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

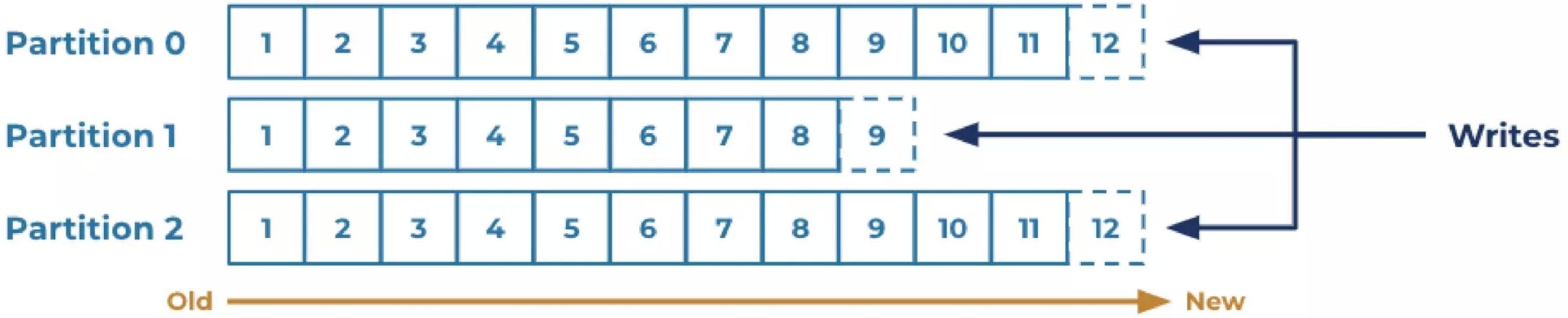
Store streams of events

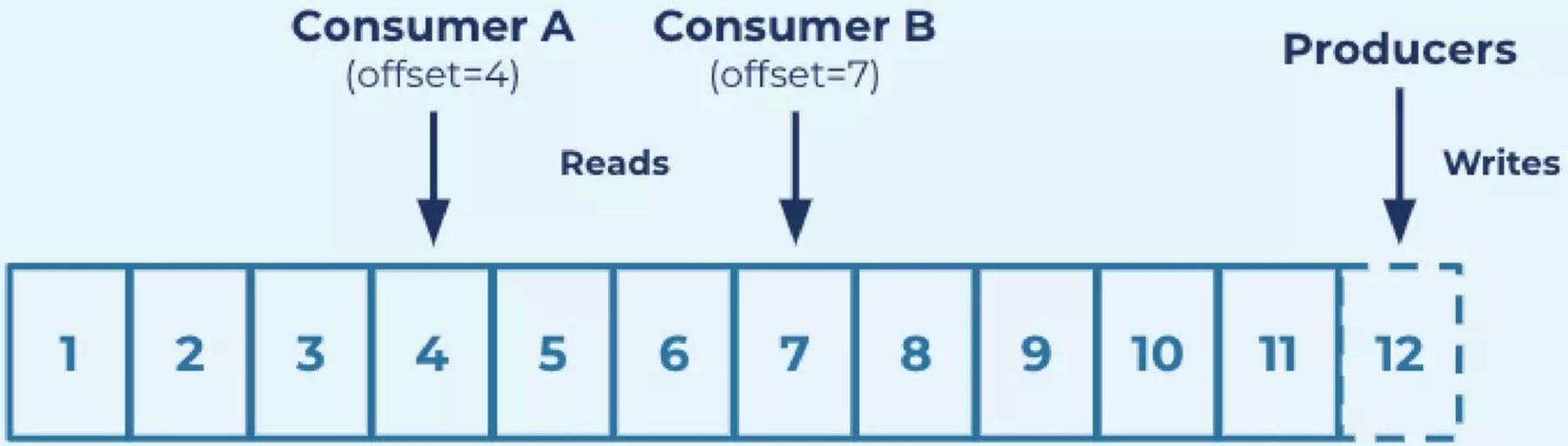
In a fault tolerant way

Process streams of events

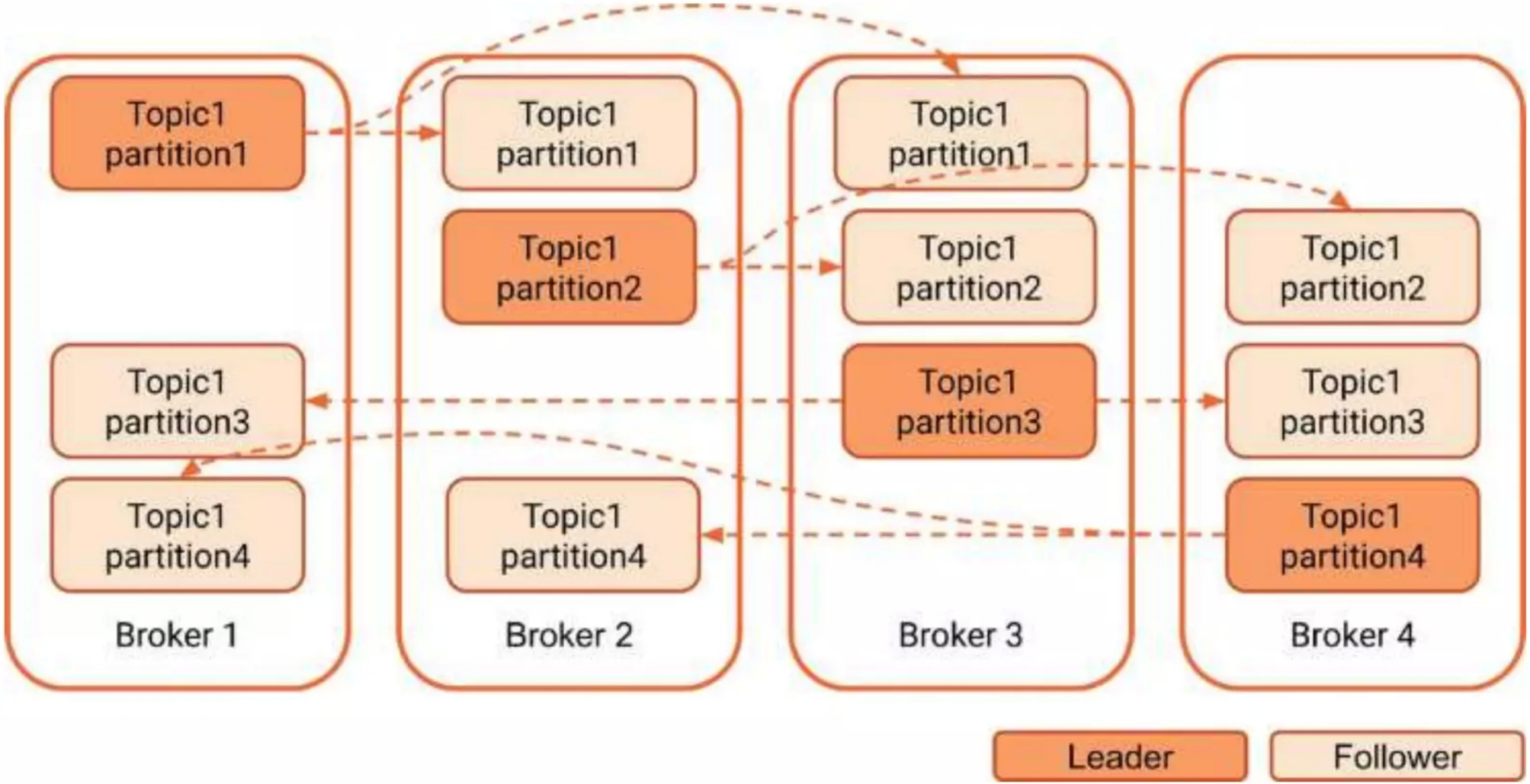
In real time, as they occur

Anatomy of a Kafka Topic

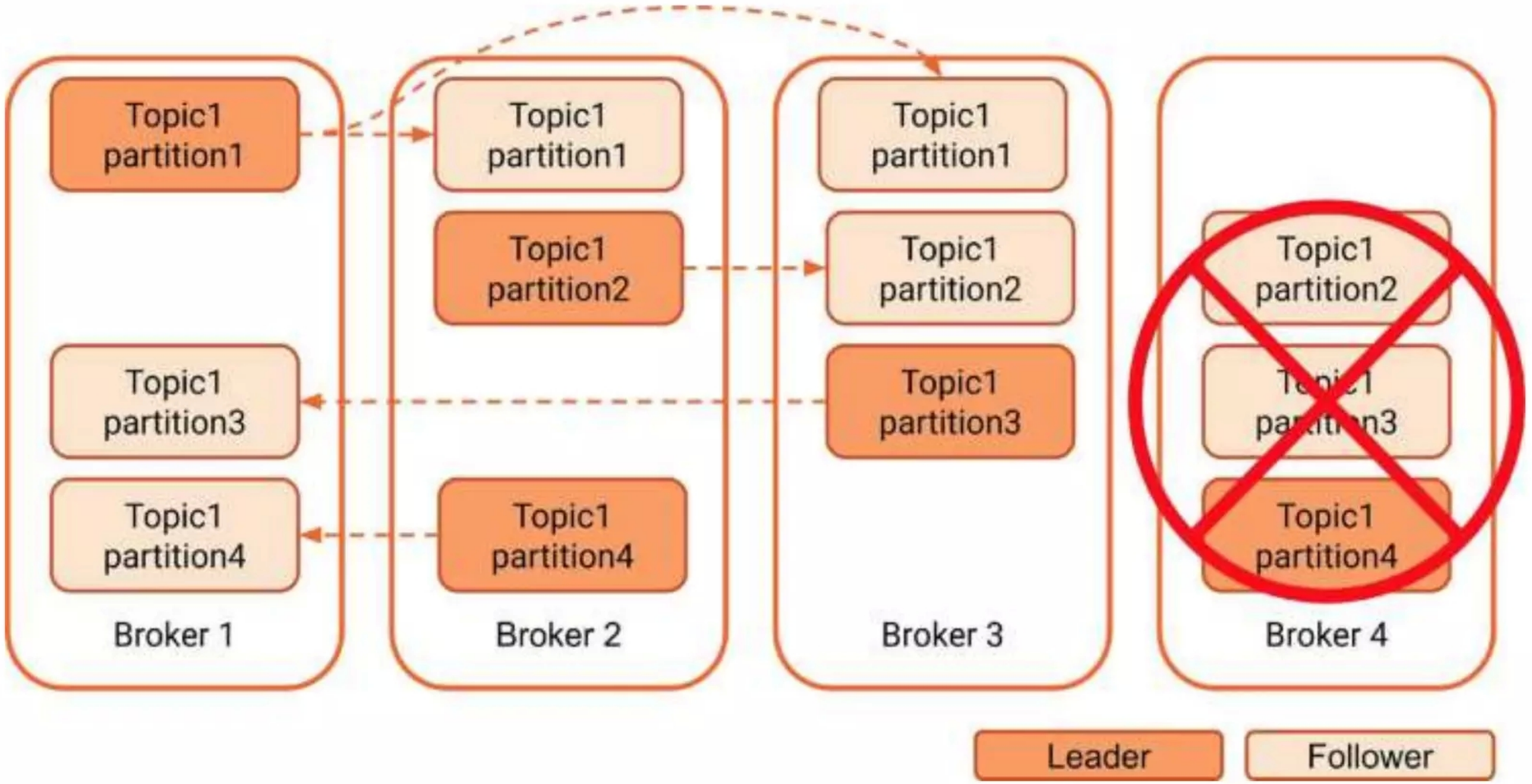




Partition Leadership & Replication



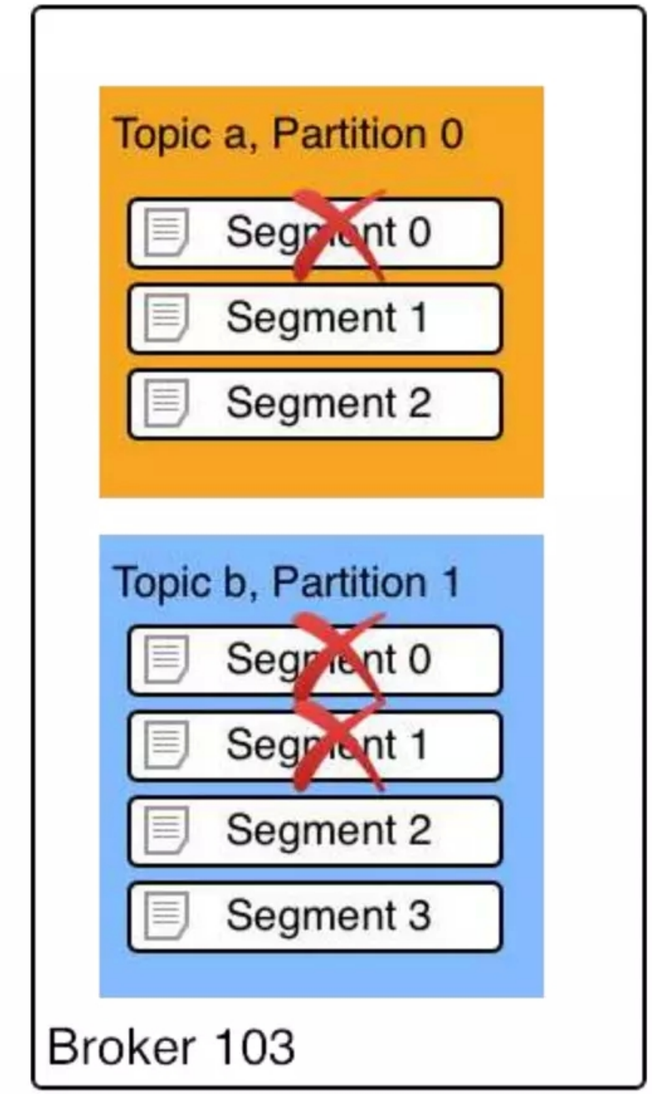
Partition Leadership & Replication



Data Retention Policy

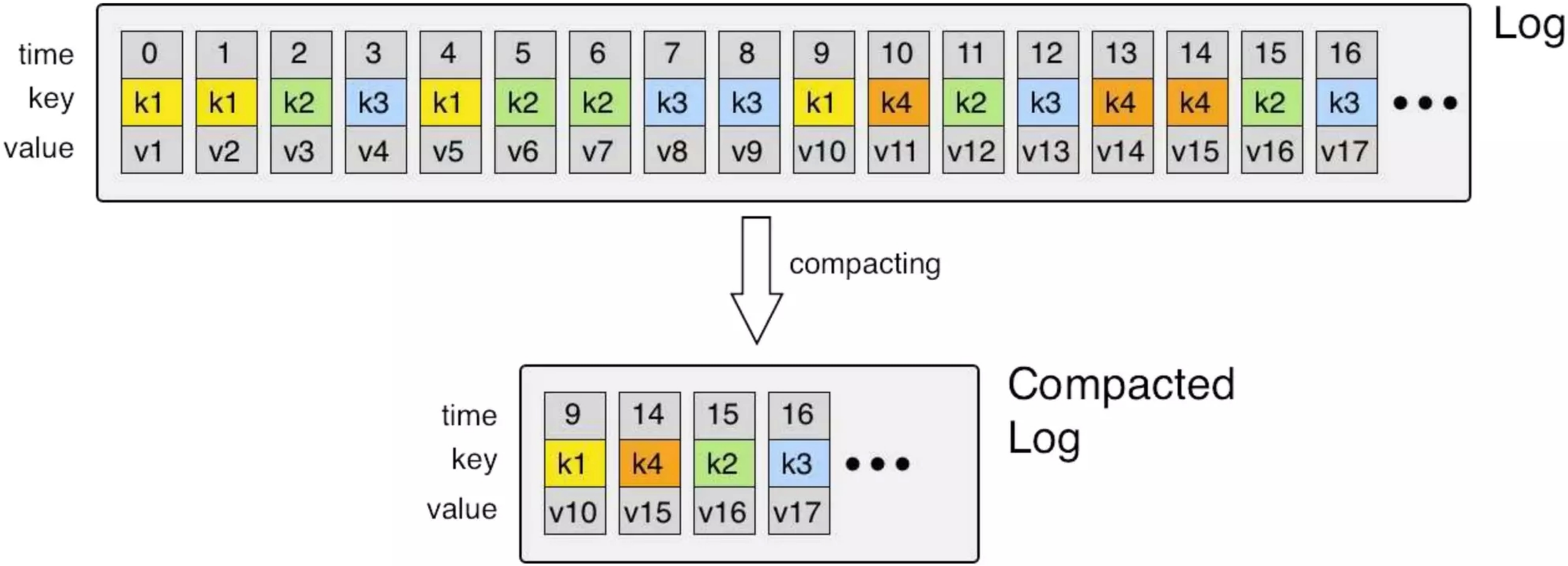
How long do I want or can I store my data?

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

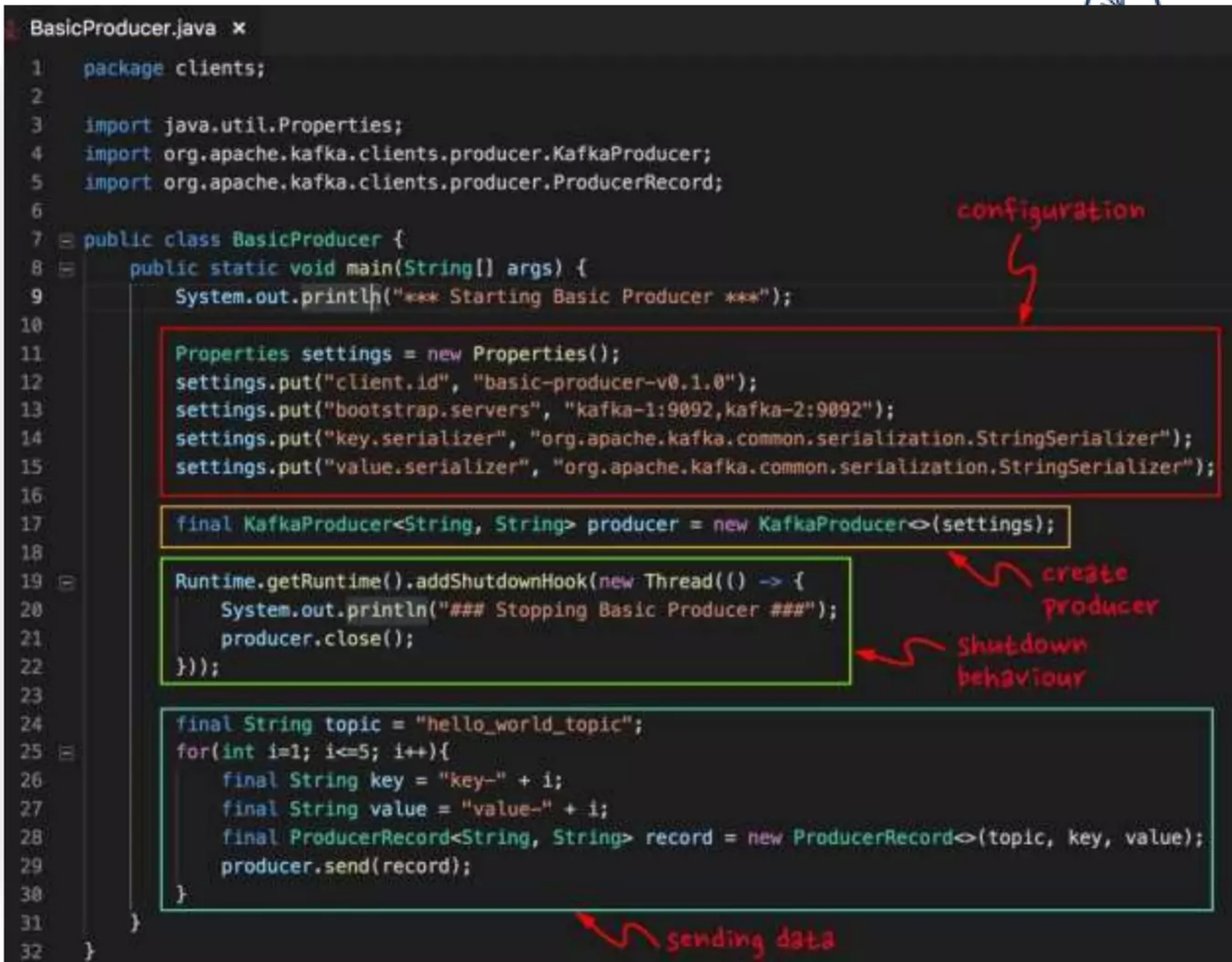


Retention Policy

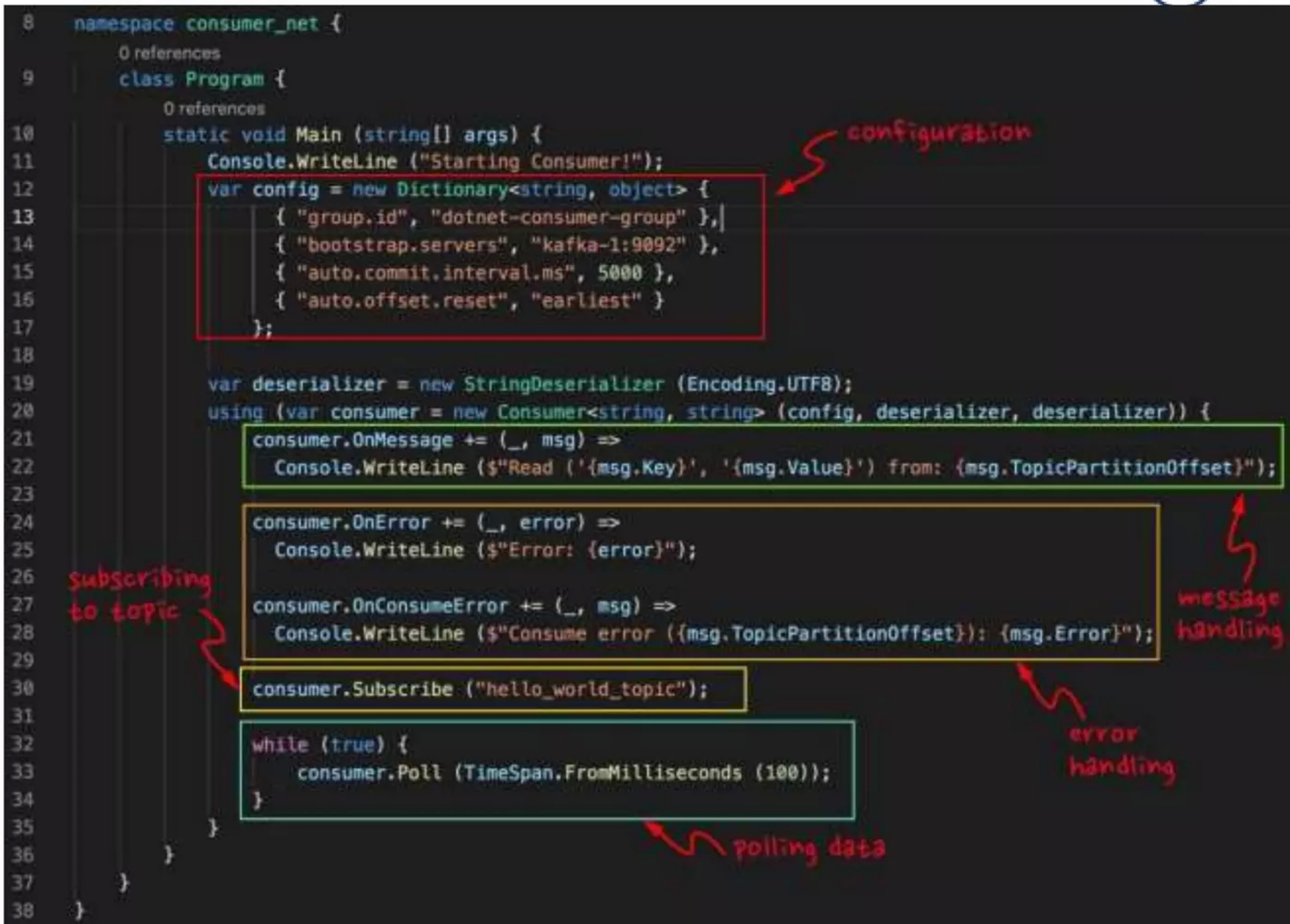
Compacted Topics



Development: A Basic Producer in Java

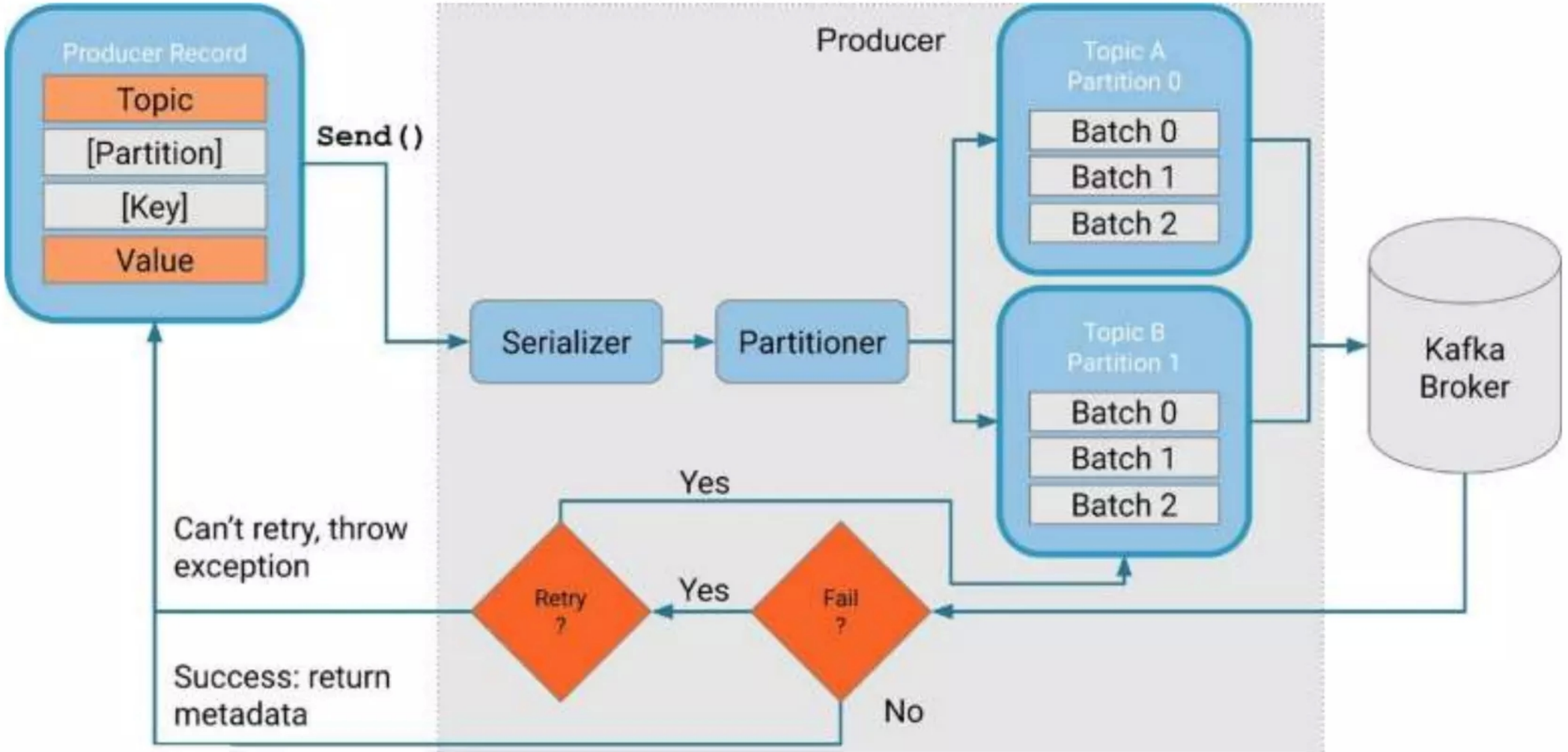


Development: A Basic Consumer in .NET/C#

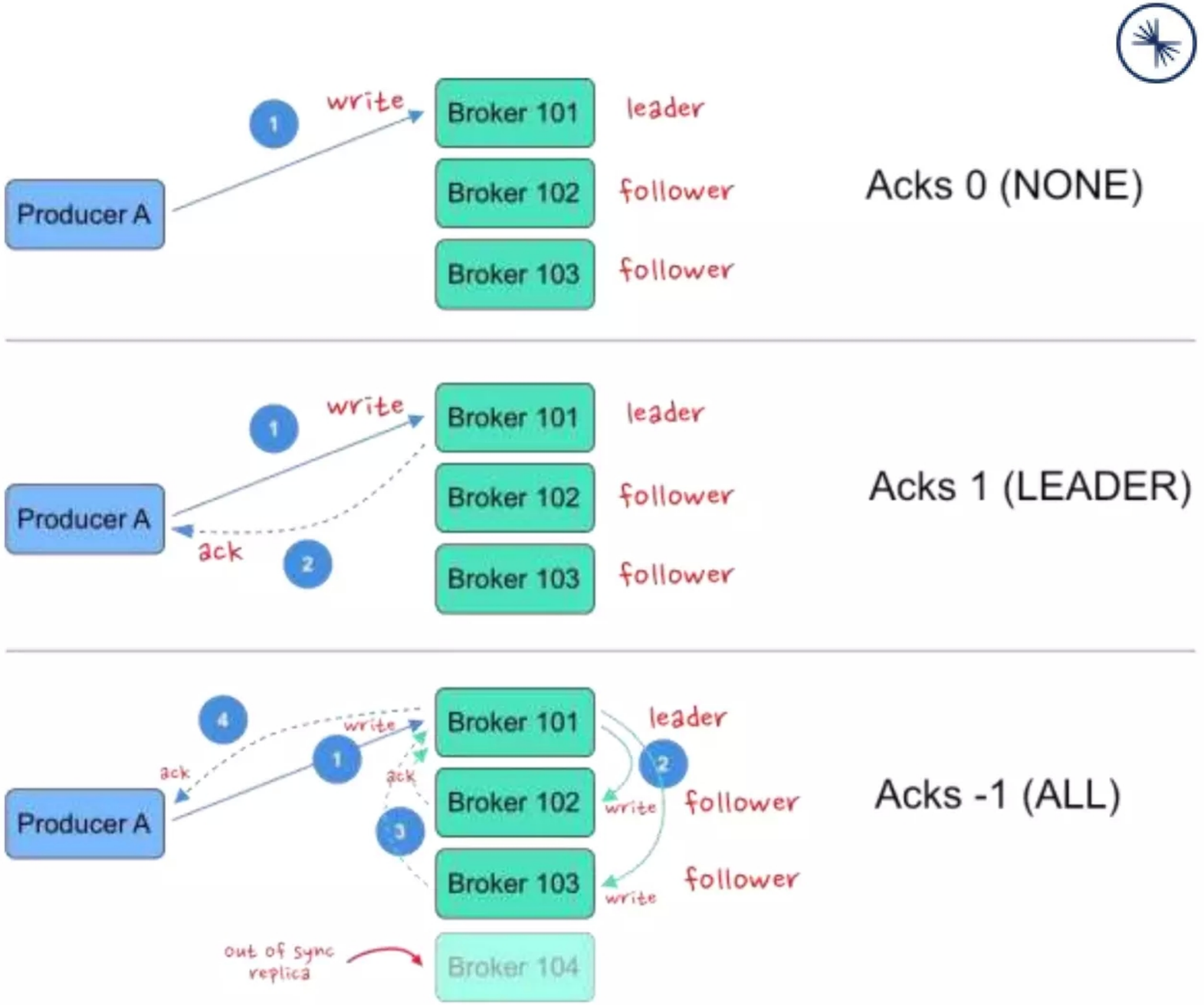



```
8   namespace consumer_net {
        0 references
9       class Program {
            0 references
10          static void Main (string[] args) {
11              Console.WriteLine ("Starting Consumer!");
12              var config = new Dictionary<string, object> {
13                  { "group.id",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

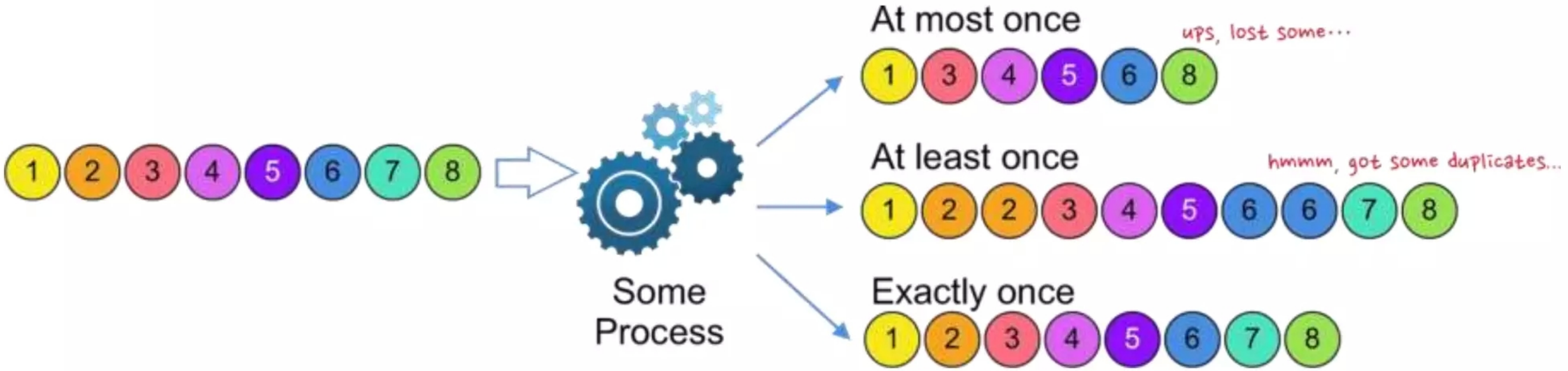
Producer Design



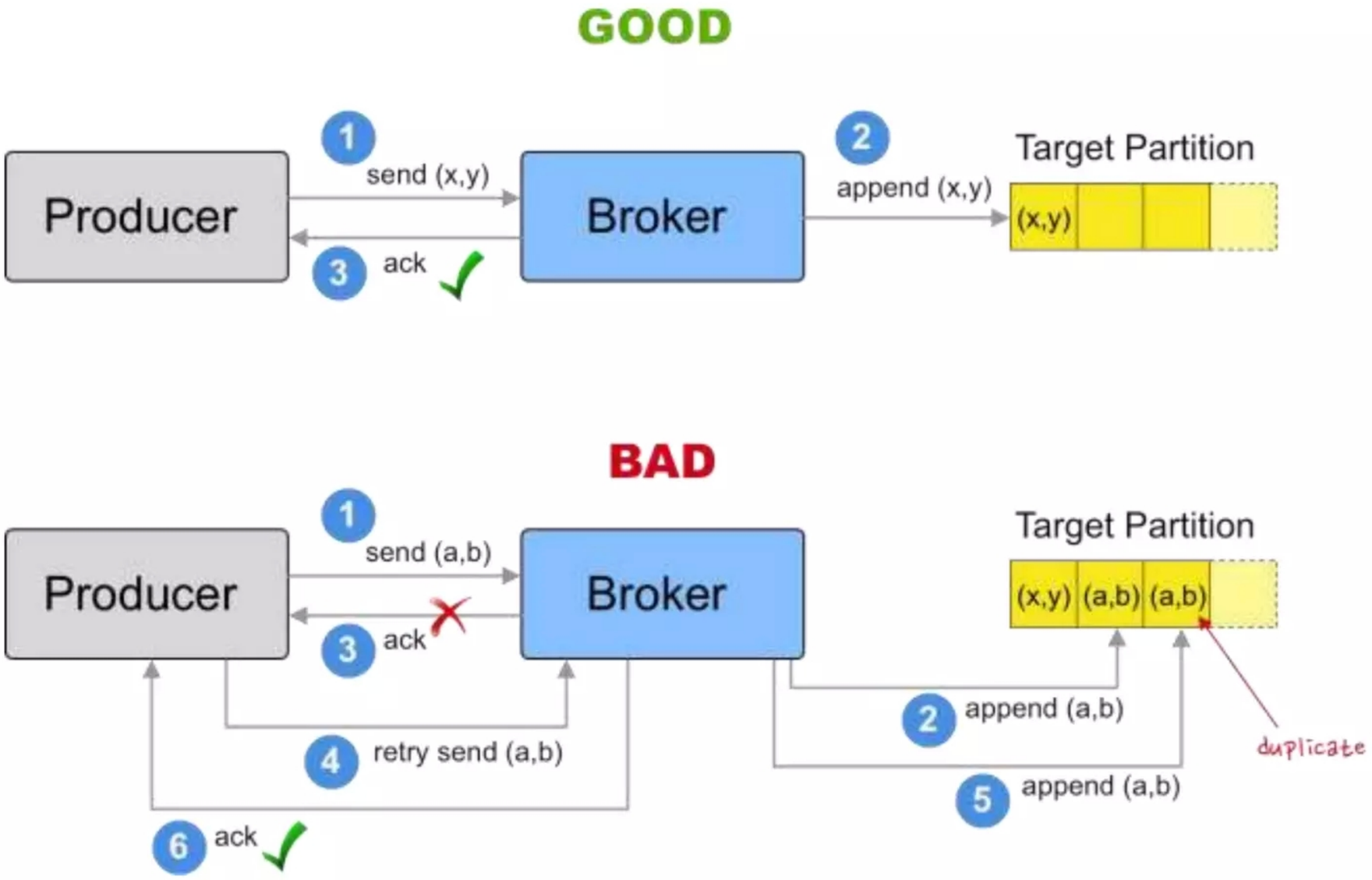
Producer Guarantees



Delivery Guarantees



Idempotent Producers



Exactly Once Semantics (EOS)

What is it?

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

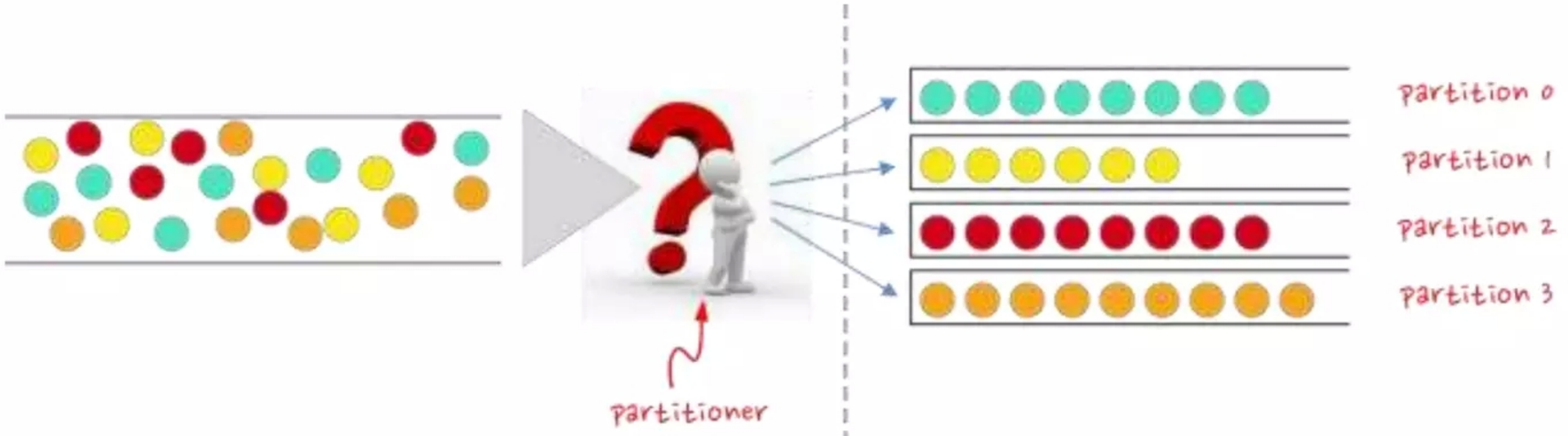
Use Cases

- Tracking ad views
- Processing financial transactions
- Stream processing

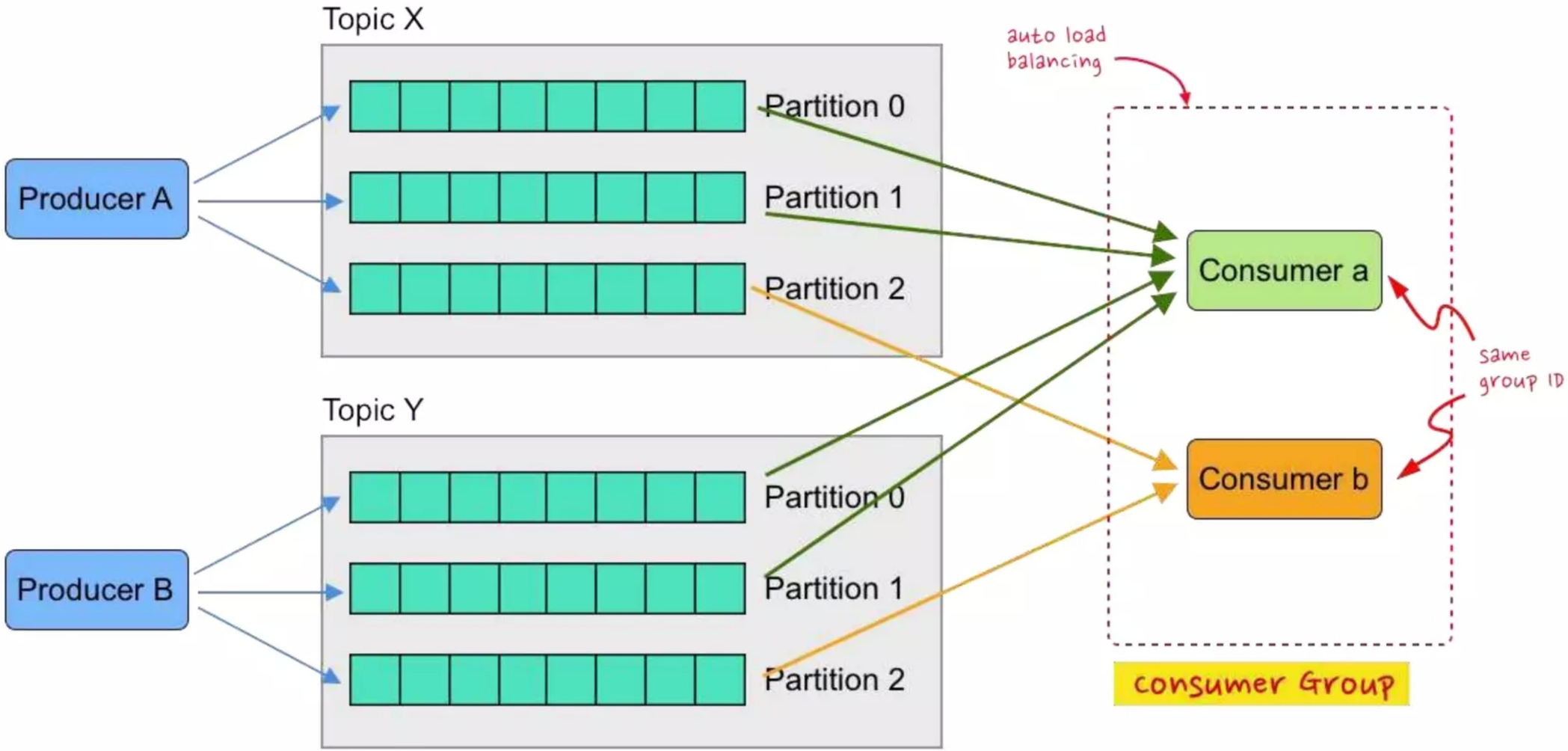
Partitioning Strategies

Why partitioning?

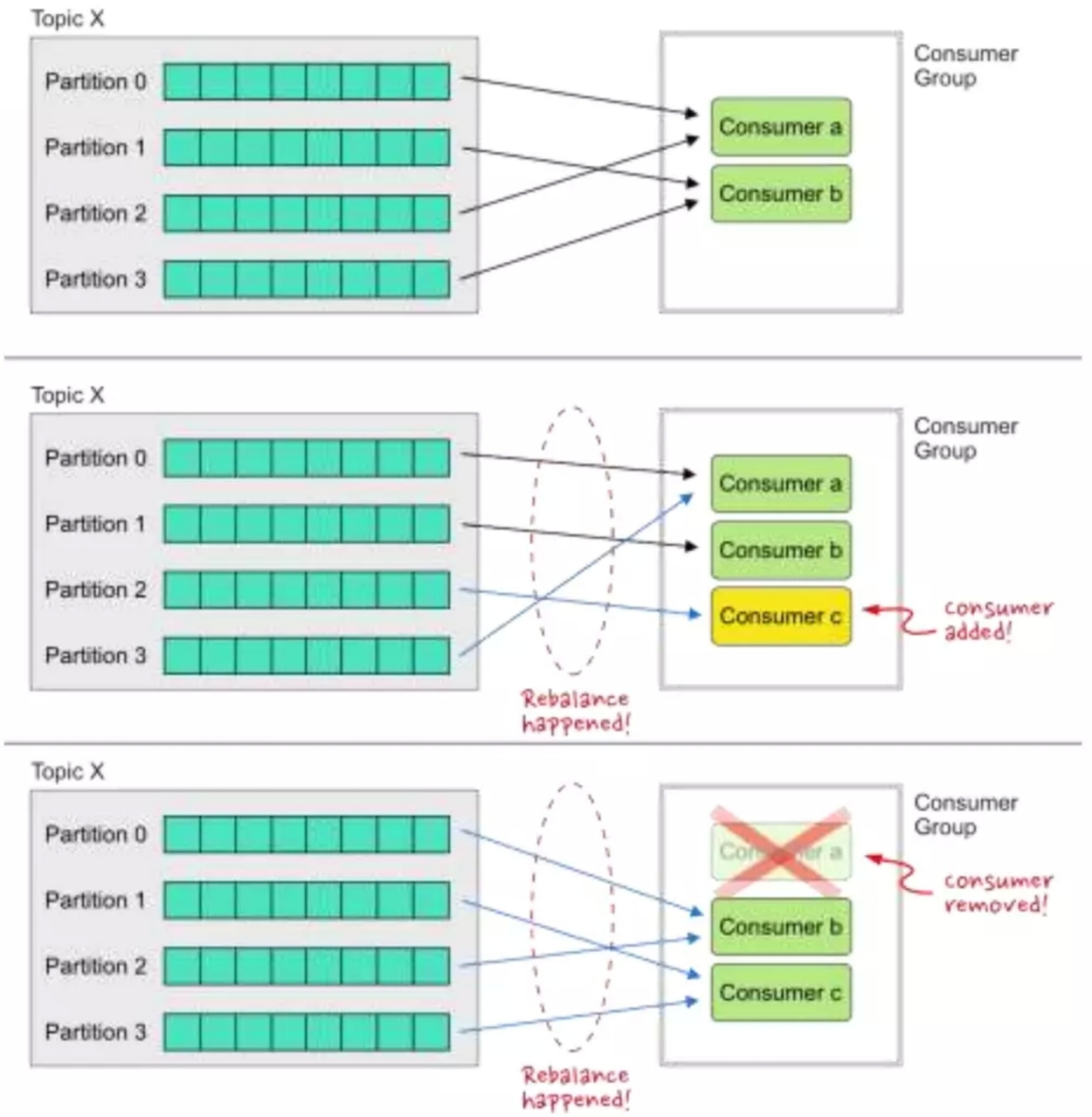
- Consumers need to **aggregate** or **join** by some key
- Consumers need **ordering guarantee**
- Concentrate data for **storage efficiency** and/or **indexing**



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