

Assignment Answer Guidelines

this chat was for only assignment answers

Got it, Adiii 👍

This chat will be only for assignment answers.

From now on, send me:

- 📌 Assignment questions
- 📷 Photos/screenshots of questions
- 📄 Unit-wise questions

I'll give you **proper exam/assignment-ready answers** in simple language, with headings, points, explanations, and diagrams/tables wherever needed.

Deploy "Hello World" app using python on Google App Engine. (With all steps).

Deploy "Hello World" App Using Python on Google App Engine

Google App Engine (GAE) is a cloud platform that allows developers to build and deploy web applications without managing servers.

Step 1: Install Required Software

Install the following:

1. Python

Check installation:

```
Bash
python --version
```

2. Google Cloud CLI (gcloud)

Check installation:

```
Bash
gcloud --version
```

Step 2: Create a Google Cloud Project

1. Open Google Cloud Console.
2. Create a new project.

3. Give the project a name, for example:

hello-world-project

4. Note the **Project ID**.

Set the project using:

```
Bash
gcloud config set project PROJECT_ID
```

Replace **PROJECT_ID** with your actual Project ID.

Step 3: Create a Project Folder

Open Command Prompt or Terminal and create a folder:

```
Bash
mkdir hello-world-app
cd hello-world-app
```

Step 4: Create main.py

Create a file named **main.py** and write the following Python code:

```
Python
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello, World! Welcome to Google App Engine."
```

This program creates a simple web application. When the user opens the home page /, it displays the **Hello World** message.

Step 5: Create requirements.txt

Create a file named **requirements.txt**:

```
Flask==3.0.0
unicorn==21.2.0
```

This file contains the required Python libraries.

Step 6: Create app.yaml

Create another file named **app.yaml**:

```
YAML
runtime: python312
entrypoint: gunicorn -b :$PORT main:app
```

Here:

- **runtime: python312** specifies Python 3.12 runtime.
- **entrypoint** specifies the command used to start the application.
- **main:app** means:
 - **main** → **main.py** file
 - **app** → Flask application object

Step 7: Authenticate Google Cloud CLI

Login to your Google account:

```
Bash
gcloud auth login
```

A browser window will open. Sign in using your Google account and allow access.

Step 8: Initialize the App Engine Application

Run:

```
Bash
gcloud app create
```

Select the required **region/location** when prompted.

For example, select a region suitable for your project.

Step 9: Deploy the Application

Deploy the application using:

```
Bash
gcloud app deploy
```

When asked for confirmation, type:

```
Y
```

Google Cloud will upload and deploy the application to App Engine.

Step 10: Open the Deployed Application

After successful deployment, run:

```
Bash
gcloud app browse
```

The application will open in the browser.

The output will be:

```
Hello, World! Welcome to Google App Engine.
```

Project Folder Structure

```
hello-world-app/
├── main.py
├── requirements.txt
└── app.yaml
```

Summary of Commands

```
Bash
mkdir hello-world-app
cd hello-world-app

gcloud auth login

gcloud config set project PROJECT_ID

gcloud app create

gcloud app deploy

gcloud app browse
```

Result

Thus, a simple **“Hello World” web application using Python was successfully created and deployed on Google App Engine.**

What is Amazon EC2 web service? Explain the prominent features of EC2. Develop a simple Hello World program in EC2 and explain it step wise.

for assignment write up task

What is Amazon EC2 Web Service? Explain the Prominent Features of EC2. Develop a Simple Hello World Program in EC2 and Explain it Step Wise.

1. Introduction to Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2) is a web service provided by Amazon Web Services (AWS) that offers **resizable virtual servers**, called **instances**, in the cloud.

It allows users to rent computing resources such as CPU, memory, storage, and networking according to their requirements. Users can quickly launch, configure, and manage virtual machines without purchasing and maintaining physical servers.

The word **Elastic** means that computing resources can be increased or decreased easily according to application demand.

2. Prominent Features of Amazon EC2

1. Elastic Computing

EC2 allows users to increase or decrease computing capacity according to their requirements. Resources can be scaled when application traffic increases.

2. Virtual Servers (Instances)

EC2 provides virtual machines called **instances**. Different types of instances are available based on CPU, memory, storage, and application requirements.

3. Multiple Operating Systems

Users can launch instances with different operating systems such as:

- Linux
- Ubuntu
- Windows Server
- Amazon Linux

4. Amazon Machine Images (AMI)

An **Amazon Machine Image (AMI)** is a pre-configured template used to launch an EC2 instance. It contains the operating system and may also include required software and configurations.

5. Scalability

EC2 supports easy scaling of applications. Users can launch additional instances when demand increases and stop unnecessary instances when demand decreases.

6. Flexible Pricing

EC2 provides different pricing options, allowing users to pay according to their usage and requirements.

7. Security

EC2 provides security using features such as:

- Key pairs for secure login
- Security Groups acting as virtual firewalls
- Access control using AWS IAM
- Private and public networking options

8. Storage Support

EC2 supports different storage services such as **Amazon EBS (Elastic Block Store)** for persistent storage.

9. Easy Management

Instances can be created, started, stopped, rebooted, and terminated using the **AWS Management Console**, AWS CLI, or APIs.

10. High Availability

EC2 can be deployed in different Availability Zones and Regions, helping to improve the availability and reliability of applications.

3. Developing a Simple Hello World Program on Amazon EC2

In this example, a simple Python web application is created and deployed on an EC2 instance using **Flask**.

Step 1: Login to AWS Console

1. Open the AWS Management Console.
2. Login using your AWS account.
3. Search for **EC2**.
4. Open the EC2 Dashboard.

Step 2: Launch an EC2 Instance

1. Click on **Launch Instance**.

2. Enter an instance name, for example:

HelloWorld-EC2

3. Select an AMI such as **Ubuntu Server** or **Amazon Linux**.

4. Select a suitable instance type, such as:

t2.micro

5. Create or select a **Key Pair** for secure SSH login.

6. Configure the network settings.

7. Create a **Security Group**.

Step 3: Configure Security Group

Allow the following inbound traffic:

Type	Protocol	Port
SSH	TCP	22
Custom TCP	TCP	5000

- Port **22** is used for SSH connection.
- Port **5000** is used to access the Flask application.

After configuration, click **Launch Instance**.

Step 4: Connect to the EC2 Instance

After the instance starts, connect to it using SSH.

For Ubuntu:

```
Bash
ssh -i "my-key.pem" ubuntu@PUBLIC_IP_ADDRESS
```

Replace:

- **my-key.pem** with your key pair file name.
- **PUBLIC_IP_ADDRESS** with the public IP address of the EC2 instance.

Step 5: Update the System

Run the following commands:

```
Bash
sudo apt update
sudo apt upgrade -y
```

These commands update the Ubuntu system packages.

Step 6: Install Python and Flask

Check Python installation:

```
Bash
python3 --version
```

Install Flask:

```
Bash
sudo apt install python3-pip -y
pip3 install flask
```

Step 7: Create the Hello World Program

Create a Python file:

```
Bash
nano app.py
```

Write the following program:

```
Python
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World! Welcome to Amazon EC2."

app.run(host="0.0.0.0", port=5000)
```

Save the file.

Step 8: Run the Application

Execute the Python program:

```
Bash
```

```
python3 app.py
```

The Flask server will start on port **5000**.

The application listens on:

```
http://0.0.0.0:5000
```

Step 9: Access the Application

Open a web browser and enter:

```
http://PUBLIC_IP_ADDRESS:5000
```

For example:

```
http://54.xxx.xxx.xxx:5000
```

The browser displays:

```
Hello World! Welcome to Amazon EC2.
```

4. Explanation of the Program

```
Python
from flask import Flask
```

This imports the Flask library used to create the web application.

```
Python
app = Flask(__name__)
```

This creates a Flask application object.

```
Python
@app.route("/")
```

This defines the home page URL of the application.

```
Python
def hello():
    return "Hello World! Welcome to Amazon EC2."
```

This function is executed when the user opens the home page and returns the Hello World message.

```
Python
app.run(host="0.0.0.0", port=5000)
```

This starts the web server on port **5000**. The address **0.0.0.0** allows the application to accept connections from outside the EC2 instance.

5. Conclusion

Thus, **Amazon EC2** is an AWS cloud computing service that provides scalable and flexible virtual servers. It offers features such as **elastic computing, virtual instances, AMIs, scalability, security, flexible pricing, storage, and high availability**. A simple **Hello World Python application using Flask** was successfully created and executed on an Amazon EC2 instance step by step.

ChatGPT can make mistakes. Check important info.