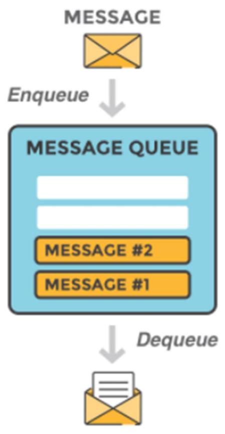
Report : RabbitMQ

Mohd Huzaifa, 2018158

Huzaifa18158@iiitd.ac.in.

1 WHAT IS RABBITMQ?

MQ here stands for Message Queuing. RabbitMQ is message queuing software where, as the name suggests, messages are stored in the form of a queue. It is also known as **queue manager** as it defines and manages the queues. Whenever an application wants to transfer a message, it connects to RabbitMQ and pushes the message to the queue. The message remains in the queue until the receiver comes and pops the message.



1.1 Act as Queue Manager

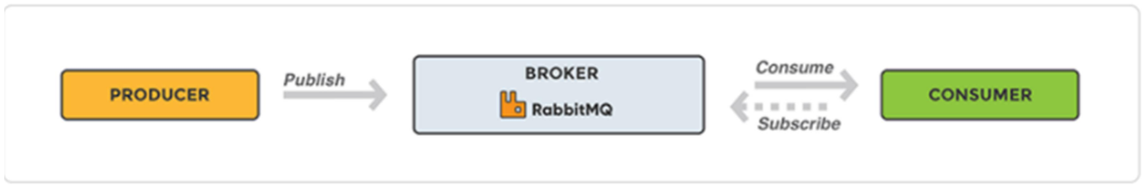
RabbitMQ itself acts as queue manager and manages many basic things like number of queues to define, which message goes to which queue, queue binding rules, etc. Number of queues and the binding criteria can be used to implement different connection patterns and protocols. These things are explained in detail in the later sections.

1.2 Acknowledgments and preventing message loss

We know that the producer generates messages and send them to RabbitMQ, which stores them in its queues. Consumer contacts rabbitMQ and retrieves the messages. These messages can be a task or request which consumer has to process. RabbitMQ provides the functionality to send acknowledgments after processing the request. Message will be deleted from the queue once it's processed without any failure. In case of any failure, it will resend the message to the consumer. In this way, it prevents message loss.

1.3 Act as Broker

It acts as message broker in a way that it is used as intermediate layer of various services like web applications, etc. It obviously acts as a middleman and helps in reducing load on the applications. They give a performance boost by giving time consuming and computationally intensive tasks to the third party applications.



2 COMPONENTS

This section will give a brief understanding of components of RabbitMQ, their working and some basic terminologies.

2.1 Messages

We talked about queuing messages and managing them, but what exactly do we mean by messages? In our context, a message can be any kind of information. It can be a text message, a request, information about a process to be started, or a task. For example: a client may want some service from a server. It can send the procedure call request in the form of a message. Based on different protocols and uses, message structure and content may vary. But these messages will be treated as same by RabbitMQ.

2.2 Queues

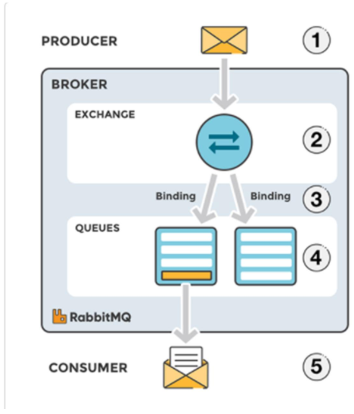
Queues are the data structures where elements are stored and retrieved in first come first server fashion. RabbitMQ is a type of queue manager which defines the queues and manages them. This implicitly implies that the messages are pushed to the queues by the sender and are popped by the receiver in FIFO order. Message entered first is also the one to be retrieved first. If we dive into the concept, we'll see that this FIFO helps in delivering messages in a particular order. There's no need for hashing or other sort of mapping to decide the order of received messages.

2.3 Binding Key and Routing Key

Binding key represents the relationship or mapping between exchange and queues. Whereas Routing is a key passed along with the message to identify the queue it's targeted to. Routing key is matched with the binding key and the message is filtered accordingly.

2.4 Exchange

As marked (2) in the image, when RabbitMQ receives a message, it doesn't just blindly put the message in a queue. The message has to go through the exchange which routes it to different queues based on the routing key. The bindings can be seen as routes to the queues. Based on the attributes like routing keys, exchange will route the messages to one of the queue via these bindings.



Types of exchange:

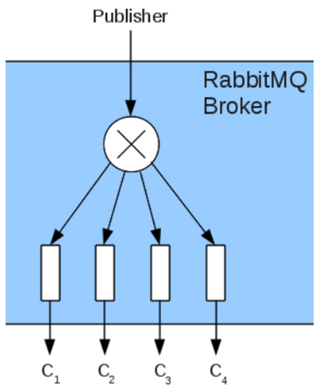
1. Fanout exchange: No specific filtering is done here. The message is pushed on all the queues as done in broadcasting.
2. Direct exchange: Binding key and routing key are exactly matched. The message is transferred to the queue where binding key is exactly equal to the routing key.
3. Topic exchange: Instead of exact matching, it allows pattern matching. Binding key can contain * and # symbols to represent patterns. Routing key can be a list of keys to provide some flexibility.

3 PROTOCOLS

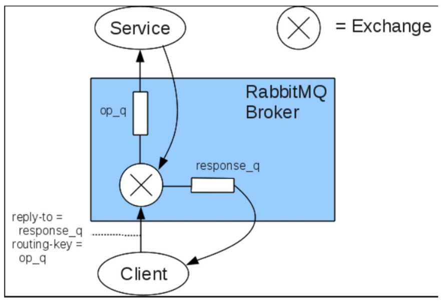
3.1 AMQP protocol

It stands for Advanced Message Queuing protocol. It allows communication between clients and the message brokers like RabbitMQ. A lot of flexibility is given in a way that the whole protocol is programmable. Parameters like durability, acknowledgments, etc. allows message retransmission, memory freeing, etc. Echange, binding rules, queues, etc can be set to implement connection patterns as per the need. Some of the implementations are :

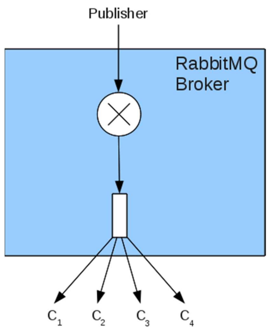
- **Pub/Sub:** In this protocol, there are publishers and subscribers. Publishers publish or produce messages. Subscribers receive and process these messages. This protocol or pattern can be achieved using RabbitMQ by binding multiple queues to an exchange. As shown in the image, C1, C2…. are consumers or subscribers. When the publisher publishes a message, it passes through an exchange, which will filter the message and route it to the corresponding subscribers.



- **Remote Procedure Call in Client server:** In RPC, a program can request a service or call a procedure of another program located in a different machine. This can be implemented using RabbitMQ by sending the request through it. Client will send a message to the queue and specify a routing key. Based on this routing key, exchange will filter the message and send to a specific queue. Server will pick up the message from the queue and process it. The response will be sent back to the exchange with corresponding routing key to the reply queue. In this way, a client can achieve service from another program remotely.



- **Work Distribution:** RabbitMQ can be used for work distribution by attaching multiple workers/consumers to the same queue. Therefore, when a message is pushed to the queue, it'll be picked up by only one consumer. Thus, producer can push multiple tasks on the queue and the available workers will take them one by one.



3.2 Other Protocols

STOMP: It's a text based messaging protocol
MQTT : It's a binary protocol and is well defined for Pub/Sub.
AMQP 1.0 : It's slightly different from the above mention AMQP.

4 WHY TO USE RABBITMQ?

→It can be used for Load Balancing, i.e. distribute the task or the message to multiple consumers.
→There's no need for the server/producer to be on the same page. For example producer may produce a message. After pushing the message on Queue, it doesn't have to worry about the consumer at all. Neither does it have to wait for the consumer to process the message. It can produce another message and put it on the queue.
→It also offers compatibility between different machines. For example, the message can be produced in one programming language and processed in other.

5 REFERENCES

→Part 1: RabbitMQ for beginners - What is RabbitMQ? - CloudAMQP
→RabbitMQ tutorial - "Hello world!" — RabbitMQ
→Understanding AMQP, the protocol used by RabbitMQ (spring.io)