CLOUD ARCHITECTURE & SERVICES LABORATORY

LAB MANUAL

Academic Year : 2023 -2024
Course Code : 20AM6L02
Regulations : R -20
Semester : VI
Branch : AIML

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

SWARNANDHRA COLLEGE OF ENGINEERING AND TECHNOLOGY
(Autonomous)
SEETHARAMPURAM, NARSAPUR -534280

INDEX

EXPERIMENT-1

1.Create an word document of your class time table and store locally and on cloud with doc and pdf format

AIM:
The objective of this lab experiment is to create a class timetable using a word processing application, save the document locally, and then store it on a cloud platform in both DOC and PDF formats. Participants will gain hands-on experience in document creation, file storage, and document format conversion.

EXPERIMENT STEPS:
Step 1: Document Creation
- Open your preferred word processing application (e.g., Microsoft Word).
- Create a new document and format it to represent your class timetable.
- Include details such as class names, instructors, and time slots.

Step 2: Save Locally in DOC Format
- Save the document locally on your computer.
- Choose the DOC (Word Document) format when saving the file.

Step 3: Save Locally in PDF Format
- Save a copy of the document locally in PDF format.
- Verify that the PDF file accurately represents the content and formatting of the timetable.

Step 4: Set Up Cloud Storage
- Access your preferred cloud storage platform (e.g., Google Drive, Dropbox).
- Create a new folder for storing class-related documents.

Step 5: Upload Documents to Cloud
- Upload the locally saved DOC file to the cloud storage folder.
- Verify that the document is successfully uploaded and accessible online.

Step 6: Convert and Upload PDF to Cloud
- Convert the locally saved PDF file to DOC format (if not done previously).
- Upload the converted DOC file to the same cloud storage folder.
- Verify that both the DOC and PDF versions are now stored on the cloud.

Step 7: Access and Share from Cloud
- Access the cloud storage platform from a different device to ensure document availability.
- Share the cloud link with a classmate or instructor to demonstrate document sharing capabilities.

Conclusion:
This lab experiment provides hands-on experience in creating a class timetable document using a word processing application, saving it locally in DOC and PDF formats, and storing it on a cloud platform. Participants gain practical insights into document creation, file storage, and cloud-based collaboration.

EXPERIMENT-2

2. Implementation of Virtualization in Cloud Computing to Learn Virtualization Basics, Benefits of Virtualization in Cloud using Open Source Operating System

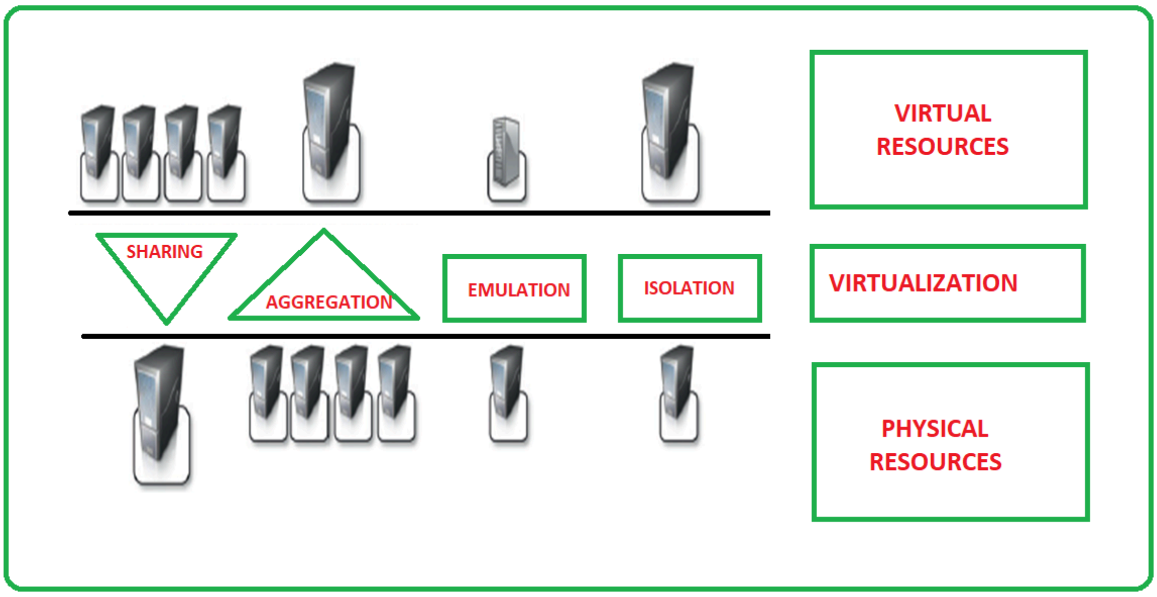
AIM:
The objective of this lab experiment is to gain hands-on experience in implementing virtualization in cloud computing using an open-source operating system. Participants will learn the basics of virtualization, understand the benefits of virtualization in the cloud, and work with relevant tools to create virtual machines.

EXPERIMENT STEPS:
Step 1: What is Virtualization Software
Virtualization is technology that you can use to create virtual representations of servers, storage, networks, and other physical machines. Virtual software mimics the functions of physical hardware to run multiple virtual machines simultaneously on a single physical machine.
The tool used for virtualization is virtualization software e.g **VirtualBox**.

Step 2: Explore about Virtual Machine
a computer system created using software on one physical computer in order to emulate the functionality of another separate physical computer.

Step 3: Open Source Operating System
The term "open source" refers to computer software or applications where the owners or copyright holders enable the users or third parties to use, see, and edit the product's source code. The source code of an open-source OS is publicly visible and editable.
Some basic examples of the open-source operating systems are Linux, Open Solaris, Free RTOS, Open BDS, Free BSD, Minix, etc.

Step 4: Virtualization Features



- Emulation
  Guest programs run within a controlled virtualization layer, allowing the emulation of diverse environments. This enables the execution of guest programs with specific characteristics not present in the physical host.

- Sharing
  Virtualization enables the creation of separate computing environments within the same host, reducing the number of active servers and limiting power consumption.

- Aggregation
  Virtualization allows the grouping of separate hosts, presented to guests as a single virtual host. Cluster management software facilitates this, utilizing the resources of multiple machines as a unified entity.

- Isolation
  Virtualization provides guests with a separate environment, interacting through an abstraction layer. The virtual machine filters and prevents harmful operations, ensuring secure execution.

Step 5: Implement Virtualization in Cloud

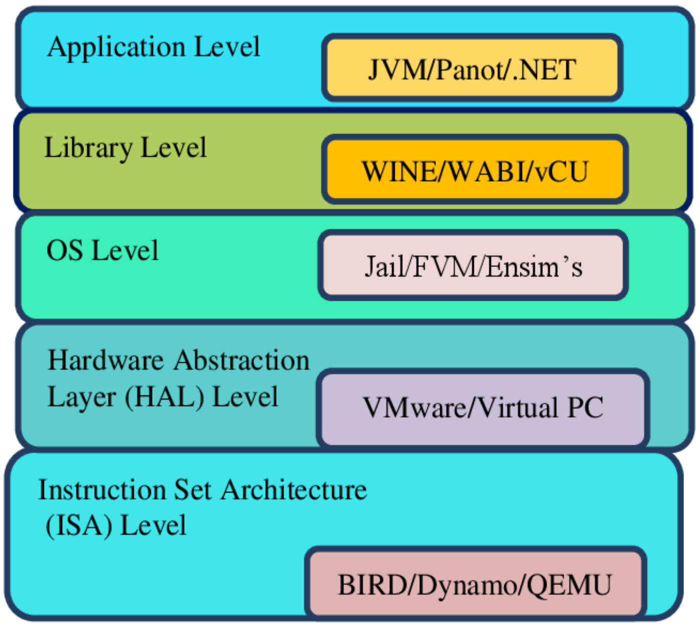


Fig. 2: Implementation Levels of Virtualization.

1. **Instruction Set Architecture Level (ISA):**
   ISA virtualization can work through ISA emulation. This is used to run many legacy codes written for a different hardware configuration. These codes run on any virtual machine using the ISA. With this, a binary code that originally needed some additional layers to run is now capable of running on the x86 machines. It can also be tweaked to run on the x64 machine. With ISA, it is possible to make the virtual machine hardware agnostic.

2. **Hardware Abstraction Level (HAL)**
   True to its name HAL lets the virtualization perform at the level of the hardware. This makes use of a hypervisor which is used for functioning. The virtual machine is formed at this level, which manages the hardware using the virtualization process. It allows the virtualization of each of the hardware components, which could be the input-output device, the memory, the processor, etc.
3. **Operating System Level**
   At the level of the operating system, the virtualization model is capable of creating a layer that is abstract between the operating system and the application. This is an isolated container on the operating system and the physical server, which uses the software and hardware. Each of these then functions in the form of a server.

4. **Library Level**
   The operating system is cumbersome, and this is when the applications use the API from the libraries at a user level. These APIs are documented well, and this is why the library virtualization level is preferred in these scenarios. API hooks make it possible as it controls the link of communication from the application to the system.

5. **Application Level**
   The application-level virtualization is used when there is a desire to virtualize only one application and is the last of the implementation levels of virtualization in Cloud Computing. One does not need to virtualize the entire environment of the platform.

Step 6: **Benefits of Virtualization in Cloud**

- **Security:**
  During the process of virtualization security is one of the important concerns. The security can be provided with the help of firewalls, which will help to prevent unauthorised access and will keep the data confidential.
  Moreover, with the help of firewalls and security, the data can protect from harmful viruses, malware and other cyber threats. Encryption process also takes place with protocols which will protect the data from other threads.
  So, the customer can virtualize all the data stores and can create a backup on a server in which the data can store.
- **Flexible operations**
  With the help of a virtual network, the work of its professionals is becoming more efficient and agile. The network switch implemented today is very easy to use, flexible and saves time.
- **Economical**
  Virtualization in Cloud Computing, save the cost for a physical system such as hardware and servers. It stores all the data in the virtual server, which is quite economical.

CONCLUSION:

This lab experiment provides hands-on experience in implementing virtualization in cloud computing using an open-source operating system. Participants gain practical insights into the basics of virtualization, understand the benefits of virtualization in the cloud, and learn to work with relevant tools to create and manage virtual machines.

EXPERIMENT-3

3.Install Virtual box/VMware Workstation with different flavours of linux or windows OS on top of windows7 or 8.

AIM:
The objective of this lab experiment is to gain hands-on experience in virtualization by installing VirtualBox or VMware Workstation on a Windows 7 or 8 host machine. Participants will further install different flavours of Linux and Windows operating systems as virtual machines to understand the virtualization process.

EXPERIMENT STEPS:
Step 1: Install Virtualization Software
- Download and install either VirtualBox or VMware Workstation on the Windows 7 or 8 host machine.
- Follow the installation wizard and configure settings as needed.

Step 2: Create a Virtual Machine for Linux OS
- Open the virtualization software and create a new virtual machine.
- Select Linux as the guest operating system and choose a flavour (e.g., Ubuntu).
- Allocate resources such as CPU, RAM, and storage.
- Mount the Linux ISO file as the installation media.

Step 3: Install Linux OS
- Start the virtual machine and proceed with the installation of the Linux operating system.
- Follow the installation wizard, selecting appropriate options.
- Create a user account and set up networking if required.

Step 4: Create a Virtual Machine for Windows OS
- Create a new virtual machine for a different flavour of the Windows operating system.
- Allocate resources and mount the Windows ISO file for installation.

Step 5: Install Windows OS
- Start the virtual machine and proceed with the installation of the Windows operating system.
- Follow the installation wizard, selecting appropriate options.
- Set up the Windows environment and create a user account.

Step 6: Explore Virtual Machine Features
- Investigate features provided by the virtualization software, such as snapshots, cloning, and resource adjustment.
- Create snapshots for both Linux and Windows virtual machines.

CONCLUSION:
This lab experiment provides practical experience in virtualization by installing VirtualBox or VMware Workstation on a Windows 7 or 8 host machine and creating virtual machines with different flavours of Linux and Windows operating systems. Participants gain insights into the configuration and management of virtualized environments.

EXPERIMENT-4

4.Install a C compiler in the virtual machine created using virtual box and execute Simple Programs.

AIM:
The objective of this lab experiment is to install a C compiler in a virtual machine created using VirtualBox and execute simple C programs. Participants will gain hands-on experience in setting up a development environment for C programming within a virtualized environment.

EXPERIMENT STEPS:

Step 1: Start Virtual Machine
- Start the virtual machine created using VirtualBox.
- Log in to the virtual machine using the provided credentials.

Step 2: Update Package Repository
- Open the terminal in the virtual machine.
- Update the package repository using the following command:
```
sudo apt-get update
```

Step 3: Install C Compiler (GCC)
- Install the GNU Compiler Collection (GCC) using the following command:
```
sudo apt-get install build-essential
```

Step 4: Verify GCC Installation
- Verify that GCC is successfully installed by checking the version:
```
gcc --version
```

Step 5: Create and Execute Simple C Program
- Using a text editor (e.g., Nano or Vim), create a simple C program.
  For example, create a file named `hello.c`:
  C Program
```
 #include <stdio.h>
int main() {
printf("Hello, C Programming!\n");
return 0;
}
```
- Save the file and compile it using GCC:
```
gcc -o hello hello.c
```
- Execute the compiled program:
```
./hello
```

CONCLUSION:
This lab experiment provides practical experience in setting up a C programming environment by installing the GCC compiler in a virtual machine created using VirtualBox. Participants gain hands-on experience in compiling and executing simple C programs, laying the foundation for further programming exploration.

EXPERIMENT-5

5.Write a Google app engine program to generate n even numbers and deploy it to Google cloud.

__AIM__:
The objective of this lab is to create a simple Google App Engine (GAE) program using Python and Flask to generate n even numbers and deploy it to Google Cloud.

__**Prerequisites:**__
- Google Cloud Platform (GCP) account
- Google Cloud SDK installed on your local machine

EXPERIMENT STEPS:

Step 1: Set up Google Cloud Project

1. Create a New GCP Project:
- Open the [Google Cloud Console](https://console.cloud.google.com/).
- Create a new project or use an existing one.

2. Enable App Engine API:
- Navigate to the "APIs & Services" > "Dashboard" in your project.
- Enable the "App Engine Admin API."

Step 2: Install Google Cloud SDK

1. Install Google Cloud SDK:
Download and install the [Google Cloud SDK](https://cloud.google.com/sdk/docs/install) on your local machine.

2. Initialize GCloud SDK:
Open a terminal and run:
```
gcloud init
```

Step 3: Create the App Engine Application

1. Create a Project Directory:
Open a terminal and create a new directory for your project:
```
mkdir gae_even_numbers
cd gae_even_numbers
```

2. Create `requirements.txt`:
Create a file named `**app.yaml**` in your project directory:
```
#requirements.txt
Flask==3.0.0
```

3. Install requirements.txt in this step:
Install Flask in your virtual environment:
```
pip install -r requirements.txt
```

4. Create `app.yaml`:
Create a file named `**app.yaml**` in your project directory:
```
#app.yaml
runtime: python39
```

5. Write the Flask App:

Create a file named `**main.py**` with the following code:

```
# main.py
from flask import Flask, jsonify

app = Flask(__name__)

@app.route('/generate_even_numbers/<int:n>', methods=['GET'])
def generate_even_numbers(n):
   even_numbers = [i * 2 for i in range(n)]
   return jsonify({'even_numbers': even_numbers})

if __name__ == '__main__':
   app.run()
```

Step 5: Deploy to Google Cloud

1. Deploy to Google Cloud:

Deploy the application to Google App Engine:

```
gcloud app deploy
```

2. Access the Deployed App:

After a successful deployment, access your application using the provided URL.

```
gcloud app browse
```

3. Test the Deployed App:

Open a web browser or use a tool like `curl` to test the deployed endpoint and Replace `**your-project-id**` with your actual project ID.

```
https://your-project-id.ew.r.appspot.com/generate_even_numbers/5
```

OUTPUT:

```
python main.py
`http://localhost:5000/generate_even_numbers/5`
{'even_numbers': 0,2,4,6,8}
```

CONCLUSION:

From this experiment we have created a Google App Engine program to generate n even numbers and deployed it to Google Cloud.

EXPERIMENT-6

6.Google app engine program multiply two matrices.

AIM

To create a Google App Engine program in Python to multiply two matrices and deploy it to Google Cloud.

Prerequisites:

- Google Cloud Platform (GCP) account
- Google Cloud SDK installed on your local machine

EXPERIMENT STEPS:

Step 1: Set up Google Cloud Project

1. Create a New GCP Project:

- Open the [Google Cloud Console](https://console.cloud.google.com/).
- Create a new project or use an existing one.

2. Enable App Engine API:

- Navigate to the "APIs & Services" > "Dashboard" in your project.
- Enable the "App Engine Admin API."

Step 2: Install Google Cloud SDK

1. Install Google Cloud SDK:

Download and install the [Google Cloud SDK](https://cloud.google.com/sdk/docs/install) on your local machine.

2. Initialize GCloud SDK:

Open a terminal and run: ***gcloud init***

Step 3: Create the App Engine Application

1. **Create a Project Directory:**Open a terminal and create a new directory for your app:
   ```
   mkdir matrix-multiplication-app
   cd matrix-multiplication-app
   ```
2. **Create `requirements.txt`:**Create a file named `**app.yaml**` in your project directory:
   ```
   #requirements.txt
   Flask==3.0.0
   ```
3. **Install requirements.txt in this step:** Install Flask in your virtual environment:
   ```
   pip install -r requirements.txt
   ```

4. **Create `app.yaml`:** Create a file named `**app.yaml**` in your project directory:
   ```
   #app.yaml
   runtime: python39
   ```
5. **Write the Flask App:**
   Create a file named `**main.py**` with the following code:

```
from flask import Flask, requests, jsonify

app = Flask(__name__)
```

```
def matrix_multiply(matrix_a, matrix_b):
    result = [[0] * len(matrix_b[0]) for _ in range(len(matrix_a))]
    for i in range(len(matrix_a)):
        for j in range(len(matrix_b[0])):
            for k in range(len(matrix_b)):
                result[i][j] += matrix_a[i][k] * matrix_b[k][j]
    return result

@app.route('/multiply', methods=['POST'])
def multiply_matrices():
    data = request.get_json()
    matrix_a = data['matrix_a']
    matrix_b = data['matrix_b']

    result = matrix_multiply(matrix_a, matrix_b)

    return jsonify({'result': result})

if __name__ == '__main__':
    app.run(debug=True)
```

Step 5: Deploy to Google Cloud

1. Deploy to Google Cloud with the following command**:** ***gcloud app deploy***
2. Access the Deployed App with the following command**:** ***gcloud app browse***
3. Test the Deployed App:

Open a web browser or use a tool like `curl` to test the deployed endpoint and Replace `**your-project-id**` with your actual project ID.

```
curl -X POST -H "Content-Type: application/json" -d "{\"matrix_a\": [[1,2],[3,4]],
\"matrix_b\": [[5,6],[7,8]]}" http://your-project-id.ew.r.appspot.com/multiply
```

OUTPUT:

```
python main.py

curl -X POST -H "Content-Type: application/json" -d "{\"matrix_a\":[[1,2],[3,4]],
\"matrix_b\": [[5,6],[7,8]]}" http://127.0.0.1:5000/multiply
{
'result':[
        [7,10],
        [15,22]
        ]
}
```

CONCLUSION:

From this experiment we have created a Google App Engine program to multiply two matrices and deployed it to Google Cloud.

EXPERIMENT-7

7.Google app engine program to validate user; create a database login(username, password)in mysql and deploy to the cloud.

AIM
To create a Google App Engine program in Python to validate user; create a database login(username, password)in mysql and deploy it to Google Cloud.

Prerequisites:
- Google Cloud Platform (GCP) account
- Google Cloud SDK installed on your local machine

EXPERIMENT STEPS:

Step 1: Set up Google Cloud Project
1. Create a New GCP Project:
- Open the [Google Cloud Console](https://console.cloud.google.com/).
- Create a new project or use an existing one.

2. Enable App Engine API:
- Navigate to the "APIs & Services" > "Dashboard" in your project.
- Enable the "App Engine Admin API."

Step 2: Install Google Cloud SDK
1. Install Google Cloud SDK:
Download and install the [Google Cloud SDK](https://cloud.google.com/sdk/docs/install) on your local machine.
2. Initialize GCloud SDK using this command: ***gcloud init***

Step 3: Create the App Engine Application
1. **Create a Project Directory:** Open a terminal and create a new directory for your app:
```
mkdir user-login-app
cd user-login-app
```
2. **Create `requirements.txt`:**Create a file named `**app.yaml**` in your project directory:
```
#requirements.txt
Flask==3.0.0
PyMySQL==1.0.3
```
3. **Install requirements.txt in this step:**
Install all required packages for your: ***pip install -r requirements.txt***
4. **Create `app.yaml`:**Create a file named `**app.yaml**` in your project directory:
```
#app.yaml
runtime: python39
env_variables:
  MYSQL_HOST: 'localhost'
  SQL_USERNAME: 'CseLab2'
```
5. **Write the Flask App:**

Create a file named `**main.py**` with the following code:

```
# app.py
import os
from flask import Flask, request, jsonify
import pymysql

db_config = {
    'host': os.environ.get('MYSQL_HOST'),
    'user': os.environ.get('SQL_USERNAME'),
    'database': os.environ.get('SQL_DATABASE_NAME','users'),
}


app = Flask(__name__)

cnx = pymysql.connect(**db_config)

cursor = cnx.cursor()

# User login route
@app.route('/login', methods=['POST'])
def login():
    data = request.json
    username = data.get('username')
    password = data.get('password')

    if not username or not password:
        return jsonify({'error': 'Both username and password are required'}), 400

    # Retrieve user from the database
    cursor.execute("SELECT * FROM login WHERE username = %s", (username,))
    user = cursor.fetchone()

    if not user:
        return jsonify({'error': 'User not found'}), 404

    elif user and user[1] == password:
        return jsonify({'message': 'Login successful'})
    else:
        return jsonify({'message': 'Invalid credentials'})



if __name__ == '__main__':
    app.run(debug=True)
```

<u>Step 4: Deploy to Google Cloud</u>

1. Deploy to Google Cloud: ***gcloud app deploy***
2. Access the Deployed App: ***gcloud app browse***
3. Test the Deployed App:

Open a web browser or use a tool like `curl` to test the deployed endpoint and Replace `**your-project-id**` with your actual project ID.

```
curl -X POST -H "Content-Type: application/json" -d "{\"username\": \"test\",
\"password\": “test123”}" http://your-project-id.ew.r.appspot.com/login
```

<u>OUTPUT</u>

python main.py

curl -X POST -H "Content-Type: application/json" -d "{\"*username*\": \"test\", \"*password*\": \"test123\"}" http://127.0.0.1:5000/login

{'message': 'Login successful'}

curl -X POST -H "Content-Type: application/json" -d "{\"*username*\": \"t\", \"*password*\": \"test123\"}" http://127.0.0.1:5000/login

{'message': 'User not found'}

curl -X POST -H "Content-Type: application/json" -d "{\"*username*\": \"test\", \"*password*\": \"test\"}" http://127.0.0.1:5000/login

{'message': 'Invalid credentials'}

CONCLUSION:
Google app engine program to validation of user by using mysql is performed

EXPERIMENT-8

8.Write a Google app engine program to display nth largest no from the given list of numbers and deploy it in Google cloud.

AIM
To create a Google App Engine program in Python to to display nth largest no from the given list of numbers and deploy it in Google cloud.

Prerequisites:

- Google Cloud Platform (GCP) account
- Google Cloud SDK installed on your local machine

EXPERIMENT STEPS:

Step 1: Set up Google Cloud Project

1. Create a New GCP Project:
- Open the [Google Cloud Console](https://console.cloud.google.com/).
- Create a new project or use an existing one.

2. Enable App Engine API:
- Navigate to the "APIs & Services" > "Dashboard" in your project.
- Enable the "App Engine Admin API."

Step 2: Install Google Cloud SDK

1. Install Google Cloud SDK:
Download and install the [Google Cloud SDK](https://cloud.google.com/sdk/docs/install) on your local machine.

2. Initialize GCloud SDK:
Open a terminal and run: ***gcloud init***

Step 3: Create the App Engine Application

1. Create a Project Directory:
Open a terminal and create a new directory for your project:

```
mkdir matrix-multiplication-app
cd matrix-multiplication-app
```

2. Create `requirements.txt`:
Create a file named `**app.yaml**` in your project directory:

```
#requirements.txt
Flask==3.0.0
```

3. Install requirements.txt in this step:
Install Flask in your virtual environment:

```
pip install -r requirements.txt
```

4. Create `app.yaml`:
Create a file named `**app.yaml**` in your project directory:

```
#app.yaml
runtime: python39
```

5. Write the Flask App:
Create a file named `**main.py**` with the following code:

```
from flask import Flask, request

app = Flask(__name__)

@app.route('/')
def display_nth_largest():
    numbers = [int(x) for x in request.args.get('numbers').split(',')]
    n = int(request.args.get('n'))
    nth_largest = sorted(numbers, reverse=True)[n-1]
    return f'The {n}th largest number is: {nth_largest}'

if __name__ == '__main__':
    app.run()
```

__Step 5: Deploy to Google Cloud__

1. Create `app.yaml`:

Create a file named `**app.yaml**` in your project directory:

```
#app.yaml
runtime: python39
```

2. Deploy to Google Cloud:

Deploy the application to Google App Engine: ***gcloud app deploy***

3. Access the Deployed App:

After a successful deployment, access your application using the provided URL.

```
gcloud app browse
```

4. Test the Deployed App:

Open a web browser or use a tool like `curl` to test the deployed endpoint and Replace `**your-project-id**` with your actual project ID.

http://*your-project-id.ew.r.appspot.com/*?numbers=1,2,3,4,5&n=2

__OUTPUT__

```
python main.py
http://localhost:5000/?numbers=10,2,4,3,5&n=2
The 2th largest number is 5
```

__CONCLUSION:__

From this experiment we have created a Google App Engine program to display nth largest no from the given list of numbers and deploy it in Google cloud.

EXPERIMENT-9

9.Google app engine program to validate the user use mysql to store user info and deploy on to cloud.

AIM

Google app engine program to validate the user use mysql to store user info and deploy on to cloud.

Prerequisites:

- Google Cloud Platform (GCP) account
- Google Cloud SDK installed on your local machine

EXPERIMENT STEPS:

Step 1: Set up Google Cloud Project

1. Create a New GCP Project:

- Open the [Google Cloud Console](https://console.cloud.google.com/).
- Create a new project or use an existing one.

2. Enable App Engine API:

- Navigate to the "APIs & Services" > "Dashboard" in your project.
- Enable the "App Engine Admin API."

Step 2: Install Google Cloud SDK

1. Install Google Cloud SDK:

Download and install the [Google Cloud SDK](https://cloud.google.com/sdk/docs/install) on your local machine.

2. Initialize GCloud SDK:

Open a terminal and run: ***gcloud init***

Step 3: Create the App Engine Application

1. Create a Project Directory:

Open a terminal and create a new directory for your project:

```
mkdir user-register-app
cd user-register-app
```

2. Create `requirements.txt`:

Create a file named `**app.yaml**` in your project directory:

```
#requirements.txt
Flask==3.0.0
PyMySQL==1.0.3
```

3. Install requirements.txt in this step:

Install Flask in your virtual environment:

```
pip install -r requirements.txt
```

4. Create `app.yaml`:

Create a file named `**app.yaml**` in your project directory:

```
#app.yaml
runtime: python39
env_variables:
  MYSQL_HOST: 'localhost'
```

SQL_USERNAME: 'CseLab2'

5. Write the Flask App:
Create a file named `**main.py**` with the following code:

```
# app.py
import os
from flask import Flask, request, jsonify
import pymysql

db_config = {
   'host': os.environ.get('MYSQL_HOST'),
   'user': os.environ.get('SQL_USERNAME'),
   'database': os.environ.get('SQL_DATABASE_NAME','users'),
}


app = Flask(__name__)

cnx = pymysql.connect(**db_config)

cursor = cnx.cursor()

# User registration route
@app.route('/register', methods=['POST'])
def register():
   data = request.json
   username = data.get('username')
   password = data.get('password')

   if not username or not password:
      return jsonify({'error': 'Both username and password are required'}), 400

   # Check if username already exists
   cursor.execute("SELECT * FROM login WHERE username = %s", (username,))
   if cursor.fetchone():
      return jsonify({'error': 'Username already exists'}), 400

   # Insert new user into the database
   cursor.execute("INSERT INTO login (username, password) VALUES (%s, %s)",
(username, password))
   cnx.commit()

   return jsonify({'message': 'User registered successfully'}), 201


if __name__ == '__main__':
   app.run(debug=True)
```

<u>Step 5: Deploy to Google Cloud</u>
1. Deploy to Google Cloud using this command**:** ***gcloud app deploy***
2. Access the Deployed App using this command: ***gcloud app browse***
3.Test the Deployed App:

Open a web browser or use a tool like `curl` to test the deployed endpoint and Replace `**your-project-id**` with your actual project ID.

curl -X POST -H "Content-Type: application/json" -d "{\"*username*\": "test",\"*password*\": "test123"}" http://127.0.0.1:5000/register

OUTPUT

python main.py

curl -X POST -H "Content-Type: application/json" -d "{\"*username*\": "test", \"*password*\": "test123"}" http://127.0.0.1:5000/register

{'message': 'User registered successfully'}

curl -X POST -H "Content-Type: application/json" -d "{\"*username*\": "test", \"*password*\": "test123"}" http://127.0.0.1:5000/register

{'message': 'Username already exists'}

CONCLUSION:

From this experiment we created a Google app engine program we performed the registration of the user to validate the user by using mysql to store user info and deployed on to cloud.

EXPERIMENT-10

10.Implement prog1-5 using Microsoft Azure.

AIM
Implement prog1-5 using Microsoft Azure.

Prerequisites:
- Microsoft Azure account

EXPERIMENT STEPS:

Step 1: Sign up for Azure: Go to [azure.com](https://azure.microsoft.com/) and sign up if you don't have an account.

Step 2: Create a Function App:
- Navigate to the Azure portal.
- Click on "Create a resource" and search for "Function App".
- Fill in the required details and create the app.

Step 3: Write and Deploy Your Functions:
1. Install Azure Functions Core Tools on your local system.
2. Use Azure Functions Core Tools or the Azure portal to create a Function App in your Azure account.
3. Develop your function code locally, ensuring each program (**exp4, exp5, exp6, exp7, exp8, exp9**) is implemented as a separate Azure Function.
4. Deploy your function to Azure using Azure Functions Core Tools or Azure CLI. Use the following command:
   func azure functionapp publish <FunctionAppName>

Step 4: Test and Monitor:
1. After deployment, test each function in the Azure portal to ensure they are working correctly.
2. Monitor the performance of your functions using Azure monitoring tools.

CONCLUSION:
Hence The prog1-5 (exp4,exp5,exp6,exp7,exp8,exp9) are deployed using Microsoft Azure Function in the Microsoft Azure Cloud.

.

EXPERIMENT-11

11.Establish an AWS account. Use the AWS Management Console to launch an EC2 instance and connect to it.

AIM

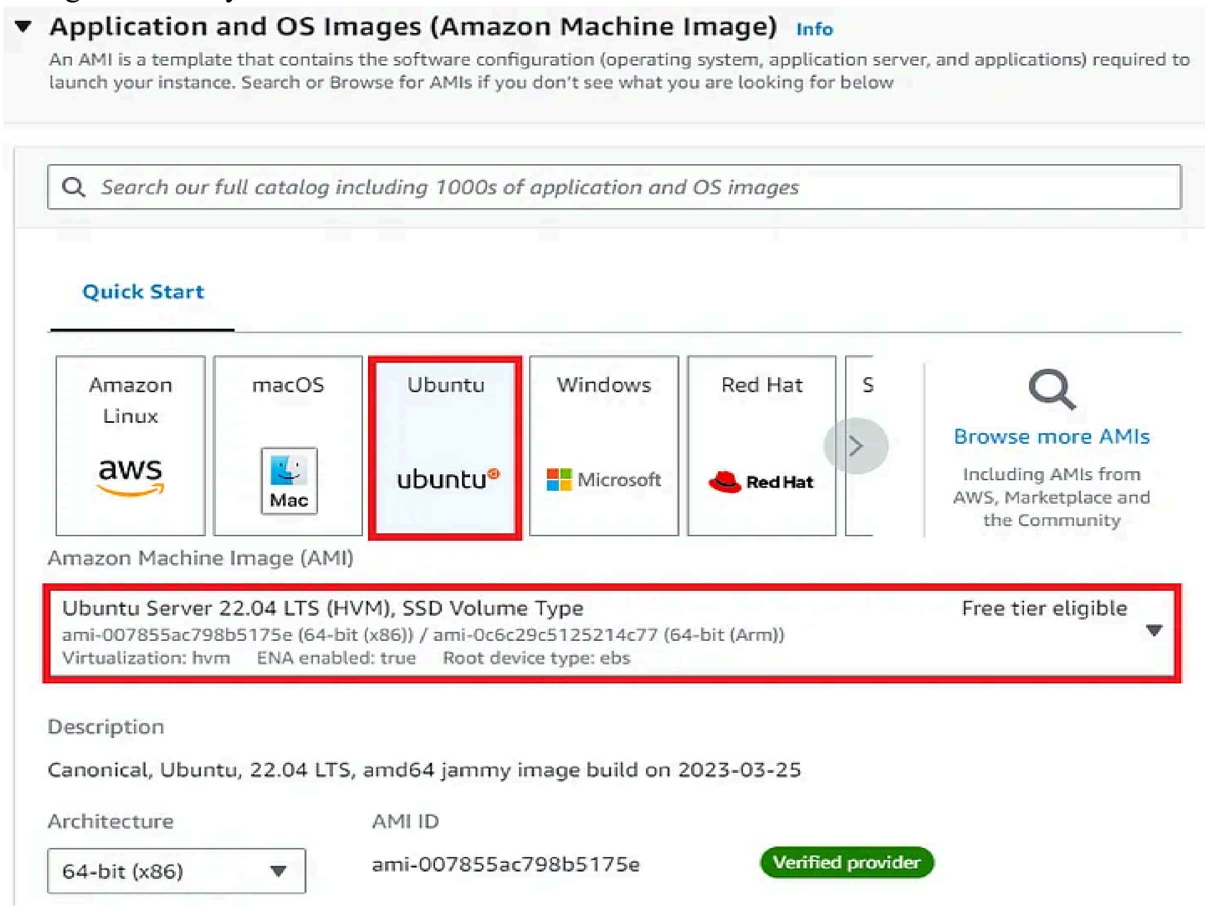
Establish an AWS account. Use the AWS Management Console to launch an EC2 instance and connect to it.

Prerequisites:

- AWS cloud account

EXPERIMENT STEPS:

1. **Sign up for an AWS Account:**Visit the AWS website (https://aws.amazon.com/) and sign up for an AWS account if you haven't already done so. Follow the prompts to provide necessary information and create your account.
2. **Access AWS Management Console:**After creating your account, log in to the AWS Management Console using your credentials.
3. **Navigate to EC2 Dashboard:**Once logged in, navigate to the EC2 Dashboard. You can find it under the "Compute" section or by searching for "EC2" in the AWS Management Console search bar.
4. **Launch an EC2 Instance:**In the EC2 Dashboard, click on the "Launch Instance" button to initiate the instance creation process.
   Follow the steps provided in the instance creation wizard:
   - Choose an Amazon Machine Image (AMI): Select the operating system and configuration for your EC2 instance.



▼ Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images
Quick Start
Amazon Linux
aws
macOS
Mac
Ubuntu
ubuntu
Windows
Microsoft
Red Hat
Red Hat
S
Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type — Free tier eligible
ami-007855ac798b5175e (64-bit (x86)) / ami-0c6c29c5125214c77 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

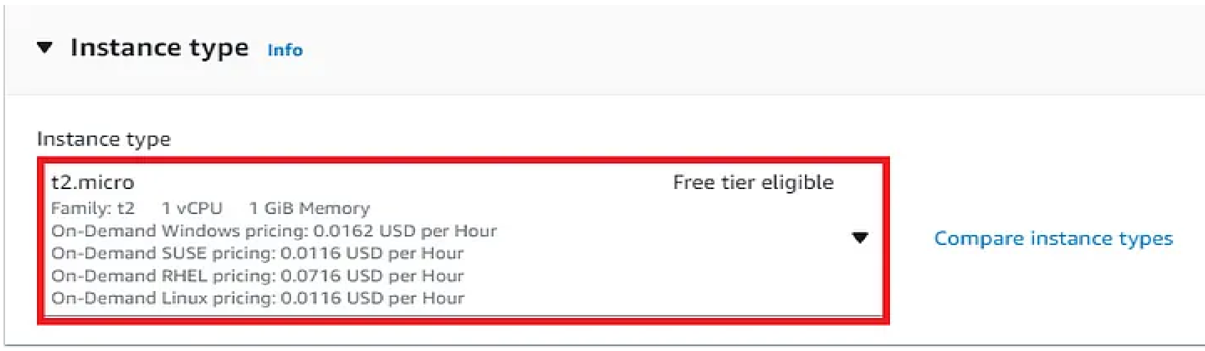
Description

Canonical, Ubuntu, 22.04 LTS, amd64 jammy image build on 2023-03-25

Architecture: 64-bit (x86)

AMI ID: ami-007855ac798b5175e

   - Choose an Instance Type: Select the type of EC2 instance based on your requirements and workload.



▼ Instance type Info

Instance type

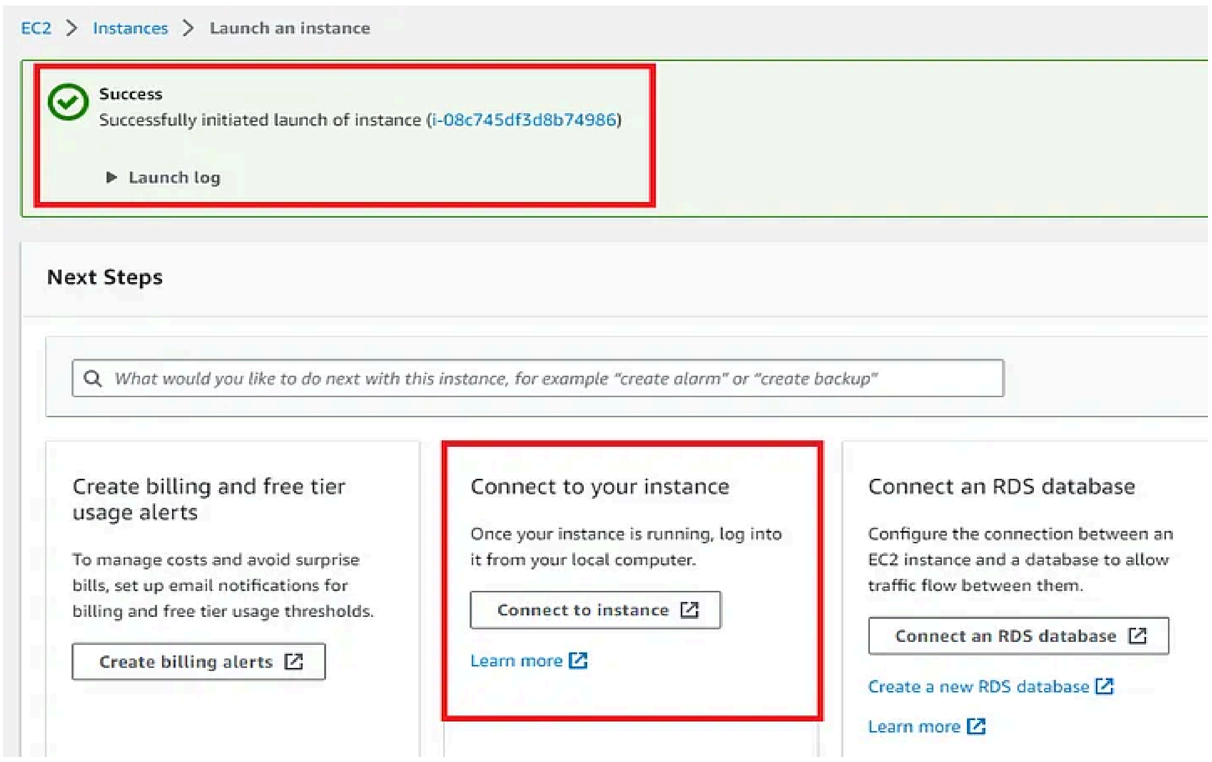
t2.micro — Free tier eligible
Family: t2 1 vCPU 1 GiB Memory
On-Demand Windows pricing: 0.0162 USD per Hour
On-Demand SUSE pricing: 0.0116 USD per Hour
On-Demand RHEL pricing: 0.0716 USD per Hour
On-Demand Linux pricing: 0.0116 USD per Hour

Compare instance types

- Configure Instance Details: Set up additional configuration options such as network settings, security groups, etc.
- Add Storage: Specify the storage requirements for your EC2 instance.
- Add Tags (Optional): Add tags to your instance for better organization and management.
- Configure Security Group: Define the security settings and rules for accessing your instance.

- Review and Launch: Review the configuration settings for your instance and launch it.

5. **Connect to the EC2 Instance:**
    - Once the instance is launched, select it from the EC2 Dashboard.
    - Under the "Connect" tab, you'll find instructions on how to connect to your instance. Depending on your operating system and preferences, you can either connect via SSH (for Linux instances) or RDP (for Windows instances).
    - Follow the provided instructions to establish a connection to your EC2 instance.



EC2 > Instances > Launch an instance

Success
Successfully initiated launch of instance (i-08c745df3d8b74986)

▶ Launch log

Next Steps

What would you like to do next with this instance, for example "create alarm" or "create backup"

Create billing and free tier usage alerts

To manage costs and avoid surprise bills, set up email notifications for billing and free tier usage thresholds.

Create billing alerts

Connect to your instance

Once your instance is running, log into it from your local computer.

Connect to instance

Learn more

Connect an RDS database

Configure the connection between an EC2 instance and a database to allow traffic flow between them.

Connect an RDS database

Create a new RDS database

Learn more

CONCLUSION:

In this experiment, we successfully established an AWS account, utilized the AWS Management Console to launch an EC2 instance, and connected to it. This process demonstrates the fundamental steps involved in provisioning and accessing compute resources in the AWS cloud environment.