Connecting to Production Database and Updating User Permissions

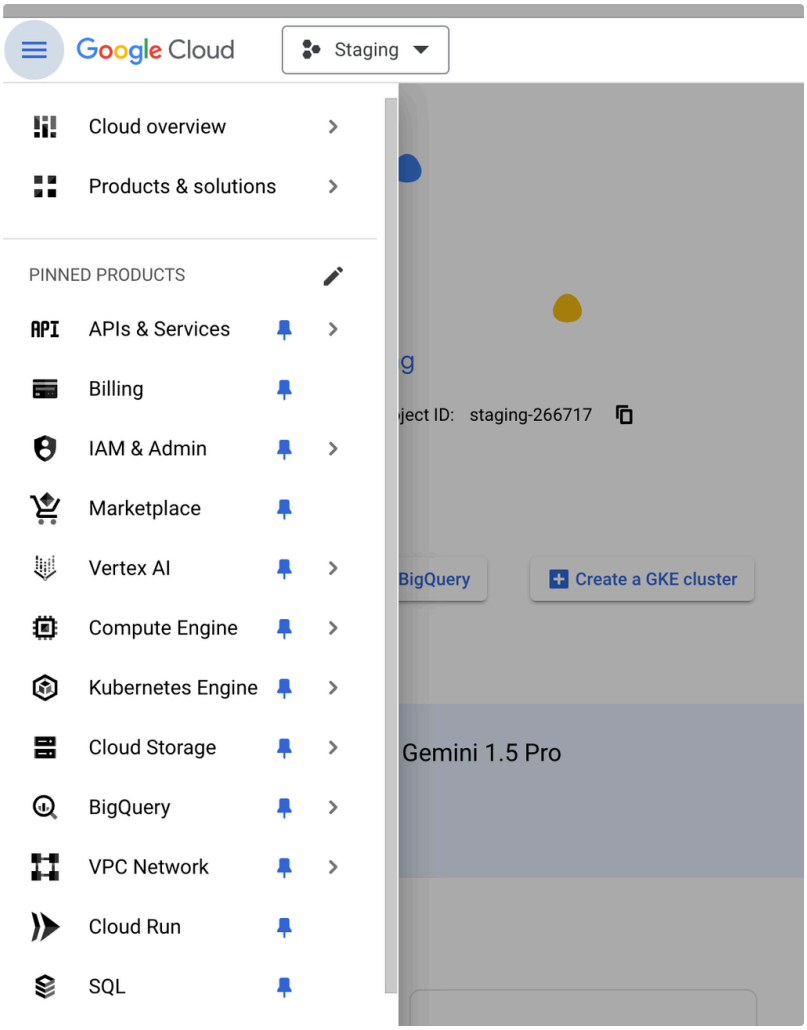
This Video walked through the addition of a user to a raft database

Prerequisites

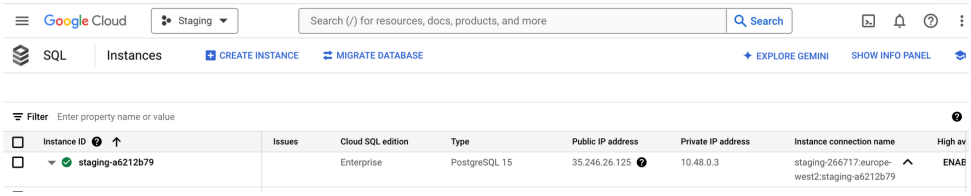
1. Download and install the Cloud SQL Proxy tool
   Connect using the Cloud SQL Auth Proxy | Cloud SQL for PostgreSQL | Google Cloud
   A good addition would to be to place the executable in your PATH eg `sudo mv ~/Downloads/cloud-sql-proxy /usr/local/bin`
2. Install a SQL client (e.g., DBeaver)
   Download

Connecting to the Database

1. Obtain the instance connection name from Google Cloud SQL
   Here I will walk through using staging project https://console.cloud.google.com/sql/instances?project=staging-266717
   - Firstly gain access to the GCP Console and then select SQL from the sidebar



- You will then see the databases available to your user



- The information required is the Instance Connection Name > Format: `<project>:<region>:<instance-name>.`
  In our case we will use `staging-266717:europe-west2:staging-a6212b79`

2. Run the Cloud SQL Proxy:
   `./cloud_sql_proxy <instance-connection-name> -p 5432 -g`
   The ./ is not needed if you have added the app to your PATH so
   `cloud_sql_proxy staging-266717:europe-west2:staging-a6212b79 -p 5432 -g`

3. To set up and connect via DBeaver we require the database login details:
   As we are using a proxy
   - Host: localhost
     Port is dependent on setting and DB type
   - Port: 5432 (or as specified in the proxy command)
     Our standard database name
   - Database: vector
     Our standard User Name

- Username: vector
- Password: Retrieve from Kubernetes secrets (see below)

Retrieving the Database Password

1. Connect to the appropriate Kubernetes cluster

   You will need `kubectl` installed and `gke-gcloud-auth-plugin`

   Install kubectl and configure cluster access | Google Kubernetes Engine (GKE) | Google Cloud

2. Auth to the cluster, here we are auth to the staging cluster

   `gcloud container clusters get-credentials staging --zone europe-west2-a --project staging-266717`

3. Run the following command:

   `pass=$(kubectl get secret postgres-secrets -o jsonpath='{.data.POSTGRES_PASSWORD}'); echo \`echo $pass | base64 -d\``

   This command does the following:

   a. Retrieves the POSTGRES_PASSWORD from the "postgres-secrets" Kubernetes secret.

   b. Stores the base64 encoded password in the `pass` variable.

   c. Decodes the base64 encoded password.

   d. Prints the decoded password.

   Breaking this down step by step:

   e. `kubectl get secret postgres-secrets`: This part retrieves the Kubernetes secret named "postgres-secrets".

      i. `kubectl get secrets` will allow you to see all secrets

   f. `-o jsonpath='{.data.POSTGRES_PASSWORD}'`: This flag specifies that we want to output the result using JSONPath. It extracts the value of the POSTGRES_PASSWORD field from the secret's data.

   g. `pass=$()`: This assigns the output of the command inside the parentheses to the variable `pass`.

   h. `echo $pass | base64 -d`: This part takes the value stored in `pass` (which is base64 encoded) and decodes it.

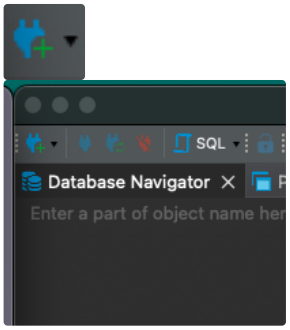
   i. The outer `echo ``` (backticks) execute the command inside and print its output.

> if you experience any issues with connection to the cluster ie
>
> `E0717 13:59:15.766932   47243 memcache.go:265] couldn't get current server API group list: Get "https://35.189.75.74/api?timeout=32s": net/http: TLS handshake timeout`
>
> Ensure you are on the VPN

Creating a DBeaver connection

1. open DBeaver
2. On first time opening this without any Databases configured, this will automatically ask to set one up.

If not, the small connection icon in the top left will open this up



3. In this case we are connection to a POSTGRES DB and so will click this icon and click NEXT

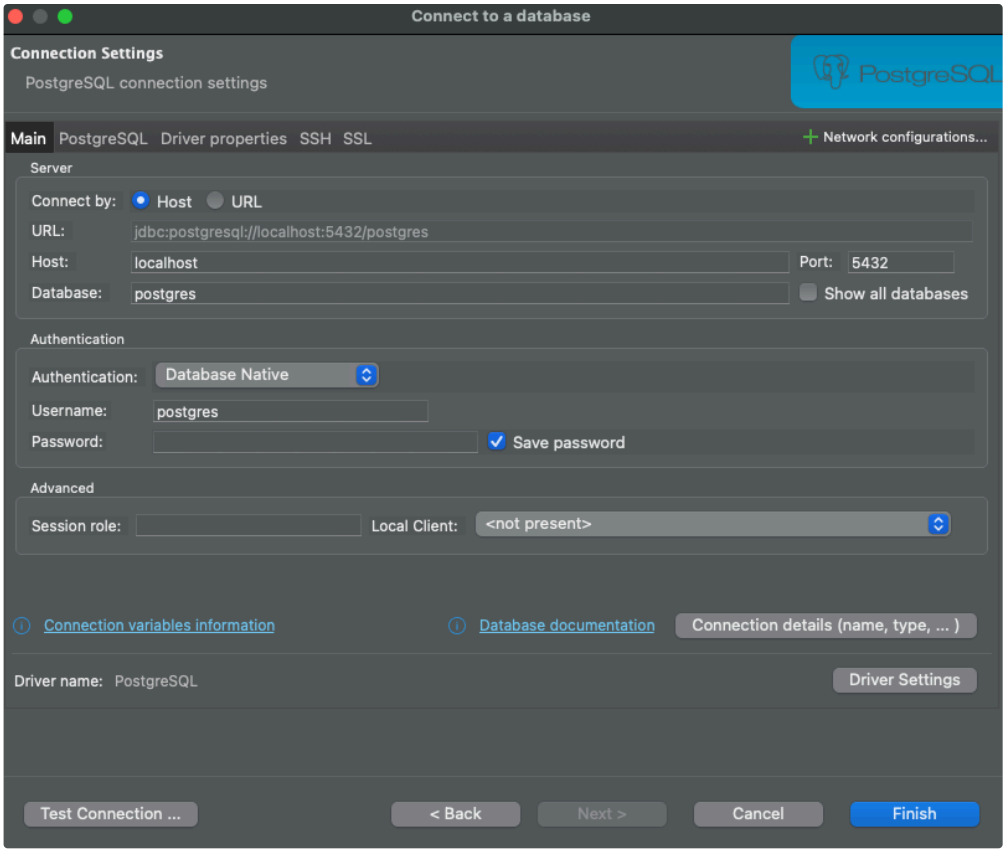
4. We can now populate this with the garnered information

   Changing:

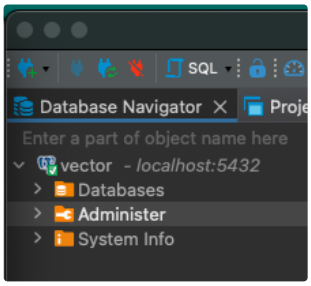
   Database = vector

   Username = vector

   Password = (from the command ran earlier)



5. Clicking test connection at the bottom left will allow you to ensure all is working as expected and then click finish
6. You should now have a connection to the DB (if not click on the icon and it will action or right click and choose connect)



Updating User Permissions

1. Connect to the database using your SQL client (as above)
2. Select the  icon to open an SQL editor
3. Find the user you wish to update

   `select * from "user" where email like 'username@raft.ai'`

   (we are working on public.Tables.user schema)

   You will see a table displayed in the bottom half of the screen that is the return data, similar to this:

UUID	authentication_id	email	name	global_permission_level
744cedc4-1fdf-422f-83b9-fce1c968a216	d1MzLO83J6KYZS82D4Hg1ktzyr5l1	user@raft.ai	Alice	user

We are in this case changing the access level via global_permission_level

4. To change the data we have many ways to edit the data, here are three options:
    a. Run the following SQL query in the SQL terminal to update user permissions:

    ```
    UPDATE "user" SET global_permission_level='global_admin' WHERE "uuid"='744cedc4-1fdf-422f-83b9-fce1c968a216';
    ```

    b. Double click the cell you wish to edit, edit and click save in the bottom left of the table view.
    c. As with b, but select the small drop down next to save and generate script. This will output the command to run:

    ```
    UPDATE public."user"
        SET global_permission_level='global_admin'
        WHERE "uuid"='744cedc4-1fdf-422f-83b9-fce1c968a216'::uuid::uuid;
    ```

Alternative Method Using Google Cloud Console

1. Navigate to the SQL section in Google Cloud Console
2. Select the appropriate instance (production or staging)
3. Use the built-in query editor to run SQL commands directly

Note: Always ensure you have the necessary permissions and follow security protocols when accessing production databases and modifying user permissions.