Build and Execute MySQL, PostgreSQL, and SQLServer to Data Catalog Connectors

experiment Lab schedule 1 hour 15 minutes universal_currency_alt No cost

show_chart Introductory

GSP814

Overview

Dataplex is an intelligent data fabric that enables organizations to centrally discover, manage, monitor, and govern their data across data lakes, data warehouses, and data marts to power analytics at scale.

Data Catalog is a fully managed, scalable metadata management service within Dataplex. It offers a simple and easy-to-use search interface for data discovery, a flexible and powerful cataloging system for capturing both technical and business metadata, and a strong security and compliance foundation with Cloud Data Loss Prevention (DLP) and Cloud Identity and Access Management (IAM) integrations.

Using Data Catalog

Using Data Catalog within Dataplex, you can search for assets to which you have access, and you can tag data assets to support discovery and access control. Tags allow you to attach custom metadata fields to specific data assets for easy identification and retrieval (such as tagging certain assets as containing protected or sensitive data); you can also create reusable tag templates to rapidly assign the same tags to different data assets.

Objectives

In this lab, you will learn how to:

- Enable the Data Catalog API.
- Configure Dataplex connectors for SQL Server, PostgreSQL, and MySQL.
- Search for SQL Server, PostgreSQL, and MySQL entries in Data Catalog within Dataplex.

Prerequisites

Note: Before starting this lab, log out of your personal or corporate gmail account, or run this lab in Incognito. This prevents sign-in confusion while the lab is running.

Setup and requirements

Before you click the Start Lab button

Read these instructions. Labs are timed and you cannot pause them. The timer, which starts when you click **Start Lab**, shows how long Google Cloud resources will be made available to you.

This hands-on lab lets you do the lab activities yourself in a real cloud environment, not in a simulation or demo environment. It does so by giving you new, temporary credentials that you use to sign in and access Google Cloud for the duration of the lab.

To complete this lab, you need:

- Access to a standard internet browser (Chrome browser recommended).

Note: Use an Incognito or private browser window to run this lab. This prevents any conflicts between your personal account and the Student account, which may cause extra charges incurred to your personal account.

- Time to complete the lab---remember, once you start, you cannot pause a lab.

Note: If you already have your own personal Google Cloud account or project, do not use it for this lab to avoid extra charges to your account.

How to start your lab and sign in to the Google Cloud console

1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is the **Lab Details** panel with the following:

    - The **Open Google Cloud console** button

    - Time remaining

    - The temporary credentials that you must use for this lab

    - Other information, if needed, to step through this lab

2. Click **Open Google Cloud console** (or right-click and select **Open Link in Incognito Window** if you are running the Chrome browser).

The lab spins up resources, and then opens another tab that shows the **Sign in** page.

Tip: Arrange the tabs in separate windows, side-by-side.

> **Note:** If you see the **Choose an account** dialog, click **Use Another Account**.

3. If necessary, copy the **Username** below and paste it into the **Sign in** dialog.

```
student-01-2ef418121b25@qwiklabs.net
```
content_c

You can also find the **Username** in the **Lab Details** panel.

4. Click **Next**.

5. Copy the **Password** below and paste it into the **Welcome** dialog.

```
IbmZbfiJurzr
```
content_c

You can also find the **Password** in the **Lab Details** panel.

6. Click **Next**.

> **Important:** You must use the credentials the lab provides you. Do not use your Google Cloud account credentials.

> **Note:** Using your own Google Cloud account for this lab may incur extra charges.

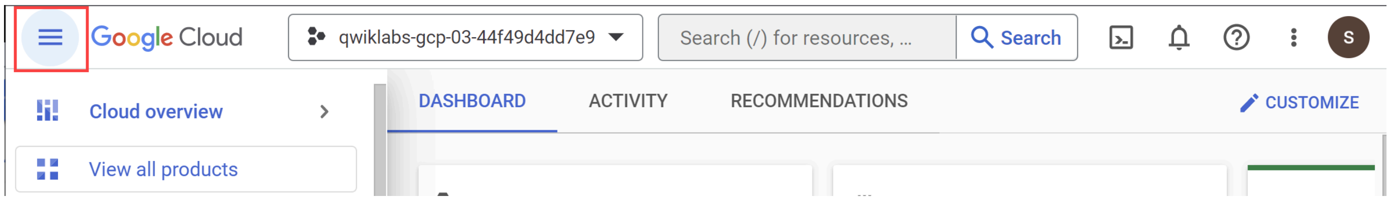
7. Click through the subsequent pages:

   - Accept the terms and conditions.
   - Do not add recovery options or two-factor authentication (because this is a temporary account).

- Do not sign up for free trials.

After a few moments, the Google Cloud console opens in this tab.

Note: To view a menu with a list of Google Cloud products and services, click the **Navigation menu** at the top-left.



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.

When you are connected, you are already authenticated, and the project is set to your **Project_ID**, `qwiklabs-gcp-03-77a600878dcb` . 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-03-77a600878dc
```

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

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

```
gcloud auth list
```

content_c

3. Click **Authorize**.

Output:

```
ACTIVE: *
ACCOUNT: student-01-2ef418121b25@qwiklabs.net

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

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

```
gcloud config list project
```

content_c

Output:

```
[core]
project = qwiklabs-gcp-03-77a600878dcb
```

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

Task 1. Enable the Data Catalog API

1. Open the **Navigation menu** and select **APIs and Services** > **Library**.

2. In the search bar, enter in "Data Catalog" and select the `Google Cloud Data Catalog API`.

3. Then click **Enable**.

Note: If you run into an "Action Failed" error after trying to enable the Data Catalog API, try the following steps: click **close**, refresh your browser tab, then click **Enable** again.

Click **Check my progress** to verify the objective.

Enable the Data Catalog API

Check my progress

Assessment Completed!

Task 2. SQL Server to Dataplex

Start by setting up your environment.

1. Open a new Cloud Shell session by clicking the **Activate Cloud Shell** icon in the top right of the console:

2. Run the following command to set your Project ID as an environment variable:

```
export PROJECT_ID=$(gcloud config get-value project)
```

content_c

Create the SQL Server database

1. In your Cloud Shell session, run the following command to download the scripts to create and populate your SQL Server instance:

```
gsutil cp gs://spls/gsp814/cloudsql-sqlserver-tooling.zip .
unzip cloudsql-sqlserver-tooling.zip
```
content_c

2. Now change your current working directory to the downloaded directory:

```
cd cloudsql-sqlserver-tooling/infrastructure/terraform
```
content_c

3. Run the following command to change the region from `us-central1` to your default assigned region:

```
export REGION=us-west1
sed -i "s/us-central1/$REGION/g" variables.tf
```
content_c

4. Now run the `init-db.sh` script.

```
cd ~/cloudsql-sqlserver-tooling
bash init-db.sh
```
content_c

This will create your SQL Server instance and populate it with a random schema.

Note: If you get an `Error: Failed to load "tfplan" as a plan file`, re-run the `init-db` script.

This will take `around 5 to 10 minutes` to complete. You can move on when you receive the following output:

```
CREATE TABLE factory_warehouse15797.employees53b82dc5 ( school80581 REAL,
reason91250 DATETIME, randomdata32431 BINARY, phone_number52754 REAL,
person66471 REAL, credit_card75527 DATETIME )

COMPLETED
```

Click **Check my progress** to verify the objective.

Create the SQL Server Database

Check my progress

Assessment Completed!

Set up the Service Account

1. Run the following command to create a Service Account:

```
gcloud iam service-accounts create sqlserver2dc-credentials \
--display-name  "Service Account for SQL Server to Data Catalog
connector" \
--project $PROJECT_ID
```

content_c

2. Now create and download the Service Account Key.

```
gcloud iam service-accounts keys create "sqlserver2dc-
credentials.json" \
--iam-account "sqlserver2dc-
credentials@$PROJECT_ID.iam.gserviceaccount.com"
```

content_c

3. Add the Data Catalog admin role to the Service Account:

```
gcloud projects add-iam-policy-binding $PROJECT_ID \
--member "serviceAccount:sqlserver2dc-
credentials@$PROJECT_ID.iam.gserviceaccount.com" \
--quiet \
--project $PROJECT_ID \
--role "roles/datacatalog.admin"
```

content_c

Click **Check my progress** to verify the objective.

Set Up the Service Account for SQLServer

Check my progress

Assessment Completed!

Execute SQL Server to Dataplex connector

You can build the SQL Server connector yourself by going to this GitHub repository.

To facilitate its usage, we are going to use a docker image.

The variables needed were output by the Terraform config.

1. Change directories into the location of the Terraform scripts:

```
cd infrastructure/terraform/
```

content_c

2. Grab the environment variables:

```
public_ip_address=$(terraform output -raw public_ip_address)
username=$(terraform output -raw username)
password=$(terraform output -raw password)
database=$(terraform output -raw db_name)
```

content_c

3. Change back to the root directory for the example code:

```
cd ~/cloudsql-sqlserver-tooling
```

content_c

4. Run the following command to execute the connector:

```
docker run --rm --tty -v \
"$PWD":/data mesmacosta/sqlserver2datacatalog:stable \
--datacatalog-project-id=$PROJECT_ID \
--datacatalog-location-id=$REGION \
--sqlserver-host=$public_ip_address \
--sqlserver-user=$username \
--sqlserver-pass=$password \
--sqlserver-database=$database
```

content_c

Soon after you should receive the following output:

```
============End sqlserver-to-datacatalog============
```

Click **Check my progress** to verify the objective.

Execute SQL Server to Data Catalog connector

Check my progress

Assessment Completed!

Search for the SQL Server Entries in Dataplex

1. After the script finishes, open the navigation menu and select **Dataplex** from the list of services.

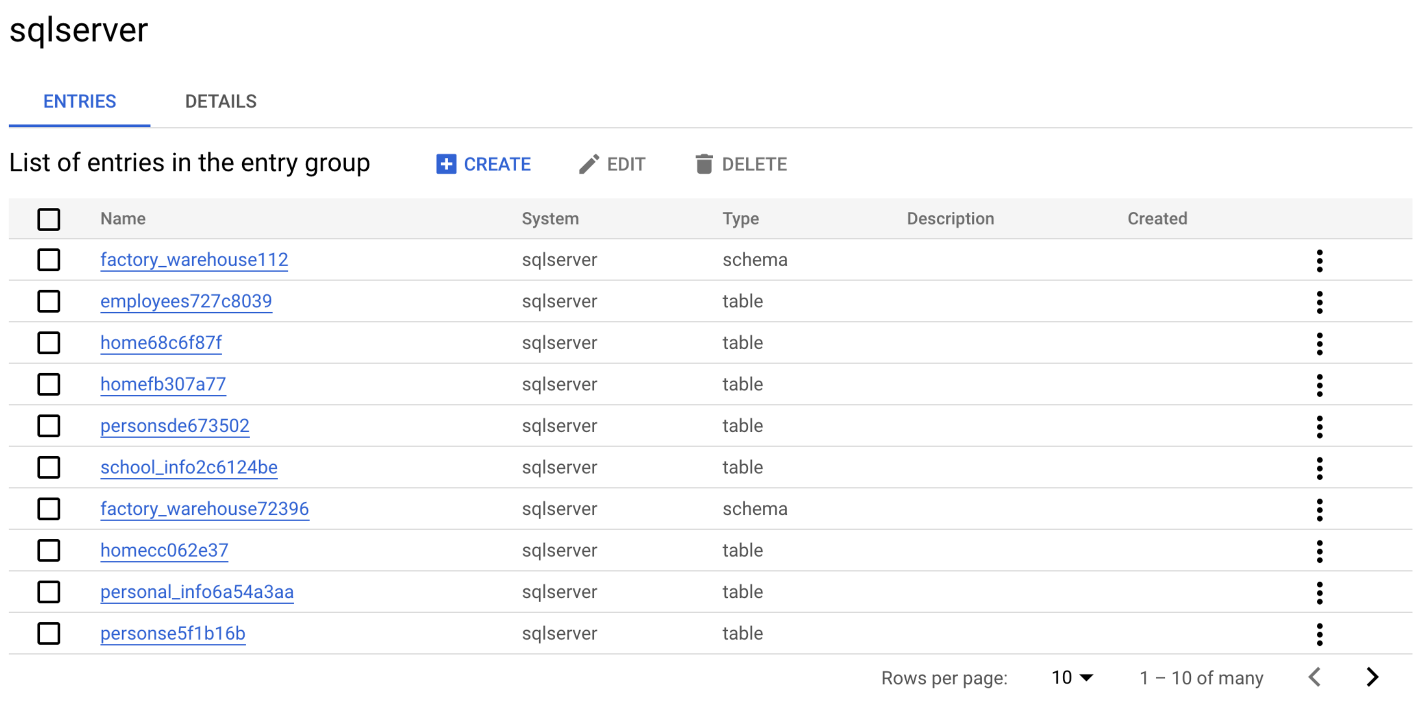
2. In the the **Dataplex** page, click on **Tag Templates**.

You should see your **sqlserver** Tag Templates listed.

3. Next, select **Entry Groups**.

You should see the **sqlserver** Entry Group in the **Entry Groups** list:

4. Now click on the `sqlserver` Entry Group. Your console should resemble the following:



Name	System	Type	Description	Created
factory_warehouse112	sqlserver	schema		
employees727c8039	sqlserver	table		
home68c6f87f	sqlserver	table		
homefb307a77	sqlserver	table		
personsde673502	sqlserver	table		
school_info2c6124be	sqlserver	table		
factory_warehouse72396	sqlserver	schema		
homecc062e37	sqlserver	table		
personal_info6a54a3aa	sqlserver	table		
personse5f1b16b	sqlserver	table		

Rows per page: 10 1 – 10 of many

This is the real value of an Entry Group—you can see all entries that belong to sqlserver using the UI.

5. Click on one of the `warehouse` entries. Look at the Custom entry details and tags.

This is the real value the connector adds — it allows you to have the metadata searchable in Dataplex.

Clean up

1. To delete the created resources, run the following command to delete the SQL Server metadata:

```
./cleanup-db.sh
```

content_c

2. Now execute the cleaner container:

```
docker run --rm --tty -v \
"$PWD":/data mesmacosta/sqlserver-datacatalog-cleaner:stable \
--datacatalog-project-ids=$PROJECT_ID \
```

content_c

```
--rdbms-type=sqlserver \
--table-container-type=schema
```

3. Now run the following command to delete the SQL Server database:

```
./delete-db.sh
```

content_c

4. From the **Navigation menu** click **Dataplex**.

5. Search for **sqlserver**.

You will no longer see the SQL Server Tag Templates in the results:

Ensure you see the following output in Cloud Shell before you move on:

```
 Cloud SQL Instance deleted
 COMPLETED
```

You will now learn how to do the same thing with a PostgreSQL instance.

Task 3. PostgreSQL to Dataplex

Create the PostgreSQL Database

1. Run the following command in Cloud Shell to return to your home directory:

```
cd
```

content_c

2. Run the following command to clone the Github repository:

```
gsutil cp gs://spls/gsp814/cloudsql-postgresql-tooling.zip .
unzip cloudsql-postgresql-tooling.zip
```

content_c

3. Now change your current working directory to the cloned repo directory:

```
cd cloudsql-postgresql-tooling/infrastructure/terraform
```

content_c

4. Run the following command to change the region from `us-central1` to your default assigned region:

```
export REGION=us-west1
sed -i "s/us-central1/$REGION/g" variables.tf
```

content_c

5. Now execute the `init-db.sh` script:

```
cd ~/cloudsql-postgresql-tooling
bash init-db.sh
```

content_c

This will create your PostgreSQL instance and populate it with a random schema. This can take `around 10 to 15 minutes` to complete.

Note: If you get an `Error: Failed to load "tfplan" as a plan file`, re-run the `init-db` script.

Soon after you should receive the following output:

```
CREATE TABLE factory_warehouse69945.home17e97c57 ( house57588 DATE,
paragraph64180 SMALLINT, ip_address61569 JSONB, date_time44962 REAL, food19478
JSONB, state8925 VARCHAR(25), cpf75444 REAL, date_time96090 SMALLINT,
reason7955 CHAR(5), phone_number96292 INT, size97593 DATE, date_time609
CHAR(5), location70431 DATE )
 COMPLETED
```

Click **Check my progress** to verify the objective.

Create the PostgreSQL Database

Check my progress

Set up the Service Account

1. Create a Service Account:

```
gcloud iam service-accounts create postgresql2dc-credentials \
--display-name  "Service Account for PostgreSQL to Data Catalog
connector" \
--project $PROJECT_ID
```

content_c

2. Next create and download the Service Account Key:

```
gcloud iam service-accounts keys create "postgresql2dc-
credentials.json" \
--iam-account "postgresql2dc-
credentials@$PROJECT_ID.iam.gserviceaccount.com"
```

content_c

3. Next add Data Catalog admin role to the Service Account:

```
gcloud projects add-iam-policy-binding $PROJECT_ID \
--member "serviceAccount:postgresql2dc-
credentials@$PROJECT_ID.iam.gserviceaccount.com" \
--quiet \
--project $PROJECT_ID \
--role "roles/datacatalog.admin"
```

content_c

Click **Check my progress** to verify the objective.

Create a Service Account for PostgreSQL

Check my progress

Execute PostgreSQL to Dataplex connector

You can build the PostgreSQL connector yourself by going to this GitHub repository.

To facilitate its usage, we are going to use a docker image.

The variables needed were output by the Terraform config.

1. Change directories into the location of the Terraform scripts:

```
cd infrastructure/terraform/
```

content_c

2. Grab the environment variables:

```
public_ip_address=$(terraform output -raw public_ip_address)
username=$(terraform output -raw username)
password=$(terraform output -raw password)
database=$(terraform output -raw db_name)
```

content_c

3. Change back to the root directory for the example code:

```
cd ~/cloudsql-postgresql-tooling
```

content_c

4. Execute the connector:

```
docker run --rm --tty -v \
"$PWD":/data mesmacosta/postgresql2datacatalog:stable \
--datacatalog-project-id=$PROJECT_ID \
--datacatalog-location-id=$REGION \
--postgresql-host=$public_ip_address \
--postgresql-user=$username \
--postgresql-pass=$password \
--postgresql-database=$database
```

content_c

Soon after you should receive the following output:

```
============End postgresql-to-datacatalog============
```

Click **Check my progress** to verify the objective.

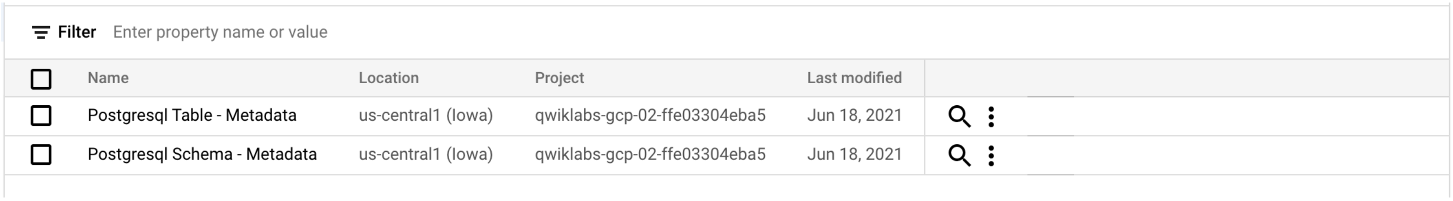
Execute PostgreSQL to Data Catalog connector

Check my progress

Check the results of the script

1. Ensure that you are in the Dataplex home page.
2. Click on **Tag Templates**.

You should see the following postgresql Tag Templates:

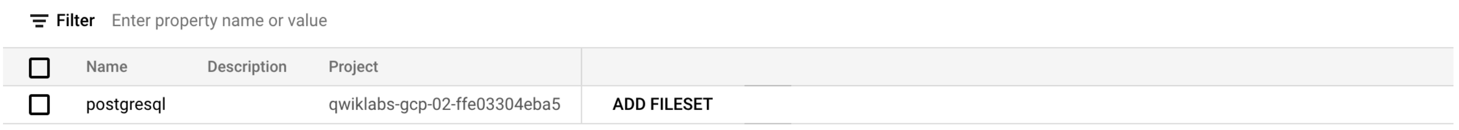


Filter Enter property name or value

Name	Location	Project	Last modified
Postgresql Table - Metadata	us-central1 (Iowa)	qwiklabs-gcp-02-ffe03304eba5	Jun 18, 2021
Postgresql Schema - Metadata	us-central1 (Iowa)	qwiklabs-gcp-02-ffe03304eba5	Jun 18, 2021

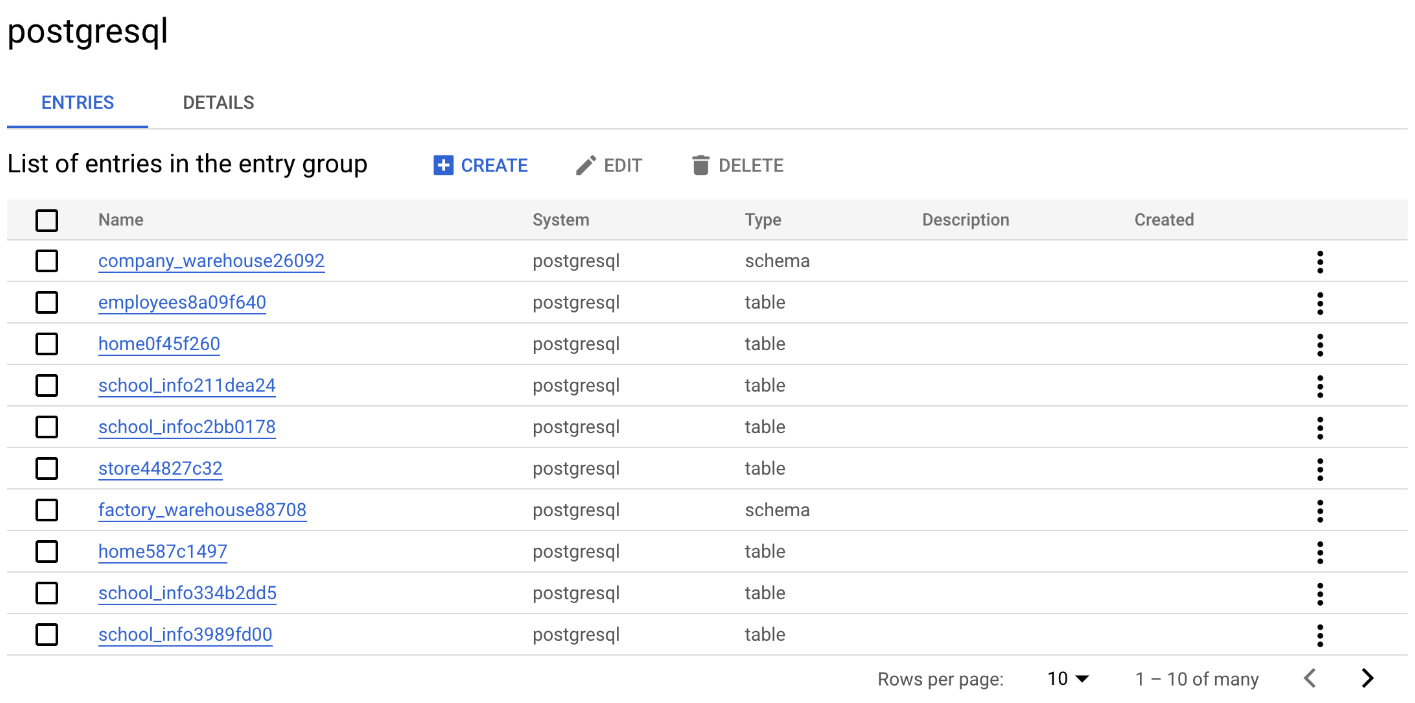
3. Click on **Entry groups**.

You should see the following postgresql Entry Group:



	Name	Description	Project	
☐	postgresql		qwiklabs-gcp-02-ffe03304eba5	ADD FILESET

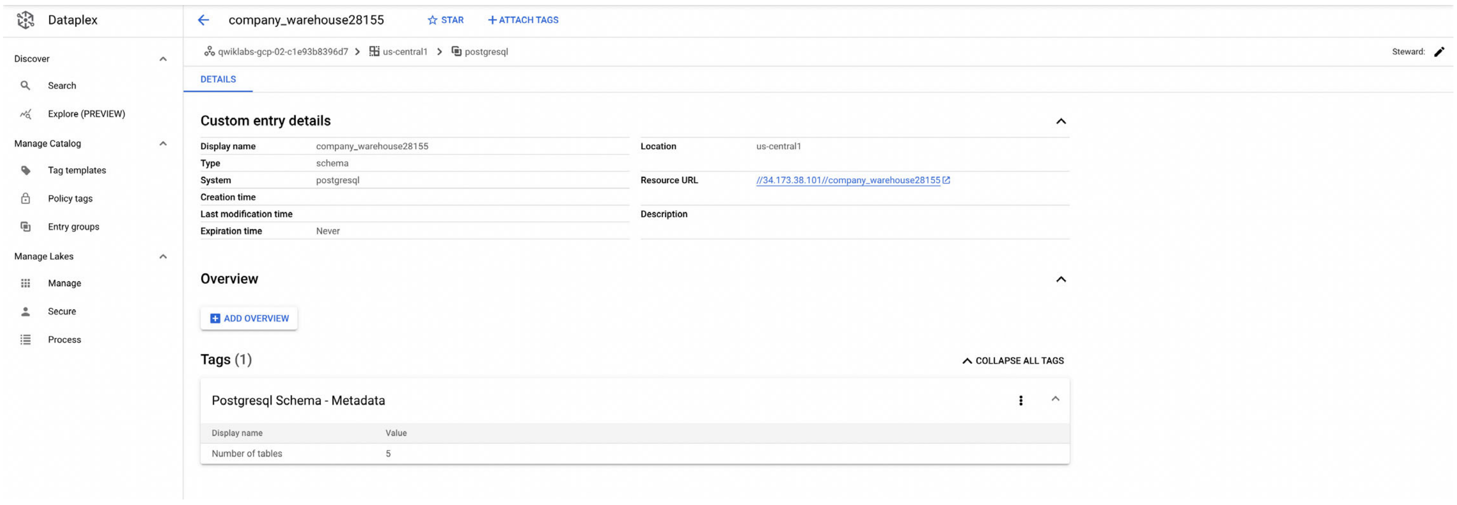
4. Now click on the `postgresql` Entry Group. Your console should resemble the following:



	Name	System	Type	Description	Created
☐	company_warehouse26092	postgresql	schema		
☐	employees8a09f640	postgresql	table		
☐	home0f45f260	postgresql	table		
☐	school_info211dea24	postgresql	table		
☐	school_infoc2bb0178	postgresql	table		
☐	store44827c32	postgresql	table		
☐	factory_warehouse88708	postgresql	schema		
☐	home587c1497	postgresql	table		
☐	school_info334b2dd5	postgresql	table		
☐	school_info3989fd00	postgresql	table		

This is the real value of an Entry Group — you can see all entries that belong to postgresql using the UI.

5. Click on one of the `warehouse` entries. Look at the Custom entry details and tags:



company_warehouse28155

Custom entry details

Display name	company_warehouse28155	Location	us-central1
Type	schema		
System	postgresql	Resource URL	//34.173.38.101//company_warehouse28155
Creation time			
Last modification time		Description	
Expiration time	Never		

Overview

Tags (1)

Postgresql Schema - Metadata

Display name	Value
Number of tables	5

This is the real value the connector adds—it allows you to have the metadata searchable in Dataplex.

Clean up

1. To delete the created resources, run the following command to delete the PostgreSQL metadata:

```
./cleanup-db.sh
```

content_c

2. Now execute the cleaner container:

```
docker run --rm --tty -v \
"$PWD":/data mesmacosta/postgresql-datacatalog-cleaner:stable \
--datacatalog-project-ids=$PROJECT_ID \
--rdbms-type=postgresql \
--table-container-type=schema
```

content_c

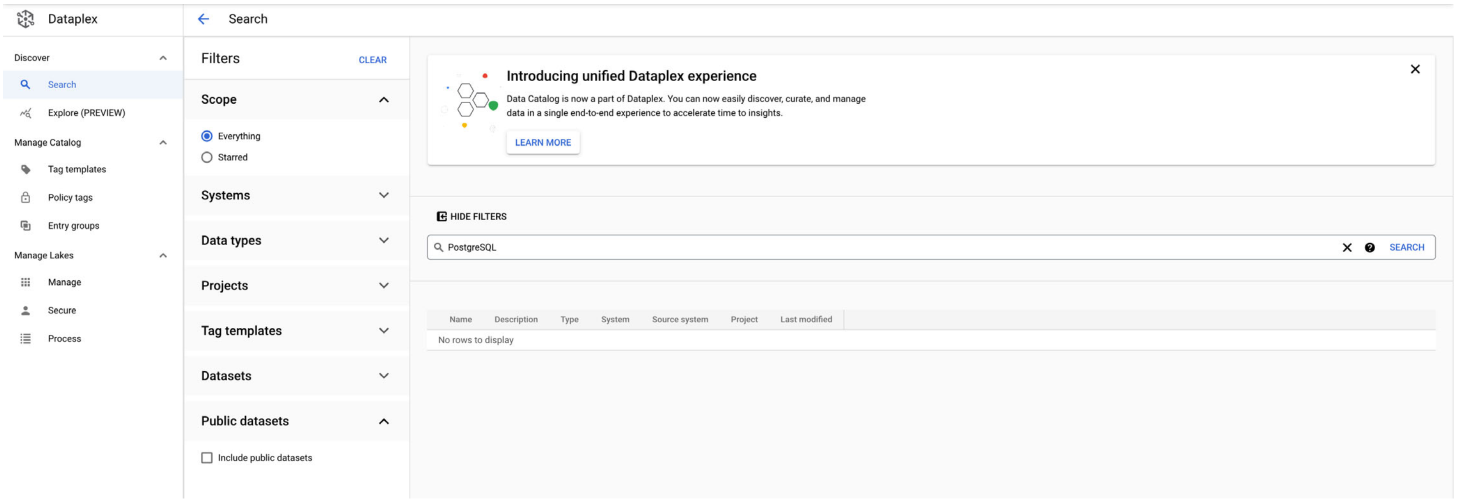
3. Finally, delete the PostgreSQL database:

```
./delete-db.sh
```

content_c

4. Now, from the **Navigation menu** click on **Dataplex**.

5. Search for **PostgreSQL**. You will no longer see the PostgreSQL Tag Templates in the results:



Introducing unified Dataplex experience

Data Catalog is now a part of Dataplex. You can now easily discover, curate, and manage data in a single end-to-end experience to accelerate time to insights.

PostgreSQL

No rows to display

Ensure you see the following output in Cloud Shell before you move on:

```
Cloud SQL Instance deleted
COMPLETED
```

You will now learn how to do the same thing with a MySQL instance.

Task 4. MySQL to Dataplex

Create the MySQL database

1. Run the following command in Cloud Shell to return to your home directory:

```
cd                                          content_c
```

2. Run the following command to download the scripts to create and populate your MySQL instance:

```
gsutil cp gs://spls/gsp814/cloudsql-mysql-tooling.zip .
unzip cloudsql-mysql-tooling.zip
```

content_c

3. Now change your current working directory to the cloned repo directory:

```
cd cloudsql-mysql-tooling/infrastructure/terraform
```

content_c

4. Run the following command to change the region from `us-central1` to your default assigned region:

```
export REGION=us-west1
sed -i "s/us-central1/$REGION/g" variables.tf
```

content_c

5. Next execute the `init-db.sh` script:

```
cd ~/cloudsql-mysql-tooling
bash init-db.sh
```

content_c

This will create your MySQL instance and populate it with a random schema. After a few minutes, you should receive the following output:

```
CREATE TABLE factory_warehouse14342.persons88a5ebc4 ( address9634 TEXT,
cpf12934 FLOAT, food88799 BOOL, food4761 LONGTEXT, credit_card44049 FLOAT,
city8417 TINYINT, name76076 DATETIME, address19458 TIME, reason49953 DATETIME )
 COMPLETED
```

Note: If you get an `Error: Failed to load "tfplan" as a plan file`, re-run the `init-db` script.

Click **Check my progress** to verify the objective.

Create the MySQL Database

Check my progress

Set up the Service Account

1. Run the following to create a Service Account:

```
gcloud iam service-accounts create mysql2dc-credentials \
--display-name  "Service Account for MySQL to Data Catalog connector"
\
--project $PROJECT_ID
```

content_c

2. Next, create and download the Service Account Key:

```
gcloud iam service-accounts keys create "mysql2dc-credentials.json" \
--iam-account "mysql2dc-
credentials@$PROJECT_ID.iam.gserviceaccount.com"
```

content_c

3. Next add Data Catalog admin role to the Service Account:

```
gcloud projects add-iam-policy-binding $PROJECT_ID \
--member "serviceAccount:mysql2dc-
credentials@$PROJECT_ID.iam.gserviceaccount.com" \
--quiet \
--project $PROJECT_ID \
--role "roles/datacatalog.admin"
```

content_c

Click **Check my progress** to verify the objective.

Create a Service Account for MySQL

Check my progress

Execute MySQL to Dataplex connector

You can build the MySQL connector yourself by going to this GitHub repository.

To facilitate its usage, this lab uses a docker image.

The variables needed were output by the Terraform config.

1. Change directories into the location of the Terraform scripts:

```
cd infrastructure/terraform/
```

content_c

2. Grab the environment variables:

```
public_ip_address=$(terraform output -raw public_ip_address)
username=$(terraform output -raw username)
password=$(terraform output -raw password)
database=$(terraform output -raw db_name)
```

content_c

3. Change back to the root directory for the example code:

```
cd ~/cloudsql-mysql-tooling
```

content_c

4. Execute the connector:

```
docker run --rm --tty -v \
"$PWD":/data mesmacosta/mysql2datacatalog:stable \
--datacatalog-project-id=$PROJECT_ID \
--datacatalog-location-id=$REGION \
--mysql-host=$public_ip_address \
--mysql-user=$username \
```

content_c

```
--mysql-pass=$password \
--mysql-database=$database
```

Soon after you should receive the following output:

```
============End mysql-to-datacatalog============
```

Click **Check my progress** to verify the objective.

Execute MySQL to Data Catalog connector

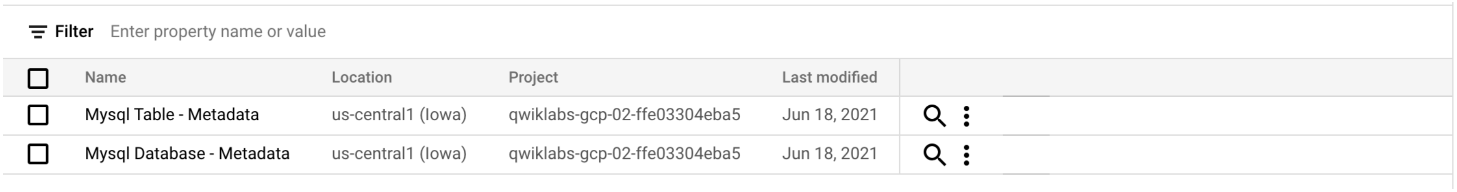
Check my progress

Check the results of the script

1. Ensure that you are in the Dataplex home page.

2. Click on **Tag Templates**.

You should see the following mysql Tag Templates:

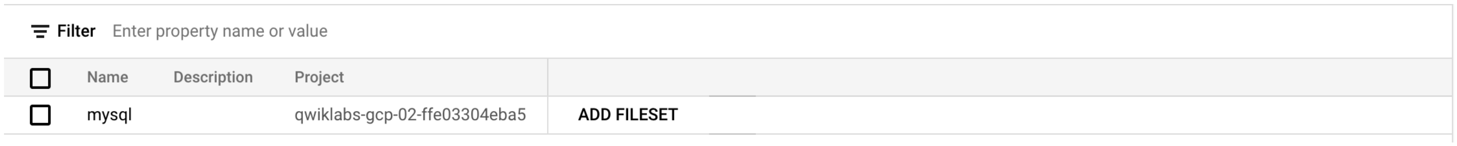


Filter Enter property name or value

Name	Location	Project	Last modified
Mysql Table - Metadata	us-central1 (Iowa)	qwiklabs-gcp-02-ffe03304eba5	Jun 18, 2021
Mysql Database - Metadata	us-central1 (Iowa)	qwiklabs-gcp-02-ffe03304eba5	Jun 18, 2021

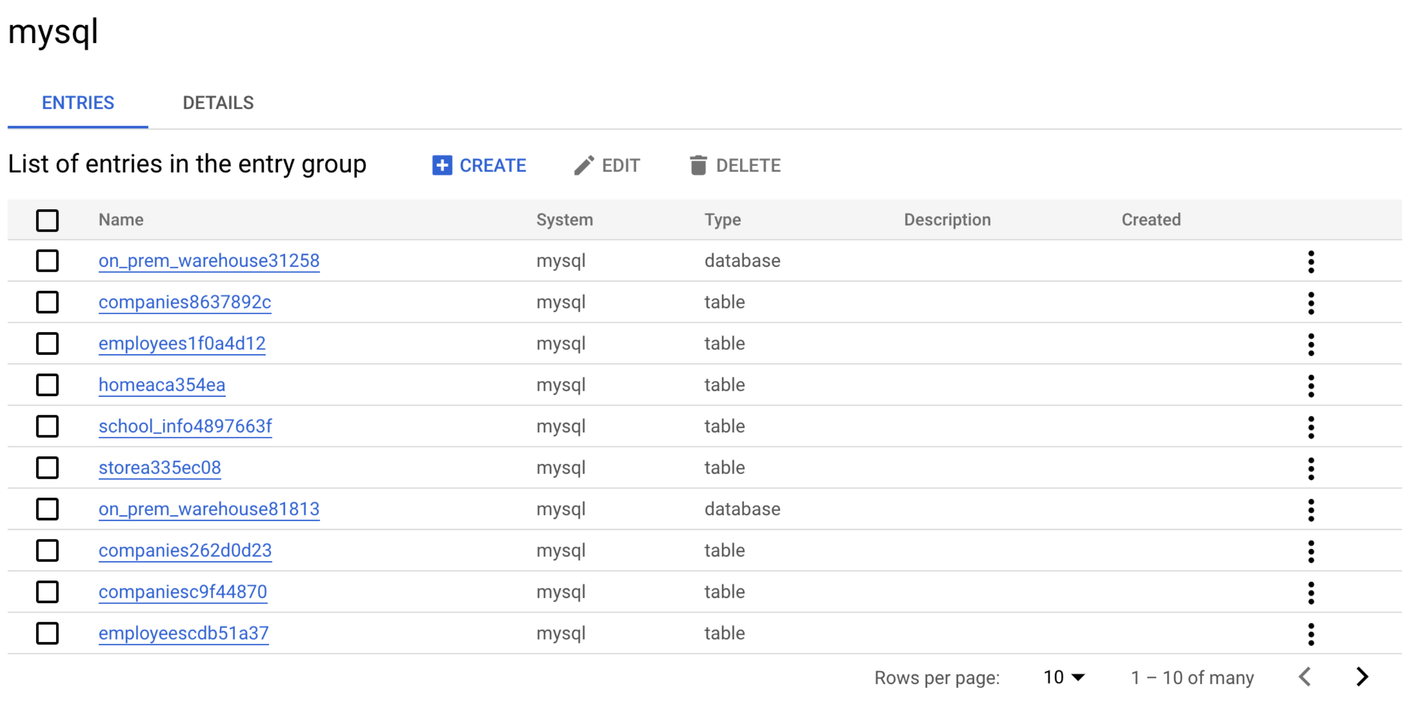
3. Click on **Entry groups**.

You should see the following mysql Entry Group:


Filter Enter property name or value

Name	Description	Project	
mysql		qwiklabs-gcp-02-ffe03304eba5	ADD FILESET

4. Now click on the `mysql` Entry Group. Your console should resemble the following:



List of entries in the entry group

Name	System	Type	Description	Created
on_prem_warehouse31258	mysql	database		
companies8637892c	mysql	table		
employees1f0a4d12	mysql	table		
homeaca354ea	mysql	table		
school_info4897663f	mysql	table		
storea335ec08	mysql	table		
on_prem_warehouse81813	mysql	database		
companies262d0d23	mysql	table		
companiesc9f44870	mysql	table		
employeescdb51a37	mysql	table		

Rows per page: 10 1 – 10 of many

This is the real value of an Entry Group — you can see all entries that belong to MySQL using the UI.

5. Click on one of the `warehouse` entries. Look at the Custom entry details and tags.

This is the real value the connector adds — it allows you to have the metadata searchable in Dataplex.

Clean up

1. To delete the created resources, run the following command to delete the MySQL metadata:

```
./cleanup-db.sh
```

content_c

2. Now execute the cleaner container:

```
docker run --rm --tty -v \
"$PWD":/data mesmacosta/mysql-datacatalog-cleaner:stable \
--datacatalog-project-ids=$PROJECT_ID \
--rdbms-type=mysql \
--table-container-type=database
```

content_c

3. Finally, delete the PostgreSQL database:

```
./delete-db.sh
```

content_c

4. From the **Navigation menu** click **Dataplex**.

5. Search for **MySQL**. You will no longer see the MySQL Tag Templates in the results.

Ensure you see the following output in Cloud Shell before you move on:

```
  Cloud SQL Instance deleted
  COMPLETED
```

Congratulations!

Congratulations! In this lab, you learned how to build and execute MySQL, PostgreSQL, and SQL Server to Dataplex connectors. You also learned how to search for SQL Server, PostgreSQL, and MySQL entries in Data Catalog within Dataplex. You can now use this knowledge to build your own connectors.

Finish your course

This self-paced lab is part of the BigQuery for Data Warehousing, BigQuery for Marketing Analysts, and Data Catalog Fundamentals courses. See the Google Cloud Skills Boost catalog to see all available courses.

Next steps / Learn more

- Read the Data Catalog Overview
- Learn How to search with Data Catalog
- Browse the Overview of APIs and Client Libraries

Google Cloud training and certification

...helps you make the most of Google Cloud technologies. Our classes include technical skills and best practices to help you get up to speed quickly and continue your learning journey. We offer fundamental to advanced level training, with on-demand, live, and virtual options to suit your busy schedule. Certifications help you validate and prove your skill and expertise in Google Cloud technologies.

Manual Last Updated October 17, 2023