Lab 01: A Gentle Introduction to Hadoop

CSC14118 Introduction to Big Data 20KHMT1

SmallData

2023-02-17

Contents

1 Lab 01: A Gentle Introduction to Hadoop

1.1 List of team members

ID	Full Name
20120366	Pham Phu Hoang Son
20120391	Ha Xuan Truong
20120393	Huynh Minh Tu
20120468	Nguyen Van Hai

1.2 Team's result

Section	Complete
Setting up SNC	100%
Introduction to MapReduce	100%
Running a warm-up problem: Word Count	100%
Bonus - Extended Word Count: Unhealthy relationships	100%
Bonus - Setting up Fully Distributed Mode	0%

1.3 Team reflection

Does your journey to the deadline have any bugs? How have you overcome it?

During the journey towards the deadline, we encountered several bugs that were related to Ubuntu, Hadoop installation, and errors while running Hadoop MapReduce jobs. In order to overcome these

challenges, we had to invest more time and effort. We also conducted research by reading documentation and watching tutorial videos. These resources provided us with useful insights and ideas for troubleshooting the issues. Additionally, we scheduled some online meetings to discuss and solve the problems together. Through these efforts, we were able to solve most of the problems we encountered and successfully complete the project.

What have you learned after this process?

Firstly, we learned the importance of clear communication among team members to ensure that everyone is on the same page and that tasks are completed efficiently. We also learned the importance of testing and debugging to ensure that any errors are caught and resolved early on in the process.

Secondly, we learned the importance of time management and task prioritization, as we encountered some unexpected challenges during the installation and setup process. This made it necessary for us to adjust our timeline and focus on the most critical tasks first.

Lastly, we learned the importance of continuous learning and self-improvement. We encountered some roadblocks that required us to do additional research and seek out new solutions, which allowed us to expand our knowledge and skills in Hadoop and MapReduce.

1.4 Setting up Single-node Hadoop Cluster

1.4.1 Step 1: Download java

1. The default Ubuntu repositories contain Java 8 and Java 11 both. Use the following command to install it.

   ```
   sudo apt update && sudo apt install openjdk-8-jdk
   ```

2. Once you have successfully installed it, check the current Java version:

   ```
   java -version
   ```

```
nvhai@20120468:~$ java -version
openjdk version "1.8.0_362"
OpenJDK Runtime Environment (build 1.8.0_362-8u362-ga-0ubuntu1~18.04.1-b09)
OpenJDK 64-Bit Server VM (build 25.362-b09, mixed mode)
nvhai@20120468:~$ dirname $(dirname $(readlink -f $(which java)))
/usr/lib/jvm/java-8-openjdk-amd64/jre
nvhai@20120468:~$
```

Figure 1.1: Download java

1.4.2 Step 2: Create User for Hadoop and install openSSH

1. Run the following command to create a new user with the name “hadoop”:

   ```
   sudo adduser hadoop
   ```

```
nvhai@20120468:~$ sudo adduser hadoop
[sudo] password for nvhai:
Adding user `hadoop' ...
Adding new group `hadoop' (1001) ...
Adding new user `hadoop' (1001) with group `hadoop' ...
Creating home directory `/home/hadoop' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for hadoop
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []: \
Is the information correct? [Y/n] Y
nvhai@20120468:~$ sudo adduser hadoop
```

Figure 1.2: Create new user

2. Switch to the newly created hadoop user:

   ```
   su - hadoop
   ```

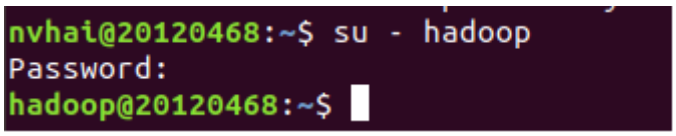

Figure 1.3: Change to hadoop user

3. Now configure password-less SSH access for the newly created hadoop user. Generate an SSH keypair first:

   ```
   ssh-keygen -t rsa
   ```

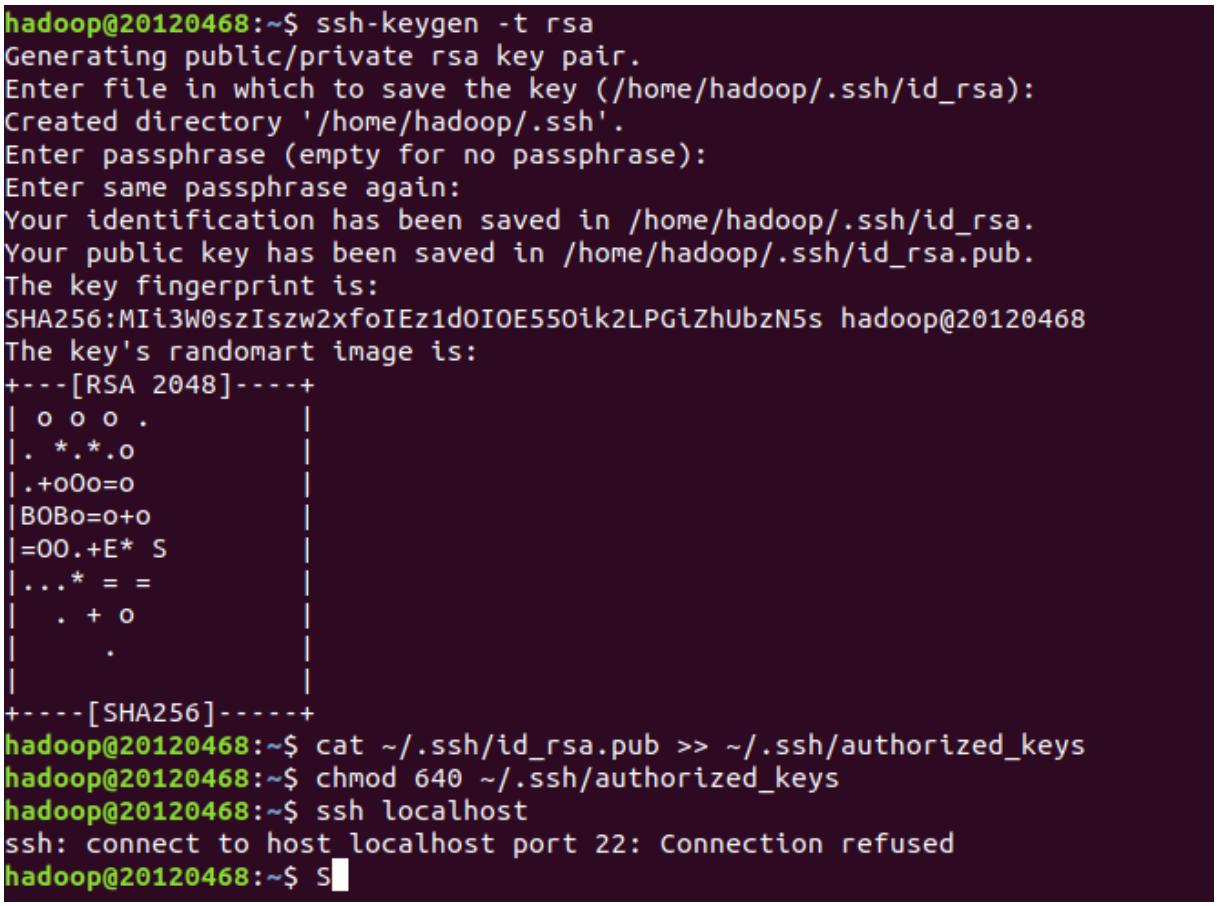

Figure 1.4: OpenSSH

4. Copy the generated public key to the authorized key file and set the proper permissions:

   ```
   cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
   chmod 640 ~/.ssh/authorized_keys
   ```

5. Now try to SSH to the localhost

   ```
   ssh localhost
   ```

   You will be asked to authenticate hosts by adding RSA keys to known hosts. Type yes and hit Enter to authenticate the localhost.

1.4.3 Step 3: Install Hadoop on Ubuntu

1. Use the following command to download Hadoop 3.3.4

   ```
   wget https://dlcdn.apache.org/hadoop/common/hadoop-3.3.4/hadoop-3.3.4.tar.gz
   ```

Figure 1.5: download hadoop

2. Once you've downloaded the file, you can unzip it to a folder on your hard drive

   ```
   tar xzf hadoop-3.3.4.tar.gz
   ```

3. Rename the extracted folder to remove version information. This is an optional step, but if you don't want to rename, then adjust the remaining configuration paths.

   ```
   mv hadoop-3.3.4 hadoop
   ```

4. Next, you will need to configure Hadoop and Java Environment Variables on your system. Open the ~/.bashrc file in your favorite text editor:

```
nano ~/.bashrc
```

Append the below lines to the file. You can find the JAVA_HOME location by running dirname $(dirname $(readlink -f $(which java))) command on the terminal.

```
  GNU nano 2.9.3                  /home/hadoop/.bashrc

# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HADOOP_HOME=/home/hadoop/hadoop
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export HADOOP_YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
```

Figure 1.6: setup environment

Save the file and close it.

5. Load the above configuration in the current environment

   ```
   source ~/.bashrc
   ```

6. You also need to configure JAVA_HOME in hadoop-env.sh file. Edit the Hadoop environment variable file in the text editor:

   ```
   nano $HADOOP_HOME/etc/hadoop/hadoop-env.sh
   ```

   Search for the “export JAVA_HOME” and configure it with the value found in step 1. See the below screenshot:

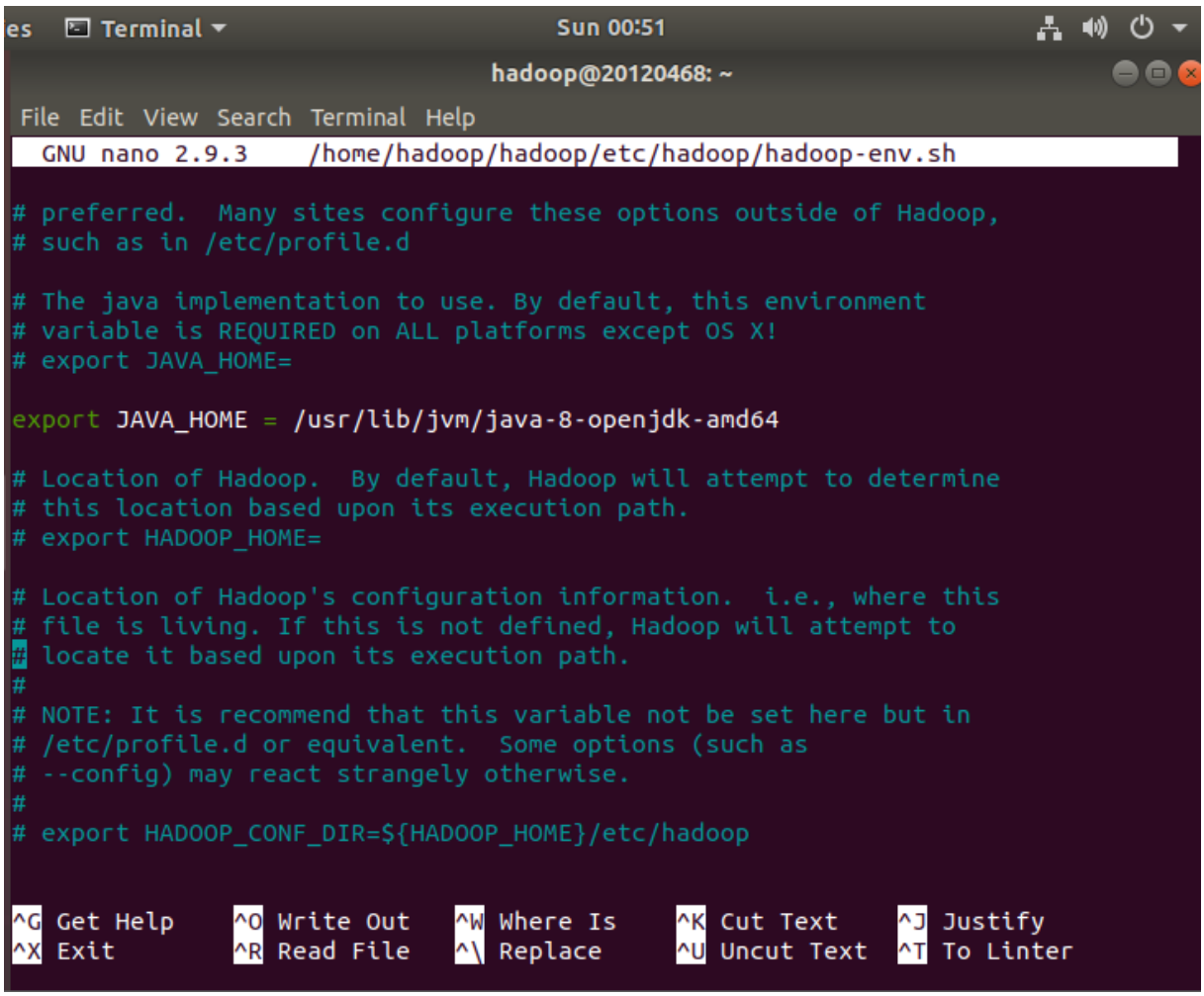

Figure 1.7: setup hadoop-env

Save the file and close it.

1.4.4 Step 4: Configuring Hadoop

Next is to configure Hadoop configuration files available under etc directory.

1. First, you will need to create the namenode and datanode directories inside the Hadoop user home directory. Run the following command to create both directories:

   ```
   mkdir -p ~/hadoopdata/hdfs/{namenode,datanode}
   ```

2. Next, edit the core-site.xml file and update with your system hostname:

   ```
   nano $HADOOP_HOME/etc/hadoop/core-site.xml
   ```

   Change the following name as per your system hostname:

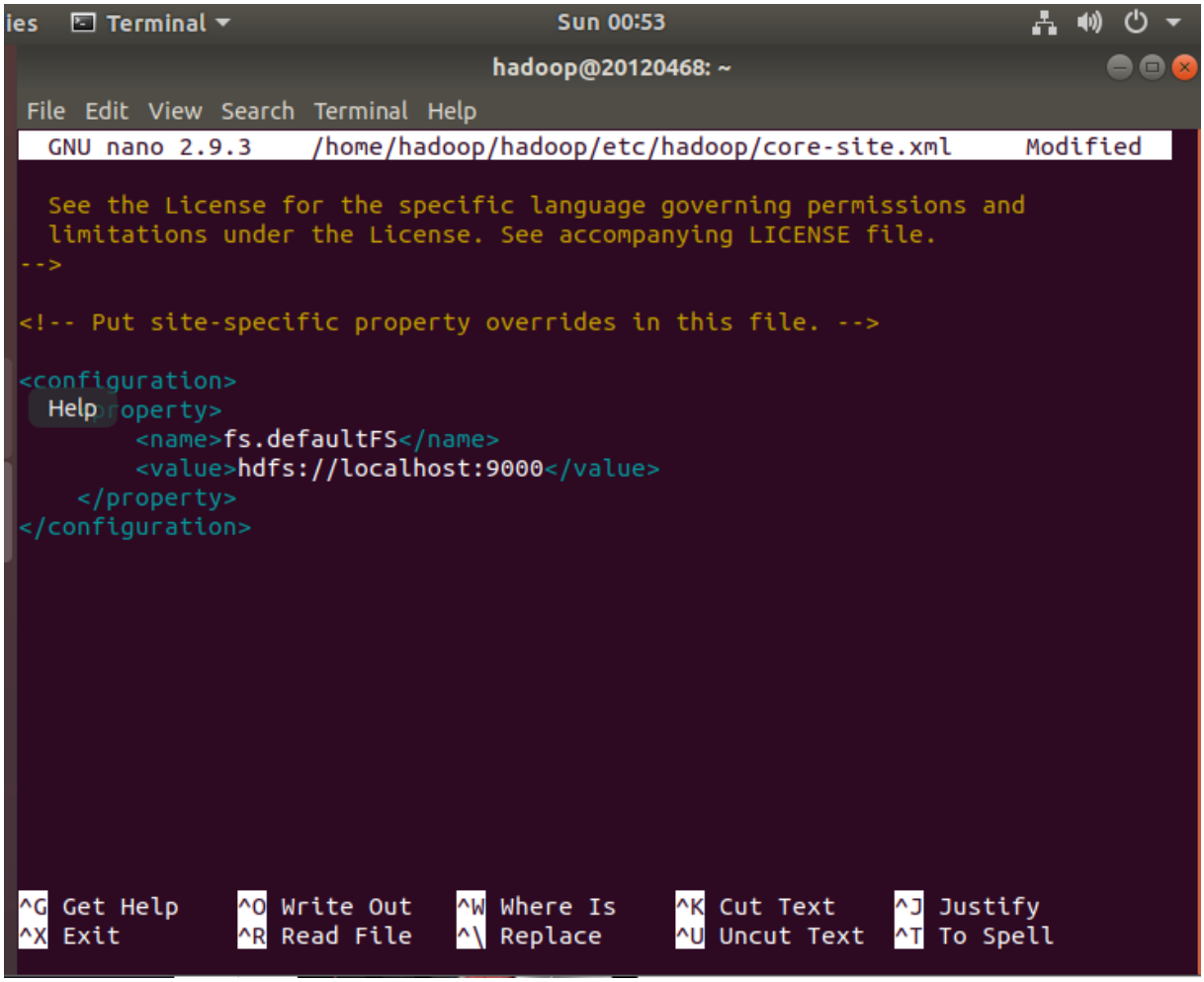

Figure 1.8: setup core-site

   Save and close the file.

3. Then, edit the hdfs-site.xml file

   ```
   nano $HADOOP_HOME/etc/hadoop/core-site.xml
   ```

   Change the NameNode and DataNode directory paths as shown below:

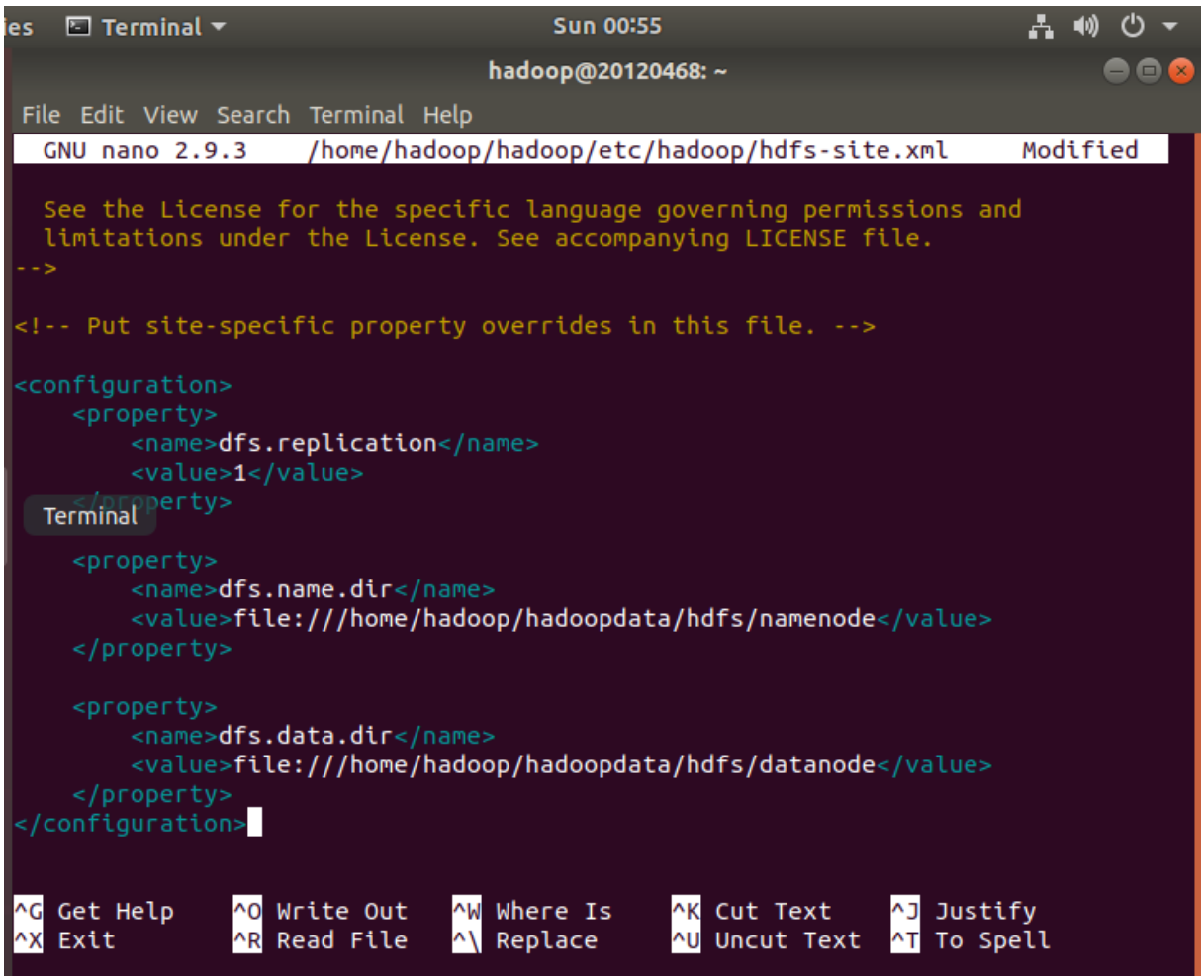

Figure 1.9: setup hdfs-site

Save and close the file.

4. Then, edit the mapred-site.xml file

   ```
   nano $HADOOP_HOME/etc/hadoop/mapred-site.xml
   ```

   Make the following changes:

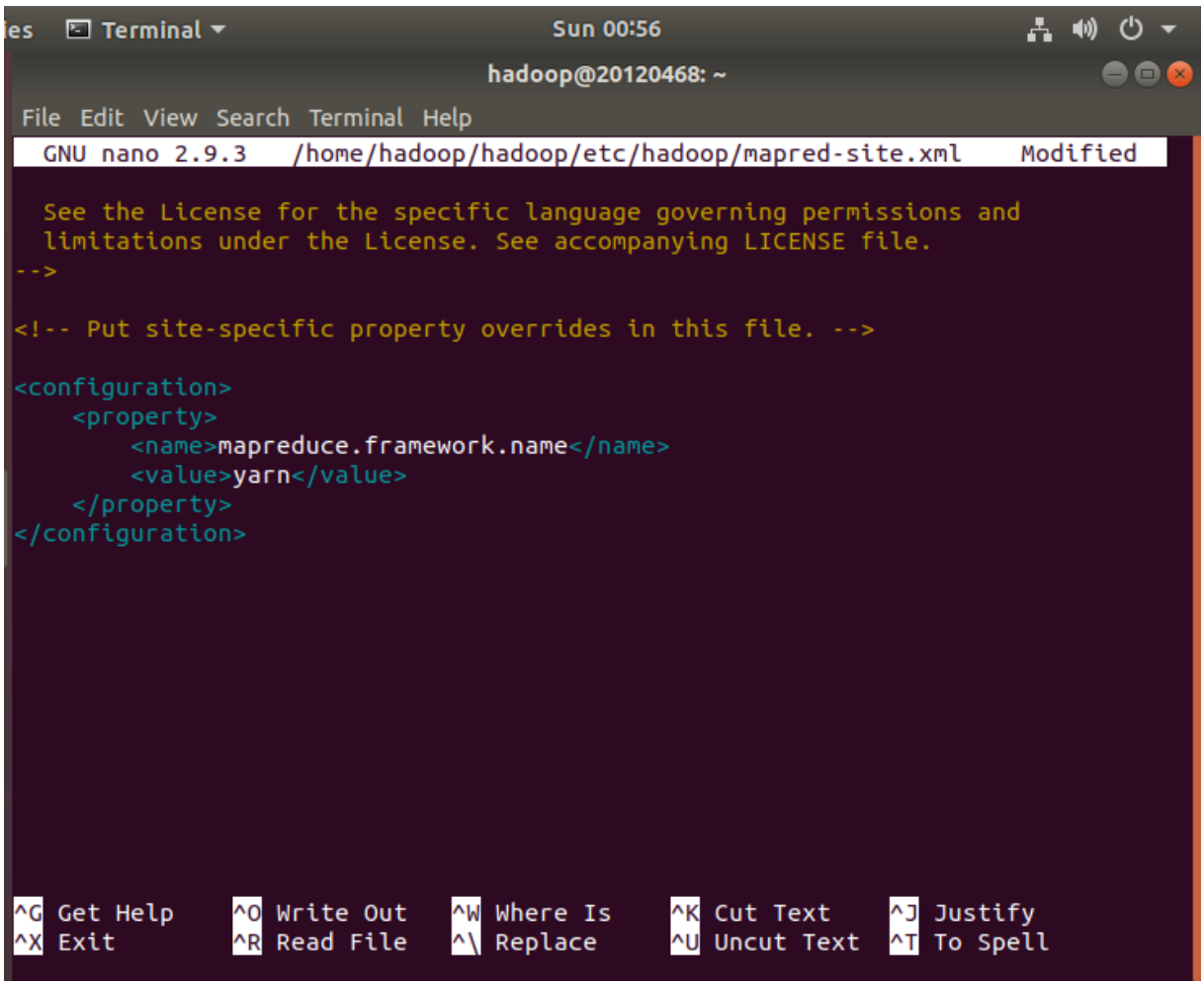

Figure 1.10: setup mapred-site

Save and close the file.

5. Then, edit the yarn-site.xml file

   ```
   nano $HADOOP_HOME/etc/hadoop/yarn-site.xml
   ```

   Make the following changes:

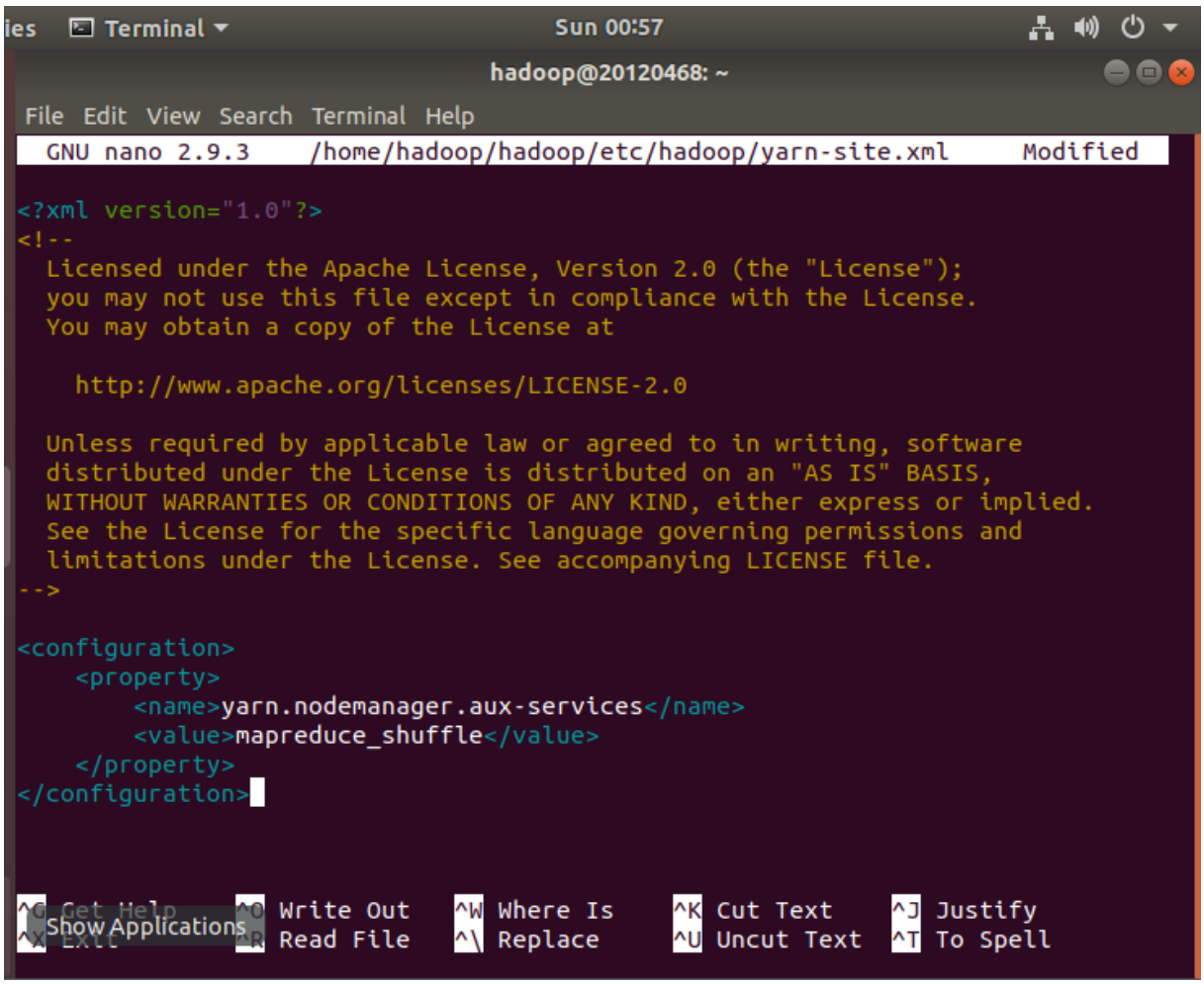

Figure 1.11: setup yarn-site

Save and close the file.

1.4.5 Step 5: Start Hadoop Cluster

Then start the Hadoop cluster with the following command

```
start-all.sh
```

Check jps

`jps`

Completed screenshots of the members:

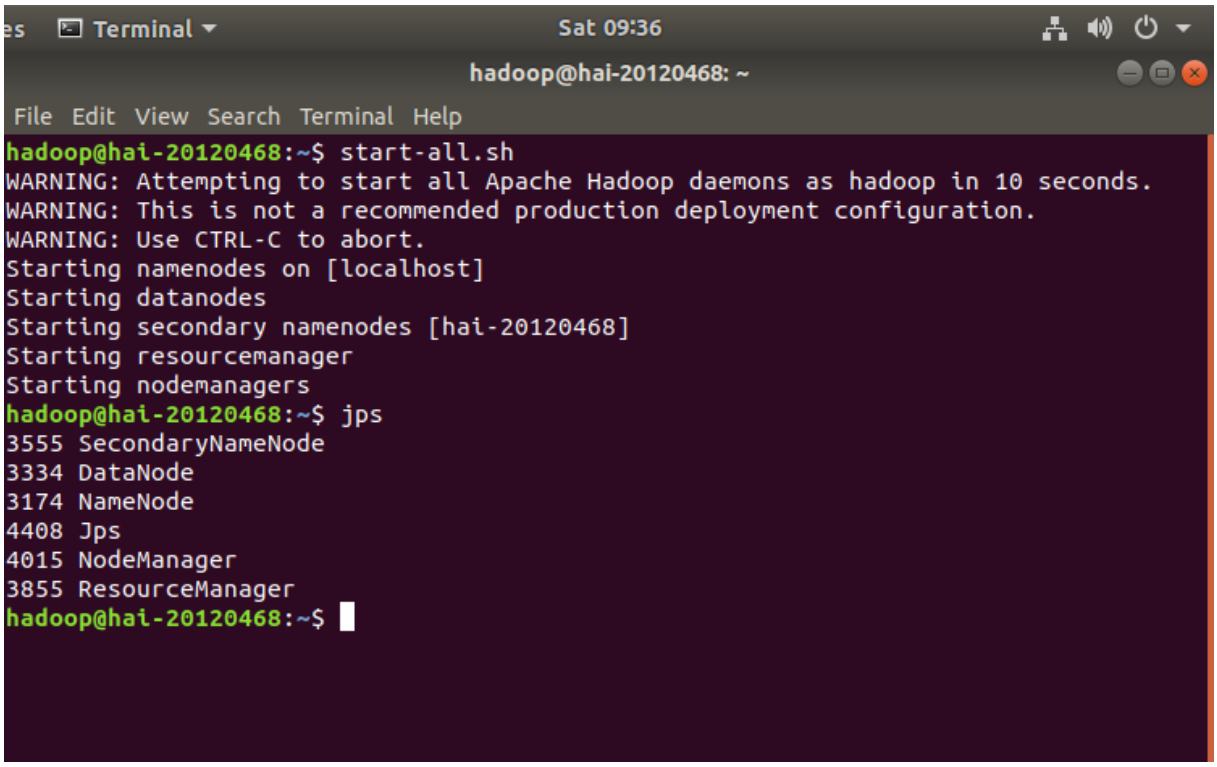

Figure 1.12: 20120468 - Nguyen Van Hai

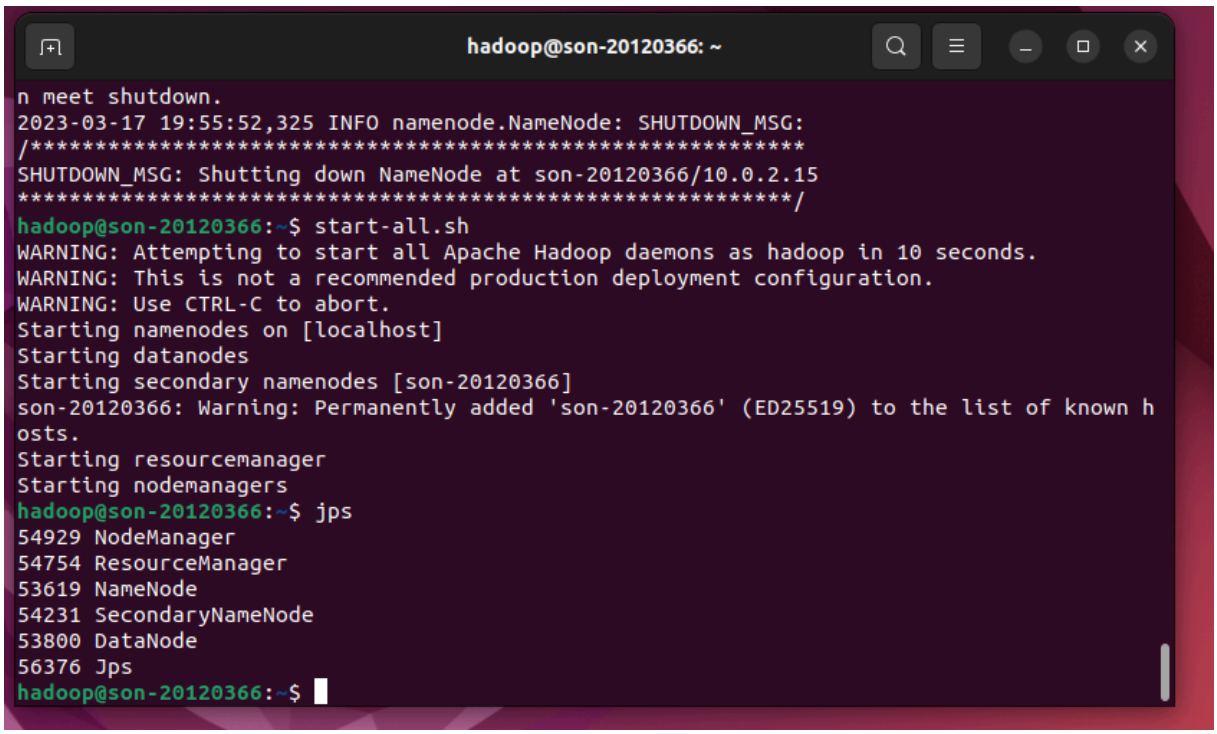

Figure 1.13: 20120366 - Pham Phu Hoang Son

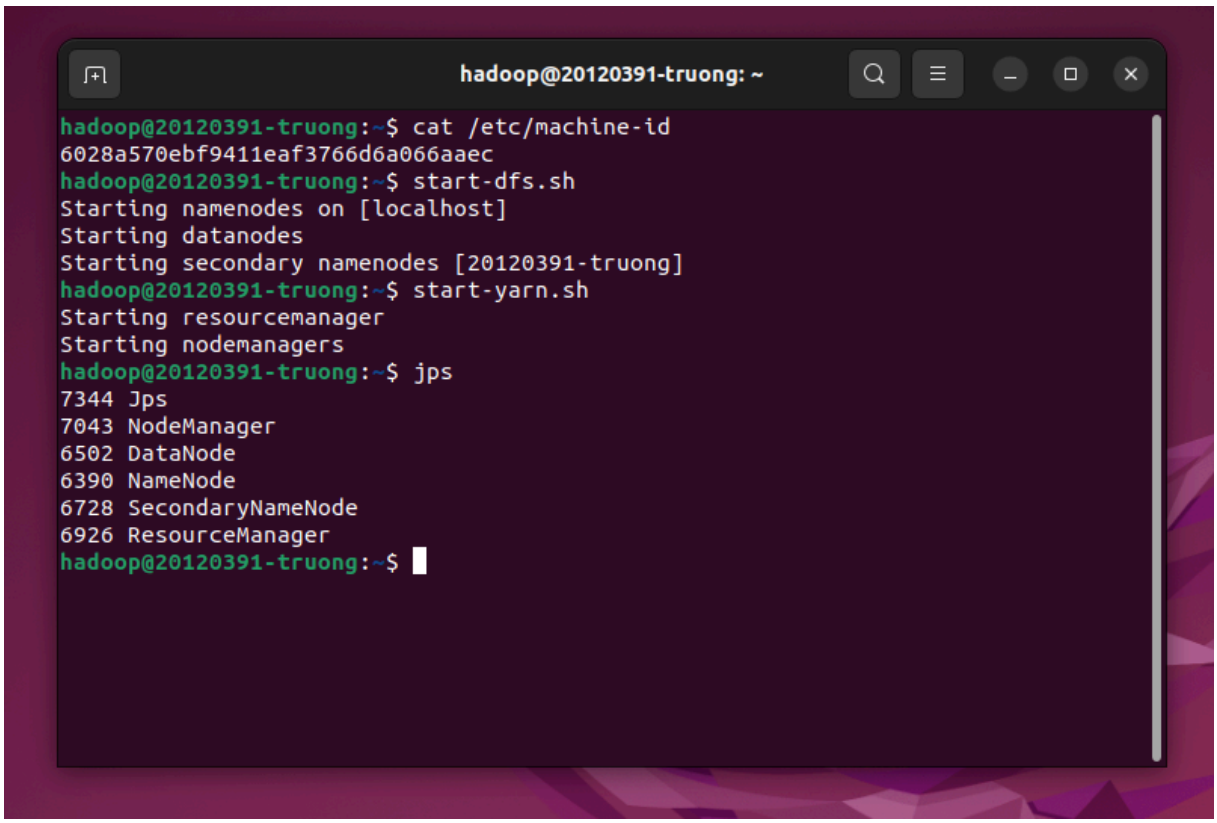

Figure 1.14: 20120391 - Ha Xuan Truong

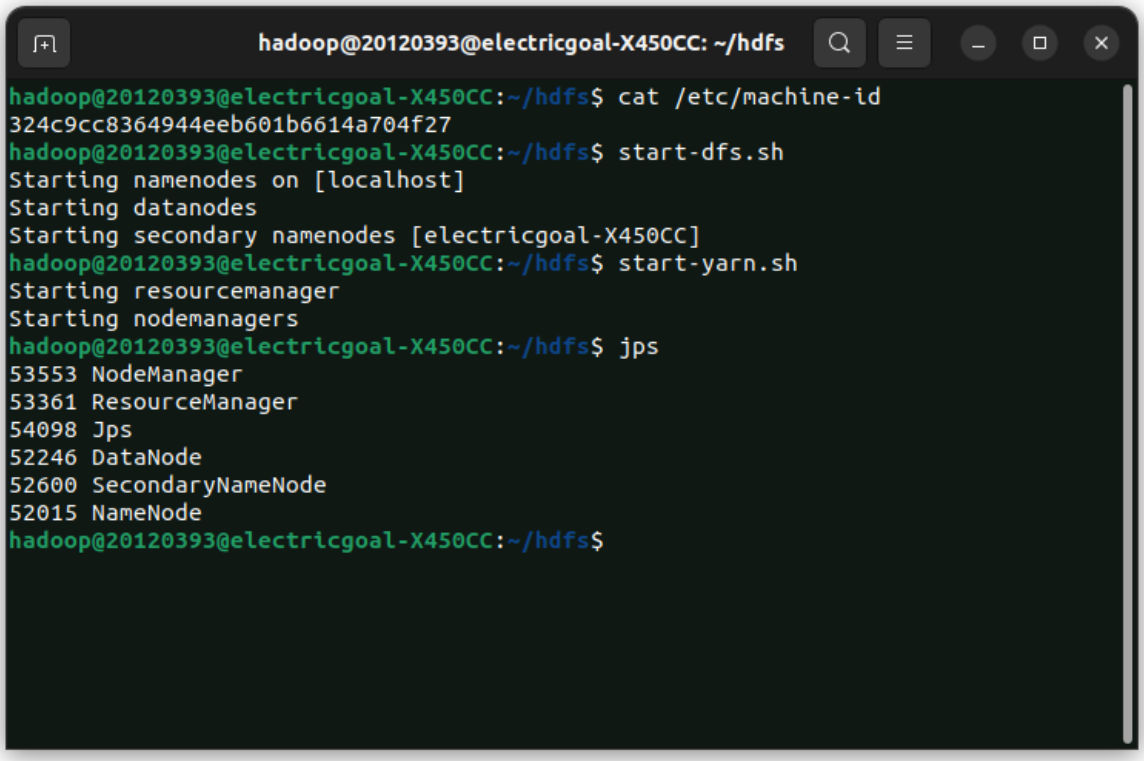

Figure 1.15: 20120393 - Huynh Minh Tu

1.5 Introduction to MapReduce

1. How do the input keys-values, the intermediate keys-values, and the output keys-values relate?

`Answer:`

- Input keys-values: The input data is divided into splits and represented as key-value pairs. Each input key-value is read into the MapReduce job using a RecordReader, which is responsible for reading the input data and converting it into key-value pairs.
- Intermediate keys-values: The map function processes the input keys-values and generates intermediate key-value pairs. The intermediate keys and values may be different from the input keys-values, depending on how the map function processes the data. The intermediate key-value pairs are sorted and grouped by key before being passed to the reduce function.
- Output keys-values: The reduce function generates the final output keys-values based on the intermediate key-value pairs that are passed to it. The output keys-values may be different from the intermediate keys-values, depending on how the reduce function processes the data.

The output keys-values are typically written to a distributed file system, such as HDFS, or to a database. The output of the MapReduce job can be used as input to other MapReduce jobs or as input to other applications.

2. How does MapReduce deal with node failures?

`Answer:`

Worker failure: The master node send heartbeat to each worker node. If a worker node fails, the master reschedule the tasks handled by the worker.

Master failure: The whole MapReduce job gets restarted through a different master based on checkpointed state of the failured master.

3. What is the meaning and implication of locality? What does it use?

`Answer:`

The concept of locality in the MapReduce refers to the idea that it is beneficial to process data on the same node where the data is stored, rather than moving it across the network to another node for processing. This is known as data locality.

MapReduce uses the concept of data locality to optimize the processing of data. The MapReduce framework is designed to distribute processing tasks to the nodes where the data is stored, in order to maximize data locality. When processing a large dataset, the framework splits the data into smaller chunks and distributes them across the cluster. Then, the Map tasks are scheduled on the same node where the data is stored, so that the data can be processed locally. Finally, the Reduce tasks are scheduled to aggregate the intermediate results generated by the Map tasks, again with the goal of minimizing data movement across the network.

4. Which problem is addressed by introducing a combiner function to the MapReduce model?

`Answer:`

The problem that is addressed by introducing a combiner function is the excessive duplicate data transfer during the shuffling phase of the MapReduce job. Without a combiner function, all the intermediate key-value pairs generated by the map tasks are transferred over the network to the reduce tasks, resulting in high network traffic and increased processing time.

By introducing a combiner function, the amount of data that needs to be transferred over the network is reduced, resulting in faster processing times and reduced network traffic. The combiner function helps to group together intermediate key-value pairs with the same key and perform a local aggregation, reducing the number of key-value pairs that need to be transferred. This is particularly useful when the same intermediate key appears multiple times across the map outputs.

1.6 Running a warm-up problem: Word Count

Use Eclipse IDE to run MapReduce on Ubuntu

1.6.1 Step 0: Install Eclipse on Ubuntu (if you had installed, please go to next step)

```
sudo snap install --classic eclipse
```

1.6.2 Step 1: Create new Java project

Open Eclipse, select **File** -> **New** -> **Java project**

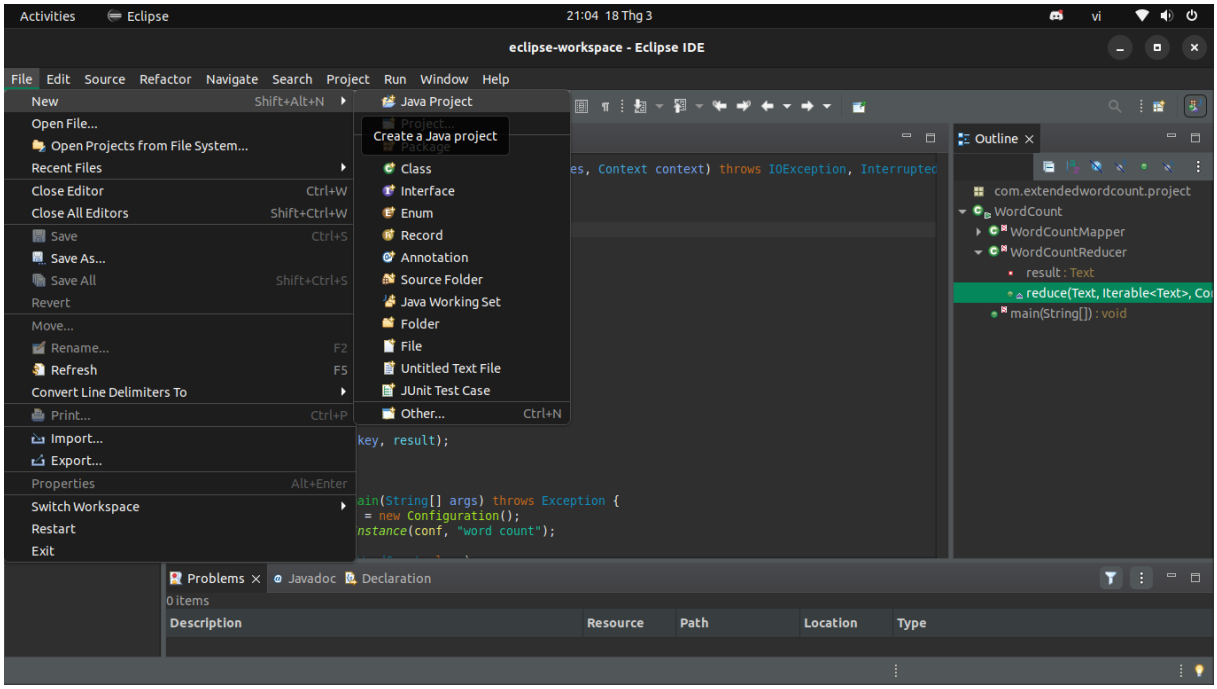

Figure 1.16: Create new Java project

Enter project name and click on **Next** button

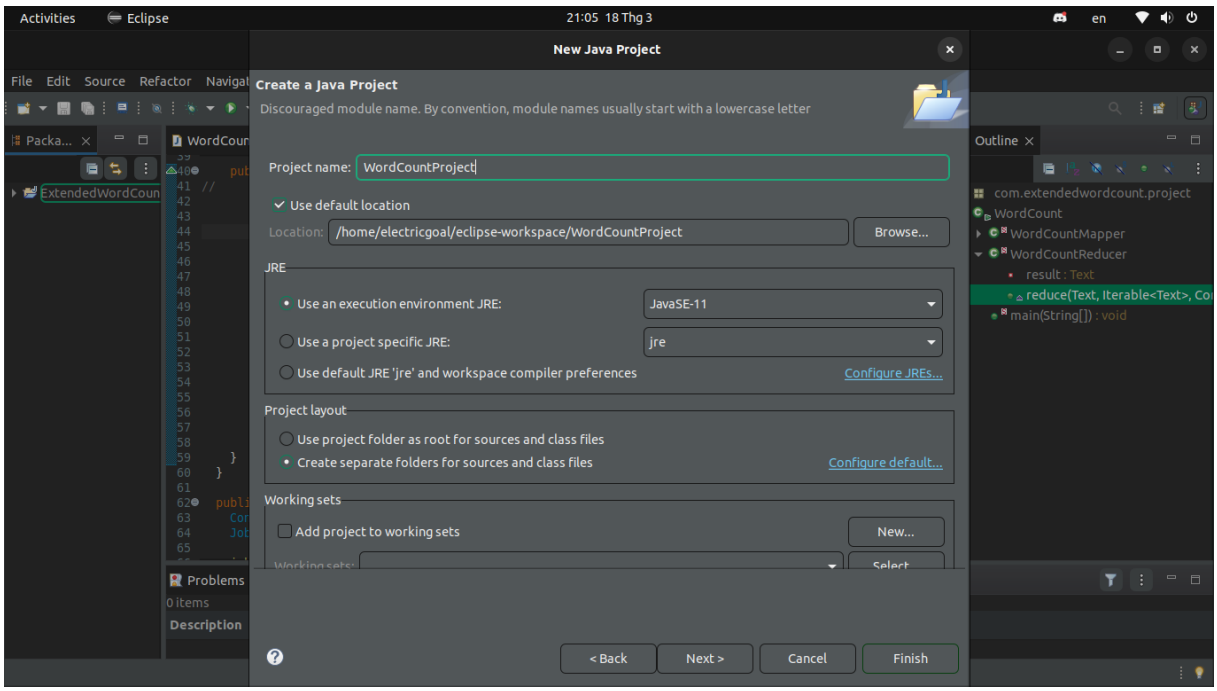

Figure 1.17: Click on Next button

Click on **Finish** button

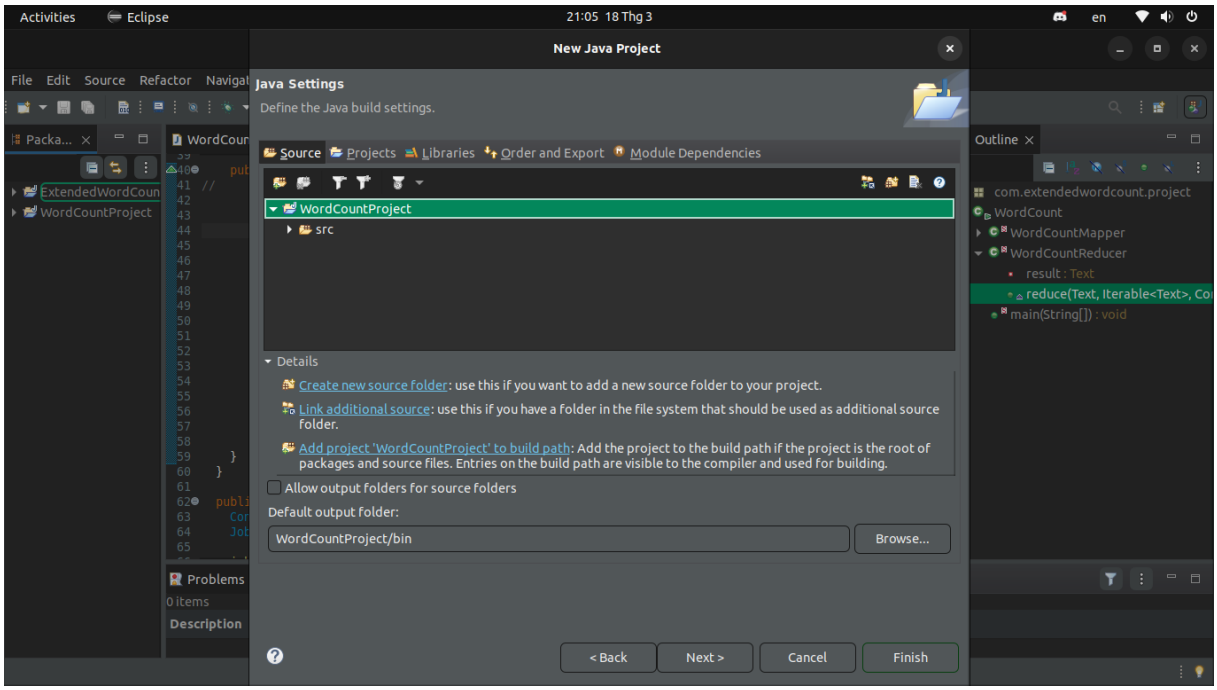

Figure 1.18: Click on Finish button

Result looks like this

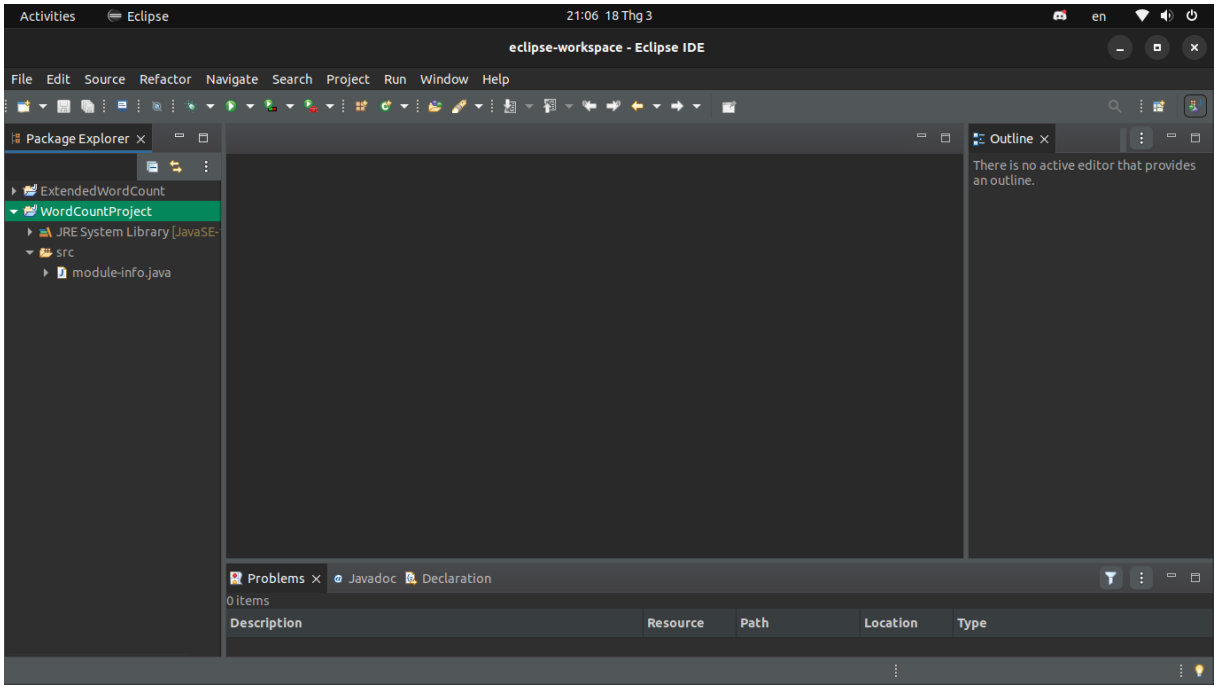

Figure 1.19: Result

1.6.3 Step 2: Delete file *module-info.java*

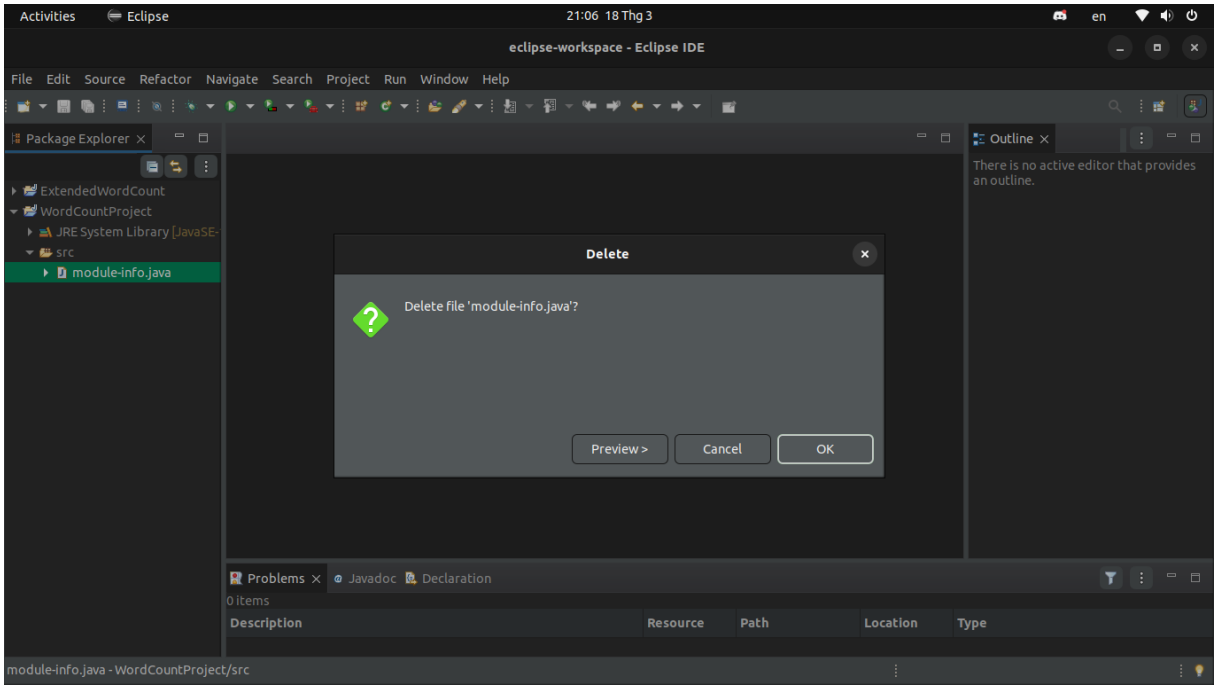

Figure 1.20: Delete module-info.java

1.6.4 Step 3: Create Java package

Right click on project name, select **New** -> **Package**

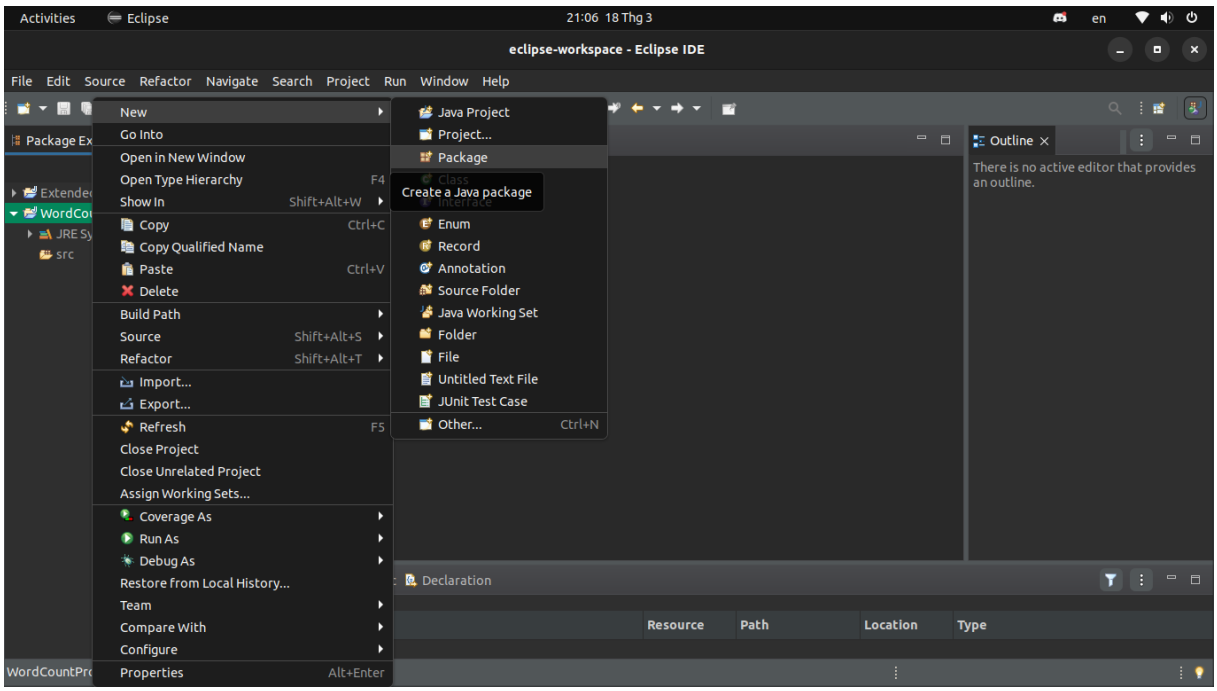

Figure 1.21: Create new Java package

Enter Package name and click on **Finish** button

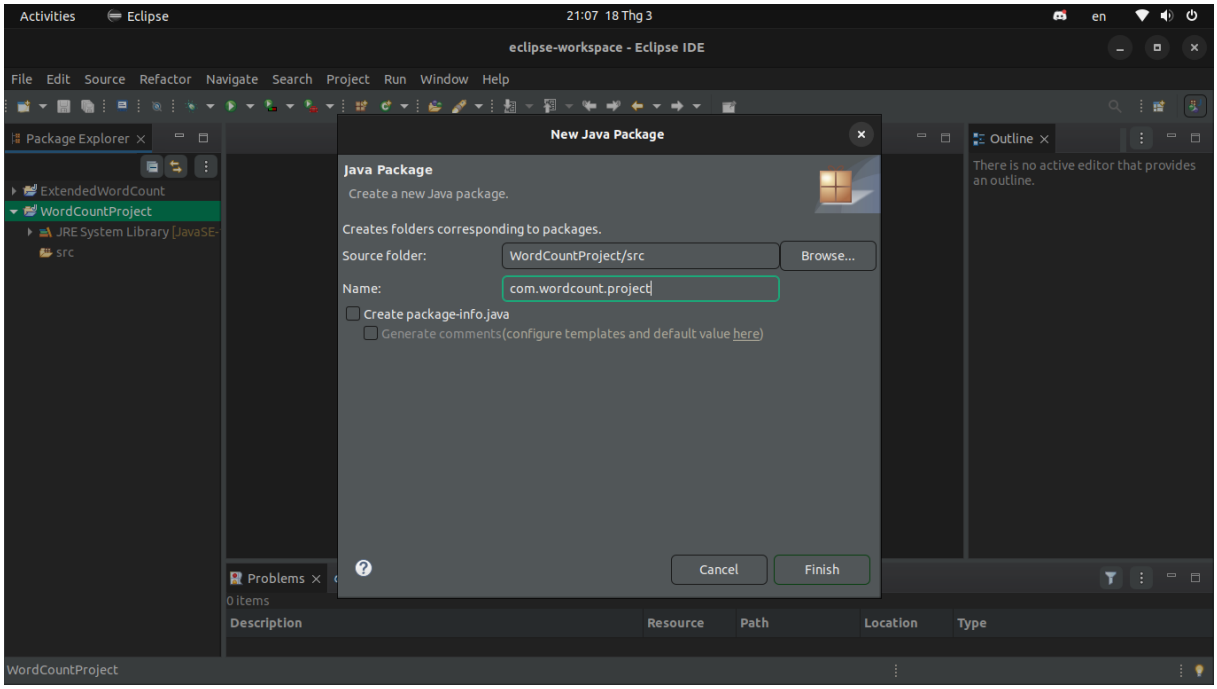

Figure 1.22: Click on Finish button

1.6.5 Step 4: Create Java class

Right click on project name, select **New** -> **Class** to create a Java class

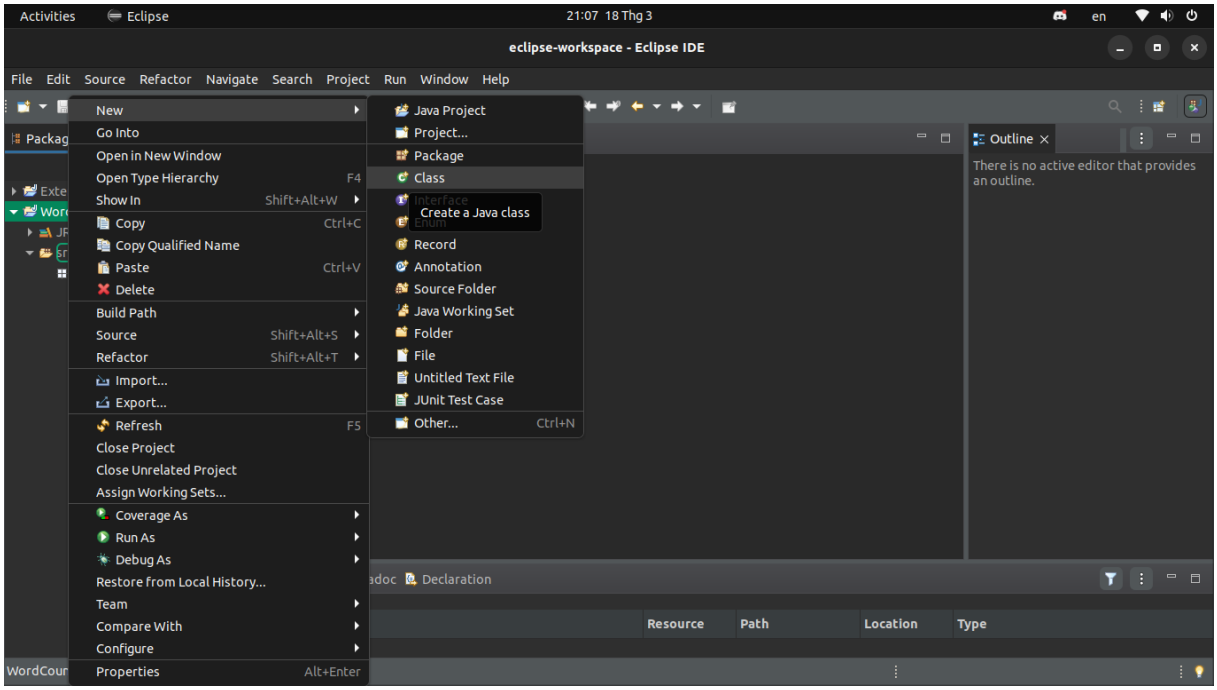

Figure 1.23: Create new Java class

Enter Class name and click on **Finish** button

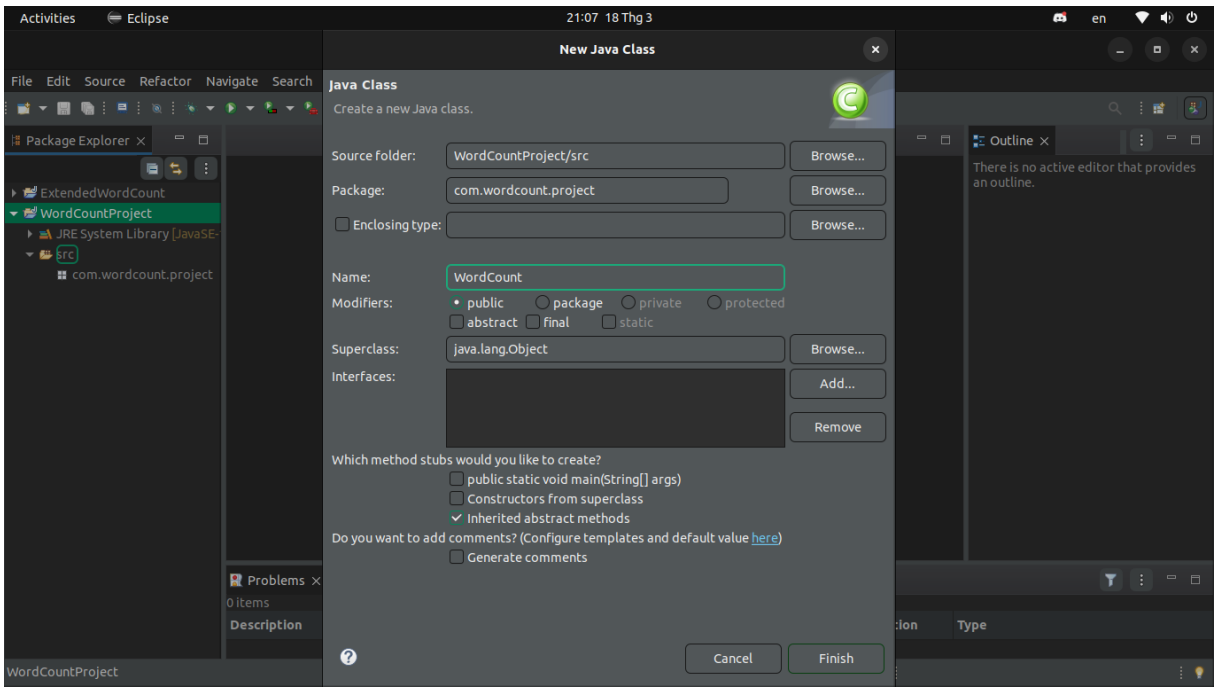

Figure 1.24: Click on Finish button

1.6.6 Step 5: Paste WordCount code to the *WordCount.java* file just created

You should see many errors

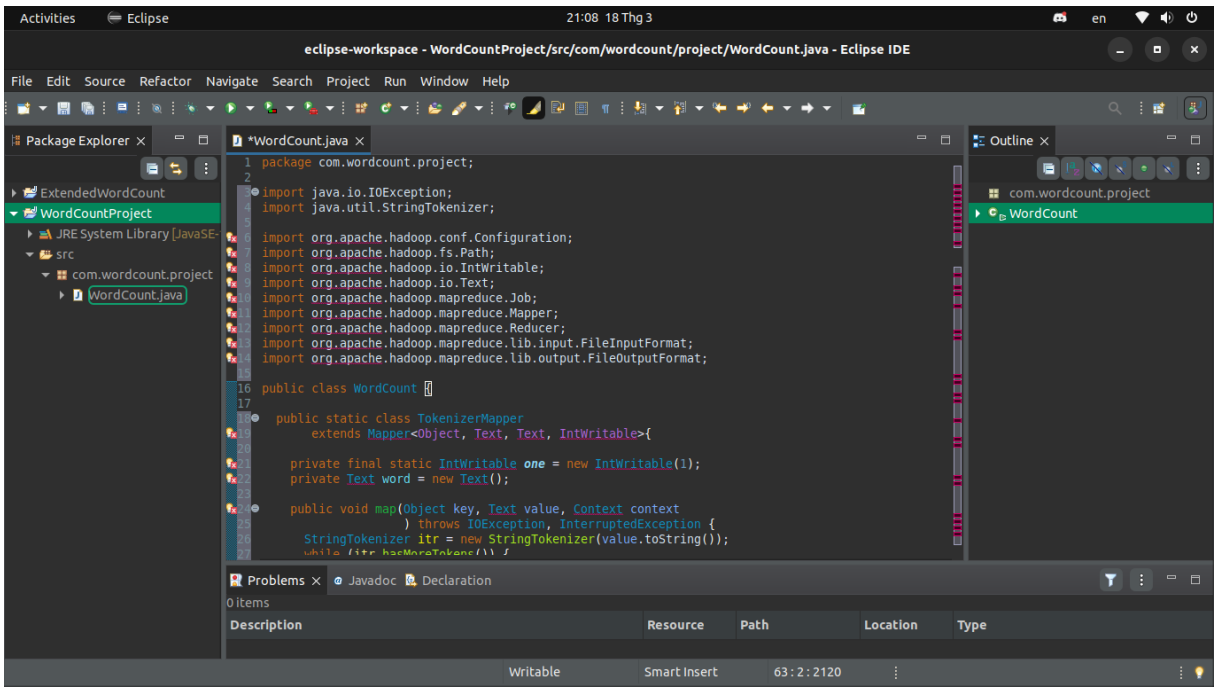

Figure 1.25: Create WordCount code

1.6.7 Step 6: Configure build path for the project

Right click on project name, select **New** -> **Build Path** -> **Configure Build Path**

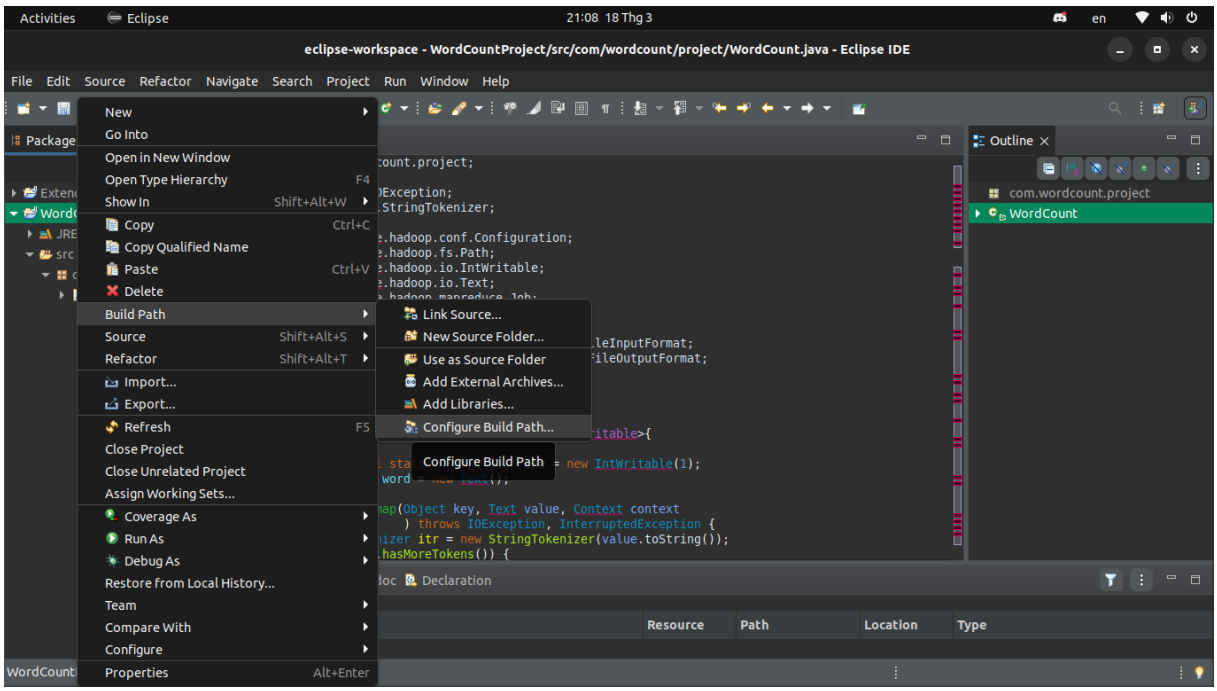

Figure 1.26: Configure Build Path

Click on the **Libraries** tab

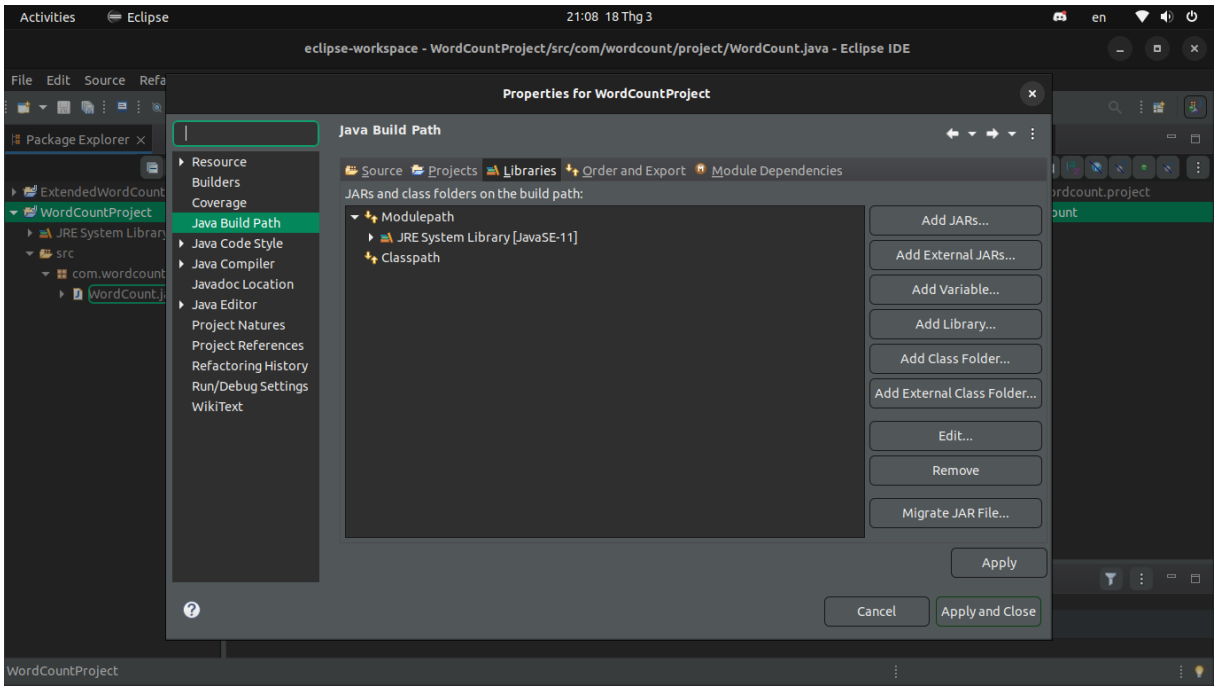

Figure 1.27: Libraries tab

Select **Classpath** section and click on the **Add External JARs** button

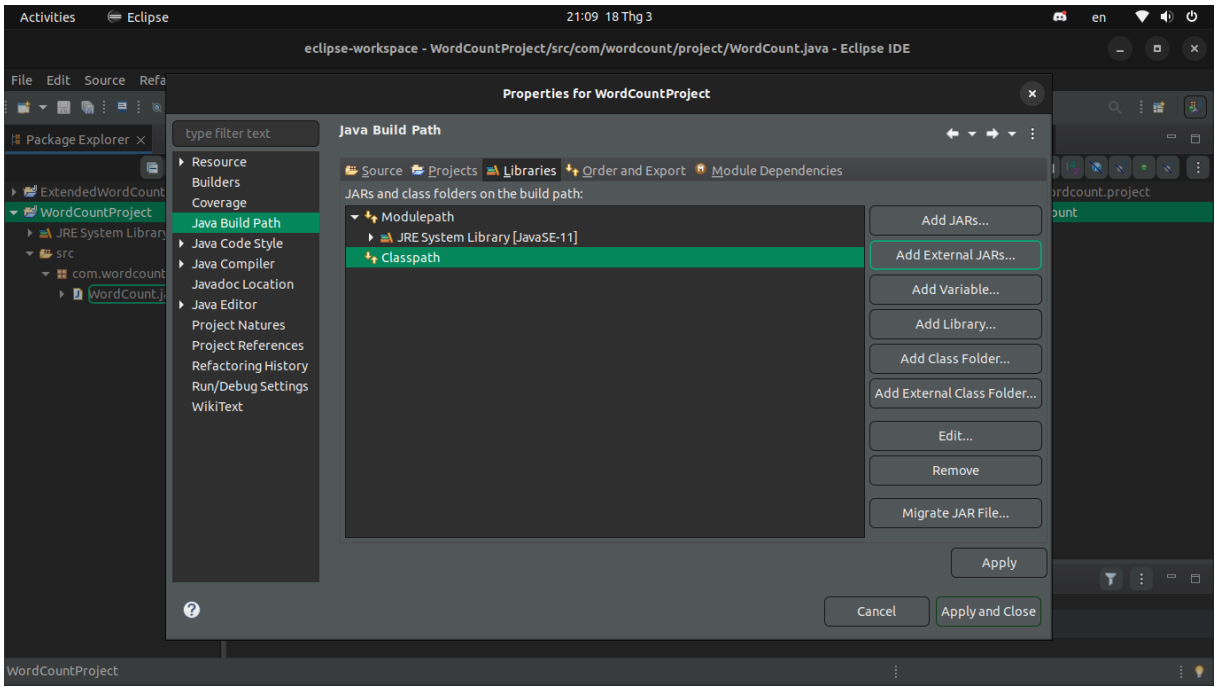

Figure 1.28: Add External JARs

Navigate to the Hadoop installation directory and select the following JAR files:

- hadoop-mapreduce-client-core-<version>.jar
- hadoop-mapreduce-client-common-<version>.jar
- hadoop-mapreduce-client-jobclient-<version>.jar
- hadoop-common-<version>.jar

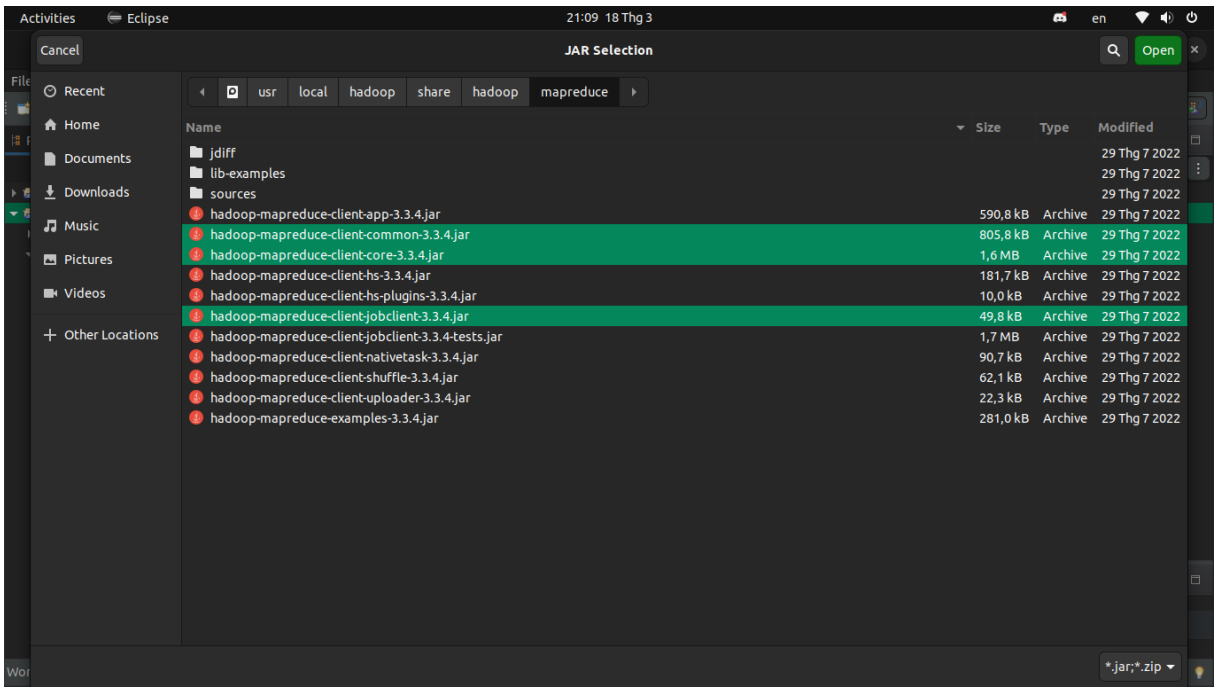

Figure 1.29: Select JAR files - 1

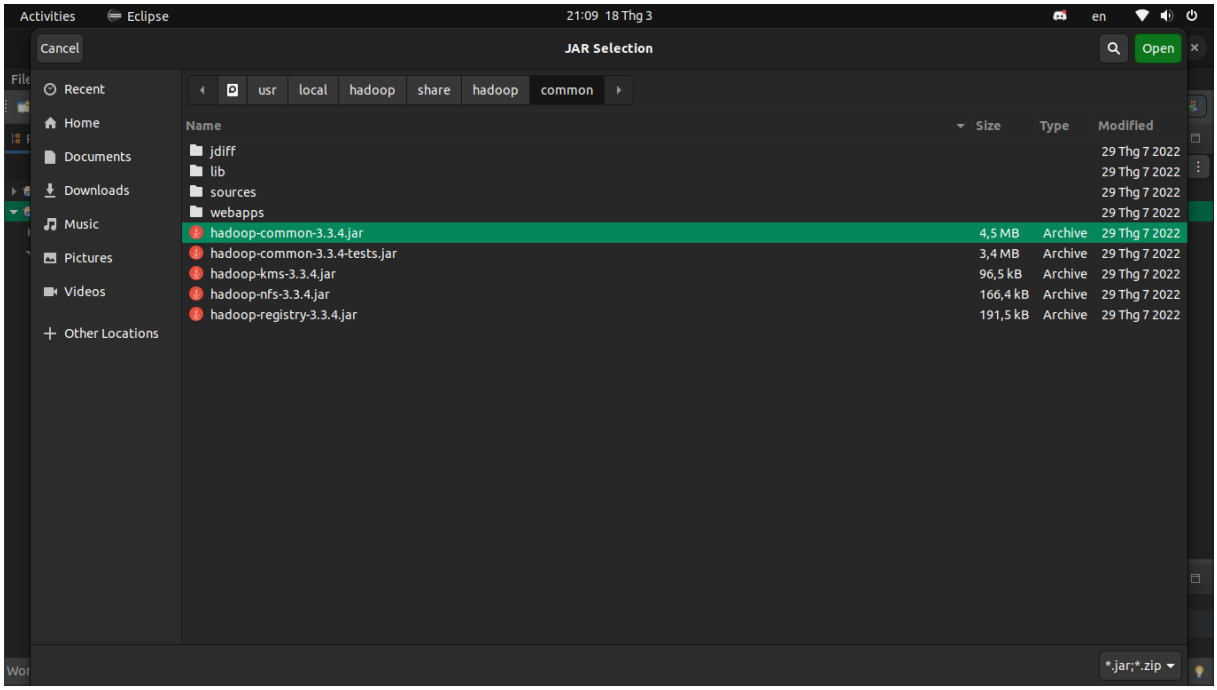

Figure 1.30: Select JAR files - 2

Click on the button **Apply and Close**

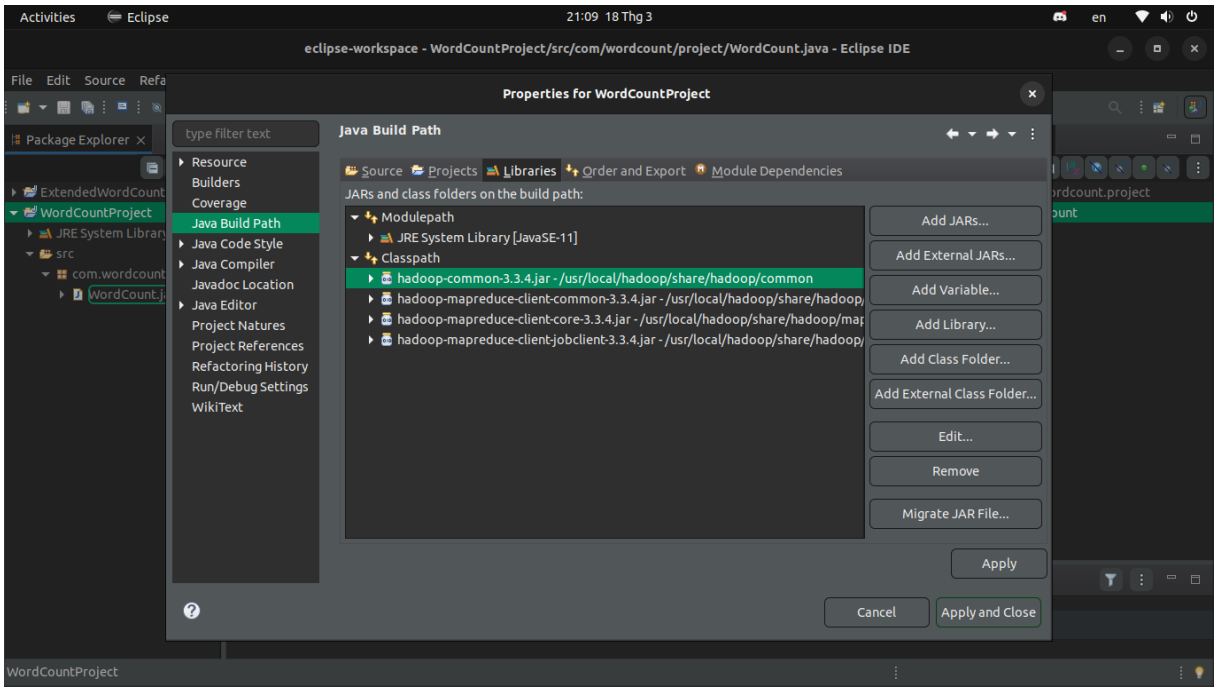

Figure 1.31: Apply and Close

After that, the errors should disappear

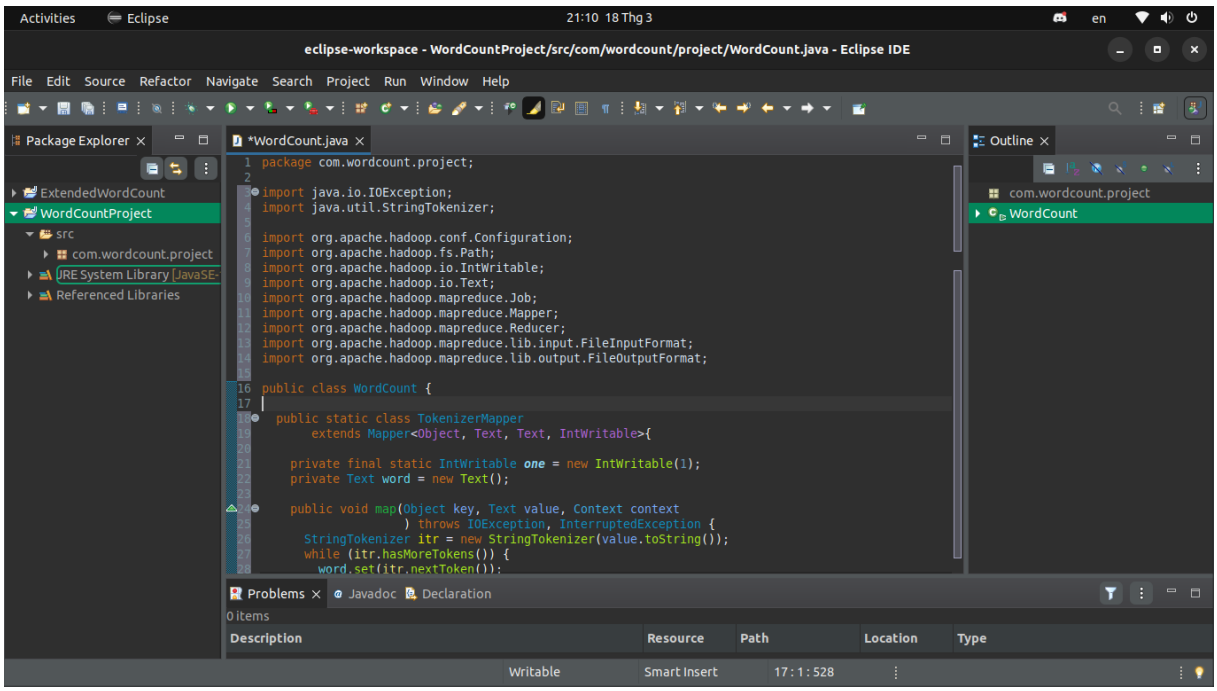

Figure 1.32: Result screen

1.6.8 Step 7: Export to JAR file

Right click to project name, select **Export**. You should see this screen, click on **JAR file** -> **Next**

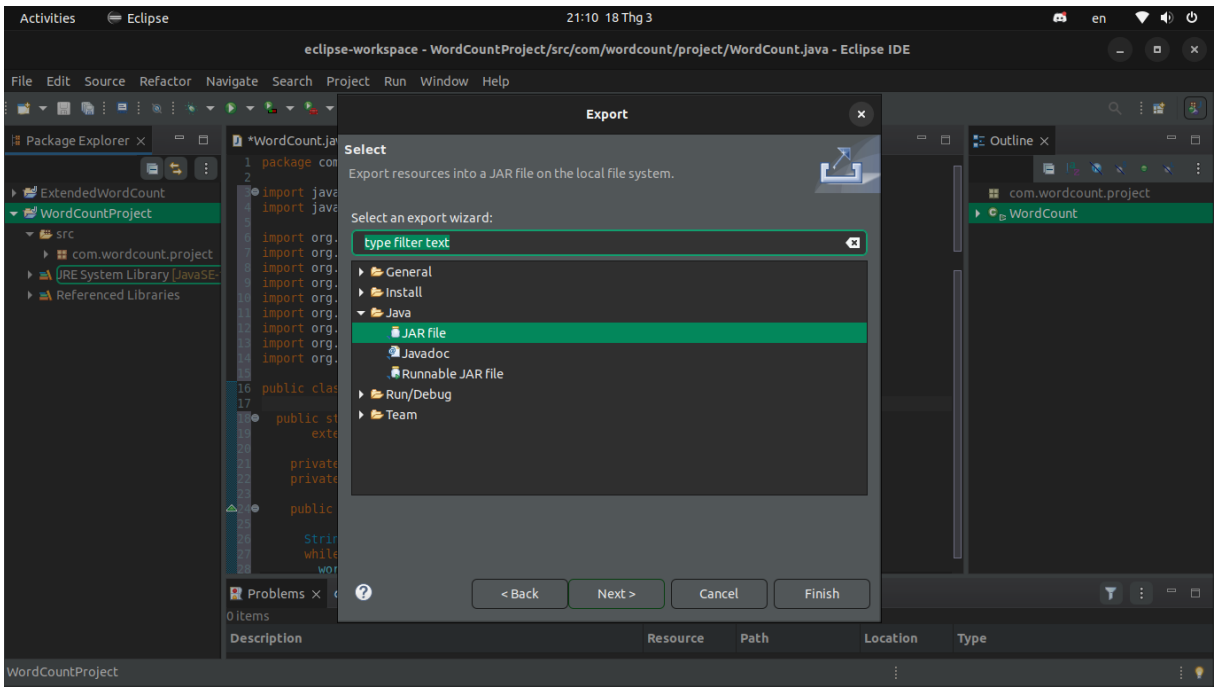

Figure 1.33: Select JAR file

Enter name of jar file and path to save this jar file and. Once done, click on **Next** button

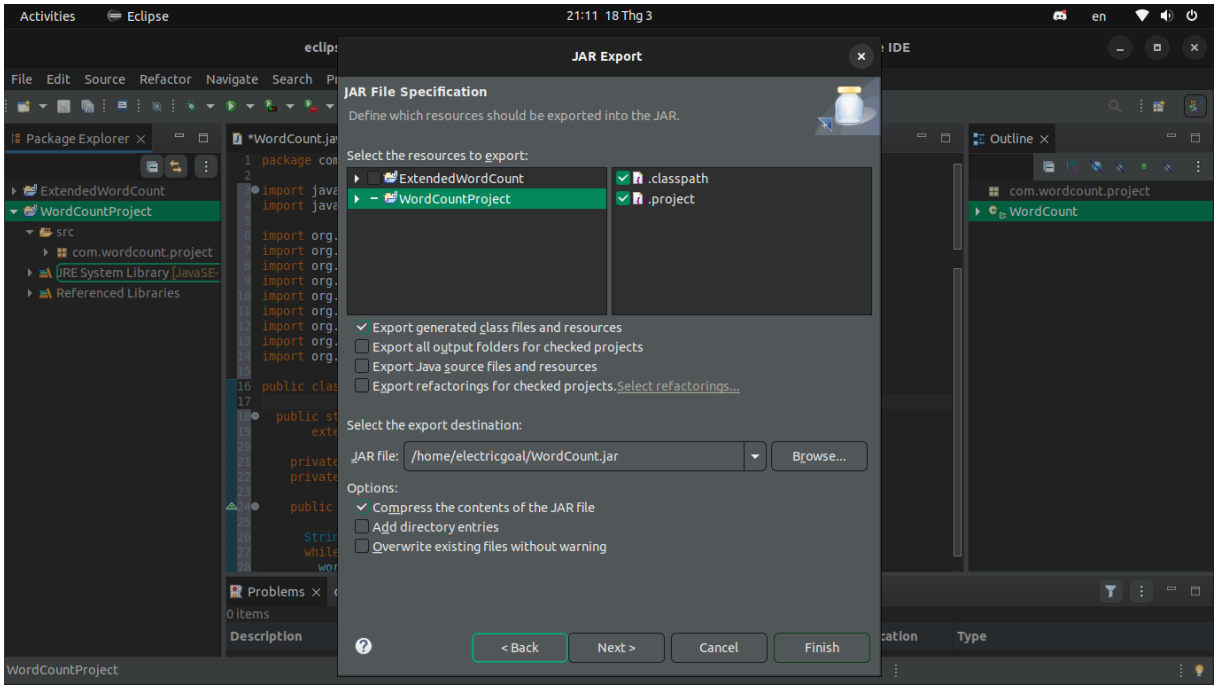

Figure 1.34: Click on Next button

Click on **Next** button until see this screen and browse the the package in this project. Once done, click on **Finish** button

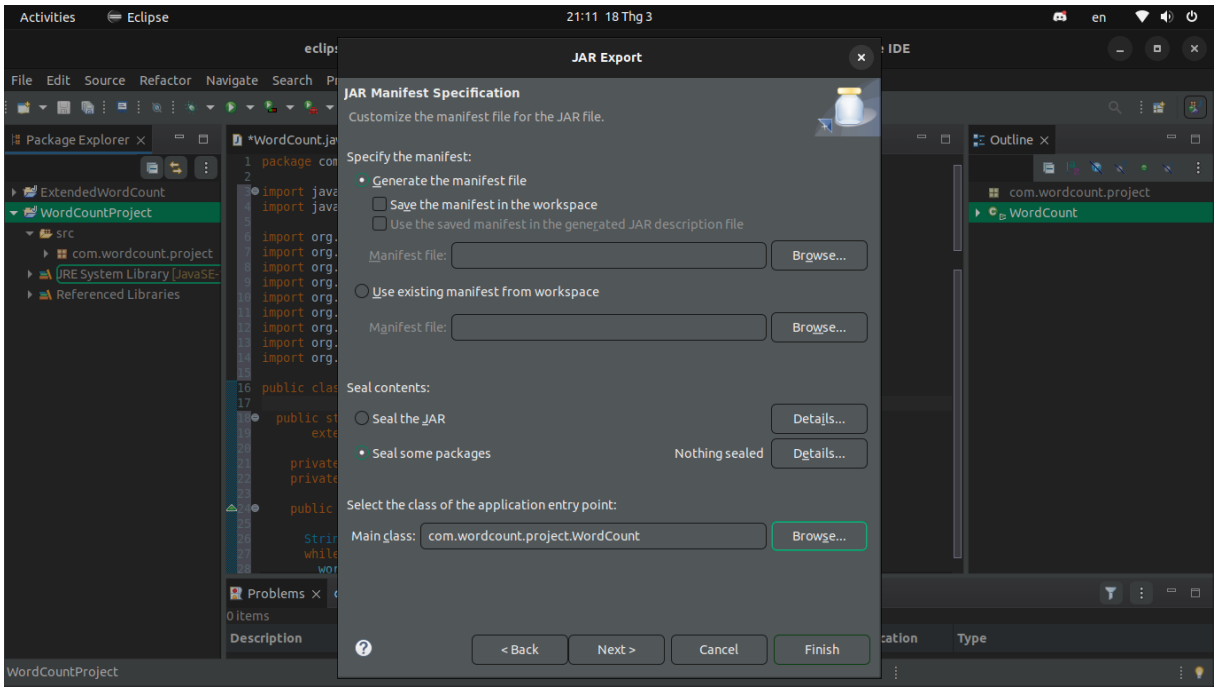

Figure 1.35: Click on Finish button

After all, you will get the Jar file

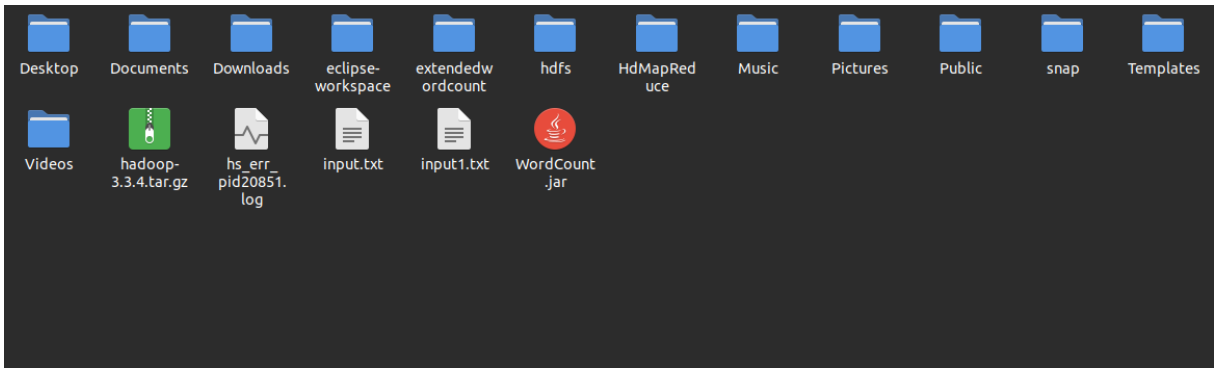

Figure 1.36: Result screen

1.6.9 Step 8: Prepare to run MapReduce

Create new folder name “wordcount” in HDFS

```
hadoop fs -mkdir -p /<your-favorite-path>/worldcount
```

Create “input” folder in “wordcount” folder to store input file

```
hadoop fs -mkdir -p /<your-favorite-path>/wordcount/input
```

Put *input.txt* file into “input” directory

```
hadoop fs -put /<local_file_path>/input.txt /<your-favorite-path>/wordcount/input
```

Open browser an enter http://localhost:9870, you should see the screen like this

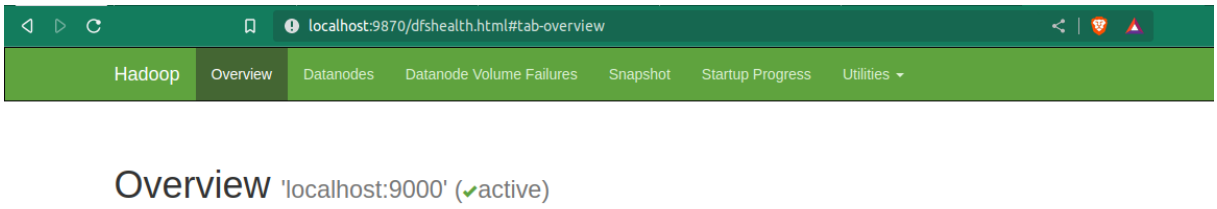

Figure 1.37: Open http://localhost:9870

Click on **Utilities** tab -> **Browse the file system**

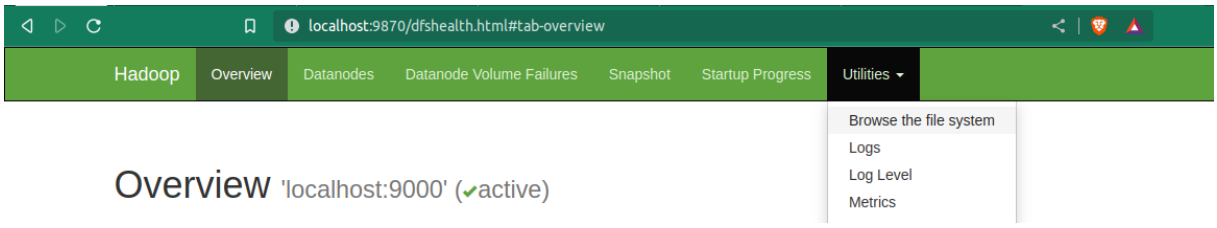

Figure 1.38: Browse the file system

Browse to your “wordcount” directory, you should see “input” folder. Click on it you will see **input.txt** file

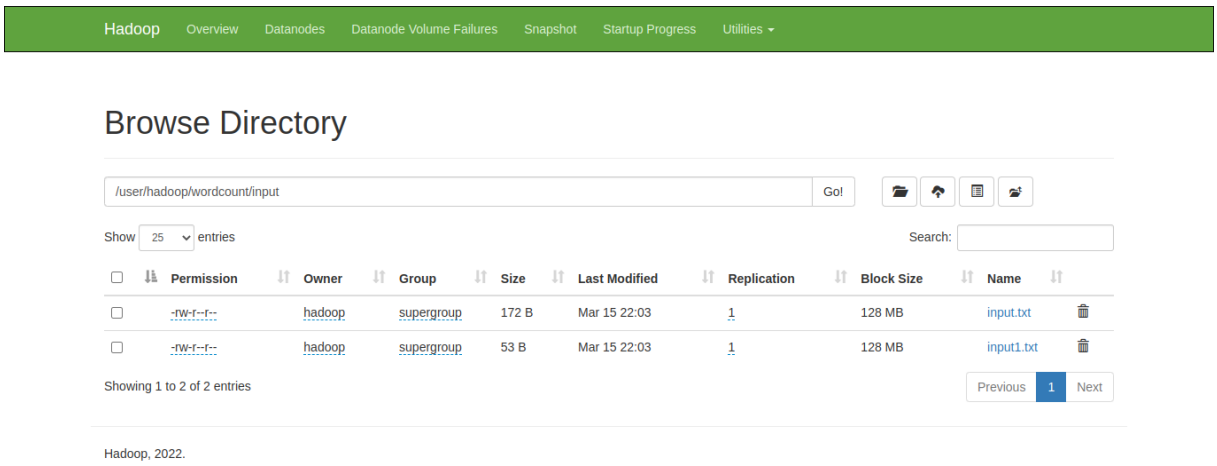

Figure 1.39: Input folder

1.6.10 Step 9: Run MapReduce

```
hadoop jar WordCount.jar /<your-favorite-path>/wordcount/input/input.txt /<your-
favorite-path>/wordcount/output

# In my case, <your-favorite-path> is user/hadoop
hadoop jar WordCount.jar /user/hadoop/wordcount/input/input.txt /user/hadoop/wordc
```

You should see something like this

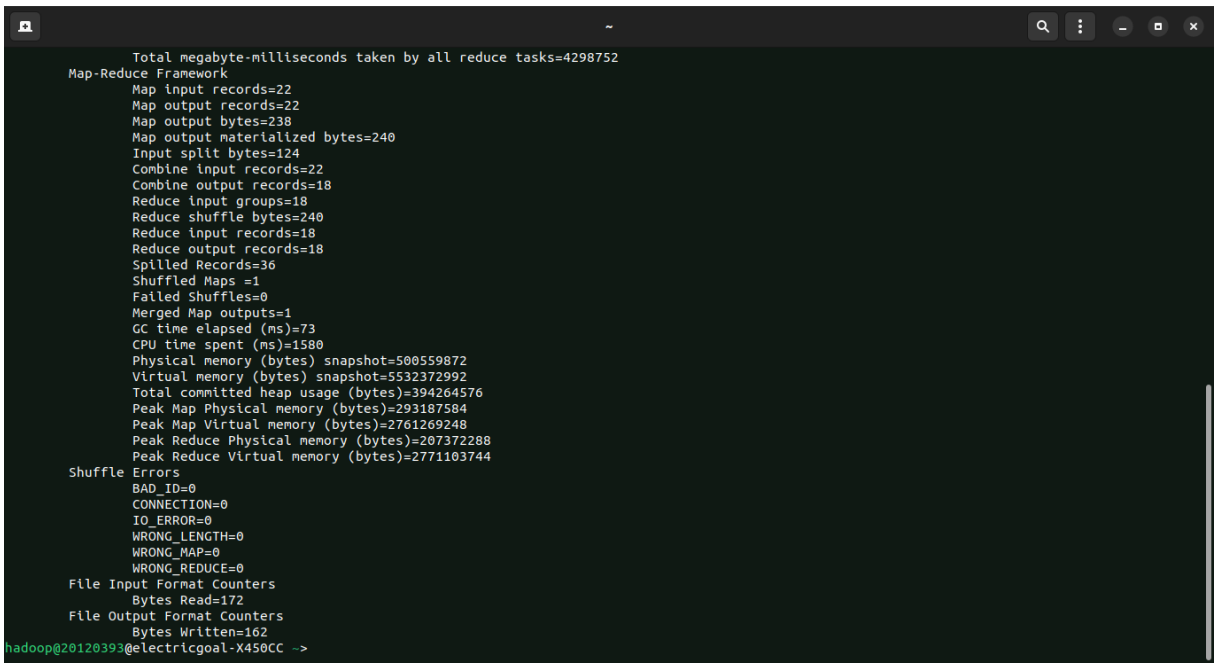

Figure 1.40: Result screen

To see the result, enter this command

```
hadoop fs -cat /<your-favorite-path>/wordcount/output/part-r-00000

# In my case
hadoop fs -cat /user/hadoop/wordcount/output/part-r-00000
```

Compare to the input

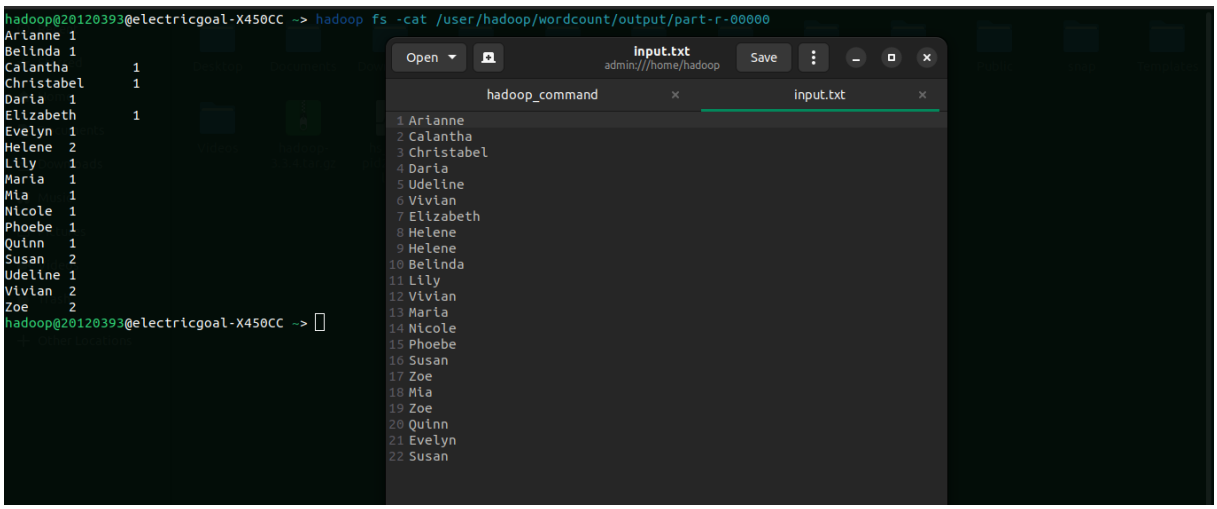

Figure 1.41: Compare to the input

1.7 Bonus

1.7.1 4.1 Extended Word Count: Unhealthy relationships

For more details, open folder `src`

Sample input:

```
faker showmaker
gumayusi deft
keria kellin
canyon oner
zeus canna
chovy faker
canyon peanut
oner peanut
zeus doran
gumayusi peyz
keria delight
deft peyz
delight kellin
chovy showmaker
doran canna
```

Expected output:

```
canna neg
canyon pos
chovy pos
deft eq
delight eq
doran eq
faker eq
gumayusi pos
kellin neg
keria pos
oner eq
peanut neg
peyz neg
showmaker neg
zeus pos
```

The result screen:

Figure 1.42: Result screen

1.7.2 4.2 Setting up Fully Distributed Mode

Incomplete

1.8 References

- Example: WordCount v1.0: https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html#Example:_WordCount_v1.0
- How to Install Apache Hadoop on Ubuntu 22.04: https://www.howtoforge.com/how-to-install-apache-hadoop-on-ubuntu-22-04/
- How to run Word Count example on Hadoop MapReduce (WordCount Tutorial): https://www.youtube.com/watc
- MapReduce Word Count Example using Hadoop and Java: https://www.youtube.com/watch?v=qgBu8Go1SyM
- Create and Execute your First Hadoop MapReduce Project in Eclipse: https://medium.com/data-science-community-srm/create-execute-your-first-hadoop-mapreduce-project-with-eclipse-9ec03105e974
- All of StackOverflow link related:
  - https://stackoverflow.com/questions/11889261/datanode-process-not-running-in-hadoop
  - https://stackoverflow.com/questions/66182686/why-output-key-value-of-mapper-needs-to-be-same-as-that-of-output-key-value-ofco