Digital Assignment – 2 & 3

Name: Namra Maniar
Registration No.: 20MIS1054

Apache Kafka

Introduction:

The Apache Software Foundation created the open-source distributed streaming framework known as Apache Kafka. It was developed specifically to manage the streaming and processing of real-time data for applications on a massive scale.

In its most fundamental form, Kafka is a distributed messaging system. This kind of system enables many applications and services to connect with one another by sending and receiving streams of data, which are referred to as messages. These messages might be in a variety of forms, including JSON, Avro, or binary, for example.

Features:

A comprehensive look at the features of Apache Kafka:

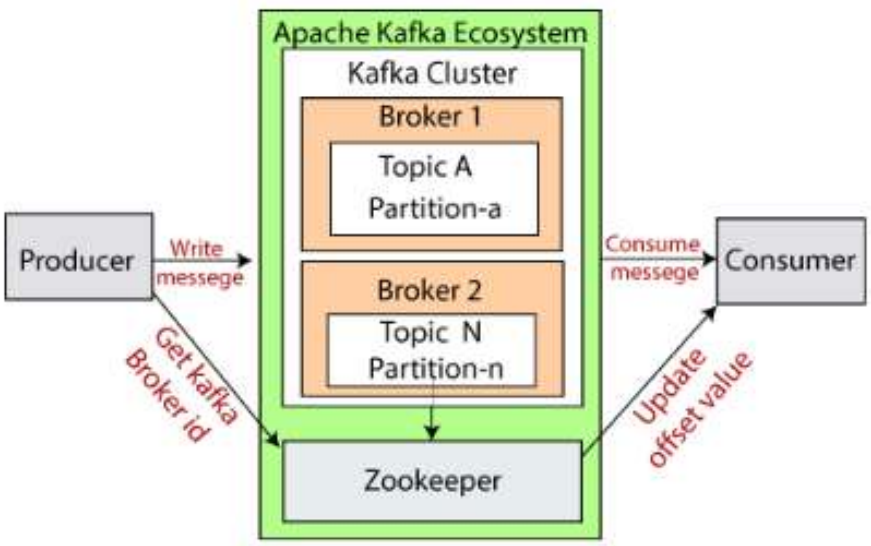
- Distributed: It implies that it may be expanded horizontally over several workstations, data centres, or even cloud providers. Since Kafka was created as a distributed system, it has been open source since 2014. Because of this feature's capacity to tolerate errors and provide high availability, data is guaranteed to be readily accessible at all times.
- Fault-Tolerant: Kafka was built from the ground up to be fault-tolerant, which means that it can continue to function normally even if some of its components stop working. It employs replication to retain multiple copies of data across various nodes, which ensures that data is always accessible even if a node fails, and it does this by spreading the data out among numerous nodes.
- High-Throughput: Kafka's architecture makes it possible to process large amounts of data in real time. Since it is capable of processing millions of messages per second, it is well suited for use cases that demand a high

throughput, such as the aggregation of logs, the consumption of data, and the processing of streams.

- Low-Latency: Since Kafka was developed with low-latency processing in mind from the beginning, it enables users to process data in real time. Since it can analyze messages in milliseconds, it is perfect for use cases that need low-latency, such as monitoring in real time, fraud detection, and real-time analytics.
- Scalability: Kafka was developed to be very scalable, which enables it to manage massive amounts of data and support an increasing number of users. Scalability also enables Kafka to serve a rising variety of applications. It is possible to grow it horizontally over numerous computers, which enables it to deal with ever-increasing amounts of work.
- Message Retention: Kafka has customizable message retention rules, which allows you to determine how long messages are held in the system. These policies may be found under the "Message Retention" section. This functionality is helpful for use cases that need the data to be stored for an extended period of time, such as compliance and auditing.
- Security: Kafka has a number of security measures, including as authentication, authorization, and encryption, to guarantee that data is kept private and is only accessible by people who have been given permission to see it.
- Many Client APIs: Kafka gives you the ability to connect Kafka with a broad variety of different applications and systems by providing numerous client APIs. These client APIs include Java, Python, and C/C++.
- Connections: Kafka includes connectors that make it possible to quickly integrate Kafka with other systems, such as databases, message queues, and data lakes. Connectors may be found in the Kafka documentation. This feature streamlines the process of integrating data and makes it simple to construct data pipelines.
- Stream Processing: Kafka gives you the ability to process and analyze data in real time by providing stream processing capabilities, which can be accessed via Kafka Streams and other third-party tools. This capability makes it possible to do real-time analytics, machine learning, and other applications that are data-driven.
- Monitoring: Kafka gives you the ability to monitor the health and performance of your Kafka cluster by providing monitoring capabilities. These capabilities come in the form of metrics, logs, and other tools.

Because of this capability, preventative maintenance and problem-solving may be performed at any time.

Architecture:



Kafka architecture's fundamentals:

- Topics: Messages are posted to topics. Scalable and fault-tolerant topics may be partitioned.
- Producers: Producers publish Kafka messages. They transmit synchronously or asynchronously.
- Consumers: Subscribers read subject messaging. Load balancing and fault-tolerance consumers might include them.
- Brokers: Kafka brokers shop and manage messages. They receive, shop, and deliver communications.
- Zookeeper: Zookeeper controls and maintains the Kafka cluster. It manages partition leaders, brokers, and configuration information.
- Kafka's architecture is built for real-time data processing with high scalability, throughput, latency, and fault-tolerance.

Application in real life:

Apache Kafka has numerous real-life applications in various industries due to its ability to handle large volumes of data and provide real-time data streaming. Some of the most common use cases for Kafka are:

- Data Pipelines: Kafka is widely used in data pipelines to transfer data between different systems and applications in real-time. This includes ETL (Extract, Transform, Load) processes, where data is extracted from different sources, transformed, and loaded into a target system.
- Real-time Analytics: Kafka's ability to handle large volumes of data in real-time makes it a popular choice for real-time analytics. It is used to collect and process data from various sources, such as web applications, IoT devices, and social media platforms.
- Log Aggregation: Kafka is used for log aggregation in applications where log data needs to be collected and analyzed in real-time. This includes applications like web servers, application servers, and database servers.
- Messaging Systems: Kafka can be used as a messaging system for various applications, such as chat applications, notification systems, and email systems.
- Event-Driven Architectures: Kafka is an ideal choice for building event-driven architectures, where events trigger actions in real-time. This includes applications like fraud detection, real-time recommendations, and IoT applications.

Conclusion:

In conclusion, Apache Kafka is a highly scalable, fault-tolerant, and distributed message streaming platform that provides real-time processing of large volumes of data. It publish-subscribe model, along with topics, producers, consumers, brokers, and Zookeeper, forms the foundation of its architecture. With its ability to handle massive amounts of data, Kafka has become a popular choice for use cases like data pipelines, real-time analytics, log aggregation, and event-driven architectures. It offers excellent performance, high throughput, and low latency, making it an ideal choice for companies that require fast and reliable data processing at scale.