SOFE 4630U: Cloud Computing

Date: January 25, 2023

Project Milestone 1: Data Ingestion System (Apache Kafka)

Group T4
Name: Preet Patel
Student Id: 100708239

Links:

Github:
https://github.com/preetpatel87/Cloud-Computing-Project-Group-T4/tree/main/Project%20Milestone%20Kafka%20Clusters/Preet%20Patel

Demonstration Links:

- **Confluent Kafka:** https://drive.google.com/file/d/1pKsSMtpwO3Q1TIQnK9J9HkFc289b4b2E/view?usp=sharing
- **Google Pub/Sub:** https://drive.google.com/file/d/1IhIVMnEkdgzr2ou_YWxkt3vU6BRennGa/view?usp=sharing

Discussion:

What is EDA? What are its advantages and disadvantages?

Event driven architecture is a software architecture consisting of producers that generate a stream of event and consumer components that consume the generated events. It is an architecture built around producing, consuming, processing and storing events. An event can be a message or a log produced by an action on the producer. Consumers are other services or applications that consume these events to trigger an action on the consumer side. Events can be communicated between the producer and consumer by the publisher and subscriber pattern or the producer can create a log of events that the consumer can access to replay the event. Communication can be targeted or conversational between the producer and consumer.

Advantages of event driven architecture is that it is highly scalable as more consumers can always be added to consume the events generated by the producer, it makes the system more resilient and maintainable by decoupling the producer and consumer, and it allows to create responsive applications that trigger actions based on a stimuli in the system. However, event driven architecture also has drawbacks such as increased complexity to the producer and consumer components, and also introduces difficulties with troubleshooting and debugging bugs.

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

Kafka is a distributed system used for event streaming and publisher subscriber messaging. Kafka consists of producer and consumers similar to the above discussed producers and consumers in event driven architecture. Producers are applications or services that produce events or messages that are required and consumed by consumers which are also applications or services that require and consume the produced events/messages. A broker is a kafka server which facilitates the communication between the producer and subscriber. A collection of kafka brokers working together forms a cluster. Clusters enable Kafka to distribute the workload amongst different brokers. Kafka topics are kafka streams that live on the broker which hold and organize the messages/events produced by producers. Kafka producers send messages to a topic while consumers will consume the events/messages from the topic that is required. It allows the producer to categorize the events so consumers can only receive the messages that are relevant to the consumer. Kafka topics can be partitioned so that the workload of a broker can be distributed making kafka scalable. Events and messages can be categorized in different partitions by using specific key values. These partitions can be replicated to ensure that the data is never lost. Such replicated partitions are called replicas. The zookeeper is responsible for keeping track of all the brokers in a cluster, the topics and partitions, electing leaders, etc. Zookeeper facilitates the

synchronization between different brokers in the cluster allowing Kafka to act as a distributed system. The Zookeeper elects a leader for replicas so that read and write functions can only be performed against the leader. The replicas that are not elected leaders are followers and are responsible for keeping the most updated version of the data. In the event the leader fails, one of the followers is elected as leader to take their place. A consumer group in Kafka allows multiple consumers to work as a group to distribute the consumption load similar to how Kafka distributes the produced events and messages into partitions.

Design:

Google has an alternative ingestion tool called Google Pub/sub. Configure it and create a topic using Cloud interface. write a python code for a Google Pub/sub consumer and producer. The producer and the consumer should act as the smart meter shown before.

- **Video Demonstration:** https://drive.google.com/file/d/1IhIVMnEkdgzr2ou_YWxkt3vU6BRennGa/view?usp=sharing
- **Python Code:** https://github.com/preetpatel87/Cloud-Computing-Project-Group-T4/tree/main/Project%20Milestone%20Kafka%20Clusters/Preet%20Patel/google%20pubsub