EXPERIMENT NO:02

Title: Implementation and case study of Google App Engine

Solution:

Google App Engine (often referred to as GAE or simply App Engine) is a cloud computing platform as a service for developing and hosting web applications in Google- managed data centers. Applications are sandboxed and run across multiple servers. App Engine offers automatic scaling for web applications—as the number of requests increases for an application, App Engine automatically allocates more resources for the web application to handle the additional demand.

Google App Engine primarily supports Go, PHP, Java, Python, Node.js, .NET, and Ruby applications, although it can also support other languages via "custom runtimes". The service is free up to a certain level of consumed resources and only in standard environment but not in flexible environment.

Fees are charged for additional storage, bandwidth, or instance hours required by the application. It was first released as a preview version in April 2008 and came out of preview in September 2011.

Why Google App Engine is used?

Google allows you to add your web application code to the platform while managing the infrastructure for you. The engine ensures that your web apps are secure and running and saves them from malware and threats by enabling the firewall.

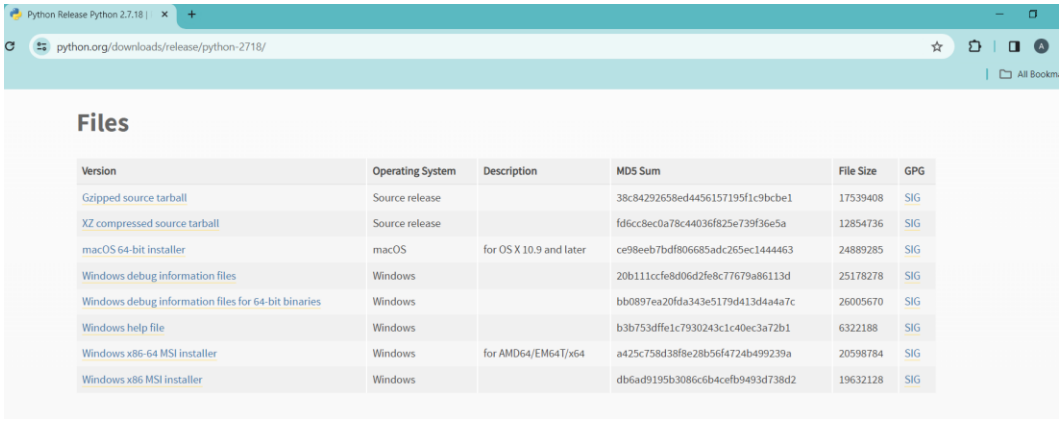
Feature of Google Cloud App Engine

1	**Popular Language**	Build your application in Node.js,Java,Ruby,C#,Python or PHP-or bring your own language runtime
2	**Open and flexible**	Custom runtimes allow you to bring any library and framework to App Engine by supplying a Docker container.
3	**Fully Managed**	A fully managed environment lets you focus on code while App Engine manages infrastructure concerns.
4	**Powerful application diagnostics**	Use Cloud Monitoring and Cloud Logging to monitor the health and performance of your app and Cloud Debugger and Error Reporting to diagnose and fix bugs quickly.

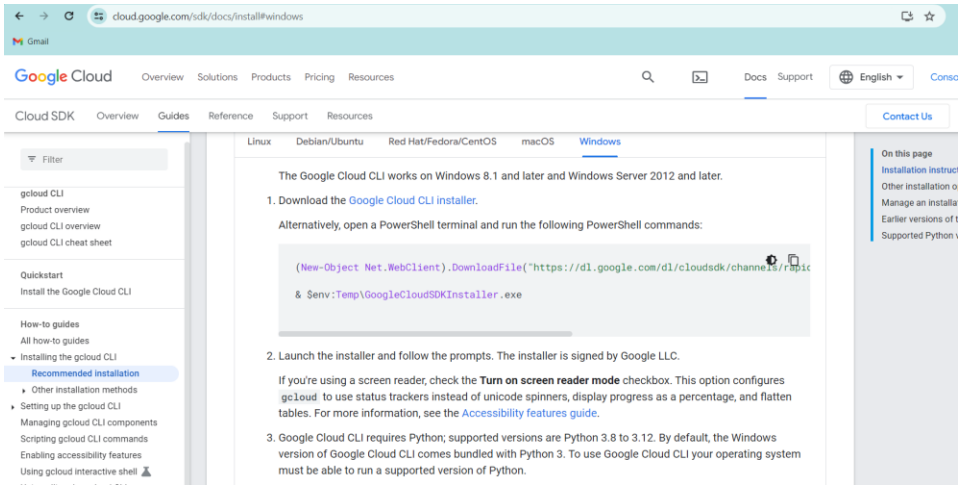
5	**Application Versioning**	Easily host different versions of your app, and easily create development, test, staging, and production environments.
6	**Traffic Splitting**	Route incoming requests to different app versions, A/B test, and do incremental feature rollouts.
7	**Application Security**	Help safeguard your application by defining access rules with App Engine firewall and leverage managed SSL/TLS certificates by default on your custom domain at no additional cost.
8	**Services Ecosystem**	Tap a growing ecosystem of Google Cloud services from your app including an excellent suite of cloud developer tools.

1. Installing Python and Google Cloud Sdk

i. Installing Python 2.7.18:
Go to link: https://www.python.org/downloads/release/python-2718/ and install the appropriate version.
After installation, go to environment variables>user variables>path><Enter your python file path>, according to the folder location.



ii. Installing Google Cloud Sdk:
   a. Go to link: https://cloud.google.com/sdk/docs/install and install the Google Cloud CLI installer.

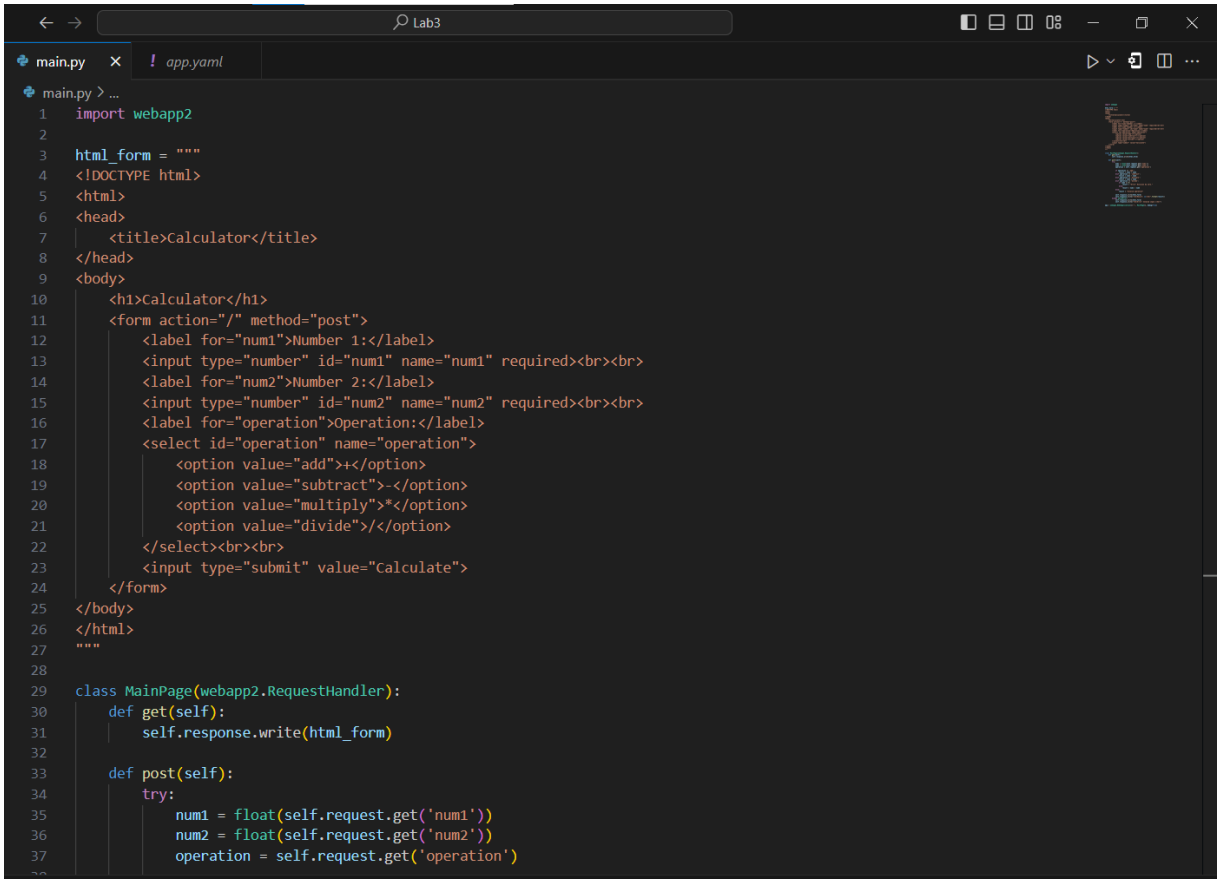


2. Writing the Program

Create a folder and create 2 files.

1. Main.py

Write the following program in main.py file. Make sure webapp2 is installed.



```python
class MainPage(webapp2.RequestHandler):
    def get(self):
        self.response.write(html_form)

    def post(self):
        try:
            num1 = float(self.request.get('num1'))
            num2 = float(self.request.get('num2'))
            operation = self.request.get('operation')

            if operation == 'add':
                result = num1 + num2
            elif operation == 'subtract':
                result = num1 - num2
            elif operation == 'multiply':
                result = num1 * num2
            elif operation == 'divide':
                if num2 == 0:
                    result = "Error! Division by zero."
                else:
                    result = num1 / num2
            else:
                result = "Invalid operation"

            self.response.write(html_form)
            self.response.write("<h2>Result: {}</h2>".format(result))
        except ValueError:
            self.response.write(html_form)
            self.response.write("<h2>Error! Invalid input.</h2>")

app = webapp2.WSGIApplication([('/', MainPage)], debug=True)
```

2. App.yaml

Write the following instructions in the app.yaml file.

```yaml
runtime: python27
api_version: 1
threadsafe: true

handlers:
- url: /
  script: main.app
```

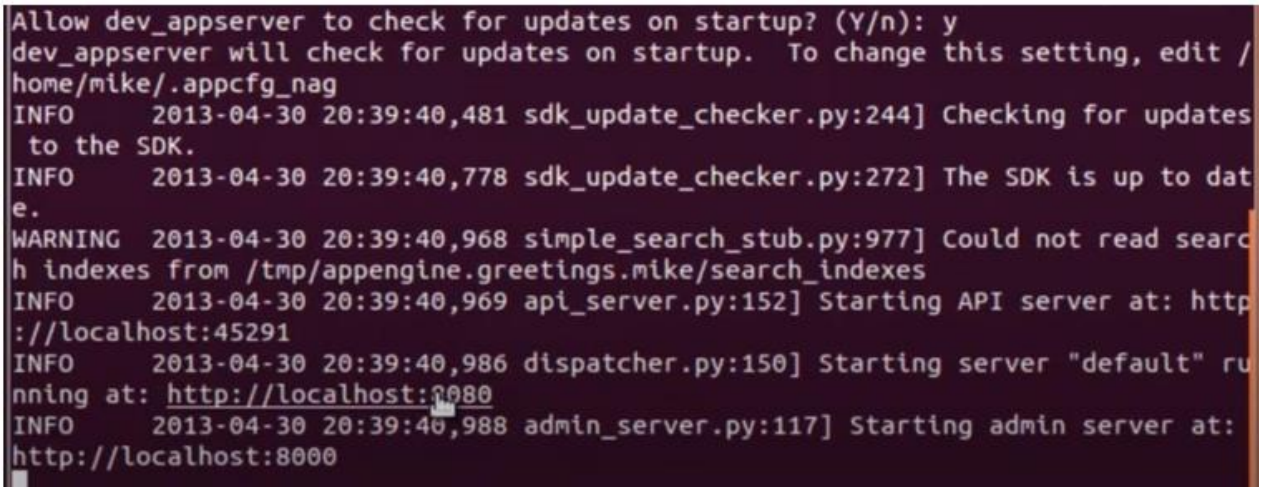
3. Running the Program

i. Check if the CLI recognizes the python version by entering: *python -V*

ii. To run the program, open Google Cloud SDK Shell and type the following command: *py google-cloud-sdk\bin\dev_appserver.py <Folder location where both programs are located>*

4. **Output**

This is the generated output window where link to the localhost is listed.



Go to browser and paste the localhost address to see the output.