LAB - 03

AIM: Learn concepts of mapreduce programming.

EXERCISE:

1. Compile Word Count java program and be able to provide HDFS input directory containing text files, execute and verify output files from the resulting folder within HDFS.

I've created a java file named WordCount.java

The WordCount program in Java using Hadoop MapReduce will read the input from an HDFS directory, count the frequency of each word, and then output the result to another HDFS directory.

Code:

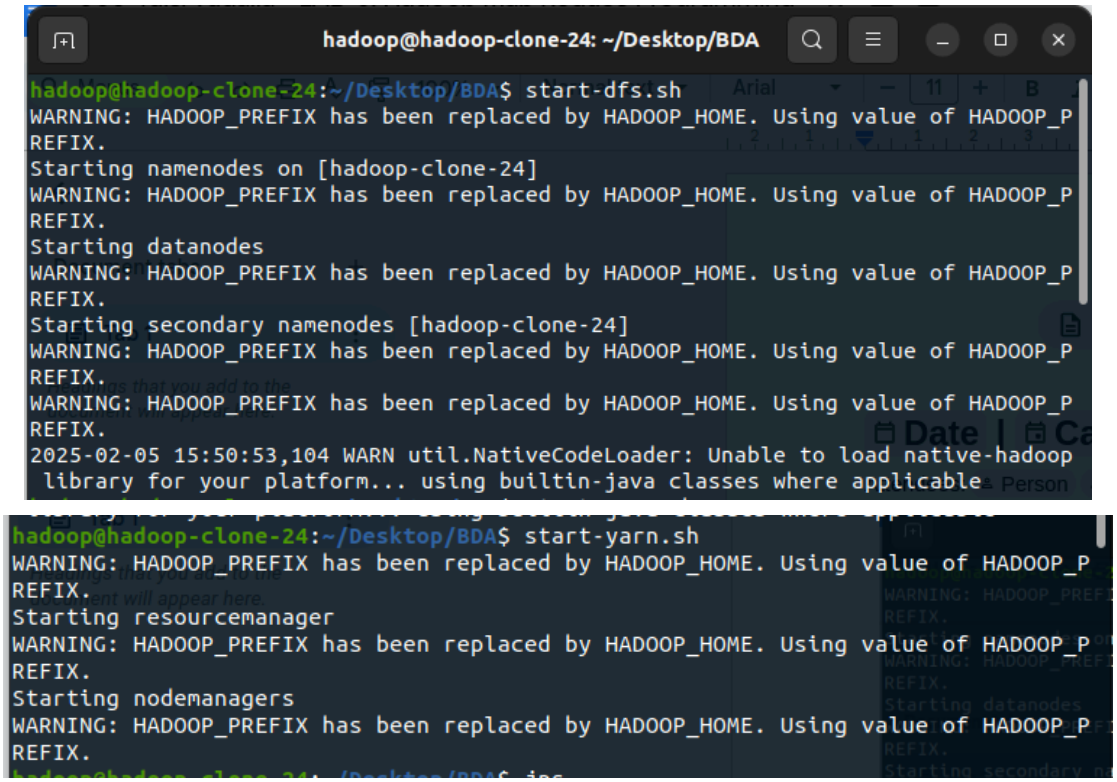
```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount
{
     public static class TokenizerMapper 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
        {
          StringTokenizer tokenizer = new StringTokenizer(value.toString());
```

```
            while (tokenizer.hasMoreTokens()) {
                word.set(tokenizer.nextToken());
                 context.write(word, one);
             }
          }
      }
public static class IntSumReducer extends Reducer < Text, IntWritable,
             Text, IntWritable >
      {
         private IntWritable result = new IntWritable();
         public void reduce(Text key, Iterable < IntWritable > values, Context context)
throws
                  IOException, InterruptedException
         {
            int sum = 0;
  for (IntWritable val: values) {
           sum += val.get();
        }
               result. set (sum);

               context. write(key, result);
      }
  }
  public static void main(String[] args) throws Exception
  {
      Configuration conf = new Configuration();
      Job job = Job.getInstance(conf, "word count");
            job. setJarByClass(WordCount.class);
            job. setMapperClass(TokenizerMapper.class);
            job. setCombinerClass(IntSumReducer.class);
            job. setReducerClass(IntSumReducer.class);
            job. setOutputKeyClass(Text.class);
            job. setOutputValueClass(IntWritable.class);
            FileInputFormat.addInputPath(job, new Path(args[0]));
            FileOutputFormat.setOutputPath(job, new Path(args[1]));
            System. exit (job.waitForCompletion(true) ? 0 : 1);
   }
}
```

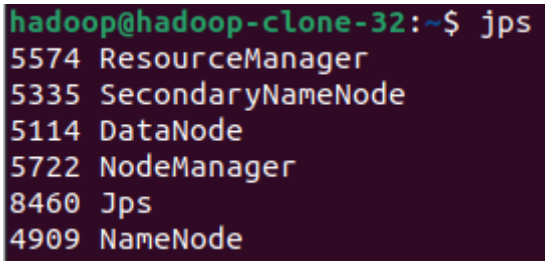
Now, we start dfs and yarn services using:

start-dfs.sh

start-yarn.sh

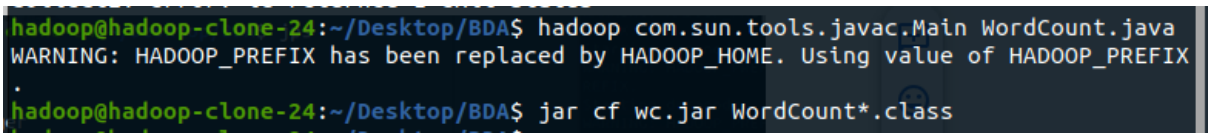


Running jps and verifying as below:



Then, compiling WordCount.java and creating a jar: (going to the directory where we have the WordCount.java file)

Assuming environment variables are set appropriately:



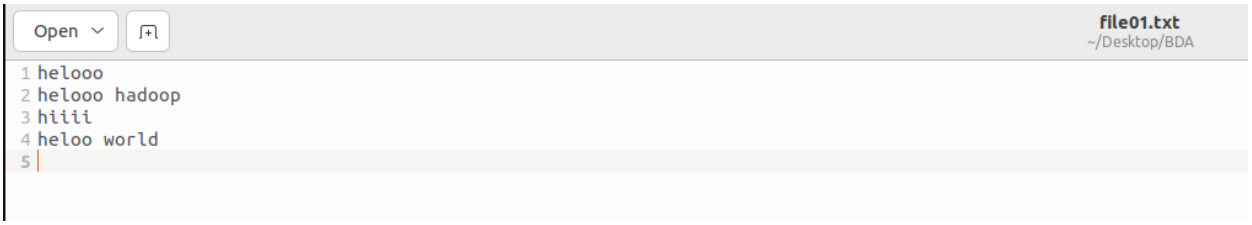
```
hadoop@hadoop-clone-24:~/Desktop/BDA$ cat /opt/hadoop/etc/hadoop/mapred-site.xml<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>

  <property>
    <name>mapreduce.jobhistory.address</name>
    <value>192.168.28.24:10020</value>
  </property>

  <property>
    <name>mapreduce.jobhistory.webapp.address</name>
    <value>192.168.28.24:19888</value>
  </property>
<property>
  <name>yarn.app.mapreduce.am.env</name>
  <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
</property>
<property>
  <name>mapreduce.map.env</name>
  <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
</property>
<property>
  <name>mapreduce.reduce.env</name>
  <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
</property>

</configuration>
```

```
hadoop@hadoop-clone-24:~/Desktop/BDA$  nano cat /opt/hadoop/etc/hadoop/mapred-site.xml
```

```
hadoop@hadoop-clone-24:~/Desktop/BDA$ cat /opt/hadoop/etc/hadoop/mapred-site.xml
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>

  <property>
    <name>mapreduce.jobhistory.address</name>
    <value>localhost:10020</value>
  </property>

  <property>
    <name>mapreduce.jobhistory.webapp.address</name>
    <value>localhost:19888</value>
  </property>
<property>
  <name>yarn.app.mapreduce.am.env</name>
  <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
</property>
<property>
  <name>mapreduce.map.env</name>
  <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
</property>
<property>
  <name>mapreduce.reduce.env</name>
  <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
</property>

</configuration>
hadoop@hadoop-clone-24:~/Desktop/BDA$
```

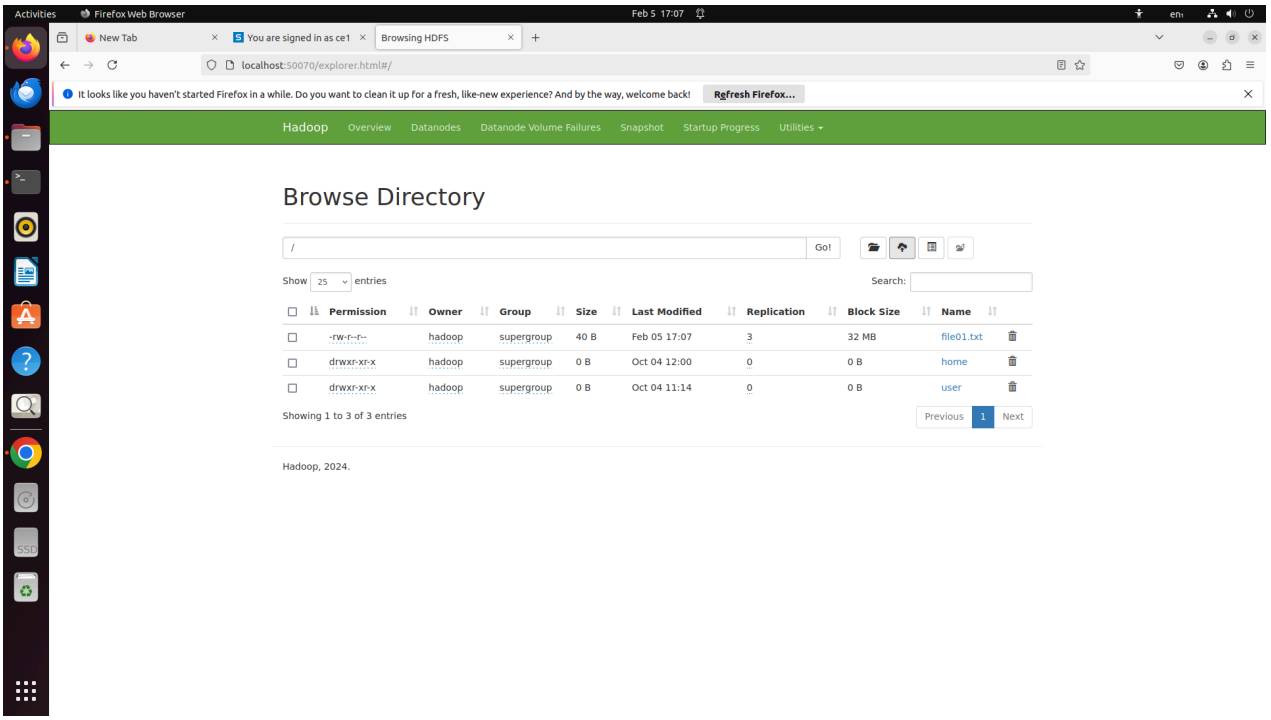
After this, storing the input data file in HDFS

Afterwards, creating a text file in local machine (file01.txt)
Path of the file on local machine - */home/hadoop/Desktop/file01.txt*



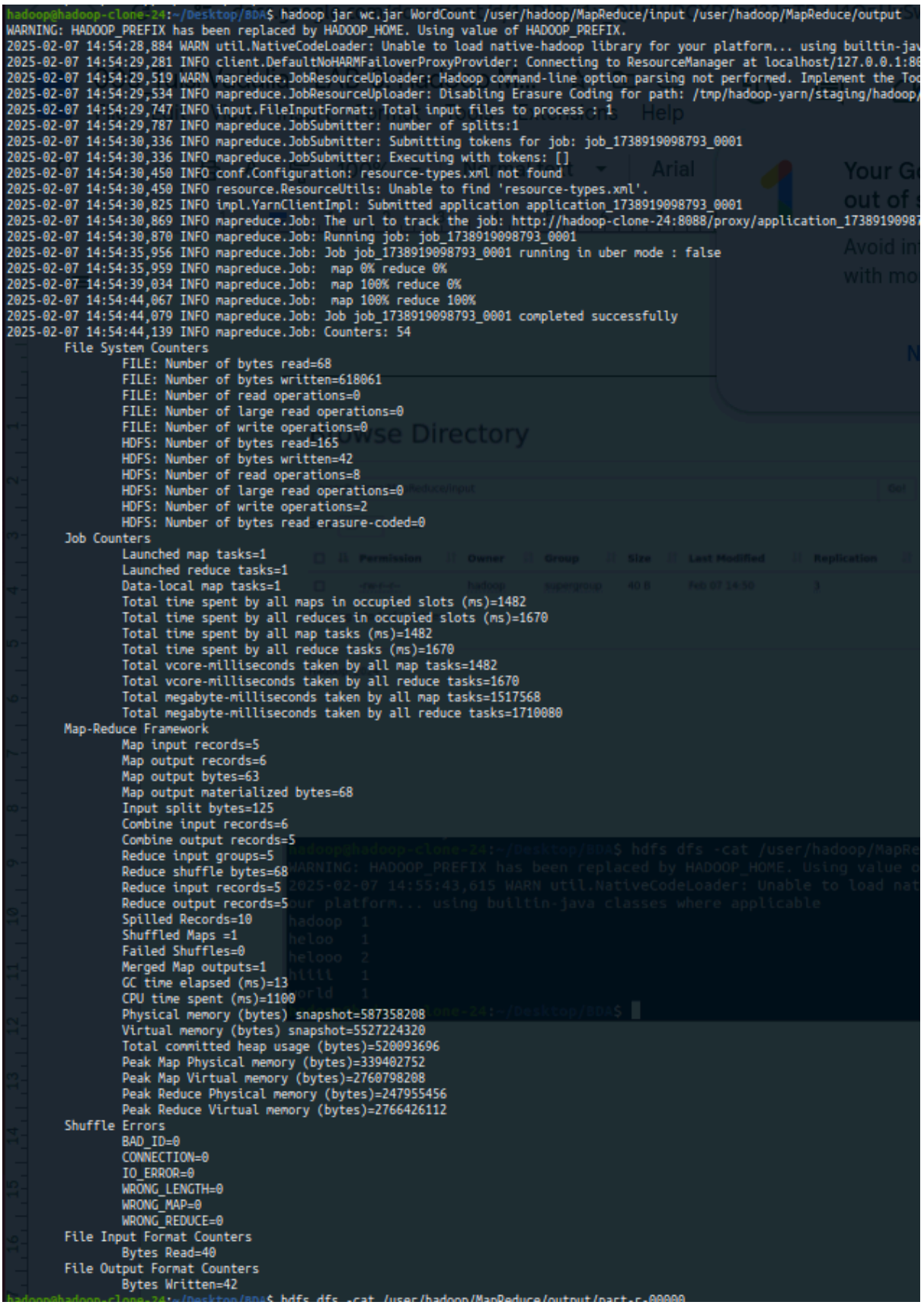
Creating a directory to store our input data file
hdfs dfs -mkdir /wordcountdemo/input

And putting the file file1.txt from local machine to HDFS directory
hdfs dfs -put /home/hadoop/Desktop/fileo1.txt /wordcountdemo/input

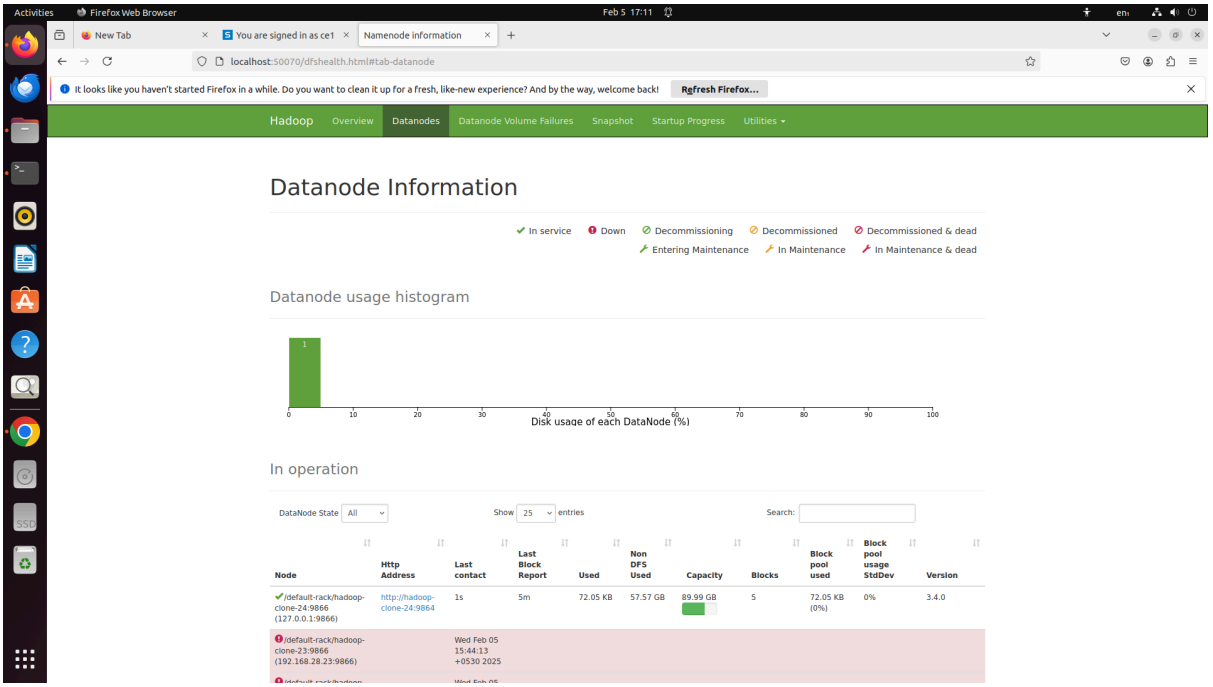


2. Write a MapReduce java program that reads any text input and computes the average length of all words that start with each character.

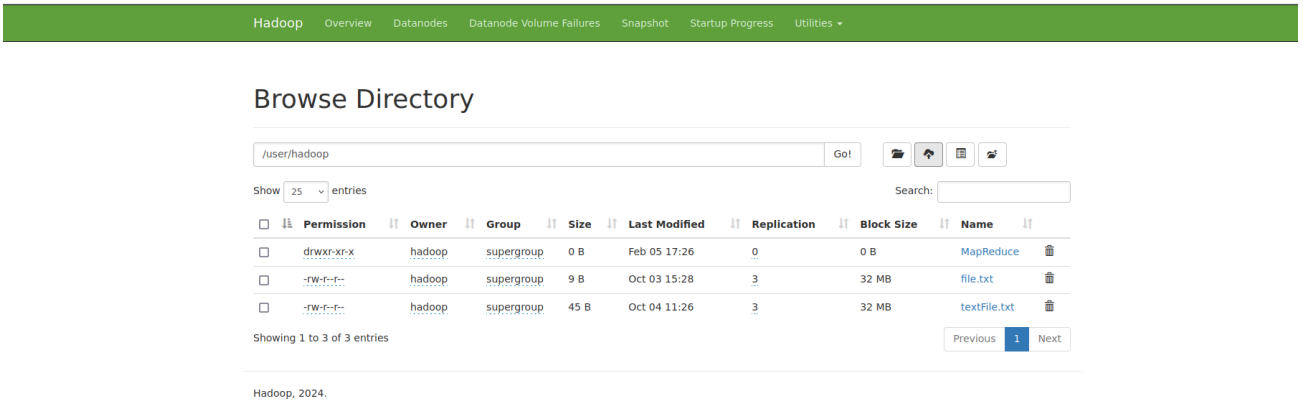
When completed the MapReduce task, run the following command to see the output:
Syntax: *hadoop jar <Path of the jar file> <classfilename> <Path of the input file> <path of output file>*



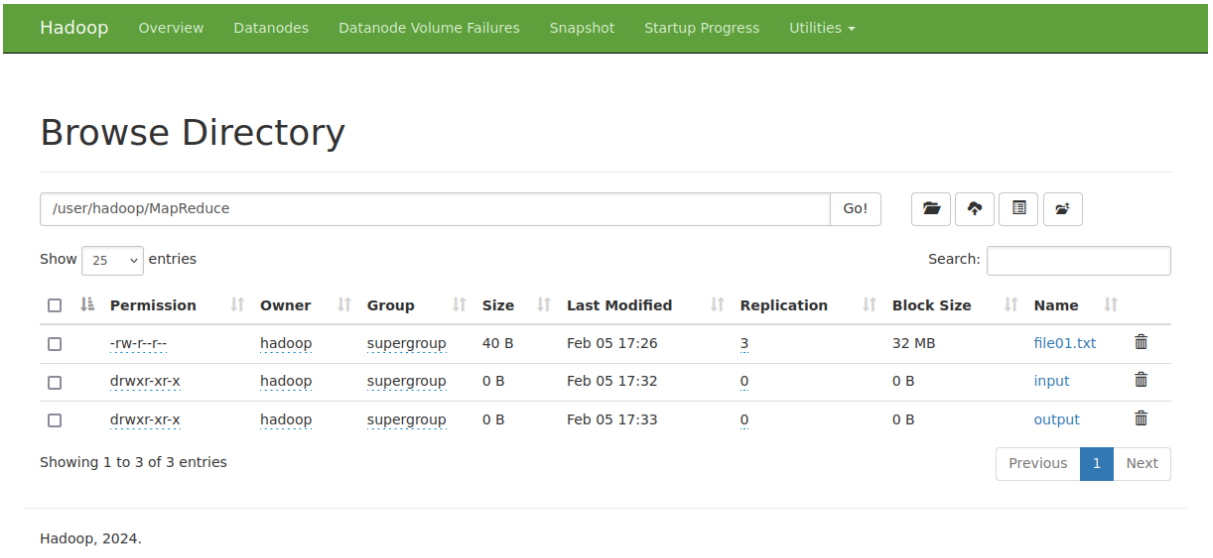
```
hadoop@hadoop-clone-24:~/Desktop/BDA$ hadoop jar wc.jar WordCount /user/hadoop/MapReduce/input /user/hadoop/MapReduce/output
WARNING: HADOOP_PREFIX has been replaced by HADOOP_HOME. Using value of HADOOP_PREFIX.
2025-02-07 14:54:28,884 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-jav
2025-02-07 14:54:29,281 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at localhost/127.0.0.1:80
2025-02-07 14:54:29,519 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Too
2025-02-07 14:54:29,534 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/hadoop/
2025-02-07 14:54:29,747 INFO input.FileInputFormat: Total input files to process : 1
2025-02-07 14:54:29,787 INFO mapreduce.JobSubmitter: number of splits:1
2025-02-07 14:54:30,336 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1738919098793_0001
2025-02-07 14:54:30,336 INFO mapreduce.JobSubmitter: Executing with tokens: []
2025-02-07 14:54:30,450 INFO conf.Configuration: resource-types.xml not found
2025-02-07 14:54:30,450 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2025-02-07 14:54:30,825 INFO impl.YarnClientImpl: Submitted application application_1738919098793_0001
2025-02-07 14:54:30,869 INFO mapreduce.Job: The url to track the job: http://hadoop-clone-24:8088/proxy/application_17389190987
2025-02-07 14:54:30,870 INFO mapreduce.Job: Running job: job_1738919098793_0001
2025-02-07 14:54:35,956 INFO mapreduce.Job: Job job_1738919098793_0001 running in uber mode : false
2025-02-07 14:54:35,959 INFO mapreduce.Job:  map 0% reduce 0%
2025-02-07 14:54:39,034 INFO mapreduce.Job:  map 100% reduce 0%
2025-02-07 14:54:44,067 INFO mapreduce.Job:  map 100% reduce 100%
2025-02-07 14:54:44,079 INFO mapreduce.Job: Job job_1738919098793_0001 completed successfully
2025-02-07 14:54:44,139 INFO mapreduce.Job: Counters: 54
        File System Counters
                FILE: Number of bytes read=68
                FILE: Number of bytes written=618061
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=165
                HDFS: Number of bytes written=42
                HDFS: Number of read operations=8
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
                HDFS: Number of bytes read erasure-coded=0
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=1482
                Total time spent by all reduces in occupied slots (ms)=1670
                Total time spent by all map tasks (ms)=1482
                Total time spent by all reduce tasks (ms)=1670
                Total vcore-milliseconds taken by all map tasks=1482
                Total vcore-milliseconds taken by all reduce tasks=1670
                Total megabyte-milliseconds taken by all map tasks=1517568
                Total megabyte-milliseconds taken by all reduce tasks=1710080
        Map-Reduce Framework
                Map input records=5
                Map output records=6
                Map output bytes=63
                Map output materialized bytes=68
                Input split bytes=125
                Combine input records=6
                Combine output records=5
                Reduce input groups=5
                Reduce shuffle bytes=68
                Reduce input records=5
                Reduce output records=5
                Spilled Records=10
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=13
                CPU time spent (ms)=1100
                Physical memory (bytes) snapshot=587358208
                Virtual memory (bytes) snapshot=5527224320
                Total committed heap usage (bytes)=520093696
                Peak Map Physical memory (bytes)=339402752
                Peak Map Virtual memory (bytes)=2760798208
                Peak Reduce Physical memory (bytes)=247955456
                Peak Reduce Virtual memory (bytes)=2766426112
        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=40
        File Output Format Counters
                Bytes Written=42
```

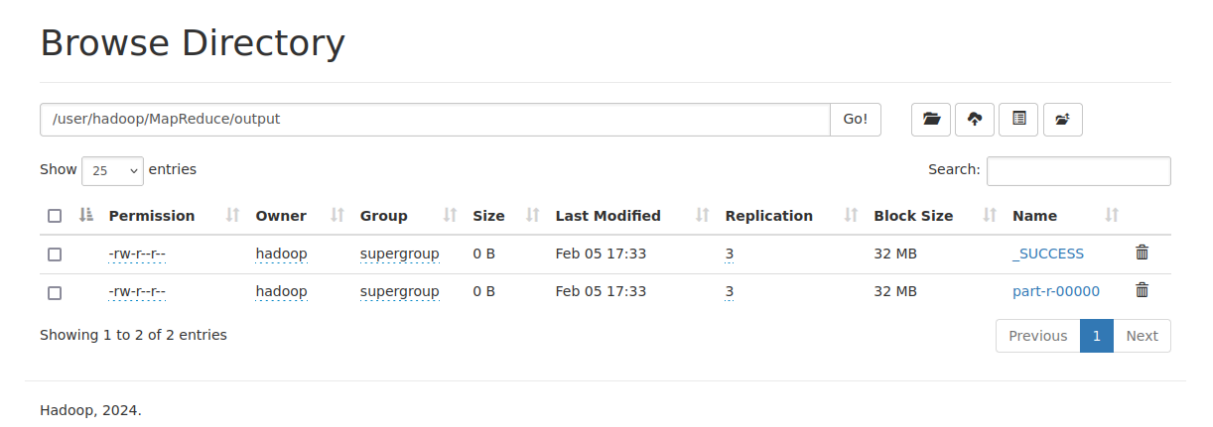


Java program jar based map-reduce job will be submitted to ResourceManager everytime wordcount is expected for set of input files.

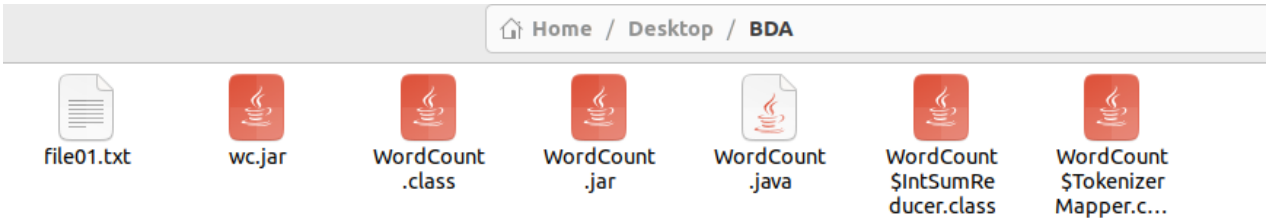


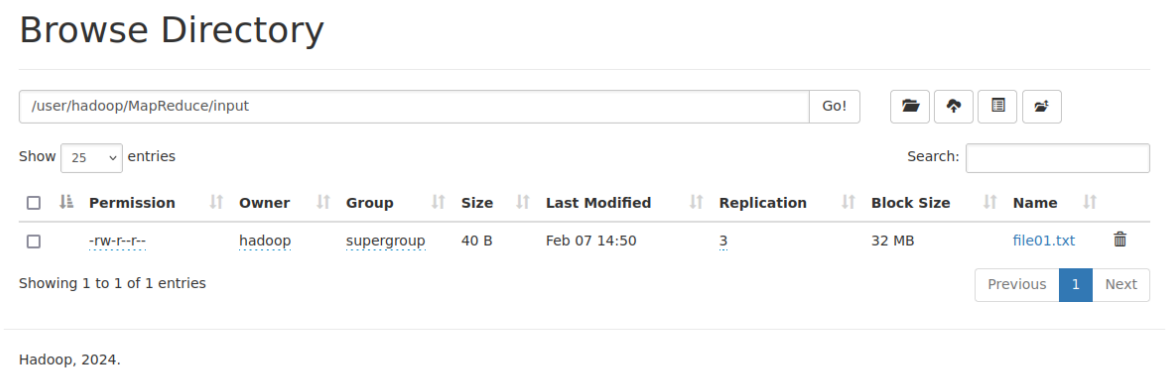
Input files containing plain text need to be stored into HDFS location. This HDFS location will be provided to the program as a command line argument - input. Also, the second command line argument -input will be the location of HDFS where the resulting files - output have to be stored.





This is how the /home/desktop location on the device looks like:





Mapper processes and generates a set of key-value pairs containing words with frequency 1 for every words in the current line for the WordCount program.

```
hadoop@hadoop-clone-24:~/Desktop/BDA$ hdfs dfs -cat /user/hadoop/MapReduce/output/part-r-00000
WARNING: HADOOP_PREFIX has been replaced by HADOOP_HOME. Using value of HADOOP_PREFIX.
2025-02-07 14:55:43,615 WARN util.NativeCodeLoader: Unable to load native-hadoop library for y
our platform... using builtin-java classes where applicable
hadoop  1
heloo   1
helooo  2
hiiii   1
world   1
hadoop@hadoop-clone-24:~/Desktop/BDA$
```

After the execution of mapper, the framework behind the scene shuffles all resulting pairs and sorts to generate <key, list of values> kind of pair where all values for the same key is added into values list. In word count program entries into values list will be number of 1's for a certain word.

Summarised learning:

I learned the concepts of mapreduce programming. Word Count - A type of hello world program with respect to Map Reduce distributed processing framework utilizing data primarily from Hadoop Distributed File System (HDFS). Both programs will produce output in the form of key-value pairs. Also, I learnt about the framework itself and identified the phases involved in which can be exploited for solutions about Big Data Analytics.

Thus, the task helped me understand how to work with HDFS for storing input and output files and how to handle text processing and aggregation using MapReduce concepts.