

JAVAMS09 Deploying to App Engine

2 hoursFree

Rate Lab

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 Platform (GCP). 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 Kubernetes cluster. You can also run your favorite platform as a service on GCP (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:

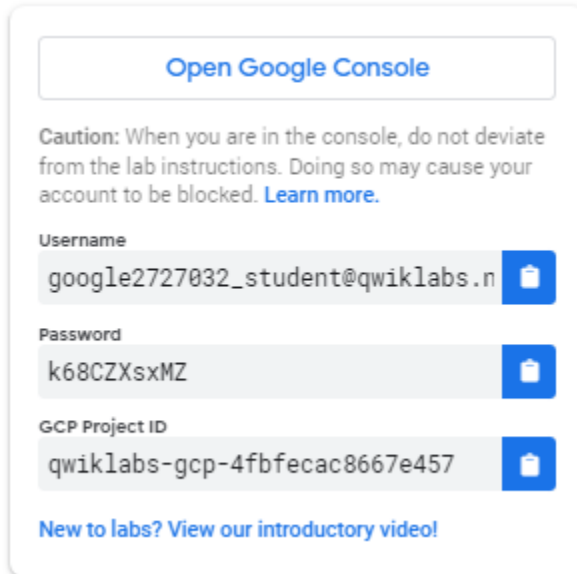
- Initialize App Engine
- Reconfigure an application to work with App Engine
- Configure the Cloud Runtime Runtime Configuration API to automatically provide the backend URL to the Frontend application
- Deploy the application to App Engine

Task 0. Lab Preparation

Access Qwiklabs


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 

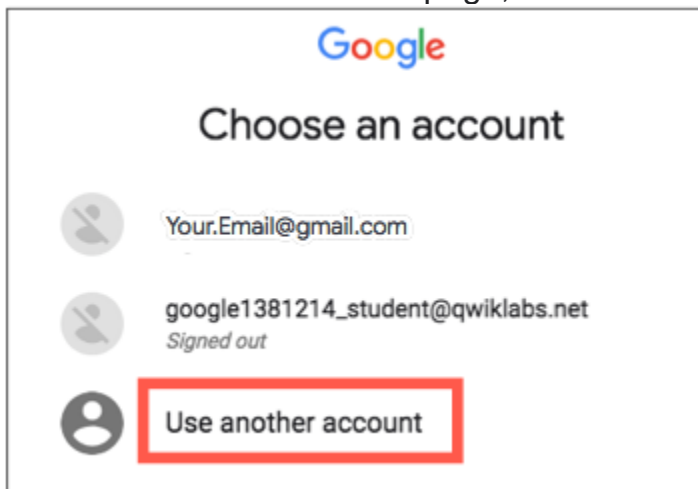
GCP Project ID
qwiklabs-gcp-4fbfecac8667e457 

[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.


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


3. On the Choose an account page, click **Use Another Account**.




Google

Choose an account

 Your.Email@gmail.com

 google1381214_student@qwiklabs.net
Signed out

 **Use another account**

4. The Sign in page opens. Paste the username that you copied from the Connection Details panel. Then copy and paste the password.

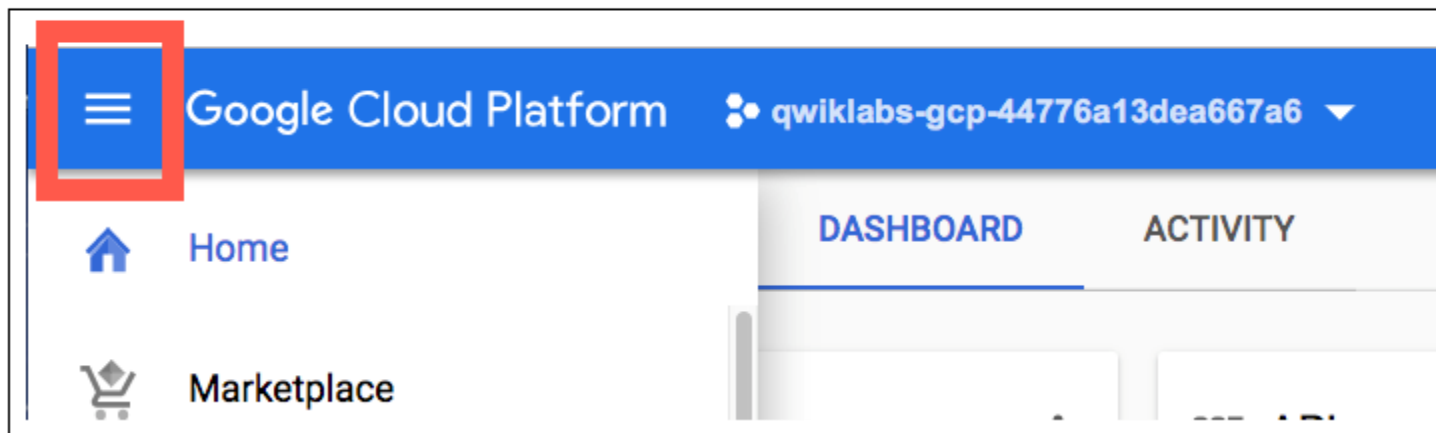
Important: You must use the credentials from the Connection Details panel. Do not use your Qwiklabs credentials. If you have your own GCP account, do not use it for this lab (avoids incurring charges).

5. Click through the subsequent pages:

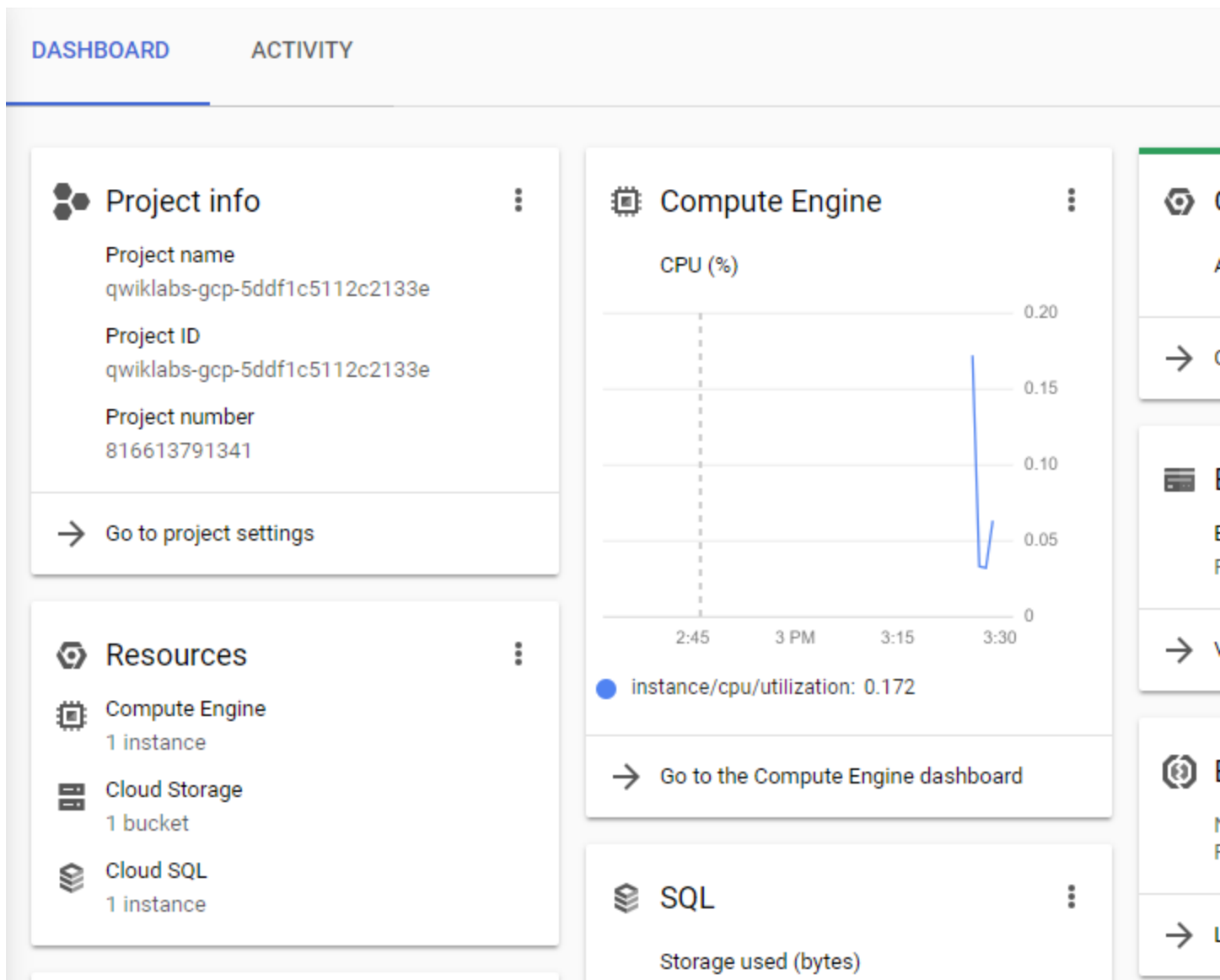
- Accept the terms and conditions.
- Do not add recovery options or two-factor authentication (because this is a temporary account).
- Do not sign up for free trials.

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

Note: You can view the menu with a list of GCP Products and Services by clicking the **Navigation menu** at the top-left, next to “Google Cloud Platform”.



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



Fetch the application source files

The lab setup includes automated deployment of the services that you configured yourself in previous labs. When the setup is complete, copies of the demo application (configured so that they are ready for this lab session) are put into a Cloud Storage bucket named using the project ID for this lab.

Before you proceed with the tasks for this lab, you must first copy the demo application into Cloud Shell so you can continue to work on it.

1. In the upper-right corner of the screen, click **Activate Cloud**



Shell () to open Cloud Shell.

2. Click **Start Cloud Shell**.

Boost mode is not needed for this lab.

3. In the Cloud Shell command line, 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)')
```

4. Verify that the demo application files were created.

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

Repeat the last step if the command reports an error or if it does not list the two folders for the `guestbook-frontend` application and the `guestbook-service` backend application.

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.

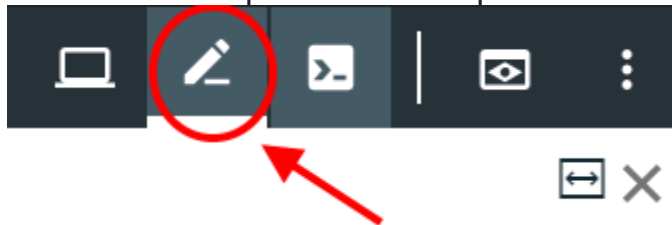
5. Copy the application folders to Cloud Shell.

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

6. Make the Maven wrapper scripts executable.

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

7. Click the pencil icon to open the Cloud Shell code editor.



Task 1. Initialize App Engine

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

1. Enable App Engine in the project.

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

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 2. Make the guestbook frontend App Engine friendly

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 `<plugins>` section, immediately before the closing `</plugins>` tag:

```
<plugin>
  <groupId>com.google.cloud.tools</groupId>
  <artifactId>appengine-maven-plugin</artifactId>
  <version>1.3.1</version>
  <configuration>
    <version>1</version>
  </configuration>
</plugin>
```

3. In the Cloud Shell create a `WEB-INF` directory in the guestbook frontend directory structure.

```
mkdir -p ~/guestbook-frontend/src/main/webapp/WEB-INF/
```

4. In the Cloud Shell code editor, select **File > Refresh**.
5. In the Cloud Shell code editor, create a file named `appengine-web.xml` in the `~/guestbook-frontend/src/main/webapp/WEB-INF/` directory.

This file is required to deploy the application to App Engine.

6. Add the following code to the `appengine-web.xml` file:

```
<appengine-web-app xmlns="http://appengine.google.com/ns/1.0">
  <service>default</service>
  <version>1</version>
  <threadsafe>true</threadsafe>
  <runtime>java8</runtime>
  <instance-class>B4_1G</instance-class>
  <sessions-enabled>true</sessions-enabled>
  <manual-scaling>
    <instances>2</instances>
  </manual-scaling>
  <system-properties>
    <property name="spring.profiles.active" value="cloud" />
  </system-properties>
</appengine-web-app>
```

Note

This configuration uses manual scaling rather than automatic scaling. Manual scaling provides fine control over the number of application instances. However, in a production setting you might want to use automatic scaling which can adapt dynamically to the load.

Task 3. Configure the frontend application to use the backend URL

Because the frontend application uses the Spring Cloud GCP Config starter, the application can get the backend endpoint URL directly from Cloud Runtime Configuration API.

In this task, you configure the `messages.endpoint` configuration variable in the frontend application to hold the backend service URL.

1. In the Cloud Shell, enter the following command to set a Configuration Runtime API variable called `messages.endpoint` to the backend URL that

App Engine will create for the backend guestbook service when the backend service is running.

The command associates the `messages.endpoint` variable with the `frontend cloud` profile.

```
gcloud beta runtime-config configs variables set messages.endpoint \
  "https://guestbook-service-dot-${PROJECT_ID}.appspot.com/guestbookMessages" \
  --config-name frontend cloud
```

Note

You created the `PROJECT_ID` variable that contains the active project ID at the beginning of the beginning of the lab. App Engine apps have a predictable base URL that includes the project ID as shown here.

You could override this property value directly in `appengine-web.xml`. But using Cloud Runtime Configuration API is a much more flexible and useful approach.

Task 4. Deploy the frontend application to App Engine

In this task, you deploy the frontend application to App Engine.

1. Change to the frontend application directory.

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

2. Use Apache Maven to deploy the frontend application to App Engine.

```
./mvnw appengine:deploy -DskipTests
```

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] -----
```

3. Find the frontend URL.

```
gcloud app browse
```

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
```

4. 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#getMessages(); content: {"timestamp":15316

5. Leave the tab open.

Task 5. Make the backend service application App Engine friendly

In this task, you configure the guestbook backend service application for App Engine by adding the App Engine plugin to the backend service application's `pom.xml` file.

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

```
<plugin>
  <groupId>com.google.cloud.tools</groupId>
  <artifactId>appengine-maven-plugin</artifactId>
  <version>1.3.1</version>
  <configuration>
    <version>1</version>
  </configuration>
</plugin>
```

3. In the Cloud Shell, create a `WEB-INF` directory in the backend service application directory structure.

```
mkdir -p ~/guestbook-service/src/main/webapp/WEB-INF/
```

4. In the Cloud Shell code editor, select **File > Refresh**.
5. In the Cloud Shell code editor, create a file named `appengine-web.xml` in the `~/guestbook-service/src/main/webapp/WEB-INF/` directory.

This file is required to deploy to App Engine.

6. Add the following code to the `appengine-web.xml` file:

```
<appengine-web-app xmlns="http://appengine.google.com/ns/1.0">
  <service>guestbook-service</service>
  <version>1</version>
  <threadsafe>true</threadsafe>
  <runtime>java8</runtime>
  <instance-class>B4_1G</instance-class>
  <manual-scaling>
    <instances>2</instances>
  </manual-scaling>
  <system-properties>
    <property name="spring.profiles.active" value="cloud" />
  </system-properties>
</appengine-web-app>
```

Task 6. Deploy the backend guestbook service application to App Engine

In this task, you deploy the backend guestbook service application to App Engine.

1. In the Cloud Shell change to the frontend application directory.

```
cd ~/guestbook-service
```

2. Use Apache Maven to deploy the backend guestbook service application to App Engine.

```
./mvnw appengine:deploy -DskipTests
```

3. When the build reports success, find the deployed backend service URL.

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

This command reports a clickable link to the URL for the backend guestbook service.

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

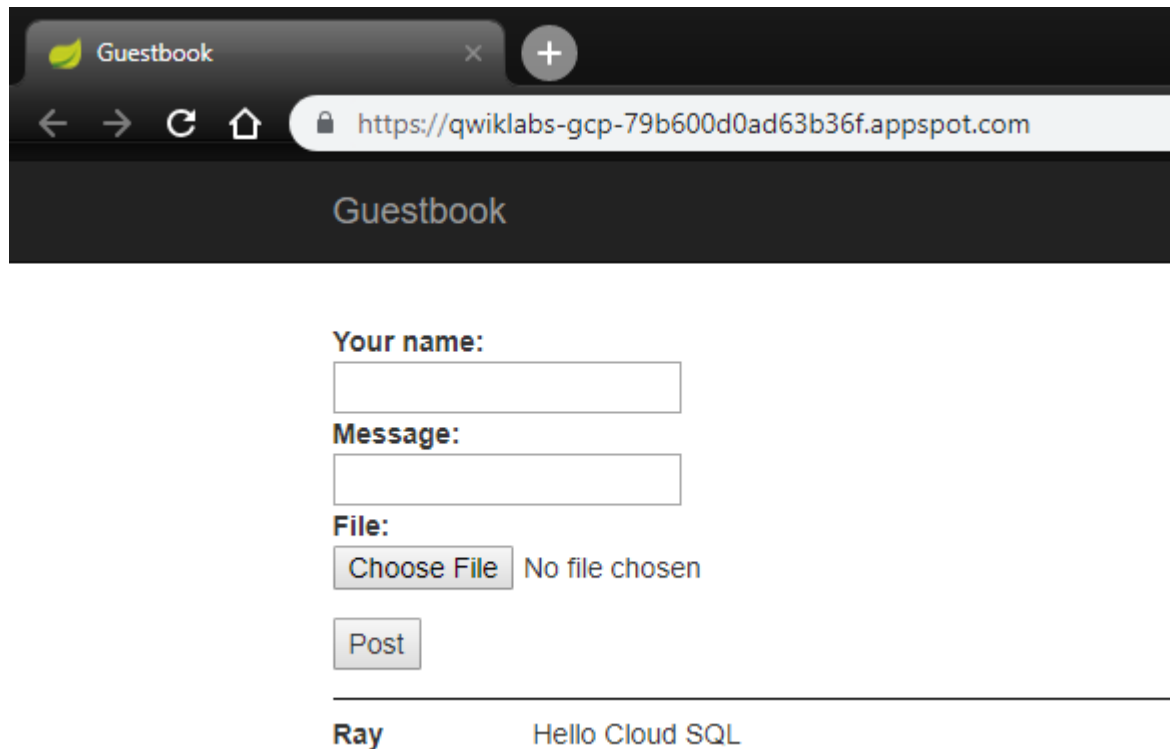
4. Click the URL link for the backend guestbook service.

Links similar 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" : "https://guestbook-service-dot-spring  
      "templated" : true  
    },  
    "profile" : {  
      "href" : "https://guestbook-service-dot-spring  
    }  
  }  
}
```

5. 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.



The screenshot shows a web browser window with the title 'Guestbook'. The address bar displays the URL 'https://qwiklabs-gcp-79b600d0ad63b36f.appspot.com'. The page content includes a 'Guestbook' header, followed by three input sections: 'Your name:' with a text box, 'Message:' with a text box, and 'File:' with a 'Choose File' button and the text 'No file chosen'. Below these is a 'Post' button. At the bottom, a horizontal line separates the header from the footer, which contains the text 'Ray' and 'Hello Cloud SQL'.

End your lab

When you have completed your lab, click **End Lab**. Qwiklabs removes the resources you've used and cleans the account for you. You'll be given an opportunity to rate the lab experience. Select the applicable number of stars, type a comment, and then click **Submit**. The number of stars indicates your rating:

- 1 star = Very dissatisfied
- 2 stars = Dissatisfied
- 3 stars = Neutral
- 4 stars = Satisfied
- 5 stars = Very satisfied

You can close the dialog box if you don't want to provide feedback.

For feedback, suggestions, or corrections, use the **Support** tab.

Copyright 2019 Google LLC All rights reserved. Google and the Google logo are trademarks of Google LLC. All other company and product names may be trademarks of the respective companies with which they are associated.