

JAVAMS09 Deploying to App Engine

Overview

In this series of labs, you take a demo microservices Java application built with the Spring framework and modify it to use an external database server. You adopt some of the best practices for tracing, configuration management, and integration with other services using integration patterns.

In all of the labs so far, you tested the application by running the application in Cloud Shell. Many options are available to deploy your application on Google Cloud. For example, you can deploy the application to virtual machines that you configure yourself. Or you can containerize your application and deploy it into a managed Google Kubernetes Engine cluster. You can also run your favorite platform as a service on Google Cloud (for example, Cloud Foundry and OpenShift).

App Engine is Google's fully managed serverless application platform. With App Engine, you can build and deploy applications on a fully managed platform and scale your applications without having to worry about managing the underlying infrastructure. App Engine includes capabilities such as automatic scaling-up and scaling-down of your application, and fully managed patching and management of your servers.

In this lab, you deploy the application into App Engine. You need to convert the application from fat WARs into the thin-WAR deployments that App Engine can deploy.

Objectives

In this lab, you learn how to perform the following tasks:

- Initialize App Engine
- Reconfigure an application to work with App Engine
- Configure the Cloud Runtime Configuration API to automatically provide the backend URL to the Frontend application
- Deploy the application to App Engine


Setup and requirements


How to start your lab and sign in to the Console


1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is a panel populated with the temporary credentials that you must use for this lab.

[Open Google Console](#)

Caution: When you are in the console, do not deviate from the lab instructions. Doing so may cause your account to be blocked. [Learn more.](#)

Username
google2727032_student@qwiklabs.n 

Password
k68CZXsxMZ 

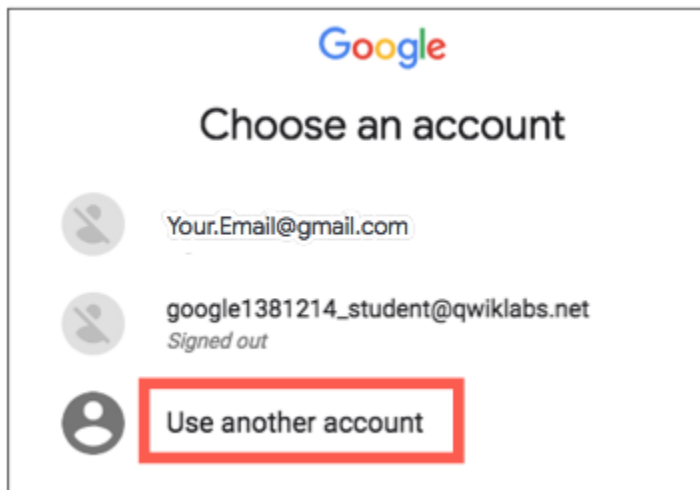
GCP Project ID
qwiklabs-gcp-4fbfecac8667e457 

[New to labs? View our introductory video!](#)

2. Copy the username, and then click **Open Google Console**. The lab spins up resources, and then opens another tab that shows the **Choose an account** page.

Note: Open the tabs in separate windows, side-by-side.

3. On the Choose an account page, click **Use Another Account**. The Sign in page opens.



4. Paste the username that you copied from the Connection Details panel. Then copy and paste the password.

Note: You must use the credentials from the Connection Details panel. Do not use your Google Cloud Skills Boost credentials. If you have your own Google Cloud account, do not use it for this lab (avoids incurring charges).

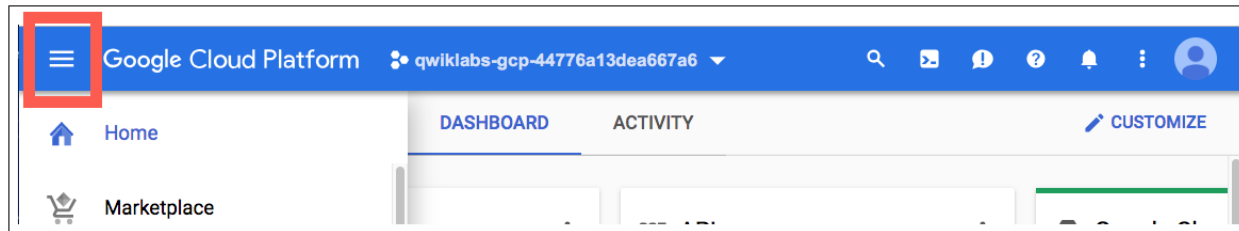
5. Click through the subsequent pages:

- Accept the terms and conditions.
- Do not add recovery options or two-factor authentication (because this is a temporary account).
- Do not sign up for free trials.

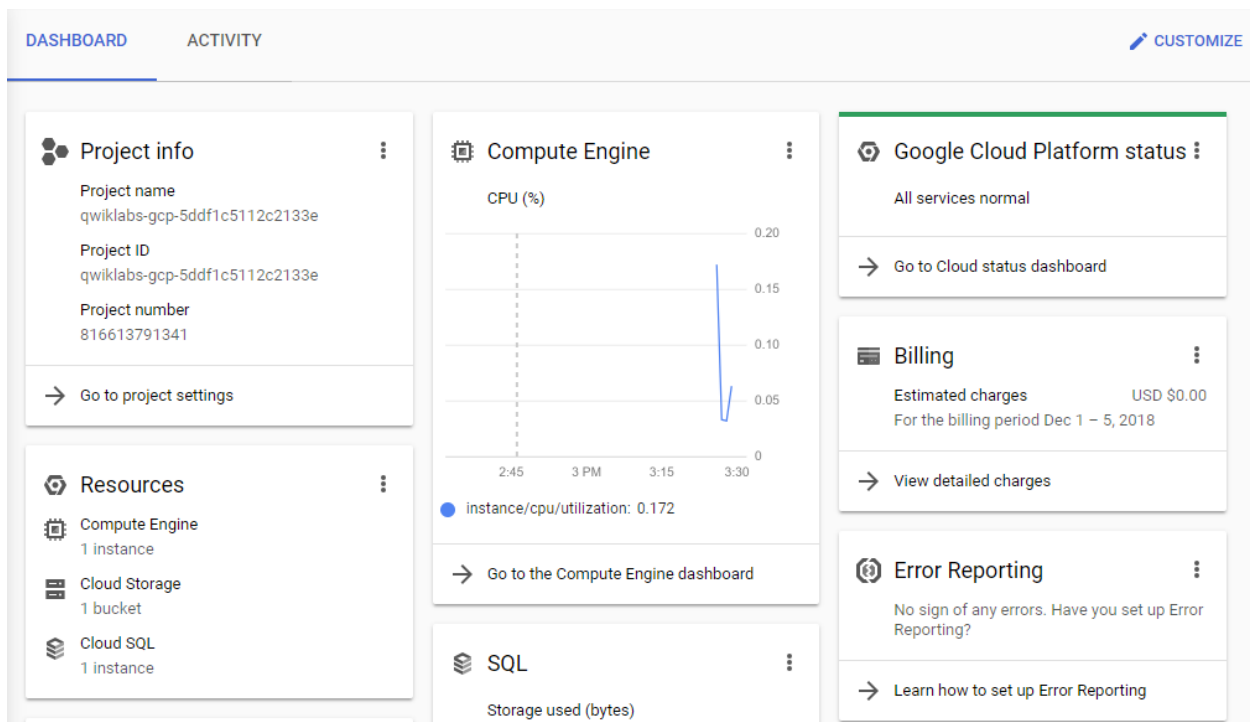
After a few moments, the Cloud console opens in this tab.

Note: You can view the menu with a list of Google Cloud Products and Services by clicking the **Navigation menu** at the top-

left.



After you complete the initial sign-in steps, the project dashboard appears.



Task 1. Fetch the application source files

In this task you clone the source repository files that are used throughout this lab.

1. To begin the lab, click the **Activate Cloud Shell** button at the top of the Google Cloud Console.
2. To activate the code editor, click the `Open Editor` button on the toolbar of the Cloud Shell window. You can switch between Cloud Shell and the code editor by using `Open Editor` and `Open Terminal` icon as required.

Note: A Cloud Storage bucket that is named using the project ID for this lab is automatically created for you by the lab setup. The source code for your applications is copied into this bucket when the Cloud SQL server is ready. You might have to wait a few minutes for this action to complete.

3. In Cloud Shell, enter the following command to create an environment variable that contains the project ID for this lab:

```
export PROJECT_ID=$(gcloud config list --format 'value(core.project)')
```

Copied!

content_copy

4. Verify that the demo application files were created:

```
gsutil ls gs://$PROJECT_ID
```

Copied!

content_copy

5. Copy the application folders to Cloud Shell:

```
gsutil -m cp -r gs://$PROJECT_ID/* ~/
```

Copied!

content_copy

6. Make the Maven wrapper scripts executable:

```
chmod +x ~/guestbook-frontend/mvnw
```

```
chmod +x ~/guestbook-service/mvnw
```

Copied!

content_copy

Now you're ready to go!

Task 2. Initialize App Engine

In this task, you initialize App Engine by enabling it in the project.

- Enable App Engine in the project:

```
gcloud app create --region=us-central
```

Copied!

content_copy

The message `The app is now created` indicates that App Engine is enabled.

Note: App Engine deployments are regional. That is, your application might be deployed to multiple zones within a region. Because the Cloud SQL instance is in `us-central1`, you should deploy the application to the `us-central` region for App Engine.

Task 3. Deploy Guestbook Frontend

In this task, you add the App Engine plugin to the guestbook frontend's `pom.xml` file.

1. In the Cloud Shell code editor, open `~/guestbook-frontend/pom.xml`.
2. Add the following code at the end of the `<plugin>` section, immediately before the closing `</plugins>` tag:

```
<plugin>
  <groupId>com.google.cloud.tools</groupId>
  <artifactId>appengine-maven-plugin</artifactId>
  <version>2.2.0</version>
  <configuration>
    <version>1</version>
    <deploy.projectId>GLOUD_CONFIG</deploy.projectId>
  </configuration>
</plugin>
```

Copied!

content_copy

3. In Cloud Shell, create an App Engine directory in Guestbook Frontend:

```
mkdir -p ~/guestbook-frontend/src/main/appengine
```

Copied!

content_copy

4. In the Cloud Shell code editor, **create** a file named `app.yaml` in the `~/guestbook-frontend/src/main/appengine/` directory. This file is required to deploy the application to App Engine.
5. Add the following code to the `app.yaml` file. Make sure to **replace** `PROJECT_ID` with your Project ID. You can use the command `echo $PROJECT_ID` in the Cloud Shell to retrieve it:

```
runtime: javall
instance_class: B4_1G
manual_scaling:
  instances: 2
env_variables:
  SPRING_PROFILES_ACTIVE: cloud
  # REPLACE PROJECT_ID with your project ID!
  MESSAGES_ENDPOINT: https://guestbook-service-dot-
PROJECT_ID.appspot.com/guestbookMessages
```

Copied!

content_copy

6. In Cloud Shell, change to the frontend application directory:

```
cd ~/guestbook-frontend
```

Copied!

content_copy

7. Use Apache Maven to deploy the frontend application to App Engine. This should take a few minutes:

```
mvn package appengine:deploy -DskipTests
```

Copied!

content_copy

Note: You may need to run the command twice if there is a build error! This command reports out success like the following example:

```
[INFO] GCloud: Deployed service [default] to
[https://PROJECT_ID.appspot.com]
[INFO] GCloud:
[INFO] GCloud: You can stream logs from the command line by running:
[INFO] GCloud:   $ gcloud app logs tail -s default
[INFO] GCloud:
[INFO] GCloud: To view your application in the web browser run:
[INFO] GCloud:   $ gcloud app browse
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 03:00 min
[INFO] Finished at: 2018-12-08T10:30:09Z
[INFO] -----
```

8. Find the frontend URL:

```
gcloud app browse
```

Copied!

content_copy

This command reports a URL that links to your application's frontend:

```
Did not detect your browser. Go to this link to view your app:  
https://....appspot.com
```

9. Click the link to open a browser tab to the frontend URL.

An error occurs because the backend isn't deployed yet:

Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Sat Jul 14 20:55:36 UTC 2018

There was an unexpected error (type=Internal Server Error, status=500).

status 404 reading GuestbookMessagesClient#getMessage(); content: {"timestamp":1531601736455,"status":404,"error":"Not Found","message":"No message available","path":"/guestbookMessages/"}

Leave the tab open. You'll come back to this later.

Task 4. Deploy Guestbook Service

In this task, you will add the App Engine Plugin to guestbook-service's `pom.xml`.

1. In the `guestbook-service/pom.xml` file, add the following code at the end of the `<plugin>` section, immediately before the closing `</plugins>` tag:

```
<plugin>  
  <groupId>com.google.cloud.tools</groupId>  
  <artifactId>appengine-maven-plugin</artifactId>  
  <version>2.2.0</version>  
  <configuration>  
    <version>1</version>  
    <deploy.projectId>GCLLOUD_CONFIG</deploy.projectId>  
  </configuration>  
</plugin>
```

Copied!

content_copy

2. Create an App Engine directory in Guestbook Service:


```
mkdir -p ~/guestbook-service/src/main/appengine
```

Copied!

content_copy

3. In the Cloud Shell code editor, **create** a file named `app.yaml` in the `~/guestbook-service/src/main/appengine/` directory. This file is required to deploy the application to App Engine.

4. Add the following code to the `app.yaml` file:

```
runtime: javall
service: guestbook-service
instance_class: B4_1G
manual_scaling:
  instances: 2
env_variables:
  SPRING_PROFILES_ACTIVE: cloud
```

Copied!

content_copy

Note: This configuration uses manual scaling rather than automatic scaling. This is great if you want to have fine control over the number of application instances. However, in a production setting, you may want to use automatic scaling that can adapt dynamically to the load.

5. In Cloud Shell, first change to your `guestbook-service` directory, then use Maven to deploy the application. This will take a few minutes:

```
cd ~/guestbook-service
./mvnw package appengine:deploy -DskipTests
```

Copied!

content_copy

6. Find the deployed backend URL. Click the URL link for the backend guestbook service:

```
gcloud app browse -s guestbook-service
```

Copied!

content_copy

Links similar to those in the screenshot are displayed, You can use these URLs to list and inspect the contents of messages that have been posted to the application:

```
{
  "_links" : {
    "guestbookMessages" : {
      "href" : "http://guestbook-service-dot-qwiklabs-gcp-02-dccb3a0cb46d.uc.r.appspot.com/guestbookMessages?page,size,sort",
      "templated" : true
    },
    "profile" : {
      "href" : "http://guestbook-service-dot-qwiklabs-gcp-02-dccb3a0cb46d.uc.r.appspot.com/profile"
    }
  }
}
```

7. Follow the href links to see all of the messages, e.g.: <https://guestbook-service-dot-PROJECT.appspot.com/guestbookMessages>, and see the sample message that is preloaded by the lab setup process:

```
{
  "_embedded" : {
    "guestbookMessages" : [ {
      "name" : "Ray",
      "message" : "Hello Cloud SQL",
      "imageUri" : null,
      "_links" : {
        "self" : {
          "href" : "http://guestbook-service-dot-qwiklabs-gcp-02-dccb3a0cb46d.uc.r.appspot.com/guestbookMessages/1"
        },
        "guestbookMessage" : {
          "href" : "http://guestbook-service-dot-qwiklabs-gcp-02-dccb3a0cb46d.uc.r.appspot.com/guestbookMessages/1"
        }
      }
    } ]
  },
  "_links" : {
    "self" : {
      "href" : "http://guestbook-service-dot-qwiklabs-gcp-02-dccb3a0cb46d.uc.r.appspot.com/guestbookMessages?page,size,sort",
      "templated" : true
    },
    "profile" : {
      "href" : "http://guestbook-service-dot-qwiklabs-gcp-02-dccb3a0cb46d.uc.r.appspot.com/profile/guestbookMessages"
    }
  },
  "page" : {
    "size" : 20,
    "totalElements" : 1,
    "totalPages" : 1,
    "number" : 0
  }
}
```

8. Lastly, switch to the tab showing the error generated by the frontend application and refresh the page. You should now see the correct user message posting interface that you are familiar with from earlier labs and the initial sample message that is preloaded by the lab setup process. Instead of running in Cloud Shell, the application is now running as two separate services on App Engine!

Guestbook

Your name:

Message:

File:

No file chosen

Ray

Hello Cloud SQL

Task 5. Review

In this lab you initialized App Engine and reconfigured an application to work with App Engine. You also configured the Cloud Runtime Configuration API to automatically provide the backend URL to the Frontend application. Finally, you deployed the application to App Engine