High-level Architecture

This lesson gives a brief overview of Kafka’s architecture.

We'll cover the following

Kafka common terms#

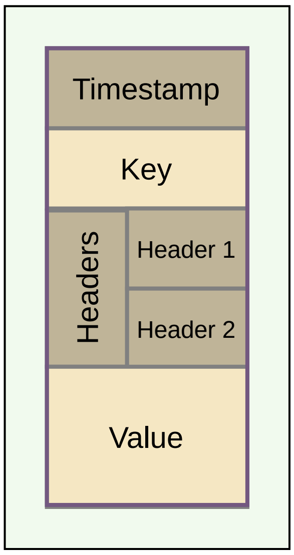
Before digging deep into Kafka’s architecture, let’s first go through some of its common terms.

Brokers#

A Kafka server is also called a broker. Brokers are responsible for reliably storing data provided by the producers and making it available to the consumers.

Records#

A record is a message or an event that gets stored in Kafka. Essentially, it is the data that travels from producer to consumer through Kafka. A record contains a key, a value, a timestamp, and optional metadata headers.



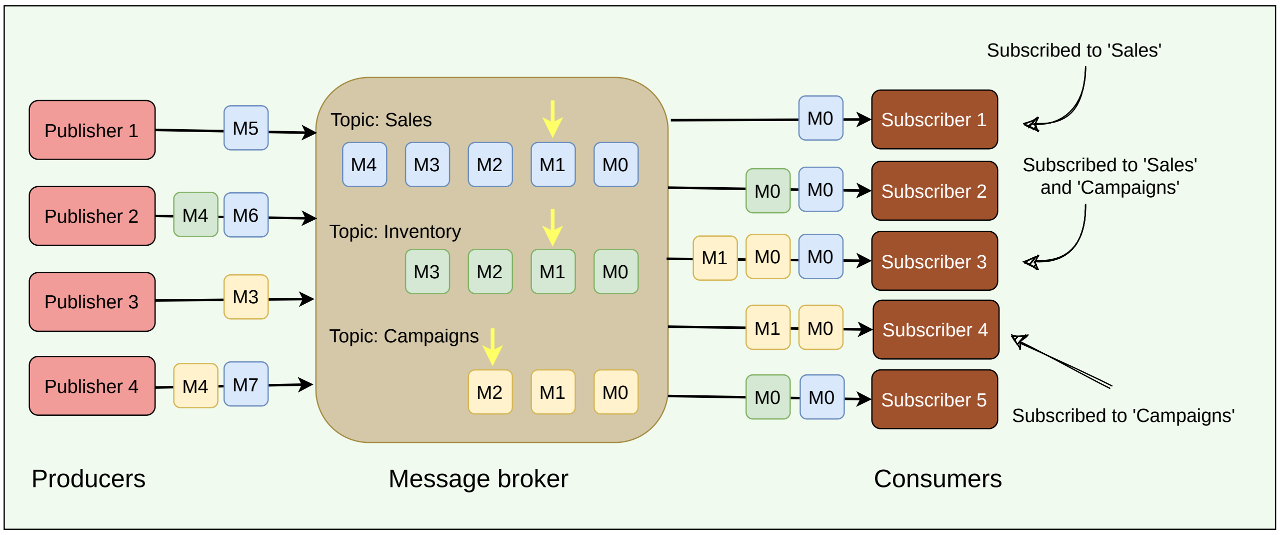
Kafka message

Topics#

Kafka divides its messages into categories called Topics. In simple terms, a topic is like a table in a database, and the messages are the rows in that table.

- Each message that Kafka receives from a producer is associated with a topic.
- Consumers can subscribe to a topic to get notified when new messages are added to that topic.
- A topic can have multiple subscribers that read messages from it.
- In a Kafka cluster, a topic is identified by its name and must be unique.

Messages in a topic can be read as often as needed — unlike traditional messaging systems, messages are not deleted after consumption. Instead, Kafka retains messages for a configurable amount of time or until a storage size is exceeded. Kafka's performance is effectively constant with respect to data size, so storing data for a long time is perfectly fine.



Kafka topics

Producers#

Producers are applications that publish (or write) records to Kafka.

Consumers#

Consumers are the applications that subscribe to (read and process) data from Kafka topics. Consumers subscribe to one or more topics and consume published messages by pulling data from the brokers.

In Kafka, producers and consumers are fully decoupled and agnostic of each other, which is a key design element to achieve the high scalability that Kafka is known for. For example, producers never need to wait for consumers.

High-level architecture#

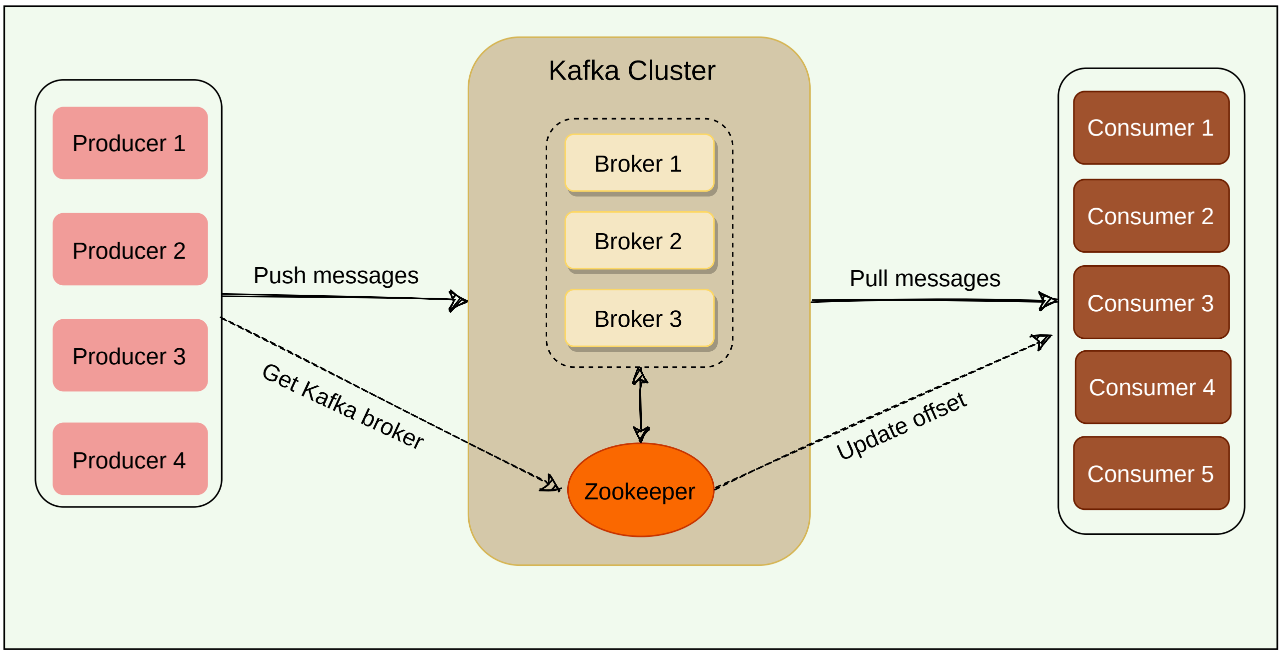
At a high level, applications (producers) send messages to a Kafka broker, and these messages are read by other applications called consumers. Messages get stored in a topic, and consumers subscribe to the topic to receive new messages.

Kafka cluster#

Kafka is deployed as a cluster of one or more servers, where each server is responsible for running one Kafka broker.

ZooKeeper#

ZooKeeper is a distributed key-value store and is used for coordination and storing configurations. It is highly optimized for reads. Kafka uses ZooKeeper to coordinate between Kafka brokers; ZooKeeper maintains metadata information about the Kafka cluster. We will be looking into this in detail later.



High level architecture of Kafka

Back

Next

Kafka: Introduction

Kafka: Deep Dive

✓ Mark as Completed

Report an Issue