Start Lab 02:00:00

JAVAMS02 Configuring and Connecting to Cloud SQL

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.

In this lab, you reconfigure the demo application to use an external database. The demo application is initially configured to use an embedded HSQL database. You 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 Platform (GCP) has a managed MySQL and PostgreSQL service called Cloud SQL.

In this lab, you 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 GCP, Cloud SQL provides a database infrastructure for applications running anywhere.

Objectives

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

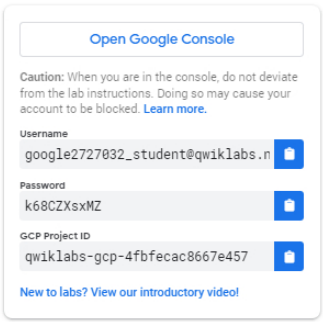
- Create a Cloud SQL instance, database, and table
- Use Spring to add Cloud SQL support to your application
- Configure an application profile for Cloud SQL
- Verify that an application is using Cloud SQL

Task 0. Lab Preparation

Access Qwiklabs

How to start your lab and sign in to the Console

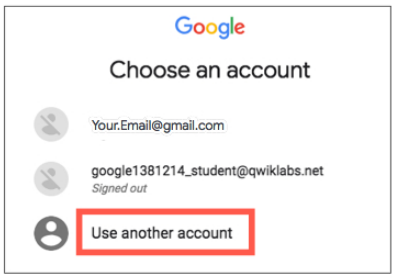
1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is a panel populated with the temporary credentials that you must use for this lab.



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.

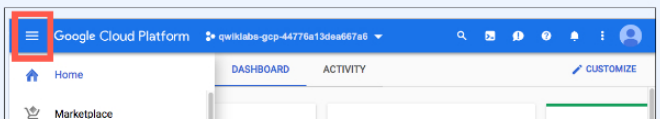
   Important: You must use the credentials from the Connection Details panel. Do not use your Qwiklabs credentials. If you have your own GCP account, do not use it for this lab (avoids incurring charges).

5. Click through the subsequent pages:

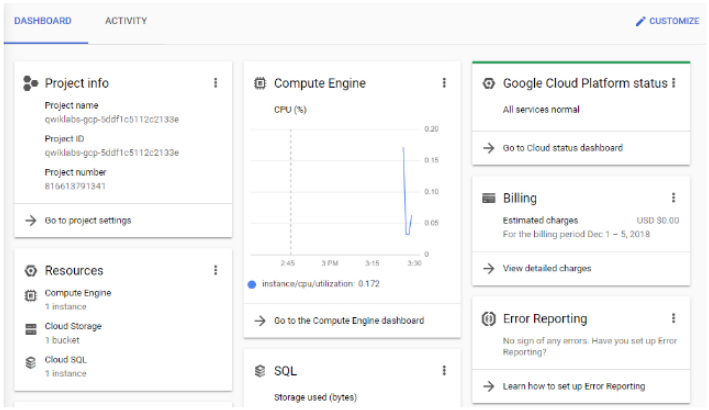
   - Accept the terms and conditions.
   - Do not add recovery options or two-factor authentication (because this is a temporary account).
   - Do not sign up for free trials.

   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.

1. Start by cloning the project repository:

```
git clone --single-branch --branch cloud-learning
https://github.com/saturnism/spring-cloud-gcp-guestbook.git
```

2. Now copy the relevant folders to your home directory. You're ready to go!

```
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 1. 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 the Cloud Shell enable Cloud SQL Administration API.

```
gcloud services enable sqladmin.googleapis.com
```

2. Confirm that Cloud SQL Administration API is enabled.

```
gcloud services list | grep sqladmin
```

3. List the Cloud SQL instances

3. List the Cloud SQL instances:

```
gcloud sql instances list
```

No instances are listed yet.

```
Listed 0 items.
```

Create a Cloud SQL instance

You create a new Cloud SQL instance.

1. Provision a new Cloud SQL instance.

```
gcloud sql instances create guestbook --region=us-central1
```

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

```
Creating Cloud SQL instance...done.
Created [...].
NAME       DATABASE_VERSION REGION       TIER              ADDRESS
STATUS
guestbook  MYSQL_5_6        us-central1  db-n1-standard-1  92.3.4.5
RUNNABLE
```

Create a database in the Cloud SQL instance

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

1. 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.

   This command temporarily allowlists the IP address for the connection.

```
gcloud sql connect guestbook
```

2. Press ENTER at the following prompt to leave the password empty for this lab.

   The root password is empty by default.

```
Allowlisting your IP for incoming connection for 5 minutes...done.
Connecting to database with SQL user [root].Enter password:
```

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.

3. 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.

```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| messages           |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.04 sec)
```

4. Switch to the `messages` database.

```
use messages;
```

5. 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)
);
```

6. Exit the MySQL management utility.

You return to the standard Cloud Shell user prompt.

```
exit
```

Task 2. 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 GCP 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 GCP 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 `guestbook-service/pom.xml` 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.

2. 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 3. 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 the Cloud Shell find the instance connection name by running the following command:

```
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-central1:guestbook
```

2. In the Cloud Shell code editor **create** an `application-cloud.properties` file in the `guestbook-service/src/main/resources` directory.

3. In the Cloud Shell code editor, open `guestbook-service/src/main/resources/application-cloud.properties` and add the following properties:

1. In the Cloud Shell find the instance connection name by running the following command:

```
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-central1:guestbook
```

2. In the Cloud Shell code editor **create** an `application-cloud.properties` file in the `guestbook-service/src/main/resources` directory.

3. In the Cloud Shell code editor, open `guestbook-service/src/main/resources/application-cloud.properties` and add the following properties:

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.

1. 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 the Cloud Shell change to the `guestbook-service` directory.

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

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

```
./mvnw test
```

3. Start the Guestbook Service with the cloud profile.

```
./mvnw spring-boot:run \
  -Dspring-boot.run.jvmArguments="-Dspring.profiles.active=cloud"
```

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

```
… 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-central1:guestbook] on …
… Connecting to Cloud SQL instance [...:us-central1:guestbook] on …
… Connecting to Cloud SQL instance [...:us-central1:guestbook] on …
...
```

5. 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
```

6. You can also list all the messages:

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

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

```
gcloud sql connect guestbook
```

8. Press ENTER at the Enter password prompt.

```
Allowlisting your IP for incoming connection for 5 minutes...done.
Connecting to database with SQL user [root].Enter password:
```

9. 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.

```
+----+------+----------------+-----------+
| id | name | message        | image_uri |
+----+------+----------------+-----------+
|  1 | Ray  | Hello Cloud SQL | NULL      |
+----+------+----------------+-----------+
1 row in set (0.04 sec)
```

9. Close the Cloud SQL interactive client.

```
exit
```

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

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.