SOFE 4630U
Cloud Computing

Project Milestone #2
2022/02/15

Data Ingestion Software and Kafka Clusters

Sabesan Sivakumar(100701928),
Jerusha Macwan(100723319),
Matthew English(100704553),
Pranjal Saloni (100653360)

https://github.com/PranjalS1/CloudComputingPM2/tree/main

1. What is EDA? What are its advantages and disadvantages?

Event-driven architecture is a software architecture paradigm promoting the production, detection, consumption of, and reaction to events.EDA follows the publish and subscribe model and is loosely coupled as each publisher(event) does not know who will subscribe(consume).

Advantages:

- Processing real-time streaming data
    - Data does not need to wait to be processed at a later time
- Scalability
    - Loosely Coupled: each publisher(event) doesn't know which subscriber(consumer) they are listening to.

- Fault tolerance:Since its event driven each event does not rely on each other meaning if a event were to fail it will only affect that event and not the entire system

Disadvantages:

- Duplicated events
    - Singular events can trigger duplicate messages across multiple services if not careful
- Error handling
    - Extensive monitoring tools are needed to track hundreds or thousands of message brokers that are constantly passing and receiving events.

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

Cluster: A Kafka cluster consists of one or more servers (Kafka brokers) running Kafka.

Broker: The broker Handles requests from clients and replicates data within the cluster.

Topic: The topic is used as the categories to organize all the messages the application gets. Each topic has its own unique name that is known throughout the entire Kafka cluster. Each message that has been sent will contain the unique name to which that topic will only be able to be sent to and read from that message.

Replica: As you may assume by the name, it just means to have multiple copies of the same data and place it all across multiple servers so that if a server fails we can still retrieve the data from the application.

Partition: A single log file where records are written to.

Zookeeper: Tracks the status of nodes in Kafka clusters and maintains a list of topics.

Controller: One of the brokers in the cluster, acts as the controller, handling the partition and replica states as well as administrative activities such as reallocating partitions. Only one controller server is active in the cluster at any given moment.

Leader: One server serves as the leader of each partition, while others function as followers. Leader is in charge of all read-write queries for the partition, while the followers duplicate the leader.

Consumer: The consumer is what subscribes to a topic. Consumers will be able to read and process the event they have subscribed too.

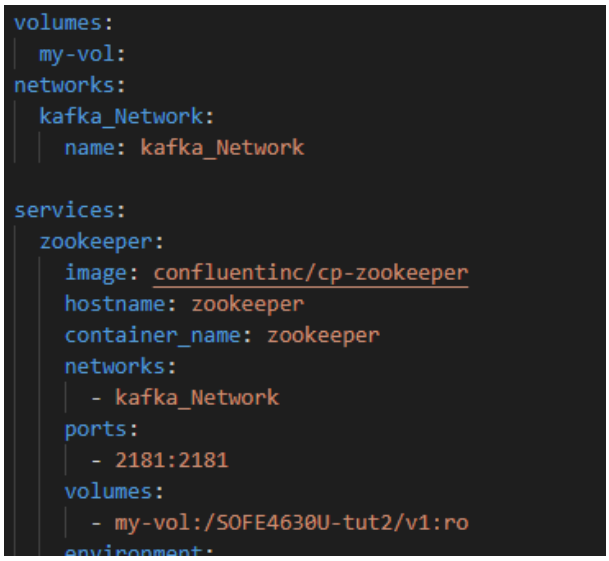
Producer: The producer is what produces(Writes) to an event in Kafka.

Consumer Group: a group of consumers that divide the partitions of the topics among themselves.

3. 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.

https://drive.google.com/file/d/1BJOMMtZe5bZYAWCkGuA2SPILeqz99rNg/view?usp=sharing

4. 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.



```yaml
volumes:
  my-vol:
networks:
  kafka_Network:
    name: kafka_Network

services:
  zookeeper:
    image: confluentinc/cp-zookeeper
    hostname: zookeeper
    container_name: zookeeper
    networks:
      - kafka_Network
    ports:
      - 2181:2181
    volumes:
      - my-vol:/SOFE4630U-tut2/v1:ro
    environment:
```

By creating a volume and entering its name and path into the yml file within zookeeper and the brokers. It will create and use the new volume to store data for the containers to enable persistent data.

5. 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 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.

https://drive.google.com/file/d/1Ecp_qyN_CF5410lGjBJVgL_Ocd0Ci7o_/view?usp=sharing