Taller de Modularización con Virtualización e Introducción a Docker y a AWS

Alejandro Toro Daza

Profesor:
Luis Daniel Benavides Navarro

Arquitecturas Empresariales

Escuela Colombiana de Ingeniería Julio Garavito
28 de Febrero del 2021
Bogotá D.C.

TABLA DE CONTENIDO

1. INTRODUCCIÓN

En el Taller de Modularización con Virtualización e Introducción a Docker y a AWS se realizó primero el aprendizaje del manejo de contenedores en Docker, el cual se desplegó la primera instancia en Docker. Después, se realizó el respectivo aprendizaje de AWS para poder con el contenedor creado en Docker, desplegarlo en AWS en la máquina virtual, para así tener base para poder iniciar la tarea la cual consiste en una implementación de una arquitectura la cual consiste en un balanceador de carga en donde se usa el método de RoundRobin. Este balanceador de carga se encarga de realizar las respectivas peticiones a los LogService, los cuales se encargan de realizar la conexión con las bases de datos montada en MongoDB, que almacena todos los mensajes entrantes. En la arquitectura del programa se encuentran en total cinco contenedores, uno para el RoundRobin, tres para el LogService y uno para las bases de datos en Mongo.

2. OBJETIVOS

- Implementar un programa capaz de manejar un balanceador de carga usando el método de RoundRobin en donde se manejen peticiones a LogServices.
- Montar una base de datos en MongoDB que permita almacenar todos los mensajes entrantes.
- Compilar y ejecutar el programa en Docker para su respectivo despliegue en la misma.
- Montar Docker en una máquina virtual utilizando AWS para así instalar Docker dentro de ella y poder desplegar el programa utilizando el servicio de la nube.

3. MARCO TEÓRICO

- **JavaScript:** Lenguaje de programación o de secuencias de comandos que te permite implementar funciones complejas en páginas web, cada vez que una página web hace algo más que sentarse allí y mostrar información estática para que la veas, muestra oportunas actualizaciones de contenido, mapas interactivos, animación de Gráficos 2D/3D, desplazamiento de máquinas reproductoras de vídeo, etc., puedes apostar que probablemente JavaScript está involucrado. Es la tercera capa del pastel de las tecnologías web estándar, dos de las cuales (HTML y CSS) hemos cubierto con mucho más detalle en otras partes del Área de aprendizaje.
- **CSS:** Lenguaje que define la apariencia de un documento escrito en un lenguaje de marcado (por ejemplo, HTML).
  Así, a los elementos de la página web creados con HTML se les dará la apariencia que se desee utilizando CSS: colores, espacios entre elementos, tipos de letra, ... separando de esta forma la estructura de la presentación.
  Esta separación entre la estructura y la presentación es muy importante, ya que permite que sólo cambiando los CSS se modifique completamente el aspecto de una

página web. Esto posibilita, entre otras cosas, que los usuarios puedan usar hojas de estilo personalizadas (como hojas de estilo de alto contraste o de accesibilidad).

- **HTML:** Lenguaje de marcado de hipertexto, y le permite al usuario crear y estructurar secciones, párrafos, encabezados, enlaces y elementos de cita en bloque (blockquotes) para páginas web y aplicaciones.
  HTML no es un lenguaje de programación, lo que significa que no tiene la capacidad de crear una funcionalidad dinámica. En cambio, hace posible organizar y formatear documentos, de manera similar a Microsoft Word.
- **Framework:** Conjuntos de componentes de software que se utilizan para crear la estructura de un software o una aplicación. Un framework puede verse como una caja de herramientas en la que el desarrollador busca los componentes necesarios. Este marco proporciona una estructura general para facilitar la labor de desarrollo.
  Los frameworks funcionan mediante el lenguaje de programación y desarrollan todo tipo de soporte: sitios web, juegos, aplicaciones móviles, etc. Sin embargo, también puede crear su propio marco de software.
- **Round Robin:** Round robin es un método para seleccionar todos los elementos en un grupo de manera equitativa y en un orden racional, normalmente comenzando por el primer elemento de la lista hasta llegar al último y empezando de nuevo desde el primer elemento. El planeamiento Round Robin es tan simple como fácil de implementar, y está libre de inanición.
  El nombre del algoritmo viene del principio de Round-Roubin conocido de otros campos, donde cada persona toma una parte de un algo compartido en cantidades parejas.
  Una forma sencilla de entender el round robin es imaginar una secuencia para "tomar turnos". En operaciones computacionales, un método para ejecutar diferentes procesos de manera concurrente, para la utilización equitativa de los recursos del equipo, es limitando cada proceso a un pequeño periodo de tiempo (quantum), y luego suspendiendo este proceso para dar oportunidad a otro proceso y así sucesivamente. A esto se le denomina comúnmente como Planificación Round-Robin.
- **Docker:** Plataforma de software que le permite crear, probar e implementar aplicaciones rápidamente. Docker empaqueta software en unidades estandarizadas llamadas contenedores que incluyen todo lo necesario para que el software se ejecute, incluidas bibliotecas, herramientas de sistema, código y tiempo de ejecución. Con Docker, puede implementar y ajustar la escala de aplicaciones rápidamente en cualquier entorno con la certeza de saber que su código se ejecutará.
  La ejecución de Docker en AWS les ofrece a desarrolladores y administradores una manera muy confiable y económica de crear, enviar y ejecutar aplicaciones distribuidas en cualquier escala.
  Docker le proporciona una manera estándar de ejecutar su código. Docker es un sistema operativo para contenedores. De manera similar a cómo una máquina

virtual virtualiza (elimina la necesidad de administrar directamente) el hardware del servidor, los contenedores virtualizan el sistema operativo de un servidor. Docker se instala en cada servidor y proporciona comandos sencillos que puede utilizar para crear, iniciar o detener contenedores.

- **AWS:** Es un proveedor de servicios en la nube, nos permite disponer de almacenamiento, recursos de computación, aplicaciones móviles, bases de datos, entre otros.
  Amazon Web Services ofrece herramientas en las siguientes categorías:
  - **Cloud computing:** todo lo necesario para la creación de instancias y el mantenimiento o el escalado de las mismas. Amazon EC2 es el rey indiscutible dentro de los servicios de computación en la nube de Amazon.
  - **Bases de datos:** distintos tipos de bases de datos pueden permanecer en la nube mediante el servicio Amazon RDS, que incluye distintos tipos a elegir MySQL, PosgreSQL, Oracle, SQL Server y Amazon Aurora, o Amazon DynamoDB para NoSQL.
  - **Creación de redes virtuales:** permite la creación de redes privadas virtuales a través de la nube, gracias principalmente al servicio Amazon VPC.
  - **Aplicaciones empresariales:** Amazon WorkMail es el servicio de correo empresarial que ofrece Amazon, al que pueden unirse otros servicios como Amazon WorkDocs y Amazon WorkSpaces.
  - **Almacenamiento y gestores de contenido:** tipos de almacenamiento diferentes, tanto para archivos con acceso regular, poco frecuente o incluso como archivo. Amazon S3 es el servicio principal, aunque complementan la oferta otros como Amazon Glacier o Amazon EBS. En el siguiente vídeo (12:13 min.), puede verse un videotutorial con una definición general de AWS y una explicación del funcionamiento de Amazon S3:
  - **Inteligencia de negocios o Business Intelligence (BI):** sistemas para análisis de datos empresariales a gran escala y otros servicios para la gestión de flujos de datos.
  - **Gestión de aplicaciones móviles:** herramientas como Amazon Mobile Hub permiten la gestión, creación, testeo y mantenimiento de aplicaciones móviles a través de la nube.
  - **Internet de las cosas (Internet of Things, IoT):** para establecer conexiones y análisis de todos los dispositivos conectados a internet y los datos recogidos por los mismos.
  - **Herramientas para desarrolladores:** para almacenar código, implementarlo automáticamente o incluso publicar software mediante un sistema de entrega continua.
  - **Seguridad y control de acceso:** se pueden establecer autenticaciones en varios pasos para poder proteger el acceso a sus sistemas internos, ya estén en la nube o instalados de forma local en sus instalaciones.

4. DISEÑO Y CONSTRUCCIÓN

Para la creación del proyecto, se dispuso de dos programas, **RoundRobin** y **LogService**. Round Robin contiene dos clases en total que se encargan de tanto la ejecución del programa como del cliente HTTP, estas clases son **App** y **HttpClient**. La clase **App** como se mencionó, se encarga de ejecutar el programa, en la cual se encarga de manejar directamente los recursos de la página, que son **/index.html** y **/results**. Se encarga de desplegarlo tanto de manera local en el puerto 4567 como en el contenedor Docker y en AWS. La clase **HttpClient** se encarga de realizar la conexión HTTP con el cliente, utilizando los puertos :8001, :8002 y :8003, y usando la URL 192.168.99.100. Asimismo, se encarga de realizar las peticiones GET y POST a LogService, que es el otro programa dispuesto encargado de realizar todos los logs de los usuarios.
Por otra parte, también se implemento un segundo programa llamado **LogService**, que se encarga de gestionar los logs de los usuarios. Este programa tiene 3 clases que se encargan de gestionar los logs del cliente. La primera clase es la clase **App**, que se encarga de manejar el único regurso **/messages**, el cual también se encarga de recibir peticiones GET y POST. La clase **Message** se encarga de tener control sobre todos los mensajes ingreesados por el cliente o usuario, la cual se encarga también de obtener la fecha y la información del mensaje. La clase **DBConnection** se encarga de realizar la conexión con las bases de datos en MongoDB, la cual contiene el puerto e IP correspondiente, y los datos como lo son la información de los mensajes con sus respectivas fechas.

Diagrama de Clases del programa Round Robin:

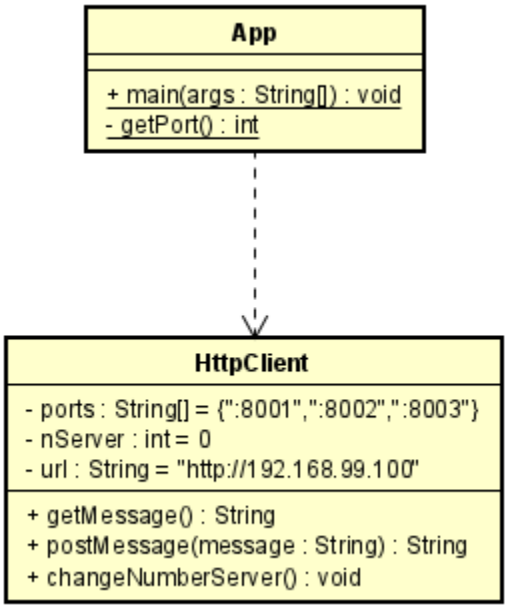
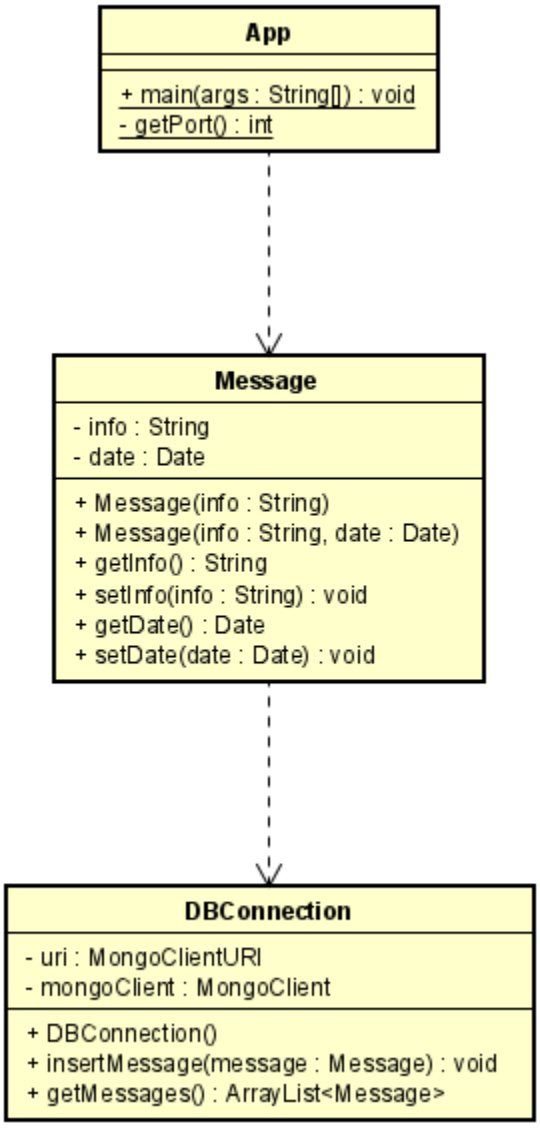


Diagrama de Clases del programa LogService:



4.1. PRUEBAS

Pruebas en Maven

Para verificar que todo el código está funcionando con total normalidad, se implementaron las pruebas en **Junit**, las cuales fueron implementadas para cada uno de los programas **Round Robin** y **LogService**. Para comprobar que primero las pruebas compilaron correctamente en **Round Robin**, se corrió el comando de Maven **mvn test**, el cual arrojó los siguientes resultados, demostrando que las pruebas fueron ejecutadas correctamente y que el código no contiene ningún tipo de error.

```
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running edu.escuelaing.arep.app.AppTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.005 s - in edu.escuelaing.arep.app.AppTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  2.352 s
[INFO] Finished at: 2021-02-25T19:04:33-05:00
[INFO] ------------------------------------------------------------------------
```

Luego, se realizó exactamente el mismo procedimiento, pero esta vez con el programa **LogService**, en el que se ejecutó el mismo comando de Maven **mvn test** y arrojó también resultados exitosos, demostrando también que el código de **LogService** y las pruebas fueron realizadas correctamente.

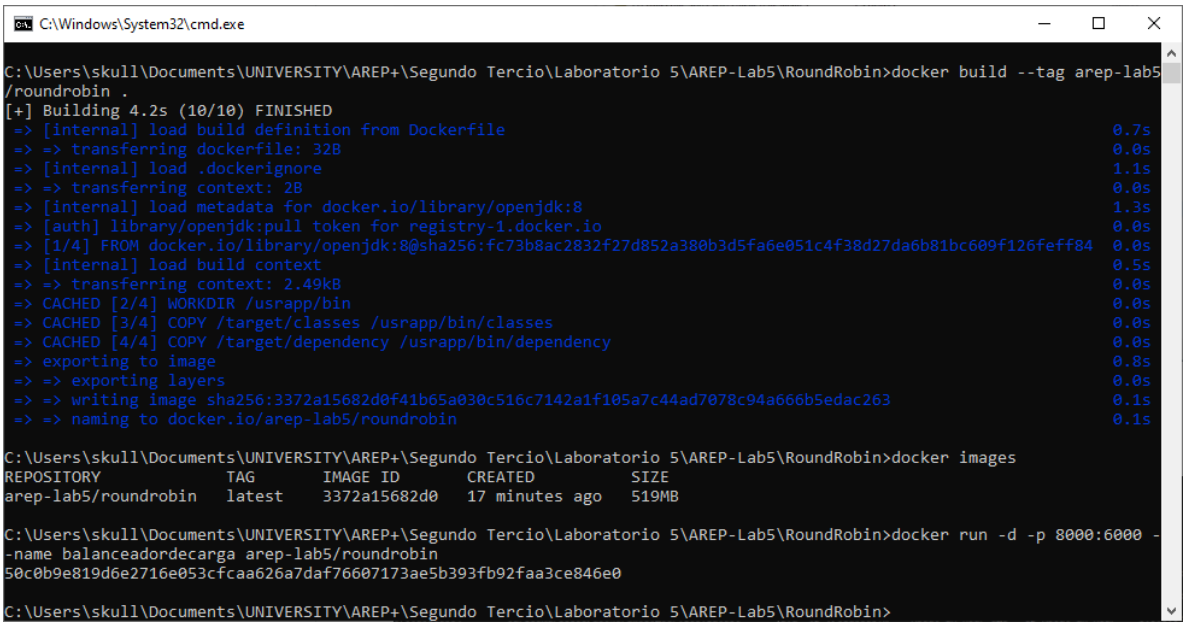
```
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running edu.escuelaing.arep.app.AppTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.025 s - in edu.escuelaing.arep.app.AppTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  2.346 s
[INFO] Finished at: 2021-02-25T20:00:12-05:00
[INFO] ------------------------------------------------------------------------
```

Pruebas en Localhost

Para probar ahora el correcto funcionamiento del Docker de manera local o localhost del programa **RoundRobin**, primero ejecutamos los siguientes comandos en orden.

```
docker build --tag arep-lab5/roundrobin .
docker images
docker run -d -p 8000:6000 --name balanceadordecarga arep-lab5/roundrobin
```

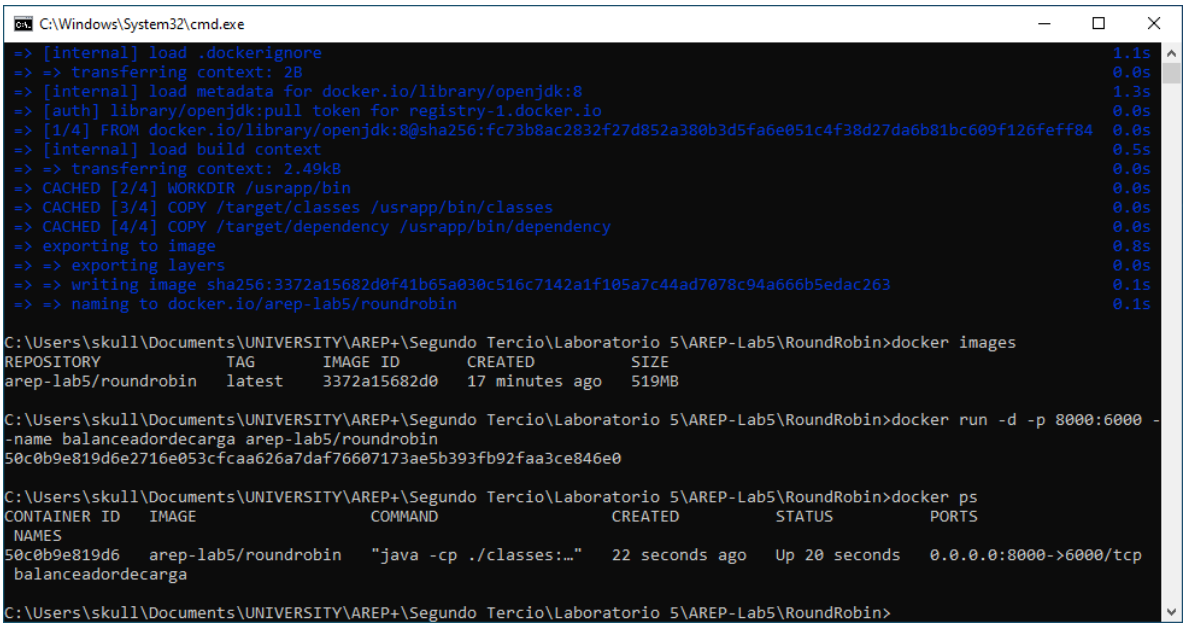
Luego de ejecutarlos en exactamente ese mismo orden, tenemos el siguiente resultado en pantalla.



Ahora, se ejecuta el siguiente comando para comprobar que el Docker de RoundRobin se está ejecutando.

```
docker ps
```

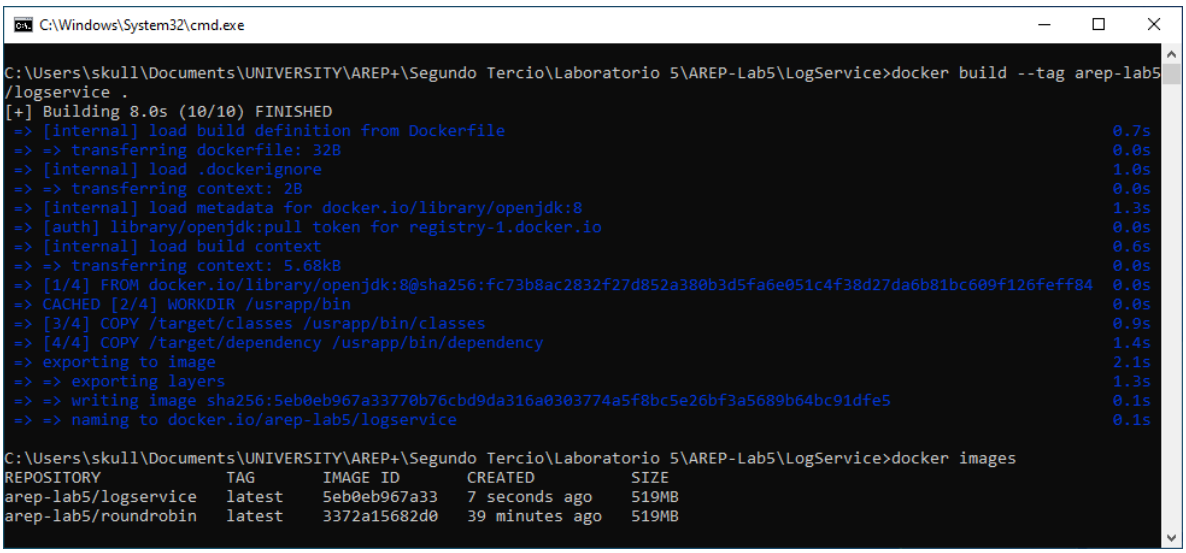
Luego de ejecutar el comando, tenemos el siguiente resultado en pantalla.



Ahora, para probar de manera local o localhost el programa **LogService** con sus tres respectivos logs, ejecutamos los siguientes comandos en orden.

```
docker build --tag arep-lab5/logservice .
docker images
```

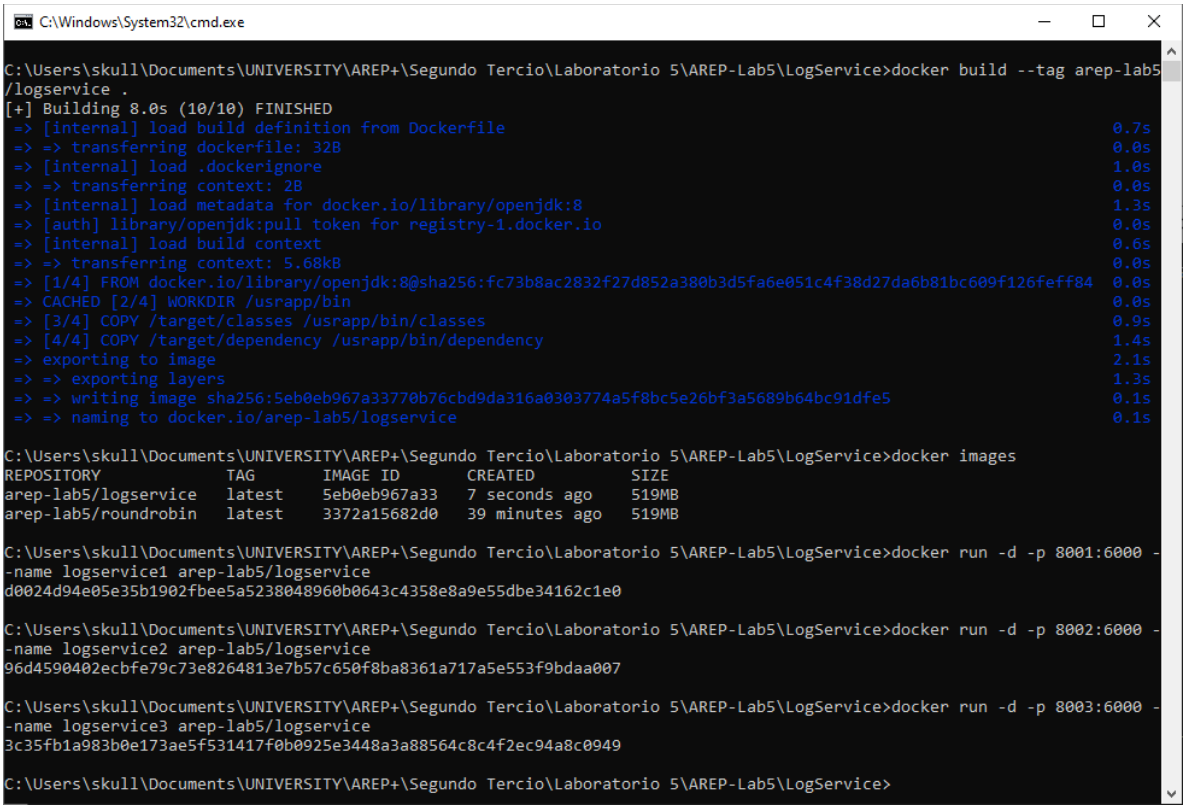
Luego de ejecutarlos en exactamente ese mismo orden, tenemos el siguiente resultado en pantalla.



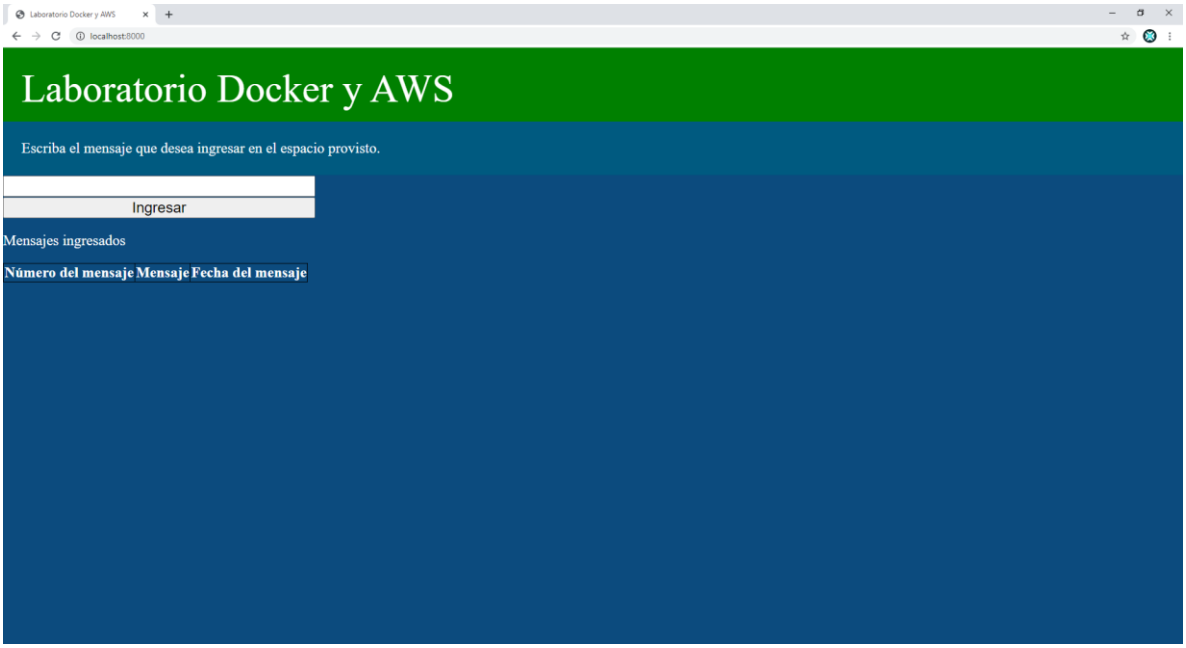
Ahora, para correr los tres logs en puertos diferentes, se ejecutan los siguientes comandos en orden.

```
docker run -d -p 8001:6000 --name logservice1 arep-lab5/logservice
docker run -d -p 8002:6000 --name logservice2 arep-lab5/logservice
docker run -d -p 8003:6000 --name logservice3 arep-lab5/logservice
```

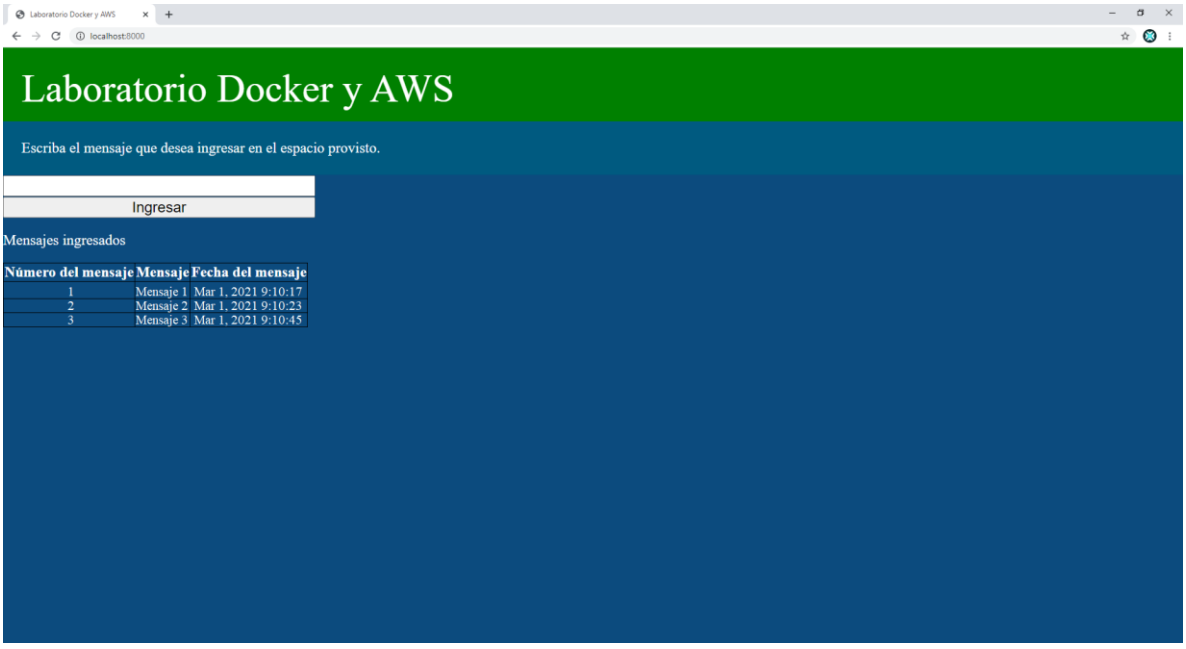
Luego de ejecutarlos en exactamente ese mismo orden, tenemos el siguiente resultado en pantalla.



Para comprobar que la página web ha sido desplegada con éxito, se ingresa en el navegador la siguiente URL: **localhost:8000**. Luego de ingresar la URL en el navegador, se obtiene el siguiente resultado.



Luego de ingresar tres mensajes en el espacio provisto, **Mensaje 1**, **Mensaje 2** y **Mensaje 3** respectivamente, se observa que los mensajes han sido agregados con éxito a las bases de datos en **MongoDB** y a la página de forma estática como se ve a continuación.

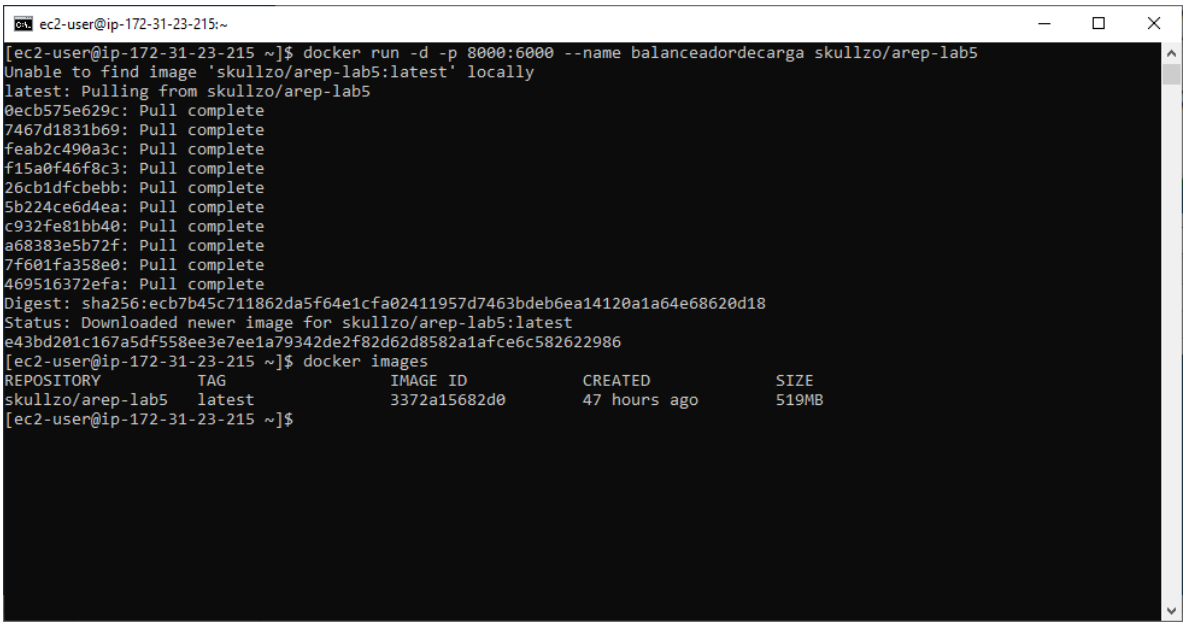


Pruebas en AWS

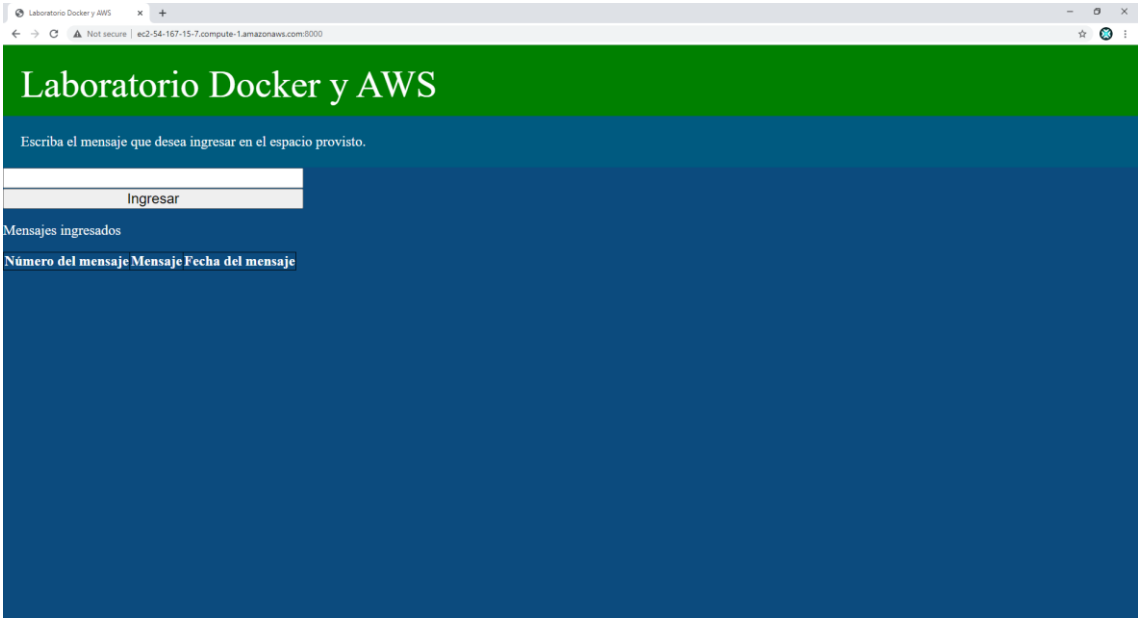
Para probar que el programa funciona correctamente en AWS, luego de realizar todos los pasos correspondientes para poder montar una máquina virtual en AWS con Docker, se procede a correr el balanceador de carga dentro de la máquina virtual de AWS, ejecutando los siguientes comandos.

```
docker run -d -p 8000:6000 --name balanceadordecarga skullzo/arep-lab5
docker images
```

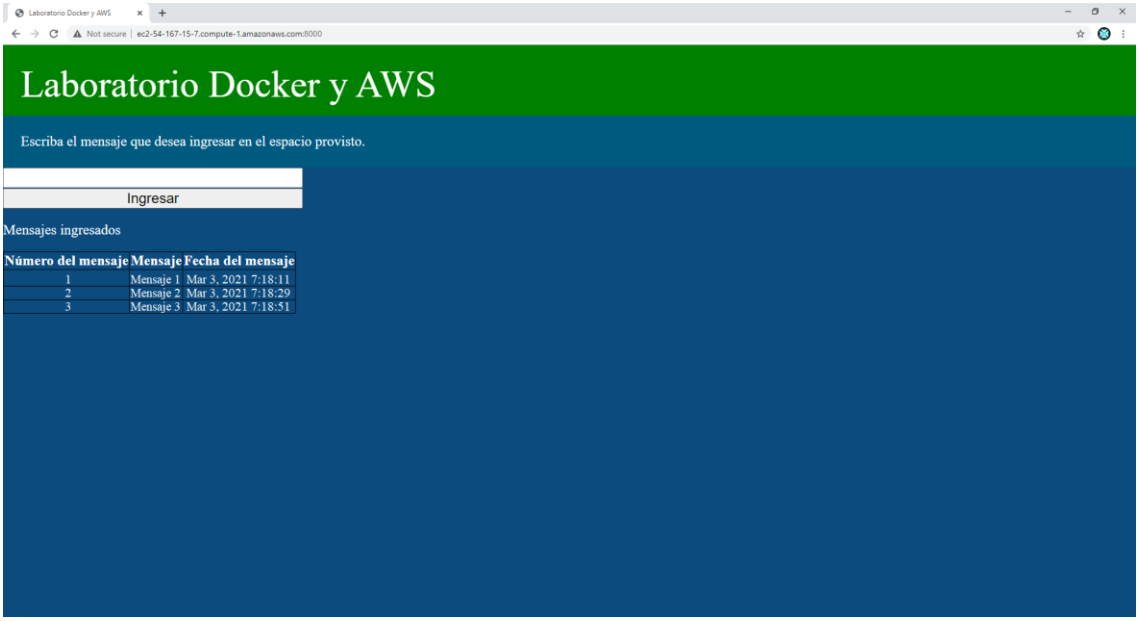
Como se ve a continuación, el Docker ya previamente instalado en la máquina virtual de AWS ha clonado el contenedor almacenado en el respositorio **arep-lab5**, y lo ha ejecutado para así poder correrlo dentro de la máquina virtual, como se muestra a continuación.



Ahora, para comprobar que el contenedor se encuentra activo desde la máquina virtual, ingresamos en el navegador la siguiente URL: http://ec2-54-167-15-7.compute-1.amazonaws.com:8000/. Como se puede observar, el contenedor ha sido desplegado satisfactoriamente desde la máquina virtual montada en AWS.

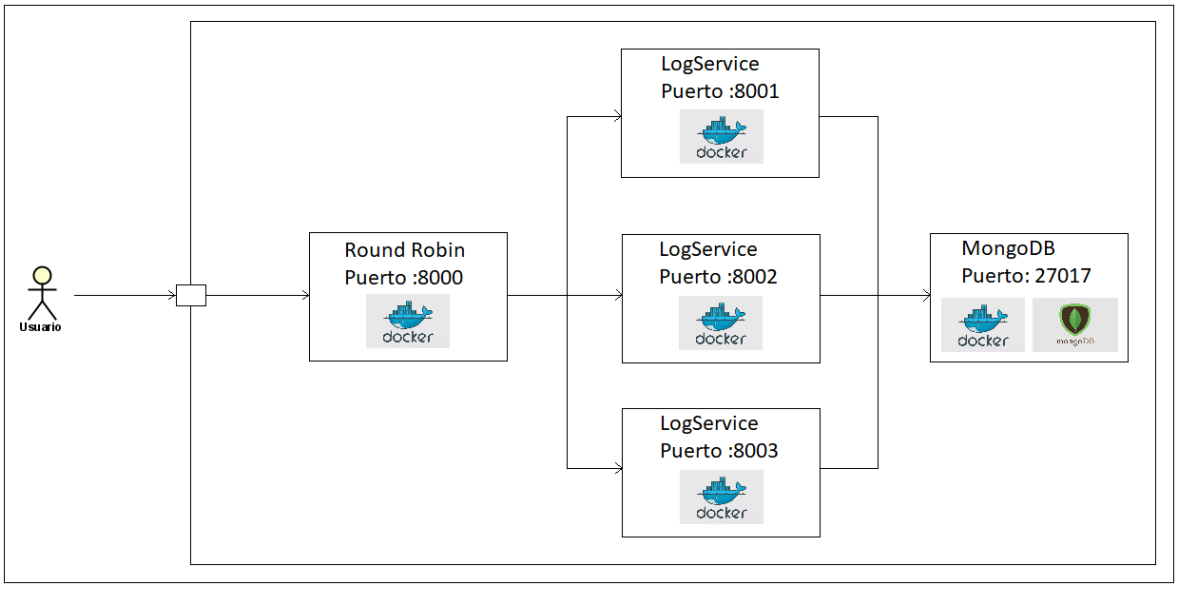


Por último, se ingresan tres mensajes para probar el funcionamiento del programa, que son **Mensaje 1**, **Mensaje 2** y **Mensaje 3** respectivamente. Como se puede observar, el programa funciona perfectamente, agregando los tres mensajes con su respectivo número del mensaje y fecha del mensaje.



5. ARQUITECTURA

- La arquitectura planteada para la realización de todo el programa fue primero planteada para que funcionara de manera local, luego se implementó en el archivo **Docker** todo lo necesario para poder desplegarlo en Docker, y, por último, se realizaron pasos adicionales en AWS para poder también desplegarlo en una máquina virtual en AWS para poder desplegarlo satisfactoriamente.
- El programa Round Robin consta de un balanceador de carga, el cual tiene dos partes, una hecha en HTML que se encarga de ejecutar todo el front, que es la interfaz de usuario en la que el usuario ingresa el mensaje, hecha en HTML y funcional en un servicio REST, donde son recibidas peticiones GET y POST, acompañada de funcione lambda y dependiendo del tipo de petición, esta es procesada y enviada a HttpClient, una clase implementada dentro del programa encargada de realizar el método de Round Robin que distribuye las peticiones realizadas y hacer la respectiva comunicación con los tres LogService.
- LogService se encarga de realizar las conexiones, una de ella es con la base de datos alojada en MongoDB, que se encarga de realizar las inserciones y consultas correspondientes. Por otra parte, se tiene el modelo de la información en la que se consulta la base de datos para que una vez sea extraída de ella pueda ser mapeada por objetos de java como tercera y última parte, en la cual está el controlador donde se reciben las peticiones GET y POST, enviadas por el balanceador de carga correspondiente.
- En MongoDB se almacenan los mensajes ingresados por el usuario, en los cuales se conectan los tres LogService correspondientes. A continuación, se puede ver detalladamente la arquitectura de todo el programa, propuesta en el enunciado e implementada en el programa, con todos sus respectivos puertos y programas correspondientes.



6. CONCLUSIONES

- En el laboratorio se utilizó un programa con los requerimientos preestablecidos en el enunciado propuesto, el cual es capaz de manejar un balanceador de carga utilizando Round Robin, en la que se manejaron tres peticiones de LogServices en tres puertos diferentes correspondientes a cada uno de ellos, el cual fue posteriormente desplegado en Docker y AWS.
- Asimismo, se montó una base de datos en MongoDB que funciona exitosamente, al almacenar todos los mensajes, y realizar la conexión con los LogService para así poder almacenar todos los mensajes y realizar las consultas correspondientes.
- Por otro lado, se implementó integración continua en todo el código fuente para llevar un control de calidad del código, para asegurar que el código esté funcionando totalmente sin ningún problema tanto de compilación como de los resultados retornados después de realizar las respectivas operaciones, y así poder desplegar la aplicación sin presentar ningún tipo de errores.
- Finalmente, luego de implementar todo el código con sus respectivas bases de datos, se pudo realizar el contenedor en Docker del código, el cual fue desplegado en Docker primero para comprobar total funcionalidad en Docker, para así proceder con el despliegue en AWS al instalar Docker dentro de la máquina virtual y así poder desplegar el programa en la nube desde la máquina virtual que provee AWS.

7. REFERENCIAS

- Vargas, Julián. “Aprende Sobre Desarrollo Web.” Aprende Sobre Desarrollo Web | MDN, 23 Junio 2015, developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/Qu%C3%A9_es_JavaScript.
- Delgadillo, Daniel. “CSS, ¿Qué Es?” *Arume*, 1 Abr. 2019, www.arumeinformatica.es/blog/css/.
- Bernal, Gustavo. “¿Qué Es HTML? Explicación De Los Fundamentos.” Tutoriales Hostinger, 10 Dic. 2020, www.hostinger.es/tutoriales/que-es-html/.
- Sánchez, Sebastián. “¿Qué Es Un Framework? - Wild Code School.” Www.wildcodeschool.com, 13 Junio 2019, www.wildcodeschool.com/es-ES/blog/que-es-un-framework.
- Gómez, Carlos. “Planificación Round-Robin.” *Los Diccionarios y Las Enciclopedias Sobre El Académico*, 11 Junio 2010, esacademic.com/dic.nsf/eswiki/937202.
- Emaz, Antoine. “¿Qué Es Docker?” *Amazon*, UAP, 25 Nov. 2016, aws.amazon.com/es/docker/.
- Rodríguez, Claudio. “Amazon Web Services (AWS) : ¿Qué Es y Qué Ofrece?” *TIC Portal*, 2 Feb. 2021, www.ticportal.es/temas/cloud-computing/amazon-web-services.