Start Lab 02:00:00

# JAVAMS01 Bootstrapping the Application Frontend and Backend



2 hours 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.

When implementing such techniques in a traditional on-premises environment, you have to build and manage these servers and service capabilities yourself.

But what do you do when you move to the cloud? Going cloud native is not only about replicating the architectures that you have to implement in your own data center, simply substituting virtual machine-based workloads for bare-metal workloads. Cloud-native applications can adopt the fully managed cloud services and platforms that require little or no manual operational overhead.

In these labs, you learn how to replace the external dependencies that you would otherwise need to maintain and operate yourself with fully managed services on Google Cloud Platform (GCP). Through the use of Spring Cloud GCP components and preconfigured Spring Boot starters, you can quickly replace RDBMs like MySQL with Cloud SQL, messaging pipeline services like RabbitMQ with Cloud Pub/Sub, distributed trace stores like Zipkin with Cloud Trace, and centralized configuration server services with Cloud Runtime Configuration API.

You first deploy and test the application in Cloud Shell, starting with this lab, where you bring the application up by running the two microservices components in separate Cloud Shell console tabs.

In later labs, you deploy the application components to App Engine and Kubernetes Engine and explore how to use Cloud Trace to debug and monitor your application when it is deployed to those services.

Cloud Shell is an interactive, Linux command-line shell environment for managing resources hosted on GCP. When you start Cloud Shell, it provisions a small Compute Engine virtual machine running a Debian-based Linux operating system. Cloud Shell provides command-line access to the virtual machine instance in a terminal window that opens in the GCP console. You can open multiple shell connections to the same instance. The beta version of Cloud Shell code editor is now available for use. You can use the code editor to browse file directories as well as view and edit files, with continued access to Cloud Shell.

# Objectives

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

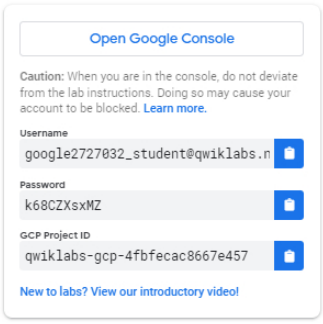
- Configure Cloud Shell to run a multi-part Java application locally
- Use Apache Maven to launch Java applications in Cloud Shell
- Use `curl` and the Cloud Shell web preview to test connectivity to web applications running locally in Cloud Shell

# Task 0. Lab Setup

## Access Qwiklabs

**How to start your lab and sign in to the Console**

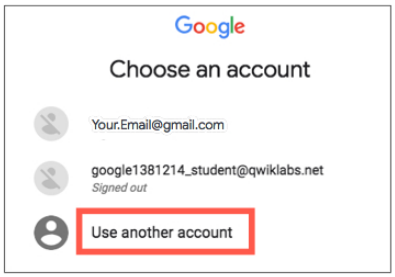
1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is a panel populated with the temporary credentials that you must use for this lab.




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).

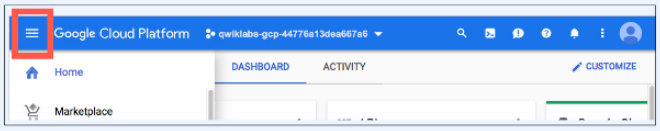
5. Click through the subsequent pages:

   - Accept the terms and conditions.

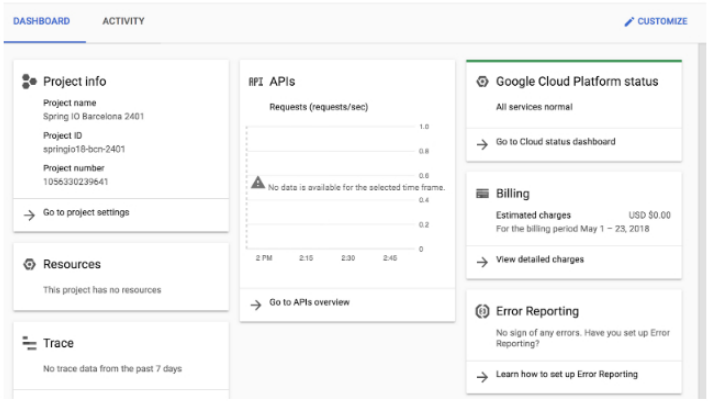
- Do not add recovery options or two-factor authentication (because this is a temporary account).
- Do not sign up for free trials.

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.



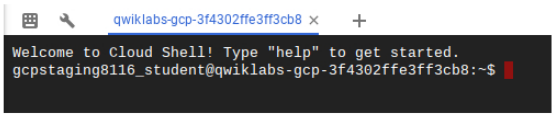

## Open Cloud Shell

You will do most of the work in Cloud Shell. Cloud Shell is a command-line environment running in the Google cloud. This Debian-based virtual machine is loaded with all the development tools you need (such as `docker`, `gcloud`, and `kubectl`) and provides a persistent 5GB home directory.

1. In the upper-right corner of the screen, click **Activate Cloud Shell** (  ) to open Cloud Shell.

2. Click **Start Cloud Shell**.

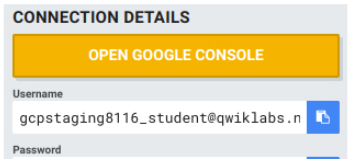
   After a moment of provisioning, a prompt appears:




3. Run the following commands to view your preset account and project. When you use `gcloud` to create resources, this is where they are stored.

```
gcloud config list account
gcloud config list project
```

4. Look for an assigned zone on your lab page. You might not have an assigned zone.



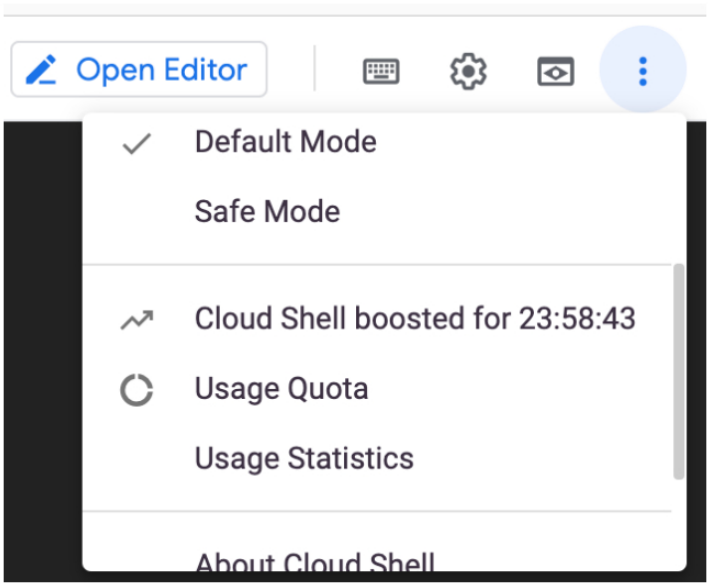

5. Run the following command to set your zone. Substitute your assigned zone for `[YOUR_ZONE]` if you have one. Otherwise, use a zone close to you for low latency.

```
gcloud config set compute/zone [YOUR_ZONE]
```

6. Run the following command to see all variables set.

```
gcloud config list
```

7. Click on **Boost Cloud Shell** after that you will see **Cloud Shell boosted** .




**Note**

When you run `gcloud` on your own machine, the configuration settings are persistent across sessions. But in Cloud Shell, you need to set this option for every new session and reconnection.

You perform most of the lab instructions directly in Cloud Shell.

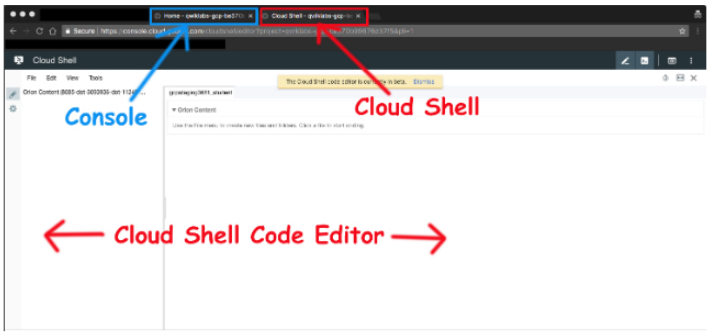
## Launch Google Cloud Shell Code Editor

Use the Google Cloud Shell Code Editor to easily create and edit directories and files in the Cloud Shell instance.

Once you activate the Google Cloud Shell, click the **Open editor** button to open the Cloud Shell Code Editor.


You now have three interfaces available:

- The Cloud Shell Code Editor
- Console (By clicking on the tab). You can switch back and forth between the Console and Cloud Shell by clicking on the tab.
- The Cloud Shell Command Line (By clicking on **Open Terminal** in the Console)




**Note**

The labs provide instructions based on the Cloud Shell code editor for actions that require editing code. However, you can use command-line editors like vi, vim, emacs, or nano in Cloud Shell if you prefer.

## Task 1. Bootstrap the application

In this task, you clone the source repository for the demo application that is used throughout these labs. The demo application has two parts:

- A frontend application (`guestbook-frontend`) that manages the user interface presented in a web browser
- A backend service application (`guestbook-service`) that processes the data and manages the messaging and database interfaces

### Clone the demo application

Clone the demo application, run the following command:

```
cd ~/
git clone https://github.com/saturnism/spring-cloud-gcp-guestbook.git
```

### Run the backend locally

To run, test, and use the backend locally, perform the following steps:

1. Make a copy of the initial version of the backend application (`guestbook-service`).

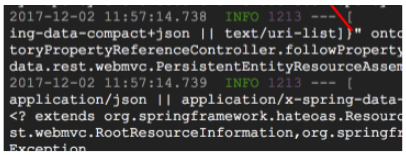
```
cp -a ~/spring-cloud-gcp-guestbook/1-bootstrap/guestbook-service \
  ~/guestbook-service
```

2. Run the backend application.

```
cd ~/guestbook-service
./mvnw -q spring-boot:run -Dserver.port=8081
```

3. Open a new Cloud Shell session tab to test the backend application by clicking the plus (+) icon to the right of the title tab for the initial Cloud Shell session.

   This action opens a second Cloud Shell console to the same virtual machine.




4. While the backend application (`guestbook-service`) is still running, test the service by running the following command in the second Cloud Shell tab:

```
curl http://localhost:8081/guestbookMessages
```

5. Post a new message.

```
curl -XPOST -H "content-type: application/json" \
  -d '{"name": "Ray", "message": "Hello"}' \
  http://localhost:8081/guestbookMessages
```

6. List all the messages.

```
curl http://localhost:8081/guestbookMessages
```

### Run the frontend locally

To run the frontend locally, perform the following steps:

1. Make a copy of the initial version of the frontend application (`guestbook-frontend`).

```
cp -a ~/spring-cloud-gcp-guestbook/1-bootstrap/guestbook-frontend \
  ~/guestbook-frontend
```

2. Run the frontend application.

   The frontend web application launches on port 8080.

```
cd ~/guestbook-frontend
./mvnw -q spring-boot:run
```

## Task 2. Test the guestbook application

In this task, you test the demo application. The demo application is a simple Java application composed of a microservices backend, and a frontend consuming it. You extend this simple application in later labs to leverage various GCP services. You eventually deploy it to the cloud, using both App Engine and Kubernetes Engine.

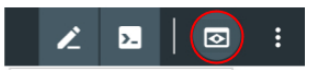
### Access the frontend application through the Cloud Shell web preview

To access and use the frontend application through the web preview, perform the following steps:

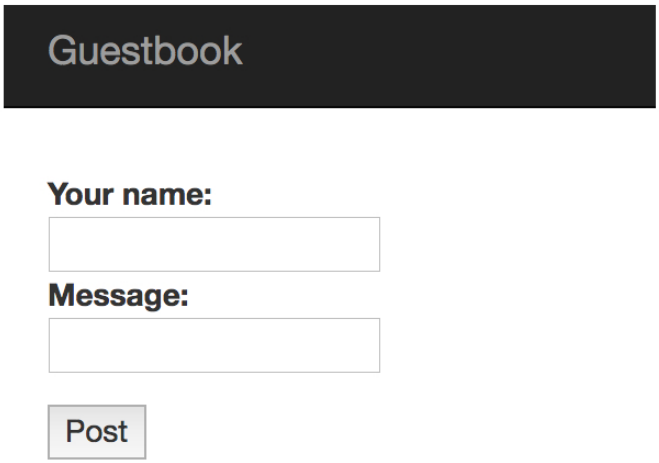
1. Click **Web Preview** ( ).

2. Select **Preview on port 8080** to access the application on port 8080.

   A new browser tab displays the connection to the frontend application.



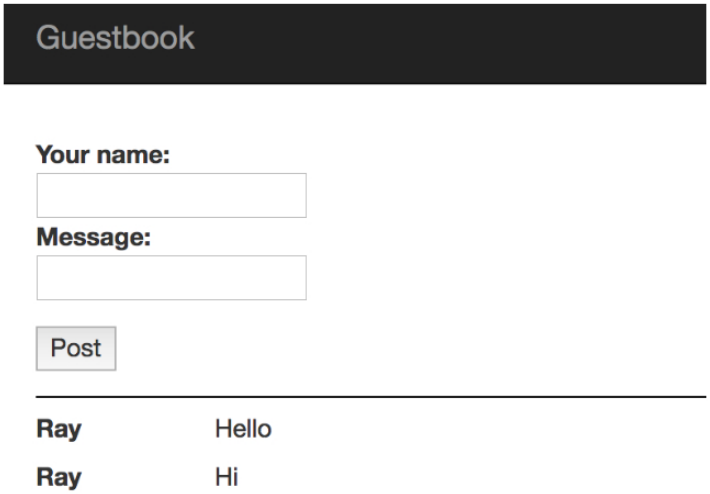

3. For **Your name**, type your name.




4. For **Message**, type a message.

5. Click **Post** to continue.

   The messages are listed below the message input section.




## Use Cloud Shell to test the backend service

To use Cloud Shell to test the backend service, perform the following steps:

1. Open a third Cloud Shell tab.

2. In the new shell tab, list all the messages that you added through a call to the backend `guestbook-service` API.

```
curl -s http://localhost:8081/guestbookMessages
```

3. Use `jq` to parse the JSON return text.

   For example, the following command prints out only the messages:

```
curl -s http://localhost:8081/guestbookMessages \
  | jq -r '._embedded.guestbookMessages[] | {name: .name, message:
```

```
.message}'
```

## 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.