Deploying a Distributed Microservices Architecture on Virtual Machines Using VirtualBox

---

1. Introduction

Objective:

This project involves setting up multiple Windows 10 Virtual Machines (VMs) using VirtualBox, establishing network communication between them, and deploying a microservice-based application. The application consists of a RESTful API built with Node.js and a MongoDB database, hosted on separate VMs. The setup replicates a distributed system where services interact over a virtual network, providing practical experience in VM management, networking, and microservice deployment.

System Requirements:

- **Host System:** Windows with sufficient RAM (8GB+ recommended) with a minimum hard disk space of 40 GB.
- **Virtual Machines:** Windows 10 (two instances) iso file(any OS can be used, e.g – Ubuntu-Linux).
- **Software Requirements:**
  - Oracle VirtualBox (for VM management)
  - VirtualBox Extension Pack (for enhanced networking)
  - Node.js (for building the API)
  - MongoDB (for database storage)
  - Git (optional for version control)
  - Postman (for API testing)
  - Windows PowerShell/Command Prompt (for command execution)

---

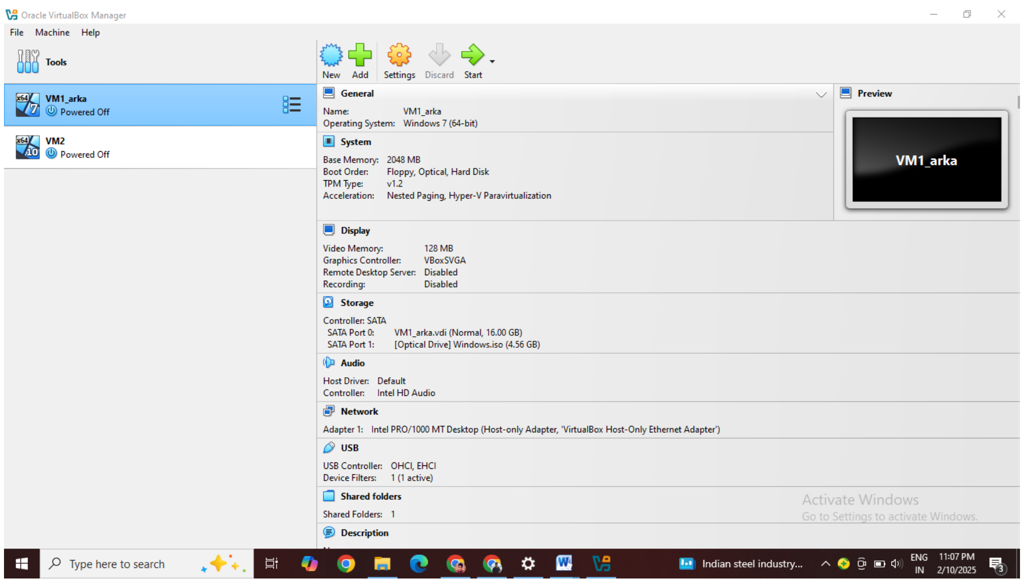
2. Virtual Machine Setup

Step 1: Install VirtualBox

1. Download **VirtualBox** from VirtualBox Official Website and install it.
2. Download and install the **VirtualBox Extension Pack** to enable additional networking features.
3. Restart your computer if prompted after installation.

Step 2: Create Virtual Machines

1. Open **VirtualBox** and click **New** to create a new VM.
2. Enter a name for the VM (e.g., VM1-API for the API server, VM2-DB for the database server).
3. Select **Windows 10 (64-bit)** as the operating system.
4. Allocate at least **2GB RAM** per VM (4GB recommended for better performance).
5. Create a **Virtual Hard Disk** (at least 10GB, dynamically allocated).
6. Install Windows 10 on the VM using an ISO file.
7. Repeat the process for the second VM.



VM1_arka
Powered Off
VM2
Powered Off

General
Name: VM1_arka
Operating System: Windows 7 (64-bit)

System
Base Memory: 2048 MB
Boot Order: Floppy, Optical, Hard Disk
TPM Type: v1.2
Acceleration: Nested Paging, Hyper-V Paravirtualization

Display
Video Memory: 128 MB
Graphics Controller: VBoxSVGA
Remote Desktop Server: Disabled
Recording: Disabled

Storage
Controller: SATA
SATA Port 0: VM1_arka.vdi (Normal, 16.00 GB)
SATA Port 1: [Optical Drive] Windows.iso (4.56 GB)

Audio
Host Driver: Default
Controller: Intel HD Audio

Network
Adapter 1: Intel PRO/1000 MT Desktop (Host-only Adapter, 'VirtualBox Host-Only Ethernet Adapter')

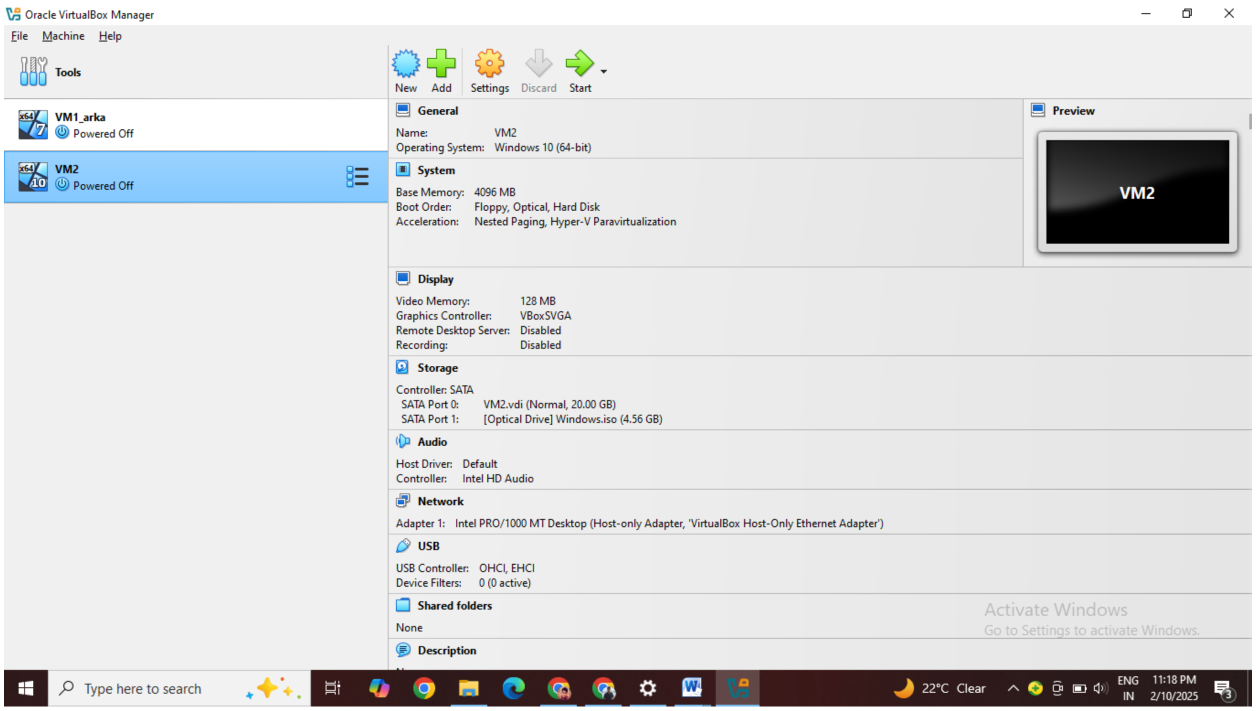
USB
USB Controller: OHCI, EHCI
Device Filters: 1 (1 active)

Shared folders
Shared Folders: 1

Description

Activate Windows
Go to Settings to activate Windows.

Figure 1: VM1 creation



General
Name: VM2
Operating System: Windows 10 (64-bit)

System
Base Memory: 4096 MB
Boot Order: Floppy, Optical, Hard Disk
Acceleration: Nested Paging, Hyper-V Paravirtualization

Display
Video Memory: 128 MB
Graphics Controller: VBoxSVGA
Remote Desktop Server: Disabled
Recording: Disabled

Storage
Controller: SATA
SATA Port 0: VM2.vdi (Normal, 20.00 GB)
SATA Port 1: [Optical Drive] Windows.iso (4.56 GB)

Audio
Host Driver: Default
Controller: Intel HD Audio

Network
Adapter 1: Intel PRO/1000 MT Desktop (Host-only Adapter, 'VirtualBox Host-Only Ethernet Adapter')

USB
USB Controller: OHCI, EHCI
Device Filters: 0 (0 active)

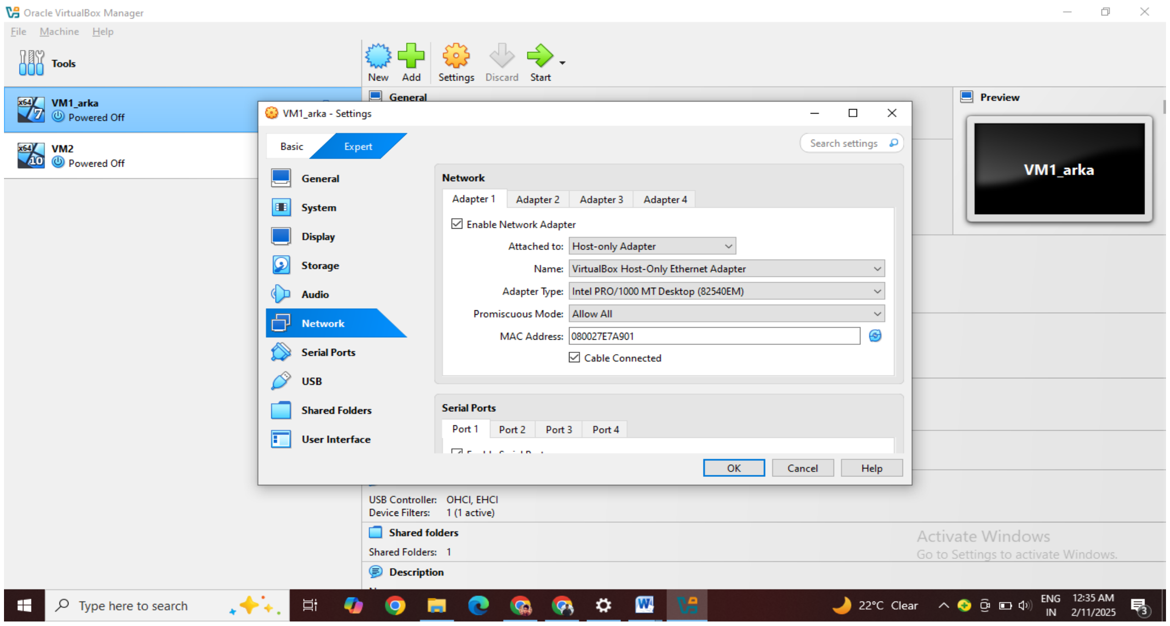
Shared folders
None

Description

Figure 2: VM2 creation

Step 3: Configure Network for VM Communication

1. In **VirtualBox**, go to **Settings → Network** for each VM.
2. Set **Adapter 1** to **Host-Only Adapter** to enable communication between the VMs.



Network
Adapter 1 | Adapter 2 | Adapter 3 | Adapter 4
☑ 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 All
MAC Address: 080027E7A901
☑ Cable Connected

Serial Ports
Port 1 | Port 2 | Port 3 | Port 4

USB Controller: OHCI, EHCI
Device Filters: 1 (1 active)
Shared folders
Shared Folders: 1
Description

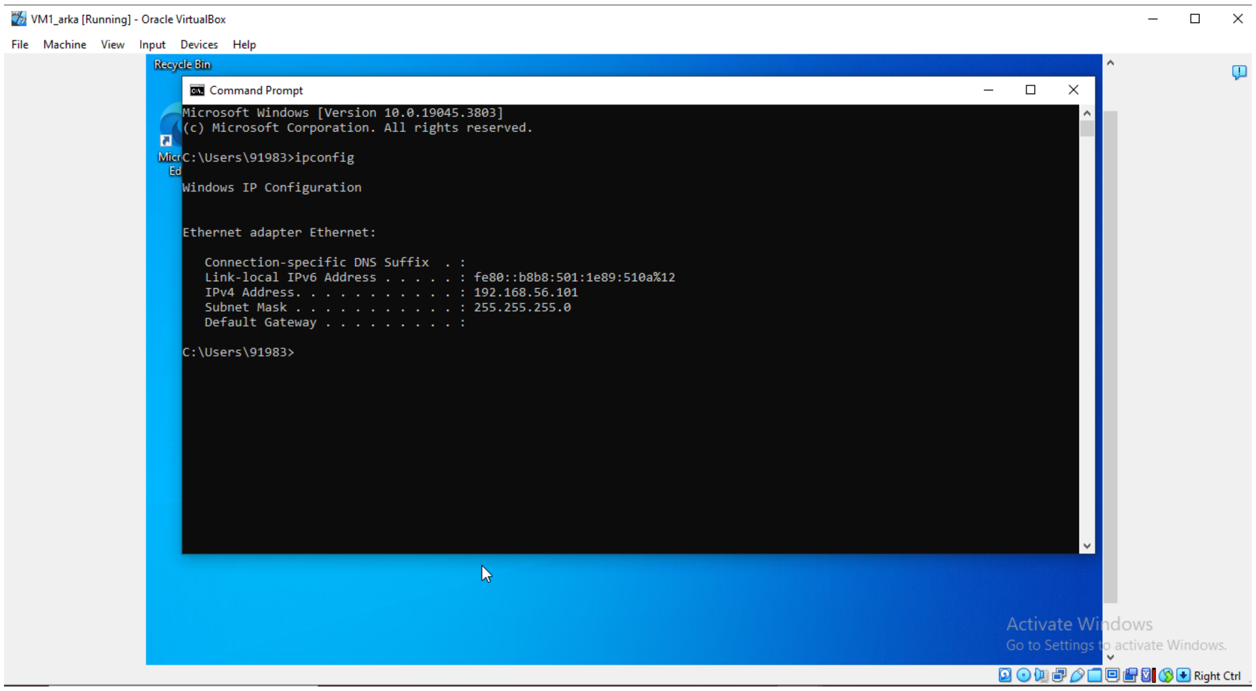
Figure 3: Network setting for each VM

3. Start both VMs and retrieve their IP addresses:

   - Open **Command Prompt (CMD)** in each VM and run:

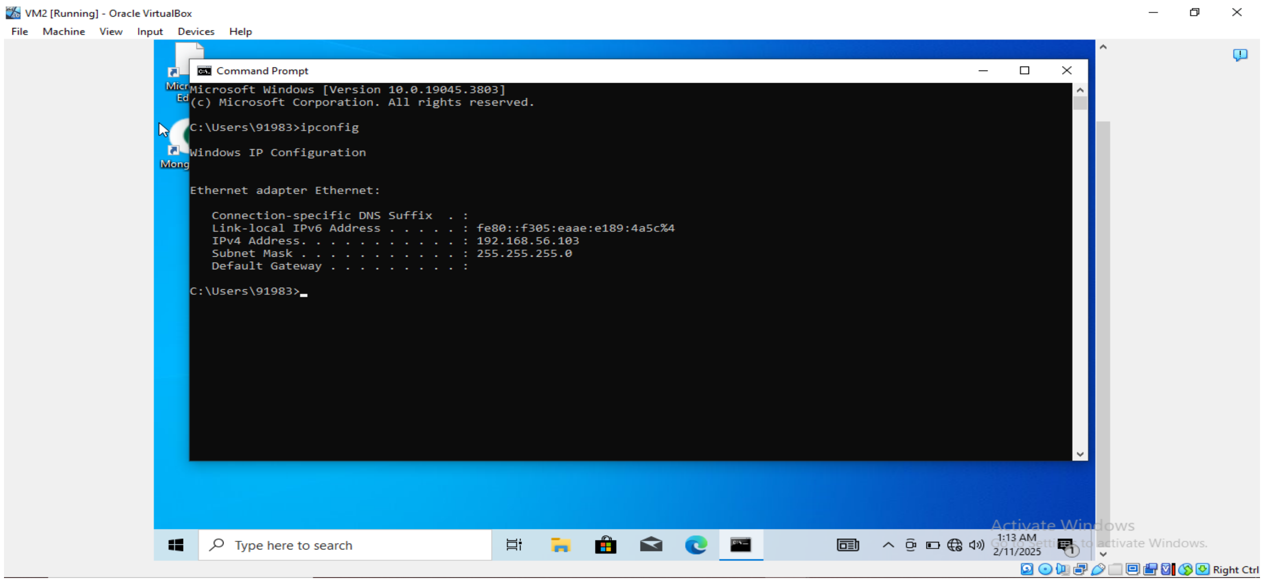
```
ipconfig
```

   - Note down the **IPv4 address** (e.g., 192.168.56.101 for VM1, 192.168.56.103 for VM2).


VM1_arka [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Recycle Bin
Command Prompt
Activate Windows
Go to Settings to activate Windows.
Right Ctrl

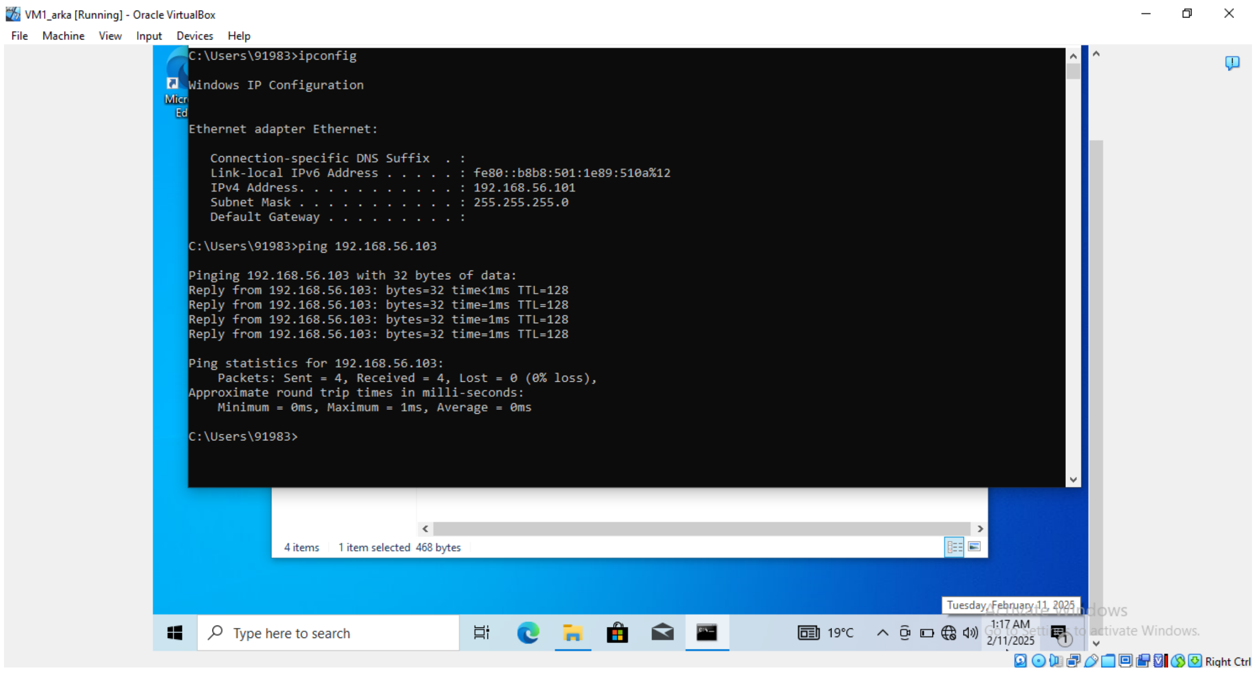
```
Microsoft Windows [Version 10.0.19045.3803]
(c) Microsoft Corporation. All rights reserved.

C:\Users\91983>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::b8b8:501:1e89:510a%12
   IPv4 Address. . . . . . . . . . . : 192.168.56.101
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

C:\Users\91983>
```

Figure 4: VM1's IPV4 address


VM2 [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Command Prompt
Type here to search
Activate Windows
Go to Settings to activate Windows.
1:13 AM
2/11/2025
Right Ctrl

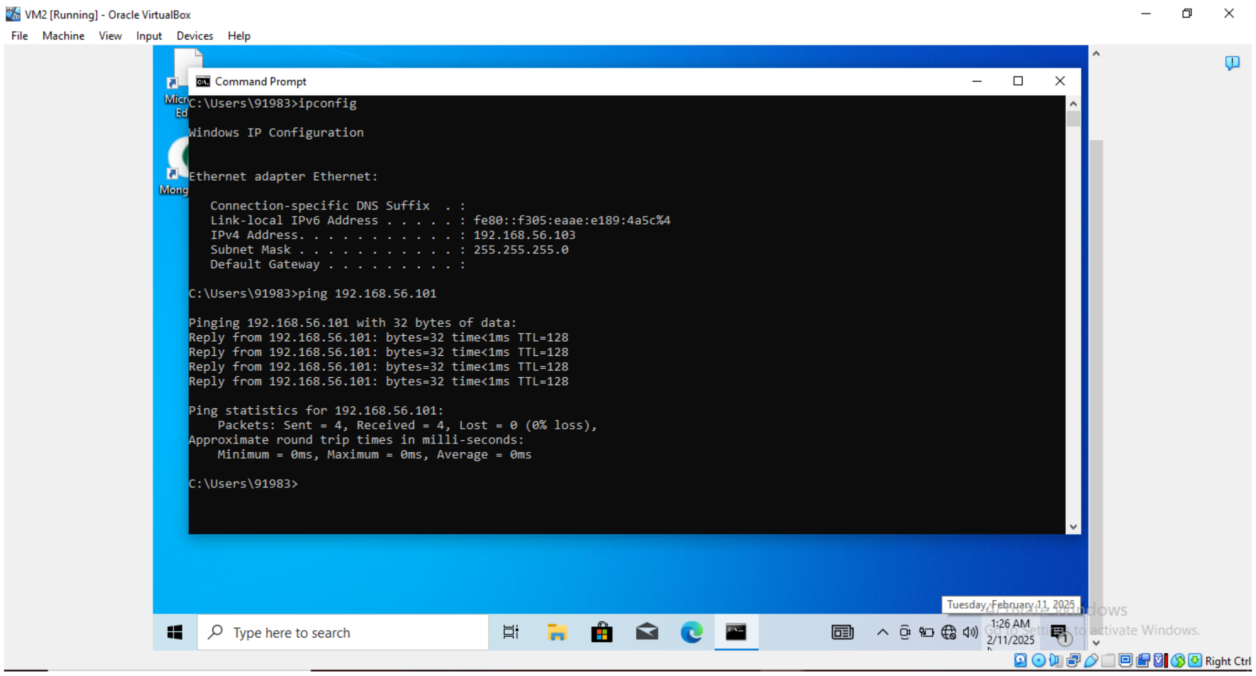
```
Microsoft Windows [Version 10.0.19045.3803]
(c) Microsoft Corporation. All rights reserved.

C:\Users\91983>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::f305:eaae:e189:4a5c%4
   IPv4 Address. . . . . . . . . . . : 192.168.56.103
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

C:\Users\91983>_
```

Figure 5: VM2's IPV4 address


VM1_arka [Running] - Oracle VirtualBox
File Machine View Input Devices Help
4 items 1 item selected 468 bytes
Type here to search
19°C
1:17 AM
2/11/2025
Tuesday, February 11, 2025
Right Ctrl

```
C:\Users\91983>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::b8b8:501:1e89:510a%12
   IPv4 Address. . . . . . . . . . . : 192.168.56.101
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

C:\Users\91983>ping 192.168.56.103

Pinging 192.168.56.103 with 32 bytes of data:
Reply from 192.168.56.103: bytes=32 time<1ms TTL=128
Reply from 192.168.56.103: bytes=32 time=1ms TTL=128
Reply from 192.168.56.103: bytes=32 time=1ms TTL=128
Reply from 192.168.56.103: bytes=32 time=1ms TTL=128

Ping statistics for 192.168.56.103:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms

C:\Users\91983>
```

Figure 6: Connectivity test for VM2 from VM1


VM2 [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Command Prompt
Type here to search
1:26 AM
2/11/2025
Tuesday, February 11, 2025
Right Ctrl

```
C:\Users\91983>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::f305:eaae:e189:4a5c%4
   IPv4 Address. . . . . . . . . . . : 192.168.56.103
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

C:\Users\91983>ping 192.168.56.101

Pinging 192.168.56.101 with 32 bytes of data:
Reply from 192.168.56.101: bytes=32 time<1ms TTL=128
Reply from 192.168.56.101: bytes=32 time<1ms TTL=128
Reply from 192.168.56.101: bytes=32 time<1ms TTL=128
Reply from 192.168.56.101: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.56.101:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Users\91983>
```

Figure 7: Connectivity test for VM1 from VM2

3. Setting Up the API Server (VM1 - Windows 10)

Step 4: Install Node.js

1. Download and install **Node.js** from Node.js Official Website.
2. Verify the installation by running:

```
node -v

npm -v
```

Step 5: Develop the Microservice

Open **Command Prompt (CMD)** and navigate to the project folder:

```
mkdir microservice

cd microservice
```

1. Initialize a new Node.js project and install dependencies:

```
npm init -y

npm install express mongoose cors
```

2. Create a new file index.js in the **microservice** folder:

```
const express = require('express');
const mongoose = require('mongoose');

const app = express();
app.use(express.json());

mongoose.connect('mongodb://192.168.56.103:27017/mydb', {
    useNewUrlParser: true,
    useUnifiedTopology: true
}).then(() => console.log('Connected to MongoDB'))
  .catch(err => console.log(err));

app.get('/', (req, res) => res.send('Microservice Running'));

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

Explanation

- **Express.js Setup:** Initializes an Express.js application and enables JSON support.
- **MongoDB Connection:** Uses Mongoose to connect to the MongoDB instance on VM2 (192.168.56.103).
- '***GET Route***': Returns a simple message to confirm the API is running.
- **Server Listening:** The application listens for requests on port 3000.

4. Start the API service:

```
node index.js
```

---

4. Setting Up the Database Server (VM2 - Windows 10)

Step 6: Install MongoDB

1. Download **MongoDB Community Edition** from MongoDB Official Website.
2. Install MongoDB and ensure it runs as a Windows service.
3. Start the MongoDB service:

```
net start MongoDB
```

4. Allow remote connections:
   - Open C:\Program Files\MongoDB\Server\6.0\bin\mongod.cfg in Notepad.
   - Locate bindIp: 127.0.0.1 and change it to bindIp: 0.0.0.0.
   - Restart MongoDB:

```
net stop MongoDB

net start MongoDB
```

---

5. Testing the Microservice

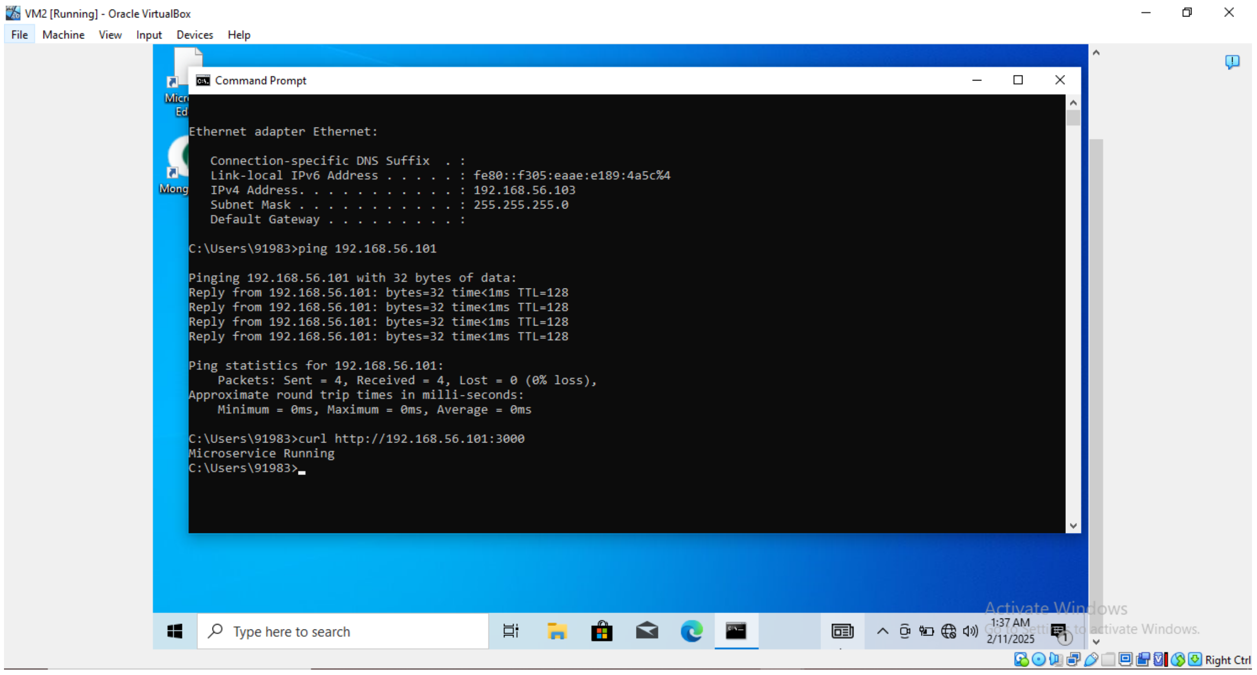
Step 7: Verify API Connection

1. From **VM2**, test the API by running:

   curl http://192.168.56.101:3000

2. Expected response:

   Microservice Running



```
Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::f305:eaae:e189:4a5c%4
   IPv4 Address. . . . . . . . . . . : 192.168.56.103
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

C:\Users\91983>ping 192.168.56.101

Pinging 192.168.56.101 with 32 bytes of data:
Reply from 192.168.56.101: bytes=32 time<1ms TTL=128
Reply from 192.168.56.101: bytes=32 time<1ms TTL=128
Reply from 192.168.56.101: bytes=32 time<1ms TTL=128
Reply from 192.168.56.101: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.56.101:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Users\91983>curl http://192.168.56.101:3000
Microservice Running
C:\Users\91983>
```

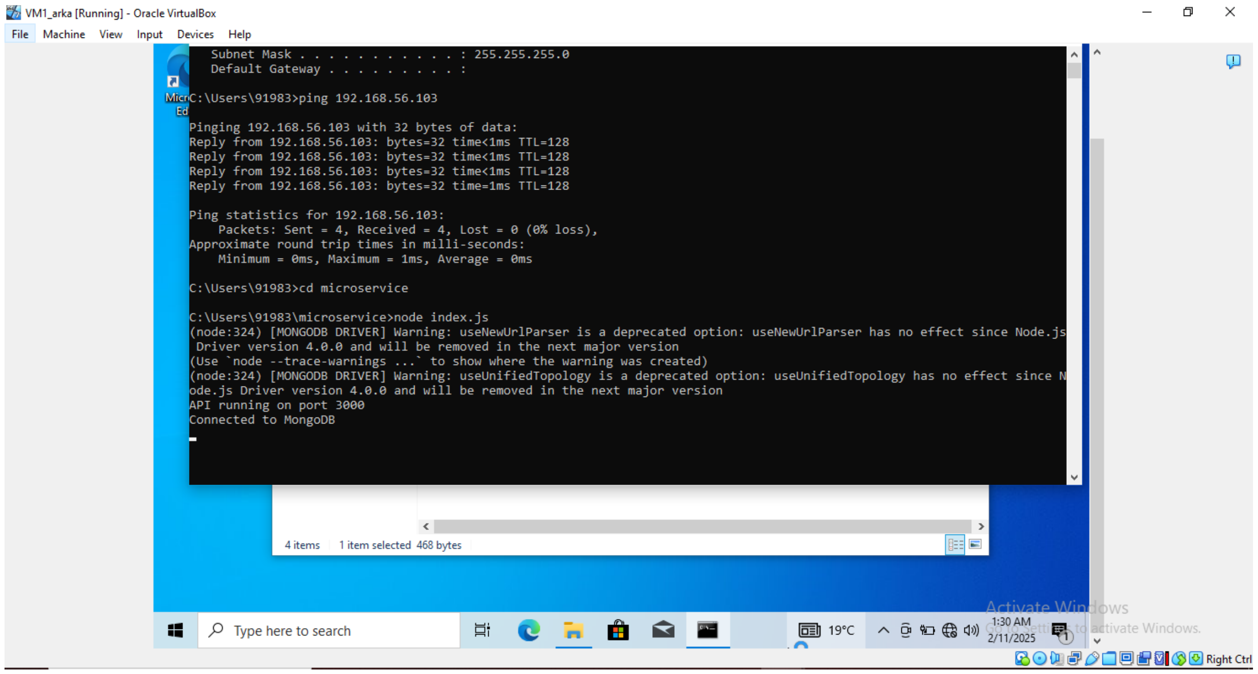
VM2 [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Command Prompt
Type here to search
1:37 AM
2/11/2025
Right Ctrl

Figure 8: Connectivity test for API server using Microservice from VM2

3. Verify MongoDB connection by checking the API logs:

   Connected to MongoDB

   API running on port 3000


VM1_arka [Running] - Oracle VirtualBox
File Machine View Input Devices Help
4 items 1 item selected 468 bytes
Type here to search
19°C
1:30 AM
2/11/2025
Activate Windows
Right Ctrl

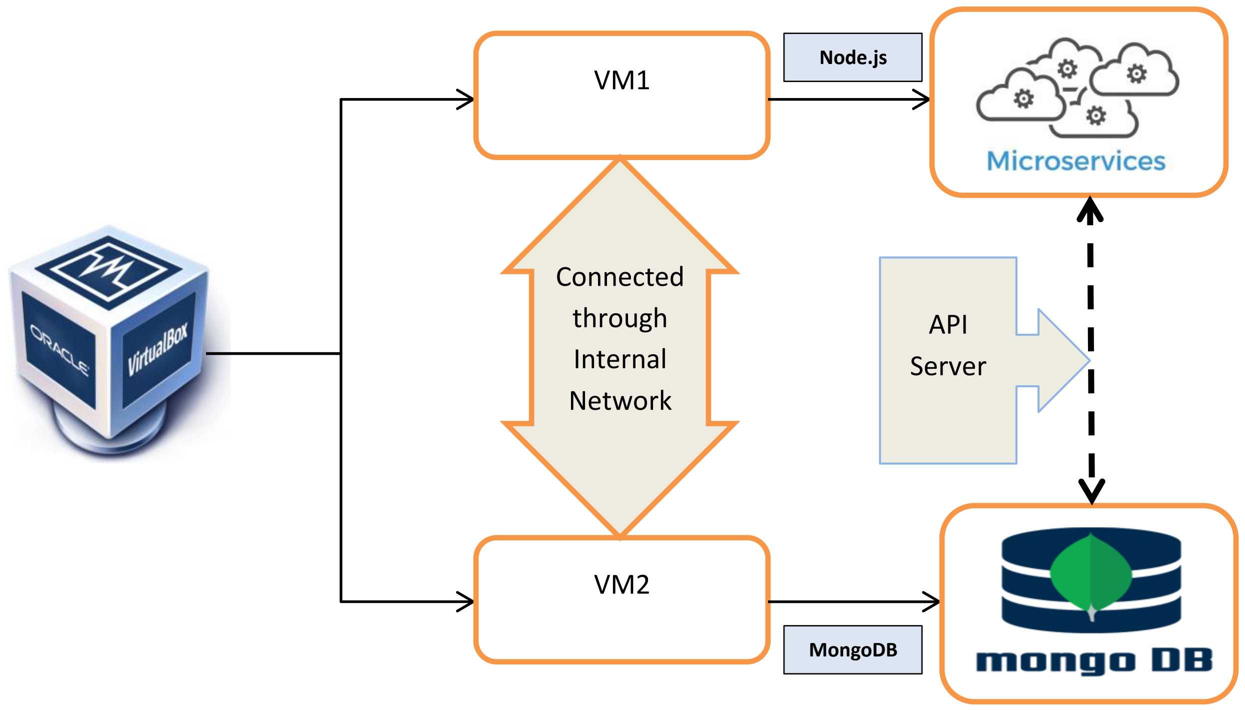
```
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

C:\Users\91983>ping 192.168.56.103

Pinging 192.168.56.103 with 32 bytes of data:
Reply from 192.168.56.103: bytes=32 time<1ms TTL=128
Reply from 192.168.56.103: bytes=32 time<1ms TTL=128
Reply from 192.168.56.103: bytes=32 time<1ms TTL=128
Reply from 192.168.56.103: bytes=32 time=1ms TTL=128

Ping statistics for 192.168.56.103:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms

C:\Users\91983>cd microservice

C:\Users\91983\microservice>node index.js
(node:324) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated option: useNewUrlParser has no effect since Node.js
 Driver version 4.0.0 and will be removed in the next major version
(Use `node --trace-warnings ...` to show where the warning was created)
(node:324) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useUnifiedTopology has no effect since N
ode.js Driver version 4.0.0 and will be removed in the next major version
API running on port 3000
Connected to MongoDB
```

Figure 9: Connectivity test for MongoDB from VM1

4. Alternatively, use **Postman** to send a GET request to:

http://192.168.56.101:3000

6. Architecture Diagram



7. Source Code Repository

Github Repository Link

https://github.com/ArkaGayen16/VccAssignment1.git

8. Recorded Video Demo

https://drive.google.com/file/d/1kMflxq0IiQmsPrilBsmB4rOVBWKSMsxd/view?usp=drive_link

9. Deliverables

Included:

1. **Project Report** (This Document)
2. **Architecture Diagram**
3. **Source Code Repository**
4. **Recorded Video Demo**

8. Conclusion

This project successfully demonstrates the end-to-end process of setting up Virtual Machines, configuring network communication, and deploying a microservice-based application using Node.js and MongoDB. The configured environment replicates a real-world microservices architecture, allowing seamless interaction between services over a network. By implementing this setup, users gain hands-on experience with virtualization, networking, backend service deployment, and database management, making it an excellent foundation for learning distributed system design and deployment strategies.