Name: Jaira Biane Maculada	**Date Performed:** 08/17/23
Course/Section: CpE31S6	**Date Submitted:** 08/17/23
Instructor: Dr. Jonathan V. Taylar	**Semester and SY:** 1st Sem (2023-2024)

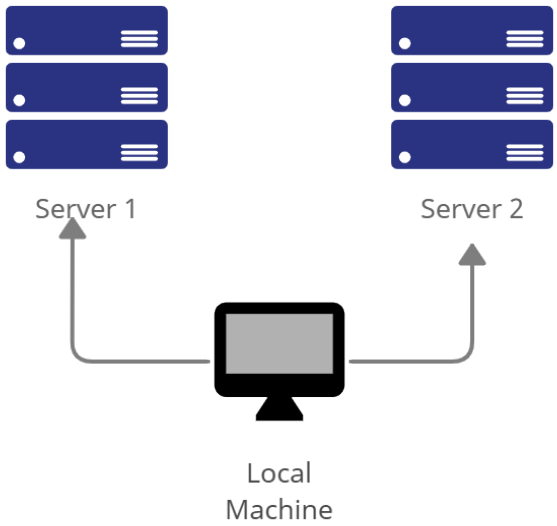
Activity 1: Configure Network using Virtual Machines

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

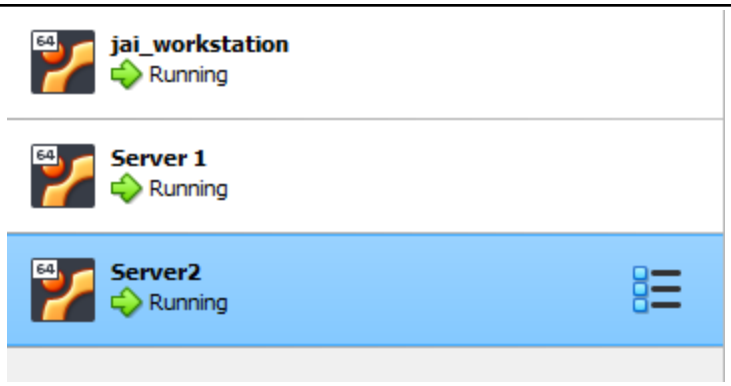
2. **Discussion:**

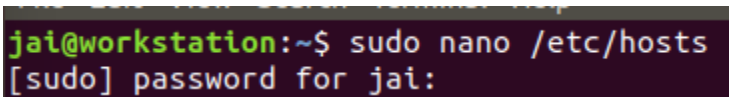
Network Topology:

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*).

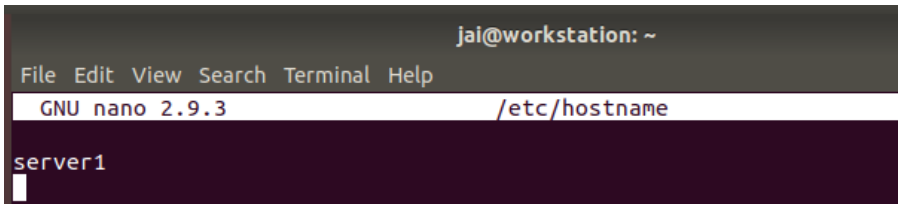


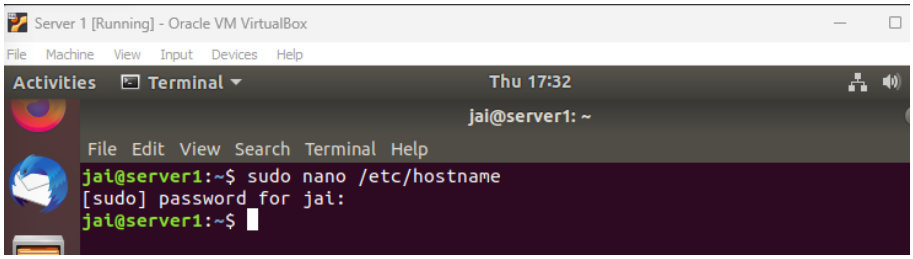
Task 1: Do the following on Server 1, Server 2, and Local Machine. In editing the file using nano command, press control + O to write out (save the file). Press enter when asked for the name of the file. Press control + X to end.



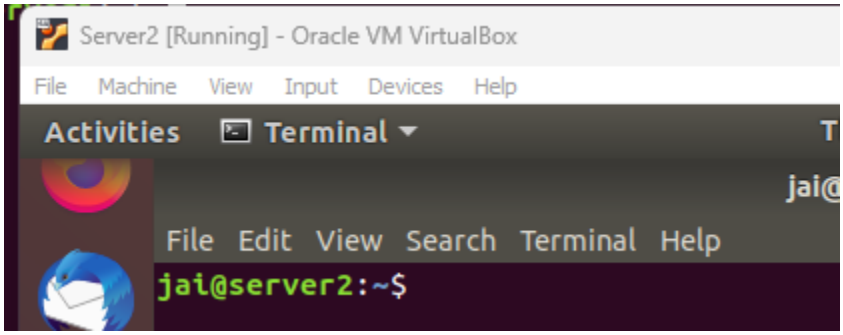


1. Change the hostname using the command *sudo nano /etc/hostname*
   1.1 Use server1 for Server 1



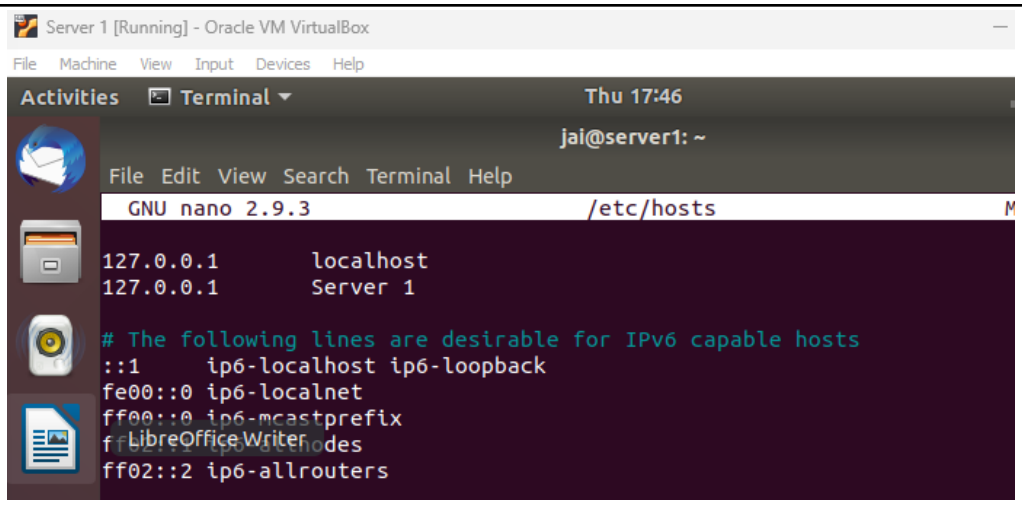


   1.2 Use server2 for Server 2

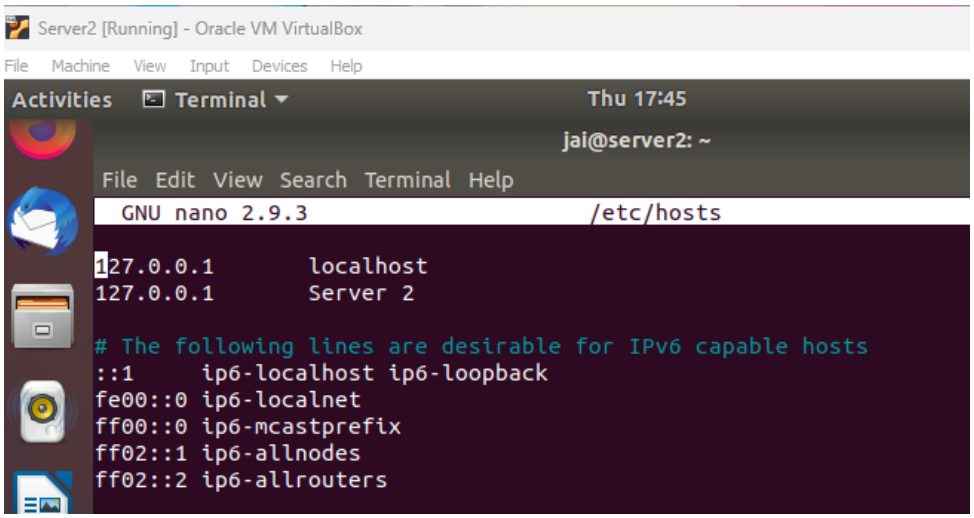


   1.3 Use workstation for the Local Machine

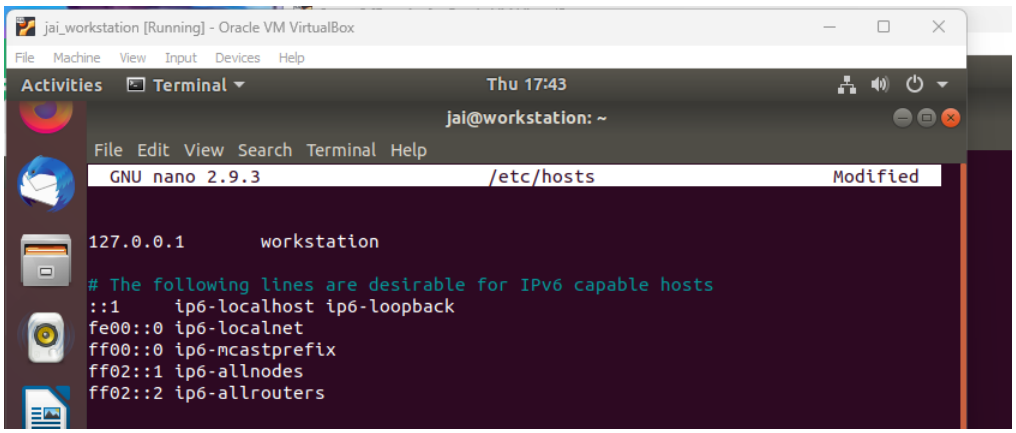
2. Edit the hosts using the command *sudo nano /etc/hosts.* Edit the second line.
   2.1 Type 127.0.0.1 server 1 for Server 1



2.2 Type 127.0.0.1 server 2 for Server 2

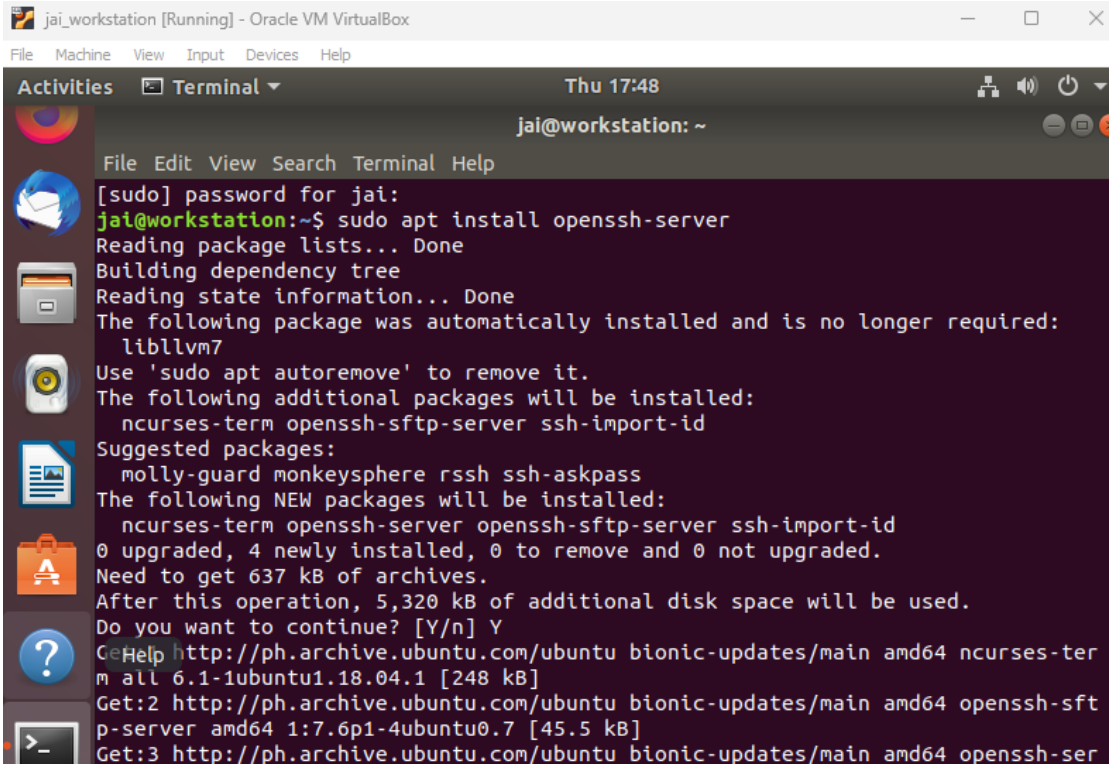


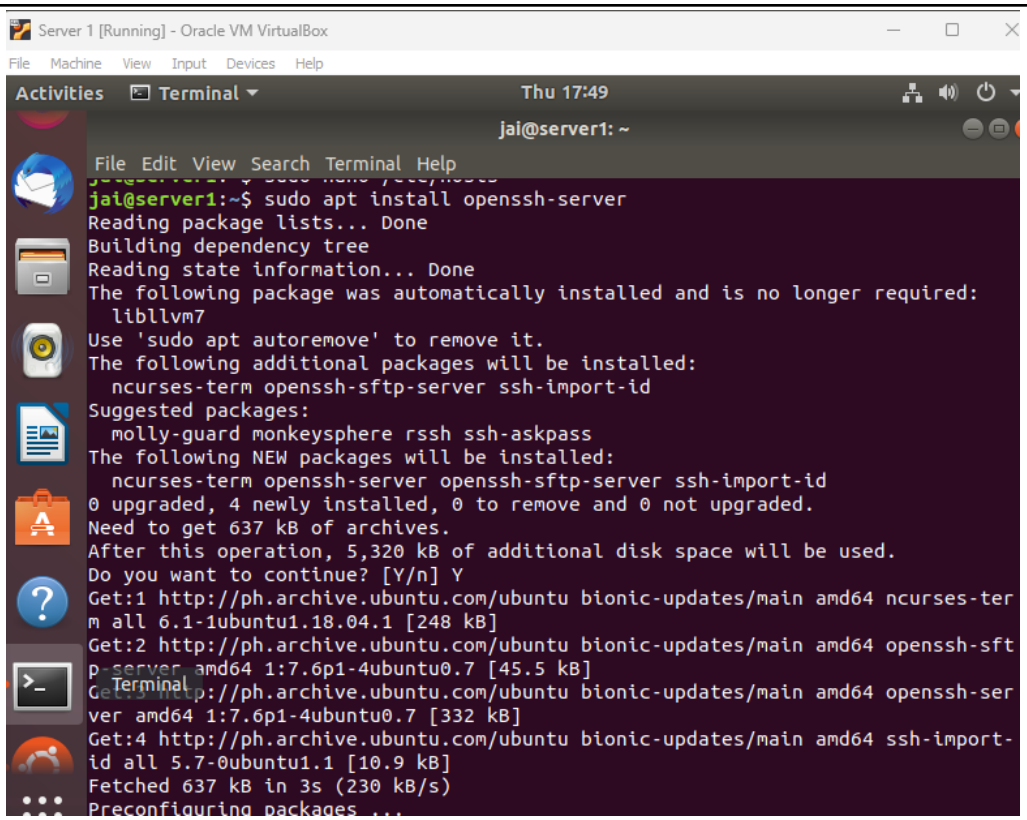
2.3 Type 127.0.0.1 workstation for the Local Machine

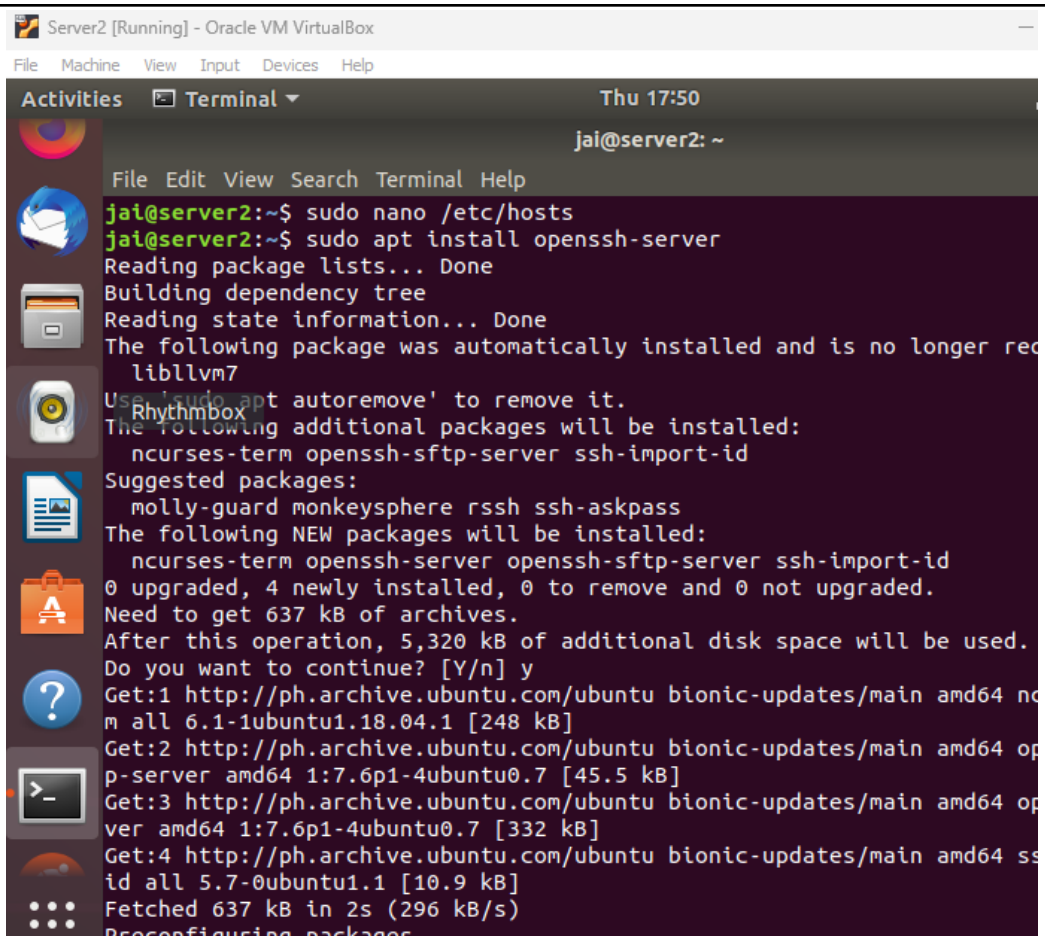


Task 2: Configure SSH on Server 1, Server 2, and Local Machine. Do the following:

1. Upgrade the packages by issuing the command *sudo apt update* and *sudo apt upgrade* respectively.
2. Install the SSH server using the command *sudo apt install openssh-server*.

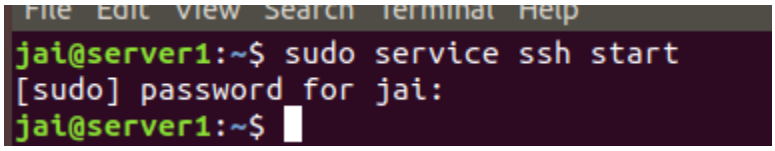


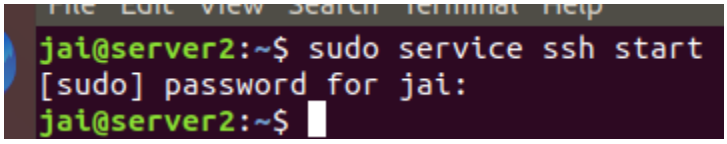

Server 1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Thu 17:49
jai@server1: ~
File Edit View Search Terminal Help
jai@server1:~$ sudo apt install openssh-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following 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 3s (230 kB/s)
Preconfiguring packages ...
Terminal

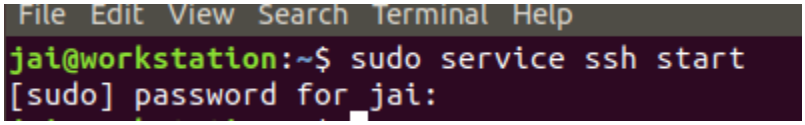


3. Verify if the SSH service has started by issuing the following commands:

   3.1 sudo service ssh start







   3.2 sudo systemctl status ssh

```
jai@server1:~$ sudo systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset:
   Active: active (running) since Thu 2023-08-17 17:54:05 PST; 4min 9s ago
  Process: 1019 ExecReload=/bin/kill -HUP $MAINPID (code=exited, status=0/SU
  Process: 1014 ExecReload=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
  Process: 567 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS
 Main PID: 576 (sshd)
    Tasks: 1 (limit: 4656)
   CGroup: /system.slice/ssh.service
           └─576 /usr/sbin/sshd -D

Aug 17 17:54:07 server1 systemd[1]: Reloading OpenBSD Secure Shell server.
Aug 17 17:54:07 server1 sshd[576]: Received SIGHUP; restarting.
Aug 17 17:54:07 server1 systemd[1]: Reloaded OpenBSD Secure Shell server.
Aug 17 17:54:07 server1 sshd[576]: Server listening on 0.0.0.0 port 22.
Aug 17 17:54:07 server1 sshd[576]: Server listening on :: port 22.
Aug 17 17:54:07 server1 systemd[1]: Reloading OpenBSD Secure Shell server.
Aug 17 17:54:07 server1 sshd[576]: Received SIGHUP; restarting.
Aug 17 17:54:07 server1 systemd[1]: Reloaded OpenBSD Secure Shell server.
Aug 17 17:54:07 server1 sshd[576]: Server listening on 0.0.0.0 port 22.
Aug 17 17:54:07 server1 sshd[576]: Server listening on :: port 22.
lines 1-21/21 (END)
```

```
jai@server2:~$ sudo systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset:
   Active: active (running) since Thu 2023-08-17 17:55:23 PST; 1min 6s ago
  Process: 974 ExecReload=/bin/kill -HUP $MAINPID (code=exited, status=0/SUC
  Process: 970 ExecReload=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
  Process: 554 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS
 Main PID: 571 (sshd)
    Tasks: 1 (limit: 4656)
   CGroup: /system.slice/ssh.service
           └─571 /usr/sbin/sshd -D

Aug 17 17:55:25 server2 systemd[1]: Reloading OpenBSD Secure Shell server.
Aug 17 17:55:25 server2 systemd[1]: Reloaded OpenBSD Secure Shell server.
Aug 17 17:55:25 server2 sshd[571]: Received SIGHUP; restarting.
Aug 17 17:55:25 server2 sshd[571]: Server listening on 0.0.0.0 port 22.
Aug 17 17:55:25 server2 sshd[571]: Server listening on :: port 22.
Aug 17 17:55:25 server2 systemd[1]: Reloading OpenBSD Secure Shell server.
Aug 17 17:55:25 server2 systemd[1]: Reloaded OpenBSD Secure Shell server.
Aug 17 17:55:25 server2 sshd[571]: Received SIGHUP; restarting.
Aug 17 17:55:25 server2 sshd[571]: Server listening on 0.0.0.0 port 22.
Aug 17 17:55:25 server2 sshd[571]: Server listening on :: port 22.
```

```
jai@workstation:~$ 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 Thu 2023-08-17 17:52:22 PST; 4min 36s ago
  Process: 921 ExecReload=/bin/kill -HUP $MAINPID (code=exited, status=0/SUCCES
  Process: 917 ExecReload=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
  Process: 662 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
 Main PID: 679 (sshd)
    Tasks: 1 (limit: 4656)
   CGroup: /system.slice/ssh.service
           └─679 /usr/sbin/sshd -D

Aug 17 17:52:23 workstation systemd[1]: Reloading OpenBSD Secure Shell server.
Aug 17 17:52:23 workstation systemd[1]: Reloaded OpenBSD Secure Shell server.
Aug 17 17:52:23 workstation sshd[679]: Received SIGHUP; restarting.
Aug 17 17:52:23 workstation sshd[679]: Server listening on 0.0.0.0 port 22.
Aug 17 17:52:23 workstation sshd[679]: Server listening on :: port 22.
Aug 17 17:52:23 workstation systemd[1]: Reloading OpenBSD Secure Shell server.
Aug 17 17:52:23 workstation sshd[679]: Received SIGHUP; restarting.
Aug 17 17:52:23 workstation systemd[1]: Reloaded OpenBSD Secure Shell server.
Aug 17 17:52:23 workstation sshd[679]: Server listening on 0.0.0.0 port 22.
Aug 17 17:52:23 workstation sshd[679]: Server listening on :: port 22.
```

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

4.1 sudo ufw allow ssh

```
jai@server1:~$ sudo ufw allow ssh
Rules updated
Rules updated (v6)
```

```
jai@server2:~$ sudo ufw allow ssh
Rules updated
Rules updated (v6)
```

```
jai@workstation:~$ sudo ufw allow ssh
Rules updated
Rules updated (v6)
```

4.2 sudo ufw enable

```
jai@server1:~$ sudo ufw enable
Firewall is active and enabled on system startup
```

```
jai@server2:~$ sudo ufw enable
Firewall is active and enabled on system startup
```

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

4.3 sudo ufw status

```
jai@server1:~$ sudo ufw status
Status: active

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

```
jai@server2:~$ sudo ufw status
Status: active

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

```
jai@workstation:~$ 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: 192.168.56.101
   1.2 Server 2 IP address: 192.168.56.102
   1.3 Server 3 IP address: 192.168.56.103
2. Make sure that they can ping each other.
   2.1 Connectivity test for Local Machine 1 to Server 1: ☐ Successful ☐ Not Successful

```
jai@workstation:~$ 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=0.698 ms
64 bytes from 192.168.56.102: icmp_seq=2 ttl=64 time=0.526 ms
64 bytes from 192.168.56.102: icmp_seq=3 ttl=64 time=0.451 ms
64 bytes from 192.168.56.102: icmp_seq=4 ttl=64 time=0.669 ms
64 bytes from 192.168.56.102: icmp_seq=5 ttl=64 time=0.457 ms
64 bytes from 192.168.56.102: icmp_seq=6 ttl=64 time=0.412 ms
64 bytes from 192.168.56.102: icmp_seq=7 ttl=64 time=0.699 ms
64 bytes from 192.168.56.102: icmp_seq=8 ttl=64 time=0.458 ms
64 bytes from 192.168.56.102: icmp_seq=9 ttl=64 time=0.605 ms
64 bytes from 192.168.56.102: icmp_seq=10 ttl=64 time=0.384 ms
64 bytes from 192.168.56.102: icmp_seq=11 ttl=64 time=0.476 ms
64 bytes from 192.168.56.102: icmp_seq=12 ttl=64 time=0.483 ms
64 bytes from 192.168.56.102: icmp_seq=13 ttl=64 time=0.870 ms
```

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

```
jai@workstation:~$ ping 192.168.56.103
PING 192.168.56.103 (192.168.56.103) 56(84) bytes of data.
64 bytes from 192.168.56.103: icmp_seq=1 ttl=64 time=0.824 m
64 bytes from 192.168.56.103: icmp_seq=2 ttl=64 time=0.386 m
64 bytes from 192.168.56.103: icmp_seq=3 ttl=64 time=0.463 m
64 bytes from 192.168.56.103: icmp_seq=4 ttl=64 time=0.491 m
```

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

```
jai@server1:~$ ping 192.168.56.103
PING 192.168.56.103 (192.168.56.103) 56(84) bytes of data.
64 bytes from 192.168.56.103: icmp_seq=1 ttl=64 time=0.921 m
64 bytes from 192.168.56.103: icmp_seq=2 ttl=64 time=0.479 m
64 bytes from 192.168.56.103: icmp_seq=3 ttl=64 time=0.452 m
64 bytes from 192.168.56.103: icmp_seq=4 ttl=64 time=0.588 m
64 bytes from 192.168.56.103: icmp_seq=5 ttl=64 time=0.661 m
64 bytes from 192.168.56.103: icmp_seq=6 ttl=64 time=0.506 m
64 bytes from 192.168.56.103: icmp_seq=7 ttl=64 time=0.571 m
64 bytes from 192.168.56.103: icmp_seq=8 ttl=64 time=0.421 m
64 bytes from 192.168.56.103: icmp_seq=9 ttl=64 time=0.743 m
```

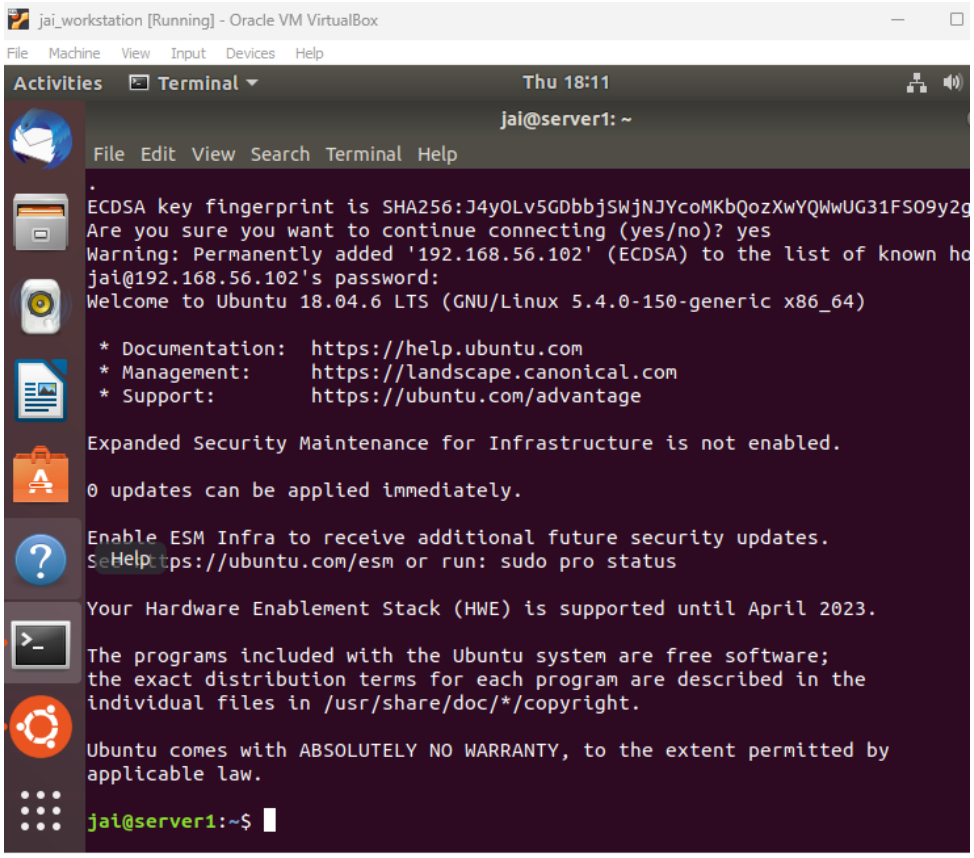
Task 4: Verify SSH connectivity on Server 1, Server 2, and Local Machine.

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

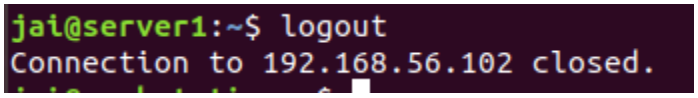
1.1 ssh username@ip_address_server1 for example, *ssh jvtaylar@192.168.56.120*

1.2 Enter the password for server 1 when prompted

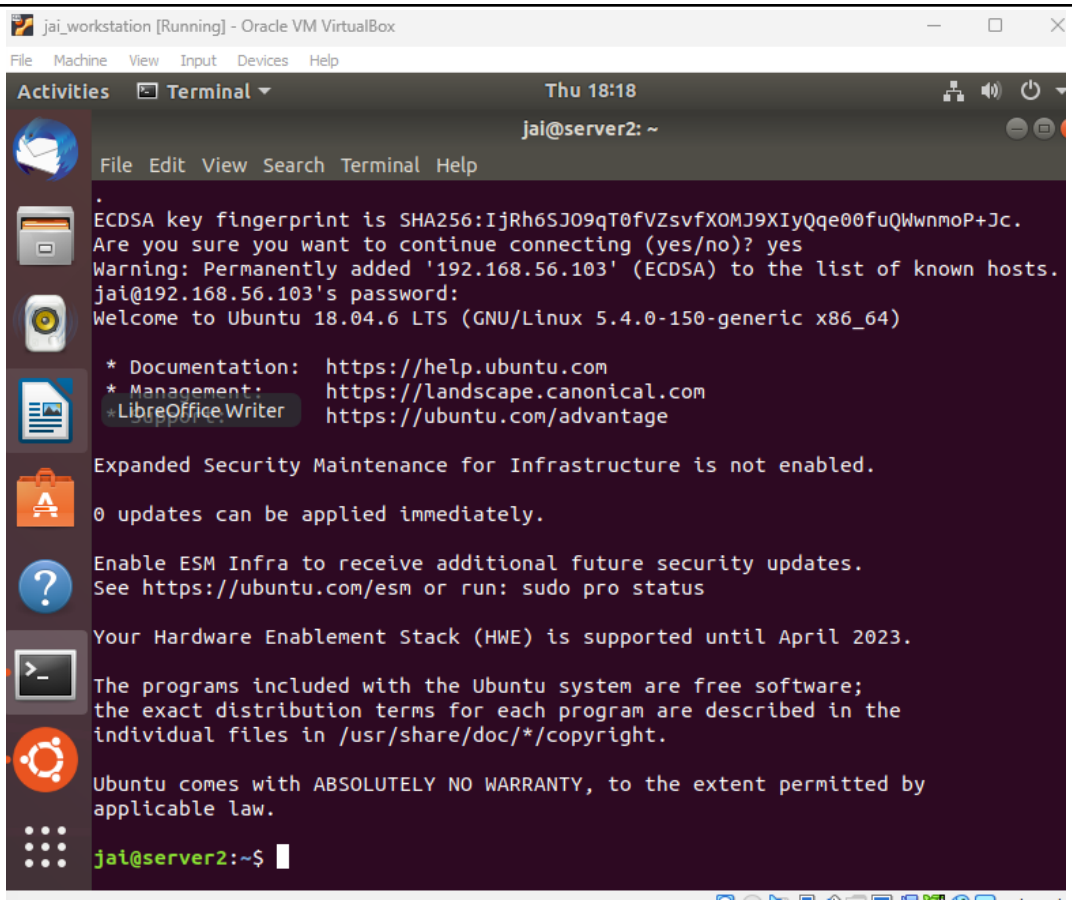
1.3 Verify that you are in server 1. The user should be in this format user@server1. For example, *jvtaylar@server1*



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



3. Do the same for Server 2.




```
jai@server2:~$ logout
Connection to 192.168.56.103 closed.
```

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)

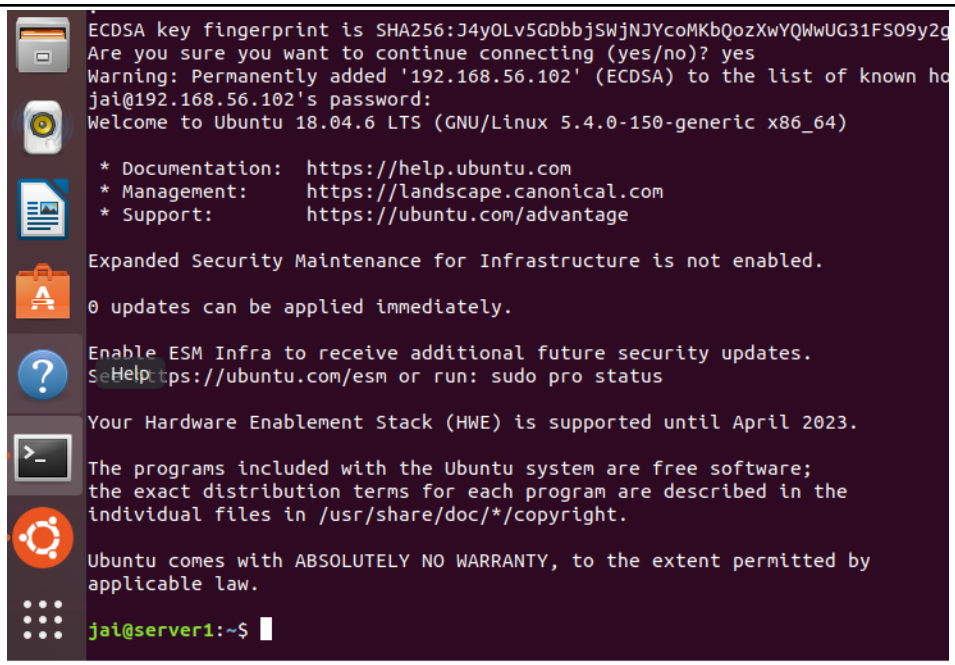
```
127.0.0.1        workstation
192.158.56.102   server1
```

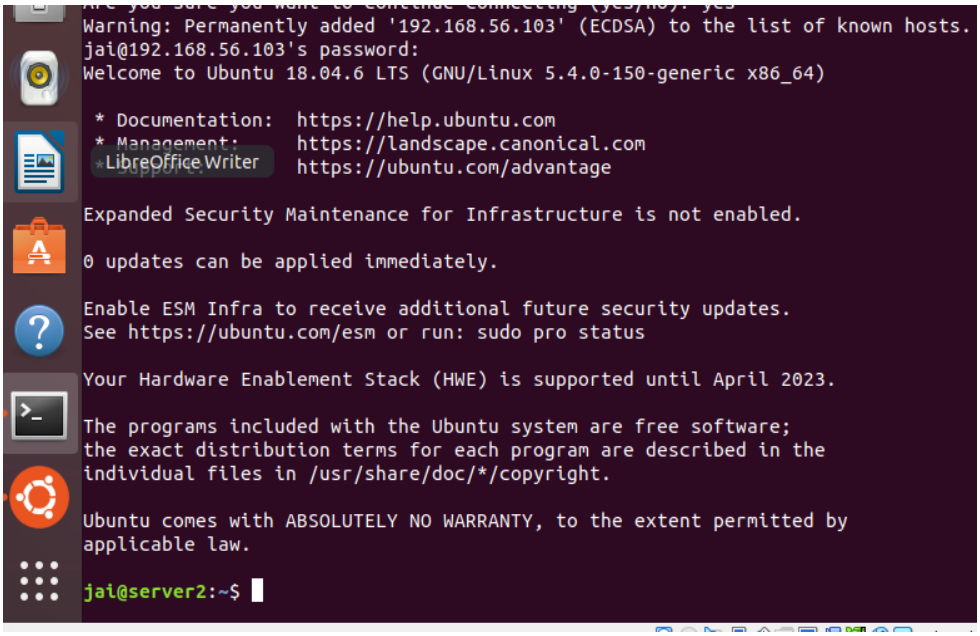
4.2 IP_address server 2 (provide the ip address of server 2 followed by the hostname)

```
192.158.56.103   server2
```

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?
   In Linux, usin hostnames in SSH commands is possible through DNS (Domain Name System). DNS translates human-readable hostnames into IP addresses, enaling user friendly remote connections without requiring the manual input of complex numrical addresses.

2. How secured is SSH?

SSH in Linux is highly secures sincs it employs strong encryption and authentication methods, safeguarding data during transmission. Key-based authentication adds an extra layer of protection, redcing the risk og password-based attacks. Regular updates and best practices frther enhance its security, making it a trusted choice for remote access.

Conclusion:

After performing this activity, I was able to accomplish all the task that is being asked. By applying all the learning, it is a worthwhile task to set up and test a virtual network for VMs in Linux. I have made it possible to share resources and conduct smooth conversation by developing this virtual environment. I have ensured the smooth operation of these virtual computers through connectivity tests. This activity highlights the value of novel approaches in contemporary computing and moves us one step closer to productive and efficient networking in the digital age.