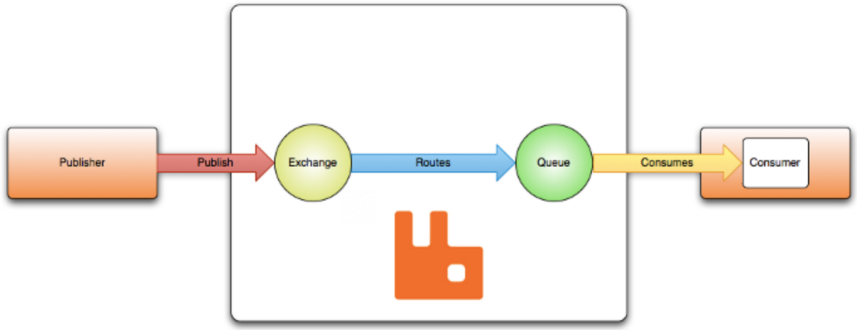
Message Queuing/Message Bus

Message Queuing/Message Bus

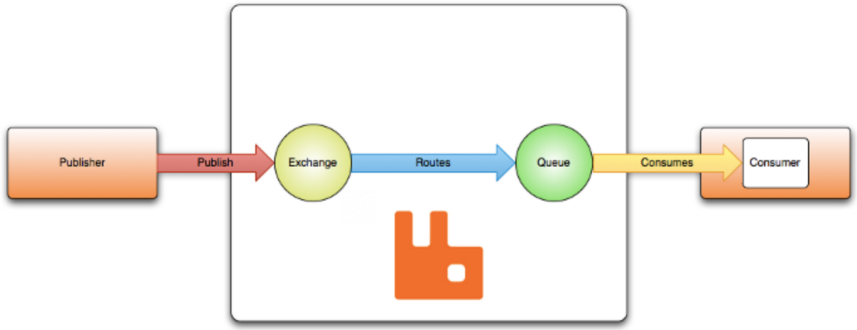
- Message Queuing allows applications to communicate by sending messages to each other. The message queue provides temporary message storage when the destination program is busy or not connected.

AMQP - Advanced Message Queuing Protocol



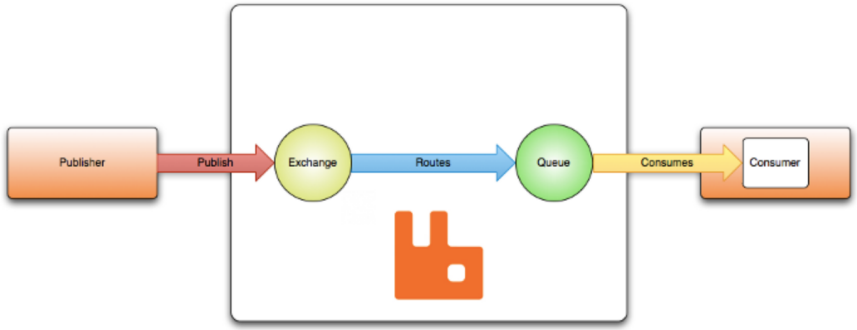
- AMQP 0.9.1 (latest version and what's supported by rabbitMQ)

AMQP - Advanced Message Queuing Protocol



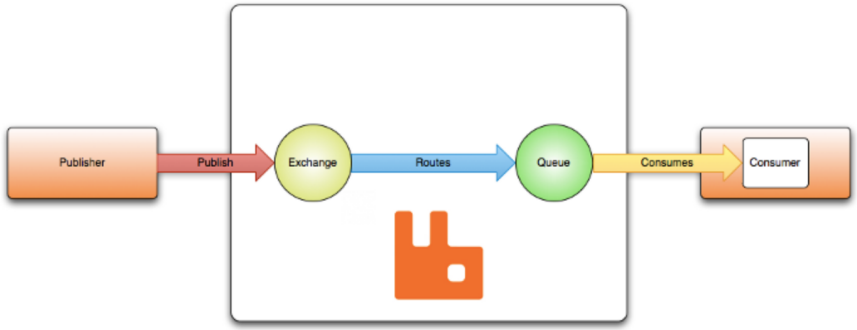
- AMQP 0.9.1 (latest version and what's supported by rabbitMQ)
- publishers writes to the exchange, there are few routing options based on which the message is put in a queue

AMQP - Advanced Message Queuing Protocol



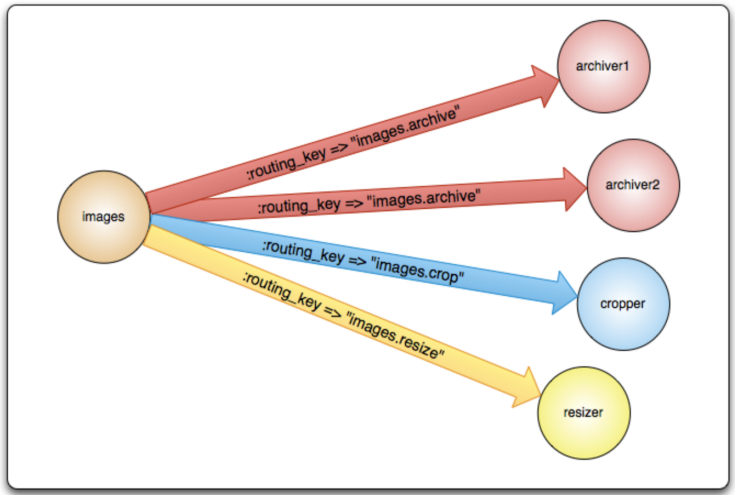
- AMQP 0.9.1 (latest version and what's supported by rabbitMQ)
- publishers writes to the exchange, there are few routing options based on which the message is put in a queue
- when a message cannot be routed, messages may be returned to publishers, dropped or put in dead letter queue

AMQP - Advanced Message Queuing Protocol



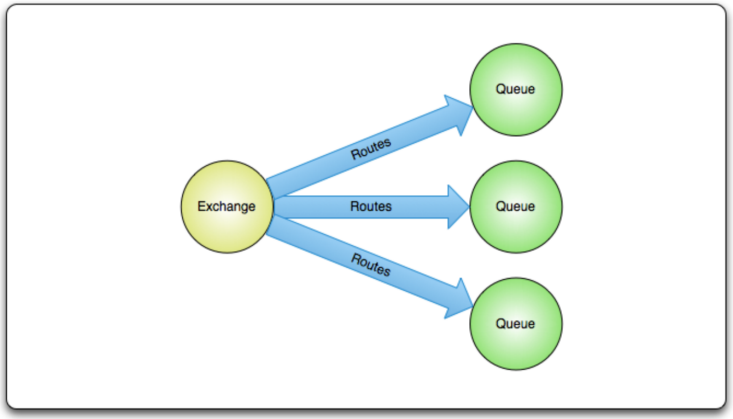
- AMQP 0.9.1 (latest version and what's supported by rabbitMQ)
- publishers writes to the exchange, there are few routing options based on which the message is put in a queue
- when a message cannot be routed, messages may be returned to publishers, dropped or put in dead letter queue
- messages are stored until a consumer receives a message and acknowledge

AMQP Direct Exchange



- Direct exchange (1-1 mapping)

AMQP Direct Exchange



- Direct exchange (1-1 mapping)
- Fanout exchange (1-n mapping)

AMQP Direct Exchange

- Direct exchange (1-1 mapping)
- Fanout exchange (1-n mapping)
- Headers exchange ('x-match' header based), ...

RabbitMQ scaling

- Consumer scaling is easy you can add more consumers when you publish quicker then you can consume

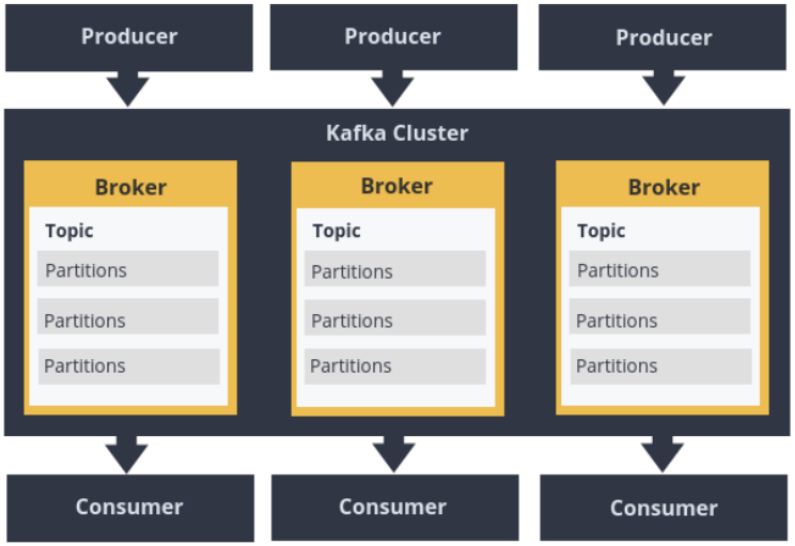
RabbitMQ scaling

- Consumer scaling is easy you can add more consumers when you publish quicker then you can consume
- Scaling broker is veritcal in RabbitMQ.

RabbitMQ scaling

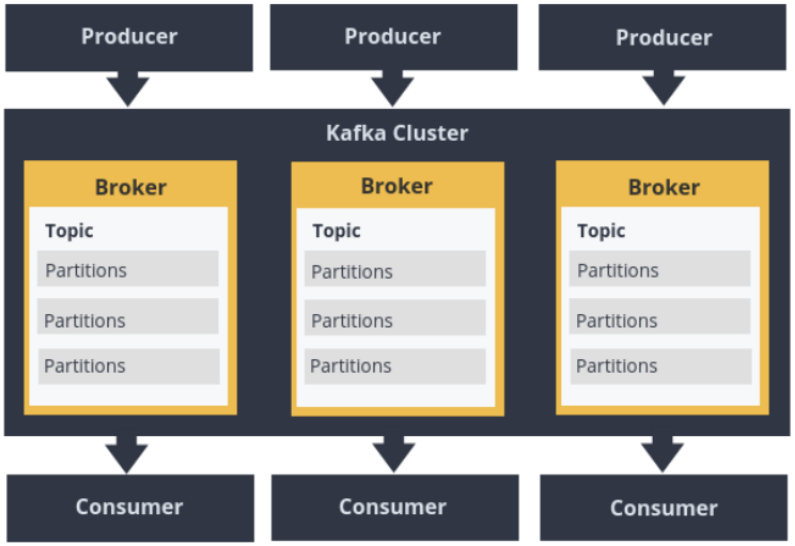
- Consumer scaling is easy you can add more consumers when you publish quicker then you can consume
- Scaling broker is veritcal in RabbitMQ.
- Horizontal scaling of borker is possible with clustering but it's annowying setup

Kafka



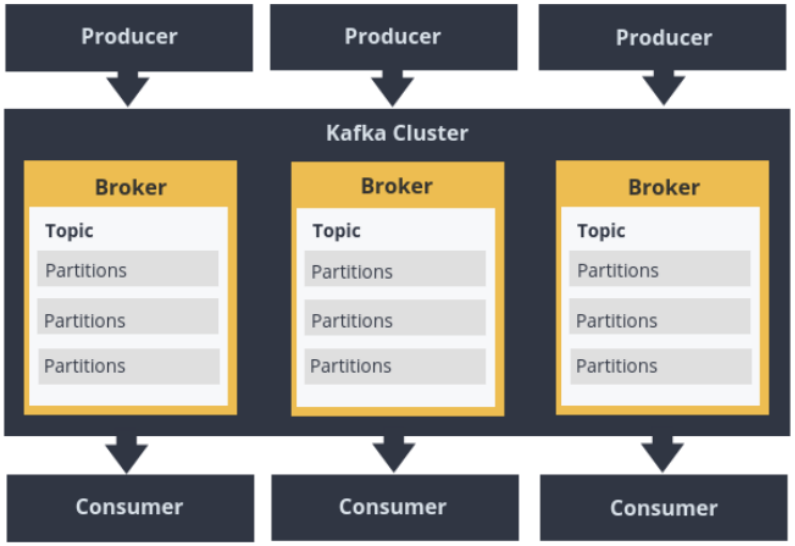
- by LinkedIn, built for large scale stream processing in a fault-tolerant fashion with at-least once delivery

Kafka



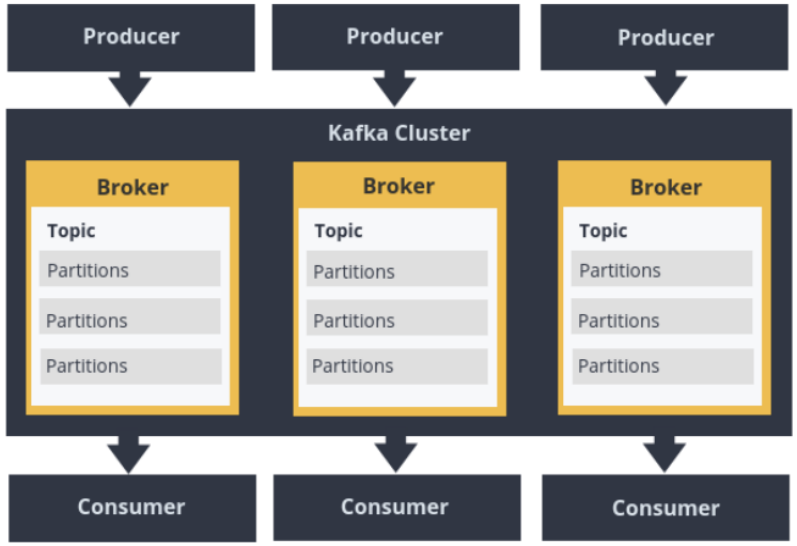
- by LinkedIn, built for large scale stream processing in a fault-tolerant fashion with at-least once delivery
- Kafka is based on the commit log, and messages comes from many producers in TCP

Kafka



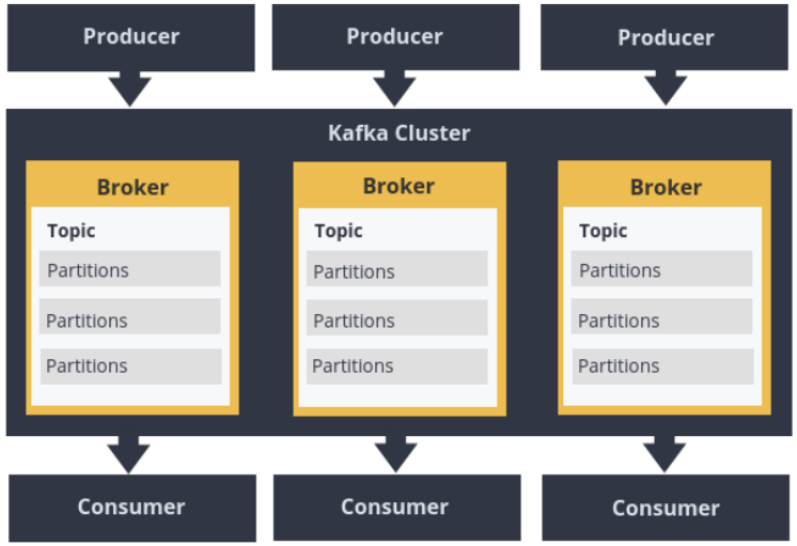
- by LinkedIn, built for large scale stream processing in a fault-tolerant fashion with at-least once delivery
- Kafka is based on the commit log, and messages comes from many producers in TCP
- Data in a topic is partitioned and within a partition, messages are strictly ordered by their offsets

Kafka



- by LinkedIn, built for large scale stream processing in a fault-tolerant fashion with at-least once delivery
- Kafka is based on the commit log, and messages comes from many producers in TCP
- Data in a topic is partitioned and within a partition, messages are strictly ordered by their offsets

Kafka

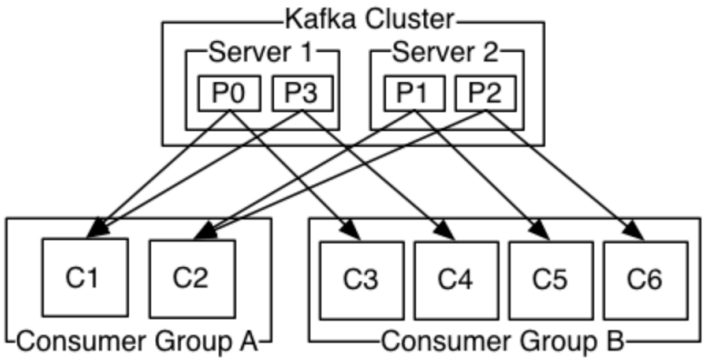


- by LinkedIn, built for large scale stream processing in a fault-tolerant fashion with at-least once delivery
- Kafka is based on the commit log, and messages comes from many producers in TCP
- Data in a topic is partitioned and within a partition, messages are strictly ordered by their offsets

Kafka

- 0.8.2 Most stable version of kafka, still with offsets stored in zookeeper

Kafka

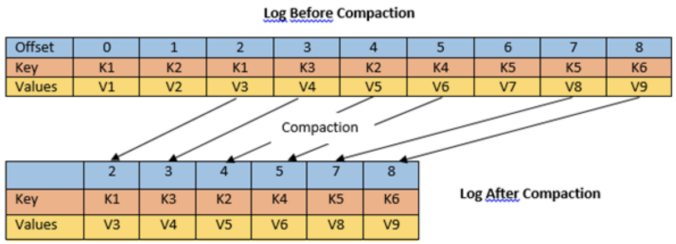


- 0.8.2 Most stable version of kafka, still with offsets stored in zookeeper
- 0.9.x Consumer groups, offsets were stored in kafka itself as just another topic with immutable updates

Kafka

- 0.8.2 Most stable version of kafka, still with offsets stored in zookeeper
- 0.9.x Consumer groups, offsets were stored in kafka itself as just another topic with immutable updates
- 0.11 Kafka supports Exactly-once delivery using idempotent producer and transaction. Blog

Kafka



- 0.8.2 Most stable version of kafka, still with offsets stored in zookeeper
- 0.9.x Consumer groups, offsets were stored in kafka itself as just another topic with immutable updates
- 0.11 Kafka supports Exactly-once delivery using idempotent producer and transaction. Blog
- 2.0 Compact topics - In simple terms, Apache Kafka will keep latest version of a record and delete the older versions with same key

Kafka and beyond

- Kafka now is a distributed streaming platform
- Connect and Streams APIs
- Mirroring (old but very powerful feature)

Kinesis

- Kinesis was modeled after Kafka 0.8.0
- Not a streaming platform like kafka, it's just a message bus.
- Managed by AWS, uses shards instead of partition but similar concept