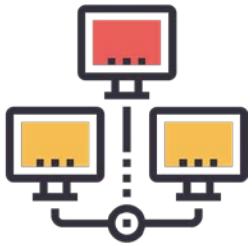
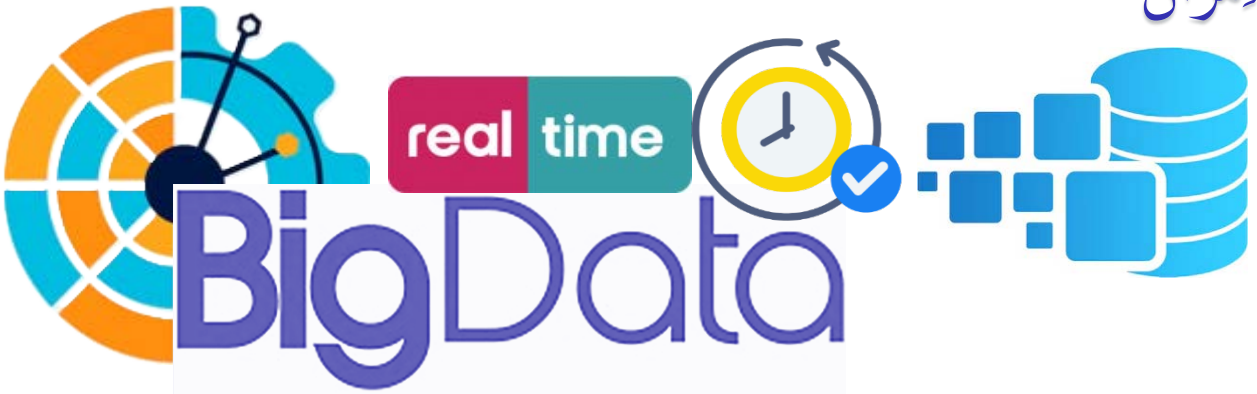




Dr. SOHAIL IMRAN

Introduction

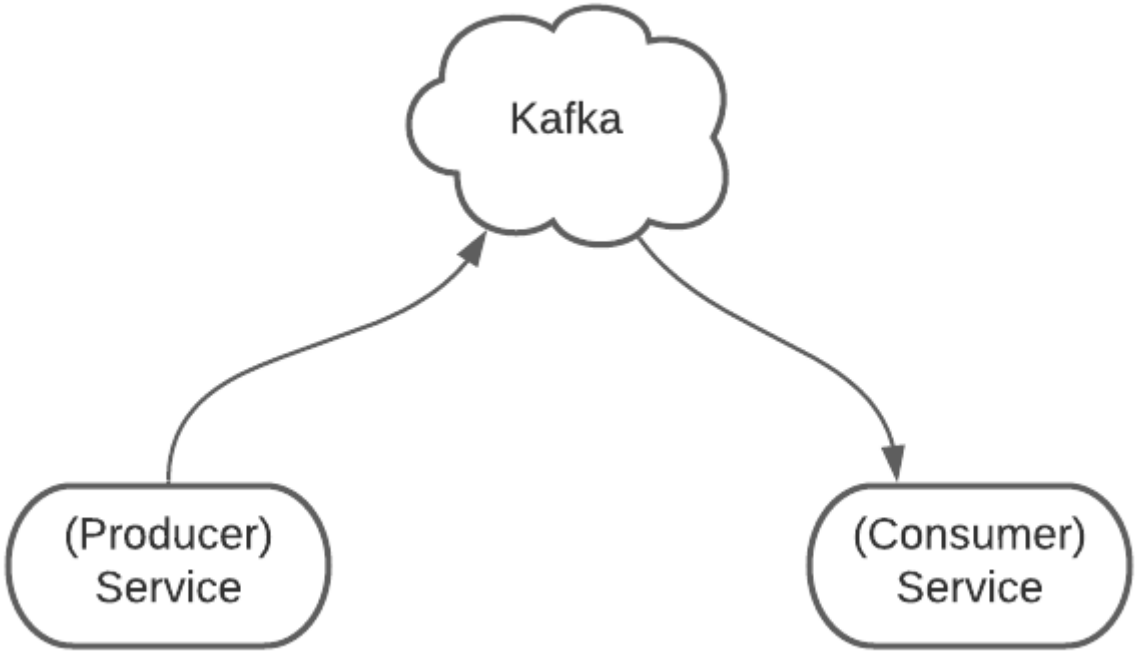
sohailimran2@yahoo.com

Ref: Web

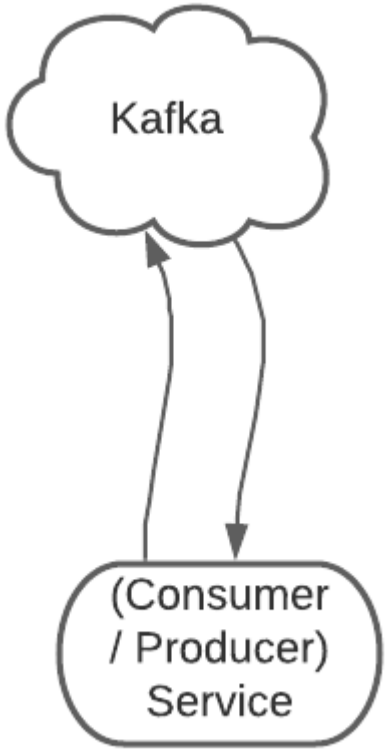
services

Apache Kafka is an event streaming software. It allows backend services (usually in micro-services architecture) to communicate with each other.

Producers and consumers are services that listen to or send messages in Kafka.



A service can be both a consumer and producer.



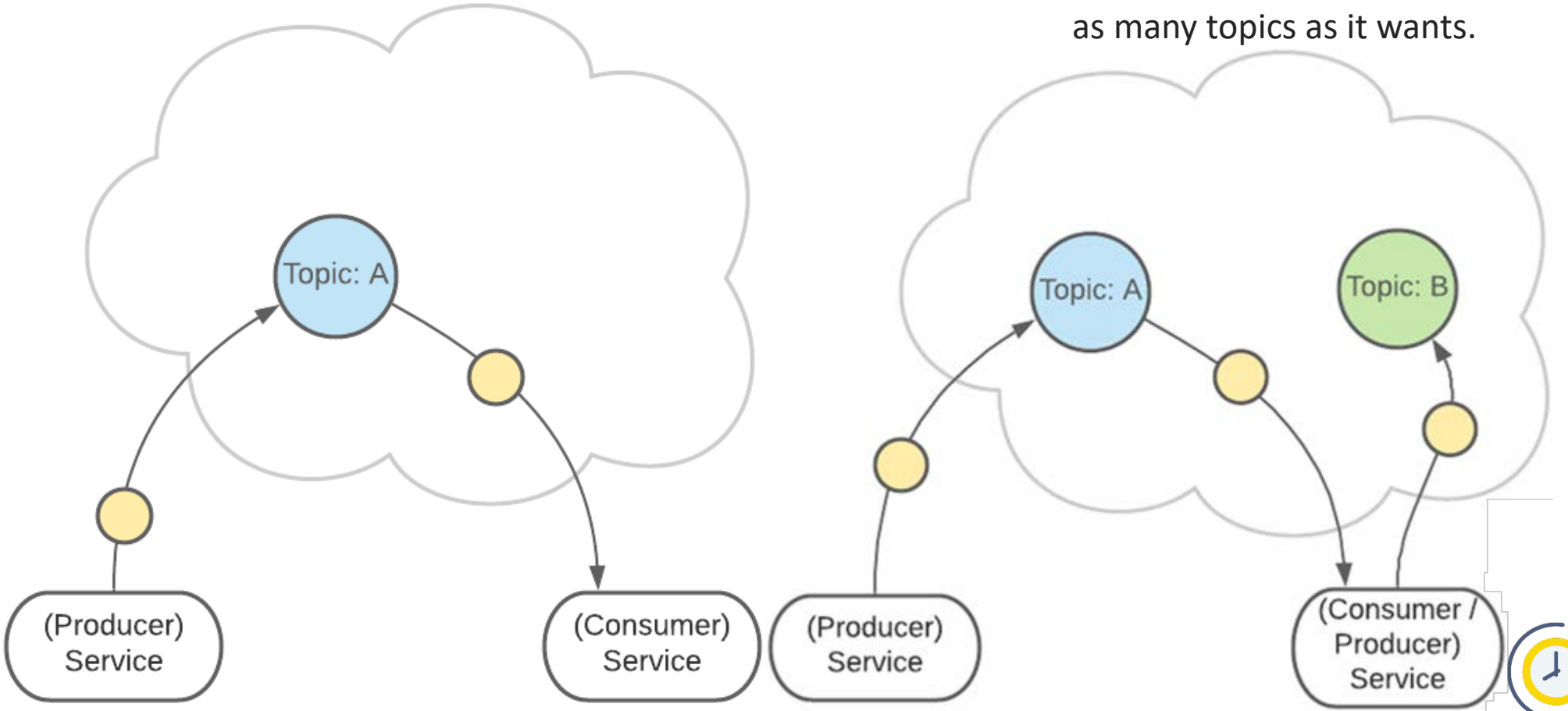
A service listening to messages and consuming them

real time

topics

Topics are addresses that producers can send messages to. Consumers can listen to these topics.

A consumer can listen and a producer can send messages to as many topics as it wants.

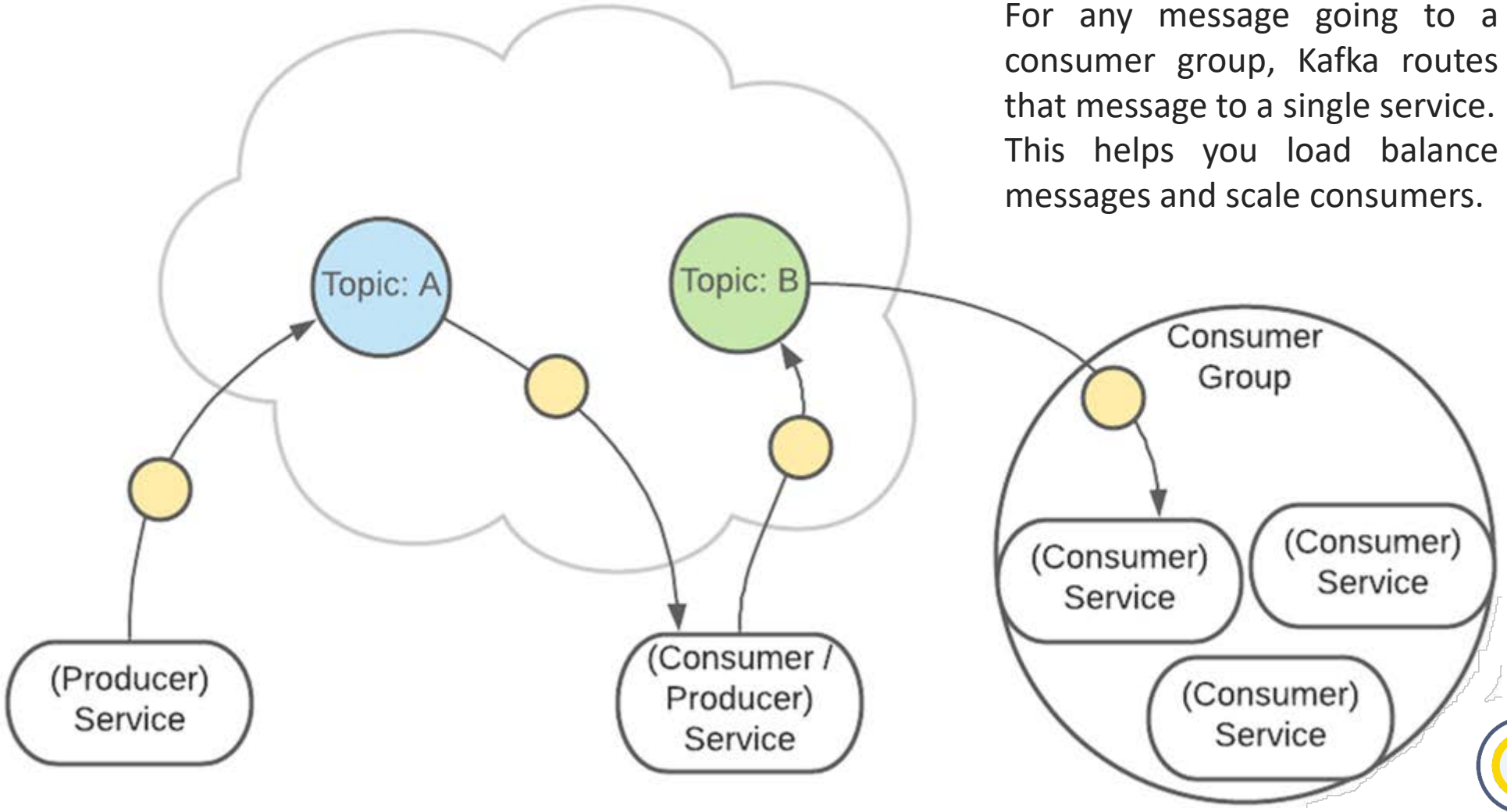


A procedure emitting a message and a consumer receiving a message from a Kafka topic

real time

consumer-group

For any message going to a consumer group, Kafka routes that message to a single service. This helps you load balance messages and scale consumers.

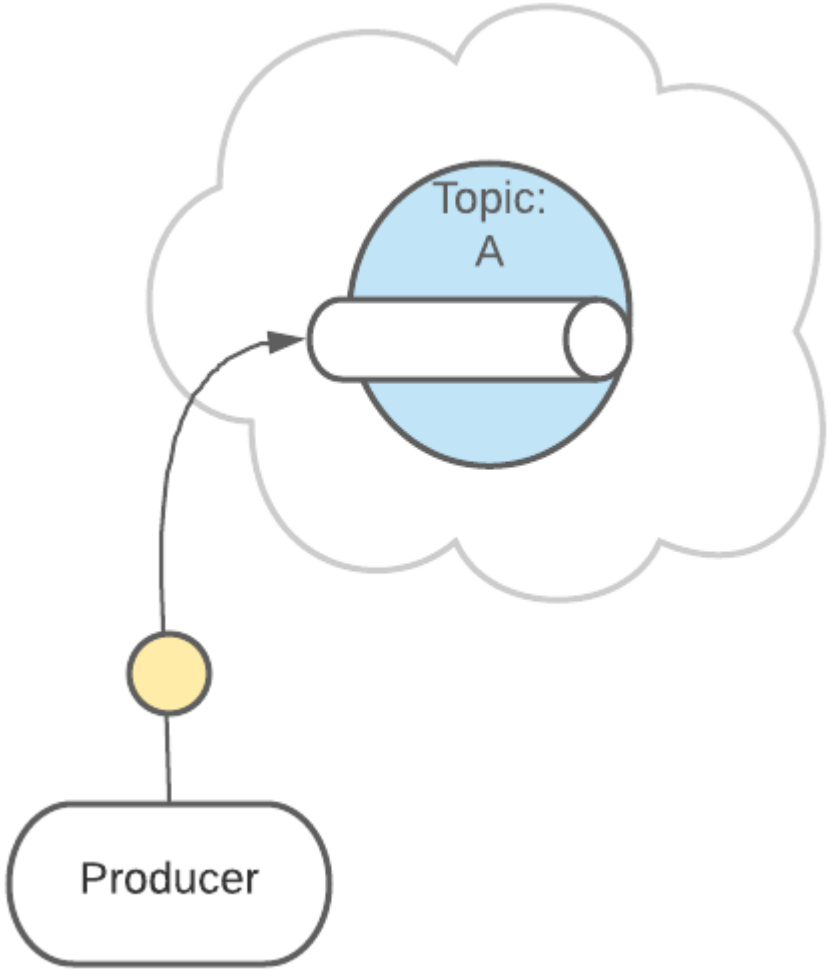


A consumer-group is a group of services that act as a single consumer.

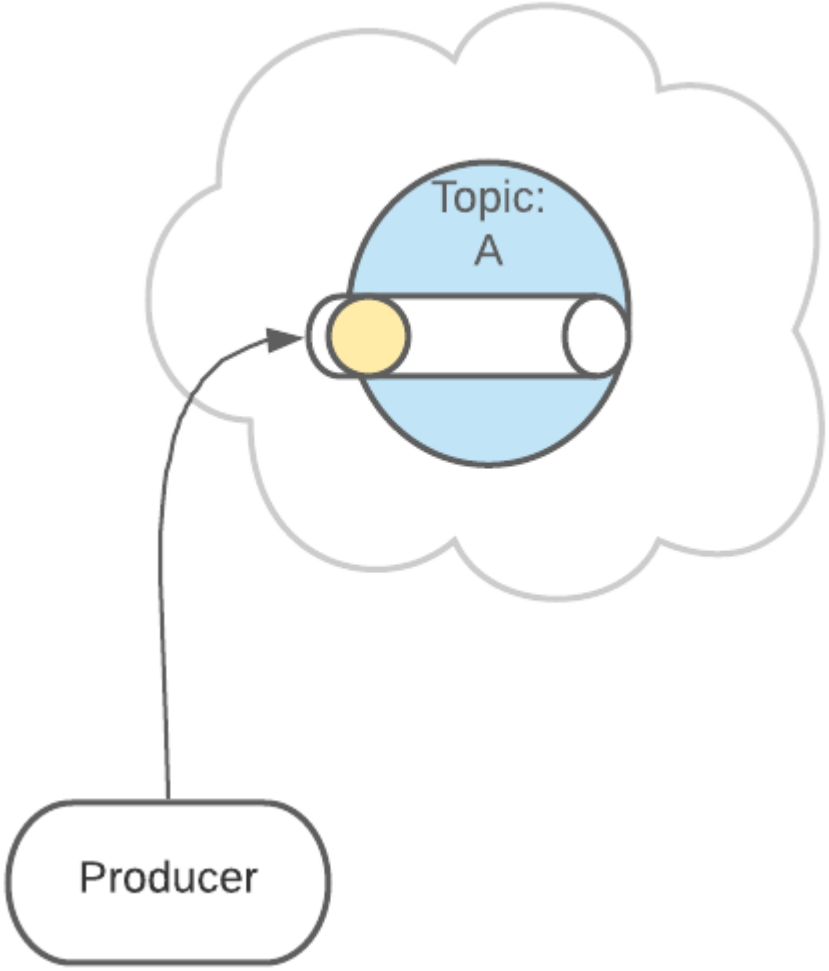
real time

queue

A topic acts as a **queue** for messages.

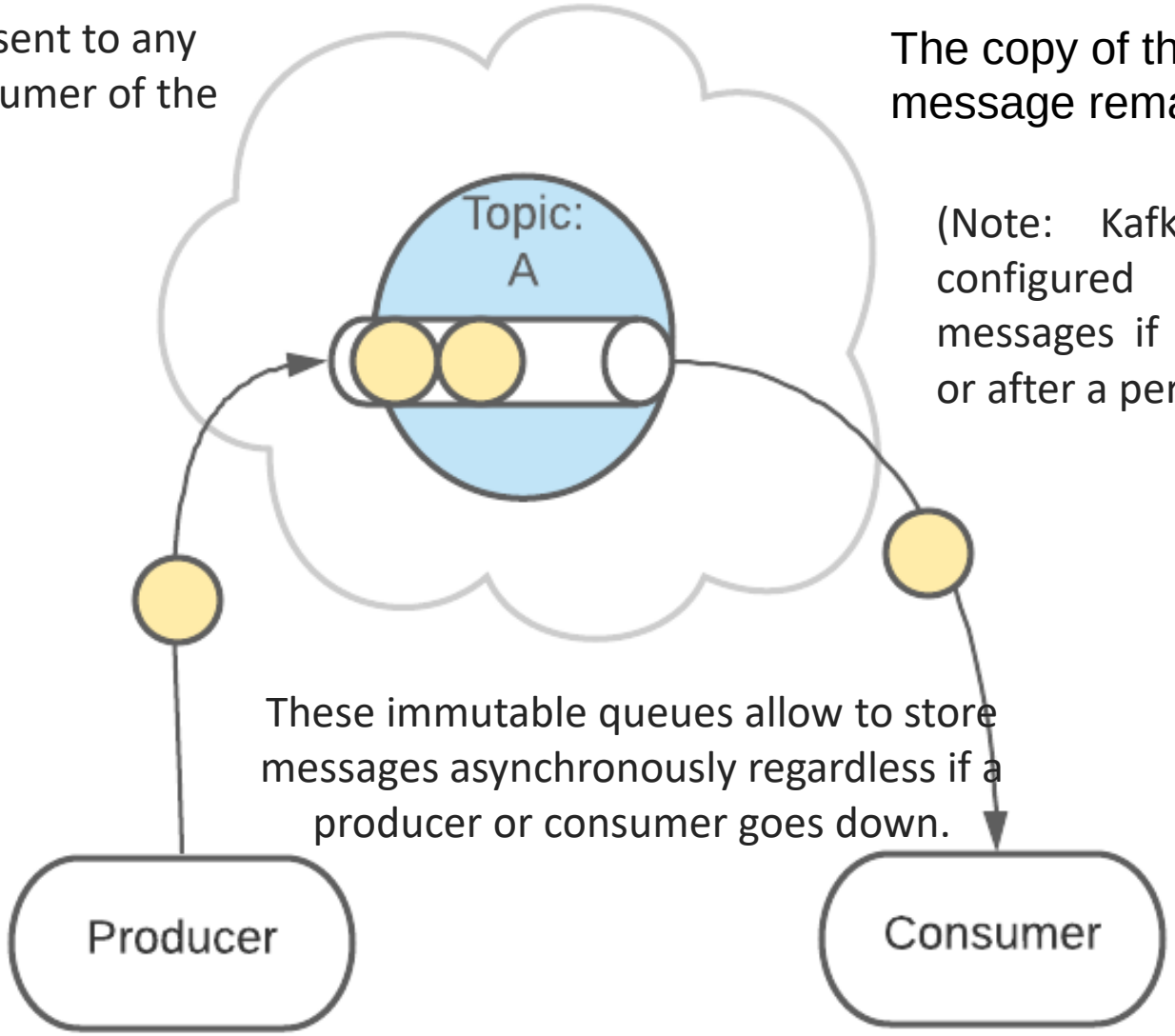


The sent message is recorded and stored in a queue. This message is immutable.



messages in a topic

The message is sent to any subscribed consumer of the topic.



The copy of the immutable message remains on the queue.

(Note: Kafka topics can be configured to remove these messages if there are too many or after a period of time)

These immutable queues allow to store messages asynchronously regardless if a producer or consumer goes down.

real time

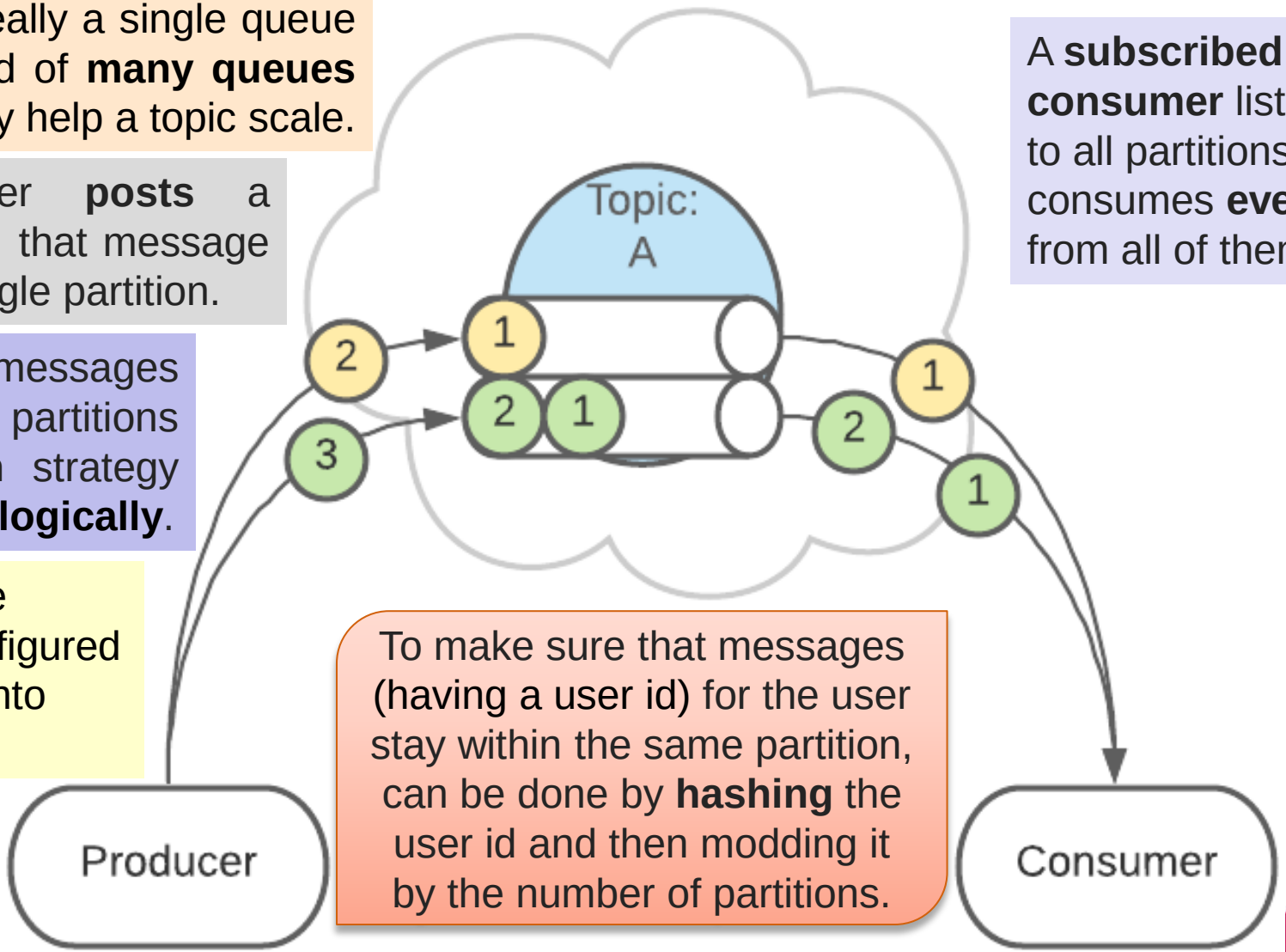
partitions

A Kafka topic is not really a single queue but actually composed of **many queues** called **partitions**. They help a topic scale.

When a producer **posts** a message to a topic, that message gets **routed** to a single partition.

By default, topic messages will be assigned to partitions via a **round robin** strategy and ordered **chronologically**.

Note: topics (not the service) can be configured to **split** messages into different partitions.



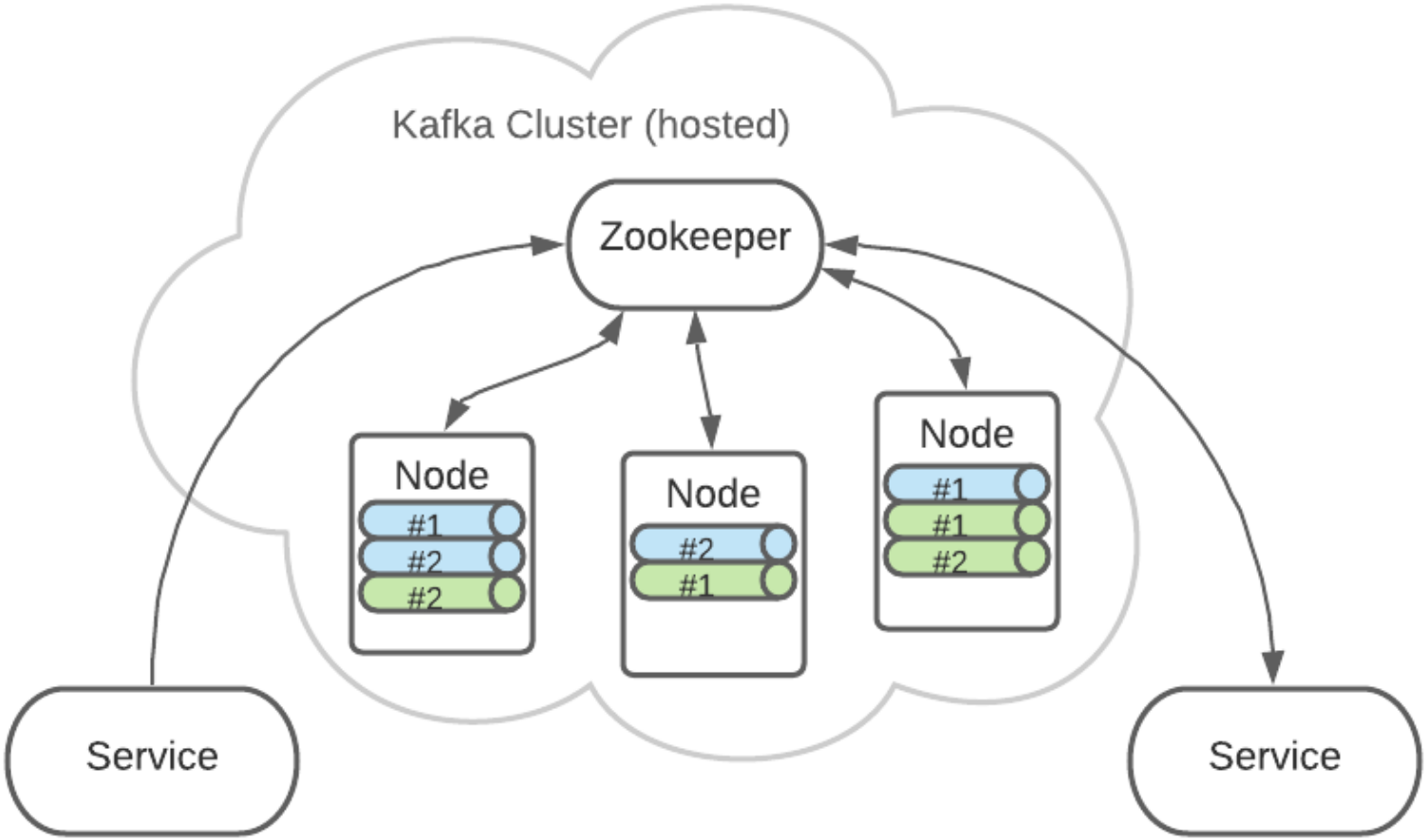
A **subscribed consumer** listens to all partitions and consumes **events** from all of them.

To make sure that messages (having a user id) for the user stay within the same partition, can be done by **hashing** the user id and then modding it by the number of partitions.

real time

infrastructure

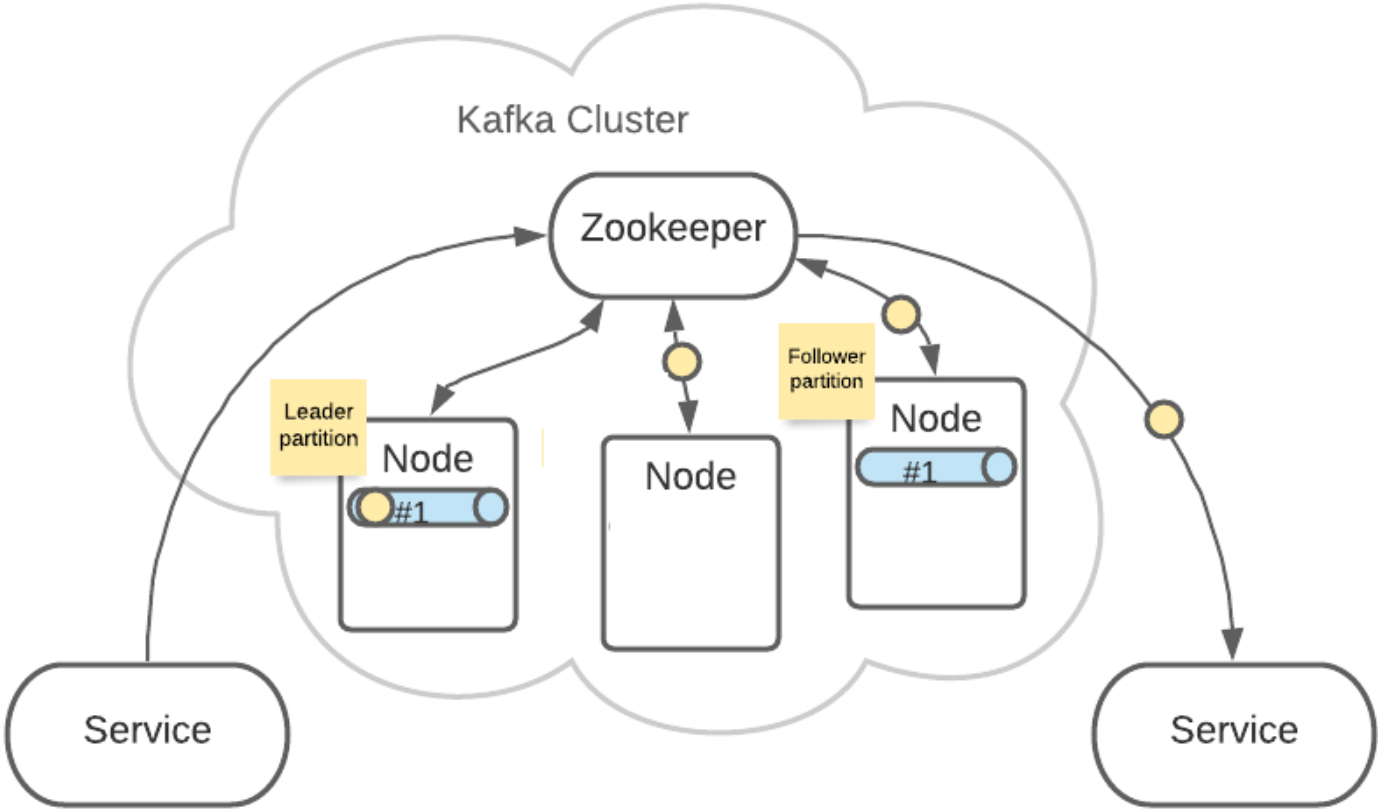
Zookeeper manages all of your topics and partitions.
It basically maintains a set of Kafka cluster nodes where topics and partitions are stored. These nodes are individual machines (for example, EC2 instances) which make up your Kafka cluster.



leader/follower

If a message comes in, it'll get routed to a partition in one of the nodes, known as a leader. Zookeeper assigns the leader.

Zookeeper will send off the message to the consumer just like before.
It will also duplicate the message to the other copies of the partition. Followers.



multiple topics

Assume two messages each of two topics for now. Kafka cluster will maintain all copies in partitions.