



Real-time Data Processing

ผู้ช่วยศาสตราจารย์ ดร.สมเกียรติ โกศลสมบัติ

สาขาวิชาวิทยาศาสตร์และนวัตกรรมข้อมูล

วิทยาลัยสหวิทยาการ มหาวิทยาลัยธรรมศาสตร์

Contents



Introduction to Real-time Data Processing



Characteristics of Real-time Data Processing



Introduction to Apache Kafka



Apache Kafka Architecture



Hand-on



References


Introduction to Real-time Data Processing

- Real-time data processing refers to the continuous and immediate processing of incoming data streams as they arrive.
- This allows businesses and systems to react quickly to events, detect anomalies, and make instant decisions without delays.



A large orange circle is positioned on the left side of the slide, partially cut off by the edge.

Characteristics of Real-time Data Processing

- Low Latency
 - Data is processed within milliseconds or seconds.
 - Continuous Processing
 - No need to wait for a batch; data flows continuously.
 - Scalability
 - Can handle large data streams efficiently.
 - Fault Tolerance
 - Systems remain operational even in case of failures.
- 
- A series of blue brushstrokes are located in the bottom right corner of the slide, arranged in a curved, upward-pointing pattern.

A large orange circle on the left side of the slide, partially cut off by the edge.

Tools and Technology of Real-time Data Processing

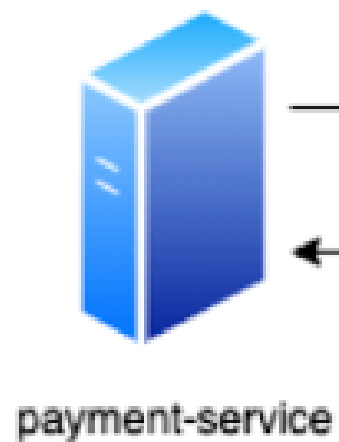
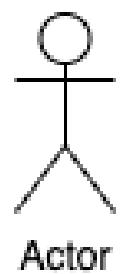
- Apache Kafka
- Apache Flink
- Apache Spark Streaming
- Google Dataflow
- AWS Kinesis

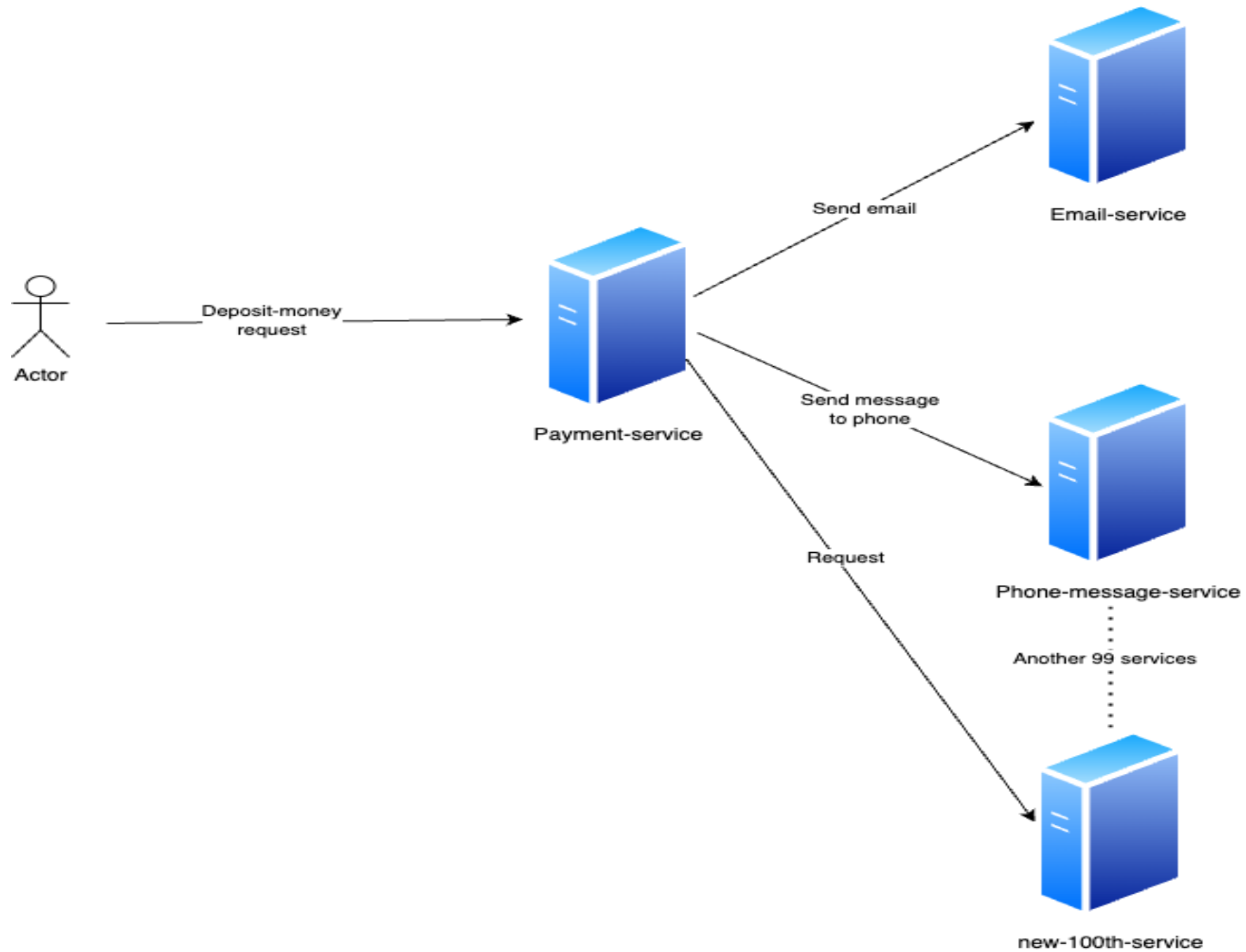


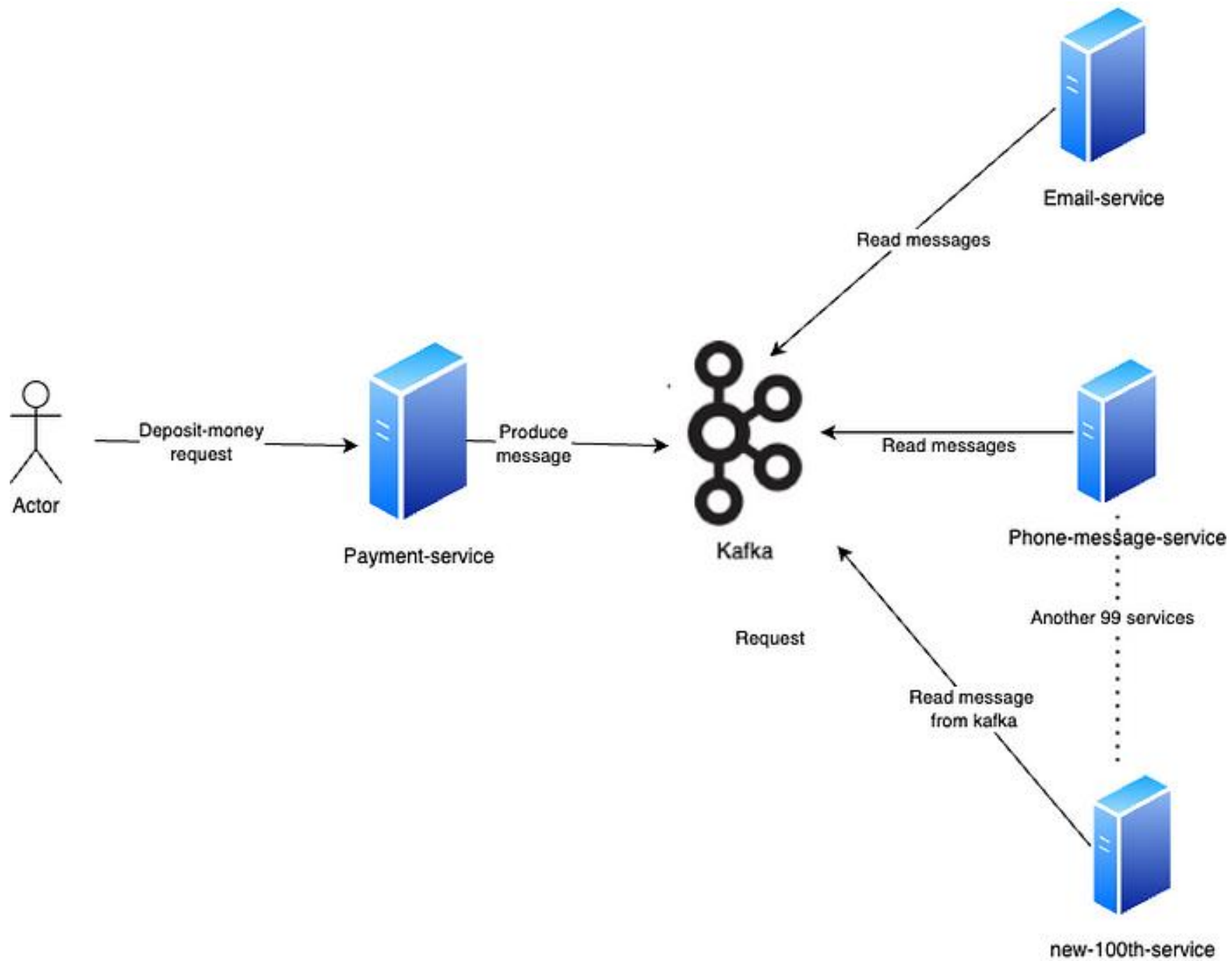


Introduction to Apache Kafka

- Apache Kafka is a distributed event streaming platform designed to handle real-time data feeds at high throughput and low latency.
- It acts as a message broker that enables applications to publish, store, and consume data streams efficiently.
- Kafka is widely used for log collection, event-driven architectures, real-time analytics, and big data processing due to its scalability, durability, and fault tolerance.







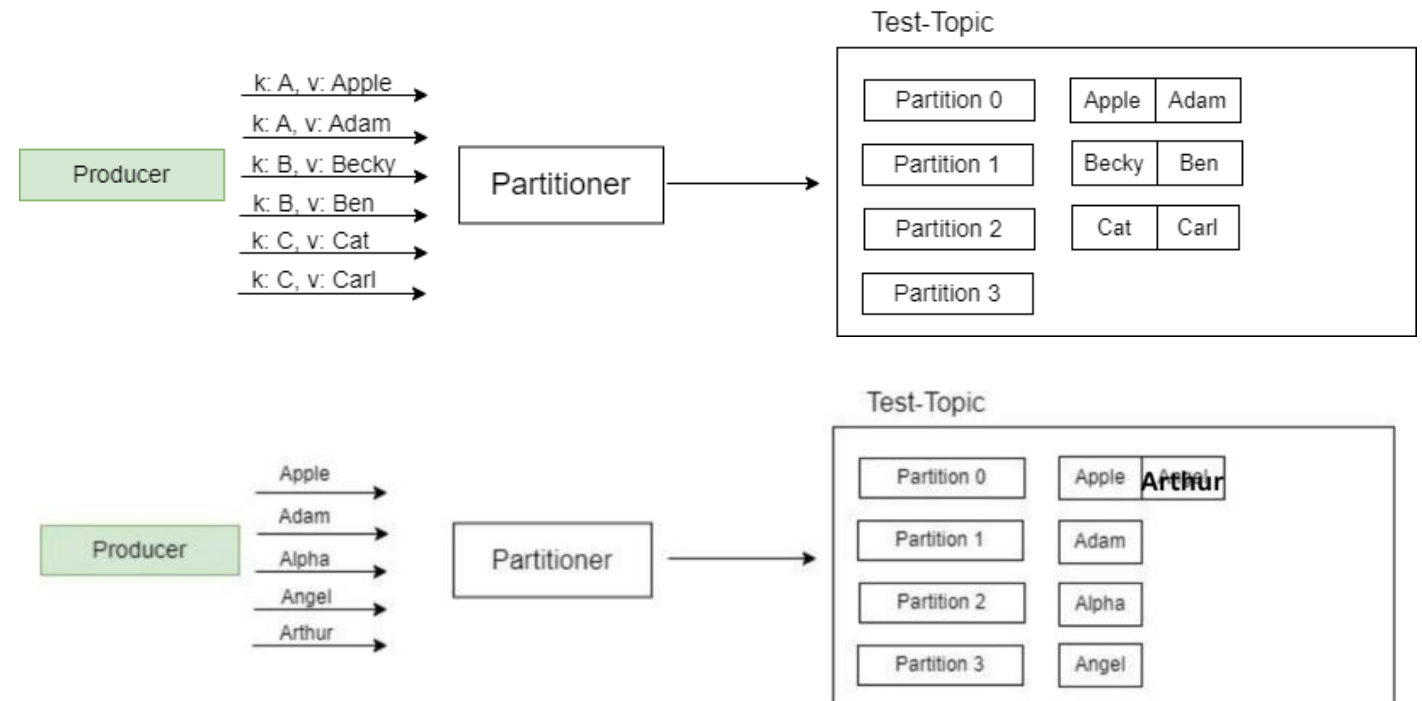


Kafka Architecture

- Producer
- Kafka Broker
- Consumer
- Customer Group
- Consumer Offset
- Bootstrap Server
- Message
- Topic
- Partitions
- Topic Replication
- Zookeeper (Kafka Controller)

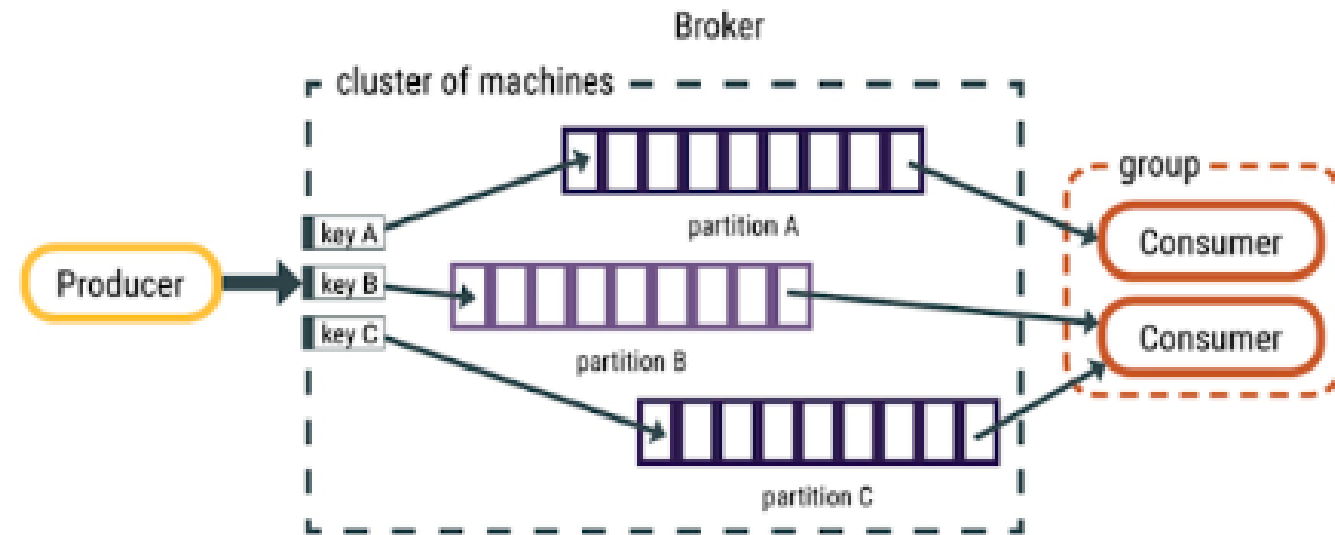
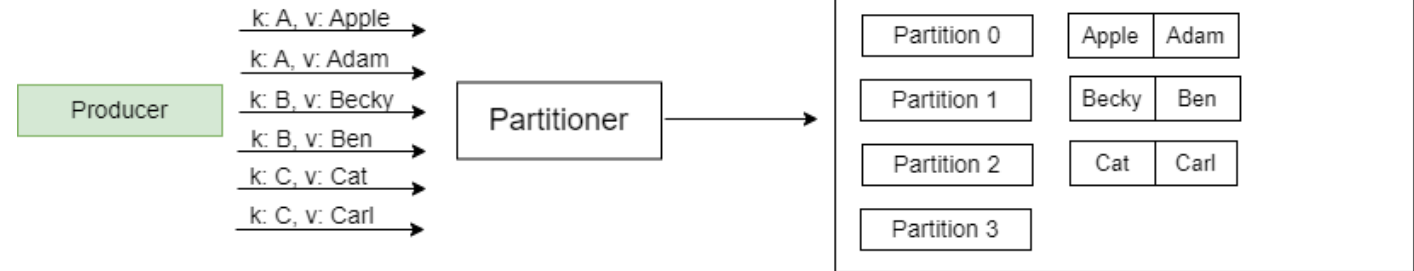
Kafka Architecture

- Producer
 - A producer is a client application that publishes messages to a Kafka topic.
 - The producer is responsible for creating messages, serializing them into a specific format, and sending them to Kafka brokers.



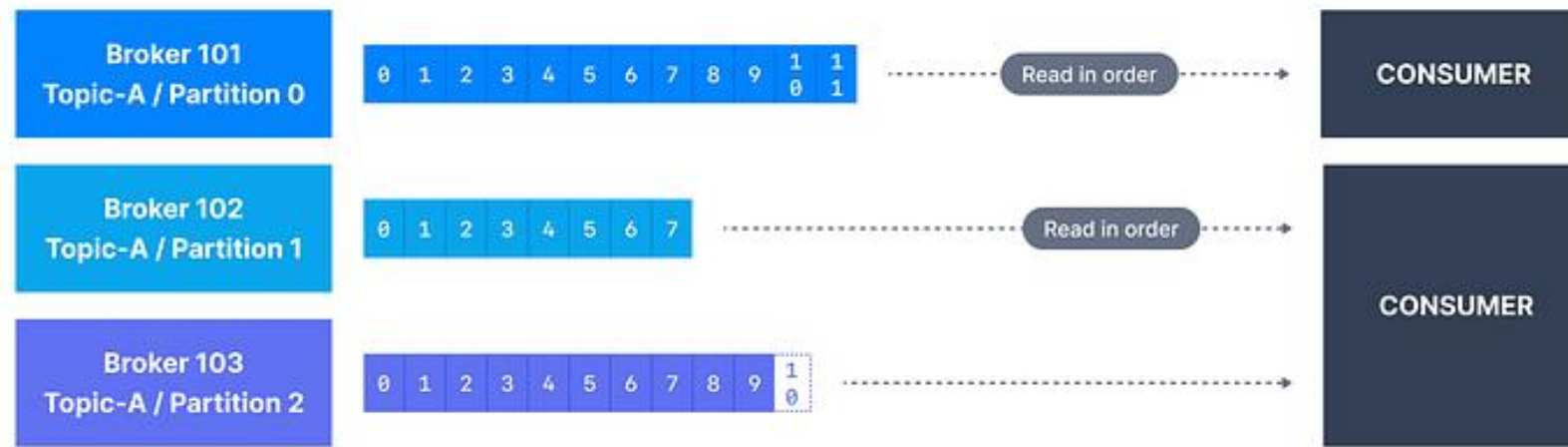
Kafka Architecture

- Kafka Broker
 - Brokers act as intermediaries between producers and consumers.
 - They facilitate message exchange, ensuring messages reach their intended recipients.



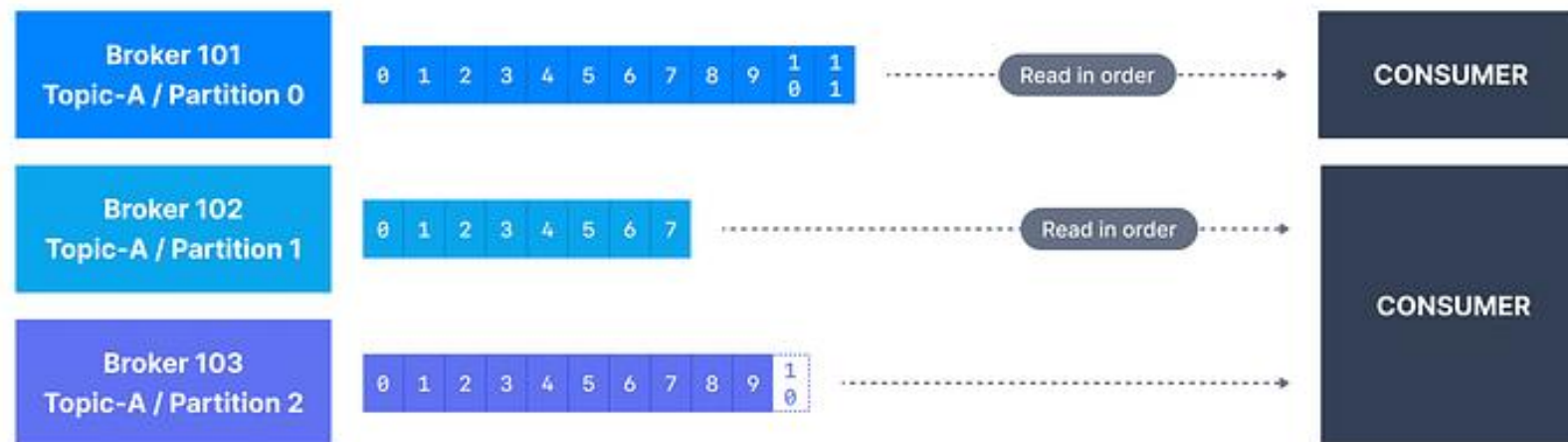
Kafka Architecture

- Consumer
 - A consumer is a client application that reads messages from one or more Kafka topics.
 - The consumer subscribes to one or more topics, specifies the offset from which to start reading messages, and receives messages in the order they were produced.



Kafka Architecture

- Consumer Groups
 - A consumer group consists of one or more consumers that work together to consume a topic.
 - Consumer groups allow for distributed processing, where each consumer in the group reads from a unique set of partitions of a topic, thereby increasing throughput.
 - Kafka ensures that each partition is only consumed by one consumer in the group at any time.



Partition 0: [msg_0][msg_1][msg_2][msg_3]
Consumer A offset: 2 → หมายถึงอ่านถึง msg_1 แล้ว กำลังจะอ่าน msg_2

Consumer A offset: 2 → หมายถึงอ่านถึง msg 1 แล้ว กำลังจะอ่าน msg 2

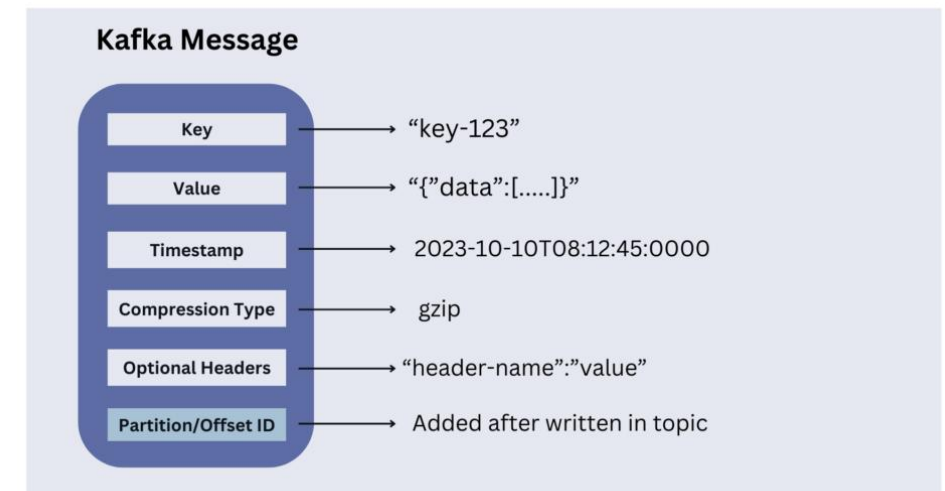
Kafka Architecture

- Bootstrap Server
 - The initial connection point used by Kafka clients to connect to the cluster.
 - Client (Hostname/IP + Port) -> Cluster (Broker)

```
bootstrap.servers=broker1:9092,broker2:9092,broker3:9092
```

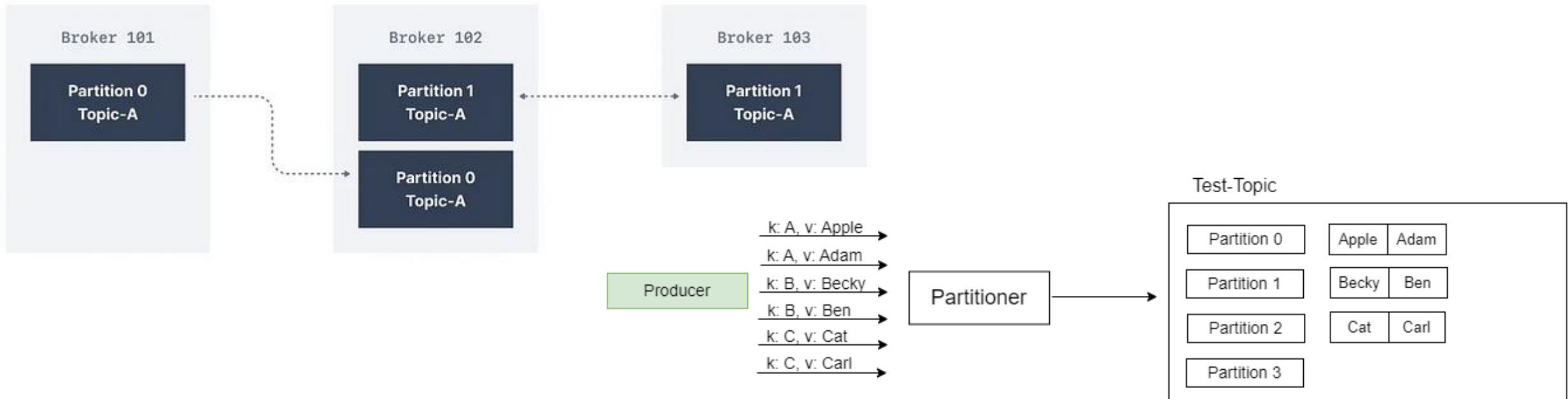

Kafka Architecture

- Message
 - A message in Kafka is the basic unit of data transmitted over the system.
 - It consists of a key, a value, and optional headers.
 - The key is used by Kafka to determine the partition within a topic where the message will be sent (if partition is not explicitly specified by the producer), while the value is the actual data payload of the message.
 - Headers can include metadata about the message.
 - In case the key (key=null) is not specified by the producer, messages are distributed evenly across partitions in a topic in a round-robin fashion.



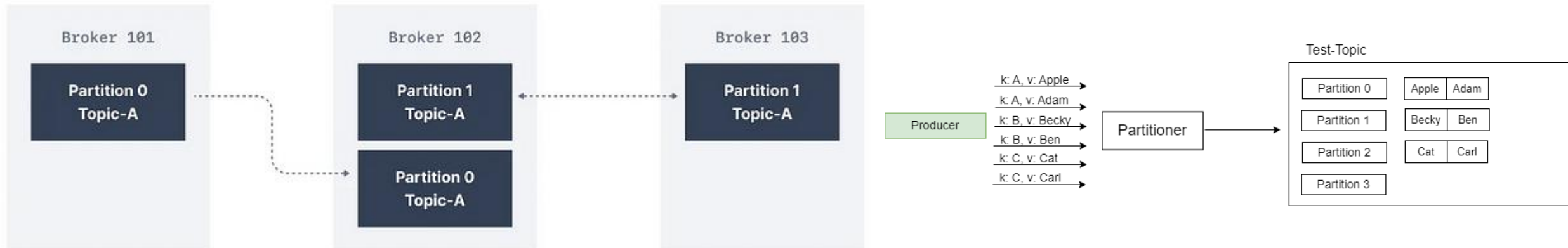
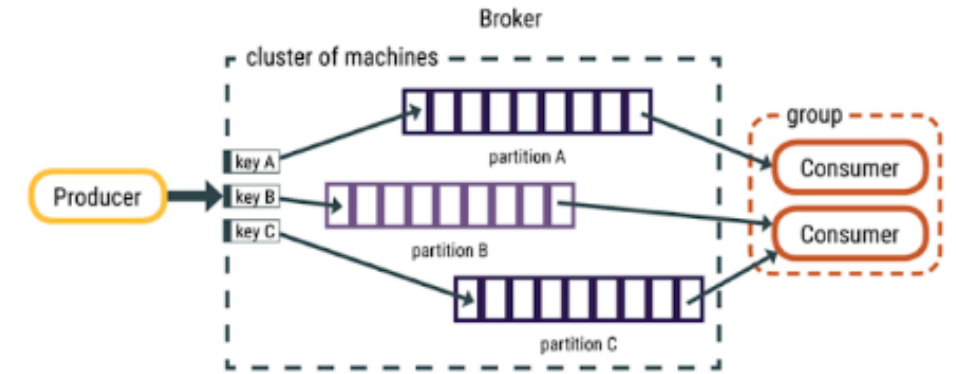
Kafka Architecture

- Topic
 - A topic in Kafka represents the logical group of messages.
 - The producer will send messages to the Kafka topic and the consumer will read messages from the Kafka topic.



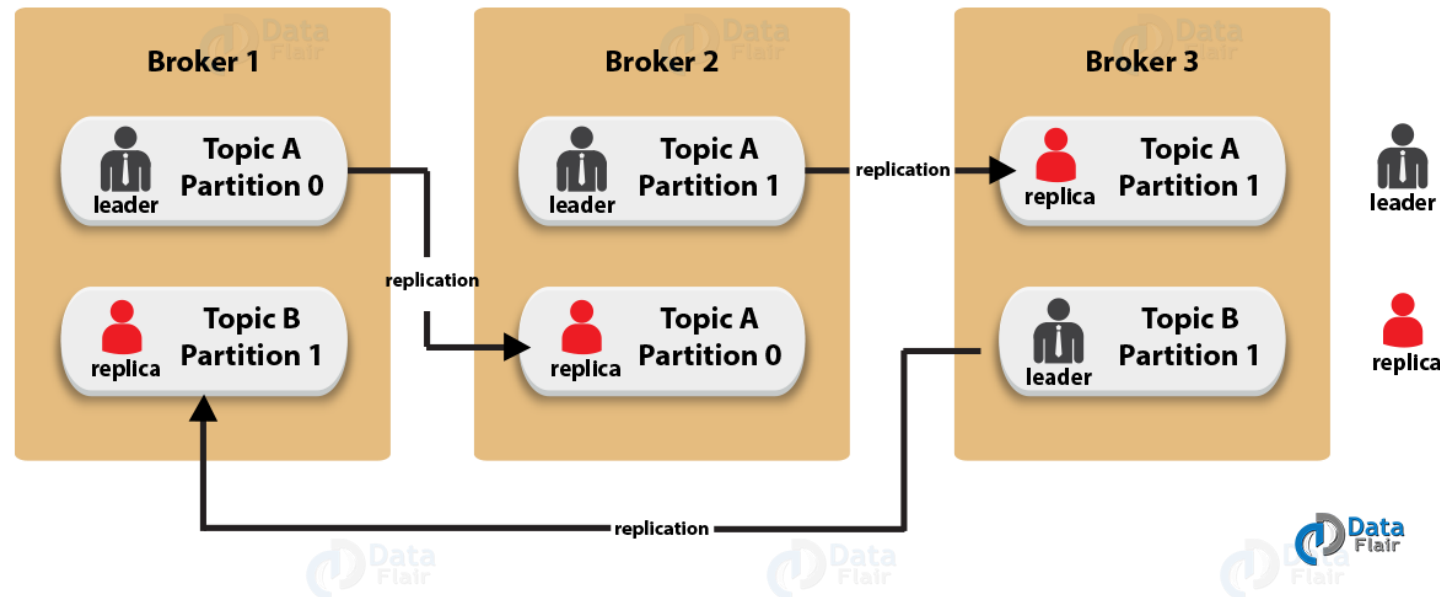
Kafka Architecture

- Partitions
 - Partitions take the single topic log and break it into multiple logs.
 - Each partition is an ordered, immutable sequence of messages that are stored on disk and replicated across multiple Kafka brokers for fault tolerance and high availability.
 - Messages within a partition are ordered based on their offset value, which is a unique identifier for each message within the partition.



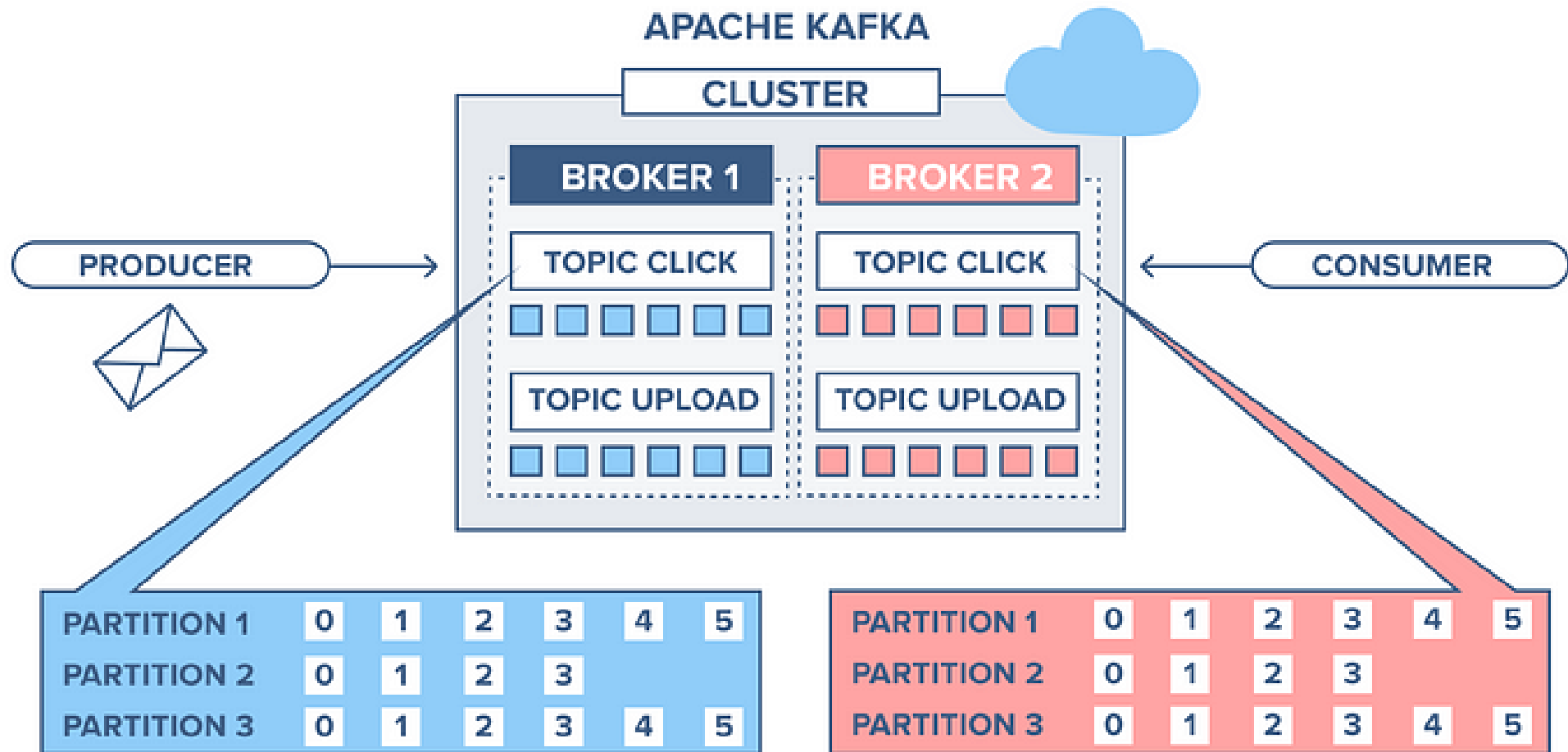
Kafka Architecture

- Kafka Topic Replication
 - Topic replication in Kafka involves creating copies (replicas) of a topic's partitions across multiple brokers in a Kafka cluster.
 - This ensures that even if a broker goes down, the data is not lost and can still be accessed from other brokers that have copies of the same partitions.



Kafka Architecture

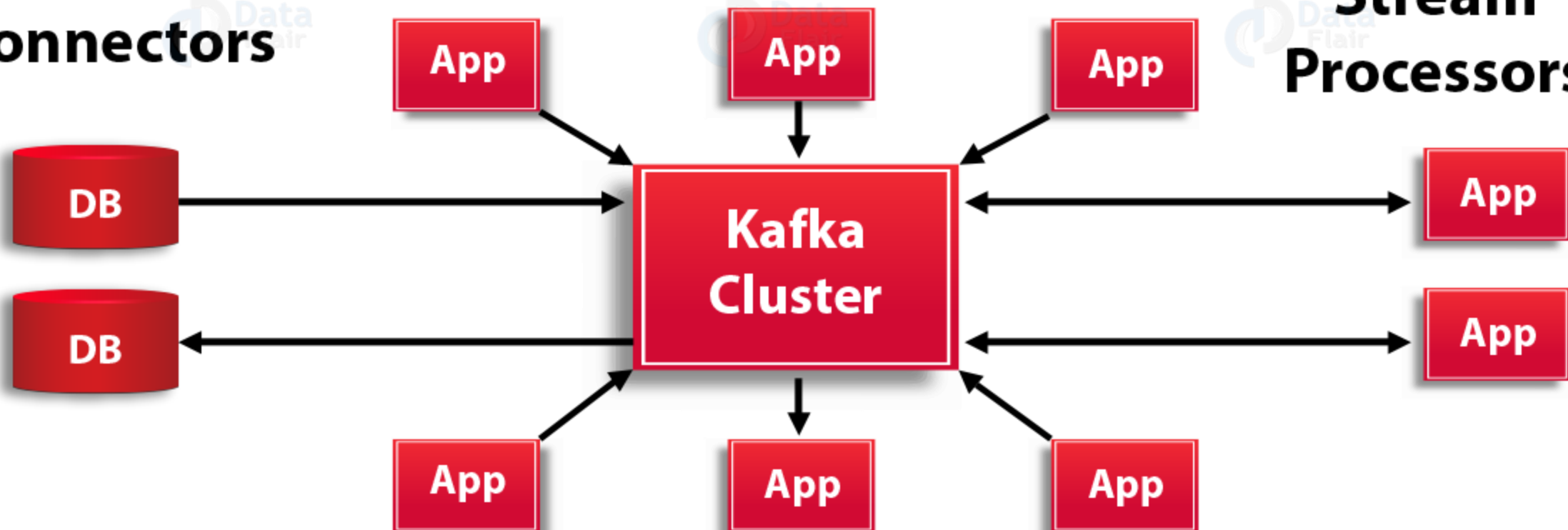
- Zookeeper (Kafka Controller)
 - A zookeeper is a crucial component in the Kafka ecosystem.
 - Kafka is a distributed system, and it relies on Zookeeper for coordination and tracking of the status of Kafka cluster nodes.
 - Zookeeper also keeps track of Kafka topics, partitions, offsets, and more.
 - In essence, Zookeeper serves as the manager for your Kafka cluster and brokers.



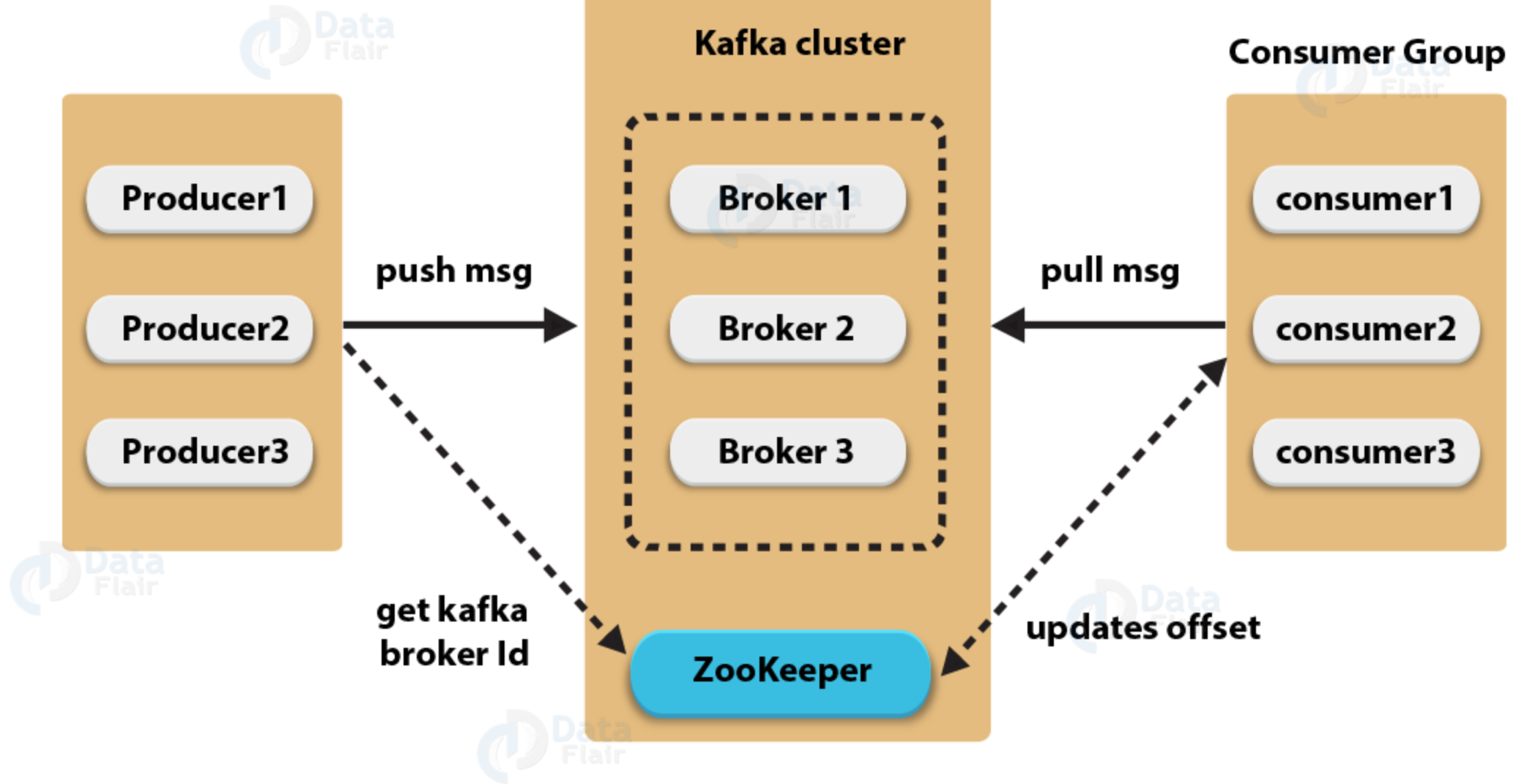
Producers

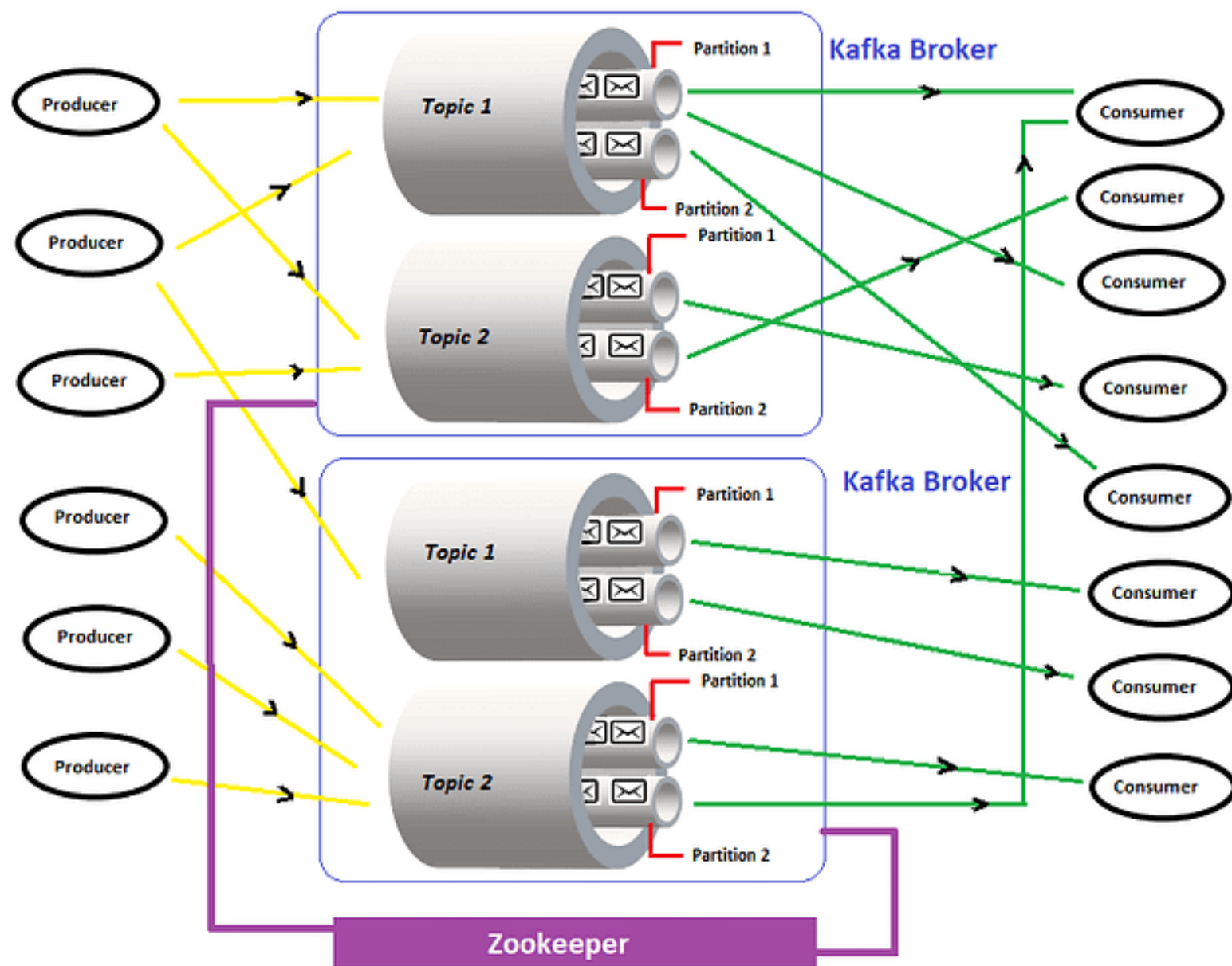
Stream Processors

Connectors



Consumers





Kafka Installation and Setup

- ดาวน์โหลด Kafka
- `$ wget https://downloads.apache.org/kafka/3.9.0/kafka_2.13-3.9.0.tgz`
- แยกไฟล์
- `$ tar -xvzf kafka_2.13-3.9.0.tgz`
- `$ cd kafka_2.13-3.9.0`

Kafka Installation and Setup

- Start Zookeeper
- `$ bin/zookeeper-server-start.sh config/zookeeper.properties &`
- Start Kafka Broker
- `$ bin/kafka-server-start.sh config/server.properties &`
- `$ bin/kafka-topics.sh --create --topic my-topic --bootstrap-server localhost:9092 --partitions 3 --replication-factor 1`

Kafka Installation and Setup

- `$ bin/kafka-console-producer.sh --topic my-topic --bootstrap-server localhost:9092`

>Message.....

>Message.....

----- อาจจะเปลี่ยนไปอีกเครื่อง -----

- `$ bin/kafka-console-consumer.sh --topic my-topic --from-beginning --bootstrap-server localhost:9092`

References

- <https://kafka.apache.org/downloads>
- <https://www.virtualbox.org/wiki/Downloads>
- <https://medium.com/@patelharshali136/apache-kafka-tutorial-kafka-for-beginners-a58140cef84f>
- <https://data-flair.training/blogs/kafka-architecture/>
- <https://medium.com/@pgulshetty/kafka-basics-a-comprehensive-guide-for-beginners-e2a051e045e5>
- <https://vishwas-agarwal.medium.com/where-data-flows-insights-grow-part-1-apache-kafka-from-basics-to-advanced-807cafb900d8>
- <https://medium.com/@bhageshwaridevnani/understanding-apache-kafka-a-beginners-guide-340ab3d1ebd0>
- <https://howtodoinjava.com/kafka/apache-kafka-tutorial/>
<https://howtodoinjava.com/kafka/apache-kafka-tutorial/>