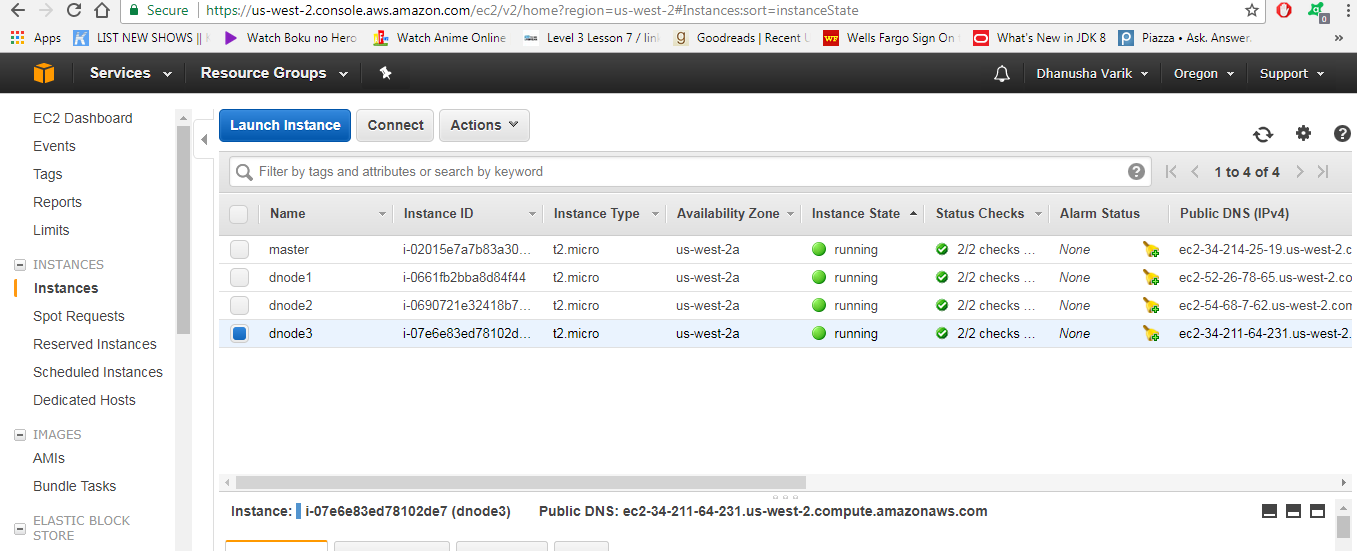
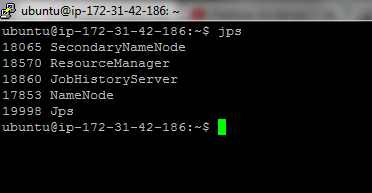
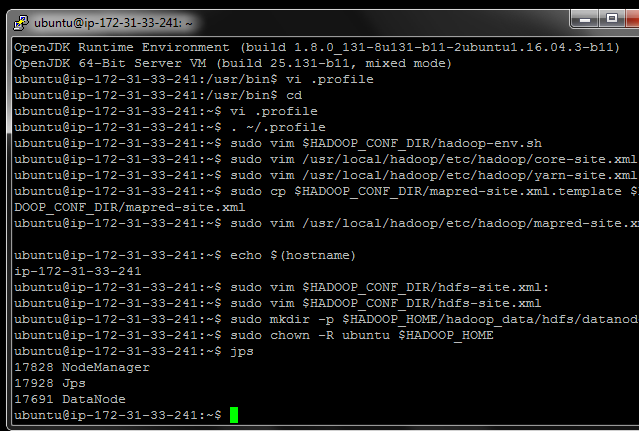
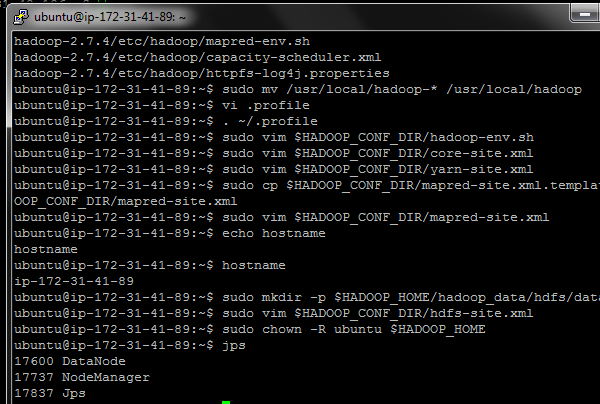
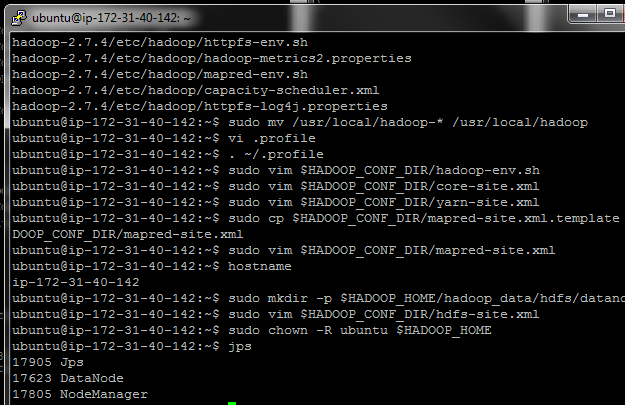
# Assignment 1 readme

1. Launched 4 AWS EC2 instances and set them up with Hadoop (1 namenode, 3 datanodes).  
     
   

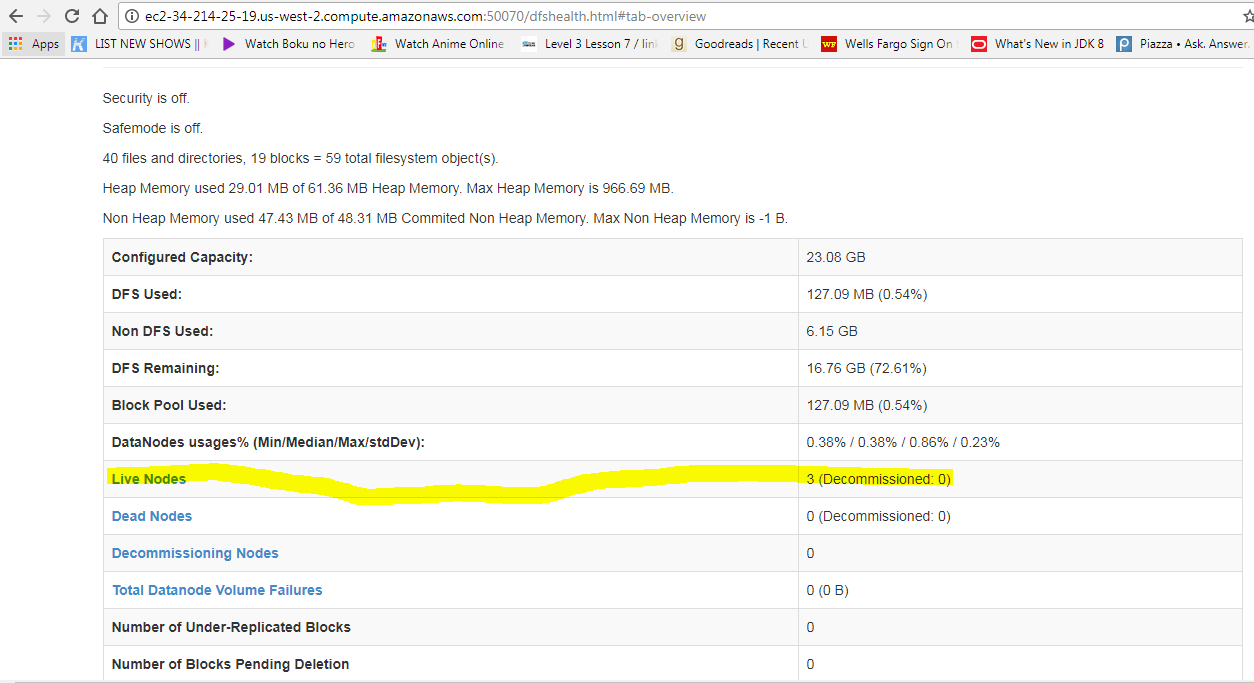
Namenode jps output:  
  


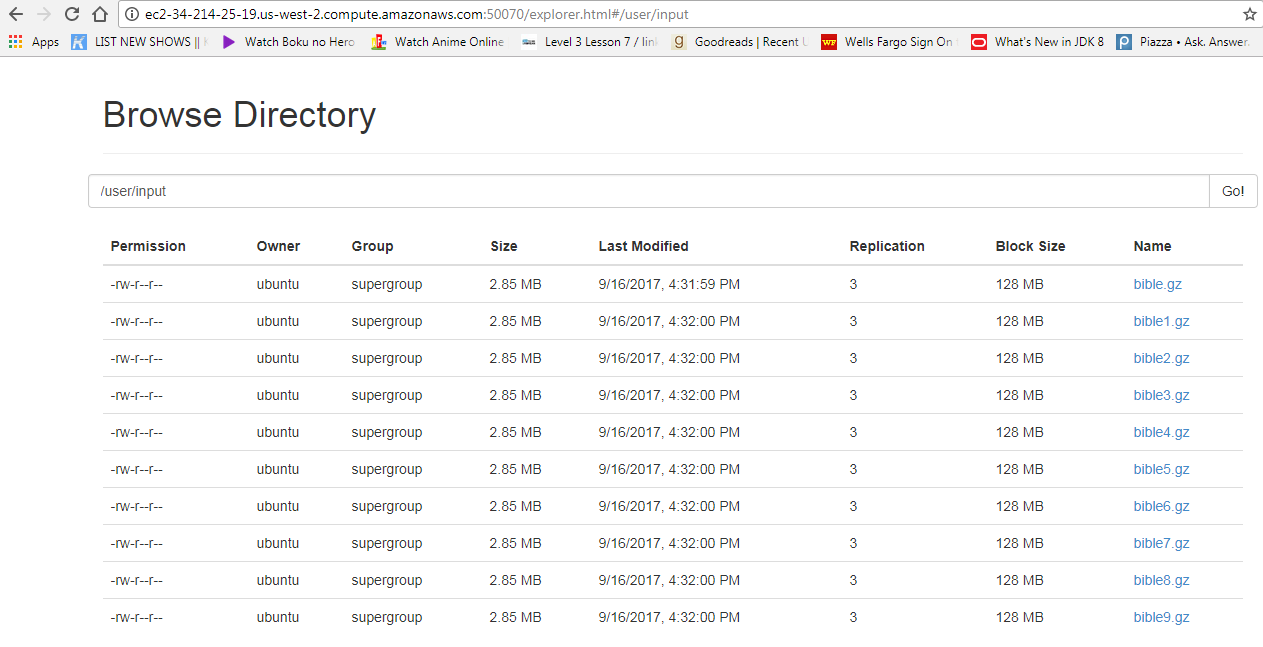
Datanode1 jps output:  
  


Datanode2 jps output:  
  


Datanode3 jps output:  
  


All instances were up and running:

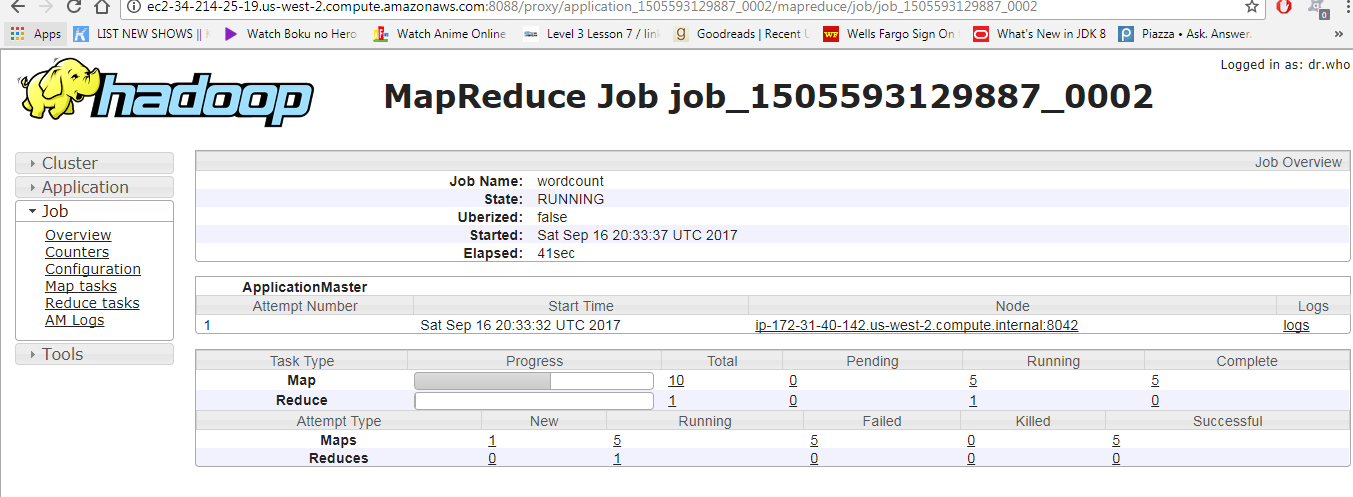


1. Made 10 copies of the input file bible.gz in the hdfs under /user/input folder:  
     
   

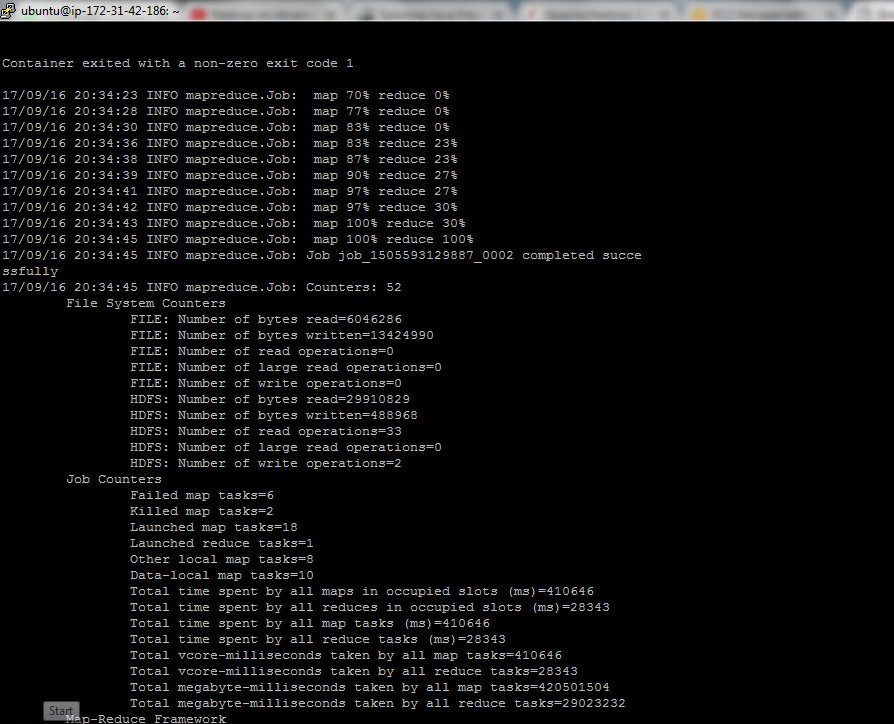
**Task1:**

* Wrote a map reduce program “WordCount.java” using to count one word frequency in the input files.
* The mapper emits all single word tokens from input with a count of 1. Reducer iterates through mapper output and sums the count for each word.
* Created a runnable task1.jar and ran the mapreduce job on the namenode as follows:

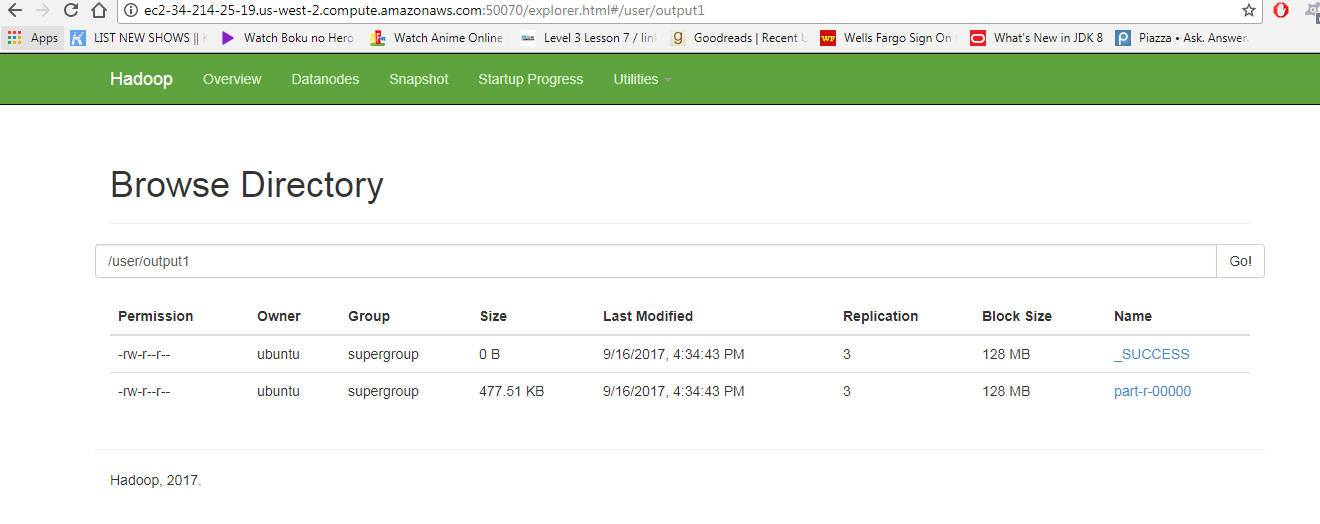
Command: hadoop jar task1.jar /user/input /user/output1

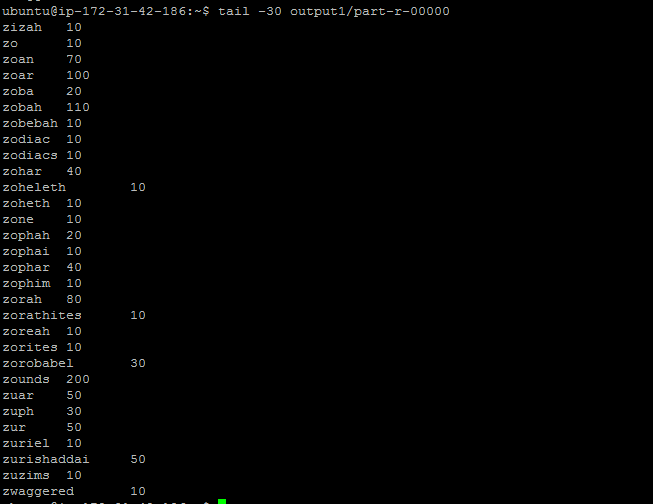
Screenshot of running job, job name ‘wordcount’:

Job completed successfully:



Output1 folder generated with part file:

  
Screenshot of the last 30 lines of output file shows each word along with its count:

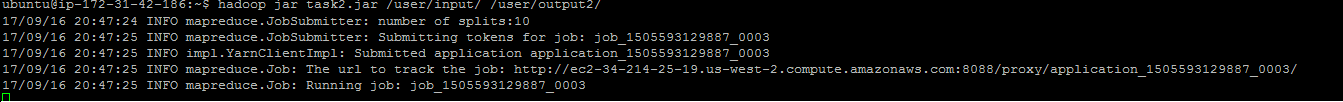


* Link to output data: <https://github.com/dvarik/EC2-HadoopMapRed-Assignment/tree/master/task1-output>

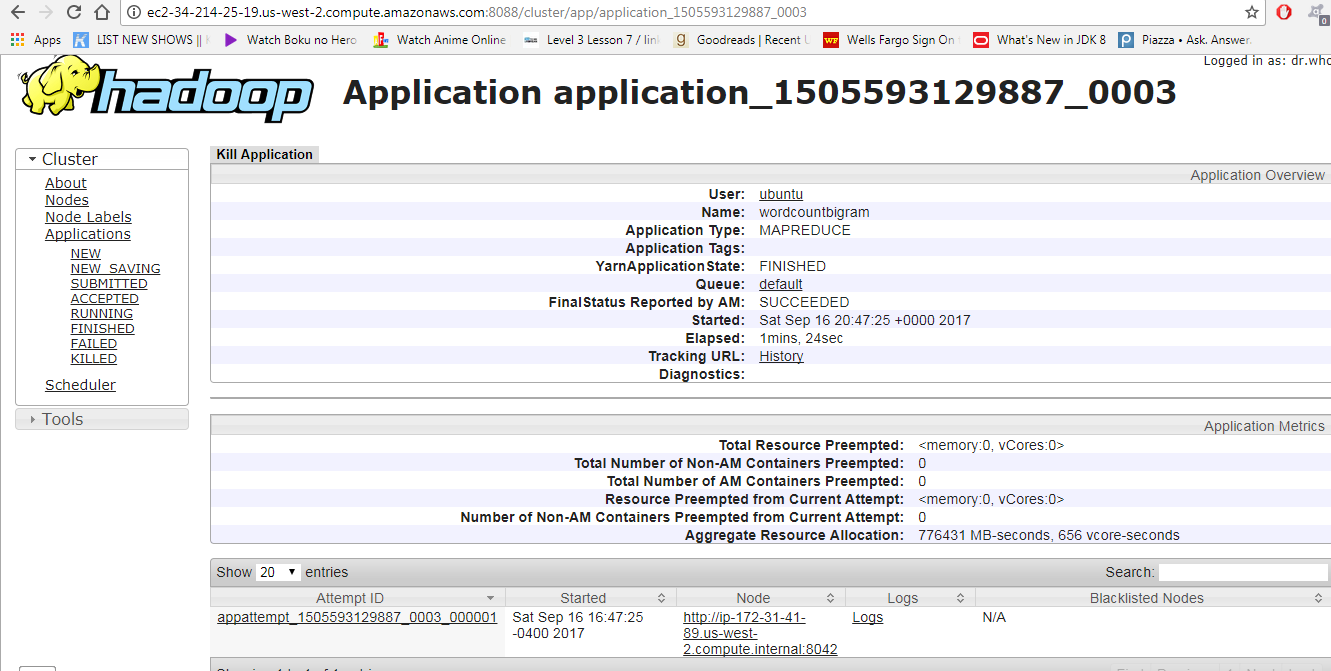
**Task2:**

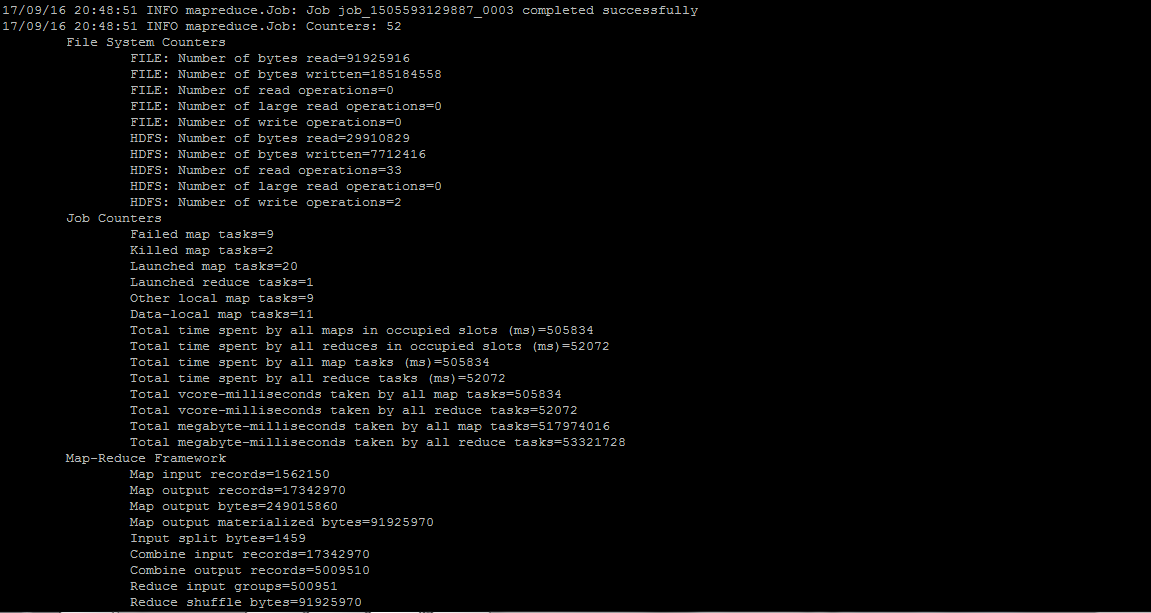
* Wrote a map reduce program “WordCountBigram.java” to count double-word or bigram frequency in the input files.
* The mapper saves previous word into a class variable and emits previous word + current word as a single (space separated) key with a count of 1. Reducer iterates through mapper output and sums the count for each bigram.
* Created a runnable task2.jar and ran the mapreduce job on the namenode as follows:

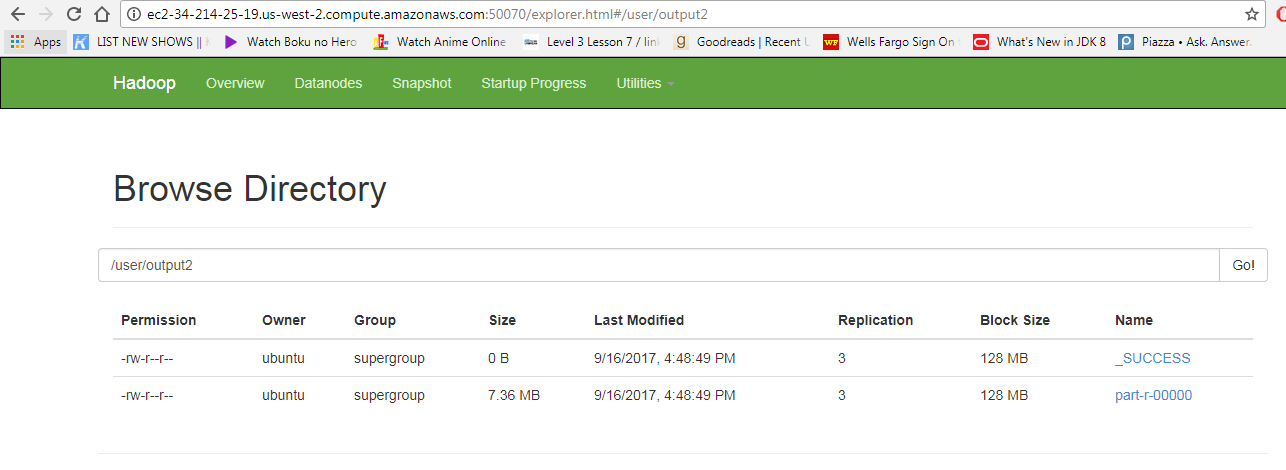
Command: hadoop jar task2.jar /user/input /user/output2



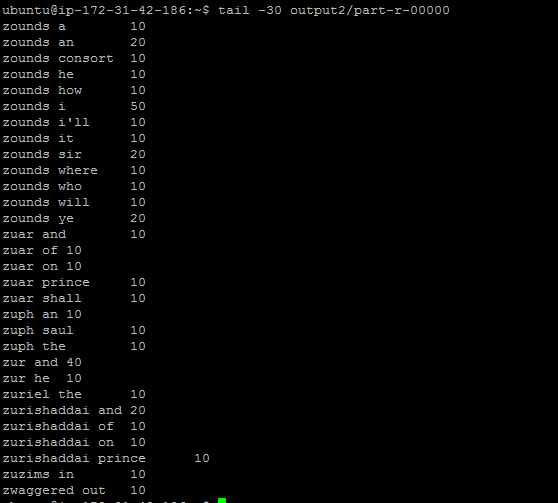
Screenshot of job, job name “wordcountbigram”:



Job successful completion:  
  


Output generated in output2 folder:  
  


Screenshot of last 30 lines of output file shows 2-words along with their counts:

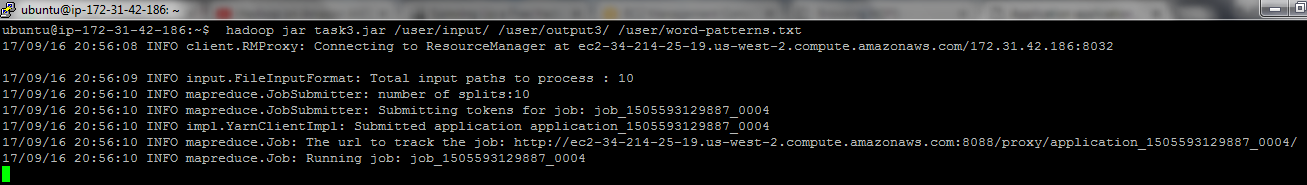


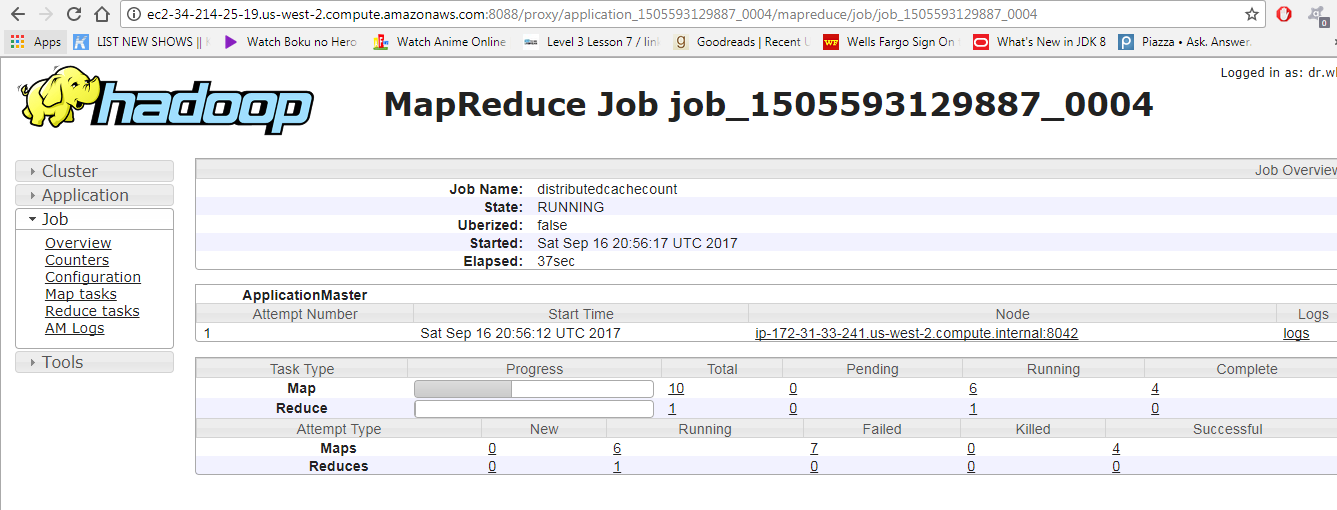
* Link to output: <https://github.com/dvarik/EC2-HadoopMapRed-Assignment/tree/master/task2-output>

**Task3:**

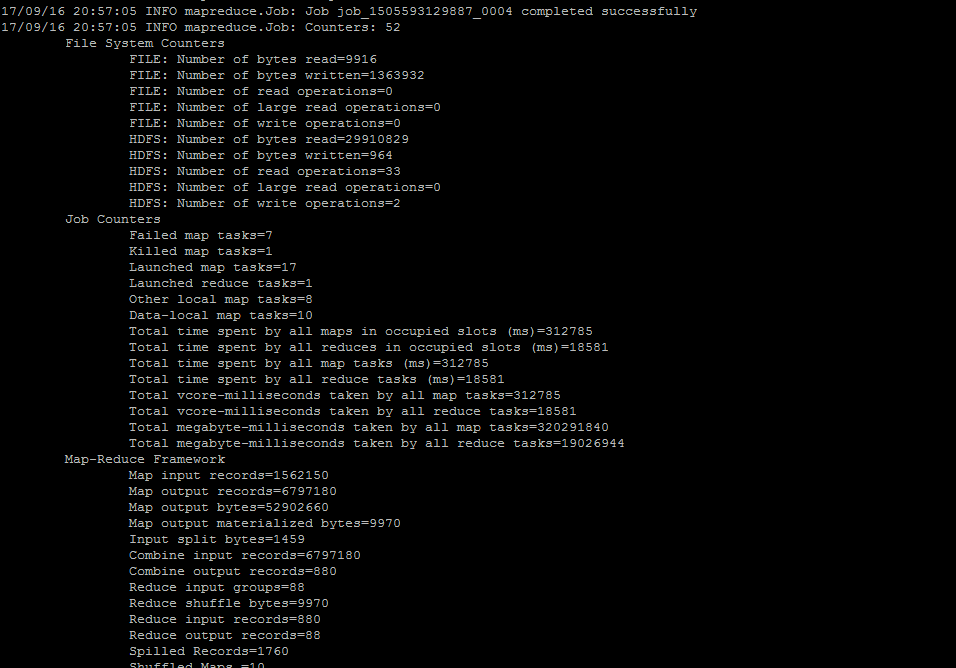
* Wrote a program “DistributedCacheCount.java” to count one word frequency of another given list, “word-patterns.txt”, in the given input file bible.gz.
* The word patterns text file is taken in as the 3rd argument and passed to the map reduce job as a distributed cache file using the code: job.addCacheFile(new Path(args[2]).toUri());
* The mapper overrides a “setup” method where it accesses the cache file and reads the words in it and saves them into a set “patternWords” which is class variable inside the mapper.
* The “map” method in the mapper then checks each word token form input file and if it is also present in the “patternWords” set, it emits that word with a count of 1. Reducer iterates through mapper output and sums the count for each such word.
* Created a runnable task3.jar and ran the mapreduce job on the namenode as follows:

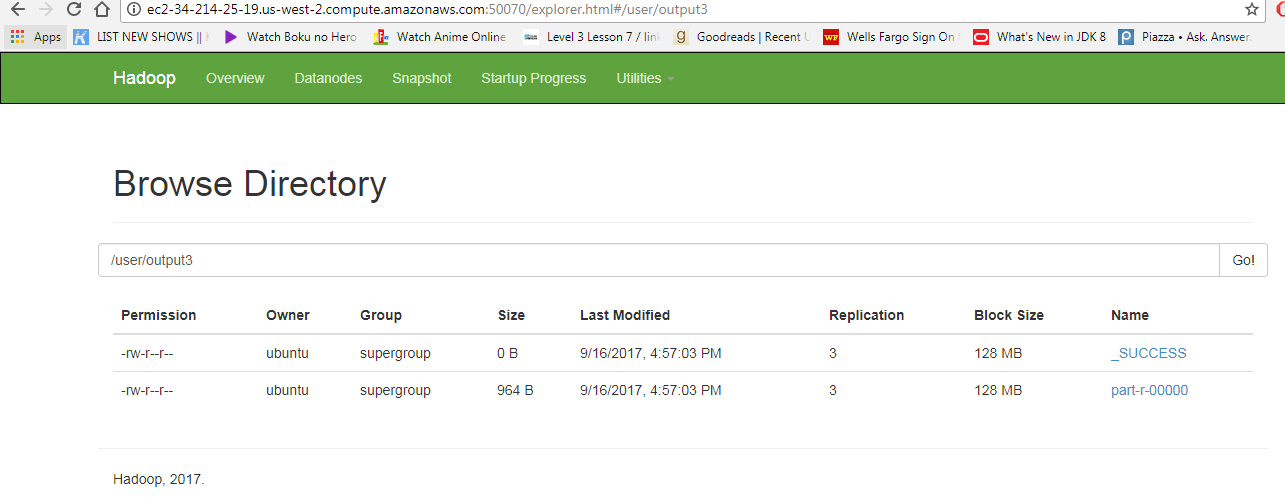
Command: hadoop jar task3.jar /user/input /user/output3 /user/word-patterns.txt

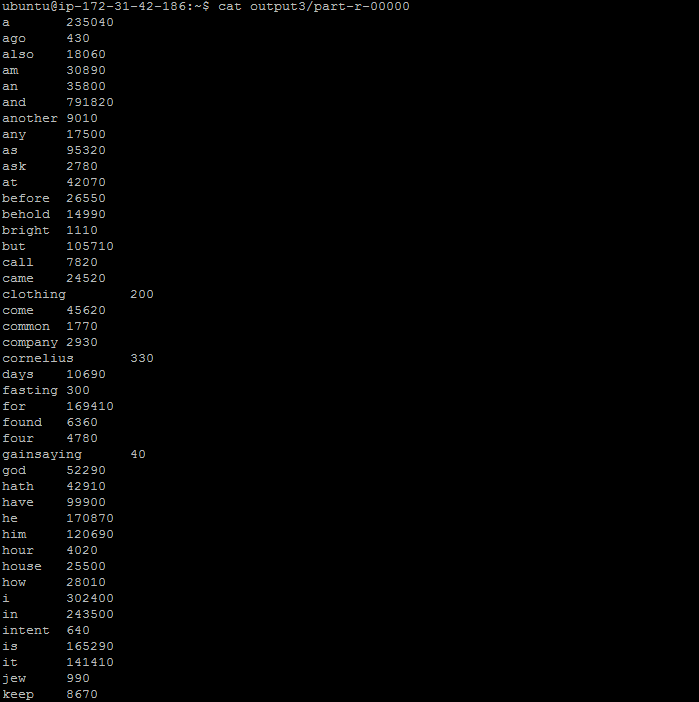


Screenshot of running job, job name “distributedcachecount”:  
  


Job completed successfully:



Output generated in output3 folder:  
  


Screenshot of output file shows counts of words from pattern file in bible.gz file:  
  


* Link to output: <https://github.com/dvarik/EC2-HadoopMapRed-Assignment/tree/master/task3-output>

Also screenshot of application manager showing all 3 mapreduce jobs along with their details:  
