DataEng: Data Maintenance In-class Assignment

This week you will gain hands-on experience with Data Maintenance by constructing an automated 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 results. Store a PDF copy of the document in your git repository along with any needed code before submitting for this week.

Your job is to develop a new, separate python Kafka consumer similar to the consumers that you have created multiple times for this class. This new consumer should be called archive.py and should receive all data from a test Kafka topic, compress the data, encrypt the data (optional) and store the compressed data in a GCP Storage Bucket.

Note that each member of your team should build their own archive mechanism. As always, it is OK to help one another, but each person should develop their own python program for archiving.

Discussion Question for Your Entire Group (do this first)

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 transforming detailed data, such as 5 second breadcrumbs for all buses on all trips, into coarser data that contains, for example, only contains a subset of the original data such as only some buses, some trips or possibly fewer breadcrumbs per trip.

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

A. Create test topic

Create new Kafka producer and consumer programs as you did with the Data Transport in-class activity (link to Transport activity). Create a new Kafka topic that is separate from the topic(s) used for your project. Call it “archivetest” or something similar. As with the Data Transport activity you should initially have your new producer produce test data and have a single consumer that consumes any/all data sent to the Kafka topic.

B. Create separate consumer groups

Similar to part H in the Transport activity, create two separate Consumer Groups for your new Kafka topic. Run a separate consumer for each group and verify that each consumer consumes all of the data sent by the producer.

Your first consumer should simply print all data that it receives. This consumer simulates the main branch of your data pipeline. Typically, the main branch of your pipeline would validate, transform, integrate, load, etc. the data, but for this in-class assignment it only needs to print the data.

C. Archive the data in a GCP Storage Bucket

Your second consumer (call it archive.py) should store all received data into a GCP Storage Bucket. You will need to create and configure a Storage Bucket for this purpose. You are free to choose any of the available storage classes. We recommend using the Nearline Storage class.

D. Compress

Modify your archive.py program 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?

E. Encrypt (Optional)

Next, modify your archive.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.

Now large is the archived data?

F. Add Archiving to your class project (Optional)

Add your archive.py program (or something like it) to your class project's pipeline(s). Mention this in your final project presentation video. Because your project is shared among your project team members you will need to coordinate the adding of new Kafka consumer groups so that each team member may safely add their own archiving service. Again, it is not necessary to securely manage your RSA private encryption key, and it is OK to keep it in a file or in your python source code.