DataEng S22: Project Assignment 1

Gather and Transport

Due date: April 17, 2022 @10pm PT
Submit: https://forms.gle/9UKRF6aCvXGESjYt7

The first step in building your data pipeline is to provide mechanisms for gathering and transporting the raw data. We have developed a simple web server that provides access to one day's worth of C-Tran GPS 5-second-interval "breadcrumb" data for all buses in the C-Tran system. See it here: http://www.psudataeng.com:8000/getBreadCrumbData

Your job is to gather and transport this data once per day. At first you can just use a web browser to visit the site once per day, download the data, and save it to a file, but by the assignment due date you will:

A. Create, configure and use a Google Cloud Platform (GCP) linux virtual machine.
B. Develop a simple python program to gather the data programmatically
C. Configure your virtual machine to run your gathering client to run daily.
D. Allocate and configure a Kafka message passing "topic" at confluence.com
E. Enhance your data gathering client to parse the breadcrumb data to produce individual JSON records.
F. Enhance your data gathering client to send the individual breadcrumb records to your Kafka topic
G. Develop a python program to consume the breadcrumb readings from the Kafka topic and save them files, one file per day.
H. Configure your VM to run your Kafka consumer constantly so that it always consumes new data.

Before you begin, make sure that you can access http://www.psudataeng.com:8000/getBreadCrumbData, and study the data that it provides. It provides JSON data containing one JSON object per breadcrumb sensor reading. Examine the attributes, and ask questions on the class slack channel if you have any questions about the data.

You can only download data for the current day and not for any past or future days. You will have a 24-hour window to download the data for that day. This website uses the Pacific Timezone.

Extra Credit

This project only requires one VM (e2-medium type VM). However, GCP allows you to create multiple Compute Engine VM's (Virtual Machines), and you can make the project more interesting/realistic (and earn extra credit) if you utilize separate VMs for your data gatherer and Kafka consumer.

A. Create, Configure and Run Your Virtual Machine

We provided you with information about how to obtain your GCP credits. Use them well and monitor your credit availability throughout the quarter. If you have never used GCP before, then this is a good time to learn. Try one of the free, online "How to Use GCP" tutorials available on the internet. Also, see: https://cloud.google.com/compute/docs/quickstart-linux Note, primarily we plan to use the "Compute Engine" features of GCP, so for now you can ignore mention of such things as "Kubernetes Engine", "App Engine" "SQL BigData" and so on.

For this class, create a new project in your GCP account. You can do so from your dashboard (https://console.cloud.google.com/home) on the top left navigation bar. **Make sure your Virtual Machine is running on Pacific time zone** so that your VM's timestamps will make sense for you and for your data. Do all of your class-related tasks within this project.

When redeeming GCP credits and in general when using GCP, **make sure you are logged out of all other google accounts other than your @pdx.edu school account.** Being logged into multiple google accounts is the #1 cause of configuration errors with GCP-related school projects. We suggest you open a new incognito window in chrome (private window in Firefox), log into your PSU google account and then redeem your credits and use cloud.google.com (GCP).

To create a VM, go to your GCP Console. That is, in a browser go to cloud.google.com, Sign In with your @pdx.edu student email account address, and then select "Console" which should appear near the top RH part of your screen after you sign in. From the Navigation menu (on the LH side of the screen) select Compute Engine > VM Instances.

Select "Create Instance". This will take you to a screen that allows you to configure the properties of your VM Instance before you actually create it. For the most part you can just leave the configuration parameters as-is and take the defaults, but you should modify these parameters:

- Select the region as "us-west1(Oregon)"
- Machine Type: choose a "e2-medium" machine type

Select the "Create" button at the bottom to create your instance. This will take you to a VM Instances management page. After a few moments you should see that your VM is up and running.

SSH to your new VM. Make sure that you can access the breadcrumb server by running a command line client such as curl, like this:

```
curl -L http://psudataeng.com:8000/getBreadCrumbData --output
output_file_name
```

See the man page for curl as it can be a very useful tool for testing connectivity. The ‘-L’ flag instructs curl to follow any server redirects.

B. Initial Python Data Gatherer

There is not much to say about this other than you need to write a python program that can make an HTTP request to a web server and retrieve the results. While there are many http-related python packages we suggest that you use the tried and true “urllib” python package. Probably you will need to install this package on your VM.

If you have a multi-person team, then consider assigning one team member to handle the creation and debugging of this python program.

At first the program only needs to write the data to a file and then exit. In Step E you will enhance this program to parse the JSON data and send individual records to a Kafka topic.

C. Run the Data Gatherer Daily

On a linux system there are many ways to run programs automatically, periodically. We suggest that you investigate a simple option such as cron. See this tutorial which explains the basics of using cron on Ubuntu linux.

Your job is to use cron (or some other mechanism of your choice) to run your python data gatherer daily. You can run it at any time of day. We suggest that during your debugging you configure cron to run your client more frequently so that both cron and your python program are working correctly. When you are happy with the results then re-configure cron to run once or twice per day.

Test: after you are comfortable that cron (or your favored mechanism) is configured properly, then wait one day and make sure that your data is being gathered automatically as expected. Success? Congratulations! You successfully built a data pipeline, an automated software system that moves a continuous, unbounded source of data from an initial state closer to its destination state. It might not be complete, but it’s a good start.

Extra Credit: make your data pipeline reliable and tolerant of VM crashes, network failures, limited storage space and other types of unexpected computer systems shortages and problems. We do not expect your pipeline to be ultra-reliable, but can be fun to do.

D. Configure Kafka

Apache Kafka is an open-source distributed stream processing platform that provides a unified, high-throughput, low-latency platform for handling real-time data feeds. Large organizations frequently use it when they have many, varied producers and consumers of data streams. It also provides many valuable systems features related to high-availability, high throughput, scalability, security and performance.

Your pipeline will be relatively small with well-known producers and consumers, so it probably does not need something like Kafka, but we will use it anyway so that you get some valuable experience with general producer/consumer stream-oriented systems. You will use a free trial version of Kafka that is hosted at Confluence.com which has a special integration relationship with GCP.

Complete the steps in the separate project instruction document for the Kafka configuration. Note that it references two additional tutorials for you to complete. If you are doing this project as a multi-person team, then consider assigning one teammate to handle all Kafka-related configuration for your project. Also, we will be using Kafka in class in week 3, so all members of your team will get some hands-on experience with it at that time.

E. Parse JSON into Individual Breadcrumb Records

Update your Data Gatherer python program to parse the JSON data received from the breadcrumbs service. There are many ways to parse JSON data in python, we recommend using the basic, built-in json module provided in python.

Handling the breadcrumb data as individual records might not be the most efficient way to transport the data, it might be more efficient to collect the breadcrumb readings into larger aggregates such as “trip”, “route”, “bus” or “hour of the day”. But we require you to send each reading individually to get a better understanding of how the data pipeline might work in a live system in which buses emit GPS locations dynamically throughout the day.

F. Send Breadcrumb Records to Kafka Topic

Create a new Kafka topic to use for your breadcrumbs sensor readings.

Modify your Data Gatherer python code to send each sensor reading to the Kafka topic.

G. Kafka Consumer

Modify the consumer.py program used in the Kafka tutorial to consume the data from your new Kafka topic. At this point this consumer does not need to do much more than write the incoming records to file. You will enhance this program in the future to do validation, transformation, enhancement and storage of the sensor readings.

H. Configure Linux to Run Kafka Consumer Continually

To keep the pipeline running you need to make sure that your Kafka consumer runs continually. There are many ways to keep a process running on Linux, we suggest that you configure your program as a system service. See this tutorial for more information about how to configure a systemd system service: https://www.ubuntudoc.com/how-to-create-new-service-with-systemd

Again, if you have a multi-person team, then consider assigning this as a separate concurrent task because it can be done separately long before the completion of other tasks in this project assignment.

Submission

Congratulations! You now have a working data pipeline consisting of multiple python programs connected with an event stream. It runs automatically and is ready to be enhanced with data transformations and loading to a database server (Assignment 2).

To submit your completed Assignment 1, create a google document containing the table shown below. Share the document as viewable by anybody at PSU who has the link. You do NOT need to share it with the individual instructors. Then include the URL of the document in the DataEng Project Assignment Submission form.

DataEng Project Assignment 1 Submission Document

Construct a table showing each day for which your pipeline successfully, automatically processed one complete day’s worth of sensor readings. Don’t worry if you have few/many rows. Your grade does not depend on how many rows you filled in the table. The table should look like this:

Date	Day of Week	Approximate Time of day for your data access	# Sensor Readings	Total Data (KBs)	# Kafka messages