Project Milestone

Data Ingestion Software-- Kafka Clusters

Name: Mihir Patel
Student Number: 100702168
Date: January 31, 2022

Procedure:

1. **Watch the first 27 minutes from the following video that describes the concepts of EDA**
    - https://youtu.be/OqN6x_vNsds
2. **Watch the following video that describes the main components of Kafka as a core of EDA**
    - https://youtu.be/JalUUBKdcA0

Answer the following questions:

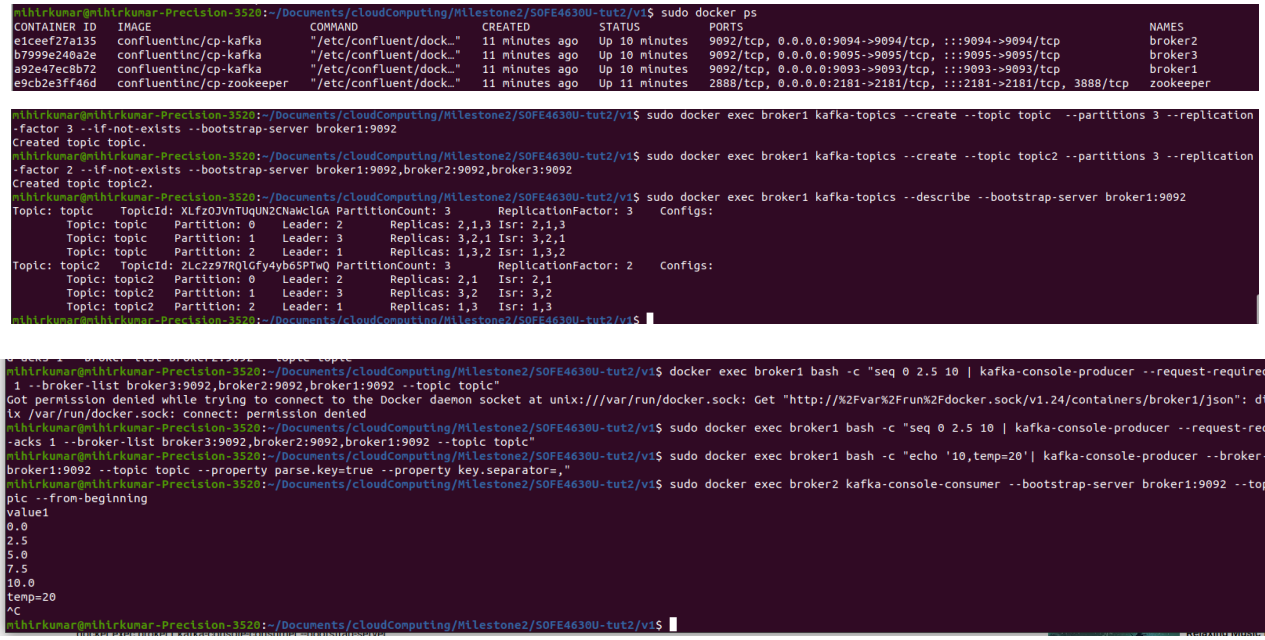
3. **What is EDA? What are its advantages and disadvantages?**

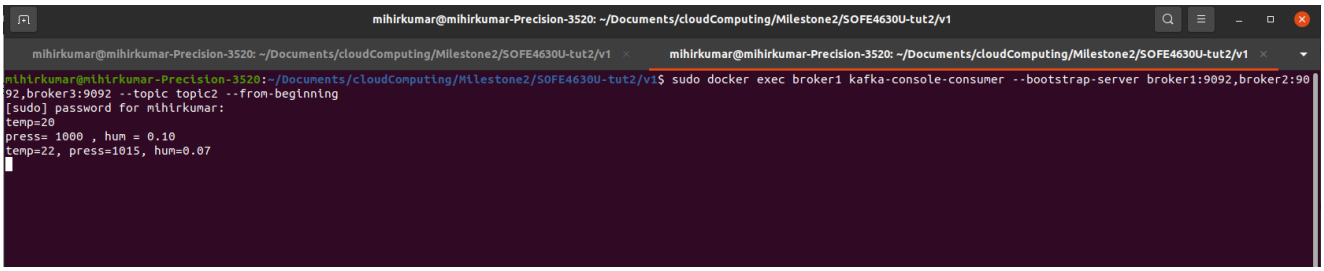
    - EDA stands for Event driven architecture.and it is mostly used in modern applications that are built using microservices. In this architecture, the system incorporates the customers/user data in real-time to archive a flexible system that can adapt to changes and make decisions.
    - Some of the advantages of using EDA is it allows the system to be loosely coupled which frees the system from being dependent on other components of the system. It also provides services like fault tolerance which keeps the system running in case of application failure.Some other common advantages are security, scalability and granularity.
    - The drawbacks of using EDA is that the design of the application can be a bit more complex than required. For example a simple call between API requires the system to establish a good interface structure which required more cost then needed. EDA makes it hard to resolve errors.
4. **In Kafka, what's meant by cluster, broker, topic, replica, partition, zookeeper, controller, leader, consumer, producer, and consumer group?**

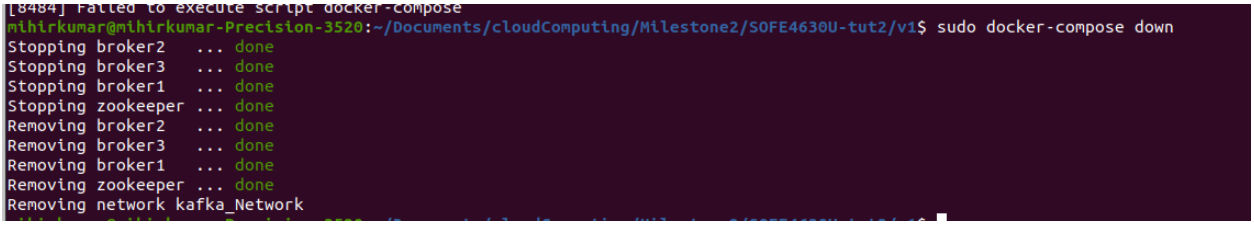
    - **Topic** - It is a message queue that stores the key-value pair as the form of data. The data is stored for seven days as the default time but can be changed depending on the usage. The topic name is unique within a cluster.
    - **Cluster-** kafka cluster is a set of either servers or a broker.
    - **Partition**- As the name suggests it partitions the data into multiple partitions such that it can make the distributed system more scalable. The data is stored to its unique partition with the help of consistent hashing.

- **Consumers**- A consumer task is to consume data that is assigned to the same topic that it belongs to. There is no limit to the number of consumers within a system which then can be subscribed to one or more topics.
- The **controller** is a broker; its job is to monitor the health of all the other brokers and coordinate partition leader election on broker failover. Each Kafka cluster is responsible for electing a controller which is responsible for managing partitions, replicas and other duties.
- **Zookeeper** is responsible for checking the status of the kafka broker and informs the user if any changes are observed in the broker cluster.
- **Replicas**- Each partition of data is stored in a broker that is connected to the system. The broker system has its own partitions data and other partition replica data.
- **Broker**- Brokers are the machines that are used for distributing data to partitions.
- **Producers**- Producer is a program that writes to the topic and is a client application and is usually written by a developer.

5. **Follow the following video to install Kafka into your local machine and create topics, consumers, and producers using Kafka's built-in tools.**
    - https://youtu.be/Kx0o2jeMB0o
    - Following are the image for installation of Kafka in the local machine







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

- https://youtu.be/R873BlNVUB4?t=2424
- Use the docker application from the repository not 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

7. **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.

8. **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.**
    - Below is the video link that demonstrates the use of topic, producer and consumer in a kafka environment in NodeJS
        - https://drive.google.com/file/d/1razX2EZ2IkiF8ppe6fDNsybGN2QEy8P1/view?usp=sharing
    - Below is the video link that demonstrates the use of topic, producer and consumer in a kafka environment in Python
        - https://drive.google.com/file/d/1razX2EZ2IkiF8ppe6fDNsybGN2QEy8P1/view?usp=sharing

9. **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.**

10. **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.**
    - https://youtu.be/vllYRz65tgA
    - https://drive.google.com/file/d/1yvdF-vvAYNlhqo88ijS203_Y_TUfWJTU/view?usp=sharing

11. **Upload the used files, reports, and videos (links) into your repository.**