SEIS 736: BIG DATA ENGINEERING PROJECT

NAME: CARL EDEM DEKPOR

TOPIC: DATA STREAMING WITH KAFKA ON DATABRICKS NOTEBOOK

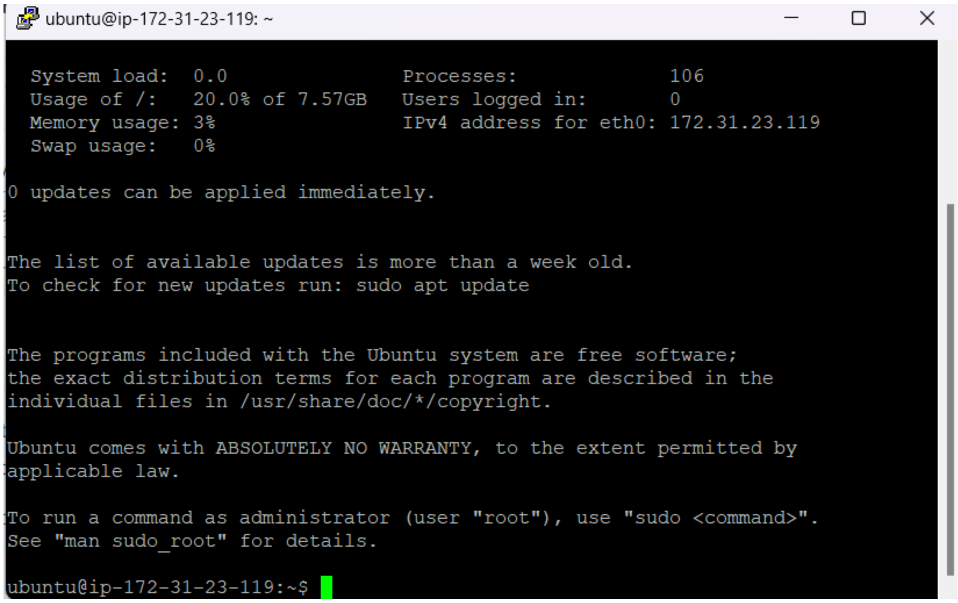
Background

Challenges

My initial goal of this project was to create an Amazon linux EC2 instance, download java, install Kafka, run the Kafka server, zookeeper server, create my Kafka topic and grab data from this API (https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present-Dashboard/5cd6-ry5g).

My first challenge was connection to SSH. I was generating the .pem key instead of .pkk since I was using PuTTY. I eventually figured that out.

Second challenge was memory issue after creating my instance. I had to increase the processing memory size of my instance as recommended by the professor to t2.large on AWS (from 1 to 8). Which worked fine. I was able to get my instance running.



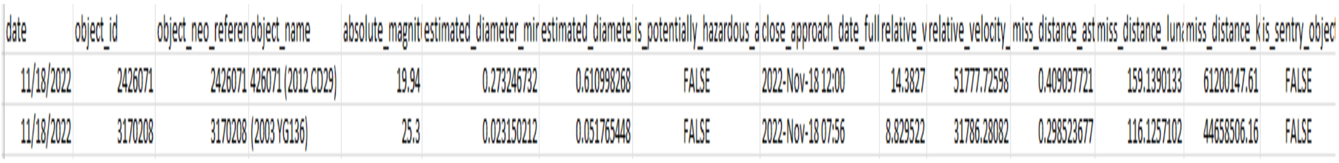
```
  System load:  0.0               Processes:             106
  Usage of /:   20.0% of 7.57GB   Users logged in:       0
  Memory usage: 3%                IPv4 address for eth0: 172.31.23.119
  Swap usage:   0%

0 updates can be applied immediately.


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-23-119:~$
```

After installing Kafka and running the zookeeper server and creating my topic, I encountered challenges grabbing data from the API to my topic. I checked the status of the topic which was active and could read data written to it but couldn't grab data from the topic. Several codes and troubleshoot from online resources didn't work. So, I figured there was a problem with the API. I however pivoted and installed Kafka on the databricks notebook and proceeded to use a different API.

In this project I used Kafka to grab data from an API provided by NASA (https://api.nasa.gov/) to process and stream data and visualized the output on databricks notebook. All artifacts used in the project have been documented in databricks notebook and submitted.

About the dataset:

Near Earth Object Web Service (NeoWs) is a RESTful web service for near earth Asteroid information. With NeoWs a user can: search for Asteroids based on their closest approach date to Earth, lookup a specific Asteroid with its NASA JPL small body id, as well as browse the overall dataset. 15 fields were used from the dataset.



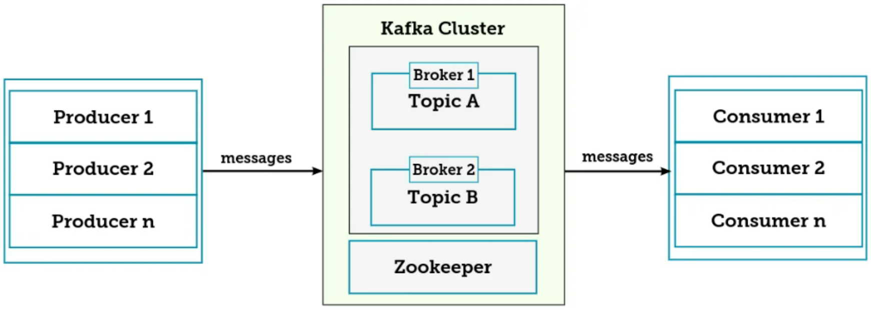
date	object_id	object_neo_referen	object_name	absolute_magnit	estimated_diameter_mir	estimated_diamete	is_potentially_hazardous_a	close_approach_date_full	relative_v	relative_velocity_	miss_distance_ast	miss_distance_luna	miss_distance_k	is_sentry_object
11/18/2022	2426071	2426071	426071 (2012 CD29)	19.94	0.273246732	0.610998268	FALSE	2022-Nov-18 12:00	14.3827	51777.72598	0.409097721	159.1390133	61200147.61	FALSE
11/18/2022	3170208	3170208	(2003 YG136)	25.3	0.023150212	0.051765448	FALSE	2022-Nov-18 07:56	8.829522	31786.28082	0.298523677	116.1257102	44658506.16	FALSE

Kafka

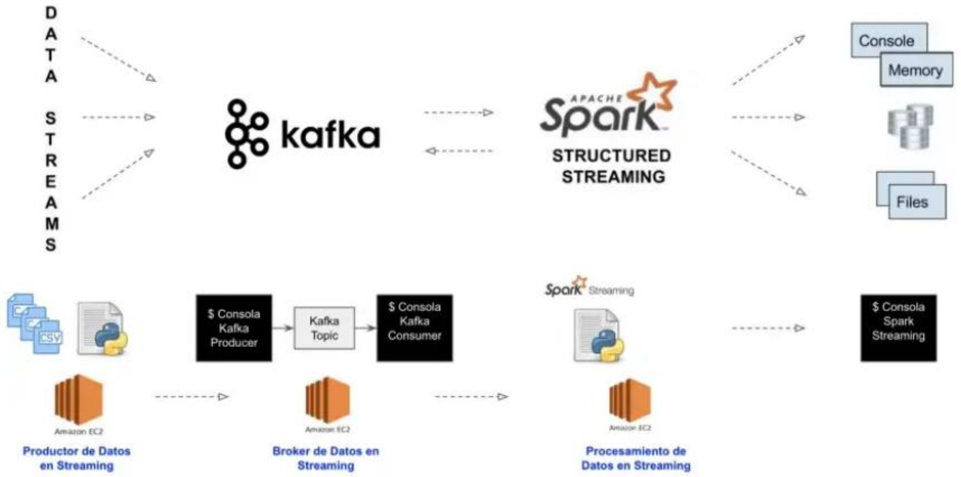
Kafka is an open-source publishing and subscribe messaging system that is used for building streaming analytics platform and data integration pipelines. Kafka is both a queue for parallelizing tasks and a messaging-oriented middleware for service integration. The Kafka message broker (cluster) ingests and stores streams of messages (records) from event producers, which are later distributed to consumer services asynchronously when requested.

- **Topic**: A named resource to which a particular stream of messages is stored and published.
- **Producer**: A client application that creates and publishes records/messages to a Kafka topic(s).
- **Consumer**: A client application that subscribes to a topic(s) to receive data from it.
- **Message**: The combination of data and associated metadata that a producer application writes to a topic and is eventually consumed by consumers

Kafka Architecture



streaming architecture diagram

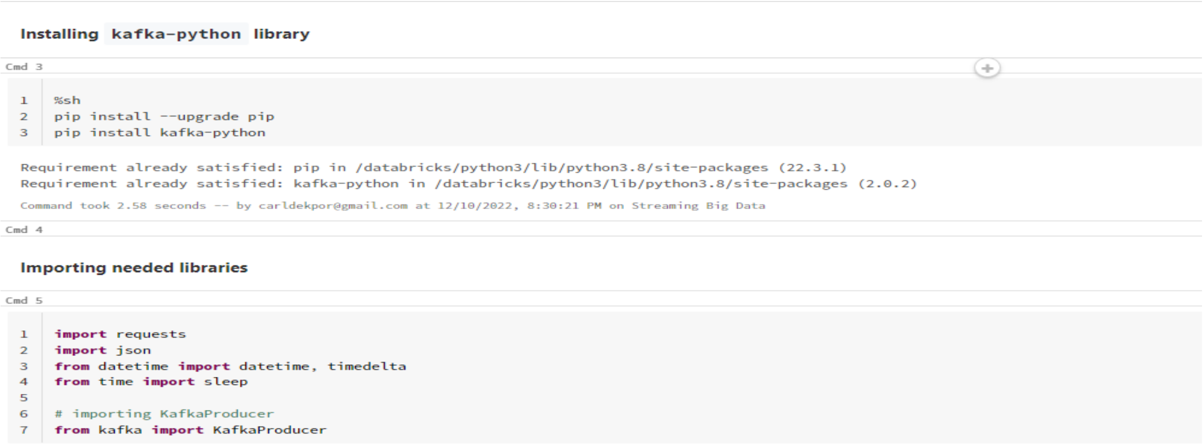


Methodology

I created a cluster (name: big-data-stream), installed Kafka on databricks notebook and initialized zookeeper. Zookeeper is centralized manager that will help to store the metadata information of the consumers, producers, brokers.

Since zookeeper was running on one notebook, I created a new notebook for the Kafka server. A new notebook was also created for the creation of the Kafka topic which I named '*nasatopic*'.

I then created a Kafka producer with python by installing Kafka-python libraries.



Installing `kafka-python` library

Cmd 3

```
%sh
pip install --upgrade pip
pip install kafka-python
```

```
Requirement already satisfied: pip in /databricks/python3/lib/python3.8/site-packages (22.3.1)
Requirement already satisfied: kafka-python in /databricks/python3/lib/python3.8/site-packages (2.0.2)
Command took 2.58 seconds -- by carldekpor@gmail.com at 12/10/2022, 8:30:21 PM on Streaming Big Data
```

Cmd 4

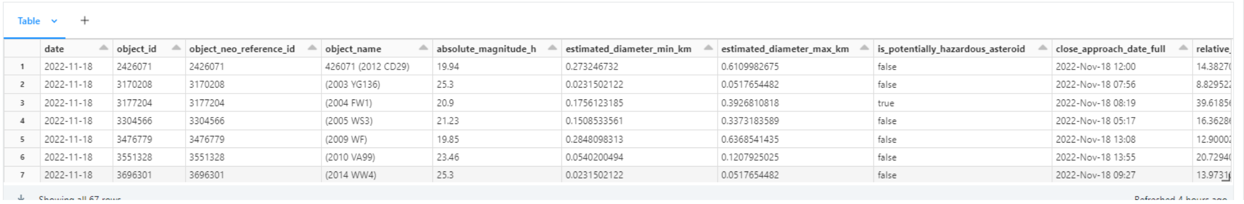
Importing needed libraries

Cmd 5

```
import requests
import json
from datetime import datetime, timedelta
from time import sleep

# importing KafkaProducer
from kafka import KafkaProducer
```

The producer will call the API once (from a range of N days of information) and every 2 seconds the process will send each item (asteriod info - info date) to the Kafka server.

Using pyspark the data structure was defined for the final 15 columns for visualization. Data was read from the Kafka topic, which was written to a delta table.

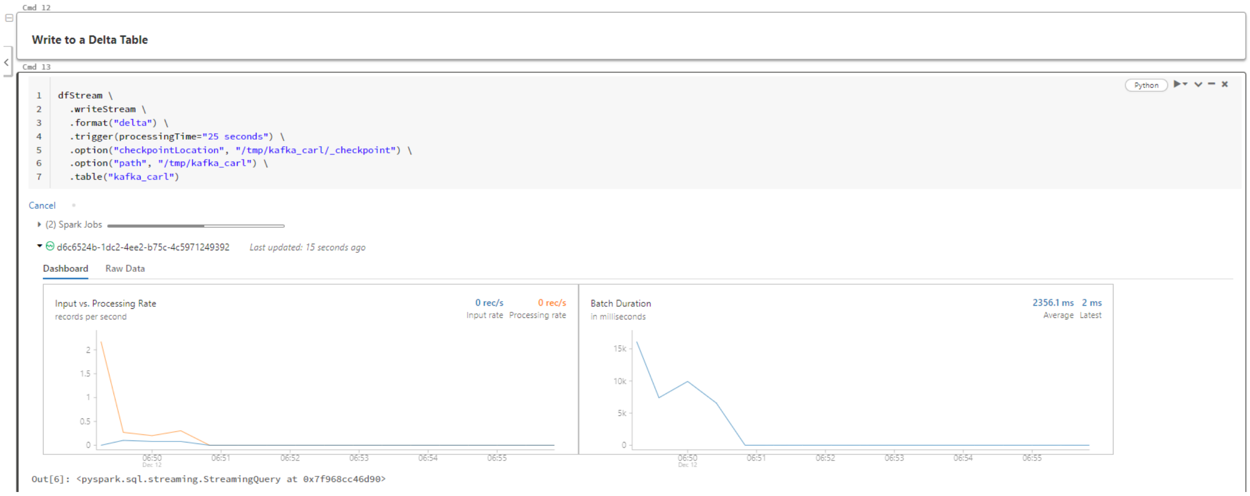


	date	object_id	object_neo_reference_id	object_name	absolute_magnitude_h	estimated_diameter_min_km	estimated_diameter_max_km	is_potentially_hazardous_asteroid	close_approach_date_full	relative_
1	2022-11-18	2426071	2426071	426071 (2012 CD29)	19.94	0.273246732	0.6109982675	false	2022-Nov-18 12:00	14.3827
2	2022-11-18	3170208	3170208	(2003 YG136)	25.3	0.0231502122	0.0517654482	false	2022-Nov-18 07:56	8.82952
3	2022-11-18	3177204	3177204	(2004 FW1)	20.9	0.1756123185	0.3926810818	true	2022-Nov-18 08:19	39.6185
4	2022-11-18	3304566	3304566	(2005 WS3)	21.23	0.1508533561	0.3373183589	false	2022-Nov-18 05:17	16.3628
5	2022-11-18	3476779	3476779	(2009 WF)	19.85	0.2848098313	0.6368541435	false	2022-Nov-18 13:08	12.9000
6	2022-11-18	3551328	3551328	(2010 VA99)	23.46	0.0540200494	0.1207925025	false	2022-Nov-18 13:55	20.7294
7	2022-11-18	3696301	3696301	(2014 WW4)	25.3	0.0231502122	0.0517654482	false	2022-Nov-18 09:27	13.9731

Showing all 67 rows

Refreshed 4 hours ago

Snapshot of delta table



Cmd 12

Write to a Delta Table

Cmd 13

```python
dfStream \
  .writeStream \
  .format("delta") \
  .trigger(processingTime="25 seconds") \
  .option("checkpointLocation", "/tmp/kafka_carl/_checkpoint") \
  .option("path", "/tmp/kafka_carl") \
  .table("kafka_carl")
```

Cancel

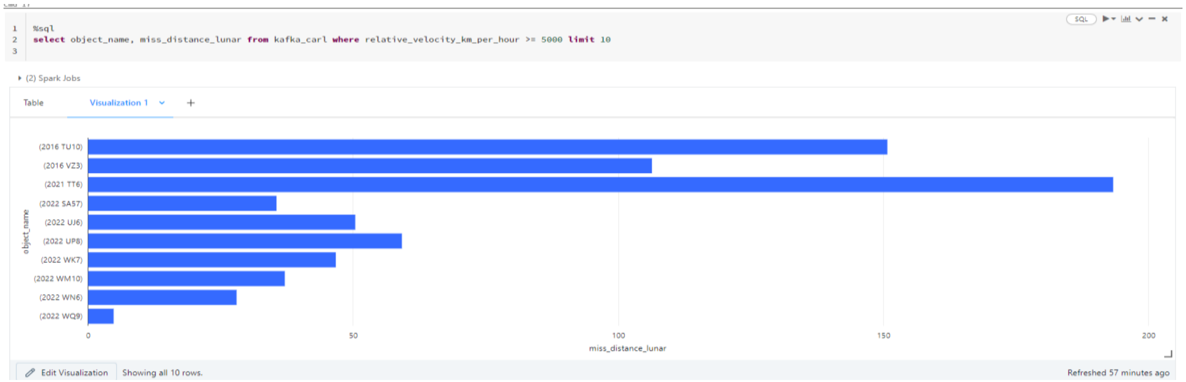
(2) Spark Jobs

d6c6524b-1dc2-4ee2-b75c-4c5971249392 *Last updated: 15 seconds ago*

```
Out[6]: <pyspark.sql.streaming.StreamingQuery at 0x7f968cc46d90>
```

I wrote SQL queries to retrieve data from the table for visualization.

Visualization of 10 asteroids with relative velocity of 5000km/hr



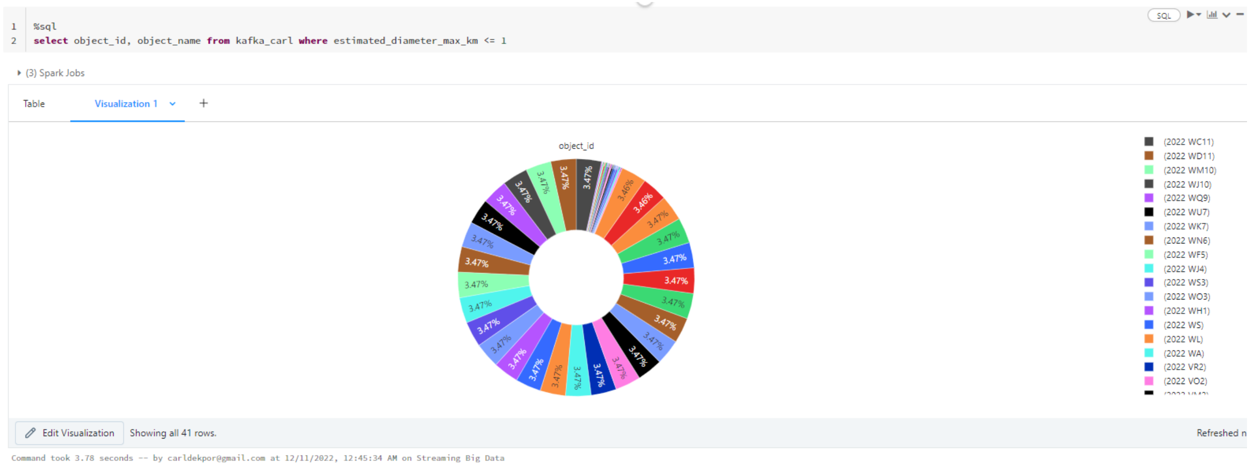
```sql
%sql
select object_name, miss_distance_lunar from kafka_carl where relative_velocity_km_per_hour >= 5000 limit 10
```

(2) Spark Jobs

Showing all 10 rows.

Refreshed 57 minutes ago

Visualization of asteroid names with estimated diameter less than or equal to 1km



```
%sql
select object_id, object_name from kafka_carl where estimated_diameter_max_km <= 1
```

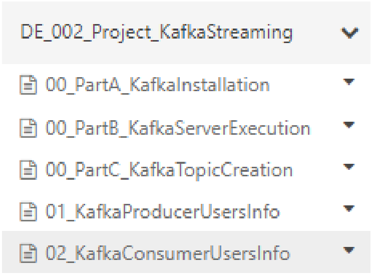
(3) Spark Jobs

Showing all 41 rows.

```
Command took 3.78 seconds -- by carldekpor@gmail.com at 12/11/2022, 12:45:34 AM on Streaming Big Data
```

Running of artifacts

Open the attached .dbc file and open the tabs in the order below:



DE_002_Project_KafkaStreaming

- 00_PartA_KafkaInstallation
- 00_PartB_KafkaServerExecution
- 00_PartC_KafkaTopicCreation
- 01_KafkaProducerUsersInfo
- 02_KafkaConsumerUsersInfo

Conclusion

I got a better understanding of the Kafka architecture and how the various components work together. The various codes used in the project gave me practical experience on how streaming architecture is managed and designed. The use of the databricks notebook community edition was help in providing compute for processing and visualization. Overall, this was a very good learning curve.

References

Important Kafka CLI Commands to Know in 2022 - Learn
https://hevodata.com/learn/kafka-cli-commands/

Structured Streaming Kafka Example - Databricks
https://docs.databricks.com/_static/notebooks/structured-streaming-kafka.html

Kafka basic concepts and building a streaming architecture
Data Streaming with Kafka: Basic Concepts and Building a Project in Databricks | by Luis Miguel Miranda | Nov, 2022 | Medium

Data Streaming with Kafka: Basic Concepts and Building a Project in Databricks
https://medium.com/@lmirandad27/data-streaming-with-kafka-basic-concepts-and-building-a-project-in-databricks-cd762946bab7

Apache Kafka
https://docs.databricks.com/structured-streaming/kafka.html

Procesamiento de Datos en Streaming usando Kafka y Spark Structured Streaming
https://mtpradoc.medium.com/procesamiento-de-datos-en-streaming-usando-kafka-y-spark-structured-streaming-10f91b68b402