BIG DATA ANALYTICS

LAB ASSIGNMENT 2

MAHESH PACHARE

FINAL YEAR B.TECH IT

191080054

AIM: To setup Hadoop Multi Node cluster

THEORY:

What is Hadoop ?

Hadoop is an open source framework from Apache and is used to store, process and analyze data which are very huge in volume.
Hadoop is written in Java and is not OLAP (online analytical processing). It is used for batch/offline processing. It is being used by Facebook, Yahoo, Google, Twitter, LinkedIn and many more.
Moreover it can be scaled up just by adding nodes in the cluster.

Modules of Hadoop

1. **HDFS**: Hadoop Distributed File System. Google published its paper GFS and on the basis of that HDFS was developed. It states that the files will be broken into blocks and stored in nodes over the distributed architecture.

2. **Yarn**: Yet another Resource Negotiator is used for job scheduling and managing the cluster.

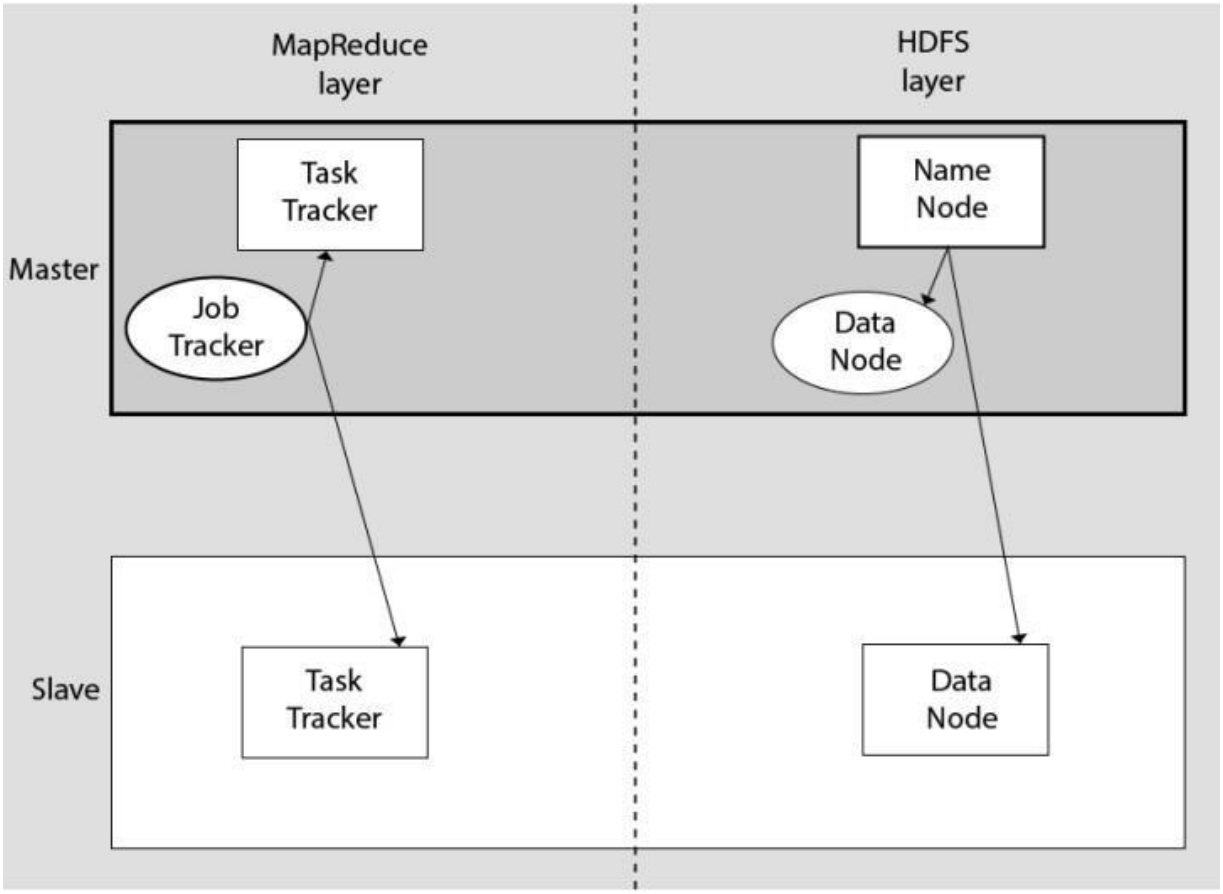
3. **Map Reduce**: This is a framework which helps Java programs to do the parallel computation on data using key value pairs. The Map task takes input data and converts it into a data set which can be computed in Key value pairs. The output of Map task is consumed by reduce task and then the output of reducer gives the desired result.

4. **Hadoop Common**: These Java libraries are used to start Hadoop and are used by other Hadoop modules.

Hadoop Architecture

The Hadoop architecture is a package of the file system, MapReduce engine and the HDFS (Hadoop Distributed File System). The MapReduce engine can be MapReduce/MR1 or YARN/MR2.

A Hadoop cluster consists of a single master and multiple slave nodes. The master node includes Job Tracker, Task Tracker, NameNode, and DataNode whereas the slave node includes DataNode and TaskTracker.



Hadoop Distributed File System

The Hadoop Distributed File System (HDFS) is a distributed file system for Hadoop. It contains a master/slave architecture. This architecture consists of a single NameNode performing the role of master, and multiple DataNodes performing the role of a slave.

Both NameNode and DataNode are capable enough to run on commodity machines. The Java language is used to develop HDFS. So any machine that supports Java language can easily run the NameNode and DataNode software.

NameNode

o It is a single master server that exists in the HDFS cluster.

o As it is a single node, it may become the reason for single point failure.

o It manages the file system namespace by executing an operation like the opening, renaming

and closing the files.

o It simplifies the architecture of the system.

DataNode

o The HDFS cluster contains multiple DataNodes.

o Each DataNode contains multiple data blocks.

o These data blocks are used to store data.

o It is the responsibility of DataNode to read and write requests from the file system's clients.

o It performs block creation, deletion, and replication upon instruction from the NameNode.

Job Tracker

o The role of Job Tracker is to accept the MapReduce jobs from client and process the data by using NameNode.

o In response, NameNode provides metadata to Job Tracker. Task Tracker

o It works as a slave node for Job Tracker.

o It receives tasks and code from Job Tracker and applies that code on the file. This process can also be called a Mapper.

MapReduce Layer

The MapReduce comes into existence when the client application submits the MapReduce job to Job Tracker. In response, the Job Tracker sends the request to the appropriate Task Trackers. Sometimes, the TaskTracker fails or time out. In such a case, that part of the job is rescheduled.

Advantages of Hadoop

o Fast: In HDFS the data is distributed over the cluster and are mapped which helps in faster retrieval. Even the tools to process the data are often on the same servers, thus reducing the processing

time. It is able to process terabytes of data in minutes and Peta bytes in hours.

o **Scalable**: Hadoop cluster can be extended by just adding nodes in the cluster.

o **Cost Effective**: Hadoop is open source and uses commodity hardware to store data so it is really cost effective as compared to traditional relational database management systems.

o **Resilient to failure**: HDFS has the property with which it can replicate data over the network, so if one node is down or some other network failure happens, then Hadoop takes the other copy of data and uses it. Normally, data are replicated thrice but the replication factor is configurable.

History of Hadoop

Hadoop was started by Doug Cutting and Mike Cafarella in 2002. Its origin was the Google File System paper, published by Google.

Let's focus on the history of Hadoop in the following steps: -

o In 2002, Doug Cutting and Mike Cafarella started to work on a project, Apache Nutch. It is an open source web crawler software project.

o While working on Apache Nutch, they were dealing with big data. To store that data they have to spend a lot of money which becomes the consequence of that project. This problem becomes one of the important reasons for the emergence of Hadoop.

o In 2003, Google introduced a file system known as GFS (Google file system). It is a proprietary distributed file system developed to provide efficient access to data.

o In 2004, Google released a white paper on Map Reduce. This technique simplifies the data

processing on large clusters.

o In 2005, Doug Cutting and Mike Cafarella introduced a new file system known as NDFS (Nutch Distributed File System). This file system also includes Map reduce.

o In 2006, Doug Cutting quit Google and joined Yahoo. On the basis of the Nutch project, Dough Cutting introduces a new project Hadoop with a file system known as HDFS (Hadoop Distributed File System). Hadoop first version 0.1.0 released this year.

o Doug Cutting named his project Hadoop after his son's toy elephant.

o In 2007, Yahoo ran two clusters of 1000 machines.

o In 2008, Hadoop became the fastest system to sort 1 terabyte of data on a 900 node cluster within 209 seconds.

o In 2013, Hadoop 2.2 was released.

o In 2017, Hadoop 3.0 was released.

What is a Hadoop Cluster?

In general, a computer cluster is a collection of various computers that work collectively as a single system.

“A hadoop cluster is a collection of independent components connected through a dedicated network to work as a single centralized data processing resource. “

“A hadoop cluster can be referred to as a computational computer cluster for storing and analyzing big data (structured, semi-structured and unstructured) in a distributed environment.”

“A computational computer cluster that distributes data analysis workload across various cluster nodes that work collectively to process the data in parallel.”

Hadoop clusters are also known as “Shared Nothing” systems because nothing is shared between the nodes in a hadoop cluster except for the network which connects them. The shared nothing paradigm of a hadoop cluster reduces the processing latency so when there is a need to process queries on huge amounts of data the cluster-wide latency is completely minimized.

Advantages of a Hadoop Cluster Setup

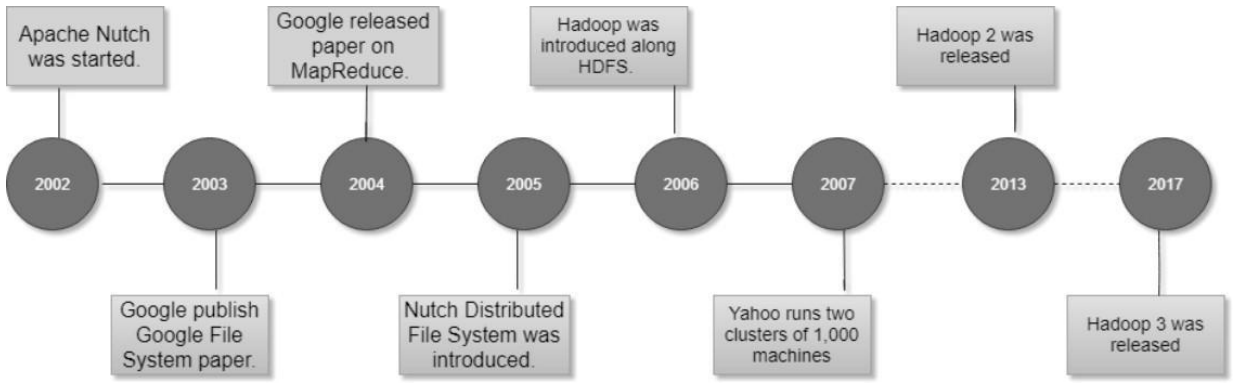
- As big data grows exponentially, parallel processing capabilities of a Hadoop cluster help in increasing the speed of the analysis process. However, the processing power of a hadoop cluster might become inadequate with increasing volume of data. In such scenarios, hadoop clusters can be scaled out easily to keep up with speed of analysis by

adding extra cluster nodes without having to make modifications to the application logic.

- Hadoop cluster setup is inexpensive as they are held down by cheap commodity hardware. Any organization can set up a powerful hadoop cluster without having to spend on expensive server hardware.

- Hadoop clusters are resilient to failure meaning whenever data is sent to a particular node for analysis, it is also replicated to other nodes on the hadoop cluster. If the node fails then the replicated copy of the data present on the other node in the cluster can be used for analysis.

Hadoop Cluster Architecture



A hadoop cluster architecture consists of a data center, rack and the node that actually executes the jobs. Data center consists of the racks and racks consist of nodes. A medium to large cluster consists of a two or three level hadoop cluster architecture that is built with rack mounted servers. Every rack of servers is interconnected through 1 gigabyte of Ethernet (1 GigE). Each rack level switch in a hadoop cluster is connected to a cluster level switch which are in turn

connected to other cluster level switches or they uplink to other switching infrastructure.

Components of a Hadoop Cluster

Hadoop cluster consists of three components -

- **Master Node** – Master node in a hadoop cluster is responsible for storing data in HDFS and executing parallel computation of the stored data using MapReduce. Master Node has 3 nodes – NameNode, Secondary NameNode and JobTracker. JobTracker monitors the parallel processing of data using MapReduce while NameNode handles the data storage function with HDFS. NameNode keeps a track of all the information on files (i.e. the metadata on files) such as the access time of the file, which user is accessing a file on current time and which file is saved in which hadoop cluster. The secondary NameNode keeps a backup of the NameNode data.

- **Slave/Worker Node**- This component in a hadoop cluster is responsible for storing the data and performing computations. Every slave/worker node runs both a TaskTracker and a DataNode service to communicate with the Master node in the cluster. The DataNode service is secondary to the NameNode and the TaskTracker service is secondary to the JobTracker.

- **Client Nodes** – Client node has hadoop installed with all the required cluster configuration settings and is responsible for loading all the data into the hadoop cluster. Client node submits mapreduce jobs describing how data needs to be processed and then the output is retrieved by the client node once the job processing is completed.

Single Node Hadoop Cluster vs. Multi Node Hadoop Cluster

As the name says, a Single Node Hadoop Cluster has only a single machine whereas a Multi-Node Hadoop Cluster will have more than one machine.

In a single node hadoop cluster, all the daemons i.e. DataNode, NameNode, TaskTracker and JobTracker run on the same machine/host. In a single node hadoop cluster setup everything runs on a single JVM instance. The hadoop user need not make any configuration settings except for setting the JAVA_HOME variable. For any single node hadoop cluster setup, default replication factor is 1.

In a multi-node hadoop cluster, all the essential daemons are up and running on different machines/hosts. A multi-node hadoop cluster setup has a master slave architecture where in one machine acts as a master that runs the NameNode daemon while the other machines acts as slave or worker nodes to run other hadoop daemons. Usually in a multi-node hadoop cluster there are cheaper machines (commodity computers) that run the TaskTracker and DataNode daemons while other services are run on powerful servers. For a
multi-node hadoop cluster, machines or computers can be present in any location irrespective of the location of the physical server.

Best Practices for Building a Hadoop Cluster

Hadoop's performance depends on various factors based on the hardware resources which use hard drive (I/O storage), CPU, memory, network bandwidth and other well-configured software layers. Building a Hadoop cluster is a complex task that requires

consideration of several factors like choosing the right hardware, sizing the hadoop cluster and configuring it correctly.

• Choosing the Right Hardware for a Hadoop Cluster

Many organizations are in a predicament when setting up hadoop infrastructure as they are not aware on what kind of machines they need to purchase for setting up an optimized hadoop environment and what is the ideal configuration they must use. The foremost thing that bothers users is deciding on the hardware for the hadoop cluster. Hadoop runs on industry-standard hardware but there is no ideal cluster configuration like providing a list of hardware specifications to setup cluster hadoop. The hardware chosen for a hadoop cluster setup should provide a perfect balance between performance and economy for a particular workload. Choosing the right hardware for a hadoop cluster is a standard chicken-and-egg problem that

requires complete understanding of the workloads (IO bound or CPU bound workloads) to fully optimize it after thorough testing and validation. The number of machines or the hardware specification of machines depends on factors like –

I. Volume of the Data

II. The type of workload that needs to be processed (CPU driven or Use-Case/IO Bound)

III. Data storage methodology (Data container , data compression technique used , if any)

IV. Data retention policy (How long can you afford to keep the data before flushing it out)

• Sizing a Hadoop Cluster

The data volume that the hadoop users will process on the hadoop cluster should be a key consideration when sizing the hadoop cluster. Knowing the data volume to be processed helps decide as to how many nodes or machines would be required to process the data efficiently and how much memory capacity will be required for each machine. The best practice to size a hadoop cluster is sizing it based on the amount of storage required. Whenever a new node is added to the hadoop cluster, more computing resources will be added to the new storage capacity.

• Configuring the Hadoop Cluster

To obtain maximum performance from a Hadoop cluster, it needs to be configured correctly. However, finding the ideal configuration for a hadoop cluster is not easy. Hadoop framework needs to be adapted to the cluster it is running and also to the job. The best way to decide on the ideal configuration for the cluster is to run the hadoop jobs with the default configuration available to get a baseline. After that the job history log files can be analyzed to see if there is any resource
weakness or if the time taken to run the jobs is higher than expected. Repeating the same process can help fine tune the hadoop cluster configuration in such a way that it best fits the business requirements. The number of CPU cores and memory resources that need to be allocated to the daemons also has a great impact on the performance of the cluster. In case of small to medium data context, one CPU core is reserved on each DataNode whereas 2 CPU cores are reserved on

each DataNode for HDFS and MapReduce daemons, in case of huge data context.

Are hadoop clusters a good solution for all data analysis requirements?

Having listed out the benefits of a hadoop cluster setup, it is extremely important to understand if it is ideal to use a hadoop cluster setup for all data analysis needs. For example, if a company has intense data analysis requirements but has relatively less data then under such circumstances the company might not benefit from using Hadoop cluster setup. A hadoop cluster setup is always optimized for large datasets. For instance, 10MB of data when fed to a hadoop cluster for processing will take more time to process when compared to traditional systems.

Hadoop clusters make an assumption that data can be torn apart and analyzed by parallel processes running on different cluster nodes. Thus, a hadoop cluster is the right tool for analysis only in a parallel processing environment. The answer to when you should consider building a hadoop cluster depends on whether or not your organization's data analysis needs can be met by the capabilities of a hadoop cluster setup.

Here are a few scenarios where hadoop cluster setup might not be a right fit –

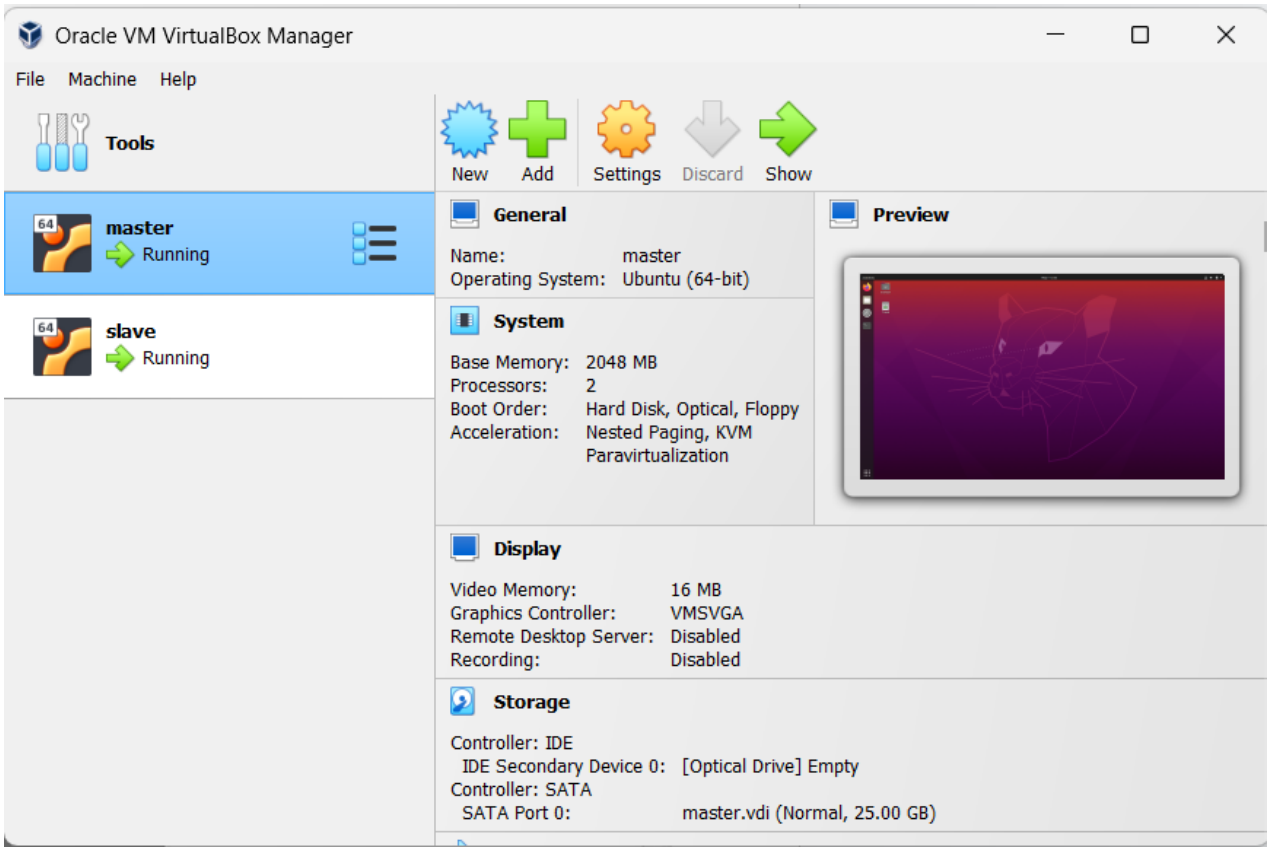
- If the analysis requires processing a large number of small files then it might not be ideal to use a hadoop cluster because the amount of memory that will be required to store the metadata within the namenode will be huge.

• If the task requires multiple write scenarios between files.

• Tasks that require near real-time data access or any other low latency tasks.

PROCEDURE:

2 virtual machines: master and slave are setup using Virtualbox

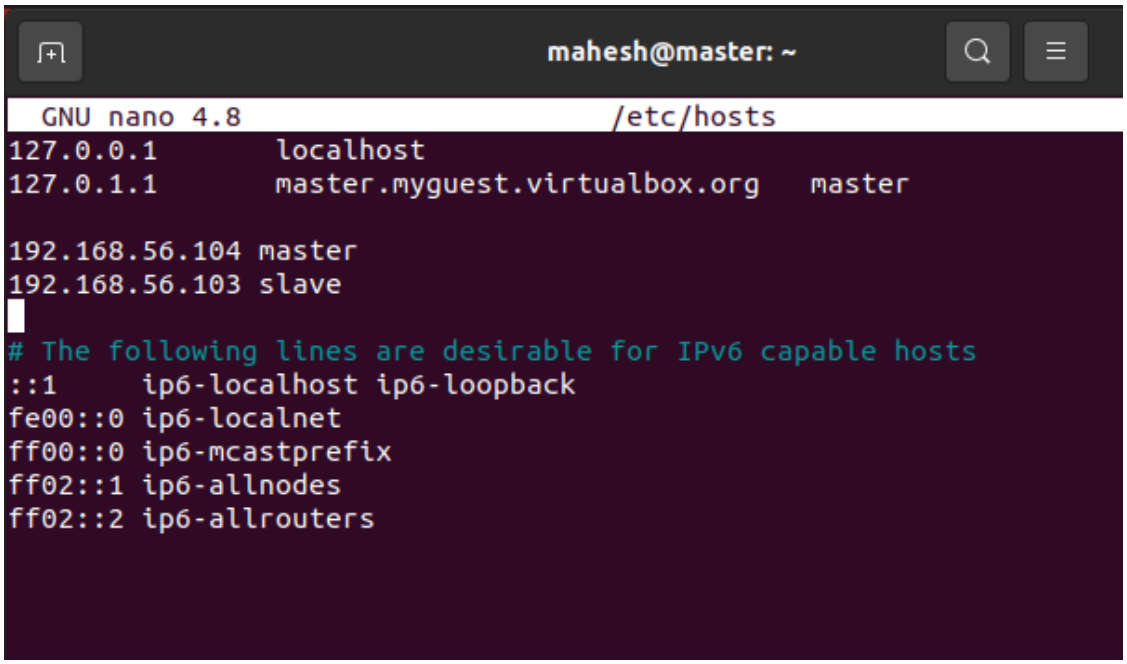


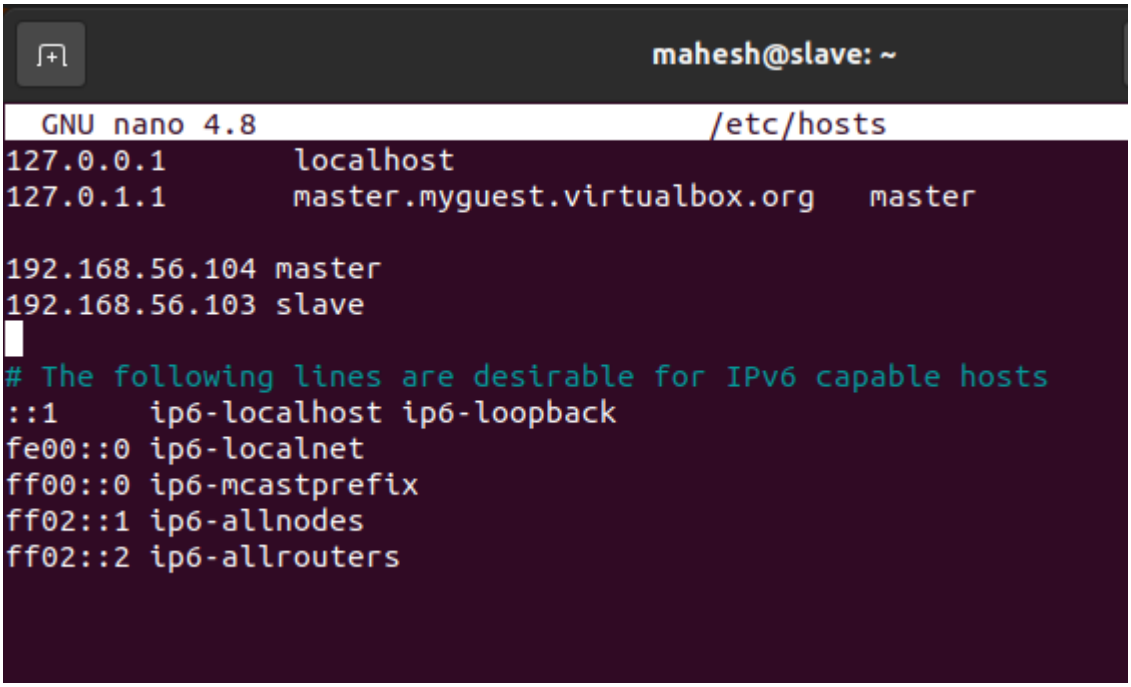
Select Bridger Adapter from Network in both virtual machines. Find IP Address of master and slave using **ifconfig**- (sahilmaster: 192.168.56.103, sahilslave: 192.168.56.104)

```
mahesh@master:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::4eee:1cc6:cbc1:e18f  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:96:9c:b8  txqueuelen 1000  (Ethernet)
        RX packets 12075  bytes 9195097 (9.1 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8492  bytes 2677475 (2.6 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.56.104  netmask 255.255.255.0  broadcast 192.168.56.255
        inet6 fe80::da5:1539:3458:3423  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:07:04:8e  txqueuelen 1000  (Ethernet)
        RX packets 28  bytes 6414 (6.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 75  bytes 9842 (9.8 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 8366  bytes 1555308 (1.5 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8366  bytes 1555308 (1.5 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

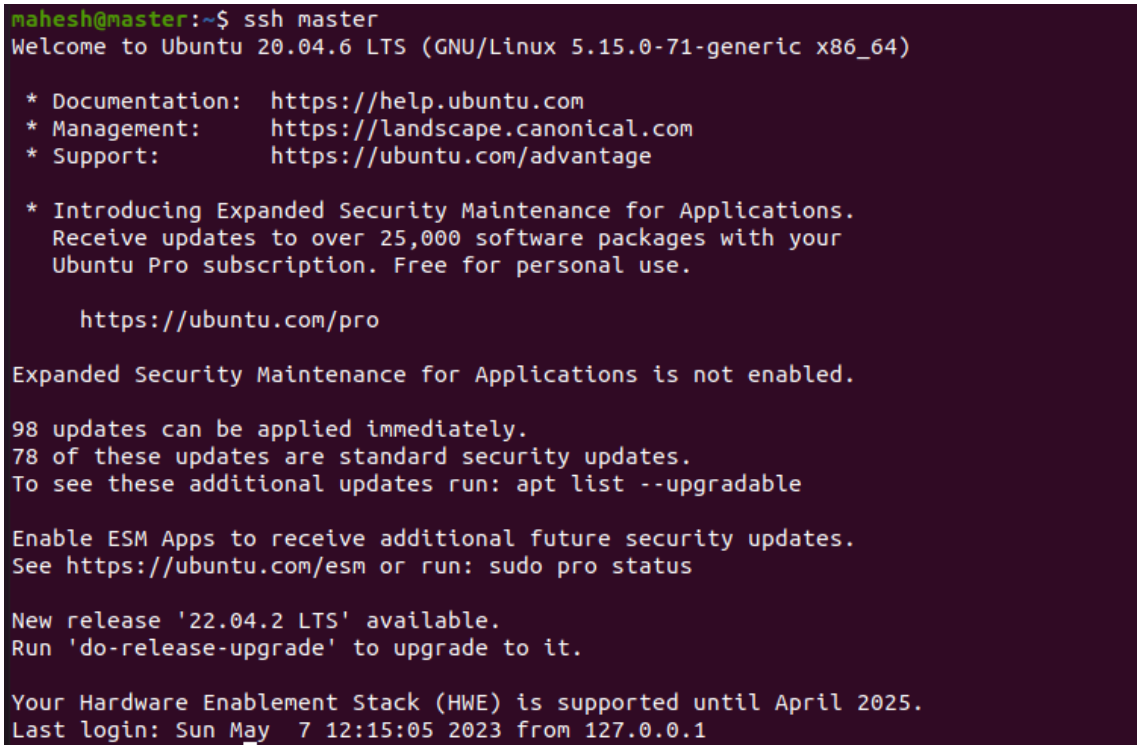
```
mahesh@slave:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::933e:a1e7:5ea9:e4fd  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:de:79:66  txqueuelen 1000  (Ethernet)
        RX packets 430942  bytes 649943936 (649.9 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 63608  bytes 4013247 (4.0 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.56.103  netmask 255.255.255.0  broadcast 192.168.56.255
        inet6 fe80::216b:b03b:d81:6ad7  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:64:c4:fb  txqueuelen 1000  (Ethernet)
        RX packets 68  bytes 9130 (9.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 75  bytes 9136 (9.1 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 1896  bytes 566565 (566.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1896  bytes 566565 (566.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Updating /etc/hosts on master and slave virtual machines





Connecting master to master



Connecting from master to slave

```
mahesh@master:~$ ssh slave
The authenticity of host 'slave (192.168.56.103)' can't be established.
ECDSA key fingerprint is SHA256:FS4sGEQPDRUaz7fQInRcsMX4ZWbIVT42oOE3VeZ4uk4.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'slave,192.168.56.103' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-71-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 * Introducing Expanded Security Maintenance for Applications.
   Receive updates to over 25,000 software packages with your
   Ubuntu Pro subscription. Free for personal use.

     https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

98 updates can be applied immediately.
78 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

New release '22.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Sun May  7 12:15:05 2023 from 127.0.0.1
```

Update workers in master

Update masters file with master instead of localhost

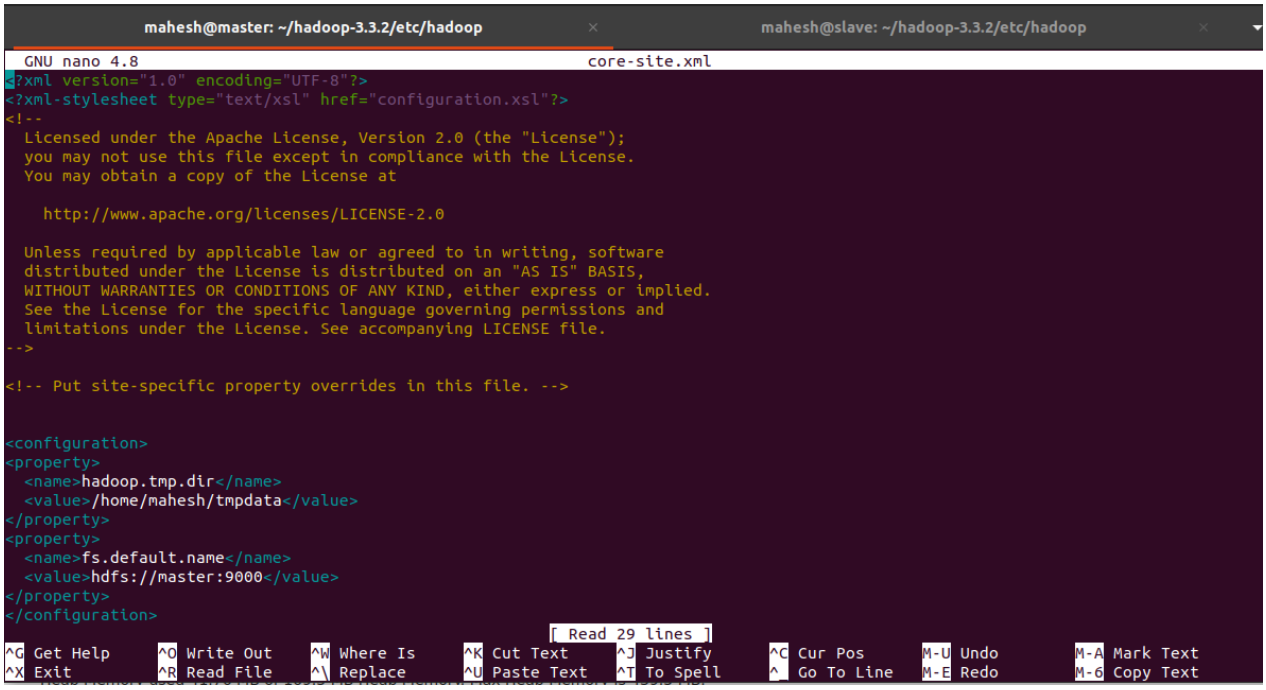
Update slaves file with master slave instead of localhost

```
mahesh@master:~$ cd hadoop-3.3.2/etc/hadoop/
mahesh@master:~/hadoop-3.3.2/etc/hadoop$ sudo nano workers
mahesh@master:~/hadoop-3.3.2/etc/hadoop$ sudo nano core-site.xml
mahesh@master:~/hadoop-3.3.2/etc/hadoop$ sudo nano hdfs-site.xml
mahesh@master:~/hadoop-3.3.2/etc/hadoop$ sudo nano mapred-site.xml
mahesh@master:~/hadoop-3.3.2/etc/hadoop$ sudo nano yarn-site.xml
```

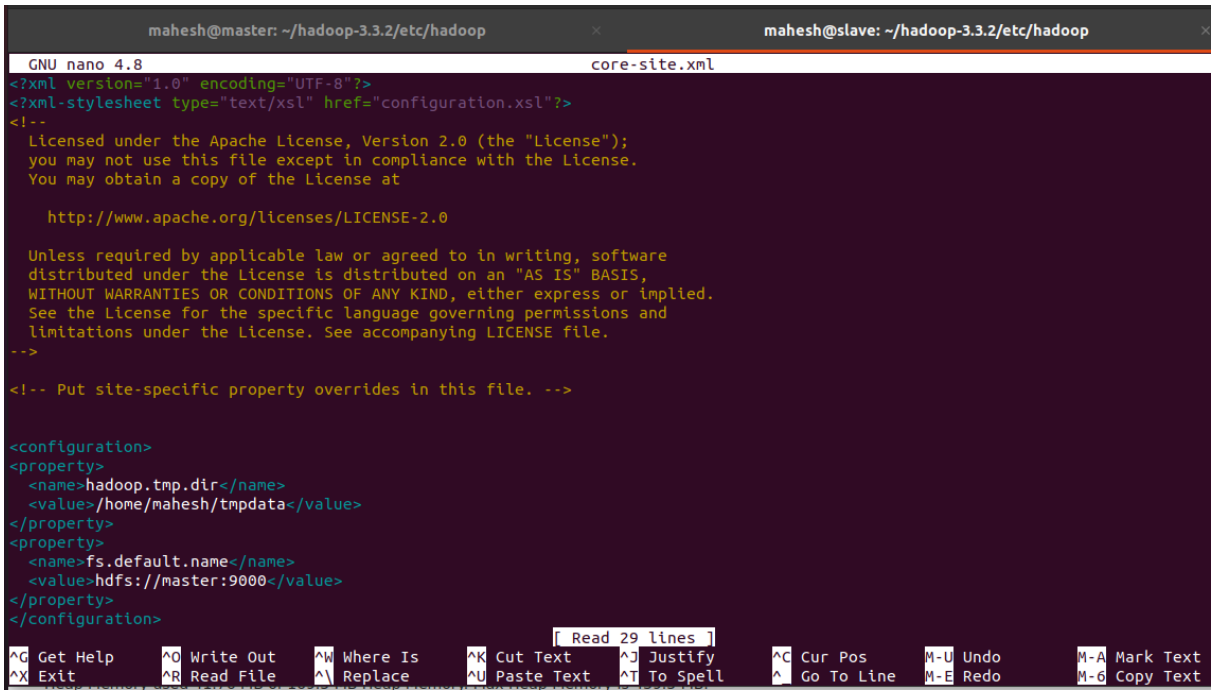
We must change the configuration files core-site.xml, mapred-site.xml and hdfs-site.xml

a) core-site.xml

i. Master

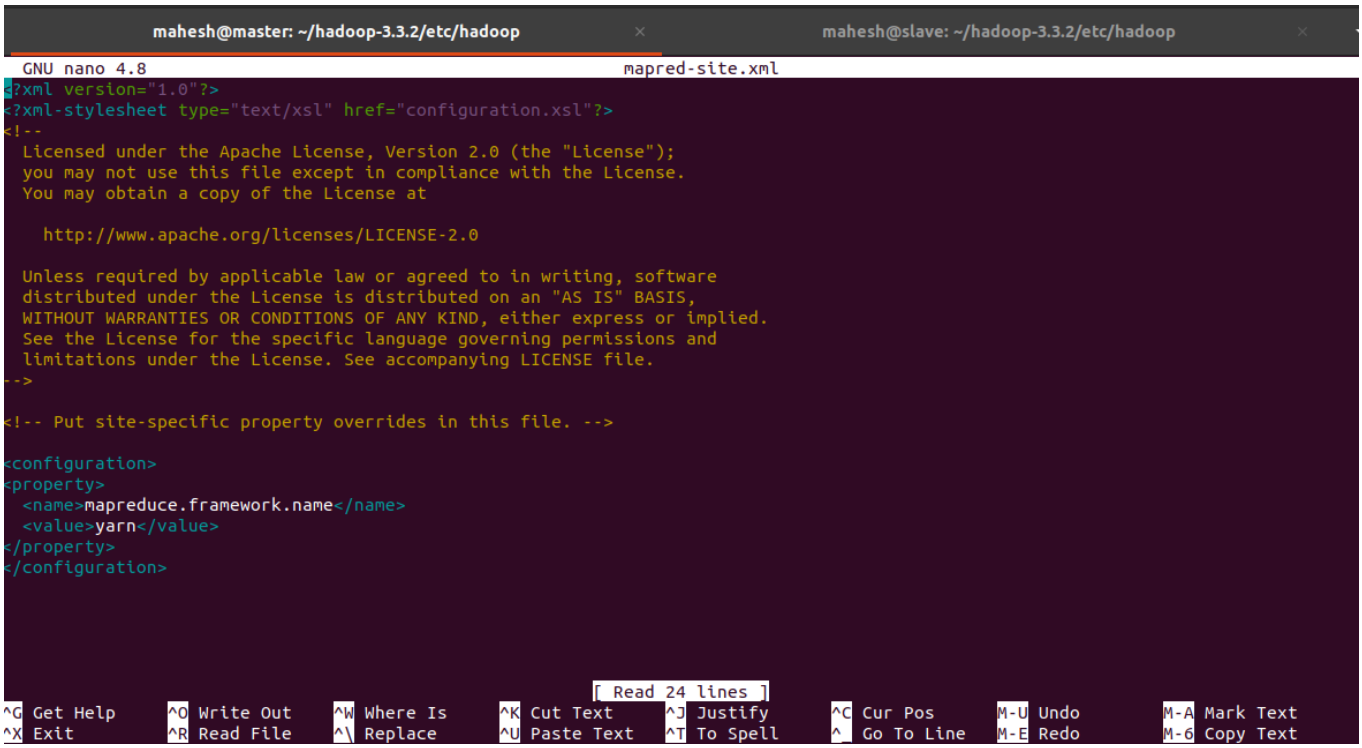


ii. Slave



b) mapred-site.xml

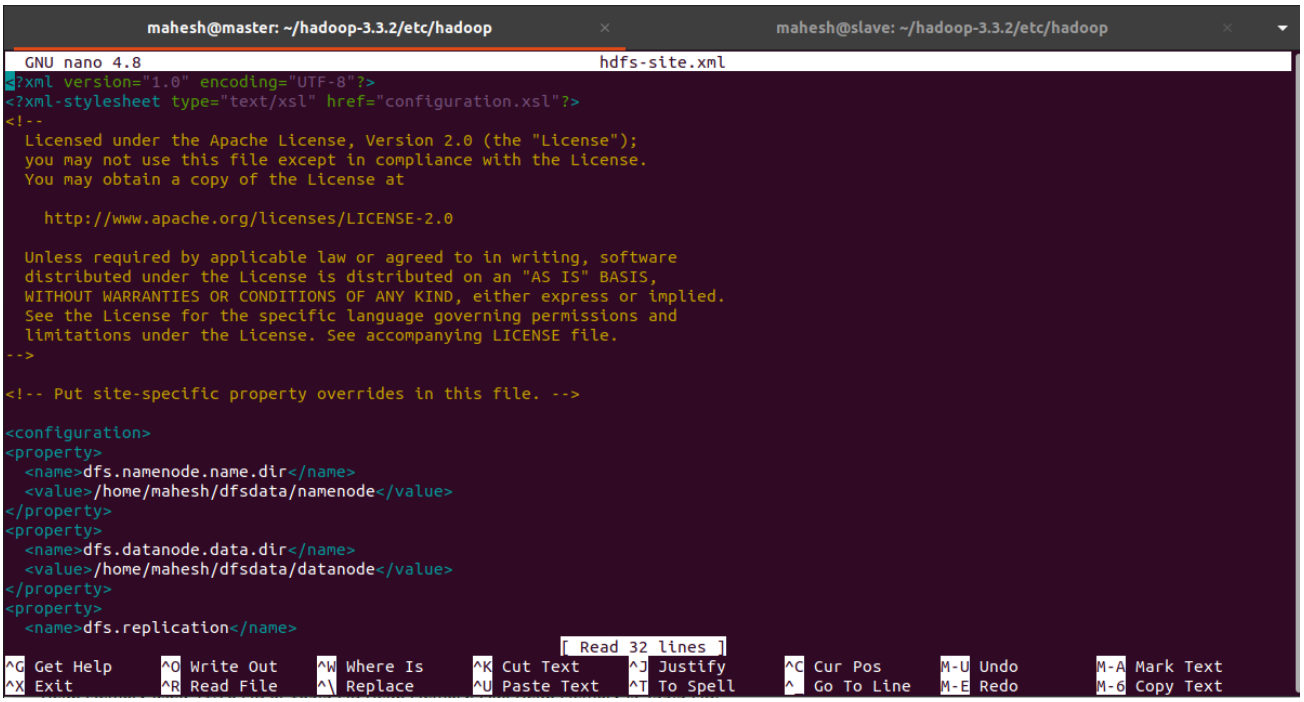
i. Master



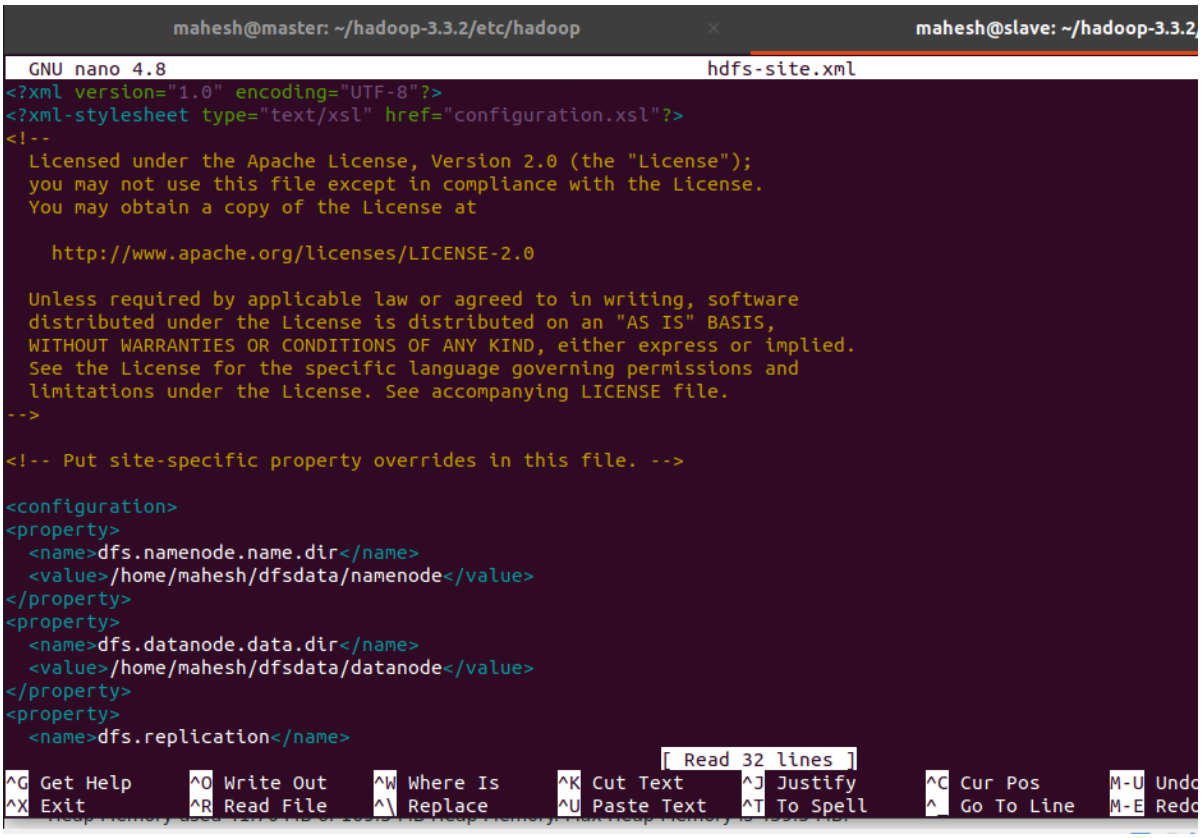
ii. Slave

c) hdfs-site.xml

i. Master



ii. Slave



Formatting hdfs via name node, run bin/start-dfs.sh on master, viewing processes in master

```
mahesh@master:~/hadoop-3.3.2$ bin/hadoop namenode -format
WARNING: Use of this script to execute namenode is deprecated.
WARNING: Attempting to execute replacement "hdfs namenode" instead.

2023-05-07 14:52:50,980 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = master/127.0.1.1
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 3.3.2
STARTUP_MSG:   classpath = /home/mahesh/hadoop-3.3.2/etc/hadoop:/home/mahesh/hadoop-3.3.2/shar
/mahesh/hadoop-3.3.2/share/hadoop/common/lib/slf4j-log4j12-1.7.30.jar:/home/mahesh/hadoop-3.3.
/home/mahesh/hadoop-3.3.2/share/hadoop/common/lib/kerb-server-1.0.1.jar:/home/mahesh/hadoop-3.
2-2.1.1.jar:/home/mahesh/hadoop-3.3.2/share/hadoop/common/lib/checker-qual-2.5.2.jar:/home/mah
```

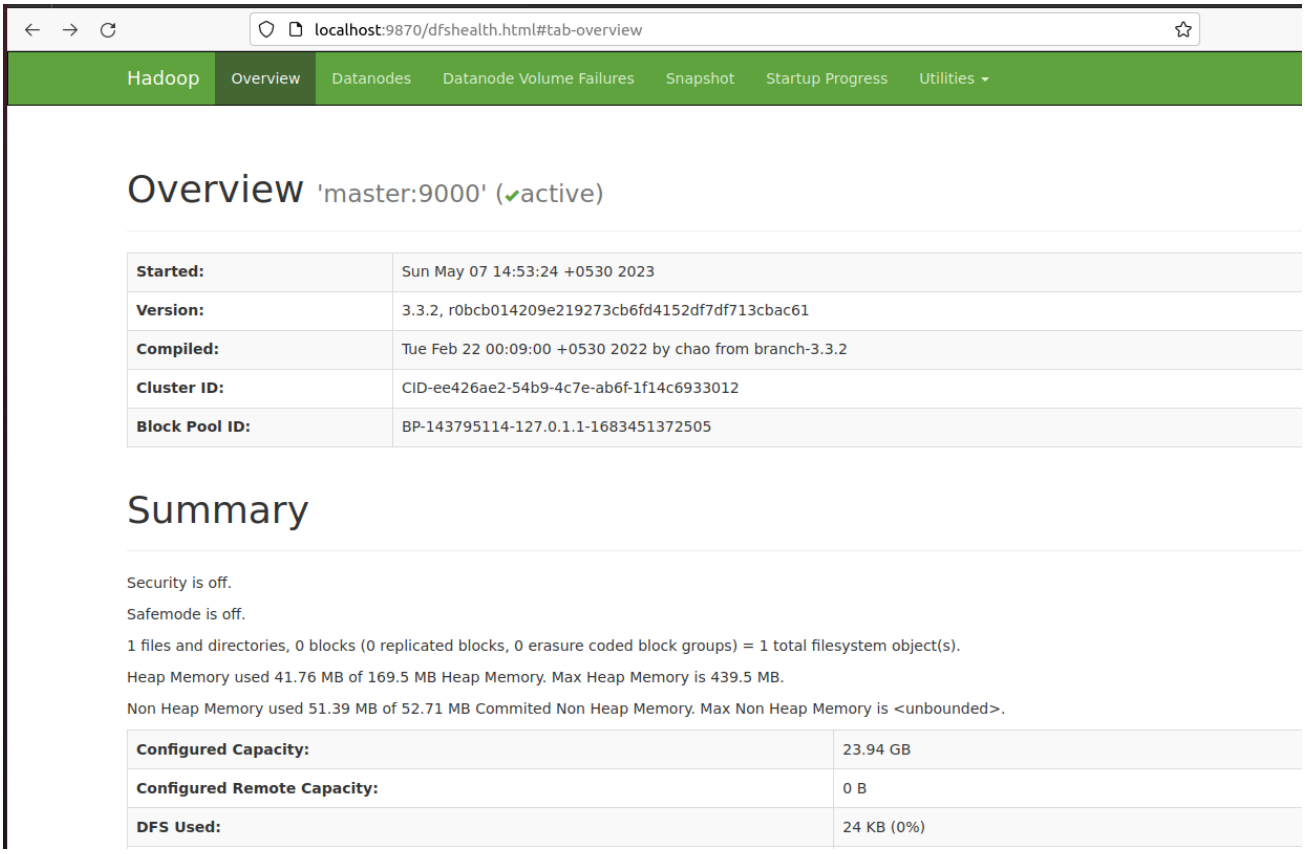
Viewing processes in master

```
mahesh@master:~/hadoop-3.3.2$ cd sbin
mahesh@master:~/hadoop-3.3.2/sbin$ ./start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as mahesh in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [master]
Starting datanodes
Starting secondary namenodes [master]
Starting resourcemanager
Starting nodemanagers
mahesh@master:~/hadoop-3.3.2/sbin$ jps
9954 NameNode
10100 DataNode
10999 Jps
10648 NodeManager
10302 SecondaryNameNode
10510 ResourceManager
```

Viewing processes in master and slave

```
mahesh@slave:~/hadoop-3.3.2/etc/hadoop$ jps
6531 Jps
6419 NodeManager
6284 DataNode
```

Viewing the live nodes on browser:



Stopping processes on master

```
mahesh@master:~/hadoop-3.3.2/sbin$ ./stop-all.sh
WARNING: Stopping all Apache Hadoop daemons as mahesh in 10 seconds.
WARNING: Use CTRL-C to abort.
Stopping namenodes on [master]
Stopping datanodes
Stopping secondary namenodes [master]
Stopping nodemanagers
slave: WARNING: nodemanager did not stop gracefully after 5 seconds: Trying to kill with kill -9
Stopping resourcemanager
```

CONCLUSION:

In this experiment, I learned to set up a Hadoop multi node cluster on Ubuntu using Virtualbox by creating the two virtual machines (Master and Slave). I set up a network using Bridged adapter mode in virtualbox for connecting the two virtual machines. In our case, the master machine performed as master as well as slave node since there were only 2 machines. From the master machine, I started connection from master to master and master to slave using the ssh command. I added the public key of master authorized_keys in slave and updated the conf/master and conf/slave files. A multi-node cluster was started in two steps. In the first step, only the HDFS daemons
were started where DataNode, NameNode and SecondaryNameNode ran on master and only DataNode ran on slave. Success or failure of starting dfs on master can be verified on slave by inspecting the log file. In the second step, the MapReduce daemons were started and which additionally started JobTracker and TaskTracker processes on master node and TaskTracker on slave node. Success or failure of starting dfs on master can be verified on slave by inspecting the log
file. While stopping the multi node cluster, I followed the reverse order where first the MapReduce daemons were stopped and then the HDFS daemons were stopped. At each step, I checked which processes were actively running using the jps command on both master and slave virtual machines.