DataEng: Data Maintenance In-class Assignment

This week you will construct a data archiver that compresses, encrypts and stores pipelined data into a low-cost, high-capacity GCP Storage Bucket.

Submit: Make a copy of this document and use it to record your responses and results (use colored highlighting when recording your responses/results). Store a PDF copy of the document in your git repository along with your code before submitting for this week.

Develop a new python PubSub subscriber similar to the subscribers that you have created multiple times for this class. This new subscriber (archive.py) will receive data from a PubSub topic, compress the data, encrypt the data and store the resulting data into a GCP Storage Bucket.

A. [MUST] Discussion Questions

When archiving data for a data pipeline we could (a) compress, (b) encrypt and/or (c) reduce the data. Here, “reducing the data” refers to the process of interpolating or aggregating detailed data, such as 5 second breadcrumbs for all buses on all trips, into coarser data. For example, we could aggregate 5-second breadcrumbs into 30-second breadcrumbs.

Under what circumstances might each of these transformations (compress, encrypt, reduce) be desirable for data archival?

If we wanted to get rid of redundant data let's say for example the 30 second vs 5 second breadcrumb example, if we needed to save storage we could compress those or reduce the data as well by just getting rid of them.

Compress - Saving storage, getting rid of data we don’t need, or just getting rid of columns

Reduce - Could also be getting rid of data we don’t need, or combining different data that say the same thing, like making the speed column in our project and getting rid of ACT meters.

Encrypt - When you need to, let's say for example mask sensitive data.

Would it make sense to combine these transformations?

You can combine reduce and compress, but otherwise encrypt

Record your responses here and then discuss them with your work group.

B. [MUST] Create Test Pipeline

Create a new PubSub topic called “archivetest” or something similar. Create a new subscriber program (call it archiver.py) that subscribes to the topic, receives the data and (for now) discards it.

To produce test data, copy/reuse the publisher program used for your class project, and alter it to publish to the new archivetest topic. Run this test publisher manually to gather data (1 day and 100 vehicles) from busdata.cs.pdx.edu and test the archivetest topic and your archiver.py program.

As always, you can/should test your code with smaller data sets first. Try it with just one bus or one trip, and then when everything is working, run it with 100 vehicles.

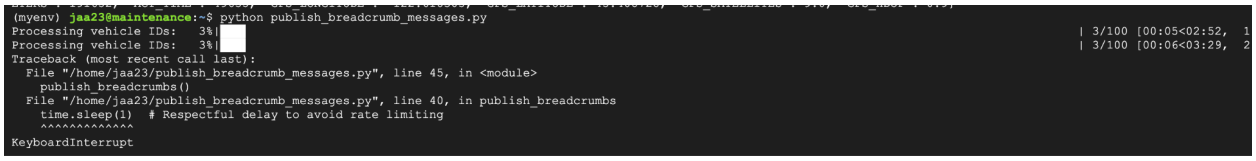
```python
import requests
import pandas as pd
from datetime import datetime
from google.cloud import pubsub_v1
import json
import time
from tqdm import tqdm
import logging

logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')

def get_vehicle_ids():
    doc_key = "1OVKMye65LhbEgMLld5Ol3lOocWUwCaEgnPVgFQf9em0"
    url = f"https://docs.google.com/spreadsheets/d/{doc_key}/export?format=csv"
    response = requests.get(url)
    csv_data = response.content
    with open("vehicle_ids_sheet.csv", "wb") as file:
        file.write(csv_data)
    vehicle_ids = pd.read_csv("vehicle_ids_sheet.csv")['Doodle'].tolist()
    return vehicle_ids

def publish_breadcrumbs():
    project_id = "data-engineering-420705"
    topic_id = "archivetest"
    publisher = pubsub_v1.PublisherClient()
    topic_path = publisher.topic_path(project_id, topic_id)

    vehicle_ids = get_vehicle_ids()

    for vehicle_id in tqdm(vehicle_ids, desc="Processing vehicle IDs"):
        url = f"https://busdata.cs.pdx.edu/api/getBreadCrumbs?vehicle_id={vehicle_id}"
        response = requests.get(url)
        if response.status_code == 200:
            breadcrumbs = response.json()
            for breadcrumb in breadcrumbs:
                data_str = json.dumps(breadcrumb)
                data_bytes = data_str.encode('utf-8')
                future = publisher.publish(topic_path, data_bytes)
#                logging.info(f"Published message: {data_str}")
        time.sleep(1)  # Respectful delay to avoid rate limiting

    print('Published to', topic_path, data_bytes)

if __name__ == "__main__":
    publish_breadcrumbs()
```

This is my modified version of the publisher code from the project to work for the assignment. I included some logging to help me debug, since at first it wasn’t working.

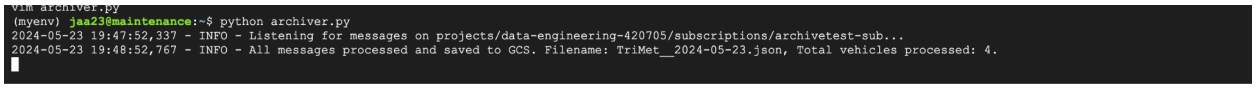
C. [MUST] Store Data to GCP Storage Bucket

Modify archiver.py to store all received data to a GCP Storage Bucket. You will need to create and configure a Storage Bucket for this purpose. We recommend using the Nearline Storage class for this assignment though you are free to choose any of the offered classes of service. Be sure to remove the bucket at the end of the week to reduce GCP credit usage.

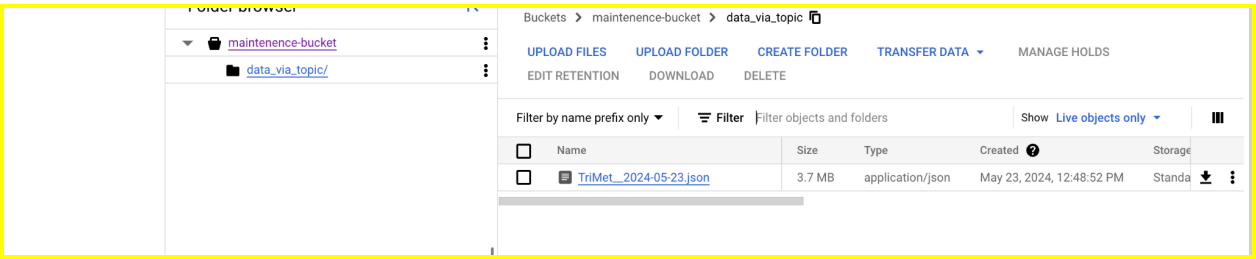
How much bucket space (in KiBs) does it take to store 1 day of breadcrumbs for 100 vehicles?



Here I ran my publisher code, only running 3% (15000 messages)



Here I ran the archiver, which received the 15000 messages, grouped them then stored them in the bucket



Here you see the json object my code created and stored in the bucket, didn't work at first because I didn't realize I had misspelled maintenance in my bucket

This was for 4 vehicles and it was 3.7MB. If we do the math for 100 vehicles, 4*25=100 so 3.7*25 =92.5MB and converting to Kilobytes would then be 92500 KB

D. [SHOULD] Compress

Modify archiver.py to compress the data before it stores the data to the storage bucket. Use zlib compression which is provided by default by python. How large is the archived data compared to the original?

How much bucket space (in KiBs) does it take to store the compressed data?

E. [SHOULD] Encrypt

Modify archiver.py to encrypt the data prior to writing it to the Storage Bucket. Your archive.py program should encrypt after compressing the data. Use RSA encryption as described here: link There is no need to manage your private encryption keys securely for this assignment, and you may keep your private key in a file or within your python code.

Be sure to test your archiver by decrypting and decompressing the data stored in the Storage Bucket. We suggest that you create a separate python program for this purpose.

How much bucket space (in KiBs) does it take to store the encrypted, compressed data?

F. [ASPIRE] Add Archiving to your class project

Add an archiver to your class project's pipeline(s). To receive extra credit, mention your archiver when submitting the next part of your project. You should only need one archiver for the entire project, so coordinate with your teammates if you choose to take this step. For the class project, we recommend storing to a Google Storage Bucket and compressing. Encryption is OK too but not necessary.