EX.NO 7: Deploy python Web application using Google AppEngine

Date:

READING MATERIALS:

Google AppEngine is a PaaS implementation that provides services for developing and hosting scalable Web applications. AppEngine is essentially a distributed and scalable runtime environment that leverages Google's distributed infrastructure to scale out applications facing a large number of requests by allocating more computing resources to them and balancing the load among them. The runtime is completed by a collection of services that allow developers to design and implement applications that naturally scale on AppEngine. Developers can develop applications in Java, Python, and Go, a new programming language developed by Google to simplify the development of Web applications. Application usage of Google resources and services is metered by AppEngine, which bills users when their applications finish their free quotas.

Application services Applications hosted on AppEngine take the most from the services made available through the runtime environment. These services simplify most of the common operations that are performed in Web applications: access to data, account management, integration of external resources, messaging and communication, image manipulation, and asynchronous computation. UrlFetch Web 2.0 has introduced the concept of composite Web applications. Different resources are put together and organized as meshes within a single Web page. Meshes are fragments of HTML generated in different ways. They can be directly obtained from a remote server or rendered from an XML document retrieved from a Web service, or they can be rendered by the browser as the result of an embedded and remote component. A common characteristic of all these examples is the fact that the resource is not local to the server and often not even in the same administrative domain. Therefore, it is fundamental for Web applications to be able to retrieve remote resources.

The sandbox environment does not allow applications to open arbitrary connections through sockets, but it does provide developers with the capability of retrieving a remote resource through HTTP/HTTPS by means of the UrlFetch service. Applications can make synchronous and asynchronous Web requests and integrate the resources obtained in this way into the normal requesthandling cycle of the application. One of the interesting features of UrlFetch is the ability to set deadlines for requests so that they can be completed (or aborted) within a given time. Moreover, the ability to perform such requests asynchronously allows the applications to continue with their logic while the resource is retrieved in the background. UrlFetch is not only used to integrate meshes into a Web page but also to leverage remote Web services in accordance with the SOA reference model for distributed applications.

MemCache

AppEngine provides developers with access to fast and reliable storage, which is DataStore. Despite this, the main objective of the service is to serve as a scalable and long-term storage, where data are persisted to disk redundantly in order to ensure reliability and availability of data against failures. This design poses a limit on how much faster the store can be compared to other solutions, especially for objects that are frequently accessed—for example, at each Web request. AppEngine provides caching services by means of MemCache. This is a distributed in-memory cache that is optimized for fast access and provides developers with a volatile store for the objects that are frequently accessed. The caching algorithm implemented by MemCache will automatically remove the objects that are rarely accessed. The use of MemCache can significantly reduce the access time to data; developers can structure their applications so that each object is first looked up into MemCache and if there is a miss, it will be retrieved from DataStore and put into the cache for future lookups.

Mail and instant messaging

Communication is another important aspect of Web applications. It is common to use email for following up with users about operations performed by the application. Email can also be used to trigger activities in Web applications. To facilitate the implementation of such tasks, AppEngine provides developers with the ability to send and receive mails through Mail. The service allows sending email on behalf of the application to specific user accounts. It is also possible to include several types of attachments and to target multiple recipients. Mail operates asynchronously, and in case of failed delivery the sending address is notified through an email detailing the error. AppEngine provides also another way to communicate with the external world: the Extensible Messaging and Presence Protocol (XMPP). Any chat service that supports XMPP, such as Google Talk, can send and receive chat messages to and from the Web application, which is identified by its own address. Even though the chat is a communication medium mostly used for human interactions, XMPP can be conveniently used to connect the Web application with chat bots or to implement a small administrative console.

Account management

Web applications often keep various data that customize their interaction with users. These data normally go under the user profile and are attached to an account. AppEngine simplifies account management by allowing developers to leverage Google account management by means of Google Accounts. The integration with the service also allows Web applications to offload the implementation of authentication capabilities to Google's authentication system. Using Google Accounts, Web applications can conveniently store profile settings in the form of key-value pairs, attach them to a given Google account, and quickly retrieve them once the user authenticates. With respect to a custom solution, the use of Google Accounts requires users to have a Google account, but it does not require any further implementation. The use of Google Accounts is particularly advantageous for developing Web applications within a corporate environment using Google Apps. In this case, the applications can be easily integrated with all the other services (and profile settings) included in Google Apps.

Image manipulation Web applications render pages with graphics. Often simple operations, such as adding watermarks or applying simple filters, are required. AppEngine allows applications to perform image resizing, rotation, mirroring, and enhancement by means of Image Manipulation, a service that is also used in other Google products. Image Manipulation is mostly designed for lightweight image processing and is optimized for speed.

Compute services

Web applications are mostly designed to interface applications with users by means of a ubiquitous channel, that is, the Web. Most of the interaction is performed synchronously: Users navigate the Web pages and get instantaneous feedback in response to their actions. This feedback is often the result of some computation happening on the Web application, which implements the intended logic to serve the user request. Sometimes this approach is not applicable—for example, in long computations or when some operations need to be triggered at a given point in time. A good design for these scenarios provides the user with immediate feedback and a notification once the required operation is completed. AppEngine offers additional services such as Task Queues and Cron Jobs that simplify the execution of computations that are off-bandwidth or those that cannot be performed within the timeframe of the Web request.

EX.NO 7: Deploy python Web application using Google AppEngine

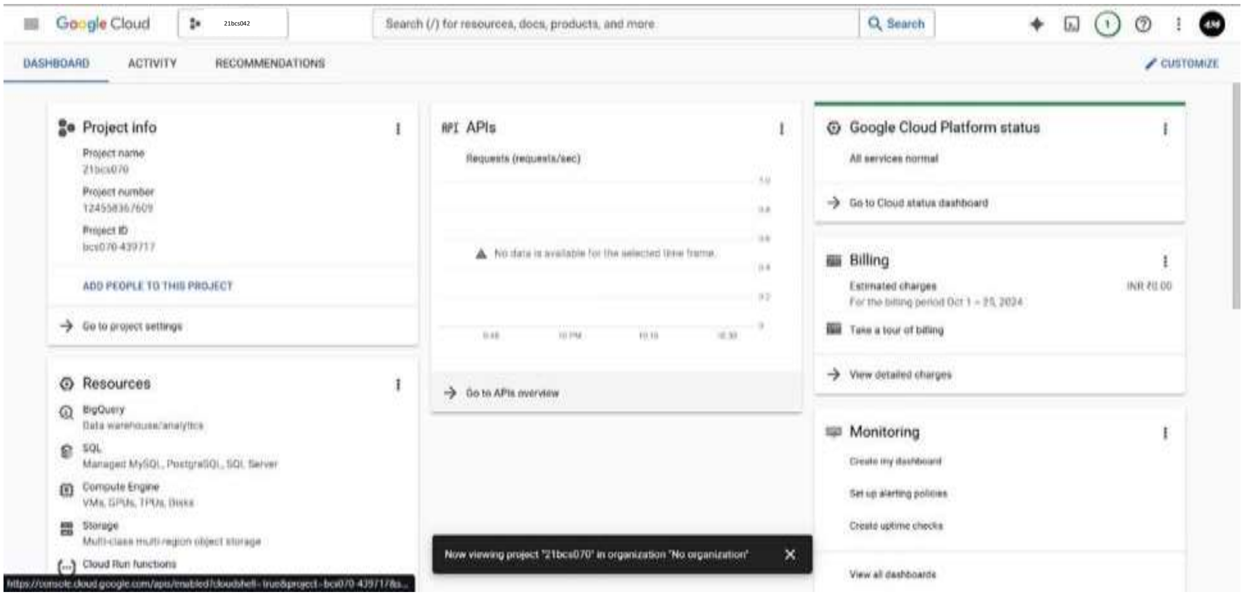
Date:

AIM:

To Deploy Python web application using Google AppEngine.

PROCEDURE with SCREENSHOTS:

Step 1: Setup Google Cloud Environment



Project name
21bcs070

Project number
124558367609

Project ID
bcs070-439717

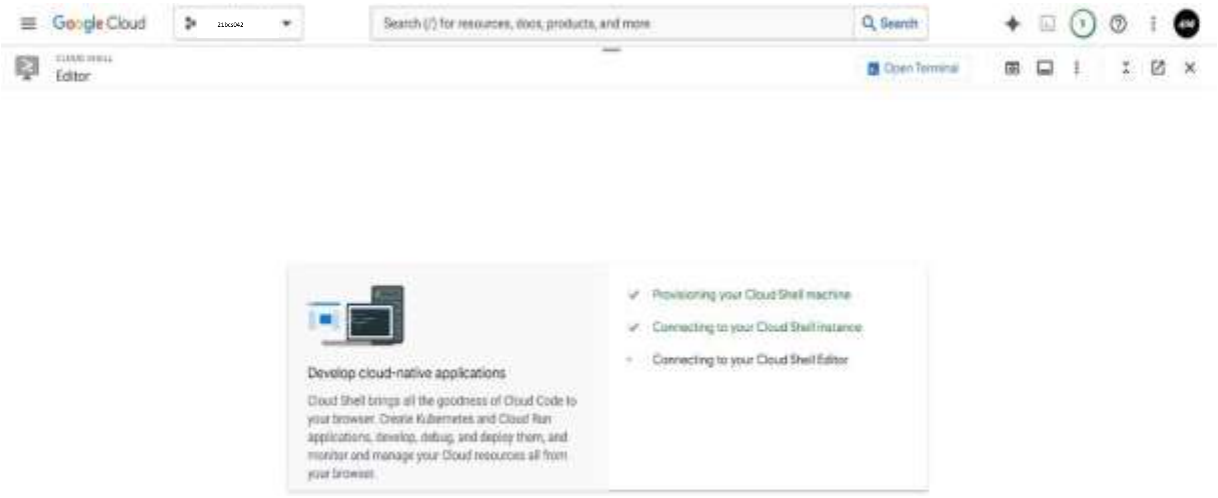
No data is available for the selected time frame.

All services normal

Estimated charges INR ₹0.00
For the billing period Oct 1 – 25, 2024

Now viewing project "21bcs070" in organization "No organization"

https://console.cloud.google.com/apis/enabled?cloudshell=true&project=bcs070-439717&s...

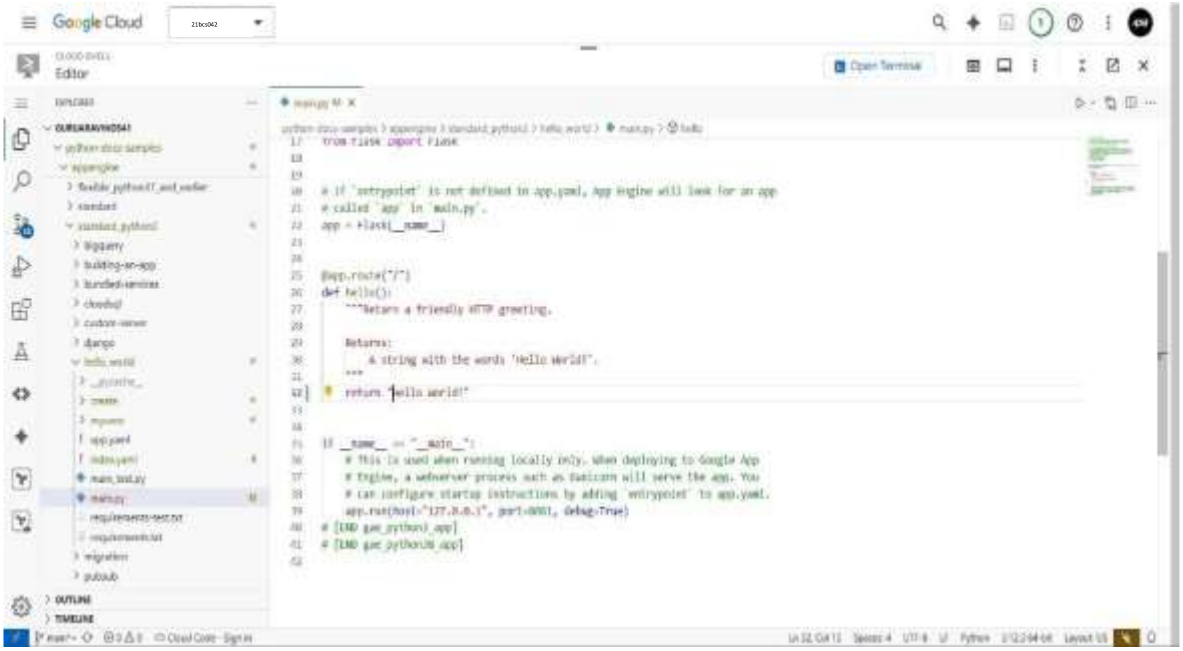


Develop cloud-native applications

Cloud Shell brings all the goodness of Cloud Code to your browser. Create Kubernetes and Cloud Run applications, develop, debug, and deploy them, and monitor and manage your Cloud resources all from your browser.

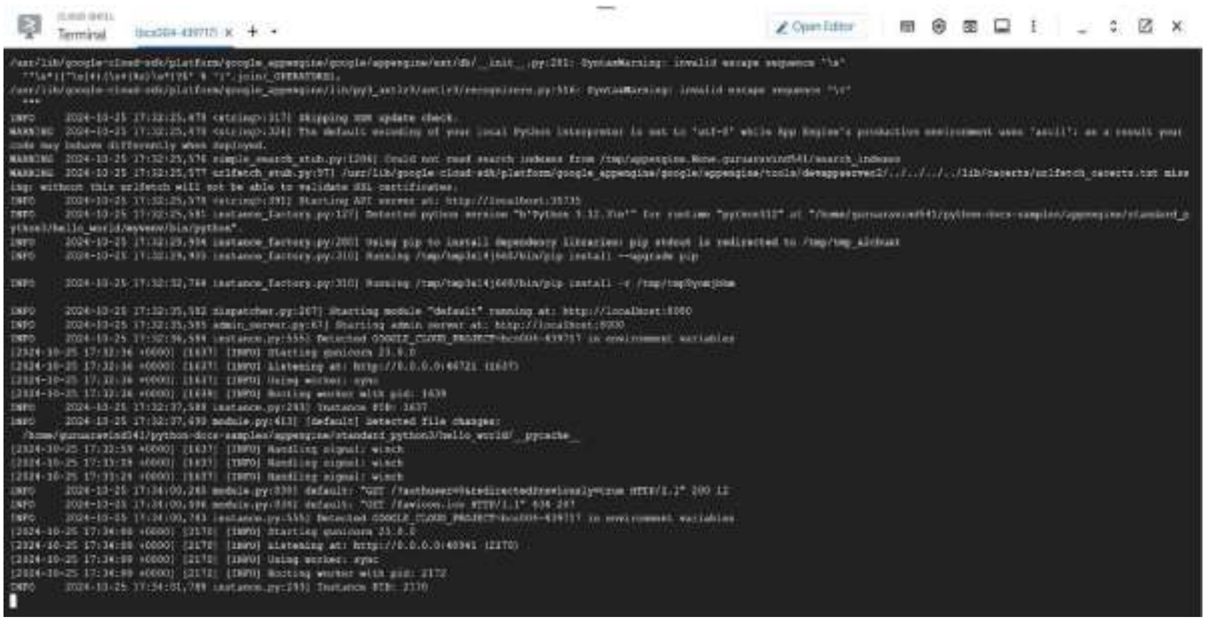
- ✓ Provisioning your Cloud Shell machine
- ✓ Connecting to your Cloud Shell instance
- Connecting to your Cloud Shell Editor

Step 2: Develop the Application



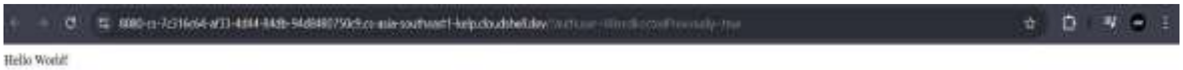
```
from flask import Flask


# If `entrypoint` is not defined in app.yaml, App Engine will look for an app
# called `app` in `main.py`.
app = Flask(__name__)


@app.route("/")
def hello():
    """Return a friendly HTTP greeting.

    Returns:
        A string with the words 'Hello World!'.
    """
    return "Hello World!"


if __name__ == "__main__":
    # This is used when running locally only. When deploying to Google App
    # Engine, a webserver process such as Gunicorn will serve the app. You
    # can configure startup instructions by adding `entrypoint` to app.yaml.
    app.run(host="127.0.0.1", port=8080, debug=True)
# [END gae_python3_app]
# [END gae_python38_app]
```

Step 3: Configure App Engine



```
/usr/lib/google-cloud-sdk/platform/google_appengine/google/appengine/ext/db/__init__.py:281: SyntaxWarning: invalid escape sequence '\s'
  '"\s*([^\s(]+)\s*(%s)\s*$' % '|'.join(_OPERATORS),
/usr/lib/google-cloud-sdk/platform/google_appengine/lib/py3_antlr3/antlr3/recognizers.py:516: SyntaxWarning: invalid escape sequence '\t'
  """
INFO     2024-10-25 17:32:25,478 <string>:317] Skipping SDK update check.
WARNING  2024-10-25 17:32:25,478 <string>:326] The default encoding of your local Python interpreter is set to 'utf-8' while App Engine's production environment uses 'ascii'; as a result your
code may behave differently when deployed.
WARNING  2024-10-25 17:32:25,576 simple_search_stub.py:1206] Could not read search indexes from /tmp/appengine.None.gurusaravind541/search_indexes
WARNING  2024-10-25 17:32:25,577 urlfetch_stub.py:97] /usr/lib/google-cloud-sdk/platform/google_appengine/google/appengine/tools/devappserver2/../../../../lib/cacerts/urlfetch_cacerts.txt miss
ing; without this urlfetch will not be able to validate SSL certificates.
INFO     2024-10-25 17:32:25,578 <string>:391] Starting API server at: http://localhost:35735
INFO     2024-10-25 17:32:25,581 instance_factory.py:127] Detected python version "b'Python 3.12.3\n'" for runtime "python312" at "/home/gurusaravind541/python-docs-samples/appengine/standard_p
ython3/hello_world/myvenv/bin/python".
INFO     2024-10-25 17:32:29,934 instance_factory.py:200] Using pip to install dependency libraries; pip stdout is redirected to /tmp/tmp_4lchuat
INFO     2024-10-25 17:32:29,935 instance_factory.py:310] Running /tmp/tmp3a14j668/bin/pip install --upgrade pip

INFO     2024-10-25 17:32:32,768 instance_factory.py:310] Running /tmp/tmp3a14j668/bin/pip install -r /tmp/tmp8yomjbbm

INFO     2024-10-25 17:32:35,582 dispatcher.py:267] Starting module "default" running at: http://localhost:8080
INFO     2024-10-25 17:32:35,585 admin_server.py:67] Starting admin server at: http://localhost:8000
INFO     2024-10-25 17:32:36,599 instance.py:555] Detected GOOGLE_CLOUD_PROJECT=bcs004-439717 in environment variables
[2024-10-25 17:32:36 +0000] [1637] [INFO] Starting gunicorn 23.0.0
[2024-10-25 17:32:36 +0000] [1637] [INFO] Listening at: http://0.0.0.0:46721 (1637)
[2024-10-25 17:32:36 +0000] [1637] [INFO] Using worker: sync
[2024-10-25 17:32:36 +0000] [1639] [INFO] Booting worker with pid: 1639
INFO     2024-10-25 17:32:37,588 instance.py:293] Instance PID: 1637
INFO     2024-10-25 17:32:37,693 module.py:413] [default] Detected file changes:
  /home/gurusaravind541/python-docs-samples/appengine/standard_python3/hello_world/__pycache__
[2024-10-25 17:32:59 +0000] [1637] [INFO] Handling signal: winch
[2024-10-25 17:33:19 +0000] [1637] [INFO] Handling signal: winch
[2024-10-25 17:33:24 +0000] [1637] [INFO] Handling signal: winch
INFO     2024-10-25 17:34:00,265 module.py:330] default: "GET /?authuser=0&redirectedPreviously=true HTTP/1.1" 200 12
INFO     2024-10-25 17:34:00,696 module.py:330] default: "GET /favicon.ico HTTP/1.1" 404 207
INFO     2024-10-25 17:34:00,783 instance.py:555] Detected GOOGLE_CLOUD_PROJECT=bcs004-439717 in environment variables
[2024-10-25 17:34:00 +0000] [2170] [INFO] Starting gunicorn 23.0.0
[2024-10-25 17:34:00 +0000] [2170] [INFO] Listening at: http://0.0.0.0:40341 (2170)
[2024-10-25 17:34:00 +0000] [2170] [INFO] Using worker: sync
[2024-10-25 17:34:00 +0000] [2172] [INFO] Booting worker with pid: 2172
INFO     2024-10-25 17:34:01,789 instance.py:293] Instance PID: 2170
█
```

Step 4: Deploy the Application



```
8080-cs-7c316e64-af33-4d44-84db-54d8480750c9.cs-asia-southeast1-kelp.cloudshell.dev/?authuser=0&redirectedPreviously=true
```

Hello World!

Evaluation by faculty	
Criteria	Marks
Preparation	/20
Program	/25
Output/Result	/20
Viva	/10
Total	/75
Faculty Signature with Date	

RESULT: