Connect to Cloud SQL from an Application in Kubernetes Engine

GSP449

Google Cloud Self-Paced Labs

Overview

This lab shows how easy it is to connect an application in Kubernetes Engine to a Cloud SQL instance using the Cloud SQL Proxy container as a sidecar container. You will deploy a Kubernetes Engine cluster and a Cloud SQL Postgres instance and use the Cloud SQL Proxy container to allow communication between them.
While this lab is focused on connecting to a Cloud SQL instance with a Cloud SQL Proxy container, the concepts are the same for any Google Cloud managed service that requires API access.

This lab was created by GKE Helmsman engineers to help you gain a better understanding of Cloud SQL through a proxy container. You can view this demo on on Github here. We encourage any and all to contribute to our assets!
The key takeaways are:

- How to protect your database from unauthorized access by using an unprivileged service account on your Kubernetes Engine nodes.
- How to put privileged service account credentials into a container running on Kubernetes Engine.
- How to use the Cloud SQL Proxy to offload the work of connecting to your Cloud SQL instance and reduce your applications knowledge of your infrastructure.

Unprivileged service accounts

All Kubernetes Engine nodes are assigned the default Compute Engine service account. This service account is fairly high privilege and has access to many Google Cloud services. Because of the way the Cloud SDK is setup, software that you write will use the credentials assigned to the compute engine instance on which it is running. Since you don't want all of your containers to have the privileges that the default Compute Engine service account has, you need to make a least-privilege service account for your Kubernetes Engine nodes and then create more specific (but still least-privilege) service accounts for your containers.

Privileged service accounts in containers

The only two ways to get service account credentials are through:

1. Your host instance (which you don't want)
2. A credentials file

This lab will show you how to get the credentials file into your container running in Kubernetes Engine so your application has the privileges it needs.

Cloud SQL Proxy

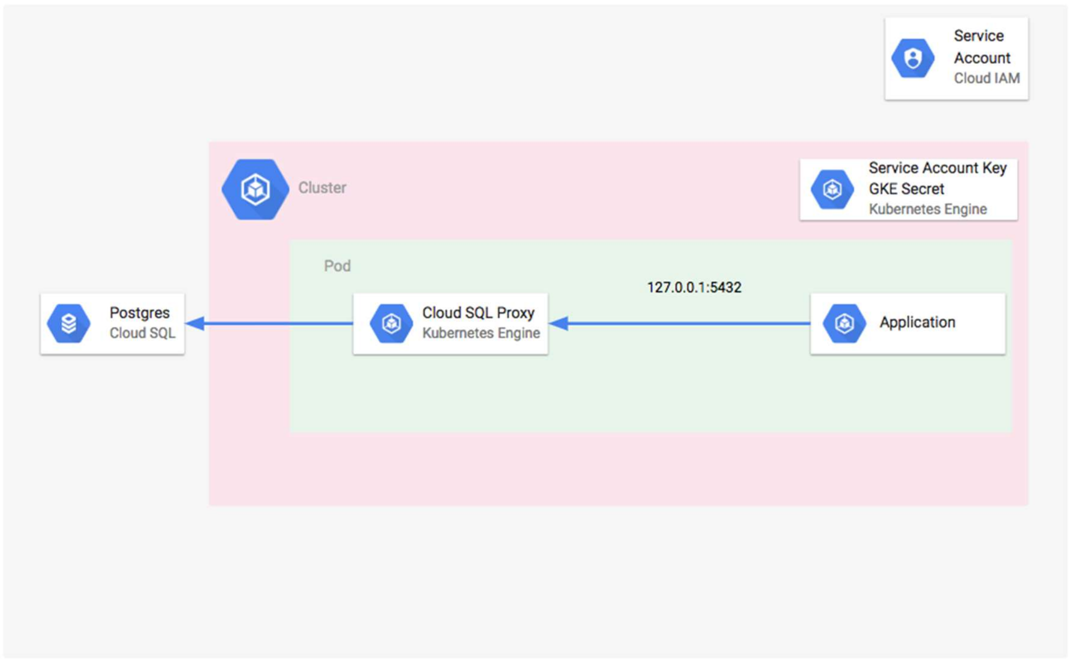
The Cloud SQL Proxy allows you to offload the burden of creating and maintaining a connection to your Cloud SQL instance to the Cloud SQL Proxy process. Doing this allows your application to be unaware of the connection details and simplifies your secret management. The Cloud SQL Proxy comes pre-packaged by Google as a Docker container that you can run alongside your application container in the same Kubernetes Engine pod.

Architecture

The application and its sidecar container are deployed in a single Kubernetes (k8s) pod running on the only node in the Kubernetes Engine cluster. The application communicates with the Cloud SQL instance via the Cloud SQL Proxy process listening on localhost.

The k8s manifest builds a single-replica Deployment object with two containers, pgAdmin and Cloud SQL Proxy. There are two secrets installed into the Kubernetes Engine cluster: the Cloud SQL instance connection information and a service account key credentials file, both used by the Cloud SQL Proxy containers Cloud SQL API calls.

The application doesn't have to know anything about how to connect to Cloud SQL, nor does it have to have any exposure to its API. The Cloud SQL Proxy process takes care of that for the application. It's important to note that the Cloud SQL Proxy container is running as a 'sidecar' container in the pod.



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

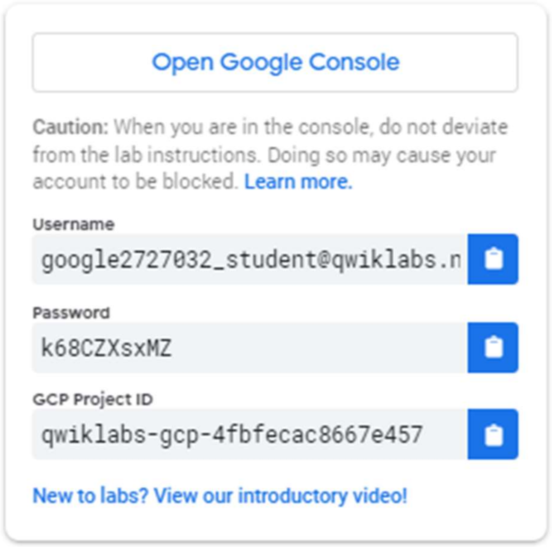
To complete this lab, you need:

- 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

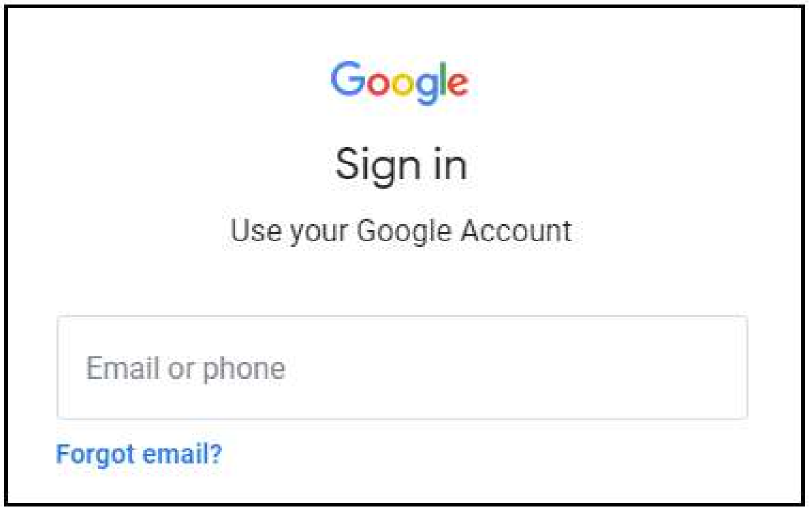
k68CZXsxMZ

GCP Project ID

qwiklabs-gcp-4fbfecac8667e457

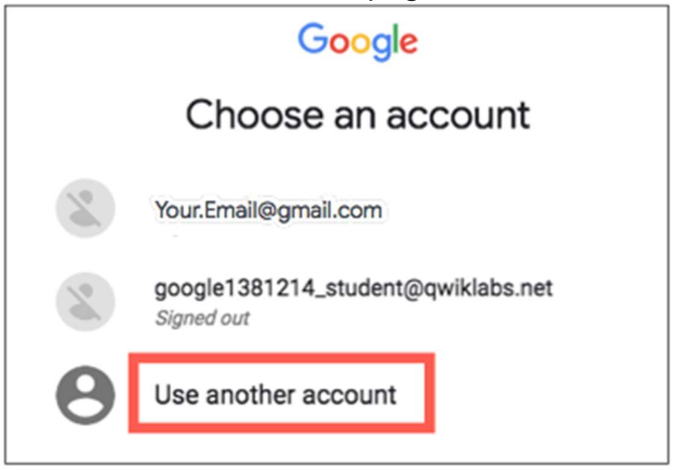
New to labs? View our introductory video!

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.



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.

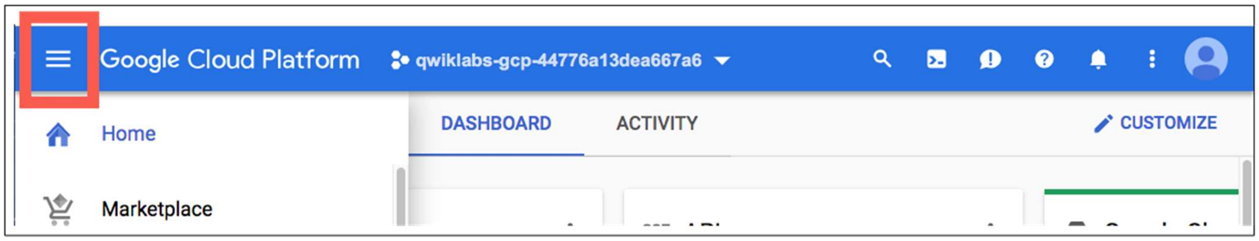
   Important: You must use the credentials from the Connection Details panel. Do not use your Qwiklabs credentials. If you have your own Google Cloud account, do not use it for this lab (avoids incurring charges).

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.

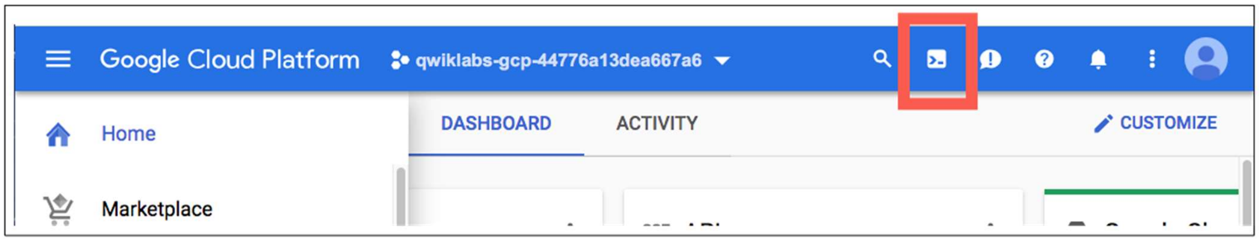
Note: You can view the menu with a list of Google Cloud Products and Services by clicking the **Navigation menu** at the top-
left.



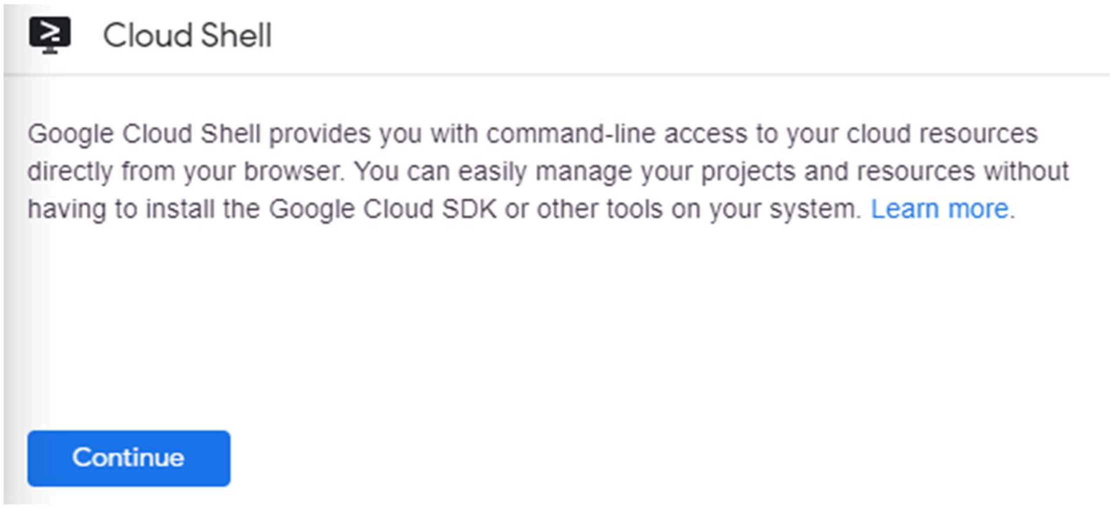
Activate Cloud Shell

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.

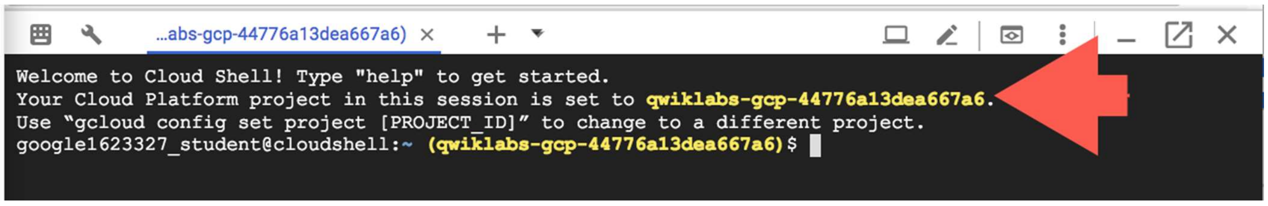


Click **Continue**.


Cloud Shell

Google Cloud Shell provides you with command-line access to your cloud resources directly from your browser. You can easily manage your projects and resources without having to install the Google Cloud SDK or other tools on your system. Learn more.

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:



```
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
```

(Output)

```
Credentialed accounts:
 - <myaccount>@<mydomain>.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 = <project_ID>
```

(Example output)

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

For full documentation of `gcloud` see the gcloud command-line tool overview.

Set your region and zone

Certain Compute Engine resources live in regions and zones. A region is a specific geographical location where you can run your resources. Each region has one or more zones.

Learn more about regions and zones and see a complete list in Regions & Zones documentation.

Run the following to set a region and zone for your lab (you can use the region/zone that's best for you):

```
gcloud config set compute/region us-central1
gcloud config set compute/zone us-central1-a
```

Copy the demo

```
gsutil cp gs://spls/gsp449/gke-cloud-sql-postgres-demo.tar.gz .
tar -xzvf gke-cloud-sql-postgres-demo.tar.gz
```

Go into the directory for this demo:

```
cd gke-cloud-sql-postgres-demo
```

Deployment

Deployment is fully automated. The script you will deploy takes the following parameters, in order:

- A username for your Cloud SQL instance
- A username for the pgAdmin console
- USER_PASSWORD - the password to login to the Postgres instance
- PG_ADMIN_CONSOLE_PASSWORD - the password to login to the pgAdmin UI
You can create any username for the Cloud SQL instance and use any email for the pgAdmin console; the example here uses "dbadmin" and your temporary student email.

Save your student account in a variable:

```
PG_EMAIL=$(gcloud config get-value account)
```

Run the following to deploy the script and create the 2 usernames; you will be asked to create a password for `dbadmin` and `$PG_EMAIL` (your student@qwiklabs.net account) in the output:

```
./create.sh dbadmin $PG_EMAIL
```

The passwords will be used again later in the lab; they don't need to be difficult.

During the deployment, `create.sh` will run the following scripts:

1. `enable_apis.sh` - enables the Kubernetes Engine API and Cloud SQL Admin API.
2. `postgres_instance.sh` - creates the Cloud SQL instance and additional Postgres user. Note that gcloud will timeout when waiting for the creation of a Cloud SQL instance so the script manually polls for its completion instead.
3. `service_account.sh` - creates the service account for the Cloud SQL Proxy container and creates the credentials file.
4. `cluster.sh` - Creates the Kubernetes Engine cluster.
5. `configs_and_secrets.sh` - creates the Kubernetes Engine secrets and configMap containing credentials and connection string for the Cloud SQL instance.
6. `pgadmin_deployment.sh` - creates the pgAdmin4 pod.

Deployment of the Cloud SQL instance can take up to 10 minutes.

Next, use load balancer to expose the pod in order to connect to the instance, then delete the services when finished to avoid unauthorized access.

Run the following to get the Pod ID:

```
POD_ID=$(kubectl --namespace default get pods -o name | cut -d '/' -f 2)
```

Expose the pod via load balancer:

```
kubectl expose pod $POD_ID --port=80 --type=LoadBalancer
```

Get the service IP address:

```
kubectl get svc
```

Output:

```
NAME                                   TYPE           CLUSTER-IP      EXTERNAL-IP
PORT(S)        AGE

kubernetes                             ClusterIP      <CLUSTER_IP>    <none>
443/TCP        96m

<SVC_NAME LoadBalancer   <CLUSTER_IP>    <SVC_IP>      80:31789/TCP   45m
```

Run the last command again until you see an external IP address for the pgAdmin service.
Open a new tab and connect to pgAdmin in your browser the using th pgAdmin <SVC_IP>:

```
http://<SVC_IP>
```

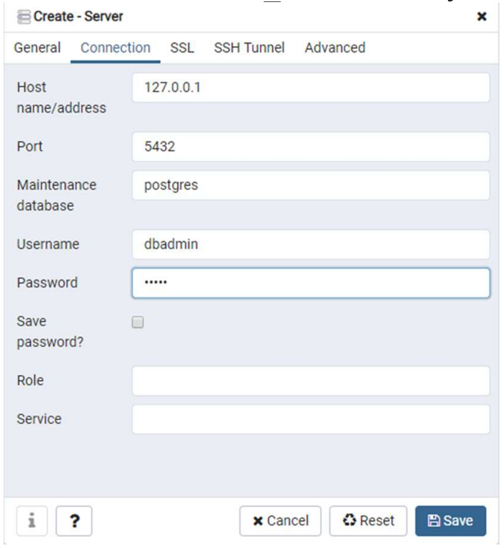
Sign in to the pgAdmin UI with the following:

- <PGADMIN_USERNAME> (your temporary student@qwiklabs.net account) in the "Email Address" field
- <PG_ADMIN_CONSOLE_PASSWORD> that you defined earlier

If you're unsure what your full student account is, run **gcloud config get-value account** in your Cloud Shell and copy the output.

In the pgAdmin console, click **Add New Server**.
On the **General** tab, give your server a name, then click on the **Connections** tab.
Use the <DATABASE_USER_NAME>(dbadmin) and <USER_PASSWORD> you created earlier to connect to 127.0.0.1:5432:

- **Host name:** `127.0.0.1`
- **Username:** `<DATABASE_USER_NAME>`(dbadmin)
- **Password:** `<USER_PASSWORD>` you created



Host name/address	127.0.0.1
Port	5432
Maintenance database	postgres
Username	dbadmin
Password	•••••
Save password?	☐
Role	
Service	

Click **Save**.

Test Completed Task

Click **Check my progress** to verify your performed task. If you have successfully created required resources with the fully automated deployment, you will see an assessment score.

Validation

Validation is fully automated. The validation script checks for the existence of the Cloud SQL instance, the Kubernetes Engine cluster, and the running pod. All of these resources should exist after the deployment script completes.

In Cloud Shell, validate these three deployments by executing:

```
make validate
```

The script takes the following parameters, in order:

- INSTANCE_NAME - the name of the existing Cloud SQL instance

A successful output will look like this:

```
Cloud SQL instance exists
GKE cluster exists
pgAdmin4 Deployment object exists
```

Teardown

Teardown is fully automated. The teardown script deletes every resource created in the deployment script.

In order to teardown, run:

```
make teardown
```

The script takes the following parameters, in order:

- INSTANCE_NAME - the name of the existing Cloud SQL instance

`teardown.sh` will run the following scripts:

1. `delete_resources.sh` - deletes everything but the Cloud SQL instance
2. `delete_instance.sh` - deletes the Cloud SQL instance

Troubleshooting in your own environment

When creating a Cloud SQL instance you get the error:

```
is the subject of a conflict: The instance or operation is not
in an appropriate state to handle the request.
```

Resolution

You cannot reuse an instance name for up to a week after you have deleted an instance https://cloud.google.com/sql/docs/mysql/delete-instance.

Congratulations

Finish Your Quest

This self-paced lab is part of the Qwiklabs Google Kubernetes Engine Best Practices Quest. A Quest is a series of related labs that form a learning path. Completing this Quest earns you the badge above, to recognize your achievement. You can make your badge (or badges) public and link to them in your online resume or social media account. Enroll in this Quest and get immediate completion credit if you've taken this lab. See other available Qwiklabs Quests.
Next Steps / Learn More
Kubernetes Installation instructions to deploy a containerized application for multiple platforms.

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.

Manual Last Updated: November 10, 2020
Lab Last Tested: November 10, 2020