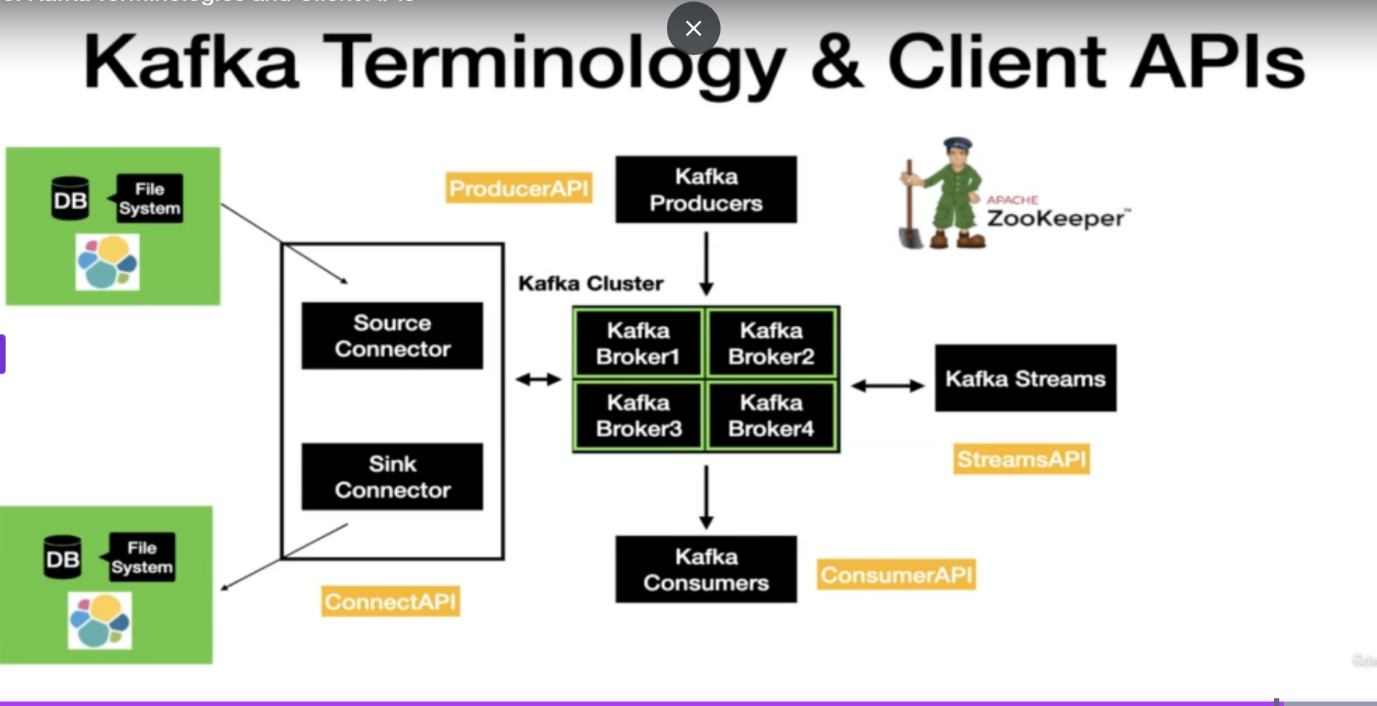
Kafka:

It’s an event streaming platform

The producer and consumer subscribe to a stream of records

It stores events on multiple servers based on the retention period

It’s distributed streaming system and has the ability to handle heavy loads.



What Is a Consumer Group?

A **consumer group** is a set of consumers that **work together to consume data** from a Kafka topic **in parallel**. Each consumer in the group gets **a subset of partitions** to consume.

## **How Consumer Groups Enable Scalability**

### **1. Parallel Data Processing**

* Kafka topics are divided into **partitions**.
* Each partition can be read by **only one consumer in a group at a time**.
* By adding more consumers to the group, Kafka can assign **one or more partitions per consumer**.
* **Committing offsets** lets Kafka know which messages have been **read and processed** so that:
* You **don’t reprocess** them on restart.
* You **can resume** from the correct point if the app crashes.
* It provides **at-least-once** or **exactly once** delivery semantics (with some effort).

**In Synch Replication:**

In Apache Kafka, **in-sync replication (ISR)** ensures **high availability and durability** of data by making sure that **replicas are up-to-date with the leader**.

## **Basic Replication in Kafka**

Each Kafka **partition** can have:

* One **leader replica** – handles reads/writes.
* One or more **follower replicas** – replicate data from the leader.

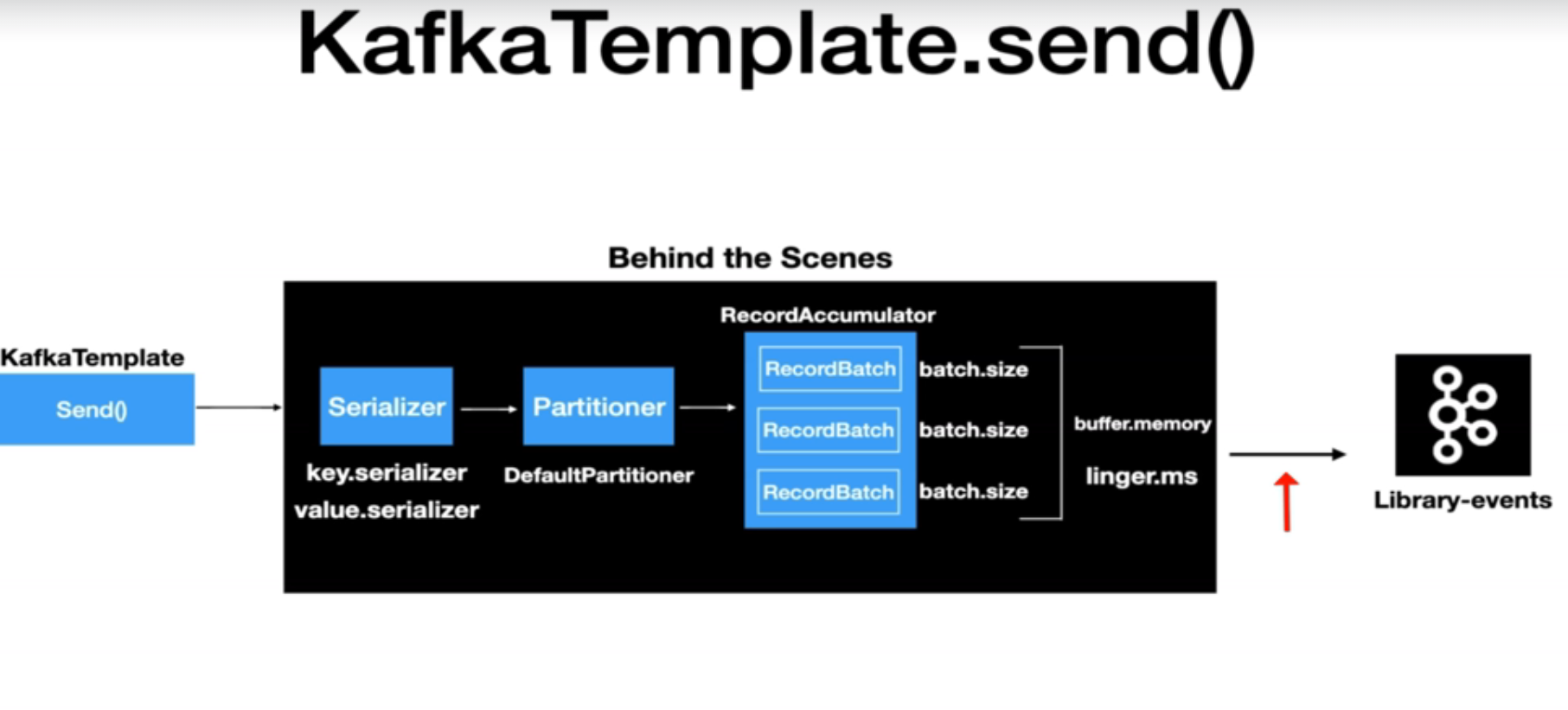
Kafka uses ISR to **guarantee durability**:

### **Only commits messages when:**

* They are **written to all in-sync replicas**.

This ensures that **even if the leader crashes**, another replica **can take over with no data loss**.

* If the **leader broker fails**, Kafka picks a **new leader from the ISR**.
* If **no ISR is available**, writes can fail (depending on min.insync.replicas and acks settings).

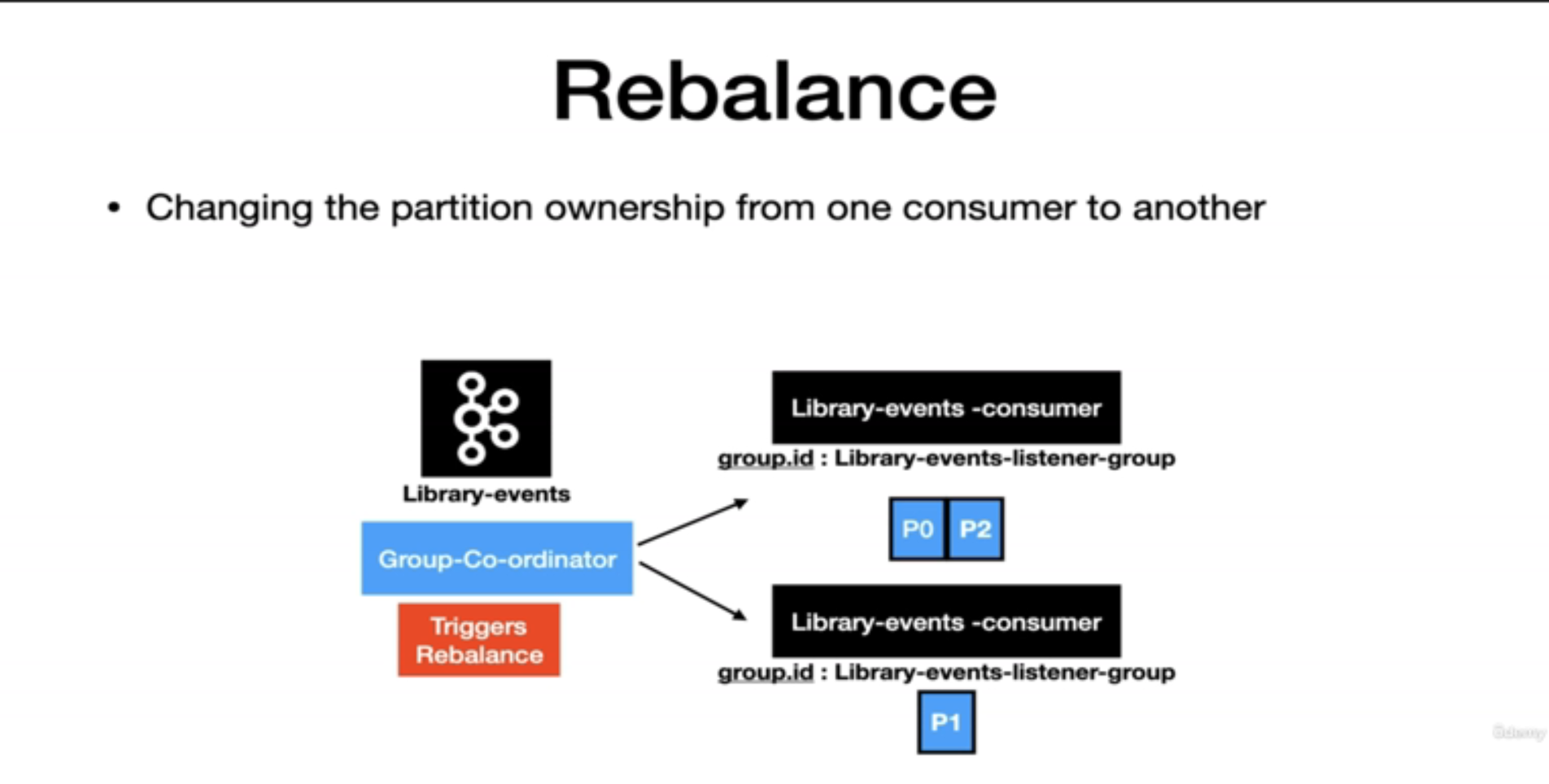


Boostrap Server:

The "bootstrap server" in Kafka is a crucial configuration setting for any Kafka client (producers, consumers, or admin clients) that wants to connect to a Kafka cluster.

Its primary purpose is to serve as an **initial entry point** for the client to discover the full topology and metadata of the Kafka cluster.

* A Kafka cluster is distributed, meaning it consists of multiple Kafka brokers (servers). A client doesn't need to know the IP addresses or hostnames of *all* brokers in the cluster.
* Instead, you provide the bootstrap.servers configuration, which is a comma-separated list of host:port pairs for **one or more** (typically 2-3 for redundancy) Kafka brokers in your cluster.
* When a client first starts, it attempts to connect to one of these specified bootstrap servers.

Rebalance : 

**Retry Listener:**

A retry listener in **Kafka** is a mechanism that allows you to **reprocess messages that failed to be processed** initially. It's often used in conjunction with a **dead-letter queue (DLQ)** to handle transient processing failures without losing data.

**CommonErrorHandler:**

In Spring for Apache Kafka, the CommonErrorHandler is a versatile interface designed to handle exceptions that occur during message processing. It is the successor to the deprecated ErrorHandler interface and provides a more flexible way to manage various types of exceptions, including ListenerExecutionFailedException, which is a common occurrence.