WHITEPAPER

Apache Kafka: The Key concepts

PREPARED BY

SION SMITH

PUBLISHED

October 2024

Who are we?

Here at OSO, our mission is to help teams build event driven applications on Apache Kafka, enabling real-time access to your data and introducing meaningful innovations that drive business growth. Our developer-first culture, combined with our cross-industry experience and battle-tested delivery methods allow us to deliver the most impactful solutions through increased efficiency using our expertise in completing similar assignments for other large automotive institutions.

Founded in 2017, we set out to create a consultancy where customers get the benefits of a small specialist team that are dedicated to their success. As a team of engineers with a combined circa 50 years of expertise in Apache Kafka, event driven architecture and distributed platforms in general. Our vision is to enable businesses to greatly benefit from the power of these technologies which is why we have built a team of Subject Matter Experts (SMEs) who are solely focused on Apache Kafka.

Since being formed, we have worked with a number of well-established public sector clients in the UK. Case studies are available on our website at: https://oso.sh/customer-stories/

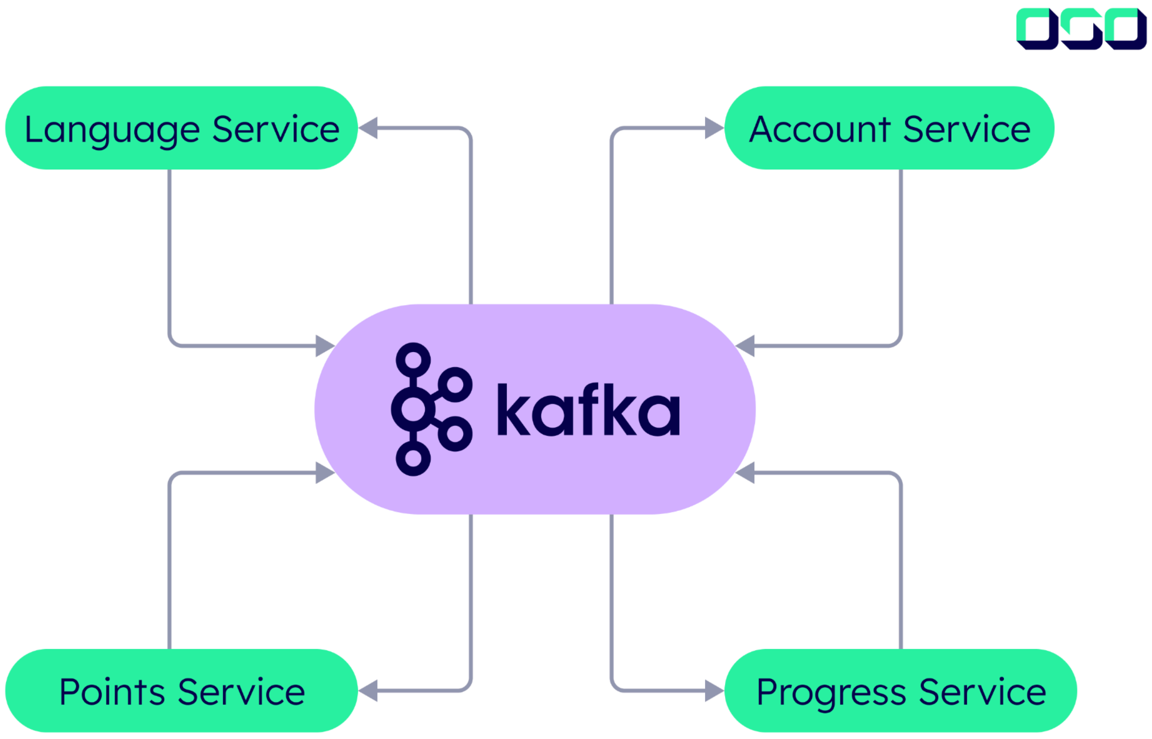
We are distributed computing experts and pride ourselves on being event driven technology evangelists. We are passionate about enabling businesses to make real time decisions.

Why Apache Kafka?

At OSO we are seen as the Kafka experts throughout most of Europe, we are the SAS team you call when you need to understand what is going on with your Apache Kafka clusters. Apache Kafka is widely used in modern systems for real-time data streaming and communication between different services. Apache Kafka powers mission-critical systems as the backbone for lightning-fast data streaming and seamless service-to-service communication.

Its robust architecture chews through massive data volumes with ease, making it the go-to solution for teams who need split-second data processing capabilities.

The aim of this document is to cover the basics and high level concepts which you should understand when adopting an event driven architecture. Below is a simple conceptual diagram of multiple microservices all interacting with each a single Apache Kafka cluster.



This diagram illustrates a microservices architecture using Apache Kafka as a central message broker/event streaming platform. The system consists of four distinct services that communicate with each other through Kafka:

- Language Service
- Account Service
- Points Service
- Progress Service

Kafka (shown in purple in the centre) acts as the central hub, with bidirectional communication flows (indicated by the grey arrows) between each service and Kafka. This architecture allows these services to:

- Operate independently
- Exchange data asynchronously
- Communicate without direct dependencies on each other
- Process events and messages in real-time

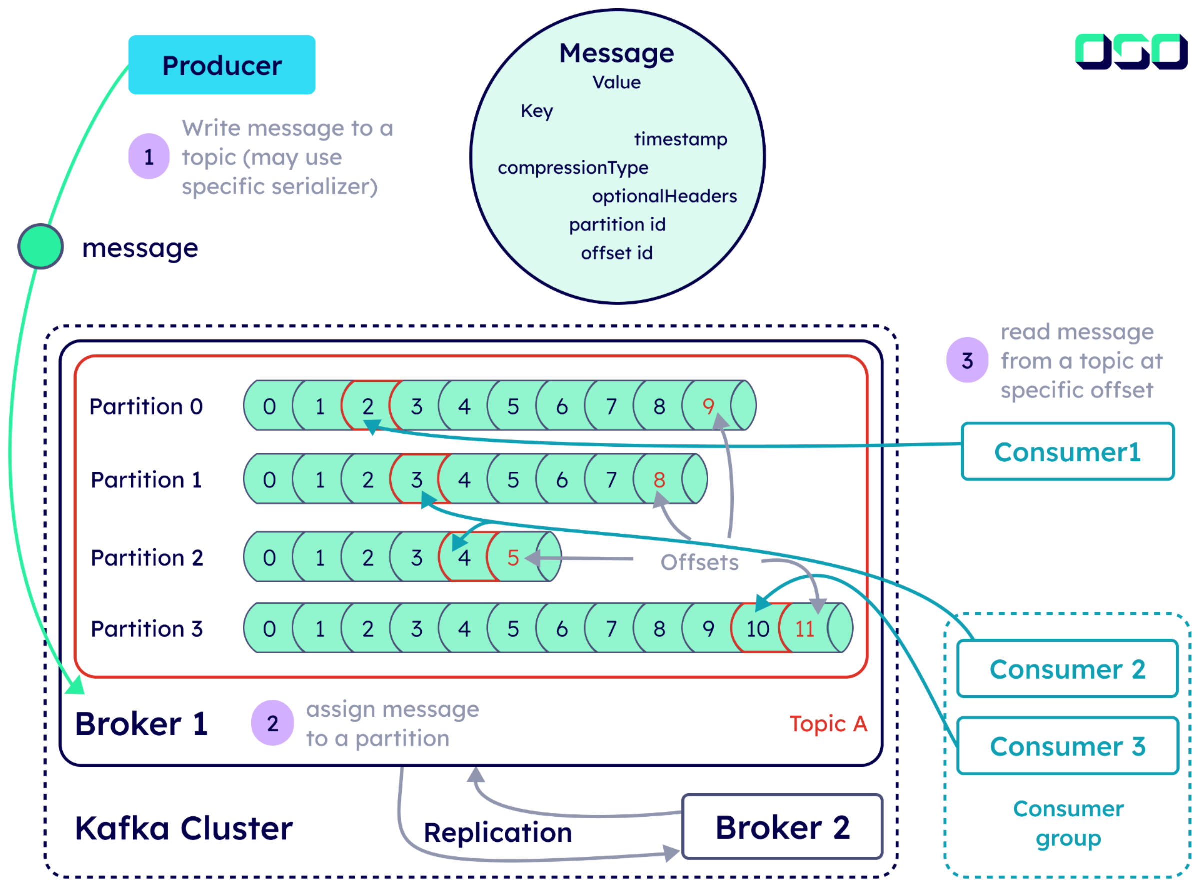
The design follows a typical event-driven architecture pattern where Kafka handles the routing and delivery of messages between services, enabling loose coupling and scalability. Each service can publish messages to Kafka topics and subscribe to relevant topics to receive updates from other services.

Key concepts and workflows

Think of a message as the fundamental building block of Kafka's world - it's the atom of data streaming. Like a universal container, a message is simply a string of bytes that Kafka treats as a blank canvas, not caring what artistic masterpiece of data you've painted inside.

Sometimes, you can tag your message with a special marker called a key - think of it as a smart postal code that helps guide your messages to specific neighbourhoods called partitions. For maximum efficiency, Kafka bundles messages together like a carefully packed shipping container, grouping similar packages heading to the same destination and partition. Each partition works like a giant scroll - new messages are written at the end, and when you read them, you start at the beginning and work your way through, like reading an ancient manuscript.

While a topic is like a city containing many neighbourhood (partitions), remember that only within each individual neighbourhood can you guarantee messages will arrive in the exact order they were sent. Partitions are how Kafka provides scalability and redundancy.



Think of a broker as a powerful vault keeper - each partition can live in a different vault (server), letting your topic flex and grow across a network of secure facilities as your data empire expands.

Like a savvy investor diversifying their portfolio, partitions can be replicated across multiple servers - creating backup copies that stand ready to jump into action if disaster strikes one location.

A stream is the grand river of your data, flowing endlessly from source to destination. Whether it splits into multiple channels (partitions) or not, it's all one mighty current of information surging from producers to consumers.

Kafka Clients

In Kafka's ecosystem, clients play two starring roles: **producers**, the storytellers who create, and **consumers**, the audience who listens. Producers are like authors, crafting fresh messages and publishing them to their chosen topic.

Consumers are like meticulous readers, devouring messages chapter by chapter, keeping their place with special bookmarks called **offsets** - ensuring they never lose track of what they've read in each storyline.

Think of an **offset** as nature's perfect page number - a never-decreasing counter that Kafka stamps on each message as it arrives. Just as a reader can slide their bookmark into place before taking a break, consumers can pause and resume their reading adventure without missing a single word of the story.

Consumers band together in **consumer groups** - imagine a book club where members divide up chapters of a massive novel. Like well-organised readers, the group ensures no two members accidentally read the same chapter simultaneously. This clever system of ownership means if one book club member falls asleep, their fellow readers smoothly pick up their assigned chapters - ensuring no part of the story goes unread.

Kafka Infrastructure

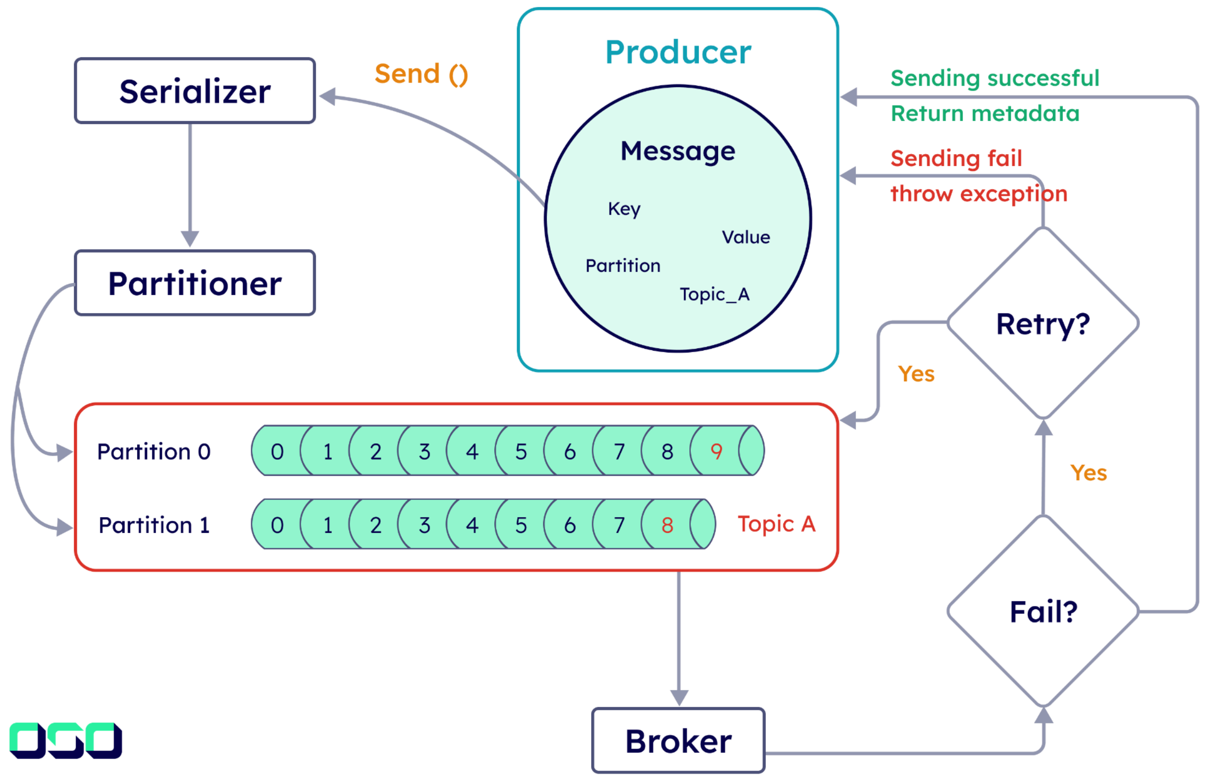
Think of a Kafka broker as a digital librarian - a standalone guardian that carefully catalogues incoming messages, stamps them with precise locations, and archives them in its vast storage halls. This masterful curator also serves as a reference desk, efficiently retrieving requested volumes for knowledge-hungry consumers.

Like any world-class institution, Kafka's library system operates as a sophisticated network of interconnected branches. Within this network, each partition answers to a single authority - the leader broker, much like a department head overseeing their specialised collection.

For safeguarding precious data, mirror collections are maintained by **follower** brokers, standing ready like trained successors to step up if their department head steps down. This careful replication strategy ensures no knowledge is lost - if one library branch closes, another seamlessly opens its doors to protect the continuous flow of information.

While new manuscripts must be submitted to the department head, scholars can access their research materials from either the head librarian or their trusted deputies. Like a modern archive, Kafka doesn't keep its collections forever - it employs smart preservation policies.

Each library branch follows preset **retention** guidelines, maintaining its collections based on either time limits or storage space constraints - much like a rotating exhibition schedule.



Shipping data to Kafka starts with preparing your package - you'll need the destination topic (like a shipping address) and the actual contents. For premium service, you can add special handling instructions: a sorting key, specific warehouse number, timestamp, or custom shipping labels.

Before your package hits the road, the producer acts like a professional packing service, carefully converting your key and value contents into standardised shipping units that can zip through Kafka's delivery network.

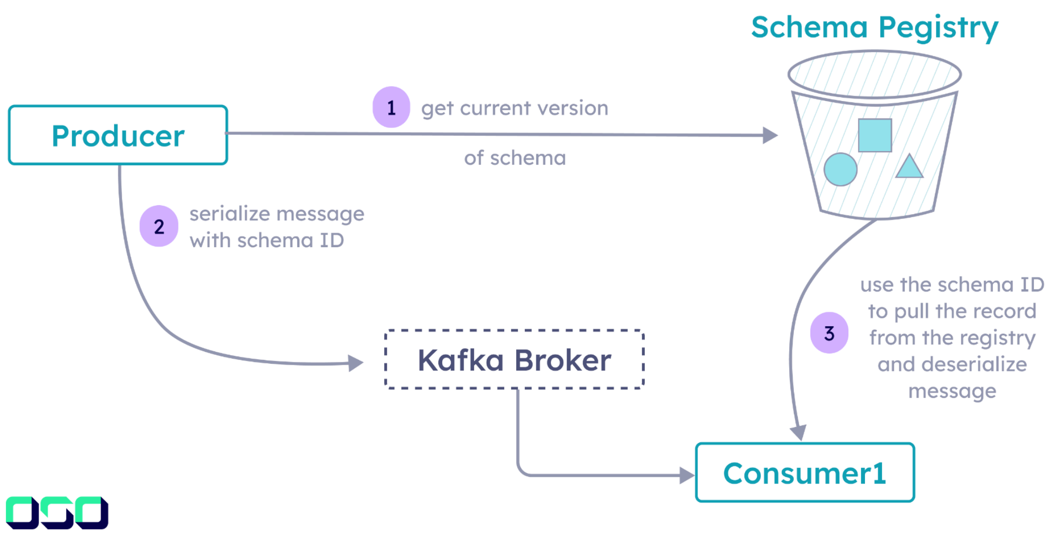
When you don't specify a particular warehouse, your package goes through our smart sorting facility (the partitioner), which chooses the optimal storage location, usually based on your package's sorting key.

Once your shipment reaches the broker's distribution centre, you'll receive a detailed delivery receipt - think of it as a digital tracking number that tells you exactly where to find your package: the destination hub (topic), the specific warehouse (partition), and its precise shelf location (offset).

Message Formats

Kafka can accept many different formats of message as long as they can be serialised down to bytes. Think of Apache Avro as the universal translator in your data embassy - speaking a diplomatic language that all systems understand through serialisation. At the heart of many data kingdoms stands a master blueprint library, the Schema Registry, safeguarding the official templates for all data structures. While this architectural marvel shares the neighbourhood with Kafka, it maintains its own sovereign territory.

Picture the Schema Registry as the central blueprint office, archiving every approved design pattern used to construct messages in the Kafka ecosystem. By using well-defined schemas stored in a shared repository, Kafka messages can be interpreted without the need for coordination.


Schema Pegistry
Producer
1 get current version of schema
2 serialize message with schema ID
Kafka Broker
3 use the schema ID to pull the record from the registry and deserialize message
Consumer1