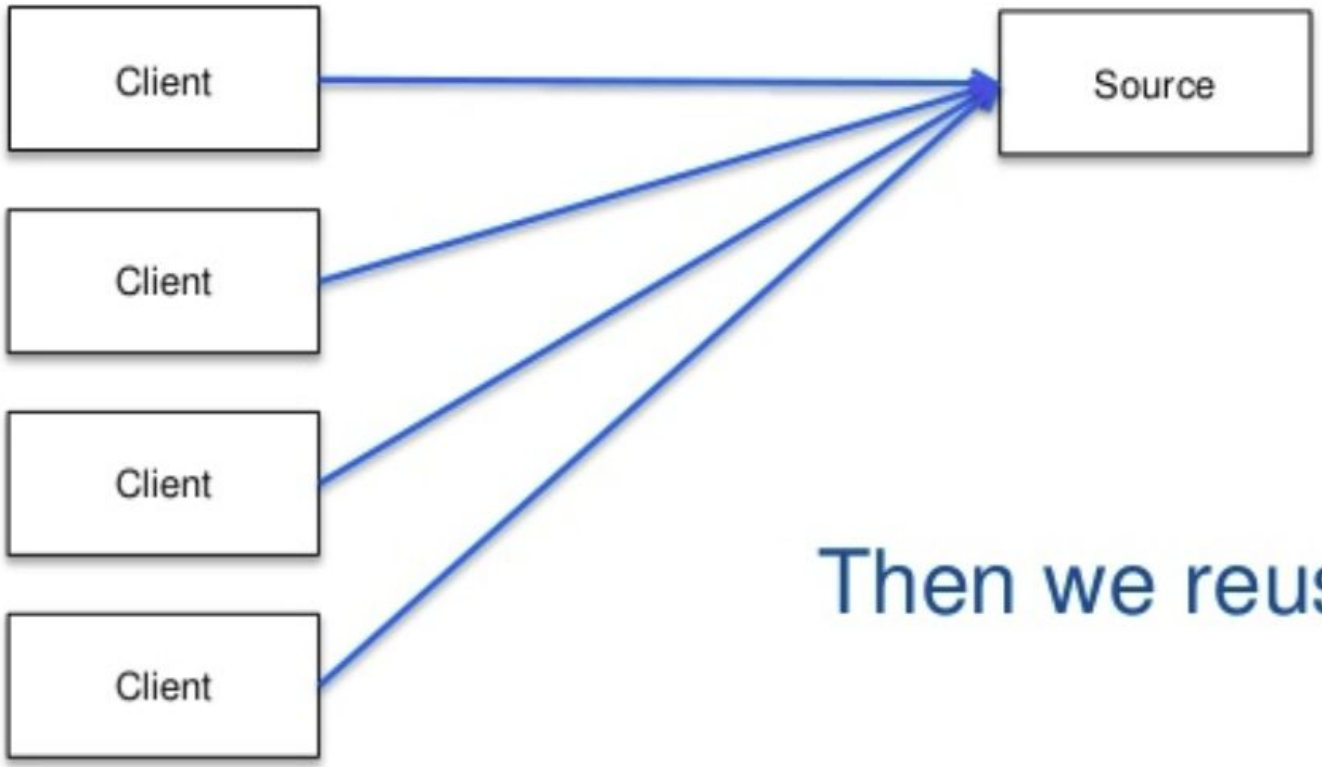
Apache Kafka

Samuel Adámik, Ľuboš Beňo,
Viktória Tóthová

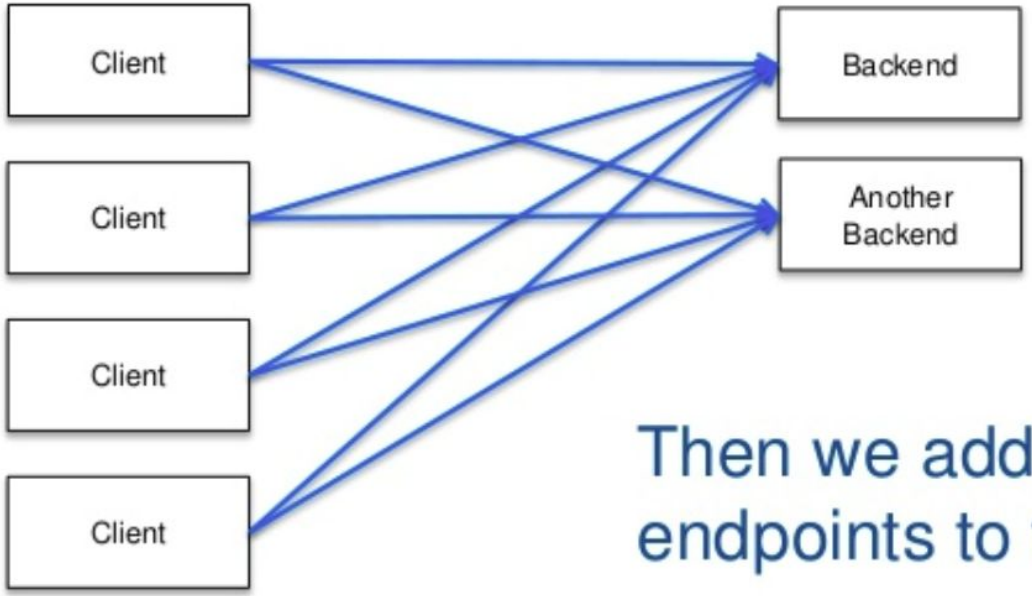

APACHE
kafka®

Why Kafka

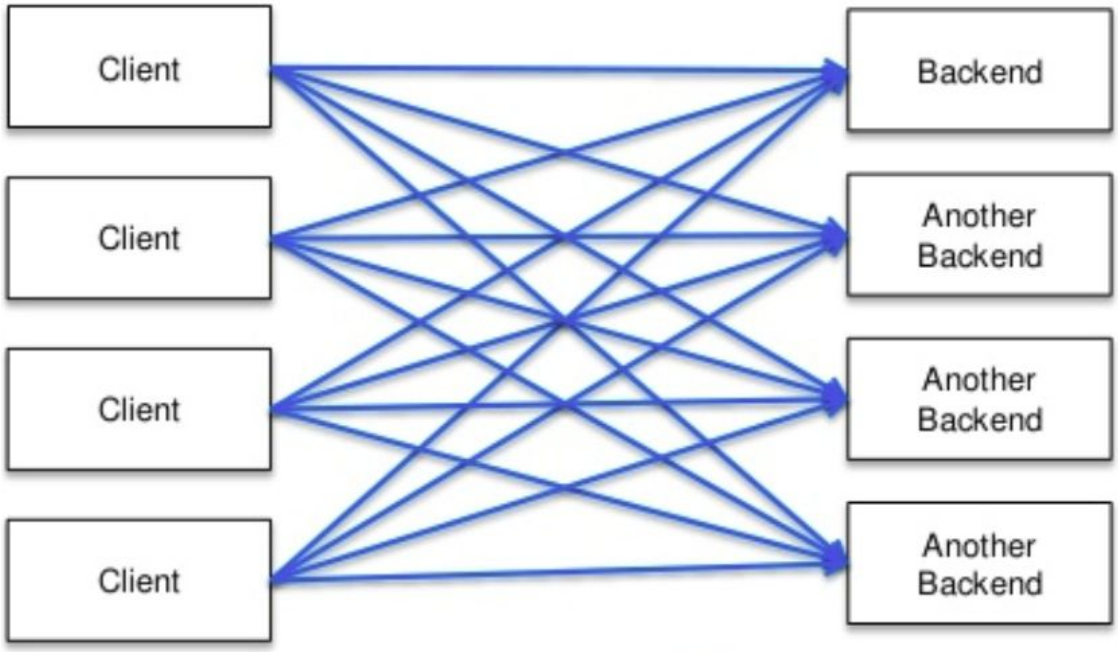
Data Pipelines Start like this.



Then we reuse them



Then we add additional endpoints to the existing sources



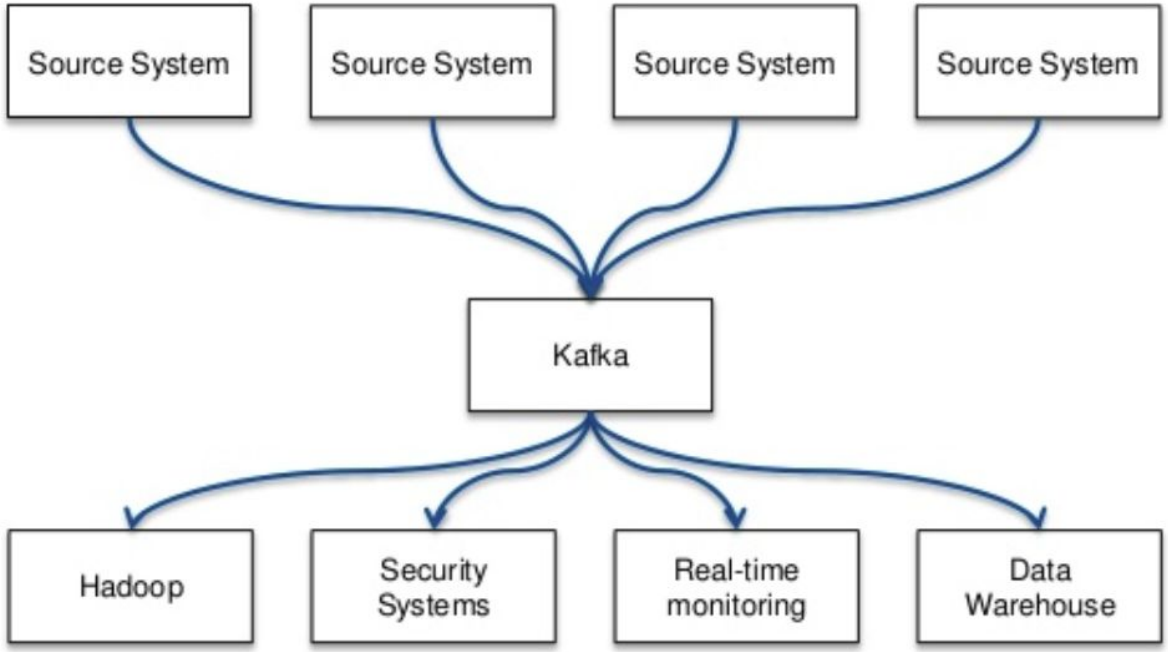
Then it starts to look like this

Kafka decouples data pipelines

Producers

Brokers

Consumers



What is Apache Kafka

- A distributed streaming platform
- Originally developed by LinkedIn in 2010
- Open-sourced in 2011 at Apache
- Since 2014 managed by Confluent
- Written in Scala and Java
- A unified, high-throughput, low-latency platform for handling real-time data feeds
- Real-time streaming data pipelines that reliably get data between systems or applications
- Real-time streaming applications that transform or react to the streams of data

Streaming platform

Has three key capabilities:

- Publish and subscribe to streams of records, similar to a message queue or enterprise messaging system.
- Store streams of records in a fault-tolerant durable way.
- Process streams of records as they occur.

Basic concepts

- Kafka is run as a cluster on one or more servers that can span multiple datacenters.
- The Kafka cluster stores streams of records in categories called topics.
- Each record consists of a key, a value, and a timestamp.

Kafka has 4 APIs

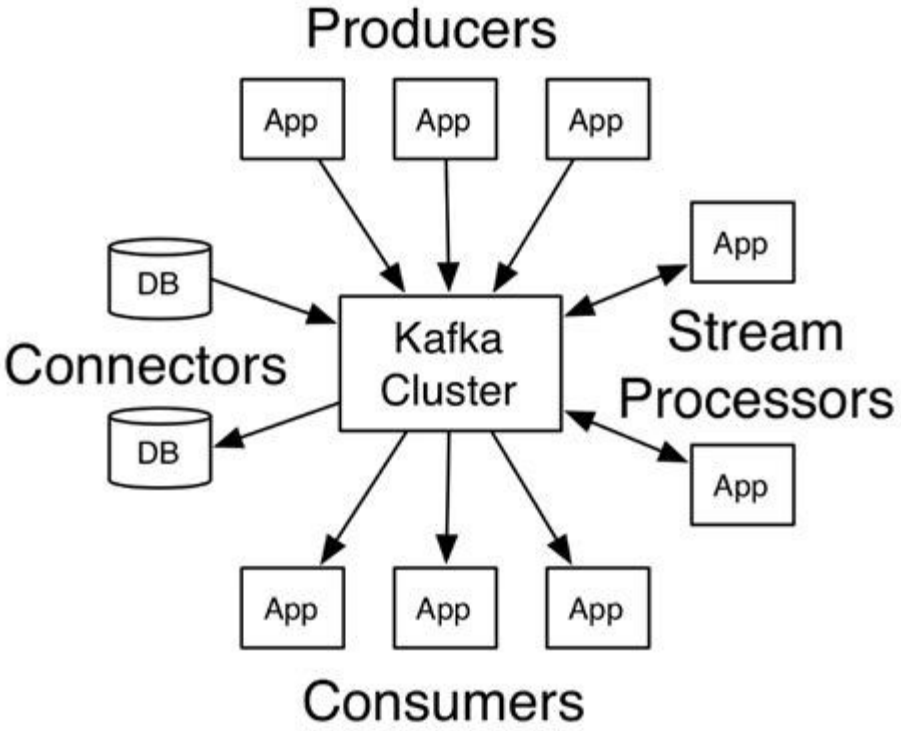
Topic - stored stream of records

Producer - publish a stream of records to Kafka topics.

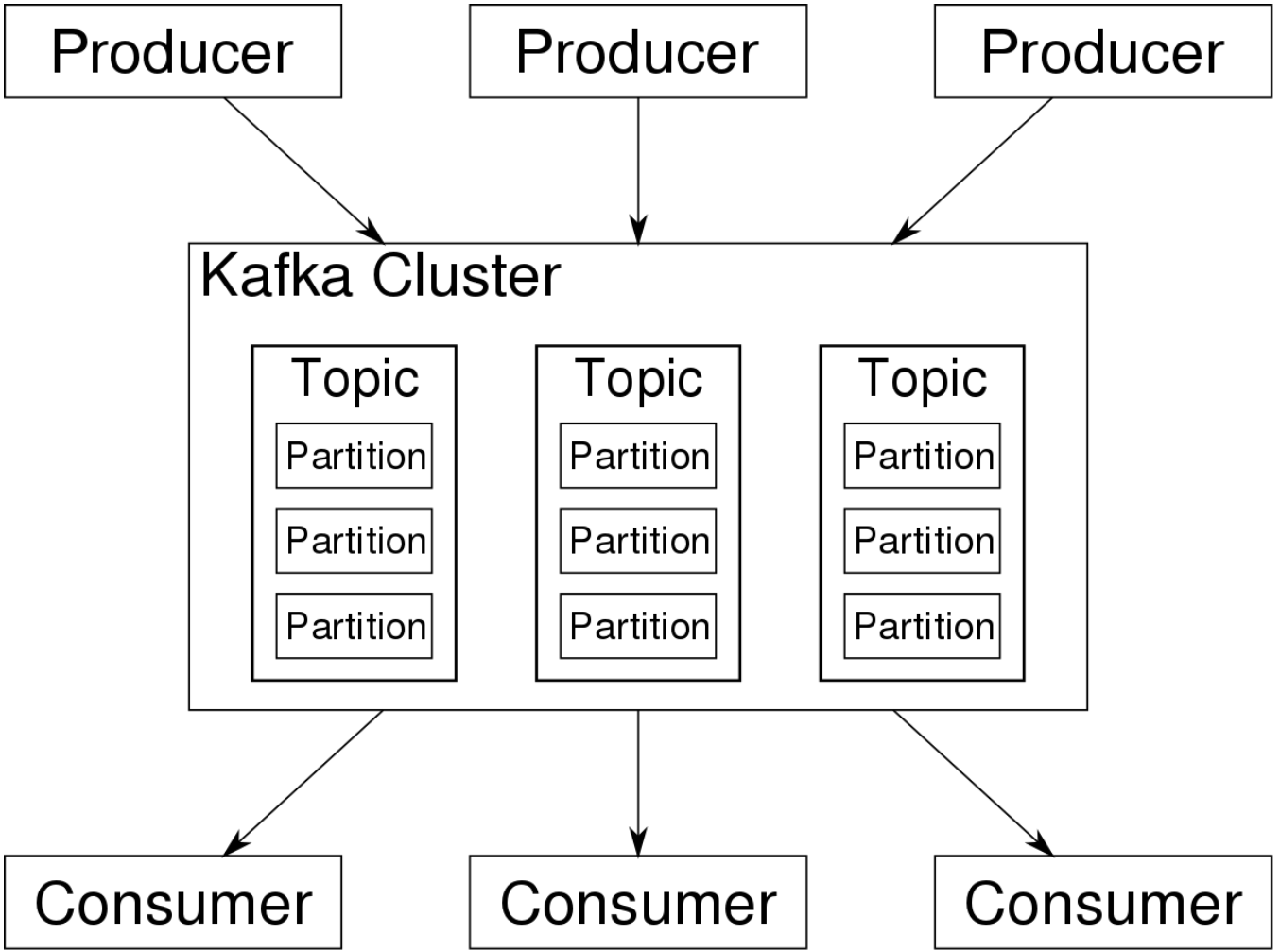
Consumer - subscribe to topic(s) and process the stream of records

Stream processor - consume an input stream from topic(s) and produce an output stream to output topic(s)

Connector - build and run reusable producers or consumers that connect Kafka topics to existing applications or data systems. For example, a connector to a relational database might capture every change to a table.



Kafka Architecture

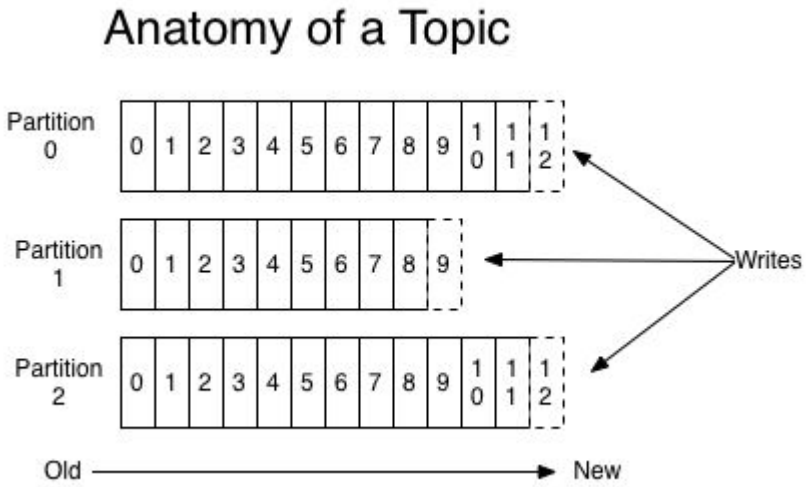


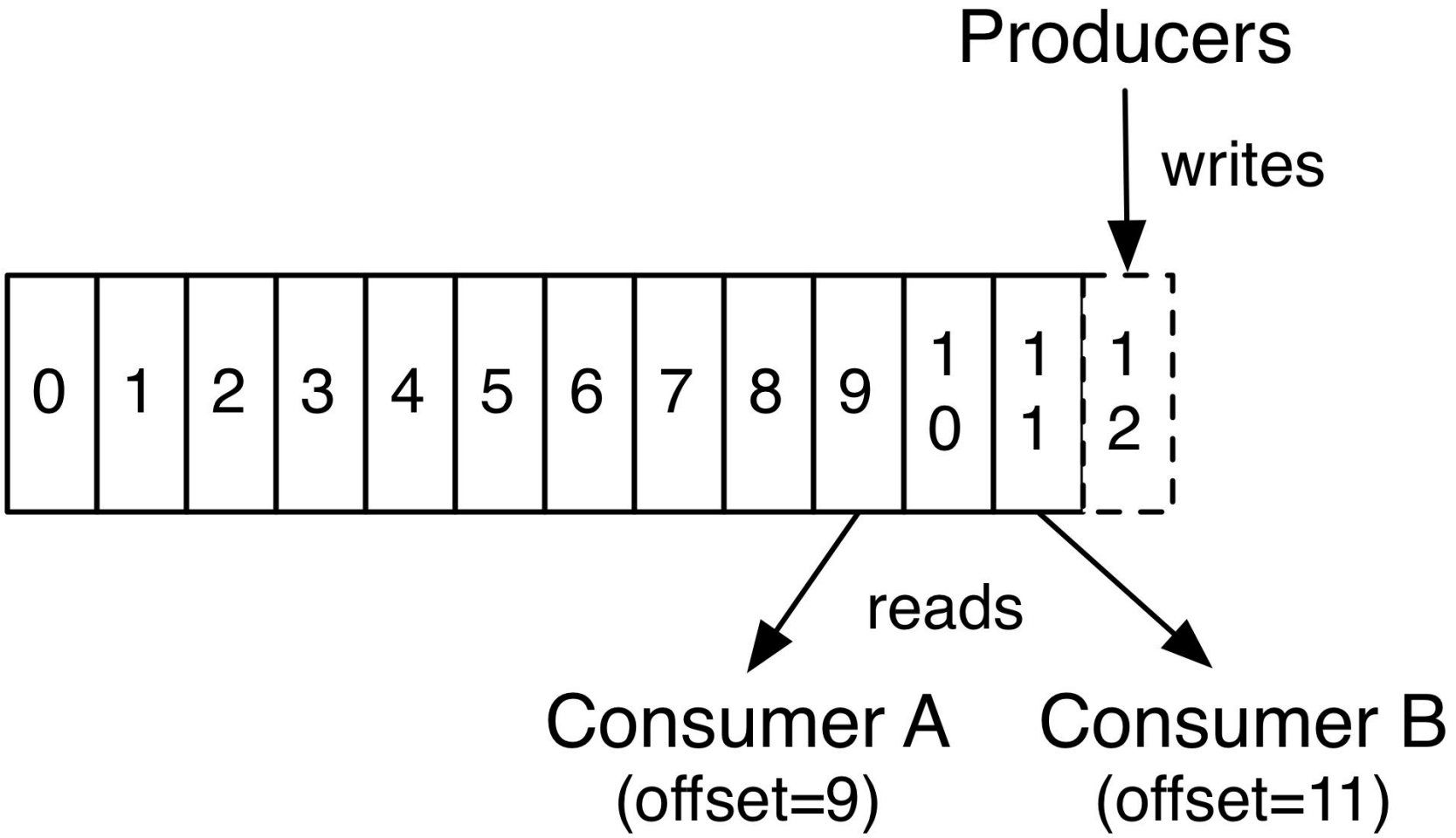
Topic

- A category to which records are published.
- Multi-subscriber (0 - N consumers)

Partition

- Ordered commit logs
- The records are assigned a sequential id number called the offset
- Configurable retention period
- Allow scalability




Producers
writes
0
1
2
3
4
5
6
7
8
9
10
11
12
reads
Consumer A
(offset=9)
Consumer B
(offset=11)

Distribution

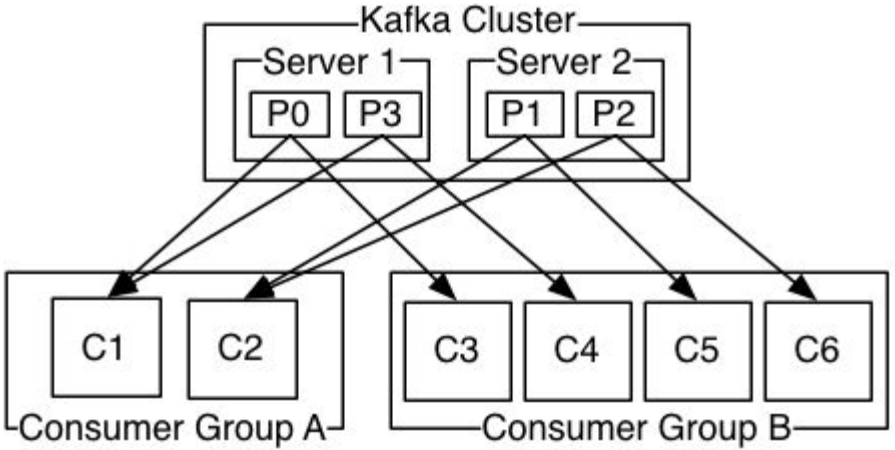
- Partitions are distributed over servers in cluster
- Each server handles data and requests for a share of partitions
- Each partition replicated for fault tolerance
- Each partition has servers called “leader and “followers”
- Leader handles all read and write requests for the partition
- Followers passively replicate the leader
- If leader fails, one follower becomes new leader.

Producers

- Publish data to topics
- Choose which record to assign to which partition within the topic
  - Round-robin (balanced load)
  - Semantic partition function (based on key in record)

Consumers

- Consumer Instances are grouped into Consumer Groups
- Each record published is delivered to one Instance
- Consumer instances can be in separate processes or on separate machines.



Guarantees

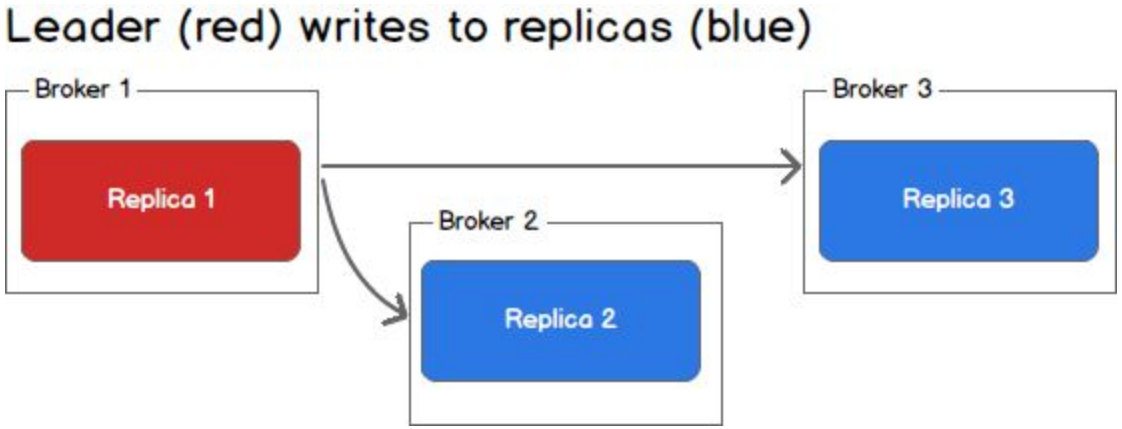
- Messages sent by a producer to a particular topic partition will be appended in the order they are sent. That is, if a record M1 is sent by the same producer as a record M2, and M1 is sent first, then M1 will have a lower offset than M2 and appear earlier in the log.
- A consumer instance sees records in the order they are stored in the log.
- For a topic with replication factor N, we will tolerate up to N-1 server failures without losing any records committed to the log.

Consistency and Availability

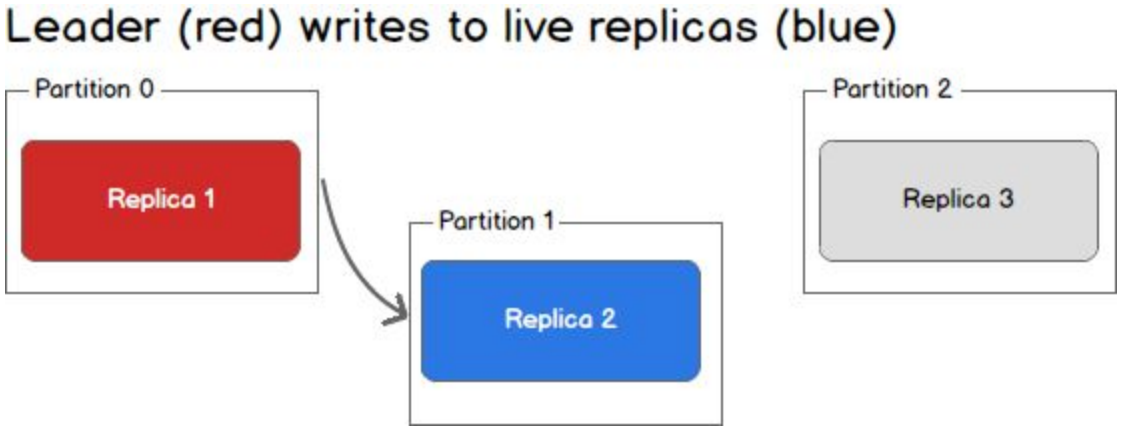
- All guarantees are off if you are reading from the same partition using two consumers or writing to the same partition using two producers

- Guarantees by Kafka
  a. messages sent to a topic partition will be appended to the commit log in the order they are sent
  b. a single consumer instance will see messages in the order they appear in the log
  c. a message is ‘committed’ when all in sync replicas have applied it to their log
  d. any committed message will not be lost, as long as at least one in sync replica is alive

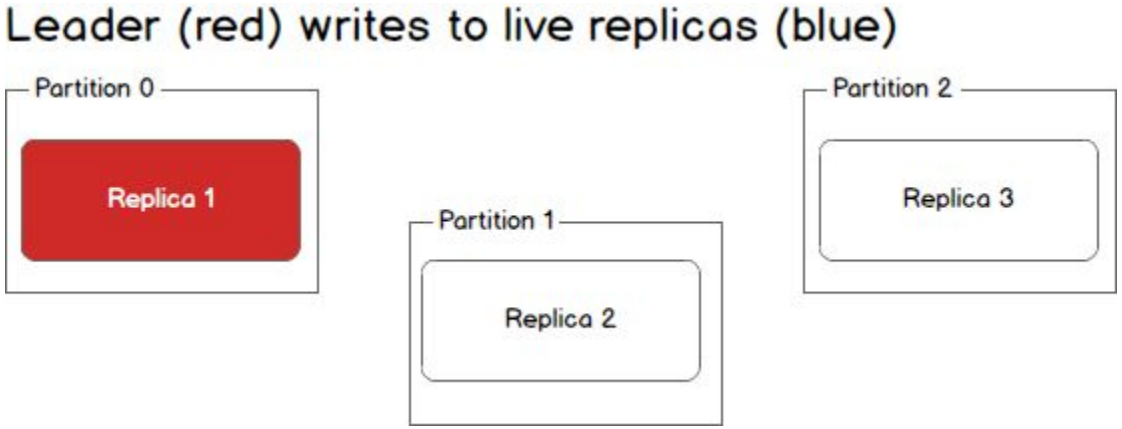
Handling writes



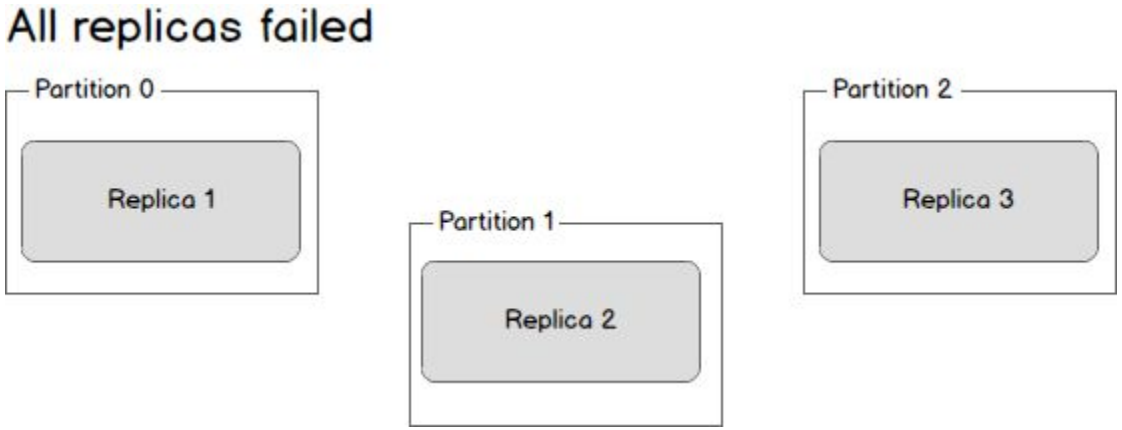
Handling failure 1/4



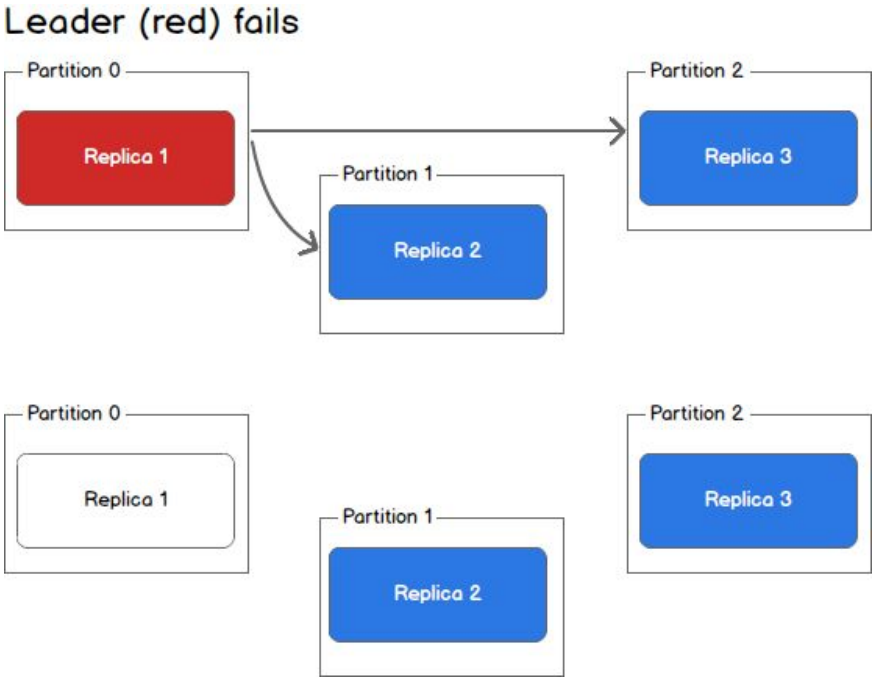
Handling failure 2/4



Handling failure 3/4



Handling failure 4/4



Use cases

- Messaging - Replacement for messaging systems (e.g. ActiveMQ, RabbitMQ)
- Website Activity Tracking - Real-time processing of site activity, page views, searches...
- Metrics
  - Operational monitoring of data, aggregating statistics from distributed applications
- Log Aggregation
  - Collecting log files from servers and putting them in a central place for processing
- Stream Processing
  - raw input data is consumed from Kafka topics and then aggregated, enriched, or transformed into new topics for further consumption or follow-up processing
- Event Sourcing
  - style of application design where state changes are logged as a time-ordered sequence of records
- Commit Log
  - external commit-log for a distributed system with replication between nodes

Enterprises that use Kafka

Sources

- https://kafka.apache.org/
- https://www.slideshare.net/jhols1/kafka-atlmeetuppublicv2
- https://sookocheff.com/post/kafka/kafka-in-a-nutshell/

Thank you for your attention!