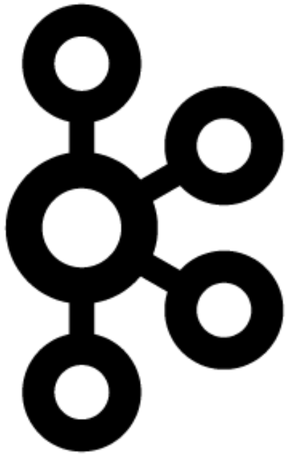
Fundamentals of

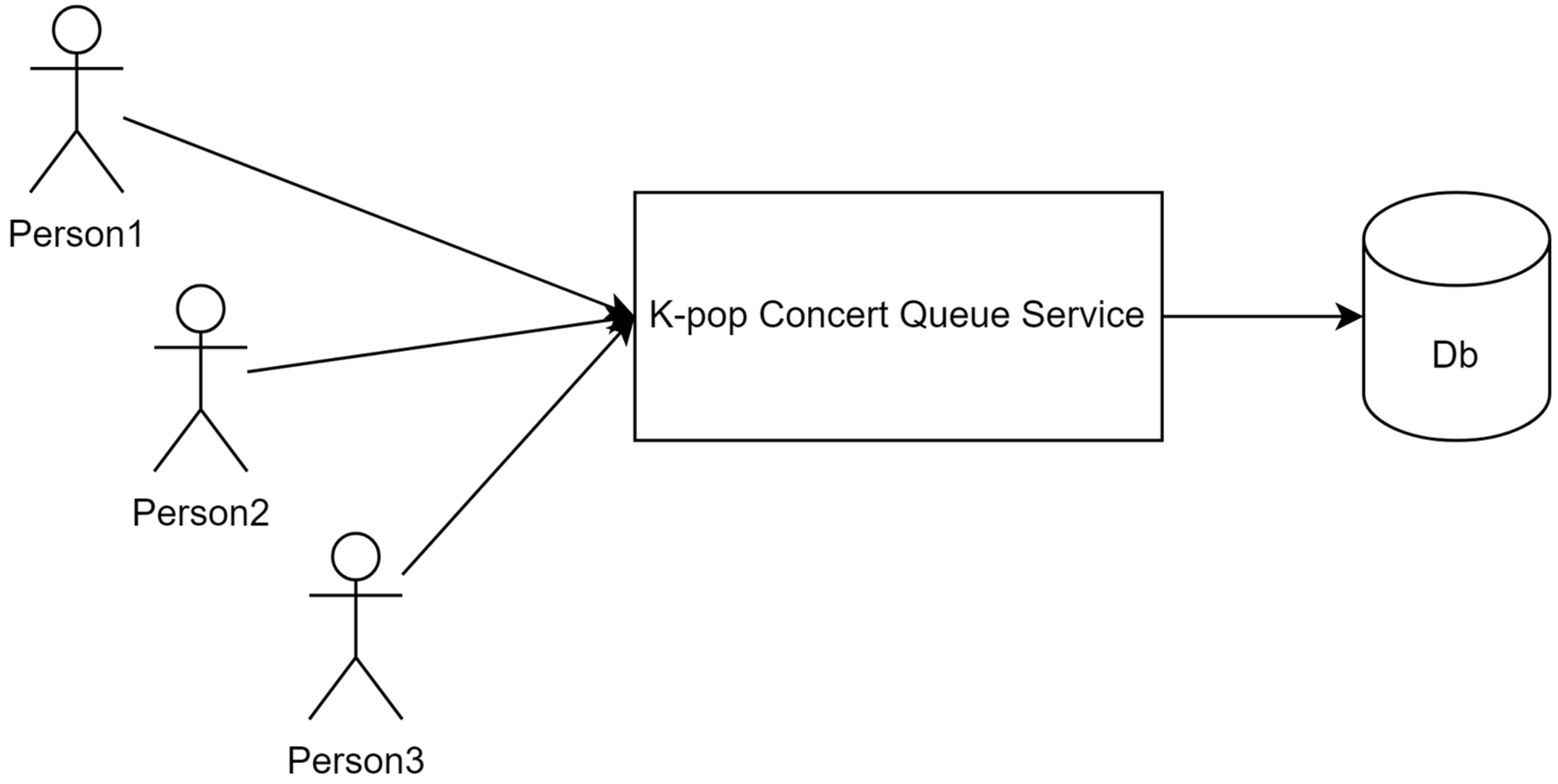


kafka

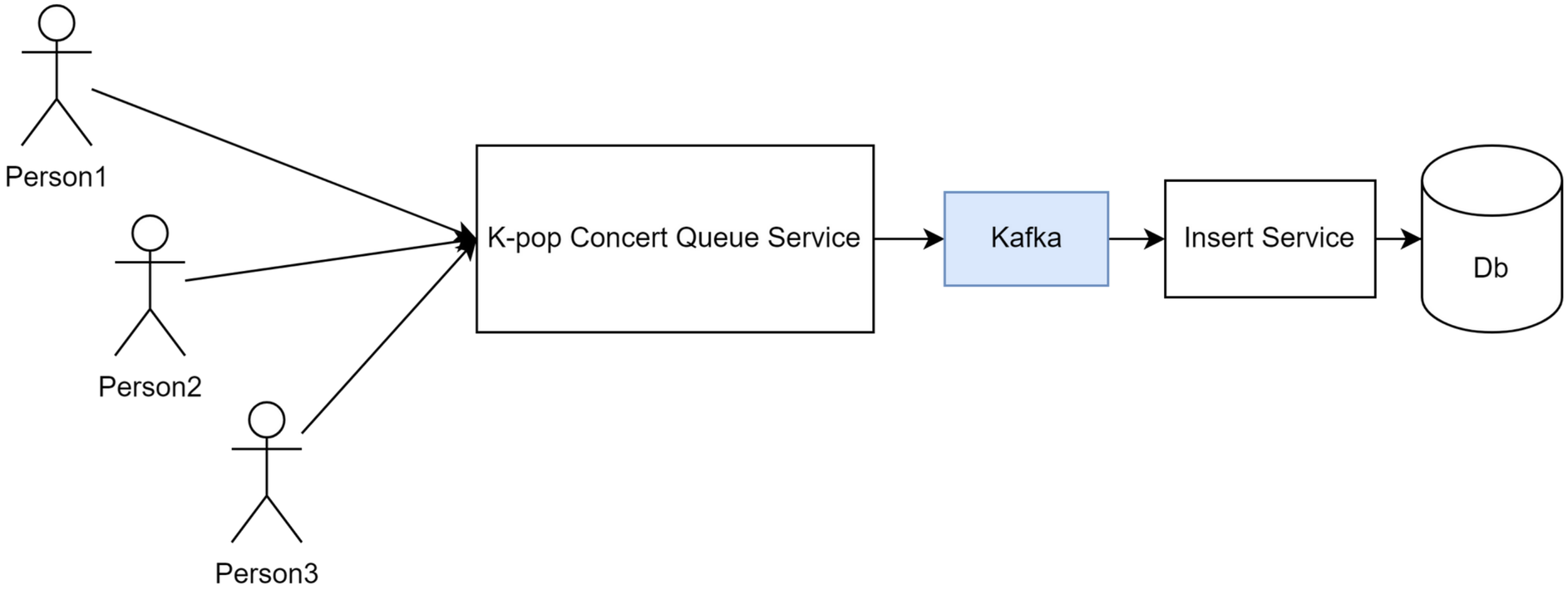
for Beginners

By Lookhin@Neversitup
Backend Developer

Problem

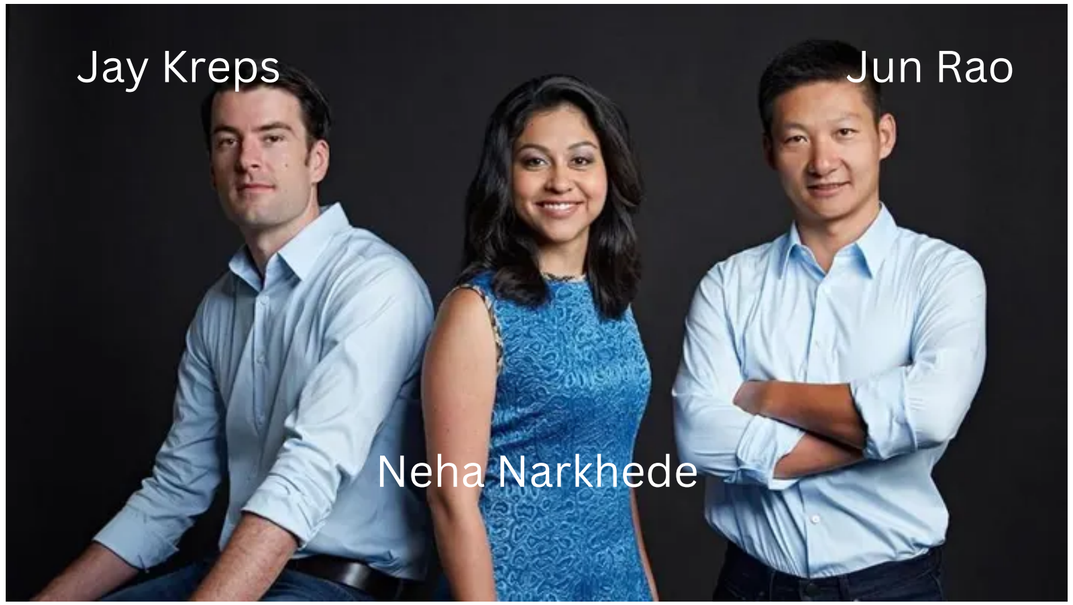

Person1
Person2
Person3
K-pop Concert Queue Service
Db

How to solve this???

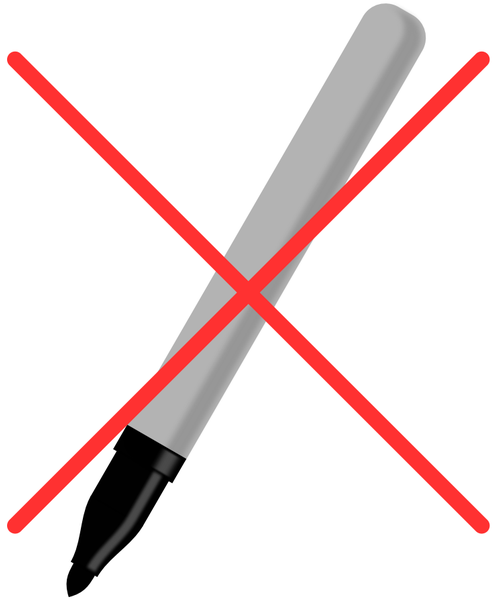

Person1
Person2
Person3
K-pop Concert Queue Service
Kafka
Insert Service
Db

A Brief History of Kafka

- Kafka was originally developed by LinkedIn and was subsequently open-sourced in early 2011, **Jay Kreps**, **Neha Narkhede**, and **Jun Rao** helped co-create Kafka.
- Kafka was written in **Java** and **Scala** programming languange.



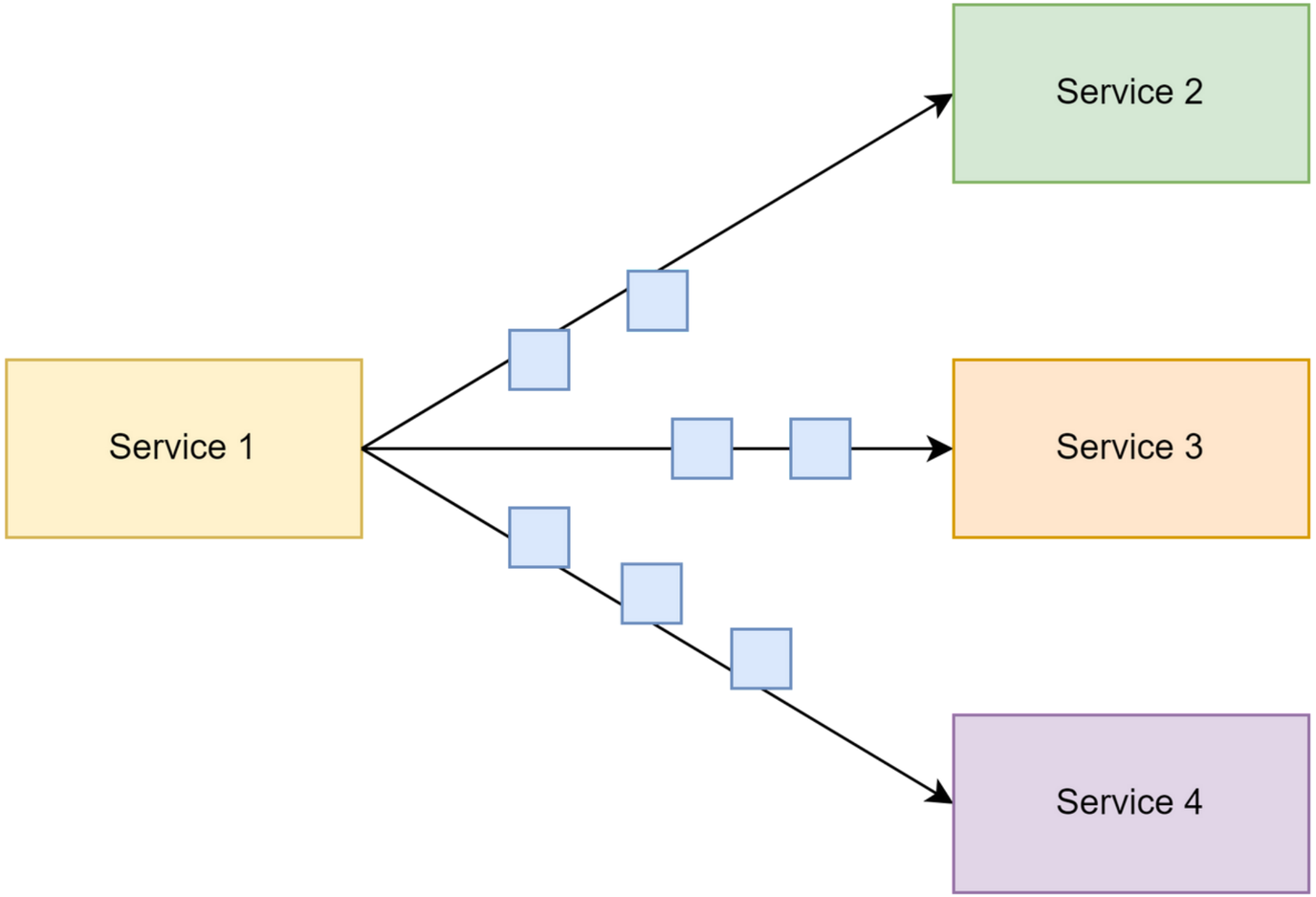
Concept & Fundamentals



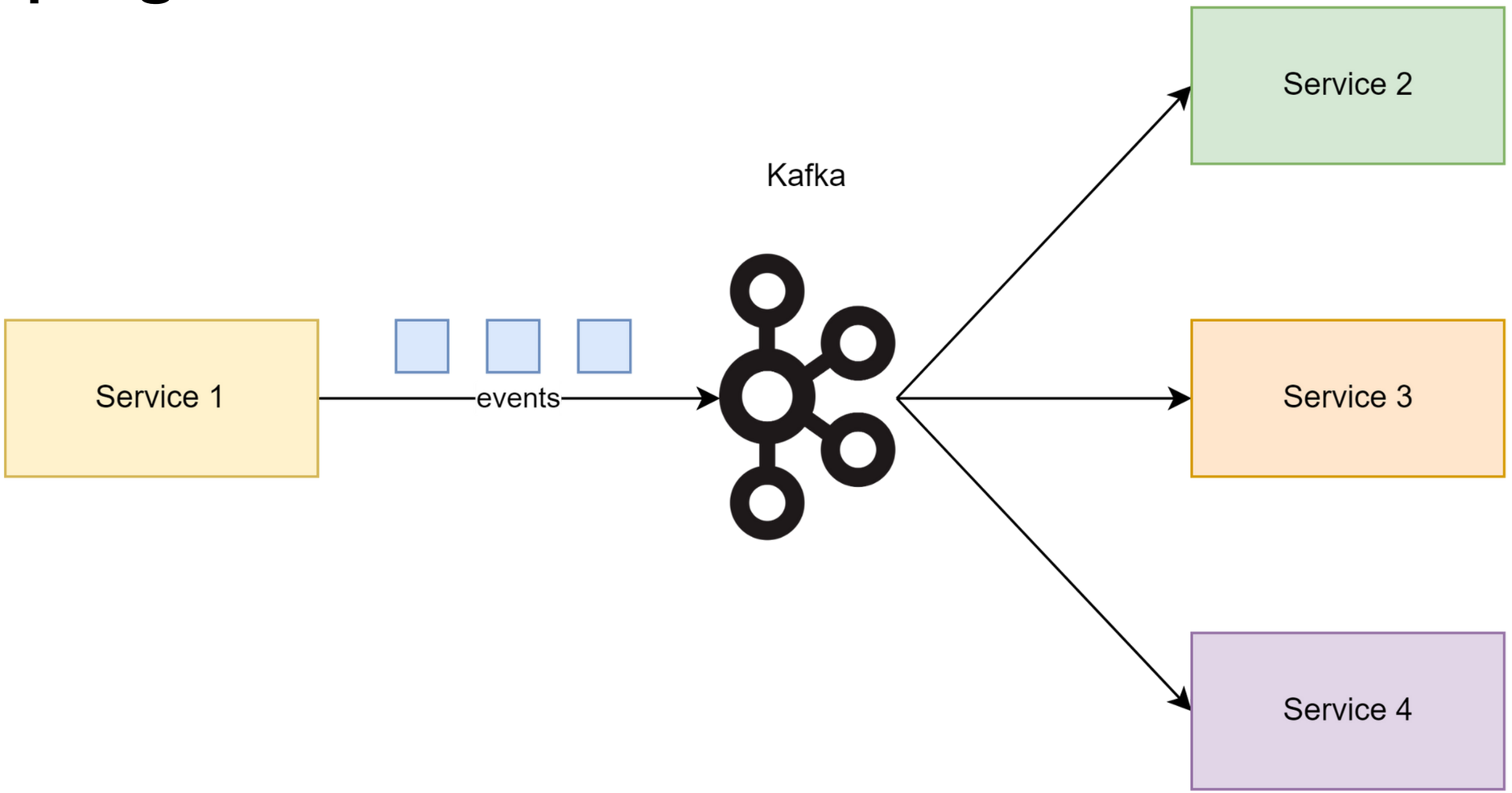
What is Kafka???

Kafka is a high-performance distributed streaming events platform that communicates via **TCP-network protocol**. Kafka can evaluate couping service to service, scalability, and fault tolerance guarantee.

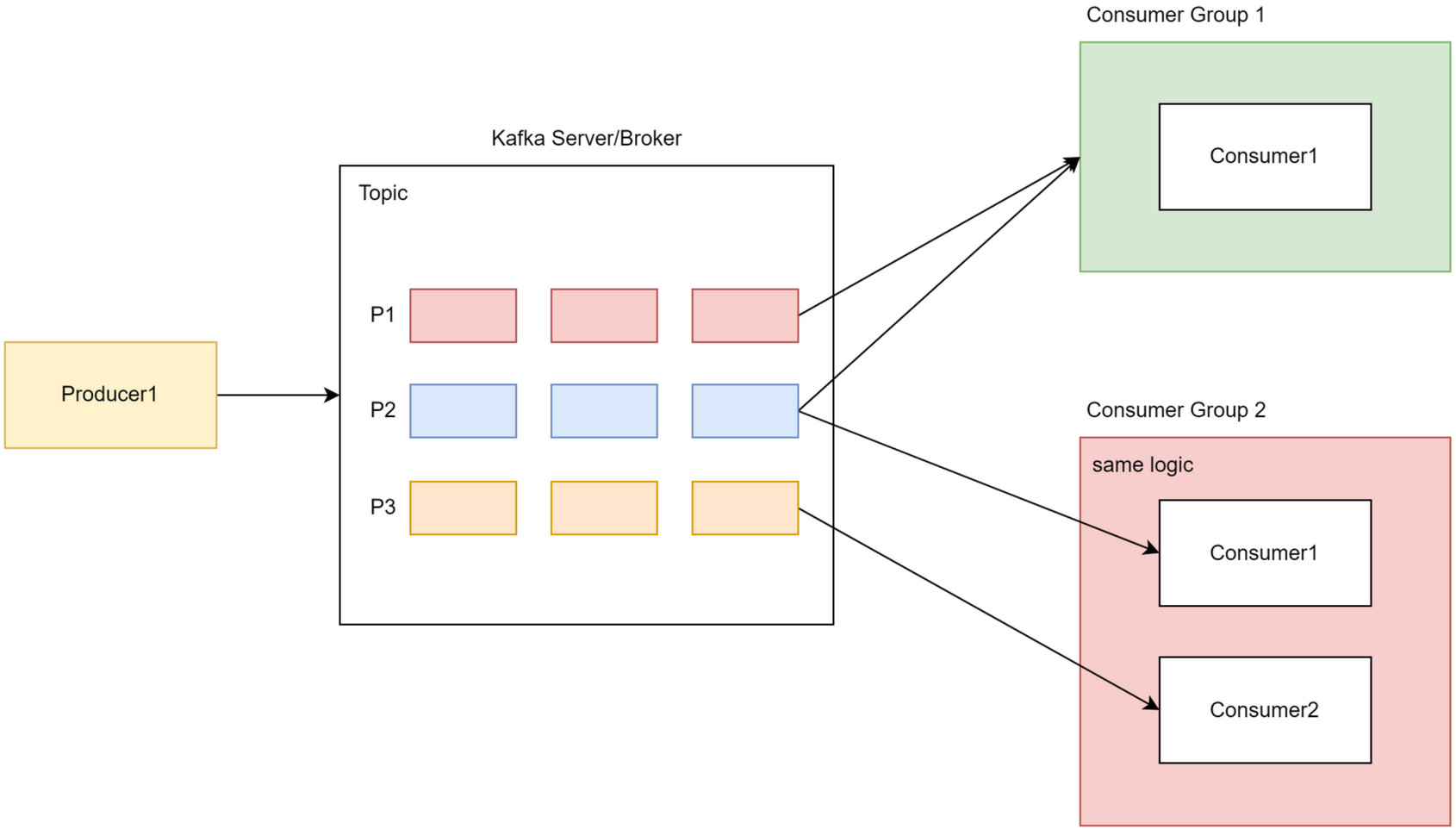
Coupling service-to-service



Decoupling service-to-service



Components of Kafka


Consumer Group 1
Consumer1
Kafka Server/Broker
Topic
P1
P2
P3
Producer1
Consumer Group 2
same logic
Consumer1
Consumer2

Broker

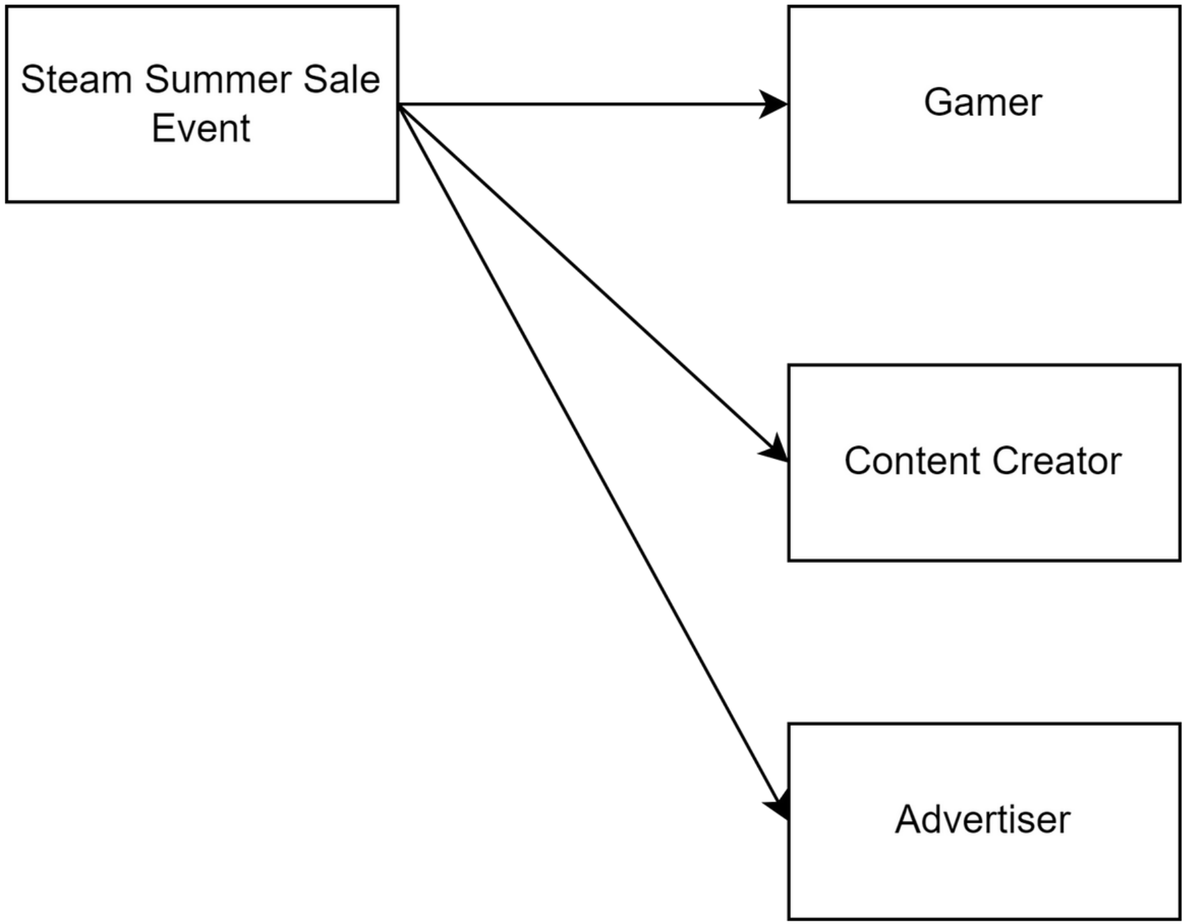
It arranges transactions of topics between producers and consumers, Thus Kafka is the broker.

Topic

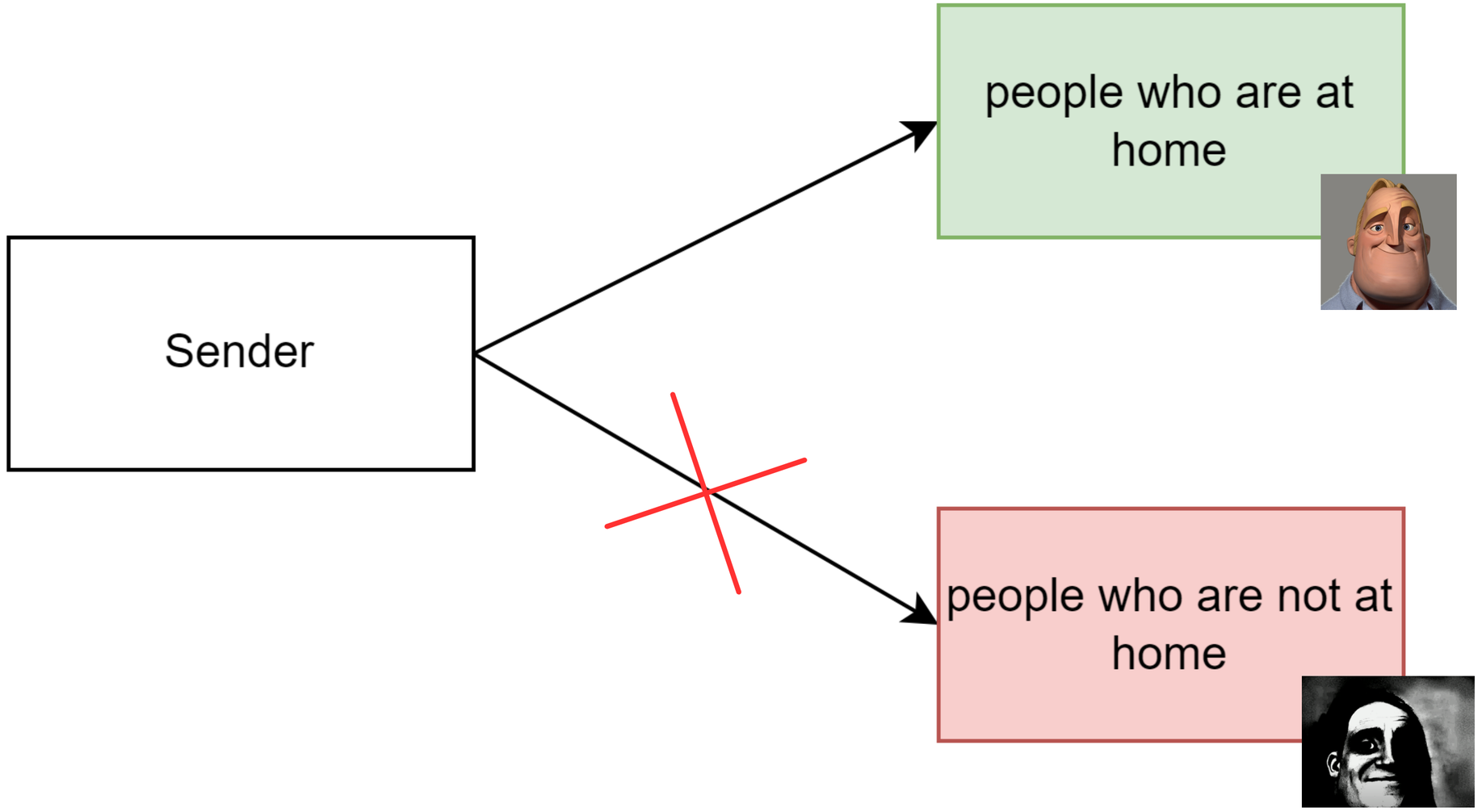
- It's a group of events or data that can contain a group of partitions.
- A single topic can be consumed by multiple consumers in parallel.

Note that: Kafka's topic used fan-out messaging solution.

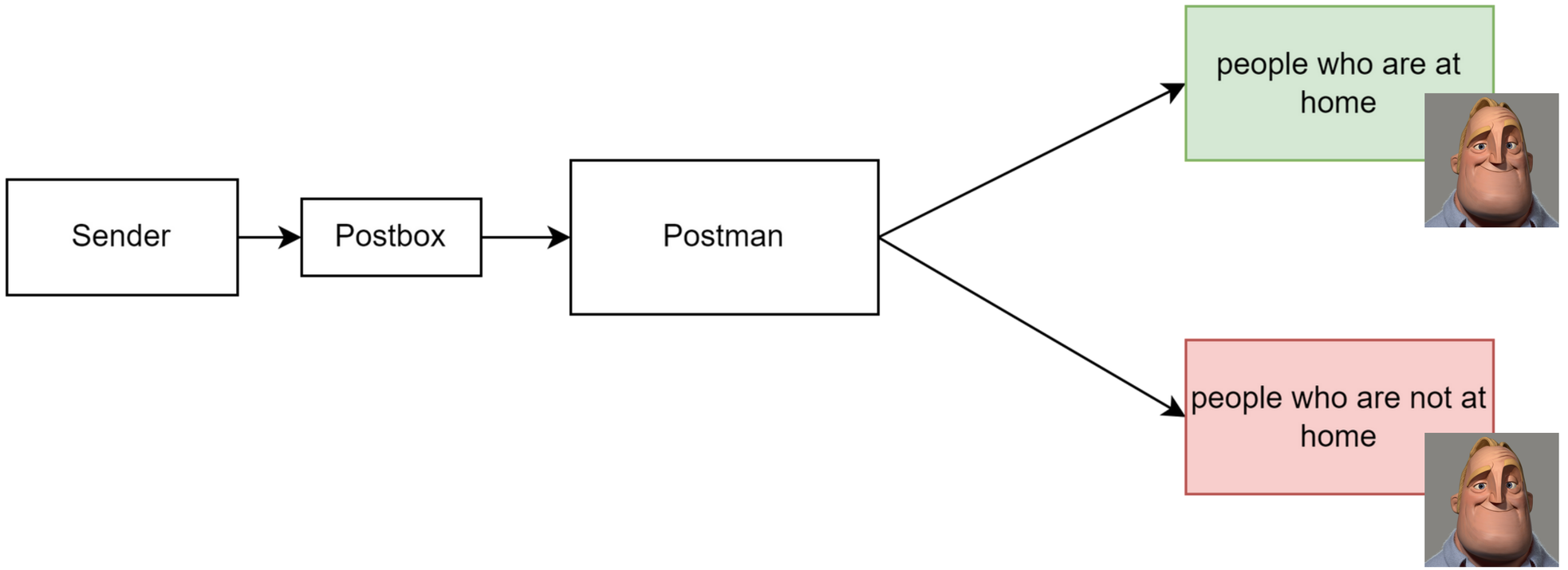
Fan-out is a messaging pattern that implies the delivery of a message to one or multiple destinations possibly in parallel and doesn't wait for any response from the receiver.


Steam Summer Sale Event
Gamer
Content Creator
Advertiser

Non Fan-Out



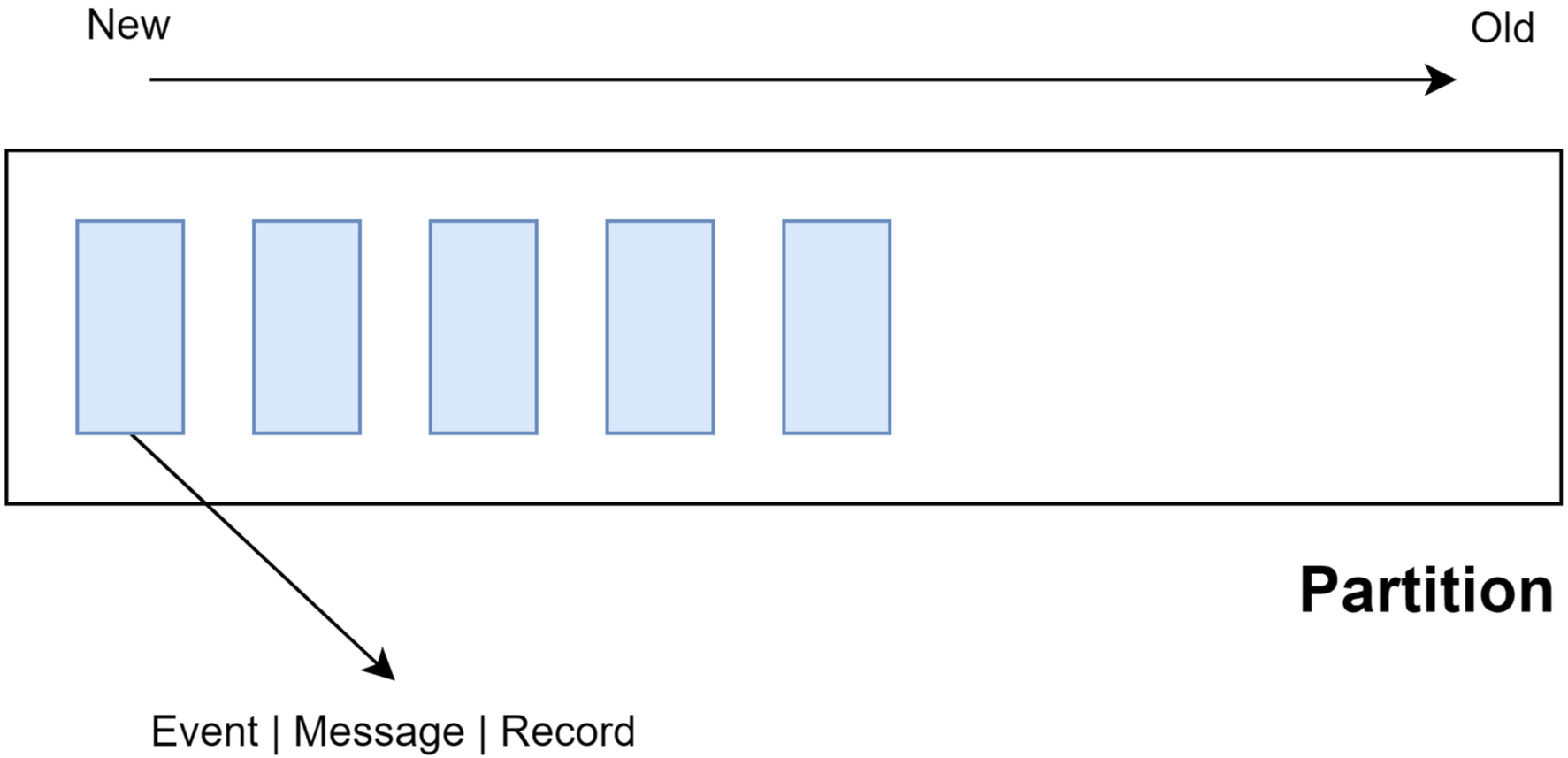
Fan-Out

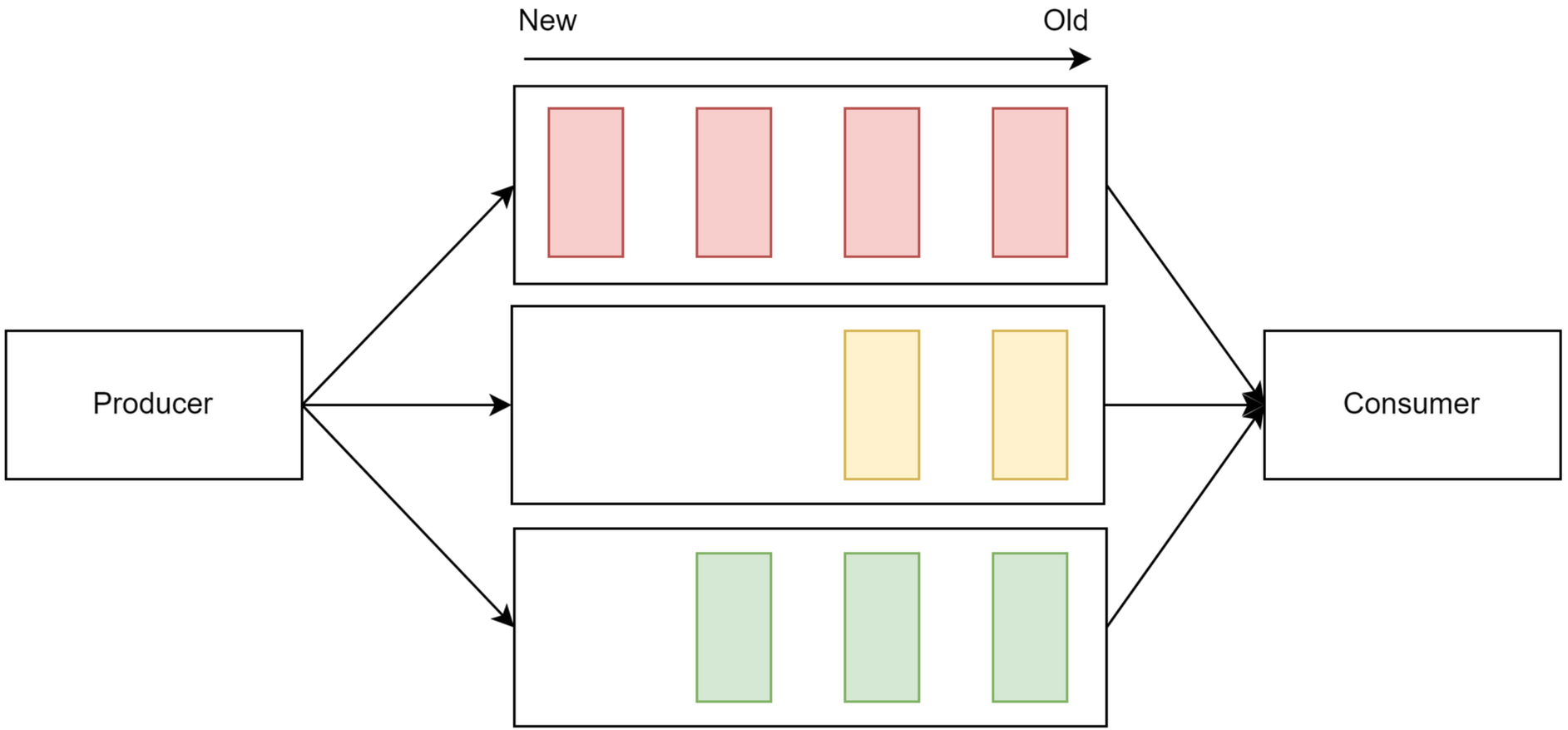

Sender
Postbox
Postman
people who are at home
people who are not at home

Partitions

- Partitions contain events|messages|records|data.
- It's can scalable from 1 to N partitions for each topics

Note that: Kafka's topics are divided into several **partitions**. While the topic is a logical concept.


New
Old
Partition
Event | Message | Record

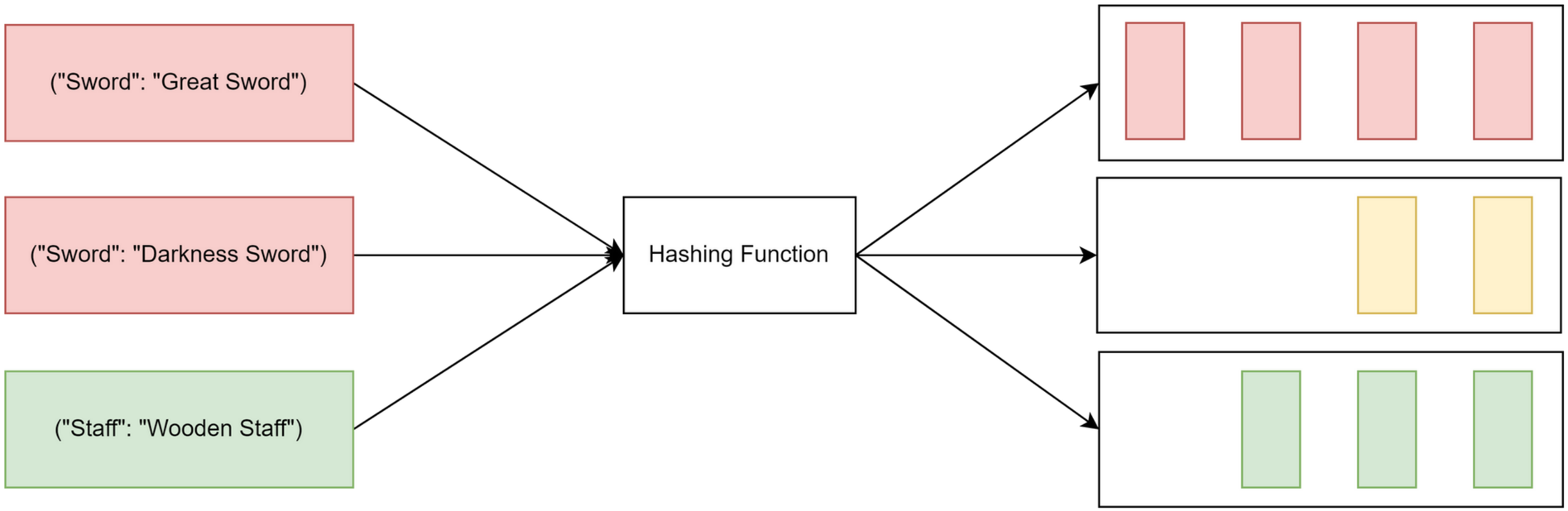

New
Old
Producer
Consumer

Use case: The consumer number must be less or equal to partitions to prevent an **idle consumer**.

Using a partition key to specify the partition

- A producer can use a partition key to direct messages to a specific partition.
- By default, the partition key is passed through a hashing function, which creates the partition assignment.
- That assures that all records produced with the same key will arrive at the same partition.

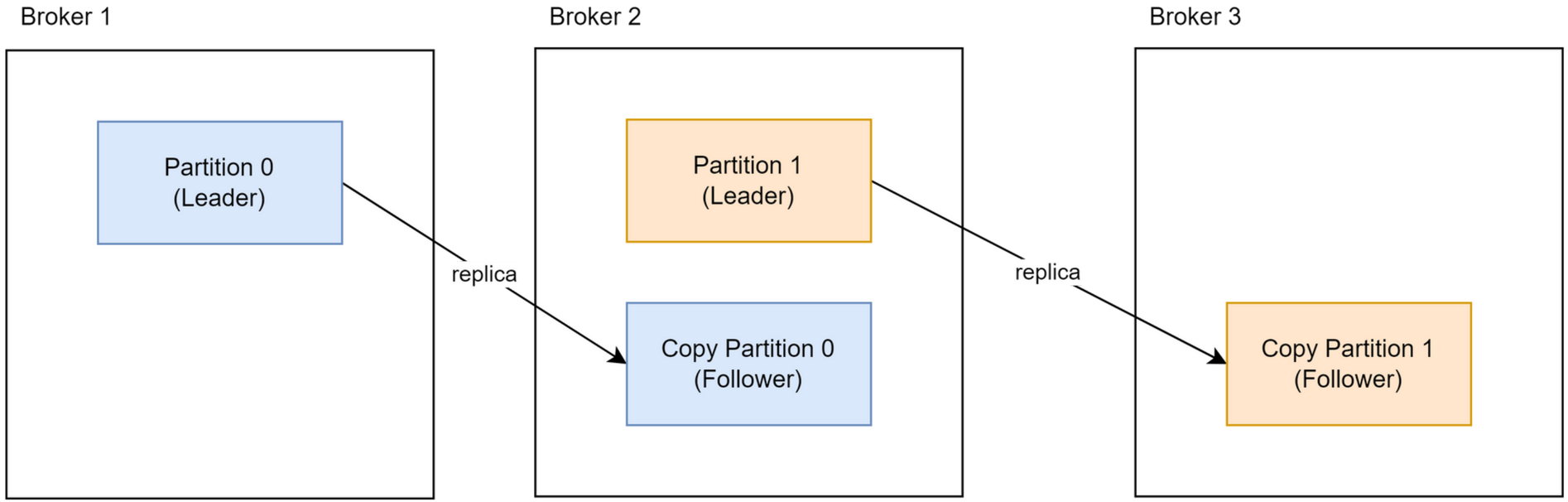
Note that: If a producer doesn’t specify a partition key when producing a record, Kafka will use a round-robin partition assignment.


("Sword": "Great Sword")
("Sword": "Darkness Sword")
("Staff": "Wooden Staff")
Hashing Function

Replication Factors

It's the number of copies of data over multiple brokers.

Note that: The replication factor value should be greater than 1 always (between 2 or 3). This helps to store a replica of the data in another broker from where the user can access it.


Broker 1
Broker 2
Broker 3
Partition 0
(Leader)
replica
Partition 1
(Leader)
Copy Partition 0
(Follower)
replica
Copy Partition 1
(Follower)

Leader

- The replica that allowed to serve the client's request.
- The leader handles all the read and writes operations of data for the partitions.

Note That: Each partition can only have one leader at a time.

Followers

Its an other replica that are not the leader.

Note That: Leader and their followers are determined by the zookeeper.

???

APACHE

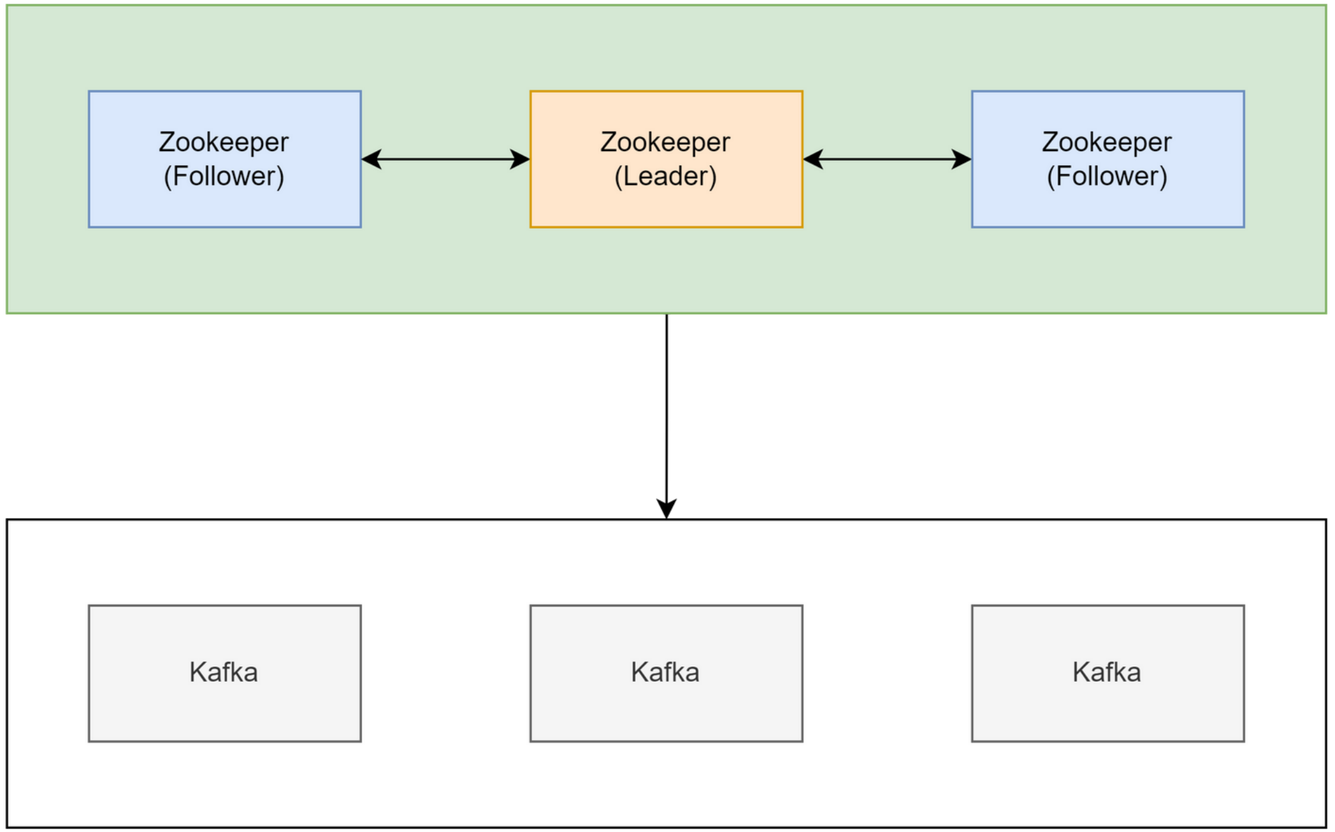
ZooKeeper™

Apache Zookeeper

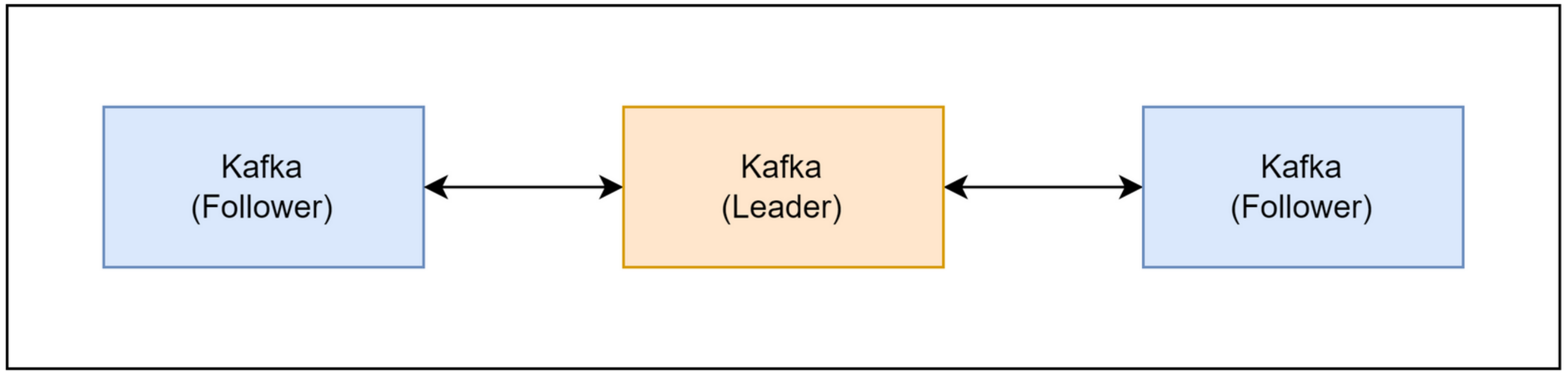
It's a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services.

Note that: Nowadays, Apache Kafka is trying to separate Zookeeper out and change to **KRaft** mode to provide each leader to lead their followers by themselves.

with Apache Zookeeper



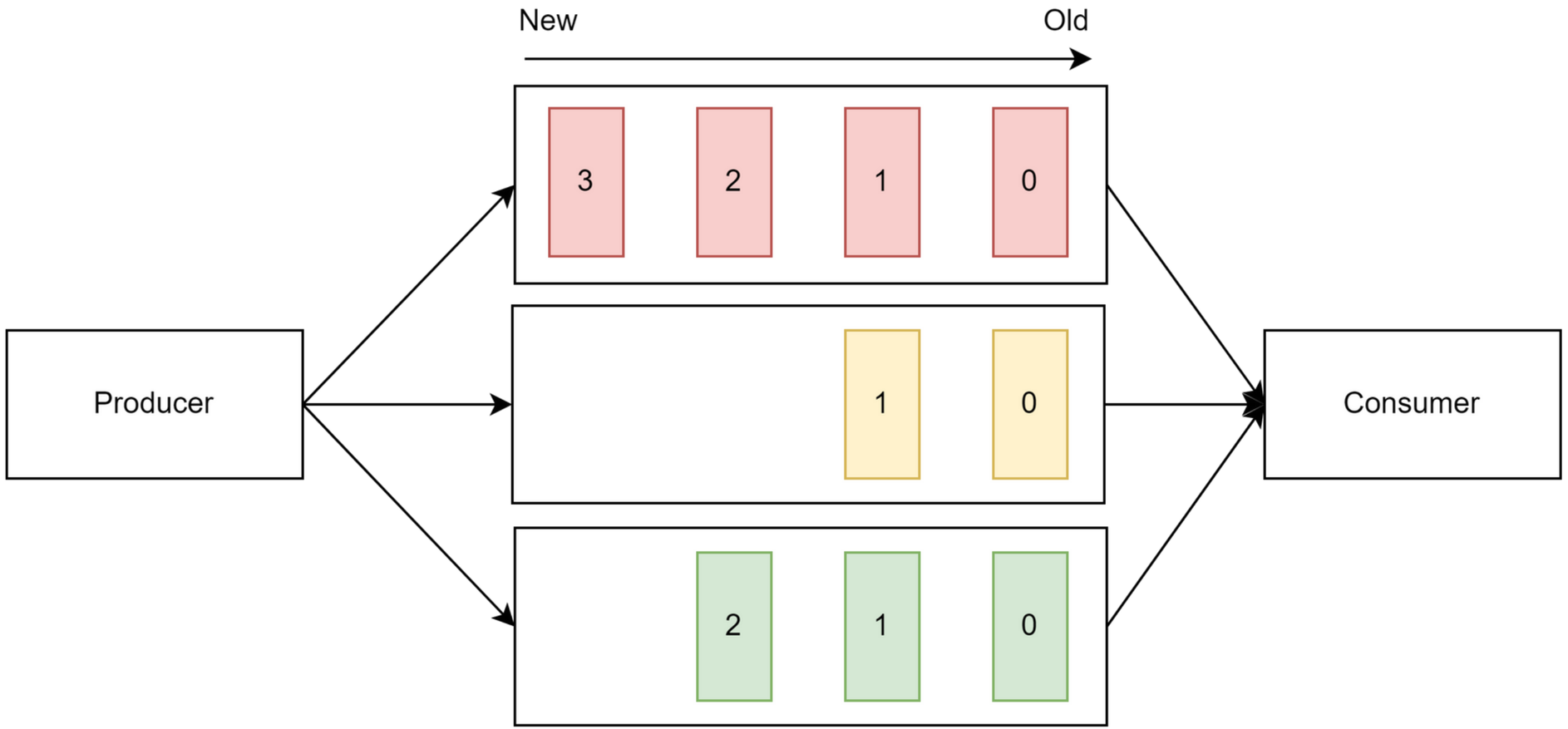
with KRaft mode



Offsets and the ordering of records

- The records in the partitions are each assigned a sequential identifier called the offset, which is unique for each record within the partition.
- The offset is an incremental and immutable number, maintained by Kafka.

Note that: Although records are the same, the offsets are still unique.


New
Old
3
2
1
0
1
0
2
1
0
Producer
Consumer

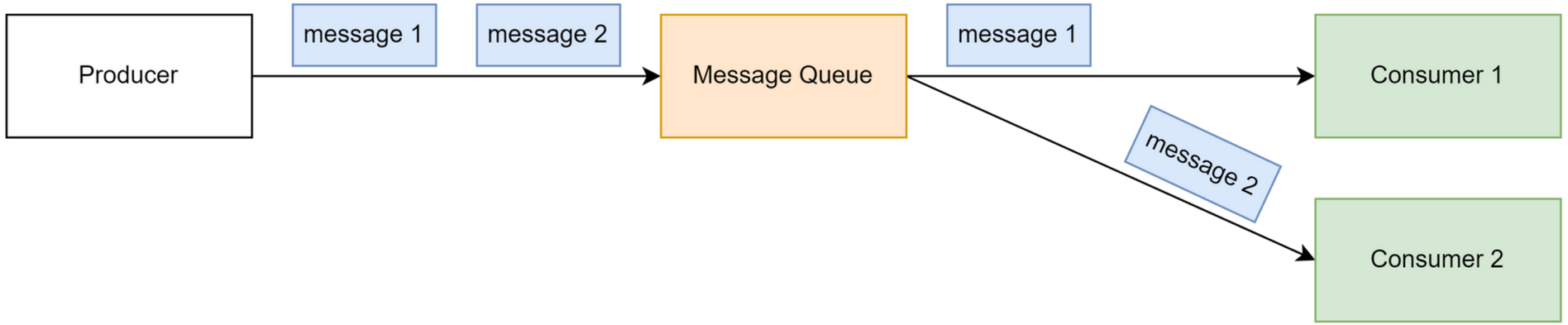
Note that: Although the messages within a partition are ordered, messages across a partition are not guaranteed to be ordered.

Event | Message | Record

- A Messaging System is responsible for transferring data from one application to another.
- Messages are queued asynchronously between client applications and messaging system.

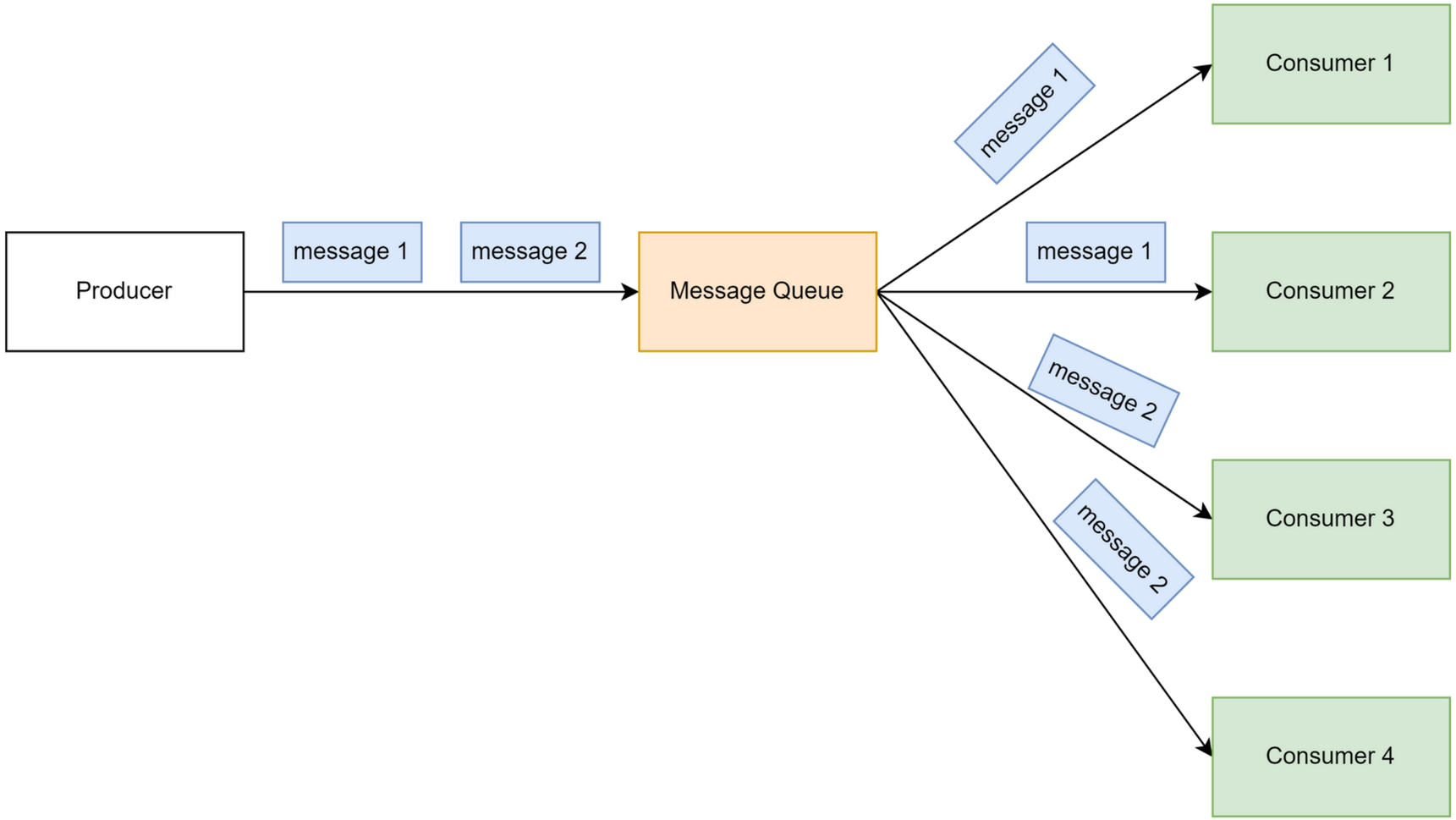
Point-to-Point Messaging System

- Messages are persisted in a queue.
- One or more consumers are available to consume a message in the queue, but a particular message can be consumed by a maximum of one consumer only.
- Once a consumer reads a message in the queue, it disappears from that queue.


Producer
message 1
message 2
Message Queue
message 1
Consumer 1
message 2
Consumer 2

Publish-Subscribe Messaging System

- Messages are persisted in a topic.
- Consumers can subscribe to one or more topics and are available to consume all messages.
- Message producers are called publishers and message consumers are called subscribers.


Producer
message 1
message 2
Message Queue
message 1
message 1
message 2
message 2
Consumer 1
Consumer 2
Consumer 3
Consumer 4

Kafka's Messaging System

- Apache Kafka is a distributed publish-subscribe messaging system.
- A robust queue that can handle a high volume of data and enables you to pass messages from one end-point to another.
- Messages are persisted on the disk and replicated within the cluster to prevent data loss.

What if consuming a message failed? How can we handle this situation?

Don't worry, for the reason that Apache Kafka is a **"Publish-Subscribe Messaging System".**

If you can't consume a message in that time you can re-try to consume it any time you need.

Producer

The producer sends data directly to the broker that is the leader for the partition.

Note that: Kafka is available to clients to config a compression of messages.

Message Compression in Kafka

- Reducing network bandwidth usage.
- Saving disk space on Kafka brokers.

Disk space requirements calculation

(avg-msg-size) * (msgs-per-day) * (retention-period-days) * (replication-factor)

Example

- Average message size is 10 kiB
- Messages per day is 1,000,000
- Retention period is 5 days
- Replication factor is 3

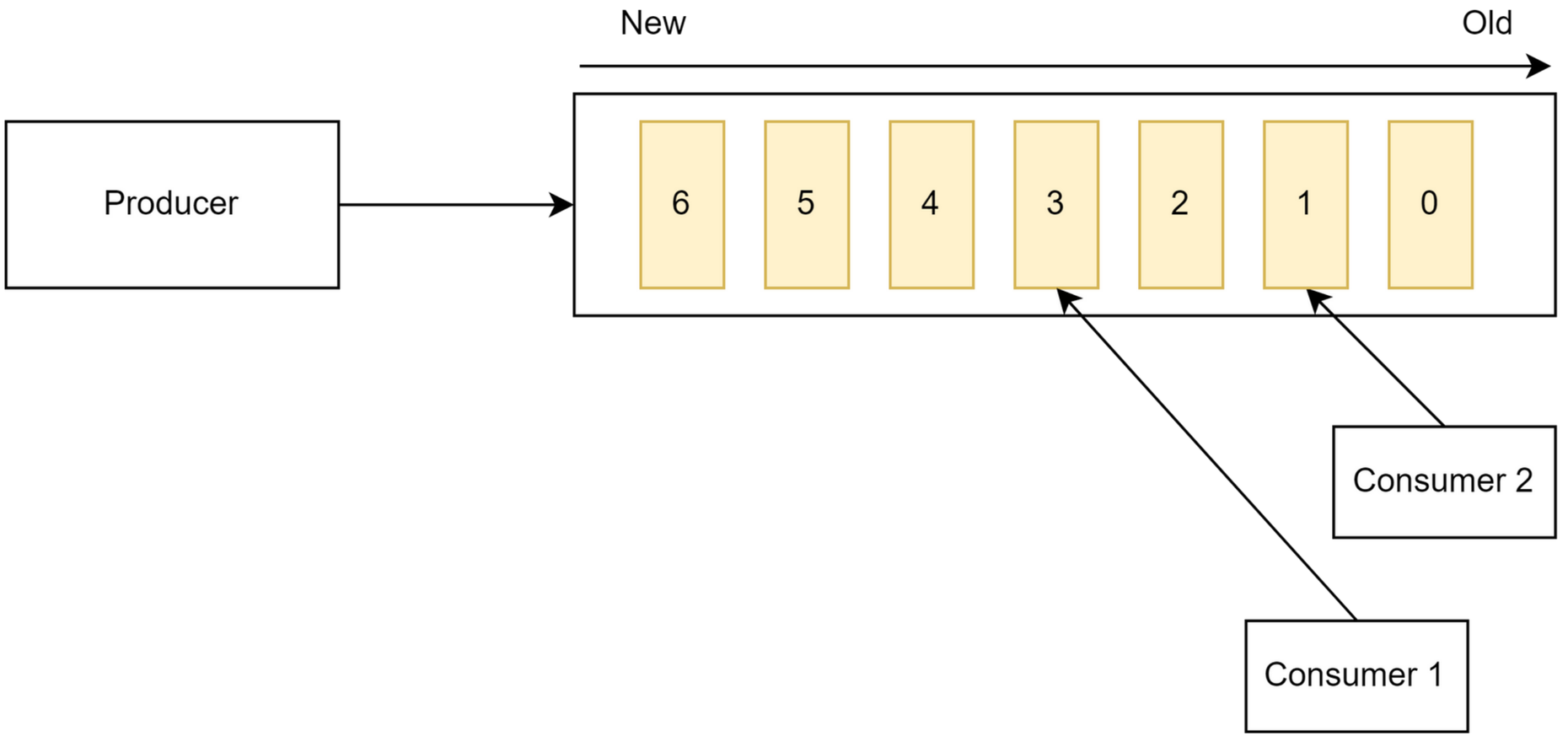
10 x 1000000 x 5 x 3 = 150,000,000 kiB = 146484 MiB = **143** GiB

Supported Compression Types in Kafka

Compression type	Compression ratio	CPU usage	Compression speed	Network bandwidth usage
Gzip	Highest	Highest	Slowest	Lowest
Snappy	Medium	Moderate	Moderate	Medium
Lz4	Low	Lowest	Fastest	Highest
Zstd	Medium	Moderate	Moderate	Medium

Consumer

- The Kafka consumer works by issuing "fetch" requests to the brokers leading the partitions it wants to consume.
- The consumer specifies its offset in the log with each request and receives back a chunk of log beginning from that position.
- The consumer thus has significant control over this position and can rewind it to re-consume data if need be.

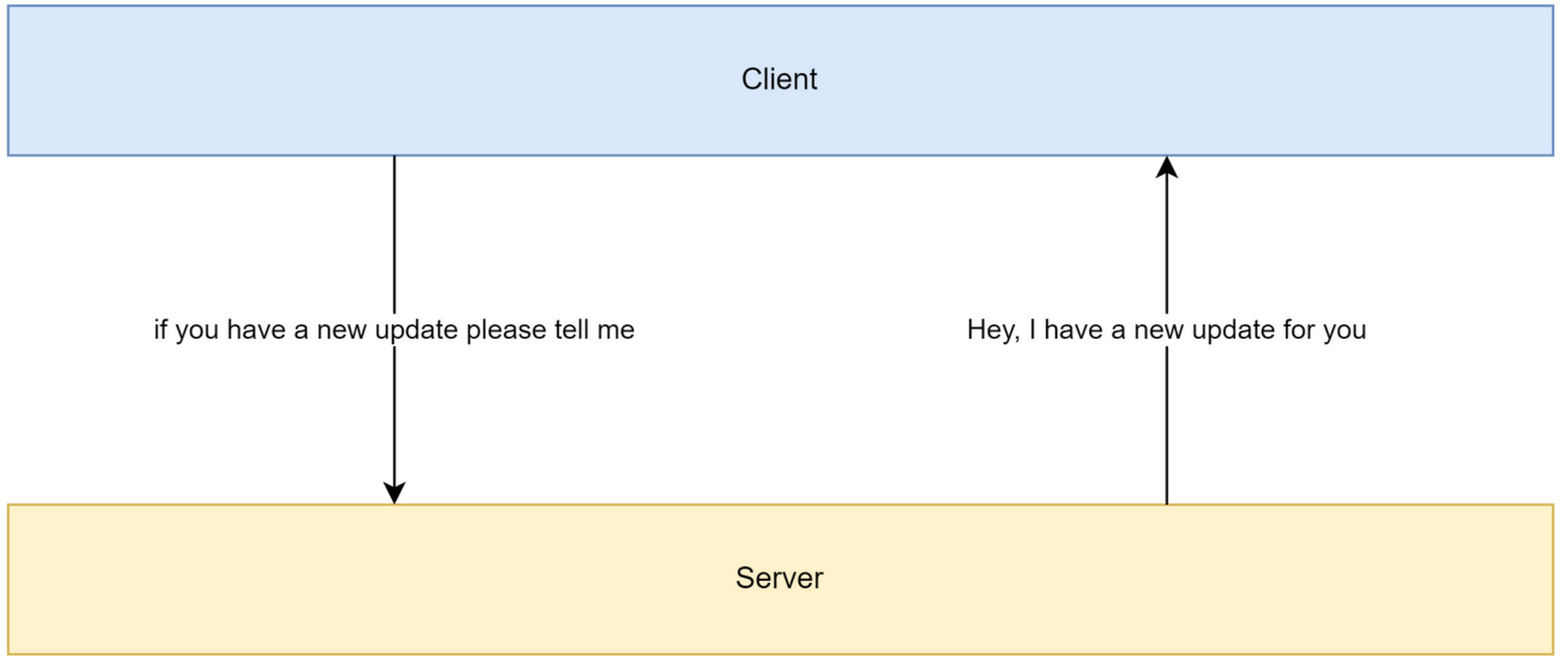

New
Old
Producer
6
5
4
3
2
1
0
Consumer 2
Consumer 1

Consumer Group

It a set of consumers which cooperate to consume data from some topics.

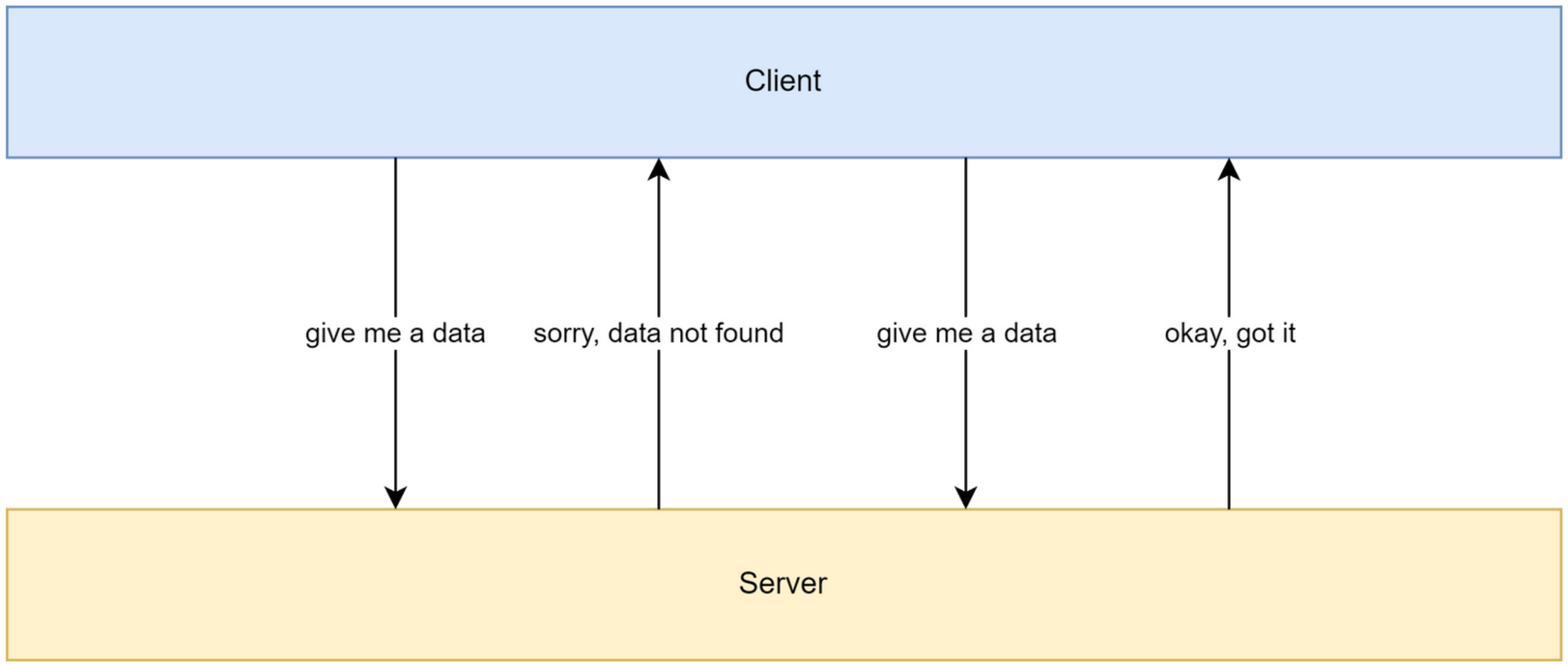
Best Practice: The same consumer group must have the same logic for each consumer.

Push-Based



Push architecture is **event driven**: the server pushes data to clients as updates become available.

Pull-Based



Pull architecture is **request driven:** the client sends the request, and the server responds accordingly.

Push vs. Pull

- **Push-based** system has difficulty dealing with diverse consumers as the broker controls the rate at which data is transferred.
- **Pull-based** the consumer to be able to consume at the maximum possible rate and simply falls behind and catches up when it can.

Kafka is **pull-based** because Kafka consumers pull data from the topic, different consumers can consume the messages at a different pace.

References

- https://kafka.apache.org/documentation
- https://medium.com/@tpbabparn/kafka-brief-concept-%E0%B9%81%E0%B8%A5%E0%B8%B0-reactor-kafka-in-spring-webflux-kotlin-f6249e978457
- https://medium.com/event-driven-utopia/understanding-kafka-topic-partitions-ae40f80552e8
- https://developer.ibm.com/articles/benefits-compression-kafka-messaging
- https://www.javatpoint.com/kafka-topic-replication
- https://medium.com/@parisa_moghaddam/a-brief-description-of-kafka-replication-mechanism-bafcb6ca949
- https://www.tutorialspoint.com/apache_kafka/apache_kafka_introduction.html