

CS2032 - Cloud Computing (Ciclo 2024-2)

Balanceo de Carga y Alta disponibilidad

Semana 6 - Taller 2: Balanceador de Carga

ELABORADO POR: GERALDO COLCHADO

Contenido

Balanceador de Carga

1. **Objetivo del taller 2**
2. Ejercicio 1: Crear imagen de api-employees con acceso a BD MySQL en MV desarrollo
3. Ejercicio 2: Desplegar contenedor api-employees en 2 MV de producción
4. Ejercicio 3: Configurar y probar Balanceador de Carga
5. Ejercicio 4: Diagrama de Arquitectura de Solución
6. Cierre

Objetivo del taller 2:

Balanceador de Carga

- Probar Balanceo de Carga y Alta disponibilidad con Api REST con acceso a base de datos MySQL

Contenido

Balanceador de Carga

1. Objetivo del taller 2
2. **Ejercicio 1: Crear imagen de api-employees con acceso a BD MySQL en MV desarrollo**
3. Ejercicio 2: Desplegar contenedor api-employees en 2 MV de producción
4. Ejercicio 3: Configurar y probar Balanceador de Carga
5. Ejercicio 4: Diagrama de Arquitectura de Solución
6. Cierre

Ejercicio 1:

Crear imagen de api-employees con acceso a BD MySQL en MV desarrollo

- **Paso 1:** Cree un repositorio "api-employees" en github y suba los archivos indicados por el docente, luego ingrese a "MV desarrollo" /home/ubuntu/ y haga git clone.
- **Paso 2:** Analice el Dockerfile y main.py. Modifique el host_name en main.py
- **Paso 3:** Cree la imagen
\$ docker build -t api-employees .

Ejercicio 1:

Crear imagen de api-employees con acceso a BD MySQL en MV desarrollo

- **Paso 4:** Suba la imagen a <https://hub.docker.com>

\$ docker login -u gcolchado

(Reemplace amarillo)

\$ docker tag api-employees gcolchado/api-employees

(Reemplace amarillo)

\$ docker push gcolchado/api-employees

(Reemplace amarillo)

\$ docker logout

Ejercicio 1:

Crear imagen de api-employees con acceso a BD MySQL en MV desarrollo

- **Paso 5:** Cree la BD y Tabla en MySQL

Ingresa por ssh a la MV “MV Bases de Datos” y ejecute los 2 contenedores

```
$ docker run -d --rm --name mysql_c -e MYSQL_ROOT_PASSWORD=utec -p 8005:3306 -v mysql_data:/var/lib/mysql mysql:8.0
```

```
$ docker run -d --rm --name adminer_c -p 8080:8080 adminer
```

Ingresa a adminer y ejecute el script de base de datos entregado por el docente:

```
DROP DATABASE IF EXISTS bd_api_employees;
CREATE DATABASE bd_api_employees CHARSET utf8mb4;
USE bd_api_employees;

CREATE TABLE employees (
  id INT(11) NOT NULL AUTO_INCREMENT,
  name VARCHAR(100) NOT NULL,
  age INT(11) NOT NULL,
  PRIMARY KEY (id)
);

INSERT INTO employees(name, age) VALUES('Jake', 21);
INSERT INTO employees(name, age) VALUES('Mathew', 24);
INSERT INTO employees(name, age) VALUES('Bob', 35);
commit;
```

Contenido

Balanceador de Carga

1. Objetivo del taller 2
2. Ejercicio 1: Crear imagen de api-employees con acceso a BD MySQL en MV desarrollo
3. **Ejercicio 2: Desplegar contenedor api-employees en 2 MV de producción**
4. Ejercicio 3: Configurar y probar Balanceador de Carga
5. Ejercicio 4: Diagrama de Arquitectura de Solución
6. Cierre

Ejercicio 2:

Desplegar contenedor api-employees en 2 MV de producción

- **Paso 1:** Ingrese por ssh y ejecute el contenedor en las 2 MV de producción
\$ docker run -d --rm --name api-employees_c -p 8000:8000 gcolchado/api-employees

Contenido

Balanceador de Carga

1. Objetivo del taller 2
2. Ejercicio 1: Crear imagen de api-employees con acceso a BD MySQL en MV desarrollo
3. Ejercicio 2: Desplegar contenedor api-employees en 2 MV de producción
4. **Ejercicio 3: Configurar y probar Balanceador de Carga**
5. Ejercicio 4: Diagrama de Arquitectura de Solución
6. Cierre

Ejercicio 3:

Configurar y probar Balanceador de Carga

- **Paso 1:** En grupo de seguridad “GS-Prod”, que usan las 2 MV de producción, abra puerto 8000
- **Paso 2:** Crear un Target Group con las 2 MV de producción para el puerto 8000

Target group name

TG-Prod-8000

A maximum of 32 alphanumeric characters including hyphens and underscores

Protocol

Port

HTTP



:

8000

Ejercicio 3:

Configurar y probar Balanceador de Carga

- **Paso 3:** Agregue un agente de escucha en el Balanceador de Carga

lb-prod | **Agregar agente de escucha**

Los agentes de escucha pertenecientes a balanceadores de carga de aplicac
escucha debe incluir una acción predeterminada para garantizar que se enru
direccionamiento adicionales que necesite. [Más información](#)

Protocolo - Puerto

Seleccione el protocolo para las conexiones desde el cliente al balanceador c

:

Acciones predeterminadas

Indique cómo este agente de escucha enrutará el tráfico no enrutado por otra

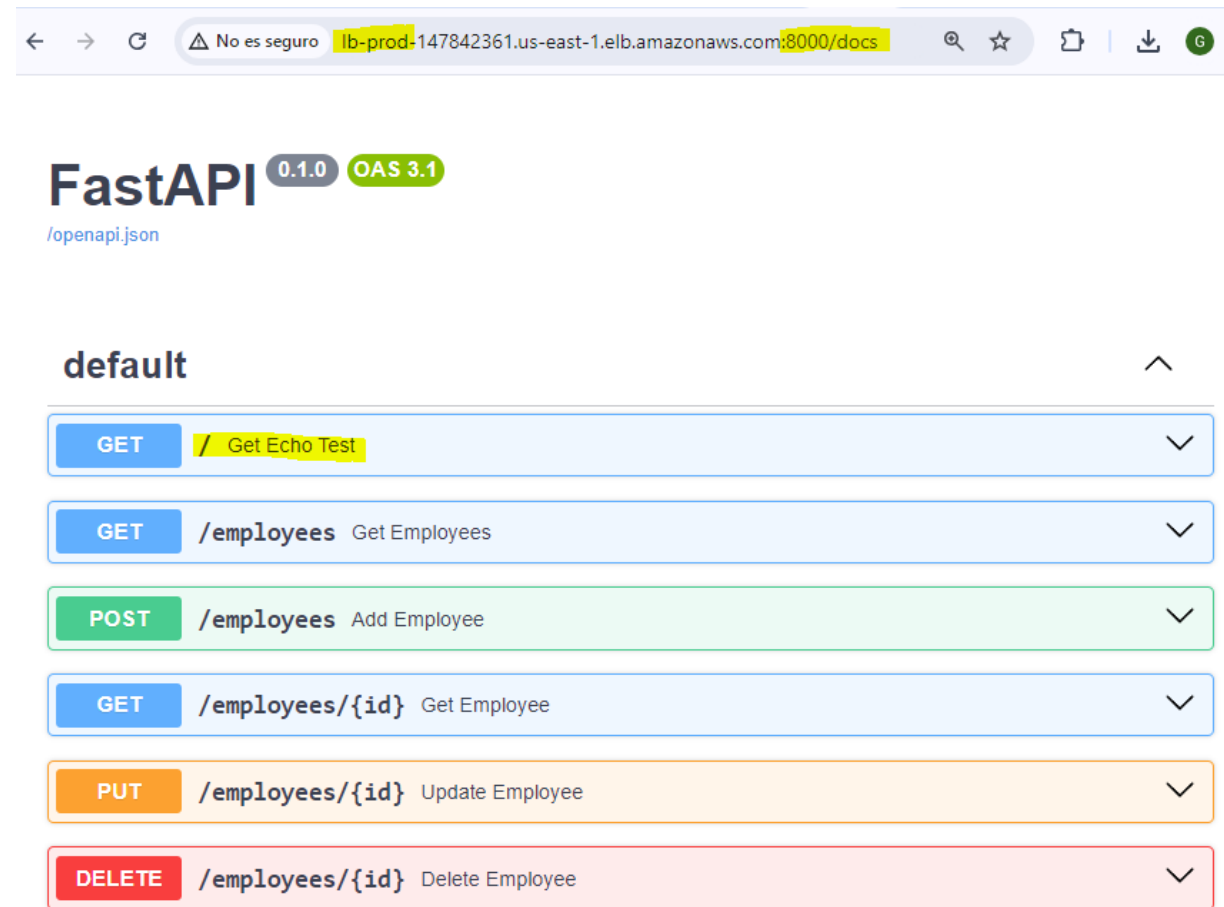
1. Reenviar a...

Grupo de destino: peso (0-999)

Ejercicio 3:

Configurar y probar Balanceador de Carga

- **Paso 4:** Consulte la documentación y **pruebe** el api



Nota: The **Echo Test** is used to detect whether the service is available

Ejercicio 3:

Configurar y probar Balanceador de Carga

- **Paso 5:** Pruebe también en postman el api-employees con el enlace del balanceador usando la colección postman.

Ejercicio 3:

Configurar y probar Balanceador de Carga

api-employees / Consultar empleados

GET http://lb-prod-1845552211.us-east-1.elb.amazonaws.com:8000/employees

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Body Cookies Headers (5) Test Results

200 OK 110

Pretty Raw Preview Visualize JSON

```
1 {
2   "employees": [
3     [
4       1,
5       "Juan Pérez",
6       30
7     ],
8     [
9       2,
10      "Mathew",
11      24
12     ],
13     [
14       3,
15       "Bob",
16       35
17     ]
18   ]
19 }
```

api-employees / Consulta un empleado

GET http://lb-prod-1845552211.us-east-1.elb.amazonaws.com:8000/employees/1

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

	Key	Value	Descri
	Key	Value	Descri

Body Cookies Headers (5) Test Results

200 OK 334

Pretty Raw Preview Visualize JSON

```
1 {
2   "employee": [
3     1,
4     "Juan Pérez",
5     30
6   ]
7 }
```

Ejercicio 3:

Configurar y probar Balanceador de Carga

HTTP api-employees / Nuevo empleado

POST http://lb-prod-1845552211.us-east-1.elb.amazonaws.com:8000/employees

Params Authorization Headers (9) Body Pre-request Script Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL ☒ JSON

```
1 {
2   "name": "Juan Pérez",
3   "age": 25
4 }
```

Body Cookies Headers (5) Test Results 200 OK 582 ms 19

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "Employee added successfully"
3 }
```


Ejercicio 3:

Configurar y probar Balanceador de Carga

The screenshot shows a REST client interface with the following details:

- URL:** `http://lb-prod-1845552211.us-east-1.elb.amazonaws.com:8000/employees/1`
- Method:** `PUT`
- Body Type:** `JSON`
- Request Body:**

```
1 {  
2   "name": "Jorge Carrasco",  
3   "age": 19  
4 }
```
- Response Status:** `200 OK` (363 ms)
- Response Body (Pretty):**

```
1 {  
2   "message": "Employee modified successfully"  
3 }
```

Ejercicio 3:

Configurar y probar Balanceador de Carga

HTTP api-employees / Eliminar empleado

DELETE ▼ http://lb-prod-1845552211.us-east-1.elb.amazonaws.com:8000/employees/15

Params Authorization Headers (7) Body Scripts Tests Settings

Body Cookies Headers (5) Test Results 🌐 200 OK 4.92

Pretty Raw Preview Visualize JSON ▼ ⌵

```
1 {  
2   "message": "Employee deleted successfully"  
3 }
```

Ejercicio 3:

Configurar y probar Balanceador de Carga

- **Paso 6:** Detener la instancia “MV Prod 1” y probar
- **Paso 7:** Detener la instancia “MV Prod 2” y probar
- **Paso 8:** Iniciar la instancia “MV Prod 1”, [ejecutar el contenedor](#) y probar
\$ docker run -d --rm --name api-employees_c -p 8000:8000 gcolchado/api-employees
- **Paso 9:** Iniciar la instancia “MV Prod 2”, [ejecutar el contenedor](#) y probar
\$ docker run -d --rm --name api-employees_c -p 8000:8000 gcolchado/api-employees

Contenido

Balanceador de Carga

1. Objetivo del taller 2
2. Ejercicio 1: Crear imagen de api-employees con acceso a BD MySQL en MV desarrollo
3. Ejercicio 2: Desplegar contenedor api-employees en 2 MV de producción
4. Ejercicio 3: Configurar y probar Balanceador de Carga
5. **Ejercicio 4: Diagrama de Arquitectura de Solución**
6. Cierre

Ejercicio 4:

Diagrama de Arquitectura de Solución

- Elabore en draw.io el Diagrama de Arquitectura de Solución del Api REST con acceso a base de datos MySQL balanceada en carga usando el puerto 8000. Publique su diagrama en el padlet. Este ejercicio es guiado.

Contenido

Balanceador de Carga

1. Objetivo del taller 2
2. Ejercicio 1: Crear imagen de api-employees con acceso a BD MySQL en MV desarrollo
3. Ejercicio 2: Desplegar contenedor api-employees en 2 MV de producción
4. Ejercicio 3: Configurar y probar Balanceador de Carga
5. Ejercicio 4: Diagrama de Arquitectura de Solución
6. Cierre

Cierre:

Balanceador de Carga - Qué aprendimos?

- Balanceo de Carga y Alta disponibilidad con Api REST con acceso a base de datos MySQL

Gracias

Elaborado por docente: Geraldo Colchado