KAFKA

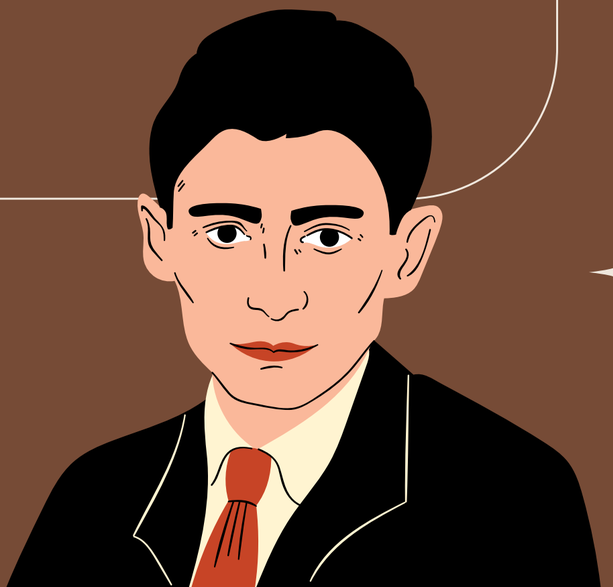
27.07.2023

Elifnur Kabalcı

elifnurkabalci

Yanımda yürüyordun Milena, düşünsene yanımda yürümüştün!

Milenaya Mektuplar | Franz Kafka



Agenda

WHAT IS KAFKA?

TOPIC PARTITION

BROKER

PRODUCER

CONSUMER

TOPIC REPLİCATION

MESSAGE DELIVERY SEMANTICS

API

What is Kafka?

It is a framework developed by LinkedIn in 2011 with java and later developed as open source under the umbrella of apache.

Provides error-free transfer of data from one system to other systems

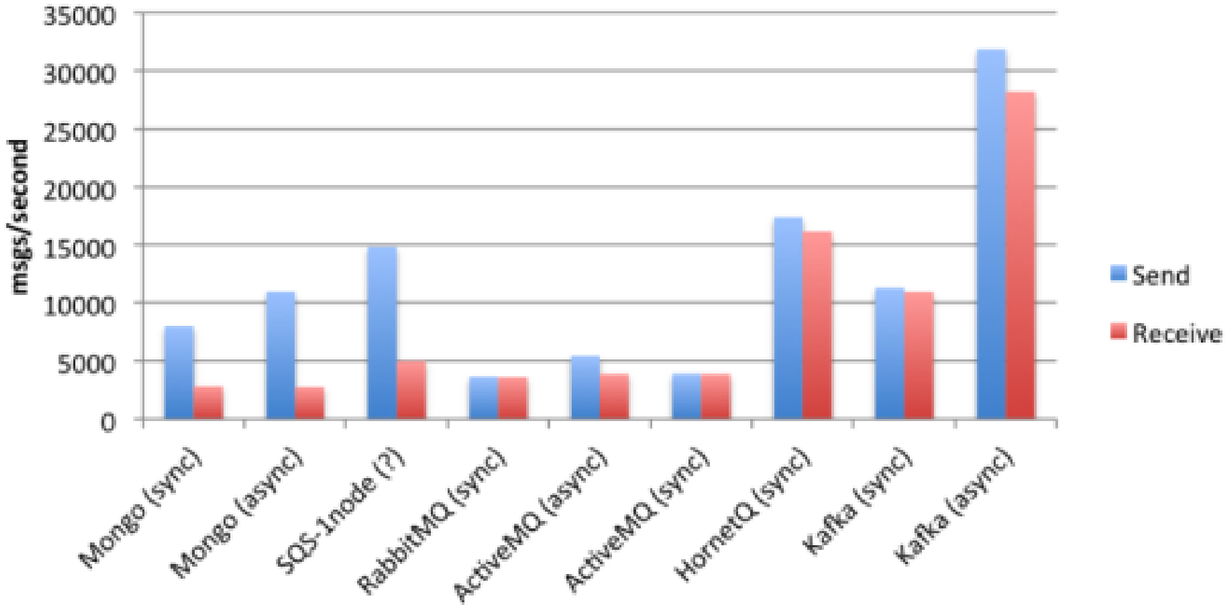
Kafka Technology

Görsel: Confluent

Advantages

1. Publishing and subscribing to event streams.
2. Storing event streams reliably and continuously for the desired period of time.
3. Process event streams as they occur or retrospectively.

Kafka vs.



Şekil 1.1. Apache Kafka'nın diğer sistemlerle karşılaştırılması [1].

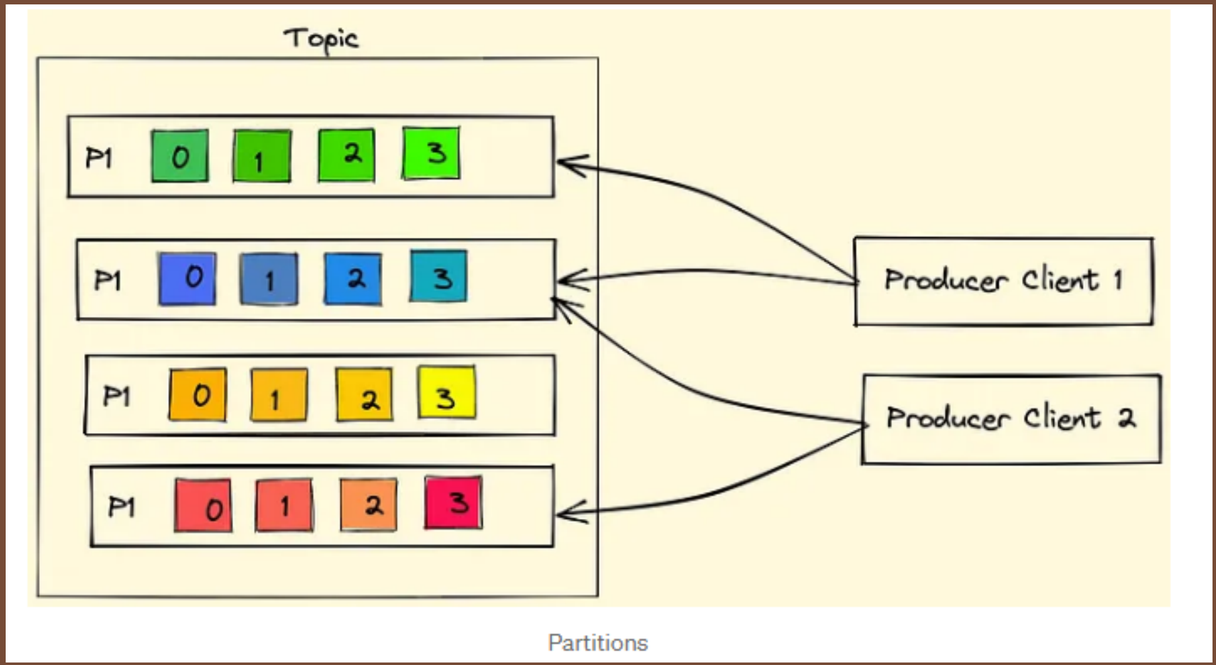
Topic Partition

Topic: data category from which data is sent and received

Partition: partitions of each data

Once the data is written to a partition, it is not deleted – immutability

Determines by offset 0-infinite



Partitions are sequential in themselves. We can't compare since the first offset written first will also be there.
If the number of partitions is fixed, the key value is used. With this, we send the data with the key. Then we can provide direct access. If there is no key, the data is randomly distributed to the partitions.
If the partition number is selected as 1, the consumer accesses the data in the order that the producer transmits it.

Broker

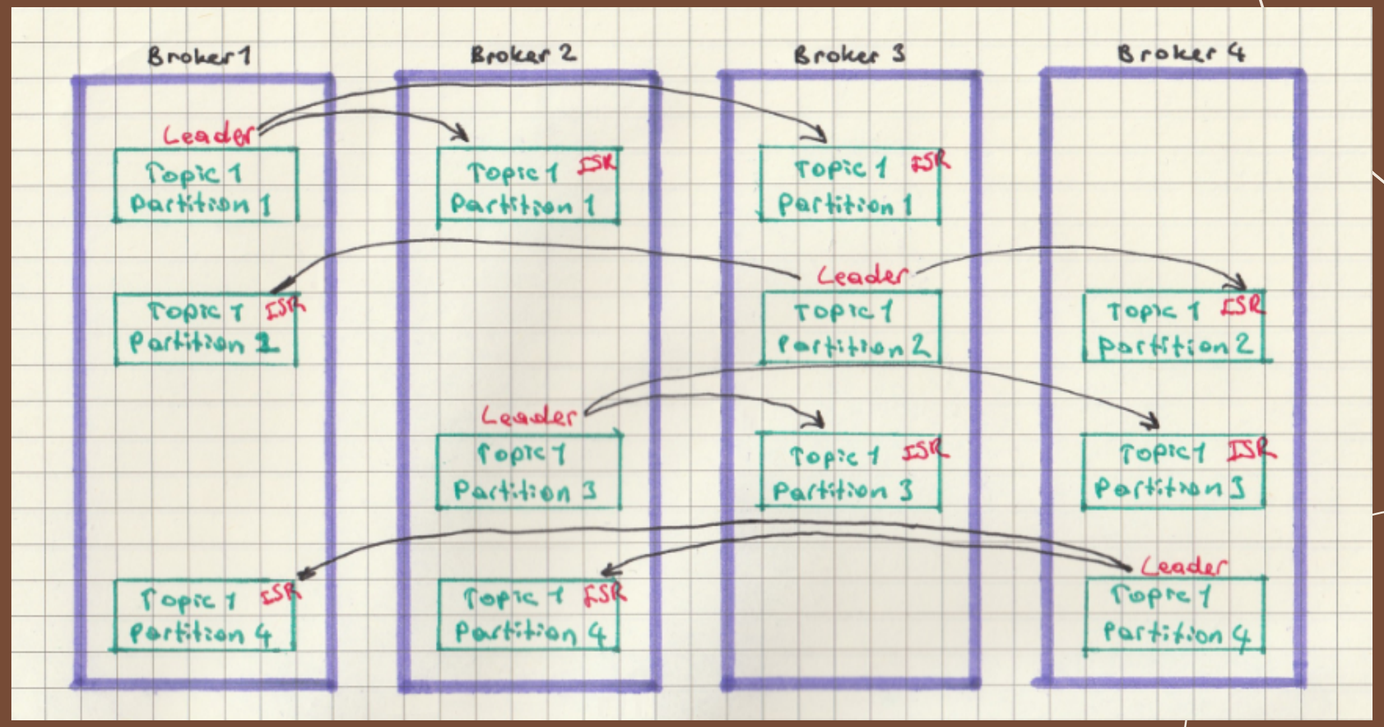
Servers that create Kafka clusters. Each broker is identified by ID

When creating a cluster, it starts with 3 brokers. It can be increased if necessary.

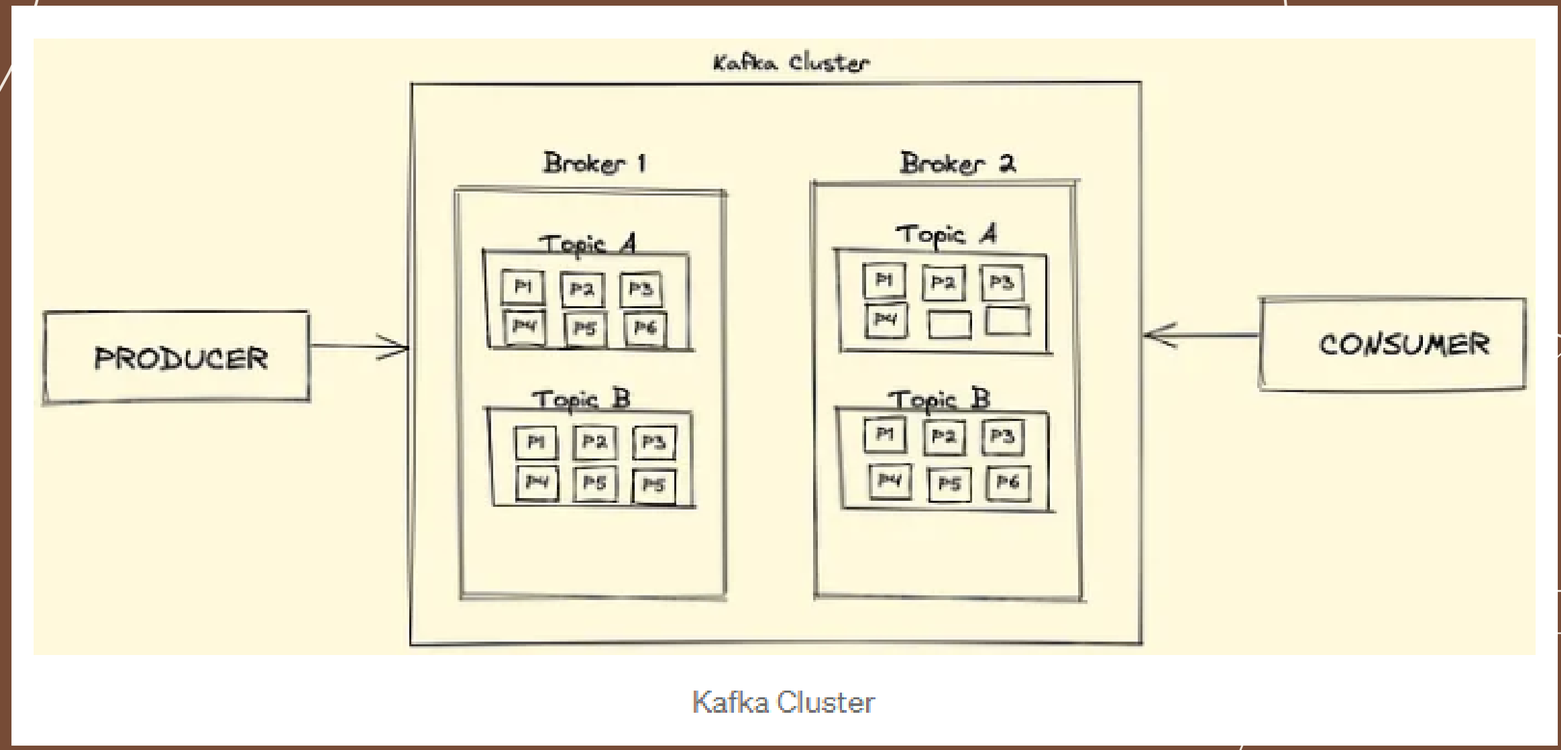
The connected server is called bootstrap broker.

! When we connect to bootstrap broker, we can access Kafka cluster data directly. This is thanks to the meta where all the brokers are kept.

Broker

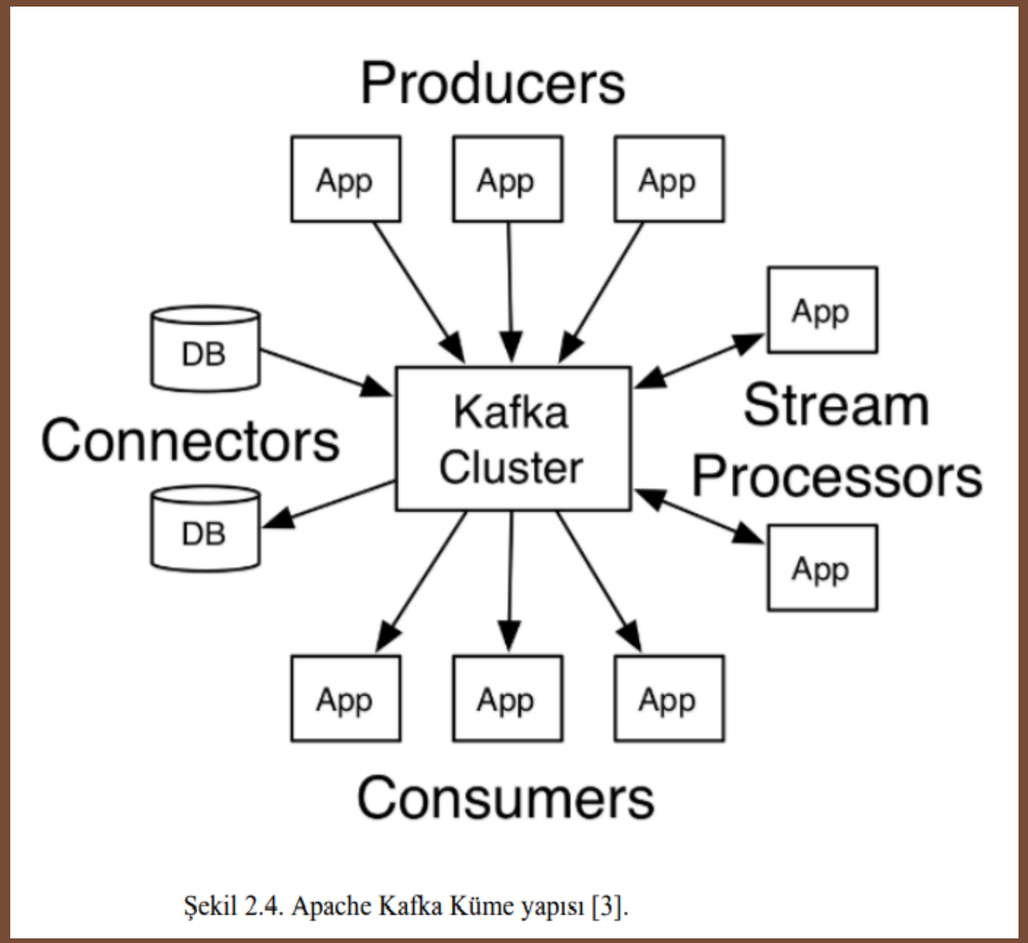

Broker 1
Broker 2
Broker 3
Broker 4
Leader
Topic 1 Partition 1
ISR
Topic 1 Partition 1
Topic 1 Partition 1
Topic 1 ISR Partition 2
Leader
Topic 1 Partition 2
Topic 1 ISR Partition 2
Leader
Topic 1 Partition 3
Topic 1 ISR Partition 3
Topic 1 ISR Partition 3
Topic 1 ISR Partition 4
Topic 1 ISR Partition 4
Leader
Topic 1 Partition 4

Kafka Cluster



Kafka Cluster

Kafka Cluster Design



Şekil 2.4. Apache Kafka Küme yapısı [3].

Producer

The component that sends a message to Topic is in the Publisher position.

Sending with a key: we send it by specifying the area to be sent, then it can be accessed.

Send without key: load balancing (to distribute the workload)

Round robin (uses sequential algorithm)

Sequential when sending, but then sequential access is not possible.

! If the data we send is sent to a non-existent topic, kafka creates the topic. But it is better to create it ourselves. We can specify replication and partition.

Acknowledge Mechanisms

ACKS	Gecikme (Latency)	Verimlilik (Throughput)	Devamlılık - Sağlamlık (Durability)
0	Düşük (Low)	Yüksek (High)	Garanti Edilmez (No Guarantee)
1	Orta (Medium)	Orta (Medium)	Sadece Liderler (Leaders Only)
All	Yüksek (High)	Düşük (Low)	Tüm Liderler ve Kopyalar (All Leaders and All Replicas)

Acknowledgement Tablosu

ACKS=0

Producer sends data to broker and does not wait for transaction confirmation and continues trading. As the transaction continues without approval, it is not known whether the broker is operational or the data has been sent successfully, data loss may occur.

ACKS=1

After the producer sends the data, it waits for the leader's approval. The leader learns whether the broker has successfully received the data and sends a feedback to the producer. In this case, a limited number of data loss may occur because the followers are not approved.

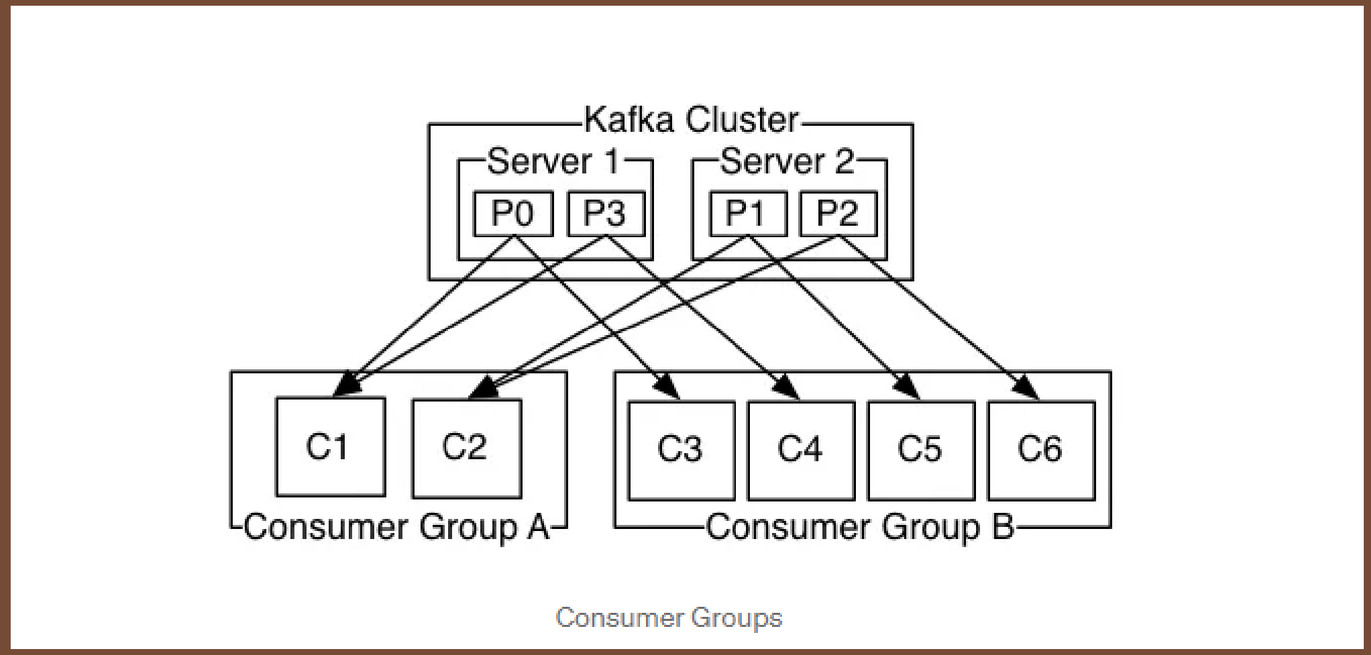
ACKS=ALL

Producer receives feedback from both the leader and followers that the data has been successfully delivered. In this case, the probability of losing data is very, very low! Of course, the compensation for not experiencing this data loss situation is paid by experiencing some delay.

Consumer

Subscriber position, which is a message from a certain topic
Parallel reading in partitions

Consumer



Consumer Groups

Topic Replication

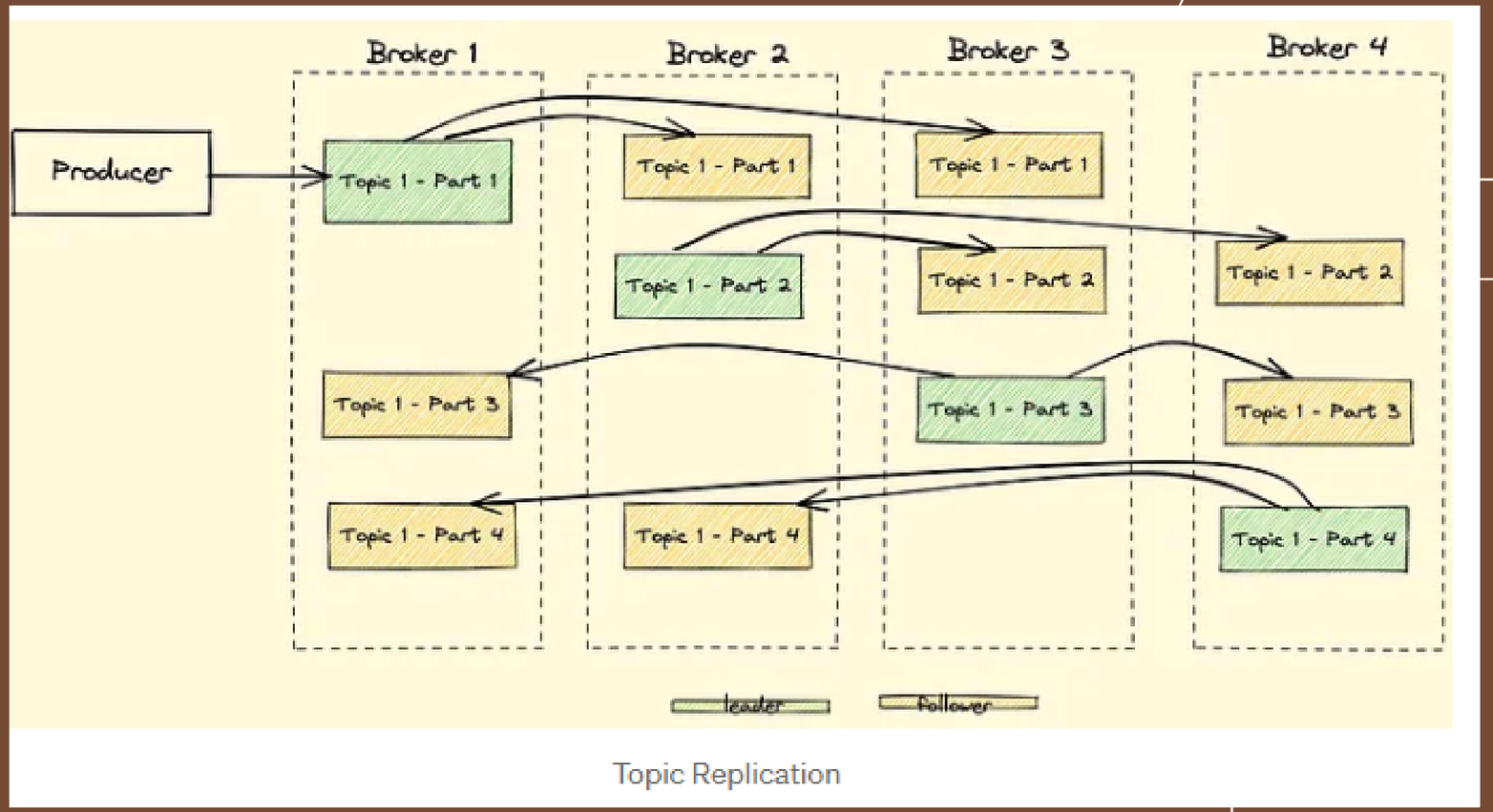
Data backup to prevent data loss

Each partition of the topics is stored on more than one server. One of them is the leader. Others ISR(in-sync replica) dneiled copy.

Leaders and ISRs are determined by the zookeeper.

Replications are determined by the replication-factor parameter when creating a topic.

Topic Replication



Topic Replication

Message Delivery Semantics

	At Most Once	At Least Once	Exactly Once
Veri Tekrarı	Yok	Var	Yok
Veri Kaybı	Var	Yok	Yok
İşlem Sayısı	0 veya 1 kere	1 veya daha fazla	sadece 1 kere

delivery-semantics.md hosted with ❤ by GitHub view raw

Delivery Semantics Tablosu

AT MOST ONCE

Offset information is written as soon as the message is received. The message is lost if an error occurs during its processing. It is the most efficient method, but it is not a preferred method as it may cause message loss.

AT LEAST ONCE

Offset information is written after the message is processed. The message is read again if an error occurs during its processing. Message loss does not occur, but there is a possibility of reading multiple messages. Therefore, our message processing process should be idempotent, that is, our system should not be affected in cases of duplicates. It is the most preferred method.

EXACTLY ONCE

This model is only provided with Kafka Streams API in Kafka to Kafka workflows. Transactional producer and consumer are used while the message is being transferred and processed between Kafka topics.

API

In addition to command line tools, Kafka provides us with five core APIs for Java and Scala.

ADMIN API

It allows to manage topics, brokers and other Kafka objects.

CONSUMER API

Allows you to read the topics and process the generated event streams.

KAFKA STREAMS API

Provides higher-level functionality to handle event streams.

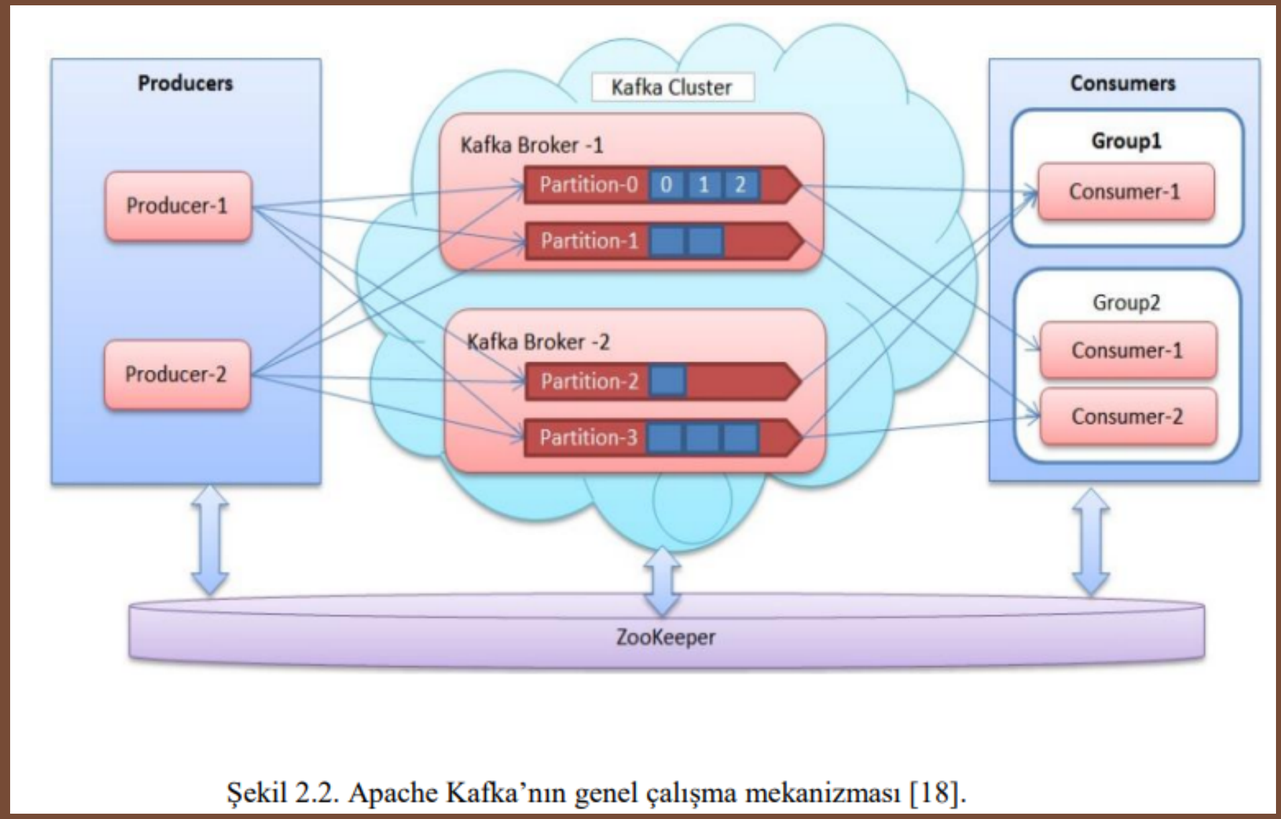
PRODUCER API

Allows writing event streams to topics.

KAFKA CONNECT API

It enables to read and write incoming and outgoing event streams from external applications that can be integrated with Kafka.

Kafka Mechanism



Şekil 2.2. Apache Kafka'nın genel çalışma mekanizması [18].

Apache Zookeeper

ZooKeeper is used to keep information about Kafka Cluster and consumer. ZooKeeper, which manages the brokers within itself by keeping a list, also undertakes the task of choosing leaders for partitions.

Apache Kafka Raft (KRaft)

KRaft is the consensus protocol that allows us to remove Kafka's dependency on ZooKeeper for data management. KRaft simplifies the Kafka architecture by merging data responsibility within Kafka instead of splitting it into two different systems.

Installation Links

Download kafka
http://kafka.apache.org/downloads.html

Install Kafka in terminal
https://devveri.com/big-data/apache-kafka

Install and use KafkaConnect-Cassandra Entegration
https://ravenfo.com/2021/09/04/kafkaconnect-kafka-cassandra-entegrasyonu/

Source

https://medium.com/sahibinden-technology/nedir-bu-apache-kafka-615b9582c270

https://medium.com/devopsturkiye/apache-kafkaya-giri%C5%9F-3399e5f33f8e

https://devveri.com/big-data/apache-kafka

https://efilli.com/blog/veri-bilimi-serisi-3-apache-kafka

https://www.veribilimiokulu.com/apache-kafka-temel-kavramlar/

https://ravenfo.com/2021/09/04/kafkaconnect-kafka-cassandra-entegrasyonu/

https://acikerisim.sakarya.edu.tr/bitstream/handle/20.500.12619/79310/T07528.pdf?sequence=1&isAllowed=y

THANK YOU FOR LISTENING

Do you have any questions?

@elifnurkabalci