On a high-level scaling is achieved by increasing Partition and increasing consumers in a consumer group.

Adding more Producers puts System in reducing throughput.

Kafka uses partitions to scale a topic across many servers for producer writes. Also, Kafka also uses partitions to facilitate parallel consumers. Consumers consume records in parallel up to the number of partitions.

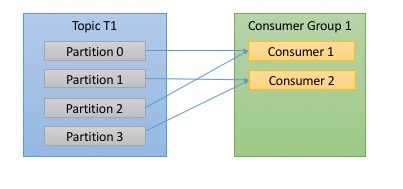
**How Scaling is achieved in Kafka?**

Consumers work as part of a *consumer group*. This is one or more consumers that work together to consume a topic. The group assures that each partition is only consumed by one consumer

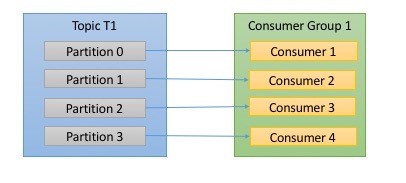
This is a good reason to create topics with a large number of partitions - it allows adding more consumers when the load increases. Note again that there is no point in adding more consumers than you have partitions in a topic - some of the consumers will just be ideal.

Lets take topic t1 with 4 partitions. Now suppose we created a new consumer,c1, which is the only consumer in group g1 and use it to subscribe to topic t1. Consumer c1 will get all messages from all four of t1 partitions.

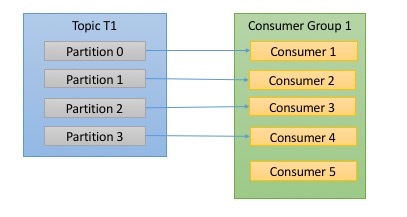
If we add another consumer, c2 to group g1, each consumer will only get messages from two partitions.



If g1 has 4 consumers, then each will read messages from a single partitions.



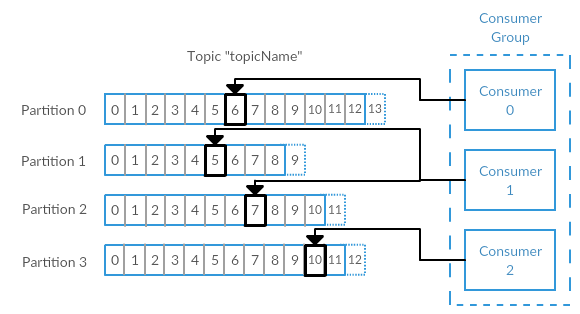
If we add more consumers to a single group with a single topic than we have partitions, than some of the consumers will be idle and get no messages at all.

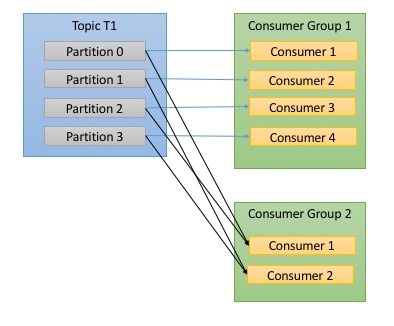


In below figure, there are three consumers in a single group consuming a topic. Two of the consumers are working from one partition each, while the third consumer is working from two partitions. The mapping of a consumer to a partition is often called *ownership* of the partition by the consumer.

In this way, consumers can horizontally scale to consume topics with a large number of messages. Additionally, if a single consumer fails, the remaining members of the group will rebalance the partitions being consumed to take over for the missing member.

For 1000 messages, 10 partitions can be created under a topic. 5 to 10 consumer can parallelly process the messages. If 1 or more consumer fails load will be balanced.





**Scaling within consumer group:**

Add consumers to an existing consumer group to scale, each additional consumer in a group will only get a subset (whichever is not read) of the messages.

**Scaling by creating new consumer group:**

Reads all the messages from one or more topics.