Name: Tania Rajabally

Branch: Comps

Batch: C

Roll no: 43

UID:2017130047

Date: 21/08/2020

**Experiment 2**

**Aim:** To solve Word Count Problem using MapReduce.

**Theory:**

A MapReduce is a data processing tool which is used to process the data parallely in a distributed form. It was developed in 2004, on the basis of paper titled as "MapReduce: Simplified Data Processing on Large Clusters," published by Google.

The MapReduce is a paradigm which has two phases, the mapper phase, and the reducer phase. In the Mapper, the input is given in the form of a key-value pair. The output of the Mapper is fed to the reducer as input. The reducer runs only after the Mapper is over. The reducer too takes input in key-value format, and the output of reducer is the final output.

* The map takes data in the form of pairs and returns a list of <key, value> pairs. The keys will not be unique in this case.
* Using the output of Map, sort and shuffle are applied by the Hadoop architecture. This sort and shuffle acts on these list of <key, value> pairs and sends out unique keys and a list of values associated with this unique key <key, list(values)>.
* An output of sort and shuffle sent to the reducer phase. The reducer performs a defined function on a list of values for unique keys, and Final output <key, value> will be stored/displayed.

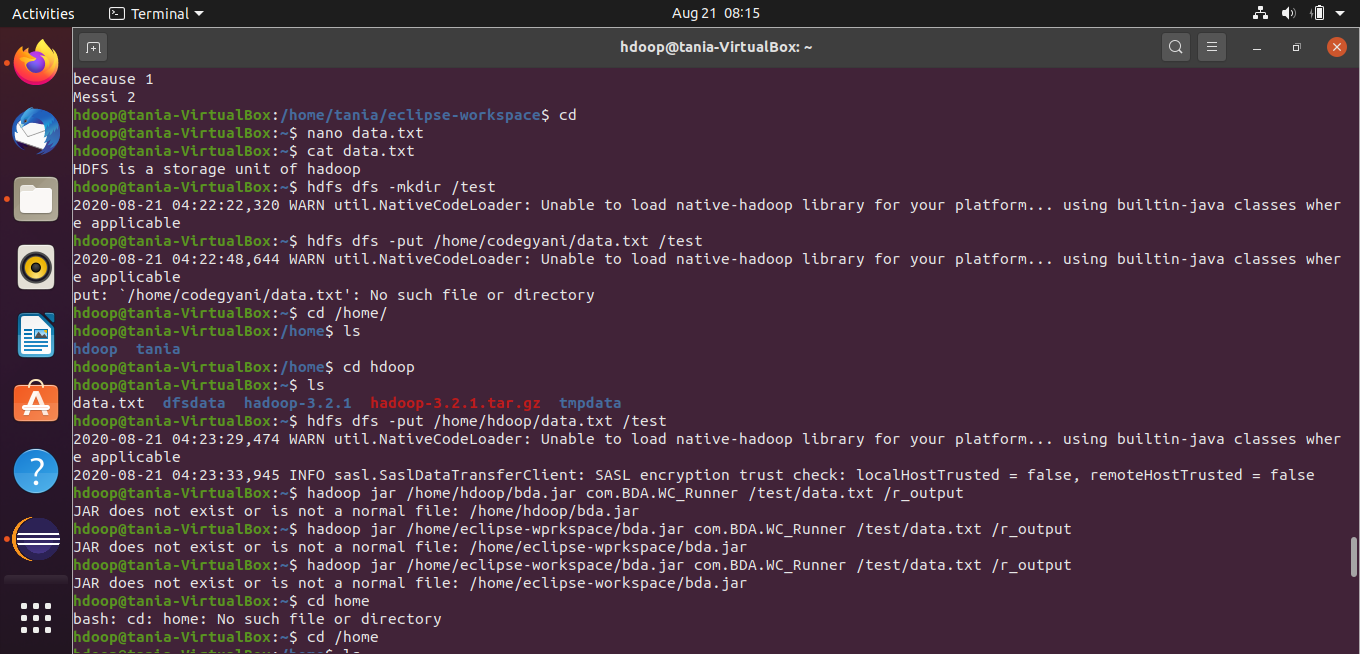
Usage of Map Reduce:

* It can be used in various applications like document clustering, distributed sorting, and web link-graph reversal.
* It can be used for distributed pattern-based searching.
* We can also use MapReduce in machine learning.
* It was used by Google to regenerate Google's index of the World Wide Web.
* It can be used in multiple computing environments such as multi-cluster, multi-core, and mobile environment.

**Steps:**

**For a single file:**

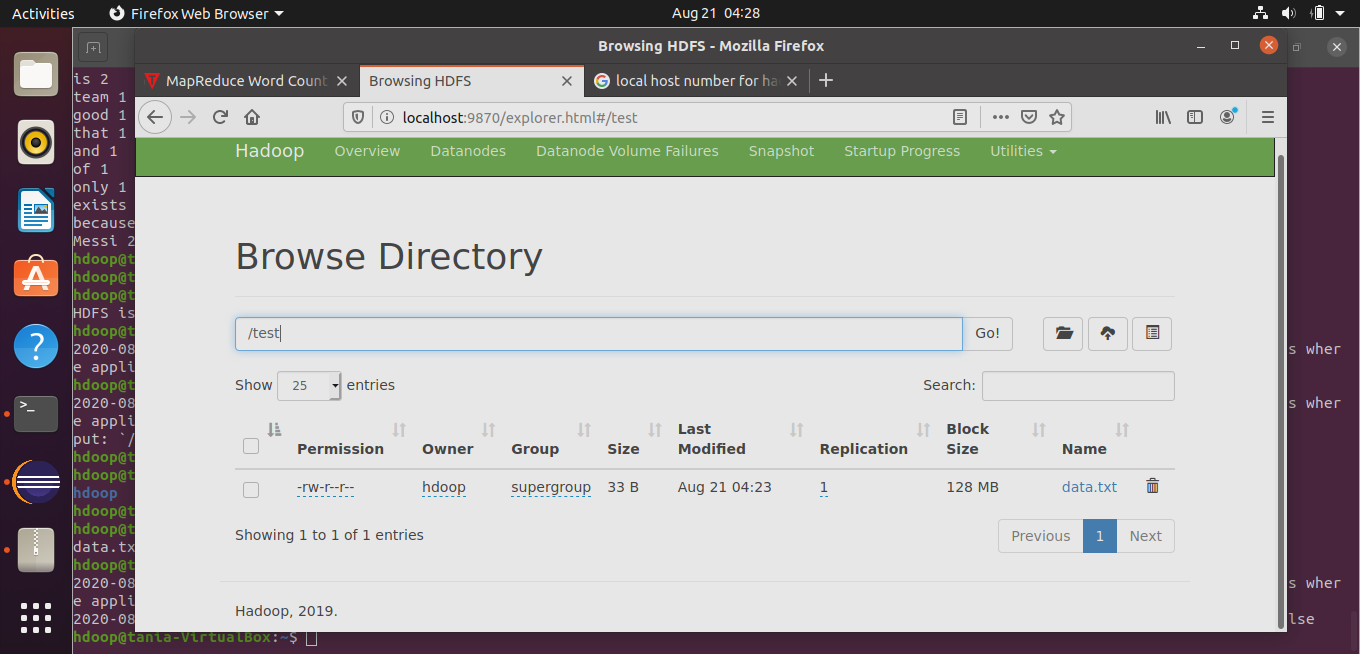
Step 1: Create a text file in your local machine and write some text into it.

****

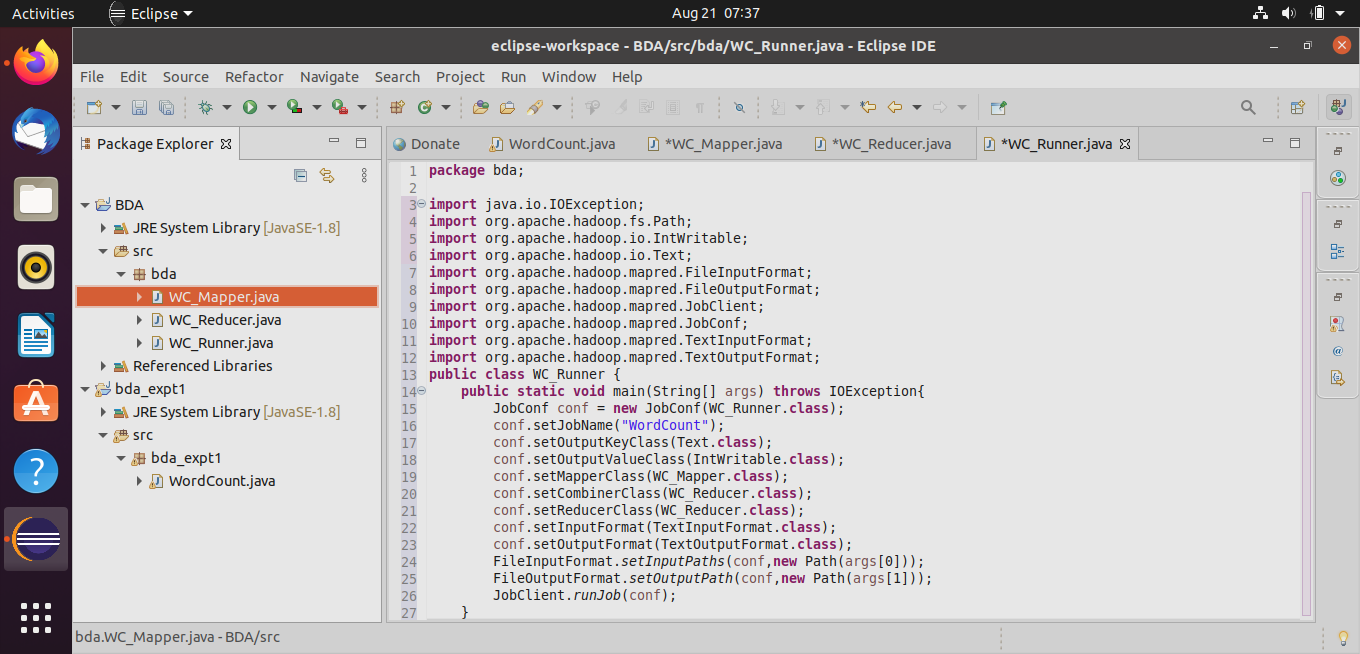
We find out the frequency of each word exists in this text file.

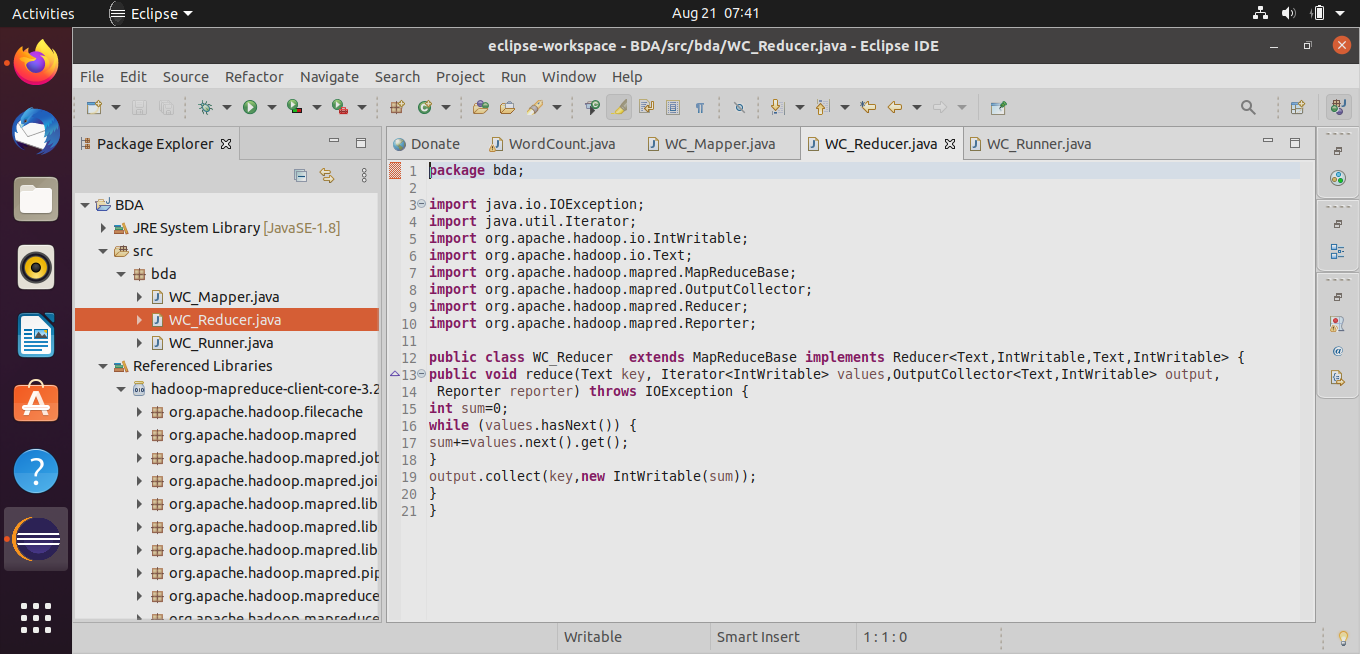
Step 2: Create a directory in HDFS, where to keep text file.

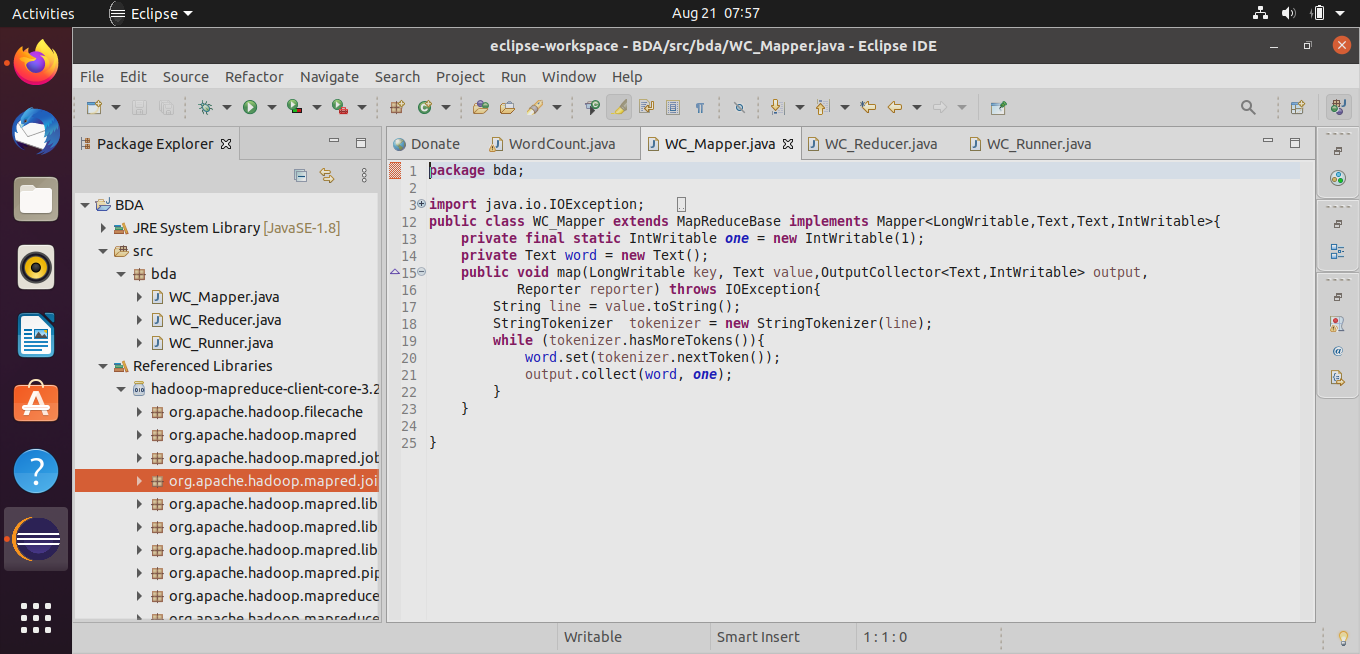
Upload the data.txt file on HDFS in the specific directory.

****

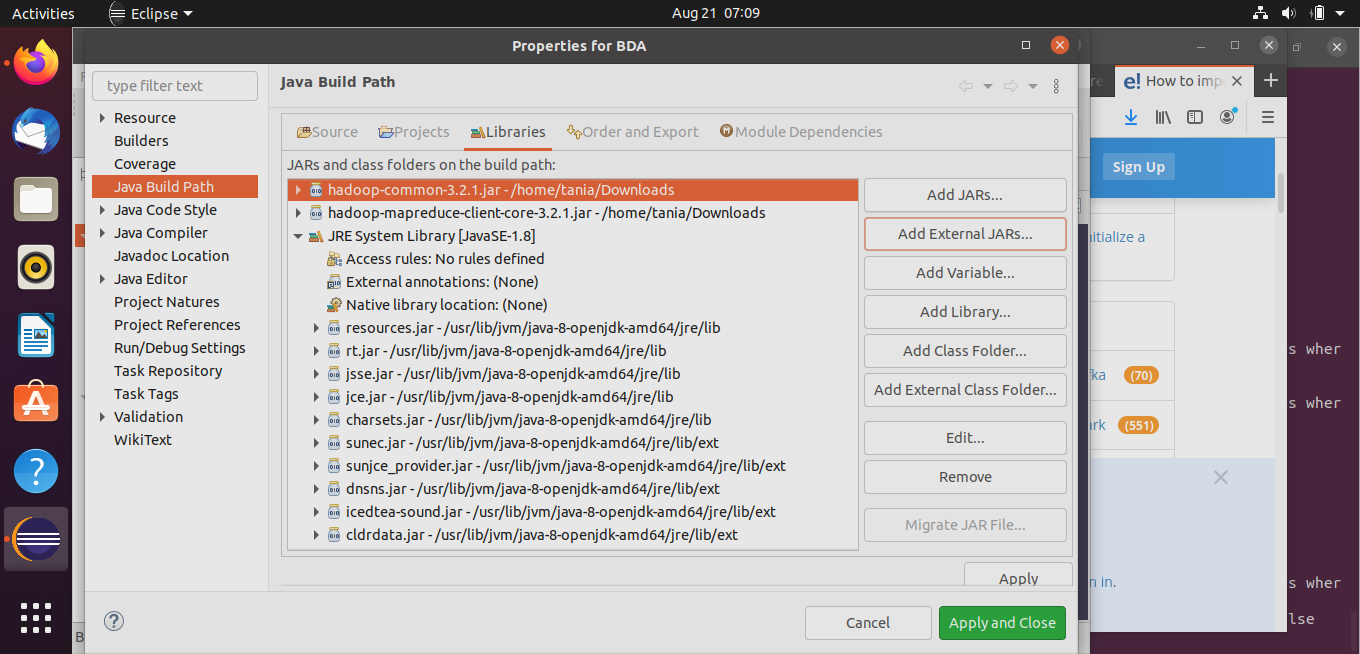
Step 3: Write the MapReduce program using eclipse.

****

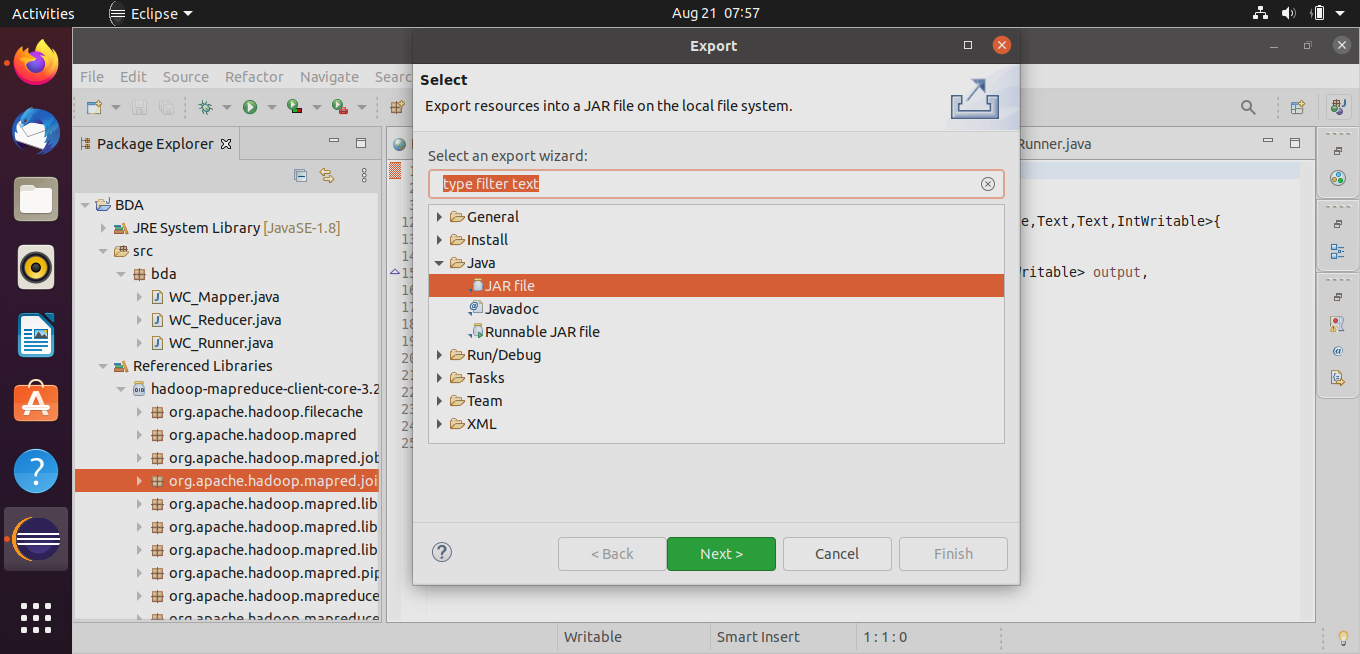
****

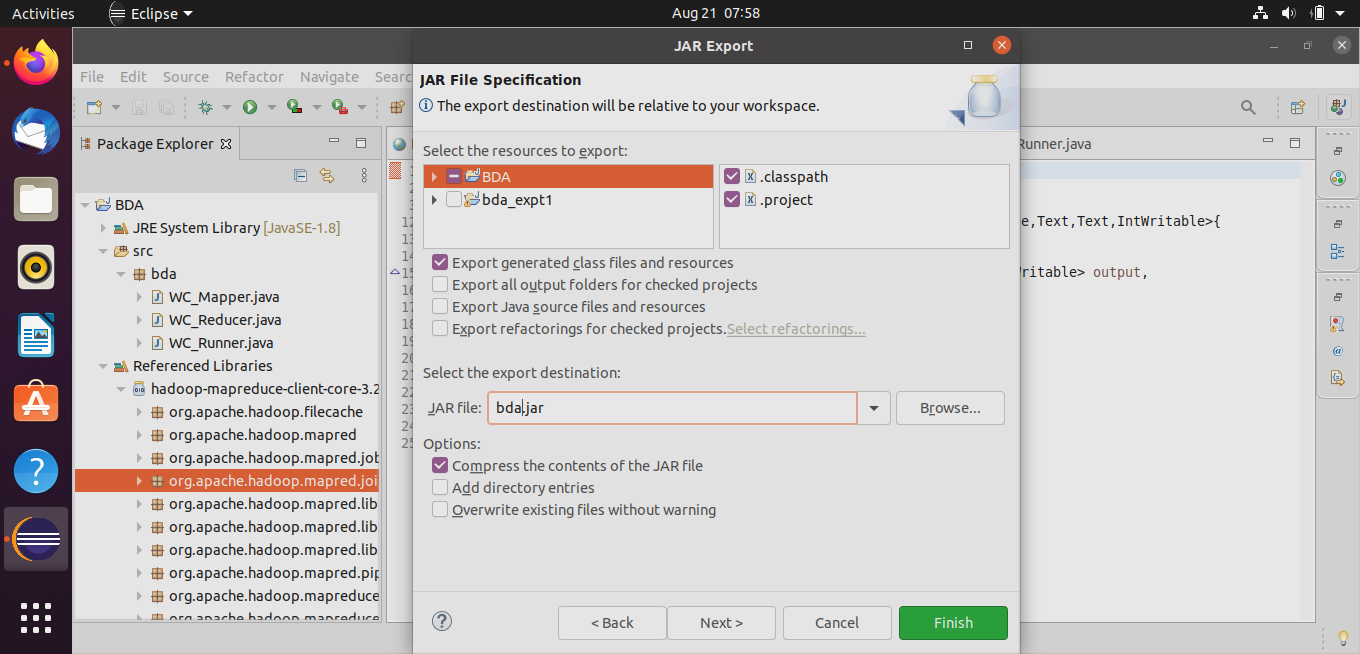
****

Step 4: Import the external JAR Files

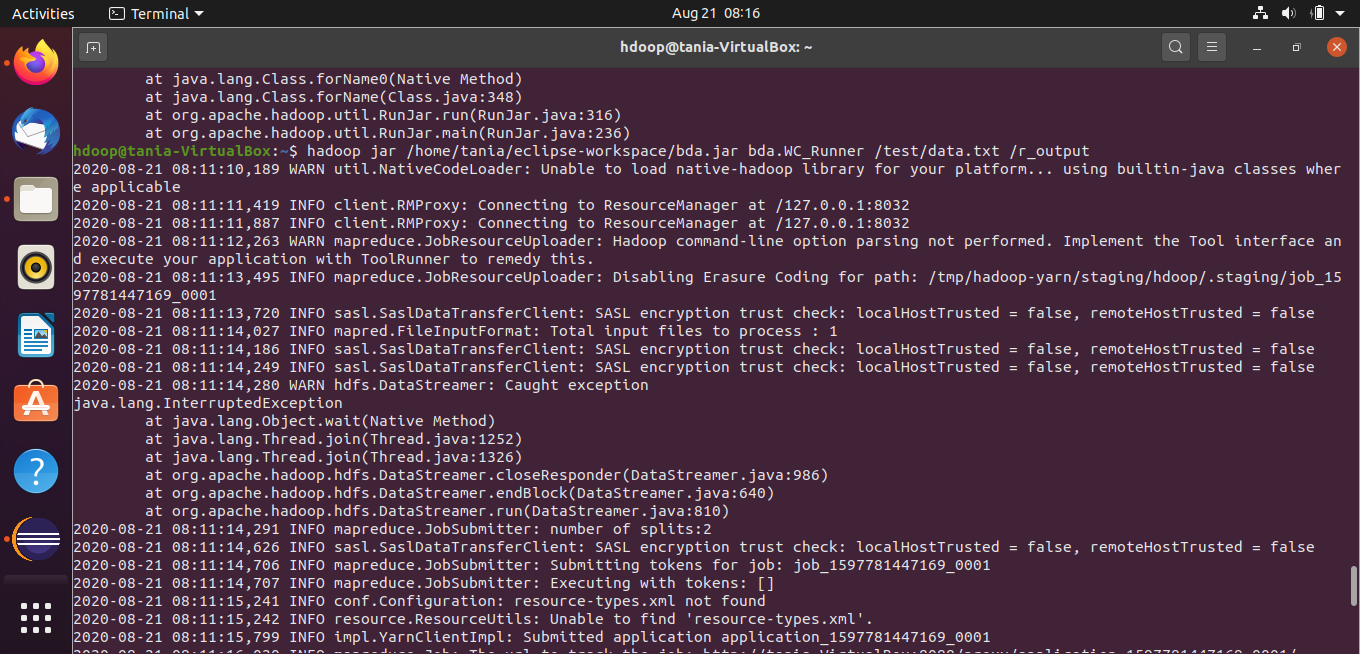
****

Step 5: Create the jar file of this program

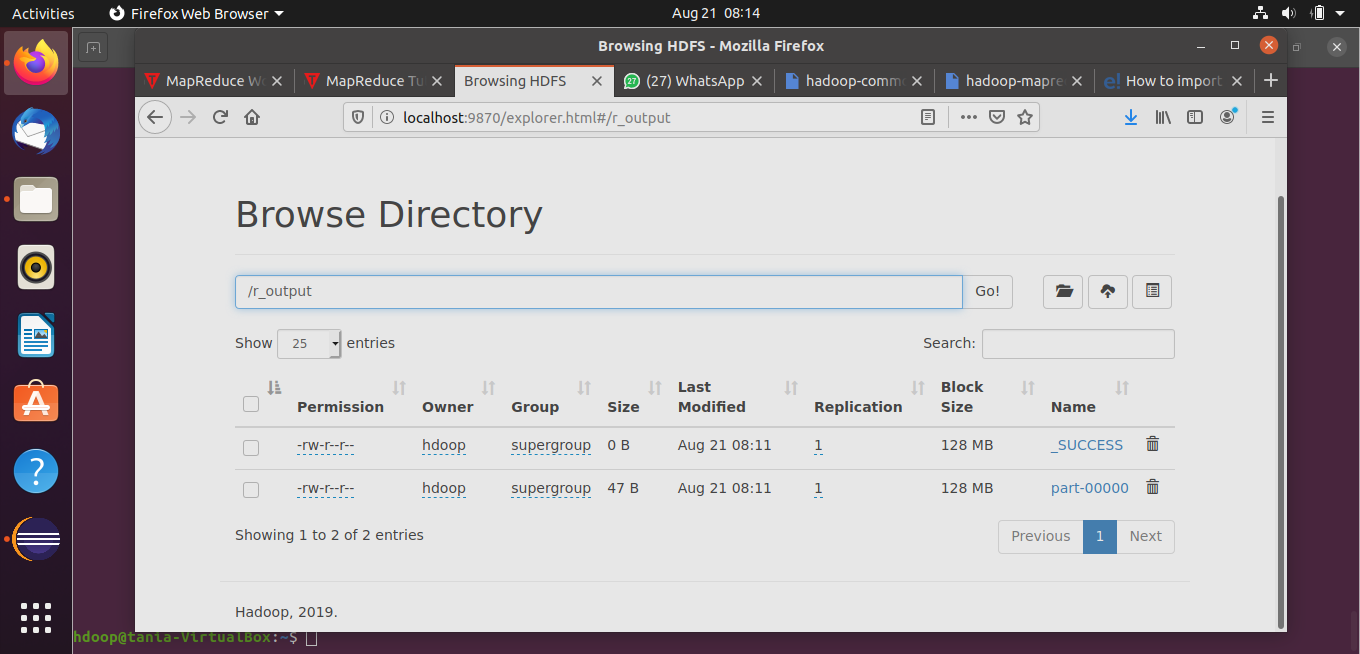
****

****

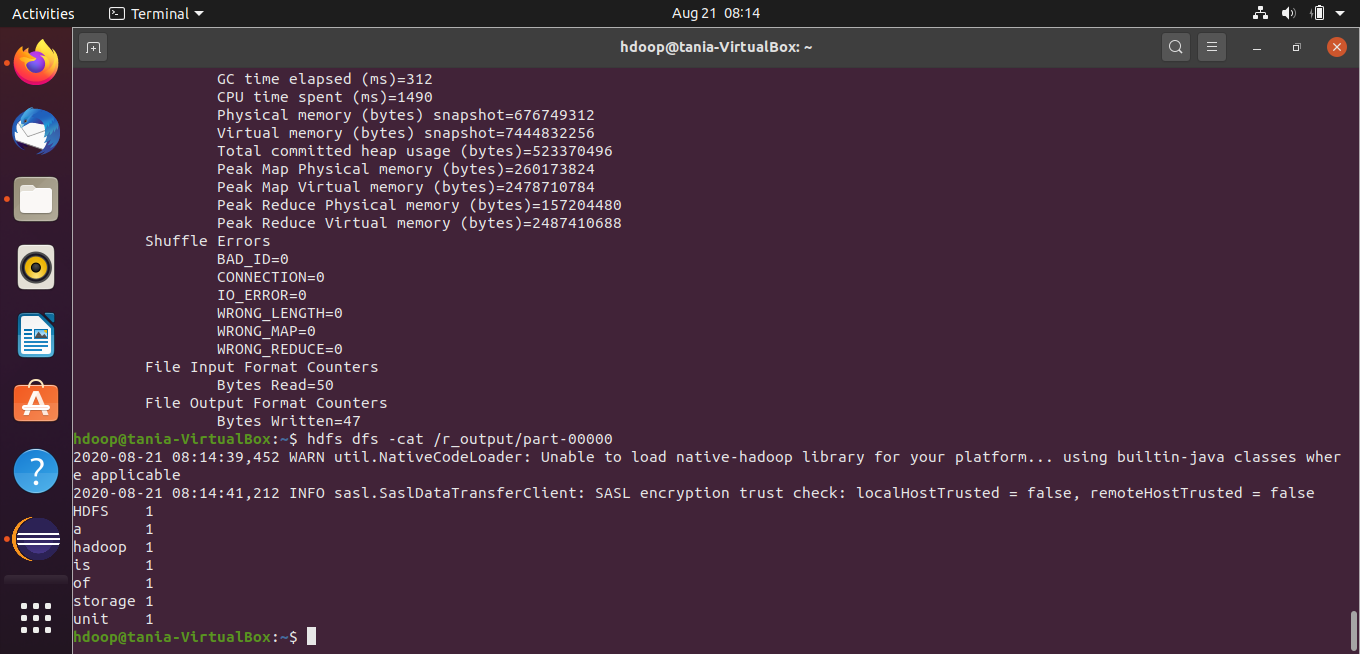
Step 6: Run the jar file

****

Step 7: The output is stored in /r\_output/part-00000

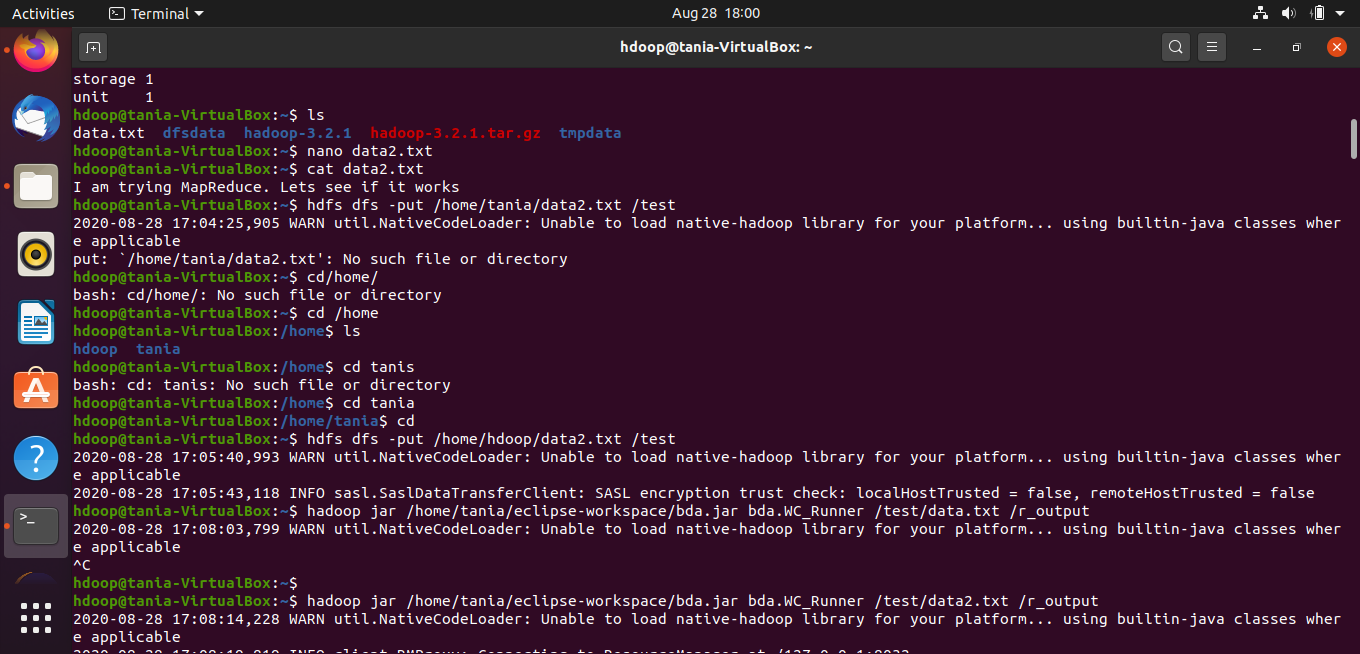
****

Step 8: Now execute the command to see the output.

****

**For 2 files:**

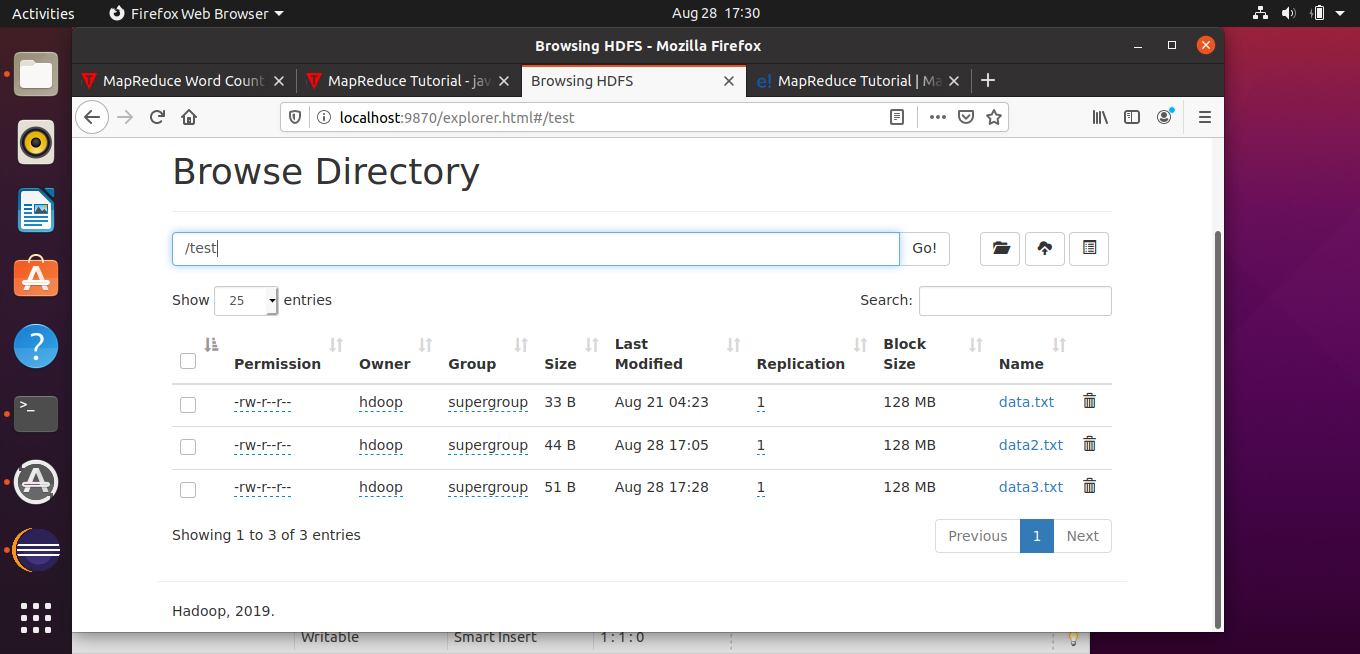
Step 1: Create 2 text files in your local machine and write some text into it.



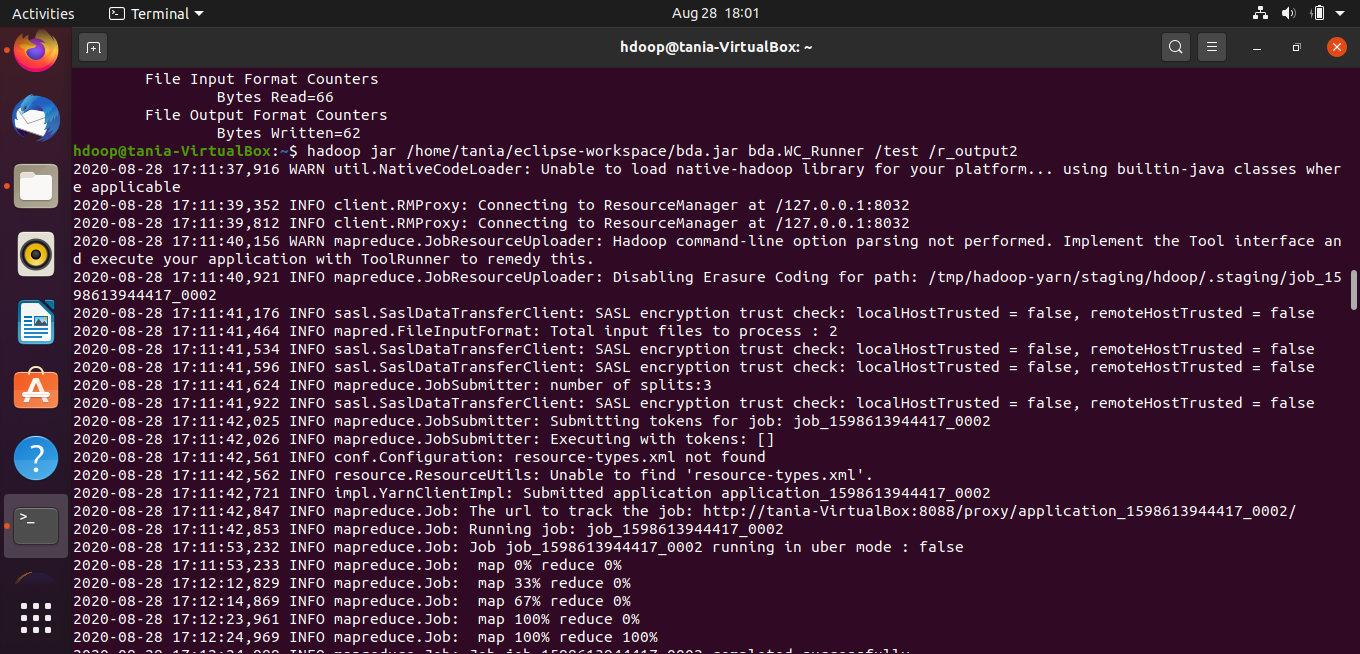
We find out the frequency of each word exists in this text file.

Step 2: Create a directory in HDFS, where to keep text file.

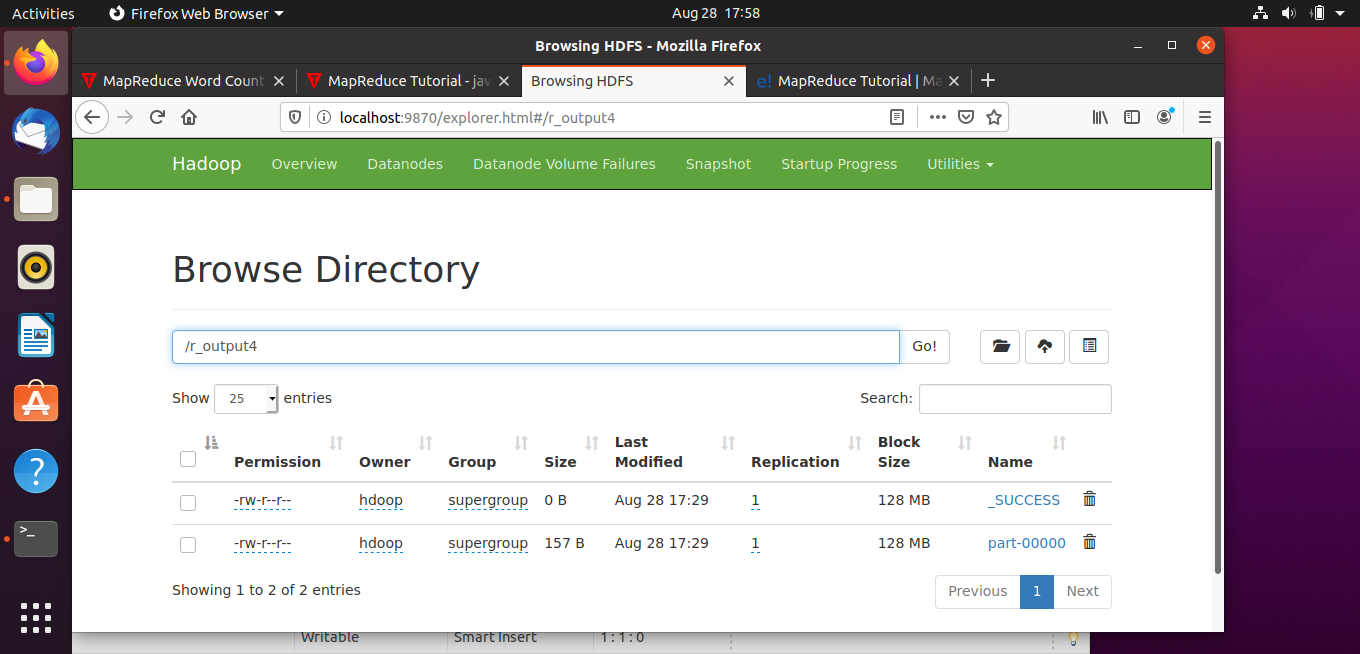
Upload the data.txt file on HDFS in the specific directory.



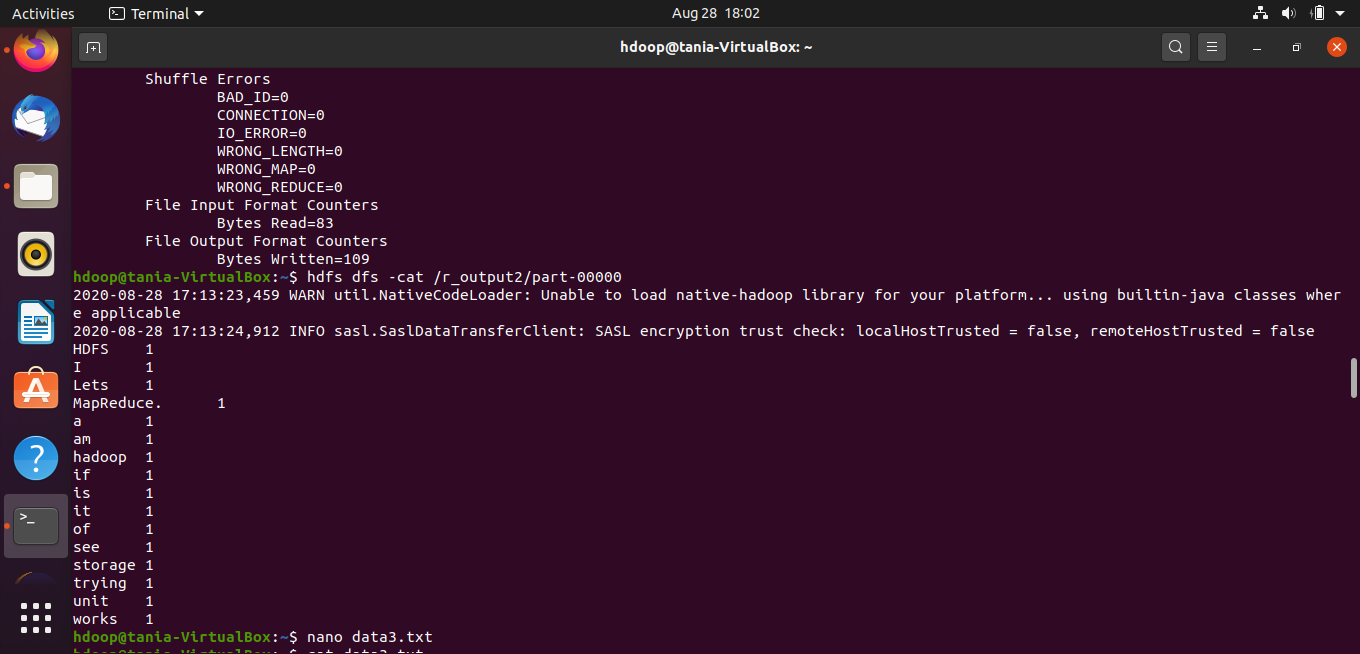
Step 3: Use the above created jar file and run it. Just mention the directory and not the txt file. It creates an output file considering both the text files.



Step 4: The output is stored in /r\_output/part-00000

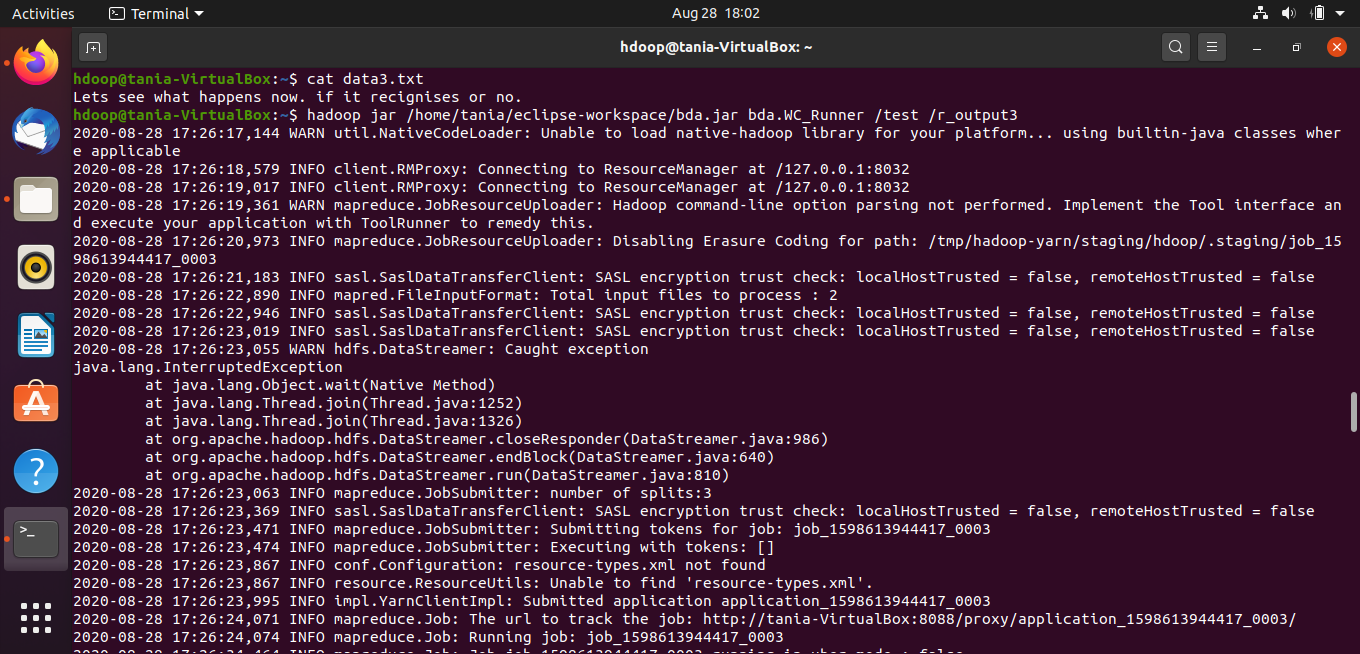


Step 5: Now execute the command to see the output.



**For 3 files:**

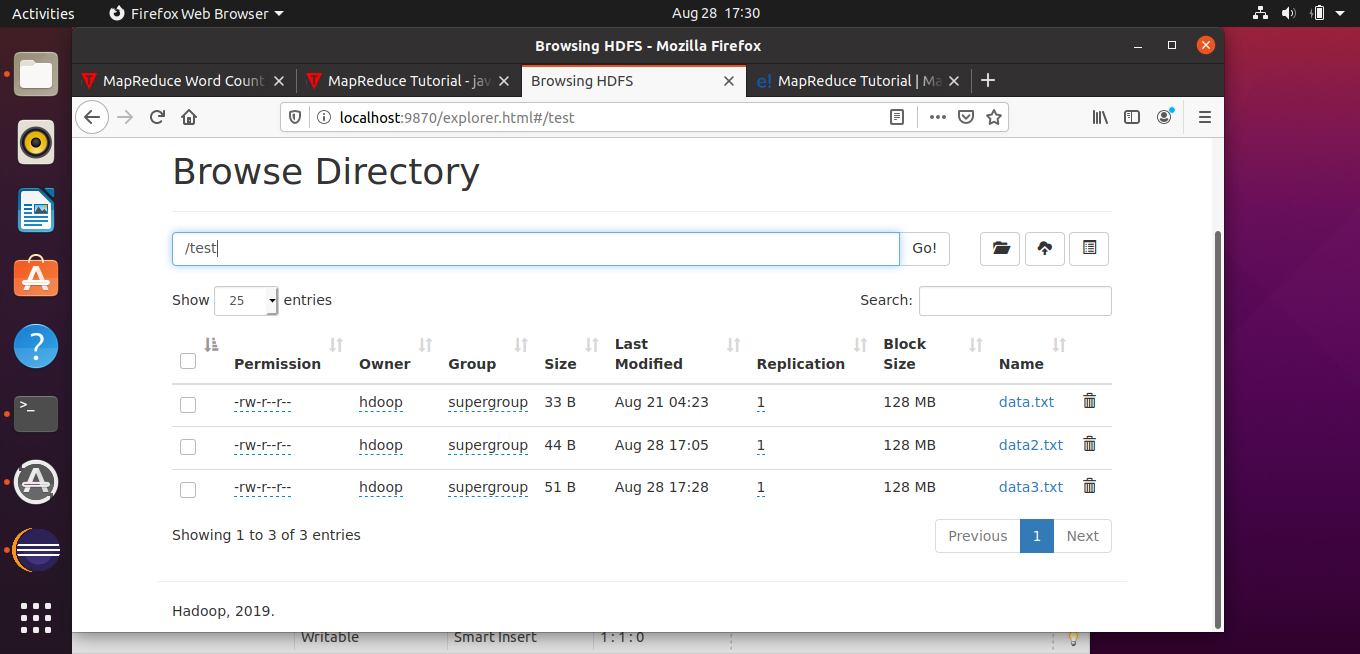
Step 1: Create 3 text files in your local machine and write some text into it.



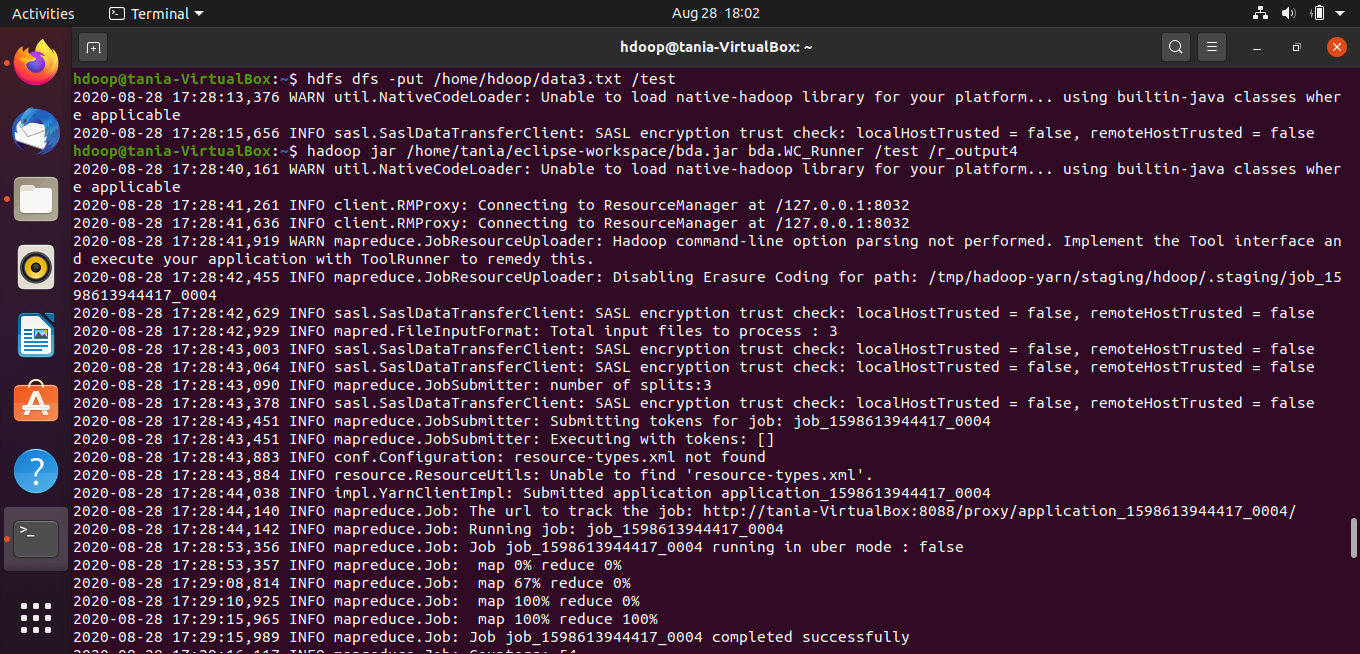
We find out the frequency of each word exists in this text file.

Step 2: Create a directory in HDFS, where to keep text file.

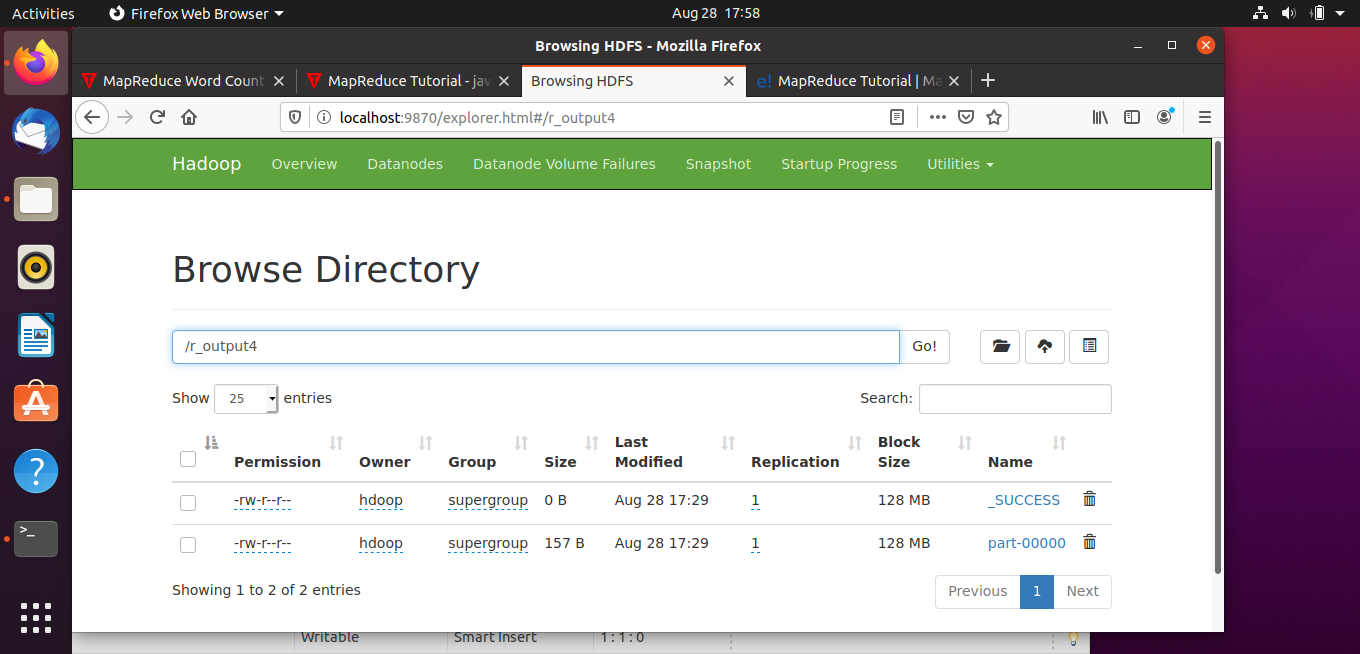
Upload the data.txt file on HDFS in the specific directory.



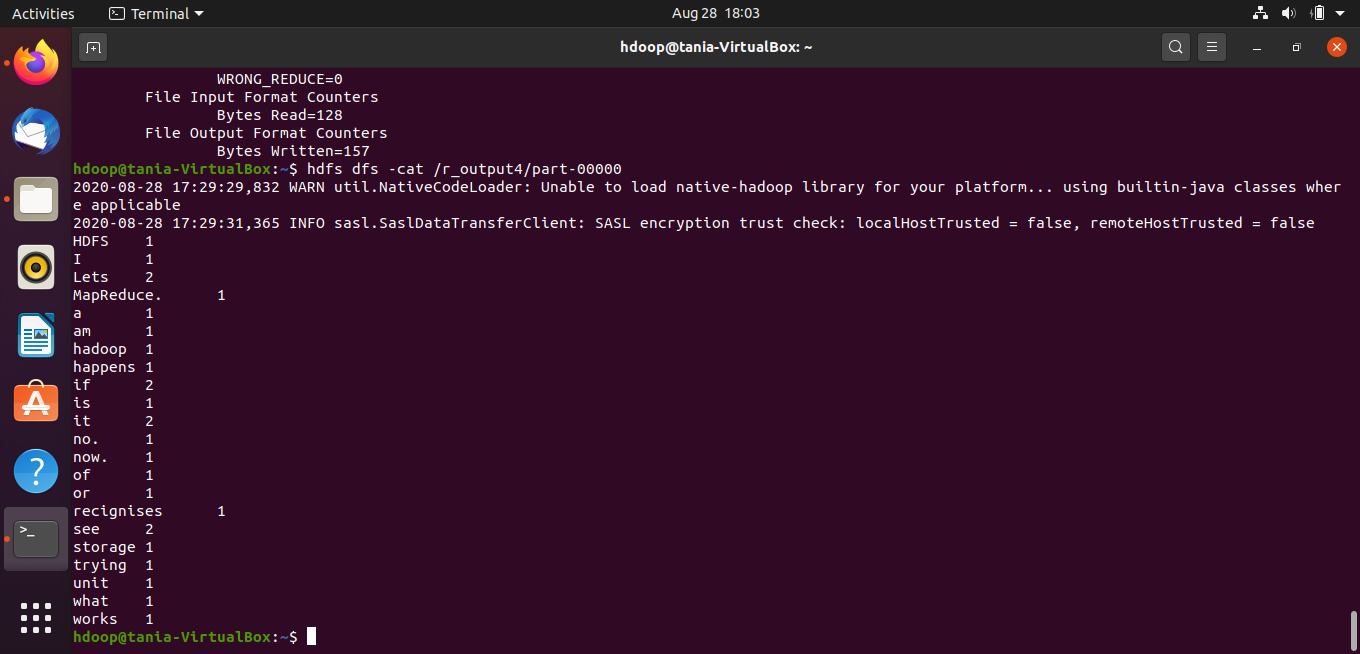
Step 3: Use the above created jar file and run it. Just mention the directory and not the txt file. It creates an output file considering all 3 text files.



Step 4: The output is stored in /r\_output4/part-00000



Step 5: Now execute the command to see the output.



**Conclusion:**

The MapReduce word count problem was implemented on the Hadoop system using Eclipse and jar files.

We had to import external jar files for org.apache.hadoop. We then created a jar file and ran it to get the output. We also used multiple text files and solved the word count problem considering the words from all the files.