

Table of Contents

Hadoop Cluster Helper Lab Introduction	3
Lab Pre-requisites	4
Network Settings.....	5
NAT Network Mode Settings (Hadoop Should Not Be Running):	6
Host-Only Network Mode Settings (Enable for the Hadoop Cluster to Function).....	7
Test SSH and Host-Only Mode Network Settings	11
Install PDHS	14
Hadoop Cluster Configuration Files	15
Edit hadoop-env.sh File	16
Example etc/hadoop/hadoop-env.sh	18
Edit core-site.xml File.....	18
Edit hdfs-site.xml File.....	19
etc/hadoop/yarn-site.xml.....	19
etc/hadoop/mapred-site.xml.....	22
Hostname and DNS Settings.....	23
NameNode Format and HDFS Troubleshooting if HDFS Will Not Start	24
Fixing HDFS Failures	24
Cloning Hadoop Nodes	26
Cloning Method	27
Configuring A Hadoop Cluster	30
Steps to Starting and Shutting Down a Hadoop Cluster	30
Configuring DataNodes	30
Install OpenSSH on Ubuntu.....	34
Create Username Using Your FirstName and LastName on EACH server	34
Enable Passwordless SSH for the Hadoop User on EACH server	34
Copy the SSH key to each server.....	35
Add the DataNodes as Workers	36
Starting a Hadoop Cluster for the First Time.....	38

Hadoop Cluster Helper Lab Introduction

Fewer tutorials exist for Hadoop clusters on VirtualBox then other installations we have worked with. Regardless, you may find a few that help with 90% or more of what we are going to cover. This tutorial should get you 100% there but there are a few steps you may need to troubleshoot. For example, Hadoop clusters are picky like most database clusters about checkpoints on clustered nodes. **If different clustered VMs have different data in their datanode, namenode, or tempdata directories, you will find that the cluster may not function correctly.** A recommendation I have is to write your own cleanup scripts for these directories as you identify how to synchronize Hadoop secondary nodes with the primary node. Another issue is disk size. **I would ensure you allocate 34-38 GBs per VM** or YARN may mark your datanode as unhealthy and when this happens it may not run MapReduce jobs.

Another common issue concerns networking and SSH. Clustered nodes must be able to communicate properly. Thus, you need to make sure SSH is working flawlessly. Proper hostname entries must exist in /etc/hosts. Additionally, the primary Hadoop VM needs to be listening on its IP address on port 9000 or whatever port you have configured for other datanodes to connect. You may find port 9000 listening on the loopback address instead, for example (e.g. localhost or 127.0.0.1).

netstat -lnpr

If port 9000 is listening on your loopback address, the cluster will not function properly. Thus, it is essential to be able to troubleshoot networking problems between distributed systems using log files and tools like netstat.

This tutorial references and/or has similar configurations as several of the following sources and thus these are helpful guides:

<https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/ClusterSetup.html>

<https://phoenixnap.com/kb/install-hadoop-ubuntu>

<https://linuxconfig.org/ubuntu-20-04-hadoop>

Class Textbook: <https://github.com/PacktPublishing/Big-Data-Architects-Handbook>

Recommended textbook: <https://github.com/PacktPublishing/Big-Data-Analytics-with-Hadoop-3>

Recommended textbook: <https://github.com/tomwhite/hadoop-book>

Recommended textbook: <https://github.com/PacktPublishing/Mastering-Hadoop-3>

If you are running Raspberry: <https://github.com/ptaranti/RaspberryPiCluster>

Lab Pre-requisites

In order to perform this tutorial, you need sufficient space to have three (3) Ubuntu Hadoop machines. This may require an external USB drive and/or external hard drive. If you are running Linux with a GUI, I would allocate 34 GBs or more depending on what else you plan to install.

This tutorial assumes you already have your Ubuntu single-node Hadoop cluster operating including Java/Java JDK, SSH, HDFS, YARN, name nodes, data nodes, and MapReduce.

If your single node Hadoop instance and/or MapReduce are not functioning properly, it is critical to finish this lab work first before beginning this lab.

We will start with a working single instance Hadoop Virtual Machine (VM).

***Please follow the previous single node Hadoop tutorial to achieve this prior to beginning this tutorial.**

Network Settings

In order for Hadoop clusters to function, all of the namenodes and datanodes need to be able to communicate. In our lab environment, we are going to accomplish this by creating a sandbox local area network (LAN). This LAN will allow the cluster to function properly from a networking standpoint.

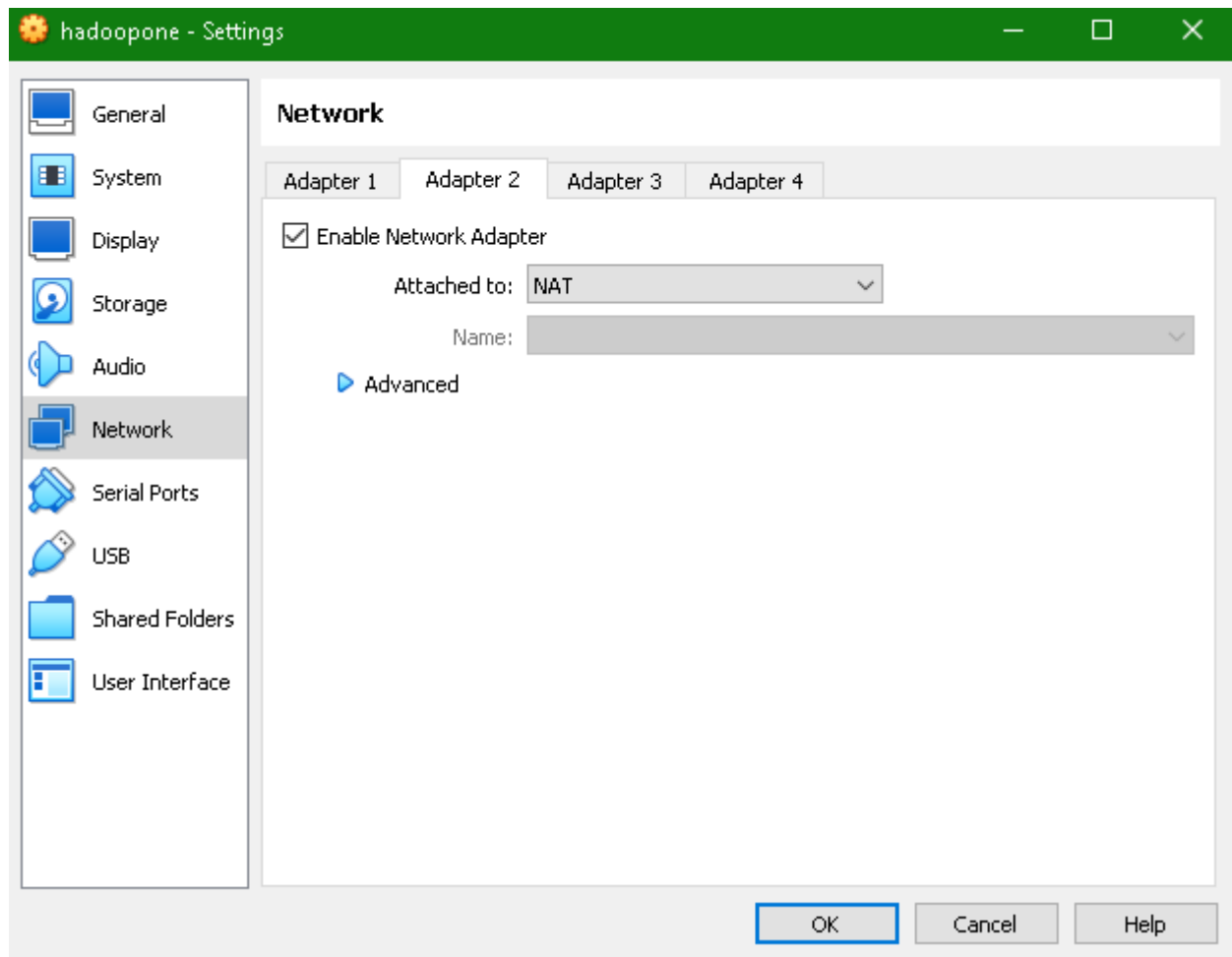
A problem the VirtualBox environments can have is that the network interface cards (NICs) have to use part of the physical NIC of the hardware they run on. This can create networking conflicts.

To avoid networking conflicts, there are two different VirtualBox modes we are going to use. 1) NAT mode, and 2) Host-Only mode.

NAT mode will be used for installing new software only. Or, if you need to use the Internet to login to our LMS for example. NAT is only for Internet use. We never want to startup our Hadoop cluster with NAT because it could result in corrupting the Hadoop distributed file system (HDFS), which requires proper network connectivity between the datanodes to function properly.

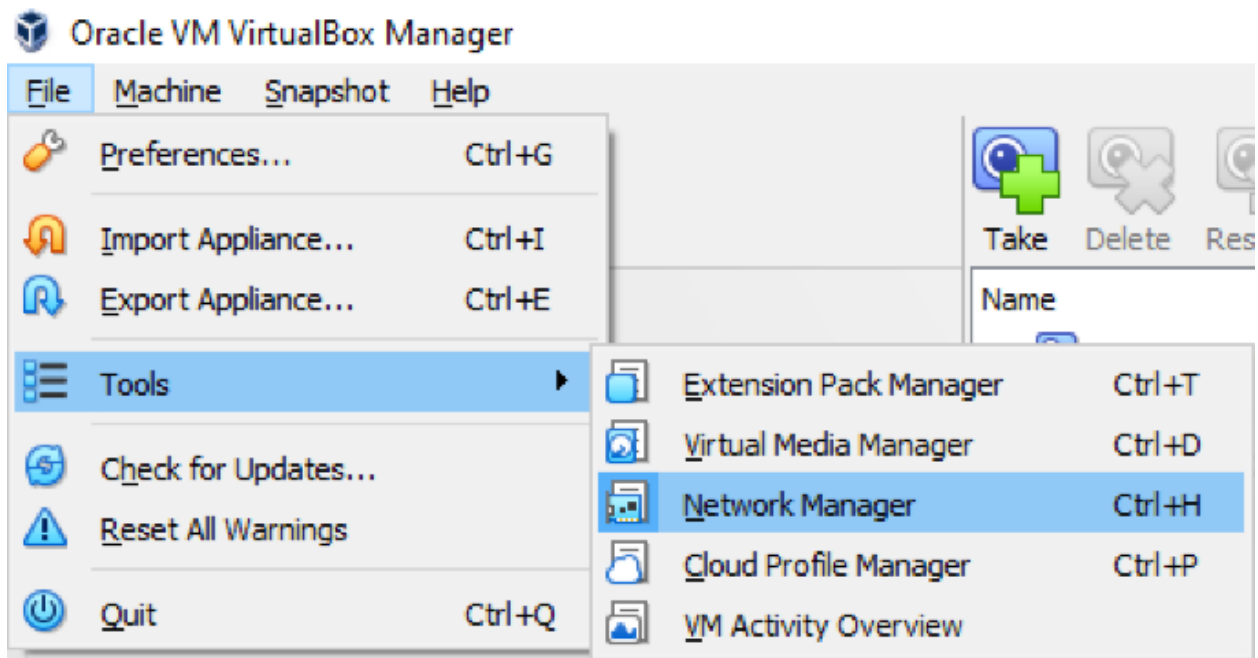
Only use NAT mode when the Hadoop cluster is not in operation!

NAT Network Mode Settings (Hadoop Should Not Be Running):



Host-Only Network Mode Settings (Enable for the Hadoop Cluster to Function)

To create new NIC cards go to your Network Manager in VirtualBox:



Next create a Host-Only network using the IP address subnet of 10.100.100.0/24. Ensure the Adapter is set to Configure Adapter Automatically. It should NOT be manually set. In the DHCP settings, ensure the server is set to 10.100.100.1, the server mask 255.255.255.0, the lower bound 10.100.100.10 and the upper bound 10.100.100.254.

 Create

 Remove

 Properties

Host-only Networks

NAT Networks

Cloud Networks

Name	IPv4 Prefix	IPv6 Prefix	DHCP Server
VirtualBox Host-Only Ethernet Adapter	169.254.222.24/16		Enabled

Adapter

DHCP Server

- ☒ Configure Adapter Automatically
- ☐ Configure Adapter Manually

IPv4 Address:

169.254.222.24

IPv4 Network Mask:

255.255.0.0

IPv6 Address:

fe80::21a1:22ed:381a:354f

IPv6 Prefix Length:

64

Apply

Reset



Create



Remove



Properties

Host-only Networks

NAT Networks

Cloud Networks

Name	IPv4 Prefix	IPv6 Prefix	DHCP Server
VirtualBox Host-Only Ethernet Adapter	169.254.222.24/16		Enabled

Adapter

DHCP Server

☒ Enable Server

Server Address: 10.100.100.1

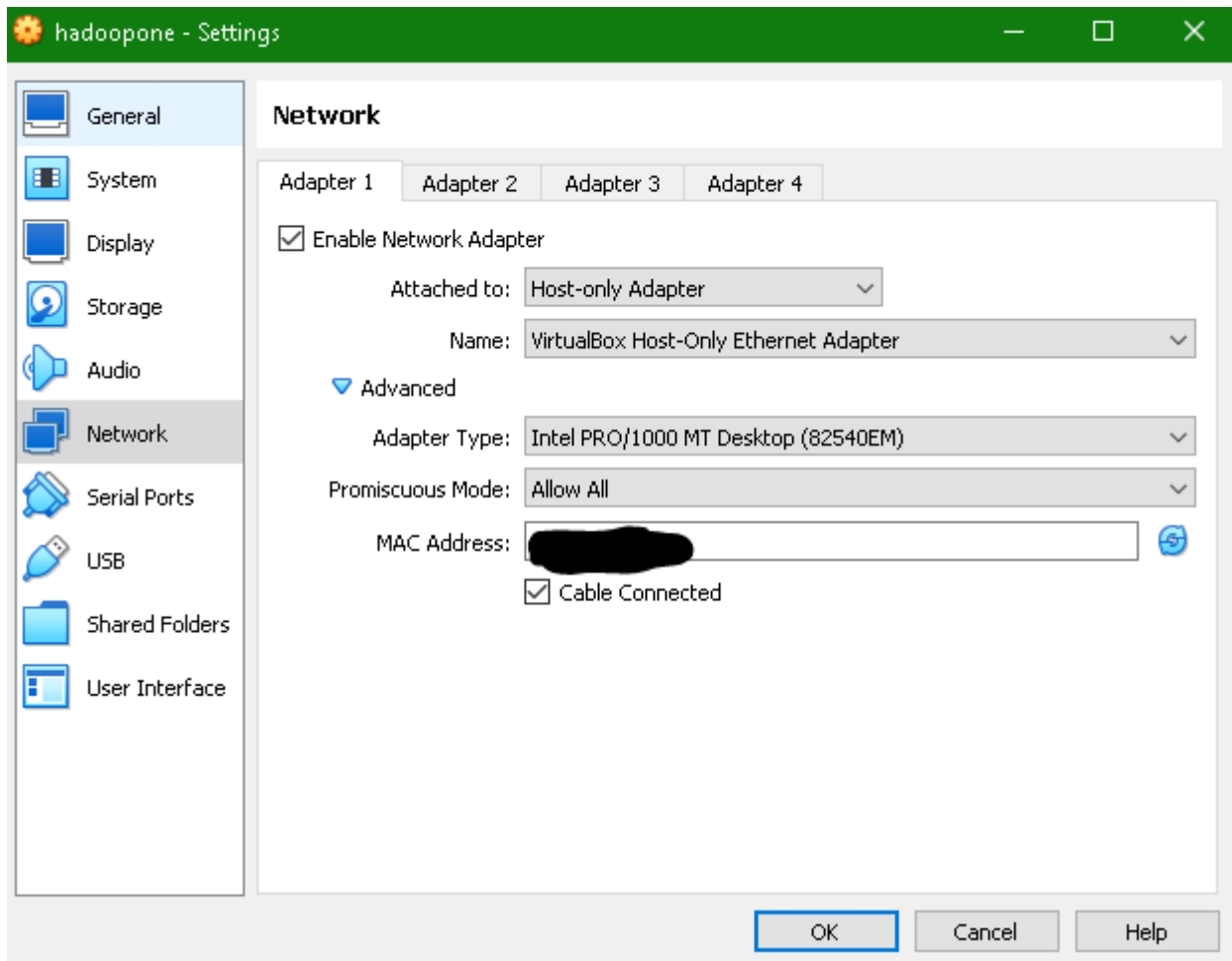
Server Mask: 255.255.255.0

Lower Address Bound: 10.100.100.10

Upper Address Bound: 10.100.100.254

Apply

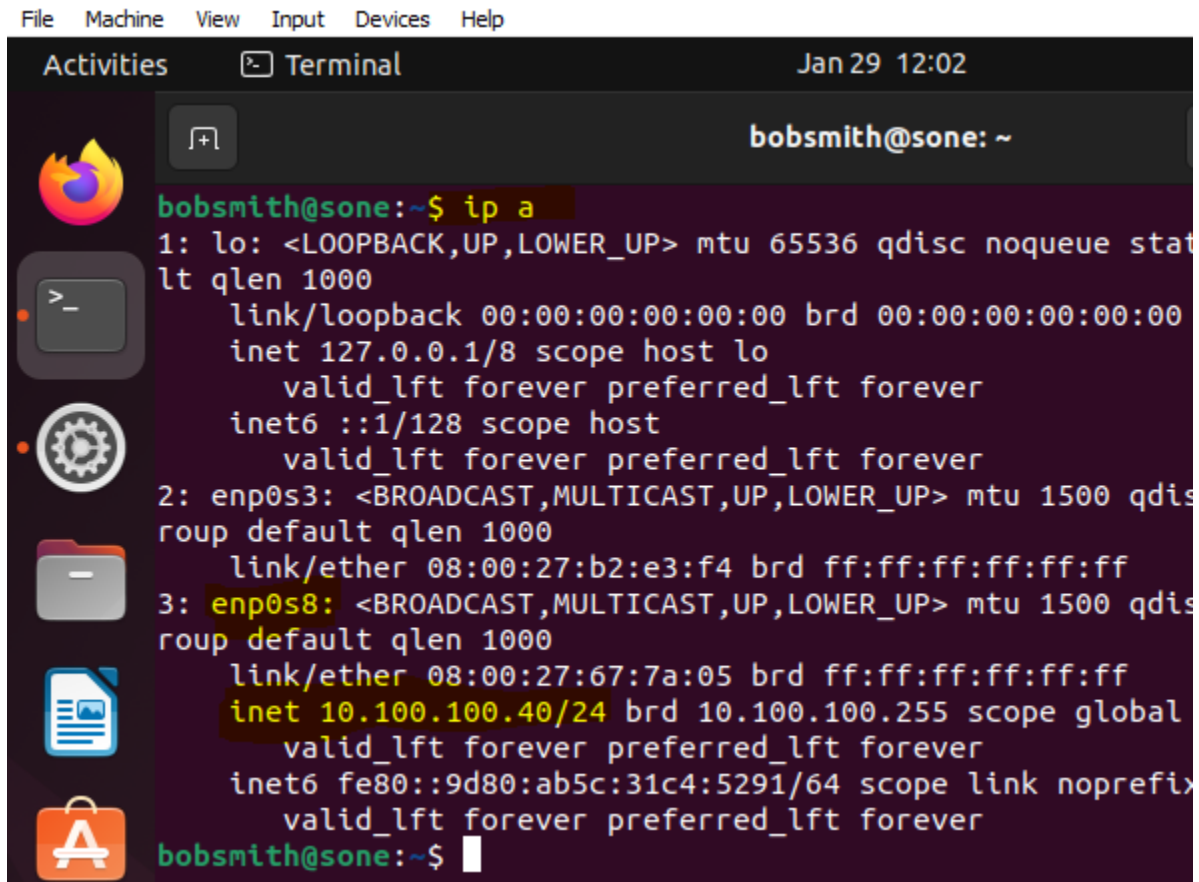
Reset



Test SSH and Host-Only Mode Network Settings

Start your VM in Host-Only mode.

Check your IP address to ensure it is between your lower and upper address boundaries. The first address should be 10.100.100.10.

A screenshot of a Linux terminal window. The title bar shows 'File Machine View Input Devices Help' and 'Activities Terminal Jan 29 12:02'. The terminal prompt is 'bobsmith@sone: ~'. The user has entered the command 'ip a'. The output shows three network interfaces: 'lo' (loopback), 'enp0s3' (ethernet), and 'enp0s8' (ethernet). The 'enp0s8' interface is highlighted with a yellow box, showing its IP address as '10.100.100.40/24'.

```
bobsmith@sone:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UP
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc default state UP
    link/ether 08:00:27:b2:e3:f4 brd ff:ff:ff:ff:ff:ff
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc default state UP
    link/ether 08:00:27:67:7a:05 brd ff:ff:ff:ff:ff:ff
    inet 10.100.100.40/24 brd 10.100.100.255 scope global dynamic enp0s8
        valid_lft forever preferred_lft forever
    inet6 fe80::9d80:ab5c:31c4:5291/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
bobsmith@sone:~$
```

If necessary, disable your firewall:

```
sudo ufw disable
```

Assuming your firewall is not blocking traffic and promiscuous mode was set to allow all in your host-only mode advanced settings (screenshot above), you should be able to use SSH to remotely administer the VM.

You can download and install the Putty SSH client to connect to your VM remotely here:

<https://www.putty.org/>

Simply put the IP address of your VM in the Putty Hostname to connect. Then, enter your username and password.

In order to use SSH, it does need to be installed and running on your VM server:

```
bobsmith@sone:~$ sudo systemctl status ssh
[sudo] password for bobsmith:
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; vendor preset: enabled)
   Active: active (running) since Sun 2023-08-27 12:34:12 UTC; 1min 45s ago
     Docs: man:sshd(8)
           man:sshd_config(5)
  Process: 750 ExecStartPre=/usr/sbin/sshd -f /etc/ssh/sshd_config (code=exited, status=0/SUCCESS)
 Main PID: 771 (sshd)
    Tasks: 1 (limit: 3444)
   Memory: 3.8M
```

If not, install it.

In addition, ensure the sshd_config settings allow for you to connect:

```
bobsmith@sone:~$ cd /etc/ssh/
bobsmith@sone:/etc/ssh$ ls
moduli          sshd_config.d      ssh_config
ssh_config      ssh_host_ecdsa_key ssh_host_ecdsa_key.pub
ssh_config.d    ssh_host_ecdsa_key.pub
sshd_config     ssh_host_ed25519_key
bobsmith@sone:/etc/ssh$ nano sshd_config
```

```
GNU nano 6.2
PubkeyAuthentication yes

# Expect .ssh/authorized_keys2 to be readable by root or the user
#AuthorizedKeysFile .ssh/authorized_keys2

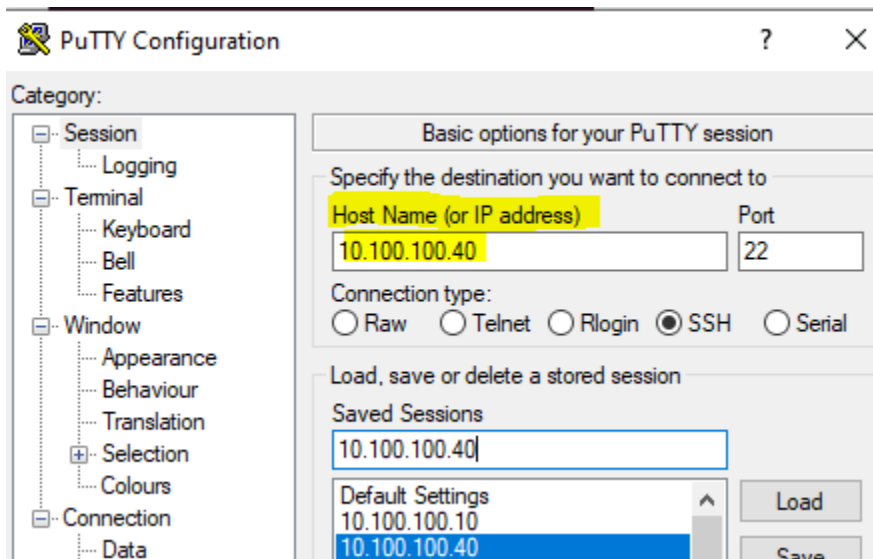
#AuthorizedPrincipalsFile none

#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody

# For this to work you will also need:
#HostbasedAuthentication no
# Change to yes if you don't trust host keys
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here:
PasswordAuthentication yes
#PermitEmptyPasswords no
```

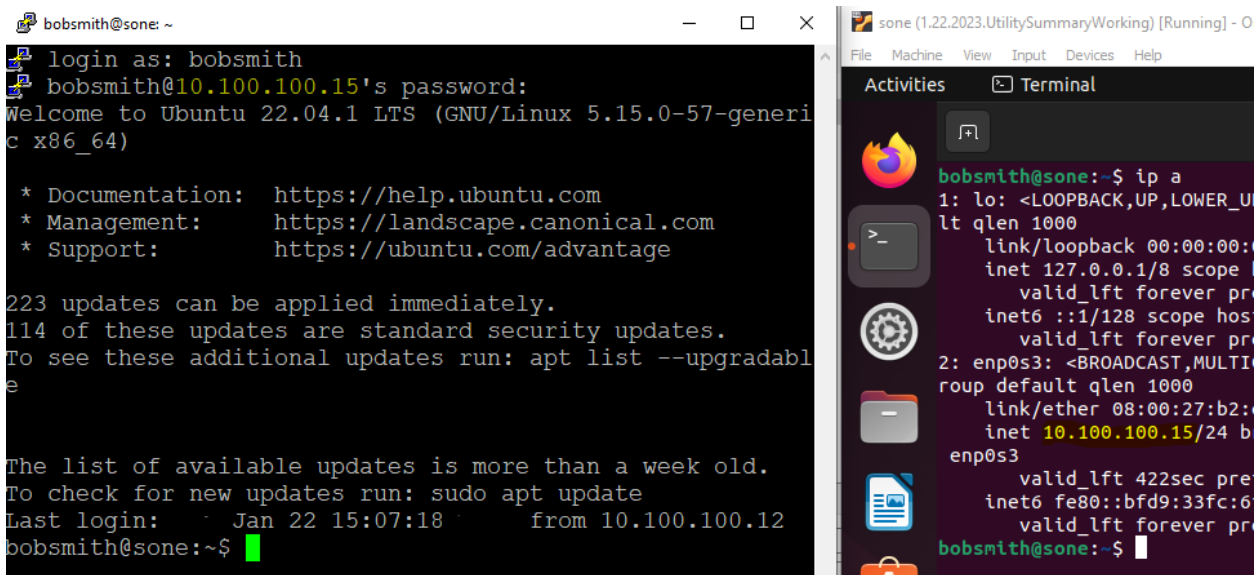
Next, open a Putty SSH session to your VM using your host-only mode IP address:



If you cannot connect, check your network settings and ensure they are correct per previous instructions and/or check your firewall and ensure that it is disabled:

```
bobsmith@sone:/etc/ssh$ sudo ufw disable
Firewall stopped and disabled on system startup
bobsmith@sone:/etc/ssh$ sudo systemctl stop ufw
bobsmith@sone:/etc/ssh$
```

If everything is working, you should be able to connect and login to your VM using Putty:



If this is not working, generate a new SSH key from our previous tutorial.

Install PDHS

The pdsh tool allows a parallel shell across multiple clustered nodes/servers. Install psdh with the following command:

```
sudo apt-get install pdsh
```

Select “Y” to install it

Make sure you are the Hadoop user or your firstname, lastname username that runs Hadoop from prior tutorials. For example:

```
su – amysmith
```

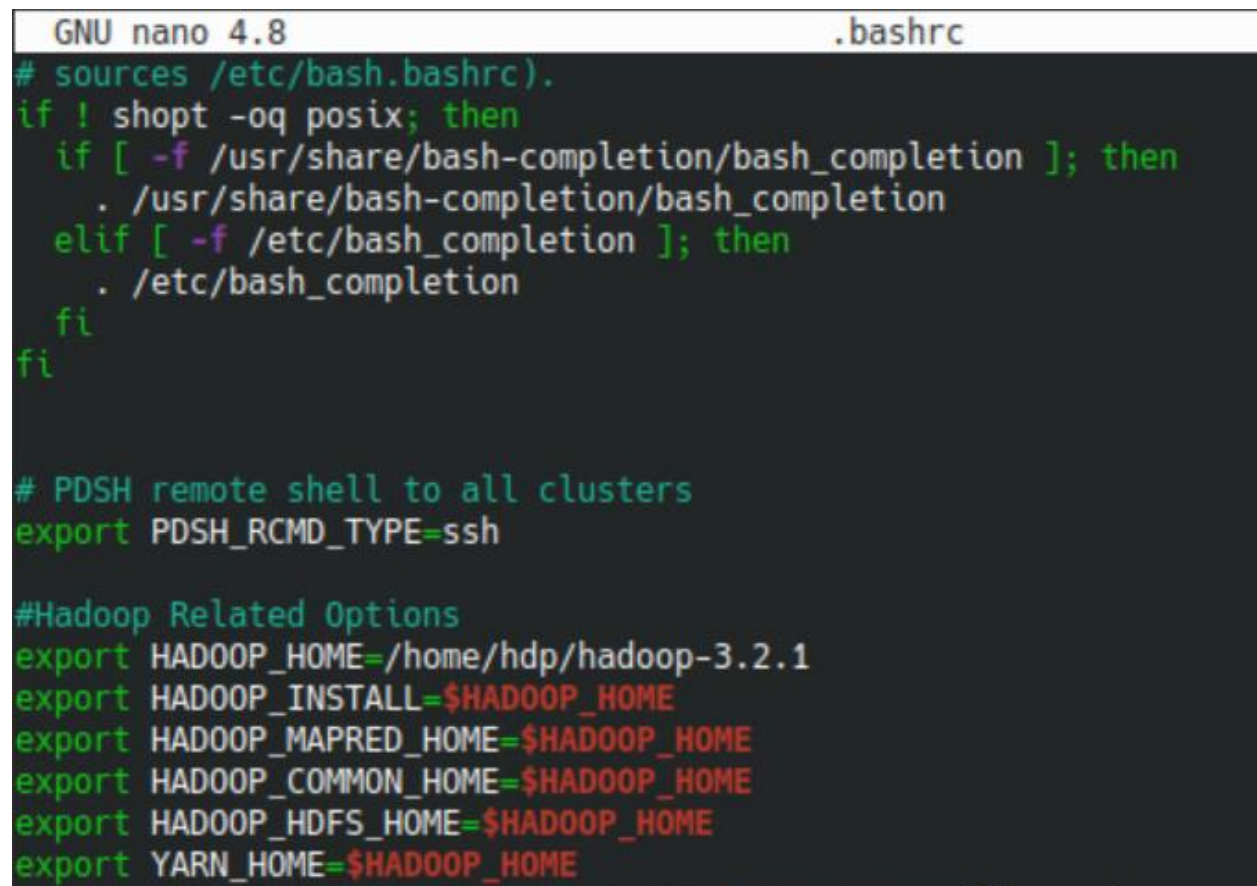
Set your pdsh environment to ssh

Open the Bashrc file with nano:

```
sudo nano .bashrc
```

Add a PDSH RCMD export to the end of the file:

```
export PDSH_RCMD_TYPE=ssh
```



```
GNU nano 4.8 .bashrc
# 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

# PDSH remote shell to all clusters
export PDSH_RCMD_TYPE=ssh

#Hadoop Related Options
export HADOOP_HOME=/home/hdp/hadoop-3.2.1
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
```

Hadoop Cluster Configuration Files

The optimal documentation to reference for Hadoop and all environments in this course is the official documentation. Hadoop documentation should be referenced here:

<https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/ClusterSetup.html>

If this link is no longer available, search your version of Hadoop on the official Apache documentation site for a cluster setup. Next, navigate to your Hadoop etc configuration directory to configure your cluster:

```
bobsmith@sone: ~/hadoop/etc/hadoop
bobsmith@sone:~/hadoop/etc/hadoop$ pwd
/home/bobsmith/hadoop/etc/hadoop
bobsmith@sone:~/hadoop/etc/hadoop$ ls
capacity-scheduler.xml          kms-log4j.properties
configuration.xml               kms-site.xml
container-executor.cfg          log4j.properties
core-site.xml                   mapred-env.cmd
hadoop-env.cmd                  mapred-env.sh
hadoop-env.sh                   mapred-queues.xml.template
hadoop-metrics2.properties      mapred-site.xml
hadoop-policy.xml               shellprofile.d
hadoop-user-functions.sh.example ssl-client.xml.example
hdfs-rbf-site.xml               ssl-server.xml.example
hdfs-site.xml                   user_ec_policies.xml.template
httpfs-env.sh                   workers
httpfs-log4j.properties         yarn-env.cmd
httpfs-site.xml                 yarn-env.sh
kms-acls.xml                    yarnservice-log4j.properties
kms-env.sh                      yarn-site.xml
bobsmith@sone:~/hadoop/etc/hadoop$
```

Follow the official Hadoop documentation to configure your cluster. The following are examples only and will need customization based upon your specific OS and versions.

As an initial example, the Hadoop official documentation states the `Hadoop-env.sh` needs to minimally include the `JAVA_HOME` path. Thus, determine your specific Java Home directory and set your path variable.

Edit `hadoop-env.sh` File

The `hadoop-env.sh` file serves as a master file to configure YARN, [HDFS](#), [MapReduce](#), and Hadoop-related project settings.

When setting up a **single node Hadoop cluster**, you need to define which Java implementation is to be utilized. Use the previously created `$HADOOP_HOME` variable to access the `hadoop-env.sh` file:

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

Uncomment the `$JAVA_HOME` variable (i.e., remove the `#` sign) and add the full path to the OpenJDK installation on your system. If you have installed the same version as presented in the first part of this tutorial, add the following line:

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

The path needs to match the location of the Java installation on your system.


```
GNU nano 2.9.3 /home/hadoop/hadoop-3.2.1/etc/hadoop/hadoop-env.sh

###

# Technically, the only required environment variable is JAVA_HOME.
# All others are optional. However, the defaults are probably not
# preferred. Many sites configure these options outside of Hadoop,
# such as in /etc/profile.d

# The java implementation to use. By default, this environment
# variable is REQUIRED on ALL platforms except OS X!
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

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

# Location of Hadoop's configuration information. i.e., where this
# file is living. If this is not defined, Hadoop will attempt to
# locate it based upon its execution path.
#

# NOTE: It is recommend that this variable not be set here but in
# /etc/profile.d or equivalent. Some options (such as

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

If you need help to locate the correct Java path, run the following command in your terminal window:

```
which javac
```

The resulting output provides the path to the Java binary directory.

```
hadoop@pnap-VirtualBox:~$ which javac
/usr/bin/javac
```

Use the provided path to find the OpenJDK directory with the following command:

```
readlink -f /usr/bin/javac
```

The section of the path just before the `/bin/javac` directory needs to be assigned to the `$JAVA_HOME` variable.

```
hadoop@pnap-VirtualBox:~$ readlink -f /usr/bin/javac
/usr/lib/jvm/java-8-openjdk-amd64/bin/javac
```

Example etc/hadoop/hadoop-env.sh

```
GNU nano 6.2 hadoop-env.sh *
# Registry DNS specific parameters
###
# For privileged registry DNS, user to run as after
# This will replace the hadoop.id.str Java property
# export HADOOP_REGISTRYDNS_SECURE_USER=yarn

# Supplemental options for privileged registry DNS
# By default, Hadoop uses jsvc which needs to know t
# server jvm.
# export HADOOP_REGISTRYDNS_SECURE_EXTRA_OPTS="-jvm

# Set the Hadoop and Java paths
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HADOOP_CLASSPATH+=" $HADOOP_HOME/lib/*.jar"
export HADOOP_CLASSPATH=${JAVA_HOME}/lib/tools.jar
export PATH=${JAVA_HOME}/bin:${PATH}
```

Edit core-site.xml File

The *core-site.xml* file defines HDFS and Hadoop core properties.

To set up Hadoop in a pseudo-distributed mode, you need to **specify the URL** for your NameNode, and the temporary directory Hadoop uses for the map and reduce process.

Open the *core-site.xml* file in a text editor:

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

Add the following configuration to override the default values for the temporary directory and add your HDFS URL to replace the default local file system setting:

```
<configuration>
<property>
  <name>hadoop.tmp.dir</name>
  <value>/home/hadoop/tmpdata</value>
</property>
<property>
  <name>fs.default.name</name>
  <value>hdfs://michaelhartnamenode:9000</value>
</property>
</configuration>
```

Edit hdfs-site.xml File

The properties in the *hdfs-site.xml* file govern the location for storing node metadata, fsimage file, and edit log file. Configure the file by defining the **NameNode** and **DataNode storage directories**.

Additionally, the default `dfs.replication` value of 3 needs to be changed to 2 to match a two node cluster setup.

Use the following command to open the *hdfs-site.xml* file for editing:

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

Always replace Amy Smith or Bob Smith with your actual username. Add the following configuration to the file and, if needed, adjust the NameNode and DataNode directories to your custom locations:

```
<configuration>
<property>
  <name>dfs.data.dir</name>
  <value>/home/amysmith/namenode</value>
</property>
<property>
  <name>dfs.data.dir</name>
  <value>/home/amysmith/datanode</value>
</property>
<property>
  <name>dfs.replication</name>
  <value>1</value>
</property>
</configuration>
```

If necessary, create the specific directories you defined for the `dfs.data.dir` value.

etc/hadoop/yarn-site.xml

Yarn is what we are using as our resource manager. Yarn-site contains configurations for the ResourceManager and NodeManager.

Note, we need to use the correct hostname of the NameNode under the `yarn.resourcemanager.hostname` setting. Under this setting, the current screenshot shows “namenode” which must be the hostname of the Ubuntu VM. Alternatively, you can use the IP address of your NameNode.

Note, the screenshot does not show the entire setting for `yarn.nodemanager.env-whitelist`. Here is the value per the Hadoop documentation at the time of this writing:

```
JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF_D
IR,CLASSPATH_PREPEND_DISTCACHE,HADOOP_YARN_HOME,HADOOP_HOME,PA
TH,LANG,TZ,HADOOP_MAPRED_HOME
```

```
GNU nano 6.2          yarn-site.xml *
-->

<configuration>

<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property>
<property>
  <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
  <value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
<property>
  <name>yarn.resourcemanager.hostname</name>
  <value>namenode</value>
</property>
<property>
  <name>yarn.acl.enable</name>
  <value>0</value>
</property>
<property>
  <name>yarn.nodemanager.env-whitelist</name>
  <value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF_DIR,CL>
</property>

</configuration>

<property>
  <name>yarn.nodemanager.env-whitelist</name>
  <value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF_DIR,CLASSPATH_PREPEND_DISTCACHE,HADOOP_YARN_HOME,HADOOP_HOME,PATH,LANG,TZ,
HADOOP_MAPRED_HOME</value>
</property>
```

```
<configuration>

<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property>
<property>
  <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
  <value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
<property>
  <name>yarn.resourcemanager.hostname</name>
  <value>namenode</value>
</property>
<property>
  <name>yarn.acl.enable</name>
  <value>0</value>
</property>
<property>
  <name>yarn.nodemanager.env-whitelist</name>
  <value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF_DIR,CLASSPATH_PREPEND_DISTCACHE,HADOOP_YARN_HOME,HADOOP_HOME,PATH,LANG,TZ,HADOOP_MAPRED_HOME</value>
</property>

</configuration>
```

etc/hadoop/mapred-site.xml

MapReduce settings likely are already correct per previous instructions but here is an example configuration:

```
GNU nano 6.2 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>
  <property>
    <name>mapreduce.application.classpath</name>
    <value>$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/*:$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/lib/*</value>
  </property>
</configuration>
```

```
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
  <property>
    <name>mapreduce.application.classpath</name>

    <value>$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/*:$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/lib/*</value>
  </property>
</configuration>
```

Hostname and DNS Settings

As we have discussed, the network settings are essential for a distributed cluster to function. It is critical to ensure that the NameNodes and DataNodes have DNS settings and static IP addresses configured so they can remain in constant communication. We are going to configure the hostnames and DNS settings for each distributed cluster node to ensure this. To do this, we are going to go into the /etc directory and edit the hostname and hosts files in Ubuntu. If you are in a different OS, you need to modify the proper files for your hostnames, static IP addresses, and DNS settings to be similar to these settings.

```
bobsmith@sone: /etc
bobsmith@sone:/etc$ cd /etc
bobsmith@sone:/etc$ pwd
/etc
bobsmith@sone:/etc$ sudo nano /etc/hostname
[sudo] password for bobsmith:
bobsmith@sone:/etc$ sudo nano /etc/hosts
```

A suggestion is to use sequential IP addresses and hostnames for your cluster such as HadoopOne, HadoopTwo or SparkOne, SparkTwo, etc. Try to use unique names from friends/peers to show your personal environment is different.

It is required for the IP addresses to parallel the IP addresses of the VMs. If the IP addresses are not the same as the hosts, this will not work! In addition, Host-Only mode IP addresses are essential historically for the clustered nodes to communicate.

Here is an example of an /etc/hosts file with one NameNode and two DataNodes:

```
bobsmith@sone: /etc
GNU nano 6.2 /etc/hosts *
127.0.0.1 localhost
#127.0.1.1 sone.myquest.virtualbox.org namenode

10.100.100.15 namenode
10.100.100.16 datanodeone
10.100.100.17 datanodetwo
```

NameNode Format and HDFS Troubleshooting if HDFS Will Not Start

One of the key aspects necessary for distributed systems to function is for their data to be parallel. If data between NameNodes and/or DataNodes is different, the cluster will not function. One of the steps you may need to run to get your cluster to work again is to reset your data. Note, this will erase work you have done in HDFS. **Thus, you need to backup any outputs or results from MapReduce jobs if you want to keep these BEFORE AN HDFS FORMAT.**

At this time, format your NameNode by running the command:

```
hdfs namenode -format
```

Note, once you clone your NameNode and create your DataNodes, you will need to reformat HDFS most likely before you startup the cluster. Again, this is to ensure all of your data is synchronized between your clustered systems.

If at any time you switch between DFS replication (e.g. set replication to 1), run Hadoop in single instance mode again, or run with a different set of DataNodes you may need to re-format your NameNode again to get HDFS running again. Other methods can be used to synchronize NameNodes and DataNodes but for the purpose of brevity, we are not going to cover this in this lab. Feel free to do your own research on these methods if you want to retain HDFS data.

Fixing HDFS Failures

Hadoop HDFS issues can be complicated when you have a cluster environment. You may be able to synchronize the file system if it becomes incongruent between the nodes. As a last resort, you can remove your /home/hadoop/tmpdata, /home/hadoop/namenode and /home/hadoop/datanode HDFS directories and start over.

Be careful running remove commands in Linux as you could remove the entire operating system.

Thus, make a backup of your VM first! You can use the rm command with a -R to remove a directory and all of its subdirectories:

```
rm -R /home/hadoop/namenode
```

```
rm -R /home/hadoop/datanode
```

```
rm -R /home/hadoop/tmpdata
```

```
rm -R /home/namenode
```

```
rm -R /home/datanode
```

```
rm -R /home/tmpdata
```

If necessary, you can use “rm -Rf” to force the removal.

```
rm -Rf /home/hadoop/namenode
```


Once these directories are removed, create them again and re-format the namenode. Make sure your **etc/hadoop/hdfs-site.xml** has the correct directory paths to your new namenode and datanode directories if you changed their names or directory locations.

```
mkdir /home/namenode
```

```
mkdir /home/datanode
```

```
mkdir /home/hadoop/tmpdata
```

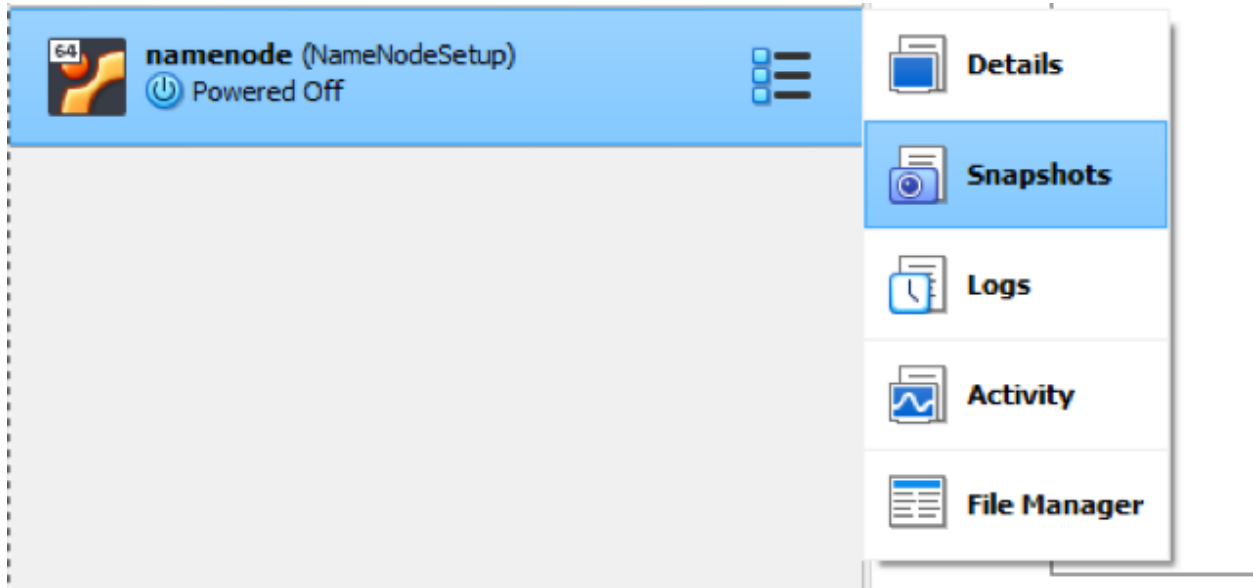
Once you have your new directories, you should format your namenode and it should create the proper directories and files under these newly created directories:

```
hdfs namenode -format
```

Cloning Hadoop Nodes

The minimum Hadoop cluster in this class will have one (1) NameNode and two (2) DataNodes. Prior to cloning the Hadoop single instance VM, ensure you 1) Boot up the Hadoop instance with the new settings and verify Hadoop is working prior so that you do not need to fix things on multiple VMs, 2) Take a snapshot / backup of the VM, and 3) Export an OVA image in case of a clone or other failure.

Allot of time has been spent setting up Hadoop. Thus, proper backups are essential.



You can keep snapshots or use the current state of the machine depending on your preference. Snapshots will require more disk space, of course.

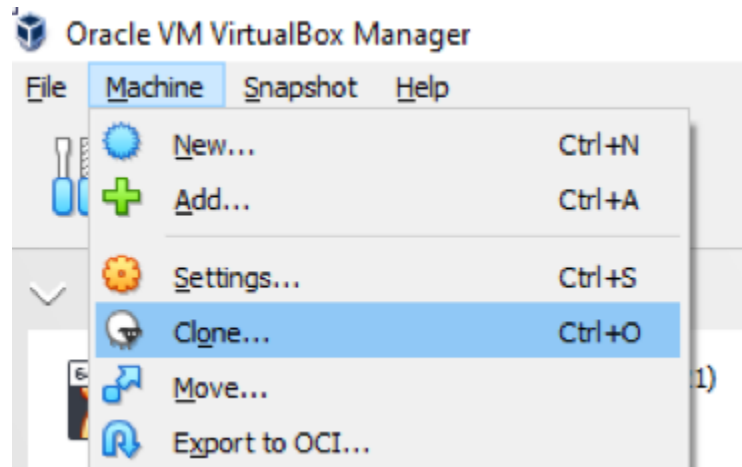
Students generally take two approaches to creating multiple NameNodes and DataNodes. Cloning or creating brand new VMs and installing Ubuntu again like in the first part of this class.

The method that is most consistent and less error prone is to install Hadoop individually on separate new VMs rather than cloning.

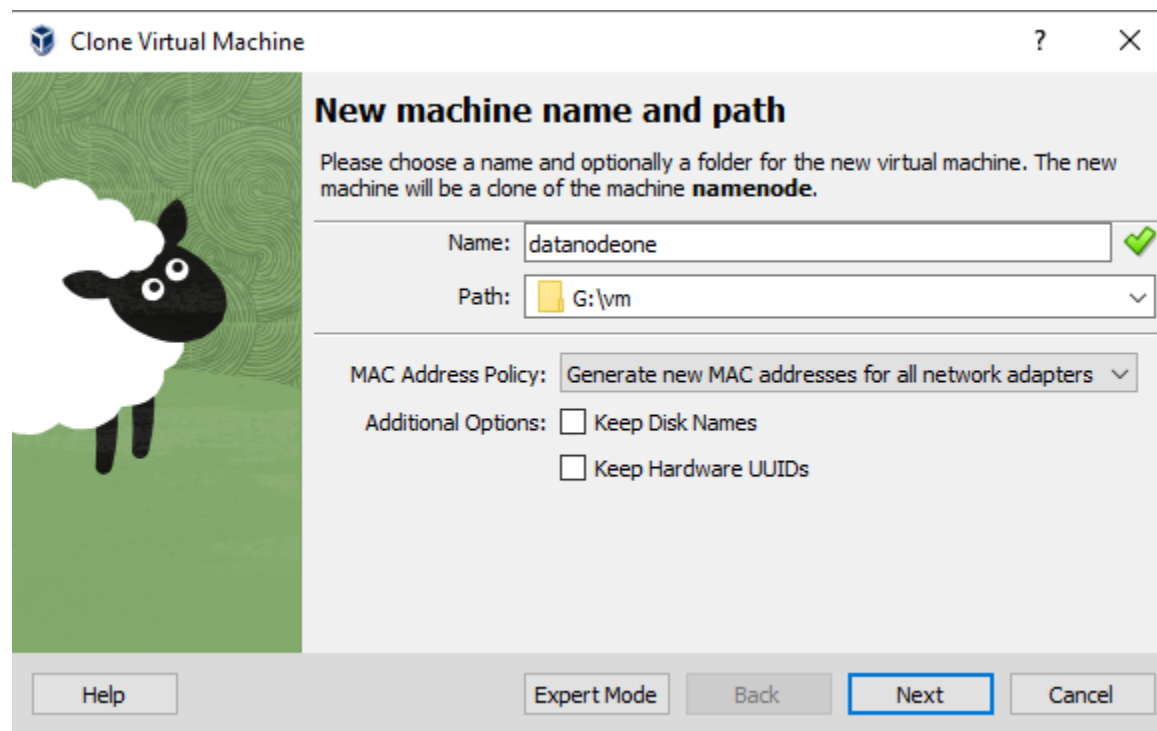
Cloning Method

The method that is most consistent and less error prone is to install Hadoop individually on separate new VMs rather than cloning.

Next, go to the VirtualBox settings of your NameNode VM and clone it:

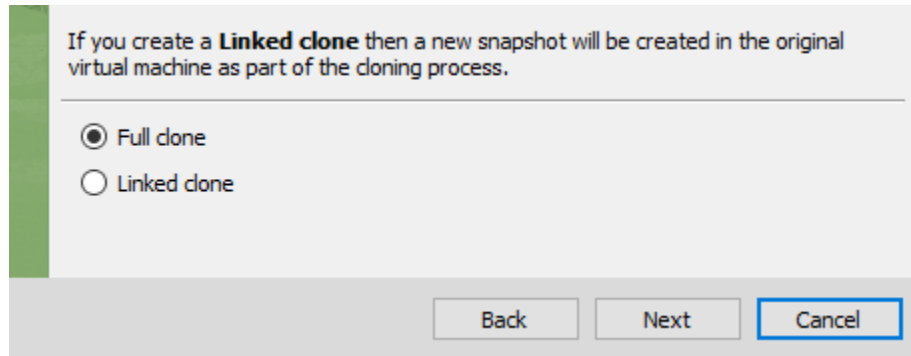


When you make a clone, ensure the path does not have any spaces or odd permissions. Note, use a new folder directly in your root hard drive:



Within the clone ensure you generate a new mac address for all network adapters. Change the name to the naming conventions of your Hadoop DataNodes. Do not keep disk names or hardware UUIDs.

Perform a Full Clone:



Cloning VMs can create problems. A common problem is that the DataNodes and NameNodes think they have the same UUIDs.

Instead of cloning, the recommended method is to install new Ubuntu VMs like we learned with the single instance Hadoop node.

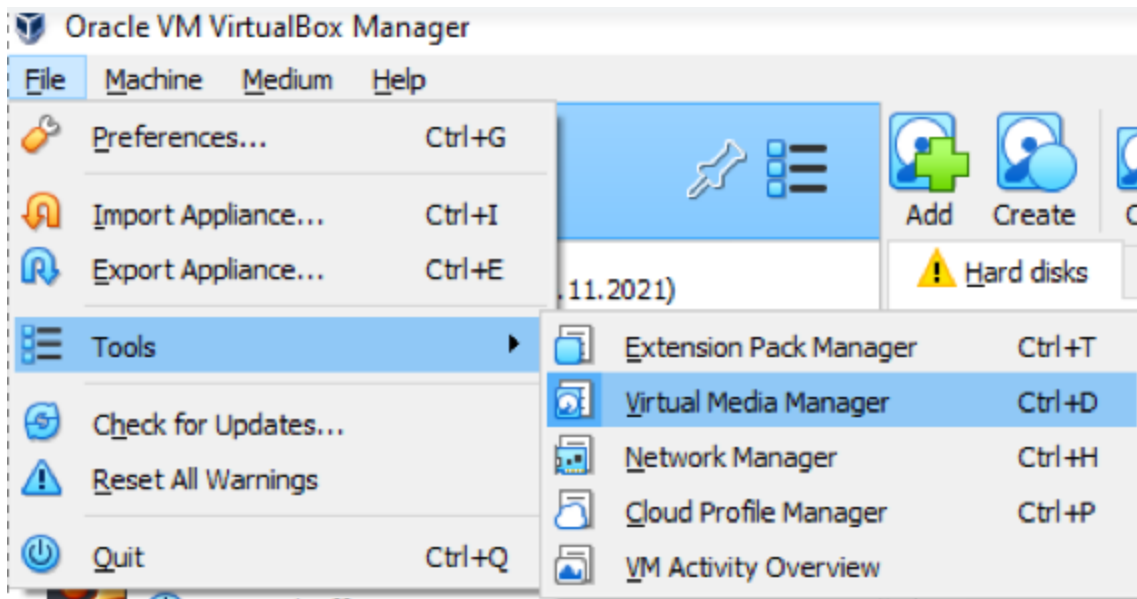
If you choose to clone, you will have to reset all of your UUIDs. To do this, you have to open a Windows Command Prompt and use the “VBoxManage” tool to change your UUIDs.

VBoxManage is located under your Oracle\VirtualBox installation folder. You have to navigate to this folder from a Windows CLI and run a command on your VDI files. Here is an example:

VBoxManage internalcommands sethduuid G:\vm\namenode\sone.vdi

```
Administrator: Command Prompt
UUID changed to: 2c22d8f9-efd7-42e7-8d49-663439fa65c1
G:\programfiles\Oracle\VirtualBox>
G:\programfiles\Oracle\VirtualBox>VBoxManage internalcommands sethduuid "G:\vm\datanodetwo\datanodetwo-disk1.vdi"
UUID changed to: 9de0781e-7cba-4457-a527-7a7f5b82b0f0
G:\programfiles\Oracle\VirtualBox>
G:\programfiles\Oracle\VirtualBox>VBoxManage internalcommands sethduuid "G:\vm\namenode\sone.vdi"
UUID changed to: 5db53f2d-31d1-4345-b0b4-2b3119f92751
```

While the command is fairly simply, it may require additional troubleshooting to get your new UUIDs to work. One step often needed is to release your VDI disk from your VM and re-add it. Again, this can cause your VM to fail. To release and remove VDI disks from VMs you can use the VirtualMediaManager under Tools:



Repeat the clone steps for EACH DataNode you create. A minimum of two (2) DataNodes and one (1) NameNode is required for the course assignments and projects.

Configuring A Hadoop Cluster

This part of the lab assumes that you have a working Hadoop NameNode that was configured as a cluster and cloned at least twice. Thus, there should be three (3) working Hadoop VMs to start this portion of the lab.

Steps to Starting and Shutting Down a Hadoop Cluster

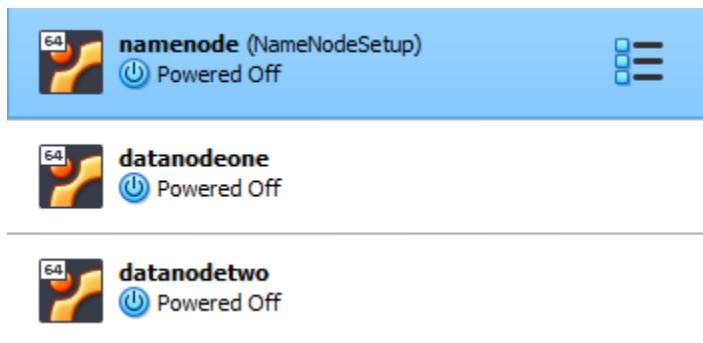
Once the three Hadoop VMs exist ensure that you always follow this order of events sequentially:

- 1) Always start the NameNode VM first
- 2) Start the second and third DataNodes in order.
- 3) Before starting Hadoop, ensure all nodes are in Host-Only networking mode and in the same IP address subnet.
- 4) Once the VMs are running, login to all three using your firstname lastname username.
- 5) Before starting the Hadoop cluster, ensure you can SSH to all VMs from the namenode without a password.
- 6) If you cannot SSH between the VMs, the cluster will not function properly and thus you should NOT start HDFS or you may corrupt the file system.
- 7) Always start HDFS and Yarn from the NameNode
- 8) Always start HDFS first, then other Hadoop applications
- 9) Always shutdown HDFS first, then other Hadoop applications
- 10) Always shutdown your operating system properly “sudo shutdown now”

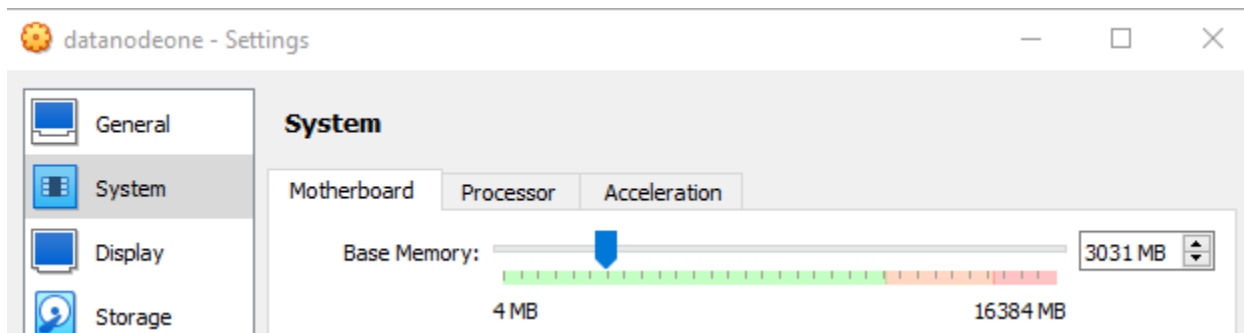
Configuring DataNodes

The new VMs whether additional NameNodes or DataNodes need to have unique hostnames and unique IP addresses in the same IP subnet. Thus, you need to configure the new NIC cards in these VMs, configure a new hostname, and configure the /etc/hosts file.

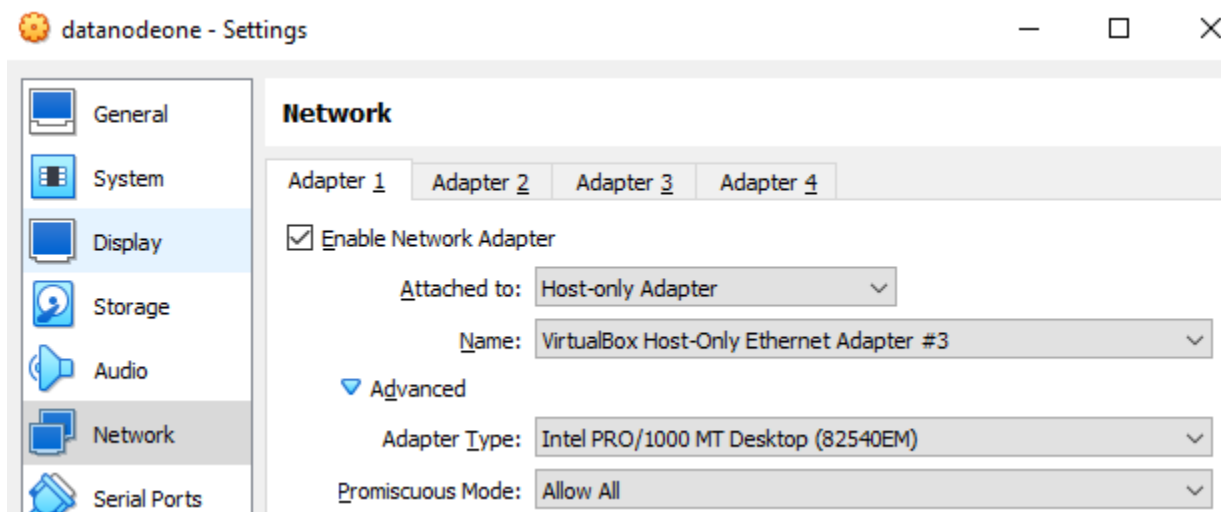
After clone the NameNode at least twice, you should see three VMs in VirtualBox:



Ensure you have sufficient available RAM and CPU cores for all three (3) VMs to be running at the same time.



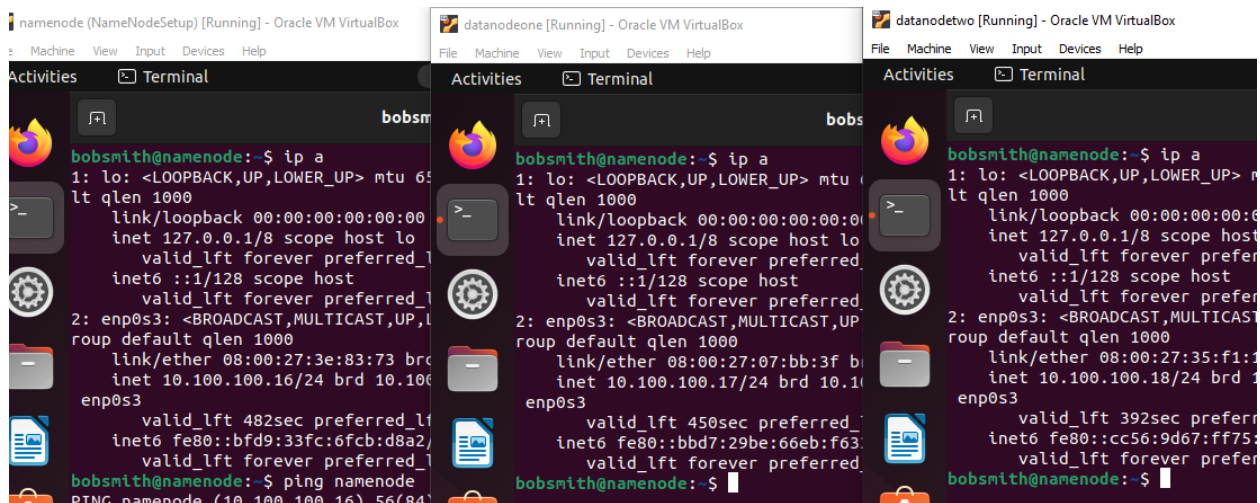
Refresh the NIC MAC address of all VMs. Make sure all VMs are in Host-Only Mode and in the same IP subnet:



Start the NameNode VM, wait 30 seconds, then start the first DataNode, wait 30 seconds, and finally start the second DataNode.

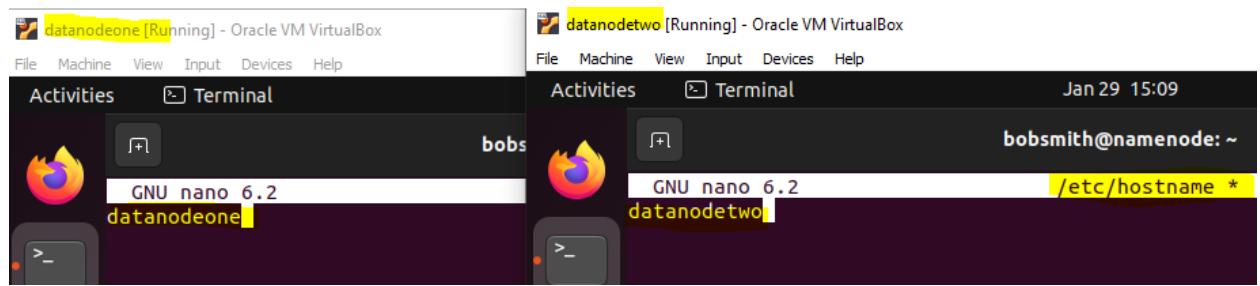
Ensure you change the hostname of the cloned VMs to be unique. They are still likely going to have the hostname of the cloned VM. Notice the VM names are different in the next screenshot but the hostnames need to be changed.

```
sudo nano /etc/hostname
```



Three terminal windows are shown, each displaying the output of the `ip a` command. The first window, titled 'namenode (NameNodeSetup) [Running] - Oracle VM VirtualBox', shows the IP address `10.100.100.16` for the `enp0s3` interface. The second window, titled 'datanodeone [Running] - Oracle VM VirtualBox', shows the IP address `10.100.100.17` for the `enp0s3` interface. The third window, titled 'datanodetwo [Running] - Oracle VM VirtualBox', shows the IP address `10.100.100.18` for the `enp0s3` interface. All three windows show the same network configuration details for the loopback and ethernet interfaces.

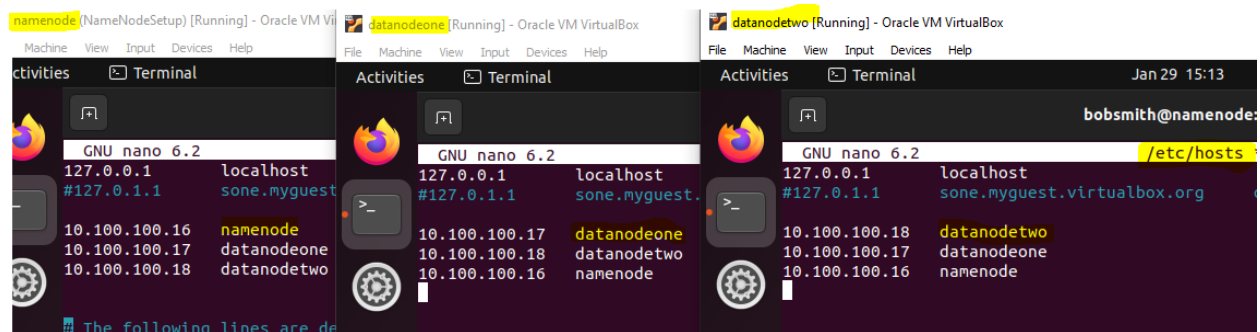
New hostnames:



Two terminal windows are shown, each displaying the `/etc/hostname` file being edited with `GNU nano 6.2`. The first window, titled 'datanodeone [Running] - Oracle VM VirtualBox', shows the file content `datanodeone`. The second window, titled 'datanodetwo [Running] - Oracle VM VirtualBox', shows the file content `datanodetwo`. Both windows show the file path `/etc/hostname` and the user `bobsmith@namenode`.

Save these files with the correct hostnames.

Next, edit the `/etc/hosts` file to have the correct IP addresses for each host:



Three terminal windows are shown, each displaying the `/etc/hosts` file being edited with `GNU nano 6.2`. The first window, titled 'namenode (NameNodeSetup) [Running] - Oracle VM VirtualBox', shows the file content `127.0.0.1 localhost` and `#127.0.1.1 sone.myquest`. The second window, titled 'datanodeone [Running] - Oracle VM VirtualBox', shows the file content `127.0.0.1 localhost` and `#127.0.1.1 sone.myquest`. The third window, titled 'datanodetwo [Running] - Oracle VM VirtualBox', shows the file content `127.0.0.1 localhost` and `#127.0.1.1 sone.myquest.virtualbox.org`. All three windows show the file path `/etc/hosts` and the user `bobsmith@namenode`.

If your network settings, hostname settings, and /etc/hosts files are all correct, you should be able to communicate/talk/ping all of the VMs with their hostnames. Test your network settings at this time to ensure ICMP replies exist between all hosts:

```
bobsmith@namenode: ~  
bobsmith@namenode:~$ sudo nano /etc/hosts  
[sudo] password for bobsmith:  
bobsmith@namenode:~$ ping namenode  
PING namenode (10.100.100.16) 56(84) bytes of data.  
64 bytes from namenode (10.100.100.16): icmp_seq=1 ttl=64 time=0.018 ms  
64 bytes from namenode (10.100.100.16): icmp_seq=2 ttl=64 time=0.021 ms  
^C  
--- namenode ping statistics ---  
2 packets transmitted, 2 received, 0% packet loss, time 1028ms  
rtt min/avg/max/mdev = 0.018/0.021/0.025/0.003 ms  
bobsmith@namenode:~$ ping datanodeone  
PING datanodeone (10.100.100.17) 56(84) bytes of data.  
64 bytes from datanodeone (10.100.100.17): icmp_seq=1 ttl=64 time=0.408 ms  
64 bytes from datanodeone (10.100.100.17): icmp_seq=2 ttl=64 time=0.432 ms  
64 bytes from datanodeone (10.100.100.17): icmp_seq=3 ttl=64 time=0.454 ms  
^C  
--- datanodeone ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 2033ms  
rtt min/avg/max/mdev = 0.408/0.432/0.454/0.018 ms  
bobsmith@namenode:~$ ping datanodetwo  
PING datanodetwo (10.100.100.18) 56(84) bytes of data.  
64 bytes from datanodetwo (10.100.100.18): icmp_seq=1 ttl=64 time=0.416 ms  
64 bytes from datanodetwo (10.100.100.18): icmp_seq=2 ttl=64 time=0.577 ms  
64 bytes from datanodetwo (10.100.100.18): icmp_seq=3 ttl=64 time=0.867 ms  
^C  
--- datanodetwo ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 2052ms  
rtt min/avg/max/mdev = 0.416/0.577/0.867/0.205 ms
```

Install OpenSSH on Ubuntu

Install the OpenSSH server and client using the following command:

```
sudo apt install openssh-server openssh-client -y
```

In the example below, the output confirms that the latest version is already installed.

```
pnap@pnap-VirtualBox:~$ sudo apt-get install openssh-server openssh-client
Reading package lists... Done
Building dependency tree
Reading state information... Done
openssh-client is already the newest version (1:7.6p1-4ubuntu0.3).
openssh-server is already the newest version (1:7.6p1-4ubuntu0.3).
0 upgraded, 0 newly installed, 0 to remove and 54 not upgraded.
```

If you have installed OpenSSH for the first time, use this opportunity to implement these vital [SSH security recommendations](#).

Create Username Using Your FirstName and LastName on EACH server

Sudo in as root:

```
sudo su -
```

Utilize the **adduser** command to create a new Hadoop user where “amysmith” is your actual first name and last name:

```
sudo adduser amysmith
```

Add the user to the sudoers group so they have elevated OS permissions, replace “amysmith” with you’re the new user account you created:

```
sudo usermod -aG sudo amysmith
```

The username, in this example, is **hdoop**. You are free to use any username and password you see fit. Switch to the newly created user and enter the corresponding password:

```
su - amysmith
```

The user now needs to be able to SSH to the localhost without being prompted for a password.

Enable Passwordless SSH for the Hadoop User on EACH server

[Generate an SSH key pair on EACH server](#) and define the location it is to be stored in. NOTE, be careful with the quotes, these are SINGLE quote marks and no space in between. There are no spaces in file directories either. Otherwise, there are spaces between the options.

```
ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
```

The system proceeds to generate and save the SSH key pair.

```
hadoop@pnap-VirtualBox:~$ ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
Generating public/private rsa key pair.
Your identification has been saved in /home/hadoop/.ssh/id_rsa.
Your public key has been saved in /home/hadoop/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:9GEXT7rDLjtcp5dT4KE0LevsYzloAKxir/E0zPjP58Y hadoop@pnap-VirtualBox
The key's randomart image is:
+---[RSA 2048]---+
|                 . .                 |
|                =                   |
|               . . 0 0..            |
|              0. 0 ++.+            |
|             .S ..+* 0             |
|    0+.      . .+.0 .             |
|   .00=     . 0.=.+ 0             |
|    =.0  E  =00 +                |
|   ..0.0+...+.+ .              |
+---[SHA256]-----+
hadoop@pnap-VirtualBox:~$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
hadoop@pnap-VirtualBox:~$ chmod 0600 ~/.ssh/authorized_keys
```

Use the **cat** command to store the public key as **authorized_keys** in the **ssh** directory:

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

Check the **authorized_keys** file and compare it to the public certificated.

Set the permissions for your user with the **chmod** command:

```
chmod 0600 ~/.ssh/authorized_keys
```

The new user is now able to SSH without needing to enter a password every time. Verify everything is set up correctly by using the **new user** to SSH to localhost:

```
ssh localhost
```

After an initial prompt, the Hadoop user is now able to establish an SSH connection to the localhost seamlessly.

Copy the SSH key to each server

In order for the cluster to function using SSH, we need to ensure all HDFS servers have the proper SSH keys in the SSH **authorized_keys** file. For more information see the tutorial: <https://linuxhint.com/use-ssh-copy-id-command/> or <https://linuxopsys.com/topics/ssh-copy-id-command>

We can manually move the SSH key from the NameNode to each DataNode. We can also use the **ssh-copy-id** command to do this. Here are examples run from the NameNode server:

```
ssh-copy-id bobsmith@namenode
```

```
ssh-copy-id bobsmith@datanodeone
```

```
ssh-copy-id bobsmith@datanodetwo
```

When these commands are successful, it should say at least one (1) key was successfully added to the server.

Add the DataNodes as Workers

Edit the Hadoop workers file to include the two DataNodes as workers:

```
bobsmith@namenode:~$ cd hadoop/etc/hadoop/
bobsmith@namenode:~/hadoop/etc/hadoop$ pwd
/home/bobsmith/hadoop/etc/hadoop
bobsmith@namenode:~/hadoop/etc/hadoop$ ls
capacity-scheduler.xml      kms-log4j.properties
configuration.xml           kms-site.xml
container-executor.cfg      log4j.properties
core-site.xml               mapred-env.cmd
hadoop-env.cmd              mapred-env.sh
hadoop-env.sh               mapred-queues.xml.templ
hadoop-metrics2.properties  mapred-site.xml
hadoop-policy.xml           shellprofile.d
hadoop-user-functions.sh.example  ssl-client.xml.example
hdfs-rbf-site.xml           ssl-server.xml.example
hdfs-site.xml               user_ec_policies.xml.te
httpfs-env.sh               workers
httpfs-log4j.properties     yarn-env.cmd
httpfs-site.xml             yarn-env.sh
kms-acls.xml                yarnservice-log4j.prope
kms-env.sh                  yarn-site.xml
bobsmith@namenode:~/hadoop/etc/hadoop$ nano workers
```

Workers file should have both your DataNodes:

```
bobsmith@namenode: ~/hadoop/etc/hadoop
GNU nano 6.2 workers *
datanodeone
datanodetwo

```

At this time, shutdown all of your VMs properly:

```
sudo shutdown now
```

Once they are shut down, start them up again and ensure they can communicate by pinging and SSHing to each of the servers from the NameNode. Ensure you select “Yes” to add every host to the known hosts:

```
bobsmith@datanodeone: ~  
bobsmith@namenode:~$ ssh datanodeone  
The authenticity of host 'datanodeone (10.100.100.17)' can't be established.  
ED25519 key fingerprint is SHA256:Ht9mQvv2vLy4iCcnPeXBromi/QR+VqR601tcgY6thD0.  
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 'datanodeone' (ED25519) to the list of known hosts.  
  
bobsmith@datanodetwo: ~  
bobsmith@namenode:~$ ssh datanodetwo  
The authenticity of host 'datanodetwo (10.100.100.18)' can't be established.  
ED25519 key fingerprint is SHA256:Ht9mQvv2vLy4iCcnPeXBromi/QR+VqR601tcgY6thD0.  
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]  
  ~/.ssh/known_hosts:6: [hashed name]  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added 'datanodetwo' (ED25519) to the list of known hosts.  
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-57-generic x86_64)
```

SSH to EACH DataNode and SSH to each NameNode and DataNode from these VMs adding every host to the list of known hosts.

Starting a Hadoop Cluster for the First Time

After you can SSH to all of your NameNodes and DataNodes seamlessly without issue and without a password, it is time to start HDFS.

A suggestion is to remove all the contents of the namenode and datanode directories on ALL NameNodes and DataNodes first. After this, you should not have to remove this data assuming you do not corrupt the HDFS file system.

Be careful running remove commands in Linux as you could remove the entire operating system. Thus, make a backup of your VM first! You can use the rm command with a -R to remove a directory and all of its subdirectories on your DataNodes and NameNodes:

```
rm -R /home/hadoop/namenode/*
```

```
rm -R /home/hadoop/datanode/*
```

```
rm -R /home/hadoop/tmpdata/*
```

```
rm -R /home/namenode/*
```

```
rm -R /home//datanode/*
```

Once these directories are removed, create them again if necessary and re-format the namenode. Make sure your **etc/hadoop/hdfs-site.xml** has the correct directory paths to your new namenode and datanode directories if you changed their names or directory locations.

```
mkdir /home/namenode
```

```
mkdir /home/datanode
```

```
mkdir /home/tmpdata
```

Format HDFS:

```
hdfs namenode -format
```

In a Hadoop cluster, we only start HDFS and Yarn from the NameNode. Thus, on the NameNode go to the /hadoop/sbin directory and start Hadoop like normal.

```
bobsmith@namenode: ~/hadoop/sbin
bobsmith@namenode:~/hadoop/sbin$ cd hadoop/sbin/
bobsmith@namenode:~/hadoop/sbin$ ls
distribute-exclude.sh  refresh-namenodes.sh  start-yarn.cmd
FederationStateStore  start-all.cmd         start-yarn.sh
hadoop-daemon.sh       start-all.sh         stop-all.cmd
hadoop-daemons.sh     start-balancer.sh     stop-all.sh
httpfs.sh              start-dfs.cmd         stop-balancer.sh
kms.sh                 start-dfs.sh          stop-dfs.cmd
mr-jobhistory-daemon.sh start-secure-dns.sh   stop-dfs.sh
bobsmith@namenode:~/hadoop/sbin$ ./start-dfs.sh
Starting namenodes on [namenode]
Starting datanodes
Starting secondary namenodes [namenode]
bobsmith@namenode:~/hadoop/sbin$ ./start-yarn.sh
Starting resourcemanager
Starting nodemanagers
bobsmith@namenode:~/hadoop/sbin$
```

Check to see if the NameNode is running on the NameNode and the DataNode is running on your DataNodes:

```
bobsmith@namenode: ~/hadoop
bobsmith@namenode:~/hadoop$ jps
5568 ResourceManager
5382 SecondaryNameNode
5159 NameNode
5879 Jps
bobsmith@namenode:~/hadoop$

bobsmith@datanodeone: ~/hadoop
bobsmith@datanodeone:~/hadoop$ jps
3507 NodeManager
3623 Jps
3370 DataNode
bobsmith@datanodeone:~/hadoop$
```

Check the NameNode log files on the NameNode and the DataNode log files on the DataNodes. You should see health checks are good, successful heartbeats, block reports, etc:

```
bobsmith@datanodeone: ~/hadoop/logs
bobsmith@datanodeone:~/hadoop/logs$ tail -f hadoop-bobsmith-datanode-datanodeone.log
2023-01-29 19:20:11,732 INFO org.apache.hadoop.hdfs.server.datanode.checker.ThrottledAsyncChecker: Scheduling a c
me/bobsmith/datanode
2023-01-29 19:20:11,740 INFO org.apache.hadoop.hdfs.server.datanode.checker.DatasetVolumeChecker: Scheduled healt
volume /home/bobsmith/datanode
2023-01-29 19:20:11,751 INFO org.apache.hadoop.hdfs.server.datanode.VolumeScanner: VolumeScanner(/home/bobsmith/d
2b26544f-df2b-4de6-9ed6-8124f807f2ec): no suitable block pools found to scan. Waiting 1814110095 ms.
2023-01-29 19:20:11,753 WARN org.apache.hadoop.hdfs.server.datanode.DirectoryScanner: dfs.datanode.directoryscan.
it.ms.per.sec set to value above 1000 ms/sec. Assuming default value of -1
2023-01-29 19:20:11,754 INFO org.apache.hadoop.hdfs.server.datanode.DirectoryScanner: Periodic Directory Tree Ver
an starting in 10177442ms with interval of 21600000ms and throttle limit of -1ms/s
2023-01-29 19:20:11,759 INFO org.apache.hadoop.hdfs.server.datanode.DataNode: Block pool BP-1114889918-10.100.100
00980 (Datanode Uuid 71da12af-5a84-4862-b024-ad09cde0928f) service to namenode/10.100.100.19:9000 beginning hands
2023-01-29 19:20:11,885 INFO org.apache.hadoop.hdfs.server.datanode.DataNode: Block pool BP-1114889918-10.100.100
00980 (Datanode Uuid 71da12af-5a84-4862-b024-ad09cde0928f) service to namenode/10.100.100.19:9000 successfully re
h NN
2023-01-29 19:20:11,885 INFO org.apache.hadoop.hdfs.server.datanode.DataNode: For namenode namenode/10.100.100.19
BLOCKREPORT_INTERVAL of 21600000msecs CACHEREPORT_INTERVAL of 10000msecs Initial delay: 0msecs; heartBeatInterval
2023-01-29 19:20:12,021 INFO org.apache.hadoop.hdfs.server.datanode.DataNode: Successfully sent block report 0xfd
e3 with lease ID 0x22962dd320fa88e0 to namenode: namenode/10.100.100.19:9000, containing 1 storage report(s), of
nt 1. The reports had 0 total blocks and used 1 RPC(s). This took 4 msecs to generate and 47 msecs for RPC and NN
Got back one command: FinalizeCommand/5.
2023-01-29 19:20:12,022 INFO org.apache.hadoop.hdfs.server.datanode.DataNode: Got finalize command for block pool
18-10.100.100.19-1675041300980
bobsmith@datanodeone:~/hadoop/logs$
```


Finally, ensure that you can create HDFS directories, place files in the HDFS directories, and run a MapReduce job or two. If the cluster is functional, you should see the DataNodes running the MapReduce job in Yarn:



▼ Cluster

[About](#)
[Nodes](#)
[Node Labels](#)
[Applications](#)
[NEW](#)
[NEW_SAVING](#)
[SUBMITTED](#)
[ACCEPTED](#)
[RUNNING](#)
[FINISHED](#)
[FAILED](#)
[KILLED](#)
[Scheduler](#)

► Tools

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers
0	0	0	0	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	De
2	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memor

Show 20 ▼ entries

Node Labels ▲	Rack ◆	Node State ◆	Node Address ◆	Node HTTP Address ◆
	/default-rack	RUNNING	datanodetwo:40075	datanodetwo:8042
	/default-rack	RUNNING	datanodeone:41307	datanodeone:8042