Name: Ruud Van G. Apostol	**Date Performed:** 08/08/25
Course/Section: CPE212 / CPE31S4	**Date Submitted:** 08/08/25
Instructor: Engr. Robin Valenzuela	**Semester and SY:** 1stSem. / 2025-2026

Activity 1: Configure Network using Virtual Machines

1. **Objectives:**

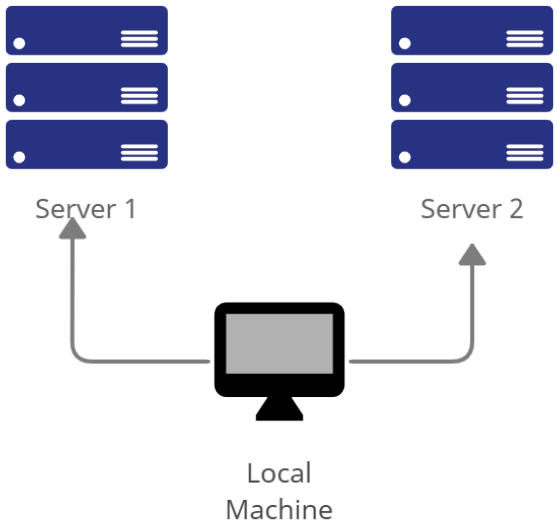
1.1. Create and configure Virtual Machines in Microsoft Azure or VirtualBox

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

2. **Discussion:**

Network Topology:

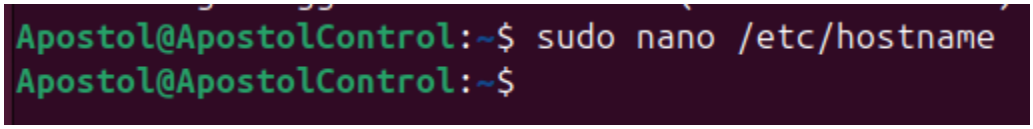
Assume that you have created the following network topology in Virtual Machines, *provide screenshots for each task*. (Note: *it is assumed that you have the prior knowledge of cloning and creating snapshots in a virtual machine*).

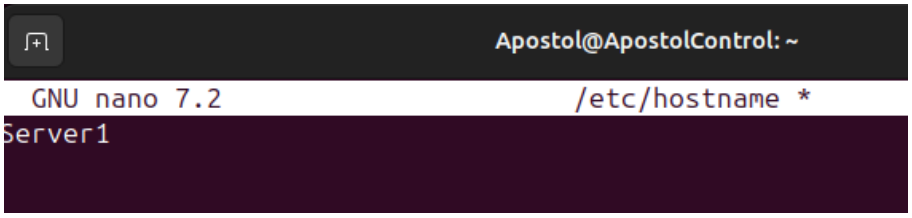


Task 1: Do the following on Server 1, Server 2, and Local Machine. In editing the file using nano command, press control + O to write out (save the file). Press enter when asked for the name of the file. Press control + X to end.

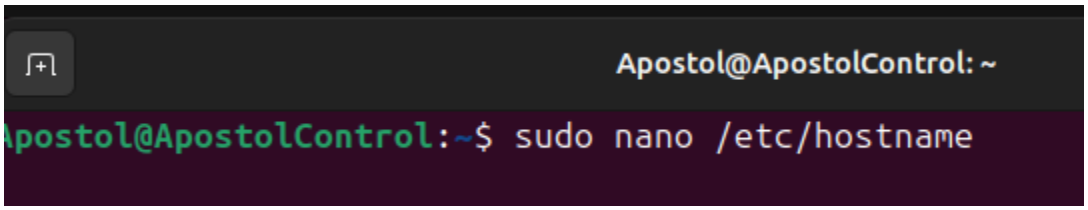
1. Change the hostname using the command *sudo nano /etc/hostname*

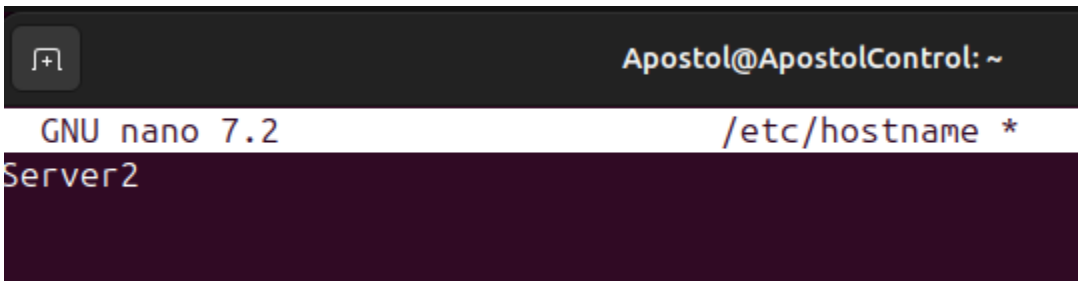
1.1 Use server1 for Server 1



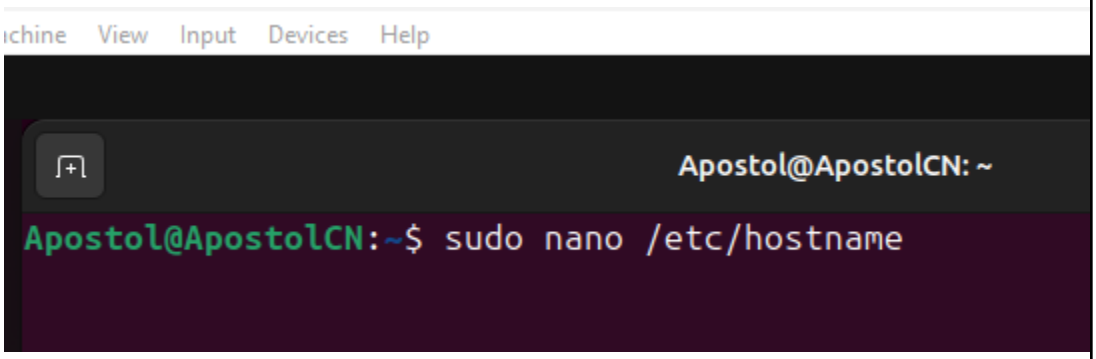


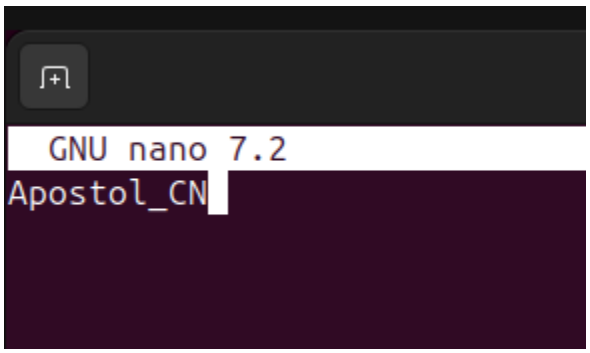
1.2 Use server2 for Server 2



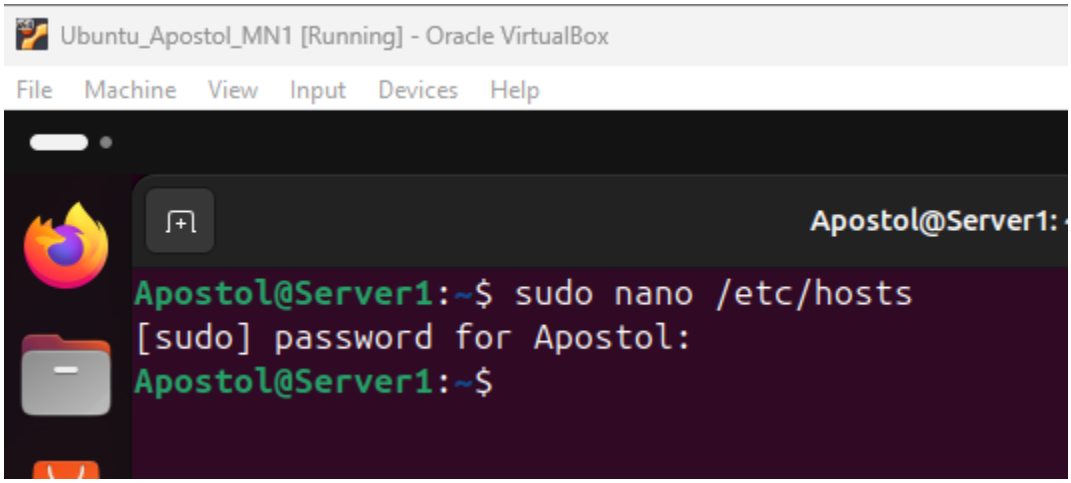


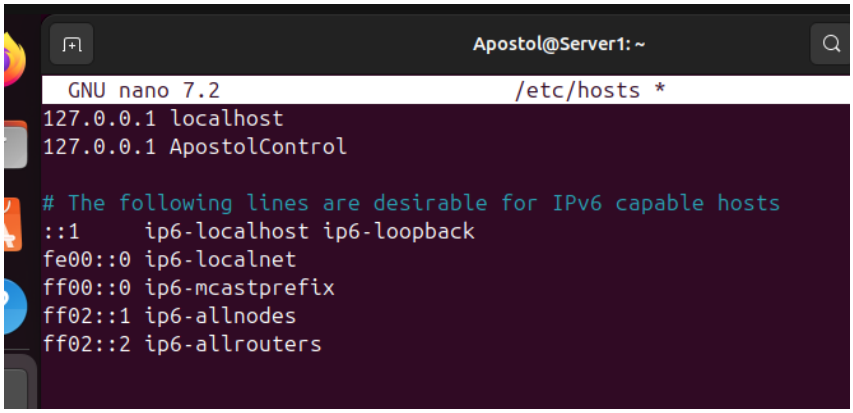
1.3 Use workstation for the Local Machine



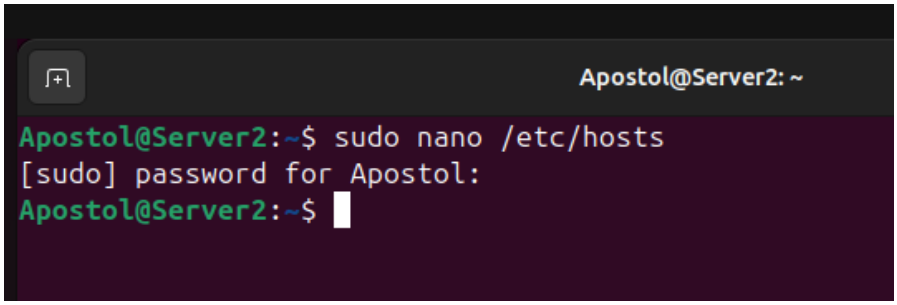


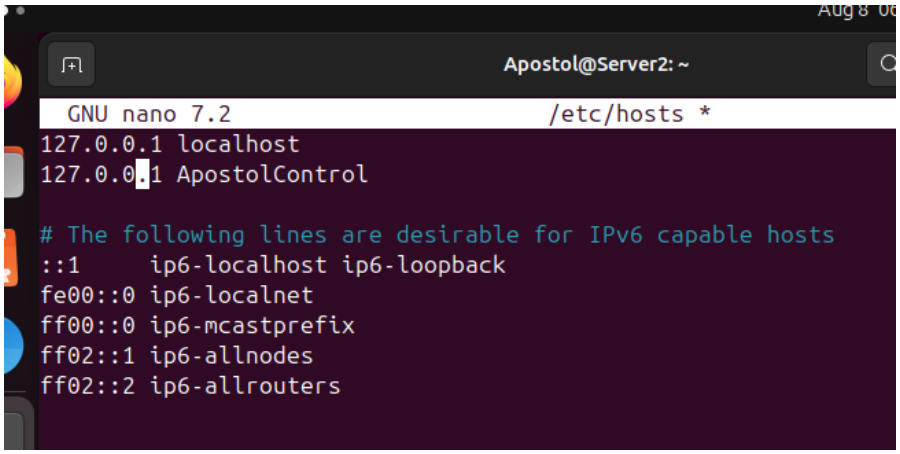
2. Edit the hosts using the command *sudo nano /etc/hosts*. Edit the second line.
   2.1 Type 127.0.0.1 server 1 for Server 1



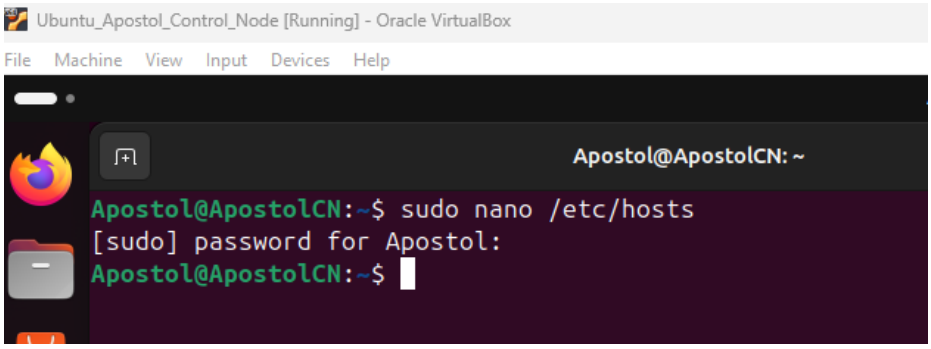


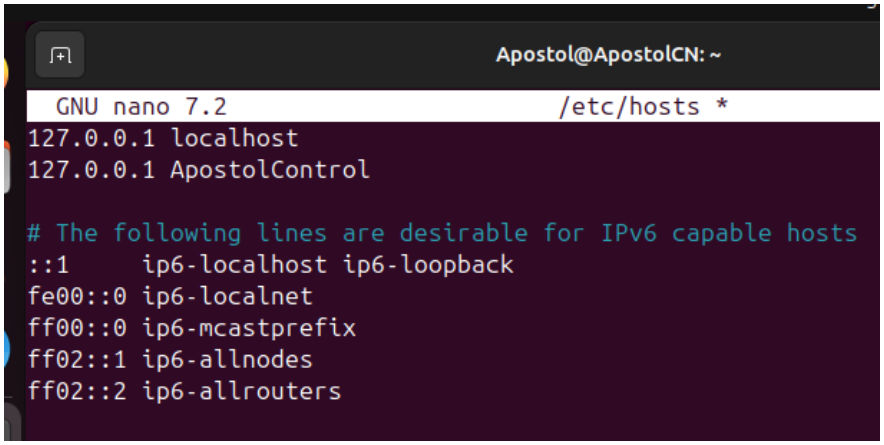
   2.2 Type 127.0.0.1 server 2 for Server 2





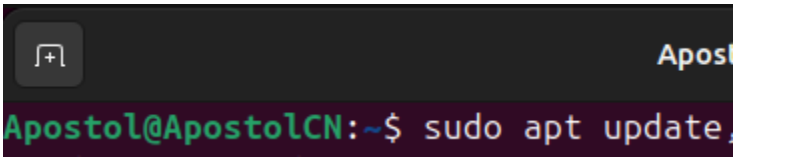
2.3 Type 127.0.0.1 workstation for the Local Machine

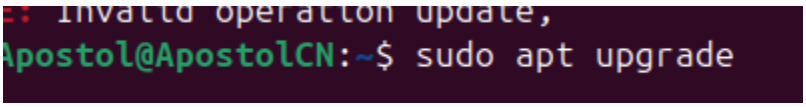




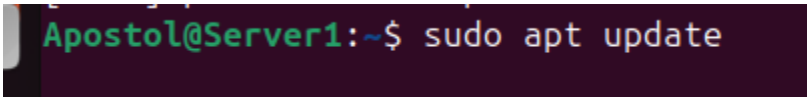
Task 2: Configure SSH on Server 1, Server 2, and Local Machine. Do the following:

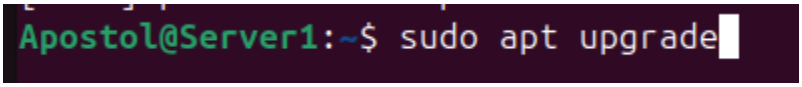
1. Upgrade the packages by issuing the command *sudo apt update* and *sudo apt upgrade* respectively.



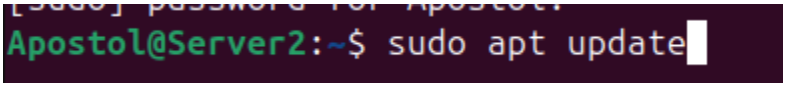


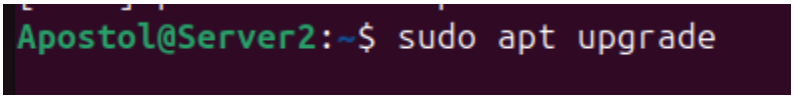
Server 1:





Server 2:





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

Server 1:

```
Apostol@Server1:~$ sudo apt install openssh-server
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libgl1-amber-dri libglapi-mesa
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  ncurses-term openssh-sftp-server ssh-import-id
```

Server 2:

```
Apostol@Server2:~$ sudo apt install openssh-server
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are n
  libgl1-amber-dri libglapi-mesa
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
```

Control_Node:

```
Apostol@ApostolCN:~$ sudo apt install openssh-server
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no l
  libgl1-amber-dri libglapi-mesa
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  ncurses-term openssh-sftp-server ssh-import-id
Suggested packages:
```

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

3.1 *sudo service ssh start*

Server 1:

```
Apostol@Server1:~$ sudo service ssh start
Apostol@Server1:~$
```

Server 2:

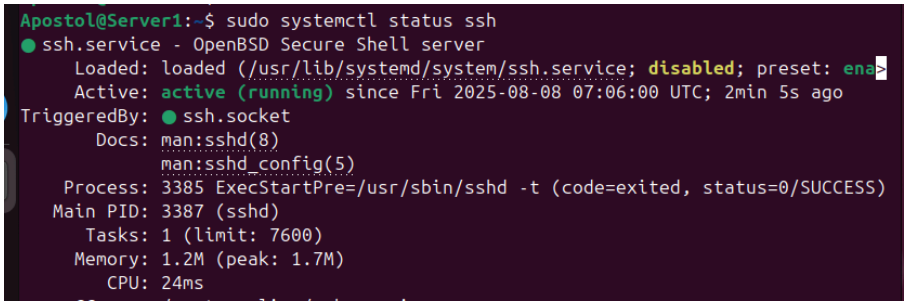
```
Apostol@Server2:~$ sudo service ssh start
Apostol@Server2:~$
```

Control_Node:

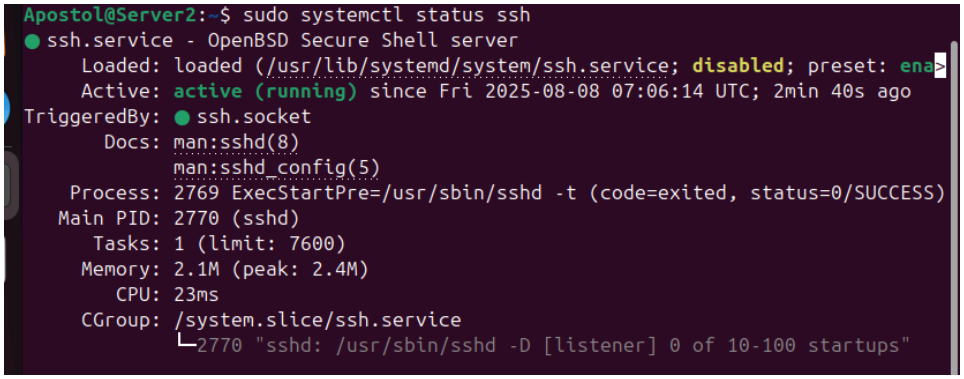
```
Apostol@ApostolCN:~$ sudo service ssh start
Apostol@ApostolCN:~$
```

3.2 *sudo systemctl status ssh*

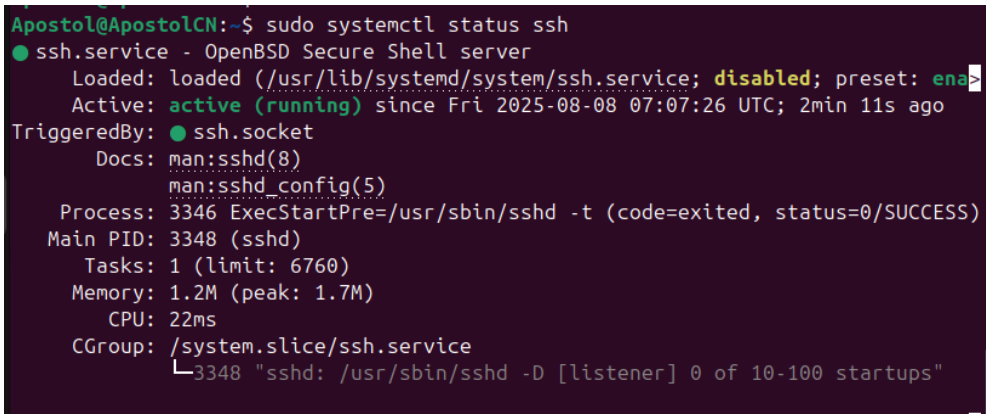
Server 1:



Server 2:



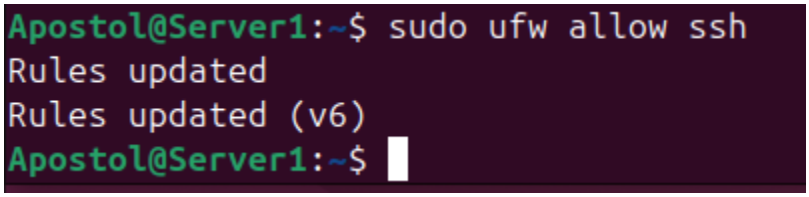
Control_Node:



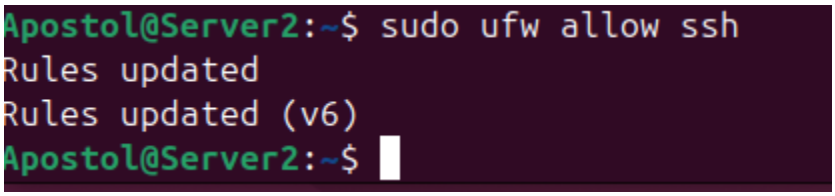
4. Configure the firewall to all port 22 by issuing the following commands:

4.1 *sudo ufw allow ssh*

Server 1:



Server 2:



Control Node:

```
Apostol@ApostolCN:~$ sudo ufw allow ssh
Rules updated
Rules updated (v6)
Apostol@ApostolCN:~$
```

4.2 *sudo ufw enable*

Server 1:

```
Apostol@Server1:~$ sudo ufw enable
Firewall is active and enabled on system startup
Apostol@Server1:~$
```

Server 2:

```
Apostol@Server2:~$ sudo ufw enable
Firewall is active and enabled on system startup
Apostol@Server2:~$
```

Control Node:

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

4.3 *sudo ufw status*

Server 1:

```
Apostol@Server1:~$ sudo ufw status
Status: active

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

Server 2:

```
Apostol@Server2:~$ sudo ufw status
Status: active

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

Control Node:

```
Apostol@ApostolCN:~$ 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 Server 1 IP address: 192.168.56.**106**

```
enp0s8: flags=4163<UP,BROADC
        inet 192.168.56.106
```

   1.2 Server 2 IP address: 192.168.56.**108**

```
enp0s8: flags=4163<UP,BROADCA
        inet 192.168.56.108
```

   1.3 Server 3 IP address: 192.168.56.**104**

```
enp0s8: flags=4163<UP,BROADC
        inet 192.168.56.104
```

2. Make sure that they can ping each other.
   2.1 Connectivity test for Local Machine 1 to Server 1: **X** Successful ☐ Not Successful

```
Apostol@ApostolCN:~$ ping 192.168.56.106
PING 192.168.56.106 (192.168.56.106) 56(84) bytes of data.
64 bytes from 192.168.56.106: icmp_seq=1 ttl=64 time=0.953 ms
64 bytes from 192.168.56.106: icmp_seq=2 ttl=64 time=0.437 ms
64 bytes from 192.168.56.106: icmp_seq=3 ttl=64 time=0.397 ms
64 bytes from 192.168.56.106: icmp_seq=4 ttl=64 time=0.504 ms
64 bytes from 192.168.56.106: icmp_seq=5 ttl=64 time=0.467 ms
64 bytes from 192.168.56.106: icmp_seq=6 ttl=64 time=0.569 ms
64 bytes from 192.168.56.106: icmp_seq=7 ttl=64 time=0.416 ms
64 bytes from 192.168.56.106: icmp_seq=8 ttl=64 time=0.523 ms
64 bytes from 192.168.56.106: icmp_seq=9 ttl=64 time=0.438 ms
64 bytes from 192.168.56.106: icmp_seq=10 ttl=64 time=0.695 ms
64 bytes from 192.168.56.106: icmp_seq=11 ttl=64 time=0.645 ms
```

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

```
Apostol@ApostolCN:~$ ping 192.168.56.108
PING 192.168.56.108 (192.168.56.108) 56(84) bytes of data.
64 bytes from 192.168.56.108: icmp_seq=1 ttl=64 time=1.19 ms
64 bytes from 192.168.56.108: icmp_seq=2 ttl=64 time=0.485 ms
64 bytes from 192.168.56.108: icmp_seq=3 ttl=64 time=0.488 ms
64 bytes from 192.168.56.108: icmp_seq=4 ttl=64 time=0.464 ms
64 bytes from 192.168.56.108: icmp_seq=5 ttl=64 time=0.446 ms
^Z
```

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

```
Apostol@Server1:~$ ping 192.168.56.104
PING 192.168.56.104 (192.168.56.104) 56(84) bytes of data
64 bytes from 192.168.56.104: icmp_seq=1 ttl=64 time=0.62
64 bytes from 192.168.56.104: icmp_seq=2 ttl=64 time=0.42
64 bytes from 192.168.56.104: icmp_seq=3 ttl=64 time=0.62
64 bytes from 192.168.56.104: icmp_seq=4 ttl=64 time=0.55
^Z
```

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*

```
Apostol@ApostolCN:~$ ssh Apostol@192.168.56.106
The authenticity of host '192.168.56.106 (192.168.56.106)' can't be established.
ED25519 key fingerprint is SHA256:YyUxOlmQGK2q61G390Hm5n5DV4Xii//RMURmhORRtYY.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.56.106' (ED25519) to the list of known hosts
.
```

1.2 Enter the password for server 1 when prompted

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

```
Apostol@192.168.56.106's password:
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-27-generic x86_64)

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

Expanded Security Maintenance for Applications is not enabled.
```

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

```
Connection to 192.168.56.106 closed.
Apostol@ApostolCN:~$
```

3. Do the same for Server 2.

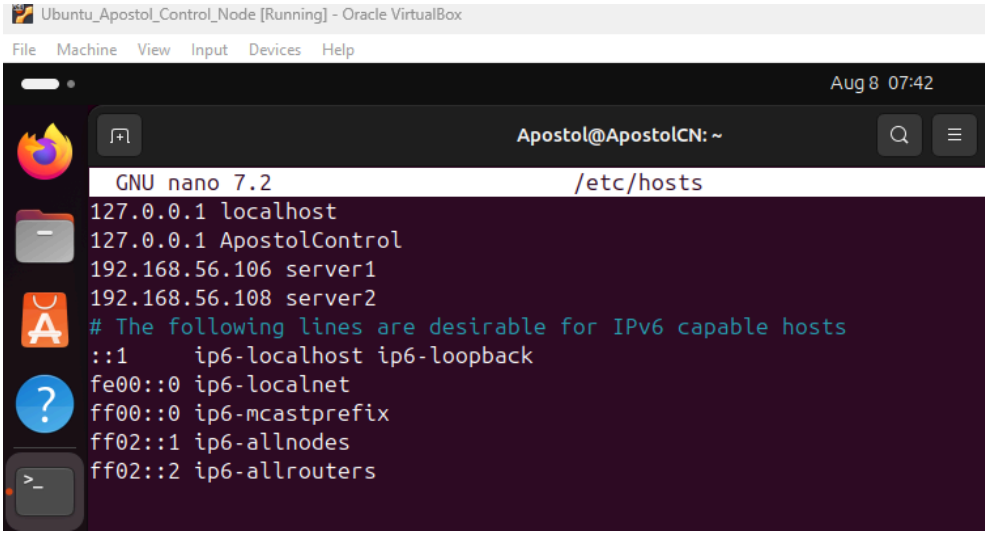
```
Apostol@ApostolCN:~$ ssh Apostol@192.168.56.108
The authenticity of host '192.168.56.108 (192.168.56.108)' can't be establis
ED25519 key fingerprint is SHA256:YyUxOlmQGK2q61G390Hm5n5DV4Xii//RMURmhORRtY
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:1: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.56.108' (ED25519) to the list of known h
.
Apostol@192.168.56.108's password:
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-27-generic x86_64)

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

Expanded Security Maintenance for Applications is not enabled.
```

```
Apostol@Server2:~$
logout
Connection to 192.168.56.108 closed.
Apostol@ApostolCN:~$
```

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

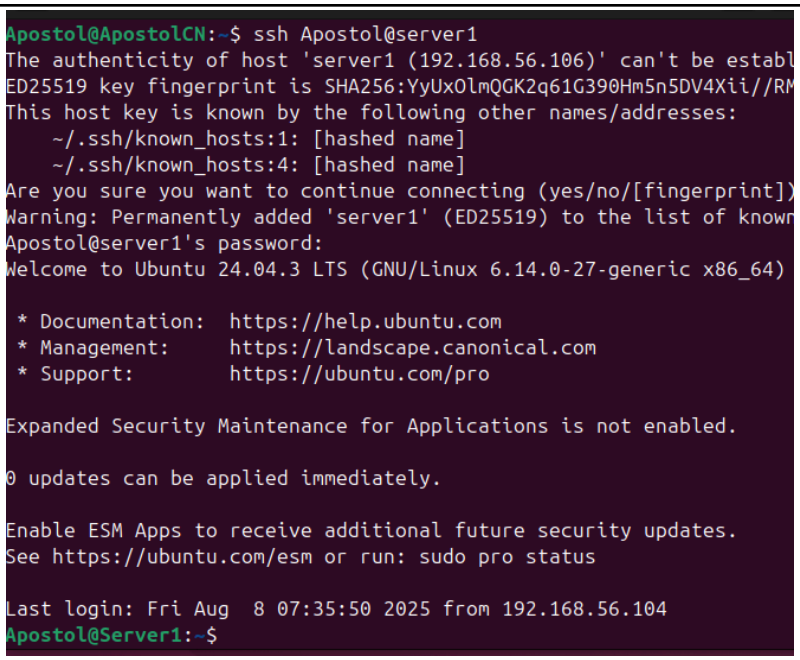
4.1 IP_address server 1 (provide the ip address of server 1 followed by the hostname)

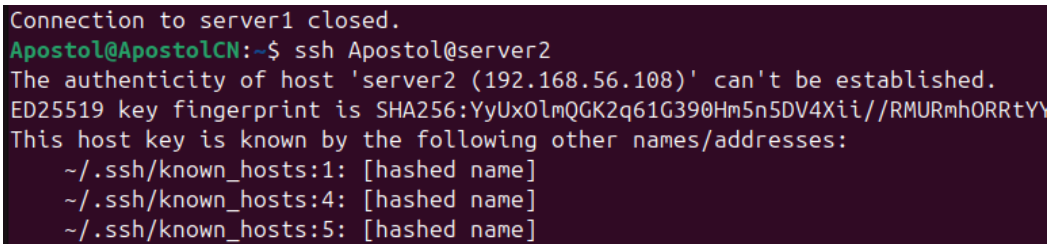
4.2 IP_address server 2 (provide the ip address of server 2 followed by the hostname)

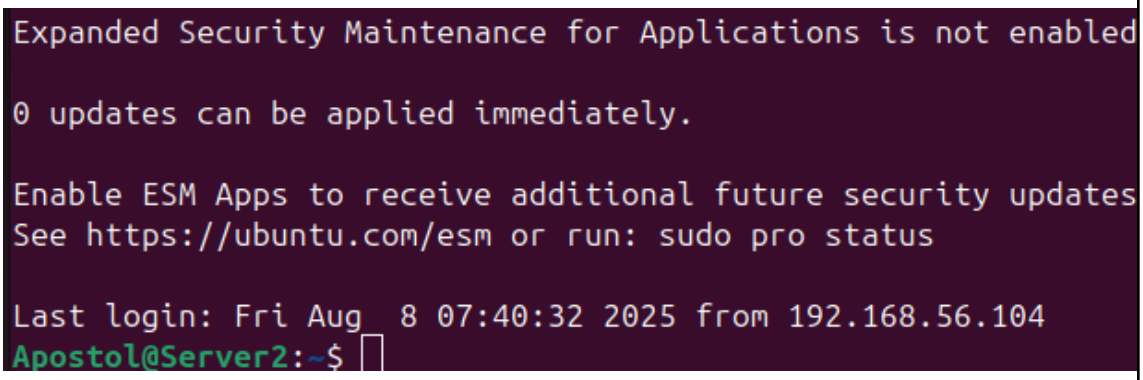


4.3 Save the file and exit.

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.







Reflections:

Answer the following:

1. How are we able to use the hostname instead of IP address in SSH commands?
   - **Because I included the two nodes' IP addresses within the script, it redirected us to server 1 when we called the hostname.**
2. How secured is SSH?
   - **SSH is "secure" because it incorporates encryption and authentication via a process called public key cryptography. Public key cryptography is a way to encrypt data, or sign data, with two different keys. One of the keys, the public key, is available for anyone to use.**