Event Management & Kafka

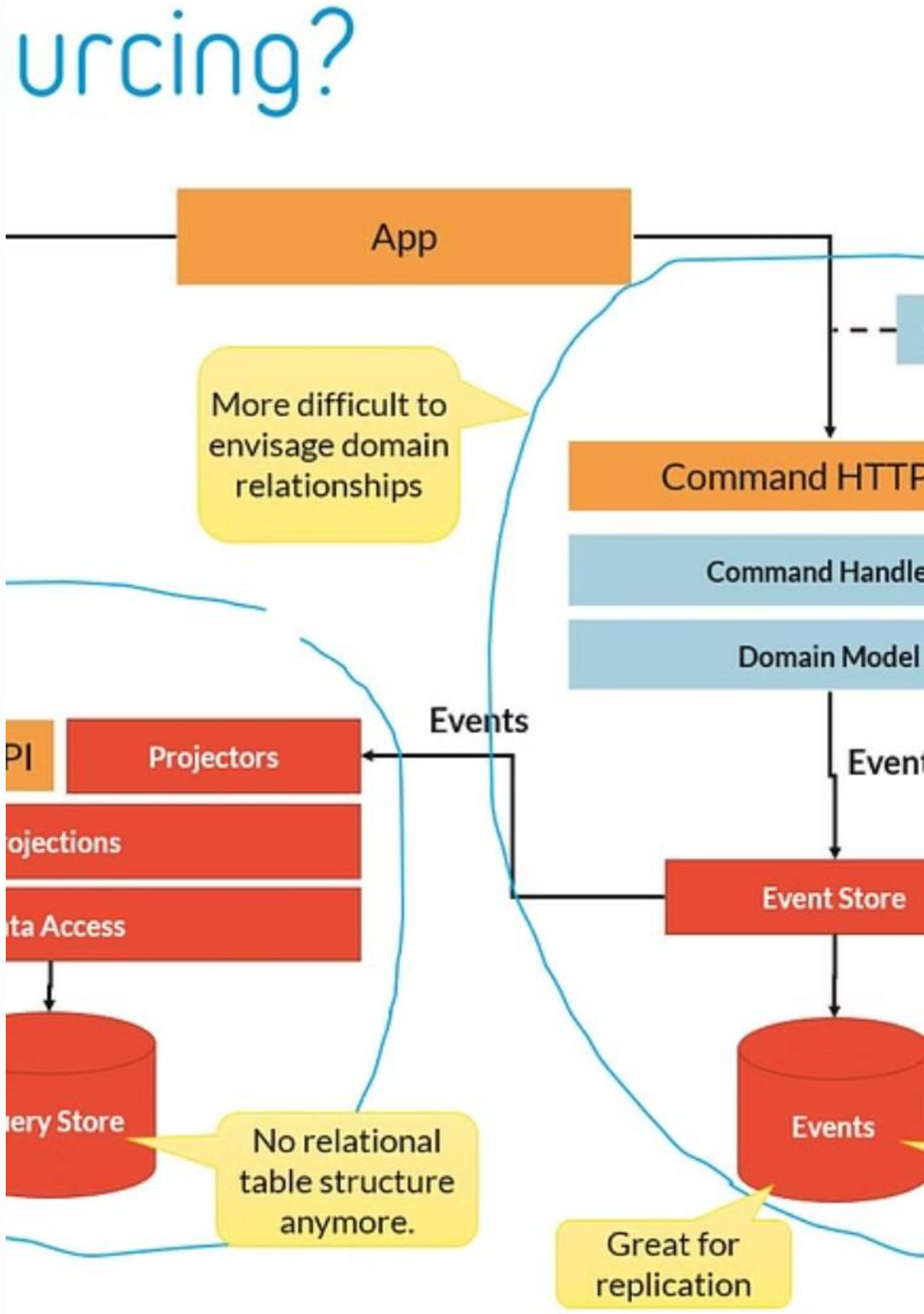
a

Introduction to Event Management

Event Management is a crucial process in today's fast-paced digital world. It involves capturing, processing, and analyzing events to gain valuable insights and drive business decisions. In this card, we'll explore the fundamentals of Event Management and its importance in optimizing operations and enhancing customer experiences.

Introduction to Event Sourcing

Event sourcing is a crucial architectural pattern that captures all changes to an application state as a sequence of events. Unlike traditional data storage methods, event sourcing doesn't store the current state of an entity. Instead, it keeps a log of all changes, enabling a comprehensive and granular view of past actions. This provides valuable historical context and an audit trail for the data, which is essential in complex applications and systems.



Benefits of Event Sourcing

1

Scalability

Event sourcing facilitates scalability by allowing systems to handle a vast number of events efficiently. This can be advantageous in scenarios with high-throughput systems and distributed architectures.

2

Auditability

By keeping a comprehensive log of events, event sourcing provides a transparent and immutable history of all actions taken within an application, supporting regulatory compliance and audit requirements.

3

Flexibility

It offers the flexibility to rebuild application state at any point in time, enabling easy debugging, analysis, troubleshooting, and historical view of the data.

Event Management and Its Importance

Real-time Responsiveness

Event management allows systems to react and respond instantly to changes or events, enabling real-time decision-making capabilities.

Data Integration

By efficiently managing events, organizations can seamlessly integrate data from various sources, creating a cohesive and comprehensive view of information.

Enhanced Monitoring

Event management facilitates improved monitoring and analysis of system behaviors, performance, and trends, helping organizations to make data-driven decisions and optimizations.

Key Components of Event Management

Event Bus

The event bus serves as the backbone, enabling asynchronous communication of events between different components within a system.

Event Processing

Event processing mechanisms handle the ingestion, enrichment, and transformation of incoming events, ensuring efficient and meaningful usage of the data.

Event Handlers

These are responsible for reacting to specific events, triggering appropriate actions, and executing necessary tasks based on received events.

Event-Driven Architecture

1

Events as First-Class Citizens

Event-driven architecture places a strong emphasis on events, enabling them to act as first-class citizens within the system and driving the core of the application logic.

2

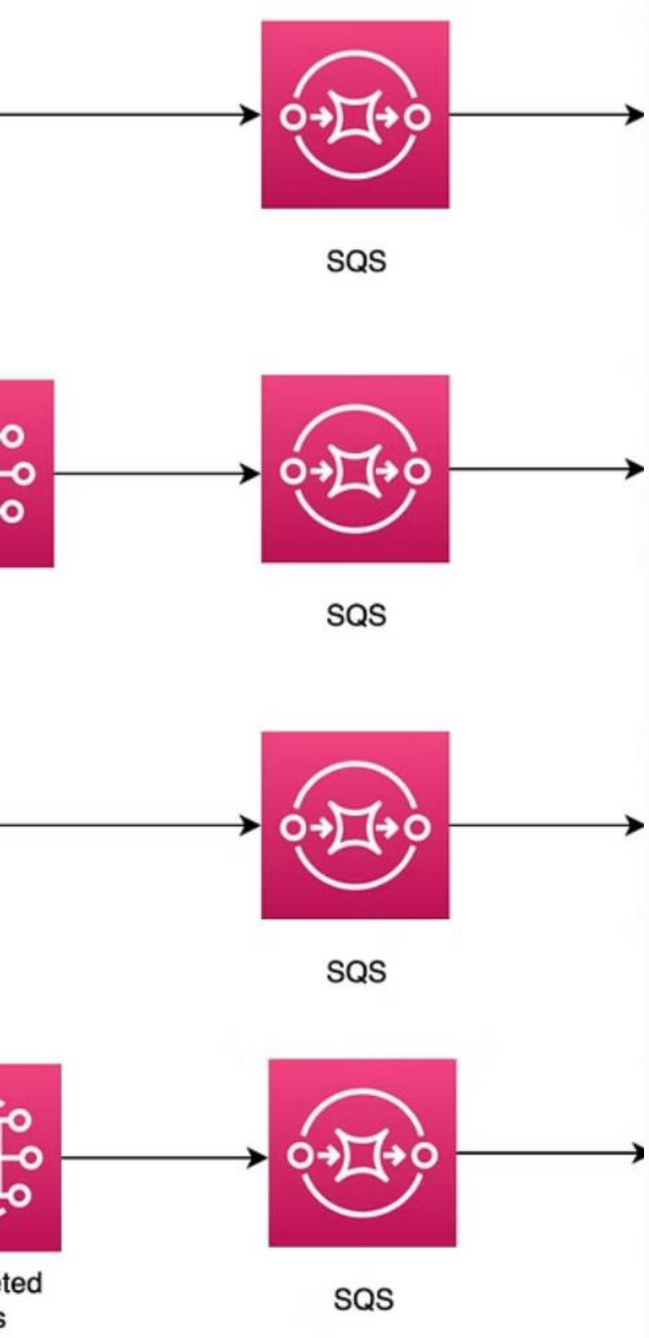
Asynchronous Communication

The architecture promotes asynchronous communication between different components, providing flexibility, decoupling, and scalability, essential for modern, distributed systems.

3

Loose Coupling

By decoupling the producers and consumers of events, event-driven architecture ensures a more resilient and flexible system, capable of handling change and evolution efficiently.



Event-Driven Design Patterns

1 Publish-Subscribe

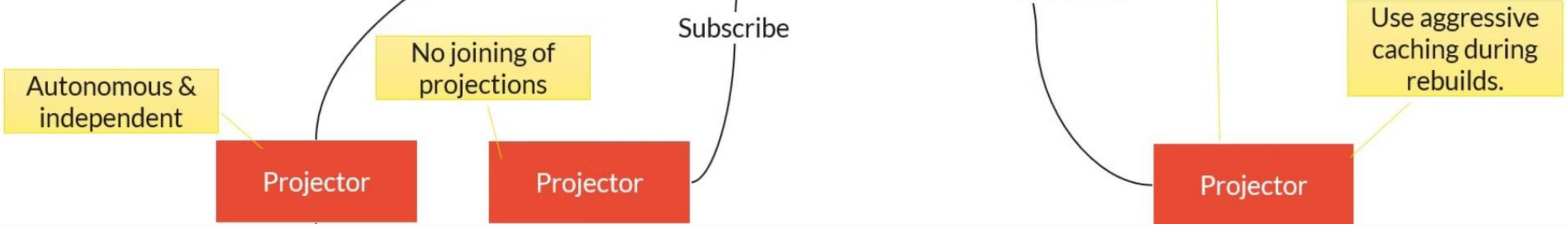
The publish-subscribe pattern enables loose coupling and scalability by allowing multiple subscribers to receive events generated by publishers without direct dependencies.

2 Message Queues

Message queuing systems facilitate the distribution and asynchronous processing of events, helping to manage system complexity and enable smooth inter-component communication.

3 Event Carried State Transfer

This pattern emphasizes transmitting the entire state of an entity within an event, enabling systems to reconstruct the entity's state following an event.



Use Cases and Applications of Event Sourcing

1

Financial Systems

Event sourcing is utilized in financial systems to provide an immutable record of transactions and facilitate auditing, compliance, and analytics.

2

Supply Chain Management

It is employed to track and manage supplies, shipments, and inventory, providing a transparent and auditable history in complex supply chain ecosystems.

3

Healthcare Data Management

Event sourcing offers a robust method for managing and analyzing patient data, ensuring data integrity, and supporting compliance with healthcare regulations.

In Real-Life Usage of Event Sourcing

1

E-commerce Order Fulfillment

Event sourcing can be used in e-commerce platforms to track and process order events, ensuring accurate order fulfillment and providing a complete audit trail of order history.

3

Banking Transaction Processing

Event sourcing is beneficial in banking systems to handle transaction events, maintaining an immutable record of every transaction for auditing purposes and enabling efficient reconciliation.

Event Management

- **Apache Kafka Apache**
- **Pulsar**
- **RabbitMQ**
- **Amazon Simple Queue Service (SQS)**
- **Google Cloud Pub/Sub**
- **Microsoft Azure Service Bus**
- **IBM MQ**

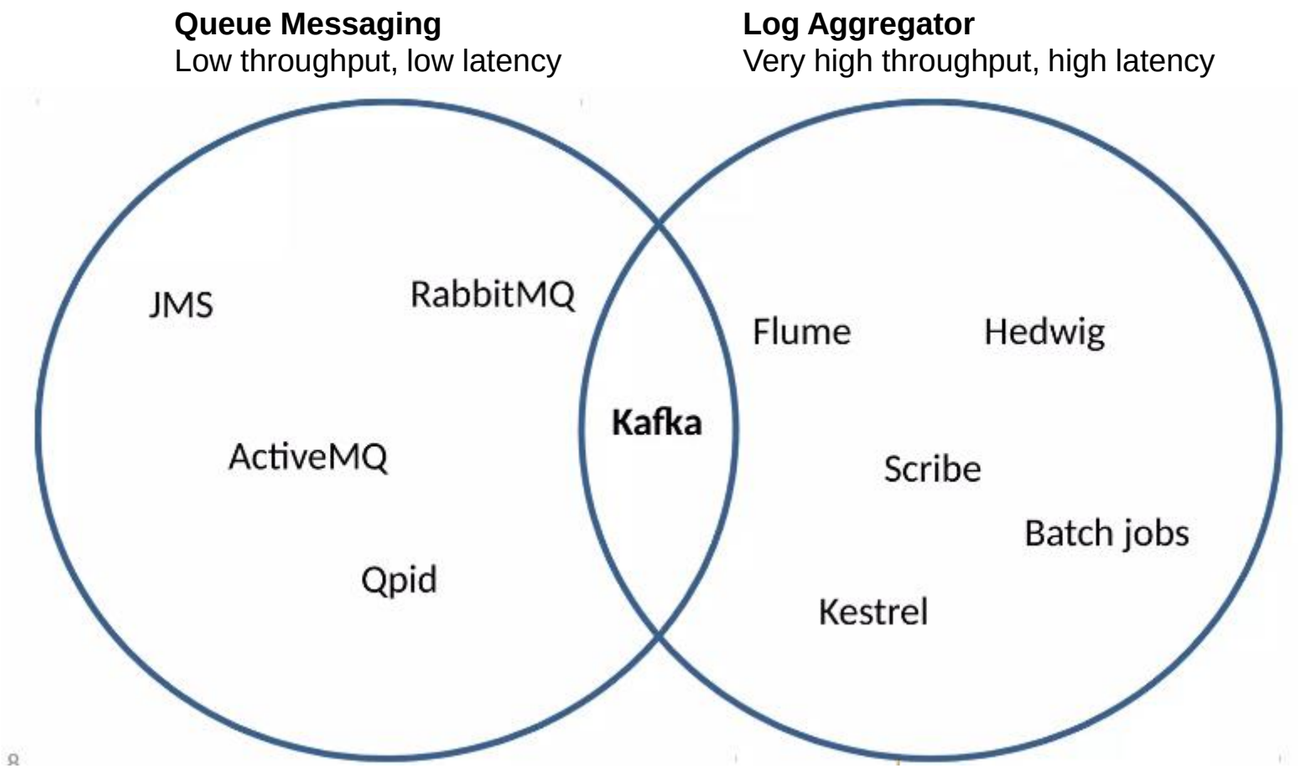
Event Management

Service	Description	Use Cases
Apache Kafka	Distributed streaming platform that focuses on high-throughput, fault-tolerant messaging, and real-time data processing.	Real-time data ingestion and processing, event streaming, log aggregation, messaging systems, and more.
RabbitMQ	Open-source message broker that implements the Advanced Message Queuing Protocol (AMQP) and provides reliable message delivery.	Enterprise messaging, task and workflow management, routing and filtering, and more.
Amazon Simple Queue Service (SQS)	Fully managed message queuing service that enables decoupling and scaling of distributed systems on the AWS platform.	Cloud-based messaging, decoupling of microservices, event-driven compute, and more.
Google Cloud Pub/Sub	Globally distributed messaging service that allows asynchronous communication between independently developed applications.	Real-time messaging, event-driven architectures, decoupling of microservices, and more.

Event Management

- **Apache Kafka**: Distributed streaming platform that focuses on high-throughput, fault-tolerant messaging, and real-time data processing.
- **Apache Pulsar**: Pub-sub messaging system that offers scalability, durability, and low-latency messaging, with support for both streaming and queuing models.
- **RabbitMQ**: Open-source message broker that implements the Advanced Message Queuing Protocol (AMQP) and provides reliable message delivery.
- **Amazon Simple Queue Service (SQS)**: Fully managed message queuing service that enables decoupling and scaling of distributed systems on the AWS platform.
- **Google Cloud Pub/Sub**: Globally distributed messaging service that allows asynchronous communication between independently developed applications.
- **Microsoft Azure Service Bus**: Fully managed enterprise messaging service that provides asynchronous communication and decoupling for cloud-based applications.
- **IBM MQ**: Messaging middleware that facilitates communication and integration between applications, systems, and services across various platforms.

Message Queue Management Services



What is Kafka?

- Kafka is an open-source distributed event streaming platform
- It provides a highly scalable, fault-tolerant, and real-time data processing solution
- It allows organizations to efficiently capture, store, and process large volumes of streaming data, enabling seamless integration and real-time analytics

Concept of Kafka and its Specifications

Scalability

Kafka is designed to handle high volumes of data and supports horizontal scaling by distributing data across multiple brokers.

Fault tolerance

Kafka ensures fault tolerance by replicating data across multiple brokers, allowing for automatic failover in case of node failures.

Real-time processing

Kafka provides low-latency message delivery, making it suitable for real-time data processing and stream processing applications.

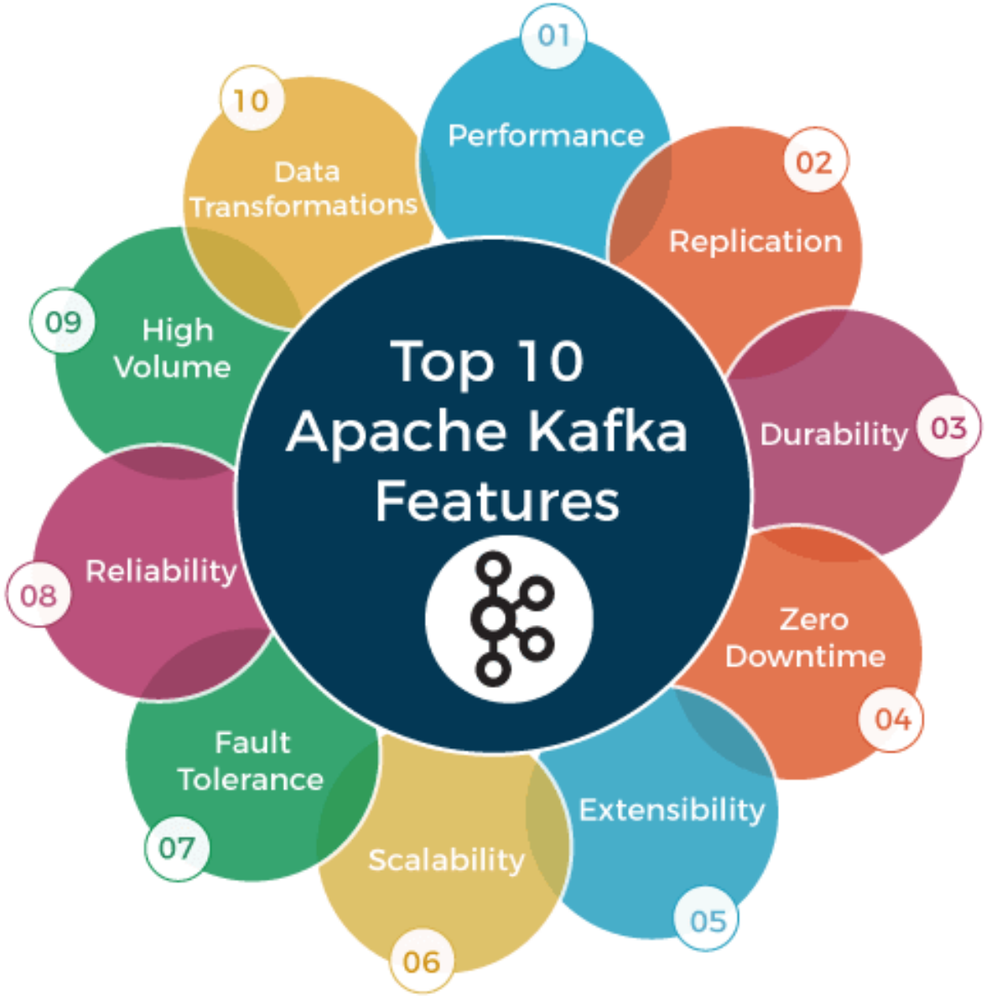
Efficient storage

Kafka stores data in a distributed and durable manner, enabling efficient data retention and retrieval.

Integration capabilities

Kafka integrates seamlessly with various data systems, databases, and frameworks, making it versatile for data integration and pipeline management.

Concept of Kafka and its Specifications



Benefits of Kafka for Event Management

1

Scalability

Kafka's architecture allows for seamless scalability, making it an ideal choice for handling a massive influx of events with ease.

2

Reliability

With features like fault-tolerance and high availability, Kafka ensures reliable event processing and delivery.

3

Real-time Processing

Kafka enables real-time event processing, providing insights and actions based on the most recent data.

What is Kafka?

Distributed Pub-Sub System

Distributed, high throughput, publish/subscribe (pub-sub) messaging system

High Speed and Robust

Fast, Robust, Scalable, Durable

Key Use Cases

Main use cases: Log aggregation, Real-time processing, Monitoring, Queues

Origin at LinkedIn

Originally developed at LinkedIn

Kafka Clusters Architecture: 5 Major Components

01. Topics

02. Brokers

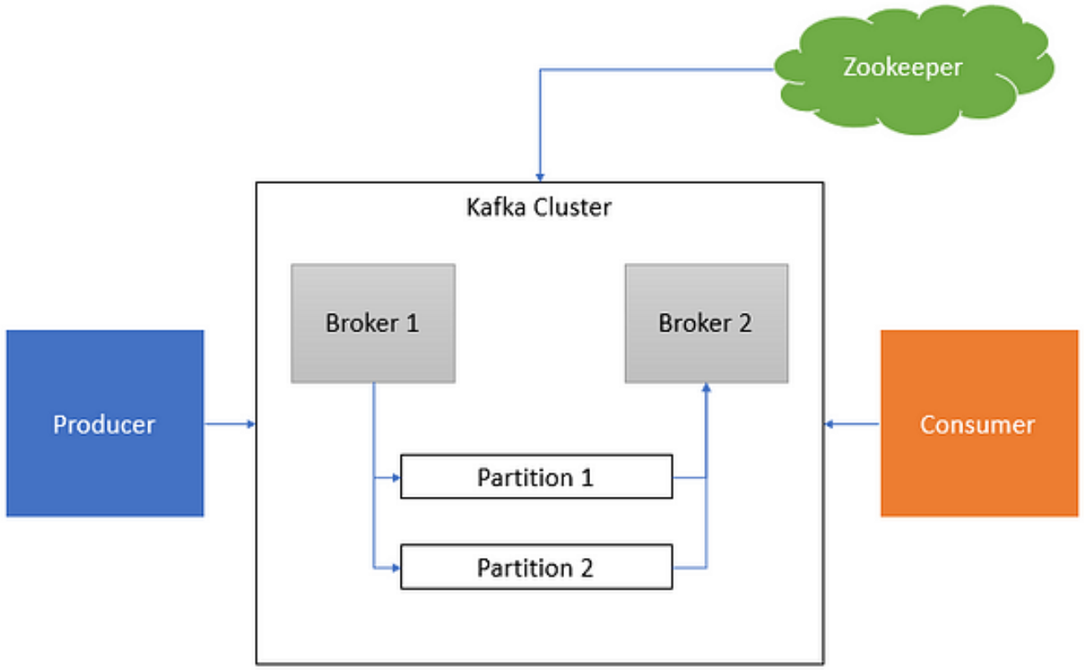
03. ZooKeeper

04. Producers

05. Consumers

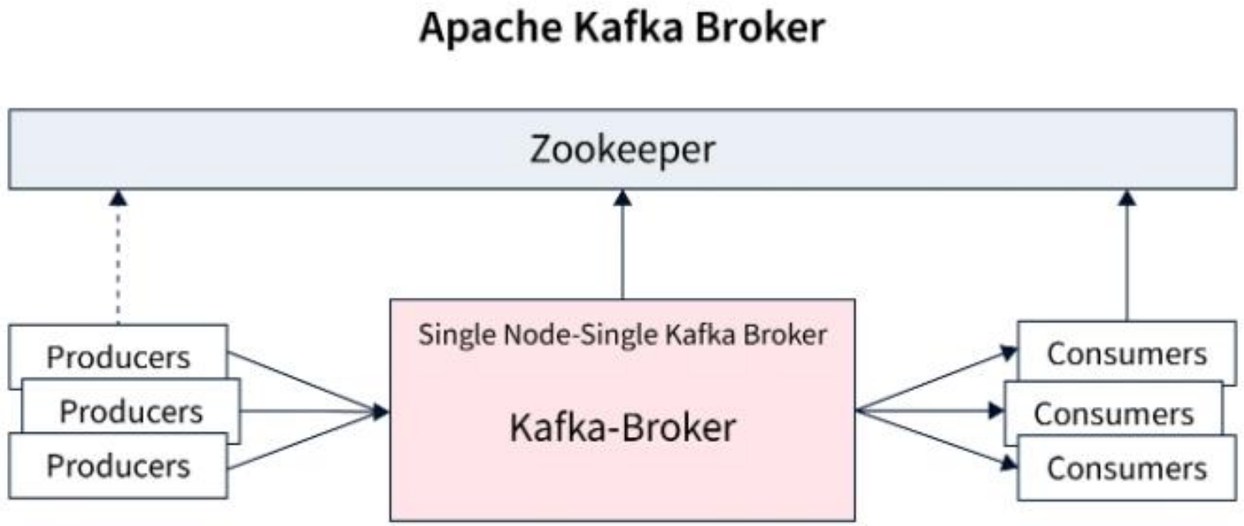
Kafka Architecture

- **Zookeeper**: Coordination service for discovery and registration of Kafka components
- **Kafka Cluster**: Set of servers called brokers that store and distribute messages
- **Producer**: Client that sends messages to Kafka topics
- **Partition**: Division of a Kafka topic to distribute messages across multiple brokers
- **Consumer**: Client that reads messages from one or more Kafka topics



Kafka Architecture

- **Zookeeper**: Coordination service for discovery and registration of Kafka components
- **Kafka Cluster**: Set of servers called brokers that store and distribute messages
- **Producer**: Client that sends messages to Kafka topics
- **Partition**: Division of a Kafka topic to distribute messages across multiple brokers
- **Consumer**: Client that reads messages from one or more Kafka topics
- **Brokers :** Brokers are the servers that make up a Kafka cluster. They are responsible for storing and transmitting the messages within the group.



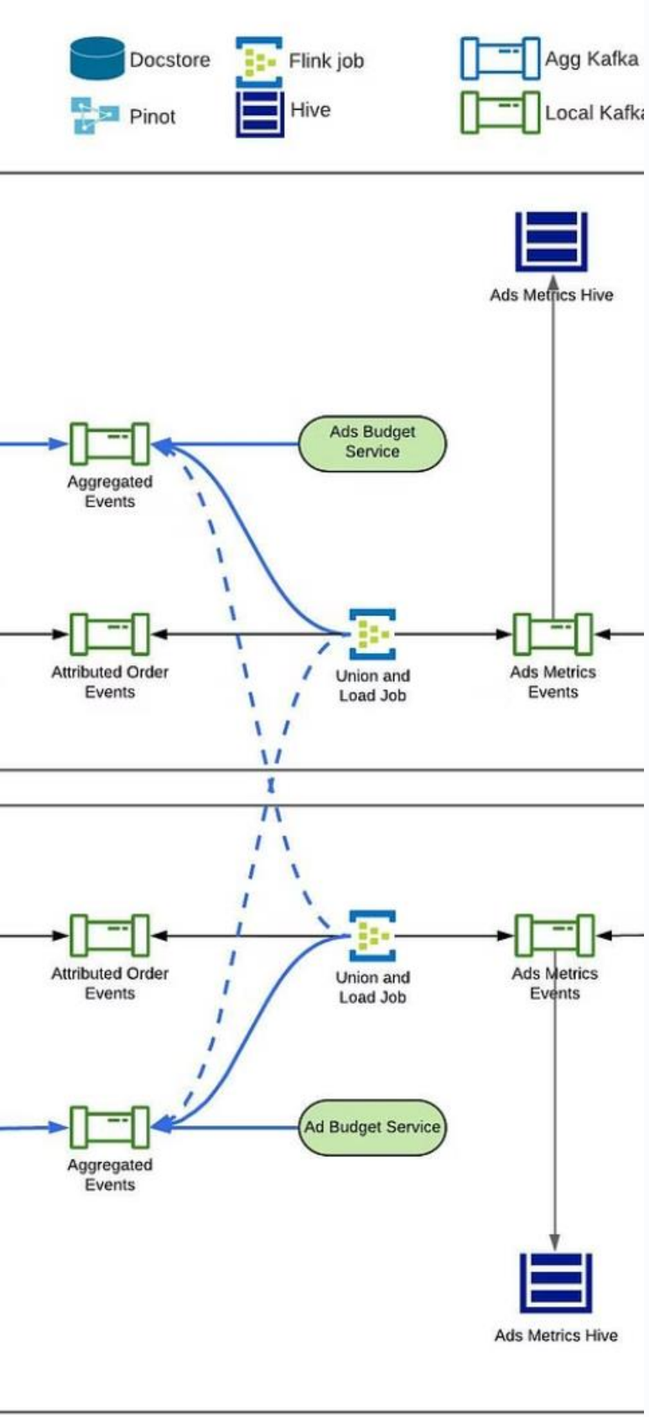
Understanding Kafka's Event Management Architecture

Core Components

Learn about the core components, such as producers, topics, partitions, and consumers.

Reliability, Scalability, and Fault-Tolerance

Explore the reliability, scalability, and fault-tolerance features that make Kafka an ideal choice for handling event-driven data streams.



Introducing Kafka: Powering Real-Time Event Processing

1

Kafka Revolutionizes Event-Driven Architectures

Kafka, a distributed streaming platform, revolutionizes the way organizations handle event-driven architectures.

2

High Throughput and Fault-Tolerant Design

With its high throughput, fault-tolerant design, and real-time processing capabilities, Kafka enables businesses to handle massive volumes of events with ease.

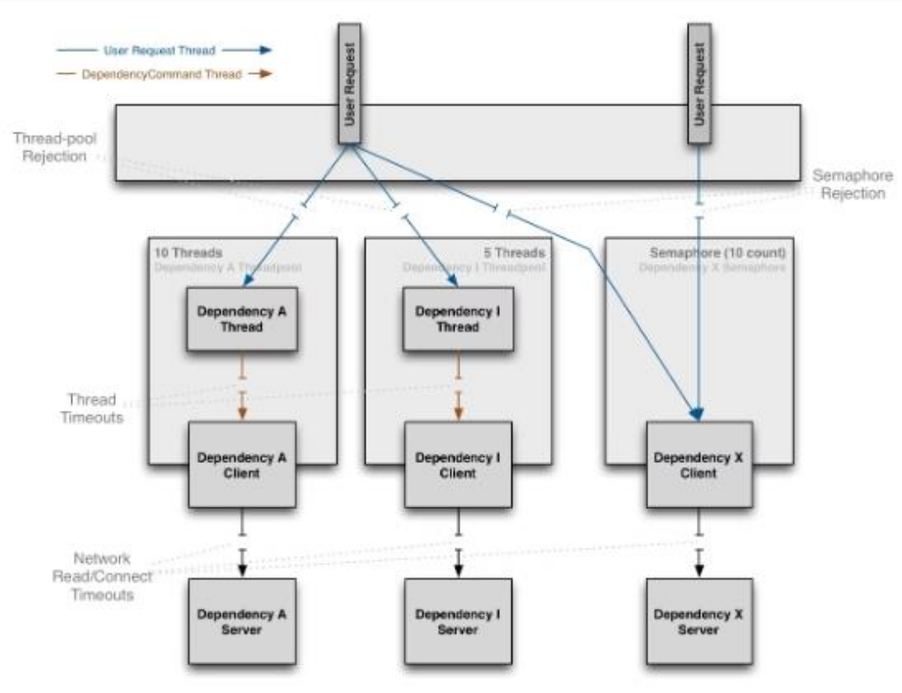
3

Empowering Event-Driven Applications

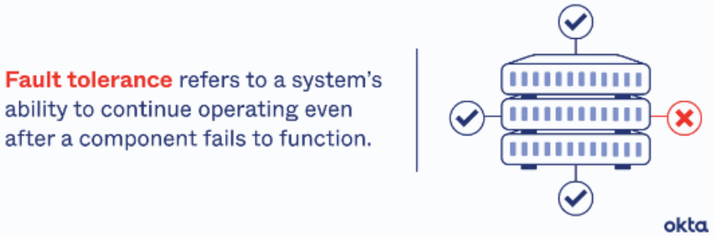
In this card, we'll explore how Kafka empowers event-driven applications and drives business success.

Comparing Kafka with Other Technologies

Kafka Provides high throughput, fault-tolerance, and real-time data processing.



Other Technologies May lack the robust capabilities and fault-tolerance found in Kafka, impacting event management.



Comparing Kafka with Other Technologies

Comparison of Kafka with Other Brokers

Scalability

Designed for handling high volumes of data and supports horizontal scaling.

Fault Tolerance

Replicates data across multiple brokers for automatic failover.

Real-time Processing

Enables low-latency message delivery for real-time data and stream processing.

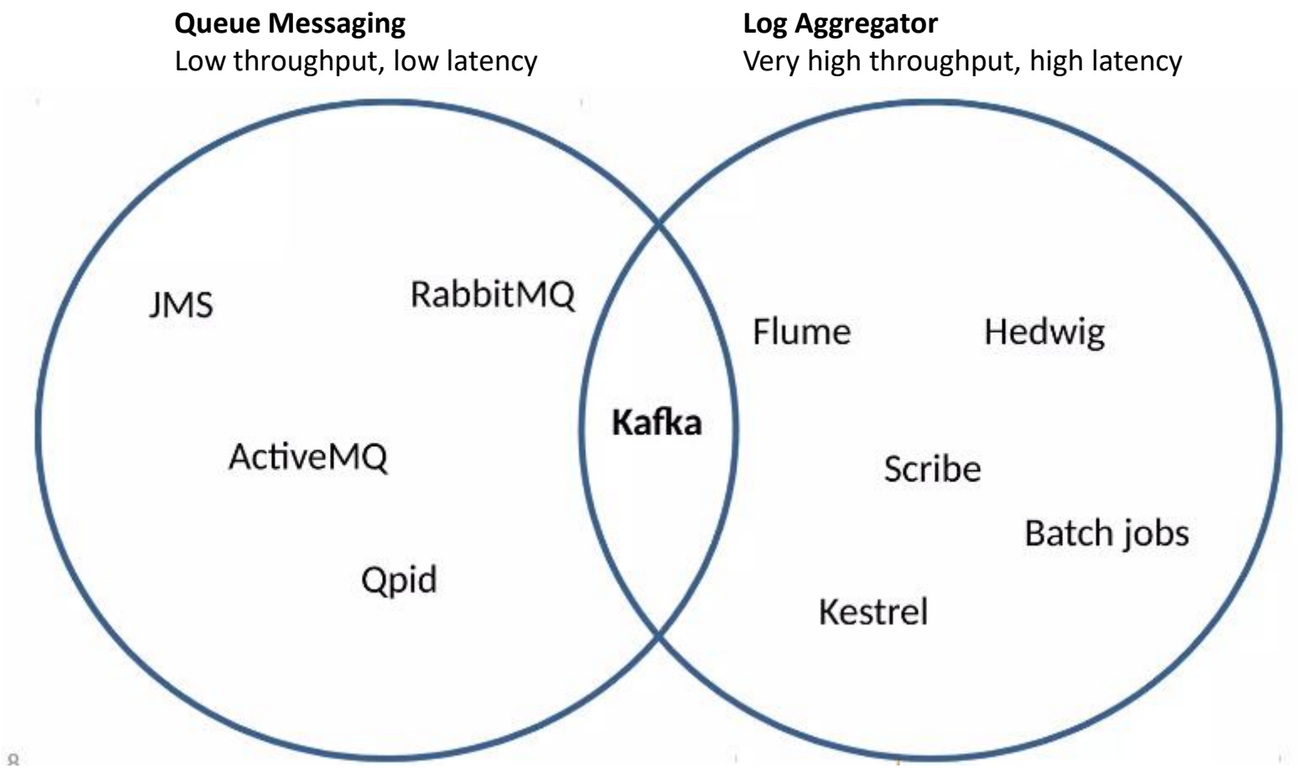
Storage Efficiency

Stores data in a distributed and durable manner for efficient retention and retrieval.

Integration Capabilities

Integrates seamlessly with various data systems, databases, and frameworks.

Comparison of Kafka with Other Brokers



Comparison of Kafka with Other Brokers

Feature	**Kafka**	**ActiveMQ**	**RabbitMQ**
Scalability	Designed for handling high volumes of data and supports horizontal scaling.	Can scale horizontally but may face performance issues with large data sets.	Can scale horizontally but may require additional configurations for high scalability.
Fault Tolerance	Replicates data across multiple brokers for automatic failover.	Provides fault tolerance but may require manual configuration for replication.	Provides fault tolerance through clustering and replication.
Real-time Processing	Enables low-latency message delivery for real-time data and stream processing.	Supports real-time processing but may have higher latency compared to Kafka.	Supports real-time processing but may have higher latency compared to Kafka.
Storage Efficiency	Stores data in a distributed and durable manner for efficient retention and retrieval.	Stores data but may have limitations on storage capacity and performance.	Stores data but may have limitations on storage capacity and performance.
Integration Capabilities	Integrates seamlessly with various data systems, databases, and frameworks.	Offers good integration capabilities but may require additional configurations.	Offers good integration capabilities but may require additional configurations.

Cas d'utilisation

LinkedIn

Activity stream, operational metrics tracking, data bus

OVH

Anti-DDOS (OVH)

Netflix

Real-time tracking, real-time processing (Netflix)

Twitter

Component of their real-time architecture

Spotify

Log processing (from 4am to 10am)

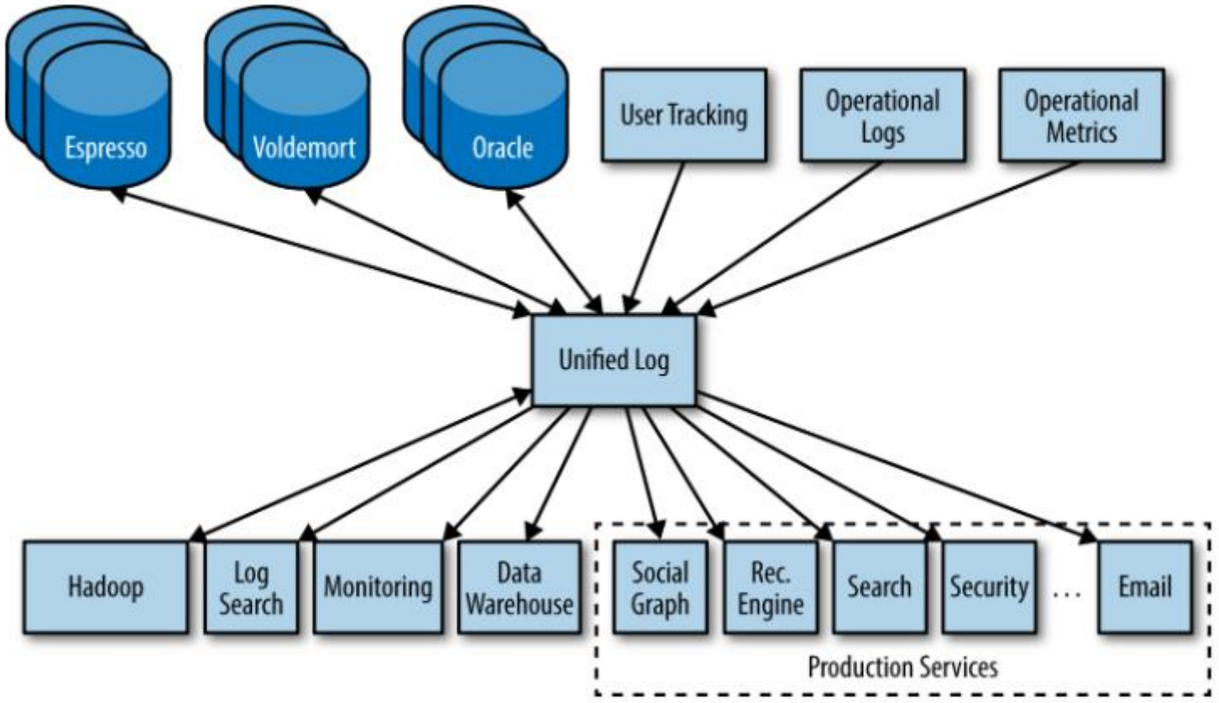
Mozilla

Metric management

Performance - LinkedIn

- 15 Brokers
- 15,500 partitions (replication factor 2)
- Input:
  - 400,000 messages/second
  - 70 MB/second
- Output:
  - 400 MB/second

Performance - LinkedIn



LinkedIn

kafka

Performance - LinkedIn

Data Ingestion:
- Unified Log: Kafka is used to collect and store unified event logs from the system.
- Monitoring Data: Kafka is used to collect and store system monitoring data.

Data Processing:
- Espresso: Kafka is used to stream data to the Espresso data processing platform.
- Voldemort: Kafka is used to stream data to the Voldemort NoSQL data storage system.
- Hadoop: Kafka is used to stream data to the open-source Hadoop data processing platform.

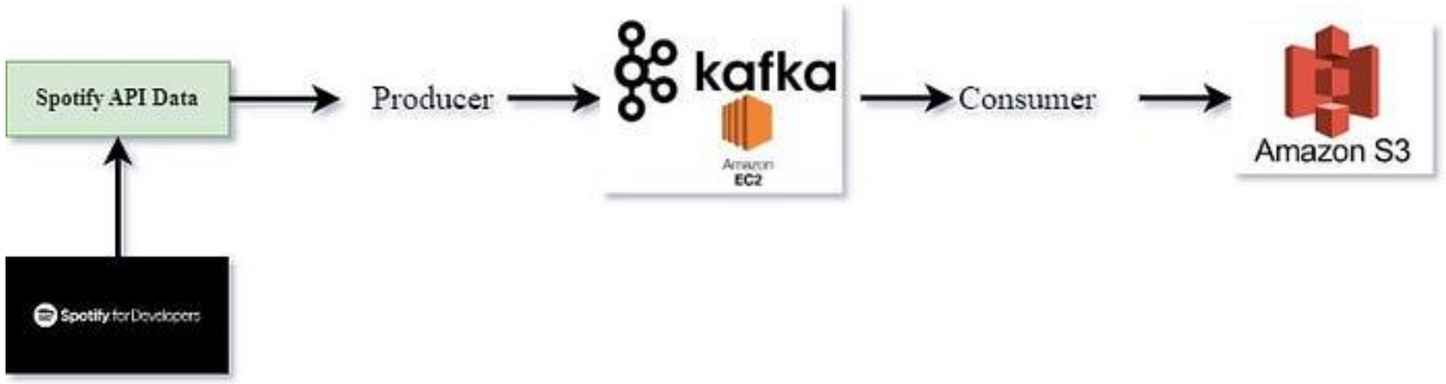
Data Analysis:
- Log Search: Kafka is used to stream logs to the log search and analysis service.
- Recommendation Engine: Kafka is used to stream data to the recommendation engine for job postings and training.

Production Services:
- Production Services: Kafka is used to stream data to LinkedIn's production services.

LinkedIn

kafka

Kafka - Spotify: Real-world Utilization



Spotify

kafka

Kafka - Spotify: Real-world Utilization

Discover how Spotify leverages Kafka to handle millions of music streams and provide real-time recommendations to users, ensuring a seamless music streaming experience.

Example of Data:

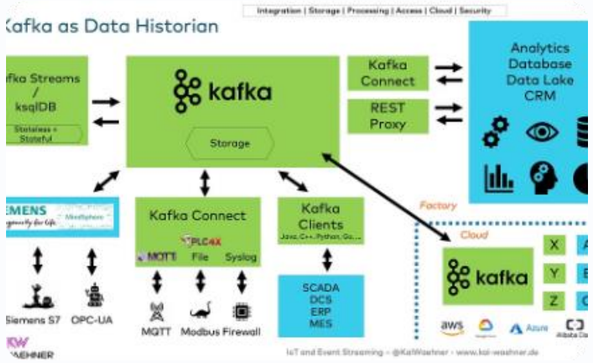
Spotify's data includes information about user preferences, song metadata, and streaming activity.

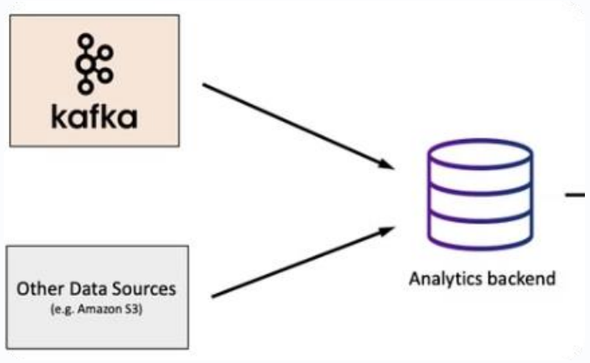
Architecture:

Spotify's architecture involves multiple Kafka clusters for data ingestion, streaming, and processing. The data flows through various components, such as producers, topics, brokers, and consumers, enabling real-time data processing and recommendation generation.

Performance - Spotify

Kafka - Spotify





Music Streaming Integration

Kafka is integrated with Spotify for efficient music streaming and real-time data processing.

Real-time Data Processing

Spotify utilizes Kafka for real-time data processing and low-latency message delivery.

High Volume Event Handling

Kafka provides solutions for efficiently handling high volumes of event data for Spotify's platform.

Addressing Enterprise Challenges with Kafka

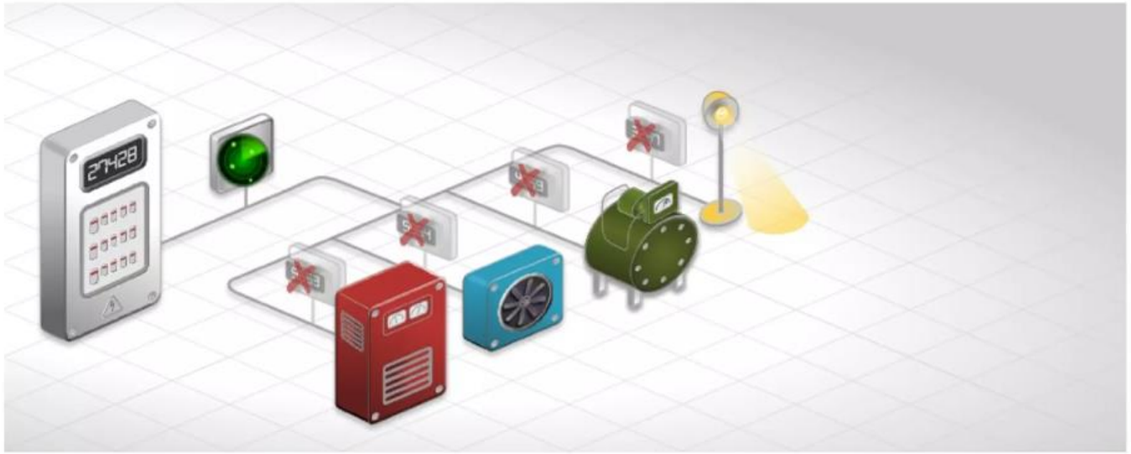
Legacy Systems Integration

Kafka facilitates seamless integration with legacy systems, addressing compatibility challenges faced by enterprises.

High Volume Event Processing

Kafka offers solutions for efficiently handling high volumes of event data, a common challenge for enterprises.

Addressing Enterprise Challenges with Kafka



Kafka facilitates smooth integration with legacy systems, addressing compatibility challenges faced by enterprises.

Kafka offers solutions for efficiently handling high volumes of event data, a common challenge for enterprises.

Addressing Enterprise Challenges with Kafka

Kafka facilitates smooth integration with legacy systems, addressing compatibility challenges faced by enterprises.

Kafka offers solutions for efficiently handling high volumes of event data, a common challenge for enterprises.

Addressing Enterprise Challenges with Kafka

Kafka facilitates smooth integration with legacy systems, addressing compatibility challenges faced by enterprises.

Kafka offers solutions for efficiently handling high volumes of event data, a common challenge for enterprises.

Addressing Enterprise Challenges with Kafka

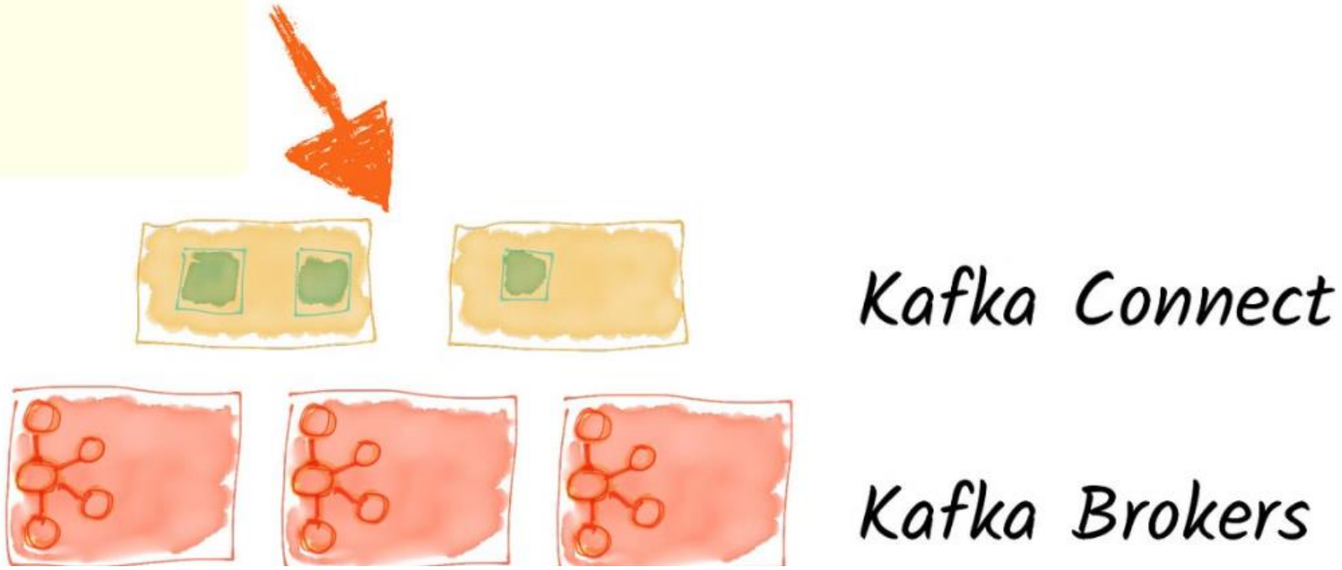
Kafka facilitates smooth integration with legacy systems, addressing compatibility challenges faced by enterprises.

Kafka offers solutions for efficiently handling high volumes of event data, a common challenge for enterprises.

Concepts de Kafka

Streaming Integration with Kafka Connect

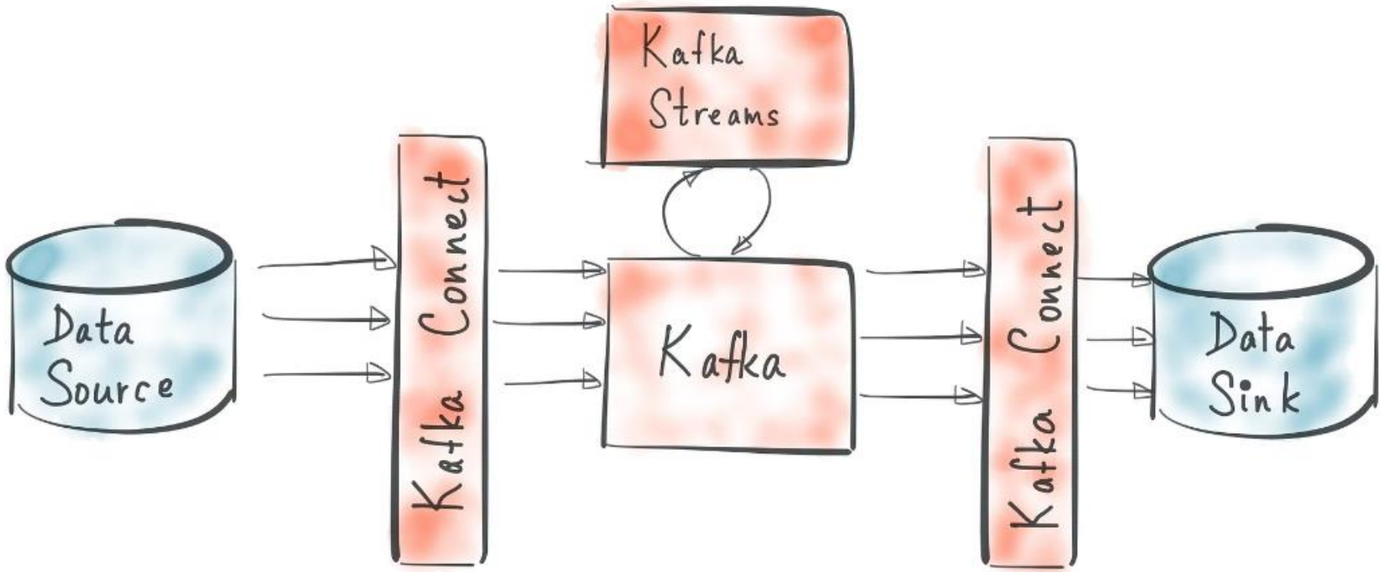
Sources



Kafka Connect

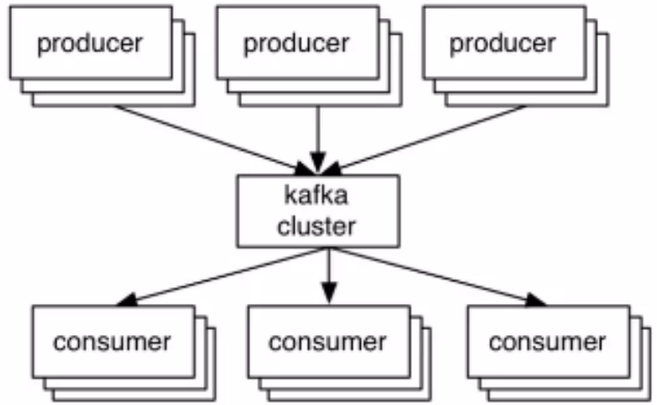
Kafka Brokers

Kafka Connect + Streams



Concepts de Kafka

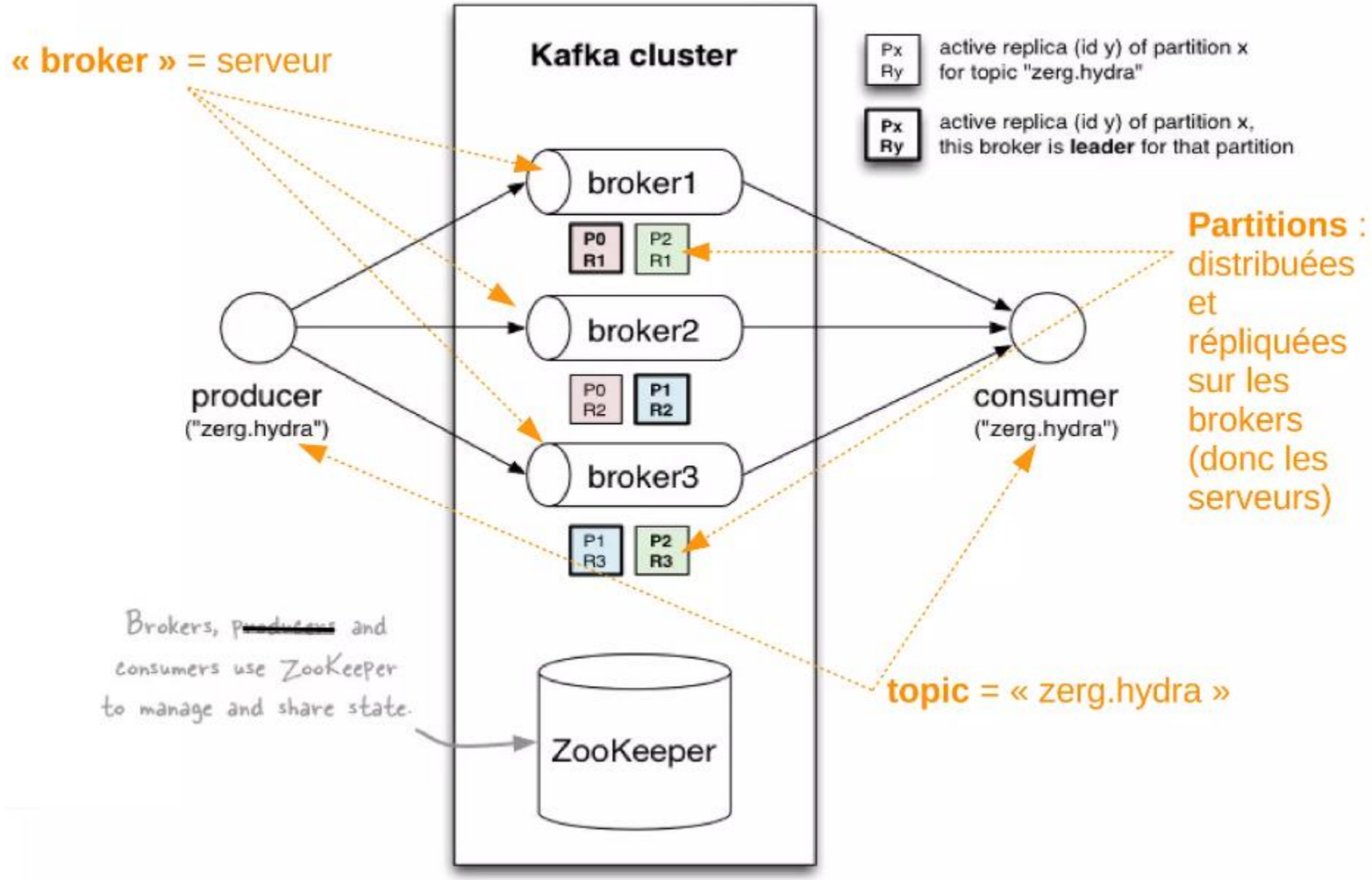
- **Overview**
  - Producers publish data to brokers.
  - Consumers subscribe and retrieve data from brokers.
  - All services are distributed.
  - Data is stored in topics.



- **Details**
  - Topics are divided into partitions and replicated to ensure high availability and scalability.
  - Kafka is not a database system and does not handle SQL queries.

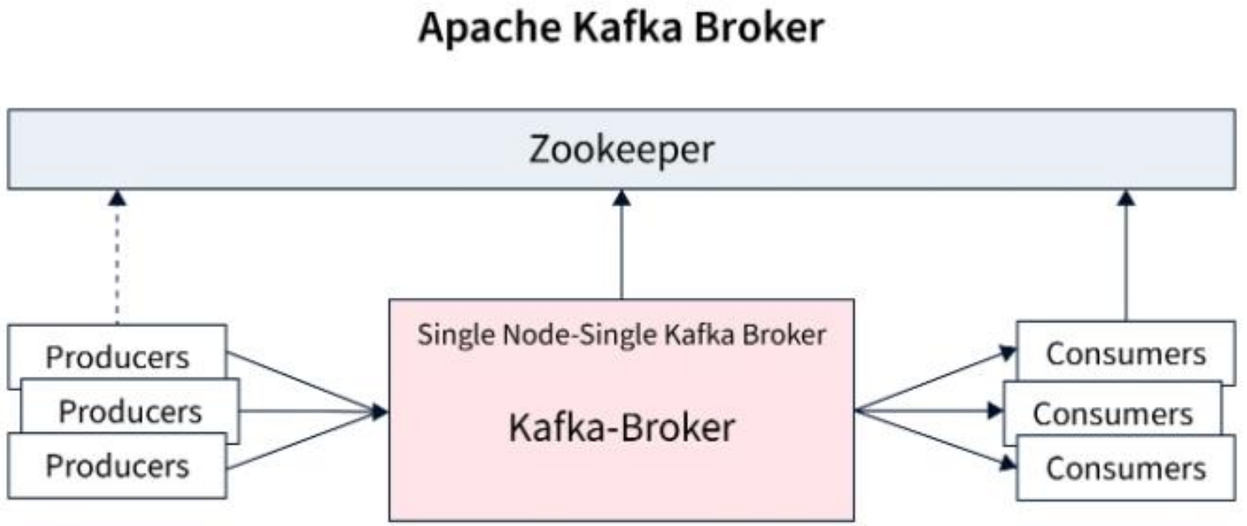
- **Advantages**
  - High availability and scalability
  - Low latency
  - High throughput
  - Asynchronous messaging
  - Real-time streaming

Concepts de Kafka - Overview



How Kafka Works

1. Producers send messages to Kafka topics.
2. Messages are stored on multiple brokers for fault tolerance.
3. Consumers read messages from brokers in order or disorder.
4. Consumers can subscribe to a single partition or a group of partitions.

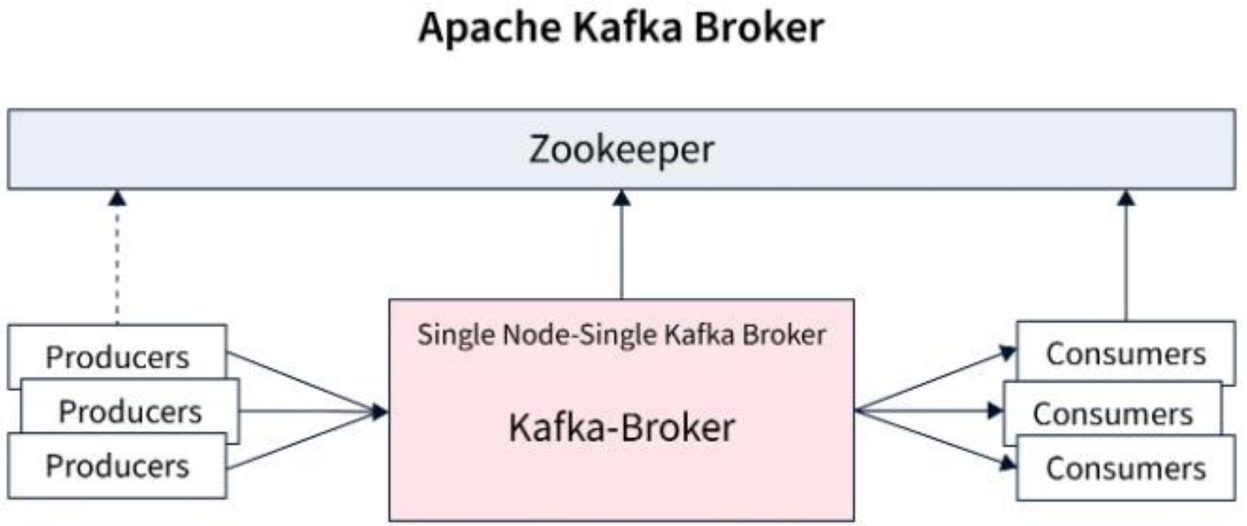


How to Start Kafka Broker?

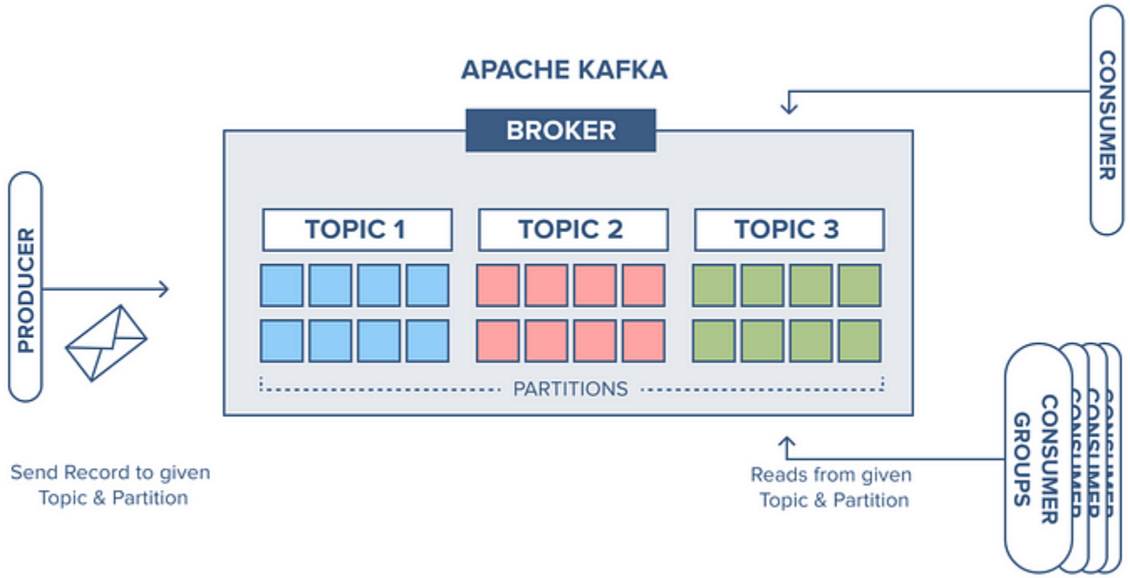
```
./bin/zookeeper-server-start.sh config/zookeeper.properties
```

```
./bin/kafka-server-start.sh config/server.properties
kafka-server-start.sh script
```

```
$ ./bin/kafka-server-start.sh
USAGE: ./bin/kafka-server-start.sh [-daemon] server.properties [--override property=value]*
```

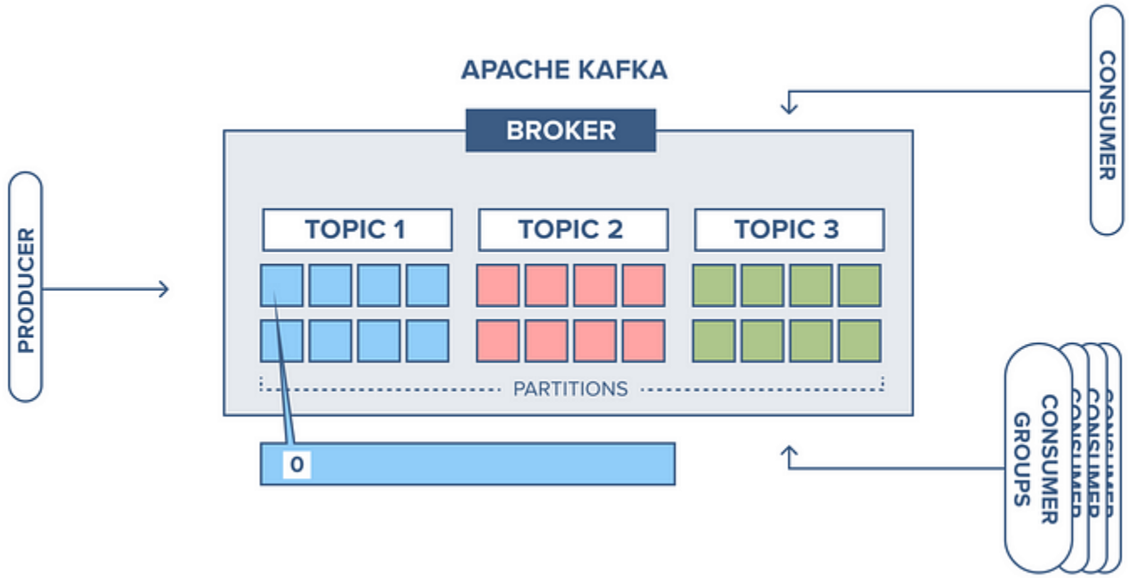


How Kafka Works



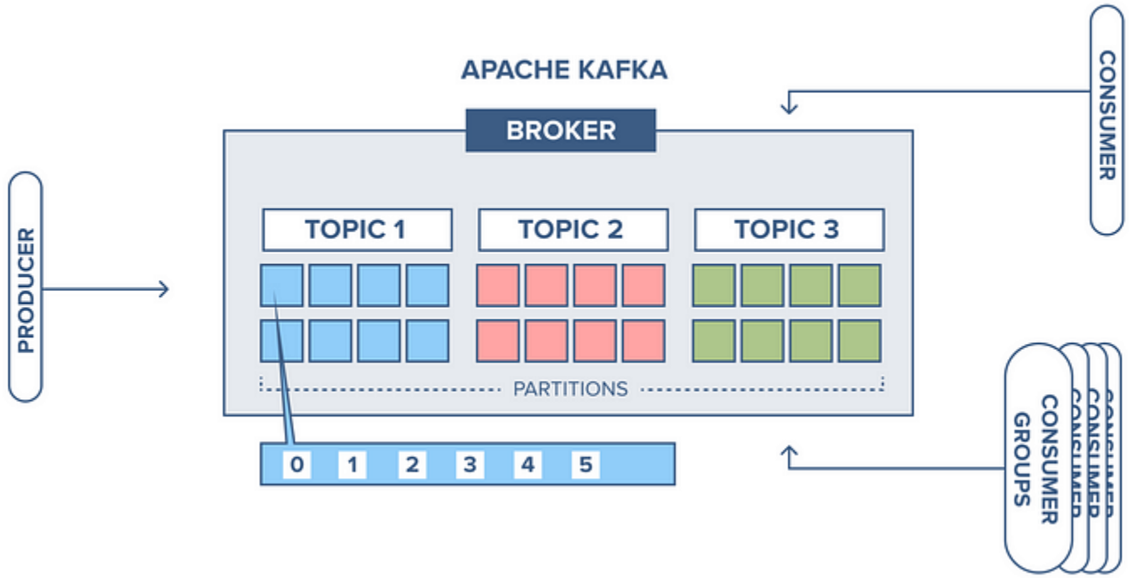
A broker with three topics, where each topic has 8 partitions.

How Kafka Works



The producer sends a record to partition 1 in topic 1 and since the partition is empty the record ends up at offset 0.

How Kafka Works



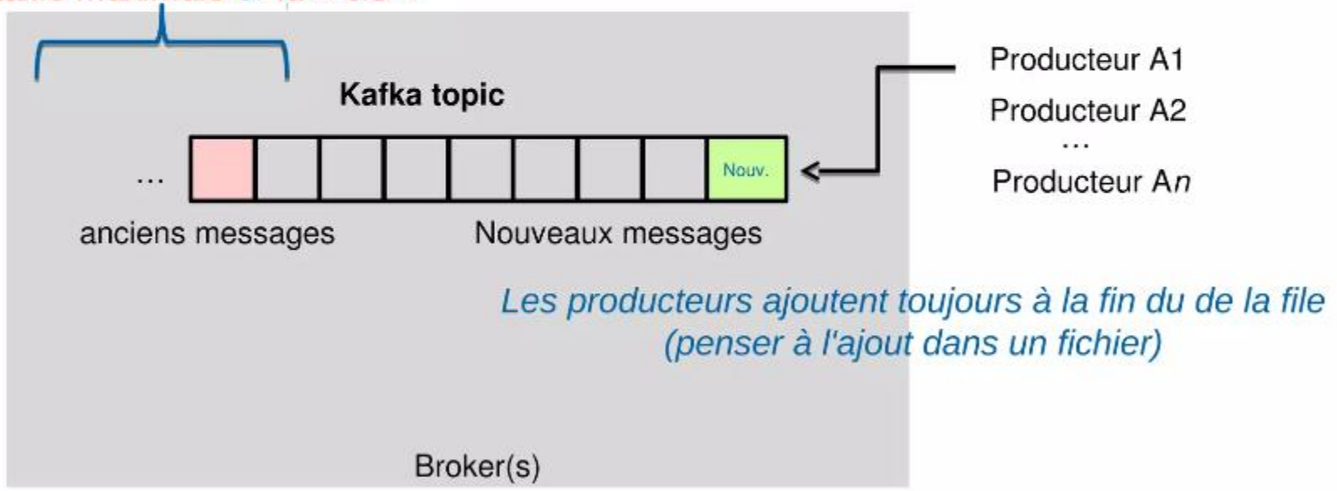
Next record is added to partition 1 will and up at offset 1, and the next record at offset 2 and so on.

Concepts de Kafka - Topic

Topic: correspond au nom du flux sur lequel les messages vont être publiés

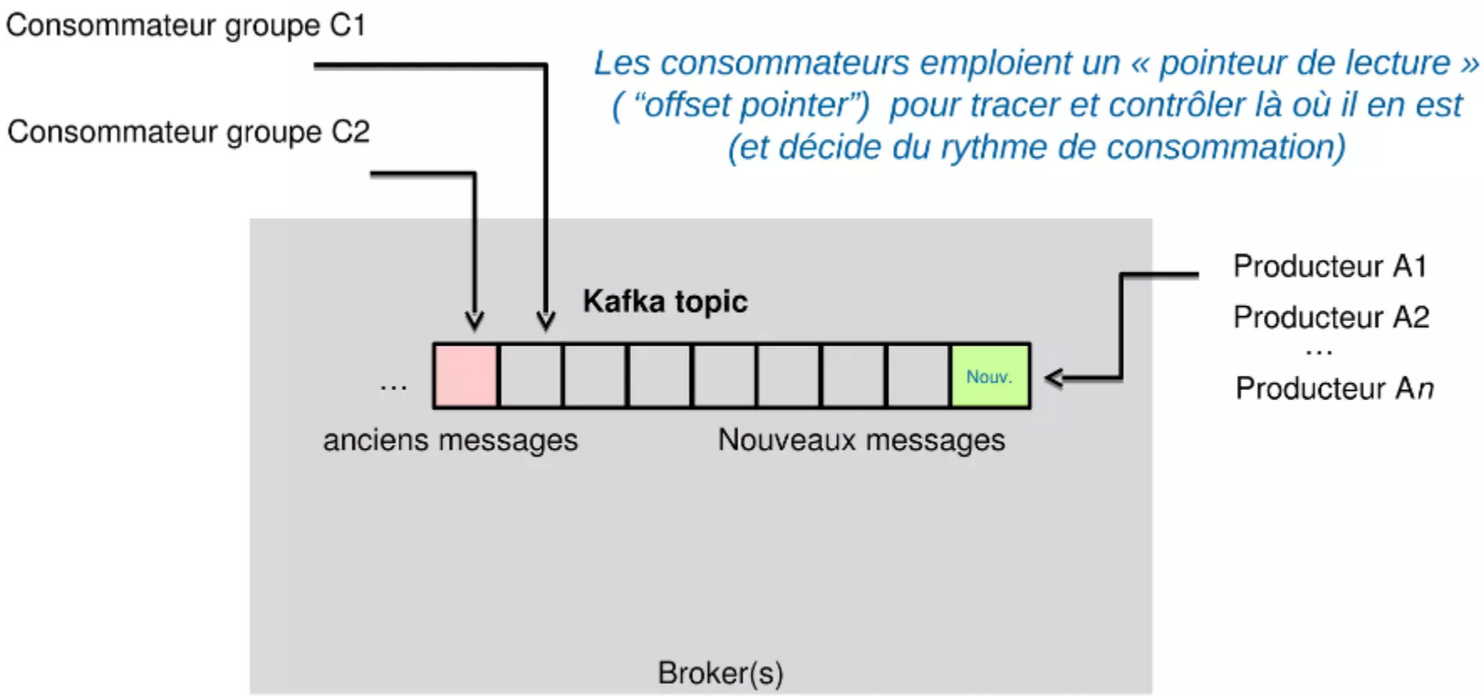
- Par exemple : “zerg.hydra”

Kafka élague depuis la “tête” en se basant sur l'âge ou la taille maximale or la « clé »



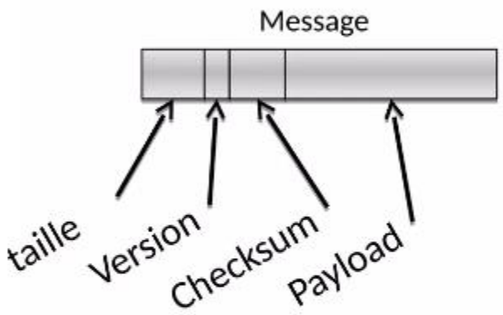
Les producteurs ajoutent toujours à la fin du de la file (penser à l'ajout dans un fichier)

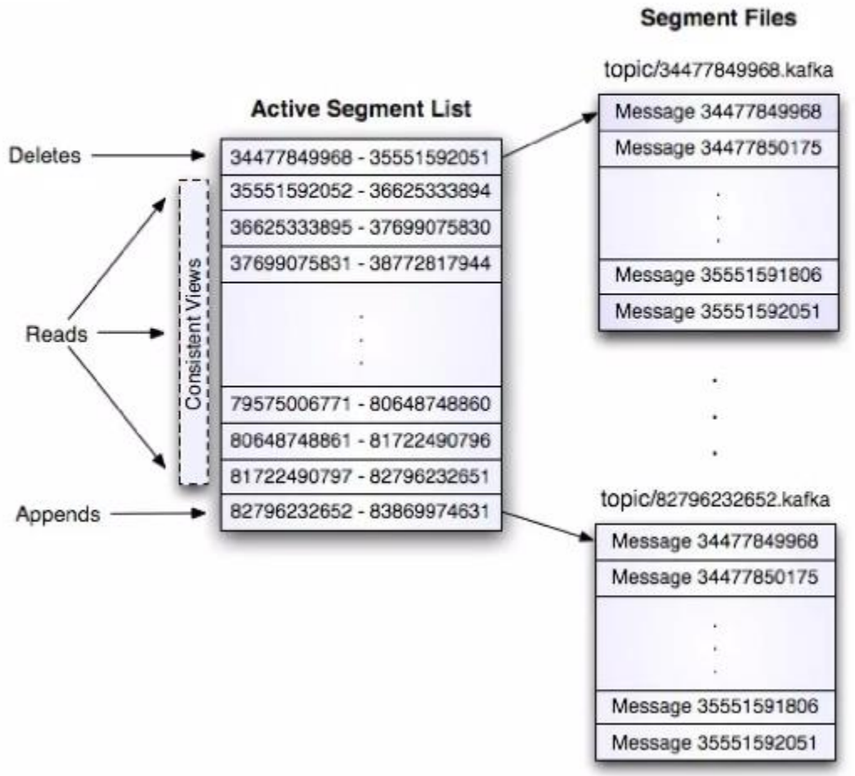
Concepts de Kafka - Topic



Concepts de Kafka - Message

Protocole léger
Traitement des messages par lot (Producteur & Consommateur)
Compression de bout en bout



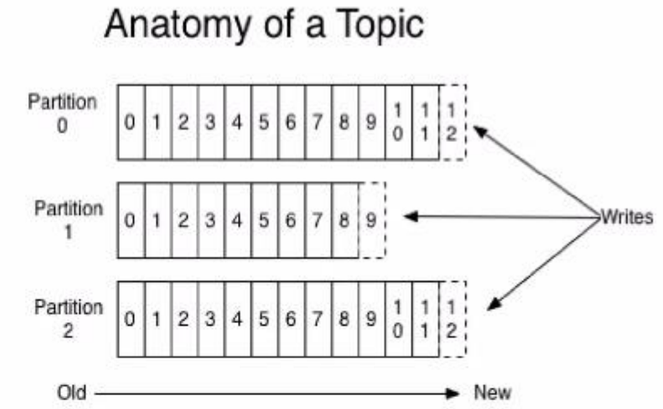


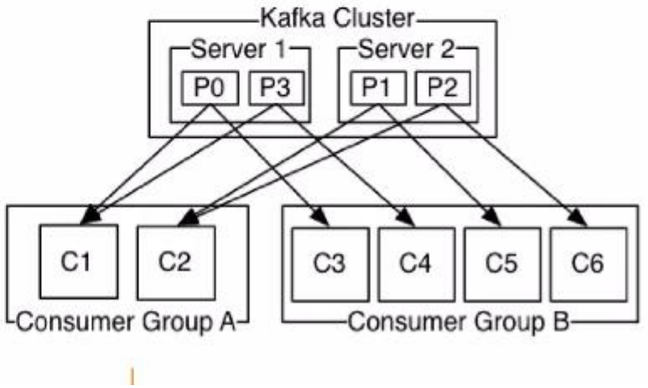
Concepts de Kafka - Partition

Les partitions

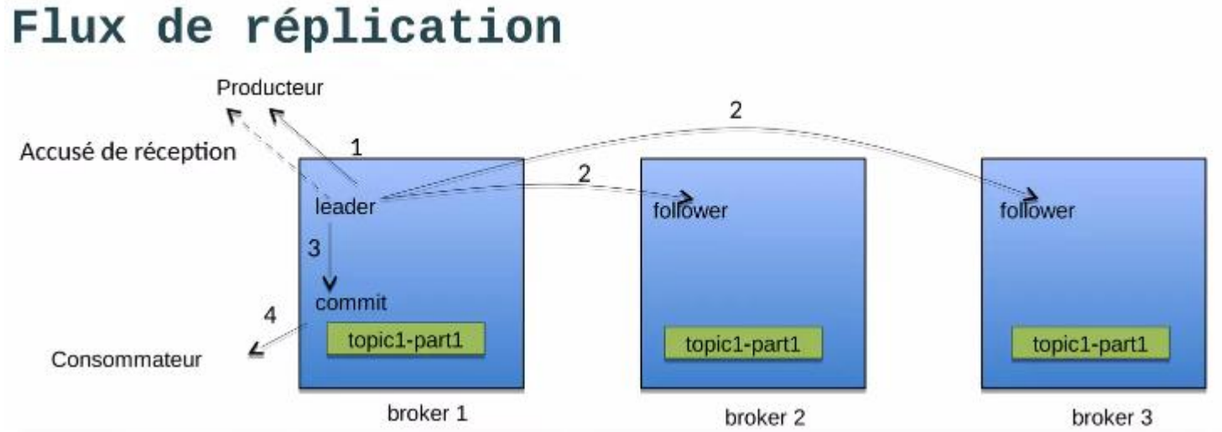
- Ordonnées
- Séquence immuable

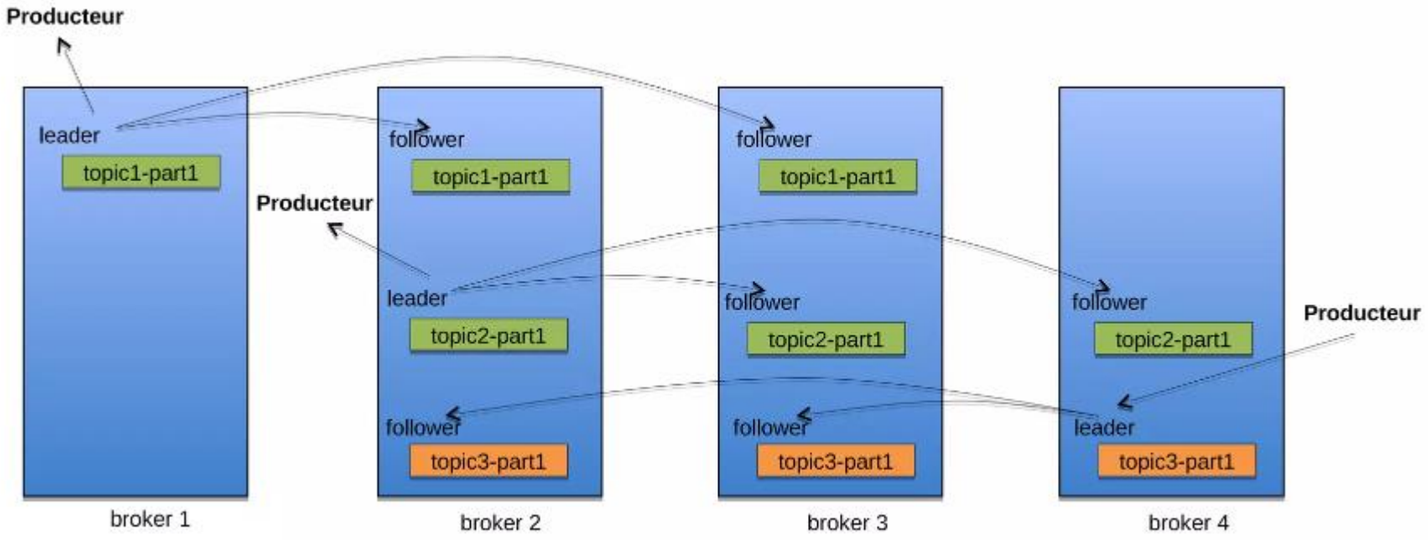
- Le nombre de partitions détermine le nombre maximum de (groupes de) consommateurs





Concepts de Kafka - Replicate





How Kafka Works

Steps to follow when setting up a connection and publishing a message/consuming a message.

1. First of all, we need to set up a secure connection. A TCP connection will be set up between the application and Apache Kafka.
2. In publisher: Publish a message to a partition on a topic.
3. In subscriber/consumer: Consume a message from a partition in a topic.

```
> import java.util.Properties
  import kafkashaded.org.apache.kafka.clients.producer._
  import org.apache.spark.sql.ForeachWriter


   class  KafkaSink(topic:String, servers:String) extends ForeachWriter[(String, String)] {
        val kafkaProperties = new Properties()
        kafkaProperties.put("bootstrap.servers", servers)
        kafkaProperties.put("key.serializer", "kafkashaded.org.apache.kafka.common.serialization.StringSerializer")
        kafkaProperties.put("value.serializer", "kafkashaded.org.apache.kafka.common.serialization.StringSerializer")
        val results = new scala.collection.mutable.HashMap[String, String]
        var producer: KafkaProducer[String, String] = _

        def open(partitionId: Long,version: Long): Boolean = {
          producer = new KafkaProducer(kafkaProperties)
          true
        }

        def process(value: (String, String)): Unit = {
            producer.send(new ProducerRecord(topic, value._1 + ":" + value._2))
        }

        def close(errorOrNull: Throwable): Unit = {
          producer.close()
        }
    }
```

```
import java.util.Properties
import kafkashaded.org.apache.kafka.clients.producer._
import org.apache.spark.sql.ForeachWriter
defined class KafkaSink
```

Enhancing Event Scheduling with Kafka

Enhancing Event Scheduling with Kafka

1

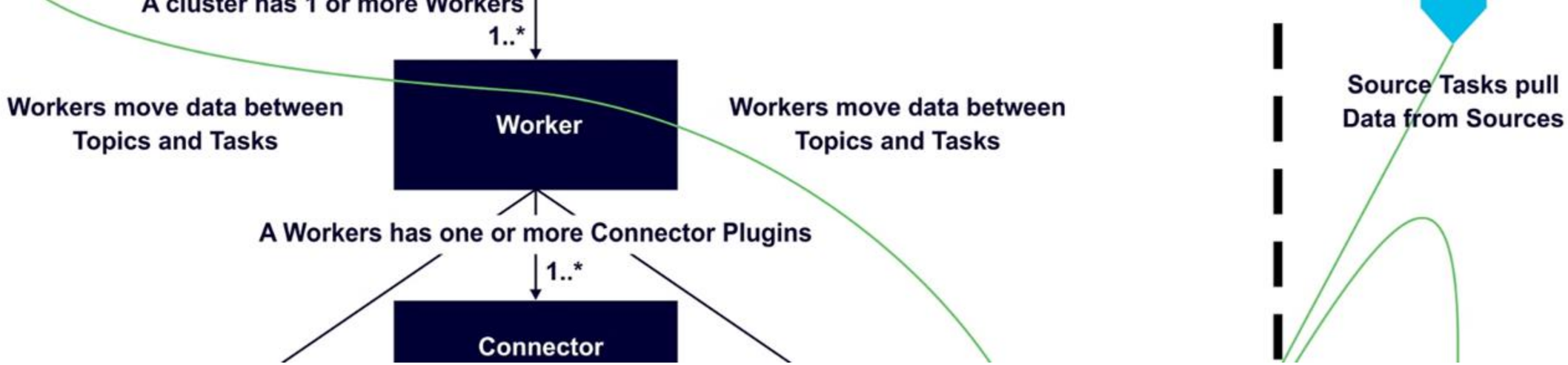
CRON for Recurring Events

Kafka enables the scheduling of recurring events using CRON, streamlining automated processes.

2

Message Scheduling

Through its robust features, Kafka supports the efficient scheduling and management of messages, optimizing event handling.



Utilizing Kafka in Multi-Server Environments

1

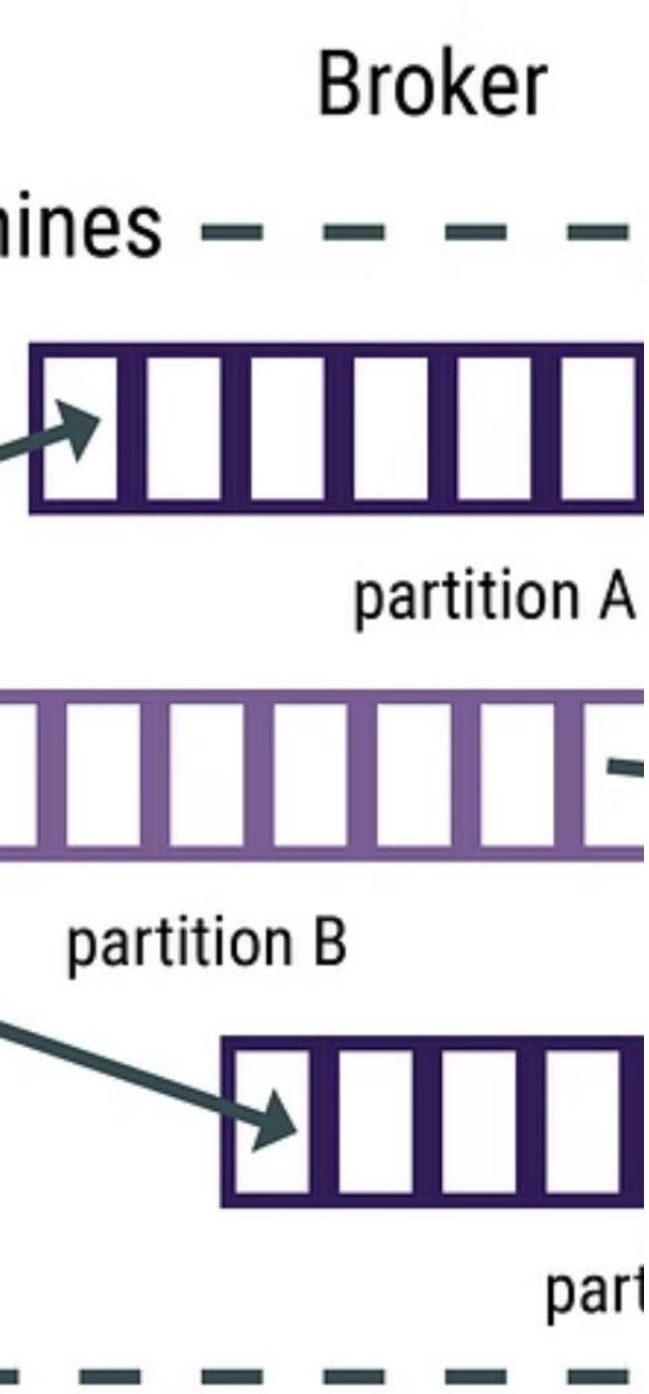
Cluster Scalability

Kafka's infrastructure allows for effective scaling across multiple servers, ensuring efficient event management.

2

Seamless Data Replication

With Kafka, seamless data replication and synchronization are achievable, enhancing data consistency across servers.



Understanding Kafka's Message Specificities

1

Unique Message Attributes

Kafka's message specificity ensures the preservation of unique attributes, enhancing event data integrity.

2

Consumer Message Handling

Efficient handling and processing of specific message types are facilitated by Kafka's consumer-centric design.

The Role of Kafka in Modern Event Management

Streamlining Events

Kafka plays a crucial role in streamlining event management processes, optimizing data flow and processing.

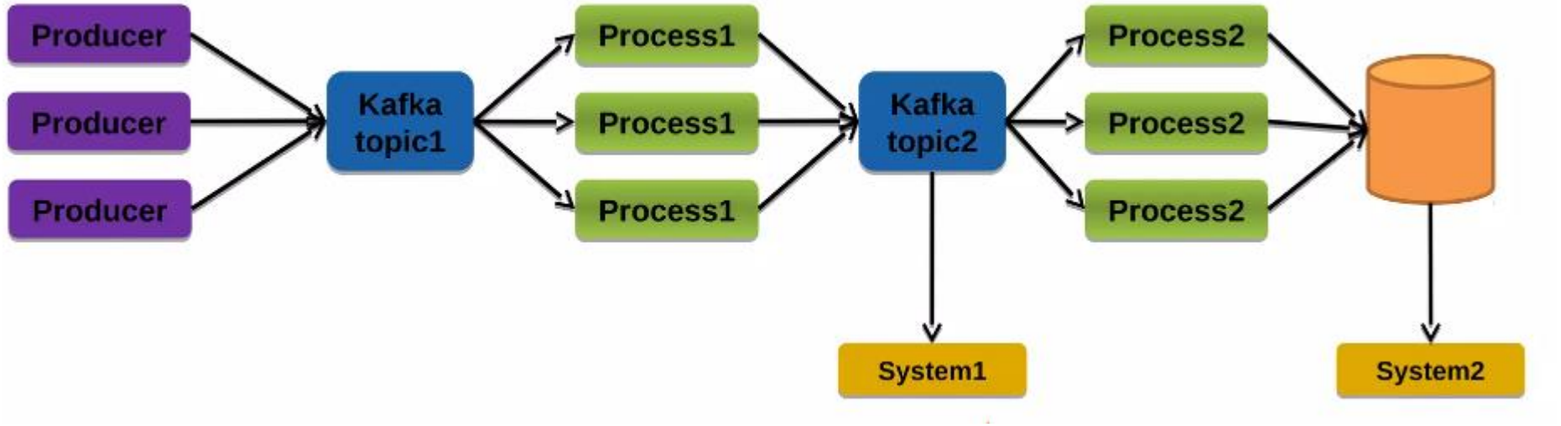
Real-time Insights

Through real-time data processing, Kafka empowers timely decision-making and actions based on the most current information.

Scalability

Kafka's scalable architecture addresses the evolving needs of event management, allowing seamless expansion and optimization.

Ecosystem


Producer
Producer
Producer
Kafka topic1
Process1
Process1
Process1
Kafka topic2
Process2
Process2
Process2
System1
System2

Limitations of Using Direct Kafka API Calls

Performance Bottlenecks

Can introduce performance bottlenecks, especially with large volumes of data or frequent data transfers. This can impact the overall system performance and responsiveness.

Complexity and Maintenance

Managing direct API integrations can be complex, requiring ongoing maintenance and updates to accommodate changes in the API endpoints or data formats.

Scalability Challenges

Might face scalability challenges as the data volume grows, potentially leading to increased latency and resource utilization.


Apache Kafka adoption spans companies across industries.
intuit
NETFLIX
ebay
Pinterest
Goldman Sachs
airbnb
cisco
box
Square
Adobe
salesforce
yelp
dish NETWORK
Cerner
WIKIPEDIA
The Free Encyclopedia
verizon
confluent
Spotify
Square
salesforce
fitbit
GoPro
slack
airbnb
coursera
intuit
PayPal
Dropbox
yahoo!
Uber
WIKIPEDIA
The Free Encyclopedia