

Cloud SQL with Terraform

GSP234



Overview

In this hands-on lab you will learn how to create Cloud SQL instances with Terraform, then set up the Cloud SQL Proxy, testing the connection with a MySQL client.

Objectives

In this lab, you will:

- Create a Cloud SQL instance
- Install the Cloud SQL Proxy
- Test connectivity with MySQL client using Cloud Shell

Setup and Requirements

Before you click the Start Lab button

Read these instructions. Labs are timed and you cannot pause them. The timer, which starts when you click **Start Lab**, shows how long Google Cloud resources will be made available to you.

This Qwiklabs hands-on lab lets you do the lab activities yourself in a real cloud environment, not in a simulation or demo environment. It does so by giving you new, temporary credentials that you use to sign in and access Google Cloud for the duration of the lab.

What you need

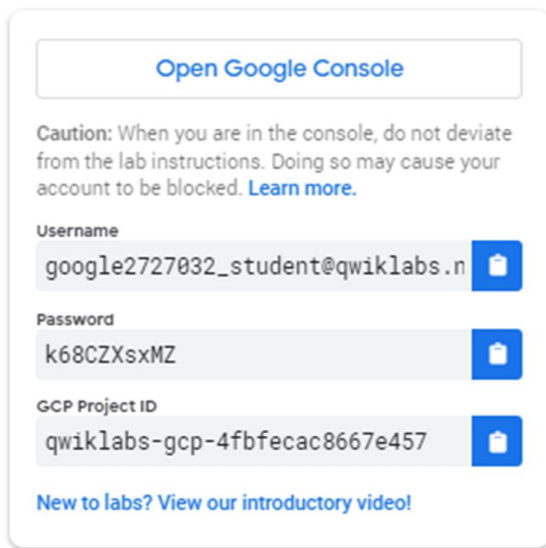
To complete this lab, you need:

- Access to a standard internet browser (Chrome browser recommended).
 - Time to complete the lab.
- Note:** If you already have your own personal Google Cloud account or project, do not use it for this lab.

Note: If you are using a Pixelbook, open an Incognito window to run this lab.


How to start your lab and sign in to the Google Cloud 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.](#)

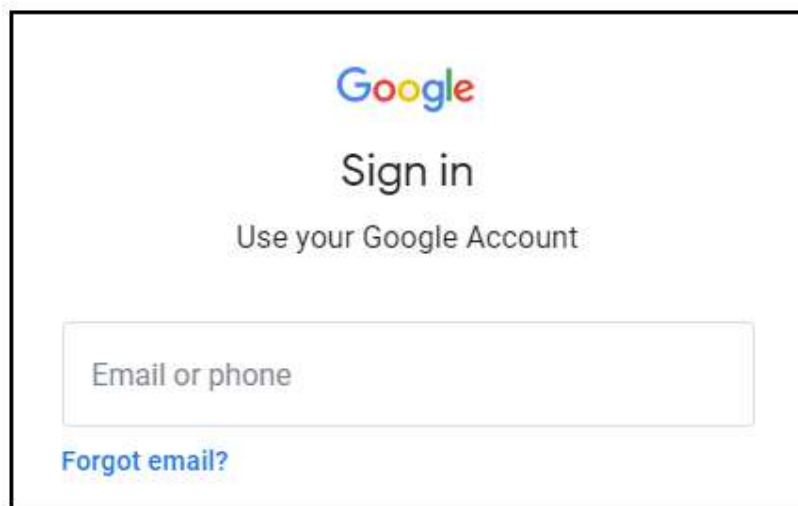
Username
google2727032_student@qwiklabs.n 

Password
k68CZXsxMZ 

GCP Project ID
qwiklabs-gcp-4fbfecac8667e457 

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

2. Copy the username, and then click **Open Google Console**. The lab spins up resources, and then opens another tab that shows the **Sign in** page.



Google

Sign in

Use your Google Account

Email or phone

[Forgot email?](#)

Tip: Open the tabs in separate windows, side-by-side.

If you see the **Choose an account** page, click **Use Another**



Account.

3. In the **Sign in** page, 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 Google Cloud account, do not use it for this lab (avoids incurring charges).

4. 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.

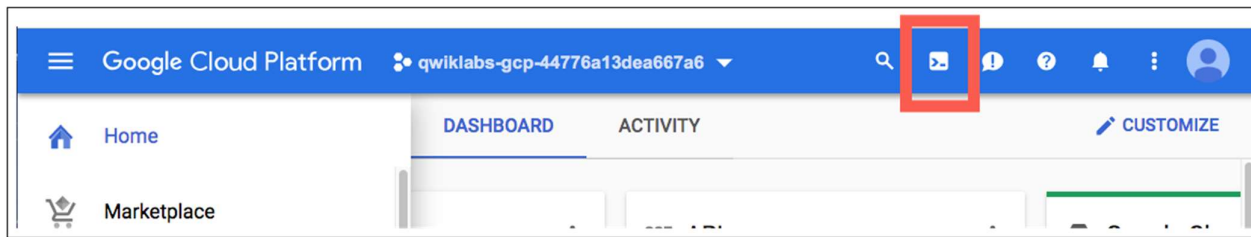
Note: You can view the menu with a list of Google Cloud Products and Services by clicking the **Navigation menu** at the top-left.



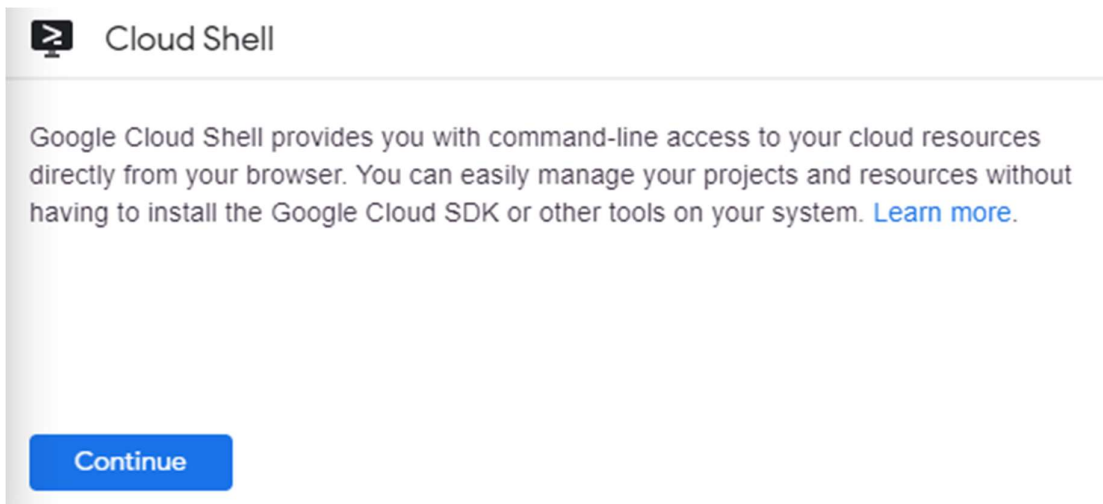
Activate Cloud Shell

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

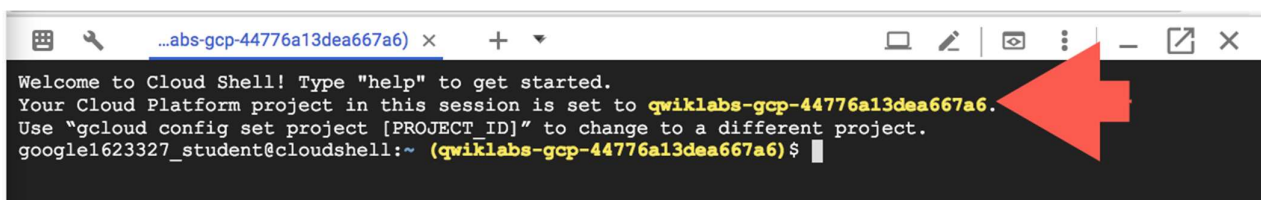
In the Cloud Console, in the top right toolbar, click the **Activate Cloud Shell** button.



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:
- <myaccount>@<mydomain>.com (active)
```

(Example output)

```
Credentialed accounts:  
- google1623327 student@gwiklabs.net
```

You can list the project ID with this command:

```
gcloud config list project
```

(Output)

```
[core]  
project = <project_ID>
```

(Example output)

```
[core]  
project = qwiklabs-gcp-44776a13dea667a6
```

For full documentation of `gcloud` see the [gcloud command-line tool overview](#).

Cloud SQL

Cloud SQL is a fully managed database service that makes it easy to set up, maintain, manage, and administer your relational databases on Google Cloud. You can use Cloud SQL with either [MySQL](#) or [PostgreSQL](#).

Download necessary files

1. Create a directory and fetch the required Terraform scripts from the Cloud Storage bucket with:

```
mkdir sql-with-terraform
cd sql-with-terraform
gsutil cp -r gs://spl5/gsp234/gsp234.zip .
```

2. Unzip the downloaded content.

```
unzip gsp234.zip
```

Understand the code

Take a look at the contents of the `main.tf` file:

```
cat main.tf
```

Example Output:

```
...
resource "google_sql_database_instance" "master" {
  name                = "example-mysql-${random_id.name.hex}"
  project              = var.project
  region              = var.region
  database_version     = var.database_version
  master_instance_name = var.master_instance_name

  settings {
    tier                        = var.tier
    activation_policy           = var.activation_policy
    authorized_gae_applications = var.authorized_gae_applications
    disk_autoresize            = var.disk_autoresize
    dynamic "backup_configuration" {
      for_each = [var.backup_configuration]
      content {
        binary_log_enabled = lookup(backup_configuration.value, "binary_log_enabled",
null)
        enabled            = lookup(backup_configuration.value, "enabled", null)
        start_time         = lookup(backup_configuration.value, "start_time", null)
      }
    }
    dynamic "ip_configuration" {
      for_each = [var.ip_configuration]
      content {
```

```

    ipv4_enabled    = lookup(ip_configuration.value, "ipv4_enabled", true)
    private_network = lookup(ip_configuration.value, "private_network", null)
    require_ssl     = lookup(ip_configuration.value, "require_ssl", null)

    dynamic "authorized_networks" {
      for_each = lookup(ip_configuration.value, "authorized_networks", [])
      content {
        expiration_time = lookup(authorized_networks.value, "expiration_time",
null)
        name            = lookup(authorized_networks.value, "name", null)
        value           = lookup(authorized_networks.value, "value", null)
      }
    }
  }
}
...

```

Run Terraform

The `terraform init` command is used to initialize a working directory containing Terraform configuration files.

This command performs several different initialization steps in order to prepare a working directory for use. This command is always safe to run multiple times, to bring the working directory up to date with changes in the configuration.

1. Run `terraform init`:

```
terraform init
```

The `terraform plan` command is an optional but recommended command and is used to create an execution plan. Terraform performs a refresh, unless explicitly disabled, and then determines what actions are necessary to achieve the desired state specified in the configuration files.

This command is a convenient way to check whether the execution plan for a set of changes matches your expectations without making any changes to real resources or to the state. For example, `terraform plan` might be run before committing a change to version control, to create confidence that it will behave as expected.

2. Run `terraform plan`:

```
terraform plan -out=tfplan
```

The optional `-out` argument can be used to save the generated plan to a file for later execution with `terraform apply`.

The `terraform apply` command is used to apply the changes required to reach the desired state of the configuration or the pre-determined set of actions generated by a `terraform plan` execution plan.

3. Apply the Terraform plan you just created:

```
terraform apply tfplan
```

This will take a little while to complete. Once complete you will see an output as below:

```
Apply complete! Resources: 5 added, 0 changed, 0 destroyed.

The state of your infrastructure has been saved to the path
below. This state is required to modify and destroy your
infrastructure, so keep it safe. To inspect the complete state
use the `terraform show` command.

State path: terraform.tfstate

Outputs:

generated_user_password = <sensitive>
instance_address = 35.232.204.44
instance_address_time_to_retire =
instance_name = example-mysql-6808
self_link =
https://www.googleapis.com/sql/v1beta4/projects/[PROJECT_ID]/instances/example-mysql-
6808
```

Test Completed Task

Click **Check my progress** to verify your performed task.

Create Cloud SQL instance using Terraform script.

Check my progress

Cloud SQL Proxy

What the proxy provides

The Cloud SQL Proxy provides secure access to your Cloud SQL Second Generation instances without having to [allowlist IP addresses](#) or [configure SSL](#).

Accessing your Cloud SQL instance using the Cloud SQL Proxy offers these advantages:

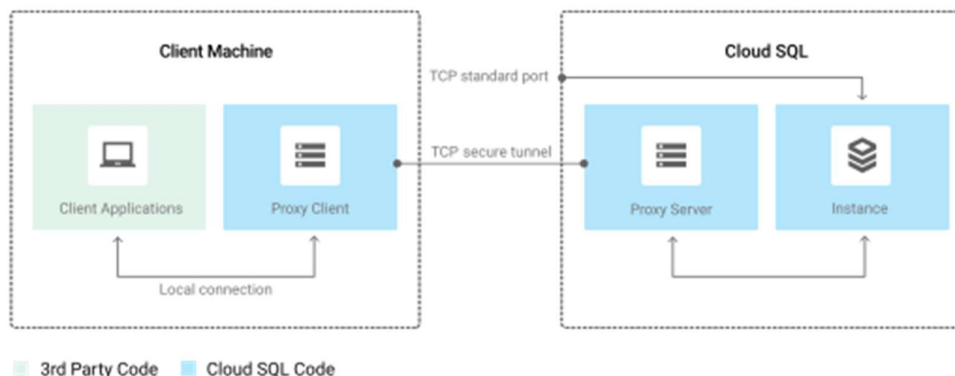
- **Secure connections:** The proxy automatically encrypts traffic to and from the database using TLS 1.2 with a 128-bit AES cipher; SSL certificates are used to verify client and server identities.
- **Easier connection management:** The proxy handles authentication with Cloud SQL, removing the need to provide static IP addresses.

Note: You do not need to use the proxy or configure SSL to connect to Cloud SQL from App Engine standard or the flexible environment. These connections use a "built-in" proxy implementation automatically.

How the Cloud SQL Proxy works

The Cloud SQL Proxy works by having a local client, called the proxy, running in the local environment. Your application communicates with the proxy with the standard protocol used by your database. The proxy uses a secure tunnel to communicate with its companion process running on the server.

The following diagram shows how the proxy connects to Cloud SQL:



Installing the Cloud SQL Proxy

1. Download the proxy:

```
wget https://dl.google.com/cloudsql/cloud_sql_proxy.linux.amd64 -O cloud_sql_proxy
```

2. Make the proxy executable:

```
chmod +x cloud_sql_proxy
```

You can install the proxy anywhere in your environment. The location of the proxy binaries does not impact where it listens for data from your application.

Proxy startup options

When you start the proxy, you provide it with the following sets of information:

- What Cloud SQL instances it should establish connections to
 - Where it will listen for data coming from your application to be sent to Cloud SQL
 - Where it will find the credentials it will use to authenticate your application to Cloud SQL
- The proxy startup options you provide determine whether it will listen on a TCP port or on a Unix socket. If it is listening on a Unix socket, it creates the socket at the location you choose; usually, the `/cloudsql/` directory. For TCP, the proxy listens on `localhost` by default.

Test connection to the database

1. Start by running the Cloud SQL proxy for the Cloud SQL instance:

```
export GOOGLE_PROJECT=$(gcloud config get-value project)

MYSQL_DB_NAME=$(terraform output -json | jq -r '.instance_name.value')
MYSQL_CONN_NAME="${GOOGLE_PROJECT}:us-central1:${MYSQL_DB_NAME}"
```

2. Run the following command:

```
./cloud_sql_proxy -instances=${MYSQL_CONN_NAME}=tcp:3306
```

Now you'll start another Cloud Shell tab by clicking on plus (+) icon. You'll use this shell to connect to the Cloud SQL proxy.

3. Navigate to `sql-with-terraform` directory:

```
cd ~/sql-with-terraform
```

4. Get the generated password for MySQL:

```
echo MYSQL_PASSWORD=$(terraform output -json | jq -r '.generated_user_password.value')
```

5. Test the MySQL connection:

```
mysql -udefault -p --host 127.0.0.1 default
```

6. When prompted, enter the value of `MYSQL_PASSWORD`, found in the output above, and press **Enter**.
7. You should successfully log into the MySQL command line. Exit from MySQL by typing **Ctrl + d**.
8. If you go back to the first Cloud Shell tab you'll see logs for the connections made to the Cloud SQL Proxy.

Test your Understanding

Below are multiple-choice questions to reinforce your understanding of this lab's concepts. Answer them to the best of your abilities.

Which command is used to create a Terraform execution plan?

☐ `terraform apply`

☐ `terraform plan`

☐ `terraform init`

☐ Submit

The Cloud SQL Proxy provides secure access to your Cloud SQL instances without having to allowlist IP addresses or configure SSL.

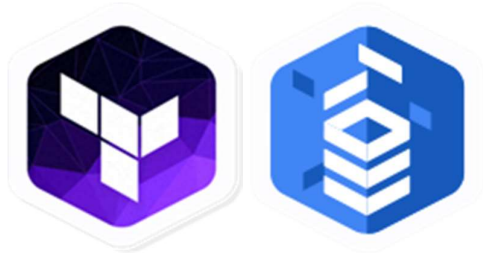
☐ True



False

Congratulations!

In this lab, you used Terraform to create a Cloud SQL instance and set up the Cloud SQL Proxy. You then tested the connection between the two with a MySQL client.



Finish Your Quest

This self-paced lab is part of the [Managing Cloud Infrastructure with Terraform](#) and [Cloud SQL](#) Quests. A Quest is a series of related labs that form a learning path. Completing this Quest earns you one of the badges above, to recognize your achievement. You can make your badge public and link to them in your online resume or social media account. Enroll in [Managing Cloud Infrastructure with Terraform](#) or [Cloud SQL](#) and get immediate completion credit if you've taken this lab. [See other available Qwiklabs Quests](#).

Take Your Next Lab

Continue your learning with [Using Vault on Compute Engine for Secret Management](#), or check out these suggestions:

- [Using a NAT Gateway with Kubernetes Engine](#)
- [HTTPS Content-Based Load balancer with Terraform](#)

Next Steps / Learn More

- Read about [how to manage Google Cloud projects with Terraform](#)

Google Cloud Training & Certification

...helps you make the most of Google Cloud technologies. [Our classes](#) include technical skills and best practices to help you get up to speed quickly and continue your learning journey. We offer fundamental to advanced level training, with on-demand, live, and virtual

options to suit your busy schedule. [Certifications](#) help you validate and prove your skill and expertise in Google Cloud technologies.

Manual Last Updated January 19, 2021

Lab Last Tested January 19, 2021

Copyright 2021 Google LLC All rights reserved. Google and the Google logo are trademarks of Google LLC. All other company and product names may be trademarks of the respective companies with which they are associated.