SOFE 4630U: Cloud Computing

Dr. Mohamed El-darieby

Date: February 15th, 2022

Project Milestone #2

Github: [Yusuf-shaik/Cloud-Lab-2 (github.com)]

Videos: [Watch Videos]

COURSE GROUP #1	
Shanjay Kailayanathan	100624670
Jana Kanagalingam	100603975
Tolu Elebute	100724471
Yusuf Shaik	100655451
Yakhneshan Sivaperuman	100703400

<u>What is EDA? What are its advantages and disadvantages?</u>

EDA is Event-Driven Architecture. It's a design pattern that allows for detecting of events and acting upon them in real time. It's used to facilitate communication between decoupled micro-services within a system.
In EDA, you don't need to store the data. We can just put our data in a database inside each microservice and we can use the stream of events to keep it up to date. An event is any action taken by the system (update status, new tweet, update page, add item to cart, remove item from cart).

Event sourcing: converting events into a data model

Streaming: processing windows of the event stream

Advantages: Fault tolerance, loose coupling & reduced technical debt, reduced integration complexity, high scalability, easy configurability, high throughput, custom based disk retention, "fairly low latency"

Disadvantages: Duplicated events, lack of clear workflow order, naming convention confusion, error handling & troubleshooting, high learning curve, not optimal for very low latency systems

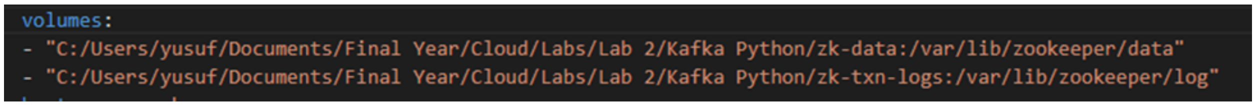
<u>In Kafka, what's meant by cluster, broker, topic, replica, partition, zookeeper, controller, leader, consumer, producer, and consumer group?</u>

A **broker** is a middleware between publishers and subscribers (also known as consumers and producers. It is a server running on kafka inside a cluster. **Clusters** contain one or more brokers. Brokers form a cluster. In clusters there is always one broker responsible for cluster controllers. **Controllers** are responsible for partitions and monitoring for broker failures. **Producers** publish data to a topic, and **consumers** subscribe to that topic, receiving information from said topic. The purpose of a broker is to decouple the consumers from their producers. Consumers belong to a specific consumer group. **Consumer groups** make sure each partition is consumed by one member of a consumer group so it can be scaled horizontally.

Brokers contain multiple partitions, a **partition** contains either part of, or a whole topic (partitions cannot contain multiple topics). A topic is essentially split up across multiple partitions and the partitions are split across multiple brokers so that multiple clients do not decrease the throughput of the system. These split partitions are referred to as **replicas**. Partitions contain **leaders** which are the "parent" of the follower replicas. Producers send updates to this leader, and the replicas will request updates from the leader based on their configured "heartbeat".

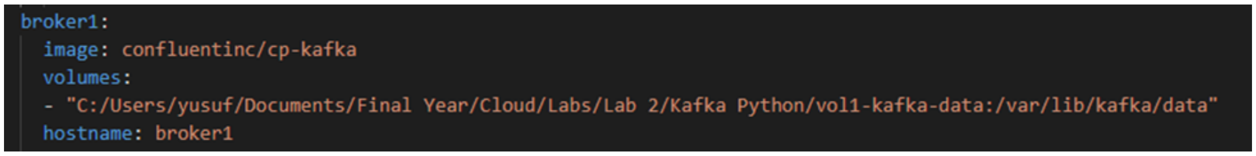
8. persistent data in yaml file:

Based on the docker documentation, created directories for zookeeper, and 1 directory for each broker. Mounted the absolute path for the host to the relative path on the container.

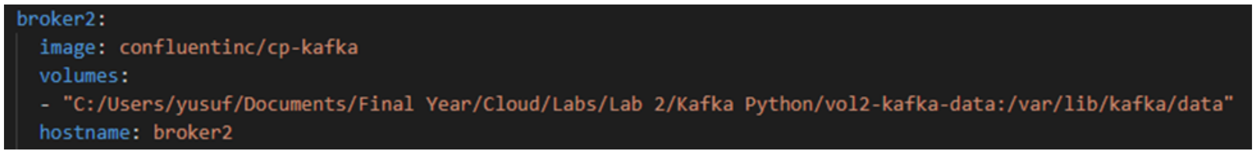


```yaml
volumes:
- "C:/Users/yusuf/Documents/Final Year/Cloud/Labs/Lab 2/Kafka Python/zk-data:/var/lib/zookeeper/data"
- "C:/Users/yusuf/Documents/Final Year/Cloud/Labs/Lab 2/Kafka Python/zk-txn-logs:/var/lib/zookeeper/log"
```

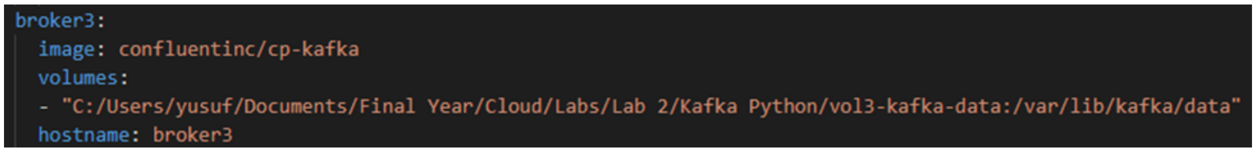
And followed these similar steps for the remaining brokers:



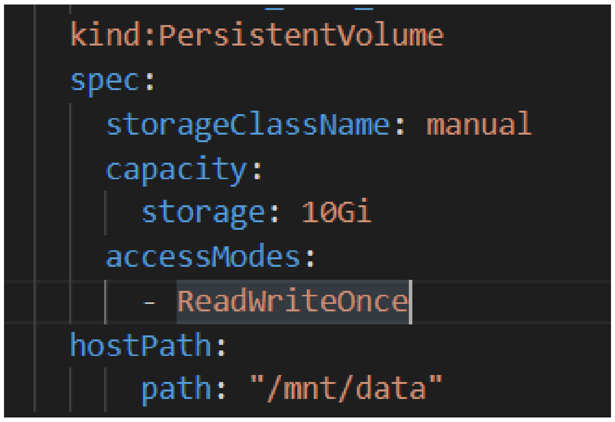
```yaml
broker1:
  image: confluentinc/cp-kafka
  volumes:
  - "C:/Users/yusuf/Documents/Final Year/Cloud/Labs/Lab 2/Kafka Python/vol1-kafka-data:/var/lib/kafka/data"
  hostname: broker1
```



```yaml
broker2:
  image: confluentinc/cp-kafka
  volumes:
  - "C:/Users/yusuf/Documents/Final Year/Cloud/Labs/Lab 2/Kafka Python/vol2-kafka-data:/var/lib/kafka/data"
  hostname: broker2
```



```yaml
broker3:
  image: confluentinc/cp-kafka
  volumes:
  - "C:/Users/yusuf/Documents/Final Year/Cloud/Labs/Lab 2/Kafka Python/vol3-kafka-data:/var/lib/kafka/data"
  hostname: broker3
```



```yaml
  kind:PersistentVolume
  spec:
    storageClassName: manual
    capacity:
      storage: 10Gi
    accessModes:
      - ReadWriteOnce
  hostPath:
      path: "/mnt/data"
```

This solves the problem of containers not containing information after the container has been deleted. It creates a persistent location on the host, so that even when the container is removed, the data still remains. Every event that occurs within zookeeper and each broker is stored in a local copy on the host machine as well, so that if the container starts again, the new data will be read from these folders described in the volumes when they were mounted. Updating the volume options in the yaml file provides persistence to the data as kafka is, afterall, architected for persistent storage.