Lab 01: A Gentle Introduction to Hadoop

I. Team's result

ID	Name	Contribution
21120207	Nguyễn Thái Bình	25%
21120235	Trần Anh Duy	25%
21120240	Nguyễn Vân Hào	25%
21120257	Tôn Anh Huy	25%

II. Tasks

1. Setting up SNC - Single Node Cluster (4 points)

a. Nguyễn Thái Bình

Prerequisites

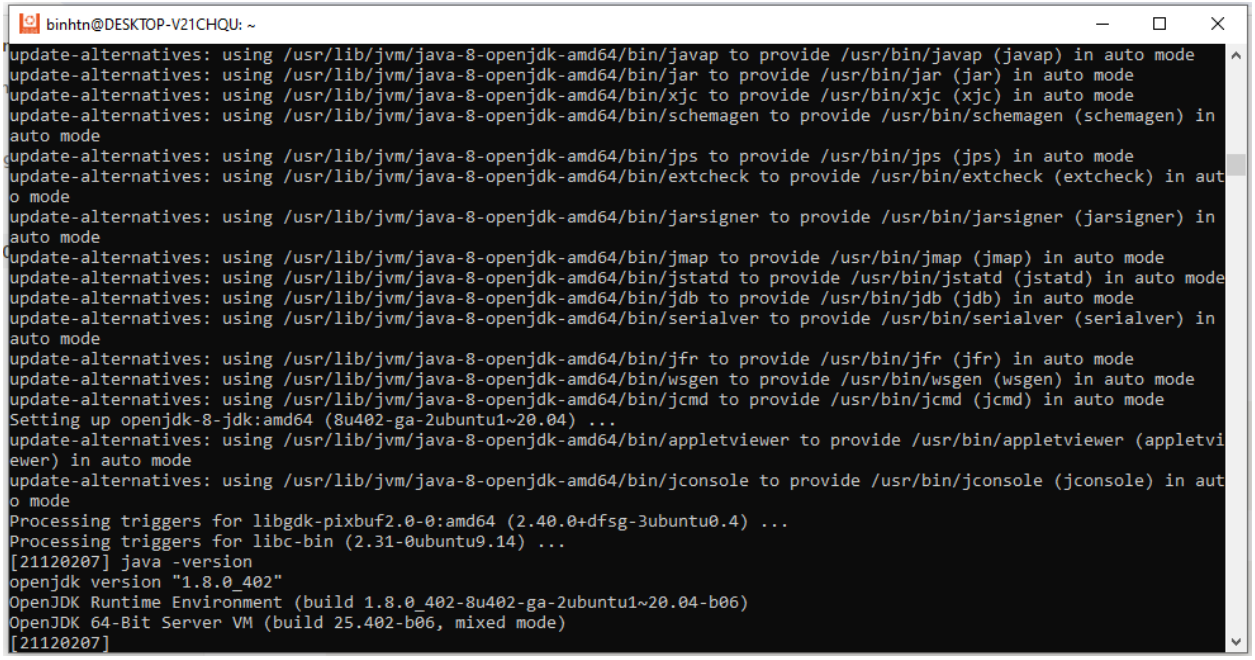
- **Java**

Install java

```
sudo apt install openjdk-8-jdk -y
```

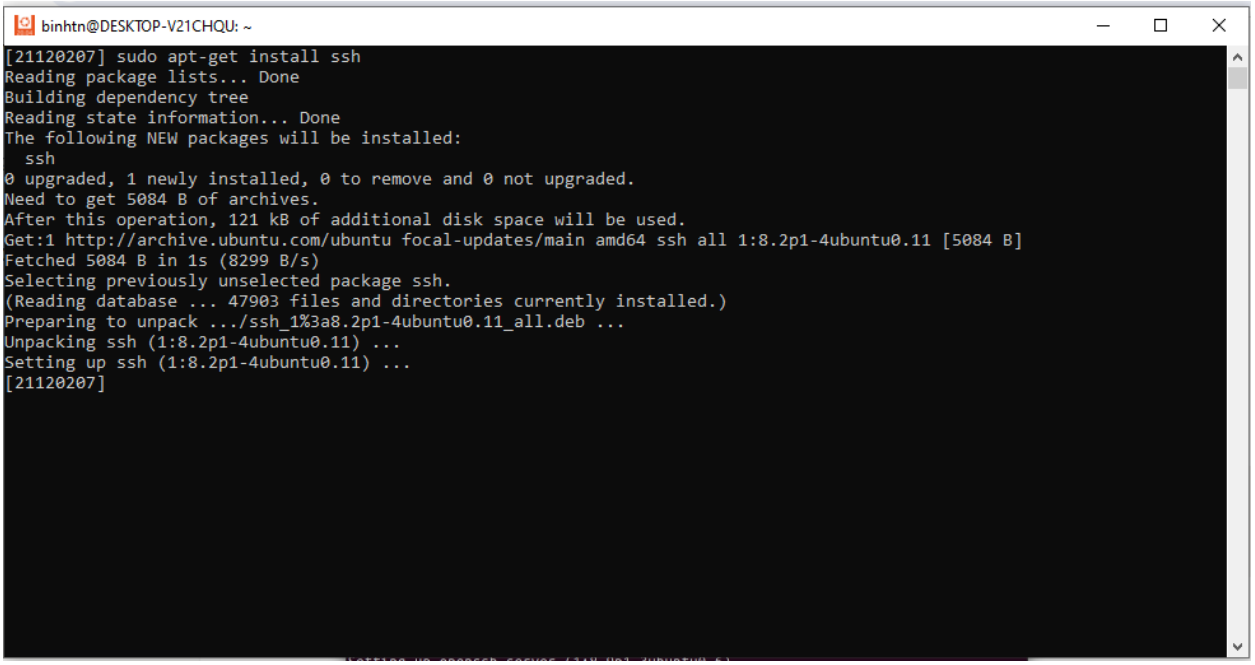
Checking Java installation

```
java -version
```



- **SSH**

```
sudo apt-get install ssh
```



Install Hadoop

Get the latest stable version from Apache Download Mirrors

Extract the downloaded file

```
tar xvzf hadoop-3.4.0.tar.gz
```

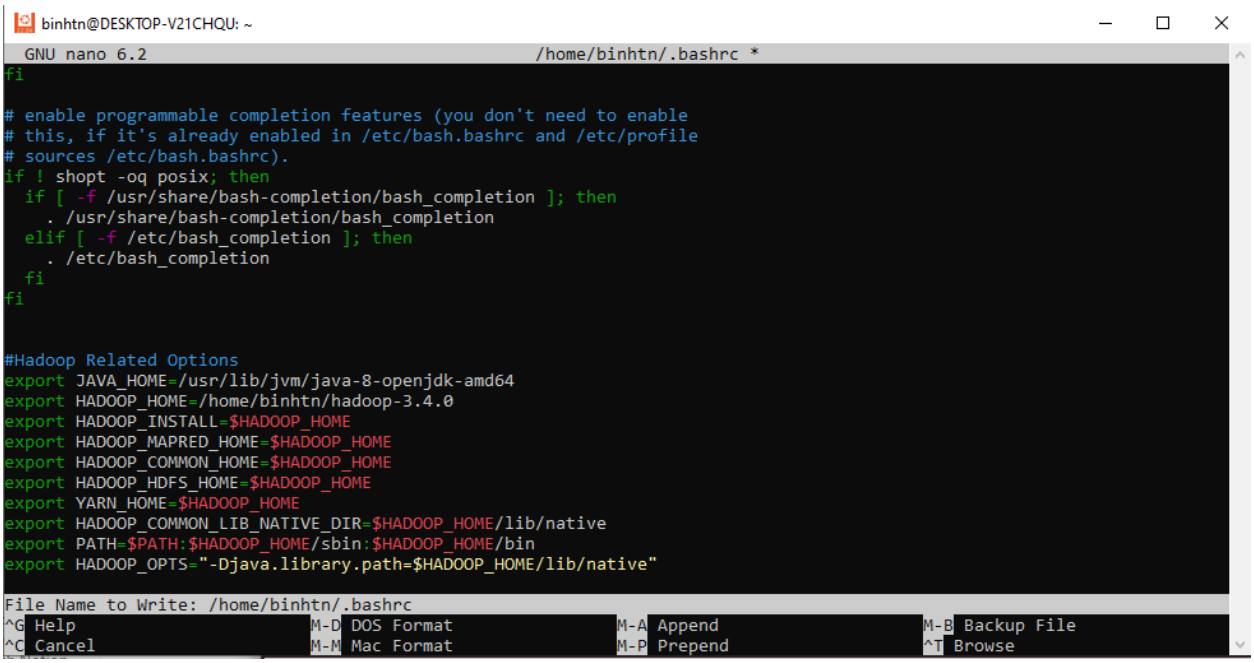
Configure Hadoop environment variables

Open .bashrc file

```
sudo nano ~/.bashrc
```

Add the following variables to the end of the file then save it

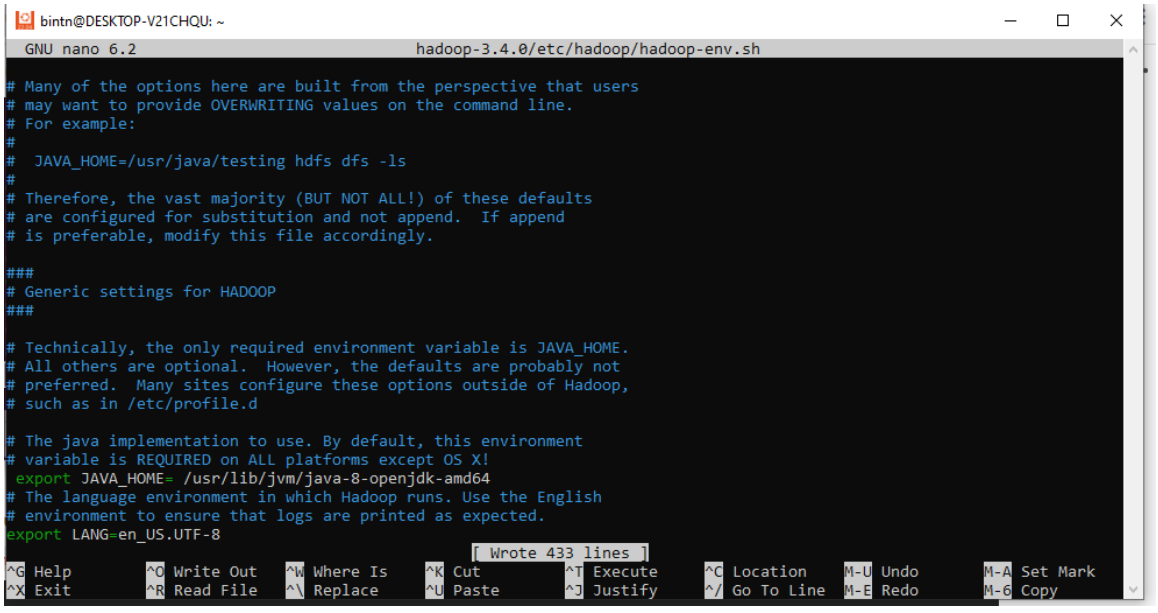
```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HADOOP_HOME=/home/binhtn/hadoop-3.4.0
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
```



Edit the file hadoop-evn.sh

```
sudo nano hadoop-3.4.0/etc/hadoop/hadoop-env.sh
```

Find JAVA_HOME and change it as shown in the image below

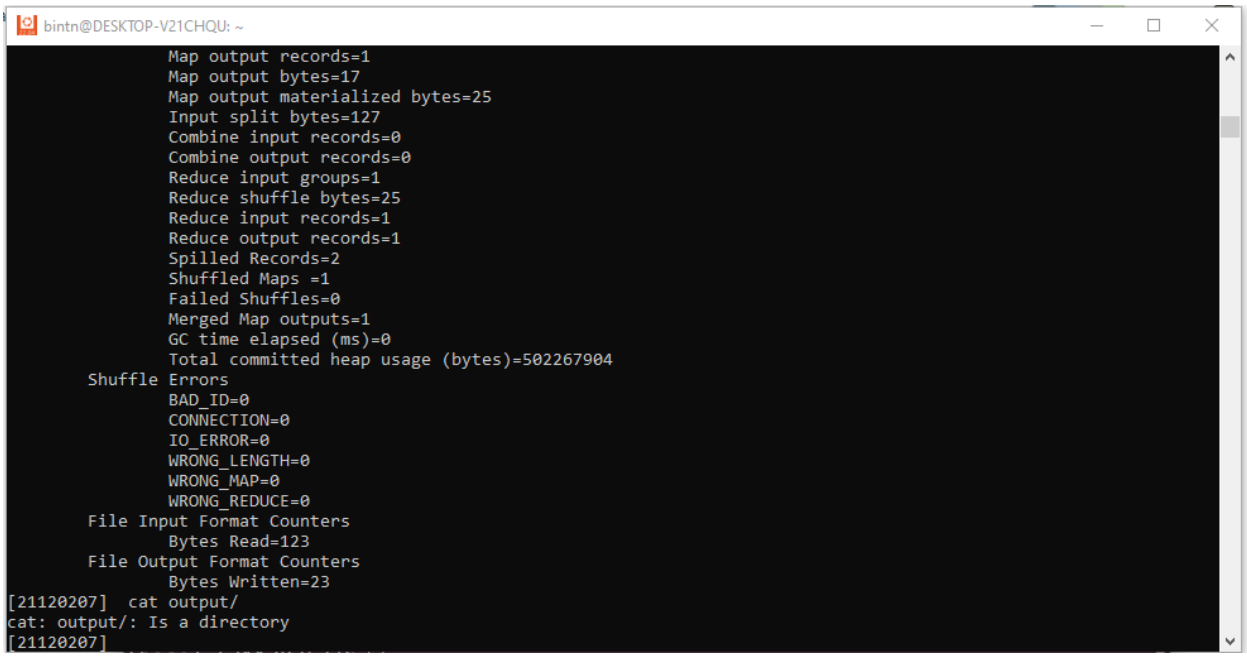


Standalone Operation

In `hadoop-3.4.0` folder, type these command

```
mkdir input
cp etc/hadoop/*.xml input
bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-example
cat output/
```

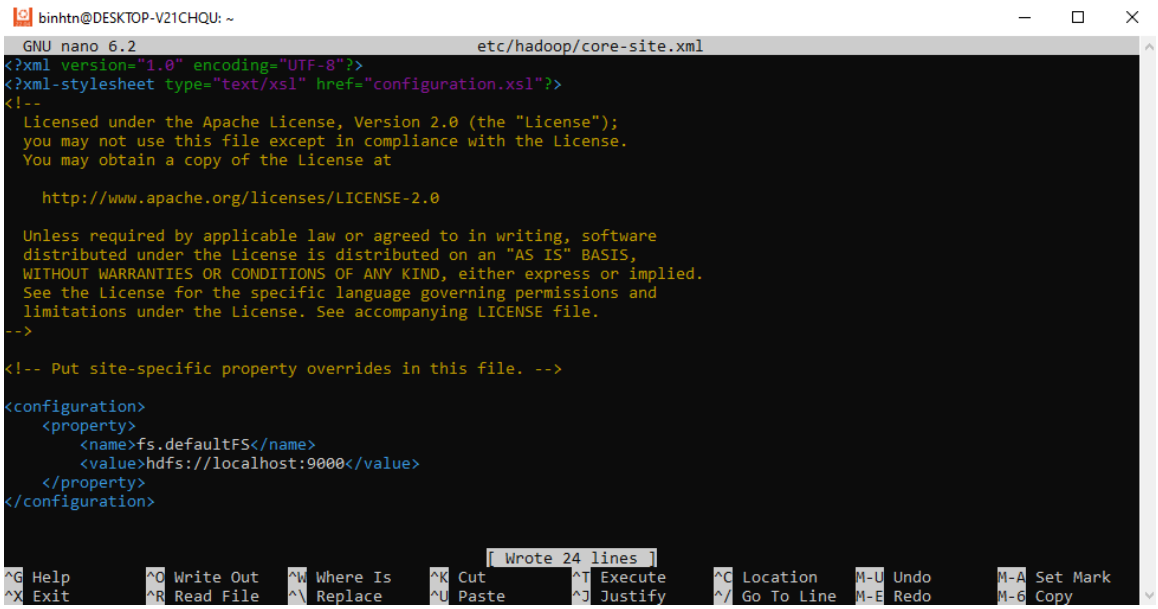
The following example copies the unpacked conf directory to use as input and then finds and displays every match of the given regular expression. Output is written to the given output directory.



Pseudo-Distributed Operation

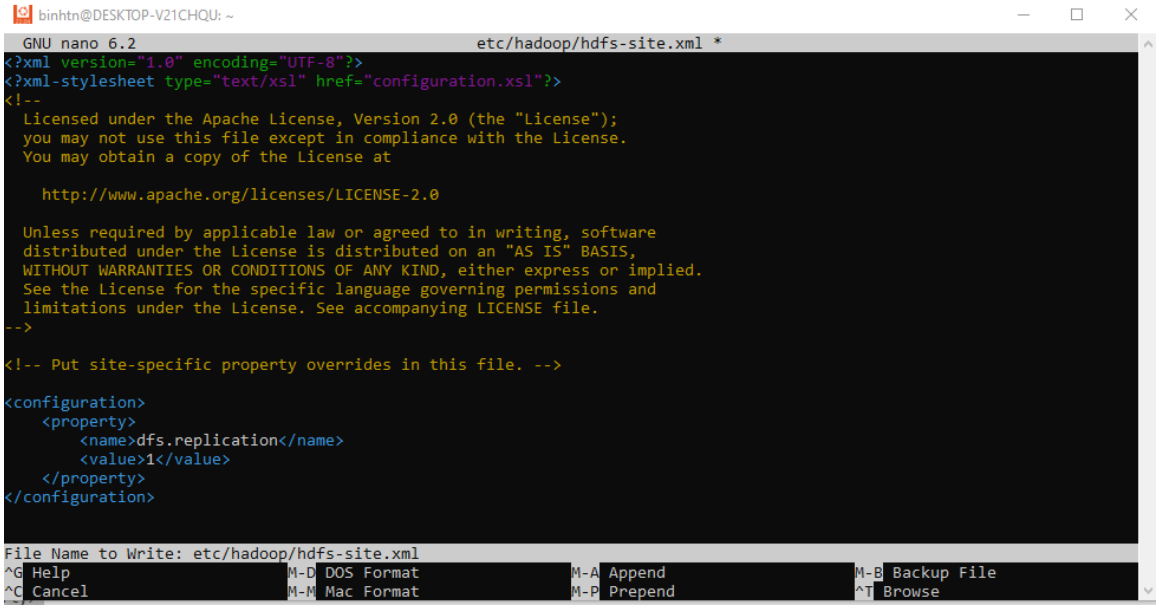
Config `etc/hadoop/core-site.xml`

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



Config `etc/hadoop/hdfs-site.xml`

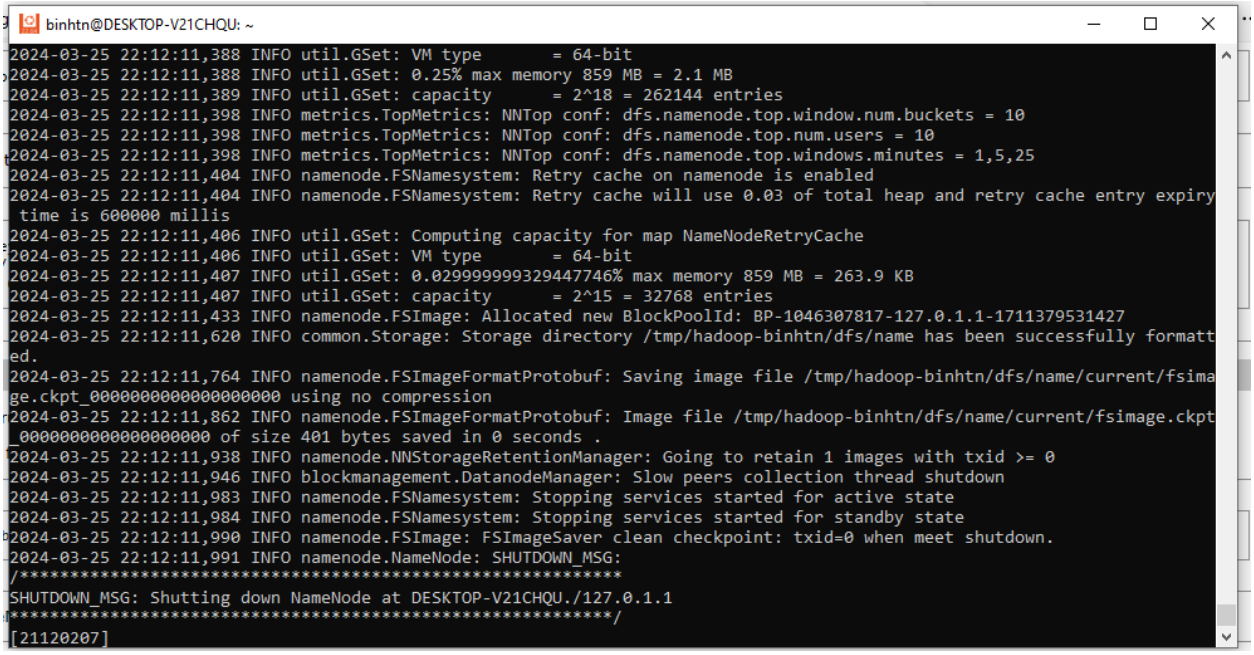
```
<configuration>
    <property>
        <name>dfs.replication</name>
        <value>1</value>
    </property>
</configuration>
```



Execution

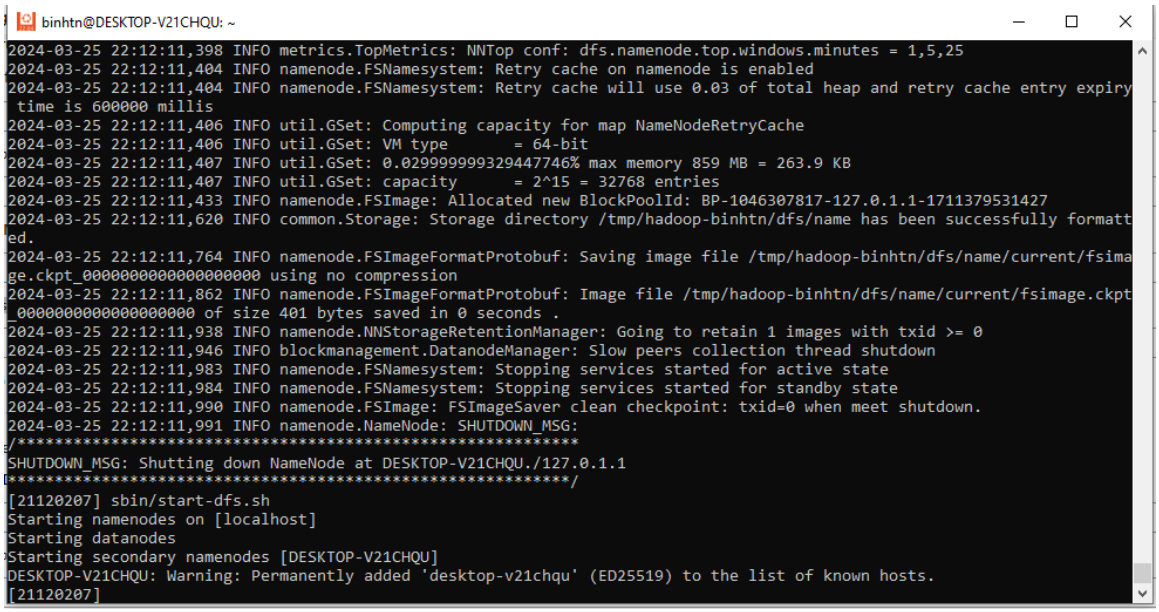
Format filesystem

```
bin/hdfs namenode -format
```

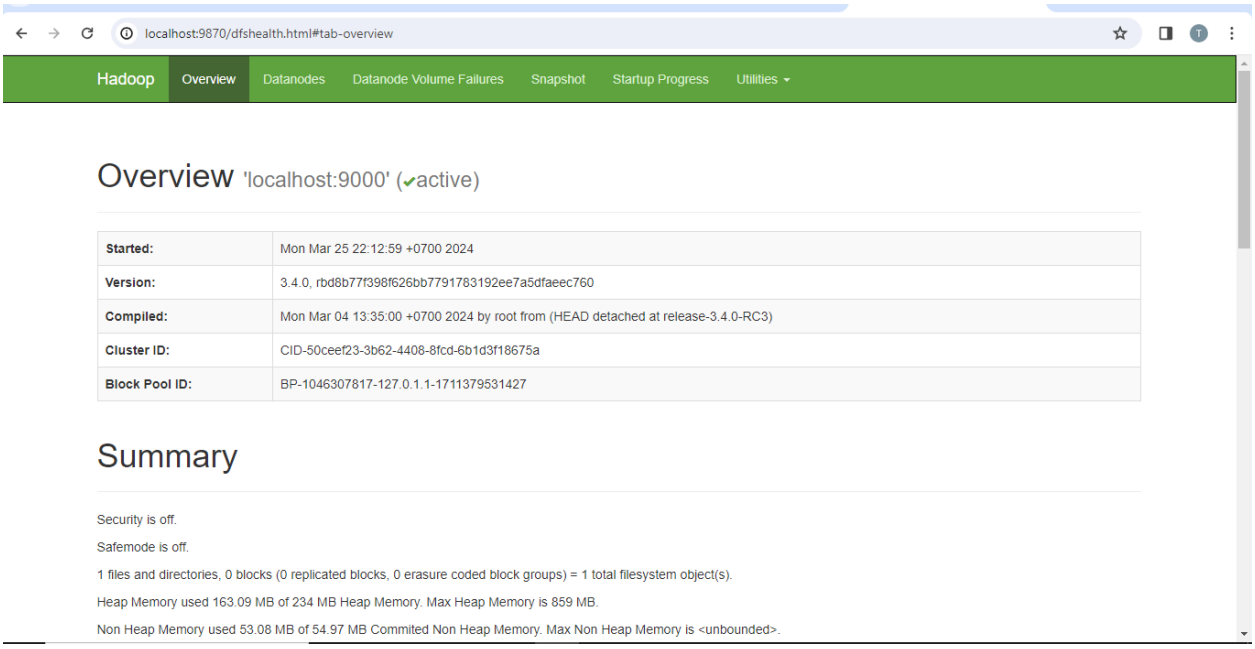


Start NameNode daemon and DataNode daemon

```
sbin/start-dfs.sh
```



Browse the web interface for the NameNode; by default it is available at localhost:9870

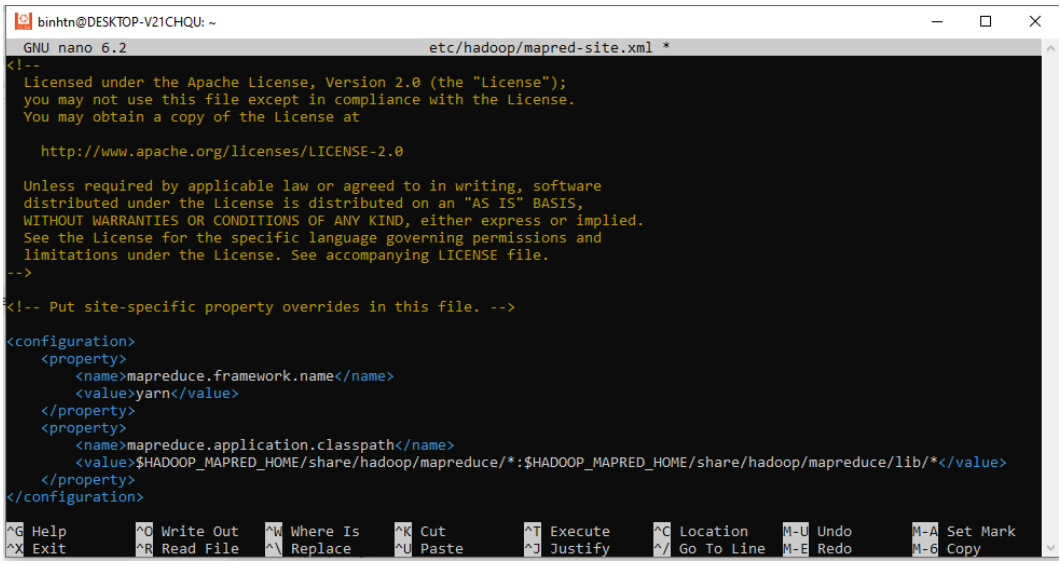


Make the HDFS directories required to execute MapReduce jobs:

```
bin/hdfs dfs -mkdir -p /user/21120207
```

Config `etc/hadoop/mapred-site.xml`

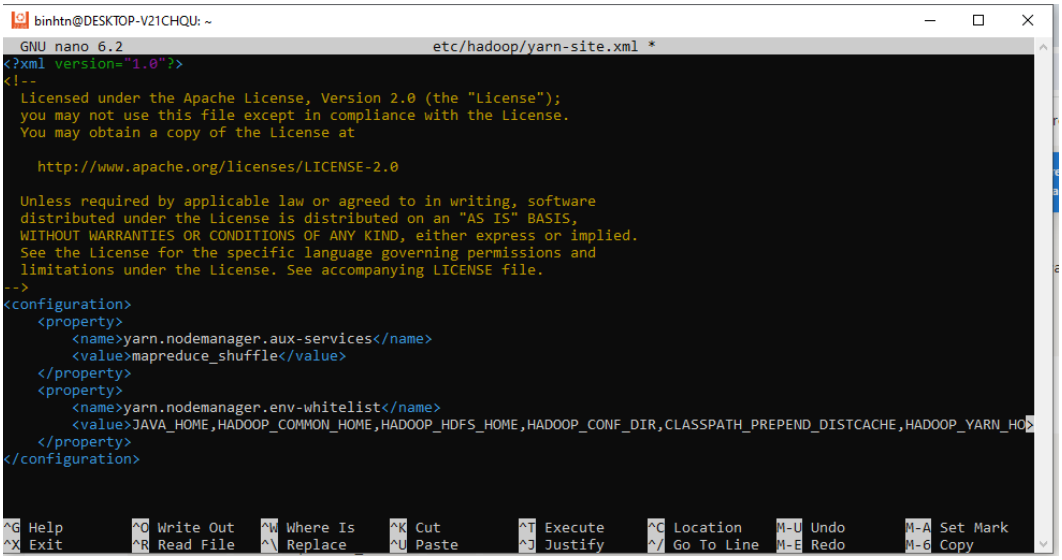
```
<configuration>
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>
    <property>
        <name>mapreduce.application.classpath</name>
        <value>$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/*:$HAI
    </property>
</configuration>
```



Config `etc/hadoop/yarn-site.xml`

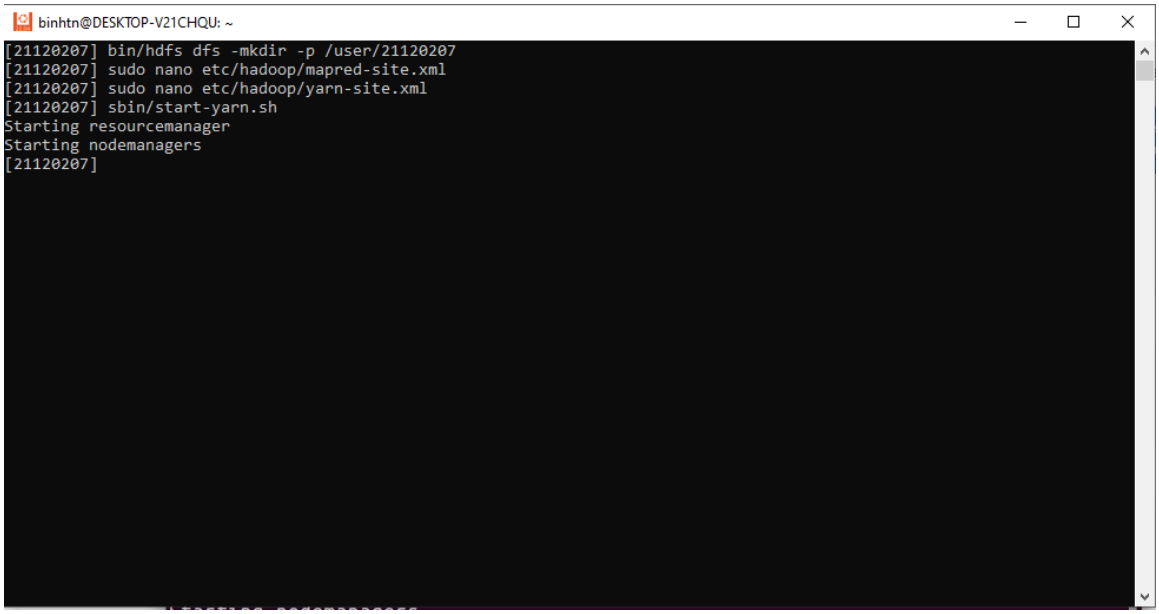
```
<configuration>
    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    </property>
    <property>
        <name>yarn.nodemanager.env-whitelist</name>
        <value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HAI
```

```
    </property>
</configuration>
```

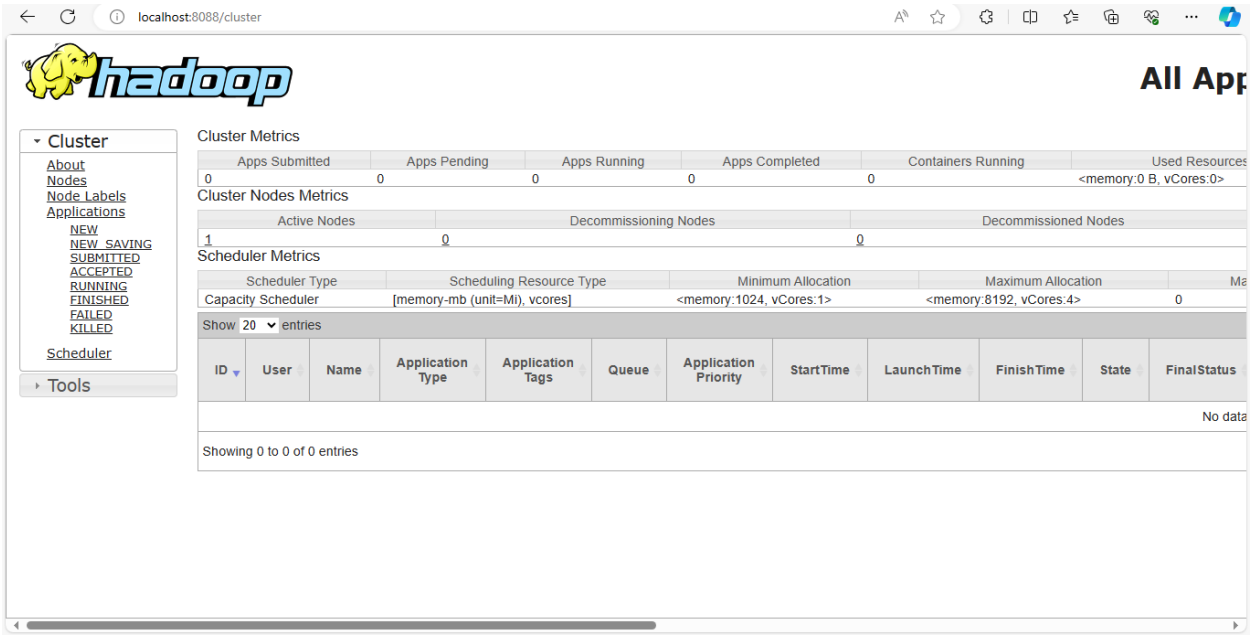


Start ResourceManager daemon and NodeManager daemon

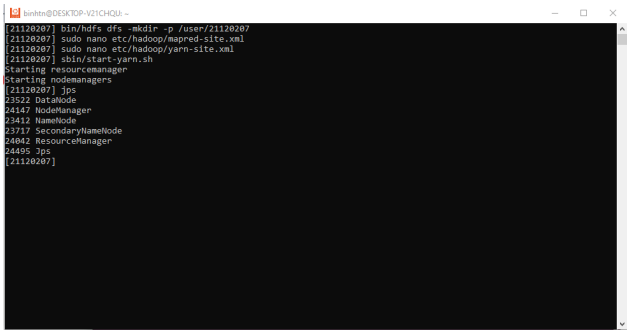
```
sbin/start-yarn.sh
```



Check web interface of ResourceManager at localhost:8088



Run `jps` to check



b. Trần Anh Duy - 21120235

Prerequisites

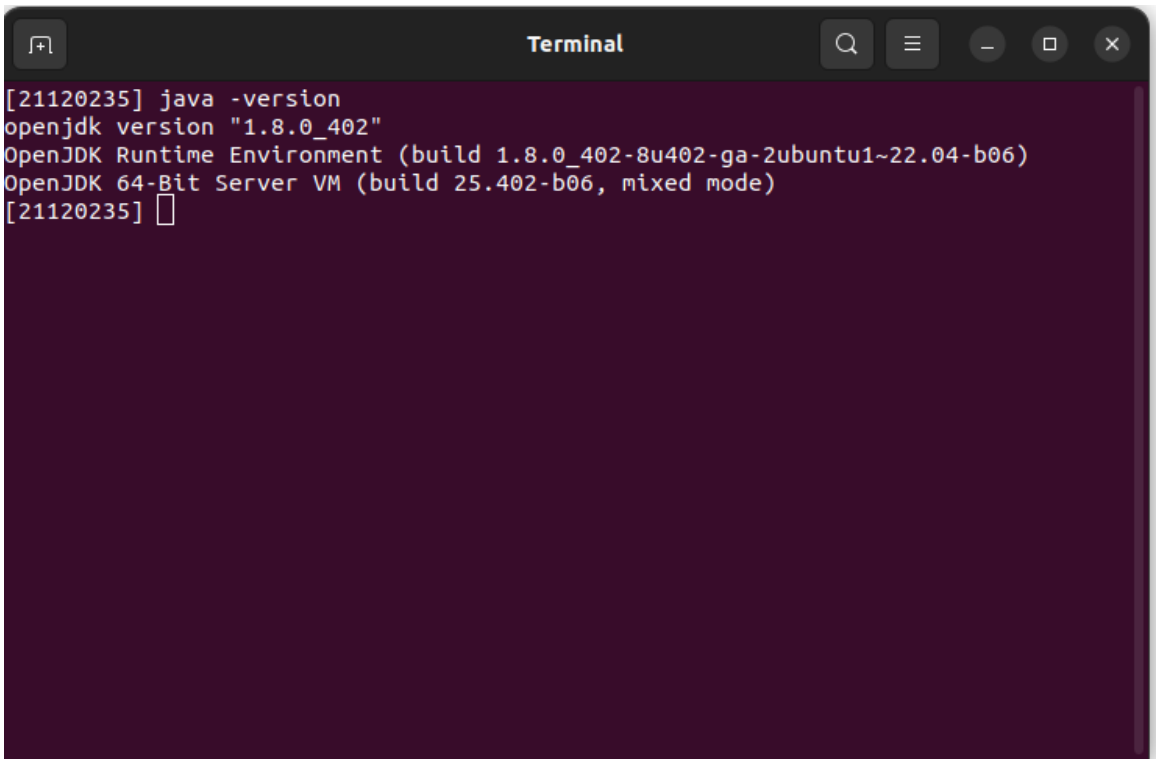
- Java

Install Java

```
sudo apt install openjdk-8-jdk -y
```

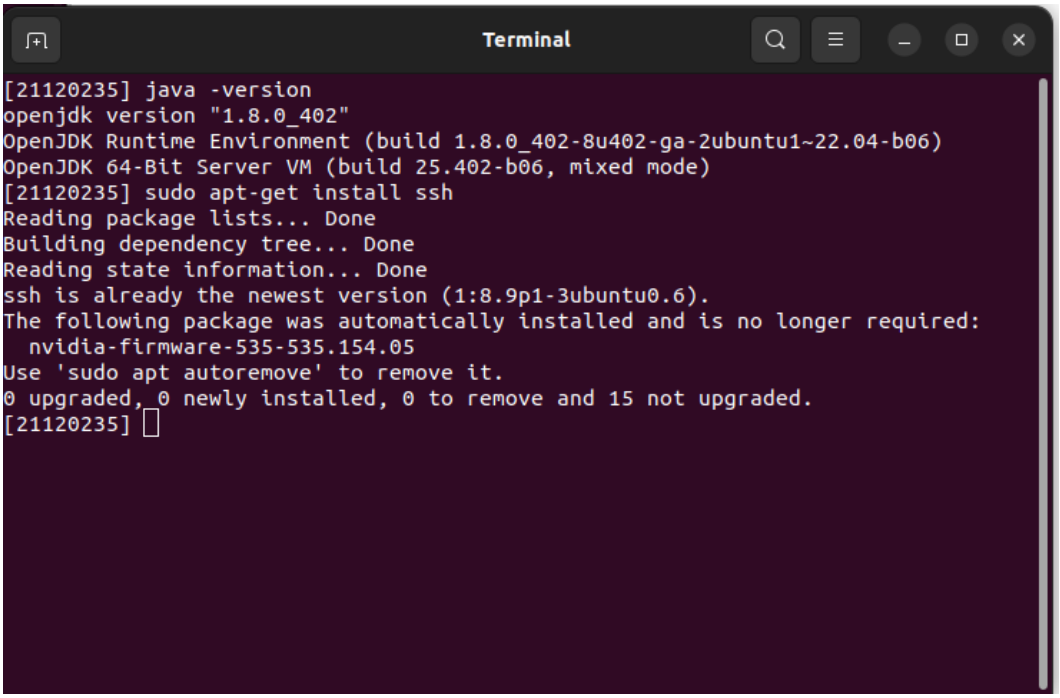
Check if Java is already installed:

```
java -version
```



- SSH

```
sudo apt-get install ssh
```



Download Hadoop

Download the latest stable version Hadoop in this link: https://hadoop.apache.org/releases.html t

Extract file after download:

```
tar xvzf hadoop-3.4.0.tar.gz
```

After that, in the current directory will has a folder named `hadoop-3.4.0`

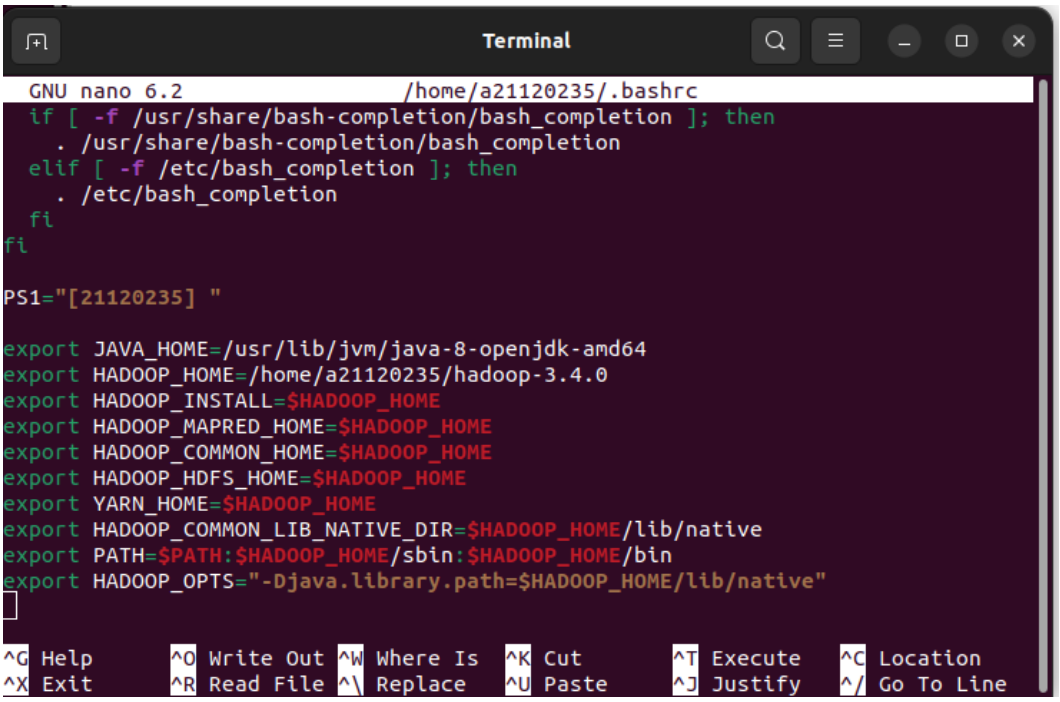
Set up environment variables for Hadoop (file .bashrc)

Open .bashrc file

```
sudo nano ~/.bashrc
```

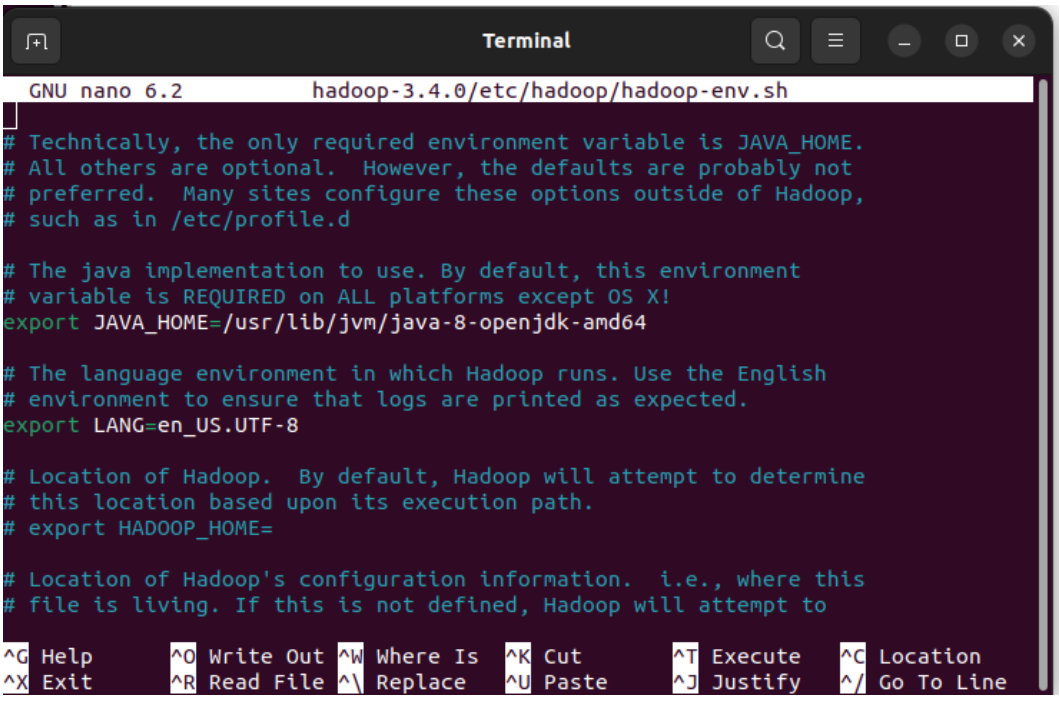
Paste all lines to .bashrc then save it

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HADOOP_HOME=/home/duy/hadoop-3.4.0
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export 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"
```



Edit hadoop-env `sudo nano hadoop-3.4.0/etc/hadoop/hadoop-env.sh`

Find `export JAVA_HOME` line to edit to `/usr/lib/jvm/java-8-openjdk-amd64`



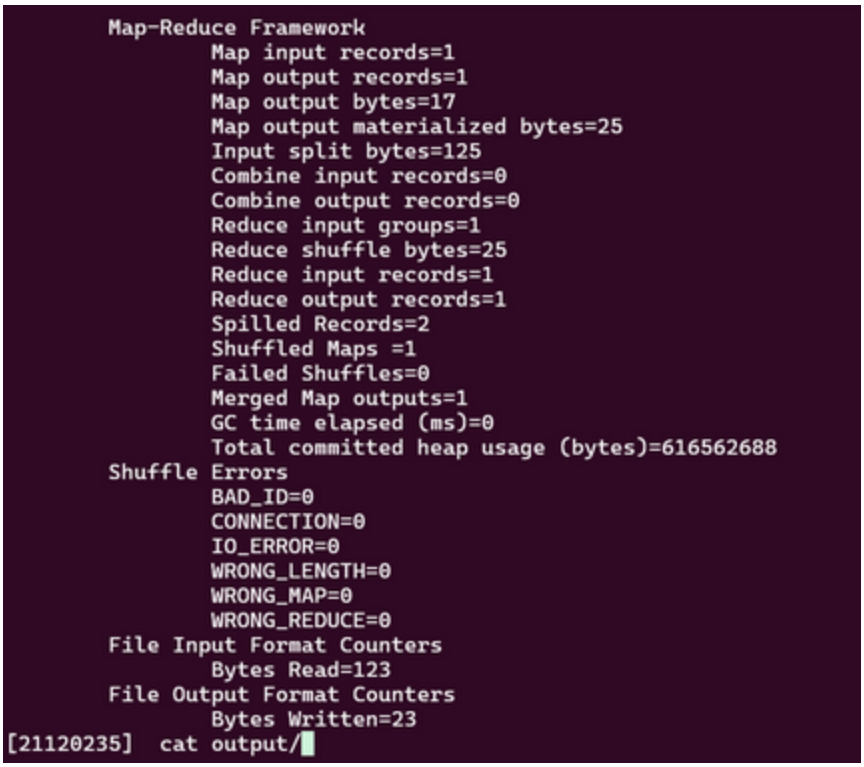
Standalone Operation

In `hadoop-3.4.0` folder, type these command

```
mkdir input
cp etc/hadoop/*.xml input
bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-example
cat output/
```

By default, Hadoop is configured to run in a non-distributed mode, as a single Java process. This is useful for debugging.

The following example copies the unpacked conf directory to use as input and then finds and displays every match of the given regular expression. Output is written to the given output directory.



Check the output content:

Pseudo-Distributed Operation

Hadoop can also be run on a single-node in a pseudo-distributed mode where each Hadoop daemon runs in a separate Java process.

Config `etc/hadoop/core-site.xml`

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

Config `etc/hadoop/hdfs-site.xml`

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

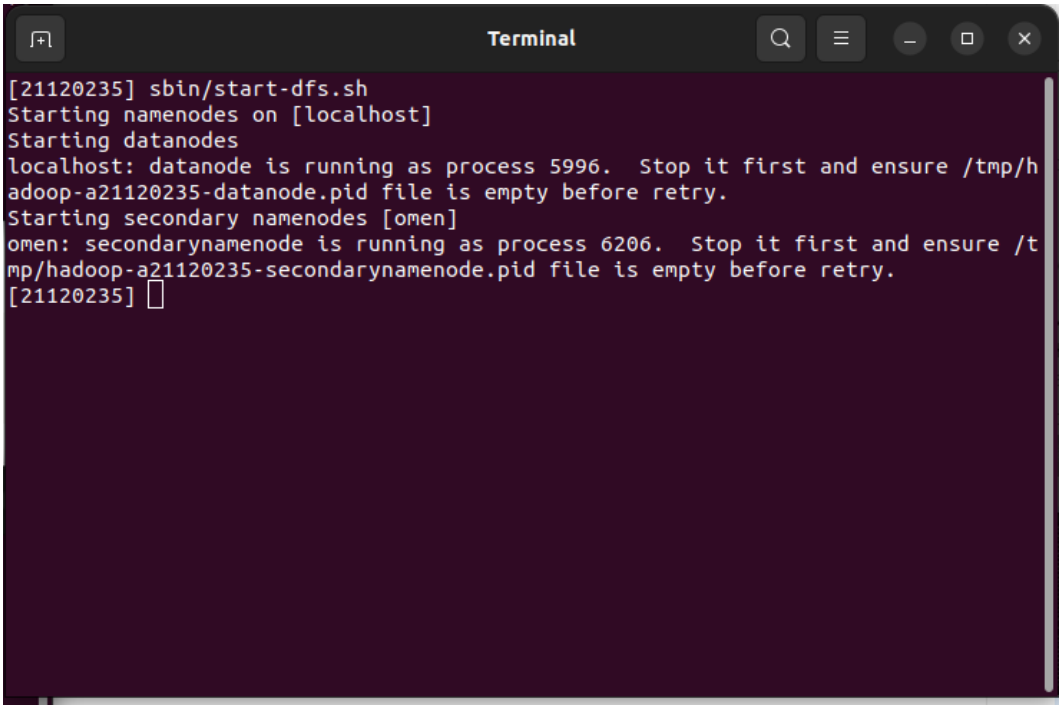
Format filesystem

```
bin/hdfs namenode -format
```

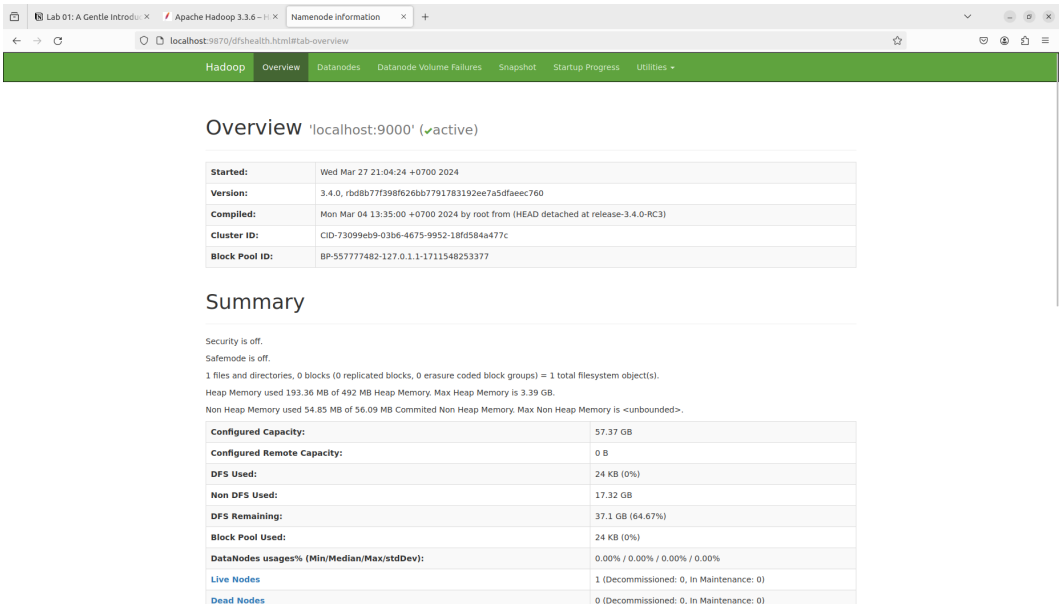
```
ation= null ? (Y or N) y
2024-03-27 19:14:04,531 INFO namenode.FSImage: Allocated new BlockPoolId: BP-
1277433387-127.0.1.1-1711541644523
2024-03-27 19:14:04,532 INFO common.Storage: Will remove files: [/tmp/hadoop-
duy/dfs/name/current/fsimage_000000000000000000.md5, /tmp/hadoop-duy/dfs/nam
e/current/VERSION, /tmp/hadoop-duy/dfs/name/current/seen_txid, /tmp/hadoop-du
y/dfs/name/current/fsimage_000000000000000000]
2024-03-27 19:14:04,549 INFO common.Storage: Storage directory /tmp/hadoop-du
y/dfs/name has been successfully formatted.
2024-03-27 19:14:04,581 INFO namenode.FSImageFormatProtobuf: Saving image fil
e /tmp/hadoop-duy/dfs/name/current/fsimage.ckpt_000000000000000000 using no
compression
2024-03-27 19:14:04,702 INFO namenode.FSImageFormatProtobuf: Image file /tmp/
hadoop-duy/dfs/name/current/fsimage.ckpt_000000000000000000 of size 395 byte
s saved in 0 seconds .
2024-03-27 19:14:04,717 INFO namenode.NNStorageRetentionManager: Going to ret
ain 1 images with txid >= 0
2024-03-27 19:14:04,724 INFO blockmanagement.DatanodeManager: Slow peers coll
ection thread shutdown
2024-03-27 19:14:04,745 INFO namenode.FSNamesystem: Stopping services started
 for active state
2024-03-27 19:14:04,746 INFO namenode.FSNamesystem: Stopping services started
 for standby state
2024-03-27 19:14:04,749 INFO namenode.FSImage: FSImageSaver clean checkpoint:
 txid=0 when meet shutdown.
2024-03-27 19:14:04,750 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at Omen/127.0.1.1
************************************************************/
[21120235]
```

Start NameNode daemon and DataNode daemon

```
sbin/start-dfs.sh
```



Browse the web interface for the NameNode at http://localhost:9870

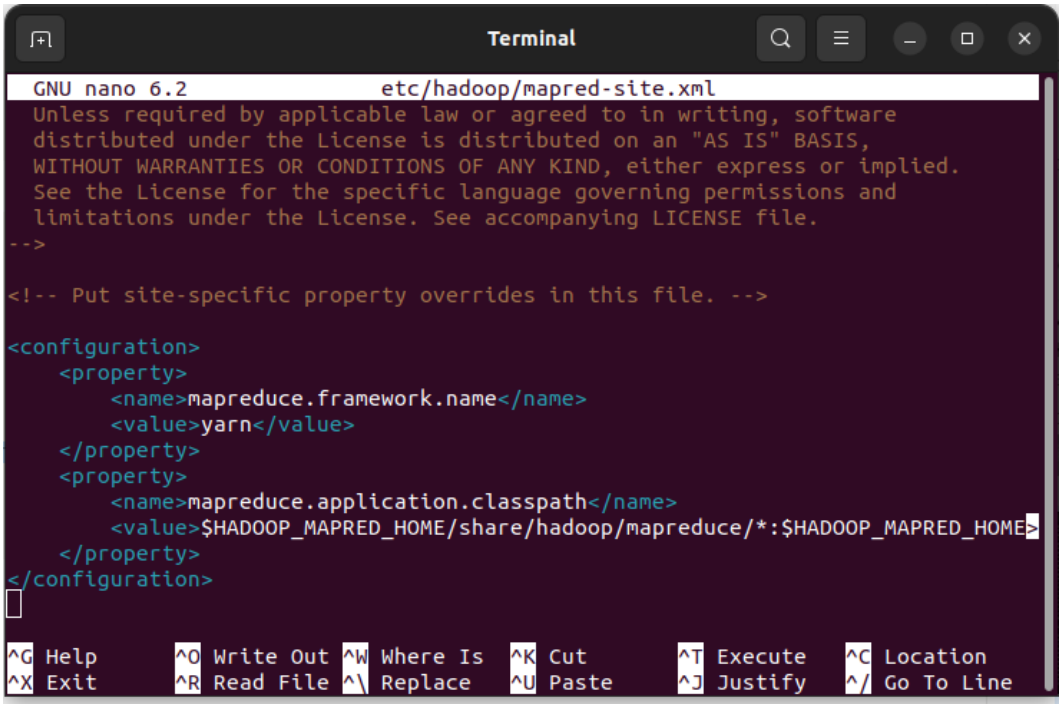


Create directory required to execute MapReduce jobs:

```
bin/hdfs dfs -mkdir -p /user/21120235
```

Config `etc/hadoop/mapred-site.xml`

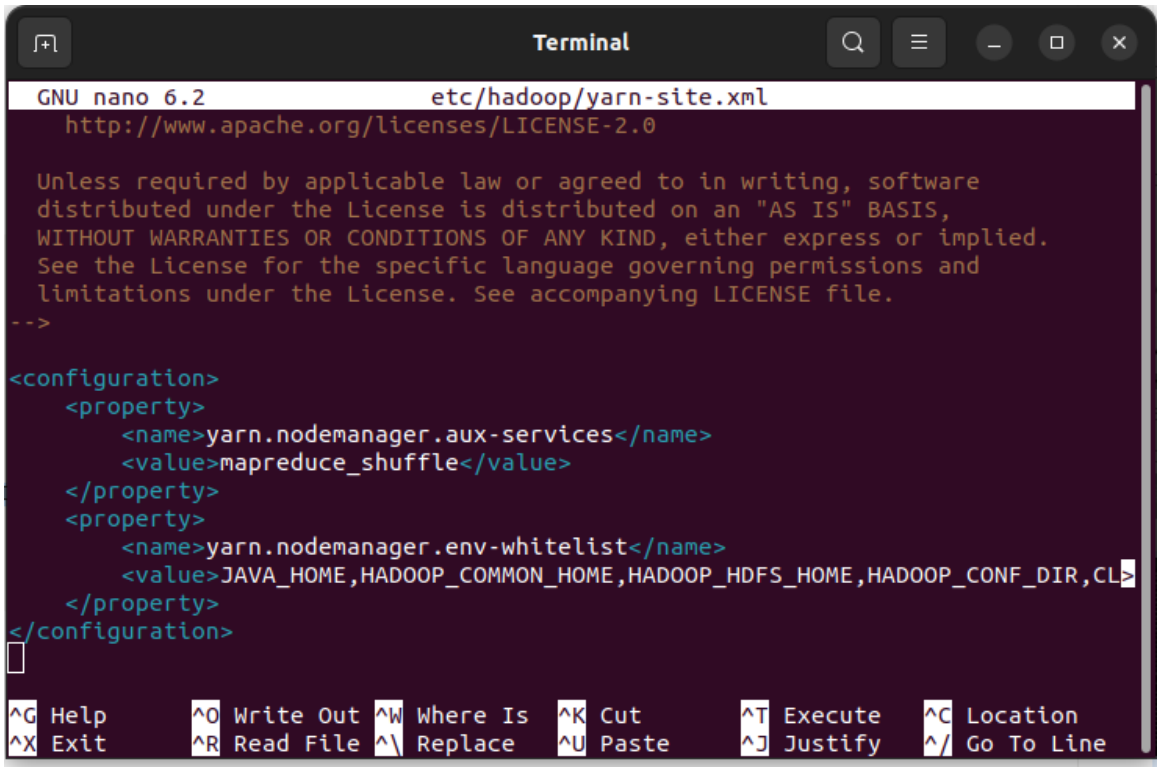
```
<configuration>
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>
    <property>
        <name>mapreduce.application.classpath</name>
        <value>$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/*:$HAI
    </property>
</configuration>
```



Config `etc/hadoop/yarn-site.xml`

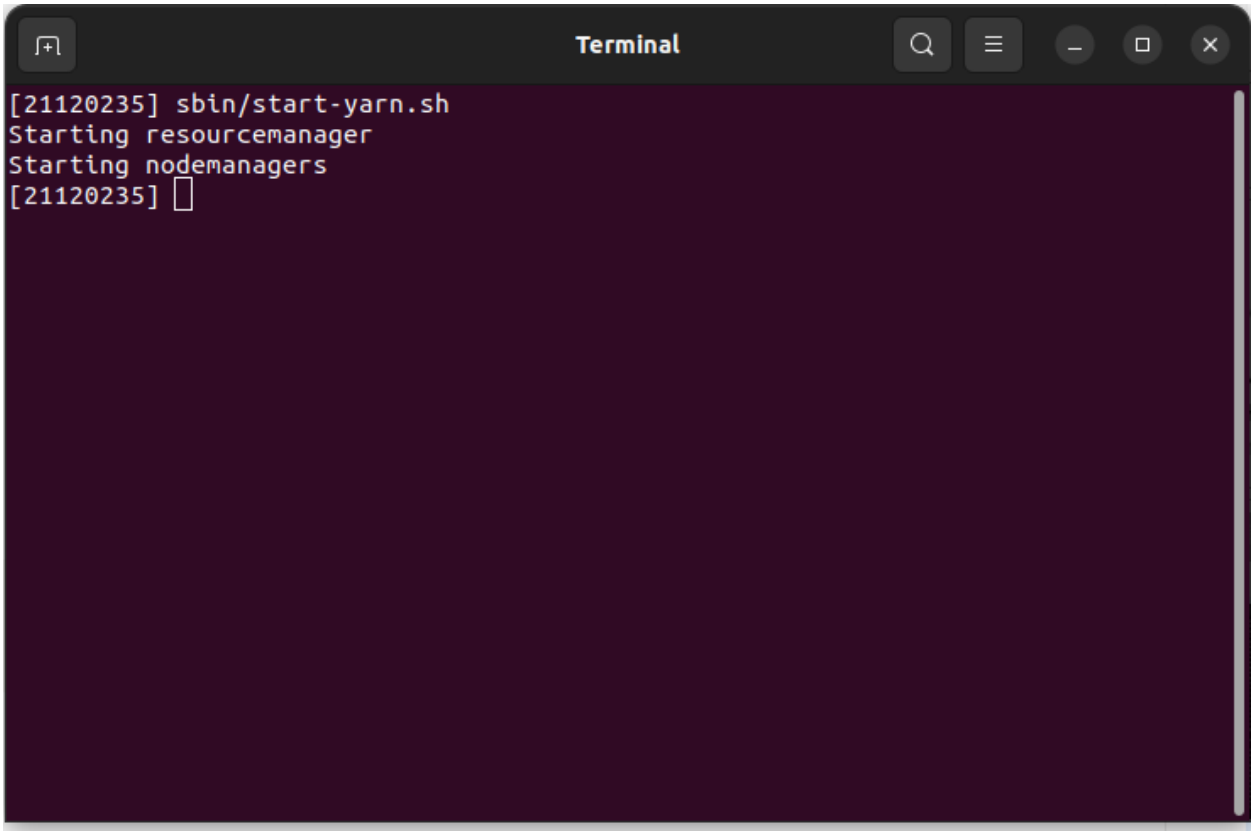
```
<configuration>
    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
```

```
    </property>
    <property>
        <name>yarn.nodemanager.env-whitelist</name>
        <value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HAI
    </property>
</configuration>
```

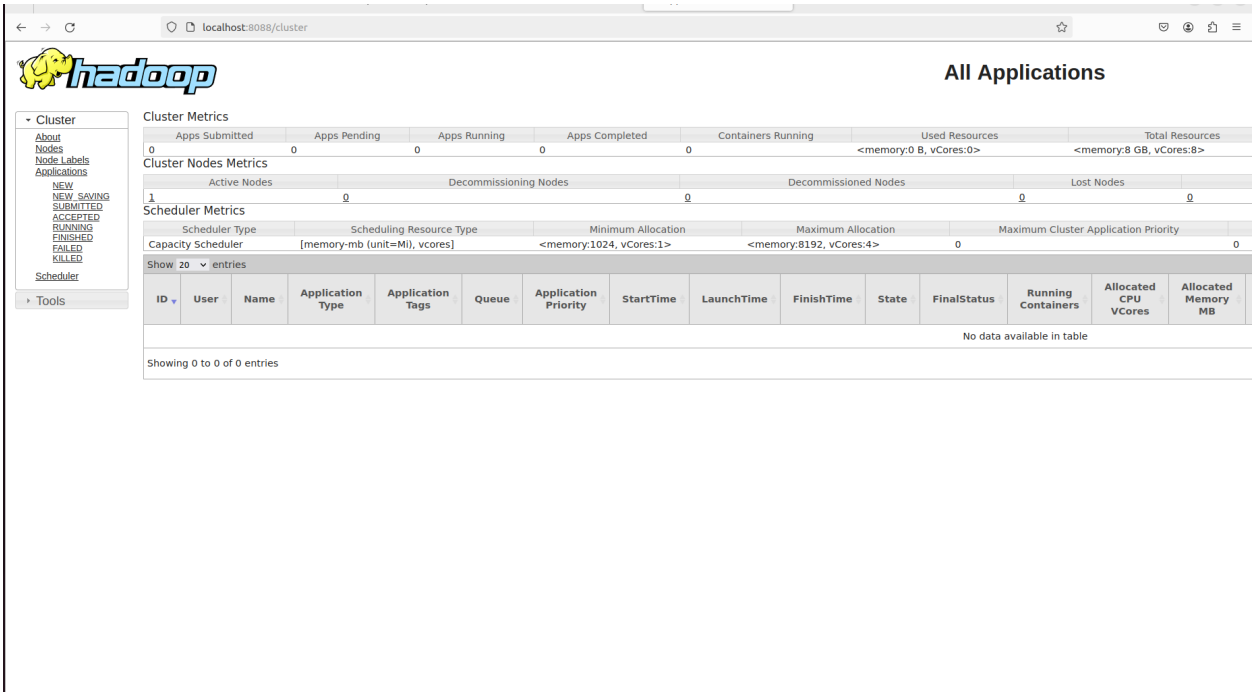


Start ResourceManager daemon and NodeManager daemon

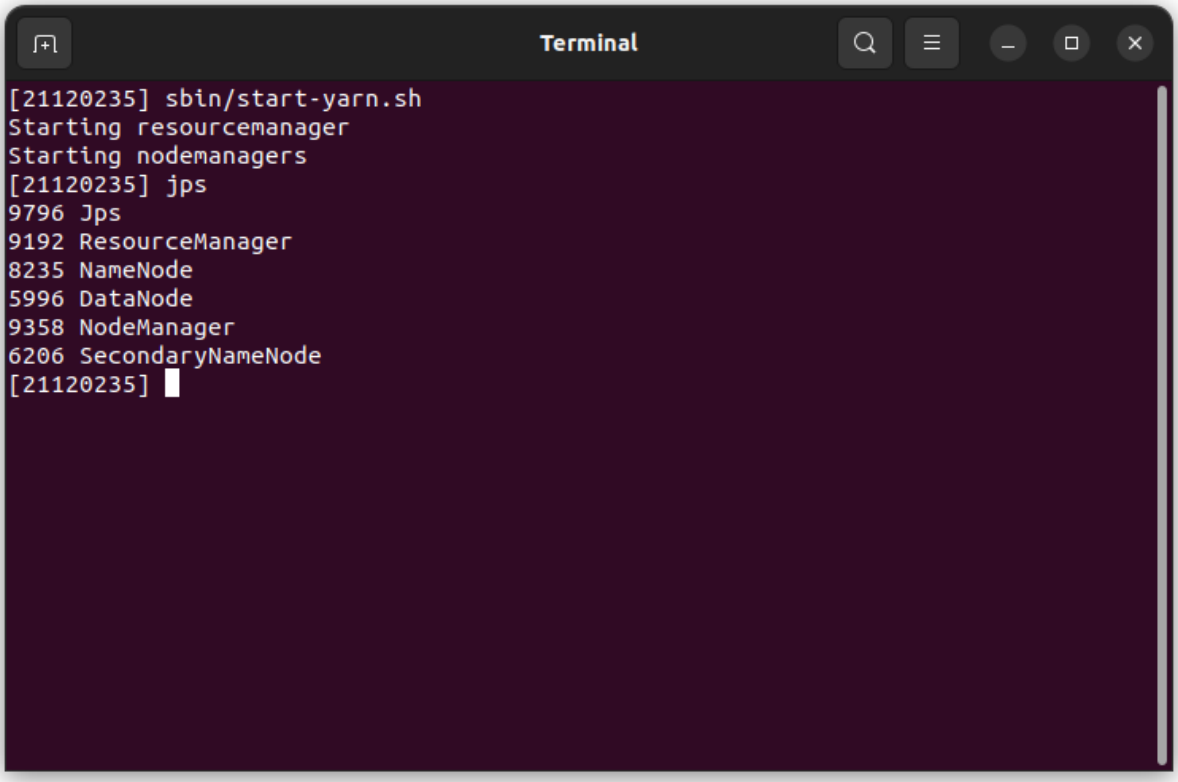
```
sbin/start-yarn.sh
```



Check web interface of ResourceManager at localhost:8088



Run `jps` to check



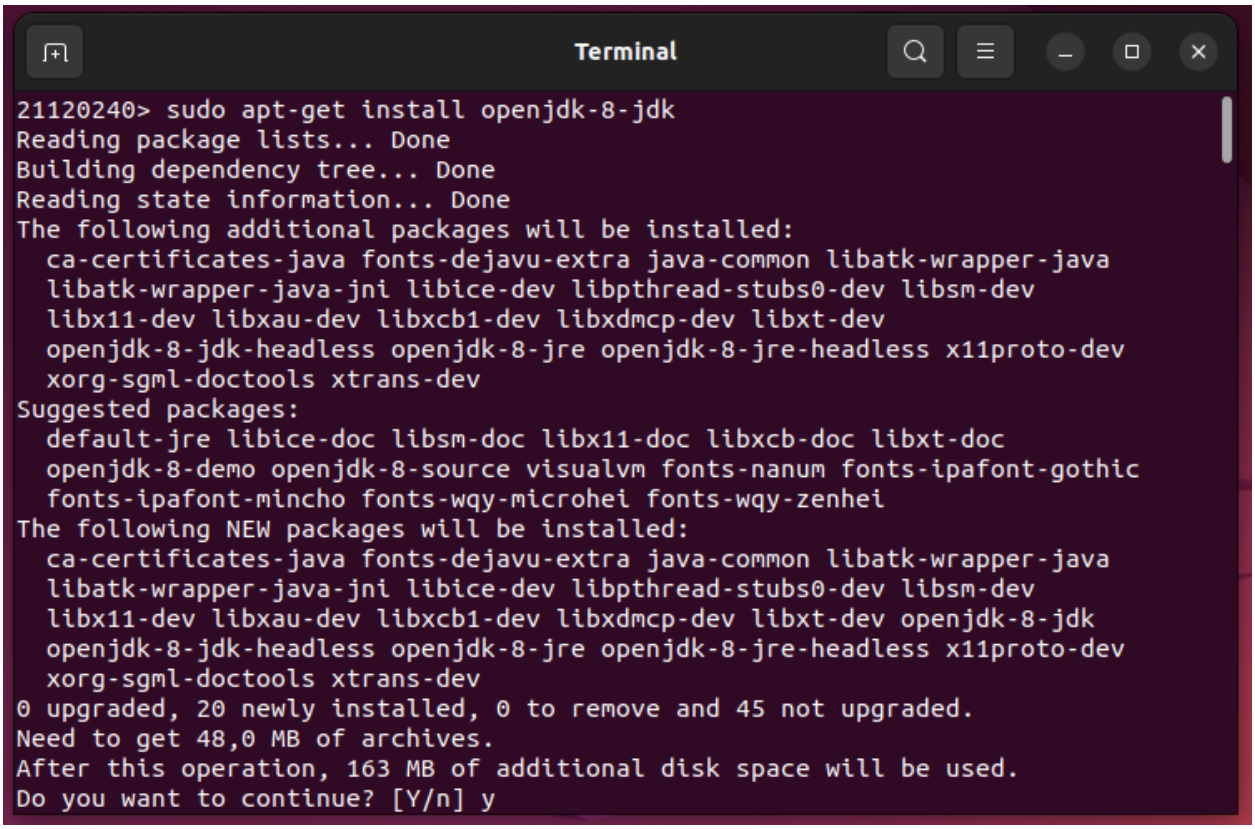
c. Nguyễn Vân Hào - 21120240

Prerequisites

- **Java**

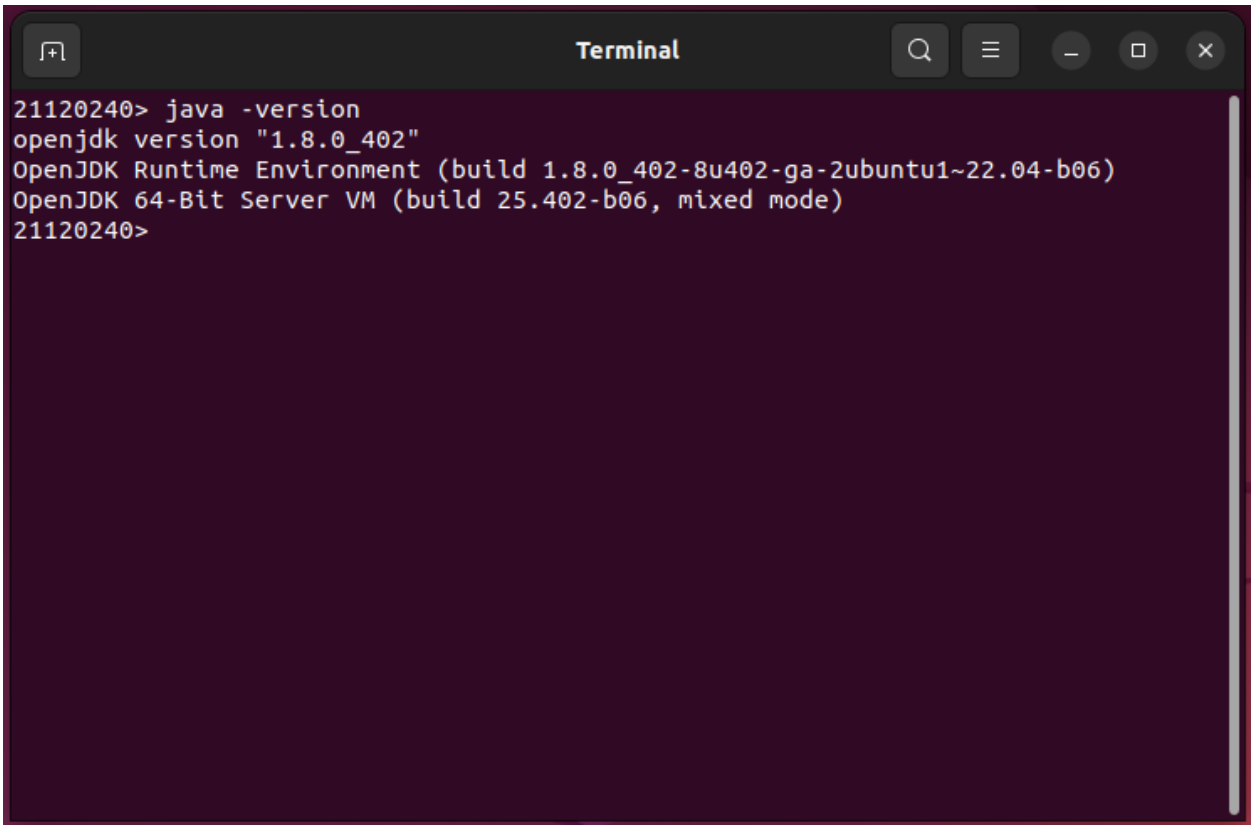
Install Java

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



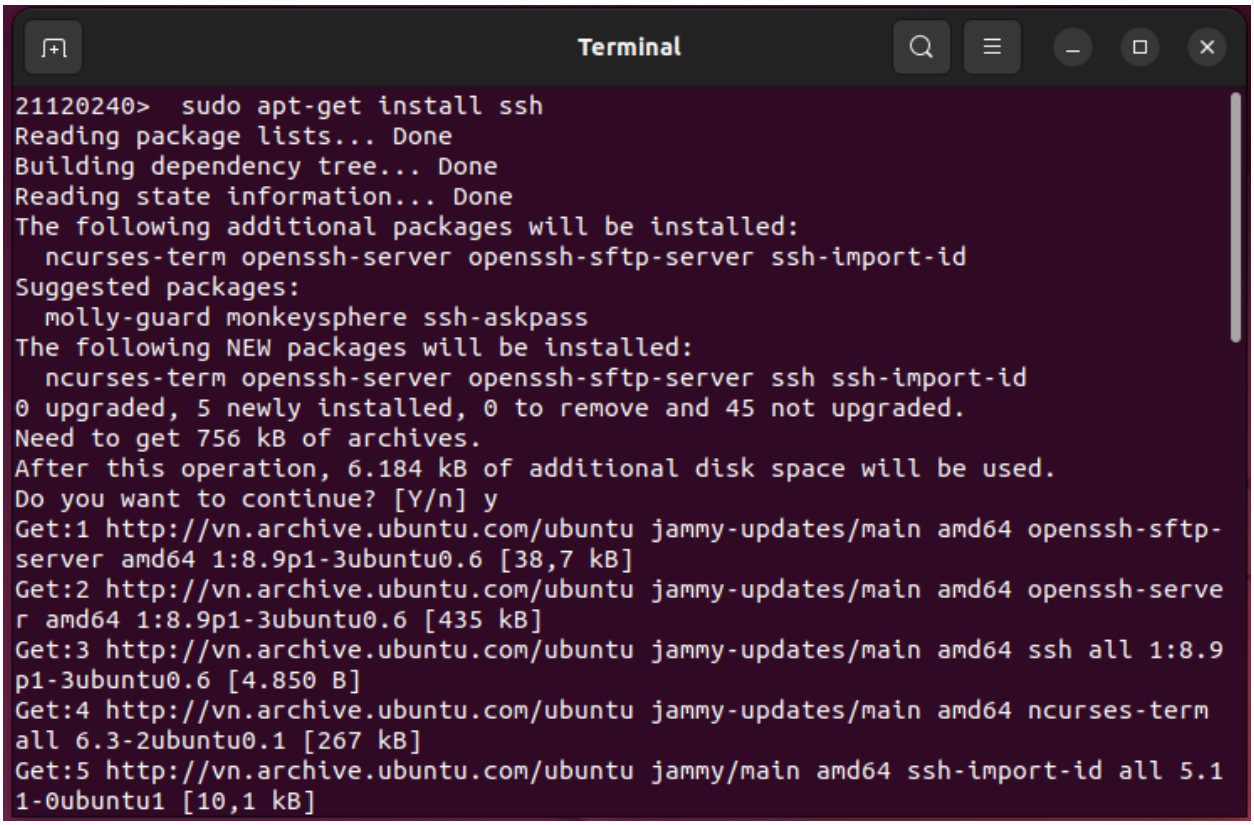
Check if Java was installed successfully

```
java -version
```



- **SSH**

```
sudo apt-get install ssh
```

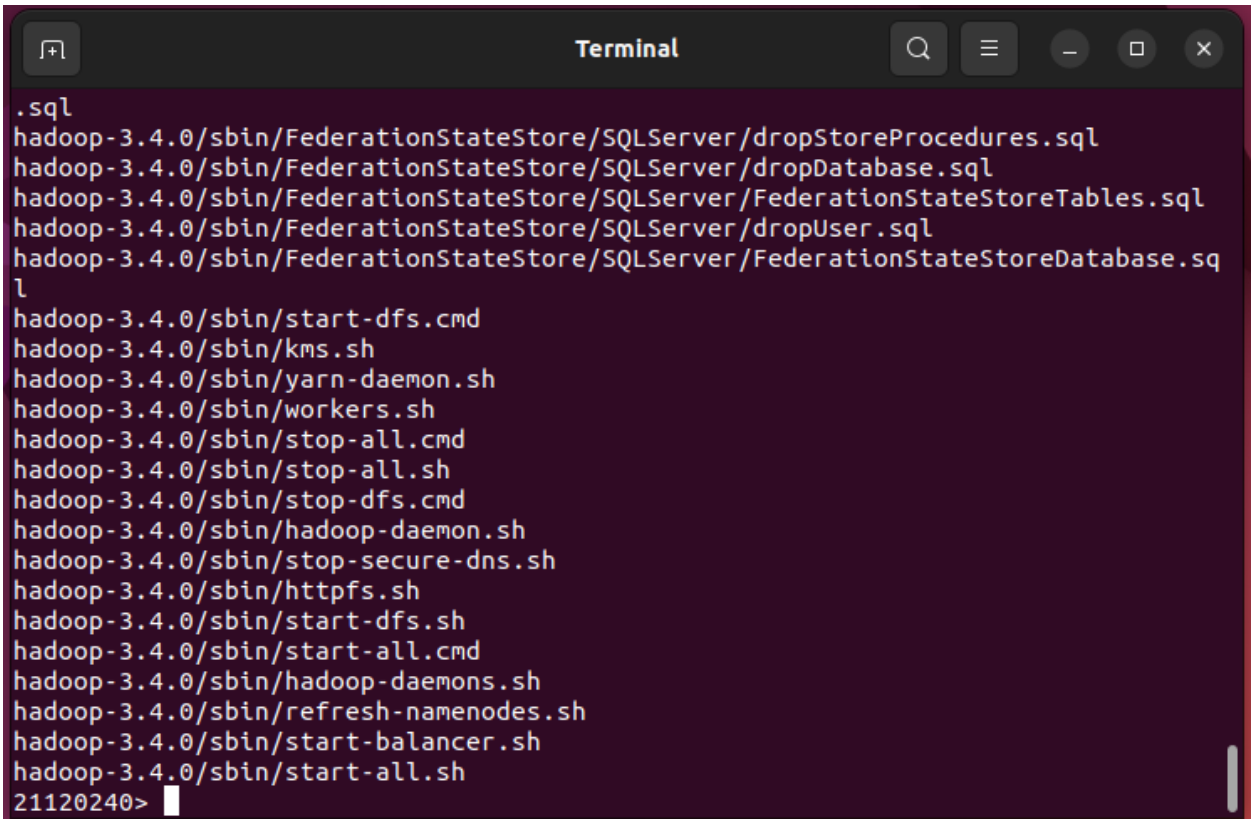


Download Hadoop

Download Hadoop in https://dlcdn.apache.org/hadoop/common/

Extract the file using this command:

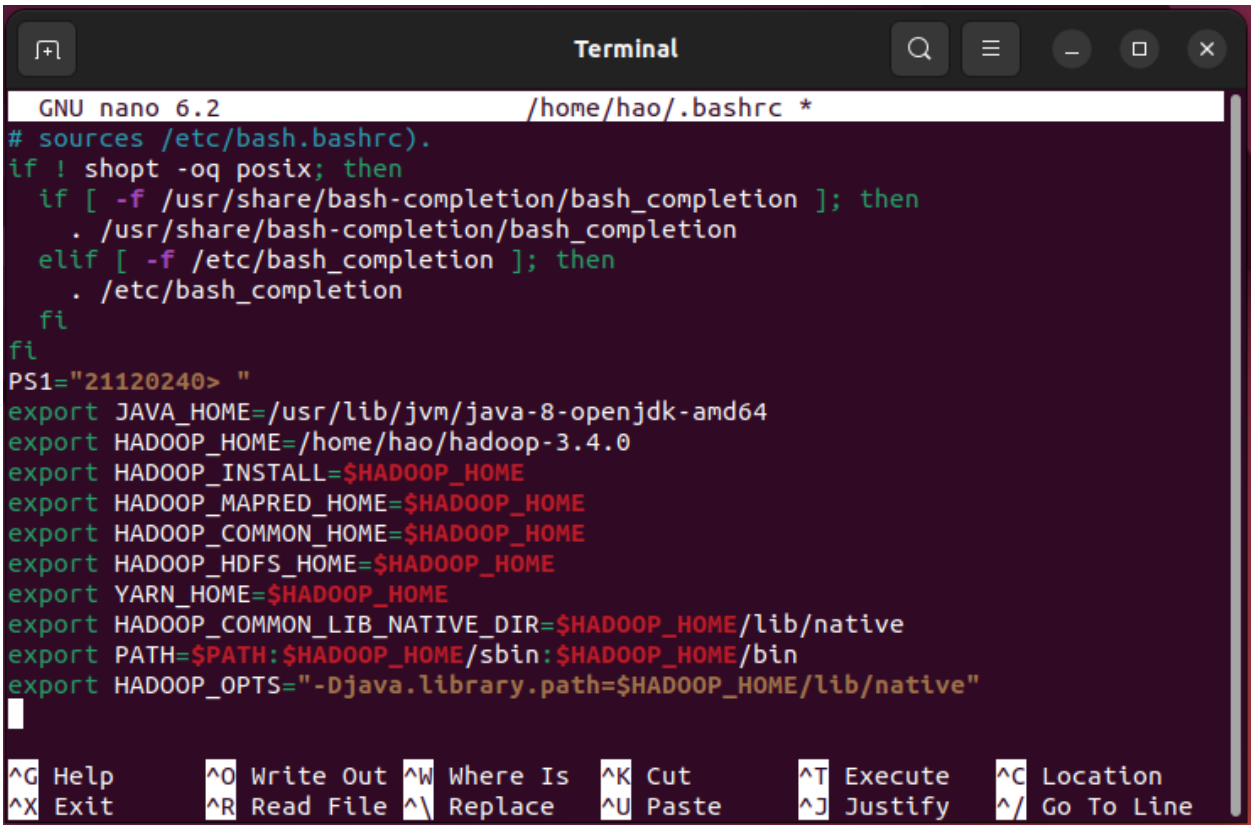
```
tar xvzf hadoop-3.4.0.tar.gz
```



Set up environment variables for Hadoop (file .bashrc)

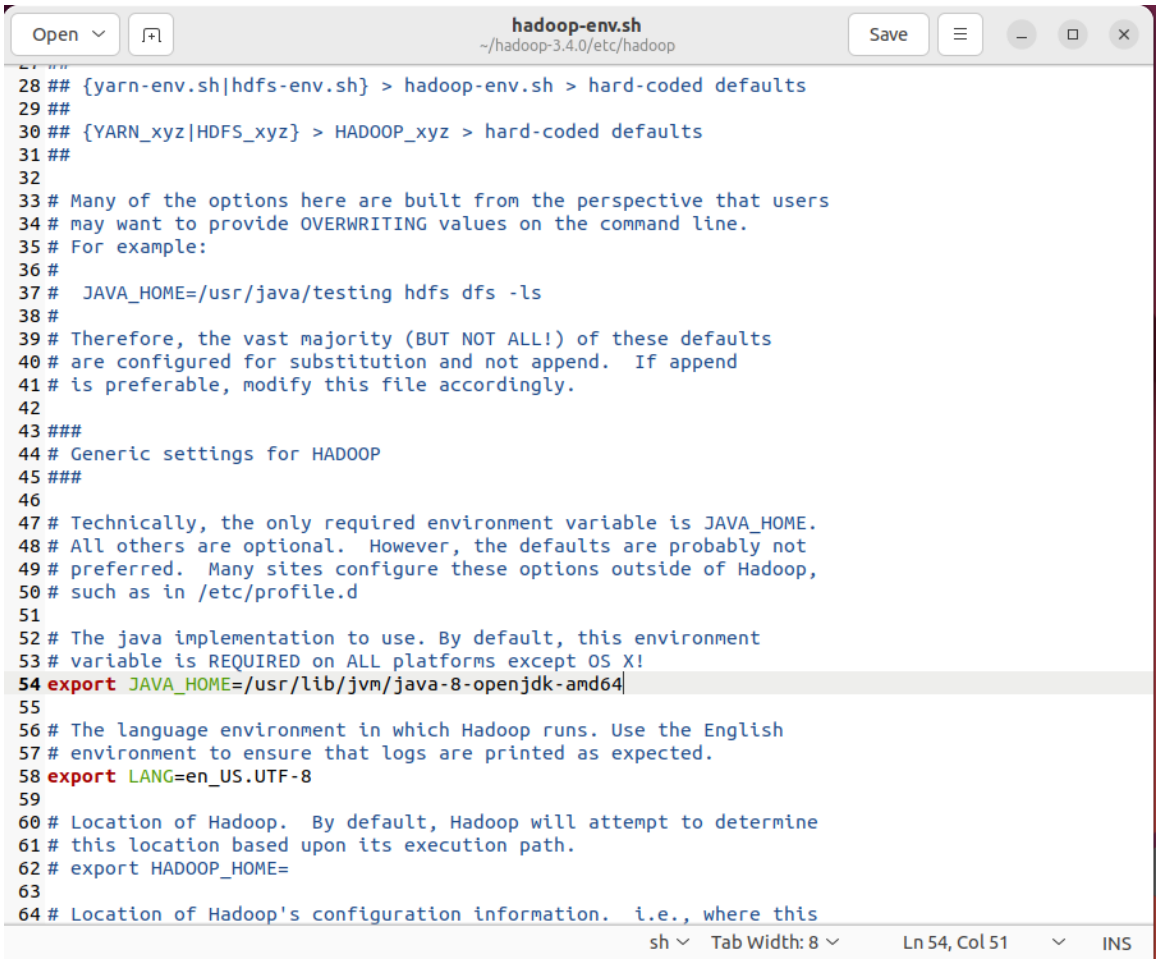
Open file .bashrc and paste these rows:

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HADOOP_HOME=/home/hao/hadoop-3.4.0
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export 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"
```



Set up JAVA_HOME

Edit `export JAVA_HOME` in `hadoop-3.4.0/etc/hadoop/hadoop-env.sh`

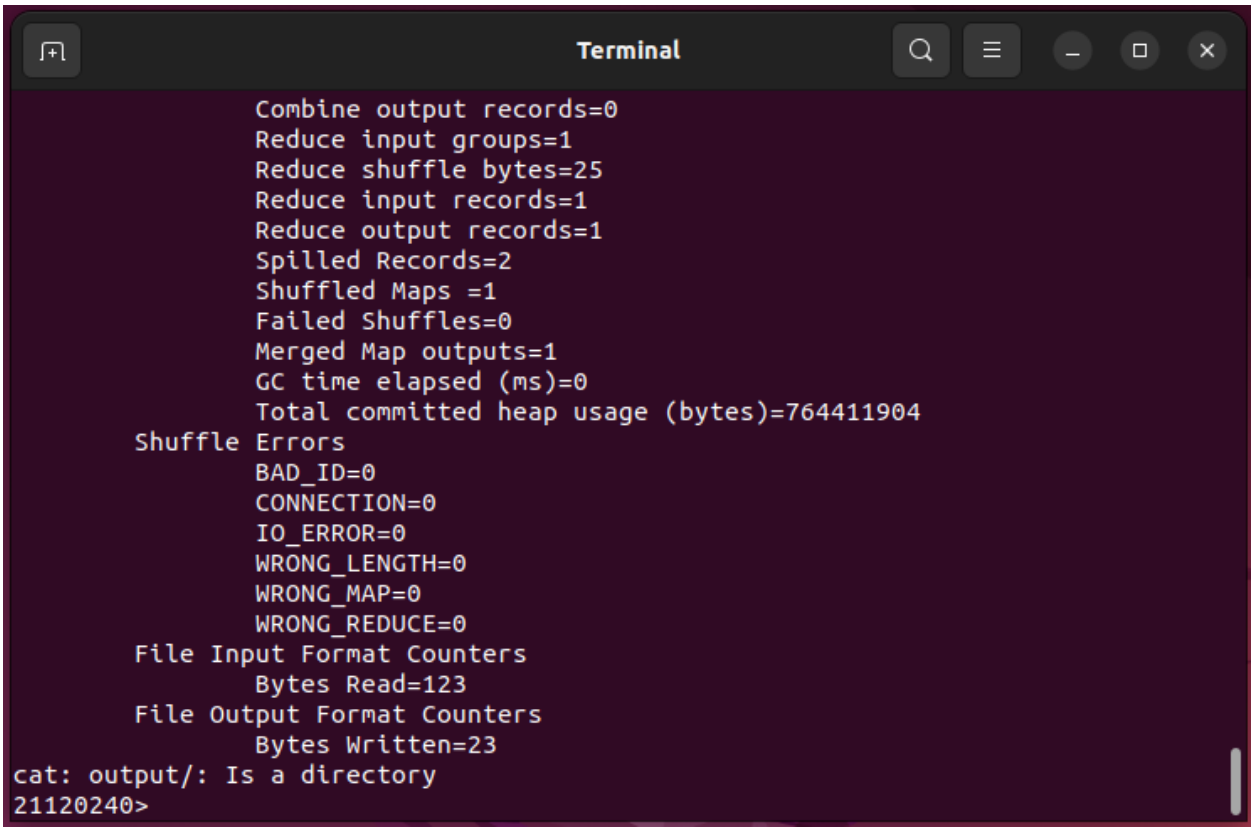


Standalone Operation

Open terminal in `hadoop-3.4.0` and run this command:

```
  mkdir input
cp etc/hadoop/*.xml input
bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-example
cat output/
```

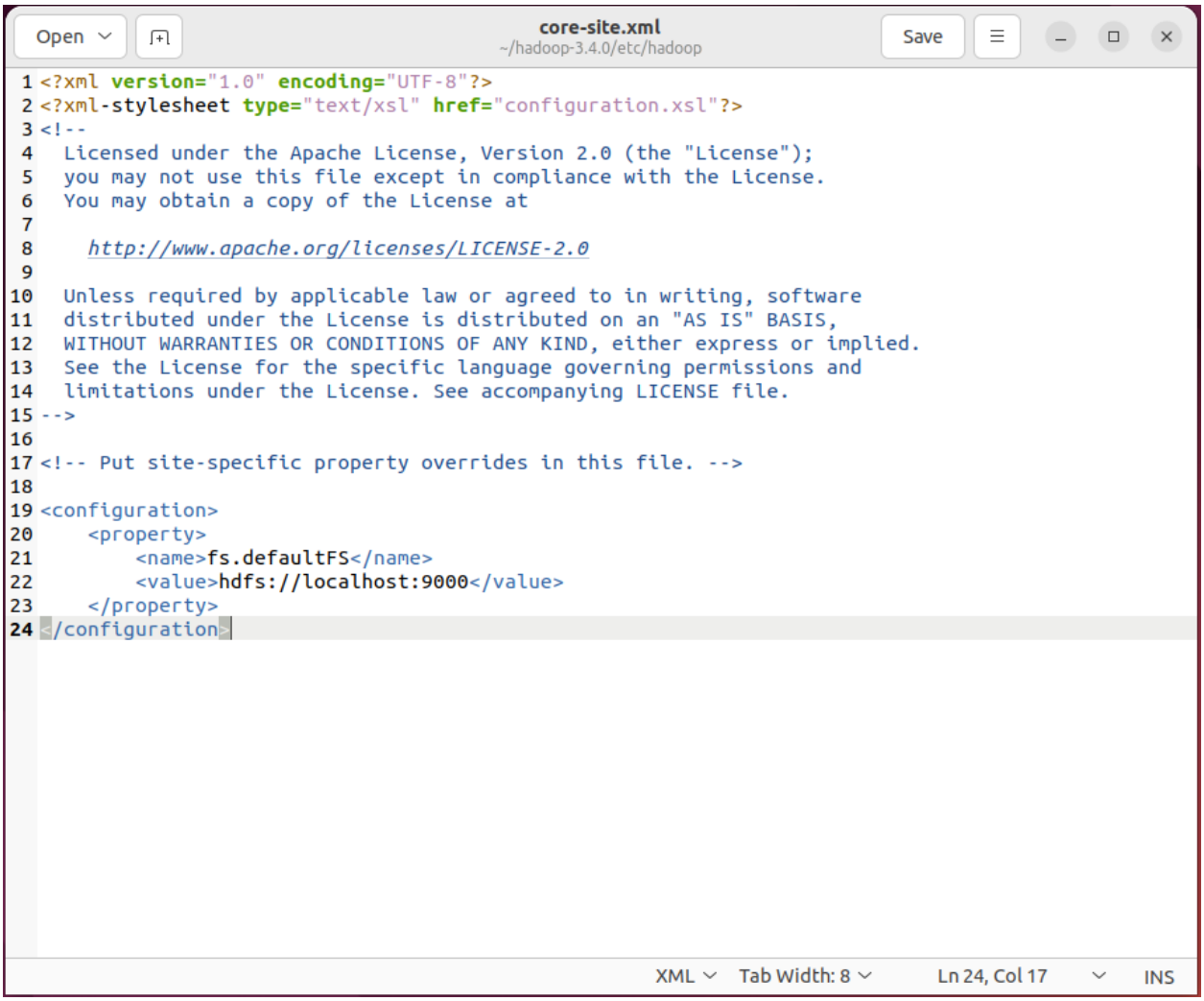
It copies the directory to input folder and then finds and displays every match of the regex 'dfs[a-z.]+' then write to output folder.



Pseudo-Distributed Operation

Config `etc/hadoop/core-site.xml`

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

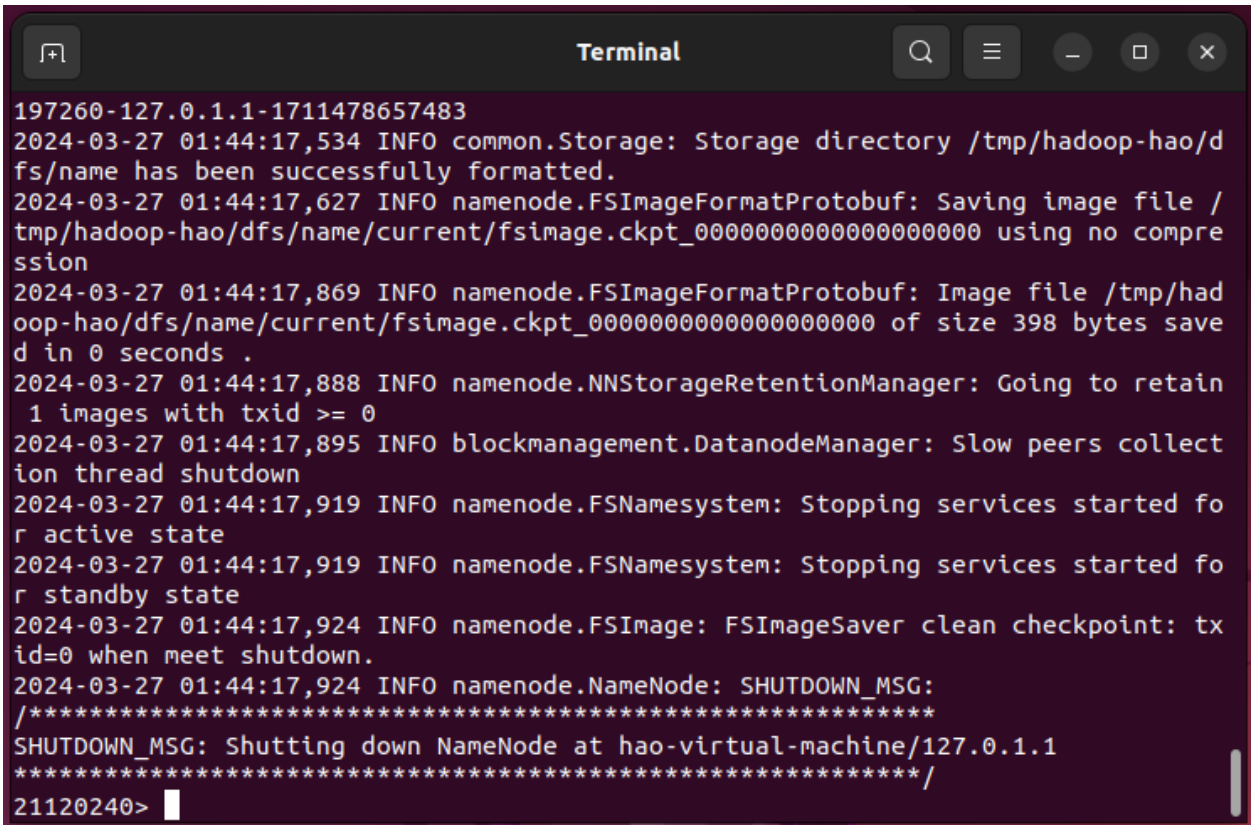


Config `etc/hadoop/hdfs-site.xml`

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

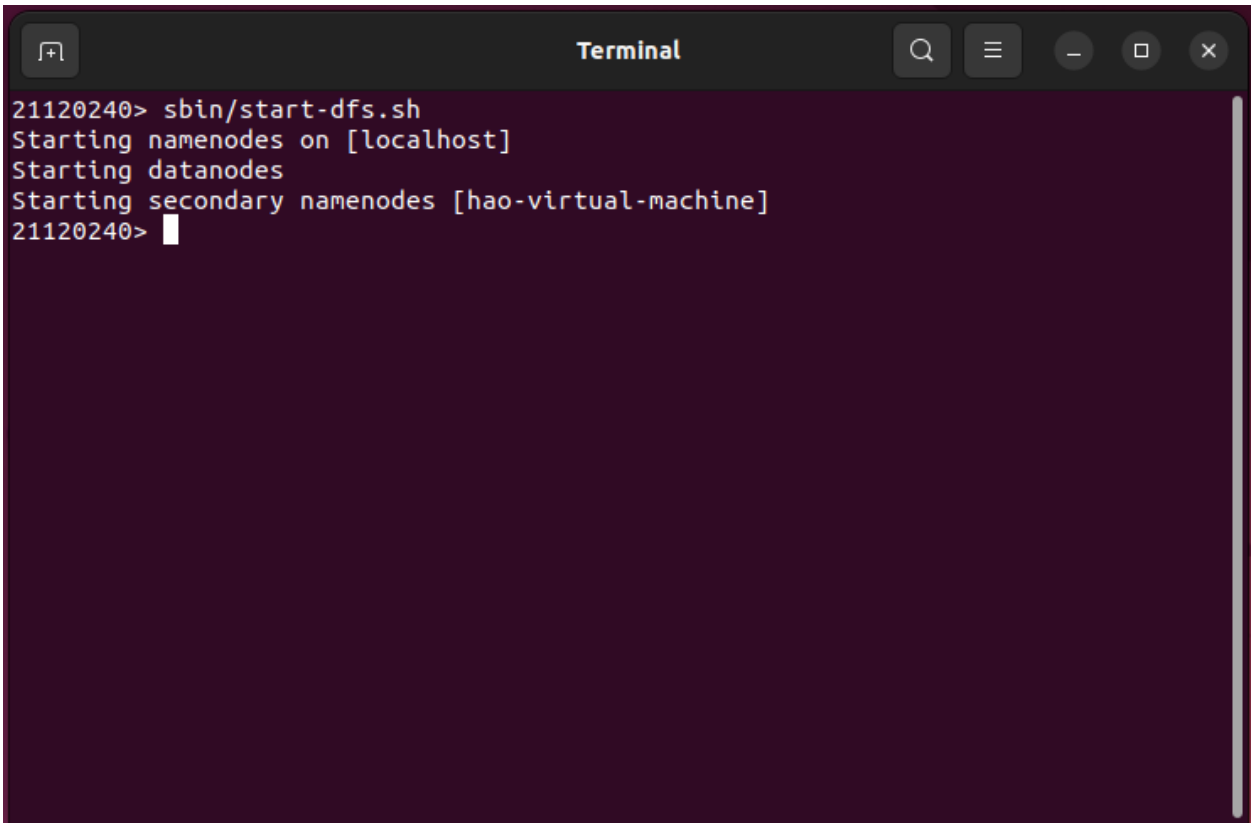
Format filesystem

```
bin/hdfs namenode -format
```

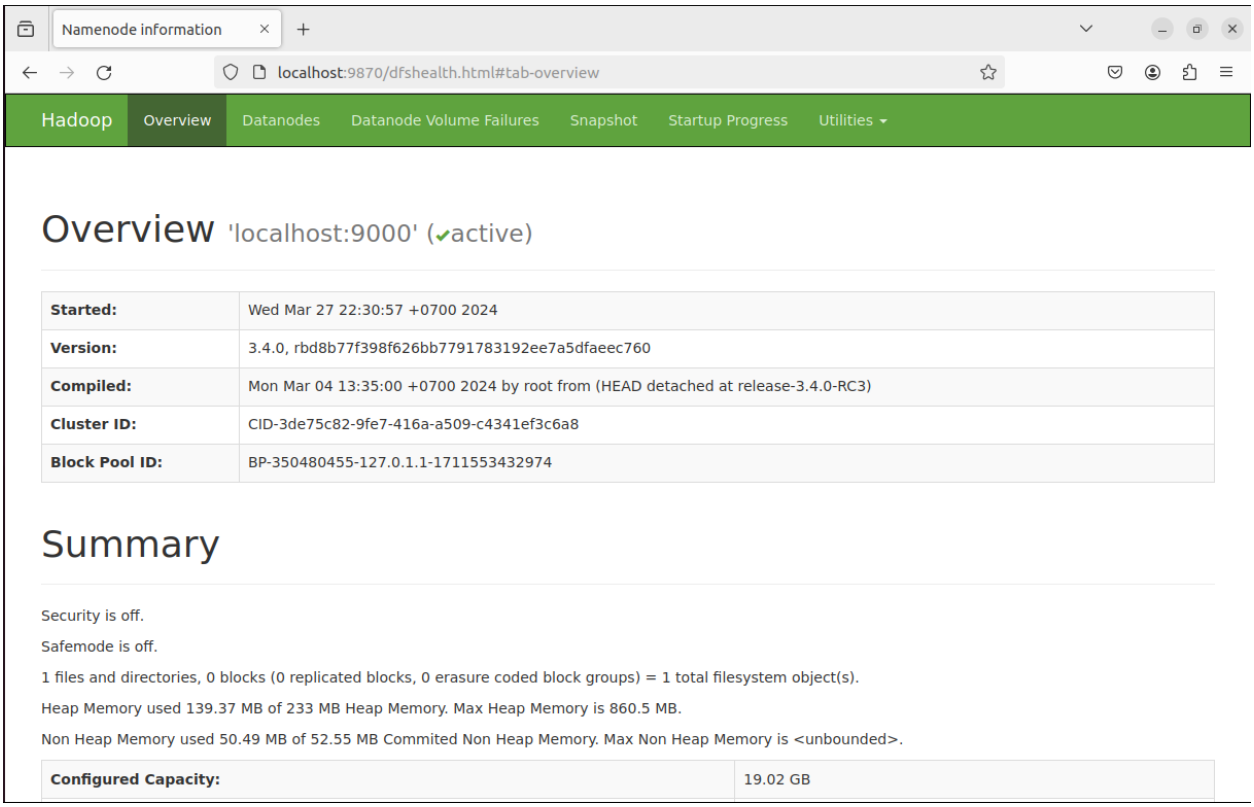


Start NameNode daemon and DataNode daemon

```
sbin/start-dfs.sh
```



Check by accessing localhost:9870

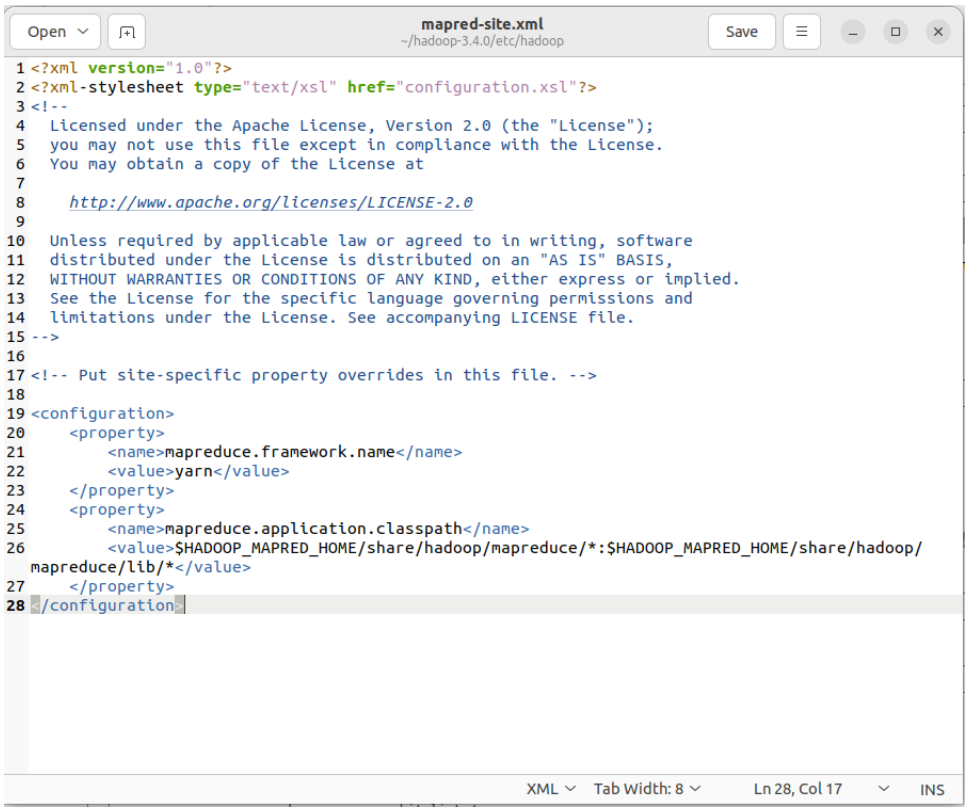


Create directory required to execute MapReduce jobs:

```
bin/hdfs dfs -mkdir -p /user/21120240
```

Config `etc/hadoop/mapred-site.xml`

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



Config `etc/hadoop/yarn-site.xml`

```
<configuration>
    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    </property>
    <property>
        <name>yarn.nodemanager.env-whitelist</name>
        <value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HAI
    </property>
</configuration>
```

Open

yarn-site.xml
~/hadoop-3.4.0/etc/hadoop

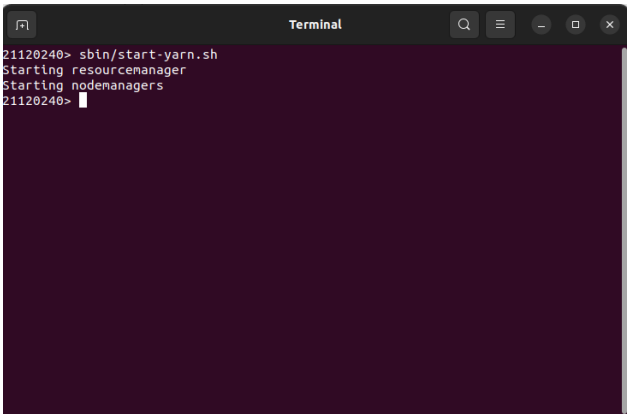
Save

```
<?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>
<!-- Site specific YARN configuration properties -->
    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    </property>
    <property>
        <name>yarn.nodemanager.env-whitelist</name>

<value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF_DIR,CLASSPATH_PREPEND_DISTCACHE,
value>
    </property>
</configuration>
```

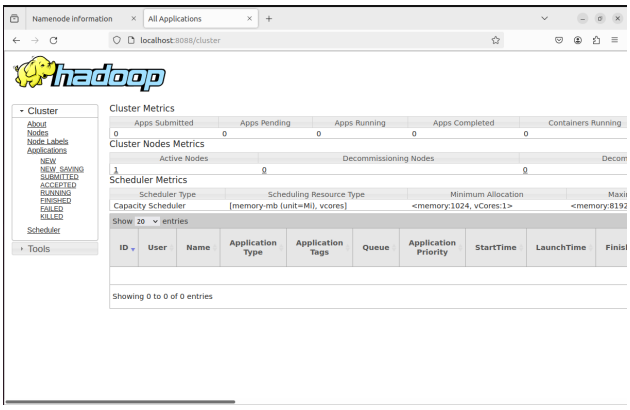
Saving file "/home/hao/hadoop-3.4.0/etc/hadoop/yarn-site.xml"... XML Tab Width: 8 Ln 25, Col 17 INS

Start ResourceManager daemon and NodeManager daemn

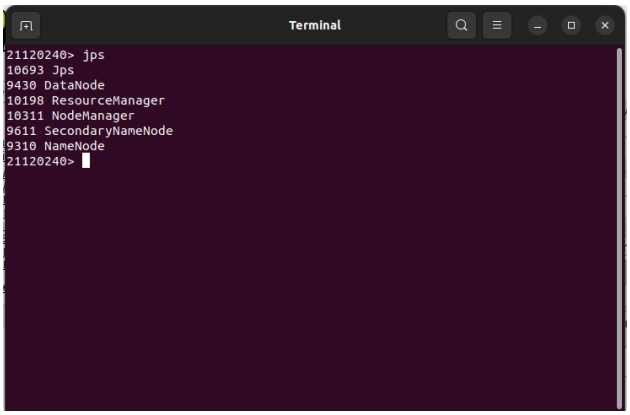
```
sbin/start-yarn.sh
```



Check ResourceManager by accessing localhost:8088



Run `jps` to check



d. Tôn Anh Huy - 21120257

Prerequisites

- **Java**

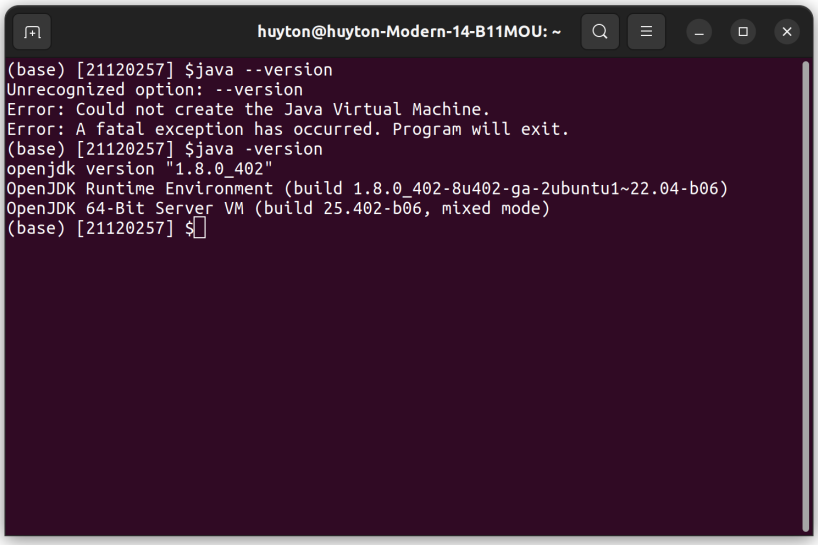
According to suggestions, we should install Java 8 (because Java 11 runtime only)

Type this in terminal to install Java

```
sudo apt install openjdk-8-jdk -y
```

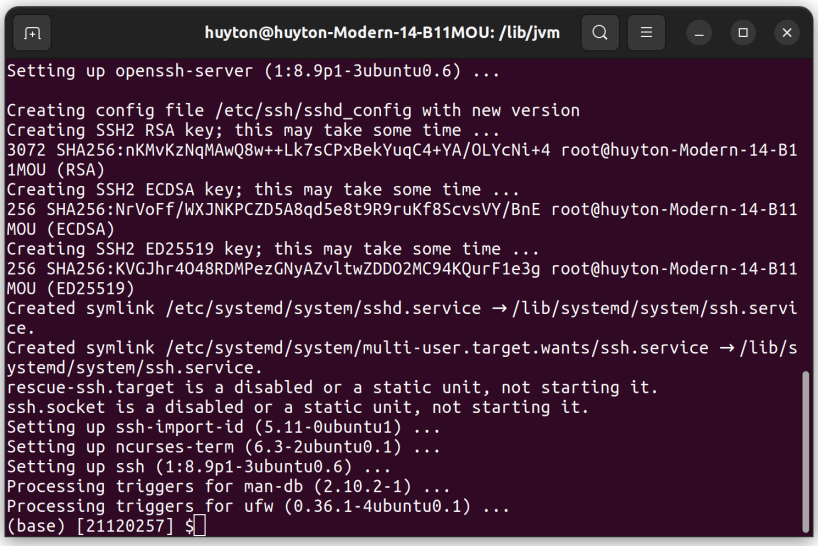
Check if we install successfully

```
java -version
```



- **SSH**

```
sudo apt-get install ssh
```



Download Hadoop

Go to this link: https://hadoop.apache.org/releases.html then download the latest stable version

In directory that containing downloaded file, type this line to extract

```
tar xvzf hadoop-3.4.0.tar.gz
```

After that, in the current directory will has a folder named `hadoop-3.4.0`

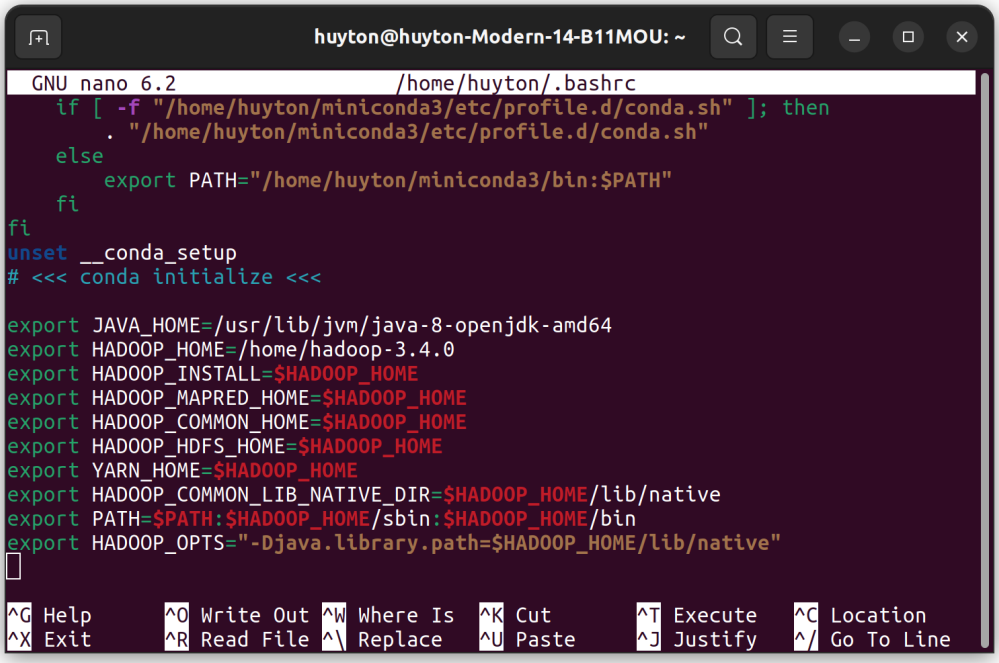
Set up environment variables for Hadoop (file .bashrc)

Open .bashrc file

```
sudo nano ~/.bashrc
```

Paste all lines to .bashrc then save it

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HADOOP_HOME=/home/huyton/hadoop-3.4.0
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export 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"
```



Set up JAVA_HOME

In `hadoop-3.4.0/etc/hadoop/hadoop-env.sh` , find `export JAVA_HOME` line to edit

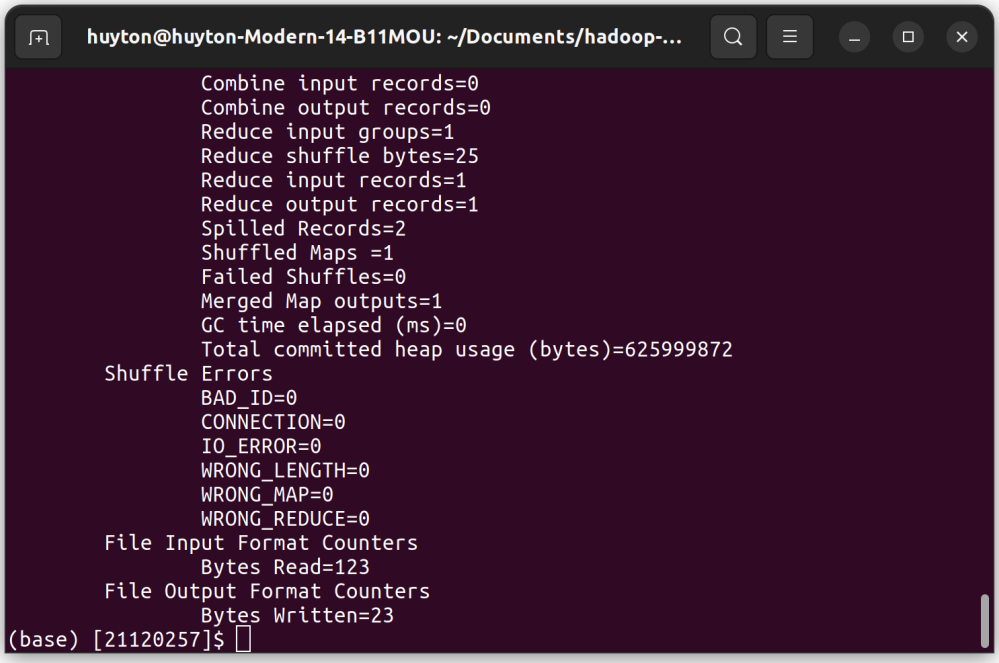
Standalone Operation

By default, Hadoop is configured to run in a non-distributed mode, as a single Java process.

In `hadoop-3.4.0` folder, type these command

```
mkdir input
cp etc/hadoop/*.xml input
bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-example
cat output/
```

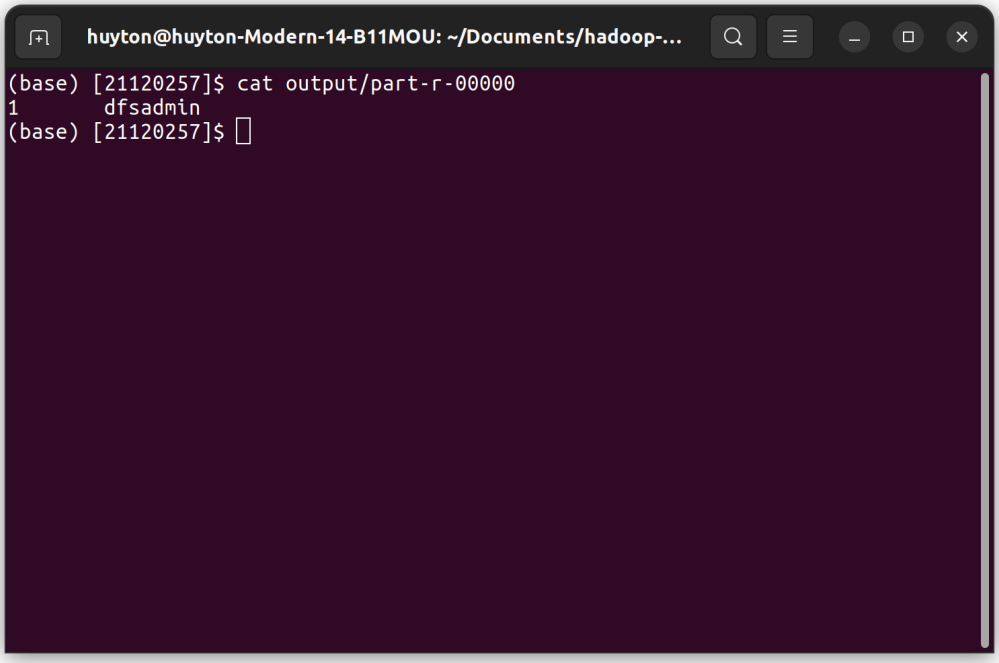
It will copy the unpacked config directory to input folder and then finds and displays every match of the given regex. Output is written to the given output directory.



Check the output content by this command:

```
cat output/part-r-00000
```

Here are what we get:



Pseudo-Distributed Operation

Config `etc/hadoop/core-site.xml`

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

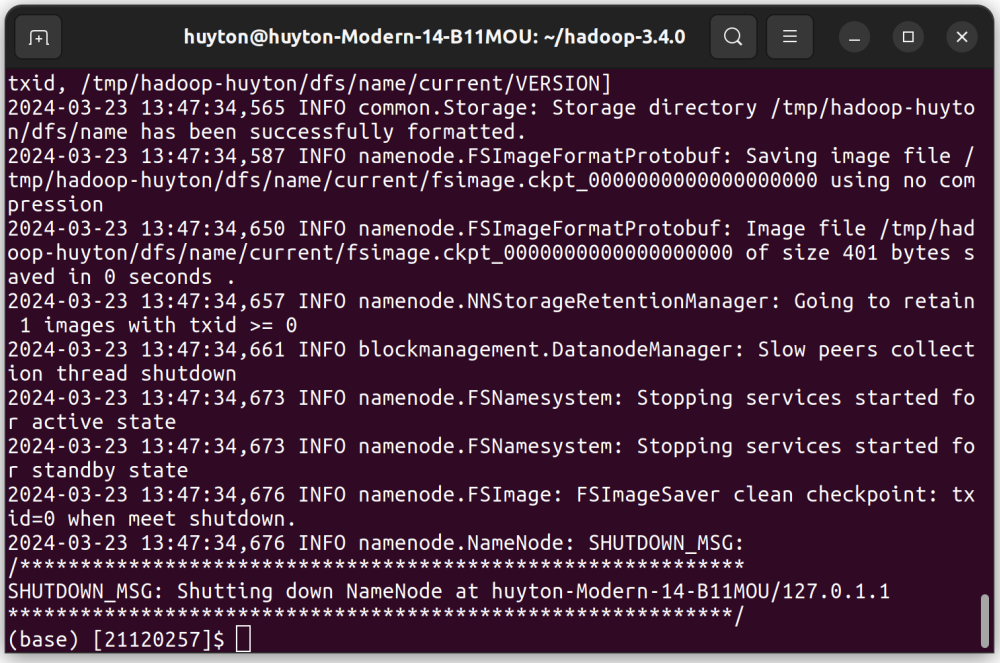
Config `etc/hadoop/hdfs-site.xml`

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

```
    </property>
</configuration>
```

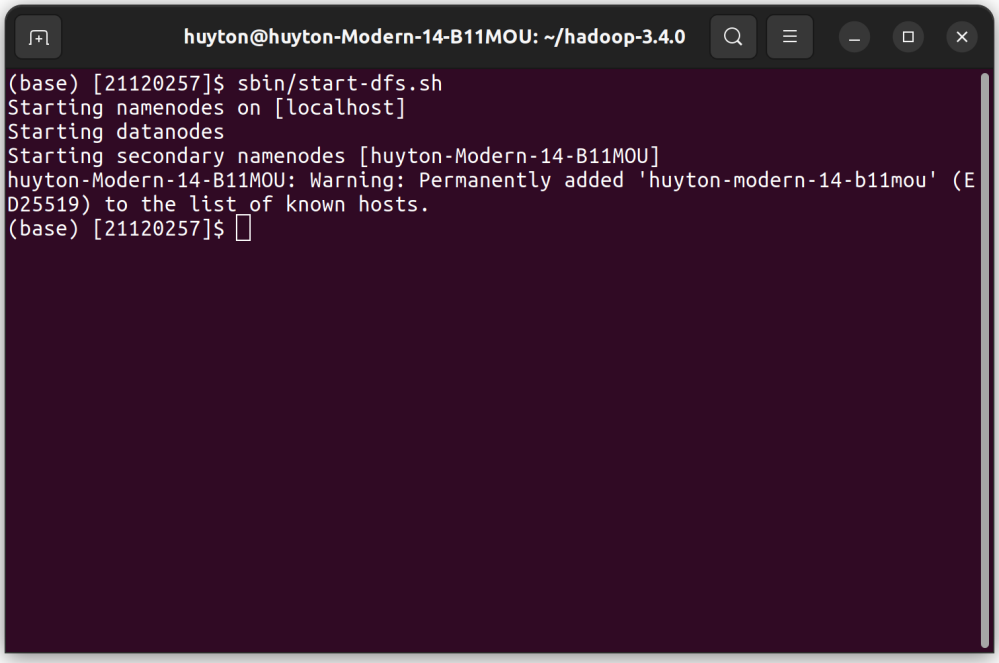
Format filesystem

```
bin/hdfs namenode -format
```

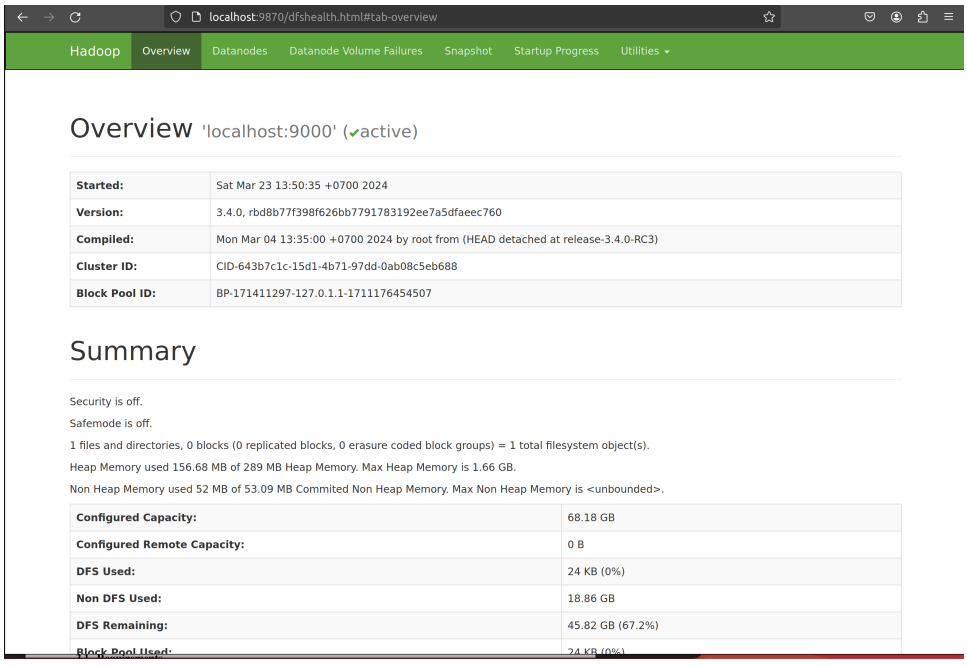


Start NameNode daemon and DataNode daemon

```
  sbin/start-dfs.sh
```



Check web interface of NameNode at localhost:9870



Create directory required to execute MapReduce jobs:

```
bin/hdfs dfs -mkdir -p /user/21120257
```

Config `etc/hadoop/mapred-site.xml`

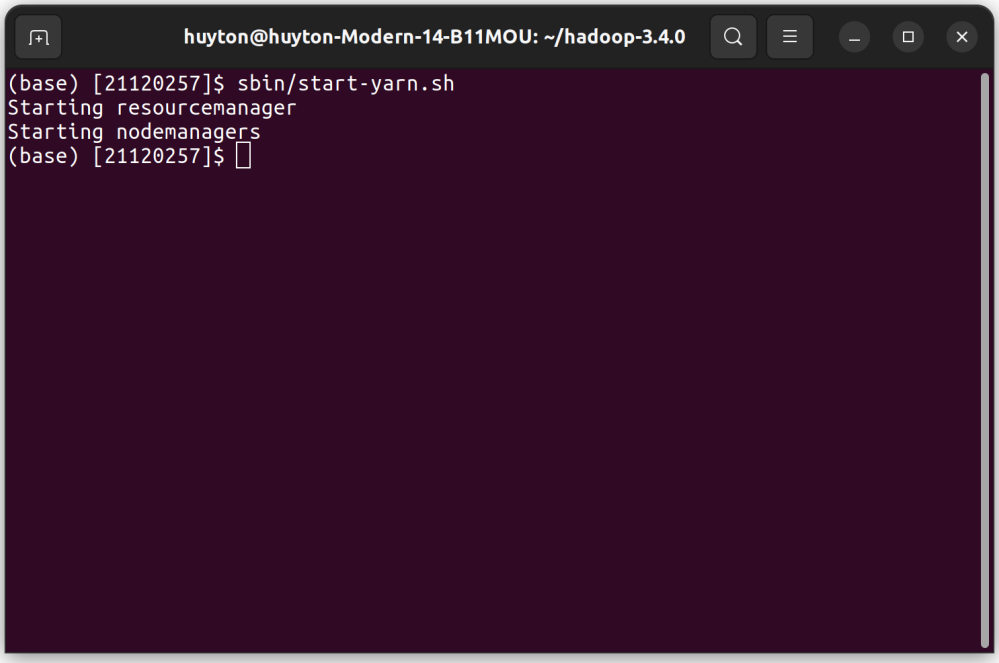
```
<configuration>
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>
    <property>
        <name>mapreduce.application.classpath</name>
        <value>$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/*:$HAI
    </property>
</configuration>
```

Config `etc/hadoop/yarn-site.xml`

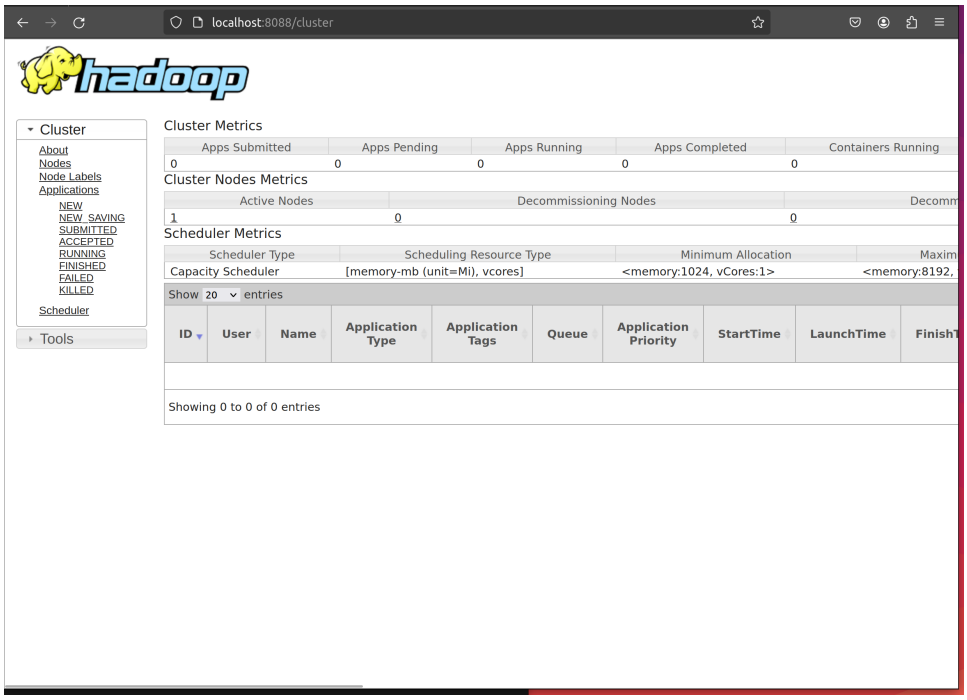
```
<configuration>
    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    </property>
    <property>
        <name>yarn.nodemanager.env-whitelist</name>
        <value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HAI
    </property>
</configuration>
```

Start ResourceManager daemon and NodeManager daemn

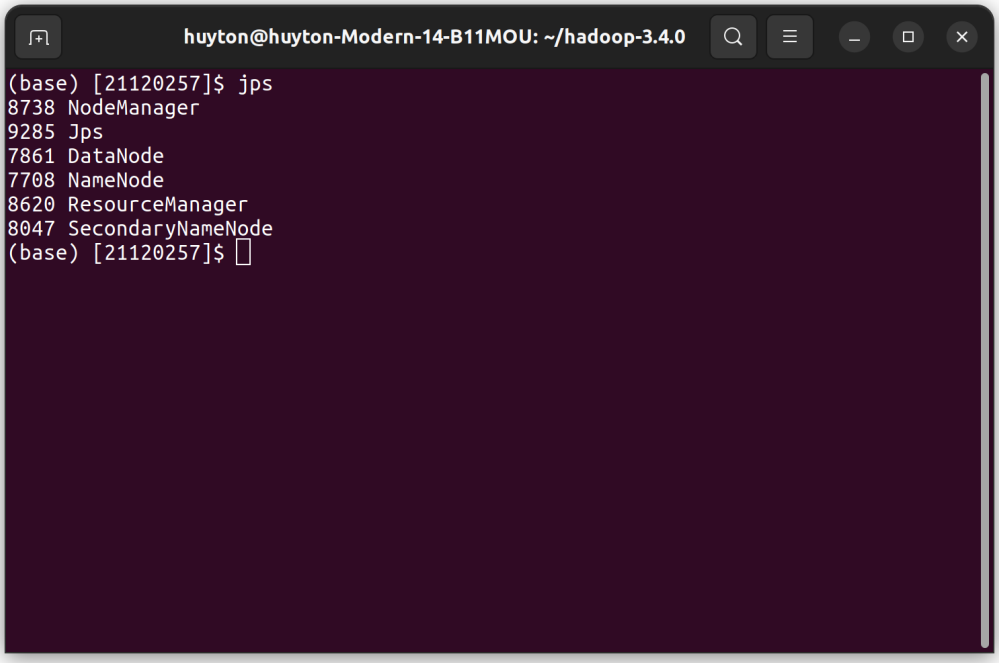
```
sbin/start-yarn.sh
```



Check web interface of ResourceManager at localhost:8088



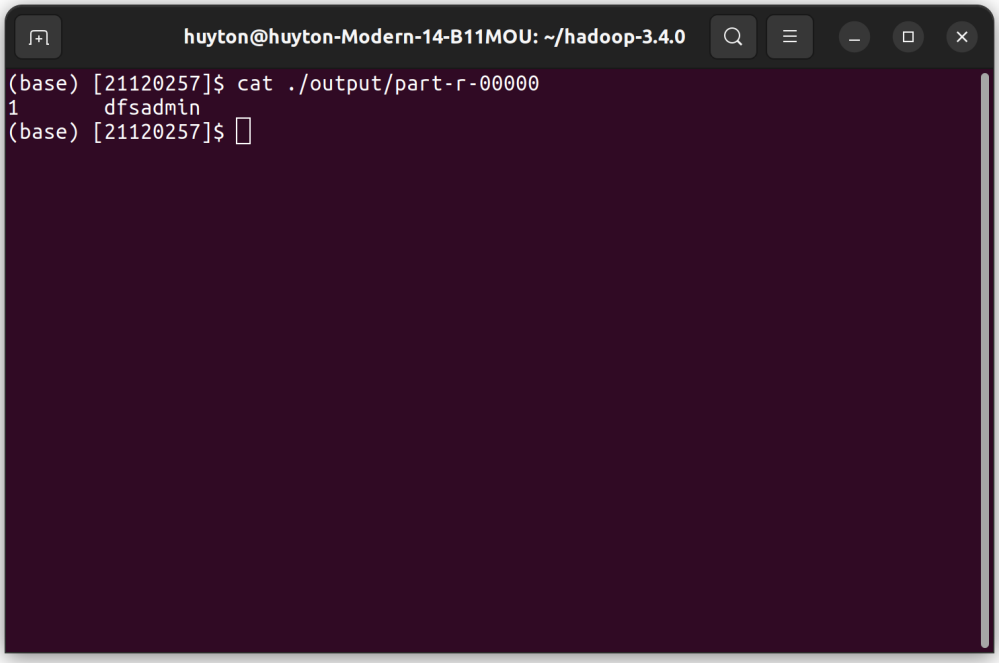
Run `jps` to check



Run mapreduce task like above by:

- Copy input files from local to hdfs
- Run mapreduce on hdfs
- Copy output to local

Result:



2. Introduction to MapReduce (2 points)

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

- In Map funtion:
  • Map takes an input pair and produces a set of intermediate key-value pairs.
  • These intermediate pairs serve as temporary storage before reaching the reducer.

- In Reduce function:
  • The reducer receives an intermediate key “I” and a set of values associated with that key.
  • The reducer processes these grouped pairs and merges together the values for key “I” to form a possibly smaller set of values.
  • The final output key-value pairs are generated by the reducer.

In summary, the input keys-values represent the initial data, intermediate keys-values capture temporary results during computation, and output keys-values represent the final outcome of the MapReduce process

How does MapReduce deal with node failures?

The master pings workers (map nodes and reduce nodes) periodically to make sure that they are in a good condition. If master does not receive any responses from workers then it will mark them as failed.

- Map node

Completed map tasks must be re-executed whenever its node failed. Because their output (intermediate output) is stored in local machine that result in inaccessible problem.

Whenever a map node failed, it will transfer the current task to another node to execute again. Besides, all reduce nodes must be notified that there is a re-execution.

- Reduce node

Once reduce node complete its execution and produces output, that output will be stored in a global file system (such as HDFS). Therefore, there is no need to re-execute the task because output now is accessible.

- Master node

The master node write periodically checkpoints of state of nodes, location... . If it failed, the system can restore its functionality by starting a new copy from the last checkpoint.

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

The “locality” term consider the location between input file and machine that execute map task.

There are some replicas of each data which stored in different machines.

The master node will manage the location information of input file and try to schedule a map task on a machine that already has data. If it meet failure, it will try

to schedule this task on other machines that are near input file.

This strategy minimizes data transfer over the network because data can be processed locally on the same machine where already has input data.

Moreover, it also help to reduce network bandwidth usage which lead to less execution time and enhance performance

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

In map tasks, the repetition in intermediate keys usually exists

In WordCount example, we know that words frequencies tend to follow Zipf distribution (word has rank 1 will occur approximately twice as the word rank 2). Therefore, we produced a lot of same records which lead to sending a significant data across the network (because intermediate output from Map machines are sent to Reduce machines).

With combiner function, we reduce the amount of data above by performing local merging in each map machine before sending them across the network. The code used to implement combiner function and reduce function usually are the same.

The advantages of applying combiner function in MapReduce model:

- Network congestion problem: because the amount of data are significantly smaller
- Data transfer costs problem: because combiner function try to minimize the data by combining repetitions so that the size of data is smaller
- Speed up the MapReduce job: the faster data transfer and decrease network congestion help enhance the performance considerably

3. Running a warm-up problem: Word Count (2 points)

3.1 Create source code files

Firstly, we must create `WordCount.java` using the provided source code from the tutorial

Apache Hadoop 3.4.0 – MapReduce Tutorial

This document comprehensively describes all user-facing facets of the Hadoop MapReduce framework and serves as a tutorial.

https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html#Purpose

3.2 Compile `WordCount.java` and create a jar

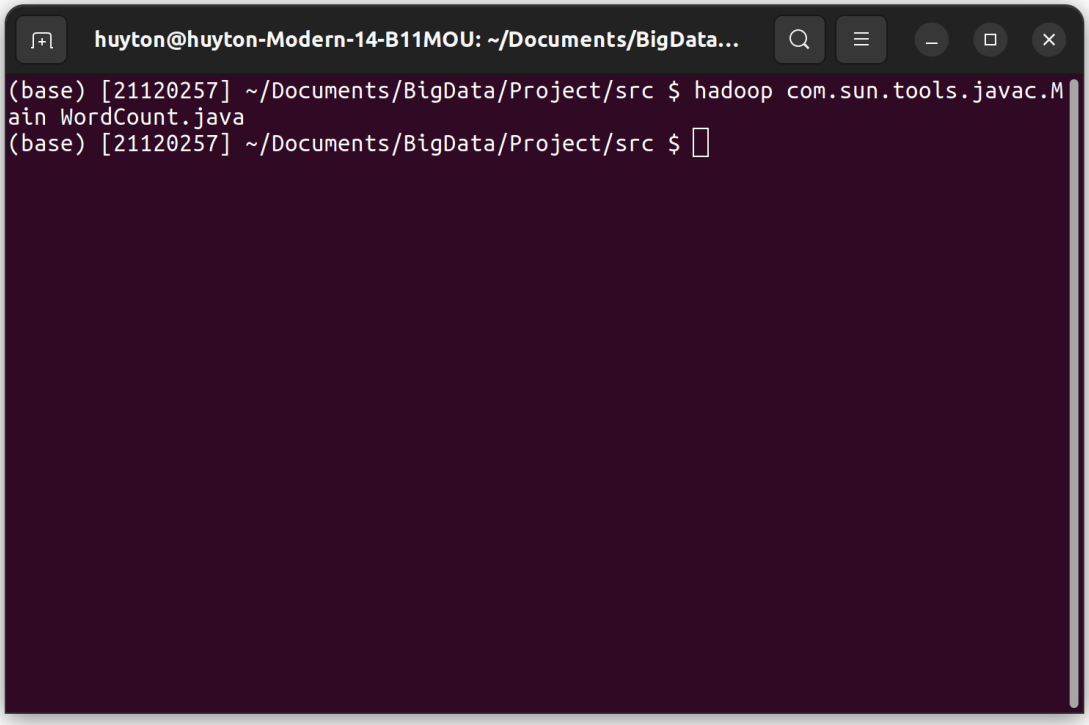
It may raise an error if we do not add the following line into `.bashrc`

```
export HADOOP_CLASSPATH=${JAVA_HOME}/lib/tools.jar
```

This line used to add the `tools.jar` file from Java installation to the Hadoop classpath.

However, we can add it only for the current session instead of writing in `.bashrc` as well

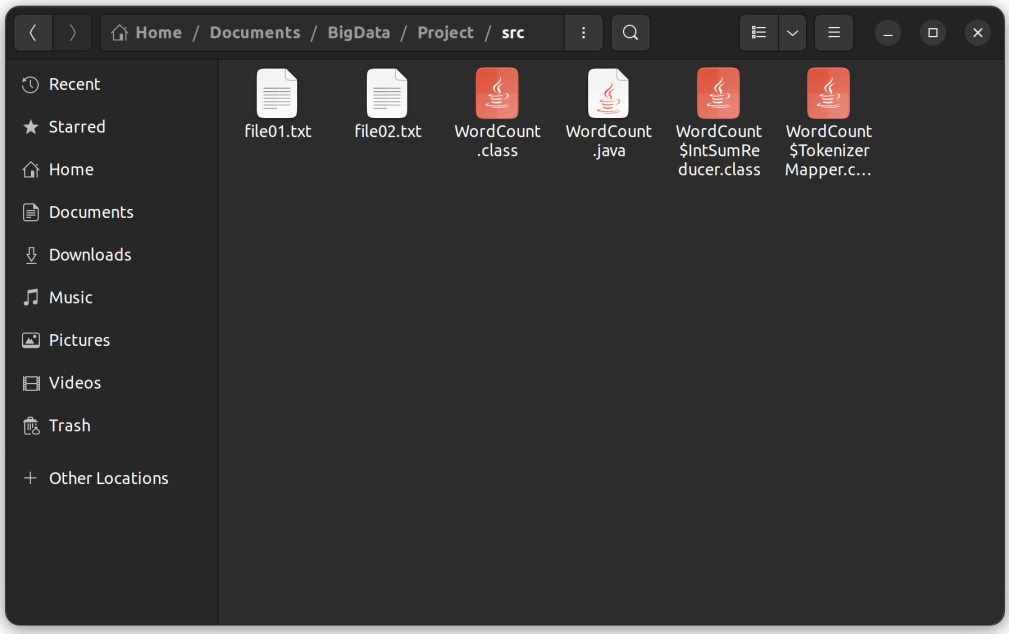
```
$ bin/hadoop com.sun.tools.javac.Main WordCount.java
```



The above command will use Hadoop's bundled Java compilers to compile our file `WordCount.java`

`Com.sun.tools.javac.Main` is the main class in the compiler

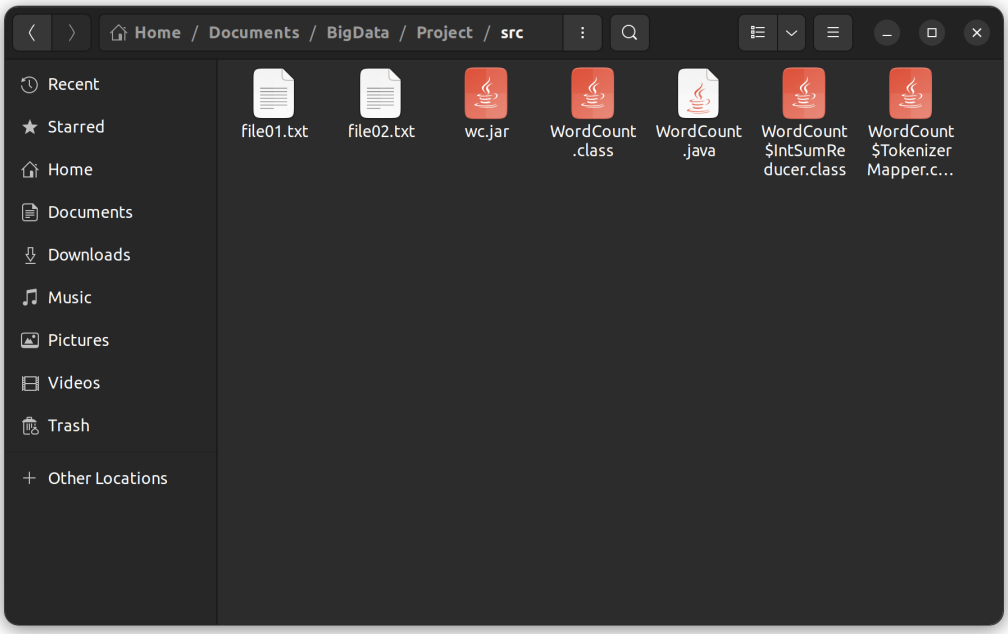
After the compiling process, we should reiceve `.class` files as below:




```
$ jar cf wc.jar WordCount*.class
```

This command will create a .jar (Java Archive) file then named it `wc.jar` in current folder by using all `.class` files whose name starts with `WordCount`

Once it finish, we must get a file `wc.jar` as below



3.3 Create input files and input folders

In the current working directory, we will create 2 files named `file01.txt` and `file02.txt` that are considered as input files

Data in `file01.txt`

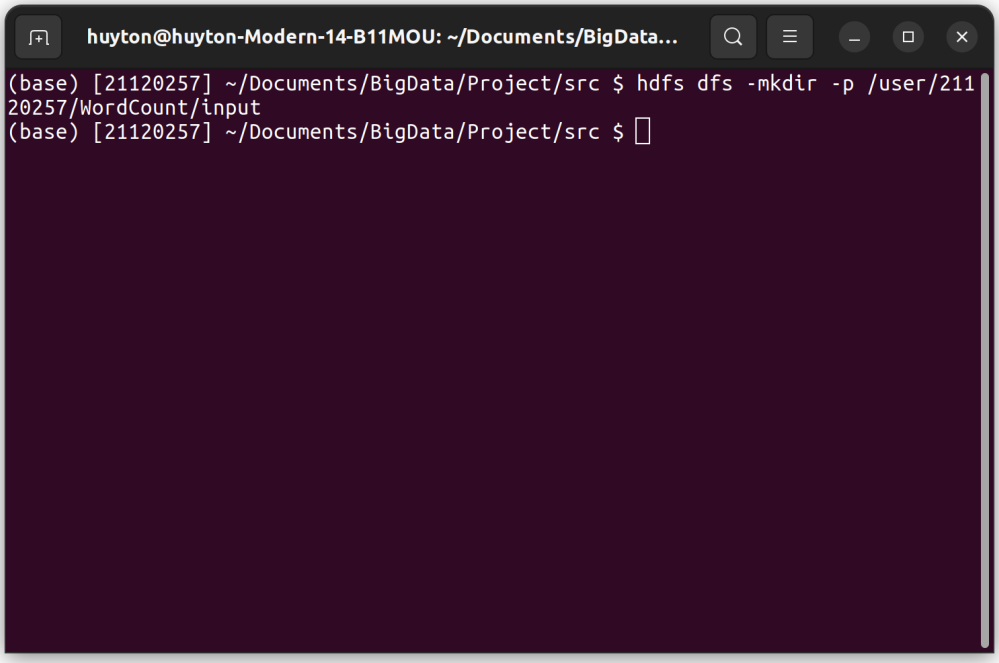
```
Hello World Bye World
```

Data in `file02.txt`

```
Hello Hadoop Goodbye Hadoop
```

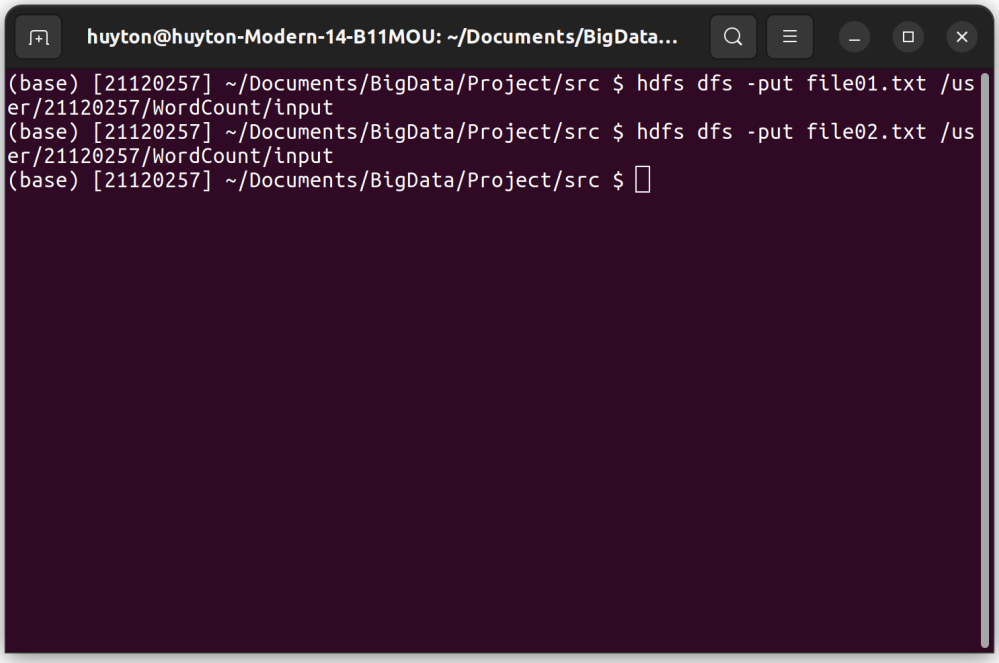
Next, we will create input folder in hdfs

```
hdfs dfs -mkdir -p /user/21120257/WordCount/input
```



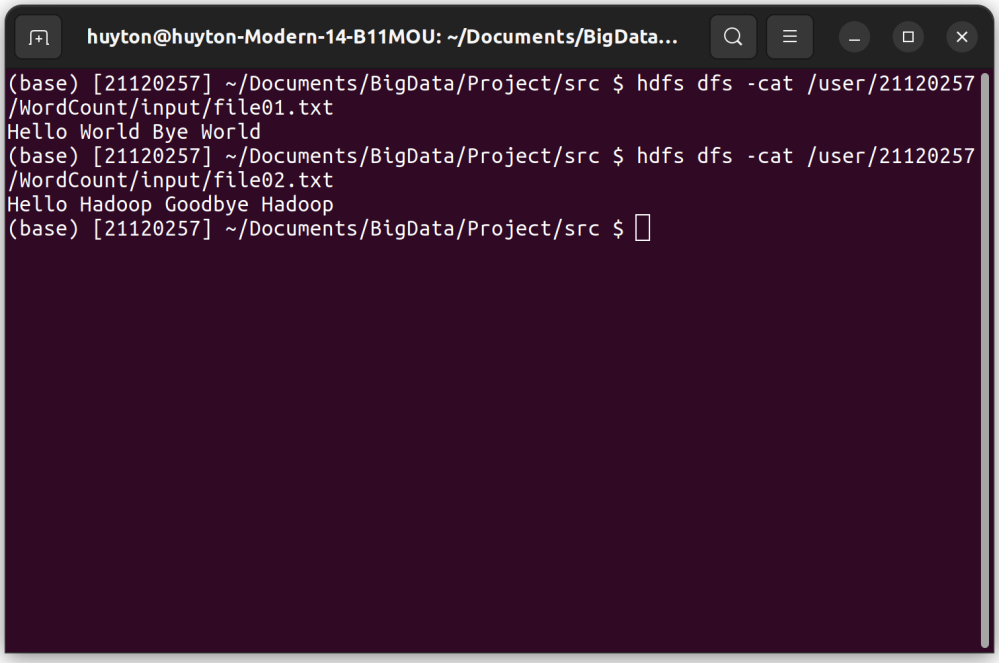
Now, we will copy input file that we have just created before to hdfs

```
hdfs dfs -put file01.txt /user/21120257/WordCount/input
hdfs dfs -put file02.txt /user/21120257/WordCount/input
```



We will check to make sure that the input files are ready

```
hdfs dfs -cat /user/21120257/input/WordCount/file01.txt
hdfs dfs -cat /user/21120257/input/WordCount/file02.txt
```



Now, the input files are ready for MapReduce

3.4 Run the application

To run the WordCount we run the following command

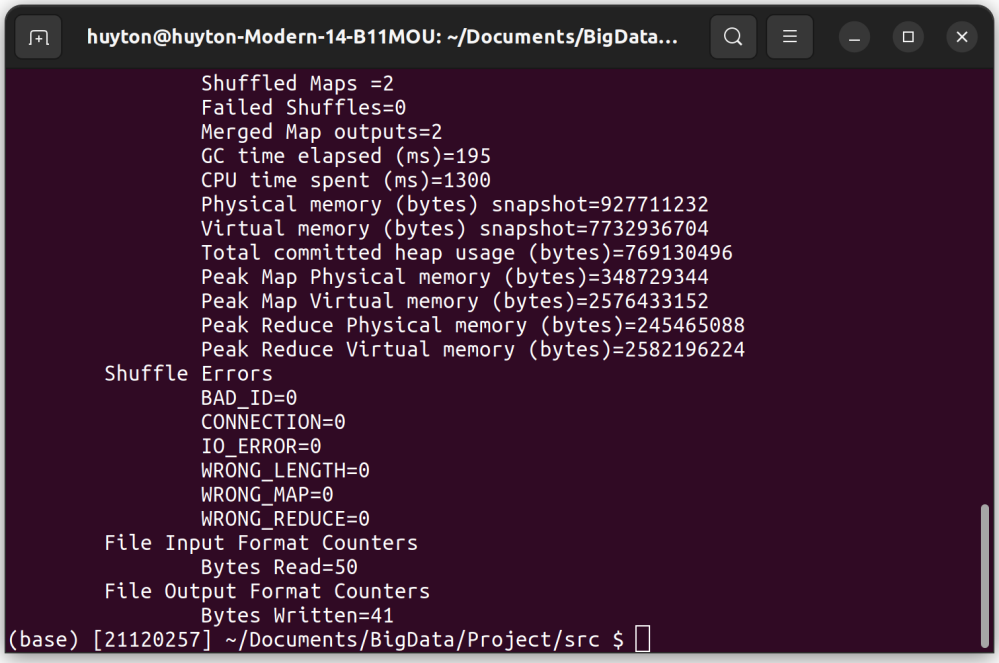
```
hadoop jar wc.jar WordCount /user/21120257/WordCount/input /use
```

The command is used to run a Hadoop job using the `WordCount` class from the `wc.jar` file.

It use all files in `/user/2112057/WordCount/input` as input and save the result to `/user/2112057/WordCount/output` folder

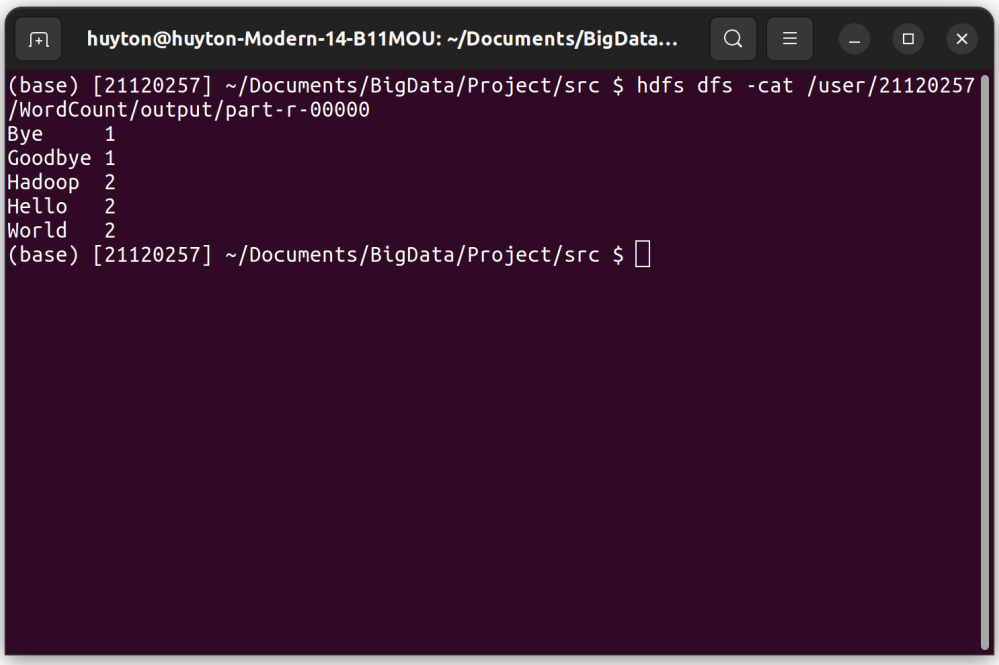
Note: the output folder should'nt be created before because Hadoop MapReduce will thrown an error when output folder already exists. This is build to prevent accidental overwriting data in hdfs.

The result in terminal screen:



Now we will check our output (the content in `part-r-00000`)

```
hdfs dfs -cat /user/21120257/WordCount/output/part-r-00000
```



4. Bonus (2 points)

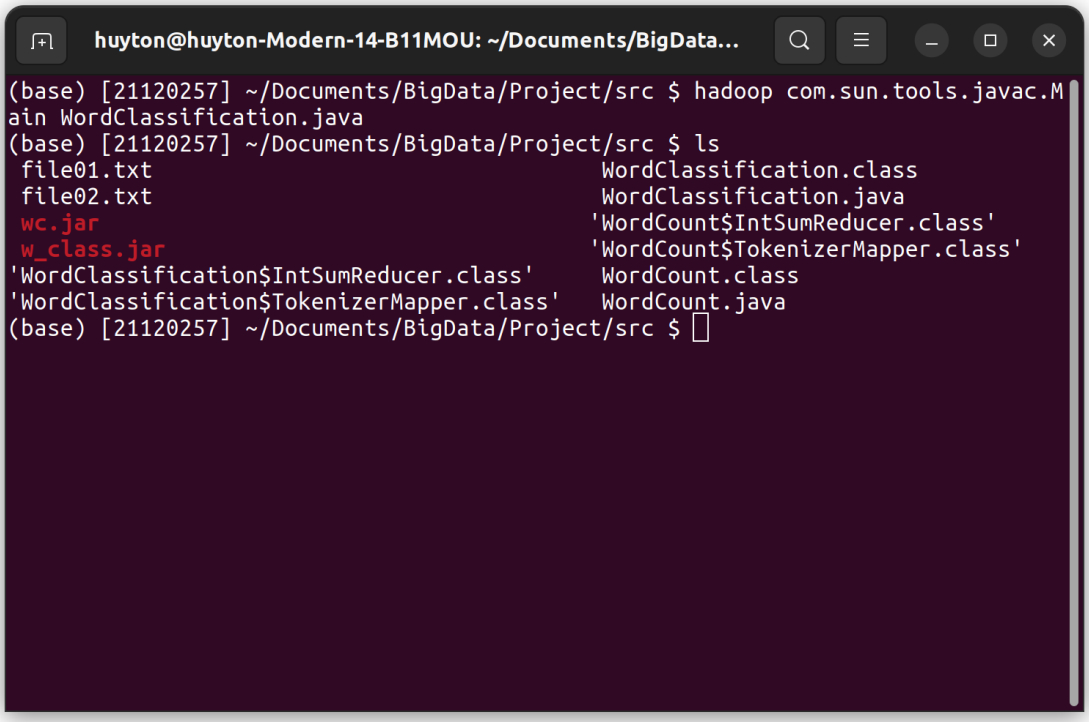
4.1 Word Length Count (0.5 points)

In `WordCount.java` , we will reuse and modify the map method in order to handle classification tasks.

The new files named `WordClassification.java`

Compile WordClassification

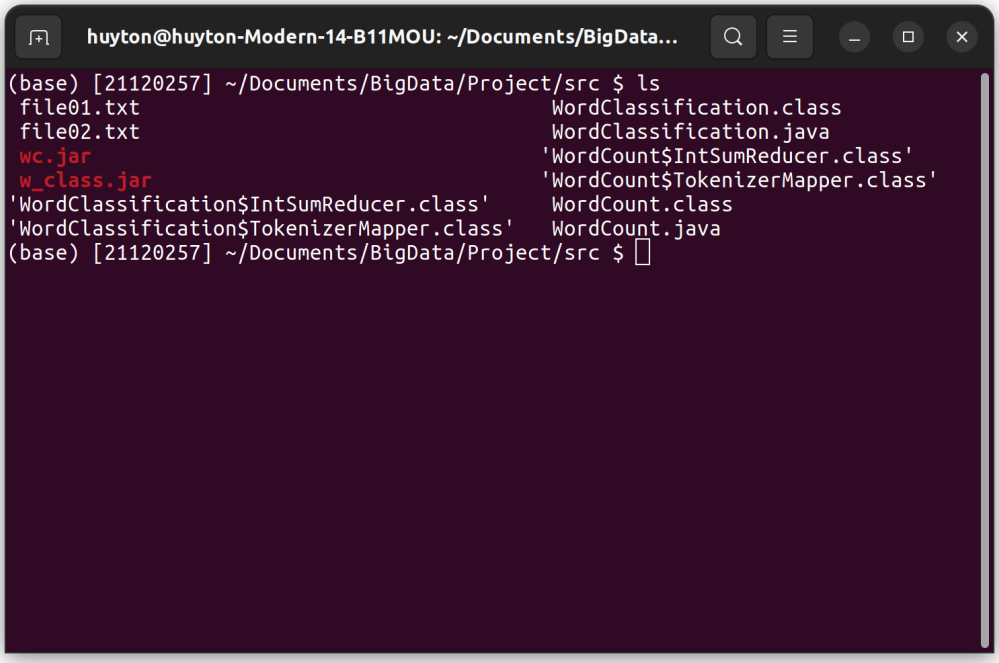
```
$ hadoop com.sun.tools.javac.Main WordClassification.java
```



After this step, we will get `.class` files

Create `.jar` file from `.class` files we have just created

```
$ hadoop jar cf w_class.jar WordClassification*.class
```

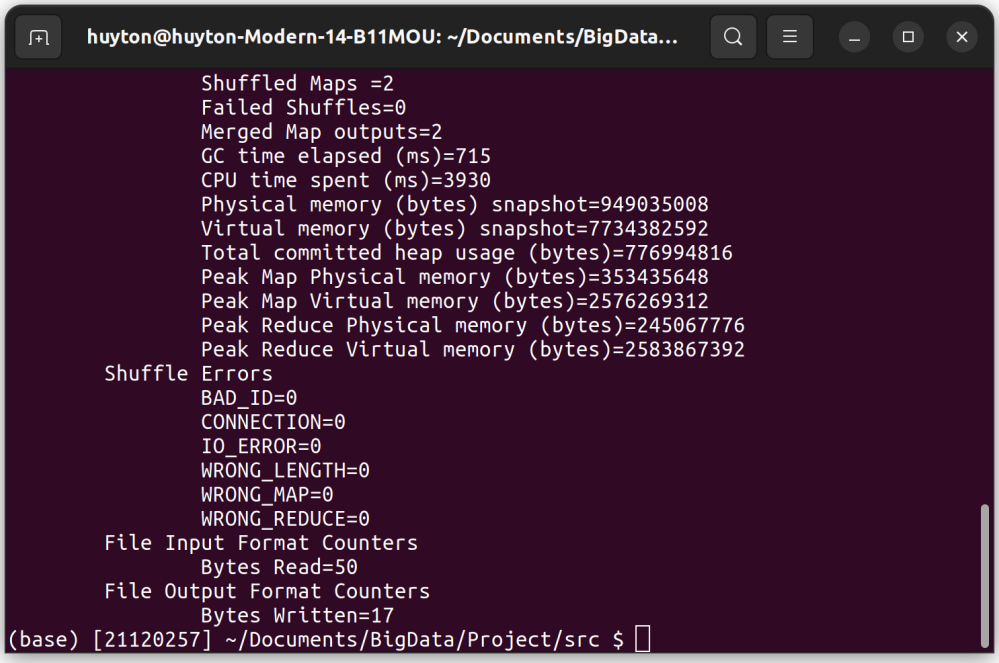


After this step, we should get `w_class.jar`

Run the application

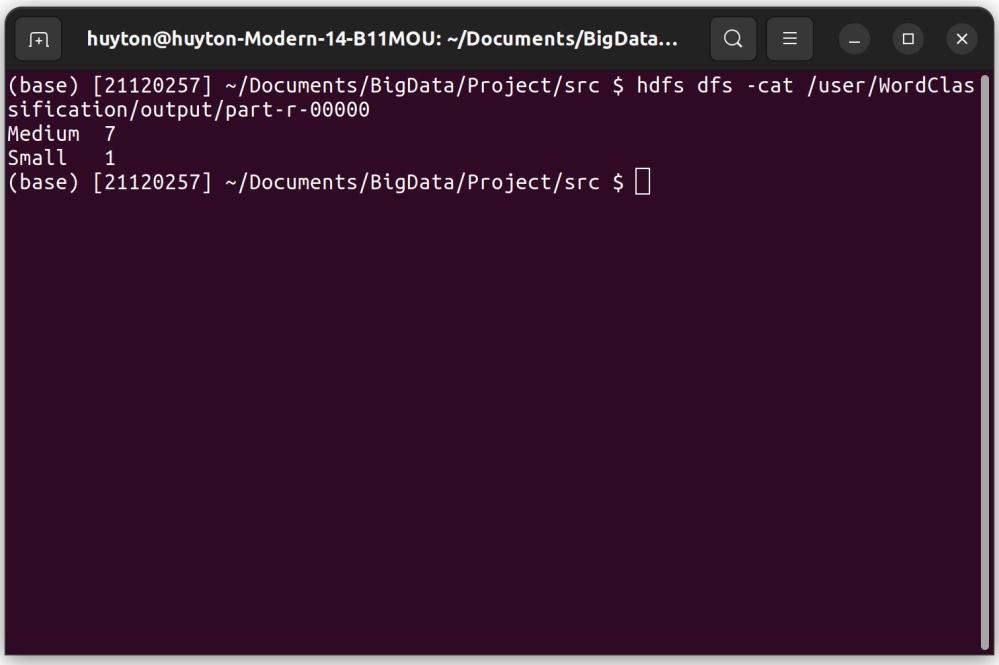
Assume that we have input files in `/user/WordClassification/input` already

```
$ hadoop jar w_class.jar WordClassification /user/WordClassifica
```



Now we will check the output

```
$ hdfs dfs -cat /user/WordClassification/output/part-r-00000
```



III. Reflection

- Knowing the workflow of MapReduce job
- All members have installed Hadoop successfully
- Knowing how to create and run a program of MapReduce
- Knowing some basic command line in HDFS

IV. References

https://research.google/pubs/mapreduce-simplified-data-processing-on-large-clusters/