Lab 01: Introduction to Hadoop Ecosystem

Student's Name: Lê Võ Nhật Minh

Student ID: 22120210

This tutorial was conducted on a Multipass instance (most operations are the same as installing with WSL[1])

Table of contents

I. Setup a Hadoop cluster

Step 1: Prepare the environment

Ensure the system is up to date using command:

```
$ sudo apt update
```

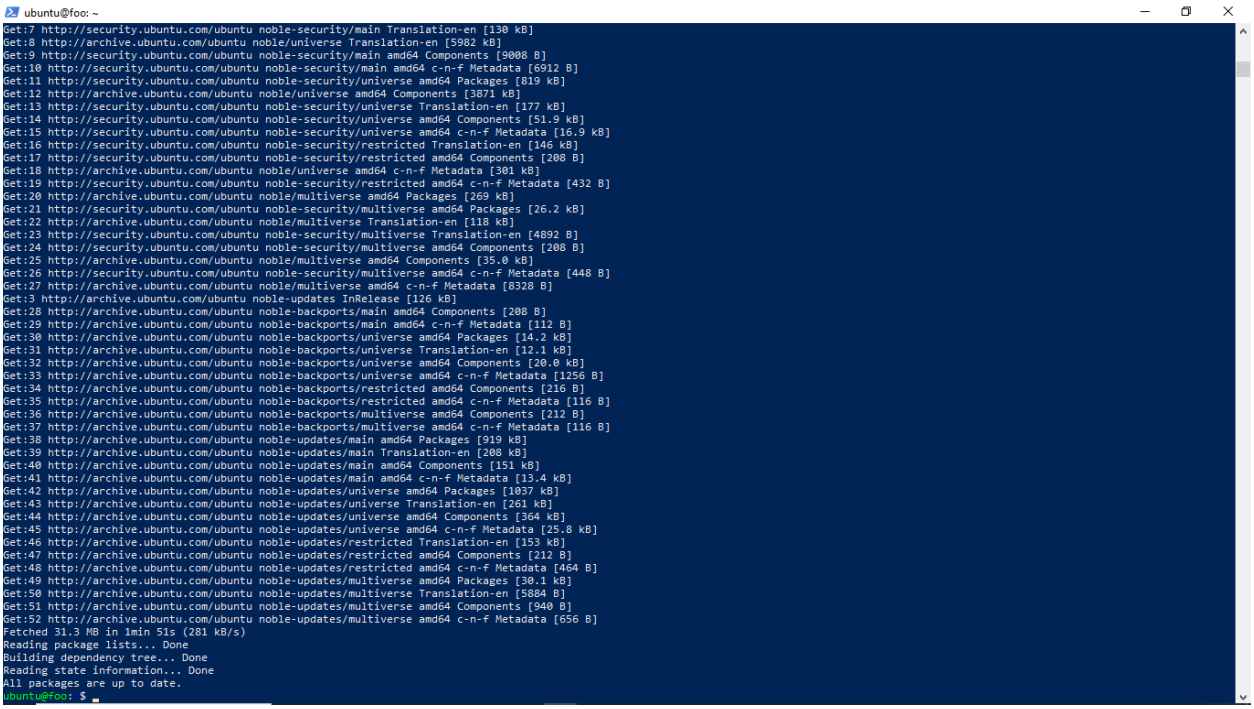

Figure 1.1: Check for the latest version of all packages.

Upgrade all system packages to the latest version.The flag -y used to automatically approve the upgrade

```
$ sudo apt upgrade -y
```

```
ubuntu@foo:~$ sudo apt upgrade -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

Figure 1.2: Upgrade all packages

Install OpenJDK 11 – Java development toolkit for running Java applications (Java Runtime Environment) and developing Java applications (Java Development Kit)

```
$ sudo apt install openjdk-11-jdk -y
```

```
Setting up openjdk-11-jre:amd64 (11.0.26+4-1ubuntu1~24.04) ...
Setting up openjdk-11-jdk-headless:amd64 (11.0.26+4-1ubuntu1~24.04) ...
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jar to provide /usr/bin/jar (jar) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jarsigner to provide /usr/bin/jarsigner (jarsigner) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/javac to provide /usr/bin/javac (javac) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/javadoc to provide /usr/bin/javadoc (javadoc) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/javap to provide /usr/bin/javap (javap) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jcmd to provide /usr/bin/jcmd (jcmd) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jdb to provide /usr/bin/jdb (jdb) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jdeprscan to provide /usr/bin/jdeprscan (jdeprscan) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jdeps to provide /usr/bin/jdeps (jdeps) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jfr to provide /usr/bin/jfr (jfr) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jimage to provide /usr/bin/jimage (jimage) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jinfo to provide /usr/bin/jinfo (jinfo) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jlink to provide /usr/bin/jlink (jlink) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jmap to provide /usr/bin/jmap (jmap) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jmod to provide /usr/bin/jmod (jmod) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jps to provide /usr/bin/jps (jps) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jrunscript to provide /usr/bin/jrunscript (jrunscript) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jshell to provide /usr/bin/jshell (jshell) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jstack to provide /usr/bin/jstack (jstack) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jstat to provide /usr/bin/jstat (jstat) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jstatd to provide /usr/bin/jstatd (jstatd) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/rmic to provide /usr/bin/rmic (rmic) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/serialver to provide /usr/bin/serialver (serialver) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jaotc to provide /usr/bin/jaotc (jaotc) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jhsdb to provide /usr/bin/jhsdb (jhsdb) in auto mode
Setting up openjdk-11-jdk:amd64 (11.0.26+4-1ubuntu1~24.04) ...
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jconsole to provide /usr/bin/jconsole (jconsole) in auto mode
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@foo:~$
```

Figure 1.3: Install Java development toolkit

To verify your installation, use command:

$ java -version

```
ubuntu@foo:~$ java -version
openjdk version "11.0.26" 2025-01-21
OpenJDK Runtime Environment (build 11.0.26+4-post-Ubuntu-1ubuntu124.04)
OpenJDK 64-Bit Server VM (build 11.0.26+4-post-Ubuntu-1ubuntu124.04, mixed mode, sharing)
```

Figure 1.4: Java version detail

Step 2: Create user "khtn_22120210"

Create user "khtn_22120210", using command:

$ sudo adduser khtn_22120210

```
ubuntu@foo:~$ sudo adduser khtn_22120210
info: Adding user `khtn_22120210' ...
info: Selecting UID/GID from range 1000 to 59999 ...
info: Adding new group `khtn_22120210' (1001) ...
info: Adding new user `khtn_22120210' (1001) with group `khtn_22120210 (1001)' ...
info: Creating home directory `/home/khtn_22120210' ...
info: Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for khtn_22120210
Enter the new value, or press ENTER for the default
        Full Name []: Ming3993
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] Y
info: Adding new user `khtn_22120210' to supplemental / extra groups `users' ...
info: Adding user `khtn_22120210' to group `users' ...
```

Figure 2.1: Add a new user to the Linux system

To grant administrative privileges to the new user, add them to the sudo group using the following command:

```
$ sudo usermod -aG sudo khtn_22120210
```

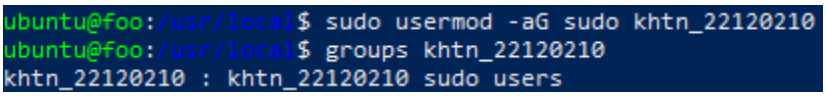

Figure 2.2: Granting administrative privileges to the new user

As you can see, when using command `$ groups khtn_22120210`, user "khtn_22120210" have an alias "sudo".

Upon completion, switch to the user "khtn_22120210" using command:

```
$ sudo su – khtn_22120210
```

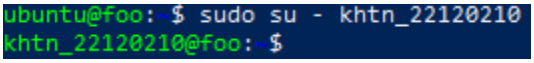

Figure 2.3: Switch to the newly created user

Step 3: Download Hadoop

Download Hadoop 3.3.6 from Hadoop's official website using command:

```
$ wget https://dlcdn.apache.org/hadoop/common/hadoop-3.3.6/hadoop-3.3.6.tar.gz
```

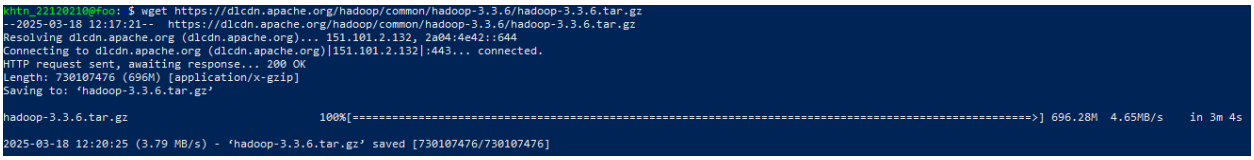

Figure 3.1: Downloading Hadoop

Extract Hadoop using command:

```
$ tar -xvzf hadoop-3.3.6.tar.gz
```

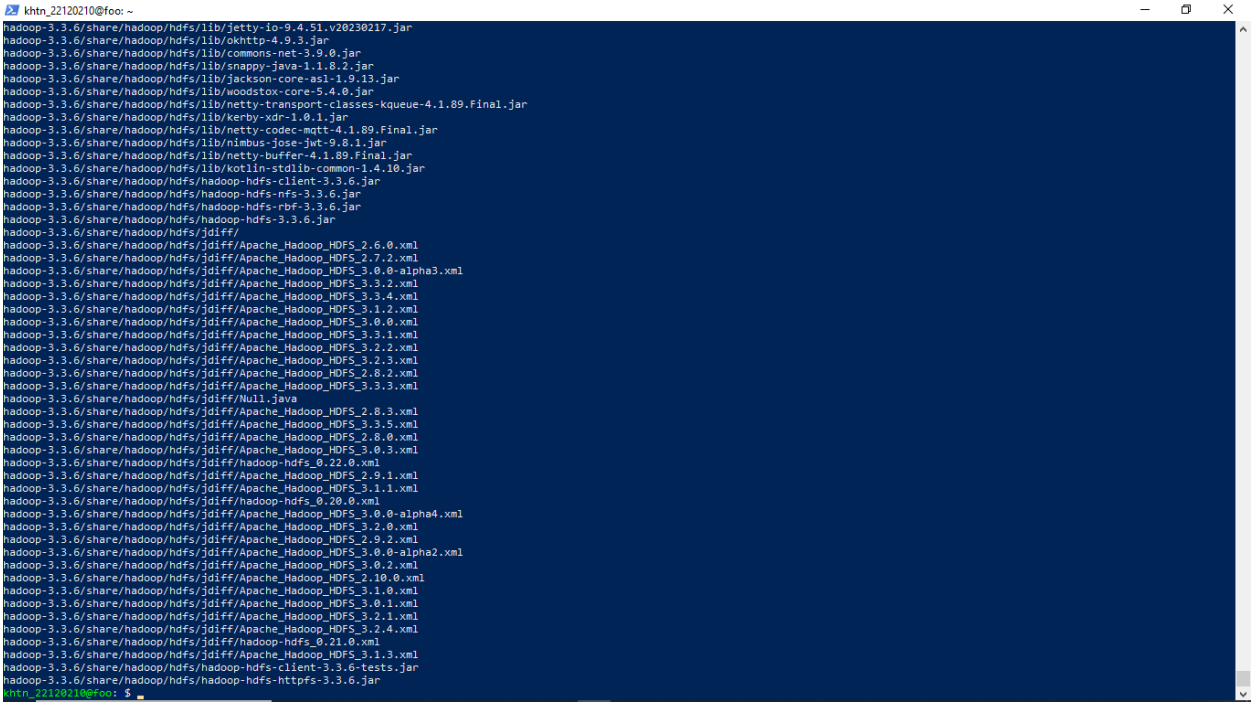

Figure 3.2: Extracting Hadoop

Check whether we are currently in the same directory as hadoop-3.3.6 (default: ***/home/khtn_22120210***). If so, move the files to ***/usr/local/hadoop*** using the following command:

```
$ sudo mv hadoop-3.3.6 /usr/local/hadoop
```

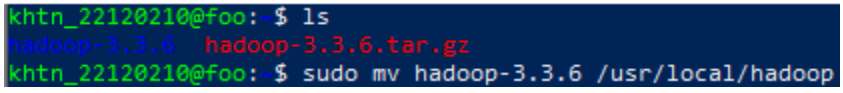

Figure 3.3: Move Hadoop to the new folder

Create a directory to store logs by using:

```
$ sudo mkdir /usr/local/hadoop/logs
```

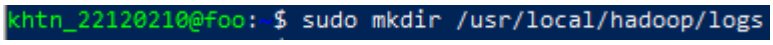

Figure 3.4a: Create directory for storing logs

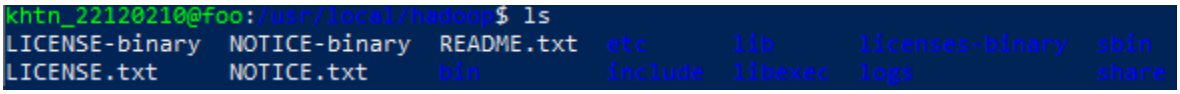

Figure 3.4b: Verify successful creation

Alter the ownership of the ***/usr/local/hadoop*** directory to the user "khtn_22120210" using following command:

```
$ sudo chown -R khtn_22120210 /usr/local/hadoop
```

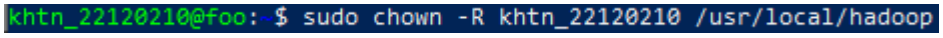

Figure 3.5: Alter directory's ownership

Step 4: Create SSH key

Install the OpenSSH server and client:

```
khtn_22120210@foo:~$ sudo apt install ssh
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  ssh
0 upgraded, 1 newly installed, 0 to remove and 15 not upgraded.
Need to get 4658 B of archives.
After this operation, 57.3 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 ssh all 1:9.6p1-3ubuntu13.8 [4658 B]
Fetched 4658 B in 1s (7838 B/s)
Selecting previously unselected package ssh.
(Reading database ... 77216 files and directories currently installed.)
Preparing to unpack .../ssh_1%3a9.6p1-3ubuntu13.8_all.deb ...
Unpacking ssh (1:9.6p1-3ubuntu13.8) ...
Setting up ssh (1:9.6p1-3ubuntu13.8) ...
Scanning processes...
Scanning candidates...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

User sessions running outdated binaries:
 ubuntu @ session #3: java[2082]

No VM guests are running outdated hypervisor (qemu) binaries on this host.
```

Figure 4.1: Instal OpenSSH Server package

Use the subsequent command to generate both private and public keys (it's important not to set the passphrase):

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

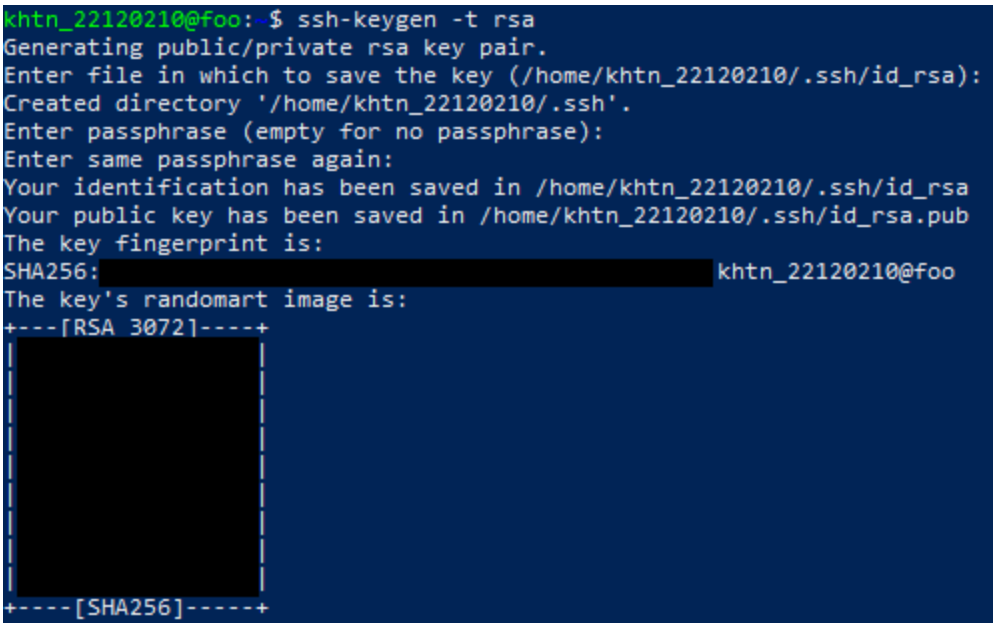

Figure 4.2: Generate private and public keys

Add the public key to *authorized_keys* using command:

```
$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

Using the following command change the file permissions of *authorized_keys*:

```
$ sudo chmod 640 ~/.ssh/authorized_keys
```

Start the SSH service:

```
$ sudo service ssh start
```

Confirm the SSH configuration:

```
$ ssh localhost
```

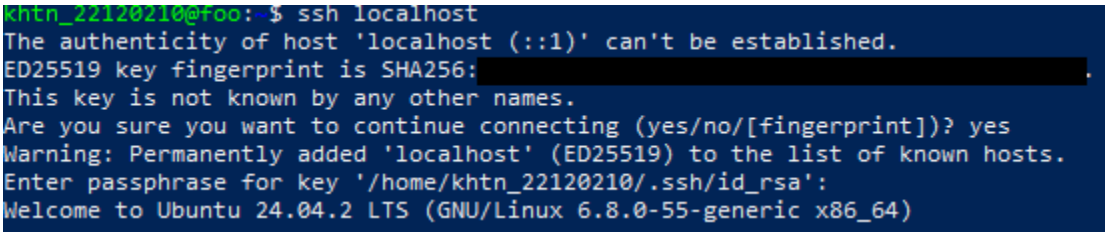

Figure 4.3: Localhost's information

Step 5: Configuring Hadoop's environmental variables

Open configuration file using command:

```
$ nano ~/.bashrc
```

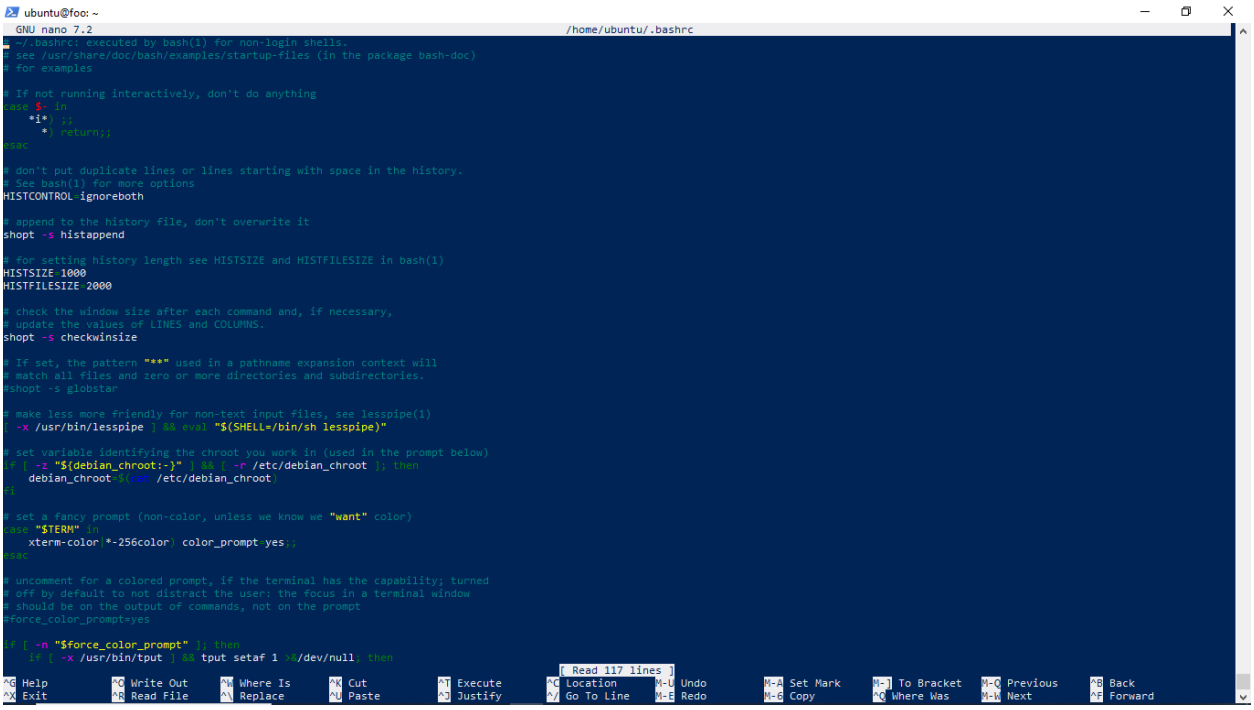

Figure 5.1: Open *.bashrc* file

Move to the end of the file using Ctrl + / then Ctrl + V

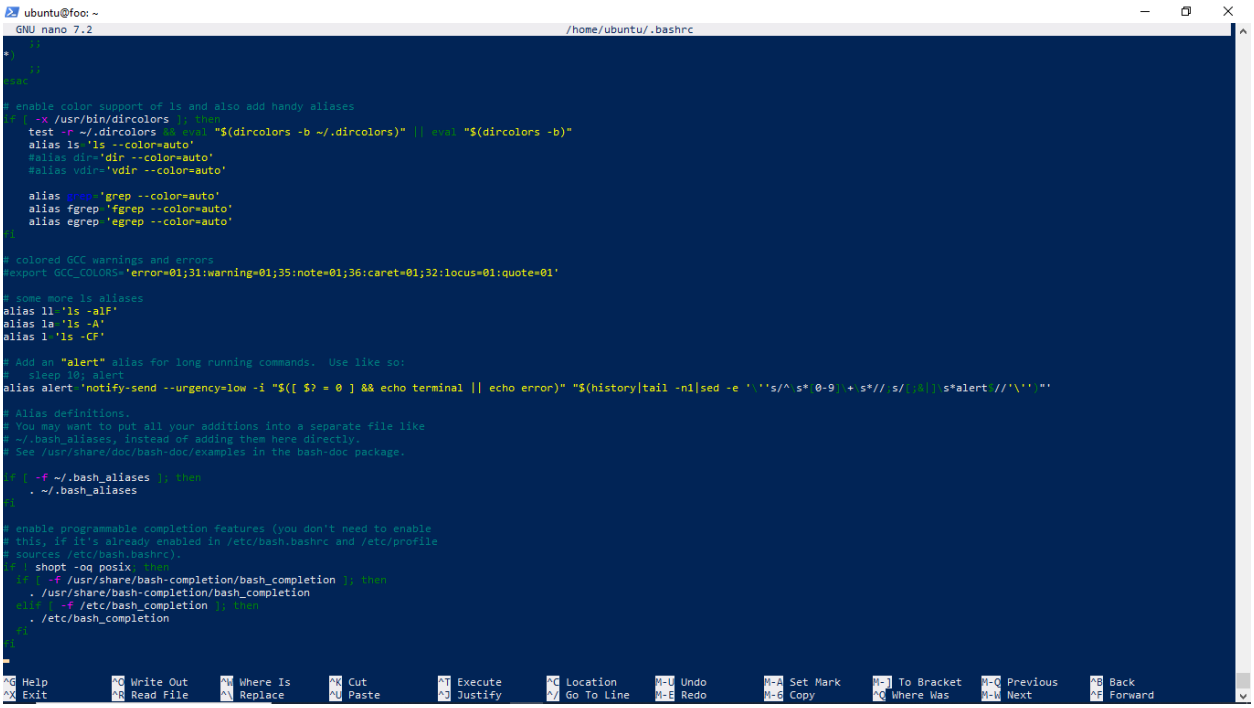

Figure 5.2: The end of *.bashrc* file

Paste Hadoop's configuration at the end of the file:

```
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
export HADOOP_HOME=/usr/local/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"
```

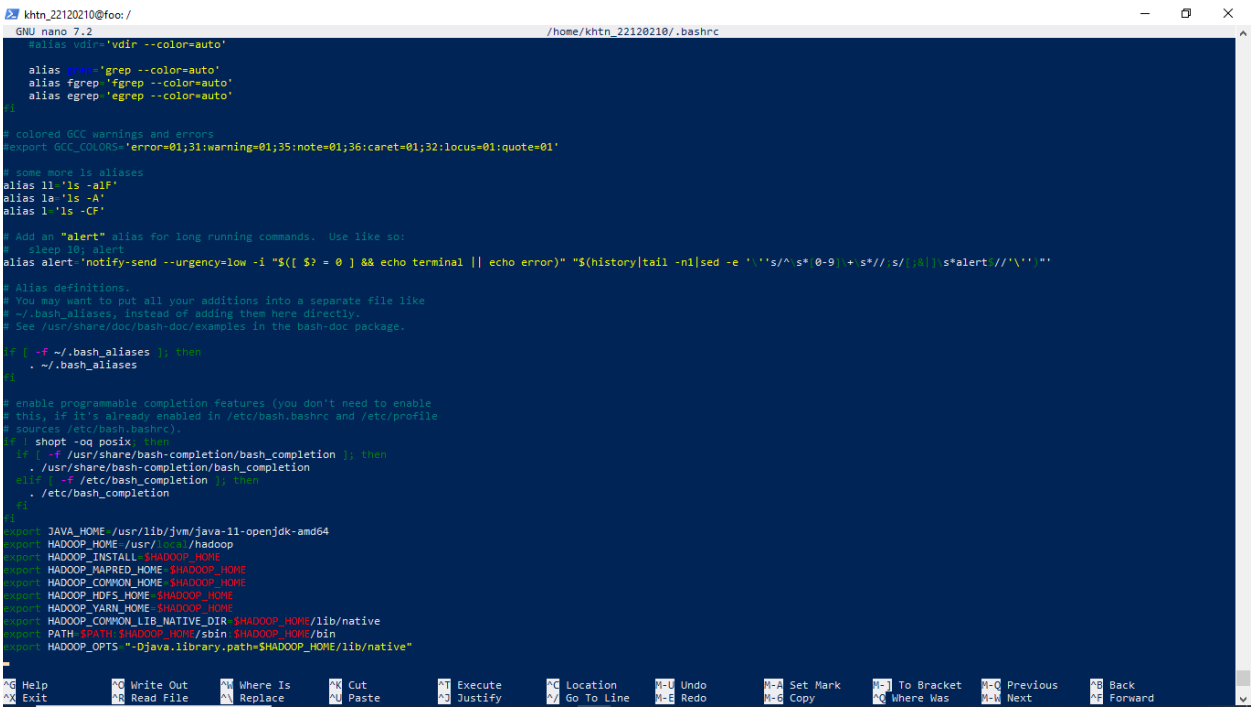

Figure 5.3: Adding configuration to *.bashrc* file

Save the changes by pressing Ctrl+S and exit the *nano* text editor by pressing Ctrl+X

To enable the changes, source the *.bashrc* file using command:

```
$ source ~/.bashrc
```

```
khtn_22120210@foo:~$ source ~/.bashrc
khtn_22120210@foo:~$
```

Figure 5.4: Enable the changes

Check for Hadoop version using command:

```
$ hadoop version
```

```
khtn_22120210@foo:/$ hadoop version
Hadoop 3.3.6
Source code repository https://github.com/apache/hadoop.git -r 1be78238728da9266a4f88195058f08fd012bf9c
Compiled by ubuntu on 2023-06-18T08:22Z
Compiled on platform linux-x86_64
Compiled with protoc 3.7.1
From source with checksum 5652179ad55f76cb287d9c633bb53bbd
This command was run using /usr/local/hadoop/share/hadoop/common/hadoop-common-3.3.6.jar
```

Figure 5.5: Hadoop version information

Step 6: Config Hadoop

Define java environment variables in *hadoop-env.sh* file:

To open *hadoop-env.sh* file use the command:

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

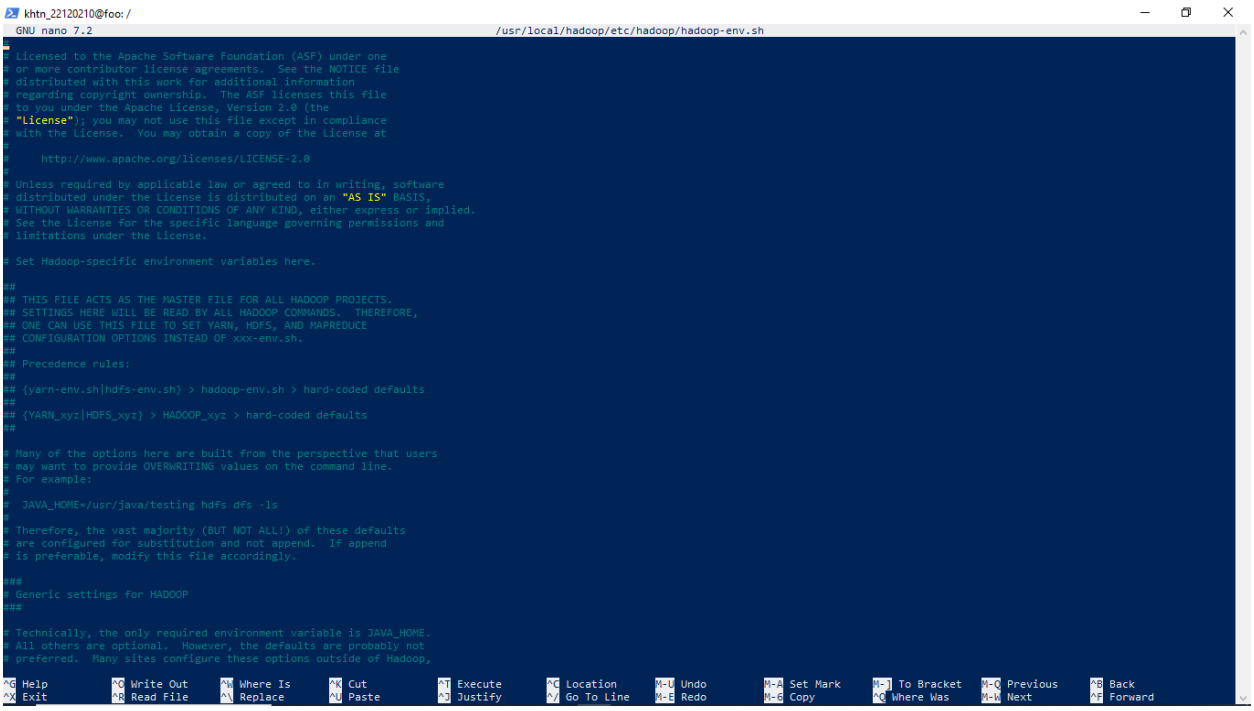

Figure 6.1: Open *hadoop-env.sh* file

To search for the “export JAVA_HOME” phrase use Ctrl + W

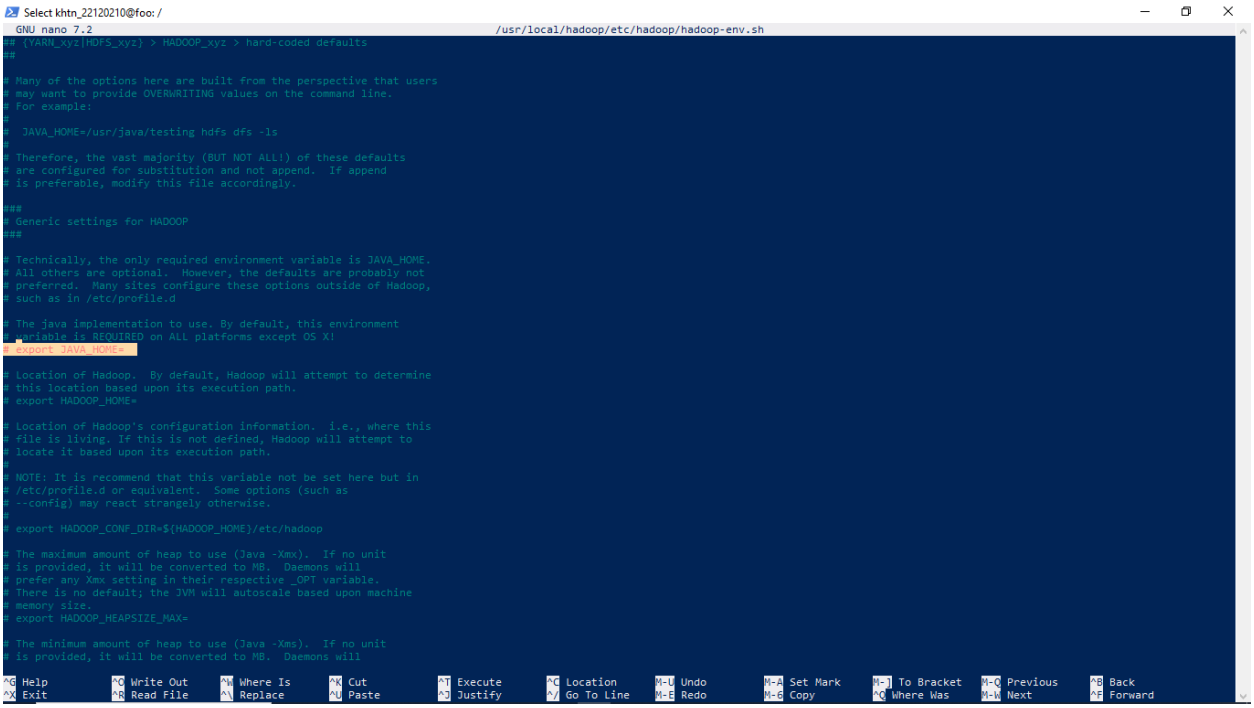

Figure 6.2: Search for “export JAVA_HOME” phrase

And change it as below:

```
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
```

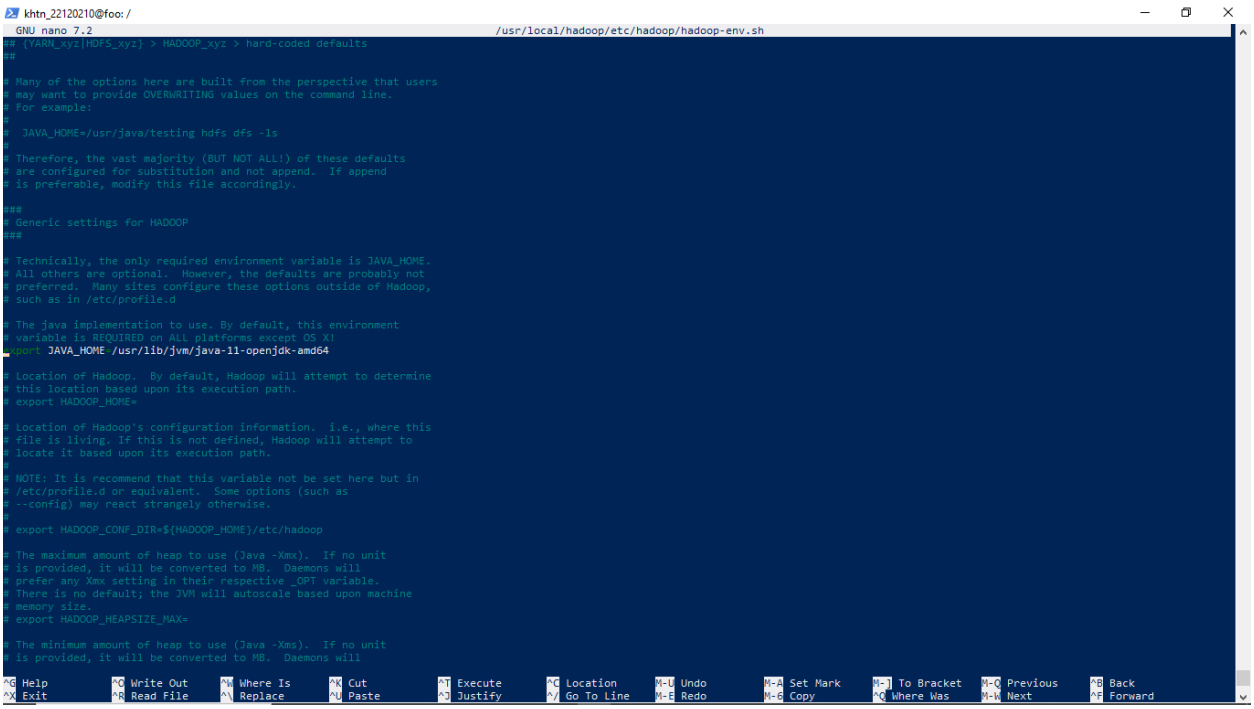

Figure 6.3: Change JAVA_HOME directory path

Configuring Hadoop to Pseudo-Distributed Mode:

Open file *core-site.xml* to configure HDFS using command:

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

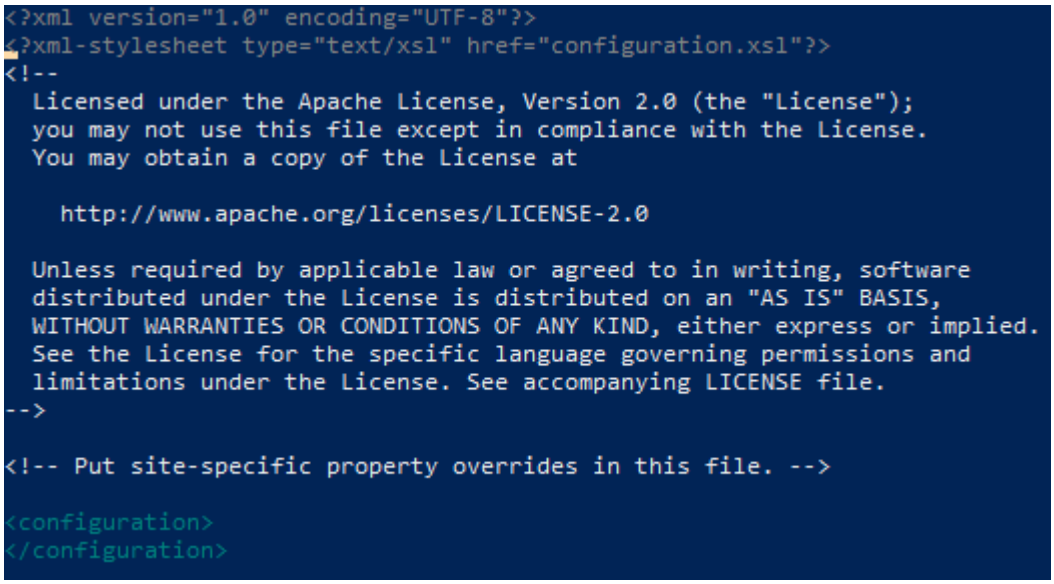

Figure 6.4: Open *core-site.xml* file

Add the following lines between <configuration> </configuration>

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

(To edit the file press Ctrl + Y, then Ctrl + V to go to the end of file and right-click to paste those above properties in place)

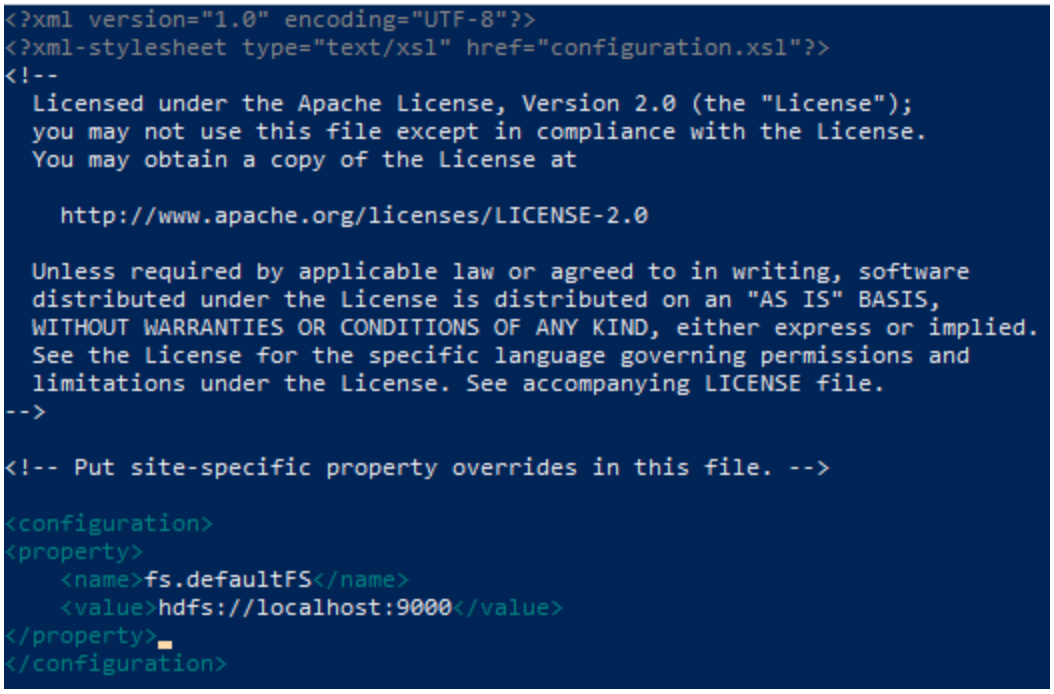

Figure 6.5: Config *core-site.xml* file

Then press Ctrl + S to save the file then Ctrl + X to exit

Create a directory to store node metadata using the following command:

```
$ sudo mkdir -p /home/hadoop/hdfs/{namenode,datanode}
```

Figure 6.6: Create directory to store metadata

Grant the ownership of the created directory to the "khtn_22120210" user:

```
$ sudo chown -R khtn_22120210 /home/hadoop/hdfs
```

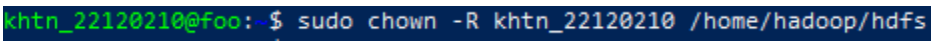

Figure 6.7: Grant the newly created directory to the "khtn_22120210" user

Open file *hdfs-site.xml* to configure data storage folder, using command:

```
$ sudo nano $HADOOP_HOME/etc/hadoop/hdfs-site.xml
```

Add the following properties between <configuration> </configuration>

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

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

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

Figure 6.8: Config *hdfs-site.xml* file

Open file *mapred-site.xml* to configure MapReduce mode (default is Standalone Mode), using command:

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

Add the following lines between <configuration> </configuration>

```
<property>
   <name>mapreduce.framework.name</name>
   <value>yarn</value>
</property>
```

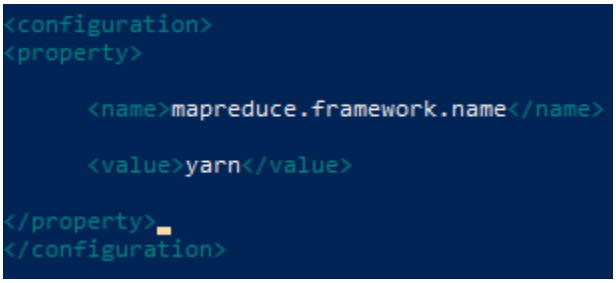

Figure 6.9: Config *mapred-site.xml* file

Open file yarn-site.xml to configure auxiliary service (default is Standalone Mode), using command:

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

Add the following lines between <configuration> </configuration>

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

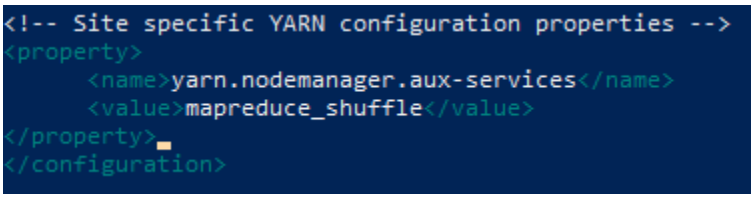

Figure 6.10: Config *yarn-site.xml* file

Format HDFS' namenode, using command:

```
$ hdfs namenode -format
```

After starting YARN and DFS on “foo,” I realized that I hadn’t allocated enough RAM to the virtual machine. As a result, “foo” would not be able to handle other tasks needed to complete this lab. Therefore, I will create a new instance called “bigdata” and use it as an alternative. All setups are similar to the tutorial shown above.

Step 7: Start Hadoop

To start YARN and DFS, using command:

```
$ start-dfs.sh
$ start-yarn.sh
```

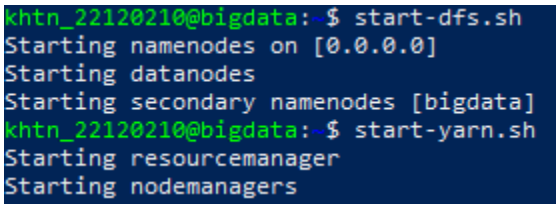

Figure 7.1: Start YARN and DFS

To verify running services, using command:

```
$ jps
```

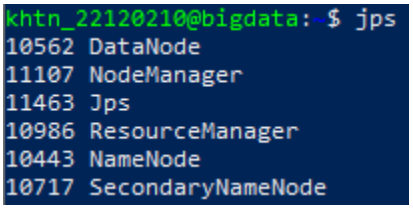

Figure 7.2: Running Java service

Now you can use the provided Hadoop UI (**Note:** If you set up the environment on a virtual machine, use the VM's IPv4 address instead of **localhost** to get access to Hadoop web services. As shown below, my instance's IPv4 address is **172.22.235.75**)

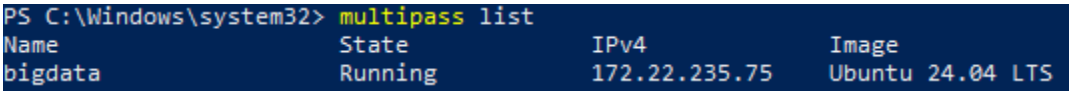

Figure 7.3: Multipass instance's IPv4 address

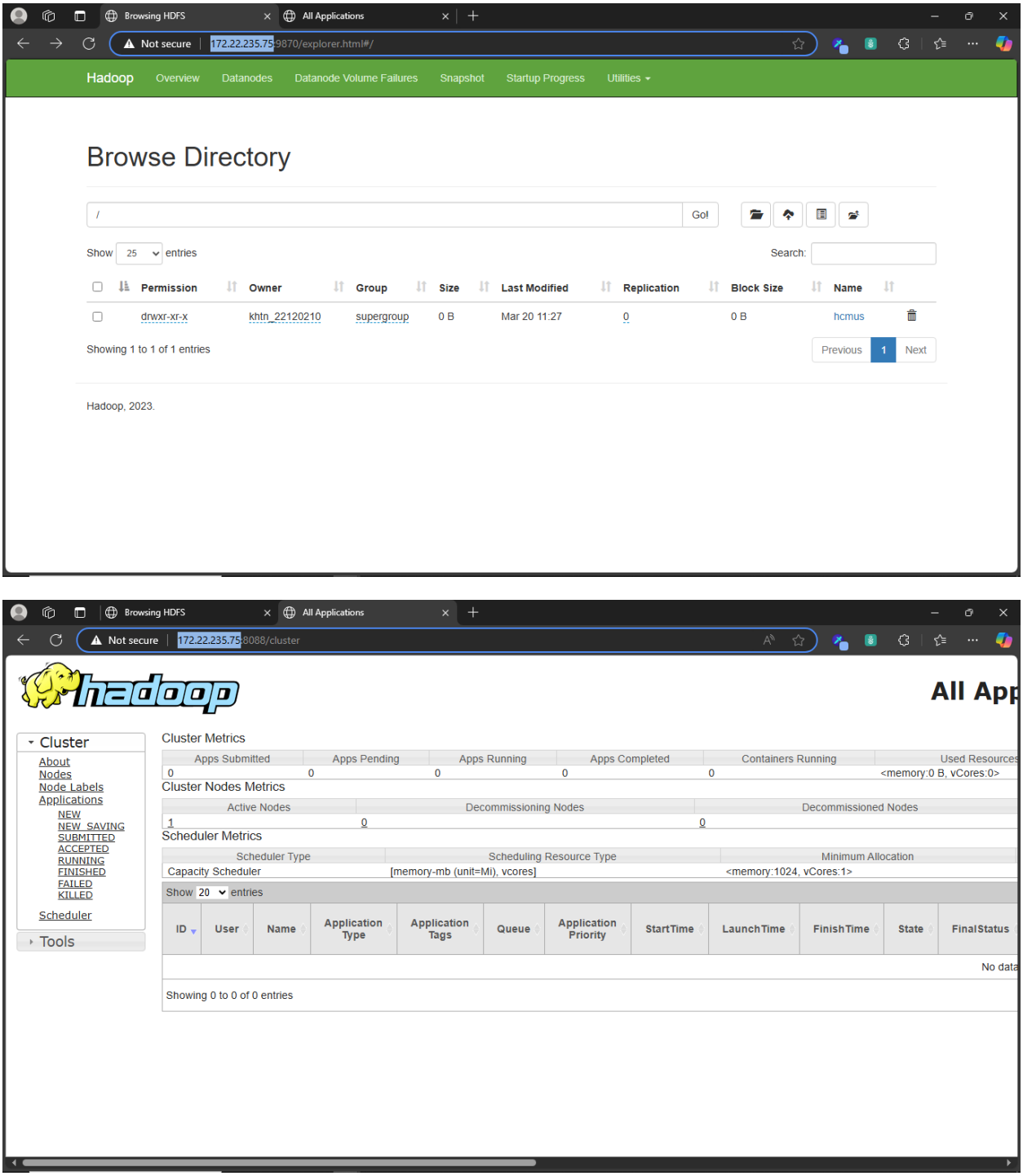

Figure 7.4a & 7.4b: Hadoop web services' UI

Step 8: File manipulation with Hadoop

Use the following commands to create a directory for *hadoop-test.jar* on HDFS:

```
$ hdfs dfs -mkdir /hcmus
```

```
$ hdfs dfs -mkdir /hcmus/22120210
```

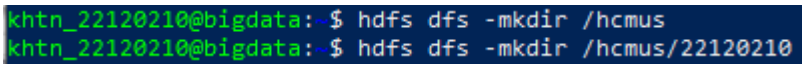

Figure 8.1: Create directory on DFS

Verify the owner of the directory:

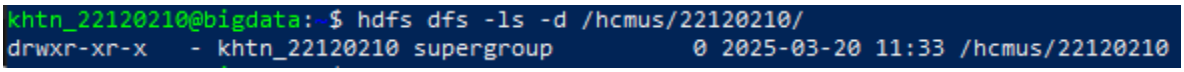

Figure 8.2: Directory's information

khtn_22120210 is the creator also owner of the directory so it is not necessary to change the owner

To transfer file between host and instance "bigdata", open Windows Powershell with Administrator previllige and type the following command (for ease of using, you can also use mount[2]):

```
multipass transfer "E:\hadoop-test.jar" bigdata:/home/ubuntu
```
[3]

After that, move the file to ***/home/khtn_22120210***, using command:

```
$ sudo mv hadoop-test.jar /home/khtn_22120210
```

Verifying the transfer:

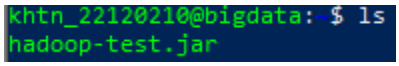

Figure 8.3: Verifying the transfer

Upon completion, use the following command to upload file on HDFS:

```
$ hdfs dfs -put hadoop-test.jar /hcmus/22120210
```

Verify if the file was uploaded successfully:

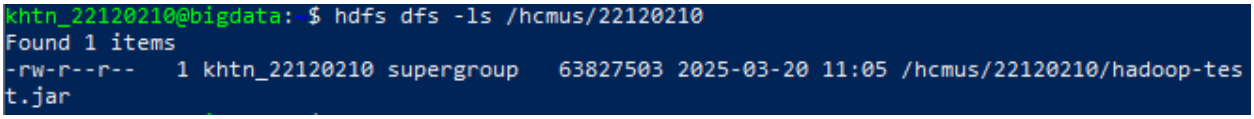

Figure 8.4: Verifying the upload

You can also use the web service UI to upload the file:

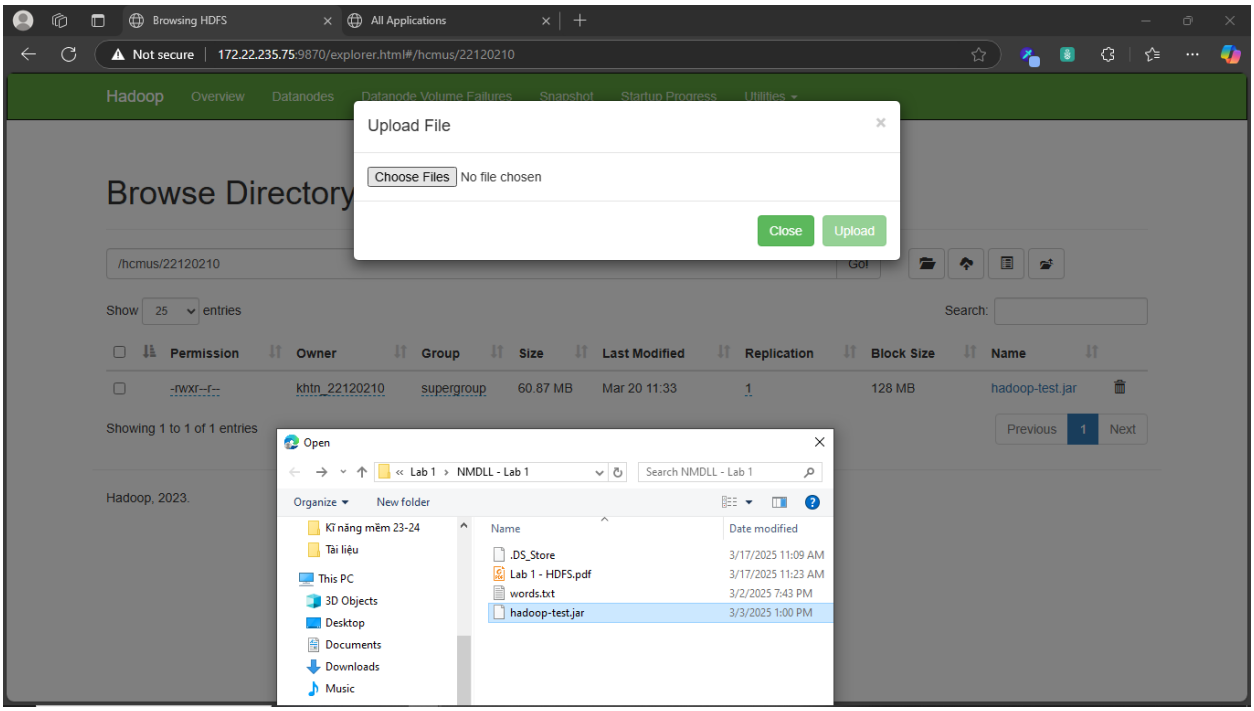

Figure 8.5: Upload file using web service UI

Change the file permission to execute the file on hdfs:

```
$ hdfs dfs -chmod 744 /hcmus/22120210/hadoop-test.jar
```

Step 9: Run the programme

To execute the file, use command:

```
$ java -jar hadoop-test.jar 9000/hcmus/22120210
```

```
khtn_22120210@bigdata:~$ java -jar hadoop-test.jar 9000 /hcmus/22120210
Trying to read /hcmus/22120210
log4j:WARN No appenders could be found for logger (org.apache.hadoop.util.Shell).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Found hdfs://localhost:9000/hcmus/22120210/hadoop-test.jar
Your student ID: 22120210 (ensure it matches your student ID)
The first method to get MAC address is failed: Could not get network interface
Trying the alternative method
The first method to get MAC address is failed: Could not get network interface
Trying the alternative method
File written at /home/khtn_22120210/22120210_verification.txt
```

Figure 9.1: Output after executing the *hadoop-test.jar* file

Verifying the result:

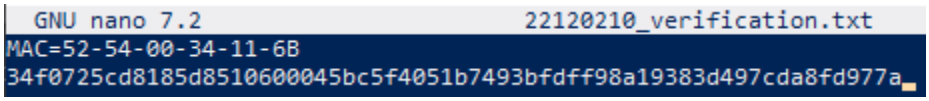

Figure 9.2: The content of *22120210_verification.txt* file

II. Warm up with Word Count

1. Word Count Mapper.

```
letters_set = {'a', 'f', 'g', 'j', 'h', 'c', 'm', 'u', 's'}

for line in sys.stdin:
    words = re.split(r"[^a-zA-Z]+", line.strip())

    for word in words:
        if word and word[0].lower() in letters_set:
            print(f"{word[0].lower()}\t1")
```

First, we create a set containing the specified letters for later use.

Next, we read each line from the system input (**`sys.stdin`**) and process it.

The line **`re.split(r"[^a-zA-Z]+", line.strip())`** splits the input line into a list of words, using any non-alphabetical character (such as numbers, punctuation, or spaces) as a delimiter.

After splitting the words, we iterate through each one. If the word is not empty and its first letter (converted to lowercase) is in the predefined set, we print the first letter followed by the number 1, separated by a tab (\t).

2. Word Count Reducer.

```
desired_order = ['a', 'f', 'j', 'g', 'h', 'c', 'm', 'u', 's']
result = OrderedDict((key, 0) for key in desired_order)

for line in sys.stdin:
    letter, cnt = line.split('\t', 1)
    try:
        cnt = int(cnt)
    except ValueError:
        continue

    if letter in result:
        result[letter] += cnt

for letter in desired_order:
    if result[letter] > 0:
        print(f"{letter}\t{result[letter]}")
```

First, we create a list of letters in the desired order for output.

Next, we initialize an ordered dictionary where each key is a letter from the predefined list, and all values are initially set to zero.

Then, we iterate through each line of the system input, which comes from the mapper output. Each line contains a letter and a count, separated by a tab (**\t**). We attempt to convert the count into an integer, and if successful, we update the corresponding value in the dictionary.

Finally, we print the results in the predefined order, only including letters that have a count greater than zero.

3. Word Count Result.

Here is the result:

a	32921
f	18793
j	4530
g	16002
h	20911
c	42817
m	27239
u	24301
s	59567

III. References:

[1] Iqbal, M. (2024, January 9). Apache Hadoop 3.3.6 installation on Ubuntu 22.04.2 LTS WSL for Windows. Retrieved 2025, March 17th, from *Medium*. https://medium.com/@madihaiqbal606/apache-hadoop-3-3-6-installation-on-ubuntu-22-04-2-lts-wsl-for-windows-bb57ed599bc6

[2] Abstract programmer. (2021, April 29). *Ubuntu Multipass tutorial for Linux* [Video]. Retrieved 2025, March 24th, from YouTube. https://www.youtube.com/watch?v=Ky21G-ilMok

[3] Ltd, C. (n.d.-a). `*multipass transfer*` *command* | *Canonical*. Retrieved 2025, March 20th, from Canonical. https://canonical.com/multipass/docs/transfer-command