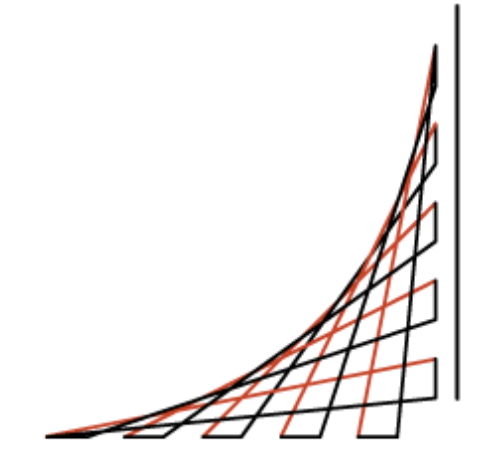
Taller 5 Arem

Fernando Barrera Barrera

Luis Daniel Benavides Navarro

Arquitecturas Empresariales



ESCUELA COLOMBIANA DE INGENIERÍA
JULIO GARAVITO

Índice

1. Introduccion

Este taller fue hecho para comprender la gran utilidad de los dockers ,ademas entender el funcionamiento de los balanceadores round robin y como estos distribuyen equitativamente las cargas de trabajo,como tambien entender como se puede montar una arquitectura de servicios contenidos en dokers y desplegarlos en un ambiente cloud en este caso Aws.

2. Conceptos Basicos

- **Maven:** Herramienta de sofware dedicada a la estructutracion y construccion de proyectos java. [4]
- **Git:** sofware de control de versiones de proyectos [2]
- **Java :** es un lenguaje de programacion orientado a objetos que se desarrollo en los años 90
- **Docker:** Es un contenedor que permite empaquetar todos los recursos necesarios para el despliegue de una aplicacion en cualquier entorno [3]
- **EC2 AWS:** Es una maquina virtual que o recurso de infraestructura virtualizada que ofrece AWS [1]
- **Mongo DB :** Una base de datos no relacional que usa documentos JSON para el almacenmiento datos [5]

3. Diseño

En este taller cuenta con tres log services,cada uno almacenados en dockers independientes en los puertos 8001,8002y 8003 y estos log services se encargan de insertar los nuevos logs y consulta los logs registrados en la base de datos que se encuentra en otro docker en el puerto 27017.

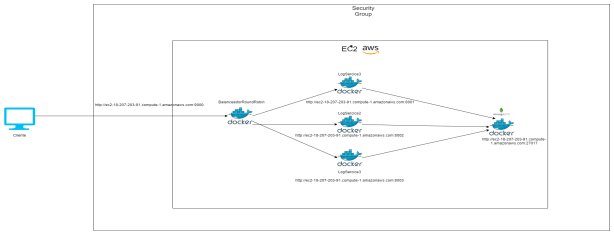


Figura 1:

Los log services se encargan de seleccionar los ultimos 10 logs insertados y se los retorna al docker que contiene el balancedador round robin en el peuerto 9000 y este balancedor se encarga de balancear las cargas de trabajo de los log services equitativamente , ya que despues de cada operacion ya sea de consulta o de insercion cambia a otro log service y asi sucesivamente va enviadoles peticiones a los tres log services y por ultimo se llevaron los 5 dockers a una instancia ec2 aws donde se evidencia su correcto funcionamiento y se verifca que los puertos donde se despliega cada docker este abierto en el security group de la instancia ec2.

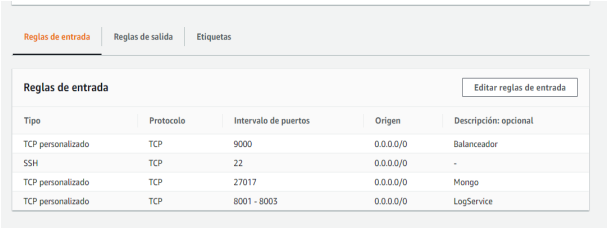


Figura 2:

4. Pre-Requisitos

- Git
- Java
- Maven
- Docker

5. Montaje en Intancia EC2 AWS

Para realizar el montaje de nuestra arquitectura docker en aws primero accedemos a nuestra intancia ec2 por medio de ssh que ya debe tener el puerto 8001,8002,8003,9000 y 27017 abiertos y primero descargamos la imagen de mongo para crear el docker

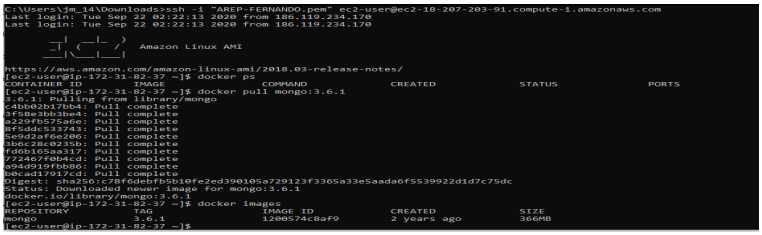


Figura 3:

Luego procedemos a crear la docker con una instancia de la base de datos mongoDB y el docker se llamara db y se ejecutara por el puerto 27017

Figura 4:

Una vez el docker con la instancia de la base de datos mongoDB este ejecutando entramos al cliente mongo para crear la base de datos arem y la collecion logs que sera donde los log services realizaran consultas e inserciones

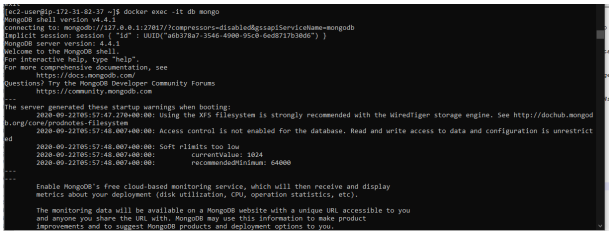


Figura 5:

Ahora vamos a desplegar el Docker del balanceador haciendo uso de la imagen fernando15/balanceadorroundrobin y lo pondremos a ejecutar en el puerto 9000 con el comando docker run

Figura 6:

Ahora vamos a desplegar el docker con el primer log service haciendo uso de la imagen fernando15/logservice y lo llameremos log service1 y lo pondremos a ejecutar en el puerto 8001 con el comando docker run

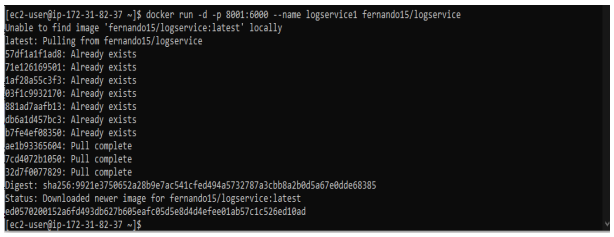


Figura 7:

Para los dockers del log service 3 y 2 se hara tambien ncon el comando docker run en los puertos 8002 y 8003 respectivamente,con la imagen fernando15/logservice que se descargo para la creacion del docker del log service 1

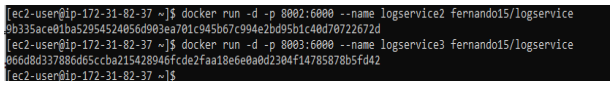


Figura 8:

Ahora con el comando docker ps podemos ver que ya tenemos nuestros 5 dockers con sus respectivos servicios corrriendo en la instancia ec2 de aws

Figura 9:

6. Ejecucion y Pruebas

Para ejecutar probar el funcionamineto de los servicios contenidos en dockers en la instacia ec2 simplemente accedemos con la direccion ip de la instancia ec2 y el puerto 9000 que corresponde al servicio de balanceador que recibe los datos y balancea la cargas de trabajo

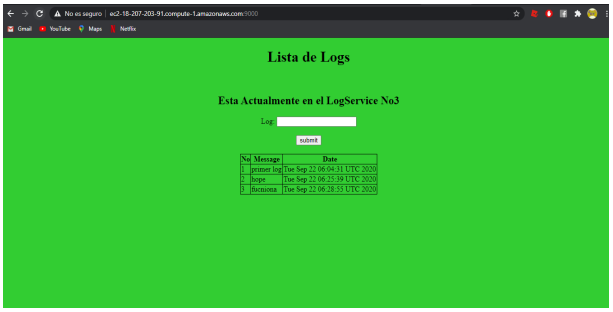


Figura 10:

Al entrar podemos ver que ya hay tres logs en la base de datos asi que vamos a insertar mas hasta llegar a la insercion del log numero 11 asi se vera la tabla de logs

Figura 11:

Ahora al insertar el log 11 se vera que este sera agregado al final de la tabla y primer log desparecera ya que los log service estan programados solo para retornar los 10 ultimos logs registrados en la base de datos como se vera a continuacion

Figura 12:

Podemos tambien acceder a un log service con la ip de la instancia ec2 y el puerto 8001 que corresponde al puerto del log service 1 y vemos que este lo que le retorna al balanceador es un archivo json con string separados por comas que corresponden a los menajes y fechas de los 10 ultimos log registrados en la base de datos

Figura 13:

Tambien podemos ver todos los logs registrados en la base de datos mongoDB accediendo al cliente y verficando los datos insertados en la coleccion logs de la base de datos arem

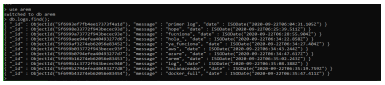


Figura 14:

Para probar el correcto funcionameinto del balanaceadro en la pagina de entrada hay una linea de texto que nos dice en que log service estamos actualmente es decir ellog service que realiazo la ultima operacion en este caso de consulta vemos que estamos en log service 3 es decir ellogservice 3 fue el que hizo la consulta a la base datos y se lo retorno al balanceador

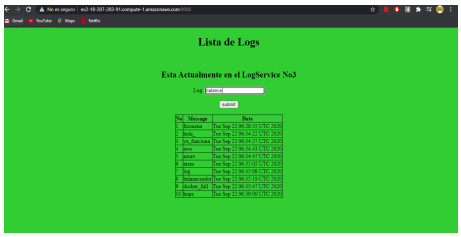


Figura 15:

Ahora vamos a insertar un nuevo mensaje y dado que el logservice3 fue el que hizo la ultima operacion ,el logservice 1 se encargara de adicionar el nuevo registro a la base de datos y el logservice2 se encargara de realizar otra consulta para actualizar los datos de la tabla con el nuevo registro por esa razon despues de la insercion, el balanceador nos indica que nos encontramos en el logservice2 que fuel que realizo la ultima operacion

Figura 16:

7. Conclusion

Este taller fue util para comprender la gran utilidad de los dockers y como podemos montar una arquitectura basica de servicios usando dockers como tambien comprender la importancia del balanceador de cargas para distribuir las cargas de trabajo de forma mas equitativa y se mejore la optimizacion del flujo de operaciones del flujo de trabajo de la arquitectura de servicios.

8. Bibliografía

[1] AWS. *EC2*. URL: https://aws.amazon.com/es/ec2/instance-types/. (entered: 2020).

[2] Codigo Facilito. *Git*. URL: https://codigofacilito.com/articulos/que-es-git. (entered: 16-08-2015).

[3] techtarget. *Docker*. URL: https://searchdatacenter.techtarget.com/es/definicion/Docker. (entered: 2020).

[4] Wikipedia. *Maven*. URL: https://es.wikipedia.org/wiki/Maven. (entered: 31-03-2020).

[5] wikipedia. *MongoDB*. URL: https://es.wikipedia.org/wiki/MongoDB. (entered: 27-08-2020).