Modularización con virtualización e Introducción a Docker y a AWS

Autor: Daniel Felipe Alfonso Bueno[1]

[1]Escuela Colombiana de Ingeniería Julio Garavito
[*]Arquitectura Empresarial Laboratorio Cinco

21 de septiembre de 2020

Resumen

- Modularización: Consiste en la descomposición de un problema general en una serie de subproblemas, buscando reducir la complejidad de un problema grande (Divide y conquista).
- Docker: Simplifica y acelera los flujos de trabajo de desarrollo con un canal de desarrollo integrado y mediante la consolidación de los componentes de la aplicación ofreciendo el encapsulamiento de aplicaciones en contenedores mediante software como docker desktop y con la ayuda de repositorios en dockerhub
- AWS: Es una plataforma de nube que ofrece muchos servicios PaaS, IaaS y SaaS.
- Container: Es una tecnología que permite aislar aplicaciones, empaquetar los archivos y el entorno de ejecución en alguna instancia virtualizada.
- DockerHub: Es un repositorio para el almacenamiento de imagenes de docker, para facilitar la ejecución de estos contendores.

1. Introducción

En la actualidad y con la evolución de las tecnologías las empresas optan por la virtualización de aplicaciones y despliegue de estas en la nube por la simplicidad, escalabilidad, aumento en la velocidad del ciclo de desarrollo, también por la tercerización en temas de seguridad, alta disponibilidad y fácil de gestión de los contenedores en la nube de AWS. En este laboratorio, realizaremos la implementación de una aplicación web con tres módulos separados por containers en docker y desplegados en la nube de AWS. Los cuales son: Base de datos, Servicio REST, Cliente Web

2. Diseño

La arquitectura de la solución es la que se observa en Figura 1

Podemos observar en el diagrama que las instancias de la aplicación, las tendremos en una instancia de Amazon EC2, una máquina virtual que tiene instalado Docker la cual será la encargada de ejecutar y gestionar los containers Docker, Esta máquina tendrá un security group en el cual se tendrán las reglas de los puertos que debemos abrir para la ejecución de los servicios en internet y la conexión a la máquina virtual.

La instancia APP-LB-RoundRobin será la encargada de la implementación del servidor web que recibe las solicitudes del formulario para agregar nuevas palabras y la consulta de las palabras, así mismo de la

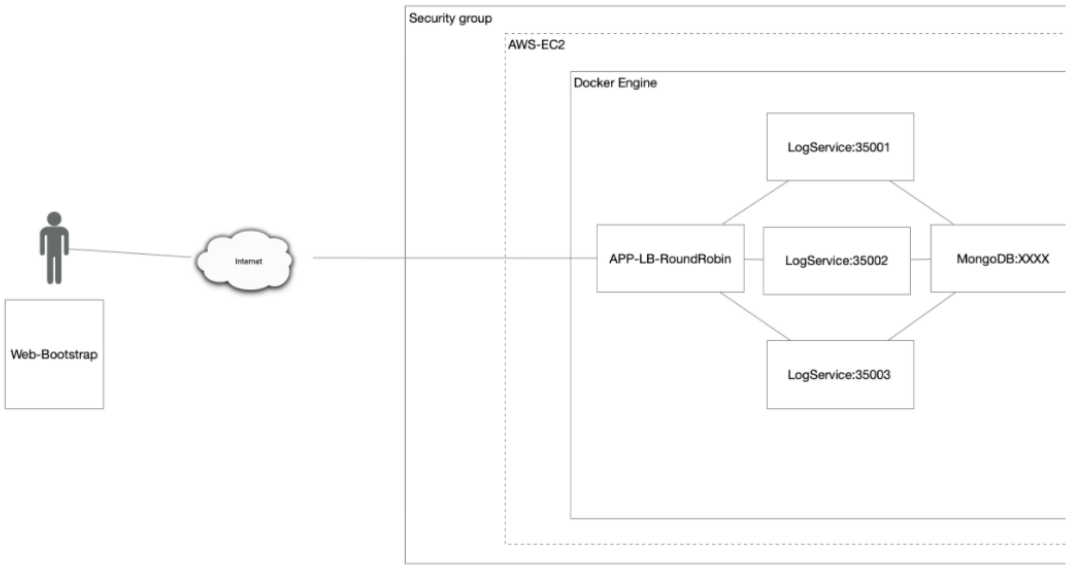

Figura 1: Diagrama de Componentes

implementación del cliente web mediante plantillas de Thymeleaf. Esta instancia usara un balanceador de carga el cual será el encargado de desviar el tráfico a alguna de las tres instancias de LogService que proveerá los servicios REST que consumirá el cliente web los cuales son: Añadir una palabra al hacer click en .ªñadir palabrasz re direccionar a la consulta de palabras agregadas mostrando las 10 últimas, esta instancia será la encargada de añadir las nuevas palabras a la base de datos mediante un APIREST y la que realizara la consulta a la base de datos.

Como último tendremos una instancia de base de datos NO SQL en la cual almacenaremos los datos de nuevas palabras.

Cada instancia estará desplegada en un contenedor de Docker diferente buscando un bajo acoplamiento en la aplicación y puntos de redundancia.

Las imágenes que tenemos configuradas en nuestro repositorio de DockerHub son las siguientes:

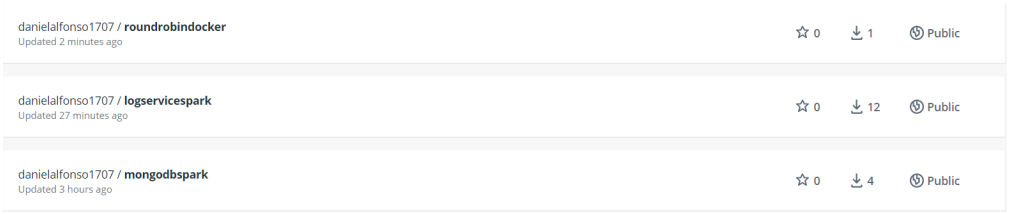

Figura 2: DockerHub Images

Estas imágenes las guardamos y ejecutamos en nuestra máquina virtual de AWS

```
[ec2-user@ip-172-31-17-25 ~]$ docker images
REPOSITORY                          TAG                 IMAGE ID            CREATED             SIZE
danielalfonso1707/roundrobindocker  latest              efd4669d1ace        8 minutes ago       516MB
danielalfonso1707/logservicespark   latest              d73ec03ee1bb        32 minutes ago      518MB
danielalfonso1707/mongodbspark      latest              50e17a9fdd96        4 days ago          492MB
```

Figura 3: AWS EC2 instances images

También ya tenemos corriendo estas imágenes en nuestro Docker de la instancia EC2

```
[ec2-user@ip-172-31-17-25 ~]$ docker ps
CONTAINER ID        IMAGE                                COMMAND                  CREATED             STATUS              PORTS                     NAMES
2031ae4b5083        danielalfonso1707/roundrobindocker   "java -cp ./classes:…"   4 minutes ago       Up 4 minutes        0.0.0.0:35004->6000/tcp   dockerroundrobin
a1d5d0476266        danielalfonso1707/logservicespark    "java -cp ./classes:…"   28 minutes ago      Up 28 minutes       0.0.0.0:35003->6000/tcp   dockerlogservice3
4948645a4e48        danielalfonso1707/logservicespark    "java -cp ./classes:…"   28 minutes ago      Up 28 minutes       0.0.0.0:35002->6000/tcp   dockerlogservice2
b39f8c0cdee9        danielalfonso1707/logservicespark    "java -cp ./classes:…"   28 minutes ago      Up 28 minutes       0.0.0.0:35001->6000/tcp   dockerlogservice
4e87f0934d3b        danielalfonso1707/mongodbspark       "docker-entrypoint.s…"   3 hours ago         Up 3 hours          0.0.0.0:32768->27017/tcp  mongodbspark
```

Figura 4: AWS EC2 instances images

Configuramos las reglas de entrada(Puertos) de la máquina virtual de la siguiente manera

Figura 5: AWS EC2 reglas de entrada

Como resultado obtenemos nuestra aplicación funcional modularizada en contenedores y desplegada en la nube de AWS

Figura 6: AWS Deploy

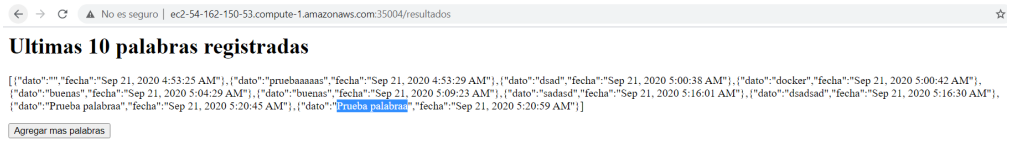

Figura 7: AWS Deploy

3. Conclusiones

- La modularización es una estrategia / técnica fundamental para la realización de aplicaciones granulando problemas grandes en problemas cada vez más pequeños y fáciles de solucionar permitiendo mayor cohesión y facilidad a la hora de entender los componentes y depurar los componentes de una aplicación
- Modularizar la aplicación mediante containers de Docker es una excelente solución para hacer más simples las aplicaciones web, ahorrando tiempo en el desarrollo de estas mismas, el poco espacio que ocupan permitiendo tener varios contenedores en una sola máquina, la alta disponibilidad que podemos generar desplegando varios contenedores o instancias de una sola aplicación utilizando balanceadores de carga. Docker nos facilita la gestión y portabilidad de los contenedores generados, con el plus de repositorios de imágenes de docker que podemos usar para ejecutar nuestras aplicaciones.
- AWS provee un set completo de herramientas y servicios para la administración de aplicaciones e infraestructuras y modelos de nube PaaS, SaaS, IaaS
- La implementación de la aplicación se logró con éxito generando un cliente web que consume un servicio REST que a su vez realiza acciones sobre una base de datos NOSQL en contenedores aislados en una misma máquina virtual EC2 desplegada en la nube de AWS

Referencias

[1] Wikipedia®: *Modularidad (informática).* `https://es.wikipedia.org/wiki/Modularidad_(inform%C3%A1tica)`, junio 2020.

[2] Docker®: *Why Docker?* `https://www.docker.com/why-docker`, 2020.

[3] AWS®: *What is AWS?* `https://aws.amazon.com/es/what-is-aws/`, 2020.