Step-by-Step Guide to Creating and Deploying a New Spring Boot Application to Google App Engine

Prerequisites:

1. **Google Cloud CLI:**
    - Download and install the Google Cloud CLI for your operating system. Follow the instructions on the official page: Google Cloud CLI Download.
2. **GCP Credit Application:**
    - Ensure you have applied for Google Cloud Platform (GCP) credits with the instructions mentioned on Ed Discussion.
3. **Billing Information:**
    - Set up your billing information in your Google Cloud account. Even with credits, GCP requires billing information to deploy applications. Follow the GCP Setup Guide on class website – points 1-4.

Process:

Step 1: Create a Google Cloud Project

1. **Navigate to the Google Cloud Console:**
    - Log in with your Google account.
2. **Create a New Project:**
    - Click on the "Select a Project" drop-down and then "New Project."
    - Name your project (e.g., "SpringBootDemo") and choose a billing account.
    - Click "Create" to set up your new project.

Step 2: Create a New Spring Boot Application

1. **Go to Spring Initializr:**
    - Spring Initializr is a web-based tool for generating new Spring Boot projects.
2. **Configure Your Project:**
    - **Project:** Select **Maven**.
    - **Packaging:** Choose **JAR**.
    - **Java Version:** Select **Java 17/21/22** or a compatible version.
    - **Group:** Enter a group ID (e.g., `com.example`).
    - **Artifact:** Enter an artifact ID (e.g., `springbootdemo`).
    - **Dependencies:** Add **Spring Web** (needed for creating REST endpoints).
    - Click **Generate** to create and download your project as a ZIP file.
3. **Unzip the Project:**
    - Unzip the downloaded file to a folder on your computer.
    - Open the project in your preferred development environment (e.g., Visual Studio Code, IntelliJ, or Eclipse).

Step 3: Create a HelloWorld Controller

1. **Navigate to the Main Java Package:**
   - Go to `src/main/java/com/example/springbootdemo` (replace `com/example/springbootdemo` with your package structure).
2. **Create a New Java Class:**
   - Name the class `HelloWorldController.java`.
3. **Add the Following Code to the Class:**

```
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HelloWorldController {

    @GetMapping("/hello")
    public String hello() {
        return "Hello, World!";
    }
}
```

   - **Explangation:**
     - `@RestController` marks the class as a RESTful web service controller.
     - `@GetMapping("/hello")` maps HTTP GET requests to the `/hello` endpoint.
     - When accessed, this endpoint will return the text `"Hello, World!"`.

Step 4: Update the `pom.xml` File

1. **Open the `pom.xml` File:**
   - Locate and open the `pom.xml` file in the root directory of your project.
2. **Add the Google Cloud Tools Plugin:**
   - Add the following plugin configuration inside the `<build>` section:

```
<build>
    <plugins>
        <plugin>
            <groupId>com.google.cloud.tools</groupId>
            <artifactId>appengine-maven-plugin</artifactId>
            <version>2.3.0</version>
        </plugin>
    </plugins>
</build>
```

   - **Explanation:**
     - This plugin allows you to deploy the application to Google App Engine directly from Maven.
3. **Save the `pom.xml` File:**
   - Save your changes and close the file.

Step 5: Create the `app.yaml` File

1. **Go to the Root Directory of Your Project:**
   - Make sure you are in the root directory of your Spring Boot project (where your `pom.xml` file is located).
2. **Create a New File Named `app.yaml`:**
   - In your code editor, create a new file named `app.yaml`.
3. **Add the Following Content to the `app.yaml` File:**

```
runtime: java17
env: standard
service: default
```

   - **Explanation:**
     - `runtime: java`: Specifies the Java runtime.
     - `env: standard`: Uses the standard App Engine environment.
     - `service: default`: Defines the default service name to handle all traffic.
4. **Save the `app.yaml` File:**
   - Ensure the `app.yaml` file is saved in the root directory of your project.

Step 6: Build Your Spring Boot Application

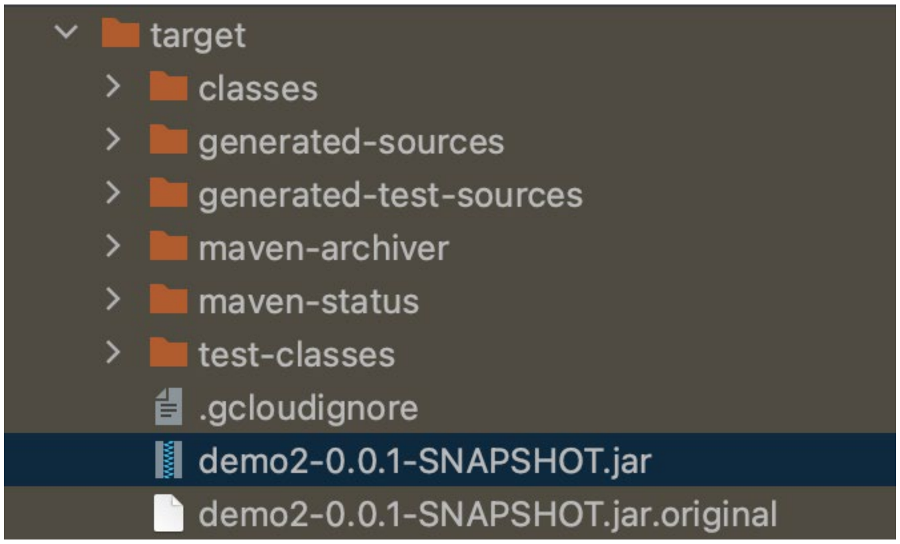
1. **Open a Terminal in the Root Directory of Your Project:**
   - You can use the integrated terminal in your development environment or a system terminal.
2. **Run the Maven Build Command:**
   - For Linux or macOS:

```
./mvnw clean install -DskipTests
```

   - For Windows:

```
.\mvnw.cmd clean install -DskipTests
```

   - This command compiles your project and packages it into a JAR file located in the `target` directory.



```
target
  classes
  generated-sources
  generated-test-sources
  maven-archiver
  maven-status
  test-classes
  .gcloudignore
  demo2-0.0.1-SNAPSHOT.jar
  demo2-0.0.1-SNAPSHOT.jar.original
```

Step 7: Deploy to Google App Engine

1. **Initialize Google Cloud CLI:**
   - Open your terminal and check if the Google Cloud CLI is installed:

```
gcloud -v
```

   - Initialize the CLI with:

```
gcloud init
```

   - Follow the prompts to:
     - Choose the default configuration.
     - Log in with your Google account (ensure it has billing set up).
     - Select the project you created in Step 1.
2. **Deploy Your Application:**
   - Deploy your application to App Engine using the following command at the root level:

```
gcloud app deploy
```

   - This command uses the `app.yaml` file to determine the configuration for the deployment.
3. **Follow the Prompts:**
   - Choose a deployment region (e.g., `us-east1`).
   - Wait for the deployment to complete. A URL will be displayed where your application is hosted.

Step 8: Verify Deployment

1. **Open Your Application:**

- Run the following command to open your deployed application in the browser:

```
gcloud app browse
```

- This command will take you to the URL of your deployed application.

Step 9: Clean Up to Avoid Billing Charges

1. **Shut Down the GCP Project If Not Needed:**
    - If you do not need the deployed application anymore or want to avoid billing charges (which you should)
    - Go to the Google Cloud Console.
    - Navigate to **Project Settings** and select **Shut Down Project**.