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.

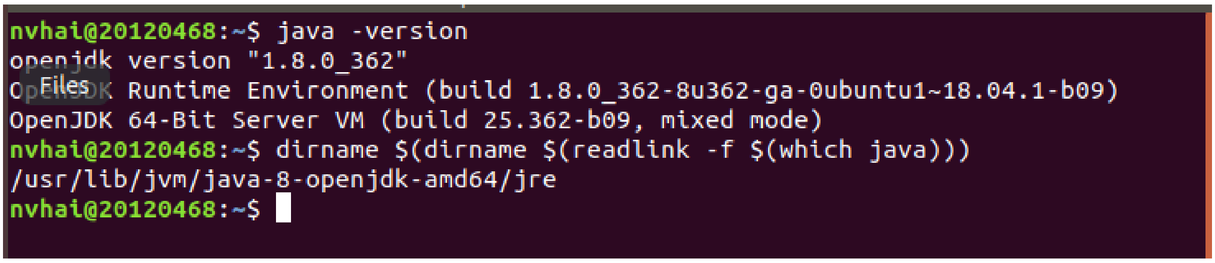
   ```
   sudo apt update && sudo apt install openjdk-8-jdk
   ```

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

   ```
   java -version
   ```

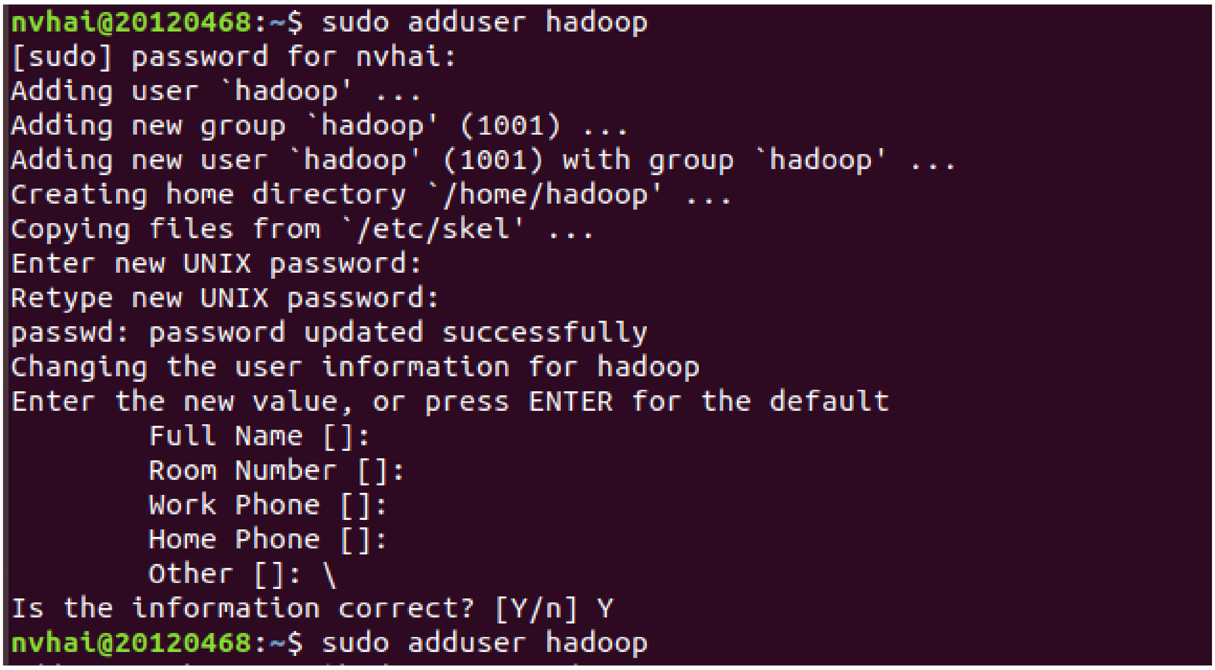
1.4.2 Step 2: Create User for Hadoop and install openSSH

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



```
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



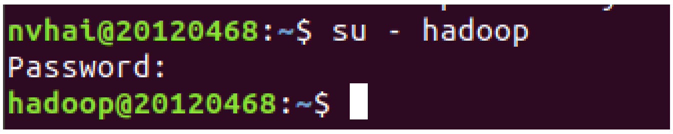
```
nvhai@20120468:~$ sudo adduser hadoop
[sudo] password for nvhai:
Adding user `hadoop' ...
Adding new group `hadoop' (1001) ...
Adding new user `hadoop' (1001) with group `hadoop' ...
Creating home directory `/home/hadoop' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new 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

```
sudo adduser hadoop
```

2. Switch to the newly created hadoop user:

```
su - hadoop
```

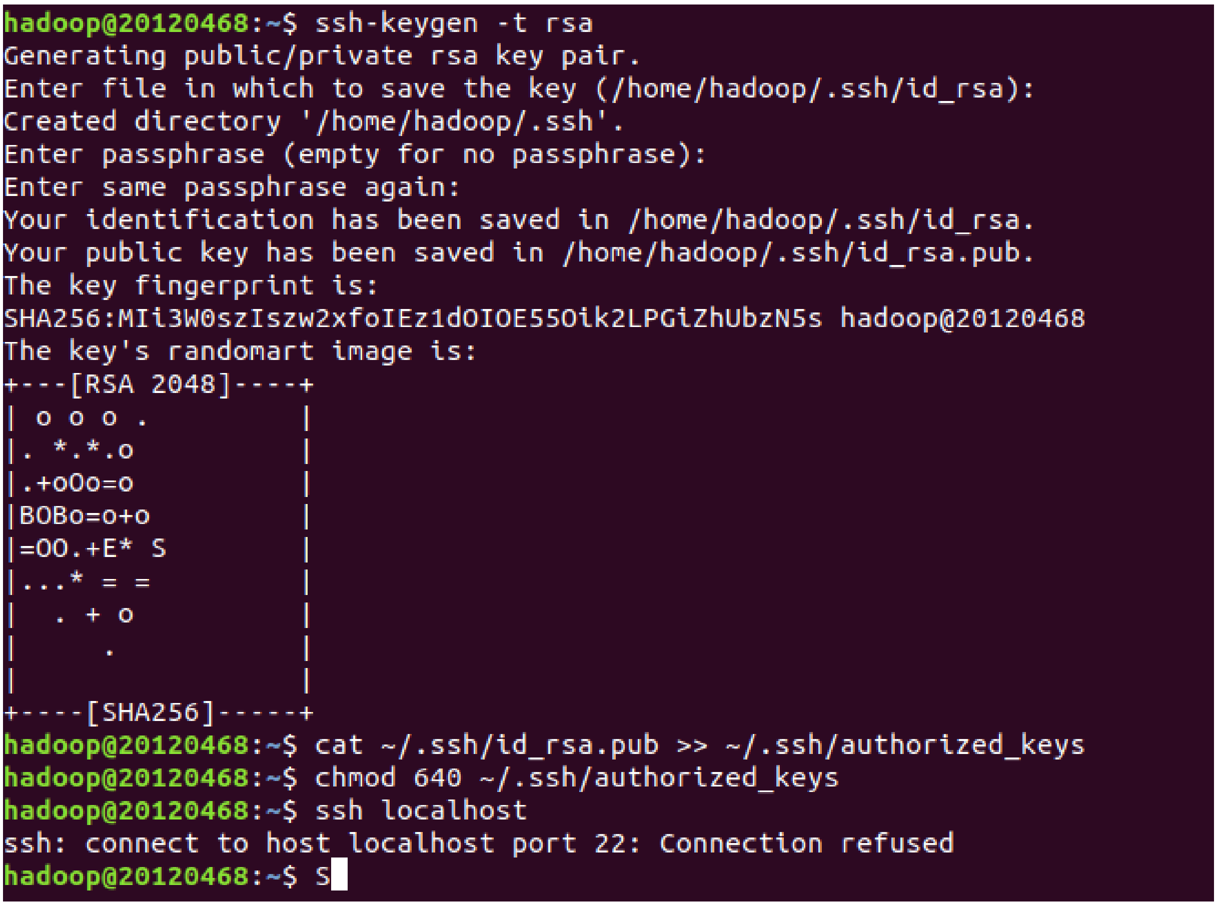


```
nvhai@20120468:~$ su - hadoop
Password:
hadoop@20120468:~$
```

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
```



```
hadoop@20120468:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hadoop/.ssh/id_rsa):
Created directory '/home/hadoop/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/hadoop/.ssh/id_rsa.
Your public key has been saved in /home/hadoop/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:MIi3W0szIszw2xfoIEz1dOIOE55Oik2LPGiZhUbzN5s hadoop@20120468
The key's randomart image is:
+---[RSA 2048]----+
| o o o .         |
|. *.*.o          |
|.+oOo=o          |
|BOBo=o+o         |
|=OO.+E* S        |
|...* = =         |
|  . + o          |
|     .           |
|                 |
+----[SHA256]-----+
hadoop@20120468:~$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
hadoop@20120468:~$ chmod 640 ~/.ssh/authorized_keys
hadoop@20120468:~$ ssh localhost
ssh: connect to host localhost port 22: Connection refused
hadoop@20120468:~$ S
```

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

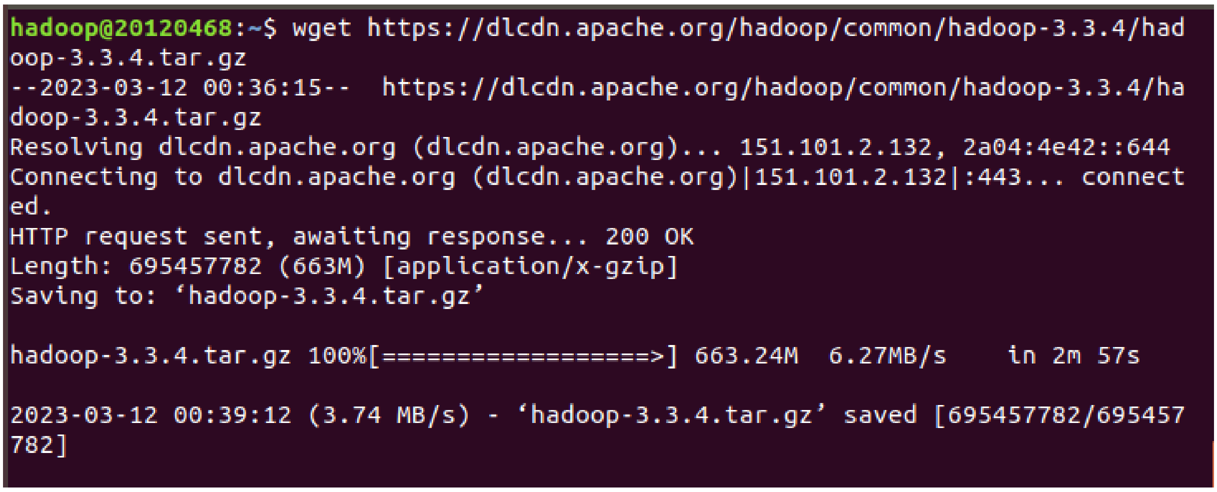
   ```
   ssh localhost
   ```

   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
   ```



```
hadoop@20120468:~$ wget https://dlcdn.apache.org/hadoop/common/hadoop-3.3.4/had
oop-3.3.4.tar.gz
--2023-03-12 00:36:15--  https://dlcdn.apache.org/hadoop/common/hadoop-3.3.4/ha
doop-3.3.4.tar.gz
Resolving dlcdn.apache.org (dlcdn.apache.org)... 151.101.2.132, 2a04:4e42::644
Connecting to dlcdn.apache.org (dlcdn.apache.org)|151.101.2.132|:443... connect
ed.
HTTP request sent, awaiting response... 200 OK
Length: 695457782 (663M) [application/x-gzip]
Saving to: ‘hadoop-3.3.4.tar.gz’

hadoop-3.3.4.tar.gz 100%[==================>] 663.24M  6.27MB/s    in 2m 57s

2023-03-12 00:39:12 (3.74 MB/s) - ‘hadoop-3.3.4.tar.gz’ saved [695457782/695457
782]
```

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
   ```

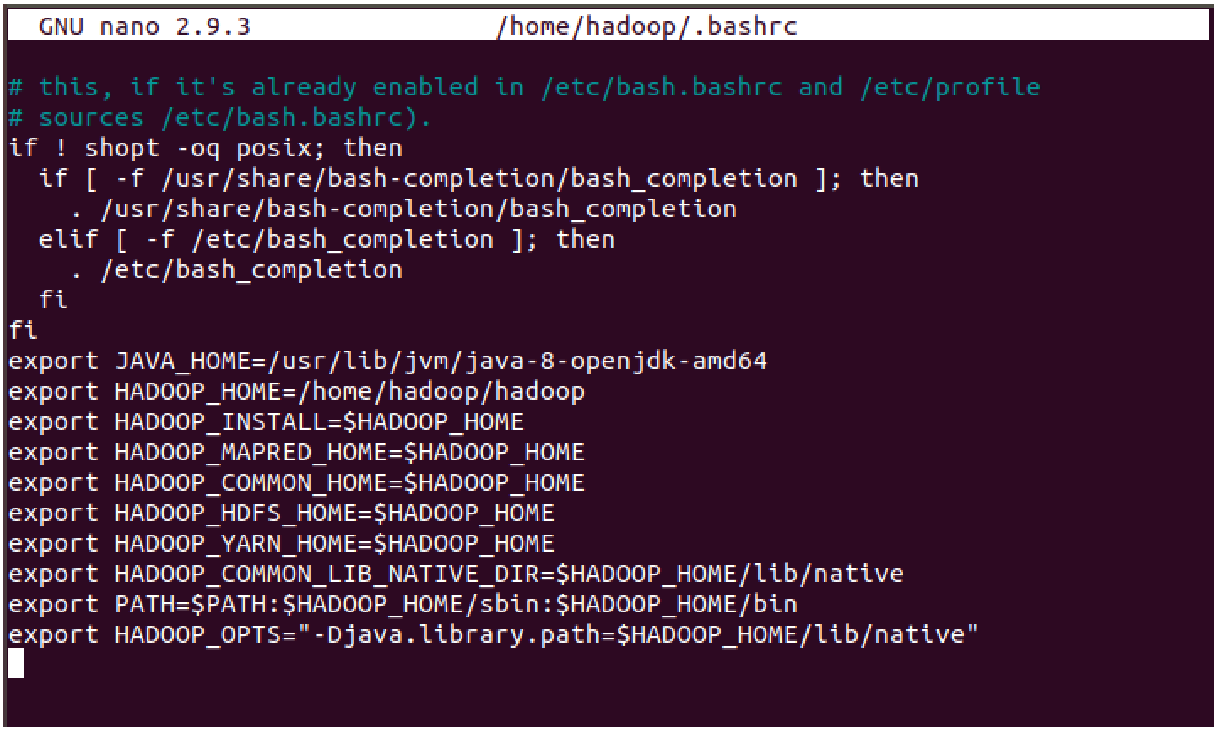
3. Rename the extracted folder to remove version information. This is an optional step, but if you don't want to rename, then adjust the remaining configuration paths.

```
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.

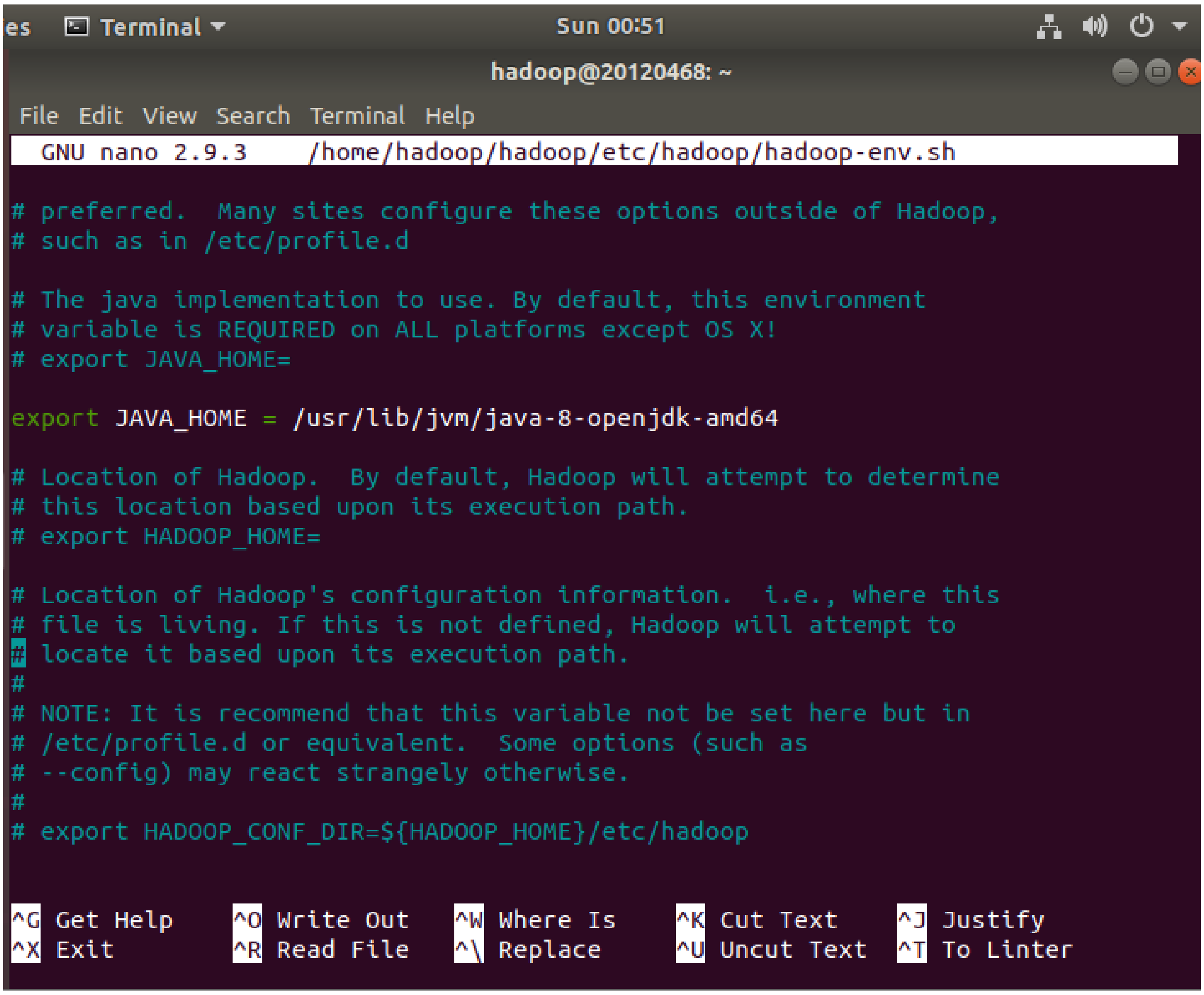
5. Load the above configuration in the current environment

   ```
   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:



```
  GNU nano 2.9.3     /home/hadoop/hadoop/etc/hadoop/hadoop-env.sh

# preferred.  Many sites configure these options outside of Hadoop,
# such as in /etc/profile.d

# The java implementation to use. By default, this environment
# variable is REQUIRED on ALL platforms except OS X!
# export JAVA_HOME=

export JAVA_HOME = /usr/lib/jvm/java-8-openjdk-amd64

# Location of Hadoop.  By default, Hadoop will attempt to determine
# this location based upon its execution path.
# export HADOOP_HOME=

# Location of Hadoop's configuration information.  i.e., where this
# file is living. If this is not defined, Hadoop will attempt to
# locate it based upon its execution path.
#
# NOTE: It is recommend that this variable not be set here but in
# /etc/profile.d or equivalent.  Some options (such as
# --config) may react strangely otherwise.
#
# export HADOOP_CONF_DIR=${HADOOP_HOME}/etc/hadoop


^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Linter
```

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:

   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:

   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:

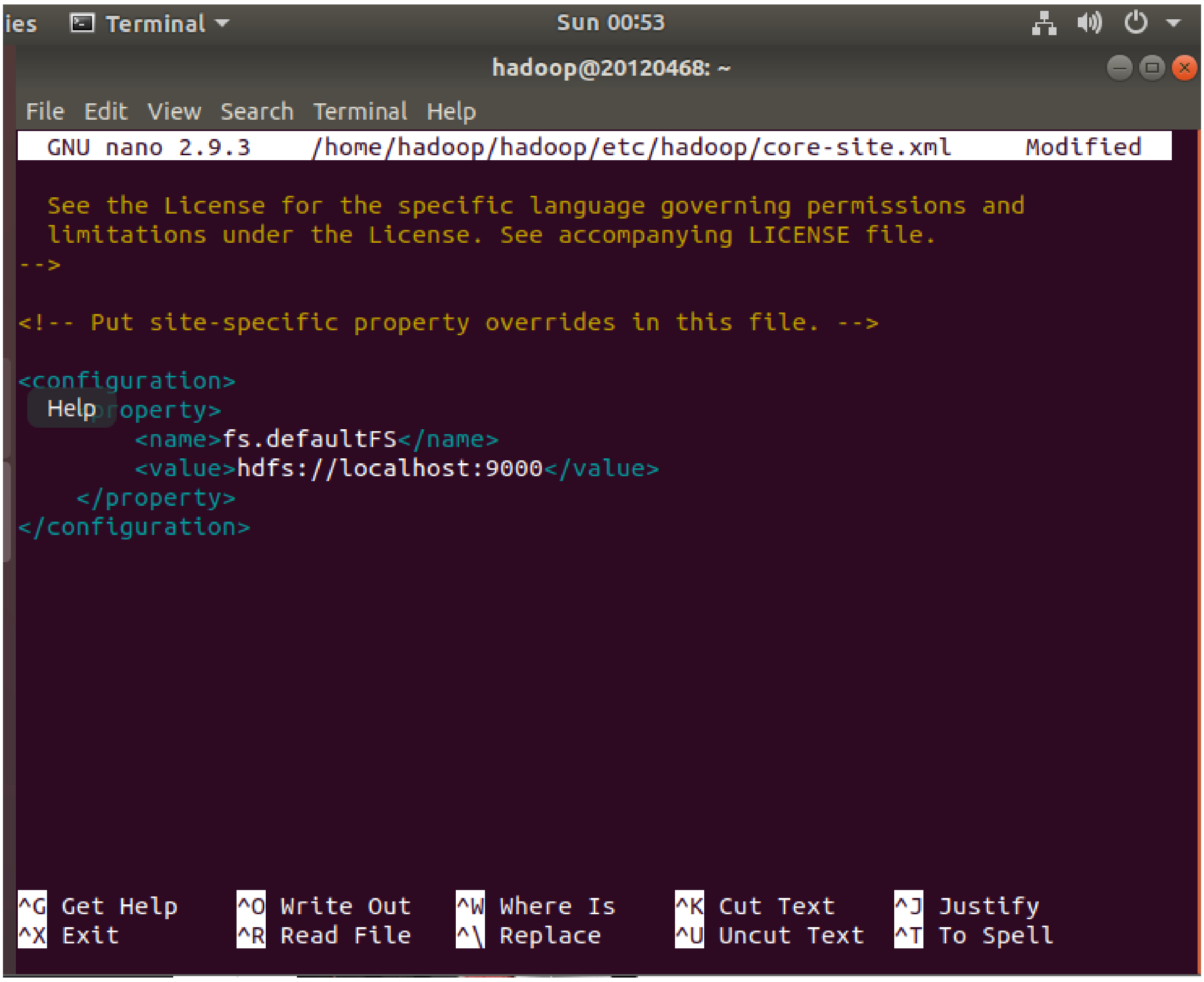
   Save and close the file.

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

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

   Make the following changes:

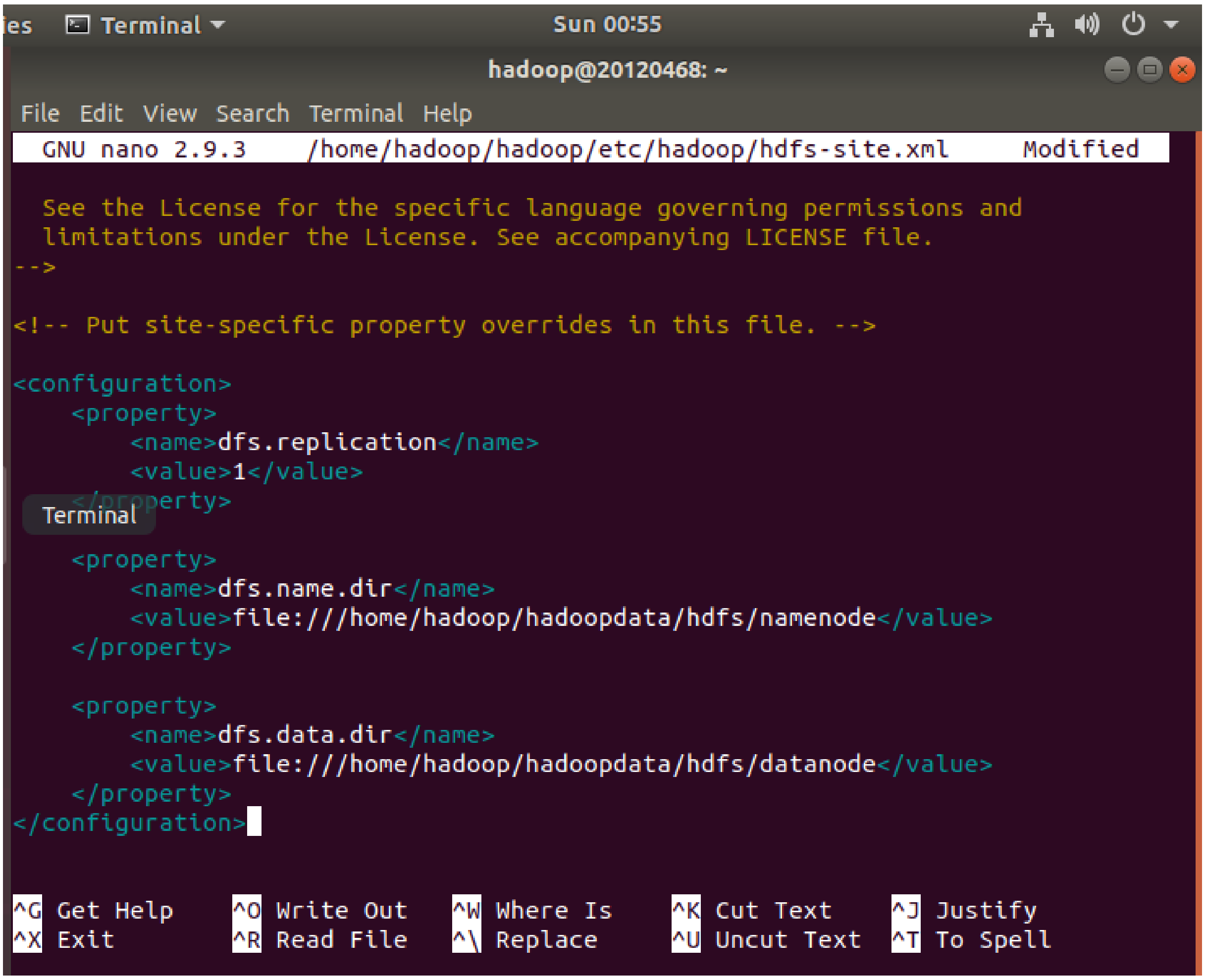
   Save and close the file.



ies
Terminal
Sun 00:53
hadoop@20120468: ~
File Edit View Search Terminal Help
GNU nano 2.9.3 /home/hadoop/hadoop/etc/hadoop/core-site.xml Modified
Help

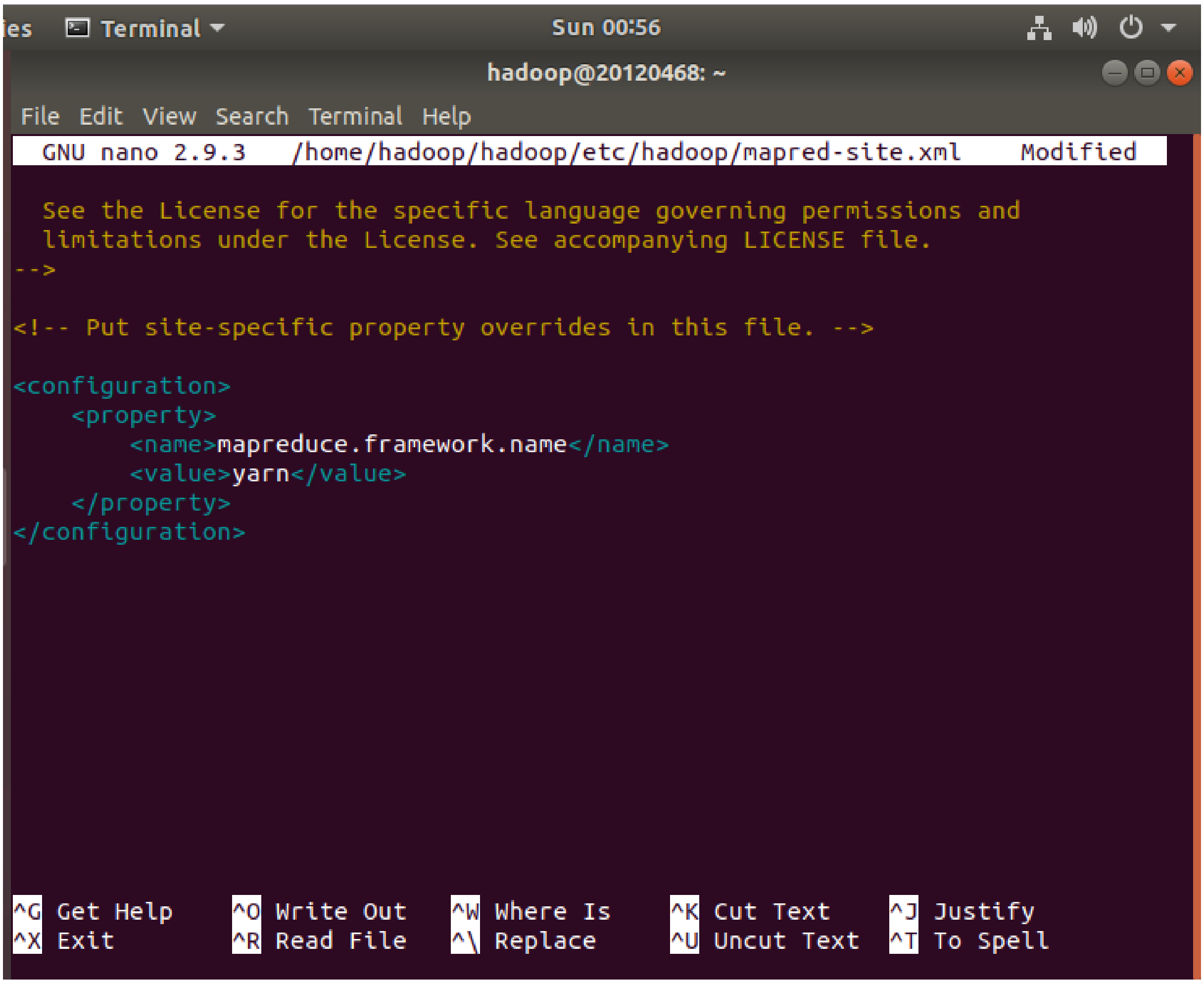
```
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

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

<configuration>
    <property>
        <name>fs.defaultFS</name>
        <value>hdfs://localhost:9000</value>
    </property>
</configuration>
```

Figure 1.8: setup-core-site



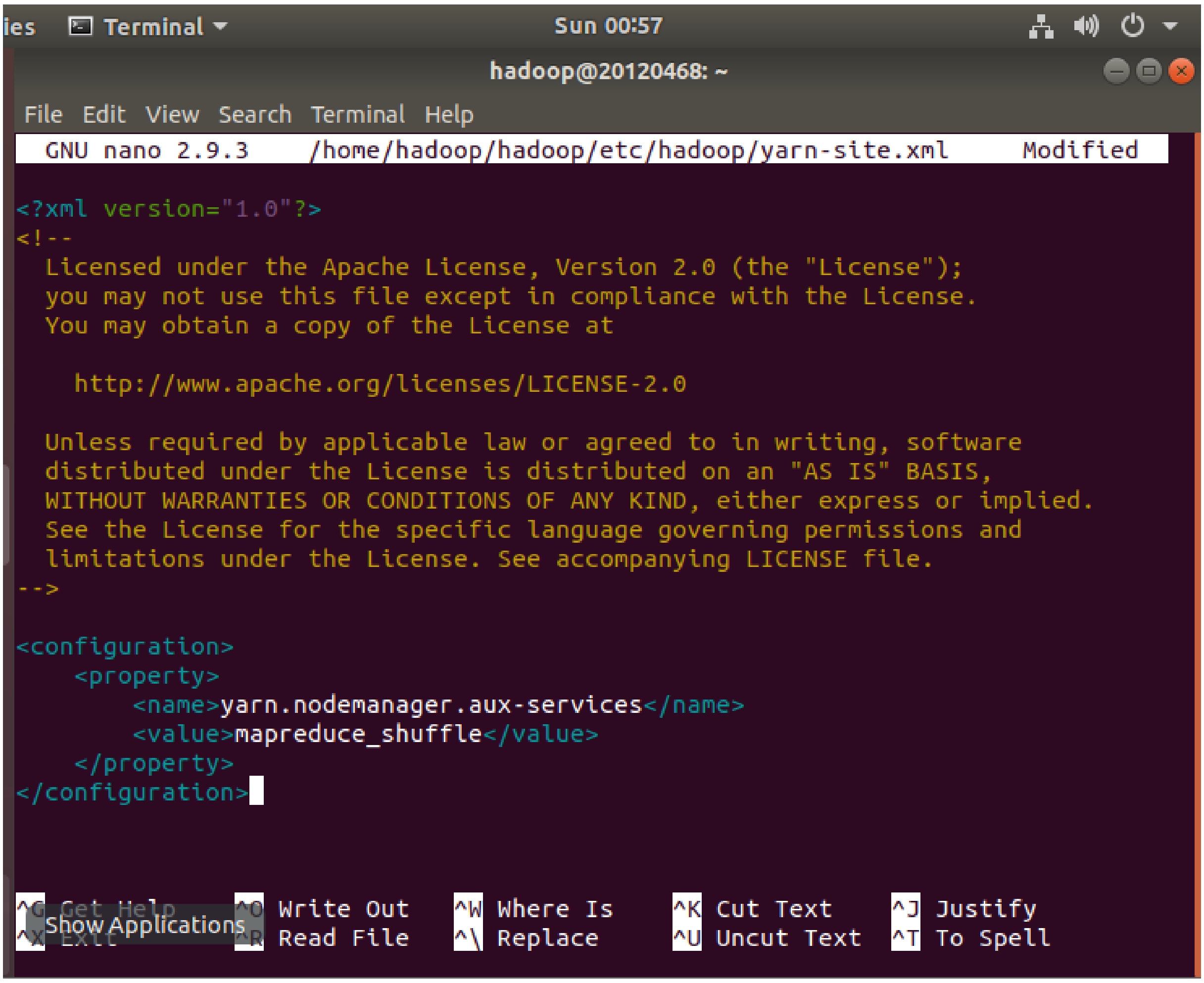
```
  GNU nano 2.9.3     /home/hadoop/hadoop/etc/hadoop/hdfs-site.xml      Modified

  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>dfs.replication</name>
        <value>1</value>
    </property>

    <property>
        <name>dfs.name.dir</name>
        <value>file:///home/hadoop/hadoopdata/hdfs/namenode</value>
    </property>

    <property>
        <name>dfs.data.dir</name>
        <value>file:///home/hadoop/hadoopdata/hdfs/datanode</value>
    </property>
</configuration>

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Spell
```

Figure 1.9: setup-hdfs-site



```
  GNU nano 2.9.3    /home/hadoop/hadoop/etc/hadoop/mapred-site.xml     Modified

  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>
</configuration>
```

Figure 1.10: setup-mapred-site



```
  GNU nano 2.9.3     /home/hadoop/hadoop/etc/hadoop/yarn-site.xml       Modified

<?xml version="1.0"?>
<!--
  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.
-->

<configuration>
    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    </property>
</configuration>

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Spell
```

Figure 1.11: setup-yarn-site

1.4.5 Step 5: Start Hadoop Cluster

Then start the Hadoop cluster with the following command

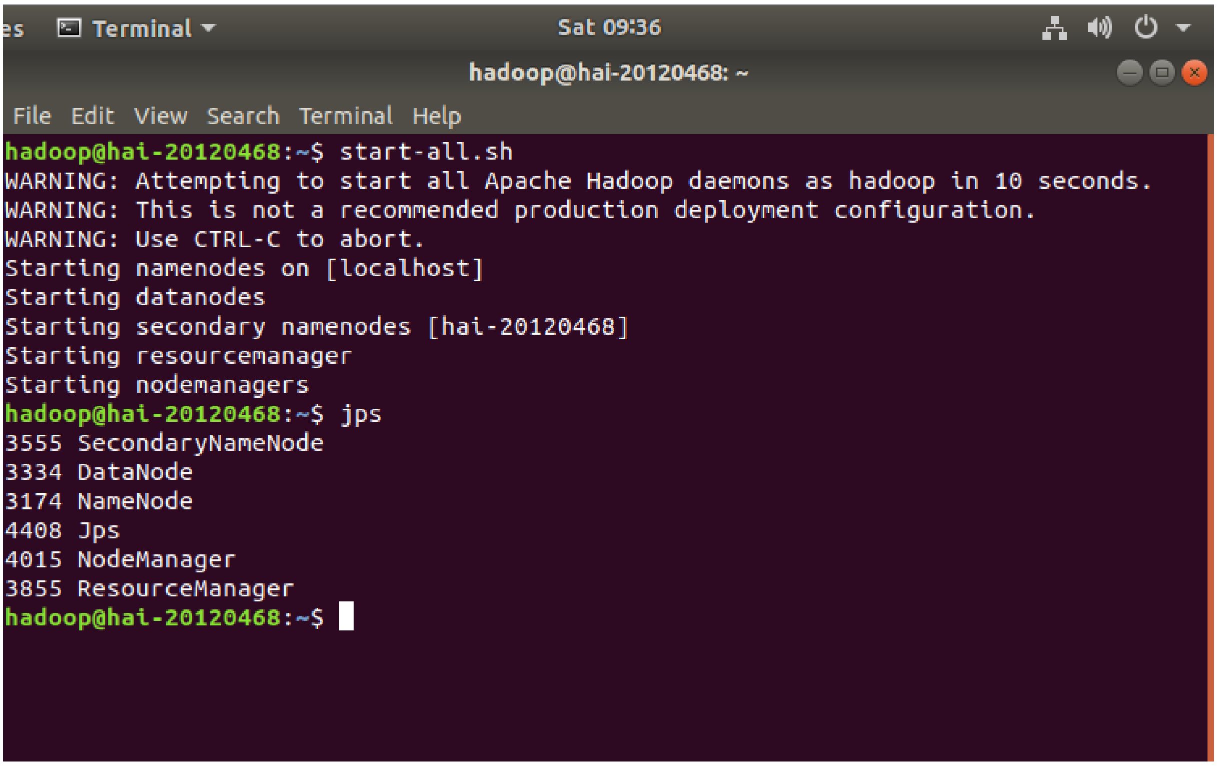
```
start-all.sh
```

Check jps

```
jps
```

Completed screenshots of the members:

20120468 - Nguyen Van Hai



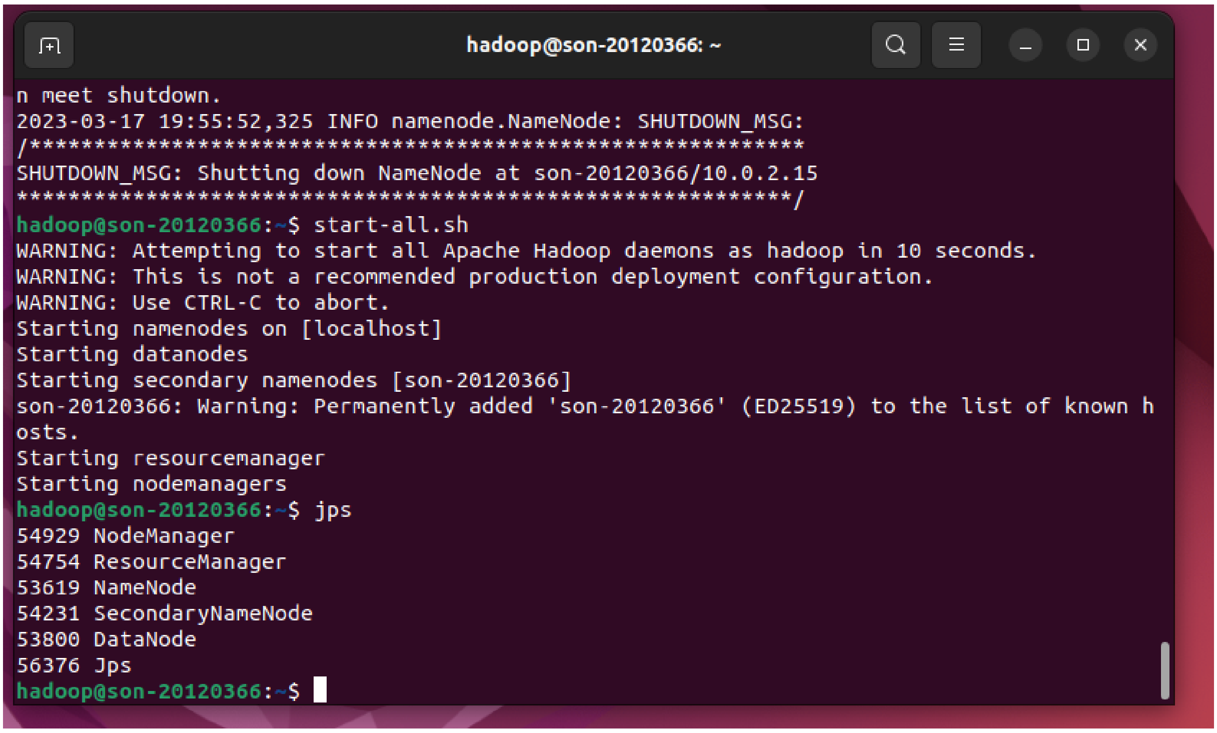
```
hadoop@hai-20120468:~$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as hadoop in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [hai-20120468]
Starting resourcemanager
Starting nodemanagers
hadoop@hai-20120468:~$ jps
3555 SecondaryNameNode
3334 DataNode
3174 NameNode
4408 Jps
4015 NodeManager
3855 ResourceManager
hadoop@hai-20120468:~$
```

Figure 1.12: 20120468 done

20120366 - Pham Phu Hoang Son

20120391 - Ha Xuan Truong

20120393 - Huynh Minh Tu



```
n meet shutdown.
2023-03-17 19:55:52,325 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at son-20120366/10.0.2.15
************************************************************/
hadoop@son-20120366:~$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as hadoop in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [son-20120366]
son-20120366: Warning: Permanently added 'son-20120366' (ED25519) to the list of known h
osts.
Starting resourcemanager
Starting nodemanagers
hadoop@son-20120366:~$ jps
54929 NodeManager
54754 ResourceManager
53619 NameNode
54231 SecondaryNameNode
53800 DataNode
56376 Jps
hadoop@son-20120366:~$
```

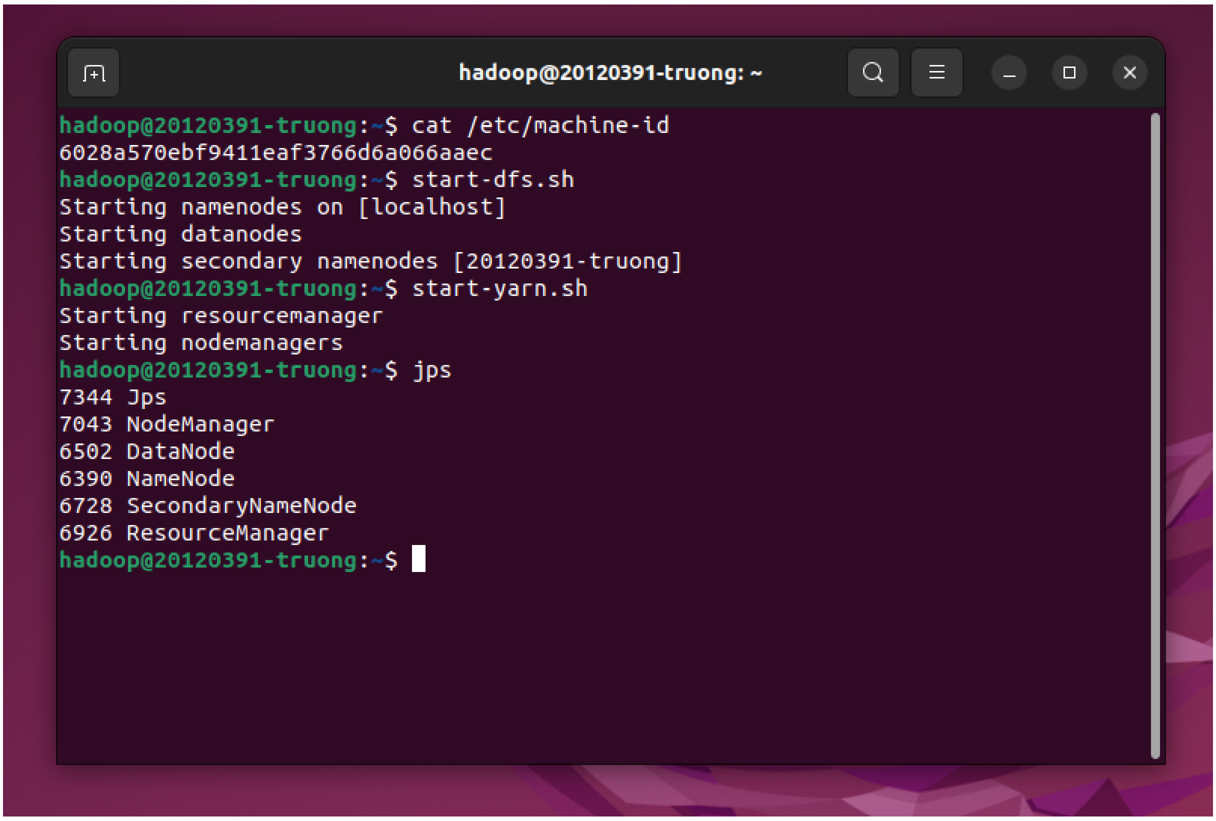
Figure 1.13: 20120366 done

1.5 Introduction to MapReduce

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

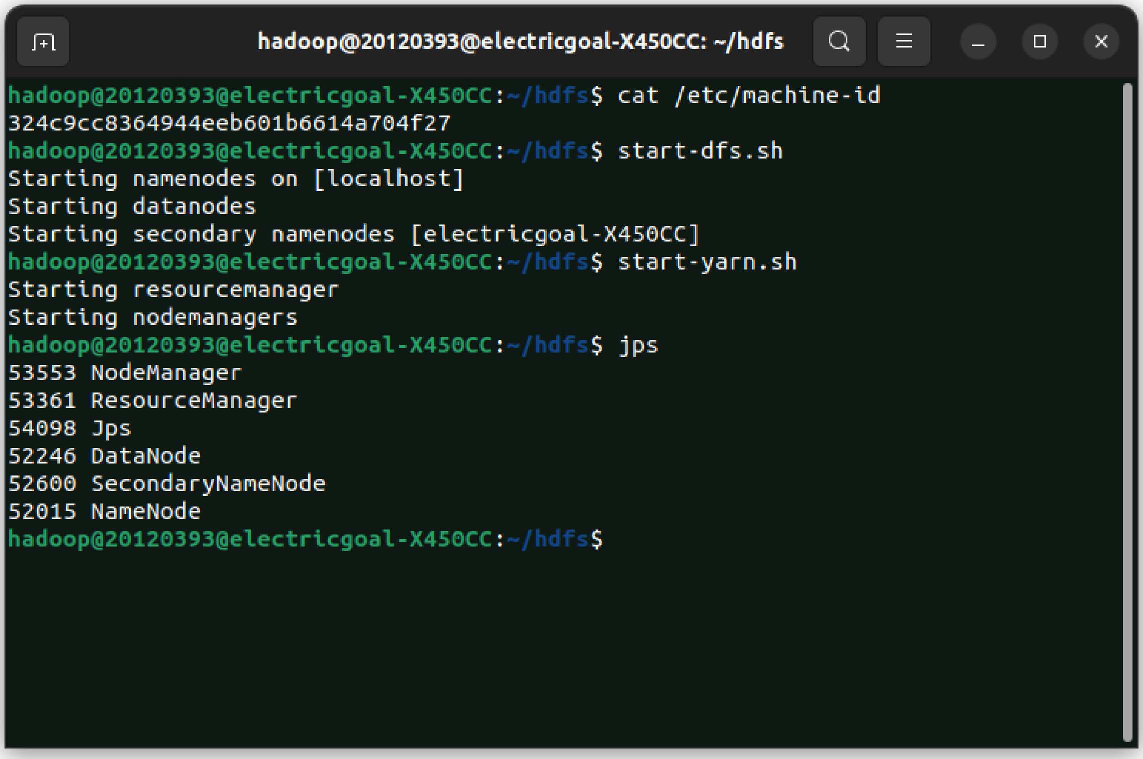
Answer:

- 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.



```
hadoop@20120391-truong:~$ cat /etc/machine-id
6028a570ebf9411eaf3766d6a066aaec
hadoop@20120391-truong:~$ start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [20120391-truong]
hadoop@20120391-truong:~$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
hadoop@20120391-truong:~$ jps
7344 Jps
7043 NodeManager
6502 DataNode
6390 NameNode
6728 SecondaryNameNode
6926 ResourceManager
hadoop@20120391-truong:~$
```

Figure 1.14: 20120391 done



```
hadoop@20120393@electricgoal-X450CC:~/hdfs$ cat /etc/machine-id
324c9cc8364944eeb601b6614a704f27
hadoop@20120393@electricgoal-X450CC:~/hdfs$ start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [electricgoal-X450CC]
hadoop@20120393@electricgoal-X450CC:~/hdfs$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
hadoop@20120393@electricgoal-X450CC:~/hdfs$ jps
53553 NodeManager
53361 ResourceManager
54098 Jps
52246 DataNode
52600 SecondaryNameNode
52015 NameNode
hadoop@20120393@electricgoal-X450CC:~/hdfs$
```

Figure 1.15: 20120393 done

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

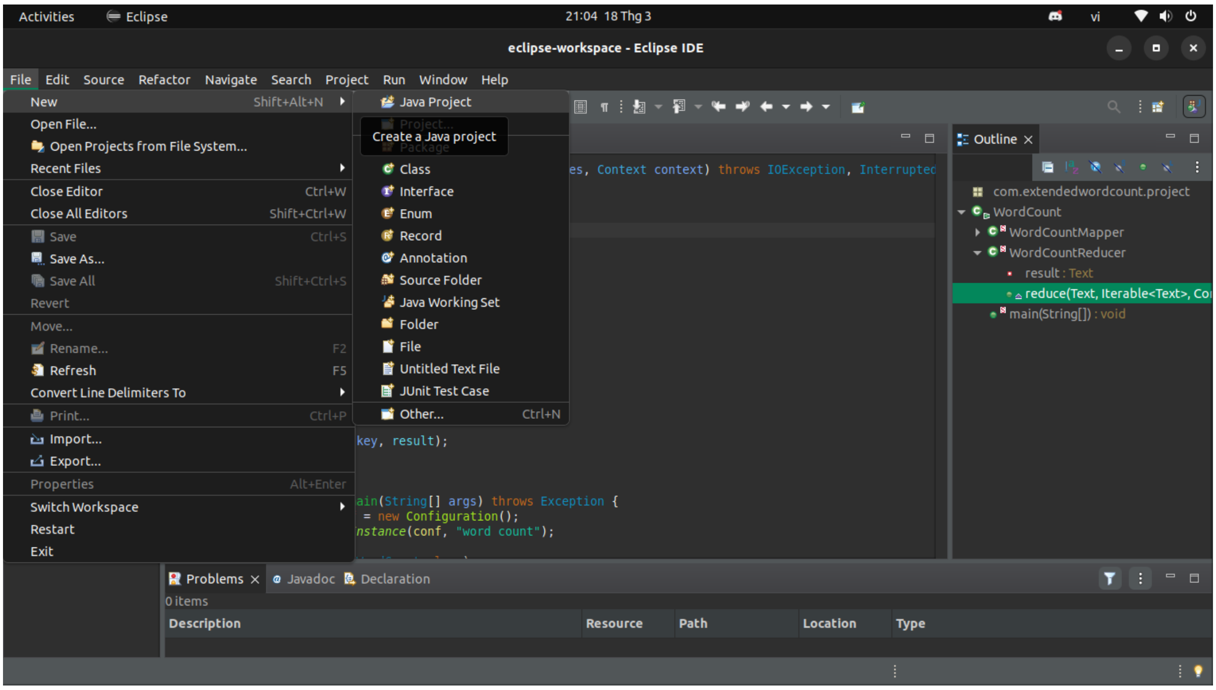
Use Eclipse IDE to run MapReduce on Ubuntu

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**



```
es, Context context) throws IOException, Interruptec
key, result);
ain(String[] args) throws Exception {
 = new Configuration();
nstance(conf, "word count");
```

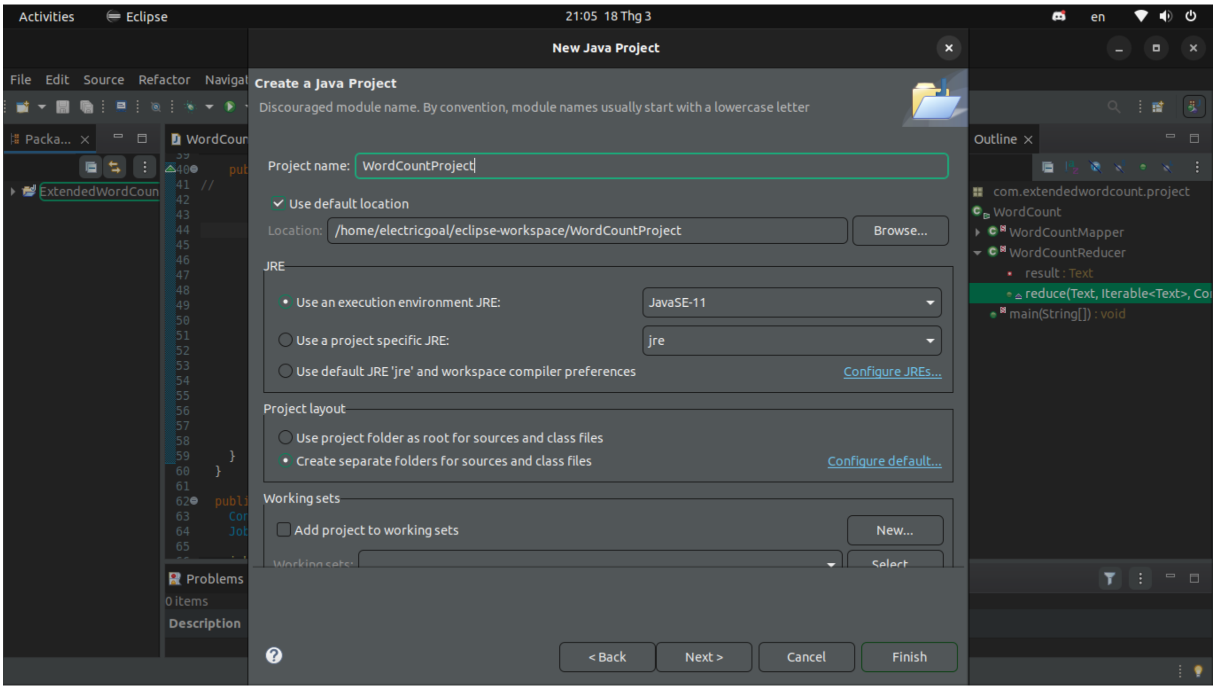
Figure 1.16: Run MapReduce

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

Click on **Finish** button

Result looks like this

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



Create a Java Project

Discouraged module name. By convention, module names usually start with a lowercase letter

Project name: WordCountProject

☑ Use default location

Location: /home/electricgoal/eclipse-workspace/WordCountProject

JRE

● Use an execution environment JRE: JavaSE-11

○ Use a project specific JRE: jre

○ Use default JRE 'jre' and workspace compiler preferences

Project layout

○ Use project folder as root for sources and class files

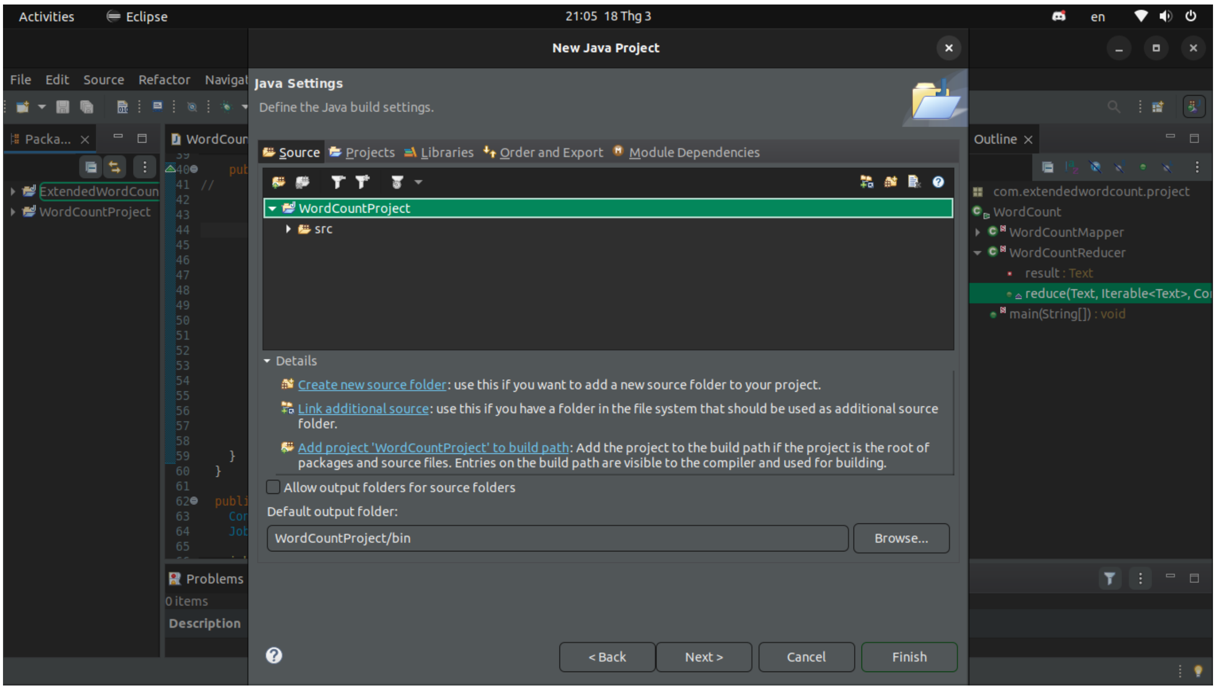
● Create separate folders for sources and class files

Working sets

□ Add project to working sets

Working sets:

Figure 1.17: Run MapReduce



Java Settings

Define the Java build settings.

WordCountProject
src

Details

Create new source folder: use this if you want to add a new source folder to your project.

Link additional source: use this if you have a folder in the file system that should be used as additional source folder.

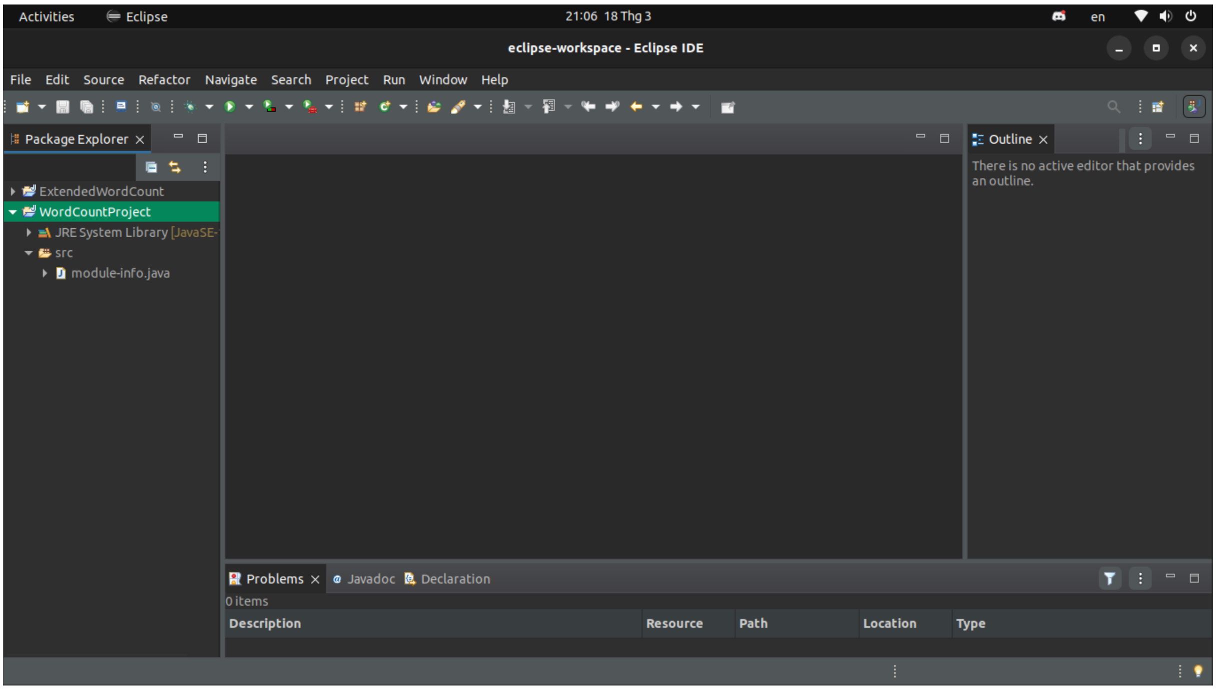
Add project 'WordCountProject' to build path: Add the project to the build path if the project is the root of packages and source files. Entries on the build path are visible to the compiler and used for building.

□ Allow output folders for source folders

Default output folder:

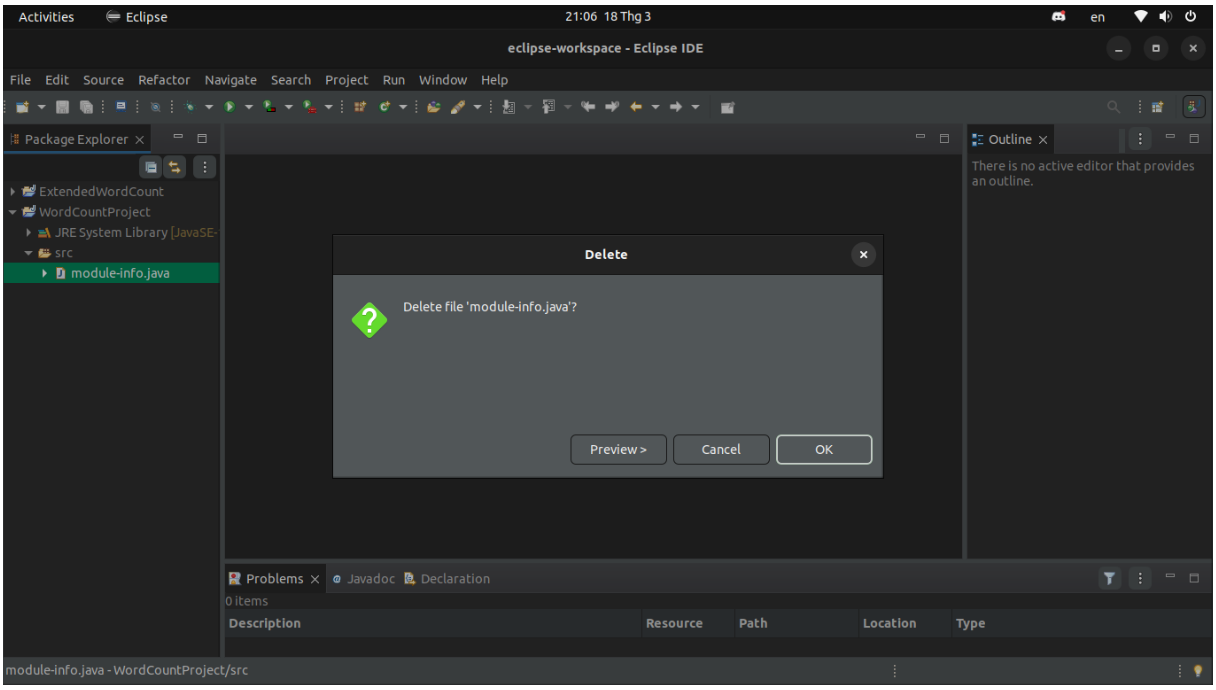
WordCountProject/bin

Figure 1.18: Run MapReduce



There is no active editor that provides an outline.

Figure 1.19: Run MapReduce



There is no active editor that provides an outline.

Delete file 'module-info.java'?

Figure 1.20: Run MapReduce

1.6.4 Step 3: Create Java package

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

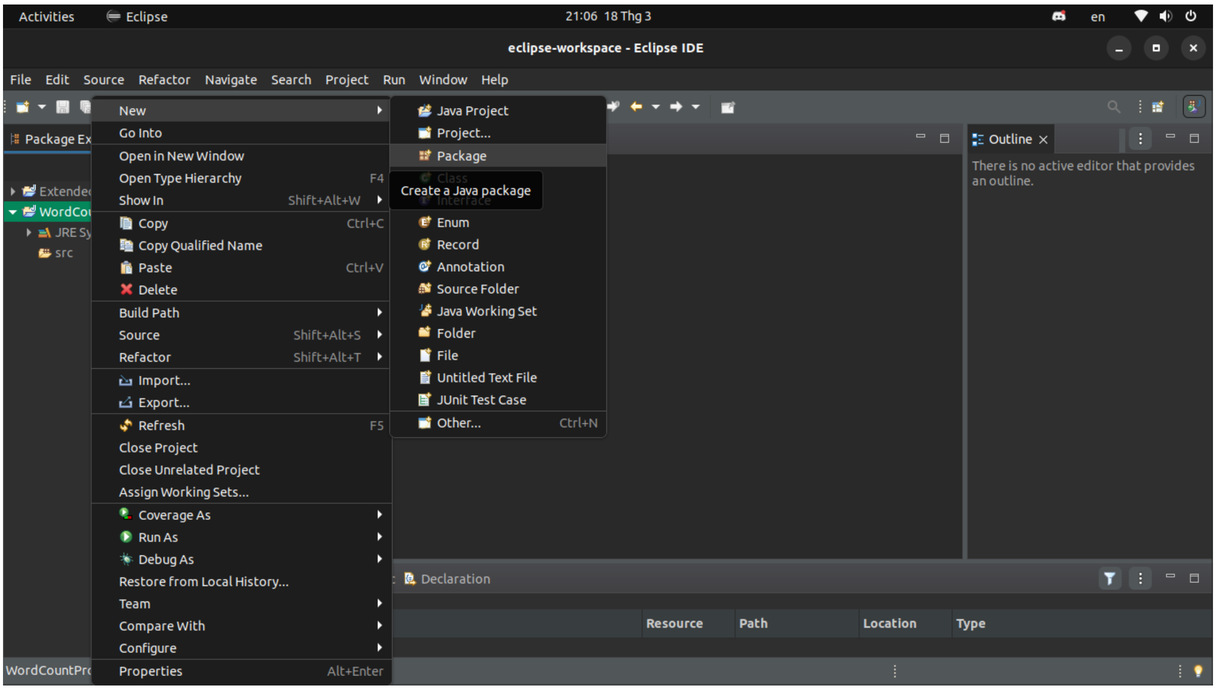


Figure 1.21: Run MapReduce

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

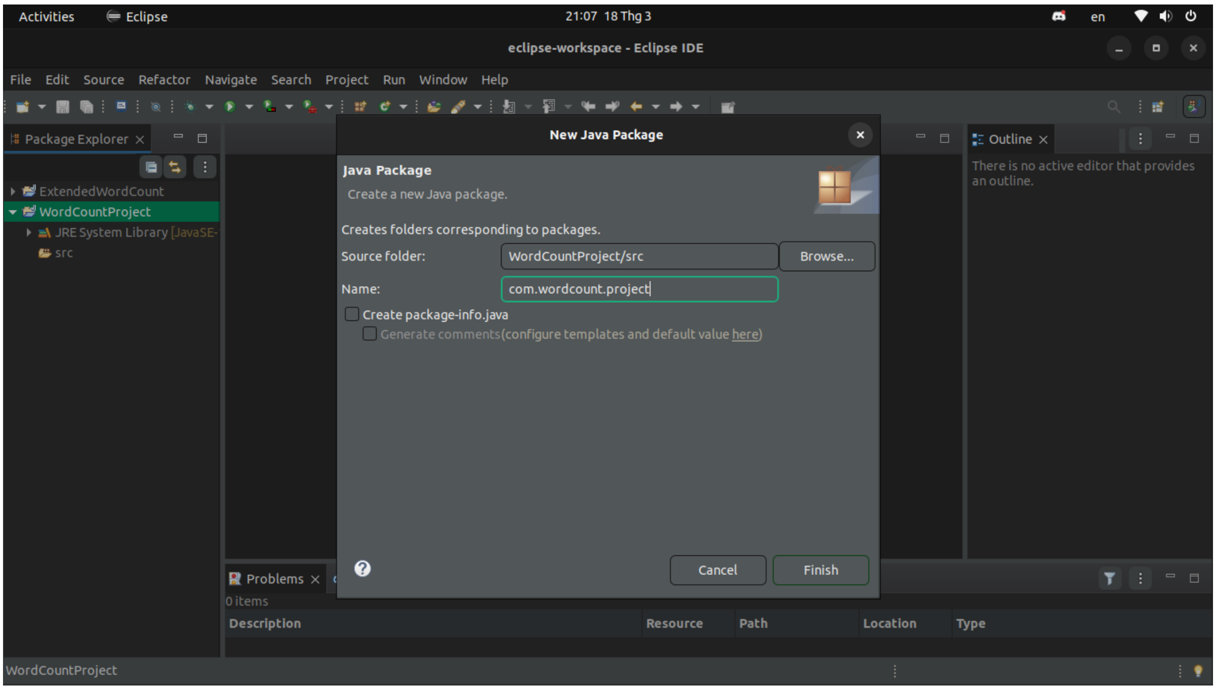
1.6.5 Step 4: Create Java class

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

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

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

You should see many errors


Activities
Eclipse
21:07 18 Thg 3
en
eclipse-workspace - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
ExtendedWordCount
WordCountProject
JRE System Library [JavaSE-
src
Outline
There is no active editor that provides an outline.
Problems
0 items
Description
Resource
Path
Location
Type
WordCountProject

New Java Package

Java Package

Create a new Java package.

Creates folders corresponding to packages.

Source folder: WordCountProject/src

Name: com.wordcount.project

□ Create package-info.java

□ Generate comments(configure templates and default value here)

Figure 1.22: Run MapReduce

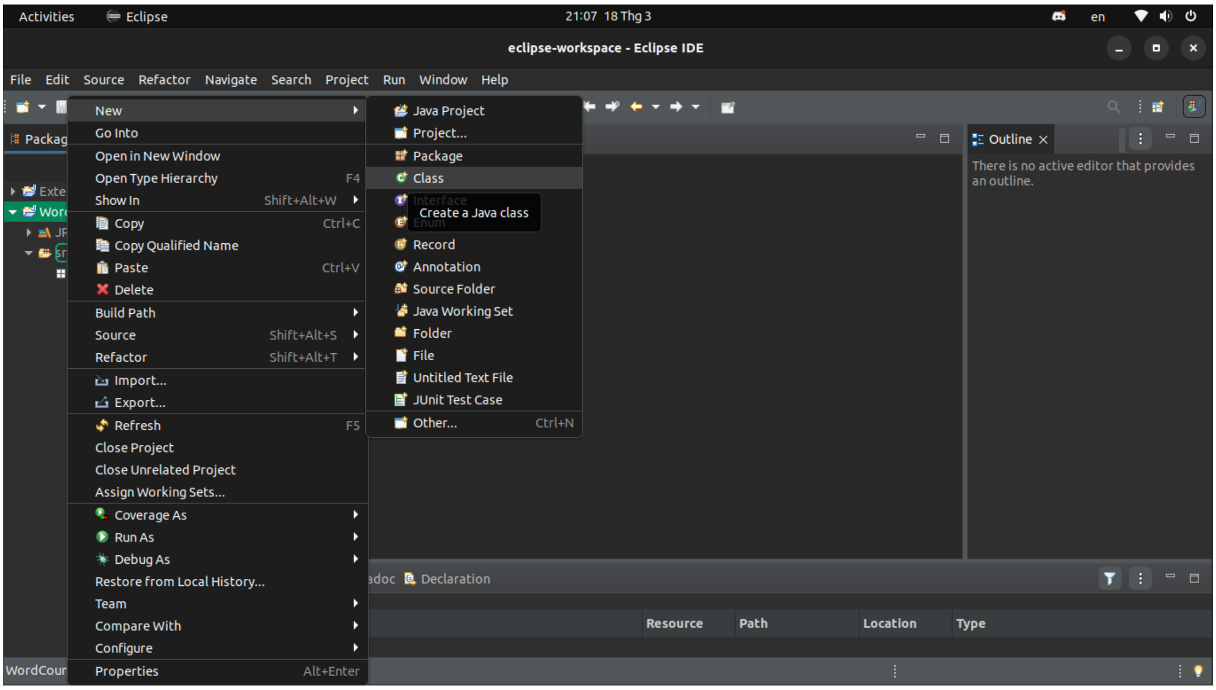
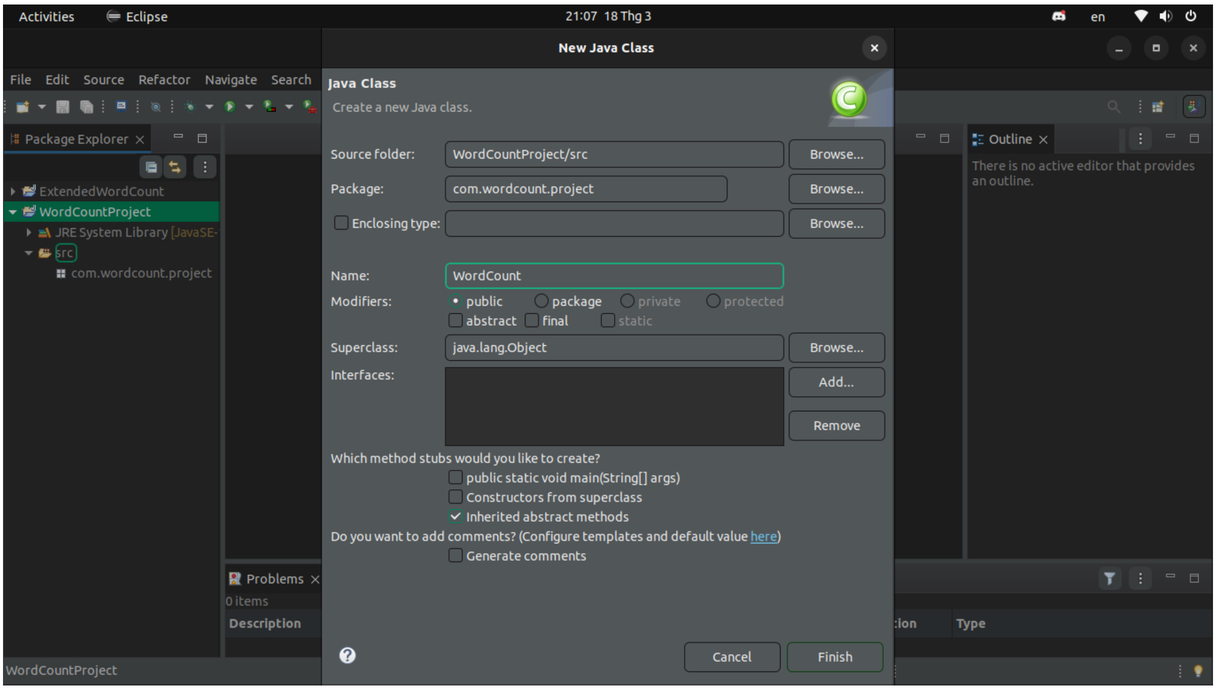


Figure 1.23: Run MapReduce



Java Class

Create a new Java class.

Source folder: WordCountProject/src [Browse...]

Package: com.wordcount.project [Browse...]

□ Enclosing type: [Browse...]

Name: WordCount

Modifiers: ● public ○ package ○ private ○ protected

□ abstract □ final □ static

Superclass: java.lang.Object [Browse...]

Interfaces: [Add...] [Remove]

Which method stubs would you like to create?

□ public static void main(String[] args)

□ Constructors from superclass

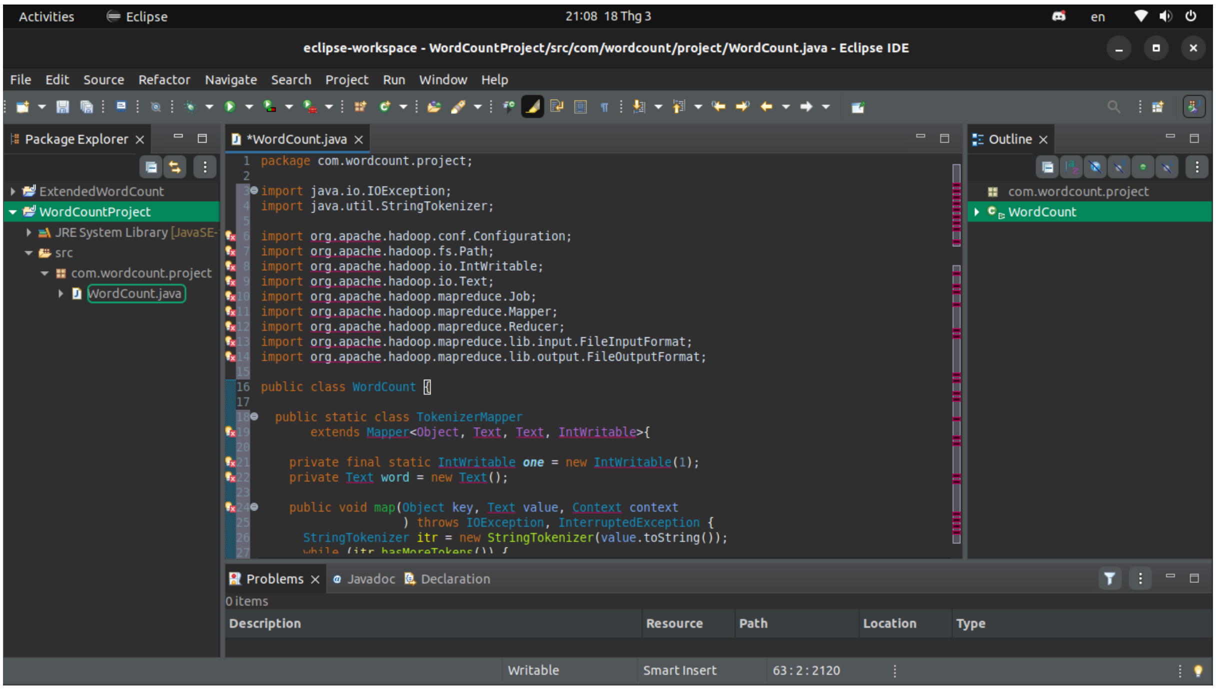
☑ Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)

□ Generate comments

[Cancel] [Finish]

Figure 1.24: Run MapReduce

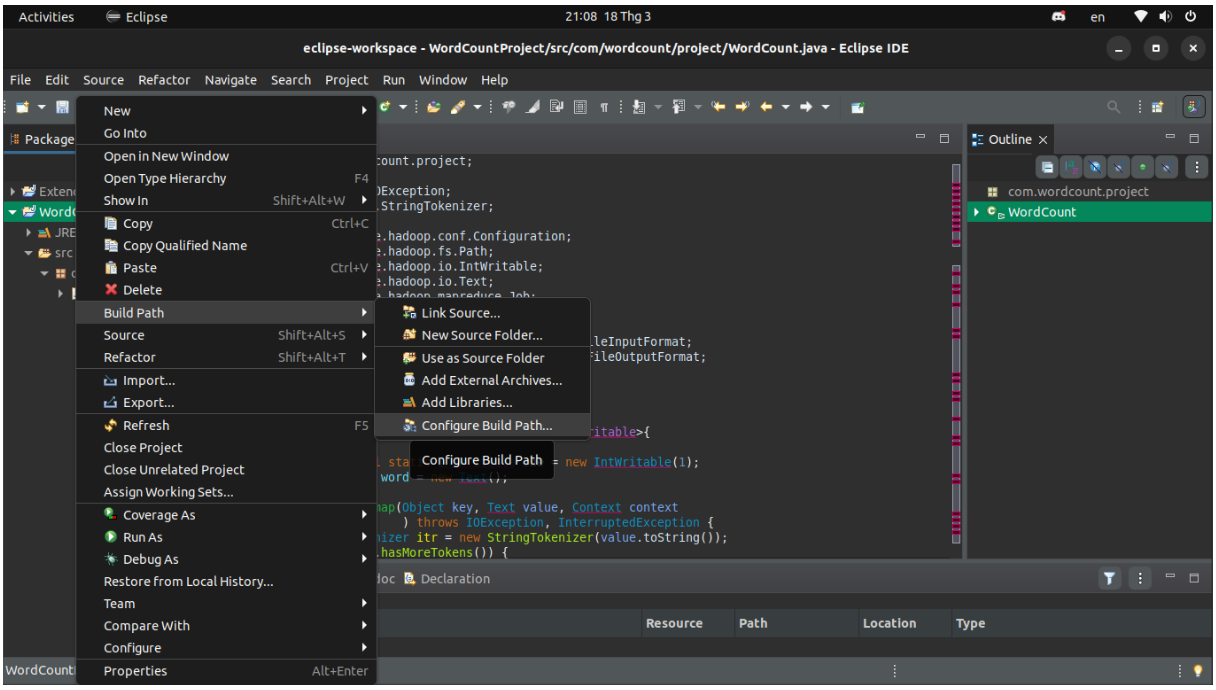


```
package com.wordcount.project;

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()) {
```

Figure 1.25: Run MapReduce

1.6.7 Step 6: Configure build path for the project

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



```
count.project;

)Exception;
.StringTokenizer;

e.hadoop.conf.Configuration;
e.hadoop.fs.Path;
e.hadoop.io.IntWritable;
e.hadoop.io.Text;
e.hadoop.mapreduce.Job;

leInputFormat;
FileOutputFormat;

itable>{

 sta = new IntWritable(1);
 word = new Text();

map(Object key, Text value, Context context
    ) throws IOException, InterruptedException {
nizer itr = new StringTokenizer(value.toString());
.hasMoreTokens()) {
```

Figure 1.26: Run MapReduce

Click on the **Libraries** tab

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

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

Click on the button **Apply and Close**

After that, the errors should disappear

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**

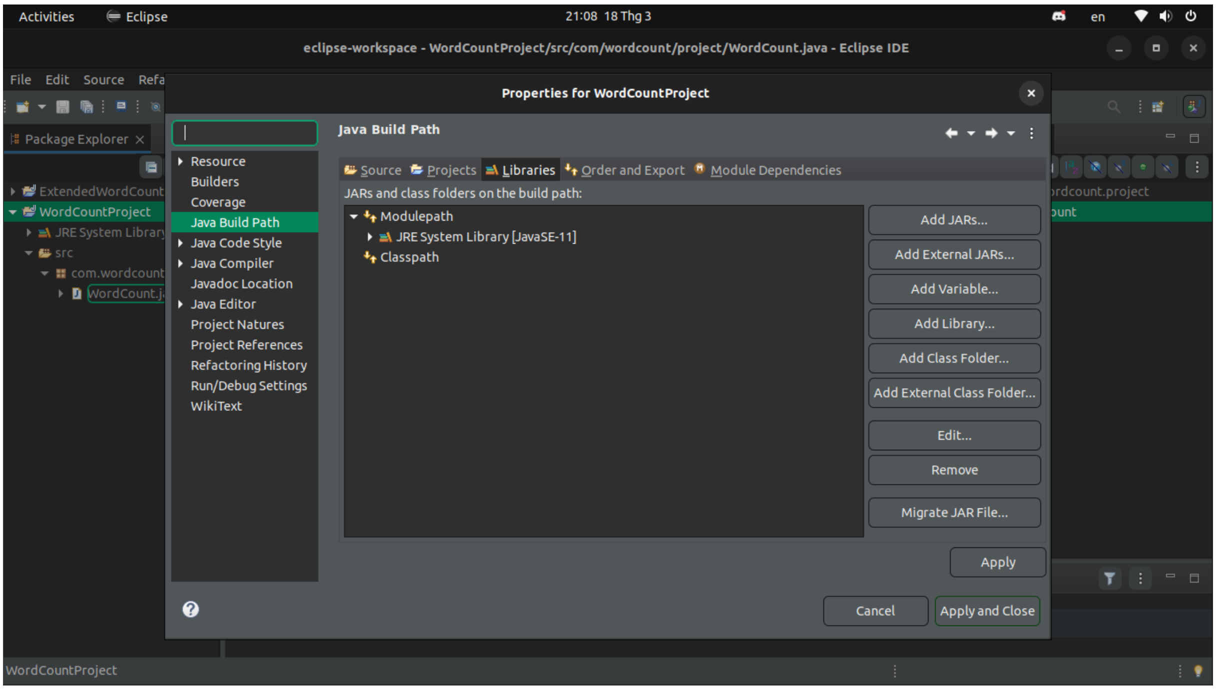


Figure 1.27: Run MapReduce

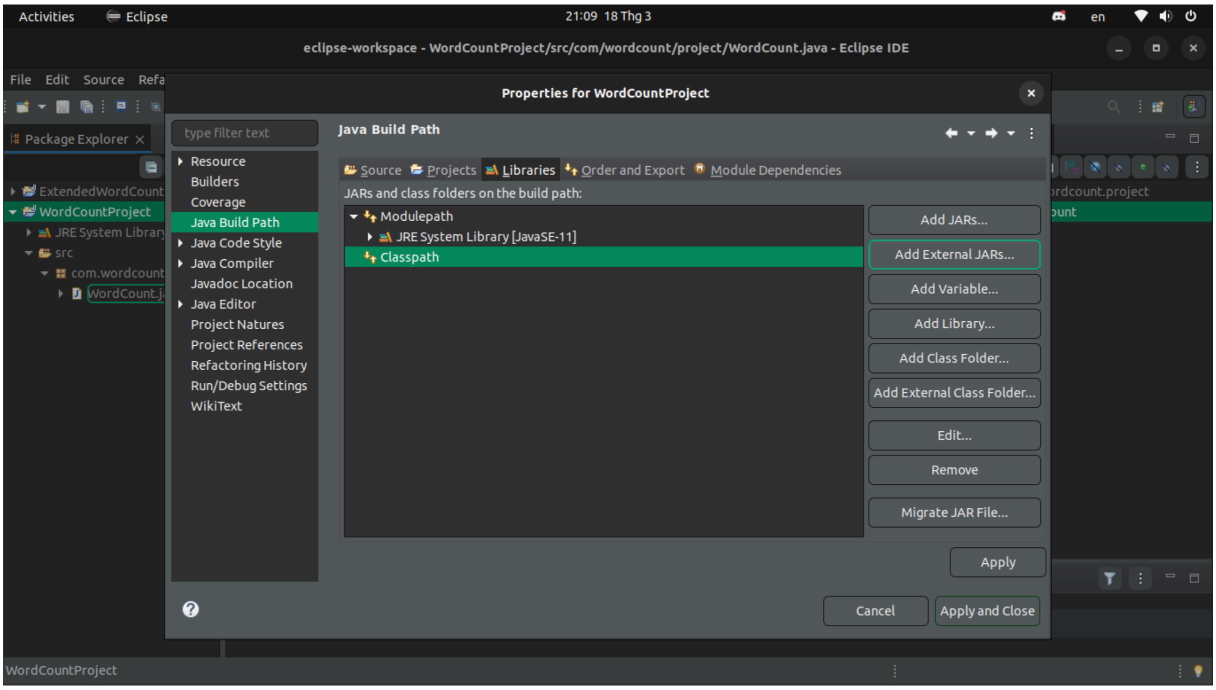
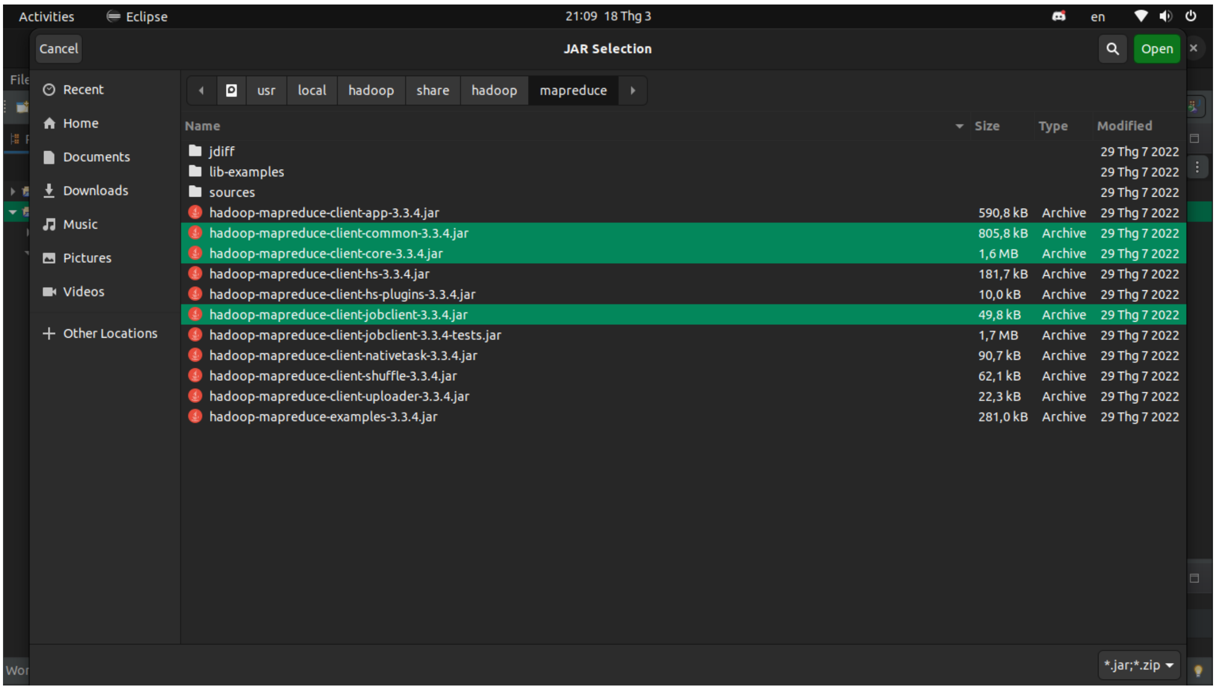
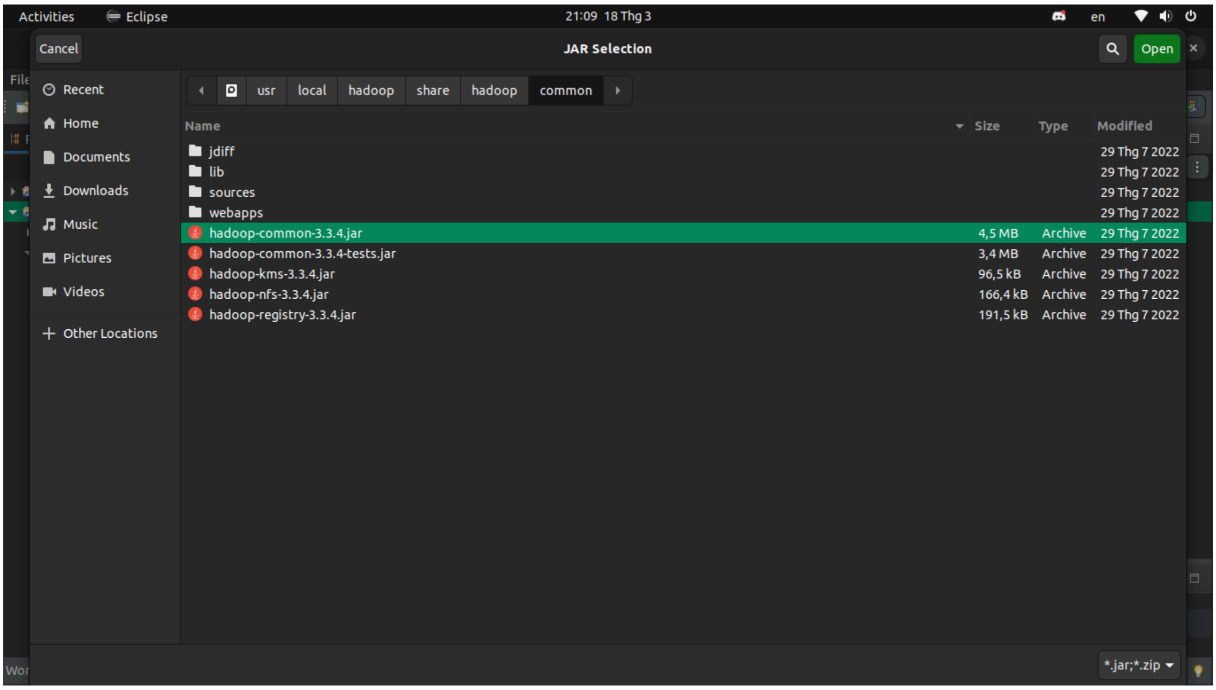


Figure 1.28: Run MapReduce



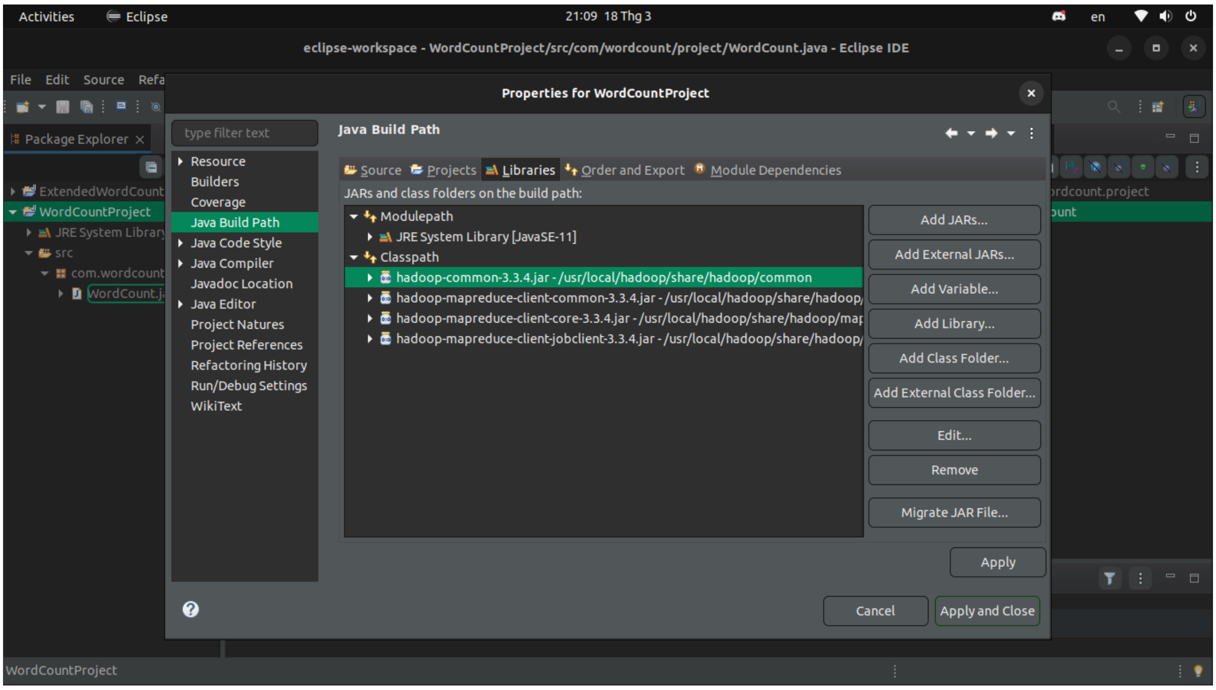
Name	Size	Type	Modified
jdiff			29 Thg 7 2022
lib-examples			29 Thg 7 2022
sources			29 Thg 7 2022
hadoop-mapreduce-client-app-3.3.4.jar	590,8 kB	Archive	29 Thg 7 2022
hadoop-mapreduce-client-common-3.3.4.jar	805,8 kB	Archive	29 Thg 7 2022
hadoop-mapreduce-client-core-3.3.4.jar	1,6 MB	Archive	29 Thg 7 2022
hadoop-mapreduce-client-hs-3.3.4.jar	181,7 kB	Archive	29 Thg 7 2022
hadoop-mapreduce-client-hs-plugins-3.3.4.jar	10,0 kB	Archive	29 Thg 7 2022
hadoop-mapreduce-client-jobclient-3.3.4.jar	49,8 kB	Archive	29 Thg 7 2022
hadoop-mapreduce-client-jobclient-3.3.4-tests.jar	1,7 MB	Archive	29 Thg 7 2022
hadoop-mapreduce-client-nativetask-3.3.4.jar	90,7 kB	Archive	29 Thg 7 2022
hadoop-mapreduce-client-shuffle-3.3.4.jar	62,1 kB	Archive	29 Thg 7 2022
hadoop-mapreduce-client-uploader-3.3.4.jar	22,3 kB	Archive	29 Thg 7 2022
hadoop-mapreduce-examples-3.3.4.jar	281,0 kB	Archive	29 Thg 7 2022

Figure 1.29: Run MapReduce



Name	Size	Type	Modified
jdiff			29 Thg 7 2022
lib			29 Thg 7 2022
sources			29 Thg 7 2022
webapps			29 Thg 7 2022
hadoop-common-3.3.4.jar	4,5 MB	Archive	29 Thg 7 2022
hadoop-common-3.3.4-tests.jar	3,4 MB	Archive	29 Thg 7 2022
hadoop-kms-3.3.4.jar	96,5 kB	Archive	29 Thg 7 2022
hadoop-nfs-3.3.4.jar	166,4 kB	Archive	29 Thg 7 2022
hadoop-registry-3.3.4.jar	191,5 kB	Archive	29 Thg 7 2022

Figure 1.30: Run MapReduce



eclipse-workspace - WordCountProject/src/com/wordcount/project/WordCount.java - Eclipse IDE

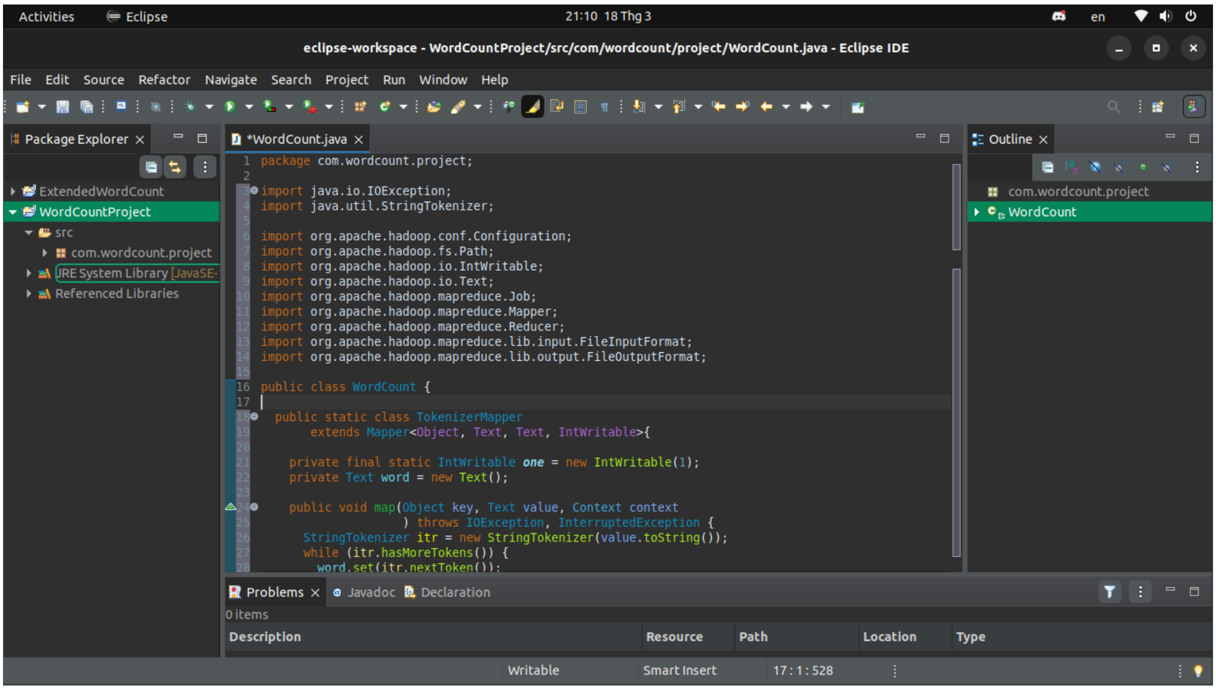
Properties for WordCountProject

Java Build Path

JARs and class folders on the build path:

- Modulepath
  - JRE System Library [JavaSE-11]
- Classpath
  - hadoop-common-3.3.4.jar - /usr/local/hadoop/share/hadoop/common
  - hadoop-mapreduce-client-common-3.3.4.jar - /usr/local/hadoop/share/hadoop/
  - hadoop-mapreduce-client-core-3.3.4.jar - /usr/local/hadoop/share/hadoop/map
  - hadoop-mapreduce-client-jobclient-3.3.4.jar - /usr/local/hadoop/share/hadoop/

Figure 1.31: Run MapReduce



eclipse-workspace - WordCountProject/src/com/wordcount/project/WordCount.java - Eclipse IDE

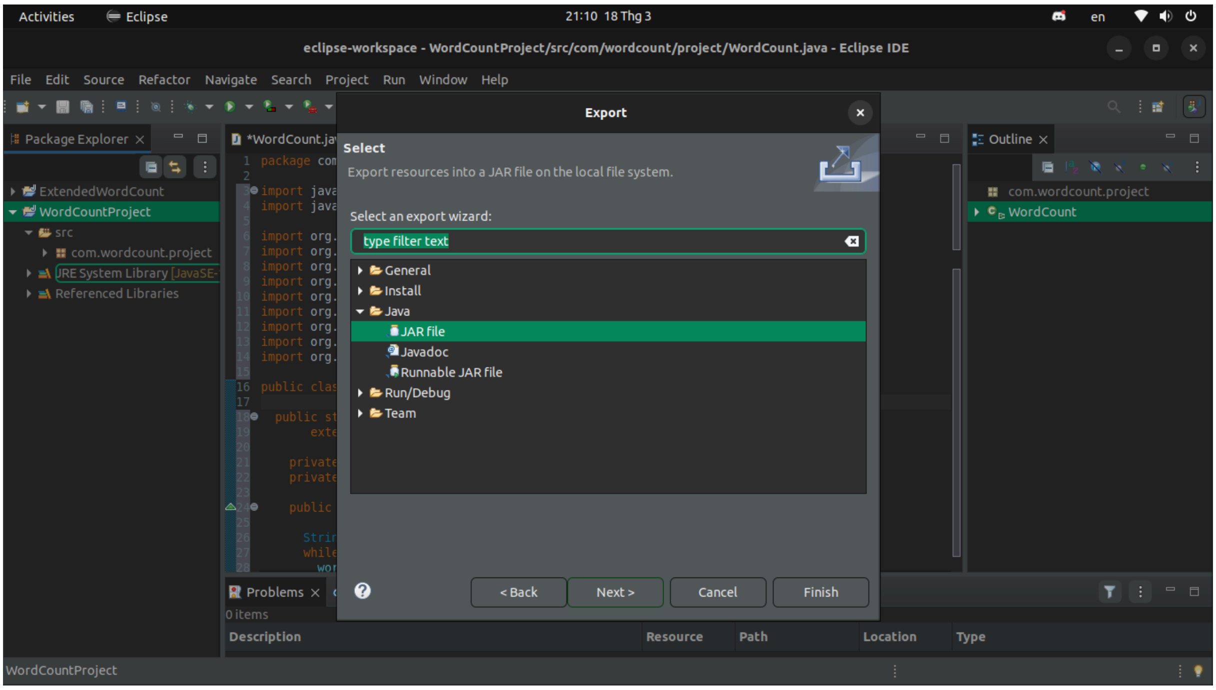
```java
package com.wordcount.project;

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());
```

Problems: 0 items

Description	Resource	Path	Location	Type

Writable Smart Insert 17 : 1 : 528

Figure 1.32: Run MapReduce



Export resources into a JAR file on the local file system.

Select an export wizard:

Figure 1.33: Run MapReduce

Enter name of jar file and path to save this jar file and. Once done, 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

After all, you will get the Jar file

1.6.9 Step 8: Prepare to run MapReduce

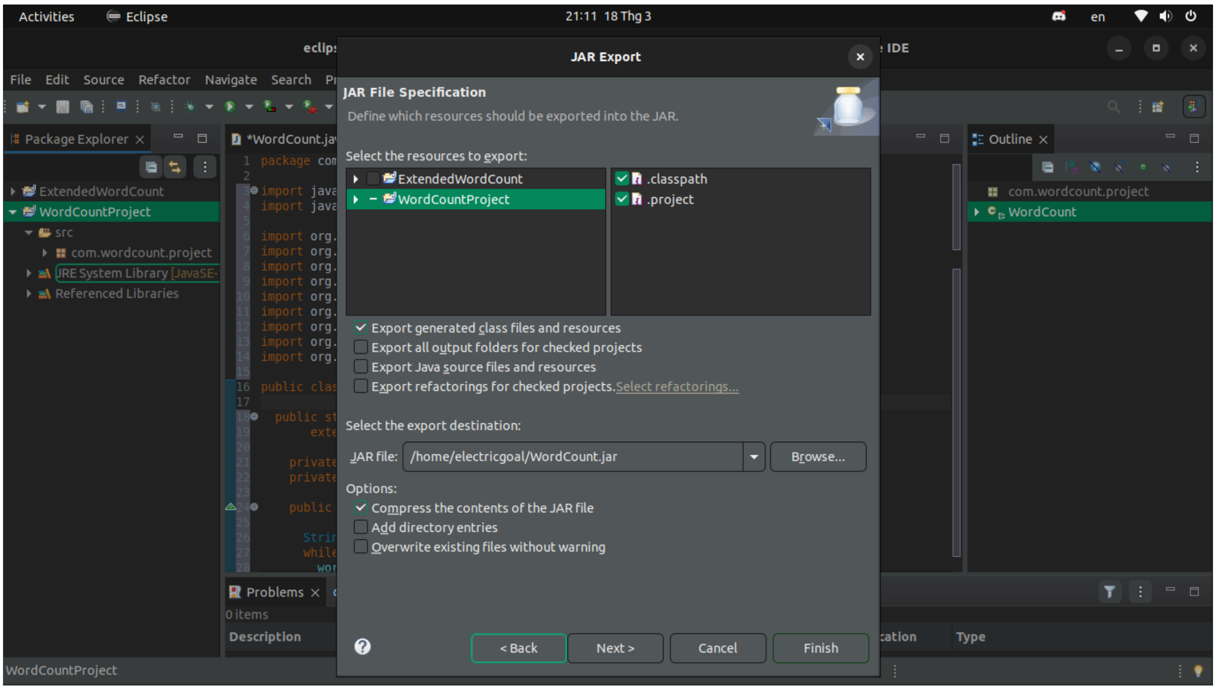
Create new folder name “wordcount” in HDFS

```
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



JAR Export

JAR File Specification

Define which resources should be exported into the JAR.

Select the resources to export:

- ☐ ExtendedWordCount
- ▣ WordCountProject
- ☑ .classpath
- ☑ .project

☑ Export generated class files and resources
☐ Export all output folders for checked projects
☐ Export Java source files and resources
☐ Export refactorings for checked projects. Select refactorings...

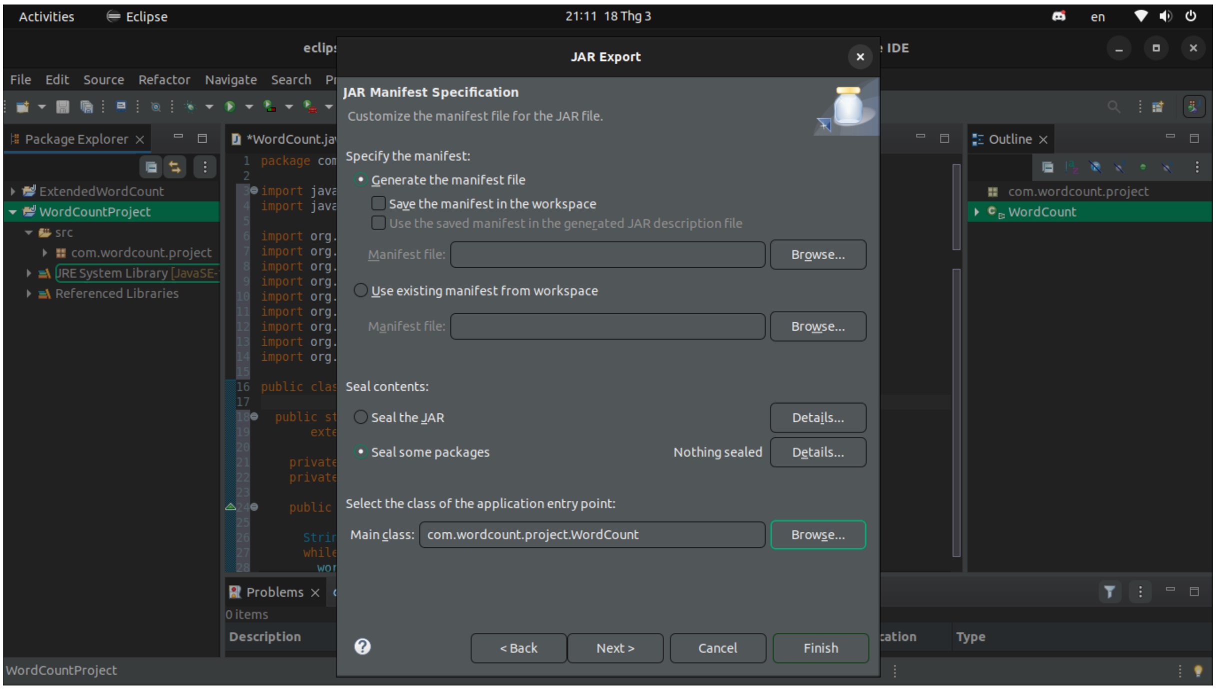
Select the export destination:

JAR file: /home/electricgoal/WordCount.jar [Browse...]

Options:

☑ Compress the contents of the JAR file
☐ Add directory entries
☐ Overwrite existing files without warning

Figure 1.34: Run MapReduce



JAR Export

JAR Manifest Specification

Customize the manifest file for the JAR file.

Specify the manifest:

● Generate the manifest file
☐ Save the manifest in the workspace
☐ Use the saved manifest in the generated JAR description file
Manifest file: [Browse...]

○ Use existing manifest from workspace
Manifest file: [Browse...]

Seal contents:

○ Seal the JAR [Details...]
● Seal some packages Nothing sealed [Details...]

Select the class of the application entry point:

Main class: com.wordcount.project.WordCount [Browse...]

Figure 1.35: Run MapReduce

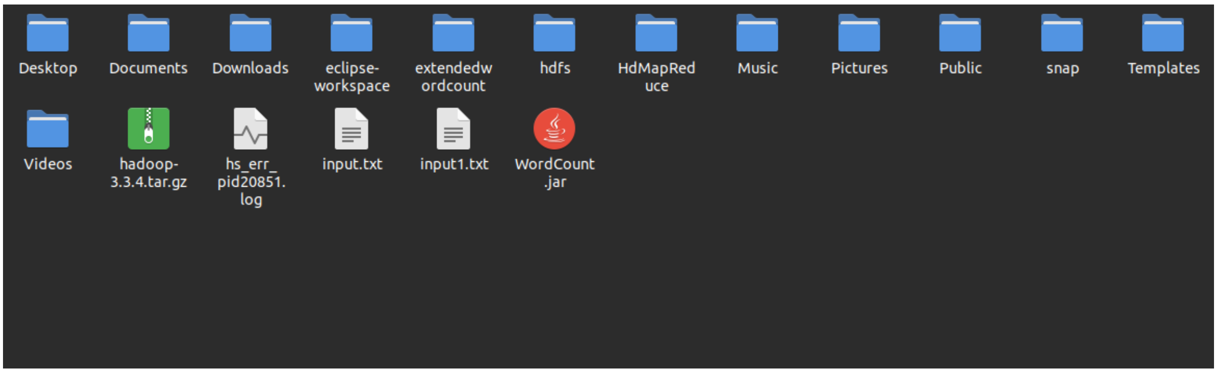


Figure 1.36: Run MapReduce

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

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

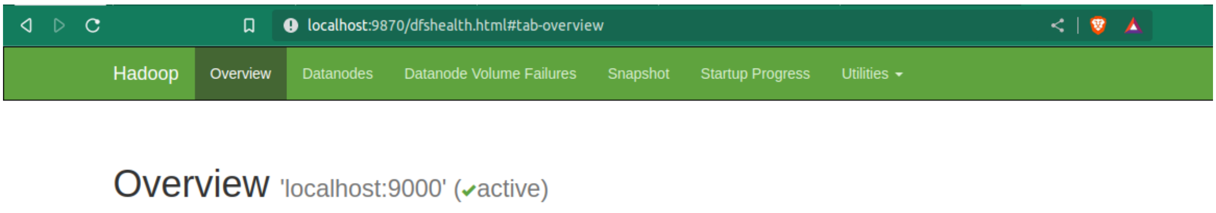


Figure 1.37: Run MapReduce

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

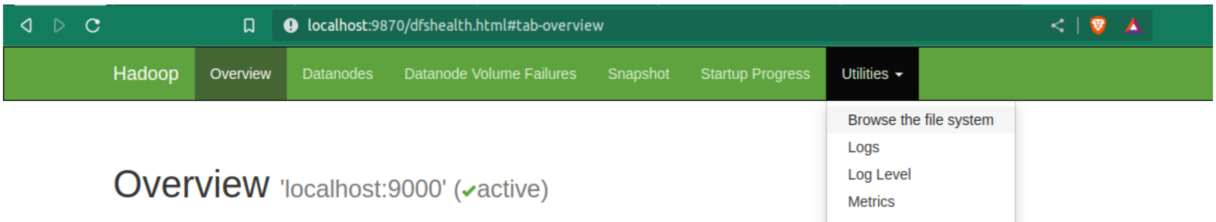
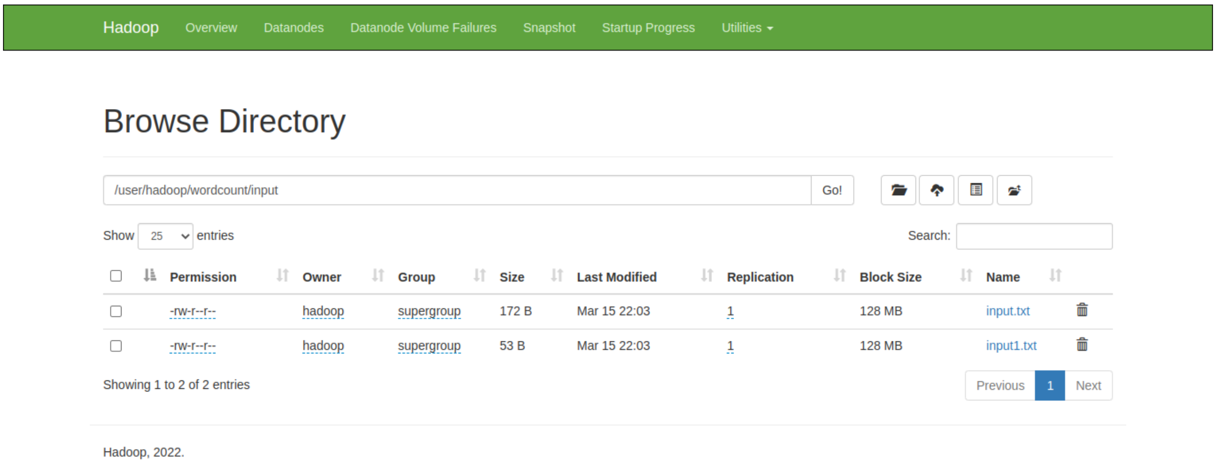


Figure 1.38: Run MapReduce

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



Browse Directory

/user/hadoop/wordcount/input

☐	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
☐	-rw-r--r--	hadoop	supergroup	172 B	Mar 15 22:03	1	128 MB	input.txt	🗑
☐	-rw-r--r--	hadoop	supergroup	53 B	Mar 15 22:03	1	128 MB	input1.txt	🗑

Showing 1 to 2 of 2 entries

Hadoop, 2022.

Figure 1.39: Run MapReduce

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/wor
```

You should see something like this

To see the result, enter this command

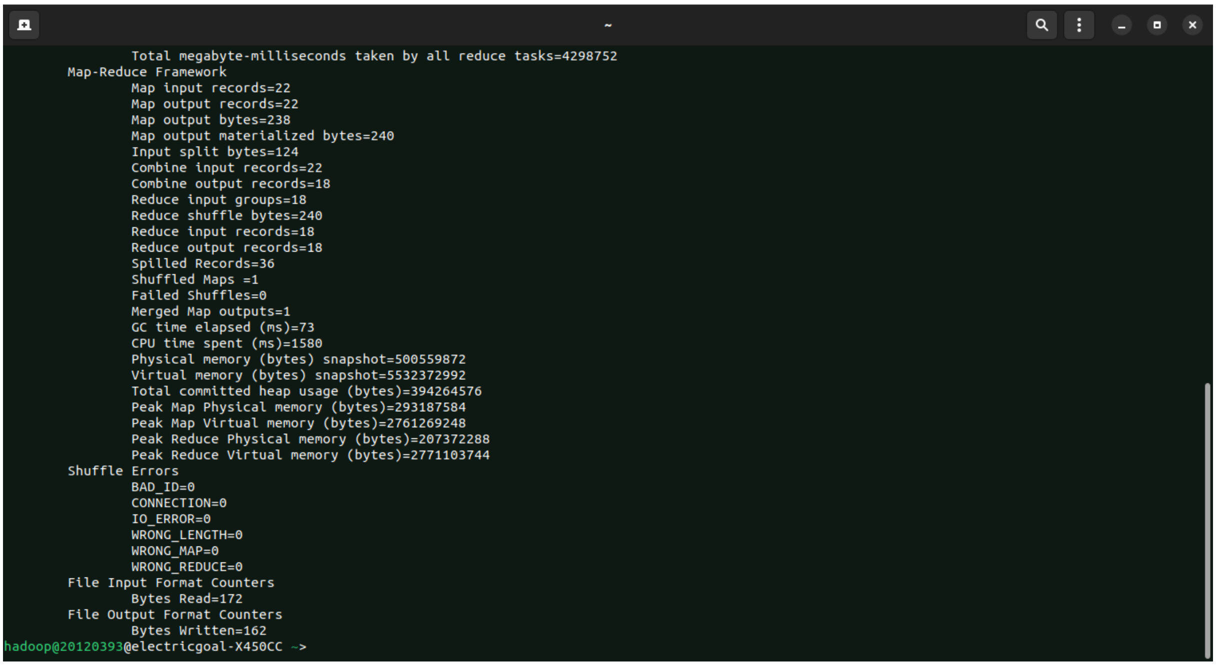
```
hadoop fs -cat /<your-favorite-path>/wordcount/output/part-r-00000

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

Compare to the input

1.7 Bonus

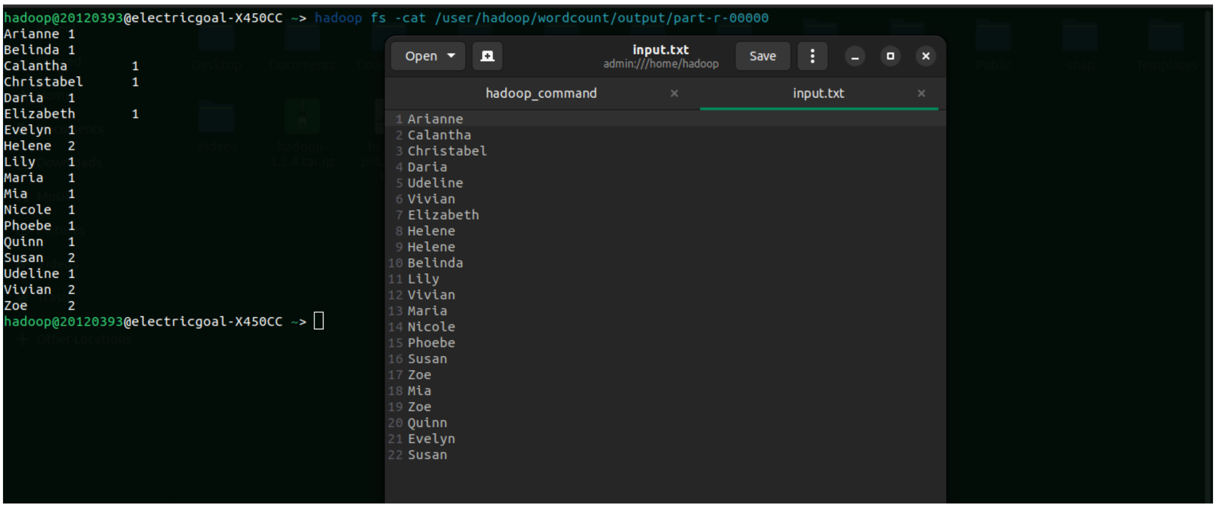
1.7.1 4.1 Extended Word Count: Unhealthy relationships

For more details, open folder `src`



```
		Total megabyte-milliseconds taken by all reduce tasks=4298752
	Map-Reduce Framework
		Map input records=22
		Map output records=22
		Map output bytes=238
		Map output materialized bytes=240
		Input split bytes=124
		Combine input records=22
		Combine output records=18
		Reduce input groups=18
		Reduce shuffle bytes=240
		Reduce input records=18
		Reduce output records=18
		Spilled Records=36
		Shuffled Maps =1
		Failed Shuffles=0
		Merged Map outputs=1
		GC time elapsed (ms)=73
		CPU time spent (ms)=1580
		Physical memory (bytes) snapshot=500559872
		Virtual memory (bytes) snapshot=5532372992
		Total committed heap usage (bytes)=394264576
		Peak Map Physical memory (bytes)=293187584
		Peak Map Virtual memory (bytes)=2761269248
		Peak Reduce Physical memory (bytes)=207372288
		Peak Reduce Virtual memory (bytes)=2771103744
	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=172
	File Output Format Counters
		Bytes Written=162
hadoop@20120393@electricgoal-X450CC ~>
```

Figure 1.40: Run MapReduce



```
hadoop@20120393@electricgoal-X450CC ~> hadoop fs -cat /user/hadoop/wordcount/output/part-r-00000
Arianne	1
Belinda	1
Calantha	1
Christabel	1
Daria	1
Elizabeth	1
Evelyn	1
Helene	2
Lily	1
Maria	1
Mia	1
Nicole	1
Phoebe	1
Quinn	1
Susan	2
Udeline	1
Vivian	2
Zoe	2
hadoop@20120393@electricgoal-X450CC ~>
```

```
Arianne
Calantha
Christabel
Daria
Udeline
Vivian
Elizabeth
Helene
Helene
Belinda
Lily
Vivian
Maria
Nicole
Phoebe
Susan
Zoe
Mia
Zoe
Quinn
Evelyn
Susan
```

Figure 1.41: Run MapReduce

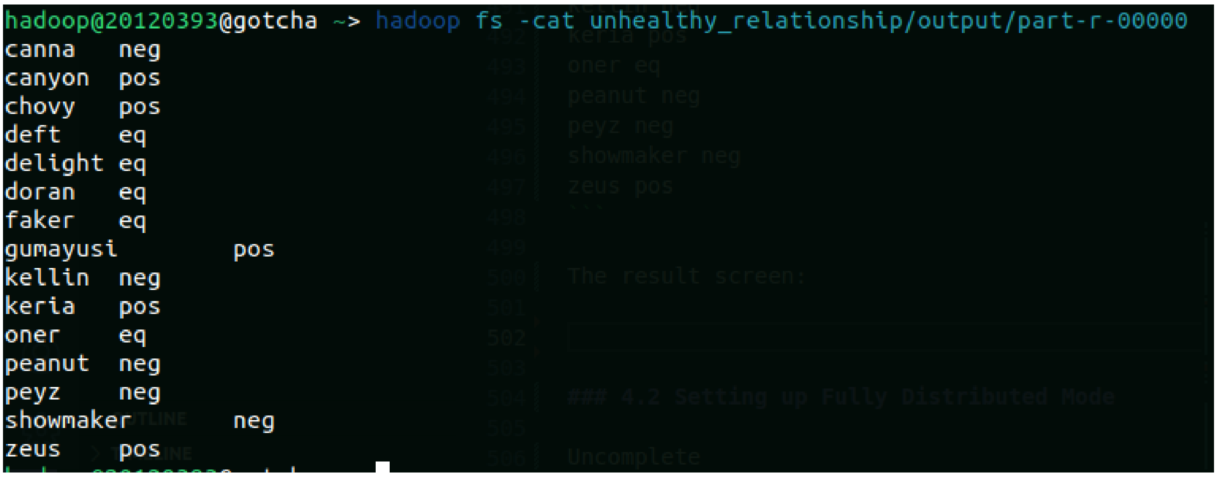
Sample input:

```
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 putput:

```
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:



```
hadoop@20120393@gotcha ~> hadoop fs -cat unhealthy_relationship/output/part-r-00000
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
```

Figure 1.42: Result

1.7.2 4.2 Setting up Fully Distributed Mode

Uncomplete

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/watch?
- 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