**What is Kafka?**

* Kafka is an open-source distributed event streaming platform that is used to handle large amounts of data in real-time. It was initially developed by LinkedIn and is now maintained as an Apache Software Foundation project.
* At its core, Kafka is a messaging system that allows for the collection, processing, and storage of data streams in a distributed manner. It can be used to handle various types of data, such as log data, metrics, user activity, and application events.
* Kafka provides a highly scalable and fault-tolerant system for processing large volumes of data in real-time. Its distributed architecture allows for the parallel processing of data across multiple servers, enabling high throughput and low-latency data processing. Additionally, Kafka is designed to handle data streams from multiple sources, making it an ideal choice for building real-time data pipelines.
* Kafka can be used in a variety of use cases, such as data integration, stream processing, real-time analytics, and messaging systems. It has become a popular choice for building large-scale data processing systems in various industries, including finance, e-commerce, and social media.

**Messaging System:**

* A messaging system is a software system that facilitates the exchange of messages between different applications, services, or components. In a messaging system, messages are typically sent asynchronously, which means that the sender and receiver do not need to be actively communicating at the same time.
* Messaging systems provide a way for applications to communicate with each other without being tightly coupled, which can make them more flexible and easier to maintain. They can also help to decouple the sending and receiving of messages, which can improve scalability and reliability.

There are two ways to store records:

1. Centralized
2. Distributed

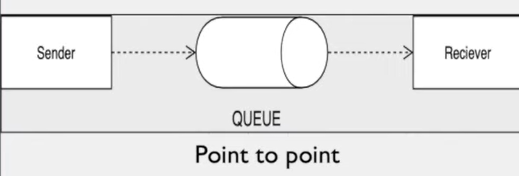
**Centralized deployment:**

* In a centralized deployment, all Kafka brokers in the cluster are deployed on a single machine or a small set of machines. This configuration is generally used for smaller clusters that do not require a high level of fault tolerance or scalability. In this deployment, all brokers are managed centrally, and there is a single point of failure.

**Distributed deployment:**

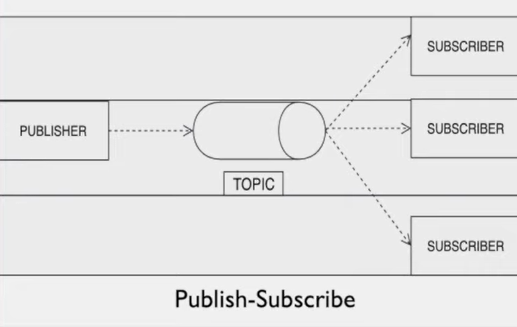
* In a distributed deployment, Kafka brokers are deployed across multiple machines, typically in a cluster or data center. This configuration is used for larger clusters that require higher levels of fault tolerance, scalability, and performance. In a distributed deployment, each Kafka broker is managed independently, and there is no single point of failure.
* In a distributed deployment, Kafka brokers are typically organized into groups, or "replication groups," where each group contains one or more replicas of a given partition. This allows for data redundancy and high availability in case of broker failures. Additionally, in a distributed deployment, Kafka topics and partitions can be automatically partitioned and distributed across multiple brokers, allowing for parallel processing of data across the cluster.
* Overall, a distributed deployment provides greater scalability, fault tolerance, and performance than a centralized deployment. However, it also requires more resources and management overhead, as the cluster must be carefully designed and configured to ensure proper operation.
* There are two primary types of messaging systems:

1. **Point-to-Point (P2P) Messaging:**



In a P2P messaging system, messages are sent from a single sender to a single receiver. The sender sends a message to a specific destination, which is typically referred to as a queue. The receiver then retrieves messages from the queue when they are ready to process them.

1. **Publish-Subscribe (Pub/Sub) Messaging:**



In a Pub/Sub messaging system, messages are sent from a single sender to multiple receivers. The sender publishes messages to a specific topic, and any interested receivers can subscribe to that topic to receive messages.

* Messaging systems can be used for a variety of purposes, including integrating applications, sharing data between systems, and building distributed systems. Some popular messaging systems include Apache Kafka, RabbitMQ, and ActiveMQ.