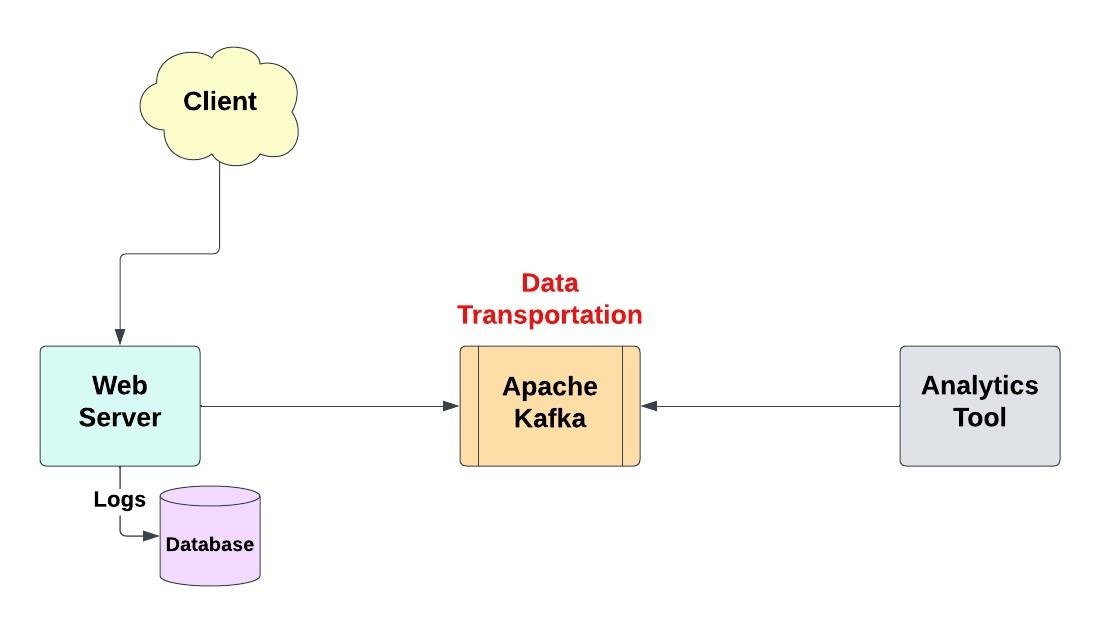
* **Apache Kafka** is a distributed streaming platform that can be used to collect, store, and process large amounts of data. It is often used for real- time streaming applications, such as website analytics and fraud detection.
* Apache Kafka is used for data transport. It can handle a high volume of data and enables you to pass messages from one end-point to another.
* It is designed to handle data streams from multiple sources and deliver them to multiple consumers.

# Let's understand how Apache Kafka works with a weblog example:

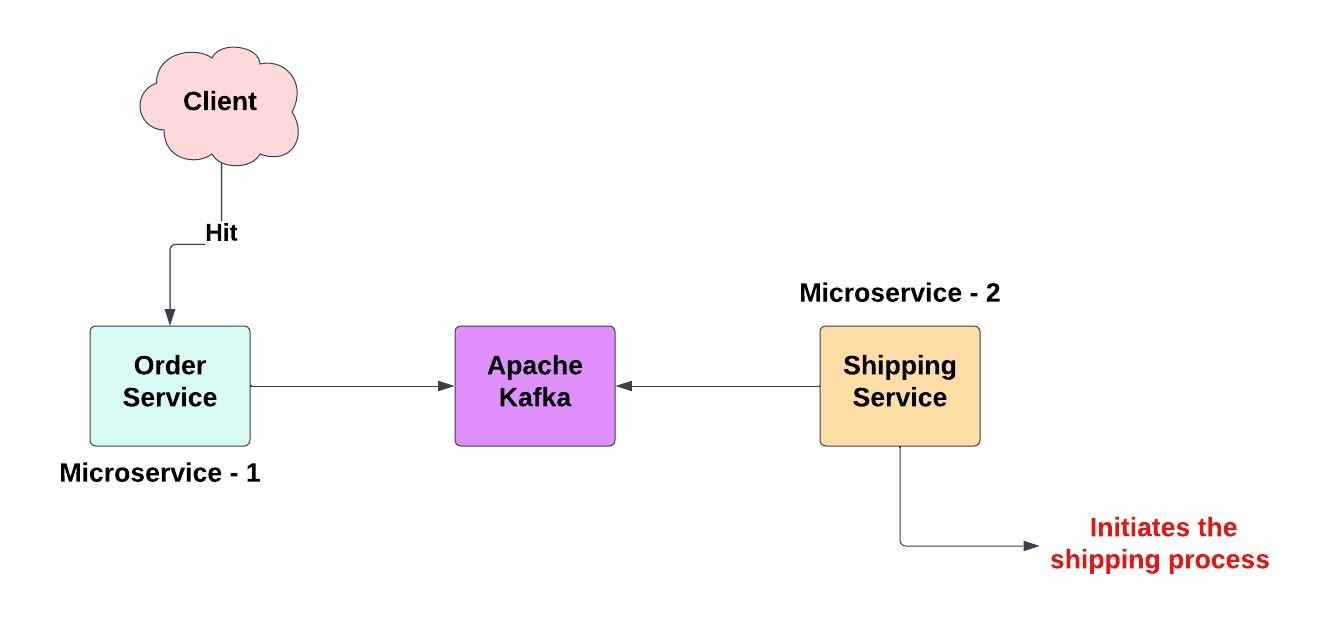
A website that receives a lot of traffic, and you want to analyze the web logs generated by the visitors. Web logs contain information like the time of each request, the pages visited, and the user's IP address.



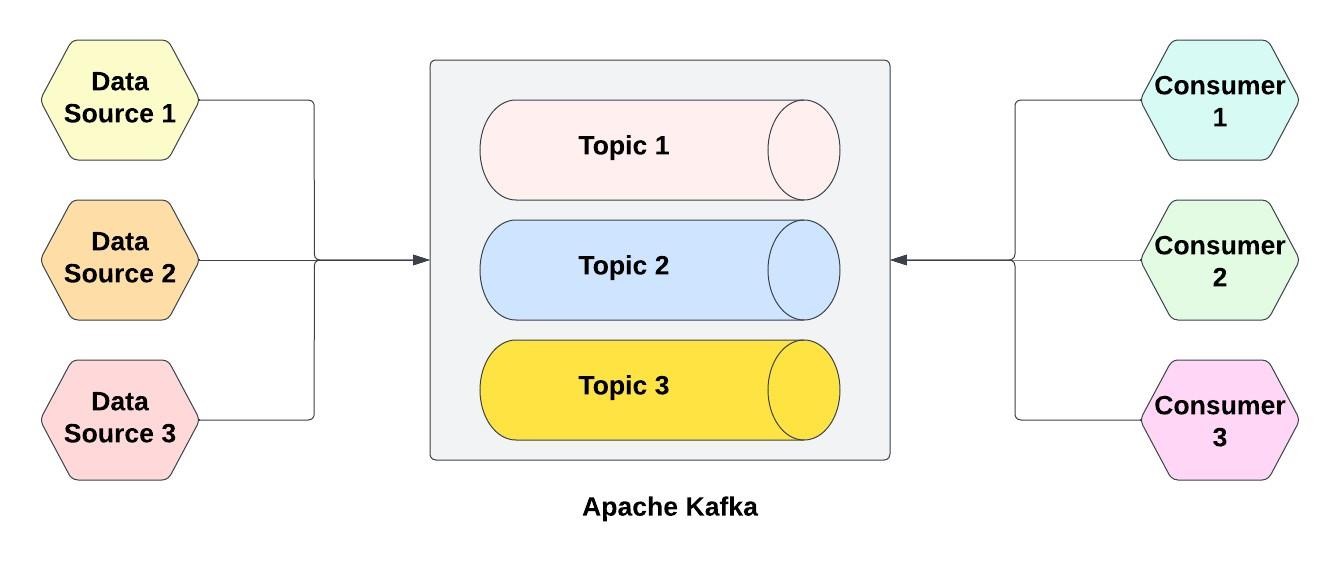
Now, let's see how Apache Kafka can help with this scenario:

* 1. **Collecting Web Logs**: Apache Kafka acts as a central hub for collecting weblogs. Whenever a visitor makes a request to your website, instead of directly processing and analyzing the log, you send it to Kafka.
  2. **Kafka Topics:** In Kafka, you organize logs into "topics." Think of a topic as a category or channel where logs with similar characteristics are grouped together. For our example, we can create a topic called "weblogs."
  3. **Producers:** To send weblogs to Kafka, you use a producer. A producer is a piece of code that takes the web log data and publishes it to the Kafka topic. Each log entry is sent as a message to the "web-logs" topic.
  4. **Consumers:** On the other end, you have consumers that read the weblog messages from Kafka. Consumers can be applications or processes that subscribe to the "web-logs" topic and pull log messages from it.
  5. **Processing Web Logs:** Once the consumers receive the log messages, they can perform various operations on the data. For example, you can analyze the logs to identify trends, track user behavior, or detect anomalies. You can write code that processes the logs and performs these tasks.
  6. **Storing or Forwarding Processed Logs:** After processing the logs, you can store the results in a database for future analysis or send them to another system for real-time monitoring. This allows you to gain insights from the weblogs and take appropriate actions based on the analysis.
* Kafka stores data for a specified period of time, but it does not provide a persistent storage retention period by default. The **default retention period** is 7 days. You can configure Kafka to retain data for a longer period of time.
* Apache Kafka can be used in **microservices architectures** to enable asynchronous communication and data integration between services.

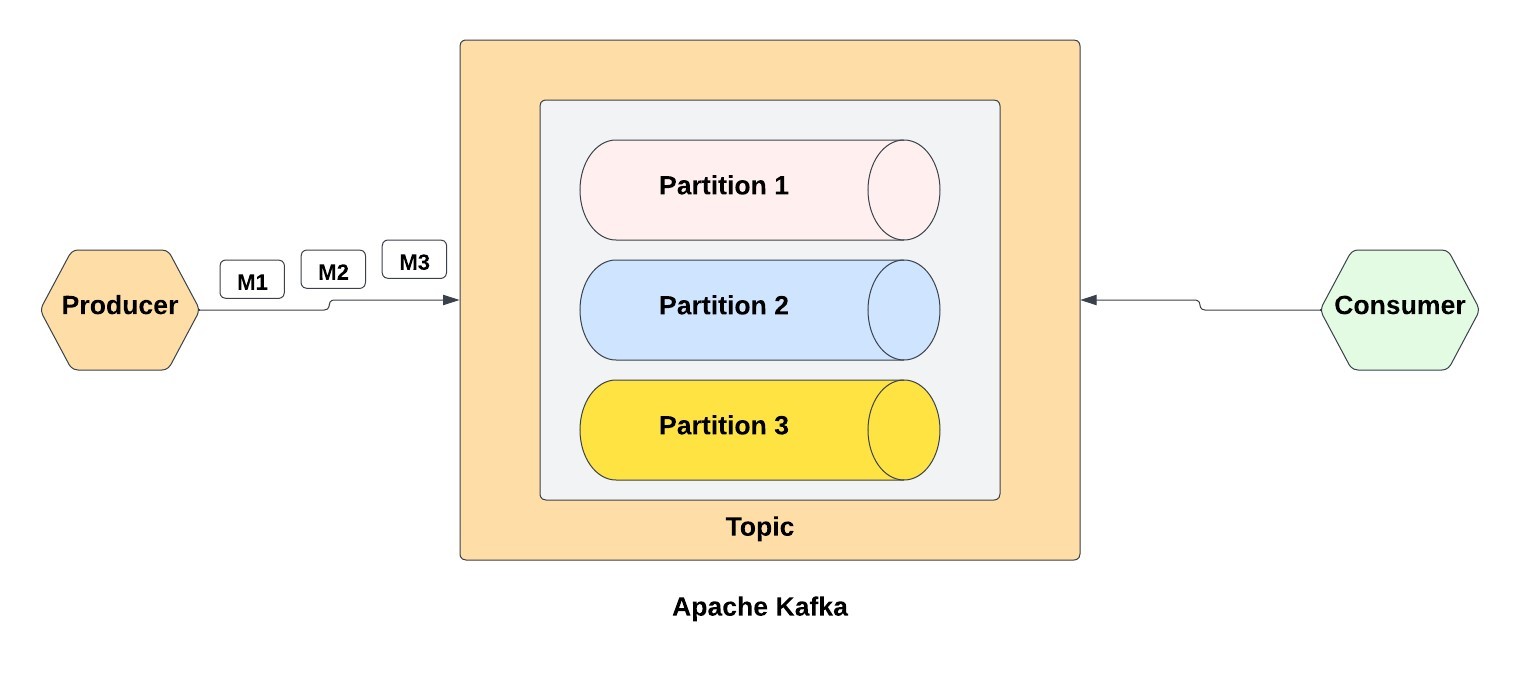
Here's an example of how Kafka can be utilized in a microservices environment:

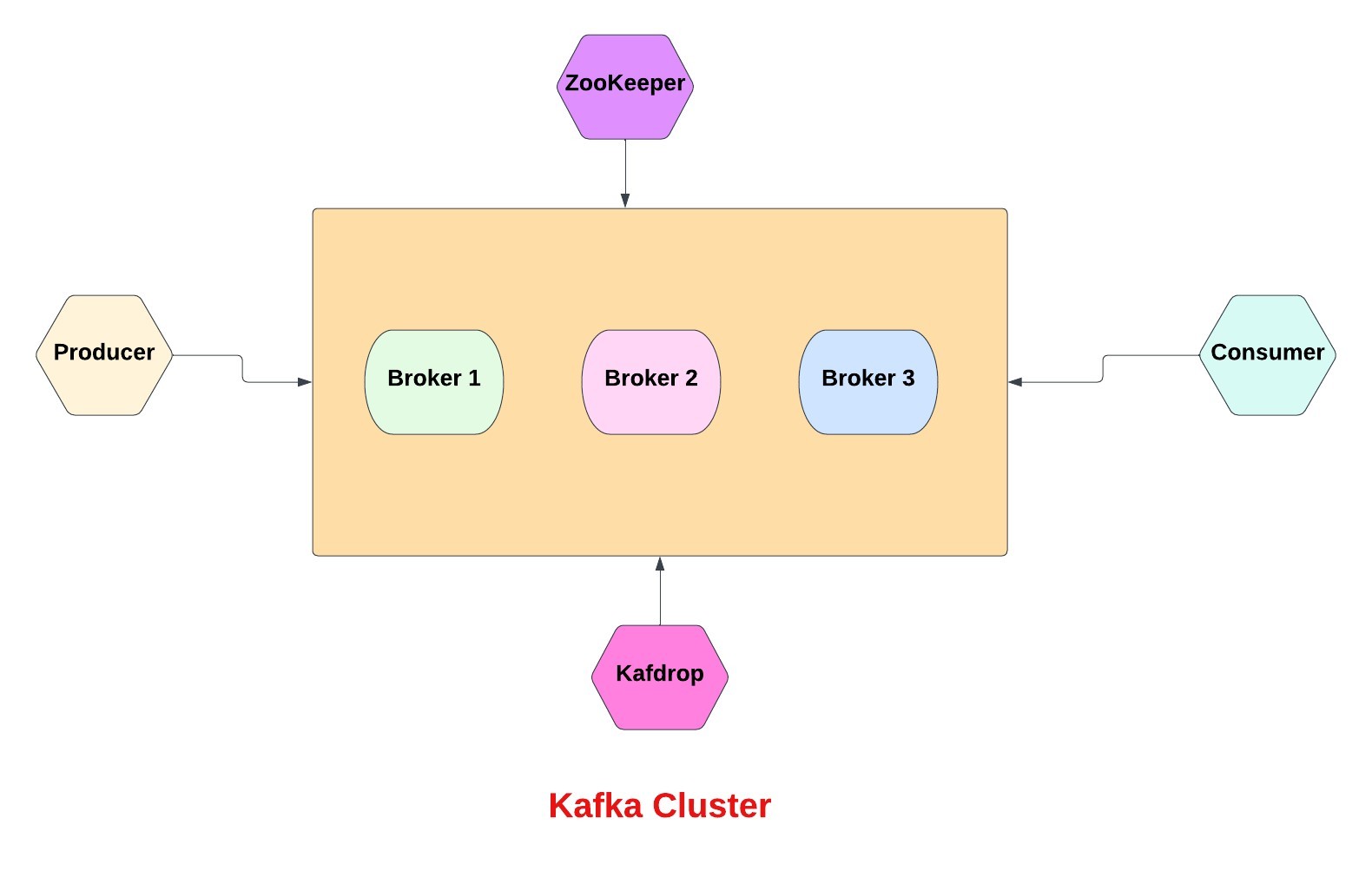


* When a user places an order, the Order Service publishes an order event to a Kafka topic, such as "order-events." The event contains details about the order, such as the products, quantity, and customer information.
* The Shipping Service consumes the order event from the "order- events" topic. It processes the event and initiates the shipping process for the ordered products.
* By utilizing Kafka in this microservices architecture, we achieve loose coupling and enable asynchronous communication between the services.
* Apache Kafka can be used effectively for **ETL** (Extract, Transform, Load) operations in data processing pipelines. ETL is a common data integration process that involves extracting data from various sources, transforming it into a desired format, and loading it into a target destination.
* In Apache Kafka, a **topic** is a category or stream of records or messages. It represents a particular type of data that you want to publish or consumes. Messages are published to topics by producers and consumed by consumers.



* Topics are **partitioned** to distribute the load of producing and consuming messages across multiple brokers. Each Kafka topic is divided into one or more partitions, and each partition is an ordered and immutable sequence of records.



* In Apache Kafka, each partition within a topic is identified by a unique integer called the "**offset**." The offset represents the position of a record within a partition and serves as a unique identifier for that record.
* A **Kafka broker** is a server that runs the Kafka software and stores data for one or more Kafka topics.
* A **Kafka cluster** is a group of one or more Kafka brokers that are connected together.
* 
* Kafka cluster also provides a number of features:
* **Durability**: Kafka stores data in a durable way, so it is not lost if there is a problem with one of the servers in the cluster.
* **High throughput:** Kafka can process data at high throughput, which is important for real-time applications.
* **Low latency**: Kafka can deliver data with low latency, which is important for applications that need to respond to events in real time.
* **Disaster recovery**: Kafka clusters can be used to store data for disaster recovery purposes. This ensures that you can recover your data in the event of a hardware failure or other disaster.
* **Replication**: Kafka replicates data across multiple brokers, so you can be sure that your data is always available, even if one broker fails.
* In Apache Kafka, the **leader of a partition** within a Kafka cluster is responsible for handling all read and write operations for that partition. Each partition in Kafka has one leader and multiple follower replicas.
* Kafka uses a **replication mechanism** to provide fault tolerance and data redundancy. The leader replica receives all write requests and appends them to the partition's log. It then replicates these messages to the follower replicas, ensuring that the replicas have an identical copy of the data.
* **ZooKeeper** is a distributed coordination service that is used to manage Kafka clusters. It provides a way for Kafka brokers to communicate with each other and to keep track of the state of the cluster.
* ZooKeeper is a critical component of Kafka clusters. It is used for the following:
* **Leader election:** ZooKeeper is used to elect the leader for each partition.
* **Replication**: ZooKeeper is used to keep the data in Kafka topics in sync.
* **Topic creation**: ZooKeeper is used to create new topics.
* **Topic deletion**: ZooKeeper is used to delete topics.
* **Broker registration:** When a new broker joins the cluster, it registers itself with ZooKeeper.
* **Broker deregistration:** When a broker leaves the cluster, it deregisters itself with Zookeeper