Virtualization and Cloud Computing
Assignment 1

Name: Rishabh Acharya
Roll Number: B22CS090

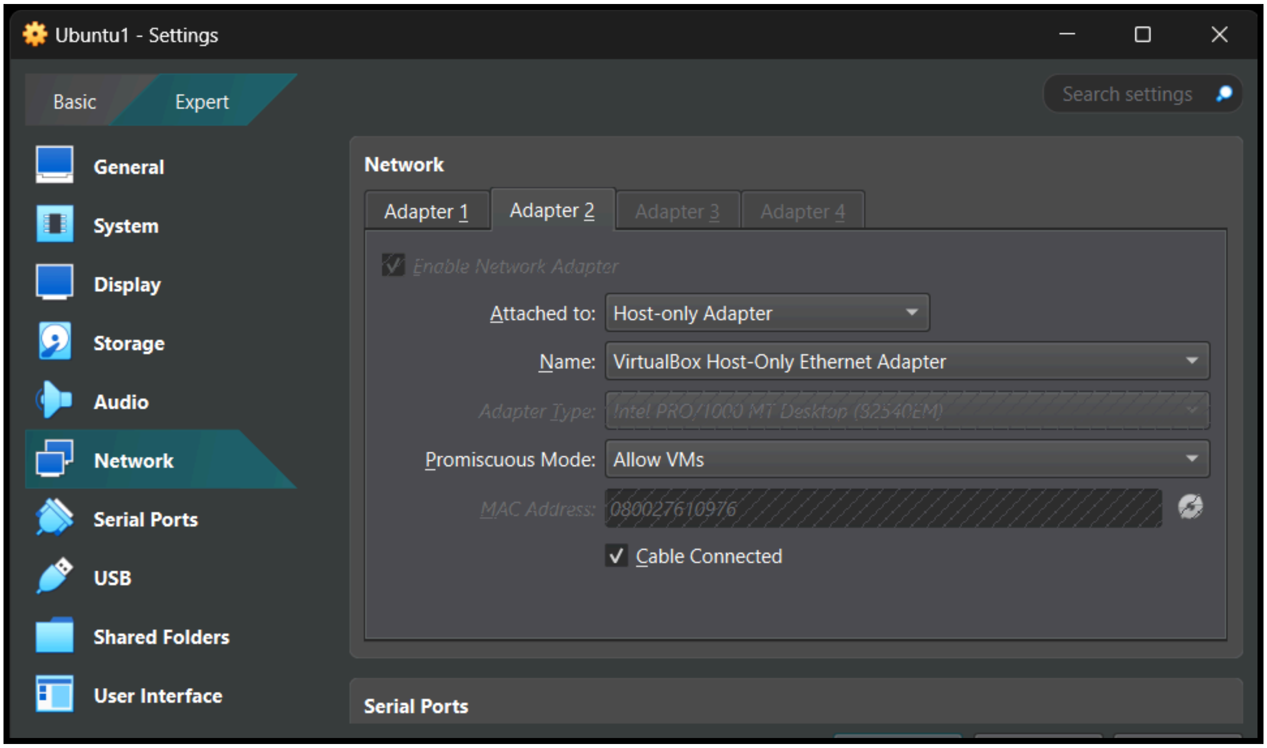
Installation of VirtualBox and creation of multiple VMs

- Firstly, we need to download VirtualBox from Oracle VM VirtualBox - Downloads
- Then, we need to download OS of our choice. I downloaded **Ubuntu 24.04.1 LTS** from Download Ubuntu Desktop
- Then, we install VirtualBox and choose a new VM to be created.
- We add **name, type, version, folder** where to create the virtual machine, and the **.iso file.**
- We assign memory size, number of processors, and virtual hard disk size to the VM. To my VMs, I have given **6GB RAM, 4 CPUs** and **20GB virtual hard disk size** to each.
- After this, I change **"Shared Clipboard"** and **"Drag and Drop"** options to **bidirectional** so that I can copy materials from one VM to another without any hassle.
- Now, the VM is set up but I need to install Linux inside the VM now.
- I set my language and keyboard within Linux, and choose **"normal installation."**
- In the installation type section, I have chosen **"Erase disk and install Ubuntu."** This doesn't erase all contents of my actual drive. It just erases contents from the virtual hard drive which has been allocated to the VM, for clean installation.
- Finally, I set my username and password and the installation begins.
- After installation, I restart.
- One useful feature to have is that when I change the size of my VM window, the VM screen should get adjusted automatically. But sometimes, this doesn't happen.
- To account for this, we have to do the steps as mentioned from **15:50** in this video

- After this, the VM screen gets adjusted automatically based on window size.
- Now one VM is set up with necessary settings.
- We can just **clone** it by right clicking on this VM and doing **“Full Clone”**
- This creates 2 instances of the VM and **both of them have their own identity.**

Configuration of network settings to connect the VMs

- To establish connection between 2 VMs, we use a **Host-Only Network.**
- A host-only network allows 2 VMs to communicate between themselves.
- Go to Tools and create a host-only network if not created already.
- Then, go to the network setting of each VM and set Adapter 2 to Host-Only Network so that Inter VM communication can happen.
- It should look like this:



Network

☑ Enable Network Adapter

Attached to: Host-only Adapter

Name: VirtualBox Host-Only Ethernet Adapter

Adapter Type: Intel PRO/1000 MT Desktop (82540EM)

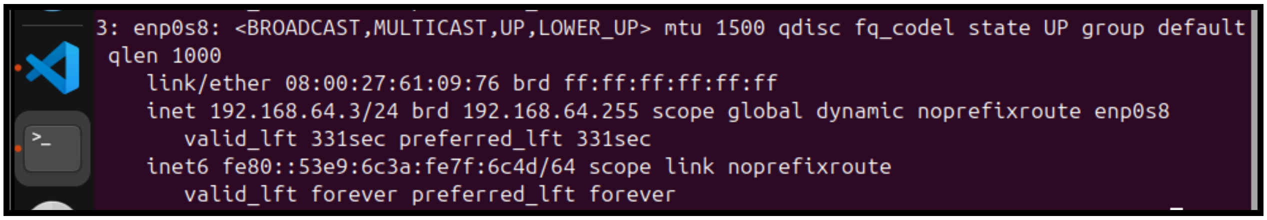
Promiscuous Mode: Allow VMs

MAC Address: 080027610976

☑ Cable Connected

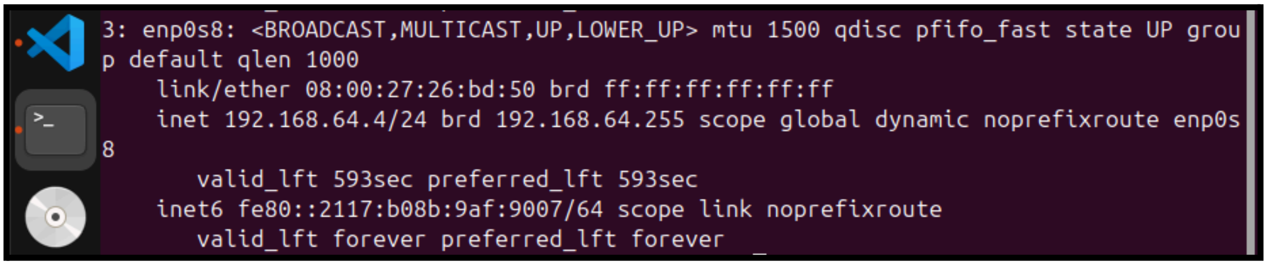
Serial Ports

- After such setup for both VMs, an IP address gets assigned to both the VMs. This can be verified by entering **“ip a”** to the terminal of both the VMs.



```
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default
 qlen 1000
    link/ether 08:00:27:61:09:76 brd ff:ff:ff:ff:ff:ff
    inet 192.168.64.3/24 brd 192.168.64.255 scope global dynamic noprefixroute enp0s8
       valid_lft 331sec preferred_lft 331sec
    inet6 fe80::53e9:6c3a:fe7f:6c4d/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

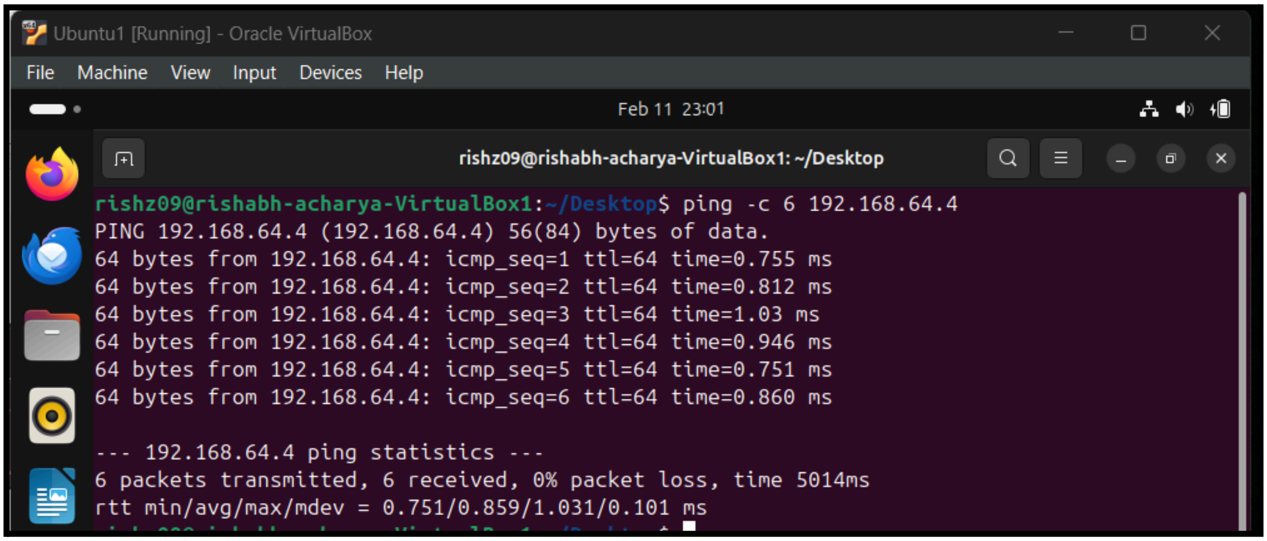
IP Address of VM1: 192.168.64.3



```
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP grou
p default qlen 1000
    link/ether 08:00:27:26:bd:50 brd ff:ff:ff:ff:ff:ff
    inet 192.168.64.4/24 brd 192.168.64.255 scope global dynamic noprefixroute enp0s
8
       valid_lft 593sec preferred_lft 593sec
    inet6 fe80::2117:b08b:9af:9007/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

IP Address of VM2: 192.168.64.3

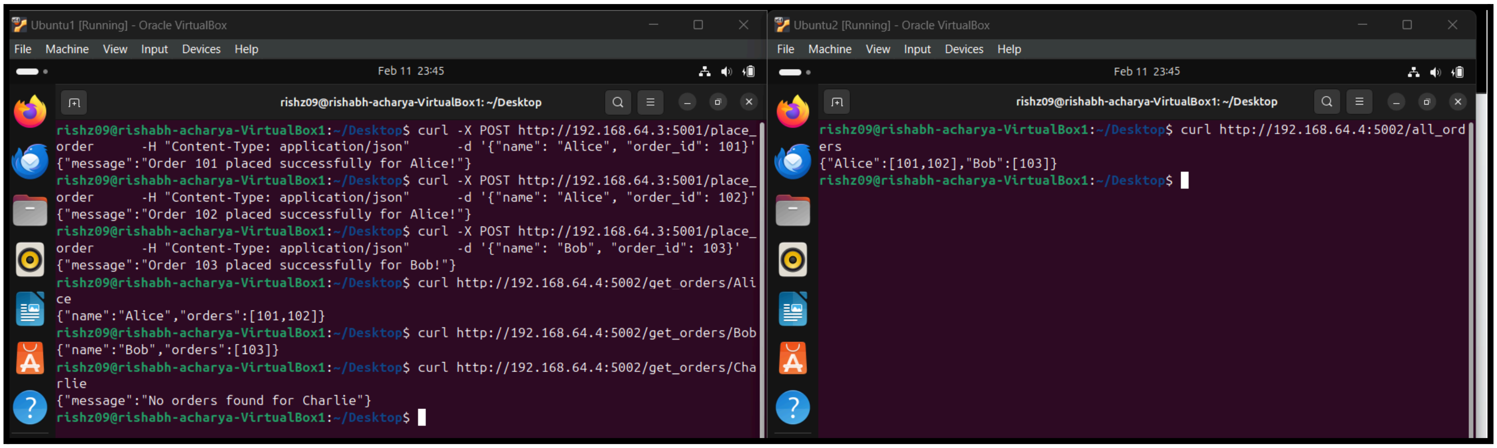
- We can verify that communication occurs by pinging IP of VM2 from VM1.


Ubuntu1 [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Feb 11 23:01
rishz09@rishabh-acharya-VirtualBox1: ~/Desktop

```
rishz09@rishabh-acharya-VirtualBox1:~/Desktop$ ping -c 6 192.168.64.4
PING 192.168.64.4 (192.168.64.4) 56(84) bytes of data.
64 bytes from 192.168.64.4: icmp_seq=1 ttl=64 time=0.755 ms
64 bytes from 192.168.64.4: icmp_seq=2 ttl=64 time=0.812 ms
64 bytes from 192.168.64.4: icmp_seq=3 ttl=64 time=1.03 ms
64 bytes from 192.168.64.4: icmp_seq=4 ttl=64 time=0.946 ms
64 bytes from 192.168.64.4: icmp_seq=5 ttl=64 time=0.751 ms
64 bytes from 192.168.64.4: icmp_seq=6 ttl=64 time=0.860 ms

--- 192.168.64.4 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5014ms
rtt min/avg/max/mdev = 0.751/0.859/1.031/0.101 ms
```

Screenshot of the working



Ubuntu1 [Running] - Oracle VirtualBox
File
Machine
View
Input
Devices
Help
Feb 11 23:45
rishz09@rishabh-acharya-VirtualBox1: ~/Desktop

```
rishz09@rishabh-acharya-VirtualBox1:~/Desktop$ curl -X POST http://192.168.64.3:5001/place_
order      -H "Content-Type: application/json"      -d '{"name": "Alice", "order_id": 101}'
{"message":"Order 101 placed successfully for Alice!"}
rishz09@rishabh-acharya-VirtualBox1:~/Desktop$ curl -X POST http://192.168.64.3:5001/place_
order      -H "Content-Type: application/json"      -d '{"name": "Alice", "order_id": 102}'
{"message":"Order 102 placed successfully for Alice!"}
rishz09@rishabh-acharya-VirtualBox1:~/Desktop$ curl -X POST http://192.168.64.3:5001/place_
order      -H "Content-Type: application/json"      -d '{"name": "Bob", "order_id": 103}'
{"message":"Order 103 placed successfully for Bob!"}
rishz09@rishabh-acharya-VirtualBox1:~/Desktop$ curl http://192.168.64.4:5002/get_orders/Ali
ce
{"name":"Alice","orders":[101,102]}
rishz09@rishabh-acharya-VirtualBox1:~/Desktop$ curl http://192.168.64.4:5002/get_orders/Bob
{"name":"Bob","orders":[103]}
rishz09@rishabh-acharya-VirtualBox1:~/Desktop$ curl http://192.168.64.4:5002/get_orders/Cha
rlie
{"message":"No orders found for Charlie"}
rishz09@rishabh-acharya-VirtualBox1:~/Desktop$
```

Ubuntu2 [Running] - Oracle VirtualBox
File
Machine
View
Input
Devices
Help
Feb 11 23:45
rishz09@rishabh-acharya-VirtualBox1: ~/Desktop

```
rishz09@rishabh-acharya-VirtualBox1:~/Desktop$ curl http://192.168.64.4:5002/all_ord
ers
{"Alice":[101,102],"Bob":[103]}
rishz09@rishabh-acharya-VirtualBox1:~/Desktop$
```

- Left side corresponds to VM1, and right side corresponds to VM2.

Deployment of a simple microservice application across the VMs

This has been done using Python and Flask.

- GitHub Link (codes)
- Video Demonstration

Note: Kindly read about the code below, before watching the video

user_service.py (VM1)

This service acts as the interface for users to place and retrieve their orders.

Endpoints:

- **/place_order (POST)**
  - ➔ Accepts a JSON request containing a name and order_id.
  - ➔ Forwards this request to **order_service.py** on VM2 (http://192.168.64.4:5002/receive_order).
  - ➔ Returns the response received from order_service.py.

- **/get_orders/<name> (GET)**
  - ➔ Requests all orders associated with a specific user (name) from order_service.py (http://192.168.64.4:5002/get_orders/{name}).
  - ➔ Returns the response received from order_service.py.

    - How it Communicates with VM2 (Order Service)
    - Uses HTTP POST to send order details to VM2.
    - Uses HTTP GET to fetch order details from VM2.

How it Communicates with VM2 (Order Service)

- Uses HTTP POST to send order details to VM2.
- Uses HTTP GET to fetch order details from VM2.

order_service.py (VM2)

This service acts as the order storage and processing system.

Endpoints:

- **/receive_order (POST)**
    - Receives order details from user_service.py.
    - Stores the order in **orders_db** (dictionary where names are keys and order IDs are values).
    - Returns a success message.
- **/get_orders/<name> (GET)**
    - Checks if the user (name) has any orders stored.
    - Returns the list of orders if found, otherwise returns a 404 error.

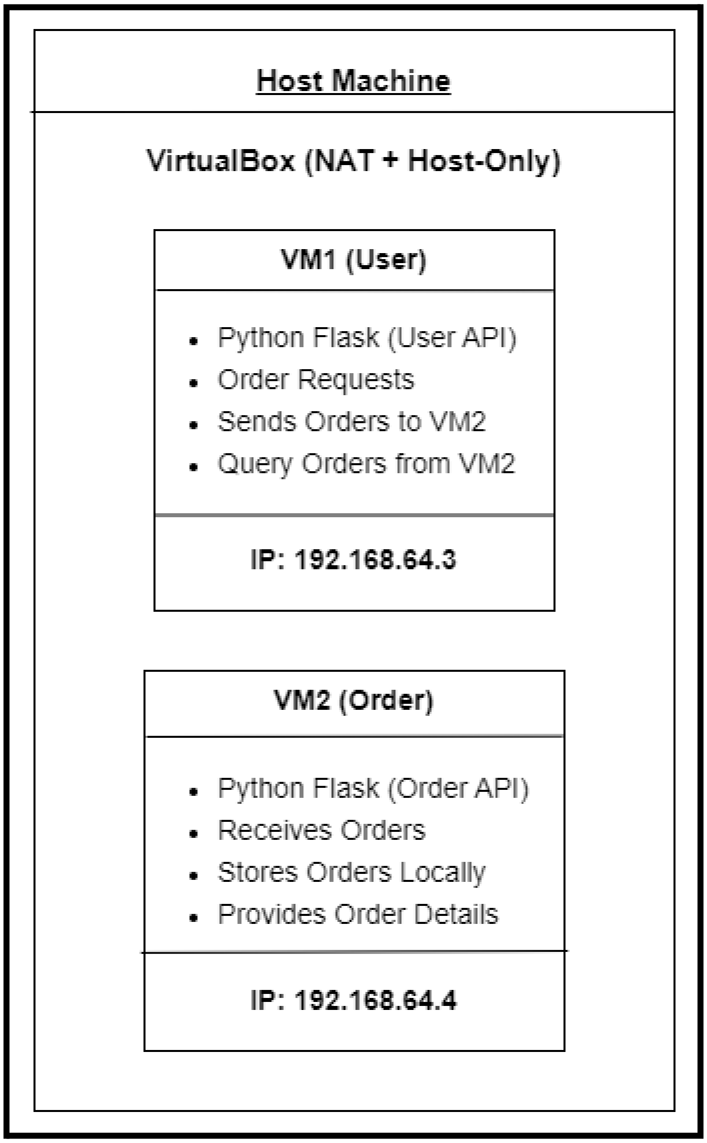
- **/all_orders (GET):** Returns all stored orders.

How They Communicate

- **user_service.py (VM1)** acts as a client to **order_service.py (VM2).**
- **order_service.py (VM2)** acts as a server to process and store the orders.
- Communication happens over HTTP requests using Flask and Requests Library.
- The interaction happens over a Host-Only Network with IPs:
    - **VM1: 192.168.64.3**
    - **VM2: 192.168.64.4**
- VM1 sends data to VM2 using **http://192.168.64.4:5002.**

- This setup allows users to place and retrieve orders using an **API-based microservices** approach.

Architecture Design


Host Machine
VirtualBox (NAT + Host-Only)
VM1 (User)
Python Flask (User API)
Order Requests
Sends Orders to VM2
Query Orders from VM2
IP: 192.168.64.3
VM2 (Order)
Python Flask (Order API)
Receives Orders
Stores Orders Locally
Provides Order Details
IP: 192.168.64.4