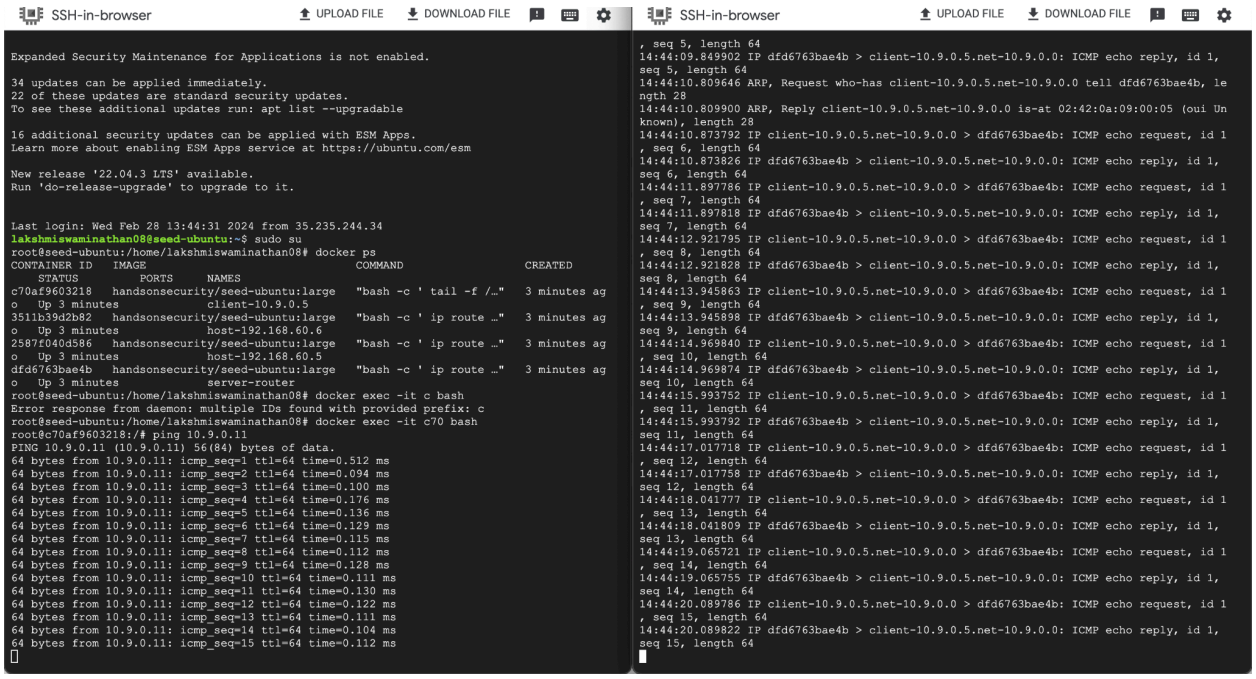
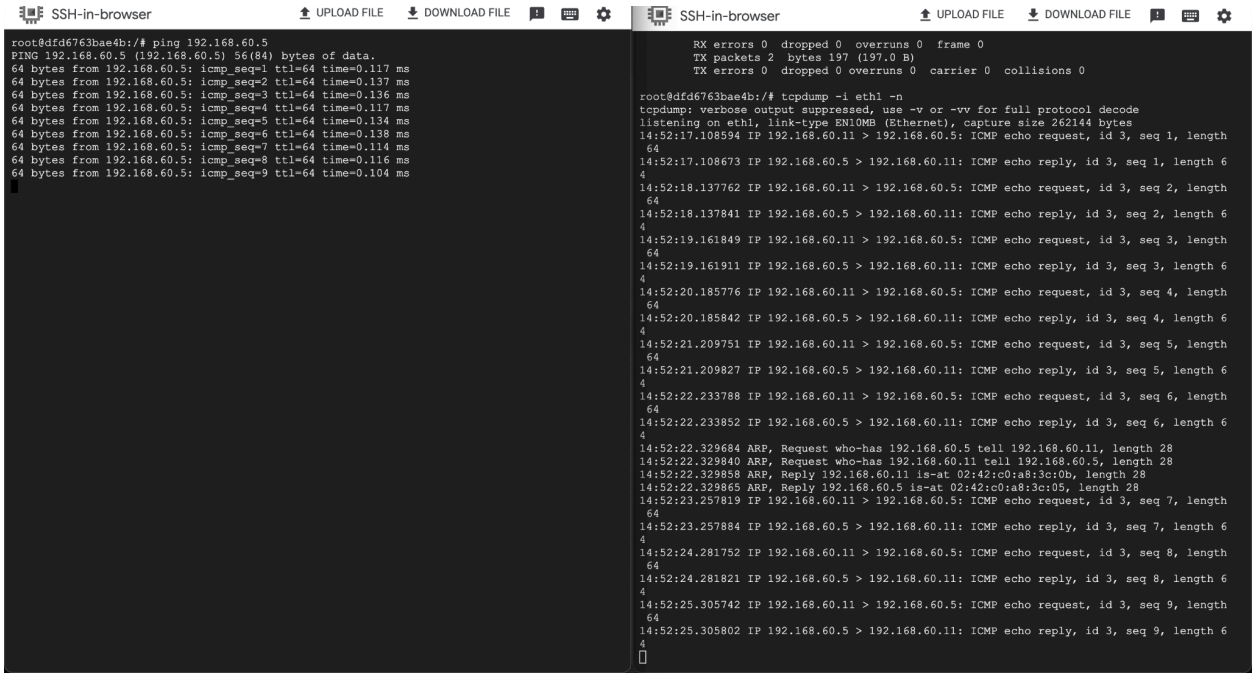
Task 1:

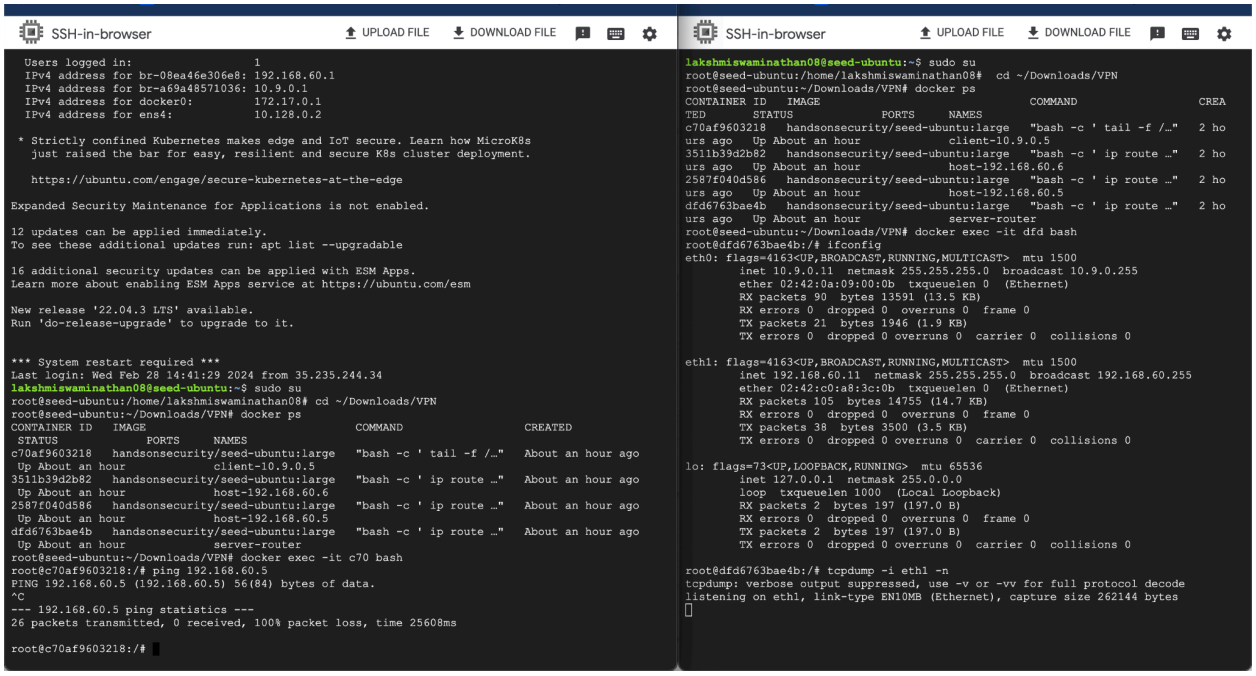
Host U can communicate with VPN Server.



VPN Server can communicate with Host V.

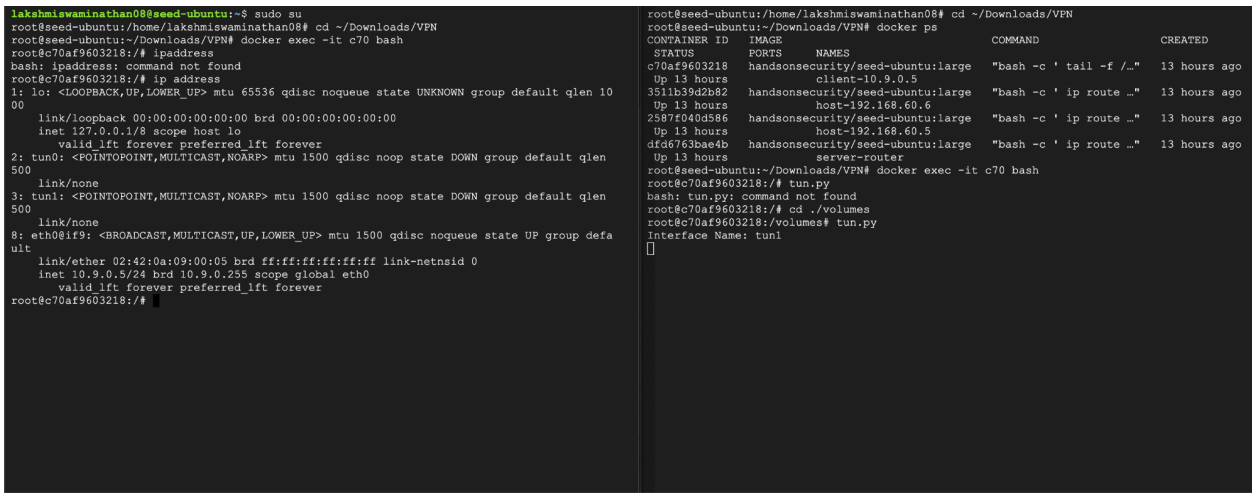


Host U is not able to communicate with Host V.



Task 2: Create and Configure TUN Interface

Task 2.a: Name of the Interface



Task 2.b: Set up the TUN Interface

SSH-in-browser — UPLOAD FILE — DOWNLOAD FILE

```
root@c70af9603218:/# ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 10
00
    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
2: tun0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN group default qlen
500
    link/none
4: tun1: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN
group default qlen 500
    link/none
    inet 192.168.53.99/24 scope global tun1
       valid_lft forever preferred_lft forever
8: eth0@if9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group defa
ult
    link/ether 02:42:0a:09:00:05 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.9.0.5/24 brd 10.9.0.255 scope global eth0
       valid_lft forever preferred_lft forever
root@c70af9603218:/#
```

SSH-in-browser — UPLOAD FILE — DOWNLOAD FILE

```
root@seed-ubuntu:/home/lakshmiswaminathan08# cd ~/Downloads/VPN
root@seed-ubuntu:~/Downloads/VPN# docker ps
CONTAINER ID   IMAGE                                COMMAND                   CREATED
 STATUS          PORTS     NAMES
c70af9603218   handsonsecurity/seed-ubuntu:large   "bash -c ' tail -f /…"   13 hours ago
 Up 13 hours              client-10.9.0.5
3511b39d2b82   handsonsecurity/seed-ubuntu:large   "bash -c ' ip route …"   13 hours ago
 Up 13 hours              host-192.168.60.6
2587f040d586   handsonsecurity/seed-ubuntu:large   "bash -c ' ip route …"   13 hours ago
 Up 13 hours              host-192.168.60.5
dfd6763bae4b   handsonsecurity/seed-ubuntu:large   "bash -c ' ip route …"   13 hours ago
 Up 13 hours              server-router
root@seed-ubuntu:~/Downloads/VPN# docker exec -it c70 bash
root@c70af9603218:/# tun.py
bash: tun.py: command not found
root@c70af9603218:/# cd ./volumes
root@c70af9603218:/volumes# tun.py
Interface Name: tun1
^CTraceback (most recent call last):
  File "./tun.py", line 24, in <module>

KeyboardInterrupt

root@c70af9603218:/volumes# tun.py
Interface Name: tun1
```

Task 2.c: Read from the TUN Interface

```
root@c70af9603218:/# ping 192.168.53.1
PING 192.168.53.1 (192.168.53.1) 56(84) bytes of data.
```

```
root@c70af9603218:/volumes# tun.py
Interface Name: lakshmi0
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
```

```
root@c70af9603218:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
```

```
root@c70af9603218:/volumes# tun.py
Interface Name: lakshmi0
```

Task 2.d: Write to the TUN Interface

```
#!/usr/bin/env python3

import fcntl
import struct
import os
import time
from scapy.all import *

TUNSETIFF = 0x400454ca
IFF_TUN   = 0x0001
IFF_TAP   = 0x0002
IFF_NO_PI = 0x1000

# Create the tun interface
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'lakshmi%d', IFF_TUN | IFF_NO_PI)
ifname_bytes  = fcntl.ioctl(tun, TUNSETIFF, ifr)

# Get the interface name
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
print("Interface Name: {}".format(ifname))
os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

while True:
        # Get a packet from the tun interface
        packet = os.read(tun, 2048)
        if packet:
                ip = IP(packet)
                print(ip.summary())
                # Send out a spoof packet using the tun interface
                newip = IP(src='192.168.53.3', dst=ip.src)
                newpkt = newip/ip.payload
                os.write(tun, bytes(newpkt))
```

```
root@c70af9603218:/# ping 192.168.53.1
PING 192.168.53.1 (192.168.53.1) 56(84) bytes of data.
64 bytes from 192.168.53.3: icmp_seq=6 ttl=64 time=3.81 ms
64 bytes from 192.168.53.3: icmp_seq=7 ttl=64 time=3.57 ms
64 bytes from 192.168.53.3: icmp_seq=8 ttl=64 time=3.19 ms
64 bytes from 192.168.53.3: icmp_seq=9 ttl=64 time=3.29 ms
64 bytes from 192.168.53.3: icmp_seq=10 ttl=64 time=3.31 ms
64 bytes from 192.168.53.3: icmp_seq=11 ttl=64 time=3.35 ms
64 bytes from 192.168.53.3: icmp_seq=12 ttl=64 time=2.59 ms
64 bytes from 192.168.53.3: icmp_seq=13 ttl=64 time=3.25 ms
64 bytes from 192.168.53.3: icmp_seq=14 ttl=64 time=3.15 ms
```

```
root@c70af9603218:/volumes# tun.py
Interface Name: lakshmi0
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.3 echo-reply 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.3 echo-reply 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.3 echo-reply 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.3 echo-reply 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.3 echo-reply 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.3 echo-reply 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.3 echo-reply 0 / Raw
```

```
root@c70af9603218:/# ping 192.168.53.1
PING 192.168.53.1 (192.168.53.1) 56(84) bytes of data.
```

```
root@c70af9603218:/volumes# tun.py
Interface Name: lakshmi0
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
IP / ICMP 192.168.53.99 > 192.168.53.1 echo-request 0 / Raw
```

ssh.cloud.google.com/v2/ssh/projects/seed-ubuntu-410600/zones/us-central1-a/instances/seed-ubuntu?authuser=0&hl=en_US&projectNumber=633632528987&useAdminProx...

ssh.cloud.google.com/v2/ssh/projects/seed-ubuntu-410600/zones/us-central1-a/instances/seed-ubuntu?authuser=0&hl=en_US&projectNumber=633632528987&useAd...

SSH-in-browser

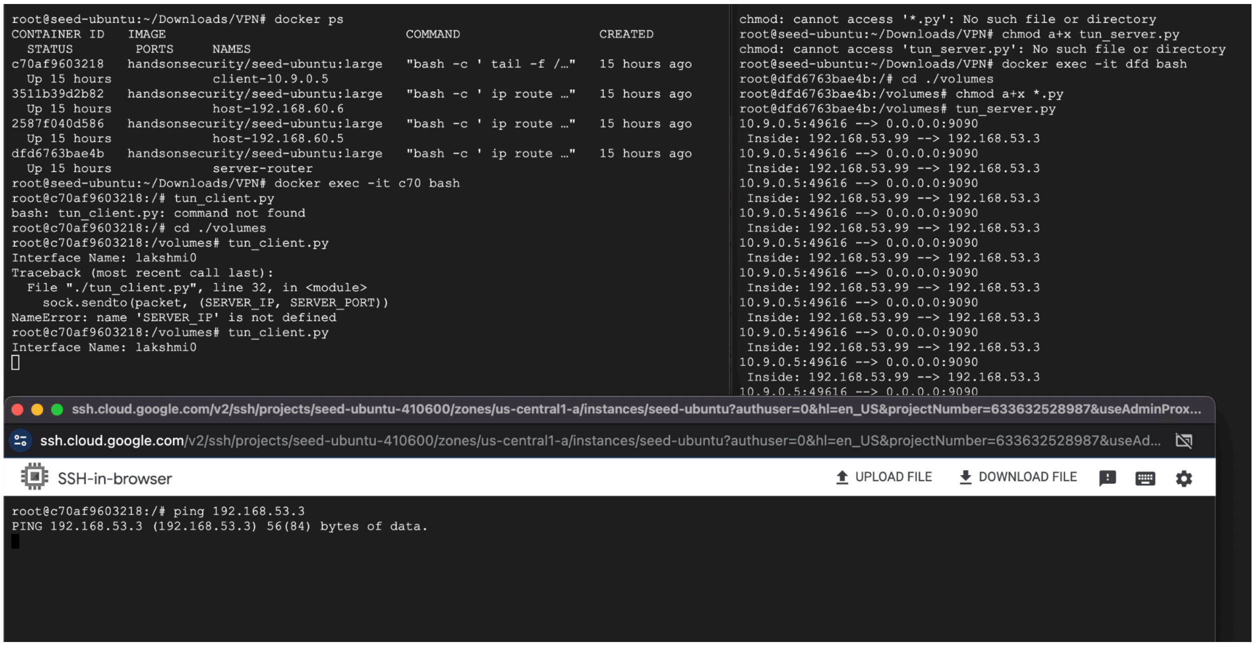
UPLOAD FILE DOWNLOAD FILE

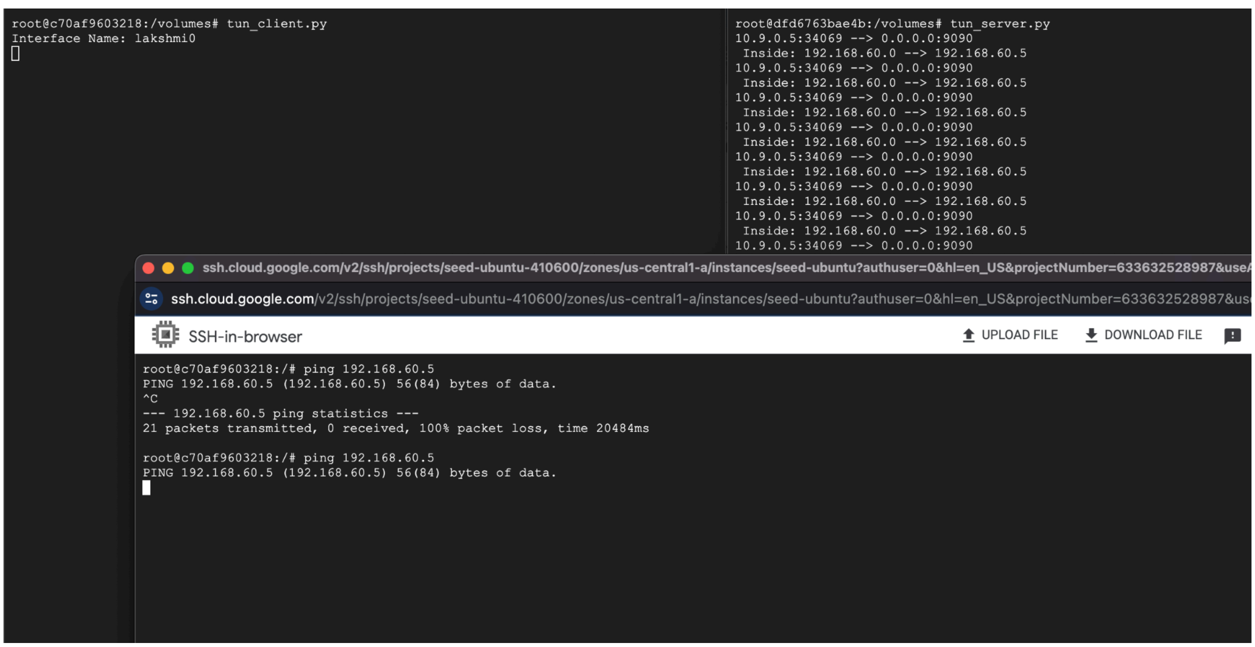
```
        TX packets 8  bytes 652 (652.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@c70af9603218:/# tcpdump -i lakshmi0 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on lakshmi0, link-type RAW (Raw IP), capture size 262144 bytes
04:35:39.385736 IP 192.168.53.99 > 192.168.53.1: ICMP echo request, id 21, seq 36, length 64
04:35:39.386972 IP [|ip]
04:35:40.409741 IP 192.168.53.99 > 192.168.53.1: ICMP echo request, id 21, seq 37, length 64
04:35:40.411130 IP [|ip]
04:35:41.433729 IP 192.168.53.99 > 192.168.53.1: ICMP echo request, id 21, seq 38, length 64
04:35:41.434705 IP [|ip]
04:35:42.457760 IP 192.168.53.99 > 192.168.53.1: ICMP echo request, id 21, seq 39, length 64
04:35:42.459022 IP [|ip]
04:35:43.481693 IP 192.168.53.99 > 192.168.53.1: ICMP echo request, id 21, seq 40, length 64
04:35:43.483091 IP [|ip]
04:35:44.505707 IP 192.168.53.99 > 192.168.53.1: ICMP echo request, id 21, seq 41, length 64
04:35:44.506711 IP [|ip]
04:35:45.529725 IP 192.168.53.99 > 192.168.53.1: ICMP echo request, id 21, seq 42, length 64
04:35:45.530694 IP [|ip]
04:35:46.553797 IP 192.168.53.99 > 192.168.53.1: ICMP echo request, id 21, seq 43, length 64
04:35:46.554809 IP [|ip]
04:35:47.577758 IP 192.168.53.99 > 192.168.53.1: ICMP echo request, id 21, seq 44, length 64
04:35:47.578790 IP [|ip]
04:35:48.601763 IP 192.168.53.99 > 192.168.53.1: ICMP echo request, id 21, seq 45, length 64
```

Task 2 -Final code:

```
#!/usr/bin/env python3

import fcntl
import struct
import os
import time
from scapy.all import *

TUNSETIFF = 0x400454ca
IFF_TUN   = 0x0001
IFF_TAP   = 0x0002
IFF_NO_PI = 0x1000

# Create the tun interface
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'lakshmi%d', IFF_TUN | IFF_NO_PI)
ifname_bytes  = fcntl.ioctl(tun, TUNSETIFF, ifr)

# Get the interface name
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
print("Interface Name: {}".format(ifname))
os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

while True:
        # Get a packet from the tun interface
        packet = os.read(tun, 2048)
        if packet:
                ip = IP(packet)
                print(ip.summary())
                # Send out a spoof packet using the tun interface
                newip = IP(src='192.168.53.3', dst=ip.src)
                newpkt = newip/ip.payload
                arb_data=b'Any arbitrary data'
                os.write(tun, arb_data)
```

Task 3: Send the IP Packet to VPN Server Through a Tunnel





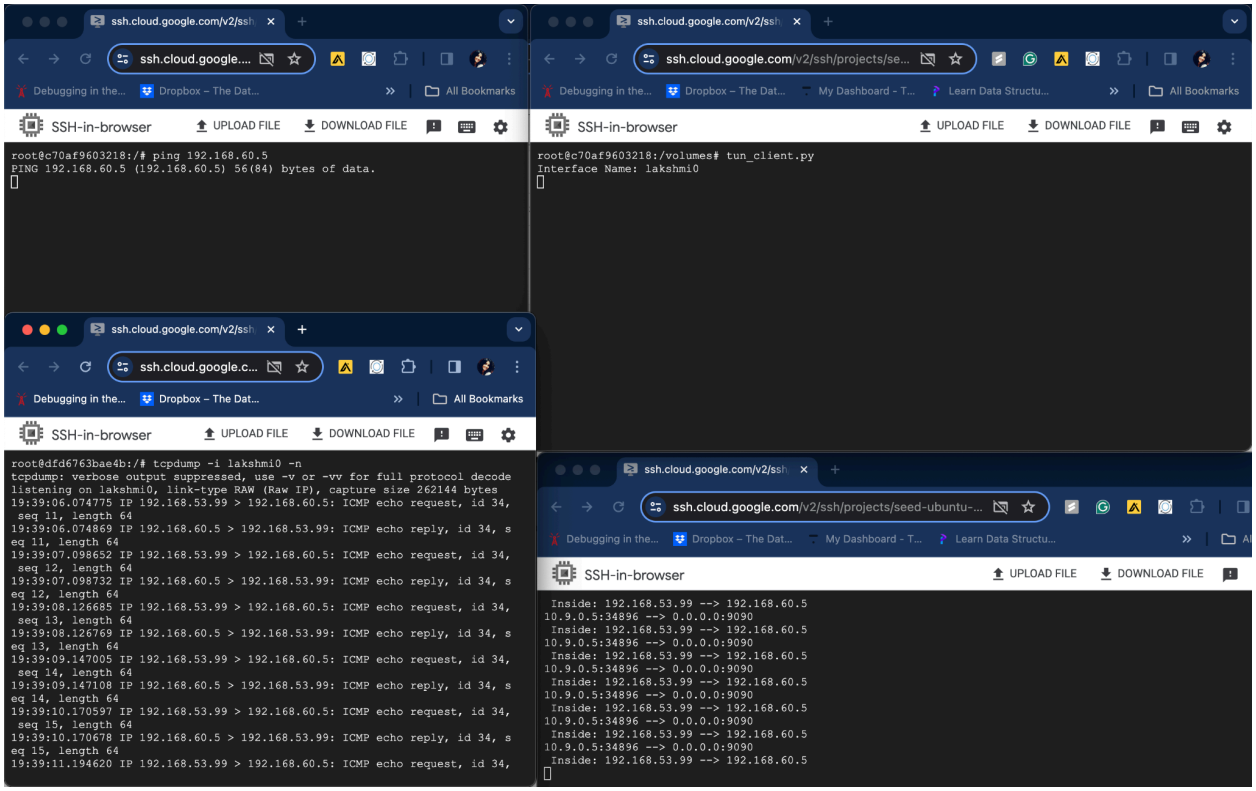
tun_client code

```
#!/usr/bin/env python3

import fcntl
import struct
import os
import time
from scapy.all import *

TUNSETIFF = 0x400454ca
IFF_TUN   = 0x0001
IFF_TAP   = 0x0002
IFF_NO_PI = 0x1000

# Create the tun interface
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'lakshmi%d', IFF_TUN | IFF_NO_PI)
ifname_bytes  = fcntl.ioctl(tun, TUNSETIFF, ifr)

# Get the interface name
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
print("Interface Name: {}".format(ifname))
os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
os.system("ip addr add 192.168.60.0/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

# Create UDP socket
SERVER_IP='10.9.0.11'
SERVER_PORT=9090
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
while True:
# Get a packet from the tun interface
        packet = os.read(tun, 2048)
        if packet:
        # Send the packet via the tunnel
                sock.sendto(packet, (SERVER_IP, SERVER_PORT))
```

tun_server code

```
#!/usr/bin/env python3
from scapy.all import *
IP_A = "0.0.0.0"
PORT = 9090
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind((IP_A, PORT))
while True:
        data, (ip, port) = sock.recvfrom(2048)
        print("{}:{} --> {}:{}".format(ip, port, IP_A, PORT))
        pkt = IP(data)
        print(" Inside: {} --> {}".format(pkt.src, pkt.dst))
```

Task 4: Set Up the VPN Server




```
root@2587f040d586:/# tcpdump -i eth0 -n
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
19:28:41.530859 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 33, seq 5, length 64
19:28:41.530892 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 33, seq 5, length 64
19:28:42.554939 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 33, seq 6, length 64
19:28:42.554968 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 33, seq 6, length 64
19:28:42.685662 ARP, Request who-has 192.168.60.11 tell 192.168.60.5, length 28
19:28:42.686065 ARP, Request who-has 192.168.60.5 tell 192.168.60.11, length 28
19:28:42.686093 ARP, Reply 192.168.60.5 is-at 02:42:c0:a8:3c:05, length 28
19:28:42.686095 ARP, Reply 192.168.60.11 is-at 02:42:c0:a8:3c:0b, length 28
19:28:43.578774 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 33, seq 7, length 64
19:28:43.578806 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 33, seq 7, length 64
19:28:44.602584 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 33, seq 8, length 64
19:28:44.602627 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 33, seq 8, length 64
19:28:45.626632 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 33, seq 9, length 64
19:28:45.626668 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 33, seq 9, length 64
19:28:46.650663 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 33, seq 10, length 64
19:28:46.650698 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 33, seq 10, length 64
19:28:47.674597 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 33, seq 11, length 64
19:28:47.674623 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 33, seq 11, length 64
19:28:48.698619 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 33, seq 12, length 64
19:28:48.698726 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 33, seq 12, length 64
19:28:49.722699 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 33, seq 13, length 64
19:28:49.722735 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 33, seq 13, length 64
19:28:50.746764 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 33, seq 14, length 64
19:28:50.746802 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 33, seq 14, length 64
19:28:51.770876 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 33, seq 15, length 64
19:28:51.770908 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 33, seq 15, length 64
19:28:52.794601 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 33, seq 16, length 64
19:28:52.794634 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 33, seq 16, length 64
19:28:53.818821 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 33, seq 17, length 64
19:28:53.818925 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 33, seq 17, length 64
19:28:54.843191 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 33, seq 18, length 64
19:28:54.843227 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 33, seq 18, length 64
19:28:55.866755 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 33, seq 19, length 64
19:28:55.866788 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 33, seq 19, length 64
19:28:56.890842 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 33, seq 20, length 64
19:28:56.890875 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 33, seq 20, length 64
19:28:57.914522 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 33, seq 21, length 64
19:28:57.914553 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 33, seq 21, length 64
19:28:58.938608 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 33, seq 22, length 64
19:28:58.938644 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 33, seq 22, length 64
19:28:59.962661 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 33, seq 23, length 64
19:28:59.962697 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 33, seq 23, length 64
19:29:00.986927 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 33, seq 24, length 64
19:29:00.986961 IP 192.168.60.5 > 192.168.53.99: ICMP echo reply, id 33, seq 24, length 64
19:29:02.010644 IP 192.168.53.99 > 192.168.60.5: ICMP echo request, id 33, seq 25, length 64
```

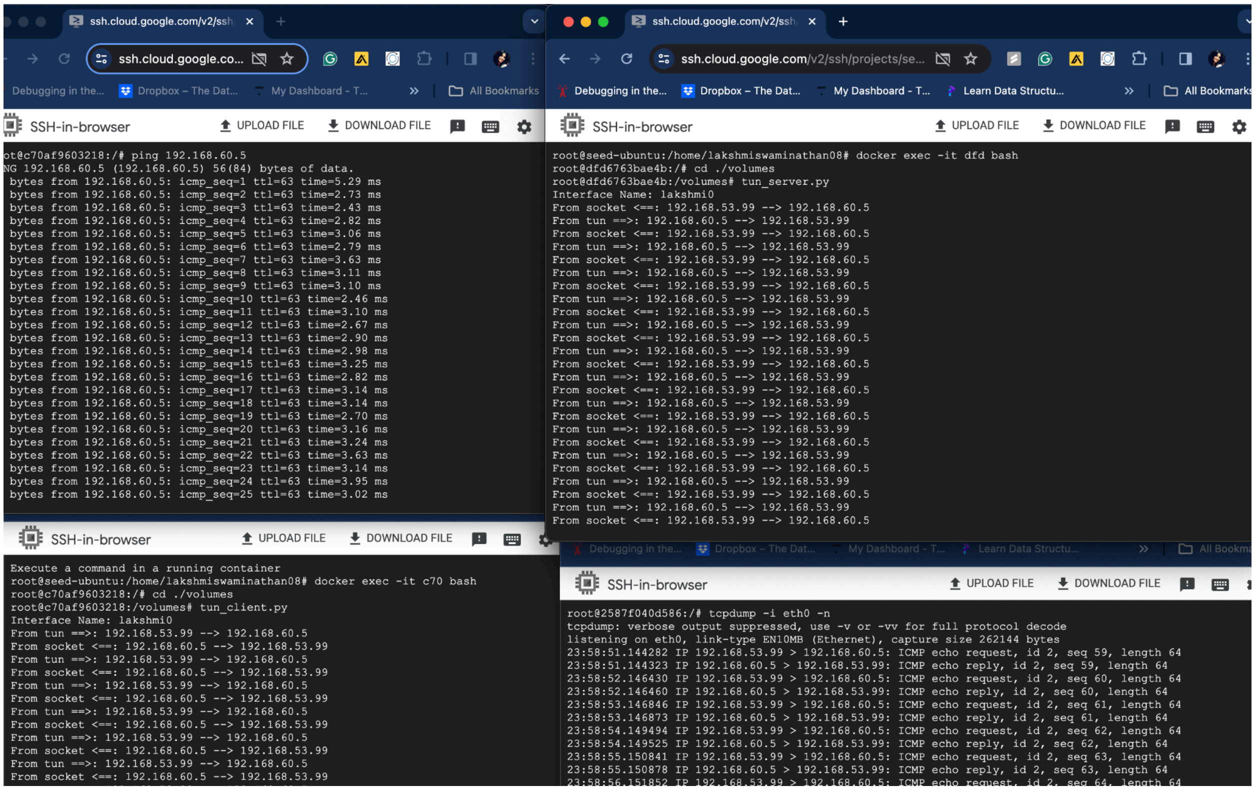
tun_server code

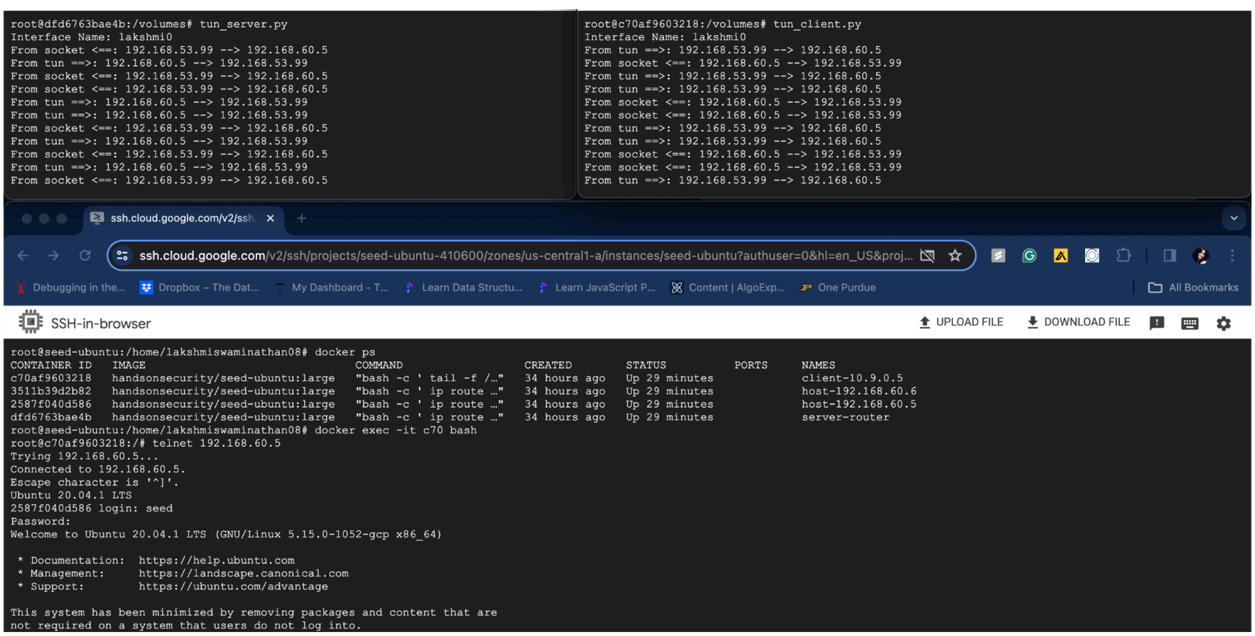
```
#!/usr/bin/env python3
import fcntl
import struct
import os
import time
from scapy.all import *
TUNSETIFF = 0x400454ca
IFF_TUN   = 0x0001
IFF_TAP   = 0x0002
IFF_NO_PI = 0x1000

# Create the tun interface
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'lakshmi%d', IFF_TUN | IFF_NO_PI)
ifname_bytes  = fcntl.ioctl(tun, TUNSETIFF, ifr)

# Get the interface name
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
print("Interface Name: {}".format(ifname))
os.system("ip addr add 192.168.53.50/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

IP_A = "0.0.0.0"
PORT = 9090
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind((IP_A, PORT))
while True:
        data, (ip, port) = sock.recvfrom(2048)
        print("{}:{} --> {}:{}".format(ip, port, IP_A, PORT))
        pkt = IP(data)
        print(" Inside: {} --> {}".format(pkt.src, pkt.dst))
        os.write(tun,bytes(pkt))
```

Tun_client code

```
#!/usr/bin/env python3

import fcntl
import struct
import os
import time
from scapy.all import *

TUNSETIFF = 0x400454ca
IFF_TUN   = 0x0001
IFF_TAP   = 0x0002
IFF_NO_PI = 0x1000

# Create the tun interface
tun = os.open("/dev/net/tun", os.O_RDWR)
ifr = struct.pack('16sH', b'lakshmi%d', IFF_TUN | IFF_NO_PI)
ifname_bytes  = fcntl.ioctl(tun, TUNSETIFF, ifr)

# Get the interface name
ifname = ifname_bytes.decode('UTF-8')[:16].strip("\x00")
print("Interface Name: {}".format(ifname))
os.system("ip addr add 192.168.53.99/24 dev {}".format(ifname))
os.system("ip link set dev {} up".format(ifname))

os.system("ip route add 192.168.60.0/24 dev {}".format(ifname))
# Create UDP socket
SERVER_IP='10.9.0.11'
SERVER_PORT=9090
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
while True:
# Get a packet from the tun interface
        packet = os.read(tun, 2048)
        if packet:
        # Send the packet via the tunnel
                sock.sendto(packet, (SERVER_IP, SERVER_PORT))
```

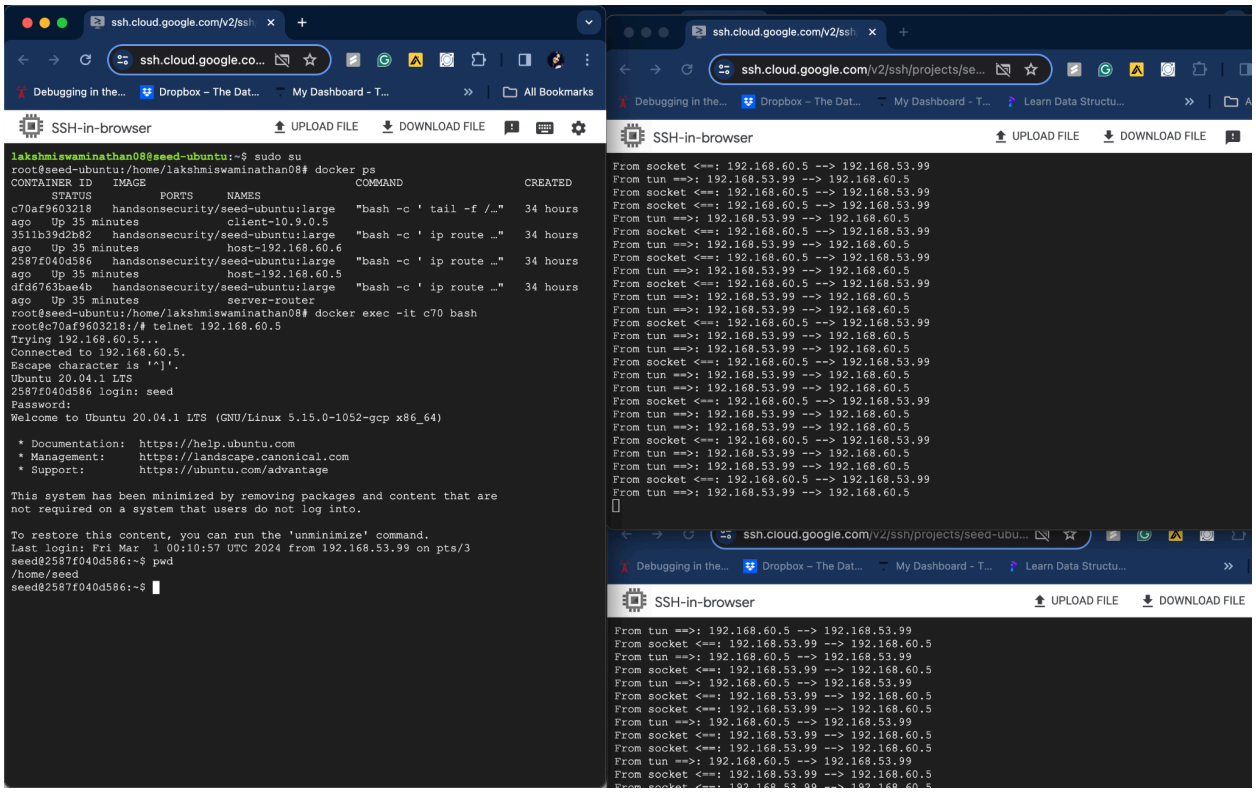
Task 5: Handling Traffic in Both Directions

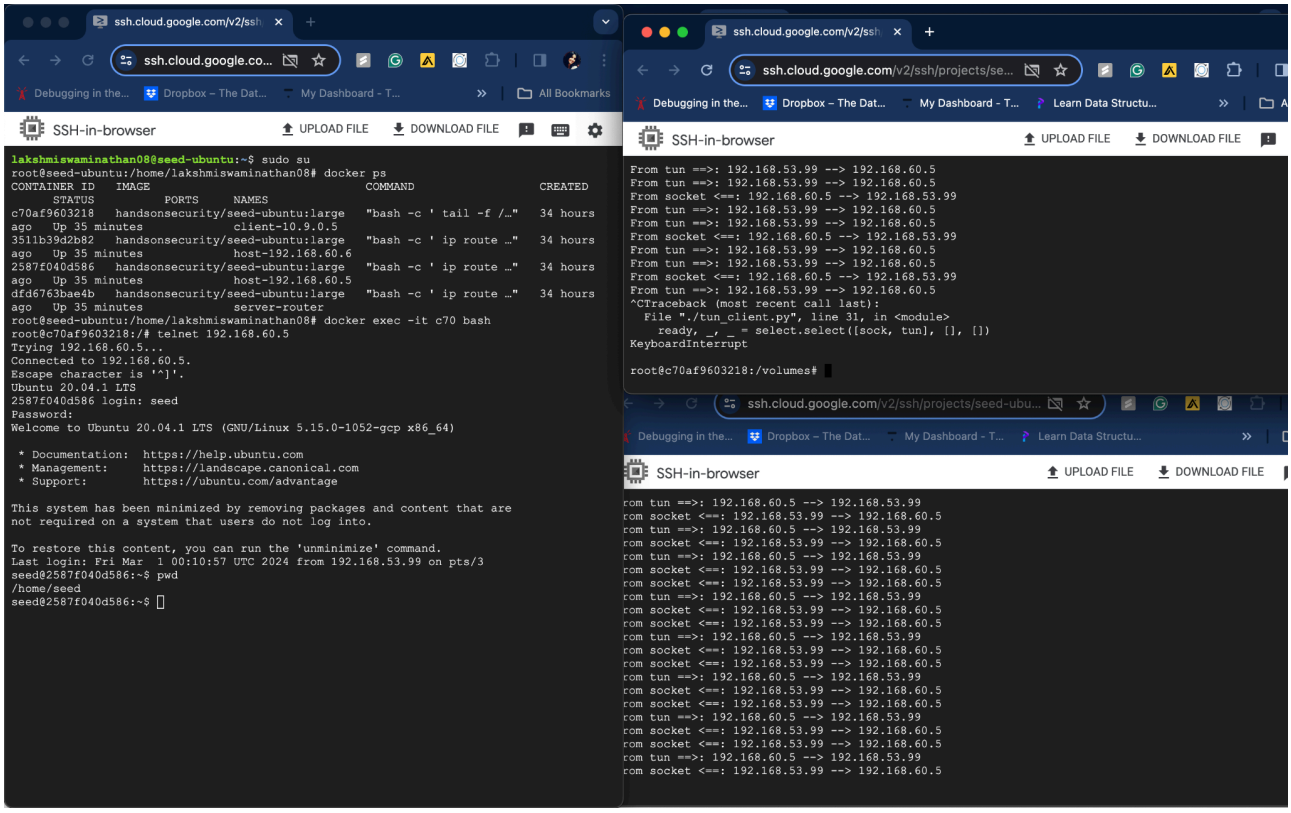




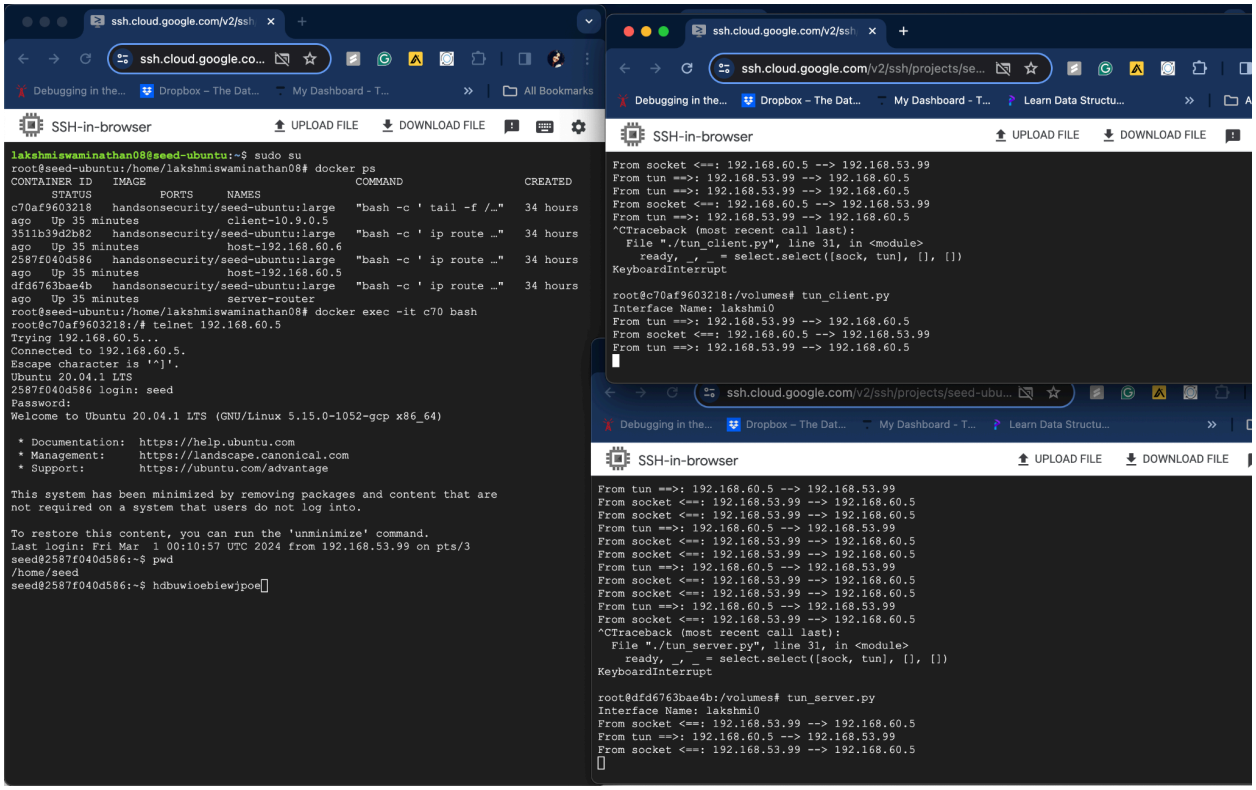
Task 6: Tunnel-Breaking Experiment

Telnet connection established between client and host within a private network(192.168.60.5)





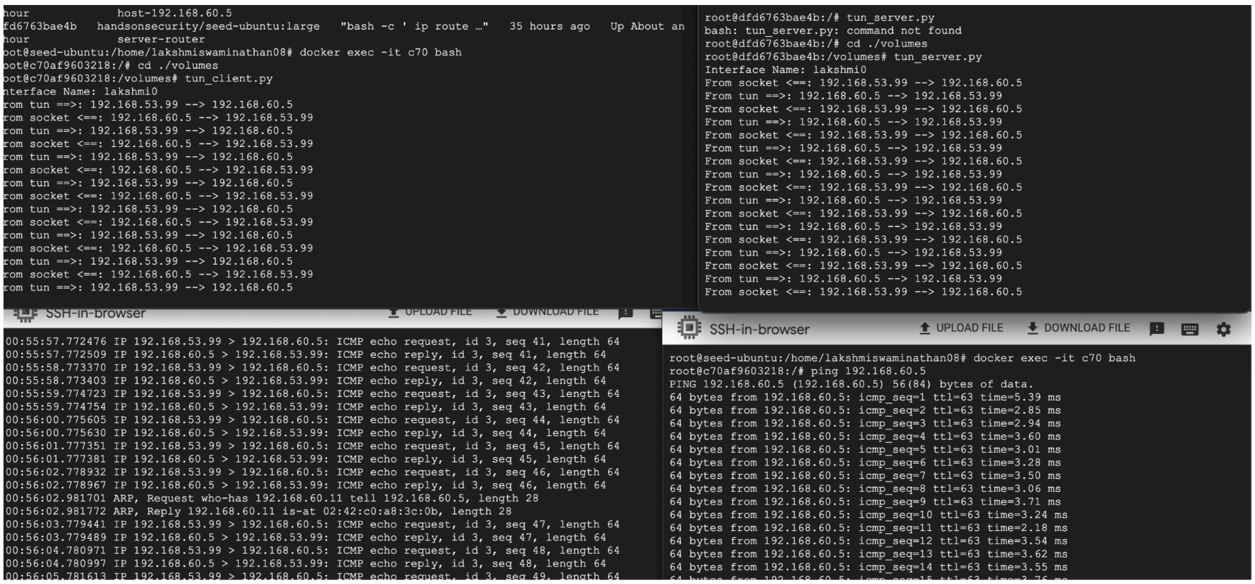
VPN Tunnel is broken by stopping tun_client/tun_server for a while and the telnet connection is frozen. Column't type any cmds



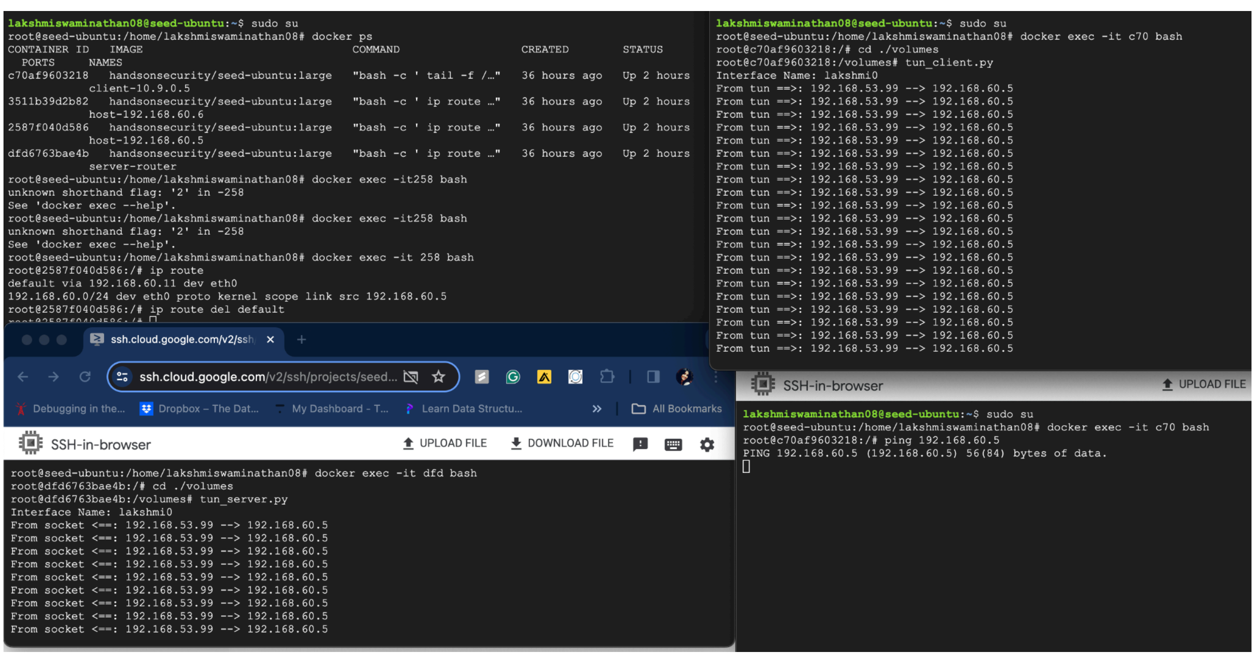
Once the connection is revived having the tunnel broken for a while, the text that was typed when the connection is broken is showing up.

Task 7: Routing Experiment on Host V

Default route for 192.168.60.5 is 192.168.60.11- Host u(client) is successfully able to ping host v.



After deleting the default route, connection broke between client and hosts in private network



Connection between client and private network is preserved once the default route is added back.