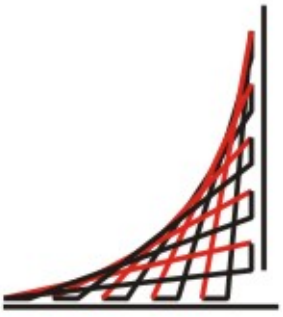

ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO

IMPLEMENTACIÓN DE UN WEVIDOR WEB CON DOCKER EN AWS

LUIS DANIEL BENAVIDES NAVARRO

Arquitecturas Empresariales

AUTOR:
JOHANN SEBASTIAN PÁEZ CAMPOS

SEPTIEMBRE 21 DEL 2020

Tabla de Contenidos

1. Introducción

El objetivo de este laboratorio es implementar un aplicativo web usando el framework de Spark para java, con la cual se construirá un contenedor docker para la aplicación y se configurará para nuestra máquina local. Posteriormente crearemos un repositorio en DockerHub y subiremos la imagen al repositorio. Finalmente crearemos una máquina virtual en AWS, instalaremos Docker y desplegaremos el contenedor creado anteriormente para poder utilizar el aplicativo web.

2. Definiciones

Socket: Un socket es conocido como un tipo de software que actúa como un punto final que funciona estableciendo un enlace de comunicación de red bidireccional entre el extremo del servidor y el programa receptor del cliente. [1]

Servidor Web: El servidor web (también llamado webserver en inglés) es el software que se encarga de despachar el contenido de un sitio web al usuario. [2]

Get: El método GET solicita un recurso del servidor indicado en el campo URI. Si la URI apunta a una base de datos de producción de recursos como un servlet, los datos serán devueltos dentro del mensaje de respuesta. [3]

Post: El método HTTP POST envía datos al servidor. El tipo del cuerpo de la solicitud es indicada por la cabecera Content-Type. [4]

Docker: El software de TI, es una tecnología de creación de contenedores que permite la creación y el uso de contenedores de Linux. [5]

MongoDB: Es una base de datos de documentos que ofrece una gran escalabilidad y flexibilidad, y un modelo de consultas e indexación avanzado.

3. Diseño y Arquitectura

El aplicativo web cuenta con un balanceador de carga y tres nodos, el balanceador se encargará de enviar las peticiones a los distintos nodos mediante el algoritmo de balanceo de cargas de Round Robin. Estos nodos se conectarán a una base de datos mongo para poder obtener los datos y retornarlos.

Cabe destacar que toda esta arquitectura está en contenedores de docker y se comunican con la base de datos mongo que fue creada a partir de una imagen base de docker.

Para una mejor implementación, se desplegó el aplicativo en una máquina virtual EC2 en AWS, utilizando como referencias las imagenes de los repositorios creados en DockerHub.

En la figura 1 se muestra la arquitectura descrita anteriormente.

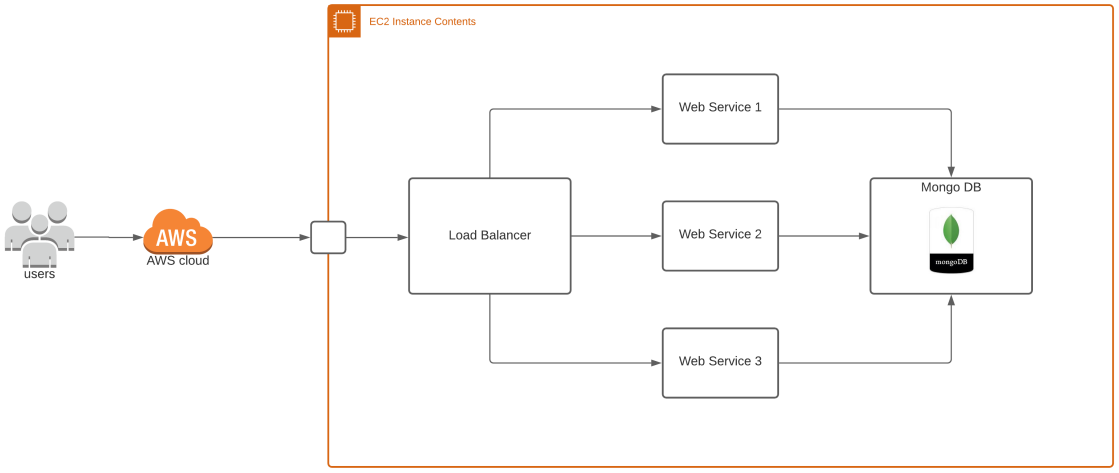

Figura 1: Representación de la arquitectura descrita anteriormente.

4. Pruebas

Verificando que el aplicativo web funciona correctamente, procedemos a mostrar algunos archivos principales como el index.html para probar que la solicitud get funciona correctamente al consultar en la base de datos y añadir un nuevo dato para probar que la peticion post añade correctamente los datos.

4.1. Localmente

Se procederá a mostrar las pruebas locales que se hicieron al aplicativo web.

En ellas encontraremos distintas imágenes con la respectiva descripción de cada prueba y el resultado obtenido.

En la figura 2 se puede evidenciar el aplicativo funcionando correctamente sobre le puerto 10000 y el contenido de la base de datos.

Figura 2: Aplicativo web funcionando correctamente en el puerto 10000 y contenido de los mensajes de la base de datos Mongo.

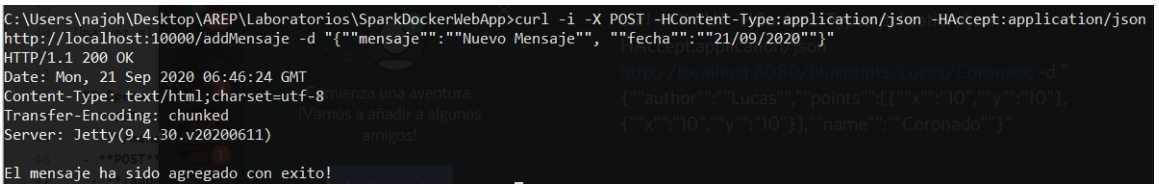

Figura 3: Petición Post al servidor web usando Curl para añadir un nuevo mensaje a la base de datos.

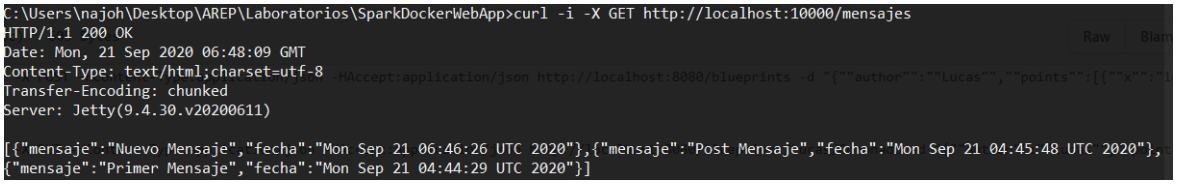

Figura 4: Petición Get al servidor web usando Curl para ver todos los mensajes de la base de datos.

```
Request enviado a nodo http://172.17.0.1:10001/mensajes
Request enviado a nodo http://172.17.0.1:10002/mensajes
Sending POST to node: http://172.17.0.1:10002/addMensaje
Request enviado a nodo http://172.17.0.1:10001/mensajes
Request enviado a nodo http://172.17.0.1:10002/mensajes
Request enviado a nodo http://172.17.0.1:10003/mensajes
```

Figura 5: Logs del balanceador de carga al realizar las anteriores operaciones (se puede evidenciar el balanceo de cargas de Round Robin, delegando el procesamiento del mensaje).

4.2. AWS

Se procederá a mostrar las pruebas que se hicieron al aplicativo web sobre AWS.

En ellas encontraremos distintas imágenes con la respectiva descripción de cada prueba y el resultado obtenido.

En la figura 6 se puede evidenciar el aplicativo funcionando correctamente sobre le puerto 10000 y el contenido de la base de datos.

Figura 6: Aplicativo web funcionando correctamente en AWS en el puerto 10000 y contenido de los mensajes de la base de datos Mongo.

```
[ec2-user@ip-172-31-24-39 ~]$ docker logs --details AREP-LOAD-BALANCER
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Request enviado a nodo http://172.17.0.1:10001/mensajes
Sending POST to node: http://172.17.0.1:10002/addMensaje
Request enviado a nodo http://172.17.0.1:10002/mensajes
Request enviado a nodo http://172.17.0.1:10001/mensajes
Sending POST to node: http://172.17.0.1:10002/addMensaje
Request enviado a nodo http://172.17.0.1:10003/mensajes
Request enviado a nodo http://172.17.0.1:10001/mensajes
```

Figura 7: Logs del balanceador de carga de AWS (se puede evidenciar el balanceo de cargas de Round Robin, delegando el procesamiento del mensaje).

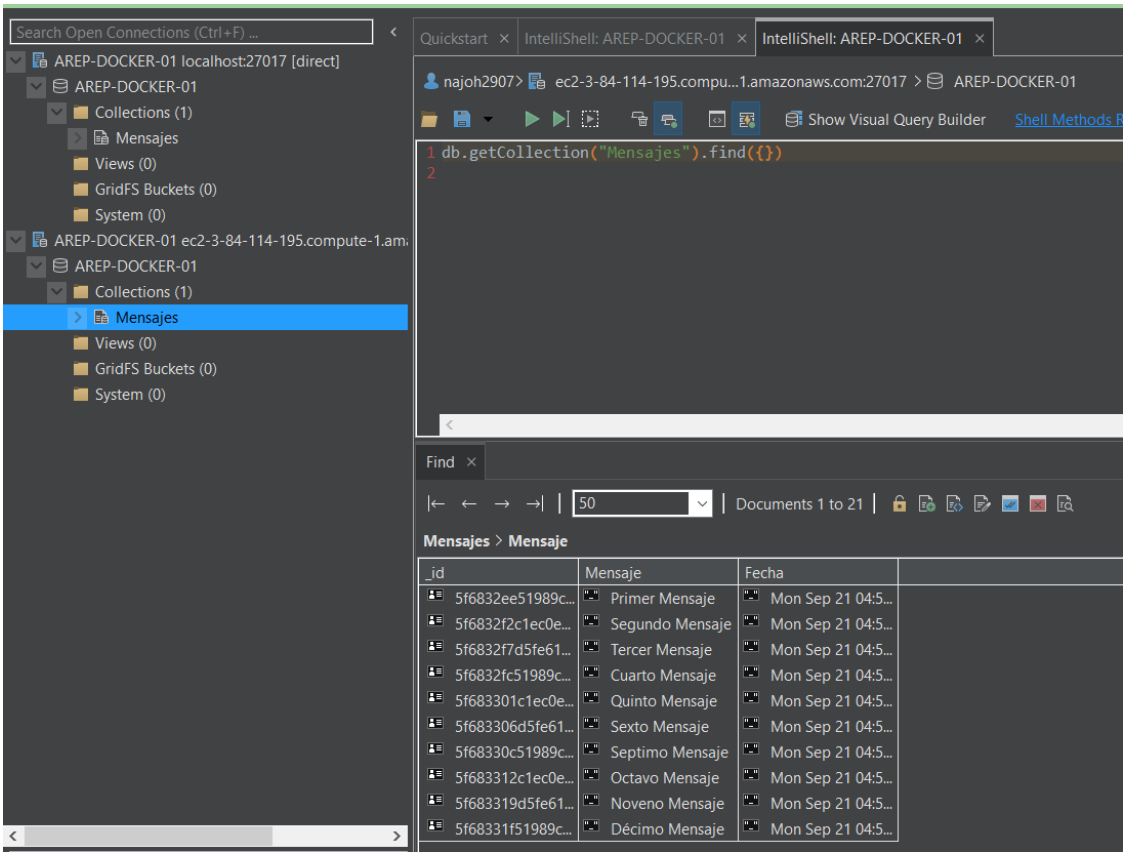

Figura 8: Contenido de la base de datos mediante el cliente Studio 3T.

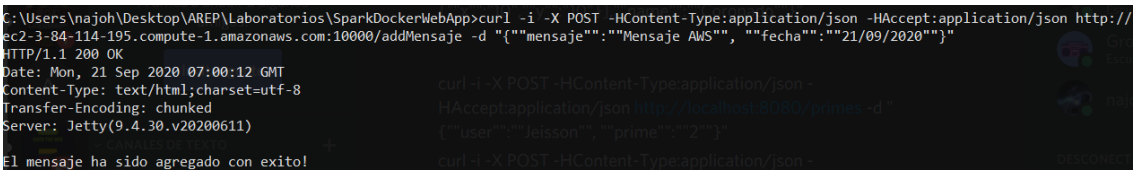

Figura 9: Petición Post al servidor web en AWS usando Curl para añadir un nuevo mensaje a la base de datos.

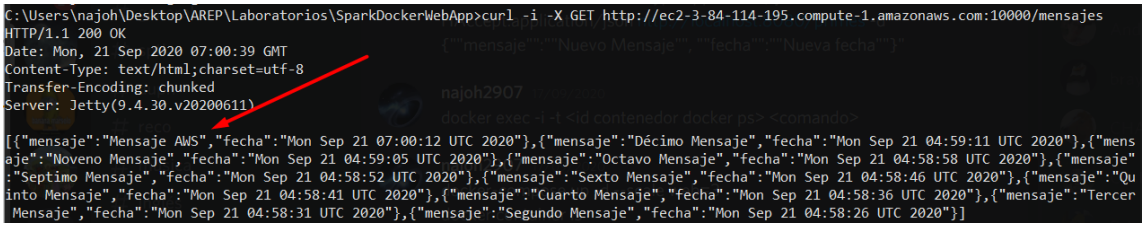

Figura 10: Petición Get al servidor web de AWS usando Curl para comprobar que el mensaje anterior se agrego correctamente.

Figura 11: Contenido del aplicativo web después de realizar la petición Post mediante el curl.

5. Conclusiones

- Con las herramientas que nos ofrece el lenguaje de programación Java, es posible implementar un servidor web con un balanceador que se encargue de enviar las respectivas solicitudes a los nodos.
- Docker es una herramienta mediante la cual podemos configurar proyectos de manera local muy fácil y luego utilizarlos en cualquier máquina que tenga incorporada esta tecnología.
- Mediante las imagenes creadas en DockerHub, se puede desplegar el aplicativo web en una máquina de AWS en menos de 5 minutos, haciendo que esta tecnología cobre un gran valor y nos de alta disponibilidad en caso de fallo.

6. Referencias

[1] *Socket*, `https://www.speedcheck.org/es/wiki/socket/#fn1`, Accessed on 2020-09-03.
[2] *Servidor Web*, `https://blog.infranetworking.com/servidor-web/`, Accessed on 2020-09-03.
[3] *HTTP Request / Response*, `https://sites.google.com/site/conceptoprogramacion/request-response`, Accessed on 2020-09-03.
[4] *POST*, `https://developer.mozilla.org/es/docs/Web/HTTP/Methods/POST`, Accessed on 2020-09-03.
[5] *¿Qué es DOCKER?*, `https://www.redhat.com/es/topics/containers/what-is-docker`, Accessed on 2020-09-21.