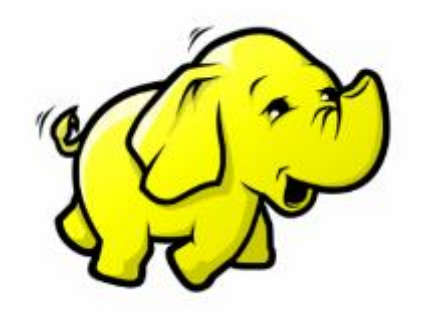


HADOOP CLUSTER

By: Numaira Zaib

TABLE OF CONTENTS

INTRODUCTION

Distributed computing is a model in which components of a software system are shared among multiple computers. Even though the components are spread out across multiple computers, they are run as one system. This is done in order to improve efficiency and performance. Hadoop is an open-source software framework for storing data and running applications on clusters of commodity hardware. It provides massive storage for any kind of data, enormous processing power, and the ability to handle virtually limitless concurrent tasks or jobs. The four important libraries of Hadoop are:

1. Hadoop Common – The role of this character is to provide common utilities that can be used across all modules.
2. Hadoop MapReduce - The right hand of our actor, carrying out all the work assigned to it i.e. It does the job scheduling and processing across the cluster. Hadoop is like a data warehousing system so it needs a library like MapReduce to actually process the data.
3. Hadoop Distributed File System (HDFS) – The left hand, maintains all the records i.e. file system management across the cluster.
4. Hadoop YARN – This is the newer and improved version of MapReduce, from version 2.0, and does the same work.

Hadoop is used where there is a large amount of data generated and your business requires insights from that data. The power of Hadoop lies in its framework, as virtually most of the software can be plugged into it and can be used for data visualization. Hadoop is used in big data applications that gather data from disparate data sources in different formats. HDFS is flexible in storing diverse data types.

ADVANTAGES OF HADOOP

Scalable

Hadoop is a highly scalable storage platform because it can store and distribute very large data sets across hundreds of inexpensive servers that operate in parallel. Unlike traditional relational database systems (RDBMS) that can’t scale to process large amounts of data.

Cost-effective

Hadoop also offers a cost-effective storage solution for businesses exploding data sets. The problem with traditional relational database management systems is that it is extremely cost prohibitive to scale to such a degree in order to process such massive volumes of data. In an effort to reduce costs, many companies in the past would have had to down-sample data and classify it based on certain assumptions as to which data was the most valuable.

Flexible

Hadoop enables businesses to easily access new data sources and tap into different types of data (both structured and unstructured) to generate value from that data. This means businesses can use Hadoop to derive valuable business insights from data sources such as social media, email conversations.

Fast

Hadoop's unique storage method is based on a distributed file system that basically 'maps' data wherever it is located on a cluster. The tools for data processing are often on the same servers where the data is located, resulting in much faster data processing. If you're dealing with large volumes of unstructured data, Hadoop is able to efficiently process terabytes of data in just minutes, and petabytes in hours.

Resilient to failure

A key advantage of using Hadoop is its fault tolerance. When data is sent to an individual node, that data is also replicated to other nodes in the cluster, which means that in the event of failure, there is another copy available for use.

METHODOLOGY

We opened a virtual machine, then we install java and run other commands and make changes in 5 files to install hadoop. Now hadoop is installed and we can access hadoop directly through the terminal and it is for a single node cluster.

Then we make master and slave from one virtual machine by cloning and making it 'Master' and 'Slave'. Then we find out the addresses of the master and slave and then ping their addresses. Then we install SQL through hadoop. We make changes in the database of slave and when we write command show database in master then it displays the same result as a slave. So we can access the same records and database from the master as well. The connection between master and slave is fine.

Then we write commands for running MapReduce. We used MapReduce 3.2.3 in the project. We implemented MapReduce in java and python in both languages. MapReduce is a software framework and programming model used for processing huge amounts of data. MapReduce program work in two phases, namely, Map and Reduce. Map tasks deal with splitting and mapping of data while Reducing tasks shuffle and reducing the data. Hadoop is capable of running MapReduce programs written in various languages: Java, Ruby, Python, and C++. The programs of Map Reduce in cloud computing are parallel in nature, and thus are very useful for performing large-scale data analysis using multiple machines in the cluster. The input to each phase is key-value pairs. In addition, every programmer needs to specify two functions: map function and reduce function.

HADOOP ON UBUNTU

INSTALLING HADOOP

sudo apt update: This command is used for updating the system.

sudo apt install openjdk-8-jdk -y: This command is used for installing java in the machine.

java -version; javac -version: This command is used for checking java version.

sudo apt install openssh-server openssh-client -y: This command is for opening ssh server. Since hadoop will run in the same machine where my unix box in running so there has to be a connection between hadoop and my unix machine. This ssh server will give me that connection.

sudo adduser hdoop: This command is used for adding a new user 'hdoop' . This user will use hadoop in my system.

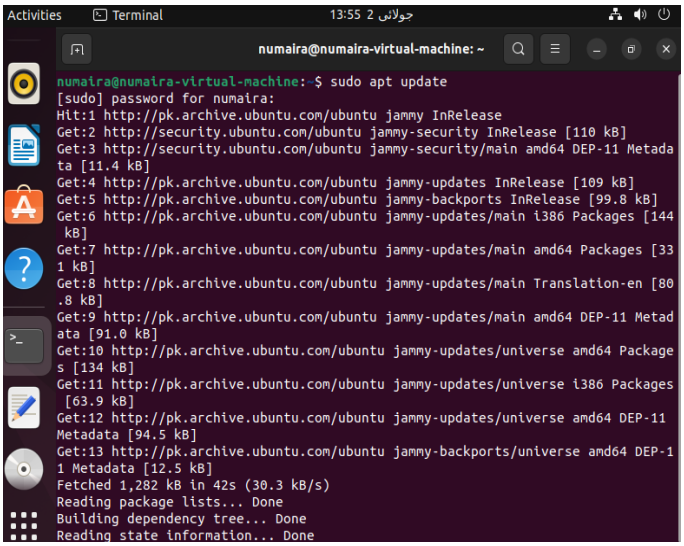
su - hdoop: This command is used for changing the user.

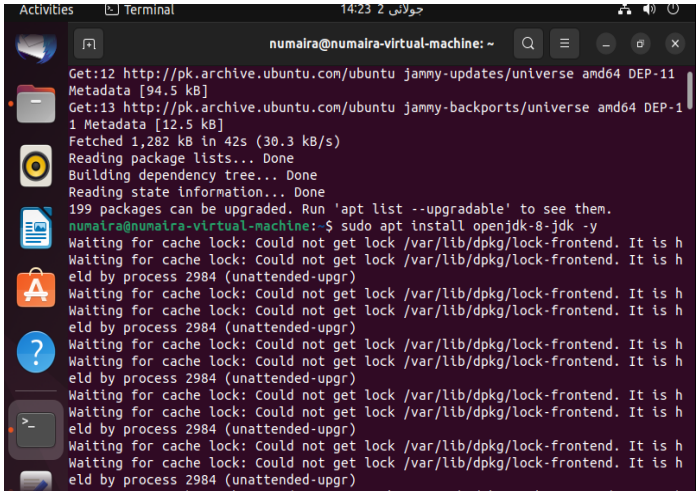
ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa: This command is used for generating a key which I will use for the connection or the coordination to the local machine to the hadoop.

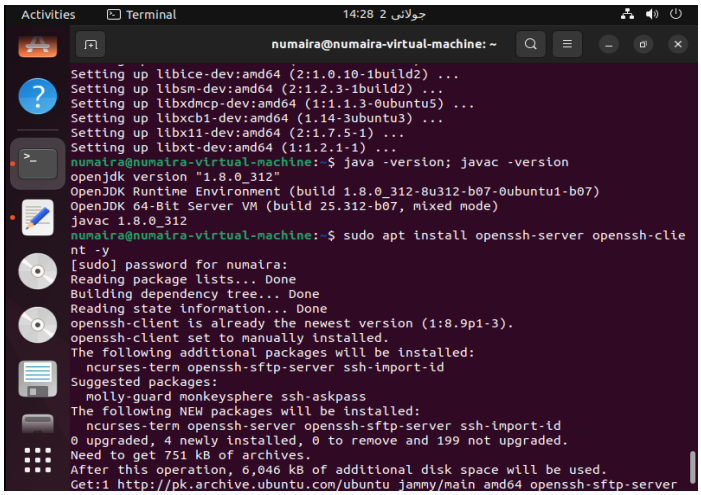
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys: This command is used for putting my key in my authorized key list. This is why we are making hdoop user able to make connection with hadoop.

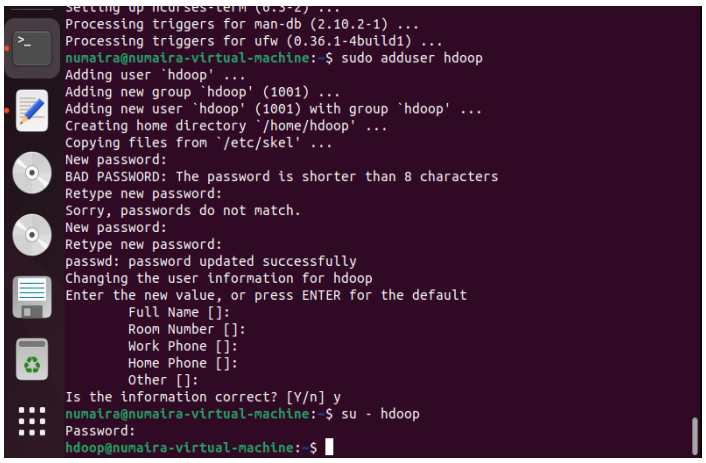
chmod 0600 ~/.ssh/authorized_keys: This command is used for changing the permission of this authorized key file.

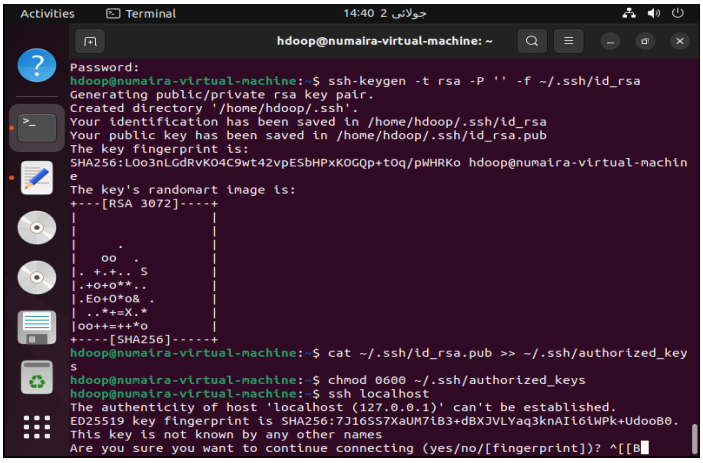
ssh localhost: This means the connection is going.











DOWNLOADING HADOOP

wget https://downloads.apache.org/hadoop/common/hadoop-3.2.3/hadoop-3.2.3.tar.gz

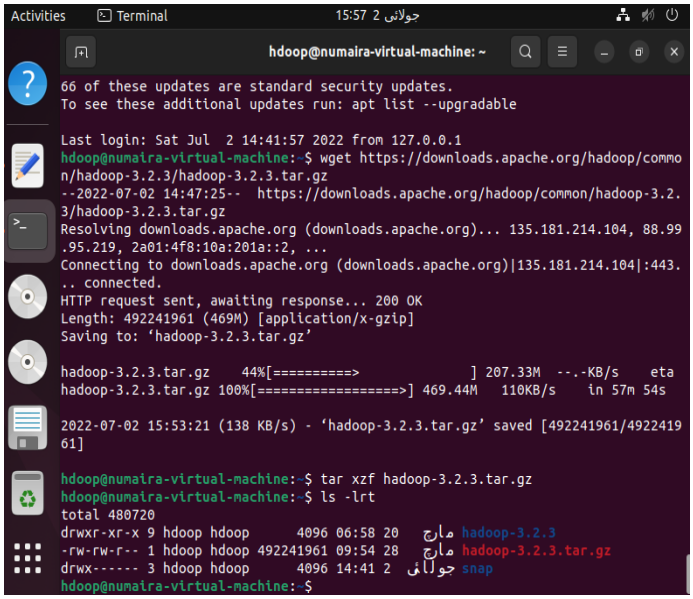
For downloading hadoop from this location.

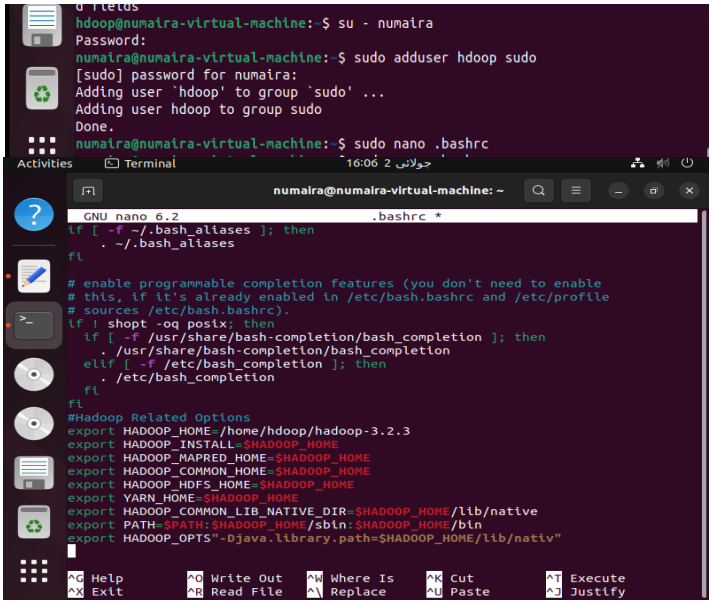
tar xzf hadoop-3.2.3.tar.gz

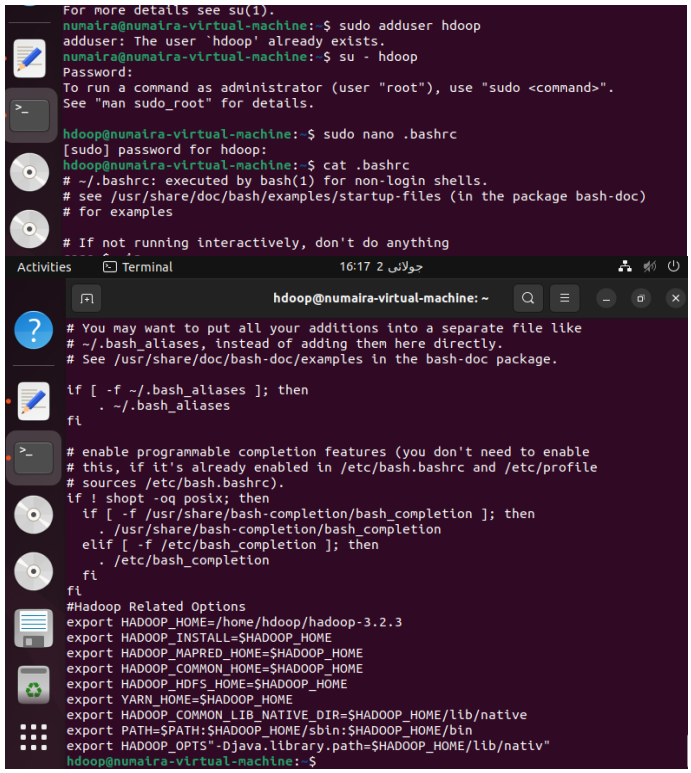
For unzipping the file.

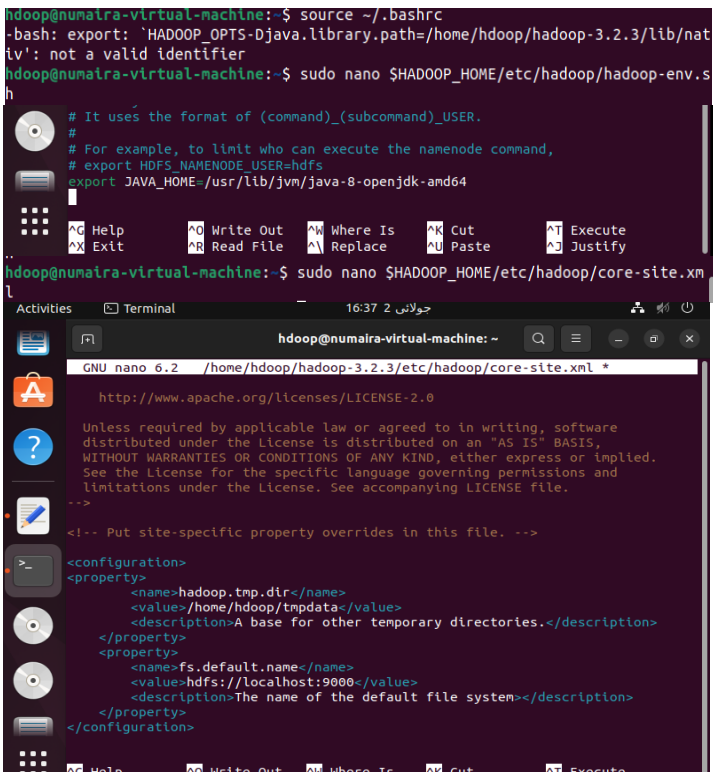
Then we edited the 6 important files.

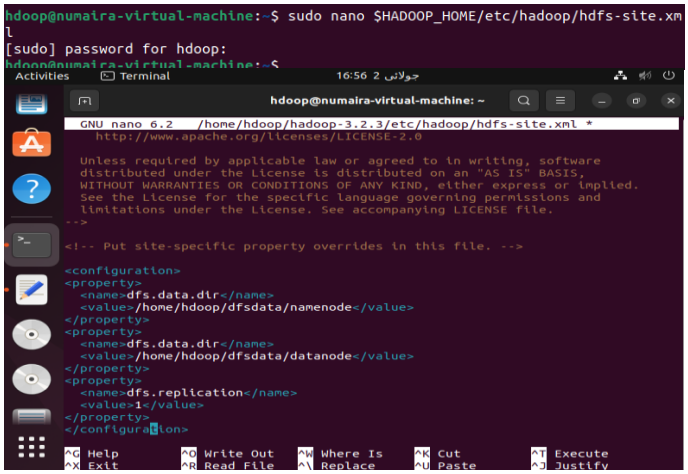
1) .bashrc
2) hadoop-env.sh
3) core-site.xml
4) hdfs-site.xml
5) mapred-site.xml
6) yarn-site.xml

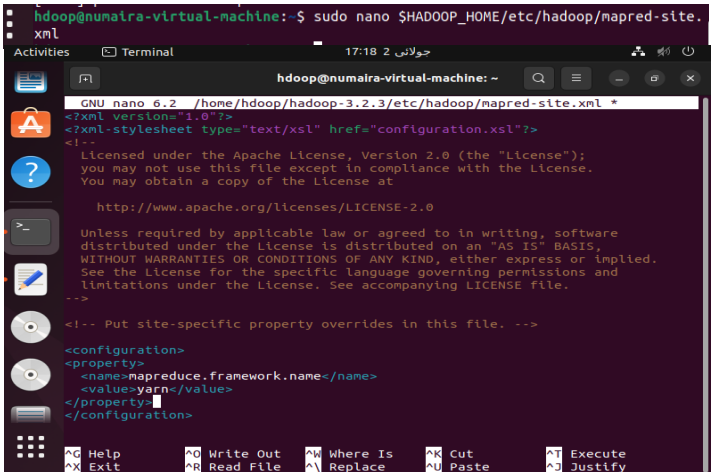

Activities
Terminal
15:57 2 جولائی
hdoop@numaira-virtual-machine: ~
66 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable
Last login: Sat Jul 2 14:41:57 2022 from 127.0.0.1
hdoop@numaira-virtual-machine:~$ wget https://downloads.apache.org/hadoop/common/hadoop-3.2.3/hadoop-3.2.3.tar.gz
--2022-07-02 14:47:25-- https://downloads.apache.org/hadoop/common/hadoop-3.2.3/hadoop-3.2.3.tar.gz
Resolving downloads.apache.org (downloads.apache.org)... 135.181.214.104, 88.99.95.219, 2a01:4f8:10a:201a::2, ...
Connecting to downloads.apache.org (downloads.apache.org)|135.181.214.104|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 492241961 (469M) [application/x-gzip]
Saving to: 'hadoop-3.2.3.tar.gz'
hadoop-3.2.3.tar.gz 44%[==========>] 207.33M --.-KB/s eta
hadoop-3.2.3.tar.gz 100%[==========>] 469.44M 110KB/s in 57m 54s
2022-07-02 15:53:21 (138 KB/s) - 'hadoop-3.2.3.tar.gz' saved [492241961/492241961]
hdoop@numaira-virtual-machine:~$ tar xzf hadoop-3.2.3.tar.gz
hdoop@numaira-virtual-machine:~$ ls -lrt
total 480720
drwxr-xr-x 9 hdoop hdoop 4096 06:58 20 مارچ hadoop-3.2.3
-rw-rw-r-- 1 hdoop hdoop 492241961 09:54 28 مارچ hadoop-3.2.3.tar.gz
drwx------ 3 hdoop hdoop 4096 14:41 2 جولائی snap
hdoop@numaira-virtual-machine:~$

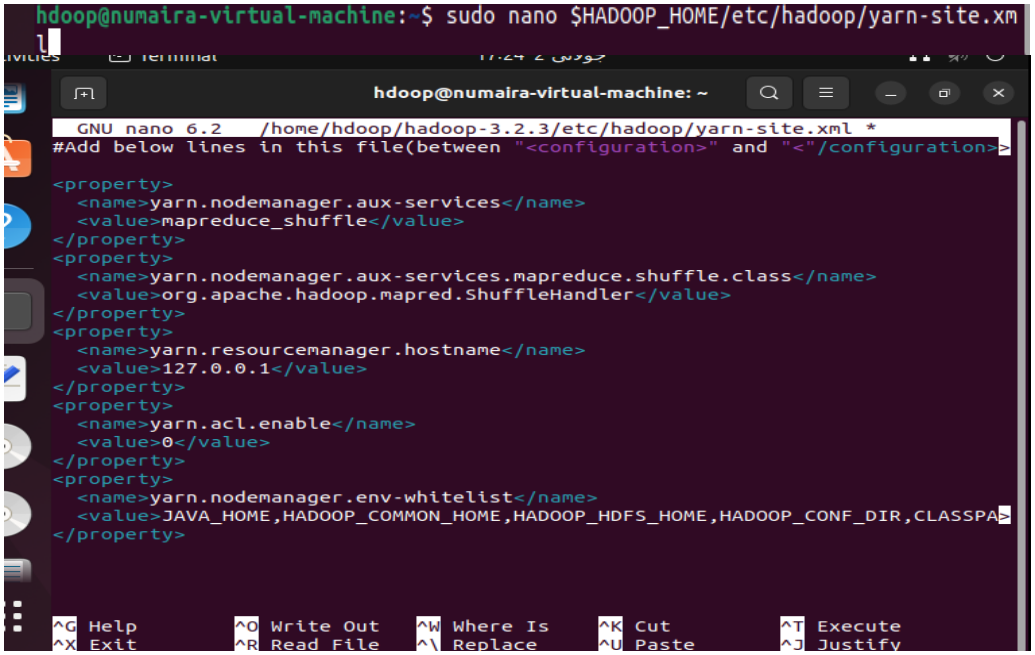

hdoop@numaira-virtual-machine:~$ su - numaira
Password:
numaira@numaira-virtual-machine:~$ sudo adduser hdoop sudo
[sudo] password for numaira:
Adding user `hdoop' to group `sudo' ...
Adding user hdoop to group sudo
Done.
numaira@numaira-virtual-machine:~$ sudo nano .bashrc
Activities
Terminal
16:06 2 جولائی
numaira@numaira-virtual-machine: ~
GNU nano 6.2 .bashrc *
if [-f ~/.bash_aliases]; then
. ~/.bash_aliases
fi
# enable programmable completion features (you don't need to enable
# 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
#Hadoop Related Options
export HADOOP_HOME=/home/hdoop/hadoop-3.2.3
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/nativ"
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify


For more details see su(1).
numaira@numaira-virtual-machine:~$ sudo adduser hdoop
adduser: The user `hdoop' already exists.
numaira@numaira-virtual-machine:~$ su - hdoop
Password:
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
hdoop@numaira-virtual-machine:~$ sudo nano .bashrc
[sudo] password for hdoop:
hdoop@numaira-virtual-machine:~$ cat .bashrc
~/.bashrc: executed by bash(1) for non-login shells.
see /usr/share/doc/bash/examples/startup-files (in the package bash-doc)
for examples
If not running interactively, don't do anything
Activities
Terminal
16:17 2 جولائی
hdoop@numaira-virtual-machine: ~
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.
if [-f ~/.bash_aliases]; then
. ~/.bash_aliases
fi
# enable programmable completion features (you don't need to enable
# 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
#Hadoop Related Options
export HADOOP_HOME=/home/hdoop/hadoop-3.2.3
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/nativ"
hdoop@numaira-virtual-machine:~$


hdoop@numaira-virtual-machine:~$ source ~/.bashrc
-bash: export: `HADOOP_OPTS-Djava.library.path=/home/hdoop/hadoop-3.2.3/lib/nativ': not a valid identifier
hdoop@numaira-virtual-machine:~$ sudo nano $HADOOP_HOME/etc/hadoop/hadoop-env.sh
It uses the format of (command)_(subcommand)_USER.
#
For example, to limit who can execute the namenode command,
export HDFS_NAMENODE_USER=hdfs
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify
hdoop@numaira-virtual-machine:~$ sudo nano $HADOOP_HOME/etc/hadoop/core-site.xml
Activities
Terminal
16:37 2 جولائی
hdoop@numaira-virtual-machine: ~
GNU nano 6.2 /home/hdoop/hadoop-3.2.3/etc/hadoop/core-site.xml *
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.
-->
<!-- Put site-specific property overrides in this file. -->
<configuration>
<property>
<name>hadoop.tmp.dir</name>
<value>/home/hdoop/tmpdata</value>
<description>A base for other temporary directories.</description>
</property>
<property>
<name>fs.default.name</name>
<value>hdfs://localhost:9000</value>
<description>The name of the default file system></description>
</property>
</configuration>


hdoop@numaira-virtual-machine:~$ sudo nano $HADOOP_HOME/etc/hadoop/hdfs-site.xml
[sudo] password for hdoop:
Activities
Terminal
16:56 2 جولائی
hdoop@numaira-virtual-machine: ~
GNU nano 6.2 /home/hdoop/hadoop-3.2.3/etc/hadoop/hdfs-site.xml *
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.
-->
<!-- Put site-specific property overrides in this file. -->
<configuration>
<property>
<name>dfs.data.dir</name>
<value>/home/hdoop/dfsdata/namenode</value>
</property>
<property>
<name>dfs.data.dir</name>
<value>/home/hdoop/dfsdata/datanode</value>
</property>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
</configuration>
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify


hdoop@numaira-virtual-machine:~$ sudo nano $HADOOP_HOME/etc/hadoop/mapred-site.xml
Activities
Terminal
17:18 2 جولائی
hdoop@numaira-virtual-machine: ~
GNU nano 6.2 /home/hdoop/hadoop-3.2.3/etc/hadoop/mapred-site.xml *
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
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.
-->
<!-- Put site-specific property overrides in this file. -->
<configuration>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
</configuration>
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify

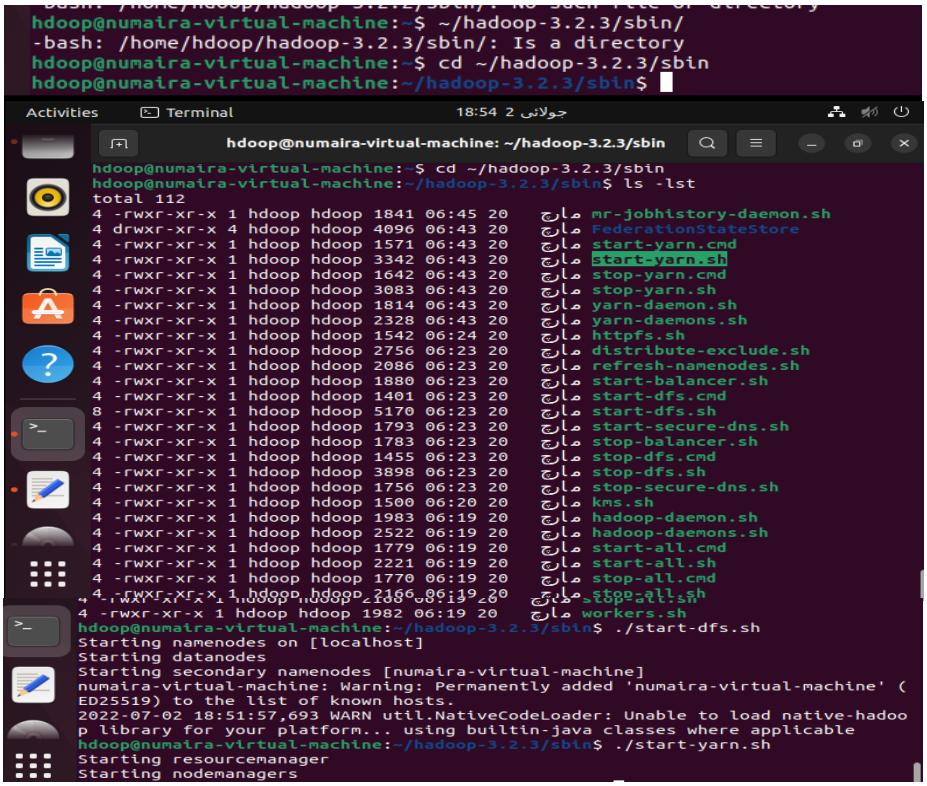


LAUNCHING HADOOP

hdfs namenode –format

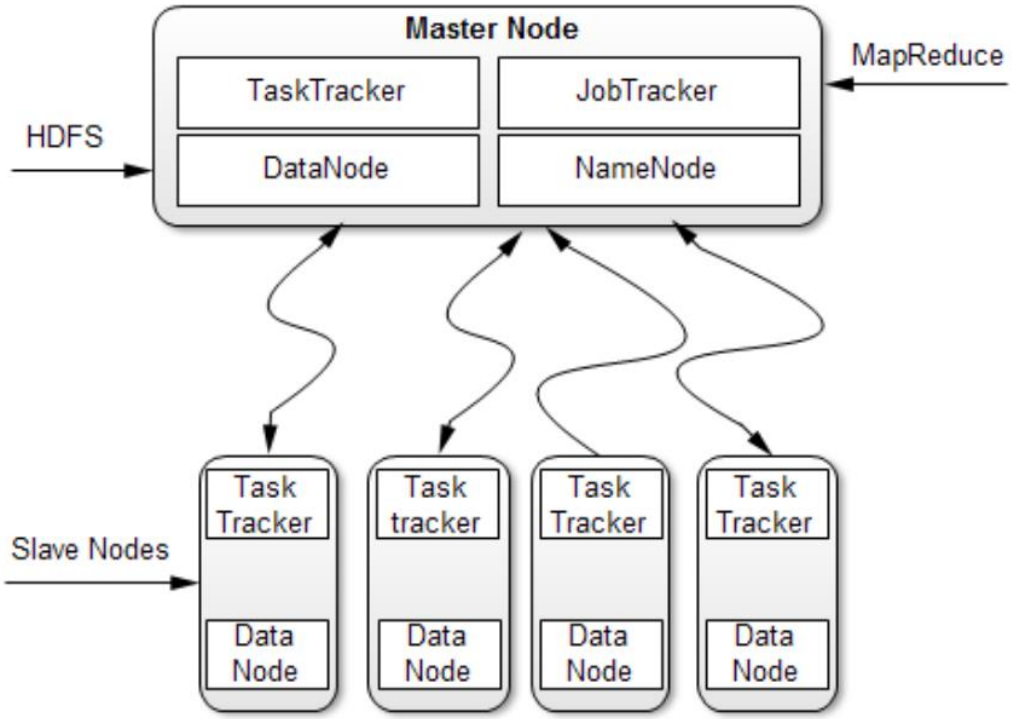
./start-dfs.sh

Then we move a local file to hdfs



MASTER & SLAVE

Hadoop is based on Master/Slave architecture. Master is responsible for NameNode and JobTracker while Slave is responsible for DataNode and TaskTracker. Job Tracker initiates the task and keeps track of each task. TaskTracker manages local data processing and collects its result as per the requirement of application. It also sends a report on the progress of tasks to JobTracker. NameNode and DataNode are responsible for HDFS tasks while JobTracker and TaskTracker are responsible for MapReduce tasks.



SSH MASTER SLAVE

Copy the public key of user master into the slave

ssh-copy-id-i.ssh/id_rsa.pub hdoop@slave

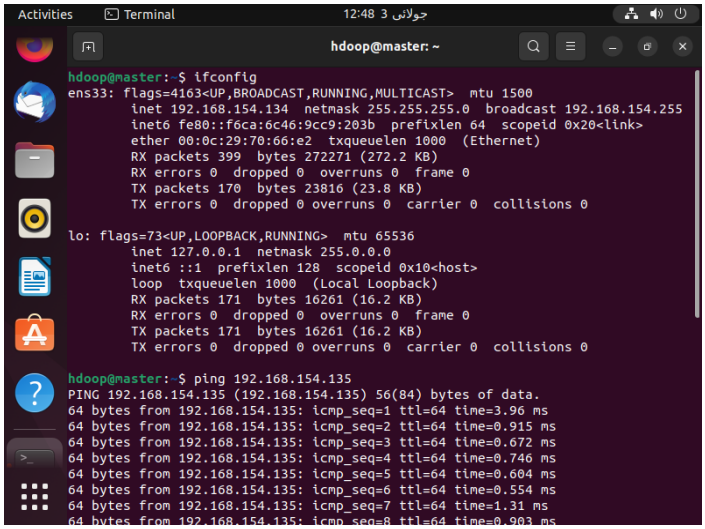
This command will prompt you for the login password for user hdoop on slave, then copy the public SSH key for you, creating directory and fixing the permissions as necessary.

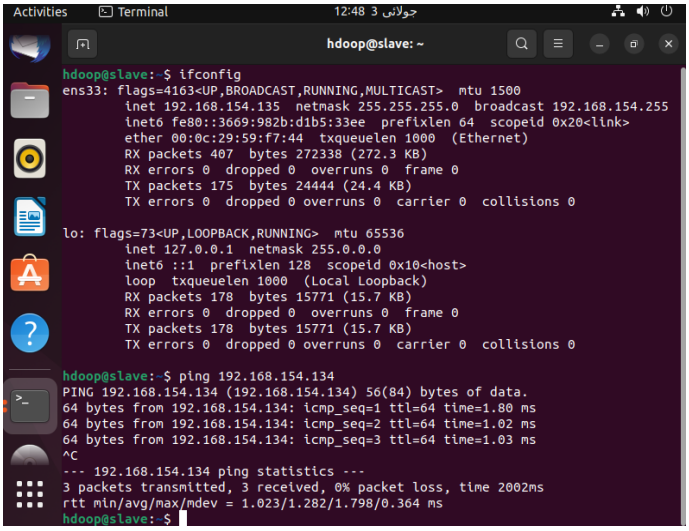
Connecting from master to master

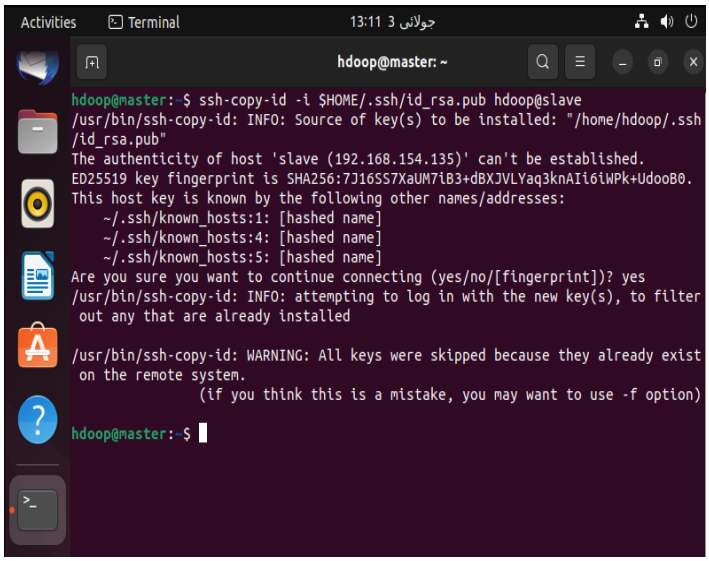
ssh master

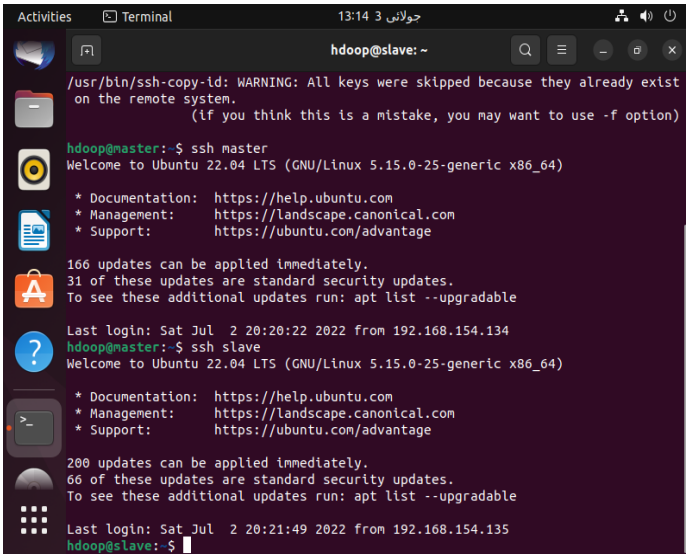
Connecting from master to slave

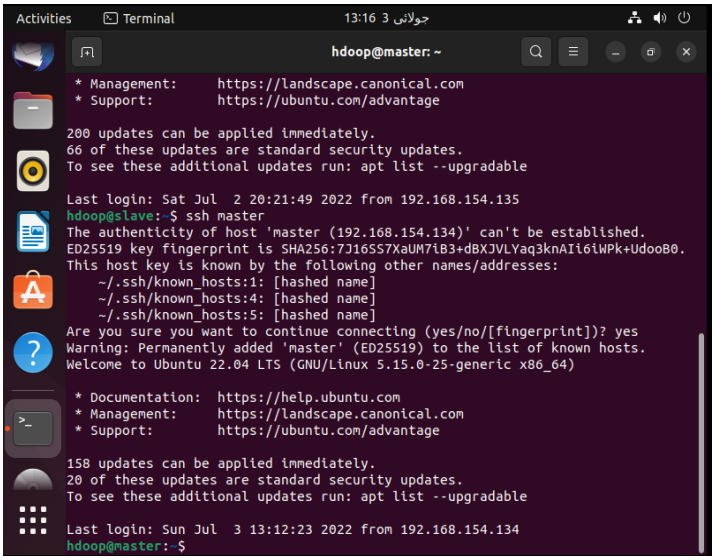
Ssh slave


Activities
Terminal
12:48 3 جولائی
hdoop@master: ~
hdoop@master:~$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.154.134 netmask 255.255.255.0 broadcast 192.168.154.255
inet6 fe80::f6ca:6c46:9cc9:203b prefixlen 64 scopeid 0x20<link>
ether 00:0c:29:70:66:e2 txqueuelen 1000 (Ethernet)
RX packets 399 bytes 272271 (272.2 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 170 bytes 23816 (23.8 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Local Loopback)
RX packets 171 bytes 16261 (16.2 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 171 bytes 16261 (16.2 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
hdoop@master:~$ ping 192.168.154.135
PING 192.168.154.135 (192.168.154.135) 56(84) bytes of data.
64 bytes from 192.168.154.135: icmp_seq=1 ttl=64 time=3.96 ms
64 bytes from 192.168.154.135: icmp_seq=2 ttl=64 time=0.915 ms
64 bytes from 192.168.154.135: icmp_seq=3 ttl=64 time=0.672 ms
64 bytes from 192.168.154.135: icmp_seq=4 ttl=64 time=0.746 ms
64 bytes from 192.168.154.135: icmp_seq=5 ttl=64 time=0.604 ms
64 bytes from 192.168.154.135: icmp_seq=6 ttl=64 time=0.554 ms
64 bytes from 192.168.154.135: icmp_seq=7 ttl=64 time=1.31 ms


Activities
Terminal
12:48 3 جولائی
hdoop@slave: ~
hdoop@slave:~$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.154.135 netmask 255.255.255.0 broadcast 192.168.154.255
inet6 fe80::3669:982b:d1b5:33ee prefixlen 64 scopeid 0x20<link>
ether 00:0c:29:59:f7:44 txqueuelen 1000 (Ethernet)
RX packets 407 bytes 272338 (272.3 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 175 bytes 24444 (24.4 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Local Loopback)
RX packets 178 bytes 15771 (15.7 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 178 bytes 15771 (15.7 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
hdoop@slave:~$ ping 192.168.154.134
PING 192.168.154.134 (192.168.154.134) 56(84) bytes of data.
64 bytes from 192.168.154.134: icmp_seq=1 ttl=64 time=1.80 ms
64 bytes from 192.168.154.134: icmp_seq=2 ttl=64 time=1.02 ms
64 bytes from 192.168.154.134: icmp_seq=3 ttl=64 time=1.03 ms
^C
--- 192.168.154.134 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 1.023/1.282/1.798/0.364 ms
hdoop@slave:~$


Activities
Terminal
13:11 3 جولائی
hdoop@master: ~
hdoop@master:~$ ssh-copy-id -i $HOME/.ssh/id_rsa.pub hdoop@slave
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/hdoop/.ssh/id_rsa.pub"
The authenticity of host 'slave (192.168.154.135)' can't be established.
ED25519 key fingerprint is SHA256:7J16SS7XaUM7iB3+dBXJVLYaq3knAIi6iWPk+UdooB0.
This host key is known by the following other names/addresses:
~/.ssh/known_hosts:1: [hashed name]
~/.ssh/known_hosts:4: [hashed name]
~/.ssh/known_hosts:5: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: WARNING: All keys were skipped because they already exist on the remote system.
(if you think this is a mistake, you may want to use -f option)
hdoop@master:~$

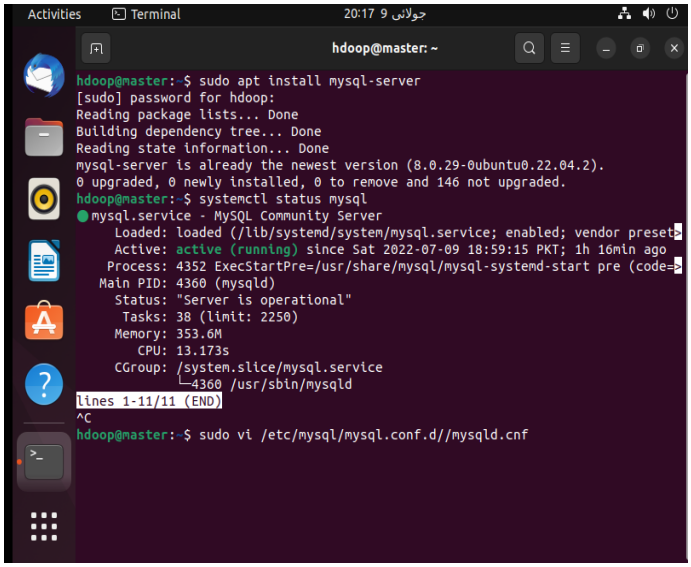

Activities
Terminal
13:14 3 جولائی
hdoop@slave: ~
/usr/bin/ssh-copy-id: WARNING: All keys were skipped because they already exist on the remote system.
(if you think this is a mistake, you may want to use -f option)
hdoop@master:~$ ssh master
Welcome to Ubuntu 22.04 LTS (GNU/Linux 5.15.0-25-generic x86_64)
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage
166 updates can be applied immediately.
31 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable
Last login: Sat Jul 2 20:20:22 2022 from 192.168.154.134
hdoop@master:~$ ssh slave
Welcome to Ubuntu 22.04 LTS (GNU/Linux 5.15.0-25-generic x86_64)
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage
200 updates can be applied immediately.
66 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable
Last login: Sat Jul 2 20:21:49 2022 from 192.168.154.135
hdoop@slave:~$

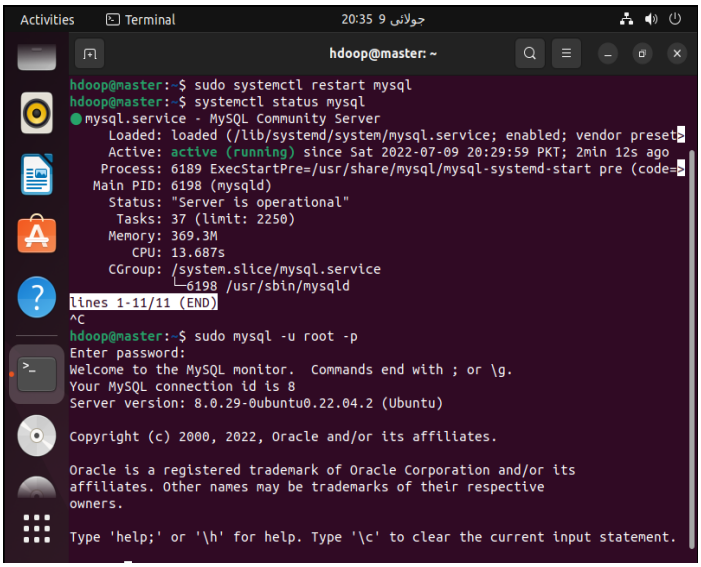

Activities
Terminal
13:16 3 جولائی
hdoop@master: ~
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage
200 updates can be applied immediately.
66 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable
Last login: Sat Jul 2 20:21:49 2022 from 192.168.154.135
hdoop@slave:~$ ssh master
The authenticity of host 'master (192.168.154.134)' can't be established.
ED25519 key fingerprint is SHA256:7J16SS7XaUM7iB3+dBXJVLYaq3knAIi6iWPk+UdooB0.
This host key is known by the following other names/addresses:
~/.ssh/known_hosts:1: [hashed name]
~/.ssh/known_hosts:4: [hashed name]
~/.ssh/known_hosts:5: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'master' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04 LTS (GNU/Linux 5.15.0-25-generic x86_64)
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage
158 updates can be applied immediately.
20 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable
Last login: Sun Jul 3 13:12:23 2022 from 192.168.154.134
hdoop@master:~$

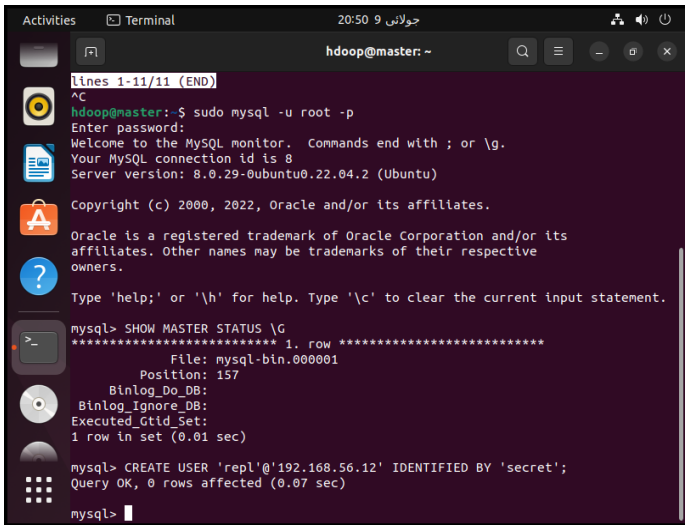
Numaira Zaib

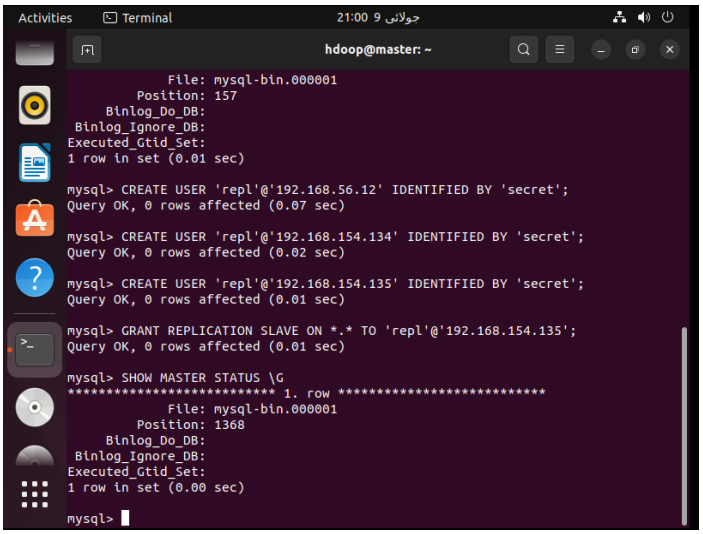
MySQL DATABASE REPLICATION WITH MASTER & SLAVE

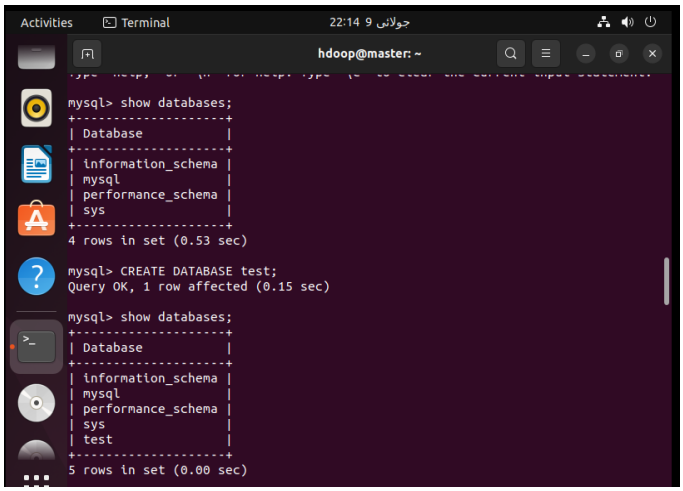
MASTER MACHINE

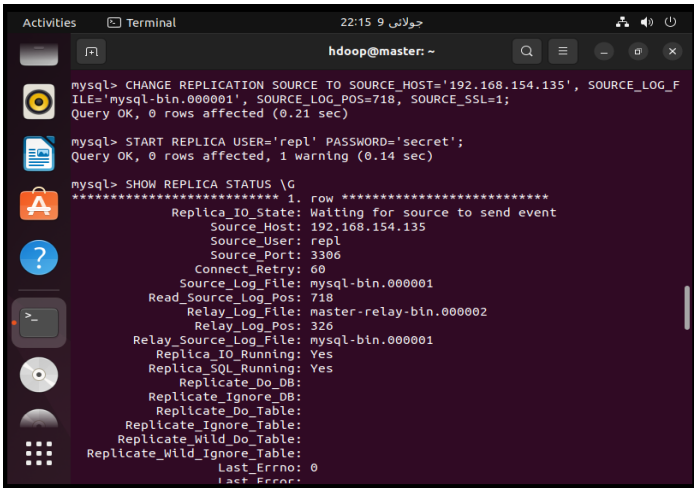


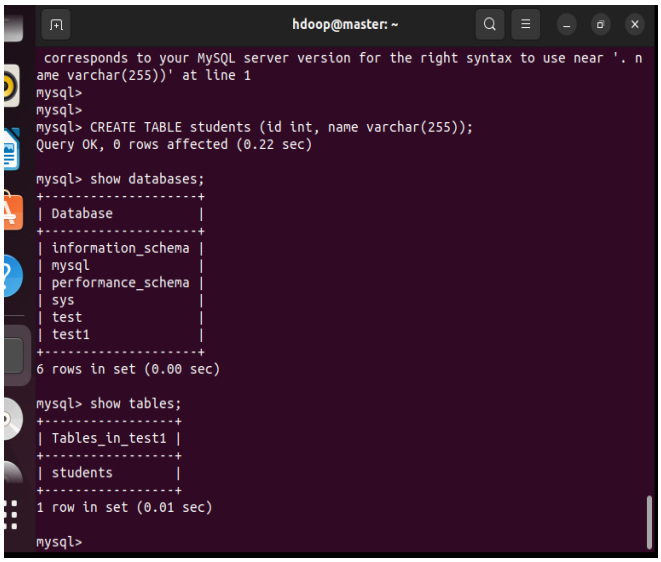


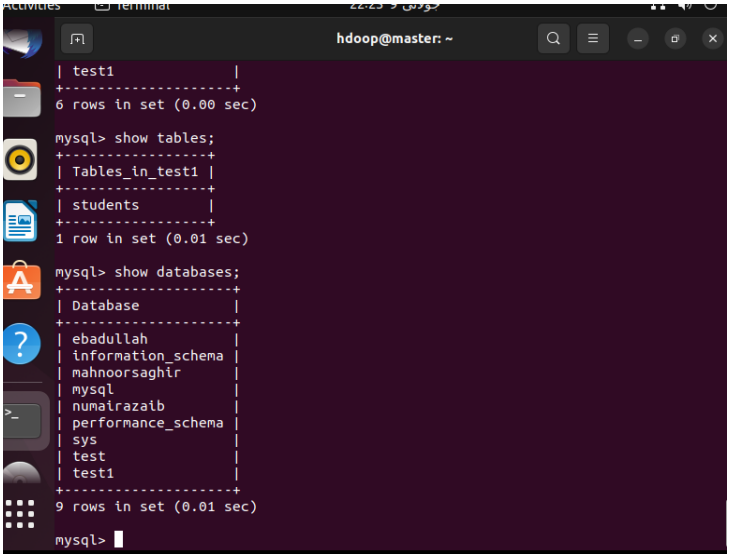




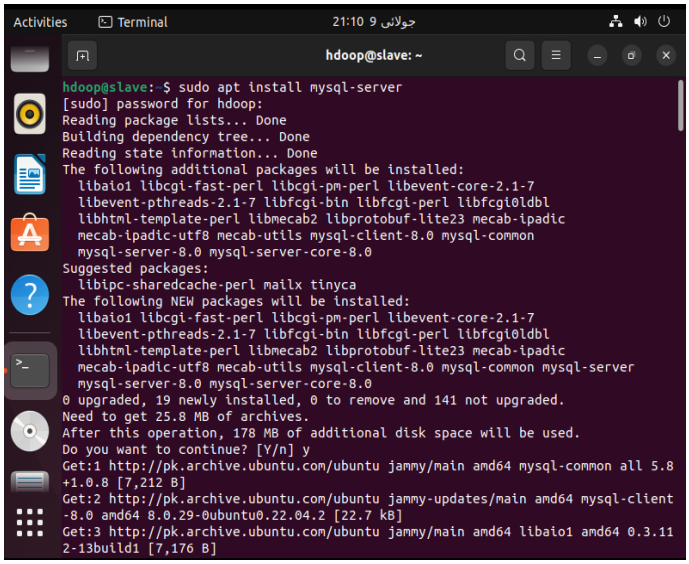


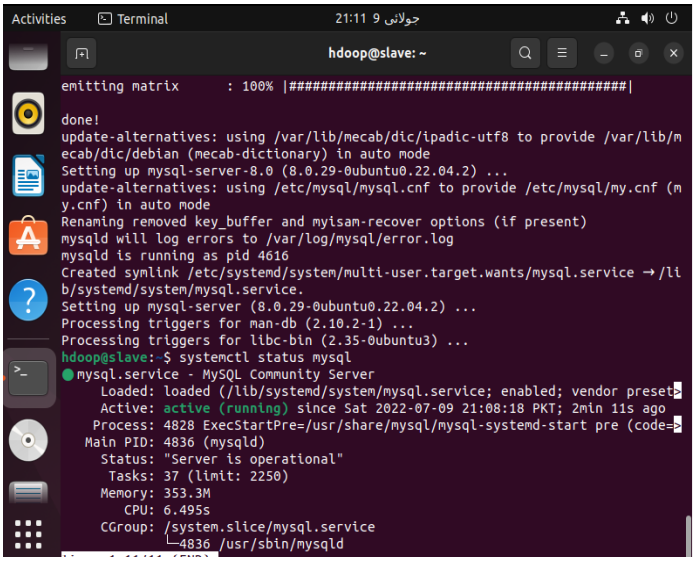


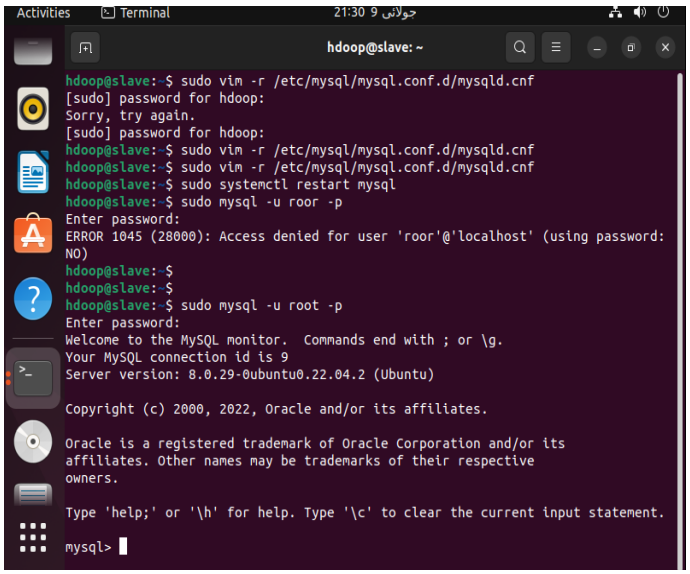


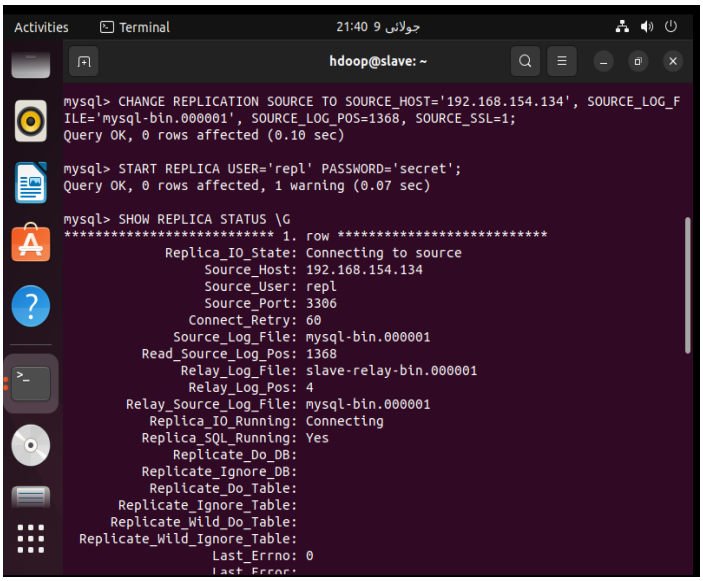


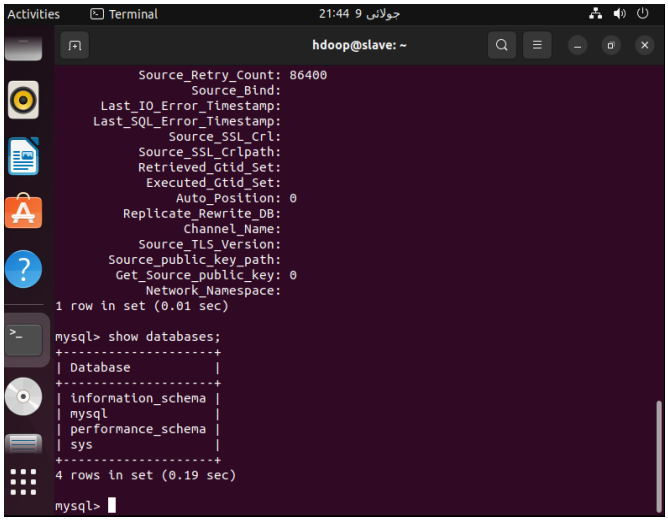
SLAVE MACHINE

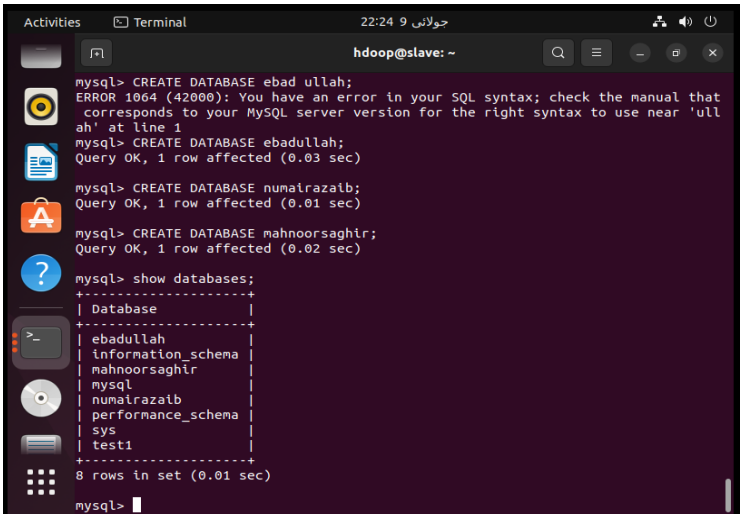












The benefits of master-slave replication

1. **Data is more secure:** Made data redundancy, No data loss due to single server downtime
2. **Improved performance:** One master, many followers, Different users read from different databases, Performance improvement
3. **Better scalability:** When the flow rate increases, you can easily add from the server, which does not affect the use of the system
4. **Load balancing:** One master and many slaves share the host task, Load balancing is done.

MAPREDUCE

MapReduce is a programming paradigm that enables massive scalability across hundreds or thousands of servers in a Hadoop cluster.

WORDCOUNT PROGRAM (JAVA)

WordCount is a simple application that counts the number of occurrences of each word in a given input set.

- **sudo su - hadoop**

We can't run the HDFS command directly. First of all we have to run all the demons and we have to ensure that all the demons of Hadoop is up and running. Then type ls and you can find out that there's a script start-all.sh. This will start all the demons of Hadoop. Then we check once by typing jps for ensuring that all demons of Hadoop is running. Now we can work with the HDFS file system commands.

- **hdfs dfs – help**

  It shows all the existing command inside the Hadoop.

- **hdfs fsck /**

  For checking the health of file system.

- **hdfs dfs - ls ~/**

  List all the files in a particular path.

- **hdfs dfs - ls ~/f1**

  For checking what is there inside this particular directory.

- **hdfs dfs - mkdir - p ~/test**

  For creating a folder test if it is present then it will leave it as it is and it is not present it will create a test folder as we used a prefix p.

- **hdfs dfs - ls ~/**

  For checking whether the folder is created or not.

- **hdfs dfs - rm - r ~/test/**

  This command is for recursively remove all the files first and then remove the directory.

- **hdfs dfs - touchz ~/sample.txt**

  This command is used for creating a empty file inside the HDFS.

- **hdfs dfs - du - s ~/sample.txt**

  This command is used to check the size of the file.

- **hdfs dfs - appendToFile - ~/sample.txt**

  This command is used for adding content to the file by using terminal.

- **hdfs dfs - cat ~/sample.txt**

  This command is used to see the contents of the file.

- **vim host.txt**

  This is the second way of creating a file inside the HDFS

- **hdfs dfs – copyFromLocal host.txt ~/**

  This command is used for moving a particular file from NFS TO HDFS.

- **vim readme.md**

  For creating a new file.

- **hdfs dfs - put readme.md ~/**

  This command is used for recent version of hadoop to move file from NFS to HDFS.

- **hdfs dfs - copyToLocal ~/readme.md ~/sample/**

  This command is used to move file from HDFS to NFS.

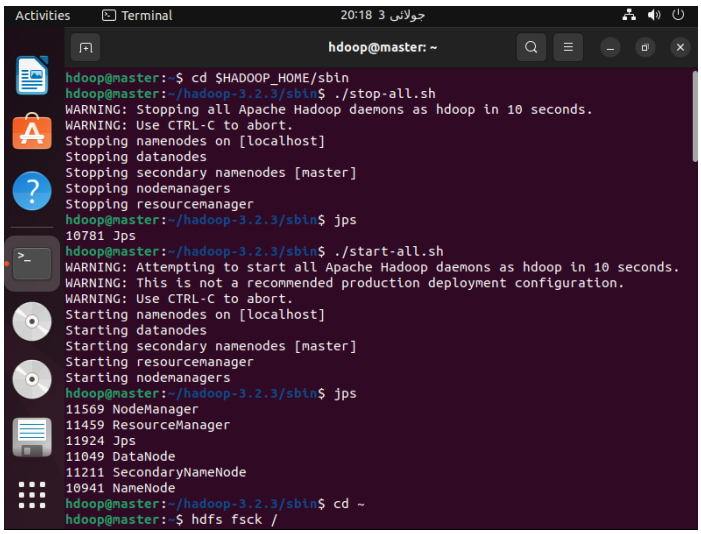
- **hdfs dfs - get ~/host.txt ~/sample/**

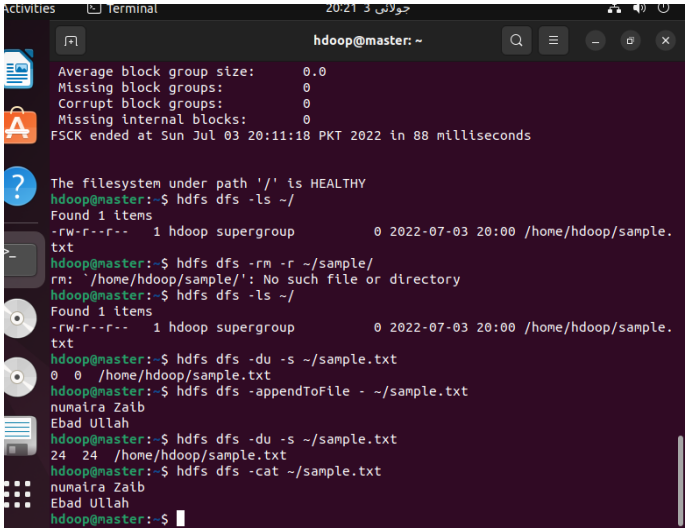
  This is the other way of getting the files from the HDFS to the local file system.

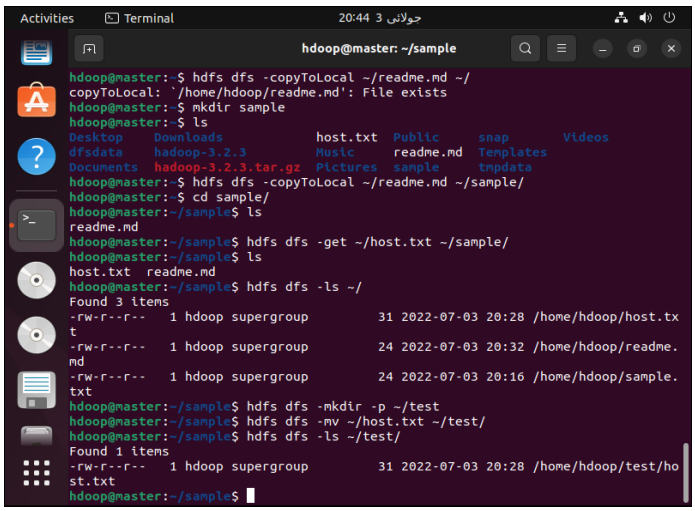
- **hdfs dfs – mv ~/host.txt ~/test/**

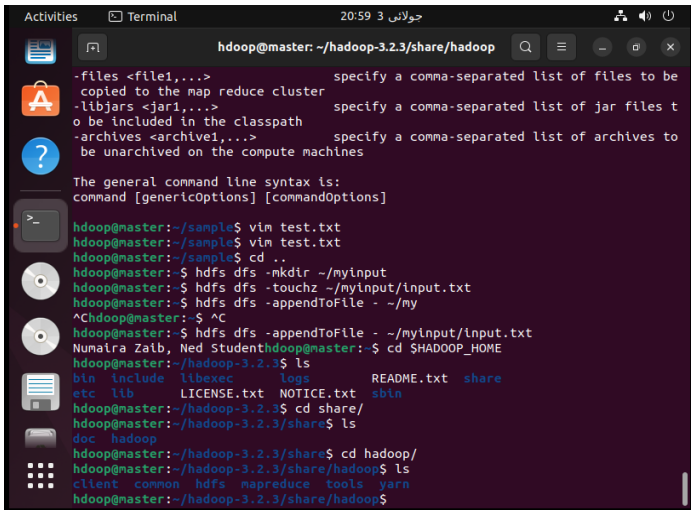
  mv command is going to helps us move the contents inside the HDFS File System itself,

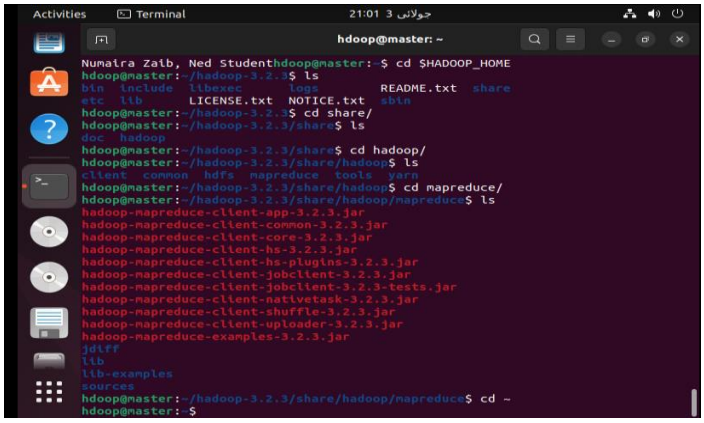
Numaira Zaib

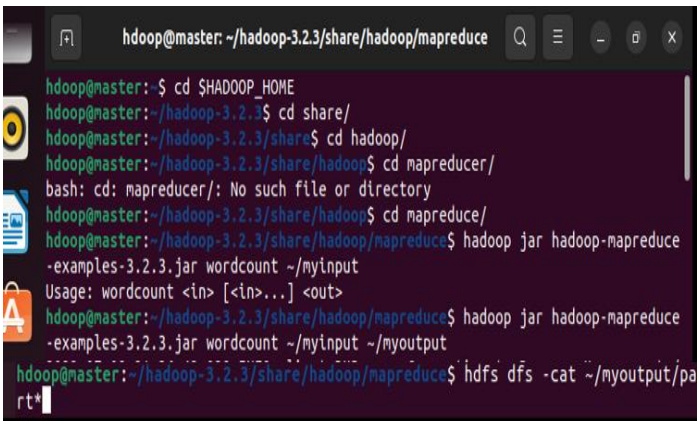












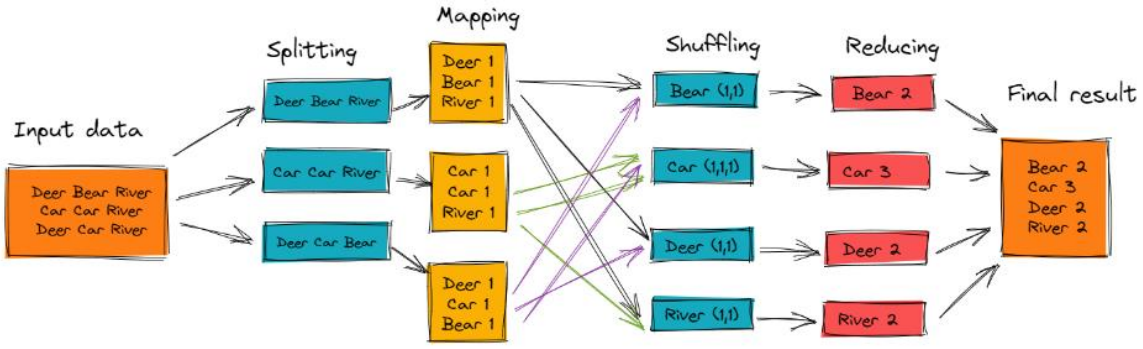
MAPREDUCE WITH PYTHON

MapReduce With Python (WORDCOUNT PROGRAM)

Mapper: It is the first phase of MapReduce application which is used to process a series of key-value pairs.

Shuffler/Combiner: It is the second phase of MapReduce application used to sort, group and shuffle the output coming from the Mapper function. This process of moving output from the mappers to the reducers is known as shuffling.

Reducer: It is the third phase of MapReduce application, it aggregates the values for each unique key and produces zero or more output key-value pairs.



Hadoop Streaming

The mapper and reducer are both executables that read input, line by line, from the standard input (stdin), and write output to the standard output (stdout)

- Mapper
  - Reads the input from stdin line by line.
  - Splits the word separated by tab, space or comma etc.
  - Pairs key-values.

- Reducer
  - Reads the result of mapper from stdout
  - Sums the occurrences of each word
  - Writes the result to stdout

WORD COUNT PROGRAM

mapper.py

First the path is provided then we used sys library. Reading the line from stdin. If there are spaces between the lines that will be removed using strip function then we have only

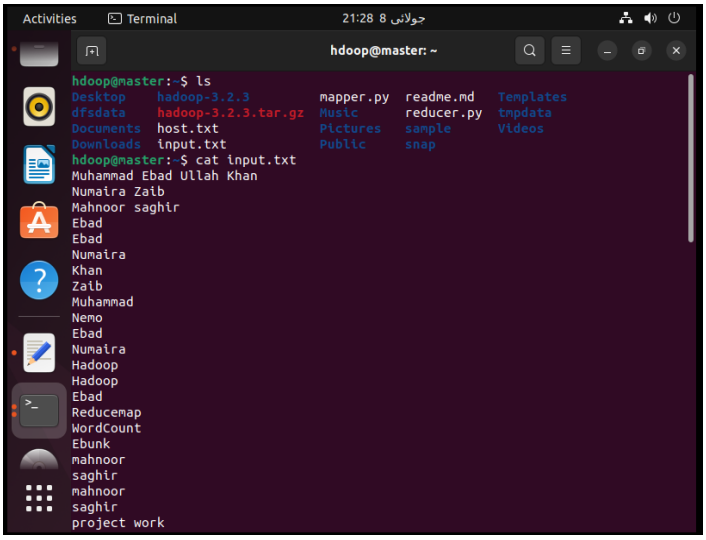
consecutive lines. From the lines, we will be splitting the words one by one and then for each word we will be assigning the value “1” so this is called peering.

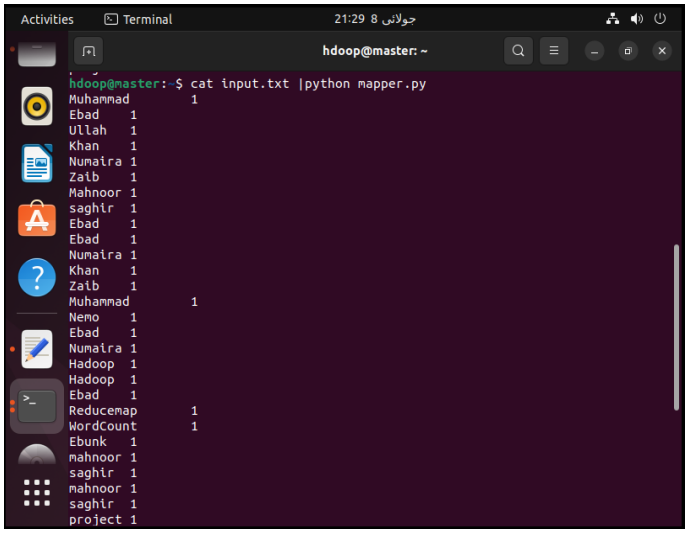
reducer.py

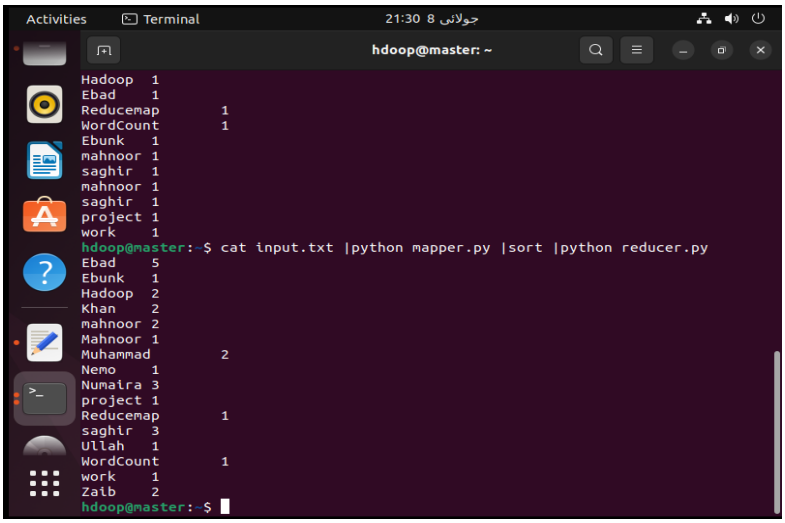
We used sys library, then we initialized 2 variables prev_word and prev_count. Then reading the output of the mapper line by line so it is stripping again. One by one they are all read it from the standard input and they will be assigned to these 2 variables “word” and “count”. Then converting this count by typecasting to integer. Then we put conditions to check if prev_word is equal to word so just increment its counter or else if there is any prev_word, then just print it. Then again initializing the prev_count to the count and prev_word to the word. Then we used one last if statement for checking whether the prev_word is equal to word and if it is, then also print the last word.

Testing locally

To test the python programs locally before running them as a Map-Reduce job.
It is highly recommended to test all programs locally before running them across a Hadoop cluster.
Once the mapper and reducer programs are executing successfully against tests, they can be run as a MapReduce application using the Hadoop streaming utility.







RPC IN HADOOP

It allows one computer program to remotely call another computer's subprogram without having to care about the details of the underlying network communication, which is transparent to us. Therefore, it is often used in distributed network communications.

Hadoop's inter-process interaction is carried out through RPC, such as the direct connection between Namenode and Datanode, and between Jobtracker and Tasktracker.

Therefore, it can be said that the operation of Hadoop is based on RPC . Hadoop RPC is an important part of Hadoop, which provides object call functions in a distributed environment. The source code is in org.apache.hadoop.ipc. HBase has almost completely copied the source code of this part, but has changed the configuration items.

RPC mechanism in Hadoop

Like other RPC frameworks, Hadoop RPC is divided into four parts:

Serialization layer: The information communicated between Client and Server uses the serialization class provided in Hadoop or the custom Writable type
Function call layer: Hadoop RPC realizes function call through dynamic proxy and java reflection
Network transmission layer: Hadoop RPC uses a socket mechanism based on TCP/IP
Server-side framework layer: RPC Server uses java NIO and an event-driven I/O model to improve the concurrent processing capabilities of RPC Server

Hadoop RPC is widely used throughout Hadoop, and the communication between Client, DataNode, and NameNode depends on it. For example: When we usually operate HDFS, we use the FileSystem class, which has a DFSClient object inside, which is responsible for dealing with the NameNode. At runtime, DFSClient creates a NameNode agent locally, and then operates the agent. The agent will call the method of the NameNode remotely through the network and return the value.

TRANSPARENCY IN HADOOP

HDFS implements transparent, end-to-end encryption. Once configured, data read from and written to special HDFS directories is transparently encrypted and decrypted without requiring changes to user application code. This encryption is also end-to-end, which means the data can only be encrypted and decrypted by the client. HDFS never stores or has access to unencrypted data or unencrypted data encryption keys. This satisfies two typical requirements for encryption: at-rest encryption (meaning data on persistent media, such as a disk) as well as in-transit encryption (e.g. when data is travelling over the network).

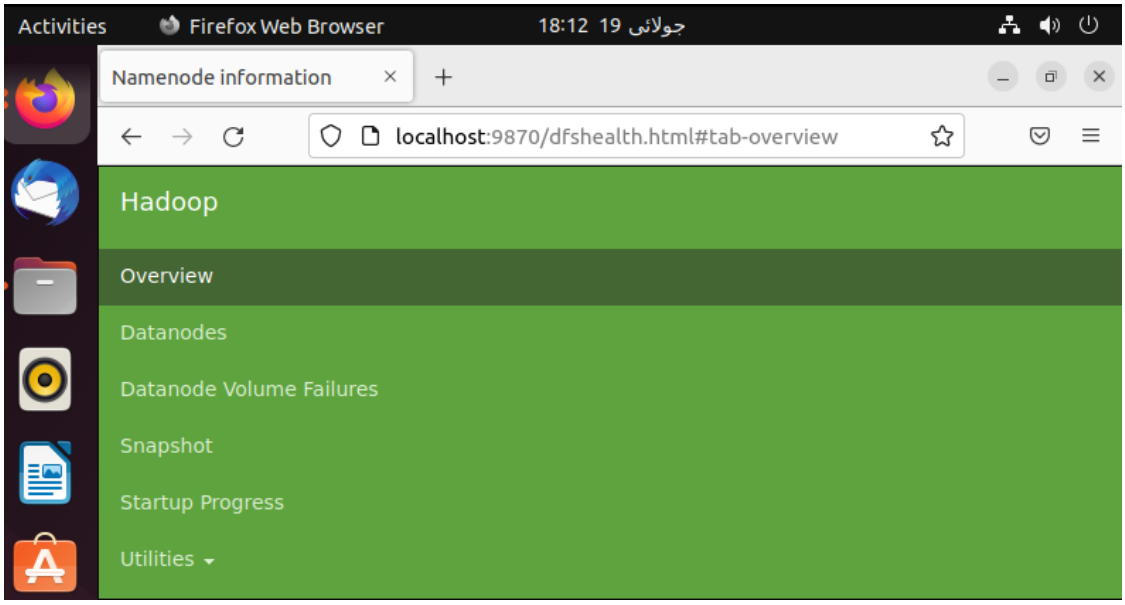
Encryption can be done at different layers in a traditional data management software/hardware stack. Choosing to encrypt at a given layer comes with different advantages and disadvantages.

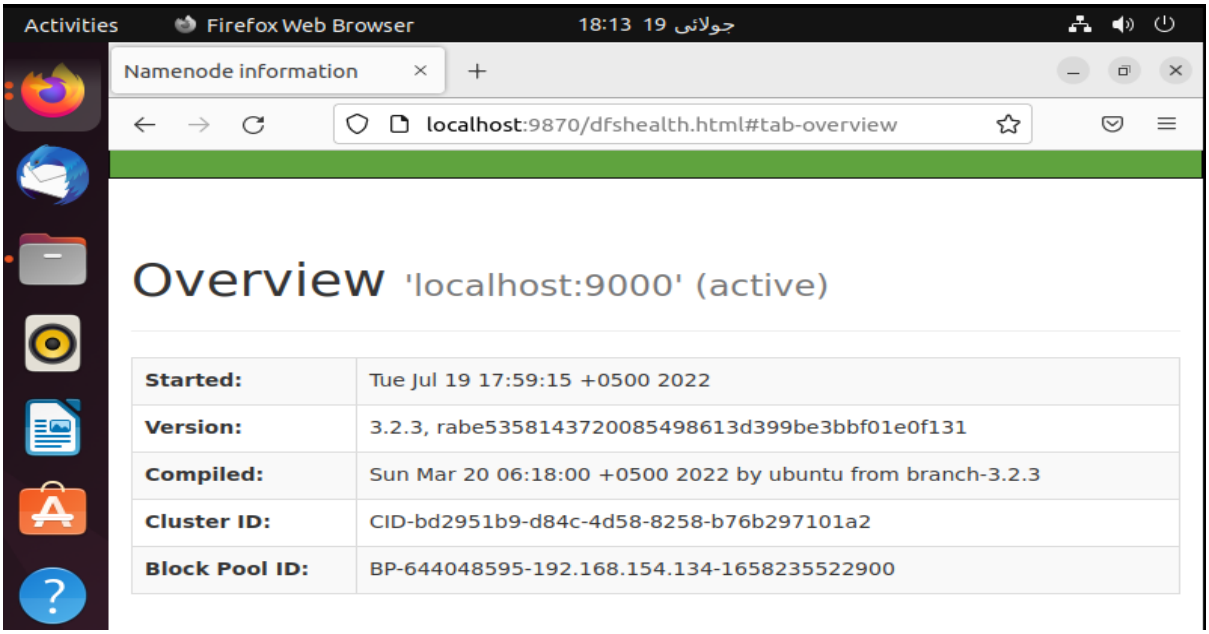
- **Application-level encryption**. This is the most secure and most flexible approach. The application has ultimate control over what is encrypted and can precisely reflect the requirements of the user. However, writing applications to do this is hard. This is also not an option for customers of existing applications that do not support encryption.
- **Database-level encryption**. Similar to application-level encryption in terms of its properties. Most database vendors offer some form of encryption. However, there can be performance issues. One example is that indexes cannot be encrypted.
- **Filesystem-level encryption**. This option offers high performance, application transparency, and is typically easy to deploy. However, it is unable to model some application-level policies. For instance, multi-tenant applications might want to encrypt based on the end user. A database might want different encryption settings for each column stored within a single file.
- **Disk-level encryption**. Easy to deploy and high performance, but also quite inflexible. Only really protects against physical theft.

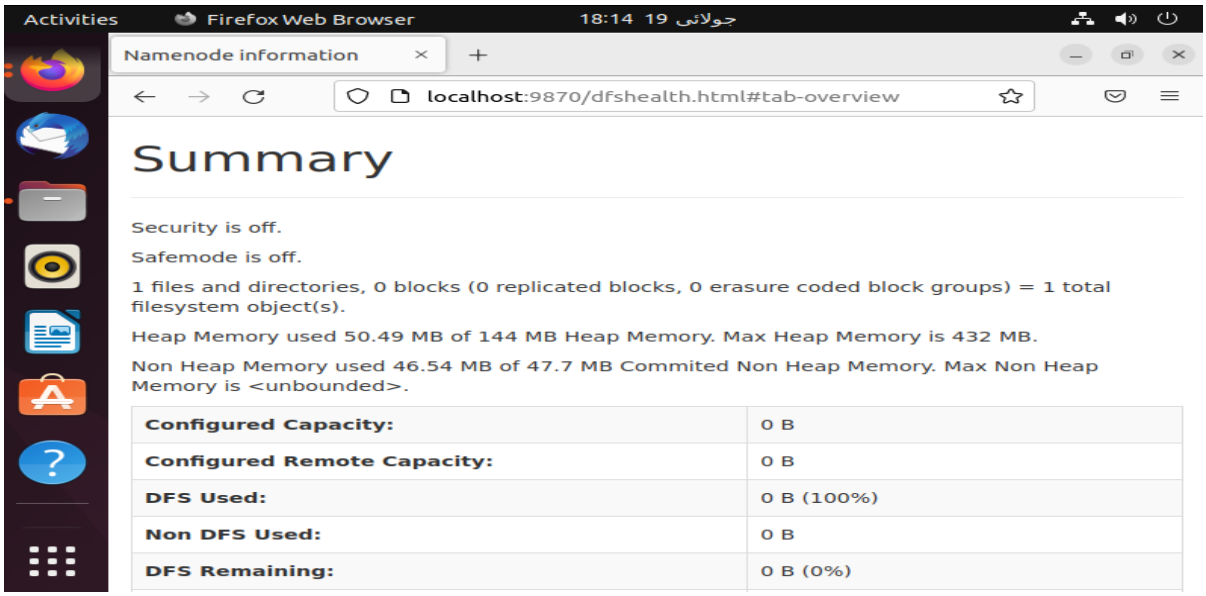
HDFS-level encryption fits between database-level and filesystem-level encryption in this stack. This has a lot of positive effects. HDFS encryption is able to provide good performance and existing Hadoop applications are able to run transparently on encrypted data. HDFS also has more context than traditional filesystems when it comes to making policy decisions.

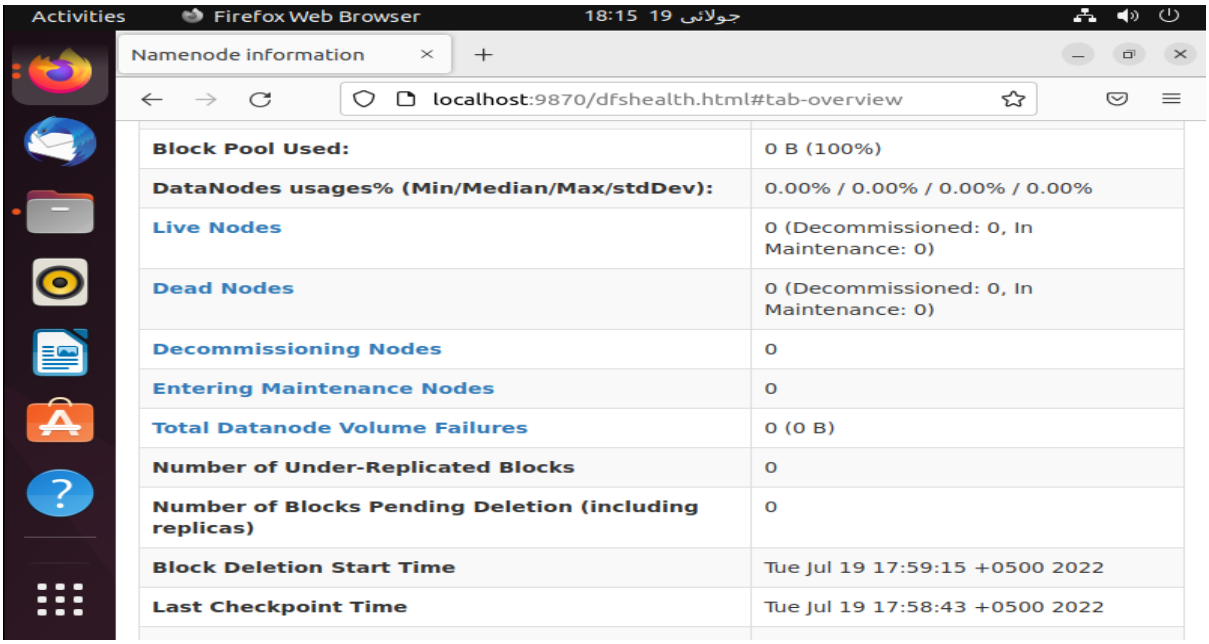
HDFS-level encryption also prevents attacks at the filesystem-level and below (so-called “OS-level attacks”). The operating system and disk only interact with encrypted bytes, since the data is already encrypted by HDFS.

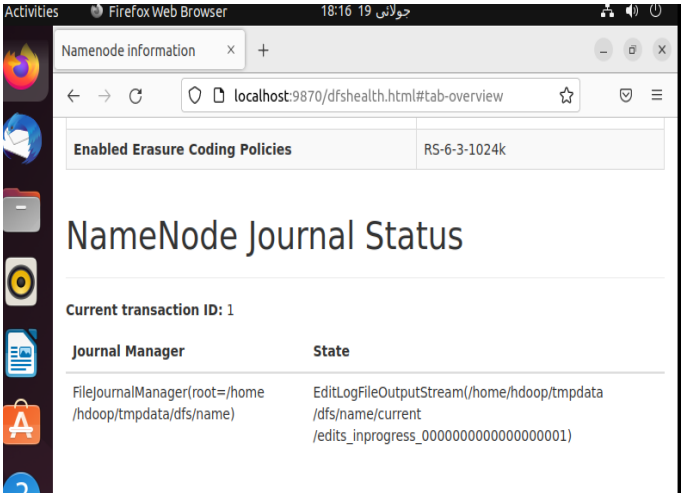
HADOOP WEB INTERFACE

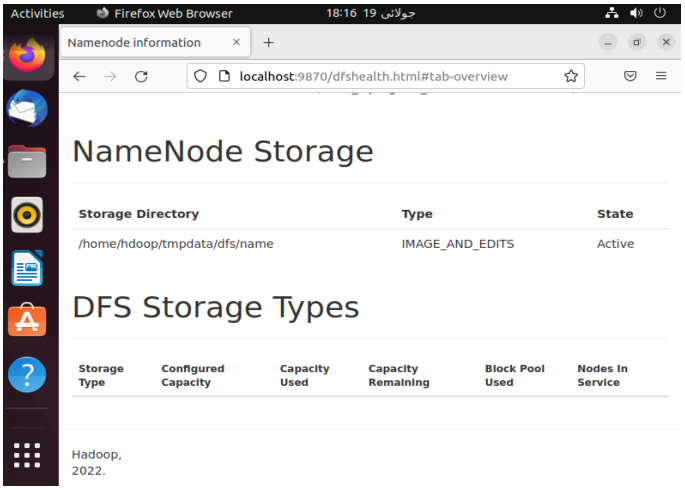


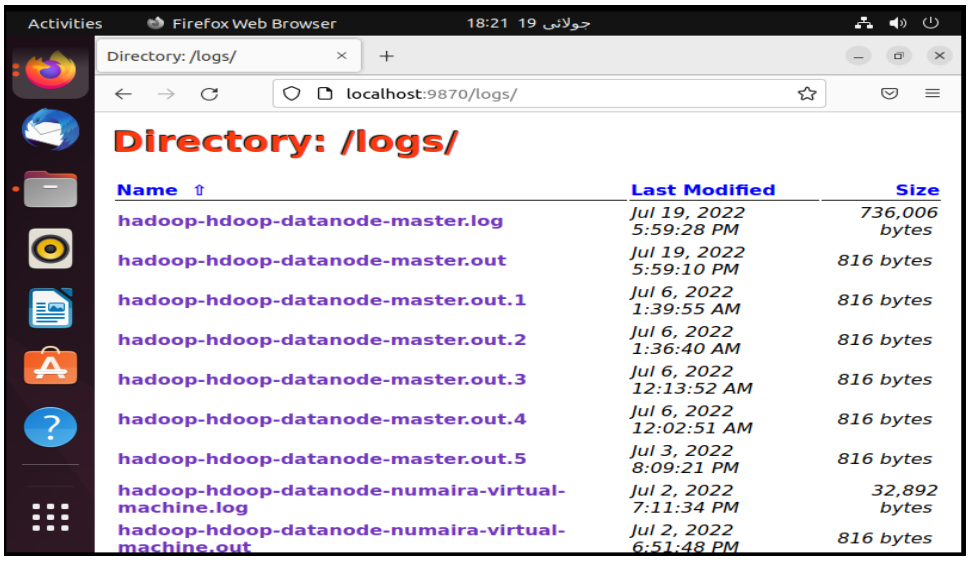

Activities
Firefox Web Browser
18:13 19 جولائی
Namenode information
localhost:9870/dfshealth.html#tab-overview
Overview 'localhost:9000' (active)
Started: Tue Jul 19 17:59:15 +0500 2022
Version: 3.2.3, rabe5358143720085498613d399be3bbf01e0f131
Compiled: Sun Mar 20 06:18:00 +0500 2022 by ubuntu from branch-3.2.3
Cluster ID: CID-bd2951b9-d84c-4d58-8258-b76b297101a2
Block Pool ID: BP-644048595-192.168.154.134-1658235522900


Activities
Firefox Web Browser
18:14 19 جولائی
Namenode information
localhost:9870/dfshealth.html#tab-overview
Summary
Security is off.
Safemode is off.
1 files and directories, 0 blocks (0 replicated blocks, 0 erasure coded block groups) = 1 total filesystem object(s).
Heap Memory used 50.49 MB of 144 MB Heap Memory. Max Heap Memory is 432 MB.
Non Heap Memory used 46.54 MB of 47.7 MB Commited Non Heap Memory. Max Non Heap Memory is <unbounded>.
Configured Capacity: 0 B
Configured Remote Capacity: 0 B
DFS Used: 0 B (100%)
Non DFS Used: 0 B
DFS Remaining: 0 B (0%)


Activities
Firefox Web Browser
18:15 19 جولائی
Namenode information
localhost:9870/dfshealth.html#tab-overview
Block Pool Used: 0 B (100%)
DataNodes usages% (Min/Median/Max/stdDev): 0.00% / 0.00% / 0.00% / 0.00%
Live Nodes 0 (Decommissioned: 0, In Maintenance: 0)
Dead Nodes 0 (Decommissioned: 0, In Maintenance: 0)
Decommissioning Nodes 0
Entering Maintenance Nodes 0
Total Datanode Volume Failures 0 (0 B)
Number of Under-Replicated Blocks 0
Number of Blocks Pending Deletion (including replicas) 0
Block Deletion Start Time Tue Jul 19 17:59:15 +0500 2022
Last Checkpoint Time Tue Jul 19 17:58:43 +0500 2022







CONCLUSION

Hadoop is playing a significant role in today's life and can be fitted with any domain as per requirements we can adopt this technology in our organization as per our business requirements. Hadoop has been a very effective solution for companies dealing with data in petabytes. It has solved many problems in the industry related to huge data management and distributed systems.