Migrating Cloud Oracle to Cloud SQL for PostgreSQL Using ORA2PG

Objective:

The objective is to migrate the current Oracle database infrastructure to Cloud SQL PostgreSQL using ORA2PG with minimal disruption. The migration process should prioritize maintaining data integrity, scalability, and cost-efficiency while capitalizing on the advantages of cloud-native technologies. Additionally, the aim is to optimize performance and manageability to meet the evolving needs of the organization.

NOTE:- This tutorial is tailored for a technical audience responsible for database management and migration. It assumes familiarity with database administration concepts such as schema conversion and change data capture. Basic knowledge of shell scripts and a fundamental understanding of Google Cloud are recommended prerequisites

Source Database Requirements:

- Version: Ensure your Oracle database version is either 10g v10.2, 11g v11.2.0.4, 12c v12.1.0.2/1, 18c, or 19c.
- Instance Type: Single instance or Real Application Clusters (RAC) are supported. Avoid using the Single Client Access Name (SCAN) listener.
- Multi-tenancy: Container databases (CDB), pluggable databases (PDB), and autonomous databases are not supported.
- Change Replication: LogMiner with supplemental logging is necessary.
- Size: Tables should be under 100 GB. Larger tables can be handled with tools like Ora2Pg.
- Throughput: Aim for about 5 MB per second with a maximum row size of 3 MB. Larger throughputs may cause delays or data loss.
- Network Connectivity: Ensure both the Datastream service and Compute Engine instance can connect to the source Oracle database.

Notable Behaviors and Limitations:

- Primary Key and ROWID: It's recommended to have a primary key on all tables. Migration works for all tables, but note:

- Tables without a primary key: Oracle ROWID is added as a BIGINT primary key column.
- Tables with a primary key: The row ID is copied as an indexed BIGINT column for consistency.

- DML only: No DDL changes are supported during replication.
- Column Support:
  - Unsupported data types are replaced with NULL values.
  - Tables with unsupported data types may not be replicated.
  - Deferred constraints are converted to non-deferred constraints by default.
- Unsupported Features in Oracle:
  - External tables
  - DBLinks
  - Index Only Tables (IOTs)
  - Oracle Label Security (OLS)
- Replication Lag: Datastream uses Oracle LogMiner to capture changes, but recent changes may not be visible until a log switch occurs. Force a log switch before final migration to ensure all changes are captured.

<u>Set up a Google Cloud project:</u>

Follow these steps:

- Create a Google Cloud project
- Enable billing for your Google Cloud project

When you complete the tutorial, you can prevent ongoing billing by deleting the resources you've created. Refer to the "Cleaning up" section at the end of the tutorial for instructions.

<u>Deploy migration resources:</u>

“Type this in the gcloud terminal”

- **Enable APIs:**

  In Cloud Shell, enable APIs for the required services:

```
gcloud services enable \
compute.googleapis.com \
storage.googleapis.com \
dataflow.googleapis.com \
sqladmin.googleapis.com \
pubsub.googleapis.com \
datastream.googleapis.com \
servicenetworking.googleapis.com
```

- **Set Environment Variables:**

  Set up environment variables for your project, region, zone, instance names, passwords, etc.

```
export PROJECT_ID="[YOUR_PROJECT_ID]"
export GCP_REGION_ID="[YOUR_GOOGLE_CLOUD_REGION]"
export GCP_ZONE_ID="[YOUR_GOOGLE_CLOUD_ZONE]"
export
BASTION_VM_NAME="[NAME_OF_BASTION_COMPUTE_ENGIN
E_INSTANCE]"
export GCS_BUCKET="[YOUR_CLOUD_STORAGE_BUCKET]"
export PUBSUB_TOPIC="[YOUR_PUB_SUB_TOPIC]"
export CLOUD_SQL="[CLOUD_SQL_INSTANCE_ID]"
export
CLOUD_SQL_PG_VERSION="[CLOUD_SQL_PG_VERSION]"
export DATABASE_PASSWORD="[DATABASE_PASSWORD]"
```

Create Bastion Compute Engine VM Instance:

```
gcloud compute instances create ${BASTION_VM_NAME} \
    --zone=${GCP_ZONE_ID} \
    --boot-disk-device-name=${BASTION_VM_NAME} \
    --boot-disk-size=10GB \
    --boot-disk-type=pd-balanced \
    --image-family=debian-10 \
    --image-project=debian-cloud
```

Set Up Pub/Sub Notifications:

```
    export PUBSUB_SUBSCRIPTION=${PUBSUB_TOPIC}-subscription
    gcloud pubsub topics create ${PUBSUB_TOPIC} --project=${PROJECT_ID}
    gcloud pubsub subscriptions create ${PUBSUB_SUBSCRIPTION} \
      --topic=${PUBSUB_TOPIC} \
      --project=${PROJECT_ID}
    gsutil notification create -f "json" \
      -p "ora2pg/" \  -t "projects/${PROJECT_ID}/topics/${PUBSUB_TOPIC}" \
    "gs://${GCS_BUCKET}"
```

Create Target Cloud SQL Instance:

```
        gcloud compute addresses create psql-reserve-ip-range \
          --global \
          --purpose=VPC_PEERING \
          --prefix-length=16 \
          --description="Test for Oracle Migration" \
          --network=default \
          --project=${PROJECT_ID}
        gcloud beta sql instances create ${CLOUD_SQL} \
          --database-version=${CLOUD_SQL_PG_VERSION} \
          --cpu=4 --memory=3840MiB \
          --region=${GCP_REGION_ID} \
          --no-assign-ip \
          --network=default \
          --root-password=${DATABASE_PASSWORD} \
          --project=${PROJECT_ID}
```

NOTE: The below code or command is very important in aspect of bucket access bucket in the

```
SERVICE_ACCOUNT=$(gcloud sql instances describe ${CLOUD_SQL} --project=${PROJECT_ID} | grep 'serviceAccountEmailAddress' | awk '{print $2;}')

gsutil iam ch serviceAccount:${SERVICE_ACCOUNT}:objectViewer "gs://${GCS_BUCKET}"
```

Set up the bastion VM with Oracle Access:

Make sure that your vm IP address must be static and it is whitelisted in aws oracle RDP

- **Connect to the Bastion VM with SSH:**

```
gcloud compute ssh ${BASTION_VM_NAME} \
   --zone ${GCP_ZONE_ID} \
   --project ${PROJECT_ID}
```

- **Install Docker on the VM:**

```
sudo apt-get update -y
sudo apt-get install -y \
   apt-transport-https \
   ca-certificates \
   curl \
   gnupg-agent \
   software-properties-common
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key add -
sudo add-apt-repository -y \
   "deb [arch=amd64] https://download.docker.com/linux/debian \
   $(lsb_release -cs) \
    stable"
sudo apt-get update -y
sudo apt-get install -y docker-ce docker-ce-cli
sudo usermod -aG docker ${USER}
```

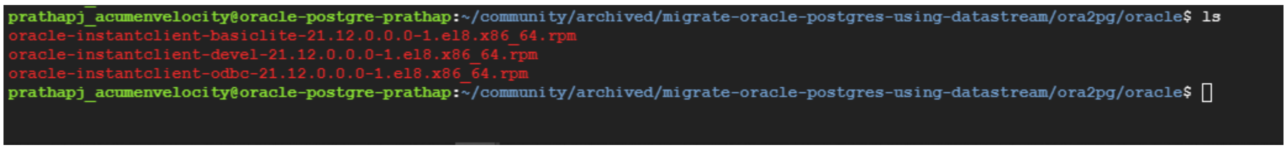
- **NOTE :-** Exit the current VM shell and start a new shell session to pick up the new Docker user group.
- **Set environment variables:**

```
export PROJECT_ID="[YOUR_PROJECT_ID]"
export GCP_REGION_ID="[YOUR_GOOGLE_CLOUD_REGION]"
export GCP_ZONE_ID="[YOUR_GOOGLE_CLOUD_ZONE]"
export GCS_BUCKET="[YOUR_CLOUD_STORAGE_BUCKET]"
export CLOUD_SQL="[YOUR_CLOUD_SQL_INSTANCE_ID]"
export ORA2PG_VERSION="[ORA2PG_VERSION]"
export ORACLE_ODBC_VERSION="[ORACLE_ODBC_VERSION]"
export
ORACLE_SCHEMAS="[SPACE_SEPARATED_ORACLE_SCHEMA_
LIST]"
export ORACLE_TYPES="[SPACE_SEPARATED_OBJECT_TYPES]"
```

- **Install git and clone the repository:**

```
sudo apt-get install git -y
git clone https://github.com/GoogleCloudPlatform/community.git
```

- **Download Oracle Instant Client packages:**
  - Go to the Oracle Instant Client download page.
  - Download the RPM packages for your source database version.
  - Copy the RPM files to community/tutorials/migrate-oracle-postgres-using-datastream/ora2pg/oracle/ on the bastion VM.



```
prathapj_acumenvelocity@oracle-postgre-prathap:~/community/archived/migrate-oracle-postgres-using-datastream/ora2pg/oracle$ ls
oracle-instantclient-basiclite-21.12.0.0.0-1.el8.x86_64.rpm
oracle-instantclient-devel-21.12.0.0.0-1.el8.x86_64.rpm
oracle-instantclient-odbc-21.12.0.0.0-1.el8.x86_64.rpm
prathapj_acumenvelocity@oracle-postgre-prathap:~/community/archived/migrate-oracle-postgres-using-datastream/ora2pg/oracle$ 
```

- **Go to the ora2pg directory:**

```
cd ~/community/archived/migrate-oracle-postgres-using-datastream/ora2pg
```

- **Build the Ora2Pg Docker image:**

```
docker build . \
    -f Ora2PGDockerfile \
    -t ora2pg \
    --build-arg ORA2PG_VERSION=${ORA2PG_VERSION} \
--build-arg
ORACLE_ODBC_VERSION=${ORACLE_ODBC_VERSION}
```

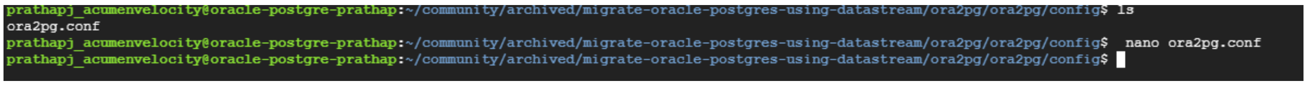
NOTE: If you are fail to create the docker try to build it in single shot don't build recursively

Perform schema conversion:

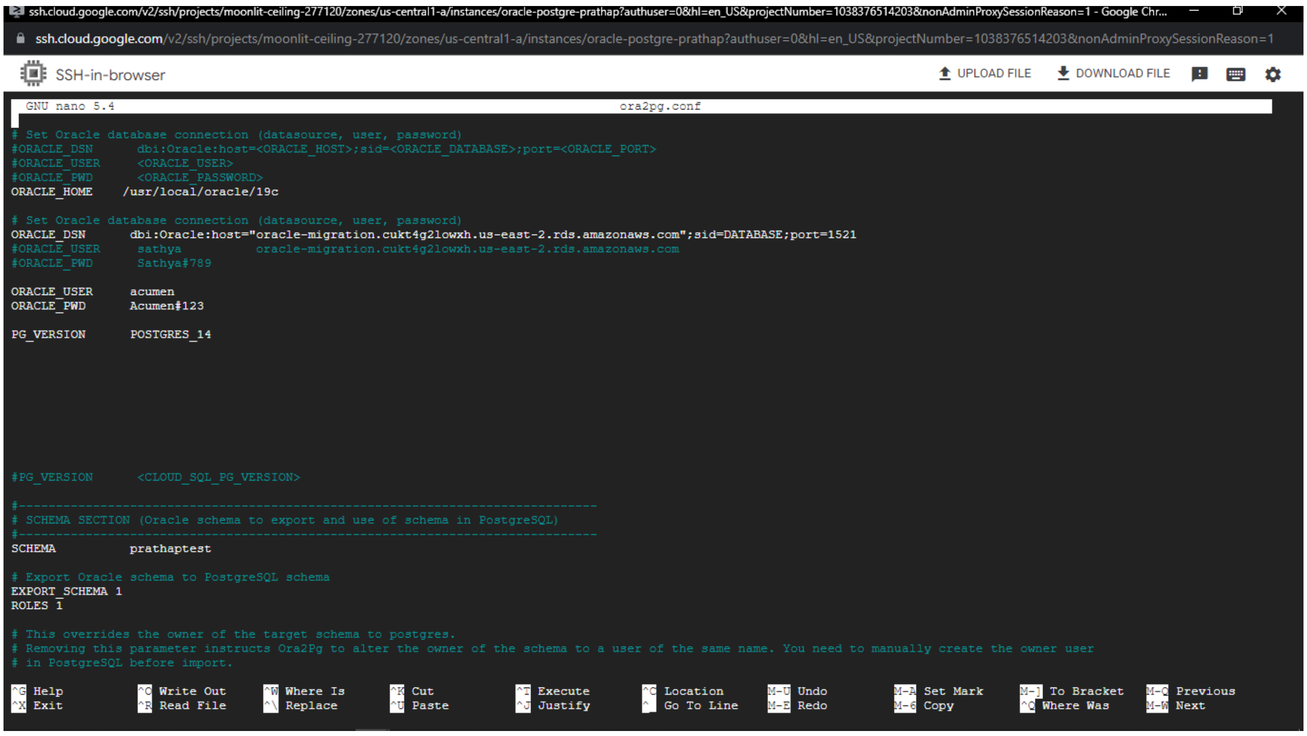
The wrapper script ora2pg.sh takes the ORACLE_SCHEMAS and ORACLE_TYPES environment variables as input and runs the ora2pg container to perform the schema export and conversion process. The result is a PostgreSQL-compliant SQL file stored in ora2pg/data/output.sql. This file is then uploaded to the Cloud Storage bucket and imported to the target PostgreSQL instance.

Steps to Schema conversion:

- On the bastion VM, edit the community/tutorials/migrate-oracle-postgres-using-datastream/ora2pg/ora2pg/config/ora2pg.conf configuration file to set up database connection details, target PostgreSQL version, and Oracle schema to export.



```
prathapj_acumenvelocity@oracle-postgre-prathap:~/community/archived/migrate-oracle-postgres-using-datastream/ora2pg/ora2pg/config$ ls
ora2pg.conf
prathapj_acumenvelocity@oracle-postgre-prathap:~/community/archived/migrate-oracle-postgres-using-datastream/ora2pg/ora2pg/config$  nano ora2pg.conf
prathapj_acumenvelocity@oracle-postgre-prathap:~/community/archived/migrate-oracle-postgres-using-datastream/ora2pg/ora2pg/config$
```

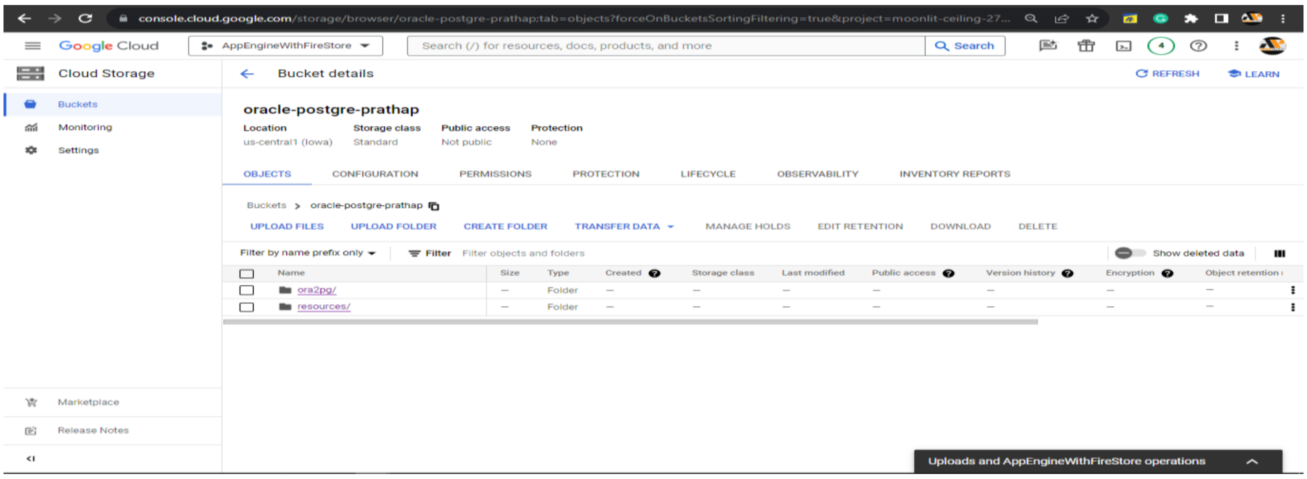


ssh.cloud.google.com/v2/ssh/projects/moonlit-ceiling-277120/zones/us-central1-a/instances/oracle-postgre-prathap?authuser=0&hl=en_US&projectNumber=1038376514203&nonAdminProxySessionReason=1 - Google Chr...
ssh.cloud.google.com/v2/ssh/projects/moonlit-ceiling-277120/zones/us-central1-a/instances/oracle-postgre-prathap?authuser=0&hl=en_US&projectNumber=1038376514203&nonAdminProxySessionReason=1
SSH-in-browser
UPLOAD FILE
DOWNLOAD FILE

```
  GNU nano 5.4                                          ora2pg.conf

# Set Oracle database connection (datasource, user, password)
#ORACLE_DSN      dbi:Oracle:host=<ORACLE_HOST>;sid=<ORACLE_DATABASE>;port=<ORACLE_PORT>
#ORACLE_USER     <ORACLE_USER>
#ORACLE_PWD      <ORACLE_PASSWORD>
ORACLE_HOME    /usr/local/oracle/19c

# Set Oracle database connection (datasource, user, password)
ORACLE_DSN      dbi:Oracle:host="oracle-migration.cukt4g2lowxh.us-east-2.rds.amazonaws.com";sid=DATABASE;port=1521
#ORACLE_USER     sathya          oracle-migration.cukt4g2lowxh.us-east-2.rds.amazonaws.com
#ORACLE_PWD      Sathya#789

ORACLE_USER     acumen
ORACLE_PWD      Acumen#123

PG_VERSION      POSTGRES_14






#PG_VERSION      <CLOUD_SQL_PG_VERSION>

#------------------------------------------------------------------------------
# SCHEMA SECTION (Oracle schema to export and use of schema in PostgreSQL)
#------------------------------------------------------------------------------
SCHEMA          prathaptest

# Export Oracle schema to PostgreSQL schema
EXPORT_SCHEMA 1
ROLES 1

# This overrides the owner of the target schema to postgres.
# Removing this parameter instructs Ora2Pg to alter the owner of the schema to a user of the same name. You need to manually create the owner user
# in PostgreSQL before import.

^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo       M-A Set Mark   M-] To Bracket M-Q Previous
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^_ Go To Line  M-E Redo       M-6 Copy       ^Q Where Was   M-W Next
```

"**NOTE:** Status of the AWS Route was set to Blackhole. Due to this, client couldn't access AWS RDS database. The solution is to make the status of Route to "Active" by setting appropriate Destination and Target for the Route under VPC "Route tables" option.(Iin AWS make 0.0.0.0/0)"

- Run the ora2pg.sh wrapper script to perform schema export and conversion:
  - cd ~/community/tutorials/migrate-oracle-postgres-using-datastream/ora2pg
  - ./ora2pg.sh
- (Optional) Examine the schema conversion result stored in ora2pg/data/output.sql:
  - less ora2pg/data/output.sql
- Upload the script to Cloud Storage:
  - gsutil cp ora2pg/data/output.sql "gs://${GCS_BUCKET}/resources/ora2pg/output.sql"


console.cloud.google.com/storage/browser/oracle-postgre-prathap;tab=objects?forceOnBucketsSortingFiltering=true&project=moonlit-ceiling-27...

Bucket details

oracle-postgre-prathap

Location	Storage class	Public access	Protection
us-central1 (Iowa)	Standard	Not public	None

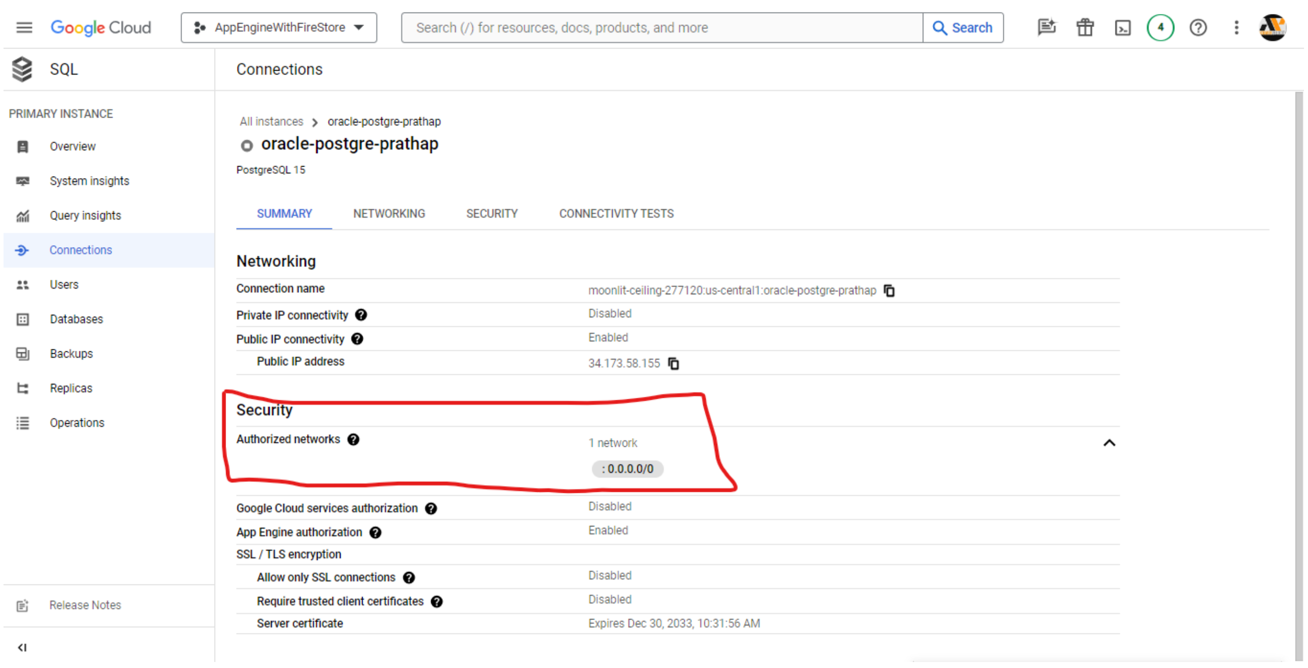
Buckets > oracle-postgre-prathap

Name	Size	Type	Created	Storage class	Last modified	Public access	Version history	Encryption	Object retention
ora2pg/	—	Folder	—	—	—	—	—	—	—
resources/	—	Folder	—	—	—	—	—	—	—

- **NOTE:** : if you face the access denied error while doing upload output.sql to cloud run the below code :
  - SERVICE_ACCOUNT=$(gcloud sql instances describe ${CLOUD_SQL} --project=${PROJECT_ID} | grep 'serviceAccountEmailAddress' | awk '{print $2;}')
  - gsutil iam ch serviceAccount:${SERVICE_ACCOUNT}:objectViewer "gs://${GCS_BUCKET}"
- From Cloud Shell, import the conversion result into the target Cloud SQL for PostgreSQL instance to create the destination schema:

```
gcloud sql import sql \
  ${CLOUD_SQL} \
  "gs://${GCS_BUCKET}/resources/ora2pg/output.sql" \
  --user=postgres \
  --project=${PROJECT_ID} \
  --database=postgres \
  --quiet
```

NOTE: Make sure to add new network 0.0.0.0/0 to allow access all the incoming IP address



oracle-postgre-prathap
PostgreSQL 15

Networking

Connection name	moonlit-ceiling-277120:us-central1:oracle-postgre-prathap
Private IP connectivity	Disabled
Public IP connectivity	Enabled
Public IP address	34.173.58.155

Security

Authorized networks	1 network : 0.0.0.0/0
Google Cloud services authorization	Disabled
App Engine authorization	Enabled
SSL / TLS encryption	
Allow only SSL connections	Disabled
Require trusted client certificates	Disabled
Server certificate	Expires Dec 30, 2033, 10:31:56 AM

Set up a Dataflow job to replicate data from Cloud Storage into the target PostgreSQL database:

To set up Datastream for capturing data from the source Oracle database and migrating it to a Cloud Storage bucket, follow these simplified steps:

1. **Create a Connection Profile for the Source Oracle Database:**
   a. Go to the Datastream connection profiles page.
   b. Click Create profile and select Oracle.
   c. Set connection details:
   d. Profile name: orclsrc
   e. Hostname/IP, Port, SID, Username, Password.
   f. Choose a Google Cloud region close to the source Oracle database.

   g. Test the connection and click Create.
2. **Create a Connection Profile for the Cloud Storage Bucket:**
   a. Go to the Datastream connection profiles page.
   b. Click Create profile and select Cloud Storage.
   c. Set connection details:
   d. Profile name: ora2pggcs
   e. Bucket name: [GCS_BUCKET]
   f. Path prefix: /ora2pg
   g. Select the same Google Cloud region chosen for the source connection profile.
   h. Click Create.
3. **Create a Stream to Capture Data:**
   a. Go to the Datastream Streams page.
   b. Click Create stream.
   c. Set stream details:
   d. Stream name: ora2pgstream
   e. Stream ID: ora2pgstream
   f. Region: Choose the same Google Cloud region as the source connection profile.
   g. Source: Oracle, Destination: Cloud Storage.
   h. Choose the source connection profile (orclsrc).
   i. Specify source details: Include Oracle schema (e.g., DEMOAPP.*).
   j. Choose the destination connection profile (ora2pggcs).
   k. Keep default settings for stream destination.
   l. Validate the stream setup.
   m. Click Create & start to complete setup and start the Datastream job

These steps ensure that Datastream captures changes from the Oracle database and stores them in the designated Cloud Storage bucket, facilitating the migration process

Set up a Dataflow job to replicate data from Cloud Storage into the target PostgreSQL database:

A Dataflow Flex Template reads the Avro-formatted files output by the Datastream stream as they are written and applies them to the target PostgreSQL database.

```
export NEW_UUID=$(cat /proc/sys/kernel/random/uuid) && \

export DATAFLOW_JOB_PREFIX=ora2pg && \
export DATAFLOW_JOB_NAME="${DATAFLOW_JOB_PREFIX}-${NEW_UUID}" && \

export DATABASE_HOST=$(gcloud sql instances list --project=${PROJECT_ID} | grep "${CLOUD_SQL}" | awk '{print $6;}') && \

export GCS_STREAM_PATH="gs://${GCS_BUCKET}/ora2pg/" && \

export DATABASE_PASSWORD="<TARGET_PG_DATABASE_PASSWORD>" && \


gcloud beta dataflow flex-template run "${DATAFLOW_JOB_NAME}" \

   --project="${PROJECT_ID}" --region="${GCP_REGION_ID}" \

   --template-file-gcs-location="gs://dataflow-templates/latest/flex/Cloud_Datastream_to_SQL" \

   --parameters inputFilePattern="${GCS_STREAM_PATH}",\gcsPubSubSubscription="projects/${PROJECT_ID}/subscriptions/${PUBSUB_TOPIC}-subscription",\

databaseHost=${DATABASE_HOST},\

databasePort="5432",\

databaseUser="postgres",\

databasePassword="${DATABASE_PASSWORD}",\

schemaMap=":",\

maxNumWorkers=10,\

autoscalingAlgorithm="THROUGHPUT_BASED”
```

NOTE: after running the dataflow job if the job running continuously and data is not reflected in the database check the Bucket path, DataStream must be running, make sure to set the SUP/SUB notification

If you are facing any issue refer this github link :

https://github.com/GoogleCloudPlatform/community/blob/master/archived/migrate-oracle-postgres-using-datastream/index.md