Data Engineering

LTAT.02.007

Ass Prof. Riccardo Tommasini

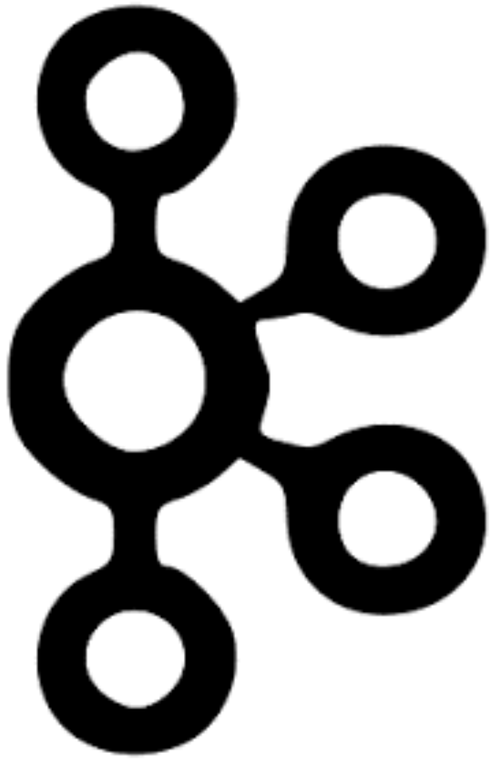
Assistants: Fabiano Spiga, Mohamed Ragab, Hassan Eldeeb

https://courses.cs.ut.ee/2020/dataeng

Forum

Moodle

Apache Kafka[^1]



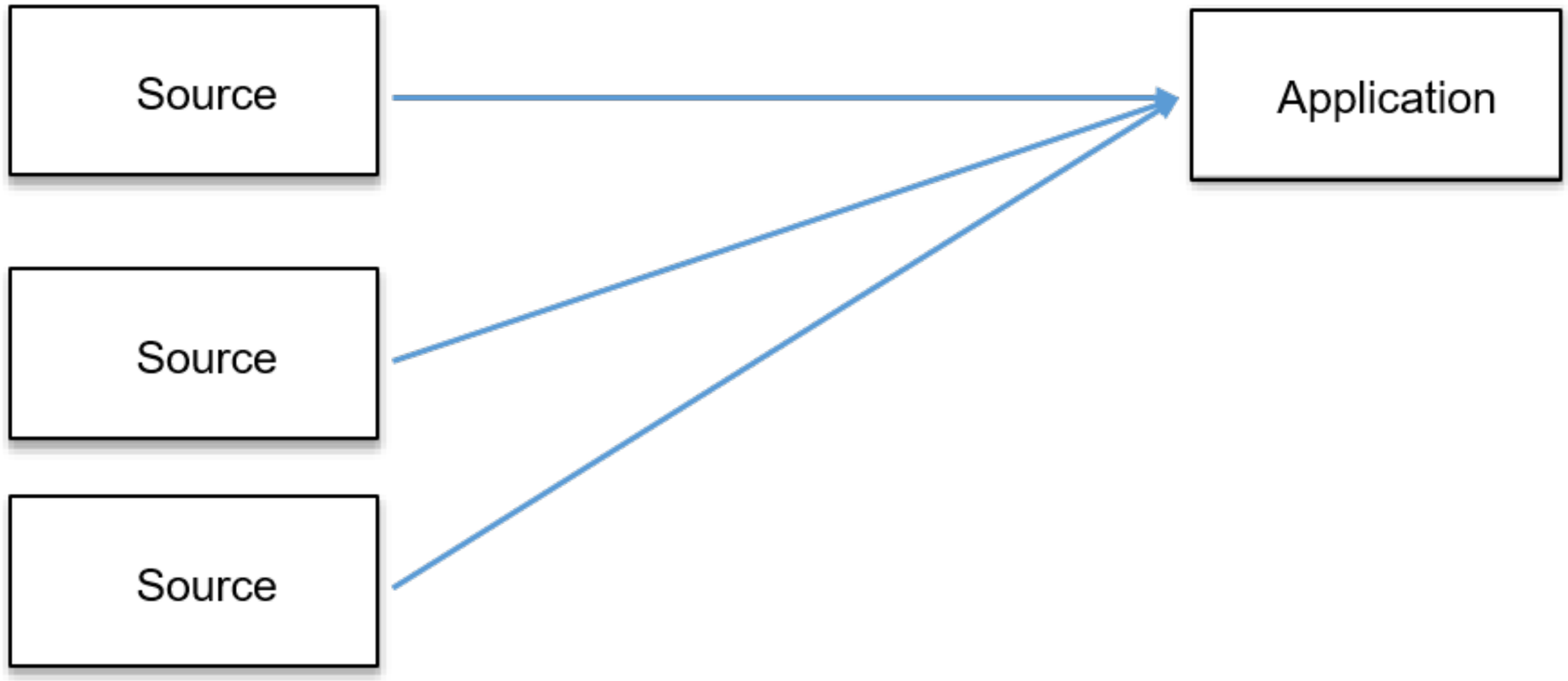
An overview

Motivation

Data pipelines start with a small number of systems to integrates. A single ETL (extract, transform, load) process move data from the source to the interested applications.

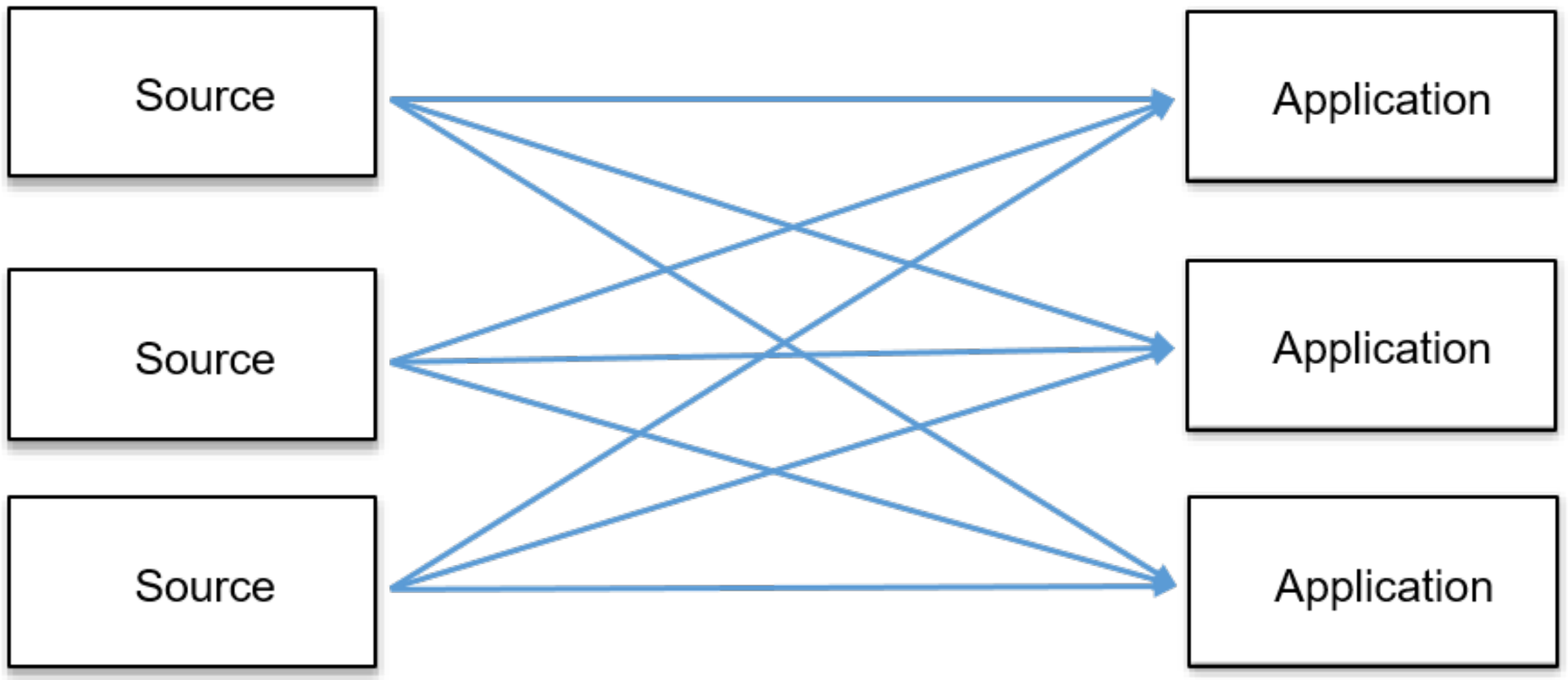
Motivation

But data pipeline grow over time. Adding new system causes the need of new ETL process. The code-base grows together with data formats and services.



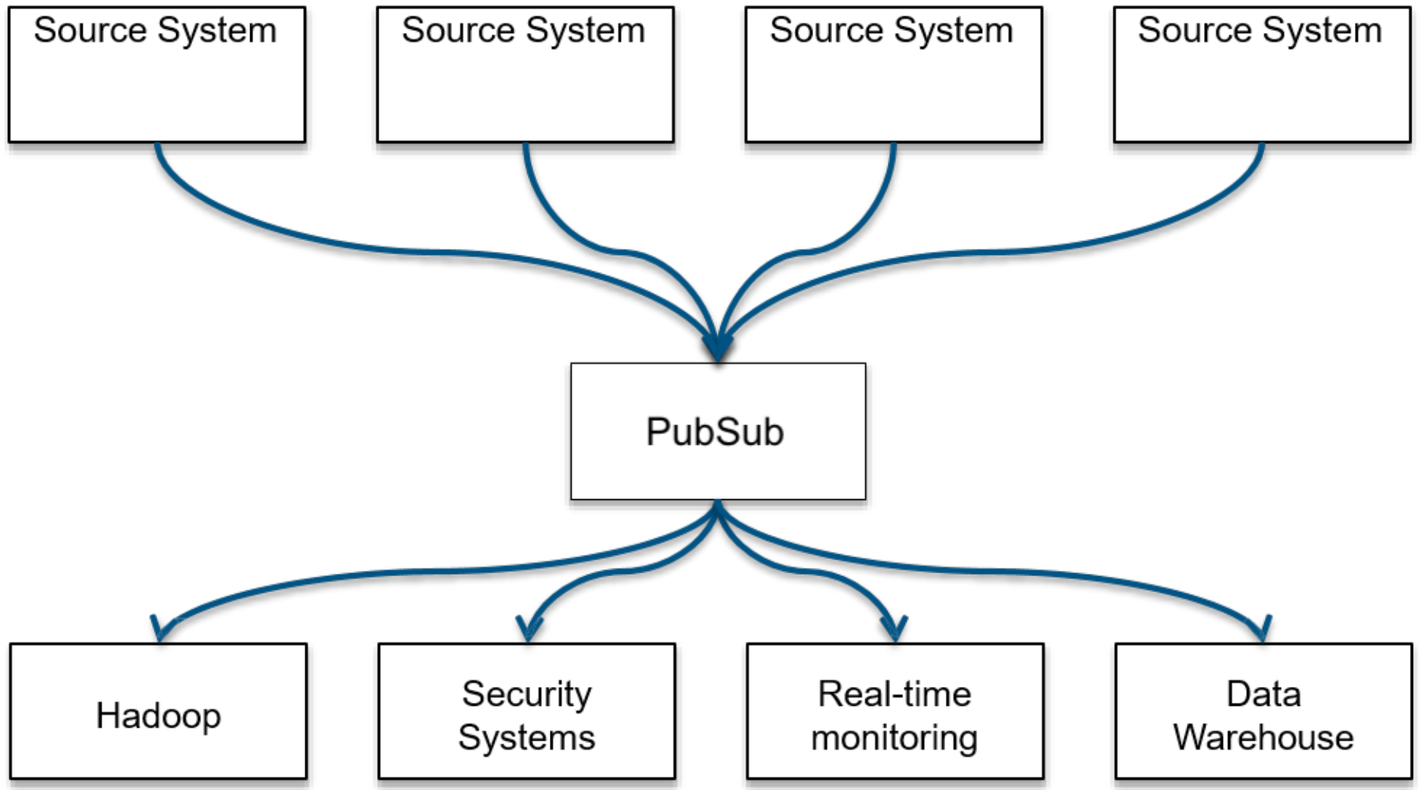
Motivation

Things end up messy when sources and sinks are coupled!



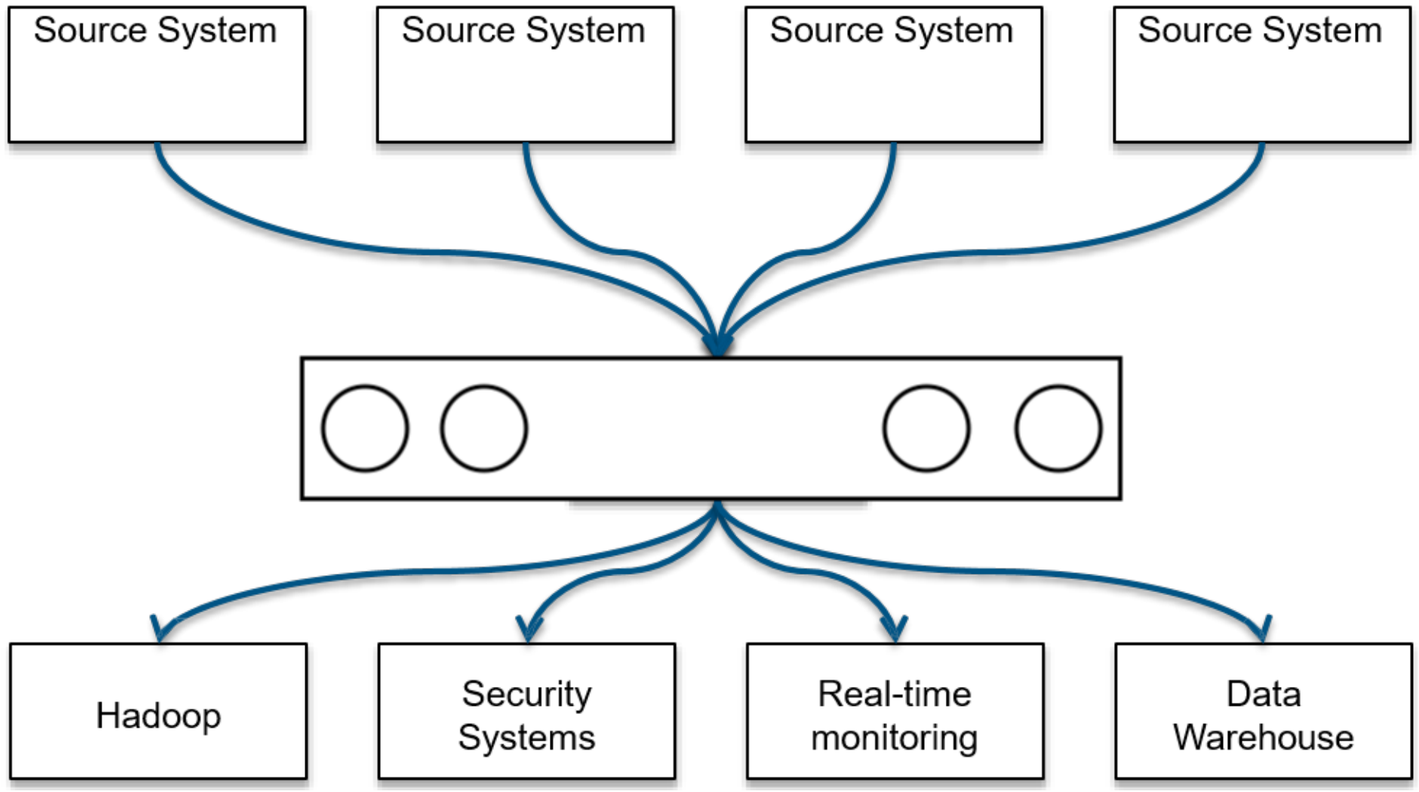
An alternative: Publish/Subscribe

PubSubs decouple data sources and their consumers making communication asynchronous and processing scalable.



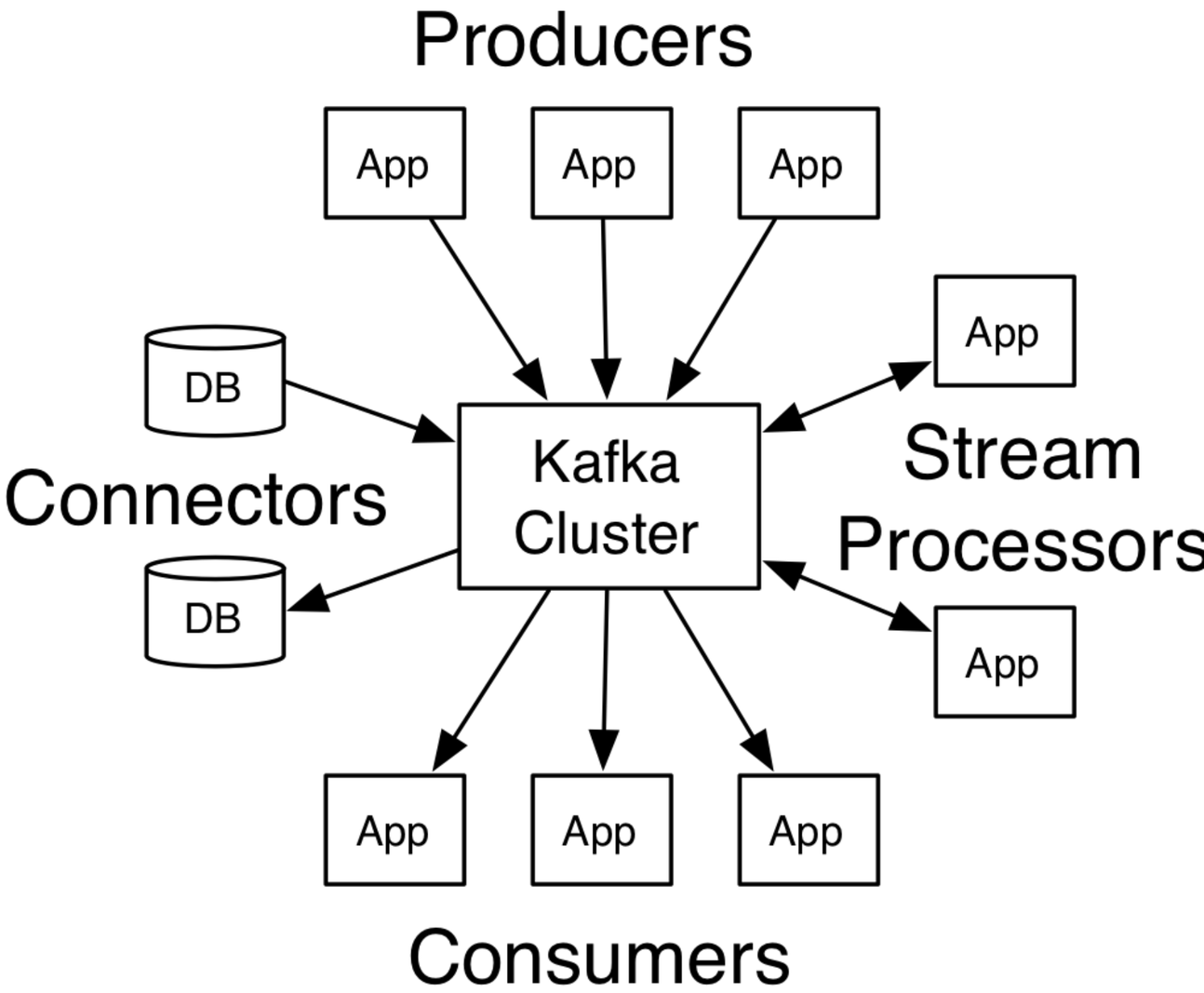
An alternative: Publish/Subscribe

PubSubs organize messages logically so that it is easier for the interested consumers to access.

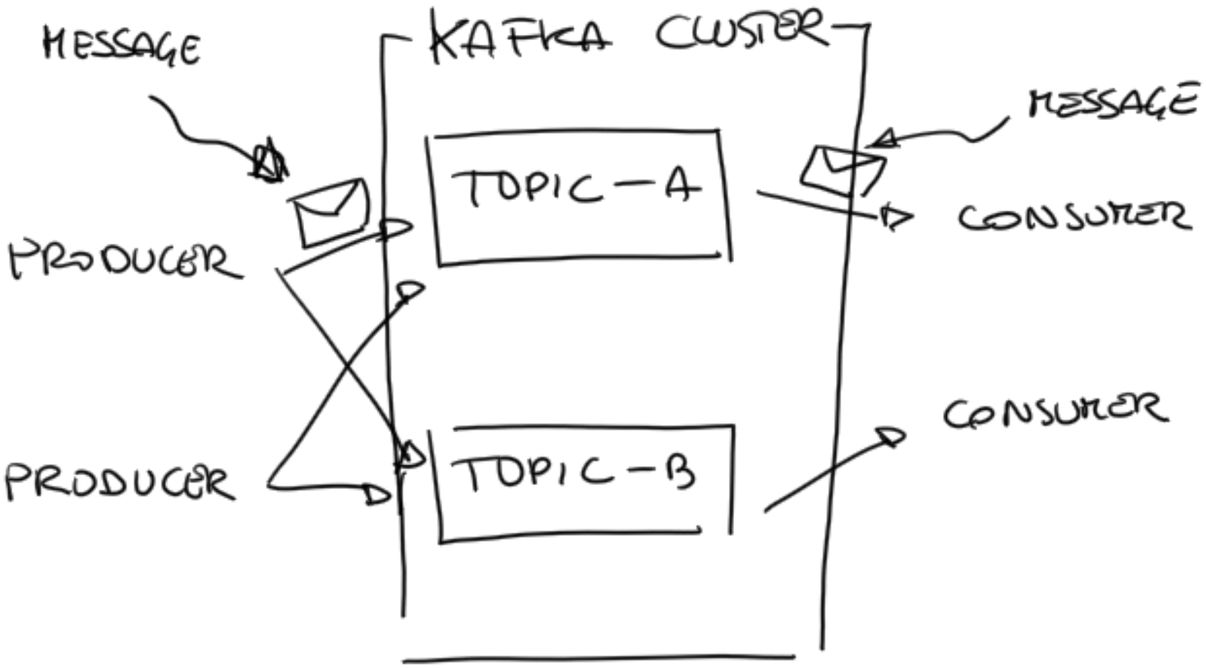


Apache Kafka

Apache Kafka is an horizontally scalable, fault-tolerant, publish-subscribe system. It can process over 1 trillion messages without neglecting durability, i.e., it persists data on disk.

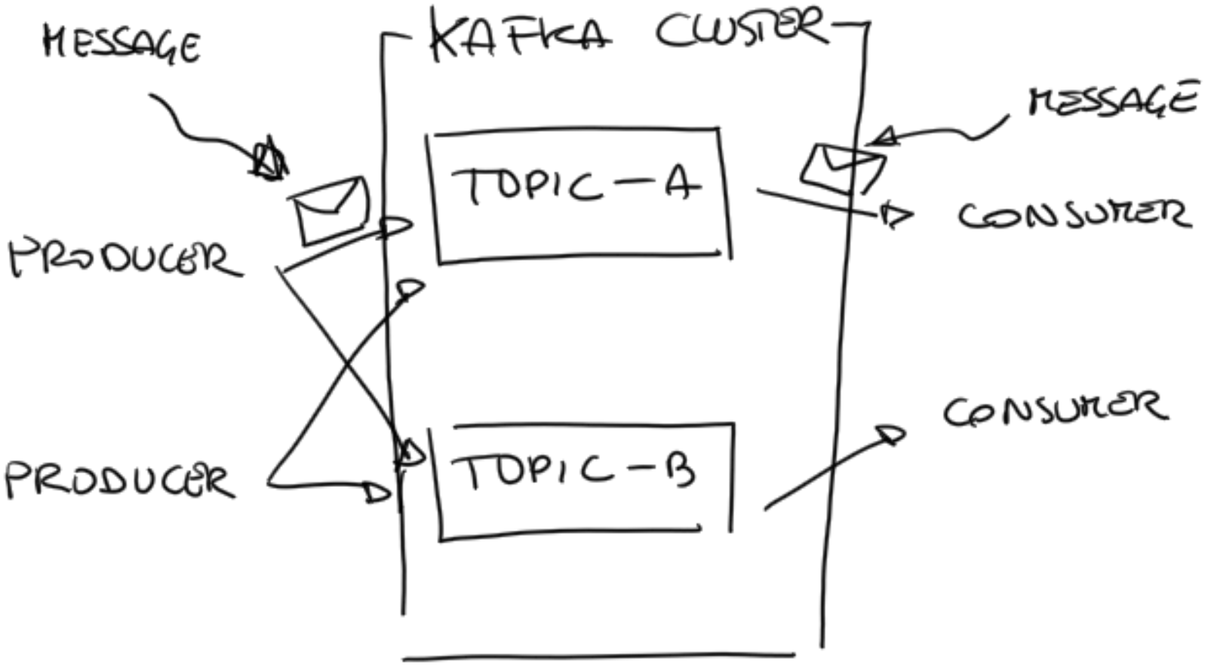


Kafka Conceptual View



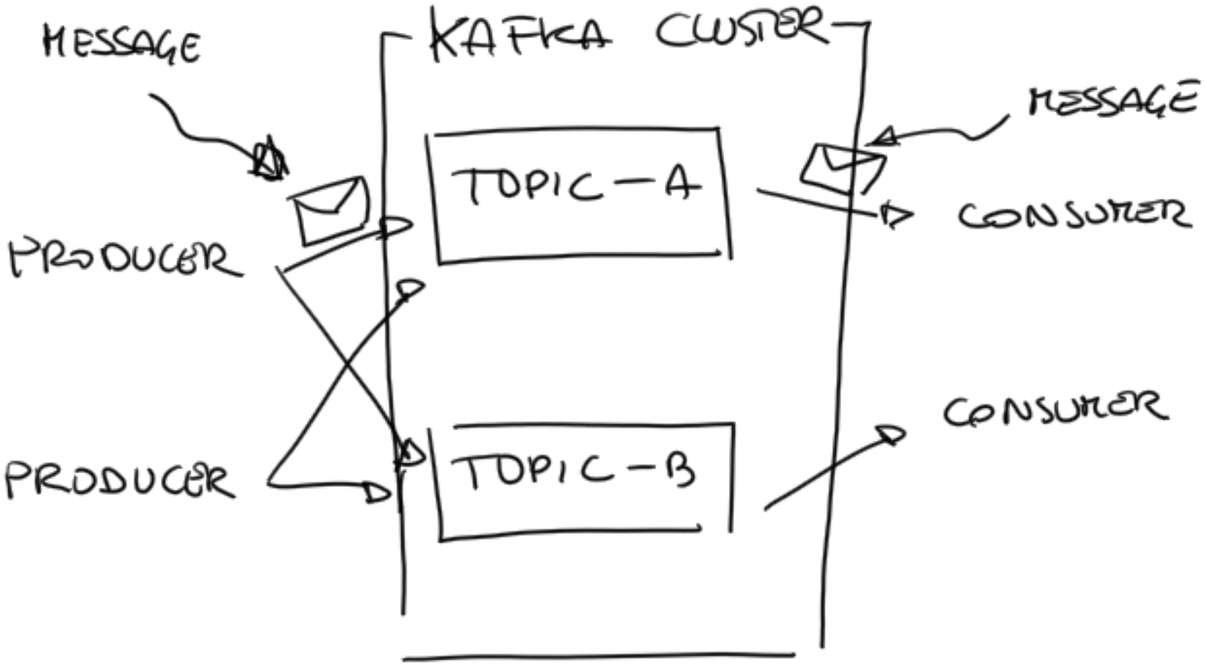
Kafka Conceptual View

- **Messages**, the basic unit in Kafka, are organized in **Topics**



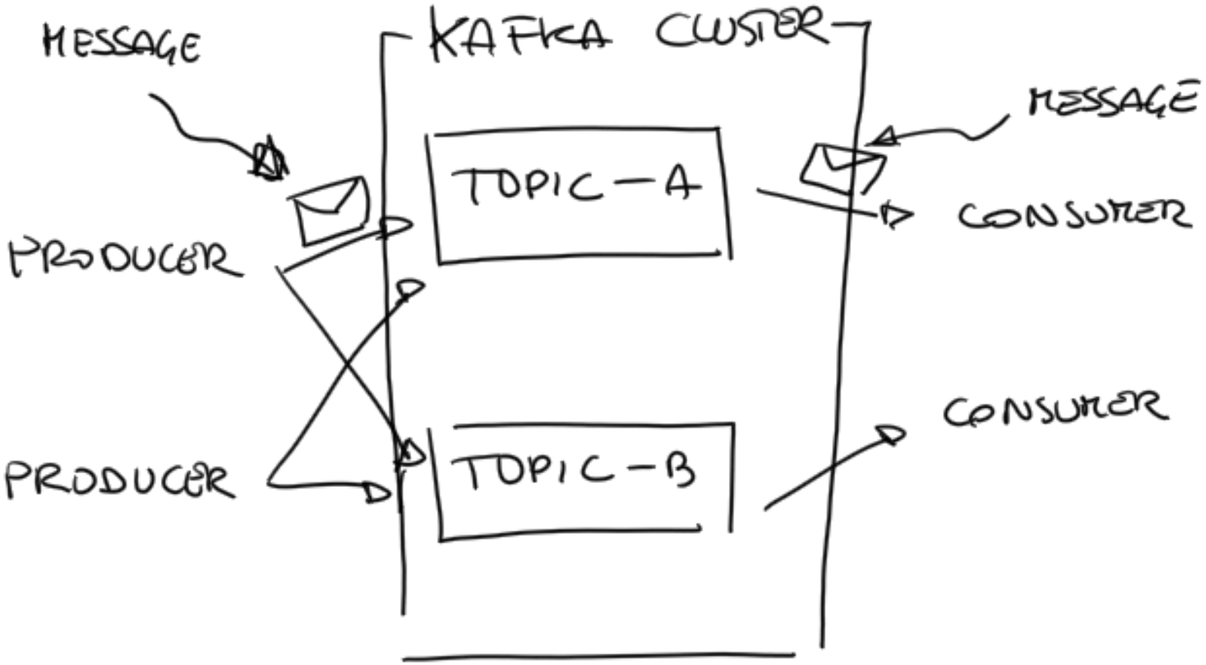
Kafka Conceptual View

- **Messages**, the basic unit in Kafka, are organized in **Topics**
- **Producers** write messages topics

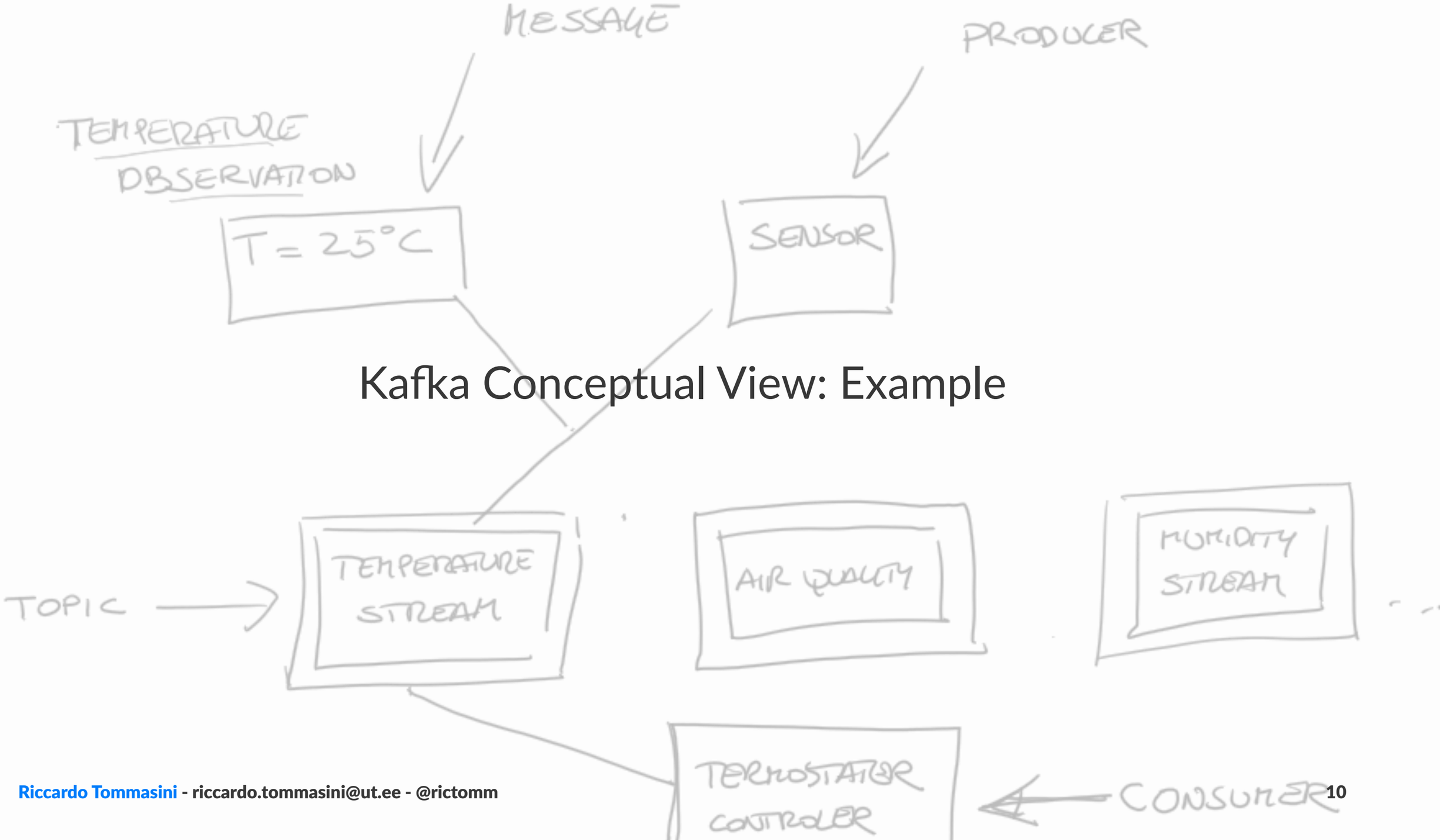


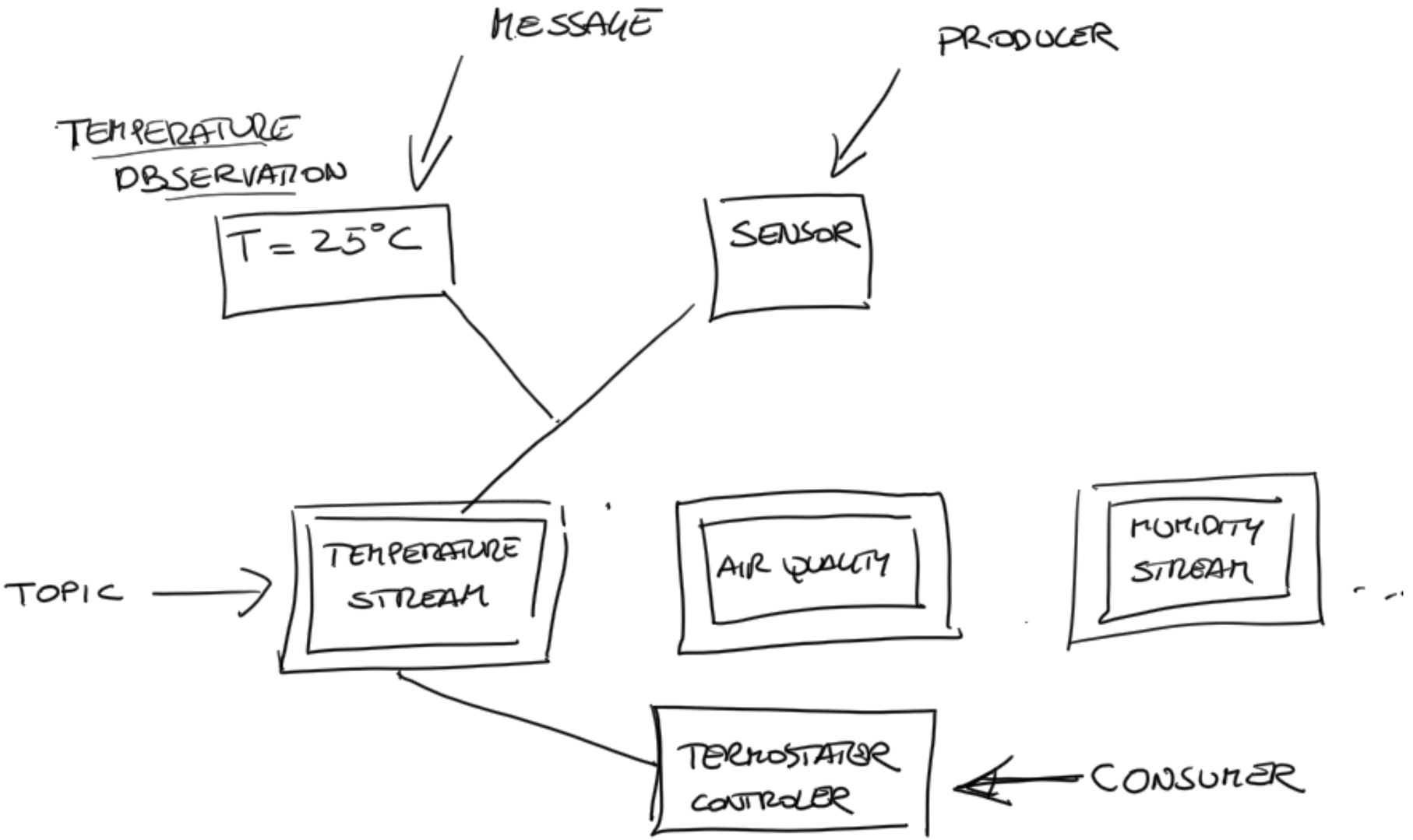
Kafka Conceptual View

- **Messages**, the basic unit in Kafka, are organized in **Topics**
- **Producers** write messages topics
- **Consumers** read messages by from topics

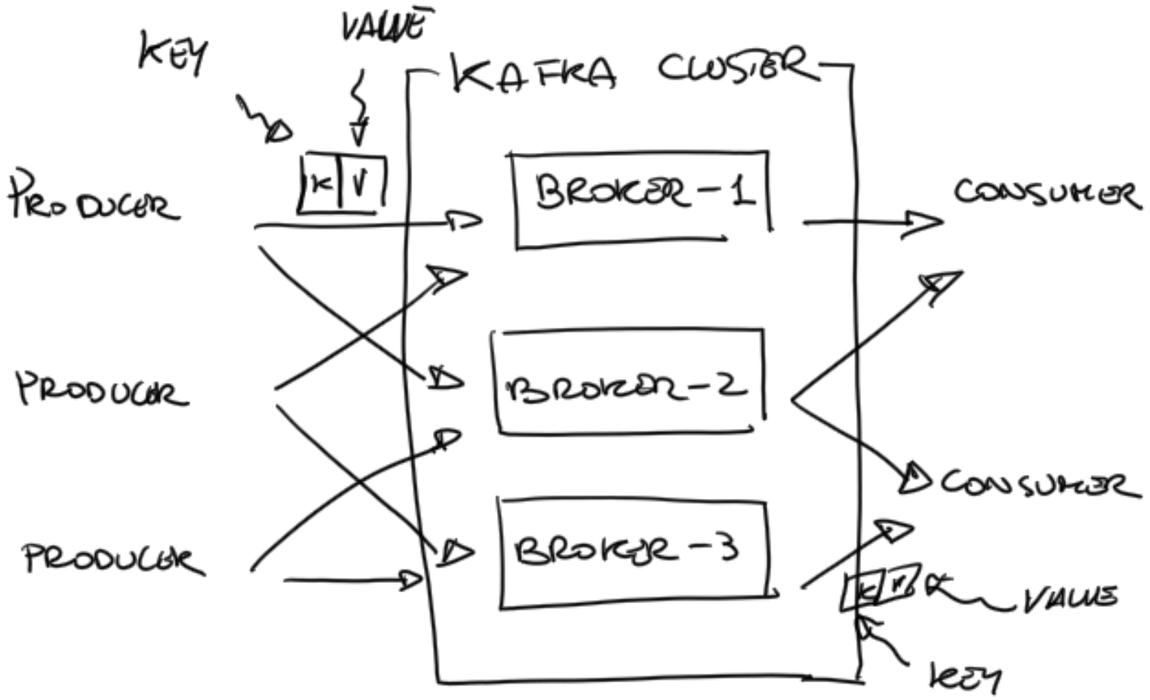


Kafka Conceptual View: Example

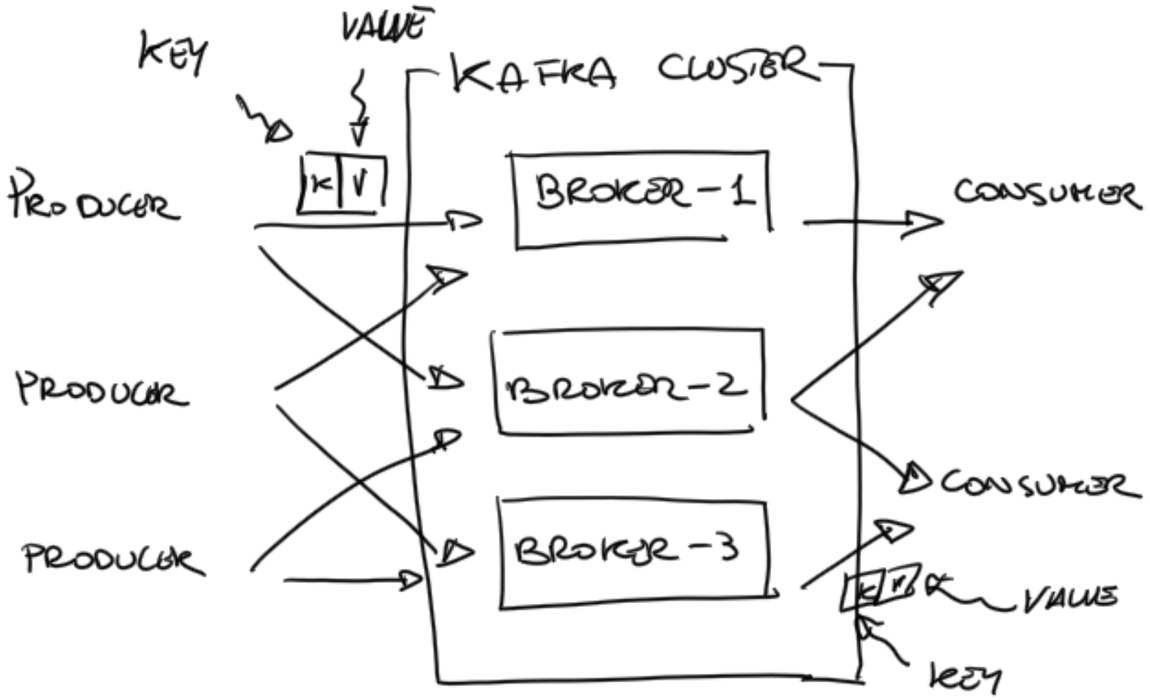



MESSAGE
PRODUCER
TEMPERATURE OBSERVATION
T = 25°C
SENSOR
TOPIC
TEMPERATURE STREAM
AIR QUALITY
HUMIDITY STREAM
TERMOSTATIC CONTROLLER
CONSUMER

Kafka Logical View

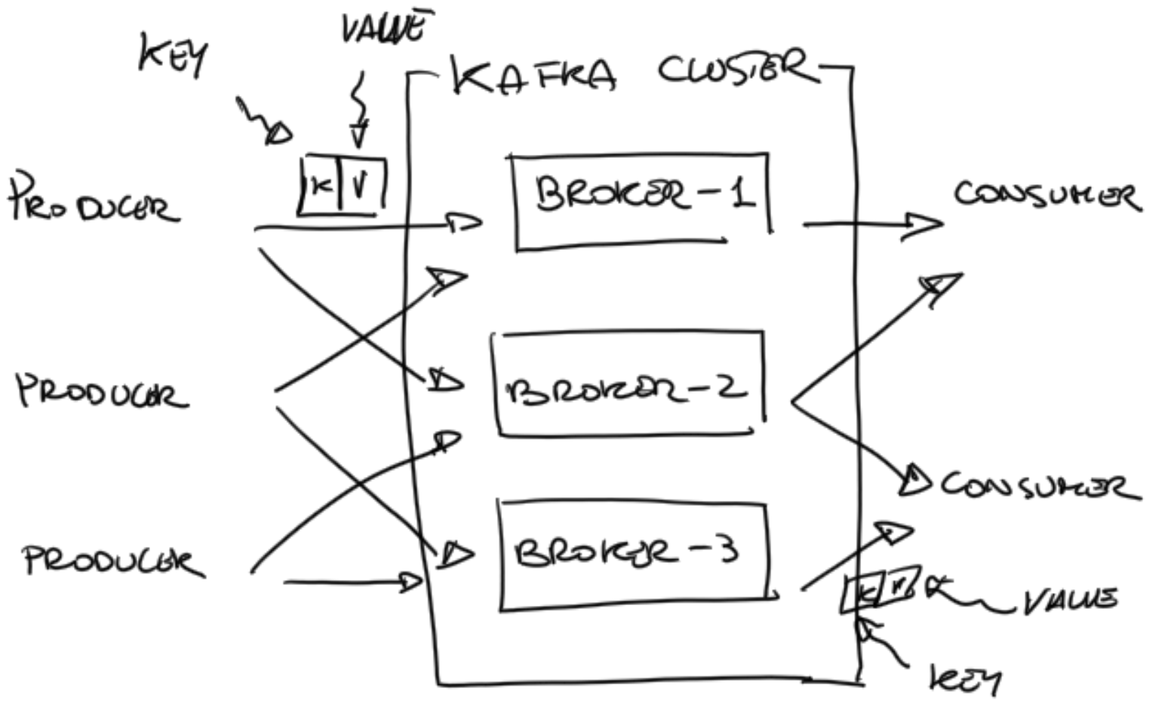


Kafka Logical View



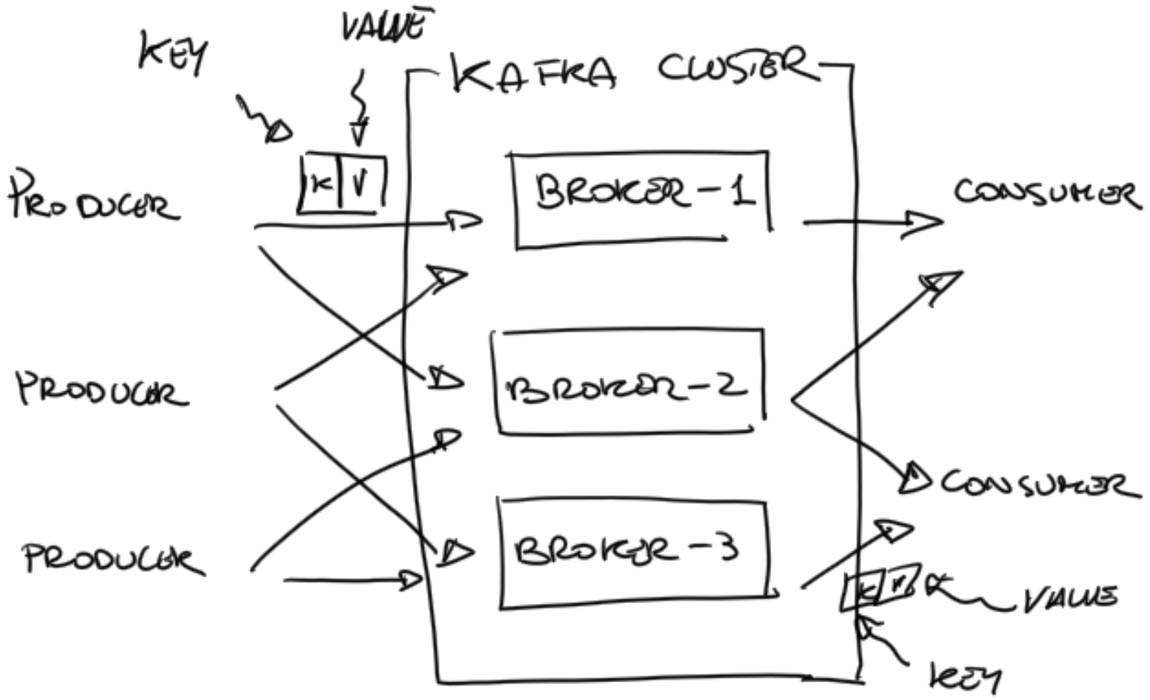
- **Messages** are key-value pairs

Kafka Logical View



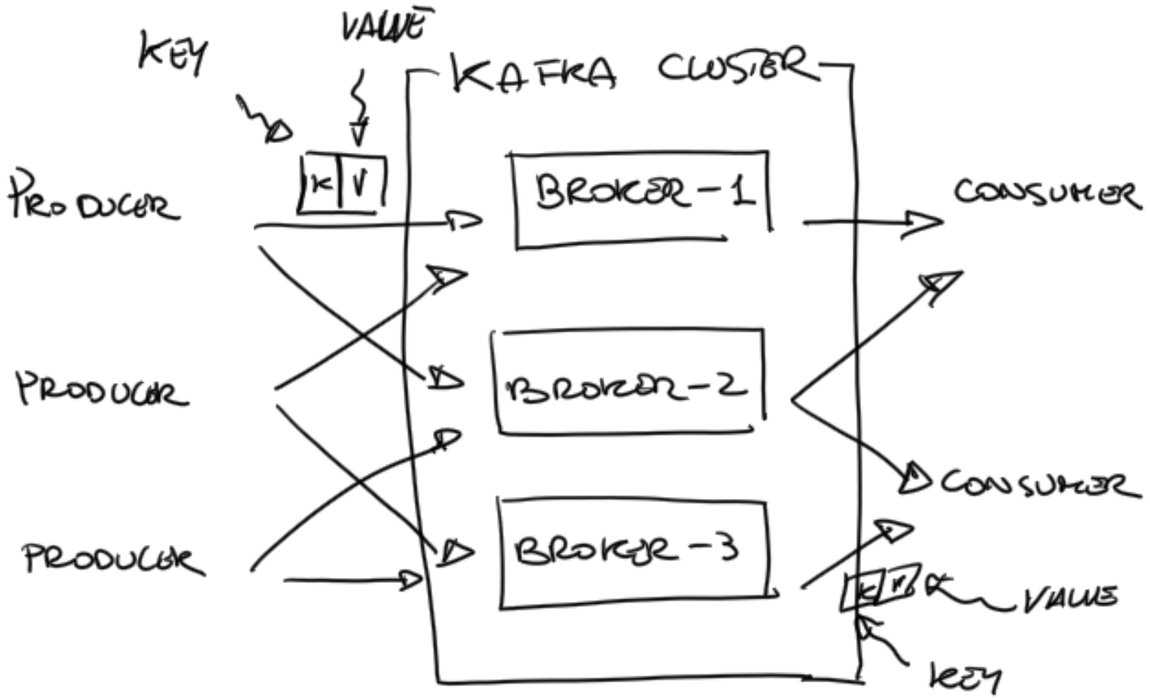
- **Messages** are key-value pairs
- **Brokers** are the main component inside the Kafka Cluster.

Kafka Logical View



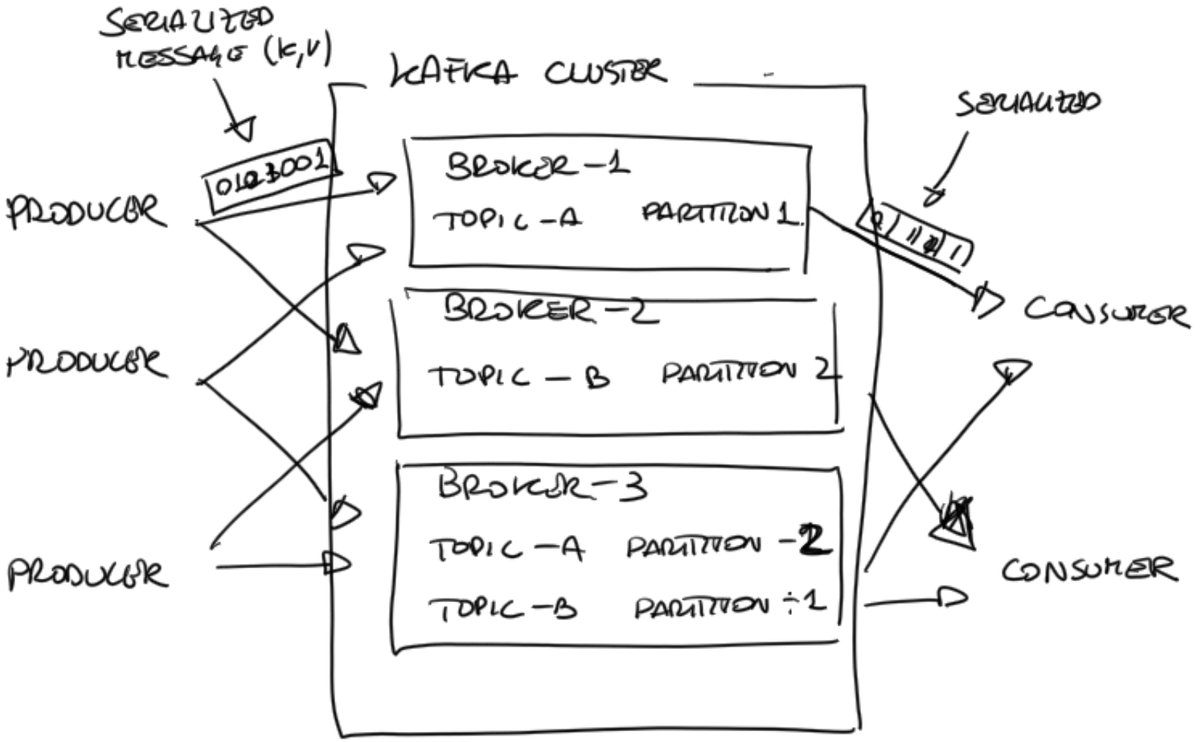
- **Messages** are key-value pairs
- **Brokers** are the main component inside the Kafka Cluster.
- **Producers** write messages to a certain broker

Kafka Logical View



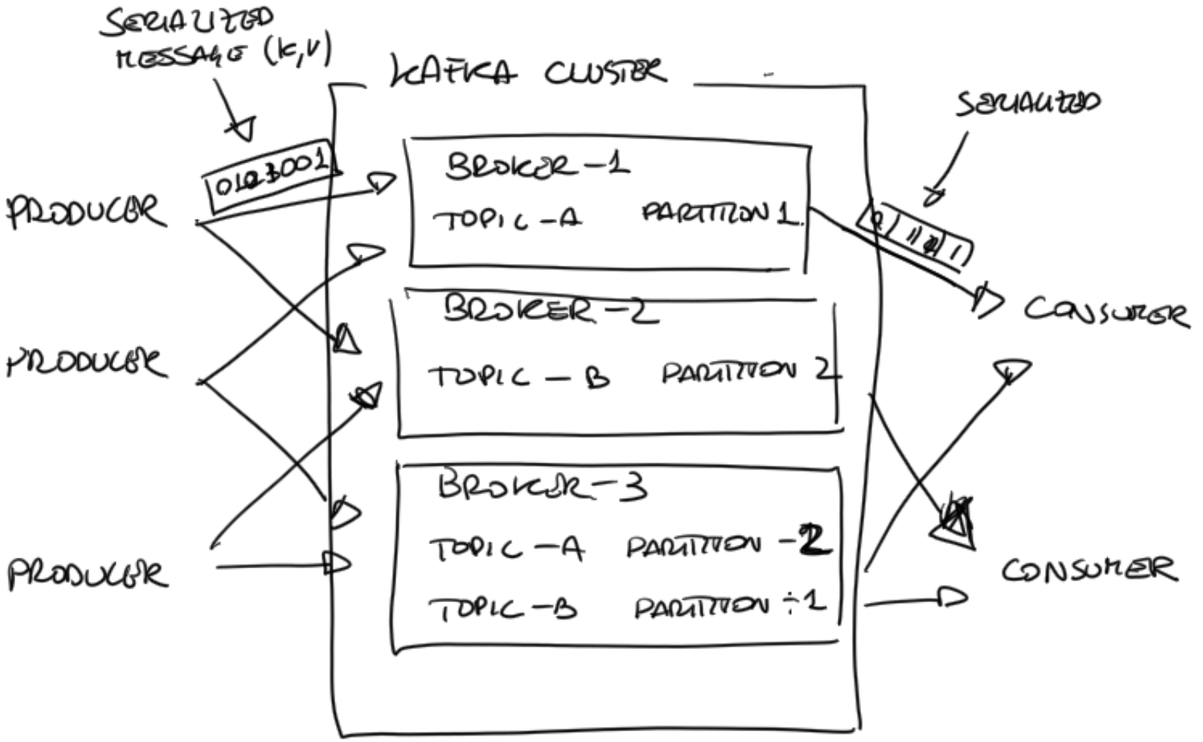
- **Messages** are key-value pairs
- **Brokers** are the main component inside the Kafka Cluster.
- **Producers** write messages to a certain broker
- **Consumers** read messages by from a certain broker

Kafka Physical View



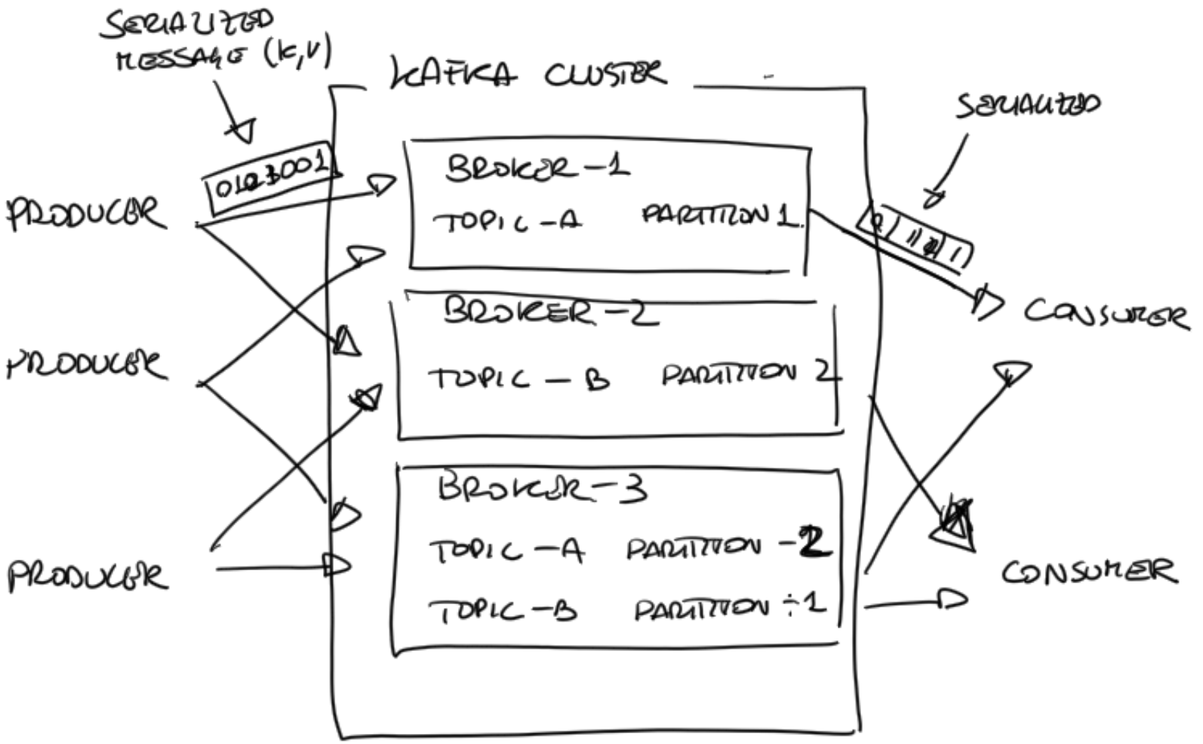
Kafka Physical View

- **Topics** are partitioned across brokers using the message **Key**.



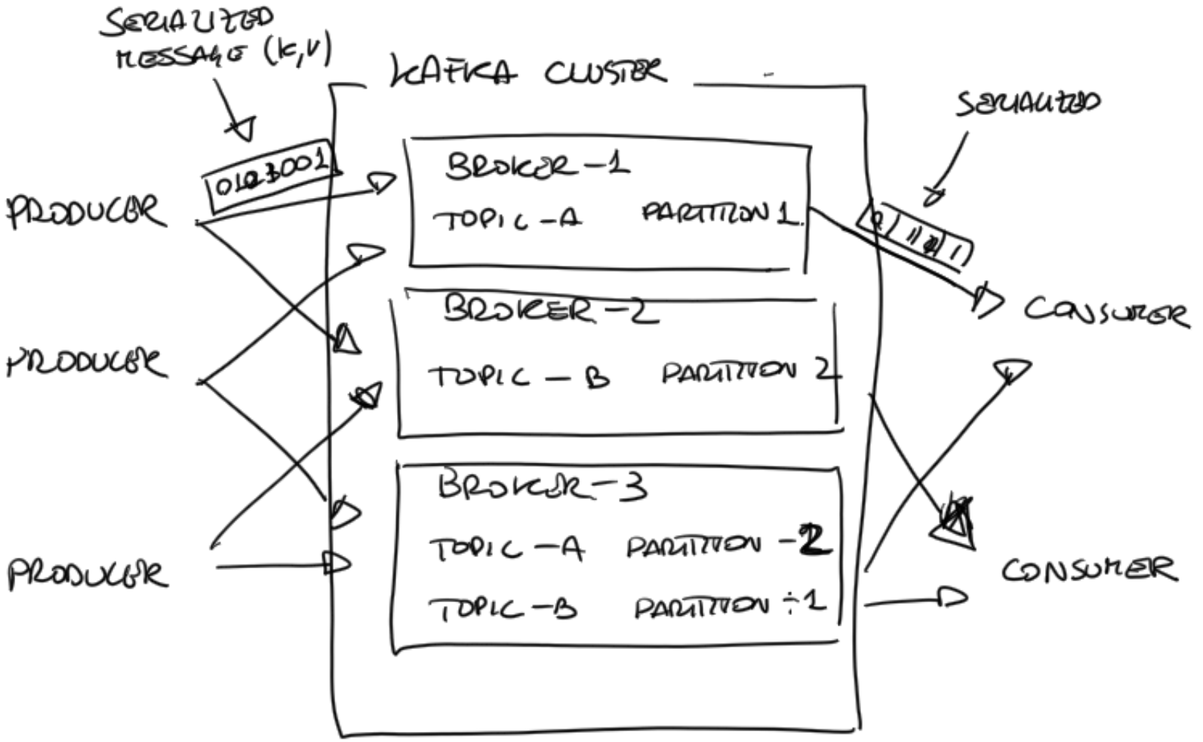
Kafka Physical View

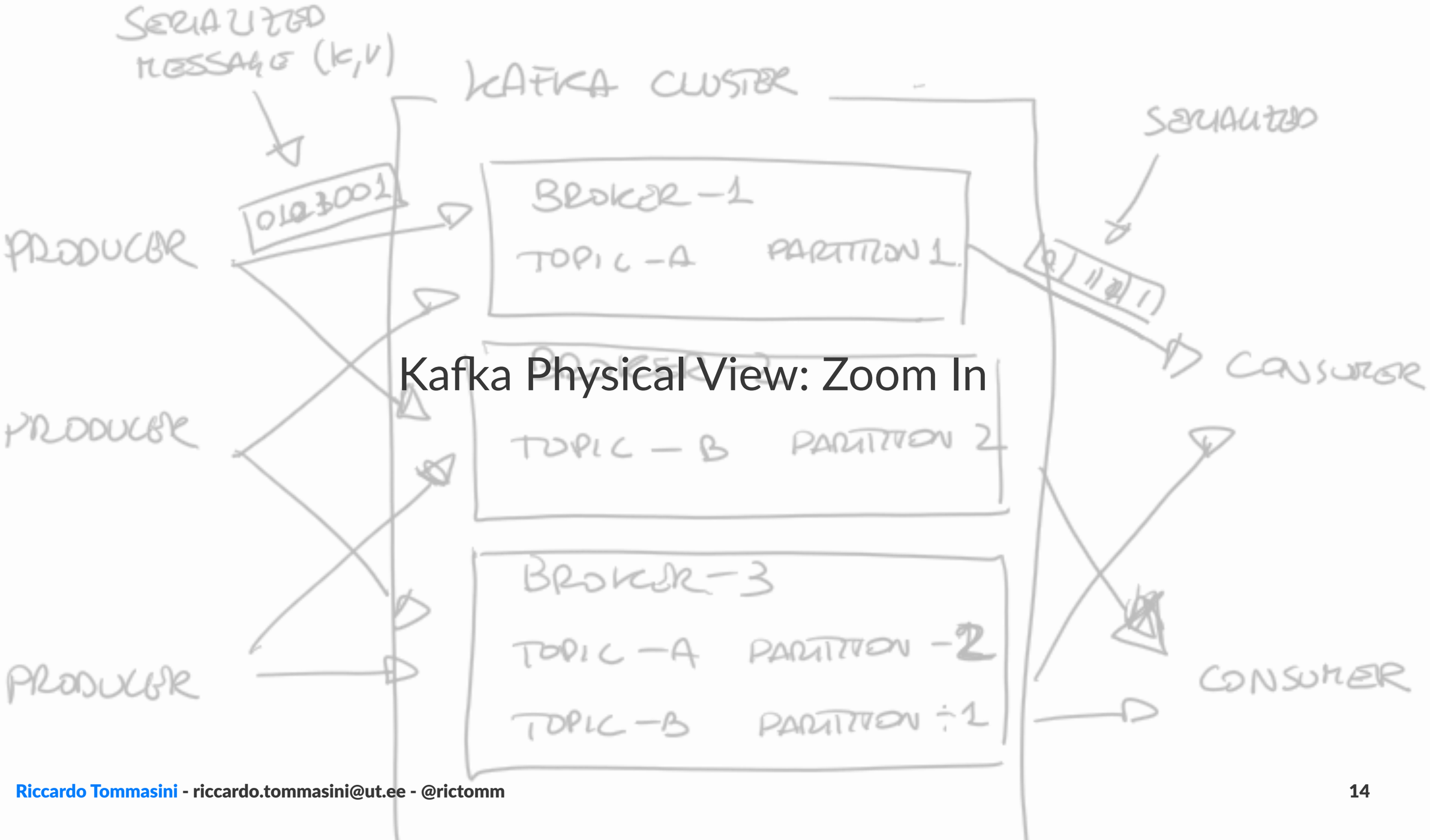
- **Topics** are partitioned across brokers using the message **Key**.
- Typically, **Producers** has the message key to determine the partition. Also they serialize the message



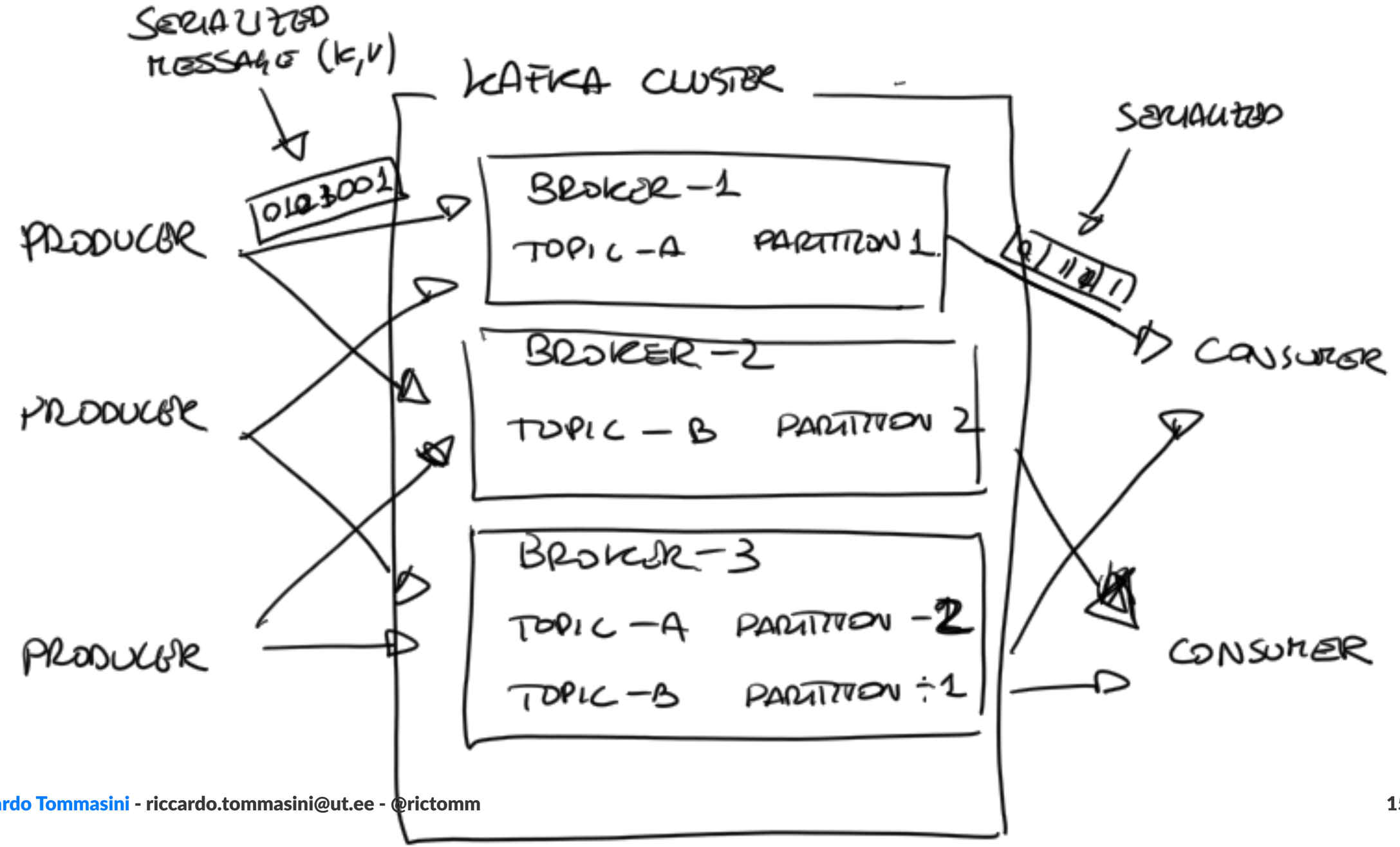
Kafka Physical View

- **Topics** are partitioned across brokers using the message **Key**.
- Typically, **Producers** has the message key to determine the partition. Also they serialize the message
- **Consumers** read messages by from brokers and de-serialize them





Kafka Physical View: Zoom In


SERIALIZED MESSAGE (K,V)
KAFKA CLUSTER
SERIALIZED
PRODUCER
PRODUCER
PRODUCER
BROKER-1
TOPIC-A
PARTITION 1
BROKER-2
TOPIC-B
PARTITION 2
BROKER-3
TOPIC-A
PARTITION -2
TOPIC-B
PARTITION ÷1
CONSUMER
CONSUMER

Topics Partitions

Producers shard data over a set of Partitions

Topics Partitions

Producers shard data over a set of Partitions

- Each Partition contains a subset of the Topic's messages

Topics Partitions

Producers shard data over a set of Partitions

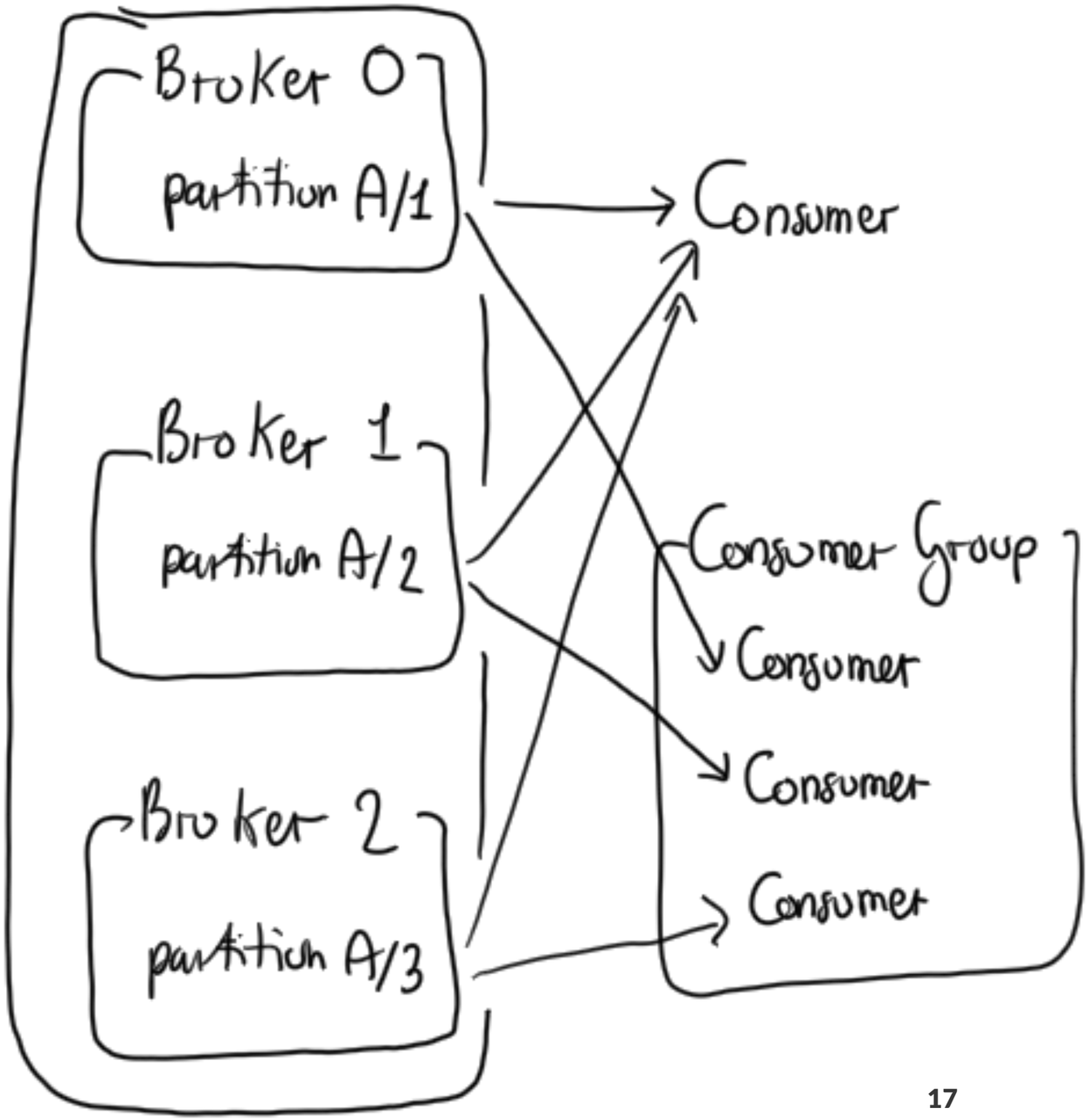
- Each Partition contains a subset of the Topic's messages
- Typically, the message key is used to determine which Partition a message is assigned to

Topics Partitions

Producers shard data over a set of Partitions

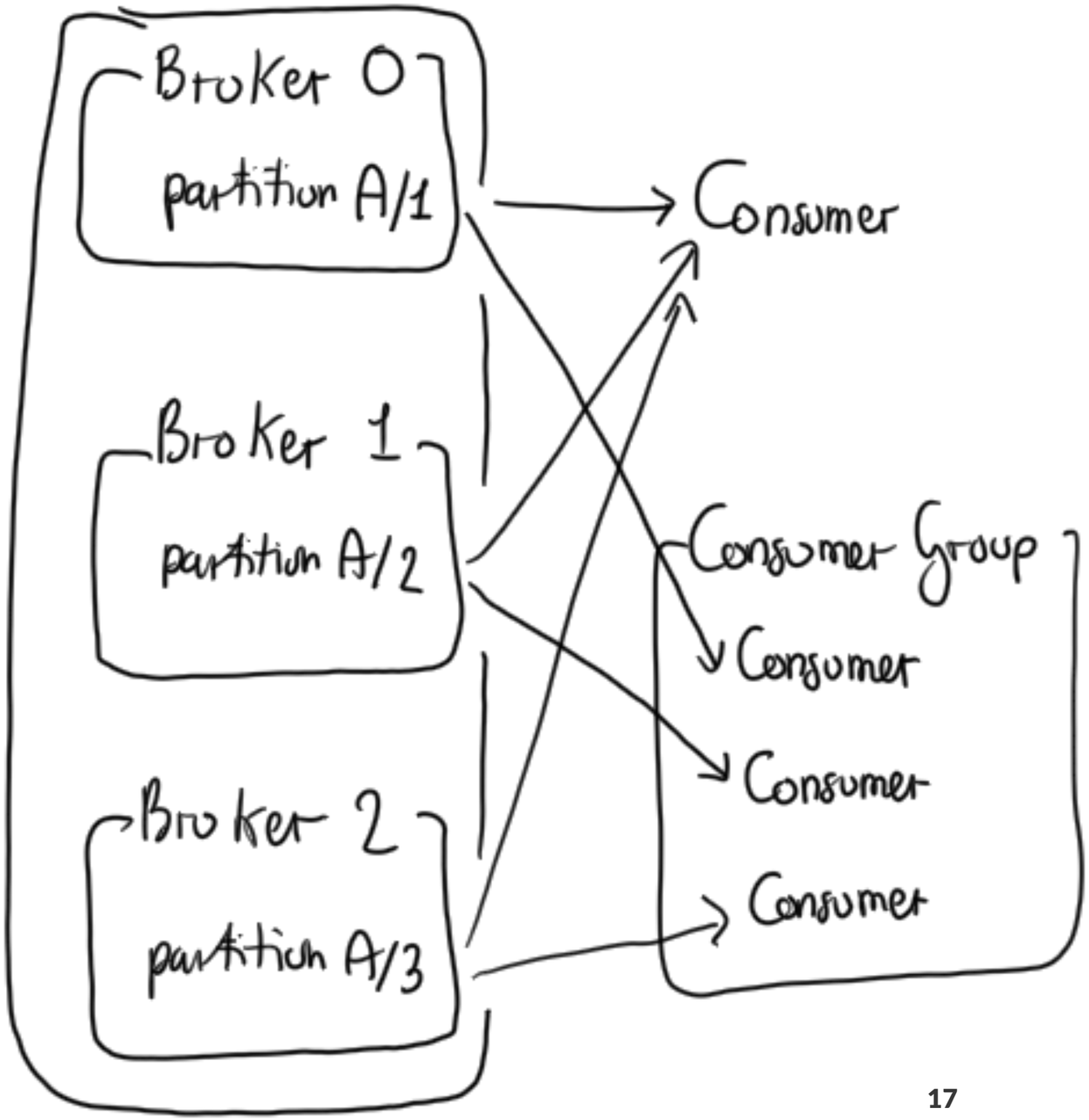
- Each Partition contains a subset of the Topic's messages
- Typically, the message key is used to determine which Partition a message is assigned to
- Each Partition is an ordered, immutable log of messages

Topics Partitions and Distributed Consumption



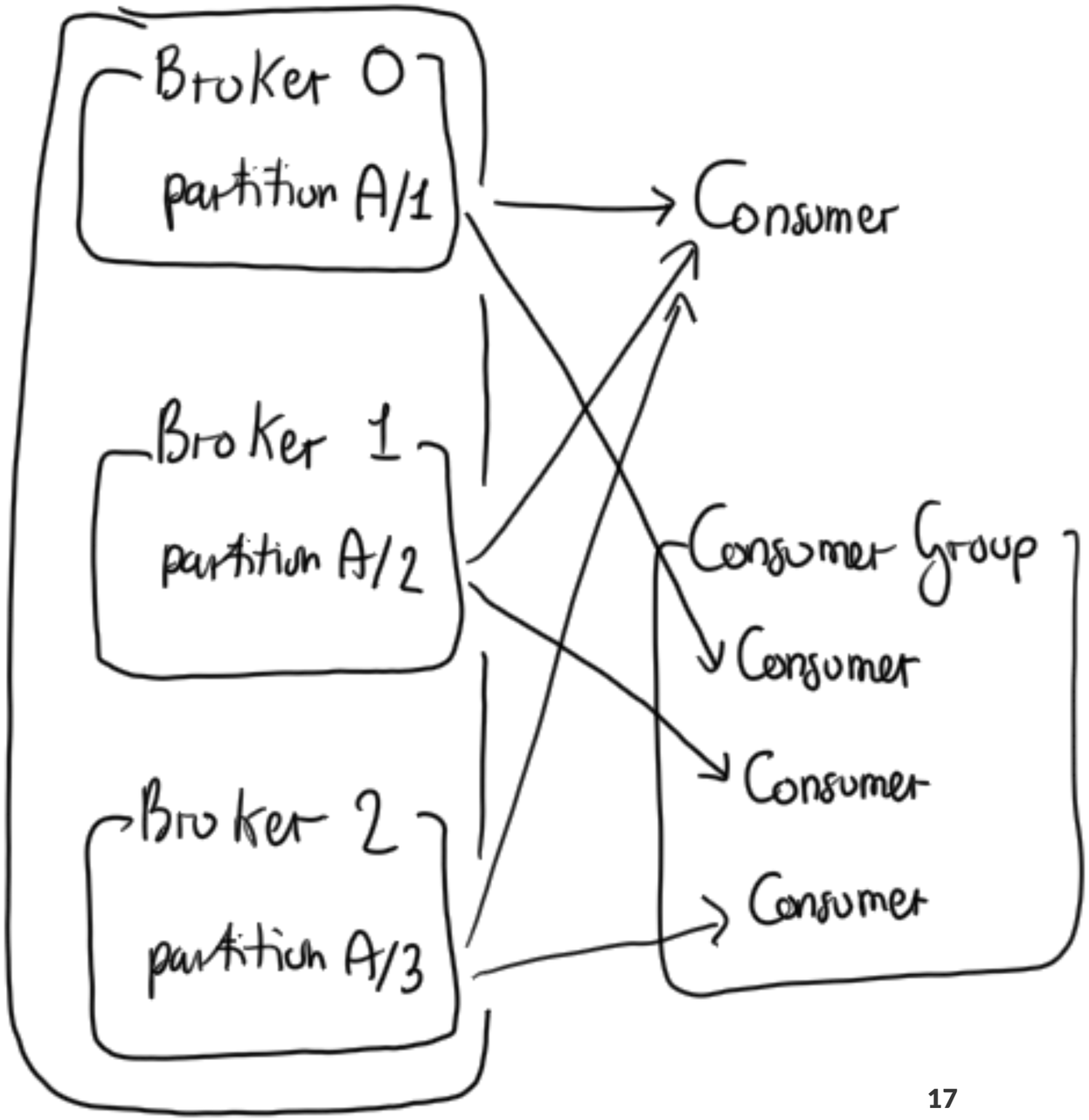
Topics Partitions and Distributed Consumption

- Different Consumers can read data from the same Topic



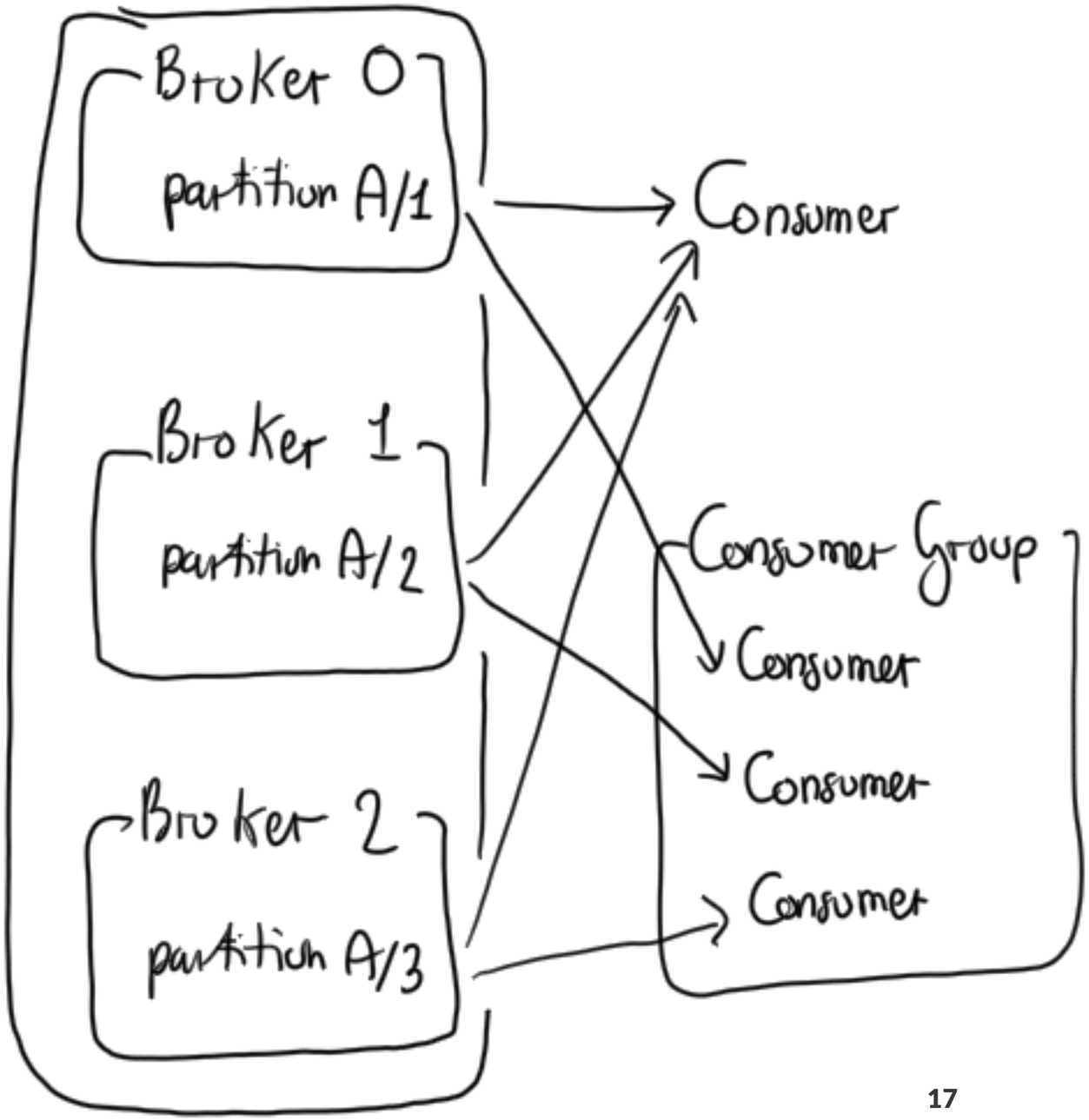
Topics Partitions and Distributed Consumption

- Different Consumers can read data from the same Topic
  - By default, each Consumer will receive all the messages in the Topic



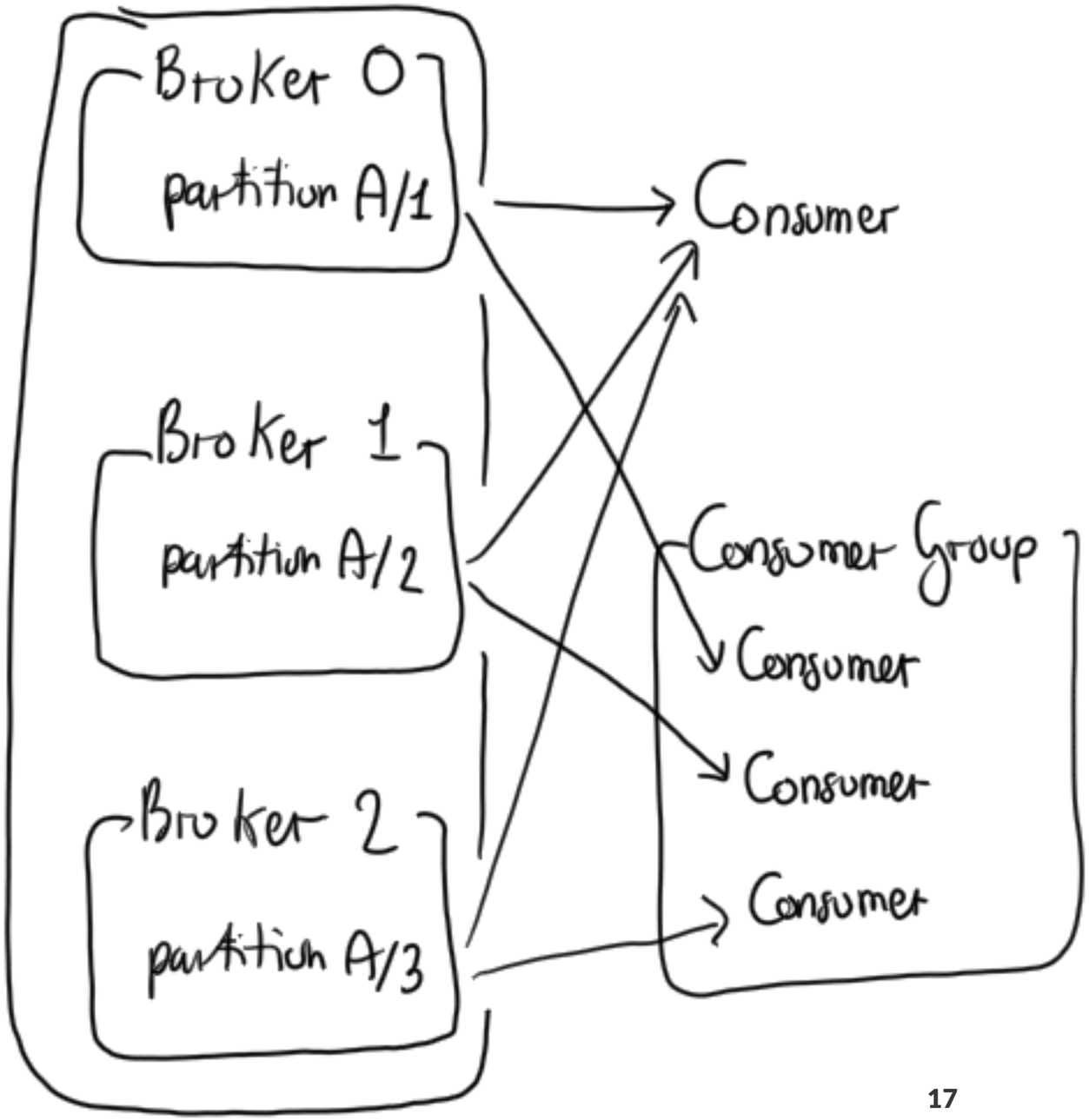
Topics Partitions and Distributed Consumption

- Different Consumers can read data from the same Topic
  - By default, each Consumer will receive all the messages in the Topic
- Multiple Consumers can be combined into a Consumer Group



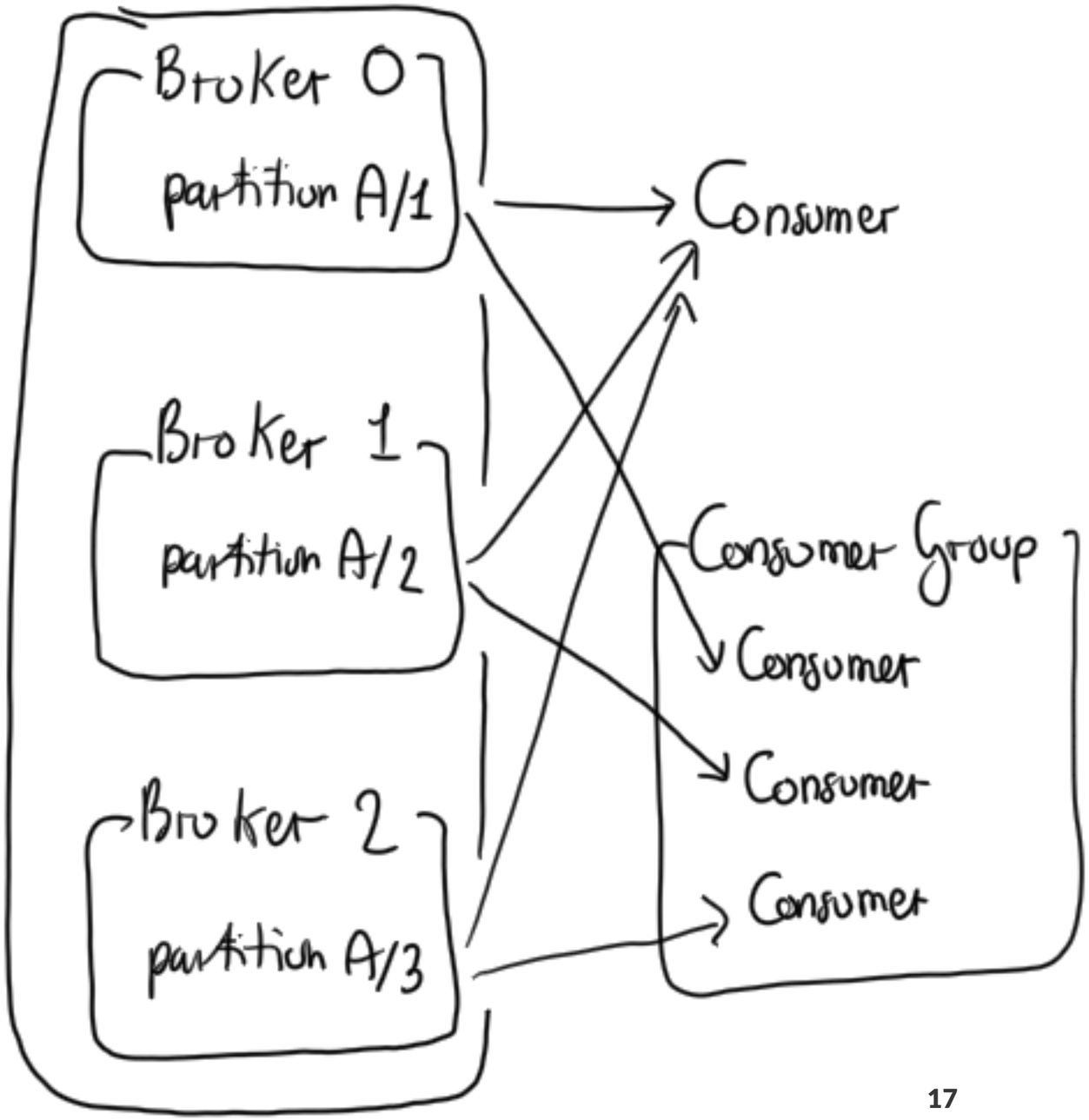
Topics Partitions and Distributed Consumption

- Different Consumers can read data from the same Topic
  - By default, each Consumer will receive all the messages in the Topic
- Multiple Consumers can be combined into a Consumer Group
  - Consumer Groups provide scaling capabilities

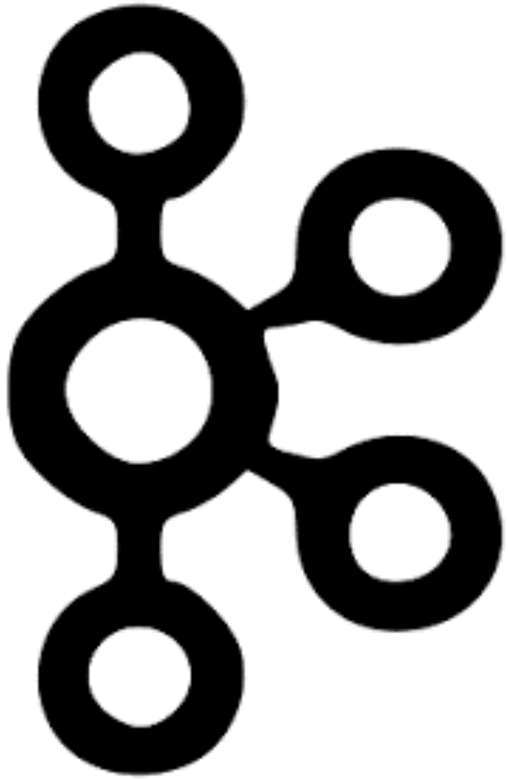


Topics Partitions and Distributed Consumption

- Different Consumers can read data from the same Topic
  - By default, each Consumer will receive all the messages in the Topic
- Multiple Consumers can be combined into a Consumer Group
  - Consumer Groups provide scaling capabilities
  - Each Consumer is assigned a subset of Partitions for consumption



Apache Kafka[^2]

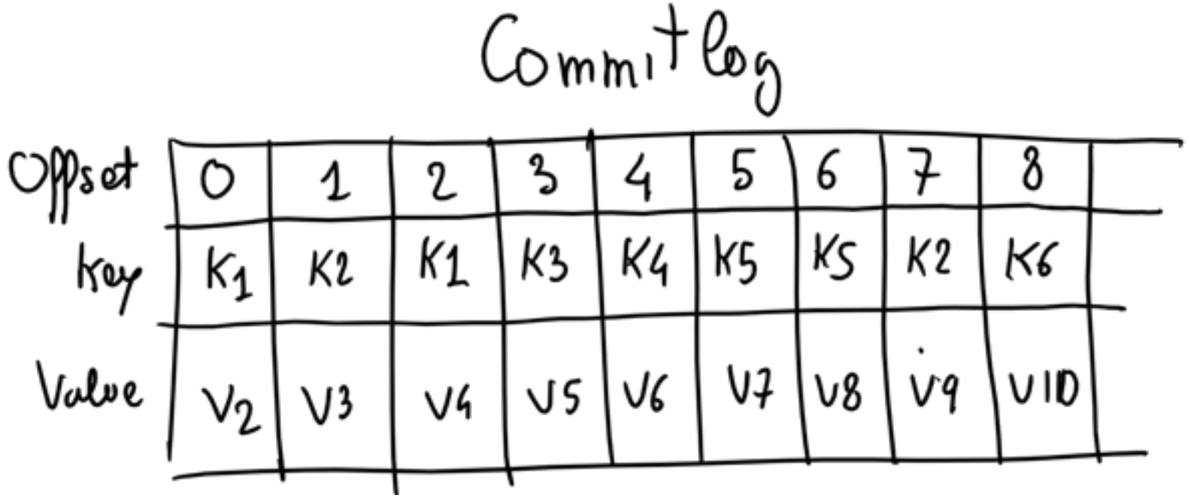


Internals

Messages and Metadata

Messages are Key-Value pairs and there is not restriction on what each of them can be.

Additionally, messages are enriched with metadata:

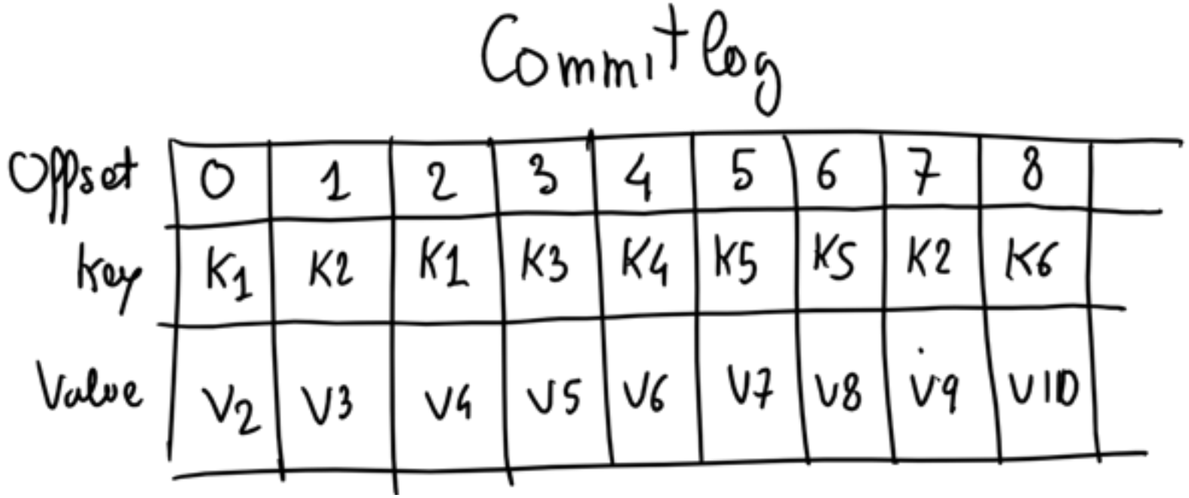


Messages and Metadata

Messages are Key-Value pairs and there is not restriction on what each of them can be.

Additionally, messages are enriched with metadata:

- Offset

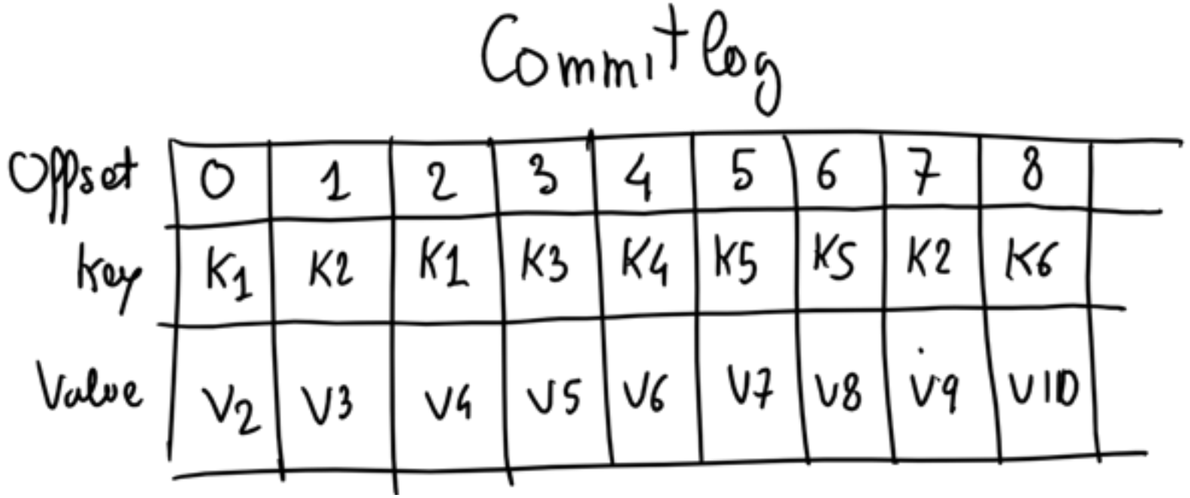


Messages and Metadata

Messages are Key-Value pairs and there is not restriction on what each of them can be.

Additionally, messages are enriched with metadata:

- Offset
- Timestamp

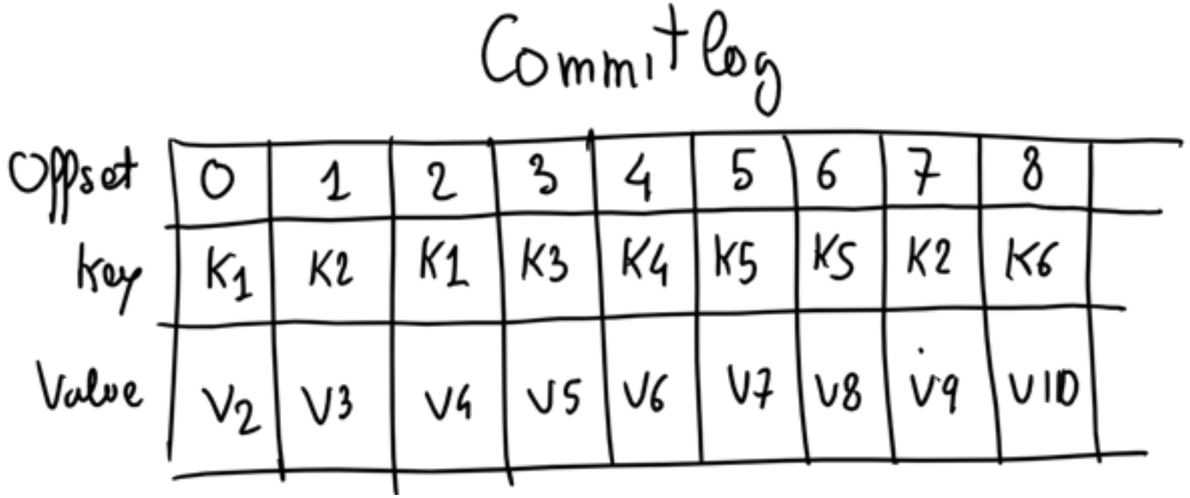


Messages and Metadata

Messages are Key-Value pairs and there is not restriction on what each of them can be.

Additionally, messages are enriched with metadata:

- Offset
- Timestamp
- Compression type

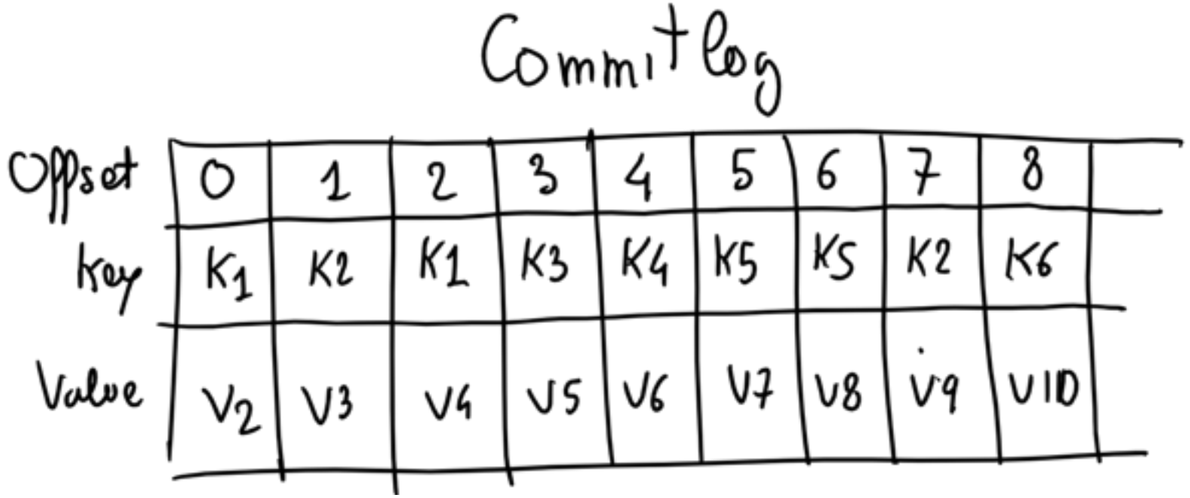


Messages and Metadata

Messages are Key-Value pairs and there is not restriction on what each of them can be.

Additionally, messages are enriched with metadata:

- Offset
- Timestamp
- Compression type
- Magic byte

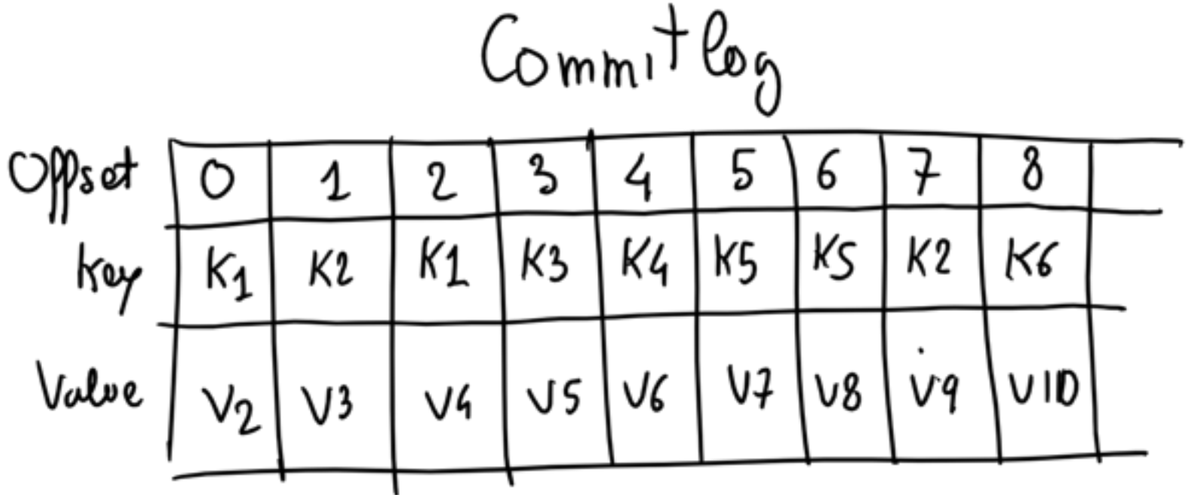


Messages and Metadata

Messages are Key-Value pairs and there is not restriction on what each of them can be.

Additionally, messages are enriched with metadata:

- Offset
- Timestamp
- Compression type
- Magic byte
- Optional message headers API

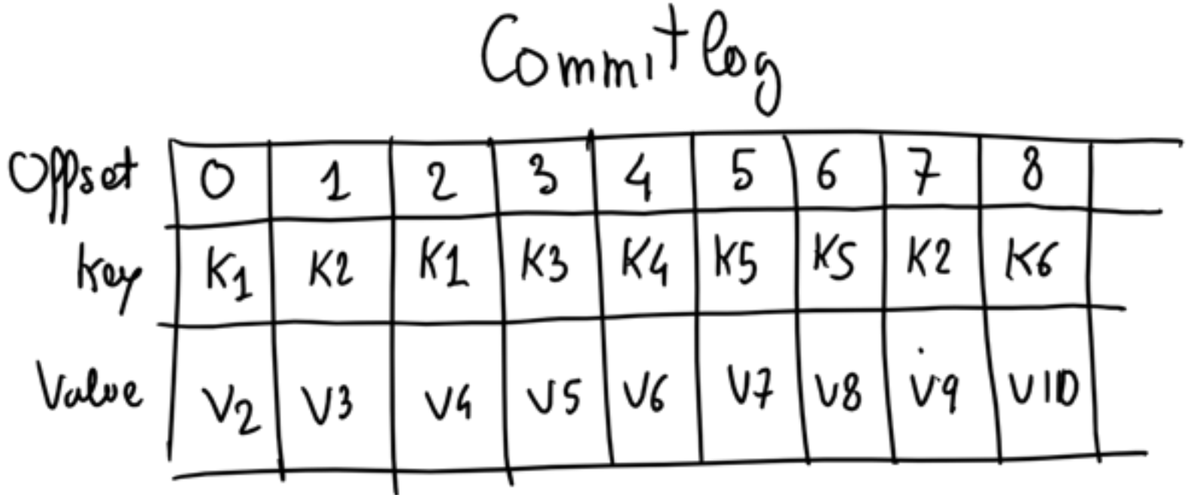


Messages and Metadata

Messages are Key-Value pairs and there is not restriction on what each of them can be.

Additionally, messages are enriched with metadata:

- Offset
- Timestamp
- Compression type
- Magic byte
- Optional message headers API
- Application teams can add custom key-value paired metadata to messages

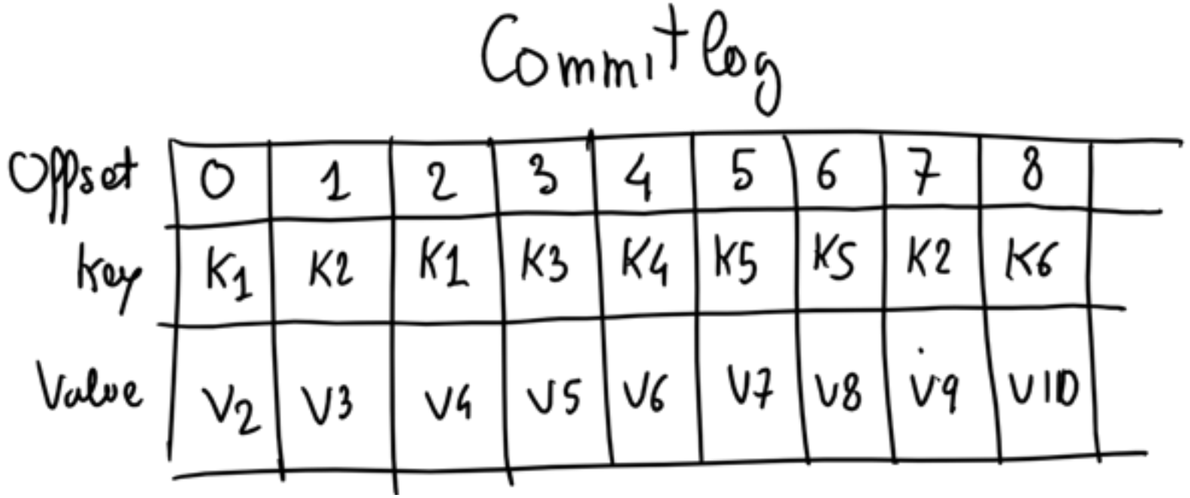


Messages and Metadata

Messages are Key-Value pairs and there is not restriction on what each of them can be.

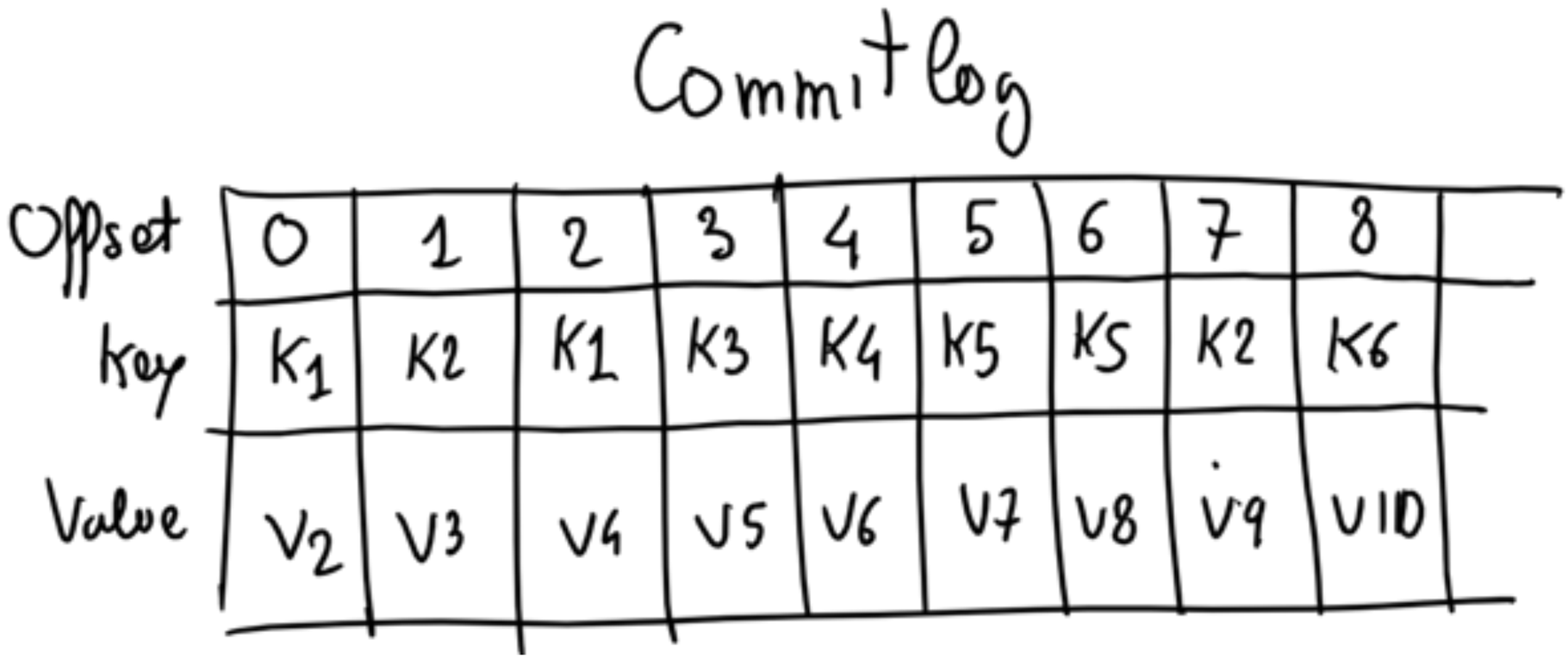
Additionally, messages are enriched with metadata:

- Offset
- Timestamp
- Compression type
- Magic byte
- Optional message headers API
- Application teams can add custom key-value paired metadata to messages
- Additional fields to support batching, exactly once semantics, replication protocol



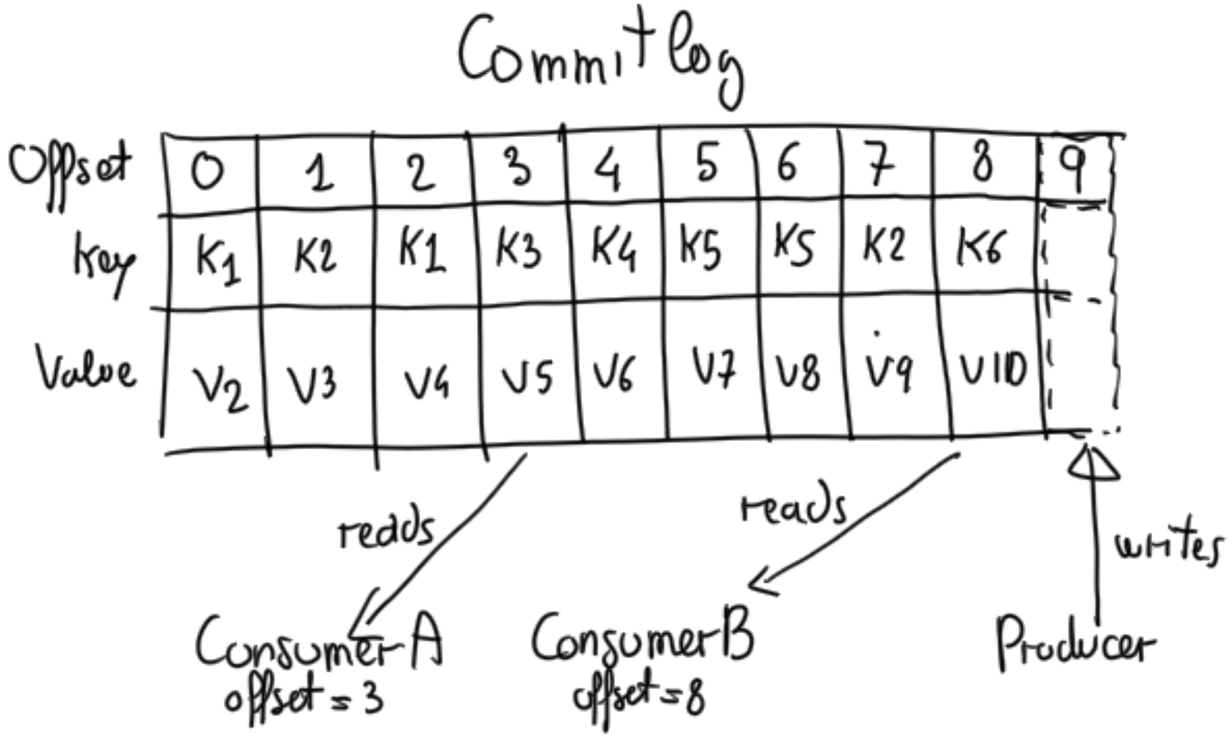
Topics Partitions: Physical View

Each Partition is stored on the Broker's disk as one or more log files Each message in the log is identified by its offset number



Topics Partitions: Physical View

Messages are always appended. Consumers can consume from different offset. Brokers are single thread to guarantee consistency



Topics Partitions: Load Balancing

Producers use a partition strategy to assign each message a partition

You can customize the partition strategy, but!

Topics Partitions: Load Balancing

Producers use a partition strategy to assign each message a partition

- To ensure load balancing across the Brokers

You can customize the partition strategy, but!

Topics Partitions: Load Balancing

Producers use a partition strategy to assign each message a partition

- To ensure load balancing across the Brokers
- To allow user-specified key

You can customize the partition strategy, but!

Topics Partitions: Load Balancing

Producers use a partition strategy to assign each message a partition

- To ensure load balancing across the Brokers
- To allow user-specified key

You can customize the partition strategy, but!

Topics Partitions: Load Balancing

Producers use a partition strategy to assign each message a partition

- To ensure load balancing across the Brokers
- To allow user-specified key

You can customize the partition strategy, but!

- it must ensure load balancing across the Brokers too, i.e., hash(key) % number_of_partitions

Topics Partitions: Load Balancing

Producers use a partition strategy to assign each message a partition

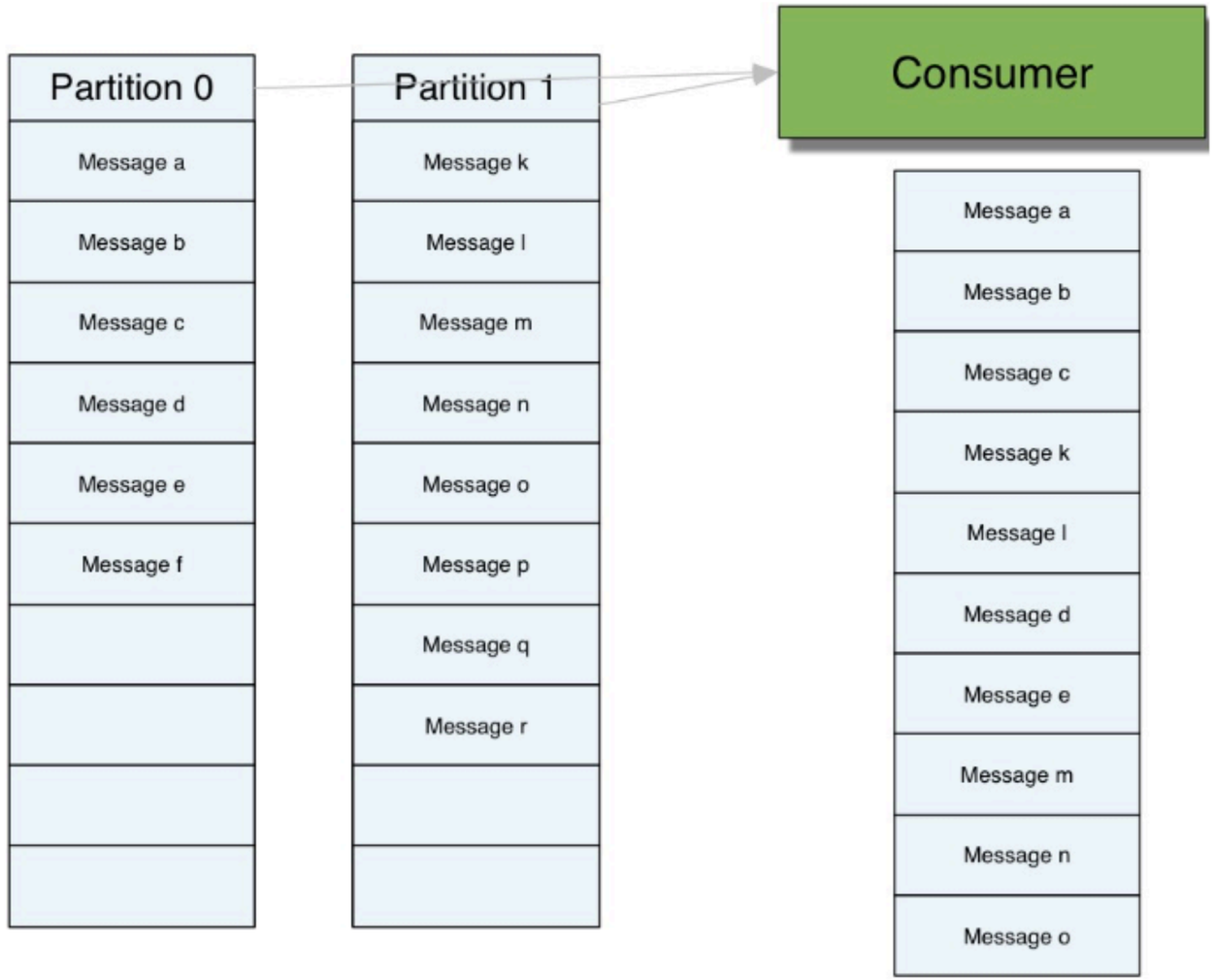
- To ensure load balancing across the Brokers
- To allow user-specified key

You can customize the partition strategy, but!

- it must ensure load balancing across the Brokers too, i.e., hash(key) % number_of_partitions
- if key is not specified, messages are sent to Partitions on a round-robin basis

Important: About Ordering

If there are multiple Partitions, you will not get total ordering across all messages when reading data



Log Retention

Log Retention

- Duration default: messages will be retained for seven days

Log Retention

- Duration default: messages will be retained for seven days
- Duration is configurable per Broker by setting

Log Retention

- Duration default: messages will be retained for seven days
- Duration is configurable per Broker by setting
  - a time period

Log Retention

- Duration default: messages will be retained for seven days
- Duration is configurable per Broker by setting
  - a time period
  - a size limit

Log Retention

- Duration default: messages will be retained for seven days
- Duration is configurable per Broker by setting
  - a time period
  - a size limit
- Topic can override a Broker's retention policy

Log Retention

- Duration default: messages will be retained for seven days
- Duration is configurable per Broker by setting
  - a time period
  - a size limit
- Topic can override a Broker's retention policy
- When cleaning up a log

Log Retention

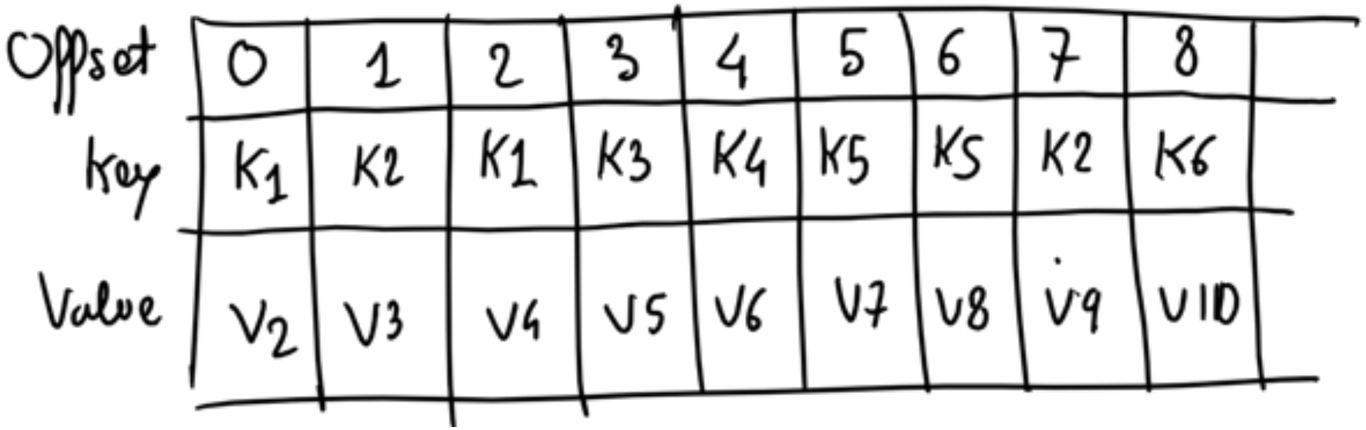
- Duration default: messages will be retained for seven days
- Duration is configurable per Broker by setting
  - a time period
  - a size limit
- Topic can override a Broker's retention policy
- When cleaning up a log
  - the default policy is delete

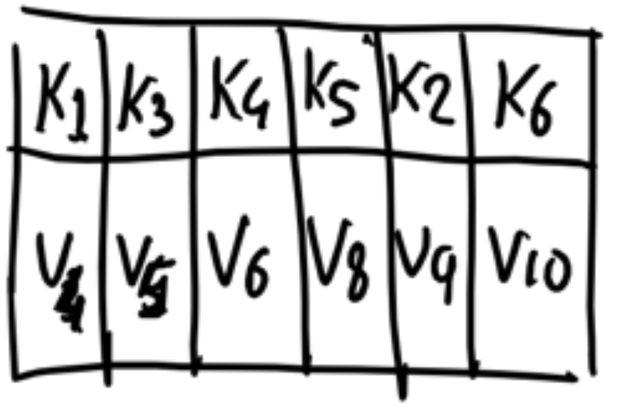
Log Retention

- Duration default: messages will be retained for seven days
- Duration is configurable per Broker by setting
  - a time period
  - a size limit
- Topic can override a Broker's retention policy
- When cleaning up a log
  - the default policy is delete
  - An alternate policy is compact

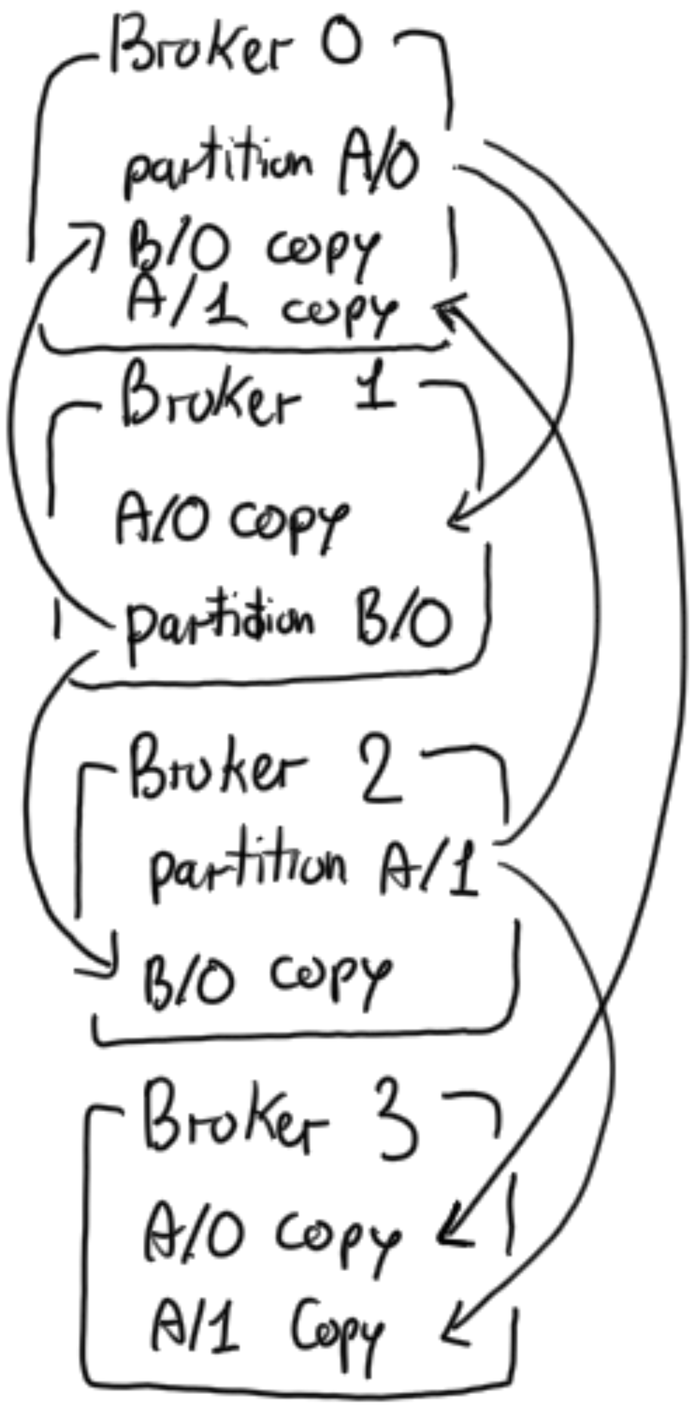
Log Compaction

A compacted log retains at least the last known message value for each key within the Partition
Before After



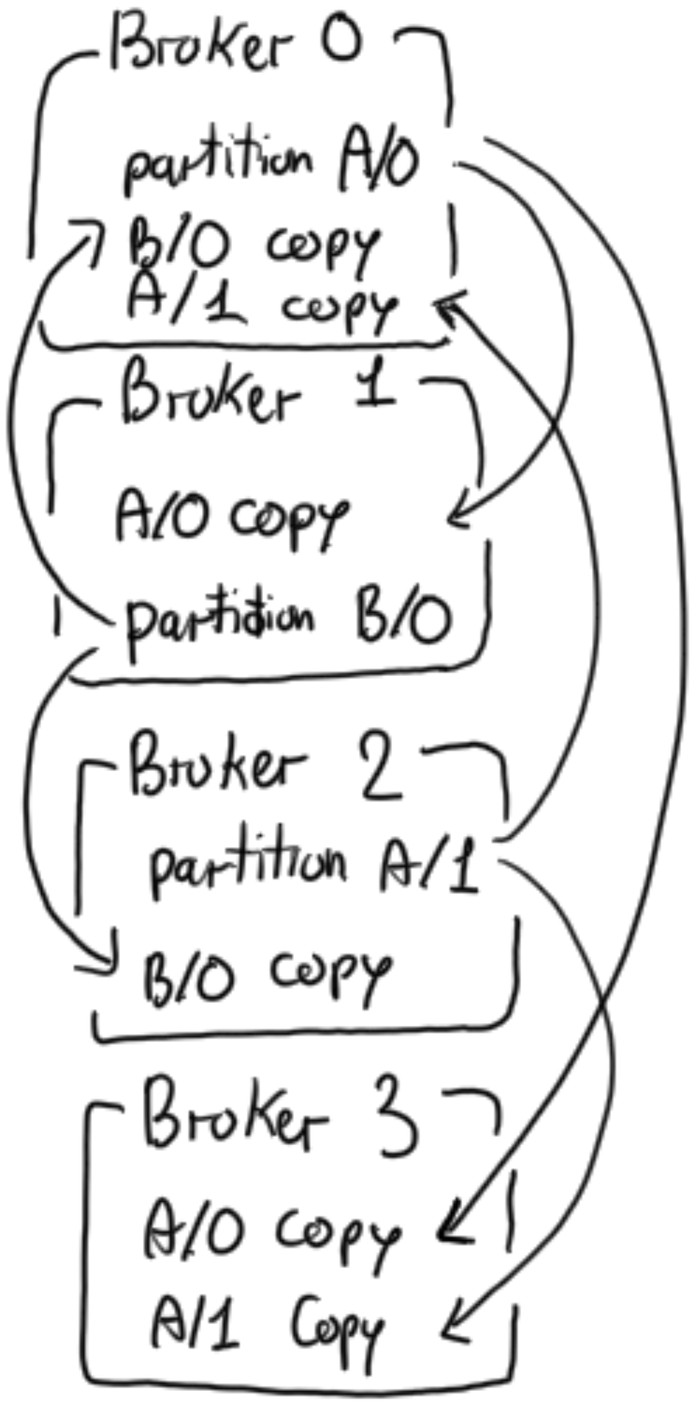


Fault Tolerance via a Replicated Log



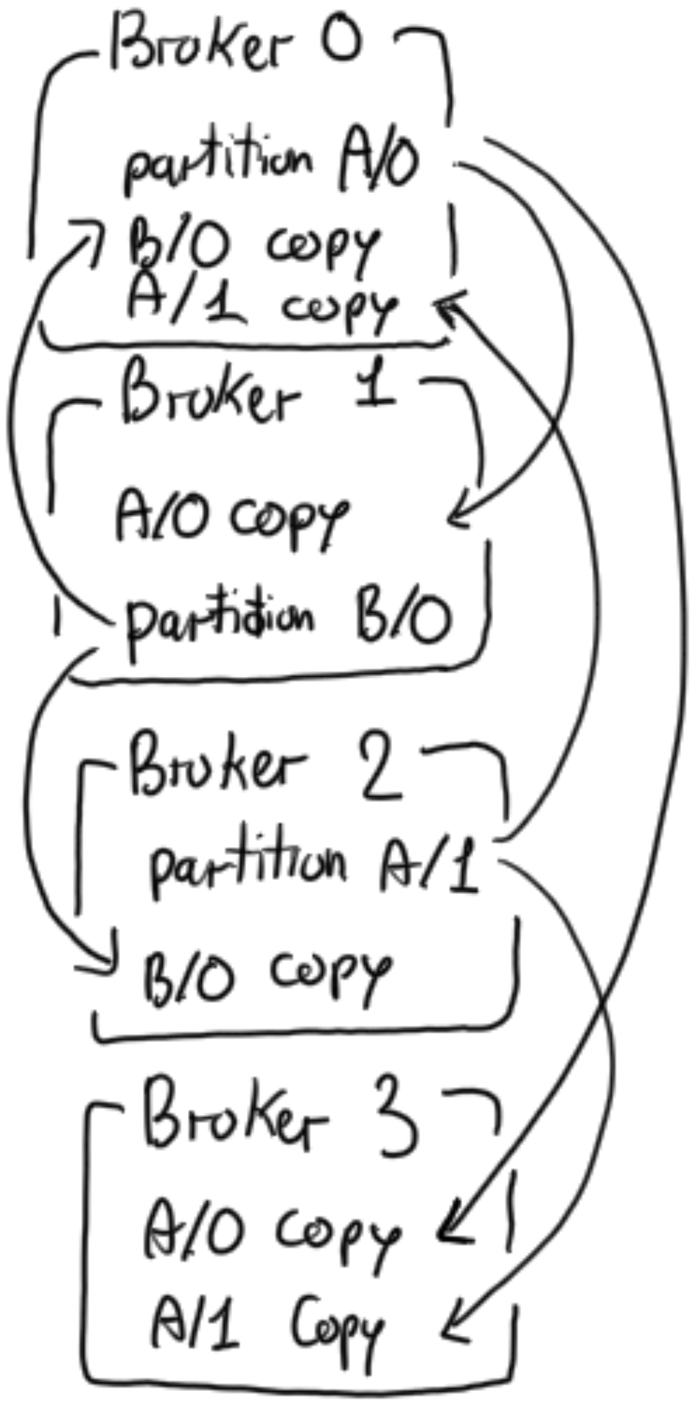
Fault Tolerance via a Replicated Log

- Kafka maintains replicas of each partition on other Brokers in the cluster



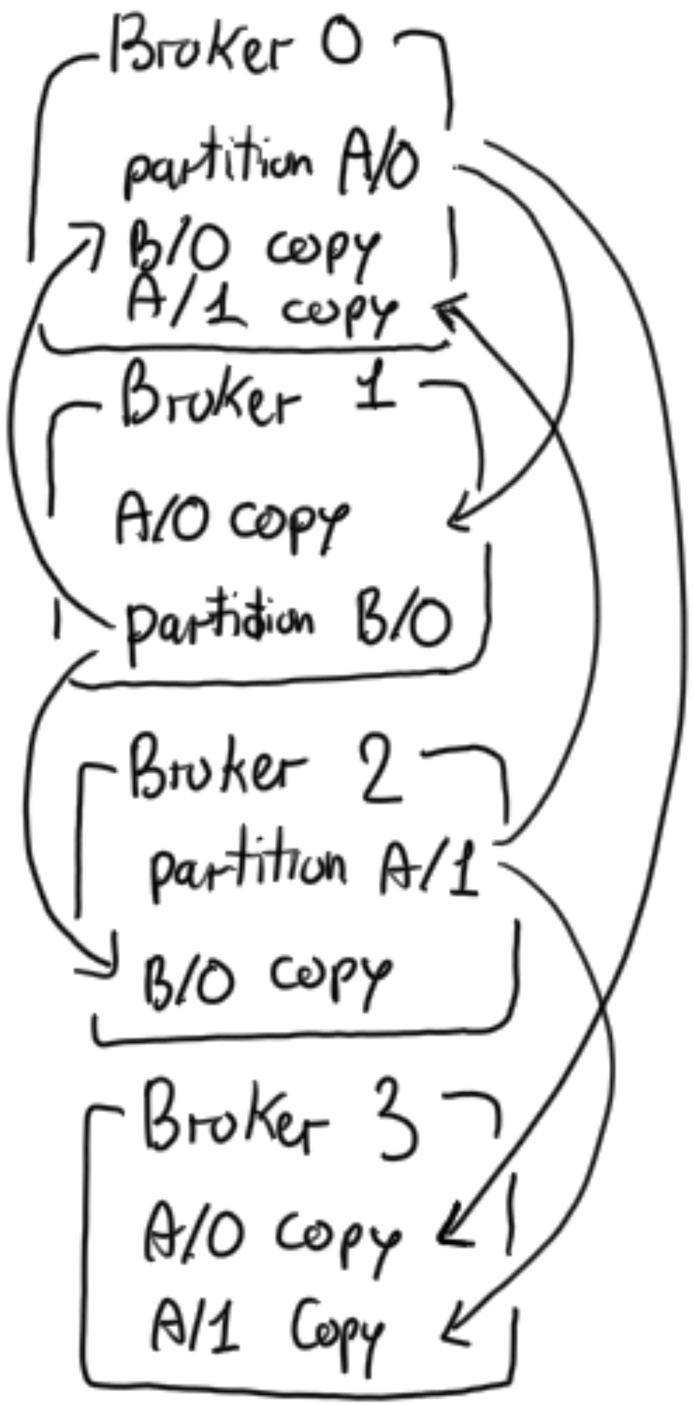
Fault Tolerance via a Replicated Log

- Kafka maintains replicas of each partition on other Brokers in the cluster
  - Number of replicas is configurable



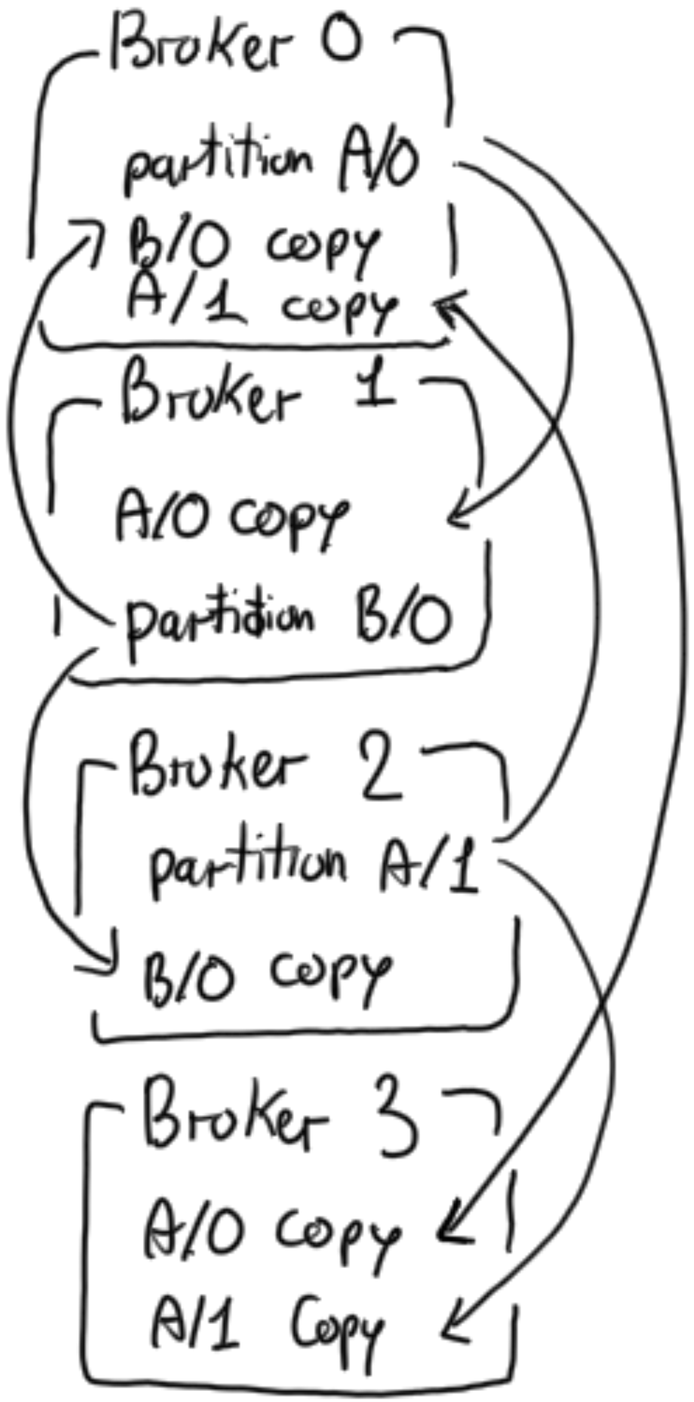
Fault Tolerance via a Replicated Log

- Kafka maintains replicas of each partition on other Brokers in the cluster
  - Number of replicas is configurable
- One Broker is the leader for that Partition



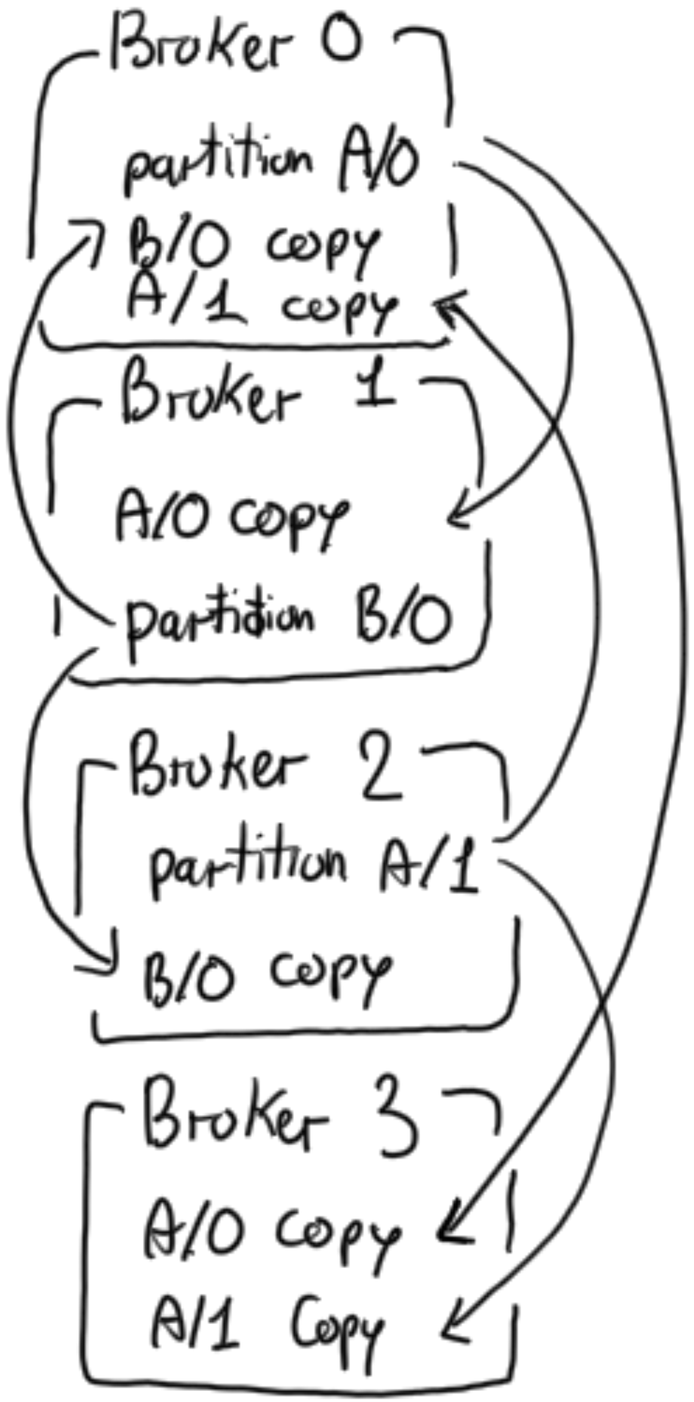
Fault Tolerance via a Replicated Log

- Kafka maintains replicas of each partition on other Brokers in the cluster
  - Number of replicas is configurable
- One Broker is the leader for that Partition
  - All writes and reads go to and from the leader



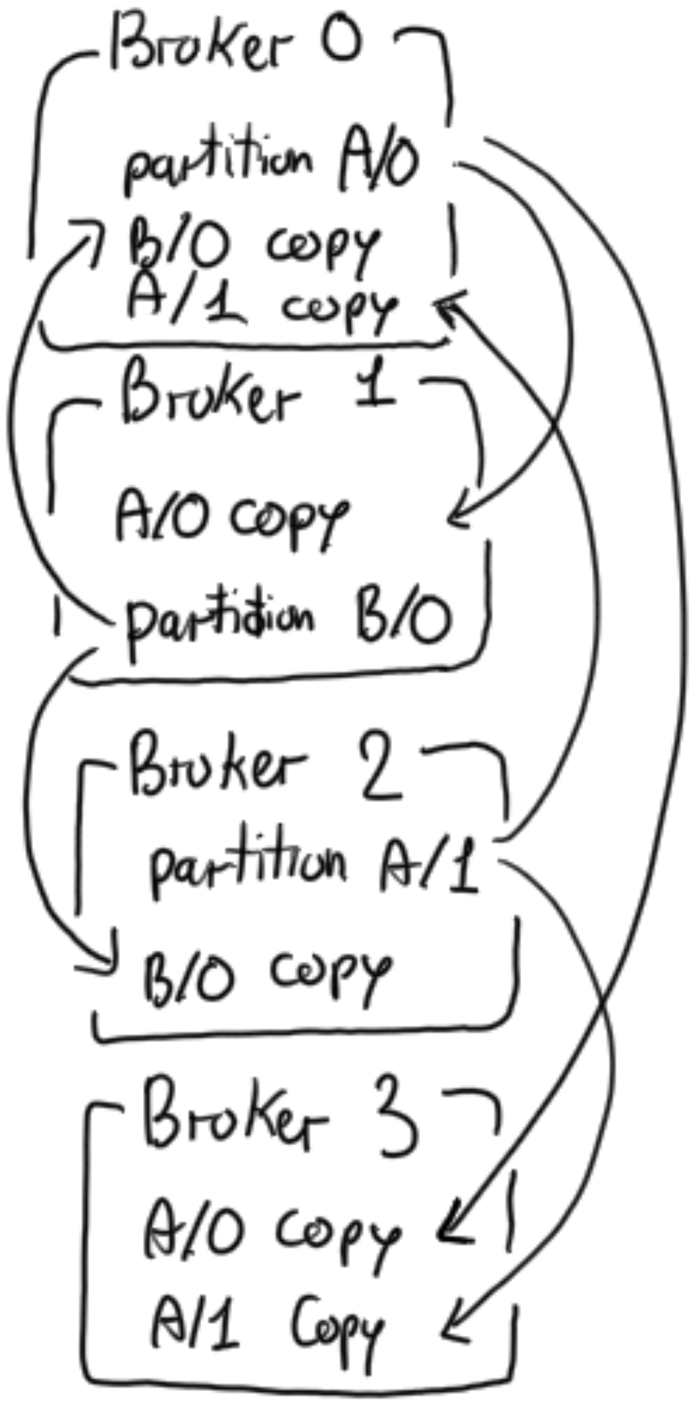
Fault Tolerance via a Replicated Log

- Kafka maintains replicas of each partition on other Brokers in the cluster
  - Number of replicas is configurable
- One Broker is the leader for that Partition
  - All writes and reads go to and from the leader
  - Other Brokers are followers



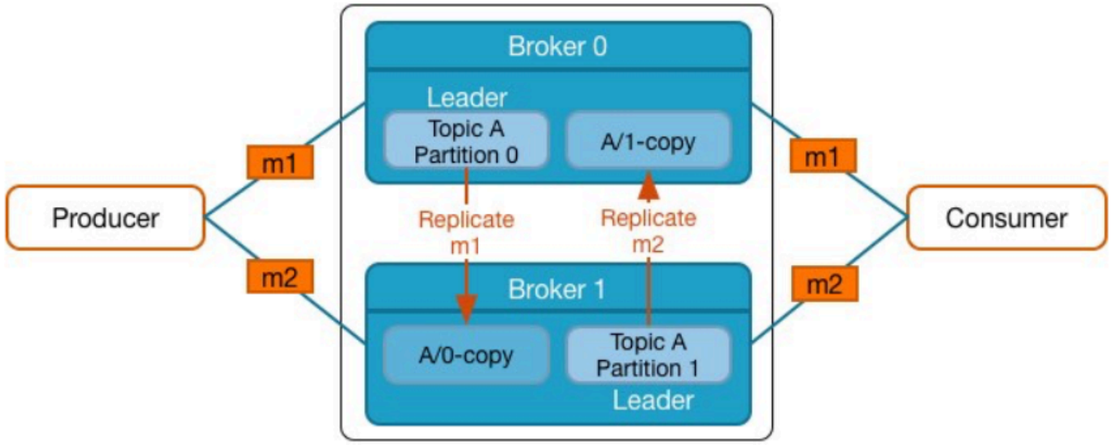
Fault Tolerance via a Replicated Log

- Kafka maintains replicas of each partition on other Brokers in the cluster
  - Number of replicas is configurable
- One Broker is the leader for that Partition
  - All writes and reads go to and from the leader
  - Other Brokers are followers
- Replication provides fault tolerance in case a Broker goes down



Important: Clients do not Access Followers

It is important to understand that Producers and Consumers only write/read to/from the leader

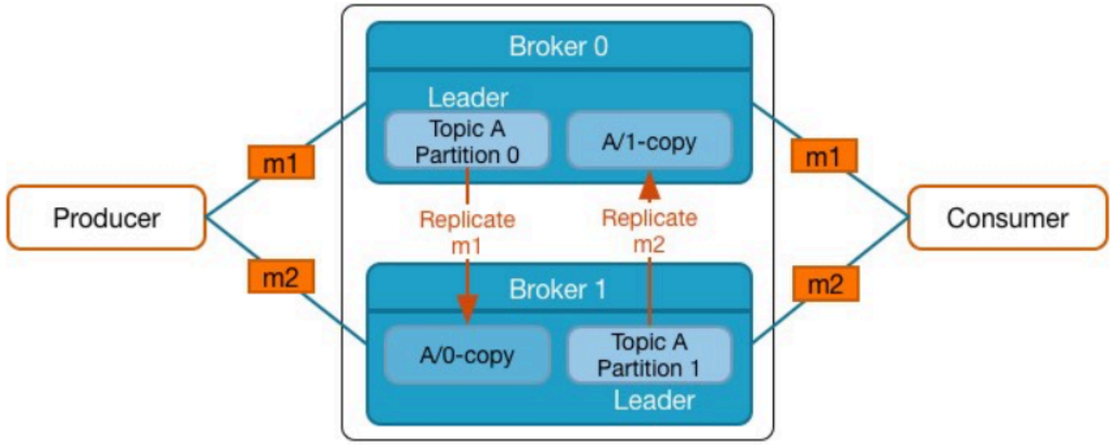


In the diagram, m1 hashes to Partition 0 and m2 hashes to Partition 1

Important: Clients do not Access Followers

It is important to understand that Producers and Consumers only write/read to/from the leader

- Replicas only exist to provide reliability in case of\
  Broker failure



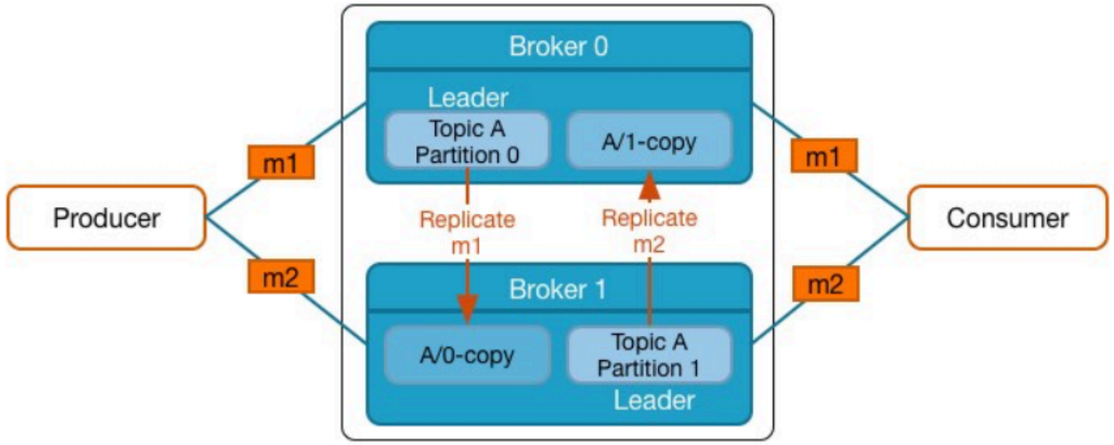
In the diagram, m1 hashes to Partition 0 and m2 hashes to Partition 1

Important: Clients do not Access Followers

It is important to understand that Producers and Consumers only write/read to/from the leader

- Replicas only exist to provide reliability in case of\
  Broker failure
- If a leader fails,\
  the Kafka cluster will\
  elect a new leader\
  from among the followers

In the diagram, m1 hashes to Partition 0 and m2 hashes to Partition 1



Delivery Semantics

Delivery Semantics

- At least once

Delivery Semantics

- At least once
  - Messages are never lost but may be redelivered

Delivery Semantics

- At least once
  - Messages are never lost but may be redelivered
- At most once

Delivery Semantics

- At least once
  - Messages are never lost but may be redelivered
- At most once
  - Messages are lost but never redelivered

Delivery Semantics

- At least once
  - Messages are never lost but may be redelivered
- At most once
  - Messages are lost but never redelivered
- Exactly once

Delivery Semantics

- At least once
  - Messages are never lost but may be redelivered
- At most once
  - Messages are lost but never redelivered
- Exactly once
  - Messages are delivered once and only once

Zookeeper

Zookeeper

- ZooKeeper is a centralized service that stores configurations for distributed applications

Zookeeper

- ZooKeeper is a centralized service that stores configurations for distributed applications
- Kafka Brokers use ZooKeeper for a number of important internal features

Zookeeper

- ZooKeeper is a centralized service that stores configurations for distributed applications
- Kafka Brokers use ZooKeeper for a number of important internal features
  - Cluster management

Zookeeper

- ZooKeeper is a centralized service that stores configurations for distributed applications
- Kafka Brokers use ZooKeeper for a number of important internal features
  - Cluster management
  - Failure detection and recovery

Zookeeper

- ZooKeeper is a centralized service that stores configurations for distributed applications
- Kafka Brokers use ZooKeeper for a number of important internal features
  - Cluster management
  - Failure detection and recovery
  - Access Control List (ACL) storage

Quiz

Provide the correct relationship - 1:1, 1:N, N:1, or N:N -

Quiz

Provide the correct relationship - 1:1, 1:N, N:1, or N:N -

- Broker to Partition - ?

Quiz

Provide the correct relationship - 1:1, 1:N, N:1, or N:N -

- Broker to Partition - ?
- Key to Partition - ?

Quiz

Provide the correct relationship - 1:1, 1:N, N:1, or N:N -

- Broker to Partition - ?
- Key to Partition - ?
- Producer to Topic - ?

Quiz

Provide the correct relationship - 1:1, 1:N, N:1, or N:N -

- Broker to Partition - ?
- Key to Partition - ?
- Producer to Topic - ?
- Consumer Group to Topic - ?

Quiz

Provide the correct relationship - 1:1, 1:N, N:1, or N:N -

- Broker to Partition - ?
- Key to Partition - ?
- Producer to Topic - ?
- Consumer Group to Topic - ?
- Consumer (in a Consumer Group) to Partition - ?

Quiz

Provide the correct relationship - 1:1, 1:N, N:1, or N:N -

Quiz

Provide the correct relationship - 1:1, 1:N, N:1, or N:N -

- Broker to Partition - N:N

Quiz

Provide the correct relationship - 1:1, 1:N, N:1, or N:N -

- Broker to Partition - N:N
- Key to Partition - N:1

Quiz

Provide the correct relationship - 1:1, 1:N, N:1, or N:N -

- Broker to Partition - N:N
- Key to Partition - N:1
- Producer to Topic - N:N

Quiz

Provide the correct relationship - 1:1, 1:N, N:1, or N:N -

- Broker to Partition - N:N
- Key to Partition - N:1
- Producer to Topic - N:N
- Consumer Group to Topic - N:N

Quiz

Provide the correct relationship - 1:1, 1:N, N:1, or N:N -

- Broker to Partition - N:N
- Key to Partition - N:1
- Producer to Topic - N:N
- Consumer Group to Topic - N:N
- Consumer (in a Consumer Group) to Partition - 1:N

Getting Exactly Once Semantics

Getting Exactly Once Semantics

- Must consider two components

Getting Exactly Once Semantics

- Must consider two components
  - Durability guarantees when publishing a message

Getting Exactly Once Semantics

- Must consider two components
  - Durability guarantees when publishing a message
  - Durability guarantees when consuming a message

Getting Exactly Once Semantics

- Must consider two components
  - Durability guarantees when publishing a message
  - Durability guarantees when consuming a message
- Producer

Getting Exactly Once Semantics

- Must consider two components
  - Durability guarantees when publishing a message
  - Durability guarantees when consuming a message
- Producer
  - What happens when a produce request was sent but a network error returned before an ack?

Getting Exactly Once Semantics

- Must consider two components
  - Durability guarantees when publishing a message
  - Durability guarantees when consuming a message
- Producer
  - What happens when a produce request was sent but a network error returned before an ack?
  - Use a single writer per partition and check the latest committed value after network errors

Getting Exactly Once Semantics

- Must consider two components
  - Durability guarantees when publishing a message
  - Durability guarantees when consuming a message
- Producer
  - What happens when a produce request was sent but a network error returned before an ack?
  - Use a single writer per partition and check the latest committed value after network errors
- Consumer

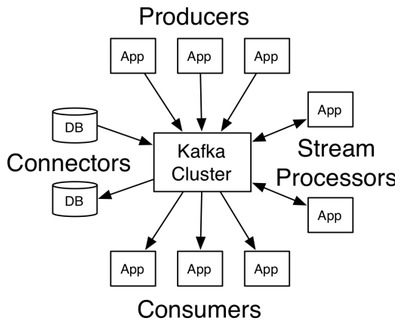
Getting Exactly Once Semantics

- Must consider two components
  - Durability guarantees when publishing a message
  - Durability guarantees when consuming a message
- Producer
  - What happens when a produce request was sent but a network error returned before an ack?
  - Use a single writer per partition and check the latest committed value after network errors
- Consumer
  - Include a unique ID (e.g. UUID) and de-duplicate.

Getting Exactly Once Semantics

- Must consider two components
  - Durability guarantees when publishing a message
  - Durability guarantees when consuming a message
- Producer
  - What happens when a produce request was sent but a network error returned before an ack?
  - Use a single writer per partition and check the latest committed value after network errors
- Consumer
  - Include a unique ID (e.g. UUID) and de-duplicate.
  - Consider storing offsets with data

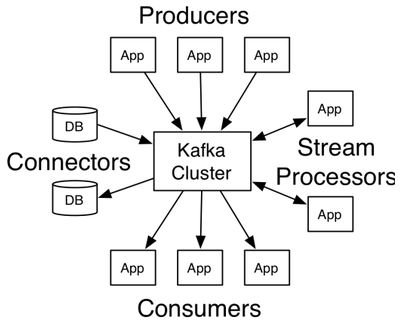
Systems Overview: [[Apache Kafka]]



References:

Kreps, Jay, Neha Narkhede, and Jun Rao. "Kafka: A distributed messaging system for log processing." Proceedings of the NetDB. Vol. 11. 2011.
Wang, Guozhang, et al. "Building a replicated logging system with Apache Kafka." Proceedings of the VLDB Endowment 8.12 (2015): 1654-1655.

Systems Overview: [[Apache Kafka]]

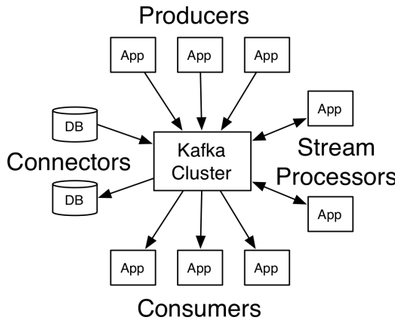


- Apache Kafka is a scalable replicated commit log that enables stream processing at scale.

References:

Kreps, Jay, Neha Narkhede, and Jun Rao. "Kafka: A distributed messaging system for log processing." Proceedings of the NetDB. Vol. 11. 2011.
Wang, Guozhang, et al. "Building a replicated logging system with Apache Kafka." Proceedings of the VLDB Endowment 8.12 (2015): 1654-1655.

Systems Overview: [[Apache Kafka]]

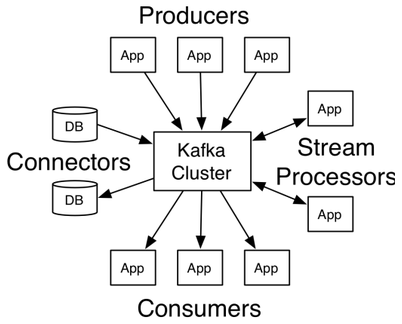


- Apache Kafka is a scalable replicated commit log that enables stream processing at scale.
- It can handle huge numbers of concurrent reads and writes

References:

Kreps, Jay, Neha Narkhede, and Jun Rao. "Kafka: A distributed messaging system for log processing." Proceedings of the NetDB. Vol. 11. 2011.
Wang, Guozhang, et al. "Building a replicated logging system with Apache Kafka." Proceedings of the VLDB Endowment 8.12 (2015): 1654-1655.

Systems Overview: [[Apache Kafka]]

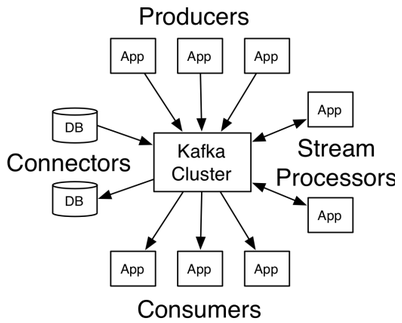


- Apache Kafka is a scalable replicated commit log that enables stream processing at scale.
- It can handle huge numbers of concurrent reads and writes
- It comes with connector to a number of Big Data Framework, e.g., `Storm`, `Samza`, `Flink`, `Spark`.

References:

Kreps, Jay, Neha Narkhede, and Jun Rao. "Kafka: A distributed messaging system for log processing." Proceedings of the NetDB. Vol. 11. 2011.
Wang, Guozhang, et al. "Building a replicated logging system with Apache Kafka." Proceedings of the VLDB Endowment 8.12 (2015): 1654-1655.

Systems Overview: [[Apache Kafka]]



- Apache Kafka is a scalable replicated commit log that enables stream processing at scale.
- It can handle huge numbers of concurrent reads and writes
- It comes with connector to a number of Big Data Framework, e.g., `Storm`, `Samza`, `Flink`, `Spark`.
- It persists messages on disk and replicated within the cluster.

References:

Kreps, Jay, Neha Narkhede, and Jun Rao. "Kafka: A distributed messaging system for log processing." Proceedings of the NetDB. Vol. 11. 2011.
Wang, Guozhang, et al. "Building a replicated logging system with Apache Kafka." Proceedings of the VLDB Endowment 8.12 (2015): 1654-1655.