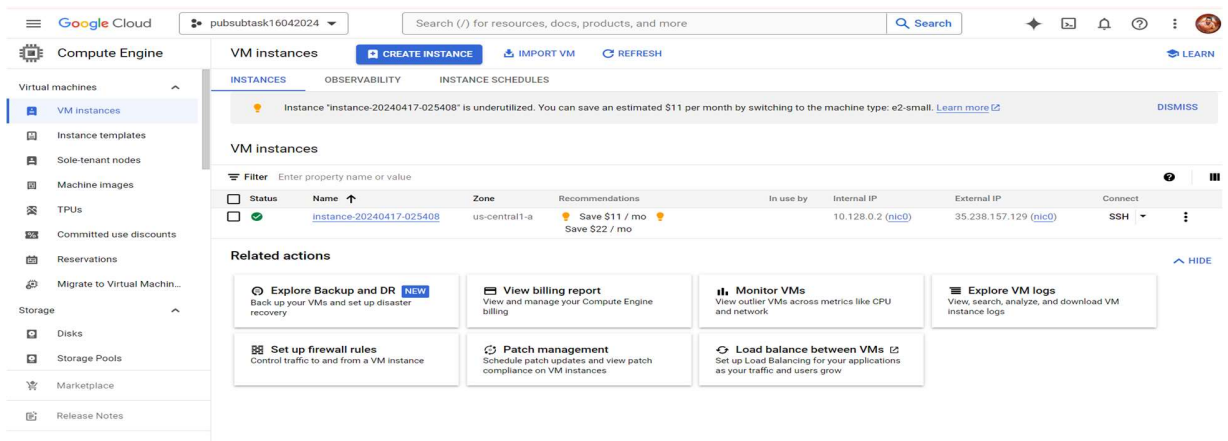


DataEng S24: PubSub

A. [MUST] PubSub Tutorial

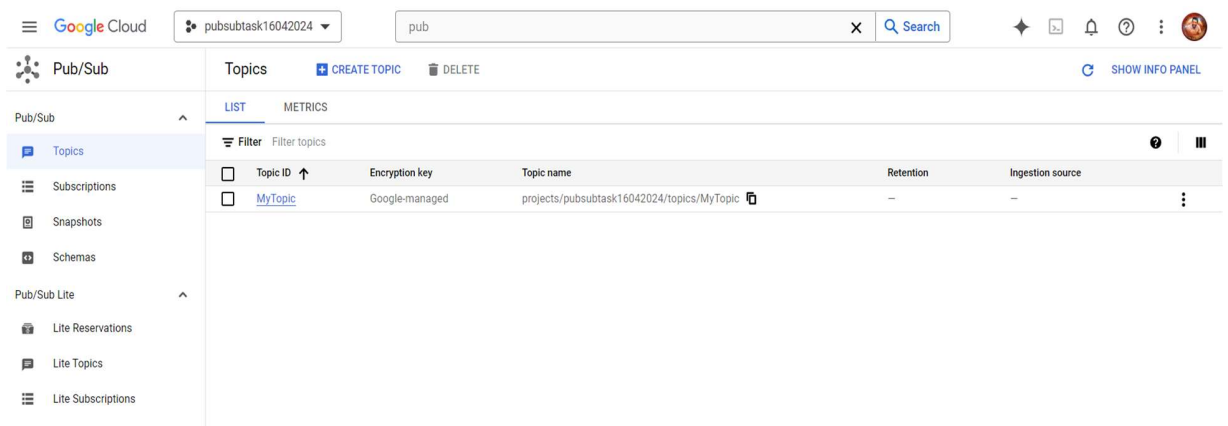
1. Get your cloud.google.com account up and running
 - a. Redeem your GCP coupon
 - b. Login to your GCP console
 - c. Create a new, separate VM instance

Created a new VM instance in my GCP:



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.

Created topics with MyTopic name:



Created subscriptions:

Pub/Sub

Topics

Subscriptions

Snapshots

Schemas

Pub/Sub Lite

Lite Reservations

Lite Topics

Lite Subscriptions

Manage Resources

Release Notes

LIST

METRICS

Filter Filter subscriptions

☐

State

Subscription ID

↑

Delivery type

Topic name

Ack deadline

Reten

☐

✓

MySub

Pull

projects/dataeng-acti...

10 seconds

7 day

☐

✓

MyTopic-sub

Pull

projects/dataeng-acti...

10 seconds

7 day

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.

Publisher Program:

```
GNU nano 5.4
import requests
import json
from google.cloud import pubsub_v1

# Set your Google Cloud project ID and Pub/Sub topic ID
project_id = "pubsubtask16042024"
topic_id = "MyTopic"

# Initialize the Pub/Sub Publisher client
publisher = pubsub_v1.PublisherClient()
topic_path = publisher.topic_path(project_id, topic_id)

# Function to fetch data from the API and publish to Pub/Sub
def fetch_and_publish_data(vehicle_id):
    url = f"https://busdata.cs.pdx.edu/api/getBreadCrumbs?vehicle_id={vehicle_id}"
    response = requests.get(url)
    if response.status_code == 200:
        data = response.json()
        with open(f"bcsample_{vehicle_id}.json", "w") as file:
            json.dump(data, file)
        for record in data:
            data_str = json.dumps(record)
            data_bytes = data_str.encode("utf-8")
            future = publisher.publish(topic_path, data_bytes)
            print(f"Published message: {future.result()}")
    else:
        print(f"Failed to fetch data for vehicle ID {vehicle_id}")

# Vehicle IDs for which data needs to be fetched
vehicle_ids = ["3010", "3951"]

for vehicle_id in vehicle_ids:
    fetch_and_publish_data(vehicle_id)

print(f"Published messages to {topic_path}.")
```

Published Messages:

```
(pubsubtask16042024) X + ▾  
Published message: 10948197463997109  
Published message: 10948097627539218  
Published message: 10948073222215688  
Published message: 10948297522960087  
Published message: 10948471150750931  
Published message: 10948382426311014  
Published message: 10948981094816957  
Published message: 10948663727987642  
Published message: 10948766272632205  
Published message: 10948415416722255  
Published message: 10948700712132325  
Published message: 10948309388155950  
Published message: 10948707123766275  
Published message: 10948123408461043  
Published message: 10948940655888764  
Published message: 10947795404217930  
Published message: 10948444628304515  
Published message: 10948835388865415  
Published message: 10947683411794784  
Published message: 10948897518856405  
Published message: 10948656506154781  
Published message: 10947749852395125  
Published message: 10948380276203216  
Published message: 10949184450707145  
Published message: 10948800983617537  
Published message: 10948439647228453  
Published message: 10948856413687391  
Published message: 10948336105592717  
Published message: 10948713153097617  
Published message: 10948548996423253  
Published message: 10948167550915306  
Published message: 10948884706749023  
Published message: 10948337034452905  
Published message: 10948833994967953  
Published message: 10948952616214621  
Published message: 10948691829437511  
Published message: 10948964123213391  
Published message: 10948737562152840  
Published message: 10948645226574822  
Published messages to projects/pubsubtask16042024/topics/MyTopic.  
jithendrabojedla9999@cloudshell:~ (pubsubtask16042024) $
```

4. Use your receiver python program (from the tutorial) to consume your records.

Receiver code:

```
GNU nano 5.4
from concurrent.futures import TimeoutError
from google.cloud import pubsub_v1

# Set your Google Cloud project ID and Pub/Sub subscription ID
project_id = "pubsubtask16042024"
subscription_id = "MySub"
timeout = 5.0

# Initialize the Pub/Sub Subscriber client
subscriber = pubsub_v1.SubscriberClient()
subscription_path = subscriber.subscription_path(project_id, subscription_id)

# Define the callback function to process incoming messages
def callback(message):
    print(f"Received message: {message.data.decode('utf-8')}")
    message.ack()

# Subscribe to the Pub/Sub topic
streaming_pull_future = subscriber.subscribe(subscription_path, callback=callback)
print(f"Listening for messages on {subscription_path}...\n")

# Wait for messages
try:
    streaming_pull_future.result(timeout=timeout)
except TimeoutError:
    streaming_pull_future.cancel() # Trigger the shutdown
    streaming_pull_future.result() # Block until the shutdown is complete
```

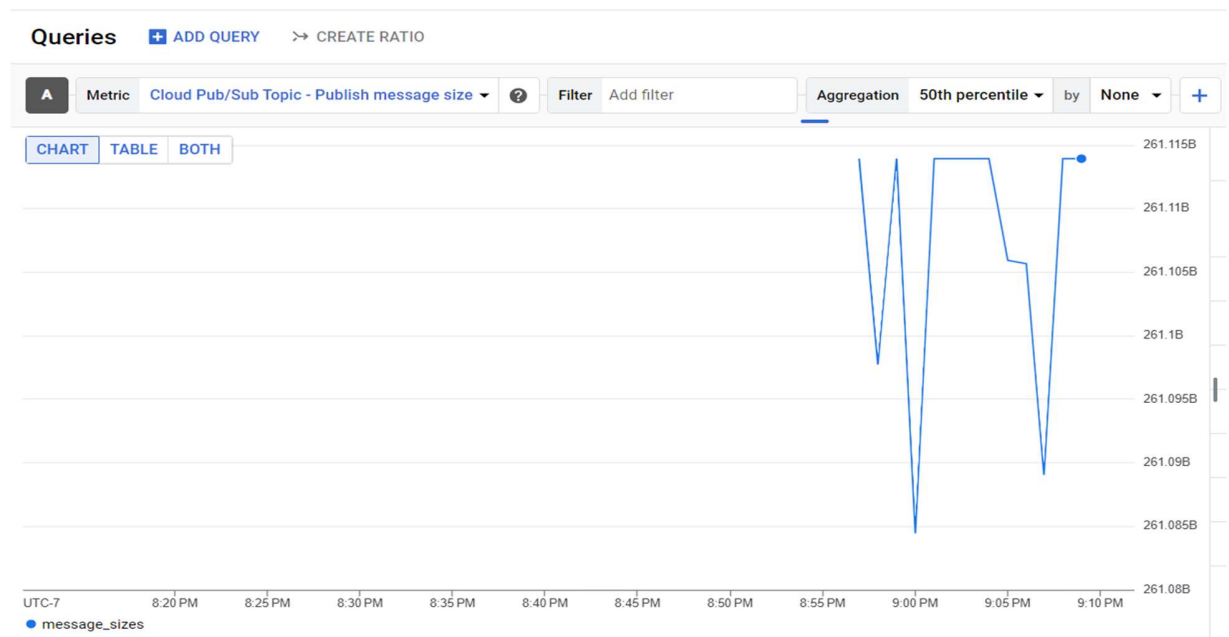
Messages Received:

```
, "GPS_LATITUDE": 45.596702, "GPS_SATELLITES": 11.0, "GPS_HDOP": 0.8}
Received message: {"EVENT_NO_TRIP": 222305889, "EVENT_NO_STOP": 222305903, "OPD_DATE": "19DEC2022:00:00:00", "VEHICLE_ID": 3235, "METERS": 301724, "ACT_TIME": 77147, "GPS_LONGITUDE": -122.661468, "GPS_LATITUDE": 45.58164, "GPS_SATELLITES": 12.0, "GPS_HDOP": 0.7}
Received message: {"EVENT_NO_TRIP": 222305689, "EVENT_NO_STOP": 222305713, "OPD_DATE": "19DEC2022:00:00:00", "VEHICLE_ID": 3235, "METERS": 241605, "ACT_TIME": 64002, "GPS_LONGITUDE": -122.6606, "GPS_LATITUDE": 45.534305, "GPS_SATELLITES": 12.0, "GPS_HDOP": 0.7}
Received message: {"EVENT_NO_TRIP": 222305490, "EVENT_NO_STOP": 222305491, "OPD_DATE": "19DEC2022:00:00:00", "VEHICLE_ID": 3235, "METERS": 179787, "ACT_TIME": 49446, "GPS_LONGITUDE": -122.684857, "GPS_LATITUDE": 45.612147, "GPS_SATELLITES": 12.0, "GPS_HDOP": 0.7}
Received message: {"EVENT_NO_TRIP": 222306137, "EVENT_NO_STOP": 222306141, "OPD_DATE": "19DEC2022:00:00:00", "VEHICLE_ID": 3235, "METERS": 377739, "ACT_TIME": 89360, "GPS_LONGITUDE": -122.681302, "GPS_LATITUDE": 45.539282, "GPS_SATELLITES": 11.0, "GPS_HDOP": 0.7}
Received message: {"EVENT_NO_TRIP": 222306137, "EVENT_NO_STOP": 222306140, "OPD_DATE": "19DEC2022:00:00:00", "VEHICLE_ID": 3235, "METERS": 373866, "ACT_TIME": 89206, "GPS_LONGITUDE": -122.678792, "GPS_LATITUDE": 45.572993, "GPS_SATELLITES": 12.0, "GPS_HDOP": 0.8}
Received message: {"EVENT_NO_TRIP": 222305889, "EVENT_NO_STOP": 222305922, "OPD_DATE": "19DEC2022:00:00:00", "VEHICLE_ID": 3235, "METERS": 305318, "ACT_TIME": 77966, "GPS_LONGITUDE": -122.661662, "GPS_LATITUDE": 45.548927, "GPS_SATELLITES": 12.0, "GPS_HDOP": 0.7}
Received message: {"EVENT_NO_TRIP": 222306137, "EVENT_NO_STOP": 222306139, "OPD_DATE": "19DEC2022:00:00:00", "VEHICLE_ID": 3235, "METERS": 369998, "ACT_TIME": 89056, "GPS_LONGITUDE": -122.68172, "GPS_LATITUDE": 45.607257, "GPS_SATELLITES": 11.0, "GPS_HDOP": 0.7}
Received message: {"EVENT_NO_TRIP": 222305950, "EVENT_NO_STOP": 222305969, "OPD_DATE": "19DEC2022:00:00:00", "VEHICLE_ID": 3235, "METERS": 316756, "ACT_TIME": 80349, "GPS_LONGITUDE": -122.660645, "GPS_LATITUDE": 45.52925, "GPS_SATELLITES": 12.0, "GPS_HDOP": 0.8}
Received message: {"EVENT_NO_TRIP": 222305557, "EVENT_NO_STOP": 222305560, "OPD_DATE": "19DEC2022:00:00:00", "VEHICLE_ID": 3235, "METERS": 199270, "ACT_TIME": 53187, "GPS_LONGITUDE": -122.685393, "GPS_LATITUDE": 45.515535, "GPS_SATELLITES": 12.0, "GPS_HDOP": 1.5}
Received message: {"EVENT_NO_TRIP": 222305490, "EVENT_NO_STOP": 222305493, "OPD_DATE": "19DEC2022:00:00:00", "VEHICLE_ID": 3235, "METERS": 180834, "ACT_TIME": 49677, "GPS_LONGITUDE": -122.680742, "GPS_LATITUDE": 45.609087, "GPS_SATELLITES": 12.0, "GPS_HDOP": 0.6}
Received message: {"EVENT_NO_TRIP": 222306072, "EVENT_NO_STOP": 222306134, "OPD_DATE": "19DEC2022:00:00:00", "VEHICLE_ID": 3235, "METERS": 368170, "ACT_TIME": 88776, "GPS_LONGITUDE": -122.680182, "GPS_LATITUDE": 45.613002, "GPS_SATELLITES": 12.0, "GPS_HDOP": 0.7}
Received message: {"EVENT_NO_TRIP": 222305689, "EVENT_NO_STOP": 222305706, "OPD_DATE": "19DEC2022:00:00:00", "VEHICLE_ID": 3235, "METERS": 239738, "ACT_TIME": 63517, "GPS_LONGITUDE": -122.660688, "GPS_LATITUDE": 45.517312, "GPS_SATELLITES": 12.0, "GPS_HDOP": 0.9}
Received message: {"EVENT_NO_TRIP": 222305689, "EVENT_NO_STOP": 222305754, "OPD_DATE": "19DEC2022:00:00:00", "VEHICLE_ID": 3235, "METERS": 251557, "ACT_TIME": 65904, "GPS_LONGITUDE": -122.678662, "GPS_LATITUDE": 45.59505, "GPS_SATELLITES": 12.0, "GPS_HDOP": 0.7}
Received message: {"EVENT_NO_TRIP": 222305950, "EVENT_NO_STOP": 222305965, "OPD_DATE": "19DEC2022:00:00:00", "VEHICLE_ID": 3235, "METERS": 315461, "ACT_TIME": 80154, "GPS_LONGITUDE": -122.66072, "GPS_LATITUDE": 45.51747, "GPS_SATELLITES": 12.0, "GPS_HDOP": 0.8}
Received message: {"EVENT_NO_TRIP": 222305490, "EVENT_NO_STOP": 222305495, "OPD_DATE": "19DEC2022:00:00:00", "VEHICLE_ID": 3235, "METERS": 182329, "ACT_TIME": 49812, "GPS_LONGITUDE": -122.685457, "GPS_LATITUDE": 45.595907, "GPS_SATELLITES": 12.0, "GPS_HDOP": 0.7}
Received message: {"EVENT_NO_TRIP": 222305490, "EVENT_NO_STOP": 222305495, "OPD_DATE": "19DEC2022:00:00:00", "VEHICLE_ID": 3235, "METERS": 182634, "ACT_TIME": 49852, "GPS_LONGITUDE": -122.686392, "GPS_LATITUDE": 45.595245, "GPS_SATELLITES": 12.0, "GPS_HDOP": 0.7}
Received message: {"EVENT_NO_TRIP": 222306137, "EVENT_NO_STOP": 222306144, "OPD_DATE": "19DEC2022:00:00:00", "VEHICLE_ID": 3235, "METERS": 383386, "ACT_TIME": 89620, "GPS_LONGITUDE": -122.719045, "GPS_LATITUDE": 45.507285, "GPS_SATELLITES": 10.0, "GPS_HDOP": 1.0}
```

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.

Cloud PubSub subscription:



D. [MUST] PubSub Storage

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

Ans:

Messages from vehicle data.json files are reviewed by the publisher before being published to the my-topic. On my-sub, the subscriber keeps an ear out for communications. The subscriber processes every message sent by the publisher's single run on the first run. Until fresh messages are published by the publisher, the subscriber does not get any new messages in subsequent runs.

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

Ans:

The publisher's data may be temporarily held in Pub/Sub before the consumer executes. Google Cloud offers Pub/Sub, a fully managed messaging service built to handle massive, real-time message intake and delivery. When a publisher posts a message to a Pub/Sub topic, the message is held there indefinitely until a subscriber acknowledges it. Messages will stay in the topic for a defined retention time if no subscriber is actively consuming them.

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:

To determine how much data Pub/Sub is storing for your topic, you can use Pub/Sub monitoring tools such as Cloud Monitoring. Pub/Sub provides a number of metrics that you can keep an eye on to measure the effectiveness and utilization of your subscriptions and topics. The amount of unacknowledged messages, the message throughput rate, and the message backlog size are examples of metrics. We can learn more about the quantity of material stored in your subject and the general condition of your Pub/Sub system by looking at these metrics.

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.

Ans:

Topic clean.py:

```
GNU nano 5.4
from google.cloud import pubsub_v1
project_id = "pubsubtask16042024"
topic_name = "MyTopic"
subscriber = pubsub_v1.SubscriberClient()
topic_path = subscriber.topic_path(project_id, topic_name)
subscriptions = subscriber.list_subscriptions(project=f"projects/{project_id}")
for subscription in subscriptions:
    if topic_path in subscription.topic:
        subscription_path = subscription.name
        subscriber.delete_subscription(request={"subscription": subscription_path})
        print(f"All messages in subscription '{subscription_path}' have been deleted.")

print(f"All messages in topic '{topic_path}' have been deleted.")
```

This program defines a callback function to promptly delete each message it receives and creates a subscription to the given topic. After then, it begins to consume messages continuously unless manually stopped.