

# Modularización con virtualización e Introducción a Docker y a AWS

Carlos Julian Gomez Ardila

22/09/2020

# Contents

<b>1</b>	<b>Introducción</b>	<b>3</b>
<b>2</b>	<b>Informacion General</b>	<b>3</b>
2.1	Docker . . . . .	3
2.2	Amazon EC2 . . . . .	3
2.3	MongoDB . . . . .	3
2.4	Roundrobin . . . . .	3
<b>3</b>	<b>Descripción</b>	<b>4</b>
<b>4</b>	<b>Implementación</b>	<b>5</b>
<b>5</b>	<b>Prueba de roundrobin</b>	<b>7</b>

## 1 Introducción

Este taller describe la arquitectura de una aplicación propuesta, la cual será desplegada en AWS usando EC2 y Docker, que nos permitirá crear diferentes contenedores para el mismo servicio. Para esto utilizaremos una base de datos no relacional (MongoDB) y como framework Spark.

## 2 Informacion General

### 2.1 Docker

Docker tiene como propósito crear contenedores ligeros y portables para las aplicaciones que puedan ejecutarse en cualquier máquina con Docker instalado, independientemente del sistema operativo que la máquina tenga por debajo, facilitando así también los despliegues..

### 2.2 Amazon EC2

Puede usar Amazon EC2 para lanzar tantos servidores virtuales como necesite, configurar la seguridad y las redes y administrar el almacenamiento. Amazon EC2 le permite escalar hacia arriba o hacia abajo para controlar cambios en los requisitos o picos de popularidad, con lo que se reduce la necesidad de prever el tráfico..

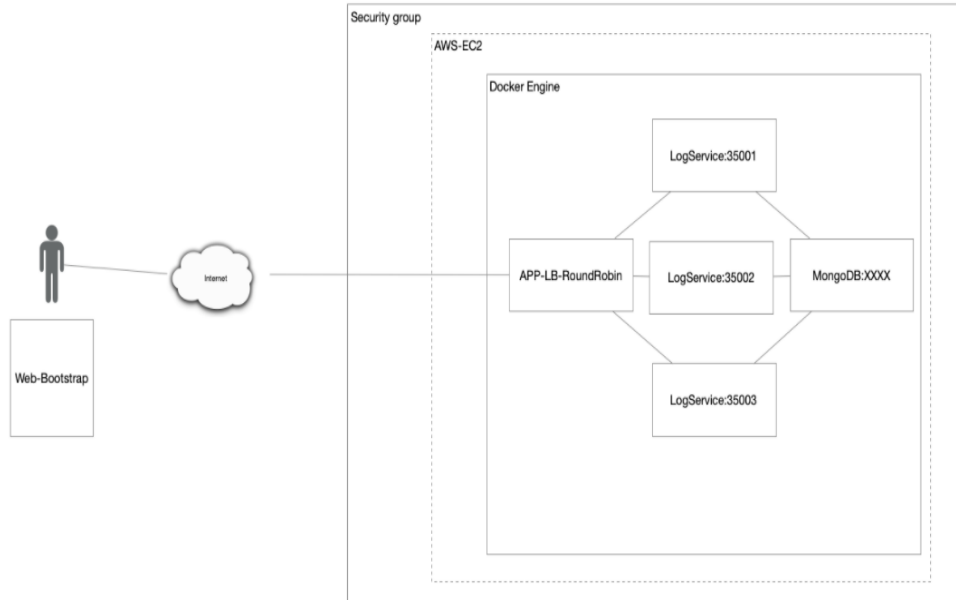
### 2.3 MongoDB

MongoDB es una base de datos de documentos que ofrece una gran escalabilidad y flexibilidad, y un modelo de consultas e indexación avanzado. Además, almacena datos en documentos flexibles similares a JSON, por lo que los campos pueden variar entre documentos y la estructura de datos puede cambiarse con el tiempo.

### 2.4 Roundrobin

Round-robin es un método para seleccionar todos los abstractos 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.

### 3 Descripción



El servicio MongoDB es una instancia de MongoDB corriendo en un container de docker en una máquina virtual de EC2

LogService es un servicio REST que recibe una cadena, la almacena en la base de datos y responde en un objeto JSON con las 10 ultimas cadenas almacenadas en la base de datos y la fecha en que fueron almacenadas.

La aplicación web APP-LB-RoundRobin está compuesta por un cliente web y al menos un servicio REST. El cliente web tiene un campo y un botón y cada vez que el usuario envía un mensaje, este se lo envía al servicio REST y actualiza la pantalla con la información que este le regresa en formato JSON. El servicio REST recibe la cadena e implementa un algoritmo de balanceo de cargas de Round Robin, delegando el procesamiento del mensaje y el retorno de la respuesta a cada una de las tres instancias del servicio LogService.

## 4 Implementación

Para su implementación necesitamos tener en cuenta la URL de nuestra máquina virtual EC2, y la ip del contenedor que crea docker para la base de datos Mongo. Como se pueden evidenciar en las siguientes imagenes, donde se indica lo ya mencionado respectivamente.

```
public static String get(String data,int port) throws UnirestException {
    int valor= 3 + port;

    HttpResponse<String> response = null;

    response = Unirest.get("http://ec2-34-230-83-190.compute-1.amazonaws.com:3500"+port+"
        .asString();

    return response.getBody();
}

private static MongoClient crearConexion() {
    MongoClient mongo = null;
    try {
        mongo = new MongoClient( host: "172.17.0.2", port: 27017);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return mongo;
}
```

Además, de los puertos necesarios tanto en los security groups como en los puertos que se van a abrir para los contenedores que en este caso son del 35000 al 35003 incluidos.

Información

Información

TCP personalizado ▼	TCP	35000 - 35004	Person... ▼ Q		Eli min ar
			0.0.0.0/0 ✕		
SSH ▼	TCP	22	Person... ▼ Q		Eli min ar
			0.0.0.0/0 ✕		
TCP personalizado ▼	TCP	42000 - 42002	Person... ▼ Q		Eli min ar
			0.0.0.0/0 ✕		
TCP personalizado ▼	TCP	27017	Person... ▼ Q		Eli min ar
			0.0.0.0/0 ✕		

Agregar regla

```
[ec2-user@ip-172-31-45-137 ~]$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS                               NAMES
2b64bd2693e4   carlosgomez380/roundrobin          "java -cp ./classes:..." 6 minutes ago  Up 6 minutes  0.0.0.0:35003->6001/tcp             roundrobin
eb5e06033bd3   carlosgomez380/logservice          "java -cp ./classes:..." 45 hours ago  Up 7 minutes  0.0.0.0:35002->6000/tcp             logservice3
76a6f415de5d   carlosgomez380/logservice          "java -cp ./classes:..." 45 hours ago  Up 7 minutes  0.0.0.0:35001->6000/tcp             logservice2
b0af5d52f405   carlosgomez380/logservice          "java -cp ./classes:..." 45 hours ago  Up 7 minutes  0.0.0.0:35000->6000/tcp             logservice1
d3dcf15f202e   mongo                              "docker-entrypoint.s..." 45 hours ago  Up 7 minutes  0.0.0.0:27017->27017/tcp            primer-mongo
[ec2-user@ip-172-31-45-137 ~]$
```

## 5 Prueba de roundrobin

Configurados los puertos y las direcciones necesarias podemos realizar las pruebas del roundrobin, para esto en cada contenedor del servicio de logs revisaremos lo que está imprimiendo y almacenando

LogService 1

```
r Mon Sep 21 02:30:13 UTC 2020
j Mon Sep 21 02:49:59 UTC 2020
q Tue Sep 22 23:07:30 UTC 2020
```

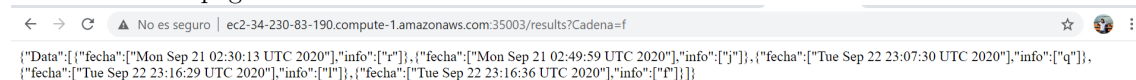
LogService 2

```
Listar los registros de la tabla:
4
r Mon Sep 21 02:30:13 UTC 2020
j Mon Sep 21 02:49:59 UTC 2020
q Tue Sep 22 23:07:30 UTC 2020
l Tue Sep 22 23:16:29 UTC 2020
```

LogService 3

```
Listar los registros de la tabla:
5
r Mon Sep 21 02:30:13 UTC 2020
j Mon Sep 21 02:49:59 UTC 2020
q Tue Sep 22 23:07:30 UTC 2020
l Tue Sep 22 23:16:29 UTC 2020
f Tue Sep 22 23:16:36 UTC 2020
```

Resultado en la página web



```
{
  "Data": [
    {
      "fecha": "Mon Sep 21 02:30:13 UTC 2020",
      "info": "r"
    },
    {
      "fecha": "Mon Sep 21 02:49:59 UTC 2020",
      "info": "j"
    },
    {
      "fecha": "Tue Sep 22 23:07:30 UTC 2020",
      "info": "q"
    },
    {
      "fecha": "Tue Sep 22 23:16:29 UTC 2020",
      "info": "l"
    },
    {
      "fecha": "Tue Sep 22 23:16:36 UTC 2020",
      "info": "f"
    }
  ]
}
```