Name: Ejercito, Marlon Jason A.	**Date Performed:15/08/2023**
Course/Section: CPE 232 / CPE31S4	**Date Submitted:15/08/2023**
Instructor: Dr. Jonathan Taylar	**Semester and SY: 1st Semester, 2023-2024**

Activity 1: Configure Network using Virtual Machines

1. Objectives:

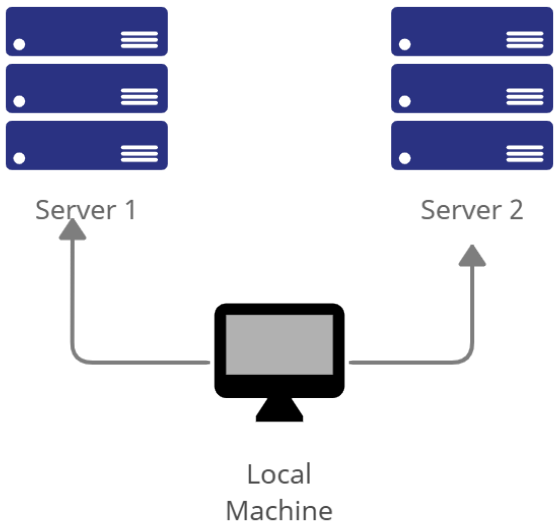
1.1. Create and configure Virtual Machines in Microsoft Azure or VirtualBox

1.2. Set-up a Virtual Network and Test Connectivity of VMs

2. Discussion:

Network Topology:

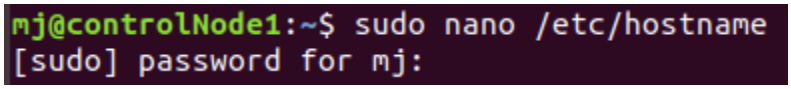
Assume that you have created the following network topology in Virtual Machines, *provide screenshots for each task*. (Note: *it is assumed that you have the prior knowledge of cloning and creating snapshots in a virtual machine*).

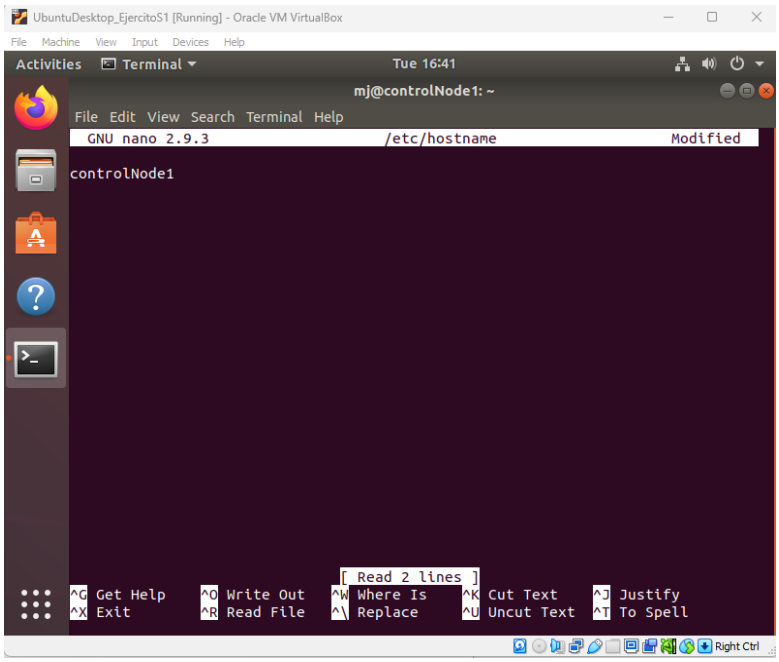


Task 1: Do the following on Server 1, Server 2, and Local Machine. In editing the file using nano command, press control + O to write out (save the file). Press enter when asked for the name of the file. Press control + X to end.

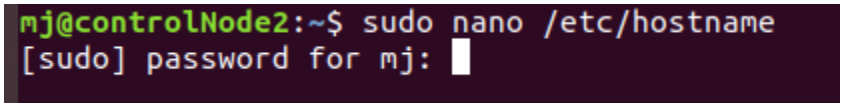
1. Change the hostname using the command *sudo nano /etc/hostname*

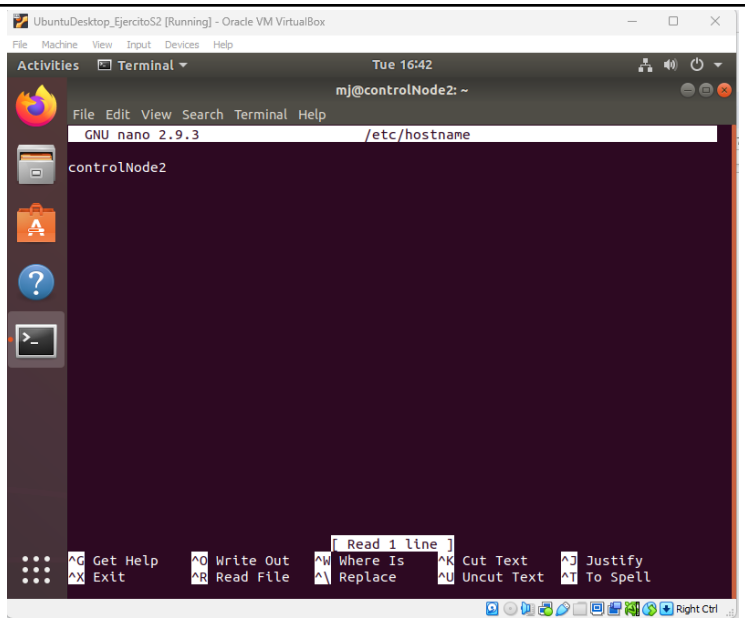
1.1 Use server1 for Server 1



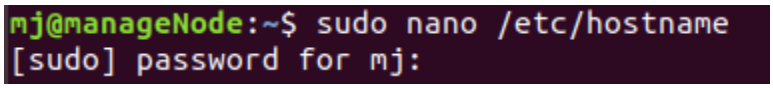


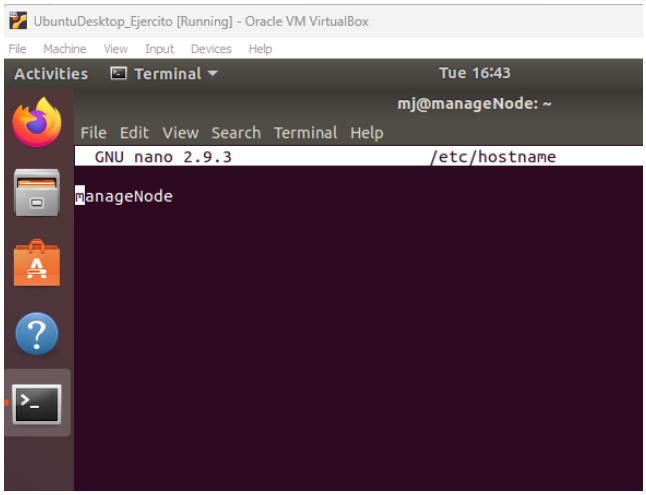
1.2 Use server2 for Server 2



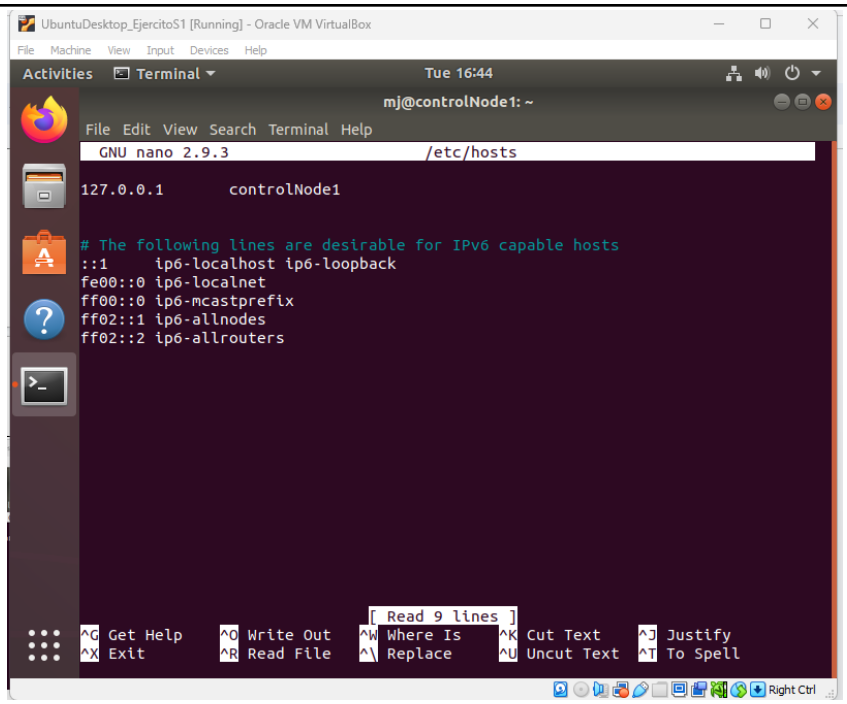


1.3 Use workstation for the Local Machine

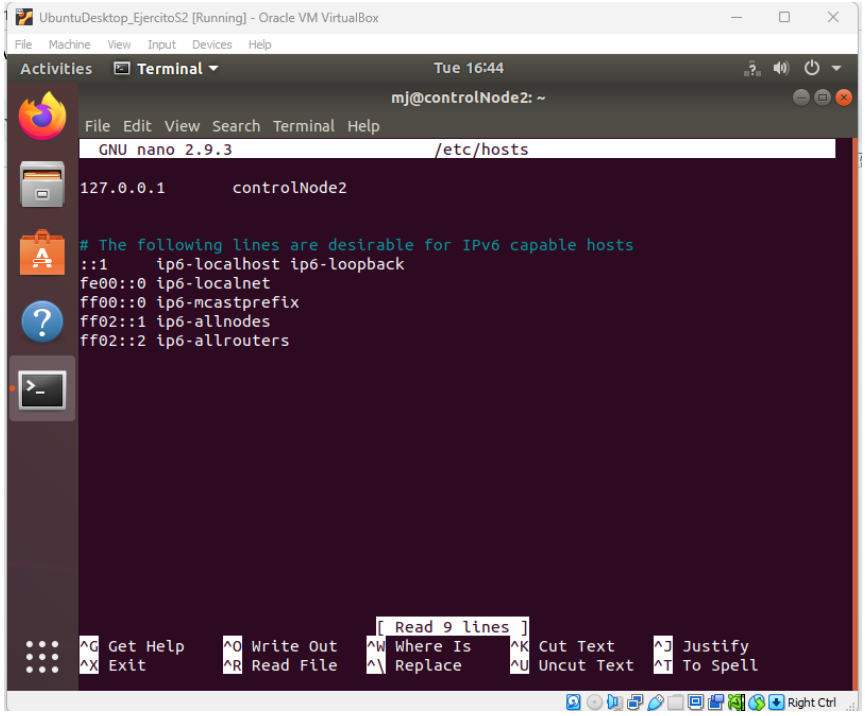




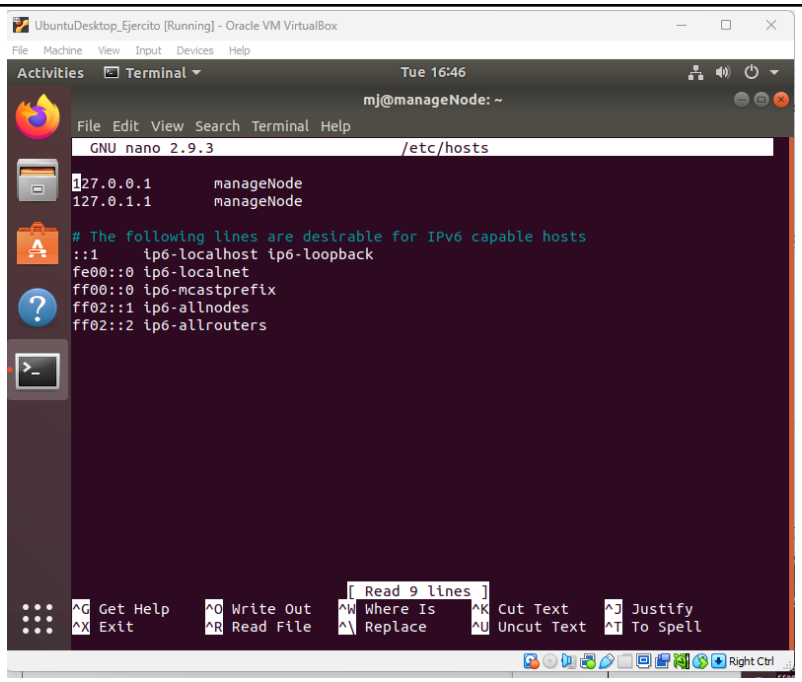
2. Edit the hosts using the command *sudo nano /etc/hosts*. Edit the second line.
   2.1 Type 127.0.0.1 server 1 for Server 1



2.2 Type 127.0.0.1 server 2 for Server 2



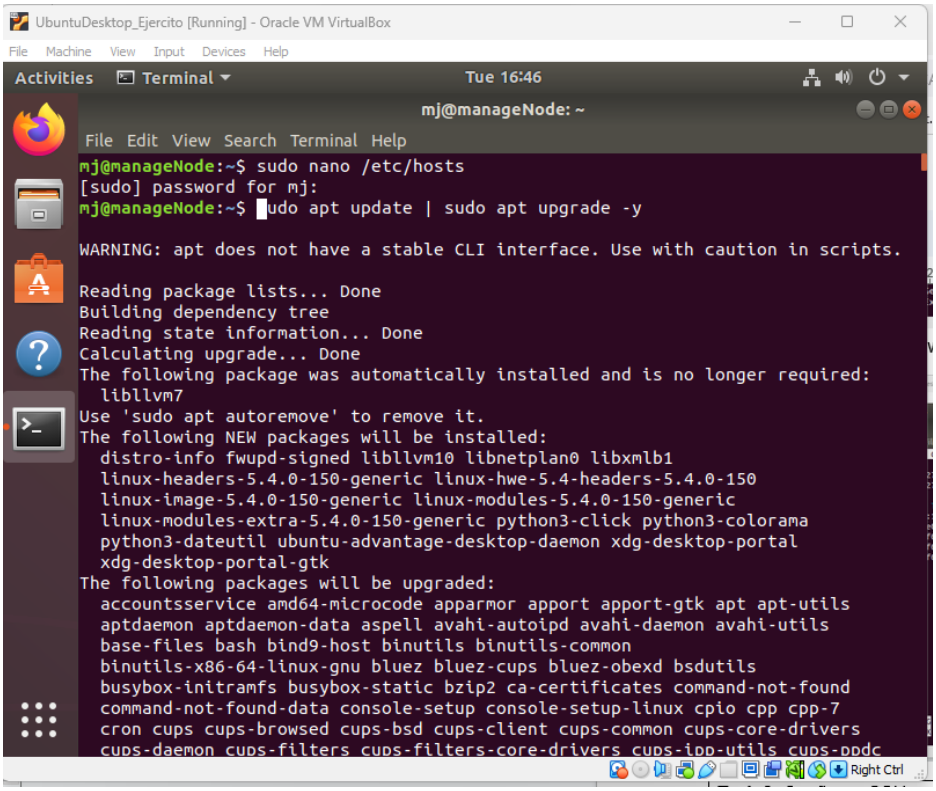
2.3 Type 127.0.0.1 workstation for the Local Machine



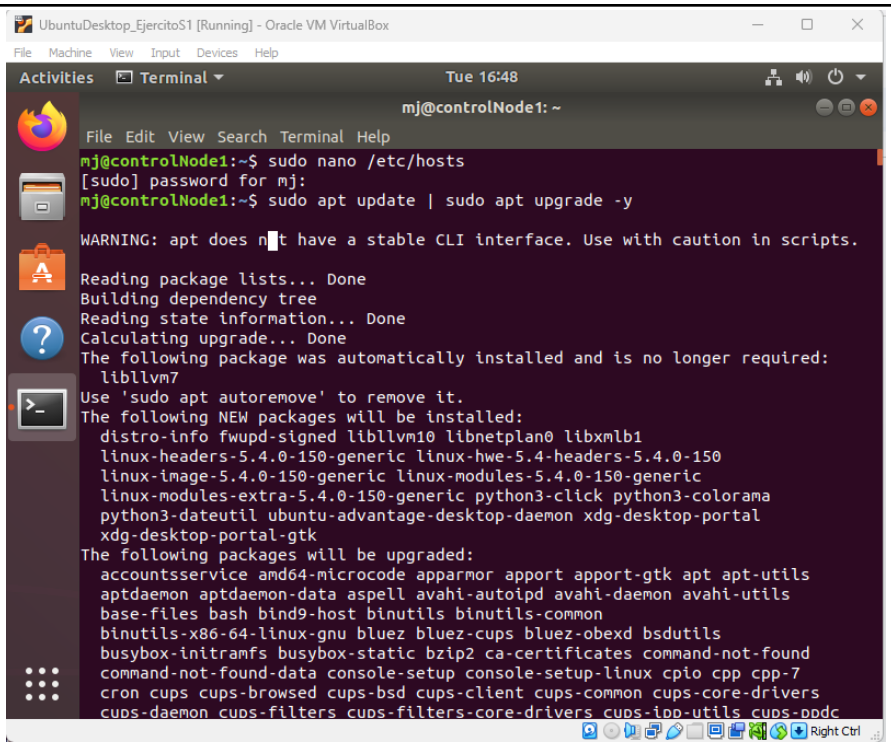
Task 2: Configure SSH on Server 1, Server 2, and Local Machine. Do the following:

1. Upgrade the packages by issuing the command *sudo apt update* and *sudo apt upgrade* respectively.

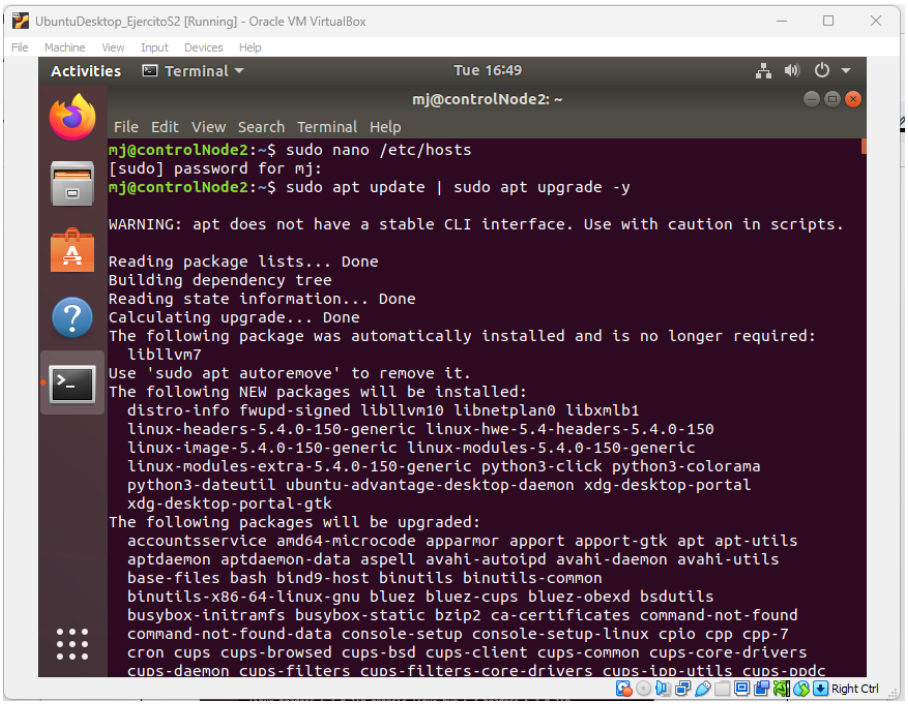
   Local Machine:



   Server 1:



Server 2:



2. Install the SSH server using the command *sudo apt install openssh-server*.
   Local Machine:

```
mj@manageNode:~$ sudo apt install openssh-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  libllvm7
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  ncurses-term openssh-sftp-server ssh-import-id
Suggested packages:
  molly-guard monkeysphere rssh ssh-askpass
The following NEW packages will be installed:
  ncurses-term openssh-server openssh-sftp-server ssh-import-id
0 upgraded, 4 newly installed, 0 to remove and 0 not upgraded.
Need to get 637 kB of archives.
After this operation, 5,320 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ph.archive.ubuntu.com/ubuntu bionic-updates/main amd64 ncurses-ter
m all 6.1-1ubuntu1.18.04.1 [248 kB]
Get:2 http://ph.archive.ubuntu.com/ubuntu bionic-updates/main amd64 openssh-sft
p-server amd64 1:7.6p1-4ubuntu0.7 [45.5 kB]
Get:3 http://ph.archive.ubuntu.com/ubuntu bionic-updates/main amd64 openssh-ser
ver amd64 1:7.6p1-4ubuntu0.7 [332 kB]
Get:4 http://ph.archive.ubuntu.com/ubuntu bionic-updates/main amd64 ssh-import-
id all 5.7-0ubuntu1.1 [10.9 kB]
Fetched 637 kB in 15s (42.2 kB/s)
```

Server 1:

```
Files
mj@controlNode1:~$ sudo apt install openssh-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  libllvm7
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  ncurses-term openssh-sftp-server ssh-import-id
Suggested packages:
  molly-guard monkeysphere rssh ssh-askpass
The following NEW packages will be installed:
  ncurses-term openssh-server openssh-sftp-server ssh-import-id
0 upgraded, 4 newly installed, 0 to remove and 0 not upgraded.
Need to get 637 kB of archives.
After this operation, 5,320 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ph.archive.ubuntu.com/ubuntu bionic-updates/main amd64 ncurses-ter
m all 6.1-1ubuntu1.18.04.1 [248 kB]
Get:2 http://ph.archive.ubuntu.com/ubuntu bionic-updates/main amd64 openssh-sft
p-server amd64 1:7.6p1-4ubuntu0.7 [45.5 kB]
Get:3 http://ph.archive.ubuntu.com/ubuntu bionic-updates/main amd64 openssh-ser
ver amd64 1:7.6p1-4ubuntu0.7 [332 kB]
Get:4 http://ph.archive.ubuntu.com/ubuntu bionic-updates/main amd64 ssh-import-
id all 5.7-0ubuntu1.1 [10.9 kB]
Fetched 637 kB in 18s (35.2 kB/s)
Preconfiguring packages ...
```

Server 2:

```
mj@controlNode2:~$ sudo apt install openssh-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  libllvm7
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  ncurses-term openssh-sftp-server ssh-import-id
Suggested packages:
  molly-guard monkeysphere rssh ssh-askpass
The following NEW packages will be installed:
  ncurses-term openssh-server openssh-sftp-server ssh-import-id
0 upgraded, 4 newly installed, 0 to remove and 0 not upgraded.
Need to get 637 kB of archives.
After this operation, 5,320 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ph.archive.ubuntu.com/ubuntu bionic-updates/main amd64 ncurses-ter
m all 6.1-1ubuntu1.18.04.1 [248 kB]
Get:2 http://ph.archive.ubuntu.com/ubuntu bionic-updates/main amd64 openssh-sft
p-server amd64 1:7.6p1-4ubuntu0.7 [45.5 kB]
Get:3 http://ph.archive.ubuntu.com/ubuntu bionic-updates/main amd64 openssh-ser
ver amd64 1:7.6p1-4ubuntu0.7 [332 kB]
Get:4 http://ph.archive.ubuntu.com/ubuntu bionic-updates/main amd64 ssh-import-
id all 5.7-0ubuntu1.1 [10.9 kB]
Fetched 637 kB in 5s (124 kB/s)
Preconfiguring packages ...
```

3. Verify if the SSH service has started by issuing the following commands:

   3.1 *sudo service ssh start*

   Local Machine:

```
mj@manageNode:~$ sudo service ssh start
```

   Server 1:

```
mj@controlNode1:~$ sudo service ssh start
```

   Server 2:

```
mj@controlNode2:~$ sudo service ssh start
```

   3.2 *sudo systemctl status ssh*

   Local Machine:

```
mj@manageNode:~$ sudo systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: ena
   Active: active (running) since Tue 2023-08-15 16:54:26 PST; 1min 15s ago
 Main PID: 20061 (sshd)
    Tasks: 1 (limit: 2318)
   CGroup: /system.slice/ssh.service
           └─20061 /usr/sbin/sshd -D

Aug 15 16:54:26 manageNode systemd[1]: Starting OpenBSD Secure Shell server...
Aug 15 16:54:26 manageNode sshd[20061]: Server listening on 0.0.0.0 port 22.
Aug 15 16:54:26 manageNode sshd[20061]: Server listening on :: port 22.
Aug 15 16:54:26 manageNode systemd[1]: Started OpenBSD Secure Shell server.
lines 1-12/12 (END)
```

   Server 1:

```
mj@controlNode1:~$ sudo systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: ena
     Active: active (running) since Tue 2023-08-15 16:54:24 PST; 3min 50s ago
   Main PID: 19817 (sshd)
      Tasks: 1 (limit: 2318)
     CGroup: /system.slice/ssh.service
             └─19817 /usr/sbin/sshd -D

Aug 15 16:54:24 controlNode1 systemd[1]: Starting OpenBSD Secure Shell server..
Aug 15 16:54:24 controlNode1 sshd[19817]: Server listening on 0.0.0.0 port 22.
Aug 15 16:54:24 controlNode1 sshd[19817]: Server listening on :: port 22.
Aug 15 16:54:24 controlNode1 systemd[1]: Started OpenBSD Secure Shell server.
lines 1-12/12 (END)
```

Server 2:

```
mj@controlNode2:~$ sudo systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: ena
     Active: active (running) since Tue 2023-08-15 16:54:04 PST; 6min ago
   Main PID: 20327 (sshd)
      Tasks: 1 (limit: 2318)
     CGroup: /system.slice/ssh.service
             └─20327 /usr/sbin/sshd -D

Aug 15 16:54:04 controlNode2 systemd[1]: Starting OpenBSD Secure Shell server..
Aug 15 16:54:04 controlNode2 sshd[20327]: Server listening on 0.0.0.0 port 22.
Aug 15 16:54:04 controlNode2 sshd[20327]: Server listening on :: port 22.
Aug 15 16:54:04 controlNode2 systemd[1]: Started OpenBSD Secure Shell server.
lines 1-12/12 (END)
```

4. Configure the firewall to all port 22 by issuing the following commands:

   4.1 *sudo ufw allow ssh*

   Local Machine:

```
mj@manageNode:~$ sudo ufw allow ssh
Rules updated
Rules updated (v6)
```

   Server 1:

```
mj@controlNode1:~$ sudo ufw allow ssh
Rules updated
Rules updated (v6)
```

   Server 2:

```
mj@controlNode2:~$ sudo ufw allow ssh
Rules updated
Rules updated (v6)
```

   4.2 *sudo ufw enable*

   Local Machine:

```
mj@manageNode:~$ sudo ufw enable
Firewall is active and enabled on system startup
```

   Server 1:

```
mj@controlNode1:~$ sudo ufw enable
Firewall is active and enabled on system startup
```

Server 2:

```
mj@controlNode2:~$ sudo ufw enable
Firewall is active and enabled on system startup
```

4.3 *sudo ufw status*

Local Machine:

```
mj@manageNode:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
22/tcp                     ALLOW       Anywhere
22/tcp (v6)                ALLOW       Anywhere (v6)
```

Server 1:

```
mj@controlNode1:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
22/tcp                     ALLOW       Anywhere
22/tcp (v6)                ALLOW       Anywhere (v6)
```

Server 2:

```
mj@controlNode2:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
22/tcp                     ALLOW       Anywhere
22/tcp (v6)                ALLOW       Anywhere (v6)
```

Task 3: Verify network settings on Server 1, Server 2, and Local Machine. On each device, do the following:

1. Record the ip address of Server 1, Server 2, and Local Machine. Issue the command *ifconfig* and check network settings. Note that the ip addresses of all the machines are in this network 192.168.56.XX.
   1.1 Local Machine IP address: 192.168.56.**103**

```
mj@manageNode:~$ 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::eb3a:b011:71af:b27b  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:c4:05:23  txqueuelen 1000  (Ethernet)
        RX packets 849178  bytes 1280807009 (1.2 GB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 169448  bytes 10490478 (10.4 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::4f:9573:992e:c28c  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:8d:26:ec  txqueuelen 1000  (Ethernet)
        RX packets 270  bytes 38661 (38.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 125  bytes 17595 (17.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

1.2 Server 1 IP address: 192.168.56.**102**

```
mj@controlNode1:~$ 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::e7bf:fa0d:6545:c618  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:8e:cb:61  txqueuelen 1000  (Ethernet)
        RX packets 850315  bytes 1281128630 (1.2 GB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 207052  bytes 12828790 (12.8 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.56.102  netmask 255.255.255.0  broadcast 192.168.56.255
        inet6 fe80::f1d0:2866:48ef:bfe  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:8f:a6:20  txqueuelen 1000  (Ethernet)
        RX packets 354  bytes 48727 (48.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 128  bytes 17875 (17.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

1.3 Server 2 IP address: 192.168.56.**101**

```
mj@controlNode2:~$ 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::db35:7b4e:3330:3379  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:dc:2f:da  txqueuelen 1000  (Ethernet)
        RX packets 850330  bytes 1281147071 (1.2 GB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 170215  bytes 10622221 (10.6 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.56.101  netmask 255.255.255.0  broadcast 192.168.56.255
        inet6 fe80::1c3a:a79a:5da0:3a17  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:e8:43:bc  txqueuelen 1000  (Ethernet)
        RX packets 305  bytes 41428 (41.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 126  bytes 17723 (17.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

2. Make sure that they can ping each other.

2.1 Connectivity test for Local Machine 1 to Server 1: ☐ Successful ☐ Not Successful

```
mj@manageNode:~$ ping 192.168.56.102
PING 192.168.56.102 (192.168.56.102) 56(84) bytes of data.
64 bytes from 192.168.56.102: icmp_seq=1 ttl=64 time=0.728 ms
64 bytes from 192.168.56.102: icmp_seq=2 ttl=64 time=0.425 ms
64 bytes from 192.168.56.102: icmp_seq=3 ttl=64 time=0.329 ms
64 bytes from 192.168.56.102: icmp_seq=4 ttl=64 time=0.930 ms
64 bytes from 192.168.56.102: icmp_seq=5 ttl=64 time=1.54 ms
64 bytes from 192.168.56.102: icmp_seq=6 ttl=64 time=1.22 ms
64 bytes from 192.168.56.102: icmp_seq=7 ttl=64 time=0.498 ms
64 bytes from 192.168.56.102: icmp_seq=8 ttl=64 time=0.577 ms
64 bytes from 192.168.56.102: icmp_seq=9 ttl=64 time=1.40 ms
64 bytes from 192.168.56.102: icmp_seq=10 ttl=64 time=0.704 ms
64 bytes from 192.168.56.102: icmp_seq=11 ttl=64 time=0.388 ms
64 bytes from 192.168.56.102: icmp_seq=12 ttl=64 time=0.695 ms
64 bytes from 192.168.56.102: icmp_seq=13 ttl=64 time=0.625 ms
64 bytes from 192.168.56.102: icmp_seq=14 ttl=64 time=0.700 ms
64 bytes from 192.168.56.102: icmp_seq=15 ttl=64 time=0.899 ms
64 bytes from 192.168.56.102: icmp_seq=16 ttl=64 time=2.01 ms
64 bytes from 192.168.56.102: icmp_seq=17 ttl=64 time=1.24 ms
64 bytes from 192.168.56.102: icmp_seq=18 ttl=64 time=0.910 ms
64 bytes from 192.168.56.102: icmp_seq=19 ttl=64 time=0.870 ms
^C
--- 192.168.56.102 ping statistics ---
19 packets transmitted, 19 received, 0% packet loss, time 18228ms
rtt min/avg/max/mdev = 0.329/0.879/2.018/0.427 ms
```

2.2 Connectivity test for Local Machine 1 to Server 2: ☐ Successful ☐ Not Successful

```
mj@manageNode:~$ ping 192.168.56.101
PING 192.168.56.101 (192.168.56.101) 56(84) bytes of data.
64 bytes from 192.168.56.101: icmp_seq=1 ttl=64 time=1.16 ms
64 bytes from 192.168.56.101: icmp_seq=2 ttl=64 time=1.67 ms
64 bytes from 192.168.56.101: icmp_seq=3 ttl=64 time=0.409 ms
64 bytes from 192.168.56.101: icmp_seq=4 ttl=64 time=1.51 ms
64 bytes from 192.168.56.101: icmp_seq=5 ttl=64 time=0.343 ms
^C
--- 192.168.56.101 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4035ms
rtt min/avg/max/mdev = 0.343/1.023/1.678/0.553 ms
mj@manageNode:~$
```

2.3 Connectivity test for Server 1 to Server 2: ☐ Successful ☐ Not Successful

```
mj@controlNode1:~$ ping 192.168.56.101
PING 192.168.56.101 (192.168.56.101) 56(84) bytes of data.
64 bytes from 192.168.56.101: icmp_seq=1 ttl=64 time=1.07 ms
64 bytes from 192.168.56.101: icmp_seq=2 ttl=64 time=1.37 ms
64 bytes from 192.168.56.101: icmp_seq=3 ttl=64 time=1.31 ms
64 bytes from 192.168.56.101: icmp_seq=4 ttl=64 time=0.472 ms
64 bytes from 192.168.56.101: icmp_seq=5 ttl=64 time=0.472 ms
64 bytes from 192.168.56.101: icmp_seq=6 ttl=64 time=0.800 ms
64 bytes from 192.168.56.101: icmp_seq=7 ttl=64 time=0.455 ms
64 bytes from 192.168.56.101: icmp_seq=8 ttl=64 time=0.764 ms
```

Task 4: Verify SSH connectivity on Server 1, Server 2, and Local Machine.

1. On the Local Machine, issue the following commands:

1.1 ssh username@ip_address_server1 for example, *ssh jvtaylar@192.168.56.120*

```
mj@manageNode:~$ ssh mj@192.168.56.102
The authenticity of host '192.168.56.102 (192.168.56.102)' can't be established
.
ECDSA key fingerprint is SHA256:ZpoI4S5Y3kD3NmRXgvvz+sHLatLsNTEF2mNhLxORzdQ.
Are you sure you want to continue connecting (yes/no)? y
Please type 'yes' or 'no': yes
```

1.2 Enter the password for server 1 when prompted

```
mj@192.168.56.102's password:
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 4.18.0-15-generic x86_64)

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

Expanded Security Maintenance for Infrastructure is not enabled.

0 updates can be applied immediately.

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

Your Hardware Enablement Stack (HWE) is supported until April 2023.
*** System restart required ***

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

mj@controlNode1:~$
```

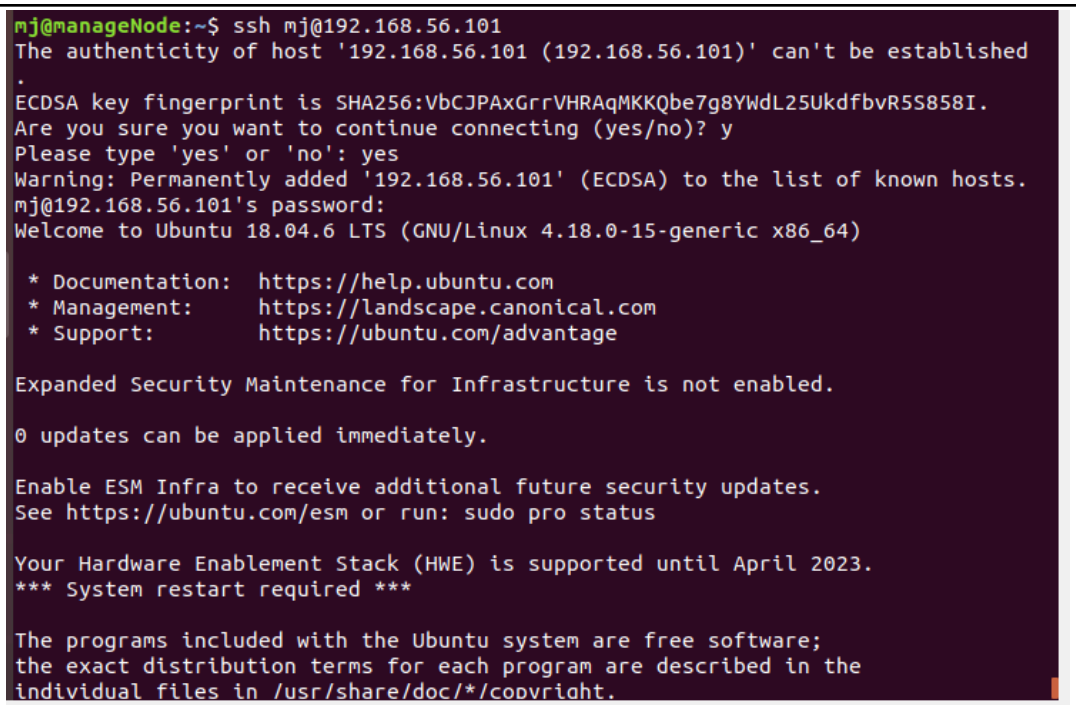
1.3 Verify that you are in server 1. The user should be in this format user@server1. For example, *jvtaylar@server1*

```
mj@controlNode1:~$
```

2. Logout of Server 1 by issuing the command *control + D.*

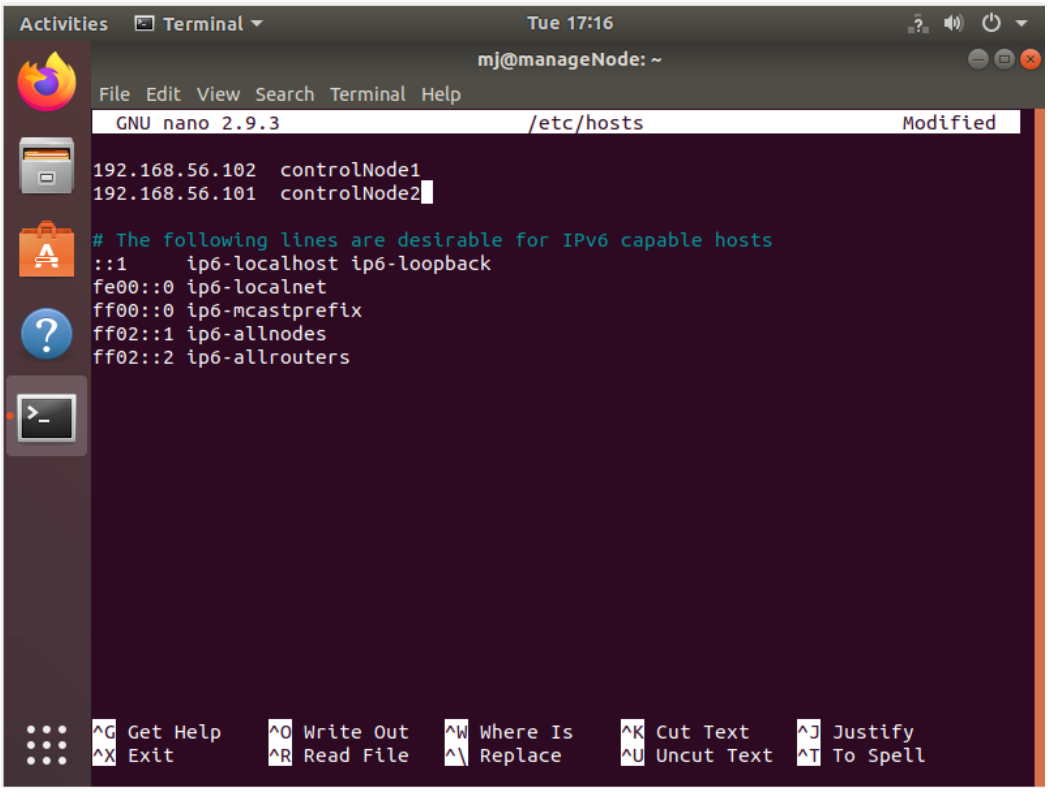
```
mj@controlNode1:~$ logout
Connection to 192.168.56.102 closed.
mj@manageNode:~$
```

3. Do the same for Server 2.

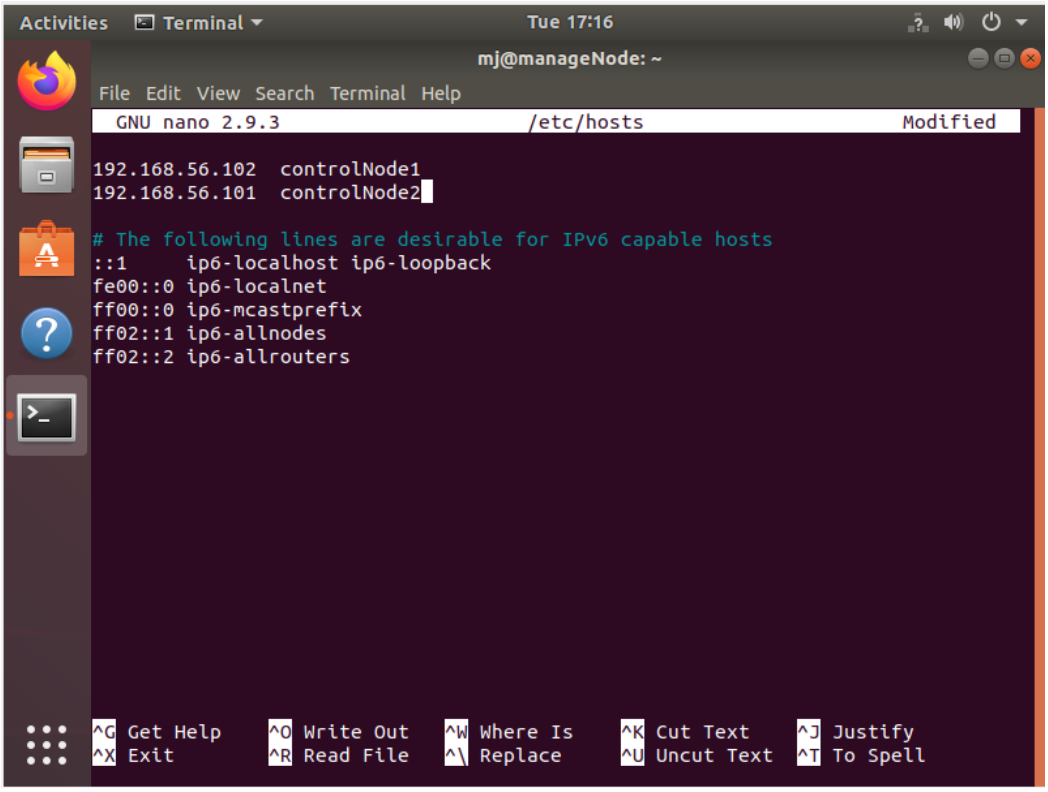


4. Edit the hosts of the Local Machine by issuing the command *sudo nano /etc/hosts.* Below all texts type the following:

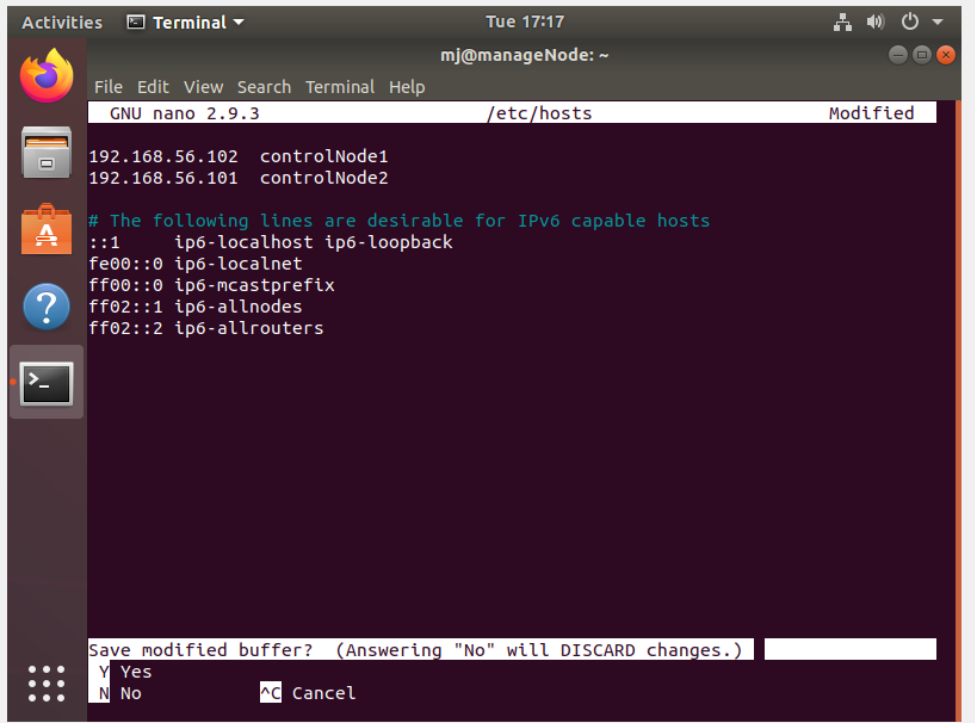
4.1 IP_address server 1 (provide the ip address of server 1 followed by the hostname)



4.2 IP_address server 2 (provide the ip address of server 2 followed by the hostname)



4.3 Save the file and exit.

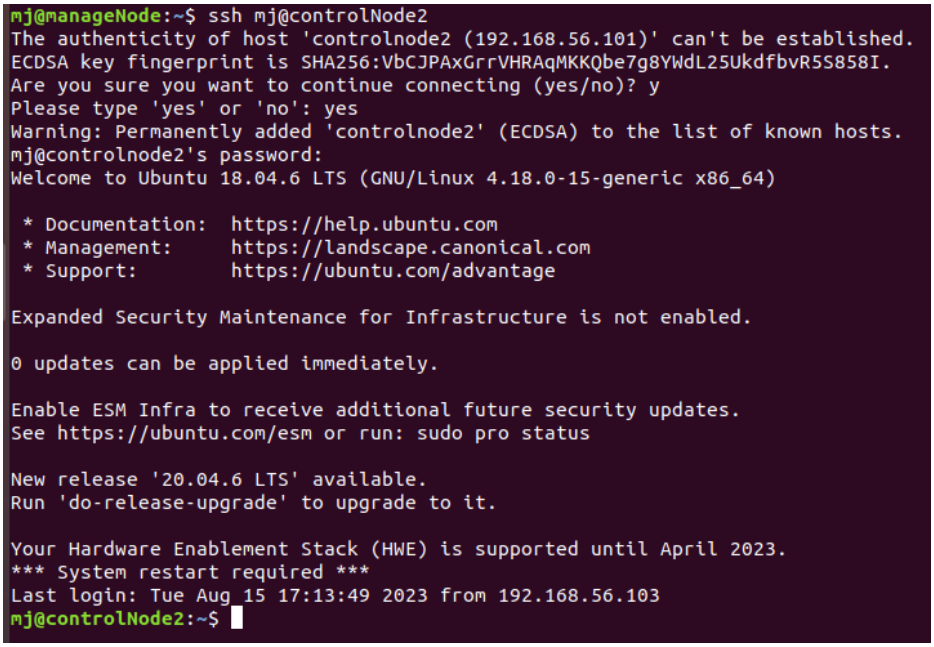


5. On the local machine, verify that you can do the SSH command but this time, use the hostname instead of typing the IP address of the servers. For example, try to do *ssh jvtaylar@server1*. Enter the password when prompted. Verify that you have entered Server 1. Do the same for Server 2.

   Server 1:

```
mj@manageNode:~$ ssh mj@controlNode1
The authenticity of host 'controlnode1 (192.168.56.102)' can't be established.
ECDSA key fingerprint is SHA256:ZpoI4S5Y3kD3NmRXgvvz+sHLatLsNTEF2mNhLxORzdQ.
Are you sure you want to continue connecting (yes/no)? y
Please type 'yes' or 'no': yes
Warning: Permanently added 'controlnode1' (ECDSA) to the list of known hosts.
mj@controlnode1's password:
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 4.18.0-15-generic x86_64)

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

Expanded Security Maintenance for Infrastructure is not enabled.

0 updates can be applied immediately.

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

New release '20.04.6 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Your Hardware Enablement Stack (HWE) is supported until April 2023.
*** System restart required ***
Last login: Tue Aug 15 17:11:43 2023 from 192.168.56.103
mj@controlNode1:~$
```

Server 2:



Reflections:

Answer the following:

1. How are we able to use the hostname instead of IP address in SSH commands?

   A procedure known as "hostname resolution" or "name resolution" enables your system to convert human-readable hostnames into corresponding IP addresses, allowing you to utilize hostnames instead of IP addresses in SSH commands. As we have used the “sudo nano /etc/hosts” command, we can change to map hostnames to IP addresses manually. In that case, we can use either the hostname OR the IP address, it will both work.

2. How secured is SSH?

   Secure Shell or Secure Socket Shell (SSH) is a strong authentication and encryption method, allowing for safe remote access and communication, because it provides strong encryption and authentication protocols, it is regarded as one of the most well-liked and secure methods of accessing and managing distant computers. As shown and demonstrated in the activity, SSH is a great tool for administrators for securing networks, and ease of access to other devices remotely.

Conclusion:

In this activity, we were taught how to clone and configure a Linux Operating System using a Virtual Machine and we were able to manipulate files, control remotely, configure, and verify SSH.

We were able to understand how each of the network of each virtualboxes behave and we were able to test their connection through commands that were taught in this activity.

In conclusion, we were able to attain the objectives of this activity, which is to create and configure virtual machines in virtualbox, and set-up a virtual network and test connectivity of each virtualboxes.