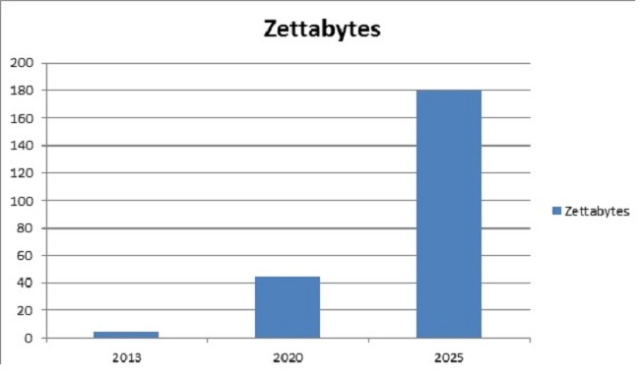
The quantity of digital data generated every day is growing exponentially with the advent of Digital Media, Internet of Things among other developments. This scenario has given rise to challenges in creating next generation tools and technologies to store and manipulate these data. This is where Hadoop Streaming comes in! Given below is a graph which depicts the growth of data generated annually in the world from 2013. IDC estimates that the amount of data created annually will reach 180 Zettabytes in 2025!



Source: IDC

IBM states that, every day, almost 2.5 quintillion bytes of data are created, with 90 percent of world’s data created in the last two years! It is a challenging task to store such an expansive amount of data. Hadoop can handle large volumes of structured and unstructured data more efficiently than the traditional enterprise Data Warehouse. It stores these enormous data sets across distributed clusters of computers. Hadoop Streaming uses MapReduce framework which can be used to write applications to process humongous amounts of data.

Since MapReduce framework is based on Java, you might be wondering how a developer can work on it if he/ she does not have experience in Java. Well, developers can write mapper/Reducer application using their preferred language and without having much knowledge of Java, using *Hadoop Streaming* rather than switching to new tools or technologies like Pig and Hive.

**What is Hadoop Streaming?**

Hadoop Streaming is a utility that comes with the Hadoop distribution. It can be used to execute programs for big data analysis. Hadoop streaming can be performed using languages like Python, Java, PHP, Scala, Perl, UNIX, and many more. The utility allows us to create and run Map/Reduce jobs with any executable or script as the mapper and/or the reducer. For example:

$HADOOP\_HOME/bin/hadoop  jar $HADOOP\_HOME/hadoop-streaming.jar \

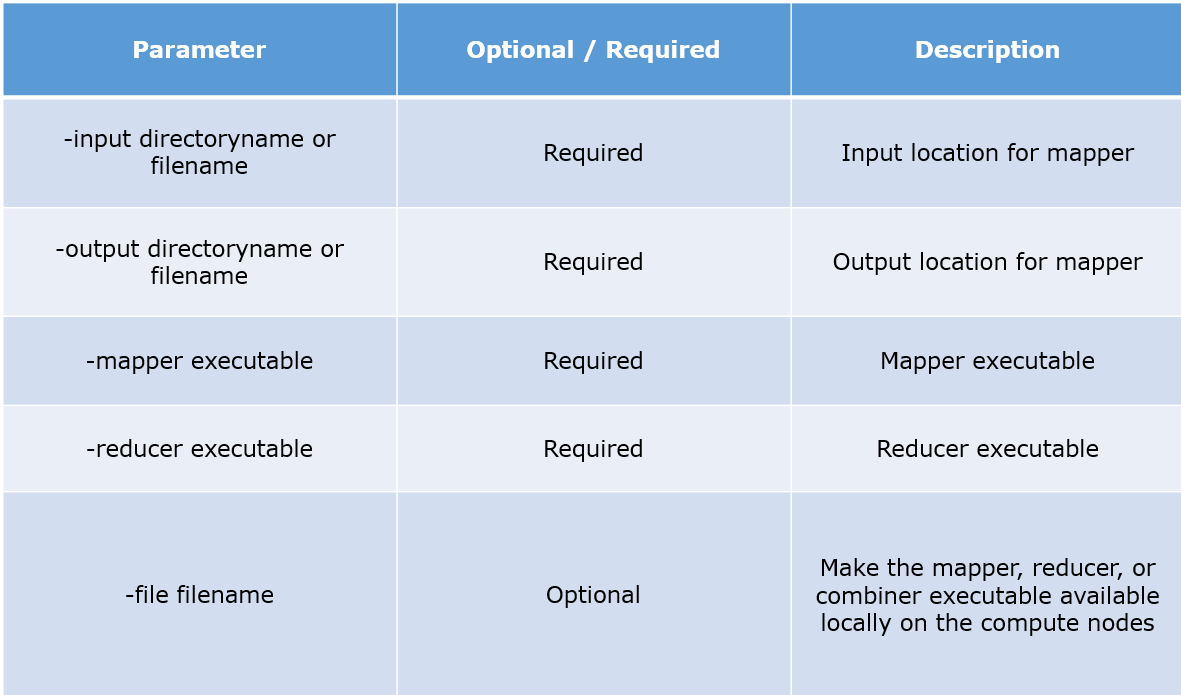
-input myInputDirs \

-output myOutputDir \

-mapper /bin/cat \

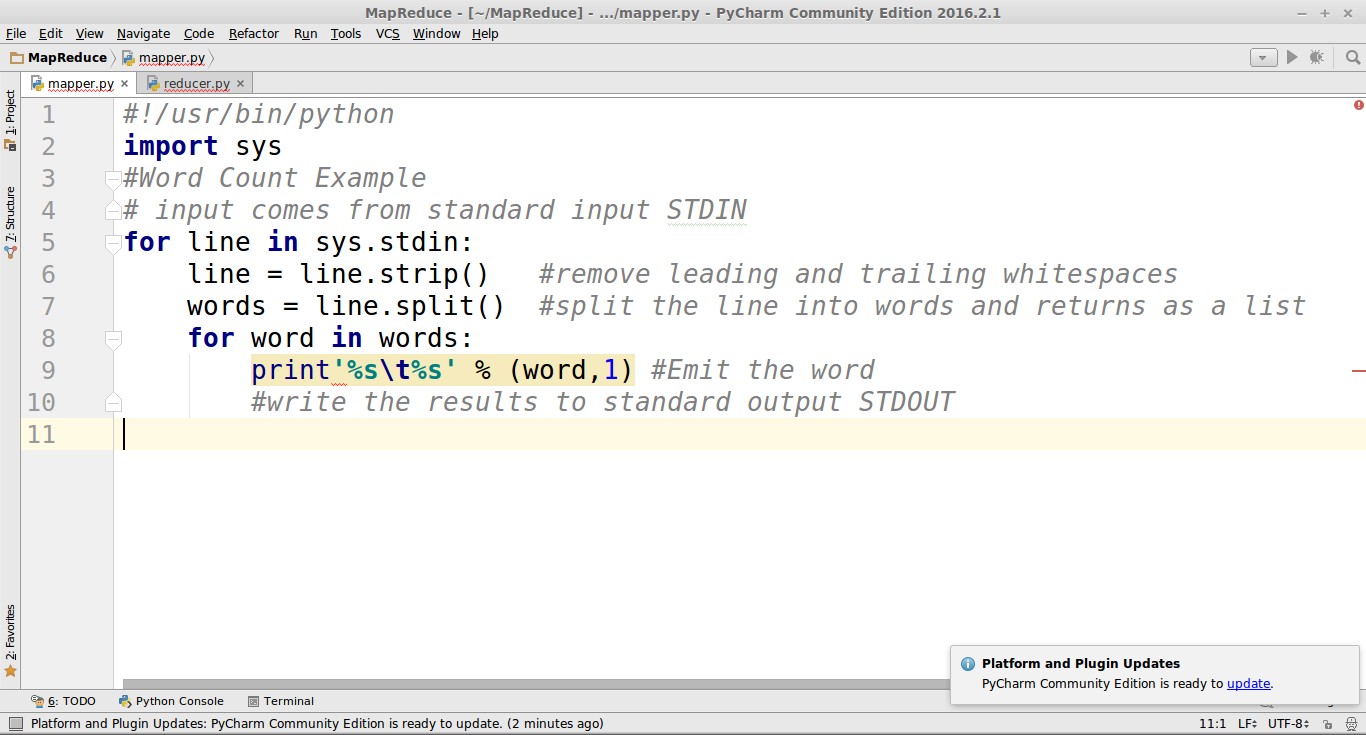
-reducer /bin/wc

**Parameters Description:**



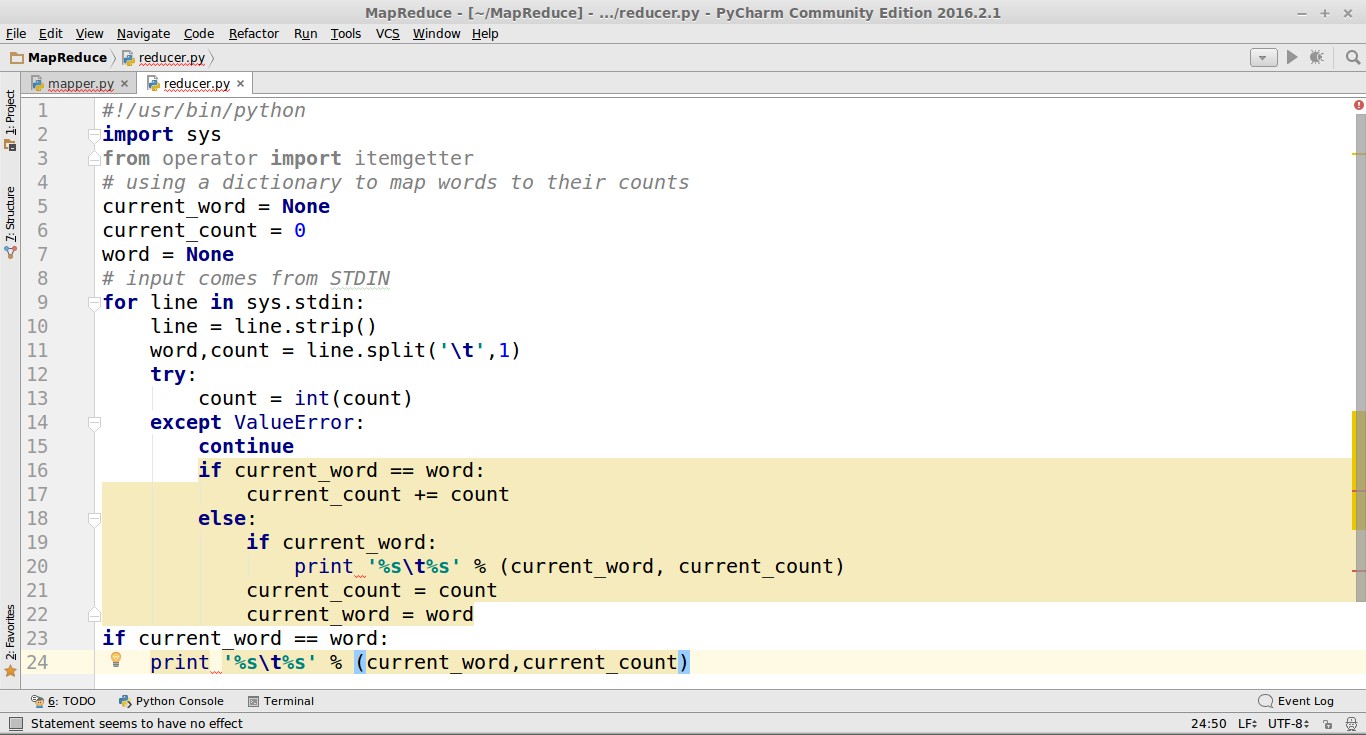
**Python MapReduce Code:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | mapper.py  #!/usr/bin/python  import sys  #Word Count Example  # input comes from standard input STDIN  for line in sys.stdin:  line = line.strip() #remove leading and trailing whitespaces  words = line.split() #split the line into words and returns as a list  for word in words:  #write the results to standard output STDOUT  print'%s\t%s' % (word,1) #Emit the word |



reducer.py

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | #!/usr/bin/python  import sys  from operator import itemgetter  # using a dictionary to map words to their counts  current\_word = None  current\_count = 0  word = None  # input comes from STDIN  for line in sys.stdin:  line = line.strip()  word,count = line.split('\t',1)  try:  count = int(count)  except ValueError:  continue  if current\_word == word:  current\_count += count  else:  if current\_word:  print '%s\t%s' % (current\_word, current\_count)  current\_count = count  current\_word = word  if current\_word == word:  print '%s\t%s' % (current\_word,current\_count) |



**Run:**

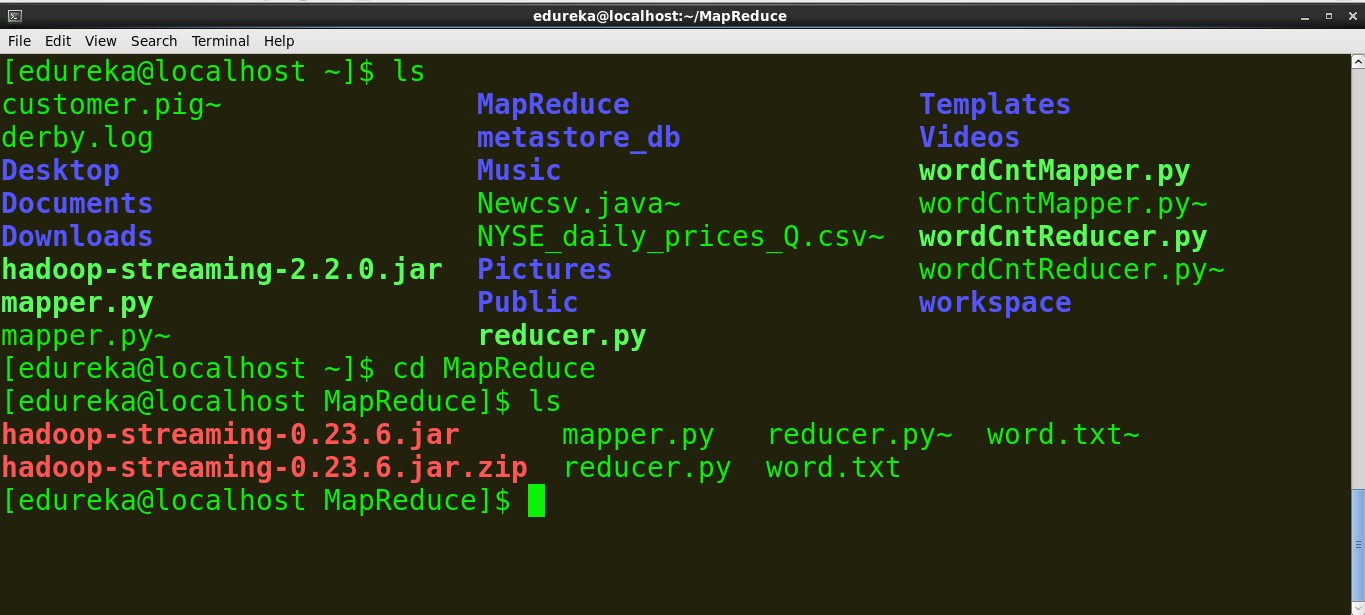
1. Create a file with the following content and name it word.txt.

Cat mouse lion deer Tiger lion Elephant lion deer

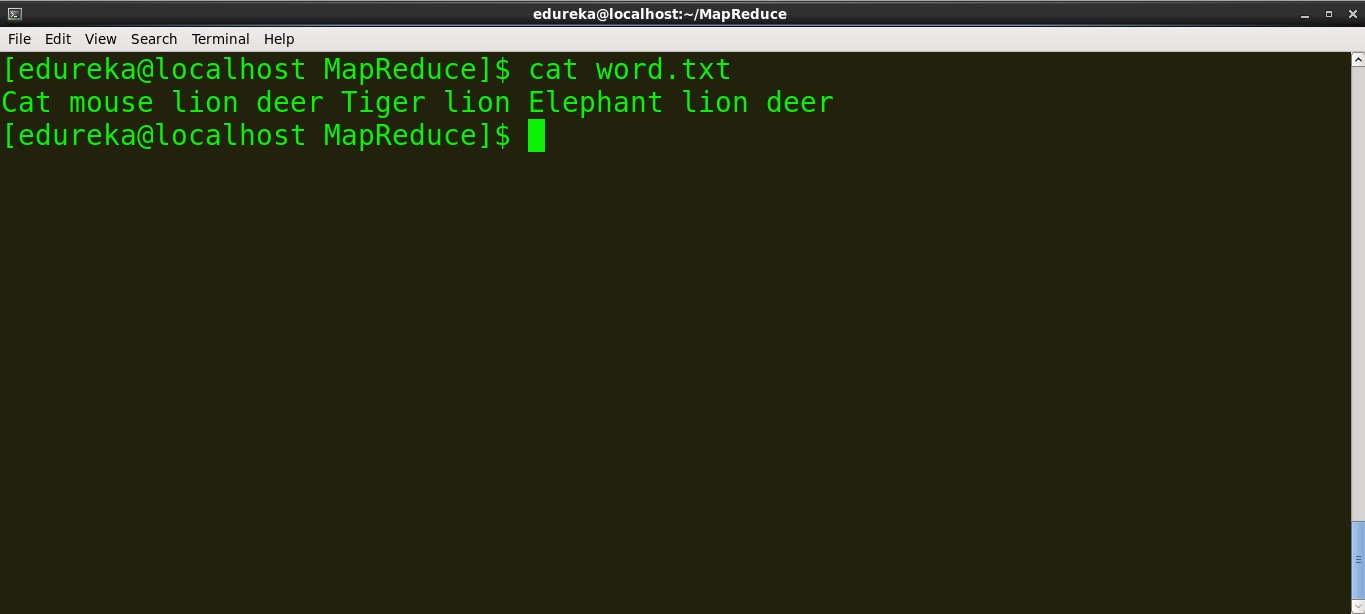
1. Copy the mapper.py and reducer.py scripts to the same folder where the above file exists.



1. Open terminal and  Locate the directory of the file.Command:ls    : to list all files in the directorycd   : to change directory/folder

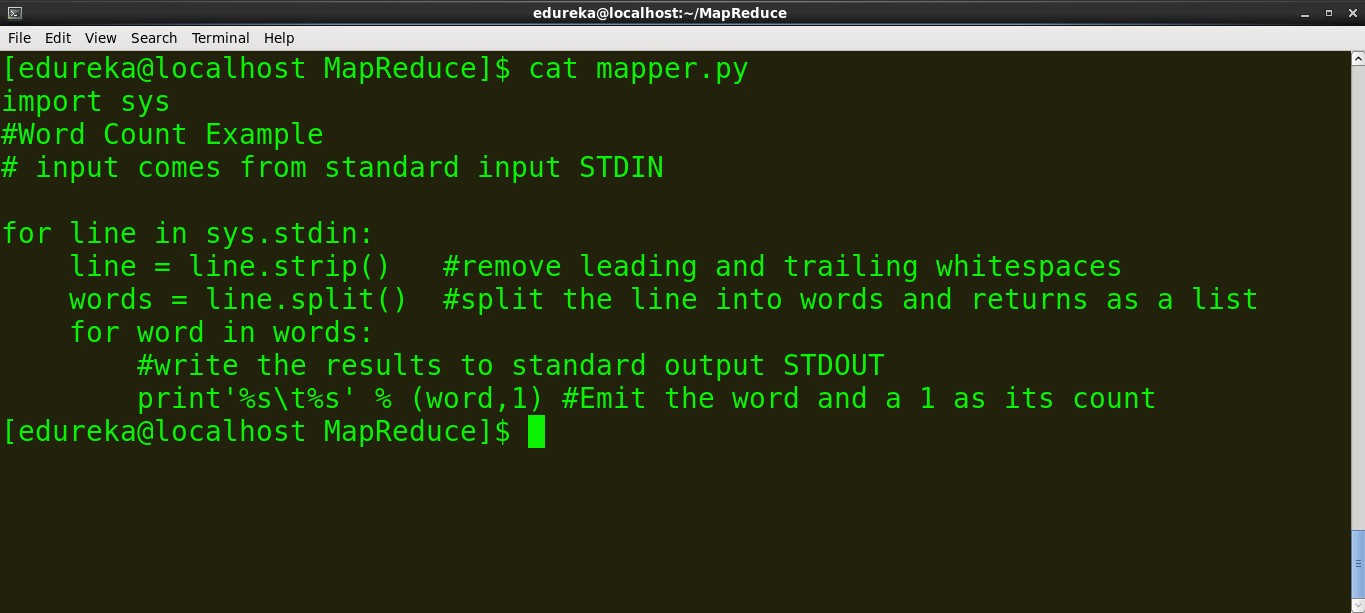


1. See the content of the file.  
   Command:  cat *file\_name*



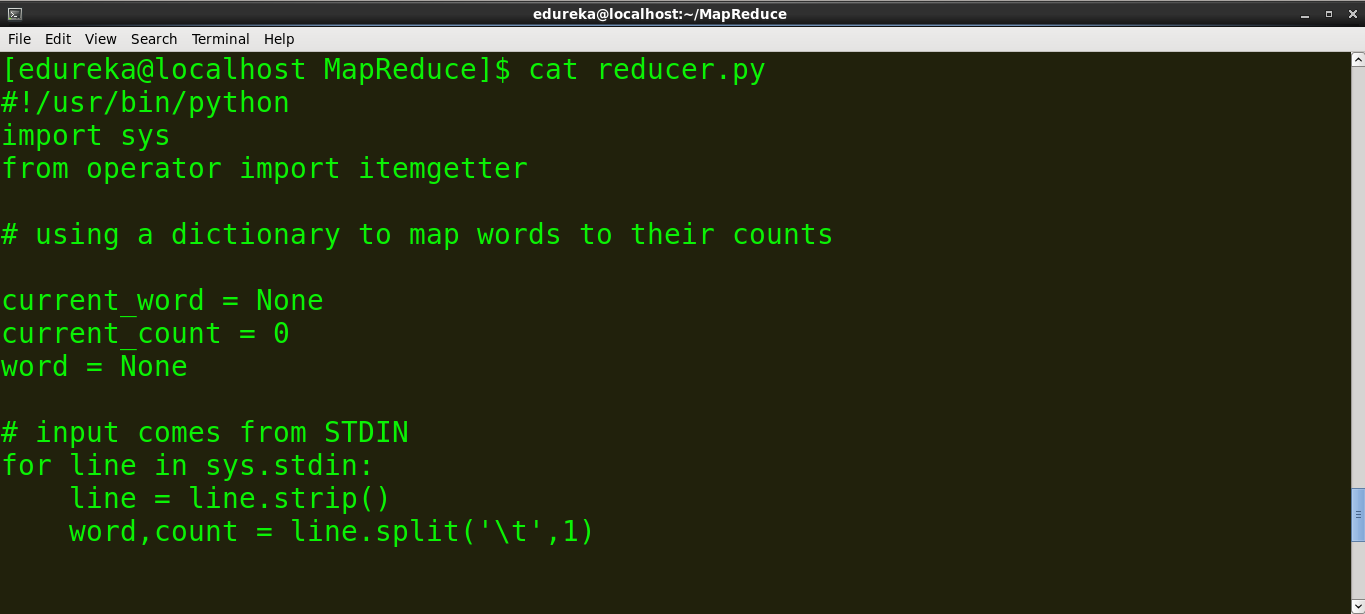
> content of mapper.py

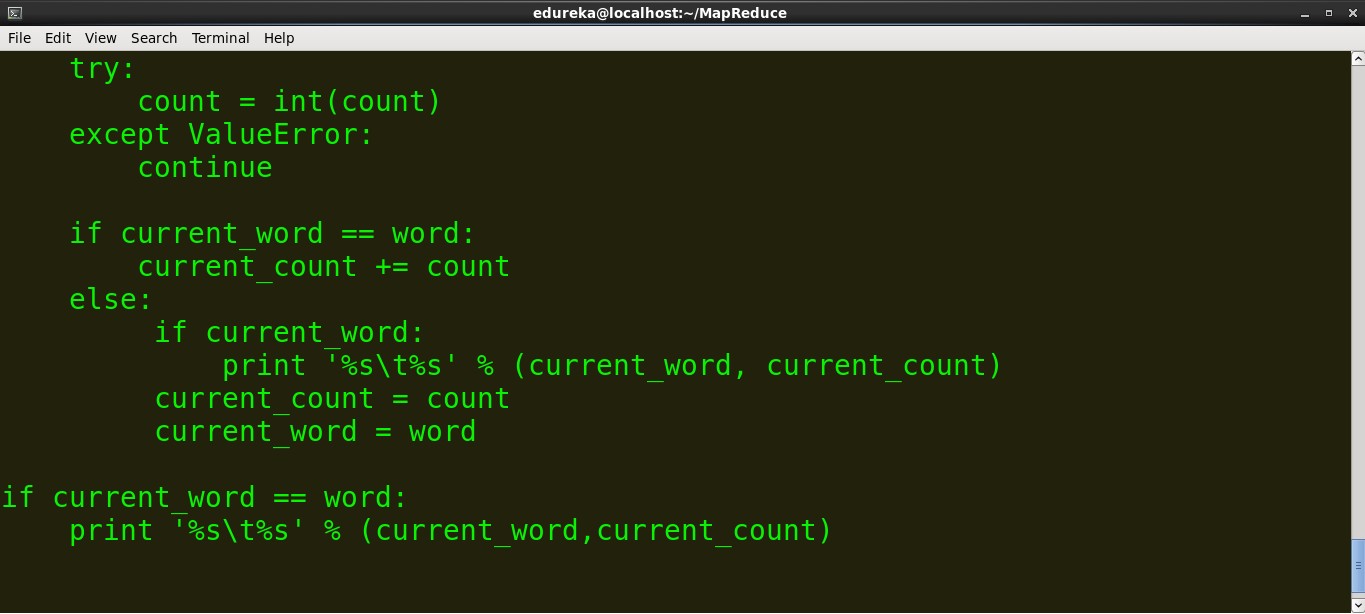
*command:  cat mapper.py*



>Content of reducer.py

command: cat *reducer.py*

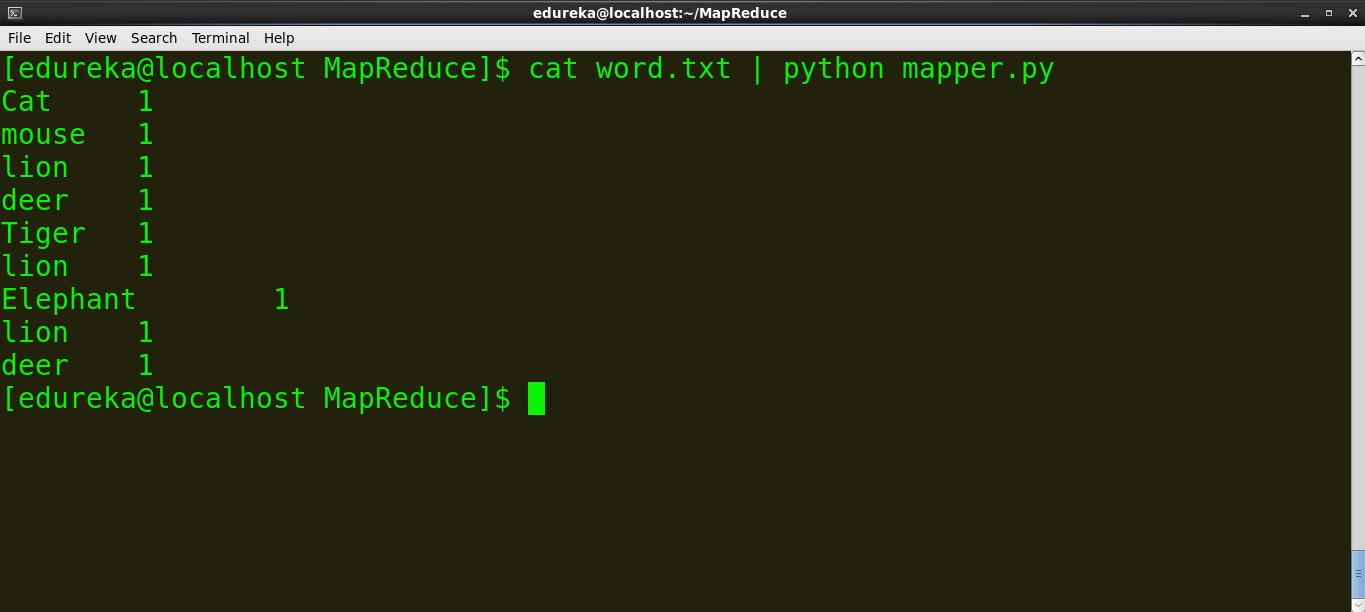




We can run mapper and reducer on local files (ex: word.txt). In order to run the Map and reduce on the Hadoop Distributed File System (HDFS), we need the *Hadoop Streaming jar.* So before we run the scripts on HDFS, let’s run them locally to ensure that they are working fine.

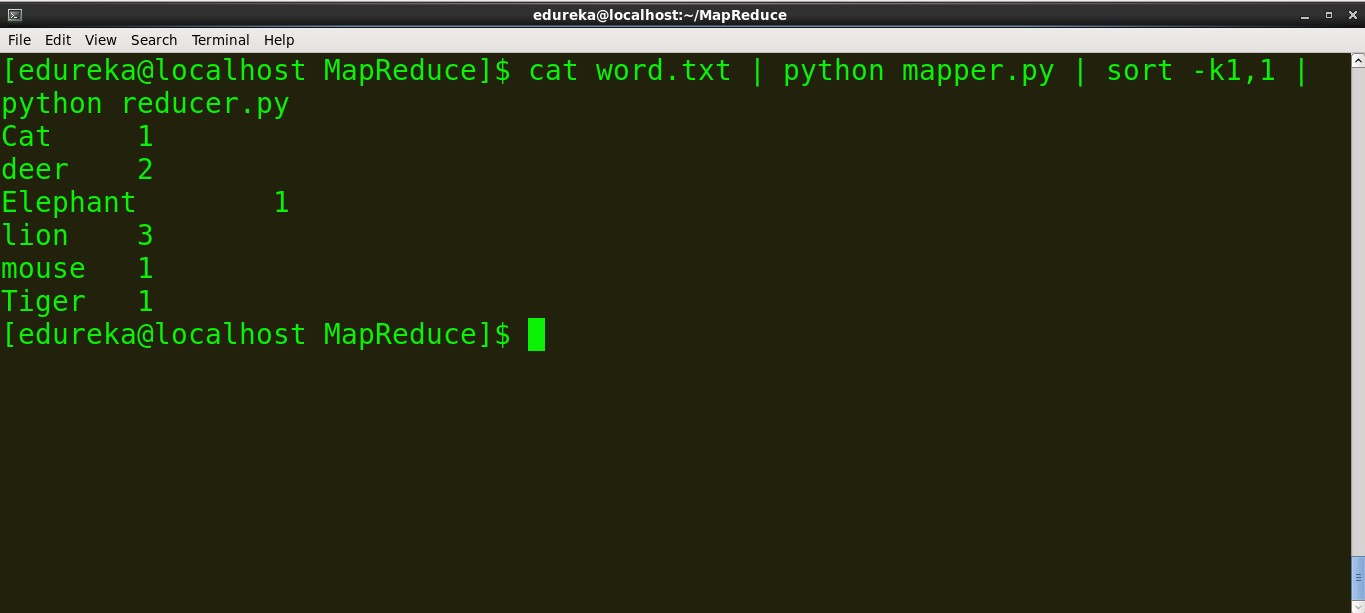
>Run the mapper

command: *cat word.txt | python mapper.py*



>Run reducer.py

command: *cat word.txt | python mapper.py | sort -k1,1 | python reducer.py*



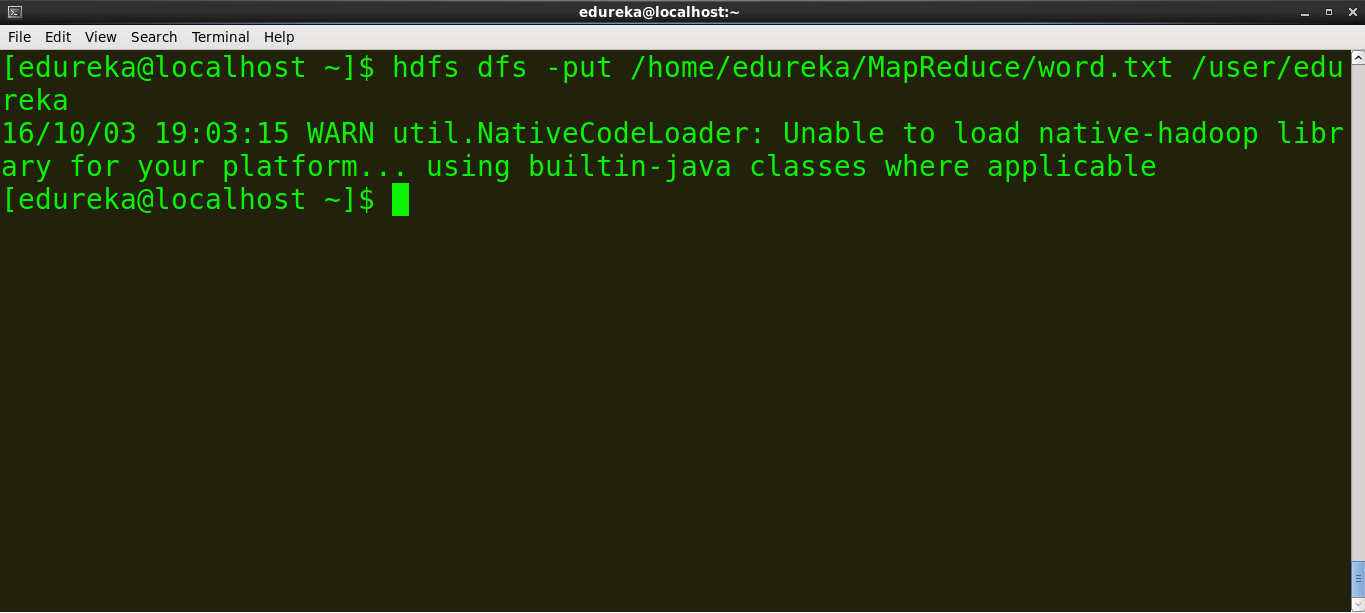
We can see that the mapper and reducer are working as expected so we won’t face any further issues.

**Running the Python Code on Hadoop**

Before we run the MapReduce task on Hadoop, copy local data (word.txt) to HDFS

>example: *hdfs dfs -put source\_directory hadoop\_destination\_directory*

command: *hdfs dfs -put /home/edureka/MapReduce/word.txt   /user/edureka*



**Copy the path of the jar file**

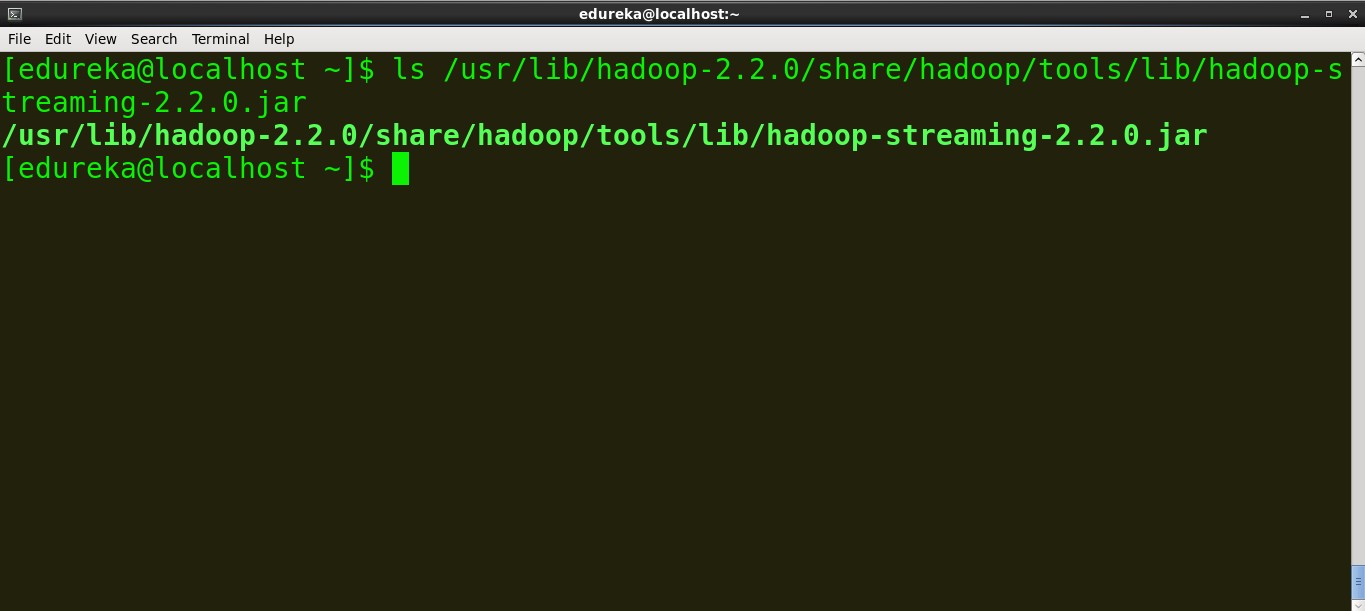
The path of Hadoop Streaming jar based on the version of the jar is:

*/usr/lib/hadoop-2.2.X/share/hadoop/tools/lib/hadoop-streaming-2.2.X.jar*

So locate the Hadoop Streaming jar on your terminal and copy the path.

command:

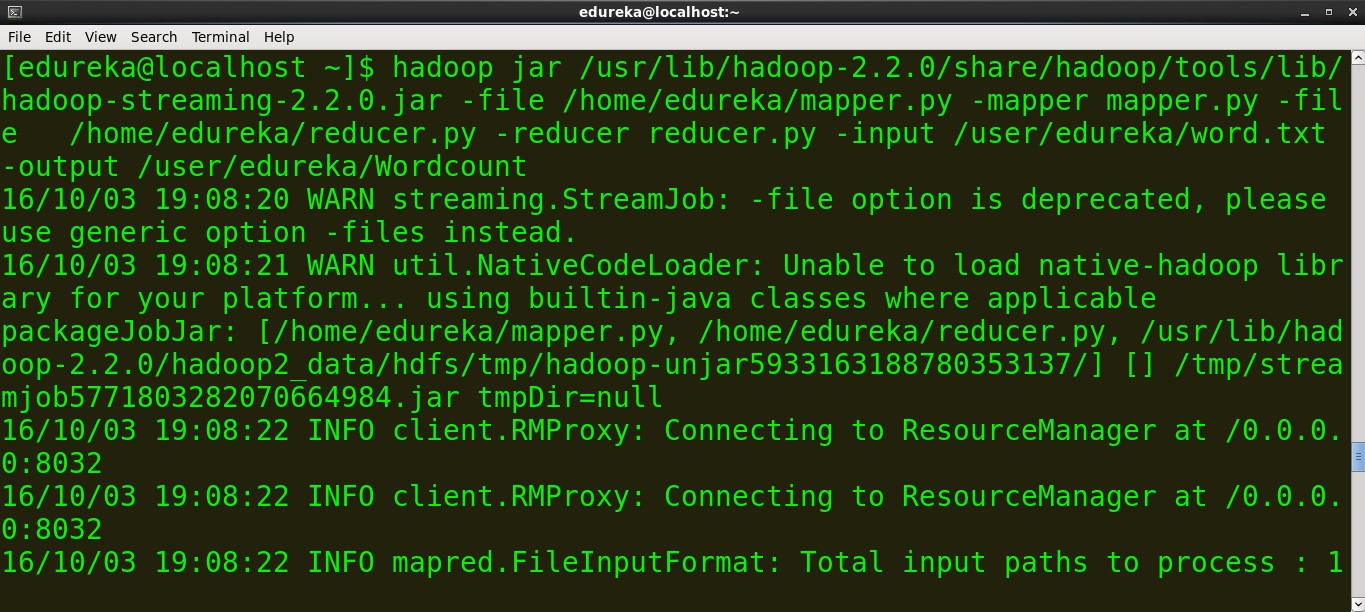
ls /usr/lib/hadoop-2.2.0/share/hadoop/tools/lib/hadoop-streaming-2.2.0.jar

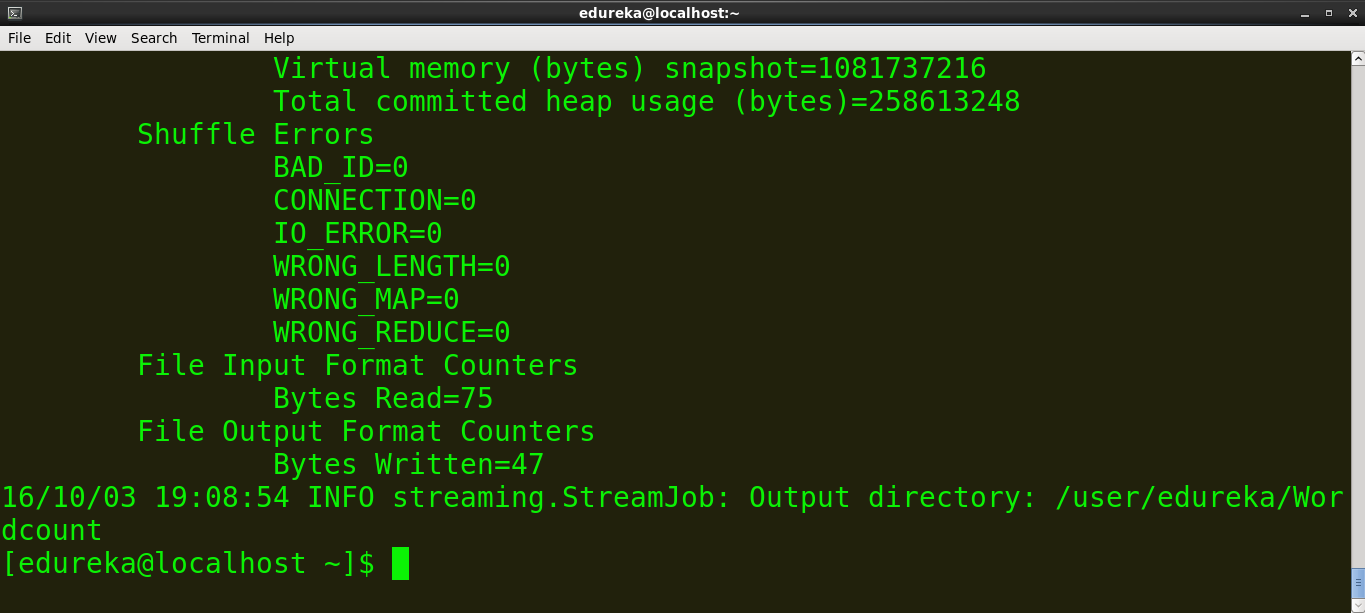


**Run the MapReduce job**

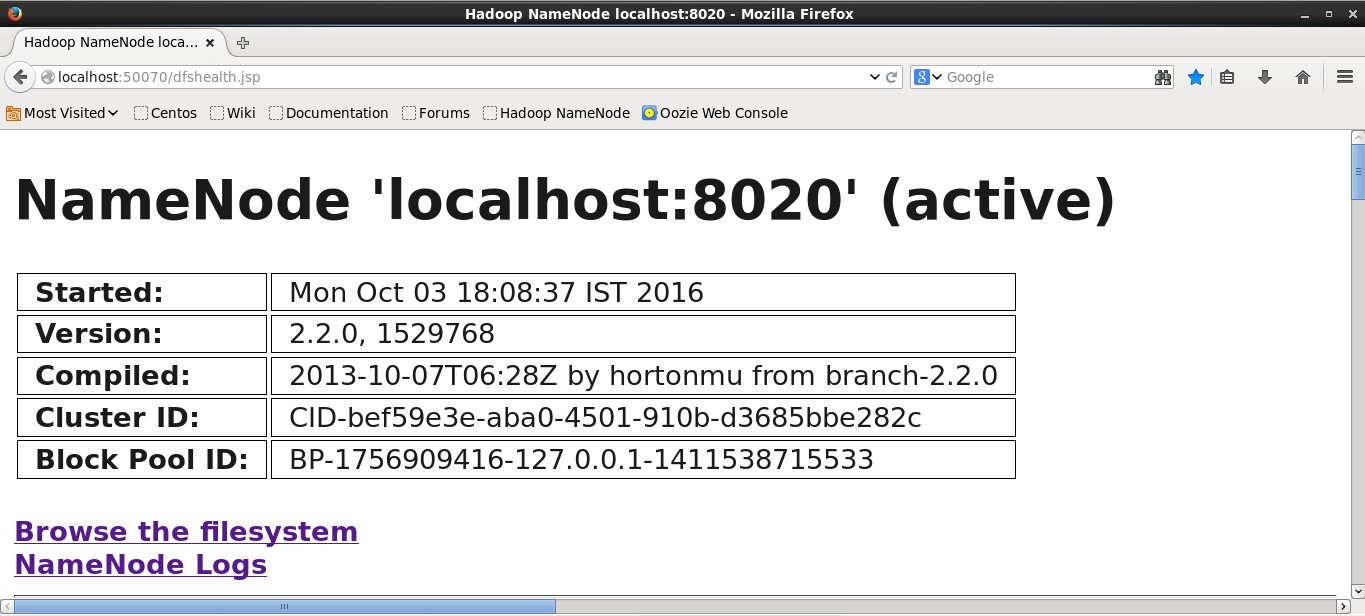
command:

*hadoop jar /usr/lib/hadoop-2.2.0/share/hadoop/tools/lib/hadoop-streaming-2.2.0.jar -file /home/edureka/mapper.py -mapper mapper.py -file   /home/edureka/reducer.py -reducer reducer.py -input /user/edureka/word -output /user/edureka/Wordcount*

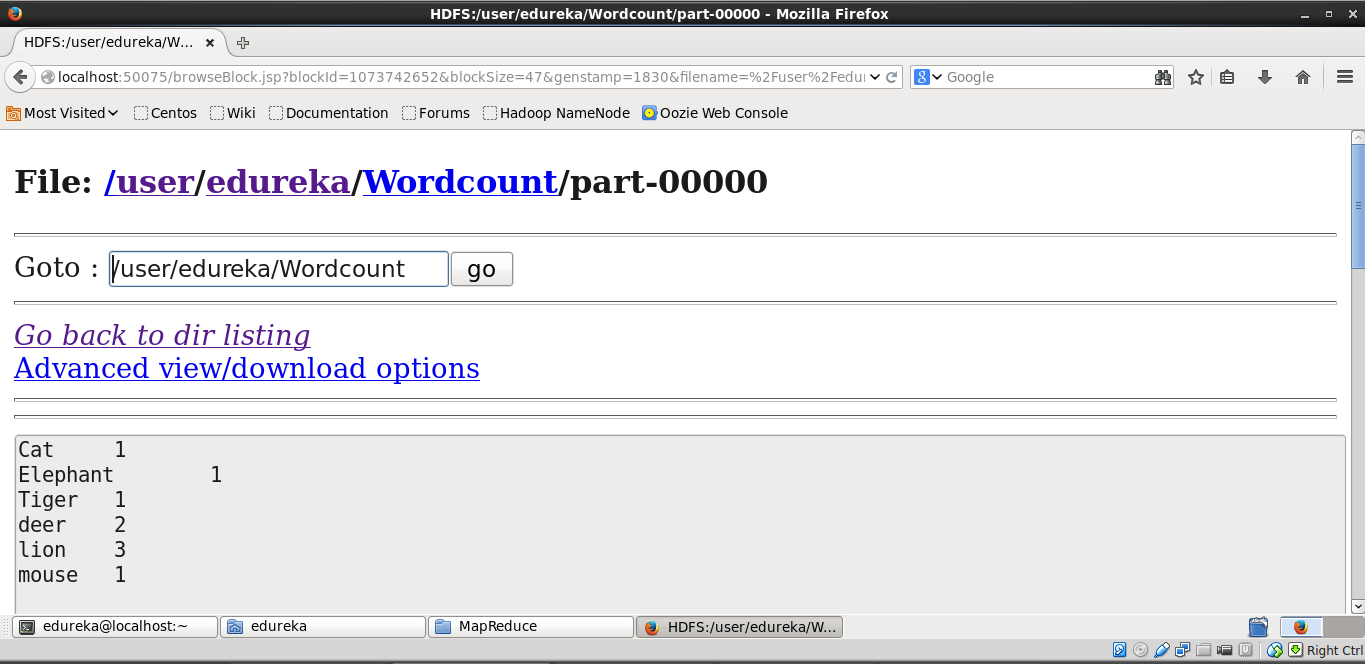




Hadoop provides a basic web interface for statistics and information. When Hadoop cluster is running open http://localhost:50070 in browser. Here is the screenshot of the Hadoop web interface.

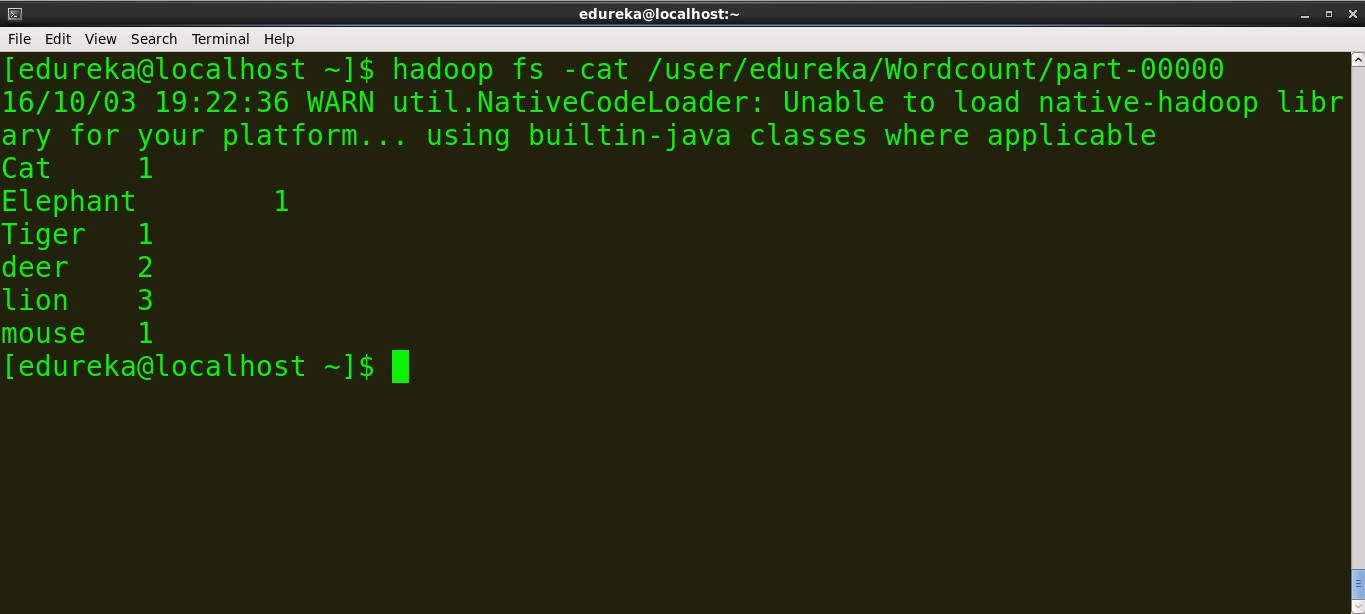


Now browse the filesystem and locate the wordcount file generated to see the output. Below is the screenshot.



We can see the output on the terminal using this command

command: *hadoop fs -cat /user/edureka/Wordcount/part-00000*



You have now learnt how to execute a MapReduce program written in Python using Hadoop Streaming!

*Edureka has a live and instructor-led course on Big Data & Hadoop, co-created by industry practitioners.* [*Click to know more*](http://www.edureka.co/big-data-and-hadoop?utm_source=blog&utm_medium=related-posts&utm_campaign=hadoop-streaming-26sept16)