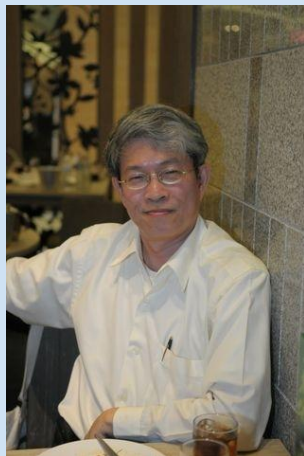


---

# Service Coordination Kafka



Dr. Yunyong Teng-amnuay

(Retired)

Dept. of Computer Engineering

Faculty of Engineering

Chulalongkorn University

November 7, 2024

---

# Kafka 101

Adapted from Stanislav Kozlovski's article.  
<https://highscalability.com/untitled-2/>



**HS Editor**

May 9, 2024 — 15 min read

<https://ambiyansyah.medium.com/the-kafka-protocol-and-how-it-works-dc8fd3653c69#:~:text=The%20Kafka%20protocol%20is%20a,of%20data%20in%20real%2Dtime.>

## The Kafka protocol and how it works



Ambiyansyah Risyal · [Follow](#)

2 min read · Dec 14, 2022

# Application Domains

1. In a real-time analytics pipeline, where data from multiple sources is ingested, processed, and analyzed in real-time.
2. In a financial trading platform, where data from multiple exchanges is ingested, processed, and used to make trading decisions.
3. In a social media platform, where data from multiple sources (e.g., user posts, comments, likes) is ingested, processed, and used to generate real-time recommendations.
4. In an IoT system, where data from multiple sensors is ingested, processed, and used to trigger real-time actions (e.g., turning on a light or sending an alert).

# Real-World Examples of Kafka Usage

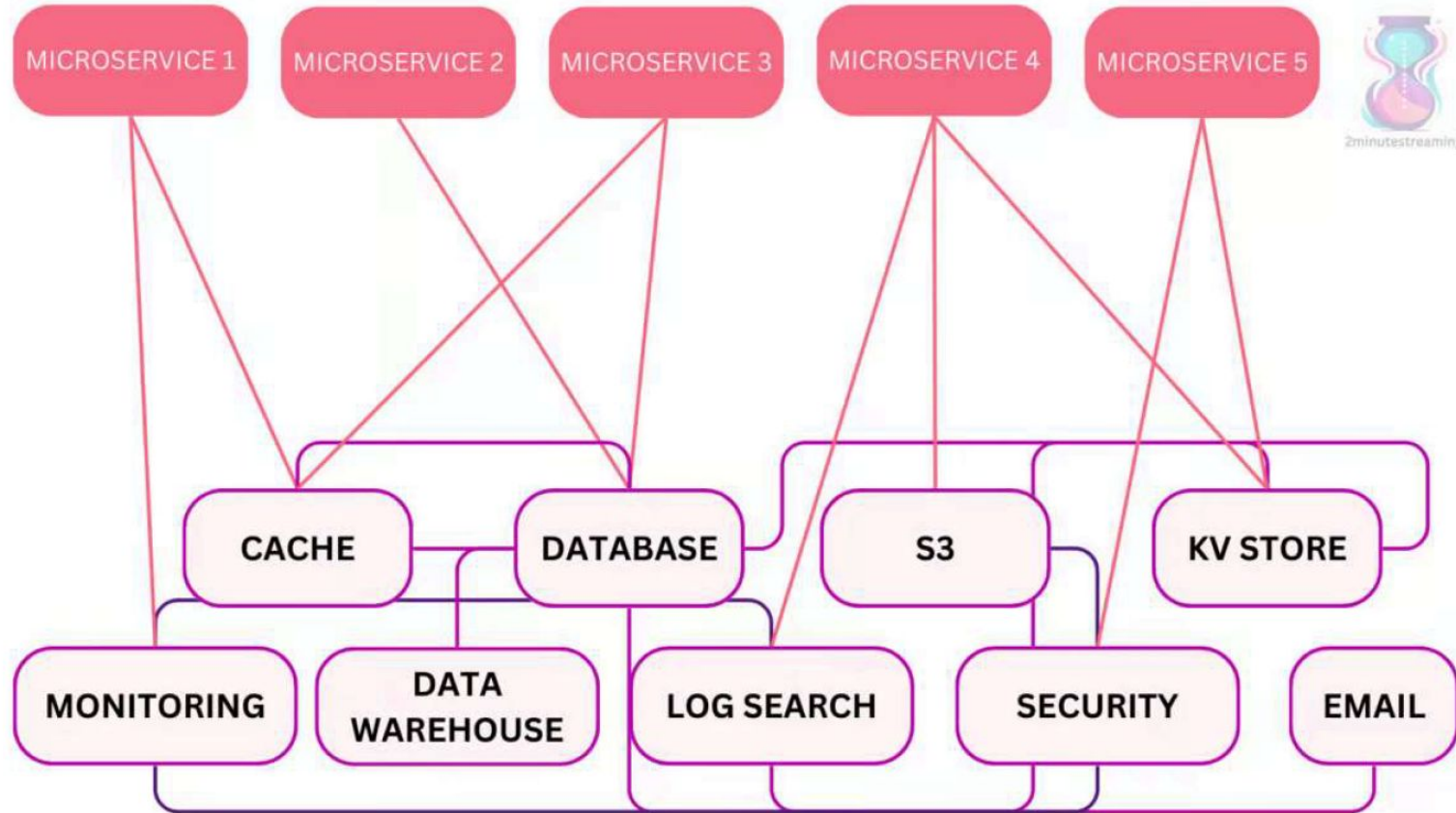
<https://www.instaclustr.com>

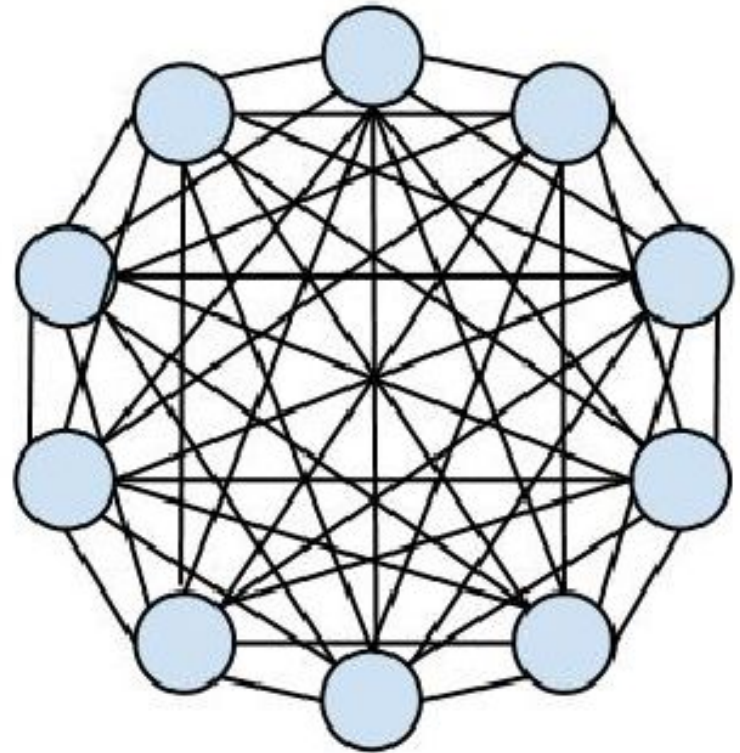
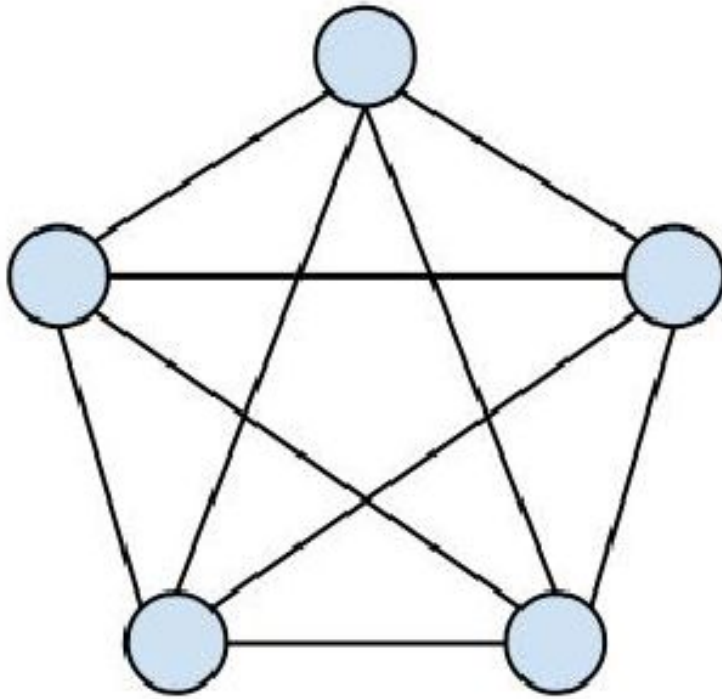
1. Goldman Sachs, a leading global investment banking firm, leveraged Apache Kafka for its SIEM system. Kafka enabled them to efficiently process large volumes of log data, significantly enhancing their ability to detect and respond to potential security threats in real-time. (SIEM: Security Information and Event Management).
2. Netflix, a major player in the streaming service industry, uses Apache Kafka for real-time monitoring and analysis of user activity on its platform. Kafka helps Netflix in handling millions of user activity events per day, allowing them to personalize recommendations and optimize user experience.
3. Pinterest utilizes Kafka for stateful stream processing, particularly in their real-time recommendation engine. Kafka's capability to process data streams in real-time allows Pinterest to update user recommendations based on their latest interactions.
4. British Sky Broadcasting (Sky UK) implemented Kafka in their video recording systems, particularly for handling data streams from their set-top boxes. Kafka's role in buffering and processing video data has been crucial for improving customer viewing experiences and content delivery.

**Apache Kafka** solves the service coordination problem. It is a central nervous system inside a company. It is optimized for large throughput at millions of messages a second and a lot of data at terabyte level.

The **Kafka** *protocol* is a popular, open-source, high-throughput, low-latency messaging platform that is designed to handle large volumes of data from multiple sources. It is a publish-subscribe messaging system that is typically used to store and process streams of data in real-time.

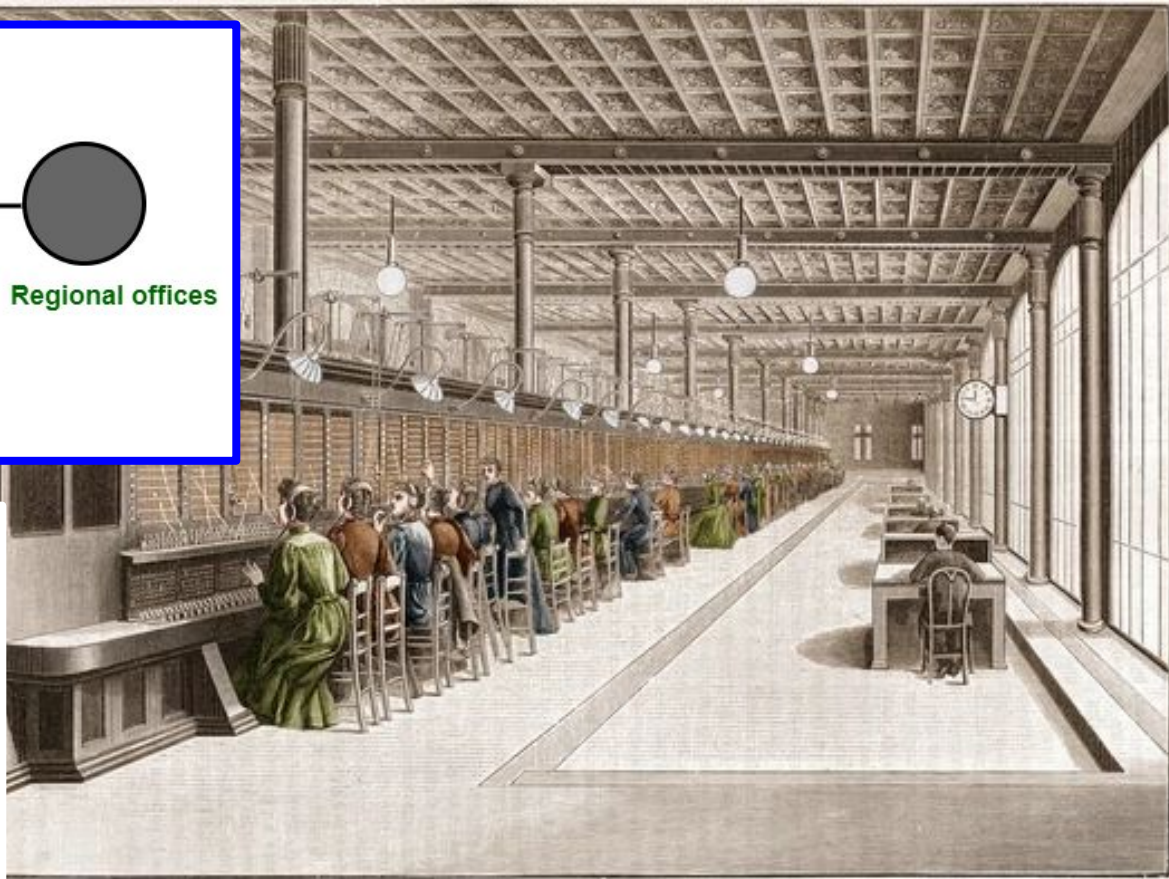
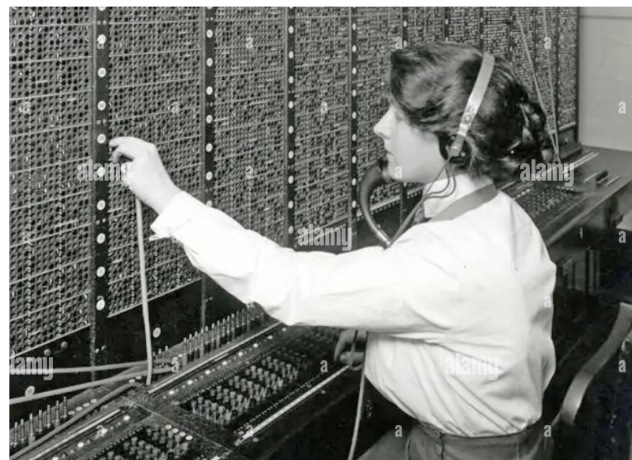
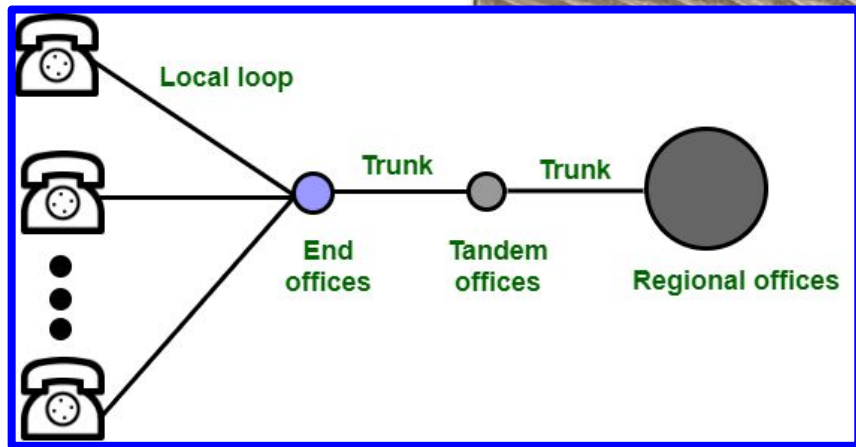
# Problems with microservice proliferation.





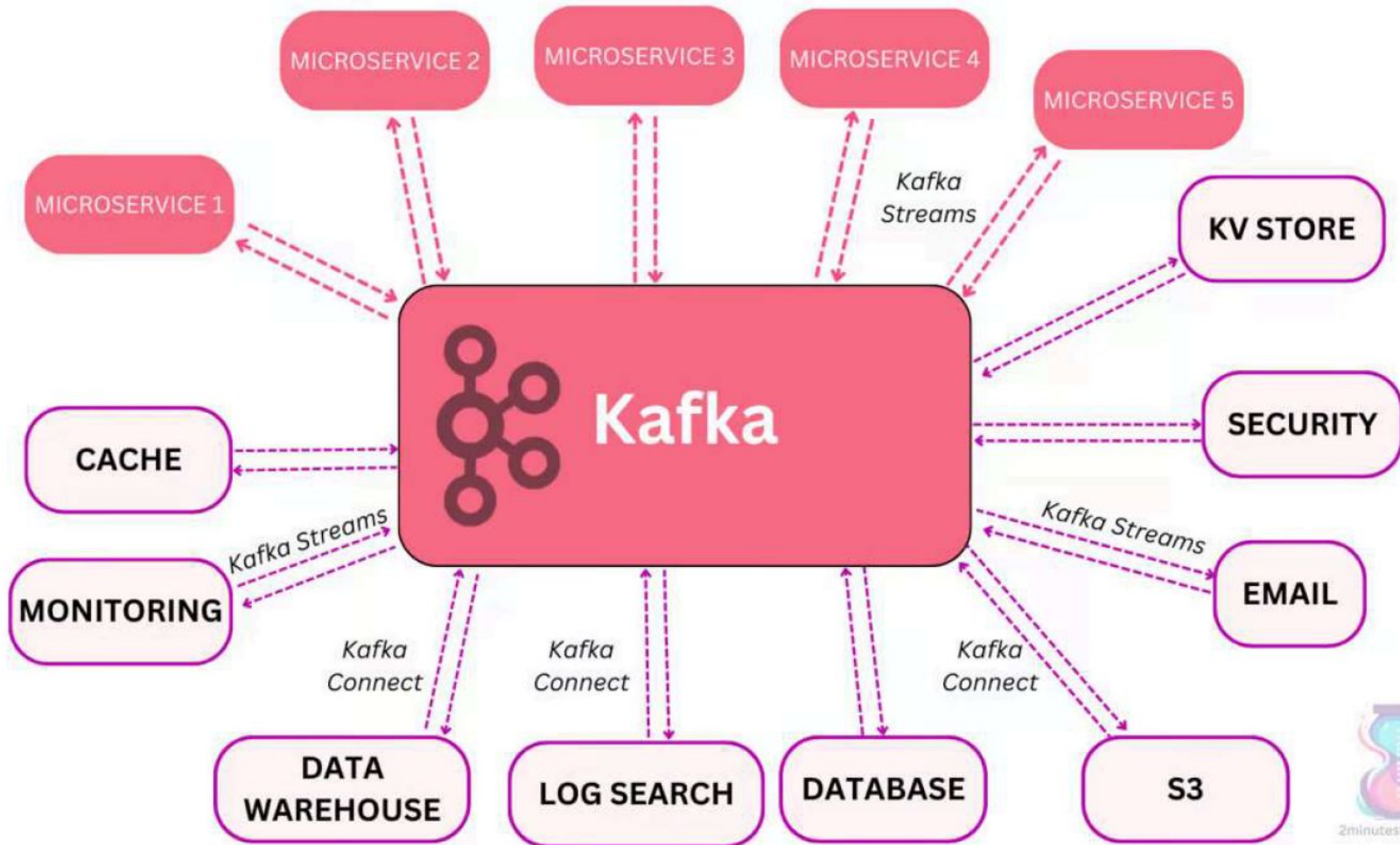
Proliferation of Connectivities





Telephone Switchboard





# Kafka

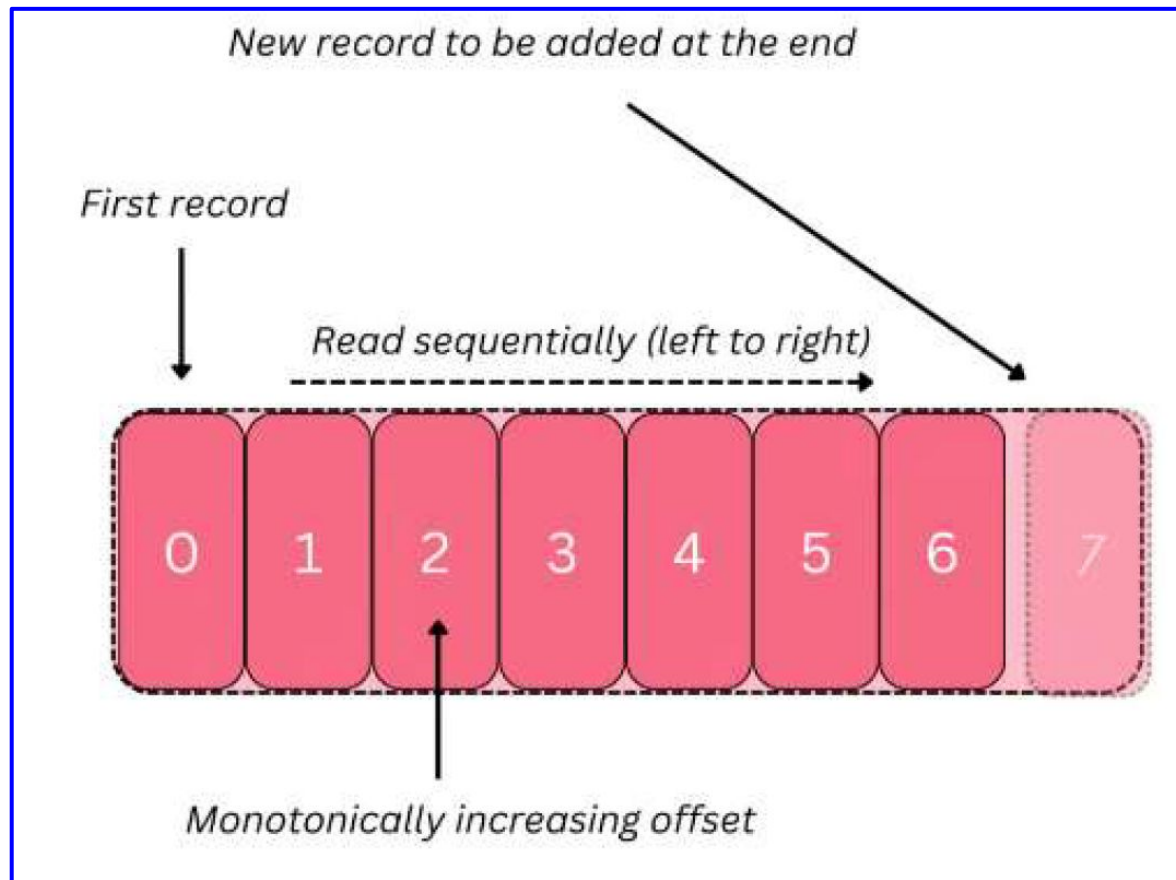
(high-level concepts - to be review as years go by)

- Distributed, partitioned, and replicated logs.
- Producers generate data and publish to logs.
- Consumers read data from logs and process them.
- Request-response model.
- Brokers manage logs and keep records and message ordering.
- Concurrent reads → high performance.
- On-premise deployment.

# Logs as Data Structure

- Data are stored as ***topics***, with logs as their data structure.
- A log is an ordered queue storing records sequentially.
- The log is ***immutable*** (cannot be changed).
- $O(1)$ : read/write at each ends.
- Design for disks. Minimize seeks. Read ahead in blocks.
- Network becomes the bottleneck.
- Persistence to disk.
- A topic is split into ***partitions/segments/shards***.
- Each partition is ***duplicated*** N times for durability & availability.
- Partitions are distributed among nodes/brokers.
- ***Replication is leader-based.***

Data is stored as topics. A topic consists of log.



Entries are immutable.

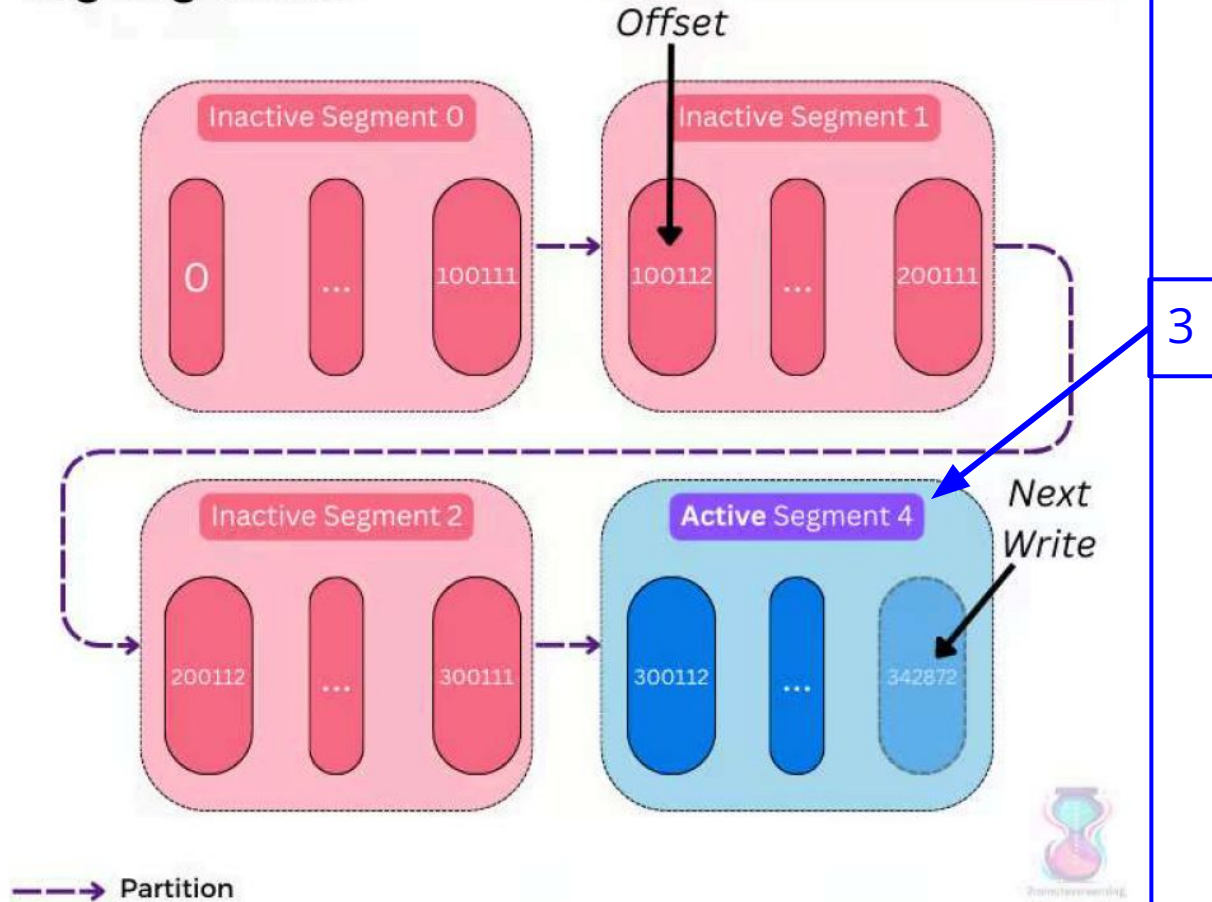
# Data Partition and Replication

- A topic/log is split into partitions/segments/shards.
- Each partition is duplicated N times for durability & availability.
- Partitions are distributed among nodes/brokers.
- Replication is leader-based.

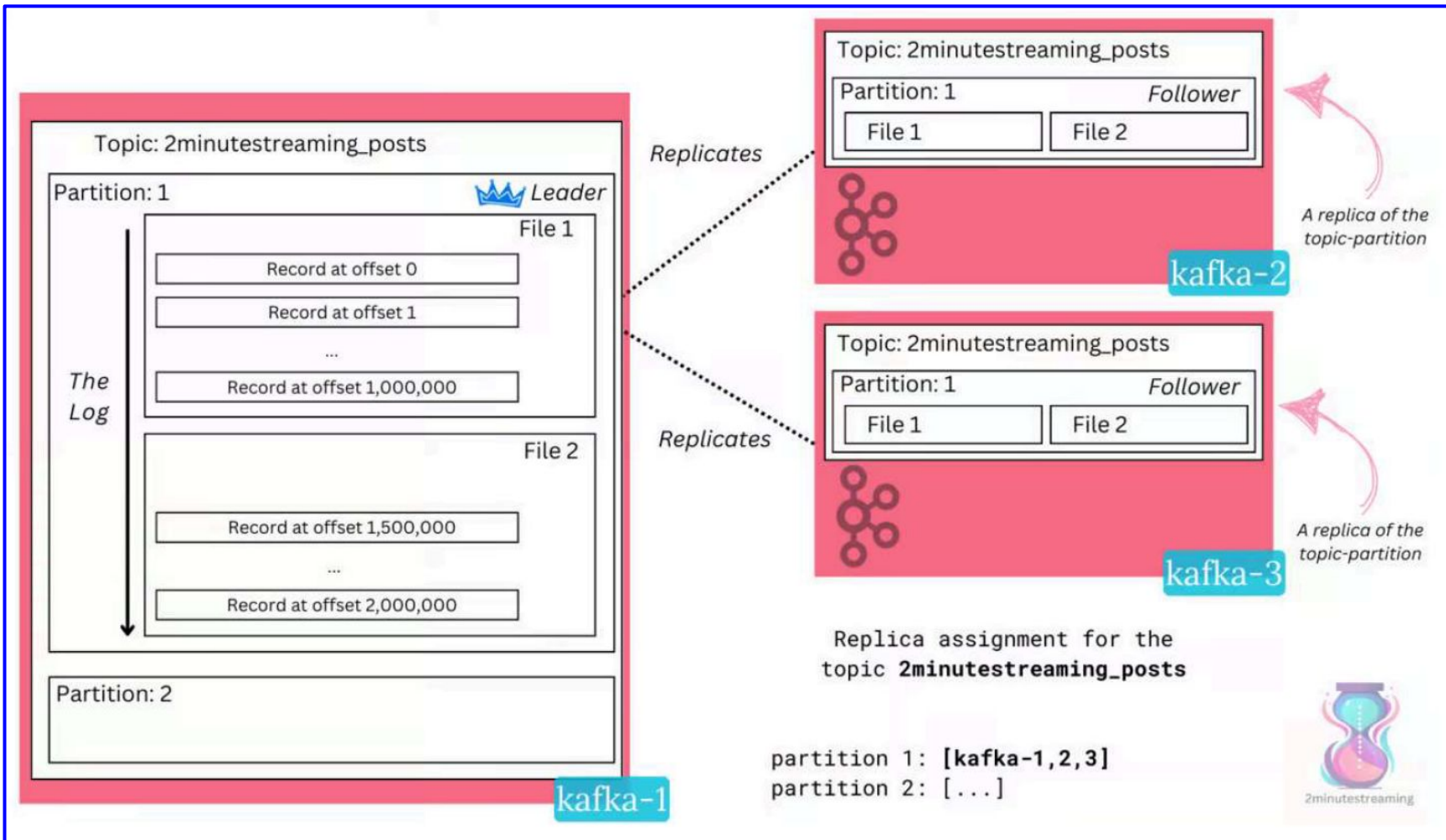
## WRITE with Update Propagation: Selectable

- Data written by clients (call producers) to the leader replica.
- ***Durability guarantee*** is configurable by producers.
- (This is fault tolerant - ***Passive Replication***).
- “acks” properties:
  - acks=0 - Do not wait for response from brokers.
  - acks=1 - Wait for leader’s acknowledgement.
  - acks=all (default) - All replica respond.

## Log Segments



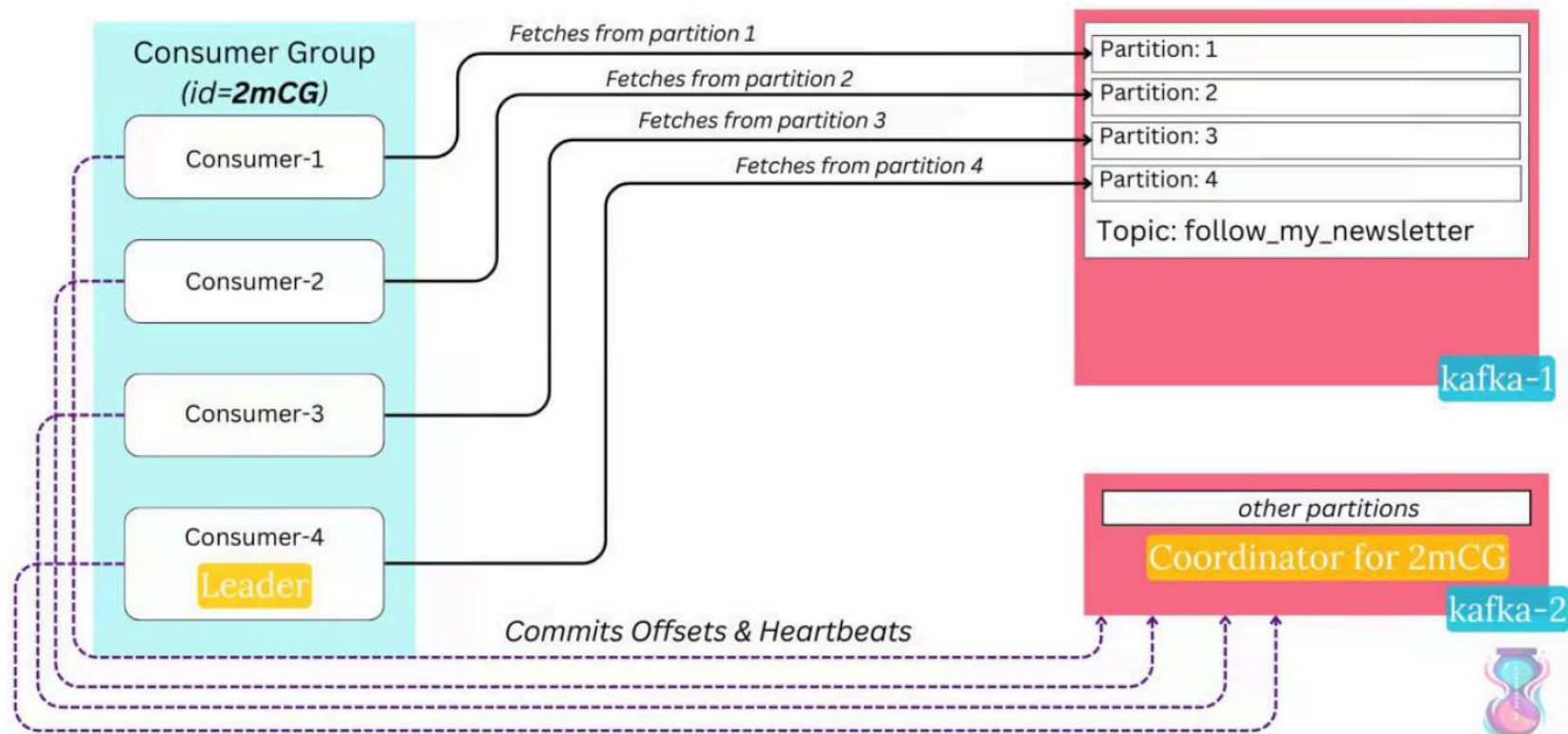




# Read

- Clients (processes) that read are called ***consumers***.
- Consumers can read from any replica, usually the nearest.
- Consumers form ***consumer groups*** with coordinator for each group.
- Members of a group synch through interactions with brokers.
- Consumers keep their progress (offset in partition) in group coordinator.
- Reading will not result in deleting of message: decoupling of read & write (consumers/producers decoupling). → ***FAST***

# Consumer Groups



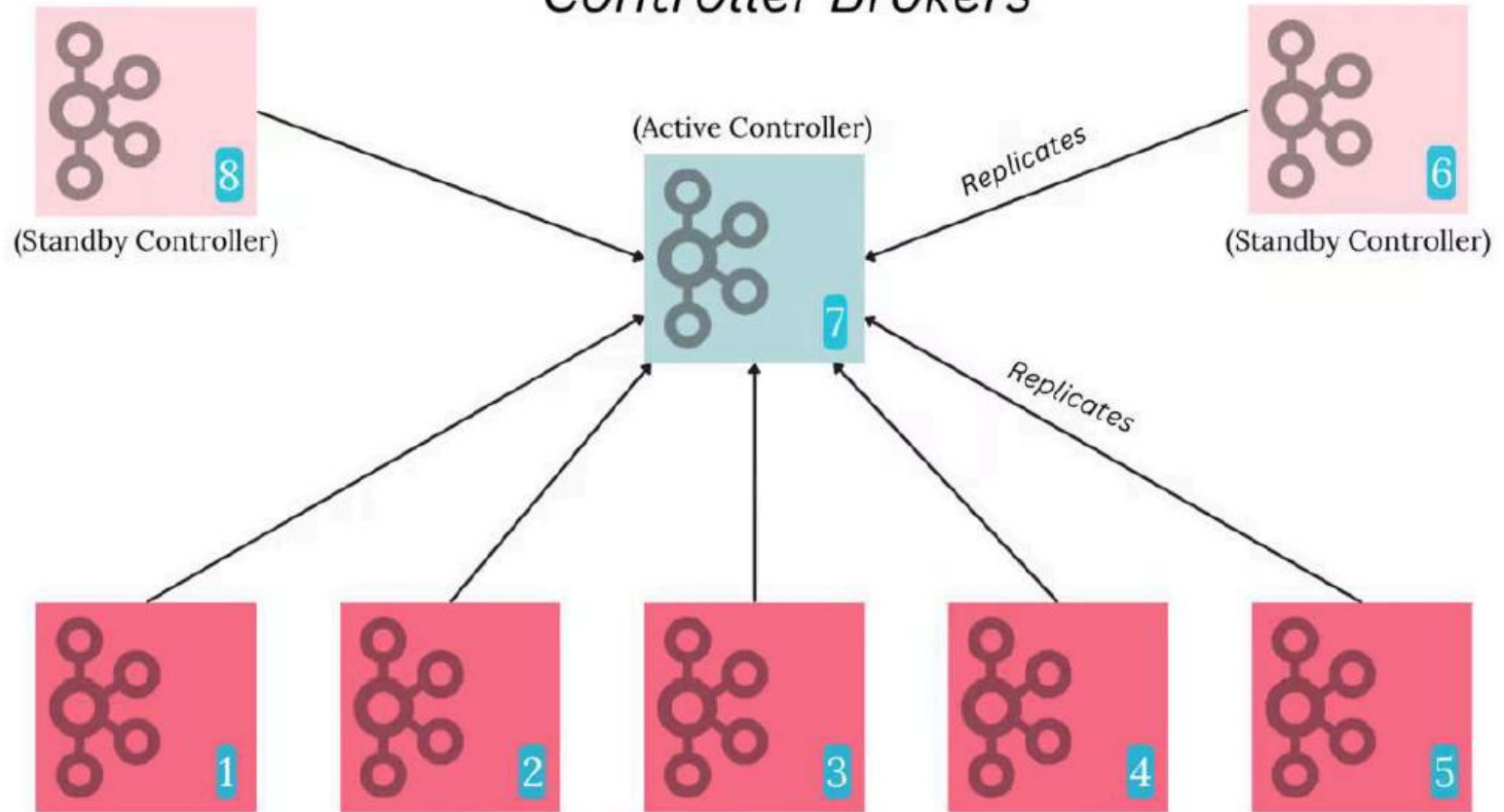
# Fault Tolerant

- Always a broker as *Active Controller* with replica (standby controllers).
- Controller controls actions based on single source of truth:
  - Creating / deleting topics
  - Adding partitions to topics
  - Modify degree of replication
  - *Leader election* (of replica set) when a leader crashes

# Consensus

- There can be failure (crash or arbitrary).
- Previously ZooKeeper performs consensus.
- ZooKeeper is also a single source of truth.
- Now using KRaft (Kafka ***Raft***).
- Metadata uses the log system with 3 brokers.
- Broker uses metadata log to rebuild the system.  
This is recovery file.

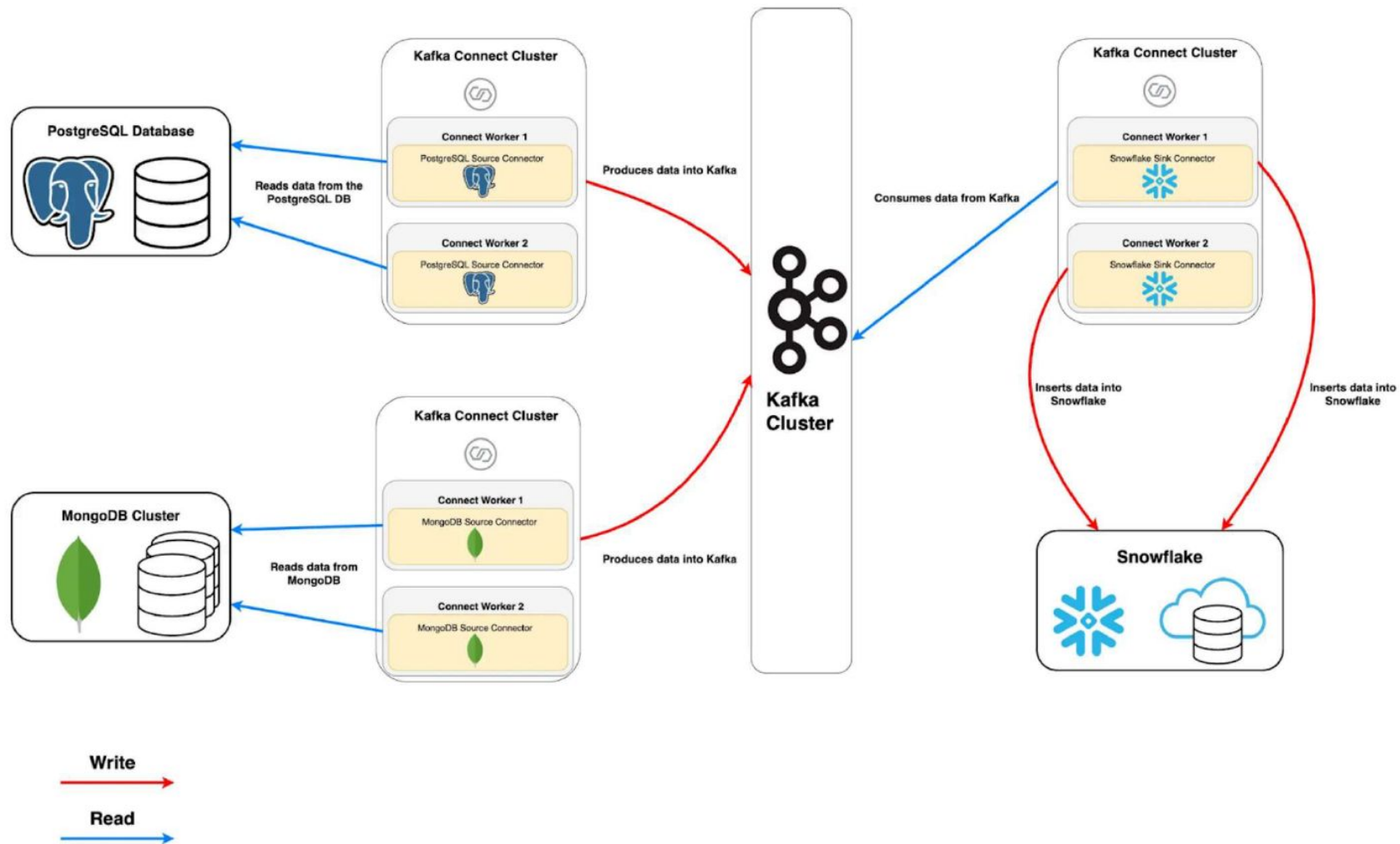
## Controller Brokers





# Kafka Connect

- Popular systems as sources and sinks.
- Standalone & distributed modes for single node & cluster.
- Connect *clusters* and *workers*
  - Fault tolerant.
  - Exactly-once invocation semantic.
  - Message ordering.
  - Use topics for their config, status, check points.



# Kafka Streams

- Real-time stream processing clients.
- Stream joining, etc.
- Exactly-once processing semantics.

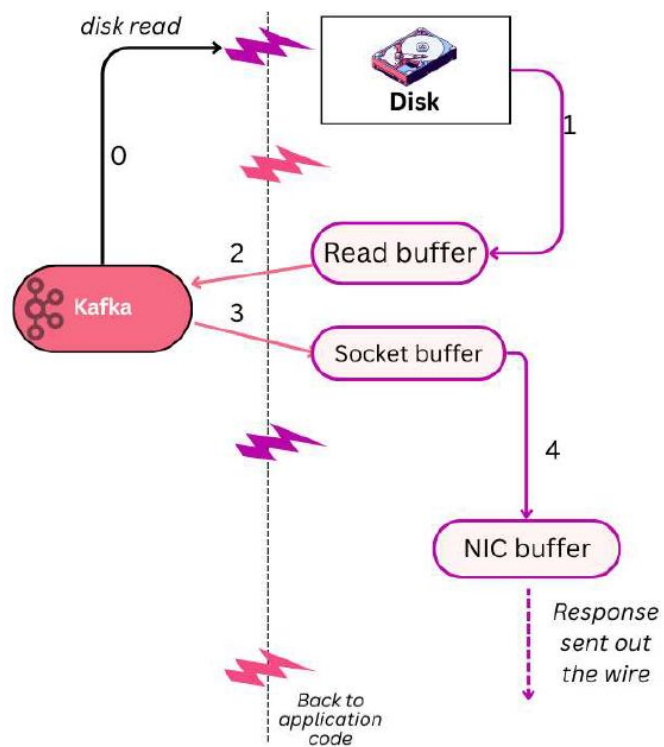
# Variants / Alternatives

- Confluent: cloud-native Cora
- RedPanda: C++ & WarpStream
- Apache Pulsar: A Kafka-compatible messaging system
- Azure EventHubs: A messaging system that integrates with other Microsoft technology
- Redpanda: faster, cheaper, and simpler than Kafka
- Upstash: includes Redis, vector db, Kafka, and HTTP-based messaging
- Amazon Simple Queue Service (SQS), Apache ActiveMQ & Spark, **Redis**, Amazon Kinesis, RabbitMQ: Messaging systems that can be used to build distributed systems

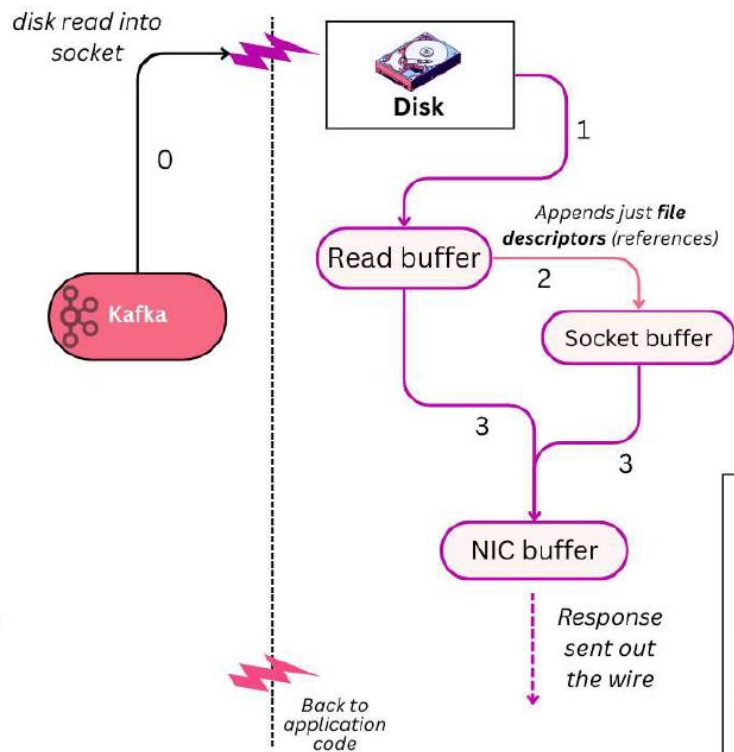
Q & A

Thank You !!





## No Zero Copy



## Zero Copy



### Legend

-  User -> Kernel mode context switch
-  Kernel -> User mode context switch
-  DMA copy
-  CPU copy