Name: KEVIN SUMAYA	**Date Performed: Thursday 17 2023**
Course/Section: CPE232 CPE31S6	**Date Submitted: Thursday 17 2023**
Instructor: Sir. Jonathan Taylar	**Semester and SY: 2023-2024**

Activity 1: Configure Network using Virtual Machines

1. Objectives:

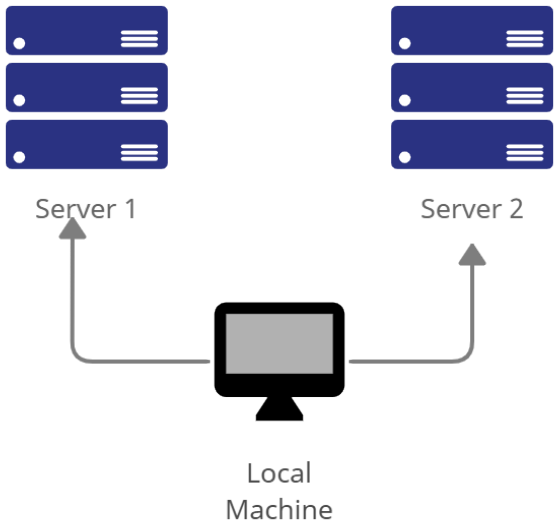
1.1. Create and configure Virtual Machines in Microsoft Azure or VirtualBox

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

```
kevin@server1:~$ sudo nano /etc/hostname
[sudo] password for kevin:
```

   1.2 Use server2 for Server 2

```
kevin@server2:~$ sudo nano /etc/hostname
```

   1.3 Use workstation for the Local Machine

```
kevin@Workstation:~$
```

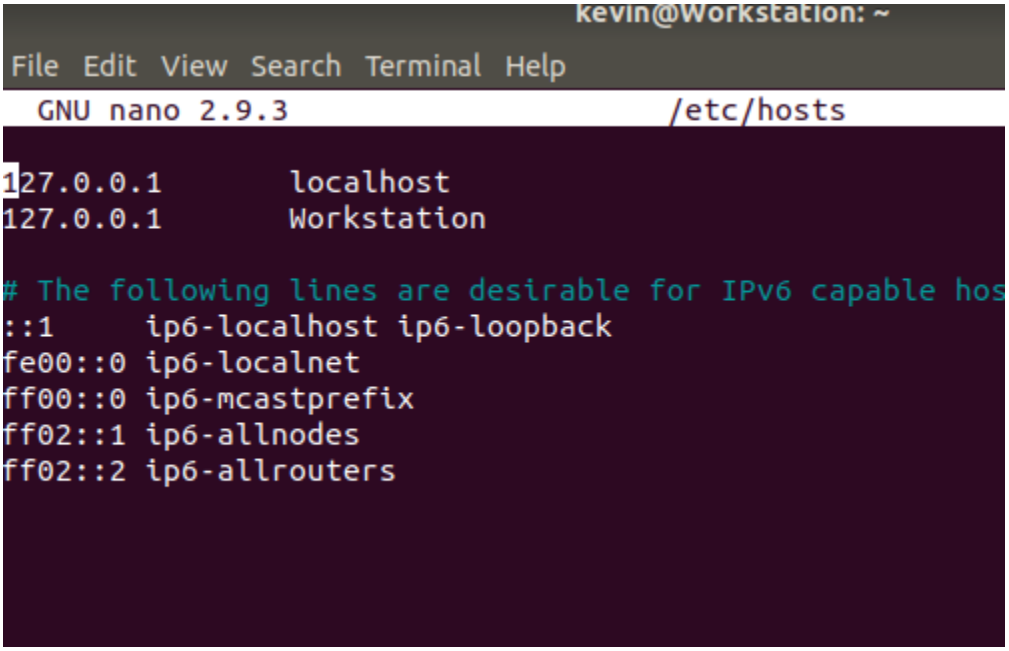
2. Edit the hosts using the command *sudo nano /etc/hosts*. Edit the second line.
   2.1 Type 127.0.0.1 server 1 for Server 1

```
  GNU nano 2.9.3                        /etc/hosts

127.0.0.1       localhost
127.0.0.1       Server 1

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

   2.2 Type 127.0.0.1 server 2 for Server 2

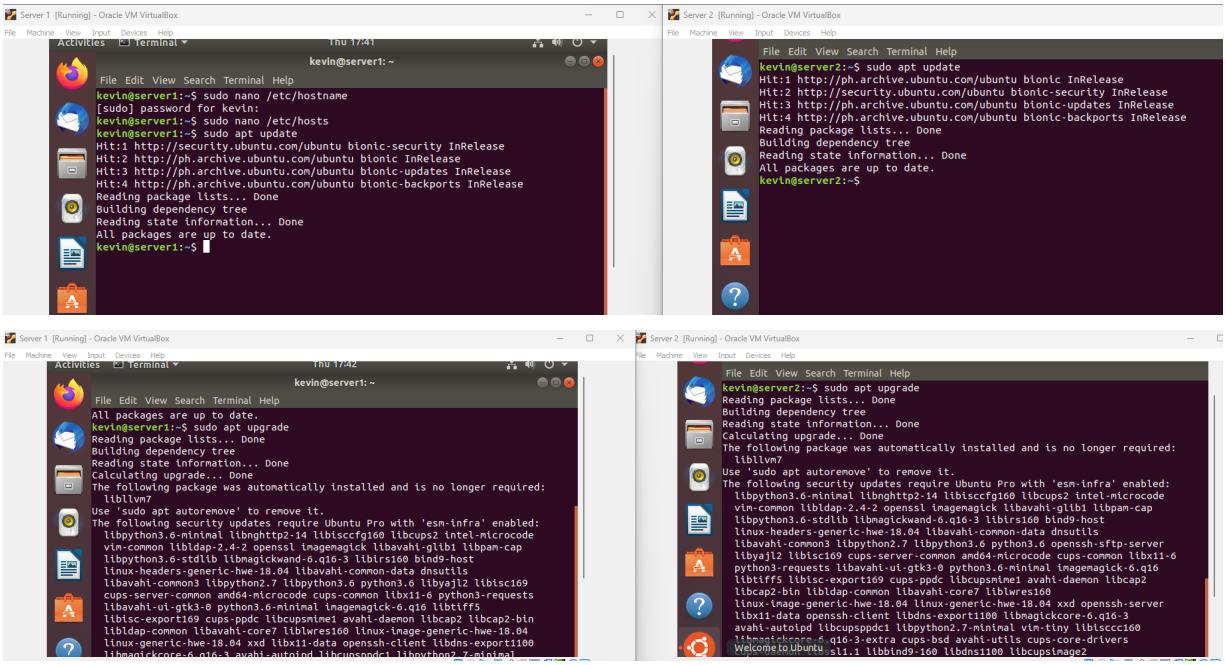
```
  GNU nano 2.9.3                        /etc/hosts

127.0.0.1       localhost
127.0.0.1       Server 2

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

2.3 Type 127.0.0.1 workstation for the Local Machine

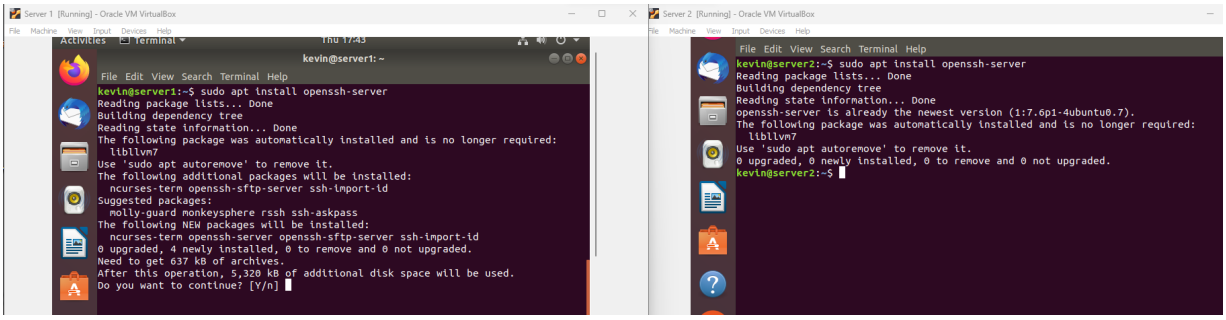


Task 2: Configure SSH on Server 1, Server 2, and Local Machine. Do the following:

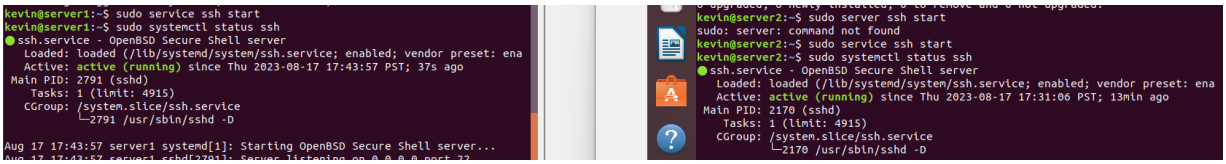
1. Upgrade the packages by issuing the command *sudo apt update* and *sudo apt upgrade* respectively.



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

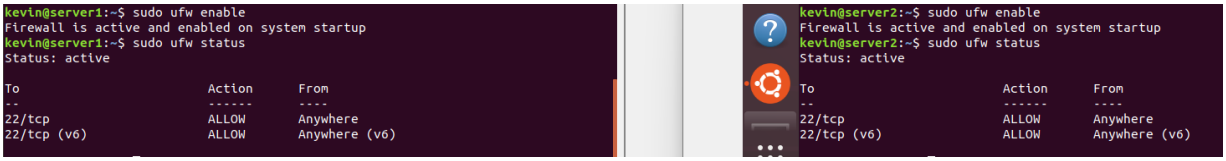


3. Verify if the SSH service has started by issuing the following commands:
   3.1 *sudo service ssh start*
   3.2 *sudo systemctl status ssh*



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*



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.**102**
   1.2 Server 2 IP address: 192.168.56.**103**
   1.3 Server 3 IP address: 192.168.56.**101**
2. Make sure that they can ping each other.
   2.1 Connectivity test for Local Machine 1 to Server 1: ☐ Successful ☐ Not Successful

```
kevin@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=0.689 ms
64 bytes from 192.168.56.102: icmp_seq=2 ttl=64 time=0.420 ms
64 bytes from 192.168.56.102: icmp_seq=3 ttl=64 time=0.403 ms
64 bytes from 192.168.56.102: icmp_seq=4 ttl=64 time=0.398 ms
64 bytes from 192.168.56.102: icmp_seq=5 ttl=64 time=0.479 ms
```

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

```
kevin@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=1.07 ms
64 bytes from 192.168.56.103: icmp_seq=2 ttl=64 time=0.449 ms
```

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

```
kevin@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=0.975 ms
64 bytes from 192.168.56.103: icmp_seq=2 ttl=64 time=0.542 ms
```

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

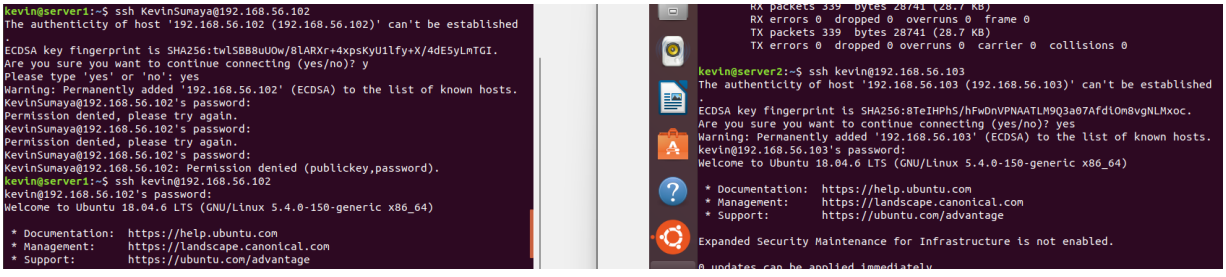
1. On the Local Machine, issue the following commands:

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

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.



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)

```
  GNU nano 2.9.3                          /etc/hosts

127.0.0.1       localhost
127.0.0.1       Workstation
192.168.56.102  Server 1
192.168.56.103  Server 2

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

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.

```
kevin@Workstation:~$ ssh kevin@Server1
The authenticity of host 'server1 (192.168.56.102)' can't be established.
ECDSA key fingerprint is SHA256:twlSBB8uUOw/8lARXr+4xpsKyU1lfy+X/4dE5yLmTGI.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'server1' (ECDSA) to the list of known hosts.
kevin@server1's password:
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-150-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.

78 additional security updates can be applied with ESM Infra.
Learn more about enabling ESM Infra service for Ubuntu 18.04 at
https://ubuntu.com/18-04

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

```
kevin@server1:~$ ssh kevin@server2
The authenticity of host 'server2 (192.168.56.103)' can't be established.
ECDSA key fingerprint is SHA256:8TeIHPhS/hFwDnVPNAATLM9Q3a07AfdiOm8vgNLMxoc.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'server2,192.168.56.103' (ECDSA) to the list of know
n hosts.
kevin@server2's password:
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-150-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.

78 additional security updates can be applied with ESM Infra.
Learn more about enabling ESM Infra service for Ubuntu 18.04 at
https://ubuntu.com/18-04

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.
Last login: Thu Aug 17 17:59:42 2023 from 192.168.56.103
```

Reflections:

Answer the following:

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

I stated or imputed the name of the hostname beside the IPaddress of the server.

2. How secured is SSH?

It is a very secure protection because it uses encryption to secure connection between a client and a server. all files are encrypted to protect against attack in the network

Conclusion:

Setting up a virtual network on linux offers the flexibility and convenience of simulating various network environments for testing and development purposes. By employing tools like VirtualBox or VMware, users can create isolated network configurations, replicate real world scenarios, and ensure optimal connectivity.