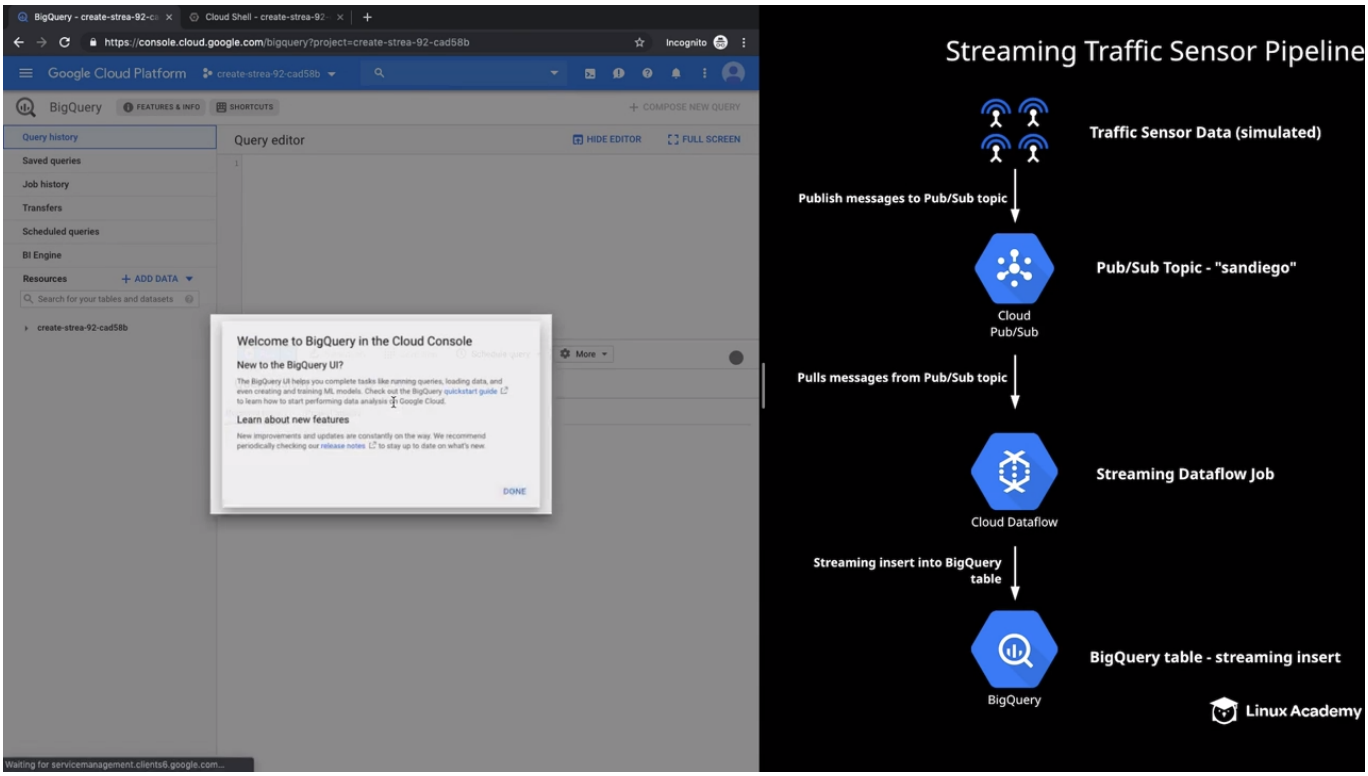
00:44:05 Exit Lab Complete Lab

Create a Streaming Data Pipeline on GCP with Cloud Pub/Sub, Dataflow, and BigQuery

Guided Mode 1 hour duration Professional Rate this lab

VIDEOS GUIDE



VIDEO 1 OF 1

Creating and Executing a Streaming Data Pipeline

Previous Next

This video will walk through the steps to create a Pub/Sub topic, Dataflow streaming job (via a script), and BigQuery dataset. We will then execute a Python script to simulate streaming traffic sensor data into your pipeline, and view the results in BigQuery.

Tools

Lab Diagram Instant Terminal

Credentials

How do I connect?

google labs Account

Username

cloud_user_p_293679...

Password

5ZLF/E9g

Open Link in Incognito Window

How do I connect?

Additional Resources

Many data engineer scenarios on GCP involve a multi-step streaming data pipeline from ingestion, to processing, to storage/analysis. In this lab, we will create a simulated end to end streaming pipeline of all steps, which will finish in analyzing captured streaming data for insights.

Be sure to launch the lab in your browser's incognito (or other private browsing) mode to avoid cached login issues.

Learning Objectives

2 of 8 completed

Optional: Run progress checks to confirm you've completed the objectives

Prepare your environment

Enable Pub/Sub and Dataflow APIs:

```
gcloud services enable dataflow.googleapis.com
gcloud services enable pubsub.googleapis.com
```

Create a Cloud Storage bucket for Dataflow staging:

```
gsutil mb gs://$DEVSHELL_PROJECT_ID
```

Download the GitHub repository used for lab resources:

```
cd ~
git clone https://github.com/linuxacademy/googledataengineer
```

Mark Incomplete

Create a pub/sub topic

```
gcloud pubsub topics create sandiego
```

Run Progress Check Skip

Last checked at 10:10PM

Create a bigquery dataset to stream data into

Create a BigQuery dataset to stream data into:

```
bq mk --dataset $DEVSHELL_PROJECT_ID:demos
```

The table will be named `average_speeds`. We do not create the table, but Dataflow will create it within the dataset for us.

Run Progress Check Skip

View the dataflow template

We will not be interacting with the template directly. We will be using a script that will install the Java environment and execute the template as a Dataflow job:

```
vim
googledataengineer/courses/streaming/process/sandiego/src/main/java/com/google/cloud/training/dataanalyst/sandiego/AverageSpeeds.java
```

Mark Complete

Create the dataflow streaming job

Go to the Dataflow job script directory:

```
cd ~/googledataengineer/courses/streaming/process/sandiego
```

Execute the script that creates the Dataflow streaming job, and subscribe to the Pub/Sub topic.

This script passes along the Project ID, staging bucket (also the Project ID), and the name of the Java template to use:

```
./run_oncloud.sh $DEVSHELL_PROJECT_ID $DEVSHELL_PROJECT_ID
AverageSpeeds
```

When complete, the streaming job will be subscribed to our Pub/Sub topic, and waiting for streaming input from our simulated sensor data.

Mark Complete

Publish simulated traffic sensor data to pub/sub via a python script and pre-created dataset

Browse to the Python script directory:

```
cd ~/googledataengineer/courses/streaming/publish
```

Install any requirements for the Python script:

```
sudo pip install -U google-cloud-pubsub
```

Download the simulated sensor data:

```
gsutil cp gs://acg-gcloud-course-
resources/sandiego/sensor_obs2008.csv.gz .
```

Execute the Python script to publish simulated streaming data to Pub/Sub:

```
./send_sensor_data.py --speedFactor=60 --
project=$DEVSHELL_PROJECT_ID
```

Mark Complete

View the streamed data in bigquery

In BigQuery, execute the following query to view the current streamed data, both in the table and in the streaming buffer:

```
SELECT *
FROM `demos.average_speeds`
```

Notice the total count of records at the bottom. Wait about a minute and run the same query again (be sure to uncheck **use cached results** in query options) and notice that the number has increased.

Mark Complete

Use aggregated queries to gain insights

Let's get some use out of this data. If we wanted to forecast some necessary road maintenance, we would need to know which lanes have the most traffic, to know which ones will require resurfacing first.

Enter the following query to view which lanes have the most sensor counts:

```
SELECT lane, count(lane) as total
FROM `demos.average_speeds`
GROUP BY lane
ORDER BY total DESC
```

We can also view which lanes have the highest average speeds:

```
SELECT lane, avg(speed) as average_speed
FROM `demos.average_speeds`
GROUP BY lane
ORDER BY average_speed DESC
```

Mark Complete