Name: Seruelas, Ronn Kristoper H.	**Date Performed:** 08/14/23
Course/Section: CPE31S4	**Date Submitted:** 08/15/23
Instructor: Dr. Jonathan V. Taylar	**Semester and SY:** 1st Sem. 2023-2024

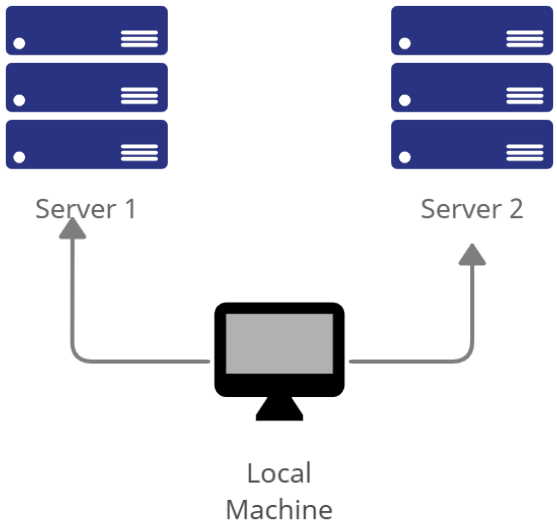
Activity 1: Configure Network using Virtual Machines

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*).



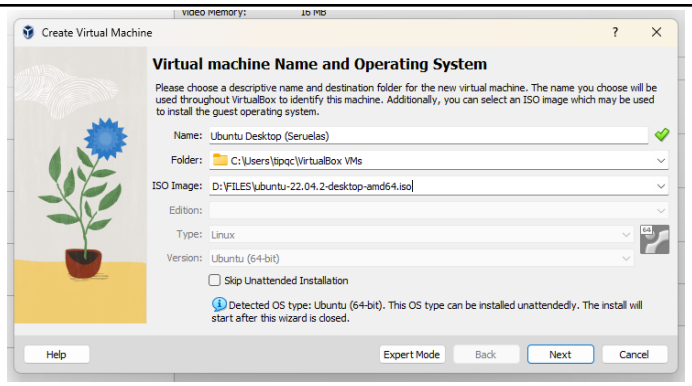
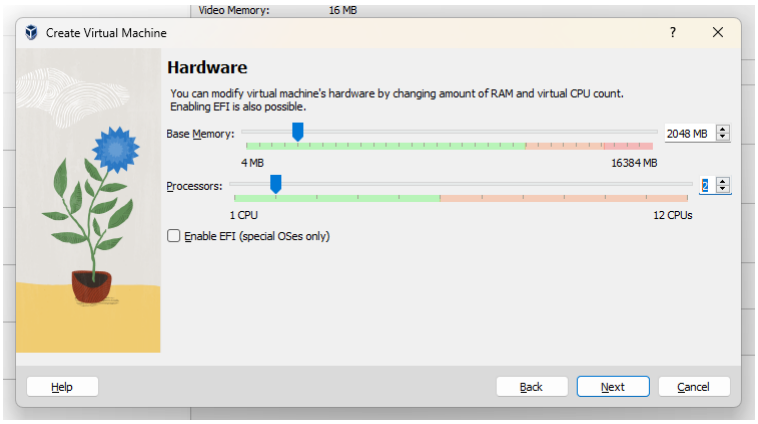


Figure 1.1 - Creating a Virtual Machine for Ubuntu Linux.



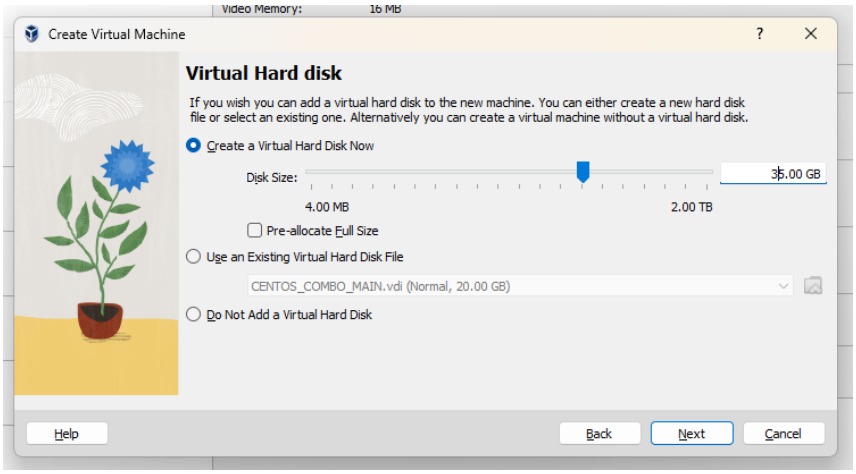


Figure 1.2-1.3 - Allocating the resources for the Virtual Machine

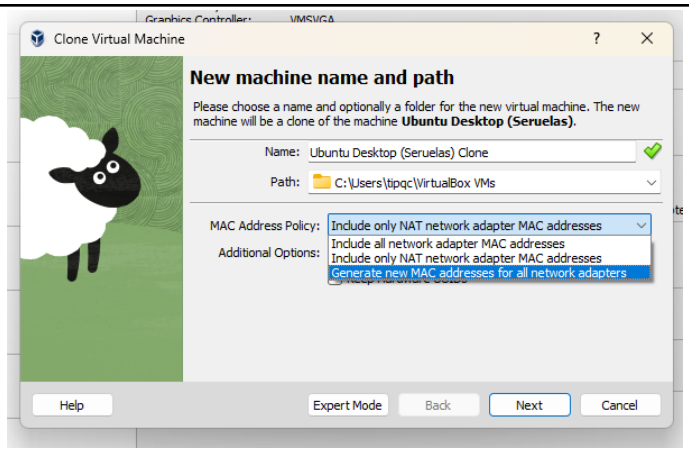


Figure 1.4 - Cloning the Virtual Machine for 2 servers.

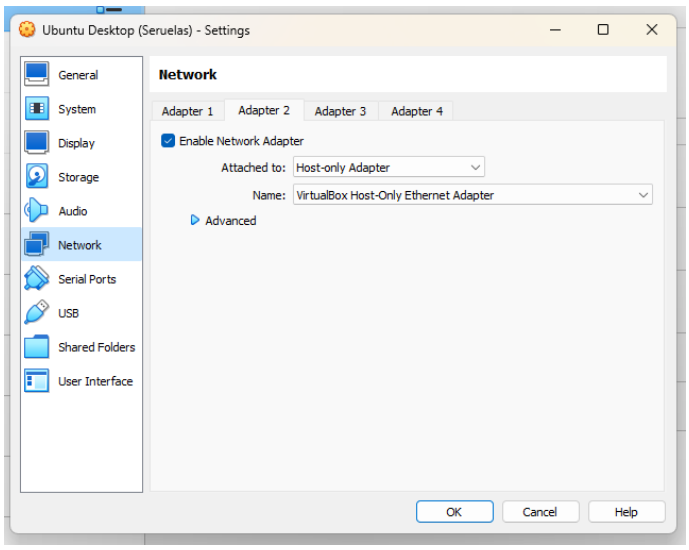


Figure 1.5 - Configuring Network Adapter #2 Settings to **Host-Only Adapter, VirtualBox Host-Only Ethernet Adapter #2.**

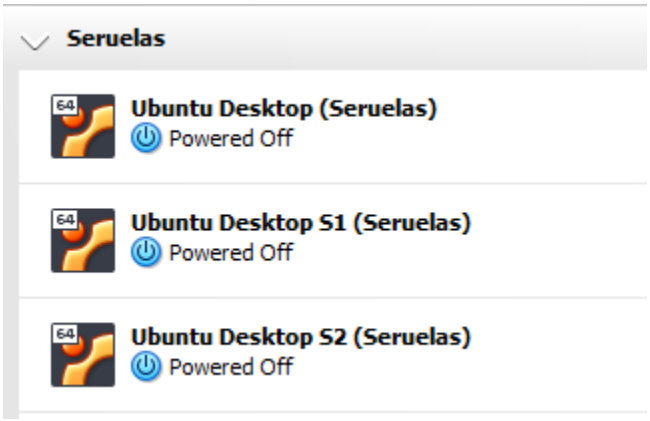
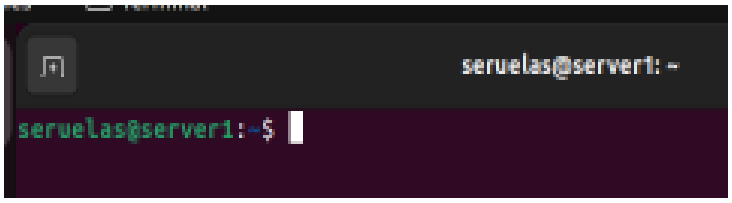


Figure 1.6 - Grouping all Virtual Machines together.

Task 2: 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

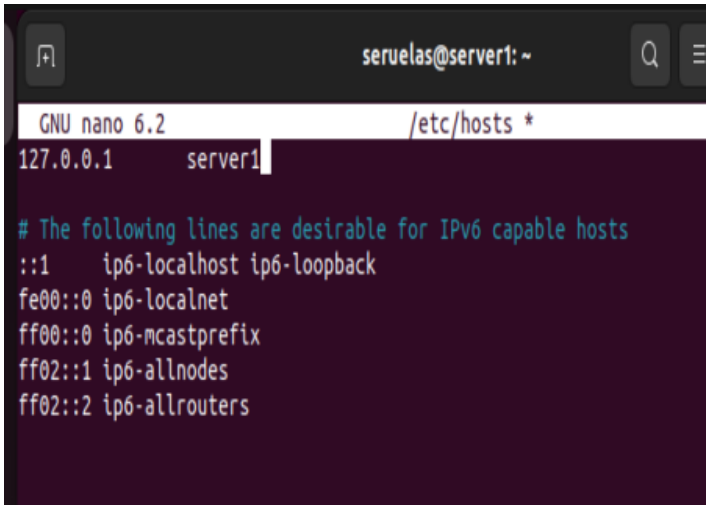


   1.2 Use server2 for Server 2

   1.3 Use workstation for the Local Machine

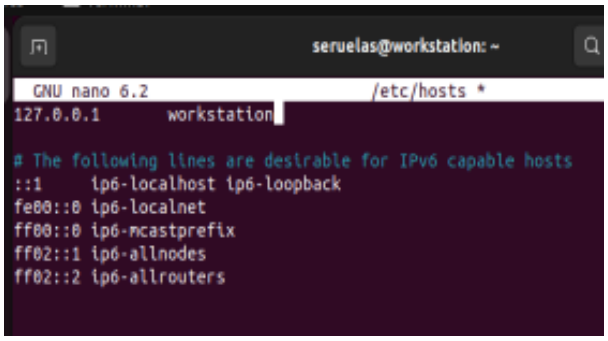
2. Edit the hosts using the command *sudo nano /etc/hosts*. Edit the second line.

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 3: 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.

```
seruelas@seruelas-ManageNode:~$ sudo apt update | sudo apt upgrade
[sudo] password for seruelas:

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
```

Figure 3.1 - Updating/Upgrading the machines prior cloning for more efficiency.

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

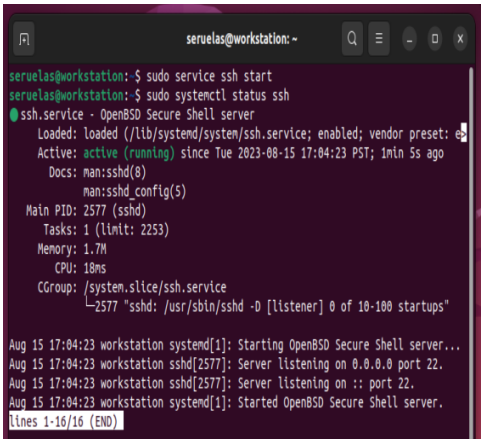
```
seruelas@workstation: ~
seruelas@workstation:~$ sudo apt install openssh-server
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  ncurses-term openssh-sftp-server ssh-import-id
Suggested packages:
  molly-guard monkeysphere 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 2 not upgraded.
Need to get 751 kB of archives.
After this operation, 6,046 kB of additional disk space will be used.
```

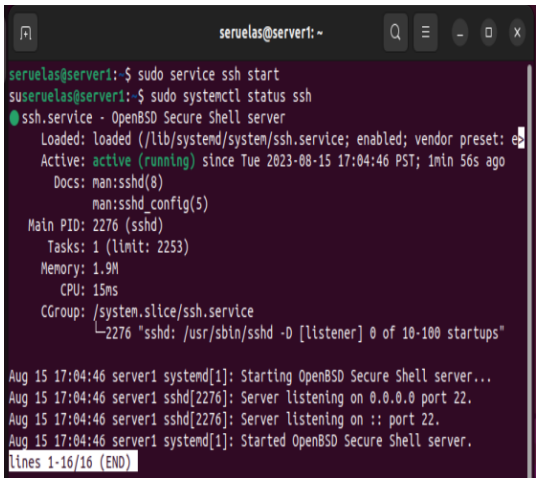
```
seruelas@server1: ~
seruelas@server1:~$ sudo apt install openssh-server
[sudo] password for seruelas:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  ncurses-term openssh-sftp-server ssh-import-id
Suggested packages:
  molly-guard monkeysphere 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 2 not upgraded.
Need to get 751 kB of archives.
After this operation, 6,046 kB of additional disk space will be used.
```

```
seruelas@server2: ~
seruelas@server2:~$ sudo apt install openssh-server
[sudo] password for seruelas:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  ncurses-term openssh-sftp-server ssh-import-id
Suggested packages:
  molly-guard monkeysphere 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 2 not upgraded.
Need to get 751 kB of archives.
After this operation, 6,046 kB of additional disk space will be used.
```

Figure 3.2.1-3.2.3 - Installation of OpenSSH in Workstation, Server1, and Server2.

3. Verify if the SSH service has started by issuing the following commands:
   3.1 *sudo service ssh start*
   3.2 *sudo systemctl status ssh*





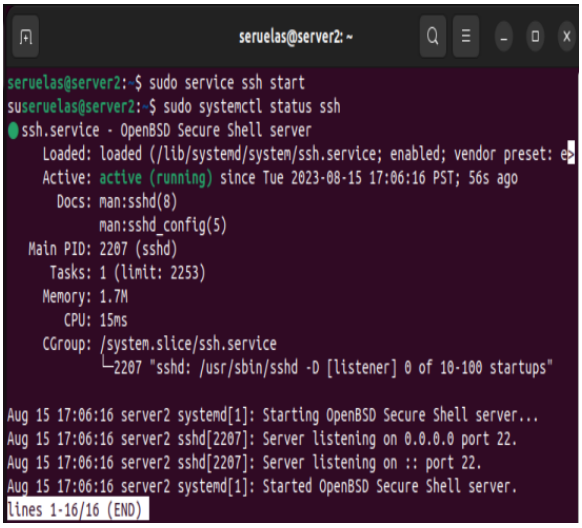


Figure 3.3.1-3.3.3 - Verifying the SSH Services done by executing the commands.

4. Configure the firewall to all port 22 by issuing the following commands:
   4.1 *sudo ufw allow ssh*
   4.2 *sudo ufw enable*
   4.3 *sudo ufw status*

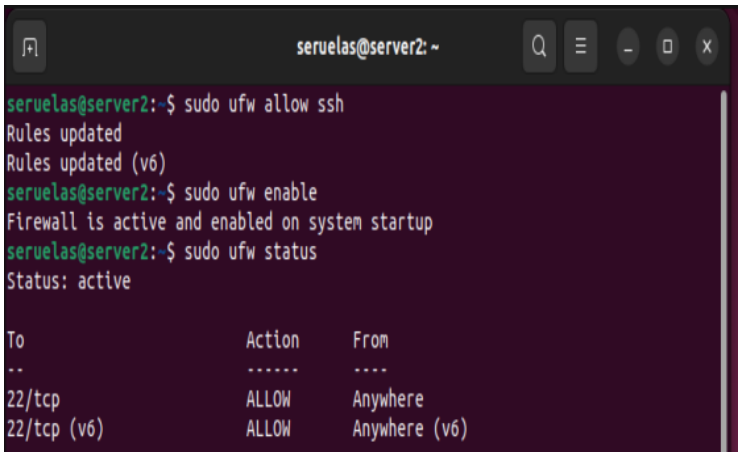


Figure 3.4.1-3.4.3 - Configuring the firewall to all port 22.

Task 4: 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.**112**
   1.2 Server 2 IP address: 192.168.56.**113**
   1.3 Local Machine IP address: 192.168.56.**111**

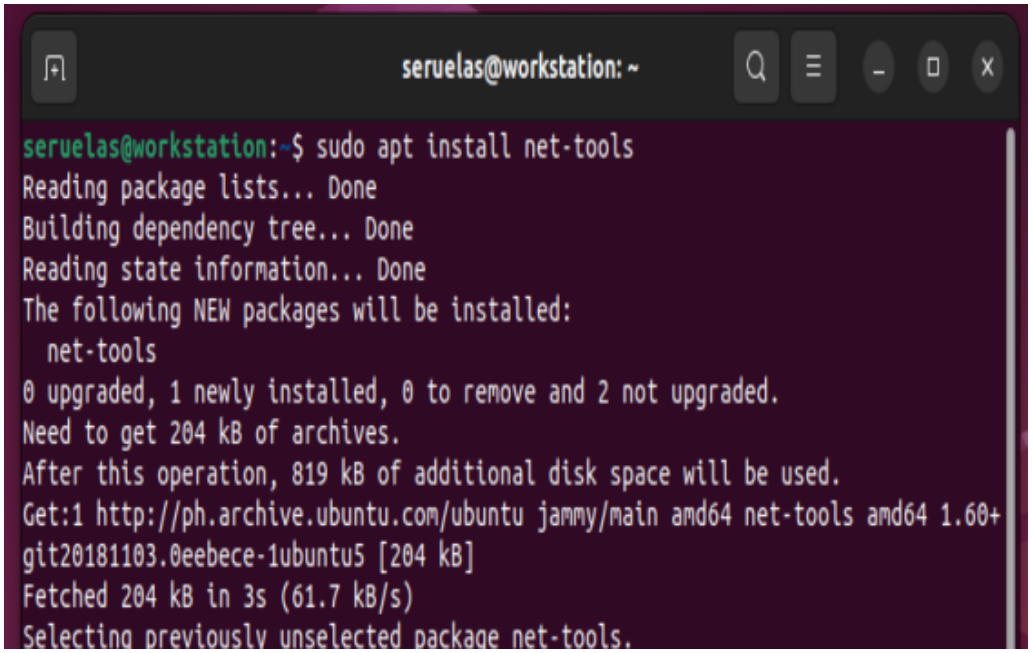


Figure 4.1.1 - Installation of package **net-tools** for the command **ifconfig**.

```
seruelas@server1: ~
seruelas@server1:~$ 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::c3d8:b813:d4eb:e906  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:b9:b7:08  txqueuelen 1000  (Ethernet)
        RX packets 28  bytes 5037 (5.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 102  bytes 11401 (11.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.56.112  netmask 255.255.255.0  broadcast 192.168.56.255
        inet6 fe80::e7dc:5a29:568d:2117  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:e6:06:99  txqueuelen 1000  (Ethernet)
        RX packets 110  bytes 14179 (14.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 55  bytes 7133 (7.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 153  bytes 15148 (15.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 153  bytes 15148 (15.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

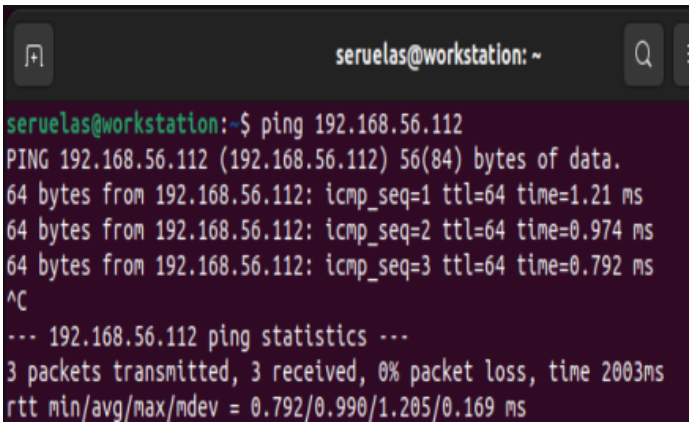
```
seruelas@server2: ~
seruelas@server2:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::b98e:bcb0:918e:3a4b  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:7d:ff:fa  txqueuelen 1000  (Ethernet)
        RX packets 32  bytes 5622 (5.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 106  bytes 11770 (11.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.56.113  netmask 255.255.255.0  broadcast 192.168.56.255
        inet6 fe80::975f:b019:8d4f:8f36  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:3d:38:30  txqueuelen 1000  (Ethernet)
        RX packets 68  bytes 9031 (9.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 68  bytes 8398 (8.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 147  bytes 14352 (14.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 147  bytes 14352 (14.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
seruelas@workstation: ~
seruelas@workstation:~$ 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::eb58:73c3:d313:15f1  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:5f:81:77  txqueuelen 1000  (Ethernet)
        RX packets 37  bytes 6360 (6.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 113  bytes 12372 (12.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.56.111  netmask 255.255.255.0  broadcast 192.168.56.255
        inet6 fe80::5dc0:8370:49c:eabc  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:ed:d2:b9  txqueuelen 1000  (Ethernet)
        RX packets 93  bytes 12700 (12.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 75  bytes 9027 (9.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 154  bytes 15254 (15.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 154  bytes 15254 (15.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

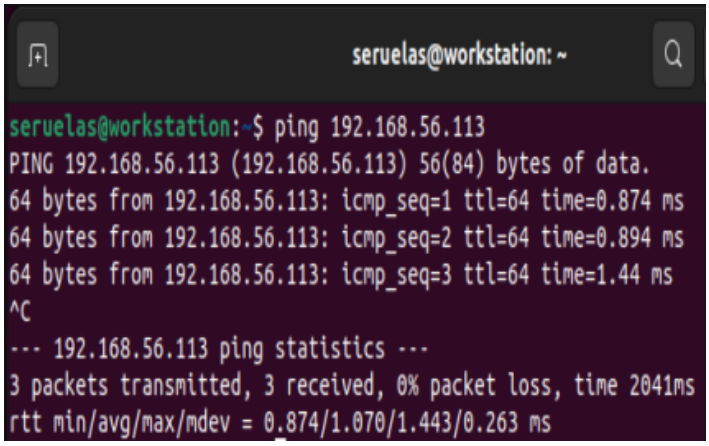
Figure 4.1.2-4.1.5 - Checking the IP Adresses of each machine.

2. Make sure that they can ping each other.

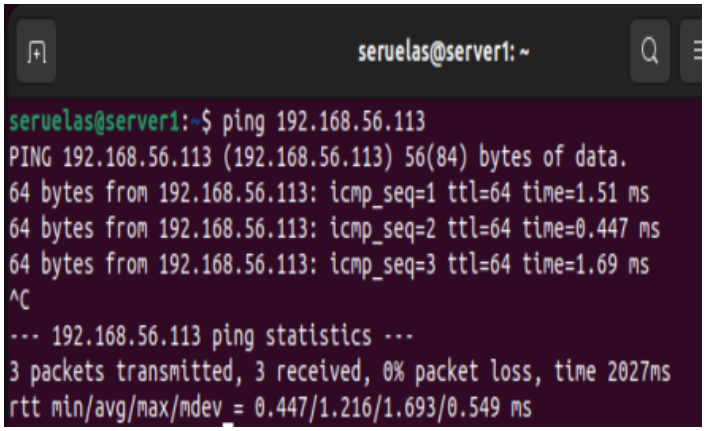
2.1 Connectivity test for Local Machine 1 to Server 1: ■ Successful ☐ Not Successful



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



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



Task 5: 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*

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*

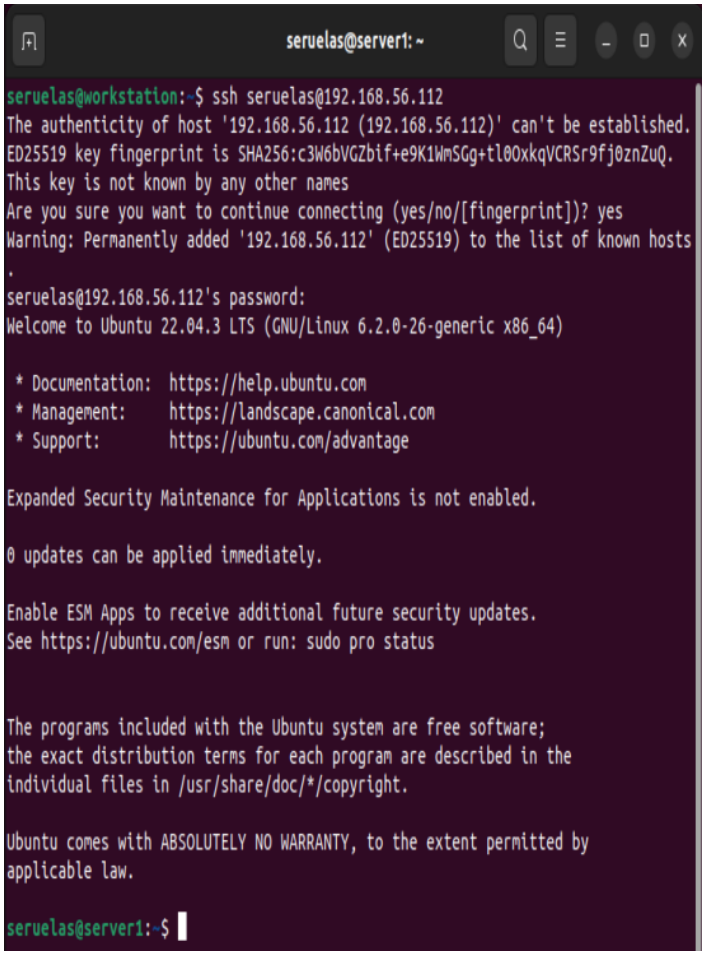


Figure 5.1.1 - Remote hosting the Server 1 via SSH Server.

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

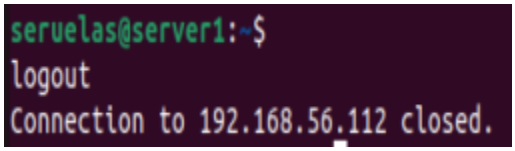


Figure 5.2.1 - Logging out of Server 1.

3. Do the same for Server 2.

```
seruelas@workstation: ~
seruelas@workstation:~$ ssh seruelas@192.168.56.113
The authenticity of host '192.168.56.113 (192.168.56.113)' can't be established.
ED25519 key fingerprint is SHA256:cNWsEKtjnqf3x+q3MgjSMafiAFEvHXunRC2J1y3oUK0.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.56.113' (ED25519) to the list of known hosts
.
seruelas@192.168.56.113's password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 6.2.0-26-generic x86_64)

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

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

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


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.

seruelas@server2:~$
logout
Connection to 192.168.56.113 closed.
```

Figure 5.3.1 - Remote hosting the Server 2 via SSH Server.

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.

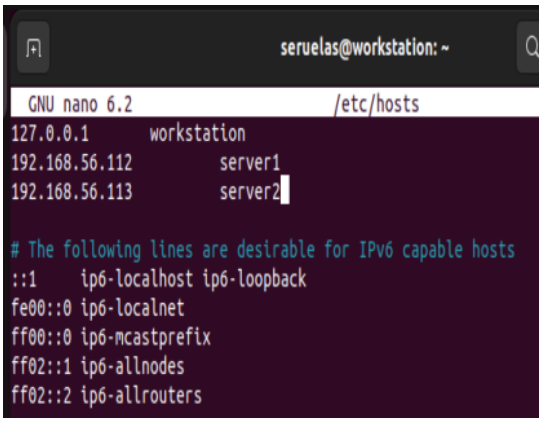


Figure 5.4.1 - Configuring the hosts by inputting their respective addresses.

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.

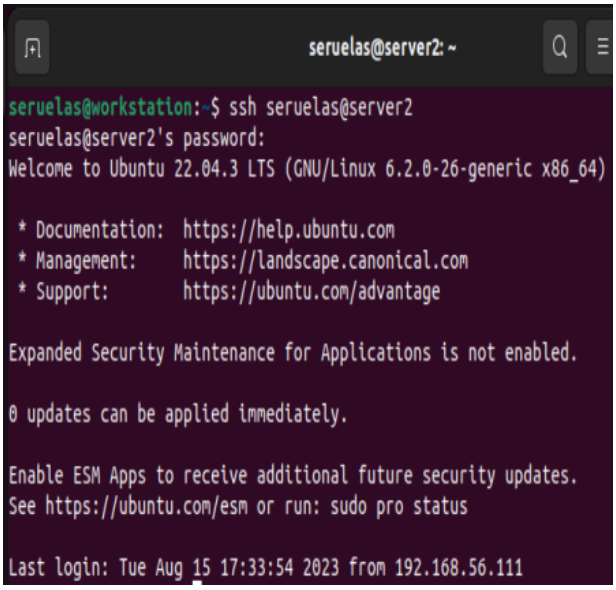


Figure 5.5.1-5.5.2 - Verifying the hosts via SSH Command by using their hostnames.

Reflections:

Answer the following:

1. How are we able to use the hostname instead of IP address in SSH commands?
   - It was made possible to use the hostnames of the IP addresses in the SSH commands by modifying or configuring the **/etc/hosts** file via **sudo nano**, by inputting the IP address of the server, then its hostname, allowing us to connect or remote host the server via their hostnames.

2. How secured is SSH?
   - SSH's security is more powerful and reliable as its security relies on encrypting its own traffic, using a public key that is used for authentication for its communications towards other computers.

"I affirm that I have not received or given any unauthorized help on this activity and that all work is my own."