RabbitMQ ⟷ kafka

Scaling

Scaling up

Scale-up is done by adding more resources to an existing system to reach a desired state of performance. For example, a database or web server needs additional resources to continue performance at a certain level to meet SLAs.

Scaling out

Scale-out is usually associated with distributed architectures. There are two basic forms of scaling out: Adding additional infrastructure capacity in pre-packaged blocks of infrastructure or nodes (i.e. hyper-converged) or use a distributed service that can retrieve customer information but be independent of applications or services.

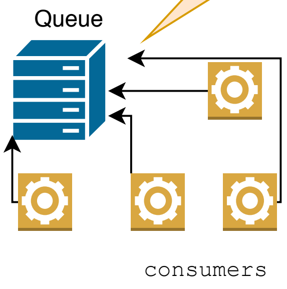
message

RabbitMQ pushes the message to the consumer, and once consumed and acknowledgment has arrived, message is removed from the queue.

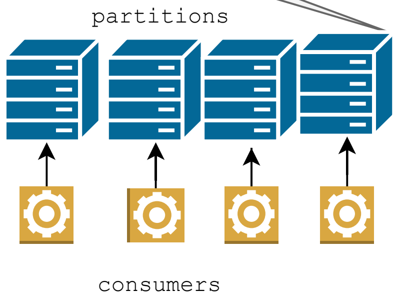
In Kafka messages are always remaining in the topic, also if they were consumed (limit time is defined by retention policy)

Distribution and parallelism

In RabbitMQ, you can scale out the number of consumers, this means, for each queue instance you will have many consumers, this called competitive consumers because they compete to consume the message, in this form the message processing work is spread by all the active consumers, but still message can be procced only once.



In Kafka, the way to distribute consumers is by topic partitions, and each consumer from the group is dedicated to one partition.
You can use the partition mechanism to send each partition different set of messages by business key, for example, by user id, location etc.


partitions
consumers