



# Apache Kafka Architectures and Fundamentals

Henrik Janzon, Solutions Engineer



# Learning Objectives

After this module you will be able to:



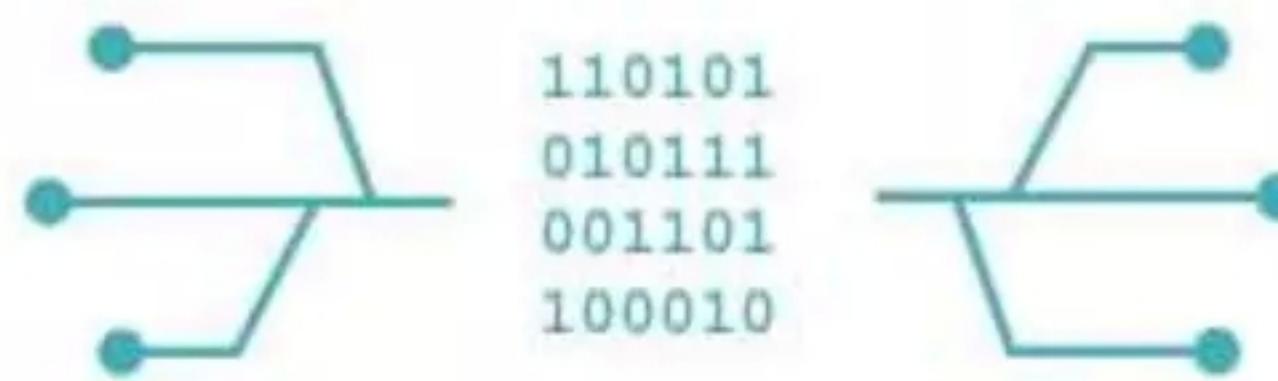
- Give a high level description of the programming logic in Kafka producer and consumer clients
- Explain how EOS works to an interested lay person
- List the means with which Kafka provides durability and HA
- Illustrate on a high level, how you can secure your Kafka cluster





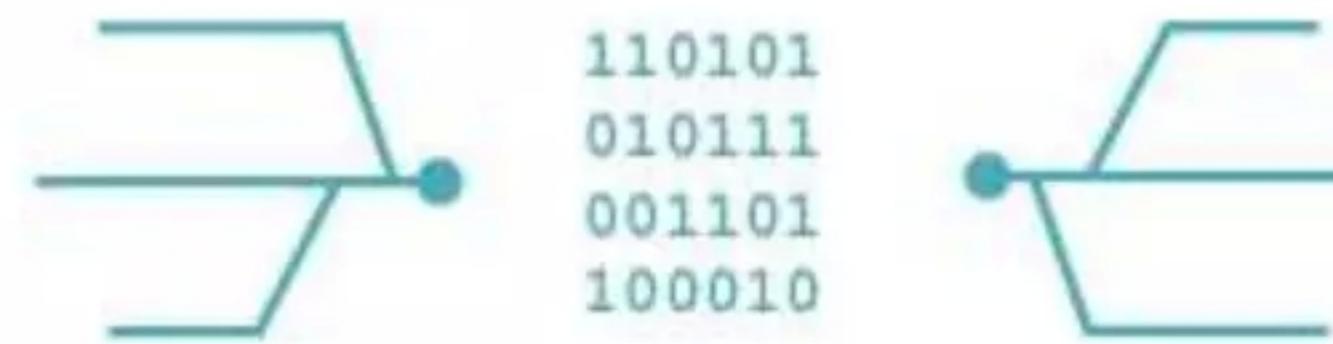
# Apache Kafka is a Distributed Event Streaming Platform

Publish and subscribe to streams of events



Similar to a message queue or enterprise messaging system

Store streams of events



In a fault tolerant way

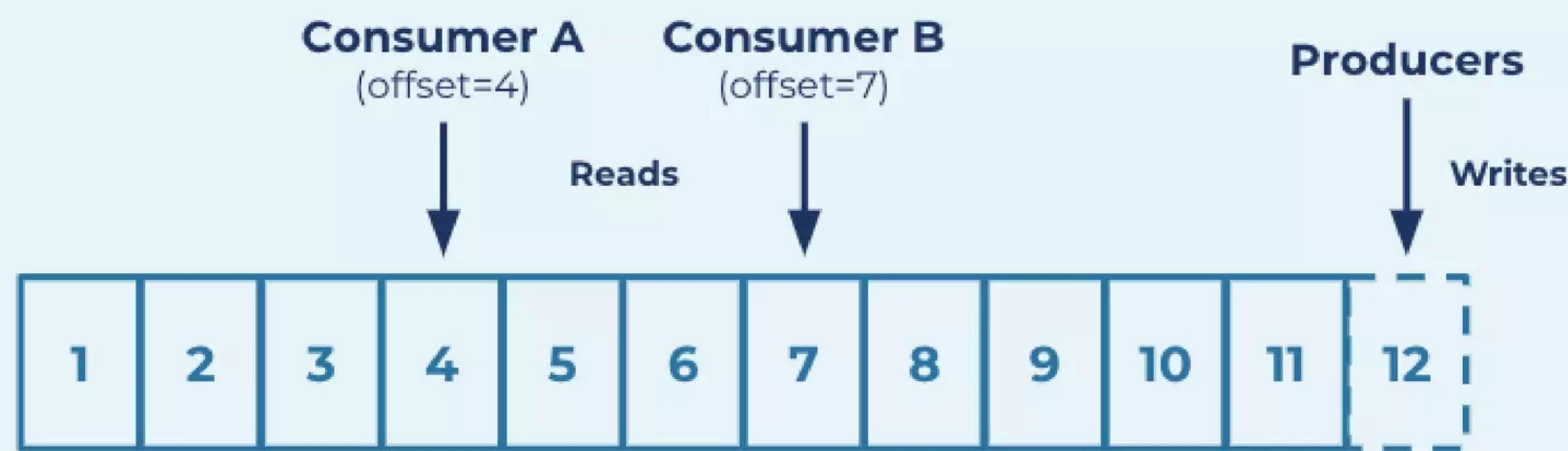
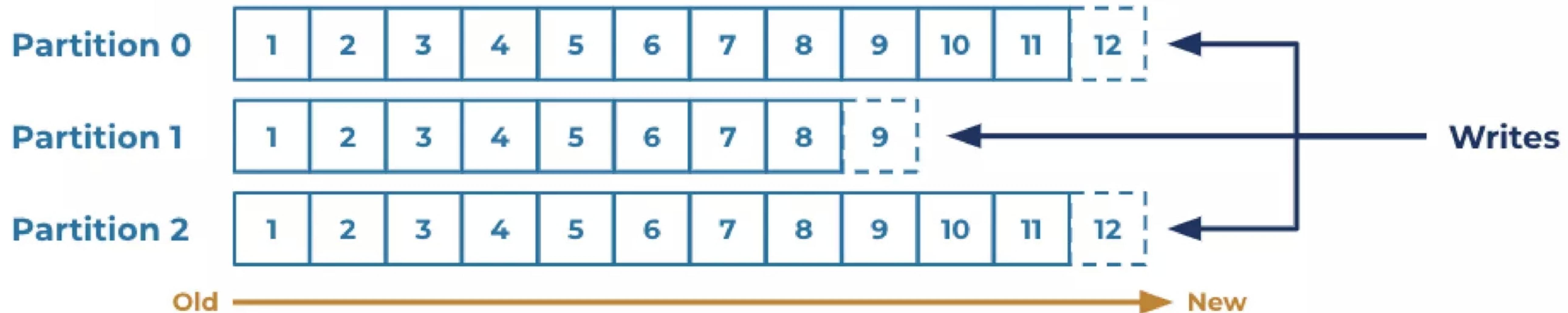
Process streams of events



In real time, as they occur

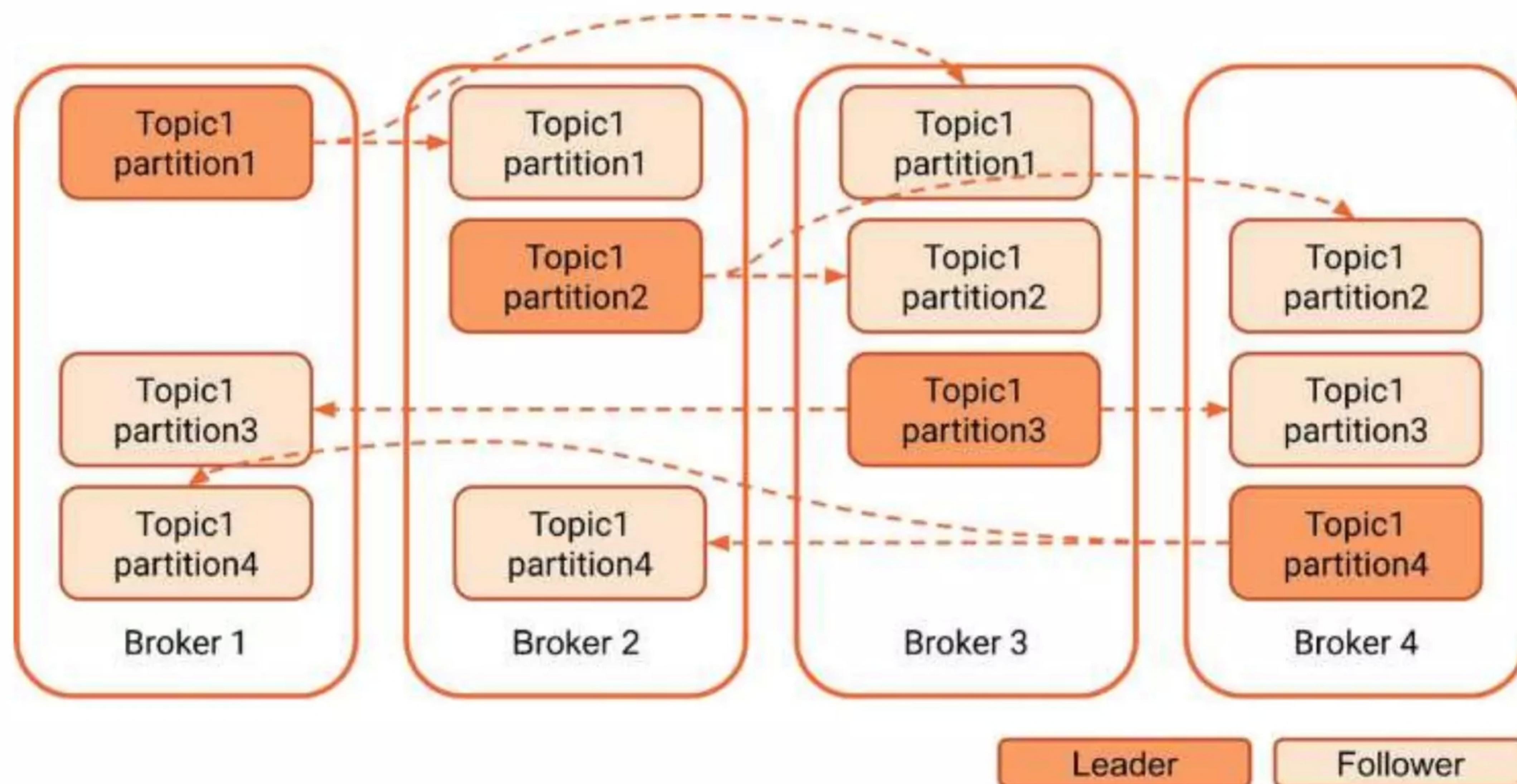


# Anatomy of a Kafka Topic



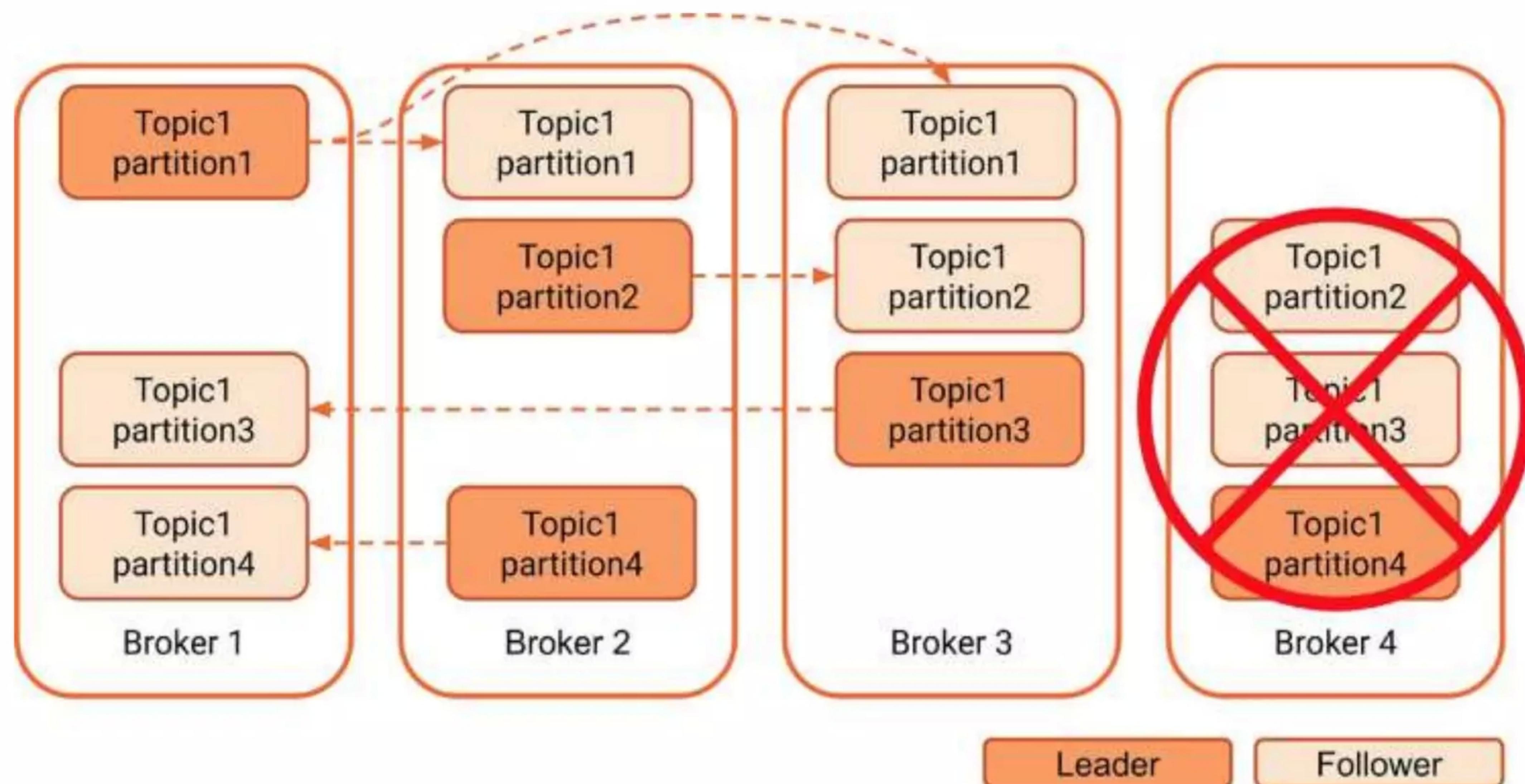


# Partition Leadership & Replication





# Partition Leadership & Replication

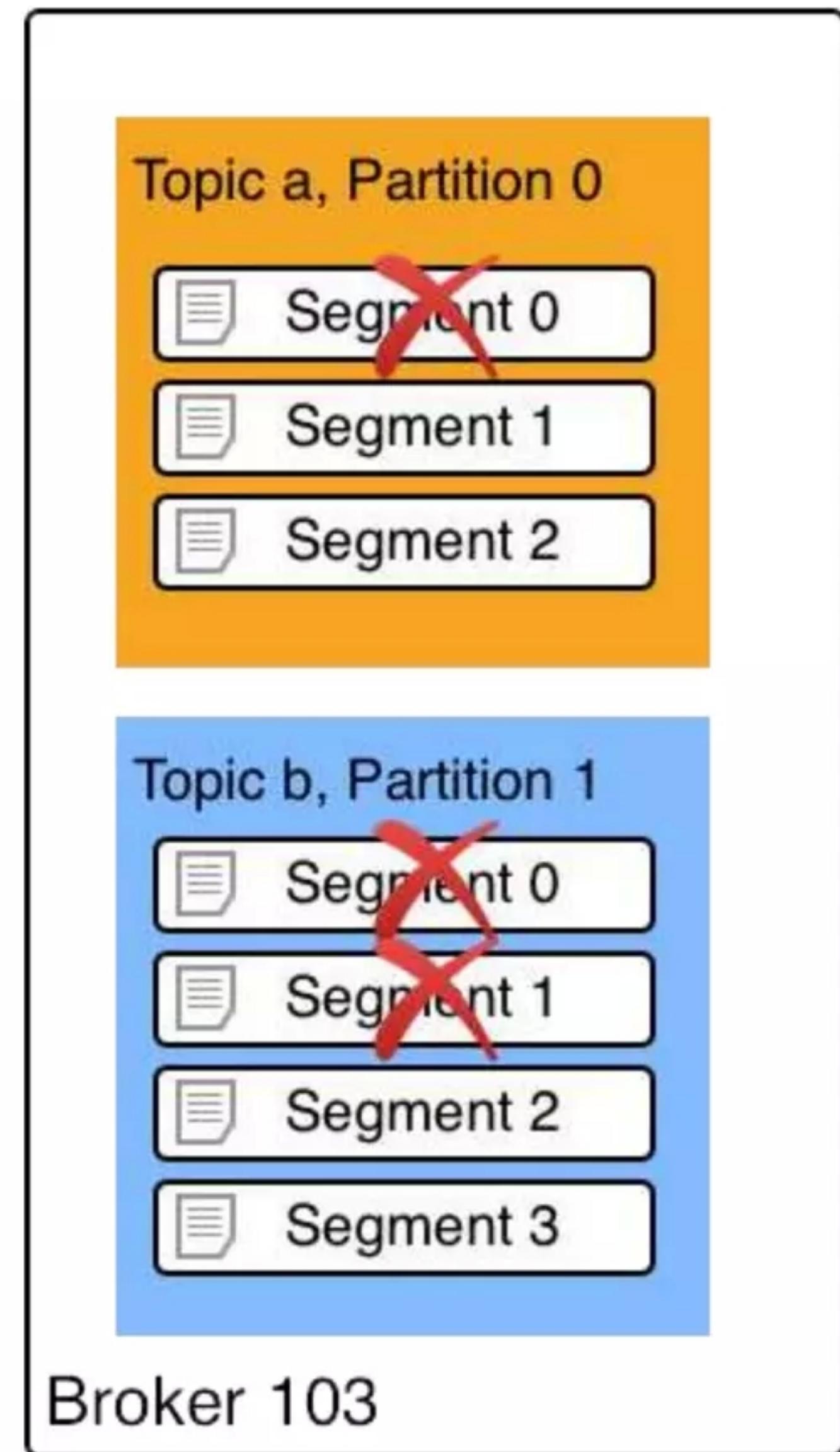




# Data Retention Policy

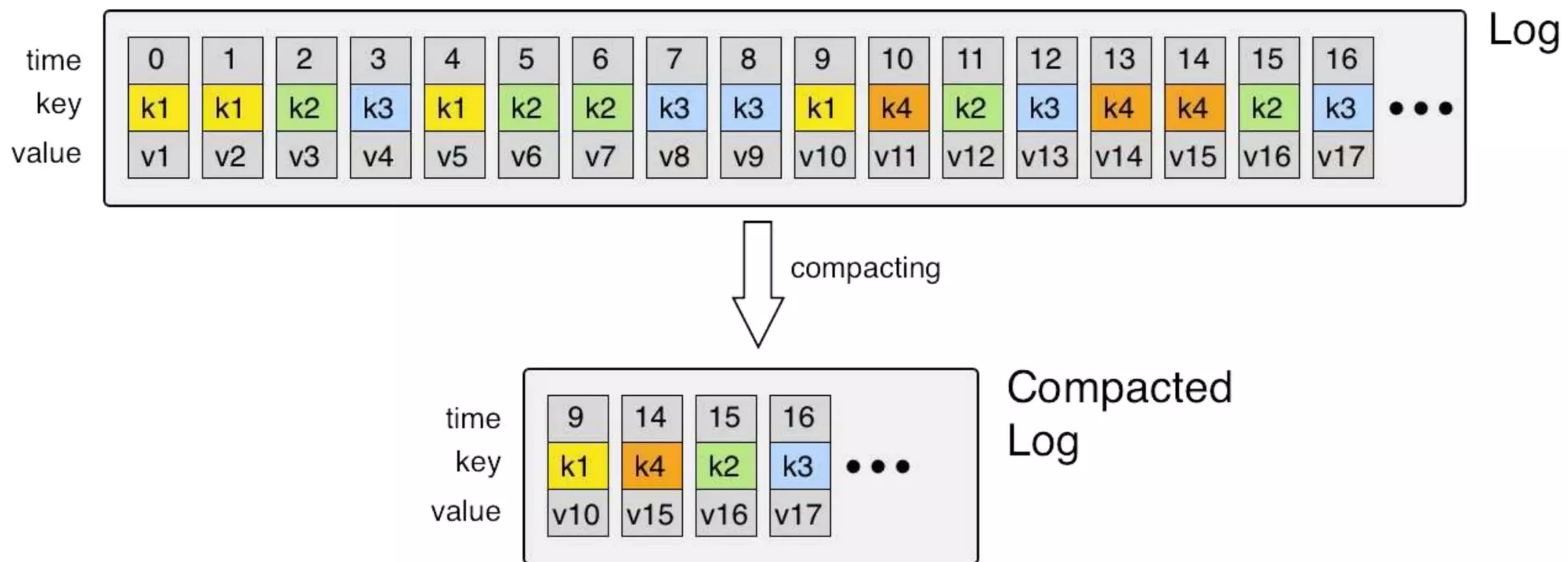
How long do I want or can I store my data?

- How long (default: one week)
- Set **globally** or **per topic**
- Business decision
- Cost factor
- Compliance factor (e.g., GDPR)





# Compacted Topics



# Development: A Basic Producer in Java

```
BasicProducer.java ✘
1 package clients;
2
3 import java.util.Properties;
4 import org.apache.kafka.clients.producer.KafkaProducer;
5 import org.apache.kafka.clients.producer.ProducerRecord;
6
7 public class BasicProducer {
8     public static void main(String[] args) {
9         System.out.println("*** Starting Basic Producer ***");
10
11     Properties settings = new Properties();
12     settings.put("client.id", "basic-producer-v0.1.0");
13     settings.put("bootstrap.servers", "kafka-1:9092,kafka-2:9092");
14     settings.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");
15     settings.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");
16
17     final KafkaProducer<String, String> producer = new KafkaProducer<>(settings);
18
19     Runtime.getRuntime().addShutdownHook(new Thread(() -> {
20         System.out.println("### Stopping Basic Producer ###");
21         producer.close();
22     }));
23
24     final String topic = "hello_world_topic";
25     for(int i=1; i<=5; i++){
26         final String key = "key-" + i;
27         final String value = "value-" + i;
28         final ProducerRecord<String, String> record = new ProducerRecord<>(topic, key, value);
29         producer.send(record);
30     }
31 }
32 }
```

configuration

create producer

shutdown behaviour

sending data



# Development: A Basic Consumer in .NET/C#

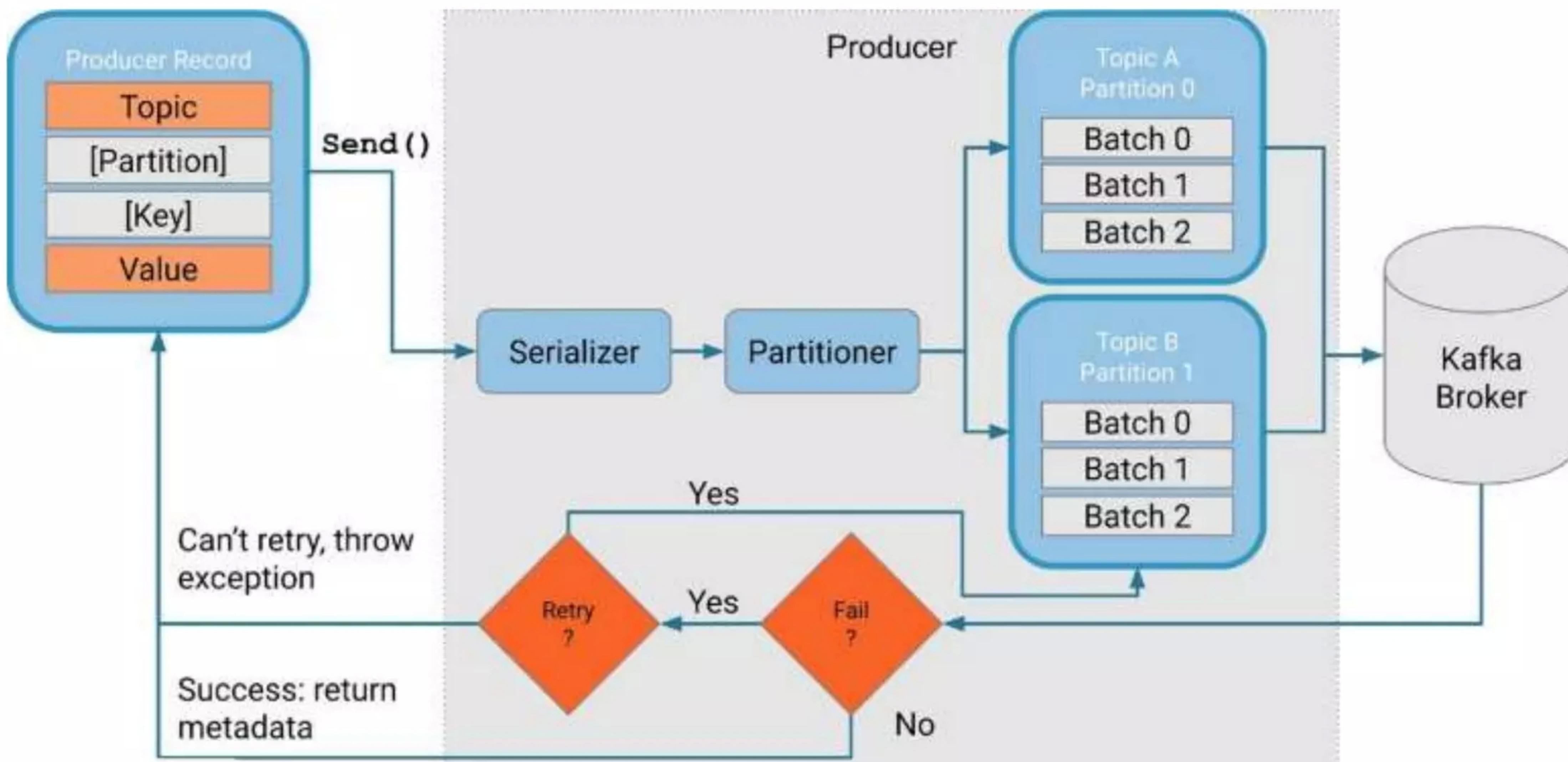
```
8  namespace consumer_net {
9      0 references
10     class Program {
11         0 references
12         static void Main (string[] args) {
13             Console.WriteLine ("Starting Consumer!");
14             var config = new Dictionary<string, object> {
15                 { "group.id", "dotnet-consumer-group" },
16                 { "bootstrap.servers", "kafka-1:9092" },
17                 { "auto.commit.interval.ms", 5000 },
18                 { "auto.offset.reset", "earliest" }
19             };
20
21             var deserializer = new StringDeserializer (Encoding.UTF8);
22             using (var consumer = new Consumer<string, string> (config, deserializer, deserializer)) {
23                 consumer.OnMessage += (_, msg) =>
24                     Console.WriteLine ($"Read '{msg.Key}', '{msg.Value}' from: {msg.TopicPartitionOffset}");
25
26                 consumer.OnError += (_, error) =>
27                     Console.WriteLine ($"Error: {error}");
28
29                 consumer.OnConsumeError += (_, msg) =>
30                     Console.WriteLine ($"Consume error ({msg.TopicPartitionOffset}): {msg.Error}");
31
32                 consumer.Subscribe ("hello_world_topic");
33
34                 while (true) {
35                     consumer.Poll (TimeSpan.FromMilliseconds (100));
36                 }
37             }
38 }
```

Annotations on the code:

- A red box highlights the configuration section (lines 12-18), with a red arrow pointing to it labeled "configuration".
- A yellow box highlights the message handling logic (consumer.OnMessage and consumer.OnError handlers), with a red arrow pointing to it labeled "message handling".
- A green box highlights the error handling logic (consumer.OnConsumeError), with a red arrow pointing to it labeled "error handling".
- A blue box highlights the poll loop (while (true) { consumer.Poll (TimeSpan.FromMilliseconds (100)); }), with a red arrow pointing to it labeled "polling data".
- A red box highlights the "subscribing to topic" line (consumer.Subscribe ("hello\_world\_topic")), with a red arrow pointing to it labeled "subscribing to topic".

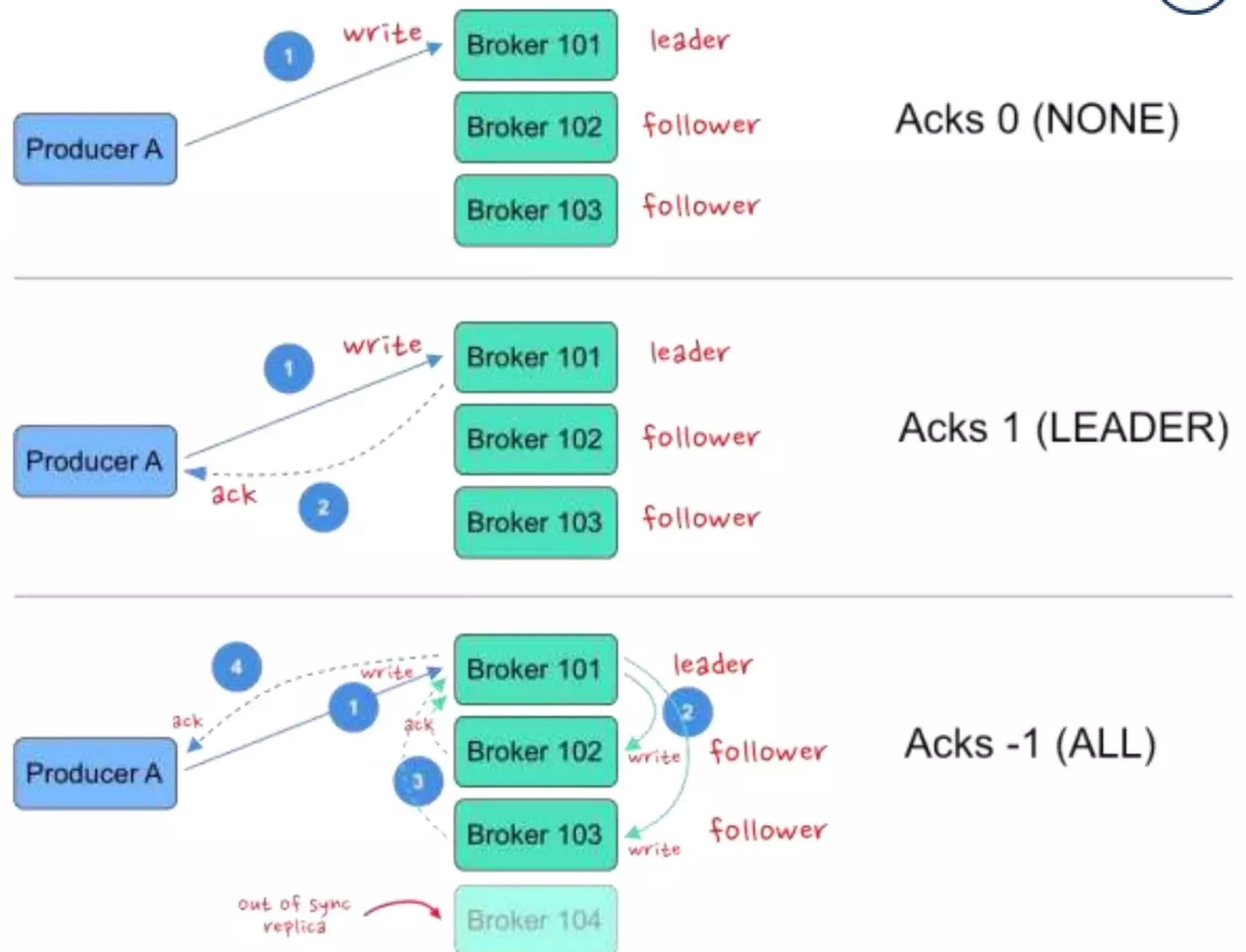


# Producer Design



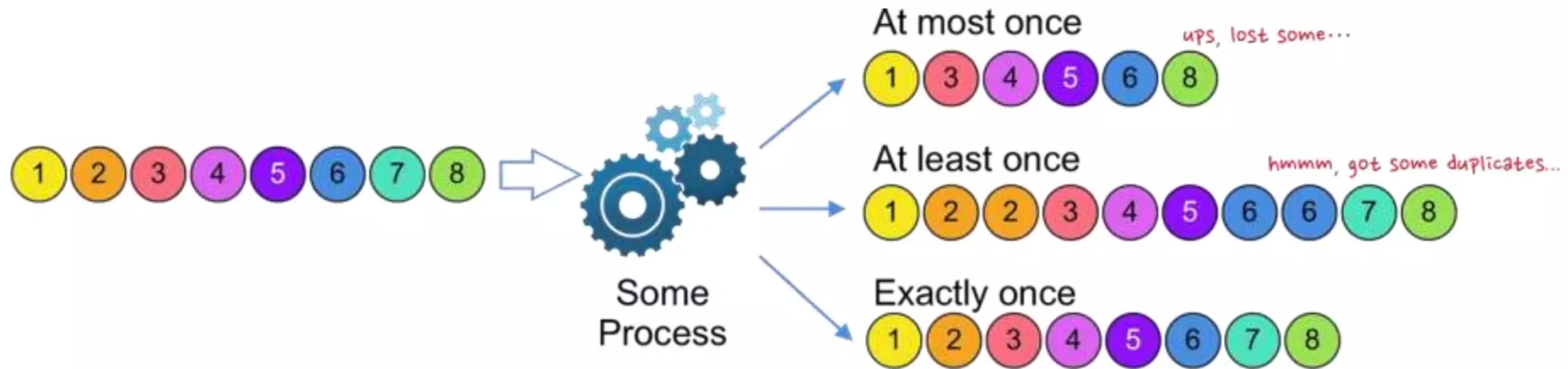


# Producer Guarantees





# Delivery Guarantees



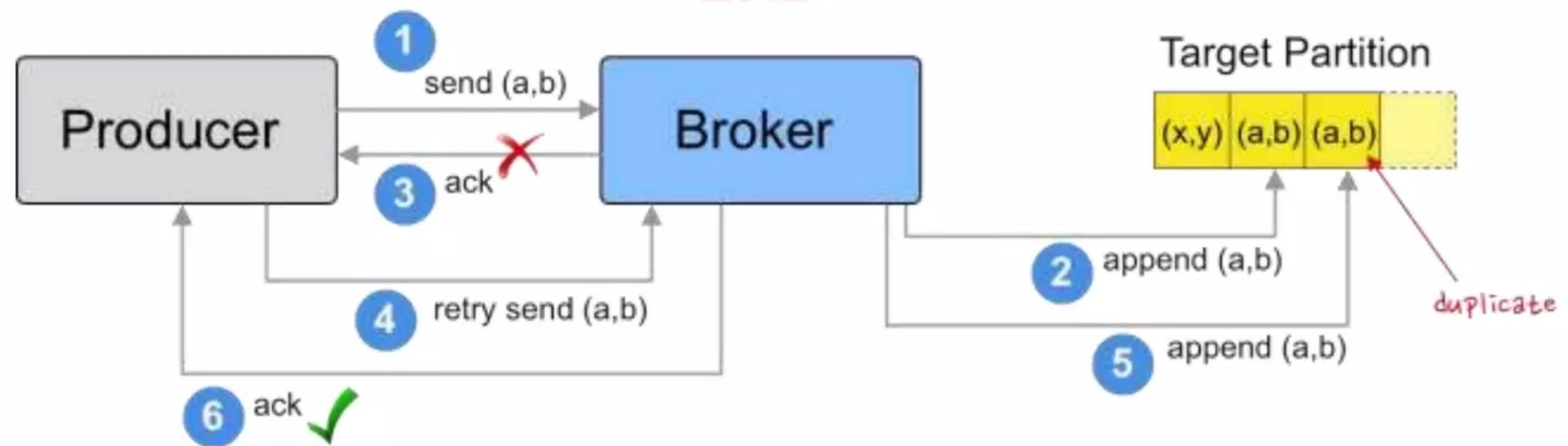


# Idempotent Producers

GOOD



BAD





# Exactly Once Semantics (EOS)

## What is it?

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

## Use Cases

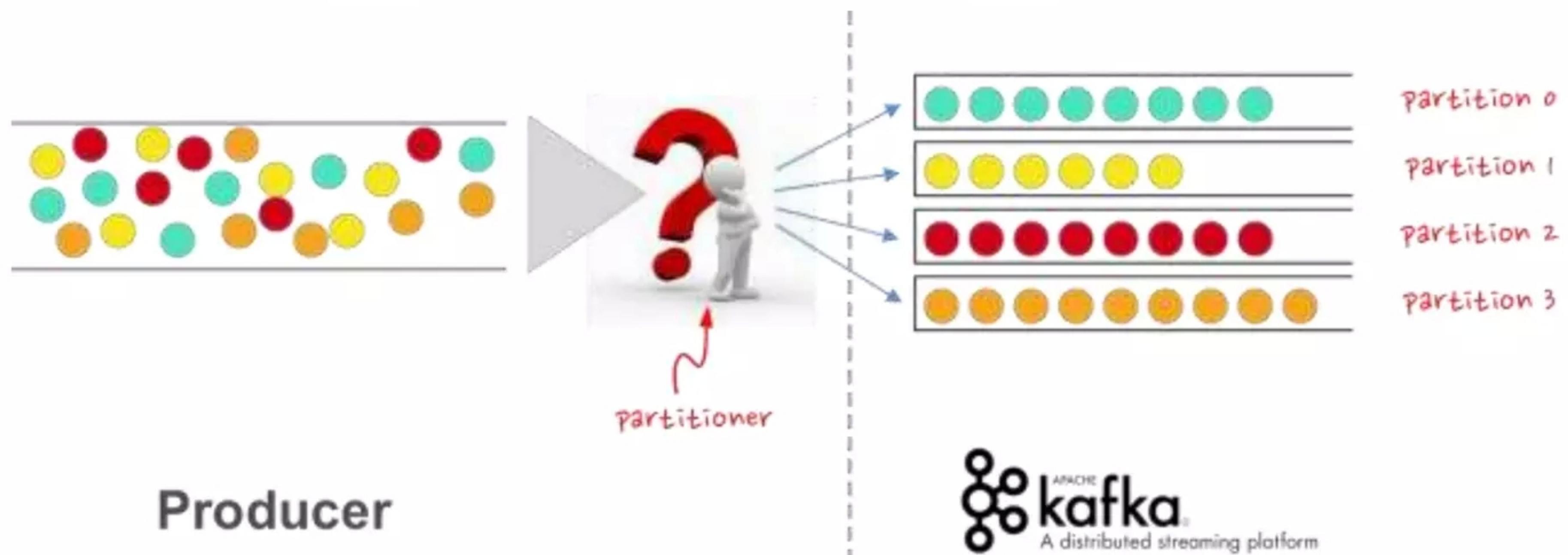
- Tracking ad views
- Processing financial transactions
- Stream processing



# Partitioning Strategies

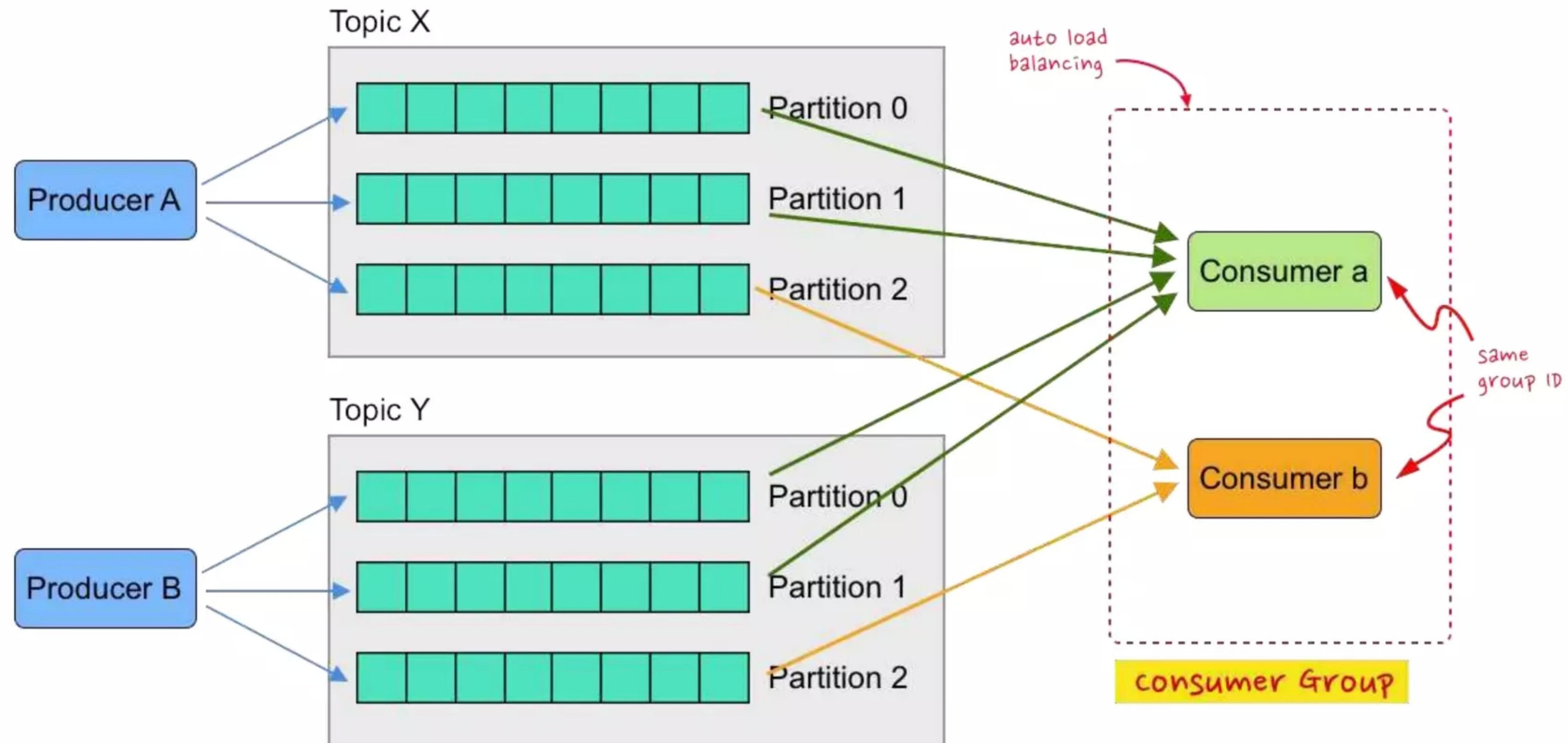
Why partitioning?

- Consumers need to **aggregate** or **join** by some key
- Consumers need **ordering guarantee**
- Concentrate data for **storage efficiency** and/or **indexing**



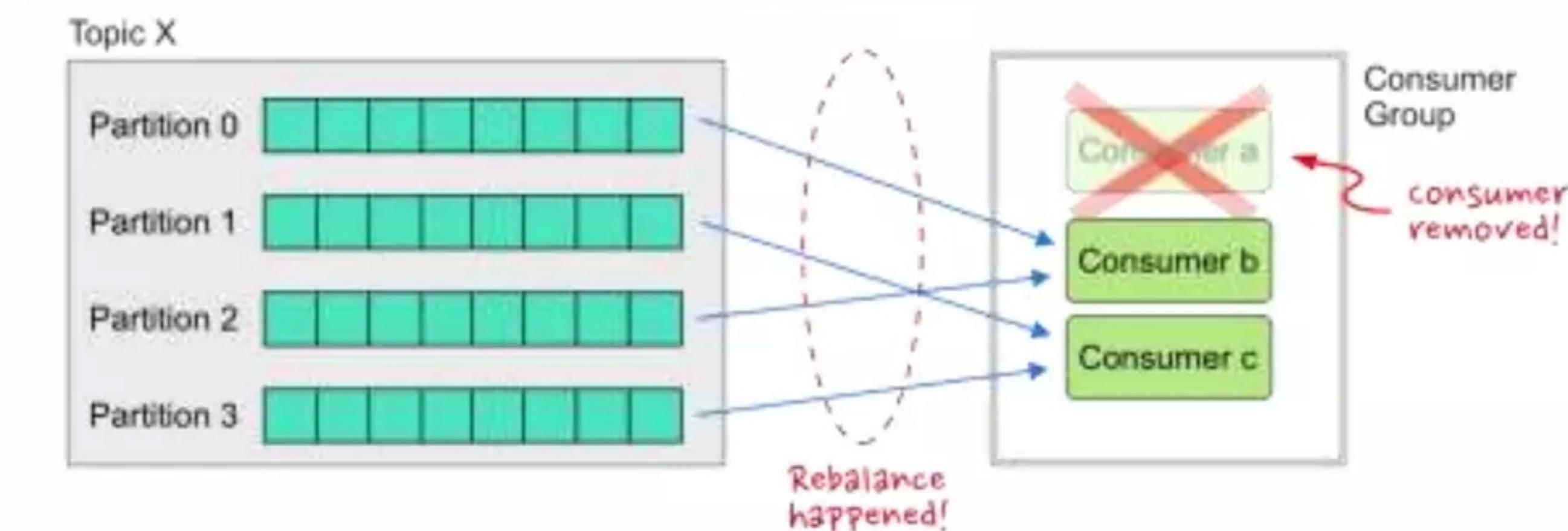
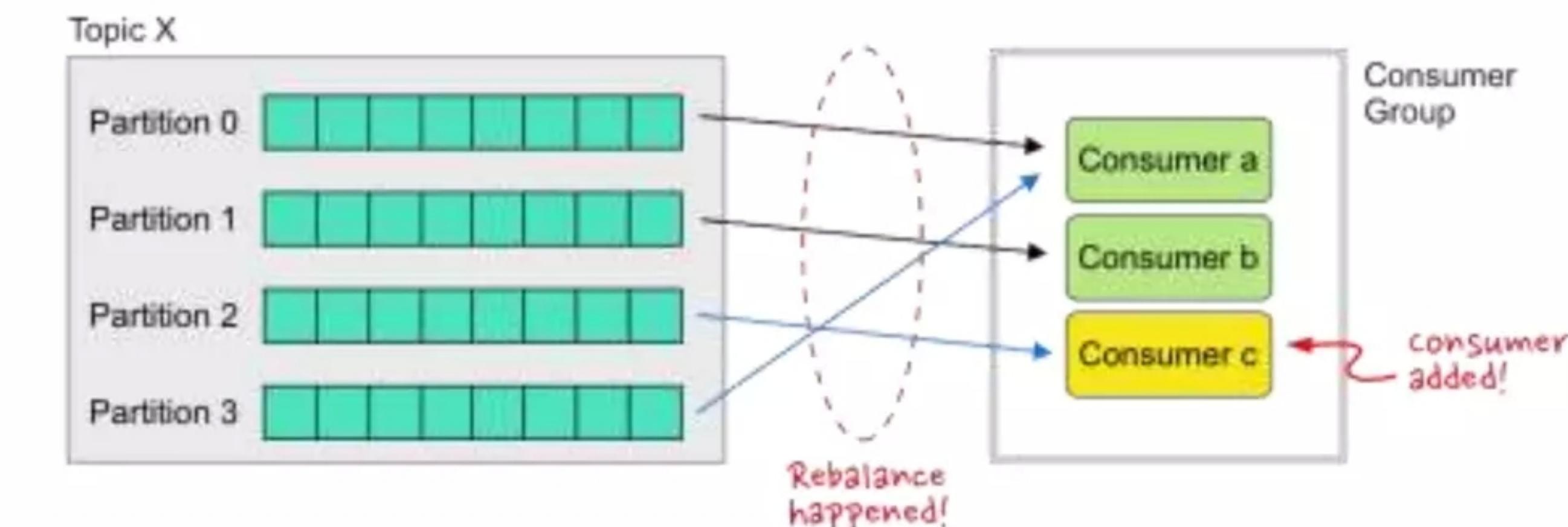
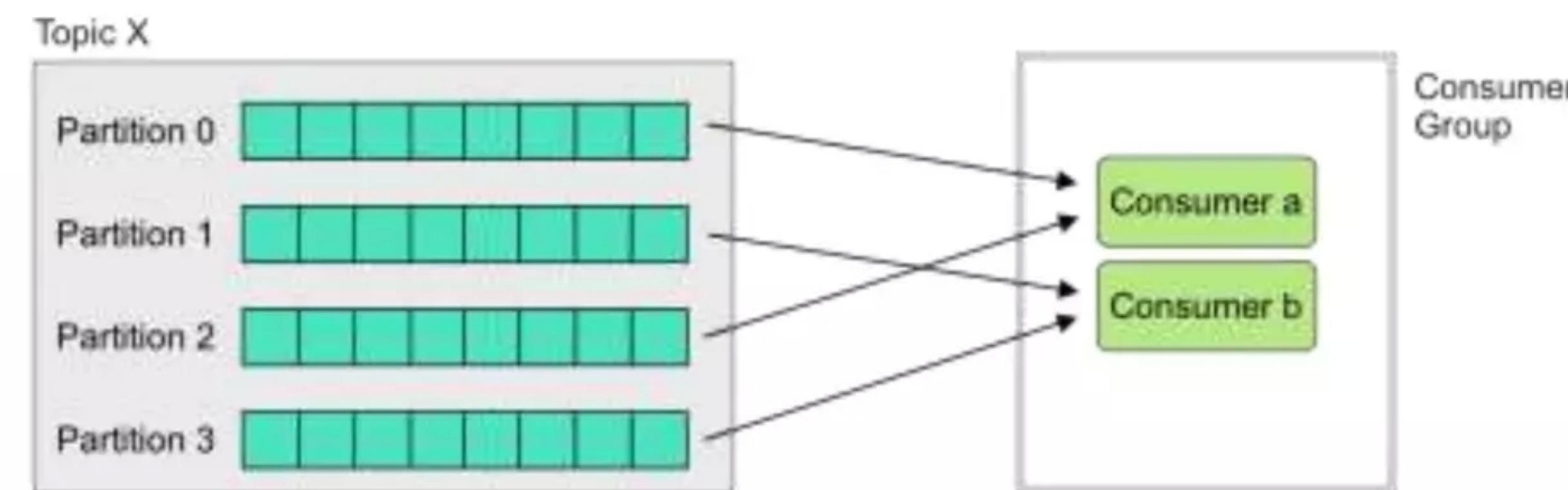


# Consumer Groups





# Consumer Rebalances





# Security Overview

- Kafka supports Encryption in Transit
- Kafka supports Authorization and Authentication
- No Encryption at Rest out of the box
- Clients can be mixed with & without Encryption & Authentication



# Client Side Security Features

- Encryption of Data in Transit
- Client Authentication
- Client Authorization



Encryption  
in transit  
SSL



authn & authz  
authn: SASL or SSL  
authz: ACLs



# Keen to learn more?

Register for one of the Confluent Streaming Events

07 October - Middle East

12 October – Nordics

15 October – Rest of Europe

Visit: <https://events.confluent.io/>





# Q&A



CONFLUENT