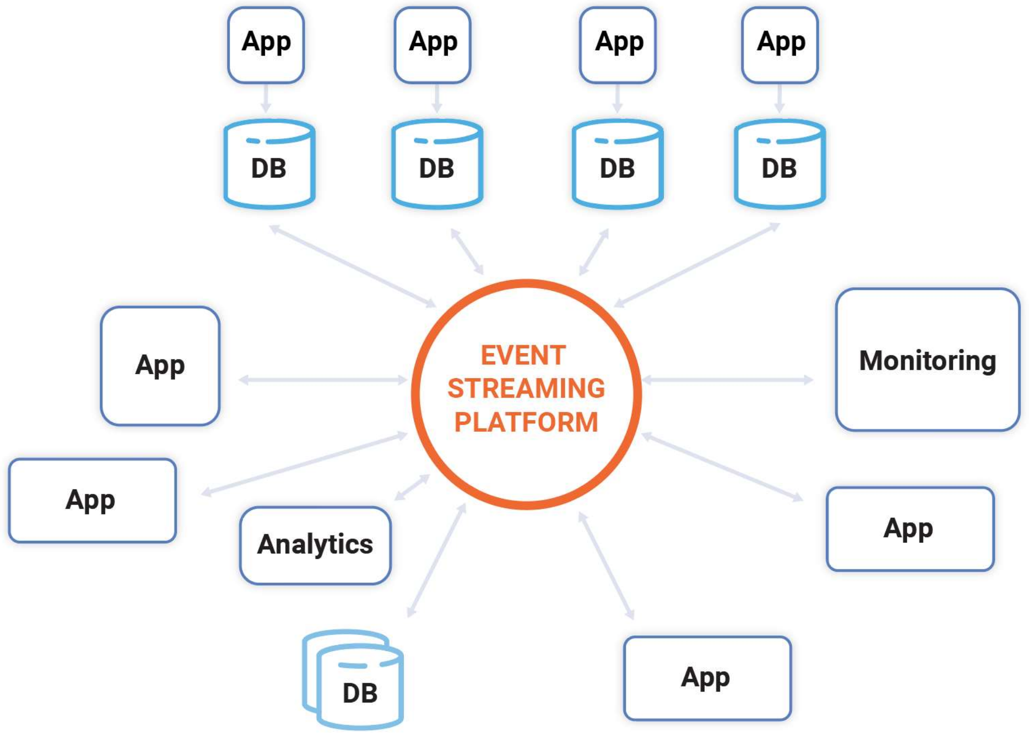
Distributed and Parallel Computing

Trong-Hop Do

Kafka – A distributed event streaming flatform

What is event streaming?

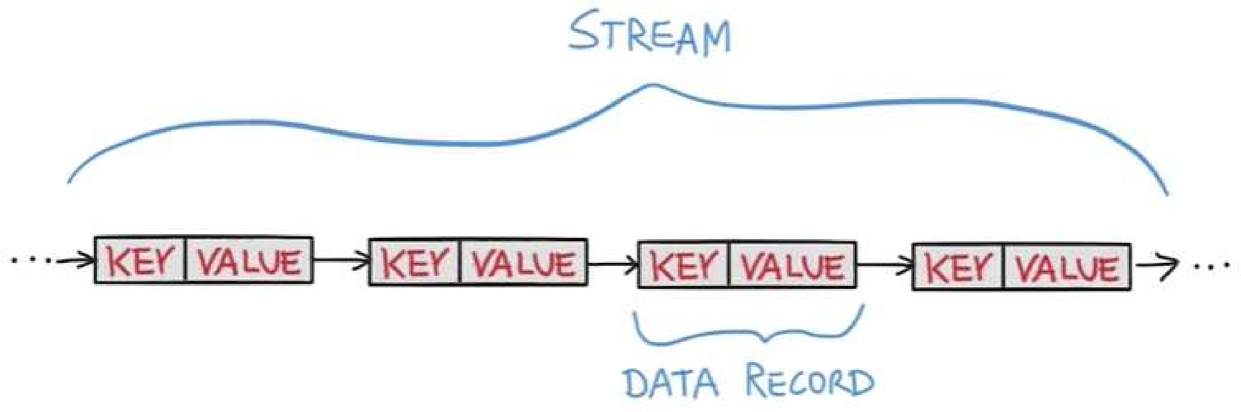


What can I use event streaming for?

- To process payments and financial transactions in real-time, such as in stock exchanges, banks, and insurances.
- To track and monitor cars, trucks, fleets, and shipments in real-time, such as in logistics and the automotive industry.
- To continuously capture and analyze sensor data from IoT devices or other equipment, such as in factories and wind parks.
- To collect and immediately react to customer interactions and orders, such as in retail, the hotel and travel industry, and mobile applications.
- To monitor patients in hospital care and predict changes in condition to ensure timely treatment in emergencies.
- To connect, store, and make available data produced by different divisions of a company.
- To serve as the foundation for data platforms, event-driven architectures, and microservices.

What is a stream?

- **Think of a stream as an unbounded, continuous real-time flow of records**
  - You don't need to explicitly request new records, you just receive them
- **Records are key-value pairs**



Motivation

The Shift to Event-driven Systems has Already Begun...

From a static snapshot... **...to a continuous stream of events**

Occasional call to a friend

A constant feed about the activities of all your friends

Daily news reports

Real time news feeds, accessible online anytime, anywhere

Motivation

This leads us to…

✓ **Single platform** to connect everyone to every event

✓ **Real-time** stream of events

✓ **All events stored** for historical view

Motivation

Successful Digital Businesses are Inherently Event-driven

Born cloud-native...

Social Networks
Enabling Event Sharing

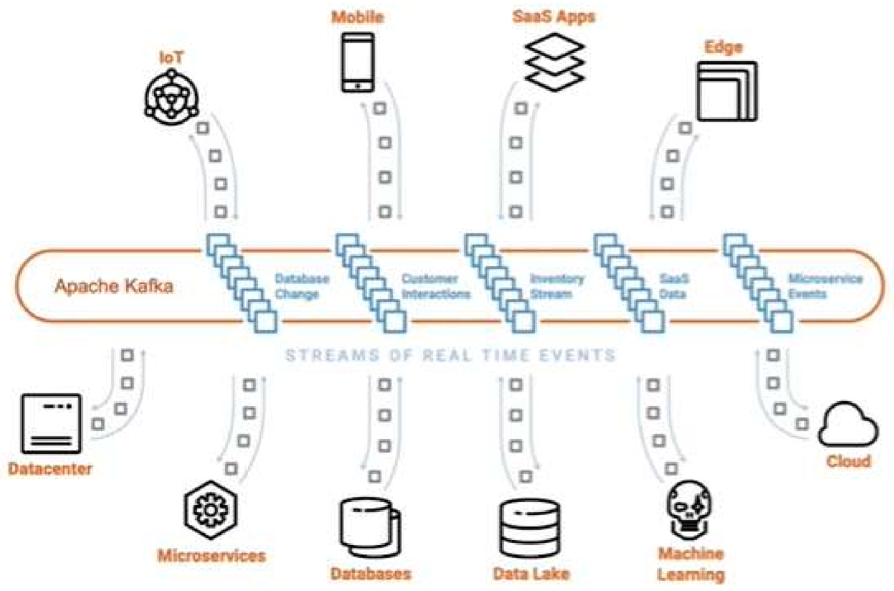
Streaming Provider
On-demand Digital Content

Traditional ones that adapt...

Newspaper
Provide a single Source of Truth

Credit Card Payments
Microservices Architecture

Motivation



Apache Kafka®: the De-facto Standard for Real-Time Event Streaming

- Global-scale
- Real-time
- Persistent Storage
- Stream Processing

Motivation

Thousands of Companies Worldwide trust Kafka for their Journey towards "**Event-driven**"

Travel

■■■■■■□□□□

6 of top 10

Global banks

■■■■■■■□□□

7 of top 10

Insurance

■■■■■■■■□□

8 of top 10

Telecom

■■■■■■■■■□

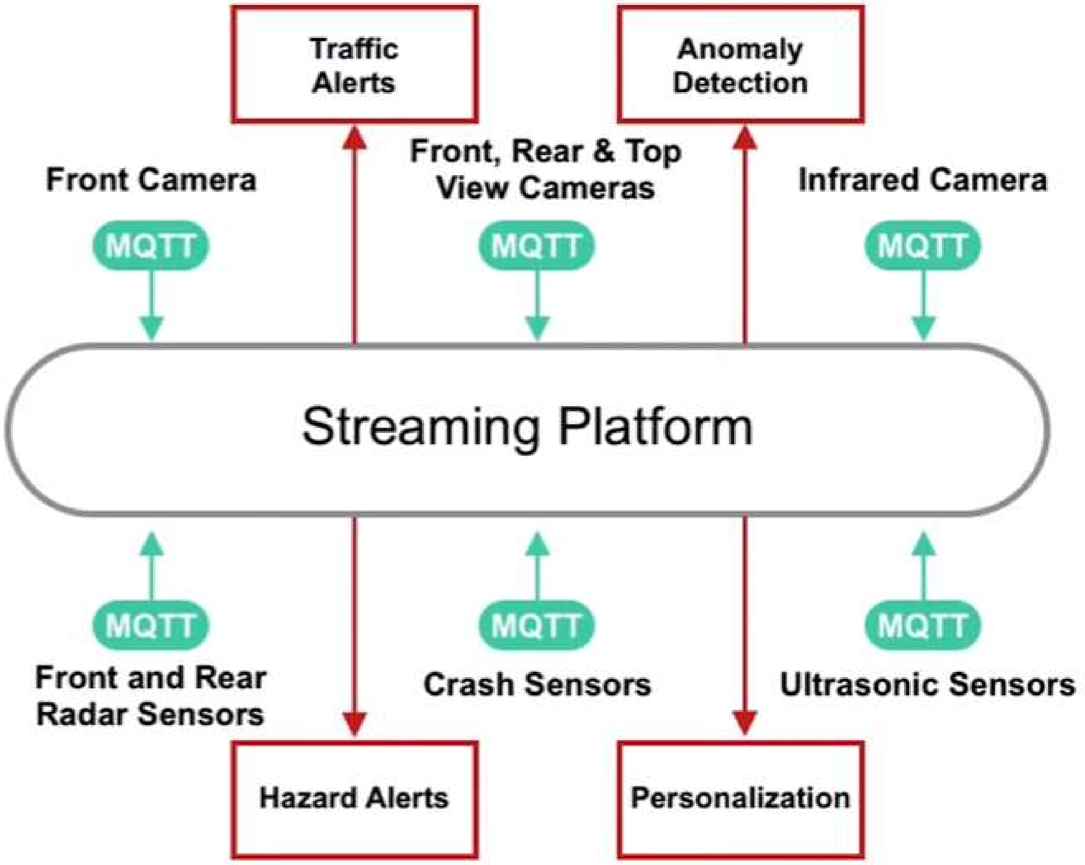
9 of top 10

Real-time Fraud Detection

- Act in real-time
- Detect fraud
- Minimize risk
- Improve customer experience

Automotive

The Future of the Automotive Industry is a Real Time Data Cluster

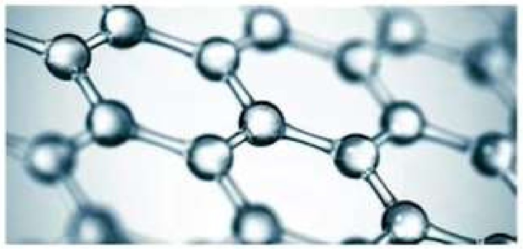


Real-time e-Commerce

Rewards Program

- Onboarding new merchants faster
- Increased speed at which mobile applications are delivered to customers
- Enabled a full 360 view of customers
- Enhanced performance and monitoring
- Projected savings of millions of dollars

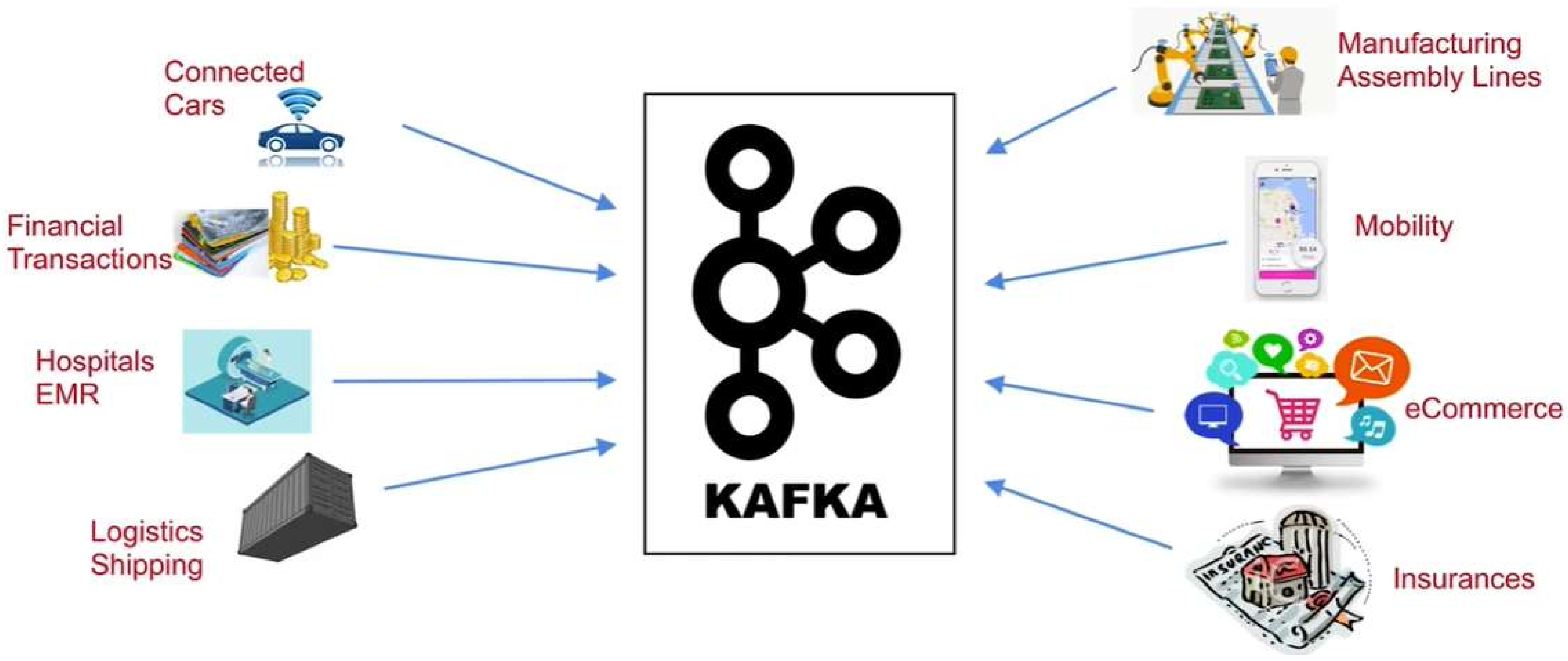
Health Care



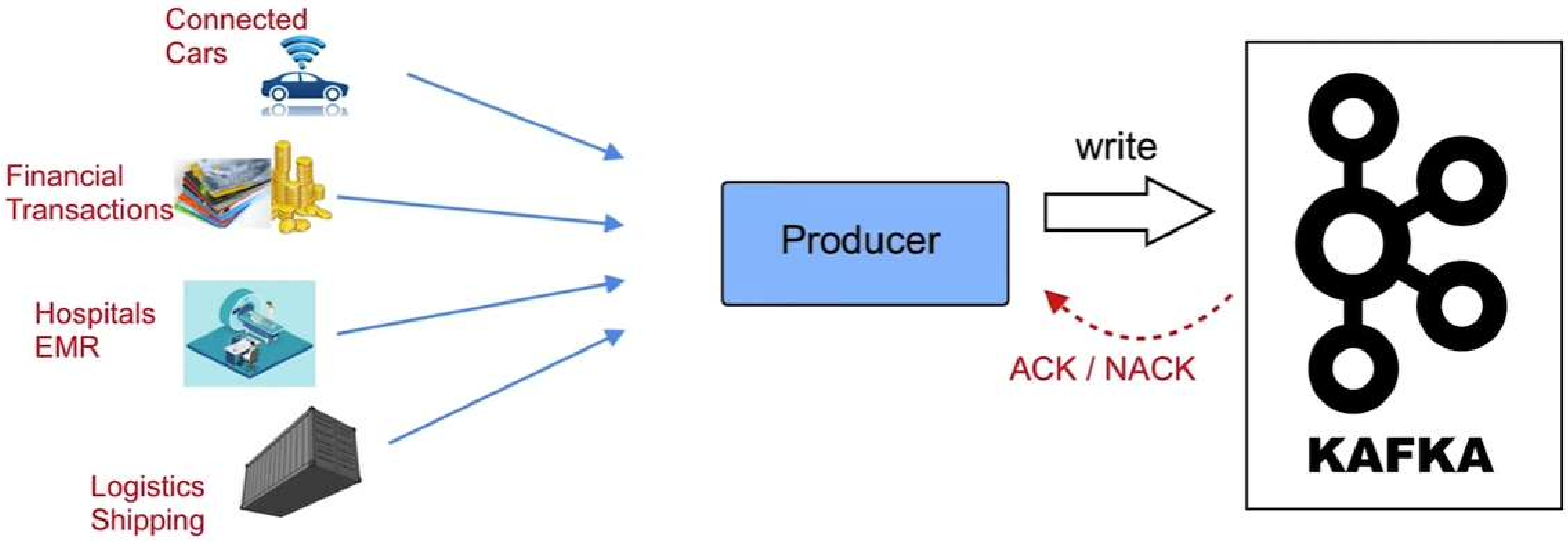
- Microservices

- Internet of Things

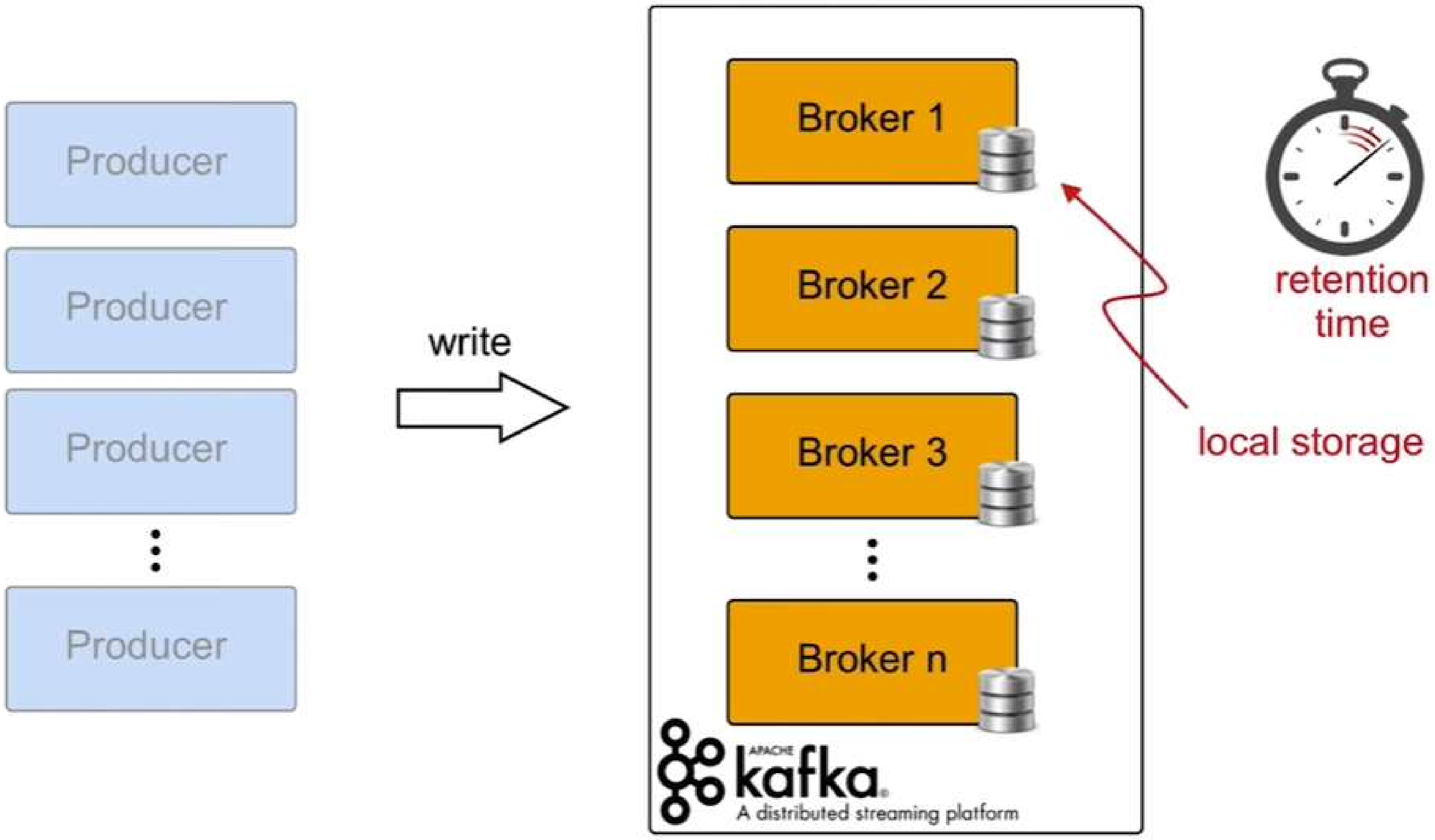
The World Produces Data



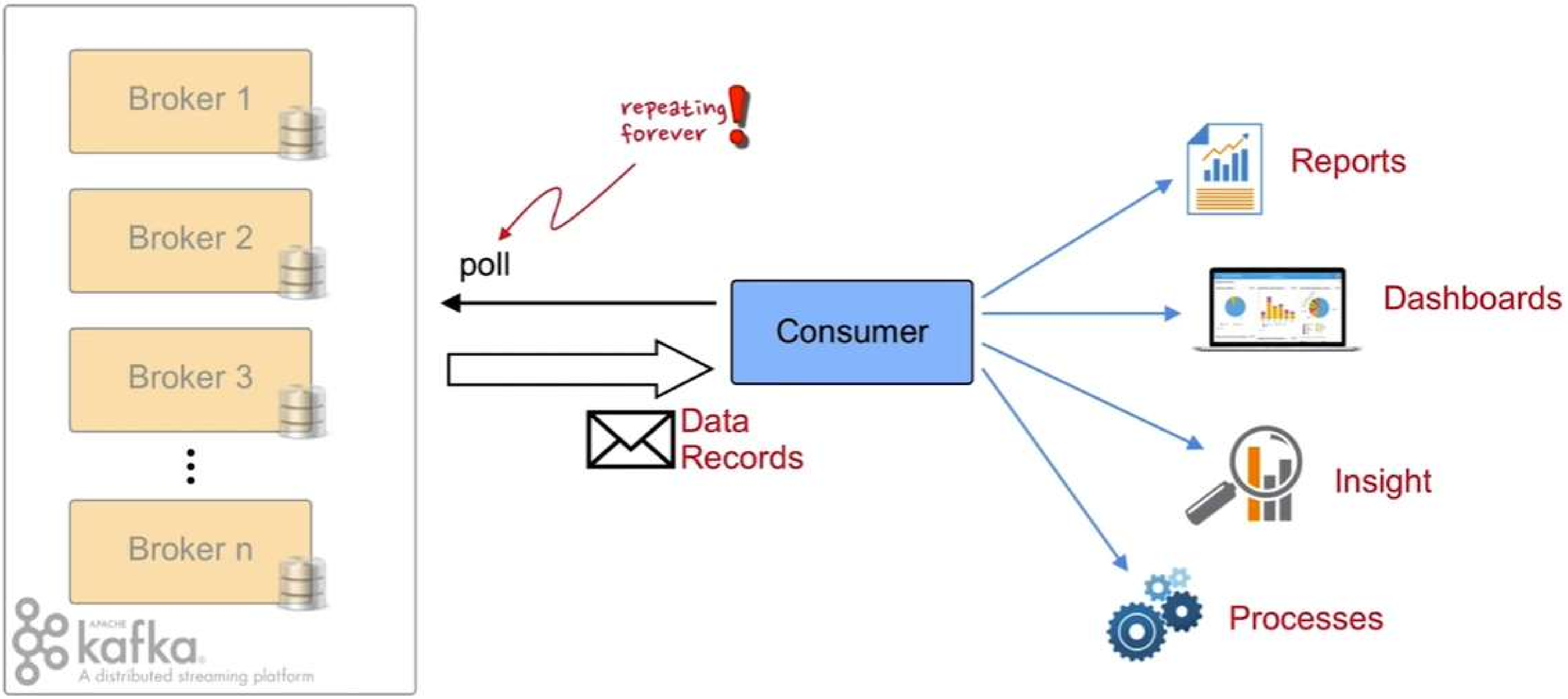
Producers


Connected Cars
Financial Transactions
Hospitals EMR
Logistics Shipping
Producer
write
ACK / NACK
KAFKA

Kafka Brokers

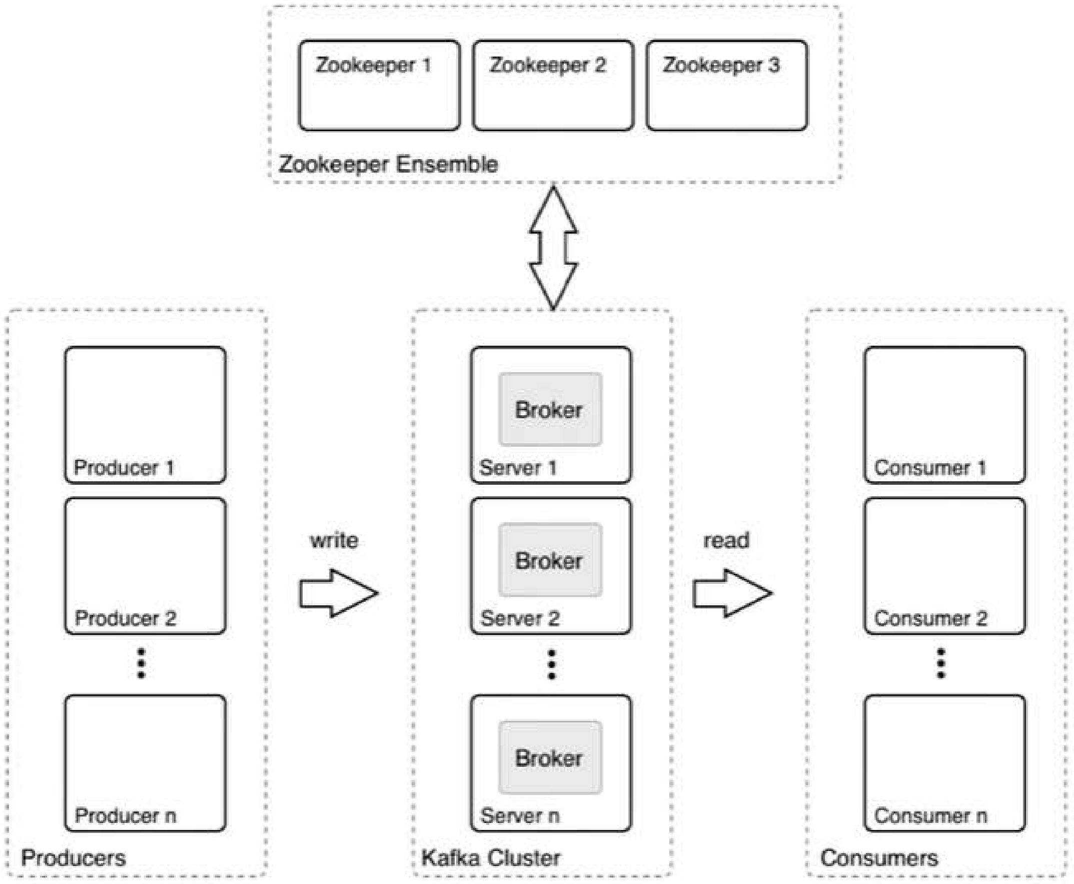


Consumers



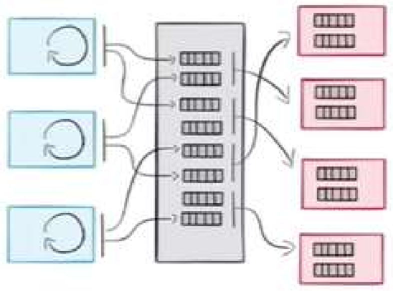
Broker 1
Broker 2
Broker 3
Broker n
kafka
A distributed streaming platform
repeating forever!
poll
Consumer
Data Records
Reports
Dashboards
Insight
Processes

Architecture

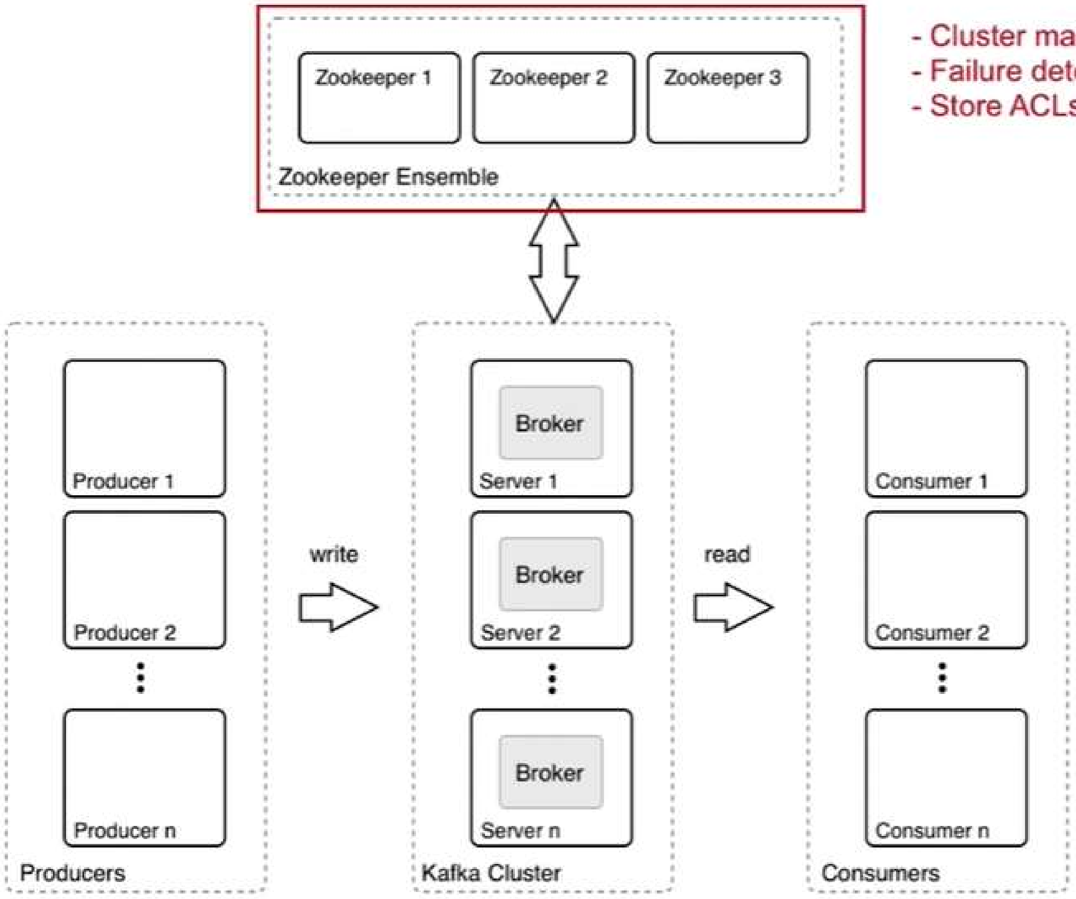


Decoupling Producers and Consumers

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



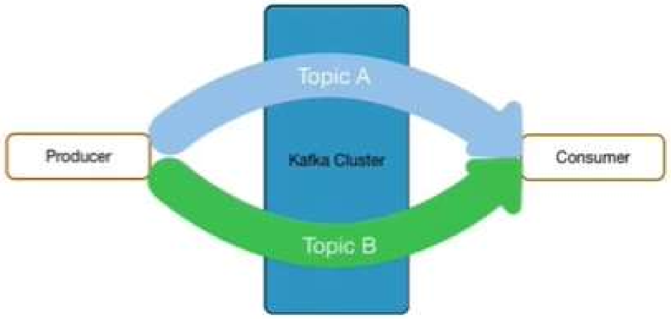
How Kafka Uses ZooKeeper



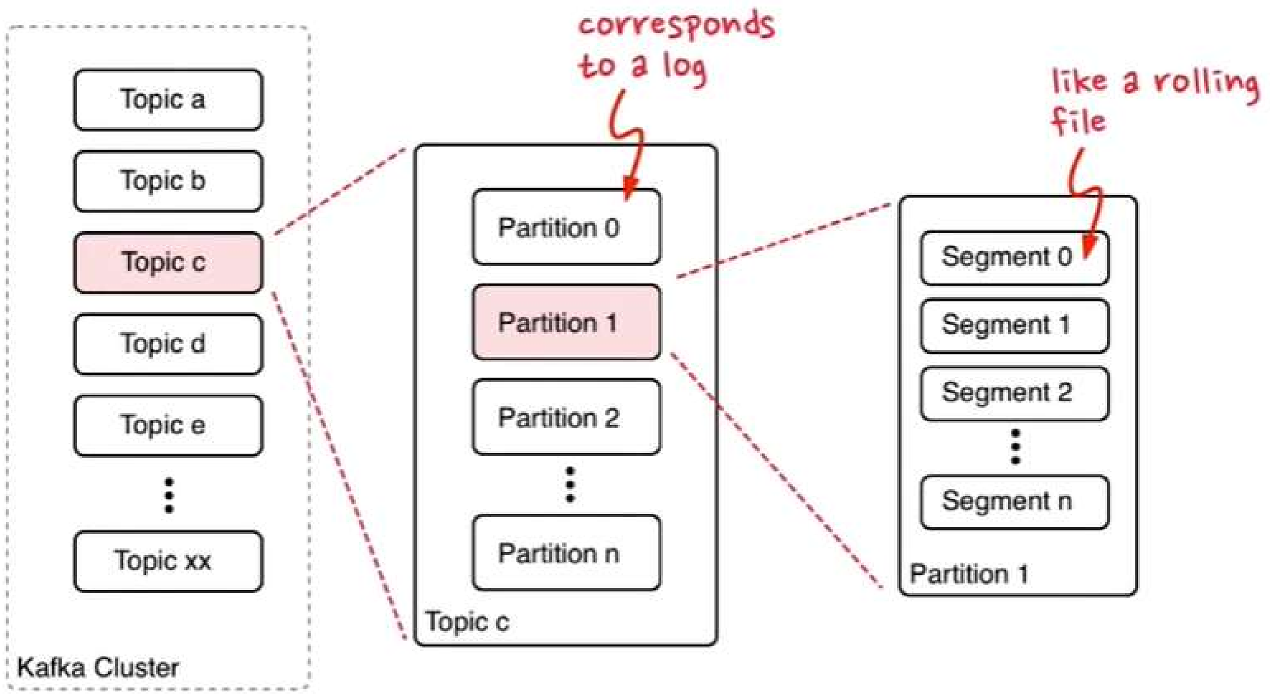
- Cluster management
- Failure detection & recovery
- Store ACLs & secrets

Topics

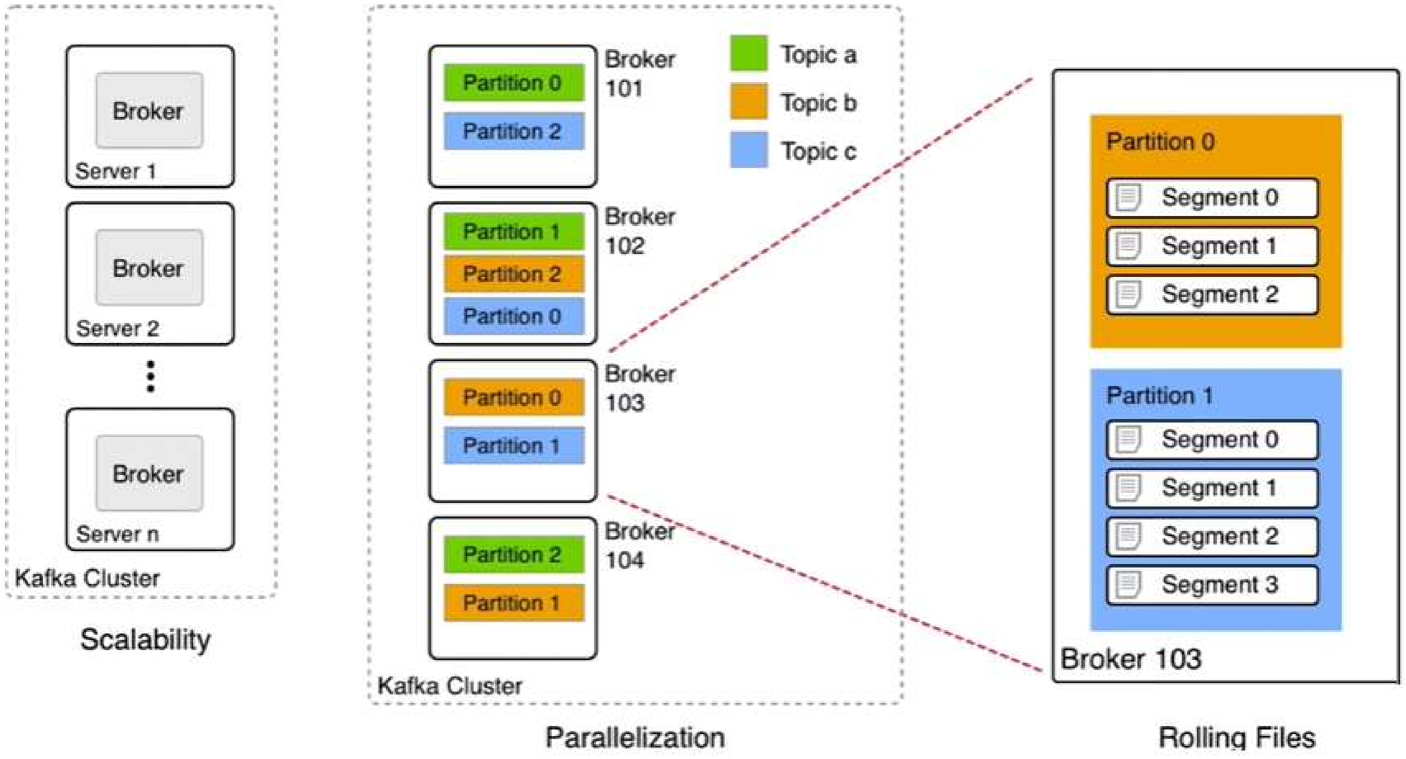
- **Topics**: Streams of "related" Messages in Kafka
  - Is a **Logical Representation**
  - **Categorizes Messages** into Groups
- Developers define Topics
- Producer ←→ Topic: N to N Relation
- Unlimited Number of Topics



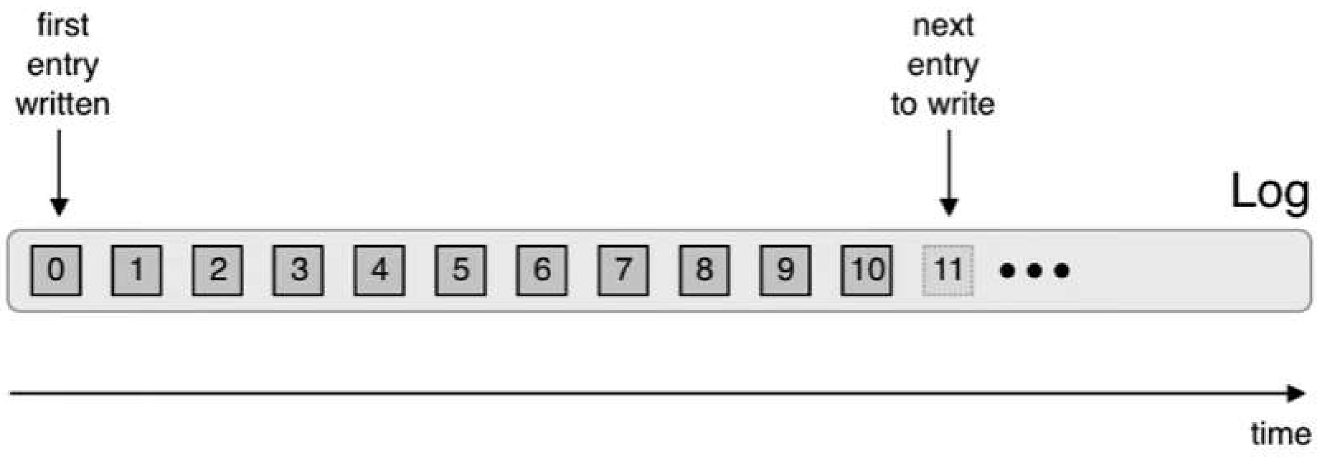
Topics, Partitions and Segments



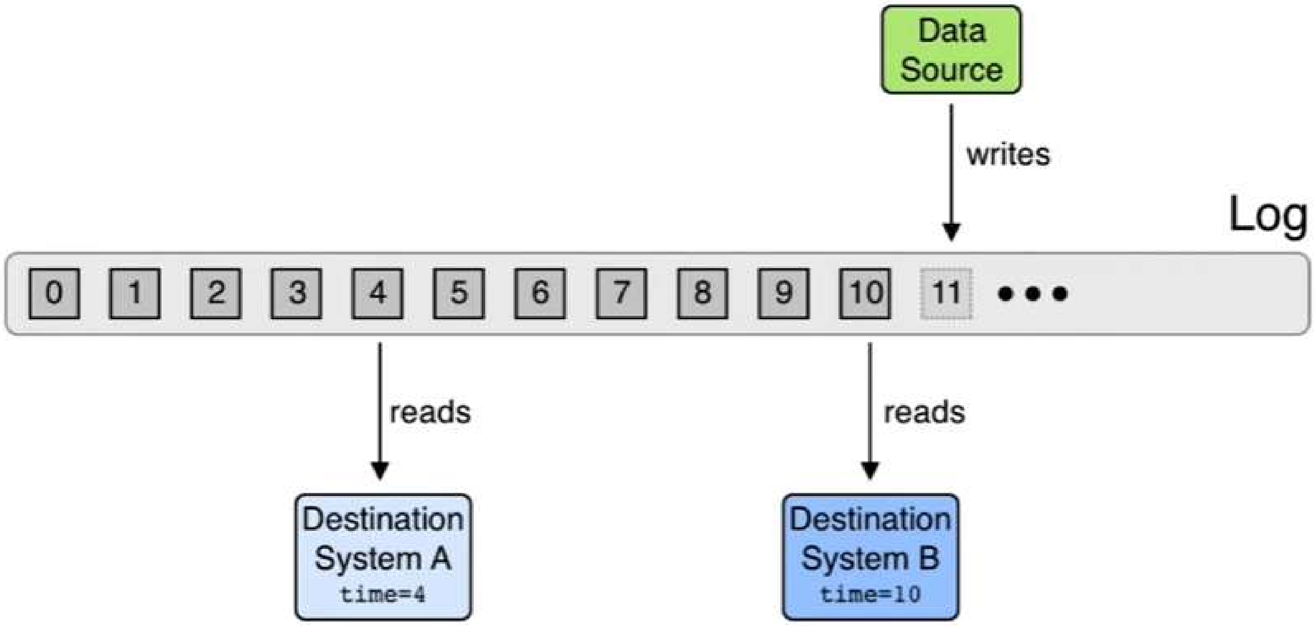
Topics, Partitions and Segments



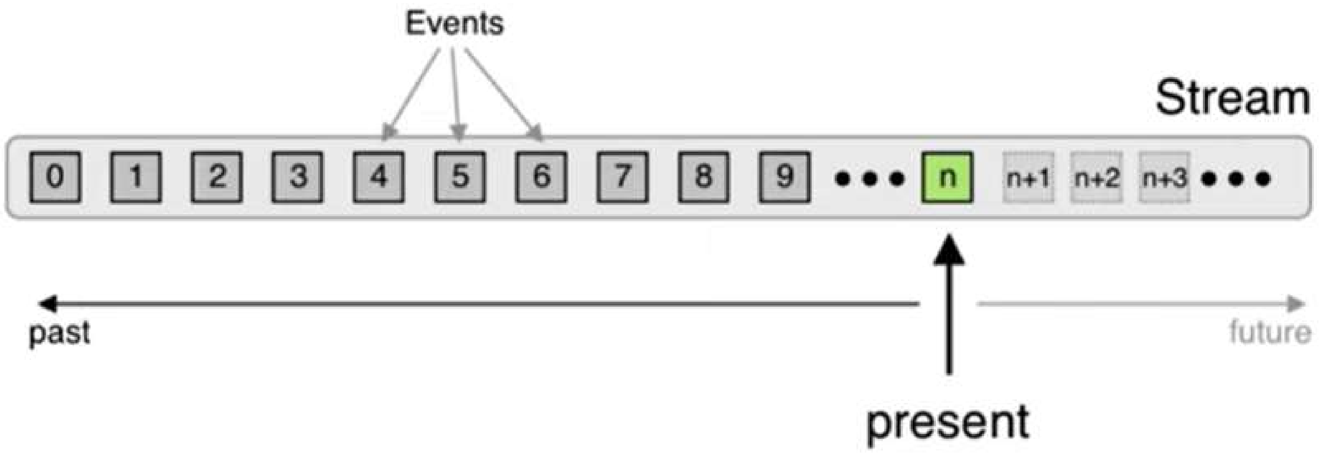
The Log


first entry written
next entry to write
Log
0
1
2
3
4
5
6
7
8
9
10
11
• • •
time

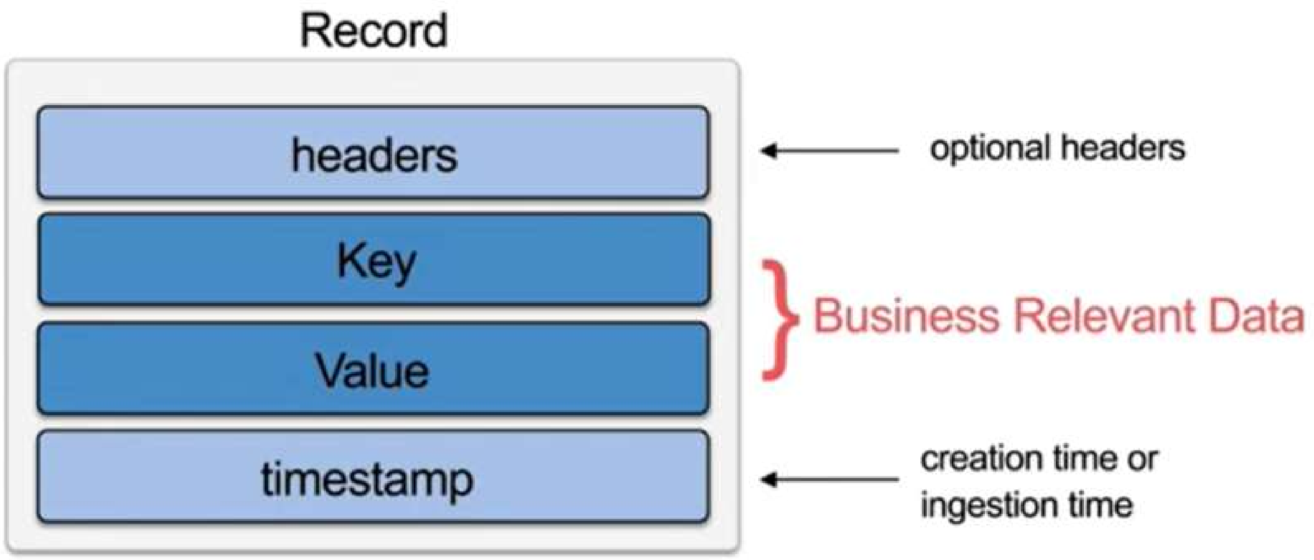
Log Structured Data Flow



The Stream



Data Elements

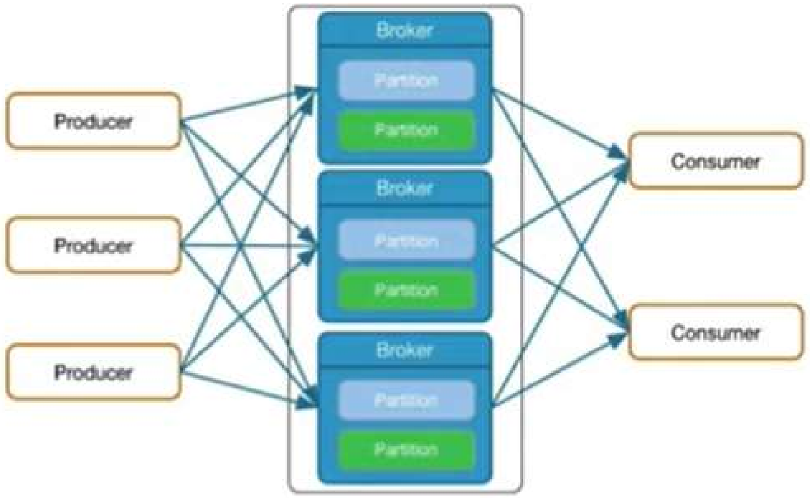


Brokers Manage Partitions

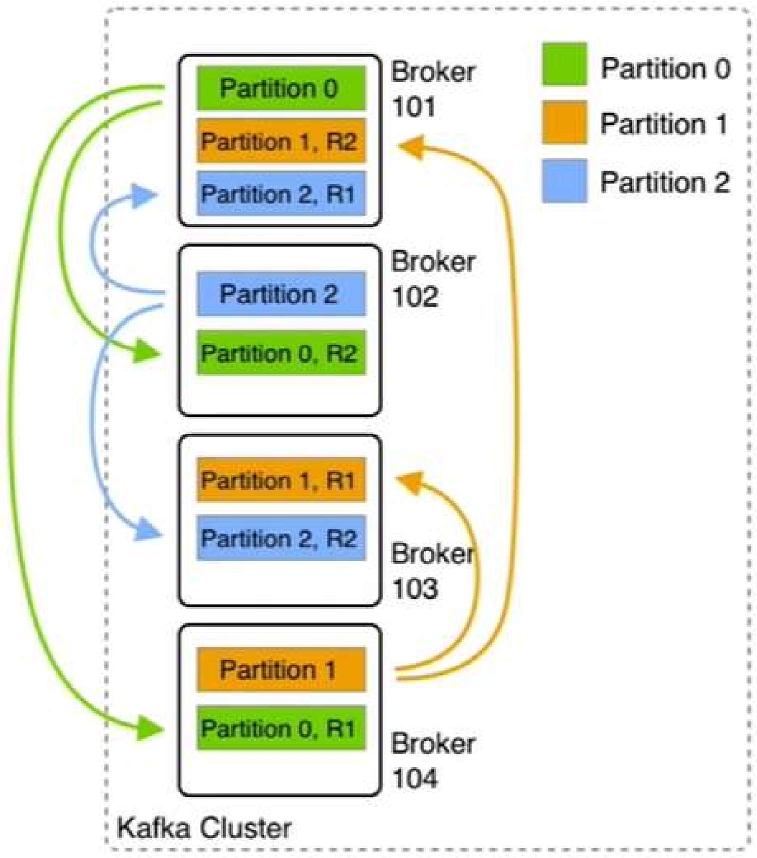
- Messages of Topic spread across Partitions
- Partitions spread across Brokers
- Each Broker handles many Partitions
- Each Partition stored on Broker's disk
- Partition: 1..n `log` files
- Each message in Log identified by *Offset*
- Configurable Retention Policy

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



Replication (Factor 3)

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 Proxy for any unsupported Language
- Command Line Producer Tool

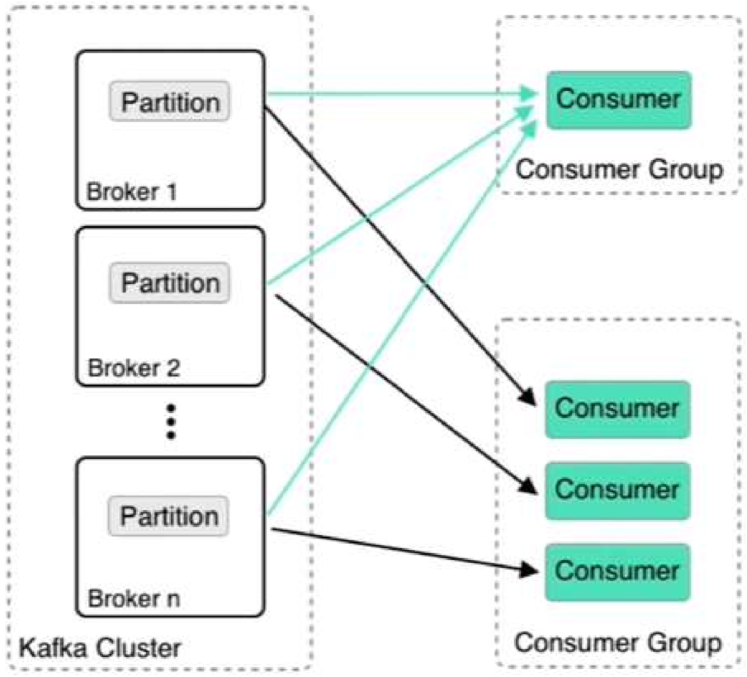
Load Balancing and Semantic Partitioning

- 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

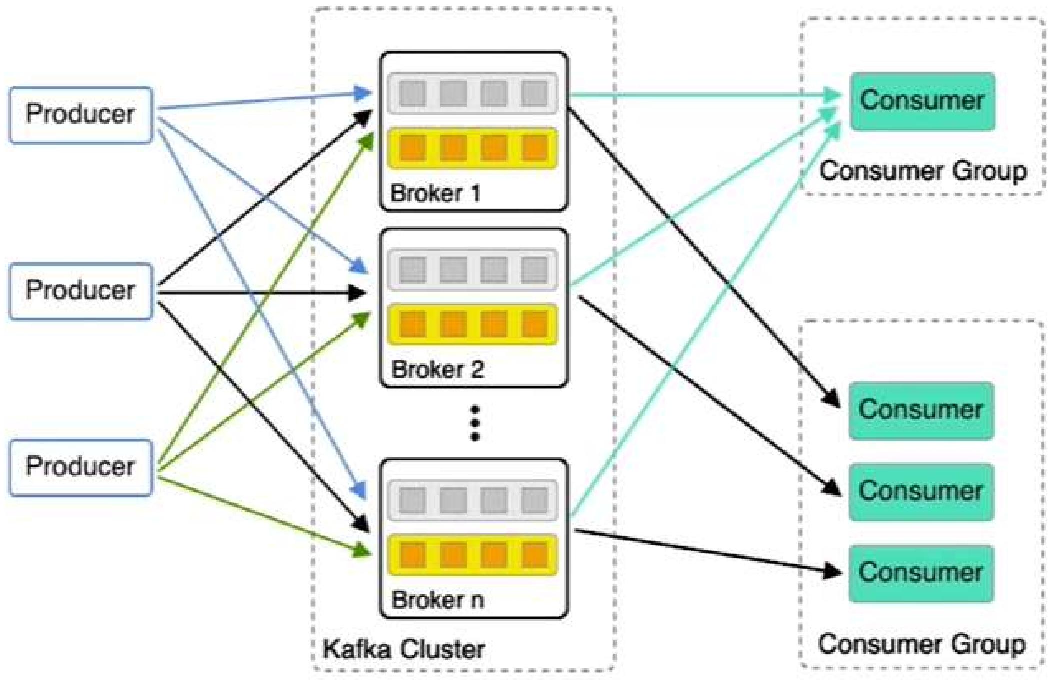
Consumer Basics

- 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

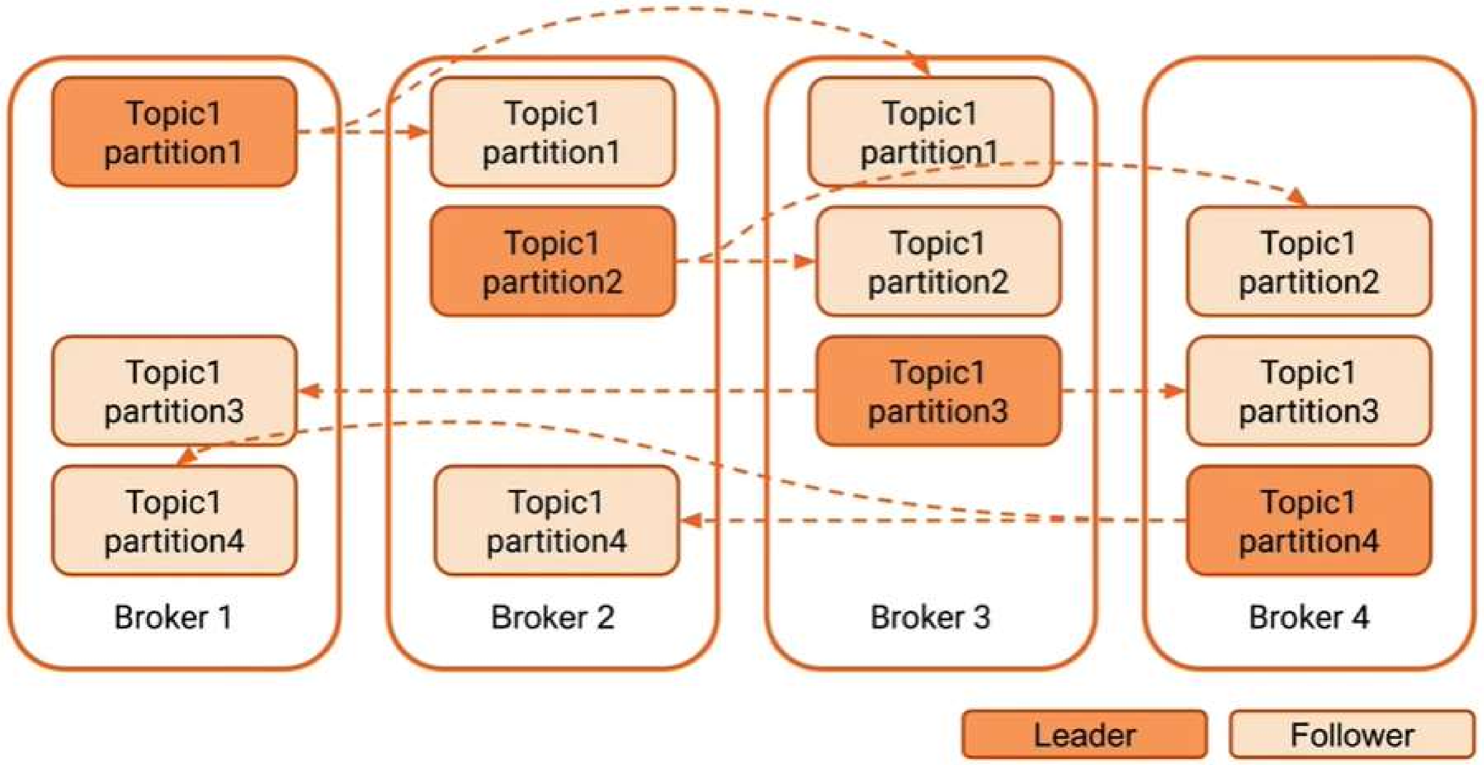
Distributed Consumption



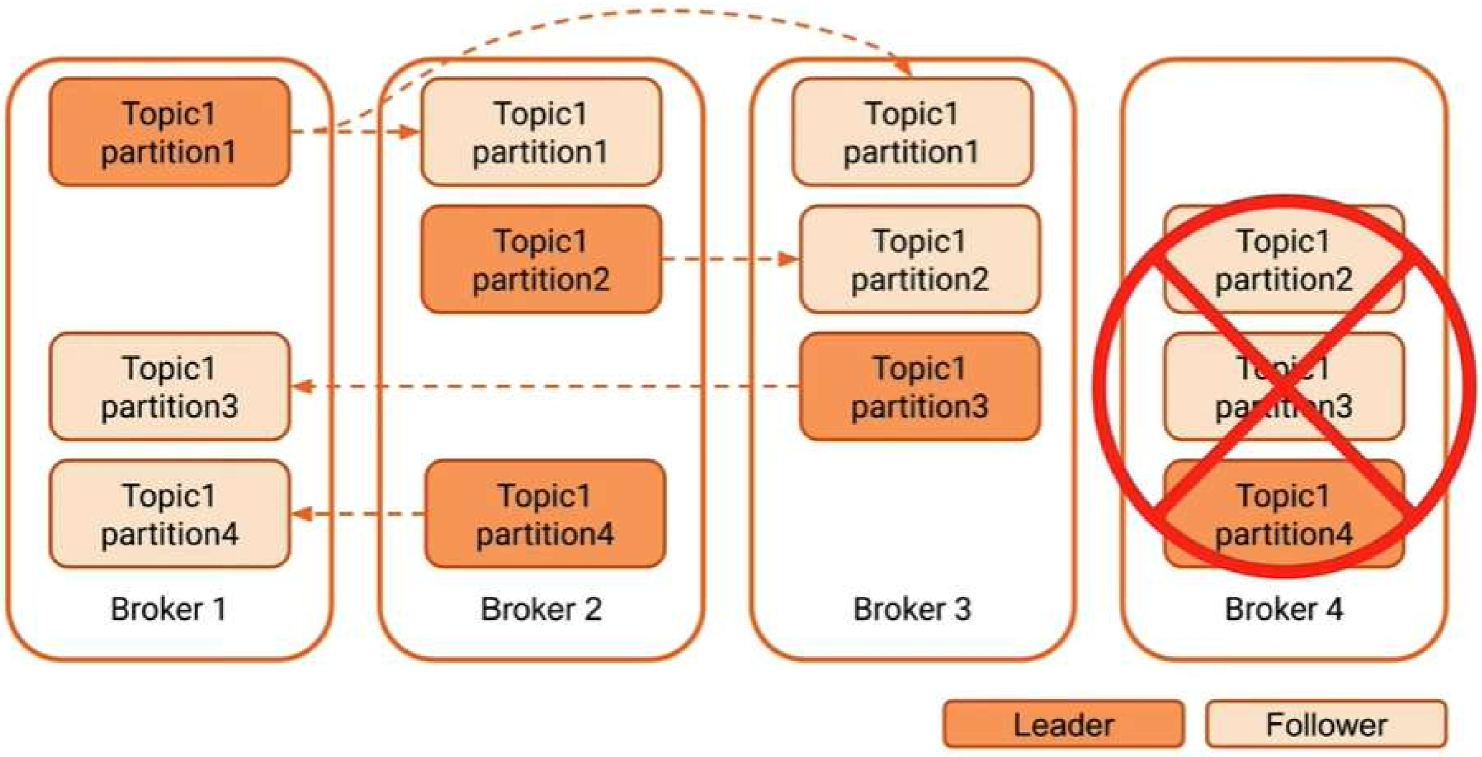
Scalable Data Pipeline



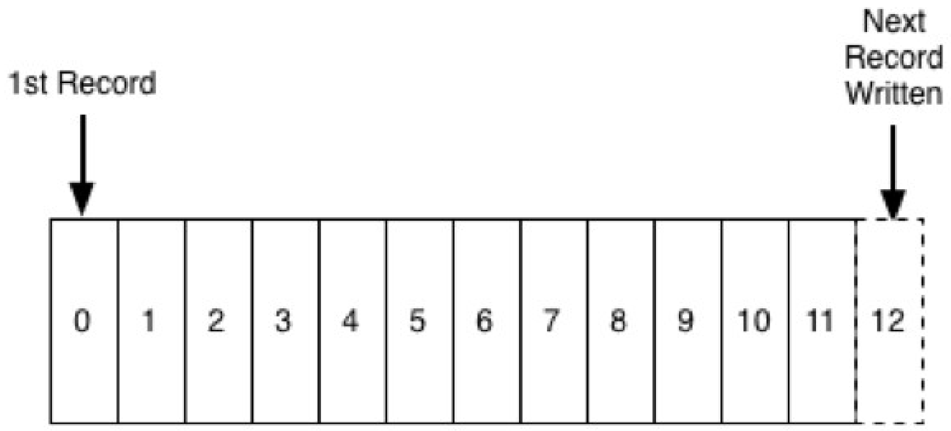
Partition Leadership & Replication



Partition Leadership & Replication



Store data in Kafka?

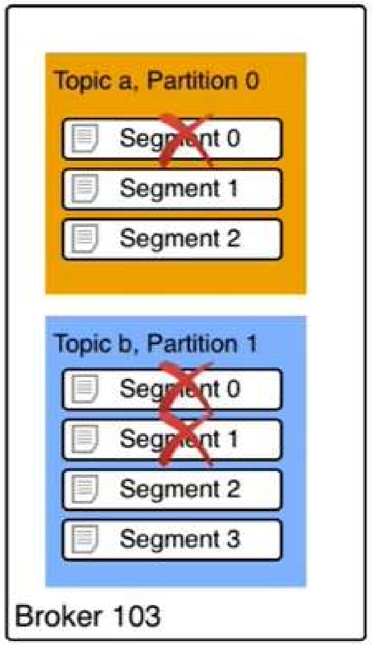


Data Retention Policy

How long do I want or can I store my data?

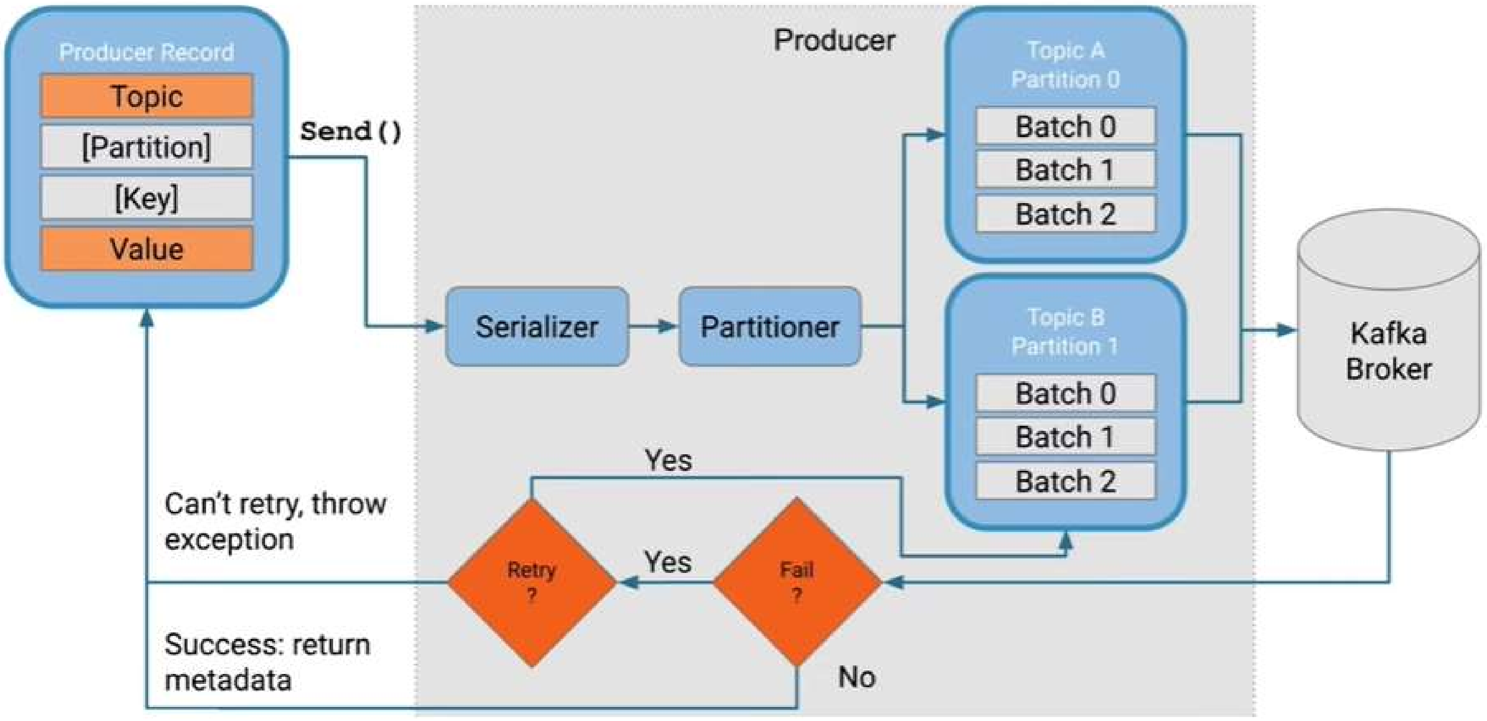
- How long (default: `1 week`)
- Set **globally** or **per topic**
- Business decision
- Cost factor
- Compliance factor → GDPR

Data purged per segment

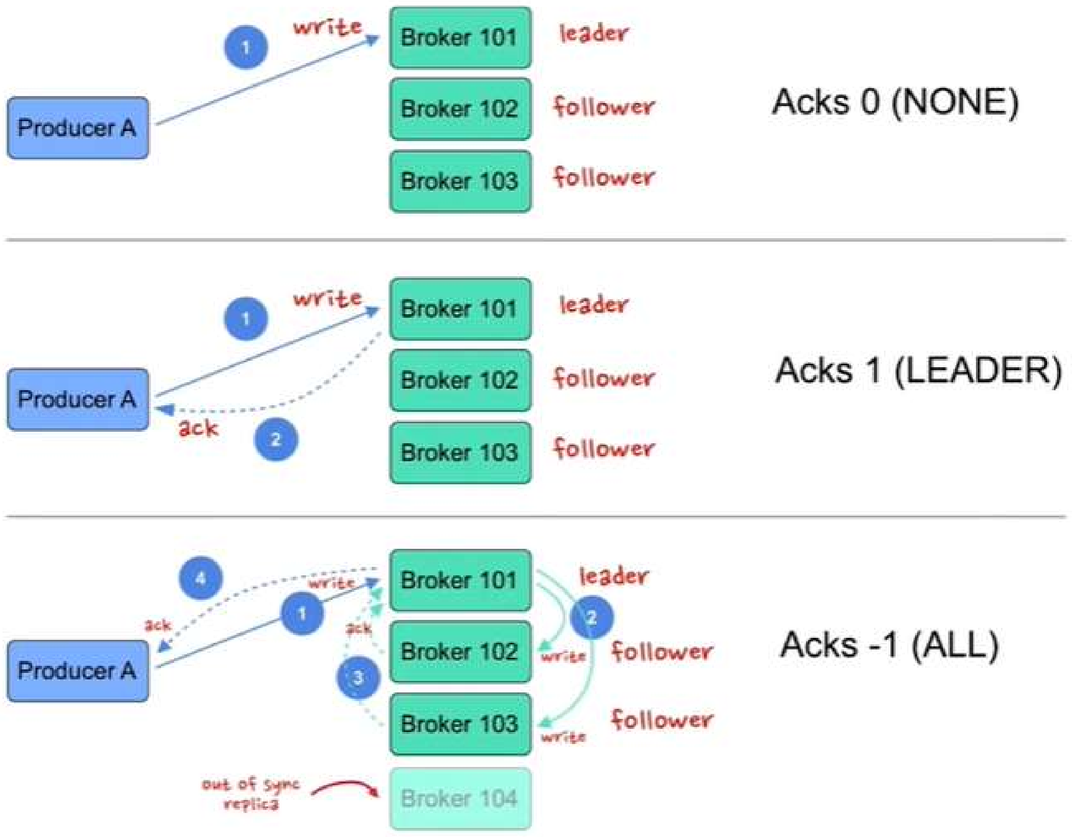


Retention Policy

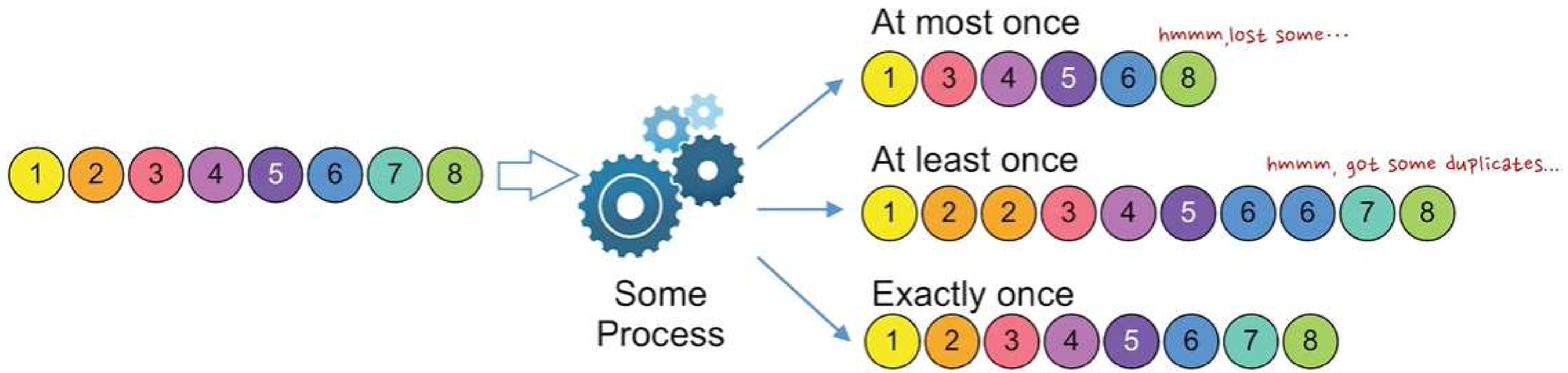
Producer Design



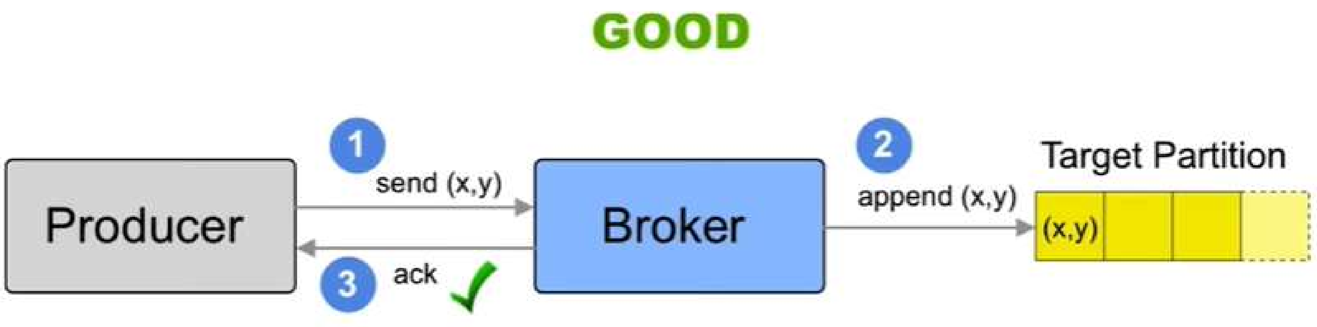
Producer Guarantees

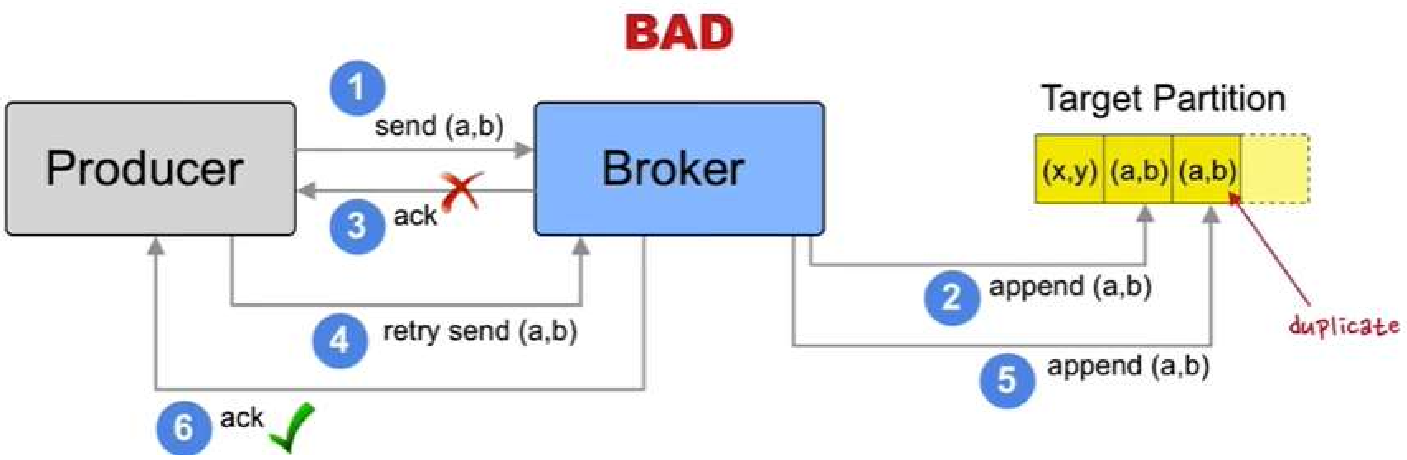


Delivery Guarantees



Idempotent Producers





Exactly Once Semantics

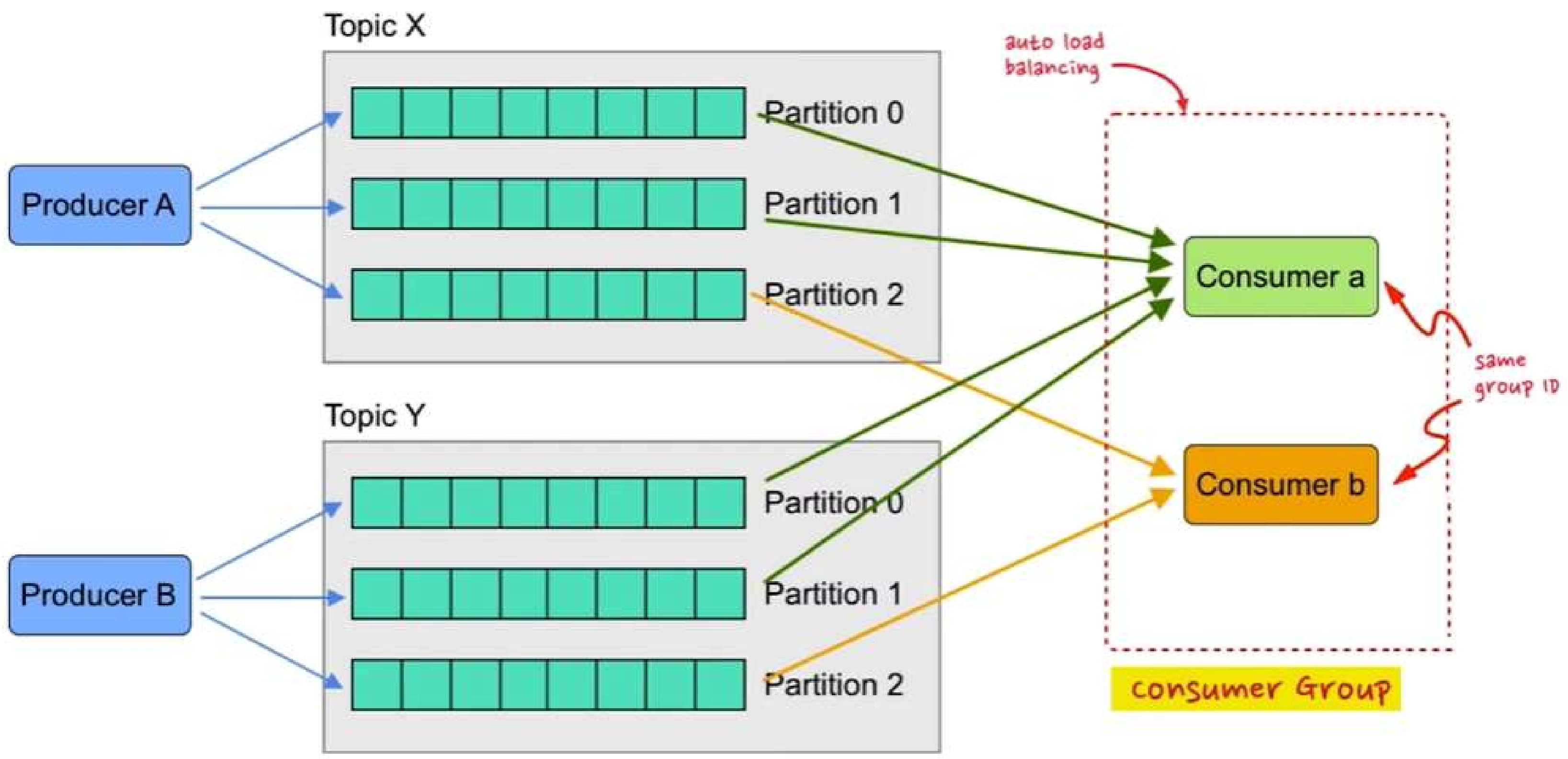
What?

- Strong **transactional guarantees** for Kafka
- Prevents clients from processing duplicate messages
- Handles failures gracefully

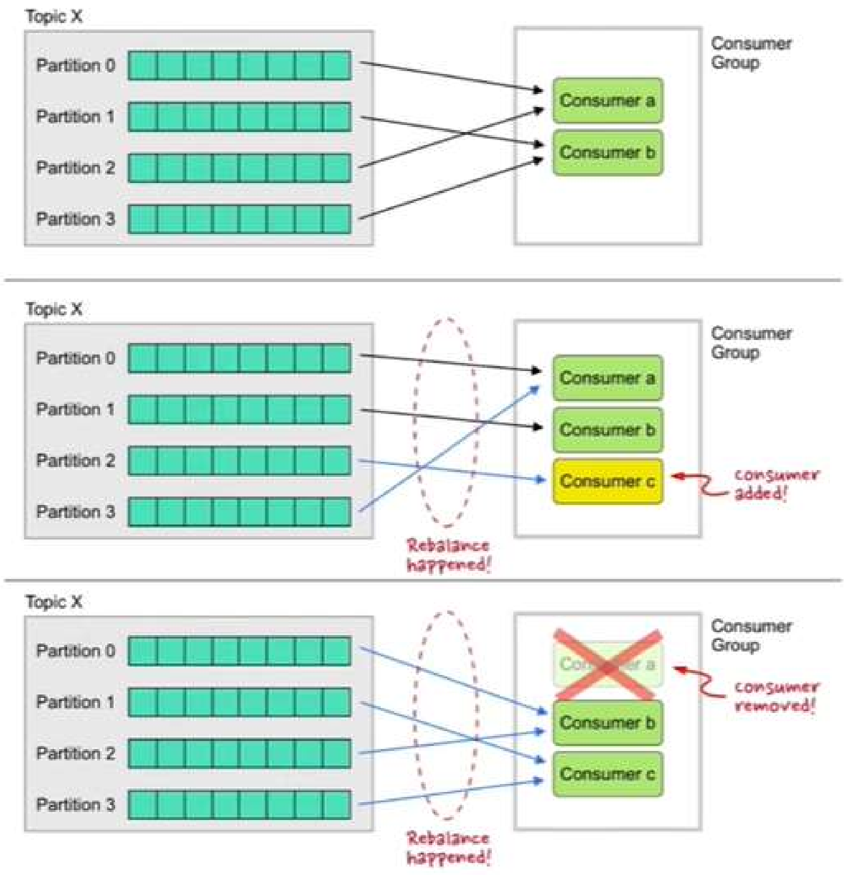
Use Cases

- Tracking ad views
- Processing financial transactions
- Stream processing

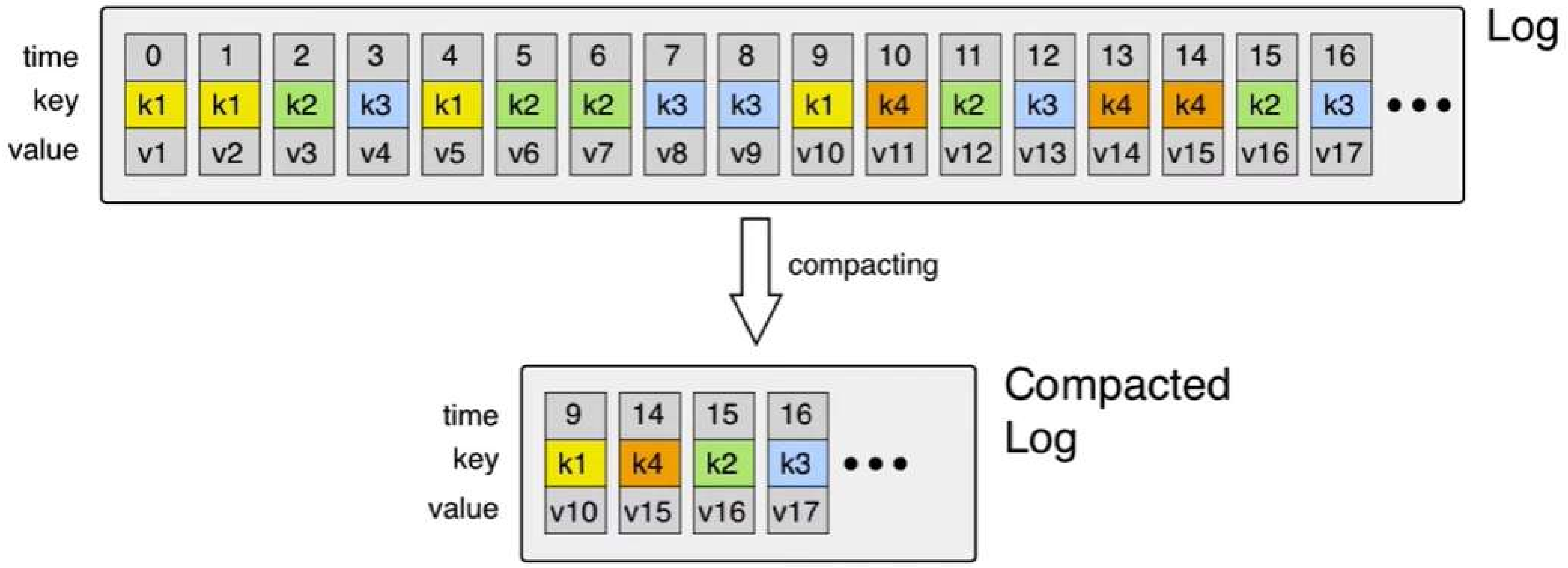
Consumer Groups



Consumer Rebalances

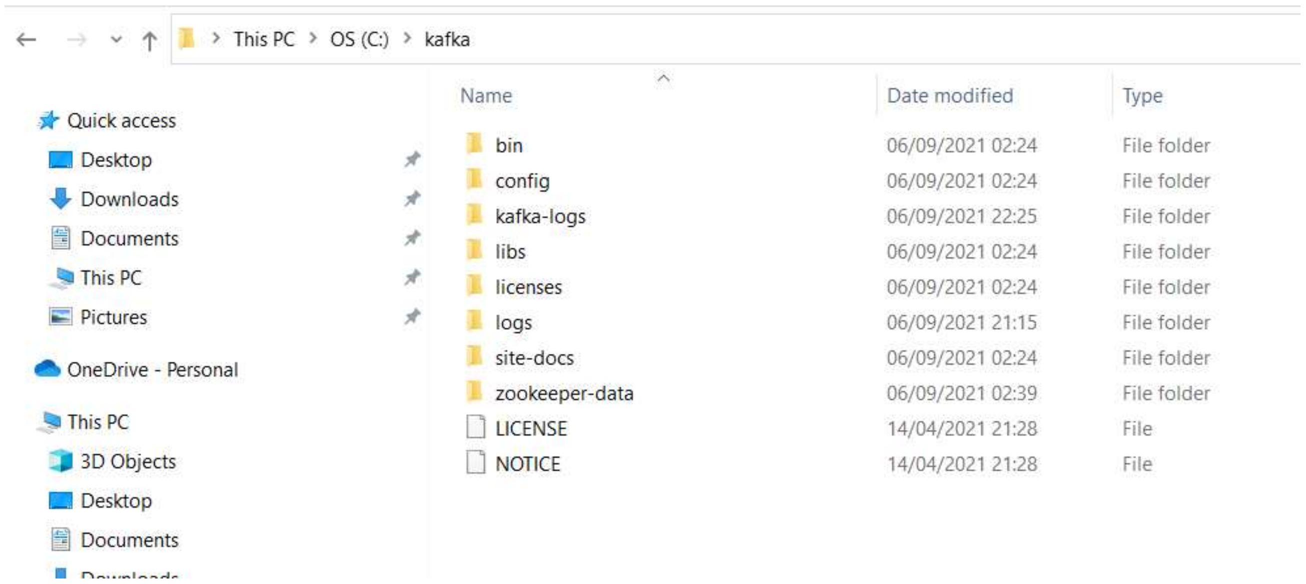


Compacted Topics



Tutorial 1: Kafka installation on Window

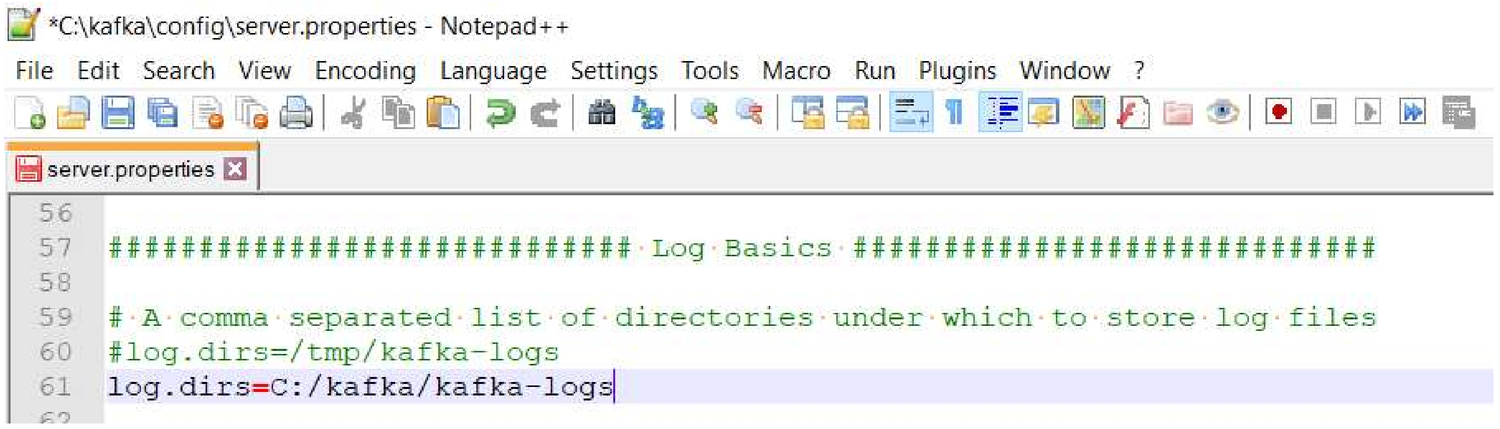
- Download Kafka from https://kafka.apache.org/
- Unzip the download file
- Rename the kafka to “kafka” and move it to C:\ drive



Name	Date modified	Type
bin	06/09/2021 02:24	File folder
config	06/09/2021 02:24	File folder
kafka-logs	06/09/2021 22:25	File folder
libs	06/09/2021 02:24	File folder
licenses	06/09/2021 02:24	File folder
logs	06/09/2021 21:15	File folder
site-docs	06/09/2021 02:24	File folder
zookeeper-data	06/09/2021 02:39	File folder
LICENSE	14/04/2021 21:28	File
NOTICE	14/04/2021 21:28	File

Kafka installation on Window

- Open C:\kafka\config\server.properties
- Change the path of log.dir


*C:\kafka\config\server.properties - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
server.properties

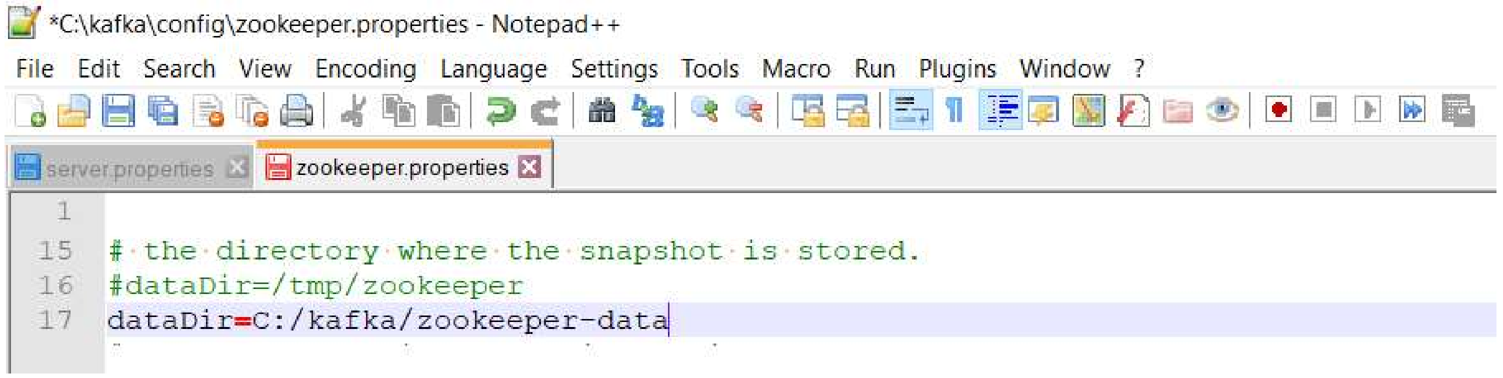
```

############################# Log Basics #############################

# A comma separated list of directories under which to store log files
#log.dirs=/tmp/kafka-logs
log.dirs=C:/kafka/kafka-logs
```

Kafka installation on Window

- Open C:\kafka\config\zookeeper.properties
- Change the path of dataDir


*C:\kafka\config\zookeeper.properties - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
server.properties
zookeeper.properties

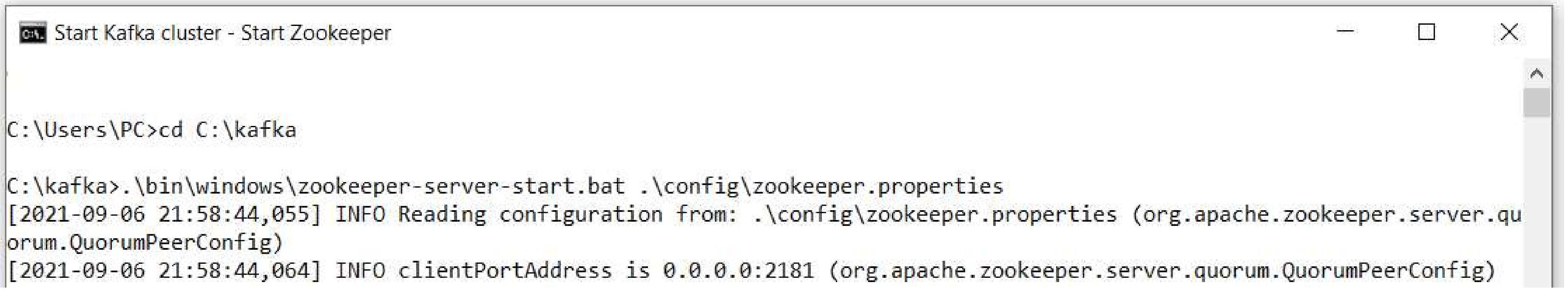
```
# the directory where the snapshot is stored.
#dataDir=/tmp/zookeeper
dataDir=C:/kafka/zookeeper-data
```

- By default Apache Kafka will run on port 9092 and Apache Zookeeper will run on port 2181.

Tutorial 2: Run Apache Kafka on Windows

- Start the Kafka cluster
    - Run the following command to start ZooKeeper:

```
cd C:\kafka\
.\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties
```



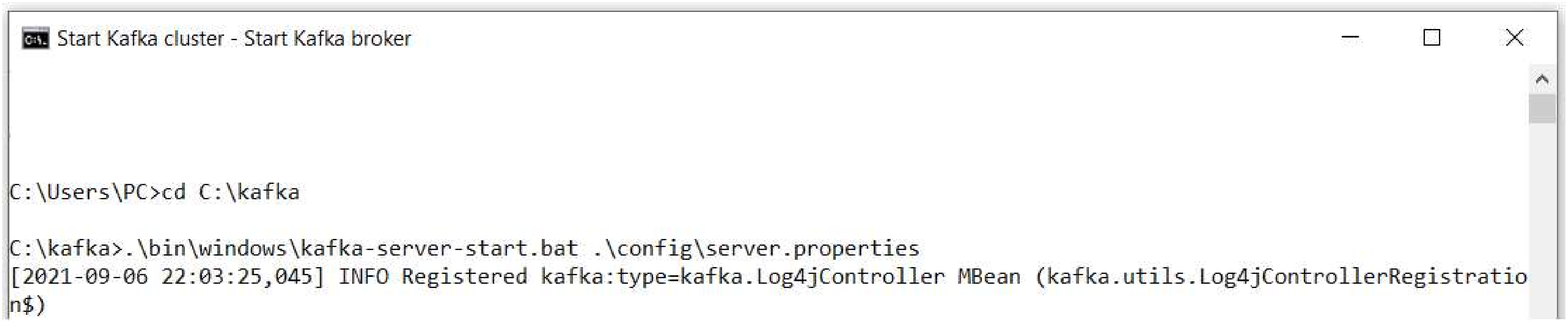
```
C:\Users\PC>cd C:\kafka

C:\kafka>.\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties
[2021-09-06 21:58:44,055] INFO Reading configuration from: .\config\zookeeper.properties (org.apache.zookeeper.server.qu
orum.QuorumPeerConfig)
[2021-09-06 21:58:44,064] INFO clientPortAddress is 0.0.0.0:2181 (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
```

Run Apache Kafka on Windows

- Start the Kafka cluster
  - Run the following command to start the Kafka broker:

```
cd C:\kafka\
.\bin\windows\kafka-server-start.bat .\config\server.properties
```



```
C:\Users\PC>cd C:\kafka

C:\kafka>.\bin\windows\kafka-server-start.bat .\config\server.properties
[2021-09-06 22:03:25,045] INFO Registered kafka:type=kafka.Log4jController MBean (kafka.utils.Log4jControllerRegistratio
n$)
```

Run Apache Kafka on Windows

- Produce and consume some messages
    - Run the kafka-topics command to create a Kafka topic named TestTopic

```
.\bin\windows\kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic TestTopic
```

- Let’s create another topic named NewTopic

```
.\bin\windows\kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic NewTopic
```

- Let’s show list of created topics

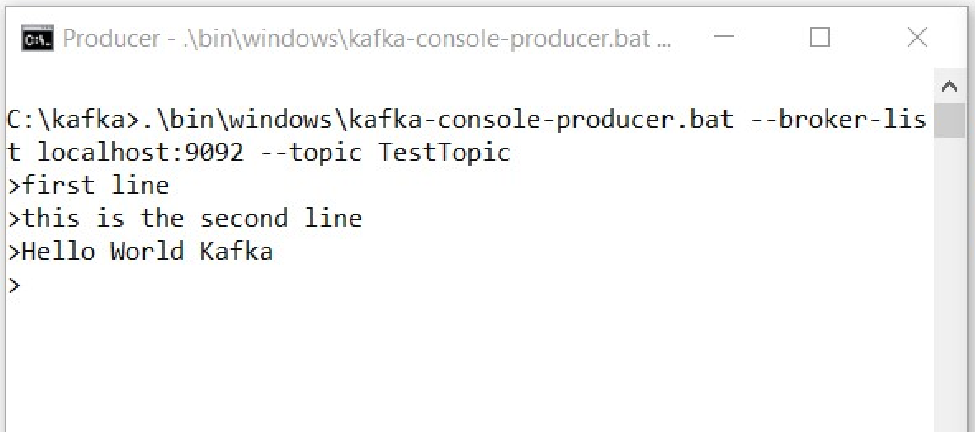
```
.\bin\windows\kafka-topics.bat --list --zookeeper localhost:2181
```

Run Apache Kafka on Windows

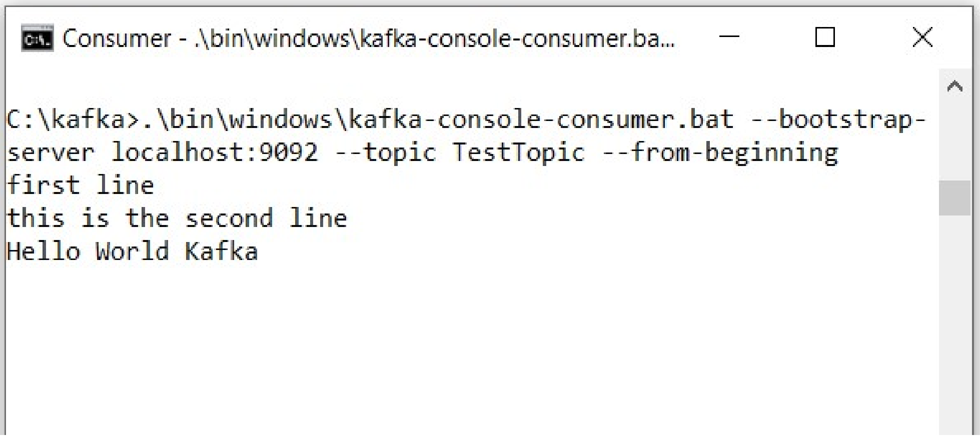
- Produce and consume some messages
  - Run the producer and consumer on separate Command Prompt:

.\bin\windows\kafka-console-producer.bat --broker-list localhost:9092 --topic TestTopic

.\bin\windows\kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic TestTopic --from-beginning



```
C:\kafka>.\bin\windows\kafka-console-producer.bat --broker-lis
t localhost:9092 --topic TestTopic
>first line
>this is the second line
>Hello World Kafka
>
```



```
C:\kafka>.\bin\windows\kafka-console-consumer.bat --bootstrap-
server localhost:9092 --topic TestTopic --from-beginning
first line
this is the second line
Hello World Kafka
```