DataEng S24: PubSub

[this lab activity references tutorials at cloud.google.com]

Make a copy of this document and use it to record your results. Store a PDF copy of the document in your git repository along with your code before submitting for this week. For your code, you create several publisher/receiver programs or you might make various features within one program. There is no one single correct way to do it. Regardless, store your code in your repository.

The goal for this week is to gain experience and knowledge of using an asynchronous data transport system (Google PubSub). Complete as many of the following exercises as you can. Proceed at a pace that allows you to learn and understand the use of PubSub with python.

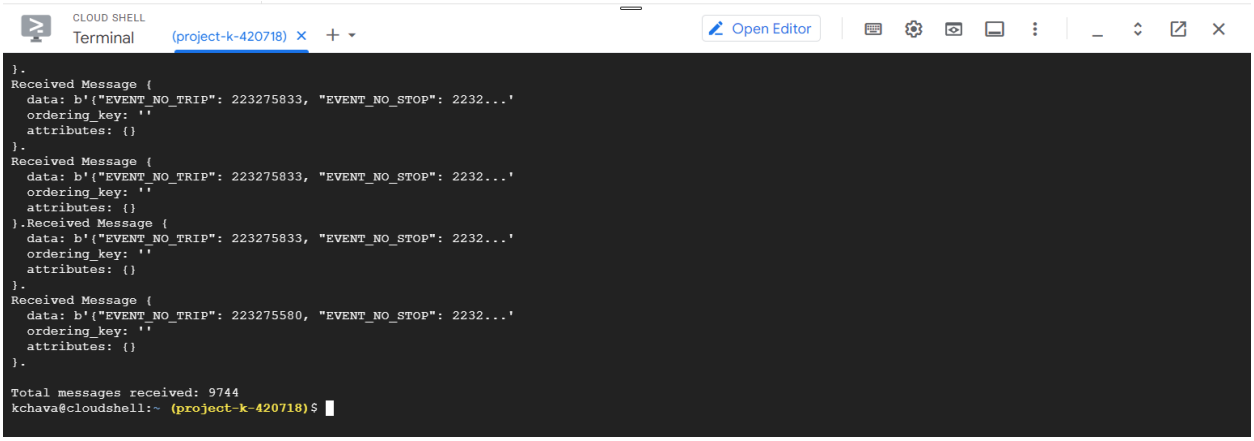
Submit: use the in-class activity submission form which is linked from the Materials page on the class website. Submit by 10pm PT this Friday.

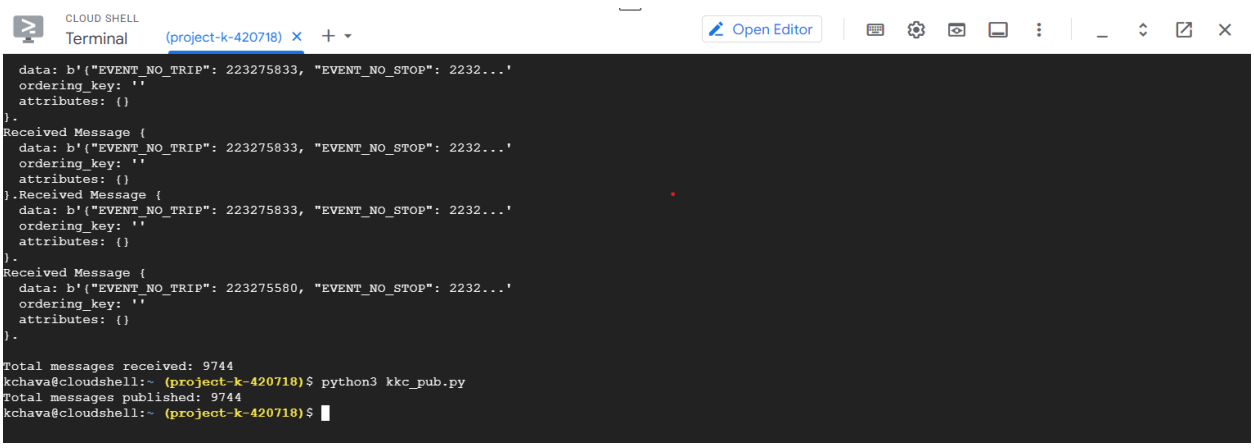
A. [MUST] PubSub Tutorial

1. Get your cloud.google.com account up and running
    a. Redeem your GCP coupon
    b. Login to your GCP console
    c. Create a new, separate VM instance
2. Complete this PubSub tutorial: link Note that the tutorial instructs you to destroy your PubSub topic, but you should not destroy your topic just yet. Destroy the topic after you finish the following parts of this in-class assignment.

B. [MUST] Create Sample Data

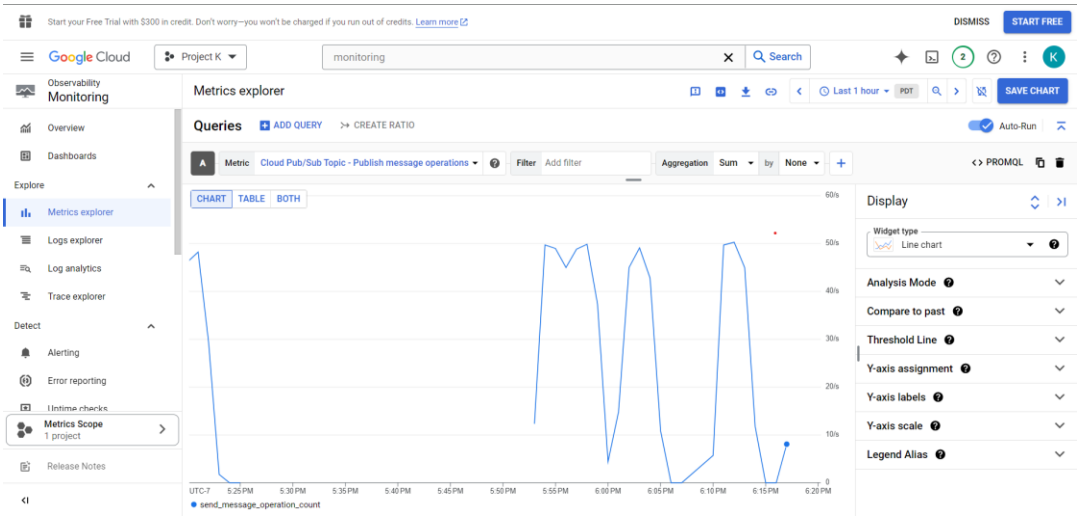
1. Get data from https://busdata.cs.pdx.edu/api/getBreadCrumbs for two Vehicle IDs from among those that have been assigned to you for the class project.
2. Save this data in a sample file (named bcsample.json)
3. Update the publisher python program that you created in the PubSub tutorial to read and parse your bcsample.json file and send its contents, one record at a time, to the my-topic PubSub topic that you created for the tutorial.
4. Use your receiver python program (from the tutorial) to consume your records.





C. [MUST] PubSub Monitoring

1. Review the PubSub Monitoring tutorial: link and work through the steps listed there. You might need to rerun your publisher and receiver programs multiple times to trigger enough activity to monitor your my-topic effectively.



D. [MUST] PubSub Storage

1. What happens if you run your receiver multiple times while only running the publisher once?
   Answer: If you publish a message once and run several receivers, only the first receiver to get the message will acknowledge it, which means it then gets removed from the queue. Any other receivers that are running won't receive that same message. They will keep waiting until their set timeout ends and then they will stop if no other messages come through.

2. Before the consumer runs, where might the data go, where might it be stored?
   Answer: Before the consumer starts, the data is stored in the "message storage" area of the Cloud Pub/Sub service.

3. Is there a way to determine how much data PubSub is storing for your topic? Do the PubSub monitoring tools help with this?
   Answer: Pub/Sub is designed mainly for passing messages rather than storing them for a long time, so it doesn't provide a direct way to measure how much storage a topic is using. However, you can estimate this by counting the undelivered messages and knowing the average size of your messages. To get a rough idea of the storage being used, multiply the number of messages waiting to be delivered by their average size.

4. Create a “topic_clean.py” receiver program that reads and discards all records for a given topic. This type of program can be very useful for debugging your project code.

E. [SHOULD] Multiple Publishers

1. Clear all data from the topic (run your topic_clean.py program whenever you need to clear your topic)
2. Run two versions of your publisher concurrently, have each of them send all of your sample records. When finished, run your receiver once. Describe the results.
   Answer: When I ran the Publisher Version 1, it Published 9744 and for another version it published 6377 and when I ran the receiver, it consumed 16121 messages.

F. [SHOULD] Multiple Concurrent Publishers and Receivers

F. [ASPIRE] Multiple Subscriptions

1. So far your receivers have all been competing with each other for data. Next, create a new subscription for each receiver so that each one receives a full copy of the data sent by the publisher. Parameterize your receiver so that you can specify a separate subscription for each receiver.
2. Rerun the multiple concurrent publishers/receivers test from the previous section. Assign each receiver to its own subscription.
3. Describe the results.