Apache Kafka

Presented by Riaz Farhanian - June-July 2022

What is Kafka?

What is Kafka?

- Apache Kafka is a distributed streaming platform
  - Publishes and subscribes to streams of records(Like other Message Queues)
  - Store streams of records in a fault tolerant durable way
  - Processes streams of records as they occur.

What is Kafka?

One of the popular definition of **Kafka is that It is a distributed Commit Log**

- As events happened in Micro-service Apps. These Applications put these events onto the log.
- Apache Kafka is a System what managing these logs.
- It categorize these logs with **Topics**.

What is Kafka?

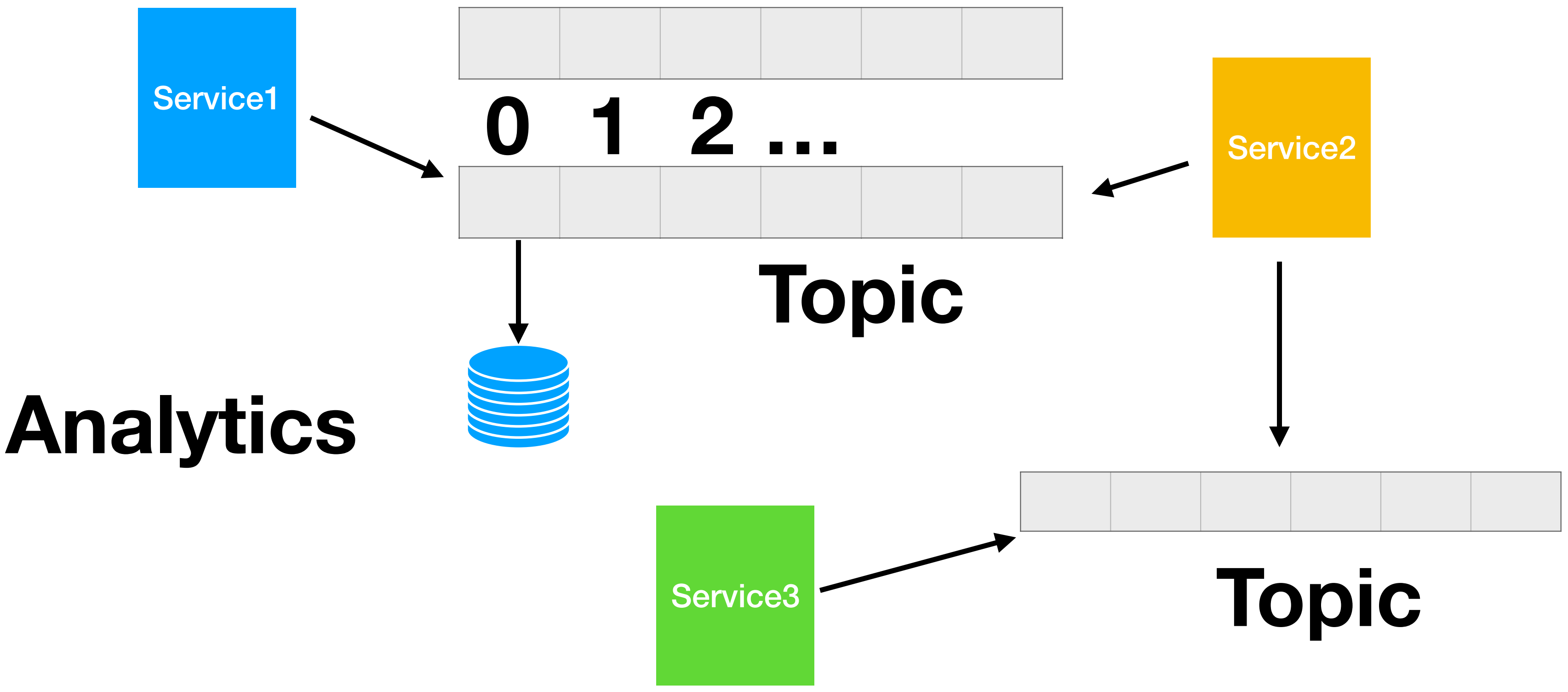
Kafka

- store events of micro services in orderly fashion,
- Write events to the disk,
- It can replicate them across disks to ensure that Events are not lost

What is Kafka?

Usually Micro-services exchange events through these topics or streams in realtime So these events can be processed in realtime and because of that we have Realtime Analytics
Like Recommendations or Alarms and Etc.

What is Kafka?



What is Kafka?

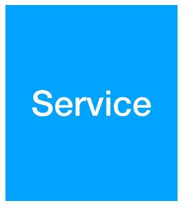
Kafka Streams API

Grouping

Aggregating

Filtering

Joining



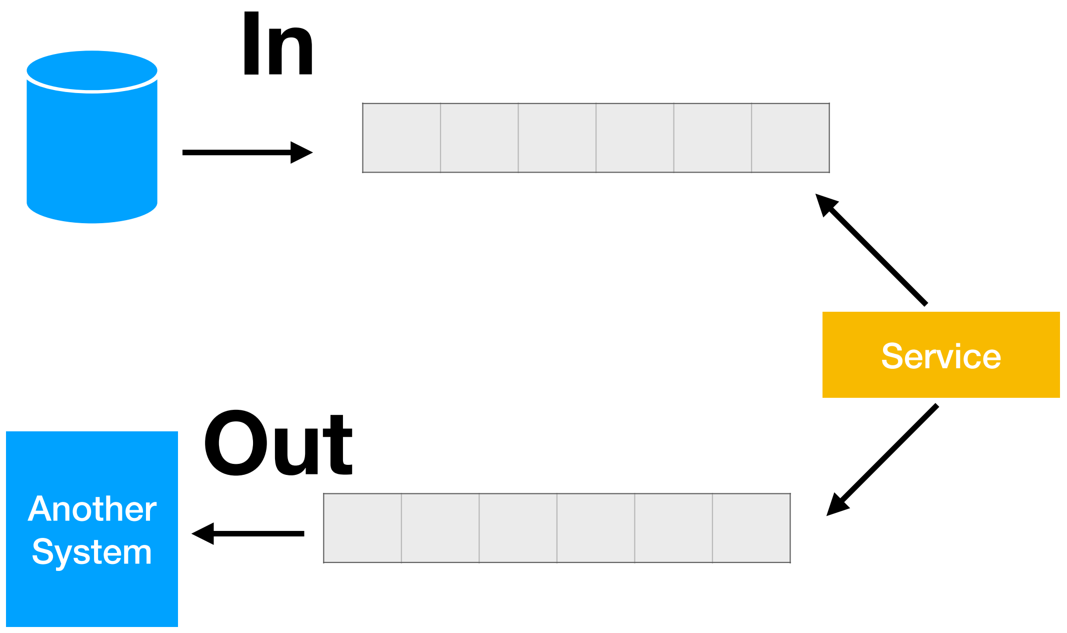
What is Kafka?

Kafka Connect

Kafka Connect works as a centralized data hub for simple data integration between databases, key-value stores, search indexes, and file systems.

What is Kafka?

Kafka Connect

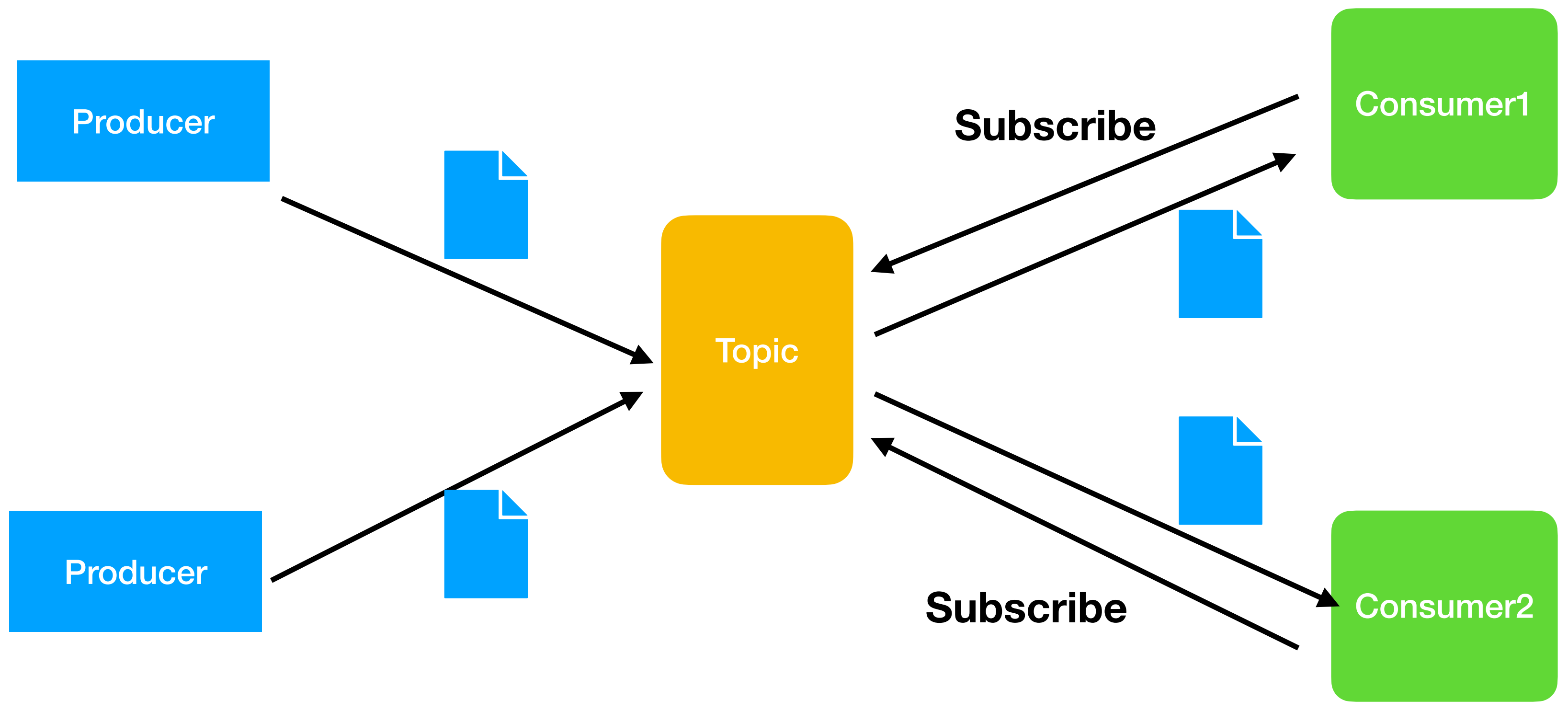


Why Kafka?

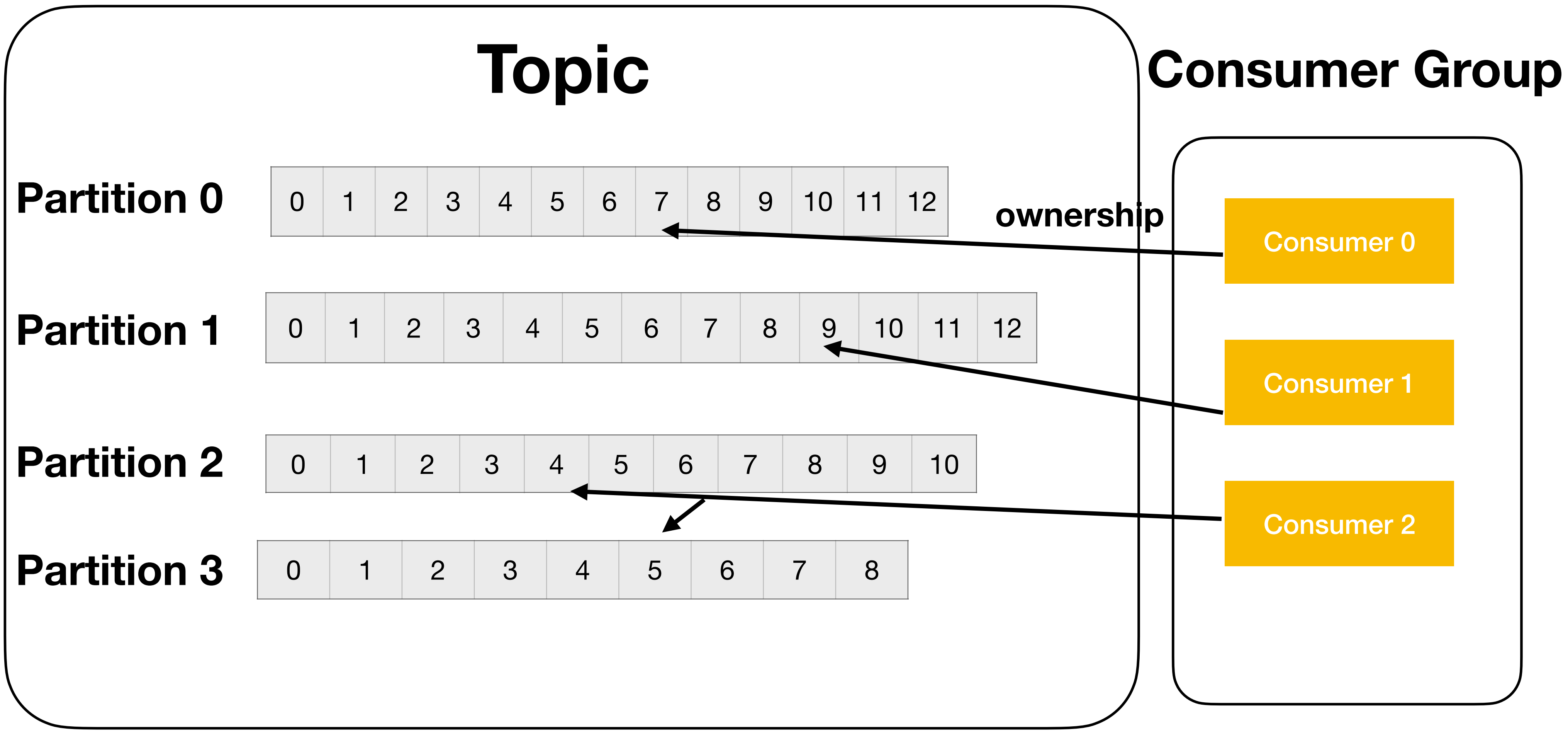
What are the advantages

Why Kafka?

Multiple Producers and Consumers

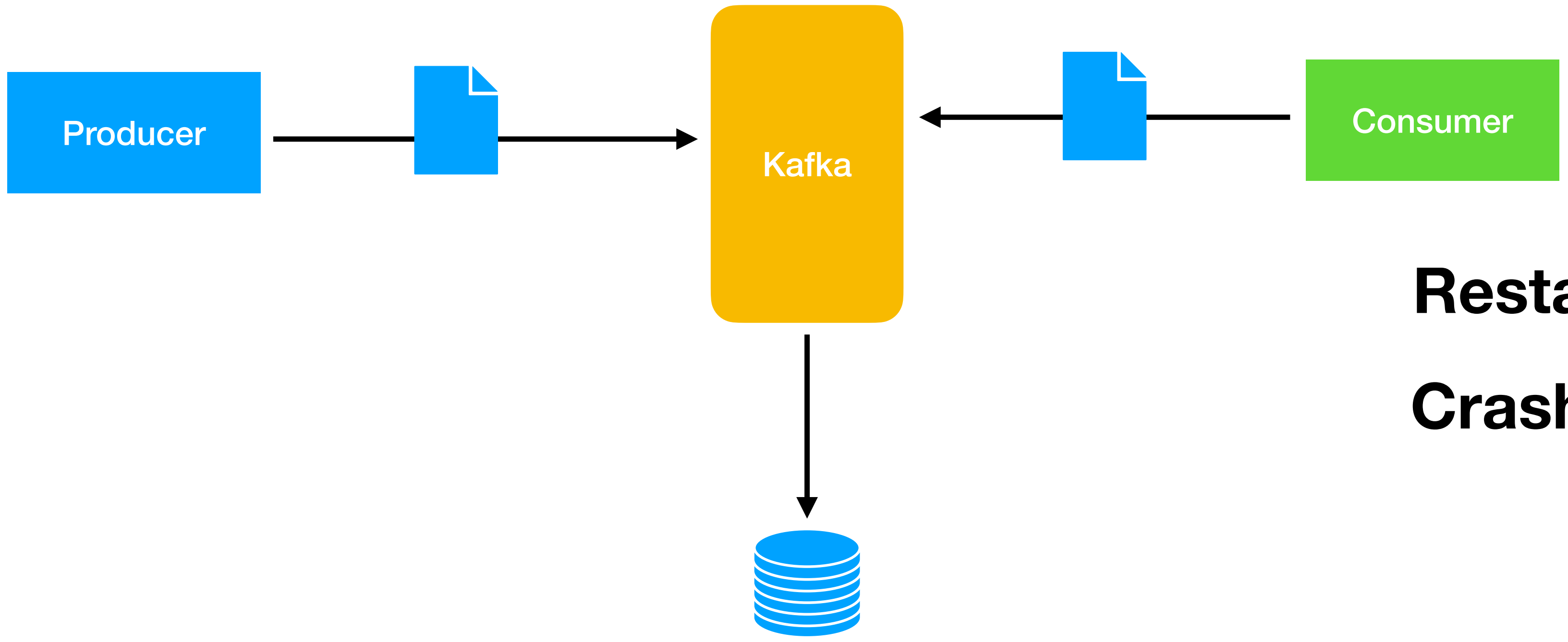


Why Kafka?


Topic
Consumer Group
Partition 0
0 1 2 3 4 5 6 7 8 9 10 11 12
Partition 1
0 1 2 3 4 5 6 7 8 9 10 11 12
Partition 2
0 1 2 3 4 5 6 7 8 9 10
Partition 3
0 1 2 3 4 5 6 7 8
ownership
Consumer 0
Consumer 1
Consumer 2

Why Kafka?

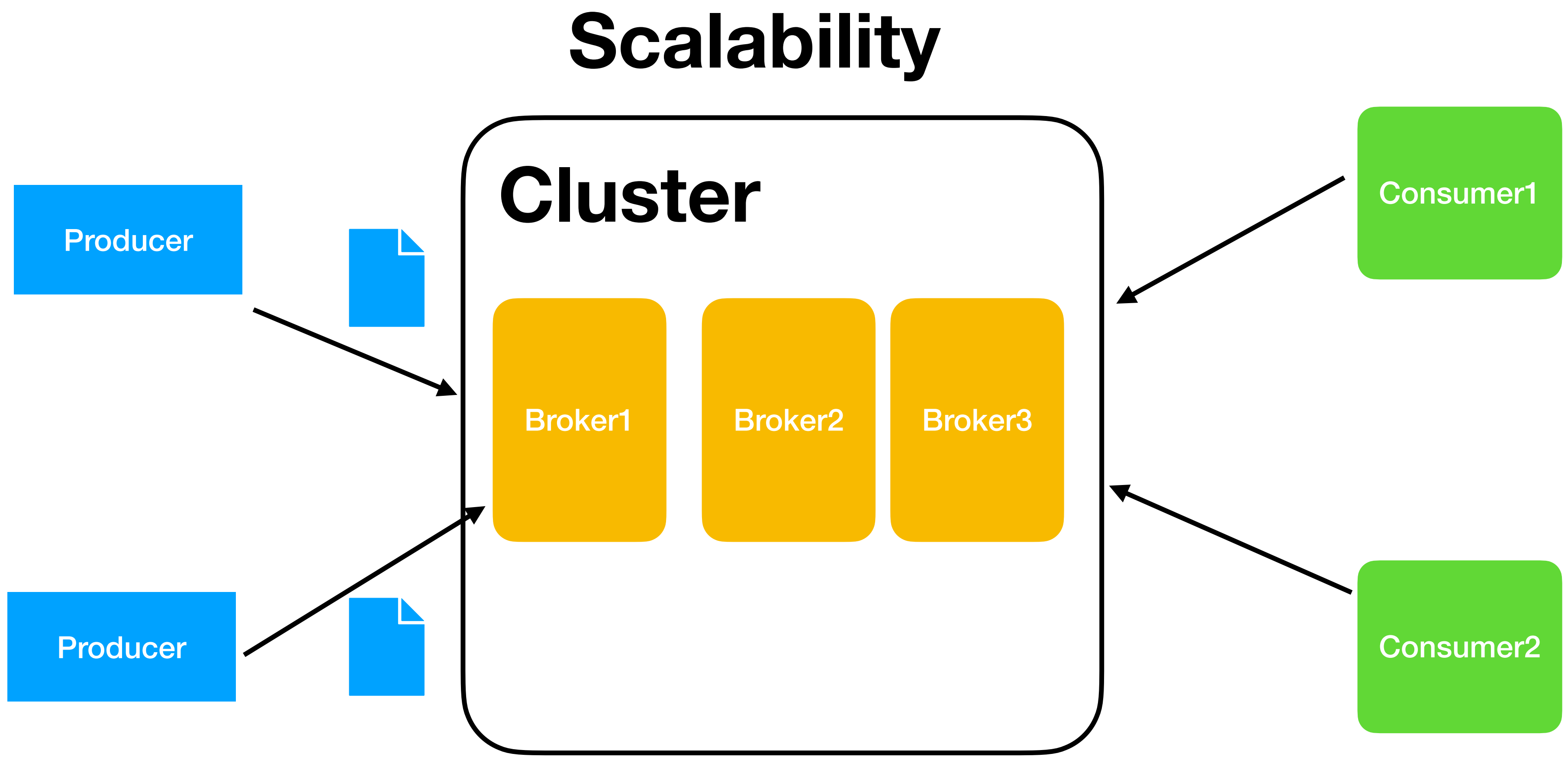
Disk Based Persistence



Restart

Crash

Why Kafka?


Scalability
Cluster
Producer
Producer
Broker1
Broker2
Broker3
Consumer1
Consumer2

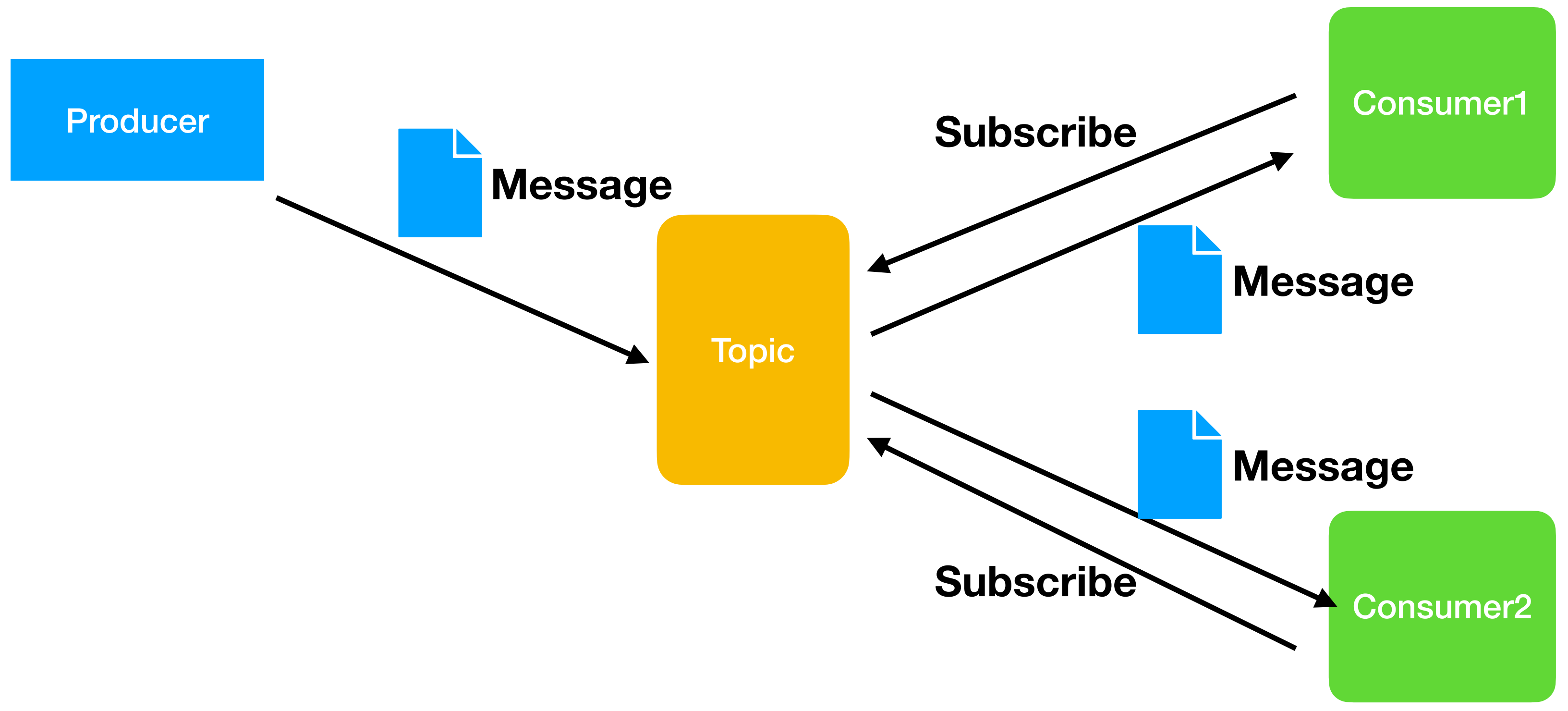
Why Kafka?

Performance

Use Cases

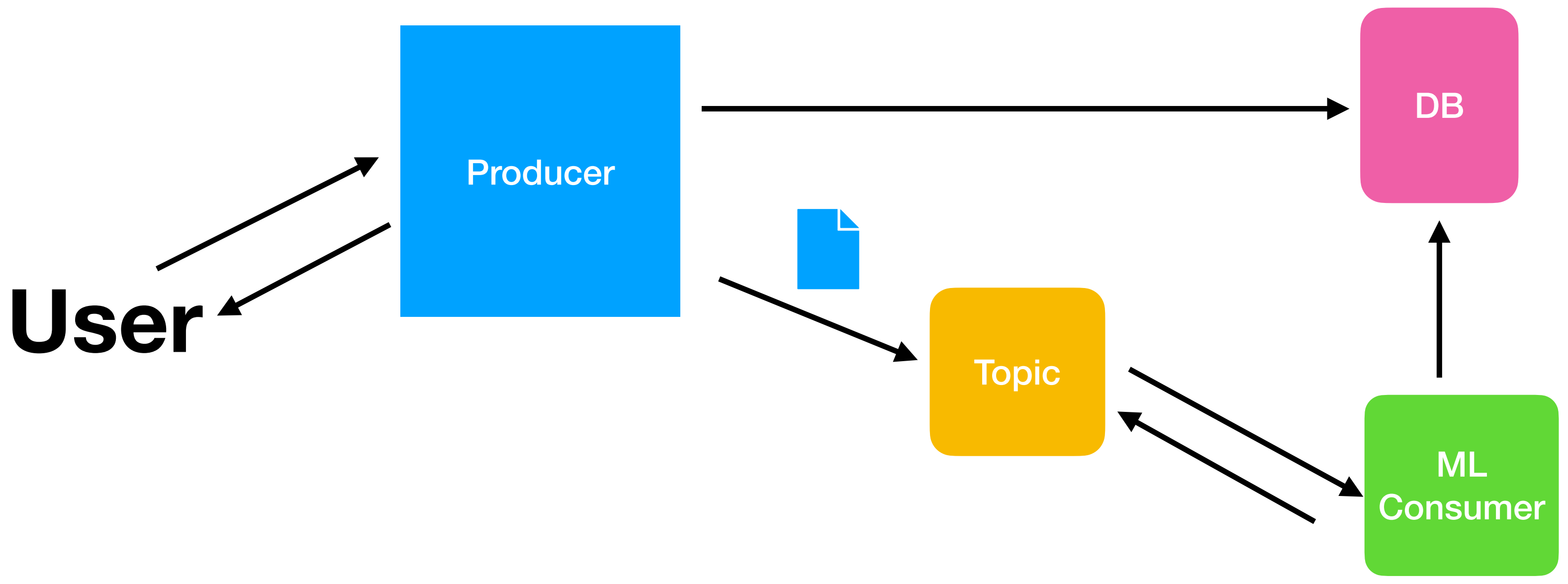
Use Cases

Pub-Sub



Use Cases

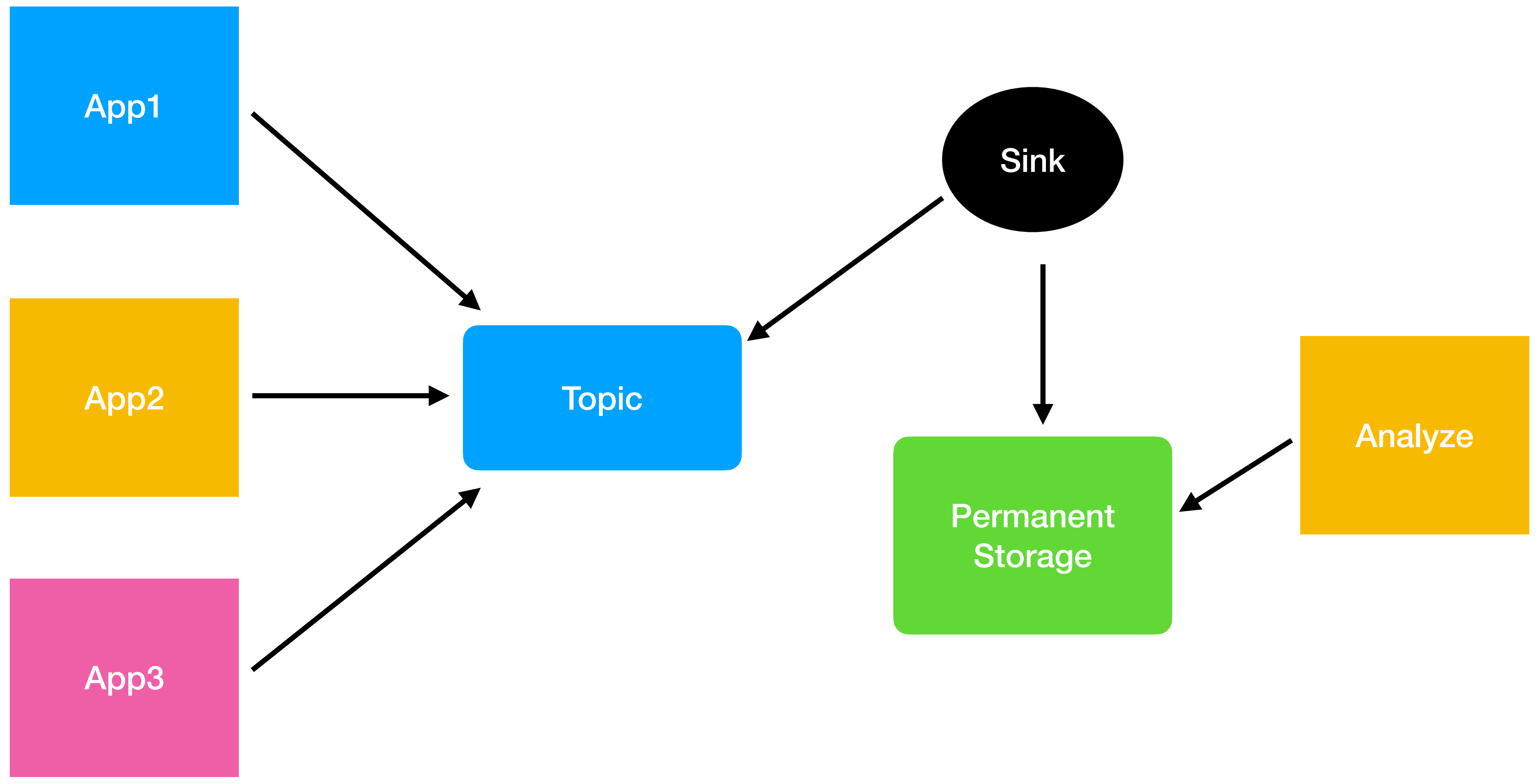
Activity Tracking



Amazon, Netflix and Linkedin

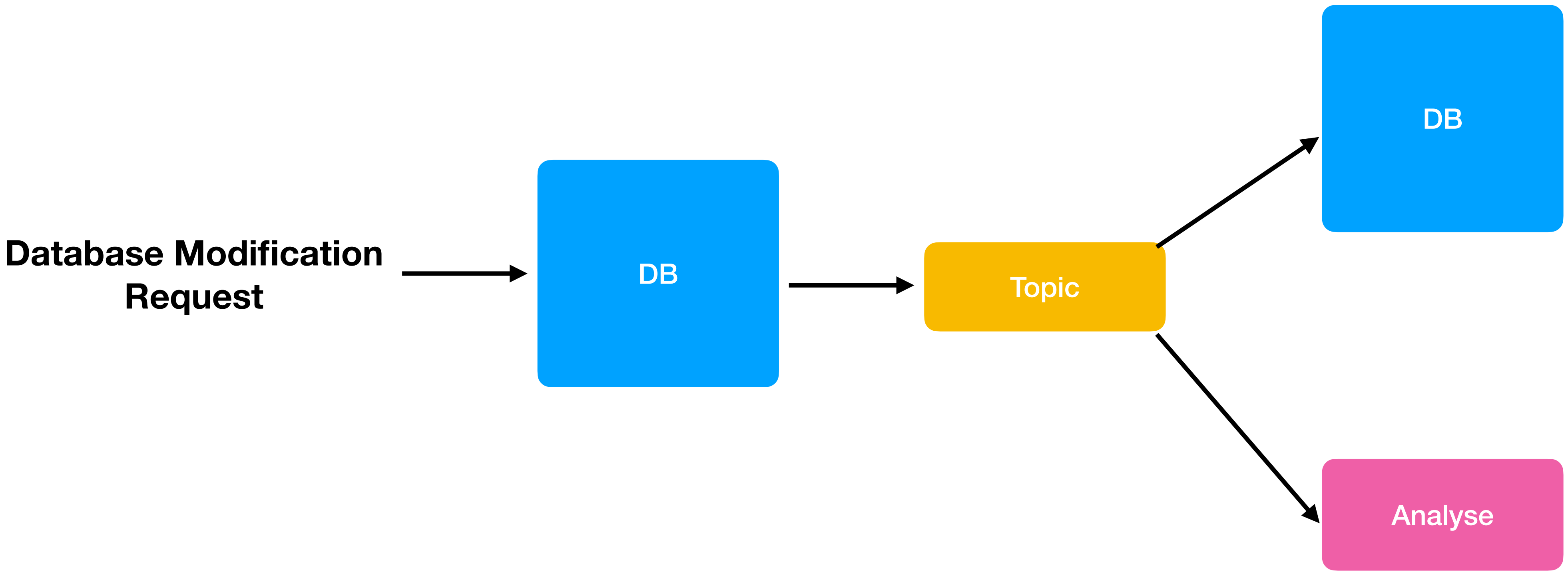
Use Cases

Metrics and Log Aggregation



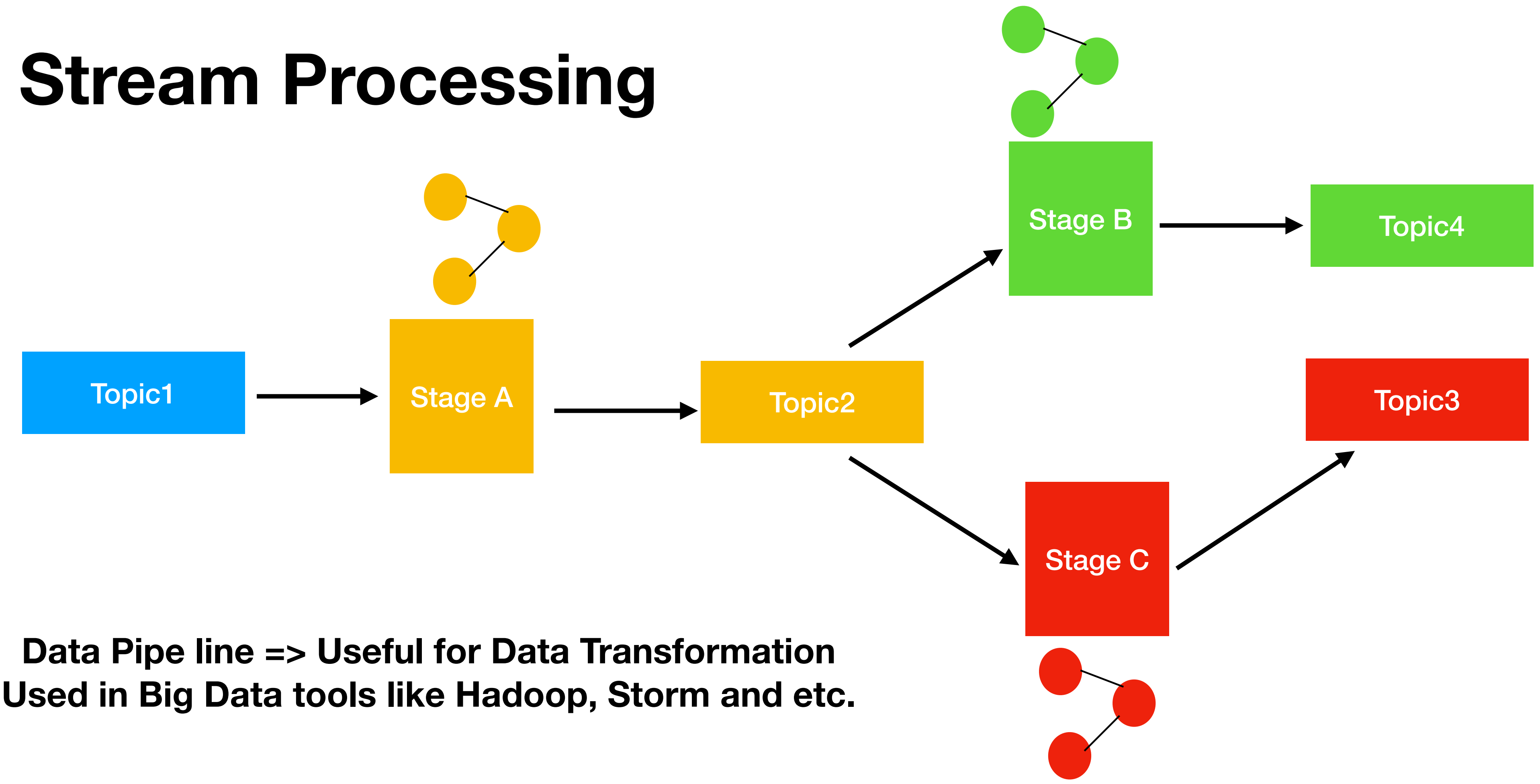
Use Cases

Commit Log



Use Cases

Stream Processing



Data Pipe line => Useful for Data Transformation
Used in Big Data tools like Hadoop, Storm and etc.

Use Cases

Industries

Data platforms,Micro Services

Monitoring Trucks,Cars

Location, Speed and Zone Monitoring

Retail,Hotel and Travel

React to Customer Interactions and generate Recommendations

Factories

Capture and analyze IOT Devices

Hospitals

Tracking and Monitoring Patient Information

Financial Transactions

Processing Bank and Stock Exchange Transaction

Use Cases

Companies

Twitter

Mobile App Performance and Analytics

5 Billion Sessions Per day

Netflix

Messaging Backbone

4000 Broker handling 700 Billion Events per day

Uber

uReplicator

Trillion Events for Log Aggregation Per day

Yahoo

Real Time Analytics

20 GB Events per Second

Pinterest

Real Time Ads

2000 Brokers handling 8 Billion Events per day

Kafka Architecture

Kafka Architecture

Key Components are:

- Broker
- Zoo Keeper
- Producer
- Consumer

Kafka Architecture

Kafka Broker

- Receives messages from producers and stores them on disk keyed by unique **offset**.
- Allows consumers to fetch messages by topic, partition and offset.
- A single broker hosts topic partitions of one or more topics actually

Kafka Architecture

Kafka Broker

- A **Kafka server**, a **Kafka broker** and a **Kafka node** all refer to the same concept and are synonyms

Kafka Architecture

Kafka Broker(In Cluster)

- Kafka brokers can create a Kafka cluster by sharing information between each other directly or indirectly using Zookeeper.
- A Kafka cluster has exactly one broker that acts as the Controller.

Kafka Architecture

Zookeeper

- ZooKeeper is used in distributed systems for service synchronization and as a naming registry
- With Apache Kafka, ZooKeeper is primarily used to track the status of nodes in the Kafka cluster and maintain a list of Kafka topics and messages.
- Kafka and ZooKeeper are bundled in installation.

Kafka Architecture

Zookeeper History

- ZooKeeper was originally developed by Yahoo to address the bugs that can arise with distributed, big data applications by storing the status of processes running on clusters.
- Like Kafka, ZooKeeper is an open source technology under the Apache License.

Kafka Architecture

ZooKeeper has five primary functions

- Controller election
- Cluster membership
- Topic configuration
- Access control lists
- Quotas

Kafka Architecture

ZooKeeper Controller Function:

- The controller is the broker responsible for maintaining the leader/follower relationship for all partitions.
- If ever a node shuts down, ZooKeeper ensures that other replicas take up the role of partition leaders replacing the partition leaders in the node that is shutting down.

Kafka Architecture

ZooKeeper Cluster Membership:

- ZooKeeper keeps a list of all functioning brokers in the cluster.

Kafka Architecture

ZooKeeper Topic Configuration:

- ZooKeeper maintains the configuration of all topics, including the list of existing topics, number of partitions for each topic, location of the replicas, configuration overrides for topics, preferred leader node, among other details.

Kafka Architecture

ZooKeeper ACL:

ZooKeeper also maintains the ACLs for all topics. This includes who or what is allowed to read/write to each topic, list of consumer groups, members of the groups, and the most recent offset each consumer group received from each partition.

Kafka Architecture

ZooKeeper Quotas:

- ZooKeeper accesses how much data each client is allowed to read/write.

Kafka Architecture

Kafka Producer:

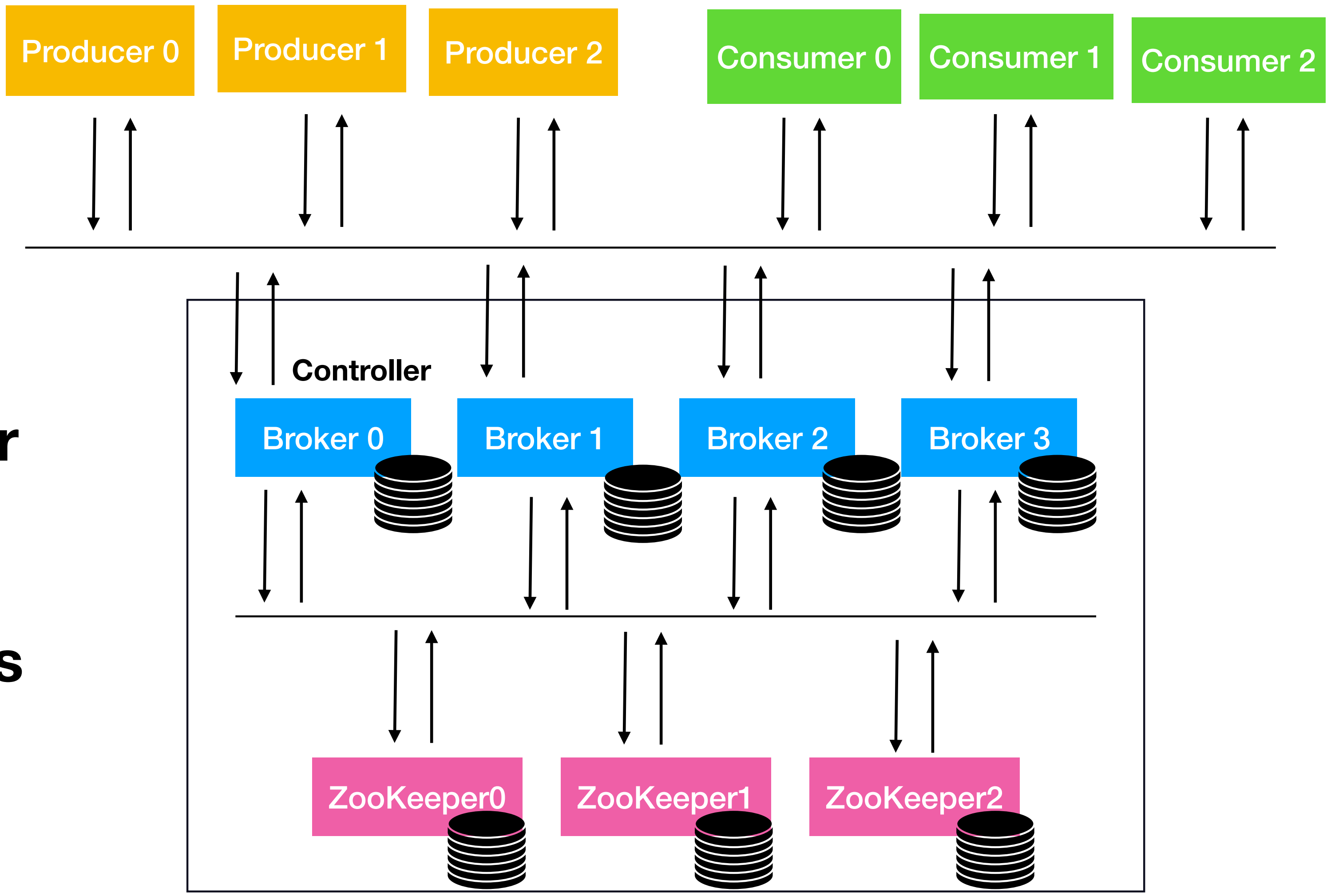
- Applications that send data into topics are known as Kafka producers.
- A Kafka producer sends messages to a topic, and messages are **distributed to partitions** according to a mechanism such as **key hashing**

Kafka Architecture

Kafka Consumer:

- Applications that read data from Kafka topics are known as consumers.
- Consumers can read from one or more partitions at a time in Apache Kafka, and data is read in order **within each partition** as shown below.
- Kafka consumers will only consume data that was produced after it first connected to Kafka.
- Kafka consumers are also known to implement a "pull model". This means that Kafka consumers must request data from Kafka brokers in order to get it

Kafka Architecture



Broker

ZooKeeper

Producers

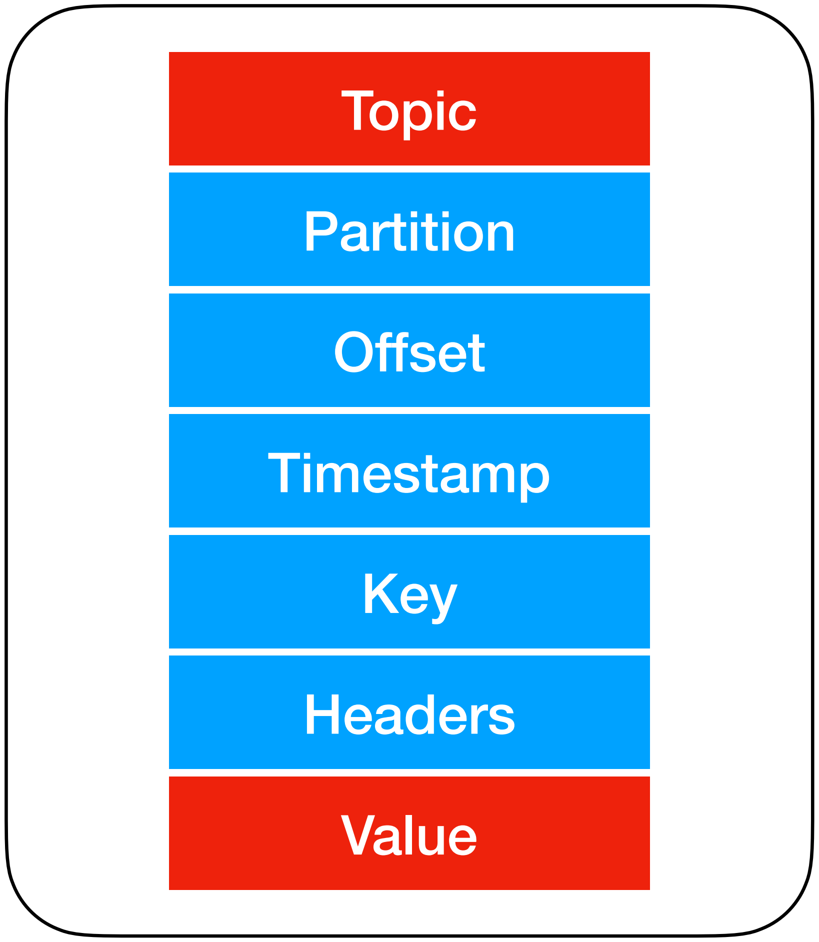
Consumers

Kafka Record

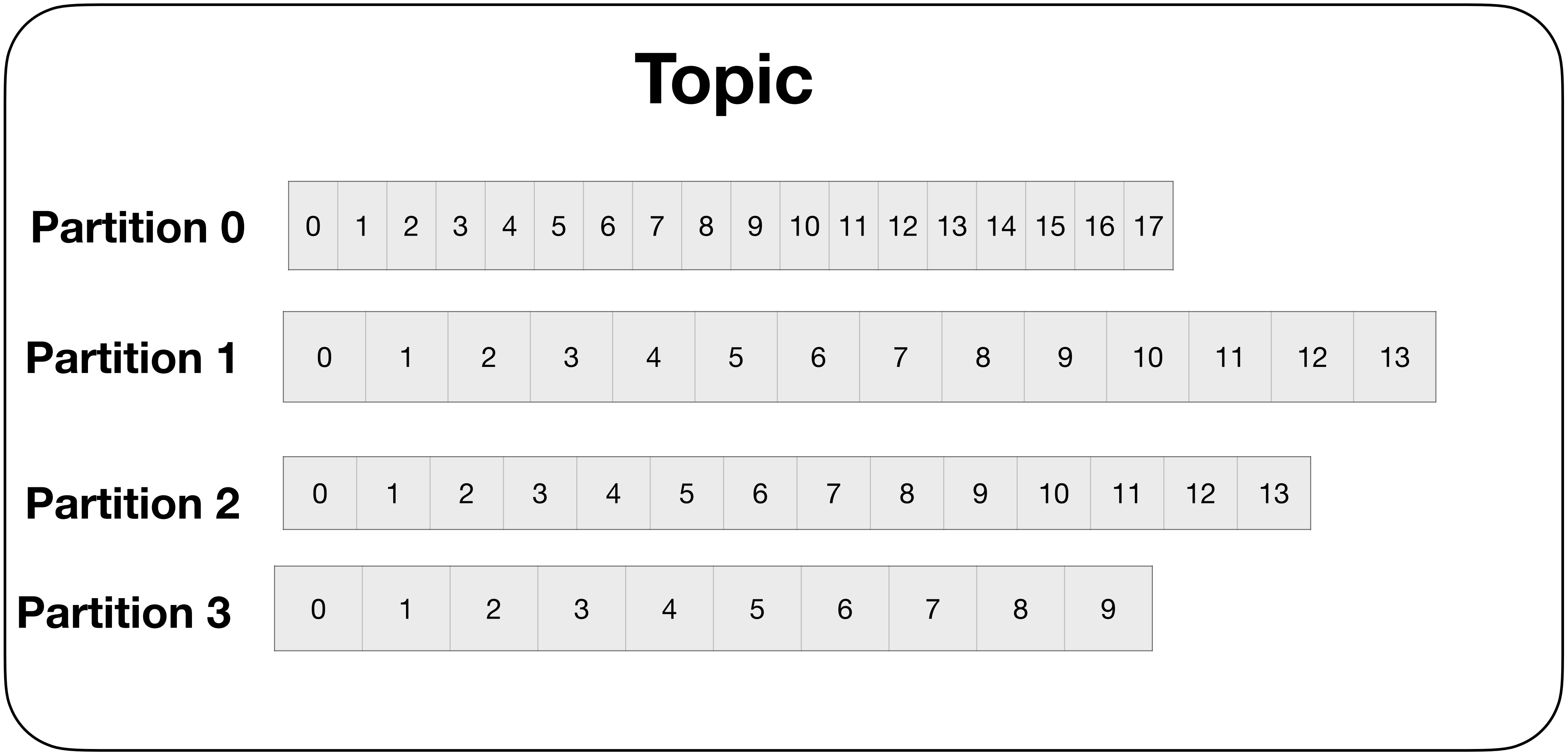
Kafka Record

- A key/value pair to be sent to Kafka.
- This consists of a **topic name** to which the record is being sent, an ***optional partition number***, and ***an optional key*** and value.

Kafka Record



Topics Partitions and Offsets



Kafka Record

Partition

- If a valid partition number is specified that partition will be used when sending the record.
- If no partition is specified but a **key** is present a partition will be chosen using a hash of the key.
- If neither key nor partition is present a partition will be assigned in a round-robin fashion.

Kafka Record

Time Stamp

- The record also has an associated timestamp.
- If the user did not provide a timestamp, the **producer** will stamp the record with its current time.
- The timestamp eventually used by Kafka depends on the timestamp type configured for the topic.

Kafka Record

Time Stamp

- If the topic is configured to use **CreateTime**, the timestamp in the producer record will be used by the broker.
- If the topic is configured to use **LogAppendTime**, the timestamp in the producer record will be overwritten by the broker with the broker local time when it appends the message to its log.

Kafka Replication

Kafka Replication

- **Kafka Replication** means having multiple copies of the data, spread across multiple servers/brokers.

- This helps in maintaining high availability in case one of the brokers goes down and is unavailable to serve the requests.

Kafka Replication

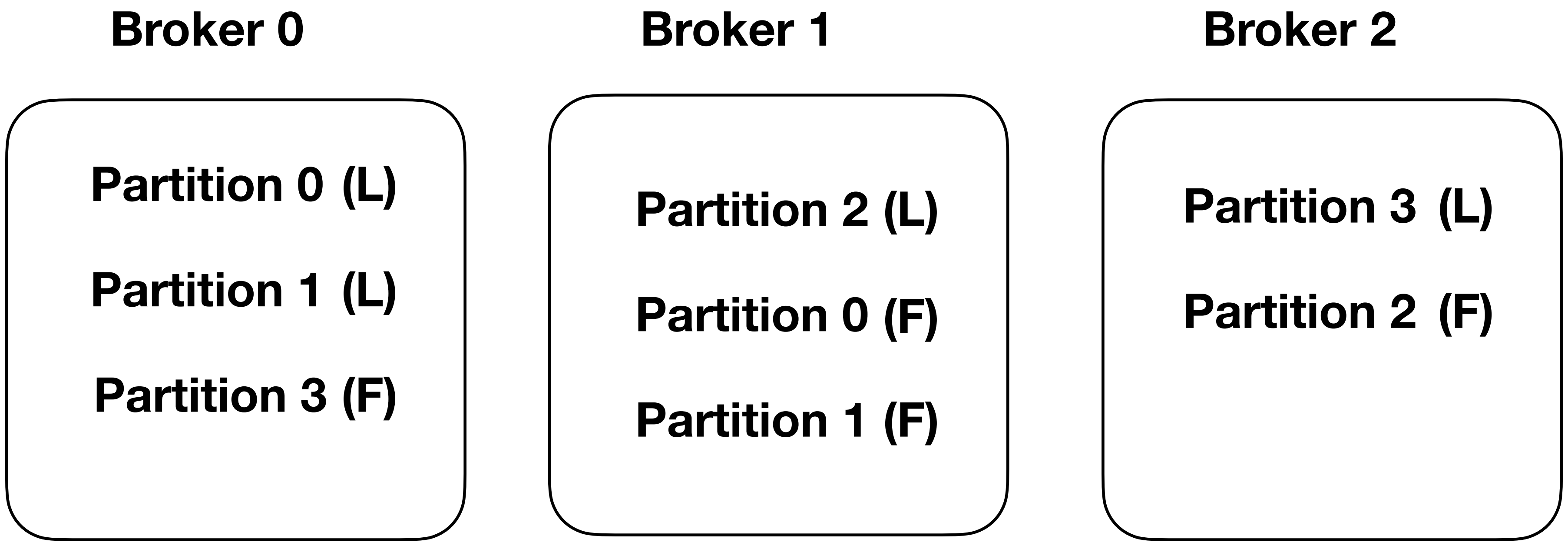
- Kafka Replication is allowed at the partition level, copies of a partition are maintained at multiple broker instances using the partition’s Write-Ahead Log.

- Amongst all the replicas of a partition, Kafka designates one of them as the “**Leader**” partition and all other partitions are followers or **“in-sync”** partitions.

Kafka Replication

- The Leader is responsible for receiving as well as sending data, for that partition.
- The total number of replicas including the leader is equal to the **Replication factor**.
- To maintain these clusters and the topics/partitions within, Kafka use the **Zookeeper**.

Replication



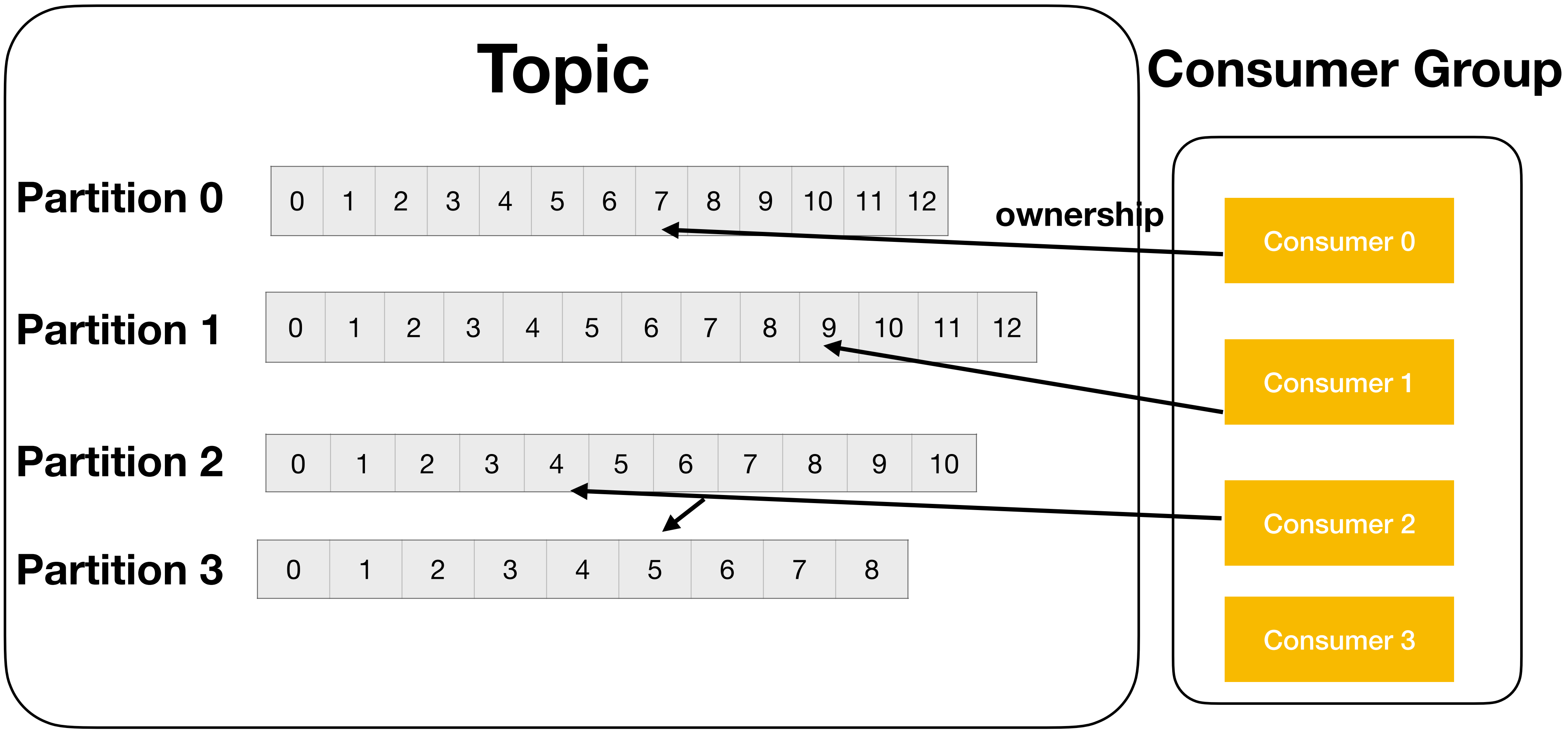
Leader Follower

Replication Factor 1 2 3

Consumer Group

Consumer Group

- A **consumer group** is a set of consumers which cooperate to consume data from some topics.
- The partitions of all the topics are divided among the consumers in the group.
- As new group members arrive and old members leave, the partitions are re-assigned so that each member receives a proportional share of the partitions. This is known as **rebalancing the group**.


Topic
Consumer Group
Partition 0
0 1 2 3 4 5 6 7 8 9 10 11 12
Partition 1
0 1 2 3 4 5 6 7 8 9 10 11 12
Partition 2
0 1 2 3 4 5 6 7 8 9 10
Partition 3
0 1 2 3 4 5 6 7 8
ownership
Consumer 0
Consumer 1
Consumer 2
Consumer 3

Kafka Batches

Kafka Batches

- By default, Kafka producers try to send records as soon as possible.
- A producer will have up to **5 requests** in flight, meaning up to 5 message batches will be sent at the same time.

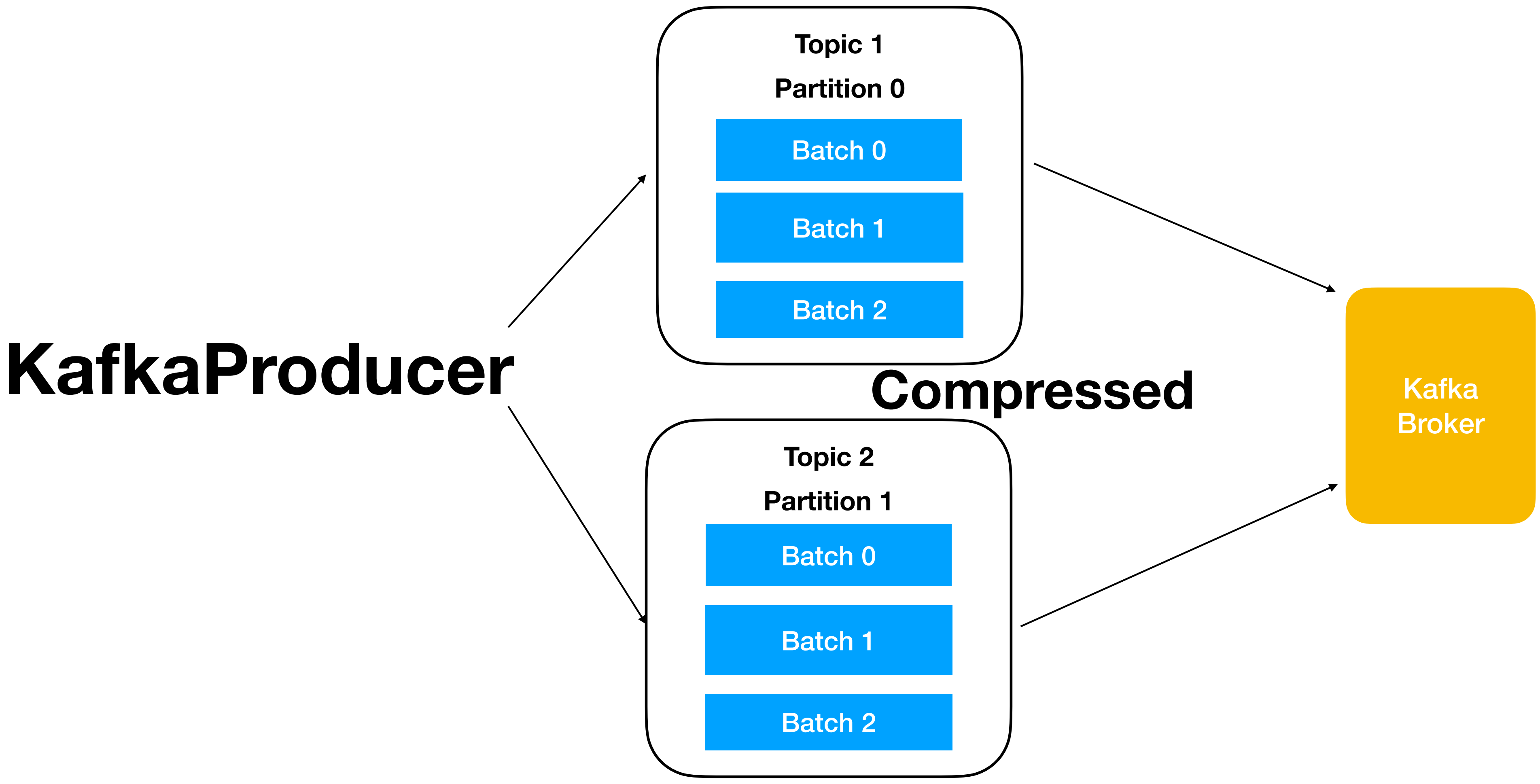
Kafka Batches

- If more messages have to be sent while others are in flight, producers will start batching messages while the previous message requests are finishing.

Kafka Batches

- This smart batching allows Kafka to increase throughput while maintaining very low latency.
- Batches have a higher compression ratio so a better disk and networking efficiency.

Kafka Batches


KafkaProducer
Topic 1
Partition 0
Batch 0
Batch 1
Batch 2
Compressed
Topic 2
Partition 1
Batch 0
Batch 1
Batch 2
Kafka Broker