Message

Topic

A categorized feed

Producer

Processes that publish messages to a topic

Consumer

Process that subscribes to a topic and processes a feed of published messages

Broker

A server within the Kafka cluster

Each broker is configured using schematic kafka configuration

Feed

A subscribable sequence of messages

Log segment file

TODO

Java consumer

Scala consumer

Offset commit

Offset fetch

Distribution

Refers to various aspects of Kafka which are spread out across multiple servers.

A Kafka cluster may include multiple brokers, each located on separate servers.

A topic may be divided into multiple partitions

Each partition may be replicated across multiple brokers.

Each partition has a lead broker which is responsible for servicing all reads and writes for that partition.

Leadership is distributed across brokers so that load is balanced

Consumer thread

A unit of parallelism

Offset

The position within a partition that a consumer is established to be reading.

Controlled by the consumer

Usually incremented sequentially as messages are read, however may be set in any order.

A consumer may set an “older” offset to consume messages previously processed.

Rebalancing

Process of reestablishing parallelism constraints when consumer threads enter or leave a group.

Consumer group

Abstraction that encapsulates both queuing and publish / subscribe

A group may consist of multiple individual consumers

Each member of a group is assigned the same group name

Each message published to a topic to which a group is subscribed is only delivered to a single consumer in the group.

There cannot be more consumer group members than there are partitions.

A logical representation of an application

Multiple groups can read the same topic concurrently

Ordering guarantees

Message ordering is guaranteed within a single partition

When there are multiple consumers in a single group, message delivery is essentially load balanced across the consumers.

There are no ordering guarantees where a consumer is receiving messages from multiple partitions.

The only way to guarantee message ordering across all messages in a topic is to have only a single partition.

Consumer load balancing

Distribution of message delivery across consumer members of a consumer group

Partition

Supports parallel processing by ensuring that all messages stored in a single partition are delivered in order to only a single consumer.

An ordered commit log where within which each message is assigned an offset indicating the order of the message within the log.

A topic may consist of multiple partitions.

Allows a topic to scale beyond the size of a single storage device.

Each partition in a cluster is replicated across a configurable number of nodes

It is up to the client to dictate which partition a certain piece of data is written to.

Log retention

The duration for which a message published to a partition is guaranteed to be available to be consumed.

Fetch position

Fetch request

Dead consumer

broker node registry

/brokers/id

Offset vector

TODO

Fault tolerance

Supported by partition replication.

Locality

Assumption that all of a group of data is stored in a single partition

Replica

a node responsible for maintaining a copy of a partition’s data, regardless of whether it is a leader, or even alive

in-sync replica

A replica which is up to date with the leader

Offset manager

Broker in a kafka cluster responsible for managing consumer offsets for a consumer group.

Lead broker

Consumer API’s

There are several consumer API’s available

Scala vs Java

Old

High-level consumer connector

Low-level simple consumer

New

Java consumer api

High-level zookeeper-based

Low-level

High-level Consumer

Doesn’t care about managing offsets

Just cares about consuming data

Stores offset read from a particular partition in Zookeeper

Hides details of what has been consumed from the client

SimpleConsumer

Used when you need more control over message consumption as a consumer

TODO

Simple Consumer

TODO

Message Delivery Semantifs

### Consumer

consumer offset