# CIS 612 Big Data and Parallel Processing LAB-4\_1 :

# Hadoop Installation

**Maaz Muhammad Khan**

**CSU ID-2788572**

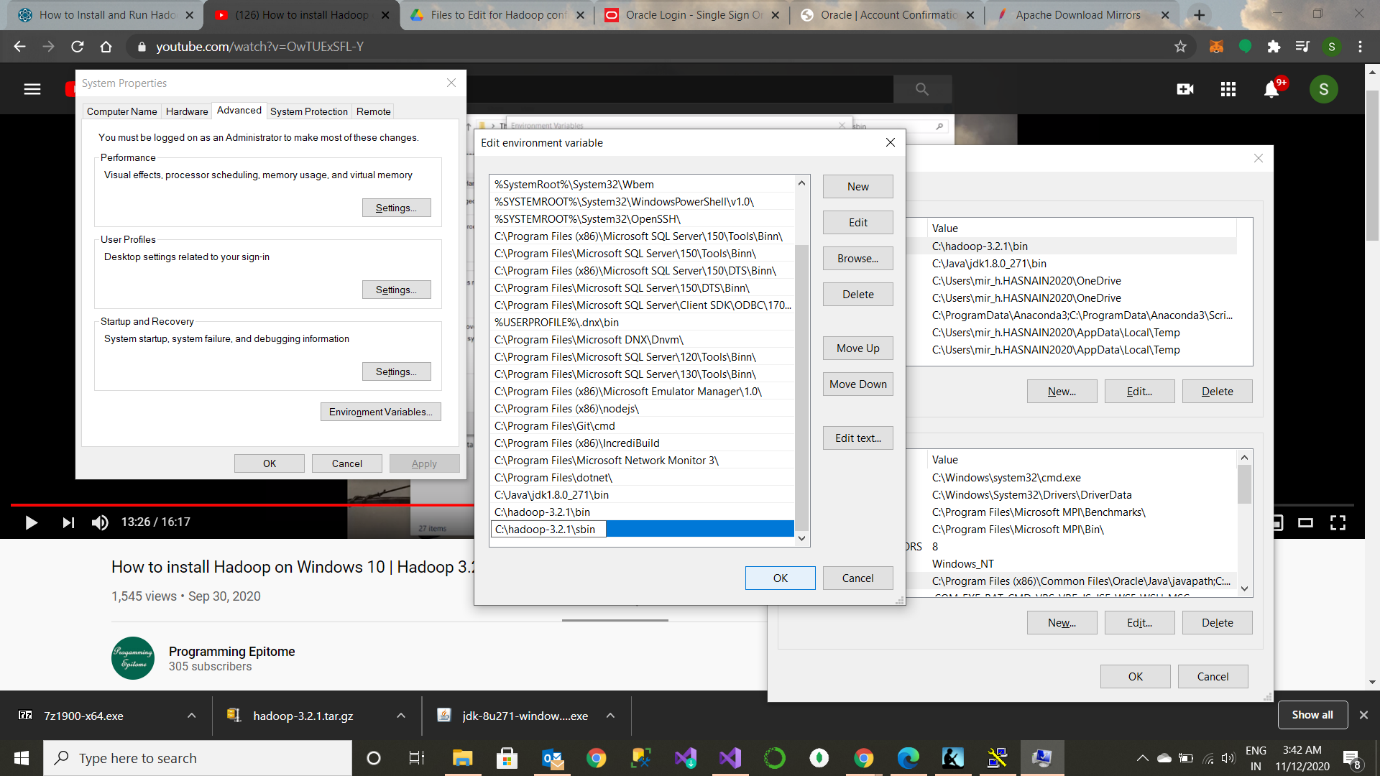
**Platform Setup:**

Hadoop 3.2.1 is installed in this lab, but in order to do that we must have JAVA SDK-8 installed on our system , so we install that first.

Hadoop would be setup as a single node system in pseudo-distributed mode

We download Hadoop from- <https://hadoop.apache.org/releases.html>

Once Hadoop is installed , we must set the path and environment variables for it as:



After that 5 configuration files in the C:\hadoop-3.2.1\etc\hadoop folder, viz

core-site.xml ,mapred-site.xml, yarn -site.xml , hdfs-site.xml and hadoop-env.cmd are edited.

a) File C:/Hadoop-3.2.1/etc/hadoop/core-site.xml, paste below xml paragraph and save this file.

<configuration>

<property>

<name>fs.defaultFS</name>

<value>hdfs://localhost:9000</value>

</property>

</configuration>

b) C:/Hadoop-3.2.1/etc/hadoop/mapred-site.xml, paste below xml paragraph and save this file.

<configuration>

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

</configuration>

c) Create folder "data" under "C:\Hadoop-3.2.1"

1) Create folder "datanode" under "C:\Hadoop-3.2.1\data"

2) Create folder "namenode" under "C:\Hadoop-3.2.1\data" data

d) Edit file C:\Hadoop-3.2.1/etc/hadoop/hdfs-site.xml, paste below xml paragraph and save this file.

<configuration>

<property>

<name>dfs.replication</name>

<value>**1**</value>

</property>

<property>

<name>dfs.namenode.name.dir</name>

<value>C:\hadoop-3.2.1\data\namenode</value>

</property>

<property>

<name>dfs.datanode.data.dir</name>

<value>C:\hadoop-3.2.1\data\datanode</value>

</property>

</configuration>

e) Edit file C:/Hadoop-3.2.1/etc/hadoop/yarn-site.xml, paste below xml paragraph and save this file.

<configuration>

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

<property>

<name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>

<value>org.apache.hadoop.mapred.ShuffleHandler</value>

</property>

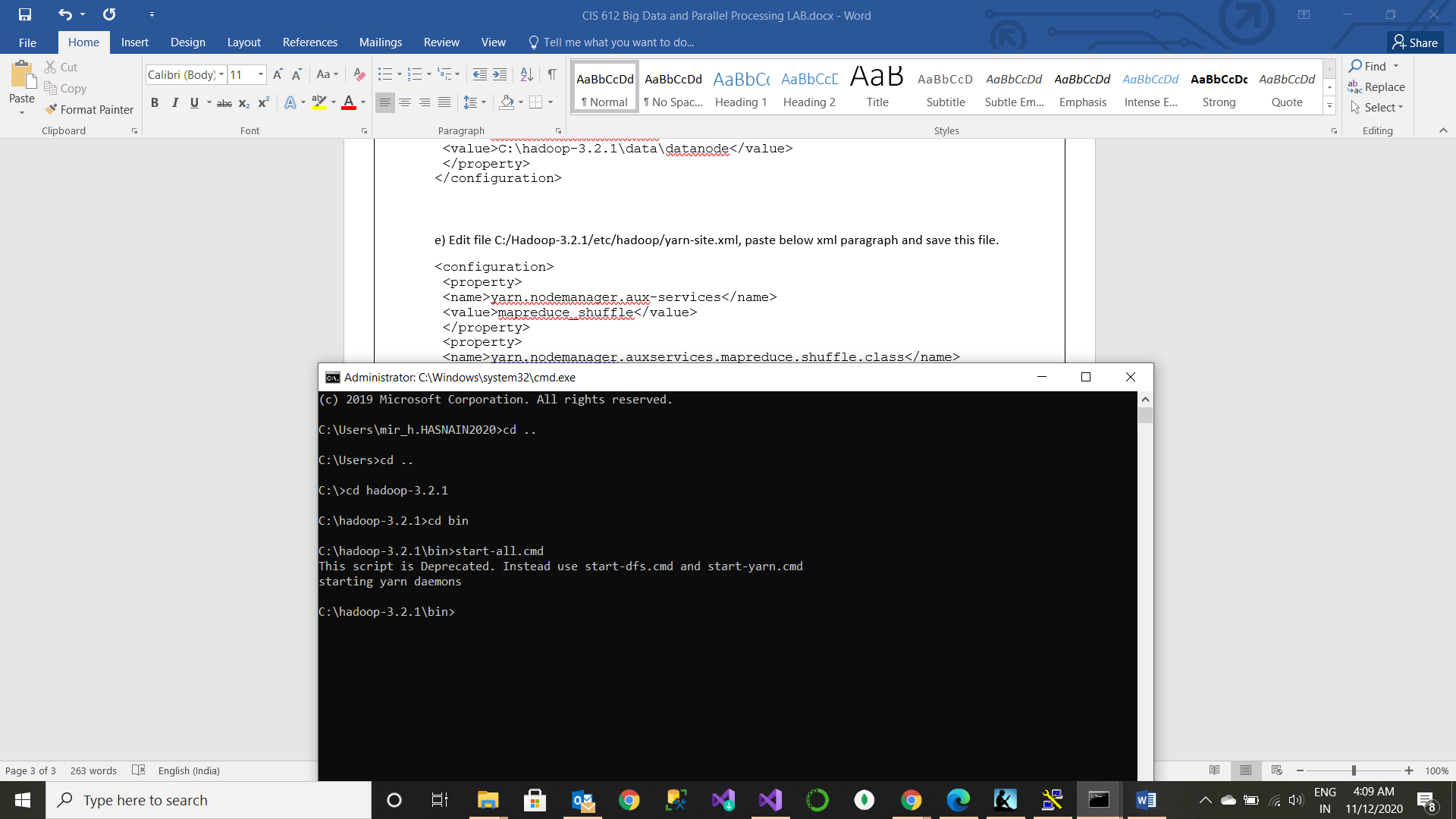
</configuration>

f) save the java path in the hadoop-env.cmd file as the path of the java sdk’s bin folder.

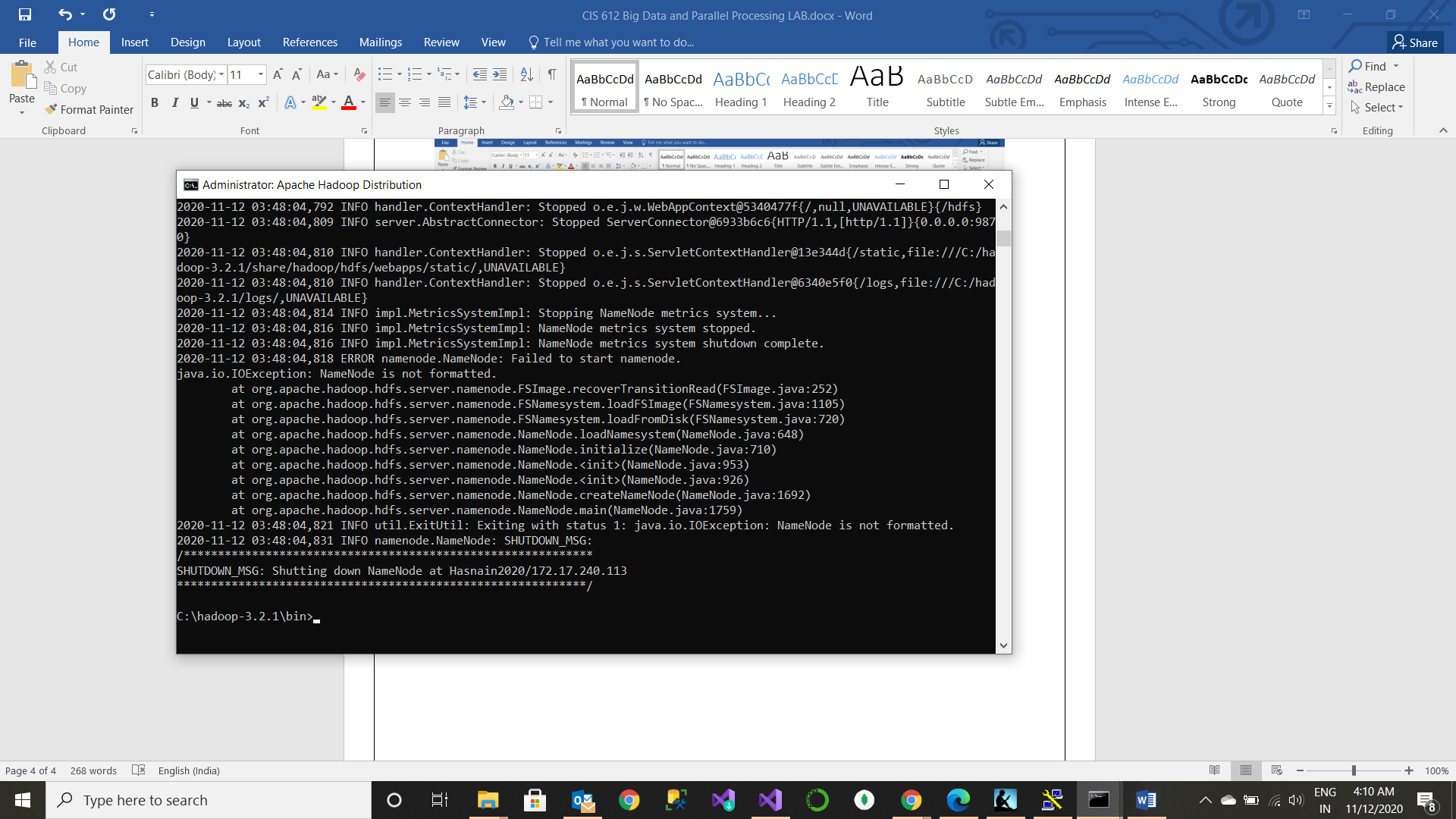
Once the configurations files have been edited and saved, the Hadoop configuration is completed successfully.

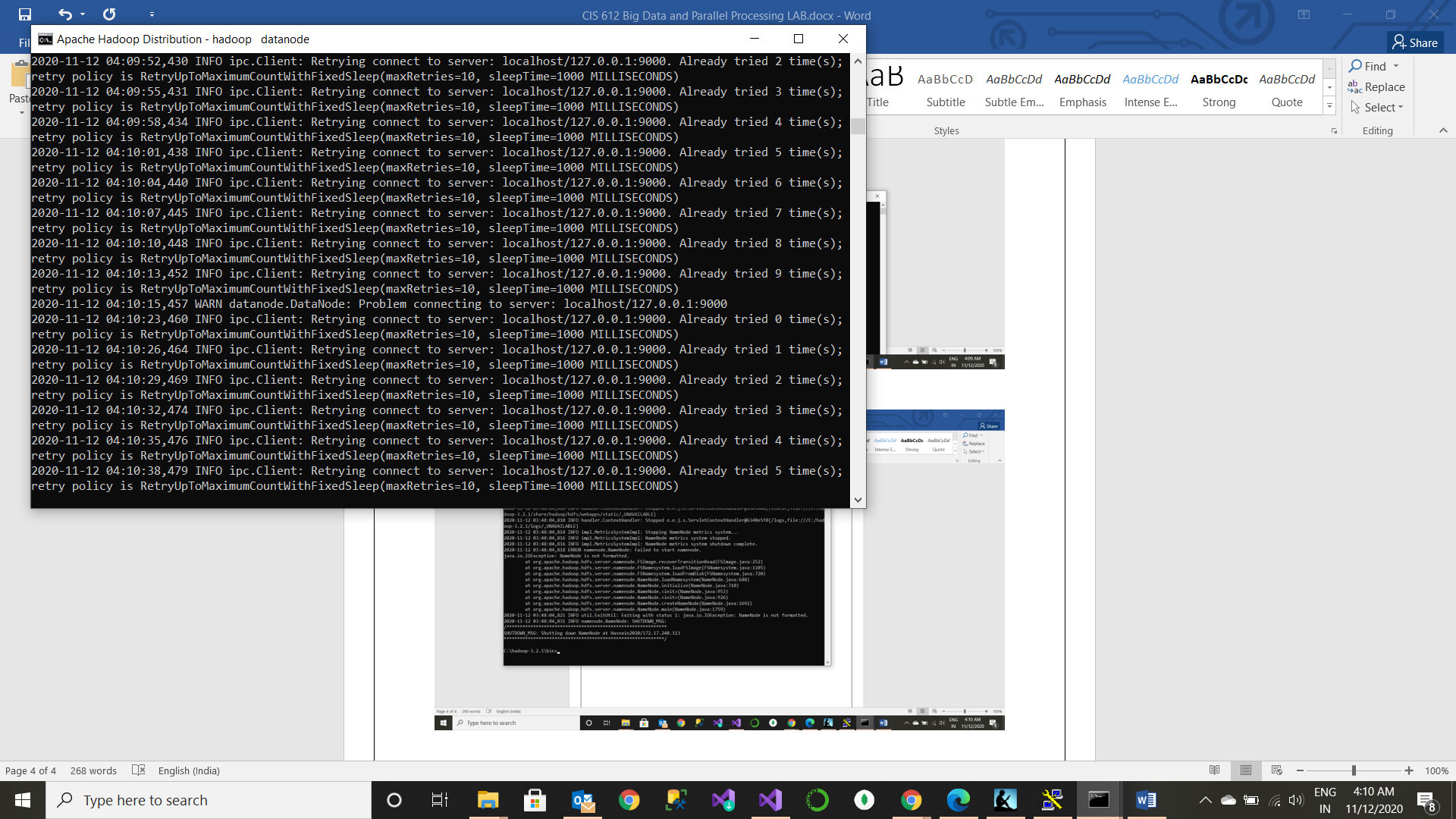
**Running the Hadoop single node system**

Go to the cmd prompt and go to the Hadoop-3.2.1/bin folder and type start-all.cmd to start all the nodes in the Hadoop system.

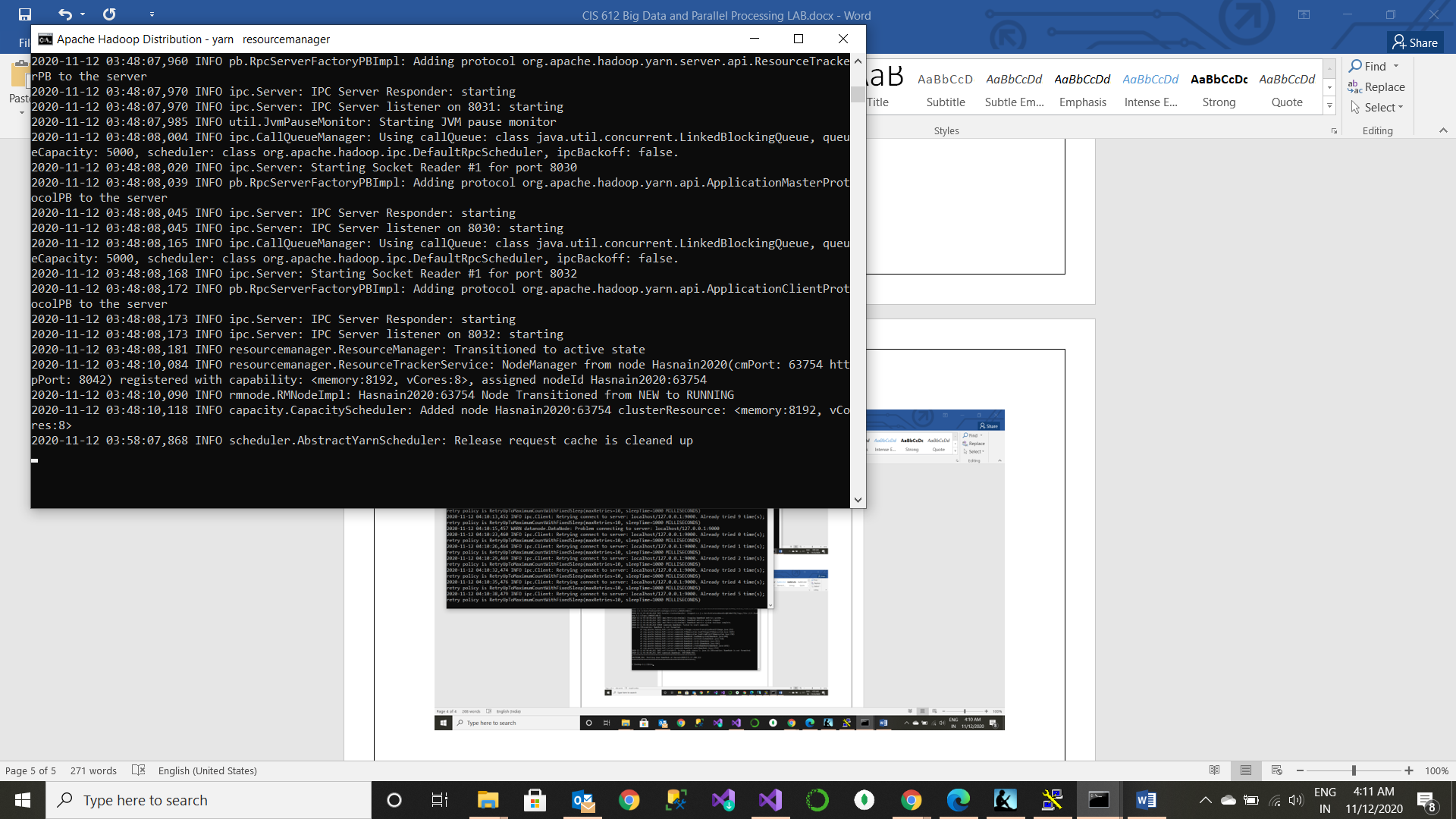


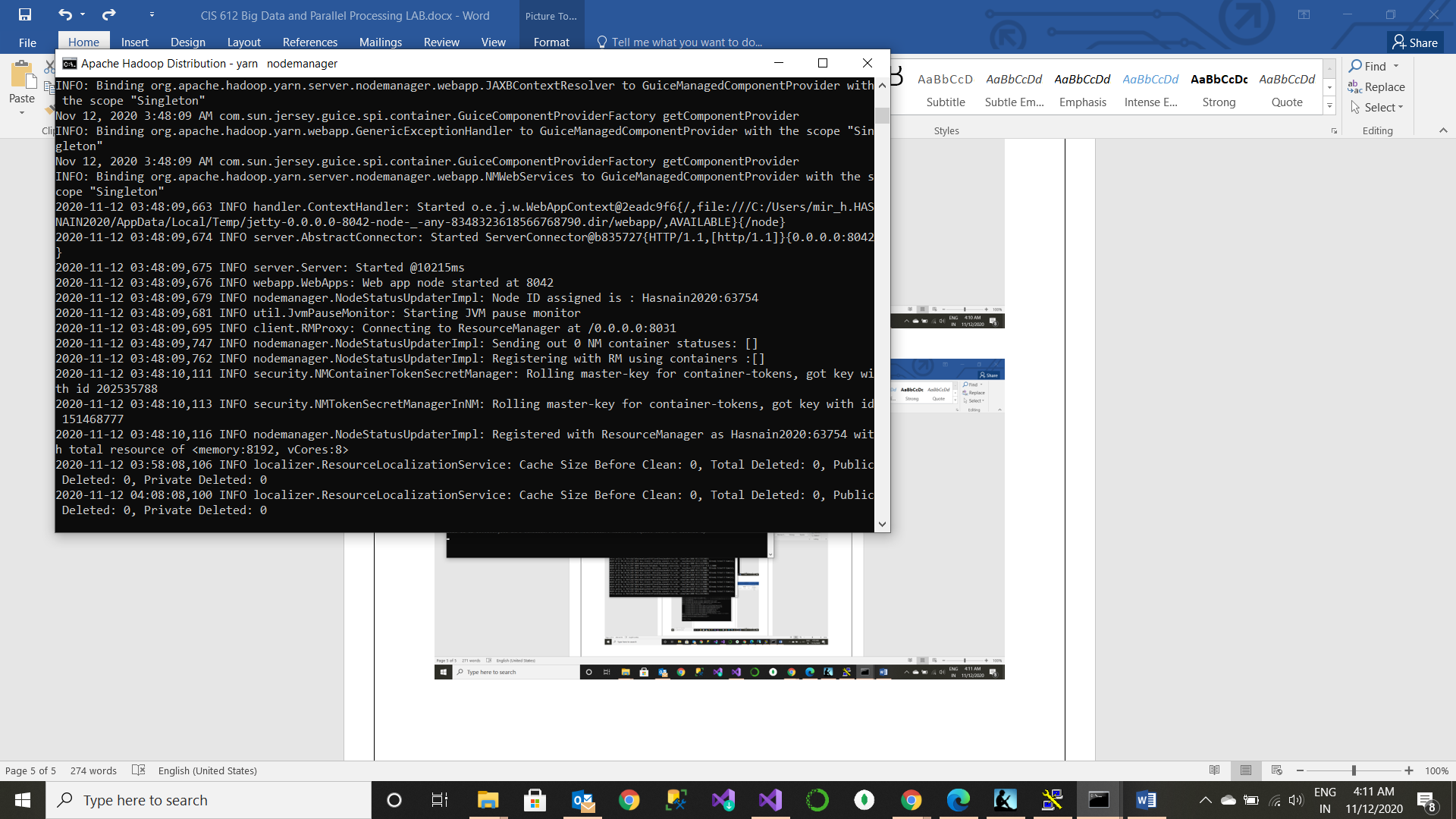
This would start the following

Name Node

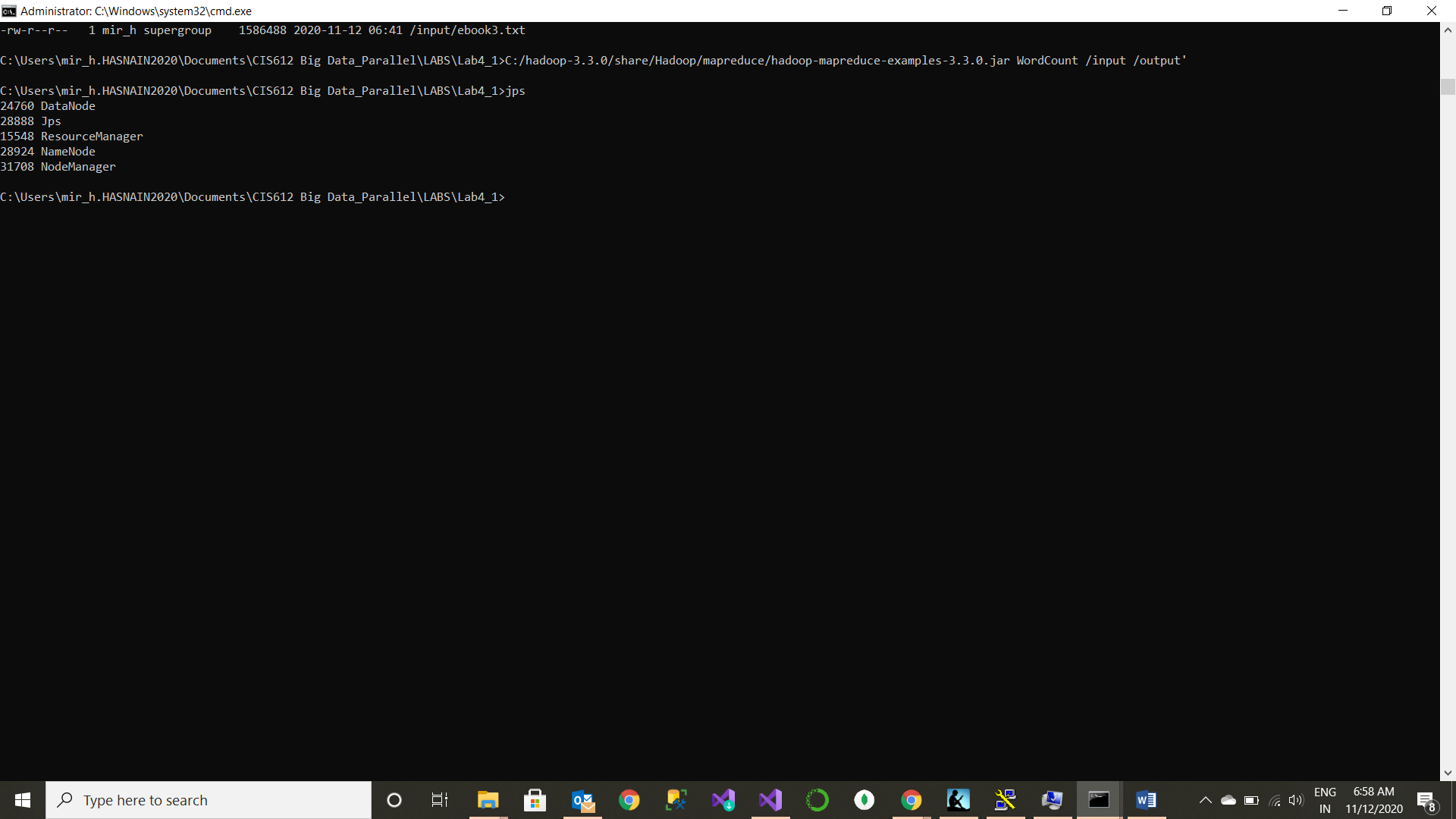
Data Node

Resource Manager

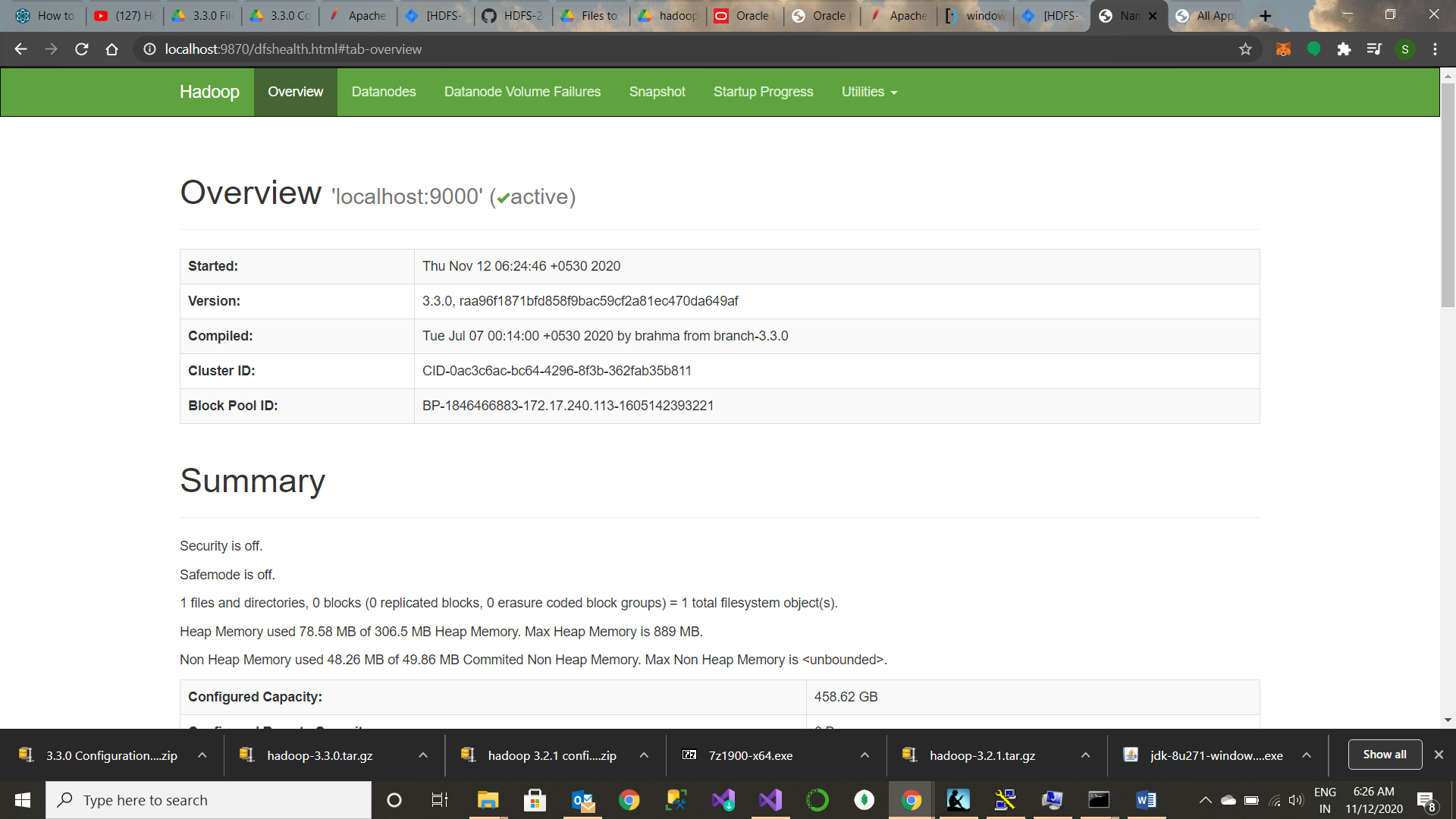


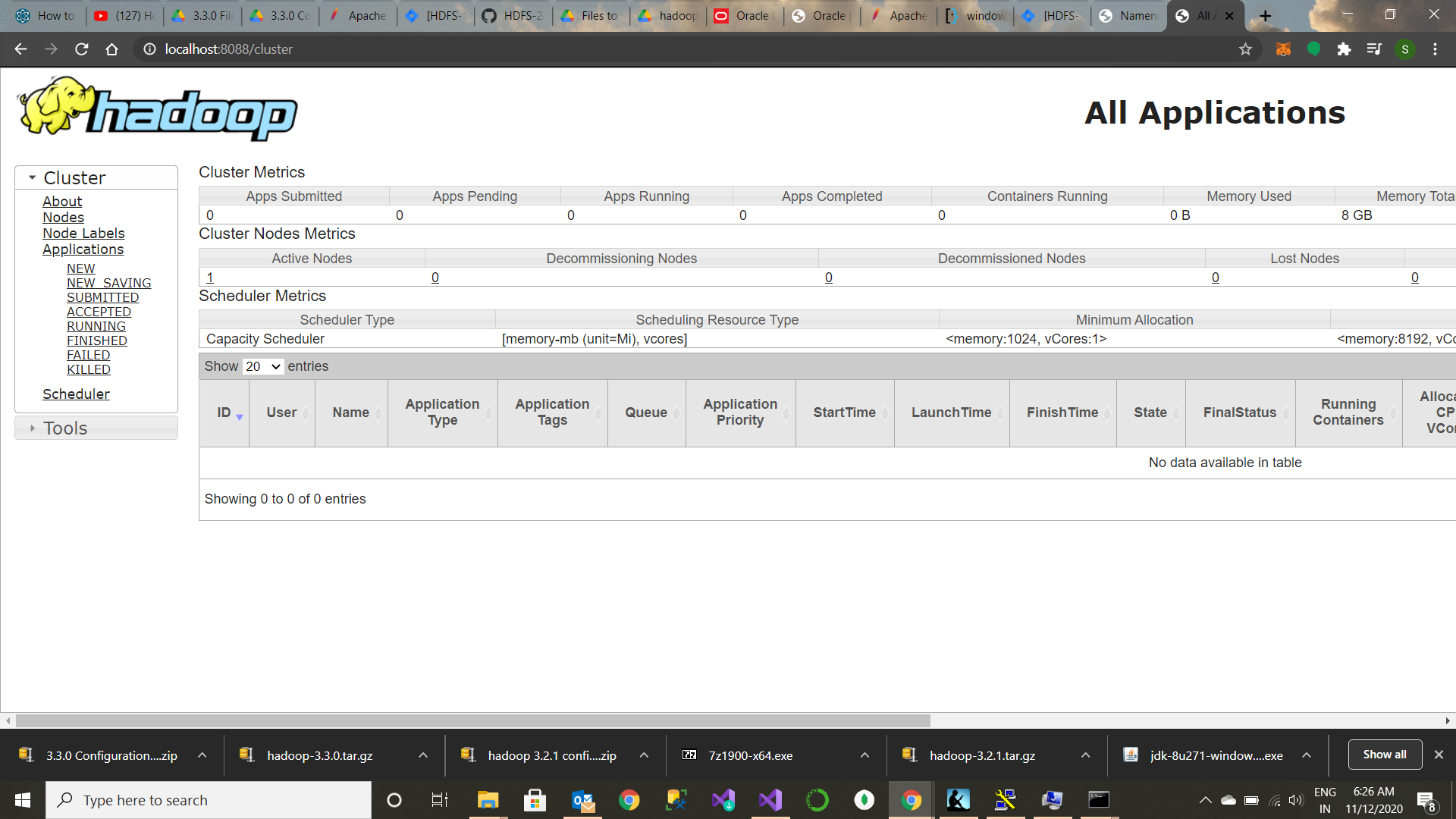
Node Manager

Checking the port of the daemons using **jps** command



Open the browser and type localhost:9870 anf localhost:8086/clusters to see the Hadoop connection and the nodes running on it .





**Putting the input files into the HDFS input folder**

Download ebooks from the internet and store them into your local file folder. Now navigate to that folder in your command prompt and type the following commands to put the input files into from the local folder to the HDFS input folder.

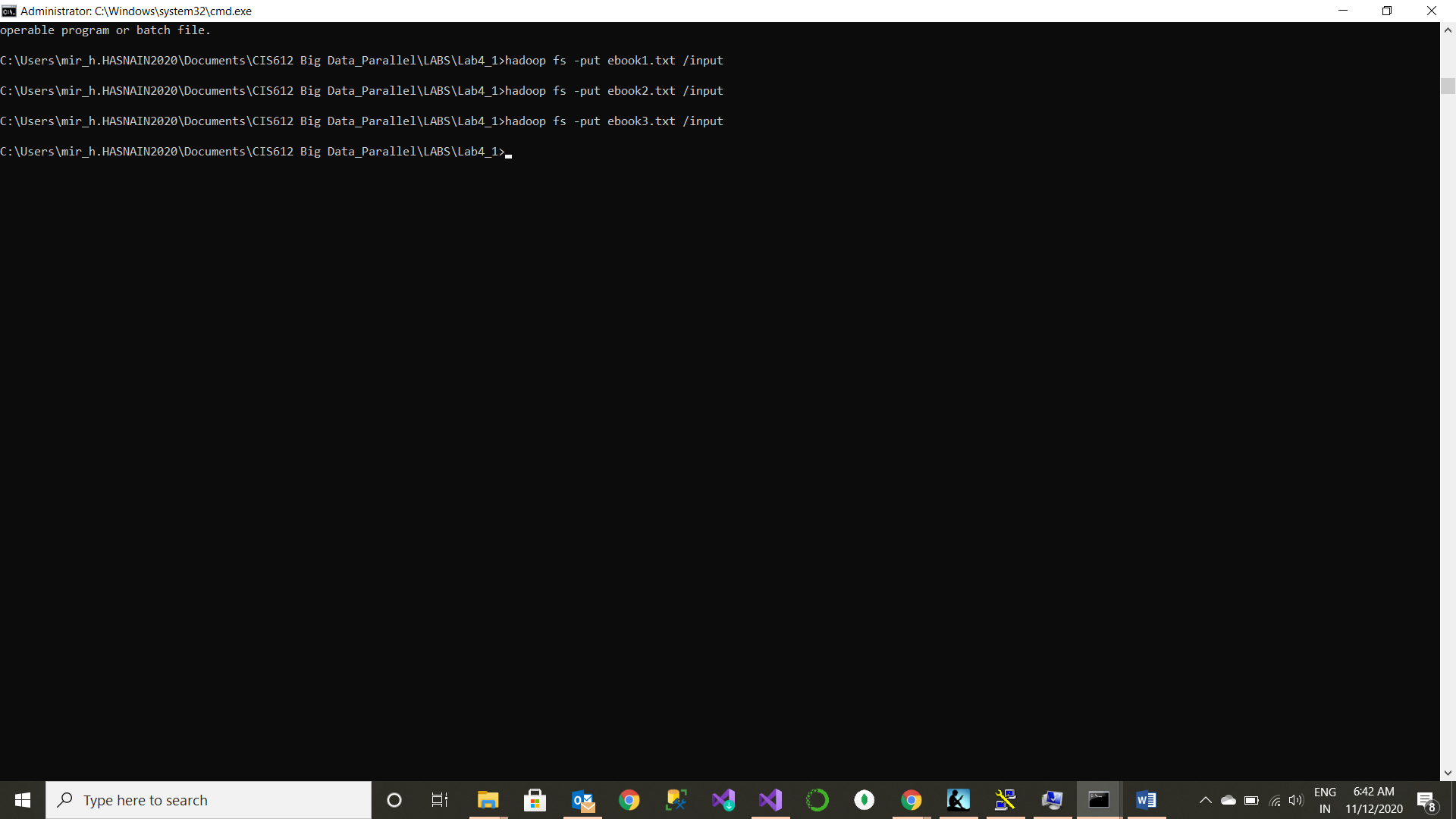
But before that we must make an input folder

**hadoop dfs -mkdir /input**

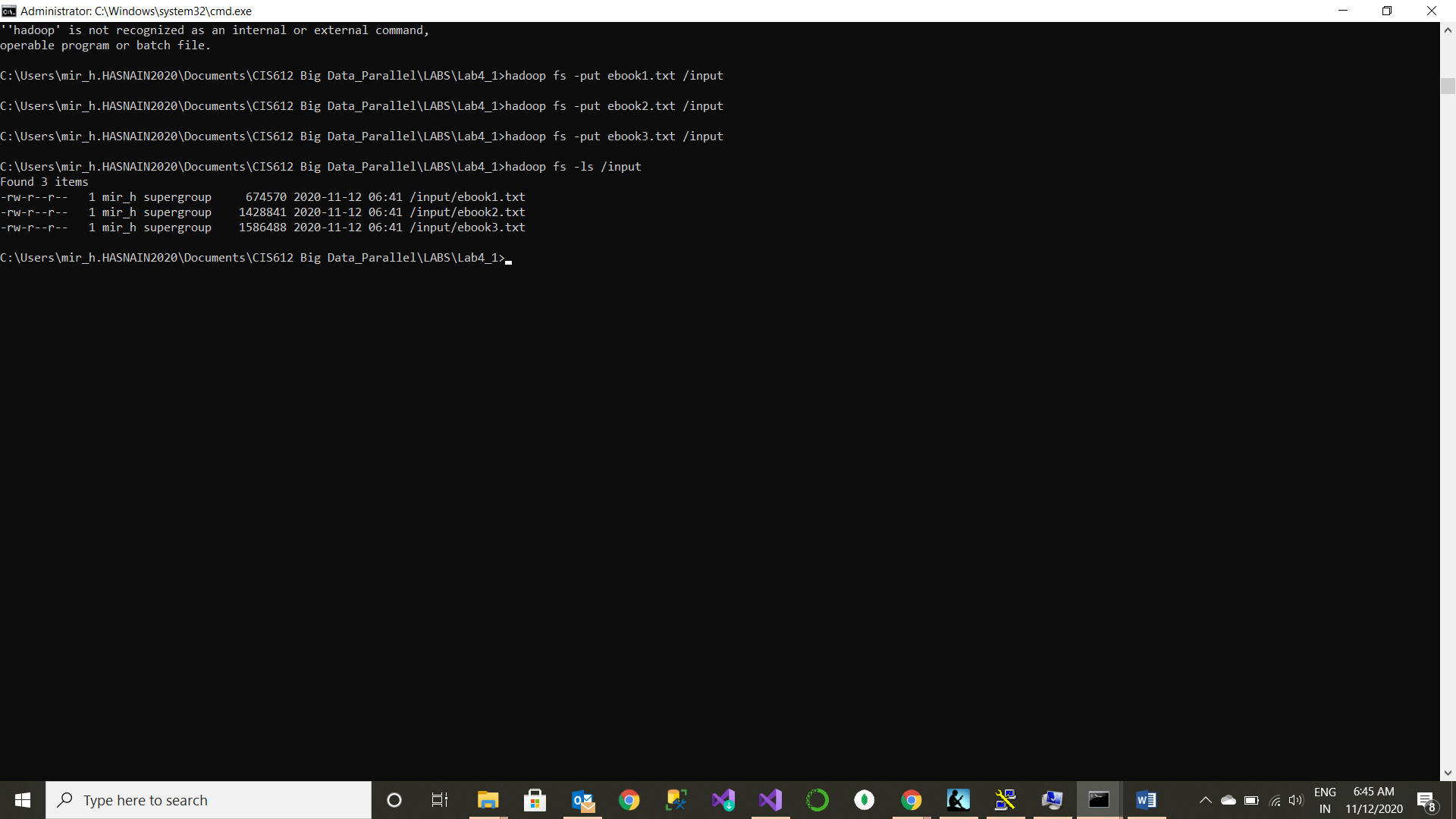
Using the command **hadoop fs -put ebook3.txt /inpu**t

hadoop fs -put ebook1.txt /input

hadoop fs -put ebook2.txt /input



We have successfully put the input files into the HDFS system and displaying it using **hadoop fs -ls /user/input**



C:/hadoop-3.3.0/share/Hadoop/mapreduce/hadoop-mapreduce-examples-3.3.0.jar WordCount /input /output'

**Running the WordCount.java file**

/hadoop jar hadoop-examples-\*.jar wordcount input output

**Source Code:**

package org.myorg;

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();

Job job = new Job(conf, "wordcount");

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);

}

}