DataEng S24: PubSub

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

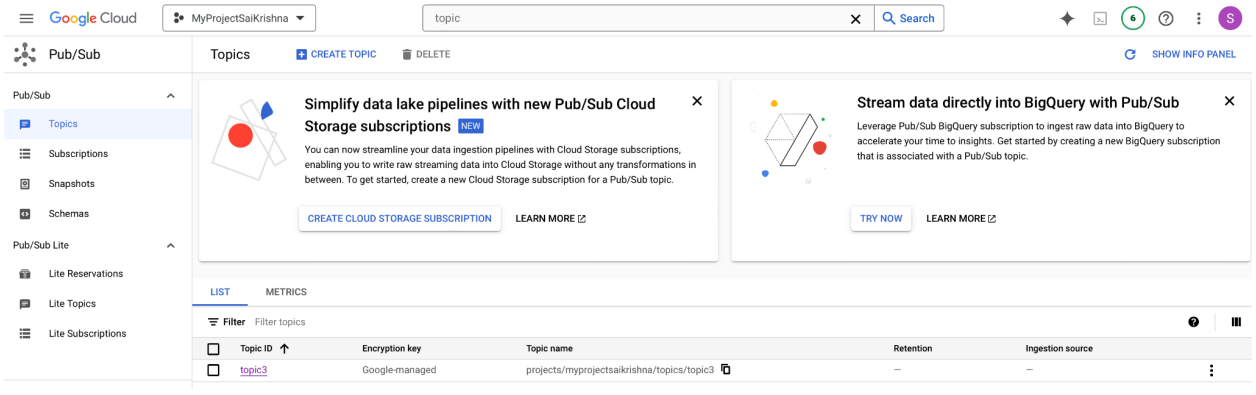
Make a copy of this document and use it to record your results. Store a PDF copy of the document in your git repository along with your code before submitting for this week. For your code, you create several publisher/receiver programs or you might make various features within one program. There is no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

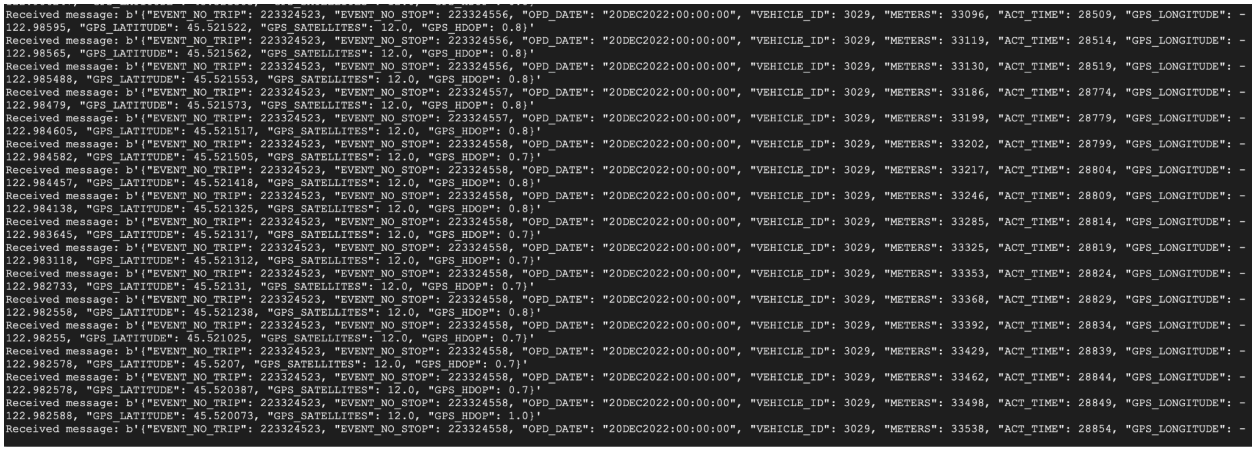
1. Get your cloud.google.com account up and running
    a. Redeem your GCP coupon
    b. Login to your GCP console
    c. Create a new, separate VM instance
2. Complete this PubSub tutorial: link Note that the tutorial instructs you to destroy your PubSub topic, but you should not destroy your topic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



B. [MUST] Create Sample Data

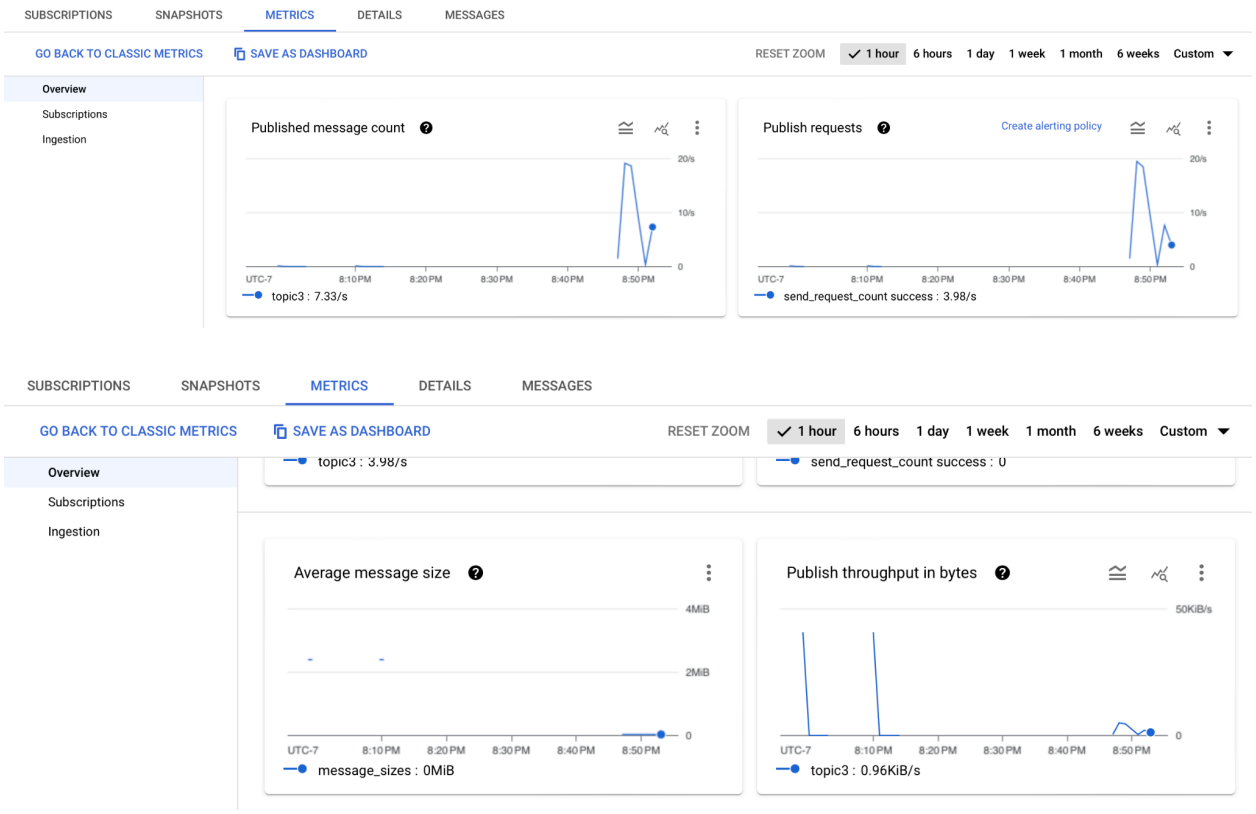
1. Get data from https://busdata.cs.pdx.edu/api/getBreadCrumbs for two Vehicle IDs from among those that have been assigned to you for the class project.
2. Save this data in a sample file (named bcsample.json)
3. Update the publisher python program that you created in the PubSub tutorial to read and 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.

```
smukund@cloudshell:~ (myprojectsaikrishna)$ python publish.py
Data saved to bcsample.json
Published message. ID: 10949518260964006
Published message. ID: 10948764900307675
Published message. ID: 10949621215566246
Published message. ID: 10949061165652854
Published message. ID: 10949525220321080
Published message. ID: 10949436569740899
Published message. ID: 10948828353946307
Published message. ID: 10948822069922426
Published message. ID: 10948782504092109
Published message. ID: 10949567095673033
Published message. ID: 10949668010543661
Published message. ID: 10949289690998125
Published message. ID: 10949563186778209
Published message. ID: 10949517853326709
```



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:
Once a message is sent and acknowledged, only one recipient will receive it if the subscriber is started more than once. Every message is delivered at least once thanks to pub/sub. This implies that a message will be delivered repeatedly until the recipient acknowledges it if it hasn't been acknowledged.

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

Answer:
Messages sit in Pub/Sub topics until the user starts. They remain there until they are acknowledged or they pass away. In order to ensure stability, Google Cloud's Pub/Sub stores messages in a distributed system rather than on disks. Pub/Sub scales automatically, efficiently spreading data among server clusters. Pub/Sub distributes messages equally for speedy

processing when subscribers fetch them.Pub/Sub offers dependable messaging infrastructure by effectively handling high message volumes.

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:
Google Cloud Pub/Sub has tools to watch what's going on, and these are now part of Google Cloud Operations Suite. With Cloud Monitoring, we can see stuff like messages that haven't been read yet, how long messages stick around, and how big they are. This helps us figure out how much data Pub/Sub is holding onto for our topics and subscriptions.

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.

Answer:
We can read and acknowledge messages in the topic's context with these simple programs, and we can execute it to remove messages from the topic if necessary. As an alternative, we can choose to "Remove Messages" from the Console UI in order to remove unacknowledged messages.

E. [SHOULD] Multiple Publishers

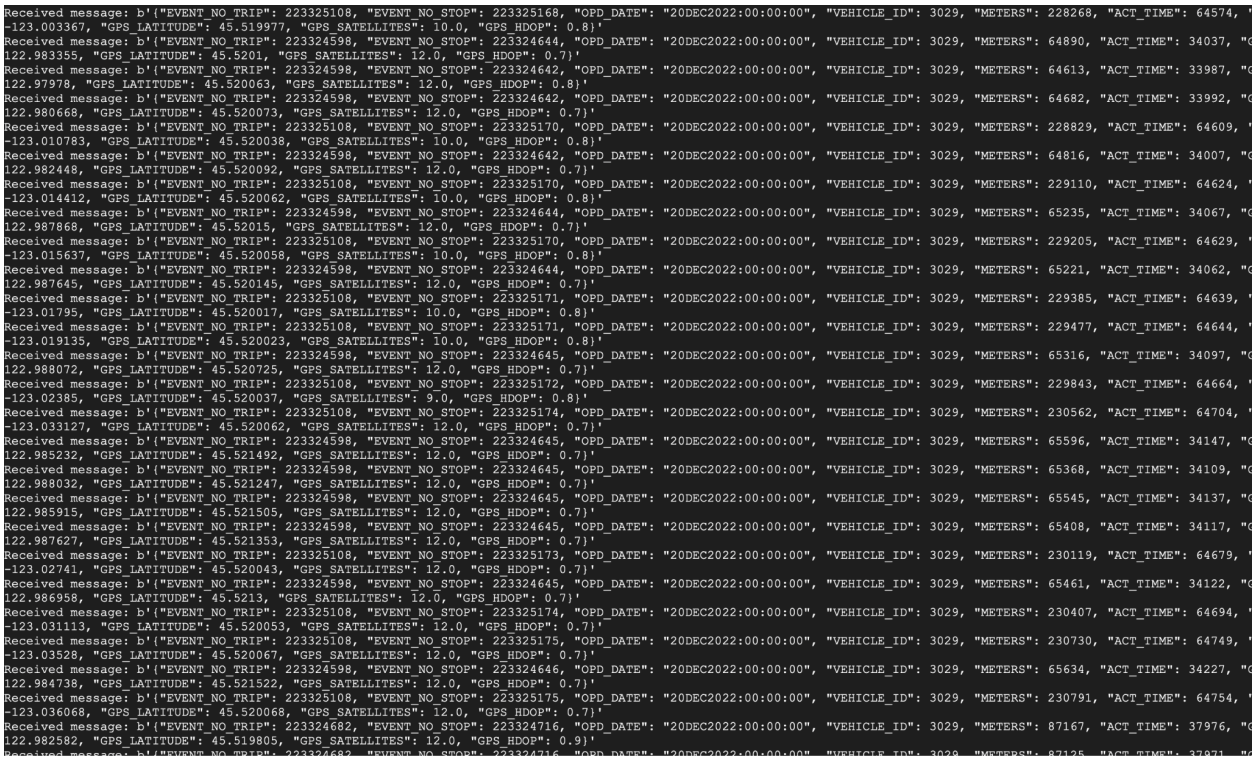
1. Clear all data from the topic (run your topic_clean.py program whenever you need to clear your topic)

```
^X^Csmukund@cloudshell:~ (myprojectsaikrishna)$ python subscriber.py
Listening for messages on projects/myprojectsaikrishna/subscriptions/sub6...
smukund@cloudshell:~ (myprojectsaikrishna)$ nano publish.py
smukund@cloudshell:~ (myprojectsaikrishna)$ nano subscriber.py
smukund@cloudshell:~ (myprojectsaikrishna)$ nano topic_clean.py
smukund@cloudshell:~ (myprojectsaikrishna)$ nano topic_clean.py
smukund@cloudshell:~ (myprojectsaikrishna)$ python topic_clean.py
Listening for messages on subscription projects/myprojectsaikrishna/subscriptions/sub6...
```

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.

```
smukund@cloudshell:~ (myprojectsaikrishna)$ python publish.py
Data saved to bcsample.json
Published message. ID: 9475798092831751
Published message. ID: 10949921509474662
Published message. ID: 10949603818279196
Published message. ID: 10949320337038559
Published message. ID: 10949849372990569
Published message. ID: 10948869642686418
Published message. ID: 10949589704944559
```

```
^X^Csmukund@cloudshell:~ (myprojectsaikrishnpython publish.py
Data saved to bcsample.json
Published message. ID: 10949788744443574
Published message. ID: 10949845702515857
Published message. ID: 10949818580405565
Published message. ID: 10949318648767279
Published message. ID: 10949683268363401
Published message. ID: 10949668068828228
Published message. ID: 10949731271681166
```



```
Received message: b'{"EVENT_NO_TRIP": 223325108, "EVENT_NO_STOP": 223325168, "OPD_DATE": "20DEC2022:00:00:00", "VEHICLE_ID": 3029, "METERS": 228268, "ACT_TIME": 64574, "
-123.003367, "GPS_LATITUDE": 45.519977, "GPS_SATELLITES": 10.0, "GPS_HDOP": 0.8}'
Received message: b'{"EVENT_NO_TRIP": 223324598, "EVENT_NO_STOP": 223324644, "OPD_DATE": "20DEC2022:00:00:00", "VEHICLE_ID": 3029, "METERS": 64890, "ACT_TIME": 34037, "
122.983355, "GPS_LATITUDE": 45.5201, "GPS_SATELLITES": 12.0, "GPS_HDOP": 0.7}'
Received message: b'{"EVENT_NO_TRIP": 223324598, "EVENT_NO_STOP": 223324642, "OPD_DATE": "20DEC2022:00:00:00", "VEHICLE_ID": 3029, "METERS": 64613, "ACT_TIME": 33987, "
122.97978, "GPS_LATITUDE": 45.520063, "GPS_SATELLITES": 12.0, "GPS_HDOP": 0.8}'
Received message: b'{"EVENT_NO_TRIP": 223324598, "EVENT_NO_STOP": 223324642, "OPD_DATE": "20DEC2022:00:00:00", "VEHICLE_ID": 3029, "METERS": 64682, "ACT_TIME": 33992, "
122.980668, "GPS_LATITUDE": 45.520073, "GPS_SATELLITES": 12.0, "GPS_HDOP": 0.7}'
Received message: b'{"EVENT_NO_TRIP": 223325108, "EVENT_NO_STOP": 223325170, "OPD_DATE": "20DEC2022:00:00:00", "VEHICLE_ID": 3029, "METERS": 228829, "ACT_TIME": 64609, "
-123.010783, "GPS_LATITUDE": 45.520038, "GPS_SATELLITES": 10.0, "GPS_HDOP": 0.8}'
Received message: b'{"EVENT_NO_TRIP": 223324598, "EVENT_NO_STOP": 223324642, "OPD_DATE": "20DEC2022:00:00:00", "VEHICLE_ID": 3029, "METERS": 64816, "ACT_TIME": 34007, "
122.982448, "GPS_LATITUDE": 45.520092, "GPS_SATELLITES": 12.0, "GPS_HDOP": 0.7}'
Received message: b'{"EVENT_NO_TRIP": 223325108, "EVENT_NO_STOP": 223325170, "OPD_DATE": "20DEC2022:00:00:00", "VEHICLE_ID": 3029, "METERS": 229110, "ACT_TIME": 64624, "
-123.014412, "GPS_LATITUDE": 45.520062, "GPS_SATELLITES": 10.0, "GPS_HDOP": 0.8}'
Received message: b'{"EVENT_NO_TRIP": 223324598, "EVENT_NO_STOP": 223324644, "OPD_DATE": "20DEC2022:00:00:00", "VEHICLE_ID": 3029, "METERS": 65235, "ACT_TIME": 34067, "
122.987868, "GPS_LATITUDE": 45.52015, "GPS_SATELLITES": 12.0, "GPS_HDOP": 0.7}'
Received message: b'{"EVENT_NO_TRIP": 223325108, "EVENT_NO_STOP": 223325170, "OPD_DATE": "20DEC2022:00:00:00", "VEHICLE_ID": 3029, "METERS": 229205, "ACT_TIME": 64629, "
-123.015637, "GPS_LATITUDE": 45.520058, "GPS_SATELLITES": 10.0, "GPS_HDOP": 0.8}'
Received message: b'{"EVENT_NO_TRIP": 223324598, "EVENT_NO_STOP": 223324644, "OPD_DATE": "20DEC2022:00:00:00", "VEHICLE_ID": 3029, "METERS": 65221, "ACT_TIME": 34062, "
122.987645, "GPS_LATITUDE": 45.520145, "GPS_SATELLITES": 12.0, "GPS_HDOP": 0.7}'
Received message: b'{"EVENT_NO_TRIP": 223325108, "EVENT_NO_STOP": 223325171, "OPD_DATE": "20DEC2022:00:00:00", "VEHICLE_ID": 3029, "METERS": 229385, "ACT_TIME": 64639, "
-123.01795, "GPS_LATITUDE": 45.520017, "GPS_SATELLITES": 10.0, "GPS_HDOP": 0.8}'
Received message: b'{"EVENT_NO_TRIP": 223325108, "EVENT_NO_STOP": 223325171, "OPD_DATE": "20DEC2022:00:00:00", "VEHICLE_ID": 3029, "METERS": 229477, "ACT_TIME": 64644, "
-123.019135, "GPS_LATITUDE": 45.520023, "GPS_SATELLITES": 10.0, "GPS_HDOP": 0.8}'
Received message: b'{"EVENT_NO_TRIP": 223324598, "EVENT_NO_STOP": 223324645, "OPD_DATE": "20DEC2022:00:00:00", "VEHICLE_ID": 3029, "METERS": 65316, "ACT_TIME": 34097, "
122.988072, "GPS_LATITUDE": 45.520725, "GPS_SATELLITES": 12.0, "GPS_HDOP": 0.7}'
Received message: b'{"EVENT_NO_TRIP": 223325108, "EVENT_NO_STOP": 223325172, "OPD_DATE": "20DEC2022:00:00:00", "VEHICLE_ID": 3029, "METERS": 229843, "ACT_TIME": 64664, "
-123.02385, "GPS_LATITUDE": 45.520037, "GPS_SATELLITES": 9.0, "GPS_HDOP": 0.8}'
Received message: b'{"EVENT_NO_TRIP": 223325108, "EVENT_NO_STOP": 223325174, "OPD_DATE": "20DEC2022:00:00:00", "VEHICLE_ID": 3029, "METERS": 230562, "ACT_TIME": 64704, "
-123.033127, "GPS_LATITUDE": 45.520062, "GPS_SATELLITES": 12.0, "GPS_HDOP": 0.7}'
Received message: b'{"EVENT_NO_TRIP": 223324598, "EVENT_NO_STOP": 223324645, "OPD_DATE": "20DEC2022:00:00:00", "VEHICLE_ID": 3029, "METERS": 65596, "ACT_TIME": 34147, "
122.985232, "GPS_LATITUDE": 45.521492, "GPS_SATELLITES": 12.0, "GPS_HDOP": 0.7}'
Received message: b'{"EVENT_NO_TRIP": 223324598, "EVENT_NO_STOP": 223324645, "OPD_DATE": "20DEC2022:00:00:00", "VEHICLE_ID": 3029, "METERS": 65368, "ACT_TIME": 34109, "
122.988032, "GPS_LATITUDE": 45.521247, "GPS_SATELLITES": 12.0, "GPS_HDOP": 0.7}'
Received message: b'{"EVENT_NO_TRIP": 223324598, "EVENT_NO_STOP": 223324645, "OPD_DATE": "20DEC2022:00:00:00", "VEHICLE_ID": 3029, "METERS": 65545, "ACT_TIME": 34137, "
122.985915, "GPS_LATITUDE": 45.521505, "GPS_SATELLITES": 12.0, "GPS_HDOP": 0.7}'
Received message: b'{"EVENT_NO_TRIP": 223324598, "EVENT_NO_STOP": 223324645, "OPD_DATE": "20DEC2022:00:00:00", "VEHICLE_ID": 3029, "METERS": 65408, "ACT_TIME": 34117, "
122.987627, "GPS_LATITUDE": 45.521353, "GPS_SATELLITES": 12.0, "GPS_HDOP": 0.7}'
Received message: b'{"EVENT_NO_TRIP": 223325108, "EVENT_NO_STOP": 223325173, "OPD_DATE": "20DEC2022:00:00:00", "VEHICLE_ID": 3029, "METERS": 230119, "ACT_TIME": 64679, "
-123.02741, "GPS_LATITUDE": 45.520043, "GPS_SATELLITES": 12.0, "GPS_HDOP": 0.7}'
Received message: b'{"EVENT_NO_TRIP": 223324598, "EVENT_NO_STOP": 223324645, "OPD_DATE": "20DEC2022:00:00:00", "VEHICLE_ID": 3029, "METERS": 65461, "ACT_TIME": 34122, "
122.986958, "GPS_LATITUDE": 45.5213, "GPS_SATELLITES": 12.0, "GPS_HDOP": 0.7}'
Received message: b'{"EVENT_NO_TRIP": 223325108, "EVENT_NO_STOP": 223325174, "OPD_DATE": "20DEC2022:00:00:00", "VEHICLE_ID": 3029, "METERS": 230407, "ACT_TIME": 64694, "
-123.031113, "GPS_LATITUDE": 45.520053, "GPS_SATELLITES": 12.0, "GPS_HDOP": 0.7}'
Received message: b'{"EVENT_NO_TRIP": 223325108, "EVENT_NO_STOP": 223325175, "OPD_DATE": "20DEC2022:00:00:00", "VEHICLE_ID": 3029, "METERS": 230730, "ACT_TIME": 64749, "
-123.03528, "GPS_LATITUDE": 45.520067, "GPS_SATELLITES": 12.0, "GPS_HDOP": 0.7}'
Received message: b'{"EVENT_NO_TRIP": 223324598, "EVENT_NO_STOP": 223324646, "OPD_DATE": "20DEC2022:00:00:00", "VEHICLE_ID": 3029, "METERS": 65634, "ACT_TIME": 34227, "
122.984738, "GPS_LATITUDE": 45.521522, "GPS_SATELLITES": 12.0, "GPS_HDOP": 0.7}'
Received message: b'{"EVENT_NO_TRIP": 223325108, "EVENT_NO_STOP": 223325175, "OPD_DATE": "20DEC2022:00:00:00", "VEHICLE_ID": 3029, "METERS": 230791, "ACT_TIME": 64754, "
-123.036068, "GPS_LATITUDE": 45.520068, "GPS_SATELLITES": 12.0, "GPS_HDOP": 0.7}'
Received message: b'{"EVENT_NO_TRIP": 223324682, "EVENT_NO_STOP": 223324716, "OPD_DATE": "20DEC2022:00:00:00", "VEHICLE_ID": 3029, "METERS": 87167, "ACT_TIME": 37976, "
122.982582, "GPS_LATITUDE": 45.519805, "GPS_SATELLITES": 12.0, "GPS_HDOP": 0.9}'
```

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.