

JAVAMS01

Bootstrapping the Application Frontend and Backend

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.

Through the use of Spring Cloud, Google Cloud components, and preconfigured Spring Boot starters, you can quickly replace RDBMs like MySQL with Cloud SQL, messaging pipeline services like RabbitMQ with 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 in this series, 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.

Objectives

In this lab, you will 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.


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 

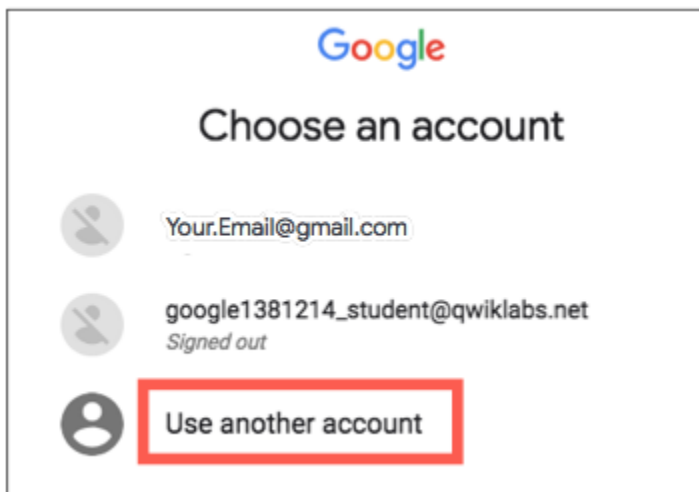
GCP Project ID
qwiklabs-gcp-4fbfecac8667e457 

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

- 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.

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



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

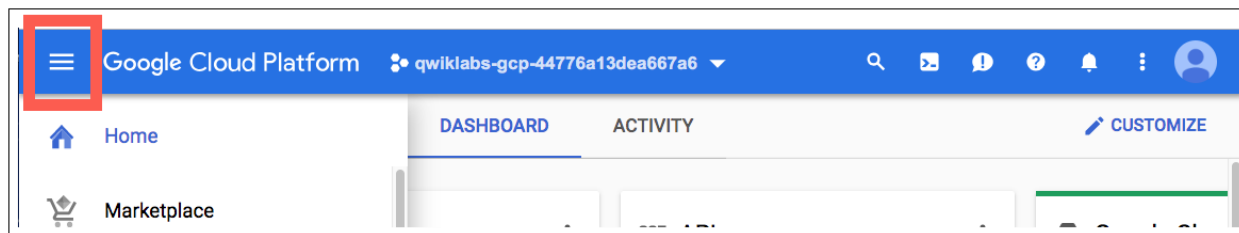
Note: You must use the credentials from the Connection Details panel. Do not use your Google Cloud Skills Boost credentials. If you have your own Google Cloud account, do not use it for this lab (avoids incurring charges).

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.

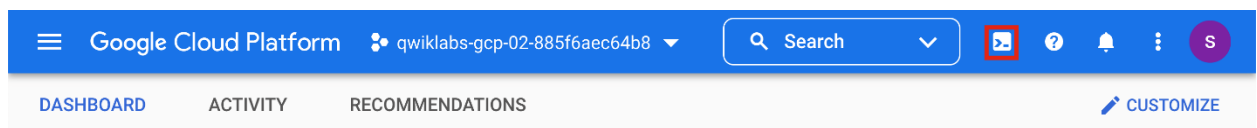


Activate Google Cloud Shell

Google Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud.

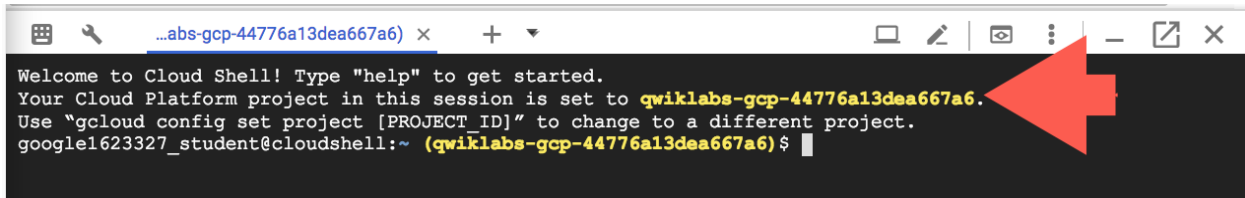
Google Cloud Shell provides command-line access to your Google Cloud resources.

1. In Cloud console, on the top right toolbar, click the Open Cloud Shell button.



2. Click **Continue**.

It takes a few moments to provision and connect to the environment. When you are connected, you are already authenticated, and the project is set to your *PROJECT_ID*. For example:



```
...abs-gcp-44776a13dea667a6) x + ▾
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to qwiklabs-gcp-44776a13dea667a6.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
google1623327_student@cloudshell:~ (qwiklabs-gcp-44776a13dea667a6)$
```

gcloud is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.

- You can list the active account name with this command:

```
gcloud auth list
```

Copied!

content_copy

Output:

```
Credentialed accounts:
```

```
- @.com (active)
```

Example output:

```
Credentialed accounts:
```

```
- google1623327_student@qwiklabs.net
```

- You can list the project ID with this command:

```
gcloud config list project
```

Copied!

content_copy

Output:

```
[core]
```

```
project =
```

Example output:

```
[core]
```

```
project = quiklabs-gcp-44776a13dea667a6
```

Note: Full documentation of **gcloud** is available in the [gcloud CLI overview guide](#).

Task 1. Bootstrap the application

In this task, you will 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

- In Cloud Shell, clone the demo application by executing the following command:

```
cd ~/
git clone --depth=1 https://github.com/GoogleCloudPlatform/training-data-analyst
ln -s ~/training-data-analyst/courses/java-microservices/spring-cloud-gcp-guestbook ~/spring-cloud-gcp-guestbook
Copied!
content_copy
```

Run the backend locally

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

1. To make a copy of the initial version of the backend application (`guestbook-service`), execute the following command:

```
cp -a ~/spring-cloud-gcp-guestbook/1-bootstrap/guestbook-service ~/guestbook-service
Copied!
content_copy
```

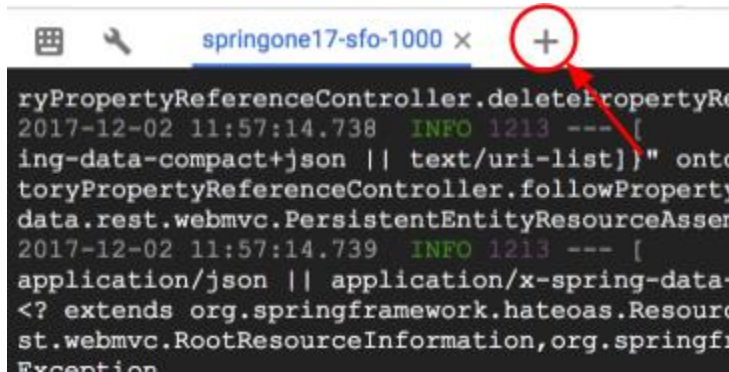
2. To run the backend application, execute the following command:

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

content_copy

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 executing the following command in the second Cloud Shell tab:

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

Copied!

content_copy

5. Post a new message:

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

Copied!

content_copy

6. List all the messages:

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

Copied!

content_copy

Run the frontend locally

To run the frontend locally, perform the following steps:

1. To make a copy of the initial version of the frontend application (guestbook-frontend), execute the following command:

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

Copied!

content_copy

2. To run the frontend application, execute the following command:

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

Copied!

content_copy


The frontend web application launches on port 8080.

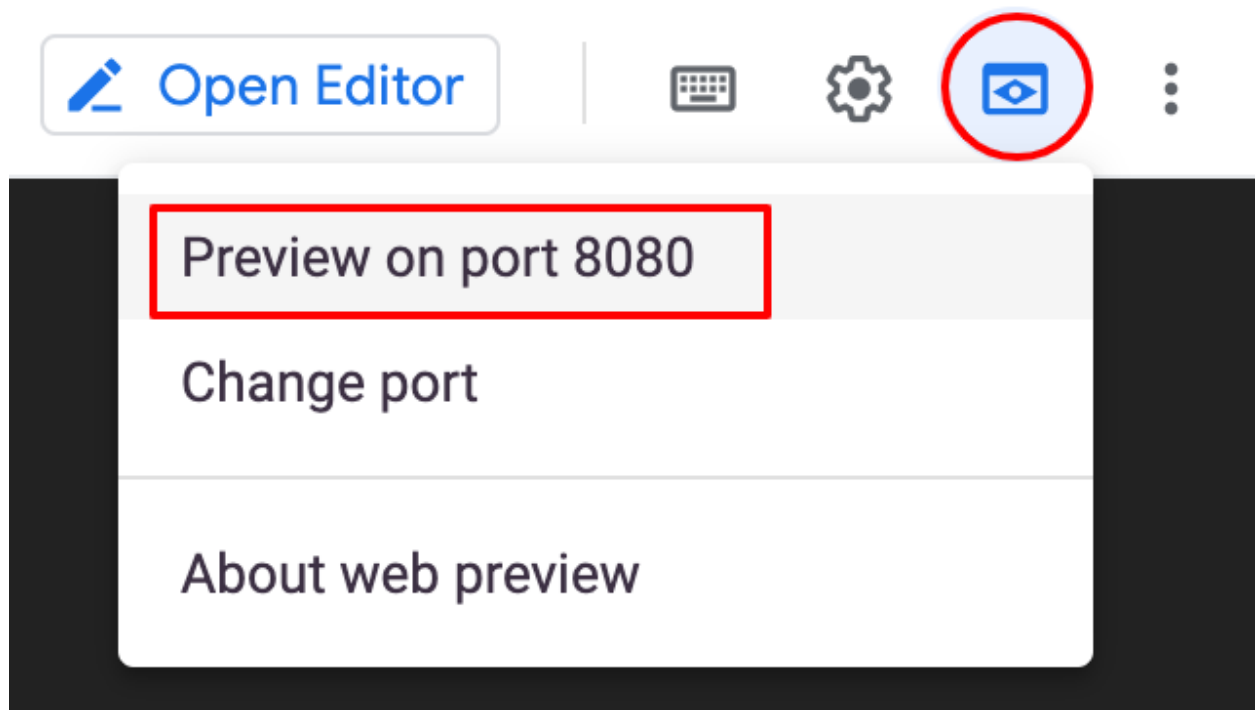
Task 2. Test the guestbook application

In this task, you will 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 Google Cloud services. You will eventually deploy it to the cloud, using both App Engine and Google 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 the **Web Preview** () icon.
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.

Guestbook

Your name:

Message:

Post

4. For *Message*, type a message.

5. Click **Post** to continue.

The messages are listed below the message input section.

Guestbook

Your name:

Message:

Post

Ray	Hello
------------	-------

Ray	Hi
------------	----

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 by executing the following command:

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

Copied!

content_copy

3. To use `jq` to parse the JSON return text, execute the following command. For example, the following command prints out only the messages.

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

Copied!

content_copy

Congratulations!

In this lab, you have configured Cloud Shell to run multi-part Java applications locally. Next, you used Maven to launch Java applications in the Cloud shell. Finally, you used `curl` and the Cloud Shell web preview to test connectivity to web applications running locally in Cloud Shell.