

Dataflow: Qwik Start – Templates

Objective

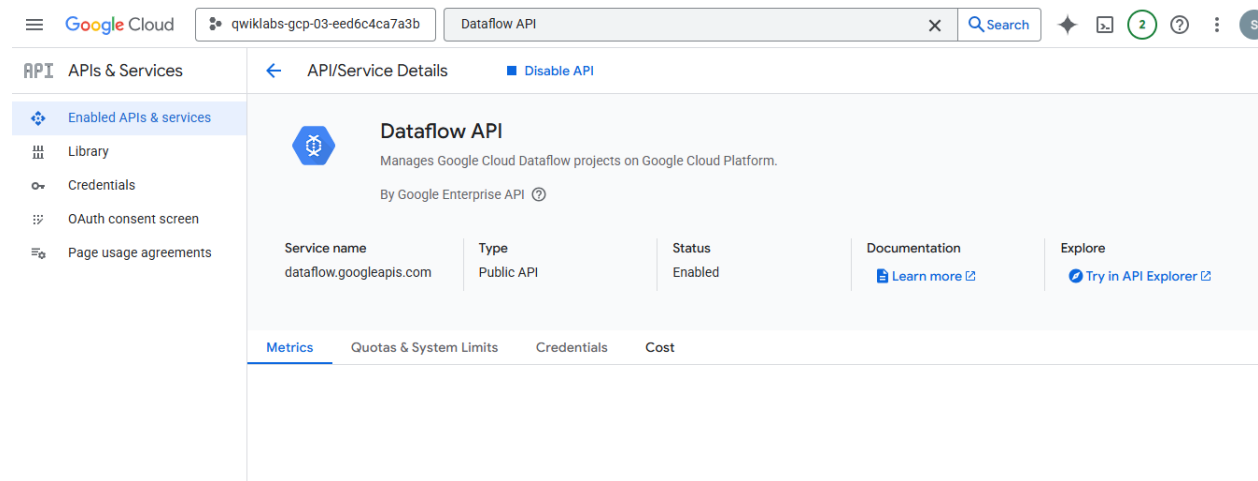
The goal of this lab was to create a real-time streaming data pipeline using Google Cloud's **Dataflow** service and a **Google-provided template (Pub/Sub to BigQuery)**. The pipeline streams taxi ride data from a Pub/Sub topic into a BigQuery table for real-time analysis.

Tools and Services Used

- **Google Cloud Console & Cloud Shell**
- **BigQuery**
- **Cloud Storage**
- **Pub/Sub**
- **Dataflow**
- **gcloud and bq CLI tools**

Task 1: Enable the Dataflow API

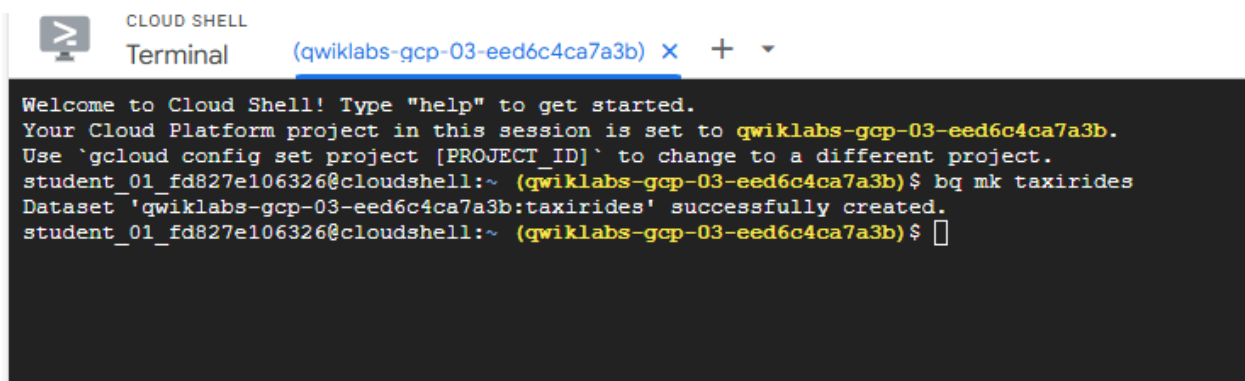
- Navigated to the **Dataflow API** section in Google Cloud Console.
- Disabled and then re-enabled the API to ensure it is ready for use.



Task 2. Create a BigQuery dataset, BigQuery table, and Cloud Storage bucket using Cloud Shell

1. Run the following command to create a dataset called taxirides:

```
bq mk taxirides
```



```
CLOUD SHELL
Terminal (qwiklabs-gcp-03-eed6c4ca7a3b) x + v

Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to qwiklabs-gcp-03-eed6c4ca7a3b.
Use `gcloud config set project [PROJECT_ID]` to change to a different project.
student_01_fd827e106326@cloudshell:~ (qwiklabs-gcp-03-eed6c4ca7a3b)$ bq mk taxirides
Dataset 'qwiklabs-gcp-03-eed6c4ca7a3b:taxirides' successfully created.
student_01_fd827e106326@cloudshell:~ (qwiklabs-gcp-03-eed6c4ca7a3b)$
```

2. Run the following command to do so:

```
bq mk
--time_partitioning_field timestamp
--schema ride_id:string,point_idx:integer,latitude:float,longitude:float,
timestamp:timestamp,meter_reading:float,meter_increment:float,ride_status:string,
passenger_count:integer -t taxirides.realtime
```

```
student_01_fd827e106326@cloudshell:~ (qwiklabs-gcp-03-eed6c4ca7a3b)$ bq mk \
--time_partitioning_field timestamp \
--schema ride_id:string,point_idx:integer,latitude:float,longitude:float,\
timestamp:timestamp,meter_reading:float,meter_increment:float,ride_status:string,\
passenger_count:integer -t taxirides.realtime
Table 'qwiklabs-gcp-03-eed6c4ca7a3b:taxirides.realtime' successfully created.
student_01_fd827e106326@cloudshell:~ (qwiklabs-gcp-03-eed6c4ca7a3b)$
```

Create a Cloud Storage bucket using Cloud Shell

Now that we have our table instantiated, let's create a bucket.

Use the Project ID as the bucket name to ensure a globally unique name: <Bucket Name>

- Run the following commands to do so:

```
export BUCKET_NAME="Bucket Name"
```

```
gsutil mb gs://$BUCKET_NAME/
```

```
student_01_fd827e106326@cloudshell:~ (qwiklabs-gcp-03-eed6c4ca7a3b)$ export BUCKET_NAME=qwiklabs-gcp-03-eed6c4ca7a3b
student_01_fd827e106326@cloudshell:~ (qwiklabs-gcp-03-eed6c4ca7a3b)$ gsutil mb gs://$BUCKET_NAME/
Creating gs://qwiklabs-gcp-03-eed6c4ca7a3b/...
student_01_fd827e106326@cloudshell:~ (qwiklabs-gcp-03-eed6c4ca7a3b)$
```

Task 3. Create a BigQuery dataset, BigQuery table, and Cloud Storage bucket using the Google Cloud console

1. From the left-hand menu, in the Big Data section, click on **BigQuery**.
2. Then click **Done**.
3. Click on the three dots next to your project name under the **Explorer** section, then click **Create dataset**.
4. Input taxirides as your dataset ID:
5. Select **us (multiple regions in United States)** in Data location.
6. Leave all of the other default settings in place and click **CREATE DATASET**.

Create dataset

Project ID *

qwiklabs-gcp-03-eed6c4ca7a3b

[Change](#)

Dataset ID *

taxirides

! Dataset already exists

Location type ?



Region

Specify a region to colocate your datasets with other Google Cloud services.



Multi-region

Allow BigQuery to select a region within a group to achieve higher quota limits.



Some locations have been restricted due to a policy set by your organization. [Learn more about restricting locations.](#)

Multi-region *

US (multiple regions in United States)

External Dataset

The selected region supports the following external dataset types: Cloud Spanner



Link to an external dataset ?

Create dataset

Cancel

7. You should now see the taxirides dataset underneath your project ID in the left-hand console.
8. Click on the three dots next to taxirides dataset and select **Open**.
9. Then select **CREATE TABLE** in the right-hand side of the console.
10. In the **Destination** > **Table Name** input, enter realtime.
11. Under Schema, toggle the **Edit as text** slider and enter the following:

ride_id:string,point_idx:integer,latitude:float,longitude:float,timestamp:timestamp,
meter_reading:float,meter_increment:float,ride_status:string,passenger_count:integer

Create table

Table *

realtime

Maximum name size is 1,024 UTF-8 bytes. Unicode letters, marks, numbers, connectors, dashes, and spaces are allowed.

Table type

Native table

☐ Create a BigQuery table for Apache Iceberg [Preview](#)

Schema

☒ Edit as text

Press Alt+F1 for Accessibility Options.

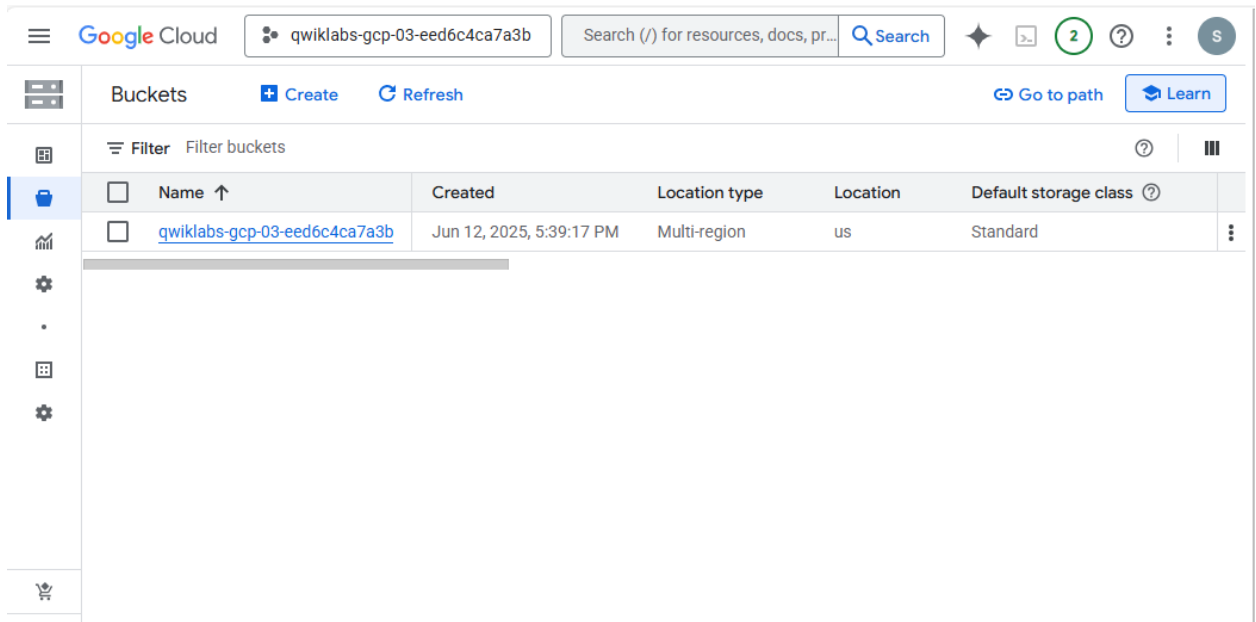
```
1 ride_id:string,point_idx:integer,latitude:float,longitude:float,timestamp:timestamp,  
2 meter_reading:float,meter_increment:float,ride_status:string,passenger_count:integer
```

[Create table](#) Cancel

12. Now, click **Create table**.

Create a Cloud Storage bucket using the Cloud console

1. Go back to the Cloud Console and navigate to **Cloud Storage > Buckets > Create bucket**.
2. Use the Project ID as the bucket name to ensure a globally unique name: <Bucket Name>
3. Leave all other default settings, then click **Create**.



Task 4. Run the pipeline

Deploy the Dataflow Template:

```
gcloud dataflow jobs run iotflow
```

```
--gcs-location gs://dataflow-templates-"Region"/latest/PubSub_to_BigQuery
```

```
--region "Region"
```

```
--worker-machine-type e2-medium
```

```
--staging-location gs://"Bucket Name"/temp
```

```
--parameters inputTopic=projects/pubsub-public-data/topics/taxirides-  
realtime,outputTableSpec="Table Name":taxirides.realtime
```

```
student_01_fd827e106326@cloudshell:~ (qwiklabs-gcp-03-eed6c4ca7a3b)$ gcloud dataflow jobs run iotflow \
--gcs-location gs://dataflow-templates-us-west1/latest/PubSub_to_BigQuery \
--region us-west1 \
--worker-machine-type e2-medium \
--staging-location gs://qwiklabs-gcp-03-eed6c4ca7a3b/temp \
--parameters inputTopic=projects/pubsub-public-data/topics/taxirides-realtime,outputTableSpec=qwiklabs-gcp-03-eed6c4ca7a3b:taxirides.realtime
createTime: '2025-06-12T12:18:25.259840Z'
currentStateTime: '1970-01-01T00:00:00Z'
id: 2025-06-12_05_18_23-15267160792125284498
location: us-west1
name: iotflow
projectId: qwiklabs-gcp-03-eed6c4ca7a3b
startTime: '2025-06-12T12:18:25.259840Z'
type: JOB_TYPE_STREAMING
student_01_fd827e106326@cloudshell:~ (qwiklabs-gcp-03-eed6c4ca7a3b)$
```

In the **Google Cloud Console**, on the **Navigation menu**, click **Dataflow > Jobs**, and you will see your dataflow job.

Task 5. Submit a query

You can submit queries using standard SQL.

1. In the BigQuery **Editor**, add the following to query the data in your project:

```
SELECT * FROM ` "Bucket Name".taxirides.realtime` LIMIT 1000
```

2. Now click **RUN**.

Query results [SAVE AS](#) [EXPLORE IN DATA STUDIO](#)

Query complete (2.116 sec elapsed, 0 B processed)

Job information [Results](#) [JSON](#) [Execution details](#)

Row	ride_id	point_idx	latitude	longitude	timestamp
1	b0810fbd-78a8-4159-b9ff-963695e2a23d	225	40.753550000000004	-73.985040000000001	2018-07-25 23:28:20.870530 UTC
2	1a10dc8b-3623-41bf-938a-9fca26c2ae10	311	40.752930000000006	-73.96584	2018-07-25 23:24:10.608380 UTC
3	5253c100-1a30-4a3e-89ee-6c0c861cf44f	224	40.74331	-73.99172	2018-07-25 23:26:34.636480 UTC
4	3efa96c2-4695-4c0b-96b6-da33a4b74ccf	8	40.7533	-73.978320000000001	2018-07-25 23:24:06.823150 UTC
5	d6d37615-ccba-4416-9932-e956e0f0ba65	747	40.682140000000004	-74.005940000000001	2018-07-25 23:24:10.103770 UTC

Key Learnings

- **Dataflow Templates** provide a simplified way to deploy real-time data pipelines.
- **Pub/Sub** can act as a real-time data ingestion mechanism.
- **BigQuery** supports partitioned tables for efficient querying of streaming data.
- Command-line tools like gcloud, bq, and gsutil streamline infrastructure setup.

Conclusion

This lab demonstrated a full streaming data pipeline using native GCP services, showcasing Dataflow's capabilities in integrating Pub/Sub and BigQuery. Real-time ingestion and querying were verified, and the project setup adhered to best practices with managed resources and templates.

