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

Santiago Martínez Martínez

Ingeniería de Sistemas
Noveno Semestre
Escuela Colombiana de Ingeniería Julio Garavito
Bogota DC - Colombia

Glosario

Docker: Es una tecnología para la creación de contenedores.

Contenedores: Son entornos ligeros, que tienen un tiempo de ejecución que proporcionan a las apps las dependencias o aplicaciones que se necesitan para su ejecución.

Docker Compose: Herramienta que simplifica el uso, orquestación de los contenedores mediante archivos YAML para poder realizar tareas y ejecuciones de los contenedores Docker.

Docker Hub: es un servicio como GitHub de registro de repositorios para almacenar contenedores.

Cloud Computing: Servicos ofrecidos para tener acceso remoto a Software, almacenamiento y alojamiento para nuestros servicios.

EC2 amazon: Permite la ceacion de instancias o servidores virtuales en la nube de AWS.

Resumen

Este documento tiene como objetivo explicar al detalle la implementación realizada para poder diseñar y orquestar contenedores docker con el fin de crear una arquitectura proporcionada por el profesor.

Este arquitectura se caracteriza por tener un load balancer round robin, este tiene como fin distribuir la carga al servidor, para esto, con docker creamos 3 instancias, las cuales corren por el puerto 8087, 8088 y 8089, tambien tenemos un servicio web para que el cliente puede almacenar sus mensajes y con esto mostrar la fecha en la que se almaceno.

Para poder cumplir este objetivo, se crearon dos proyectos con maven, el primero llamado taller-cinco es el encargado de realizar

los inserts y selects a la base de datos Mongo, de este proyecto tendremos 3 instancias para cumplir la infraestructura round robin, el segundo proyecto es el encargado de mostrar la interfaz grafica para el cliente, este se comunica mediante con las otra otra aplicacion y es el encargado de pasar la data.

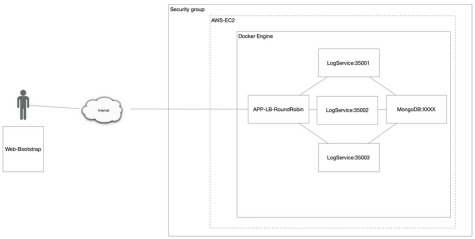
Esta interfaz se realizo usando HTML5, CSS y JavaScript para comunicarnos con nuestro servidor levantado con spark. Por ultimo, se utilizo docker compose para poder realizar y alzar las instancias necesarias y se subio a una servidor en AWS.

Contents

1 Introduction

Primero explicaré las diferencias que existen entre una maquina virtual y un Docker, se puede decir que una maquina virtual es una emulación del sistema operativo de un computador y la emulación de su hardware segun los requisitos y recursos que se asignen. Un contenedor no virtualiza la maquina entera sino que virtualiza partes necesarias del sistema operativo, quiere decir esto que cada contenedor es mas ligero y mas simple que una maquina virtual. Cada contenedero tienen sus librerias y binarias dentro de el, esto nos ayuda a evitar que se le cargue contenido que no utilizará a un contenedor.

Los contenedores es la herramienta principal que nos ayudó a realizar al arquitectura que se deseaba, esto porque ya no tenemos que tener multiples maquinas virtuales corriendo corriendo una instancia de la aplicación, sino que gracias a docker y su red virtual solo lanzamos 3 contenedores con sus respectivos puerto para poder suplir esta necesidad, con esto creamos una arquitectura mas simple y ligera.



2 Objetivos

Este taller tiene como objetivo aprender como crear y desplegar nuestras aplicaciones utilizando Docker y su herramienta de orquestación la cual es Docker Compose, la cual nos facilita la creación de multiples instancias de contenderes.

Segundo, saber como desplegar nuestra aplicaciones en la nube de AWS haciendo uso del servicio EC2 de amazon web service.

3 Solución Requerimientos

Para poder cumplir con los objetivos, primero se creo el archivo dockerfile para la aplicación taller-cinco, esta aplicación como ya se comento antes, es el encargado de realizar las peticiones a la base de datos con MongoDB, esta es la aplicaciones de la cual se crearan multiples instancias para poder cumplir con la infraestrutura, el dockerfile se compone de adjuntar la version de JDK que utilizaremos, que en nuestro caso es openjdk:8, tambien se le indica que en parte de la instancia de la maquina en AWS se copiaran las dependencias y la carpeta tarjet, en nuestro caso elegimos el directorio usrapp/bin/classes y por ultimo se le indica el comando para poder correr la aplicación.

```
FROM openjdk:8

WORKDIR /usrapp/bin

ENV PORT 6000

COPY /target/classes /usrapp/bin/classes
COPY /target/dependency /usrapp/bin/dependency

CMD ["java","-cp","./classes:./dependency/*","edu.escuelaing.arep.app.Spark.SparkServer"]
```

Esto se realizo para los dos proyecto (taller-cinco y load-balancer) y se creo el archivo docker-compose.yml, en donde se le asignan los puertos 8087, 8088 y 8089 a la aplicación taller-cinco identificada como loadbalancer, y se le asigna el puerto 8080 a la aplicacion loadbalancer que es la encargada de mostrar la interfaz grafica al usuario. tambien se crear una tarea para levantar la instancia de la base de datos MongoDB, para esto tenemos que crear un archivo init con el cual inicializamos las credenciales de la base de datos.

4 EC2

Para poder subir nuestro proyecto, lo que se realizo fue subir 2 imagenes, la primera que contenia la aplicación taller-cinco y la segunda la cual contenia la aplicación load-balancer, ademas de esto, se descargo desde git la carpeta dockercompose en la cual tenemos el archivo docker-compose.yml y el archivo init-mongo.js, para que la aplicación funcionara se aprovisiono la instancia con git, docker y docker-compose, luego de esto, lo unico que quedaba era correr nuestro docker compose con el comando dockercompose -d up. Ahora nos dirigimos al dominio de nuestra instancia en el puerto 8080

5 Conclusion

1. Me di cuenta lo facil y util que son lso contenedores Docker y como nos pueden ayudar a integrar nuestros servicios y que sirvan en cualquier sistema operativo.

2. Gracias a las instancias de EC2 podemos compartir nuestros servicios y que

sean publicos en internet.

References

[1] *MDV*.
https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos$_g lobales/Promise$

[2] *Requests*.
https://javascript.info/promise-chaining

[3] *Simple HTTP Server*.
https://dzone.com/articles/simple-http-server-in-java

[4] *Tutorial*.
https://es.overleaf.com/learn/latex/Tutorials