EXP 1

INSTALL VM

Step 1: Download the Linux ISO File

Step 2: Launch VirtualBox

Step 3: Create a New Virtual Machine

1. Click on "New" in the top-left corner to start creating a new virtual machine.
2. Name your virtual machine (e.g., "Linux VM").
3. Select the Type as "Linux" and the Version as the specific Linux version you're installing (e.g., Ubuntu 64-bit).

Step 4: Allocate Memory (RAM)

Step 5: Create a Virtual Hard Disk

1. Select Create a virtual hard disk now and click Create.
2. Choose VDI (VirtualBox Disk Image) and click Next.
3. Select Dynamically allocated
4. Set the hard disk size (at least 20 GB is recommended) and click Create.

Step 6: Configure the VM to Use the ISO File

1. Select your newly created VM from the list and click on Settings.
2. Go to the Storage tab.
3. Under Controller: IDE, click on the empty disk icon.
4. On the right, click Choose a disk file and navigate to the downloaded ISO file. Select it and click OK.

Step 7: Start the VM and Begin Installation

Step 8: Reboot and Use Your Virtual Linux Machine

EXP 2

INSTALL C COMPILER IN VM

Step 1: Open the Terminal

1. Start your Linux virtual machine in VirtualBox.
2. Open the terminal.

Step 2: Install the GCC Compiler

1. Install the GCC compiler, which is the standard compiler for C on Linux:

   sudo apt install gcc

2. After entering this command, type Y to confirm and press Enter. The installation process will complete in a few moments.

Step 4: Verify the Installation

1. Check that GCC installed correctly by running:

   gcc --version

Step 5: Write a Simple C Program

   nano hello.c

1. In the editor, type the following code:

```
#include <stdio.h>
int main() {
   printf("Hello, World!\n");
   return 0;
}
```

2. Press Ctrl + X to exit, then Y to save, and Enter to confirm.

Step 6: Compile the Program

   gcc hello.c -o hello

1. This will create an executable file named hello in the current directory.

Step 7: Run the Program

   ./hello

EXP 3

INSTALL GAE AND CREATE A SIMPLE WEB APP

Step 1: Open Google Cloud Platform

1. Open your web browser and go to Google Cloud Platform.

Step 2: Login to Your Account

Step 3: Navigate to the Dashboard

Step 4: Enable Google App Engine Admin API

1. In the dashboard, use the search bar at the top of the page to search for **Google App Engine Admin API**.
2. Click on the API in the search results and click **Enable** to activate it.

Step 5: Activate Cloud Shell

Step 6: Download the "Hello, World!" Web Application

1. Clone the sample "Hello, World!" application provided by Google:

   ```
   git clone https://github.com/GoogleCloudPlatform/python-docs-samples
   ```

2. This will download a collection of sample applications to your Cloud Shell environment.

Step 7: Change Directory to the "Hello, World!" Folder

```
cd python-docs-samples/appengine/standard_python3/hello_world
```

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"

if __name__ == "__main__":
    app.run(host="127.0.0.1", port=8080, debug=True)
```

Step 9: Check for Required Files **ls**

1. You should see files like main.py and app.yaml, which are essential for deploying the app.

Step 10: Execute the "main.py" File

```
python3 main.py
```

EXP 4

USE GAE LAUNCHER TO LAUNCH WEB APPLICATION – WEATHER APP

Step 1: Import the Google Cloud Public Key

curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo gpg --dearmor -o /usr/share/keyrings/cloud.google.gpg

Step 2: Add the gcloud CLI Distribution URI as a Package Source

echo "deb [signed-by=/usr/share/keyrings/cloud.google.gpg] https://packages.cloud.google.com/apt cloud-sdk main" | sudo tee -a /etc/apt/sources.list.d/google-cloud-sdk.list

Step 3: Update and Install the gcloud CLI

sudo apt-get update

sudo apt-get install google-cloud-cli

Step 4: Initialize Google Cloud SDK

gcloud init

Step 5: Select Configuration to Use

[1] Re-initialize this configuration [default] with new settings

[1] 22b120@psgitech.ac.in

[1] principal-truck-434504-n6

Step 8: Open Cloud Shell

gcloud cloud-shell ssh

Step 9: Create Your Web Application

nano weather.py

Step 10: Write the Program

```
import requests
import urllib.parse
API_KEY = 'af9e65722413ec1cbc3f68cdbb04794c'
BASE_URL = "http://api.openweathermap.org/data/2.5/weather?"


def get_weather(city_name):
    if not city_name.strip():
```

```
        print("City name cannot be empty!")
        return

    city_name_encoded = urllib.parse.quote(city_name)
    url = BASE_URL + "q=" + city_name_encoded + "&appid=" + API_KEY +
"&units=metric"
    response = requests.get(url)

    if response.status_code == 200:
        data = response.json()
        main = data['main']
        wind = data['wind']
        weather_desc = data['weather'][0]['description']

        print(f'City: {city_name}')
        print(f"Temperature: {main['temp']}°C")
        print(f"Humidity: {main['humidity']}%")
        print(f"Pressure: {main['pressure']} hPa")
        print(f"Weather Description: {weather_desc.capitalize()}")
        print(f"Wind Speed: {wind['speed']} m/s")
    else:
        print(f'City {city_name} not found, please check the city name.')

city = input("Enter city name: ").strip()
get_weather(city)
```

Step 11: Run the Program

1. Save and exit the editor by pressing **Ctrl + X**, then **Y**, and **Enter**.

   python3 weather.py

EXP 5

SIMULATIONG CLOUD SCENARIO USING CLOUDSIM AND EXECUTING SJF ALG

Step 1: Verify Java Installation

java -version

Step 2: Set Java Environment Path

sudo update-alternatives --config java

sudo gedit /etc/environment

- Paste the copied path into the file as JAVA_HOME="path"

source /etc/environment

Step 3: Download and Install CloudSim 3.0.3

1. Download **CloudSim 3.0.3** from GitHub at **https://github.com/Cloudslab/cloudsim/releases**

Step 4: Install Apache Ant

sudo apt install ant

Step 5: Verify CloudSim Installation (command in examples.txt - 43)

Cd directory to cloudsim

javac -classpath jars/cloudsim-3.0.3.jar:examples org/cloudbus/cloudsim/examples/CloudSimExample1.java

java -classpath jars/cloudsim-3.0.3.jar:examples org.cloudbus.cloudsim.examples.CloudSimExample1

Step 6: Implement Shortest Job First (SJF) Algorithm

Download the **DatacenterBroker.java** file from the GitHub repository: **https://github.com/koushal2001/cloudcomputing**

sources/org/cloudbus/cloudsim

Step 7: Add Simulation.java File

examples/org/cloudbus/cloudsim/examples

Step 8: Compile and Run Simulation.java

javac -classpath jars/cloudsim-3.0.3.jar:examples org/cloudbus/cloudsim/examples/Simulation.java

java -classpath jars/cloudsim-3.0.3.jar:examples org.cloudbus.cloudsim.examples.Simulation

EXP 6

TRANSFER FILES BETWEEN VIRTUAL MACHINES

Step 1: Create a NAT Network

1. Open **VirtualBox** and go to **File > Tools>Network Manager>NAT Networks**.Select **create**

Step 2: Configure Network Settings for VM1

1. Right-click on **VM1** and select **Settings**>**Network**
2. Change the **Attached to** option to **NAT Network**.
3. Select the NAT network created in Step 1.

Step 3: Configure Network Settings for VM2

Step 4: Start Both Virtual Machines

Sudo apt update on both VM.

Step 5: Find the IP Address on both VM. Ifconfig for checking in VM1 ping <ip of VM2>

Step 6: sudo apt install openssh-server

sudo systemctl enable ssh

sudo systemctl start ssh – VM2

sudo systemctl status ssh

Step 7: Create a file in VM1

pwd for current working directory of both VM

VM1 – touch hello.txt

echo hello world > hello.txt

cat hello.txt

Step 7: Transfer the File from VM1 to VM2

scp hello.txt <username_vm2>@<ip_vm2>:<pwd> username - whoami

Step 8: Authenticate and Complete the Transfer

Enter the **password** for the VM2 admin user when prompted. After authentication, the file transfer.txt will be transferred from **VM1** to **VM2. Check – ls and cat hello.txt**

EXP 8

DOCKER INSTALLATION AND EXECUTION

Step 1: Update package index

sudo apt update

Step 2: Install dependencies

sudo apt install apt-transport-https ca-certificates curl software-properties-common

Step 3: Add Docker's GPG key

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

Step 4: Add Docker repository

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"

Step 5: Update package index again

sudo apt update

Step 6: Install Docker CE (Community Edition)

sudo apt install docker-ce

Step 7: Verify Docker installation

sudo docker --version

Step 8: Start Docker service

sudo systemctl start docker

Step 9: Enable Docker on boot

sudo systemctl enable docker

Step 10: Pull an Ubuntu image

sudo docker pull ubuntu

Step 11: Run a Docker container interactively

sudo docker run -it ubuntu

Step 12: Exit the container

Exit

EXP 9

RUNNING A CONTAINER FROM DOCKER HUB

Step 1: Sign up for a free Docker account

1. Go to Docker's sign-up page and create a free account.

Step 2: Create a private repository

Step 3: Download and install Docker Desktop

1. Download and install Docker Desktop from the official website.

Step 4: Pull and run a container image from Docker Hub

1. Open a terminal or command prompt.
2. Pull a container image from Docker Hub (e.g., the "hello-world" image) using:

   docker pull hello-world
3. Run the container:

   docker run hello-world

Step 5: Build and push a container image to Docker Hub

To build and push your own container image to Docker Hub, follow these steps:

Step 6: Create a Dockerfile

1. Create a new directory for your project and navigate to it.
2. Inside that directory, create a Dockerfile with the following contents:

   FROM busybox

   CMD echo "Hello world! This is my first Docker image."

Step 7: Build the image

1. Build the Docker image using the docker build command:

   docker build -t <your-username>/my-private-repo .

Replace <your-username> with your Docker Hub username. This command will create a Docker image with the name my-private-repo.

Step 8: Push the image

1. Log in to Docker Hub using the docker login command and enter your credentials.
2. Push your Docker image to your private repository on Docker Hub:

   docker push <your-username>/my-private-repo