## **Task 1: Migrate a stand-alone PostgreSQL database to a Cloud SQL for PostgreSQL instance**

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

When you are connected, you are already authenticated, and the project is set to your **Project\_ID**, qwiklabs-gcp-02-81a40ac37518. 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-02-81a40ac37518

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

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

***gcloud auth list***

content\_copy

1. Click **Authorize**.

**Output:**

ACTIVE: \*

ACCOUNT: student-02-78ee64146dd0@qwiklabs.net

To set the active account, run:

$ gcloud config set account `ACCOUNT`

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

***gcloud config list project***

content\_copy

**Output:**

[core]

project = qwiklabs-gcp-02-81a40ac37518

**Note:**For full documentation of gcloud, in Google Cloud, refer to [the gcloud CLI overview guide](https://cloud.google.com/sdk/gcloud).

Verify that the Database Migration API is enabled

1. In the Google Cloud console, enter **Database Migration API** in the top search bar. Click on the result for **Database Migration API**.

This page will either show status information or give you the option to enable the API.

1. If necessary, **Enable** the API.

Verify that the Service Networking API is enabled

The Service Networking API is required in order to be able to configure Cloud SQL to support VPC Peering and connections over a private ip-address.

1. In the Cloud console, enter **Service Networking API** in the top search bar. Click on the result for **Service Networking API**.

This page will either show status information or give you the option to enable the API.

1. If necessary, enable the API.

gcloud services enable datamigration.googleapis.com

gcloud services enable servicenetworking.googleapis.com

**Task 1. Prepare the source database for migration**

In this task you will add supporting features to the source database which are required in order for **Database Migration Service** to perform a migration. These are:

* Installing and configuring the pglogical database extension.
* Configuring the stand-alone PostgreSQL database to allow access from Cloud Shell and Cloud SQL.
* Adding the pglogicaldatabase extension to the postgres, orders and gmemegen\_db databases on the stand-alone server.
* Creating a migration\_admin user (with Replication permissions) for database migration and granting the required permissions to schemata and relations to that user.

Upgrade the database with the pglogical extension

In this step you will download and add the pglogical database extension to the orders and postgres databases on the antern-postgresql-vm VM Instance.

1. In the Google Cloud console, on the **Navigation menu** (Navigation menu icon), click **Compute Engine** > **VM instances**.
2. In the entry for antern-antern-postgresql-vm, under Connect click **SSH**.
3. If prompted, click **Authorize**.
4. In the terminal in the new browser window, install the pglogical database extension:

***sudo apt install postgresql-13-pglogical***

1. Download and apply some additions to the PostgreSQL configuration files (to enable pglogical extension) and restart the postgresql service:

***sudo su - postgres -c "gsutil cp gs://cloud-training/gsp918/pg\_hba\_append.conf ."***

***sudo su - postgres -c "gsutil cp gs://cloud-training/gsp918/postgresql\_append.conf ."***

***sudo su - postgres -c "cat pg\_hba\_append.conf >>*** /etc/postgresql/13/main/pg\_hba.conf ***"***

***sudo su - postgres -c "cat postgresql\_append.conf >>*** /etc/postgresql/13/main/postgresql.conf ***"***

***sudo systemctl restart postgresql@13-main***

In pg\_hba.conf these commands added a rule to allow access to all hosts:

#GSP918 - allow access to all hosts

host all all 0.0.0.0/0 md5

In postgresql.conf, these commands set the minimal configuration for pglogical to configure it to listen on all addresses:

#GSP918 - added configuration for pglogical database extension

wal\_level = logical # minimal, replica, or logical

max\_worker\_processes = 10 # one per database needed on provider node

# one per node needed on subscriber node

max\_replication\_slots = 10 # one per node needed on provider node

max\_wal\_senders = 10 # one per node needed on provider node

shared\_preload\_libraries = 'pglogical'

max\_wal\_size = 1GB

min\_wal\_size = 80MB

listen\_addresses = '\*' # what IP address(es) to listen on, '\*' is all

The above code snippets were appended to the relevant files and the PostgreSQL service restarted.

1. Launch the **psql** tool:

***sudo su - postgres***

***psql***

***\l***

1. Add the pglogical database extension to the postgres, orders and gmemegen\_db databases.

***\c postgres;***

***CREATE EXTENSION pglogical;***

***\c orders;***

***CREATE EXTENSION pglogical;***

1. List the PostgreSQL databases on the server:

***\l***

Here you can see, besides the default postgresql databases, the orders and gmemegen\_db databases provided for this lab. You will not use the gmemegen\_db database in this lab, but will include it in the migration for use in a later lab.

List of databases

Name | Owner | Encoding | Collate | Ctype | Access privileges

-------------+----------+----------+---------+---------+-----------------------

gmemegen\_db | postgres | UTF8 | C.UTF-8 | C.UTF-8 |

orders | postgres | UTF8 | C.UTF-8 | C.UTF-8 |

postgres | postgres | UTF8 | C.UTF-8 | C.UTF-8 |

template0 | postgres | UTF8 | C.UTF-8 | C.UTF-8 | =c/postgres +

| | | | | postgres=CTc/postgres

template1 | postgres | UTF8 | C.UTF-8 | C.UTF-8 | =c/postgres +

| | | | | postgres=CTc/postgres

(5 rows)

Create the database migration user

In this step you will create a dedicated user for managing database migration.

1. In **psql**, enter the commands below to create a new user with the replication role:

**CREATE USER migration\_admin PASSWORD 'DMS\_1s\_cool!';**

***ALTER DATABASE orders OWNER TO migration\_admin;***

***ALTER ROLE migration\_admin WITH REPLICATION;***

Assign permissions to the migration user

In this step you will assign the necessary permissions to the migration\_admin user to enable **Database Migration Service** to migrate your database.

1. In **psql**, grant permissions to the pglogical schema and tables for the postgres database.

***\c postgres;***

***GRANT USAGE ON SCHEMA pglogical TO migration\_admin;***

***GRANT ALL ON SCHEMA pglogical TO migration\_admin;***

***GRANT SELECT ON pglogical.tables TO migration\_admin;***

***GRANT SELECT ON pglogical.depend TO migration\_admin;***

***GRANT SELECT ON pglogical.local\_node TO migration\_admin;***

***GRANT SELECT ON pglogical.local\_sync\_status TO migration\_admin;***

***GRANT SELECT ON pglogical.node TO migration\_admin;***

***GRANT SELECT ON pglogical.node\_interface TO migration\_admin;***

***GRANT SELECT ON pglogical.queue TO migration\_admin;***

***GRANT SELECT ON pglogical.replication\_set TO migration\_admin;***

***GRANT SELECT ON pglogical.replication\_set\_seq TO migration\_admin;***

***GRANT SELECT ON pglogical.replication\_set\_table TO migration\_admin;***

***GRANT SELECT ON pglogical.sequence\_state TO migration\_admin;***

***GRANT SELECT ON pglogical.subscription TO migration\_admin;***

1. In **psql**, grant permissions to the pglogical schema and tables for the orders database.

***\c orders;***

***GRANT USAGE ON SCHEMA pglogical TO migration\_admin;***

***GRANT ALL ON SCHEMA pglogical TO migration\_admin;***

***GRANT SELECT ON pglogical.tables TO migration\_admin;***

***GRANT SELECT ON pglogical.depend TO migration\_admin;***

***GRANT SELECT ON pglogical.local\_node TO migration\_admin;***

***GRANT SELECT ON pglogical.local\_sync\_status TO migration\_admin;***

***GRANT SELECT ON pglogical.node TO migration\_admin;***

***GRANT SELECT ON pglogical.node\_interface TO migration\_admin;***

***GRANT SELECT ON pglogical.queue TO migration\_admin;***

***GRANT SELECT ON pglogical.replication\_set TO migration\_admin;***

***GRANT SELECT ON pglogical.replication\_set\_seq TO migration\_admin;***

***GRANT SELECT ON pglogical.replication\_set\_table TO migration\_admin;***

***GRANT SELECT ON pglogical.sequence\_state TO migration\_admin;***

***GRANT SELECT ON pglogical.subscription TO migration\_admin;***

1. In **psql**, grant permissions to the public schema and tables for the orders database.

***GRANT USAGE ON SCHEMA public TO migration\_admin;***

***GRANT ALL ON SCHEMA public TO migration\_admin;***

***GRANT SELECT ON public.distribution\_centers TO migration\_admin;***

***GRANT SELECT ON public.inventory\_items TO migration\_admin;***

***GRANT SELECT ON public.order\_items TO migration\_admin;***

***GRANT SELECT ON public.products TO migration\_admin;***

***GRANT SELECT ON public.users TO migration\_admin;***

The source databases are now prepared for migration. The permissions you have granted to the migration\_admin user are all that is required for **Database Migration Service** to migrate the postgres and orders databases.

Make the migration\_admin user the owner of the tables in the orders database, so that you can edit the source data later, when you test the migration.

1. In **psql**, run the following commands:

***\c orders;***

***\dt***

***ALTER TABLE public.distribution\_centers OWNER TO migration\_admin;***

***ALTER TABLE public.inventory\_items OWNER TO migration\_admin;***

***ALTER TABLE public.order\_items OWNER TO migration\_admin;***

***ALTER TABLE public.products OWNER TO migration\_admin;***

***ALTER TABLE public.users OWNER TO migration\_admin;***

***\dt***

***ALTER TABLE public.inventory\_items ADD PRIMARY KEY(id);***

***\q***

***exit***

List of relations

Schema | Name | Type | Owner

--------+----------------------+-------+-------

public | distribution\_centers | table | migration\_admin

public | inventory\_items | table | migration\_admin

public | order\_items | table | migration\_admin

public | products | table | migration\_admin

public | users | table | migration\_admin

* 1. rows)

1. Exit **psql** and the postgres user session

exit

**Part 2. Create a Database Migration Service connection profile for a stand-alone PostgreSQL database**

In this task, you will create a connection profile for the PostgreSQL source instance.

Get the connectivity information for the PostgreSQL source instance

In this step, you identify the internal IP address of the source database instance that you will migrate to Cloud SQL.

1. In the Google Cloud Console, on the **Navigation menu** (Navigation menu icon), click **Compute Engine** > **VM instances**.
2. Locate the line with the instance called **antern-postgresql-vm**.
3. Copy the value for **Internal IP** (e.g., 10.128.0.2).

Create a new connection profile for the PostgreSQL source instance

A connection profile stores information about the source database instance (e.g., stand-alone PostgreSQL) and is used by the **Database Migration Service** to migrate data from the source to your destination Cloud SQL database instance. After you create a connection profile, it can be reused across migration jobs.

In this step you will create a new connection profile for the PostgreSQL source instance.

1. In the Google Cloud Console, on the **Navigation menu** (Navigation menu icon), click **Database Migration** > **Connection profiles**.
2. Click **+ Create Profile**.
3. For **Database engine**, select **PostgreSQL**.
4. For **Connection profile name**, enter **postgres-vm**.
5. For **Hostname or IP address**, enter the internal IP for the PostgreSQL source instance that you copied in the previous task (e.g., 10.128.0.2)
6. For **Port**, enter **5432**.
7. For **Username**, enter **migration\_admin**.
8. For **Password**, enter **DMS\_1s\_cool!** .
9. For **Region** select us-east1.
10. For all other values leave the defaults.
11. Click **Create**.

A new connection profile named **postgres-vm** will appear in the Connections profile list.

**Part 3. Create and start a continuous migration job**

When you create a new migration job, you first define the source database instance using a previously created connection profile. Then you create a new destination database instance and configure connectivity between the source and destination instances.

In this task, you use the migration job interface to create a new Cloud SQL for PostgreSQL database instance and set it as the destination for the continuous migration job from the PostgreSQL source instance.

Create a new continuous migration job

In this step you will create a new continuous migration job.

1. In the Google Cloud Console, on the **Navigation menu** (Navigation menu icon), click **Database Migration** > **Migration jobs**.
2. Click **+ Create Migration Job**.
3. For **Migration job name**, enter **vm-to-cloudsql**.
4. For **Source database engine**, select **PostgreSQL**.
5. For **Destination region**, select us-east1.
6. For **Destination database engine**, select **Cloud SQL for PostgreSQL**.
7. For **Migration job type**, select **Continuous**.

Leave the defaults for the other settings.

1. Click **Save & Continue**.

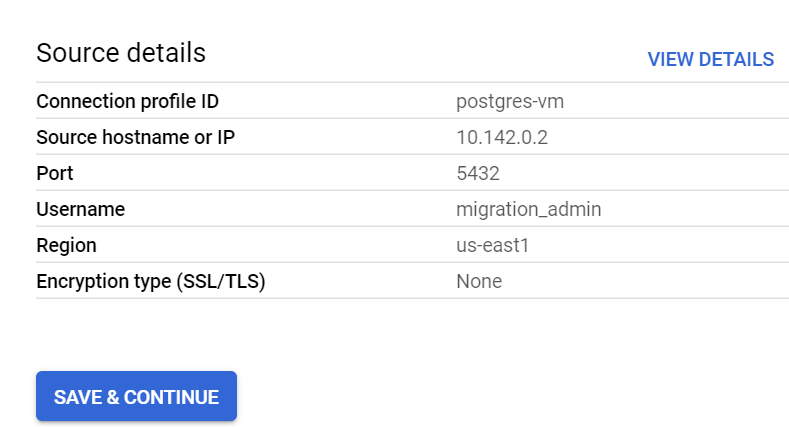
Define the source instance

In this step, you will define the source instance for the migration.

1. For **Source connection profile**, select **postgres-vm**.

Leave the defaults for the other settings.

1. Click **Save & Continue**.



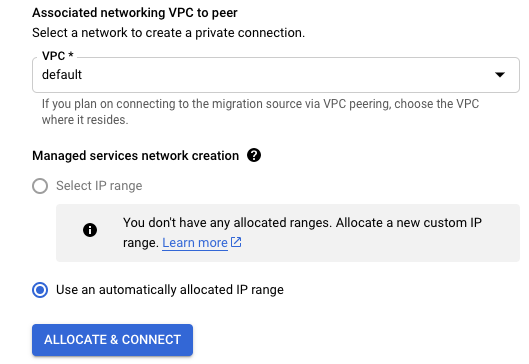
Create the destination instance

In this step, you will create the destination instance for the migration.

1. For **Destination Instance ID**, enter **postgresql-cloudsql**.
2. For **Password**, enter **supersecret!**.
3. For **Choose a Cloud SQL edition**, select **Enterprise** edition.
4. For **Database version**, select **Cloud SQL for PostgreSQL 13**.
5. In **Choose region and zone** section, select **Single zone** and select us-east1-c as **primary zone**.
6. For **Instance connectivity**, select **Private IP** and **Public IP**.
7. Select **Use an automatically allocated IP range**.

Leave the defaults for the other settings.

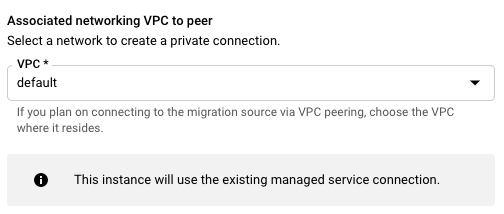
1. Click **Allocate & Connect**.



**Note:** This step may take a few minutes. If asked to retry the request, click the Retry button to refresh the Service Networking API.

When this step is complete, an updated message notifies you that the instance will use the existing managed service connection.

You will need to edit the pg\_hba.conf file on the VM instance to allow access to the IP range that is automatically generated in point 5 of the previous step. You will do this in a later step before testing the migration configuration at the end of this task.



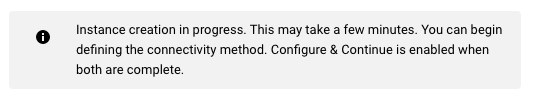
Enter the additional information needed to create the destination instance on Cloud SQL.

1. For **Machine shapes**. check **1 vCPU, 3.75 GB**
2. For **Storage type**, select **SSD**
3. For **Storage capacity**, select **10 GB**
4. Click **Create & Continue**.

If prompted to confirm, click **Create Destination & Continue**. A message will state that your destination database instance is being created. Continue to the next step while you wait.

Define the connectivity method

In this step, you will define the connectivity method for the migration.



1. For **Connectivity method**, select **VPC peering**.
2. For **VPC**, select **default**.

VPC peering is configured by **Database Migration Service** using the information provided for the VPC network (the default network in this example).

When you see an updated message that the destination instance was created, proceed to the next step.

The updated message informs you that the destination Cloud SQL instance has been created.

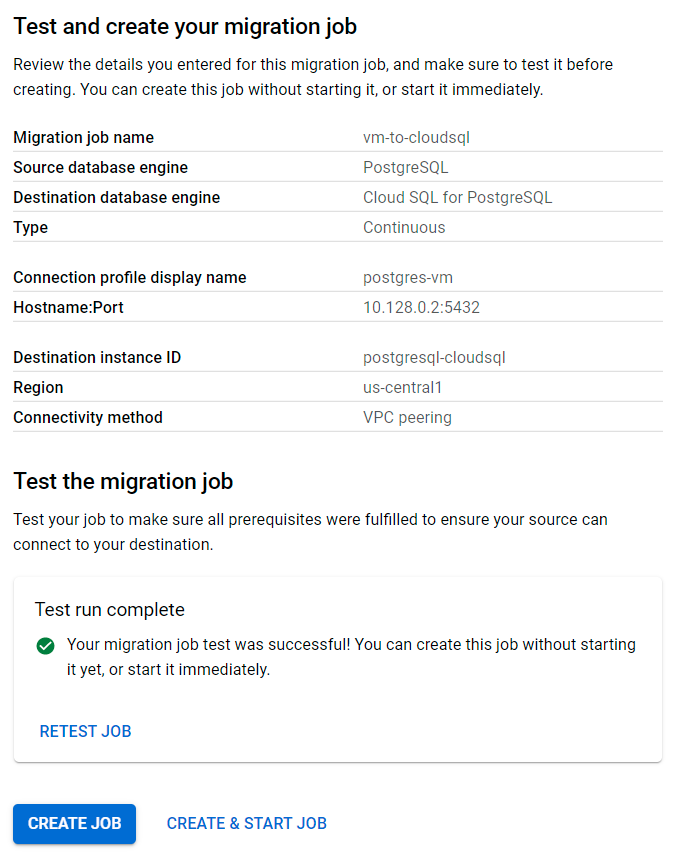
1. Click **Configure & Continue**.
2. Save and exit the nano editor with Ctrl-O, Enter, Ctrl-X
3. Restart the PostgreSQL service to make the changes take effect. In the VM instance Terminal session:

sudo systemctl start postgresql@13-main

Test and start the continuous migration job

In this step, you will test and start the migration job.

1. In the **Database Migration Service** tab you open earlier, review the details of the migration job.
2. Click **Test Job**.
3. After a successful test, click **Create & Start Job**.



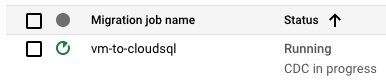
If prompted to confirm, click **Create & Start**.

Review the status of the continuous migration job

In this step, you will confirm that the continuous migration job is running.

1. In the Google Cloud Console, on the **Navigation menu** (Navigation menu icon), click **Database Migration** > **Migration jobs**.
2. Click the migration job **vm-to-cloudsql** to see the details page.
3. Review the migration job status.
   * If you have not started the job, the status will show as **Not started**. You can choose to start or delete the job.
   * After the job has started, the status will show as **Starting** and then transition to **Running Full dump in progress** to indicate that the initial database dump is in progress.
   * After the initial database dump has been completed, the status will transition to **Running CDC in progress** to indicate that continuous migration is active.

When the job status changes to **Running CDC in progress**, proceed to the next task.



**Note:** Continuous migration jobs remain in a running status to ensure that the destination database continues to receive data updates from the source.

A completed status is achieved after the destination database is promoted to be a stand-alone database for reading and writing data which you will see in the final task in the lab.

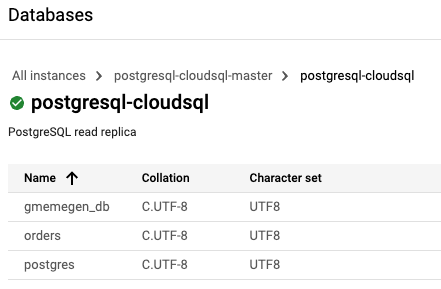
Click *Check my progress* to verify the objective.

**Part4. Confirm the data in Cloud SQL for PostgreSQL**

Check the PostgreSQL databases in Cloud SQL

1. In the Google Cloud Console, on the **Navigation menu** (Navigation menu icon),**SQL**.
2. Expand the instance ID called **postgresql-cloudsql-master**.
3. Click on the instance **postgresql-cloudsql** (PostgreSQL read replica).
4. In the **Replica Instance** menu, click **Databases**.

Notice that the databases called **postgres**, **orders** and **gmemegen\_db** have been migrated to Cloud SQL.



Connect to the PostgreSQL instance

1. In the **Replica Instance** menu, click **Overview**.
2. Scroll down to the **Connect to this instance** section and click **Open Cloud Shell**.

The command to connect to PostgreSQL will pre-populate in Cloud Shell:

gcloud sql connect postgresql-cloudsql --user=postgres --quiet

1. Run the pre-populated command.

If prompted, click **Authorize** for the API.

1. When prompted for a password, which you previously set, enter:

***supersecret!***

You have now activated the PostgreSQL interactive console for the destination instance.

Review the data in the Cloud SQL for PostgreSQL instance

1. To select the database in the PostgreSQL interactive console, run the following command:

***\c orders;***

1. When prompted for a password, enter:

supersecret!

1. Query the distribution\_centers table:

***select \* from distribution\_centers;***

(Output)

longitude | latitude | name | id

-----------+----------+---------------------------------------------+----

-89.9711 | 35.1174 | Memphis TN | 1

-87.6847 | 41.8369 | Chicago IL | 2

-95.3698 | 29.7604 | Houston TX | 3

-118.25 | 34.05 | Los Angeles CA | 4

-90.0667 | 29.95 | New Orleans LA | 5

-73.7834 | 40.634 | Port Authority of New York/New Jersey NY/NJ | 6

-75.1667 | 39.95 | Philadelphia PA | 7

-88.0431 | 30.6944 | Mobile AL | 8

-79.9333 | 32.7833 | Charleston SC | 9

-81.1167 | 32.0167 | Savannah GA | 10

1. Exit the PostgreSQL interactive console by typing:

***\q***

Update stand-alone source data to test continuous migration

1. In Cloud Shell, type the following commands to connect to the source PostgreSQL instance:

***export VM\_NAME=antern-postgresql-vm***

***export PROJECT\_ID=$(gcloud config list --format 'value(core.project)')***

***export POSTGRESQL\_IP=$(gcloud compute instances describe ${VM\_NAME} \***

***--zone=us-east1-c --format="value(networkInterfaces[0].accessConfigs[0].natIP)")***

***echo $POSTGRESQL\_IP***

***psql -h $POSTGRESQL\_IP -p 5432 -d orders -U migration\_admin***

**Note:** The above is an alternative approach to accessing the stand-alone database on the VM instance.

1. When prompted for a password, enter:

DMS\_1s\_cool!

1. In **psql**, enter the following commands:

\***c orders;***

i***nsert into distribution\_centers values(-80.1918,25.7617,'Miami FL',11);***

1. Close the interactive **psql** session:

***\q***

Connect to the Cloud SQL PostgreSQL database to check that updated data has been migrated

1. In Cloud Shell, type the following commands to connect to the destination Cloud SQL PostgreSQL instance:

***gcloud sql connect postgresql-cloudsql --user=postgres --quiet***

1. When prompted for a password, which you previously set, enter the password for the Cloud SQL instance:

***supersecret!***

You have now activated the PostgreSQL interactive console for the destination instance.

Review data in Cloud SQL for PostgreSQL database

1. In Cloud Shell, select the active database in the PostgreSQL interactive console:

**\c orders;**

1. When prompted for a password, which you previously set, enter:

***supersecret!***

1. Query the distribution\_centers table:

***select \* from distribution\_centers;***

(Output)

longitude | latitude | name | id

-----------+----------+---------------------------------------------+----

-89.9711 | 35.1174 | Memphis TN | 1

-87.6847 | 41.8369 | Chicago IL | 2

-95.3698 | 29.7604 | Houston TX | 3

-118.25 | 34.05 | Los Angeles CA | 4

-90.0667 | 29.95 | New Orleans LA | 5

-73.7834 | 40.634 | Port Authority of New York/New Jersey NY/NJ | 6

-75.1667 | 39.95 | Philadelphia PA | 7

-88.0431 | 30.6944 | Mobile AL | 8

-79.9333 | 32.7833 | Charleston SC | 9

-81.1167 | 32.0167 | Savannah GA | 10

-80.1918 | 25.7617 | Miami FL | 11

Note that the new row added on the stand-alone orders database, is now present on the migrated database.

1. Exit the PostgreSQL interactive console:

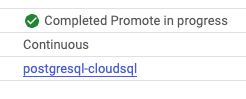
***\q***

**Part 5. Promote Cloud SQL to be a stand-alone instance for reading and writing data**

1. In the Google Cloud Console, on the **Navigation menu** (Navigation menu icon), click **Database Migration** > **Migration jobs**.
2. Click the migration job name **vm-to-cloudsql** to see the details page.
3. Click **Promote**.

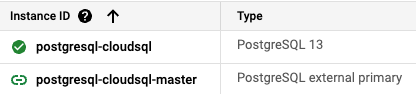
If prompted to confirm, click **Promote**.

When the promotion is complete, the status of the job will update to **Completed**.



1. In the Google Cloud Console, on the **Navigation menu** (Navigation menu icon), click **Databases** > **SQL**.

Note that **postgresql-cloudsql** is now a stand-alone instance for reading and writing data.



**Task 2: Update permissions and add IAM roles to users**

To update permissions and add IAM roles to users for the Cloud SQL instance in GCP using the GCP Console, follow these steps:

### Step-by-Step Instructions

1. \*\*Grant the Antern Editor user the Cloud SQL Instance User role\*\*

- Go to the \*\*IAM & Admin\*\* section in the Google Cloud Console.

- Click on \*\*+ Add\*\* at the top of the IAM page.

- In the \*\*New principals\*\* field, enter `student-04-09b8a3c0447d@qwiklabs.net`.

- In the \*\*Select a role\*\* drop-down menu, choose \*\*Cloud SQL > Cloud SQL Instance User\*\*.

- Click \*\*Save\*\*.

2. \*\*Grant the Cymbal Owner user the Cloud SQL Admin role\*\*

- In the \*\*IAM & Admin\*\* section, click on \*\*+ Add\*\* at the top of the IAM page.

- Enter `student-02-e7544f81bee8@qwiklabs.net` in the \*\*New principals\*\* field.

- Select \*\*Cloud SQL > Cloud SQL Admin\*\* from the \*\*Select a role\*\* drop-down menu.

- Click \*\*Save\*\*.

3. \*\*Change the Cymbal Editor user role from Viewer to Editor\*\*

- In the \*\*IAM & Admin\*\* section, find the user `student-02-02a29db28cc4@qwiklabs.net`.

- Click the \*\*pencil icon\*\* next to their current role to edit the role.

- Remove the \*\*Viewer\*\* role.

- Click \*\*+ Add another role\*\*, and select \*\*Project > Editor\*\*.

- Click \*\*Save\*\*.

### Adding Users to the Cloud SQL Database

1. \*\*Navigate to the Cloud SQL Database\*\*

- Go to the \*\*SQL\*\* section in the Google Cloud Console.

- Click on the Cloud SQL instance you created.

2. \*\*Add the Antern Editor User Account\*\*

- In the instance details page, go to the \*\*Users\*\* tab.

- Click on \*\*Add user account\*\*.

- Use Cloud IAM authentication and for the principal, enter `student-04-09b8a3c0447d@qwiklabs.net`.

- Click \*\*Add\*\*.

3. \*\*Add the Cymbal Owner User Account\*\*

- In the instance details page, go to the \*\*Users\*\* tab.

- Click on \*\*Add user account\*\*.

- Use Cloud IAM authentication and for the principal, enter `student-02-e7544f81bee8@qwiklabs.net`.

- Click \*\*Add\*\*.

### Summary of Steps

1. \*\*IAM & Admin Console\*\*:

- Add the roles to the respective users.

- Remove the Viewer role and add the Editor role for the Cymbal Editor user.

2. \*\*Cloud SQL Console\*\*:

- Add the users to the Cloud SQL database using Cloud IAM authentication.

By following these steps, you will have successfully updated the permissions and added IAM roles to the users for the Cloud SQL database in GCP.

## **Task 3: Create networks and firewalls**

**Note:** For this task, you will need to log in to the **Cymbal Project** with the **Cymbal Owner** credentials.

Let's create the VPC network, subnets, and firewall rules using the `gcloud` command-line tool.

### Step-by-Step Instructions

1. \*\*Create the VPC Network\*\*

```sh

gcloud compute networks create vpc-network-n2yg \

--subnet-mode=custom \

--bgp-routing-mode=regional

```

2. \*\*Create the Subnets\*\*

```sh

gcloud compute networks subnets create subnet-a-9j21 \

--network=vpc-network-n2yg \

--region=us-west1 \

--range=10.10.10.0/24

gcloud compute networks subnets create subnet-b-5yw3 \

--network=vpc-network-n2yg \

--region=us-east4 \

--range=10.10.20.0/24

```

3. \*\*Create Firewall Rules\*\*

```sh

# Firewall rule for SSH

gcloud compute firewall-rules create hhij-firewall-ssh \

--network=vpc-network-n2yg \

--priority=65535 \

--direction=INGRESS \

--action=ALLOW \

--rules=tcp:22 \

--source-ranges=0.0.0.0/0 \

--target-tags=all

# Firewall rule for RDP

gcloud compute firewall-rules create mspz-firewall-rdp \

--network=vpc-network-n2yg \

--priority=65535 \

--direction=INGRESS \

--action=ALLOW \

--rules=tcp:3389 \

--source-ranges=0.0.0.0/0 \

--target-tags=all

# Firewall rule for ICMP

gcloud compute firewall-rules create ricg-firewall-icmp \

--network=vpc-network-n2yg \

--priority=65535 \

--direction=INGRESS \

--action=ALLOW \

--rules=icmp \

--source-ranges=0.0.0.0/0 \

--target-tags=all

```

### Explanation

1. \*\*Creating the VPC Network\*\*:

- The `--subnet-mode=custom` flag indicates that we will define subnets manually.

- The `--bgp-routing-mode=regional` flag sets the dynamic routing mode to regional.

2. \*\*Creating the Subnets\*\*:

- The `subnet-a-9j21` and `subnet-b-5yw3` are created in the `us-west1` and `us-east4` regions respectively, with the specified IP ranges.

3. \*\*Creating Firewall Rules\*\*:

- Each firewall rule is created with a priority of 65535, ingress direction, and action set to allow.

- The rules allow traffic on specific ports for SSH (22), RDP (3389), and ICMP from any IP address (0.0.0.0/0).

Run these commands in the Cloud Shell or your local terminal configured with Google Cloud SDK and authenticated with the Cymbal Owner credentials.

**Task 4**

**Note:** For this task, you will need to log in to the **Cymbal Project** with the **Cymbal Owner** credentials.

To troubleshoot and fix the broken GKE cluster, specifically Bug #2 (stale ratings), and create a BigQuery log sink, follow these steps:

### Step 1: Investigate the Rating Service Logs

1. \*\*Log in to Google Cloud Console\*\*:

- Make sure you are logged in with the Cymbal Owner credentials.

2. \*\*Navigate to App Engine\*\*:

- Go to the App Engine section in the Google Cloud Console.

3. \*\*View Logs\*\*:

- Open the \*\*Logs Explorer\*\*.

- Look for logs with severity `ERROR` to identify issues.

4. \*\*Check main.py\*\*:

- Identify any issues related to the rating service. Another team member mentioned the problem might be in the `main.py` file. Look for error messages or stack traces related to this file.

### Step 2: Create a BigQuery Log Sink

1. \*\*Go to Logs Explorer\*\*:

- In the Google Cloud Console, navigate to \*\*Logging > Logs Explorer\*\*.

2. \*\*Create a Log Sink\*\*:

- Click on \*\*Create Sink\*\*.

- Name the sink `ratings-service-error-sink`.

- For the sink destination, choose \*\*BigQuery dataset\*\*.

- Click on \*\*Create New BigQuery Dataset\*\*.

- Name the dataset `gke\_app\_errors\_sink`.

- Set the location to `us` (multiple regions in the United States).

3. \*\*Set Inclusion Filter\*\*:

- In the filter, include the following fields:

```plaintext

resource.type

resource.labels.module\_id

severity

```

4. \*\*Complete Sink Creation\*\*:

- Follow the prompts to complete the creation of the log sink.

### Step 3: Grant IAM Roles to Users

1. \*\*Grant BigQuery Data Viewer Role\*\*:

- Navigate to \*\*IAM & Admin > IAM\*\*.

- Click on \*\*+ Add\*\*.

- In the \*\*New principals\*\* field, enter `student-04-09b8a3c0447d@qwiklabs.net`.

- In the \*\*Select a role\*\* drop-down, choose \*\*BigQuery > BigQuery Data Viewer\*\*.

- Click \*\*Save\*\*.

2. \*\*Grant BigQuery Admin Role\*\*:

- Navigate to \*\*IAM & Admin > IAM\*\*.

- Click on \*\*+ Add\*\*.

- In the \*\*New principals\*\* field, enter `student-04-28c2e0f9f813@qwiklabs.net`.

- In the \*\*Select a role\*\* drop-down, choose \*\*BigQuery > BigQuery Admin\*\*.

- Click \*\*Save\*\*.

### Step 4: Fix the Rating Service Issue

1. \*\*Navigate to App Engine\*\*:

- In the Google Cloud Console, go to \*\*App Engine\*\*.

2. \*\*Inspect main.py\*\*:

- Locate the `main.py` file in the App Engine service.

- Look for any issues or bugs that could be causing the stale ratings. Common issues might include problems with API endpoint calls or data processing logic.

3. \*\*Fix the Issue\*\*:

- Edit the `main.py` file to fix any identified issues.

- Deploy the updated service.

### Summary

- Investigate logs for the rating service using Logs Explorer.

- Create a BigQuery log sink to capture error logs.

- Grant appropriate IAM roles to users.

- Inspect and fix issues in the `main.py` file of the rating service.

Following these steps should help you identify and fix the issue with the stale ratings in the GKE cluster.