Project Design Document

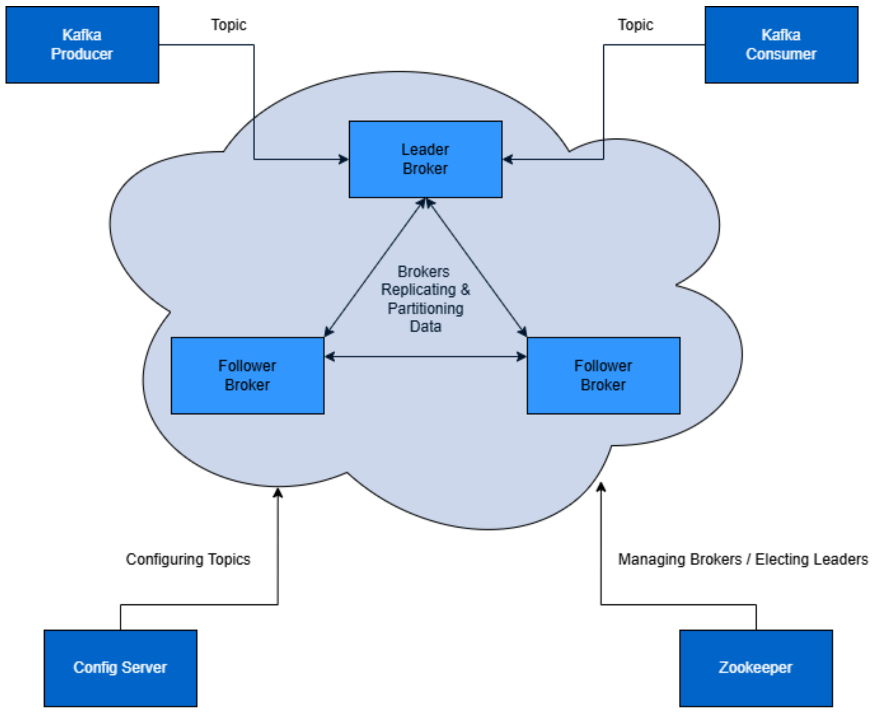
Topic : Kafka: a Distributed Messaging System for Log Processing(Paper)
Course : Software and Data Engineering
Group : 12
Authors : Anadi Sharma (B21CS008) Drishti Agrawal (B21CS027) Yatharth Shukla (B21CS080)

Architecture Overview

The architecture consists of the following primary components :

- **Kafka Producer** : Responsible for sending data to the Kafka cluster.
- **Kafka Cluster** : The core of the system, consisting of brokers (leader and followers) that manage data replication and partitioning.
- **Kafka Consumer** : Responsible for consuming data from Kafka topics.
- **Config Server** : Configures topics within Kafka.
- **Zookeeper** : Manages brokers, helps in leader election, and maintains metadata for the Kafka cluster.

The diagram illustrates how these components interact to ensure data flows seamlessly across the system.



Components

1. Kafka Producer

The Kafka Producer is responsible for publishing messages to a specific Kafka topic. It acts as the data source, sending data into the Kafka cluster for processing.

2. Kafka Cluster

The Kafka Cluster consists of multiple brokers that store data across partitions. The cluster architecture includes:

A. **Leader Broker**: Manages read and write requests for a specific partition. Each partition in a topic has one leader broker, which coordinates data replication among follower brokers.
B. **Follower Brokers**: Replicate data from the leader broker to ensure high availability and fault tolerance. In the event of a leader broker failure, one of the followers can be elected as the new leader.
C. **Replication and Partitioning**: Data is distributed across brokers, with each partition replicated to provide redundancy. This replication allows for consistent and fault-tolerant data handling.

3. Kafka Consumer

Consumers read data from specific Kafka topics. Each consumer can subscribe to one or more topics and process the data as it becomes available, enabling real-time data processing.

4. Config Server

The Config Server is used to configure topics within Kafka. It enables administrators to create, update, or delete topics and adjust configurations such as retention policies and partition counts.

5. Zookeeper

Zookeeper manages the Kafka brokers and handles leader election. It maintains metadata about the cluster, including broker configurations, topic configurations, and leader information for each partition.

Data Flow

1. **Producing Data**
   The Kafka Producer sends messages to a designated Kafka topic. The topic is partitioned, and the producer distributes data across partitions based on the partitioning strategy.

2. **Data Replication**
   Within the Kafka cluster, brokers replicate data to ensure fault tolerance. The leader broker for each partition handles read and write requests, while follower brokers store copies of the data.

3. **Consuming Data**
   Kafka Consumers subscribe to topics and process the data. Each consumer can read data from multiple partitions, allowing for parallel data processing.

4. **Topic Configuration**
   The Config Server enables administrators to manage topics, adjusting settings such as the number of partitions and replication factors to optimize performance.

6. **Broker Management and Leader Election**
   Zookeeper manages the brokers and performs leader election to ensure high availability. If a leader broker fails, Zookeeper promotes a follower broker to become the new leader.

Fault Tolerance and Scalability

The system is designed to handle broker failures gracefully through Zookeeper-managed leader elections. Kafka’s partitioning and replication mechanisms enable horizontal scalability, ensuring that the system can handle increasing data loads.