[[](https://www.cloudskillsboost.google/)](https://www.cloudskillsboost.google/)

**LAB 2**

**JAVAMS02 Configuring and Connecting to Cloud SQL**

Lab instructions and tasks

* [Overview](https://www.cloudskillsboost.google/course_templates/20/labs/499561#step1)
* [Setup and requirements](https://www.cloudskillsboost.google/course_templates/20/labs/499561#step2)
* [Task 1. Fetch the application source files](https://www.cloudskillsboost.google/course_templates/20/labs/499561#step3)
* [Task 2. Create a Cloud SQL instance, database, and table](https://www.cloudskillsboost.google/course_templates/20/labs/499561#step4)
* [Task 3. Use Spring to add Cloud SQL support to your application](https://www.cloudskillsboost.google/course_templates/20/labs/499561#step5)
* [Task 4. Configure an application profile to use Cloud SQL](https://www.cloudskillsboost.google/course_templates/20/labs/499561#step6)
* [Congratulations!](https://www.cloudskillsboost.google/course_templates/20/labs/499561#step7)
* [End your lab](https://www.cloudskillsboost.google/course_templates/20/labs/499561#step8)

Overview

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

In this lab, you will also reconfigure the demo application to use an external database. The demo application is initially configured to use an embedded HSQL database. You will use Spring Boot to modify the application to use a cloud database.

Rather than building and maintaining your own MySQL instance in the cloud, you can use managed services as much as possible to reduce operation overhead and increase reliability. Google Cloud has a managed MySQL and PostgreSQL service called Cloud SQL.

In this lab, you will create and configure a Cloud SQL instance and reconfigure the application to use Cloud SQL.

Cloud SQL is a fully managed database service that makes it easy to set up, maintain, manage, and administer your relational PostgreSQL and MySQL databases in the cloud. Cloud SQL offers high performance, scalability, and convenience. Hosted on Google Cloud, Cloud SQL provides a database infrastructure for applications running anywhere.

Objectives

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

* Create a Cloud SQL instance, database, and table.
* Use Spring to add Cloud SQL support to your application.
* Configure an application profile for Cloud SQL.
* Verify that an application is using Cloud SQL.

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.



1. 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.

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



1. 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).

1. 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.

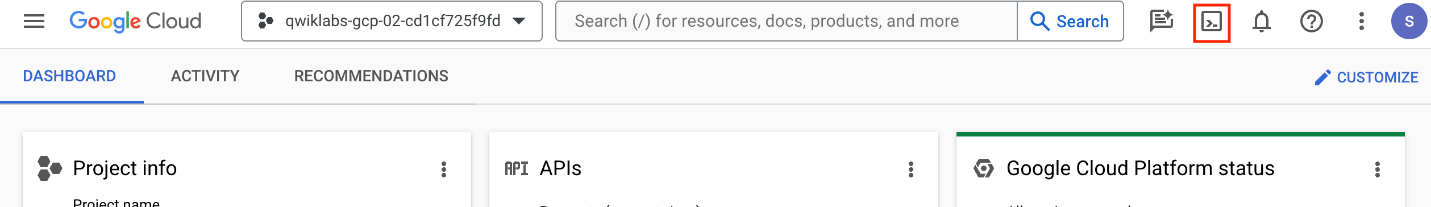
Note: You can view the menu with a list of Google Cloud Products and Services by clicking the Navigation menu at the top-left. 

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.



1. 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:



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**

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**

Output:

[core]

project =

Example output:

[core]

project = qwiklabs-gcp-44776a13dea667a6

Note: Full documentation of gcloud is available in the [gcloud CLI overview guide](https://cloud.google.com/sdk/gcloud" \t "_blank).

Task 1. Fetch the application source files

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

1. To clone the project repository, execute the following commands:

**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**

1. Copy the relevant folders to your home directory:

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

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

Task 2. Create a Cloud SQL instance, database, and table

In this task, you provision a new Cloud SQL instance, create a database on that instance, and then create a database table with a schema that can be used by the demo application to store messages.

Enable Cloud SQL Administration API

You enable Cloud SQL Administration API and verify that Cloud SQL is preprovisioned.

1. In Cloud Shell, enable the Cloud SQL Administration API:

**gcloud services enable sqladmin.googleapis.com**

1. Confirm that the Cloud SQL Administration API is enabled:

**gcloud services list | grep sqladmin**

1. List the Cloud SQL instances:

**gcloud sql instances list**

No instances are listed yet.

Create a Cloud SQL instance

You create a new Cloud SQL instance.

* Provision a new Cloud SQL instance:

**gcloud sql instances create guestbook --region=us-west1**

Provisioning the Cloud SQL instance will take a couple of minutes to complete.

The output should indicate that the Cloud SQL instance creation was done.

Create a database in the Cloud SQL instance

You create a database to be used by the demo application.

* Create a messages database in the MySQL instance:

**gcloud sql databases create messages --instance guestbook**

Connect to Cloud SQL and create the schema

By default, Cloud SQL is not accessible through public IP addresses. You can connect to Cloud SQL in the following ways:

* Use a local Cloud SQL proxy.
* Use gcloud to connect through a CLI client.
* From the Java application, use the MySQL JDBC driver with an SSL socket factory for secured connection.

You create the database schema to be used by the demo application to store messages.

1. Use gcloud CLI to connect to the database:

**gcloud sql connect guestbook**

This command temporarily allowlists the IP address for the connection.

1. Press Enter at the following prompt to leave the password empty for this lab:

Allowlisting your IP for incoming connection for 5 minutes...done.

Connecting to database with SQL user [root].Enter password:

The root password is empty by default.

The prompt changes to mysql> to indicate that you are now working in the MySQL command-line environment.

Note: For security reasons, the default setting for Cloud SQL does not allow connections to the public IP unless an address is explicitly allowlisted. The gcloud sql connect command line automatically and temporarily allowlists your incoming connection.

It takes a minute or two for the allowlisting process to complete before the MySQL administration client can connect.

1. List the databases:

**show databases;**

This command lists all of the databases on the Cloud SQL instance, which should include the messages database that you configured in previous steps.

A black screen with white text

Description automatically generated

1. Switch to the messages database:

**use messages;**

1. Create the table:

**CREATE TABLE guestbook\_message (**

**id BIGINT NOT NULL AUTO\_INCREMENT,**

**name CHAR(128) NOT NULL,**

**message CHAR(255),**

**image\_uri CHAR(255),**

**PRIMARY KEY (id)**

**);**

1. Exit the MySQL management utility by executing the following command:

**exit**

You return to the standard Cloud Shell user prompt.

Task 3. Use Spring to add Cloud SQL support to your application

In this task, you add the Spring Cloud GCP Cloud SQL starter to your project so that you can use Spring to connect to your Cloud SQL database.

Add the Spring Cloud GCP Cloud SQL starter

From a Java application, you can integrate with a Cloud SQL instance by using the standard method, where you use the JDBC driver. However, configuring the JDBC driver for use with Cloud SQL can be more complicated than connecting to a standard MySQL server because of the additional security that Google Cloud puts in place. Using the Spring Cloud GCP Cloud SQL starter simplifies this task.

The Spring Cloud GCP project provides configurations that you can use to automatically configure your Spring Boot applications to consume Google Cloud services, including Cloud SQL.

You update the guestbook service's pom.xml file to import the Spring Cloud GCP BOM and also the Spring Cloud GCP Cloud SQL starter. This process involves adding the milestone repository to use the latest Spring release candidates.

1. Edit the guestbook-service/pom.xml file in the Cloud Shell code editor or in the shell editor of your choice.

Note: The lab instructions refer only to the Cloud Shell code editor, but you can use vi, vim, emacs, or nano as your text editor if you prefer.

1. Insert an additional dependency for spring-cloud-gcp-starter-sql-mysql just before the </dependencies> closing tag:

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-gcp-starter-sql-mysql</artifactId>

</dependency>

Disable Cloud SQL in the default profile

For local testing, you can continue to use a local database or an embedded database. The demo application is initially configured to use an embedded HSQL database.

To continue to use the demo application for local runs, you disable the Cloud SQL starter in the default application profile by updating the application.properties file.

1. In the Cloud Shell code editor, open guestbook-service/src/main/resources/application.properties.
2. Add the following property setting:

**spring.cloud.gcp.sql.enabled=false**

Task 4. Configure an application profile to use Cloud SQL

In this task, you create an application profile that contains the properties that are required by the Spring Boot Cloud SQL starter to connect to your Cloud SQL database.

Configure a cloud profile

When deploying the demo application into the cloud, you want to use the production-managed Cloud SQL instance.

You create an application profile called cloud profile. The cloud profile leaves the Cloud SQL starter that is defined in the Spring configuration profile enabled. And it includes properties used by the Cloud SQL starter to provide the connection details for your Cloud SQL instance and database.

1. In Cloud Shell, find the instance connection name:

**gcloud sql instances describe guestbook --format='value(connectionName)'**

This command format filters out the connectionName property from the description of the guestbook Cloud SQL object. The entire string that is returned is the instance's connection name.

The string looks like the following example:

qwiklabs-gcp-4d0ab38f9ff2cc4c:us-west1:guestbook

1. In the Cloud Shell code editor, create an application-cloud.properties file in the guestbook-service/src/main/resources directory.
2. In the Cloud Shell code editor, open guestbook-service/src/main/resources/application-cloud.properties and add the following properties:

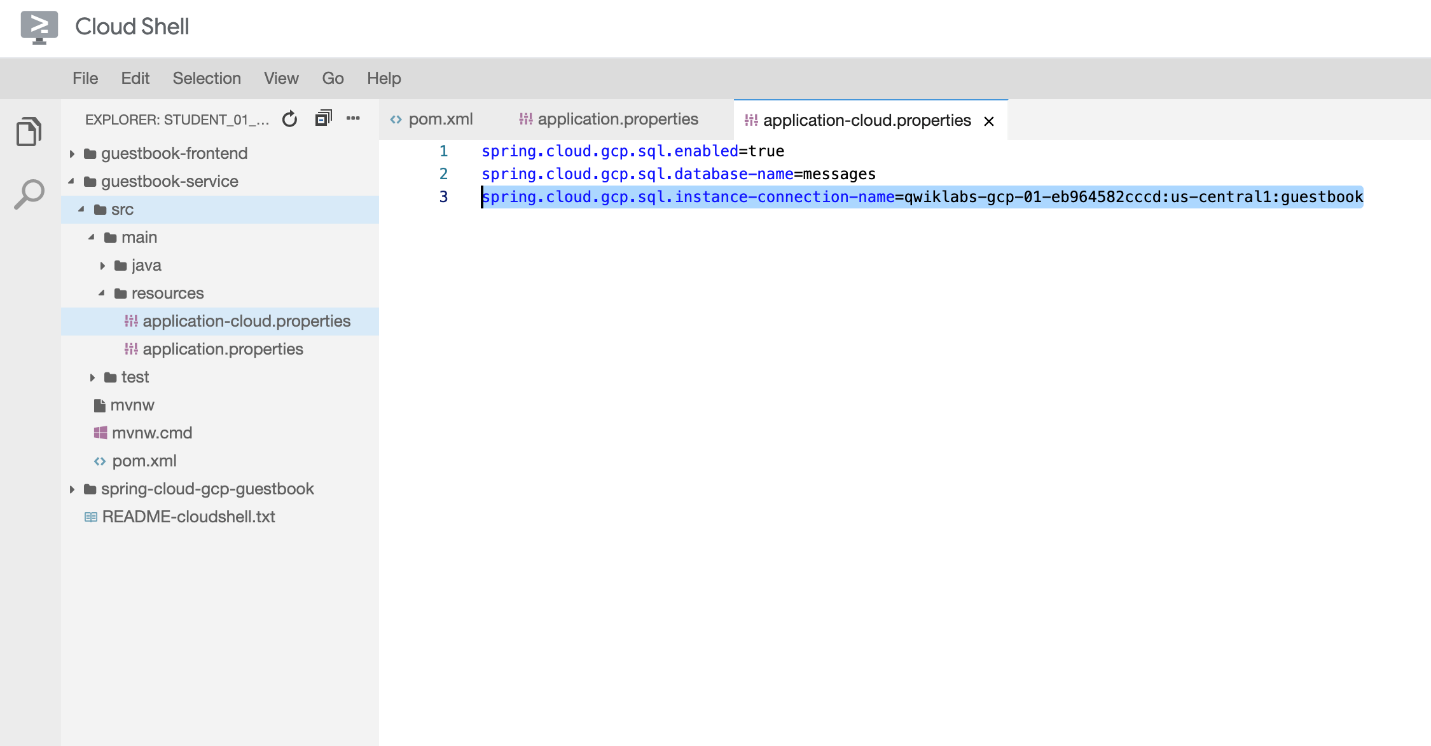
**spring.cloud.gcp.sql.enabled=true**

**spring.cloud.gcp.sql.database-name=messages**

**spring.cloud.gcp.sql.instance-connection-name=YOUR\_INSTANCE\_CONNECTION\_NAME**

1. Replace the YOUR\_INSTANCE\_CONNECTION\_NAME placeholder with the full connection name string returned in step 1 of this task.

If you worked in the Cloud Shell code editor, your screen should look like the following screenshot.



Configure the connection pool

You use the spring.datasource.\* configuration properties to configure the JDBC connection pool, as you do with other Spring Boot applications.

* Add the following property to guestbook-service/src/main/resources/application-cloud.properties that should still be open in the Cloud Shell code editor to specify the connection pool size:

**spring.datasource.hikari.maximum-pool-size=5**

Test the backend service running on Cloud SQL

You relaunch the backend service for the demo application in Cloud Shell, using the new cloud profile that configures the service to use Cloud SQL instead of the embedded HSQL database.

1. In Cloud Shell, change to the guestbook-service directory:

**cd ~/guestbook-service**

1. Run a test with the default profile and make sure there are no failures:

**./mvnw test**

1. Start the Guestbook Service with the cloud profile:

**./mvnw spring-boot:run \**

**-Dspring-boot.run.jvmArguments="-Dspring.profiles.active=cloud"**

1. During the application startup, validate that you see CloudSQL related connection logs.

The output should look similar to the following.

Output:

... First Cloud SQL connection, generating RSA key pair.

... Obtaining ephemeral certificate for Cloud SQL instance [springone17-sfo-1000:us-ce

... Connecting to Cloud SQL instance [...:us-west1:guestbook] on ...

... Connecting to Cloud SQL instance [...:us-west1:guestbook] on ...

... Connecting to Cloud SQL instance [...:us-west1:guestbook] on ...

...

1. In a new Cloud Shell tab, make a few calls using curl:

**curl -XPOST -H "content-type: application/json" \**

**-d '{"name": "Ray", "message": "Hello CloudSQL"}' \**

**http://localhost:8081/guestbookMessages**

1. You can also list all of the messages:

**curl http://localhost:8081/guestbookMessages**

1. Use the Cloud SQL client to validate that message records have been added to the database:

**gcloud sql connect guestbook**

1. Press Enter at the Enter password prompt.
2. Query the guestbook\_message table in the messages database:

**use messages;**

**select \* from guestbook\_message;**

The guestbook\_messages table now contains a record of the test message that you sent using curl in a previous step.

The output should look similar to the following.

Output:

+----+------+----------------+-----------+

| id | name | message | image\_uri |

+----+------+----------------+-----------+

| 1 | Ray | Hello Cloud SQL | NULL |

+----+------+----------------+-----------+

1 row in set (0.04 sec)

1. Close the Cloud SQL interactive client by executing the following command:

**exit**

Note: You will test the full application using the Cloud SQL enabled backend service application in the next lab.