Migrating an Oracle database to Cloud SQL for PostgreSQL using Striim

In this tutorial, you use Striim (https://www.striim.com/) to migrate Oracle® Database Enterprise Edition 18c (https://docs.oracle.com/en/database/oracle/oracle-database/18/) or later from either an on-premises environment or a cloud environment to a Cloud SQL for PostgreSQL (/sql) instance on Google Cloud. The tutorial uses tables in the Oracle HR sample schema (https://docs.oracle.com/en/database/oracle/oracle-database/18/comsc/introduction-to-sample-schemas.html#GUID-4DE9844F-0B28-4713-9AFC-CCD8D6249D76) .

This tutorial is for enterprise database architects, database engineers, and data owners who plan to use Striim to migrate or replicate Oracle databases to Cloud SQL for PostgreSQL. You should have a basic understanding of how to use Striim to build pipelines. You should also be familiar with the Striim web interface (https://www.striim.com/docs/en/web-ui-overview.html), Striim's key concepts (https://www.striim.com/docs/en/striim-concepts.html), and how to create an application using Striim's Flow Designer (https://www.striim.com/docs/en/creating-or-modifying-apps-using-the-flow-designer.html).

Striim is a Google Cloud database migration technology partner. Striim simplifies online migrations by using a drag-and-drop interface to set up continuous data movement between databases. For migrations to Google Cloud, Striim offers a non-intrusive streaming platform for extract, transform, and load (ETL) that's efficient to deploy and straightforward to iterate. To build the migration pipeline, you use Striim's Flow Designer (https://www.striim.com/docs/en/creating-or-modifying-apps-using-the-flow-designer.html) throughout this tutorial.

If database migration is not something you are familiar with, see this tech talk from Cloud Next '19 (https://www.youtube.com/watch?v=PDFC7XpGAVU) and Architecting database migration and replication using Striim (/solutions/architecting-database-migration-replication-striim).

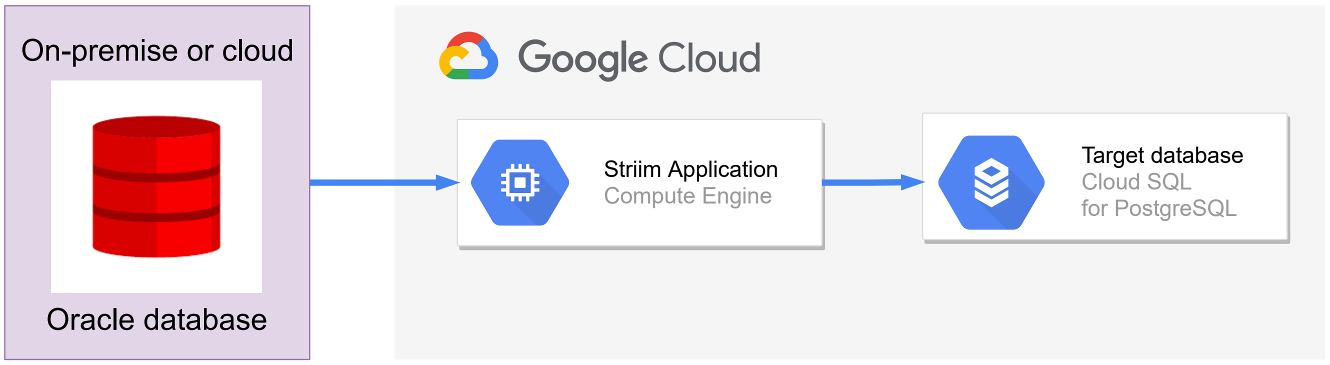
Architecture

Database migration using Striim involves two stages of sequential data movement:

- **Stage 1:** A one-time, initial replication of the Oracle database.

- **Stage 2:** The continuous replication of every change committed on the source database system thereafter by using change data capture (CDC).

The following diagram illustrates a basic deployment architecture:



This architecture involves running the Striim application on a Compute Engine instance. It connects to an Oracle database that's hosted on-premises or in the cloud, and writes data to a Cloud SQL for PostgreSQL instance on Google Cloud.

To avoid any network or connectivity issues between the Striim and Cloud SQL instances, use the same network for both instances. You can deploy Striim from Google Cloud Marketplace on a Compute Engine instance (https://www.striim.com/docs/en/deploying-striim-in-the-google-cloud-platform.html) or, if you need high-availablilty, you can deploy Striim as a cluster (https://www.striim.com/docs/en/creating-a-cluster-in-centos.html).

For this tutorial, deploy from Cloud Marketplace.

The advantage of deploying Striim from Cloud Marketplace is that it lets you connect to various databases and data sources using their built-in adapters (https://www.striim.com/docs/en/adapters-guide.html). You can connect the adapters by using Flow Designer, Striim's interactive, drag-and-drop interface, to form an acyclic graph (https://mathworld.wolfram.com/AcyclicGraph.html). This graph is also known as a *Striim pipeline* or a *Striim application*.

The migration use case in this tutorial uses three Striim adapters:

- Database Reader (https://www.striim.com/docs/en/database-reader.html): Reads data from the Oracle source database during the initial load stage.
- Oracle Reader (https://www.striim.com/docs/en/oracle-reader-properties.html): Reads data using LogMiner

(https://docs.oracle.com/en/database/oracle/oracle-database/19/sutil/oracle-logminer-utility.html) from the Oracle source database during the continuous data replication stage.

- Database Writer (https://www.striim.com/docs/en/database-writer.html): Writes data to the Cloud SQL for PostgreSQL database during the initial load and during the continuous data replication.

Objectives

- Prepare your Oracle database as a source database for migration or replication.
- Prepare a Cloud SQL for PostgreSQL database as the target database for migration or replication.
- Fulfill the prerequisites for installing and running Striim.
- Convert the schema of the Oracle database to the corresponding schema in PostgreSQL.
- Perform the initial load from your Oracle database to Cloud SQL for PostgreSQL.
- Set up the continuous replication from your Oracle database to Cloud SQL for PostgreSQL.

Costs

This tutorial uses the following billable components of Google Cloud:

- Compute Engine (/compute/all-pricing)
- Cloud SQL for PostgreSQL (/sql/pricing#pg-pricing)

To generate a cost estimate based on your projected usage, use the pricing calculator (/products/calculator).

The Striim solution in Cloud Marketplace offers a limited-term free-trial license. When the trial expires, usage charges are billed to your Google Cloud account. You can also obtain Striim licenses directly from Striim for on-premises deployment and in a Compute Engine virtual machine (VM). You also might incur costs that are associated with running an Oracle database outside Google Cloud.

Before you begin

1. In the Google Cloud Console, on the project selector page, select or create a Google Cloud project.

★ **Note**: If you don't plan to keep the resources that you create in this procedure, create a project instead of selecting an existing project. After you finish these steps, you can delete the project, removing all resources associated with the project.

Go to project selector (https://console.cloud.google.com/projectselector2/home/dashboard)

2. Make sure that billing is enabled for your Cloud project. Learn how to confirm that billing is enabled for your project (/billing/docs/how-to/modify-project).

This tutorial assumes you already have the following:

- An Oracle Database Enterprise Edition 18c or later for Linux x86-64 that you want to migrate.
- A Compute Engine running CentOS that has Striim installed. You can deploy Striim through the Google Cloud Marketplace (https://console.cloud.google.com/marketplace/product/striim/striim) solution.

Preparing the Oracle database

The following sections discuss configuration changes that you might need to connect to your Oracle database and migrate it with Striim. For configuration details, see the Striim documentation (https://www.striim.com/docs/en/oracle-configuration.html).

Choose the source for Oracle CDC

While there are different Oracle CDC sources, this tutorial uses LogMiner (https://docs.oracle.com/en/database/oracle/oracle-database/19/sutil/oracle-logminer-utility.html#GUID-3417B738-374C-4EE3-B15C-3A66E01AE2B5)
. You can read about alternate options in Alternate Oracle CDC sources (#alternate_oracle_cdc_sources).

Prepare the Oracle Database Enterprise Edition 18c (or later)

To prepare the Oracle database:

1. Enable Striim's `archivelog` (https://www.striim.com/docs/en/enabling-archivelog.html).
2. Enable Striim supplemental log data (https://www.striim.com/docs/en/enabling-supplemental-log-data.html).
3. Enable Striim primary key logging (https://www.striim.com/docs/en/enabling-supplemental-log-data.html).
4. Create an Oracle user with LogMiner privileges for Striim (https://www.striim.com/docs/en/creating-an-oracle-user-with-logminer-privileges.html).

   To run these steps, you must be connected to the container database (CDB), regardless of whether you're migrating a CDB or a pluggable database (PDB).
5. Create a Striim `quiescemarker` table (https://www.striim.com/docs/archive/394GA/en/UUID-448a48ac-b05d-4c3c-a4cb-14a980426ba9.html)

   .

   Striim's Oracle Reader adapter for CDC needs a table for storing metadata when it quiesces an application. If you use LogMiner as a source for CDC (as this tutorial does), then you need the quiescemarker table. You must be connected to the CDB when following the steps to create the table.
6. Establish network connectivity between your Oracle database and the Striim instance.

   By default, the Oracle listener (https://docs.oracle.com/cd/B10501_01/network.920/a96580/listener.htm) is on port `1521`. Ensure that the IP address for the Striim instance is allowed to connect to the Oracle listener port and that no firewall rules block it. The port on which the Oracle listener is configured is in the `$ORACLE_HOME/network/admin/tnsnames.ora` file.
7. Note the system change number (SCN) for the Oracle database (https://docs.oracle.com/cd/E11882_01/server.112/e40540/transact.htm#CNCPT039).

   The SCN is an internal timestamp that's used to reference changes made to a database.

   On your Oracle database, get the oldest SCN:

```
SELECT MIN(start_scn) FROM gv$transaction;
```

Copy this number. You'll need it later in the continuous replication pipeline steps.

Preparing the Striim instance

For information about the operating systems that Striim supports, see Installing Striim (https://www.striim.com/docs/en/installing-striim.html). To use the Oracle Reader with LogMiner, put the Oracle JDBC driver in the Java classpath (https://www.striim.com/docs/en/installing-the-oracle-jdbc-driver.html) in your Striim instance. Perform the following steps on each Striim server that runs an Oracle Reader adapter:

1. Log in to your Oracle account (https://www.oracle.com/cloud/sign-in.html), and then download the `ojdbc8.jar` file (https://www.oracle.com/database/technologies/jdbc-ucp-122-downloads.html#license-lightbox) on your local machine.
   - If you don't have an Oracle account, create one (https://profile.oracle.com/myprofile/account/create-account.jspx).
2. Click the **Download** link for the `ojdbc8.jar` file.
   - Click **I reviewed and accept the Oracle License Agreement** to download the file, if you accept the license terms.
3. In Cloud Shell, create a Cloud Storage bucket, and upload the `.jar` file to it:

```
gsutil mb -b on -l REGION gs://BUCKET_NAME
gsutil cp PATH/ojdbc8.jar gs://BUCKET_NAME
```

   Replace the following:

   - ***REGION***: the region where you want to create the Cloud Storage Bucket
   - ***BUCKET_NAME***: the name of the Cloud Storage bucket where you want to store the `ojdbc8.jar`

(https://www.oracle.com/database/technologies/jdbc-ucp-122-downloads.html#license-lightbox)
file

- ***PATH***: the path to where you downloaded the **`ojdbc8.jar`** file

After the file is saved on your local machine, we recommend that you upload the **`.jar`** file to a Cloud Storage (/storage) bucket so that you can download it to any instance.

4. Open an SSH session with your Striim instance (/compute/docs/instances/connecting-to-instance#console), and then download the **`.jar`** file onto your Striim instance and place it in the **`/opt/striim/lib`** directory:

```
sudo su - striim gsutil cp gs://BUCKET_NAME/ojdbc8.jar /opt/striim/lib
```

5. Verify that the **`ojdbc8.jar`** file has the correct file permissions:

```
 sudo ls -l /opt/striim/lib/ojdbc8.jar
```

The output should appear as follows:

`-rwxrwx--- striim striim`

6. (Optional) If the **`.jar`** file doesn't have the preceding permissions, set the correct permissions:

```
sudo chmod 770 /opt/striim/lib/ojdbc8.jar
sudo chown striim /opt/striim/lib/ojdbc8.jar
sudo chgrp striim /opt/striim/lib/ojdbc8.jar
```

7. Stop and restart Striim.

After making any configuration changes (such as the preceding permission changes), you must restart Striim.

- If you're using the CentOS 7 Linux distribution, stop Striim:

```
sudo systemctl stop striim-node
sudo systemctl stop striim-dbms
```

- If you're using the CentOS 7 Linux distribution, start Striim:

```
sudo systemctl start striim-dbms
sudo systemctl start striim-node
```

If you want to learn more about stopping and restarting Striim for a different operating system, see Starting and stopping Striim (https://www.striim.com/docs/en/starting-and-stopping-striim.html)

8. Install the psql client (/sql/docs/postgres/connect-admin-ip#install-mysql-client) on the Striim instance.

   You use this client to connect to the Cloud SQL instance and create schemas later in this tutorial.

Preparing the Cloud SQL for PostgreSQL schema

When you copy or continuously replicate tabular data from one database to another, Striim typically requires that the target database contains corresponding tables with the correct schema. Google Cloud doesn't have a utility to prepare the schema, but you can use the schema conversion utility from Striim (https://www.striim.com/docs/en/using-the-schema-conversion-utility.html) or an open source utility like ora2pg (/community/tutorials/migrate-oracle-postgres-using-ora2pg).

Maintain foreign keys during the initial load

During the initial load phase, pay attention to the treatment of foreign keys. Foreign keys establish the relationship between the tables in a relational database. The out-of-order creation or insertion of a foreign key into the target database might destroy the relationship between the two tables. If the integrity between the two databases is compromised, errors might occur. Therefore, it's important to output all foreign key declarations into a separate file during the schema export later in this section.

During continuous replication in CDC pipelines, the source database events are propagated to the target database in the order they occur. If you correctly maintain foreign keys on your source, foreign key operations replicate from the source to the target database in the same order.

In contrast, the initial load pipeline defaults to loading your tables in alphabetical order. If you don't disable foreign keys before the initial load, foreign key violation errors occur. To replicate data during the initial load from the source database tables to the target tables on Cloud SQL for PostgreSQL, you must disable foreign key constraints on the tables. Otherwise, the constraints might be violated during the replication process.

As of June 2021, Cloud SQL for PostgreSQL doesn't support configuration options to disable foreign key constraints.

To handle foreign key constraints:

1. Output all foreign key declarations into a separate file during the schema export.
2. Create table schemas in the Cloud SQL for PostgreSQL database without the foreign key constraints.
3. Complete the initial data replication.
4. Apply the foreign key constraints on the tables.
5. Create the continuous replication pipeline.

This tutorial offers two options for schema conversion, which the following sections explain:

- The Striim schema conversion utility (https://www.striim.com/docs/en/using-the-schema-conversion-utility.html) (recommended)
- The Oracle to PostgreSQL database schema converter (Ora2Pg) (http://ora2pg.darold.net/documentation.html)

Convert the schema using Striim's schema conversion utility

Use Striim's schema conversion utility to prepare Cloud SQL for PostgreSQL to integrate data with the target schema and create tables that reflect the source Oracle database.

The Striim schema conversion tool converts the following source objects into equivalent target objects:

- Tables
- Primary keys
- Data types
- Unique constraints
- `NOT NULL` constraints
- Foreign keys

Using Striim's schema conversion utility, you can analyze the source database and generate DDL scripts to create equivalent schemas in the target database.

We recommend that you manually create the schema in the target database by using the generated DDL scripts. It's easiest to select a subset of your tables, export the schema, and then import the schema into your target Cloud SQL for PostgreSQL database.

The following example demonstrates how to get your target Cloud SQL for PostgreSQL database ready for the initial load by importing your schema using Striim's schema conversion utility:

1. Open an SSH connection (/compute/docs/instances/connecting-to-instance#console) to your Striim instance.
2. Go to the `/opt/striim` directory:

   ```
   cd /opt/striim
   ```

3. List all arguments:

   ```
   bin/schemaConversionUtility.sh --help
   ```

4. Run the schema conversion utility and include the flags that are appropriate for your use case:

   ```
   bin/schemaConversionUtility.sh \
   -s=oracle \
   ```

```
-d=SOURCE_DATABASE_CONNECTION_URL \
-u=SOURCE_DATABASE_USERNAME \
-p=SOURCE_DATABASE_PASSWORD \
-b=SOURCE_TABLES_TO_CONVERT \
-t=postgres \
-f=false
```

Replace the following:

- ***SOURCE_DATABASE_CONNECTION_URL***: connection URL for Oracle database—for example, `"jdbc:oracle:thin:@12.123.123.12:1521/APPSPDB.WORLD"`
- ***SOURCE_DATABASE_USERNAME***: Oracle username to use to connect to the Oracle database
- ***SOURCE_DATABASE_PASSWORD***: Oracle password to use to connect to the Oracle database
- ***SOURCE_TABLES_TO_CONVERT***: table names from the source database that are used to convert schemas

Make sure that you use the `-f=false` argument. This argument exports the foreign key declarations into a separate file.

The output folder might contain some or all of the following files. For more details on these files, see Striim's schema conversion utility documentation (https://www.striim.com/docs/en/using-the-schema-conversion-utility.html).

Output filename	Description
`converted_tables.sql`	Contains all the converted tables that don't require any coercion
`converted_tables_with_striim_intelligence.sql`	Contains all the converted tables that have been converted with some coercion
`conversion_failed_tables.sql`	Contains tables where conversion was attempted, but a mapping was not obtained
`converted_foreignkey.sql`	Contains all foreign key constraint declarations

Output filename	Description
`conversion_failed_foreignkey.sql`	Contains all failed foreign key conversions
`conversion_report.txt`	Contains a verbose report of the schema conversion

In this tutorial, you use the `converted_tables.sql` file to create equivalent tables in the Cloud SQL for PostgreSQL database without any foreign key constraints. After the initial replication, you use the `converted_foreignkey.sql` file to apply the foreign key constraints.

Convert the schema using Ora2Pg

Another option for converting Oracle table schemas to equivalent PostgreSQL schemas is the Ora2Pg utility. You can install this utility on a separate Google Cloud VM.

The Ora2Pg utility converts the Oracle schema and exports the DDL statements that are required to create equivalent tables in the PostgreSQL database. These DDL statements are exported in an output file that is named `output.sql`.

During the schema export, you export and save all foreign key declarations into a separate file by using the following flag in the Ora2Pg configuration file:

```
FILE_PER_FKEYS 1
```

By default, foreign keys are exported into the main output file (`output.sql`). When you enable the `FILE_PER_FKEYS` flag (1), foreign keys are exported into a separate file named `FKEYS_output.sql`.

In this tutorial, you use the `output.sql` file to create equivalent tables in the Cloud SQL for PostgreSQL database without any foreign key constraints. After the initial replication, you use the `FKEY_output.sql` file to apply the foreign key constraints.

Preparing the Cloud SQL for PostgreSQL instance

To enable Striim to write data to a Cloud SQL for PostgreSQL instance, you need to create a Cloud SQL instance. You also need to create the database tables and the schema that Striim writes to:

1. In Cloud Shell, create a Cloud SQL for PostgreSQL instance (/sql/docs/postgres/create-instance#gcloud). We recommend that you configure Cloud SQL to use a private IP address (/sql/docs/postgres/private-ip). Use the `--network` parameter (/sql/docs/postgres/configure-private-ip) to configure this address:

   ```
   INSTANCE_NAME=INSTANCE_NAME

   gcloud beta sql instances create INSTANCE_NAME \
       --database-version=POSTGRES_12 \
       --network=NETWORK \
       --cpu=NUMBER_CPUS \
       --memory=MEMORY_SIZE \
       --region=REGION
   ```

   Replace the following:

   - ***INSTANCE_NAME***: the instance name
   - ***NETWORK***: the name of the VPC network that you use for this instance
   - ***NUMBER_CPUS***: number of vCPUs in the instance
   - ***MEMORY_SIZE***: amount of memory for the instance. For example, 3072MiB or 9GiB. GiB is assumed if you do not specify the unit.
   - ***REGION***: the region where you created the Cloud Storage bucket

2. Create a username and password (/sql/docs/postgres/create-manage-users#creating) on the Cloud SQL instance:

   ```
   CLOUD_SQL_USERNAME=CLOUD_SQL_USERNAME

   gcloud sql users create $CLOUD_SQL_USERNAME \
     --instance=$INSTANCE_NAME \
     --password=CLOUD_SQL_PASSWORD
   ```

Replace the following:

- ***CLOUD_SQL_USERNAME***: a username for your Cloud SQL instance
- ***CLOUD_SQL_PASSWORD***: the password for the Cloud SQL username

This user is granted ownership of the PostgreSQL tables. Striim also uses this user's credentials to connect to the Cloud SQL for PostgreSQL database.

The schema files that are exported during the schema conversion step (#convert-the-schema) might have a DDL statement that grants ownership to a user, as in the following example:

```
CREATE SCHEMA <SCHEMA_NAME>;
ALTER SCHEMA <SCHEMA_NAME> OWNER TO <USER>;
```

You might need to replace **`SCHEMA_NAME`** with **`CLOUD_SQL_SCHEMA`**and **`USER`** with the **`CLOUD_SQL_USERNAME`** created earlier.

3. Create a PostgreSQL database

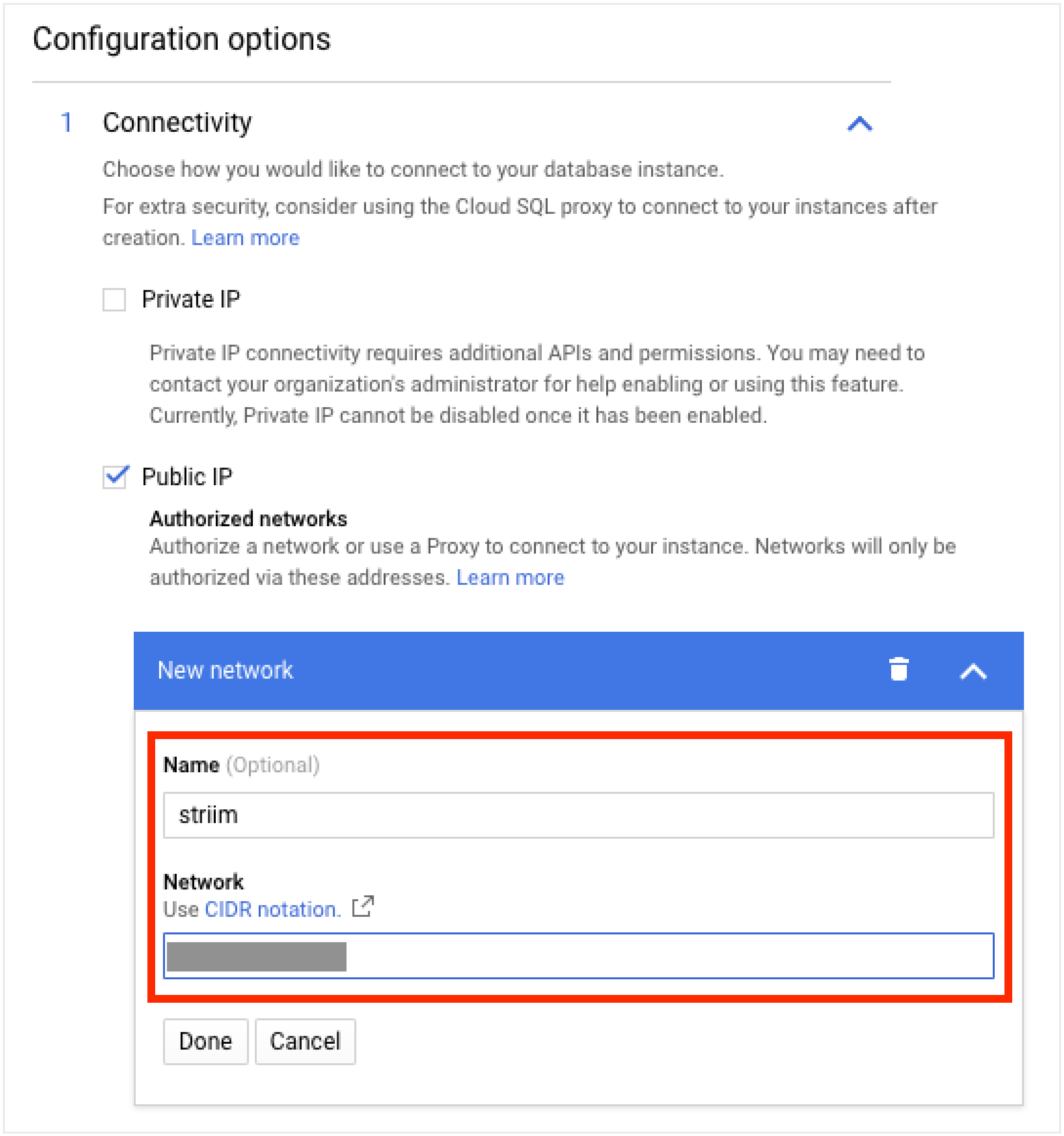
```
CLOUD_SQL_DATABASE_NAME=CLOUD_SQL_DATABASE_NAME

gcloud sql databases create $CLOUD_SQL_DATABASE_NAME \
  --instance=$INSTANCE_NAME
```

Replace the following:

- ***CLOUD_SQL_DATABASE_NAME***: PostgreSQL database name

4. Configure the Cloud SQL for PostgreSQL database to allow access from the Striim instance. Connectivity options depend on whether you configured the Cloud SQL instance to use a public or private IP address.

    a. If you configured a public IP address, add the Striim's instance IP address as an authorized address on the Cloud SQL instance (/sql/docs/mysql/configure-ip#add). The following screenshot shows how to do that from the Google Cloud Console:



Configuration options

1 Connectivity

Choose how you would like to connect to your database instance.

For extra security, consider using the Cloud SQL proxy to connect to your instances after creation. Learn more

☐ Private IP

Private IP connectivity requires additional APIs and permissions. You may need to contact your organization's administrator for help enabling or using this feature. Currently, Private IP cannot be disabled once it has been enabled.

✓ Public IP

Authorized networks
Authorize a network or use a Proxy to connect to your instance. Networks will only be authorized via these addresses. Learn more

New network

Name (Optional)

striim

Network
Use CIDR notation.

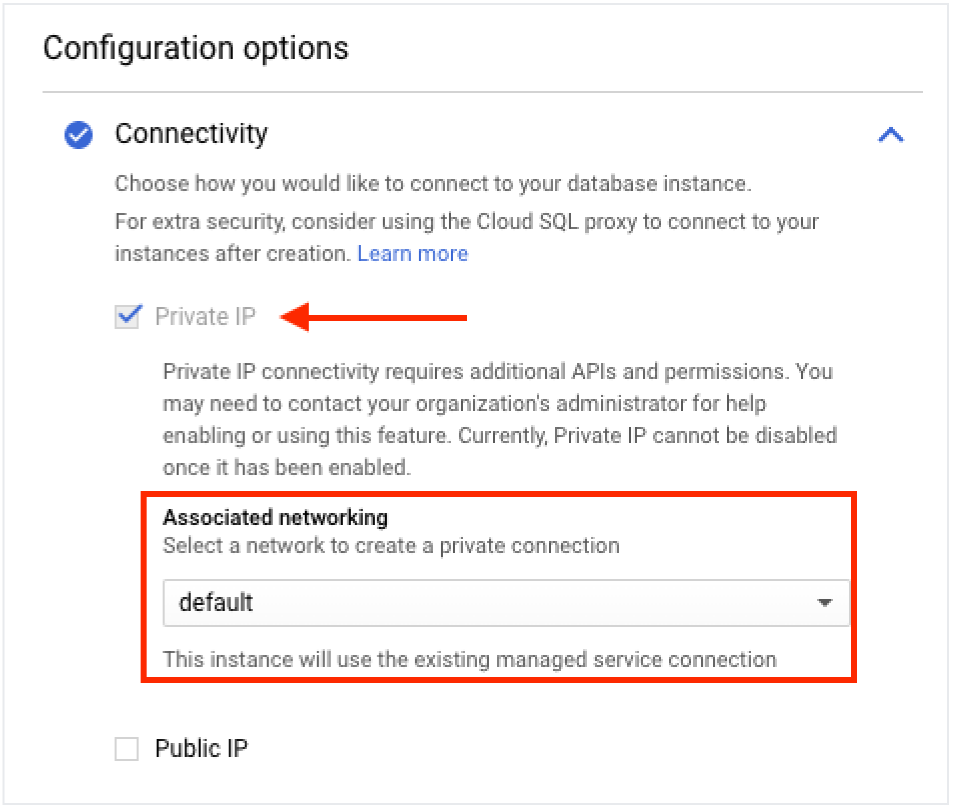
[redacted]

Done Cancel

b. If you configured a private IP address (/sql/docs/postgres/private-ip), the available connectivity options (/sql/docs/postgres/configure-private-ip#connecting_to_an_instance_using_its_private_ip) depend on whether or not the Cloud SQL instance and the Striim instance are on the same VPC network.

   i. If your Striim instance is on the same VPC network as your Cloud SQL instance, the Striim instance can establish connection with the Cloud SQL instance.

   The following screenshot shows that the Cloud SQL instance is associated with the default VPC network. If the Striim instance was also created on the default VPC network, it can privately connect with the Cloud SQL instance.


Configuration options

Connectivity

Choose how you would like to connect to your database instance.

For extra security, consider using the Cloud SQL proxy to connect to your instances after creation. Learn more

[✓] Private IP

Private IP connectivity requires additional APIs and permissions. You may need to contact your organization's administrator for help enabling or using this feature. Currently, Private IP cannot be disabled once it has been enabled.

Associated networking
Select a network to create a private connection

default

This instance will use the existing managed service connection

[ ] Public IP

ii. If your Striim instance is on a different VPC network than your Cloud SQL instance, configure private service access (/sql/docs/postgres/configure-private-services-access) on your Striim instance's VPC network.

5. Create table schemas without foreign key constraints in the Cloud SQL for PostgreSQL database.

    a. To export `output.sql` during the schema conversion step (#convert-the-schema), use the `output.sql` file to create the schemas.

    b. To export `converted_tables.sql` during the schema conversion step, use the `converted_tables.sql` file to create the schemas.

    You can run either script by using any PostgreSQL client with connectivity to the Cloud SQL for PostgreSQL instance. However, we recommend using the PostgreSQL client you installed earlier on the Striim instance.

    c. Open an SSH session with your Striim instance (/compute/docs/instances/connecting-to-instance#console).

    d. Create the schemas:

```
psql -h HOSTNAME -p CLOUD_SQL_PORT -d CLOUD_SQL_DATABASE_NAME -U
```

Replace the following:

- ***HOSTNAME***: IP address of the Cloud SQL instance
- ***CLOUD_SQL_PORT***: port of the Cloud SQL instance to connect to—by default this port is `5432`
- ***PATH_TO_MAIN_SQL_FILE***: path to the main script on the Striim instance

For example:

```
psql -h 12.123.123.123 -d testdb -U hr -p 5432 -f output.sql
```

6. Verify that the tables were created:

   a. Connect to the Cloud SQL for PostgreSQL database:

   ```
   psql -h HOSTNAME -p 5432 -d CLOUD_SQL_DATABASE_NAME -U
   CLOUD_SQL_USERNAME
   ```

   b. List the tables in this database:

   ```
   \dt
   ```

   The output is a list of tables that the table schema conversion script created in the previous step.

7. Create a checkpointing table (https://www.striim.com/docs/en/creating-the-checkpoint-table.html) on the Cloud SQL for PostgreSQL database:

   a. Connect to the Cloud SQL for PostgreSQL database:

   ```
   psql -h HOSTNAME -p 5432 -d CLOUD_SQL_DATABASE_NAME -U
   CLOUD_SQL_USERNAME
   ```

b. Create the table:

```
CREATE TABLE chkpoint (
id character varying(100) primary key,
sourceposition bytea,
pendingddl numeric(1),
ddl text);
```

Striim needs this table to maintain checkpoints during the continuous replication process.

Loading the Oracle database to Cloud SQL for PostgreSQL database

This section describes the one-time, initial replication of the Oracle database to the Cloud SQL for PostgreSQL database.

Establish a connection to Oracle from Striim

For the initial load, you use the Striim Database Reader adapter (https://www.striim.com/docs/en/database-reader.html) to connect to Oracle from Striim.

1. In the Striim Database Reader adapter, go to **Sources**, and then search for and select **Database** from the list.
2. Set the following properties in the **Database** window:
   - **Name:** identify this component of the migration pipeline.
   - **Adapter:** `DatabaseReader`
   - **Connection URL:** enter a unique string to connect to the Oracle database:

```
jdbc:oracle:thin:@HOSTNAME:ORACLE_PORT:SID
```

OR

```
jdbc:oracle:thin:@HOSTNAME:ORACLE_PORT/PDB_OR_CDB_SERVICE_NAME
```

Replace the following:

- ***ORACLE_PORT***: Oracle database port (`1521` by default)
- ***SID***: Oracle database SID
- ***PDB_OR_CDB_SERVICE_NAME***: Oracle PDB or CDB service name: If your tables are in a PDB, use `PDB_SERVICE_NAME`; if they are in a CDB, use `CDB_SERVICE_NAME`.

You can find the port and service name in the `tnsnames.ora` file located at `$ORACLE_HOME/network/admin/tnsnames.ora` on the Oracle instance.

- **Username** and **password:** Use the Oracle user (`c##striim`user) you created in the prerequisite steps. Striim uses this username and password to connect to your Oracle database and read the tables.
- **Tables:** For Oracle, the Database Reader also needs a list of table names to replicate. This property is specified in the Tables field under **Show optional properties**. The format for this property is as follows:

```
ORACLE_SCHEMA.ORACLE_TABLE_NAME
```

Replace the following:

- ***ORACLE_SCHEMA***: Oracle schema name
- ***ORACLE_TABLE_NAME***: Oracle table names in that schema

You might also specify multiple tables and materialized views as a list separated by semicolons, or with the following wildcards:

`%`: any series of characters

`_`: any single character

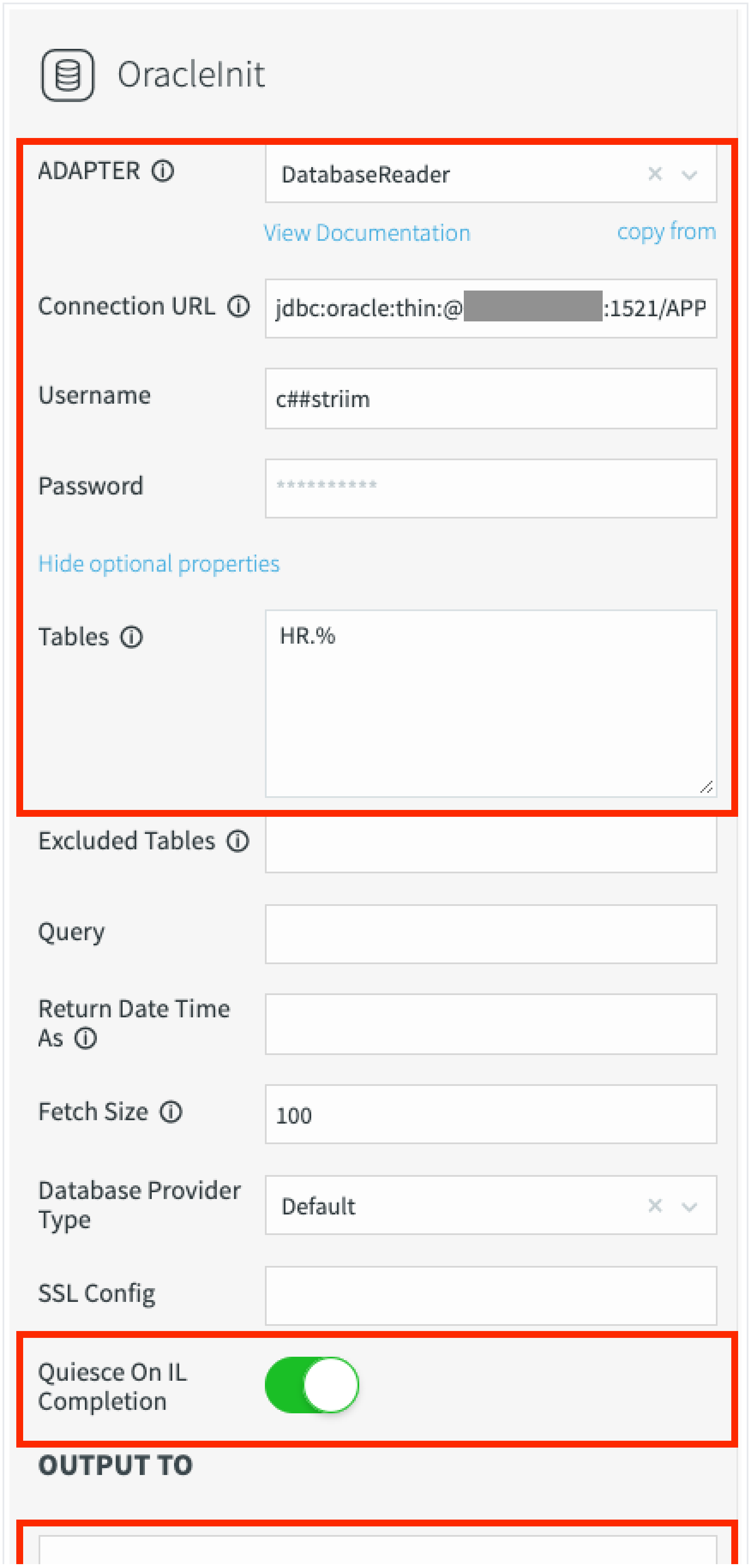
For example, `HR.%` reads all tables in the HR schema. At least one table must match the wildcard. Otherwise, the Database Reader fails with the following error:

```
Could not find tables specified in the database
```

★ **Note:** For initial development, we recommend testing with only one table.

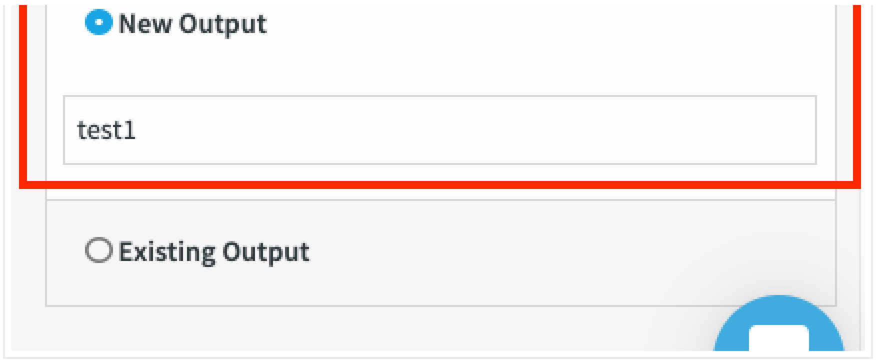
- **Quiesce On IL Completion:** Toggle this field to green by sliding it to the right to pause the pipeline when the initial load is complete.
- **Output To:** Name the output of this adapter. Use a case-sensitive string without special characters or spaces.

3. Click **Save**. The adapter properties display:



OracleInit

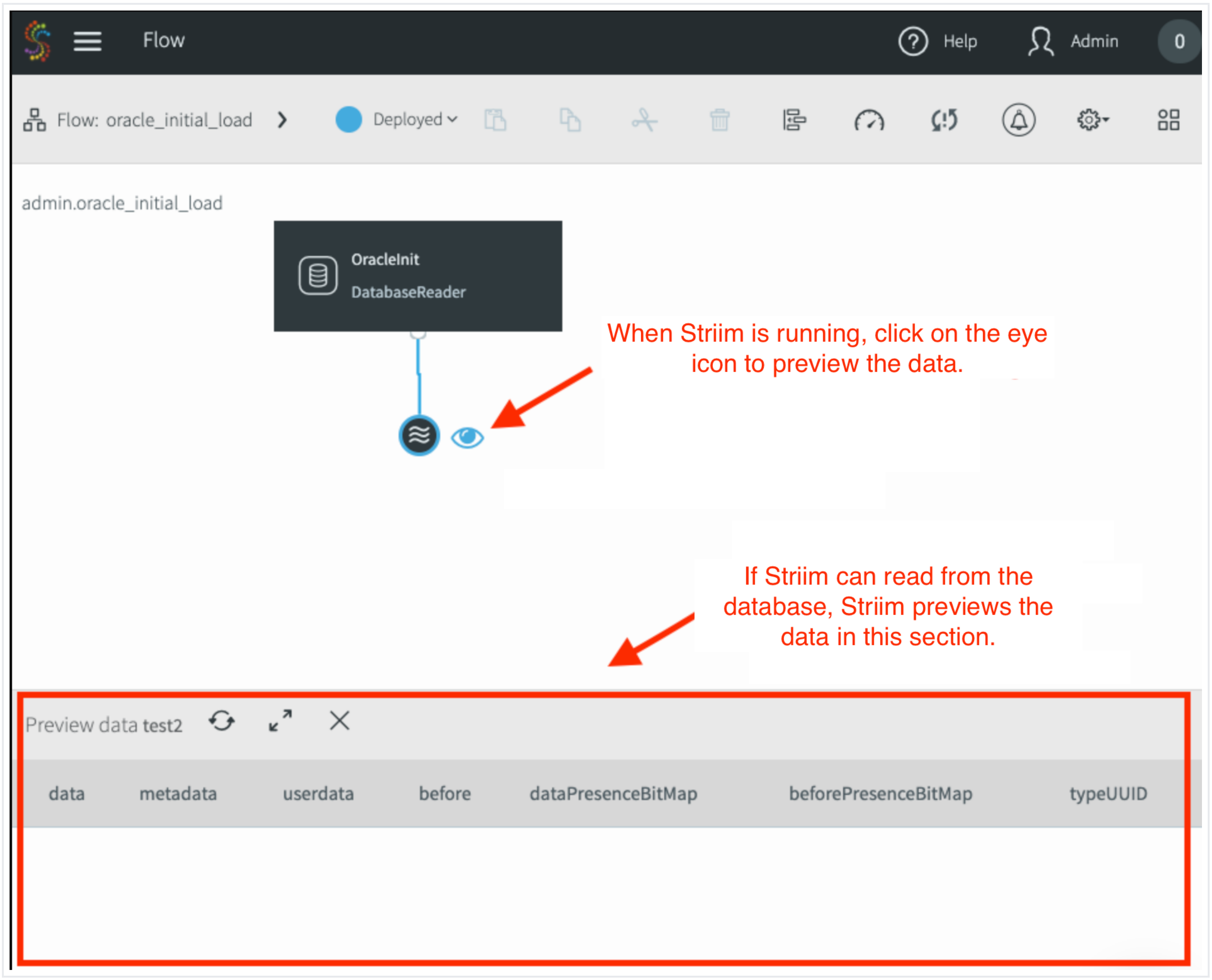
ADAPTER	DatabaseReader
Connection URL	jdbc:oracle:thin:@[redacted]:1521/APP
Username	c##striim
Password	***********
Tables	HR.%
Excluded Tables	
Query	
Return Date Time As	
Fetch Size	100
Database Provider Type	Default
SSL Config	
Quiesce On IL Completion	●



Test the connection

Now that you have connected to Oracle from Striim, test the connection.

1. Click the **Created** drop-down list to test Striim connectivity to the Oracle database.
2. Click **Deploy App**.
3. Select the output of this adapter, and then click **Preview** to display the data in real time as Striim reads it from the source.


Flow
Help
Admin
0
Flow: oracle_initial_load
Deployed
admin.oracle_initial_load
OracleInit
DatabaseReader
When Striim is running, click on the eye icon to preview the data.
If Striim can read from the database, Striim previews the data in this section.
Preview data test2
data
metadata
userdata
before
dataPresenceBitMap
beforePresenceBitMap
typeUUID

4. Click the **Deployed** drop-down list and then click **Start App**.
5. (Optional) Click the **Deployed** drop-down list and then click **Undeploy App** to fix any errors that occur.
6. (Optional) Click **Resume App** after all errors are fixed to restart the app.
7. Click the **default** deployment group.
8. Verify that the **Validate table mappings** option is toggled on, and then click **Deploy**.

   The preview data pane and the pipeline status change to **Quiesced**.

At this point in the tutorial, you have successfully verified that Striim is able to establish a connection to your Oracle database and to read the data within it.

Add a Cloud SQL for PostgreSQL database as a target

For this migration, you write data to the Cloud SQL for PostgreSQL instance. Striim provides a generic database writer adapter, called Database Writer (https://www.striim.com/docs/en/database-writer.html), that you can use for the migration.

1. In Striim Flow Designer, go to **Targets**. Search for and select **Cloud SQL Postgres** from the list.
2. Drag **Database Writer** to the pipeline.
3. Set the following properties:
   - **Adapter:** `DatabaseWriter`
   - **Connection URL:** Enter a unique string to establish a connection to the Cloud SQL instance:

     ```
     jdbc:posgresql://CLOUD_SQL_IP_ADDRESS:CLOUD_SQL_PORT/CLOUD_SQL_DAT
     ```

     Replace the following:

     - ***CLOUD_SQL_IP_ADDRESS***: IP address of the Cloud SQL instance

     For example:

     ```
     jdbc:postgresql://12.123.12.12:5432/postgres?stringtype=unspecified
     ```

   - **Username** and **password:** Enter the Cloud SQL username and password that you created earlier.
   - **Tables:** Create a mapping from your Oracle database table names to to Cloud SQL table names. Specify which Oracle database table is written to which Cloud SQL table. This mapping uses the following format:

     ```
     ORACLE_SCHEMA.ORACLE_TABLE_NAME,CLOUD_SQL_SCHEMA.CLOUD_SQL_TABLE
     ```

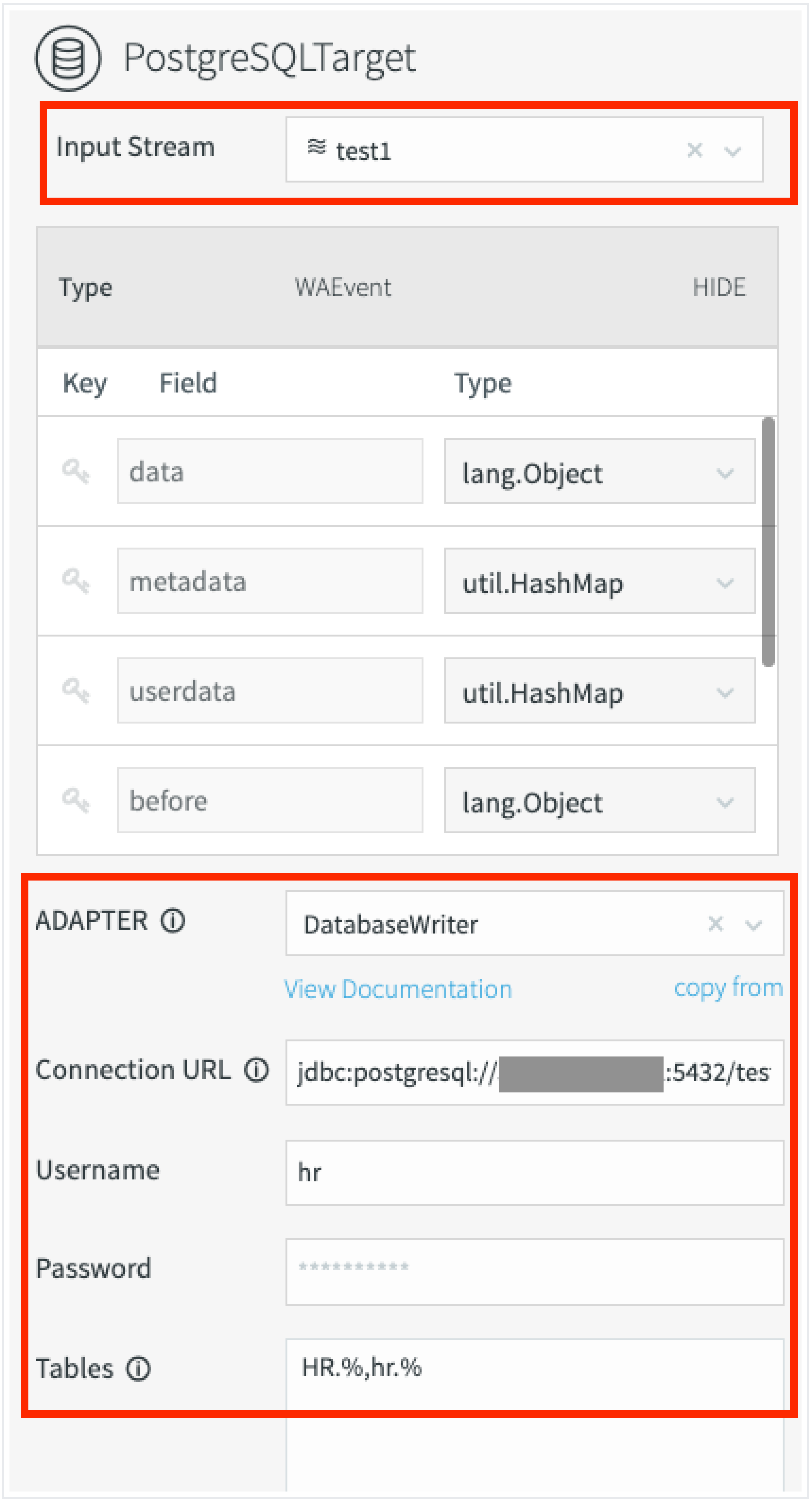
     Replace the following:

- ***CLOUD_SQL_SCHEMA***: PostgreSQL schema name
- ***CLOUD_SQL_TABLE_NAME***: PostgreSQL table name

To map multiple tables, you can use the wildcard symbol (%) in the **Tables** field—for example:

```
HR.%,hr.%
```

The required fields for Database Writer are marked in the following screenshot:



PostgreSQLTarget

Input Stream: ≋ test1

Type	WAEvent	HIDE

Key	Field	Type
	data	lang.Object
	metadata	util.HashMap
	userdata	util.HashMap
	before	lang.Object

ADAPTER: DatabaseWriter

Connection URL: jdbc:postgresql://[redacted]:5432/tes

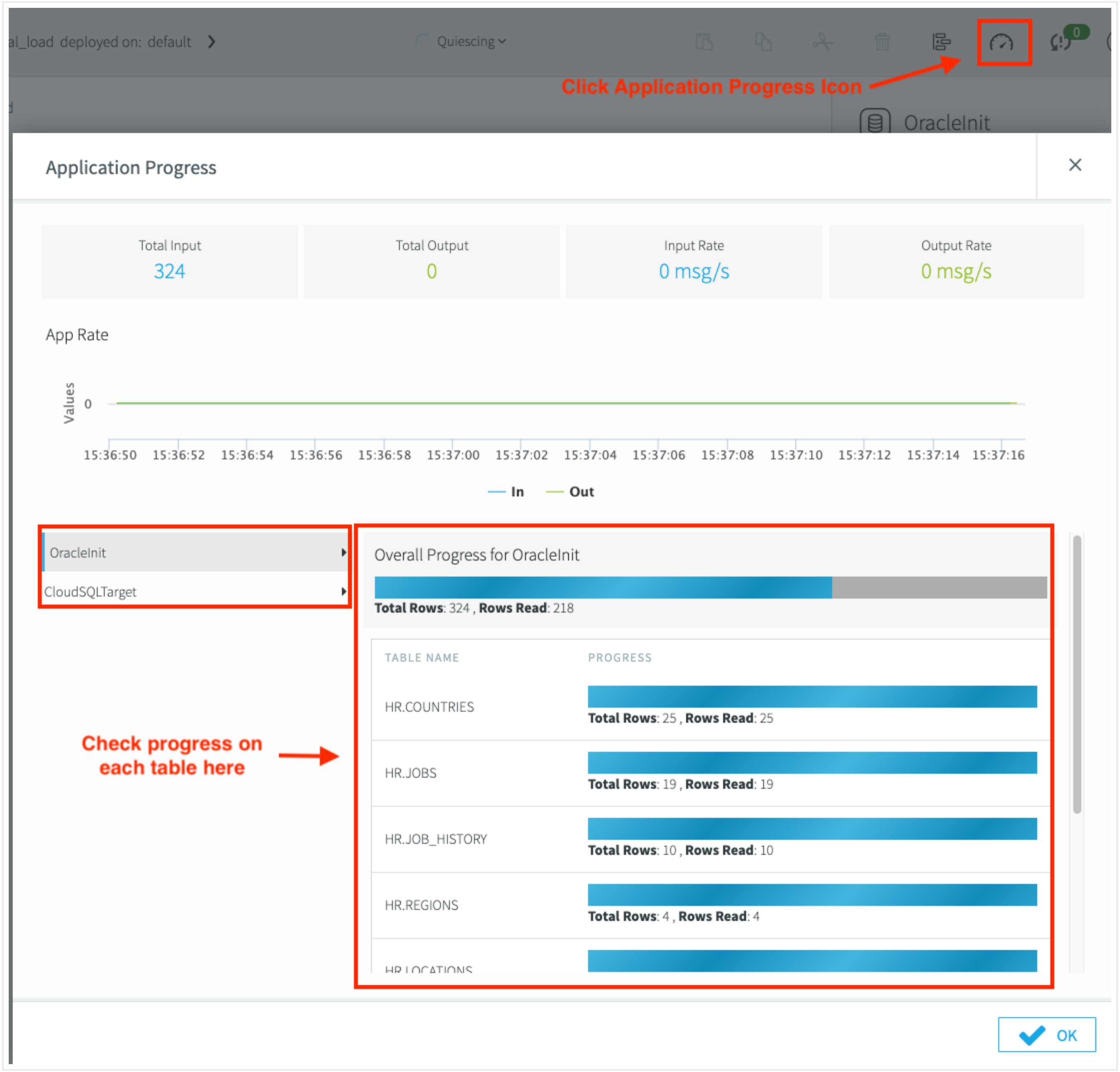
Username: hr

Password: **********

Tables: HR.%,hr.%

Deploy the migration pipeline

After the migration pipeline is ready, deploy it from the Striim Flow Designer and start the application. You can also preview the data being replicated in real time. Use Monitor reports (https://www.striim.com/docs/en/using-monitor-reports.html) to track the progress of the replication. To track the progress, select the **Application Progress** icon.


al_load deployed on: default
Quiescing
Click Application Progress Icon
OracleInit
Application Progress
App Rate
Values
In
Out
Check progress on each table here
OK

Total Input	Total Output	Input Rate	Output Rate
324	0	0 msg/s	0 msg/s

OracleInit
CloudSQLTarget

Overall Progress for OracleInit
Total Rows: 324 , **Rows Read**: 218

TABLE NAME	PROGRESS
HR.COUNTRIES	**Total Rows**: 25 , **Rows Read**: 25
HR.JOBS	**Total Rows**: 19 , **Rows Read**: 19
HR.JOB_HISTORY	**Total Rows**: 10 , **Rows Read**: 10
HR.REGIONS	**Total Rows**: 4 , **Rows Read**: 4
HR.LOCATIONS	

1. In the Striim Flow Designer, deploy the migration pipeline. Click the **Created** drop-down list, and then click **Deploy App**. After the initial load is complete, the pipeline status changes to `Quiesced`.

2. Click **Undeploy the app** to roll back the deployment.

3. Verify that the data load succeeded by checking the row count:

   ```
   SELECT COUNT(*) FROM <TARGET CLOUD SQL TABLE>;
   ```

   You should see a non-zero output. If you don't, the data load failed.

The initial data load from the Oracle database to Cloud SQL for PostgreSQL is atomic. Either the entire data load succeeds or the entire data load fails. If the initial load fails, you must load the data again (#deploy-the-pipeline).

Enabling foreign key constraints on the Cloud SQL for PostgreS tables

After the initial load is complete, enable the foreign key constraints on the target tables. Use the file with foreign key declarations (`FKEY_output.sql` or `converted_foreignkey.sql`) that you created during the schema conversion (#convert-the-schema).

1. In Striim, open an SSH session (/compute/docs/instances/connecting-to-instance#console).
2. Create foreign key constraints on the tables:

   ```
   psql -h HOSTNAME -d CLOUD_SQL_DATABASE_NAME -U CLOUD_SQL_USERNAME -p
   CLOUD_SQL_PORT -f PATH_TO_FOREIGN_KEY_FILE
   ```

   Replace the following:

   - ***CLOUD_SQL_USERNAME***: Cloud SQL for PostgreSQL username
   - ***PATH_TO_FOREIGN_KEY_FILE***: : path to the script with foreign key constraints on the Striim instance

     For example:

     `psql -h 12.123.123.123 -d testdb -U hr -p 5432 -f output.sql`

Continuously replicating the Oracle database to Cloud SQL for PostgreSQL

After you complete your initial data load, create a separate pipeline to replicate changes to the Oracle database. As long as it remains running, this pipeline also keeps the source database in sync with the target database.

Establish a connection to Oracle from Striim

For continuous replication, you use the Striim Oracle Reader adapter (https://www.striim.com/docs/en/oracle-reader-properties.html) to connect from Striim to the Oracle database. This Striim adapter can read CDC data from Oracle.

1. In the Striim Oracle Reader adapter, navigate to **Sources**.
2. Search for **Oracle**, and select **Oracle CDC** from the list that populates.
3. Set the following properties:
    a. **Connection URL:**

    ```
    HOSTNAME:ORACLE_PORT/SID
    ```

    OR

    ```
    HOSTNAME:ORACLE_PORT/CDB_SERVICE_NAME
    ```

    Replace the following:

    - ***CDB_SERVICE_NAME***: Oracle's CDB service name

    The connection URL is a unique string that's used to connect to the Oracle database. Unlike the Database Reader adapter used for the initial load, you use the CDB service name, regardless of whether your database tables are in a PDB or CDB.

    For example: `12.123.123.12:1521/ORCLCDB.WORLD`.

b. **Username/password:** Use the Oracle username (`c##striim`user) you created in the prerequisite steps.

   This Oracle user must have the privileges to read your tables.

c. **Tables:** You also need a list of table names to replicate. The name is specified in the following format, based on whether the tables are in a CDB or PDB.

   For the CDB table:

```
ORACLE_SCHEMA.ORACLE_TABLE_NAME
```

   For the PDB table:

```
PDB_NAME.ORACLE_SCHEMA.ORACLE_TABLE_NAME
```

   Replace the following:

   - ***PDB_NAME***: Oracle PDB name

   This command replicates your CDB or PDB tables. You can find your `PDB_NAME` in the file `tnsnames.ora` located at `$ORACLE_HOME/network/admin/tnsnames.ora` on the Oracle instance.

   Remember, `PDB_NAME` and `PDB_SERVICE_NAME` are different. You used the `PDB_SERVICE_NAME` earlier in the section. View the `tnsnames.ora` file to get the PDB name:

```
sudo su - oracle // Login as oracle user
cat  ORACLE_HOME/network/admin/tnsnames.ora
```

   The following is an example of the `PDB_NAME` (`APPSPDB`) in the `tnsnames.ora` file:

```
  APPSPDB =
    (DESCRIPTION =
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCP) (HOST = orainst) (PORT = 1521))
      )
      (CONNECT_DATA =
        (SERVICE NAME = APPSPDB.WORLD)
      )
    )
```

To specify multiple tables and materialized views as a list, separate the the table names or view names by semicolons or wildcards. At least one table must match the wildcard; otherwise the Oracle Reader fails with a `Could not find tables specified in the database` error.

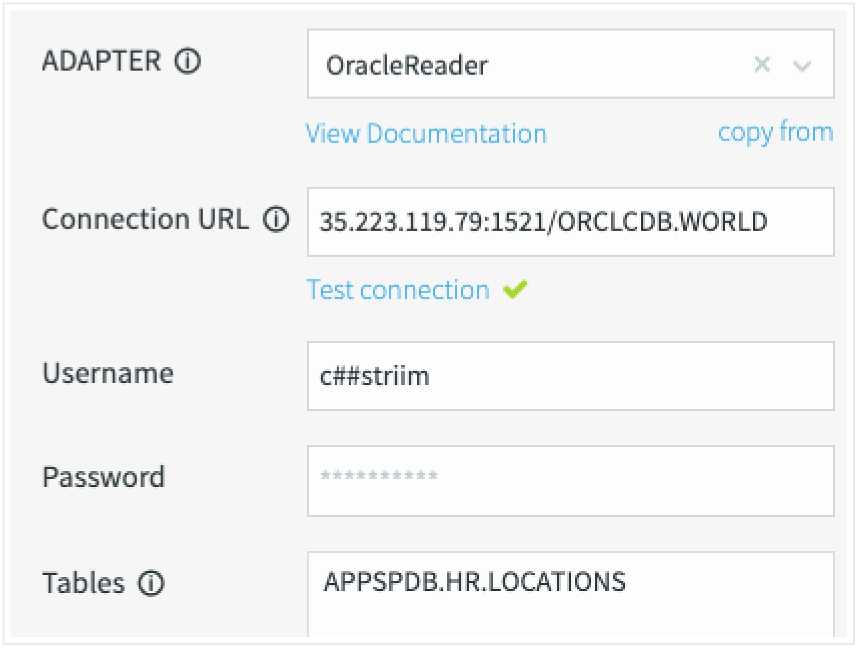
★ **Note:** We recommend that you initially test the connectivity with only one table.

d. **Start SCN:** For the continuous pipeline, you need to provide the Oracle database SCN. Striim needs it to start replicating all transactions. Enter the SCN value you generated earlier.

   i. **Support PDB and CDB:** You can use a CDB or a PDB; expand **Show optional properties** and toggle the switch to the right.

   ii. **Quiesce marker table:** Use the table name you created earlier.

   The following screenshot provides an overview of the required fields for the Oracle Reader adapter:

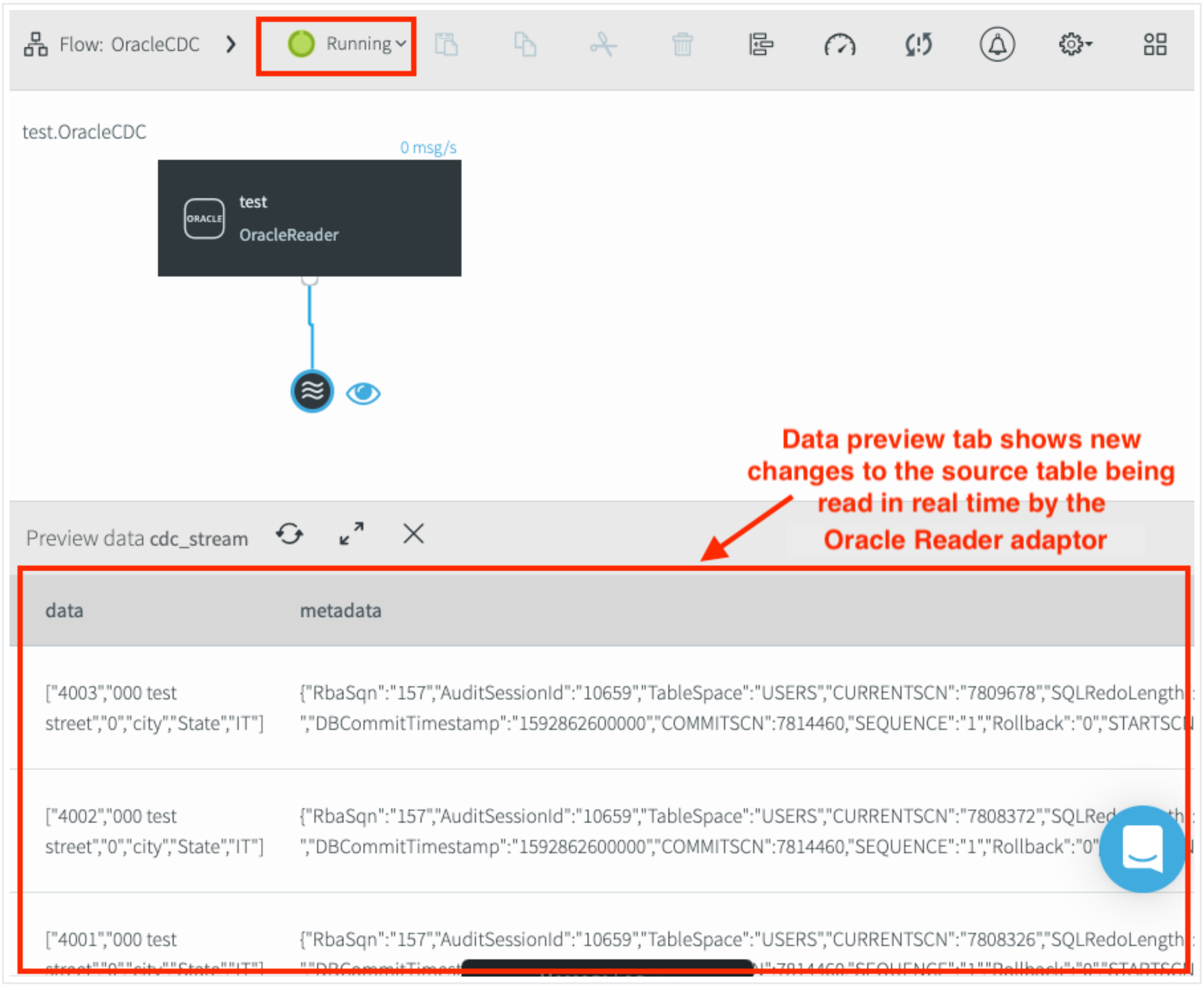


ADAPTER	OracleReader
Connection URL	35.223.119.79:1521/ORCLCDB.WORLD
Username	c##striim
Password	**********
Tables	APPSPDB.HR.LOCATIONS

4. **Test connection:** Click **Test connection**. The connection URL, username, and password are required to test database connectivity. If Striim is successfully able to establish a connection, a green check mark appears.
5. Test Striim's ability to read the Oracle database tables:
    a. In the Oracle Reader adapter, select **Deploy app**.
    b. Select the **default** deployment group.
    c. Click **Deploy**.
6. Click the wave (**Output**) icon for this adapter. The eye (**Preview**) icon that appears is used to preview the data in real time as Striim reads it from the source.
7. Click **Start App** in the **Deployed** dropdown.

   If any errors occur, select **Undeploy App** from the same dropdown and fix the errors. After you fix the errors, click **Resume App** to restart the application.

   When the pipeline starts, the pipeline status updates to **Running**. Any new changes to the source table display in the preview window. Because the Oracle Reader adapter uses CDC, the only table changes that appear in the preview data pane are those that occur after the application was launched.


Flow: OracleCDC
Running
test.OracleCDC
0 msg/s
test
OracleReader
Data preview tab shows new changes to the source table being read in real time by the Oracle Reader adaptor
Preview data cdc_stream

data	metadata
["4003","000 test street","0","city","State","IT"]	{"RbaSqn":"157","AuditSessionId":"10659","TableSpace":"USERS","CURRENTSCN":"7809678","SQLRedoLength ","DBCommitTimestamp":"1592862600000","COMMITSCN":7814460,"SEQUENCE":"1","Rollback":"0","STARTSC
["4002","000 test street","0","city","State","IT"]	{"RbaSqn":"157","AuditSessionId":"10659","TableSpace":"USERS","CURRENTSCN":"7808372","SQLRe ","DBCommitTimestamp":"1592862600000","COMMITSCN":7814460,"SEQUENCE":"1","Rollback":"0"
["4001","000 test	{"RbaSqn":"157","AuditSessionId":"10659","TableSpace":"USERS","CURRENTSCN":"7808326","SQLRedoLength

Verify the ability to read CDC data from Oracle

To test if the adapter is able to read new changes, follow these instructions:

1. Use SQL statements to insert new transactions in the Oracle source tables.
2. Verify that the new transactions appear in the **Preview data** tab of Oracle Reader adapter.
3. Stop the application and click **Undeploy**. Now you are ready to move to the next step.

Up to this point you haven't added a target adapter to the pipeline. No data is copied unless you add a target adapter. In the next section, you add a target adapter.

Add a Cloud SQL for PostgreSQL database as a target

In order to write data to the Cloud SQL for PostgreSQL database, you need to add a Database Writer adapter to your pipeline. For the continuous replication pipeline, you use the same adapter that you used in the initial load pipeline.

1. In Striim Flow Designer, go to **Targets**, and search for and select **Cloud SQL Postgres** from the list.
2. Drag **Database Writer** to the pipeline.
3. Set the following properties:
   - **Adapter:** `DatabaseWriter`.
   - **Connection URL:** Enter the connection URL you entered to establish a connection to the Cloud SQL instance:

```
jdbc:posgresql://CLOUD_SQL_IP_ADDRESS:CLOUD_SQL_PORT/CLOUD_SQL_DATABASE
```

For example:

```
jdbc:postgresql://12.123.12.12:5432/postgres?stringtype=unspecified
```

   - **Username** and **password:** Enter the Cloud SQL username and password that you created earlier.
   - **Tables:** Create a mapping from your Oracle database table names to Cloud SQL table names. Specify which Oracle database table is written to which Cloud SQL table. This mapping uses the following format:

```
ORACLE_SCHEMA.ORACLE_TABLE_NAME,CLOUD_SQL_SCHEMA.CLOUD_SQL_TABLE_NAMI
```

To map multiple tables, you can use the wildcard symbol (%) in the **Tables** field. For example:

```
HR.%,hr.%
```

In addition to those properties, you also need to set the following properties for the continuous replication pipeline:

4. Click **Show optional properties**.
5. Select the following value for the **Ignorable Exception Code** field:

   ```
   23505,NO_OP_UPDATE,NO_OP_DELETE
   ```

   Since you're starting the CDC pipeline from a historical point, there might be duplicates. Striim deduplicates on your target by using the previous ignorable exception code properties.Details on the exception codes can be found in the following table:

Exception code	Details
`23505`	Duplicate primary key value violates unique constraint
`NO_OP_UPDATE`	Couldn't update a row in the target (typically because there was no corresponding primary key)
`NO_OP_DELETE`	Couldn't delete a row in the target (typically because there was no corresponding primary key)

6. Enter `chkpoint` in the **Check Point Table** field. Striim uses this table to store metadata associated with checkpointing the continuous replication pipeline.

Enabling recovery and encryption

Before you deploy the CDC pipeline, we strongly recommend that you enable recovery. If the Striim application or the VM goes down, enabling recovery helps ensure that Striim can continue processing. This step also helps ensure exactly- once processing semantics. These semantics track the last-known-good read checkpoint on the source database, and the last-known-good write checkpoint on the target database. If an application or VM fails, Striim coordinates the two checkpoints to help ensure that no data was lost or duplicated. Recovery doesn't apply to initial load applications.

Enable recovery

1. In Striim Flow Designer, select **App Settings**.
2. Click **Recovery Interval**.
3. Type 5 and select **Second** from the drop-down list.
4. Click **enable encryption**. Striim encrypts all streams that move data between Striim servers, or from a forwarding agent, to a Striim server.

Enable encryption

- In Striim Flow Designer, select **App Settings**, and then under **Encryption**, select the checkbox.

See the Striim website (https://www.striim.com/docs/en/recovering-applications.html) to learn more about Striim's recovery methods.

Enable logging exceptions

Before you deploy the continuous replication pipeline, we recommend that you enable the exception store (https://www.striim.com/docs/en/create-exceptionstore.html) in Striim. As part of the CDC application, there might be duplicates written by the initial load application. The Striim application ignores those errors, writes them to a store (for you to review and process) and continues processing.

1. In Striim Flow Designer, select the **Exceptions** icon. The icon shows an exclamation point between two curved arrows.
2. Click **Turn on**.

Deploy the pipeline

After the pipeline is ready, you can deploy it and start the application. You can also preview the data as it's replicated in real time and view monitor reports (https://www.striim.com/docs/en/using-monitor-reports.html). When the pipeline successfully starts the continuous replication, the pipeline status changes to **Running**.

1. In the Oracle Reader adapter, select **Deploy app**.
2. Select the **default** deployment group.
3. Click **Deploy**.

You can keep the pipeline running for as long as you want to keep the Oracle tables in sync with the Cloud SQL tables.

You have finished the tutorial. If you are interested in learning about other Oracle CDC sources, the following section discusses them.

Alternate Oracle CDC sources

In addition to LogMiner, Striim's adapter can read Oracle databases from either XStream (https://docs.oracle.com/en/database/oracle/oracle-database/19/xstrm/introduction-to-xstream.htm) or Oracle Golden Gate trail files (https://www.oracle.com/integration/goldengate/).

To read from XStream, use Striim's Oracle Reader adapter (https://www.striim.com/docs/en/oracle-reader-properties.html). XStream may have better performance, but it requires a Golden Gate license and is only supported for Oracle Database 11.2.0.4 (https://docs.oracle.com/cd/E11882_01/server.112/e41360/chapter1_11204.htm#NEWFT379).

To read Golden Gate trail files, use Striim's GG Trail Reader adapter (https://www.striim.com/docs/en/gg-trail-reader-properties.html).

The following table describes differences between LogMiner and XStream:

Oracle database CDC features	**Supported by LogMiner?**	**Supported by XStream Out?**
Reading data definition language (DDL), `ROLLBACK`, and uncommitted transactions	Yes	No
Using the `DATA()` and `BEFORE()` functions	Yes	No

Oracle database CDC features	Supported by LogMiner?	Supported by XStream Out?
Using `QUIESCE` (see Console commands (https://www.striim.com/docs/en/console-commands.html))	Yes	No
Receiving CDC events	Receives events in batches as defined by Oracle Reader's `FetchSize` property	Continuous reception of change data events
Reading from tables containing unsupported types	Won't read the table	Reads the columns of supported types

Cleaning up

To avoid incurring charges to your Google Cloud account for the resources used in this tutorial, either delete the project that contains the resources, or keep the project and delete the individual resources.

Caution: Deleting a project has the following effects:

- **Everything in the project is deleted.** If you used an existing project for this tutorial, when you delete it, you also delete any other work you've done in the project.
- **Custom project IDs are lost.** When you created this project, you might have created a custom project ID that you want to use in the future. To preserve the URLs that use the project ID, such as an `appspot.com` URL, delete selected resources inside the project instead of deleting the whole project.

If you plan to explore multiple tutorials and quickstarts, reusing projects can help you avoid exceeding project quota limits.

1. In the Cloud Console, go to the **Manage resources** page.

   Go to Manage resources (https://console.cloud.google.com/iam-admin/projects)

2. In the project list, select the project that you want to delete, and then click **Delete**.

3. In the dialog, type the project ID, and then click **Shut down** to delete the project.

What's next

- Review Architecting database migration and replication using Striim (/solutions/architecting-database-migration-replication-striim#complex_deployment_architecture).
- Check out the Striim documentation: Oracle to Google Cloud PostgreSQL Migration Guide (https://www.striim.com/docs/smsgc/en/smsgc-how-to-guides/oracle-to-google-cloud-postgresql-migration-guide.html)
- Watch the video mentioned in How to migrate an Oracle database to Cloud SQL for PostgreSQL with streaming data integration (https://www.striim.com/migrate-oracle-database-to-google-cloud-sql-postgresql-streaming-data-integration)

  .
- Explore reference architectures, diagrams, tutorials, and best practices about Google Cloud. Take a look at our Cloud Architecture Center (/architecture).

Oracle, Java, and MySQL are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Last updated 2021-07-02 UTC.