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. Compression can be used for data transportation
b. Encryption can be used to store or transport sensitive data
c. Reducing can be used to store data into a database or just for general use for readability as you often don't need every piece of data within a breadcrumb.

Part 1: Data Archival to Cold Storage

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.

How 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.

Part 2: VM Monitoring and Resiliency

G. Virtual Machine Resource Usage Monitoring

This section covers how to monitor your virtual machine's health. Walk through this activity together with your project group since you have the same project VM. In particular we'll be looking into how to check the resource usage on your host. As this is a virtual machine, you have the choice of using traditional Linux command line tools like top, or using GCP UI tools.

For the questions about the pipeline, you can either use the pipeline from the activity above, or modify your course project pipeline. Be sure to set `autocommit = False` when setting up the database connection so it doesn't actually commit the data to your tables and mess up your data. Use the new endpoint: http://www.psudataeng.com:8000/getBreadCrumbDataV2 This endpoint just serves the data from the start of the course again. It'll be shut down at the end of day Friday.

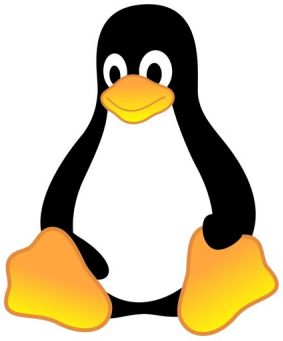
Some handy links are listed below. We recommend skimming or searching them as a reference during the rest of this activity. You are not required to read them. Remember to Google anything that isn't immediately clear!

- Linux command line monitoring tools: https://www.makeuseof.com/best-cli-tools-to-monitor-linux-performance-terminal/
    - My personal favorites: dstat, top, free -h, df -h, du -h -d1, ps aux
- GCP Monitoring overview: https://cloud.google.com/monitoring/docs/monitoring-overview
- GCP VM monitoring dashboards: https://console.cloud.google.com/monitoring/dashboards/resourceList/gce_instance?tab=overview
- GCP Monitoring Processes: https://cloud.google.com/monitoring/agent/process-metrics

- GCP Alerts doc: https://cloud.google.com/monitoring/monitor-compute-engine-virtual-machine
- GCP Alerts UI: https://console.cloud.google.com/monitoring/alerting?walkthrough_id=monitoring_quickstart_compute_uptime

Do you have any other monitoring tools you prefer?

H. Getting to know your VM



Which Linux distribution are you using? (Distro and year: Ubuntu 20)

What are your system's resource allocations? How much of each of the following resources do you have on the VM?

- CPUs
- CPU cores
- Memory
- Swap space
- Disk

Are there other resources you're used to monitoring? When are those resources important to keep an eye on?

I. Pipeline Resource usage

When your pipeline is running, what is the typical system-wide usage of these resources:

- CPU
- Load Average
- Memory
- Disk Usage (fullness)
- Disk Utilization (I/O)
- Network (I/O)

When your pipeline is running, what are some of the top processes that are running? Which ones are consuming the most CPU and memory?

J. Understanding Linux Monitoring Metrics

What is the difference between disk usage and disk utilization?

What is the difference between memory that is free and memory that is available, in the command `free -h`? Is it concerning when there is very little “free” memory, if there is a lot of “available” memory?

```
gen@dataeng:~$ free -h
               total        used        free      shared  buff/cache   available
Mem:           968Mi       189Mi        79Mi        12Mi       699Mi       624Mi
Swap:            0B          0B          0B
```

Does swap space use memory or disk? Is it a good or bad sign (or both/neither) when it is used? What is a standard amount of swap pace to allocate?

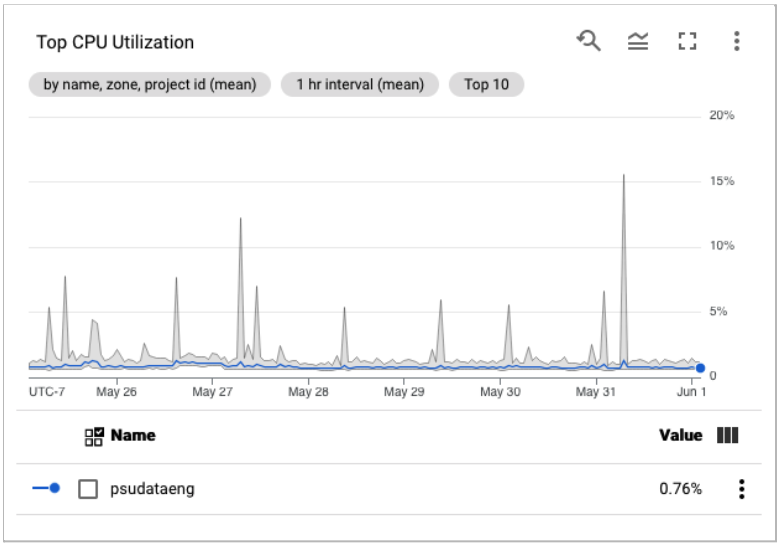
Should you be concerned if the load average is 7?

Yes, the previous question is a trick question. The answer to your followup question: the system has 32 cores. Are you concerned now?

Should you be concerned if the 1 minute load average is really high, but the 5 and 15 minute load averages are normal? Why or why not? What does this mean, is load going up or down?

K. GCP Monitoring

Where in the GCP cloud console can you see charts like the one below, showing the resource usage of your VM?



Where in the GCP cloud console can you see disk usage (not utilization)?

How can you see GCP billing usage, excluding credits from promotions and course coupons?

L. Predicting and Preventing Outages

Consider the following questions, and how you can improve the resiliency of your VM and pipeline.

Running out of resources

Judging from your current usage patterns, and if nothing were to change, on what date do you expect to run out of disk space? What about on your file archive GCP storage bucket? When will you run out of GCP credits?

A picture of a full disk on Windows. Linux just doesn't run out of disk space with quite the same flamboyance.

When your pipeline is downloading new data, is the system running out of any resources? Did it run out of memory? Does it start paging/swapping? Was the CPU pegged? What was the normalized CPU usage (total CPU usage, averaged over the count of CPU cores)?

Judging from the last question, which resource would you say is the biggest bottleneck?

Alerting on and resolving problems

Given the typical resource usage patterns we've explored here, if you were to create some resource usage alerts, what types of thresholds would you set? For example, would you get out of bed to fix a disk at 70% full or wait till 90% full? Thinking of typical human behavior, what issues can you think of if alert thresholds are set too high or too low?

If your system were running out of disk and you needed to free up space right away, what files would you delete first? Could you recreate the data from the remaining files? Do you have a safe copy of the data elsewhere?

If your system were running out of memory, and you knew it was caused by your consumer, how can you kill the consumer process? What effect (if any) would this have on the integrity of data in your database? Could you safely start the consumer again once the system is healthy again, or would you need to fix any data first? Can you think of any database features that could make it more resilient to failures in the middle of running the pipeline?