Background

You are working for a company which deals with real-time data processing and analytics. The requirement is to build a real-time streaming data pipeline using Kafka and Docker. The pipeline should be capable of ingesting streaming data, process in real-time, and storing the processed data into a new Kafka Topic.

Requirements:

1. Set up a local development environment using Docker, including Kafka and any other necessary components for the pipeline.
2. Design and implement a Kafka consumer (in any preferred language) to consume data from a Kafka topic and perform some basic processing on the data (e.g., transforming the data, aggregating, or filtering based on certain conditions) and identify some interesting insights, if possible.
3. Configure another Kafka Topic and store the processed data in the new topic
4. Ensure that the pipeline can handle the streaming data continuously and efficiently, handling any error messages or missing fields

Instructions:

1. Use Docker Compose to set up the necessary Docker containers for Kafka and other components. Instructions to get setup for the assignment are mentioned below.
2. Clearly explain your design choices, data flow, and how you ensure the pipeline's efficiency, scalability, and fault tolerance in a Readme.

Project Setup

Have Docker installed locally. Use the docker compose below to get setup with Kafka locally. The docker compose file includes a data generator which produces data to a topic named ‘user-login’. Kafka can be accessed from your machine through port 29092 (bootstrap-server = localhost:29092). The data generator is producing messages in the schema below. Messages include a timestamp field, which corresponds to the time it was created by the data generator.

Sample Message

{\"user_id\": \"424cdd21-063a-43a7-b91b-7ca1a833afae\", \"app_version\": \"2.3.0\", \"device_type\": \"android\", \"ip\": \"199.172.111.135\", \"locale\": \"RU\", \"device_id\": \"593-47-5928\", \"timestamp\":\"1694479551\"}

Docker Compose

Unset

```
version: '2'
services:
  zookeeper:
    image: confluentinc/cp-zookeeper:latest
    environment:
      ZOOKEEPER_CLIENT_PORT: 2181
      ZOOKEEPER_TICK_TIME: 2000
    ports:
      - 22181:2181
    networks:
      - kafka-network

  kafka:
    image: confluentinc/cp-kafka:latest
    depends_on:
      - zookeeper
    ports:
      - 9092:9092
      - 29092:29092
    networks:
      - kafka-network
    environment:
      KAFKA_BROKER_ID: 0
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
      KAFKA_ADVERTISED_LISTENERS:
LISTENER_INTERNAL://kafka:9092,LISTENER_EXTERNAL://localhost:29092
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP:
LISTENER_INTERNAL:PLAINTEXT,LISTENER_EXTERNAL:PLAINTEXT
      KAFKA_INTER_BROKER_LISTENER_NAME: LISTENER_INTERNAL
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
      KAFKA_TRANSACTION_STATE_LOG_REPLICATION_FACTOR: 1
      KAFKA_TRANSACTION_STATE_LOG_MIN_ISR: 1

  my-python-producer:
    image: mpradeep954/fetch-de-data-gen
    depends_on:
      - kafka
    restart: on-failure:10
    ports:
      - 9093:9093
    environment:
      BOOTSTRAP_SERVERS: kafka:9092
      KAFKA_TOPIC: user-login
    networks:
      - kafka-network

networks:
  kafka-network:
    driver: bridge
```

Additional Questions

For this assignment an ounce of communication and organization is worth a pound of execution. Please answer the following questions on next steps:

1. How would you deploy this application in production?
2. What other components would you want to add to make this production ready?
3. How can this application scale with a growing dataset?

All done, now what?

Upload your codebase to a public Git repo (GitHub, Bitbucket, etc.) and please submit your Link where it says to - under the exercise via Green House our ATS. Please double-check this is publicly accessible.

Please assume the evaluator does not have prior experience executing programs in your chosen language and needs documentation to understand how to run your code.

Evaluation Criteria

Your submission will be evaluated based on the following aspects:

1. Proper setup of Docker containers for Kafka and other components.
2. Accurate design and implementation of the Kafka consumer/Producer for real-time processing.
3. Well-documented code, clear instructions, and explanations of design decisions.

Candidates should aim to showcase their understanding of Kafka, Docker, and real-time streaming concepts while developing a functional and efficient data pipeline. They should also demonstrate good coding practices, documentation skills, and the ability to make appropriate design choices.