

# Securing a Cloud SQL for PostgreSQL Instance

## Task 1. Create a Cloud SQL for PostgreSQL instance with CMEK enabled

In this task, you create a Cloud SQL for PostgreSQL instance with CMEK enabled. It is imperative that you keep the keys safe as you cannot manage your database without them.

Create a per-product, per-project service account for Cloud SQL

You can create the service account you require for Cloud SQL CMEK using the gcloud beta services identity create command.

1. In Cloud Shell, run the following to create the service account:

```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to qwiklabs-gcp-01-4c06ef77a64f.
Use 'gcloud config set project [PROJECT_ID]' to change to a different project.
student_00_282a6087f04c@cloudshell:~ (quiklabs-gcp-01-4c06ef77a64f) $ export PROJECT_ID=$(gcloud config list --format 'value(core.project)')
gcloud beta services identity create \
  --service=sqladmin.googleapis.com \
  --project=$PROJECT_ID
Service identity created: service-388425943302@gcp-sa-cloud-sql.iam.gserviceaccount.com
student_00_282a6087f04c@cloudshell:~ (quiklabs-gcp-01-4c06ef77a64f) $
```

2. Click the **Authorize** button if prompted.

This creates the service account that you will bind to the CMEK in a later step.

Create a Cloud Key Management Service keyring and key

In this section, you create a Cloud KMS keyring and key to use with CMEK.

1. In Cloud Shell, run the following command to create the Cloud KMS keyring:

```
student_00_282a6087f04c@cloudshell:~ (quiklabs-gcp-01-4c06ef77a64f) $ export KMS_KEYRING_ID=cloud-sql-keyring
export ZONE=$(gcloud compute instances list --filter="NAME=bastion-vm" --format=json | jq -r .[].zone | awk -F "/" '{print $NF}')
export REGION=${ZONE::-2}
gcloud kms keyrings create $KMS_KEYRING_ID \
  --location=$REGION
student_00_282a6087f04c@cloudshell:~ (quiklabs-gcp-01-4c06ef77a64f) $
```

2. In Cloud Shell, run the following command to create the Cloud KMS key:

```
student_00_282a6087f04c@cloudshell:~ (quiklabs-gcp-01-4c06ef77a64f) $ export KMS_KEY_ID=cloud-sql-key
gcloud kms keys create $KMS_KEY_ID \
  --location=$REGION \
  --keyring=$KMS_KEYRING_ID \
  --purpose=encryption
student_00_282a6087f04c@cloudshell:~ (quiklabs-gcp-01-4c06ef77a64f) $
```

3. In Cloud Shell, run the following command to bind the key to the service account:

```
student_00_282a6087f04c@cloudshell:~ (qwiklabs-gcp-01-4c06ef77a64f) $ export PROJECT_NUMBER=$(gcloud projects describe ${PROJECT_ID} \
--format 'value(projectNumber)')
gcloud kms keys add-iam-policy-binding $KMS_KEY_ID \
--location=$REGION \
--keyring=$KMS_KEYRING_ID \
--member=serviceAccount:service-${PROJECT_NUMBER}@gcp-sa-cloud-sql.iam.gserviceaccount.com \
--role=roles/cloudkms.cryptoKeyEncrypterDecrypter
Updated IAM policy for key [cloud-sql-key].
bindings:
- members:
  - serviceAccount:service-388425943302@gcp-sa-cloud-sql.iam.gserviceaccount.com
  role: roles/cloudkms.cryptoKeyEncrypterDecrypter
etag: BwY4_76Jlyc=
version: 1
student_00_282a6087f04c@cloudshell:~ (qwiklabs-gcp-01-4c06ef77a64f) $
```

## Create a Cloud SQL instance with CMEK enabled

In this section, you create a Cloud SQL for PostgreSQL instance with CMEK enabled. It is not possible to patch an existing instance to enable CMEK, so you should bear this in mind if you plan to use CMEK to encrypt your data.

In order to access your Cloud SQL instance from external development or application environments, you can configure the Cloud SQL instance with a public IP address and control access by allowlisting the IP address of those environments. This limits access to the public interface to those address ranges that you specify.

You treat the Compute Engine VM instance in the lab as a development environment and therefore need to allow list the external IP address of that instance. You also add the external IP address of the Cloud Shell to the allowlist to make it easier to complete tasks later in the lab.

1. In Cloud Shell, run the following command to find the external IP address of the bastion-vm VM instance:

```
student_00_282a6087f04c@cloudshell:~ (qwiklabs-gcp-01-4c06ef77a64f) $ export AUTHORIZED_IP=$(gcloud compute instances describe basti
on-vm \
--zone=$ZONE \
--format 'value(networkInterfaces[0].accessConfigs.natIP)')
echo Authorized IP: $AUTHORIZED_IP
Authorized IP: 34.57.235.15
student_00_282a6087f04c@cloudshell:~ (qwiklabs-gcp-01-4c06ef77a64f) $
```

2. In Cloud Shell, run the following command to find the external IP address of the Cloud Shell:

```
student_00_282a6087f04c@cloudshell:~ (qwiklabs-gcp-01-4c06ef77a64f) $ export CLOUD_SHELL_IP=$(curl ifconfig.me)
echo Cloud Shell IP: $CLOUD_SHELL_IP
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           %             %       Dload  Upload    Total   Spent    Left   Speed
100    14    100    14    0    0    63      0  --:--:--  --:--:--  --:--:--    63
Cloud Shell IP: 34.143.194.147
student_00_282a6087f04c@cloudshell:~ (qwiklabs-gcp-01-4c06ef77a64f) $
```

3. In Cloud Shell, run the following command to create your Cloud SQL for PostgreSQL instance with:

```
student_00_282a6087f04c@cloudshell:~ (qwiklabs-gcp-01-4c06ef77a64f)$ export KEY_NAME=$(gcloud kms keys describe $KMS_KEY_ID \
--keyring=$KMS_KEYRING_ID --location=$REGION \
--format 'value(name)')

export CLOUDSQL_INSTANCE=postgres-orders
gcloud sql instances create $CLOUDSQL_INSTANCE \
--project=$PROJECT_ID \
--authorized-networks=${AUTHORIZED_IP}/32,$CLOUD_SHELL_IP/32 \
--disk-encryption-key=$KEY_NAME \
--database-version=POSTGRES_13 \
--cpu=1 \
--memory=3840MB \
--region=$REGION \
--root-password=supersecret!
WARNING: You are creating a Cloud SQL instance encrypted with a customer-managed key. If anyone destroys a customer-managed key, al
l data encrypted with it will be permanently lost.

Do you want to continue (Y/n)?
```

```
--region=$REGION \
--root-password=supersecret!
WARNING: You are creating a Cloud SQL instance encrypted with a customer-managed key. If anyone destroys a customer-managed key, al
l data encrypted with it will be permanently lost.

Do you want to continue (Y/n)? y

Creating Cloud SQL instance for POSTGRES 13...done.
Created [https://sqladmin.googleapis.com/sql/v1beta4/projects/qwiklabs-gcp-01-4c06ef77a64f/instances/postgres-orders].
NAME: postgres-orders
DATABASE_VERSION: POSTGRES_13
LOCATION: us-central1-c
TIER: db-custom-1-3840
PRIMARY_ADDRESS: 35.225.26.218
PRIVATE_ADDRESS: -
STATUS: RUNNING
```

4. Enter 'y' if prompted after entering the command.

## Task 2. Enable and configure pgAudit on a Cloud SQL for PostgreSQL database


In this task, you enable and configure the pgAudit database extension which enables fine-grained control of logging of all types of database activity.

1. In Cloud Shell, run the following command to add the pgAudit database flags to your Cloud SQL instance:

```
student_00_282a6087f04c@cloudshell:~ (qwiklabs-gcp-01-4c06ef77a64f)$ gcloud sql instances patch $CLOUDSQL_INSTANCE \
--database-flags cloudsql.enable_pgaudit=on,pgaudit.log=all
The following message will be used for the patch API method.
{"name": "postgres-orders", "project": "qwiklabs-gcp-01-4c06ef77a64f", "settings": {"databaseFlags": [{"name": "cloudsql.enable_pgaudit", "value": "on"}, {"name": "pgaudit.log", "value": "all"}]}}
WARNING: This patch modifies database flag values, which may require your instance to be restarted. Check the list of supported flags - https://cloud.google.com/sql/docs/postgres/flags - to see if your instance will be restarted when this patch is submitted.

Do you want to continue (Y/n)? y

Patching Cloud SQL instance...working.
```

2. Enter 'y' if prompted to confirm and continue.
3. In Cloud Console, on the **Navigation menu** () , click **SQL**.
4. Click on the Cloud SQL instance named postgres-orders.
5. In the Cloud SQL **Overview** panel, top menu, click **Restart** to restart the instance after the patch that you ran in step 1.

If prompted again, click **Restart** again in the pop-up dialog.

SQL

Instances

Backups

Instances

CREATE INSTANCE

MIGRATE DATABASE

Starting Feb 1, 2025, all instances running community end-of-life versions of PostgreSQL and MySQL are under extended support. These instances will be charged for extended support from May 1, 2025. Upgrade your instances running end-of-life versions before May 1, 2025 to prevent additional charges. [Learn more](#)

VIEW AFFECTED INSTANCES

DISMISS

Filter

Enter property name or value

?

⋮

<input type="checkbox"/>	Status	Instance ID	Issues	Cloud SQL edition	Type	Actions
<input type="checkbox"/>	✓	postgres-orders		Enterprise	PostgreSQL 13	⋮

[+ CREATE INSTANCE](#)

 **MIGRATE DATABASE**

## SQL

 Instances

Backups

**i**

Starting Feb 1, 2025, all instances running community end-of-life versions of PostgreSQL and MySQL are under extended support. These instances will be charged for extended support from May 1, 2025. Upgrade your instances running end-of-life versions before May 1, 2025 to prevent additional charges. [Learn more](#)

[VIEW AFFECTED INSTANCES](#)

DISMISS

Filter

Enter property name or value

④

II

7

Status

Instance ID 

## Issues

Cloud SQL edition

Type

### Actions

7



postgres-orders

Enterprise

## PostgreSQL 13

6. In Cloud console, in the **Connect to this instance** section, click **Open Cloud Shell**.  
A command to connect to the instance will auto-populate in Cloud Shell.
7. Run that command as is, and enter the password `supersecret!` when prompted.  
A **psql** session will start in Cloud Shell.
8. In **psql**, run the following command to create the orders database and enable the `pgAudit` extension to log all reads and writes:

A command to connect to the instance will auto-populate in Cloud Shell.

7. Run that command as is, and enter the password `supersecret!` when prompted.

A **psql** session will start in Cloud Shell.

8. In **psql**, run the following command to create the orders database and enable the pgAudit extension to log all reads and writes:

```
postgres=> CREATE DATABASE orders;
\c orders;
CREATE DATABASE
Password:
psql (16.9 (Ubuntu 16.9-1.pgdg24.04+1), server 13.21)
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression: off)
You are now connected to database "orders" as user "postgres".
orders=>
```


9. Enter the password supersecret! again.
10. In **psql**, run the following command to create and configure the database extension:

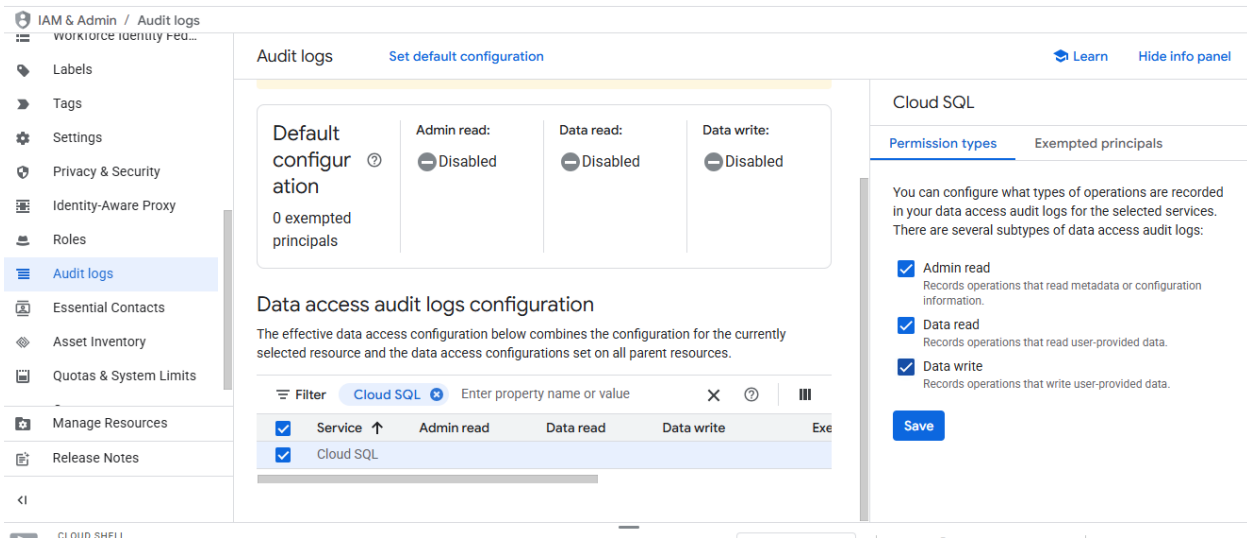
10. In **psql**, run the following command to create and configure the database extension:

```
orders=> CREATE EXTENSION pgaudit;
ALTER DATABASE orders SET pgaudit.log = 'read,write';
CREATE EXTENSION
ALTER DATABASE
orders=>
```

## Enable Audit Logging in Cloud Console

In this section, you enable Audit Logging in Cloud Console.

1. In Cloud Console, on the **Navigation menu** () , click **IAM & Admin > Audit Logs**.
2. In the **Filter** box under **Data access audit logs configuration**, type Cloud SQL, and select the entry in the drop-down list.
3. Enable the checkbox for Cloud SQL on the left, and then enable the following checkboxes in the **Info Panel** on the right:
  - **Admin read**
  - **Data read**
  - **Data write**
4. Click **Save** in the **Info Panel**.



## Populate a database on Cloud SQL for PostgreSQL

In this section, you populate the orders database with data provided to you.

1. Click the **+** icon on the Cloud Shell title bar to open a new tab in the Cloud Shell.
2. In the new tab, run the following to download the data and database population scripts:

```
student_00_282a6087f04c@cloudshell:~ (qwiklabs-gcp-01-4c06ef77a64f)$ export SOURCE_BUCKET=gs://cloud-training/gsp920
gsutil -m cp ${SOURCE_BUCKET}/create_orders_db.sql .
gsutil -m cp ${SOURCE_BUCKET}/DDL/distribution_centers_data.csv .
gsutil -m cp ${SOURCE_BUCKET}/DDL/inventory_items_data.csv .
gsutil -m cp ${SOURCE_BUCKET}/DDL/order_items_data.csv .
gsutil -m cp ${SOURCE_BUCKET}/DDL/products_data.csv .
gsutil -m cp ${SOURCE_BUCKET}/DDL/users_data.csv .
Copying gs://cloud-training/gsp920/create_orders_db.sql...
/ [1/1 files] [ 1.6 KiB/ 1.6 KiB] 100% Done
Operation completed over 1 objects/1.6 KiB.
Copying gs://cloud-training/gsp920/DDL/distribution_centers_data.csv...
/ [1/1 files] [ 371.0 B/ 371.0 B] 100% Done
Operation completed over 1 objects/371.0 B.
```

2. Continue in the new tab, and run the following to create and populate the database:

```
student_00_282a6087f04c@cloudshell:~ (qwiklabs-gcp-01-4c06ef77a64f)$ export CLOUDSQL_INSTANCE=postgres-orders
export POSTGRES_IP=$(gcloud sql instances describe $CLOUDSQL_INSTANCE --format="value(ipAddresses[0].ipAddress)")
export PGPASSWORD=supersecret!
psql "sslmode=disable user=postgres hostaddr=${POSTGRES_IP}" \
-c "\i create_orders_db.sql"
psql:create_orders_db.sql:1: ERROR: database "orders" already exists
psql (16.9 (Ubuntu 16.9-1.pgdg24.04+1), server 13.21)
You are now connected to database "orders" as user "postgres".
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
```

3. Exit the terminal session in the new tab:

Exit

4. Return to your **psql** session in the original Cloud Shell tab, and run the following to further log all SELECT operations on a particular relation (such as the order\_items table):

```
orders=> CREATE ROLE auditor WITH NOLOGIN;
ALTER DATABASE orders SET pgaudit.role = 'auditor';
GRANT SELECT ON order_items TO auditor;
CREATE ROLE
ALTER DATABASE
GRANT
orders=> █
```

5. Run the first SELECT query below :


```
student_00_282a6087f04c@cloudshell:~ (qwiklabs-gcp-01-4c06ef77a64f) $ export CLOUDSQL_INSTANCE=postgres-orders
export POSTGRES_IP=$(gcloud sql instances describe $CLOUDSQL_INSTANCE --format="value(ipAddresses[0].ipAddress)")
export PGPASSWORD=supersecret!
psql "sslmode=disable user=postgres hostaddr=${POSTGRES_IP}" \
-c "\i create_orders_db.sql"
psql:create_orders_db.sql:1: ERROR: database "orders" already exists
psql (16.9 (Ubuntu 16.9-1.pgdg24.04+1), server 13.21)
You are now connected to database "orders" as user "postgres".
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
COPY 415647
COPY 11
COPY 198553
COPY 29120
COPY 126019
student_00_282a6087f04c@cloudshell:~ (qwiklabs-gcp-01-4c06ef77a64f) $
```

- The output is 500 rows long, so you can enter q to close the results and return to the orders=> prompt.
- Repeat the steps 5-6 for the other two query tabs in the code block.
- Run the following to exit **psql**:

\q

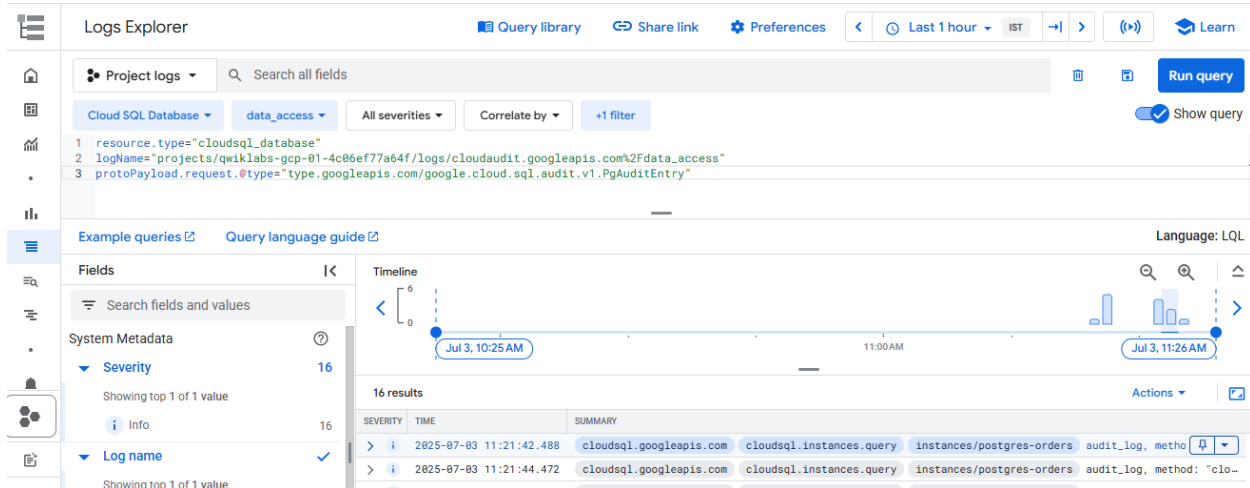
View pgAudit logs

In this step you will view the logging of your database updates and queries in the pgAudit logs.

- In Cloud Console, on the **Navigation menu** () , click **View all products**. Under **Observability**, click **Logging** to open the **Logs Explorer** page.
- In the **Query** tab of the **Logs Explorer**, paste the following, and click **Run query**

users_id	users_first_name	users_last_name	order_items_order_count	order_items_total_revenue
9368	Samuel	Patrick	14	592.3400020599365
1242	Aaron	Montgomery	13	571.6500034332275
3538	John	Stephen	13	717.3600034713745
112988	Emma	Kline	13	413.95000171661377
6889	Clara	Haley	13	440.1700038909912
15502	Vilma	Berg	13	405.82999992370605
15570	Amber	Parker	12	540.80999994659424
9167	Jack	Tina	12	538.4299983978271
111864	Fred	Andersen	12	545.2500057220459
6007	Jerry	Charles	12	568.9400005340576
4601	Sandi	Odom	12	641.6199989318848
3489	Eulalia	Patrick	12	430.280002117157
17721	Helen	Holland	12	697.1600027084351
1184	Henriette	Arellano	12	654.0400018692017
12209	Ella	Samuel	11	340.1300001144409
27956	Maragaret	Marcus	11	514.0300049781799
3446	Mattie	Mackenzie	11	485.790002822876
9655	Betty	Strickland	11	534.1400032043457
35603	James	Sanchez	11	587.2599983215332
111300	Charlotte	Williams	11	398.31999921798706
8632	Kathleen	Price	11	536.1199970245361

3. In the histogram displayed, you can see the audit activity associated with your DDL inserts and the SELECT queries you ran earlier.



4. Click on the last bar on the histogram, which corresponds to the SELECT queries you ran.

In the **Query results** panel below the histogram, the log entries are listed.

5. Expand a log entry, and under protoPayload.request you will see the SELECT query.

### Task 3. Configure Cloud SQL IAM database authentication

In this task, you configure Cloud SQL IAM database authentication. All of the database access and update tasks you have performed so far have used built-in PostgreSQL user accounts. You can also create Cloud SQL for PostgreSQL users using Cloud IAM accounts. Database users can authenticate to Cloud SQL using Cloud IAM instead of using built-in database accounts and fine-grained permissions at the database level can be granted to those users.

In this task, you configure the lab user account as a Cloud SQL IAM user, grant that user access to the orders.order\_items database table using the **postgres** administrator account, and then test access to the orders.order\_items database table from the command line using the **psql** command line utility.

The authentication process that is used in this task is explained in detail in the [IAM authentication documentation for Cloud SQL for PostgreSQL](#).

Test database access using a Cloud IAM user before Cloud SQL IAM authentication is configured.



You attempt to access the database using a Cloud IAM user before Cloud SQL IAM authentication has been enabled in order to establish that the Cloud IAM user cannot initially access the data. You will see this connection attempt fail before you proceed to the next section to address the issue.

- In Cloud Shell, test access to the orders database using the lab student account as the username:


```
student_00_282a6087f04c@cloudshell:~ (qwiklabs-gcp-01-4c06ef77a64f) $ export USERNAME=$(gcloud config list --format="value(core.account)")
export CLOUDSQL_INSTANCE=postgres-orders
export POSTGRES_IP=$(gcloud sql instances describe $CLOUDSQL_INSTANCE --format="value(ipAddresses[0].ipAddress)")
export PGPASSWORD=$(gcloud auth print-access-token)
psql --host=$POSTGRES_IP $USERNAME --dbname=orders
psql: error: connection to server at "35.225.26.218", port 5432 failed: FATAL: password authentication failed for user "student-00-282a6087f04c@qwiklabs.net"
connection to server at "35.225.26.218", port 5432 failed: FATAL: password authentication failed for user "student-00-282a6087f04c@qwiklabs.net"
student_00_282a6087f04c@cloudshell:~ (qwiklabs-gcp-01-4c06ef77a64f) $
```

This connection attempt fails, and you see an authentication failed message similar to the following because the Cloud SQL IAM user has not been created yet:

Cloud SQL IAM database authentication uses OAuth 2.0 access tokens as the Cloud IAM user password, which are short-lived and only valid for one hour so you should regenerate the token every time you need to authenticate. The access token should always be passed into the **psql** command using the **PGPASSWORD** environment variable as the buffer for the **psql** password parameter is too small to hold the OAuth 2.0 token string.

Create a Cloud SQL IAM user

In this section, you create a Cloud SQL IAM user and confirm that Cloud SQL IAM user authentication has been enabled.

1. In Cloud Console, on the **Navigation menu** () , click **SQL**
2. Click on the Cloud SQL instance named postgres-orders.

In the **Configuration** panel on the right, note that the **Database flags and parameters** list includes **pgAudit.log** and **cloudsql.enable\_pgaudit** only.

3. In the **SQL menu** (left panel) under **Primary instance**, click **Users** to open the **Users** panel.
4. Click **Add user account**.
5. Select **Cloud IAM**.
6. In the **Principal** box enter the lab student name: Username
7. Click **Add**.

Wait for the new user to be successfully added.

On the main overview page for instance, in the **Configuration** panel on the right, note **cloudsql.iam\_authentication** has been added to the **Database flags and parameters** list.


keyring/cryptosecrets/cloud-sql-key/cryptosecretversions/1

#### RE-ENCRYPT INSTANCE

#### Database flags and parameters

cloudsql.enable_pgaudit	on
pgaudit.log	all

Vertex AI Integration is disabled

 No labels set

## Add a user account to instance postgres-orders

---

### Choose how to authenticate

You can manage access to this instance using Cloud IAM or PostgreSQL built-in authentication. [Learn more](#)

☐ Built-in authentication

Creates a new username and password specific to this instance. User account will have `cloudsqlsuperuser` root access, but you can customize that later as needed. [Learn more](#)

☒ Cloud IAM

Adds an existing IAM principal as a user account on this instance. The IAM principal can be an IAM user, Service Account, or a Group. The IAM principal must have `cloudsql.instances.login` permission using the predefined Cloud SQL Instance User IAM role or custom IAM role in the IAM policy of the instance's project.

**IAM principal \***

student-00-282a6087f04c@qwiklabs.net

After you create a user account with Cloud IAM authentication, it will not have database privileges by default, so make sure to grant required database privileges using the `GRANT` statement. [Learn more](#)

ADD

CANCEL

## Users

All instances > postgres-orders

### ✓ postgres-orders

PostgreSQL 13

User accounts enable users and applications to connect to your instance. [Learn more](#)

[+ ADD USER ACCOUNT](#)

ADDED USERS

AUTHENTICATED IAM GROUP MEMBERS

These are accounts that you have granted instance access to, using either built-in or IAM authentication.

	User name ↑	Authentication	Password status	
👤	postgres	Built-in	N/A	⋮
👤	student-00-282a6087f04c@qwiklabs.net	IAM (user)	N/A	⋮

## Grant the Cloud IAM user access to a Cloud SQL database table

You now connect to the postgres-orders instance using the built in postgres administrator account and grant access to the orders.order\_items table to the Cloud IAM user.

1. On the main overview page for instance, in the **Connect to this instance** section, click **Open Cloud Shell**.

A command to connect to the instance will auto-populate in Cloud Shell.

2. Run that command as is, and enter the password supersecret! when prompted.
3. Enter the following SQL command to switch to the orders database:

```
student_00_282a6087f04c@cloudshell:~ (qwiklabs-gcp-01-4c06ef77a64f) $ gcloud sql connect postgres-orders --user=postgres --quiet
Allowlisting your IP for incoming connection for 5 minutes...done.
Connecting to database with SQL user [postgres].Password:
psql (16.9 (Ubuntu 16.9-1.pgdg24.04+1), server 13.21)
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression: off)
Type "help" for help.

postgres=> |
```

```
postgres=> \c orders
Password:
psql (16.9 (Ubuntu 16.9-1.pgdg24.04+1), server 13.21)
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression: off)
You are now connected to database "orders" as user "postgres".
orders=> |
```

When prompted for a password enter supersecret! again.

4. Enter the following SQL command to grant the lab user all permissions on the order\_items table. The Cloud IAM username for the lab has been inserted into this query for you.

```
orders=> GRANT ALL PRIVILEGES ON TABLE order_items TO "student-00-282a6087f04c@qwiklabs.net";
\q
GRANT
student_00_282a6087f04c@cloudshell:~ (qwiklabs-gcp-01-4c06ef77a64f)$
```

Test database access using a Cloud IAM user after Cloud SQL IAM authentication is configured.

You now repeat the attempt to access the database using a Cloud IAM user after Cloud SQL IAM authentication has been enabled in order to establish that the Cloud IAM user can now access the data.

You can now test access to the database again using the Cloud IAM user instead of the built-in postgres user:

1. In the Cloud Shell, run the following command to connect to the database using the Cloud IAM database user:

```
student_00_282a6087f04c@cloudshell:~ (qwiklabs-gcp-01-4c06ef77a64f)$ export PGPASSWORD=$(gcloud auth print-access-token)
psql --host=$POSTGRES_IP $USERNAME --dbname=orders
psql (16.9 (Ubuntu 16.9-1.pgdg24.04+1), server 13.21)
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression: off)
Type "help" for help.
orders=>
```

This connection succeeds, and you are now connected to the instance using Cloud IAM user authentication.

2. Test your access permission by running this query:

```
orders=> SELECT COUNT(*) FROM order_items;
count
-----
198553
(1 row)

orders=>
```

3. Confirm that you do not have access to one of the other tables by running this query:

```
orders=> SELECT COUNT(*) FROM users;
ERROR: permission denied for table users
orders=> SELECT COUNT(*) FROM users;
ERROR: permission denied for table users
orders=>
```