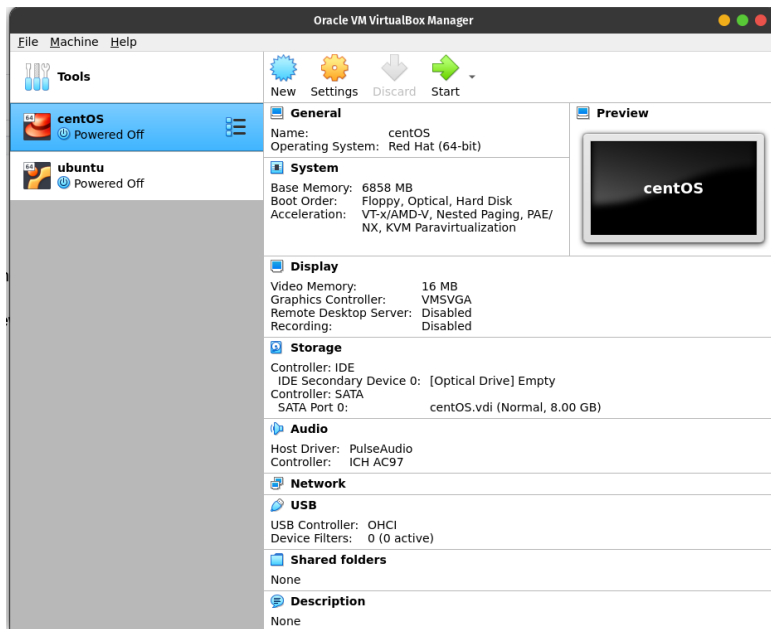


Qno1

New virtual machine CentOS was created.



Qno1(a):

Firewalld was already installed in the system (centOS) by default, commands that can be used to install firewalld and check its status is:

- `sudo yum install firewalld`
- `systemctl status firewalld`

```
[trikesh@localhost ~]$ systemctl status firewalld
■ firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2021-11-02 07:37:03 EDT; 6min ago
     Docs: man:firewalld(1)
   Main PID: 695 (firewalld)
    CGroup: /system.slice/firewalld.service
            └─695 /usr/bin/python2 -Es /usr/sbin/firewalld --nofork --nopid

Nov 02 07:37:00 localhost.localdomain systemd[1]: Starting firewalld - dynamic firewall daemon...
Nov 02 07:37:03 localhost.localdomain systemd[1]: Started firewalld - dynamic firewall daemon.
Nov 02 07:37:04 localhost.localdomain firewalld[695]: WARNING: AllowZoneDrifting is enabled. Th...w.
Hint: Some lines were ellipsized, use -l to show in full.
[trikesh@localhost ~]$ _
```

Qno1(b)

To block certain IP addresses using firewalld following command was used:

- `firewall-cmd --permanent --add-rich-rule="rule family='ipv4' source address='180.76.15.154' reject"`

```
[root@localhost rikesh]# firewall-cmd --permanent --add-rich-rule="rule family='ipv4' source address='180.76.15.154' reject"
success
[root@localhost rikesh]# firewall-cmd --reload
success
[root@localhost rikesh]#
```

IP Address '180.76.15.154' is taken as a sample.

To save the firewall rule permanently --permanent option is used in the command.

We have to reload firewalld after adding a permanent rule using the command:

- `firewall-cmd --reload`

Qno1(c)

To allow http, https and ssh connection using firewall, we use the following commands respectively:

- firewall-cmd --zone=public --add-service=http
- firewall-cmd --zone=public --add-service=https
- firewall-cmd --zone=public --add-service=ssh

The above rules will be removed after system reboot. We need to use the --permanent option to save the rules permanently as listed below:

- firewall-cmd --permanent --zone=public --add-service=http
- firewall-cmd --permanent --zone=public --add-service=https
- firewall-cmd --permanent --zone=public --add-service=ssh

```
[root@localhost rikeshl# firewall-cmd --zone=public --add-service=http
success
[root@localhost rikeshl# firewall-cmd --zone=public --add-service=https
success
[root@localhost rikeshl# firewall-cmd --zone=public --add-service=ssh
Warning: ALREADY_ENABLED: 'ssh' already in 'public'
success
[root@localhost rikeshl# firewall-cmd --permanent --zone=public --add-service=ssh
Warning: ALREADY_ENABLED: ssh
success
[root@localhost rikeshl# firewall-cmd --permanent --zone=public --add-service=https
success
[root@localhost rikeshl# firewall-cmd --permanent --zone=public --add-service=http
success
[root@localhost rikeshl#
```

Qno1(d)

TFTP rule was added to the Centos Firewall using the command:

- firewall-cmd --permanent --zone=public --add-service=tftp
- firewall-cmd --reload

```
[root@localhost rikesh]# firewall-cmd --permanent --zone=public --add-service=tftp
success
[root@localhost rikesh]# firewall-cmd --reload
success
[root@localhost rikesh]# _
```

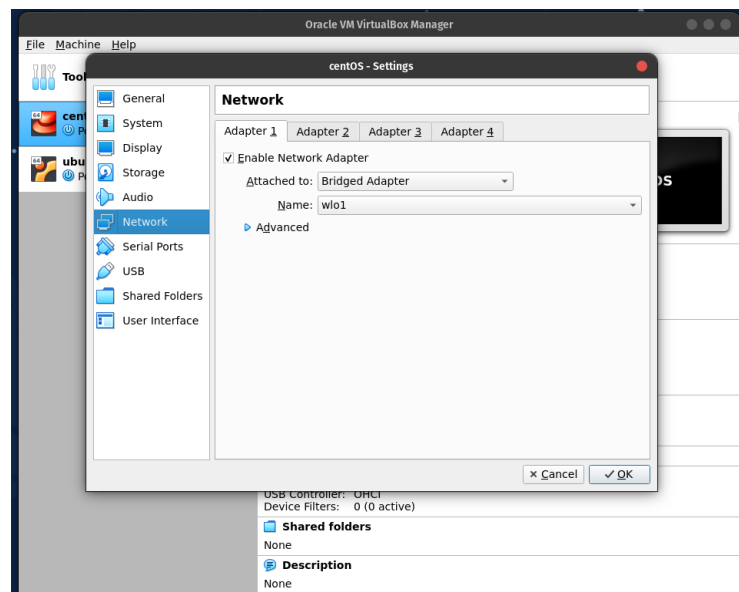
Trivial File Transfer Protocol is a simple lockstep File Transfer Protocol which allows a client to get a file from or put a file onto a remote host. One of its primary uses is in the early stages of nodes booting from a local area network. TFTP has been used for this application because it is very simple to implement.

So, I preferred adding this rule in the firewall.

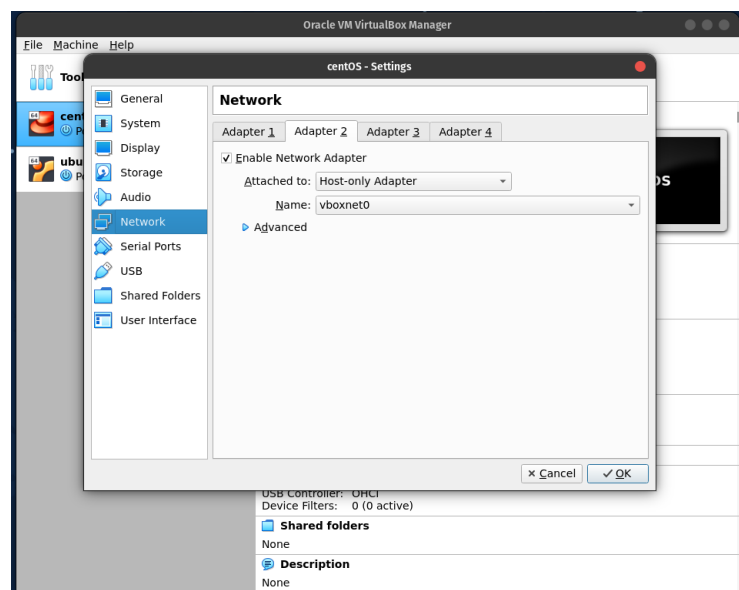
Qno2

Created the two virtual machines for this task. First virtual machine was Centos with two network interfaces where one behaves as WAN and another as LAN and the second virtual machine was Ubuntu where we attached the previously created LAN interface from VM1.

- In the first VM (centOS) we establish two network interfaces:
- ◆ Bridged Adaptor (name: wlo1/ enp0s3)



- ◆ Host-Only Adaptor (name: vboxnet0/ enp0s8)



```

enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.254.131 netmask 255.255.255.0 broadcast 192.168.254.255
    inet6 fe80::ad9:2f72:5ea:1261 prefixlen 64 scopeid 0x20<link>
    inet6 2407:5200:400:8f7b:ed5b:f53:a2aa:5742 prefixlen 64 scopeid 0x0<global>
    ether 08:00:27:7f:fe:72 txqueuelen 1000 (Ethernet)
    RX packets 48 bytes 6524 (6.3 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 88 bytes 10501 (10.2 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.99.101 netmask 255.255.255.0 broadcast 192.168.99.255
    inet6 fe80::2ab2:2701:2cce:861a prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:c6:23:9f txqueuelen 1000 (Ethernet)
    RX packets 19 bytes 5875 (5.7 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 23 bytes 4237 (4.1 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

virbr8: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 192.168.122.1 netmask 255.255.255.0 broadcast 192.168.122.255
    ether 52:54:00:79:82:c4 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

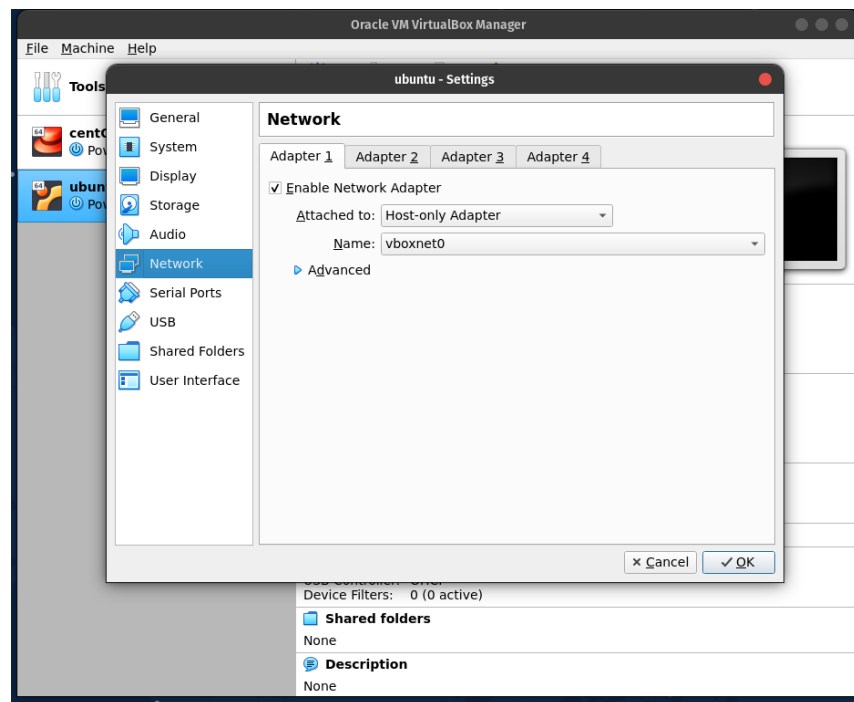
trikesh@localhost ~$ _

```

enp0s3 = 192.168.254.131

enp0s8 = 192.168.99.101

- Again on the second VM (Ubuntu) we establish another network interface:
- ◆ Host-Only Adaptor (name: vboxnet0/ enp0s3)



```
Activities Terminal नवम्बर 2 21:09
rikesh@VirtualBox: ~
rikesh@VirtualBox:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.99.102 netmask 255.255.255.0 broadcast 192.168.99.255
    inet6 fe80::a4e4:a44e:7325:6871 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:f6:9b:3d txqueuelen 1000 (Ethernet)
    RX packets 3541 bytes 4840293 (4.8 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1019 bytes 74495 (74.4 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 1145 bytes 85770 (85.7 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1145 bytes 85770 (85.7 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

rikesh@VirtualBox:~$
```

enp0s3 = 192.168.99.102

→ Now we configure *enp0s8* on VM1(centOS) using the following commands:

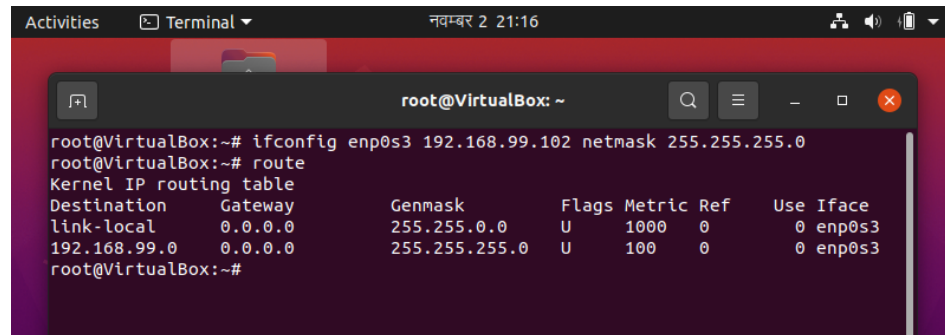
- ◆ `ifconfig enp0s8 192.168.99.101 netmask 255.255.255.0`
- ◆ `route`

```
[root@localhost rikesh]# ifconfig enp0s8 192.168.99.101 netmask 255.255.255.0
[root@localhost rikesh]# route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
default gateway 0.0.0.0 UG 100 0 0 enp0s3
192.168.99.0 0.0.0.0 255.255.255.0 U 101 0 0 enp0s8
192.168.122.0 0.0.0.0 255.255.255.0 U 0 0 0 virbr0
192.168.254.0 0.0.0.0 255.255.255.0 U 100 0 0 enp0s3
[root@localhost rikesh]#
```

After using the `route` command we can see the IP routing table. From the above table the network '192.168.254.0' is the network that we are using for internet access (enp0s3 card of VM1) and the network '192.168.99.0' represents the network that links out VM1(centOS) with VM2(Ubuntu) using the enp0s8 card of VM1.

→ Now we configure *enp0s3* on VM2(Ubuntu) using the following commands:

- ◆ `ifconfig enp0s3 192.168.99.102 netmask 255.255.255.0`
- ◆ `route`

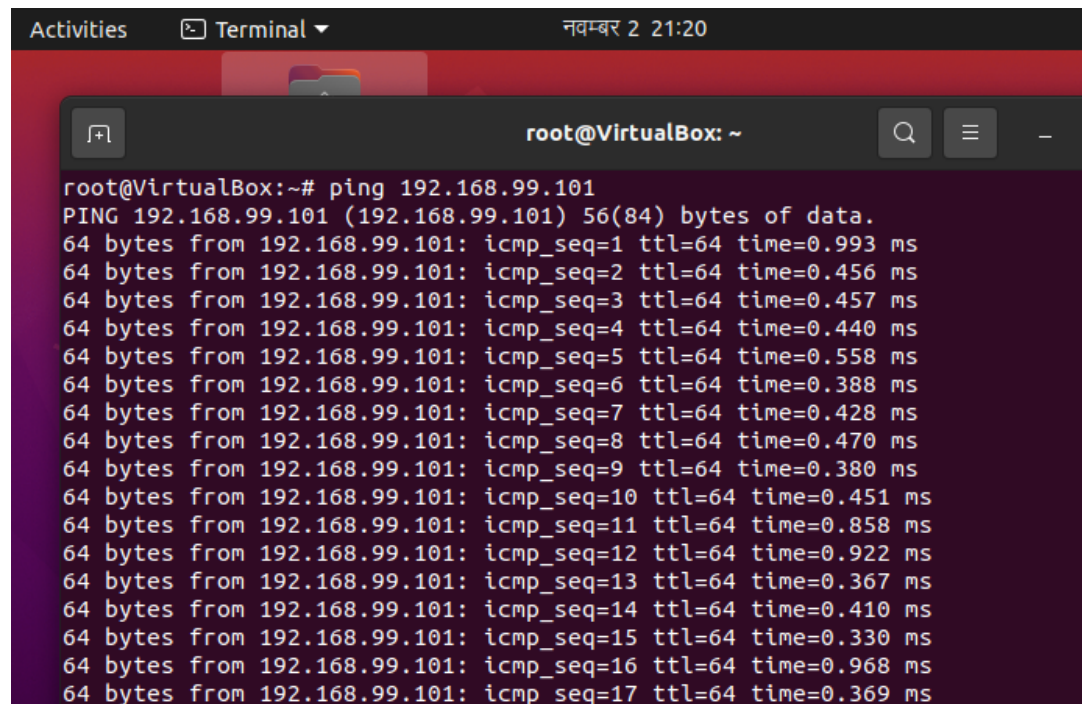


A terminal window titled 'root@VirtualBox: ~' showing the execution of network configuration commands. The first command is `ifconfig enp0s3 192.168.99.102 netmask 255.255.255.0`, and the second is `route`. The output of the `route` command displays the kernel IP routing table.

```
root@VirtualBox:~# ifconfig enp0s3 192.168.99.102 netmask 255.255.255.0
root@VirtualBox:~# route
Kernel IP routing table
Destination    Gateway         Genmask         Flags Metric Ref    Use Iface
link-local     0.0.0.0         255.255.0.0     U        1000  0      0 enp0s3
192.168.99.0   0.0.0.0         255.255.255.0   U        100   0      0 enp0s3
root@VirtualBox:~#
```

After using the `route` we can see the IP routing table. Here the network '192.168.99.0' represents the network that links VM2(Ubuntu) with VM1(CentOS) using the *enp0s3* card of VM2.

→ Now lets ping VM1(Centos) to VM2(Ubuntu) and vice versa to verify the connection between VM1 and VM2:



A terminal window titled 'root@VirtualBox: ~' showing the execution of a `ping` command to 192.168.99.101. The output shows 17 successful ping attempts with varying response times.

```
root@VirtualBox:~# ping 192.168.99.101
PING 192.168.99.101 (192.168.99.101) 56(84) bytes of data.
64 bytes from 192.168.99.101: icmp_seq=1 ttl=64 time=0.993 ms
64 bytes from 192.168.99.101: icmp_seq=2 ttl=64 time=0.456 ms
64 bytes from 192.168.99.101: icmp_seq=3 ttl=64 time=0.457 ms
64 bytes from 192.168.99.101: icmp_seq=4 ttl=64 time=0.440 ms
64 bytes from 192.168.99.101: icmp_seq=5 ttl=64 time=0.558 ms
64 bytes from 192.168.99.101: icmp_seq=6 ttl=64 time=0.388 ms
64 bytes from 192.168.99.101: icmp_seq=7 ttl=64 time=0.428 ms
64 bytes from 192.168.99.101: icmp_seq=8 ttl=64 time=0.470 ms
64 bytes from 192.168.99.101: icmp_seq=9 ttl=64 time=0.380 ms
64 bytes from 192.168.99.101: icmp_seq=10 ttl=64 time=0.451 ms
64 bytes from 192.168.99.101: icmp_seq=11 ttl=64 time=0.858 ms
64 bytes from 192.168.99.101: icmp_seq=12 ttl=64 time=0.922 ms
64 bytes from 192.168.99.101: icmp_seq=13 ttl=64 time=0.367 ms
64 bytes from 192.168.99.101: icmp_seq=14 ttl=64 time=0.410 ms
64 bytes from 192.168.99.101: icmp_seq=15 ttl=64 time=0.330 ms
64 bytes from 192.168.99.101: icmp_seq=16 ttl=64 time=0.968 ms
64 bytes from 192.168.99.101: icmp_seq=17 ttl=64 time=0.369 ms
```

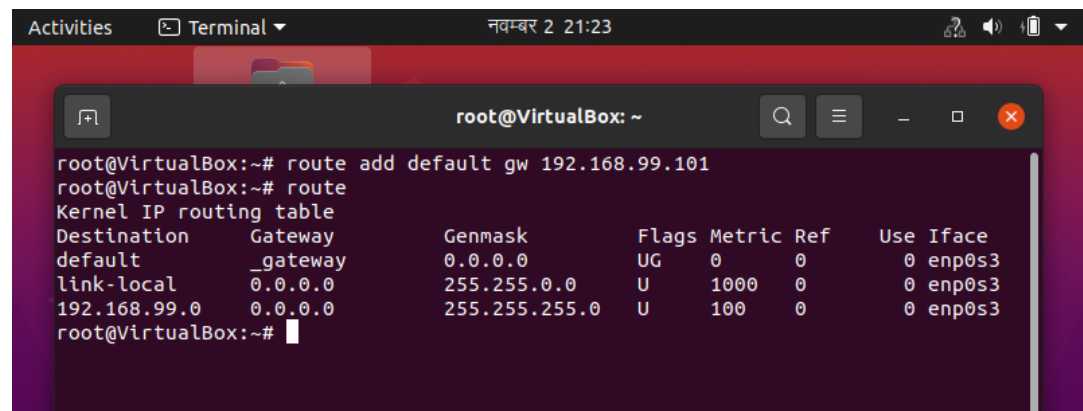
The IP address '192.168.99.101' belongs to VM1(CentOS).


```
root@localhost rikeshl# ping 192.168.99.102
PING 192.168.99.102 (192.168.99.102) 56(84) bytes of data:
64 bytes from 192.168.99.102: icmp_seq=1 ttl=64 time=0.595 ms
64 bytes from 192.168.99.102: icmp_seq=2 ttl=64 time=0.397 ms
64 bytes from 192.168.99.102: icmp_seq=3 ttl=64 time=0.412 ms
64 bytes from 192.168.99.102: icmp_seq=4 ttl=64 time=0.808 ms
64 bytes from 192.168.99.102: icmp_seq=5 ttl=64 time=0.372 ms
64 bytes from 192.168.99.102: icmp_seq=6 ttl=64 time=0.398 ms
64 bytes from 192.168.99.102: icmp_seq=7 ttl=64 time=0.370 ms
64 bytes from 192.168.99.102: icmp_seq=8 ttl=64 time=0.348 ms
64 bytes from 192.168.99.102: icmp_seq=9 ttl=64 time=0.361 ms
64 bytes from 192.168.99.102: icmp_seq=10 ttl=64 time=0.367 ms
64 bytes from 192.168.99.102: icmp_seq=11 ttl=64 time=0.369 ms
64 bytes from 192.168.99.102: icmp_seq=12 ttl=64 time=0.340 ms
64 bytes from 192.168.99.102: icmp_seq=13 ttl=64 time=0.438 ms
64 bytes from 192.168.99.102: icmp_seq=14 ttl=64 time=0.794 ms
64 bytes from 192.168.99.102: icmp_seq=15 ttl=64 time=0.822 ms
64 bytes from 192.168.99.102: icmp_seq=16 ttl=64 time=0.389 ms
64 bytes from 192.168.99.102: icmp_seq=17 ttl=64 time=0.366 ms
64 bytes from 192.168.99.102: icmp_seq=18 ttl=64 time=0.958 ms
64 bytes from 192.168.99.102: icmp_seq=19 ttl=64 time=0.908 ms
64 bytes from 192.168.99.102: icmp_seq=20 ttl=64 time=0.406 ms
64 bytes from 192.168.99.102: icmp_seq=21 ttl=64 time=0.951 ms
64 bytes from 192.168.99.102: icmp_seq=22 ttl=64 time=0.445 ms
64 bytes from 192.168.99.102: icmp_seq=23 ttl=64 time=0.354 ms
64 bytes from 192.168.99.102: icmp_seq=24 ttl=64 time=0.429 ms
```

The IP address '192.168.99.102' belongs to VM2(Ubuntu).

→ Now we configure VM2(Ubuntu) to use enp0s8 from VM1(CentOS) using following command:

- ◆ route add default gw 192.168.99.101
- ◆ route



```
root@VirtualBox: ~
root@VirtualBox:~# route add default gw 192.168.99.101
root@VirtualBox:~# route
Kernel IP routing table
Destination    Gateway         Genmask         Flags Metric Ref    Use Iface
default        _gateway        0.0.0.0         UG    0      0      0 enp0s3
link-local     0.0.0.0         255.255.0.0     U     1000   0      0 enp0s3
192.168.99.0   0.0.0.0         255.255.255.0   U     100    0      0 enp0s3
root@VirtualBox:~#
```

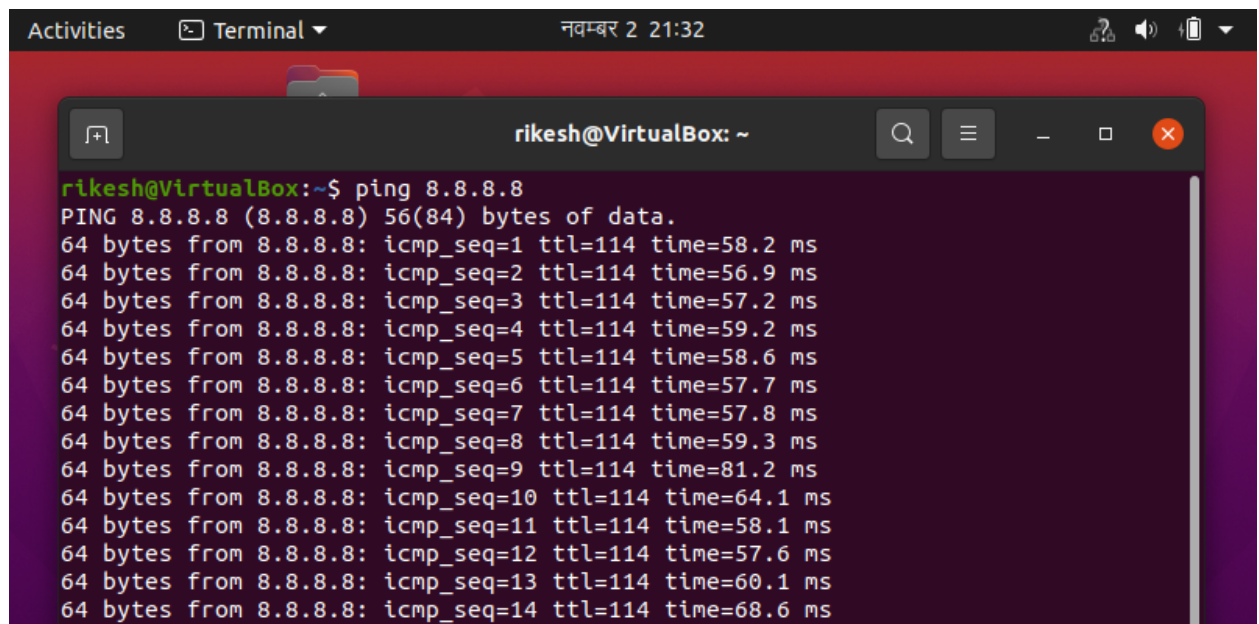
→ Finally we can share the connection between VM1(CentOS) and VM2(Ubuntu) using the following commands:

- ◆ `modprobe iptable_nat`
- ◆ `echo 1 > /proc/sys/net/ipv4/ip_forward`
- ◆ `iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE`
- ◆ `iptables -A FORWARD -i enp0s8 -j ACCEPT`

```
[root@localhost rikesh]# modprobe iptable_nat
[root@localhost rikesh]# echo 1 > /proc/sys/net/ipv4/ip_forward
[root@localhost rikesh]# iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE
[root@localhost rikesh]# iptables -A FORWARD -i enp0s8 -j ACCEPT
[root@localhost rikesh]#
```

We have established a proper connection between VM1 and VM2 sharing the internet connection from CentOS to Ubuntu through NAT. This process can be verified by executing the following ping command in the terminal of VM2(Ubuntu):

- `ping 8.8.8.8`

A screenshot of a terminal window titled 'Terminal' with a date and time of 'नवम्बर 2 21:32'. The terminal shows a user 'rikesh' at a 'VirtualBox' prompt. The user has executed the command 'ping 8.8.8.8'. The output shows 14 successful ping requests, each returning 64 bytes from 8.8.8.8 with varying times between 56.9 ms and 81.2 ms. The terminal window has standard Ubuntu window controls and a search icon.

```
rikesh@VirtualBox:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=114 time=58.2 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=114 time=56.9 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=114 time=57.2 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=114 time=59.2 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=114 time=58.6 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=114 time=57.7 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=114 time=57.8 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=114 time=59.3 ms
64 bytes from 8.8.8.8: icmp_seq=9 ttl=114 time=81.2 ms
64 bytes from 8.8.8.8: icmp_seq=10 ttl=114 time=64.1 ms
64 bytes from 8.8.8.8: icmp_seq=11 ttl=114 time=58.1 ms
64 bytes from 8.8.8.8: icmp_seq=12 ttl=114 time=57.6 ms
64 bytes from 8.8.8.8: icmp_seq=13 ttl=114 time=60.1 ms
64 bytes from 8.8.8.8: icmp_seq=14 ttl=114 time=68.6 ms
```

'8.8.8.8' is one of the public DNS servers from Google.