School of Computer Engineering and Technology

Course : PE1 Big Data Analytics Year : TY BTech Tri:IX

Lab Assignment 4

Problem Statement : Implement Map-reduce operation in Hadoop

Demo Program : Word Count Application
Pre-requisite : Apache Hadoop should be already installed.

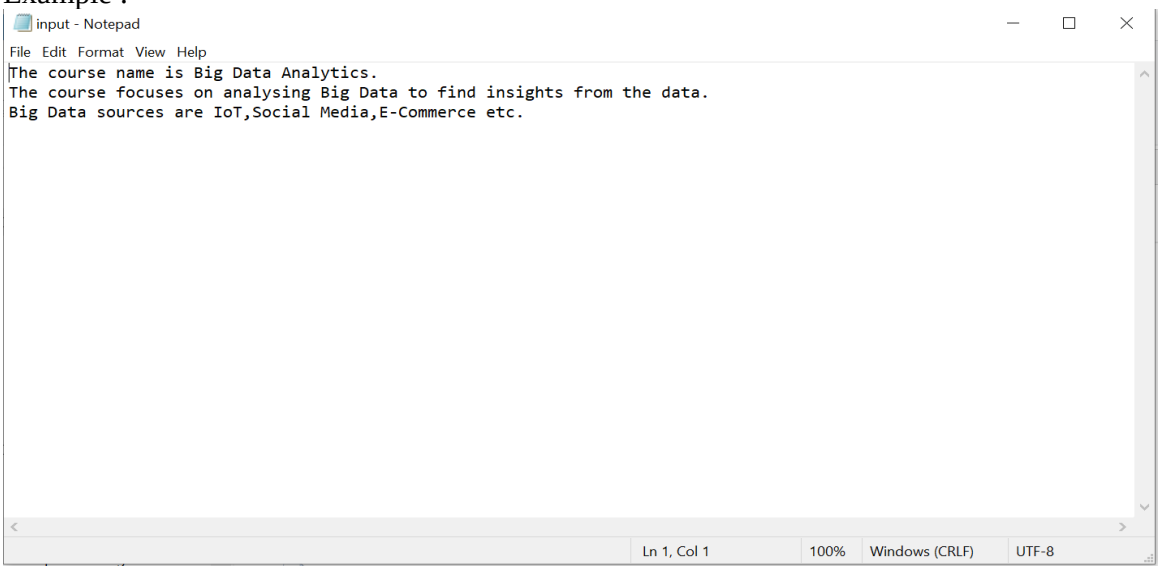
Map Reduce Program for counting the occurrences of words in a text document available on HDFS.

STEPS :

1. Create the input source file and store it in a drive in local file system.

   Example: E:\input.txt

Example :

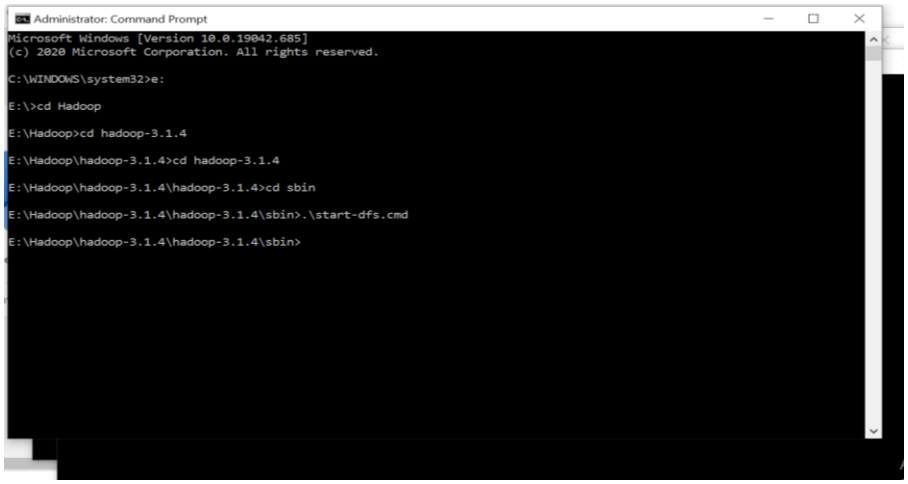


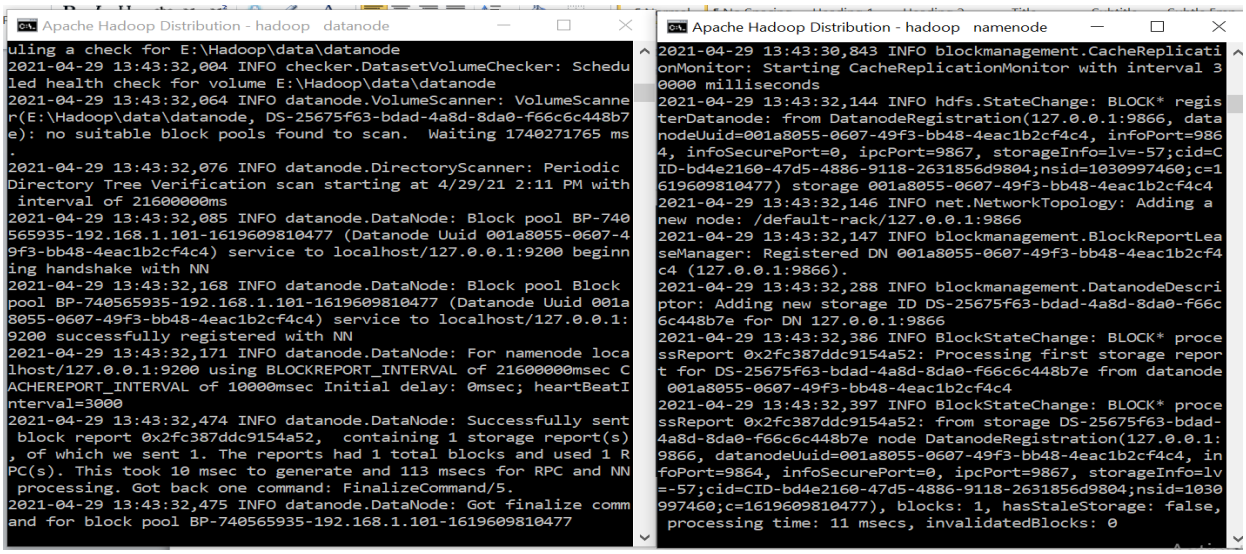
2. Start the Hadoop File system Service using following command from command prompt.

Go to the sbin folder path in Hadoop :

E:\Hadoop\hadoop-3.1.4\hadoop-3.1.4\sbin and type following command to start the namenode and datanode processes :

.\start-dfs.cmd



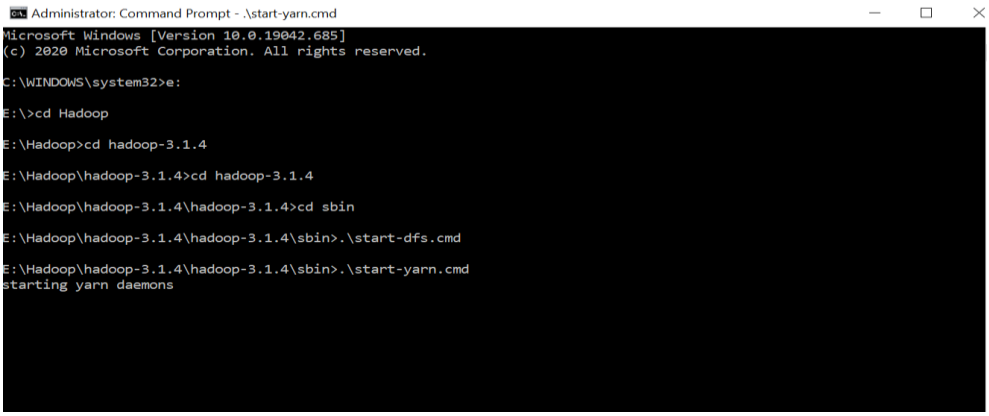


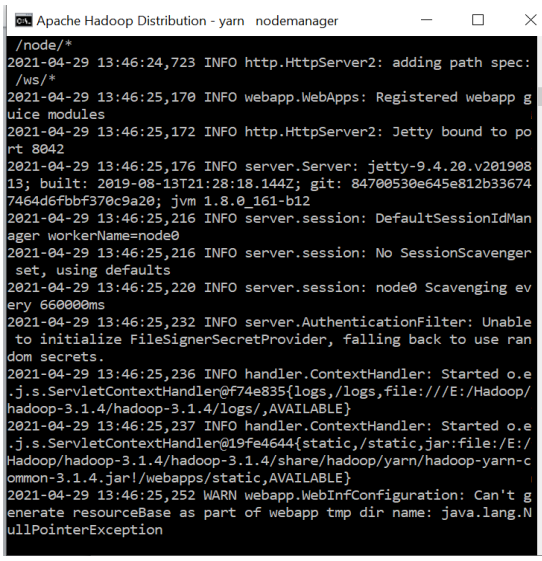
Once the namenode and datanode processes have successfully started,

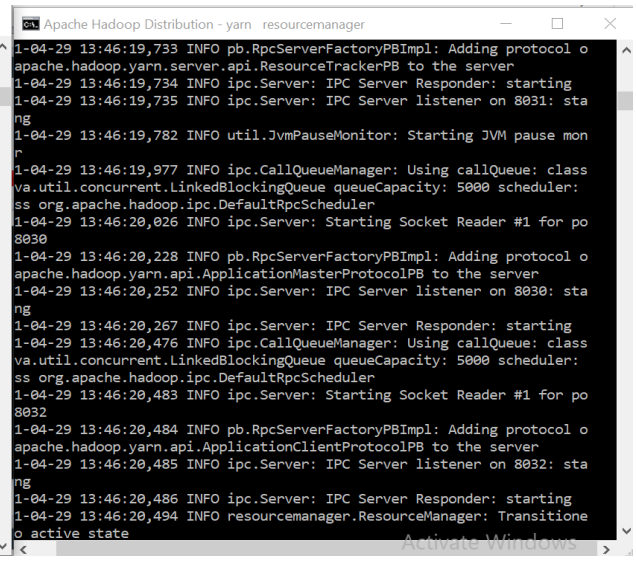
3. Start the yarn service using following command:

.\start-yarn.cmd

This command will start execution of two processes: ResourceManager and NodeManager respectively.







After both services have started perform following:

4. Copy the input.text file from local system to Hadoop File System using following command

hadoop dfs –copyFromLocal E:\input.txt /Vasundhara/

Confirm whether file is copied using following command

hadoop dfs –ls /Vasundhara/

5. Once file is available on HDFS, write the Map-reduce program in Eclipse/Netbeans IDE.

Number of programs to be created : 3 under the same project

1. word_Mapper : to provide mapper functionality code

2. word_Reducer : to provide reducer functionality code

3. word_Driver : Will have the main Method to execute the Map-reduce functionality.

JAR files to be included in the Java Project : 3

Following JAR files are required to be added in the Java Project.

1. hadoop-mapreduce-client-core-3.1.4.jar

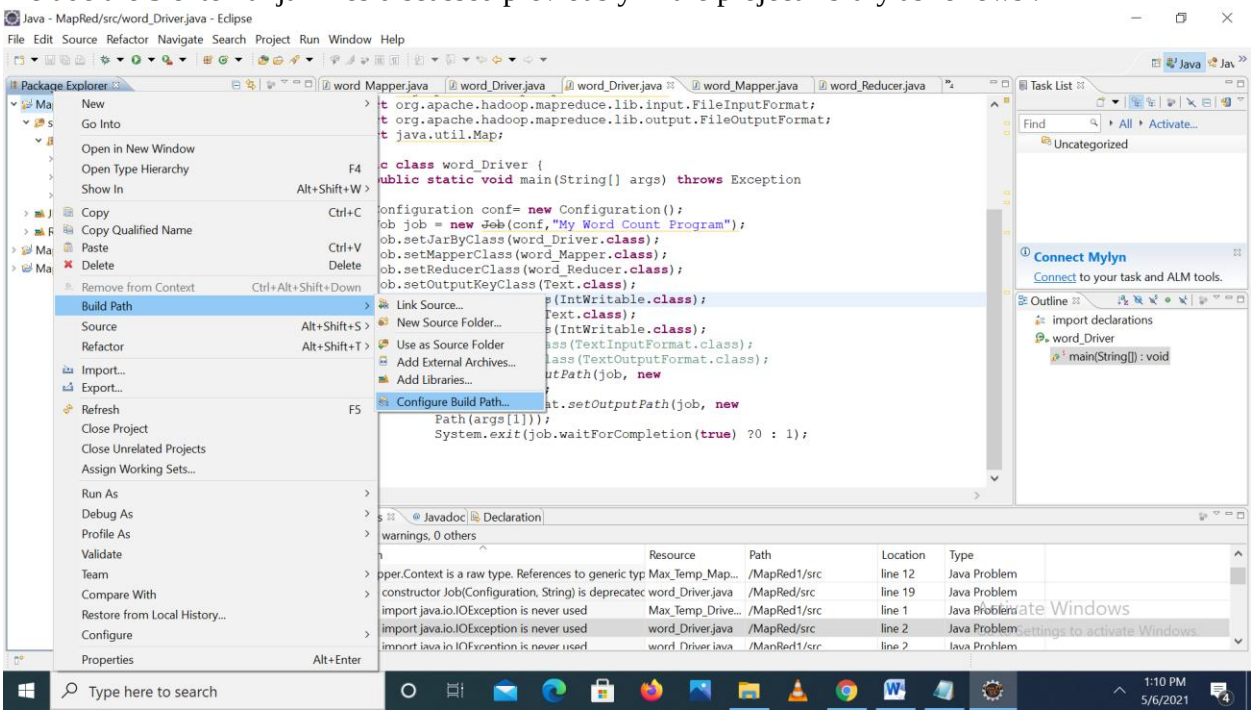
   Above file is available in the following folder path :

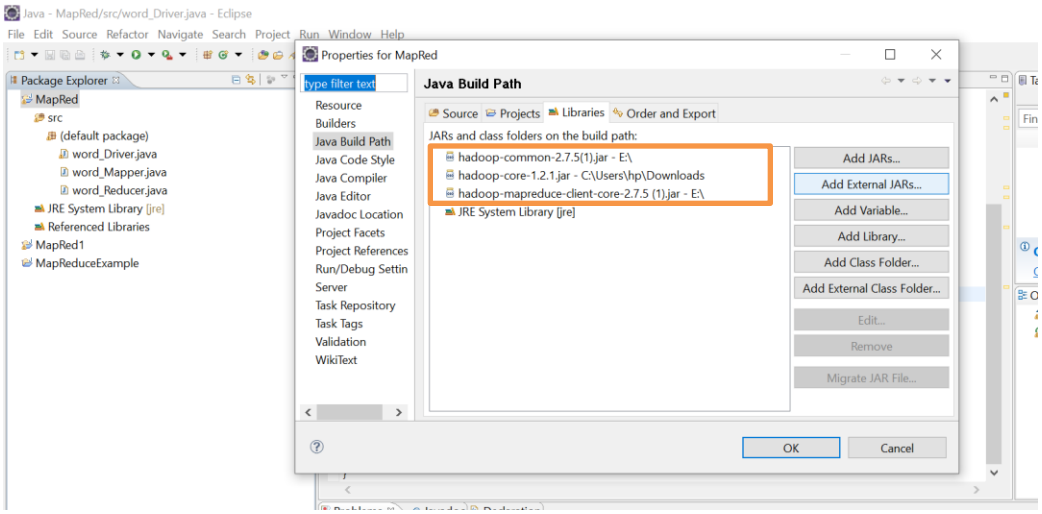
   E:\Hadoop\hadoop-3.1.4\hadoop-3.1.4\share\hadoop\mapreduce

2. hadoop-core-1.2.1 (Download this jar file)
3. hadoop-common-2.7.5.jar(Download this jar file)

Open Netbeans/eclipse->create new java project

Include the 3 external jar files discussed previously in the project library as follows :





In the Project src folder create 3 classes as follows :
1.word_Mapper (without main method)
2.word_Reducer (without main method)
3.word_Driver(with main method)

1. word_Mapper.java

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

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class word_Mapper extends Mapper<LongWritable,Text,Text,IntWritable>
{
    public void map(LongWritable key, Text value, Context context) throws IOException,InterruptedException {
        String line = value.toString();
        StringTokenizer tokenizer = new StringTokenizer(line);
        while (tokenizer.hasMoreTokens()) {
            value.set(tokenizer.nextToken());
            context.write(value, new IntWritable(1));
         }
}}
```

2. word_Reducer.java

```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;


public class word_Reducer extends Reducer <Text,IntWritable,Text,IntWritable>
```

```
{
    public void reduce(Text key, Iterable <IntWritable> values,Context context)
        throws IOException,InterruptedException {
            int sum=0;
            for(IntWritable x: values)
            {
                sum+=x.get();
            }
            context.write(key, new IntWritable(sum));
        }
    }
```

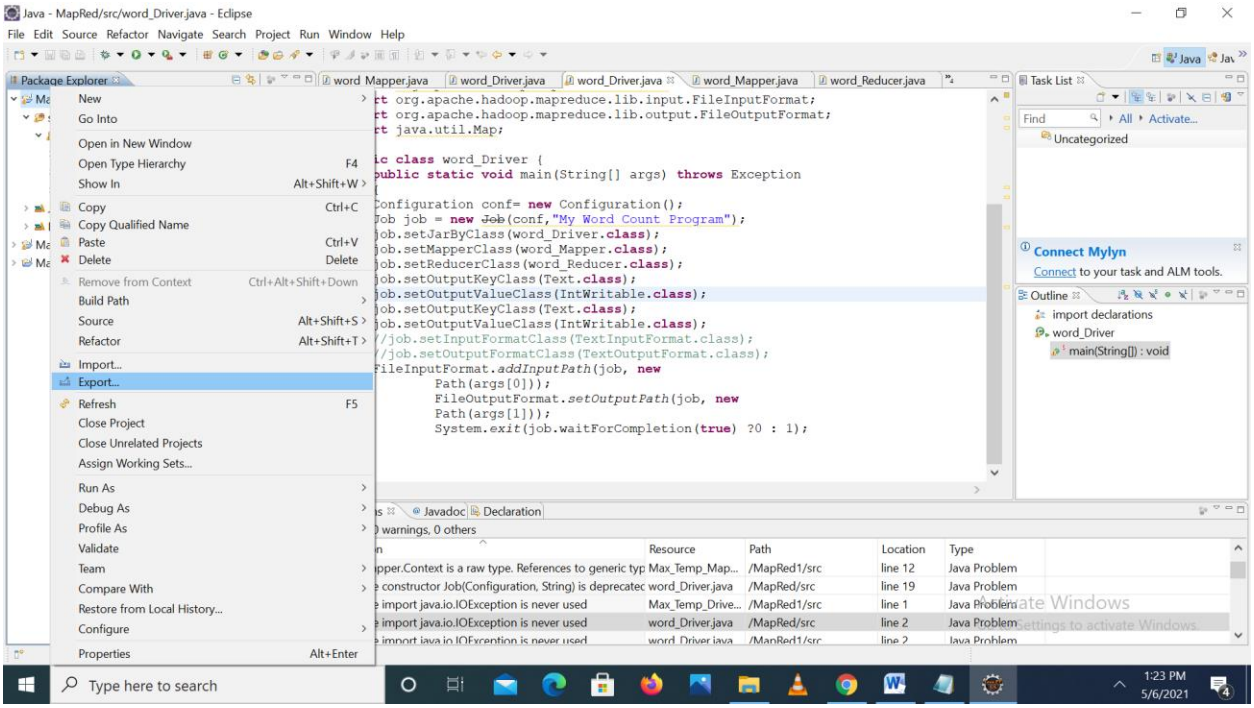
3. word_Driver.java

```
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.IntWritable;
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;
import java.util.Map;

public class word_Driver {
        public static void main(String[] args) throws Exception
        {
        Configuration conf= new Configuration();
        Job job = new Job(conf,"My Word Count Program");
        job.setJarByClass(word_Driver.class);
        job.setMapperClass(word_Mapper.class);
        job.setReducerClass(word_Reducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        //job.setInputFormatClass(TextInputFormat.class);
        //job.setOutputFormatClass(TextOutputFormat.class);
        FileInputFormat.addInputPath(job, new
                        Path(args[0]));
                        FileOutputFormat.setOutputPath(job, new
                        Path(args[1]));
                        System.exit(job.waitForCompletion(true) ?0 : 1);

}
}
```

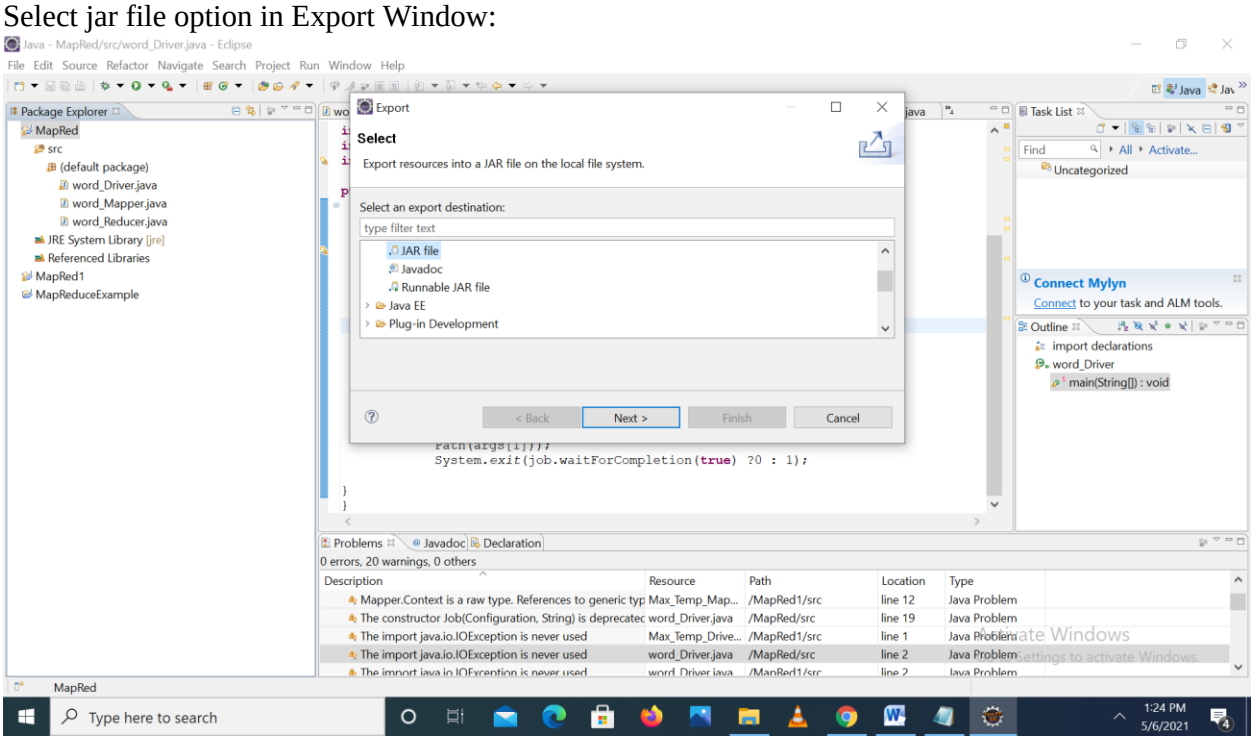
Save all the three programs if no errors.

5.Create a jar file of your project.

Right click on Project in Package explorer and select Export menu

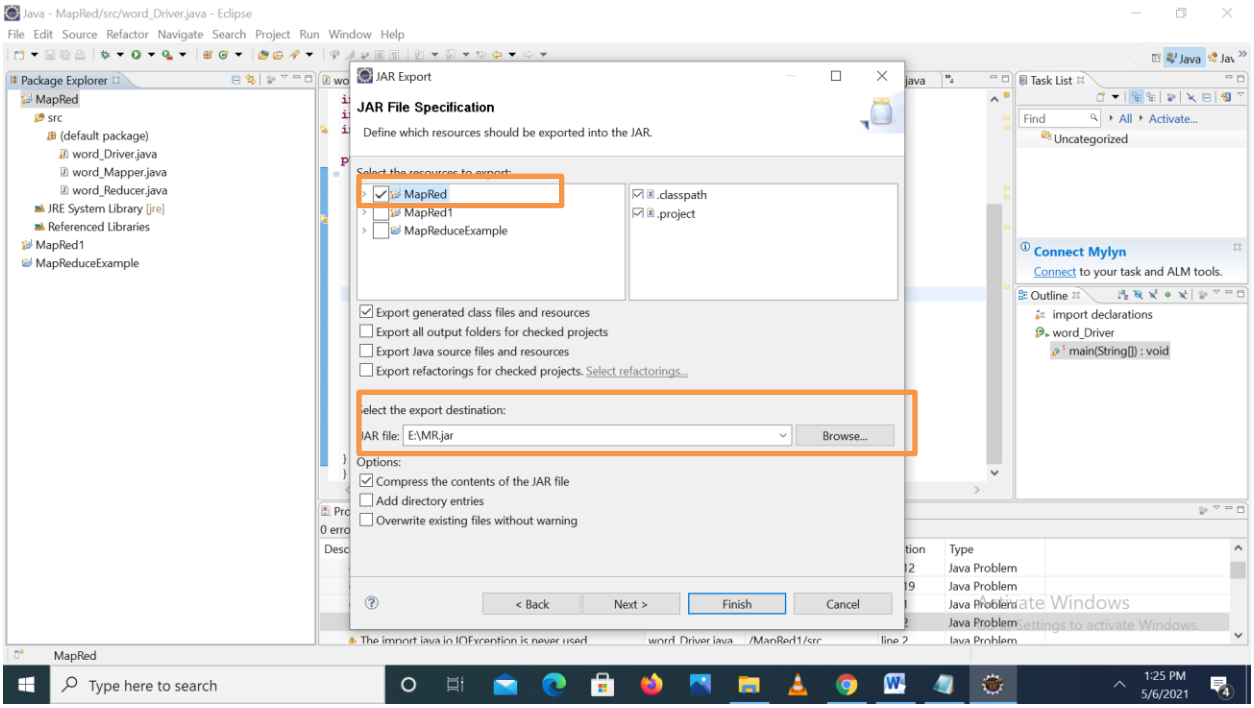


Select jar file option in Export Window:



Click on Next

Select your specific project from the list of projects and specify the path to store the jar file:



Click on finish.

After the jar file is successfully created in the mentioned path ,

6. Execute the Project from command prompt window.

Change the directory in command prompt window to /bin folder instead of sbin folder:
cd E:\Hadoop\hadoop-3.1.4\hadoop-3.1.4\bin

Syntax for executing the project :

hadoop jar Path_of_jar_file Name_of_Driver_class input_file_path_hdfs output_folder_name

E:\Hadoop\hadoop-3.1.4\hadoop-3.1.4\bin>hadoop jar E:\Project1\MR.jar word_Driver /Vasundhara/input.txt MR_Out

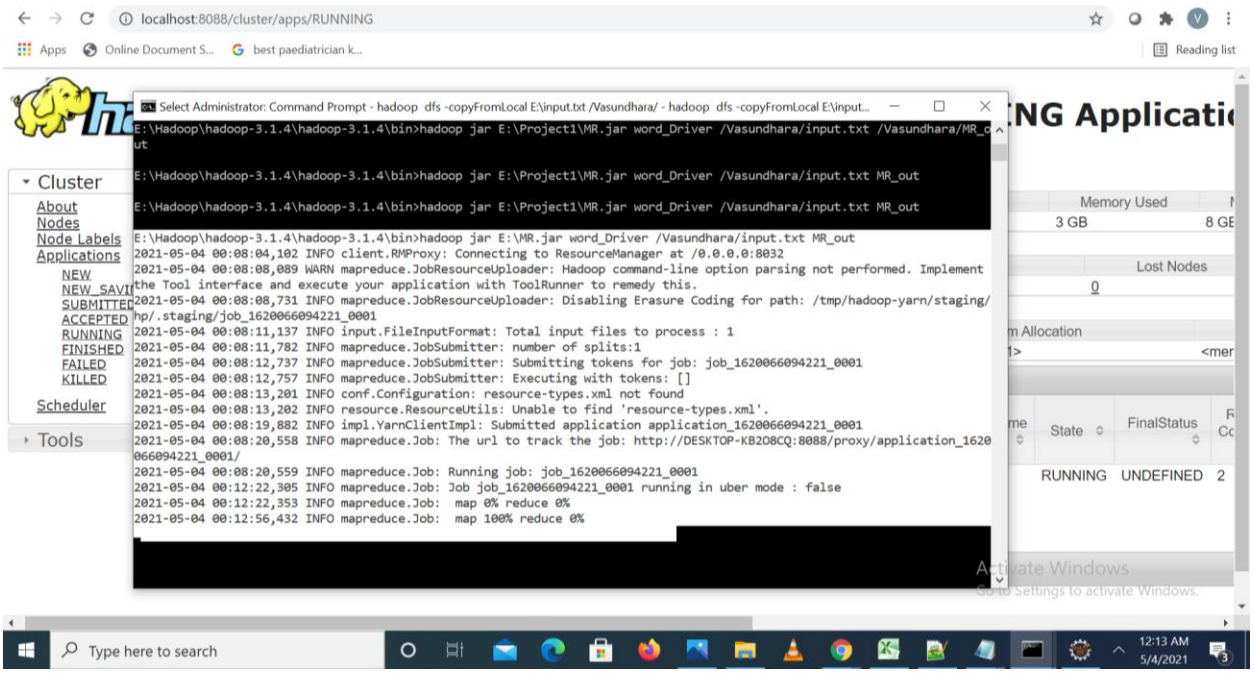
Where :

E:\Project1\MR.jar : is the exported project jar file path

word_Driver : is the class name of Driver class which has the main() method

/Vasundhara/input.txt : indicates the source file path available on HDFS

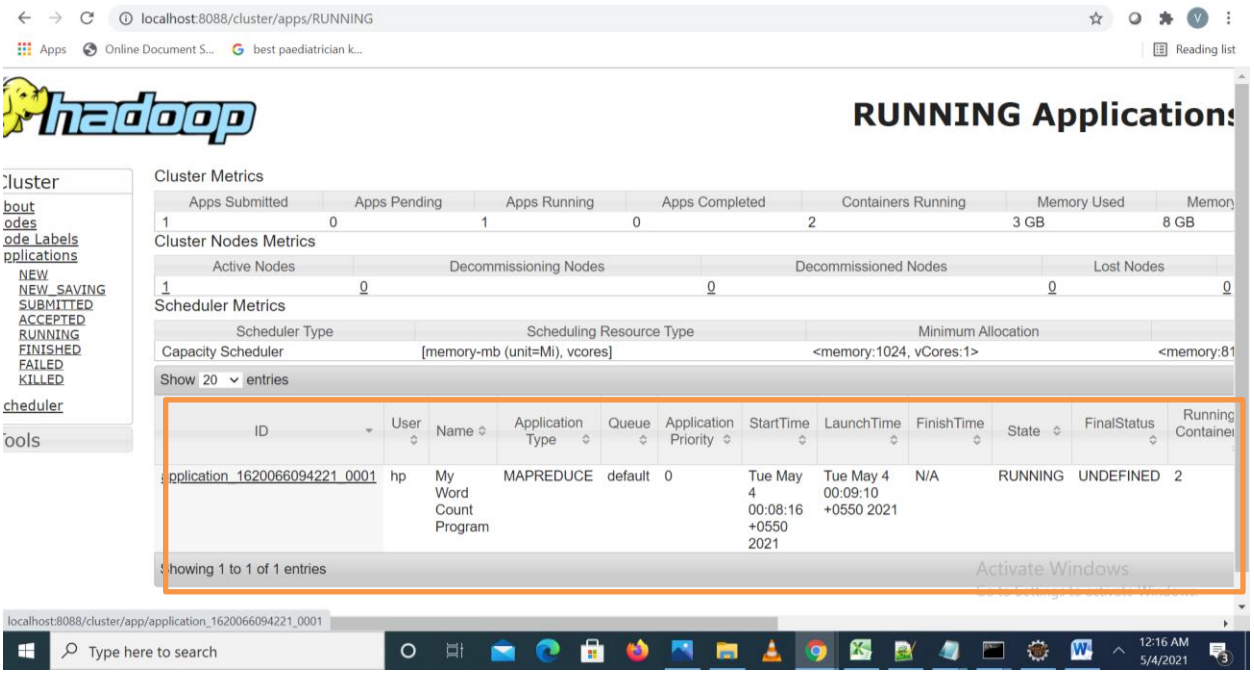
MR_Out : User-defined Folder name for storing Map-reduce Output



7. To access the status of the application we can use the Hadoop Web User Interface by providing following link in browser:

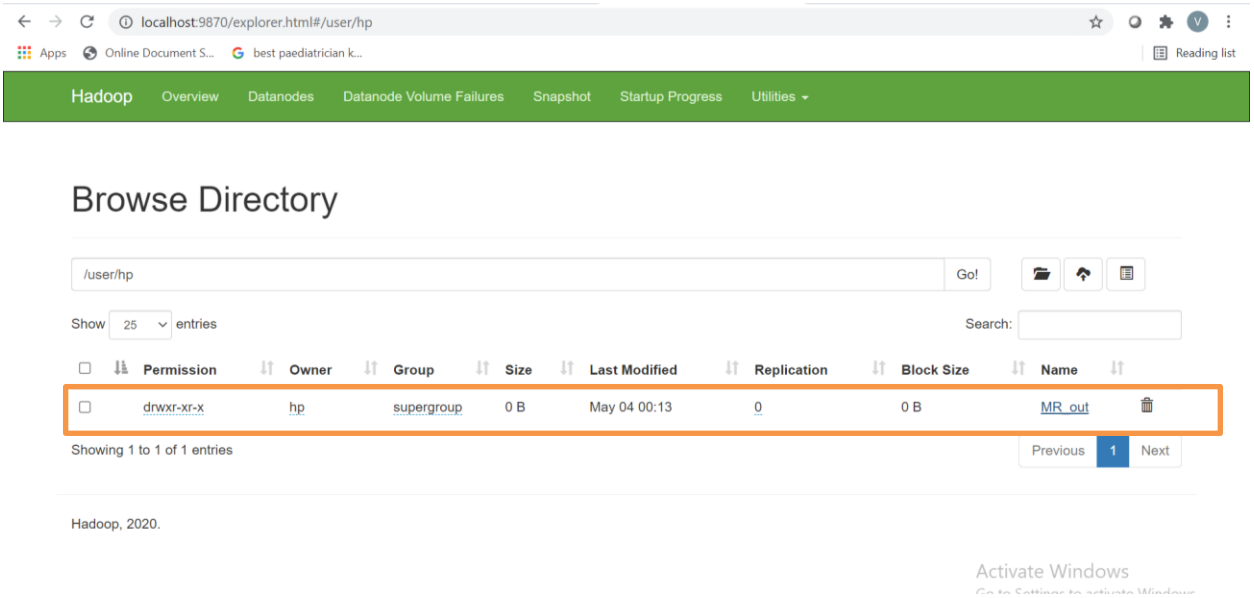
http://localhost:8088/cluster

It shows information related to active nodes and application status.



8. After the project finishes 100% execution of both map and reduce tasks we can access the output folder from Hadoop HDFS UI by providing following link in the browser:

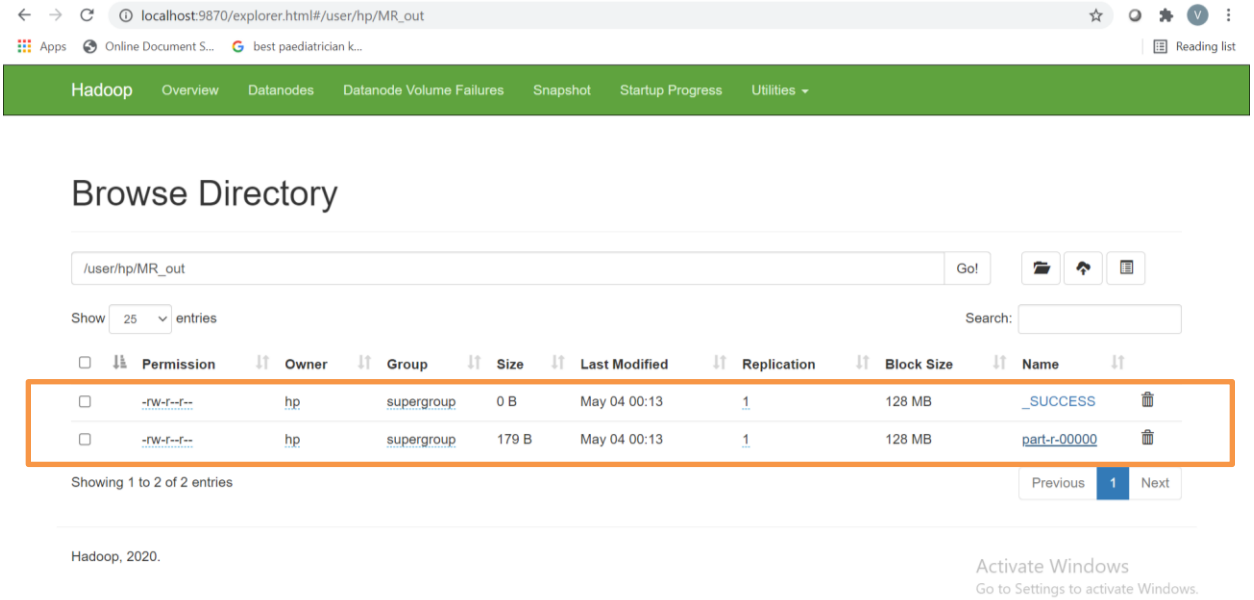
http://localhost:9870



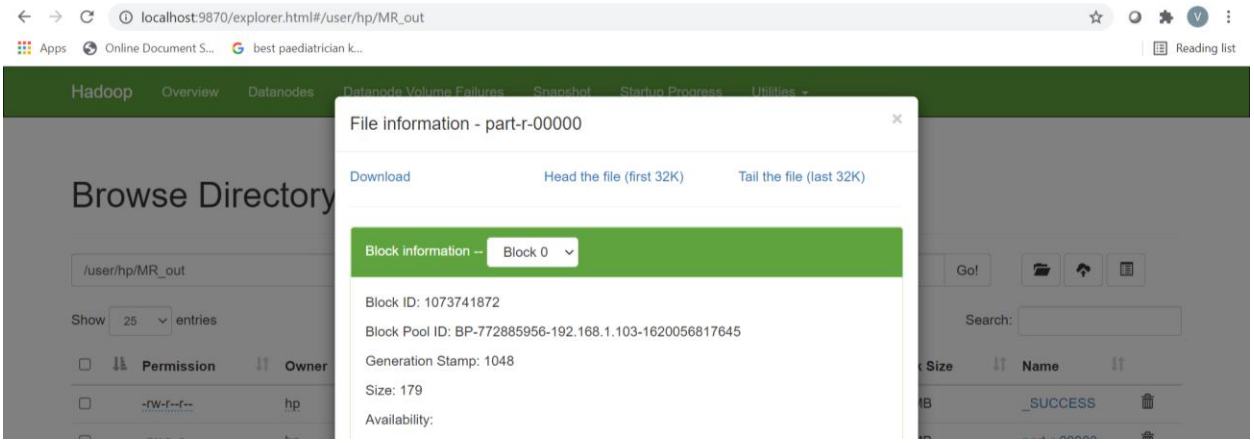
9. Click on the output folder: MR_out .It will display 2 files:

_SUCCESS : Shows the status and

part-r-00000 : has the actual output of Map-reduce



10. Click on part-r-00000 file : It will provide a window with option to download the file as follows :



11. Open the downloaded file to view the program output :

12. Another way of displaying the output is by reading the file from HDFS using following command syntax in command prompt :

hadoop fs –cat Path_of_part-r-00000_file_hdfs

E:\Hadoop\hadoop-3.1.4\hadoop-3.1.4\sbin>hadoop fs -cat /user/hp/MR_out/part-r-00000