. After that, these files are downloaded from the Amazon S3 cloud and are processed by the combiner script to produce the final report.

Local Testing

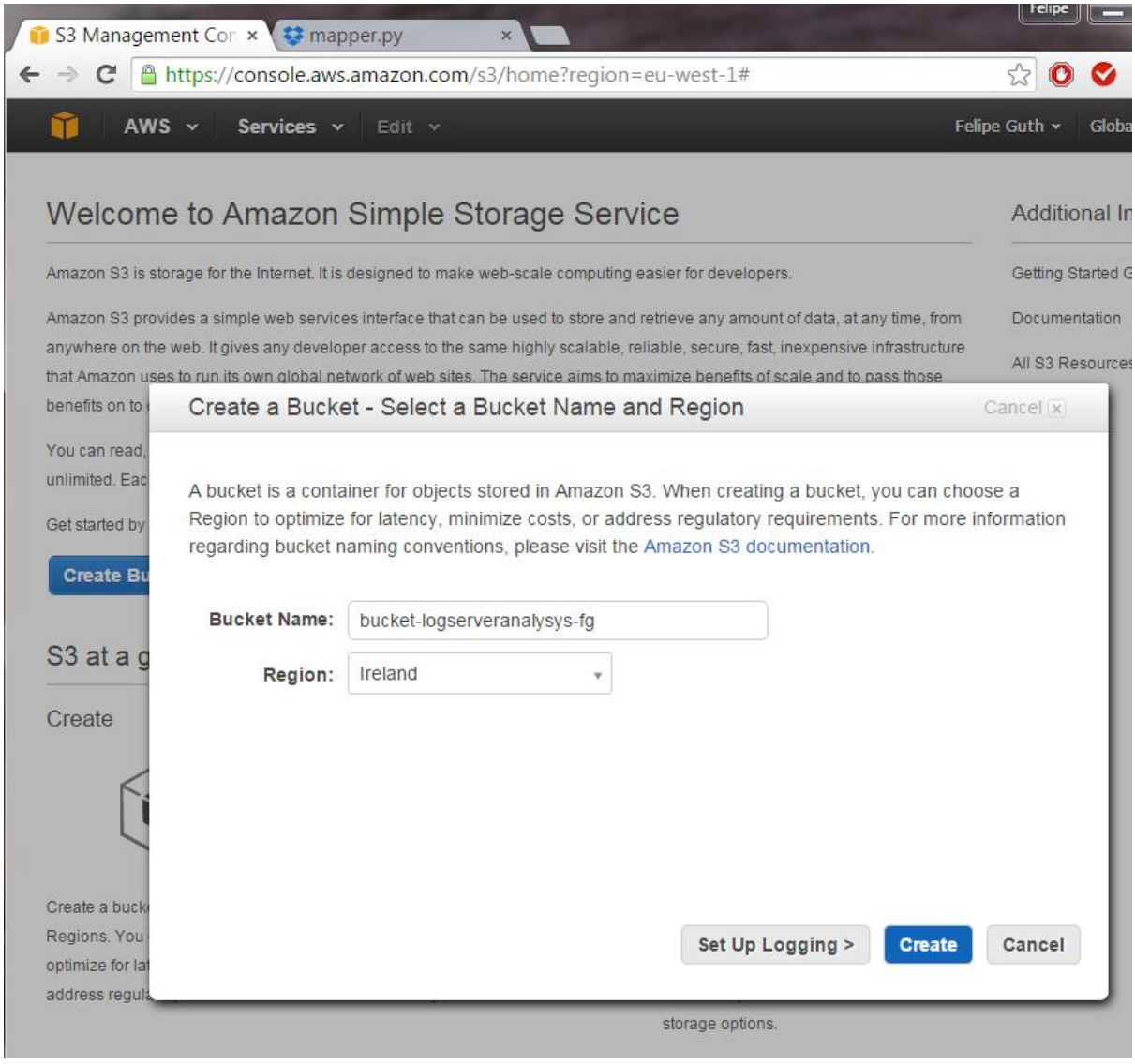
Beforehand the deployment of the system in the cloud infrastructure, local tests were performed. The scripts were development in the Python language. To test it is necessary to identify the scripts are executable archives. In Linux: `chmod +x filename`. After that the test is straightforward:

```
cat access_log | ./map.py | ./reducer.py
```

Deployment of the System in the Amazon Cloud

Creation of a storage bucket in the Amazon S3

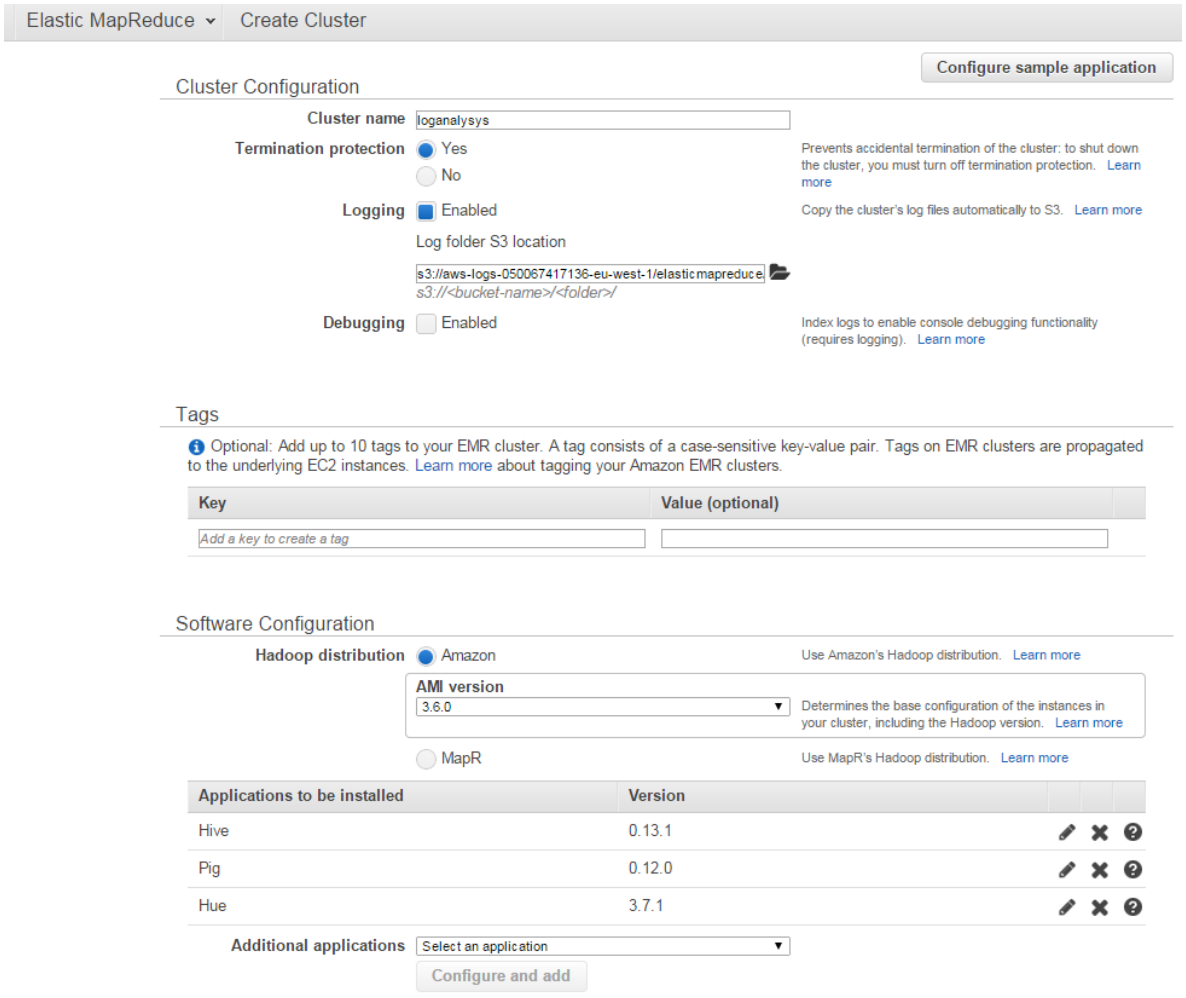
The first step was to create a bucket to store the input files, scripts and output.

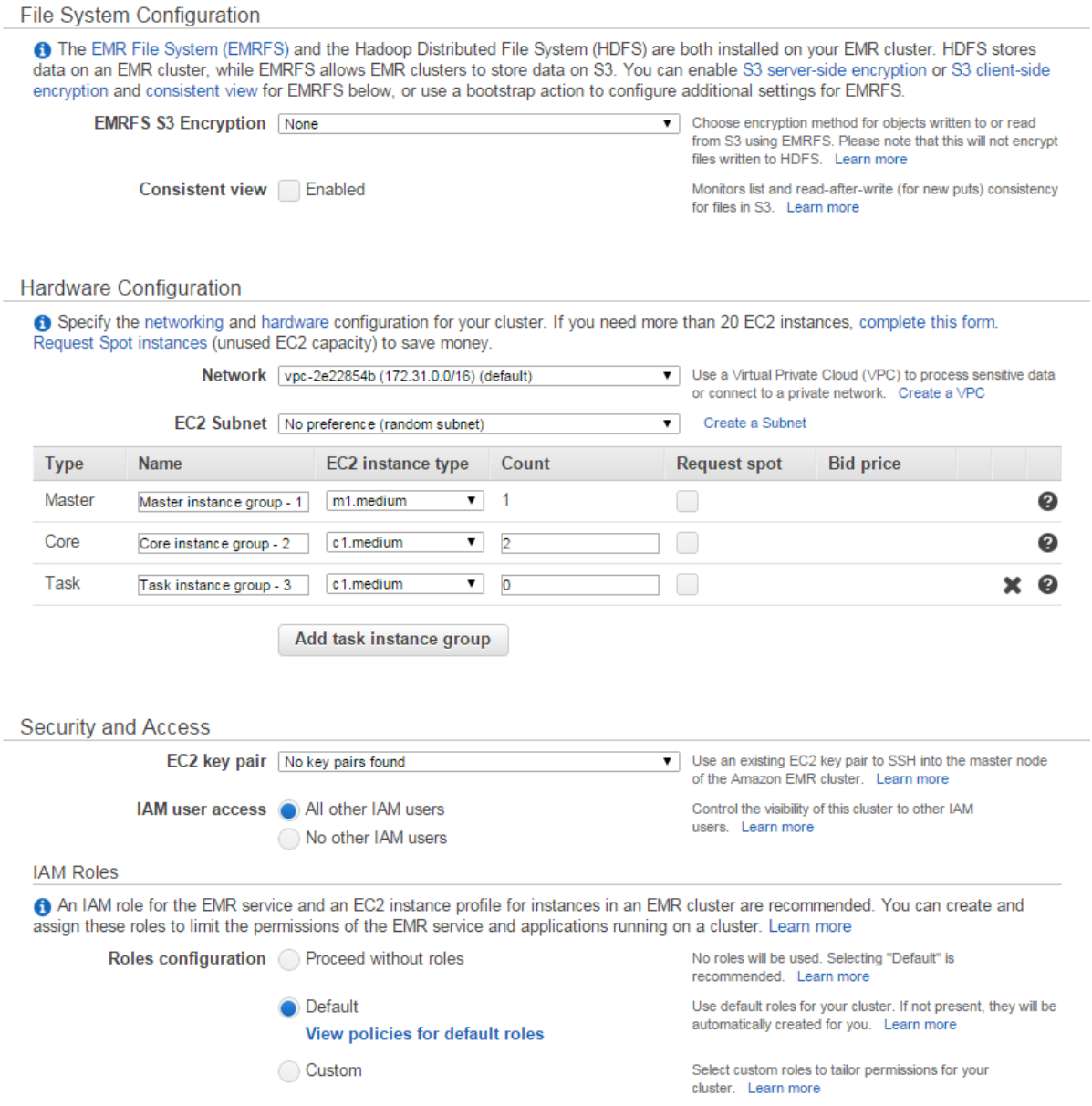


After the creation of the bucket, folders were created inside the bucket for the input log file and the script programs, afterwards the scripts and the input log file were upload to the Amazon cloud.

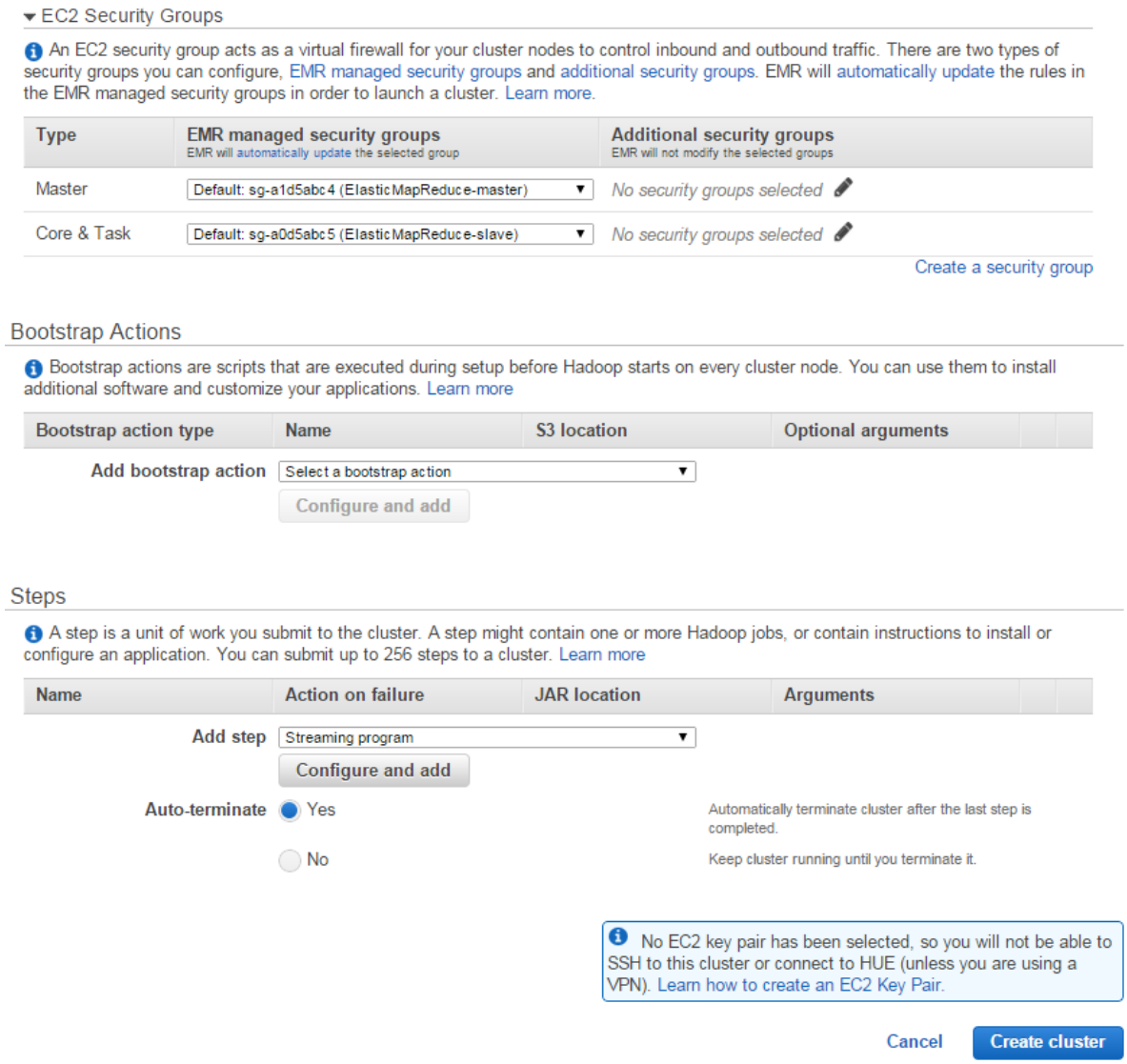
Hadoop Cluster Creation with Amazon EMR

The following screenshots show the creation of the EMR cluster in the Amazon cloud platform.





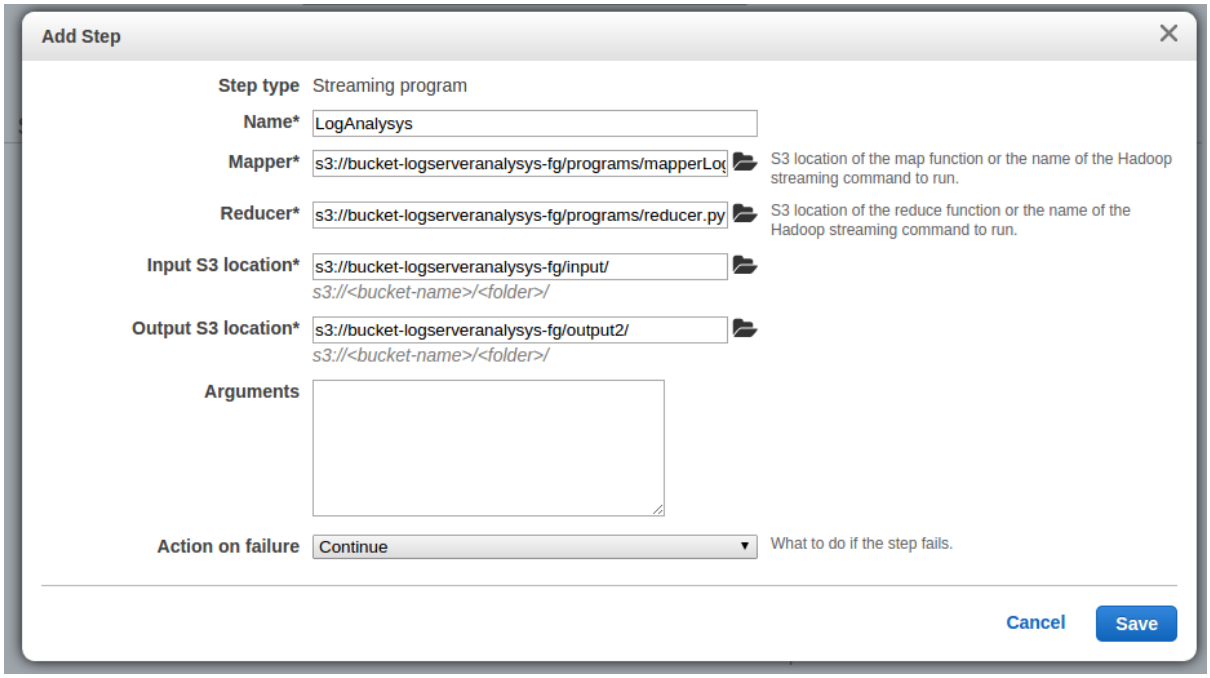
The Hardware configuration should be selected according to the needs of the user. A range of different hardware configurations are available. Best resources result in a more expensive bill.



The configuration of the Map/Reduce tasks occurs in the Steps section of the cluster setup. As Python scripts were used, in the Add step option, the Streaming program was selected, after this one should click in the “Configure and add” button.

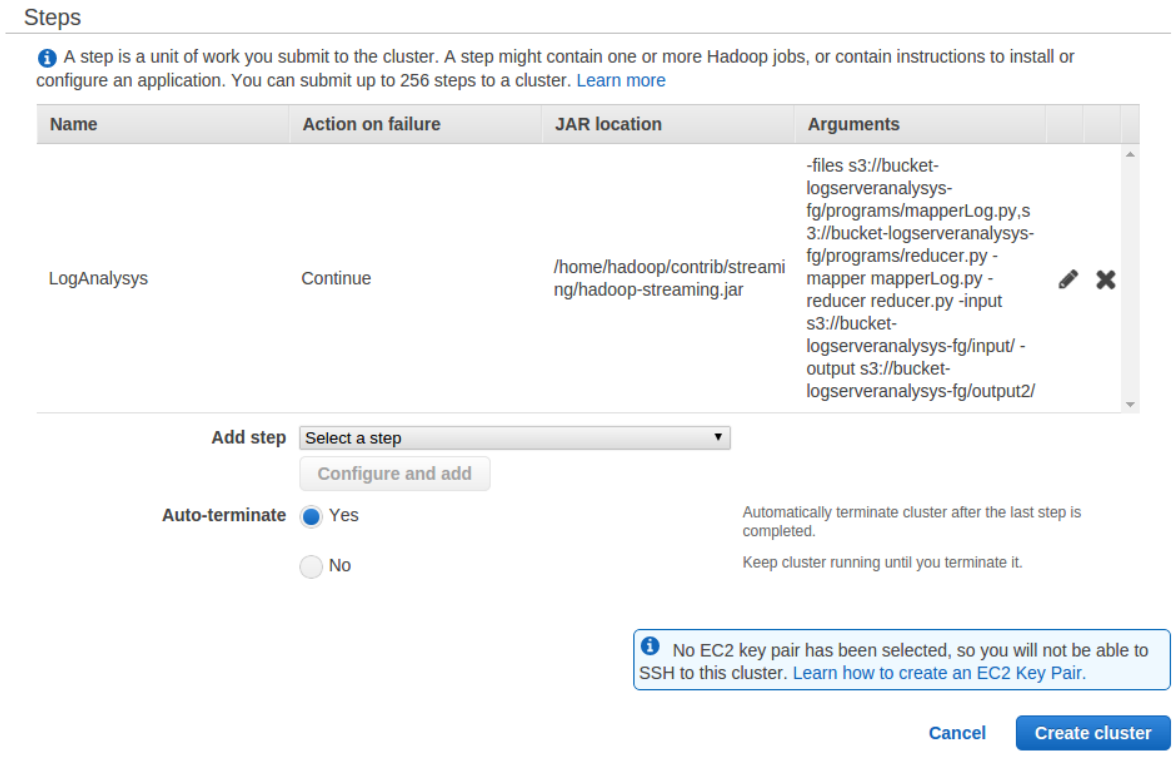
Linking EMR with S3: Select the scripts of the map and reducer and defining input and output directories

The “configure and add” button results in the following screen:



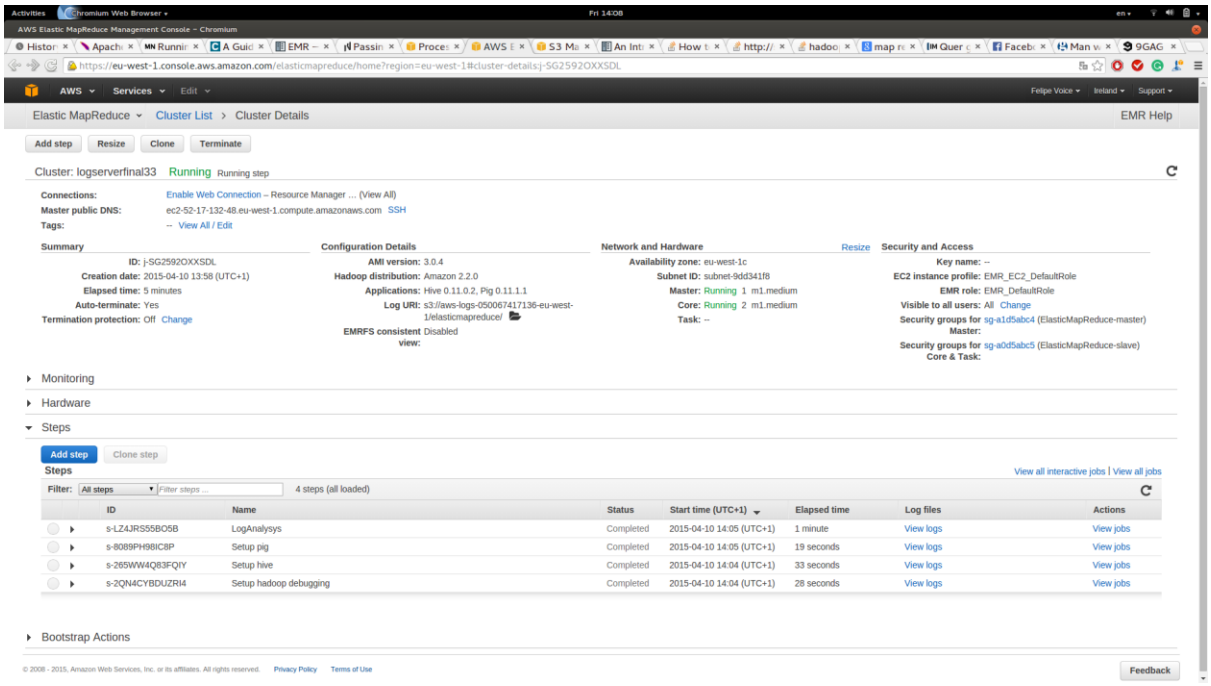
In this step, the user should select the map and reduce programs of the streaming job to be executed in the Hadoop cluster. The input and output directory must be selected as well. The output directory must be an inexistent directory that will be automatically created.

After click in save, one can see the configuration of the Map Reduce steps as shown in the next screenshot. The “create cluster” button initialize the resources and perform the configured steps.



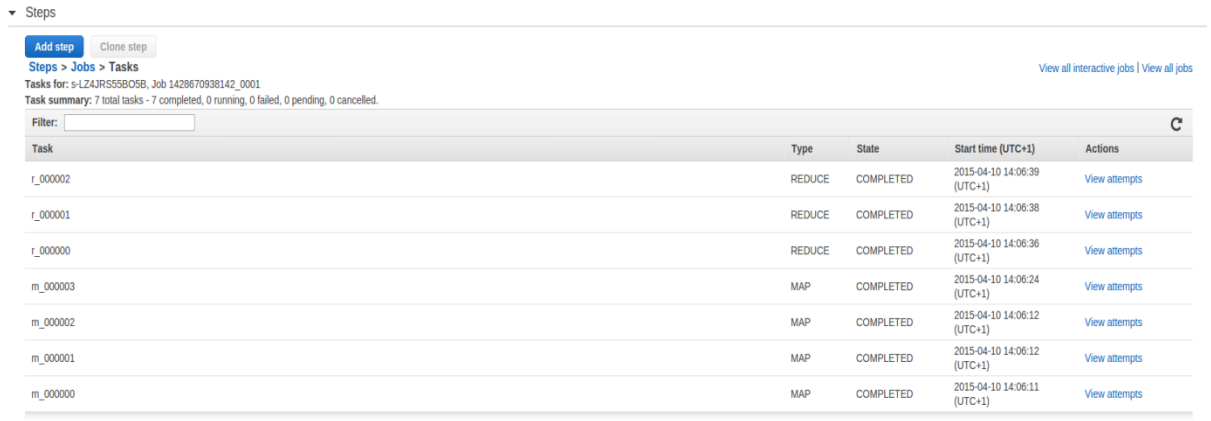
Cluster Running

The user can monitor the cluster by the Amazon Cluster Details option of the EMR.



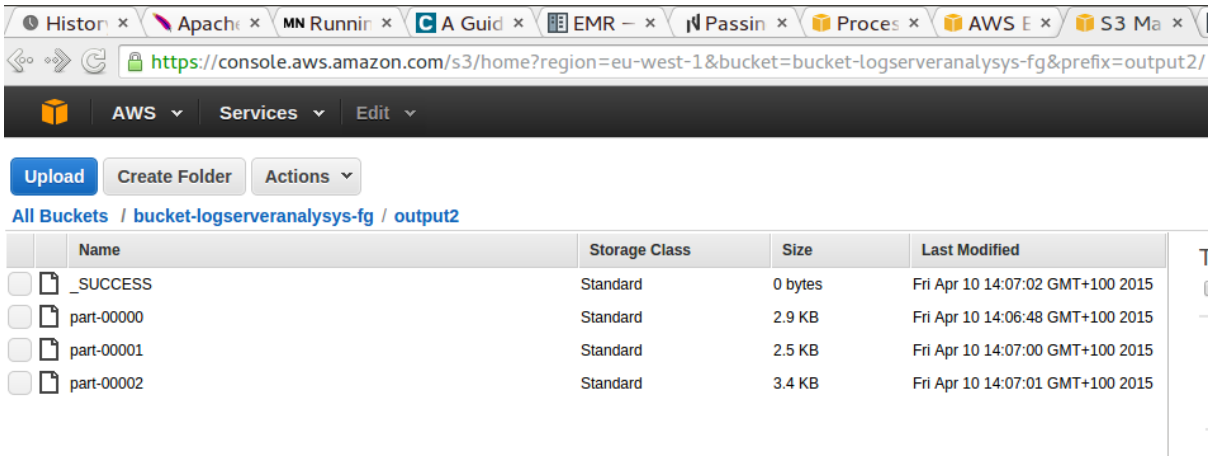
Task Detail

Also, it is possible to see the logs and details of the steps of the Map Reduce tasks as shown in the screenshot.



S3 Result

After the completion of the tasks, the user can check the output files in the configured directory, as shown in the screenshot. In this case, these outputs are the result of the log analysis performed by the map and reducer tasks. The results are distributed in files according to the number of reducers that were allocated by the Hadoop cluster.



Combining the results and showing the report

After downloading the output files from the Amazon S3, the files are processed by a combiner in order to show the result in an organized way, as demonstrated in the next screenshot. In the combiner script it is necessary to configure the directory where the files generated by the reducer are located using the variable “dirOutput”.

In this screenshot the logs were filter by date from: 05/03/2004 date to: 05/03/2004. In this range there are just 5 registers. The directory where the output files are stored can be configured in the combiner script. This was used to present the result in a feasible way in this report. The combiner can be run as:

```
./combiner.py
```

```
fgub@fgub-DELL:~/Documents/AptanaStudio3/IpAnalisys/sources$ ./combiner.py
--------------------------------------NUMBER OF CONNECTIONS TO THE SERVER:      5
--------------------------------------NUMBER OF ENTRIES FOR EACH IP:
64.242.88.10    3
prxint-sxb3.e-i.net     2
--------------------------------------LIST OF DISTINCT IPs:
64.242.88.10
prxint-sxb3.e-i.net
fgub@fgub-DELL:~/Documents/AptanaStudio3/IpAnalisys/sources$
```