**Procedure for Local Spark set up**

1. Download Spark on local HDD and extract. You can follow the steps from

<http://spark.apache.org/downloads.html>

and

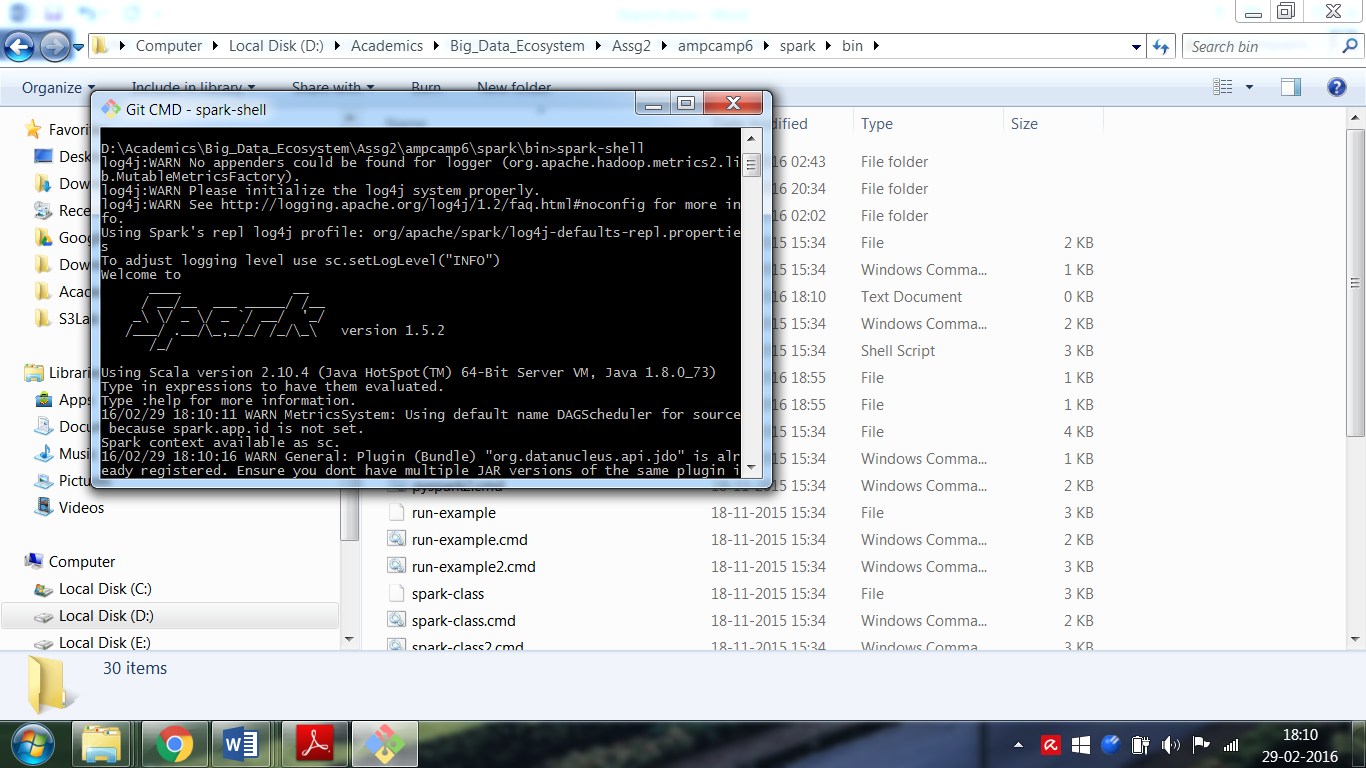
<http://ampcamp.berkeley.edu/6/exercises/getting-started.html>

1. Ensure that JDK has been installed and its path has been set in System Environment Variables

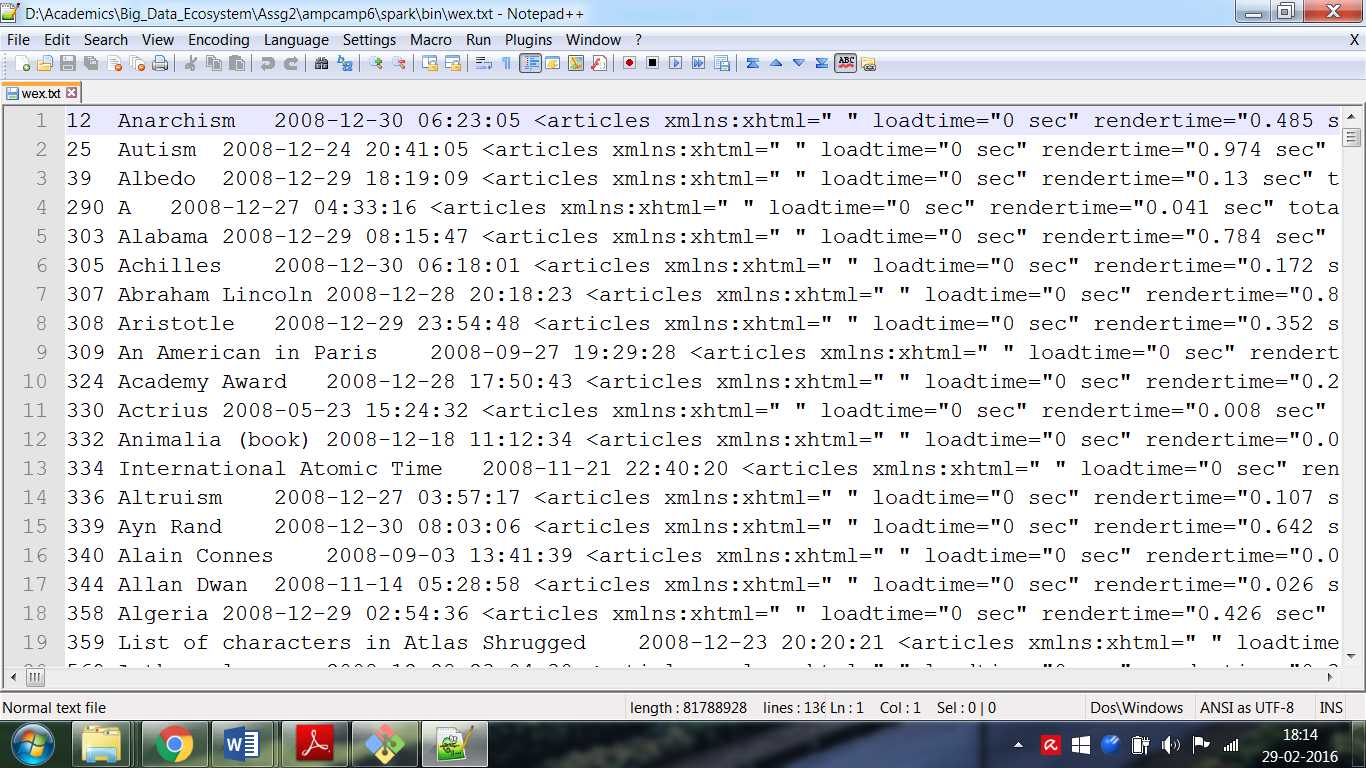
<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

1. Start spark-shell at the Command Prompt. Navigate to the Spark directory, open the bin folder and type :

spark-shell



1. At the scala prompt, type in the source code. Source code can be found in the “Source\_Files” folder. The data file “wex.txt” contains Wikipedia data and should be stored locally. A 78 MB segment of the entire 32 GB file was used.



1. After experimenting the final source code can be written in Notepad++ or Scala IDE in TypeSafe Activator

<https://notepad-plus-plus.org/>

<https://www.lightbend.com/community/core-tools/activator-and-sbt>

1. After the code is written, SBT can be installed from <http://www.scala-sbt.org/download.html>
2. Put Sources in src/main/scala and in the base directory, create empty folder “target” and create an .sbt file simple.sbt in the base directory with following contents:

name := "SparkPageRank100iters"

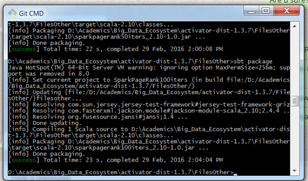
version := "1.0"

scalaVersion := "2.10.5"

libraryDependencies += "org.apache.spark" %% "spark-core" % "1.6.0"

libraryDependencies += "org.apache.spark" %% "spark-graphx" % "1.6.0"

1. Change name each time when building a new source file.
2. Initialize command prompt and enter “*sbt package*”. The .jar files will get generated in the “target” folder



1. This .jar file will be used on EMR (or EC2) to run the jobs.

**Procedure for EC2 Spark set up**

1. Download and unzip (or clone) <https://github.com/amplab/training-scripts/archive/ampcamp3.zip>
2. Navigate to the folder. Export AWS keycodes. Ensure the codes are for US-EAST region only. Also set permissions using chmod 600 the .pem file

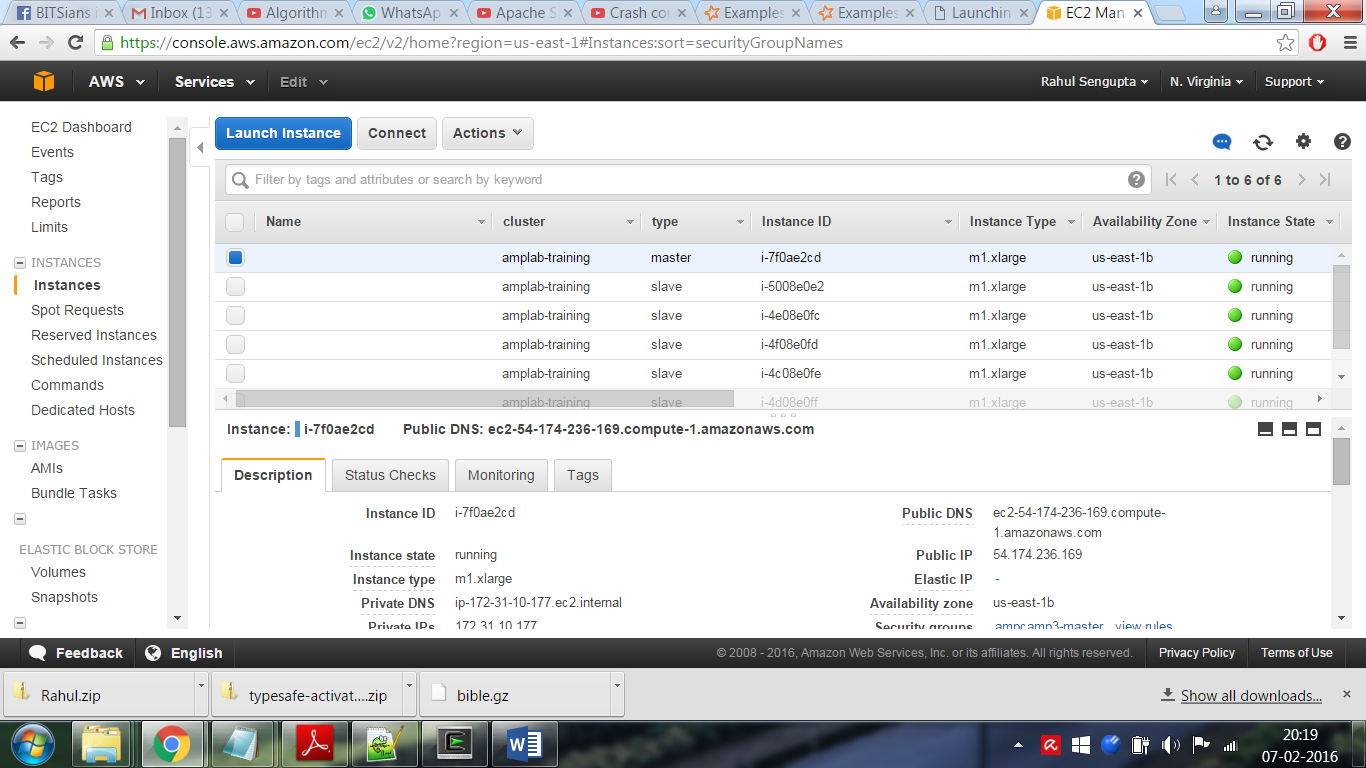
export AWS\_ACCESS\_KEY\_ID=<ACCESS\_KEY\_ID>

export AWS\_SECRET\_ACCESS\_KEY=<SECRET\_ACCESS\_KEY>

1. Execute the following command

./spark-ec2 -i USEAST.pem -k USEAST -s 5 launch amplab-training

This will launch a 5-slave cluster called “amplab-training”. Ensure that the .pem file and the key USEAST are both located in the US-EAST region

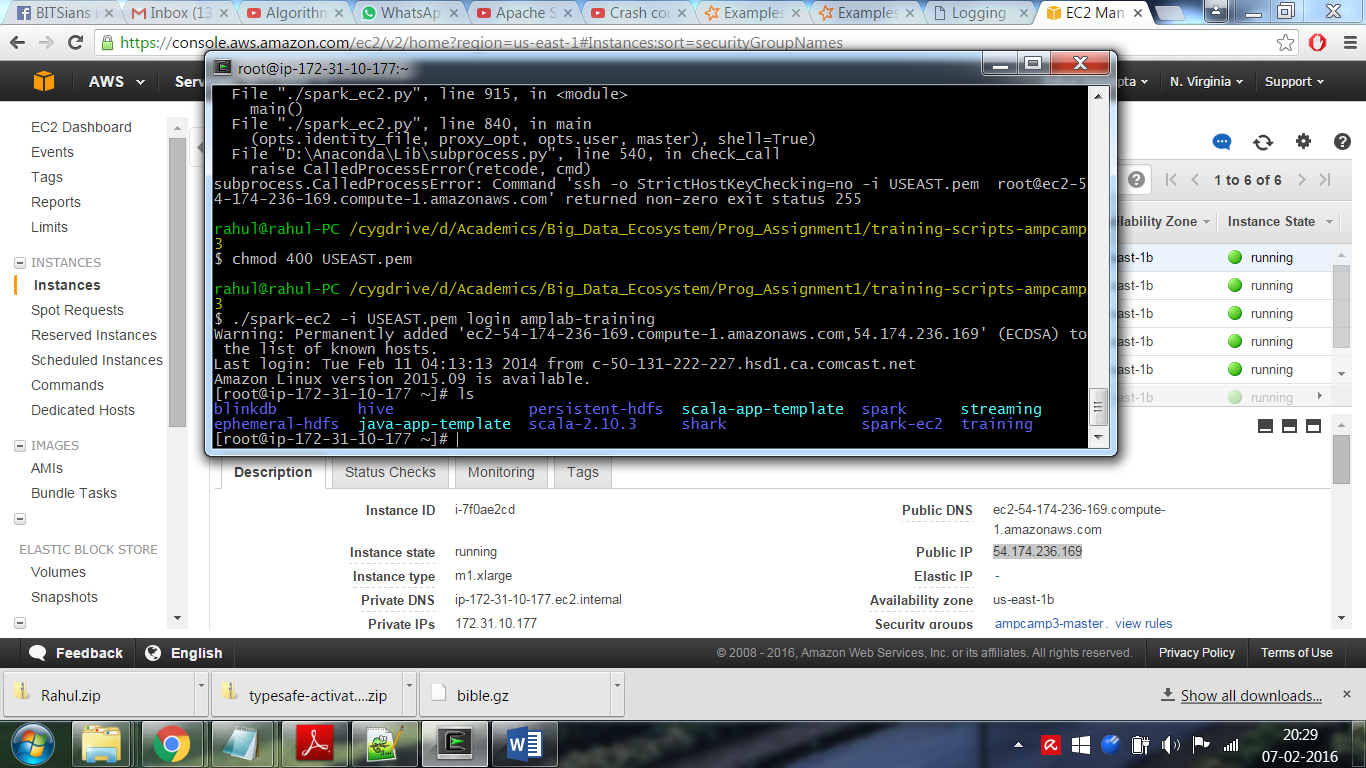


1. Open [https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#](https://console.aws.amazon.com/ec2/v2/home?region=us-east-1)

The above instances should start getting launched.

1. SSH into the master

ssh –I USEAST.pem root@ec2-54-174-236-169.compute-1.amazonaws.com

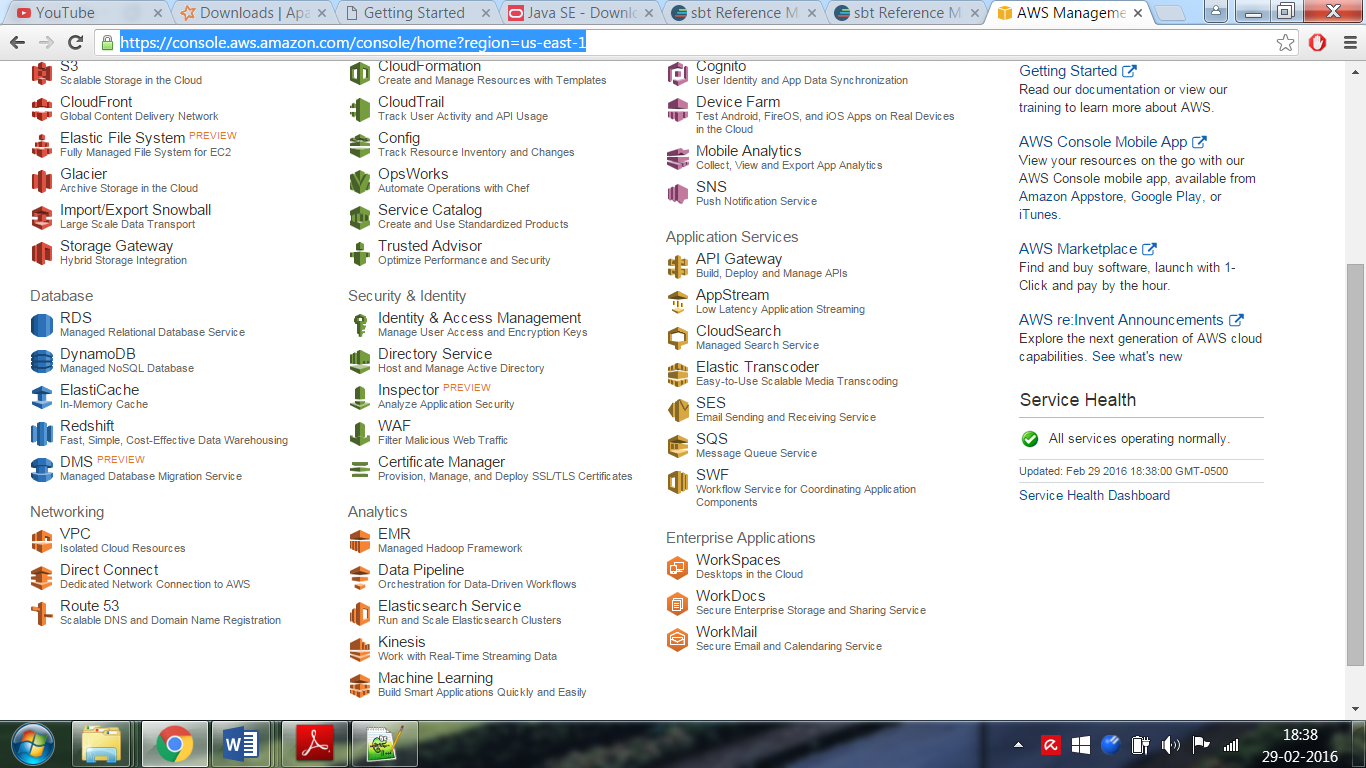


Enter commands in Spark Shell from /root/spark/spark-shell

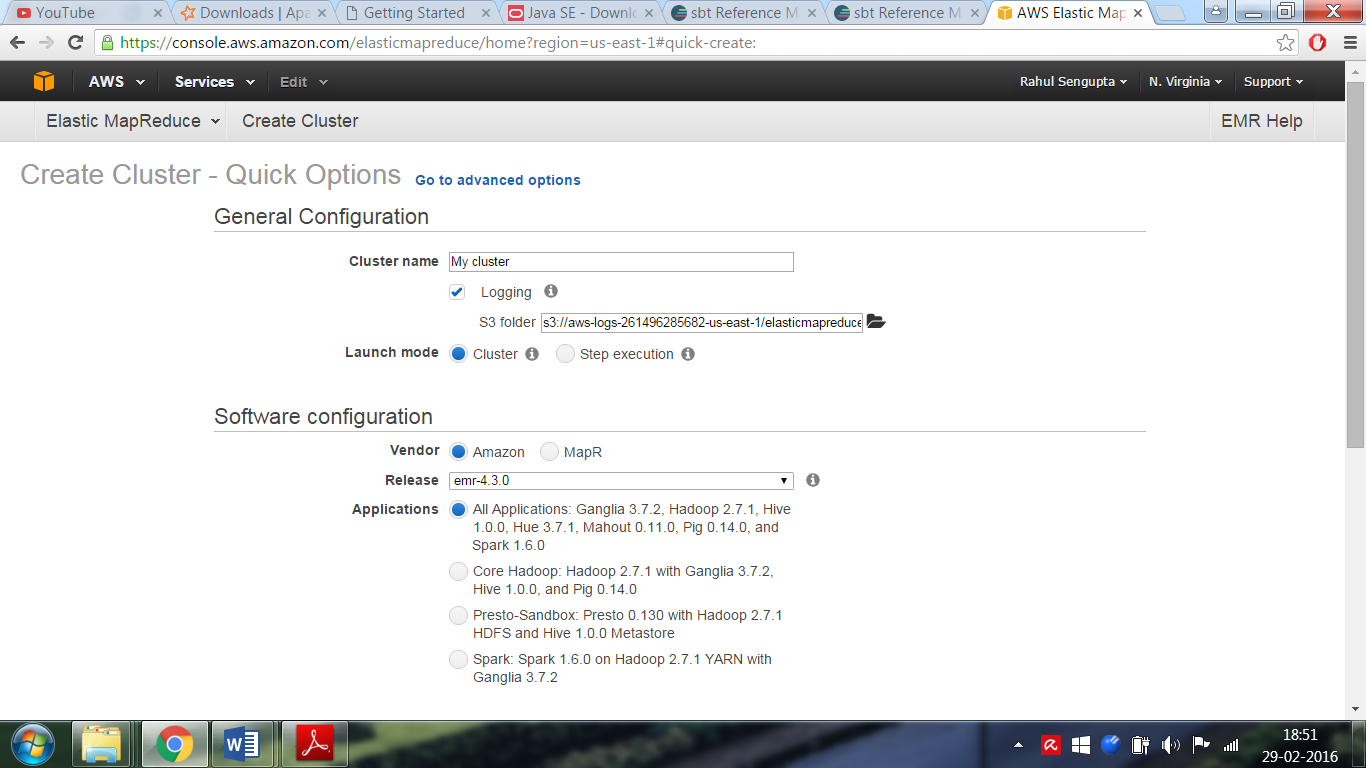
For more detailed tutorial please see <http://ampcamp.berkeley.edu/3/exercises/index.html>

**Procedure for EMR set up**

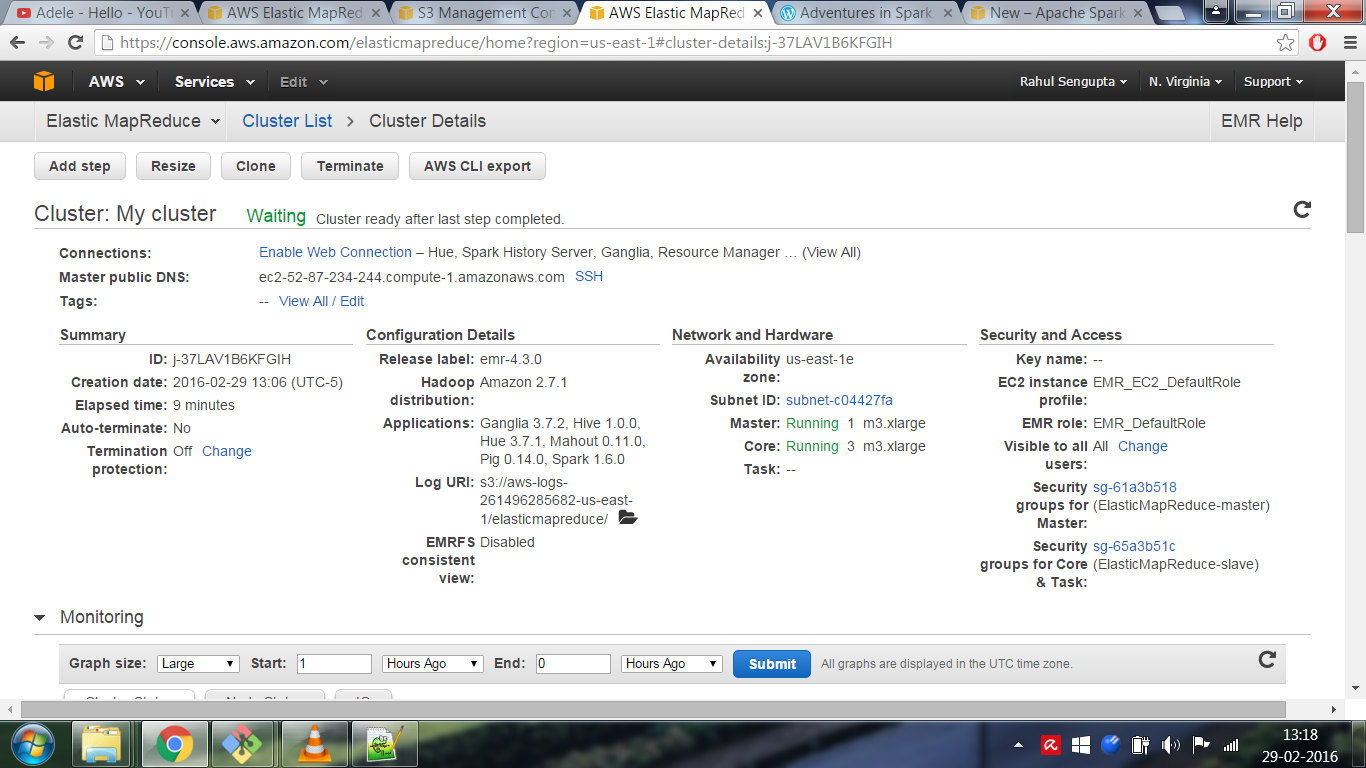
1. Log in to AWS Console <https://console.aws.amazon.com/console/home?region=us-east-1>



1. In EMR window, click on “Create Cluster. Configure the cluster. For this job, 4 Node m3.large cluster was used.

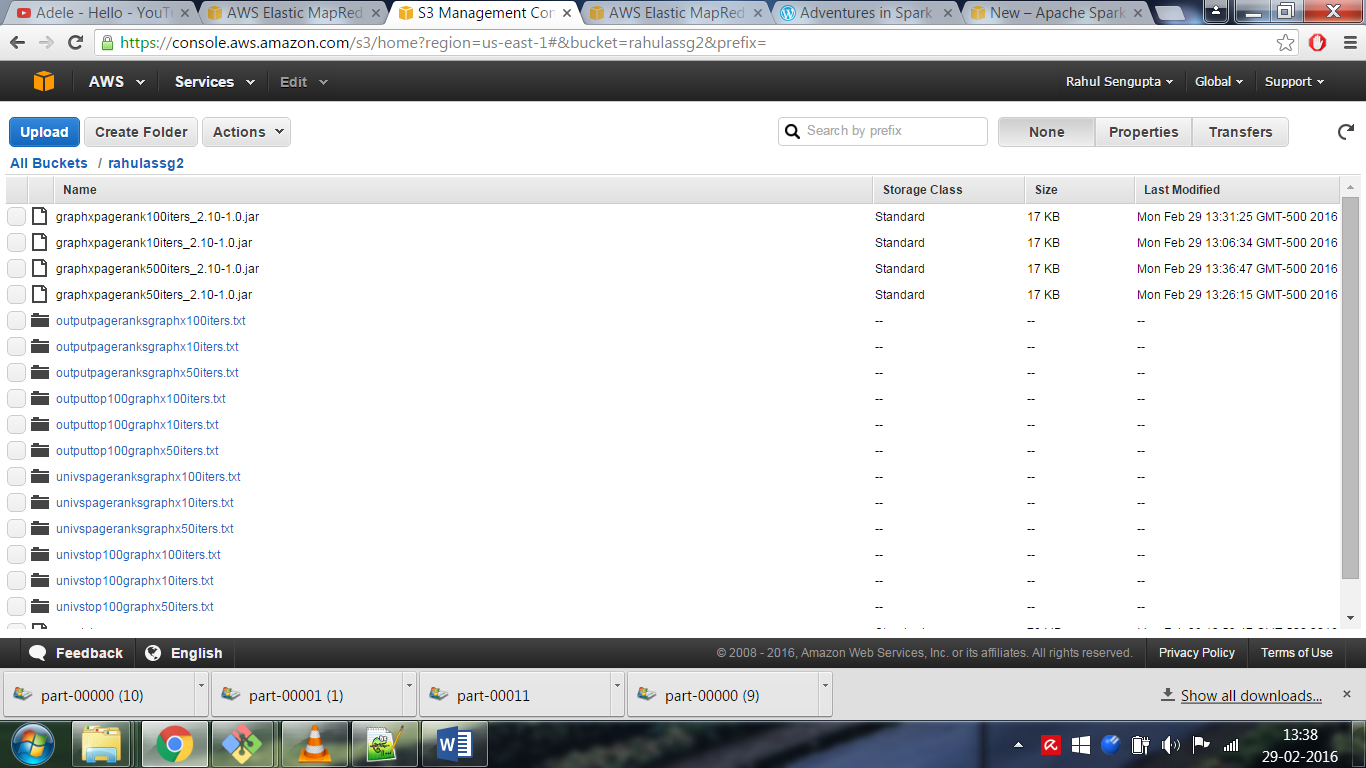


1. Wait for cluster to start



1. A cluster has been created with Master’s address ec2-52-87-234-244.compute-1.amazonaws.com
2. Go to AWS S3 site and create a bucket. Here, bucket “rahulassg2” has been used. Upload the input data file along with the .jar files. The input file can be found at "s3://rahulassg2/wex.txt"

The output files will be written to this bucket.

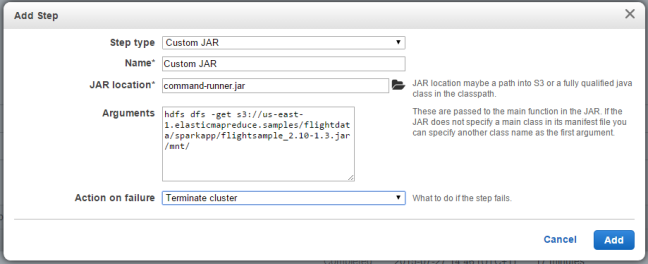


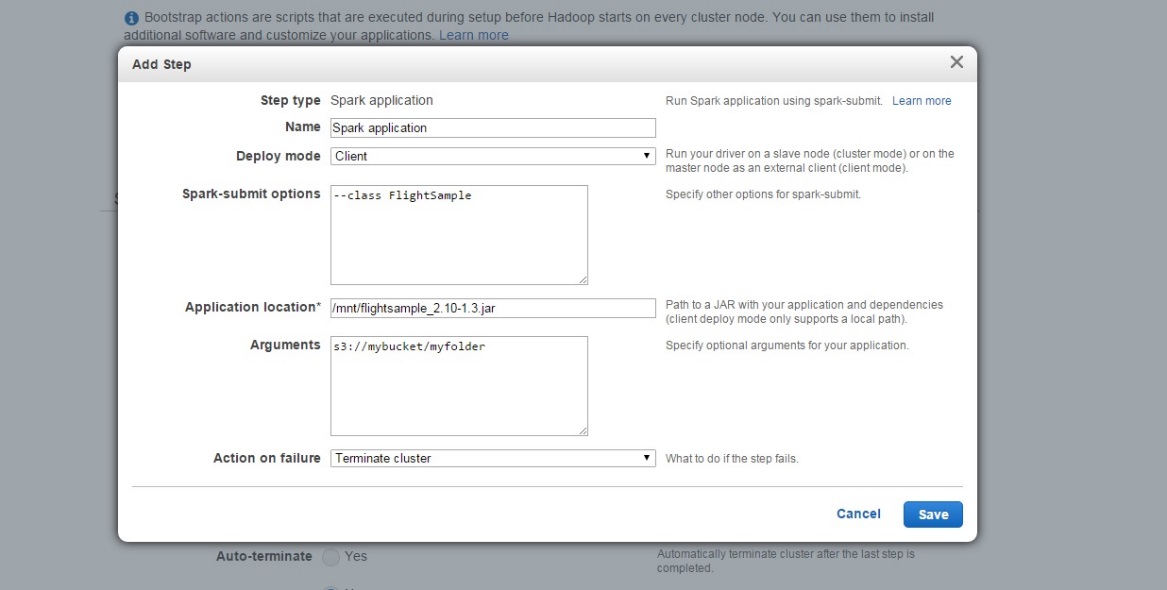
1. Using “Add Step” in the cluster window, add the first step to get the .jar file on to the local HDFS of the Master. The next step would be to actually execute the .jar file.

Detailed instructions given here and here:

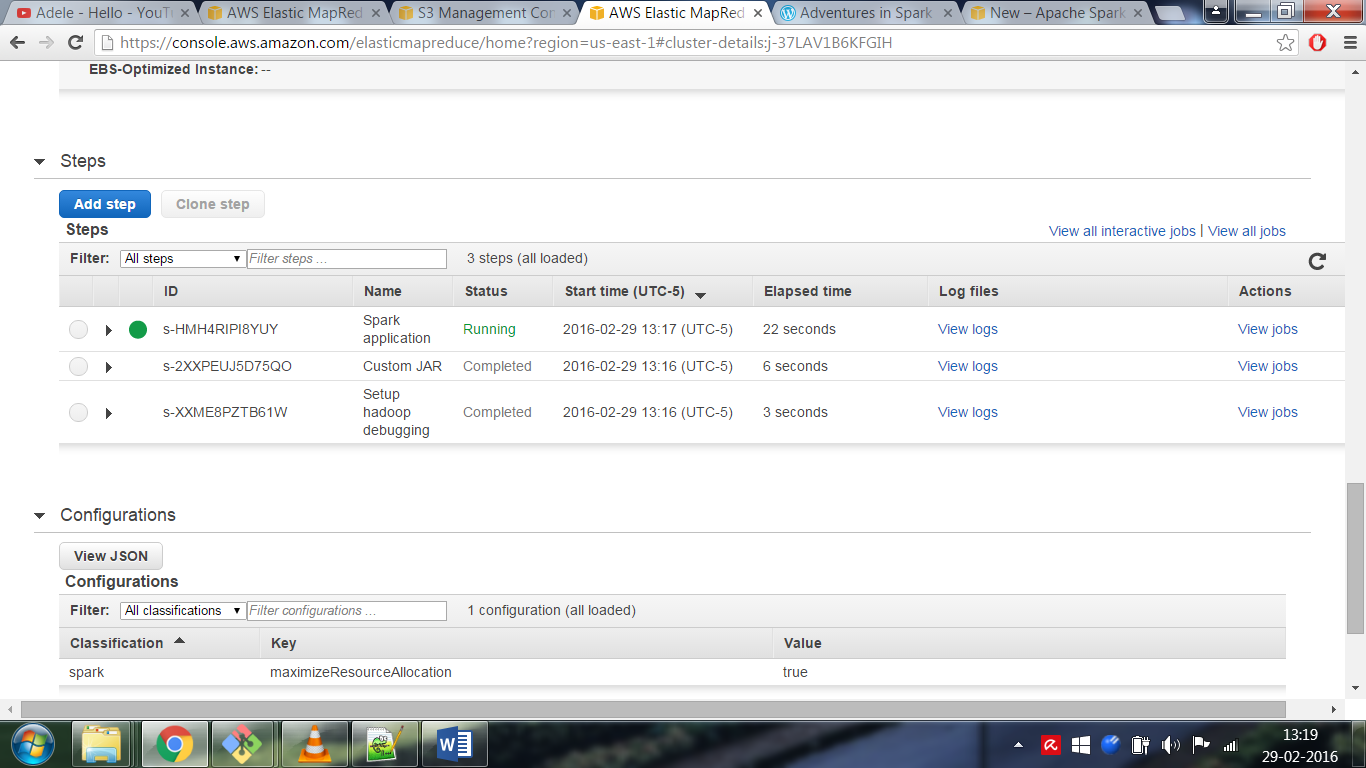
<https://aws.amazon.com/blogs/aws/new-apache-spark-on-amazon-emr/>

<https://controlf1.wordpress.com/2015/08/12/adventures-in-spark-on-elastic-mapreduce-4/>





1. The steps should get executed and the outputs will be saved in the S3 bucket.



1. The jobs were run with the same data using a pure Spark implementation of PageRank and a GraphX implementation. Each were run for 10, 50 and 100 iterations of PageRank algorithm.

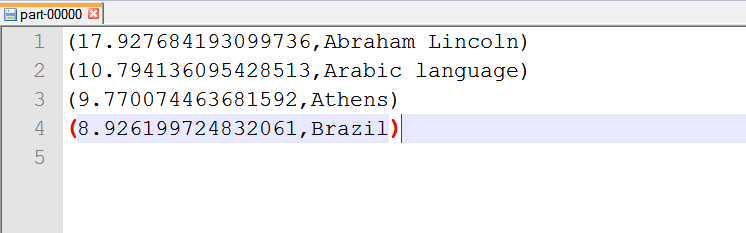
**Outputs**

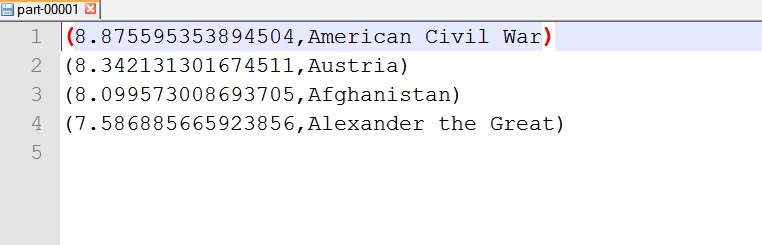
**Note that these results are for only the 78 MB WEX file portion of the overall 32 GB WEX file. Despite repeated attempts and spending several hours of processing, the runs on the 32 GB WEX file did not complete successfully. Due to time and budget constraints, these results are being presented. However, the entire processing pipeline will remain identical even if the 32 GB WEX file was used in lieu of the 78 MB one.**

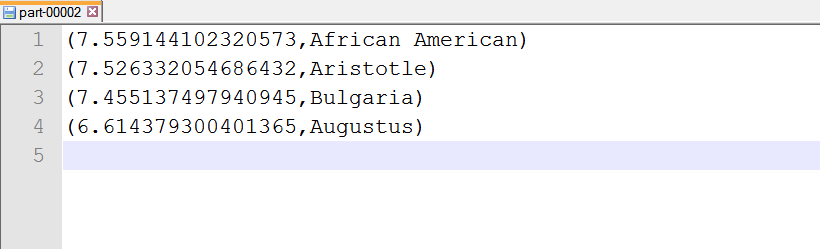
The outputs are saved in two folders “Using\_Pure\_Spark” and “Using\_GraphX”. The file name indicates the following:

* “outputpageranks” indicates the PageRank results for entire data
* “outputtop100” indicates the top 100 entries arranged in decreasing order of PageRank
* “univspageranks” indicates the PageRank results for universities within the data
* “univstop100” indicates the top 100 universities arranged in decreasing order of PageRank
* “xiters” , where “x” indicates how many iterations it was run

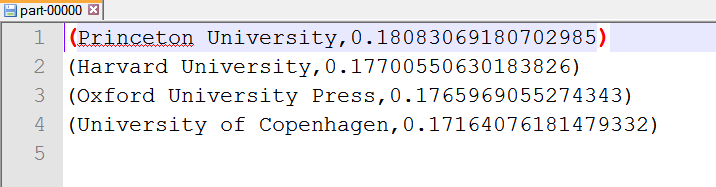
**Sample of Top 100 outputs (Outputs are spread across 25 files) from “outputtop100graphx100iters.txt”**

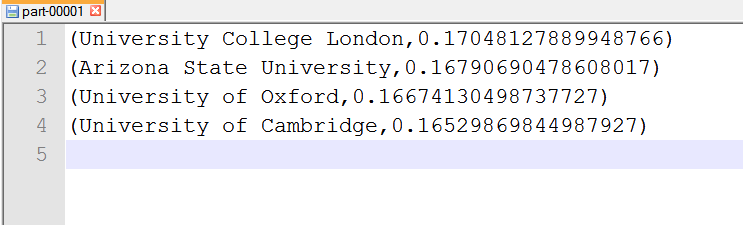


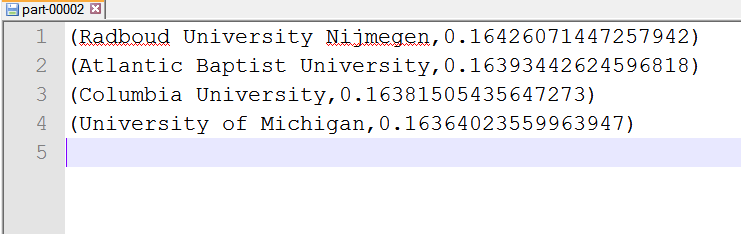




**Sample of Top 100 universities (Outputs are spread across 25 files) from “univstop100spark10iters.txt”**

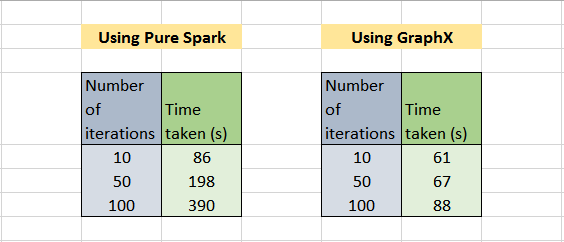


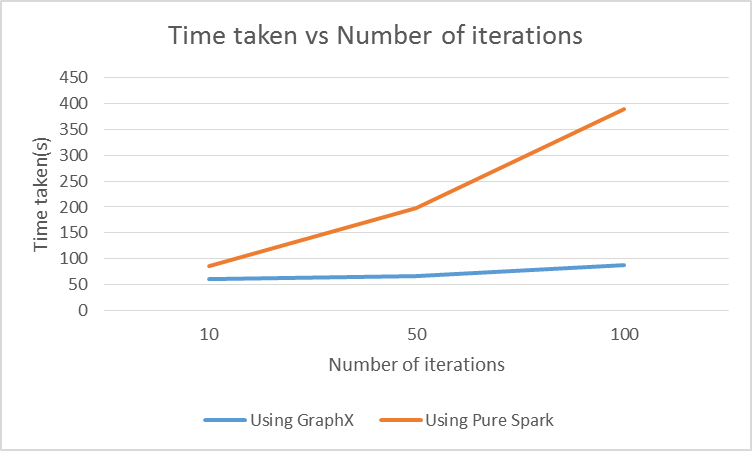




**Performance**

Iteration times using pure Spark vs GraphX





We can clearly see that as the number of iteration grows, the time for execution grows greatly for a naïve Pure Spark implementation but slowly for a GraphX implementation.

**References**

<http://spark.apache.org/>

<http://spark.apache.org/docs/latest/graphx-programming-guide.html>

<http://www.math.cornell.edu/~mec/Winter2009/RalucaRemus/Lecture3/lecture3.html>

<http://ampcamp.berkeley.edu/>

<https://github.com/apache/spark/tree/master/examples/src/main/java/org/apache/spark/examples>

<https://github.com/apache/spark/blob/master/graphx/src/main/scala/org/apache/spark/graphx/lib/PageRank.scala>

<https://yunmingzhang.wordpress.com/2015/04/24/how-does-pagerank-work-in-graphx-for-spark/>

<https://aws.amazon.com/blogs/aws/new-apache-spark-on-amazon-emr/>

<https://controlf1.wordpress.com/2015/08/12/adventures-in-spark-on-elastic-mapreduce-4/>

<http://blog.argteam.com/coding/university-ranking-wikipedia/>

<http://ilpubs.stanford.edu:8090/573/1/2002-6.pdf>