Taller de de modularización con virtualización e Introducción a Docker y a AWS

Carlos Manuel Murillo Ibañez

Luis Daniel Benavides Navarro
Arquitecturas Empresariales

Escuela Colombiana de Ingeniería Julio Garavito
21 de Septiembre del 2020
Bogotá D.C.

Índice

1. Resumen

En este artículo se expondrá la teoría y la forma en la que se implementó una arquitectura la cual consiste en un balanceador de carga en donde se usó el método de RoundRobin este balanceador realiza peticiones a cada uno de los tres nodos disponibles dependiendo del turno, los cuales se encargan de realizar la conexión con la tercera parte de la arquitectura, una base de datos mongo que almacena todos los mensajes que entran. Cada una de las partes de esta arquitectura se encuentran en un contenedor de Docker diferente.

2. Introducción

En este laboratorio se tiene como objetivo principal desarrollar una arquitectura que sea capaz de realizar consultas e insertar datos en una base de datos mongo usando el framework Spark y contenedores Docker. Para tener mayor claridad del tema en primera parte se realizará una explicación de la teoría que se necesitó para el desarrollo de esta arquitectura, en segunda parte se explicara como esta estructurada y como funciona esta arquitectura y para terminar en la tercera parte se presentaran las pruebas realizadas para confirmar el correcto funcionamiento de la arquitectura.

3. Teoria

- Contenedores: Dentro de ellos podemos alojar todas las dependencias que nuestra aplicación necesite para ser ejecutada: empezando por el propio código, las librerías del sistema, el entorno de ejecución o cualquier tipo de configuración. [1]
- MongoDB: Es una base de datos almacena datos en documentos flexibles similares a JSON, por lo que los campos pueden variar entre documentos y la estructura de datos puede cambiarse con el tiempo.[2]
- Round Robin: Es uno de los algoritmos más antiguos, sencillos y equitativos en el reparto de la CPU entre los procesos lo que significa que evita la monopolización de uso de la CPU, y es muy válido para entornos de tiempo compartido.[3]
- Balanceador de carga: Distribuye el tráfico entrante de aplicaciones entre varias instancias en varias zonas de disponibilidad. Esto aumenta la tolerancia a errores de las aplicaciones.[4]

4. Arquitectura

En esta sección se explicará el diseño que se tuvo en cuenta para el desarrollo de esta arquitectura, donde se usó Docker para crear un contenedor para el load balancer, tres contenedores para los log service y otro para tener una instancia de una base de datos mongo.

4.1. Load Balancer Round Robin

La implementación del load balancer consta de dos partes una de ellas es una fachada hecha en HTML que tiene un espacio para ingresar el mensaje y un botón para inviar ese mensaje, y la otra partes es un servicio REST donde son recibidos las peticiones GET y POST con la ayuda de funciones lambda y dependiendo del tipo de petición se procesa y se dirigen a la clase HttpClient en donde se utilizó el método de Round Robin para poder distribuir las peticiones que se le realizan y hacer la comunicación con los tres diferentes LogService.Como podemos ver en la imagen 1 se crea una sola instancia que va a estar almacenada en un solo contenedor de Docker.

4.2. Log Service

El log service consta de tres partes la primera es la conexión con la base de datos mongo donde se le realizan inserciones y consultas de la información que tiene almacenada, como segunda parte tenemos el modelo de la información que se consulta en la base de datos para que una vez sea extraída de ella pueda ser mapeada a objetos de java y como tercera y ultima parte esta el controlador donde se reciben las peticiones GET y POST enviadas por el load balancer. Como podemos ver en la imagen se crean tres instancias de LogService y cada una va a estar almacenada en un contenedor de Docker para un total de tres contenedores con esta imagen 1.

4.3. MongoDB

En la base de datos se almacenan los mensajes ingresados por el usuario en donde se van a conectar los tres LogService. Como podemos ver en la imagen 1 se crea una sola instancia que va a estar almacena en un contenedor de Docker.

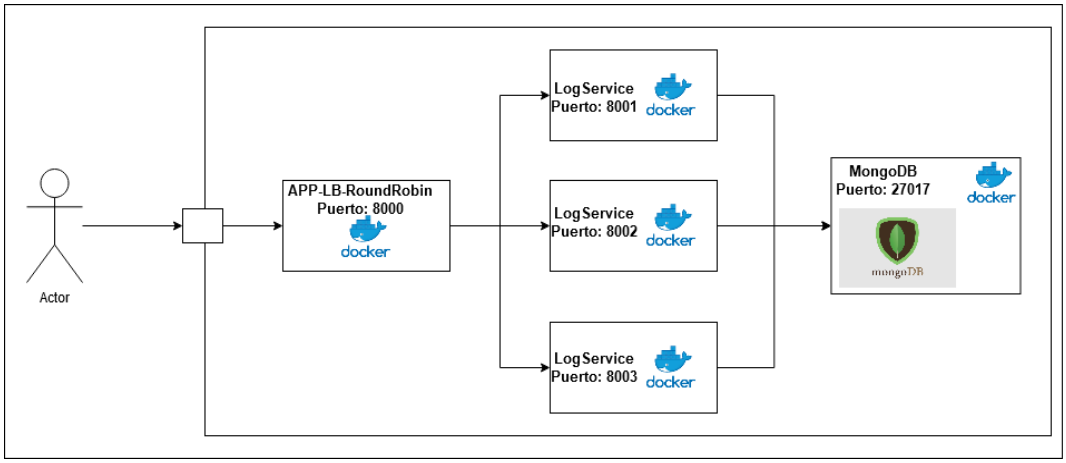

Figura 1: Diagrama.

5. Pruebas

Para asegurar la efectividad y el correcto funcionamiento de la arquitectura desarrollada se hicieron una serie de pruebas.

Figura 2: Ingreso de datos.

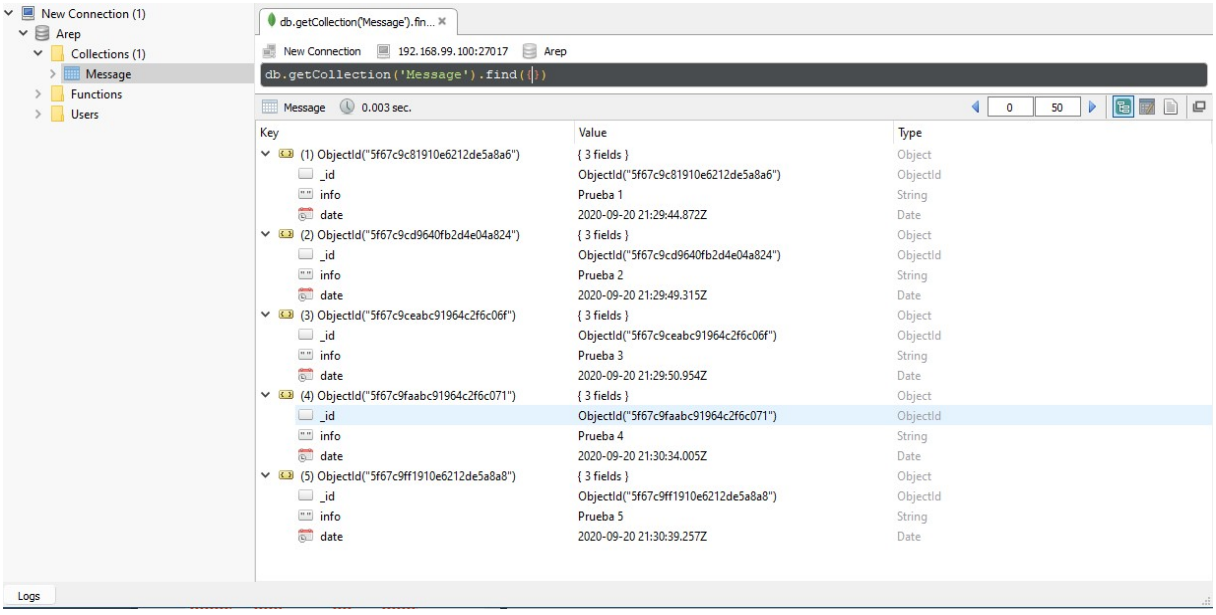

Figura 3: Información almacenada en la base de datos.

6. Conclusiones

Gracias a este laboratorio se logró entender el funcionamiento de los contenedores de Docker de como usarlos y administrarlos de manera eficiente, también se aprendió como usar estos contenedores en una instancia de una máquina creada en AWS.

Referencias

[1] Contendores. https://www.xataka.com/otros/docker-a-kubernetes-entendiendo-que-contenedores-que-mayores-revoluciones-industria-desarrollo

[2] Mongodb. https://www.mongodb.com/es/what-is-mongodb

[3] RoundRobin. https://www.udg.co.cu/cmap/sistemas_operativos/planificacion_cpu/round_robin/Round%20Robin.html

[4] Balanceador de carga. https://docs.aws.amazon.com/es_es/elasticloadbalancing/latest/classic/introduction.html