DataEng S24: PubSub

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.

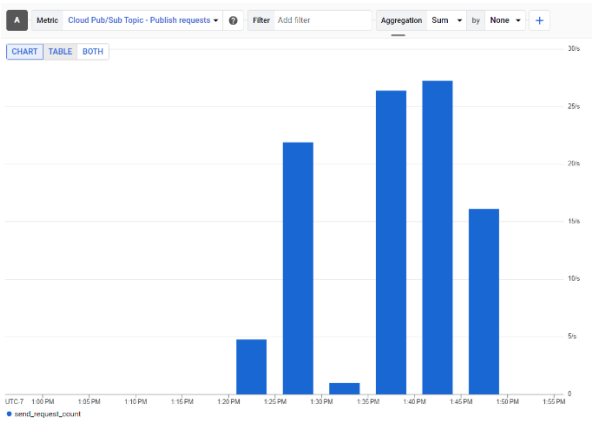
B. [MUST] Create Sample Data

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.**

Ans: I have checked the metrics for different criteria in topic and subscriber. Here is one example of published messages.



D. [MUST] PubSub Storage

1. **What happens if you run your receiver multiple times while only running the publisher once?**

Ans: When I ran my receiver code multiple times after running publisher once, In the first run the subscriber processed and acknowledged all the messages and then those messages are removed from the queue and when I run the code for the multiple times no messages were left to process, as they were acknowledged and removed from the queue, and I got the count as zero.

```
aashritk@dataengineering:~$ source myenv/bin/activate
(myenv) aashritk@dataengineering:~$ python3 subscriber.py
Listening for messages on projects/dataengineering-420322/subscriptions/my-sub...

Total Number of messages received are: 0
```

2. **Before the consumer runs, where might the data go, where might it be stored?**

Ans: Before the consumer runs, publisher first send messages to the topic and are temporarily stored in message storage, which is managed by the cloud's Pub/Sub's. Those messages remain in the message storage until it is acknowledged by a subscriber (Consumer runs).

3. **Is there a way to determine how much data PubSub is storing for your topic? Do the PubSub monitoring tools help with this?**

Ans: Google cloud monitoring doesn't give the exact storage volume for topics, instead it gives information about message count and 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 your sample records. When finished, run your receiver once. Describe the results.**

Ans: When I ran publisher for the first time, a total of 9403 messages got published and after running the publisher code for the second time by changing the vehicle id's I got count of 4088 messages. When I ran my receiver code a total of 11464 messages got acknowledged from the message storage for the timeout of 40seconds that I have mentioned in the code.

F. [SHOULD] Multiple Concurrent Publishers and Receivers

1. Clear all data from the topic.

2. Update your publisher code to include a 250 msec sleep after each send of a message to the topic.
3. Run two or three concurrent publishers and two concurrent receivers all at the same time. Have your receivers redirect their output to separate files so that you can sort out the results more easily.
4. Describe the results.

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.