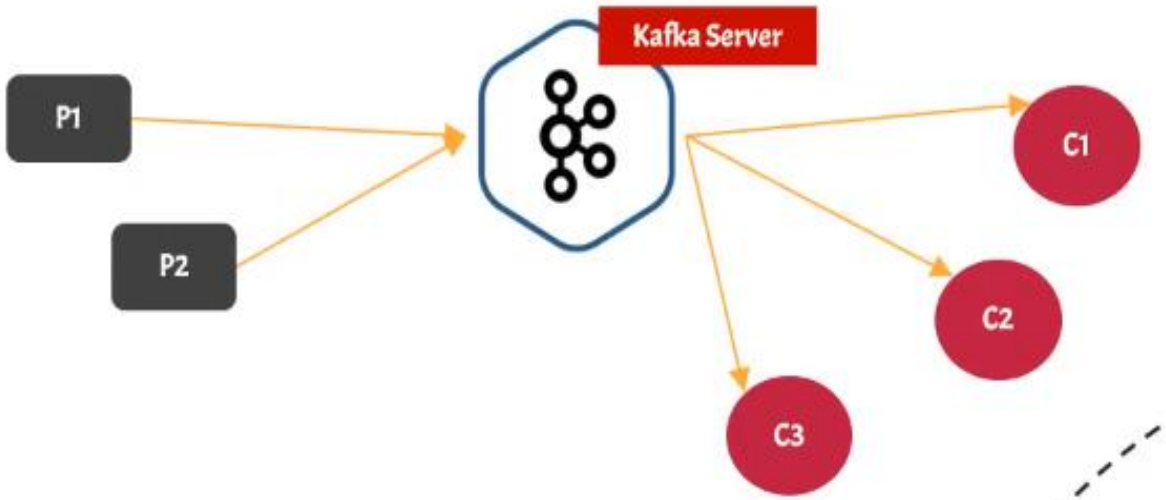
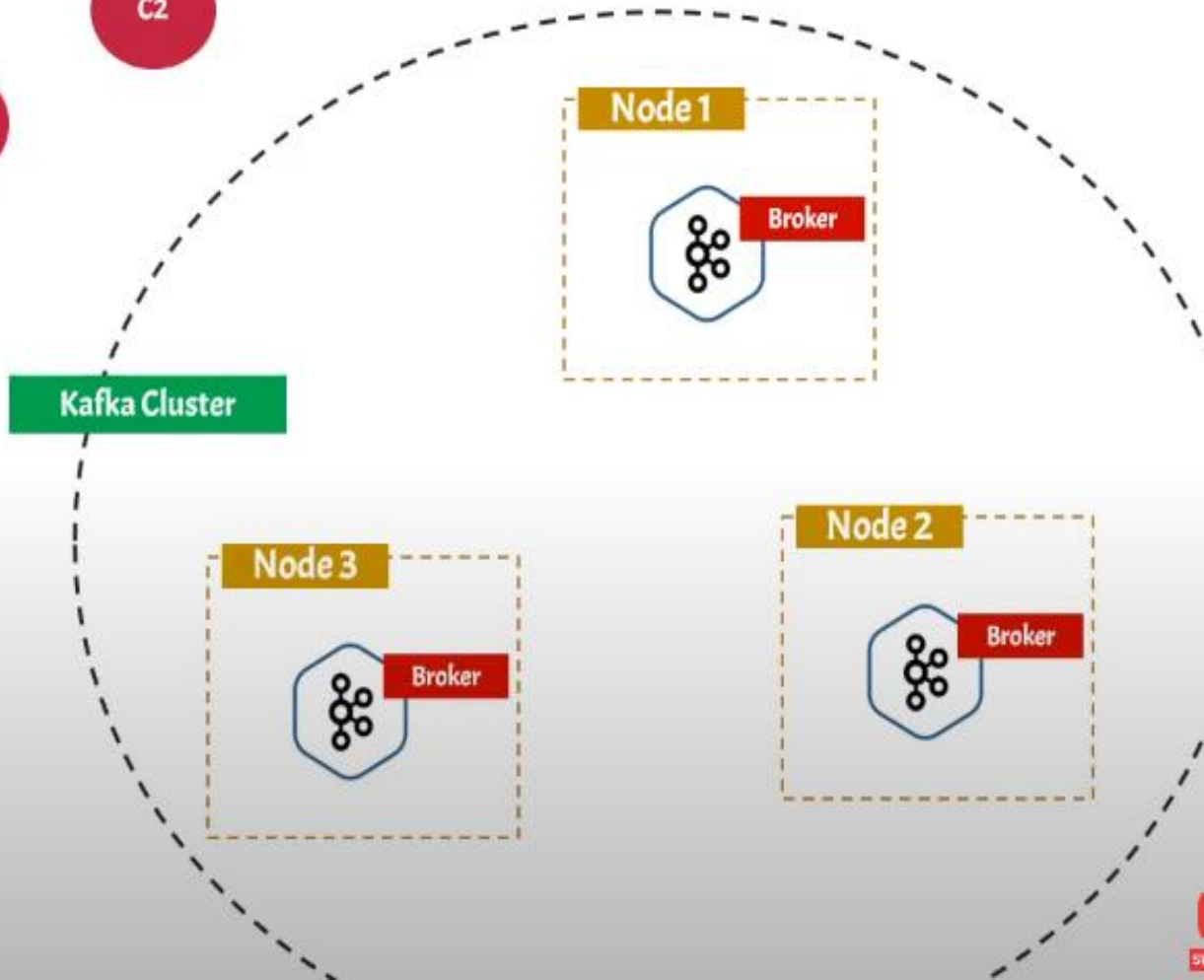
- Kafka is just like a messaging system

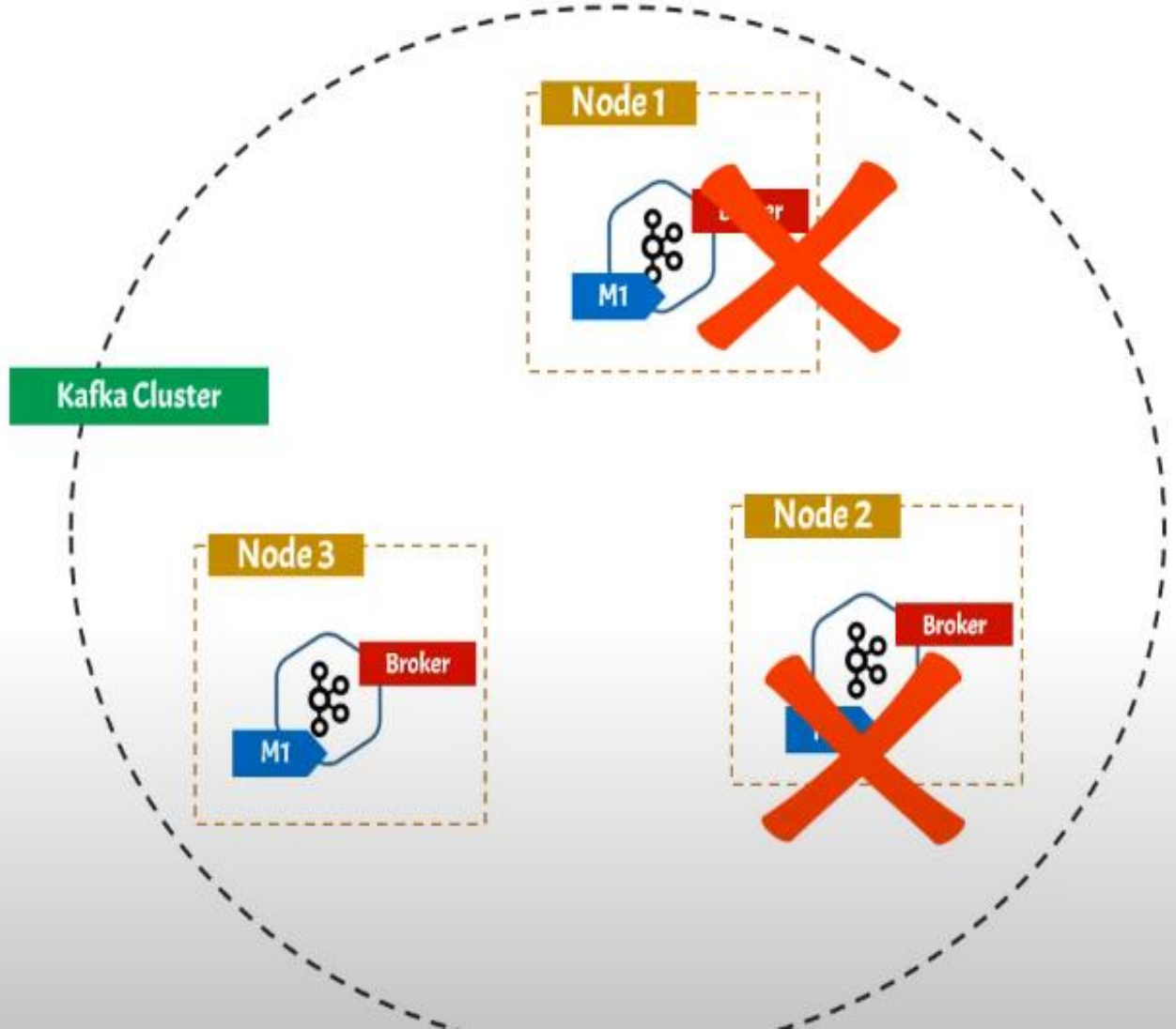


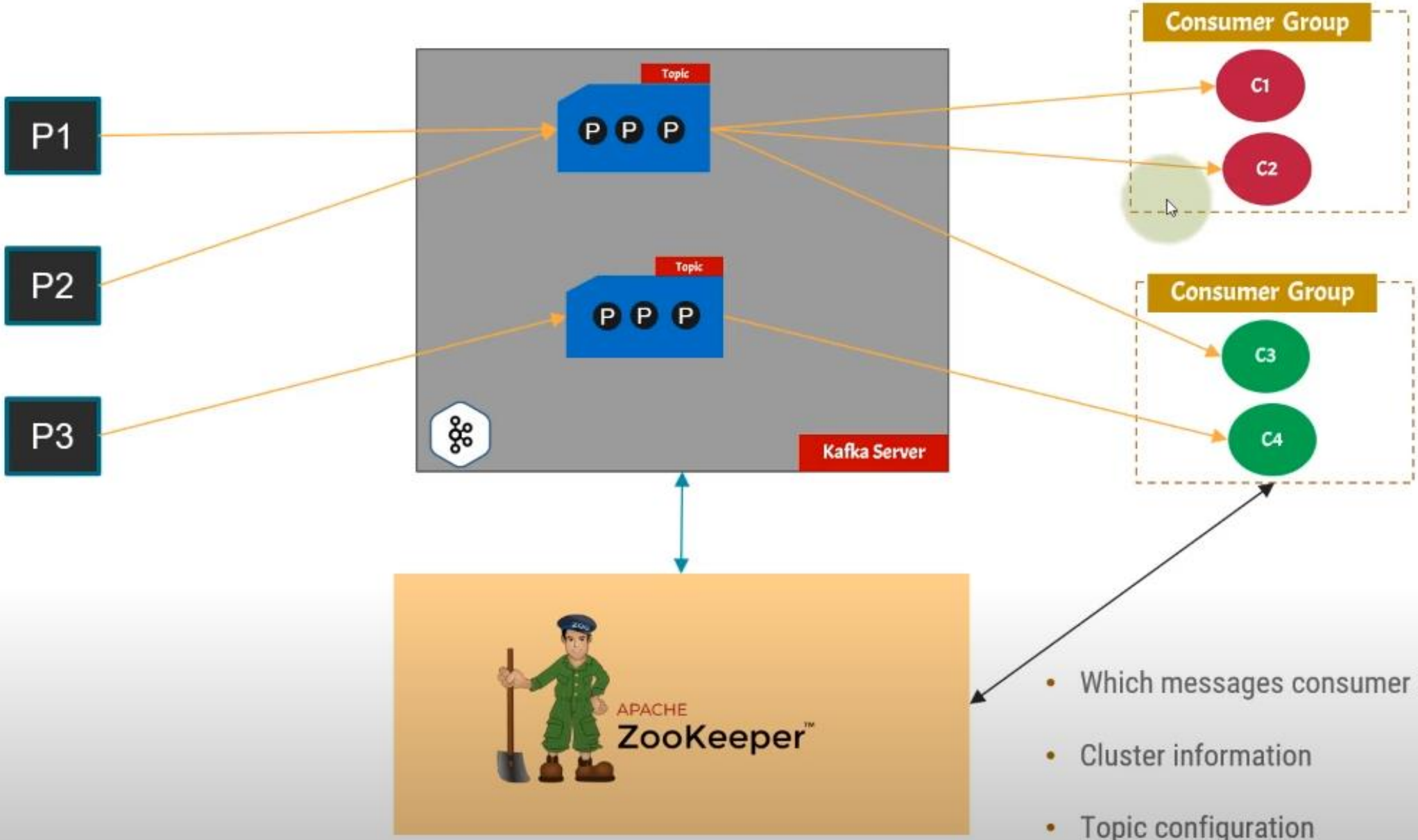
- It is distributed platform / application
  - In production environment Kafka is referred as Kafka Cluster.
  - A cluster is made up of more than one Kafka server
  - Each Kafka server is referred as Broker



- Kafka is fault-tolerant
  - Ability of a system to continue operating without interruption when one or more of its components fail
  - In Kafka cluster messages are replicated in multiple broker
  - Replication Factor

- Kafka is Scalable system
  - You can new brokers
  - You can increase the number of consumers

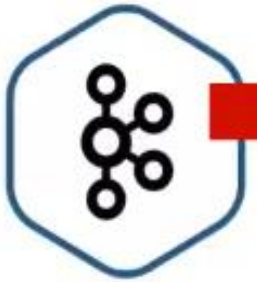




- Which messages consumer has read
- Cluster information
- Topic configuration

- Zookeeper is a distributed, open-source configuration, synchronization service

- Download the software
- You will find both zookeeper and Kafka server

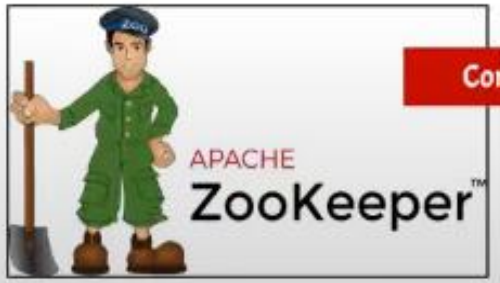


Conf file = server.properties

```
advertised.listeners=PLAINTEXT://<server-ip-address>:9092
```

```
zookeeper.connect=localhost:2181
```

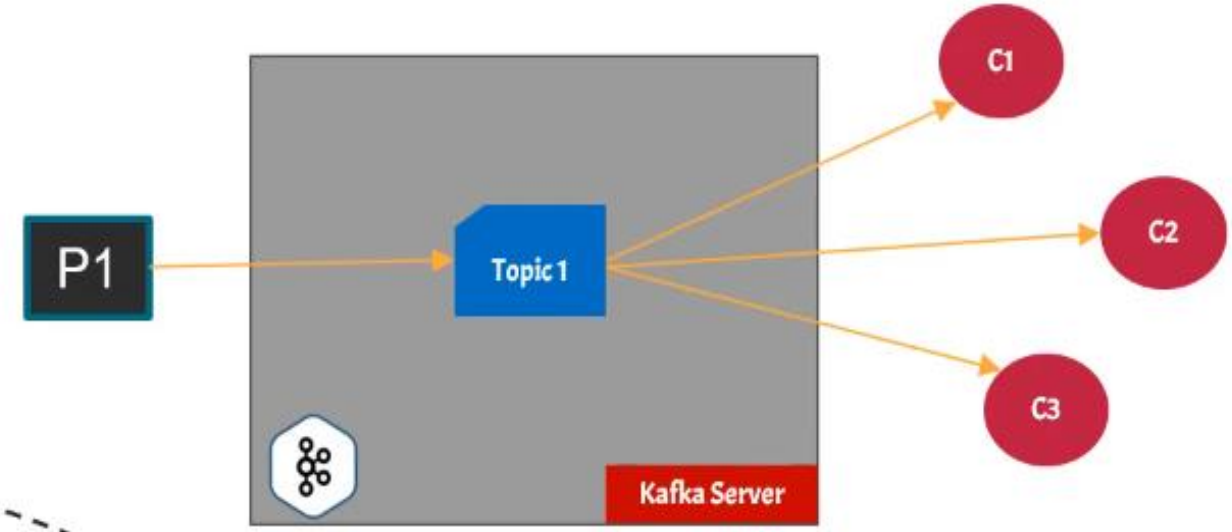
```
JMX_PORT=8004 bin/kafka-server-start.sh config/server.properties
```

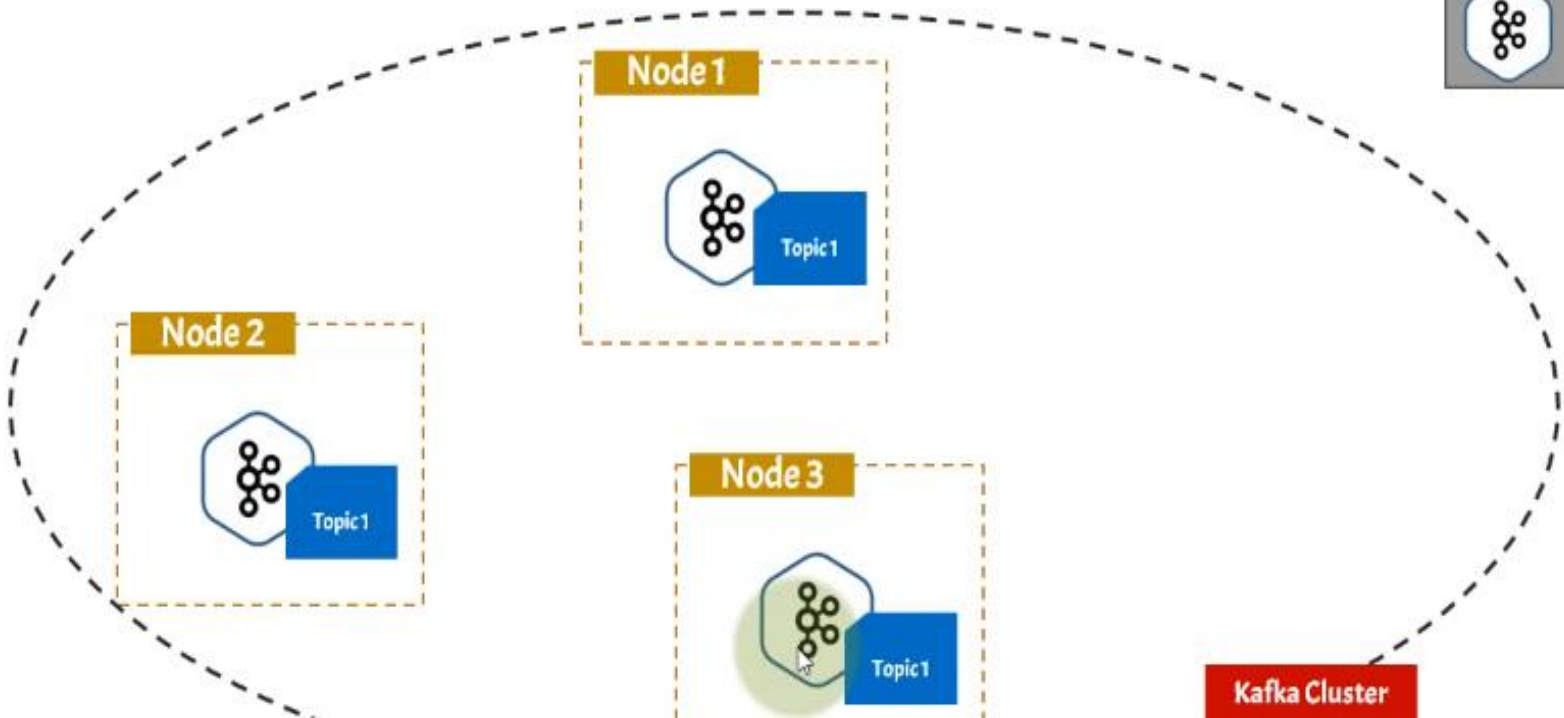


Conf file = zookeeper.properties

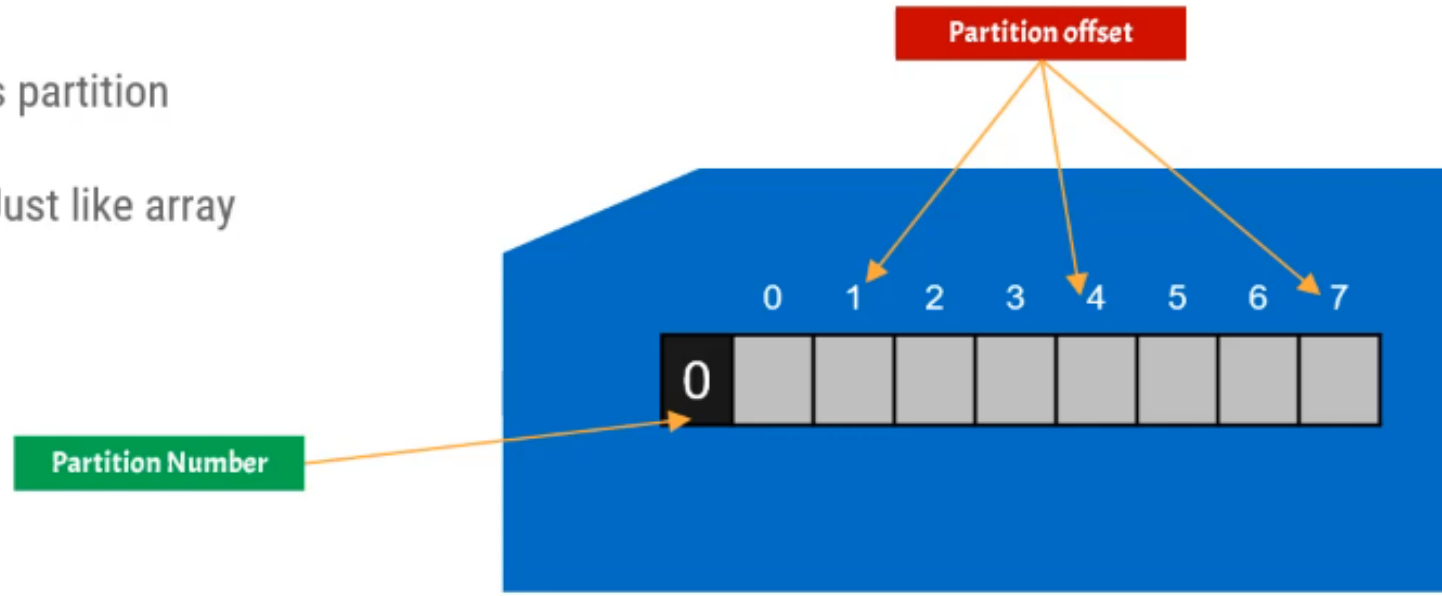
```
bin/zookeeper-server-start.sh config/zookeeper.properties
```

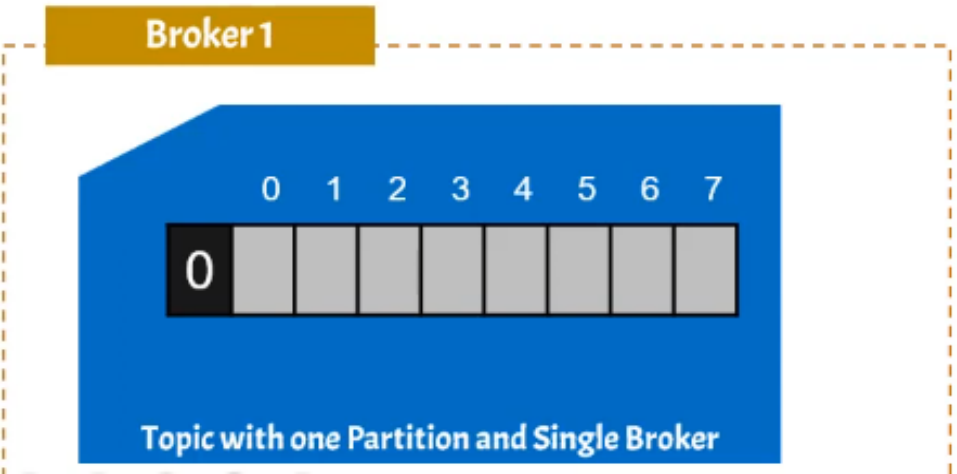
- Topic is the Kafka component where Producers are connected
- Producer publish message in Kafka Topic
- Topics in Kafka is multi subscriber.
- Topic can be considered as a logical entity
- In Kafka cluster each topic is present in every cluster node

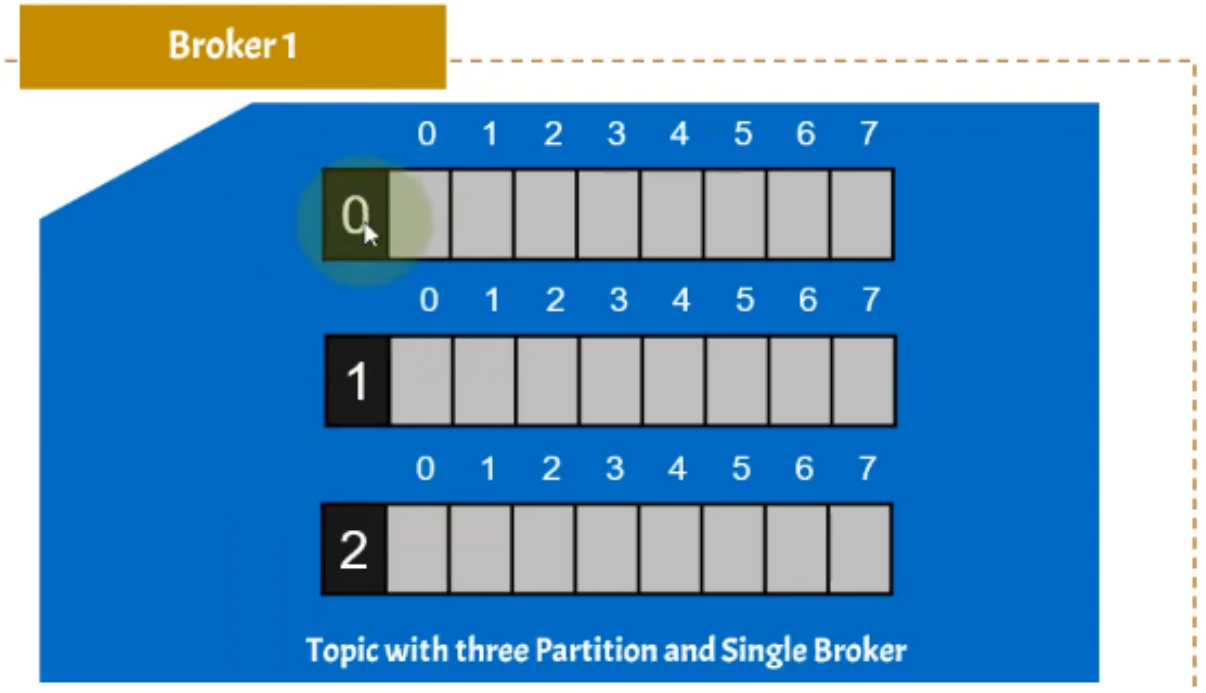




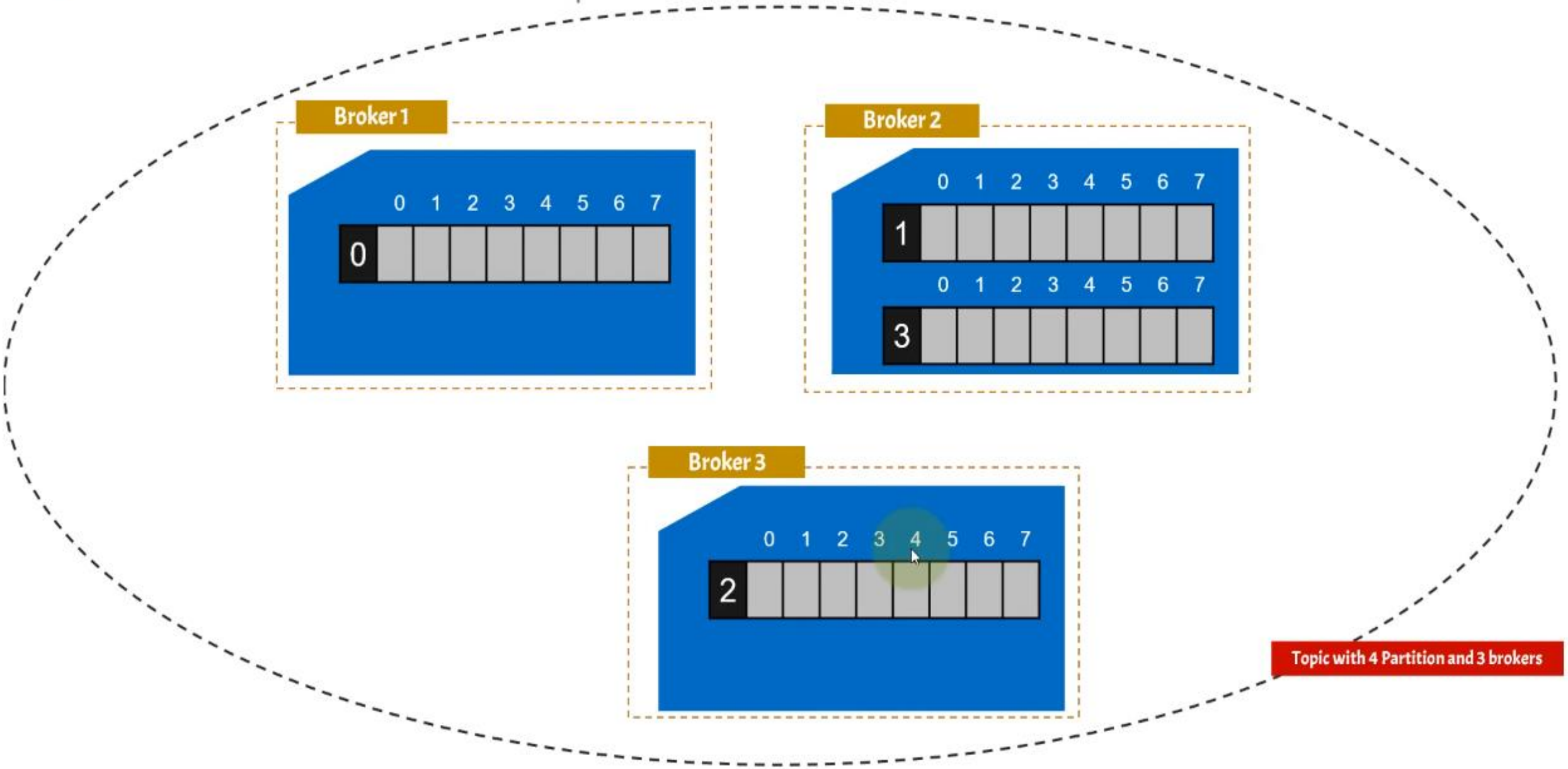
- A Kafka topic is divided into multiple parts that is called as partition
- Partitions can be considered as the linear data structure. Just like array
- Messages are actually published to a partition in the topic
- Every partition has a partition number
- Each partition has increasing index called offset
- New messages are always pushed at the read end
- Data is immutable after publish



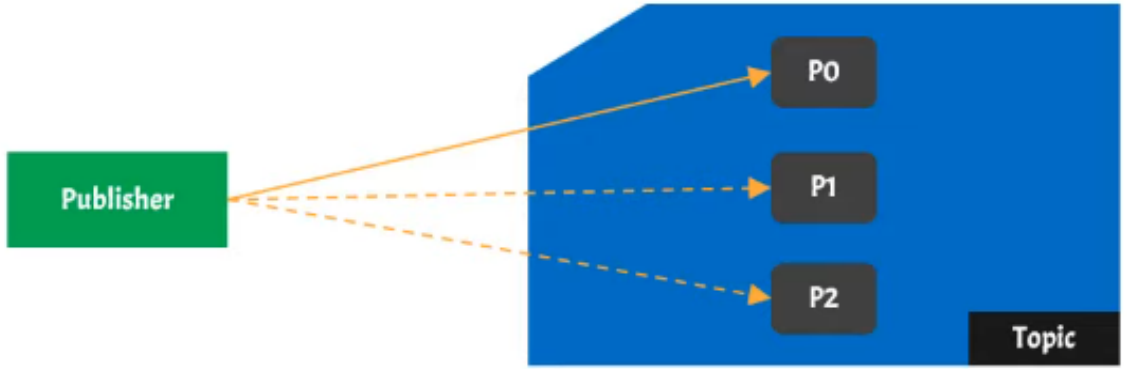




- In multi broker Kafka Cluster Partitions for a topic are distributed across the whole cluster



- Producers publish message to the topics of their choice
- In reality messages are published to topic partition




```
pip install kafka-python
```

```
pip install Faker
```

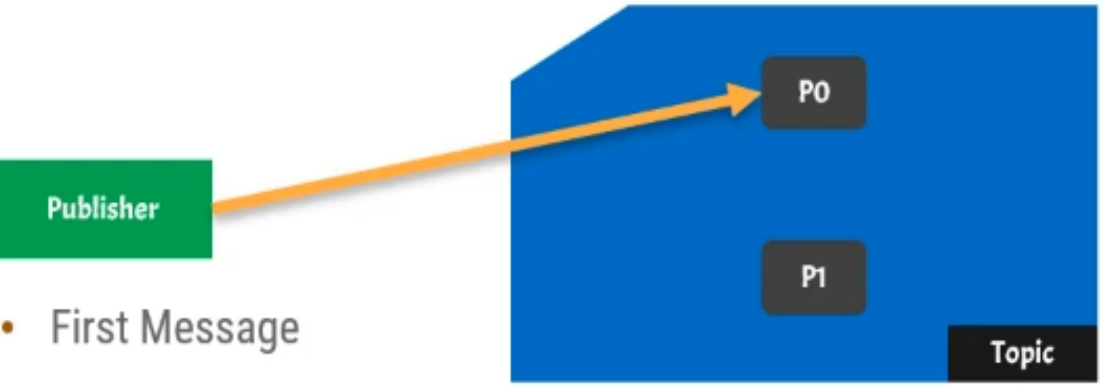
Configuration Needed by Producer

- bootstrap_servers
- topic
- value_serializer

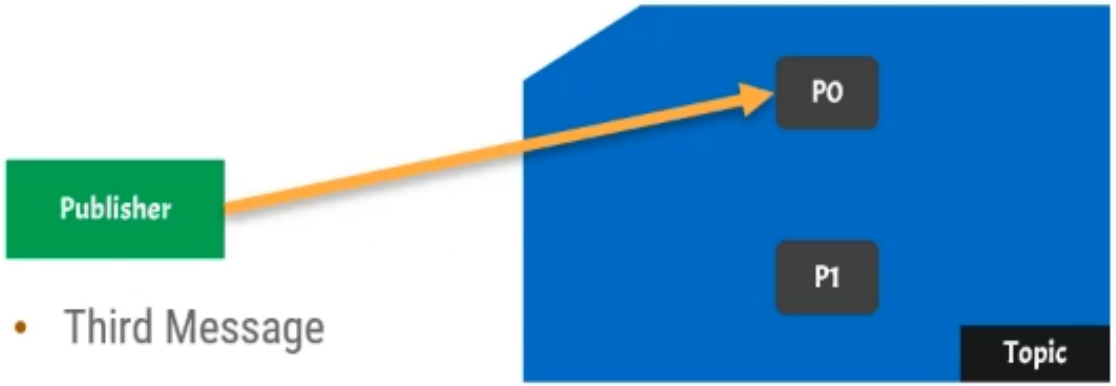
- **send** method is called on producer to publish the data

Example : Single Broker – 2 Partition

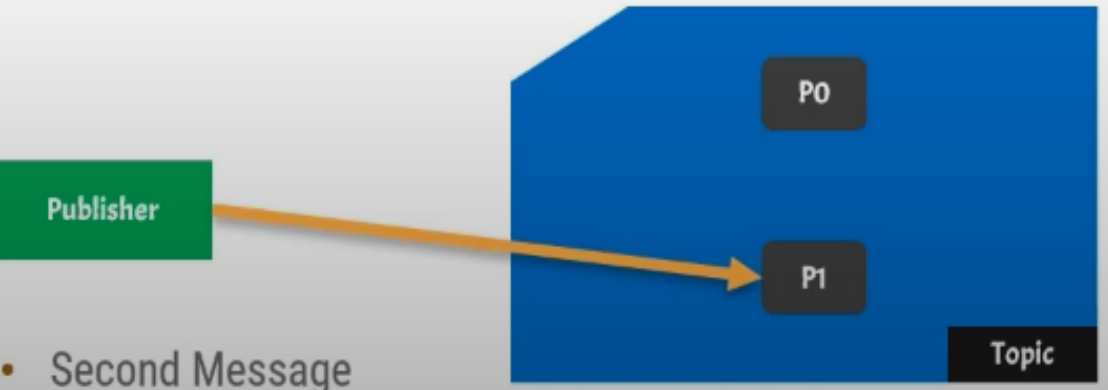
- Message will be published to both partition.
- Message publish is random



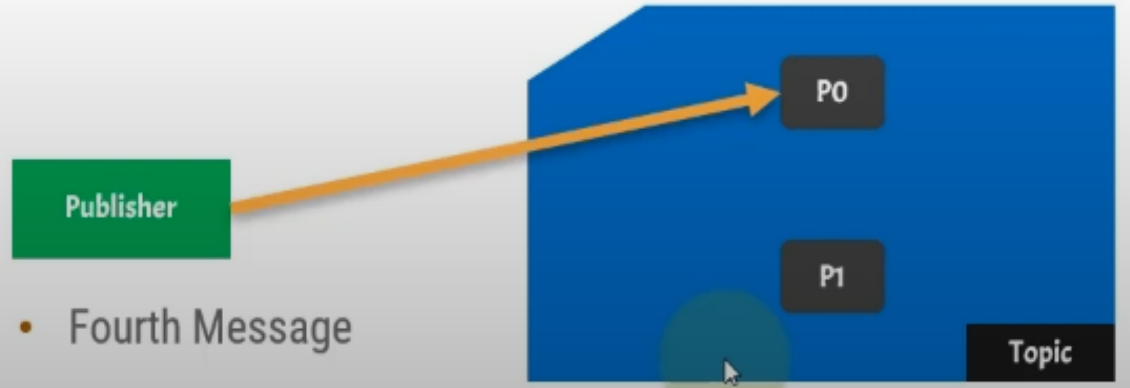
- First Message



- Third Message



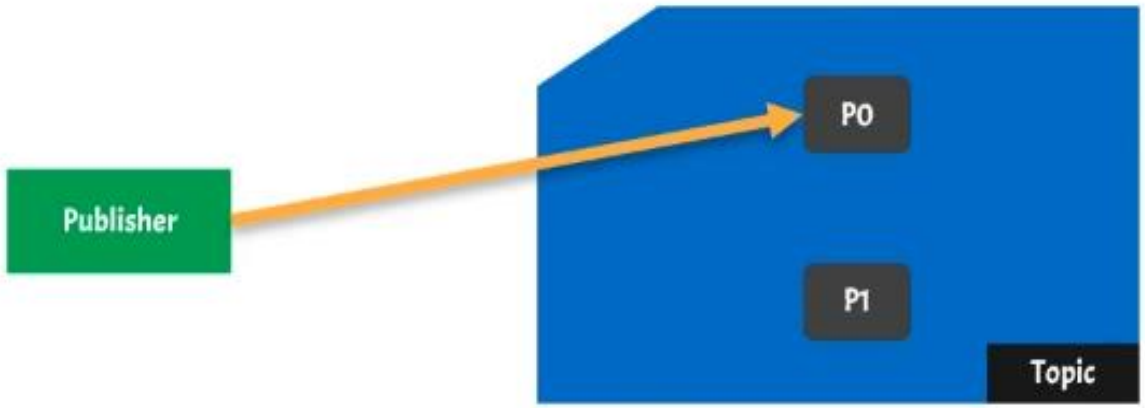
- Second Message



- Fourth Message

Example : Single Broker – 2 Partition. Send message to only one Partition

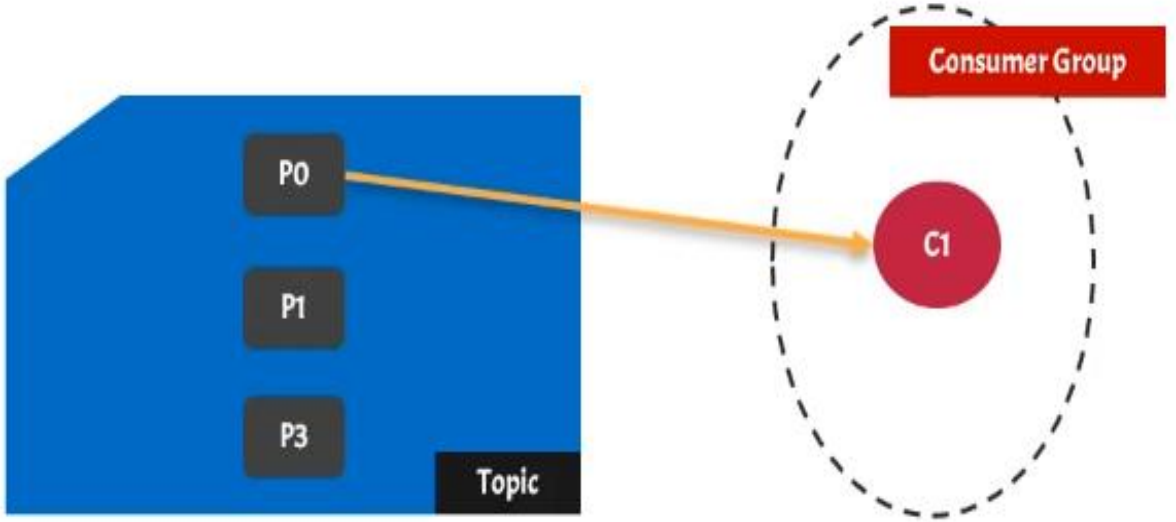
- Producer can also select the partition of their choice in a topic where it want to publish the message
- All the messages will be published to as single partition (P0)



Partitioner method with three parameters

- key_bytes
- all_partition
- available_partition

- Consumer are the Kafka components that consume message from Kafka topic
- Internally consumer consume message from Kafka topic partition
- Every consumer is always assigned to a consumer group
- If no group_id is provided then random group id is assigned

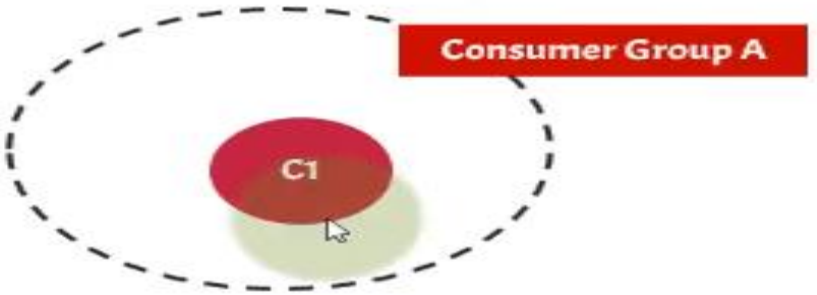



```
pip install kafka-python
```

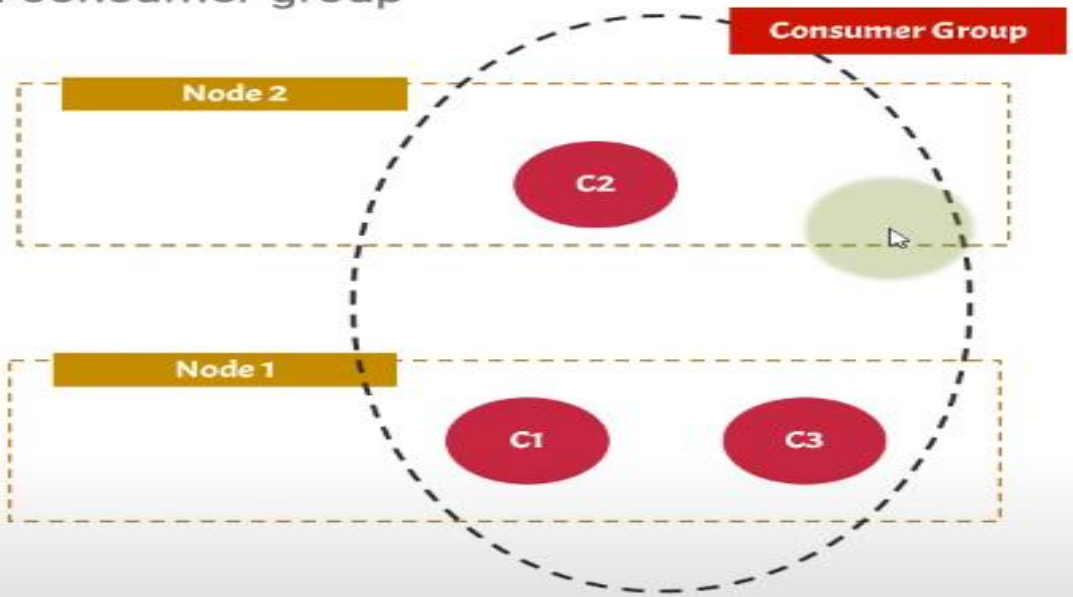
Configuration Needed by Consumer

- topic
- bootstrap_servers
- group_id

- Consumer group can be defined as logical grouping of one or more consumer
- It is mandatory for a consumer to register itself to a consumer group



- Consumer instances are separate process
- Consumer instance of same consumer group can be on different nodes
- Let's say we have three consumer instances in a consumer group



Example : 1 Topic , 1 Partition, 1 Consumer

- All message will be published to P0 partition
- All message will be consumed by C1