SOFE 4630U: Cloud Computing
February 15, 2022
Group 11 - Group Report
Project Milestone - Data Ingestion Software -- Kafka Clusters
Fajer Zayed (100672347), Ireni Ruthirakuhan (100657302), Raveenth Maheswaran (100704540), Yale Wang (100673933)

GitHub Link: https://github.com/fzayed/Project-Milestone-Group-11.git

---

What is EDA?

- Software design pattern
- Can contain data associated with the even or it can just be a notification that something happened
- Data doesn't need to be stored → instead it is stored in a database inside each microservice
    - Use stream of events to keep it up-to-date
- Event is the base of what is called event driven architecture (EDA)
- EDA has three components: producer, broker, consumer
    - Where each component of the application's architecture has the ability to complete jobs/tasks independently
- A producer creates the events that are going to be redirected by a broker into the right consumers
- The consumers will react to this event and execute the things that they need to execute
- Also referred to as publish/subscribe model
    - Obtaining of events (consumers) and production of events (producers)
- Can be communicated in the form of an event notification
    - An event is something that happens and is immutable/change from original state
    - Information/data is sent whenever the specific said events has occurred
        - Data/Information is stored in a database within a microservice

What are their advantages and disadvantages?

- Advantages include:
    - Processing of data in real time (publication of events in real-time)
        - Don't need to wait, meaning real-time updates/decisions/actions
    - Independency
        - Don't need to worry about actions excluding their own (producers and consumers viewpoint)
            - Includes services (and their status) used by other parties (producers/consumers)
    - Scalable
        - Can have a sub to an event when needed
    - Modifiability
        - Creation/deletion of events
    - Reliability
        - In the case where a component (broker) goes down, the system has the ability to recover and remain running
        - If a service crashes/stops the topic store offsets → can be picked up on where it had previously stopped
- Disadvantages include:
    - Repetition of response towards an event may be doubled/+

        - Due to their independence no components are aware of the actions taken by one another
        - Duplication of an event is also possible
    - Cannot easily find issues/errors with their inability to provide an analysis for the root cause of the problem
        - Traceback/debugging difficulty
        - Monitoring change/distribution of data is difficult
    - Security compromised
        - extensive/complex approaches can be used to fix some security issues
        - Producers and consumers are always in action so it is hard to keep track of data points moving around

Kafka Cluster

- Kafka runs in a cluster consisting of one or more servers
- Contains brokers, where each individual broker represents itself as an individual broker
    - Can obtain multiple servers simultaneously (usually three)

Kafka Broker

- A single Kafka server within the cluster
- Responsible for:
    - Receiving of messages from the producer
    - Assigning offsets
    - Committing messages to disk
    - Responding to consumers requests
    - Sending messages to consumers
- Has a predefined size which could be used for retention. It has the ability to expire/delete data/information from oldest to newest

Kafka Topic

- There can be multiple topics within a single broker.
    - The messages are sent to a specific topic where the producer specifies which topic is written to
- A topic presents a way to categorize data that is being sent
- A topic can be further broken down into partitions
- Data stored within each topic is only in `append only`
    - Data can be removed depending on specified constraints. These include time and memory space/storage

Kafka Replica

- A copy of data stored in a separate server

- Kafka replicas are like partitions
- Ensures availability in case of server failure
- A log of which replicas are available to cover which data helps organize the system
- Once a broker cannot function, a replacement replica server helps take its place

Kafka Partition

- A way of distributing work
- Each topic is divided into partitions
- Each such division contains a number of records under that topic
- Data is appended to a partition
- Recorded by a single log
- Improves scalability
    - More partitions = more throughput
    - More consumers can each use one partition
    - A consumer can be more instances, using disparate partitions

Kafka Zookeeper

- Knows which brokers are active
- Elects controller
- Manages configuration of the topics and partitions
- Offsets are stored here
- A consumer can stop and restart w/o losing its position in the stream of data

Kafka Controller

- It is represented by one broker within a group of broker
- A broker is responsible for administrative operations. These include assigning partitions to brokers and the monitoring of broker failures

Kafka Leader

- The leader of a partition is represented by a single broker within a cluster
    - In the case that the leader has a failure then another broker can obtain that leadership title
    - Different brokers can be leaders/followers for the partitions. This helps for when a broker fails/goes down since they all do not depend on the same leader broker

Kafka Consumer

- Pulls messages off of a Kafka topic
    - Can only read committed messages
- Subscribes to 1/+ topics and read messages/data in the order that it has been produced

- Is a part of a consumer group
- Keeps track of position within the stream of data as it remembers what offset had already been consumed.

Kafka Producer

- Pushes messages into a Kafka topic
    - Sent as a request to publish some data to a specific partition within a specific topic where the broker specifies published position within partition
- Thread safe
- Contains queues of records that haven't been pushed to the server
- Each of these records has its own buffer space
- These buffers are organized by records intended for each partition

Kafka Consumer Group

- Consumers within a group work together to consume a topic (parallel processing, acts like a queue).
- Ensures that each partition is only consumed by one member within the group
- In the case that a consumer fails, the consumer group will rebalance the partition to make up for the missing/failed member
- Can only consume the same message multiple times if its for different consumer groups
    - Meaning that the partition that is consumed by a consumer group member is not available for the other group members to view. In the case where there is only one consumer, that consumer can consumer all partitions from the broker

Video - Kafka built-in tools

https://drive.google.com/file/d/1HG8bVK3W84dO1TqV7ZCOSSLRsC0puiEb/view?usp=sharing

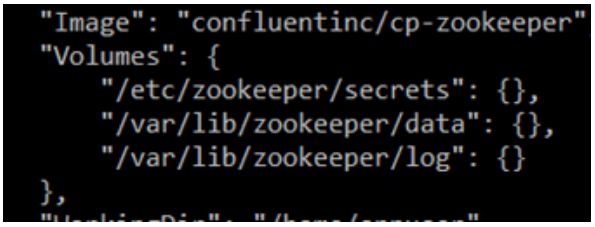
Video 1 (nodejs scripts)

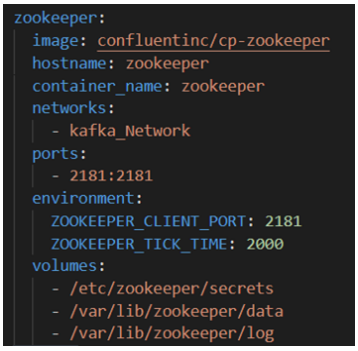
https://drive.google.com/file/d/1oNhnacUY_CoUIDyULohSakU0ivIKLvbJ/view?usp=sharing

Video 2 (kafka-python package using python scripts)

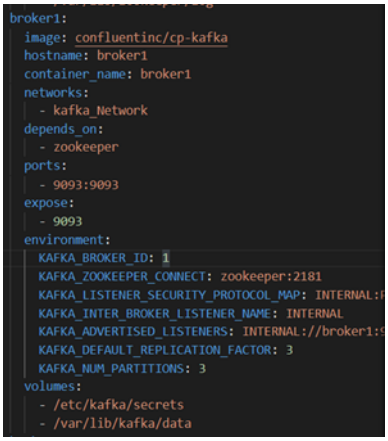
https://drive.google.com/file/d/1OpbNe0jXDCHVGV8I66FO4OsGCCsBQaKf/view?usp=sharing

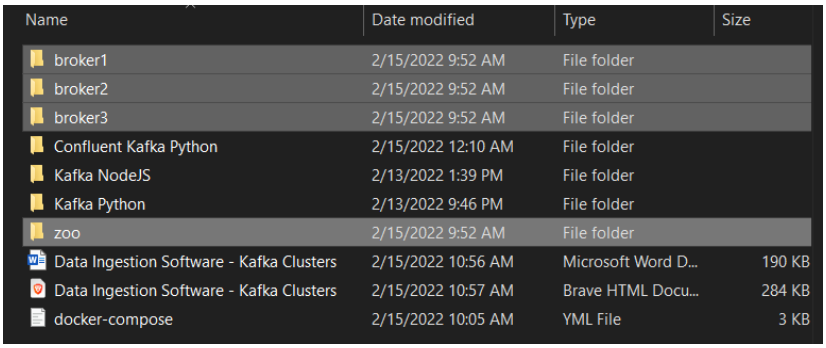
Why should the YAML file allow for persistent data?
Zookeeper:





Brokers:





Name	Date modified	Type	Size
broker1	2/15/2022 9:52 AM	File folder	
broker2	2/15/2022 9:52 AM	File folder	
broker3	2/15/2022 9:52 AM	File folder	
Confluent Kafka Python	2/15/2022 12:10 AM	File folder	
Kafka NodeJS	2/13/2022 1:39 PM	File folder	
Kafka Python	2/13/2022 9:46 PM	File folder	
zoo	2/15/2022 9:52 AM	File folder	
Data Ingestion Software - Kafka Clusters	2/15/2022 10:56 AM	Microsoft Word D...	190 KB
Data Ingestion Software - Kafka Clusters	2/15/2022 10:57 AM	Brave HTML Docu...	284 KB
docker-compose	2/15/2022 10:05 AM	YML File	3 KB

- It mounts directories at the said path within the specific container
    - To identify the containers volume `docker inspect container_name` is run
- In the case that the same path is specified as the target and container path, the mounted path is taken into consideration instead. Therefore this means that changes remain for the time that the host directory is mounted to the container

Video 3 (Confluent Cloud CLI)

https://drive.google.com/file/d/11BaDij7sXLog0QER9nthbKFz7rHTlgtj/view?usp=sharing

Video 4 (confluent-kafka package using python scripts)

https://drive.google.com/file/d/1qhNb1oW2X04rLu5PPUstrwGPG1KtzGbd/view?usp=sharing