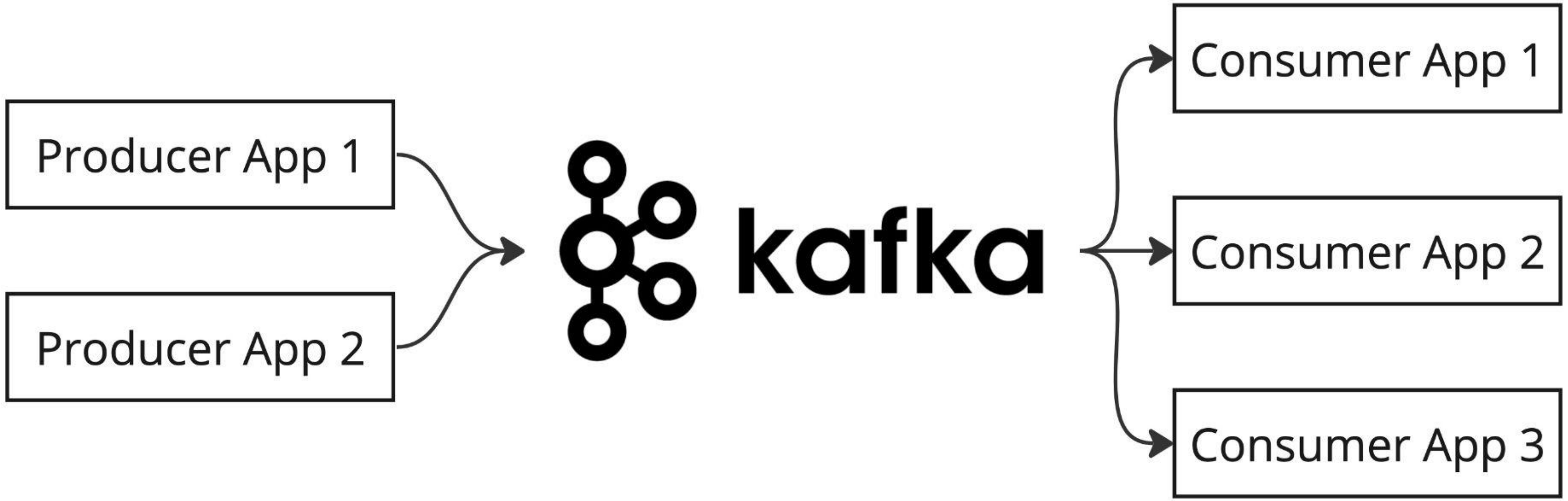
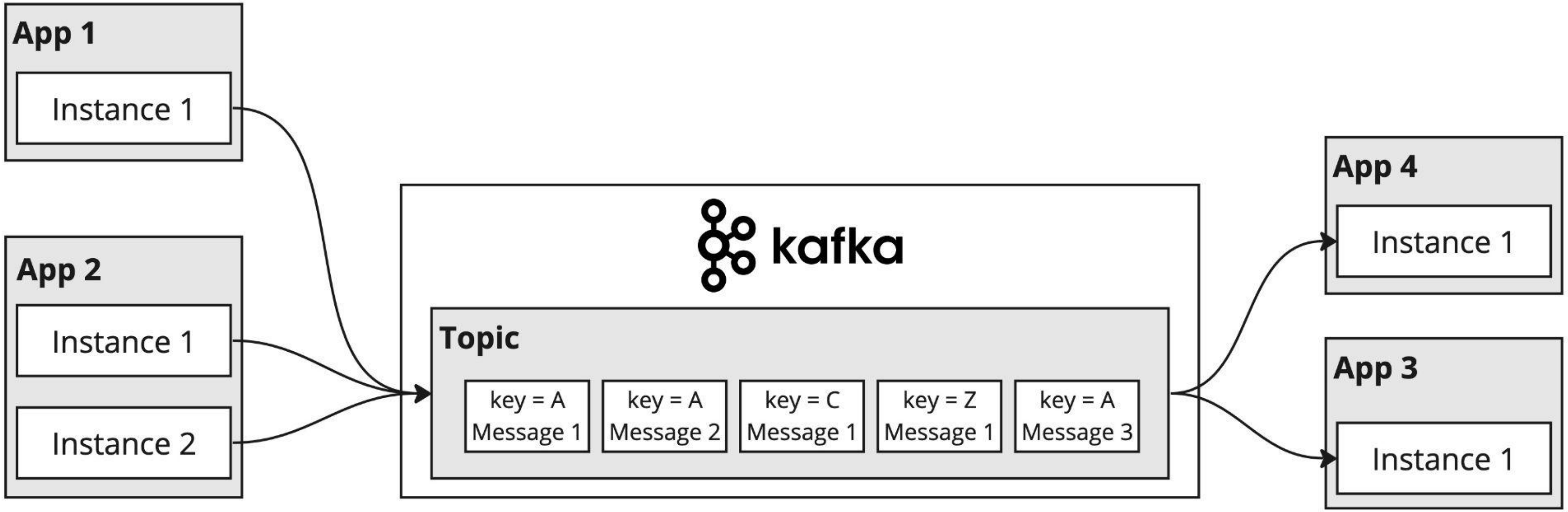
Uma introdução ao Apache Kafka

Júlio César Batista
2024-02-21

An overview

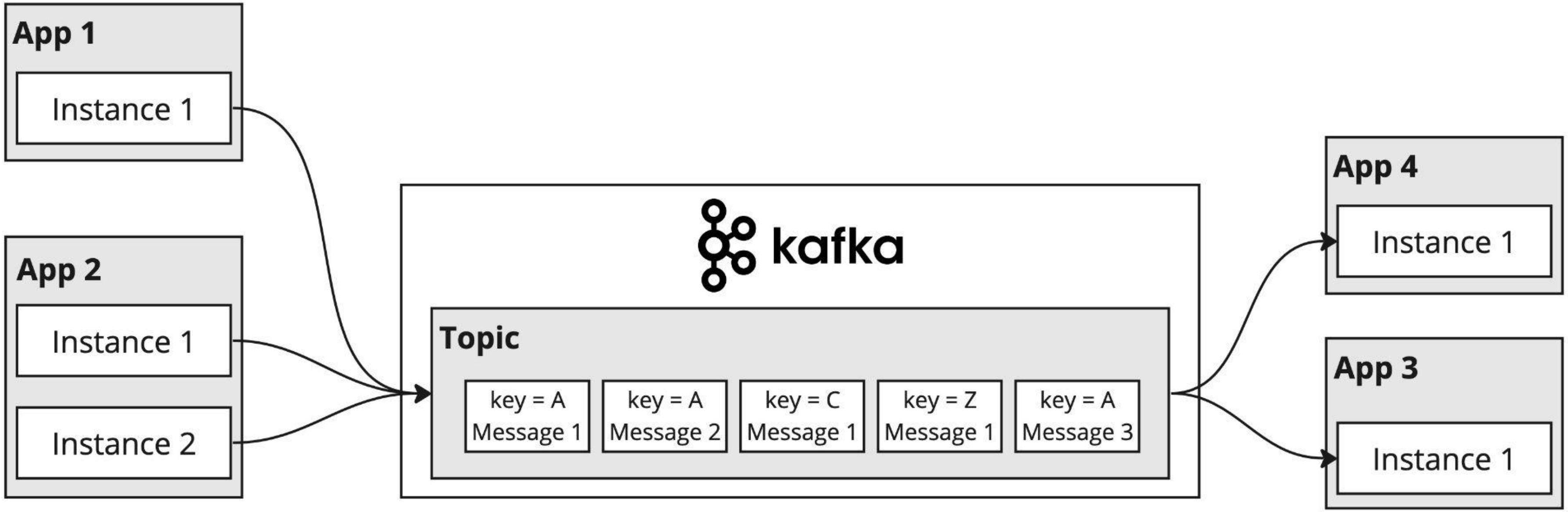

Producer App 1
Producer App 2
kafka
Consumer App 1
Consumer App 2
Consumer App 3

A Topic is a Log



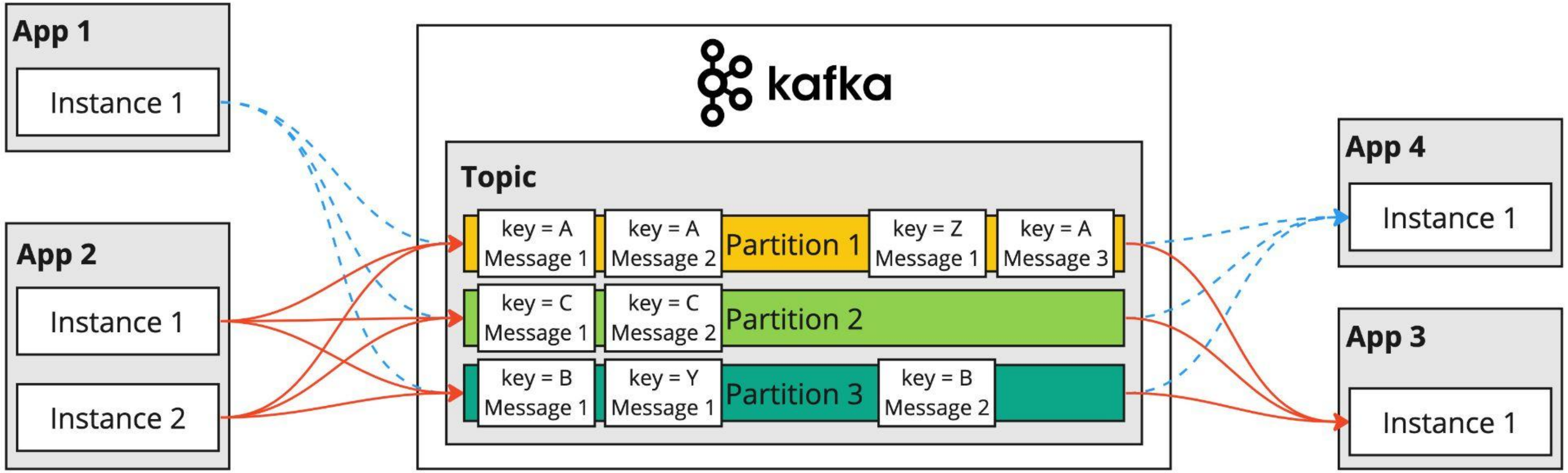
- Record of immutable events
- **Messages are not destroyed after delivery**
  - There is no "queue depth", only consumer lag for monitoring
- Messages and keys are just bytes

“Eventy” messages



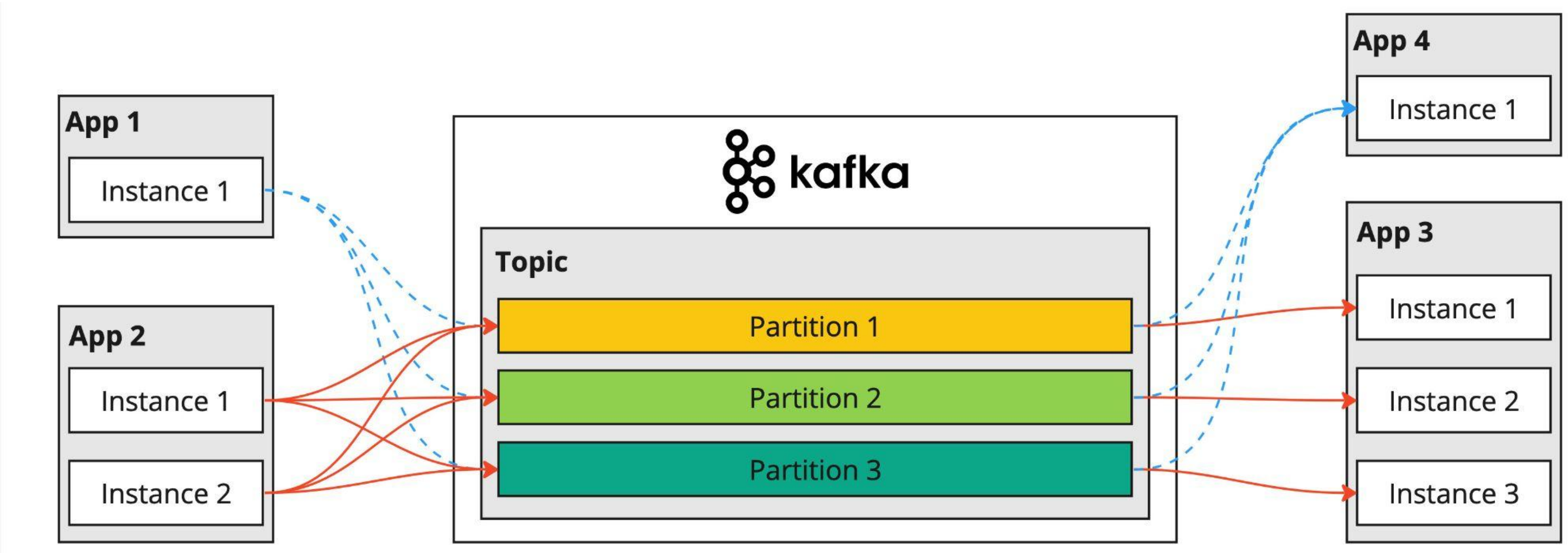
- `user_signup {email: “user@email.com”}` instead of `send_welcome_email {email: “user@email.com”}`
- Assume dumb consumers, add context to messages
  - `user_signup {email: “user@email.com”, name: “Jane Doe”, id: 123}` instead of `user_signup {id: 123}`

Partitions & Routing key



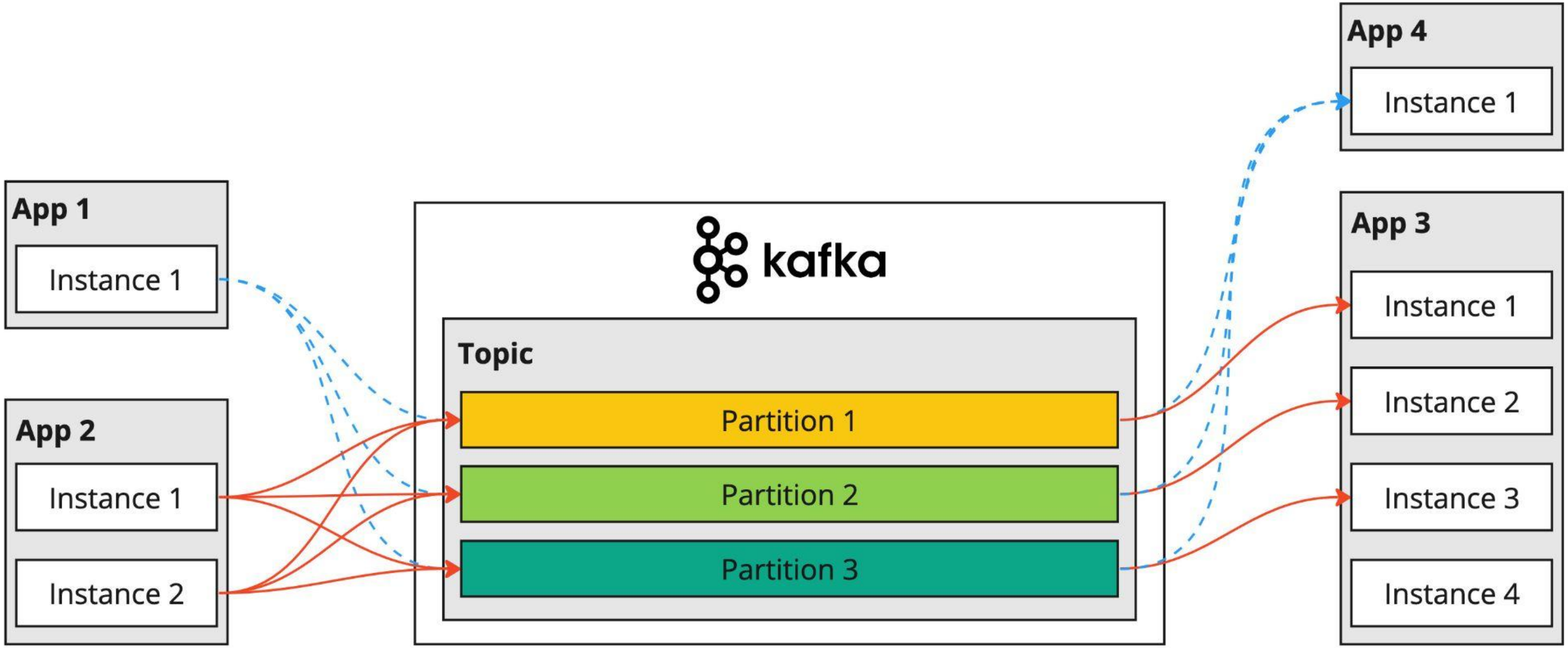
- Topics can be partitioned
  - In reality, a partition is a log
- No global order, just partition order
- **Messages with the same key are sent to the same partition**
  - Choose keys that won't overload a single partition
  - If no key is given, it generates a random one (round robin)

Partitions & Scalability



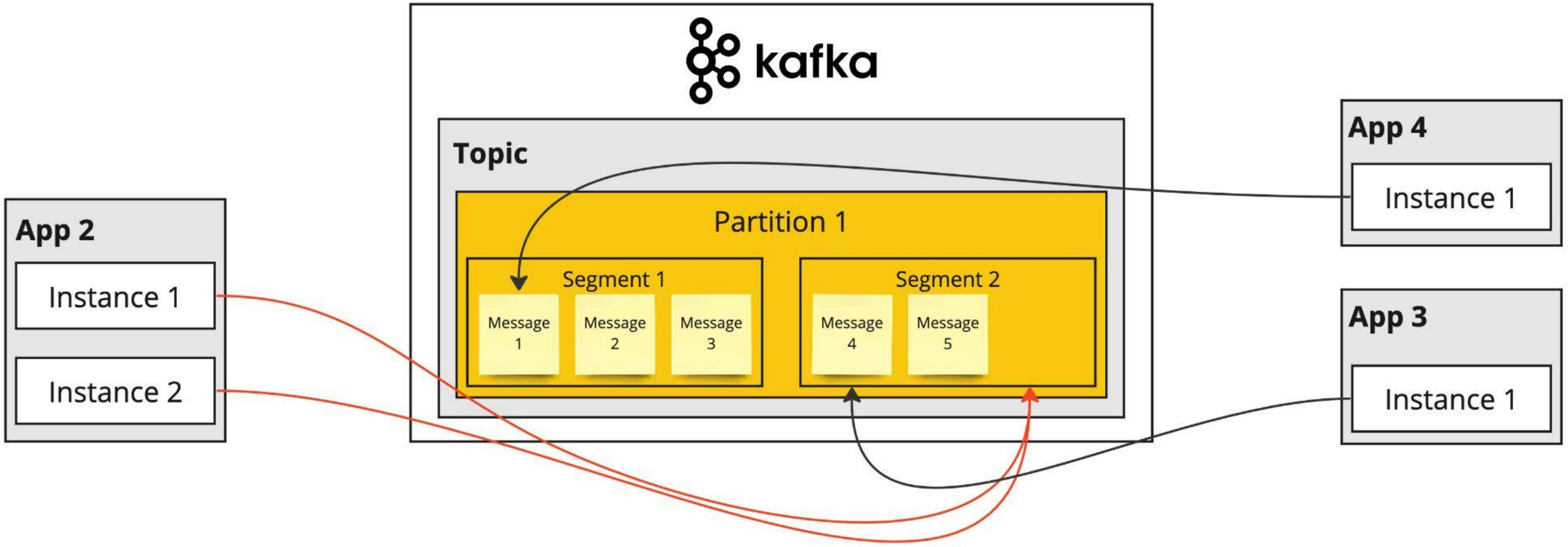
- Partitions are distributed through consumers when they join/leave (rebalance)
- Partitions can be consumed by multiple consumer groups at the same time
- **Only one consumer, per consumer group, can consume from a partition**
- Producers can write to any partition at any time

Consumer limit per consumer group



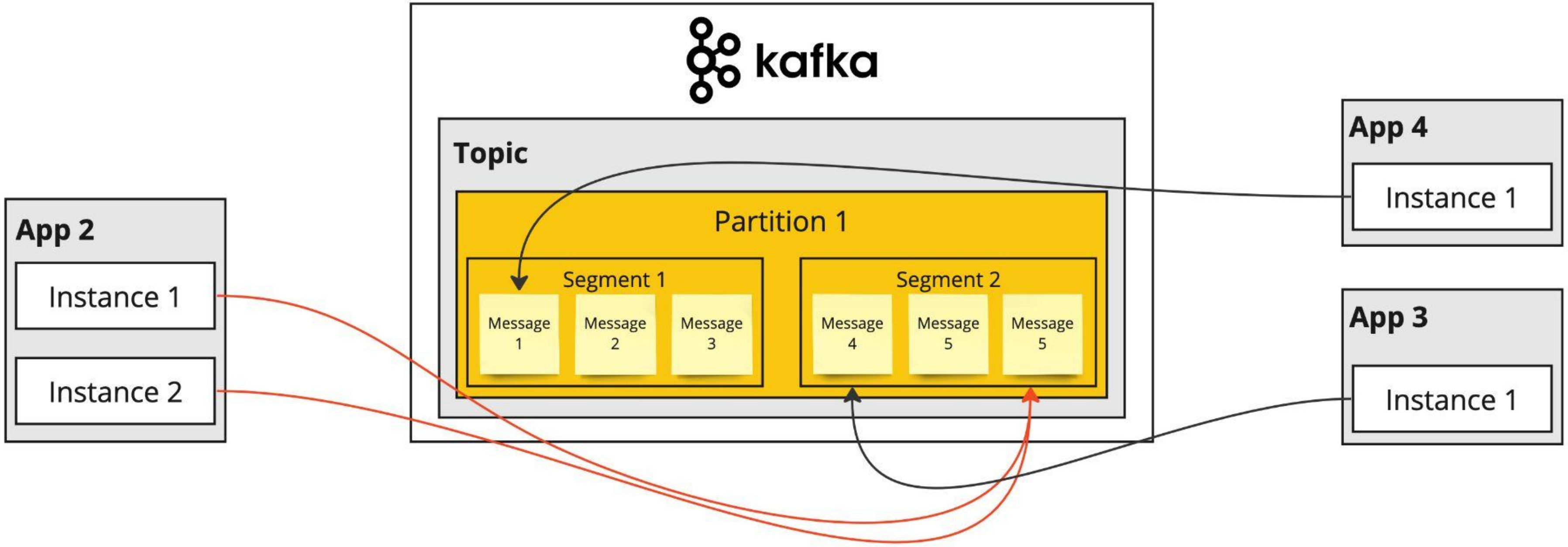
- If there are more consumers than partitions, one of them will be idle
  - Or worse, can keep triggering rebalancing

“Inside a partition”



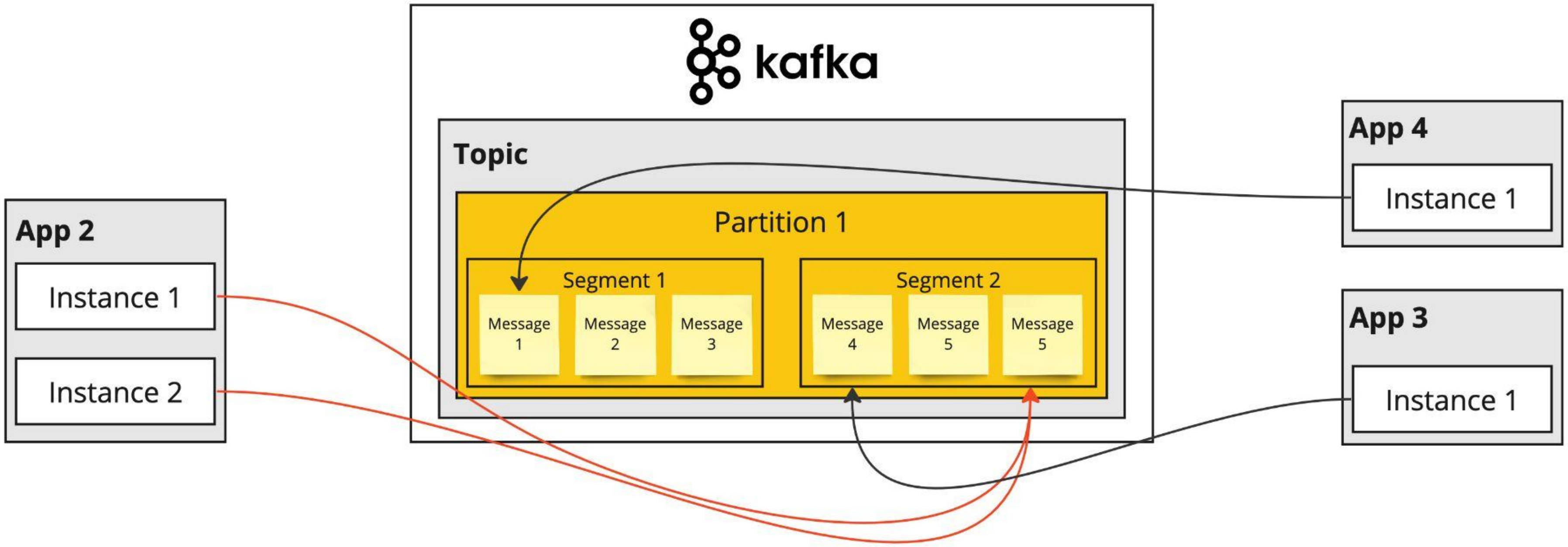
- Producers always write to the end of the log
- **Each consumer group keeps a separate message offset for a partition**
  - When a new consumer group joins, it can consume from the oldest message or start with new ones
  - This allows messages to be replayed

Message delivery semantics



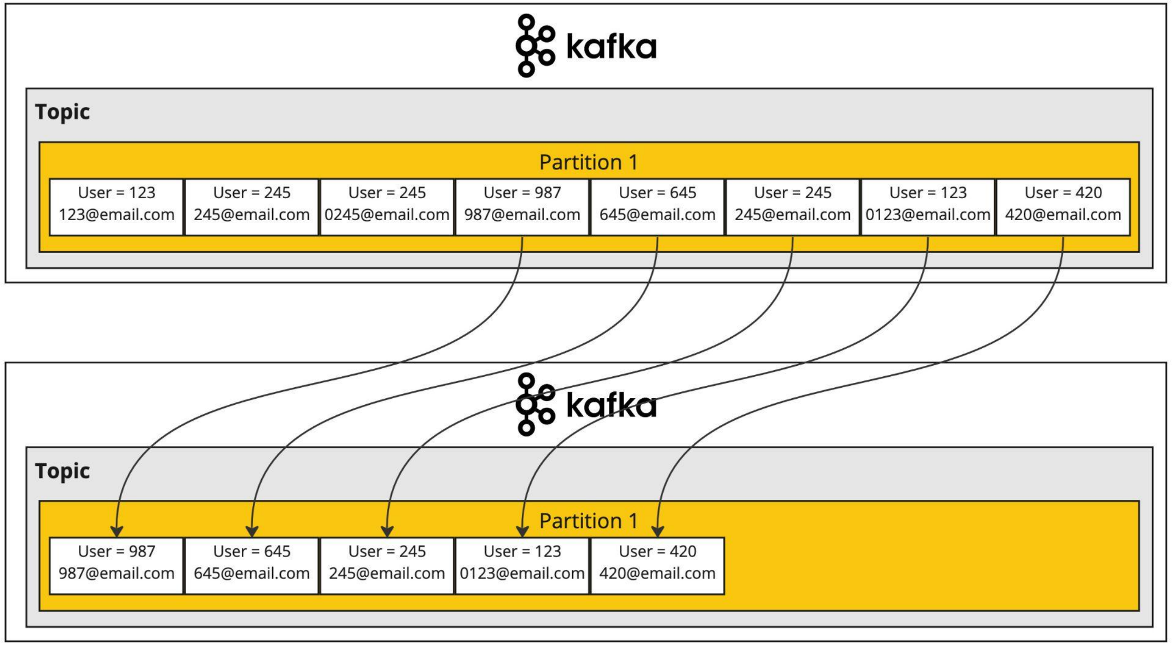
- Messages are delivered at least once (default)
  - Can be configured to be exactly once
- Consumers should be idempotent
  - Especially because of replayability

Segments



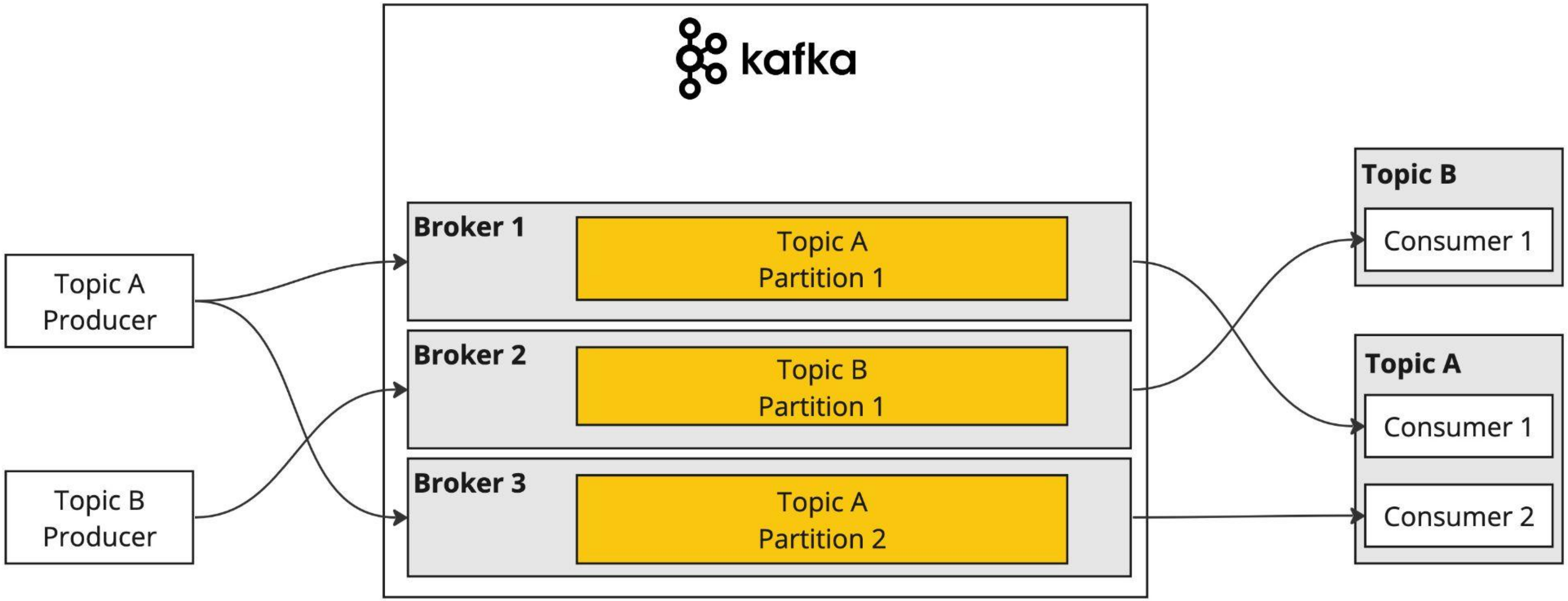
- Messages are retained by 1 week (default, configurable)
- Segments are the disk files where messages are stored
- **Only segments are deleted after the retention period**
  - There can be messages older than the retention if there were no more writes to a partition

Compaction



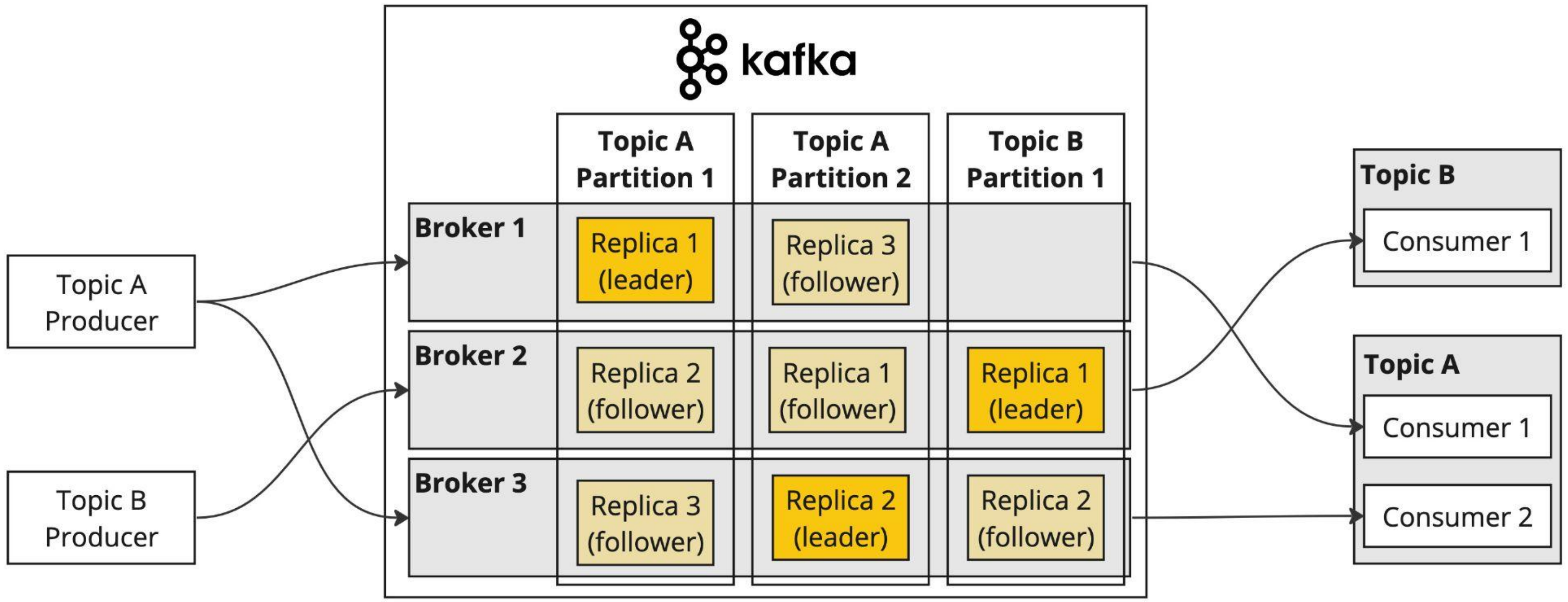
- Keep only the last record for a given key
- Enabled by default
- Doesn't make much sense for messages without keys

Brokers



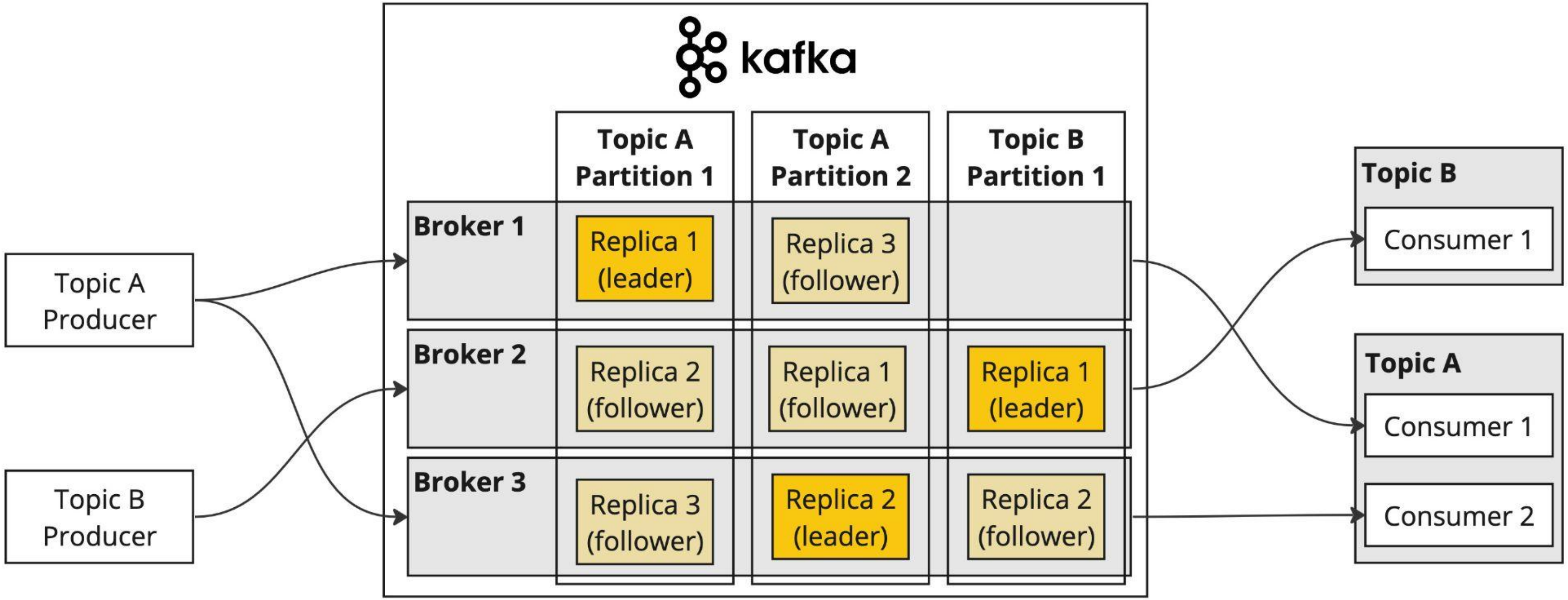
- A Cluster is composed by a set of brokers
- **Brokers store the partitions**
  - A topic with multiple partitions is spread through multiple brokers
- Producers and Consumers connect with brokers to write/read to/from a partition

Replication



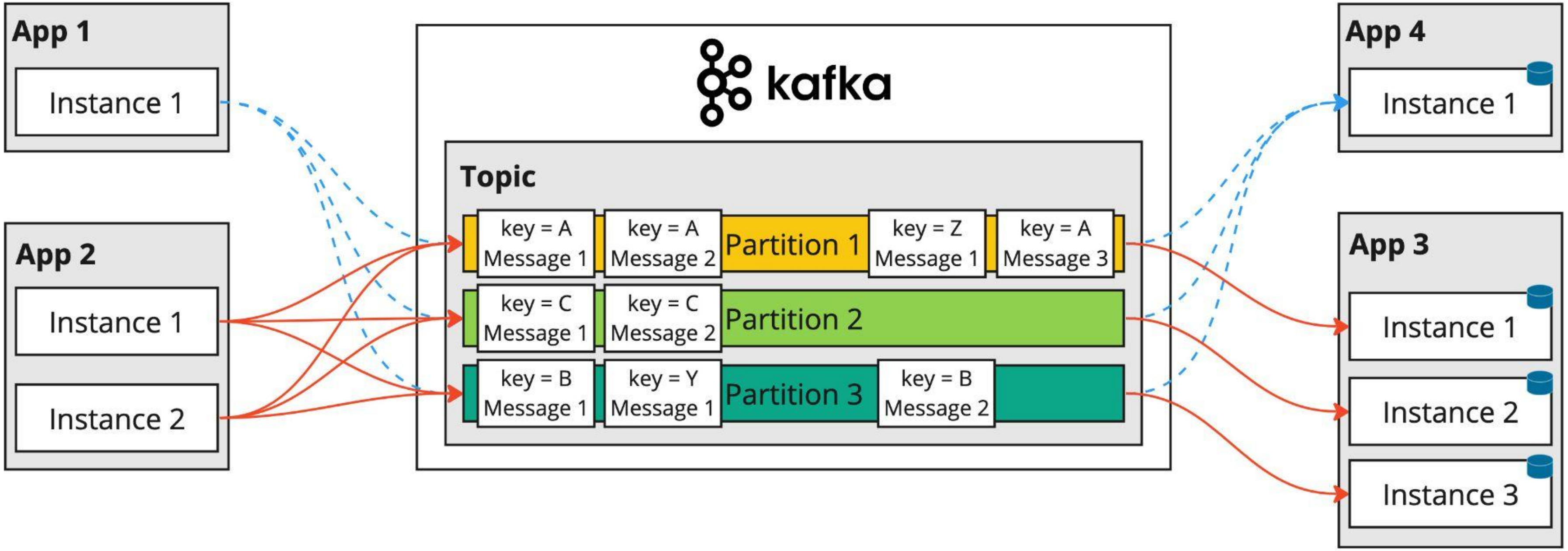
- A partition has 3 replicas (default, configurable) stored in different brokers
- **Producers & Consumers connect with the replica leader**
  - Replica followers scrape the leader to keep up to date with new messages
- When the leader goes down, one of the replicas assumes as the leader

Acknowledgement and consumption readiness



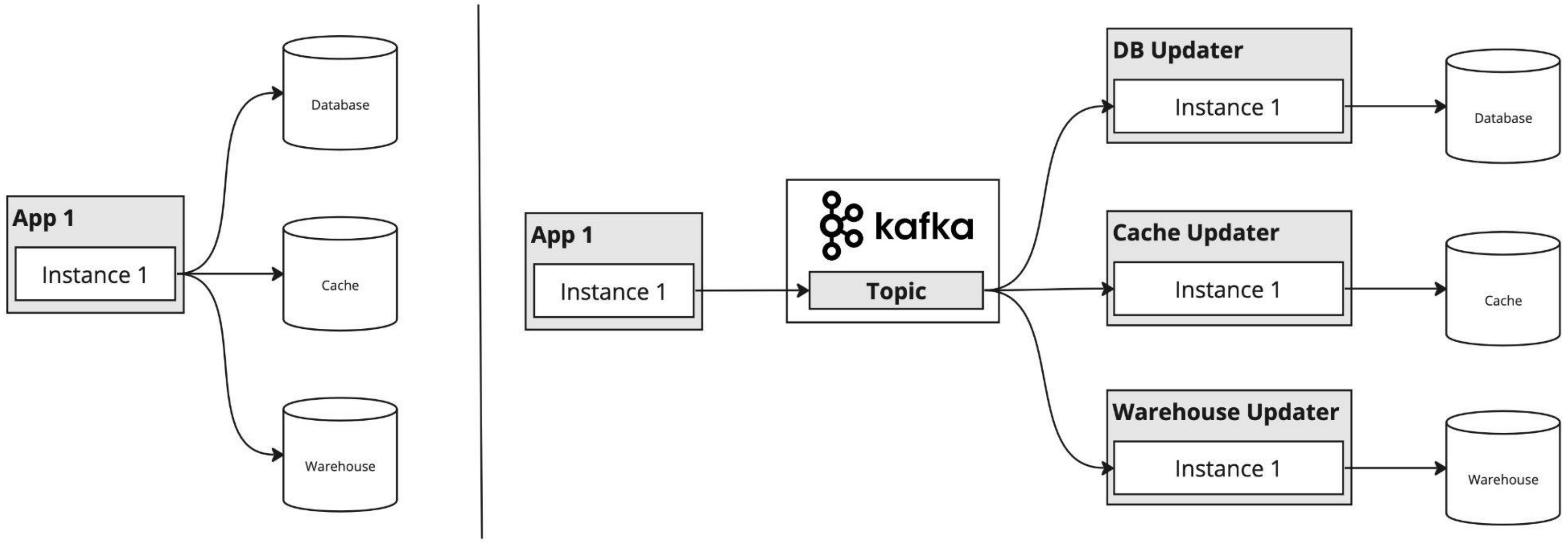
- The producer receives an ACK when the leader received the message
    - Alternatively, when it was written to the disk
    - Alternatively, when it was replicated to active followers
- **Consumers only receive messages that have been replicated**

Key ordering for the win



- No need to handle concurrency since messages with the same key are ordered and delivered/processed in order
- Consumers can keep an in memory cache since messages with the same key will be delivered to the same consumer

No more distributed transactions



- Instead of controlling distributed updates
- Write to a log and let the storage mechanism catch up eventually

Further reading/watching

- “The magical rebalance protocol of Apache Kafka” by Gwen Shapira - https://www.youtube.com/watch?v=MmLezWRI3Ys
- “Is Kafka a database?” by Martin Kleppmann - https://www.youtube.com/watch?v=v2RJQELoM6Y
- “Kafka: A modern distributed system” by Tim Belgrund - https://www.youtube.com/watch?v=Ea3aoACnbEk
- “How Kafka works” by Tim Belgrund - https://www.youtube.com/watch?v=jY02MB-sz8I
- “The Log: What every software engineer should know about real-time data's unifying abstraction” by Jay Kreps - https://engineering.linkedin.com/distributed-systems/log-what-every-software-engineer-should-know-about-real-time-datas-unifying
- Confluent resources - https://www.confluent.io/resources/?language=english