Apache Kafka: A Overview

📘Introduction & Architecture

What is Apache Kafka?

Apache Kafka is a distributed event streaming platform used for building real-time data pipelines and streaming applications. Originally developed by LinkedIn and later open-sourced through the Apache Software Foundation, Kafka is designed to handle high-throughput, fault-tolerant, and scalable messaging.

Key Concepts

* **Producer**: Sends data (messages) to Kafka topics.
* **Consumer**: Reads data from Kafka topics.
* **Topic**: A category or feed name to which records are sent.
* **Broker**: A Kafka server that stores data and serves clients.
* **Partition**: Topics are split into partitions for parallelism and scalability.
* **Offset**: A unique ID for each message in a partition.

Kafka Architecture

Kafka follows a publish-subscribe model and consists of:

* **Kafka Cluster**: A group of brokers working together.
* **ZooKeeper**: Manages cluster metadata and broker coordination (being phased out in favor of KRaft).
* **Producers and Consumers**: Communicate with brokers to send and receive data.

Data Flow

1. Producers publish messages to a topic.
2. Kafka stores messages in partitions.
3. Consumers subscribe to topics and read messages based on offsets.

📗 2: Features, Use Cases & Ecosystem

🔧 Core Features

* **High Throughput**: Handles millions of messages per second.
* **Scalability**: Easily scales horizontally by adding brokers.
* **Durability**: Messages are persisted to disk and replicated.
* **Fault Tolerance**: Replication ensures data availability.
* **Real-Time Processing**: Enables low-latency data streaming.

🌍 Common Use Cases

* **Log Aggregation**: Centralizing logs from multiple services.
* **Event Sourcing**: Capturing state changes as a stream of events.
* **Stream Processing**: Real-time analytics and transformations.
* **Messaging System**: Decoupling microservices.
* **Data Integration**: Connecting databases, applications, and analytics tools.

🧰 Kafka Ecosystem

* **Kafka Streams**: A Java library for building stream processing apps.
* **Kafka Connect**: Integrates Kafka with external systems (e.g., databases, cloud services).
* **KSQL (ksqlDB)**: SQL-like queries for stream processing.
* **MirrorMaker**: Replicates Kafka clusters across data centers.

🚀 Why Kafka?

Kafka is ideal for modern data architectures due to its:

* Resilience
* Performance
* Flexibility
* Integration capabilities