**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

content\_copy

1. 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:

content\_copy

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;

content\_copy

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)content\_copy

1. Switch to the messages database.

use messages;

content\_copy

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)

);

content\_copy

1. Exit the MySQL management utility.

You return to the standard Cloud Shell user prompt.

exit

content\_copy

**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.

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>

content\_copy

**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

content\_copy

**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)'

content\_copy

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:guestbookcontent\_copy

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

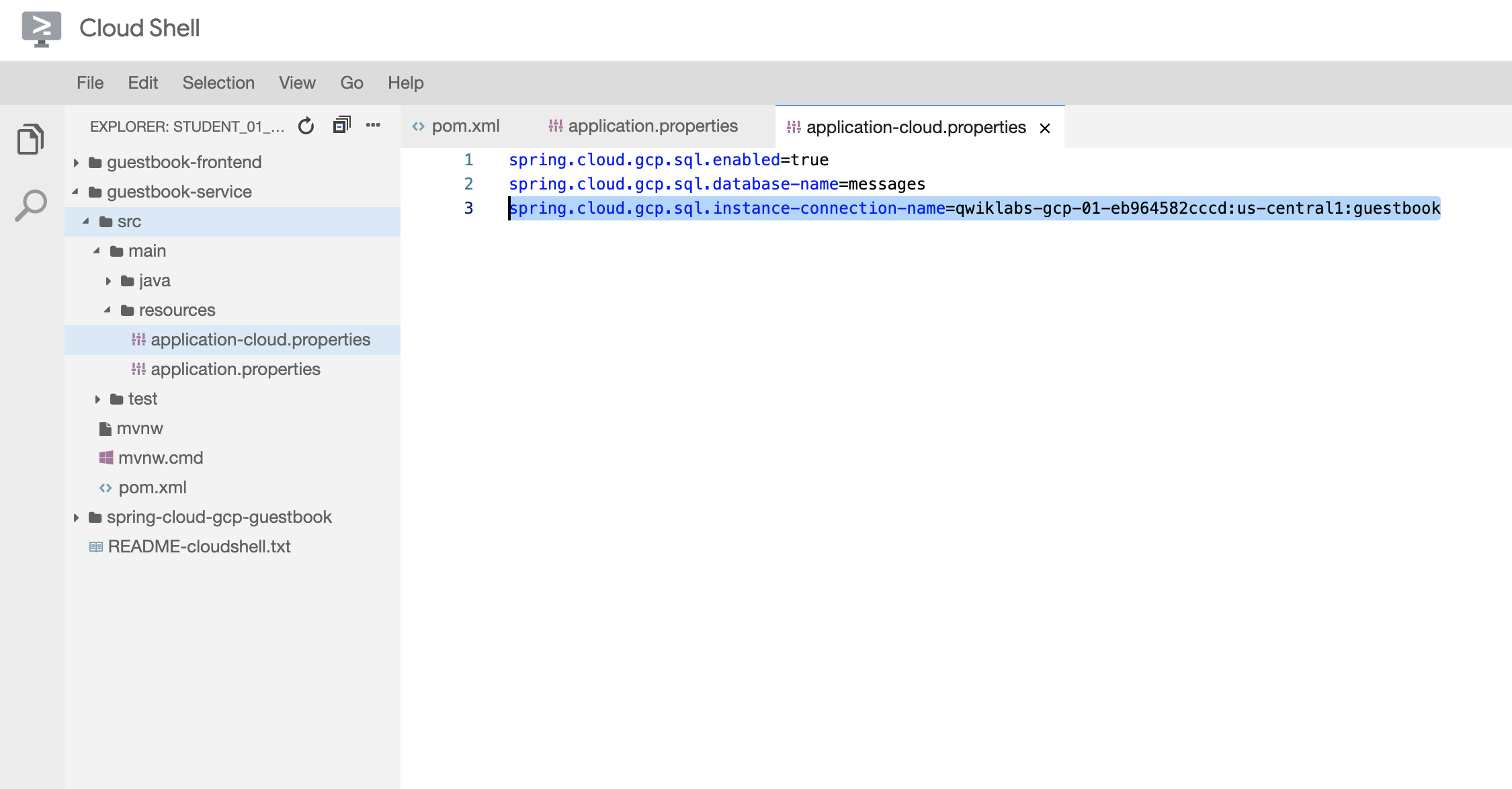
spring.cloud.gcp.sql.database-name=messages

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

content\_copy

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

If you worked in the Cloud Shell code editor, your screen looks 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.

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

content\_copy

**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

content\_copy

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

./mvnw test

content\_copy

1. Start the Guestbook Service with the cloud profile.

./mvnw spring-boot:run \

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

content\_copy

1. 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 …

...content\_copy

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

content\_copy

1. You can also list all the messages:

curl http://localhost:8081/guestbookMessages

content\_copy

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

gcloud sql connect guestbook

content\_copy

1. 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:content\_copy

1. Query the guestbook\_message table in the messages database.

use messages

select \* from guestbook\_message;

content\_copy

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)content\_copy

1. Close the Cloud SQL interactive client.

exit

content\_copy