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

Julián David Gutierrez Vanegas

September 2020

1. Introducción

En el presente documento explicaremos el diseño e implementación de una aplicación Web, desplegada en AWS, que le permite al usuario ingresar logs en una base de datos, en este caso Mongo, y obtener los últimos 10 logs registrados.

La descripción de nuestro diseño arrancará explicando la parte de almacenamiento de datos. Aquí, además de hacer una explicación breve de nuestra base de datos no relacional Mongo, mostraremos la estructura del subprograma encargado de recibir solicitudes para ingresar y retornar logs.

A continuación, continuaremos con nuestro subprograma encargado de hacer de balanceador. Este programa es el encargado de repartir las solicitudes entrantes entre las distintas instancias de nuestro programa encargado al acceso de datos. Además, este programa cuenta con una página web donde el usuario puede interactuar fácilmente con el programa.

Para finalizar, mostraremos como estos distintos subprogramas son desplegados en AWS, utilizando contenedores Docker para asegurar que sean instancias virtualmente independientes.

2. Objetivo

Implementar y desplegar, mediante el uso de Docker y AWS, un pequeño programa, dividido en subprogramas, que le permita al usuario ingresar nuevos logs a una base de datos, y al mismo tiempo, obtener los últimos 10 logs registrados allí.

3. Marco Teórico

- Docker: Es un tipo de software que permite almacenar código en contenedores que contengan todo lo necesario para su correcta ejecución. Este software permite implementar y ajustar la escala de aplicaciones de una manera rápida y sencilla.

- MongoDB: Es un tipo de base de datos, NoSQL, orientada a documentos. Es decir, en lugar de almacenar datos, esta almacena documentos, los cuales se encuentran generalmente en formato de tipo BSON que son la representación binaria de los documentos tipo JSON.

4. Explicación del diseño

A continuación, presentamos la estructura del programa

4.1. Subprograma para el registro y acceso a datos

Para comenzar, abordaremos un poco cómo los datos son almacenados y cómo es el proceso que se realiza para su inserción y recuperación. Empecemos por cómo los datos son almacenados. En nuestro programa, usamos una base de datos no relacional, o NoSQL, llamada Mongo. Esta base de datos nos permitió almacenar los logs de forma mucho más liviana.

Por otra parte, para poder ingresar y extraer los logs existentes fue necesario crear un subprograma, que, mediante Spark reciba los nuevos logs y retorne los últimos 10 logs registrados, y, mediante un cliente mongodb, puede conectarse a la base de datos para extraer la información o insertarla.

A continuación, se presenta el modelo de este subprograma:

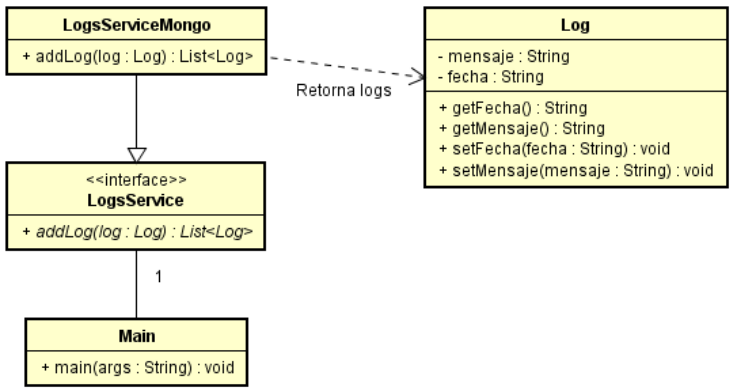

Figura 1: Diagramas de acceso a datos

Este subprograma es flexible, y puede mutar fácilmente a otro tipo de almacenamiento, como una nueva base de datos o formas distintas de acceder a los datos de la existente.

4.2. Balanceador

Para tener un programa mucho más eficiente, que responda a las solicitudes de manera más rápida, se decidió que hubiera varias instancias corriendo en simultaneo del primer subprograma. Pero para esto, era necesario tener un controlador, un programa encargado de recibir todas las solicitudes y distribuirlas entre las diferentes instancias del subprograma de acceso a datos.

Así, creamos este subprograma llamado "balanceador"que cuenta con dos funciones: primero, debe repartir equitativamente las solicitudes entrantes a las instancias existentes, y segundo, debe darle al usuario una página web fácil de manejar, donde pueda ingresar y visualizar los logs existentes.

El algoritmo de balanceo utilizado en este subprograma fue RoundRobin", el cual reparte equitativamente las solicitudes entrantes entre las distintas instancias, sin importar el tamaño o complejidad de estas.

5. Despliegue

El despliegue de este programa se hizo utilizando una máquina virtual ec2 de AWS. Para asegurarnos que el espacio en cual ibamos a desplegar nuestro programa en AWS tuviera todas los programas y librerías necesarios para la correcta ejecución de este, se utilizaron contenedores Docke.

Estos contenedores, los cuales son pequeñas máquinas virtuales, los configuramos para que tuvieran todos los programas librerías y configuraciones que se necesitaban, Esto se hizo mediante especificaciones, ya sea en los archivos de configuarión del programa (pom) o en los archivos de configuración de docker; "DockerFilez "Docker-compose".

El más importante fue "Docker.compose", ya que este nos permitió definir el esqueleto de nuestra aplicación. Allí, especificamos cuántas instancias del primer subprograma ibamos a usar (en este caso usamos 3) y qué direcciones y puertos iban a utilizar. Lo mismo hicimos allí con la base de datos y el balanceador, asegurandonos así que la conección entre todos los componentes se hiciera de manera correcta.

A continuación, se muestra el diagrama de despliegue donde se detallan las ip y los puertos usados por el programa, así cómo la ubicación, tanto física como virtual.

6. Conclusiones

Los contenedores de Docker son ideales para cuande se quieren desplegar aplicaciones que se conforman por distintos componentes. No solo porque nos permiten crear instancias livanas con solo lo que necesitamos, sino que también nos permite realizar la configuración de red de estos equipos, con el objetivo de que después no vaya a haber errores de comunicación entre ellos.

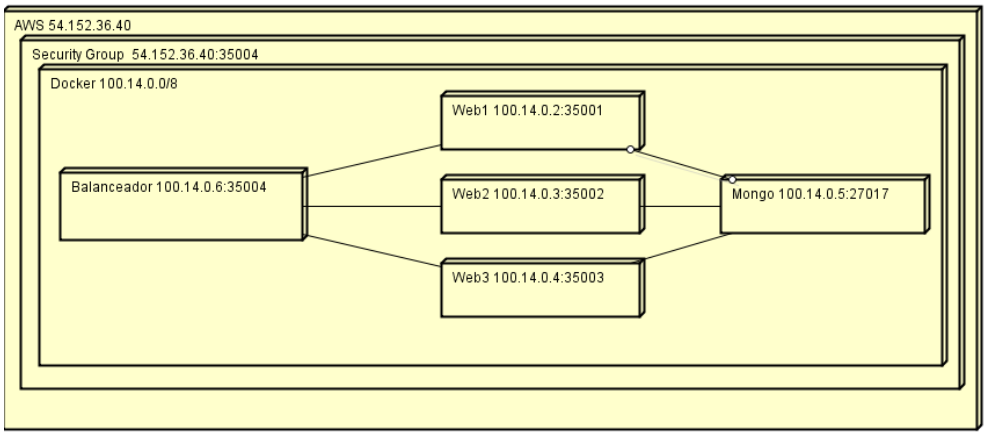

Figura 2: Diagrama de despliegue

7. Bibliografía

- Rubenfa. (2014, 3 febrero). MongoDB: qué es, cómo funciona y cuándo podemos usarlo (o no). Genbeta. https://www.genbeta.com/desarrollo/mongodb-que-es-como-funciona-y-cuando-podemos-usarlo-o-no
- Araujo, J. (2017, 19 mayo). ¿Qué es Docker? ¿Que son los contenedores? y ¿Por que no usar VMs? Platzi. https://platzi.com/contributions/guia-del-curso-de-docker/