Connect an App to a Cloud SQL for PostgreSQL Instance

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.

1. Click **Activate Cloud Shell** at the top of the Google Cloud console.

2. Click through the following windows:

   - Continue through the Cloud Shell information window.
   - Authorize Cloud Shell to use your credentials to make Google Cloud API calls.

When you are connected, you are already authenticated, and the project is set to your **Project_ID**, `qwiklabs-gcp-03-fa17659dc783`. The output contains a line that declares the **Project_ID** for this session:

```
Your Cloud Platform project in this session is set to qwiklabs-gcp-03-
fa17659dc783
```

`gcloud` is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.

3. (Optional) You can list the active account name with this command:

```
gcloud auth list
```

4. Click **Authorize**.

Output:

```
ACTIVE: *
ACCOUNT: student-03-bce0c0f35b9d@qwiklabs.net

To set the active account, run:
    $ gcloud config set account `ACCOUNT`
```

5. (Optional) You can list the project ID with this command:

```
gcloud config list project
```

Output:

```
[core]
project = qwiklabs-gcp-03-fa17659dc783
```

Note: For full documentation of `gcloud`, in Google Cloud, refer to the gcloud CLI overview guide.

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.

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

Run the following gcloud commands in Cloud Shell to set the default region and zone for your lab:

```
gcloud config set compute/zone "us-central1-c"
export ZONE=$(gcloud config get compute/zone)

gcloud config set compute/region "us-central1"
export REGION=$(gcloud config get compute/region)
```

Task 1. Initialize APIs and create a Cloud IAM service account

To complete this task you must initialize the APIs and create an IAM service account that will be used to allow your application to connect to the Cloud SQL database.

Enable the APIs

You must enable the required APIs for this lab. You will build and push a container to the Artifact Registry in a later task, so you must enable the Artifact Registry API first.

1. In Cloud Shell, run the following command to enable the Artifact Registry API:

```
gcloud services enable artifactregistry.googleapis.com
```

Create a Service Account for Cloud SQL

You need to configure IAM service account credentials for the application that you will deploy later. The service account must be bound to a role that allows it to create and access Cloud SQL databases.

1. In Cloud Shell, create a Service Account and bind it to the Cloud SQL admin role in the lab project:

```
export PROJECT_ID=$(gcloud config list --format 'value(core.project)')
export CLOUDSQL_SERVICE_ACCOUNT=cloudsql-service-account

gcloud iam service-accounts create $CLOUDSQL_SERVICE_ACCOUNT --
project=$PROJECT_ID

gcloud projects add-iam-policy-binding $PROJECT_ID \
--
member="serviceAccount:$CLOUDSQL_SERVICE_ACCOUNT@$PROJECT_ID.iam.gservi
ceaccount.com" \
--role="roles/cloudsql.admin"
```

In Cloud Shell, create and export keys to a local file:

```
gcloud iam service-accounts keys create $CLOUDSQL_SERVICE_ACCOUNT.json
\
    --iam-
account=$CLOUDSQL_SERVICE_ACCOUNT@$PROJECT_ID.iam.gserviceaccount.com \
    --project=$PROJECT_ID
```

The file will be saved to your home folder in Cloud Shell.

Task 2. Deploy a lightweight GKE application

In this task you will create a Kubernetes cluster and deploy a lightweight Google Kubernetes Engine (GKE) application on that cluster. You will configure the application to have access to the supplied Cloud SQL instance.

The application provided is a simple Flask-SQLAlchemy web application called gMemegen. It creates memes by supplying a set of photographs and capturing header and footer text, storing them in the database and rendering the meme to a local folder. It runs on a single pod with two containers; one for the application and one for the Cloud SQL Auth Proxy deployed in the side-car pattern.

A load balancer will marshal requests between the app and the database through the side-car. This load balancer will expose an external Ingress IP address through which you will access the app in your browser.

Create a Kubernetes cluster

In this step, you will create a minimal Kubernetes cluster. The cluster will take a couple of minutes to be deployed.

1. In Cloud Shell, create a minimal Kubernetes cluster as follows:

```
ZONE=us-central1-c
gcloud container clusters create postgres-cluster \
--zone=$ZONE --num-nodes=2
```

Create Kubernetes secrets for database access

In this step you will create a pair of Kubernetes secrets containing the credentials that are needed to connect to the Cloud SQL instance and database.

1. In Cloud Shell, run the following commands to create the secrets:

```
kubectl create secret generic cloudsql-instance-credentials \
--from-file=credentials.json=$CLOUDSQL_SERVICE_ACCOUNT.json

kubectl create secret generic cloudsql-db-credentials \
--from-literal=username=postgres \
--from-literal=password=supersecret! \
--from-literal=dbname=gmemegen_db
```

Download and build the GKE application container

Before you can deploy the gMemegen application to your GKE cluster you need to build the container and push it to a repository.

1. In Cloud Shell, download the provided application code and change to the application directory:

```
gsutil -m cp -r gs://spls/gsp919/gmemegen .
cd gmemegen
```

2. Create environment variables for the region, Project ID and Artifact Registry repository:

```
export REGION=us-central1
export PROJECT_ID=$(gcloud config list --format 'value(core.project)')
export REPO=gmemegen
```

3. Configure Docker authentication for the Artifact Registry:

```
gcloud auth configure-docker ${REGION}-docker.pkg.dev
```

- Enter `Y` if you are asked for confirmation.

4. Create the Artifact Registry repository:

```
gcloud artifacts repositories create $REPO \
    --repository-format=docker --location=$REGION
```

Build a local Docker image:

```
docker build -t ${REGION}-
docker.pkg.dev/${PROJECT_ID}/gmemegen/gmemegen-app:v1 .
```

For the purposes of this lab, you may ignore the warning about running 'pip' as the 'root' user, although you should note that, in general, especially when working on your local machine, it is best practice to use a virtual environment.

6. Push the image to the Artifact Registry:

```
docker push ${REGION}-docker.pkg.dev/${PROJECT_ID}/gmemegen/gmemegen-
app:v1
```

Configure and deploy the GKE application

You must modify the Kubernetes deployment manifest for the gMemegen application to point at the correct container and configure the Cloud SQL Auth Proxy side-car with the connection string for the Cloud SQL PostgreSQL instance.

The instructions explain how to edit the file using the Cloud Shell Editor, but if you prefer you can use another editor, such as `vi` or `nano`, from Cloud Shell for these steps.

1. On the Cloud Shell menu bar, click **Open Editor** to open the Cloud Shell Editor.

2. Navigate the **Explorer** panel on the left hand side, expanding the `gmemegen` folder and then selecting `gmemegen_deployment.yaml` to edit the file.

3. On **line 33**, in the `image` attribute, replace `${REGION}` with `us-central1` and `${PROJECT_ID}` with `qwiklabs-gcp-03-fa17659dc783`. The line should now read:

```
image: us-central1-docker.pkg.dev/qwiklabs-gcp-03-
fa17659dc783/gmemegen/gmemegen-app:v1
```

4. On **line 60**, replace `${REGION}` with `us-central1` and `${PROJECT_ID}` with `qwiklabs-gcp-03-fa17659dc783`. The line should now read:

```
-instances=qwiklabs-gcp-03-fa17659dc783:us-central1:postgres-
gmemegen=tcp:5432
```

To confirm that the connection name is correct, in the Cloud Console, navigate to **Databases** > **SQL**, select the `postgres-gmemegen` instance and compare with the **Connection name** in the **Overview** pane. A valid connection name is of the format `PROJECT_ID:REGION:CLOUD_SQL_INSTANCE_ID`.

5. Save your changes by selecting **File** > **Save** from the Cloud Shell Editor menu.

6. In the Cloud console click the **Open Terminal** to re-open Cloud Shell. You may need to resize the Terminal window by dragging down the handle at the centre top of the menu bar, in order to see your Cloud Console window above.

7. In Cloud Shell, deploy the application by running the following command:

```
kubectl create -f gmemegen_deployment.yaml
```

8. In Cloud Shell, check that the deployment was successful by running the following command:

```
kubectl get pods
```

It may take a minute or so for the pods to start up, because they need to pull the image from the repository. Repeat the above command until you see a pod, with 2 containers, with status `Running`.

Task 3. Connect the GKE application to an external load balancer

In this task you will create a load balancer to marshal requests between the containers in your GKE pods and access the application using its external IP address from your browser.

Create a load balancer to make your GKE application accessible from the web

In this step you will create a Kubernetes load balancer service that will provide your application with a public IP address.

1. In Cloud Shell, run the following command to create a load balancer for the application:

```
kubectl expose deployment gmemegen \
    --type "LoadBalancer" \
    --port 80 --target-port 8080
```

Test the application to generate some data

In this step you will access the gMemegen application from your web browser.

The application has a very simple interface. It launches to the application home page, which displays 6 candidate images for making memes. You can select an image by clicking on it.

The **Create Meme** page is displayed, where you enter two items of text, to be displayed at the top and bottom of the image. Clicking **Submit** renders the meme and displays it. The interface provides no navigation from the completed meme page. You will have to use the browser's back button to return to the home page.

There are two other pages, **Recent** and **Random**, which display a set of recently generated memes and a random meme, respectively. Generating memes and navigating the UI will generate database activity which you can view in the logs as described below.

Wait for the load balancer to expose an external IP, which you can retrieve as follows:

1. In Cloud Shell, copy the external IP address attribute of the `LoadBalancer Ingress` from the output of:

```
kubectl describe service gmemegen
```

Output:

```
Name:                     gmemegen
Namespace:                default
Labels:                   app=gmemegen
Annotations:              <none>
Selector:                 app=gmemegen
Type:                     LoadBalancer
IP Families:              <none>
IP:                       10.3.240.201
IPs:                      10.3.240.201
LoadBalancer Ingress:     34.67.122.203
Port:                     <unset>  80/TCP
TargetPort:               8080/TCP
NodePort:                 <unset>  31837/TCP
Endpoints:                10.0.0.7:8080
Session Affinity:         None
External Traffic Policy:  Cluster
Events:
  Type    Reason                Age   From                Message
  ----    ------                ----  ----                -------
  Normal  EnsuringLoadBalancer  85s   service-controller  Ensuring load
balancer
  Normal  EnsuredLoadBalancer   36s   service-controller  Ensured load
balancer
</unset></unset></none></none>
```

It will take a minute or so for the `LoadBalancer Ingress` attribute to be included in the output (see above), so repeat the command until it is there before performing the next step.

2. In a browser, navigate to the load balancer's Ingress IP address.

You can create a clickable link to the external IP address of the load balancer in Cloud Shell using the following commands:

```
export LOAD_BALANCER_IP=$(kubectl get svc gmemegen \
-o=jsonpath='{.status.loadBalancer.ingress[0].ip}' -n default)
echo gMemegen Load Balancer Ingress IP: http://$LOAD_BALANCER_IP’
```

4. Click the link in Cloud Shell and you will see the gMemegen application running in a new tab in your browser.

5. Create a meme as follows:

- On the **Home** page, click on one of the presented images.
- Enter text in the **Top** and **Bottom** text boxes.
- Click the **Submit** button.

Your new meme is displayed.

6. To create more memes, use the browser's back button to navigate to the home page.

7. To view existing memes, click **Recent** or **Random** in the application menu. (Note that **Random** opens a new browser tab)

8. In Cloud Shell, view the application's activity by running the following:

```
POD_NAME=$(kubectl get pods --output=json | jq -r
".items[0].metadata.name")
kubectl logs $POD_NAME gmemegen | grep "INFO"
```

This queries the logs from the `gmemegen` container and will display the activity of the application on the pod, including the SQL statements, which are logged to stderr as they are executed.

Task 4. Verify full read/write capabilities of application to database

In this task you will verify that the application is able to write to and read from the database.

Connect to the database and query an application table

In this step you will connect to the Cloud SQL instance by running **PL/SQL** in Cloud Shell.

1. In Google Cloud Console, navigate to **Databases** > **SQL** and select the `postgres-gmemegen` instance.

2. In the **Overview** pane , scroll down to **Connect to this instance** and click the **Open Cloud Shell** button.

3. Run the auto-populated command in Cloud Shell.

4. When prompted, enter the password: `supersecret!`

5. At the `postgres=>` prompt enter the following command to select the gmemegen_db database:

```
\c gmemegen_db
```

6. When prompted, enter the password: `supersecret!`

7. At the `gmemegen_db=>` prompt enter:

```
select * from meme;
```

Copied!

content_copy

This will display a row for each meme you have generated through the gMemegen app.