Apache Kafka

Open-source distributed event streaming platform

Index

About Kafka

Open Source

Kafka's open-source nature promotes collaboration and innovation, making it popular for organizations of all sizes.

Distributed

Kafka's distributed design enables scalability and fault tolerance across clusters, ensuring consistent performance..

Event Streaming

Kafka facilitates real-time processing of events, enabling applications to react instantly to data streams for analytics and event-driven architectures.

Installing Apache Kafka

1. Download Apache Kafka from https://kafka.apache.org/downloads

2. Extract it and save it to the desired location.

```
$ tar -xzf kafka_2.13-3.7.0.tgz
$ cd kafka_2.13-3.7.0
```

3. Start the Kafka environment.

```
# Start the ZooKeeper service
$ bin/zookeeper-server-start.sh config/zookeeper.properties
```

```
# Start the Kafka broker service
$ bin/kafka-server-start.sh config/server.properties
```

4. Create a topic to store events

$ bin/kafka-topics.sh --create --topic quickstart-events --bootstrap-server localhost:9092

5. Write some events into the topic.

   $ bin/kafka-console-producer.sh --topic quickstart-events --bootstrap-server localhost:9092

6. Read the events.

   $ bin/kafka-console-consumer.sh --topic quickstart-events --from-beginning --bootstrap-server localhost:9092

Basics of Apache Kafka

Events

Events are individual units of data in Kafka, representing records or messages. They can be any type of data, such as log entries, sensor readings, or user actions, and are organized into topics within Kafka.

Topics

Topics are logical channels or categories in Kafka where events are published. Each topic is identified by a name and acts as a feed of events that can be subscribed to by producers and consumers.

Brokers

Brokers are individual Kafka server instances responsible for storing and managing topic partitions. They handle message storage, replication, and serving consumer requests. A Kafka cluster typically consists of multiple brokers for fault tolerance and scalability.

Partitions

Partitions are the basic unit of parallelism and scalability in Kafka. They allow data to be distributed across multiple servers in a Kafka cluster, enabling horizontal scalability and fault tolerance. Each partition is an ordered, immutable sequence of records.

Producers

Producers are applications that publish data or messages to Kafka topics. They are responsible for creating records and sending them to Kafka brokers for storage. Producers can specify which topic to publish to and optionally specify the partition or key for the message.

Consumers

Consumers are applications that subscribe to Kafka topics and process the data or messages stored in them. They read data from Kafka partitions and can be part of a consumer group to achieve parallel processing and load balancing. Consumers track their offset (position) in each partition to maintain their progress in reading messages.

Example

We'll offer a practical example or use case to illustrate how Apache Kafka can be utilized in real-world scenarios.

Thank You

Arya Tiwari BT20CSE103
Deep Panchani BT20CSE105
Anukalp Pandey BT20CSE144