ESCOLA
POLITÈCNICA SUPERIOR
UNIVERSITAT DE LLEIDA

DISTRIBUTED COMPUTING

RabbitMQ – components, benefits and application

Marc Cervera Rosell - 47980320C

Academic course 2022 - 2023

Bachelor's degree in computer engineering

Contents

List of Figures

Introduction

The main aim of this report is going to be to discover RabbitMQ software.

As everybody knows, there're three main essential attributes when designing and developing software that helps companies to offer trustworthy digital services and products. These three main attributes are: scalability, resiliency and interoperability.

In the actual market, there is a big amount of technologies that achieve that target. Among these technologies, we can find RabbitMQ.

RabbitMQ is an open-source message broker, sometimes called message-oreinted middleware. This software implements the "Advanced message queuing protocol". The RabbitMQ server is programmed in Erlang and uses the "Open telecom platform" framework to build his distributed execution and error commutation capacities.

RabbitMQ is a cross-platform software and the source code is released under the Mozilla public license.

In a simplified way, RabbitMQ defines queues that will store the messages sent by producers until the consumer apps get the message and process it.

So, It can be said that RabbitMQ, in his broker role, has the work of routing to the correct consumer the messages that the producer sends. To download and starting to use the software, just enter in the RabbitMQ webpage.

Architecture

Originally, RabbitMQ implemented the "Advanced message queuing protocol" (AMQP) but has been extended with a plug-in architecture to support "Streaming text oriented messaging protocol" (STOMP), "MQ telemetry transport" (MQTT), "HTTP and WebSockets" and "RabbitMQ streams".

HTTP is not really a messaging protocol but RabbitMQ can transmit messages over HTTP in three ways. The first one is the web STOMP plugin that supports STOMP messaging to the browser using WebSockets. The second one, is the web MQTT plugin that supports MQTT messaging to the browser using WebSockets. The tird, and the final, way is the management plugin that supports a simple HTTP API to send and receive messages. This is primarily intended for diagnostic purposes but can be used for low volume messaging without reliable delivery.

AMQP

As AMQP is the "core" protocol supported by the broker, this section will be the most extensive.

RabbitMQ, originally only supported AMQP 0-9-1. All of the variants are fairly similar to each other, with later versions tidying up the unclear or unhelpful parts of earlier versions. AMQP 0-9-1 is a binary protocol, and defines quite strong messaging semantics. For client it's a reasonably easy protocol to implement, and as such there are a large number of client libraries available for many different programming languages and environments.

AMQP is an advanced queue messaging protocol that stands out for his fidelity. There're commercial and open-source servers and interoperable clients for many programming languages, which facilitates their use. It's used by big corporations that process millions of messages.

The deffinition of AMQP can be simplified in three words: "message-oriented middleware". Behinf this simple definition there're a lot of features available. Before AMQP there were some message-oriented middlewares such, for example, JMS, but AMQP has become the standard protocol to keep when a queue-based solution is chosen.

AMQP is pretty simple to understand, there're client applications called producers that create messages and deliver it to an AMQP server, also called, broker. Inside the broker the messages are routed and filtered until arrive to queues where another applications called consumers are connected and get the messages to be processed.
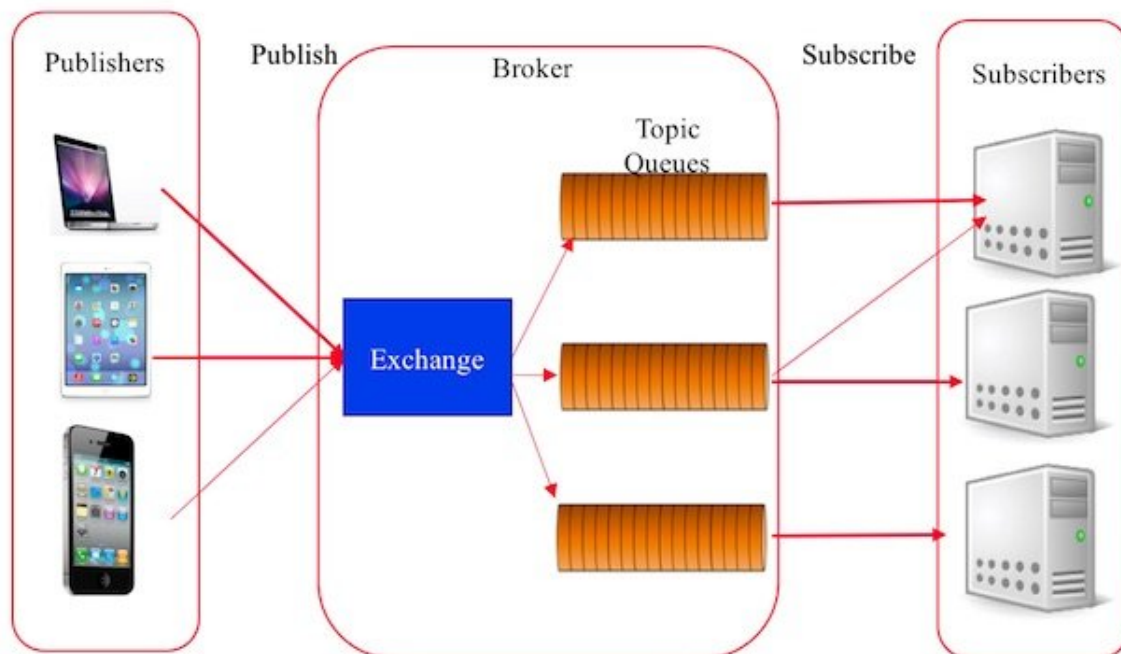

Figure 1: AMQP architecture work

Once seen this, it's time to deep inside the broker, where the magic of AMQP is done. The broker has three different parts. The exchange, the queues and the bindings.

Exchange

This component is in charge of receiving the messages that have been sent to the broker by a producer and is the responsible of place them in the proper queue according to a routing key. This means that the producer does not send the messages directly to the queue, but send it to an exchange with a routing key.

Routing key

Queue

Binding

Virtual host

MQTT

WebSockets

RabbitMQ streams

RabbitMQ vs. market

Advantages

Disadvantages

References

- [Wikipedia's RabbitMQ article](#)

- [RabbitMQ's webpage](#)

- [*Conectar microservicios con colas de mensajes usando Spring y RabbitMQ*'s webpage](#)

- [*Conozcamos sobre RabbitMQ, sus componentes y beneficios*'s webpage](#)

- [AMQP architecture work picture](#)