

Need of Messaging Systems

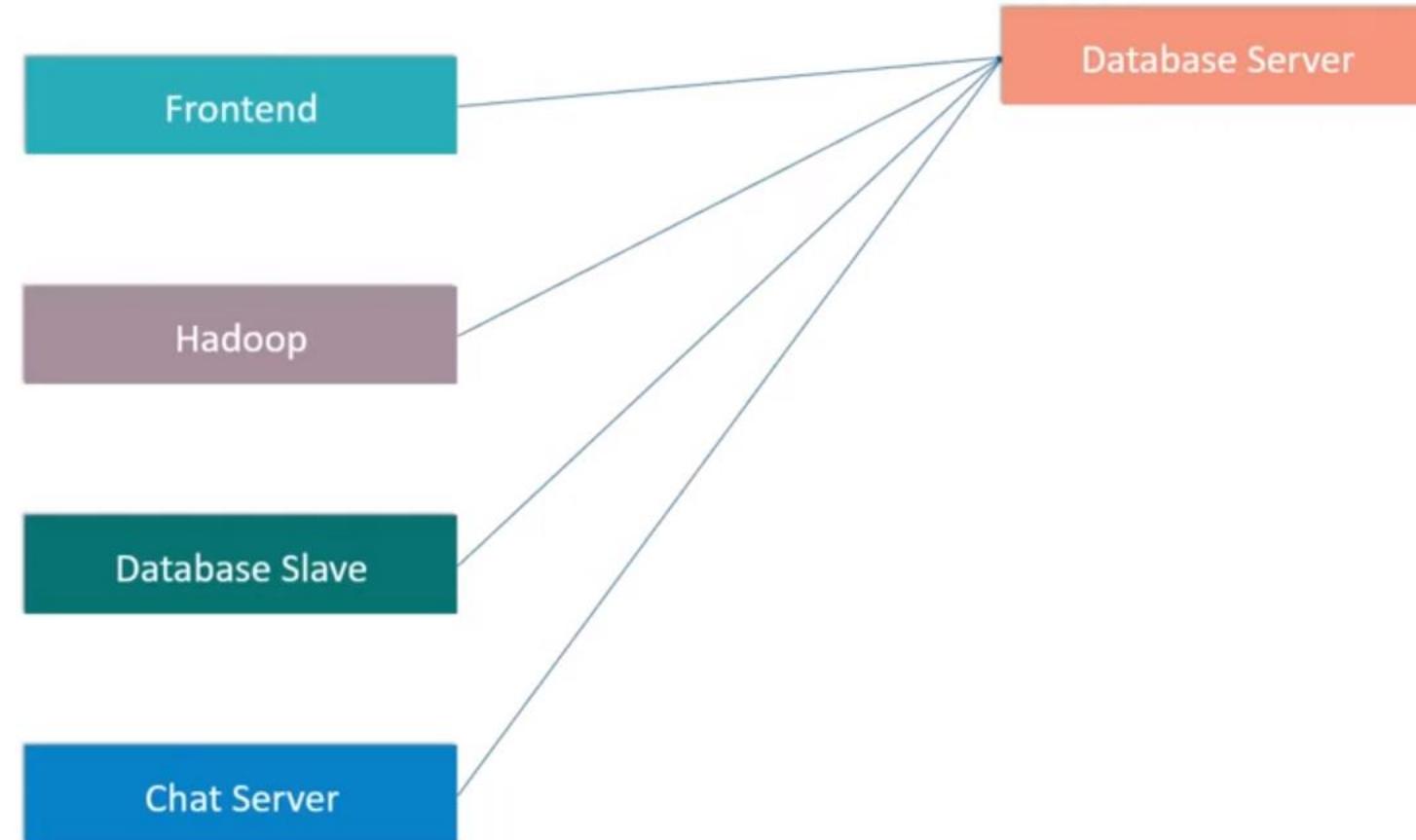
Data Pipelines

Communication is required between different systems in the real-time scenario, which is done by using data pipelines.



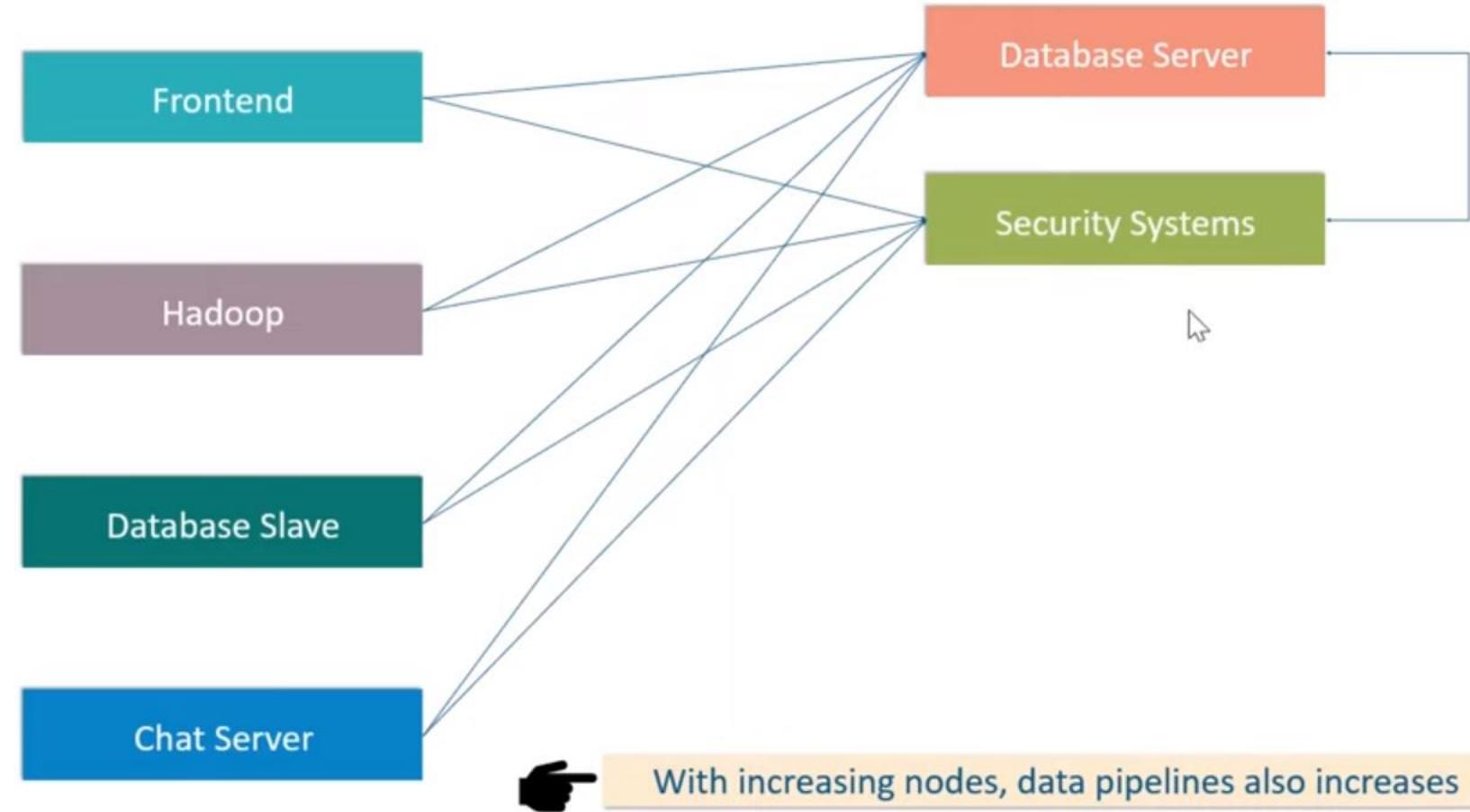
For Example: Chat Server needs to communicate with Database Server for storing messages

Increase in number of Nodes



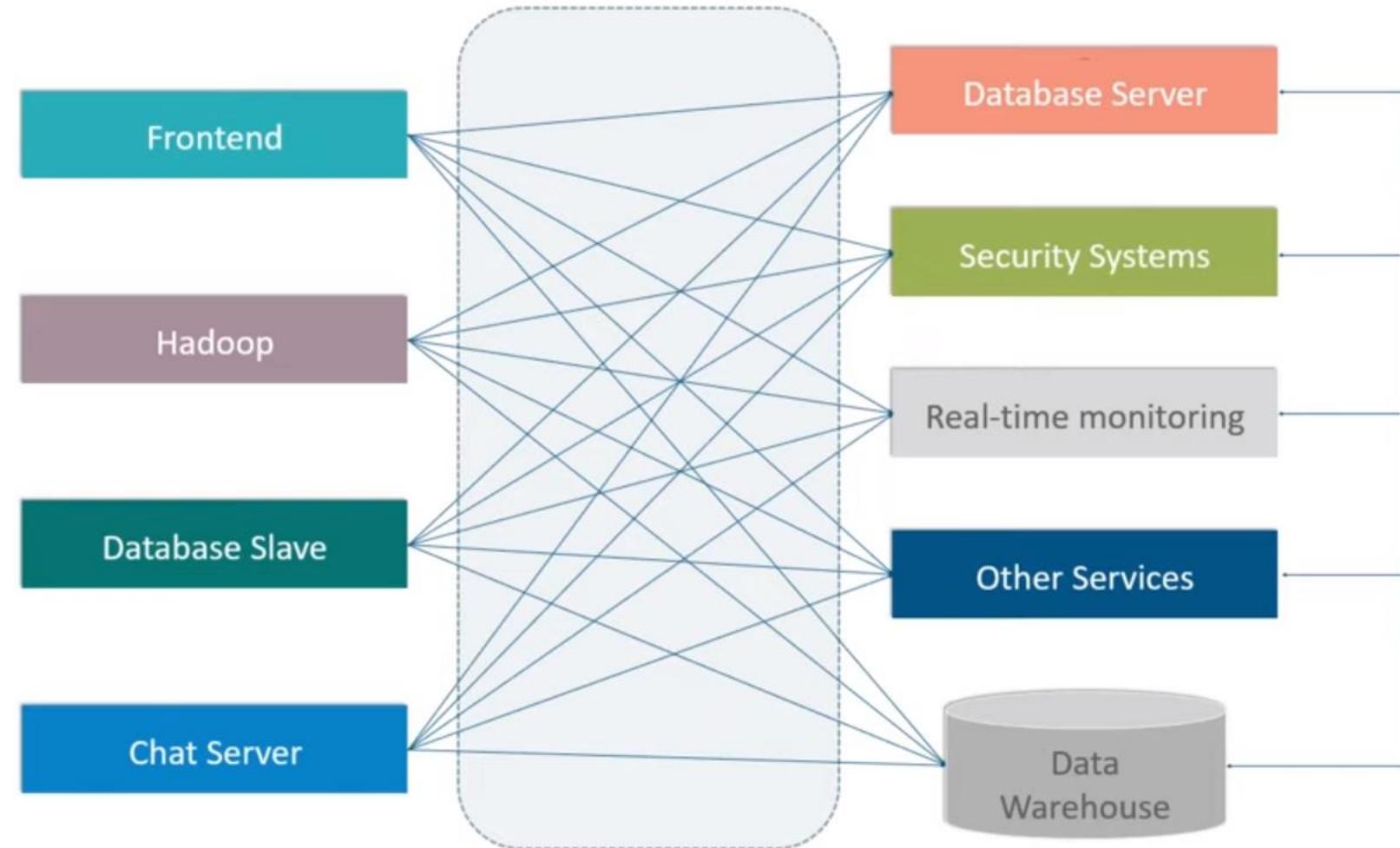
Similarly, there may be many applications wanting to access the Database Server

Increase in number of Nodes



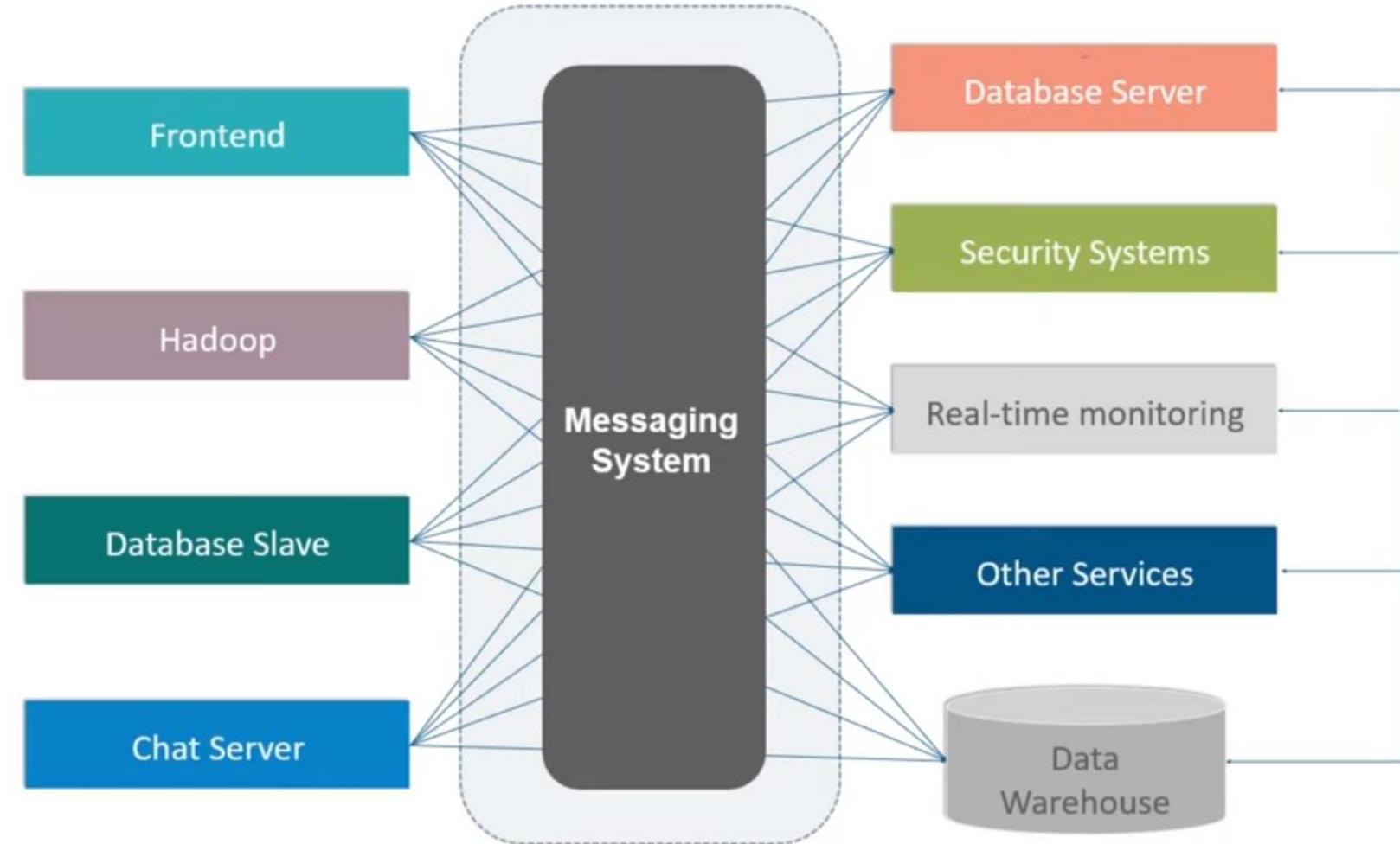
Complex Data Pipelines

Similarly, applications may also be communicating with Real-time monitoring and Other services in real-time scenario



Solution to the Complex Data Pipelines

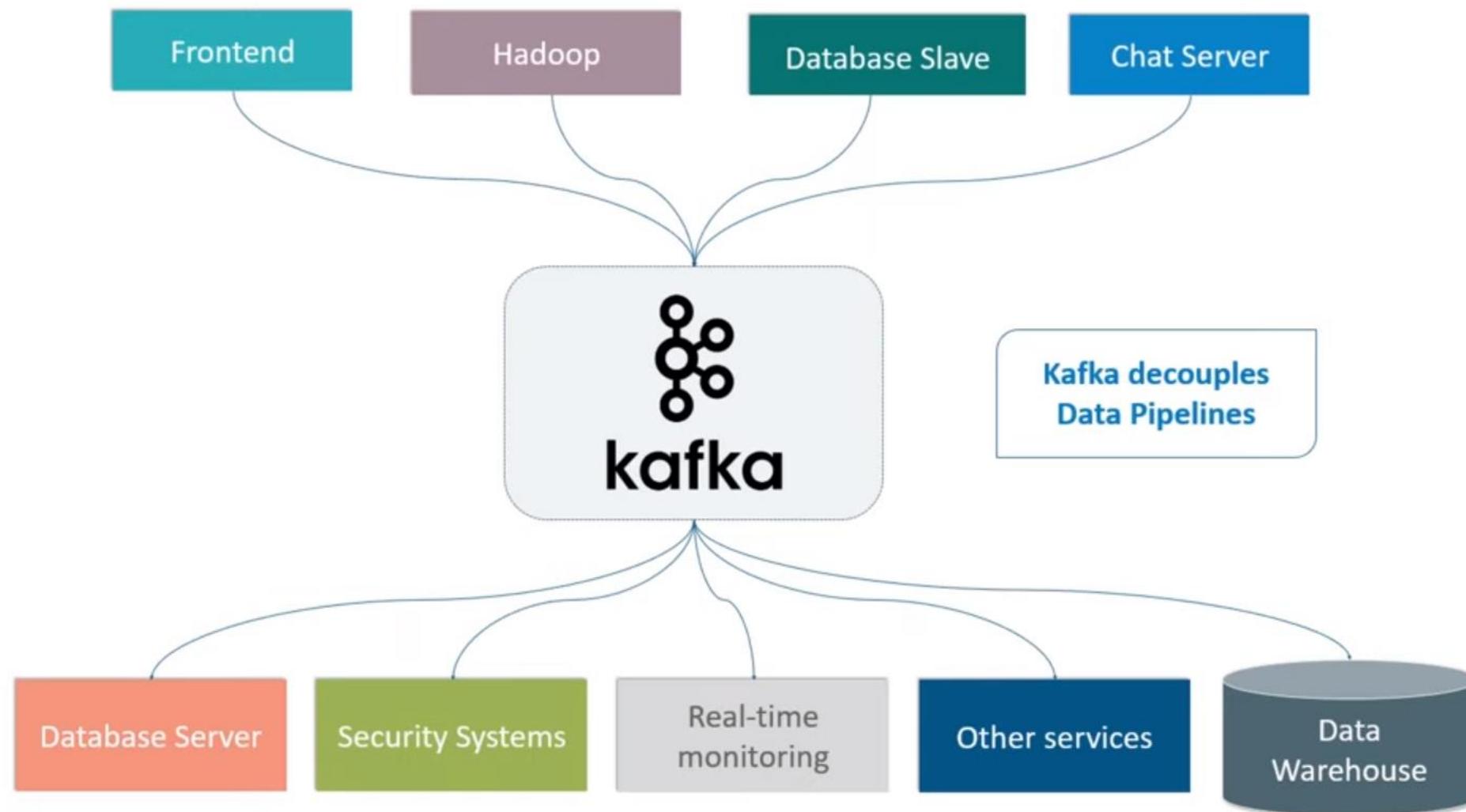
Messaging Systems helps managing the complexity of the pipelines





Let's See How Kafka Solves the Problem

Kafka Decouples Data Pipelines



What is Kafka?

- **Apache Kafka** is a distributed *publish-subscribe* messaging system
- It was originally developed at LinkedIn and later on became a part of Apache Project
- Kafka is fast, scalable, durable, fault-tolerant and distributed by design



Apache Kafka

A high-throughput distributed messaging system.

Kafka @LinkedIn

- 1100+ commodity machines
- 31,000+ topics
- 350,000+ partitions

- 675 billion messages/day
- 150 TB/day in
- 580 TB/day out

Peak Load

- 10.5 million messages/sec
- 18.5 GB/sec Inbound
- 70.5 GB/sec Outbound



Fig: A modern stream-centric data architecture built around Kafka

Kafka Growth Exploding

- More than **1/3** of all Fortune **500** companies use **Kafka**.
- These companies includes the top ten travel companies, **7** of top ten banks, **8** of top ten insurance companies, **9** of top ten telecom companies.
- **LinkedIn**, **Microsoft** and **Netflix** process billions of messages a day with Kafka (1,000,000,000,000).
- **Kafka** is used for **real-time streams** of data & used to collect big data for **real time analysis**.



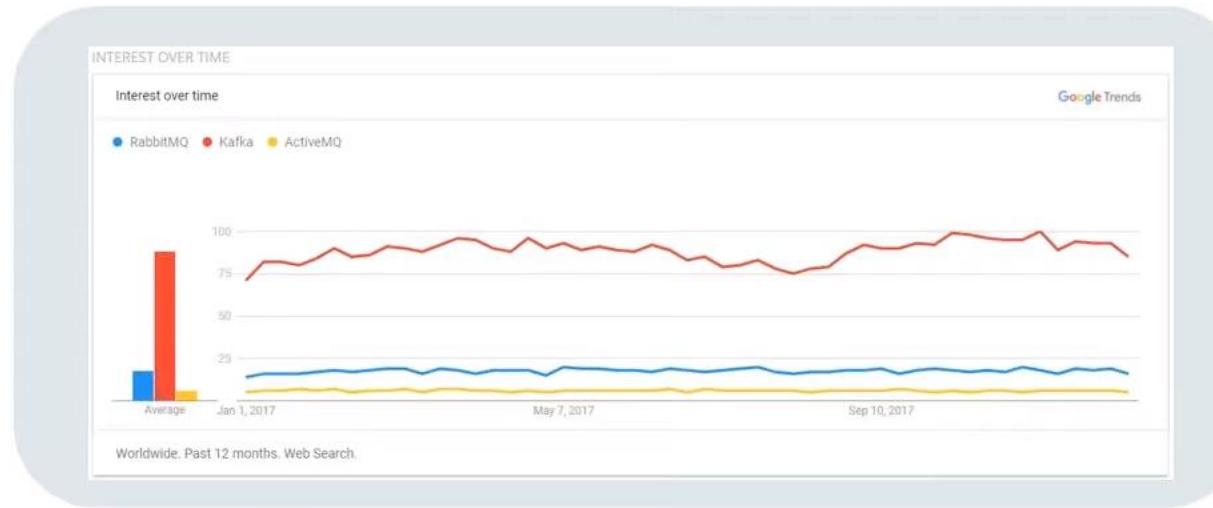
86% of respondents reported that the number of their systems that use Kafka is increasing



20% reported that the number is “growing a lot!”

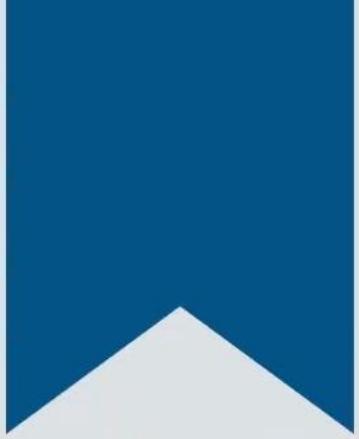


52% of organizations have at least **6** systems running Kafka



Source: Google Trends

Kafka Concepts



Kafka Terminologies

Producer

A **producer** can be any application who can publish messages to a topic

Consumer

A **consumer** can be any application that subscribes to a topic and consume the messages

Partition

Topics are *broken up into ordered commit logs called partitions*

Broker

Kafka cluster is a set of servers, each of which is called a **broker**

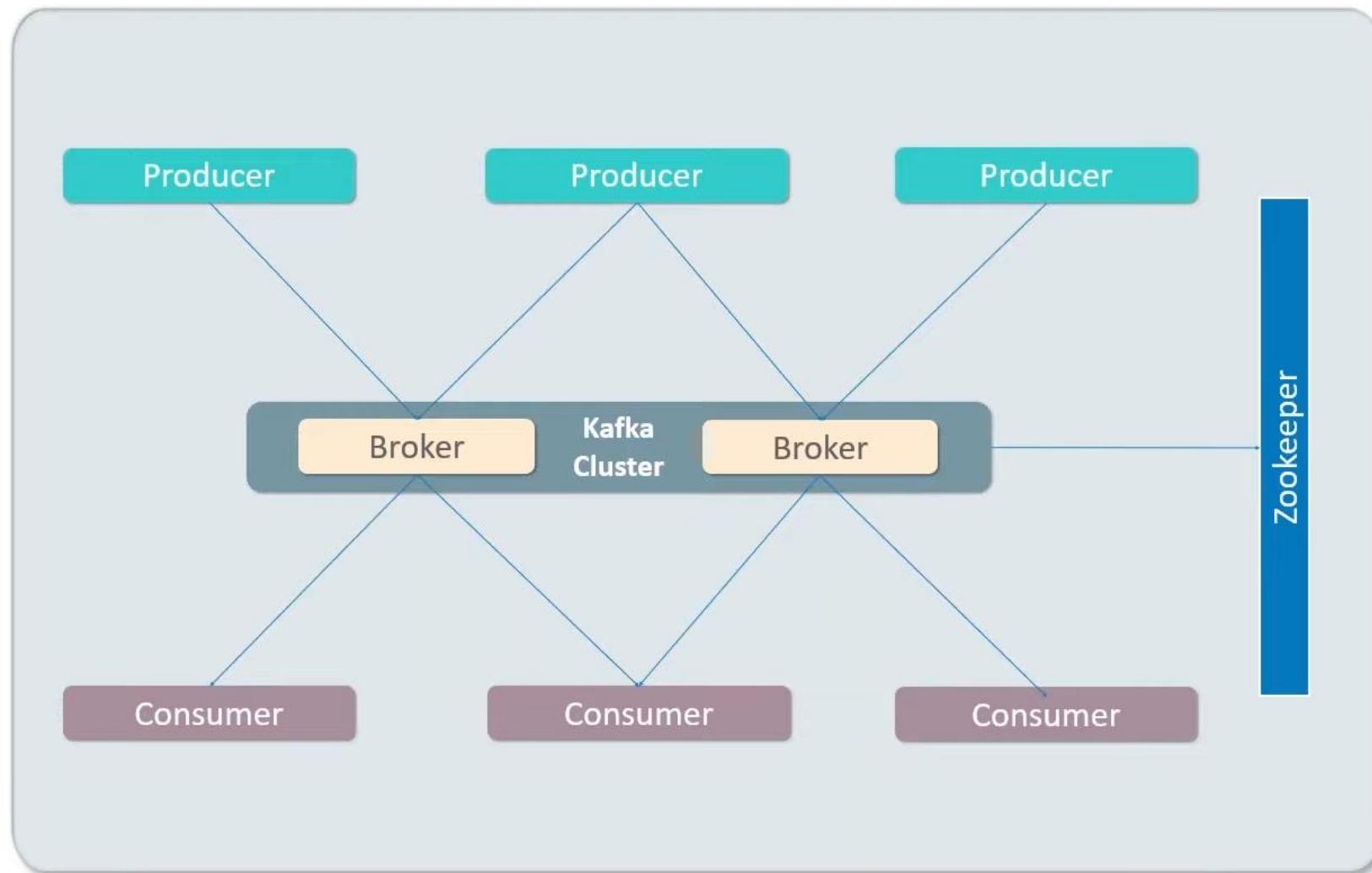
Topic

A **topic** is a category or *feed name* to which *records are published*

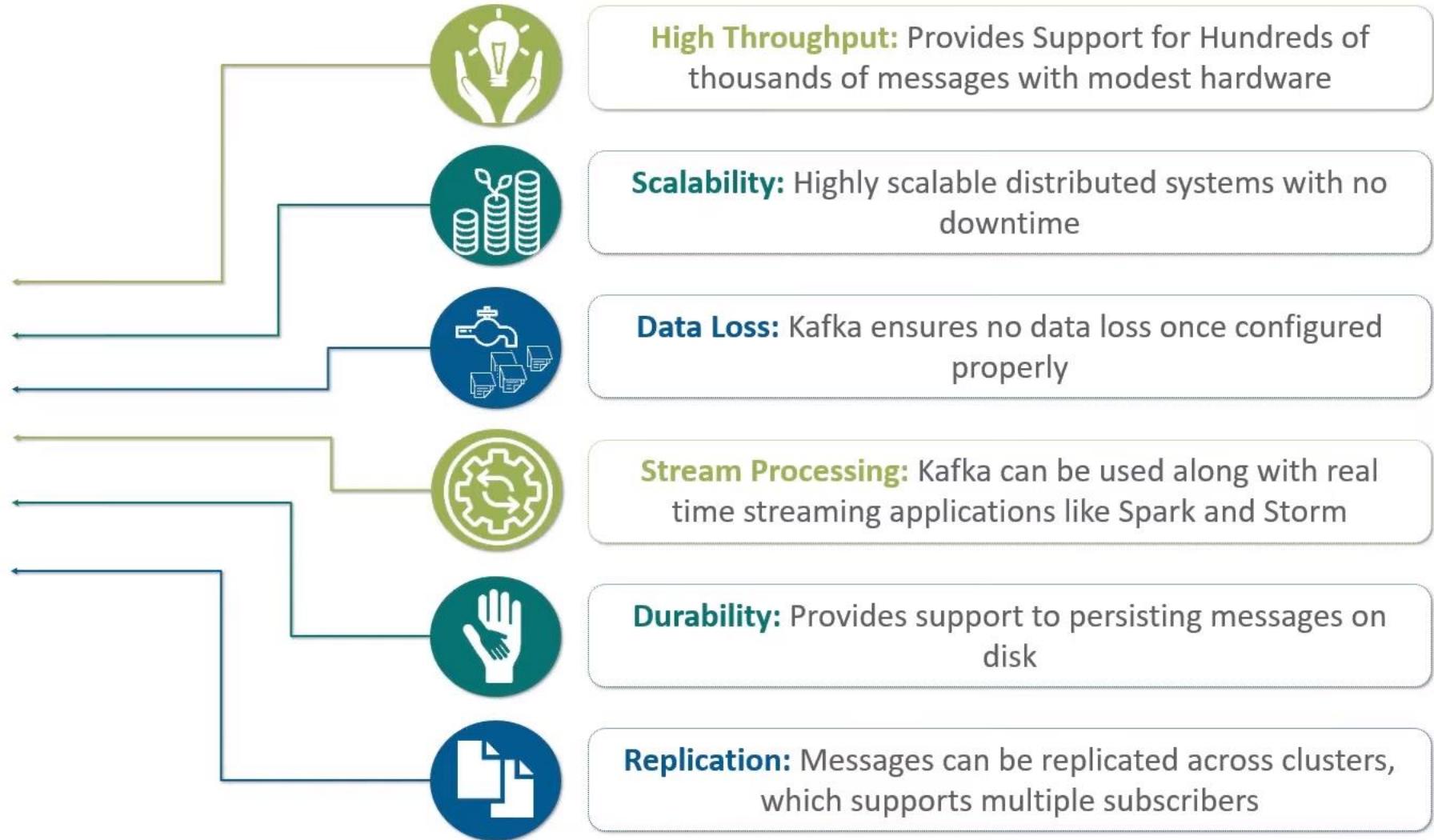
Zookeeper

ZooKeeper is used for managing and coordinating Kafka broker

Kafka Cluster



Kafka Features



Kafka Components - Topics and Partitions



A *topic* is a category or *feed name* to which *records are published*



Topics are broken up into *ordered commit logs* called *partitions*



Each *message* in a *partition* is assigned a *sequential id* called an *offset*



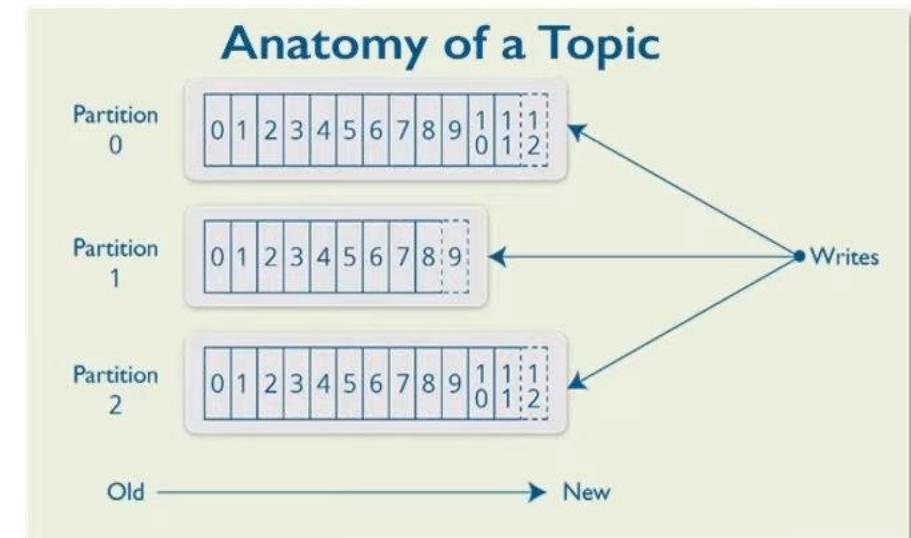
Data in a topic is retained for a *configurable period of time*



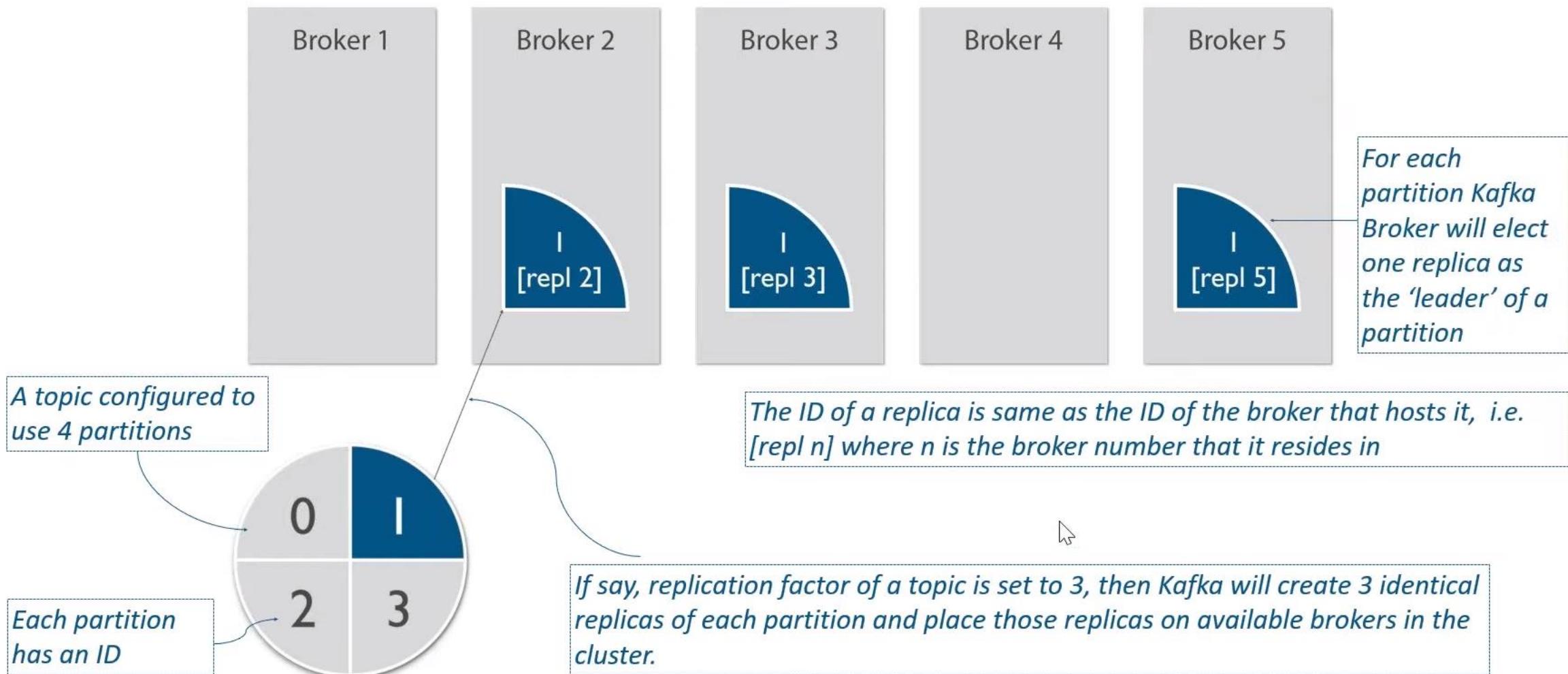
Writes to a partition are generally *sequential* thereby *reducing the number of hard disk seeks*



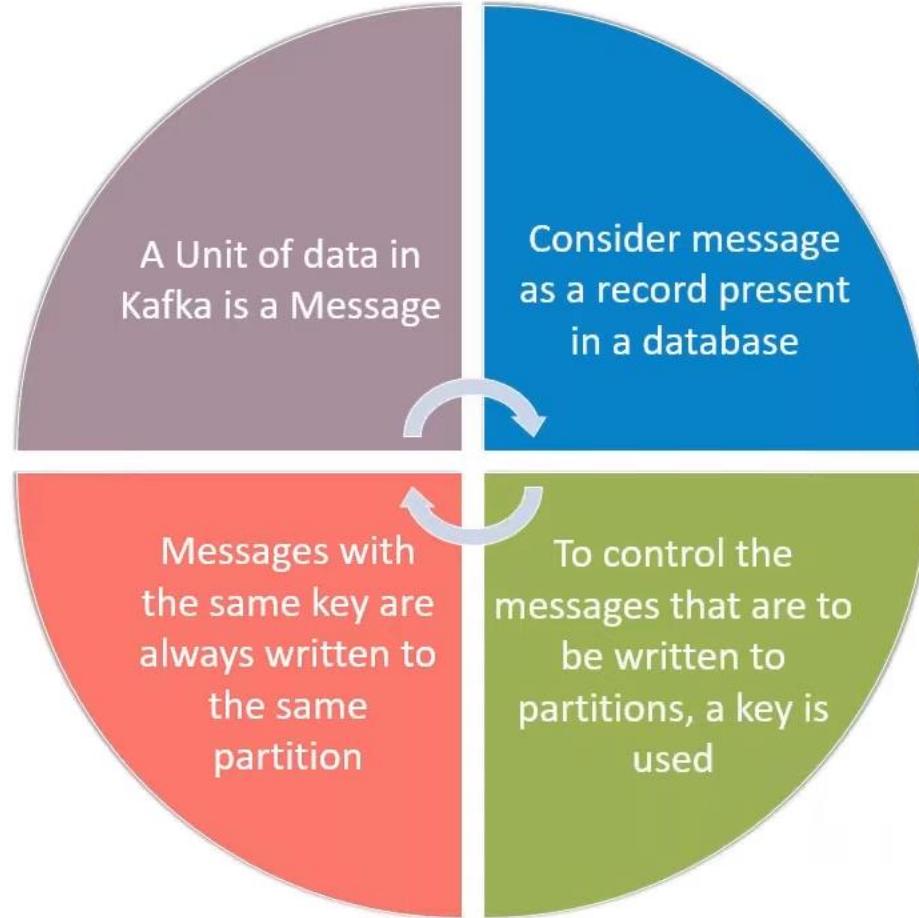
Reading messages can either be from *beginning* & also can *rewind* or *skip* to any point in partition by *giving an offset value*



Kafka Components - Topics, Partitions & Replicas



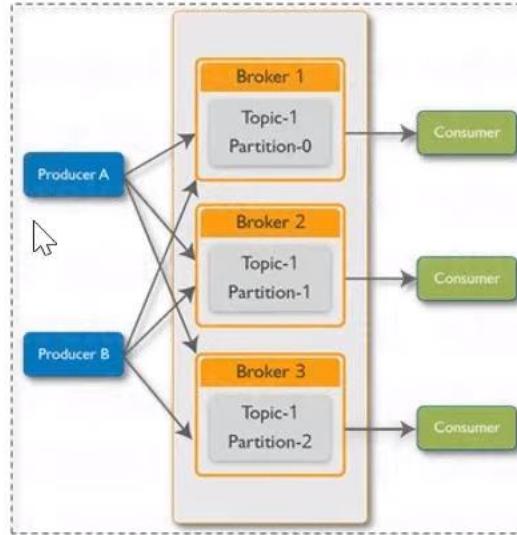
Kafka Components - Messages



Kafka Components - Producer

1

Producer (publisher or writer) publishes a new message to a **specific topic**

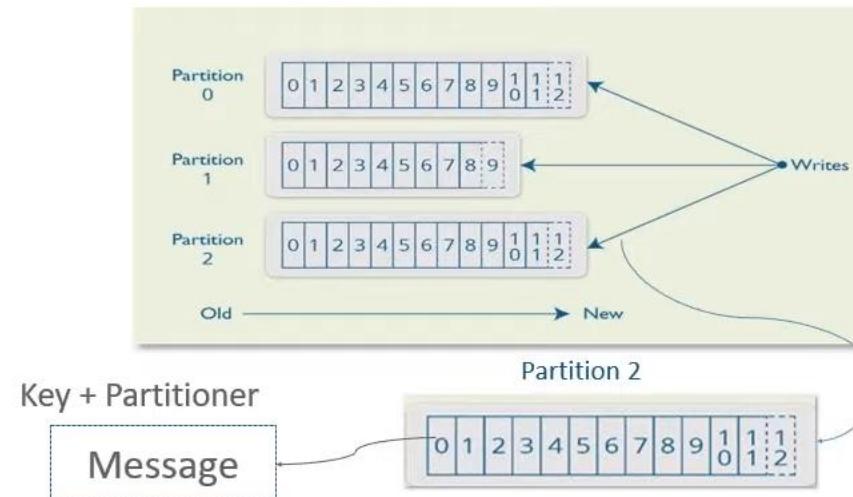


2

The producer does not care what partition a specific message is written to and will balance messages over every partition of a topic evenly

3

Directing messages to a partition is done using the **message key** and a **partitioner**, this will generate a hash of the key and map it to a partition

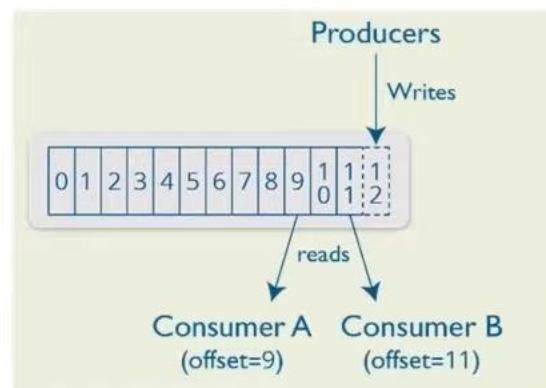


4

Every message a producer publishes in the form of a **key : value** pair

Kafka Components - Consumer

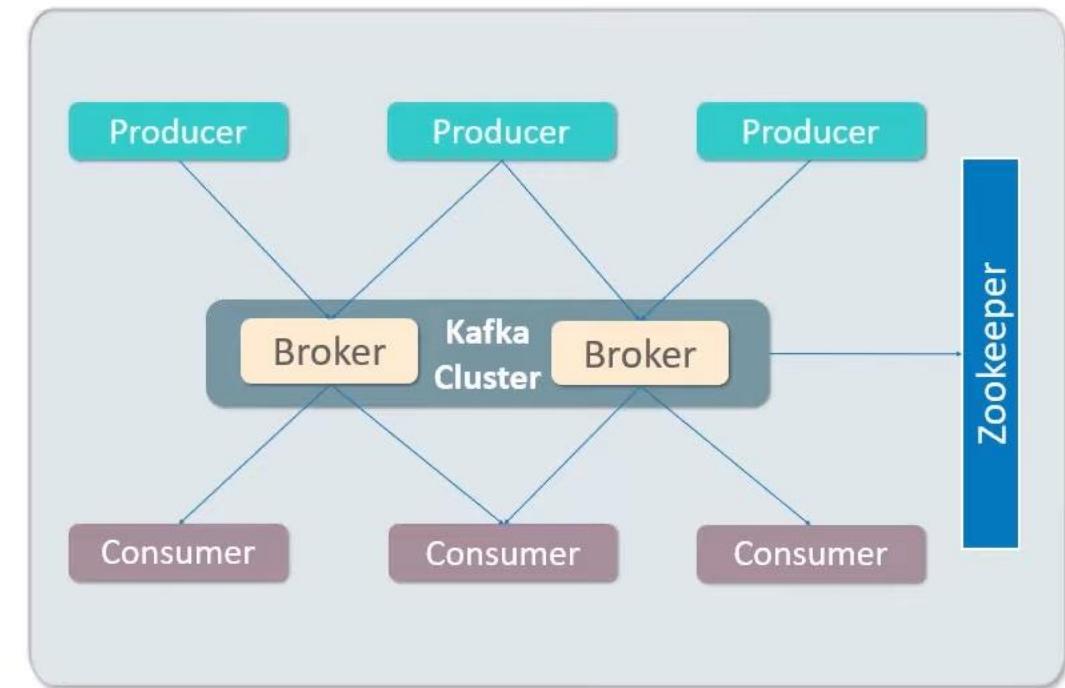
- Consumers(subscribers or readers) read messages
- The consumer subscribes to one or more topics and reads the messages sequentially
- The consumer keeps track of the messages it has consumed by keeping track on the offset of messages
- The *offset* is bit of metadata(an integer value that continually increases)that Kafka adds to each message as it is produced
- Each partition has a *unique offset* which is stored
- With the offset of the last consumed message, a consumer can *stop and restart without losing its current state*



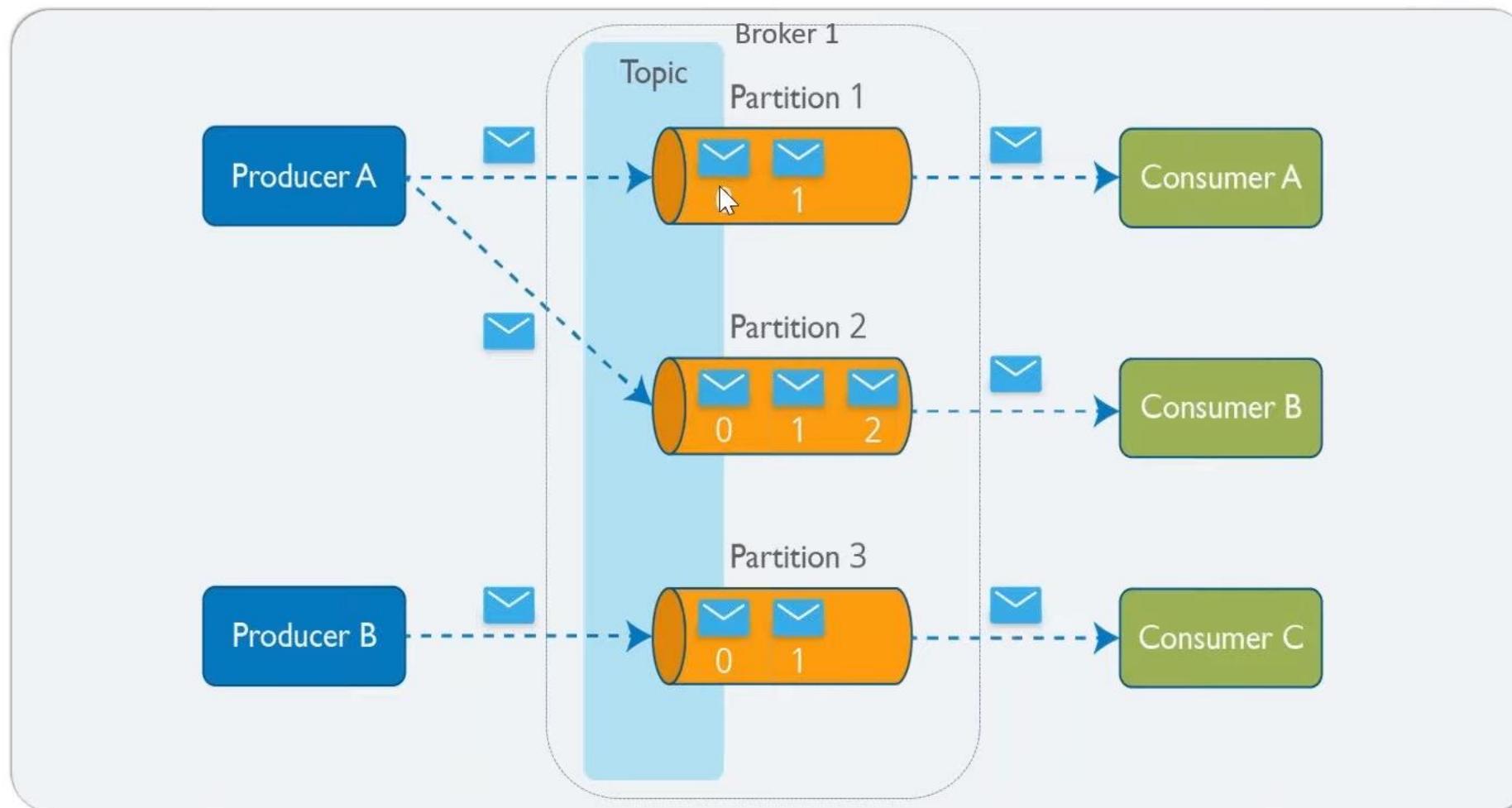
Kafka Components - ZooKeeper

ZooKeeper is used for managing and coordinating Kafka broker

- Zookeeper service is mainly used for co-ordinating between brokers in the Kafka cluster
- Kafka cluster is connected to ZooKeeper to get information about any failure nodes



Kafka Architecture



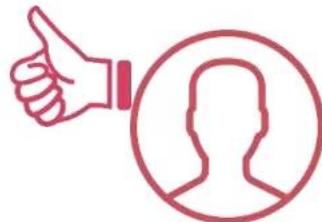
Let's see some Use Cases of Kafka

Kafka - Use Cases



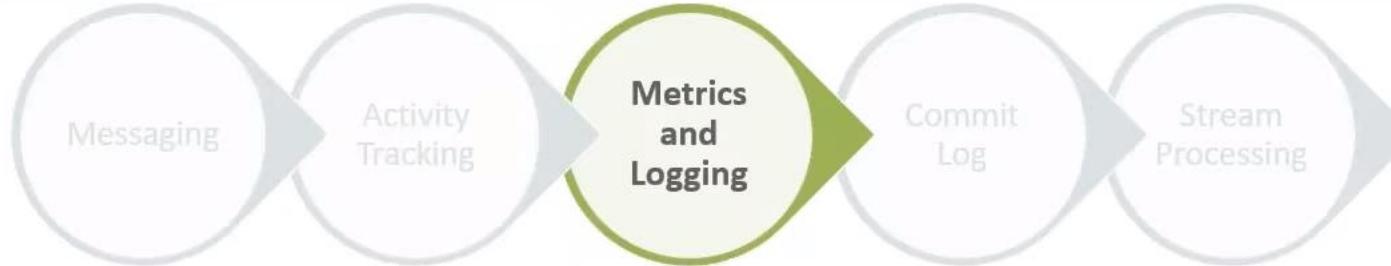
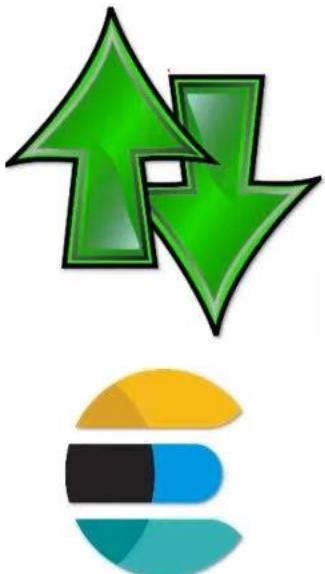
- Applications can produce messages using Kafka, without being concerned about the format of the messages
- Messages are sent and handled by a single application that can read all of them consistently, including :
 - A common formatting of messages using a common look
 - Send multiple messages in a single notification
 - Receive messages in a way that meets the users preferences

Kafka - Use Cases



- Originally Kafka was designed at LinkedIn, to track user activity
- When a user interacts with frontend applications, which generates messages regarding actions the user is taking
- Kafka keeps track of simple information like click tracking to complex information like data in a user's profile

Kafka - Use Cases



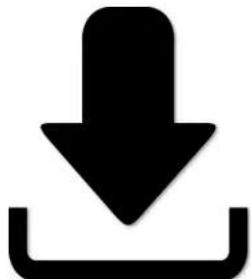
- Kafka is also ideal for collecting application's and system metrics and logs
- Applications publish metrics on a regular basis to a Kafka topic, and those metrics can be consumed by systems for monitoring and alerting
- Log messages can be published in the same way and routed to dedicated log search systems like Elasticsearch or security analysis applications

Kafka - Use Cases



- Database changes can be published to Kafka and applications can easily monitor this stream to receive live updates as they happen
- Kafka replicates database updates to a remote system for consolidating changes from multiple applications in a single database view
- Durable retention becomes useful providing a buffer for the changelog, meaning it can be replayed in the event of a failure of the consuming applications
- Log-compacted topics can be used to provide longer retention by only retaining a single change per key

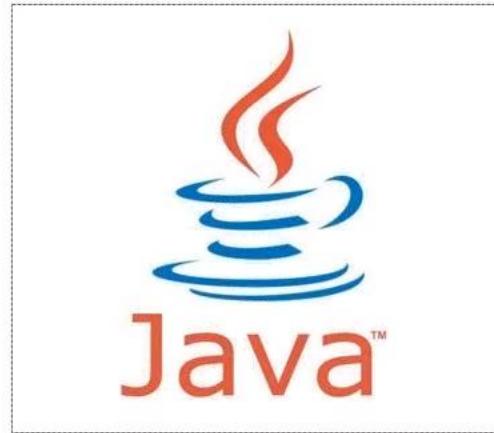
Kafka - Use Cases



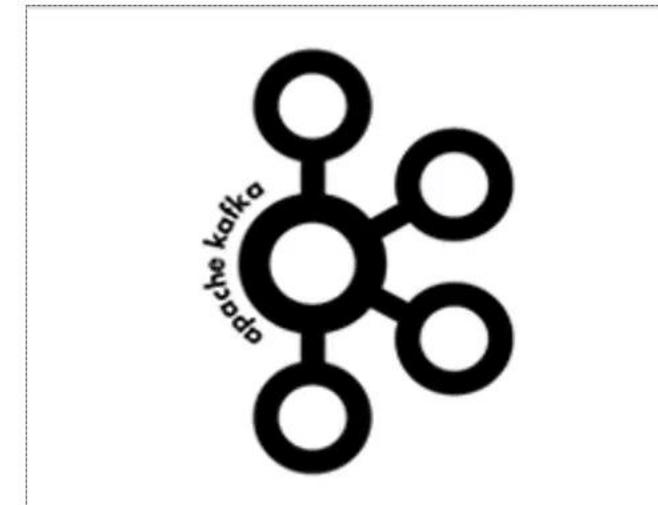
- Stream processing term is typically used to refer applications that provide similar functionality to map/reduce processing in Hadoop
- Stream processing operates on data in real-time, as quickly as messages are produced :
 - Write small applications to operate on Kafka messages,
 - Performing tasks such as counting metrics
 - Partitioning messages for efficient processing by other applications

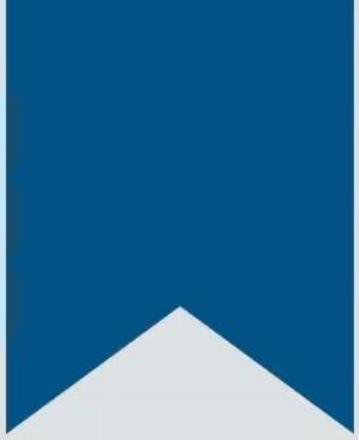
Getting Started with Kafka

- Prerequisites :



- Components :

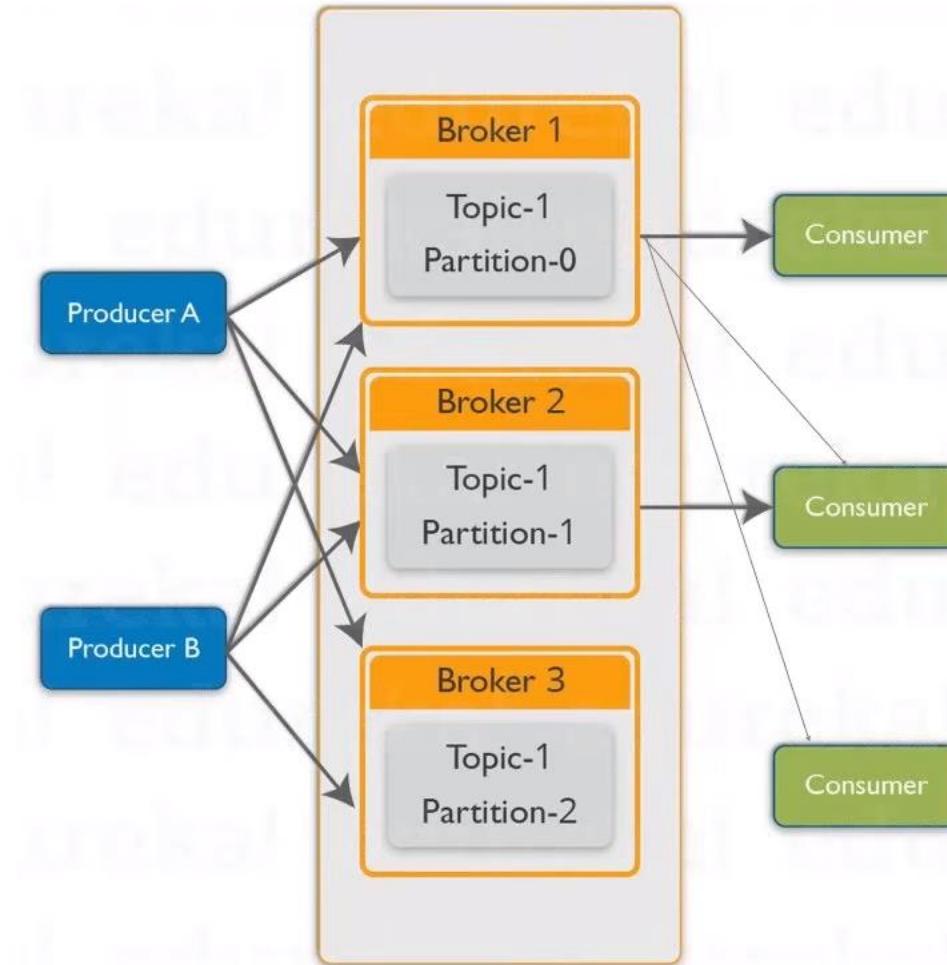




Let's Classify Different Types of Clusters in Kafka

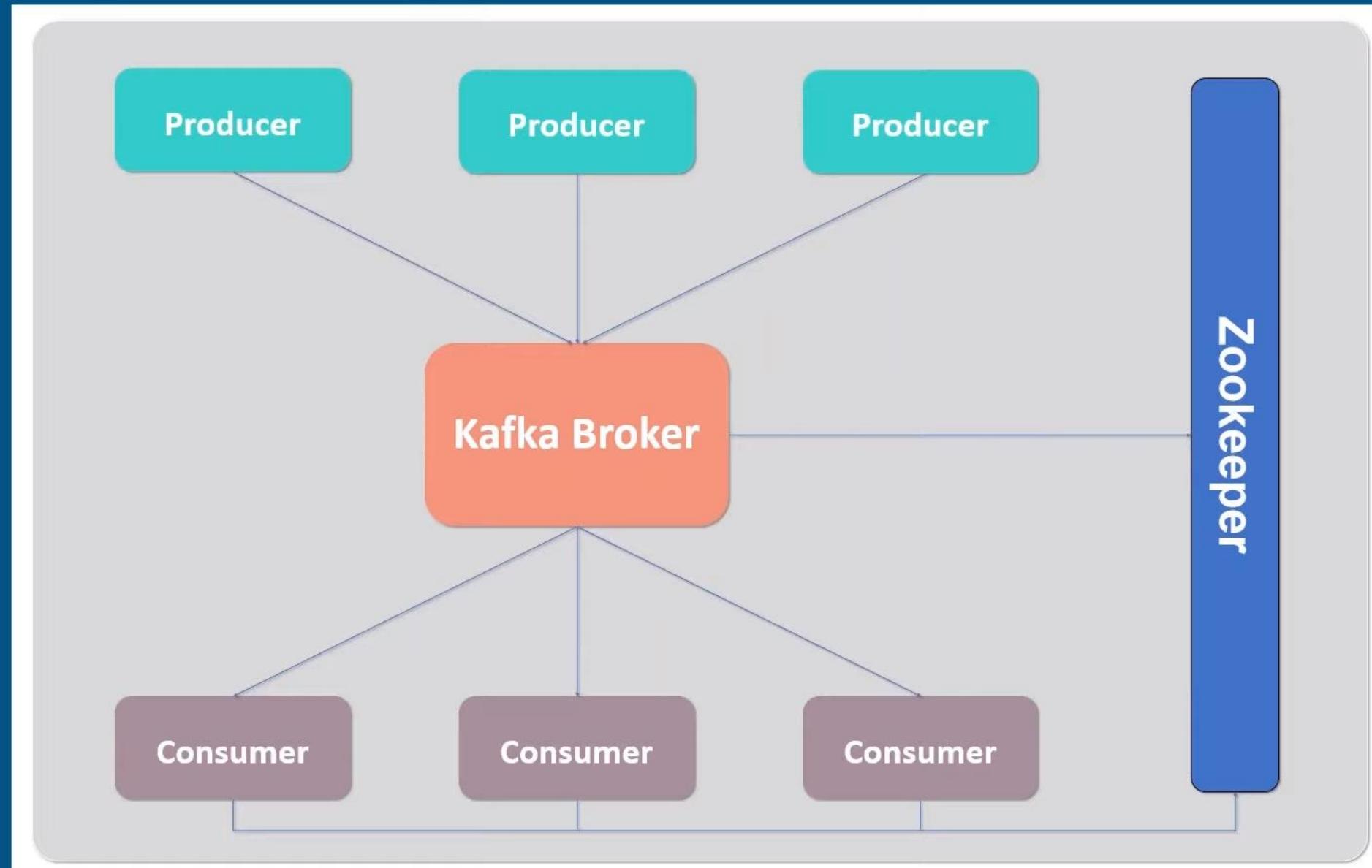
Kafka Cluster

- Kafka brokers are designed to operate as part of a cluster
- One broker will also function as the cluster controller
- Controller is responsible for administrative operations, like
 - Assigning partitions to brokers
 - Monitoring for broker failures in a cluster
- A particular partition is owned by a broker, and that broker is called the leader of the partition
- All consumers and producers operating on that partition must connect to the leader



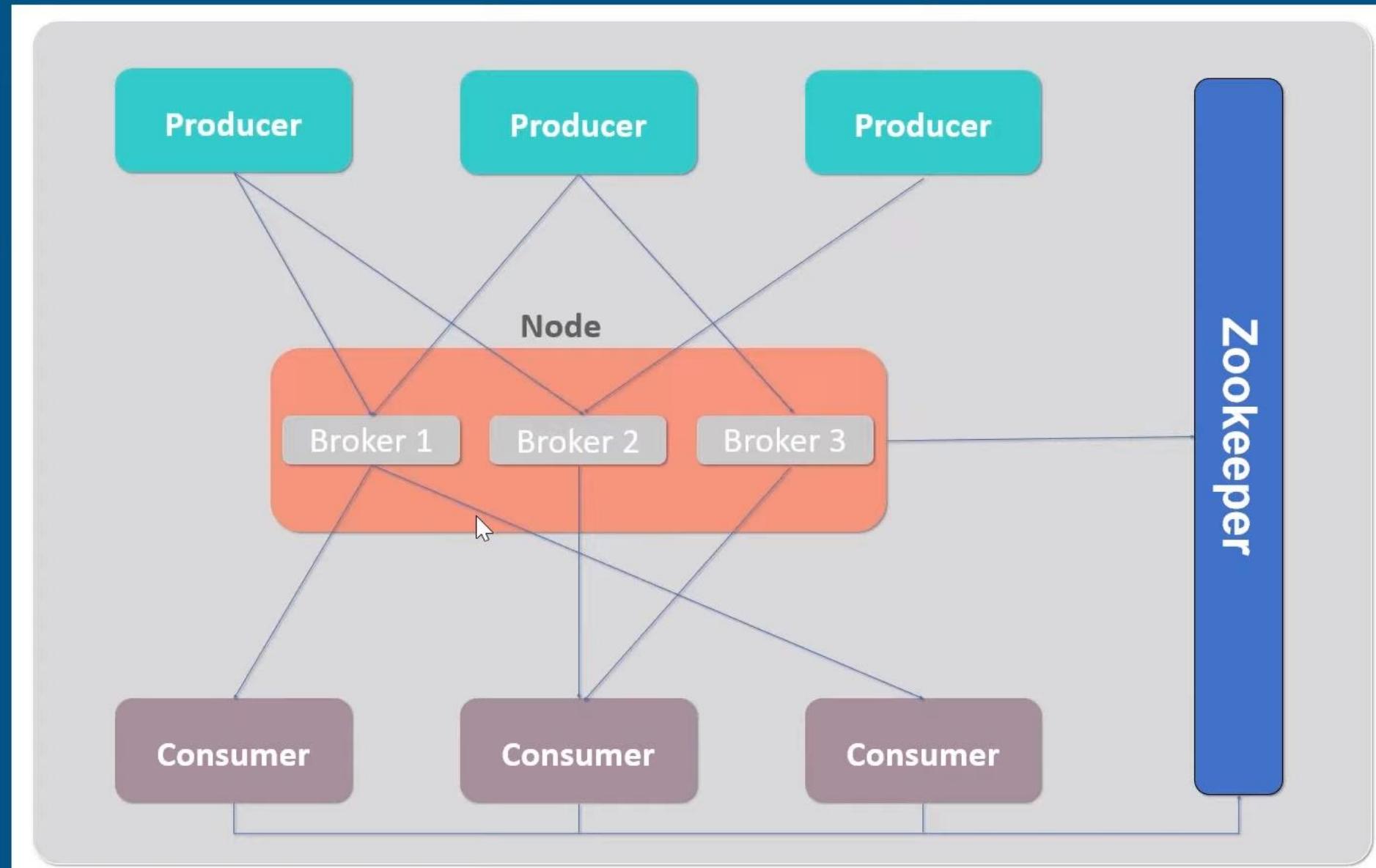
Type of Kafka Clusters

- 1 Single Node-Single Broker Cluster
- 2 Single Node-Multiple Broker Cluster
- 3 Multiple Nodes-Multiple Broker Cluster



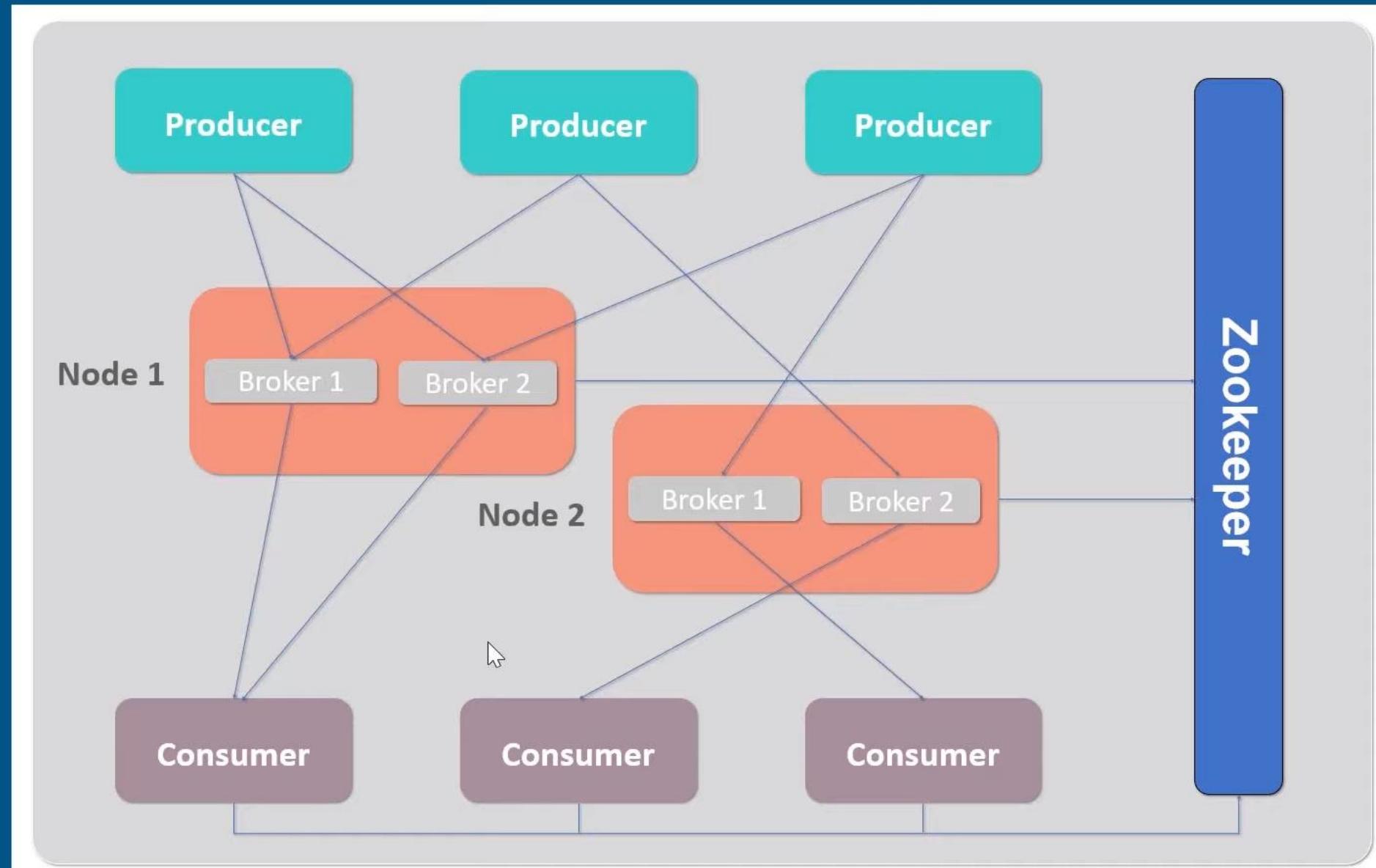
Type of Kafka Clusters

- 1 Single Node-Single Broker Cluster
- 2 Single Node-Multiple Broker Cluster
- 3 Multiple Nodes-Multiple Broker Cluster



Type of Kafka Clusters

- 1 Single Node-Single Broker Cluster
- 2 Single Node-Multiple Broker Cluster
- 3 Multiple Nodes-Multiple Broker Cluster

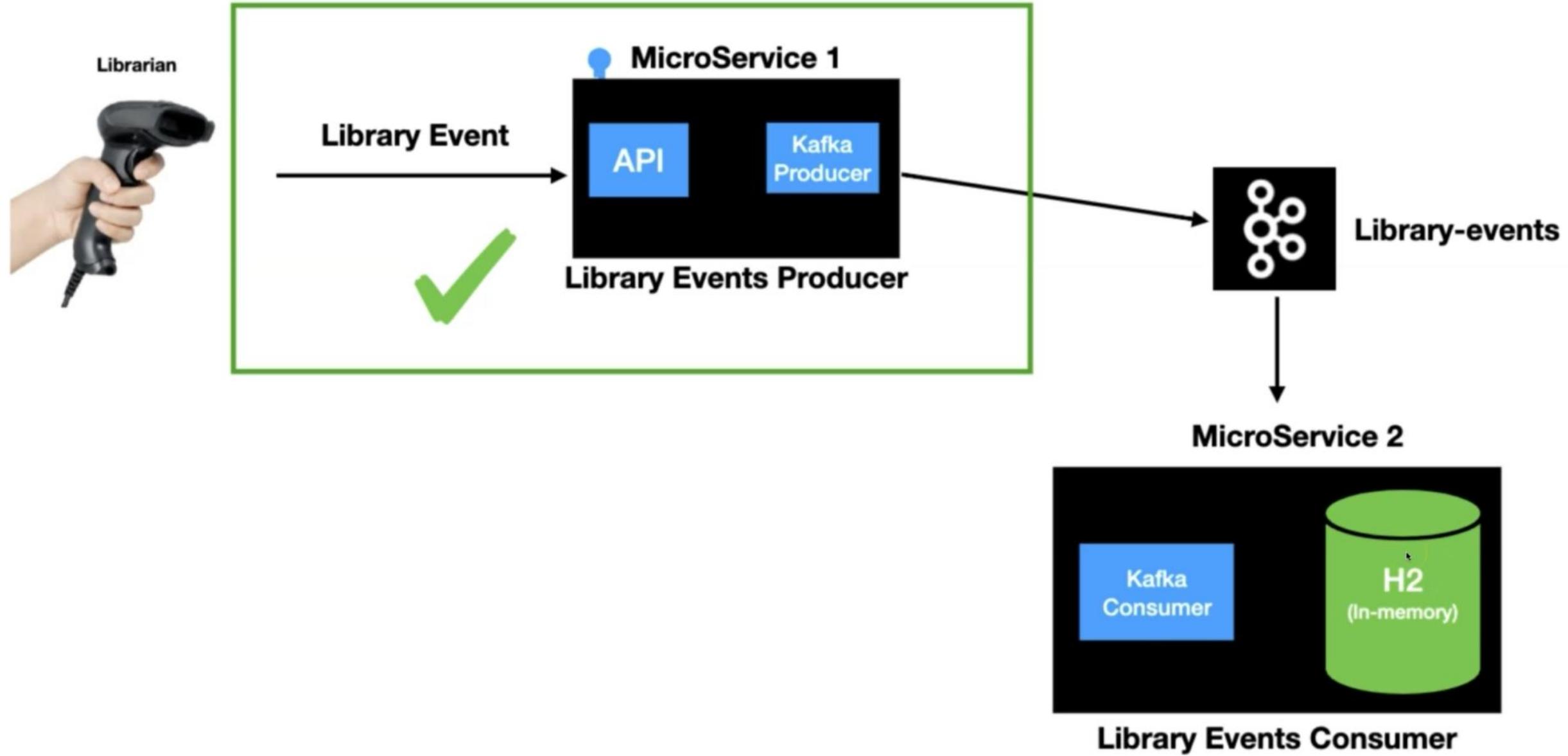


“

Getting Started with KAFKA & Spring Boot

”

Producer & Consumer



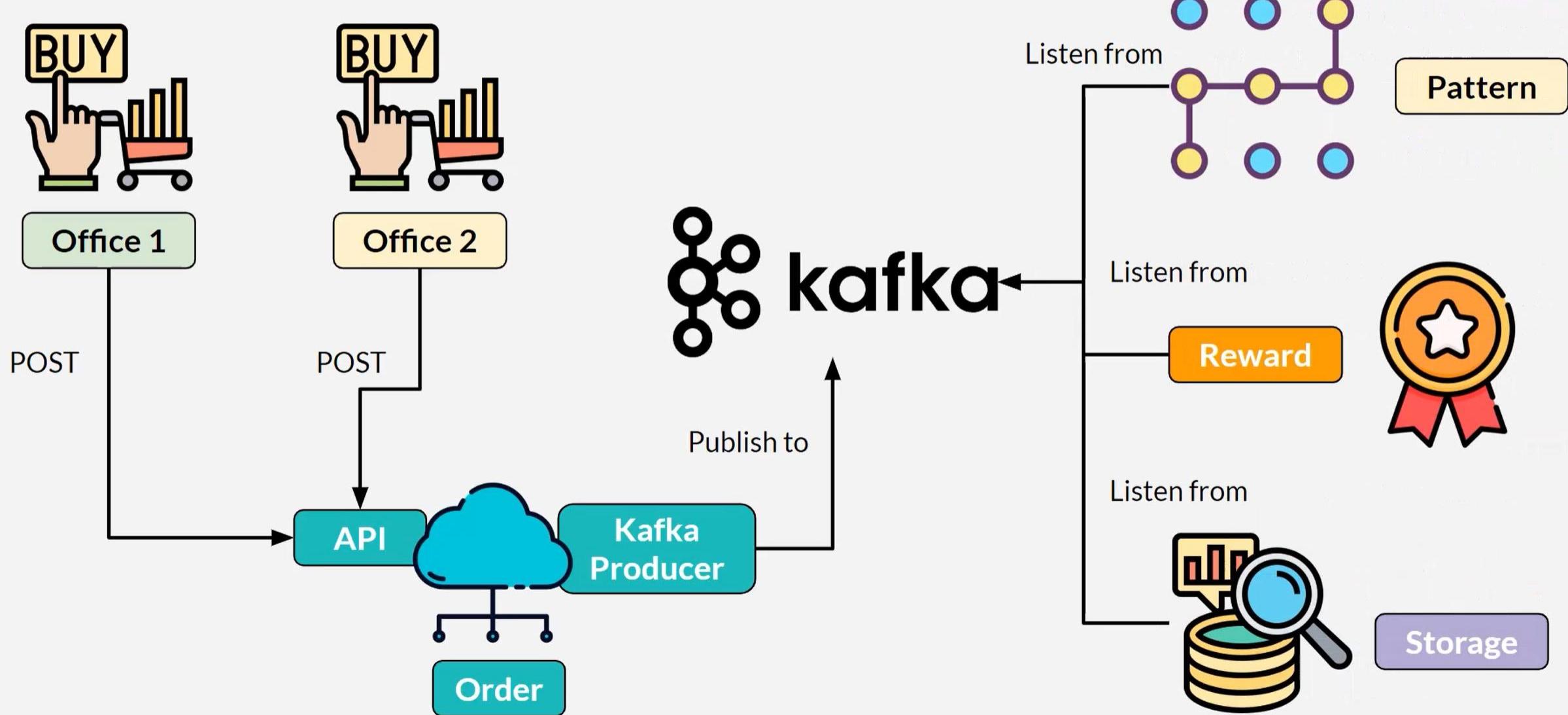
“

KAFKA BASIC PROJECT

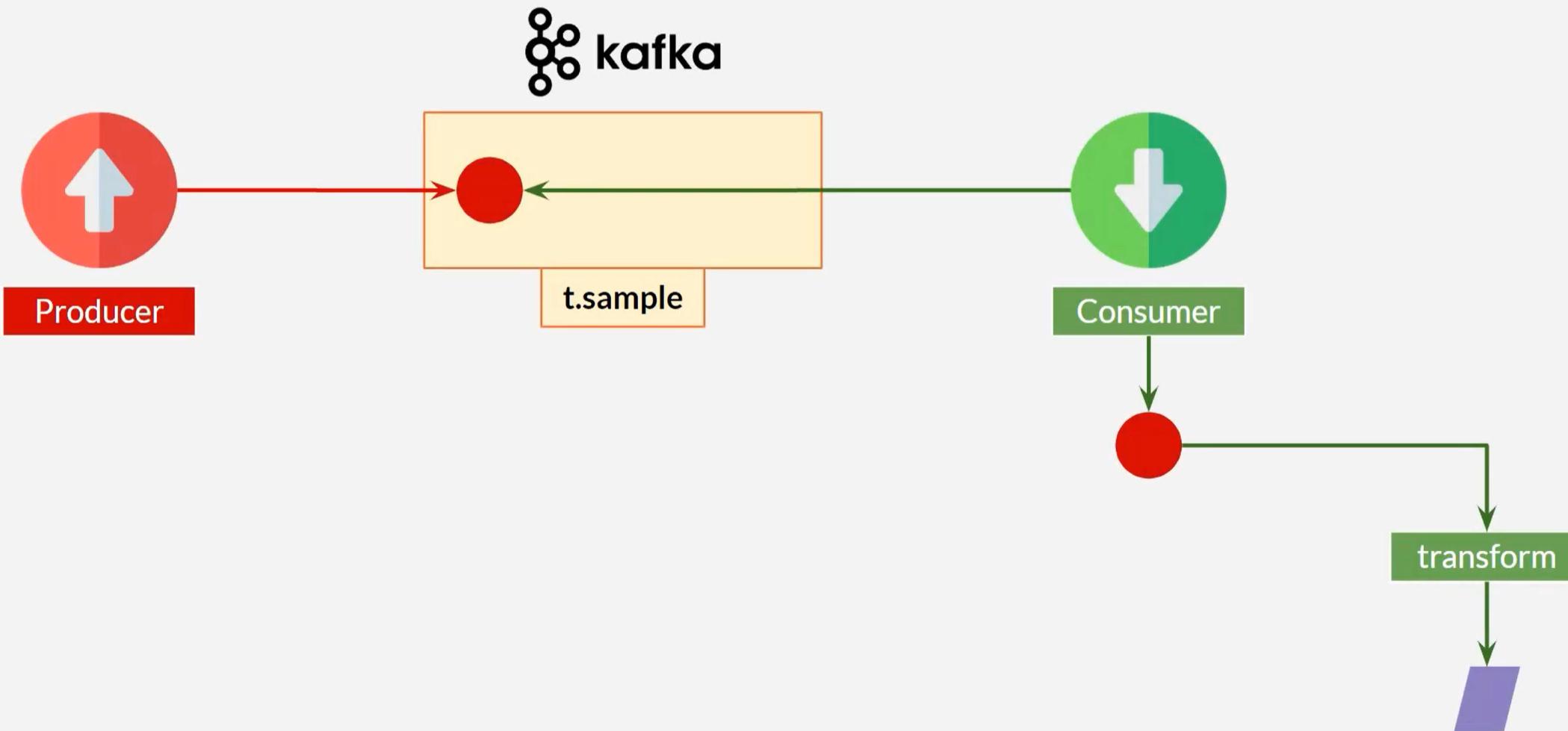
”



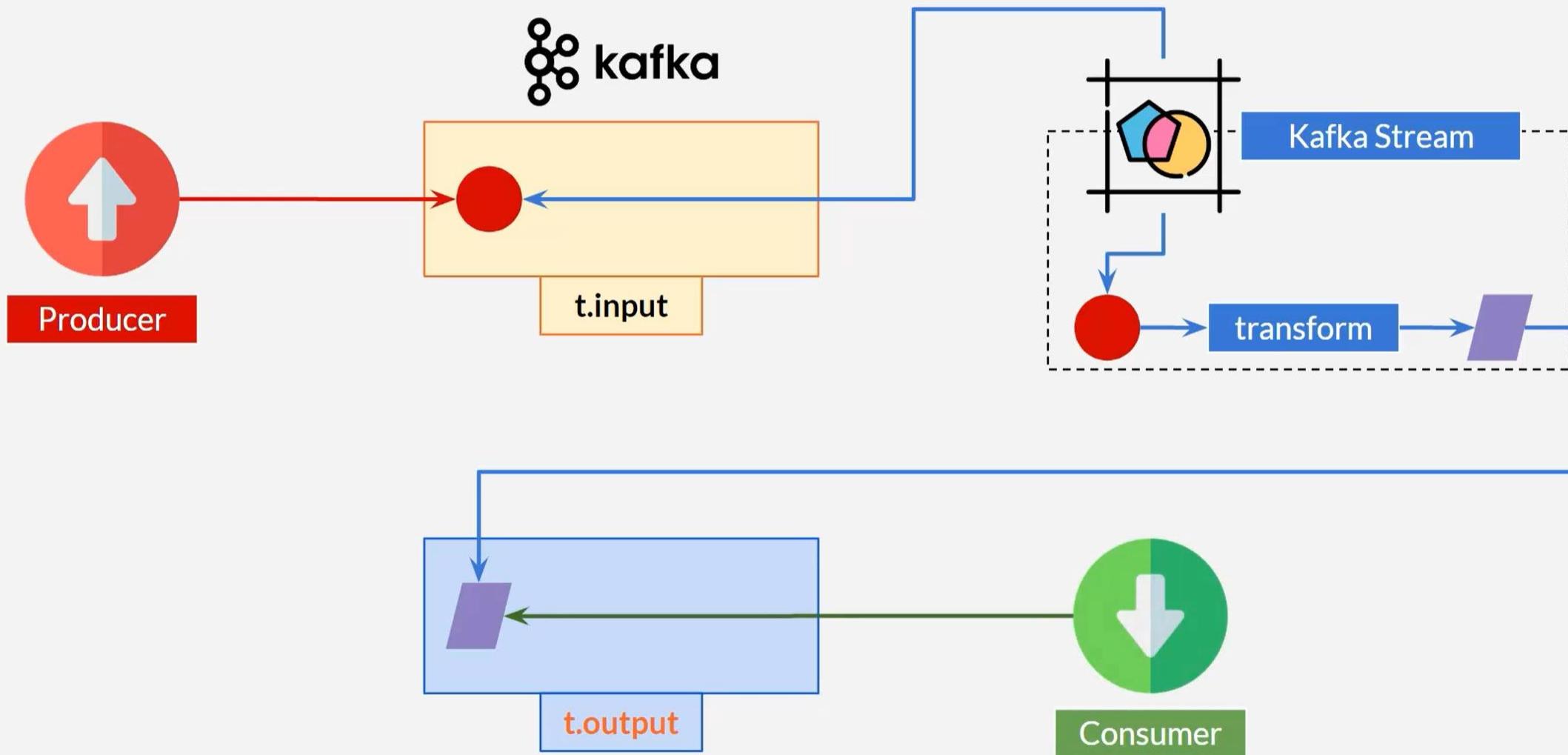
Application Architecture

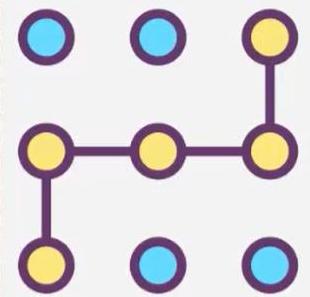


Data Transformation



Kafka Stream





Pattern

Spring Kafka

Kafka Stream

Reward



Spring Kafka

Kafka Stream



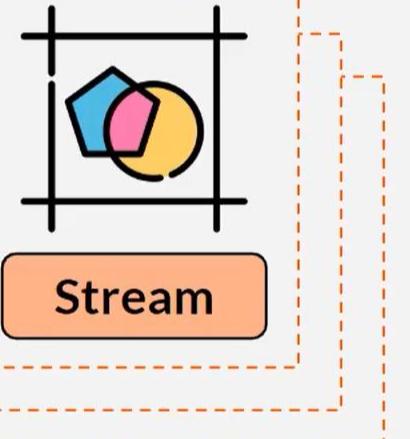
Storage

Spring Kafka

Kafka Stream

Kafka Stream

Spring Kafka



Stream

Venkat
Corporate Trainer & Motivational Speaker

