# Jude Eze's Project Report:
# Real-Time Data Ingestion and Visualization using Kafka and CockroachDB

## 1. Introduction

### 1.1 Project Overview
This project involves the real-time ingestion, processing, and visualization of data from a Kafka stream, with a specific focus on **space object data**, such as that from the **Starlink satellite network.** The project employs Apache Kafka as the messaging queue to handle streaming data, while CockroachDB is used as the storage solution for persisting this data. Additionally, the project includes a web-based interface developed with Streamlit for real-time data visualization.

### 1.2 Motivation
The motivation for this project stems from the need to efficiently manage and visualize the continuous flow of real-time data from space-based assets. With the increasing number of satellites and the complexity of their operations, there is a need for robust systems that can ingest, store, and visualize large volumes of data in real-time. This project aims to provide a scalable solution that can be applied to various use cases in space operations, such as monitoring satellite health and performance.

## 2. Methodology

### 2.1 System Architecture
The architecture of the system is designed to handle real-time data ingestion, processing, and visualization. The key components of the system include:
- Apache Kafka: A distributed streaming platform used for building real-time data pipelines and streaming applications.
- CockroachDB: A distributed SQL database used to persist the streamed data for further analysis.
- Streamlit: An open-source app framework used to visualize the data in real-time.
- Psycopg: A PostgreSQL adapter for Python used to connect and interact with CockroachDB.

### 2.2 Data Flow
The data flow in the system is as follows:
1. Data Ingestion: Data from an external API (e.g., Starlink satellite data) is fetched periodically.
2. Kafka Producer: The fetched data is sent to a Kafka topic (`jude_topic`) via a Kafka producer.
3. Kafka Consumer: A Kafka consumer listens to this topic, processes the incoming messages, and flattens any nested JSON structures.
4. Database Storage: The processed data is then inserted into a CockroachDB database for persistence.
5. Data Visualization: The data stored in CockroachDB is visualized using Streamlit, providing real-time insights into satellite operations.

## 3. Implementation

### 3.1 Data Ingestion and Kafka Producer

The data ingestion component is responsible for fetching data from the external API and sending it to the Kafka topic. The key steps include:
- Fetching data using the `requests` library.
- Serializing the data into JSON format.
- Sending the serialized data to the Kafka topic using the Kafka producer.

### 3.2 Kafka Consumer and Database Storage

The Kafka consumer component listens to the `jude_topic` and processes each message by flattening any nested JSON structures before storing them in CockroachDB. The data is sanitized to remove any non-printable characters and ensure it meets the database schema requirements.

### 3.3 Data Visualization

The data visualization component is built using Streamlit. It provides real-time visualization of the satellite data, including orbital parameters, ground track maps, and velocity distributions. The visualization is dynamically updated as new data is ingested.

## 4. Results

### 4.1 Database Setup

The project successfully set up a database in CockroachDB with a schema designed to store key orbital parameters and other relevant data points for each satellite object.

### 4.2 Real-Time Data Ingestion

The system demonstrated the capability to ingest data in real-time from a Kafka topic and store it efficiently in CockroachDB. The data was processed to ensure it was in a usable format before storage, including flattening nested JSON structures and sanitizing text fields.

### 4.3 Data Visualization

The Streamlit-based visualization component provided real-time insights into the data. Various charts and maps were generated to help users understand the dynamics of the satellite network, including orbital parameters and ground tracks.

## 5. Conclusion

### 5.1 Summary
This project successfully implemented a real-time data ingestion and visualization pipeline using Kafka and CockroachDB. The system is capable of handling streaming data, processing it, storing it in a distributed database, and visualizing it in real-time. This project demonstrates the potential of combining modern streaming platforms with distributed databases and interactive web applications for managing and analyzing real-time data.

### 5.2 Future Work
Future enhancements could include the integration of more complex data processing algorithms, the scaling of the system to handle larger datasets, and the application of machine learning models to predict satellite behavior based on ingested data.

### 5.3 Lessons Learned
The project underscored the importance of proper data handling, especially when dealing with real-time streams. Effective error handling, data validation, and resource management are crucial to ensure the reliability and scalability of the system.