CS 6381 Final Project Google Cloud Pub Sub Setup:

Plan:

Have the publisher send messages continuously to the broker via ZMQ, which send messages continuously to the subscriber via Google Cloud, and when the subscriber receives a specific number of messages, it creates a CSV file which saves in the timeMeasurements folder and terminates. Repeat this process for different text-sized messages in the messages folder.

Procedure:

1. Set up Google Cloud Pub Sub
   1. Set up Pub Sub (<https://cloud.google.com/pubsub/docs/building-pubsub-messaging-system>, <https://www.youtube.com/watch?v=UKAmZBrR300&amp;loop=0>)
      1. Spin up a Ubuntu 18.04 VM.
      2. Install Google Cloud SDK on Ubuntu: <https://cloud.google.com/sdk/docs/install>
      3. Put `` like where it shows export PROJECT=`gcloud config get-value project`
      4. Rename your JSON key to something simple like key.json and put it somewhere simple to access like in the Downloads folder with respect to the root folder.
2. To run code each time:
   1. Automatically
      1. Navigate to the tests folder and run a test using “./{{test name}}” (e. g. “./1producer1consumerTest.sh”). Wait until the test finishes (i. e. when all extra terminal tabs have closed [the last one takes a little while to show up]) and then check for results in the timeMeasurements folder.
   2. Manually
      1. Open 3 separate terminals or tabs and run each of the following separately in this order, applying the “Connect to Google Cloud Project” step as follows at the start of all the subsequent steps (sub.py, pub.py, broker.py).
         1. Connect to Google Cloud project.
            1. export GOOGLE\_APPLICATION\_CREDENTIALS=~/Downloads/key.json
            2. export PROJECT=`gcloud config get-value project`
            3. echo $PROJECT (make sure project ID shows up)
         2. Run sub.py:
            1. “python3 sub.py $PROJECT {{Google Cloud-created subscription name (e. g. sub\_one)}}”
         3. Run broker.py:
            1. “python3 broker.py $PROJECT {{Google Cloud-created topic name (e. g. hello\_topic}}”
         4. Run pub.py:
            1. "python3 pub.py {{file name}}"
      2. To stop the code properly using Ctrl + C:
         1. Stop broker.py 1st or else sub.py will receive old messages.
         2. Empty out old messages by 1 of the following ways:
            1. Run “python3 clear.py”.
            2. Start sub.py and wait until all old messages are emptied out and then stop it before running the code again.
3. Run calculateStatistics.py to calculate statistics of time measurements.
4. You can measure hardware performance continuously by running hardwareMeasure.py, which will print the latest information.
5. You can generate a new message file that holds a specific number of random characters by running messageGenerator.py and specifying that number with a command line argument.

Conditions to Test:

* 1 producer vs. 1 consumer
* 1 producer vs. 5 consumers
* 5 producers vs. 1 consumer
* 3 x (8 producers vs. 8 consumers)
* 4 x (25 producers vs. 25 consumers)