
Migrate to Cloud SQL for PostgreSQL using Database Migration Service

Project Overview

This project involves migrating a PostgreSQL database hosted on a self-managed environment (e.g., on-premises or VM) to **Cloud SQL for PostgreSQL** using **Google Cloud's Database Migration Service (DMS)**. The objective is to leverage managed services on Google Cloud for improved scalability, reliability, and maintenance.

Objectives

- Migrate PostgreSQL database with minimal downtime.
- Validate data consistency and schema integrity.
- Improve scalability and reduce administrative overhead.


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 postgresql-vm VM Instance.

1. In the Google Cloud console, on the **Navigation menu** () , click **Compute Engine > VM instances**.
2. In the entry for 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`

VM instances

Create instanceImport VMRefreshLearn

Instances

Observability

Instance schedules

VM instances

Filter

Enter property name or value

?

⋮

<input type="checkbox"/>	Status	Name ↑	Zone	Recommendations	In us	Connect
<input type="checkbox"/>	✓	postgresql-vm	us-west1-c			SSH ▾ ⋮

```
student-02-ded9db8b7e37@postgresql-vm:~$ sudo apt install postgresql-13-pglogical
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  postgresql-13-pglogical
0 upgraded, 1 newly installed, 0 to remove and 7 not upgraded.
Need to get 434 kB of archives.
After this operation, 1238 kB of additional disk space will be used.
Get:1 http://apt.postgresql.org/pub/repos/apt bullseye-pgdg/main amd64 postgresql-13-pglogical amd64 2.4.5-1.pgdg110+1 [434 kB]
Fetched 434 kB in 0s (6140 kB/s)
Selecting previously unselected package postgresql-13-pglogical.
(Reading database ... 75799 files and directories currently installed.)
Preparing to unpack .../postgresql-13-pglogical_2.4.5-1.pgdg110+1_amd64.deb ...
Unpacking postgresql-13-pglogical (2.4.5-1.pgdg110+1) ...
Setting up postgresql-13-pglogical (2.4.5-1.pgdg110+1) ...
Processing triggers for postgresql-common (278.pgdg110+1) ...
Building PostgreSQL dictionaries from installed myspell/hunspell packages...
Removing obsolete dictionary files:
student-02-ded9db8b7e37@postgresql-vm:~$
```

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

```
student-02-ded9db8b7e37@postgresql-vm:~$ sudo su - postgres -c "gsutil cp gs://cloud-training/gsp918/pg_hba_append.conf ."
student-02-ded9db8b7e37@postgresql-vm:~$ sudo su - postgres -c "gsutil cp gs://cloud-training/gsp918/postgresql_append.conf ."
student-02-ded9db8b7e37@postgresql-vm:~$ sudo su - postgres -c "cat pg_hba_append.conf >> /etc/postgresql/13/main/pg_hba.conf"
student-02-ded9db8b7e37@postgresql-vm:~$ sudo su - postgres -c "cat postgresql_append.conf >> /etc/postgresql/13/main/postgresql.conf"
student-02-ded9db8b7e37@postgresql-vm:~$ sudo systemctl restart postgresql@13-main
systemctl restart postgresql@13-main
Copying gs://cloud-training/gsp918/pg_hba_append.conf...
/ [1 files] 68.0 B / 68.0 B
Operation completed over 1 objects/68.0 B.
Copying gs://cloud-training/gsp918/postgresql_append.conf...
/ [1 files] 543.0 B / 543.0 B
Operation completed over 1 objects/543.0 B.
student-02-ded9db8b7e37@postgresql-vm:~$
```

6. Launch the **psql** tool:

`sudo su - postgres`

`Psql`

```
student-02-ded9db8b7e37@postgresql-vm:~$ sudo su - postgres
postgres@postgresql-vm:~$ psql
psql (13.21 (Debian 13.21-1.pgdg110+1))
Type "help" for help.

postgres=# \c postgres;
You are now connected to database "postgres" as user "postgres".
postgres=# CREATE EXTENSION pglogical;
CREATE EXTENSION
postgres=# CREATE EXTENSION pglogical;
ERROR:  extension "pglogical" already exists
postgres=# \c orders;
You are now connected to database "orders" as user "postgres".
orders=# CREATE EXTENSION pglogical;
CREATE EXTENSION
orders=# █
```

```
CREATE EXTENSION
gmemegen_db=# \l

              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)

gmemegen_db=# █
```

7. Add the pglogical database extension to the postgres, orders and gmemegen_db databases.

\c postgres;

CREATE EXTENSION pglogical;

```
student-02-ded9db8b7e37@postgresql-vm:~$ sudo su - postgres
postgres@postgresql-vm:~$ psql
psql (13.21 (Debian 13.21-1.pgdg110+1))
Type "help" for help.

postgres=# \c postgres;
You are now connected to database "postgres" as user "postgres".
postgres=# CREATE EXTENSION pglogical;
CREATE EXTENSION
postgres=# CREATE EXTENSION pglogical;
ERROR:  extension "pglogical" already exists
postgres=# \c orders;
You are now connected to database "orders" as user "postgres".
orders=# CREATE EXTENSION pglogical;
CREATE EXTENSION
orders=# █
```

8. List the PostgreSQL databases on the server:

```
CREATE EXTENSION
gmemegen_db=# \l
```

```
              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)

gmemegen_db=#
```

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:

```
gmemegen_db=# CREATE USER migration_admin PASSWORD 'DMS_is_cool!';
ALTER DATABASE orders OWNER TO migration_admin;
ALTER ROLE migration_admin WITH REPLICATION;
CREATE ROLE
ALTER DATABASE
ALTER ROLE
gmemegen_db=#
```

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.

[illegible]

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

[illegible]

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

```

orders=# 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;
GRANT
GRANT
GRANT
GRANT
GRANT
GRANT
GRANT
GRANT
orders=# █

```

4. In **psql**, grant permissions to the pglogical schema and tables for the gmemegen_db database.

```

orders=# \c gmemegen_db;
You are now connected to database "gmemegen_db" as user "postgres".
gmemegen_db=# 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;

GRANT
GRANT
GRANT
GRANT
GRANT
GRANT
GRANT
GRANT
GRANT
GRANT
GRANT
GRANT
GRANT
GRANT
gmemegen_db=# █

```

5. In **psql**, grant permissions to the public schema and tables for the gmemegen_db database.

```

gmemegen_db=# GRANT USAGE ON SCHEMA public TO migration_admin;
GRANT ALL ON SCHEMA public TO migration_admin;

GRANT SELECT ON public.meme TO migration_admin;
GRANT
GRANT
GRANT
gmemegen_db=# █

```

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, orders and gmemegen_db 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.

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

```

gmemegen_db=# \c orders;
\dt
You are now connected to database "orders" as user "postgres".
      List of relations
Schema |          Name          | Type  | Owner
-----+-----+-----+-----
public | distribution_centers   | table | postgres
public | inventory_items        | table | postgres
public | order_items            | table | postgres
public | products               | table | postgres
public | users                  | table | postgres
(5 rows)

orders=# 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
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
      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
(5 rows)

orders=# █

```


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

```
orders=# \q
postgres@postgresql-vm:~$ exit
logout
student-02-ded9db8b7e37@postgresql-vm:~$
```

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

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** () , click **Compute Engine > VM instances**.
2. Locate the line with the instance called **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** () , click **VIEW ALL PRODUCTS** under Databases section click on **Database Migration > Connection profiles**.
2. Click **+ Create Profile**.
3. For **Profile Role**, select **Source**.
4. For **Database engine**, select **PostgreSQL**.
5. For **Connection profile name**, enter **postgres-vm**.
6. For **Region** select (region).
7. Under **Define connection configurations** click on **DEFINE**


8. 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)
9. For **Port**, enter **5432**.
10. For **Username**, enter **migration_admin**.
11. For **Password**, enter **DMS_1s_cool!** .
12. For all other values leave the defaults.
13. Click **Create**.

Display name	In use by	Profile role	Database engine	Hostname/IP-Port	Created
postgres-vm	0 jobs	Source	PostgreSQL	10.138.0.2:5432	Jun 24, 2025

Task 3. Create and start a continuous migration job

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** () , click **VIEW ALL PRODUCTS** under Databases section click on **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 (region).
6. For **Destination database engine**, select **Cloud SQL for PostgreSQL**.
7. For **Migration job type**, select **Continuous**.

Leave the defaults for the other settings.

8. 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.

2. 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 (zone) 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.

8. Click **Allocate & Connect**.

VPC Network / Peering connections

VPC networks

IP addresses

Internal ranges

Bring your own IP

Firewall

Routes

VPC network peering

Shared VPC

Serverless VPC access

Packet mirroring

VPC network peering

Create peering connection

Refresh

Delete

Filter

Enter property name or value

<input type="checkbox"/>	Name ↑	Your VPC network	Peered VPC network	Peered project ID	Status	IP stack type	Custom routes
<input type="checkbox"/>	servicenetworking-googleapis-com	default	servicenetworking	tba57960f61b2836b-tp	Active	IPv4	None

9. For **Machine shapes**, check **1 vCPU, 3.75 GB**

10. For **Storage type**, select **SSD**

11. For **Storage capacity**, select **10 GB**

12. Click **Create & Continue**.


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**.

Allow access to the postgresql-vm instance from automatically allocated IP range

In this step you will edit the pg_hba.conf PostgreSQL configuration file to allow the Database Migration Service to access the stand-alone PostgreSQL database.

1. Get the allocated IP address range. In the Google Cloud Console on the **Navigation menu** () , right-click **VPC network** > **VPC network peering** and open it in a new tab.
2. Click on the servicenetworking-googleapis-com entry and then click on **Effective Routes View** at the bottom.
3. From the dropdown for **Network** select **default** and for **Region** select (region). Click **View**.
4. In the **Destination IP range** column ,copy the IP range (e.g. 10.107.176.0/24) next to **peering-route-xxxxx...** route.
5. In the Terminal session on the VM instance, edit the pg_hba.conf file as follows:

```
sudo nano /etc/postgresql/13/main/pg_hba.conf
```

```
GNU nano 5.4 /etc/postgresql/13/main/pg_hba.conf *
#
# If you want to allow non-local connections, you need to add more
# "host" records.  In that case you will also need to make PostgreSQL
# listen on a non-local interface via the listen_addresses
# configuration parameter, or via the -i or -h command line switches.
#
# DO NOT DISABLE!
# If you change this first entry you will need to make sure that the
# database superuser can access the database using some other method.
# Noninteractive access to all databases is required during automatic
# maintenance (custom daily cronjobs, replication, and similar tasks).
#
# Database administrative login by Unix domain socket
local all postgres peer

# TYPE DATABASE USER ADDRESS METHOD
# "local" is for Unix domain socket connections only
local all all peer
# IPv4 local connections:
host all all 127.0.0.1/32 md5
# IPv6 local connections:
host all all ::1/128 md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
local replication all peer
host replication all 127.0.0.1/32 md5
host replication all ::1/128 md5
#GSP918 ~ allow access to all hosts
host all all 10.103.64.0/24 md5

^G Help ^C Write Out ^W Where Is ^K Cut ^T Execute ^C Location M-U Undo
^X Exit ^R Read File ^\ Replace ^U Paste ^U Justify ^_ Go To Line M-E Redo
```

7. Save and exit the nano editor with Ctrl-O, Enter, Ctrl-X

8. 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**.

connect to your destination.

Test run complete

- ✓ Your migration job test was successful! You can create this job without starting it yet, or start it immediately.

Database Migration / Migration jobs / Migration job details: vm-to-cloudsql

vm-to-cloudsql EDIT RESUME STOP RESTART PROMOTE DELETE VIEW LOGS

Migration job name: vm-to-cloudsql
Status: Running
Phase: Full dump
Source/Destination: PostgreSQL / Cloud SQL for PostgreSQL
Migration type: Continuous

SHOW ALL DETAILS

DATABASES MONITORING

3 Databases REFRESH RESTART FAILED DATABASES

Filter Enter property name or value

Name	Status ↑	Phase ?	Errors	Replication delay ?
postgres	Running	Full dump	0 VIEW DETAILS	0 B
orders	Running	Full dump	0 VIEW DETAILS	0 B
gmemegen_db	Running		VIEW DETAILS	0 B

Migration job was started successfully.

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

1. In the Google Cloud Console, on the **Navigation menu** (≡), click **Database Migration > Migration jobs**.
2. Click the migration job **vm-to-cloudsql** to see the details page.
3. Review the migration job status.
 - a. If you have not started the job, the status will show as **Not started**. You can choose to start or delete the job.
 - b. 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.
 - c. 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.

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	Public	Actions
<input type="checkbox"/>	▼ postgresql-cloudsql-master		Enterprise	PostgreSQL external primary		
<input type="checkbox"/>	postgresql-cloudsql		Enterprise	PostgreSQL read replica	35.18	

Task 4. 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** (), click **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**.

All instances > postgresql-cloudsql-master > postgresql-cloudsql

postgresql-cloudsql
 PostgreSQL read replica

Managed by Database Migration Service at [vm-to-cloudsql](#).

This instance's backups settings don't follow your organization policy "Resource Location Restriction" [EDIT SETTINGS](#)

1 hour 6 hours 1 day 7 days 30 days Custom

Chart
CPU utilization

Connect to the PostgreSQL instance

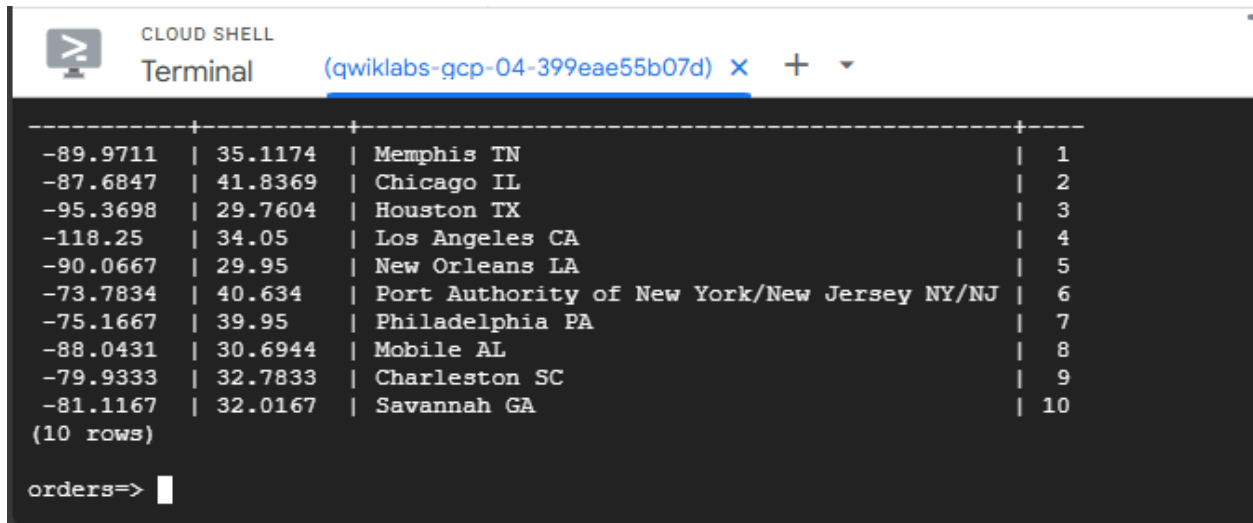
1. In the **Replica Instance** menu, click **Overview**.

2. Scroll down to the **Connect to this instance** section and click **Open Cloud Shell**.
3. Run the pre-populated command.

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

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

Supersecret!



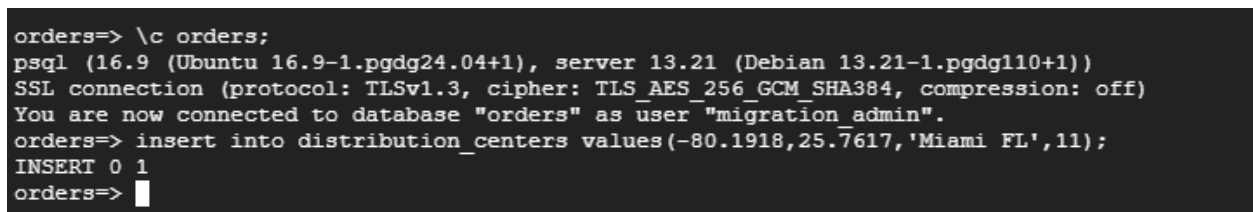
The screenshot shows a Cloud Shell terminal window with the title "CLOUD SHELL Terminal (qwiklabs-gcp-04-399eae55b07d)". The terminal displays a table of distribution centers with 10 rows. The table has four columns: longitude, latitude, city, and an index. The data is as follows:

Longitude	Latitude	City	Index
-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

Below the table, the terminal shows the command prompt "orders=>" with a cursor.

Review the data in the Cloud SQL for PostgreSQL instance

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



The screenshot shows a PostgreSQL terminal session. The user enters the command "orders=> \c orders;" to connect to the "orders" database. The terminal output shows the PostgreSQL version (16.9), server version (13.21), and SSL connection details. The user is now connected to the "orders" database as "migration_admin". The user then enters the command "orders=> insert into distribution_centers values(-80.1918,25.7617,'Miami FL',11);". The terminal output shows "INSERT 0 1", indicating that one row was inserted into the "distribution_centers" table.

3. Query the distribution_centers table:

```
CLOUD SHELL
Terminal (qwiklabs-gcp-04-399eae55b07d) x + v

-----+-----+-----+-----+
-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
(10 rows)

orders=> |
```

4. 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=postgresql-vm export PROJECT_ID=$(gcloud config list --format
'value(core.project)') export POSTGRES_IP=$(gcloud compute instances describe
${VM_NAME}
```

```
--zone=(zone) --format="value(networkInterfaces[0].accessConfigs[0].natIP)") echo
$POSTGRES_IP psql -h $POSTGRES_IP -p 5432 -d orders -U migration_admin
```

2. When prompted for a password, enter:

```
DMS_1s_cool!
```

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

```
\c orders;
```

```
insert into distribution_centers values(-80.1918,25.7617,'Miami FL',11);
```

4. 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:

```
postgres=> \q
student_02_ded9db8b7e37@cloudshell:~ (qwiklabs-gcp-04-399eae55b07d) $ gcloud sql connect postgresql-cloudsql --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.
```

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

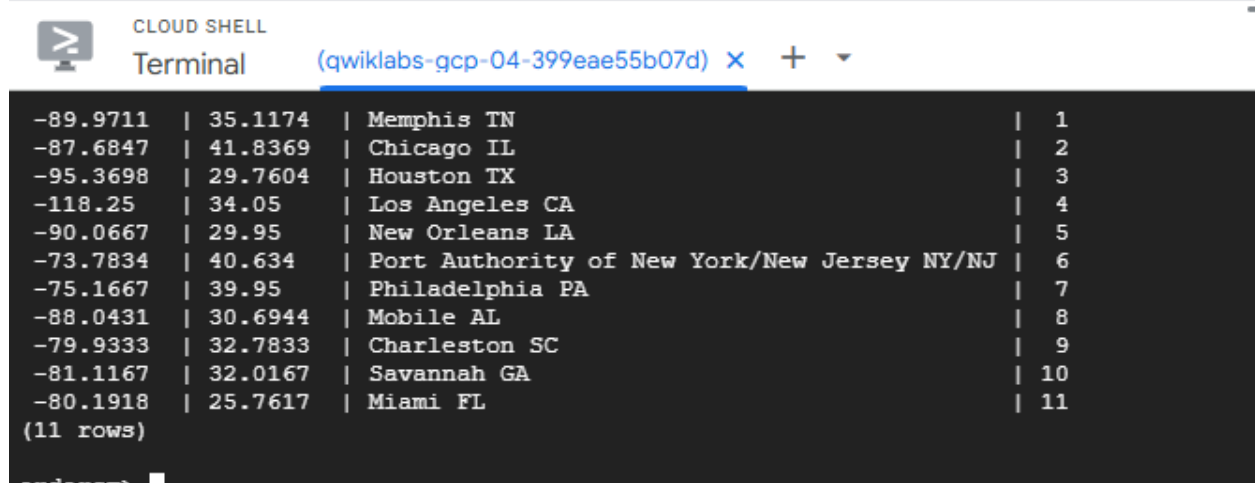
supersecret!

Review data in Cloud SQL for PostgreSQL database

1. In Cloud Shell, select the active database in the PostgreSQL interactive console:
`\c orders;`

2. When prompted for a password, which you previously set, enter:
supersecret!

3. Query the `distribution_centers` table:
`select * from distribution_centers;`




The screenshot shows a Cloud Shell terminal window with the title "CLOUD SHELL" and "Terminal (qwiklabs-gcp-04-399eae55b07d)". The terminal displays the output of a SQL query, showing 11 rows of data. The data is presented in a table-like format with columns separated by vertical bars. The first column contains longitude values, the second contains latitude values, the third contains city and state names, and the fourth contains an index from 1 to 11. The output is as follows:

-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

The terminal also shows "(11 rows)" at the bottom of the output.

Task 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** () , click **VIEW ALL PRODUCTS** under Databases section click on **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**.

The screenshot shows the Google Cloud Database Migration console. The left sidebar contains navigation links: Migration jobs, Conversion workspaces, Connection profiles, and Private connectivity. The main panel displays the 'Migration jobs' page with a table of jobs. The job 'vm-to-cloudsql' is selected, and its details are shown. A dialog box titled 'Promote migration job?' is open, asking for confirmation to promote the job. The dialog text states: 'Promote disconnects the source from the destination, and promotes the destination to be a writeable instance. It's recommended to stop writes to the source, and to wait until the replication delay is at zero before initiating promote.' The dialog has 'CANCEL' and 'PROMOTE' buttons.

Database Migration / Migration jobs

Migration jobs

Migration jobs allow you to move data between source and destination databases. You can define your source by creating a new job, or just its connection profile. [Learn more](#)

JOB (1) DRAFTS (0)

Migration job name	Status	Phase	Source connection profile	Destination ID
vm-to-cloudsql	Running	CDC	postgres-vm	postgresql-cloudsql

vm-to-cloudsql

EDIT RESUME STOP RESTART PROMOTE DELETE VIEW LOGS

Migration job name: vm-to-cloudsql

Status: Running

Phase: CDC

Source/destination: postgres-vm postgresql-cloudsql

Migration type: SHOW ALL DETAILS

DATABASES

3 Databases

Name	Status	Phase	Errors	Replication delay
postgres	Running	CDC	0 VIEW DETAILS	0 B
orders	Running	CDC	0 VIEW DETAILS	0 B
gmemegen_db	Running	CDC	0 VIEW DETAILS	0 B

Conclusion

The migration was successfully completed with minimal downtime using Google Cloud DMS. Cloud SQL provides better performance monitoring, backups, and reliability. This approach significantly simplifies PostgreSQL database management on GCP.