

Go Kafka

Pallat Anchaleechamaikorn

Technical Coach

Infinitas by KrungThai

Arise by Infinitas

yod.pallat@gmail.com

<https://github.com/pallat>

<https://dev.to/pallat>

<https://go.dev/tour> (Thai)

<https://github.com/uber-go/guide> (Thai)



Apache Kafka

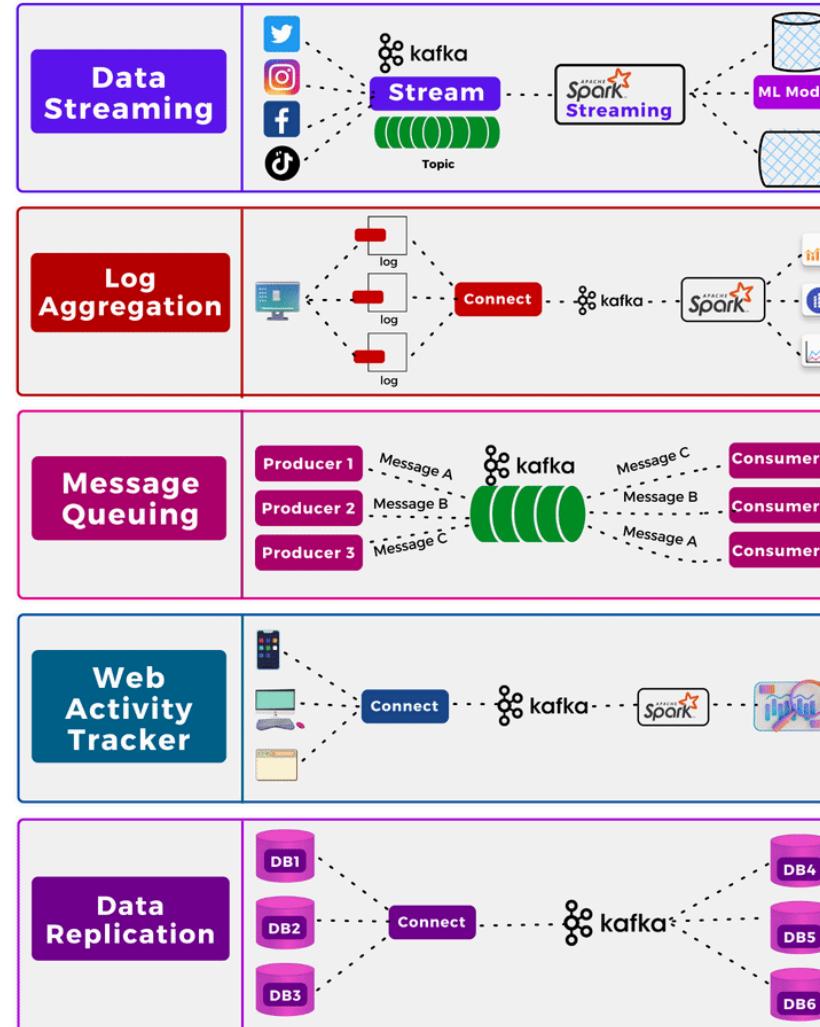
<https://kafka.apache.org/>

<https://www.confluent.io/>

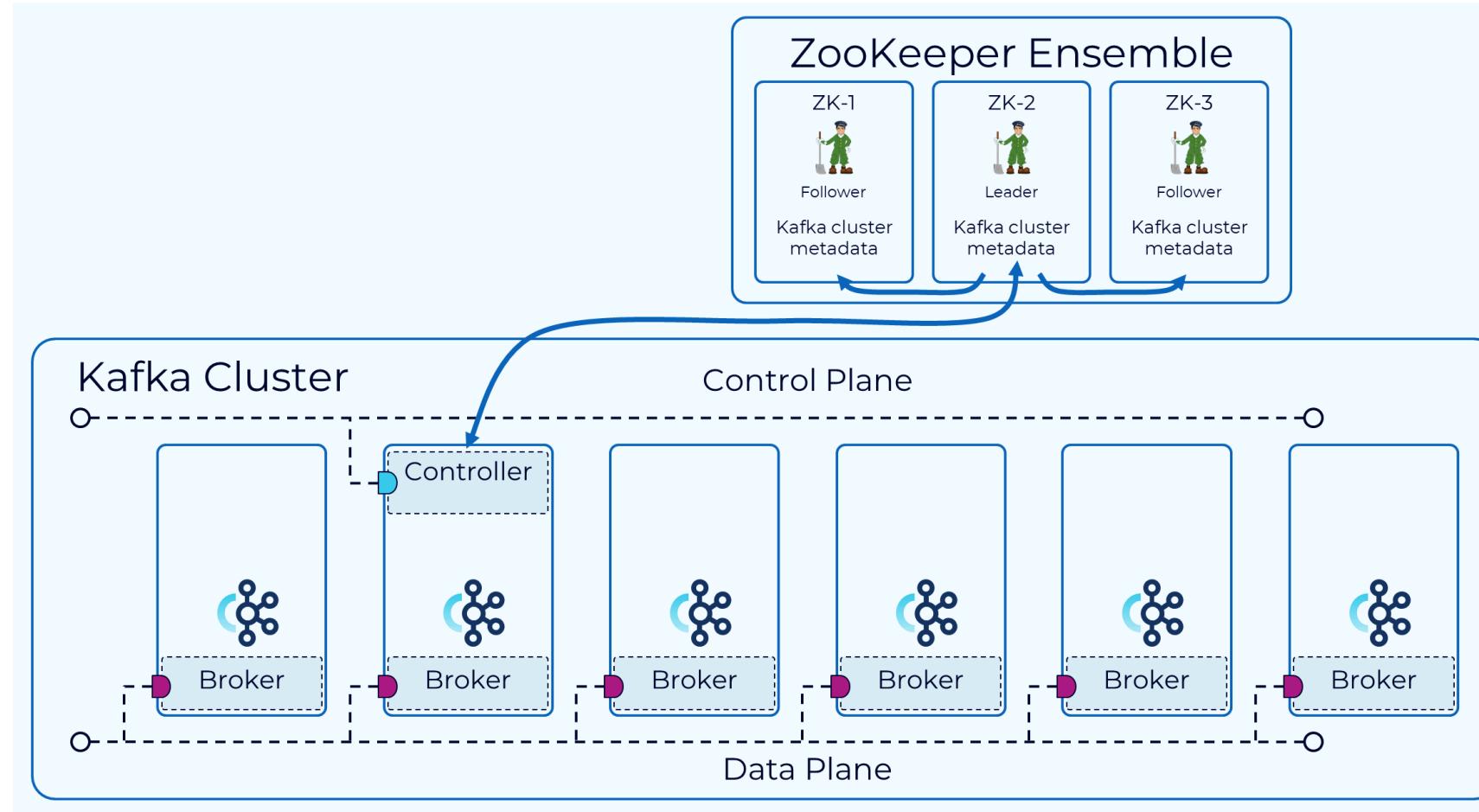
Apache Kafka is an open-source distributed event streaming platform used by thousands of companies for high-performance data pipelines, streaming analytics, data integration, and mission-critical applications.



TOP 5 KAFKA USE CASES

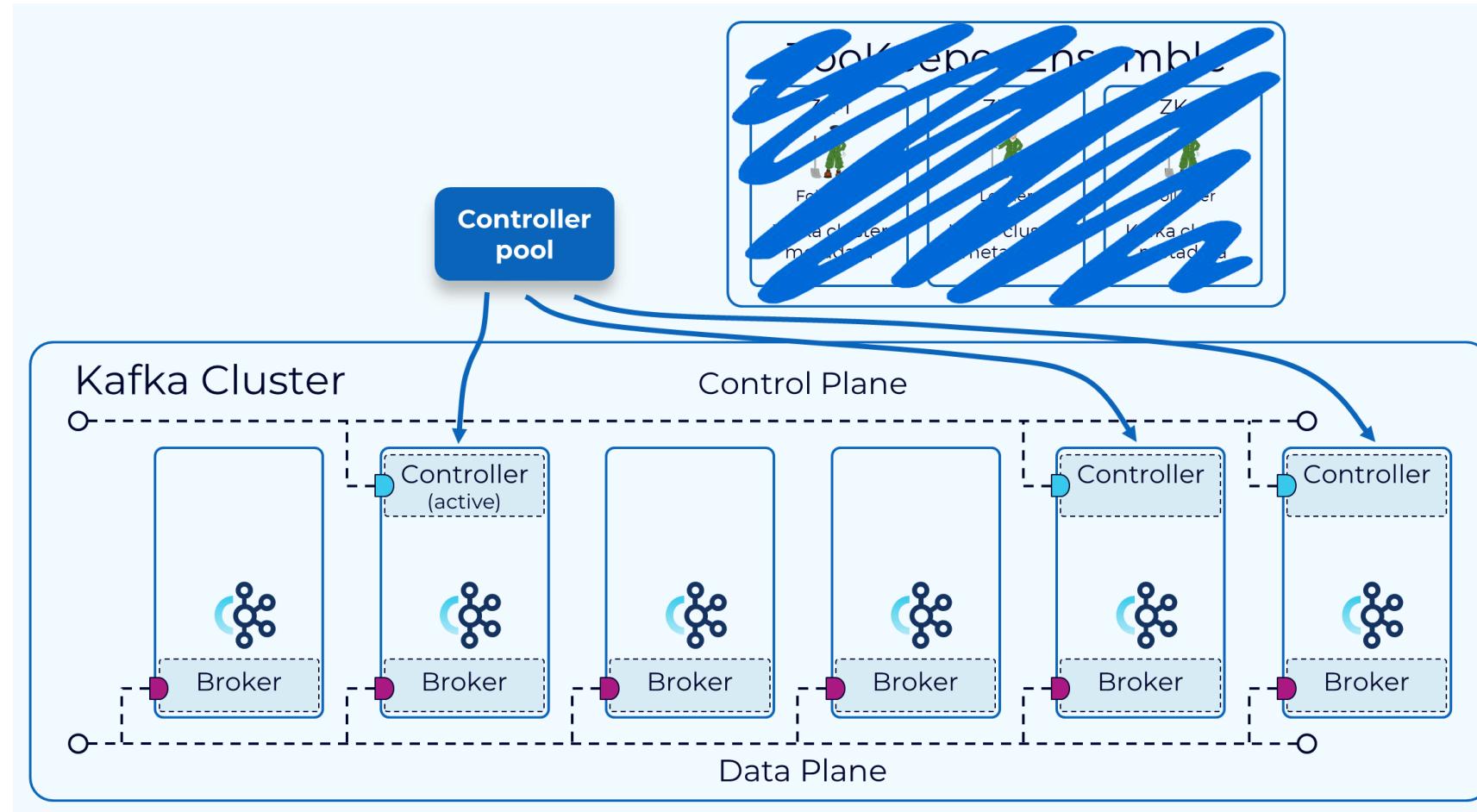


Kafka Architecture with ZooKeeper

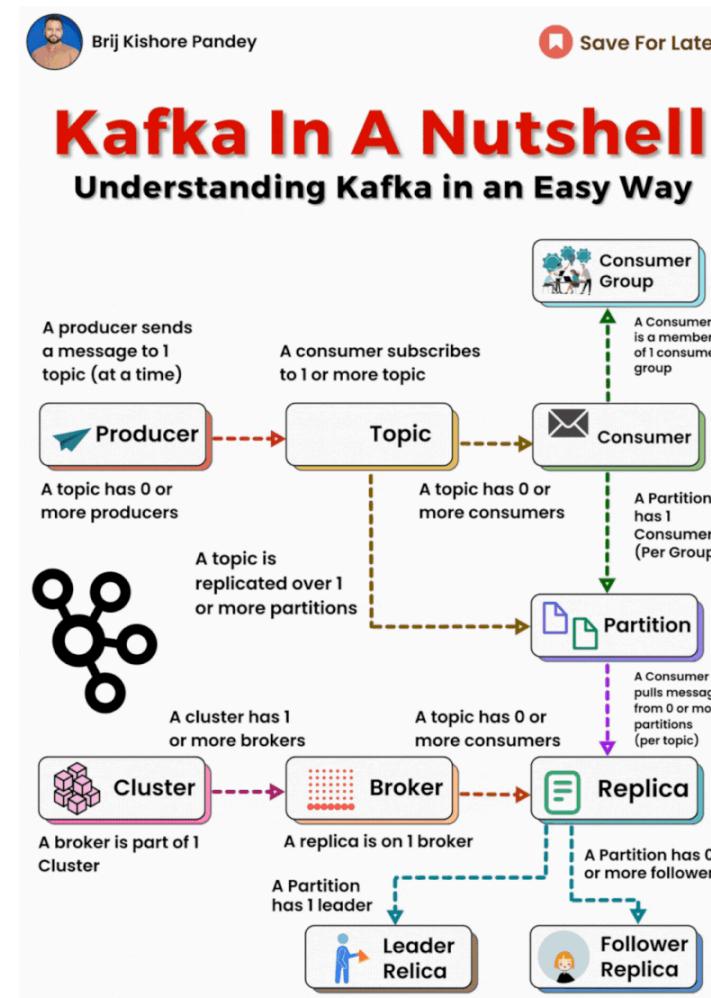


<https://developer.confluent.io/courses/architecture/control-plane/>

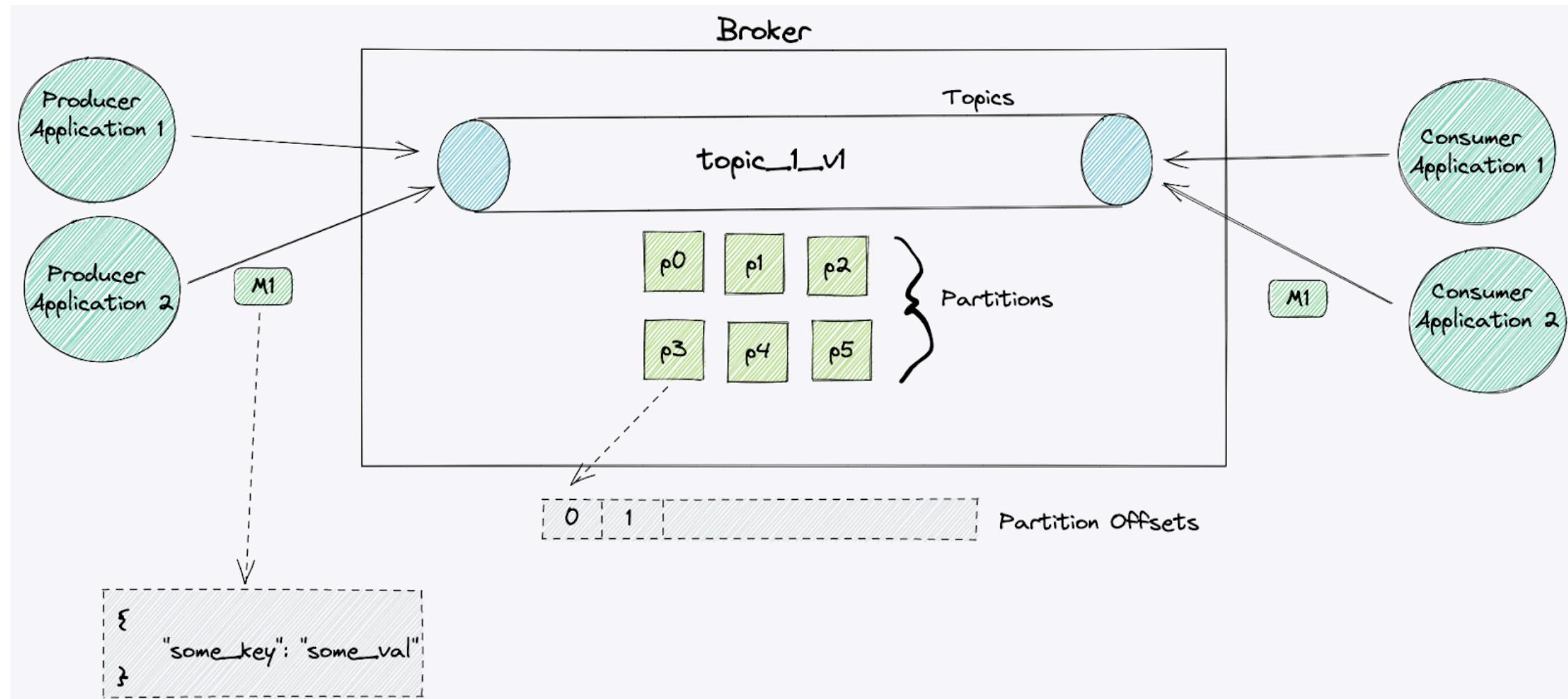
Kafka Architecture with Kraft



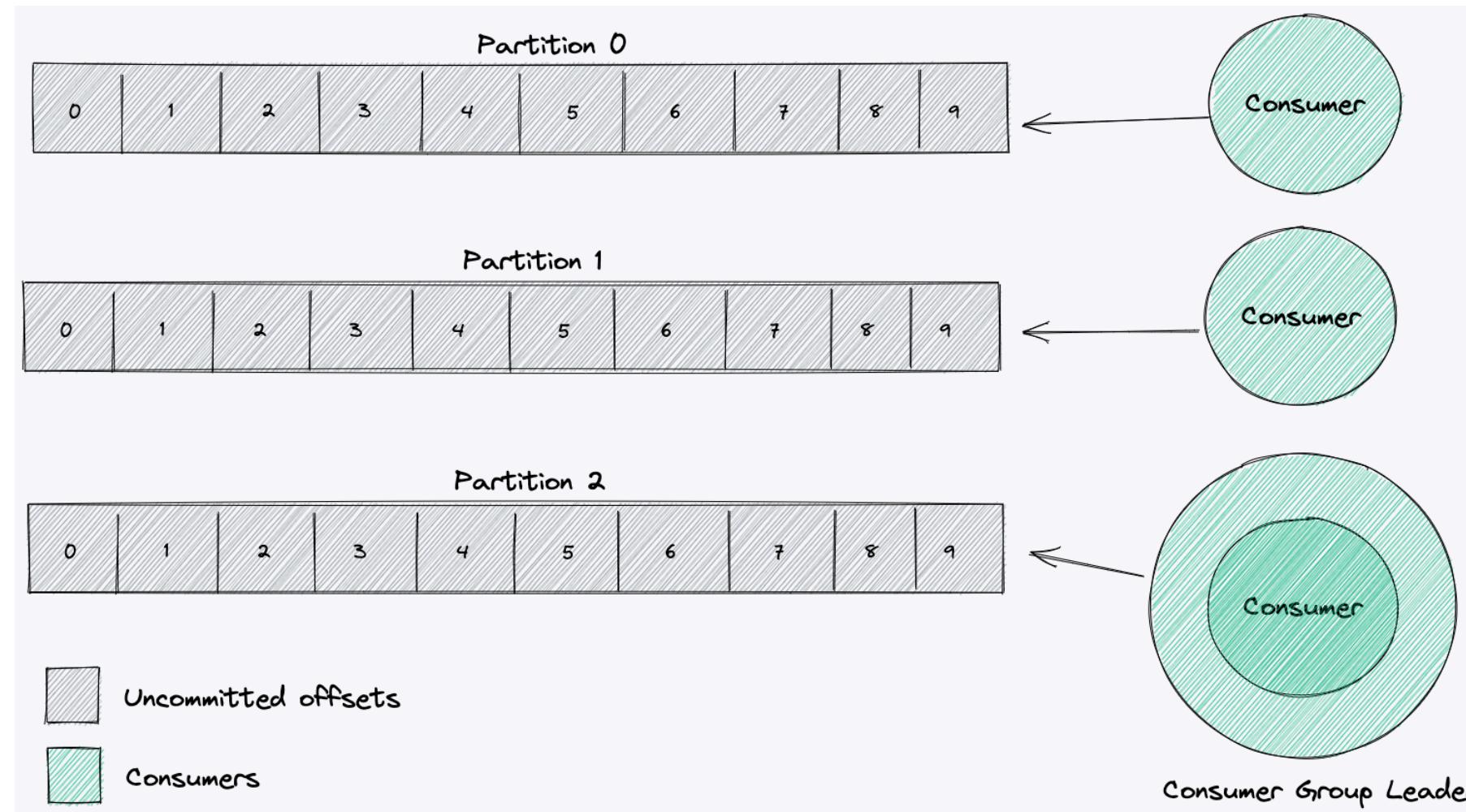
<https://www.confluent.io/blog/what-is-kraft-and-how-do-you-use-it/>



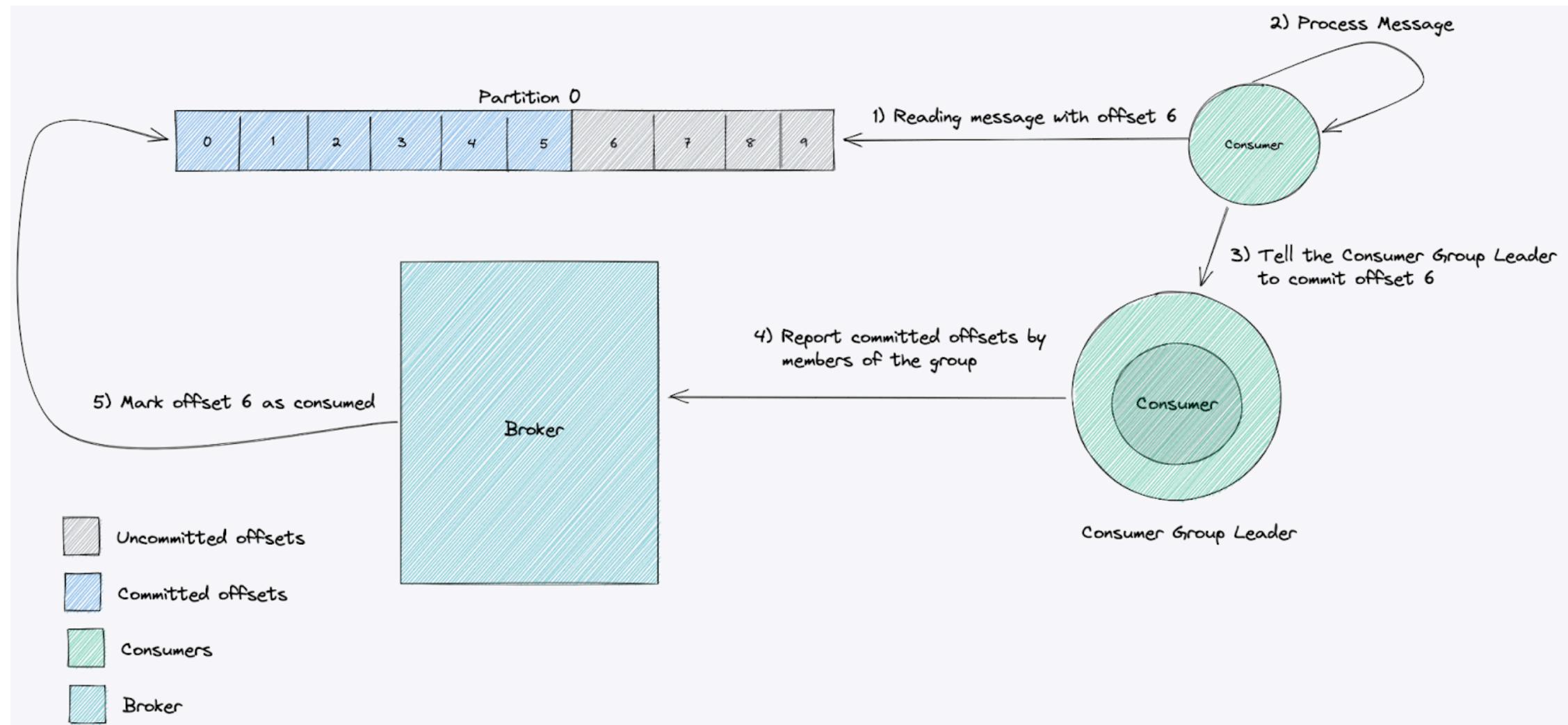
How Kafka works?



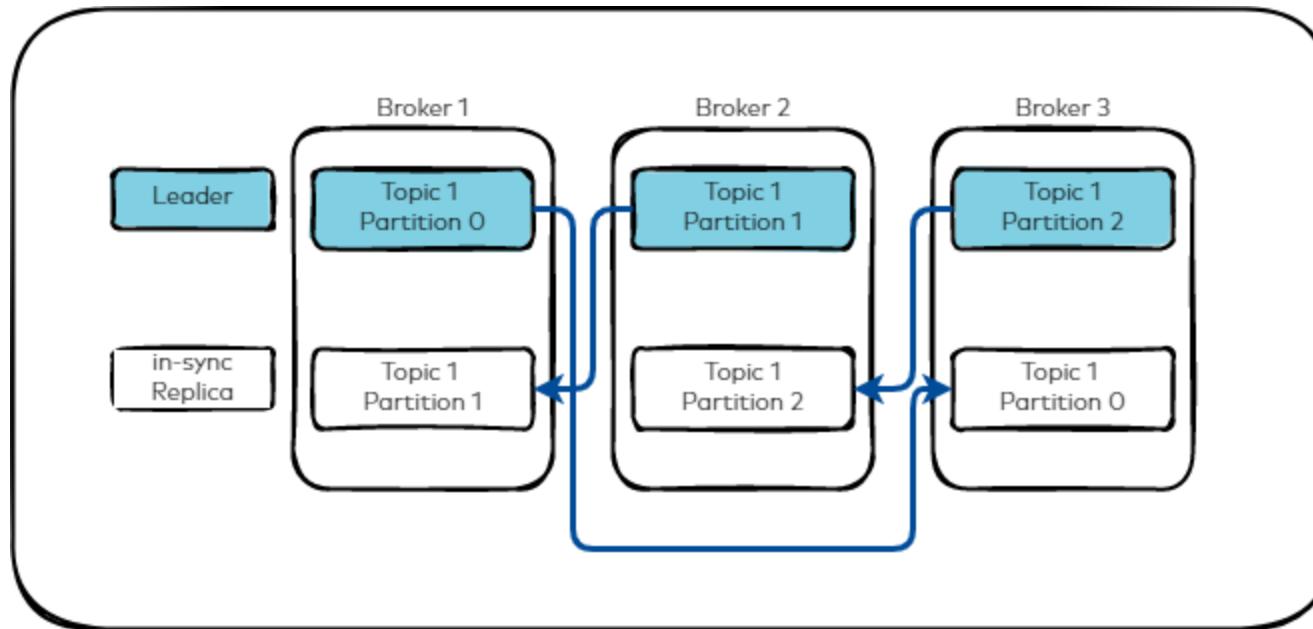
Kafka Partition



Committing



Replica & in-sync (ISR)



<https://docs.confluent.io/kafka/design/replication.html>

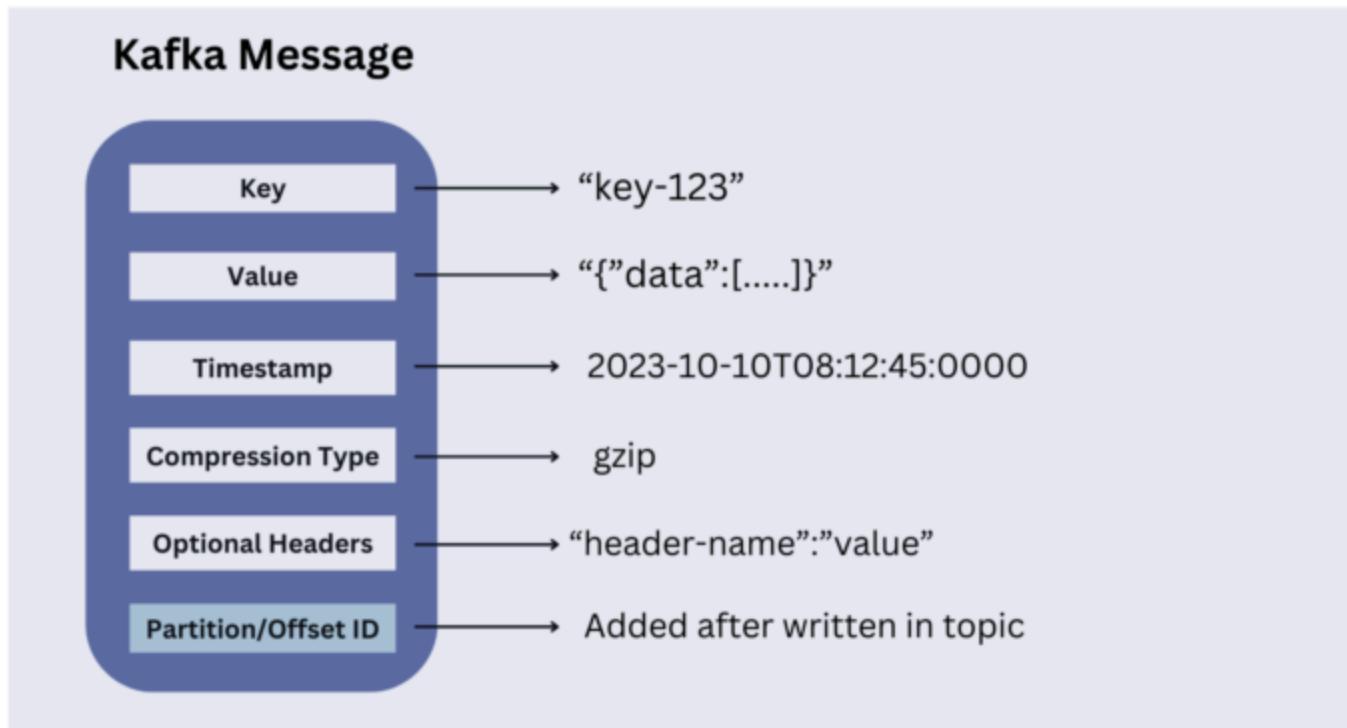
<https://www.conduktor.io/kafka/kafka-topic-replication/>

Kafka Topic Replication Factor

A replication factor of 1 means no replication. It is mostly used for development purposes and should be avoided in test and production Kafka clusters

A replication factor of 3 is a commonly used replication factor as it provides the right balance between broker loss and replication overhead.

Message



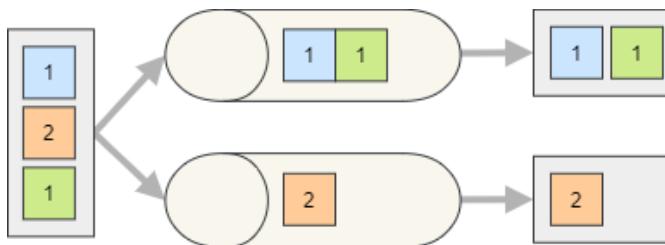
Partitioner

- Manual
- Random
- Hash
- RoundRobin

Key

Apache Kafka require a message key for different purposes, such as:

Partitioning: Kafka can guarantee ordering only inside the same partition and it is therefore important to be able to route correlated messages into the same partition. To do so you need to specify a key for each message and *Kafka will put all messages with the same key in the same partition.*



Compacting topics: A topic can be configured with `cleanup.policy=compact` to instruct Kafka to keep only the latest message related to a certain object, identified by the message key. In other words Kafka will retain only 1 message per each key value.

Message Acknowledgments

acks = 0: ไม่รอคำตอบ

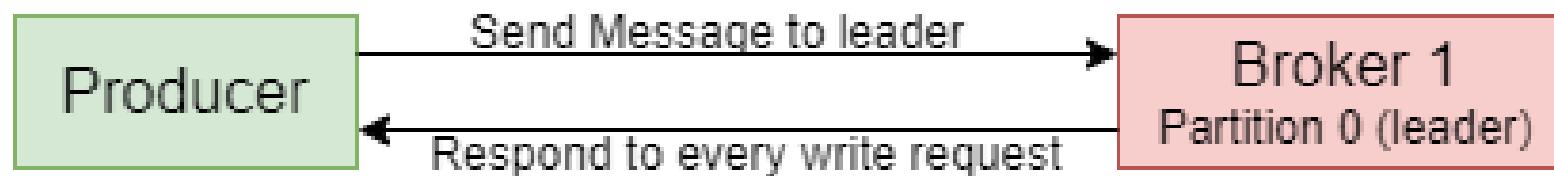
acks = 1: leader ตอบว่าได้รับ

acks = all: replica ยืนยันว่าได้รับ

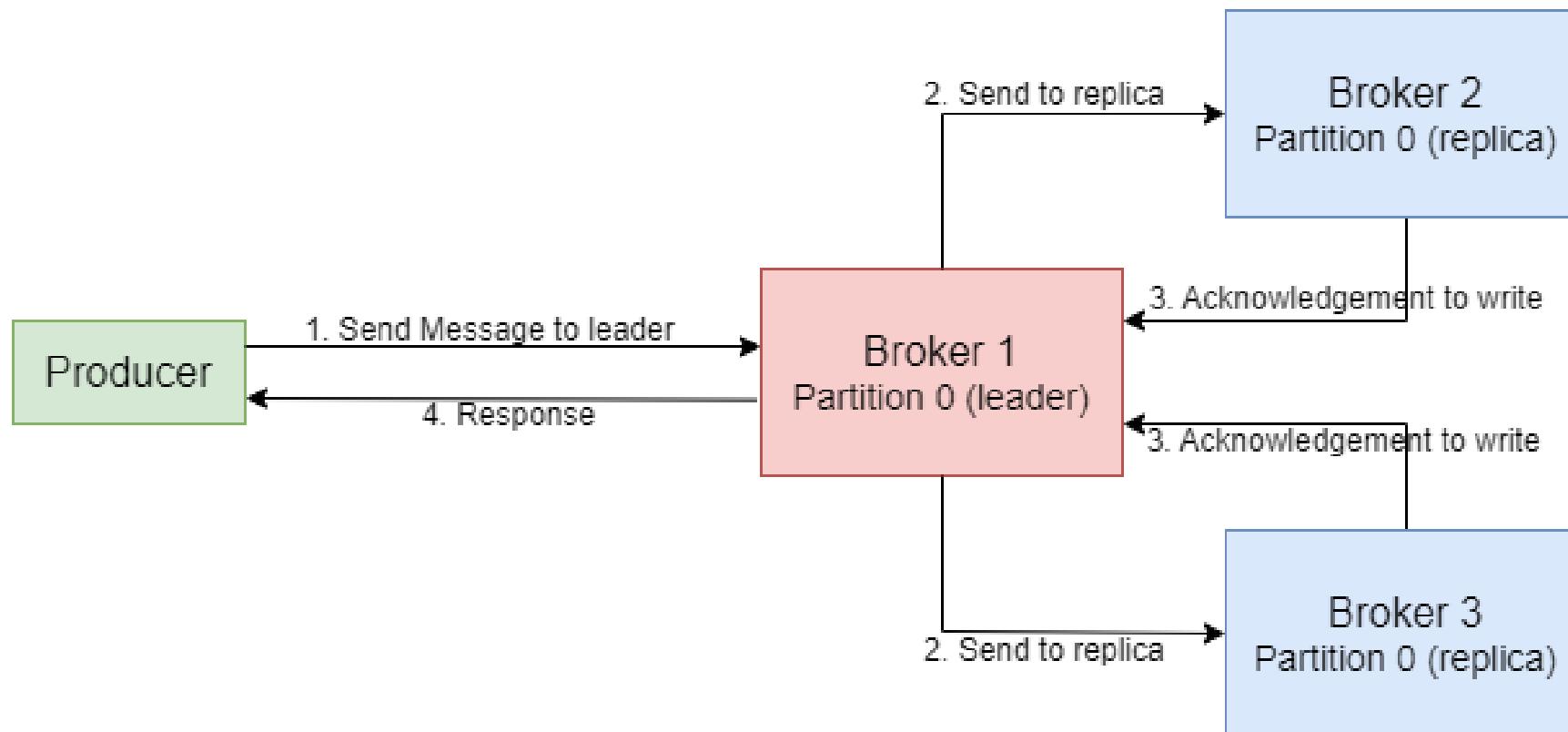
acks 0



acks 1



acks all



Message Delivery Guarantees

"Kafka messaging provides a guarantee of at-least-once delivery by default."

At most once: Messages are delivered once, and if there is a system failure, messages may be lost and are not redelivered.

At least once: This means messages are delivered one or more times. If there is a system failure, messages are never lost, but they may be delivered more than once.

Exactly once: This is the preferred behavior in that each message is delivered once and only once. Messages are never lost or read twice even if some part of the system fails.

Producers

Sync & Async

1. Sync Producer Acks = 0 (Fire and Forget)
2. Sync Producer Acks = 1 or Acks = all
3. Async Producer

Consumer rule

The rule in Kafka is a maximum of 1 consumer per partition (as each partition must only be allocated to 1 consumer), so you can only have as many consumers (in a single consumer group) as there are partitions for a topic, but you can also have less. This means that in practice consumers can process records from 1 or more partitions. The maximum throughput will be achieved with 1 consumer per partition, or as many consumers are there are partitions.

<https://www.instaclustr.com/blog/kafka-parallel-consumer-part-1/>

Thread Safety and Slow Consumers

Kafka consumers are not thread safe—you can’t share a consumer instance among multiple threads. On the other hand, Kafka producers are thread-safe and can be shared among multiple threads.

The default Kafka consumer is only single-threaded, so it can only process records sequentially, with committing done automatically upon successful record processing. This works fine if the processing time is close to 0 but becomes a problem if the processing time is longer—so-called “slow consumers”. Basically, each record has to wait in line to be processed, which increases the end-to-end latency of the records.

any problems

<https://medium.com/cloud-native-daily/how-i-dealt-with-the-kafka-issue-8c8e98fa1759>

Kafka library in Go

<https://github.com/confluentinc/confluent-kafka-go>

confluent-kafka-go is a lightweight wrapper around librdkafka, a finely tuned C client.

<https://github.com/IBM/sarama>

Pure Go

<https://github.com/segmentio/kafka-go>

Sarama

Spotify -> IBM

Compatibility and API stability

Sarama provides a "2 releases + 2 months" compatibility guarantee: we support the two latest stable releases of Kafka and Go, and we provide a two month grace period for older releases. However, older releases of Kafka are still likely to work.

Workshop

1. Sync Producer
2. Async Producer
3. Consumer Group



Pull code

<https://github.com/pallat/kafkasarama>

```
docker-compose up
docker-compose -f./cluster-docker-compose.yml up
```

Kafka-UI

<http://localhost:8080>

Sarama Tools

<https://github.com/IBM/sarama/tree/v1.43.0/tools>

[kafka-console-producer](#): a command line tool to produce a single message to your Kafka cluster.

[kafka-console-consumer](#): a command line tool to consume arbitrary partitions of a topic on your Kafka cluster.

[kafka-producer-performance](#): a command line tool to performance test producers (sync and async) on your Kafka cluster.

Sarama Mocks

<https://pkg.go.dev/github.com/IBM/sarama/mocks>

Sync Producer & Simple Consumer

1 Broker

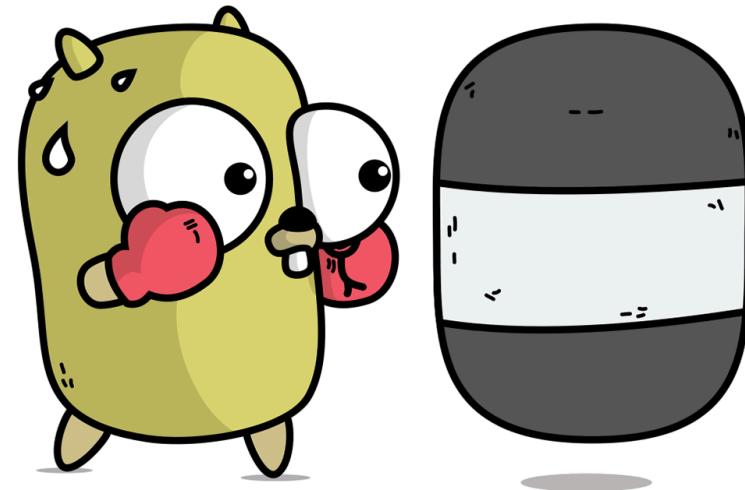
1 Partition



Partition

3 Brokers

1 Partition



Replica

3 Brokers

3 Partitions

Create topic via Kafka-UI

- 1 Replica
- 2 Replicas
- 3 Replicas



A Tour Guide of Go

Consumer Group

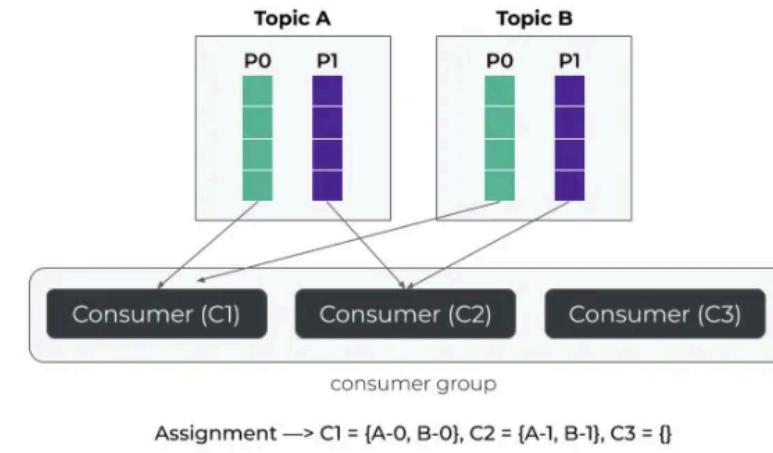
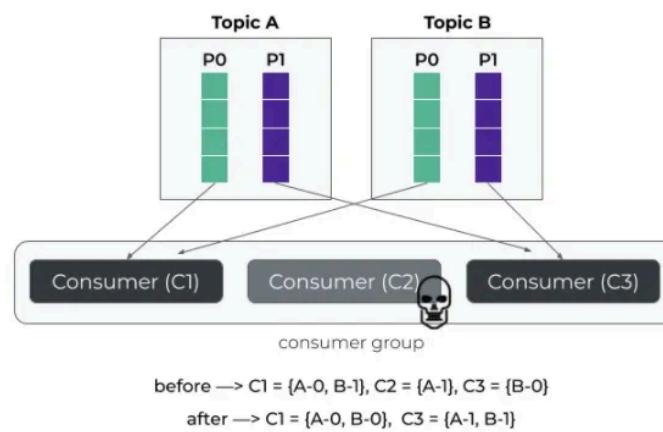
Consumer Group Rebalance GroupStrategies

- RoundRobinAssignor Strategy
- StickyAssignor Strategy (ຂໍ້ມູນອຍສຸດ)

$C1 = \{A0, B1\}$, $C2 = \{A1\}$, $C3 = \{B0\} \rightarrow C1 = \{A0, B1\}$, $C3 = \{A1, B0\}$

- RangeAssingor Strategy (default)

The “Range Assignment Strategy” aims to sort topic partitions in numeric order and also sort the consumers in lexicographic order.



Rebalancing

<https://medium.com/trendyol-tech/rebalance-and-partition-assignment-strategies-for-kafka-consumers-f50573e49609>

<https://www.linkedin.com/pulse/kafka-consumer-group-rebalance-1-2-robgolder/>

<https://www.linkedin.com/pulse/kafka-consumer-group-rebalance-2-robgolder/>

Scenario

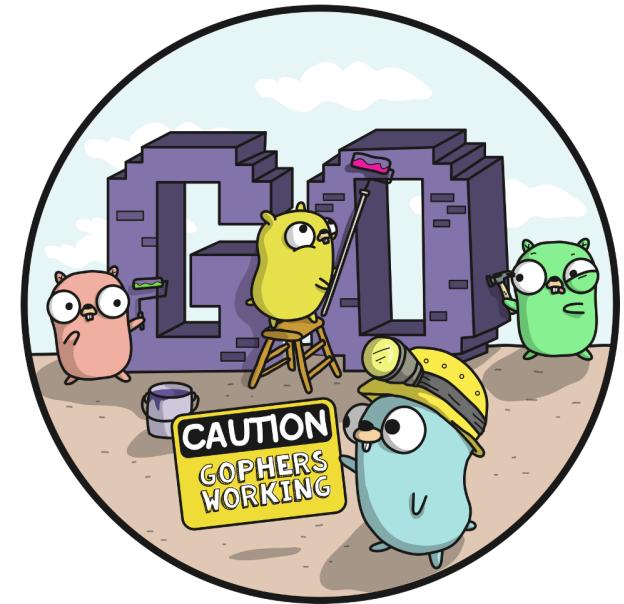
start no 1 consumer group

start no 2 consumer group

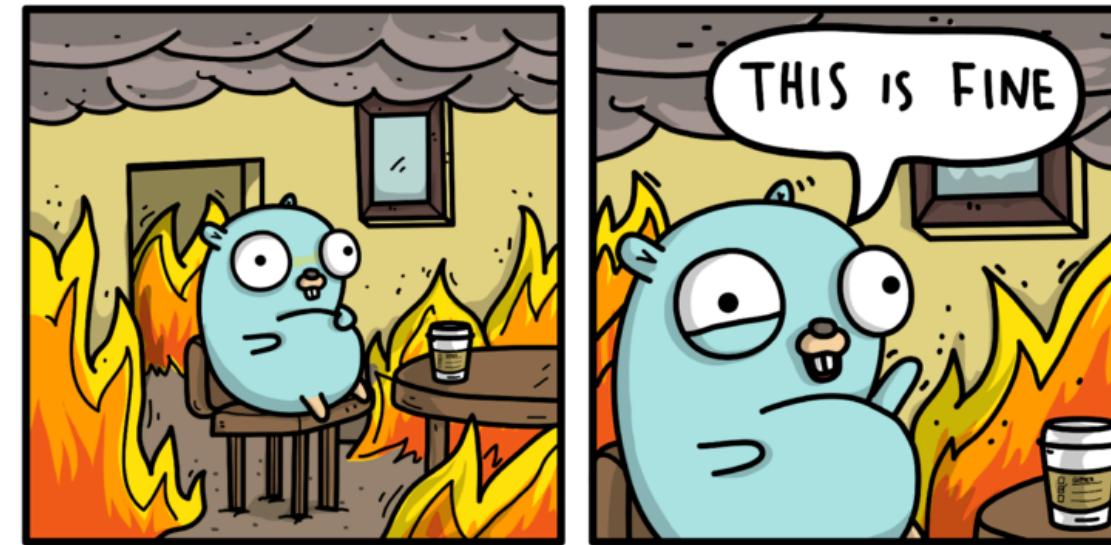
down no 1

start no 1

start no3



Graceful

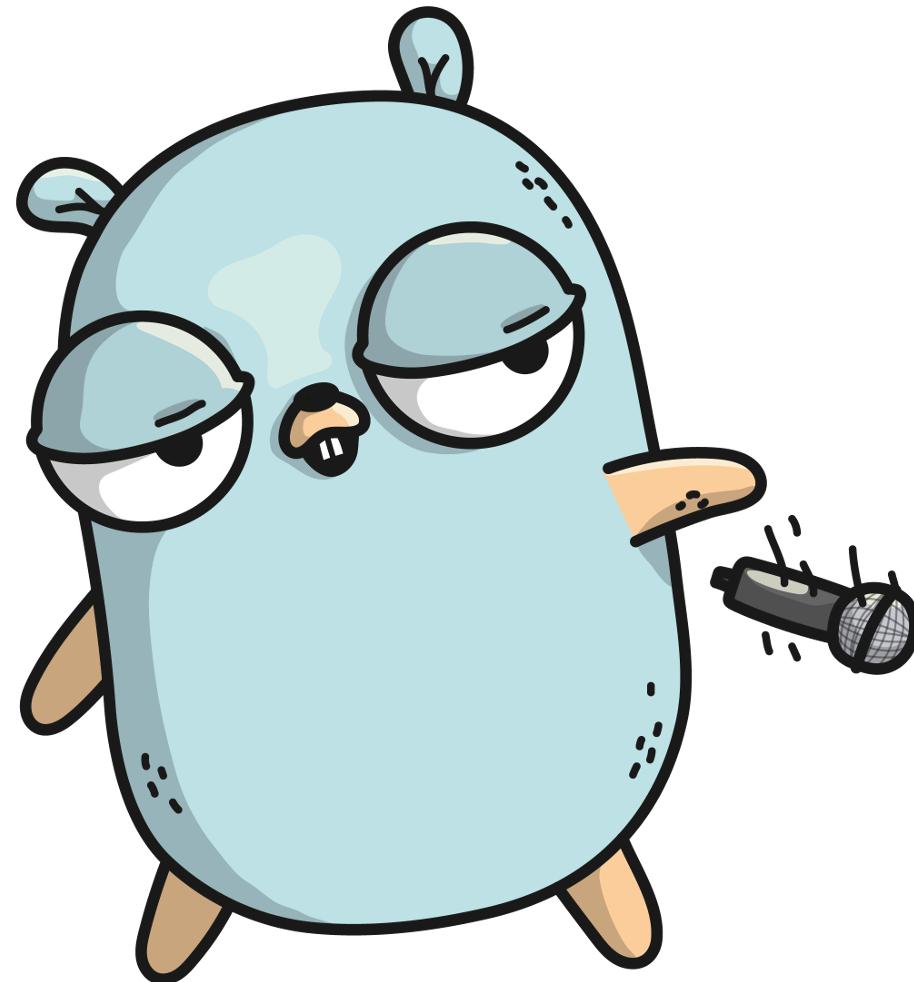




Test & Mock



A Tour Guide of Go





A Tour Guide of Go