VCC | Assignment 1

- Student Name: Gourav Garg
- Roll Number: M25AI2164
- Demo: https://drive.google.com/file/d/1WS7Pw34ti8dIYlq625oRnGLjj7BgOL1l/view?usp=sharing
- Repository: https://github.com/ggarg55/multi-vm-microservice-demo
- Architecture: https://github.com/ggarg55/multi-vm-microservice-demo/blob/main/multi-vm-microservice-demo.drawio

Microservice Deployment Using VirtualBox and Multiple VMs

Step-by-Step Instructions for Implementation:

1. VirtualBox Installation on macOS
• Downloaded **Oracle VirtualBox** from the official website
• Installed VirtualBox on macOS
• Verified installation by launching VirtualBox successfully

---

2. Ubuntu Server OS Setup
• Downloaded **Ubuntu Server ISO** from the official Ubuntu website
• Selected Ubuntu Server for lightweight VM deployment

---

3. Creation of Three Virtual Machines
Created **three Ubuntu Server Virtual Machines**:
• **Client VM** — Sends API requests
• **API VM** — Hosts Node.js Express microservice
• **DB VM** — Hosts MySQL database server
Each VM was assigned:

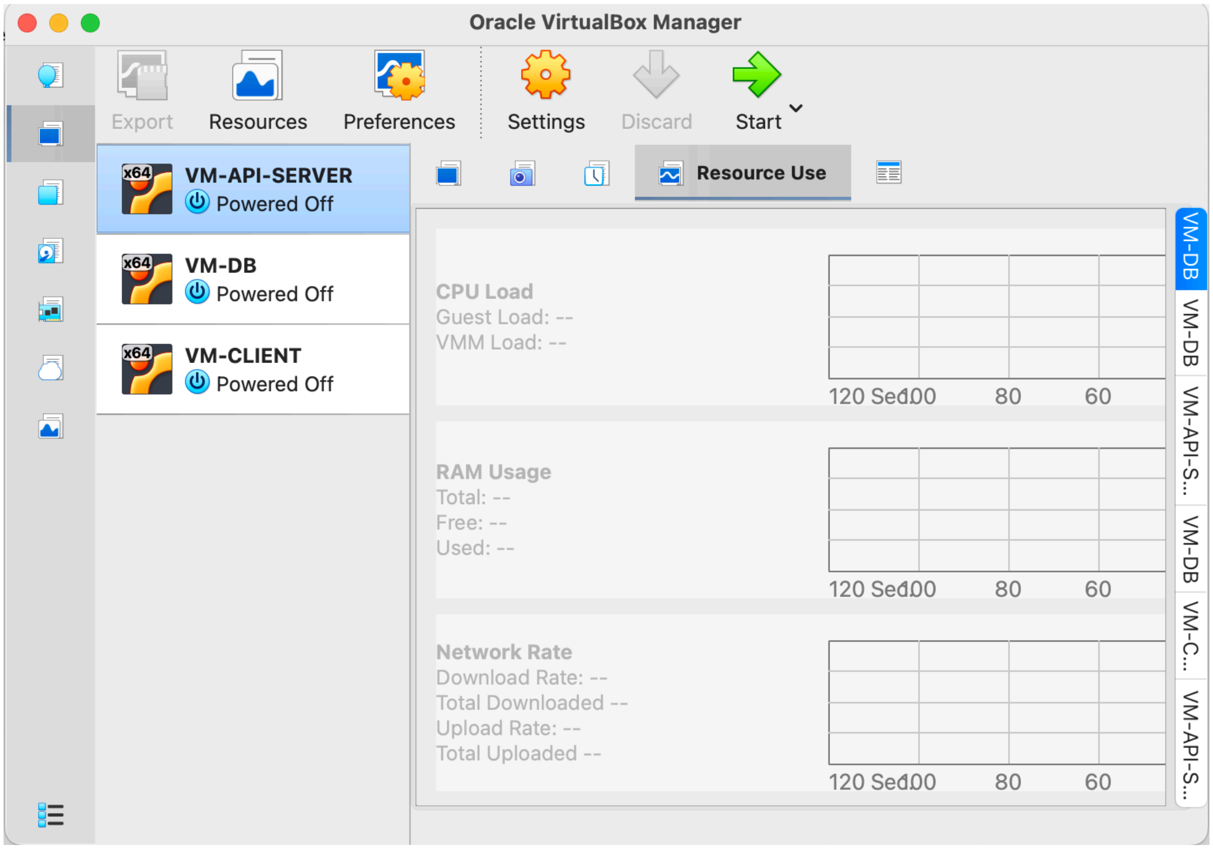
```
• 2 CPU cores ( VMs were crashing on single core and was sh
owing black screen)
• 2 GB RAM ( 1 GB ram was not working after installing expr
ss and mysql)
• 20 GB virtual disk
```

---

4. Virtual Machine Configuration

Configured system settings for each VM:

- OS: Ubuntu Server
- Set hostname based on VM role (client, api, db)
- Installed system updates using `apt update`



VM-API-SERVER
Powered Off
VM-DB
Powered Off
VM-CLIENT
Powered Off

CPU Load
Guest Load: --
VMM Load: --

RAM Usage
Total: --
Free: --
Used: --

Network Rate
Download Rate: --
Total Downloaded --
Upload Rate: --
Total Uploaded --

5. Network Adapter Setup and Configuration

Configured **two network adapters per VM**:

Adapter 1 — NAT

- Provides internet access for software installation

Adapter 2 — Internal Network

- Enables private communication between VMs
- Assigned **static IP addresses** using Netplan

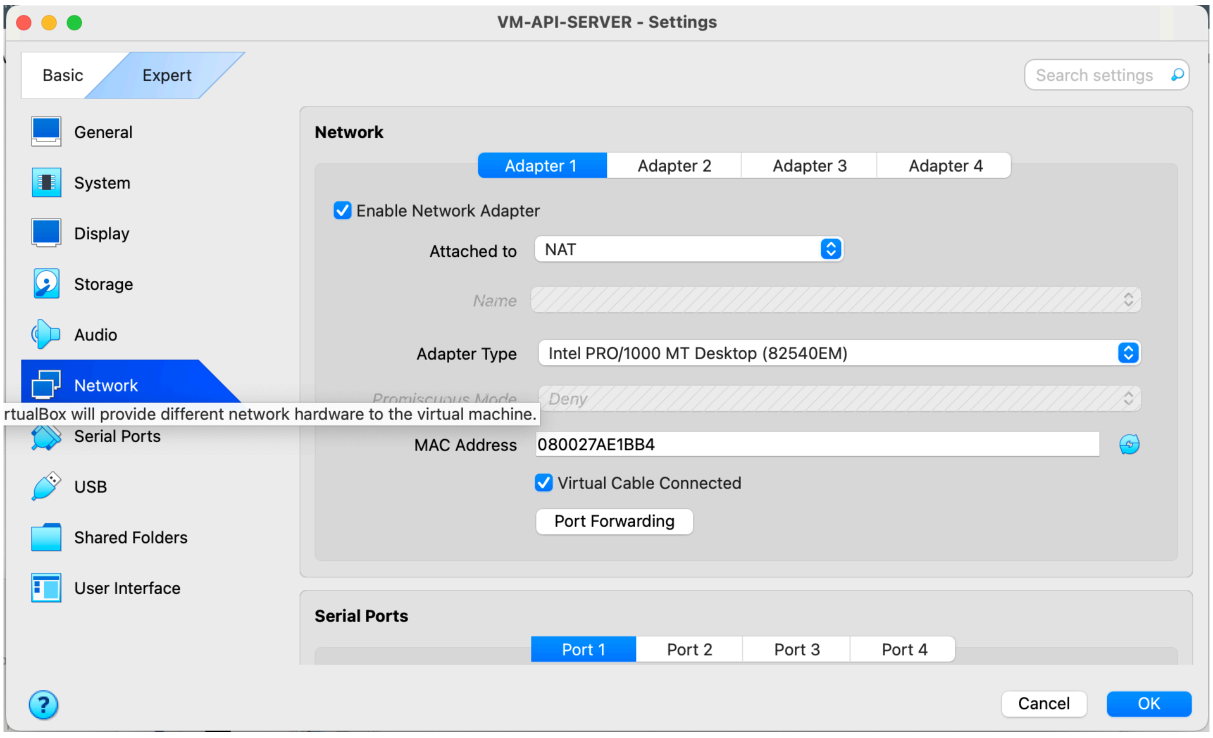
IP Assignments:

- Client VM → `192.168.100.12`
- API VM → `192.168.100.10`

- DB VM → `192.168.100.11`

Subnet used: `192.168.100.0/24`

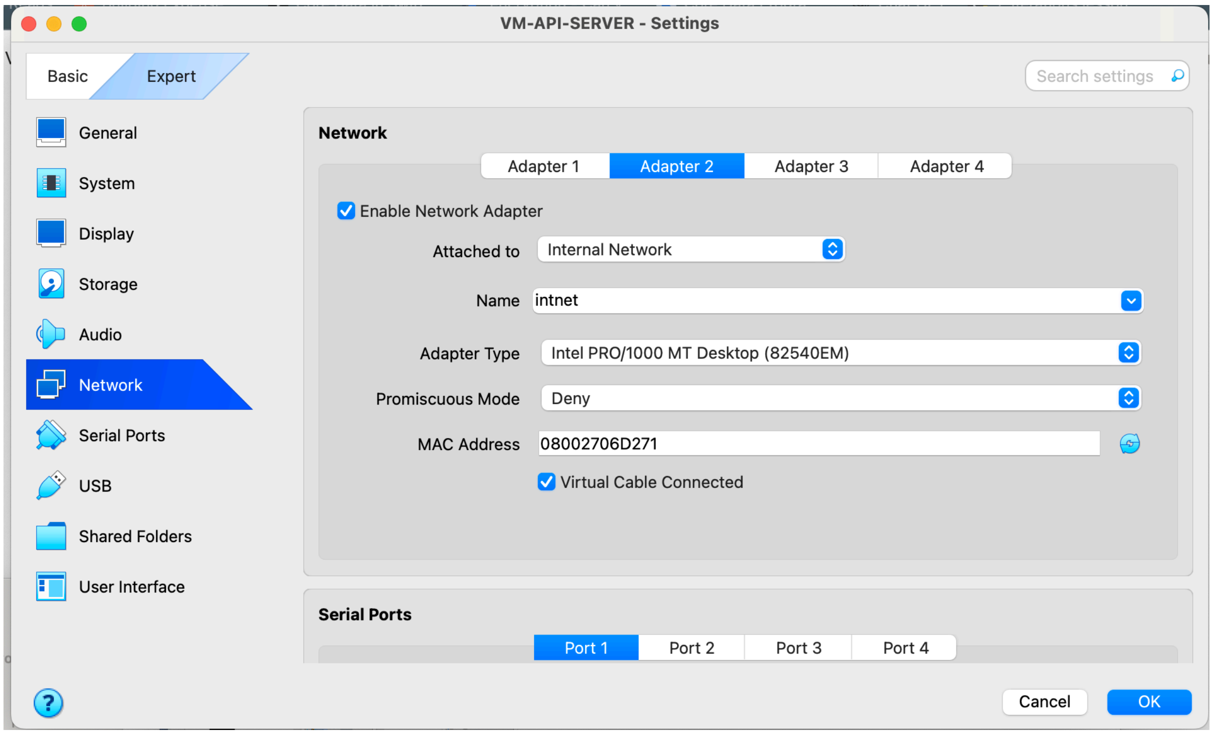
```
## network configurations:

# DB Server
network:
  version: 2
  ethernets:
    enp0s3:
      dhcp4: true

    enp0s8:
      addresses: [192.168.100.10/24]

# API Server
network:
  version: 2
  ethernets:
    enp0s3:
      dhcp4: true

    enp0s8:
      addresses: [192.168.100.11/24]


# Client Gateway
network:
  version: 2
  ethernets:
    enp0s3:
      dhcp4: true

    enp0s8:
      addresses: [192.168.100.12/24]
```


VM-API-SERVER - Settings

Network — Adapter 1

- [✓] Enable Network Adapter
- Attached to: NAT
- Name:
- Adapter Type: Intel PRO/1000 MT Desktop (82540EM)
- Promiscuous Mode: Deny
- MAC Address: 080027AE1BB4
- [✓] Virtual Cable Connected
- Port Forwarding

rtualBox will provide different network hardware to the virtual machine.


VM-API-SERVER - Settings

Network — Adapter 2

- [✓] Enable Network Adapter
- Attached to: Internal Network
- Name: intnet
- Adapter Type: Intel PRO/1000 MT Desktop (82540EM)
- Promiscuous Mode: Deny
- MAC Address: 08002706D271
- [✓] Virtual Cable Connected

---

6. MySQL Server Setup on DB VM

- Installed MySQL server on DB VM
- Created database: `microservice_db`
- Created MySQL user: `apiuser`

- Granted database access permissions
- Configured MySQL to allow **remote API access**
- Inserted sample user records into database

7. Express API Server Setup on API VM

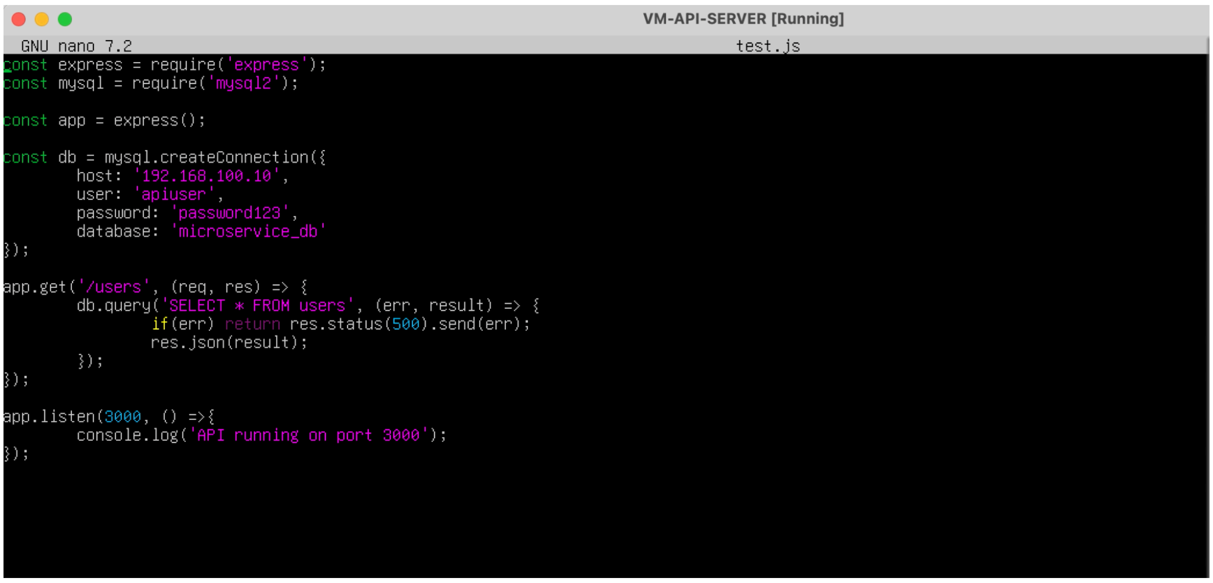
- Installed Node.js and npm
- Installed Express and MySQL driver (`mysql2`)
- Developed REST API endpoint: `GET /users`
- API fetches user data from MySQL DB VM
- Configured API to connect using DB VM IP address
- Started server on port **3000**

8. Client VM Gateway Request Flow

- Client VM sends HTTP request to API VM
- API VM queries MySQL data from DB VM
- DB VM returns records to API VM
- API VM responds to Client VM with JSON output

Data Flow:
`Client VM → API VM → DB VM → API VM → Client VM`

Microservice application Deployment:


VM-API-SERVER [Running]

```
  GNU nano 7.2                                                                       test.js
const express = require('express');
const mysql = require('mysql2');

const app = express();

const db = mysql.createConnection({
        host: '192.168.100.10',
        user: 'apiuser',
        password: 'password123',
        database: 'microservice_db'
});

app.get('/users', (req, res) => {
        db.query('SELECT * FROM users', (err, result) => {
                if(err) return res.status(500).send(err);
                res.json(result);
        });
});

app.listen(3000, () =>{
        console.log('API running on port 3000');
});
```

To deploy microservice on API Server execute following command:

```
node test.js
```

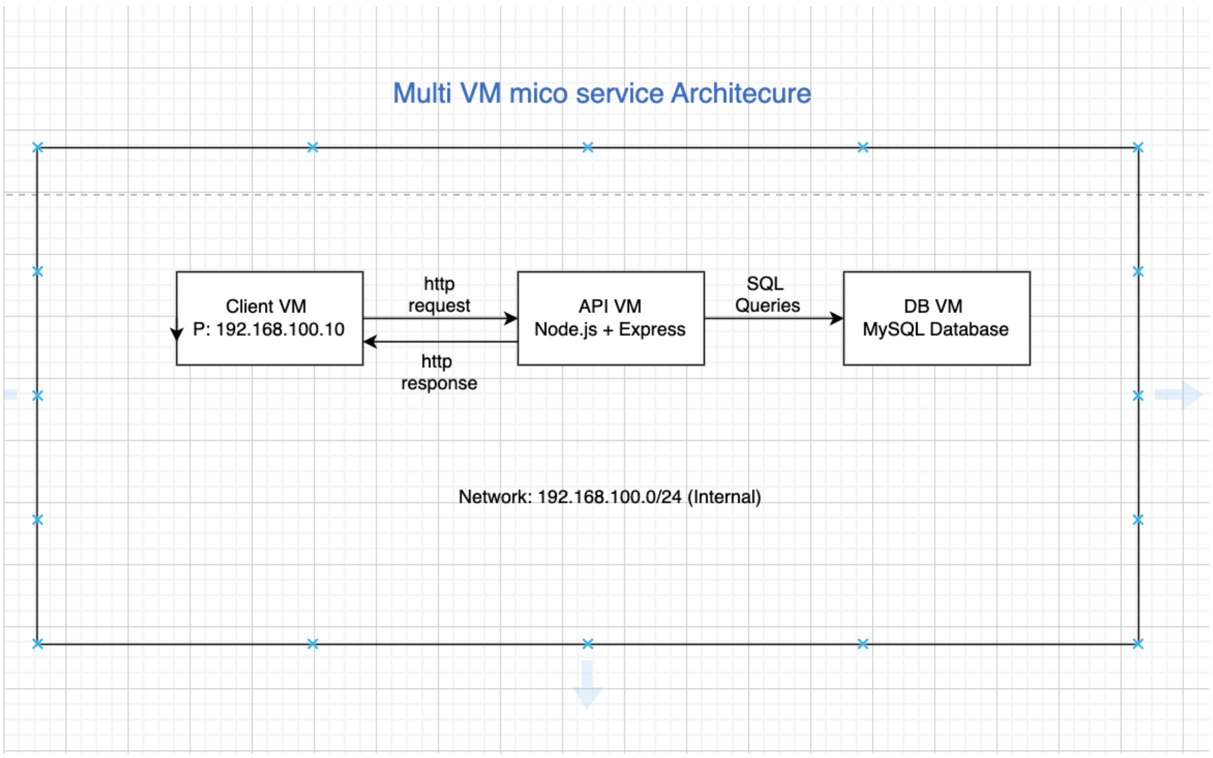
Client can make http request in following way:

```
curl http://192.168.100.11:3000/users
```

DB VM Server will return following response as we have only stored two users:

```
[{"id":1, "name": "Alice"}, {"id":2, "name": "Bob"}]
```

System Architecture:


Multi VM mico service Architecure
Client VM
P: 192.168.100.10
http
request
http
response
API VM
Node.js + Express
SQL
Queries
DB VM
MySQL Database
Network: 192.168.100.0/24 (Internal)