Colas de mensajes: RabbitMQ

+ 2015-06-19 — 0 Comments
Miguel Ángel García

Cuando se desarrolla una aplicación y ésta comienza a crecer, a menudo necesitamos interconectar distintos componentes. En estos casos se utiliza un *middleware* que nos permita comunicar las distintas piezas.

Una opción es usar una cola de mensajes. No es el mecanismo más rápido, pero probablemente sí el más sencillo, y permite realizar acciones de forma asíncrona.

Y un gestor de colas de mensajes sencillo, robusto y muy utilizado es RabbitMQ (https://www.rabbitmq.com/).

Instalando RabbitMQ

Esta aplicación *Erlang* es sencilla de instalar en Linux:

```
apt-get install rabbitmq-server
```

Y tendremos la aplicación funcionando, escuchando en el puerto 5672.

Es interesante instalar el plugin de gestión (https://www.rabbitmq.com/management.html) para poder ver qué está pasando ahí dentro. Para ello basta con hacer:

```
rabbitmq-plugins enable rabbitmq_management
```

Y automáticamente lo tendremos escuchando en http://localhost:15672/ (http://localhost:15672/)

Lo básico de RabbitMQ

Hay algunos conceptos que hay que tener claros de RabbitMQ:

Exchange
Es el punto de entrada de un mensaje. Pueden ser **Direct**, si entregan un mensaje en una cola, **Fanout** si se entregan copias del mensaje a todas las colas o **Topic** si se entregan copias del mensaje sólo a algunas colas.

Queue
Es el punto de lectura de un mensaje. Pueden ser **durable** o persistentes, si almacenan los mensajes para sobrevivir a un reinicio de RabbitMQ (https://www.rabbitmq.com/). También pueden ser **exclusivas**, si sólo un consumidor puede estar conectado a la vez.

Bindings

Son reglas que indican cómo llegar de un **Exchange** a las **Queue** asociadas.

Routing key
Filtro asociado a un **Binging** que permite seleccionar sólo algunos mensajes para dicho **binding**.

Producer o Productor
Programa que escribe en un **Exchange**

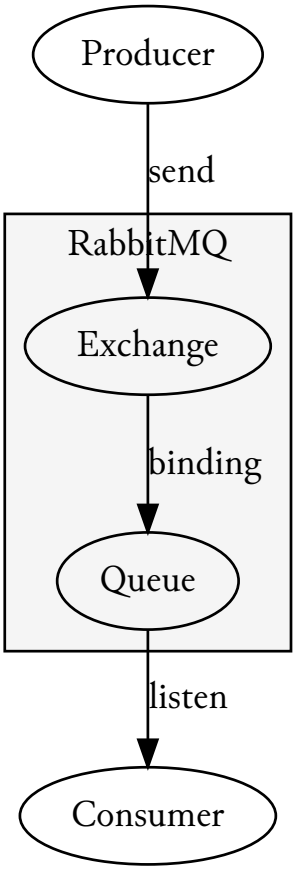
Consumer o Consumidor
Programa que escucha en una **Queue**

AMQP
Protocolo de comunicaciones utilizado por RabbitMQ. Lo usa tanto el **Productor** como el **Consumidor**.

Virtual host o vhost
Un entorno aislado, con sus propios grupos de usuarios, *exchanges*, *queues*, ...

mnesia
La base de datos interna de RabbitMQ (https://www.rabbitmq.com/)

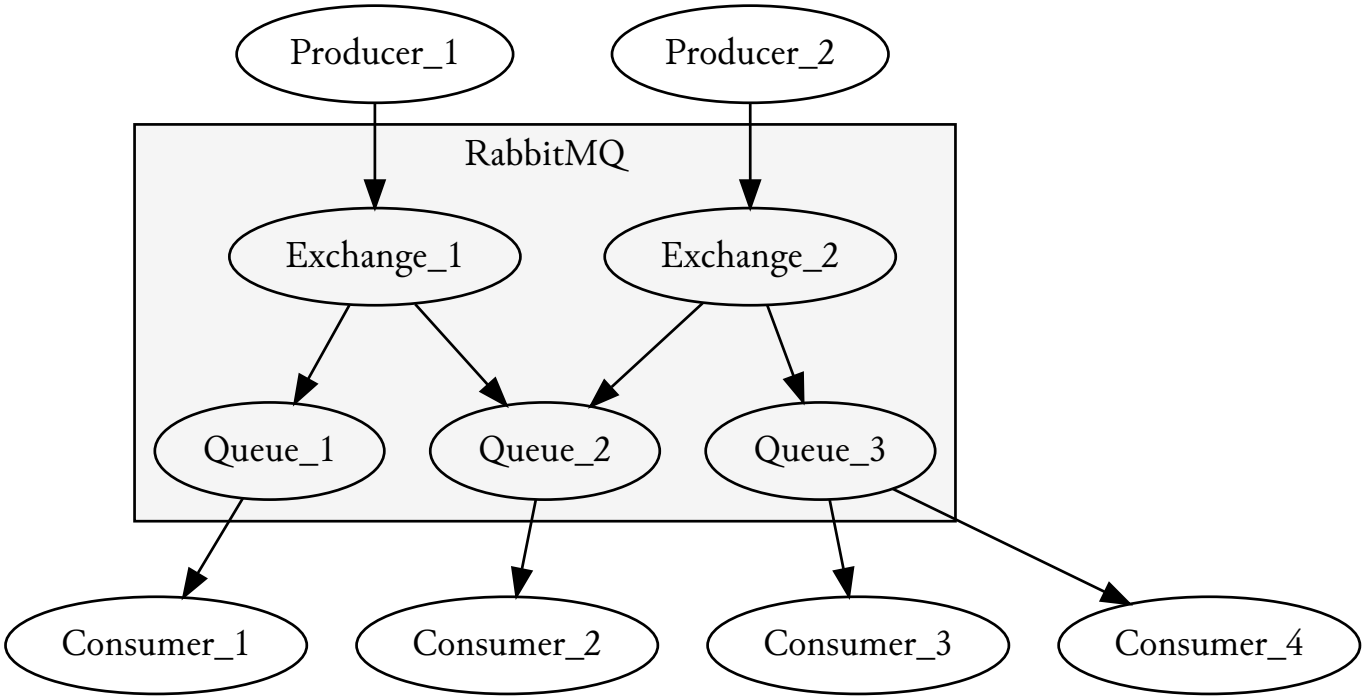
Una vez dicho esto, podemos describir el flujo básico de RabbitMQ (https://www.rabbitmq.com/):



b'\nn "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">\n\n\n \n'

Claro, que esto puede evolucionar a entornos mucho más complejos:

b'\nn "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">\n\n\n



\n'

En este entorno, y asumiendo que los *Exchange* están en modo **Topic** (es decir, que un mensaje llega a todas las colas):

- `Consumer_1` recibe todo lo que produzca `Producer_1`, pasando por `Queue_1`. Este tipo de escenario es igual que el básico.
- `Consumer_2` recibe todo lo que produzca tanto `Producer_1` como `Producer_2`, pasando por `Queue_2`. Útil para procesar eventos que se producen en distintos lugares pero tienen un destino común. Ejemplo: guardar logs de distintas máquinas en base de datos.
- `Consumer_3` y `Consumer_4` se repartirán lo producido por `Producer_2`, pasando por `Queue_3`. Útil cuando la carga producida es muy alta y se requieren muchos recursos para consimir cada mensaje. Ejemplo: Renderización de vídeo 3D.

Puertos y valores por defecto

Si no cambiamos nada, RabbitMQ (https://www.rabbitmq.com/) escucha en el puerto 5672, y la interfaz de administración en el 15672.

Se crea un usuario *guest/guest* que nos permite acceder a ambas.

RabbitMQ (https://www.rabbitmq.com/) consume mucha memoria. Recomiendo utilizar servidores dedicados, para que no se haga con toda y tengamos problemas.

Algunas tareas de administración

Crear un vhost

Con el fin de aislar un entorno, podemos crear un **vhost**. Nada más sencillo:

```
rabbitmqctl add_vhost /mi_vhost
```

Crear un usuario

Ahora podemos crear un usuario

```
rabbitmqctl add_user my_user my_pass
```

Y podemos asociarlo al **vhost**:

```
rabbitmqctl set_permissions -p /my_vhost my_user ".*" ".*" ".*"
```

Asignar permisos

Si queremos poder utilizar este usuario para entrar en la interfaz de administracción, necesitamos darle permisos de *management* y *monitoring*:

```
rabbitmqctl set_user_tags my_user management monitoring
```

Por tanto, los mensajes se escriben en un **Exchange** pero se leen de una **Queue**. Para llegar de un **Exchange** a una **Queue** se utilizan los **Routes**.

Crear un cluster

Una de las tareas típicas es crear un cluster de RabbitMQ (https://www.rabbitmq.com/clustering.html). Esto nos permite mejorar la disponibilidad del sistema (/blog/en-busca-de-los-cinco-9s), ya que si uno de nuestros servidores cae, el otro seguirá atendiendo peticiones.

Esta tarea es algo más compleja. Supongamos que tenemos dos nodos: *rabbit1* y *rabbit2*. Vamos a añadir *Rabbit2*, dejando *Rabbit1* como maestro inicial. Lo primero que podemos hacer es ver el estado actual del sistema:

```
rabbit2$ rabbitmqctl cluster_status
[{nodes,[{disc,[rabbit@rabbit2]}]},{running_nodes,[rabbit@rabbit2]}]
...done.
```

Para crear un cluster, tenemos que parar uno de los RabbitMQ (https://www.rabbitmq.com/), pero debemos hacerlo con órdenes especiales:

```
rabbit2$ rabbitmqctl stop_app
Stopping node rabbit@rabbit2 ...done.
```

A continuación lo añadimos al cluster:

```
rabbit2$ rabbitmqctl join_cluster rabbit@rabbit1
Clustering node rabbit@rabbit2 with rabbit@rabbit1 ...done.
```

Y volvemos a arrancar:

```
rabbit2$ rabbitmqctl start_app
Starting node rabbit@rabbit2 ...done.
```

Ahora podemos comprobar que todo fue bien:

```
rabbit2$ rabbitmqctl cluster_status
Cluster status of node rabbit@rabbit2 ...
[{nodes,[{disc,[rabbit@rabbit1,rabbit@rabbit2]}]},
 {running_nodes,[rabbit@rabbit1,rabbit@rabbit2]}]
...done.
```

Aunque también podemos verlo desde la herramienta de administración (mucho más molona XD)

Es **IMPORTANTE** que todos los nodos de un cluster tengan la misma versión de RabbitMQ (https://www.rabbitmq.com/) o podríamos tener problemas.

Si se va a gestionar un cluster, recomiendo encarecidamente leerse toda la documentación de RabbitMQ sobre clusters (https://www.rabbitmq.com/clustering.html). Es poco y muy conveniente. Además, es bueno tenerlo siempre a mano.

Una vez creado el cluster se puede utilizar una VIp (Virtual IP), de manera que nos respondan ambos indistintamente. Si uno cae, la VIp se puede cambiar mediante Keepalive (http://tldp.org/HOWTO/TCP-Keepalive-HOWTO/usingkeepalive.html) y, si preparamos nuestra aplicación para hacer reintentos, no se verá afectada por la caída.

En la interfaz de administración podemos ver cómo los mensajes estarán replicados en ambos nodos, el estado de los mismos, etc.

Programando

Para programar contra un RabbitMQ (https://www.rabbitmq.com/) recomiendo usar una de las librerías AMQP (https://www.rabbitmq.com/devtools.html). Hay otras opciones de más alto nivel. Por ejemplo, en Python está Kombu (https://kombu.readthedocs.org/en/latest/), que se abstrae de la cola de mensajes a usar, o incluso Celery (https://pypi.python.org/pypi/amqp), que permite usar una cola de mensajes para la gestión de tareas distribuidas.

Veamos unos ejemplillos sencillos. Recordad que hay muchas opciones y se pueden configurar bastantes historias, pero voy a ir a lo básico:

Python

Utilizaré la librería *Pika* (tomado de la propia documentación de la librería):

El consumidor:

```
import pika

connection = pika.BlockingConnection()
channel = connection.channel()
method_frame, header_frame, body = channel.basic_get('test')
if method_frame:
    print method_frame, header_frame, body
    channel.basic_ack(method_frame.delivery_tag)
else:
    print 'No message returned'
connection.close()
```

Y el productor:

```
import pika

parameters = pika.URLParameters('amqp://guest:guest@localhost:5672/%2F')
connection = pika.BlockingConnection(parameters)
channel = connection.channel()
channel.basic_publish('test_exchange',
                      'test_routing_key',
                      'Hello, world!',
                      pika.BasicProperties(content_type='text/plain',
                                           delivery_mode=1))
connection.close()
```

Java

Usando la rabbitmq-java-client (https://www.rabbitmq.com/java-client.html), el consumidor:

```
import com.rabbitmq.client.Connection;
import com.rabbitmq.client.Channel;

class Consumer
{
    public static void main(){
        ConnectionFactory factory = new ConnectionFactory();
        factory.setUri("amqp://guest:guest@hostName:5672/");
        Connection conn = factory.newConnection();

        Channel channel = conn.createChannel();

        channel.basicConsume(queueName, false, "myConsumerTag",
            new DefaultConsumer(channel) {
                @Override
                public void handleDelivery(String consumerTag,
                                           Envelope envelope,
                                           AMQP.BasicProperties properties,
                                           byte[] body)
                    throws IOException
                {
```

y el productor:

```
import com.rabbitmq.client.Connection;
import com.rabbitmq.client.Channel;

class Producer {
	public static void main(){
		ConnectionFactory factory = new ConnectionFactory();
		factory.setUri("amqp://guest:guest@hostName:5672/");
		Connection conn = factory.newConnection();

		Channel channel = conn.createChannel();

		byte[] messageBodyBytes = "Hello, world!".getBytes();
		channel.basicPublish("test_exchange", "test_routing_key", null, messageBodyBytes);

		channel.close();
		conn.close();
	}
}
```

Debo admitir que no los he probado... Así que si alguien encuentra algún fallo, que me lo diga XD

Conclusión

RabbitMQ (https://www.rabbitmq.com/) es un canal de comunicaciones. Quizá no sea muy rápido, ya que no deja de ser un *man-in-the-middle*, pero permite integrarse fácilmente con cualquier sistema existente.

Además, como no fuerza ningún formato de mensaje, resulta muy sencillo crear conectores que permitan publicar/consumir mensajes e integrar distintos escenarios. Y en las últimas versiones han añadido soporte para mensajes prioritarios (https://www.rabbitmq.com/priority.html), lo que puede dar mucho juego para enviar mensajes de control.

De todas maneras, RabbitMQ (https://www.rabbitmq.com/) es sólo un sistema de colas de mensajes, pero hay otros. También está Apache ActiveMQ (http://activemq.apache.org/), Apache Kafka (http://kafka.apache.org/), Apache Qpid (https://qpid.apache.org/), ZeroMQ (http://zeromq.org/), HornetQ (http://hornetq.jboss.org/), Sparrow (https://code.google.com/p/sparrow/), Starling (https://github.com/starling/starling/tree/master), Amazon SQS (http://aws.amazon.com/es/sqs/), Beanstalkd (http://kr.github.io/beanstalkd/), ...

- Publicado:Miguel Ángel García + 2015-06-19 Æ **rabbitmq**

ˆ Publicación anterior (/blog/monitorizacion-y-alertado/)

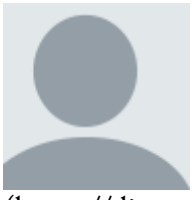
Siguiente publicación ˇ (/blog/creando-tu-propia-entidad-certificadora-ssl/)

Comentarios

Creando tu propia Entidad Certificadora SSL (y volcando certificados autofirmados) (http://magmax.org/blog/creando-tu-propia-entidad-certificadora-ssl/) · 3 months ago (http://magmax.org/blog/creando-tu-propia-entidad-certificadora-ssl/#comment-2531350429)

-  (https://disqus.com/by/edgaralfonsorosadopacheco/) Edgar Alfonso Rosado Pacheco (https://disqus.com/by/edgaralfonsorosadopacheco/)
  ¿Como instalo los certificados root en IE o Chrome? no he podido instalar el certificado con extencion .pem

  Creando tu propia Entidad Certificadora SSL (y volcando certificados autofirmados) (http://magmax.org/blog/creando-tu-propia-entidad-certificadora-ssl/) · 3 months ago (http://magmax.org/blog/creando-tu-propia-entidad-certificadora-ssl/#comment-2531081492)

-  (https://disqus.com/by/davidvillaalises/) David Villa Alises (https://disqus.com/by/davidvillaalises/)
  Estupendo tutorial para empezar con estas historias para no dormir.

  PD: Una coseja, has puesto dos veces que watcher no es libre ni gratis.

  Monitorización con ElasticSearch, Kibana y TopBeat (http://magmax.org/blog/monitorizacion-con-elasticsearch-kibana-y-topbeat/) · 3 months ago (http://magmax.org/blog/monitorizacion-con-elasticsearch-kibana-y-topbeat/#comment-2506305537)

GitHub Repos

Status updating...

@magmax (https://github.com/magmax) on GitHub