Migrate to Cloud SQL for PostgreSQL using Database Migration Service

Activate Cloud Shell

Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud. Cloud Shell provides command-line access to your Google Cloud resources.

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

2. Click through the following windows:

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

When you are connected, you are already authenticated, and the project is set to your **Project_ID**, `qwiklabs-gcp-00-2a6535be5594`. 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-00-
2a6535be5594
```

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

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

```
gcloud auth list
```

4. Click **Authorize**.

Output:

```
ACTIVE: *
ACCOUNT: student-00-18923908c754@qwiklabs.net

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

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

```
gcloud config list project
```

Output:

```
[core]
project = qwiklabs-gcp-00-2a6535be5594
```

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

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.

2. 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.

2. If necessary, enable the API.

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 `pglogical`database 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
```

Note: `pglogical` is a logical replication system implemented entirely as a PostgreSQL extension. Fully integrated, it requires no triggers or external programs. This alternative to physical replication is a highly efficient method of replicating data using a publish/subscribe model for selective replication. Read more here: https://github.com/2ndQuadrant/pglogical

5. 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.

6. Launch the **psql** tool:

```
sudo su - postgres

psql
```

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

```
\c postgres;


CREATE EXTENSION pglogical;


\c orders;


CREATE EXTENSION pglogical;


\c gmemegen_db;


CREATE EXTENSION pglogical;
```

8. 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;
```

2. 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;
```

3. 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;
```

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

```
\c 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;
```

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

```
GRANT USAGE ON SCHEMA public TO migration_admin;
GRANT ALL ON SCHEMA public TO migration_admin;

GRANT SELECT ON public.meme 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`, `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:

```
\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
```

```
               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)
```

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

```
\q


exit
```

Task 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** (☰), 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 `us-west1`.
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**.

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

Task 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** (☰), 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 `us-west1`.
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 `us-west1-a` 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**.

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.

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**.

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.

3. Click **Configure & Continue**.

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 `us-west1`. 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
```
Copied!

content_copy

6. On the last line of the file:

```
#GSP918 - allow access to all hosts
host    all all 0.0.0.0/0    md5
```
replace the "all IP addresses" range (0.0.0.0/0) with the range copied in point 3 above.

```
#GSP918 - allow access to all hosts
host    all all 10.107.176.0/24    md5
```
Note: The above step is not required to make the migration work, but it is good practice to make the source database more secure during the migration process, and also restricts access after the migration when the migrated database becomes the source of truth.

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
```
Copied!

content_copy

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** (☰), 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.

Task 4. Confirm the data in Cloud SQL for PostgreSQL

Check the PostgreSQL databases in Cloud SQL

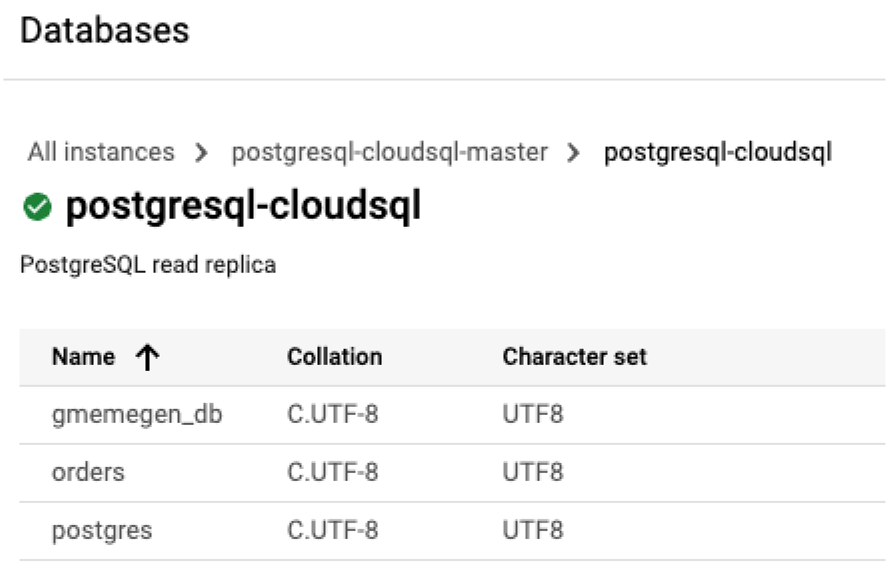
1. In the Google Cloud Console, on the **Navigation menu** (☰), click **SQL**.

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**.

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


Databases

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

postgresql-cloudsql

PostgreSQL read replica

Name ↑	Collation	Character set
gmemegen_db	C.UTF-8	UTF8
orders	C.UTF-8	UTF8
postgres	C.UTF-8	UTF8

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
```

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!
```

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;
```

2. When prompted for a password, enter:

```
supersecret!
```

3. 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
```

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=$(gcloud config list --format 'value(core.project)')
export POSTGRESQL_IP=$(gcloud compute instances describe ${VM_NAME} \
  --zone=us-west1-a --
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.

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:

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

2. 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;
```

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

```
supersecret!
```

3. 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.

4. Exit the PostgreSQL interactive console:

```
\q
```

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

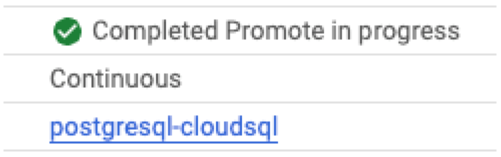
1. In the Google Cloud Console, on the **Navigation menu** (☰), click **VIEW ALL PRODUCTS** under Databases section click on **Database Migration** > **Migration jobs**.

2. Click the migration job name **vm-to-cloudsql** to see the details page.

3. Click **Promote**.

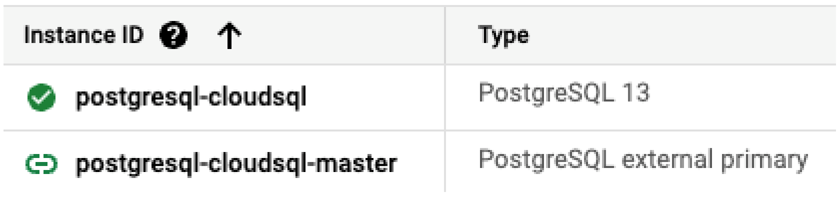
If prompted to confirm, click **Promote**.

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



✓ Completed Promote in progress
Continuous
postgresql-cloudsql

4. In the Google Cloud Console, on the **Navigation menu** (☰), click **Databases** > **SQL**.
Note that **postgresql-cloudsql** is now a stand-alone instance for reading and writing data.



Instance ID	Type
✓ postgresql-cloudsql	PostgreSQL 13
postgresql-cloudsql-master	PostgreSQL external primary