# Midterm Project

**PROG8451: BIG DATA INTEGRTION & STORAGE**

Aakash Baskaran | 8937042

Big Data Solutions Architecture, Conestoga College

PROG8451 – Big Data Integration & Storage

Professor Ms. Shanti

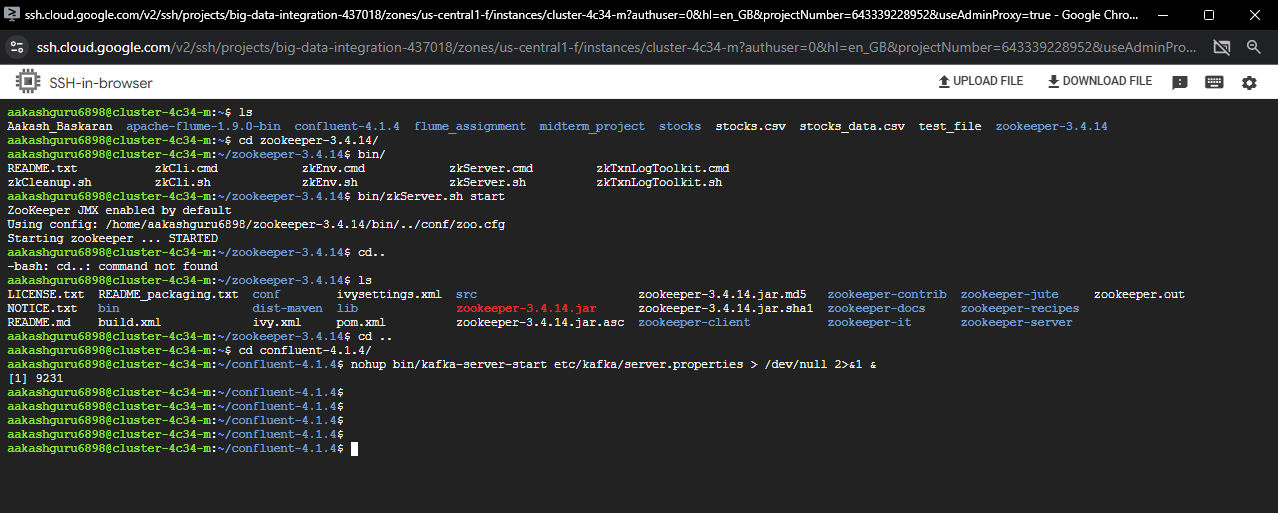
ETL REPORT ON ANALYSIS OF NEWS\_API DATA

# Introduction

This report provides an overview of the ETL pipeline put into operation for processing and analyzing data from NewsAPI. The pipeline has a number of key steps: extraction of data from NewsAPI, transformation, storage in HDFS, and finally, analysis using Apache Hive.

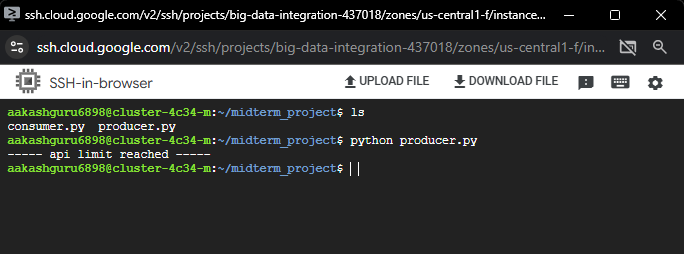
# Environment Setup

The ETL pipeline starts by setting up Apache ZooKeeper and Apache Kafka.



# Data Extraction (Producer)

A producer script is responsible for fetching data from the NewsAPI using its API. This data is then passing it on to the Kafka topic as a message broker.



A computer screen shot of a program

Description automatically generated

The first few lines import necessary libraries, including Kafka Producer for interactions with Kafka, requests to make HTTP requests, json to handle JSON data, and time to add some time delay.

A screen shot of a computer program

Description automatically generated

The get\_api\_data function is defined to fetch data from the News API, convert it to JSON, and send it to a Kafka topic. It also handles errors such as reaching API limits.

It sets up Kafka parameters, including the topic name and the address of the broker (localhost:9092).

News API Configuration: Setting the parameters of the News API, including a query term set to "football", sort by popularity, API key, page size, and the starting page.

Creates a Kafka producer object; this object is in charge of sending messages to the Kafka broker.

The API key, API base URL, query parameters, Kafka topic, and the producer object are passed to the get\_api\_data function. This function makes a request for fetching news articles in pages, converts the result into JSON, and sends it to the specified Kafka topic. It also includes handling for when the API limit has been reached or when there is an HTTP error.

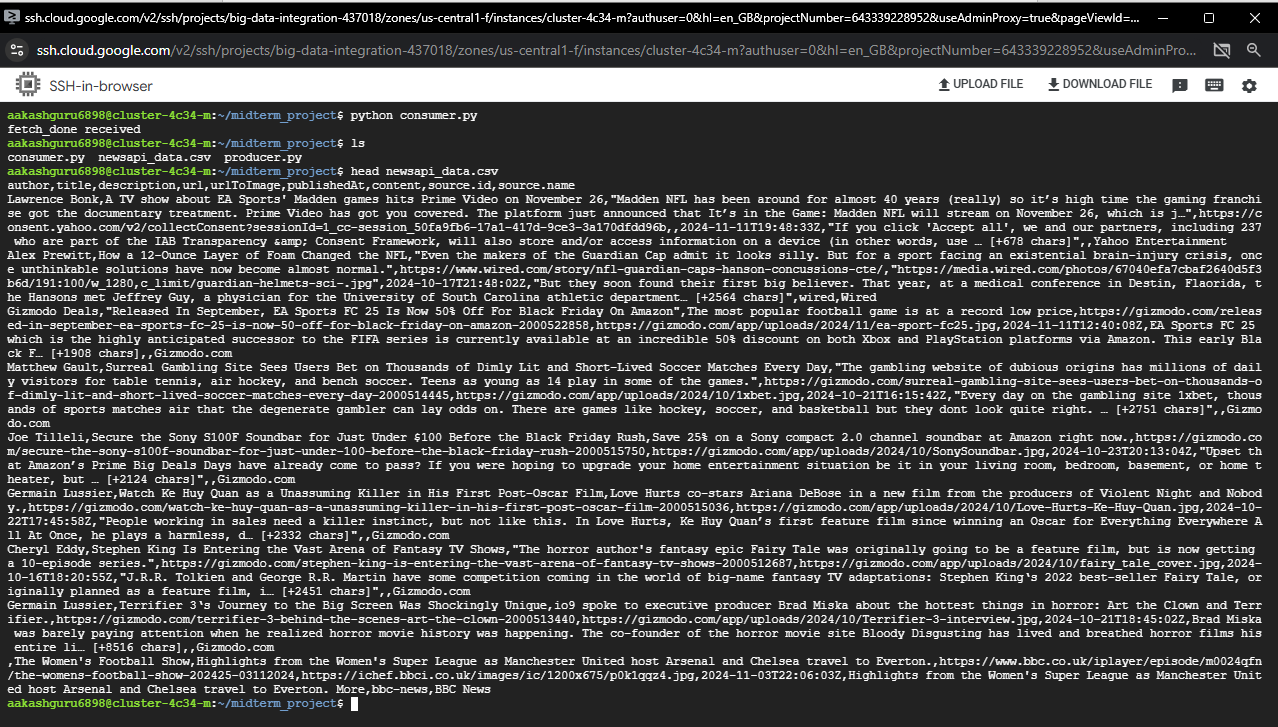
The script uses a while loop to continuously fetch news pages until there are no more articles. Each page is sent to the Kafka topic as a JSON-encoded message. The loop waits 1 second between requests.

In case the API limit is reached (HTTP status code 426), send a special message to the Kafka topic meaning that fetching is done.

When fetching has been completed or an error has occurred, close the Kafka producer.

# Data Transformation (Consumer)

The consumer script consumes messages from the Kafka topic, processes the data, and converts it into a CSV file. The CSV file is saved locally for further analysis.

****

## 

## Diving deep into consumer file

The script is used to set up Kafka parameters, including the name of the topic, broker address (localhost:9092), and specifies the path for the CSV file where the data will be stored.

It creates a Kafka consumer object to receive messages from the specified Kafka topic (BigData-console-topic).

The script enters a loop where it keeps polling for new messages from the Kafka topic.

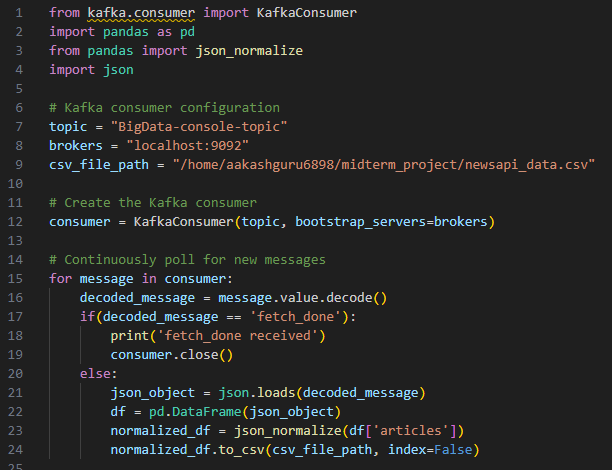
Each Kafka message is decoded from bytes to a string (decoded\_message = message.value.decode()).

If the decoded message is 'fetch\_done', then it means the producer—from the previous script—has signaled that the fetching process is done. It prints a message, closes the Kafka consumer, and exits the loop.

Otherwise, if not 'fetch\_done', then it assumes the message is JSON data representing news articles. The script proceeds to convert this JSON data into a Pandas DataFrame (df) and further normalize the JSON structure using json\_normalize. Then, it saves the resultant DataFrame in a CSV file specified by csv\_file\_path.

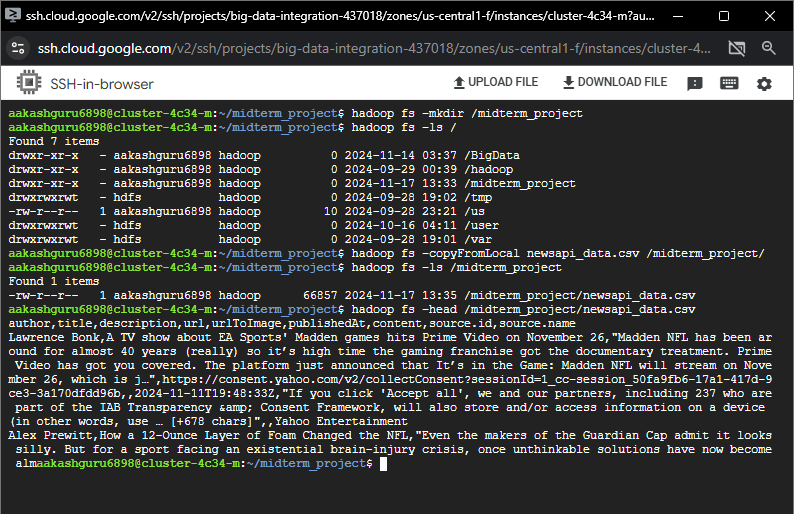
json\_normalize is used for nested JSON structures to be flattened into a flat table that can be easily represented in a DataFrame. Very common when dealing with APIs that return nested JSON.

This is the path of the CSV file where the script saves the normalized data.



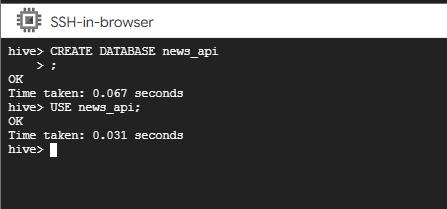
# Data Loading into HDFS

The transformed CSV file is moved into the Hadoop Distributed File System (HDFS) for efficient storage and scalability.



# Loading into Hive Table

The data in HDFS is loaded into a Hive table, allowing for structured querying and analysis. Hive provides a SQL-like interface for interacting with Hadoop.



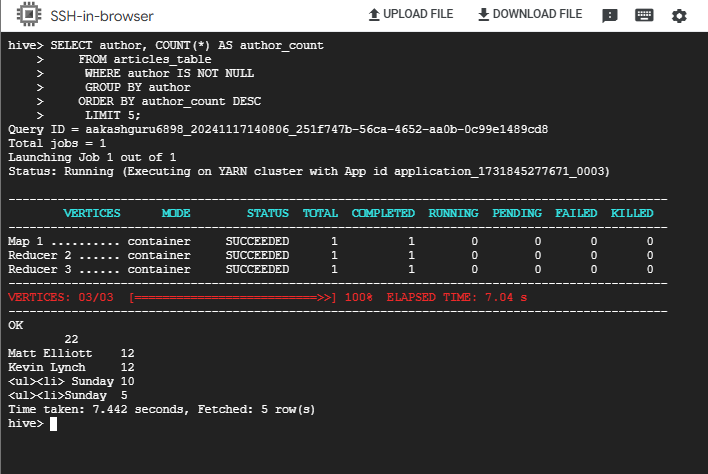
Creating table “articles\_table” and loading data into it



# Data Analysis

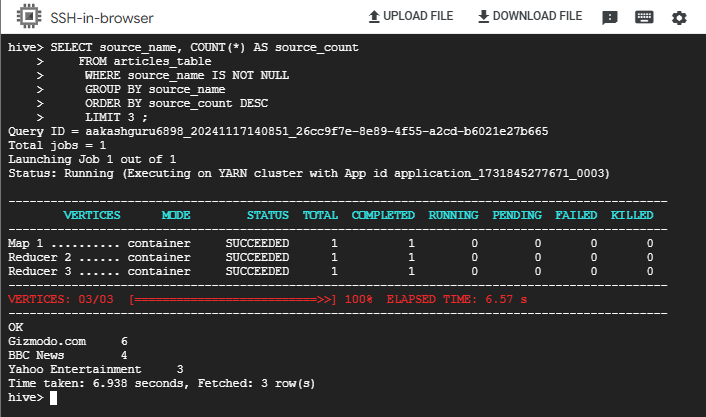
Queries are executed in Hive to analyze the loaded data. These queries could include aggregations, filtering, and other operations to derive insights.

## Query 1: Top 3 authors



* Finding prominent writers inside an article dataset.
* Examining the contributions made by authors to a platform or magazine.
* Learning about content trends, including the most active authors.
* Make editorial choices, such as giving well-known authors more prominence.

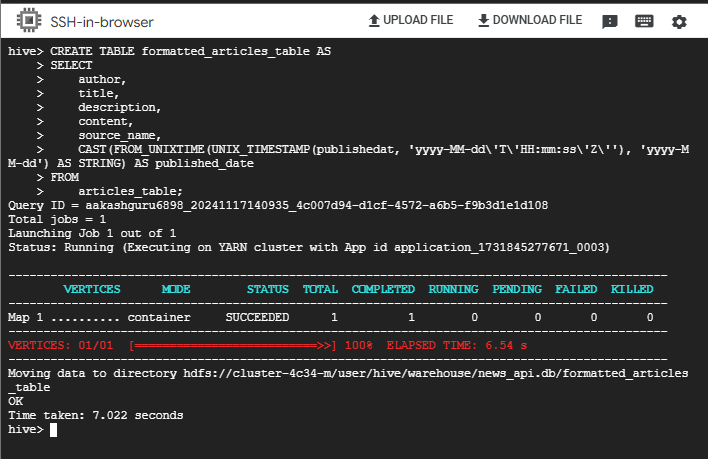
## Query2: Top 3 sources

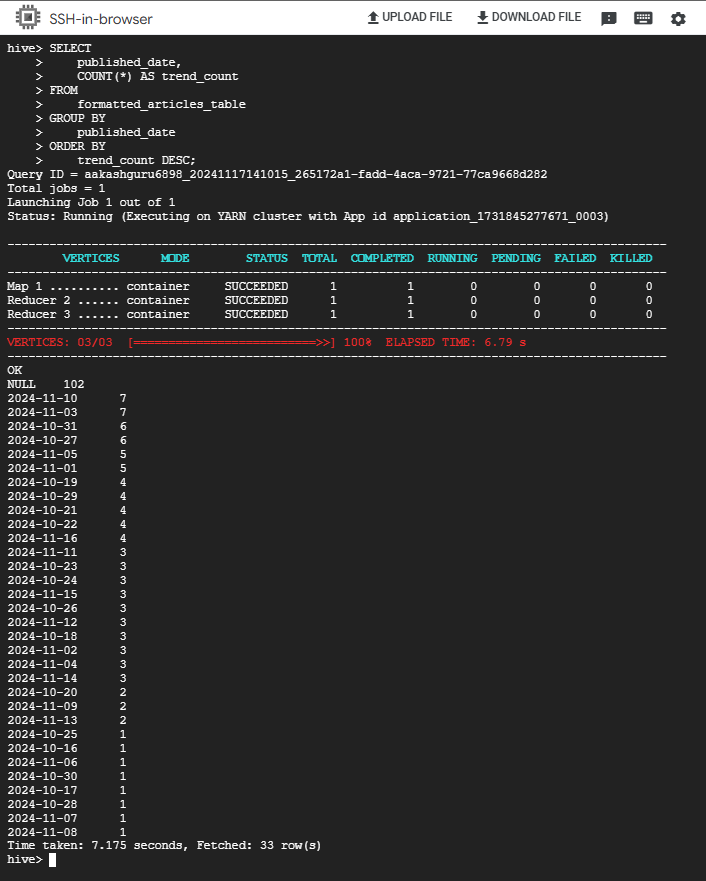


* Find out which sources contribute to articles the most frequently.
* When paired with engagement data, assist in determining which sources readers find most compelling.
* Allocate resources efficiently by prioritizing sources with high content volume.

## Query 3: Trend count as per published\_date

For this query we need to format published\_date to be unix timestamp. Creating formatted\_articles\_table by converting published\_date timestamp.





* Identify trends in publication behavior, such as the busiest days of the week or months.
* Determine high-activity dates to guide scheduling strategies in order to optimize reach or prevent content competition.

# Conclusion

The ETL pipeline has been successfully implemented, extracting, transforming, and loading data arriving from NewsAPI into HDFS for further efficient analysis using Hive. Kafka gives the guarantee of reliable, scalable streaming of data, and Hive supports SQL-like query-oriented data analysis.