**PRACTICAL 2:**

**INSTALLATION AND CONFIGURE GOOGLE APP ENGINE. CREATE HELLO WORLD APP AND OTHER SIMPLE WEB APPLICATIONS USING PYTHON/JAVA.**

1. INSTALL GOOGLE CLOUD SDK (https://cloud.google.com/sdk/docs/install)

2. DOWNLOAD AND LOGIN WHEN ASKED THRU CLI

3. After installation, do '***gcloud init***'

4. Go to console.cloud.google.com and accept terms and conditions using your account

5. Then do '***gcloud projects create project\_name***'

6. Set the project as active using '***gcloud config set project\_name*** '

7. Install Google App Engine '***gcloud components install app-engine-python***'

8. Create a folder and make "***main.py***" and "***app.yaml***" files in it.

**CODE (main.py) :**

from flask import Flask

app = Flask(\_name\_)

@app.route('/')

def hello():

return 'Hello, World from App Engine!'

if \_name\_ == '\_main\_':

app.run(host='127.0.0.1', port=8080, debug=True)

**CODE (app.yaml):**

runtime: python39

entrypoint: ***gunicorn -b :$PORT main:app***

9. \****pip install flask***\*

10. Test locally using \****python main.py***\*

If error then do \****py main.py***\*

11. Enter the URL and Port number in browser (http://127.0.0.1:8080/)

12. Deploy to GCloud: \****gcloud app deploy***\*

**PRACTICAL 9:**

**VIRTUAL MACHINE ON AWS-LAUNCHING THE VIRTUAL MACHINE ON AMAZON AWS AND DOING BASIC.**

**1. Create AWS Account**

* Go to: <https://aws.amazon.com/>
* Click **"Create an AWS Account"**.
* Fill in your email, password, account name.
* Enter payment details (AWS gives **free tier**, but payment info is mandatory).
* Verify with OTP.
* Select **"Basic Support Plan"** (which is free).
* Finish account setup.

(⚡Note: If you already have an AWS account, skip this.)

**2. Sign In to AWS Management Console**

* Visit: <https://aws.amazon.com/console/>
* Log in using your email and password.
* Choose **"Root User"** if asked.

**3. Open EC2 Service**

* Once logged in, search for **"EC2"** in the search bar (EC2 = Elastic Compute Cloud, i.e., Virtual Machine).
* Click **"EC2"**.

**4. Launch a Virtual Machine (EC2 Instance)**

🔵 Step-by-Step:

1. Click **"Launch Instance"**.
2. **Name your instance** (e.g., "MyFirstVM").
3. **Choose Amazon Machine Image (AMI)**:
   * Select **"Ubuntu Server 22.04 LTS (Free tier eligible)"** or
   * **"Amazon Linux 2"** (also free tier eligible).
4. **Choose Instance Type**:
   * Select **t2.micro** (free tier eligible).
   * (This is small and free, enough for basic tasks.)
5. **Create a Key Pair** (important!):
   * Click **"Create new key pair"**.
   * Name it something like "mykey".
   * Key pair type: **RSA**.
   * Private key file format: **.pem**.
   * Download and **save** the .pem file carefully (you will need it to connect later).
6. **Configure Network Settings**:
   * Allow **SSH (port 22)** from **My IP**.
   * Allow **HTTP (port 80)** if you want to run web apps.
7. **Leave Storage as default**:
   * 8 GB is enough for now.
8. **Review and Launch**:
   * Click **"Launch Instance"**.

**5. Connect to Your VM**

1. Go back to EC2 dashboard.
2. Click on your instance name.
3. Click **"Connect"** → **"SSH client"**.
4. You will see connection command like:

ssh -i "mykey.pem" ubuntu@your-public-ip

**Important**:

* Open **Command Prompt** or **Git Bash** on your PC.
* Go to the folder where your .pem file is saved.
* Run the SSH command given.

**6. Basic Operations inside the VM**

After you are connected:

* Check OS:

uname -a

* Update packages:

sudo apt update

sudo apt upgrade

* Install basic things:

sudo apt install git

sudo apt install python3

* Create a basic text file:

echo "Hello from AWS VM" > hello.txt

cat hello.txt

**📋 Quick Summary:**

| **Step** | **What to Do** |
| --- | --- |
| 1 | Create AWS account |
| 2 | Open EC2 |
| 3 | Launch Instance (Ubuntu, t2.micro) |
| 4 | Create Key Pair (.pem) |
| 5 | Allow SSH (Port 22) |
| 6 | Launch and Connect using SSH |
| 7 | Do basic operations |

**PROBLEMS:**

* **Problem 1:** After copy pasting the SSH Command in CMD.

**Solution:** Go to security tab in instance info and scroll down to find 'Inbound rules' and edit the one with port 22 and change IP there to 0.0.0.0/0

* **Problem 2**: Even after fixing prblm 1 still there is a issue with permissions of .pem file.

**Solution**: Follow the below steps:-

1. Go to your folder C:\AWS

2. Right-click on MyKey.pem.

3. Click on Properties.

4. Go to the Security tab.

5. Click the Advanced button (bottom right).

6. At the top, click on Disable inheritance.

7. Choose Remove all inherited permissions from this object.

8. Now click on Add ➔ Select a principal.

9. In the box, type your username and click Check Names ➔ then OK.

10. Give only "Read" permission (tick "Read" checkbox), then click OK.

11. Keep clicking OK to close all windows.

**PRACTICAL 5:**

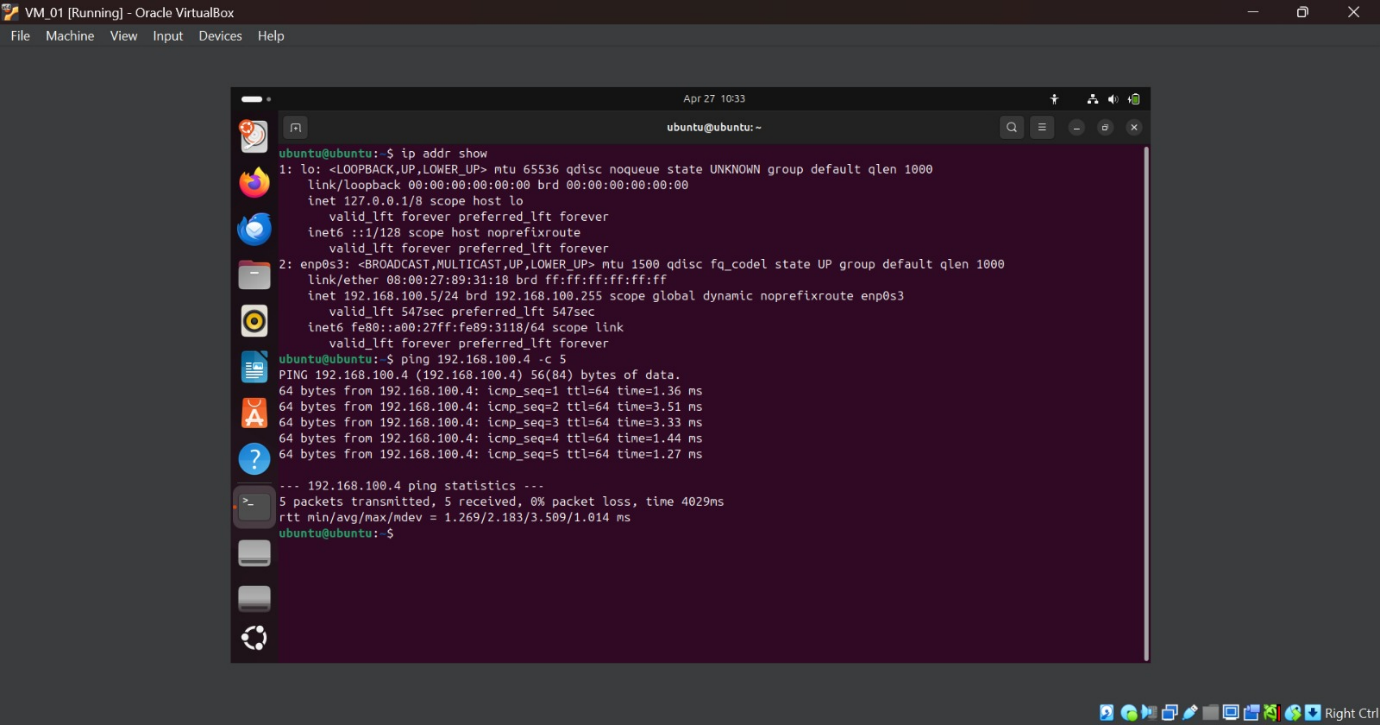
**NETWORK VIRTUALIZATION IN VMs:**

Step 1: Create 2 VMs

Step 2: Create a new adapter ***Ctrl+H*** (Host Only Adapter) ***DHCP change to 100***

Step 3: Go to ***'Network Settings'*** of both VMs and ***change*** to ***Host-Only Adapter*** and select newly created adapter

Step 4: Open both VMs and see their IPs (***ip addr show***) and ping each other as seen in image attached



**PRACTICAL 8:  
FIND A PROCEDURE TO TRANSFER THE FILES FROM ONE VIRTUAL MACHINE TO ANOTHER VIRTUAL MACHINE.**

Step 1: Create 2 VMs

Step 2: Create a new adapter ***Ctrl+H*** (Host Only Adapter) ***DHCP change to 100***

Step 3: Go to ***'Network Settings'*** of both VMs and ***change*** to ***Host-Only Adapter*** and select newly created adapter

Step 4: Open both VMs and see their IPs (***ip addr show***) and ***ping*** each other.

Step 5: Also add another NAT adapter attachment bcz we need internet access which we can’t do with only Host-only network. **You have to shut down the VM first to do this.**

Step 6: ***sudo apt update***

***sudo apt install openssh-server***

Step 7: ***echo “Hello VM2! I am VM1…” > hello.txt***

***Cat hello.txt***

Step 8: ***whoami*** (get username of VM)

Step 9: ***scp hello.txt vm-username\_vboxuser@192.168.100.4:/home/vboxuser/ --> Sending***

**PRACTICAL 6:**

**INSTALLATION AND CONFIGURATION OF VIRTUALIZATION USING KVM.**

Step 1: Make sure N/w adapter is NAT to access internet.

Step 2: sudo apt update && sudo apt upgrade

Step 3: sudo grep -c “svm\|vmx” /proc/cpuinfo

Step 4: sudo apt install cpu-checker

Step 5: sudo apt install qemu-kvm virt-manager libvirt-daemon-system libvirt-clients bridge-utils -y

Step 6: sudo systemctl enable libvirtd

sudo systemctl start libvirtd

sudo usermod -aG kvm vboxuser

sudo usermod -aG libvirt vboxuser

sudo systemctl status libvirtd

Step 7: Open “Virtual Machine Manager” you’ll see some errors… Restart the VM

Step 8: Reopen “Virtual Machine Manager” and click on top right icon under “File” to “Create a New Virtual Machine”.

Step 9: Select “Local install media”.

Step 10: Select “Browse” and click of Green Plus icon to add volume

Step 11: Keep default in next window and select “Choose volume”, then uncheck the “Automatically….“ Checkbox & write OS name as “ubuntu 24.10”.

Step 12: Select appropriate Memory and CPU and Finish

**PRACTICAL 7:**

**INSTALL AND CONFIGURE DOCKER AND KUBERNETES.**

**🛠️ STEP 1: Install Docker Desktop on Windows 11**

1. **Download Docker Desktop:**
   * Go to https://www.docker.com/products/docker-desktop/
   * Click on **Download for Windows (Windows 11 64-bit)**
2. **Install Docker Desktop:**
   * Run the installer (just double-click the .exe file you downloaded).
   * Keep clicking **Next**, **Accept** the terms.
   * Make sure "**Use WSL2 instead of Hyper-V**" is **checked**. (WSL2 = Windows Subsystem for Linux, required for Docker)
3. **Enable WSL2 (if not already done):**
   * Open **PowerShell** as Administrator and run:

wsl --install

* + Restart your laptop if it asks.

1. **Start Docker Desktop:**
   * After installation, search "**Docker Desktop**" in Start Menu and open it.
   * Docker will take a few seconds to start.
2. **Verify Docker Installation:**
   * Open **Command Prompt** or **PowerShell**.
   * Type:

docker --version

* + You should see something like:

Docker version 24.xx.xx, build xxxxx

**🛠️ STEP 2: Enable Kubernetes inside Docker Desktop**

1. **Open Docker Desktop** app.
2. Go to **Settings** (⚙️ gear icon) > **Kubernetes**.
3. Check the box: **Enable Kubernetes**.
4. Click **Apply & Restart**.

It will take 2-5 minutes. It sets up a **single-node Kubernetes cluster** locally.

✅ Now you have:

* Docker Engine
* Kubernetes cluster running on your laptop

**🛠️ STEP 3: Verify Kubernetes Installation**

1. Open **Command Prompt** or **PowerShell**.
2. Run:

kubectl version --client

kubectl get nodes

1. You should see your local machine as a **Ready** node.

**PRACTICAL 10:**

**AWS EC2 WINDOWS INSTANCE -TO LAUNCH AND CONNECT TO A WINDOWS INSTANCE AND TO REMOTELY CONNECT TO THE WINDOWS INSTANCE.**

### ****Step 1: Log in to AWS Management Console****

1. Go to the [AWS Management Console](https://aws.amazon.com/console/) and log in with your AWS account credentials.

### ****Step 2: Launch a Windows EC2 Instance****

#### 2.1: Open the EC2 Dashboard

1. In the AWS Management Console, in the search bar, type **EC2** and select **EC2** from the services list.
2. On the **EC2 Dashboard**, click **Launch Instance** to start creating a new instance.

#### 2.2: Choose an Amazon Machine Image (AMI)

1. In the **Choose an Amazon Machine Image (AMI)** section, select a **Windows AMI**. You’ll find various Windows versions like **Microsoft Windows Server 2022**, **Windows Server 2019**, etc.
   * For beginners, **Microsoft Windows Server 2022 Base** is a good choice.
   * Select **Windows Server 2022 Base** (or any version you'd like).
2. Click on the **Select** button next to the Windows Server AMI.

#### 2.3: Choose an Instance Type

1. Now, choose an **instance type**. For most basic tasks, the **t2.micro** instance is free-tier eligible (if you are on the Free Tier).
   * Select **t2.micro** for now, and then click **Next: Configure Instance Details**.

#### 2.4: Configure Instance

1. For a basic setup, the default settings work fine. You can leave them as is unless you want to change the network settings or instance details.
2. Click **Next: Add Storage** to proceed.

#### 2.5: Add Storage

1. The default storage size is usually fine. However, you can adjust it based on your requirements (e.g., 30 GB or more).
2. Click **Next: Add Tags**.

#### 2.7: Configure Security Group

1. A **Security Group** acts as a virtual firewall. You’ll need to create one for remote access.
2. Click **Add Rule** and configure the following:
   * **Type**: Select **RDP** (Remote Desktop Protocol).
   * **Protocol**: TCP.
   * **Port Range**: 3389 (default port for RDP).
   * **Source**: Choose **Anywhere** (0.0.0.0/0) or a specific IP range that you want to allow RDP access from.

This opens the necessary port (3389) to allow you to remotely connect to your Windows instance.

1. After adding the RDP rule, click **Review and Launch**.

#### 2.8: Review and Launch

1. Review all the settings and configurations you’ve chosen. Once you’re ready, click **Launch**.
2. You will now be prompted to create a new **Key Pair** or select an existing one:
   * If you don’t have one, select **Create a new key pair**.
   * Give your key pair a name (e.g., **WindowsKeyPair**).
   * Click **Download Key Pair** (this will download a .pem file, which is important for securely accessing the instance).
   * **Store the .pem file securely** (you’ll need this later for connecting via RDP).
3. After selecting the key pair, click **Launch Instances**.

Your Windows EC2 instance will now be launched. You can view its status in the **Instances** section of the EC2 Dashboard.

### ****Step 3: Connect to the Windows EC2 Instance****

#### 3.1: Get the Instance's Public IP

1. Go back to the **EC2 Dashboard** and select **Instances**.
2. Find your newly launched **Windows EC2 instance** in the list.
3. Copy the **Public IP** (or Public DNS) of the instance. You’ll use this IP to connect via **Remote Desktop Protocol (RDP)**.

#### 3.2: Get the Windows Administrator Password

1. Select the Windows instance, and then click on **Actions** > **Security** > **Get Windows Password**.
2. **Upload your private key** (.pem file) by clicking on **Browse** and selecting the **WindowsKeyPair.pem** file that you downloaded earlier.
3. Click **Decrypt Password**.
4. AWS will now provide you with the **Windows Administrator password**.

#### 3.3: Connect via RDP

##### Option 1: Using Windows (Built-in RDP Client)

1. Open **Remote Desktop Connection** (search for mstsc in the Start menu on Windows).
2. In the Remote Desktop Connection window, paste the **Public IP** or **DNS name** of the EC2 instance.
3. Click **Connect**.
4. When prompted for credentials:
   * Username: Administrator
   * Password: Use the **decrypted password** from AWS (the one you obtained in Step 3.2).
5. Click **OK** to connect. You should now be logged into your Windows EC2 instance.

**Step 4: Basic Operations on Windows EC2 Instance**

Once you’re logged into the instance, you can perform typical Windows operations:

1. **Install Software**: Just like any Windows machine, you can download and install software, use browsers, or run programs.
2. **Access the Windows Server**: You can configure the server, set up IIS (Internet Information Services), or perform other server-related tasks.
3. **Transfer Files**: You can upload or download files using **RDP** or tools like **WinSCP**.
4. **Monitor System Performance**: Use **Task Manager** or **Performance Monitor** to keep track of the instance's CPU, memory, and disk usage.