***Making Your Own Streaming Pipeline:***

***Kafka Learning Path Capstone***

**Overview:**

Welcome to your capstone project! You'll embark on an exciting journey to build a real-time data streaming application using **Apache Kafka** in **KRaft** mode. This project will empower you to:

* **Integrate with External APIs:** Choose an API that interests you and fetch data periodically.
* **Implement Producer-Consumer Architecture:** Develop a producer to send data to Kafka and a consumer to retrieve and visualize it.
* **Create Live Visualizations:** Use visualization libraries to represent data in real-time, gaining insights as data flows in.

By the end of this project, you'll have hands-on experience with real-time data processing, Kafka configurations, and dynamic data visualization.

**Project Structure:**

Your project will be organized into the following components:

1. **Docker Compose Configuration (docker-compose.yml):**
   * Sets up Kafka in **KRaft** mode without relying on ZooKeeper.
   * Manages Kafka services and their configurations.
2. **Producer Notebook (producer\_notebook.ipynb):**
   * Fetches data from your chosen API at regular intervals.
   * Sends the fetched data to a Kafka topic.
3. **Consumer Notebook (consumer\_notebook.ipynb):**
   * Consumes data from the Kafka topic.
   * Processes and visualizes the data in real-time.
4. **Supporting Scripts:**
   * **Kafka Setup Script (setup\_kafka.sh):** Automates Kafka initialization and topic creation.
   * **README File (README.md):** Provides an overview, setup instructions, and usage guidelines.

**Functional Requirements:**

To successfully complete this project, ensure that your application meets the following functional requirements:

**1. Docker Compose Configuration (docker-compose.yml)**

* **a. Kafka in KRaft Mode:**
  + Configure Kafka to run in **KRaft** mode by setting necessary environment variables.
  + Assign a unique CLUSTER\_ID and node.id to identify your Kafka cluster.
* **b. Port Mappings:**
  + Map Kafka listeners to appropriate ports (9092 for clients and 9093 for controllers).
* **c. Data Persistence:**
  + Use Docker volumes to persist Kafka data, ensuring data isn't lost between restarts.

**2. Producer Notebook (producer\_notebook.ipynb)**

* **a. API Integration:**
  + **Select an API:** Choose any public API (e.g., weather data, financial markets, social media feeds) that provides data you find interesting.
  + **Configuration:** Input your API key and set parameters such as city name, country code, or other relevant filters.
* **b. Data Fetching:**
  + Implement a mechanism to fetch data from the API at a specified interval (default: every 10 minutes).
  + Handle API authentication, request formation, and response parsing.
  + Incorporate error handling for network issues, API rate limits, and unexpected responses.
* **c. Kafka Producer Setup:**
  + Configure the Kafka producer with the correct bootstrap server address.
  + Ensure messages are sent to the designated Kafka topic with appropriate keys (optional).
* **d. Message Production:**
  + Serialize the fetched data into JSON format before sending to Kafka.
  + Implement callback functions to log successful deliveries or handle errors.
* **e. Logging:**
  + Display real-time logs indicating the status of message production.
  + Provide insights into the number of messages sent and any encountered issues.

**3. Consumer Notebook (consumer\_notebook.ipynb)**

* **a. Kafka Consumer Setup:**
  + Configure the Kafka consumer to connect to the same Kafka cluster and subscribe to the designated topic.
  + Handle consumer group management and message offset tracking.
* **b. Data Consumption:**
  + Continuously listen for new messages from Kafka.
  + Deserialize JSON messages into Python dictionaries or appropriate data structures.
* **c. Data Processing:**
  + Clean and preprocess data as necessary for visualization.
  + Implement any required data aggregation or transformation logic.
* **d. Real-Time Visualization:**
  + Create live-updating charts, graphs, or dashboards to represent incoming data.
  + Utilize visualization libraries such as **Matplotlib**, **Plotly**, or **Bokeh** for interactive visuals.
  + Optionally, integrate with **Dash** or **Streamlit** for more advanced dashboarding capabilities.
* **e. Logging:**
  + Display real-time logs indicating the status of message consumption.
  + Provide insights into the number of messages consumed and any processing issues.

**4. Supporting Scripts and Documentation**

* **a. Kafka Setup Script (setup\_kafka.sh):**
  + Automate the initialization and configuration of Kafka using Docker Compose commands.
  + Include commands to start Kafka services and create necessary topics.
* **b. README File (README.md):**
  + Provide a comprehensive overview of the project.
  + Include setup instructions, usage guidelines, and troubleshooting tips.
  + Document any dependencies and how to install them.

**Resources:**

**Key Code:** [**https://github.com/JUMP-TA/KafkaExercises/tree/main/KafkaCapstone**](https://github.com/JUMP-TA/KafkaExercises/tree/main/KafkaCapstone)