eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

ENTREGA INDIVIDUAL

RabbitMQ

Unai Solaun

ADMINISTRACIÓN DE SISTEMAS
Ingeniería Informática de Gestión y Sistemas de la Información (UPV/EHU)

2022

¿Qué es RabbitMQ?

RabbitMQ es un software de cola de mensajes, también conocido cómo broker de mensajería o administrador de colas, es decir, es un software donde se definen las colas a las que se conectan las aplicaciones para transferir un mensaje o varios mensajes. Esta tecnología implementa el estándar AMQP (Advanced Message Queuing Protocol) en la capa de aplicaciones de un sistema de comunicación, y la orientación a mensajes, encolamiento, enrutamiento, exactitud y seguridad son las características que lo definen.

El mensaje puede contener información sobre un proceso o tarea que debería iniciarse en otra aplicación o podría ser simplemente un mensaje de texto. El software del administrador de colas almacena los mensajes hasta que la aplicación receptora se conecta y retira un mensaje de la cola, y a continuación la aplicación receptora procesa el mensaje.

Funcionamiento

RabbitMQ actúa cómo intermediario para varios servicios, y para comprender su funcionamiento se usará cómo ejemplo una aplicación web. Se puede utilizar para reducir las cargas de entrega de los servidores de aplicaciones web al delegar tareas que normalmente consumirían mucho tiempo o recursos a un tercero que no tiene otro trabajo.

En un escenario en el que la aplicación web permite a los usuarios cargar información en su sitio web. El sitio maneja esta información, generará un PDF y se lo envía por correo electrónico al usuario. El manejo de la información, la generación del PDF y el envío por correo electrónico, en este caso, llevaría varios segundos. Esa es una de las razones por las que se utilizará una cola de mensajes para realizar esa tarea.

Cuando el usuario ingresa la información de usuario en la interfaz web, la aplicacion web crea un mensaje de “procesamiento de PDF” que incluye toda la información importante que el usuario necesita en un mensaje y lo coloca en una cola definida en RabbitMQ.

La arquitectura básica de una cola de mensajes es simple: existen aplicaciones cliente llamadas productores que crean mensajes y los entregan al intermediario (RabbitMQ o la cola de mensajes). Otras aplicaciones, llamadas consumidores, se conectan a la cola y se suscriben a los mensajes a procesar. El software puede actuar cómo productor, consumidor o cómo productor y consumidor de mensajes. Los mensajes colocados en la cola se almacenan hasta que el consumidor los recupera.

Tareas

Aplicación "Cliente"

La aplicación cliente, yo la he llamado **receiver**. Es la aplicación que se suscribe al topic "hello" y espera a los mensajes que se publiquen en esa cola, es la que actúa, en otras palabras, cómo consumidor.

En el fichero *receive.py* se encarga de suscribirse al topic. Para conseguir esta conexión he usado la librería *Pika*, que es una librería de cliente RabbitMQ para Python, que implementa el protocolo AMQP mencionado anteriormente. Este fichero va acompañado de su Dockerfile que incluye las instrucciones que se necesitan ejecutar de manera consecutiva para cumplir con los procesos necesarios para la creación de una nueva imagen.

Entorno Docker Compose

Para crear un entorno Docker Compose, he definido el archivo *docker-compose.yml*, que usará tres aplicaciones: la aplicación asignada, es decir, RabbitMQ, la aplicación cliente y la tercera aplicación, que se detalla posteriormente.

Lo que puedo destacar de este fichero es:

- Puertos (información obtenida en https://www.rabbitmq.com/networking.html):
  - Puerto *5672*, es el puerto principal de RabbitMQ y es necesario si se quiere hacer uso de la cola de mensajes.
  - Puerto *15672*, es el puerto para la API HTTP de RabbitMQ, que por comodidad, he redireccionado al puerto *8080*.
- Usuario: guest
- Contraseña guest
- Imagen receiver que espera 30 segundos para iniciarse (la imagen RabbitMQ tarda por defecto unos 20 en iniciarse) que referencia al fichero python *receive.py*.
- Imagen sender que espera 40 segundos para iniciarse (espera a que las imágenes RabbitMQ y receiver estén iniciadas) y referencia al fichero python *sender.py*.

Para poner en marcha este entorno hay que ejecutar el comando:

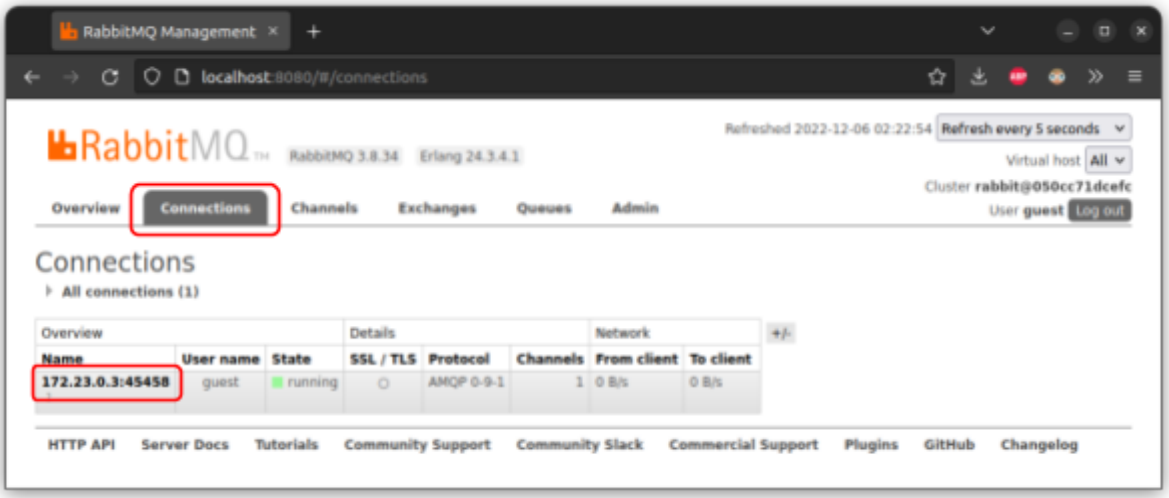
```
$ sudo docker compose up
```

Una vez iniciados todos los contenedores los mensajes de salida de la terminal muestran las conexiones entre el productor y consumidor con RabbitMQ. La IP xxx.xx.x.2:xxxxx es la correspondiente a RabbitMQ, xxx.xx.x.3:xxxx es la del consumidor y xxx.xx.x.4:xxxx del productor:

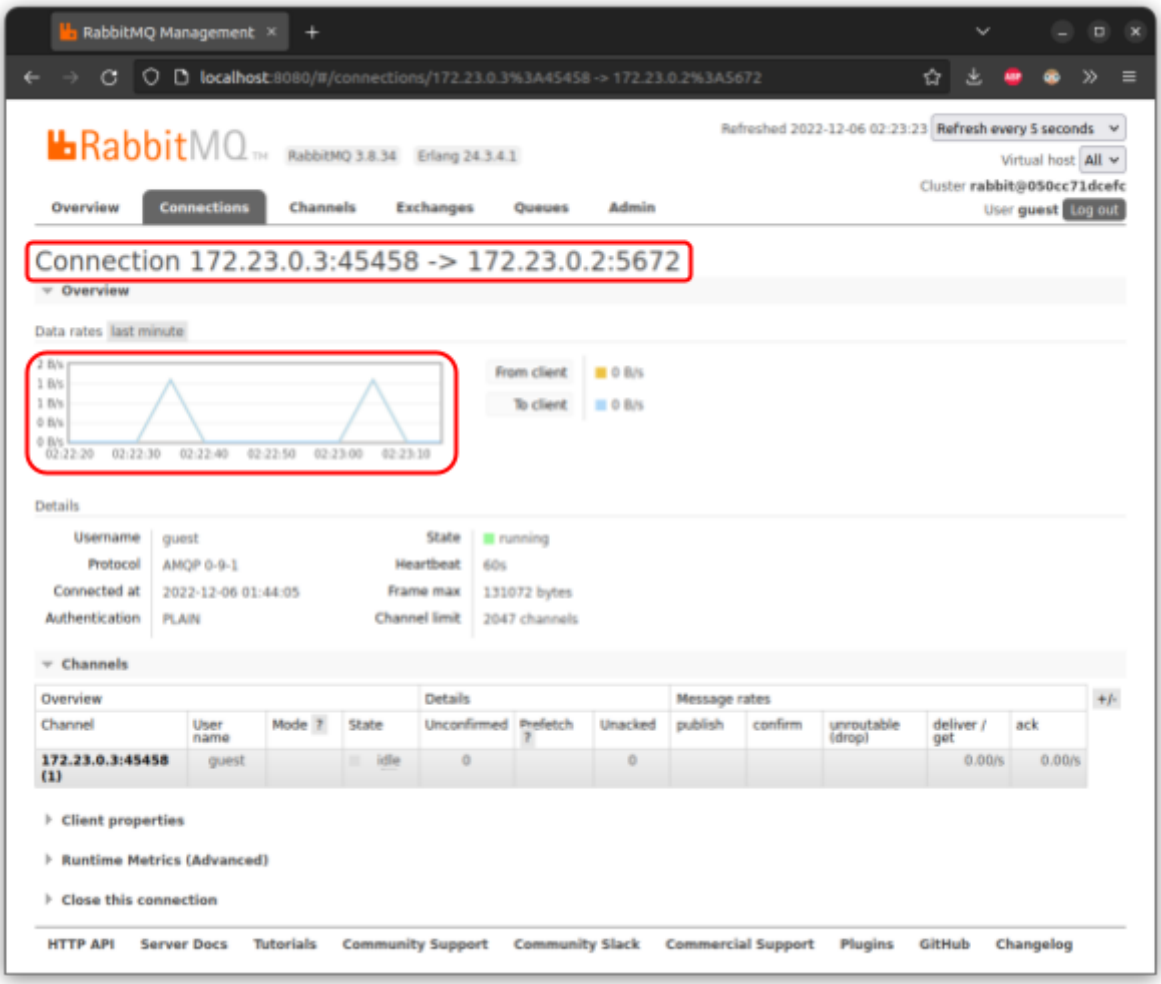
```
as_rabbitmq-rabbitmq-1  | 2022-12-06 00:37:52.359 [info] <0.704.0> accepting AMQP connection <0.704.0> (172.23.0.4:50420 -> 172.23.0.2:5672)
as_rabbitmq-rabbitmq-1  | 2022-12-06 00:38:02.288 [info] <0.725.0> accepting AMQP connection <0.725.0> (172.23.0.3:59138 -> 172.23.0.2:5672)
```

Accediendo a http://localhost:8080 e iniciando sesión con los usuario y contraseña proporcionados anteriormente, podemos navegar por la página web de monitorización de RabbitMQ, en la que podemos ver entre otros, las conexiones y la cola que pasaré a explicar un poco más detalladamente.

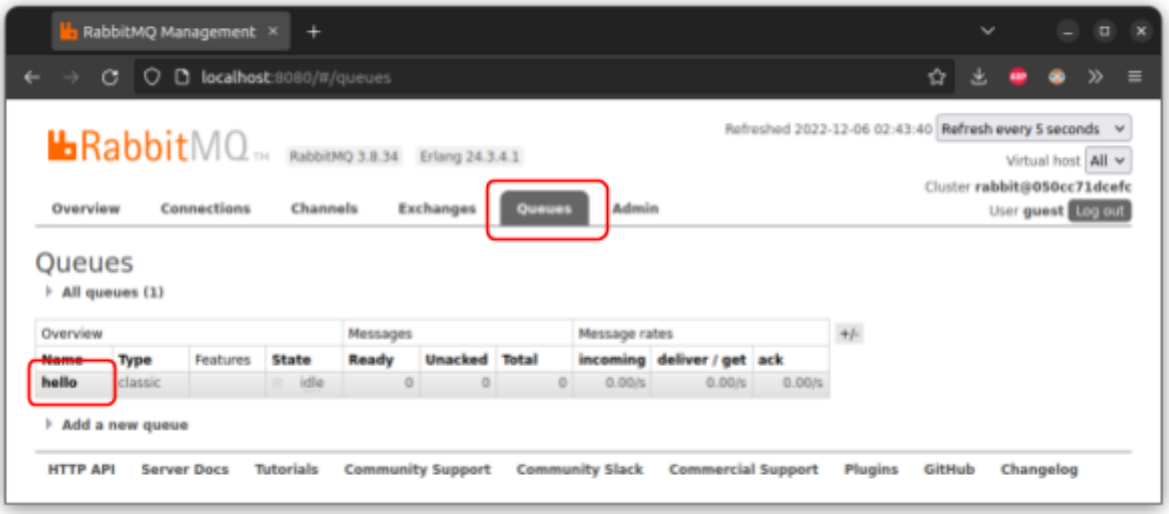
En el caso de las conexiones, se muestra una ventana cómo la siguiente:



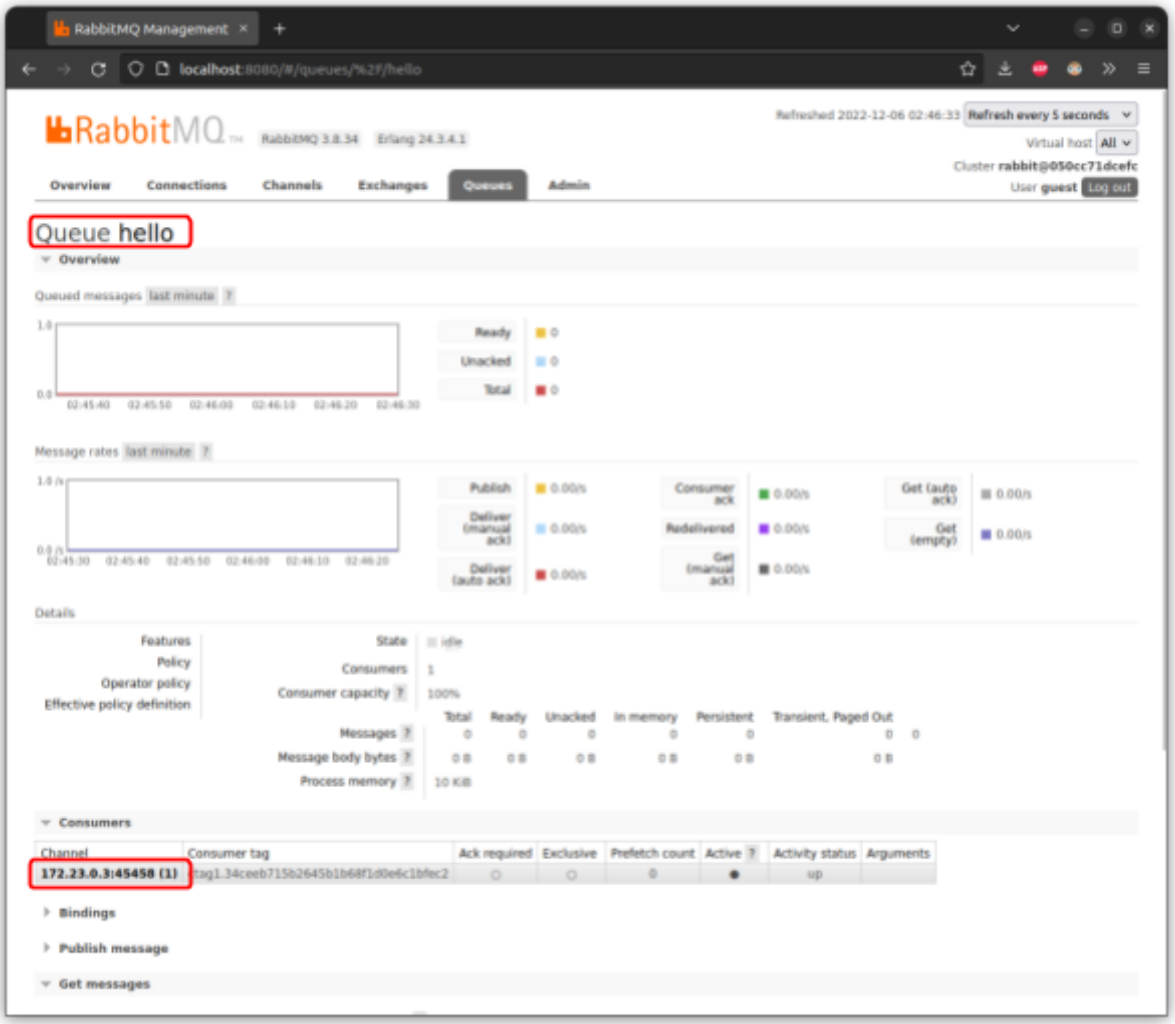
Si pulsamos en la ip correspondiente al consumidor, mostrará más información sobre el estado del mismo, cómo por ejemplo, los elementos resaltados en rojo en la siguiente imagen, acerca de la conexión y un gráfico sobre el estado del consumidor que se actualiza cada 5 segundos, que en la tercera aplicacion le daré uso para comprobar el estado del mensaje enviado.



En el caso de las colas, podemos observar que aparece la cola a la que la aplicación cliente se ha suscrito:



Podemos observar que en la cola llamada “hello” aparece la IP correspondiente a la aplicación cliente que es consumidora de esta cola:



Repositorio Docker Hub

Para subir las imágenes a un repositorio público de Docker Hub, en este caso la imagen receiver y sender, hay que seguir los siguientes pasos:

1. Registrarse en la web oficial de Docker Hub: https://hub.docker.com/
2. Abrir una terminal e introducir el siguiente comando para hacer login:
   $ docker login
   A continuación hay que indicar usuario y contraseña.
3. Ejecutar el siguiente comando para listar las imágenes:
   $ docker images

```
REPOSITORY                TAG                 IMAGE ID       CREATED         SIZE
as_rabbitmq-sender        latest              4a5026d1c4ab   3 hours ago     930MB
as_rabbitmq-receiver      latest              bb7dc935da27   3 hours ago     930MB
rabbitmq                  3.8-management-alpine   69b0f3f87ca9   6 months ago    166MB
```

4. En nuestro caso queremos subir las imágenes *as_rabbit-sender* y *as_rabbit-receiver* pero no se pueden subir a Docker Hub con esa nomenclatura, tienen que seguir la siguiente estructura:
   nombre_de_usuario/nombre_del_repositorio:etiqueta
5. Para el cambio de nombre de las imágenes se usa el siguiente comando:
   $ docker tag
6. En este caso los comandos para cambiar el nombre de las dos imágenes quedarían así:
   $ docker tag as_rabbitmq-receiver unaisolaun/as_rabbit-receiver:1.0
   $ docker tag as_rabbitmq-sender unaisolaun/as_rabbit-sender:1.0
7. Volver a ejecutar el comando del paso 3 para comprobar que se ha ejecutado correctamente.
8. Por último y para acabar el proceso, ejecutar el comando:
   $ docker push
9. Para estas imágenes los comandos a ejecutar serían:
   $ docker push unaisolaun/as_rabbit-receiver:1.0
   $ docker push unaisolaun/as_rabbit-sender:1.0

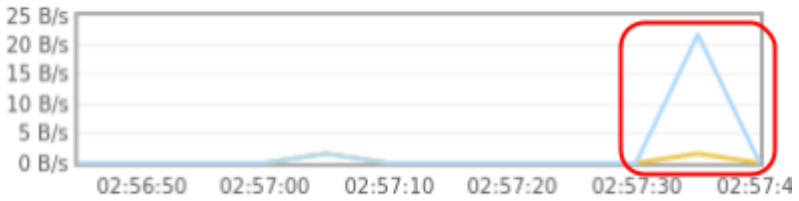
Los repositorios de las imágenes correspondientes se llaman *unaisolaun/as_rabbit-sender* y *unaisolaun/as_rabbit-receiver* y se pueden encontrar en los siguiente enlaces:

- https://hub.docker.com/repository/docker/unaisolaun/as_rabbit-sender
- https://hub.docker.com/repository/docker/unaisolaun/as_rabbit-receiver

Tercera Aplicación

La tercera aplicación que he desarrollado es un productor que envía un mensaje al topic al que está suscrito la aplicación cliente para que éste lo reciba en la cola. El fichero *sender.py* junto con el Dockerfile son los correspondientes a esta aplicación productora. El funcionamiento de esta es muy sencillo, por defecto, una vez se inician los contenedores, manda un mensaje al topic, y en la interfaz de RabbitMQ se puede ver cómo se manda el mensaje en el gráfico, después, esta imagen se para.

Para la comprobación de que el mensaje ha sido enviado correctamente, en el apartado de conexiones, podemos observar en un pico de más B/s en el gráfico:



Repositorio GitHub

https://github.com/UnaiSolaun/AS_RabbitMQ