Name: Denila, Nikko Bryner G.	**Date Performed:08/15/23**
Course/Section: CPE31S4	**Date Submitted:08/15/23**
Instructor: Dr. Jonathan V. Taylar	**Semester and SY: 1st Sem 2023-2024**

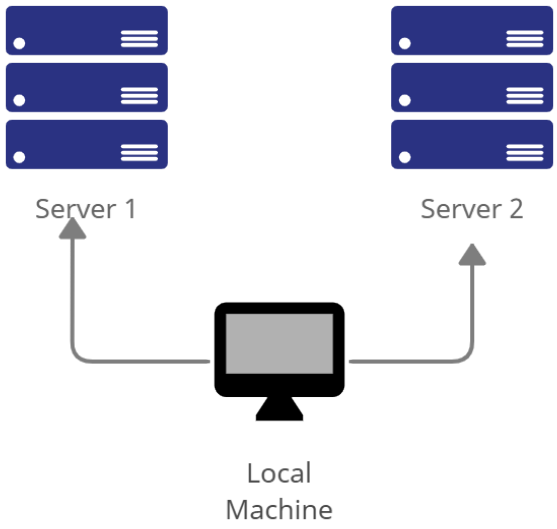
Activity 1: Configure Network using Virtual Machines

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:**

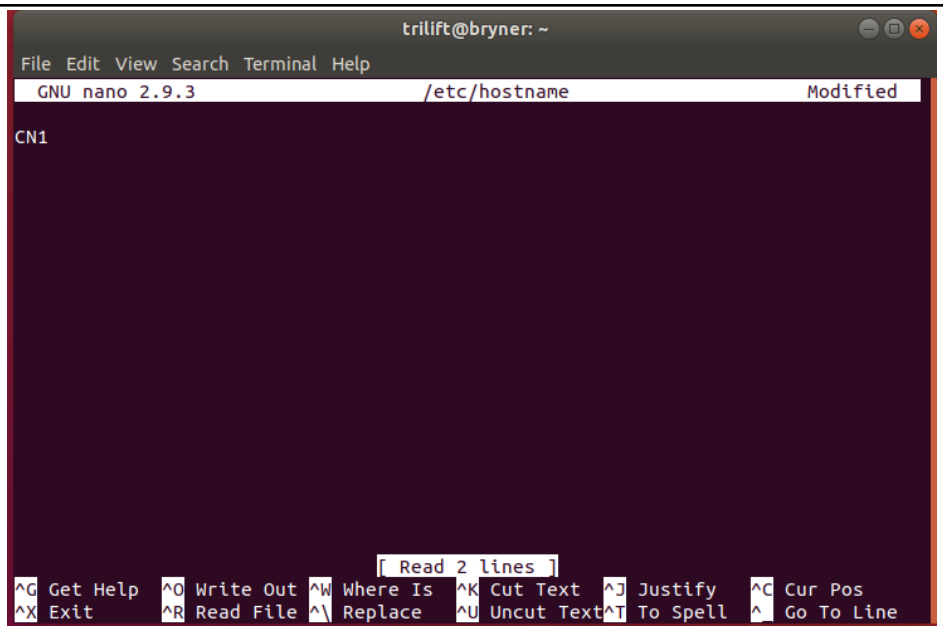
Network Topology:

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*).

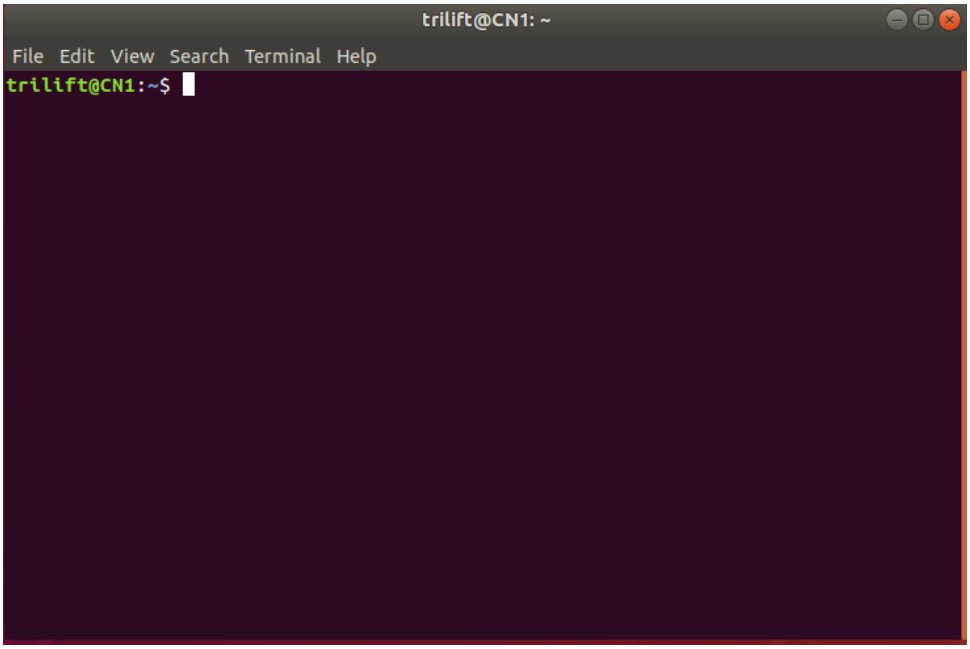


Task 1: Do the following on Server 1, Server 2, and Local Machine. In editing the file using nano command, press control + O to write out (save the file). Press enter when asked for the name of the file. Press control + X to end.

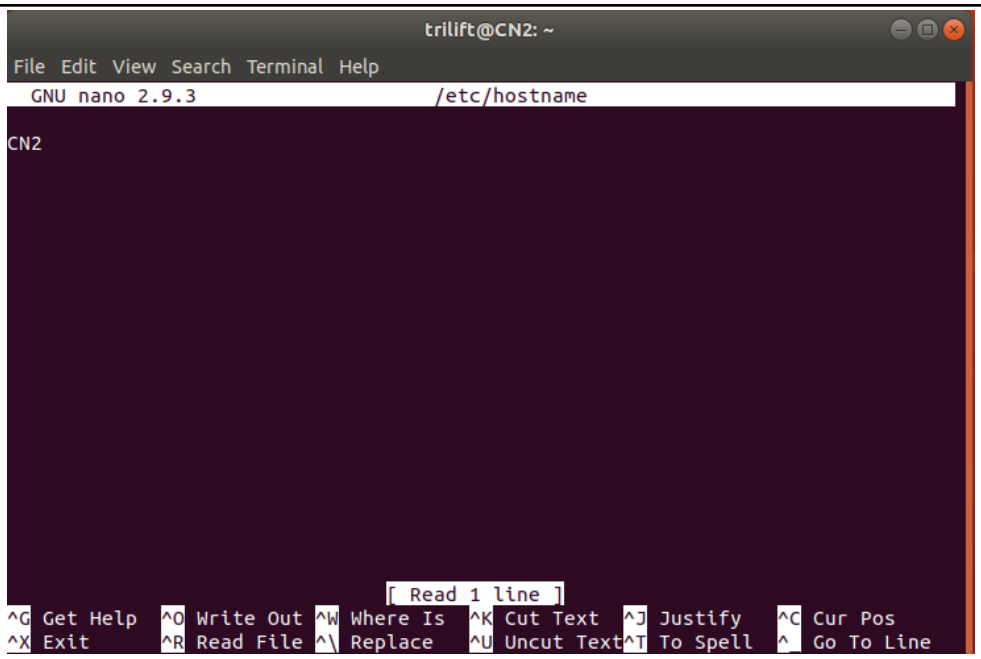
1. Change the hostname using the command *sudo nano /etc/hostname*
   1.1 Use server1 for Server 1



From the original bryner, I changed it to CN1(control node). After restarting it now changed its name.

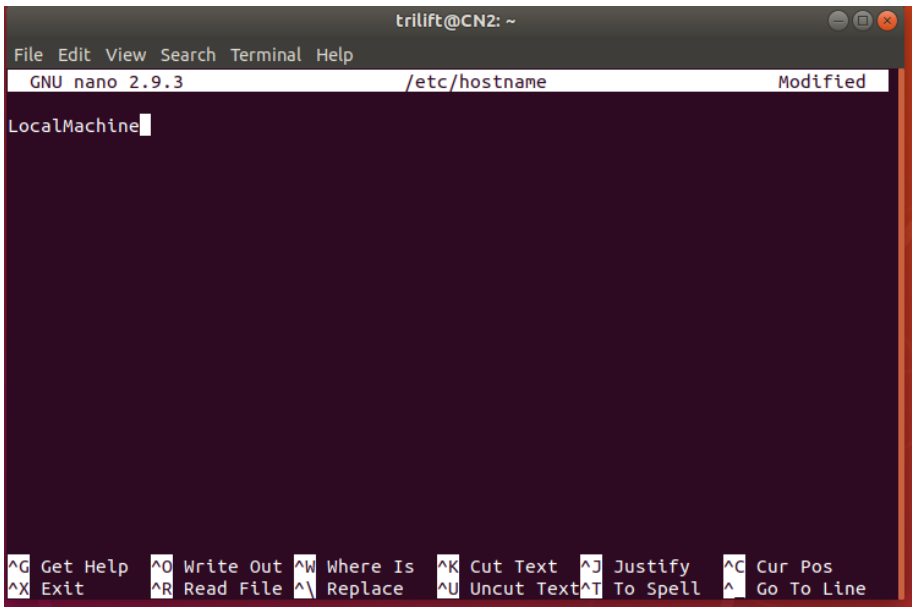


1.2 Use server2 for Server 2

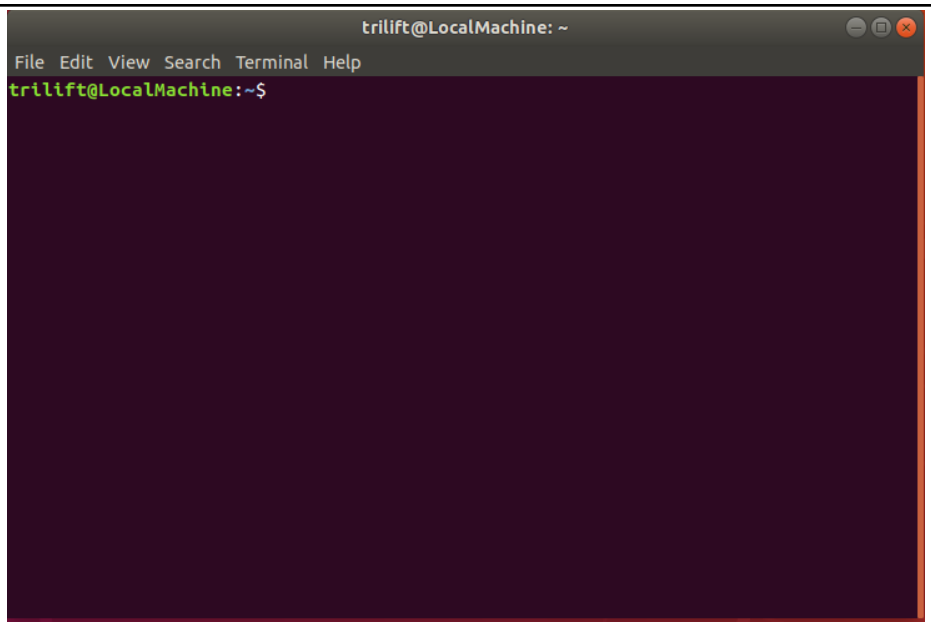


I wanted to change the name to CN2, but from prior installation i have already named it such so no changes were made.

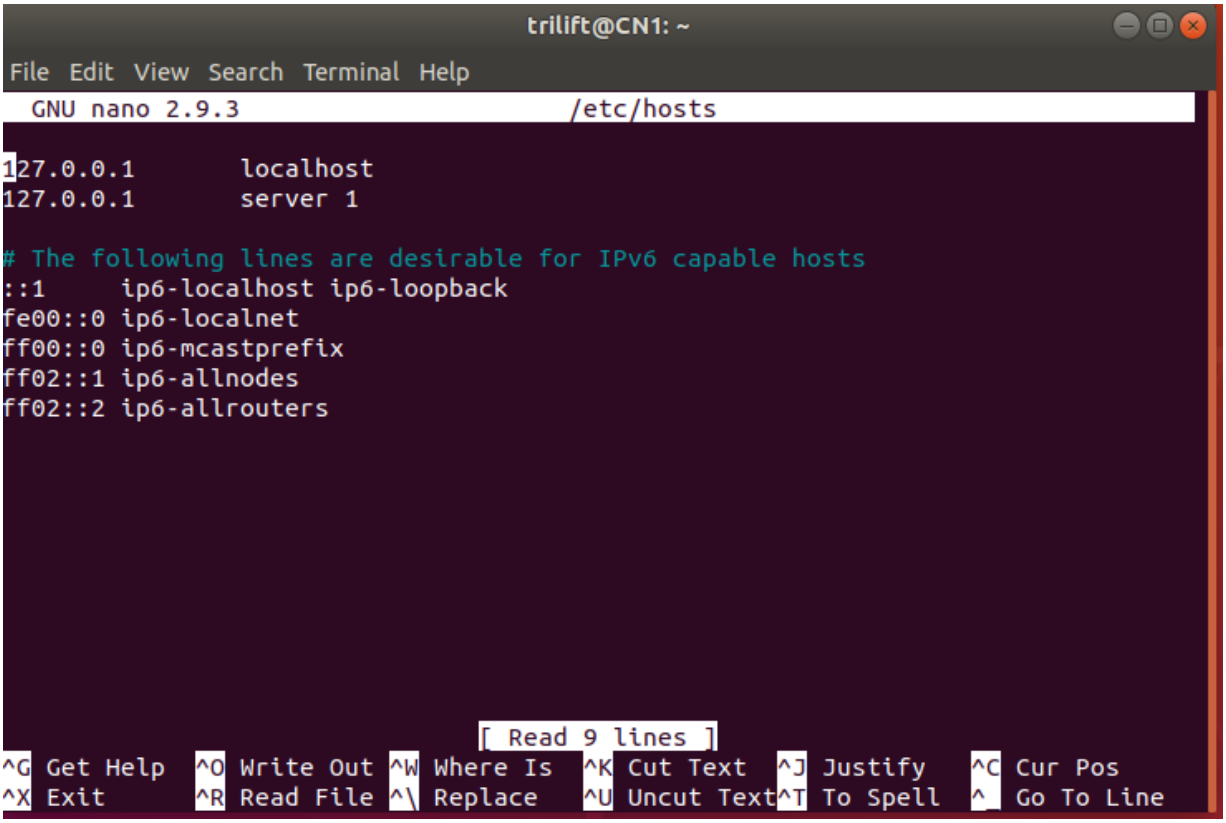
1.3 Use workstation for the Local Machine



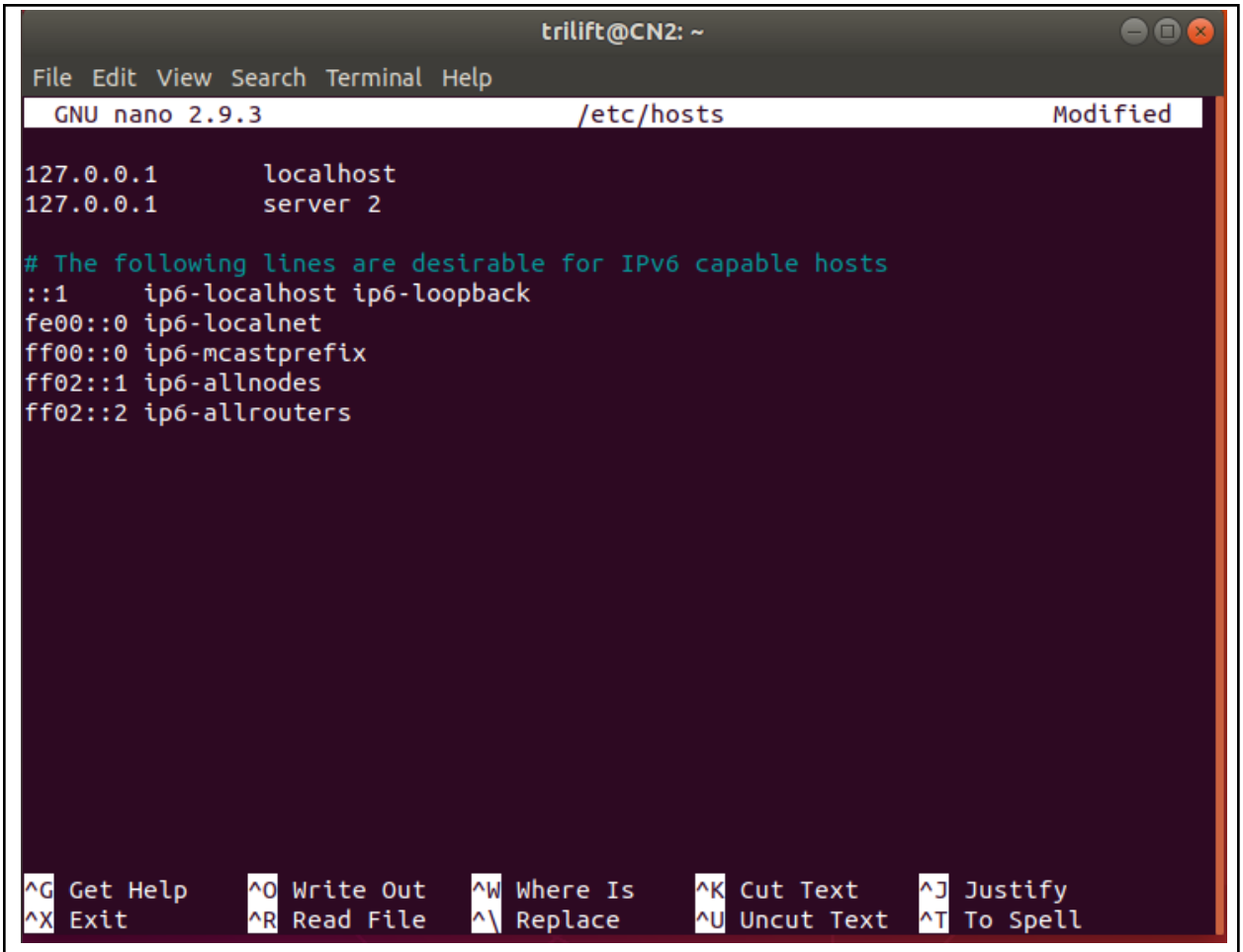
I changed the name to LocalMachine from its previous CN2. After restarting, the name changed.



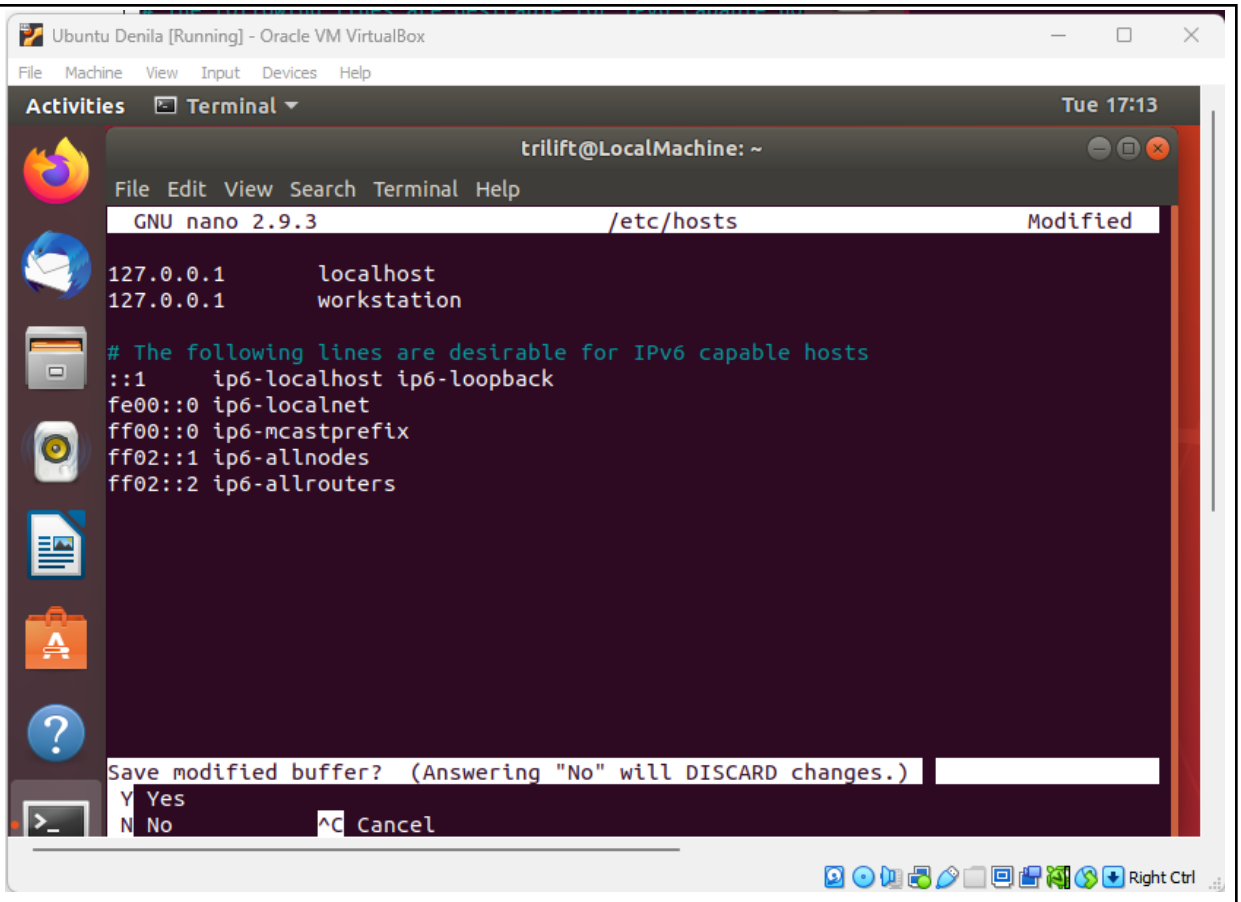
2. Edit the hosts using the command *sudo nano /etc/hosts*. Edit the second line.
   2.1 Type 127.0.0.1 server 1 for Server 1



   2.2 Type 127.0.0.1 server 2 for Server 2

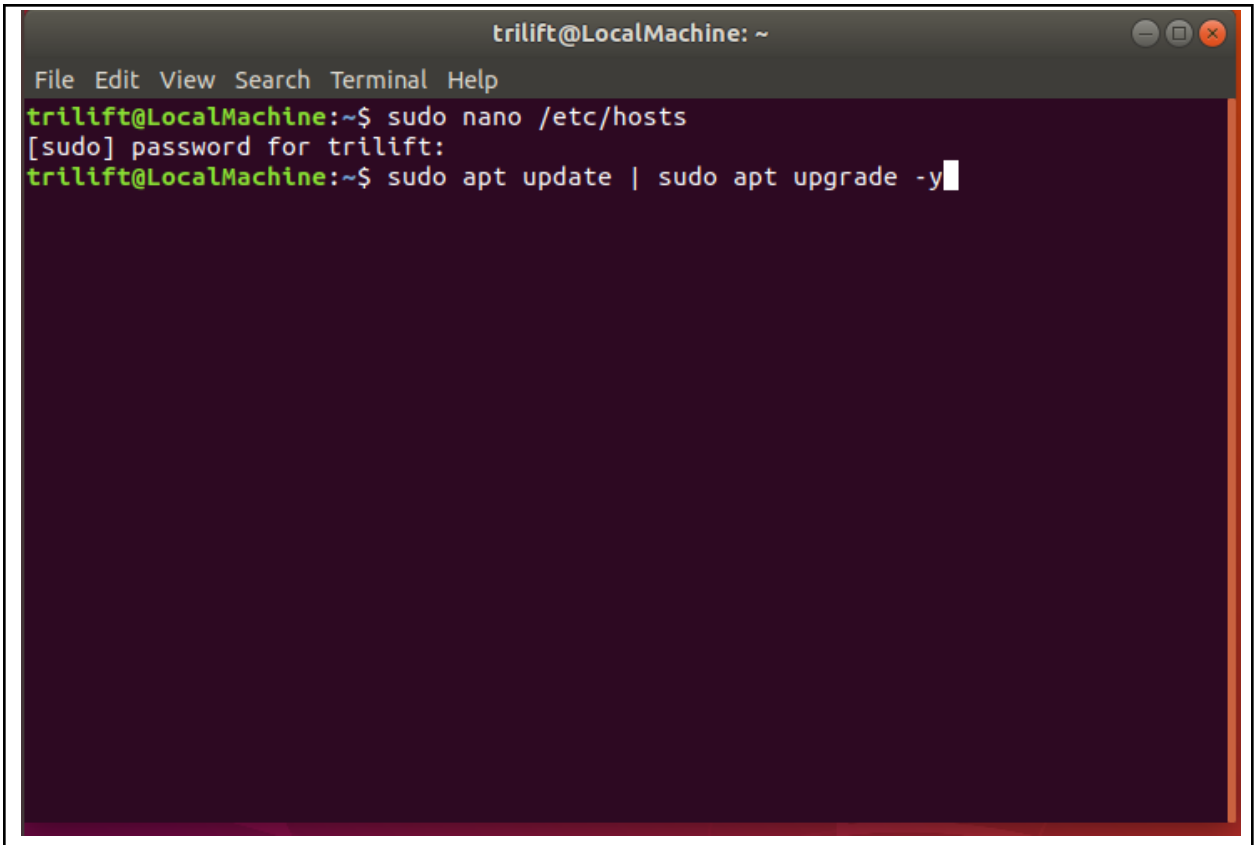


2.3 Type 127.0.0.1 workstation for the Local Machine



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.



This command is to update the system but I have recently updated the system so I did not continue this step.

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

```
trilift@LocalMachine:~$ sudo apt install openssh-server -y
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.
Get:1 http://ph.archive.ubuntu.com/ubuntu bionic-updates/main amd64 ncurses-ter
m all 6.1-1ubuntu1.18.04.1 [248 kB]
13% [1 ncurses-term 102 kB/248 kB 41%]
```

I installed the SSH server using this command to all 3 systems. I used the phrase -y to immediately answer y to the questions to install.

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*

```
trilift@LocalMachine:~$ sudo service ssh start
trilift@LocalMachine:~$ 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 17:23:45 +08; 36s ago
   Main PID: 2742 (sshd)
      Tasks: 1 (limit: 2318)
     CGroup: /system.slice/ssh.service
             └─2742 /usr/sbin/sshd -D

Aug 15 17:23:45 LocalMachine systemd[1]: Starting OpenBSD Secure Shell server..
Aug 15 17:23:45 LocalMachine sshd[2742]: Server listening on 0.0.0.0 port 22.
Aug 15 17:23:45 LocalMachine sshd[2742]: Server listening on :: port 22.
Aug 15 17:23:45 LocalMachine 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*

   4.2 *sudo ufw enable*

   4.3 *sudo ufw status*

```
trilift@LocalMachine:~$ sudo ufw allow ssh
Rules updated
Rules updated (v6)
trilift@LocalMachine:~$ sudo ufw enable
Firewall is active and enabled on system startup
trilift@LocalMachine:~$ sudo ufw status
Status: active

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

I updated the firewall in all 3 systems using these commands.

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**

   1.2 Server 2 IP address: 192.168.56.**107**

   1.3 Server 3 IP address: 192.168.56.**102**

2. Make sure that they can ping each other.

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

```
trilift@LocalMachine:~$ 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=1.02 ms
64 bytes from 192.168.56.106: icmp_seq=2 ttl=64 time=0.811 ms
64 bytes from 192.168.56.106: icmp_seq=3 ttl=64 time=1.17 ms
64 bytes from 192.168.56.106: icmp_seq=4 ttl=64 time=0.391 ms
^C
--- 192.168.56.106 ping statistics ---
```

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

```
trilift@CN2:~$ 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=2.57 ms
64 bytes from 192.168.56.102: icmp_seq=2 ttl=64 time=0.514 ms
64 bytes from 192.168.56.102: icmp_seq=3 ttl=64 time=0.583 ms
64 bytes from 192.168.56.102: icmp_seq=4 ttl=64 time=1.50 ms
64 bytes from 192.168.56.102: icmp_seq=5 ttl=64 time=0.439 ms
```

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

```
trilift@CN1:~$ ping 192.168.56.107
PING 192.168.56.107 (192.168.56.107) 56(84) bytes of data.
64 bytes from 192.168.56.107: icmp_seq=1 ttl=64 time=1.75 ms
64 bytes from 192.168.56.107: icmp_seq=2 ttl=64 time=0.891 ms
64 bytes from 192.168.56.107: icmp_seq=3 ttl=64 time=0.342 ms
64 bytes from 192.168.56.107: icmp_seq=4 ttl=64 time=0.674 ms
^C
```

All three tests were successful and access to all 3 servers are now online.

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.

```
trilift@LocalMachine:~$ ssh trilift@192.168.56.106
The authenticity of host '192.168.56.106 (192.168.56.106)' can't be established
.
ECDSA key fingerprint is SHA256:sJmKiyIxXJ5hT3/6vxqSdxPjwNS9kuhEXRMP1qI6Jzs.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.56.106' (ECDSA) to the list of known hosts.
trilift@192.168.56.106's password:
Permission denied, please try again.
trilift@192.168.56.106's password:
```

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

```
trilift@CN1:~$
```

I am now using the CN1 unit.

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

```
trilift@CN1:~$ logout
Connection to 192.168.56.106 closed.
trilift@LocalMachine:~$
```

I am now back to the local machine.

3. Do the same for Server 2.

```
trilift@CN2:~$ logout
Connection to 192.168.56.107 closed.
trilift@LocalMachine:~$
```

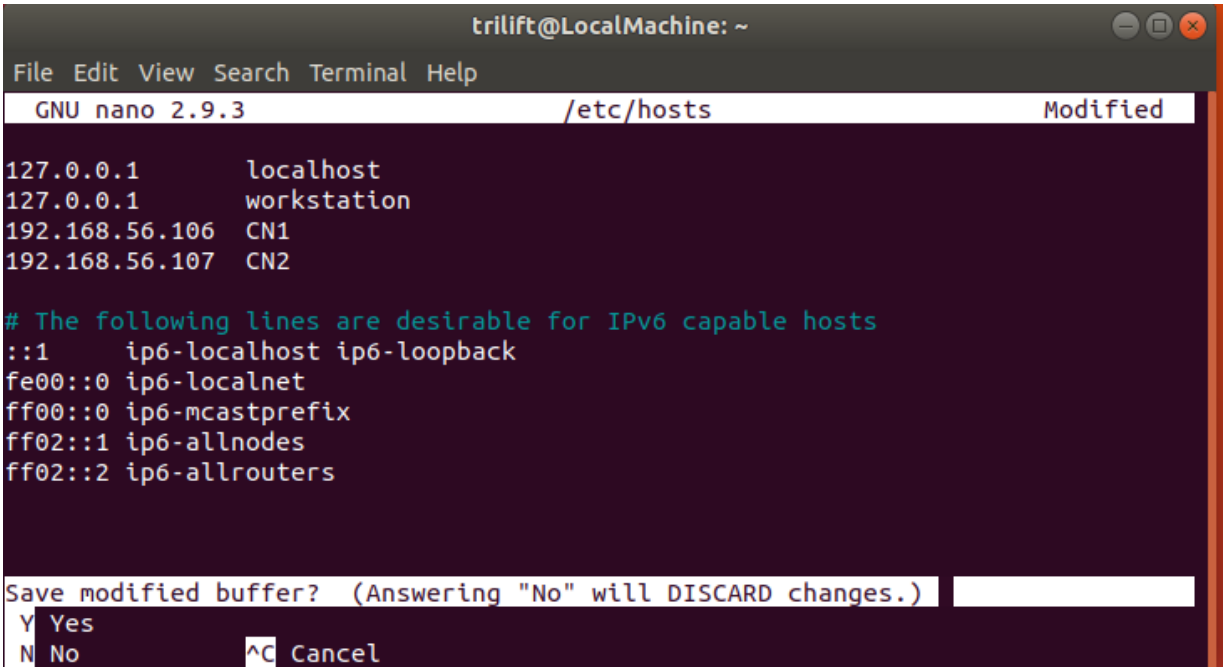
From CN2 I logged out and am now back to LocalMachine again.

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.

```
trilift@LocalMachine:~$ ssh trilift@CN1
The authenticity of host 'cn1 (192.168.56.106)' can't be established.
ECDSA key fingerprint is SHA256:sJmKiyIxXJ5hT3/6vxqSdxPjwNS9kuhEXRMP1qI6Jzs.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'cn1' (ECDSA) to the list of known hosts.
trilift@cn1's password:
```

```
trilift@CN1:~$
```

I was now able to use the name of the server (CN1 and CN2) to access their terminal and edit their settings and configurations from the original server.

Reflections:

Answer the following:

1. How are we able to use the hostname instead of IP address in SSH commands?
   We were able to use the hostname to access the control node using the SSH commands because we added their respective IP addresses to the hosts that the local machine has access to. This made it so that whenever we called their hostname, the identified IP address would be used.

2. How secured is SSH?

   SSH is very secure because only the local machine and those who have access to the IP address of the control nodes can access the information on each command and instructions. This is why securing your IP address is important because some people can access your computer files using the address itself.

Conclusion:

In this laboratory activity, I was able to learn about the SSH server and how to connect three different systems together through their CLI using the SSH command. This makes it so that we can easily access faraway units which would take a lot of time and effort to travel from one place to another. This reduces the time and makes things more efficient. This is especially important in businesses which connect multiple units to a single server. From one click we can deploy programs and scripts to other systems easier and faster.