CONFLUENT

Apache Kafka Architectures and Fundamentals

Henrik Janzon, Solutions Engineer

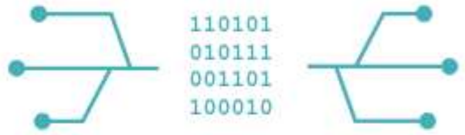
Learning Objectives

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

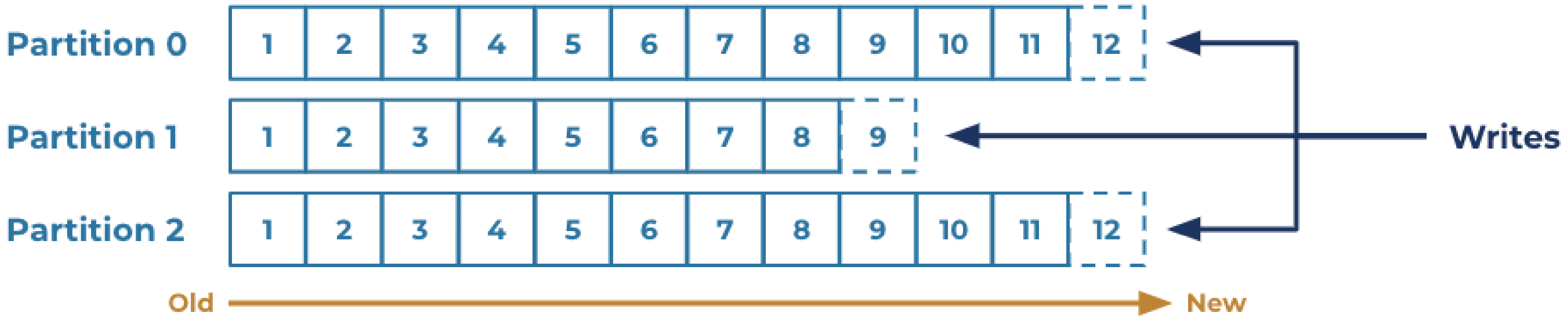
Store streams of events

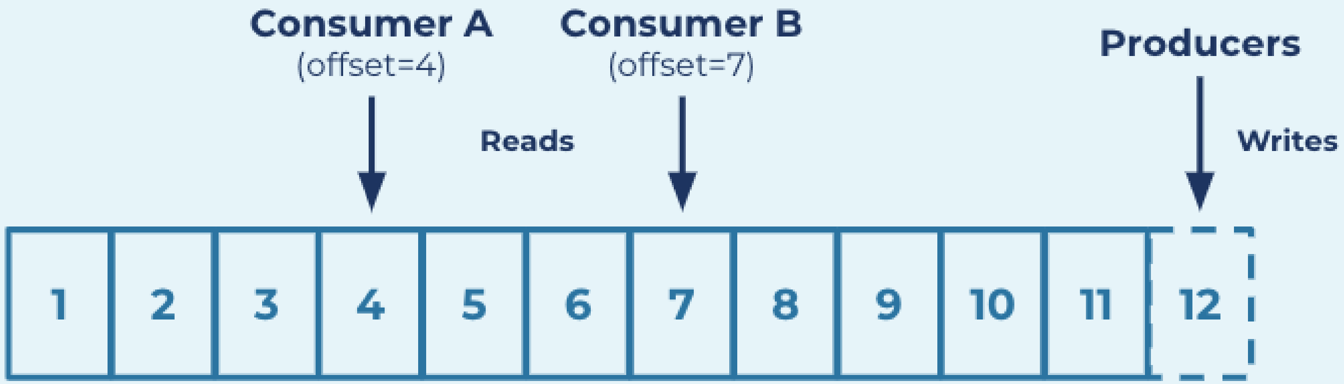
In a fault tolerant way

Process streams of events

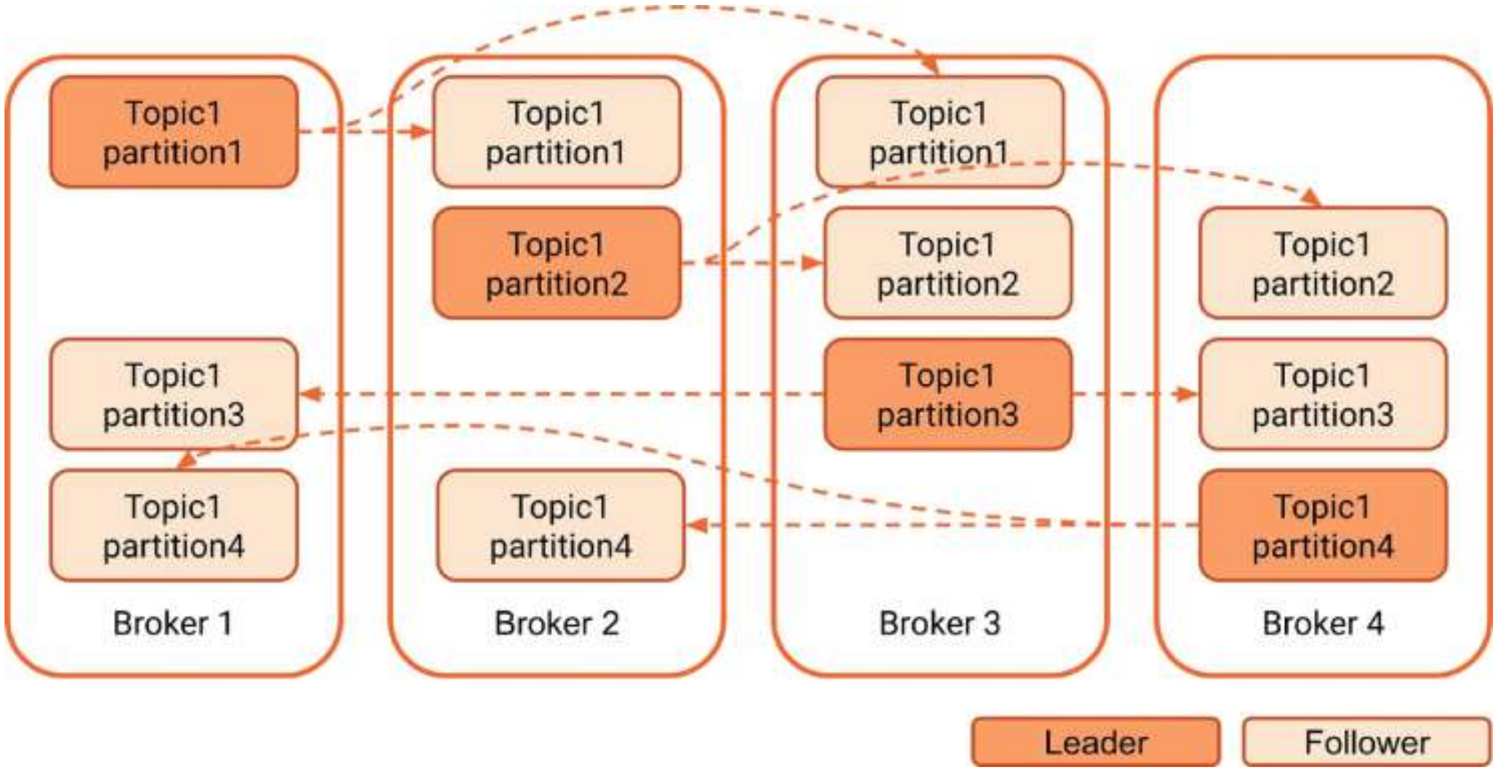
In real time, as they occur

Anatomy of a Kafka Topic

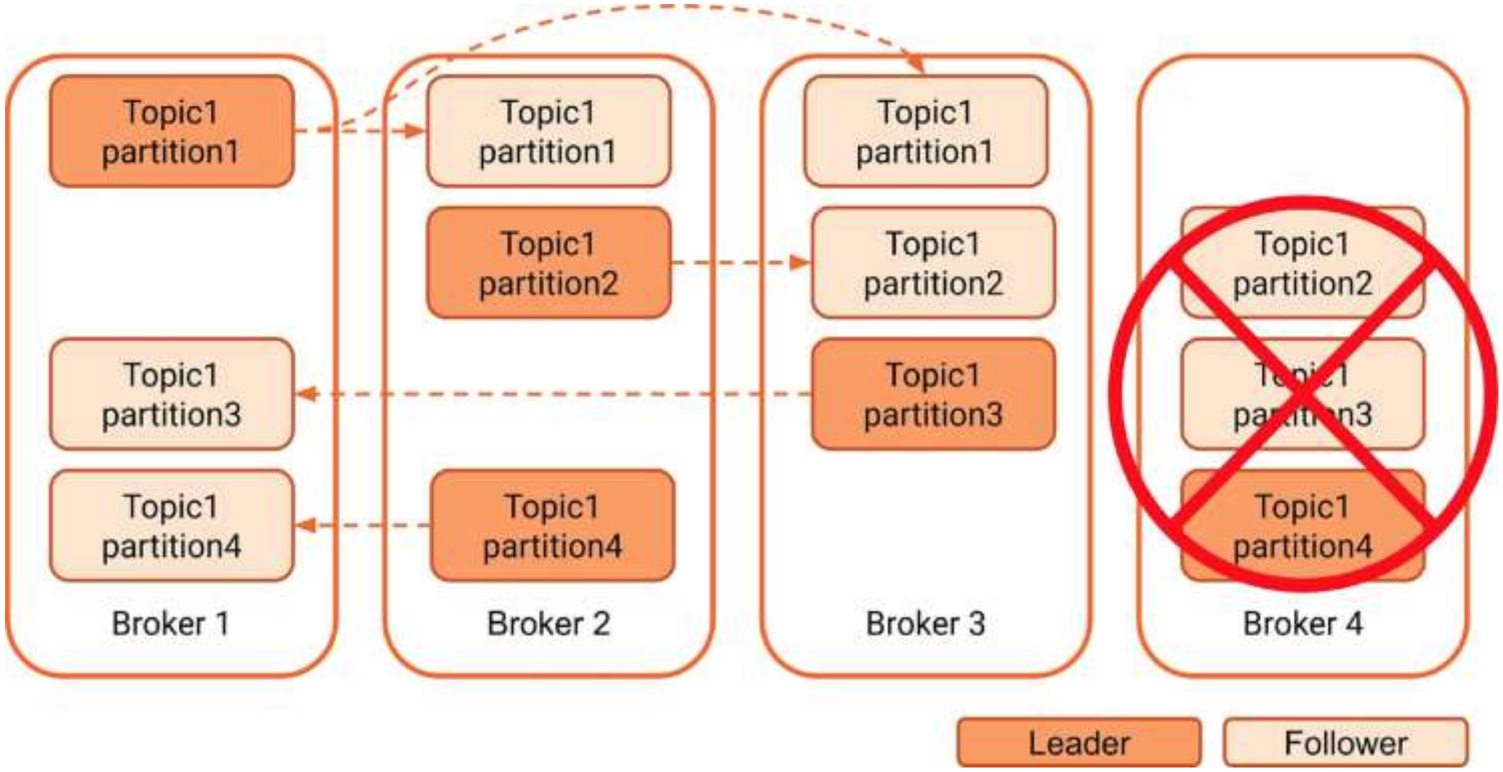




Partition Leadership & Replication



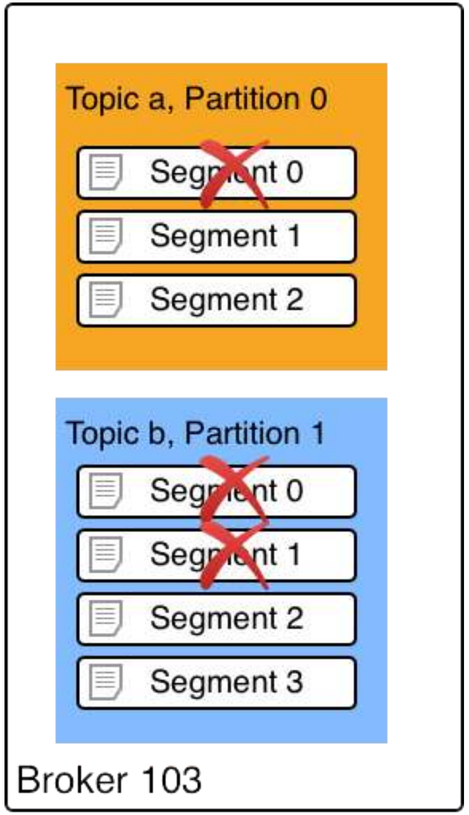
Partition Leadership & Replication



Data Retention Policy

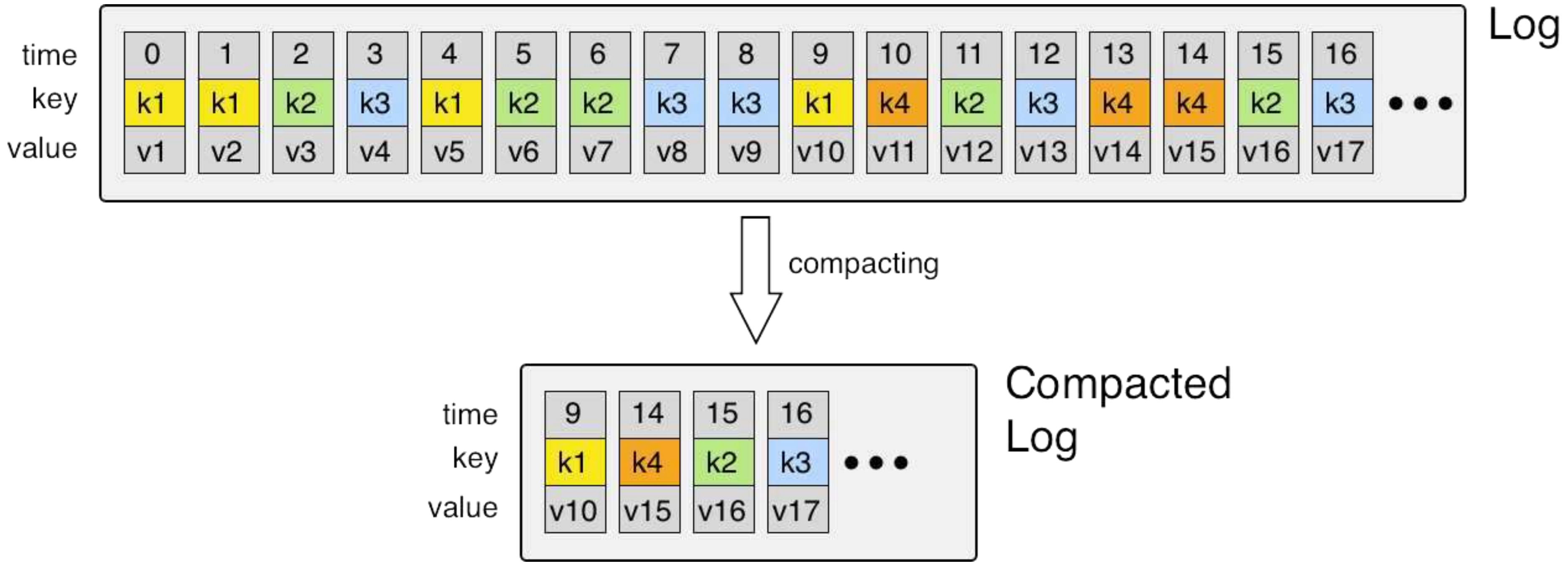
How long do I want or can I store my data?

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

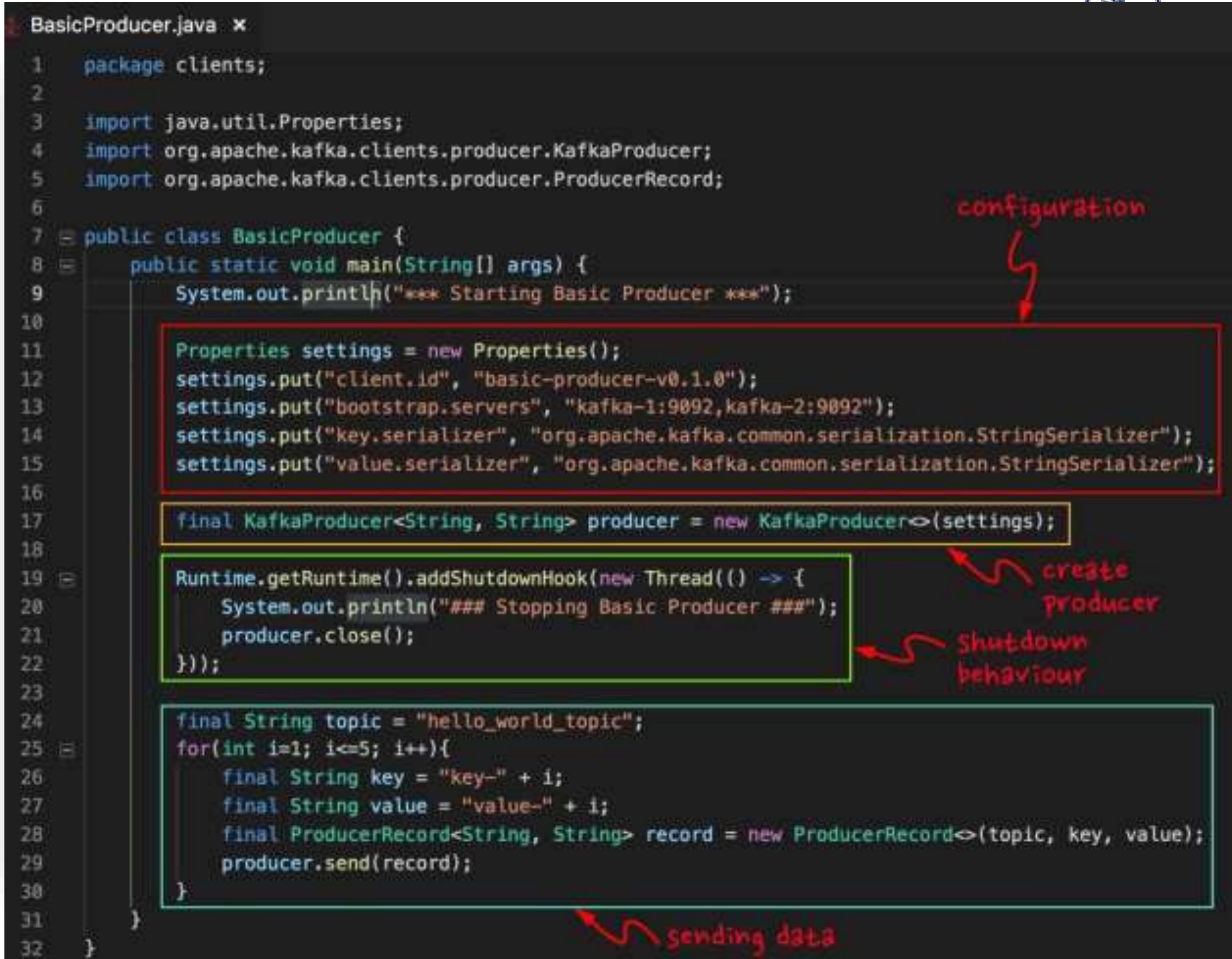


Retention Policy

Compacted Topics

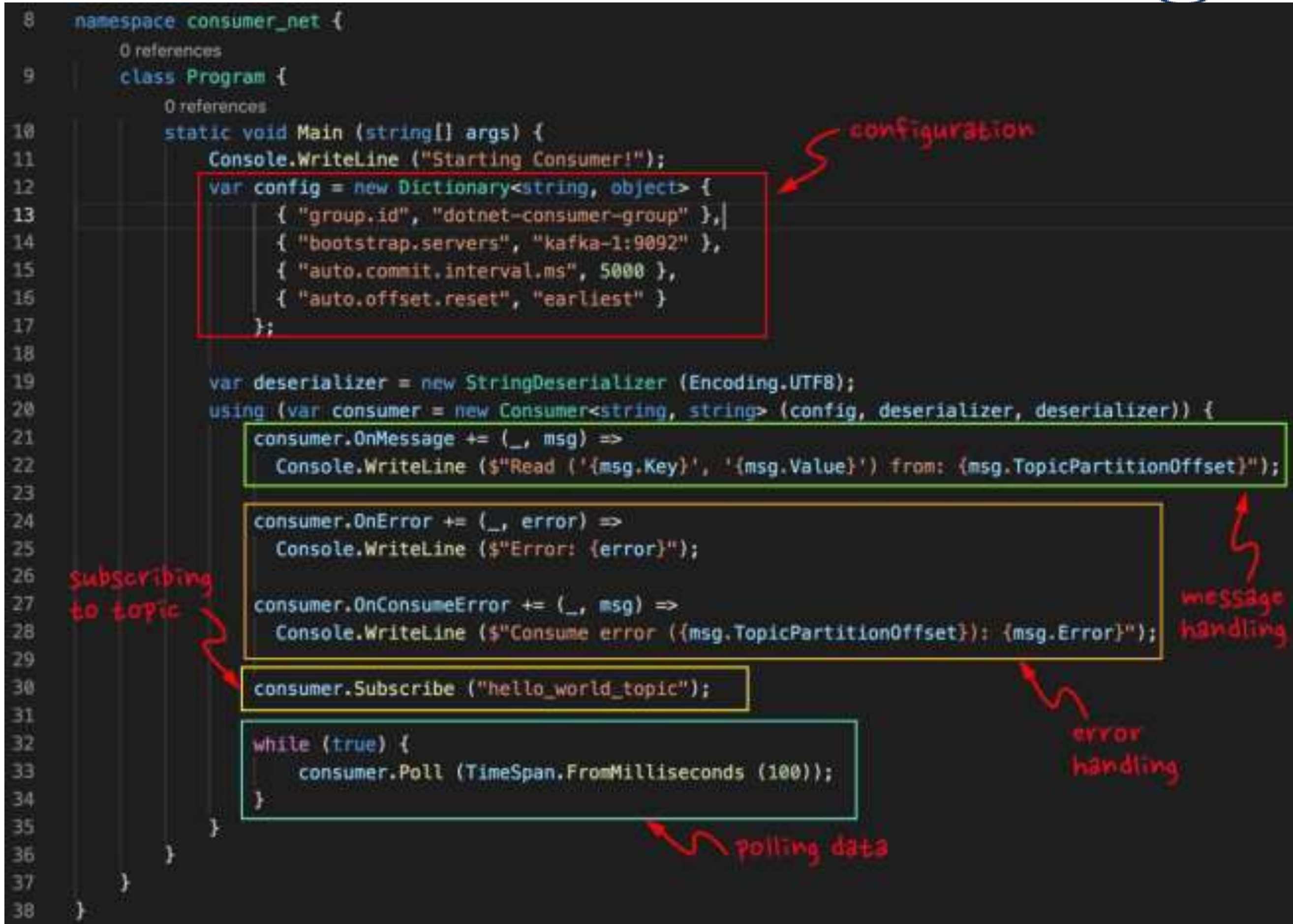


Development: A Basic Producer in Java



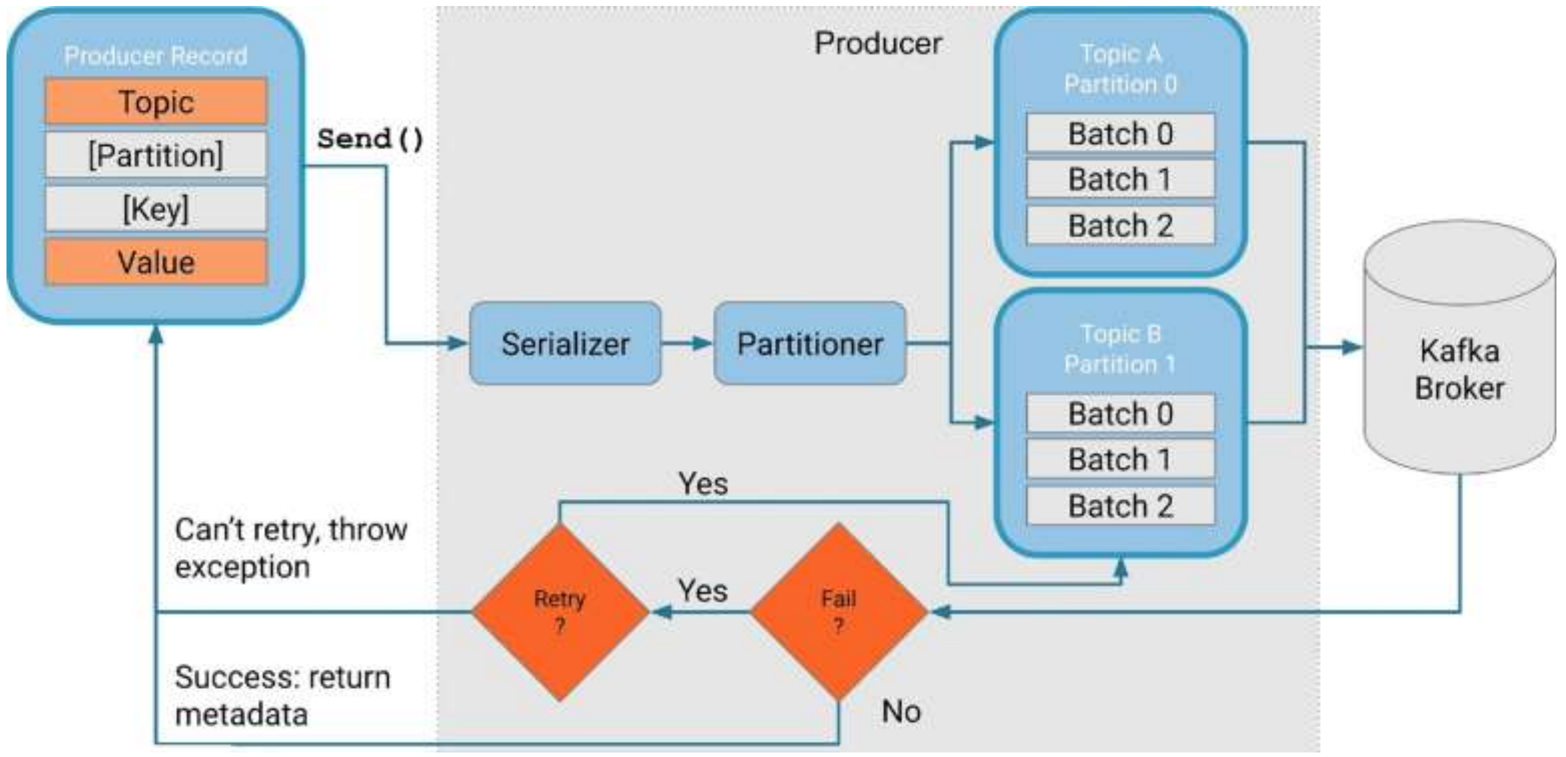
```java
package clients;

import java.util.Properties;
import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.ProducerRecord;

public class BasicProducer {
    public static void main(String[] args) {
        System.out.println("*** Starting Basic Producer ***");

        Properties settings = new Properties();
        settings.put("client.id", "basic-producer-v0.1.0");
        settings.put("bootstrap.servers", "kafka-1:9092,kafka-2:9092");
        settings.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");
        settings.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");

        final KafkaProducer<String, String> producer = new KafkaProducer<>(settings);

        Runtime.getRuntime().addShutdownHook(new Thread(() -> {
            System.out.println("### Stopping Basic Producer ###");
            producer.close();
        }));

        final String topic = "hello_world_topic";
        for(int i=1; i<=5; i++){
            final String key = "key-" + i;
            final String value = "value-" + i;
            final ProducerRecord<String, String> record = new ProducerRecord<>(topic, key, value);
            producer.send(record);
        }
    }
}
```

Development: A Basic Consumer in .NET/C#

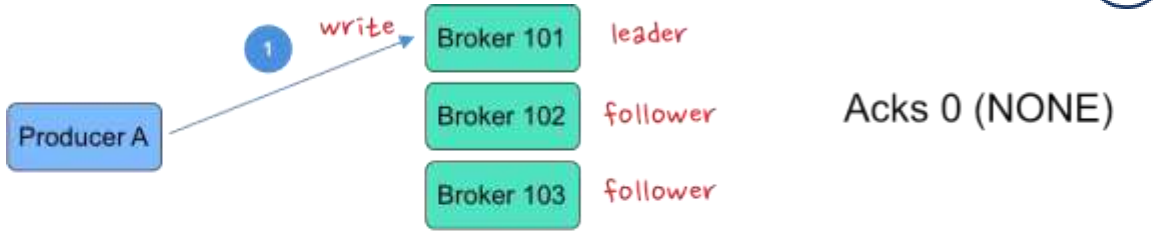


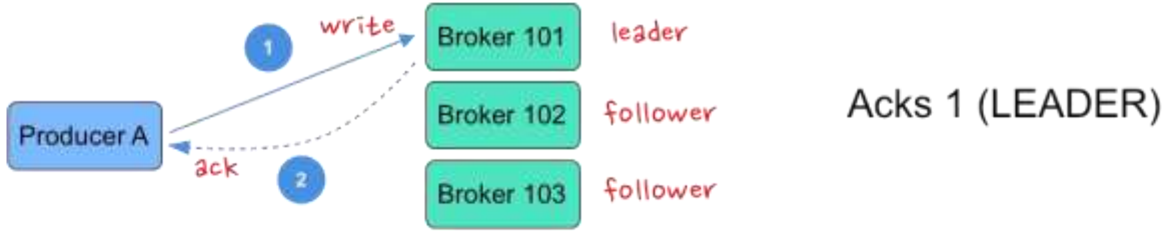
```
namespace consumer_net {
    0 references
    class Program {
        0 references
        static void Main (string[] args) {
            Console.WriteLine ("Starting Consumer!");
            var config = new Dictionary<string, object> {
                { "group.id", "dotnet-consumer-group" },
                { "bootstrap.servers", "kafka-1:9092" },
                { "auto.commit.interval.ms", 5000 },
                { "auto.offset.reset", "earliest" }
            };

            var deserializer = new StringDeserializer (Encoding.UTF8);
            using (var consumer = new Consumer<string, string> (config, deserializer, deserializer)) {
                consumer.OnMessage += (_, msg) =>
                  Console.WriteLine ($"Read ('{msg.Key}', '{msg.Value}') from: {msg.TopicPartitionOffset}");

                consumer.OnError += (_, error) =>
                  Console.WriteLine ($"Error: {error}");

                consumer.OnConsumeError += (_, msg) =>
                  Console.WriteLine ($"Consume error ({msg.TopicPartitionOffset}): {msg.Error}");

                consumer.Subscribe ("hello_world_topic");

                while (true) {
                    consumer.Poll (TimeSpan.FromMilliseconds (100));
                }
            }
        }
    }
}
```

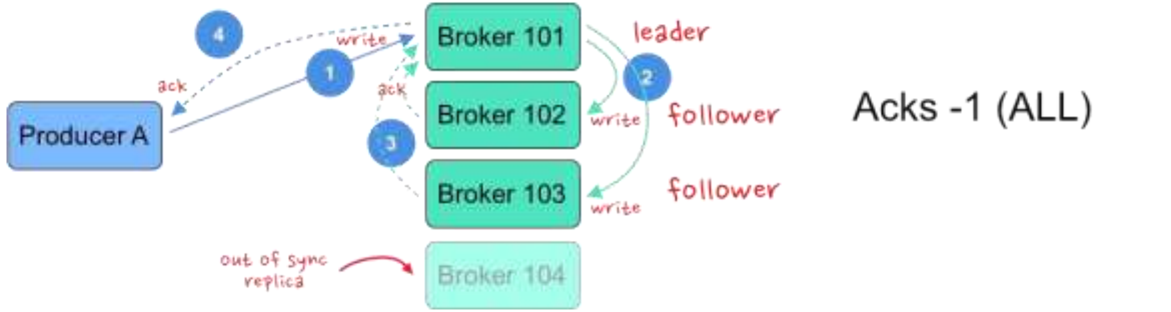
Producer Design



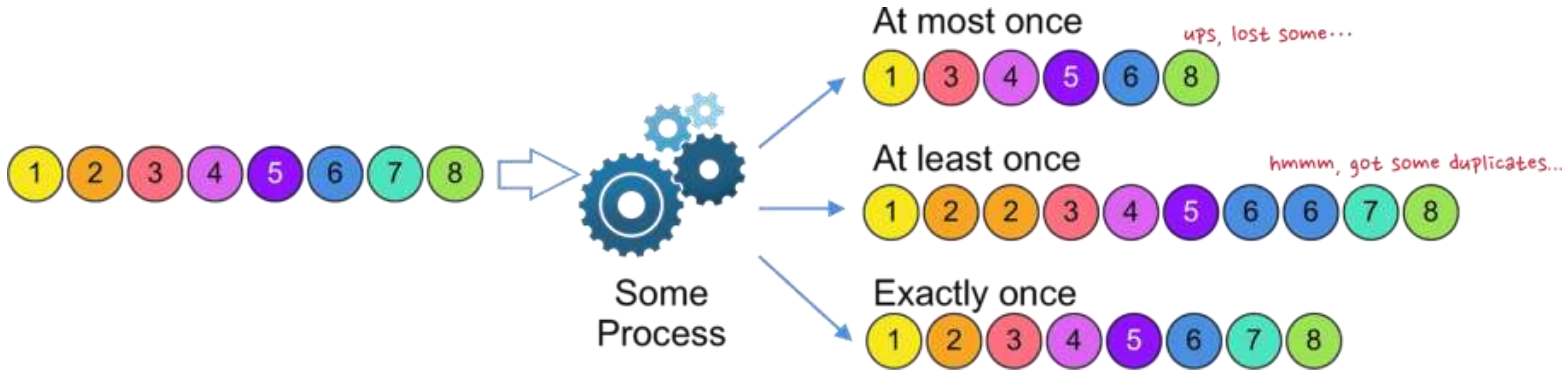
Producer Guarantees



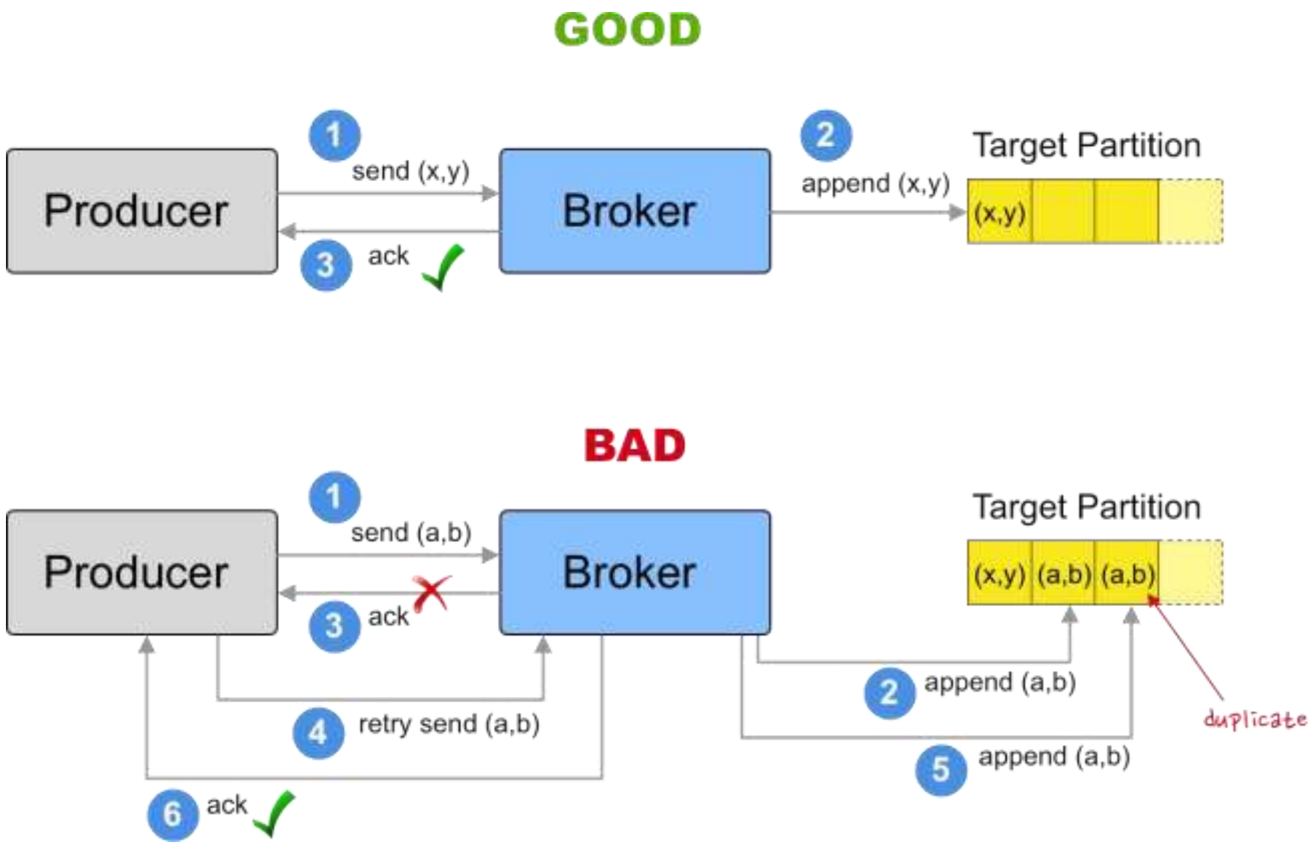




Delivery Guarantees



Idempotent Producers



Exactly Once Semantics (EOS)

What is it?

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

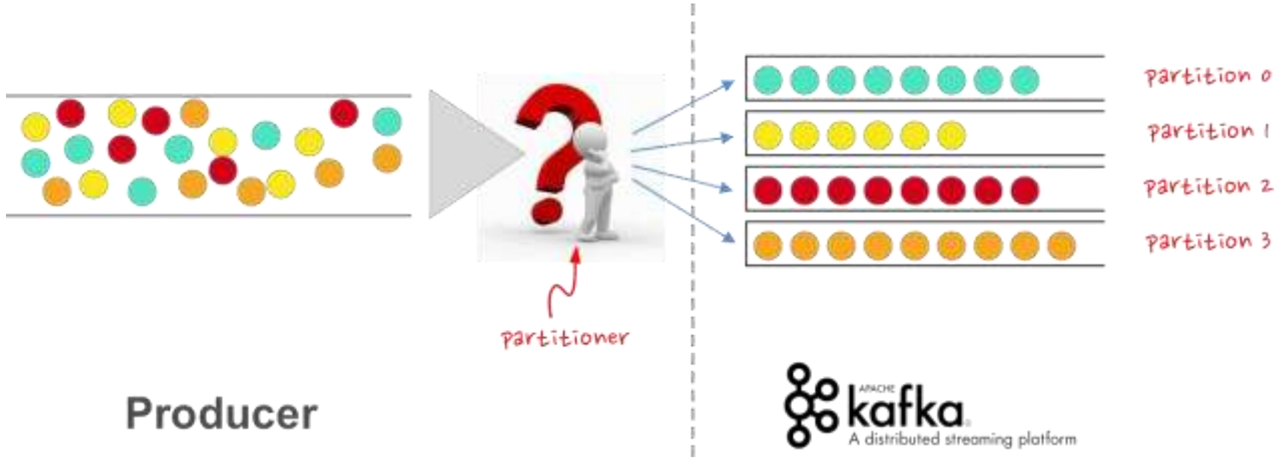
Use Cases

- Tracking ad views
- Processing financial transactions
- Stream processing

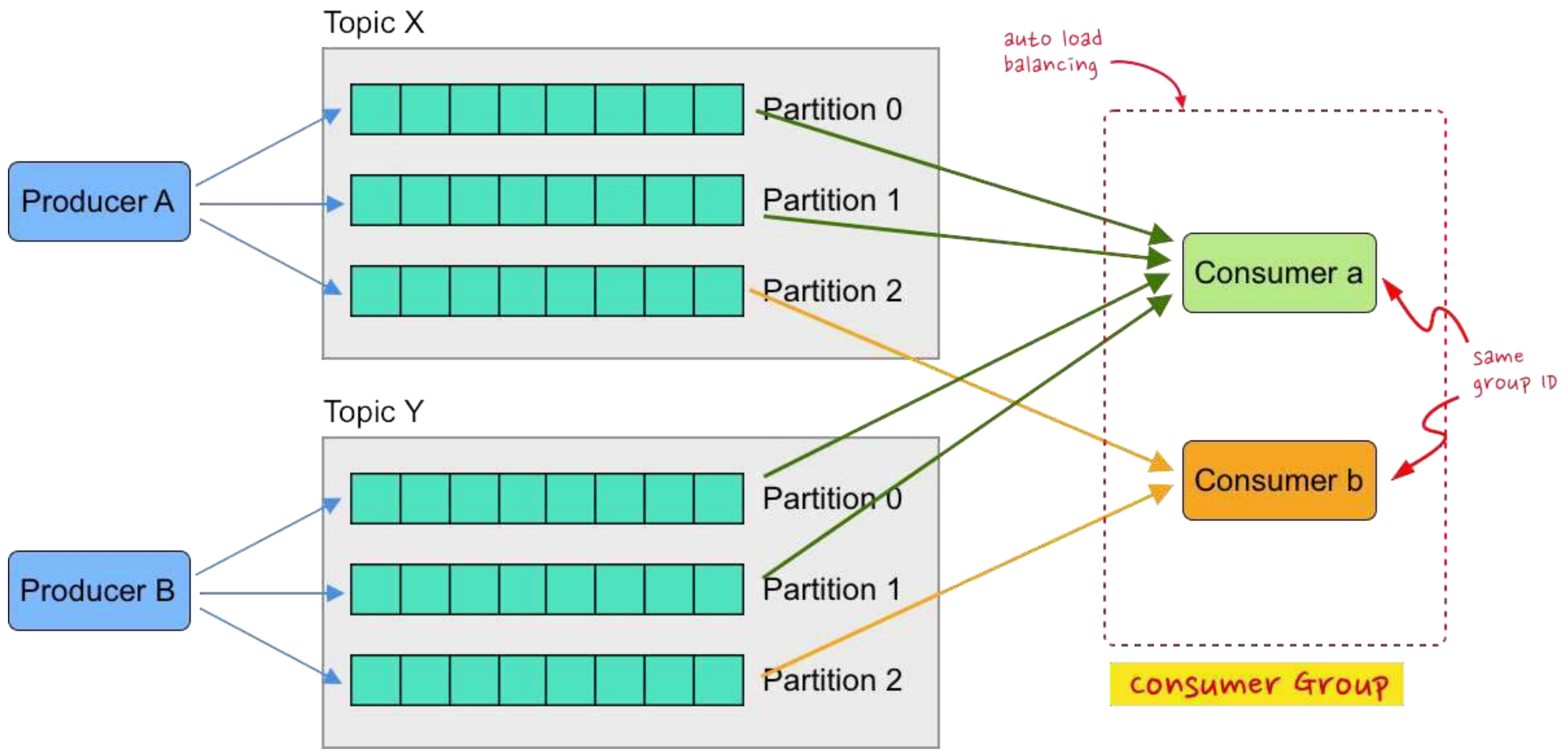
Partitioning Strategies

Why partitioning?

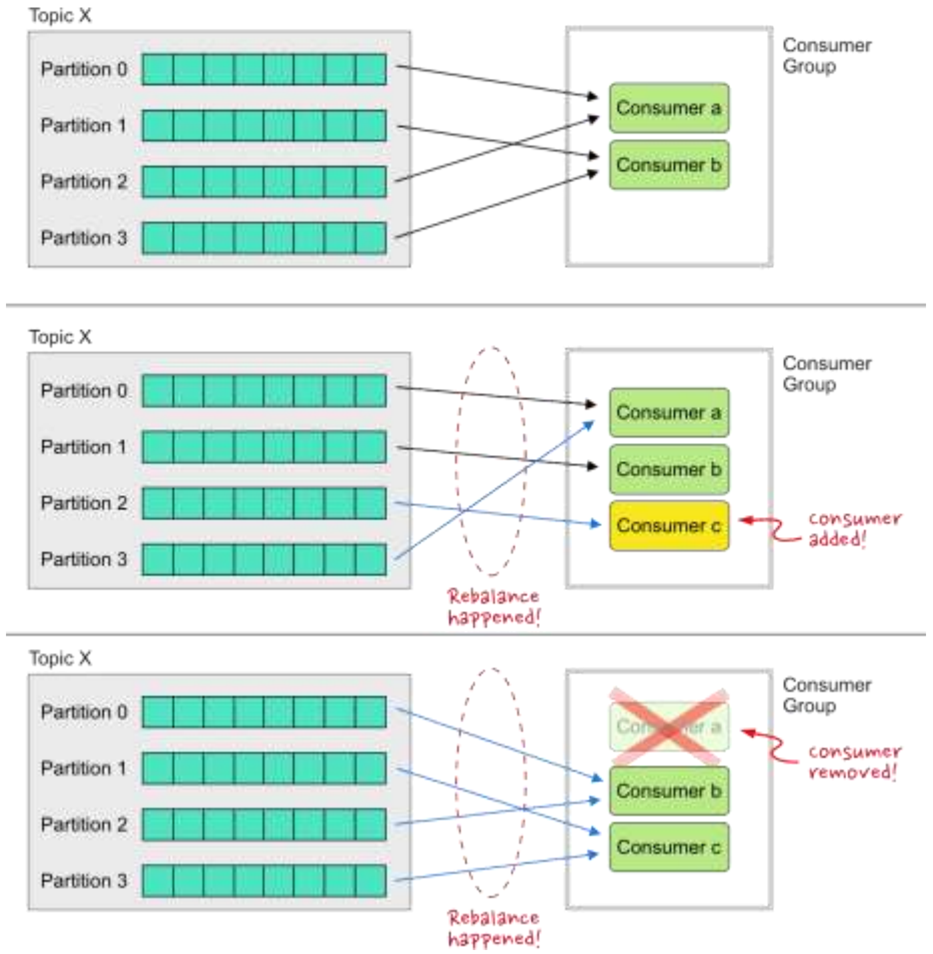
- Consumers need to **aggregate** or **join** by some key
- Consumers need **ordering guarantee**
- Concentrate data for **storage efficiency** and/or **indexing**



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

IDENTIFICATION CARD

Name:	Geoff Sample
D.O.B:	Area manager
ID No:	1238626AB4
Issued:	January 2011
Expires:	December 2013

WILLIAMS
Maintenance Services

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