

Name: Richmond Ryan L. Reyes	Date Performed: August 14, 2023
Course/Section: CPE31S4	Date Submitted: August 15, 2023
Instructor: Dr. Jonathan Taylar	Semester and SY: 3rd Year 1st Semester

Activity 1: Configure Network using Virtual Machines

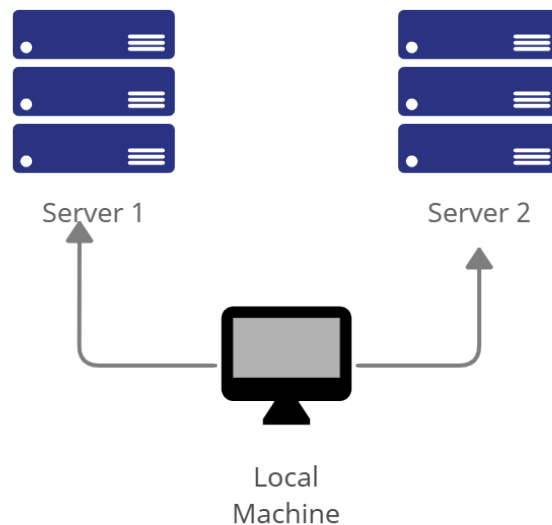
1. Objectives:

- 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

The screenshot shows a terminal window titled 'richmond@richmond-VirtualBox: ~'. The nano text editor is open, editing the file '/etc/hostname'. The current line in the editor is 'controlNode1'. The terminal prompt is 'GNU nano 6.2 /etc/hostname *'.

```
richmond@controlNode1: ~  
richmond@controlNode1:~$
```

1.2 Use server2 for Server 2

```
richmond@richmond-VirtualBox: ~  
GNU nano 6.2 /etc/hostname *  
controlNode2
```

```
richmond@controlNode2: ~  
richmond@controlNode2:~$
```

1.3 Use workstation for the Local Machine

```
richmond@richmond-VirtualBox: ~  
GNU nano 6.2 /etc/hostname  
manageNode
```

```
richmond@manageNode: ~  
richmond@manageNode:~$
```

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

```
richmond@controlNode1: ~  
GNU nano 6.2 /etc/hosts *  
127.0.0.1 controlNode1  
  
# 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

```
richmond@controlNode2: ~  
GNU nano 6.2 /etc/hosts *  
127.0.0.1 controlNode2  
  
# 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

```
richmond@manageNode: ~
GNU nano 6.2 /etc/hosts *
127.0.0.1    manageNode

# 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
```

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. \

```
richmond@manageNode: ~
richmond@manageNode:~$ sudo apt update | sudo apt upgrade -y --fix-missing

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
```

```
richmond@controlNode1: ~
richmond@controlNode1:~$ sudo apt update | sudo apt upgrade -y --fix-missing
[sudo] password for richmond:

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
```

```
richmond@controlNode2: ~
richmond@controlNode2:~$ sudo apt update | sudo apt upgrade -y --fix-missing

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
```

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

```
richmond@manageNode: ~
richmond@manageNode:~$ sudo apt install openssh-server
[sudo] password for richmond:
Sorry, try again.
[sudo] password for richmond:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
```

```
richmond@controlNode1: ~  
richmond@controlNode1:~$ sudo apt install openssh-server  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following additional packages will be installed:
```

```
richmond@controlNode2: ~  
richmond@controlNode2:~$ sudo apt install openssh-server  
[sudo] password for richmond:  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following additional packages will be installed:
```

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

3.1 *sudo service ssh start*

```
richmond@manageNode:~$ sudo service ssh start  
richmond@manageNode:~$
```

```
richmond@ControlNode1:~$ sudo service ssh start  
richmond@ControlNode1:~$
```

```
richmond@controlNode2:~$ sudo service ssh start  
richmond@controlNode2:~$
```

3.2 *sudo systemctl status ssh*

```
richmond@manageNode:~$ sudo systemctl status ssh  
● ssh.service - OpenBSD Secure Shell server  
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset:   
   Active: active (running) since Tue 2023-08-15 17:11:20 PST; 3min 16s ago  
     Docs: man:sshd(8)  
           man:sshd_config(5)  
    Main PID: 36517 (sshd)  
      Tasks: 1 (limit: 2256)  
     Memory: 1.7M  
        CPU: 14ms  
     CGroup: /system.slice/ssh.service  
             └─36517 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"  
  
Aug 15 17:11:20 manageNode systemd[1]: Starting OpenBSD Secure Shell server...  
Aug 15 17:11:20 manageNode sshd[36517]: Server listening on 0.0.0.0 port 22.  
Aug 15 17:11:20 manageNode sshd[36517]: Server listening on :: port 22.  
Aug 15 17:11:20 manageNode systemd[1]: Started OpenBSD Secure Shell server.  
lines 1-16/16 (END)
```

```

richmond@ControlNode1: ~
richmond@ControlNode1:~$ sudo systemctl status ssh
[sudo] password for richmond:
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: en
   Active: active (running) since Tue 2023-08-15 16:49:20 PST; 41s ago
     Docs: man:sshd(8)
           man:sshd_config(5)
   Process: 655 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
    Main PID: 676 (sshd)
      Tasks: 1 (limit: 2253)
     Memory: 3.7M
        CPU: 25ms
    CGroup: /system.slice/ssh.service
            └─676 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Aug 15 16:49:20 ControlNode1 systemd[1]: Starting OpenBSD Secure Shell server...
Aug 15 16:49:20 ControlNode1 sshd[676]: Server listening on 0.0.0.0 port 22.
Aug 15 16:49:20 ControlNode1 sshd[676]: Server listening on :: port 22.
Aug 15 16:49:20 ControlNode1 systemd[1]: Started OpenBSD Secure Shell server.
lines 1-17/17 (END)

richmond@ControlNode2:~$ sudo systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: en
   Active: active (running) since Tue 2023-08-15 17:10:55 PST; 4min 31s ago
     Docs: man:sshd(8)
           man:sshd_config(5)
   Main PID: 2385 (sshd)
      Tasks: 1 (limit: 2253)
     Memory: 1.7M
        CPU: 14ms
    CGroup: /system.slice/ssh.service
            └─2385 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Aug 15 17:10:55 ControlNode2 systemd[1]: Starting OpenBSD Secure Shell server.
Aug 15 17:10:55 ControlNode2 sshd[2385]: Server listening on 0.0.0.0 port 22.
Aug 15 17:10:55 ControlNode2 sshd[2385]: Server listening on :: port 22.
Aug 15 17:10:55 ControlNode2 systemd[1]: Started OpenBSD Secure Shell server.
lines 1-16/16 (END)

```

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

4.1 *sudo ufw allow ssh*

```

richmond@manageNode:~$ sudo ufw allow ssh
Rules updated
Rules updated (v6)

```

```

richmond@ControlNode1:~$ sudo ufw allow ssh
Rules updated
Rules updated (v6)

```

```

richmond@ControlNode2:~$ sudo ufw allow ssh
Rules updated
Rules updated (v6)

```

4.2 *sudo ufw enable*

```

richmond@manageNode:~$ sudo ufw enable
Firewall is active and enabled on system startup

```

```

richmond@ControlNode1:~$ sudo ufw enable
Firewall is active and enabled on system startup

```

```
richmond@ControlNode2:~$ sudo ufw enable
Firewall is active and enabled on system startup
richmond@ControlNode2:~$ sudo ufw status
```

4.3 *sudo ufw status*

```
richmond@manageNode:~$ sudo ufw status
Status: active

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

```
richmond@ControlNode1:~$ sudo ufw status
Status: active

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

```
richmond@ControlNode2:~$ 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: net-tools

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.____

```
192.168.56.108
```

1.2 Server 2 IP address: 192.168.56.____

```
192.168.56.109
```

1.3 Server 3 IP address: 192.168.56.____

```
192.168.56.110
```

2. Make sure that they can ping each other.

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

```
richmond@manageNode:~$ 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=0.029 ms
64 bytes from 192.168.56.108: icmp_seq=2 ttl=64 time=0.030 ms
64 bytes from 192.168.56.108: icmp_seq=3 ttl=64 time=0.059 ms
```

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

```
richmond@manageNode:~$ ping 192.168.56.109
PING 192.168.56.109 (192.168.56.109) 56(84) bytes of data.
64 bytes from 192.168.56.109: icmp_seq=1 ttl=64 time=0.896 ms
64 bytes from 192.168.56.109: icmp_seq=2 ttl=64 time=0.744 ms
64 bytes from 192.168.56.109: icmp_seq=3 ttl=64 time=1.77 ms
^Z
[2]+  Stopped                  ping 192.168.56.109
```

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

```
richmond@ControlNode1:~$ ping 192.168.56.110
PING 192.168.56.110 (192.168.56.110) 56(84) bytes of data.
64 bytes from 192.168.56.110: icmp_seq=1 ttl=64 time=1.14 ms
64 bytes from 192.168.56.110: icmp_seq=2 ttl=64 time=2.04 ms
64 bytes from 192.168.56.110: icmp_seq=3 ttl=64 time=0.858 ms
64 bytes from 192.168.56.110: icmp_seq=4 ttl=64 time=0.691 ms
^Z
[1]+  Stopped                  ping 192.168.56.110
```

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 jvtaylor@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, *jvtaylor@server1*

```
richmond@manageNode:~$ ssh richmond@192.168.56.109
The authenticity of host '192.168.56.109 (192.168.56.109)' can't be established.
ED25519 key fingerprint is SHA256:hEwbfcG/PXhxYvSAOWRArkPnZvJmTeKeKx9F0N8SI.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.56.109' (ED25519) to the list of known host
s.
```

```
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.
```

```
richmond@ControlNode1:~$
```

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

```
richmond@ControlNode1:~$  
logout  
Connection to 192.168.56.109 closed.
```

3. Do the same for Server 2.

```
richmond@manageNode:~$ ssh richmond@192.168.56.110  
The authenticity of host '192.168.56.110 (192.168.56.110)' can't be established  
.  
ED25519 key fingerprint is SHA256:/tLCTTuhXJZv9v28e6JwsNQ8xTFBmIwMX+tMWXGQWPM.  
This key is not known by any other names  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added '192.168.56.110' (ED25519) to the list of known host  
s.
```

```
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.
```

```
richmond@ControlNode2:~$
```

```
richmond@ControlNode2:~$  
logout  
Connection to 192.168.56.110 closed.  
richmond@manageNode:~$
```

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.

```
GNU nano 6.2 /etc/hosts *  
127.0.0.1    manageNode  
192.168.56.109 controlNode1  
192.168.56.110 controlNode2
```

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 jvtaylor@server1*. Enter the password when prompted. Verify that you have entered Server 1. Do the same for Server 2.

```
richmond@manageNode:~$ ssh richmond@controlNode1
The authenticity of host 'controlNode1 (192.168.56.109)' can't be established.
ED25519 key fingerprint is SHA256:hEwbfcG/PXhxYvSAOWRArkPpINZvJmTeKeKx9F0N8SI.
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 'controlNode1' (ED25519) to the list of known hosts.
richmond@controlNode1'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

Last login: Tue Aug 15 17:26:24 2023 from 192.168.56.108
```

```
richmond@manageNode:~$ ssh richmond@controlNode2
The authenticity of host 'controlNode2 (192.168.56.110)' can't be established.
ED25519 key fingerprint is SHA256:/tLCTTuhXJZv9v28e6JwsNQBxTFBmlwMX+tmWXGQWPM.
This host key is known by the following other names/addresses:
  ~/.ssh/known_hosts:4: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'controlNode2' (ED25519) to the list of known hosts.
richmond@controlNode2'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

Last login: Tue Aug 15 17:28:41 2023 from 192.168.56.108
```

Reflections:

In this activity, I learned how to clone the virtual machines and make a host and client servers in the virtual machines. Due to its systematic process, I get confused sometimes on what is happening in my work but overall I still manage to finish the task and setup the servers that can ping anyone in the same network.

Answer the following:

1. How are we able to use the hostname instead of IP address in SSH commands?
 - We are able to use the hostname instead of the ip address by configuring it into the nano command where we have to set up the ip address of the hostname to simply switch into servers using the hostnames.
2. How secured is SSH?
 - it is highly secured as it has a remote login where the user must enter the password first to access the server. It uses encryption and authentication via public key cryptography.