BIG DATA LAB ASSIGNMENT FOUR (LAB6)

Devansh Sanghvi
Department Of Biotechnology
Indian Institute of Technology, Madras
Chennai, India
be19b002@smail.iitm.ac.in

***Abstract*—This document aims to answer the two questions in Assignment Four of the course: Big Data Lab. The python scripts used in the submission can be found at Submission Files**

IMPORTANT NOTE: All outputs in this assignment will have the ID: **sanghvidevansh23** instead of Roll No. This is because I have signed into Google Cloud Console using my email ID.

I. Question One

Question: In this assignment, you will count the number of lines in a file uploaded to the GCS bucket in real time using Google Cloud Functions and Pub/Sub.

1) Download the file from here: https://filesamples.com/samples/document/txt/sample1.txt
2) Write a Google cloud Function that gets triggered whenever a file is added to a bucket and publishes the file name to a topic in Pub/Sub.
3) Write a python file, which acts as a subscriber to this topic and prints out the number of lines in the file in real-time

Solution

1) Start the cloud shell. Make a new bucket and directory for the Pub/Sub function and subscriptions.
2) The topic and subscription were created using the names 'pubsub-lab6-be19b002' and 'linecount-sub-final3'. The commands used to create these are as shown below:

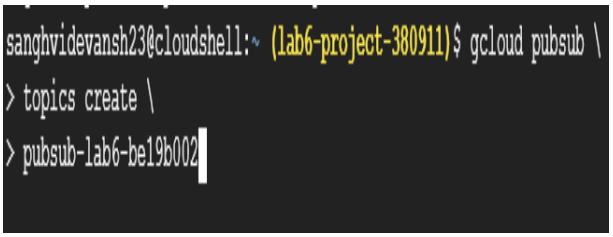

Fig. 1. Created a topic for the assignment

3) Next, the google cloud function is created, which gets triggered when a file is added to the bucket. The main.py and requirements.txt files are in the folder.
4) "data" and "context" are the inputs to the GCF. "data" is a dictionary which contains the name of the bucket, the name of the file that was added to the bucket, etc. "context" acts like metadata of the event. When triggered, the GCF publishes the file name added into the bucket into the topic "pubsub-lab6-be19b002". The GCF is deployed as shown below:

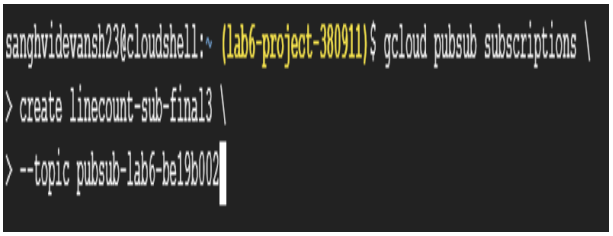

Fig. 2. Created a subscription for the assignment

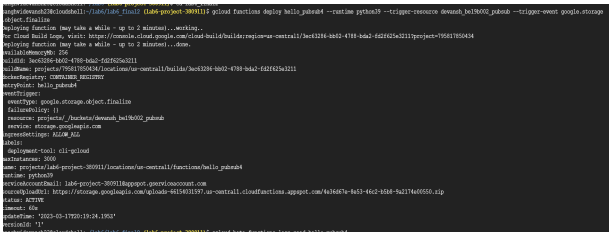

Fig. 3. Deploy function for the task

5) The text file is downloaded and saved as "a6_lab_text.txt". It is then uploaded to the google cloud shell and copied to the bucket "devansh_be19b002_pubsub".

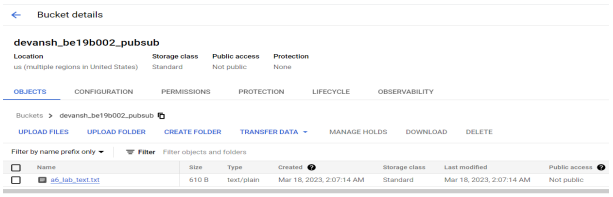

Fig. 4. Upload the text file to test if the name was published in the topic

6) You can view the logs of the function to confirm that the function is working properly. The result can be seen below. The file name is published to the topic pubsub-lab6-be19b002, as shown in the logs.
7) The subscription was created in Step 2). It is a pull subscription by default. Next, we create the subscription python file, "subs.py," which subscribes to "pubsub-

Fig. 5. Logs of the function after a file is added

lab6-be19b002" and prints the number of lines in the file added to the bucket.

8) The callback function is an important part of "subs.py." The input to this function is the message the subscriber is listening to. This is the function that processes the message.
9) To enable real-time printing of several lines, the subscriber is made to listen to messages in "pubsub-lab6-be19b002" indefinitely.
10) After triggering the GCF, the file name is published to "pubsub-lab6-be19b002". The subscriber listens to this message in real time and prints the number of lines.

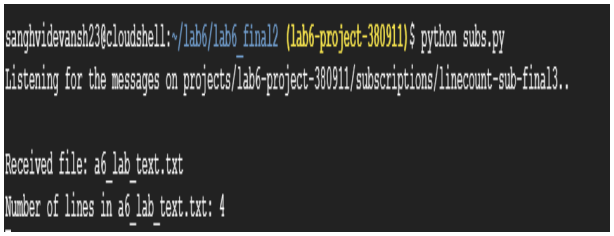

Fig. 6. Number of lines in the added file were printed using the subscription

11) To stop the subscriber from listening, we should interrupt the code using the keyboard.

II. Question Two

Question: There are two kinds of subscribers - pull and push subscribers. What are the differences, and when would you prefer one?

Solution

1) **Pull Subscription** A "pull" subscriber requests data from the source when it is ready to receive it. In other words, the receiver actively seeks out data from the source, typically by sending a request or query.
   Push Subscription A "push" subscriber, on the other hand, receives data as soon as it is available without explicitly requesting it. In this case, the source actively sends data to the receiver as soon as it is generated.
2) **Pull Subscription** Pull subscribers are useful when the receiver has control over when it wants to receive data and can handle latency in the data delivery process.
   Push Subscription Push subscribers are useful when the receiver needs to receive data in real-time or as close to real-time as possible.
3) **Pull Subscription** Multiple subscribers can access the same shared pull subscription, where each subscriber will receive a subset of the messages.
   Push Subscription The push endpoint balances the server's load to accommodate more topics.

The differences, according to use cases can be summarized as:

1) For large volume of messages pull subscribers are preferred
2) If multiple topics are to be processed, push subscribers are preferred
3) If the topics are very busy, it is better to use push subscribers because it is difficult to send out pull requests along with the processing of the images.
4) Pull subscribers have higher throughput than push subscribers.