Lab/Project 2: Kafka Clusters
Ireni Ruthirakuhan (100657302)

What is EDA?

- Software design pattern
- Where each component of the application's architecture has the ability to complete jobs/tasks independently
    - These applications don't need to wait for a response to a request regarding the pub/sub of events
        - Events are referred to as a statement that something has occurred/changed from its original state.
    - This means that there is also a separation b/w the obtaining of events (consumers) and production of events (producers)
        - Meaning consumer just focuses on obtaining events and the producers just focus on the production of events
- Has the ability to send information/data whenever the specific said has events occured
- 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
- For example, Kafka uses an event driven architecture

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
- Contains topics
    - Divided into segments where each of the segments are replicated across the kafka broker
- Contains brokers
    - 1 Brokers = 1 kafka server
    - Can have multiple servers (usually 3 to provide enough redundancy)

Kafka broker
- Responsible for
    - receiving messages from the producer
    - Assigning offsets
    - Committing messages to disk
    - Responding to consumers requests
    - Sending messages to consumers
- Has a set size/ defined time for how long it can be used for (retention)
    - Expires/deleted from oldest to newest
- Default port (9092)

Kafka topic
- Within a broker (can be multiple topics within a broker)
- Messages are sent to a specific topic
    - Producer specifies which topic is written to
- A way to categorize data that is being sent
- Can be broken down into partitions
- Data stored within the topic is only in append only
    - Data can be removed depending on specified constraints
        - Time
        - Space

Kafka replica
- Partition may be assigned to multiple brokers
    - Result in partition being replicated
        - Redundancy in messages within the partition

Kafka partition (sharding)

- Each partition acts as a separate commit log
- Order of messages are guaranteed across the same partition
- Each partition can be read by a different consumer
    - Higher scalability and throughput
- Is Always owned by a single broker within a cluster
- Multiple partitions within a single topic

Kafka zookeeper

- Offsets are stored here
- Consumer can stop and restart w/o losing its position in the stream of data

Kafka controller

- Cluster controller
    - 1 broker within a group of brokers
- A broker that is responsible for administrative operations
    - Assigning partitions to brokers
    - Monitoring for broker failures

Kafka leader

- Always owned by a single broker within a cluster
    - Single broker that owns it = leader of the partition
- Another broker can take leadership if there is a broker failure
    - Different brokers can be leaders/followers for the partitions
        - This helps for when a broker goes down/fails
- All producers and consumers operating on that partition should connect to that leader

Kafka consumer

- Usually other applications consuming/reading data/messages from kafka
- Subscribe to 1/+ topics and read messages/data in the order produced
- Keeps track of position within the stream of data
    - Remembers what offset was already consumed
        - Offsets correspond to a specific message in a specific partition
- Belongs to consumer group
- Can only read committed messages

Kafka producer

- Are usually other applications that is producing data
- Creates new messages/data and sends it to a specific topic (within the cluster)
    - Produces messages/content/data to the broker
    - Sent as a request to publish some data to a specific partition within a specific topic
        - Broker specifies published position within partition

Kafka consumer group

- Consumers within a group work together to consume a topic (parallel processing, acts like a queue)
    - Consumes messages/content/data from the broker
- Ensures that each partition is only consumed by 1 member within the group
    - Horizontal scaling of consumers
        - The partition consumed by a consumer group member is not available for other group members to view
    - If there is only 1 consumer, they can consume all needed partitions from the broker → acts like a pub/sub
- In the case that a consumer fails, the consumer group will rebalance the partition to make up for the missing member
- Can only consume the same message multiple times if its for different consumer groups
    - Polls information from position of a specific position of a specific topic

Video Running NodeJS
https://drive.google.com/file/d/1q73NCQOtLIsF-UmUV_7evgU9ZJusx-Ji/view?usp=sharing

Video Running Kafka-Python (scripts)
https://drive.google.com/file/d/1OpbNe0jXDCHVGV8I66FO4OsGCCsBQaKf/view?usp=sharing

Update the YAML File to allow for persistent data

- Updating the YAML file to include volumes displays solutions to a variety of issues
- 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 Running Kafka on Confluent Cloud CLI
https://drive.google.com/file/d/1sTtNk8iciJZwQLFgCMU_swAHGc6MjrGE/view?usp=sharing

Video Running Kafka Confluent Cloud w/ Python Scripts
https://drive.google.com/file/d/1qhNb1oW2X04rLu5PPUstrwGPG1KtzGbd/view?usp=sharing

References

1. https://solace.com/what-is-event-driven-architecture/
2. https://docs.divio.com/en/latest/reference/docker-docker-compose/
3. https://github.com/confluentinc/confluent-kafka-python