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

Introduction

This lab will simulate live highway sensor data which will be published to a Cloud Pub/Sub topic. Then, a Cloud Dataflow streaming pipeline will subscribe to it. The pipeline will take the streaming sensor data, transform it, and insert it into a BigQuery table. We will then view the streaming inserts in BigQuery while they are in progress, and attempt to gain some useful insights from the streaming data.

Solution

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.

How to Log in to Google Lab Accounts

On the lab page, right-click **Open GCP Console** and select the option to open it in a new private browser window. This option will read differently depending on your browser. In Chrome it says "Open Link in Incognito Window". In Firefox it says "Open link in new private window." In Microsoft Edge, the message will be "Open in InPrivate window." And in Safari, press **Alt** or **Option**, then right click to get a menu where you will choose "Open link in new private window."

This will avoid any cached login issues. Once you're at the login screen, sign into Google Cloud Platform using the credentials provided on the lab page.

On the *Welcome to your new account* screen, review the text, and click **Accept**. In the "Welcome L.A.!" that pops up once you're signed in, check to agree to the terms of service, choose your country of residence, and click **Agree and Continue**.

Prepare Your Environment

Most of the setting up we do will be in the Cloud Shell. So let's go ahead and click on the **Activate Cloud Shell** button, then click the blue **START CLOUD SHELL** button in the window that pops up. This will fire up a shell. We'll be switching back and forth between the shell and the web console, as we make changes and verify that they've taken effect.

First up, we need to enable the 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:

**(cloud storage need to be global unique, we need the naming schema of the DEVSHELL project ID shell variable, when we use it, we automatically resolve to whatever your individual project ID is, so our bucket name stay unique)**

gsutil mb gs://$DEVSHELL\_PROJECT\_ID

Download the GitHub repository used for lab resources:

cd ~

git clone https://github.com/linuxacademy/googledataengineer

Now that these preliminary steps are out of the way, we can actually sit down and start building a pipeline.

Create Pub/Sub Topic

Here we'll create our topic, and name it sandiego:

gcloud pubsub topics create sandiego

Verify in the Web Console

Over in the web console, let's make sure things got created. In the top-left menu, scroll down to **BIG DATA**, and then click on **Pub/Sub**. We'll see our topic in there.

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.

Verify in the Web Console

Back over in the web console, check that the process went well by going to the top-left menu, then **BIG DATA**, then click **BigQuery**. We should see our project name, with a dataset.

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. TO peek at it, run:

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

Pressing the **Esc** key, then typing **:q!** (that's colon, q, and exclamation point) will get us out of the file without making any changes to it.

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**.

Verify in the Web Console

Let's prove that it was set up though. Back in the web console, head to the top-left menu, then **BIG DATA**, then click **DataFlow**. Once we landed there, we'll see our job, just waiting there for input.

And if we navigate back to **Pub/Sub**, then click on **Subscriptions**, we can see that it's now subscribed to our topic.

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://la-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

View the Streamed Data in BigQuery

Now that the data is done streaming, we can take a look at it. Get into **DataFlow** again, then click on our streaming job. In here, we can see all of the steps that our data is going through, and that it's getting inserted into a database table. Let's look at it.

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

SELECT \*

FROM `database\_name.demos.average\_speeds LIMIT 1000`

Remember that the database name in the query will be different.

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.

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, sum(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

Conclusion

We took some raw data (what was coming in from traffic sensors on a highway) and were able to build a data pipeline that gives us information about traffic. From that pipeline, we can pull average speeds of each lane, show which lanes are more heavily trafficked, and all sorts of other handy data that might help in decision making down the road. Congratulations!