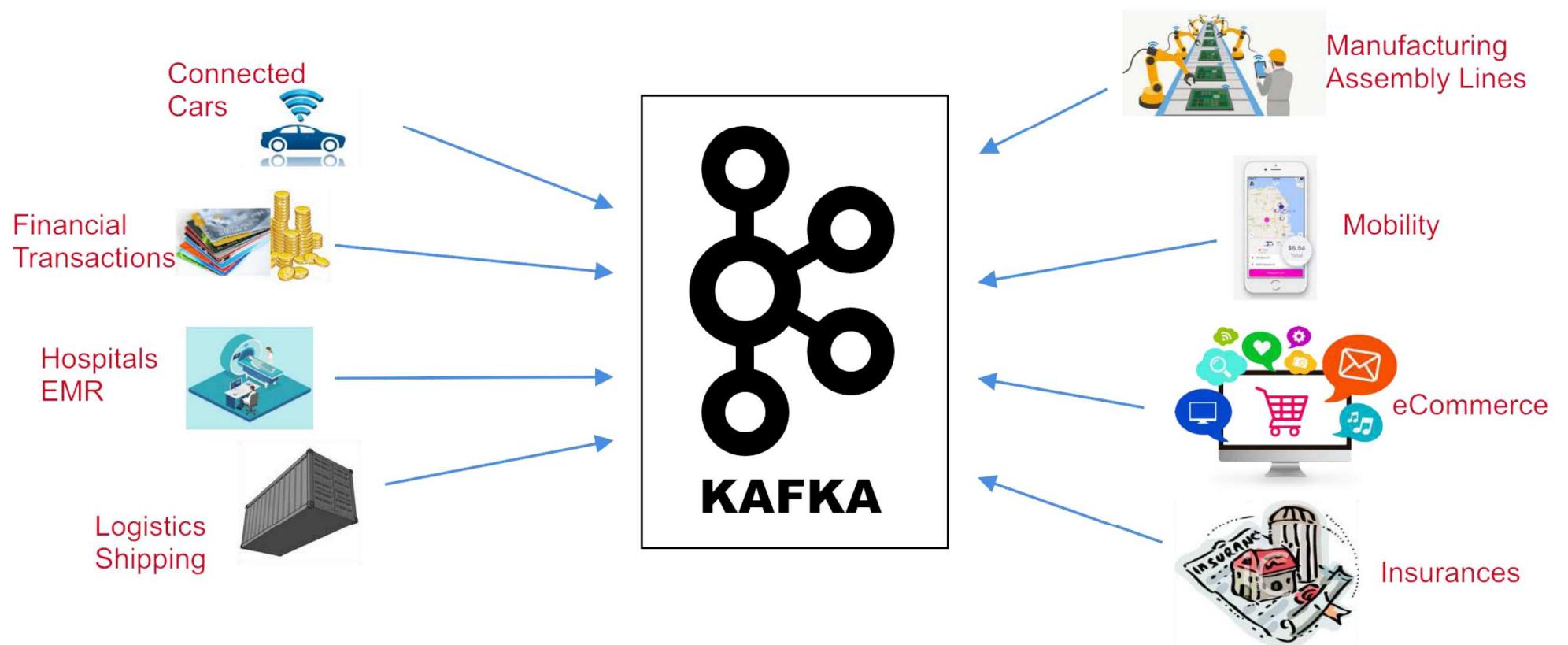


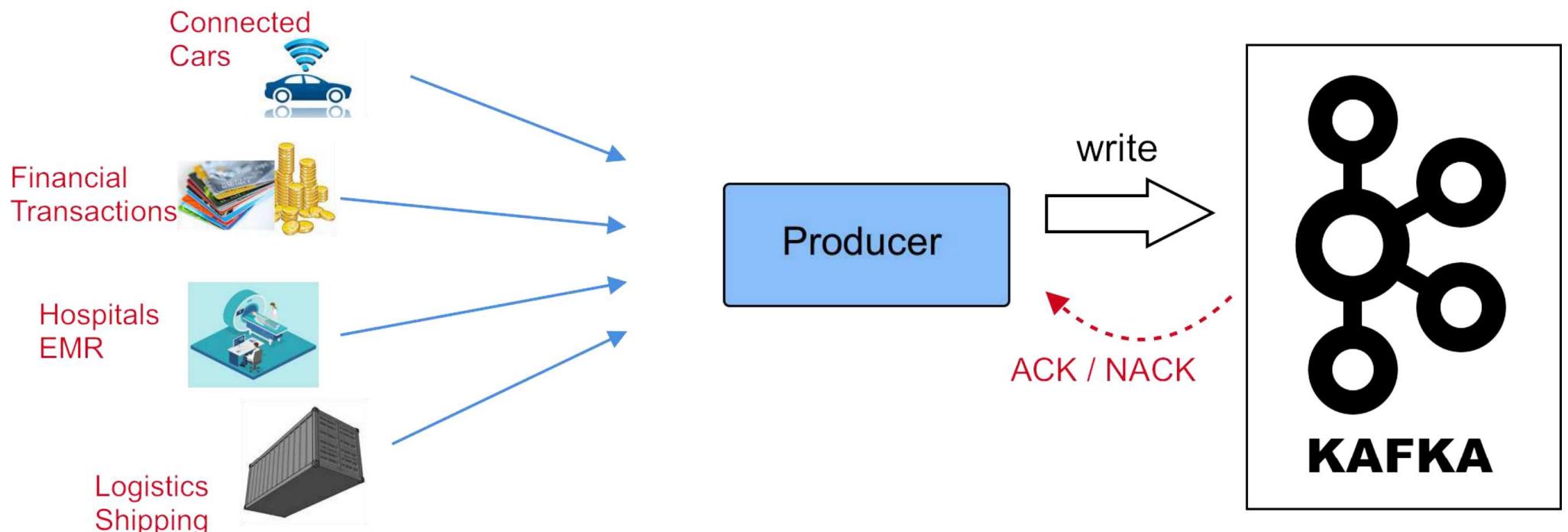


# Fundamentals

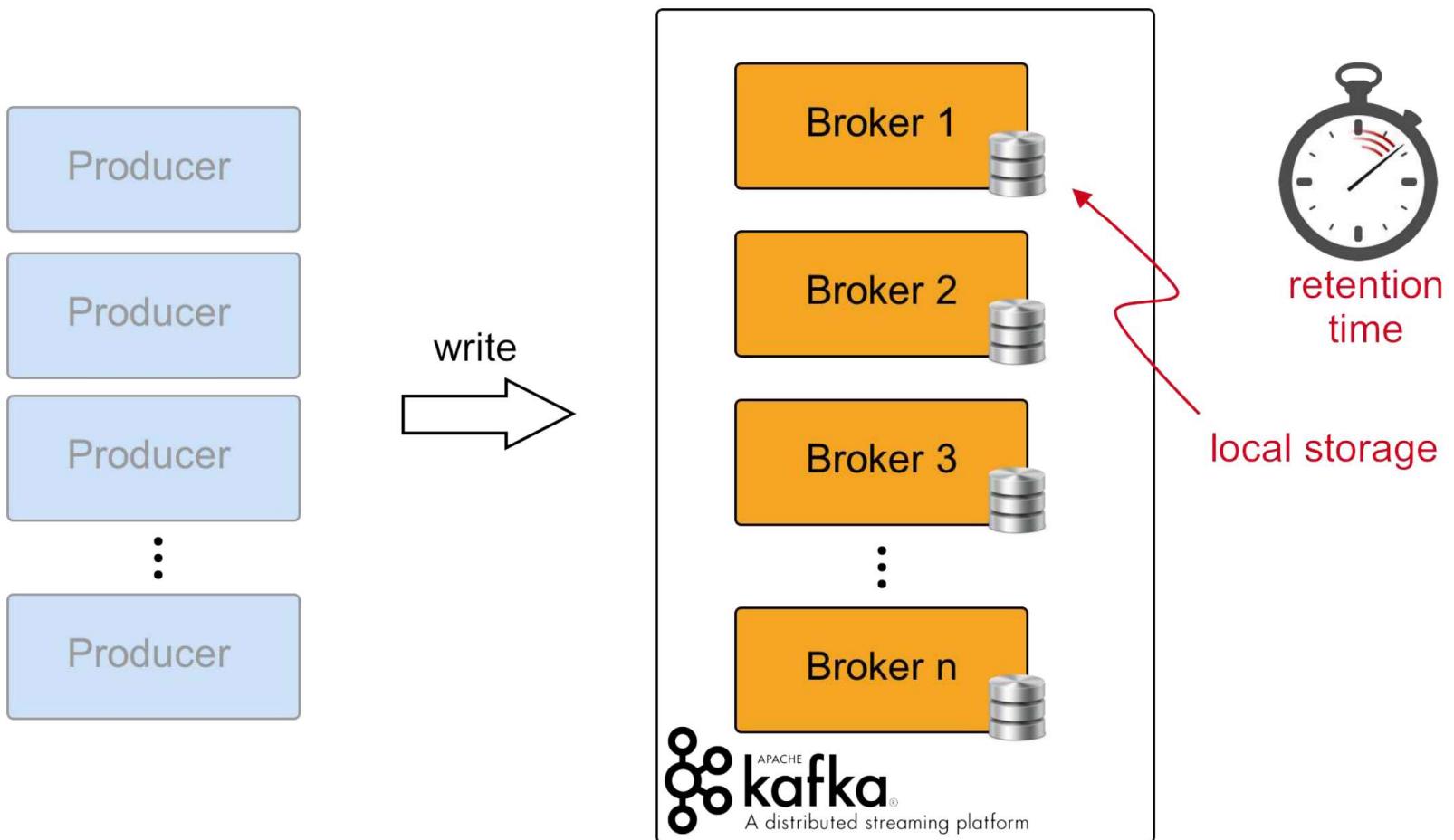
# The World Produces Data



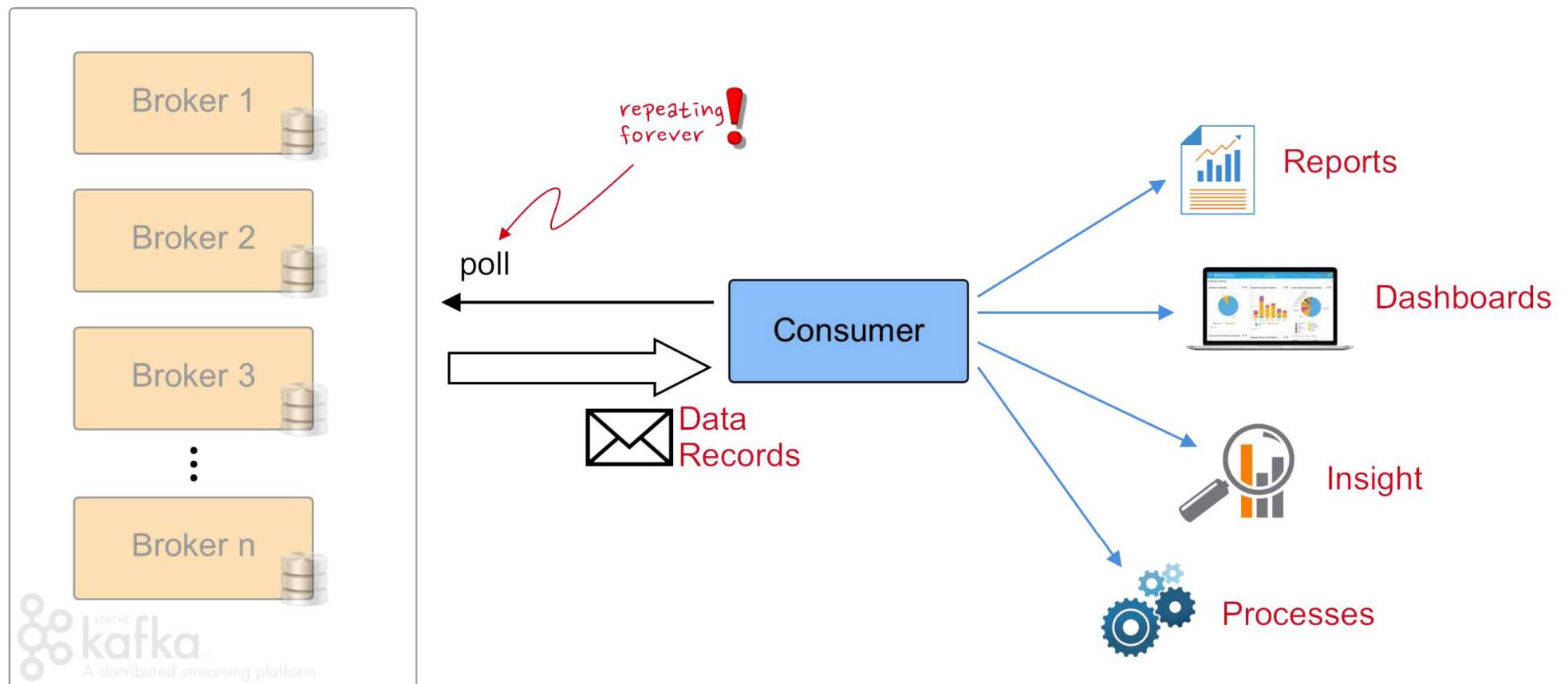
# Producers



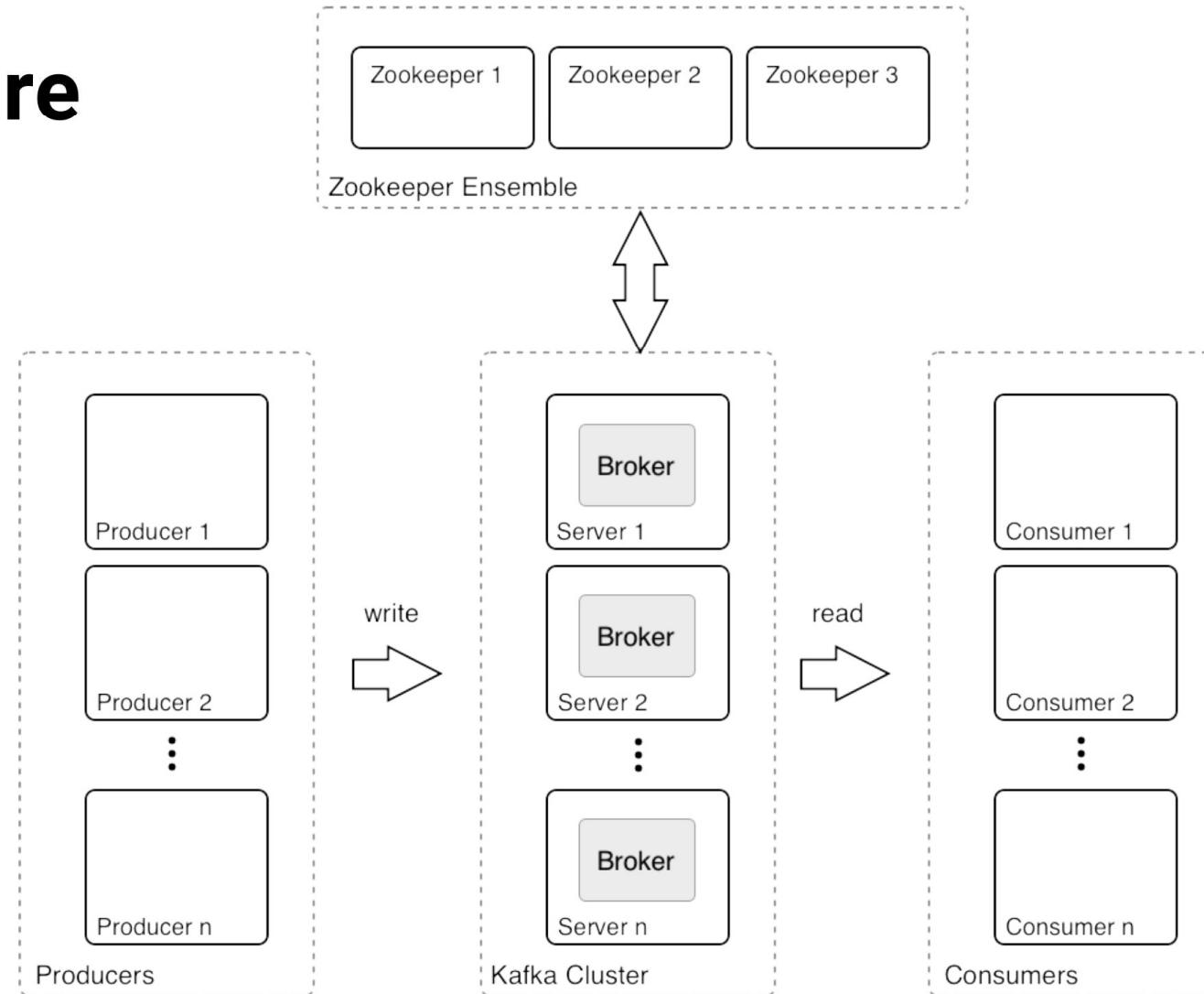
# Kafka Brokers



# Consumers

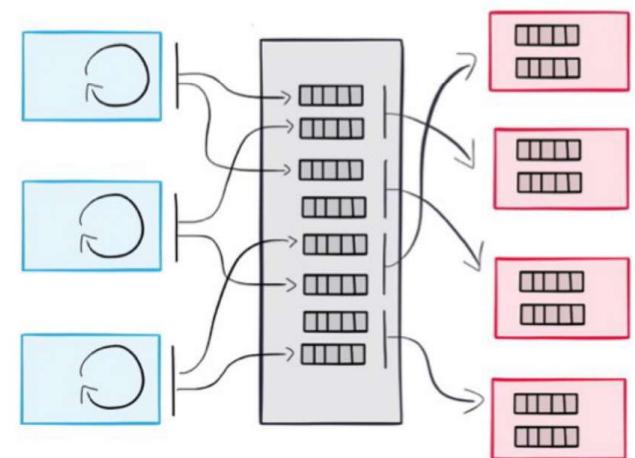


# Architecture

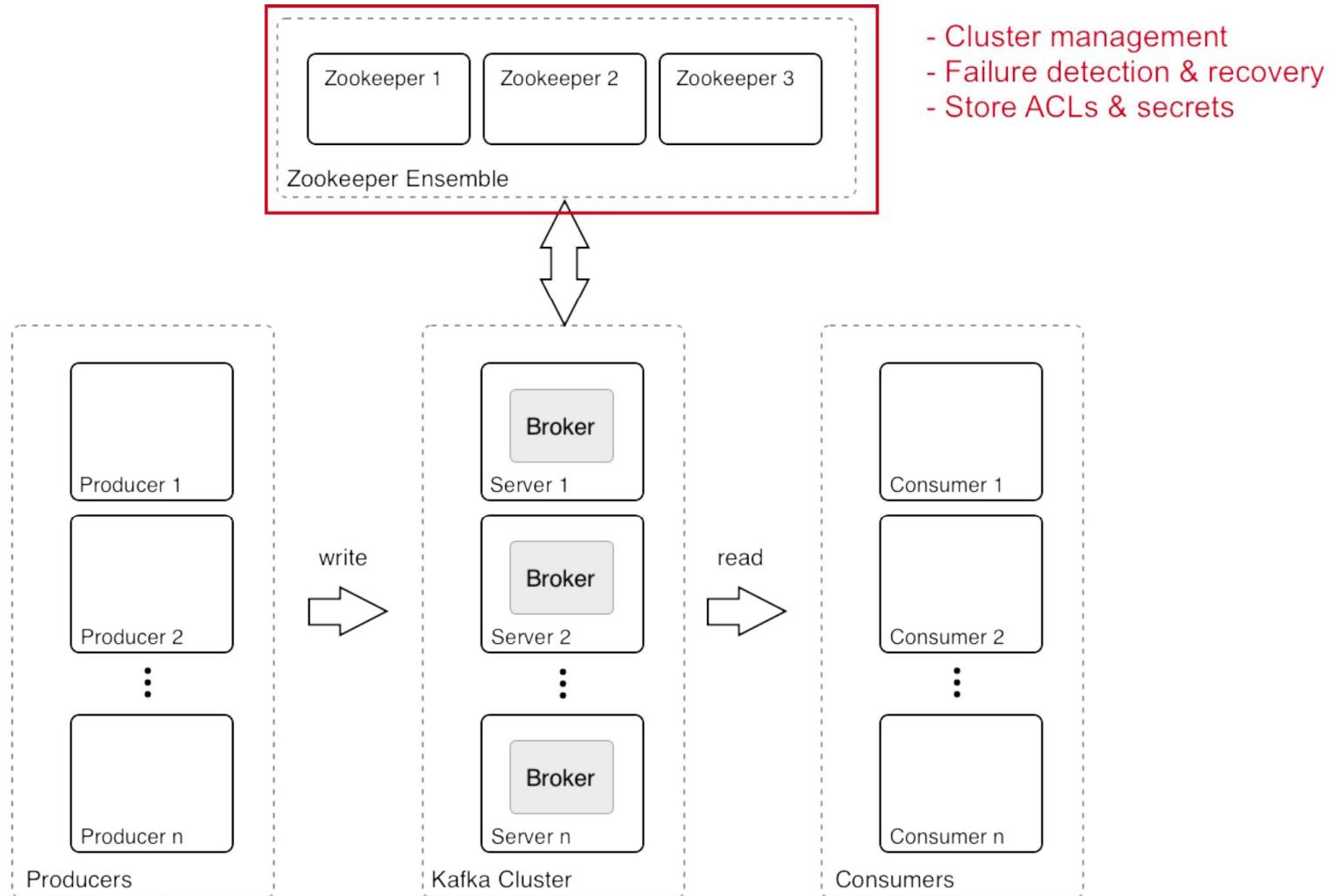


# Decoupling Producers and Consumers

- Producers and Consumers are decoupled
- Slow Consumers do not affect Producers
- Add Consumers without affecting Producers
- Failure of Consumer does not affect System



# How Kafka Uses ZooKeeper



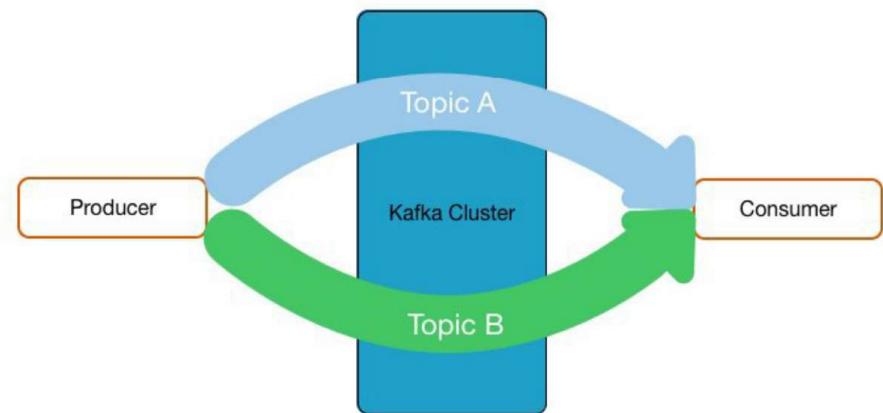
# ZooKeeper Basics

- Open Source Apache Project
- Distributed Key Value Store
- Maintains configuration information
- Stores ACLs and Secrets
- Enables highly reliable distributed coordination
- Provides distributed synchronization
- Three or five servers form an ensemble

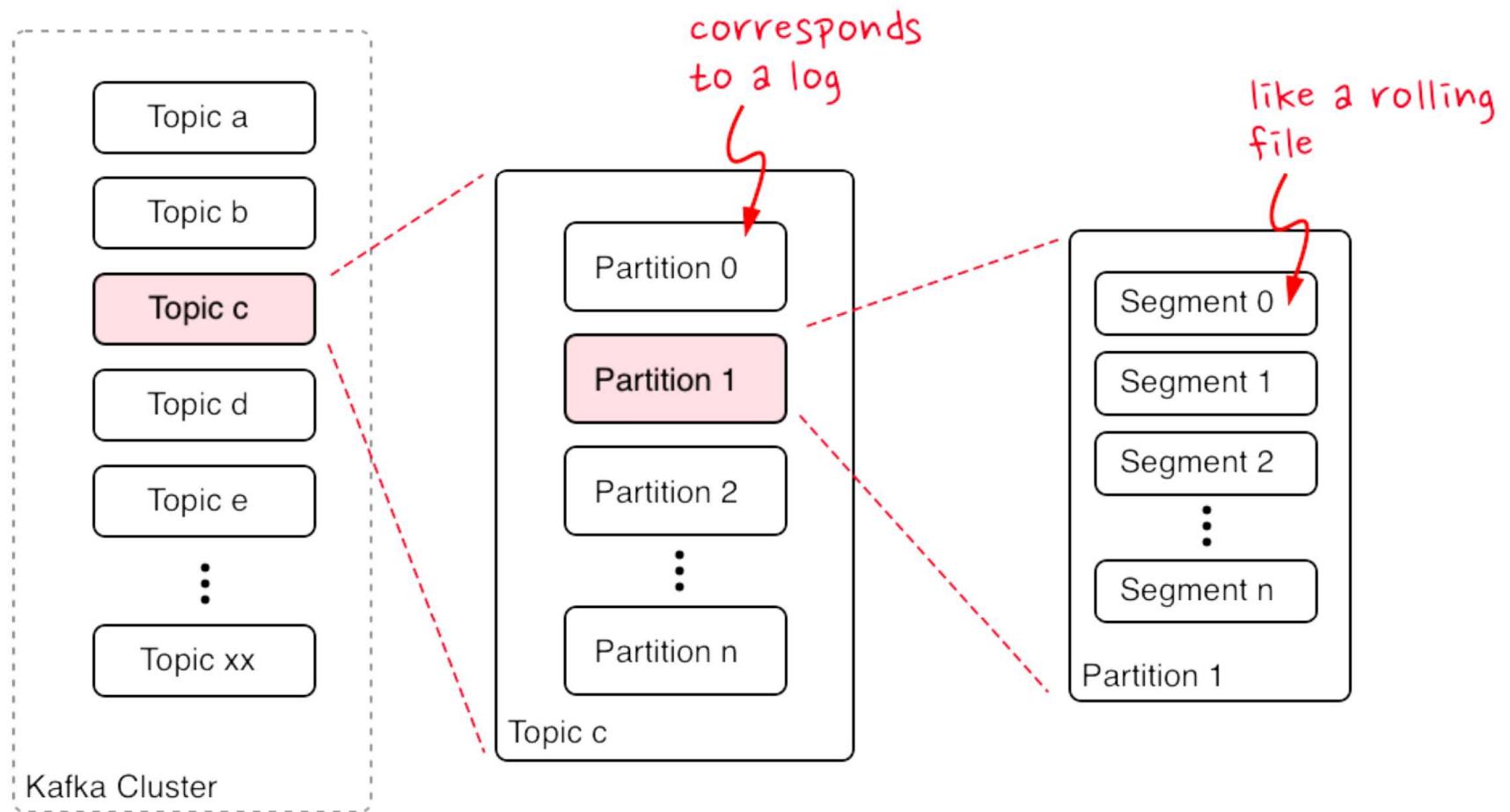


# Topics

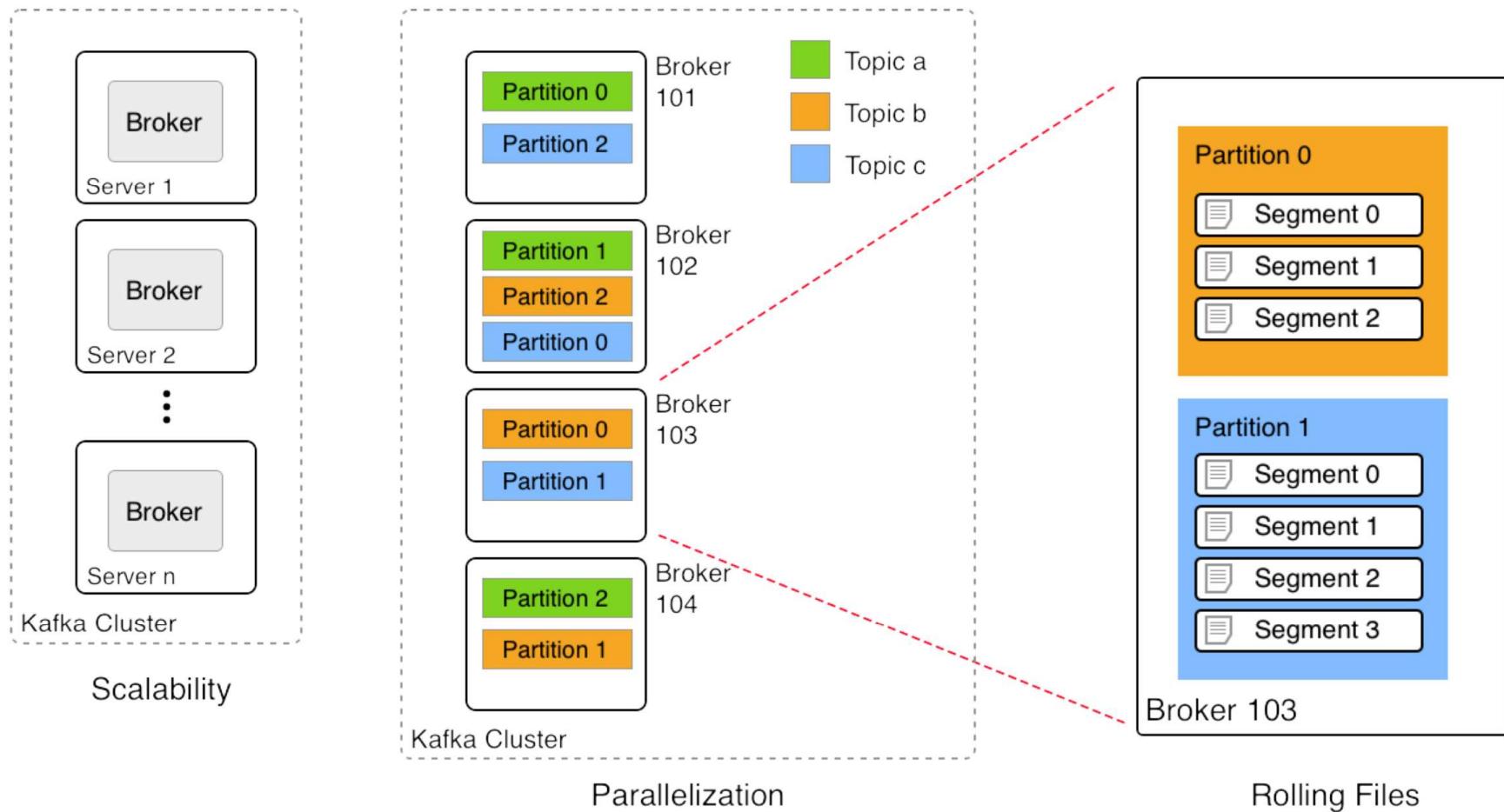
- **Topics:** Streams of “related” Messages in Kafka
  - Is a Logical Representation
  - Categorizes Messages into Groups
- Developers define Topics
- Producer ↔ Topic: N to N Relation
- Unlimited Number of Topics



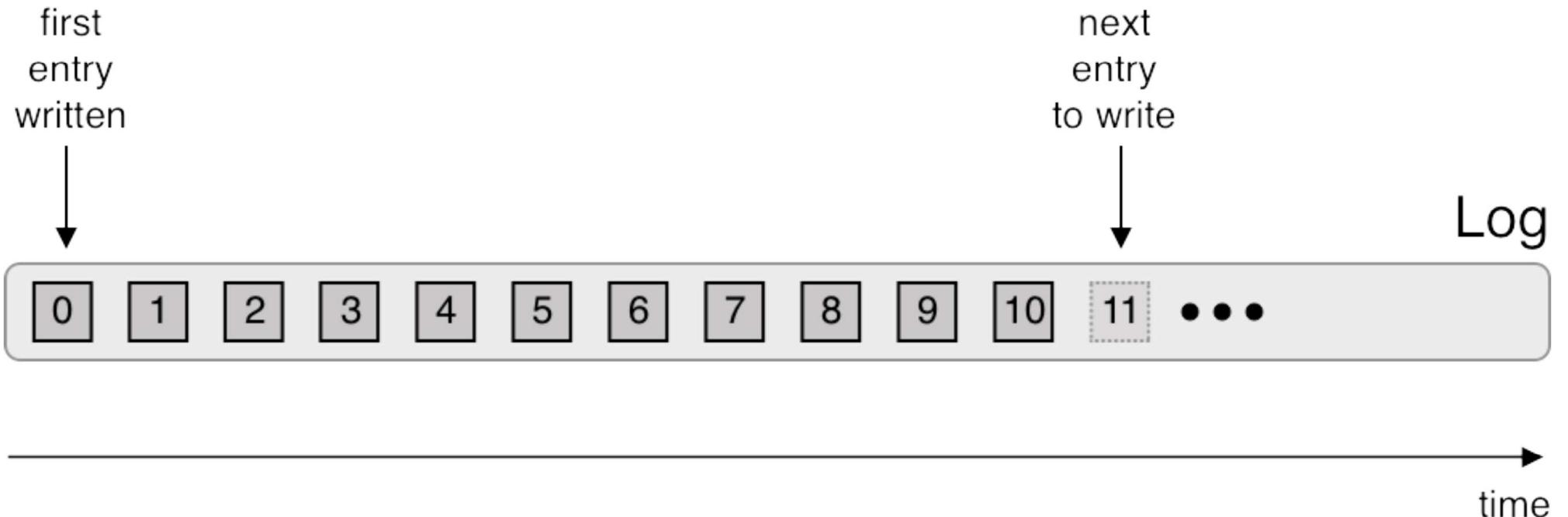
# Topics, Partitions, and Segments



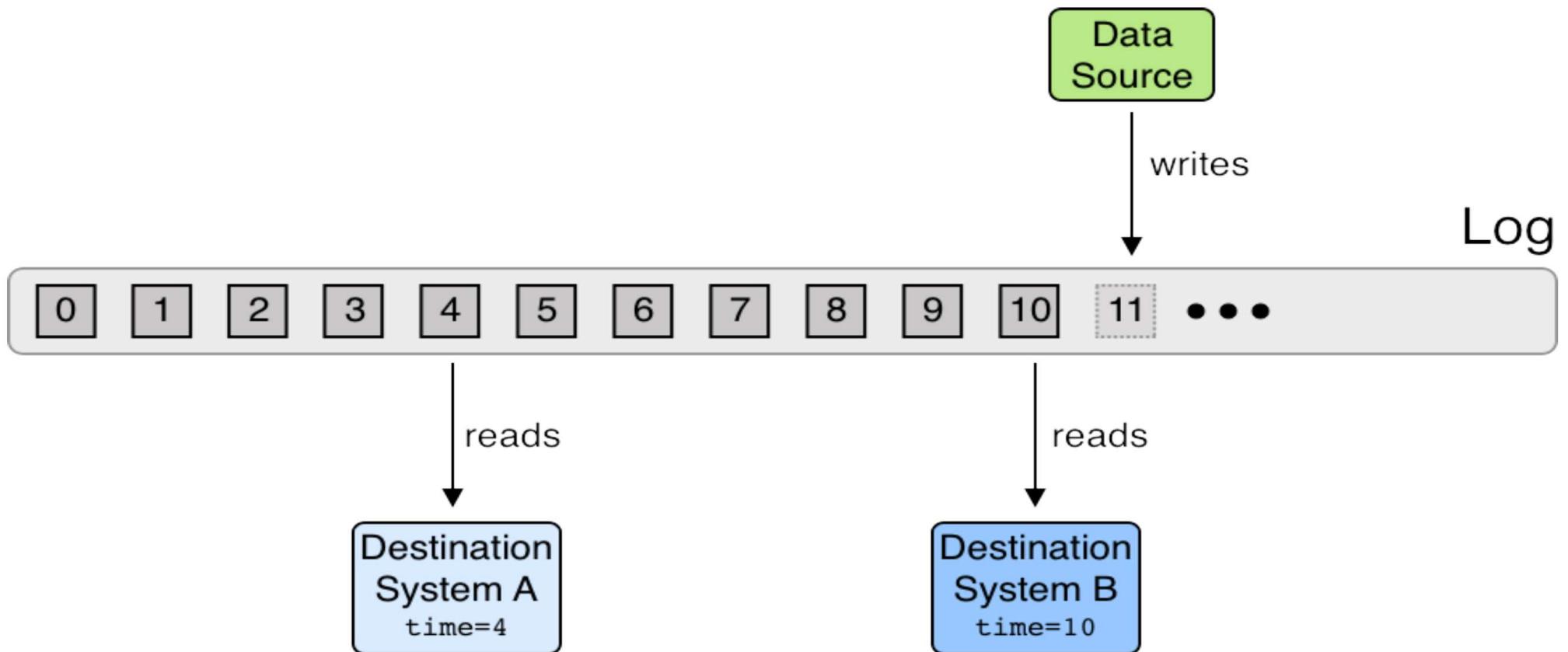
# Topics, Partitions, and Segments



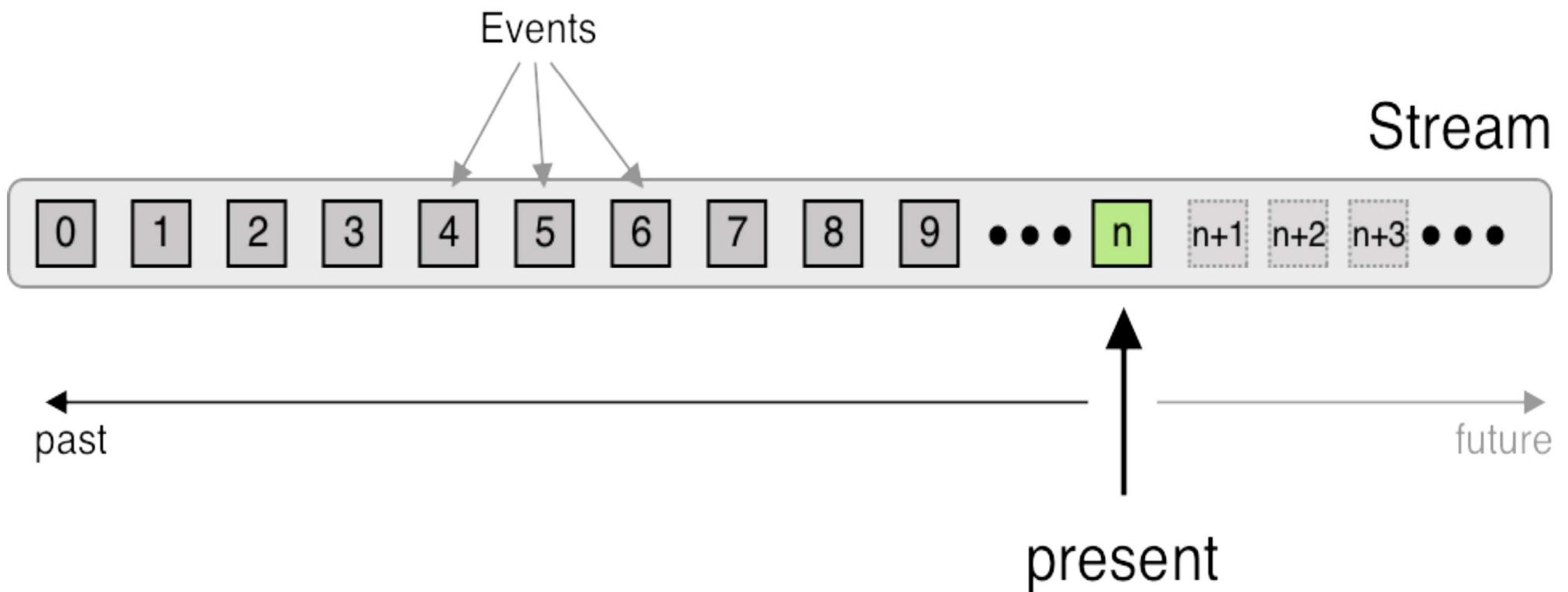
# The Log



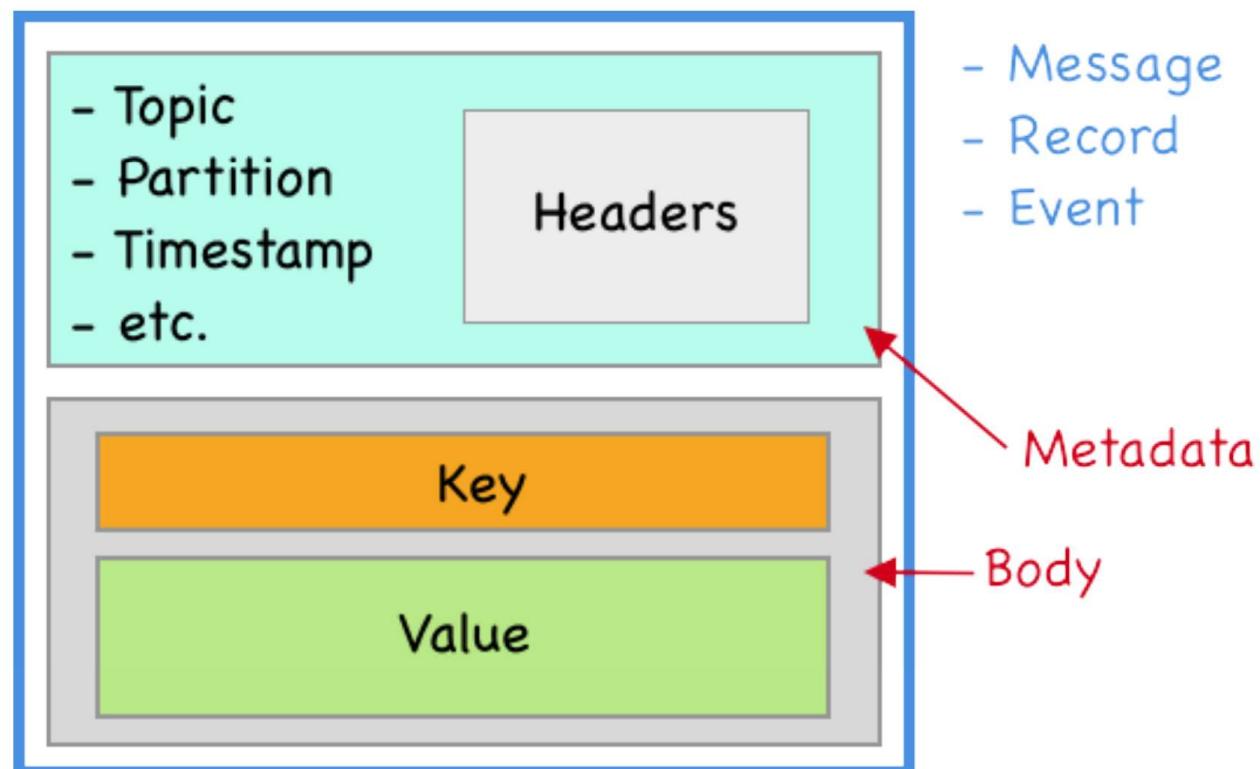
# Log Structured Data Flow



# The Stream



# Data Elements

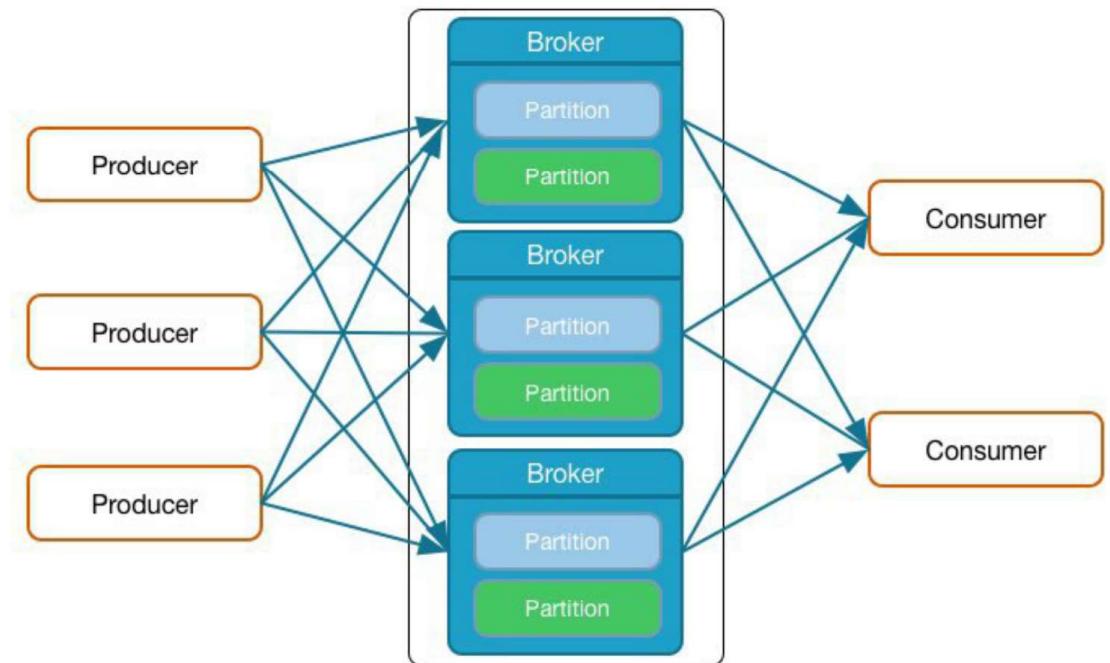


# Brokers Manage Partitions

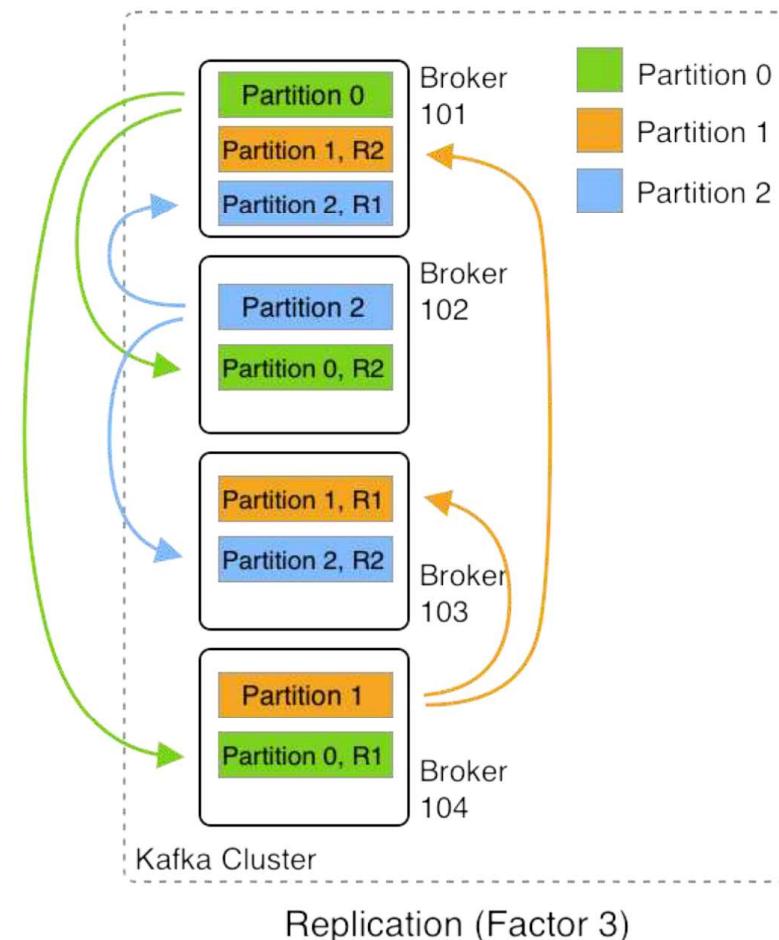
- Messages of Topic spread across Partitions
- Partitions spread across Brokers
- Each Broker handles many Partitions
- Each Partition stored on Broker's disk
- Partition: 1..n log files
- Each message in Log identified by *Offset*
- Configurable Retention Policy

# Broker Basics

- Producer sends Messages to Brokers
- Brokers receive and store Messages
- A Kafka Cluster can have many Brokers
- Each Broker manages multiple Partitions



# Broker Replication



# Producer Basics

- Producers write Data as Messages
- Can be written in any language
  - Native: Java, C/C++, Python, Go,, .NET, JMS
  - More Languages by Community
  - REST Server for any unsupported Language
- Command Line Producer Tool

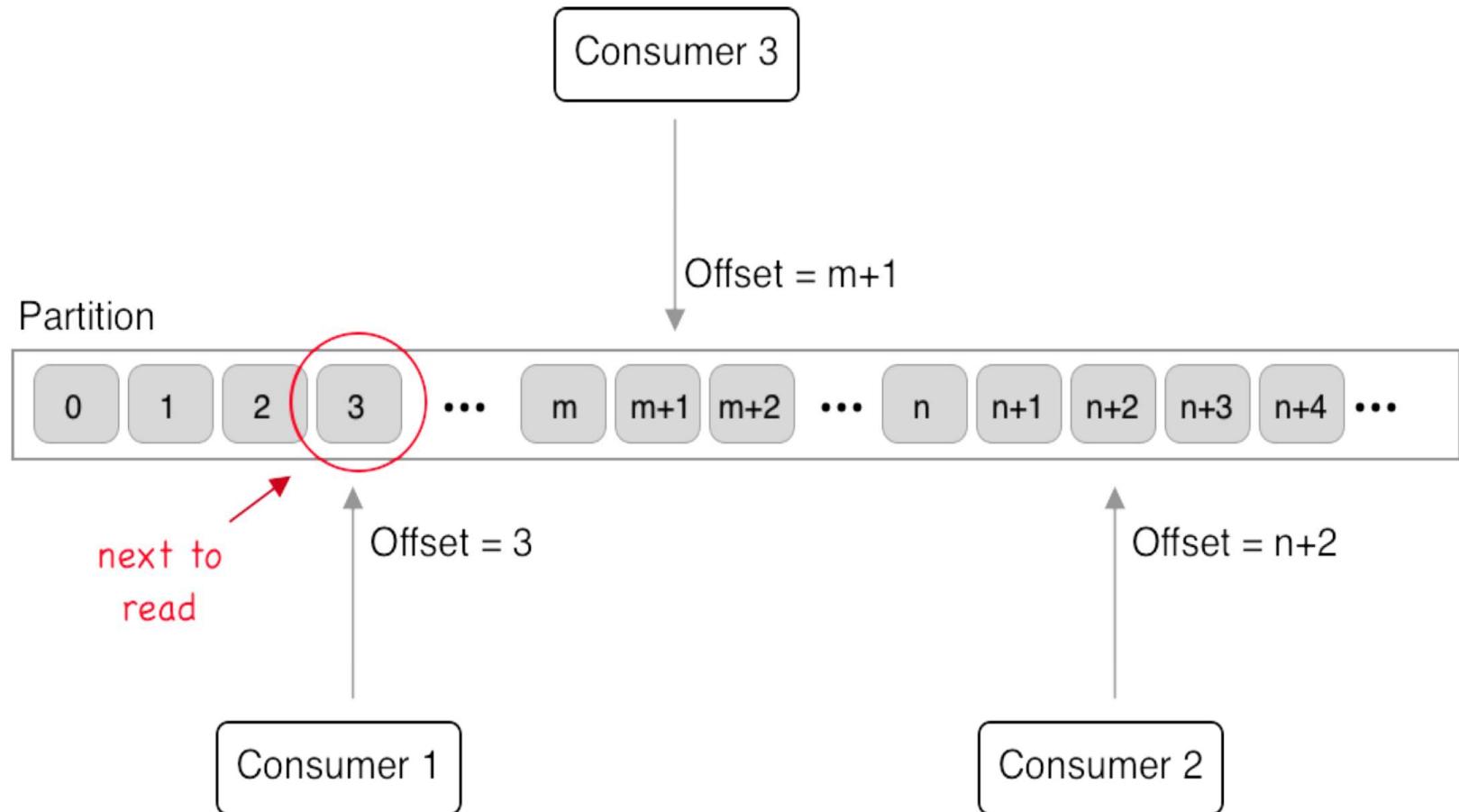
# Load Balancing and Semantic Partitioning

- Producers use a Partitioning Strategy to assign each message to a Partition
- Two Purposes:
  - Load Balancing
  - Semantic Partitioning
- Partitioning Strategy specified by Producer
  - Default Strategy: `hash(key) % number_of_partitions`
  - No Key → Round-Robin
- Custom Partitioner possible

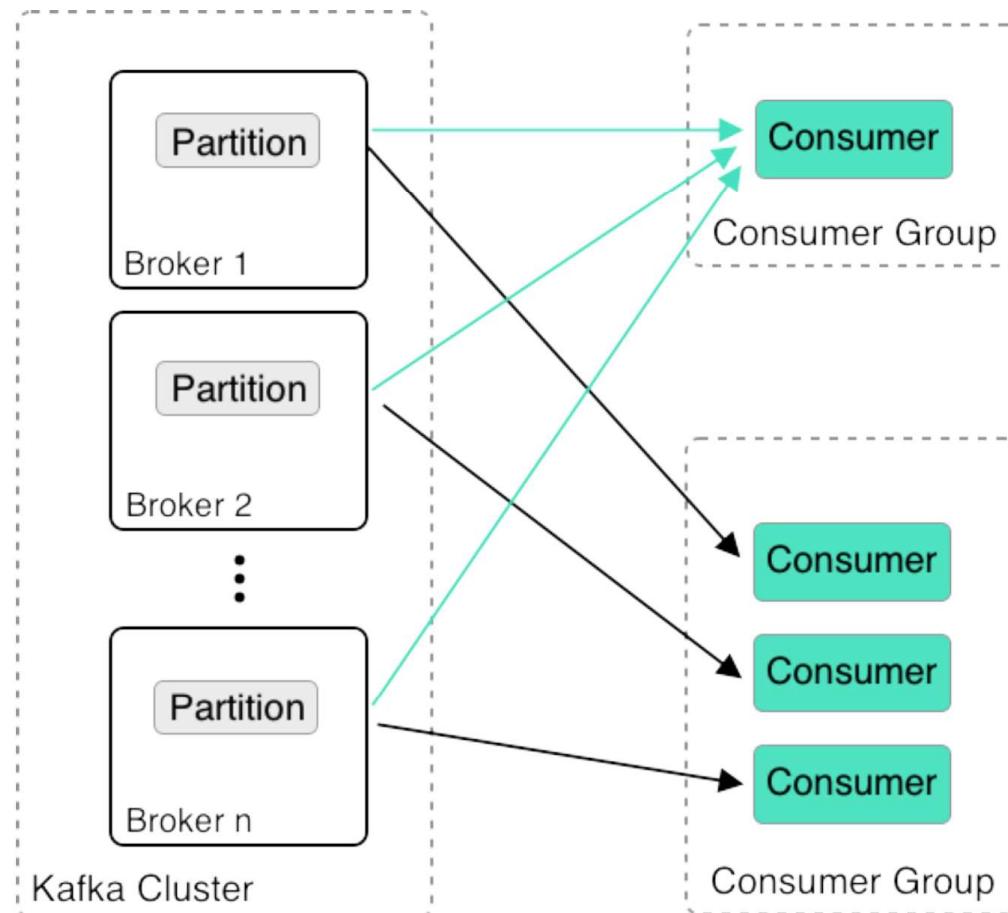
# Consumer Basics

- Consumers **pull** messages from 1..n topics
- New inflowing messages are automatically retrieved
- Consumer offset
  - Keeps track of the last message read
  - Is stored in special topic
- CLI tools exist to read from cluster

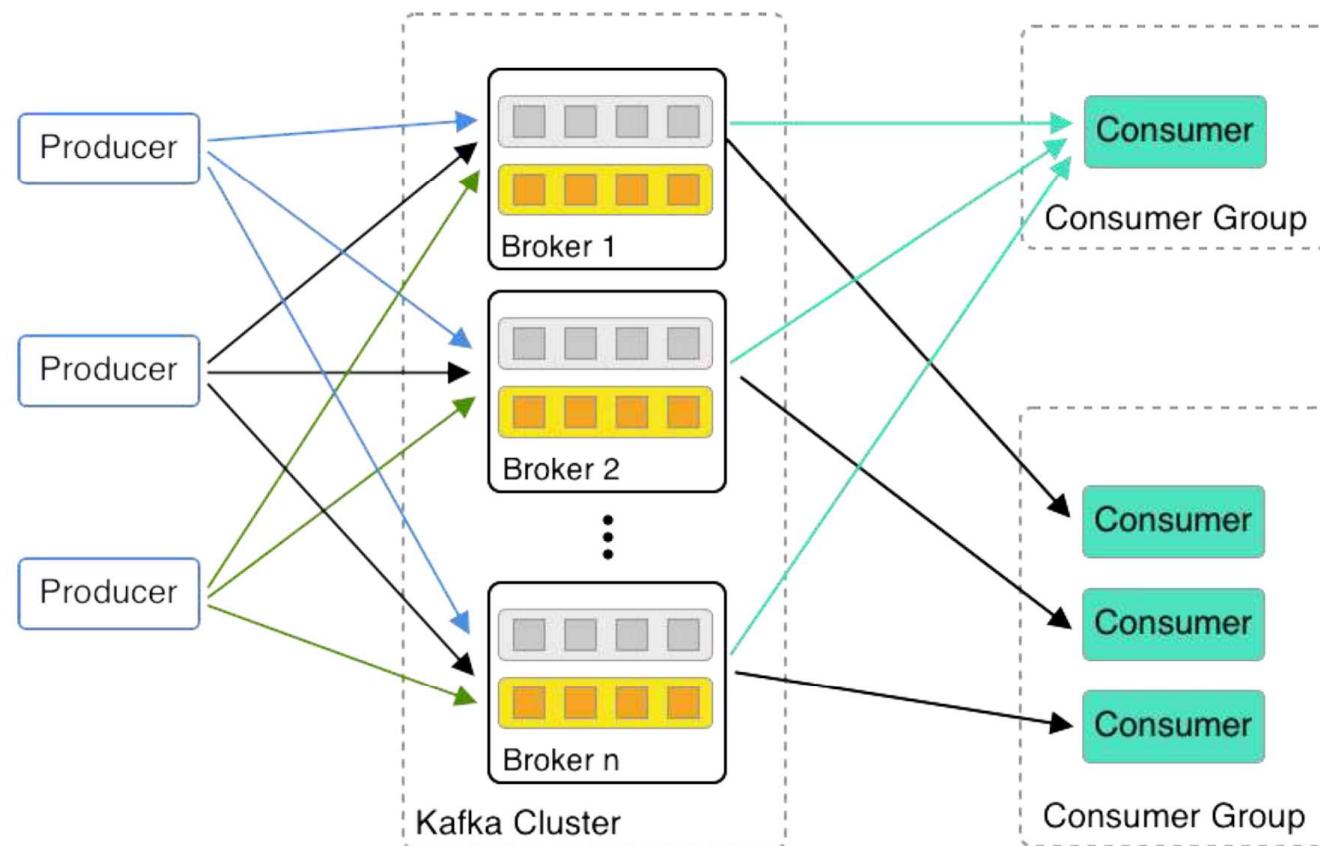
# Consumer Offset



# Distributed Consumption



# Scalable Data Pipeline



# Q&A



## Questions:

- Why do we need an odd number of ZooKeeper nodes?
- How many Kafka brokers can a cluster maximally have?
- How many Kafka brokers do you minimally need for high availability?
- What is the criteria that two or more consumers form a consumer group?