Apache Kafka Introduction

What is Kafka?

Kafka is a data streaming platform, just like Amazon Kinesis it is able to take in message and store messages durably for consumers to read in those messages.

It is very scalable because of the three different components, producers, brokers, and consumers. There is also built-in data replication to be a durable platform.

Core concepts

Messages or a record in Kafka is how you would be sending the data from the producer to the broker. The record consists of key and body. The key is used to identify which partition of a topic to write the record into.

A collection of message or record is referred to as a batch.

Kafka topic

A Kafka topic can be think of as a container that you can deposit your message / record into in the Kafka ecosystem and be read on a later time. Topics themselves are identified by a unique name, and the messages are sent to and read from a specific topic.

Kafka partition

Within a topic, it can have one or more partition, further division of a topic. When a topic is created the controller (head of the cluster) will determine how many partition a topic will have.

So a topic can be think of as a container to deposit your message into, then partition is more smaller boxes within the container to further organize your message.

Records will be stored into a topic based on the provided name, then based on the provided key which will be hashed in order to get the actual partition the record will be placed into.

Example

For example, if we have a topic name **activity-log** with three partition named **activity-log-1, activity-log-2, activity-log-3.**

Then when a source system publishes messages to the topic it will be stored into either of those partition.

Kafka broker and cluster

A Kafka broker is the server that handles the request from both Producer, Consumer, and Metadata queries. They are also responsible for keeping the data replicated within a cluster.

A Kafka cluster just have multiple Kafka broker in it handling requests to keep Kafka platform running, that is it.

Zookeeper

Zookeper is another set of servers that is responsible for metadata maintenance. They are able to direct producer the brokers to contact if there are multiple brokers.

Producer

The producer are processes that publishes records into Kafka topic via broker.

Consumer

A consumer are processes that pulls records off a Kafka topic via broker.

How are replication done?

In Kafka replication is implemented at the partition level. The redundant partition in a topic is called a replica. Each partition (that actually gets message published into) in a topic usually have one or more replicas associated with them.

Within a partition, partition and replica partition, there is the leader which handles all the read-write operations for the specific partition. Then the replicas will be replicating the leader partition.

If the leader fails, then one of the replicas will be promoted as the leader to take over.

What is a bootstrap-server

A bootstrap-server is a url for one of the Kafka brokers that allows you to fetch initial metadata about your Kafka clusters. Which topics are available and the number of partitions within each of the topics, which partition is a leader.

Producer or consumer will use these metadata to produce and consume from the appropriate topic / partition / contact the right broker.

There can be multiple bootstrap-server for failover purposes just in case one of them goes down.

Schema registry

Kafka at its core only transfer data in bytes, the data stored in topics are in raw bytes format, when you publish or consume from topic it is read in as raw bytes format. The consumer needs to know about the type of data the producer is sending in order to deserialize it later on (Get it back as an Object, how to transfer complex objects like a Linked list across network - Use serialization sent the object as array of bytes). Producer serializes the data using library like Avro in order to store it into raw bytes.

This is where Schema registry comes into play, it is an application that lives outside of Kafka cluster and handles distribution of schemas to the producer and consumer by storing the schema (layout of the object how to deserialize) in its local cache.

The producer before sending the data to Kafka, first check with registry to see if the schema is available, if not sent it and registry will cache it. Then it will serialize the data with the schema and send it to Kafka with a schema ID.

When the consumer gets the message, it will first get the schema from registry with the ID, and then deserialize it according to the schema.

The schema basically tells the consumer HOW to deserialize the bytes, what bytes constitute the first field, second bytes, and so on.

Revision #1
Created 2023-04-19 18:56:21 UTC by Tamarine
Updated 2023-04-19 19:58:23 UTC by Tamarine