

Exploring VMware Network Configurations

Group A

Computer System Technician, Loyalist College in Toronto

2025S-T4 COMP2013 - Virtualization Architecture 01 (P00 Group 1)

Maziar Sojoudian

July 18, 2025

Introduction

As we received this assignment the first time we assumed that it is just another assignment where we have to set up a VM and then just execute some commands to succeed. However, after we got into the settings of the various network modalities in VMware Workstation Pro of NAT, Bridged, and Host-only, we discovered that this was much more than simply making the toggle. It was linked to the concepts of virtual machines and how they interrelate with a variety of network settings, what it essentially amounts to in real life situations.

All of us selected one mode of on-going networks to examine. It began in a modest way: remove the adapter, restart the VM, launch a couple of ping. However, rather soon we were on our way to learning much more than we anticipated, how IPs are allocated, what is DHCP in the background, and how the VMs are isolated or exposed in various modes. Truth be spoken, we even experienced some incidences of why this ping is not coming only to find that we are actually undertaking basic troubleshooting.

The most entertaining (yes, entertaining) aspect of this assignment was the way each mode caused the VMs to behave in the way they did with one another, the host and external world. The host allowed us to have a controlled connection to the internet thanks to NAT. Bridged made the VMs appear as actual gadgets on the identical Wi-Fi network. Host-only put everything into a closed proprietary little testing bubble. It was attempting various versions of reality, within a computer.

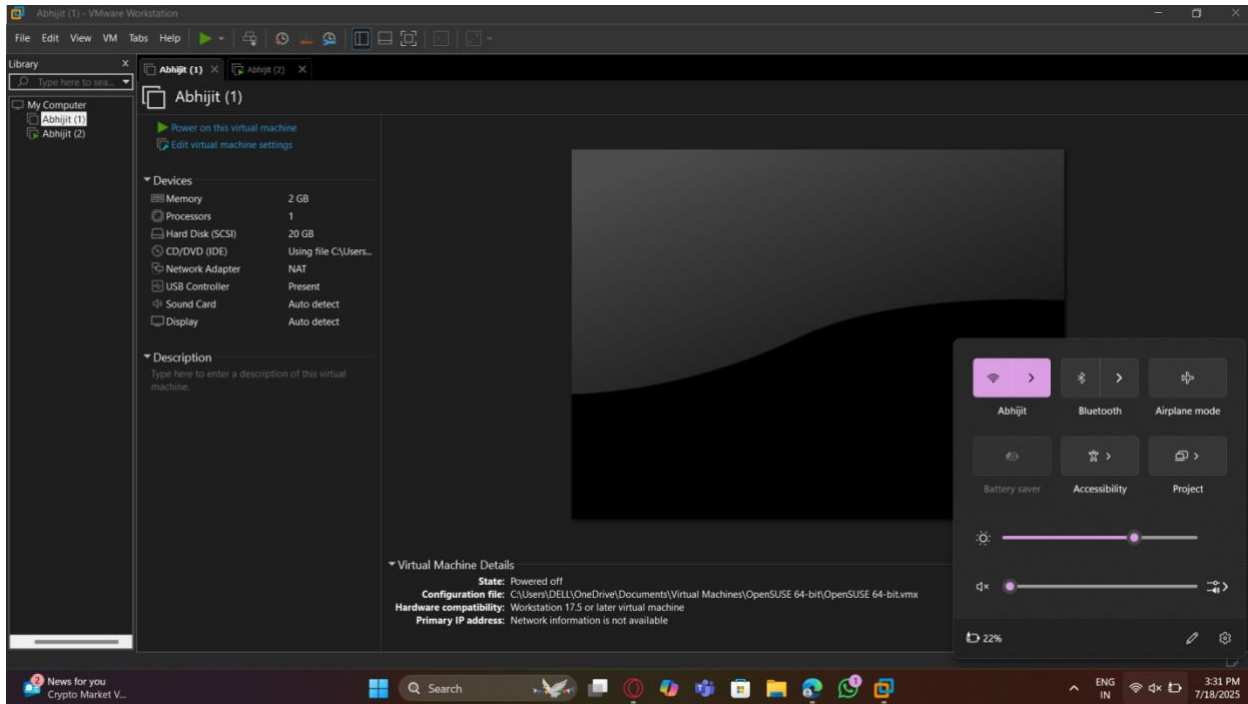
It was not only running commands, but it was linking theory and practice. It made us understand which type of network mode you should apply in practice, whether you are a developer, a test person or a barrier who is working on secure labs. It was also a reminder of how networking impacts everything in IT even when something is made up of a virtual layer.

Objective:

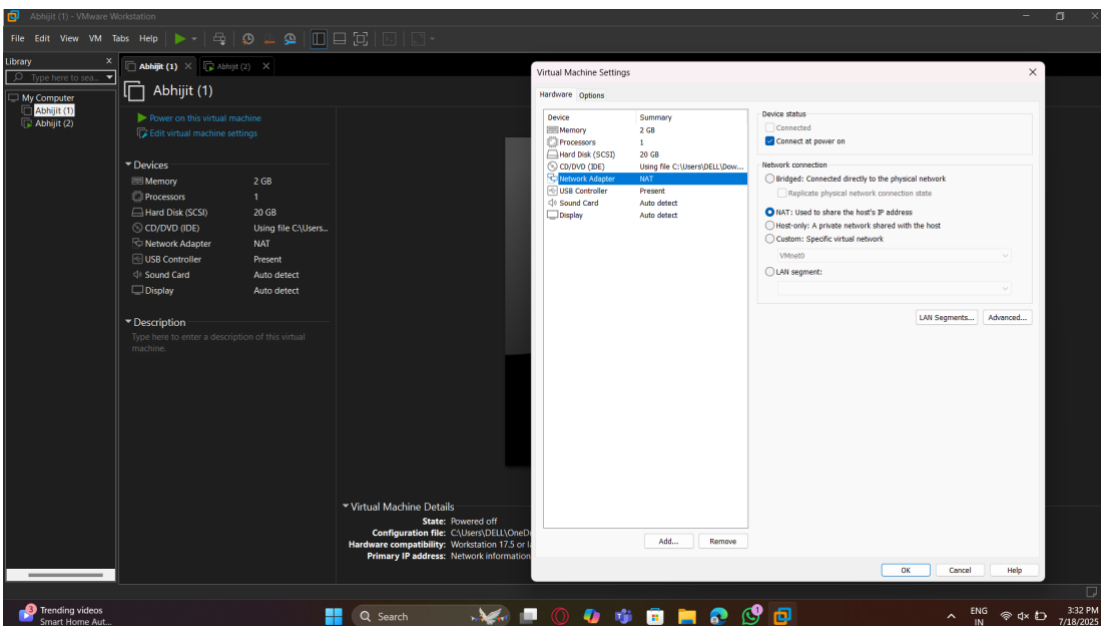
To gain hands-on experience with different network modes in VMware Workstation Pro and understand how virtual machines (VMs) interact with each other and the external network in each mode. Students will collaboratively configure VMs using NAT, Bridged, and Host-only networks, verify connectivity, and reflect on similarities, differences, and best-use scenarios.

Part 1 – NAT Configuration (Abhijit Singh 500235457)

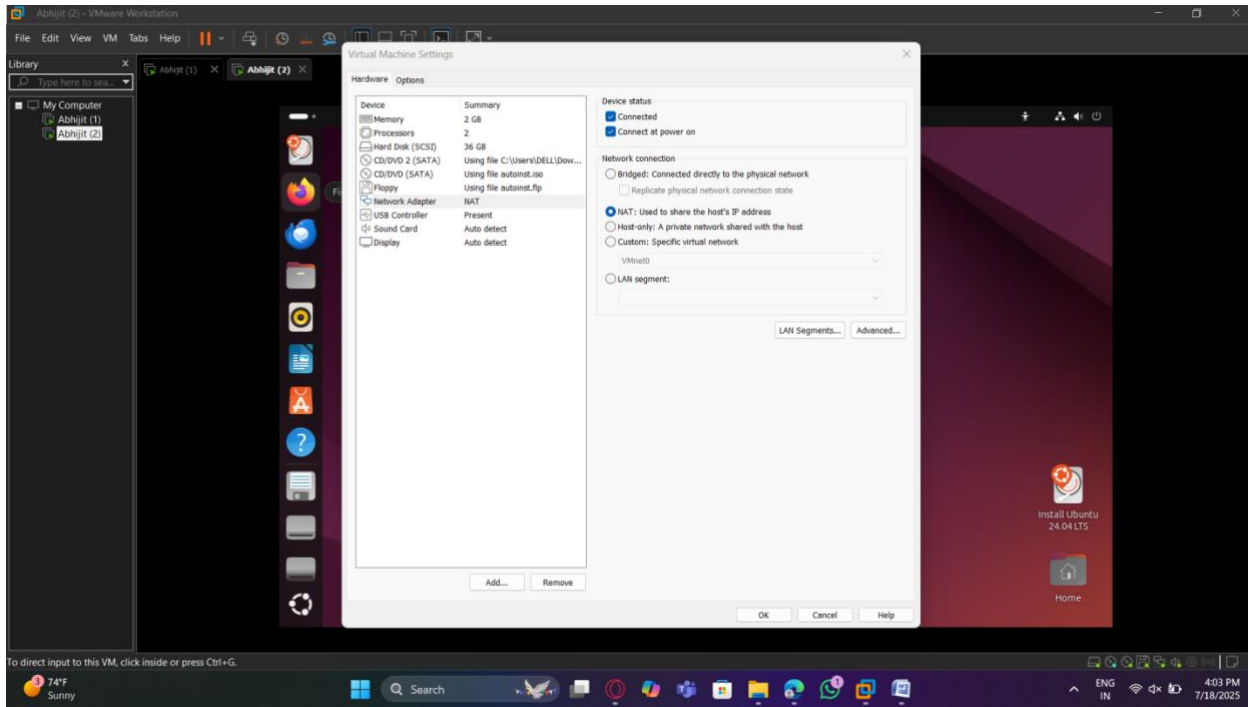
Connecting to Personal Hotspot to prevent campus network security and Ip conflicts



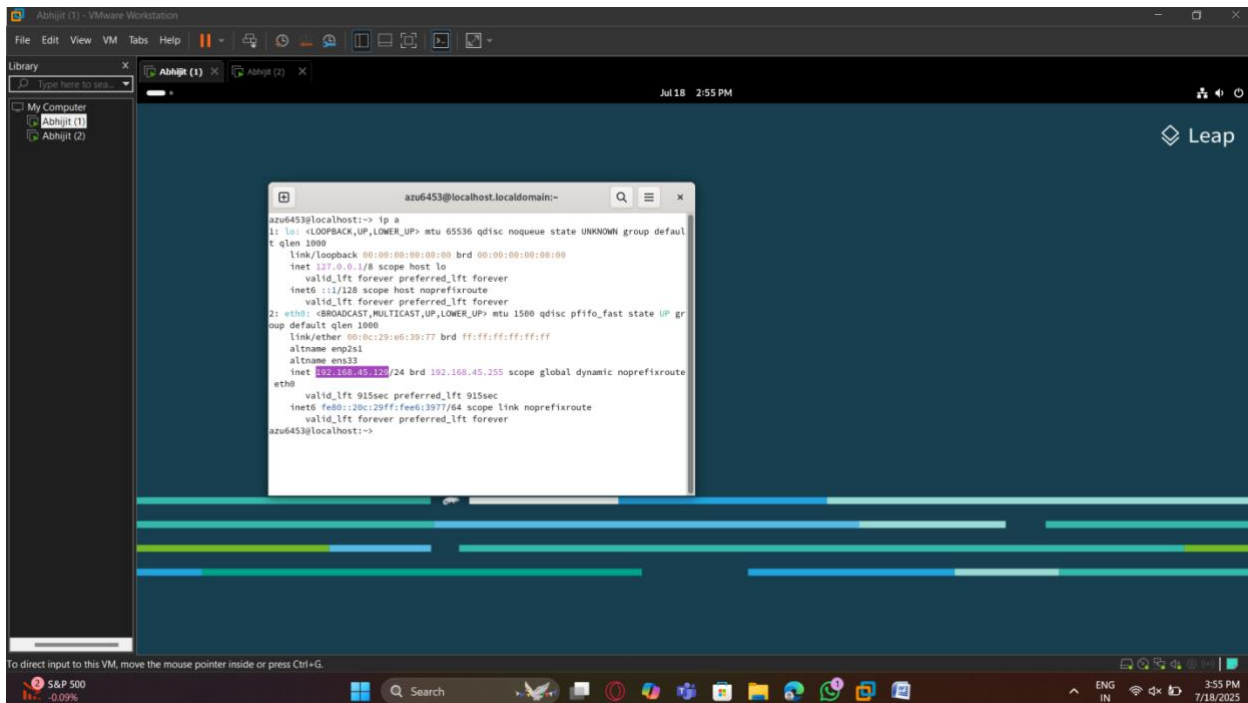
In the image below we enable NAT network connection by going to the vm's network adapter settings



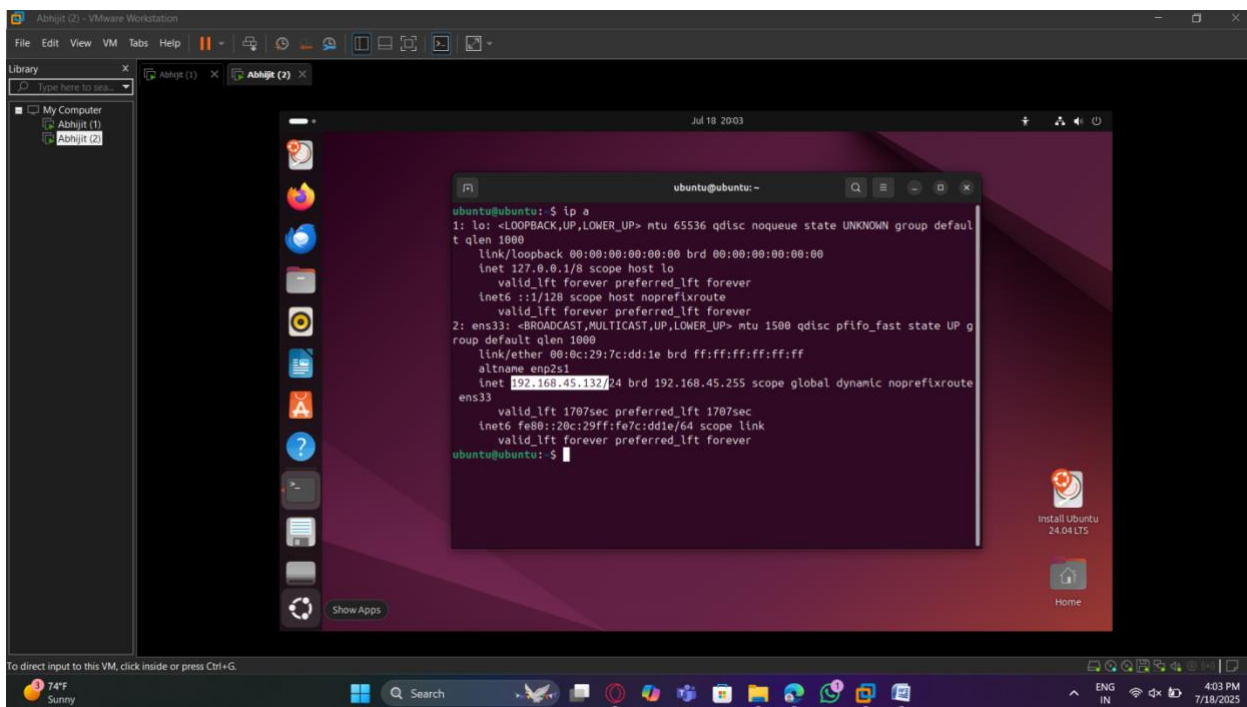
Similarly for the second virtual machine



Here, we use the “Ip a ” command in openSUSE terminal to display the first vm’s Ip address “192.168.45.129”

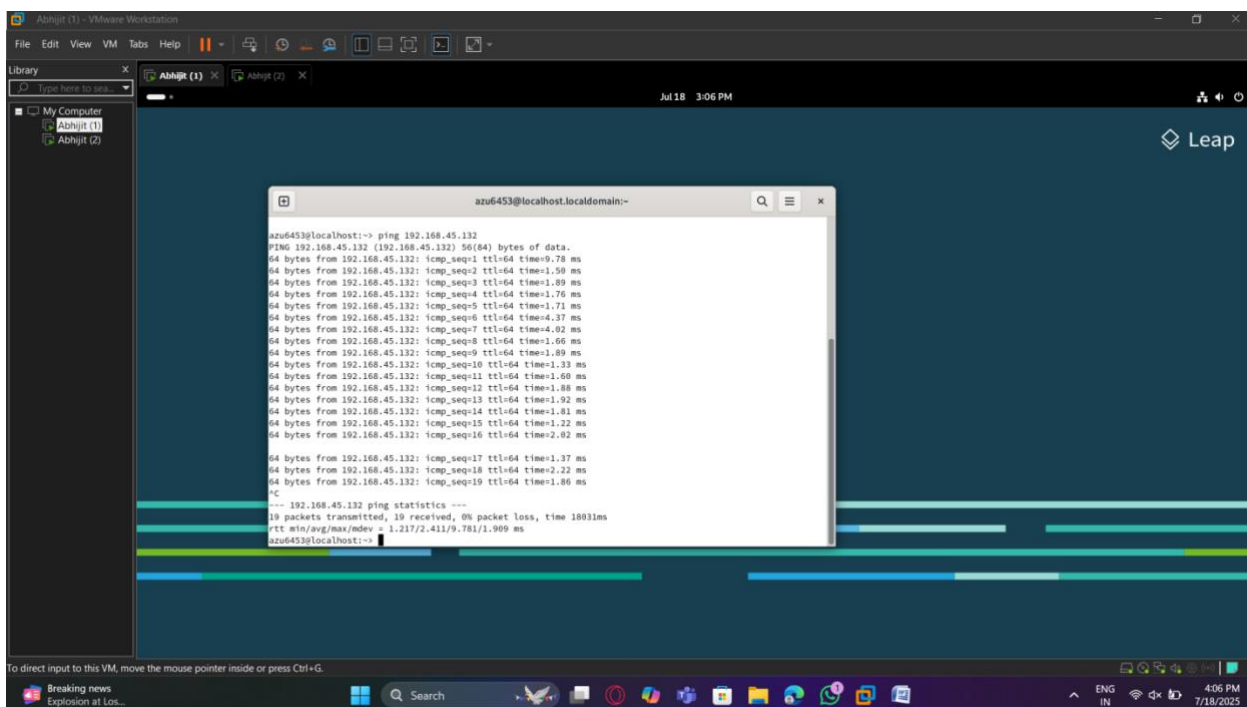


Similarly, using the “Ip a” command to display the Ip address for the second virtual machine which is ubuntu “192.168.45.132”

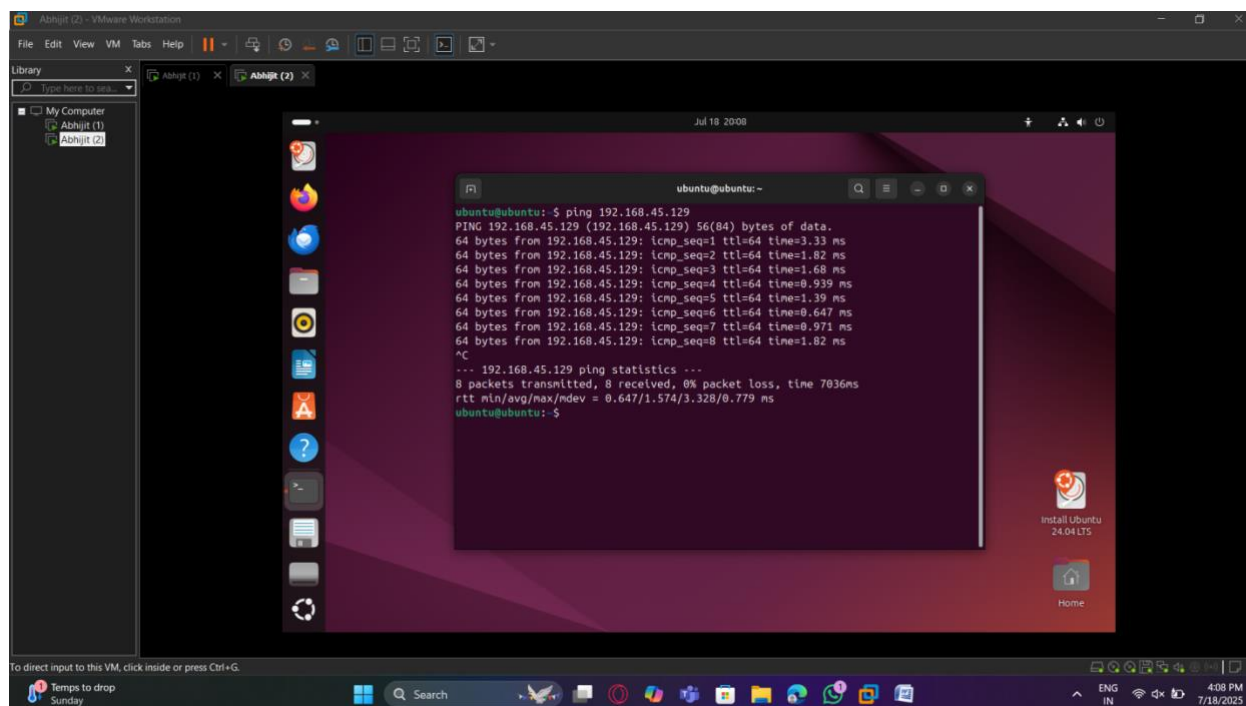


Testing Connectivity

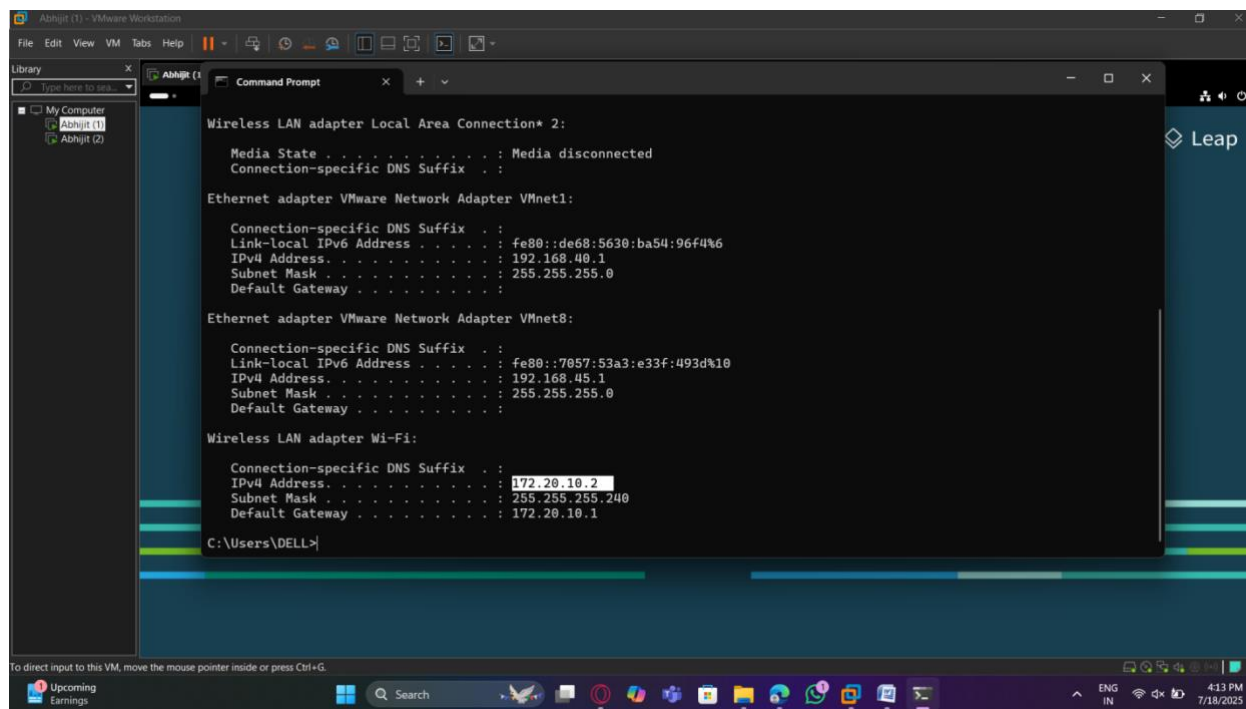
We start by testing connectivity between the two virtual machines, we do that by first going to the openSUSE terminal and using the “ping 192.168.45.132”, the Ip address used here was that of the ubuntu vm. As seen in the picture, the ping was received and successful.



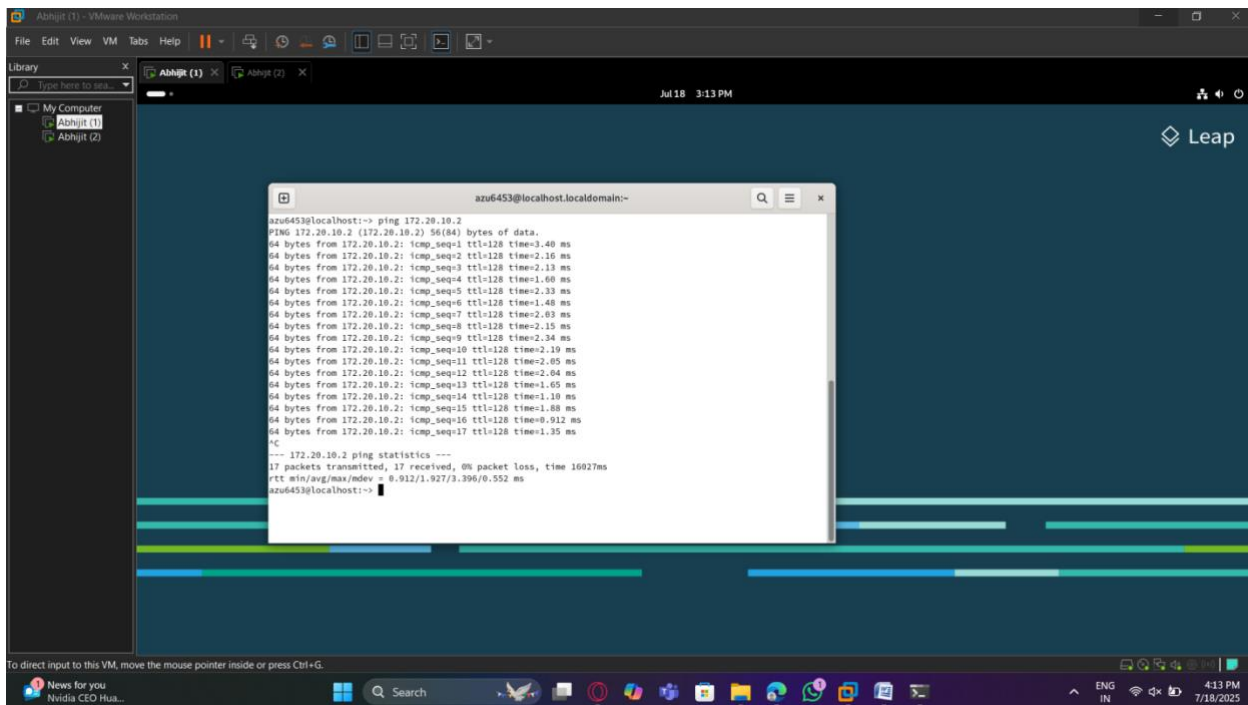
Similarly, in the second vm, the one with ubuntu, we open the terminal and use the “ping 192.168.45.129” Command, in which the Ip address is that of the openSUSE vm. As seen below, the ping was received and successful.



The image below displays the Ip address of the host machine, the Ip address is highlighted and the command we used was “ipconfig”



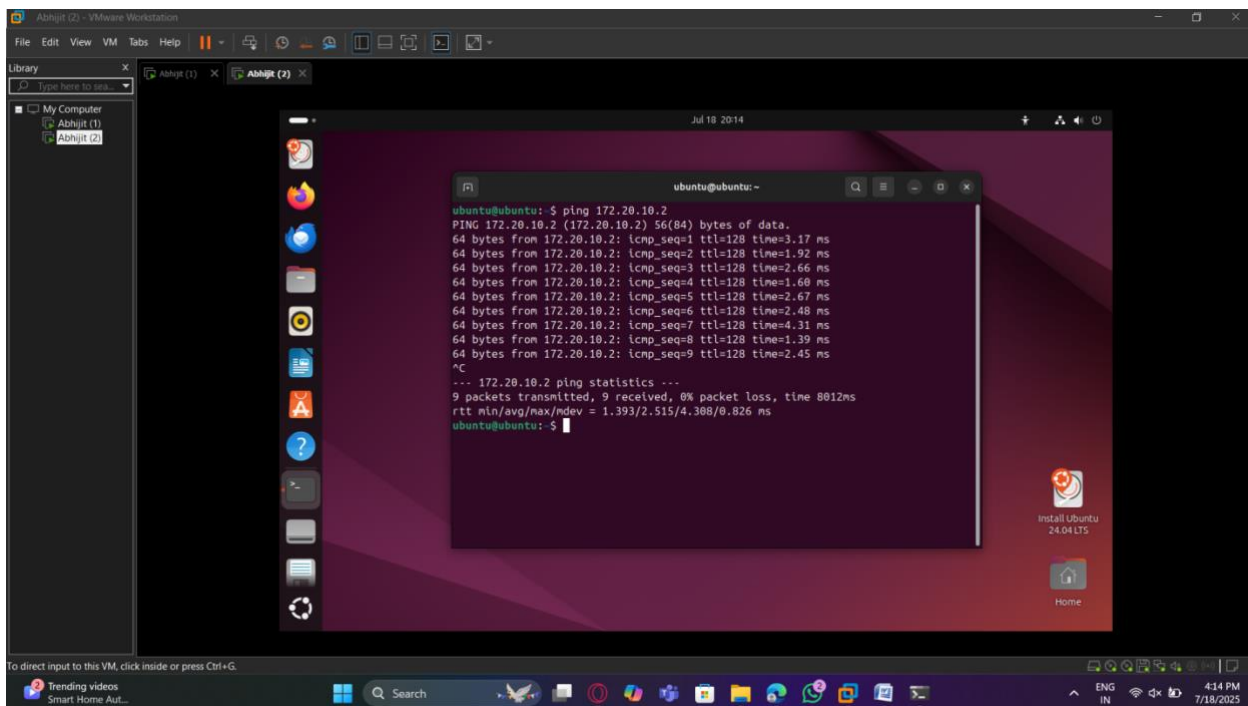
Now, we test connectivity between the openSUSE vm and the host, to do so we use the “ping” command followed by the host’s Ip address “172.20.10.2”, the result as seen below was successful



```

abzu6453@localhost:~$ ping 172.20.10.2
PING 172.20.10.2 (172.20.10.2) 56(84) bytes of data.
64 bytes from 172.20.10.2: icmp_seq=1 ttl=128 time=3.40 ms
64 bytes from 172.20.10.2: icmp_seq=2 ttl=128 time=2.16 ms
64 bytes from 172.20.10.2: icmp_seq=3 ttl=128 time=2.13 ms
64 bytes from 172.20.10.2: icmp_seq=4 ttl=128 time=1.60 ms
64 bytes from 172.20.10.2: icmp_seq=5 ttl=128 time=2.33 ms
64 bytes from 172.20.10.2: icmp_seq=6 ttl=128 time=1.48 ms
64 bytes from 172.20.10.2: icmp_seq=7 ttl=128 time=2.03 ms
64 bytes from 172.20.10.2: icmp_seq=8 ttl=128 time=2.15 ms
64 bytes from 172.20.10.2: icmp_seq=9 ttl=128 time=2.34 ms
64 bytes from 172.20.10.2: icmp_seq=10 ttl=128 time=2.19 ms
64 bytes from 172.20.10.2: icmp_seq=11 ttl=128 time=2.05 ms
64 bytes from 172.20.10.2: icmp_seq=12 ttl=128 time=2.04 ms
64 bytes from 172.20.10.2: icmp_seq=13 ttl=128 time=1.65 ms
64 bytes from 172.20.10.2: icmp_seq=14 ttl=128 time=1.10 ms
64 bytes from 172.20.10.2: icmp_seq=15 ttl=128 time=1.60 ms
64 bytes from 172.20.10.2: icmp_seq=16 ttl=128 time=0.912 ms
64 bytes from 172.20.10.2: icmp_seq=17 ttl=128 time=1.35 ms
^C
--- 172.20.10.2 ping statistics ---
17 packets transmitted, 17 received, 0% packet loss, time 1602ms
rtt min/avg/max/mdev = 0.912/1.927/3.396/0.552 ms
abzu6453@localhost:~$
  
```

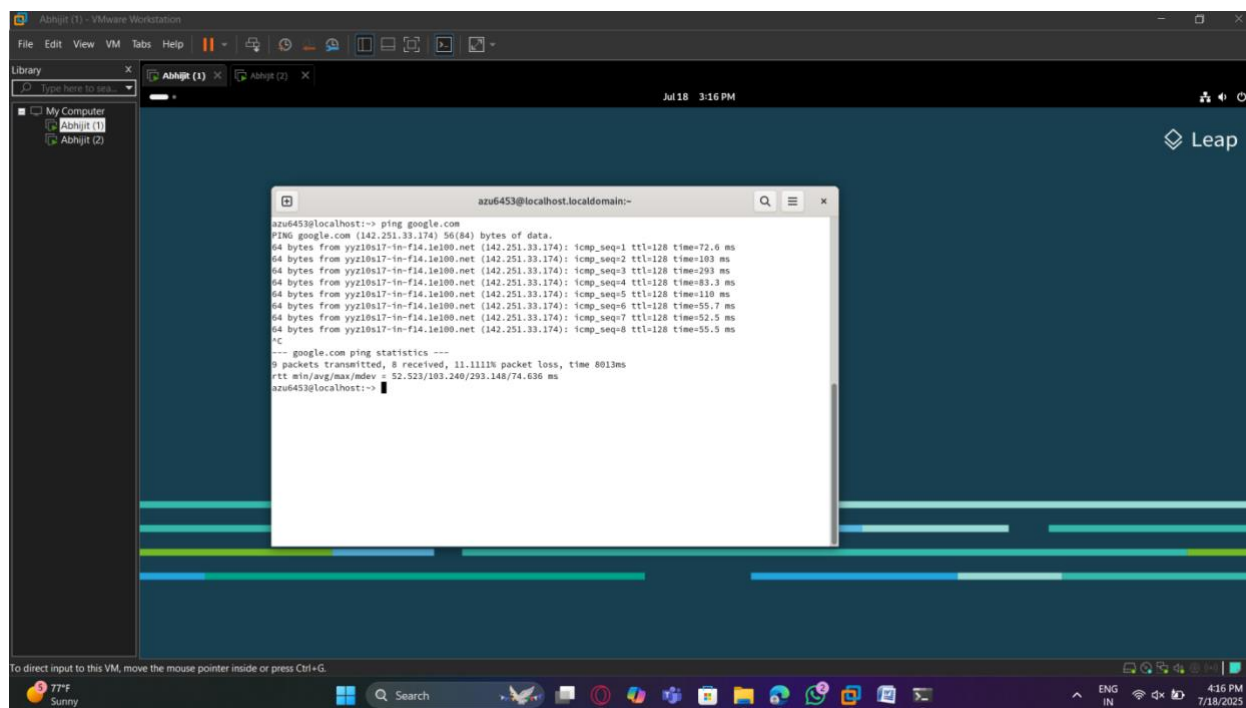
Here, we use the “ping 172.20.10.2” command once again, in the ubuntu vm’s terminal to test the connectivity between vm 2 and the host, which resulted in success



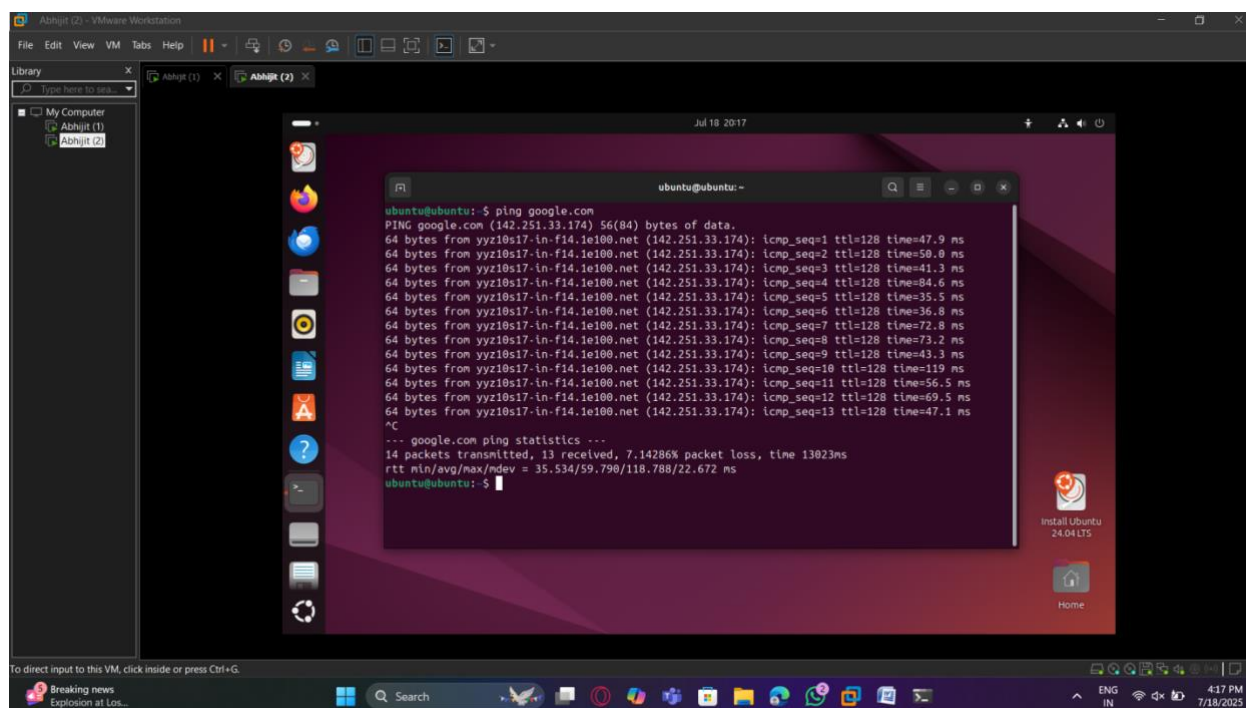
```

ubuntu@ubuntu:~$ ping 172.20.10.2
PING 172.20.10.2 (172.20.10.2) 56(84) bytes of data.
64 bytes from 172.20.10.2: icmp_seq=1 ttl=128 time=3.17 ms
64 bytes from 172.20.10.2: icmp_seq=2 ttl=128 time=1.92 ms
64 bytes from 172.20.10.2: icmp_seq=3 ttl=128 time=2.66 ms
64 bytes from 172.20.10.2: icmp_seq=4 ttl=128 time=1.60 ms
64 bytes from 172.20.10.2: icmp_seq=5 ttl=128 time=2.67 ms
64 bytes from 172.20.10.2: icmp_seq=6 ttl=128 time=2.48 ms
64 bytes from 172.20.10.2: icmp_seq=7 ttl=128 time=4.31 ms
64 bytes from 172.20.10.2: icmp_seq=8 ttl=128 time=1.39 ms
64 bytes from 172.20.10.2: icmp_seq=9 ttl=128 time=2.45 ms
^C
--- 172.20.10.2 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8012ms
rtt min/avg/max/mdev = 1.393/2.515/4.388/0.826 ms
ubuntu@ubuntu:~$
  
```


Here, we test the connectivity between the openSUSE vm and the internet by using the “ping google.com” command to prove that the vm can send packets out to the internet through the host machine, and the image below shows a successful ping



For the second time, in order to prove network connectivity in the ubuntu vm, we use the “ping google.com” command, which resulted in success



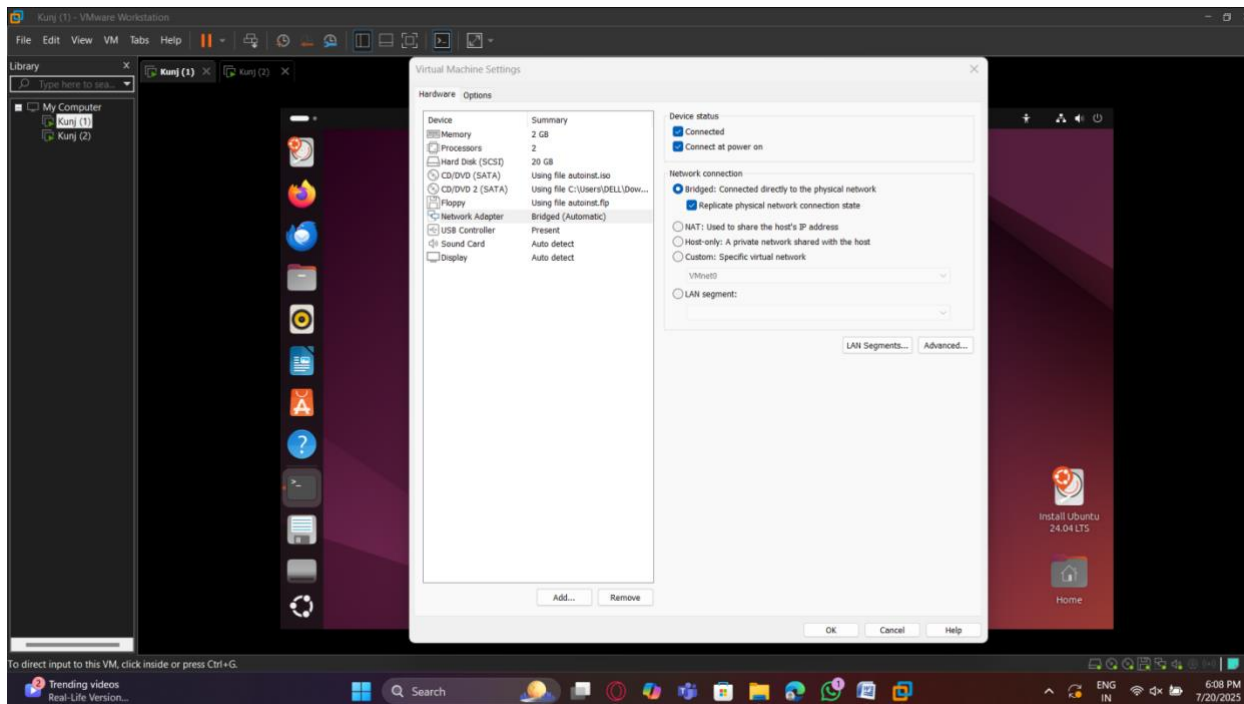
Reflections and Learnings

Using NAT configuration in VMware showed me how network translation takes place in a virtual network. It turns out that having two VMs in the NAT mode meant that each of them was given a local IP address and they continued to share the network connection of the host to access the internet. The communication test between the two VMs proved that the two VMs were able to communicate since VMware creates an internal NAT network behind the virtual NAT device. The connectivity test with the host machine showed that NAT is correctly forwarding the traffic between the host and VMs. Pinging google.com was also critical since it did not only test outbound connectivity, but also ensured that DNS resolution was functioning properly.

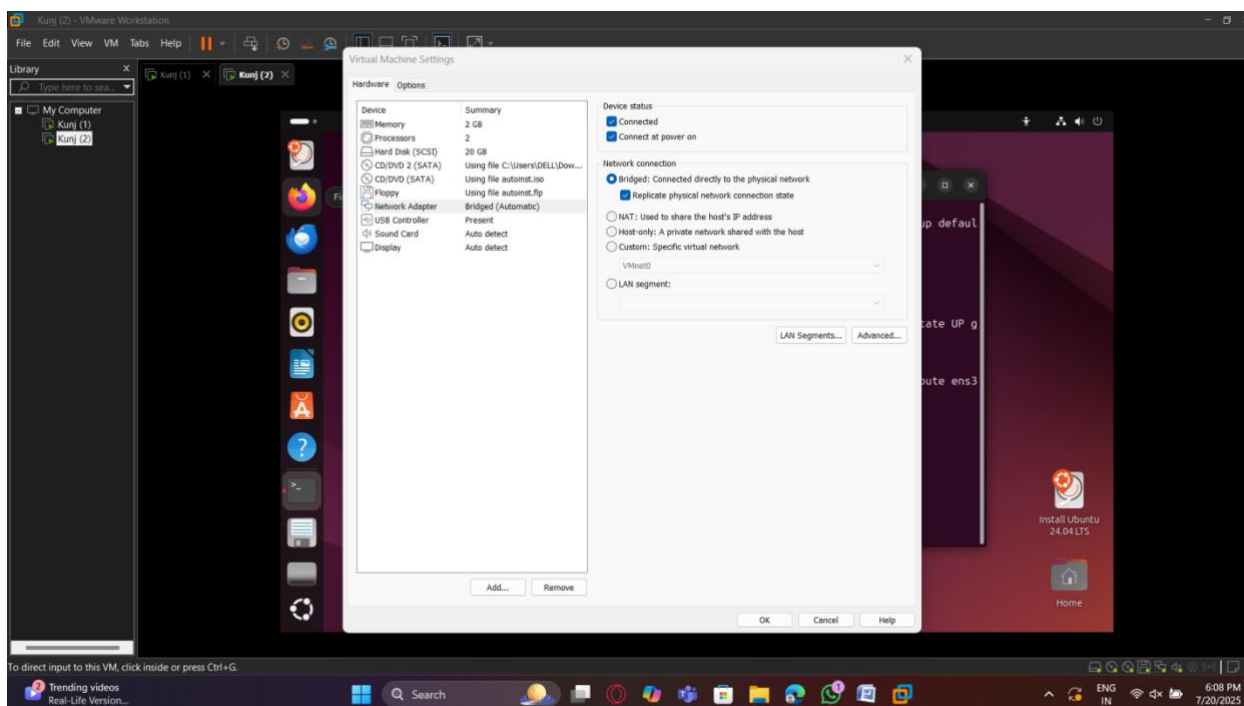
This practice has taught me the need to be structured in the trouble-shooting of network arrangements. Testing the connectivity serially, VM to VM, VM to host then VM to the internet, allowed me to research about packets and how they travel across the layers. I also got an idea on why NAT mode is preferred widely, it makes internet access easy to VMs but also adds security since, internal IPs will be blocked by the host. Nonetheless, I also came to know that NAT allows only outbound traffic by default and thus when there is a need to have an inbound access, port forwarding must be set up. On the whole, this exercise provided me with some real-life experience with IP addressing and simple tools such as ping to enable confirmation of networks.

Part 2 – Bridged Configuration (Kunj Patel 500223944)

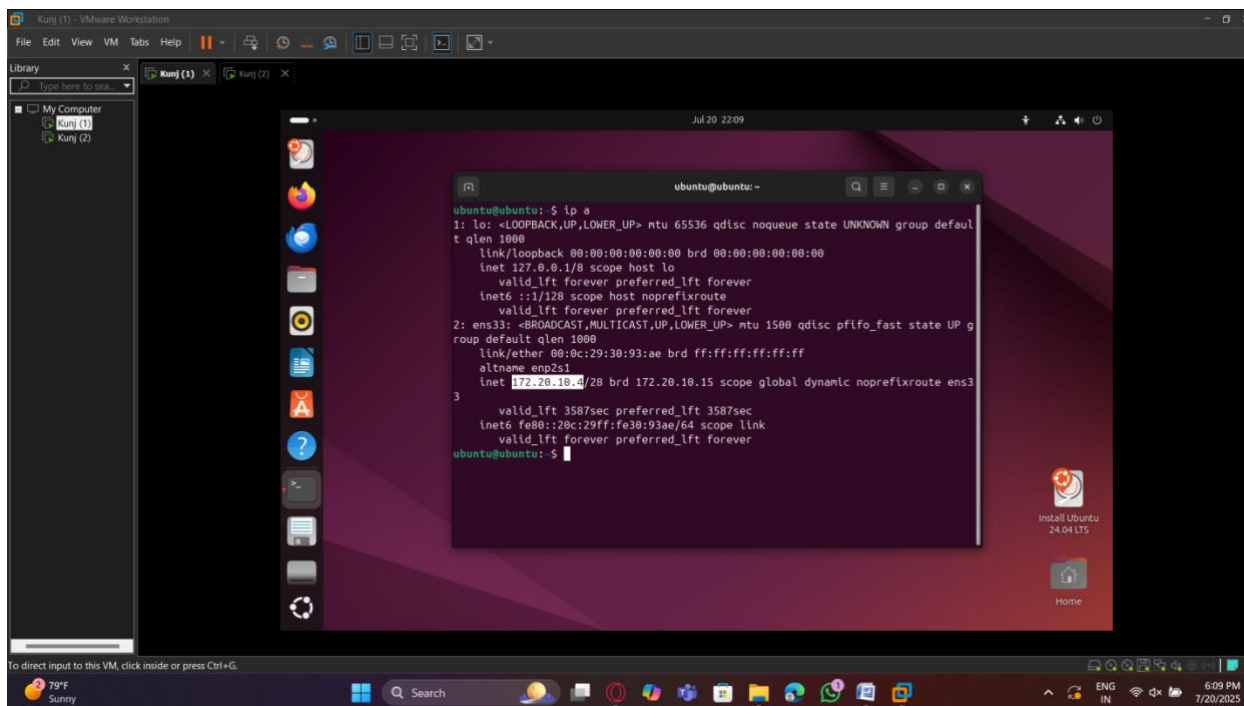
Here, we create two ubuntu virtual machines and set the network adapter the settings to Bridged mode



Similarly, for the second vm



Below, we use the “Ip a” command to display the Ip address of vm 1 “172.20.10.4”

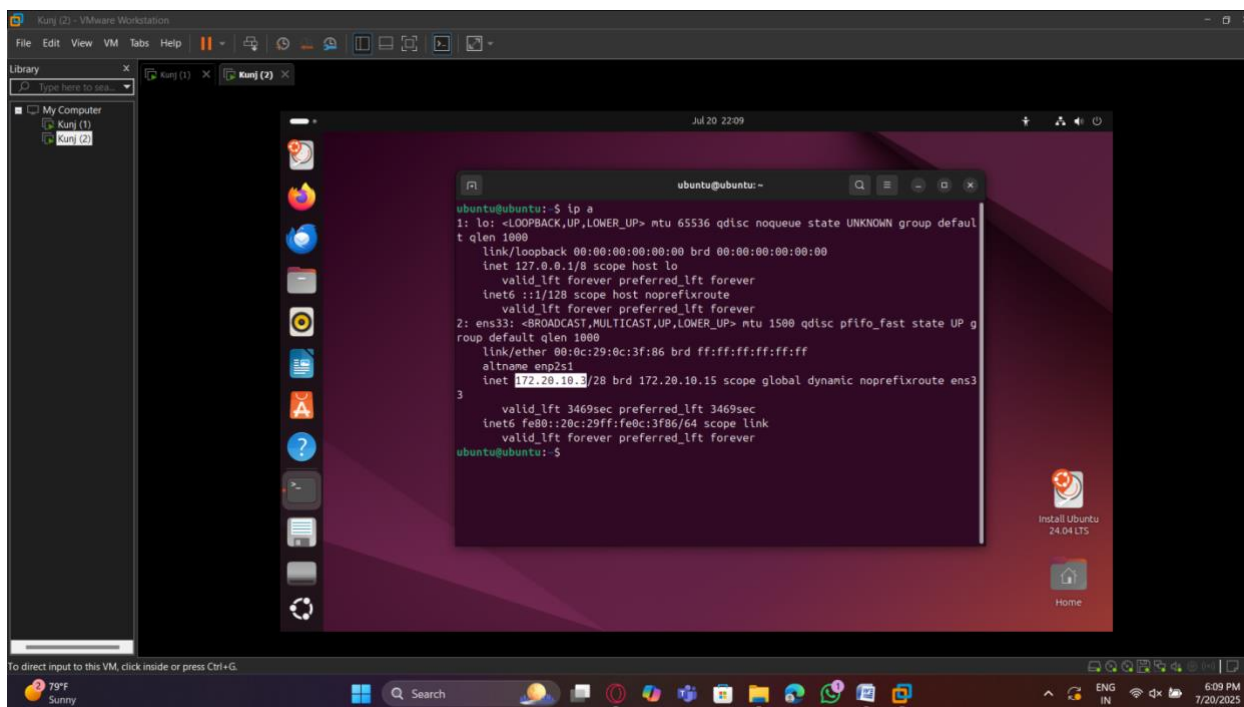


```

Kunj (1) - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Kunj (1)
Kunj (2)
Jul 20 22:09
ubuntu@ubuntu:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP g
    link/ether 00:0c:29:30:93:ae brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 172.20.10.4/28 brd 172.20.10.15 scope global dynamic noprefixroute ens3
        valid_lft 3587sec preferred_lft 3587sec
    inet6 fe80::20c:29ff:fe30:93ae/64 scope link
        valid_lft forever preferred_lft forever
ubuntu@ubuntu:~$

```

Below, we use the “Ip a” command to display the Ip address of vm 2 “172.20.10.3”



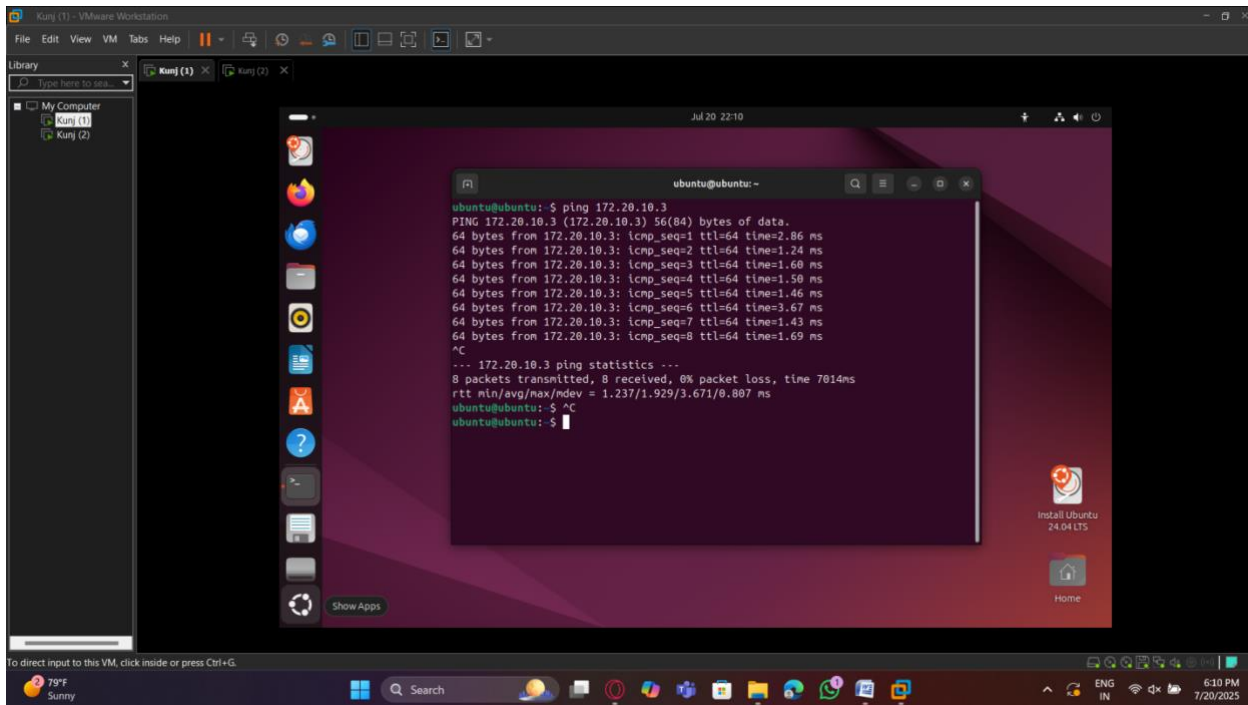
```

Kunj (2) - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Kunj (1)
Kunj (2)
Jul 20 22:09
ubuntu@ubuntu:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP g
    link/ether 00:0c:29:0c:3f:86 brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 172.20.10.3/28 brd 172.20.10.15 scope global dynamic noprefixroute ens3
        valid_lft 3469sec preferred_lft 3469sec
    inet6 fe80::20c:29ff:fe0c:3f86/64 scope link
        valid_lft forever preferred_lft forever
ubuntu@ubuntu:~$

```

Testing Connectivity

Here, we use the “ping” command to test connectivity between the two vm’s from vm 1, the result is successful as seen below



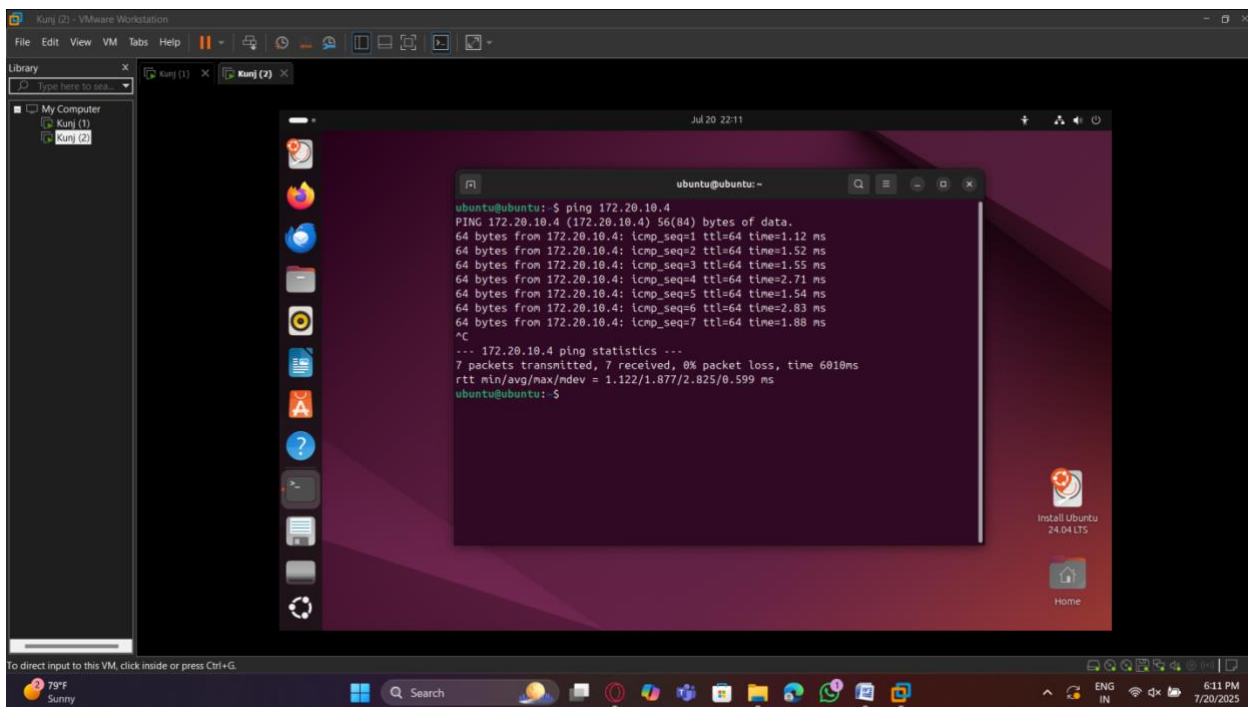
```

Kunj (1) - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Kunj (1)
Kunj (2)

ubuntu@ubuntu:~
Jul 20 22:10
ubuntu@ubuntu:~$ ping 172.20.10.3
PING 172.20.10.3 (172.20.10.3) 56(84) bytes of data:
64 bytes from 172.20.10.3: icmp_seq=1 ttl=64 time=2.86 ms
64 bytes from 172.20.10.3: icmp_seq=2 ttl=64 time=1.24 ms
64 bytes from 172.20.10.3: icmp_seq=3 ttl=64 time=1.60 ms
64 bytes from 172.20.10.3: icmp_seq=4 ttl=64 time=1.50 ms
64 bytes from 172.20.10.3: icmp_seq=5 ttl=64 time=1.46 ms
64 bytes from 172.20.10.3: icmp_seq=6 ttl=64 time=3.67 ms
64 bytes from 172.20.10.3: icmp_seq=7 ttl=64 time=1.43 ms
64 bytes from 172.20.10.3: icmp_seq=8 ttl=64 time=1.69 ms
^C
--- 172.20.10.3 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7014ms
rtt min/avg/max/mdev = 1.237/1.929/3.671/0.807 ms
ubuntu@ubuntu:~$ ^C
ubuntu@ubuntu:~$

```

Here, we use the “ping” command to test connectivity between the two vm’s from vm2, the result is successful as seen below



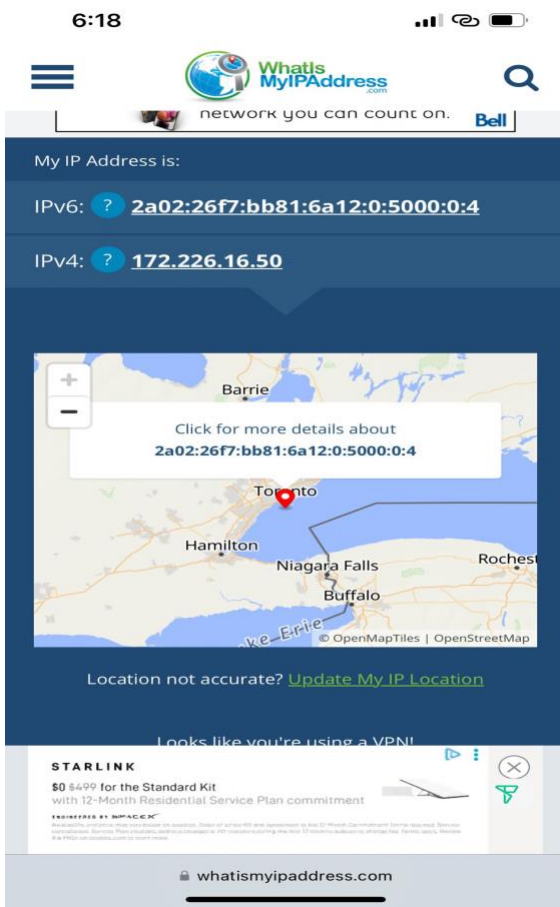
```

Kunj (2) - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Kunj (1)
Kunj (2)

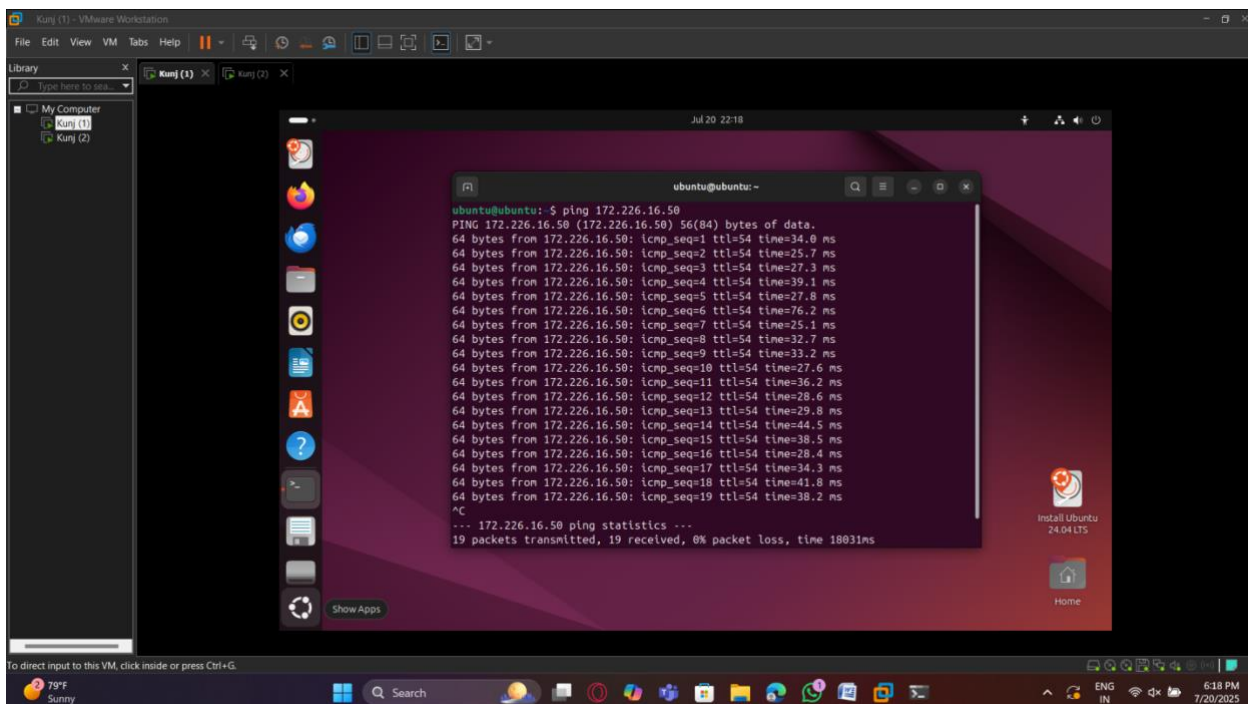
ubuntu@ubuntu:~
Jul 20 22:11
ubuntu@ubuntu:~$ ping 172.20.10.4
PING 172.20.10.4 (172.20.10.4) 56(84) bytes of data:
64 bytes from 172.20.10.4: icmp_seq=1 ttl=64 time=1.12 ms
64 bytes from 172.20.10.4: icmp_seq=2 ttl=64 time=1.52 ms
64 bytes from 172.20.10.4: icmp_seq=3 ttl=64 time=1.55 ms
64 bytes from 172.20.10.4: icmp_seq=4 ttl=64 time=2.71 ms
64 bytes from 172.20.10.4: icmp_seq=5 ttl=64 time=1.54 ms
64 bytes from 172.20.10.4: icmp_seq=6 ttl=64 time=2.83 ms
64 bytes from 172.20.10.4: icmp_seq=7 ttl=64 time=1.88 ms
^C
--- 172.20.10.4 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6010ms
rtt min/avg/max/mdev = 1.122/1.877/2.825/0.599 ms
ubuntu@ubuntu:~$

```

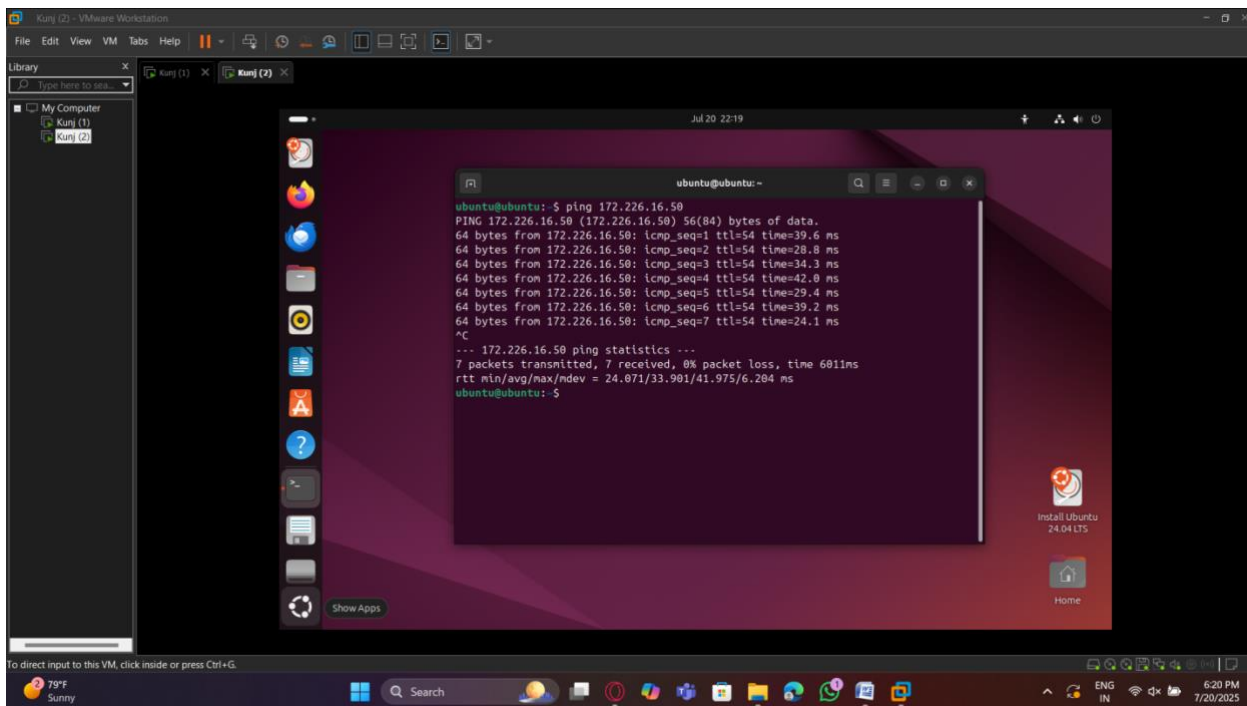
The image below displays the Ip address of a mobile phone connected to the same network as the vm “172.226.16.50”



Here, we use the “ping” command from vm1 to test it’s connectivity with the mobile phone which is on the same network, the result is successful as seen below



Here, we ping the mobile phone again, but this time we do it from vm2, the result is successful as seen below

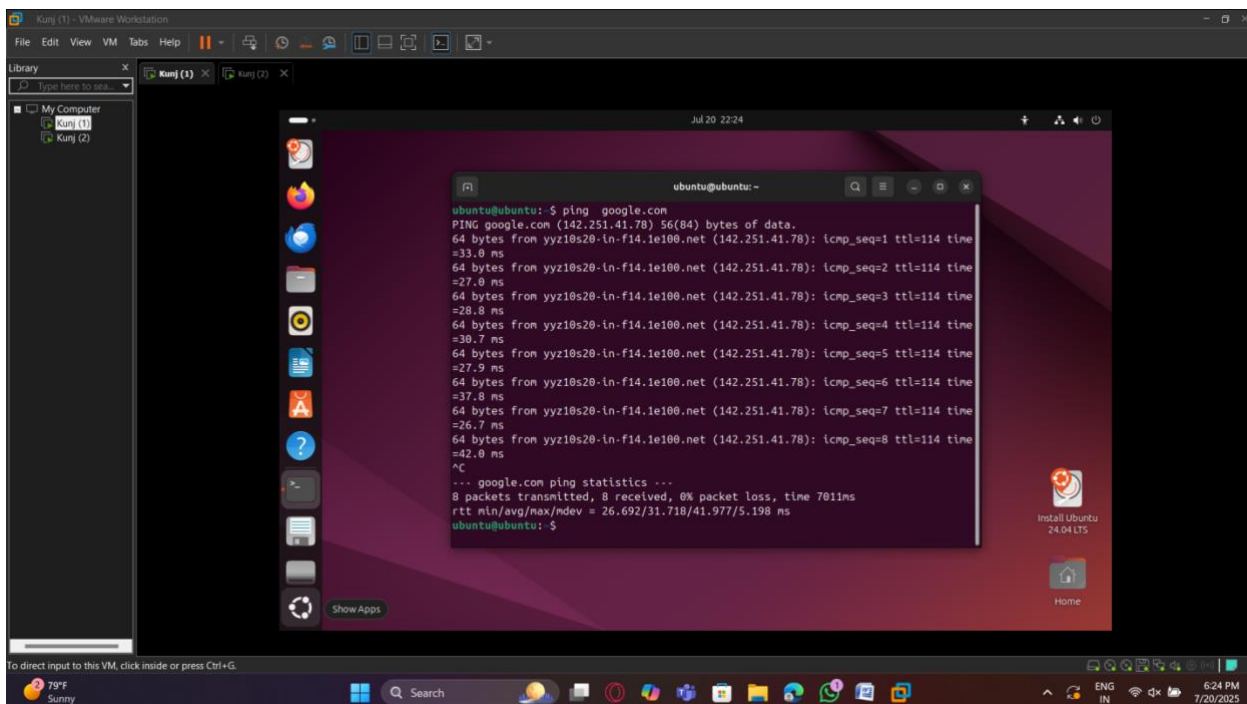


```

Kung (2) - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search...
My Computer
Kung (1)
Kung (2)
Kung (2)
Jul 20 22:19
ubuntu@ubuntu:~$ ping 172.226.16.50
PING 172.226.16.50 (172.226.16.50) 56(84) bytes of data:
64 bytes from 172.226.16.50: icmp_seq=1 ttl=54 time=39.6 ms
64 bytes from 172.226.16.50: icmp_seq=2 ttl=54 time=38.8 ms
64 bytes from 172.226.16.50: icmp_seq=3 ttl=54 time=34.3 ms
64 bytes from 172.226.16.50: icmp_seq=4 ttl=54 time=42.0 ms
64 bytes from 172.226.16.50: icmp_seq=5 ttl=54 time=29.4 ms
64 bytes from 172.226.16.50: icmp_seq=6 ttl=54 time=39.2 ms
64 bytes from 172.226.16.50: icmp_seq=7 ttl=54 time=24.1 ms
^C
--- 172.226.16.50 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 601ms
rtt min/avg/max/mdev = 24.071/33.901/41.975/6.204 ms
ubuntu@ubuntu:~$

```

Here, we test vm1's connectivity to the internet by using the "ping google.com" command, the result is successful as seen below

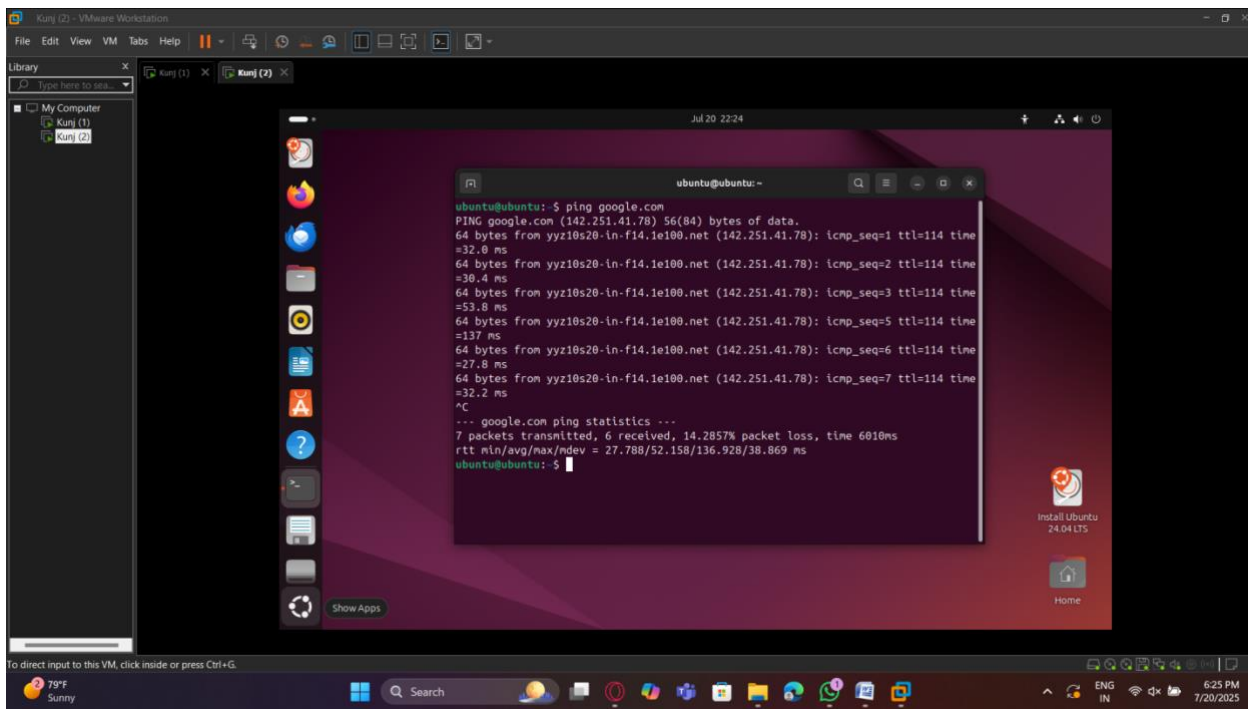


```

Kung (1) - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search...
My Computer
Kung (1)
Kung (2)
Kung (1)
Jul 20 22:24
ubuntu@ubuntu:~$ ping google.com
PING google.com (142.251.41.78) 56(84) bytes of data:
64 bytes from yyz10s20-ln-f14.1e100.net (142.251.41.78): icmp_seq=1 ttl=114 time=33.0 ms
64 bytes from yyz10s20-ln-f14.1e100.net (142.251.41.78): icmp_seq=2 ttl=114 time=27.0 ms
64 bytes from yyz10s20-ln-f14.1e100.net (142.251.41.78): icmp_seq=3 ttl=114 time=28.8 ms
64 bytes from yyz10s20-ln-f14.1e100.net (142.251.41.78): icmp_seq=4 ttl=114 time=30.7 ms
64 bytes from yyz10s20-ln-f14.1e100.net (142.251.41.78): icmp_seq=5 ttl=114 time=27.9 ms
64 bytes from yyz10s20-ln-f14.1e100.net (142.251.41.78): icmp_seq=6 ttl=114 time=37.8 ms
64 bytes from yyz10s20-ln-f14.1e100.net (142.251.41.78): icmp_seq=7 ttl=114 time=26.7 ms
64 bytes from yyz10s20-ln-f14.1e100.net (142.251.41.78): icmp_seq=8 ttl=114 time=42.0 ms
^C
--- google.com ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 701ms
rtt min/avg/max/mdev = 26.692/31.718/41.977/5.198 ms
ubuntu@ubuntu:~$

```


Here, we test vm2's connectivity to the internet by using the "ping google.com" command, the result is successful as seen below



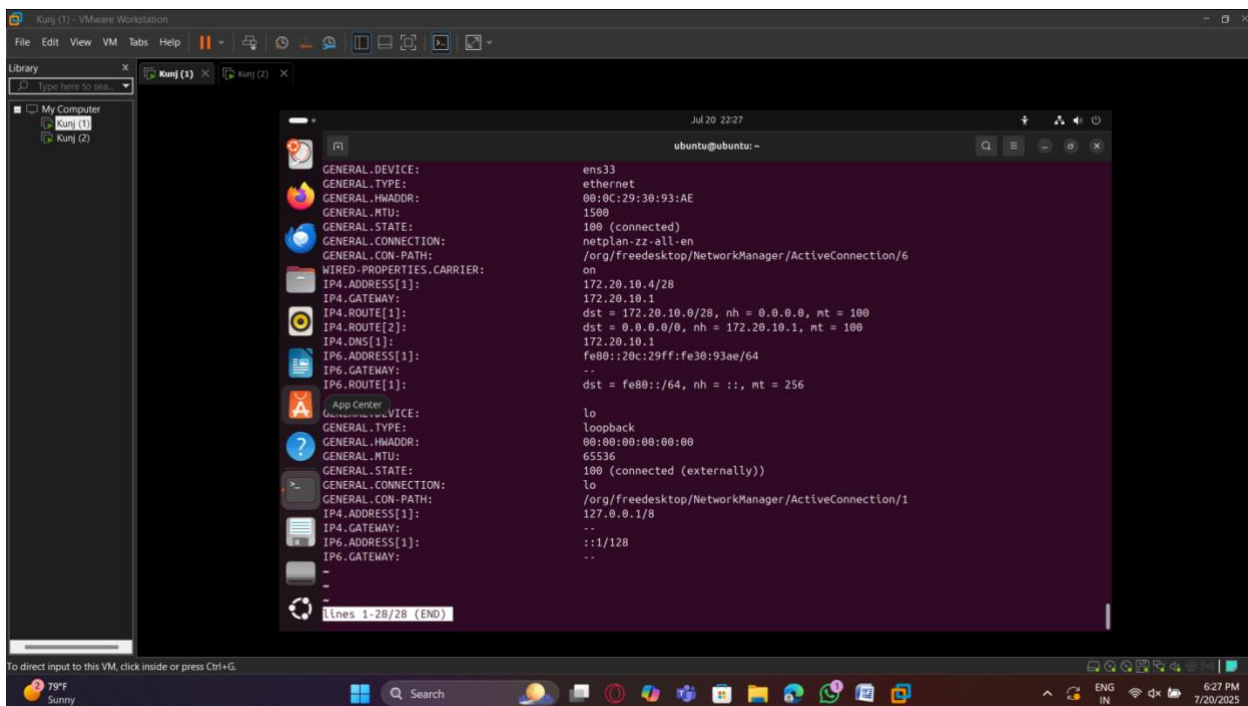
```

Kunji (2) - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Kunji (1)
Kunji (2)

Kunji (2)
Jul 20 22:24
ubuntu@ubuntu:~$ ping google.com
PING google.com (142.251.41.78) 56(84) bytes of data:
64 bytes from yyz10s20-in-f14.1e100.net (142.251.41.78): icmp_seq=1 ttl=114 time
=32.0 ms
64 bytes from yyz10s20-in-f14.1e100.net (142.251.41.78): icmp_seq=2 ttl=114 time
=30.4 ms
64 bytes from yyz10s20-in-f14.1e100.net (142.251.41.78): icmp_seq=3 ttl=114 time
=53.8 ms
64 bytes from yyz10s20-in-f14.1e100.net (142.251.41.78): icmp_seq=5 ttl=114 time
=137 ms
64 bytes from yyz10s20-in-f14.1e100.net (142.251.41.78): icmp_seq=6 ttl=114 time
=27.8 ms
64 bytes from yyz10s20-in-f14.1e100.net (142.251.41.78): icmp_seq=7 ttl=114 time
=32.2 ms
^C
--- google.com ping statistics ---
7 packets transmitted, 6 received, 14.2857% packet loss, time 6010ms
rtt min/avg/max/mdev = 27.788/52.158/136.928/38.869 ms
ubuntu@ubuntu:~$

```

Here, we document the DHCP Ip assignment and Interfaces on vm1



```

Kunji (1) - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Kunji (1)
Kunji (2)

Kunji (1)
Jul 20 22:27
ubuntu@ubuntu:~$ ip netns exec
GENERAL.DEVICE: ens33
GENERAL.TYPE: ethernet
GENERAL.HWADDR: 00:0C:29:30:93:AE
GENERAL.MTU: 1500
GENERAL.STATE: 100 (connected)
GENERAL.CONNECTION: netplan-zz-all-en
GENERAL.CON-PATH: /org/freedesktop/NetworkManager/ActiveConnection/6
WIRED-PROPERTIES.CARRIER: on
IP4.ADDRESS[1]: 172.20.10.4/28
IP4.GATEWAY: 172.20.10.1
IP4.ROUTE[1]: dst = 172.20.10.0/28, nh = 0.0.0.0, mt = 100
IP4.ROUTE[2]: dst = 0.0.0.0/0, nh = 172.20.10.1, mt = 100
IP4.DNS[1]: 172.20.10.1
IP6.ADDRESS[1]: fe80::20c:29ff:fe30:93ae/64
IP6.GATEWAY: --
IP6.ROUTE[1]: dst = fe80::/64, nh = ::, mt = 256
lo
GENERAL.DEVICE: lo
GENERAL.TYPE: loopback
GENERAL.HWADDR: 00:00:00:00:00:00
GENERAL.MTU: 65536
GENERAL.STATE: 100 (connected (externally))
GENERAL.CONNECTION: lo
GENERAL.CON-PATH: /org/freedesktop/NetworkManager/ActiveConnection/1
IP4.ADDRESS[1]: 127.0.0.1/8
IP4.GATEWAY: --
IP6.ADDRESS[1]: ::1/128
IP6.GATEWAY: --
lines 1-28/28 (END)

```


Here, we document the DHCP Ip assignment and Interfaces on vm2

```

ubuntu@ubuntu:~$ ip netns exec ens33
GENERAL.DEVICE: ens33
GENERAL.TYPE: ethernet
GENERAL.HWADDR: 08:0C:29:0C:3F:86
GENERAL.MTU: 1500
GENERAL.STATE: 100 (connected)
GENERAL.CONNECTION: netplan-xx-all-en
GENERAL.CON-PATH: /org/freedesktop/NetworkManager/ActiveConnection/11
WIRED-PROPERTIES.CARRIER: on
IP4.ADDRESS[1]: 172.20.10.3/28
IP4.GATEWAY: 172.20.10.1
IP4.ROUTE[1]: dst = 172.20.10.0/28, nh = 0.0.0.0, mt = 100
IP4.ROUTE[2]: dst = 0.0.0.0/0, nh = 172.20.10.1, mt = 100
IP4.DNS[1]: 172.20.10.1
IP6.ADDRESS[1]: fe80::20c:29ff:fe0c:3f86/64
IP6.GATEWAY: --
IP6.ROUTE[1]: dst = fe80::/64, nh = ::, mt = 256

GENERAL.DEVICE: lo
GENERAL.TYPE: loopback
GENERAL.HWADDR: 00:00:00:00:00:00
GENERAL.MTU: 65536
GENERAL.STATE: 100 (connected (externally))
GENERAL.CONNECTION: lo
GENERAL.CON-PATH: /org/freedesktop/NetworkManager/ActiveConnection/1
IP4.ADDRESS[1]: 127.0.0.1/8
IP4.GATEWAY: --
IP6.ADDRESS[1]: ::1/128
IP6.GATEWAY: --
  
```

Reflections and Learnings

It was important to know how VMware works in bridged network compare to various environments because I had configured two VMs in Bridged mode and connect a mobile device (host) to a mobile hot spot. The host, which in this arrangement was the mobile hotspot in the capacity of a router, had the same DHCP server giving IP addresses to both VMs. This enabled the VMs to act like some other devices on the same physical network and it was able to ping between the VMs and the individual VMs against the host, as well as the ability to access internet connectivity. The interesting part was that the VMs effectively became part of the network of the hotspot, simply as other physical devices do.

Because of this exercise, I got to know that Bridged mode comes in handy where VMs have to behave as actual devices in an established network even though the network could be as basic as a mobile hotspot. At the same time, I realized that the given setup assumes the existence of the DHCP server in the external network, thus without it, IP assigning would be unavailable. This incident also helped me to understand that Bridged mode could provide full connectivity but also created a scenario of IP conflicts and such a VM should be exposed to other devices on the network and that would pose a security concern. This, in general, re-confirmed in my mind how VMware translates the virtual adapters to the physical interface of the host and why Bridged mode is the most appropriate when we want to check real-world connectivity.

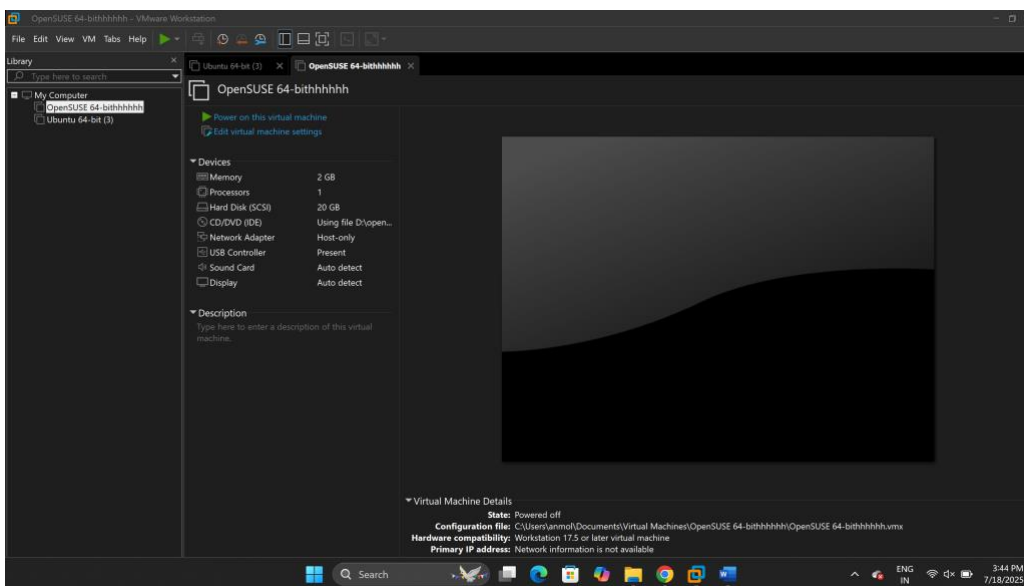
Part 3 – Host Only Configuration (Anmol Singh 500223958)

Objective

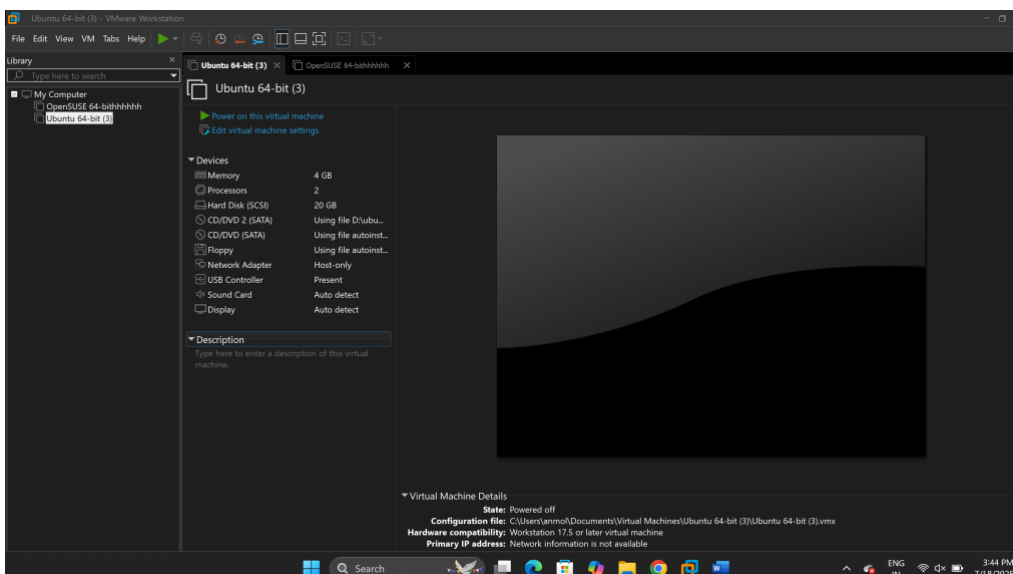
To set up and test **Host-only** networking in VMware Workstation Pro using **2 virtual machines (VMs)**, and verify that:

- The VMs can communicate **with each other**
- The VMs can communicate **with the host**
- The VMs **cannot access the internet**

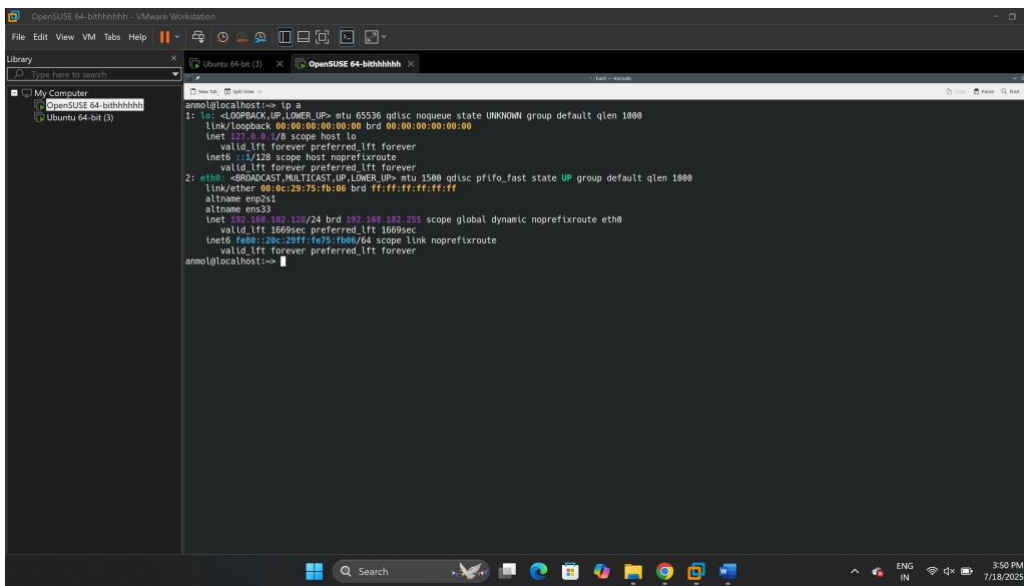
For this part, we create two virtual machines on VMware Workstation pro, one is Ubuntu and one is OpenSUSE



Now, we go to the vm network adapter settings and choose “host only ” as the network adapter



Here, we check the Ip address for vm 1, which is OpenSUSE in this case, we use the “Ip a” command and the result is “192.168.182.128”

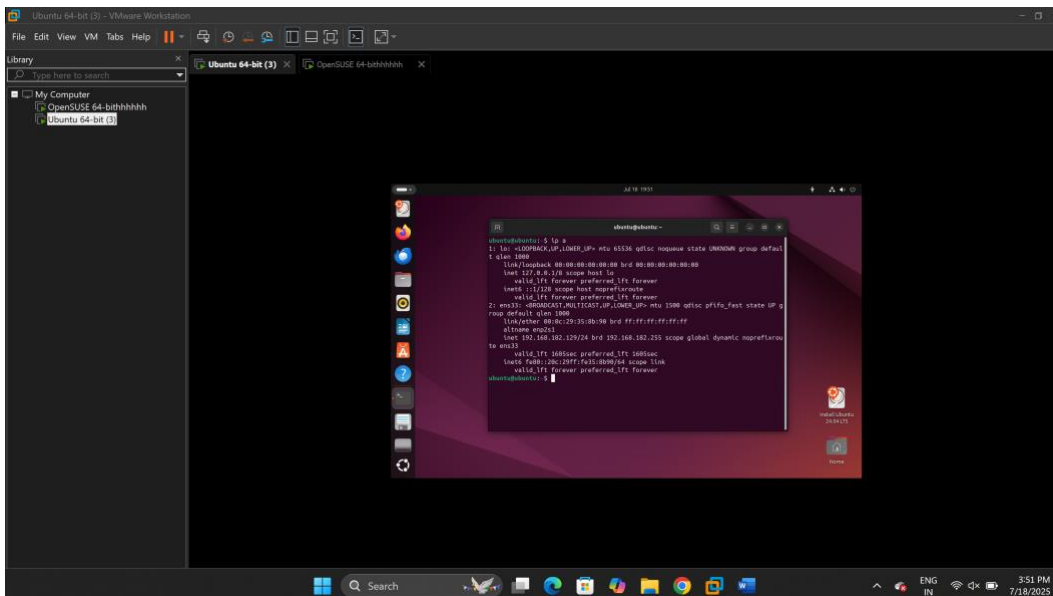


```

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:0c:29:75:7b:06 brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.182.128/24 brd 192.168.182.255 scope global dynamic noprefixroute ens33
        valid_lft 1669sec preferred_lft 1669sec
    inet6 fe80::20c:29ff:fe75:7b06/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
armol@localhost:~$

```

Once again we use the same “Ip a” command to check Ubuntu vm’s Ip address, the result is “192.168.182.129”



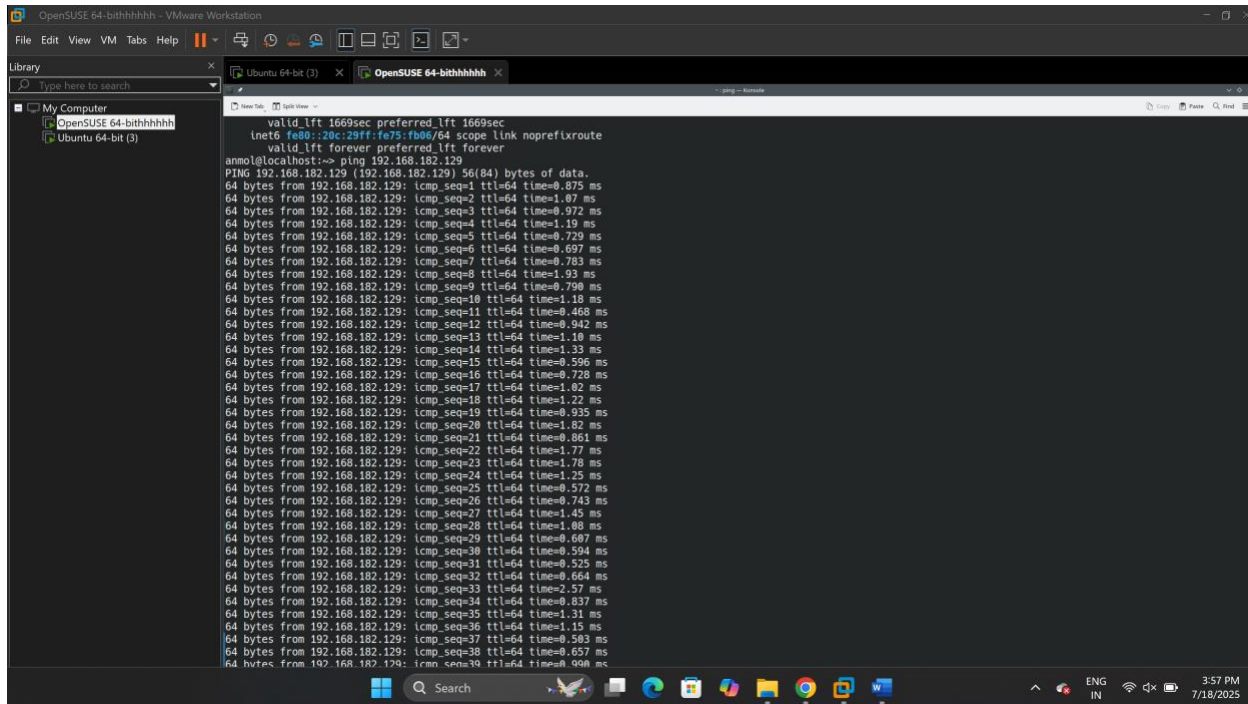
```

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:0c:29:75:7b:06 brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.182.129/24 brd 192.168.182.255 scope global dynamic noprefixroute ens33
        valid_lft 1669sec preferred_lft 1669sec
    inet6 fe80::20c:29ff:fe75:7b06/64 scope link
        valid_lft forever preferred_lft forever
ubuntu@ubuntu:~$

```

Testing Connectivity

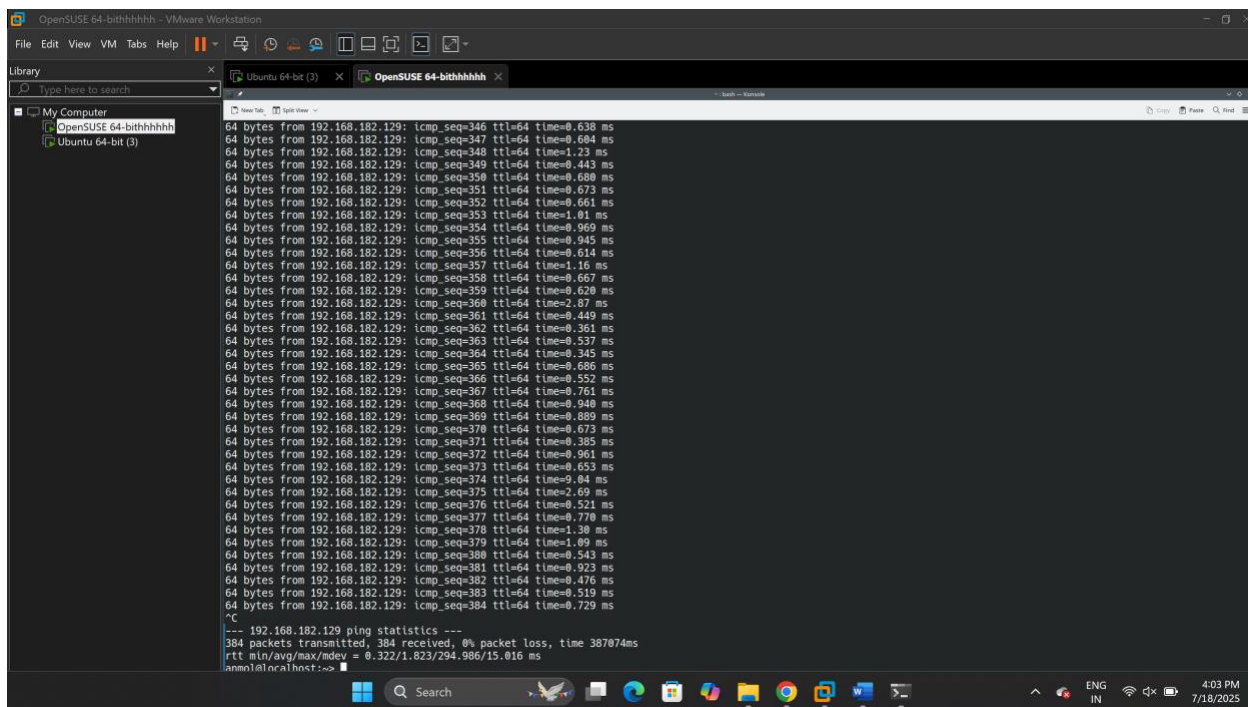
From the OpenSUSE vm, we ping the Ubuntu vm, the result is successful as seen below



```

OpenSUSE 64-bit hhhh - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
OpenSUSE 64-bit hhhh
Ubuntu 64-bit (3)
New Tab
Split View
ping - Ubuntu
valid_lft 1669sec preferred_lft 1669sec
inet6 fe80::20c:29ff:fe75:fb06/64 scope link noprefixroute
valid_lft forever preferred_lft forever
anmol@localhost:~$ ping 192.168.182.129
PING 192.168.182.129 (192.168.182.129) 56(84) bytes of data.
64 bytes from 192.168.182.129: icmp_seq=1 ttl=64 time=0.875 ms
64 bytes from 192.168.182.129: icmp_seq=2 ttl=64 time=1.07 ms
64 bytes from 192.168.182.129: icmp_seq=3 ttl=64 time=0.972 ms
64 bytes from 192.168.182.129: icmp_seq=4 ttl=64 time=1.19 ms
64 bytes from 192.168.182.129: icmp_seq=5 ttl=64 time=0.729 ms
64 bytes from 192.168.182.129: icmp_seq=6 ttl=64 time=0.697 ms
64 bytes from 192.168.182.129: icmp_seq=7 ttl=64 time=0.783 ms
64 bytes from 192.168.182.129: icmp_seq=8 ttl=64 time=1.93 ms
64 bytes from 192.168.182.129: icmp_seq=9 ttl=64 time=0.790 ms
64 bytes from 192.168.182.129: icmp_seq=10 ttl=64 time=1.18 ms
64 bytes from 192.168.182.129: icmp_seq=11 ttl=64 time=0.468 ms
64 bytes from 192.168.182.129: icmp_seq=12 ttl=64 time=0.942 ms
64 bytes from 192.168.182.129: icmp_seq=13 ttl=64 time=1.10 ms
64 bytes from 192.168.182.129: icmp_seq=14 ttl=64 time=1.33 ms
64 bytes from 192.168.182.129: icmp_seq=15 ttl=64 time=0.596 ms
64 bytes from 192.168.182.129: icmp_seq=16 ttl=64 time=0.728 ms
64 bytes from 192.168.182.129: icmp_seq=17 ttl=64 time=1.02 ms
64 bytes from 192.168.182.129: icmp_seq=18 ttl=64 time=1.22 ms
64 bytes from 192.168.182.129: icmp_seq=19 ttl=64 time=0.935 ms
64 bytes from 192.168.182.129: icmp_seq=20 ttl=64 time=1.82 ms
64 bytes from 192.168.182.129: icmp_seq=21 ttl=64 time=0.861 ms
64 bytes from 192.168.182.129: icmp_seq=22 ttl=64 time=1.77 ms
64 bytes from 192.168.182.129: icmp_seq=23 ttl=64 time=1.78 ms
64 bytes from 192.168.182.129: icmp_seq=24 ttl=64 time=1.25 ms
64 bytes from 192.168.182.129: icmp_seq=25 ttl=64 time=0.572 ms
64 bytes from 192.168.182.129: icmp_seq=26 ttl=64 time=0.743 ms
64 bytes from 192.168.182.129: icmp_seq=27 ttl=64 time=1.45 ms
64 bytes from 192.168.182.129: icmp_seq=28 ttl=64 time=1.08 ms
64 bytes from 192.168.182.129: icmp_seq=29 ttl=64 time=0.687 ms
64 bytes from 192.168.182.129: icmp_seq=30 ttl=64 time=0.594 ms
64 bytes from 192.168.182.129: icmp_seq=31 ttl=64 time=0.525 ms
64 bytes from 192.168.182.129: icmp_seq=32 ttl=64 time=0.664 ms
64 bytes from 192.168.182.129: icmp_seq=33 ttl=64 time=2.57 ms
64 bytes from 192.168.182.129: icmp_seq=34 ttl=64 time=0.837 ms
64 bytes from 192.168.182.129: icmp_seq=35 ttl=64 time=1.31 ms
64 bytes from 192.168.182.129: icmp_seq=36 ttl=64 time=1.15 ms
64 bytes from 192.168.182.129: icmp_seq=37 ttl=64 time=0.583 ms
64 bytes from 192.168.182.129: icmp_seq=38 ttl=64 time=0.657 ms
64 bytes from 192.168.182.129: icmp_seq=39 ttl=64 time=0.908 ms
^C

```

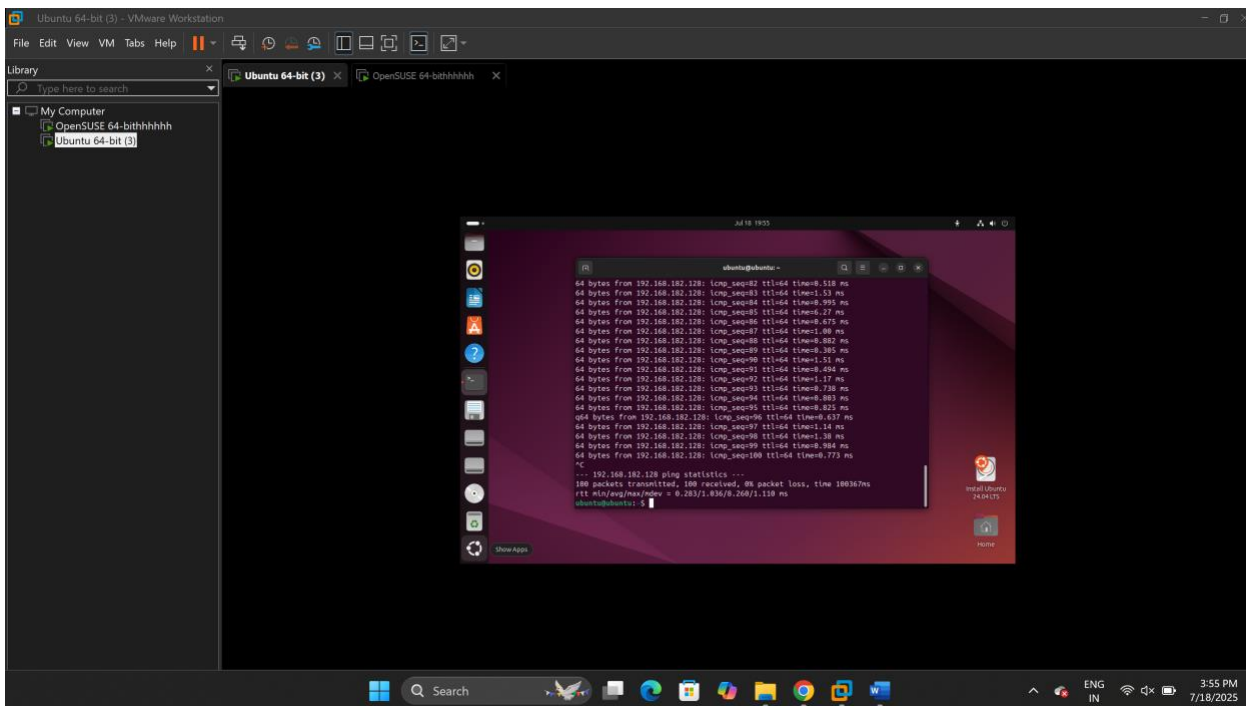
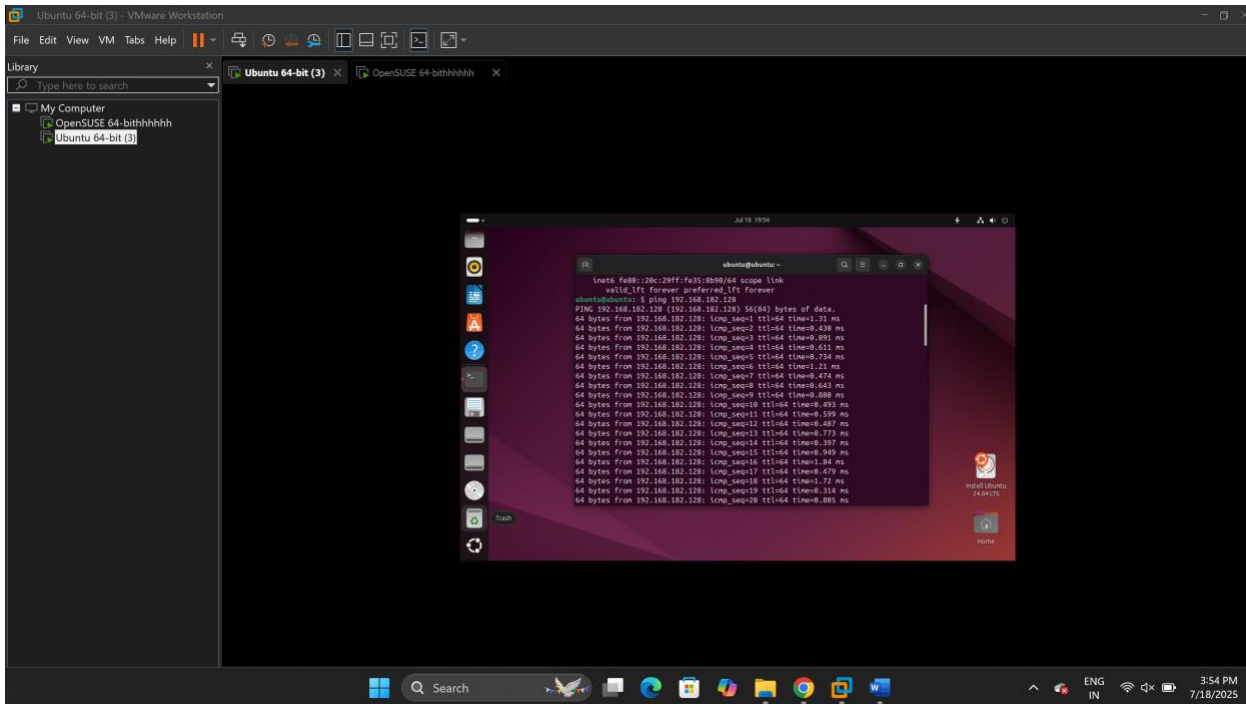


```

OpenSUSE 64-bit hhhh - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
OpenSUSE 64-bit hhhh
Ubuntu 64-bit (3)
New Tab
Split View
ping - Ubuntu
64 bytes from 192.168.182.129: icmp_seq=346 ttl=64 time=0.638 ms
64 bytes from 192.168.182.129: icmp_seq=347 ttl=64 time=0.604 ms
64 bytes from 192.168.182.129: icmp_seq=348 ttl=64 time=1.23 ms
64 bytes from 192.168.182.129: icmp_seq=349 ttl=64 time=0.443 ms
64 bytes from 192.168.182.129: icmp_seq=350 ttl=64 time=0.688 ms
64 bytes from 192.168.182.129: icmp_seq=351 ttl=64 time=0.673 ms
64 bytes from 192.168.182.129: icmp_seq=352 ttl=64 time=0.661 ms
64 bytes from 192.168.182.129: icmp_seq=353 ttl=64 time=1.01 ms
64 bytes from 192.168.182.129: icmp_seq=354 ttl=64 time=0.969 ms
64 bytes from 192.168.182.129: icmp_seq=355 ttl=64 time=0.945 ms
64 bytes from 192.168.182.129: icmp_seq=356 ttl=64 time=0.614 ms
64 bytes from 192.168.182.129: icmp_seq=357 ttl=64 time=1.16 ms
64 bytes from 192.168.182.129: icmp_seq=358 ttl=64 time=0.667 ms
64 bytes from 192.168.182.129: icmp_seq=359 ttl=64 time=0.628 ms
64 bytes from 192.168.182.129: icmp_seq=360 ttl=64 time=2.07 ms
64 bytes from 192.168.182.129: icmp_seq=361 ttl=64 time=0.449 ms
64 bytes from 192.168.182.129: icmp_seq=362 ttl=64 time=0.361 ms
64 bytes from 192.168.182.129: icmp_seq=363 ttl=64 time=0.537 ms
64 bytes from 192.168.182.129: icmp_seq=364 ttl=64 time=0.345 ms
64 bytes from 192.168.182.129: icmp_seq=365 ttl=64 time=0.686 ms
64 bytes from 192.168.182.129: icmp_seq=366 ttl=64 time=0.552 ms
64 bytes from 192.168.182.129: icmp_seq=367 ttl=64 time=0.761 ms
64 bytes from 192.168.182.129: icmp_seq=368 ttl=64 time=0.948 ms
64 bytes from 192.168.182.129: icmp_seq=369 ttl=64 time=0.809 ms
64 bytes from 192.168.182.129: icmp_seq=370 ttl=64 time=0.673 ms
64 bytes from 192.168.182.129: icmp_seq=371 ttl=64 time=0.385 ms
64 bytes from 192.168.182.129: icmp_seq=372 ttl=64 time=0.961 ms
64 bytes from 192.168.182.129: icmp_seq=373 ttl=64 time=0.653 ms
64 bytes from 192.168.182.129: icmp_seq=374 ttl=64 time=0.944 ms
64 bytes from 192.168.182.129: icmp_seq=375 ttl=64 time=2.69 ms
64 bytes from 192.168.182.129: icmp_seq=376 ttl=64 time=0.521 ms
64 bytes from 192.168.182.129: icmp_seq=377 ttl=64 time=0.778 ms
64 bytes from 192.168.182.129: icmp_seq=378 ttl=64 time=1.38 ms
64 bytes from 192.168.182.129: icmp_seq=379 ttl=64 time=1.49 ms
64 bytes from 192.168.182.129: icmp_seq=380 ttl=64 time=0.543 ms
64 bytes from 192.168.182.129: icmp_seq=381 ttl=64 time=0.923 ms
64 bytes from 192.168.182.129: icmp_seq=382 ttl=64 time=0.476 ms
64 bytes from 192.168.182.129: icmp_seq=383 ttl=64 time=0.519 ms
64 bytes from 192.168.182.129: icmp_seq=384 ttl=64 time=0.729 ms
^C
--- 192.168.182.129 ping statistics ---
384 packets transmitted, 384 received, 0% packet loss, time 387074ms
rtt min/avg/max/mdev = 0.322/1.823/294.986/15.016 ms
anmol@localhost:~$

```

From vm 2, we ping the OpenSUSE vm, the result is successful as seen below



This figure displays the host's Ip address, In order to retrieve it, we used the “Ip a” command

```

Command Prompt

IPv6 Address. . . . . : 2605:8d80:6c20:d844:155f:888e:ae7:72e4
Temporary IPv6 Address. . . . . : 2605:8d80:6c20:d844:34b3:7b0d:3ff9:779a
Link-Local IPv6 Address. . . . . : fe80::f62c:f586:b700:7199%10
IPv4 Address. . . . . : 172.20.10.5
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : fe80::646d:22ff:febc:2d64%10
                          172.20.10.1

Ethernet adapter VMware Network Adapter VMnet1:

Connection-specific DNS Suffix  . : 
Link-Local IPv6 Address. . . . . : fe80::23a8:982e:8eb0:2306%12
IPv4 Address. . . . . : 192.168.182.1
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 

Ethernet adapter VMware Network Adapter VMnet8:

Connection-specific DNS Suffix  . : 
Link-Local IPv6 Address. . . . . : fe80::b8fb:1d4b:e0f2:f7a6%7
IPv4 Address. . . . . : 192.168.202.1
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 

Ethernet adapter Bluetooth Network Connection:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix  . : 

C:\Users\anmol>
  
```

Here, we ping the host's VMnet1 Ip address “192.168.56.1” from the OpenSUSE vm, we pressed ctrl c to stop the ping message, the result is successful as seen below

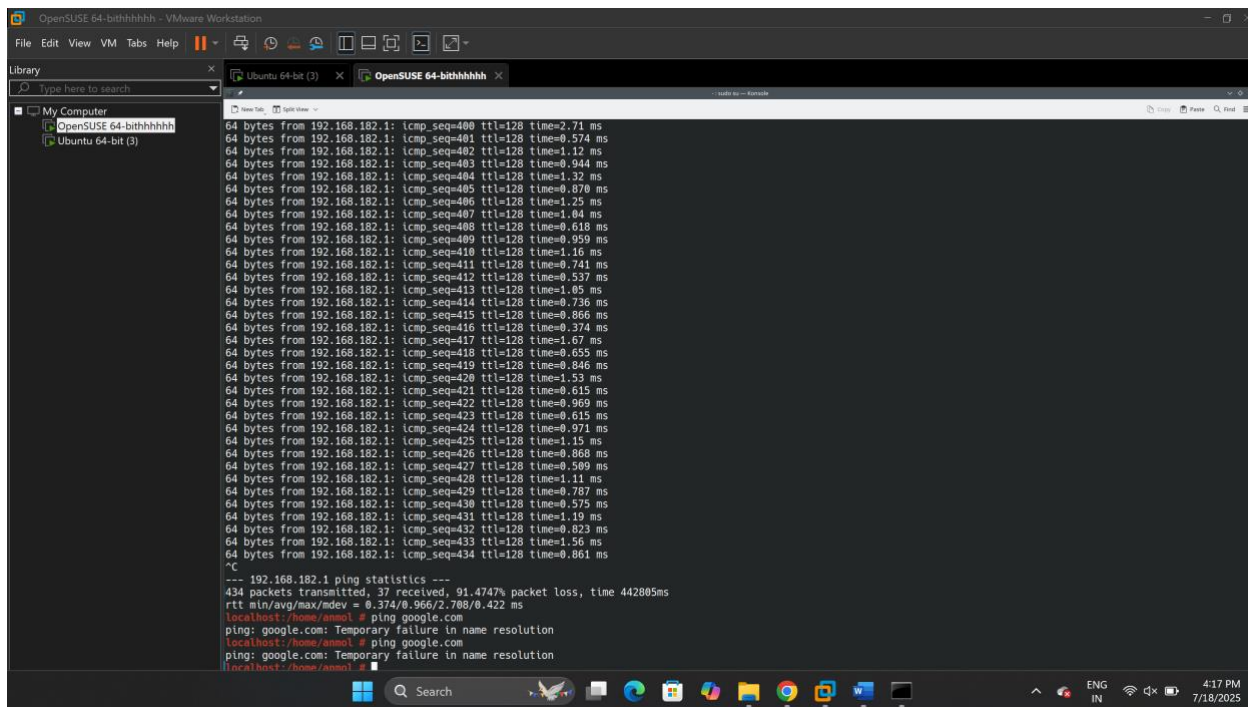
```

OpenSUSE 64-bit (3) x OpenSUSE 64-bit (3)

64 bytes from 192.168.182.1: icmp_seq=390 ttl=128 time=1.15 ms
64 bytes from 192.168.182.1: icmp_seq=399 ttl=128 time=0.448 ms
64 bytes from 192.168.182.1: icmp_seq=400 ttl=128 time=2.71 ms
64 bytes from 192.168.182.1: icmp_seq=401 ttl=128 time=0.574 ms
64 bytes from 192.168.182.1: icmp_seq=402 ttl=128 time=1.12 ms
64 bytes from 192.168.182.1: icmp_seq=403 ttl=128 time=0.944 ms
64 bytes from 192.168.182.1: icmp_seq=404 ttl=128 time=1.32 ms
64 bytes from 192.168.182.1: icmp_seq=405 ttl=128 time=0.878 ms
64 bytes from 192.168.182.1: icmp_seq=406 ttl=128 time=1.25 ms
64 bytes from 192.168.182.1: icmp_seq=407 ttl=128 time=1.64 ms
64 bytes from 192.168.182.1: icmp_seq=408 ttl=128 time=0.618 ms
64 bytes from 192.168.182.1: icmp_seq=409 ttl=128 time=0.959 ms
64 bytes from 192.168.182.1: icmp_seq=410 ttl=128 time=1.16 ms
64 bytes from 192.168.182.1: icmp_seq=411 ttl=128 time=0.741 ms
64 bytes from 192.168.182.1: icmp_seq=412 ttl=128 time=0.537 ms
64 bytes from 192.168.182.1: icmp_seq=413 ttl=128 time=1.65 ms
64 bytes from 192.168.182.1: icmp_seq=414 ttl=128 time=0.736 ms
64 bytes from 192.168.182.1: icmp_seq=415 ttl=128 time=0.866 ms
64 bytes from 192.168.182.1: icmp_seq=416 ttl=128 time=0.374 ms
64 bytes from 192.168.182.1: icmp_seq=417 ttl=128 time=1.67 ms
64 bytes from 192.168.182.1: icmp_seq=418 ttl=128 time=0.655 ms
64 bytes from 192.168.182.1: icmp_seq=419 ttl=128 time=0.846 ms
64 bytes from 192.168.182.1: icmp_seq=420 ttl=128 time=1.53 ms
64 bytes from 192.168.182.1: icmp_seq=421 ttl=128 time=0.615 ms
64 bytes from 192.168.182.1: icmp_seq=422 ttl=128 time=0.969 ms
64 bytes from 192.168.182.1: icmp_seq=423 ttl=128 time=0.615 ms
64 bytes from 192.168.182.1: icmp_seq=424 ttl=128 time=0.971 ms
64 bytes from 192.168.182.1: icmp_seq=425 ttl=128 time=1.15 ms
64 bytes from 192.168.182.1: icmp_seq=426 ttl=128 time=0.868 ms
64 bytes from 192.168.182.1: icmp_seq=427 ttl=128 time=0.589 ms
64 bytes from 192.168.182.1: icmp_seq=428 ttl=128 time=1.11 ms
64 bytes from 192.168.182.1: icmp_seq=429 ttl=128 time=0.787 ms
64 bytes from 192.168.182.1: icmp_seq=430 ttl=128 time=0.575 ms
64 bytes from 192.168.182.1: icmp_seq=431 ttl=128 time=1.19 ms
64 bytes from 192.168.182.1: icmp_seq=432 ttl=128 time=0.823 ms
64 bytes from 192.168.182.1: icmp_seq=433 ttl=128 time=1.56 ms
64 bytes from 192.168.182.1: icmp_seq=434 ttl=128 time=0.661 ms
^C

--- 192.168.182.1 ping statistics ---
434 packets transmitted, 37 received, 91.474% packet loss, time 44288ms
rtt min/avg/max/mdev = 0.374/0.966/2.788/0.422 ms
inca1hoxct:~$
  
```


Here, we try to ping “google.com” and send packets to the internet, which rightfully gave us an error, we were expecting that because the host-only mode restricts the vm’s access to the internet



```

OpenSUSE 64-bit hhhhhh - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
OpenSUSE 64-bit hhhhhh
Ubuntu 64-bit (3)
64 bytes from 192.168.182.1: icmp_seq=400 ttl=128 time=2.71 ms
64 bytes from 192.168.182.1: icmp_seq=401 ttl=128 time=0.574 ms
64 bytes from 192.168.182.1: icmp_seq=402 ttl=128 time=1.12 ms
64 bytes from 192.168.182.1: icmp_seq=403 ttl=128 time=0.944 ms
64 bytes from 192.168.182.1: icmp_seq=404 ttl=128 time=1.32 ms
64 bytes from 192.168.182.1: icmp_seq=405 ttl=128 time=0.870 ms
64 bytes from 192.168.182.1: icmp_seq=406 ttl=128 time=1.25 ms
64 bytes from 192.168.182.1: icmp_seq=407 ttl=128 time=1.64 ms
64 bytes from 192.168.182.1: icmp_seq=408 ttl=128 time=0.618 ms
64 bytes from 192.168.182.1: icmp_seq=409 ttl=128 time=0.959 ms
64 bytes from 192.168.182.1: icmp_seq=410 ttl=128 time=1.16 ms
64 bytes from 192.168.182.1: icmp_seq=411 ttl=128 time=0.741 ms
64 bytes from 192.168.182.1: icmp_seq=412 ttl=128 time=0.537 ms
64 bytes from 192.168.182.1: icmp_seq=413 ttl=128 time=1.05 ms
64 bytes from 192.168.182.1: icmp_seq=414 ttl=128 time=0.736 ms
64 bytes from 192.168.182.1: icmp_seq=415 ttl=128 time=0.866 ms
64 bytes from 192.168.182.1: icmp_seq=416 ttl=128 time=0.374 ms
64 bytes from 192.168.182.1: icmp_seq=417 ttl=128 time=1.67 ms
64 bytes from 192.168.182.1: icmp_seq=418 ttl=128 time=0.655 ms
64 bytes from 192.168.182.1: icmp_seq=419 ttl=128 time=0.846 ms
64 bytes from 192.168.182.1: icmp_seq=420 ttl=128 time=1.53 ms
64 bytes from 192.168.182.1: icmp_seq=421 ttl=128 time=0.615 ms
64 bytes from 192.168.182.1: icmp_seq=422 ttl=128 time=0.969 ms
64 bytes from 192.168.182.1: icmp_seq=423 ttl=128 time=0.615 ms
64 bytes from 192.168.182.1: icmp_seq=424 ttl=128 time=0.971 ms
64 bytes from 192.168.182.1: icmp_seq=425 ttl=128 time=1.15 ms
64 bytes from 192.168.182.1: icmp_seq=426 ttl=128 time=0.868 ms
64 bytes from 192.168.182.1: icmp_seq=427 ttl=128 time=0.509 ms
64 bytes from 192.168.182.1: icmp_seq=428 ttl=128 time=1.11 ms
64 bytes from 192.168.182.1: icmp_seq=429 ttl=128 time=0.787 ms
64 bytes from 192.168.182.1: icmp_seq=430 ttl=128 time=0.575 ms
64 bytes from 192.168.182.1: icmp_seq=431 ttl=128 time=1.19 ms
64 bytes from 192.168.182.1: icmp_seq=432 ttl=128 time=0.823 ms
64 bytes from 192.168.182.1: icmp_seq=433 ttl=128 time=1.56 ms
64 bytes from 192.168.182.1: icmp_seq=434 ttl=128 time=0.861 ms
^C
--- 192.168.182.1 ping statistics ---
434 packets transmitted, 37 received, 91.4747% packet loss, time 442805ms
rtt min/avg/max/mdev = 0.374/0.966/2.708/0.422 ms
localhost:~/home/annol # ping google.com
ping: google.com: Temporary failure in name resolution
localhost:~/home/annol # ping google.com
ping: google.com: Temporary failure in name resolution
localhost:~/home/annol #

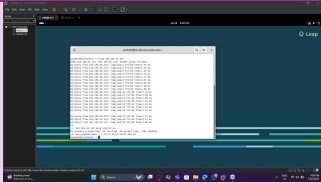
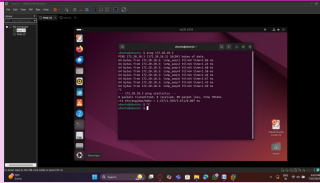
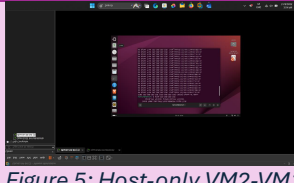
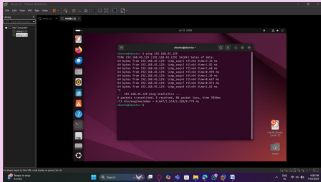
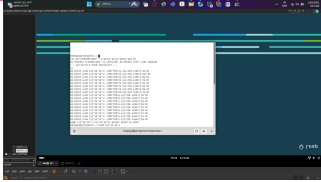
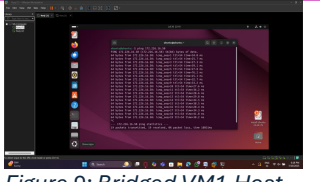
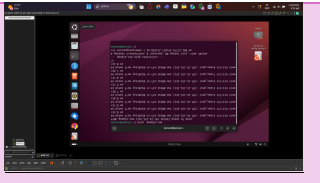
```

Reflection and Learnings

In this configuration both VM’s had their own, private IP addresses on the same subnet as the VMware Host-only adapter, which served as gateway. I verified that the two VMs were able to communicate with each other and with the host machine although they were not connected to internet. It was not an expected behavior since Host-only mode does not bridge to any other network; it builds a local LAN linking the VMs to the host. Testing led me to also learn the usefulness of this mode on a secure, controlled environment that does not need an external connection like a local testing or internal service simulation.

This activity allowed me to realize the main differences between isolation and connection in online networks. Host-only is (contrary to NAT or Bridged mode) completely isolated from all other (external) networks, less security risk as with Bridged, no more IP collisions with the physical network. It has also shown the importance of IP ranges checks and gateway settings just to make sure the communication is proper with the VMs. Comprehensively, this experience gave me a better understanding regarding network segmentation and the reason why Host-only is the desirable implementation when it comes to isolation from internet exposure.

Part 4 – Comparison Table (Mustansir Lokhandwala 500224071)

Feature/Test	NAT Mode	Bridged Mode	Host-only Mode
VM-VM Connectivity	 <p>Figure 1: NAT VM1-VM2, successful ping</p>	 <p>Figure 3: Bridged VM1-VM2, successful ping</p>	 <p>Figure 5: Host-only VM2-VM1, successful ping</p>
	 <p>Figure 2: NAT VM2-VM1, successful ping</p>	 <p>Figure 4: Bridged VM2-VM1, successful ping</p>	 <p>Figure 6: Host-only VM1-VM2, successful ping</p>
VM-Host Connectivity	 <p>Figure 7: NAT VM1-Host, Successful ping</p>	 <p>Figure 9: Bridged VM1-Host, Successful ping</p>	 <p>Figure 11: Host-only VM1-Host, Successful ping</p>
	 <p>Figure 8: NAT VM2-Host, Successful ping</p>	 <p>Figure 10: Bridged VM2-Host, successful ping</p>	
VM-Internet Connectivity	 <p>Figure 12: NAT VM1-Internet, Successful ping</p>	 <p>Figure 14: Bridged VM1-Internet, Successful ping</p>	 <p>Figure 16: Host-only VM1-Internet, successful ping</p>
	 <p>Figure 13: NAT VM2-Internet, Successful ping</p>	 <p>Figure 15: Bridged VM2-Internet, Successful ping</p>	
IP Address Type	Private (192.168.x.x - VMware NAT)	Private (172.20.x.x - from router)	Private (192.168.x.x - Host-only subnet)

DHCP Server Source	VMware NAT DHCP	Physical router's DHCP, mobile hot spot	VMware Host-only DHCP
Use Case	Secure internet access, no inbound	VMs as real network devices	Isolated Environments

Summaries

NAT

NAT , short for Network Address Translation, enables VMs to share the IP address of the host in order for it to connect to the internet. They are assigned with the private IPs and share by passing messages through the NAT gateway of the host. NAT is perfect when devs want to test software that requires internet connection but does not want to risk connecting their VMs to the physical network, such as when updating the packages or communicating with cloud APIs in a secure way.

Bridged Mode

Bridged mode connects the VMs to the physical network directly and it makes the virtual machines behave like individual physical computers using IPs of the primary router. This mode allows virtual machines to act as if real computers on the local network, and receive an Ip address from the designated router, in this case the mobile hotspot.

Host-Only Mode

Host-only mode provides the isolated network in which VM's are not able to communicate with any network besides the host, including access to the internet. This mode creates a private network between your virtual machines and your host, there's no internet access, so let's say a school project where you need the vm's to talk to each other and the host, like testing a local website where you don't want to risk external exposure, Host-only mode is perfect for situations like those.

Part 5 – Conclusion, Similarities and Differences (Mustansir Lokhandwala 500224071)

Key Differences

The first thing that hit me, when i began comparing the three modes of network, was their huge difference when it came to internet access. Both NAT and Bridged gave the virtual machines access to the internet, yet in completely different manners. NAT was a proxy- it did what the VMs did, it forwarded the traffic sent by the VMs and sent them to the host. Bridged was a different story, however, in that it placed the VMs at their own table with the same router as the host, essentially causing them to act like any other device in the network. Host-only was the most exclusive one because it entirely made the VMs inaccessible to the internet, which was much like operating in a solid bubble.

The other major contrast was when we checked where the VMs received their IP addresses from. On NAT and Host-only modes, VMware had served in the form of a mini internet service provider and it was dishing out the realm of a private IP address by VMware own internal DHCP servers. However, over Bridged mode, all IPs were issued by the physical router like a hotspot port or home wireless. This is what caused the Bridged VMs to look like genuine and separate machines on the network.

There was also a lot of difference in visibility and exposure. In the case of Bridged, the VMs appeared on LAN completely. That meant that they could communicate to other devices on the same Wi-Fi just as a real laptop would. NAT and Host-only made things more confidential. NAT concealed the VMs behind the IP of the host and Host-only did not even attempt to access the world beyond the virtual domain. We found out that indeed these differences matter when it comes to the exposure or the protection of a VM in various network configurations.

The last large difference was security. Host-only definitely was the most secure one, no internet, no outside exposure, only local communication. NAT was also safer than Bridged because no one could access the VMs directly on the outside and one had to do port forwarding to achieve that. Most open was Bridged, since it provided VMs with complete access and made them more susceptible to attacks in case the actual network was not secured.

All these distinctions were combined to demonstrate the extent to which network mode alters the behavior of the VMs, not only on the access level, but also on the identity, visibility and safety level. One could think that each mode assigned the VMs different roles in the network altogether.

Similarities

Although all the network modes were different in their own ways, there were certain things that they had in common. Regardless of the mode that we used, we were in a position to see the two VMs communicate. The VM-to-VM communication was agnostic, provided us with a solid thing to base our tests off of, and was available everywhere. It was good to know that some basic connection utility tools such as ping and ip remained the same despite the change of the IP ranges or the gateways.

The other similarity that we have observed is that all the VMs had the ability to connect to the host machine in all the modes. No matter what the mode was: NAT, Bridged or Host-only we were always talking to the host. It turned the host into a junction who monitored all the activities taking place in the whole virtual space. Additionally, the configuration process in itself quite easy as we all found ourselves in a rhythm to go into the VM settings, modify the adapter type and restart the VMs.

In one word, the final output and actions were diverse; however, some common structure allowed us to compare things in the case of overall configuration, testing, and verification of network connectivity.

Real-World Scenarios

Let's imagine that Abhijit has a web app which requires internet access to fetch some data, but he does not wish to expose his VM to the full network. NAT mode would suit him to a T, because he would have all the internet access and wouldn't have to care about another person using the network to snoop in his machine. It is quick, convenient and safe enough to do general development work.

Let's say Kunj is attempting to create a real world simulation where his VM must behave like an actual PC on a corporate network. Perhaps, he is checking what the VM would do in the context of a physical LAN, verifying file sharing, or attempting a local NAS device. A bridged mode would provide him with such a setup exactly, and the VM would simply become another device on the Wi-Fi.

Now Anmol, he is playing around with local tools, such as trying a local database or some automation scripts between two VMs, but doesn't want any external network to distract him. Host-only mode should fit right in here. With no Internet access, Host-only mode is like a secure store.

***] Reflections – The reflections for each part have been include at the end of each.**

Final Thoughts & Analysis

Looking back, this assignment may seem like just changing the network adapter settings, but it's more than that, it's about applying those different settings in real situation. We began with three paths which constantly overlapped in our tests, conversations, and even the mistakes we made in the process. It has forced us to be methodical in our procedure and accurate in noting our findings.

The most notable aspect was that all the modes are dissimilar in the narrative of connectivity and control. NAT provides you with internet but a sense of security and Bridged provides you with reality and exposure and Host-only gives you privacy and calm. It is not that you can just know when and why to use each one, but by doing it, by typing in and running commands, failing a ping, and then troubleshooting it to know what went wrong.

Such a practical laboratory session provided us with the idea of how the networking operates, not only in a virtual but also in a real environment. Just imagine, VMs are a kind of mini systems themselves, and the structure in which they are related to each other, determines the kind of environment they'll live in. It is an interesting concept for sure.

So, it can be the creation of a stable dev environment, testing a server on an actual network or simulation of a closed office space, all these options now seem like instruments which we can work with, not just settings on a screen.

Group Log

Mustansir	Part 4, 5 and helped with Part 3
Abhijit	Part 1 and helped with Part 2
Kunj	Part 2
Anmol	Part 3

GitHub Repository

<https://github.com/Mustansirlok/VIRTUALISATION---WEEK12-ASSIGNMENT--GROUP-A.git>