Start Lab 01:00:00

JAVAMS10 Debugging with Stackdriver Debugger

1 hour Free ★★★★☆ 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 the previous lab, you repackaged and then deployed the demo application to App Engine. In this lab, you configure Cloud Logging, Cloud Debugger, and Cloud Monitoring. During the lab you use cloud services to inspect logs, and debug and monitor the performance of the demo application while it is running on App Engine.

Cloud Debugger is a feature of Google Cloud Platform that enables you to inspect the state of an application at any code location, without stopping or slowing down the running app. Cloud Debugger makes it easier to view the application state without adding logging statements.

You can use Cloud Debugger with any deployment of your application, including test, development, and production. Cloud Debugger adds less than 10 milliseconds to the request latency only when the application state is captured. In most cases, this additional latency is not noticeable by users.

Objectives

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

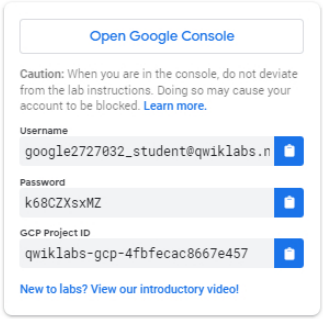
- Configure Cloud Logging for an App Engine application
- Configure Cloud Debugger source content for App Engine debugging
- Configure Cloud Debugger logpoints and snapshots
- Enable Cloud Monitoring

Task 0. Lab Preparation

Access Qwiklabs

How to start your lab and sign in to the Console

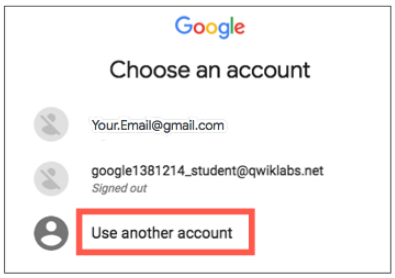
1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is a panel populated with the temporary credentials that you must use for this lab.



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**.



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

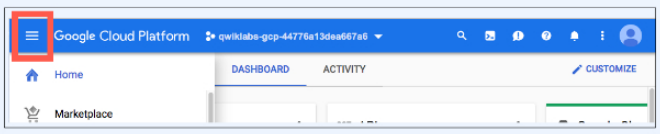
   Important: You must use the credentials from the Connection Details panel. Do not use your Qwiklabs credentials. If you have your own GCP account, do not use it for this lab (avoids incurring charges).

5. Click through the subsequent pages:

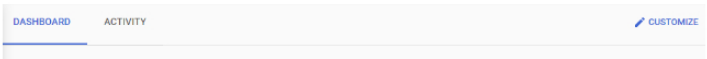
   - Accept the terms and conditions.
   - Do not add recovery options or two-factor authentication (because this is a temporary account).
   - Do not sign up for free trials.

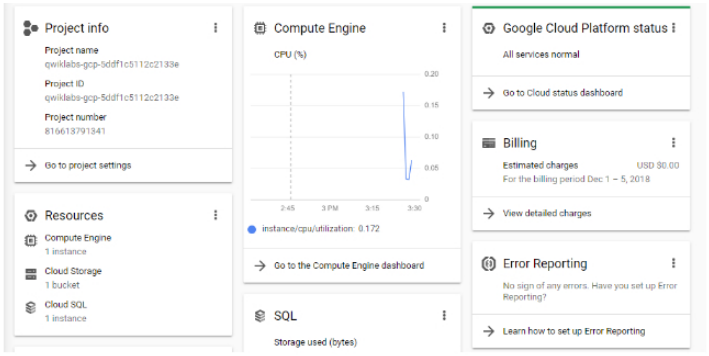
   After a few moments, the GCP console opens in this tab.

> **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

To begin the lab, click the **Activate Cloud Shell** button at the top of the Google Cloud Console. To activate the code editor, click the `Open Editor` button on the toolbar of the Cloud Shell window. This sets up the editor in a new tab with continued access to Cloud Shell.

> **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 once the Cloud SQL server is ready and both application microservices components have been deployed to App Engine. You might have to wait up to 10 minutes for the deployment tasks to complete.

1. 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)')
```

Click **Authorize** to authorize Cloud Shell to make API calls. It may take up to 10 minutes for the lab setup to complete and the bucket to be created.

2. Verify that the demo application files were created.

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

3. Copy the application folders to Cloud Shell.

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

4. Make the Maven wrapper scripts executable. Now you're ready to go!

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

Task 1. Examine Cloud logs

In this task, you examine Cloud logs for the demo application running on App Engine.

1. Open a new browser tab and navigate to the Google Cloud console.

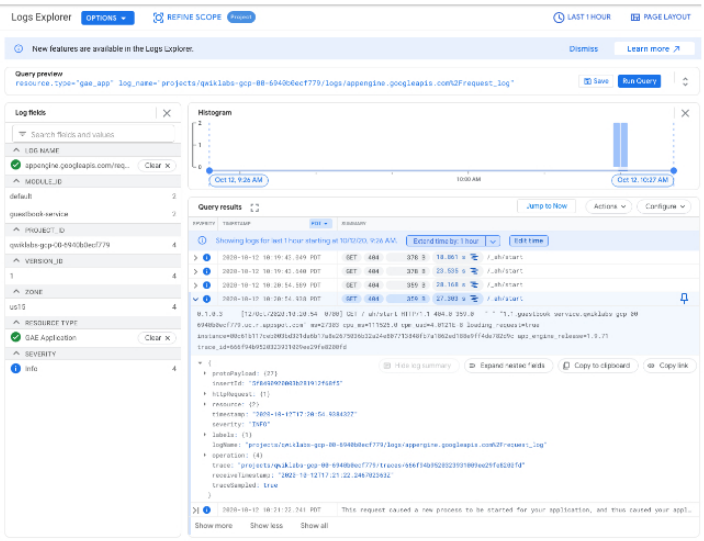
2. Navigate to **Operations** > **Logging** > **Logs Explorer**.

3. In the **Log fields** pane, select **GAE Application**.

4. In the **Log name** section, select **appengine.googleapis.com/request_log**.

The default App Engine application log is displayed. When you output a log message, it is grouped by the request. When the application first starts, the log messages are grouped under `/_ah/start` request.

5. Expand an entry to view the detailed log entry.



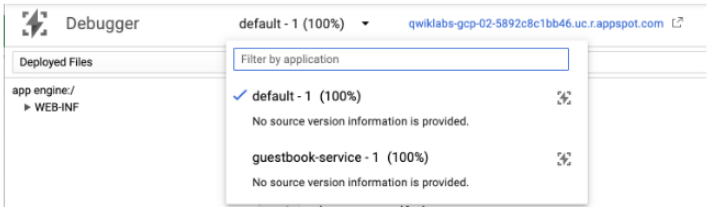
Task 2. Configure Cloud Debugger

In this task, you configure Cloud Debugger so that it can be used to debug the demo microservices application used in this set of labs. The demo application was automatically deployed to App Engine for you as part of the lab setup.

1. In the Console menu, navigate to **Operations** > **Debugger**.

   At the top, the running App Engine deployments are listed.

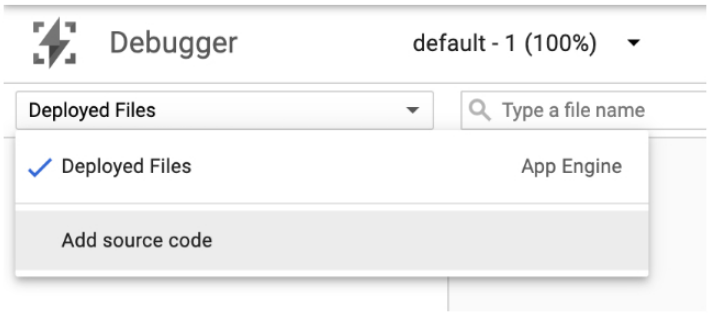
2. In the drop-down list, select **default - 1 (100%)**.



default - 1 is the frontend application. The source code is not yet available for debugging.

3. Click **Select Source** on the left of the screen to expand it and select **Deployed Files**.

   You can provide source code to Cloud Debugger in several ways.



4. Select **Add source code** in the **Deployed Files** drop-down list.

   A list of methods for providing source code is displayed. In this lab, you will add the source code to the Google Source Repositories service.

5. Switch to Cloud Shell and enable the Google Cloud Source Repository API.

```
gcloud services enable sourcerepo.googleapis.com
```

6. Create a source repository for source capture.

```
gcloud source repos create google-source-captures
```

> **Note:** You might get a warning that you may be billed for this repository. You can safely ignore this warning since you are using a QwikLabs account.

7. Change to the frontend application directory.

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

8. Configure the `git` email and username properties.

   These properties are used for the code upload.

```
git config --global user.email $(gcloud config get-value core/account)
git config --global user.name "devstar"
```

9. Clone the empty `google-source-captures` repository.

```
gcloud source repos clone google-source-captures --project=$PROJECT_ID
```

   The repository has been created at `~/guestbook-frontend/google-source-captures`.

10. Copy the source files into the local repository directory.

```
cd google-source-captures
cp -rip ../src/* .
```

11. Commit the source files to the Cloud Source Repository.

```
git add .
git commit -m "initial commit"
git push -u origin master
```

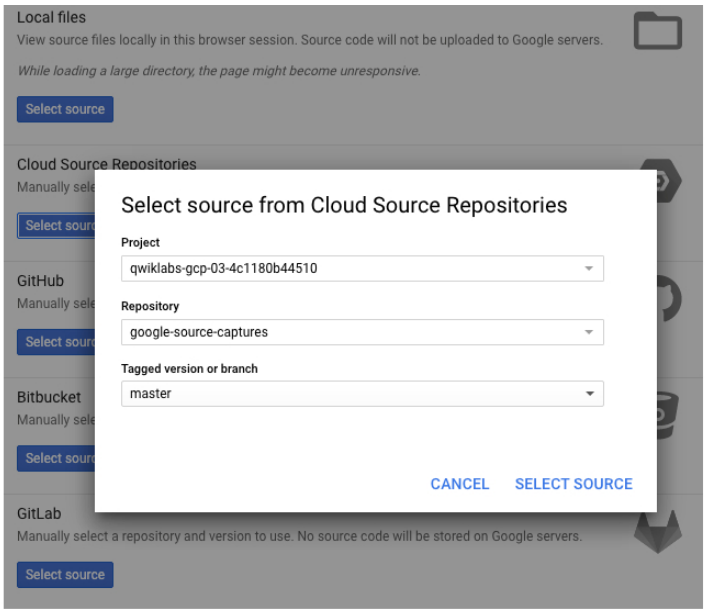
Task 3. Use Cloud Debugger to debug an application

In this task, you use Cloud Debugger to debug the demo application running on App Engine.

1. Return to the Cloud Debugger console and click **Select source** for Cloud Source Repositories.

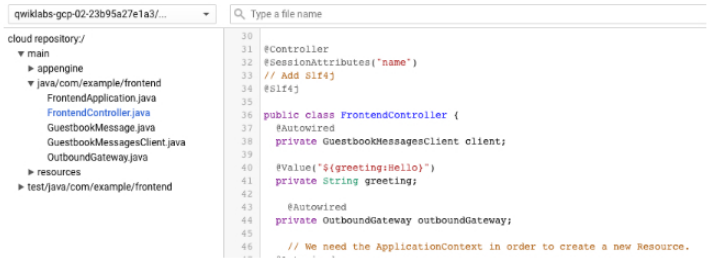
   You should see the master branch in the **google-source-captures** repository that you created.

Alternative source code



2. Click **Select Source**.

3. In the left menu under `cloud repository:/`, navigate to and open `main/java/com/example/frontend/FrontendController.java`.



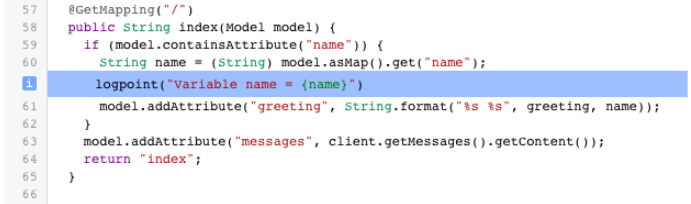
From here, you have a significant amount of control. For example, you can add a log message.

4. On the right side of the window, click **Logpoint**.

5. In the source, click the line number where you want to add a log message, and edit the message to print the text and variables that you want to see.

In this example, the message is changed to print the text `Variable name =`, followed by the value of the local variable `name`.

6. Click **Add**. You can add as many log messages as you want.



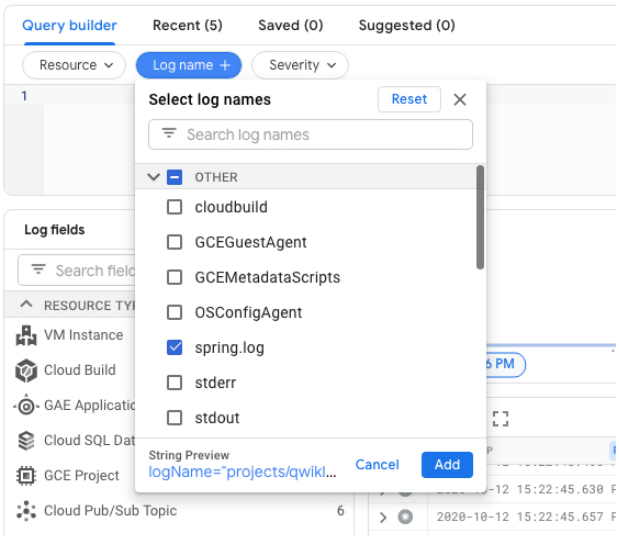
7. Open the Guestbook application tab in your browser. You can retrieve the link to your app by executing the following command in the Cloud Shell:

```
gcloud app browse
```

8. In the application, enter a name and message to trigger the code.

9. Return to the Logs Viewer: **Operations** > **Logging** > **Logs Explorer**.

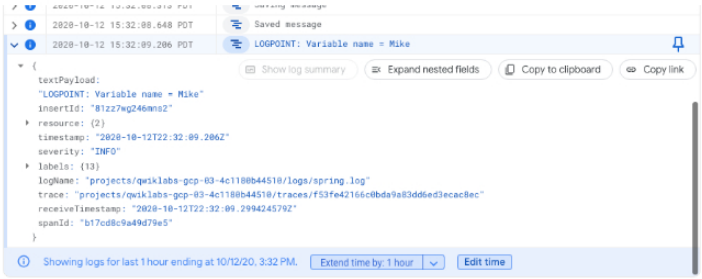
10. Click on the **Log name** dropdown in the **Query builder** pane and select **spring.log** in the list of log files:



Click **Add**.

11. Click **Run Query**.

12. Find the `LOGPOINT` messaage and expand it. This message should be close to the end and highlighted with a blue information button:



13. Return to Debugger: **Operations** > **Debugger**.

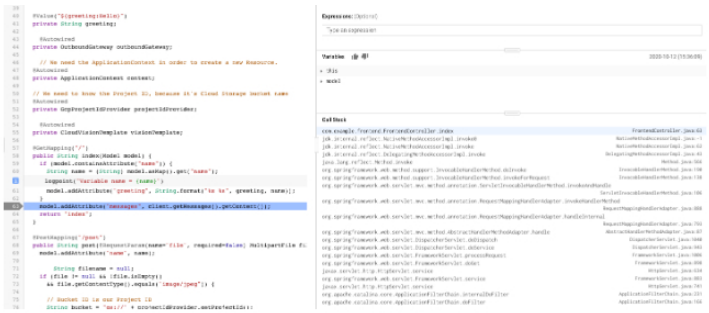
14. In the source view on the right side, click **Snapshot**.

    Snapshot allows you to capture the stack in a moment in time. It is almost like stepping through a real debugger, but it does not stop the application for your users.

15. In the source, click the line number where you want to capture information.

16. Switch to the demo application and post another guestbook message.

    As soon as a request flows through the line, the **Call Stack** is captured, and you can explore the internal state of the application at that point in time. You can add conditionals to both logpoints and snapshots, so that you view only certain requests based on variables that are in scope (for example, session ID).



Note: Cloud Debugger works with various languages, and also outside of App Engine. You can also debug your application in the same way when you deploy your application on-premises, in a VM, or in containers.

Task 4. Enable Cloud Monitoring

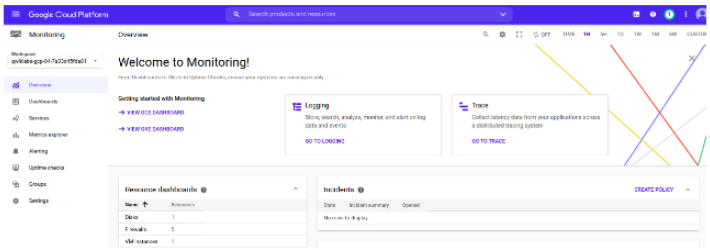
Create a Monitoring workspace

You will now setup a Monitoring workspace that's tied to your Qwiklabs GCP Project. The following steps create a new account that has a free trial of Monitoring.

1. In the Google Cloud Platform Console, click on **Navigation menu** > **Monitoring**.

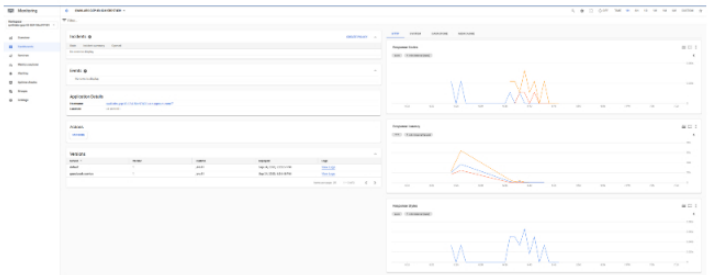
2. Wait for your workspace to be provisioned.

   When the Monitoring dashboard opens, your workspace is ready.



3. Navigate to **Dashboards**. Click **App Engine** and select your App Engine service under **Projects**.

   After a minute or two, an overview dashboard of your App Engine services appears. You might have to refresh the page.



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 will 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 the following:

- 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, please use the **Support** tab.