# 



Big Data Analytics Assignment

By: Avnish Ramvinay Singh

Roll No: 33

Course: MSc. Computational Sciences and Applications Year: 2nd Year

Course Code: CSA 407 Submitted to: J. Sarkar Sir

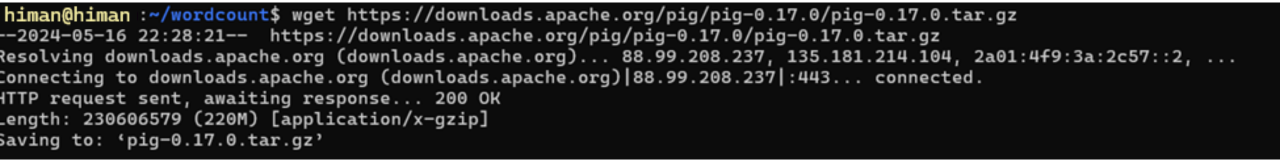
# Hadoop & Pig Installation Guide

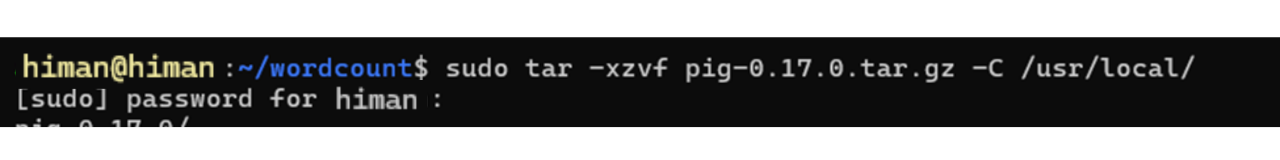
## Configuring the Environment and Running a WordCount MapReduce Program

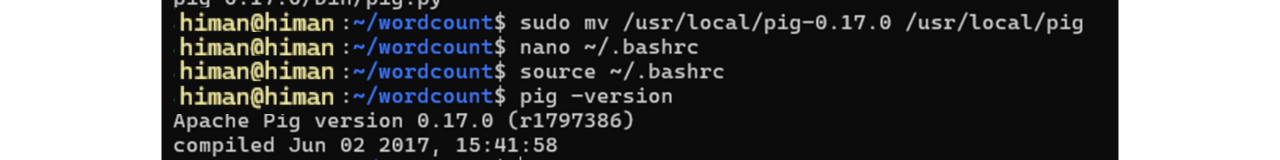
To proceed with configuring the environment for Hadoop and Pig, follow these step-by-step instructions, along with accompanying screenshots:

### Installing Hadoop and Pig:

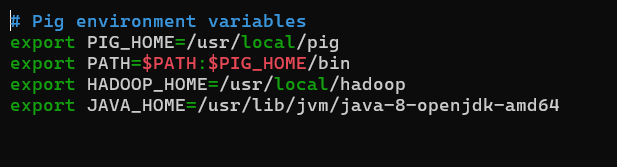
1. **Download Hadoop and Pig**: Begin by downloading the stable and compatible versions of Hadoop and Pig from their official websites.

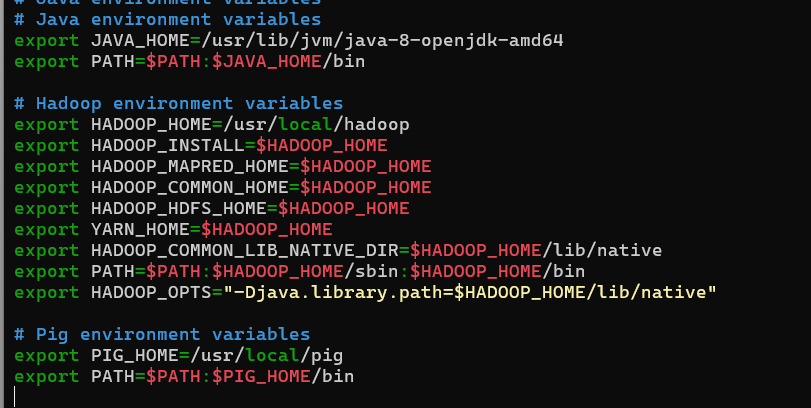
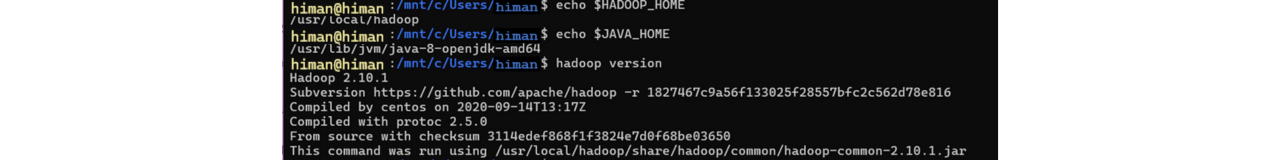






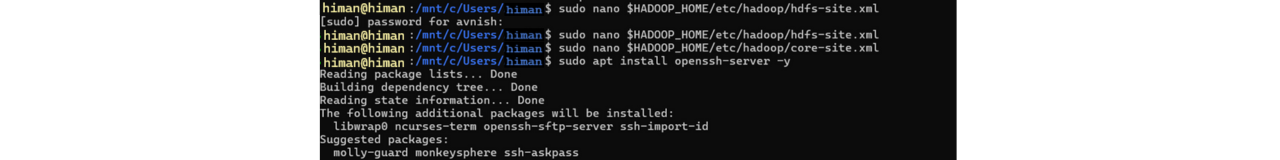
nano ~/.bashrc

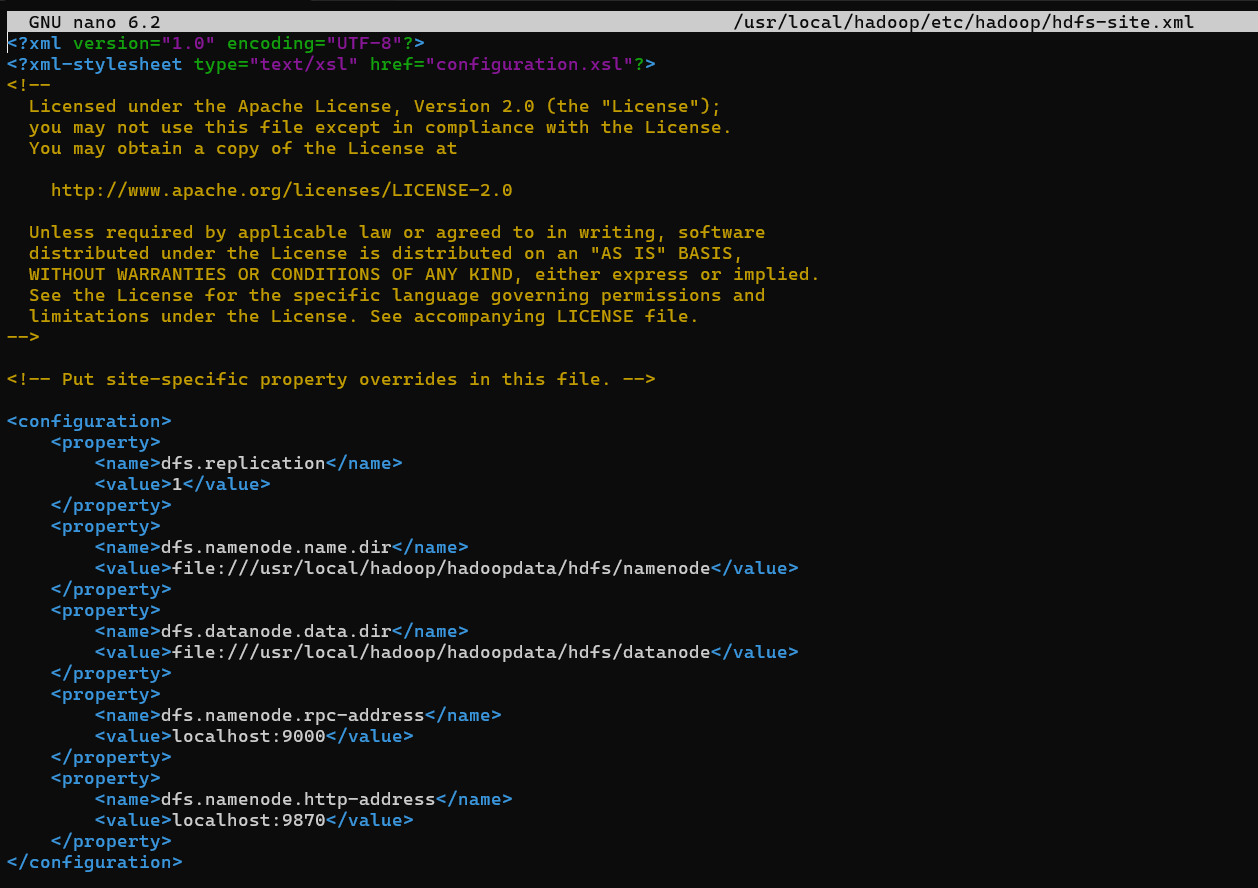


1. **Extract the Files**: Once the downloads are complete, extract the files to a preferred directory on your system.
2. **Set Environment Variables**: Configure the necessary environment variables by editing the .bashrc file. Add the following lines:
   * nano ~/.bashrc
3. 
4. **Verify Installation**: To confirm that the installations were successful, run the following commands in the terminal:
5. hadoop version  
   pig -version

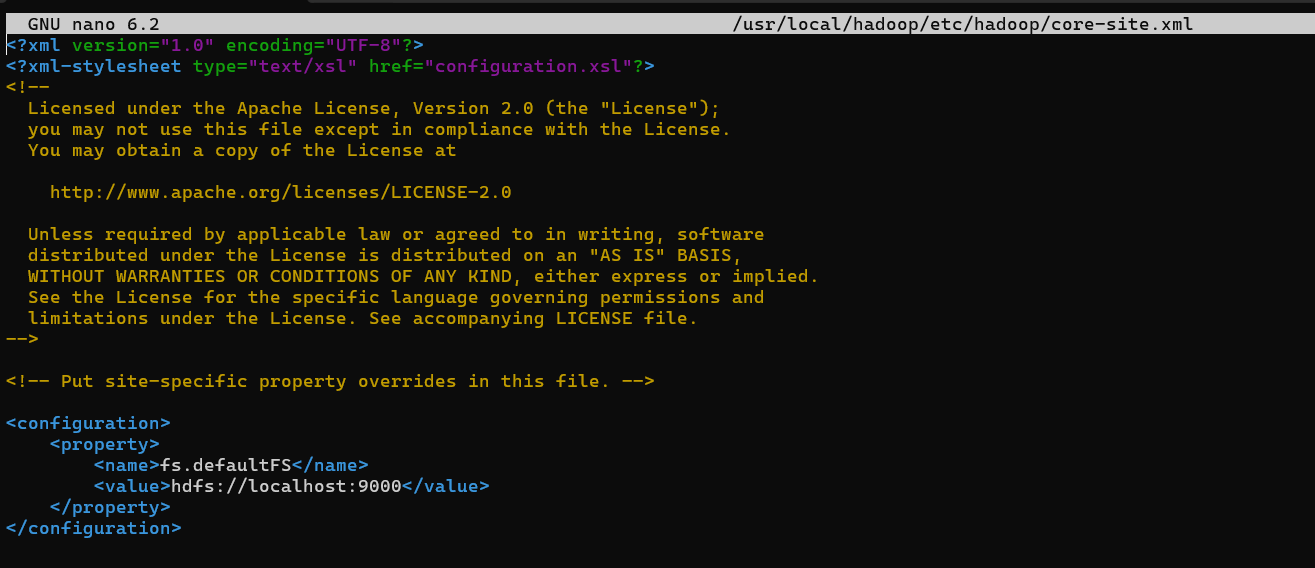
### Configuring Hadoop:

1. **Edit Configuration Files**: Navigate to the Hadoop conf directory and make the required configurations in core-site.xml, hdfs-site.xml, mapred-site.xml, and yarn-site.xml.

sudo nano $HADOOP\_HOME/etc/hadoop/hdfs-site.xml



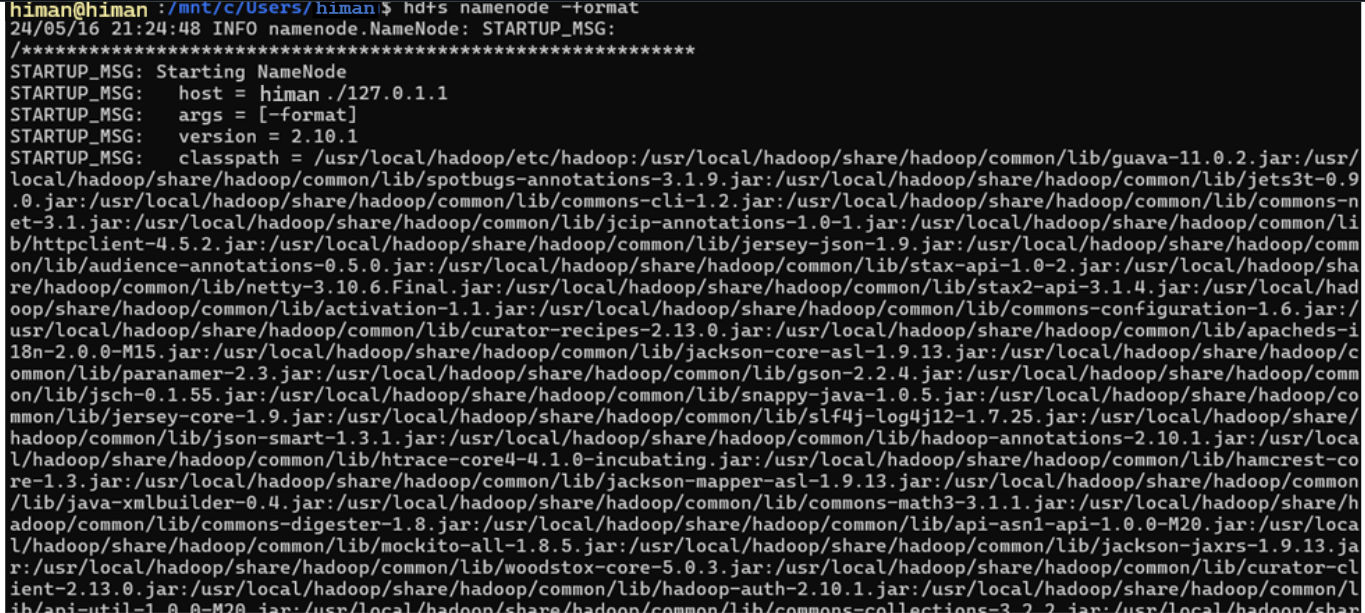
sudo nano $HADOOP\_HOME/etc/hadoop/core-site.xml



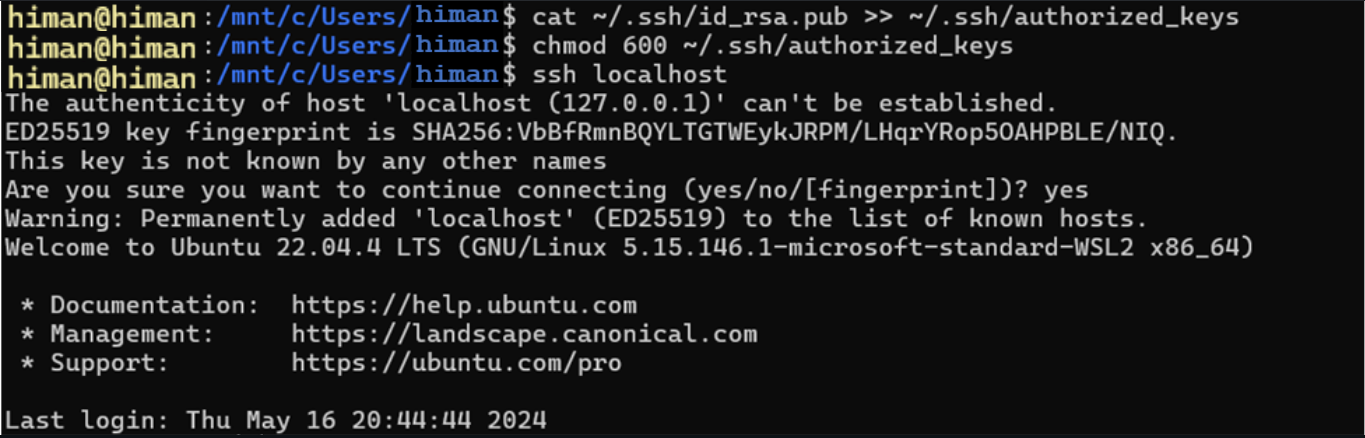
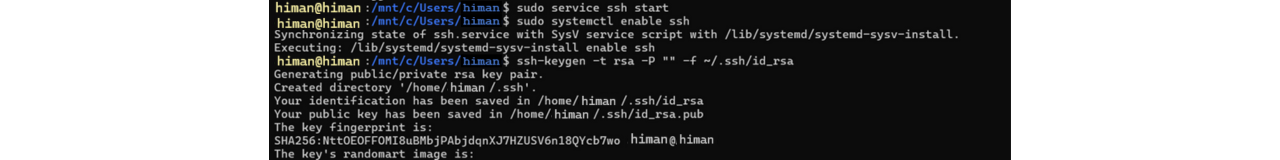


1. **Format the Hadoop Filesystem**: Run the command hdfs namenode -format to format the Hadoop filesystem.

* hdfs namenode -format

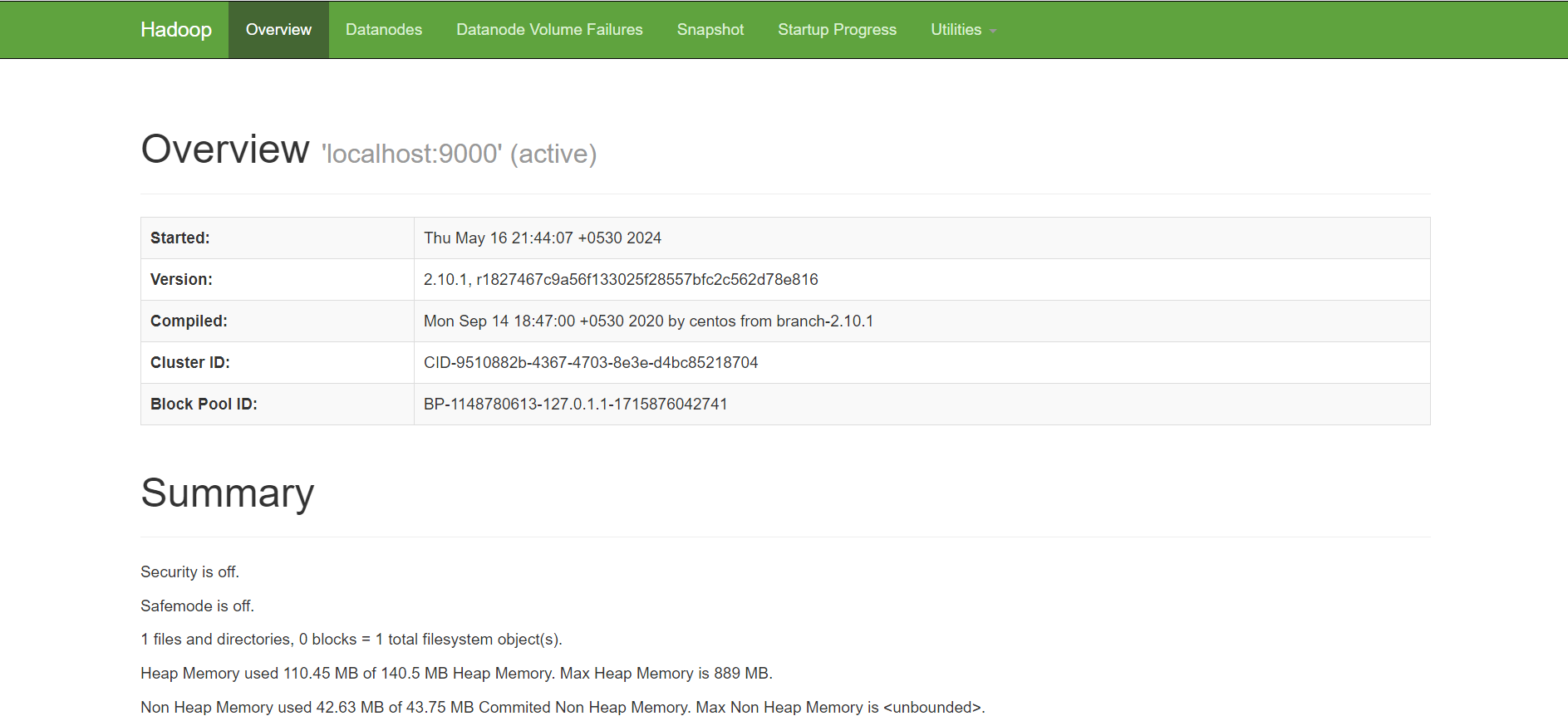


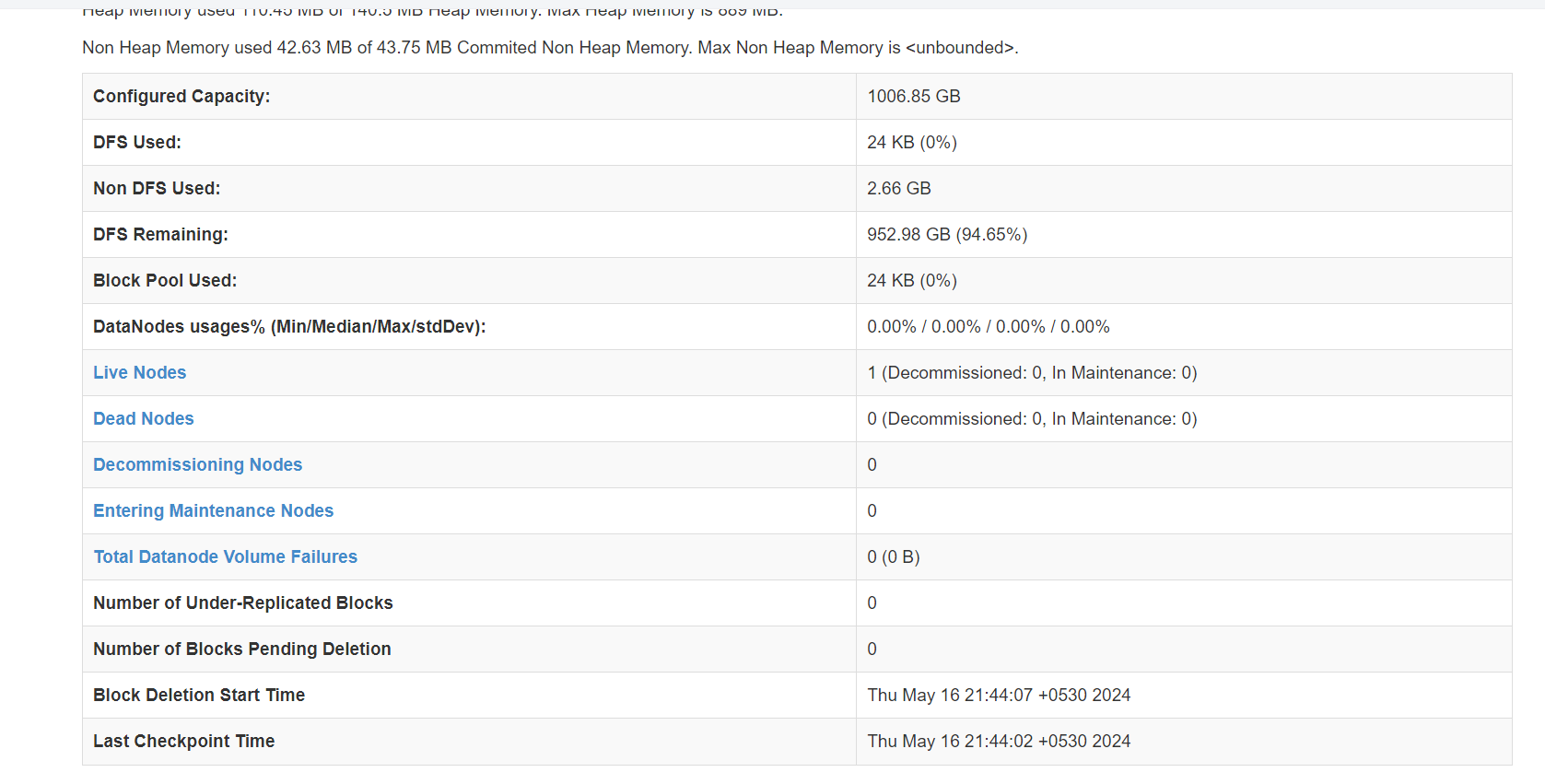
* sudo service ssh start
* sudo systemctl enable ssh
* ssh-keygen -t rsa -P "" -f ~/.ssh/id\_rsa
* cat ~/.ssh/id\_rsa.pub >> ~/.ssh/authorized\_keys
* chmod 600 ~/.ssh/authorized\_keys

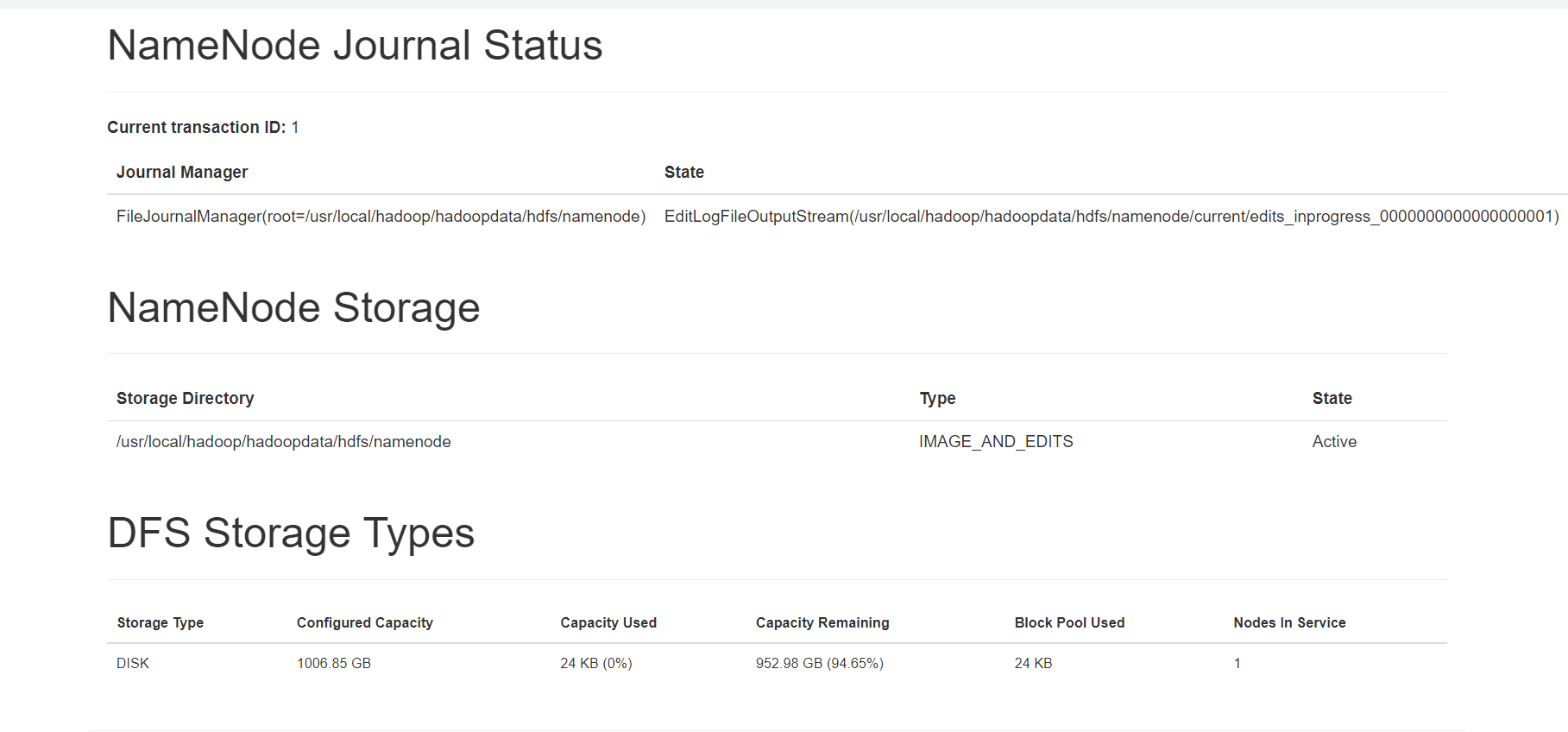


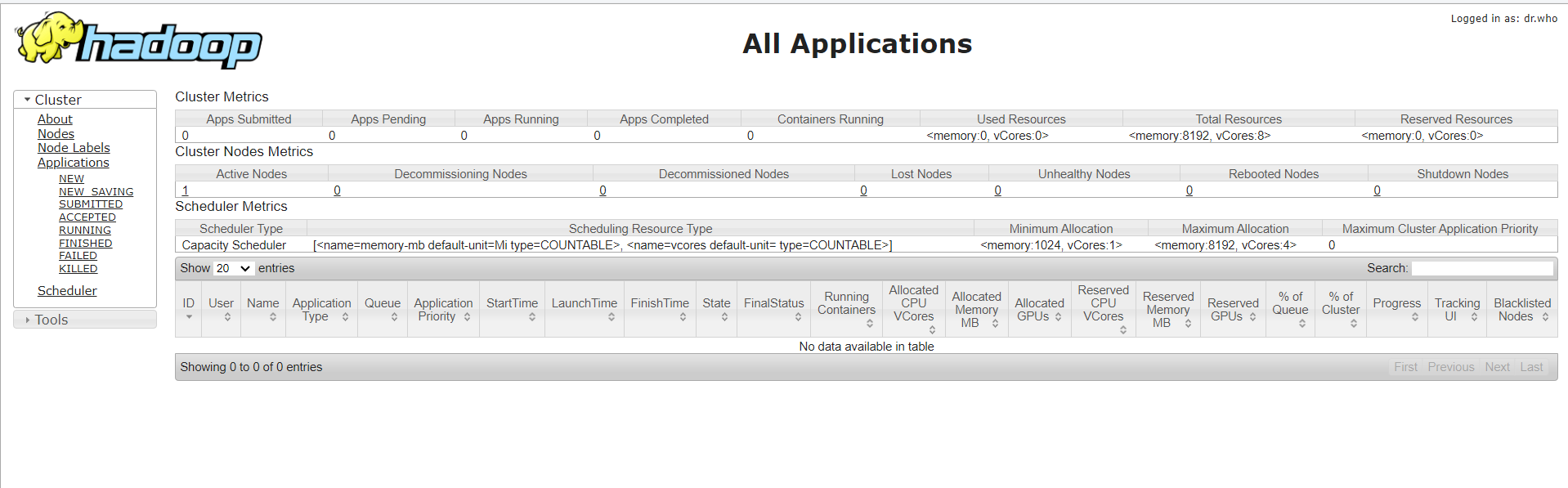
1. **Start Hadoop Services**: Initiate the Hadoop services by running the following commands:
2. start-dfs.sh  
   start-yarn.sh

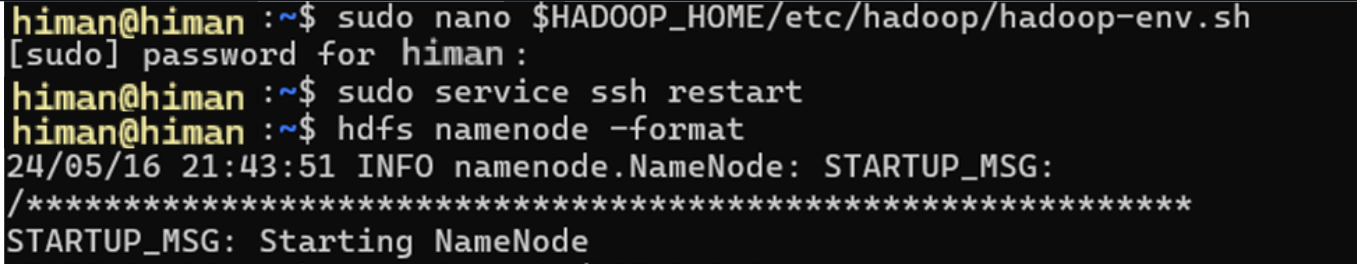
* NameNode: http://localhost:9870
* ResourceManager: http://localhost:8088





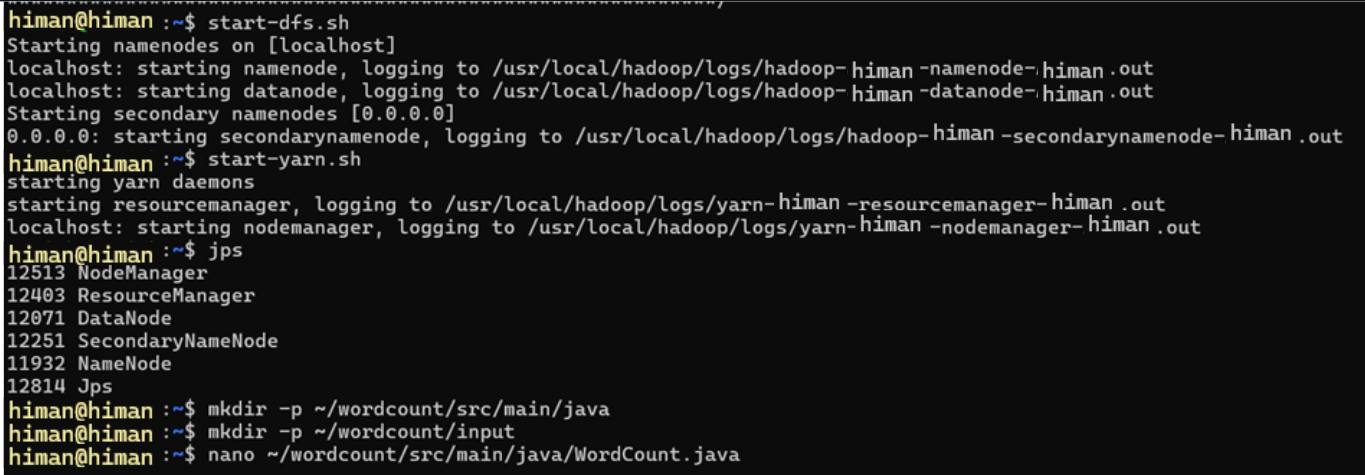


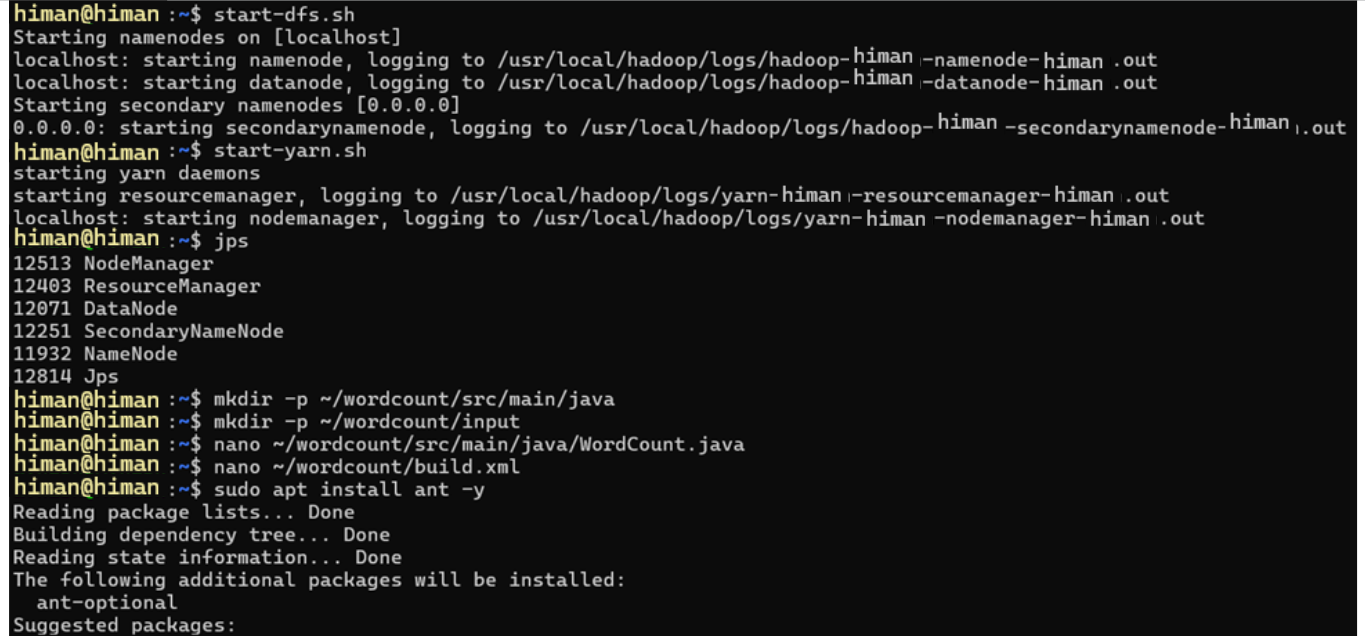




### Running a WordCount MapReduce Program:

1. **Create Input Directory**: Upload a text file to the Hadoop filesystem and create an input directory.





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;

public class WordCount {

public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable> {

private final static IntWritable one = new IntWritable(1);

private Text word = new Text();

public void map(Object key, Text value, Context context) throws IOException, InterruptedException {

StringTokenizer itr = new StringTokenizer(value.toString());

while (itr.hasMoreTokens()) {

word.set(itr.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);

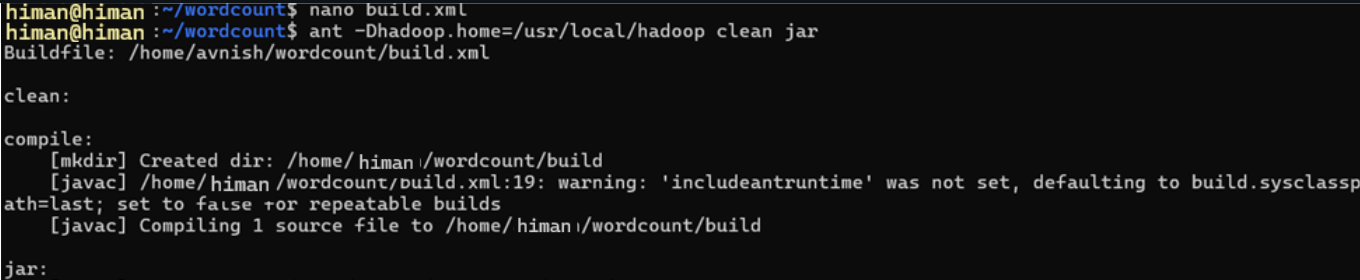
FileInputFormat.addInputPath(job, new Path(args[0]));

FileOutputFormat.setOutputPath(job, new Path(args[1]));

System.exit(job.waitForCompletion(true) ? 0 : 1);

}

}



* nano ~/wordcount/build.xml

<project name="WordCount" default="jar">

<property name="src" location="src/main/java"/>

<property name="build" location="build"/>

<property name="lib" location="$HADOOP\_HOME/share/hadoop/common/lib"/>

<property name="common" location="$HADOOP\_HOME/share/hadoop/common"/>

<property name="hdfs" location="$HADOOP\_HOME/share/hadoop/hdfs"/>

<property name="mapreduce" location="$HADOOP\_HOME/share/hadoop/mapreduce"/>

<property name="yarn" location="$HADOOP\_HOME/share/hadoop/yarn"/>

<target name="clean">

<delete dir="${build}"/>

</target>

<target name="compile">

<mkdir dir="${build}"/>

<javac srcdir="${src}" destdir="${build}">

<classpath>

<fileset dir="${lib}">

<include name="\*\*/\*.jar"/>

</fileset>

<fileset dir="${common}">

<include name="\*\*/\*.jar"/>

</fileset>

<fileset dir="${hdfs}">

<include name="\*\*/\*.jar"/>

</fileset>

<fileset dir="${mapreduce}">

<include name="\*\*/\*.jar"/>

</fileset>

<fileset dir="${yarn}">

<include name="\*\*/\*.jar"/>

</fileset>

</classpath>

</javac>

</target>

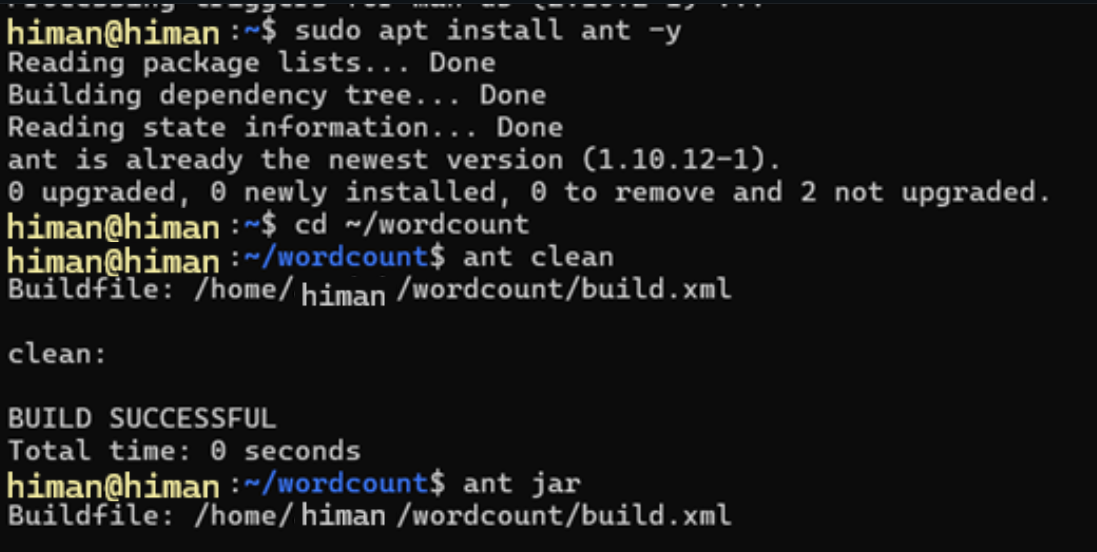
<target name="jar" depends="compile">

<mkdir dir="${build}/lib"/>

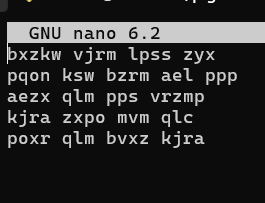
<jar destfile="${build}/WordCount.jar" basedir="${build}"/>

</target>

</project>

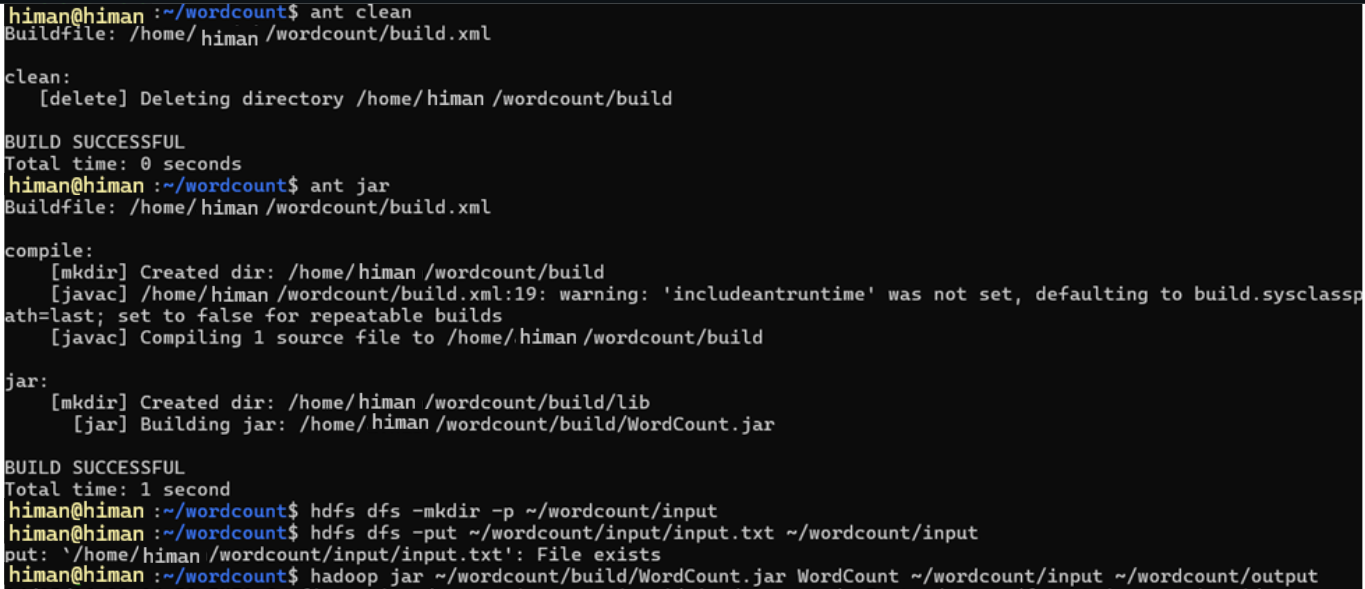


* nano ~/wordcount/input/input.txt

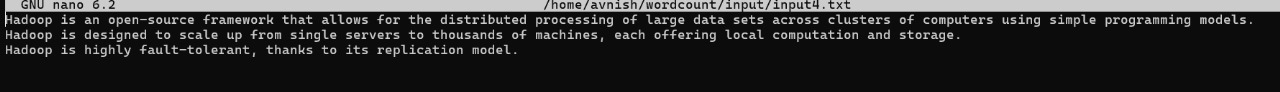


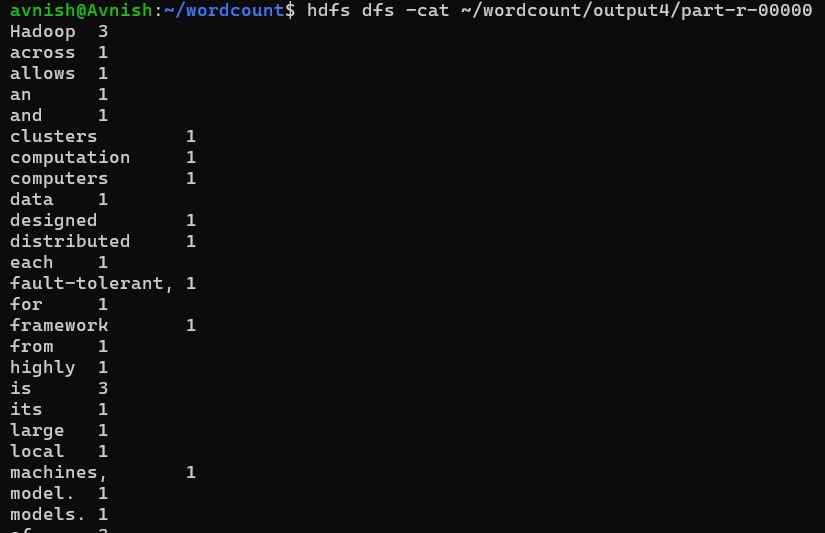
* **Upload the input file to HDFS**:
  + - hdfs dfs -mkdir -p ~ /wordcount/input
    - hdfs dfs -put ~/wordcount/input/input.txt ~/wordcount/input

1. **Run the WordCount Program**: Execute the WordCount MapReduce program by running the following command:
2. hadoop jar /path/to/hadoop-mapreduce-examples.jar wordcount input\_directory output\_directory



1. **View Output**: After the program finishes running, check the output directory to view the results of the WordCount program.





.

## Running a WordCount MapReduce Program with with pig

### Creating Input Directory and Files:

Step 1: Prepare the Input Data

Create a directory for the input file:

mkdir -p ~/pig/input

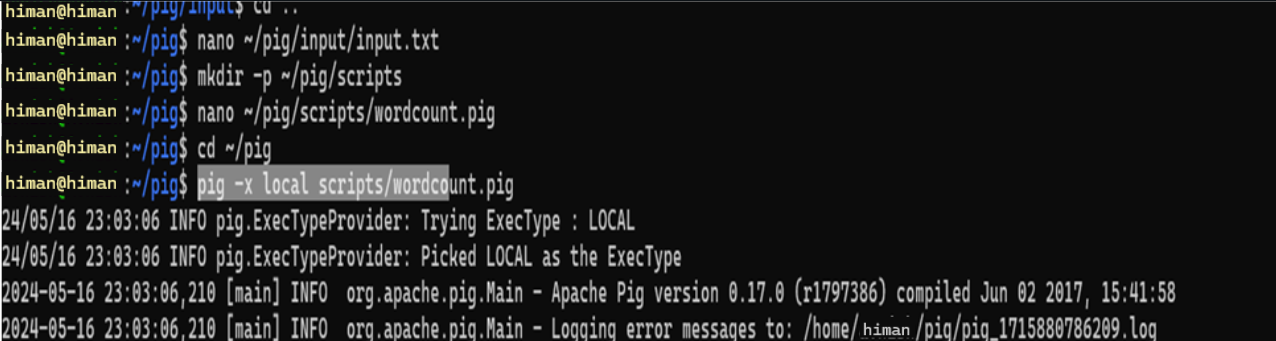
Create a text file named input.txt in the ~/pig/input directory:

nano ~/pig/input/input.txt

Add the following text to input.txt:

Hadoop is highly fault-tolerant, thanks to its replication model.

Hadoop is widely used in industry and academia for big data analytics.



Step 2: Create the Pig Script

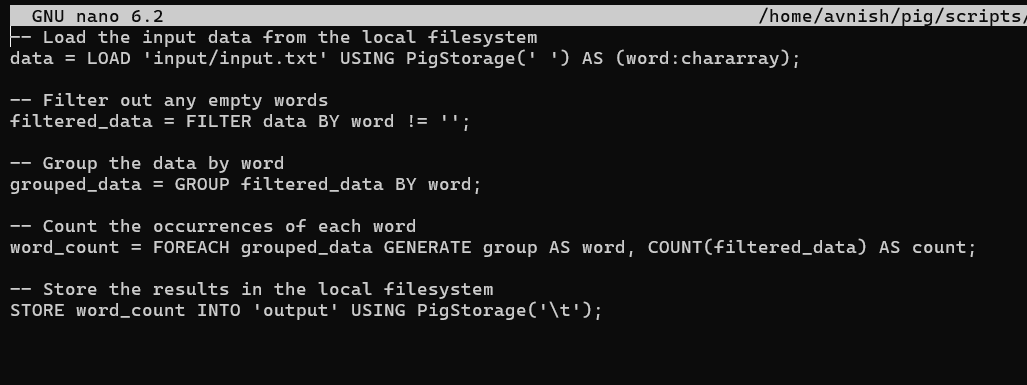
Create a directory for Pig scripts:

mkdir -p ~/pig/scripts

Create a Pig script file named wordcount.pig in the ~/pig/scripts directory:

nano ~/pig/scripts/wordcount.pig

Add the following content to wordcount.pig:



Step 3: Run the Pig Script in Local Mode

Navigate to the Pig directory:

cd ~/pig

Run the Pig script:

pig -x local scripts/wordcount.pig

##### Step 4: Verify the Output

Check the output directory:

ls ~/pig/output

View the output file:

cat ~/pig/output/part-r-00000

