Report on use of VirtualBox to Create Multiple VMs, Connect These VMs, and Host One Microservice-Based Application

Objective:
Create and configure multiple Virtual Machines (VMs) using VirtualBox, establish a network between them, and deploy a microservice-based application across the connected VMs.

Step-by-Step Instructions for Implementation:

Installation of VirtualBox and Creation of VMs

Install VirtualBox:

- Downloaded and installed VirtualBox from the official website. Since the host machine runs Windows, choose the appropriate release.
- Simply double-clicked the setup file and followed the installation wizard to complete the process.

Create Virtual Machines:

- Open VirtualBox and click **New** to create a new VM.
- Configured the following settings:
  - **Name**: VM1
  - **OS Type**: Linux →Ubuntu (64-bit)
  - **Memory Size**: Allocated 2GB
  - **Processors**: 2 CPUs
  - **Virtual Disk**: Create a VDI file of size 25GB
  - **Network**: Bridged Adapter
  - **Shared Folder**: VCC1
- After the above configuration , started machine and installed the ubuntu by following setup wizard.
- Shutdown the VM1 machine.
- Right click the VM1 and click clone.
  - Configured the settings as
  - **Name**: VM2
  - **Full Clone**: Yes

- **Mac Address Policy**: Generate new Mac addresses for all network adapters.

- After the above configuration , started the machine and installed the lubuntu by following the setup wizard.
- **Host Machine with Windows 11 64 bit operating system, RAM 16 GB.**

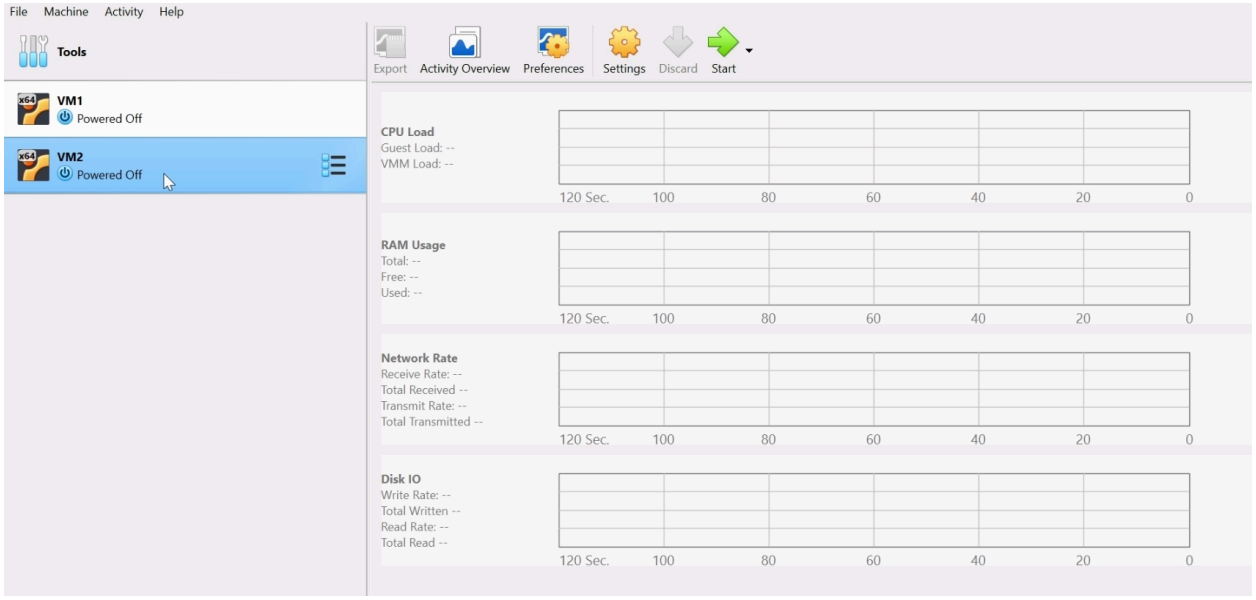

Fig: The virtual box with two machines created and configured

Configuration of Network Settings to Connect VMs

Start **VM1** and manually assigned the IP details as-

IP: 192.168.29.51

Netmask: 255.255.255.0

Gateway: 192.168.29.1

Start **VM2** Machine and manually assigned the IP details as-

IP: 192.168.29.52

Netmask: 255.255.255.0

Gateway: 192.168.29.1

Host Machine (Windows)

IP: 192.168.29.19

Netmask: 255.255.255.0

Gateway: 192.168.29.1

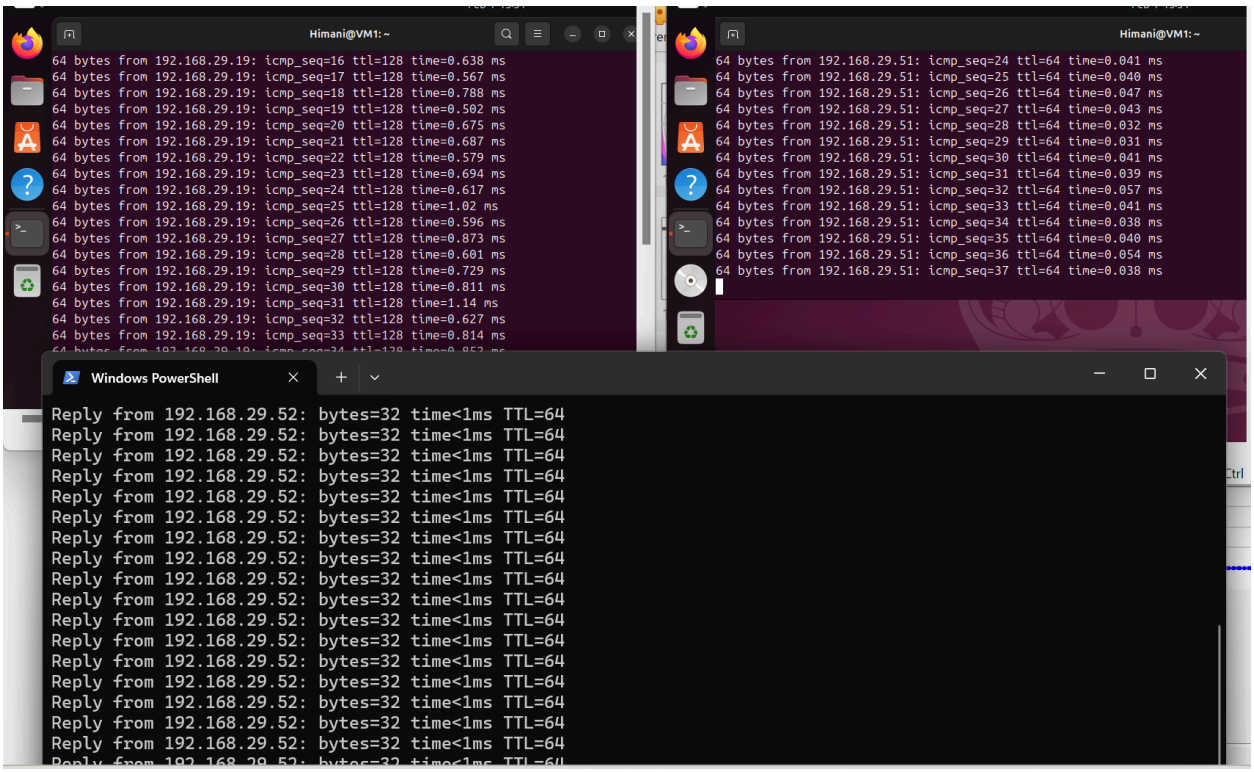

Fig: VM1 is pinging Router(192.168.29.19),VM2(192.168.29.51) and Host (192.168.29.52)

Deployment of Microservice Application

Each VM has a specific role in hosting the microservice application. Follow the configurations detailed in the scripts below.

VM1 (Data Provider)

- Hosts a simple Flask API (data_provider.py) on **port 6000**.

- Provides data when requested.

Host Machine (Middleware API)

- Acts as a **bridge** between VM1 (backend) and VM2 (frontend).
- Runs a Flask API (middleware.py) on **port 5000**.
- Fetches data from **VM1** and serves it to **VM2**.

VM2 (Frontend Web Page)

- Hosts an HTML web page (index.html).
- Sends requests to **Host Machine API** to fetch data.
- Displays responses in the UI.

Architecture Design: VM Communication for Microservice Application

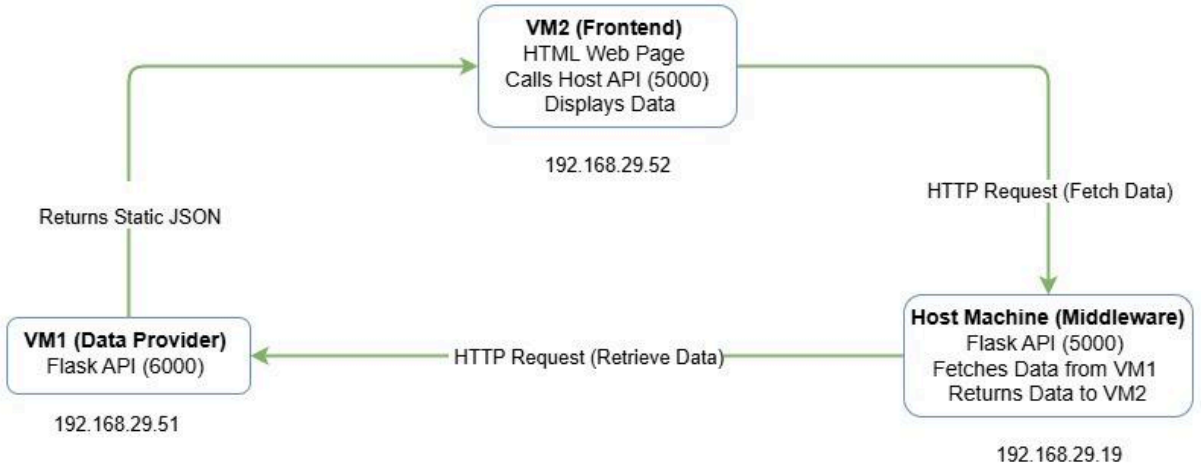

Fig: Architecture Design

Execution Workflow

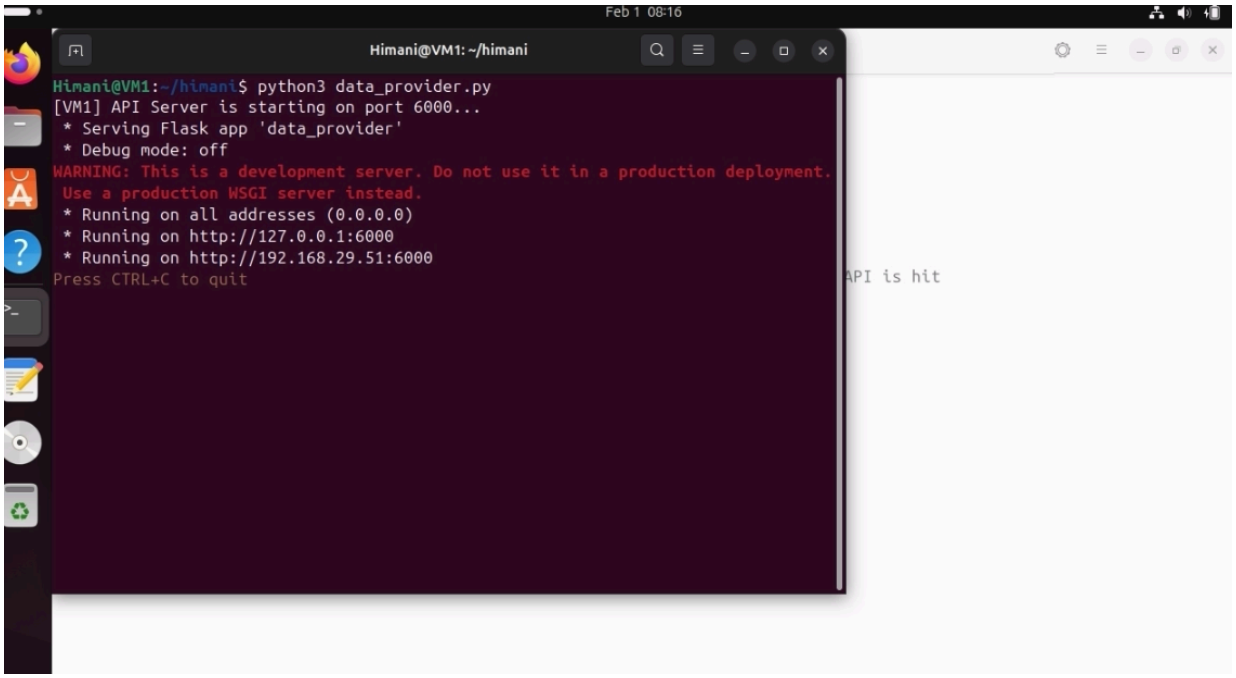

Fig: VM1 data_provider.py on **port 6000**

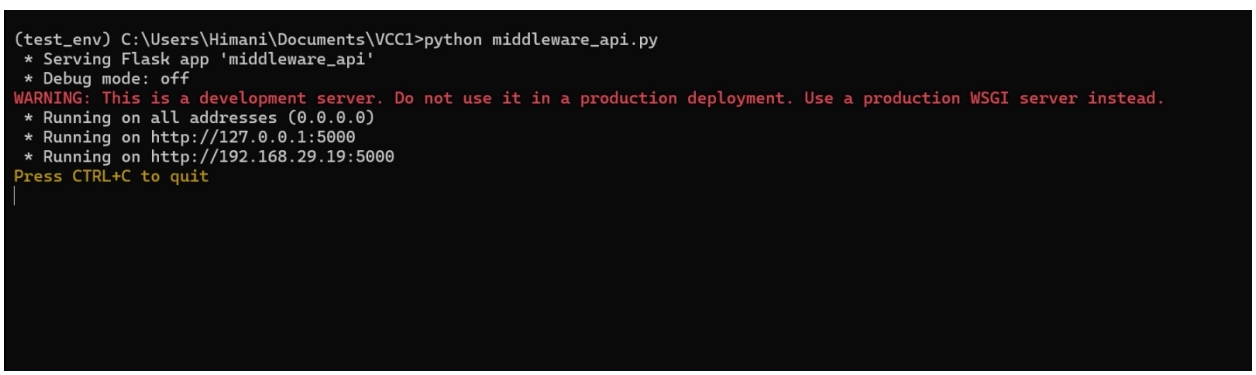

Fig: **Host Machine (Middleware API)** on **port 5000**

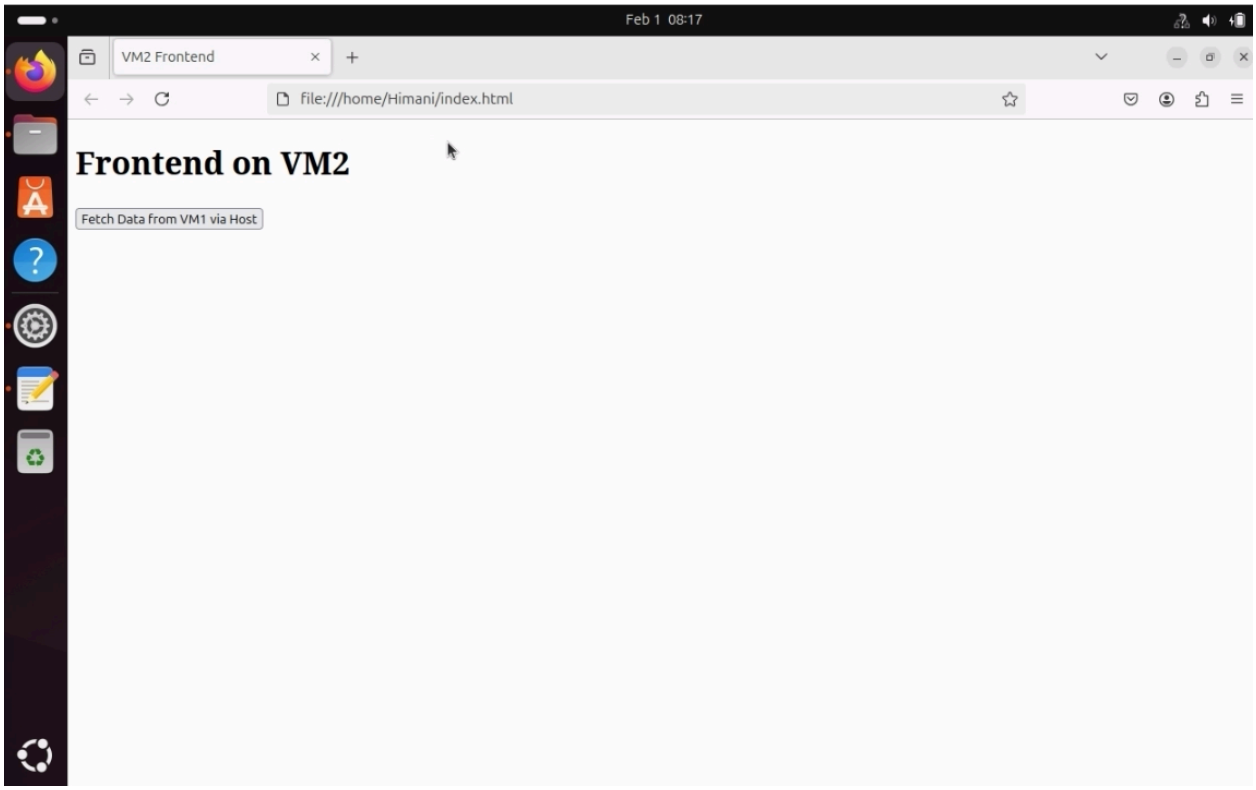

Fig: VM2 (Frontend Web Page:(index.html)

Execution Results:

Fig: Getting msg from the VM1 via host machine while clicks on the button

Observations & Insights

1. **Networking Setup:** Host-Only Adapter allows VM communication, while Bridged Adapter provides internet access. Firewall rules must allow API traffic.
2. **Shared Folders:** VirtualBox needs Guest Additions, VMware requires VMware Tools, and Samba is useful for cross-OS file sharing.
3. **API Communication:** Flask APIs efficiently connect VMs, with the Host Machine acting as a middleware.
4. **CORS Issue in Cross-VM Requests:** Browsers block API calls from **VM2 (Frontend) → Host Machine API** due to security policies.

References:

1. **VirtualBox Networking Modes** – Oracle Documentation
   https://www.virtualbox.org/manual/ch06.html
2. **VMware Networking Guide** – VMware Docs
   https://docs.vmware.com/en/VMware-Workstation-Pro/index.html
3. **Ubuntu Network Configuration** (for static IP and host-only networks)
   https://ubuntu.com/server/docs/network-configuration
4. **Flask Official Documentation** – Building REST APIs
   https://flask.palletsprojects.com/en/latest/
5. **Flask & Requests Library – API Communication**
   https://requests.readthedocs.io/en/latest/
6. **Flask Deployment Guide (Production Ready)** – Flask Docs
   https://flask.palletsprojects.com/en/2.0.x/tutorial/deploy/
7. **Flask-CORS GitHub & Docs – How to Enable CORS for Cross-Origin Requests**
   https://flask-cors.readthedocs.io/en/latest/
8. Understanding CORS (Cross-Origin Resource Sharing) – MDN Web Docs
   https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

GitHub Link: https://github.com/m23csa516/VCC_Assignment1.git

Recorded Video Link:

https://drive.google.com/file/d/18VBMmDdkE3OicpChSruFxsd56z8eaqC1/view?usp=sharing