**Final Project**

Harsh A. Jajal

Big Data Analytics, Georgian College

BDAT 1002 Data System Architecture

Saber Amini

19th August 2023

**Introduction**

One can argue that the most challenging task in a Big Data setting is getting the data that can then be used for data analysis and predictions. Towards this goal, in this project, you will be setting up a pipeline to ingest data from twitter NewsAPI using Apache Kafka, clean and process it, and load it into a Hive table for analysis. You can also optionally use a large language model (LLM) to gain further insights or transformations on the data.

**Producer and Consumer in Apache Kafka**

A "producer" and a "consumer" are two fundamental elements that play a crucial part in the design of Apache Kafka, which is a distributed streaming platform. They are in charge of, respectively, sending data (messages or records) to Kafka topics and obtaining data from those topics.

**Python producer and Consumer**

The confluent-kafka-python package can be used to implement Kafka producers and consumers in the context of Apache Kafka and utilizing Python. This module offers a Python wrapper for the Confluent Kafka client, enabling easy communication with Kafka clusters.

A Kafka producer in Python is responsible for sending messages to Kafka topics.

A Kafka consumer in Python is responsible for retrieving and processing messages from Kafka topics.

**Producer In Python**

**A black screen with many colorful lines

Description automatically generated with medium confidence**

**Import Statement:** First, the script imports the required libraries. The Kafka library's KafkaProducer is imported, enabling communication with Kafka clusters. To handle JSON serialization and deserialization of data, json is imported. To communicate with the NewsAPI service, NewsApiClient is loaded from the newsapi library.

**Kafka Configuration:** The IP and port of the Kafka broker to which the producer will connect are indicated by the bootstrap\_servers variable, which is set to "localhost:9092".

The Kafka topic to which the news articles will be transmitted is designated by the producer\_topic variable, which is set to'my-news'.

**Kafka Producer Initialization:** Using the specified bootstrap\_servers, a KafkaProducer instance is created. A lambda function that serializes Python dictionaries into byte strings with JSON encoding is the value\_serializer parameter.

**NewsAPI Configuration:** The API key needed to access the NewsAPI service is stored in the news\_api\_key variable. Using the API key, a NewsApiClient object is constructed.

**Sources:** The sources variable has a list of news sources, separated by commas, from which the script will retrieve news articles. Well-known sources including BBC News, CNN, Fox News, and others are among these sources.

**Fetching and Sending News Articles:** The newsapi.get\_everything() function is used to locate news items in the English language that contain the term "france" from the supplied sources.

Each article from the retrieved results is iterated through in a loop. For every piece:

The producer's send method is used to transmit the article as a JSON-encoded message to the Kafka topic. The article dictionary is the value argument's default setting.

The console is printed with the title of the finished piece.

**Closing the Producer:** Following the processing of all articles, the producer is shut down using the close technique.

This script demonstrates how to use a Kafka producer to send news articles to a Kafka topic after retrieving them using the NewsAPI. It's crucial to keep in mind that this script requires both a working Kafka broker at 'localhost:9092' and a valid NewsAPI API key to function properly. Additionally, handling exceptions, implementing error logging, and improving the data selection criteria for fetching news items are all additional error scenarios and edge circumstances that you might wish to manage.

**Consumer in Python**

**A computer screen shot of a program code

Description automatically generated**

**Import necessary libraries**: The code begins by importing the necessary libraries: json for processing JSON data, csv for working with CSV files, and os for file and directory operations. KafkaConsumer is used to consume Kafka messages.

**Kafka configuration:** The Kafka configuration is described in this section. The Kafka broker is executing on the local system at port 9092 since it sets the address of the Kafka broker (bootstrap\_servers) to "localhost:9092". Additionally, it lists "my-news" as the Kafka subject to consume from.

**Kafka consumer initialization:** Here, the previously specified Kafka topic, bootstrap servers, and other parameters are used to initialize a Kafka consumer. The consumer will begin reading messages from the beginning of the subject because the auto\_offset\_reset is set to 'earliest'. The byte-encoded JSON messages are converted into Python dictionaries using the value\_deserializer, which is defined.

**Create the output directory:** The resulting CSV file will be saved in the directory specified by the script. The os.makedirs() function is used to create the directory if it doesn't already exist.

**CSV file:** This line combines the output directory path with the desired filename (news\_data.csv) to define the path for the CSV file (csv\_file\_path). The open() function is then used by the script to open the CSV file in write mode ('w'), and it is then assigned to the csv\_file variable. To write data to the CSV file, a csv.writer object (csv\_writer) is created. The header row with column names is written by the script.

A screen shot of a computer program

Description automatically generated

**Consume messages and save as CSV rows:** The initialized consumer is used in this loop to consume messages from the Kafka topic. It takes the message's value and extracts the JSON data (also known as a "article") for each message. The title, description, source, publishedAt, and URL fields from the article are taken out and saved in the cleaned\_data list.

The console prints the retrieved data for viewing. A row of equal signs is printed between each field to serve as a separator.

Utilizing the csv\_writer.writerow() function, the extracted data is then written as a row to the CSV file.

**Close the CSV file and consumer:** Finally, the Kafka consumer and CSV file are closed to free up resources.

To sum up, this script establishes a connection to a Kafka topic, consumes messages containing JSON data messages representing news items, and then extracts particular fields from the JSON, prints them to the console, and writes them to a CSV file along with a header row. The Kafka topic's data is organized and processed by the code.

**Setup Kafka Consumer to read news articles**

A screenshot of a computer screen

Description automatically generated

Csv file has be created and cleaned and .head command used to observe the csv file

A screen shot of a computer

Description automatically generated

After observing the csv file, we started hive and performed some analysis.

**Load Data into a Hive Table**

You're making a database called mydb in this instance. After that, you start utilizing this database. A database called news\_data is then created with the columns Title, Description, Source, PublishedAt, and URL. News data will be kept in this table. The table is stored as a text file and is configured to accommodate comma-separated data.

A computer screen with white text

Description automatically generated

Loading Data into the Table: This command overwrites any existing data in the news\_data table by loading data from the local file location supplied (news\_data.csv).

Selecting and Displaying Data(Validation): This query displays all columns for each of the first 10 rows it receives from the news\_data dataset.

A black screen with white text

Description automatically generated

**Queries that I Run**

**Query 1: Count the Number of Articles:** This query calculates and displays the total number of articles in the news\_data table.

A screen shot of a computer

Description automatically generated

**Query 2: Find the Latest Articles:** This query retrieves the titles, sources, and publication dates of the latest articles in the news\_data table. It orders the results by the PublishedAt column in descending order and limits the output to the top 10 articles.

**A screenshot of a computer

Description automatically generated**

**Query 3: Average Description Length:** This query calculates and displays the average length of the Description column values in the news\_data table.

**A screen shot of a computer

Description automatically generated**

**Conclusion**

**Producer Script:**

Created a producer script that fetches news articles using the News API.

The producer sends the articles as JSON payloads to a Kafka topic named "my-news".

Kafka Topic Creation:

Created a Kafka topic named "my-news" with partitions to store the produced news articles.

**Consumer Script:**

Created a consumer script that reads articles from the Kafka topic "my-news".

The consumer script can be further extended to save the data to HDFS or perform other processing.

**Hive Table Creation:**

Created a Hive table named "news\_data" to store the news articles.

The table's columns match the structure of the CSV data: Title, Description, Source, PublishedAt, and URL.

**Data Loading:**

Loaded the news data from the CSV file into the Hive table using the LOAD DATA statement.

**Query Examples:**

Learned how to run various queries in Hive to analyze the loaded news data.

These queries included counting articles, Finding Articles, and more.