Build and Execute MySQL, PostgreSQL, and SQLServer to Data Catalog Connectors | Google Cloud Skills Boost

Qwiklabs : 20-25 minutes

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.

What you'll do

- 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 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 Console**. 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** from the **Lab Details** panel and paste it into the **Sign in** dialog. Click **Next**.

4. Copy the **Password** from the **Lab Details** panel and paste it into the **Welcome** dialog. Click **Next.**

    Important: You must use the credentials from the left panel. Do not use your Google Cloud Skills Boost credentials. **Note:** Using your own Google Cloud account for this lab may incur extra charges.

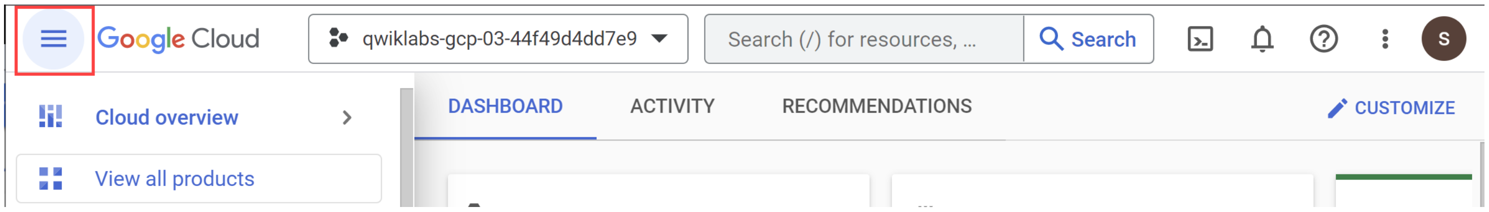
5. Click through the subsequent pages:

    - 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 Cloud Console opens in this tab.

Note: You can view the menu with a list of Google Cloud Products and Services by clicking 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**. The output contains a line that declares the **PROJECT_ID** for this session:

Your Cloud Platform project in this session is set to YOUR_PROJECT_ID

`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
```

3. Click **Authorize**.

4. Your output should now look like this:

Output:

```
ACTIVE: * ACCOUNT: student-01-xxxxxxxxxxxx@qwiklabs.net To set the active account, run: $ gcloud config set account `ACCOUNT`
```

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

```
gcloud config list project
```

Output:

```
[core] project = <project_ID>
```

Example output:

[core] project = qwiklabs-gcp-44776a13dea667a6 **Note:** For full documentation of `gcloud`, in Google Cloud, refer to the gcloud CLI overview guide.

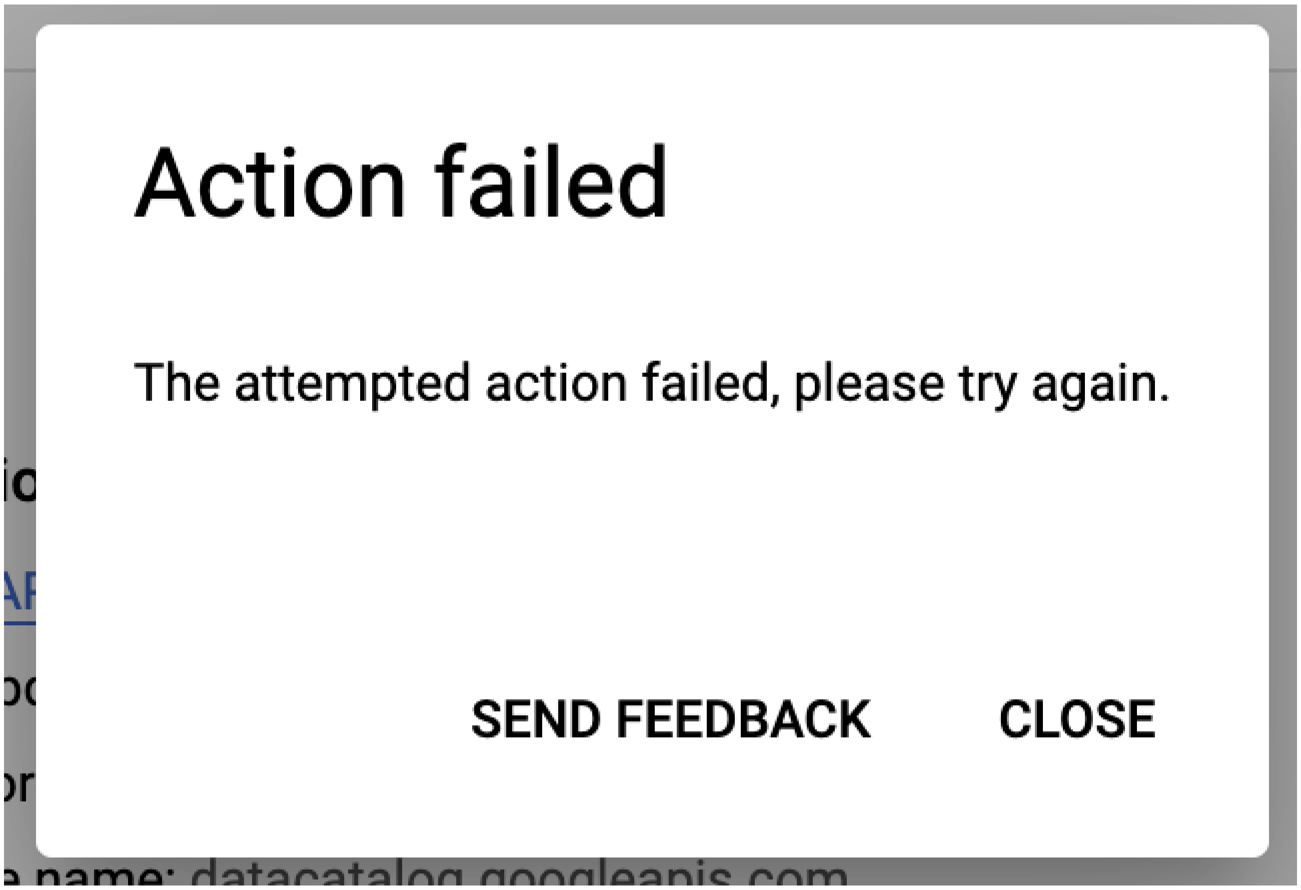
Task 1. Enable the Data Catalog API

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**.

If you run into the following error after trying to enable the Data Catalog API:

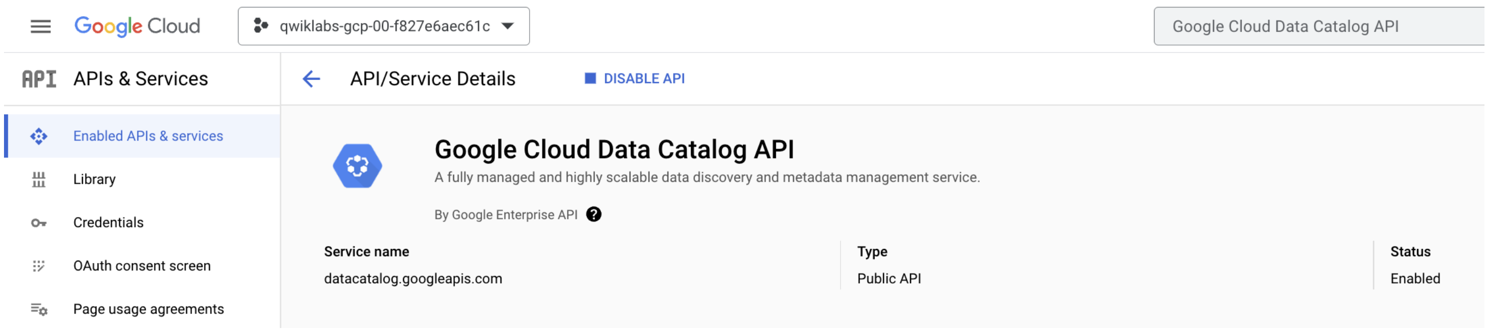


Action failed

The attempted action failed, please try again.

1. Click **Close**.
2. Refresh your browser tab.
3. Click **Enable** again.

The Data Catalog API should be successfully enabled:



Google Cloud Data Catalog API

A fully managed and highly scalable data discovery and metadata management service.

By Google Enterprise API

Service name	Type	Status
datacatalog.googleapis.com	Public API	Enabled

Click **Check my progress** to verify the objective. Enable the Data Catalog API

Task 2. SQL Server to Dataplex

Start by setting up your environment.

1. Run the following command to set your Project ID, replacing `<YOUR_PROJECT_ID>` with the Project ID found in the connection details panel:

gcloud config set project <YOUR_PROJECT_ID>

2. Next set it as an environment variable:

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

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

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

cd cloudsql-sqlserver-tooling

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

bash init-db.sh

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

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
```

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"
```

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"
```

Click **Check my progress** to verify the objective. Set Up the Service Account for SQLServer

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/
```

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)
```

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

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

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=us-central1 \ --sqlserver-host=$public_ip_address \ --sqlserver-user=$username \ --sqlserver-pass=$password \ --sqlserver-database=$database
```

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

Search for the SQL Server Entries in Dataplex

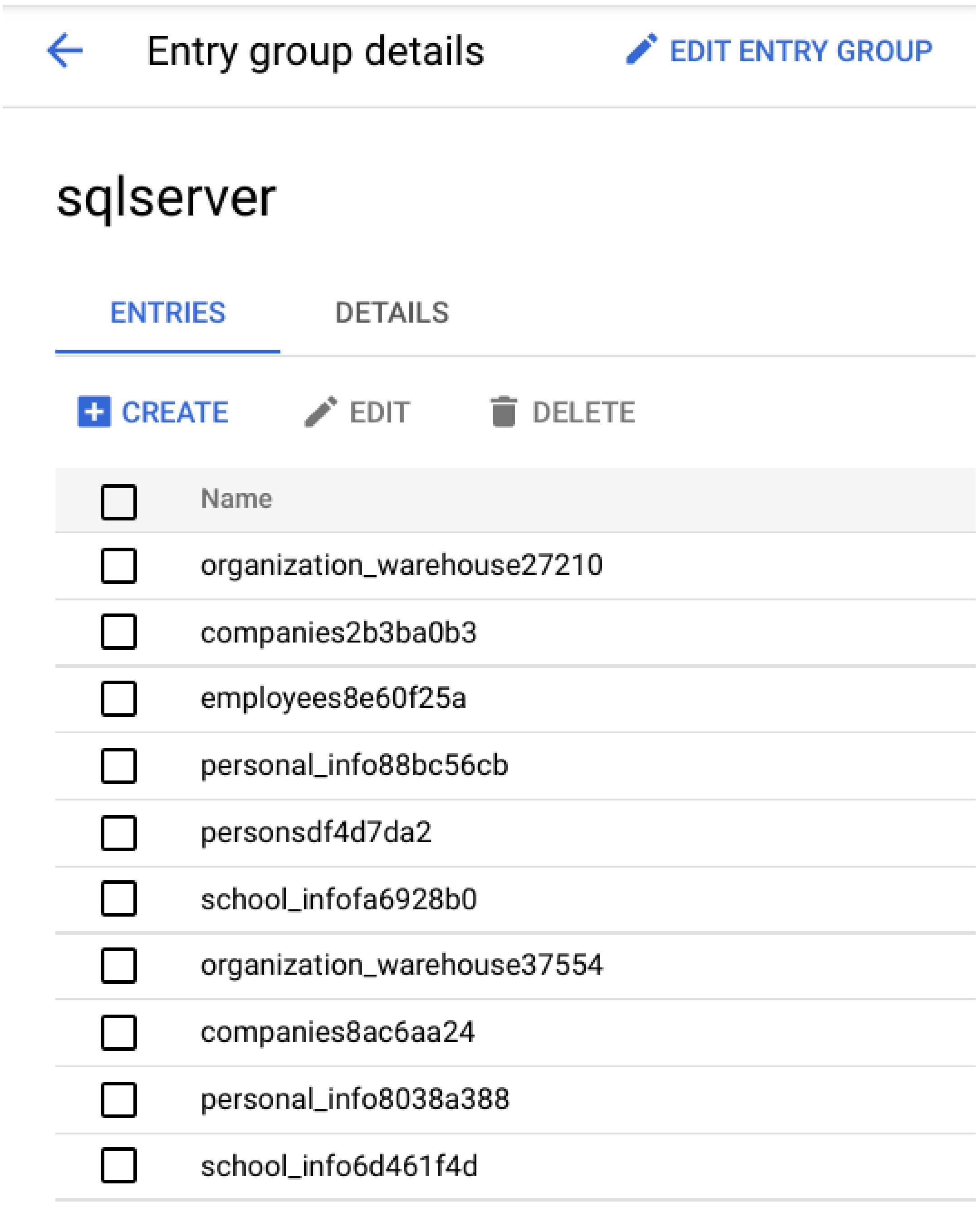
1. After the script finishes, open the navigation menu and select **Dataplex** from the list of services.
2. In the the **Dataplex** page, click on **Tag Templates**.

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:



sqlserver

Name
organization_warehouse27210
companies2b3ba0b3
employees8e60f25a
personal_info88bc56cb
personsdf4d7da2
school_infofa6928b0
organization_warehouse37554
companies8ac6aa24
personal_info8038a388
school_info6d461f4d

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
```

2. Now execute the cleaner container:

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

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

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

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
```

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

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

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

```
cd cloudsql-postgresql-tooling
```

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

```
bash init-db.sh
```

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

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
```

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"
```

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"
```

Click **Check my progress** to verify the objective. Create a Service Account for postgresql

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/
```

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)
```

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

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

4. Execute the connector:

```
docker run --rm --tty -v \ "$PWD":/data mesmacosta/postgresql2datacatalog:stable \ --datacatalog-project-id=$PROJECT_ID \ --datacatalog-location-id=us-central1 \ --postgresql-host=$public_ip_address \ --postgresql-user=$username \ --postgresql-pass=$password \ --postgresql-database=$database
```

Soon after you should receive the following output:

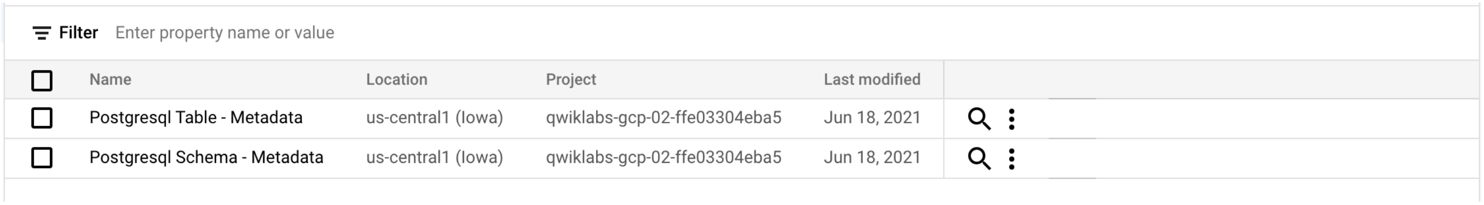
============End postgresql-to-datacatalog============

Click **Check my progress** to verify the objective. Execute PostgreSQL to Data Catalog connector

Check 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:

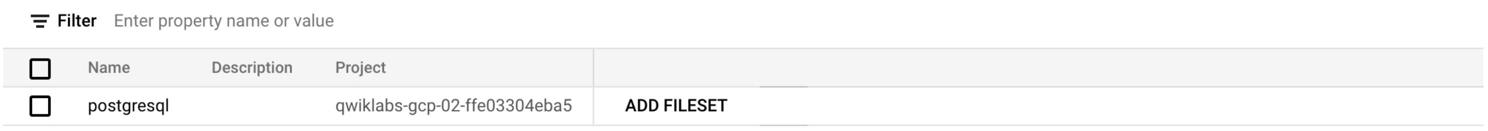


Filter Enter property name or value

☐	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**.

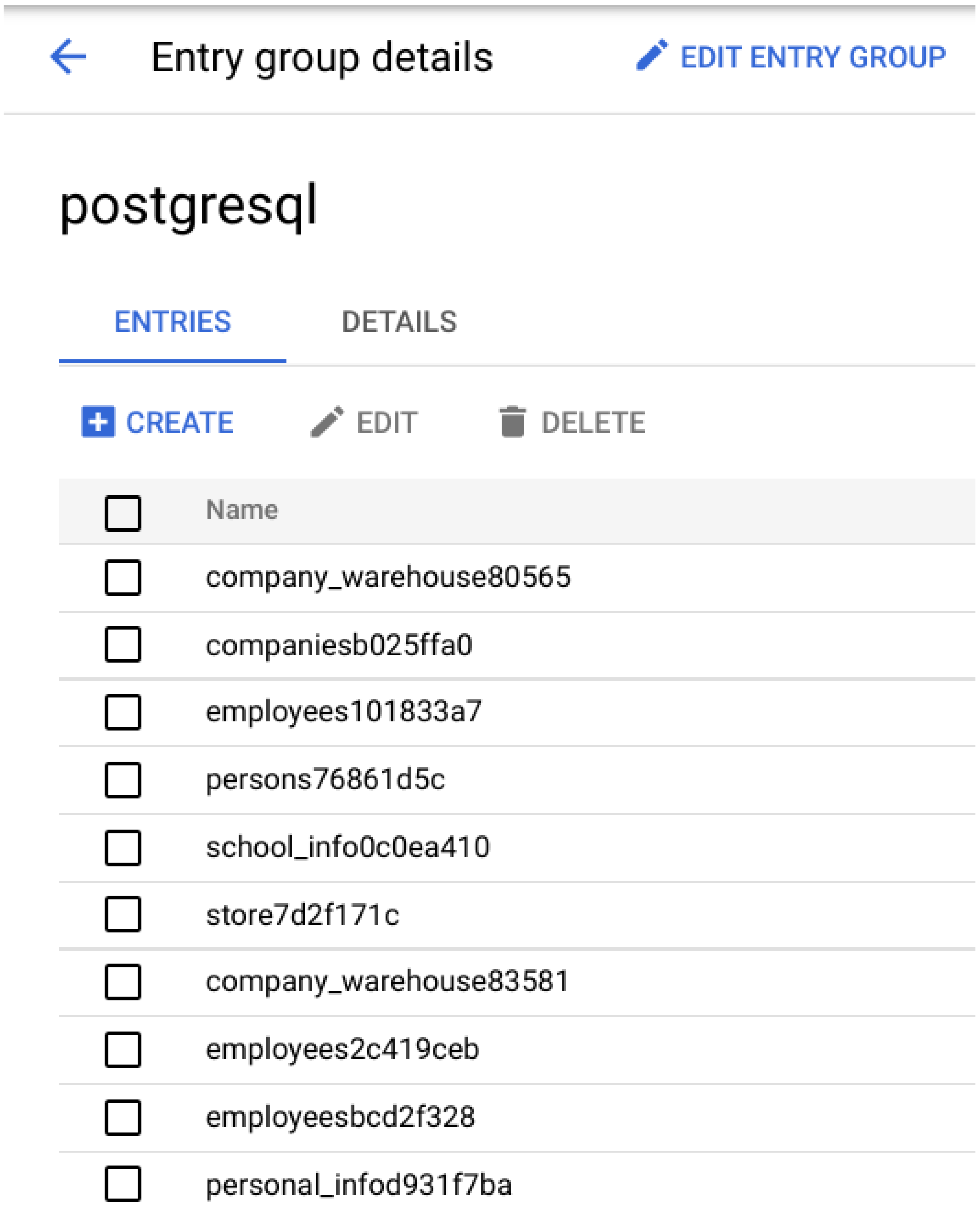
You should see the following postgresql Entry Group:



Filter Enter property name or value

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

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

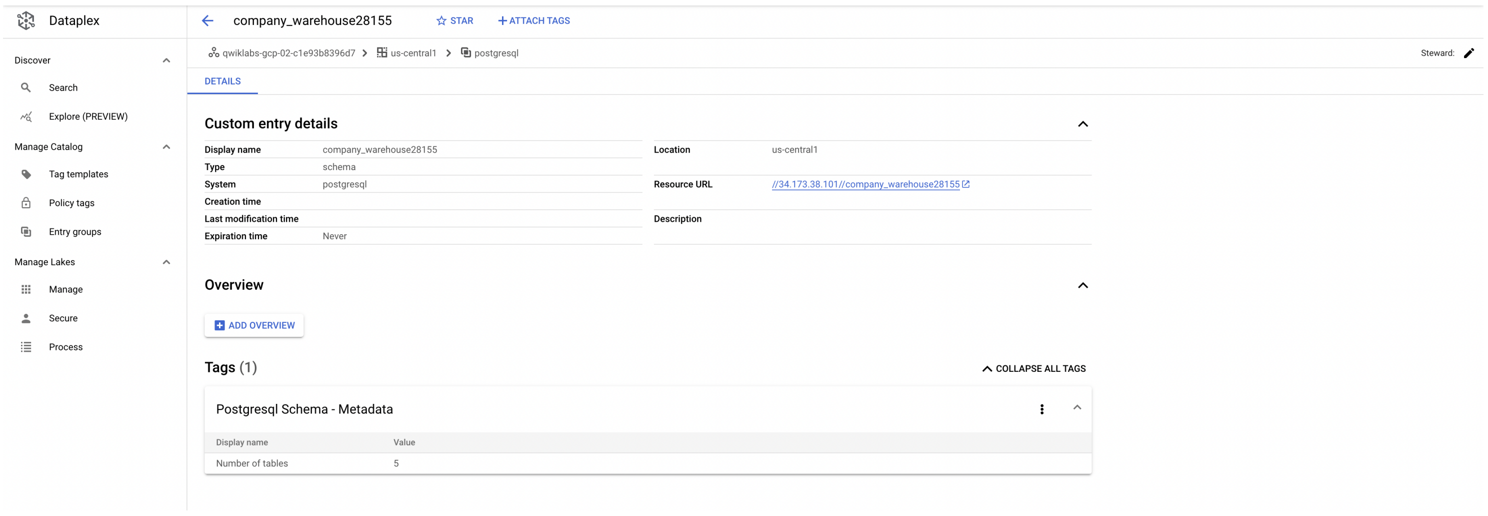


postgresql

☐	Name
☐	company_warehouse80565
☐	companiesb025ffa0
☐	employees101833a7
☐	persons76861d5c
☐	school_info0c0ea410
☐	store7d2f171c
☐	company_warehouse83581
☐	employees2c419ceb
☐	employeesbcd2f328
☐	personal_infod931f7ba

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

qwiklabs-gcp-02-c1e93b8396d7 > us-central1 > postgresql

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

2. Now execute the cleaner container:

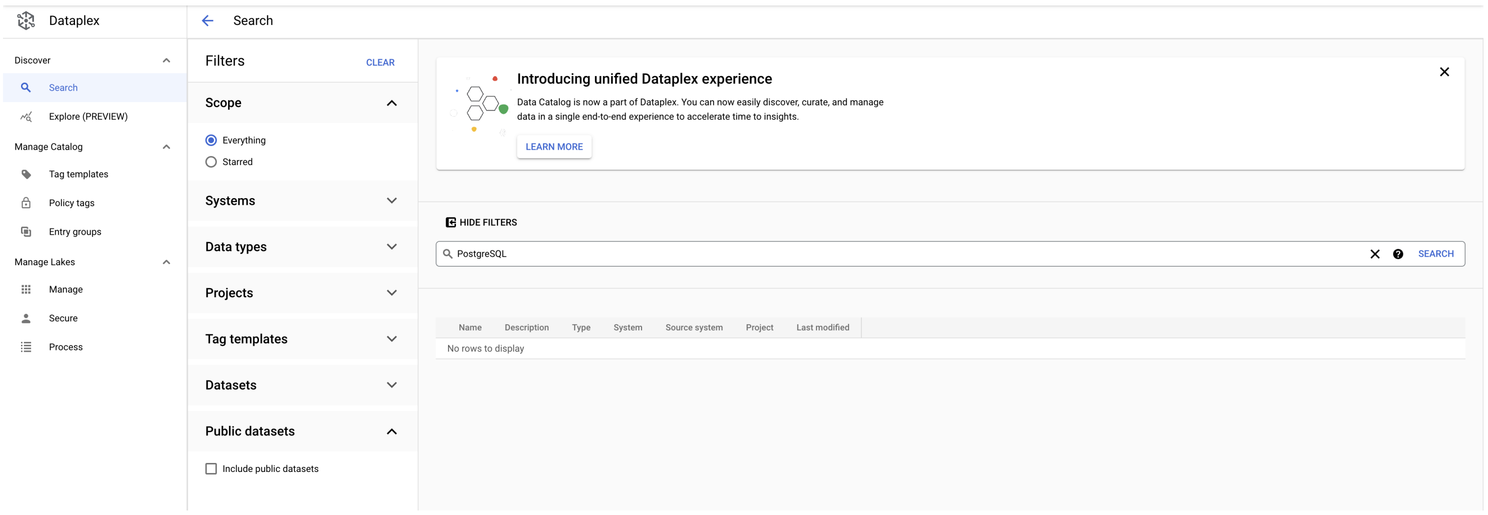
docker run --rm --tty -v \ "$PWD":/data mesmacosta/postgresql-datacatalog-cleaner:stable \ --datacatalog-project-ids=$PROJECT_ID \ --rdbms-type=postgresql \ --table-container-type=schema

3. Finally, delete the PostgreSQL database:

./delete-db.sh

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

Name	Description	Type	System	Source system	Project	Last modified
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

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

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

cd cloudsql-mysql-tooling

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

bash init-db.sh

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

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

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"

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"
```

Click **Check my progress** to verify the objective. Create a Service Account for MySQL

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/
```

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)
```

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

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

4. Execute the connector:

```
docker run --rm --tty -v \ "$PWD":/data mesmacosta/mysql2datacatalog:stable \ --datacatalog-project-id=$PROJECT_ID \ --datacatalog-location-id=us-central1 \ --mysql-host=$public_ip_address \ --mysql-user=$username \ --mysql-pass=$password \ --mysql-database=$database
```

Soon after you should receive the following output:

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

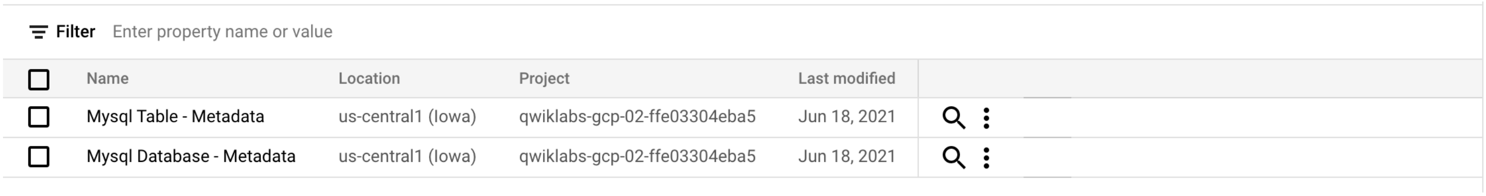
Click **Check my progress** to verify the objective. Execute MySQL to Data Catalog connector

Check 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:

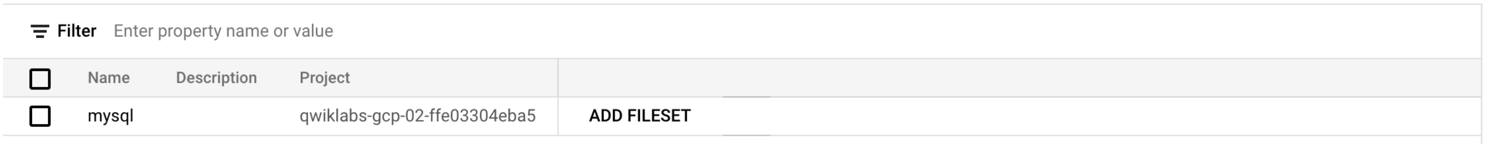


Filter Enter property name or value

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**.

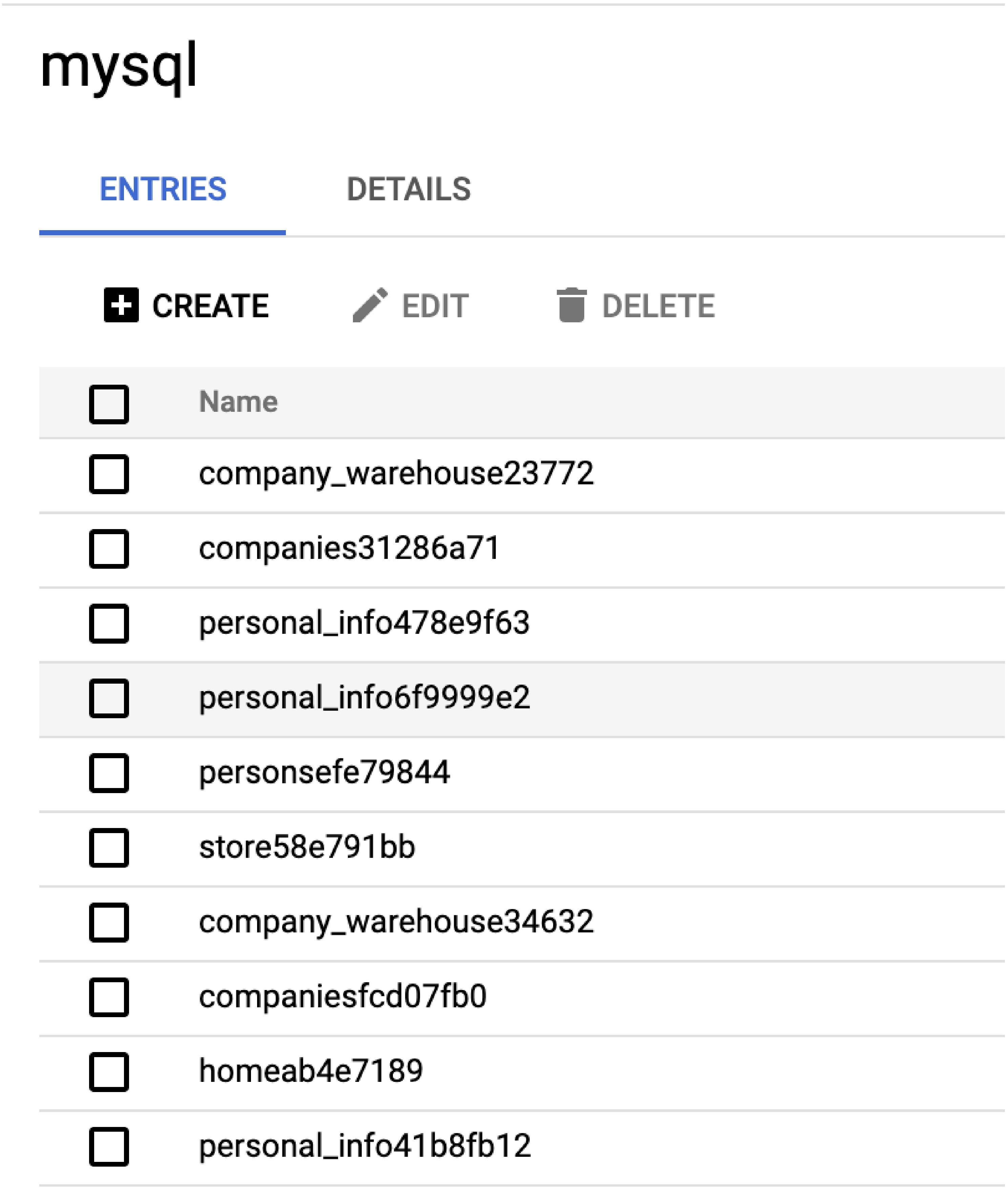
You should see the following mysql Entry Group:



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:



☐	Name
☐	company_warehouse23772
☐	companies31286a71
☐	personal_info478e9f63
☐	personal_info6f9999e2
☐	personsefe79844
☐	store58e791bb
☐	company_warehouse34632
☐	companiesfcd07fb0
☐	homeab4e7189
☐	personal_info41b8fb12

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
```

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
```

3. Finally, delete the PostgreSQL database:

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

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!

You received hands-on practice with Dataplex connectors:

- 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.

Finish your quest

This self-paced lab is part of the BigQuery for Data Warehousing, BigQuery for Marketing Analysts, and Data Catalog Fundamentals quests. A quest is a series of related labs that form a learning path. Completing a quest earns you a badge to recognize your achievement. You can make your badge or badges public and link to them in your online resume or social media account. Enroll in any quest that contains this lab and get immediate completion credit. See the Google Cloud Skills Boost catalog to see all available quests.

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 June 28, 2023

Lab Last Tested January 23, 2023