EX.NO 6: **Deploy Java 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.

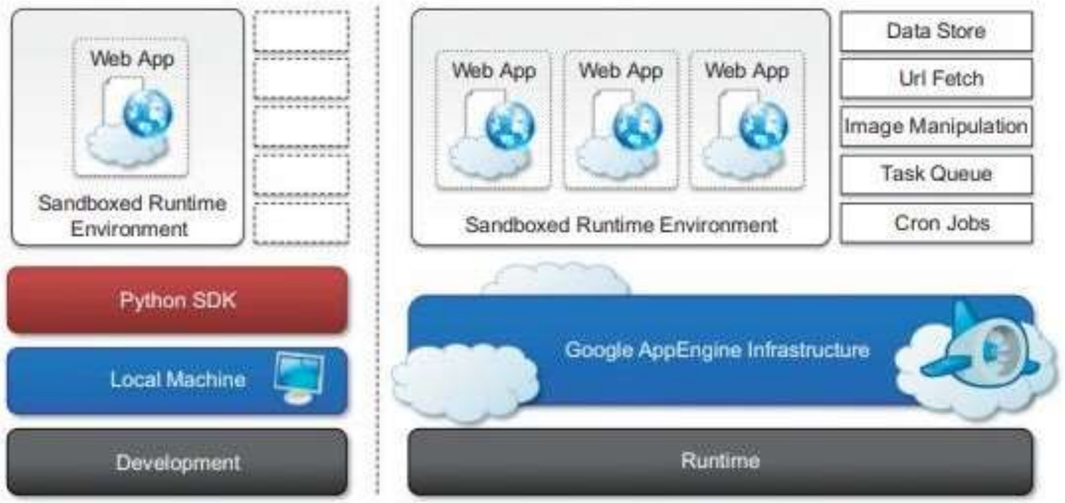
Architecture and core concepts

AppEngine is a platform for developing scalable applications accessible through the Web . The platform is logically divided into four major components: infrastructure, the runtime environment, the underlying storage, and the set of scalable services that can be used to develop applications.

Infrastructure AppEngine hosts Web applications, and its primary function is to serve users requests efficiently. To do so, AppEngine's infrastructure takes advantage of many servers available within Google datacenters. For each HTTP request, AppEngine locates the servers hosting the application that processes the request, evaluates their load, and, if necessary, allocates additional resources (i.e., servers) or redirects the request to an existing server. The particular design of applications, which does not expect any state information to be implicitly maintained between requests to the same application, simplifies the work of the infrastructure, which can redirect each of the requests to any of the servers hosting the target application or even allocate a new one. The infrastructure is also responsible for monitoring application performance and collecting statistics on which the billing is calculated.

Runtime environment

The runtime environment represents the execution context of applications hosted on AppEngine. With reference to the AppEngine infrastructure code, which is always active and running, the runtime comes into existence when the request handler starts executing and terminates once the handler has completed. Sandboxing One of the major responsibilities of the runtime environment is to provide the application environment with an isolated and protected context in which it can execute without causing a threat to the server and without being influenced by other applications. In other words, it provides applications with a sandbox. Currently, AppEngine supports applications that are developed only with managed or interpreted languages, which by design require a runtime for translating their code into executable instructions. Therefore, sandboxing is achieved by means of modified runtimes for applications that disable some of the common features normally available with their default implementations. If an application tries to perform any operation that is considered potentially harmful, an exception is thrown and the execution is interrupted. Some of the operations that are not allowed in the sandbox include writing to the server's file system; accessing computer through network besides using Mail, UrlFetch, and XMPP; executing code outside the scope of a request, a queued task, and a cron job; and processing a request for more than 30 seconds.



Supported runtimes Currently, it is possible to develop AppEngine applications using three different languages and related technologies: Java, Python, and Go. AppEngine currently supports Java 6, and developers can use the common tools for Web application development in Java, such as the Java Server Pages (JSP), and the applications interact with the environment by using the Java Servlet standard. Furthermore, access to AppEngine services is provided by means of Java libraries that expose specific interfaces of provider-specific implementations of a given abstraction layer. Developers can create applications with the AppEngine Java SDK, which allows developing applications with either Java 5 or Java 6 and by using any Java library that does not exceed the restrictions imposed by the sandbox. Support for Python is provided by an optimized Python 2.5.2 interpreter. As with Java, the runtime environment supports the Python standard library, but some of the modules that implement potentially harmful operations have been removed, and attempts to import such modules or to call specific methods generate exceptions. To support application development, AppEngine offers a rich set of libraries connecting applications to AppEngine services. In addition, developers can use a specific Python Web application framework, called webapp, simplifying the development of Web applications.

The Go runtime environment allows applications developed with the Go programming language to be hosted and executed in AppEngine. Currently the release of Go that is supported by AppEngine is r58.1. The SDK includes the compiler and the standard libraries for developing applications in Go and interfacing it with AppEngine services. As with the Python environment, some of the functionalities have been removed or generate a runtime exception. In addition, developers can include third-party libraries in their applications as long as they are implemented in pure Go.

Storage

AppEngine provides various types of storage, which operate differently depending on the volatility of the data. There are three different levels of storage: in memory-cache, storage for semistructured data, and long-term storage for static data. In this section, we describe DataStore and the use of static file servers. We cover MemCache in the application services section.

Static file servers Web applications are composed of dynamic and static data. Dynamic data are a result of the logic of the application and the interaction with the user. Static data often are mostly constituted of the components that define the graphical layout of the application (CSS files, plain HTML files, JavaScript files, images, icons, and sound files) or data files. These files can be hosted on static file servers, since they are not frequently modified. Such servers are optimized for serving static content, and users can specify how dynamic content should be served when uploading their applications to AppEngine.

DataStore

DataStore is a service that allows developers to store semistructured data. The service is designed to scale and optimized to quickly access data. DataStore can be considered as a large object database in which to store objects that can be retrieved by a specified key. Both the type of the key and the structure of the object can vary

EX.NO 6: Deploy Java web application using Google AppEngine

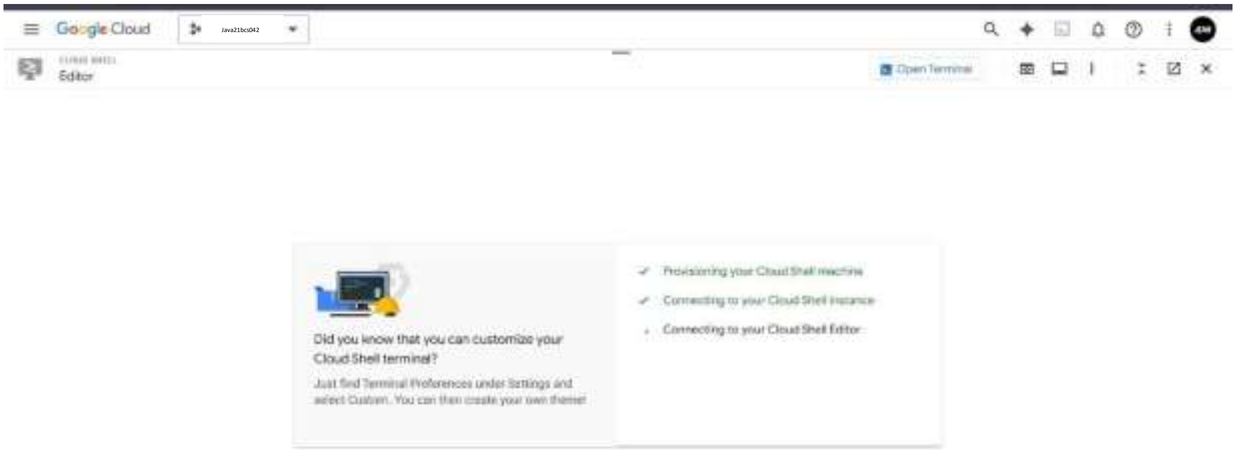
Date:

AIM:

To Deploy Java web application using Google AppEngine.

PROCEDURE with SCREENSHOTS:

Step 1: Set Up Google AppEngine SDK



✓ Provisioning your Cloud Shell machine

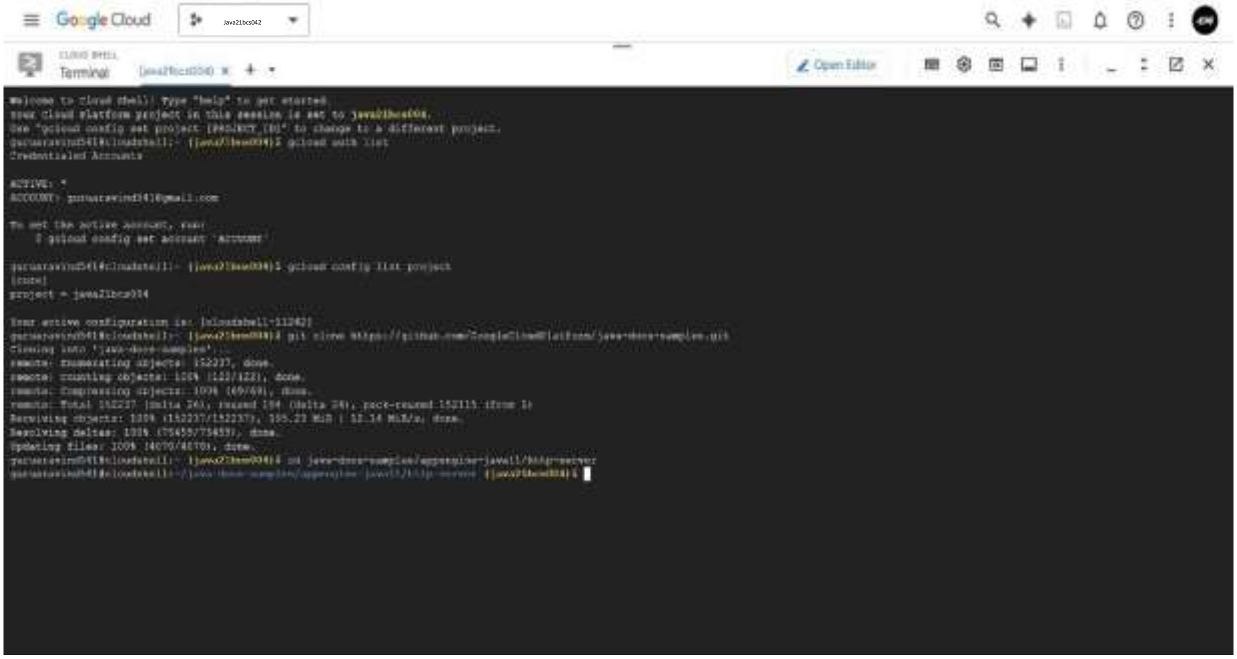
✓ Connecting to your Cloud Shell instance

Connecting to your Cloud Shell Editor

Did you know that you can customize your Cloud Shell terminal?

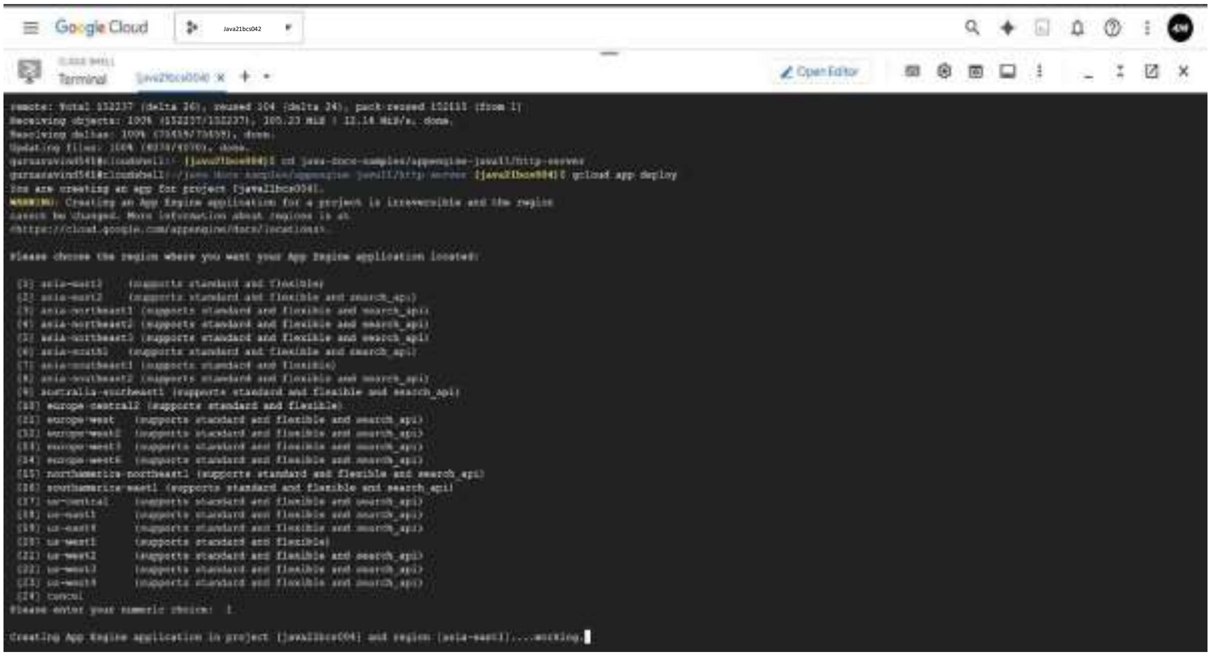
Just find Terminal Preferences under Settings and select Custom. You can then create your own theme!

Step 2: Prepare Java Web Application



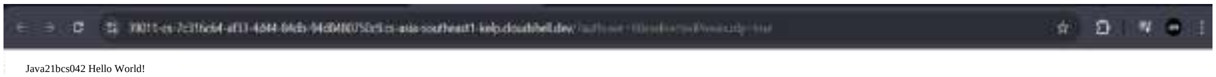
```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to java21bcs004.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
guruaravind541@cloudshell:~ (java21bcs004)$ gcloud auth list
Credentialed Accounts

ACTIVE: *
ACCOUNT: guruaravind541@gmail.com

To set the active account, run:
    $ gcloud config set account `ACCOUNT`

guruaravind541@cloudshell:~ (java21bcs004)$ gcloud config list project
[core]
project = java21bcs004

Your active configuration is: [cloudshell-11282]
guruaravind541@cloudshell:~ (java21bcs004)$ git clone https://github.com/GoogleCloudPlatform/java-docs-samples.git
Cloning into 'java-docs-samples'...
remote: Enumerating objects: 152237, done.
remote: Counting objects: 100% (122/122), done.
remote: Compressing objects: 100% (69/69), done.
remote: Total 152237 (delta 26), reused 104 (delta 24), pack-reused 152115 (from 1)
Receiving objects: 100% (152237/152237), 105.23 MiB | 12.14 MiB/s, done.
Resolving deltas: 100% (75459/75459), done.
Updating files: 100% (4070/4070), done.
guruaravind541@cloudshell:~ (java21bcs004)$ cd java-docs-samples/appengine-java11/http-server
guruaravind541@cloudshell:~/java-docs-samples/appengine-java11/http-server (java21bcs004)$
```

Step 3: Configure app.yaml File



```
remote: Total 152237 (delta 26), reused 104 (delta 24), pack-reused 152111 (from 1)
Receiving objects: 100% (152237/152237), 105.23 MiB | 12.14 MiB/s, done.
Resolving deltas: 100% (75459/75459), done.
Updating files: 100% (4070/4070), done.
guruaravind541@cloudshell:~ (java21bcs004)$ cd java-docs-samples/appengine-java11/http-server
guruaravind541@cloudshell:~/java-docs-samples/appengine-java11/http-server (java21bcs004)$ gcloud app deploy
You are creating an app for project [java21bcs004].
WARNING: Creating an App Engine application for a project is irreversible and the region
cannot be changed. More information about regions is at
<https://cloud.google.com/appengine/docs/locations>.

Please choose the region where you want your App Engine application located:

 [1] asia-east1    (supports standard and flexible)
 [2] asia-east2    (supports standard and flexible and search_api)
 [3] asia-northeast1 (supports standard and flexible and search_api)
 [4] asia-northeast2 (supports standard and flexible and search_api)
 [5] asia-northeast3 (supports standard and flexible and search_api)
 [6] asia-south1   (supports standard and flexible and search_api)
 [7] asia-southeast1 (supports standard and flexible)
 [8] asia-southeast2 (supports standard and flexible and search_api)
 [9] australia-southeast1 (supports standard and flexible and search_api)
 [10] europe-central2 (supports standard and flexible)
 [11] europe-west   (supports standard and flexible and search_api)
 [12] europe-west2  (supports standard and flexible and search_api)
 [13] europe-west3  (supports standard and flexible and search_api)
 [14] europe-west6  (supports standard and flexible and search_api)
 [15] northamerica-northeast1 (supports standard and flexible and search_api)
 [16] southamerica-east1 (supports standard and flexible and search_api)
 [17] us-central     (supports standard and flexible and search_api)
 [18] us-east1       (supports standard and flexible and search_api)
 [19] us-east4       (supports standard and flexible and search_api)
 [20] us-west1       (supports standard and flexible)
 [21] us-west2       (supports standard and flexible and search_api)
 [22] us-west3       (supports standard and flexible and search_api)
 [23] us-west4       (supports standard and flexible and search_api)
 [24] cancel
Please enter your numeric choice:  1

Creating App Engine application in project [java21bcs004] and region [asia-east1]....working.
```

Step 4: Deploy the Application



8081-cs-7c31fa6d-af33-4d44-84db-94d8480750e5.cs-asia-southeast1-kelp.cloudshell.dev/[illegible]

Java21bcs042 Hello World!

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

RESULT: