Implementation of MapReduce in the Word Count Program on the Ubuntu Distribution DigitalOcean.

A. Java and Hadoop Installation Requirements:

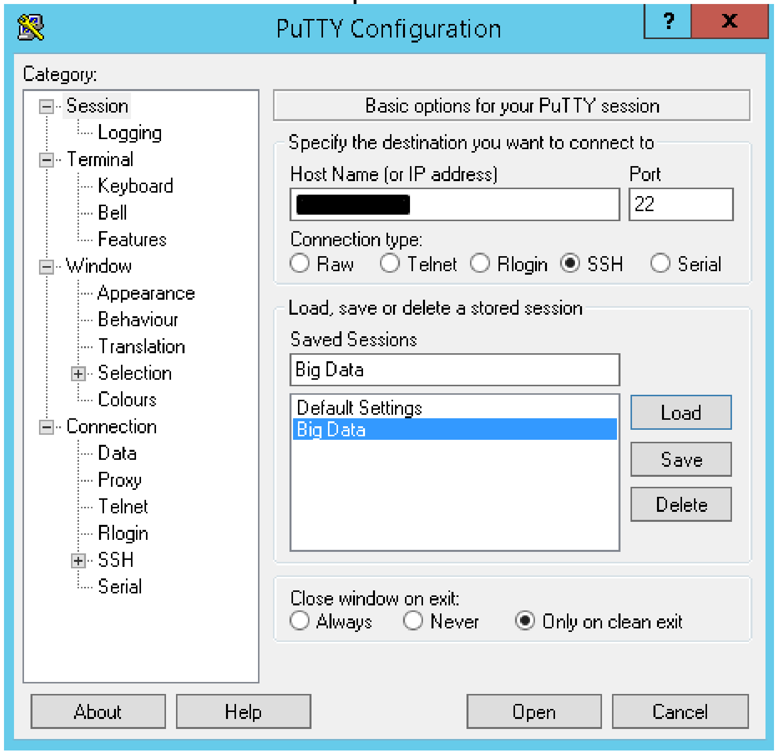
Linux Ubuntu 18.04 (DigitalOcean)

Java Development Kit (already installed and configured)

Apache Hadoop (already installed and configured)

B. MapReduce Implementation

1. First, access your Ubuntu 18.04 Linux droplet.



Specify the destination you want to connect to

Host Name (or IP address): [redacted]

Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

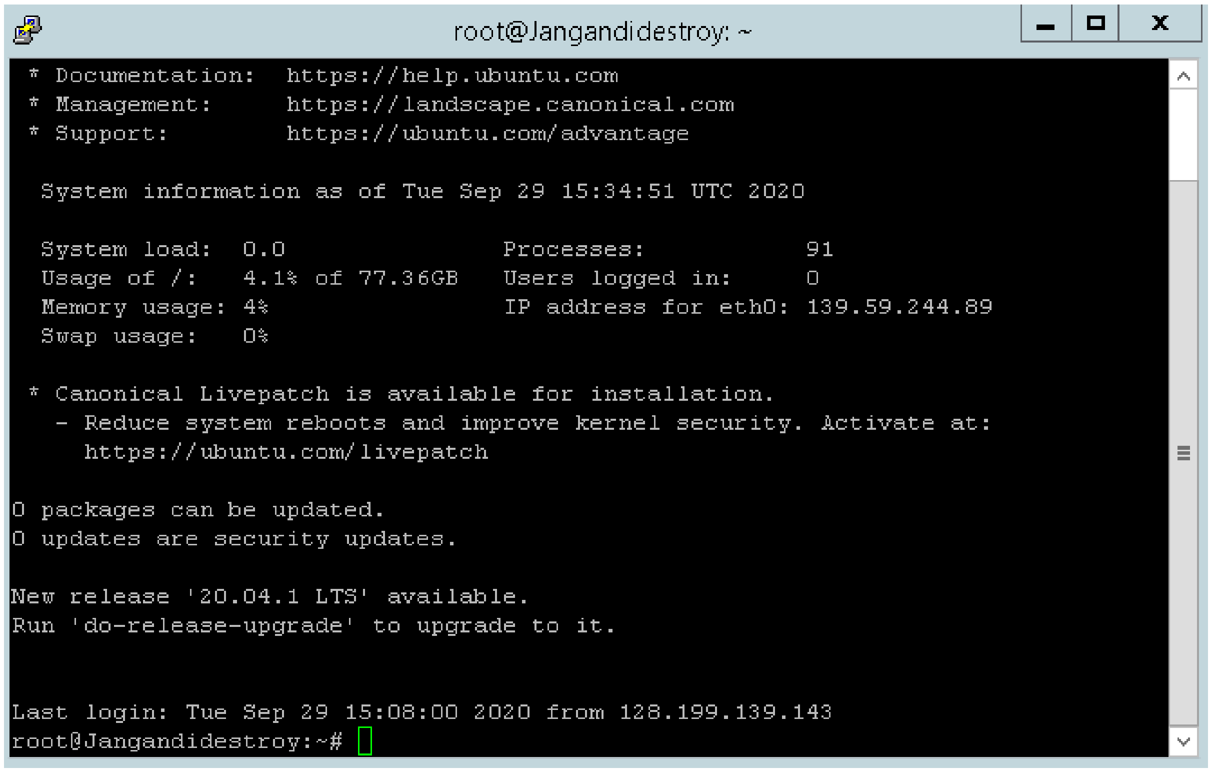
Load, save or delete a stored session

Saved Sessions: Big Data

Default Settings
Big Data

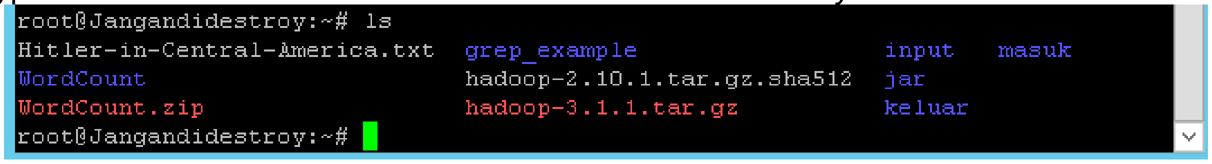
Close window on exit: Always Never [✓ Only on clean exit]

2. Then log in using the username and password you set when creating the droplet.



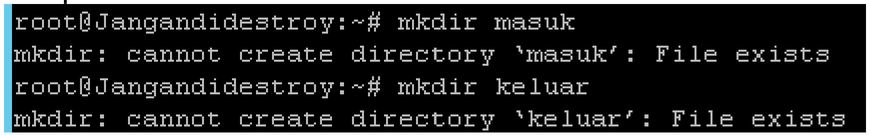
```
 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Tue Sep 29 15:34:51 UTC 2020

  System load:  0.0               Processes:           91
  Usage of /:   4.1% of 77.36GB   Users logged in:     0
  Memory usage: 4%                IP address for eth0: 139.59.244.89
  Swap usage:   0%

 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

0 packages can be updated.
0 updates are security updates.

New release '20.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Tue Sep 29 15:08:00 2020 from 128.199.139.143
root@Jangandidestroy:~#
```

3. Type the command line ls to view the contents of the directory.



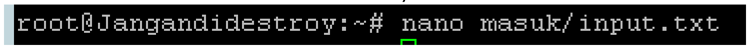
```
root@Jangandidestroy:~# ls
Hitler-in-Central-America.txt  grep_example                 input   masuk
WordCount                      hadoop-2.10.1.tar.gz.sha512  jar
WordCount.zip                  hadoop-3.1.1.tar.gz          keluar
root@Jangandidestroy:~#
```

4. Then create two folders to store the input and output files. In this case, I created folders named input and output.



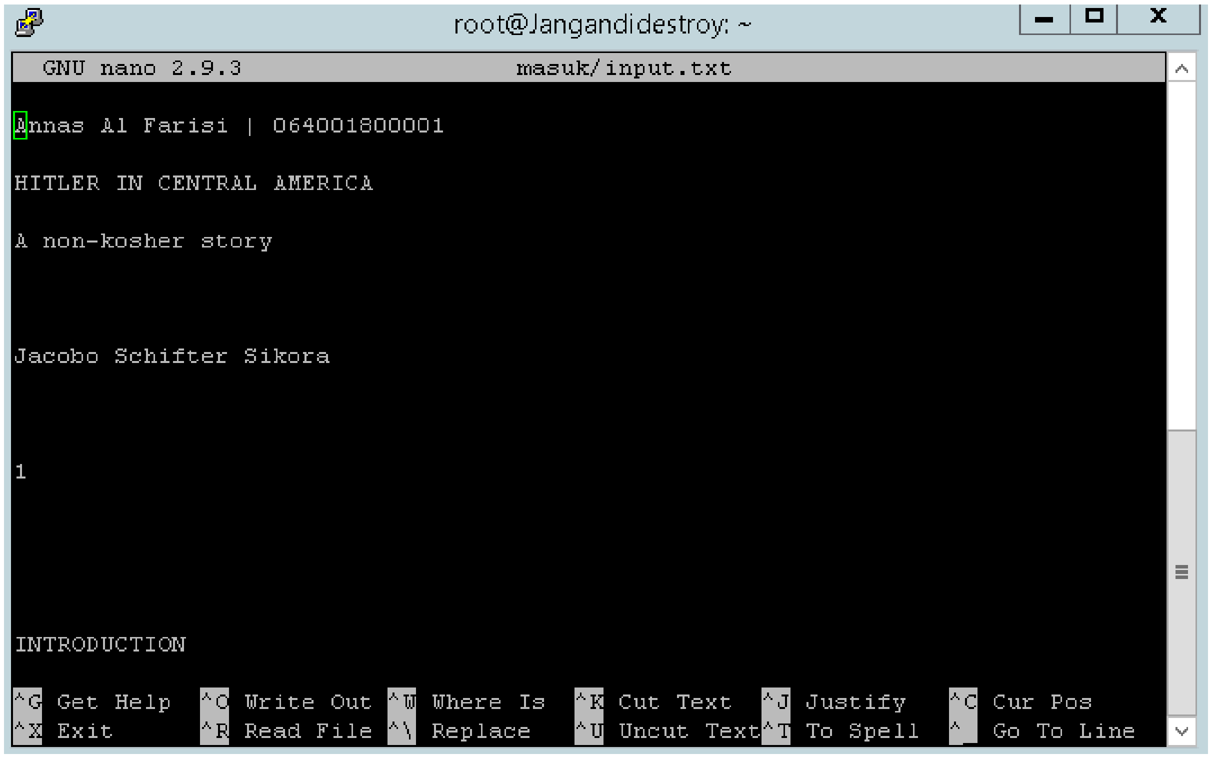
```
root@Jangandidestroy:~# mkdir masuk
mkdir: cannot create directory ‘masuk’: File exists
root@Jangandidestroy:~# mkdir keluar
mkdir: cannot create directory ‘keluar’: File exists
```

5. Create a txt file in the inbox folder. In this case, I created a file named input.txt.



```
root@Jangandidestroy:~# nano masuk/input.txt
```

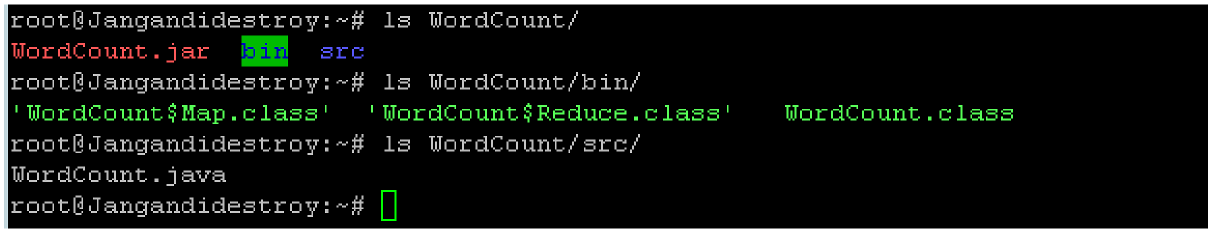
6. Once the input file is ready, you just need to fill it in with whatever you want. In this case, I filled it in with a novel.



```
root@Jangandidestroy: ~
  GNU nano 2.9.3                          masuk/input.txt

Annas Al Farisi | 064001800001

HITLER IN CENTRAL AMERICA

A non-kosher story



Jacobo Schifter Sikora



1





INTRODUCTION

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line
```

7. After editing the input.txt file, press Ctrl+X, followed by Y and then Enter to save the changes.

8. Then create several folders that will be needed to create the Word Count program. Just like in the first step, use the command line mkdir [folder name]. Here is what the directory will look like:



```
root@Jangandidestroy:~# ls WordCount/
WordCount.jar  bin  src
root@Jangandidestroy:~# ls WordCount/bin/
'WordCount$Map.class'  'WordCount$Reduce.class'   WordCount.class
root@Jangandidestroy:~# ls WordCount/src/
WordCount.java
root@Jangandidestroy:~#
```

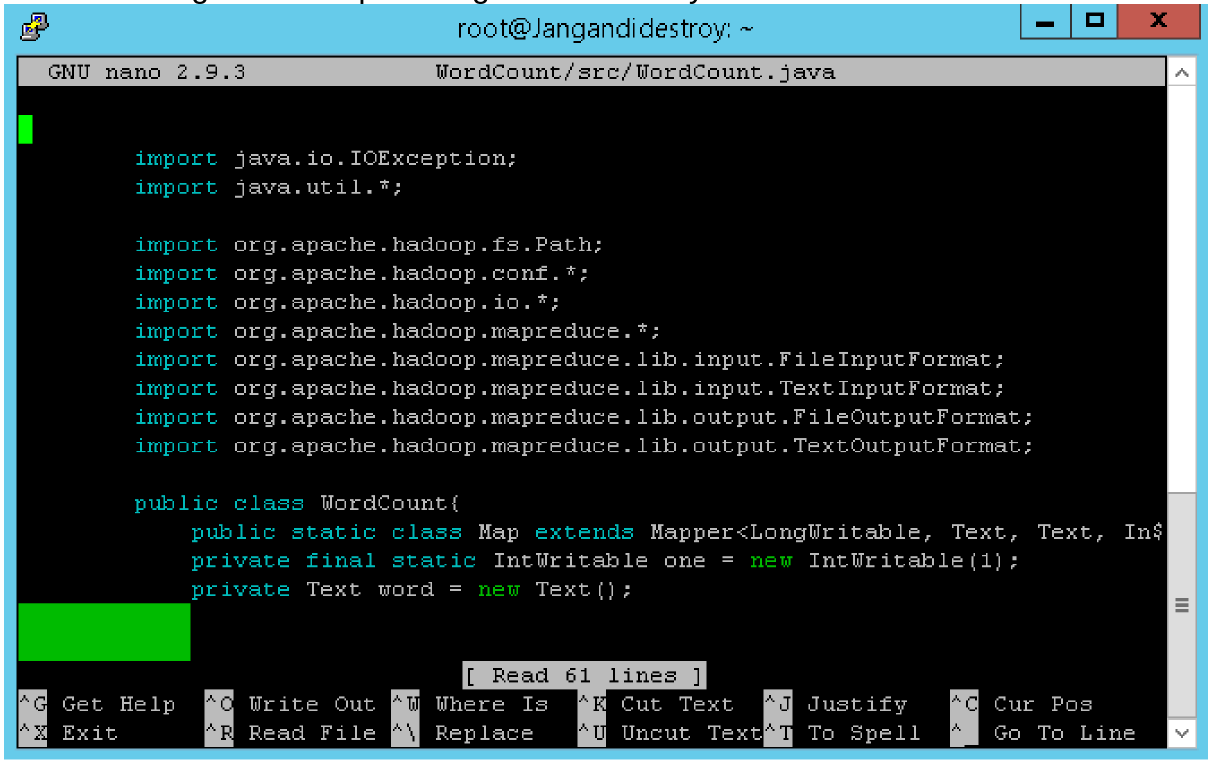
9. At this stage, write the command nano WordCount/src/WordCount.java to create a Java format file that will be used to create the program.

10. After running the command nano WordCount/src/WordCount.java, fill it with the following source code.

```
import java.io.IOException;
        import java.util.*;

        import org.apache.hadoop.fs.Path;
        import org.apache.hadoop.conf.*;
        import org.apache.hadoop.io.*;
        import org.apache.hadoop.mapreduce.*;
        import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class WordCount{
   public static class Map extends Mapper<LongWritable, Text, Text, IntWritable>{
   private final static IntWritable one = new IntWritable(1);
   private Text word = new Text();


   public void map(LongWritable key,Text value,Context context) throws IOException,
   InterruptedException{
   String line = value.toString();
   StringTokenizer tokenizer = new StringTokenizer(line);
   while (tokenizer.hasMoreTokens()){
   word.set(tokenizer.nextToken());
   context.write(word,one);

   }
   }
   }
   public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable>{
   public void reduce(Text key, Iterable<IntWritable>values, Context context)

   throws IOException, InterruptedException{
          int sum = 0;
          for (IntWritable val : values){
          sum += val.get();
          }
          context.write(key, new IntWritable(sum));
   }
   }
   public static void main (String[] args) throws Exception {
          Configuration conf = new Configuration();

          @SuppressWarnings("deprecation")
          Job job = new Job(conf, "wordcount");
          job.setJarByClass(WordCount.class);
          job.setOutputKeyClass(Text.class);
          job.setOutputValueClass(IntWritable.class);
          job.setMapperClass(Map.class);
          job.setReducerClass(Reduce.class);
          job.setInputFormatClass(TextInputFormat.class);
          job.setOutputFormatClass(TextOutputFormat.class);
```

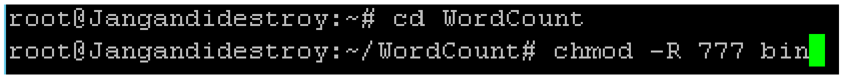
```
                        FileInputFormat.addInputPath(job, new Path(args[0]));
                        FileOutputFormat.setOutputPath(job, new Path(args[1]));
                        job.waitForCompletion(true);

                }
                }
```

11. Lalu save dengan cara seperti langkah sebelumnya.



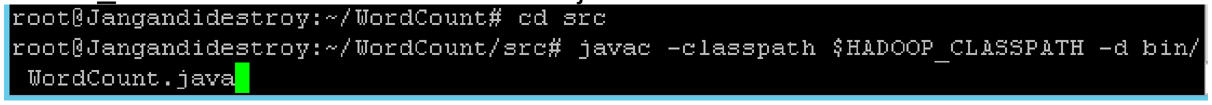
```
  GNU nano 2.9.3                      WordCount/src/WordCount.java


        import java.io.IOException;
        import java.util.*;

        import org.apache.hadoop.fs.Path;
        import org.apache.hadoop.conf.*;
        import org.apache.hadoop.io.*;
        import org.apache.hadoop.mapreduce.*;
        import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
        import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
        import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
        import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

        public class WordCount{
            public static class Map extends Mapper<LongWritable, Text, Text, In$
            private final static IntWritable one = new IntWritable(1);
            private Text word = new Text();


                                              [ Read 61 lines ]
^G Get Help     ^O Write Out  ^W Where Is    ^K Cut Text   ^J Justify     ^C Cur Pos
^X Exit         ^R Read File  ^\ Replace     ^U Uncut Text^T To Spell     ^_ Go To Line
```

12. After creating the WordCount.java file, go to the WordCount folder directory using the command below and type the command chmod -R 777 bin to change the permissions of the bin folder in the WordCount folder.



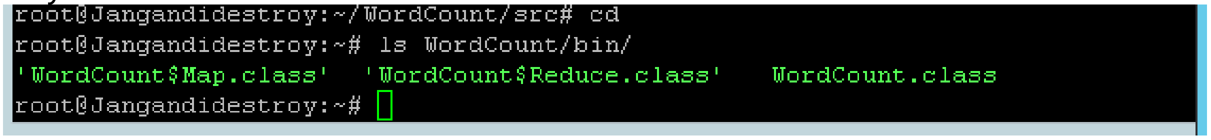
```
root@Jangandidestroy:~# cd WordCount
root@Jangandidestroy:~/WordCount# chmod -R 777 bin
```

13. Execute the cd command to src and type the command javac -classpath $HADOOP_CLASSPATH -d bin/ WordCount.java



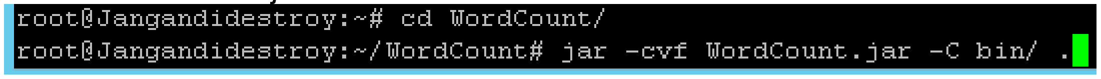
```
root@Jangandidestroy:~/WordCount# cd src
root@Jangandidestroy:~/WordCount/src# javac -classpath $HADOOP_CLASSPATH -d bin/
 WordCount.java
```

14. View the compiler results in the bin folder, but type the cd command to return to the initial directory and follow it with the ls WordCount/bin/ command.



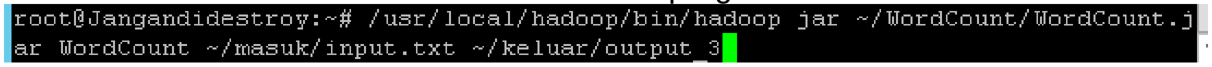
```
root@Jangandidestroy:~/WordCount/src# cd
root@Jangandidestroy:~# ls WordCount/bin/
'WordCount$Map.class'  'WordCount$Reduce.class'   WordCount.class
root@Jangandidestroy:~#
```

15. Type the command cd and enter the WordCount directory, then run the following command to convert it to a jar executable file format.



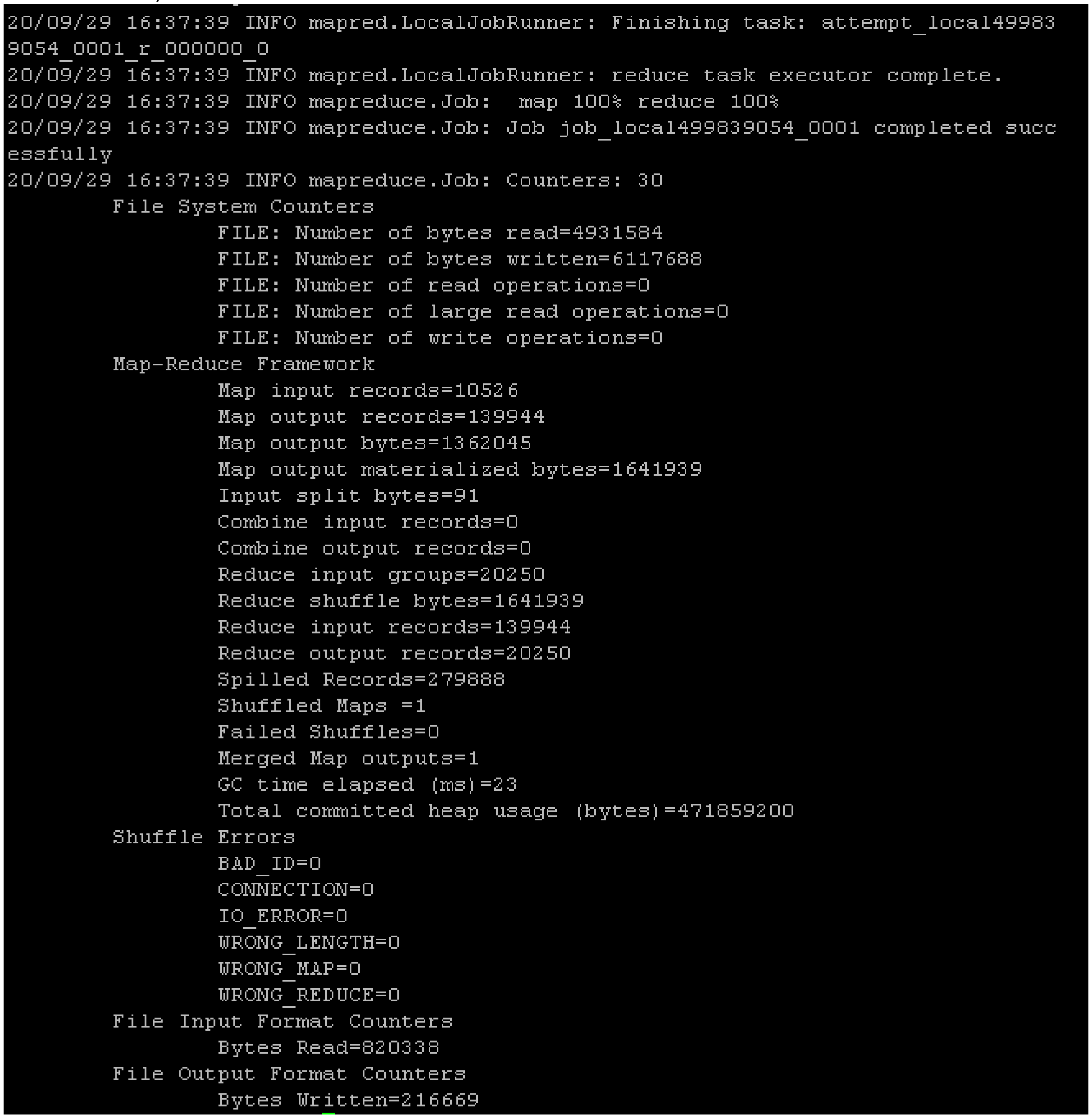
```
root@Jangandidestroy:~# cd WordCount/
root@Jangandidestroy:~/WordCount# jar -cvf WordCount.jar -C bin/ .
```

16. Then write the command to run the WordCount program.



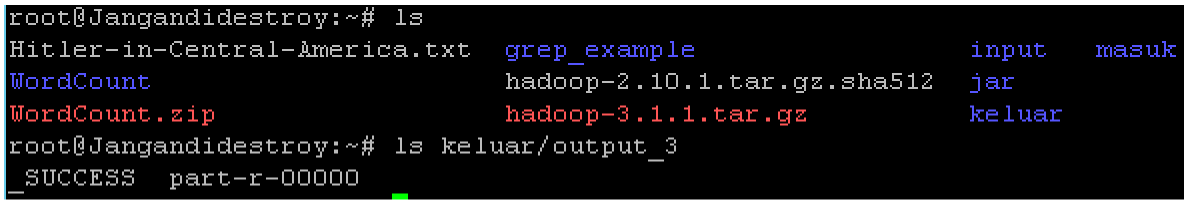
```
root@Jangandidestroy:~# /usr/local/hadoop/bin/hadoop jar ~/WordCount/WordCount.j
ar WordCount ~/masuk/input.txt ~/keluar/output_3
```

17. If successful, it will look like this.



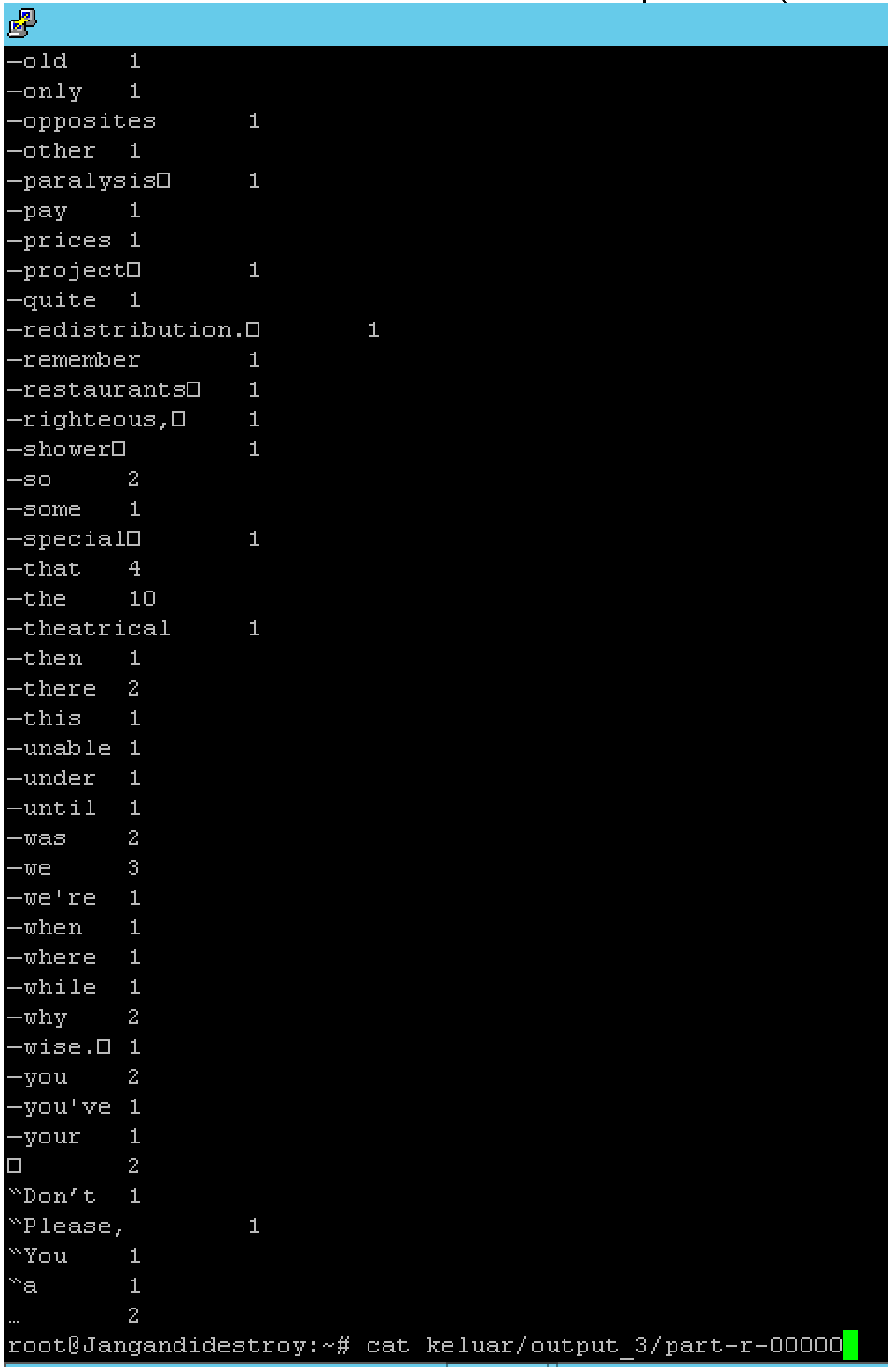
```
20/09/29 16:37:39 INFO mapred.LocalJobRunner: Finishing task: attempt_local49983
9054_0001_r_000000_0
20/09/29 16:37:39 INFO mapred.LocalJobRunner: reduce task executor complete.
20/09/29 16:37:39 INFO mapreduce.Job:  map 100% reduce 100%
20/09/29 16:37:39 INFO mapreduce.Job: Job job_local499839054_0001 completed succ
essfully
20/09/29 16:37:39 INFO mapreduce.Job: Counters: 30
        File System Counters
                FILE: Number of bytes read=4931584
                FILE: Number of bytes written=6117688
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
        Map-Reduce Framework
                Map input records=10526
                Map output records=139944
                Map output bytes=1362045
                Map output materialized bytes=1641939
                Input split bytes=91
                Combine input records=0
                Combine output records=0
                Reduce input groups=20250
                Reduce shuffle bytes=1641939
                Reduce input records=139944
                Reduce output records=20250
                Spilled Records=279888
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=23
                Total committed heap usage (bytes)=471859200
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=820338
        File Output Format Counters
                Bytes Written=216669
```

18. To view the MapReduce output file, type the following command.



```
root@Jangandidestroy:~# ls
Hitler-in-Central-America.txt  grep_example                   input   masuk
WordCount                      hadoop-2.10.1.tar.gz.sha512    jar
WordCount.zip                  hadoop-3.1.1.tar.gz            keluar
root@Jangandidestroy:~# ls keluar/output_3
_SUCCESS  part-r-00000
```

19. The results of the MapReduce algorithm are stored in the file part-r-00000. To view the contents, type the command cat output/output_3/part-r-00000 and press Enter. This will display the number of words from the text we entered in the input.txt file (case sensitive).



```
—old    1
—only   1
—opposites      1
—other  1
—paralysis□     1
—pay    1
—prices 1
—project□       1
—quite  1
—redistribution.□       1
—remember       1
—restaurants□   1
—righteous,□    1
—shower□        1
—so     2
—some   1
—special□       1
—that   4
—the    10
—theatrical     1
—then   1
—there  2
—this   1
—unable 1
—under  1
—until  1
—was    2
—we     3
—we're  1
—when   1
—where  1
—while  1
—why    2
—wise.□ 1
—you    2
—you've 1
—your   1
□       2
“Don’t  1
“Please,        1
“You    1
“a      1
…       2
root@Jangandidestroy:~# cat keluar/output_3/part-r-00000
```