SOFE 4630 Winter 202 - Cloud Computing

Project Milestone#2 : Data Ingestion Software

Course Group No: 12

Group Members:

Ashwin Shanmugam - 100700236
Faazil Shaikh - 100707829
Evans Mosomi - 100719552
Samuel Rantetoding - 100694161

1. **What is EDA? What are its advantages and disadvantages?**
    - The architecture that utilized a broker or event in order to decouple microservices
    - **Advantages:**
        - High scalability, as it is handled using the logic of distributed systems, means the data can be spread across multiple storage.
        - Data stream processing in real time, which could maximize response time of the application in order to solve a complex problem.
        - Decouple services based on the needs of the user.
        - Extensibility is high due to the event storage and replay.
        - Eliminate the need for constant polling of events, thereby reducing the network bandwidth usage.
    - **Disadvantages:**
        - Difficulty in testing the system in EDA environment
        - High complexity
        - Can have inconsistencies depending on the nature of the event trigger

2. **In Kafka, what's meant by cluster, broker, topic, replica, partition, zookeeper, controller, leader, consumer, producer, and consumer group?**

    - Cluster: consists of at least 3 brokers in order to allow the broker to provide enough redundancy, commit messages to disk and also be responsible for fetching requests.
    - Broker: A server located inside the Kafka cluster.
    - Topic: Category in which the data are stored or processed for publishing.
    - Replica: multiple copies of the data that are split across multiple servers.
    - Partition: Break a single topic into multiple topics and store them in separate nodes in the cluster.
    - Zookeeper: a place to store offsets and provide availability to the system.
    - Controller: responsible for administrative operations, like assigning partition and monitoring failures.
    - Leader: A broker that is chosen to own the partition in the cluster.
    - Consumer: Application that utilized the data from Kafka.
    - Producer: create a new message and send it to a specific topic.
    - Consumer group: consists of multiple consumers that work together to consume a topic and assign each partition to only one member of the consumer group in order for horizontal scaling.

3. Follow the following video to install Kafka into your local machine and create topics, consumers, and producers using Kafka's built-in tools.
    - Video demo link: https://drive.google.com/file/d/1Lbg0C61gF9y4MrWXaC2zbYrfcbg-7fNw/view?usp=sharing

5. Follow the following video to generate NodeJS scripts for creating topics, consumers, and producers

- Use the docker application from the repository not that shown in the video.
- To use the docker application from the repository, Kafka brokers' addresses will be localhost:9093,9094, and 9095.
- Follow only the NodeJS scripts not the docker commands.
- when the partition is not specified, Kafka will decide the partition randomly or by hashing the key.

- Video demo link: https://drive.google.com/file/d/1fAABCriWfAivmbNye5VrvnjRVKYuBfXD/view?usp=sharing

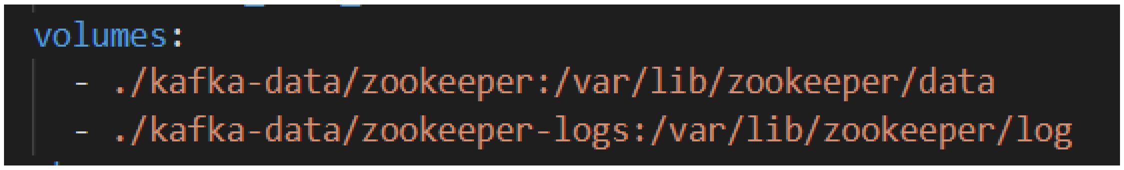
6. Using the python library Kafka-python, write three python scripts that

- Create a topic
- Produce messages on a topic. The message's key and the partition should be optional. Also, an error message should be displayed in the case of failure.
- Repeatedly consume messages from a topic and display the consumed messages whenever available. The group id should be optional.

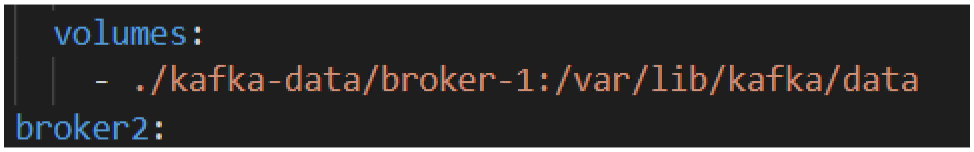
- Video demo link: https://drive.google.com/file/d/1CkAb53mfsKSKjbGZWcEEik8qjEWwhcaO/view?usp=sharing

4. **Prepare a video showing the codes that generated topics, produce messages, and consume them in both NodeJS and python. Your video should display the possible producer and consumer scenarios.**
    - Demo links are in questions number 5 and 6

5. **A problem in the used YAML file to create the docker images is that the data inside Kafka clusters are not persistent which means if the docker images are down, all its messages are lost. Update the YAML file for persistent data (hint: it's related to the volume options in Kafka brokers and zookeeper). Describe how this update solves the problem.**



```
volumes:
  - ./kafka-data/zookeeper:/var/lib/zookeeper/data
  - ./kafka-data/zookeeper-logs:/var/lib/zookeeper/log
```



```
  volumes:
    - ./kafka-data/broker-1:/var/lib/kafka/data
broker2:
```

6. **Follow the following video about Kafka in Confluent Cloud and use the shown CLI to create a topic, consumer, and producer. Also update your python code to create a topic, consumer, and producer using Kafka in Confluent Cloud (hint: only the connection information of Kafka Cluster has to be updated). Record a video illustrating those tools and showing them in action.**
    - Video demo link: https://drive.google.com/file/d/1Gd1wQWa1A3iaYuWLRCVMPXuYb50QY_UB/view?usp=sharing

7. **Upload the used files, reports, and videos (links) into your repository.**
    - Repository Link: https://github.com/ashwin1609/CloudComputing-Milestone2