UTEC
UNIVERSIDAD DE INGENIERÍA
Y TECNOLOGÍA

CS2032 - Cloud Computing (Ciclo 2024-2)
Balanceo de Carga y Alta disponibilidad
Semana 6 - Taller 3: Balanceador de Carga

ELABORADO POR: GERALDO COLCHADO

Contenido

Balanceador de Carga

1. **Objetivo del taller 3**
2. Ejercicio 1: Crear contenedor de MongoDB en MV Bases de datos
3. Ejercicio 2: Crear contenedor de api-fruits con acceso a MongoDB en MV Desarrollo
4. Ejercicio 3: Desplegar contenedor api-fruits en 2 MV de producción
5. Ejercicio 4: Configurar y probar Balanceador de Carga
6. Ejercicio 5: Diagrama de Arquitectura de Solución
7. Cierre

Objetivo del taller 3:
Balanceador de Carga

- Probar Balanceo de Carga y Alta disponibilidad con Api REST con acceso a base de datos MongoDB (NoSQL)

Contenido

Balanceador de Carga

1. Objetivo del taller 3
2. **Ejercicio 1: Crear contenedor de MongoDB en MV Bases de datos**
3. Ejercicio 2: Crear contenedor de api-fruits con acceso a MongoDB en MV Desarrollo
4. Ejercicio 3: Desplegar contenedor api-fruits en 2 MV de producción
5. Ejercicio 4: Configurar y probar Balanceador de Carga
6. Ejercicio 5: Diagrama de Arquitectura de Solución
7. Cierre

Ejercicio 1:
Crear contenedor de MongoDB en MV Bases de datos

- **Paso 1:** Ingrese a la máquina virtual “MV Bases de datos” por ssh a la IP Elástica
- **Paso 2:** Ejecute el contenedor de MongoDB y verifique se haya creado el volumen y valide los logs de ejecución:
  $ docker run -d --rm --name mongo_c -p 27017:27017 -v mongo_data:/data/db mongo:latest
  $ docker volume ls
  $ docker logs mongo_c
- **Paso 3:** Dado que el MongoDB se está ejecutando sin usuario y password (no tiene seguridad), sólo debe ser accedido por máquinas virtuales de la misma red local y no de internet por lo que en el grupo de seguridad de la máquina virtual “MV Bases de datos” abra el puerto 27017 tanto para el mismo grupo de seguridad como para el grupo de seguridad de MV Desarrollo y MV Pruebas y para el grupo de seguridad de producción “GS-Prod”

Tipo	Protocolo	Intervalo de puertos	Origen
SSH	TCP	22	0.0.0.0/0
TCP personalizado	TCP	27017	sg-04854c041c2dd5997 / GS-Prod
TCP personalizado	TCP	27017	sg-0bfb43fb5ec0fb0e2 / launch-wizard-1
TCP personalizado	TCP	27017	sg-06b919a5bf6b41dfc / crear-mv-bd-Instan
TCP personalizado	TCP	27017	sg-0b0726316a14e5387 / crear-mv-pruebas-

Ejercicio 1:
Crear contenedor de MongoDB en MV Bases de datos

- **Paso 4:** Conectarse al linux del contenedor
  $ docker exec -it mongo_c bash

- **Paso 5:** Conectarse al MongoDB
  # mongosh

- **Paso 6:** Crear base de datos “food”, colección “fruits”, consultar datos y salir

```
use food

db.createCollection("fruits")

db.fruits.insertMany([ {name: "apple", origin: "usa", price: 5},
{name: "orange", origin: "italy", price: 3}, {name: "mango",
origin: "malaysia", price: 3} ])

db.fruits.find().pretty()

exit

exit
```

Contenido

Balanceador de Carga

1. Objetivo del taller 3
2. Ejercicio 1: Crear contenedor de MongoDB en MV Bases de datos
3. **Ejercicio 2: Crear contenedor de api-fruits con acceso a MongoDB en MV Desarrollo**
4. Ejercicio 3: Desplegar contenedor api-fruits en 2 MV de producción
5. Ejercicio 4: Configurar y probar Balanceador de Carga
6. Ejercicio 5: Diagrama de Arquitectura de Solución
7. Cierre

Ejercicio 2:
Crear contenedor de api-fruits con acceso a MongoDB en MV Desarrollo

- **Paso 1:** Cree un repo “api-fruits” en github y suba los archivos indicados por el docente. Luego ingrese a “MV Desarrollo” y descargue repo con git clone.
- **Paso 2:** Analice el Dockerfile y app.py. En app.py reemplace por IP privada de su MV “MV Bases de datos”

```
class MongoAPI:
    def __init__(self, data):
        log.basicConfig(level=log.DEBUG, format='%(asctime)s %(levelname)s:\n%(message)s\n')
        self.client = MongoClient("mongodb://172.31.92.214:27017") # IP privada de MV Base de Datos
        database = data['database']
        collection = data['collection']
```

- **Paso 3:** Cree la imagen
  $ docker build -t api-fruits .

Referencia del api-fruits: https://ishmeet1995.medium.com/how-to-create-restful-crud-api-with-python-flask-mongodb-and-docker-8f6ccb73c5bc

Ejercicio 2:
Crear contenedor de api-fruits con acceso a MongoDB en MV Desarrollo

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

  $ docker login -u gcolchado (Reemplace amarillo)

  $ docker tag api-fruits gcolchado/api-fruits (Reemplace amarillo)

  $ docker push gcolchado/api-fruits (Reemplace amarillo)

  $ docker logout

Ejercicio 2:
Crear contenedor de api-fruits con acceso a MongoDB en MV Desarrollo

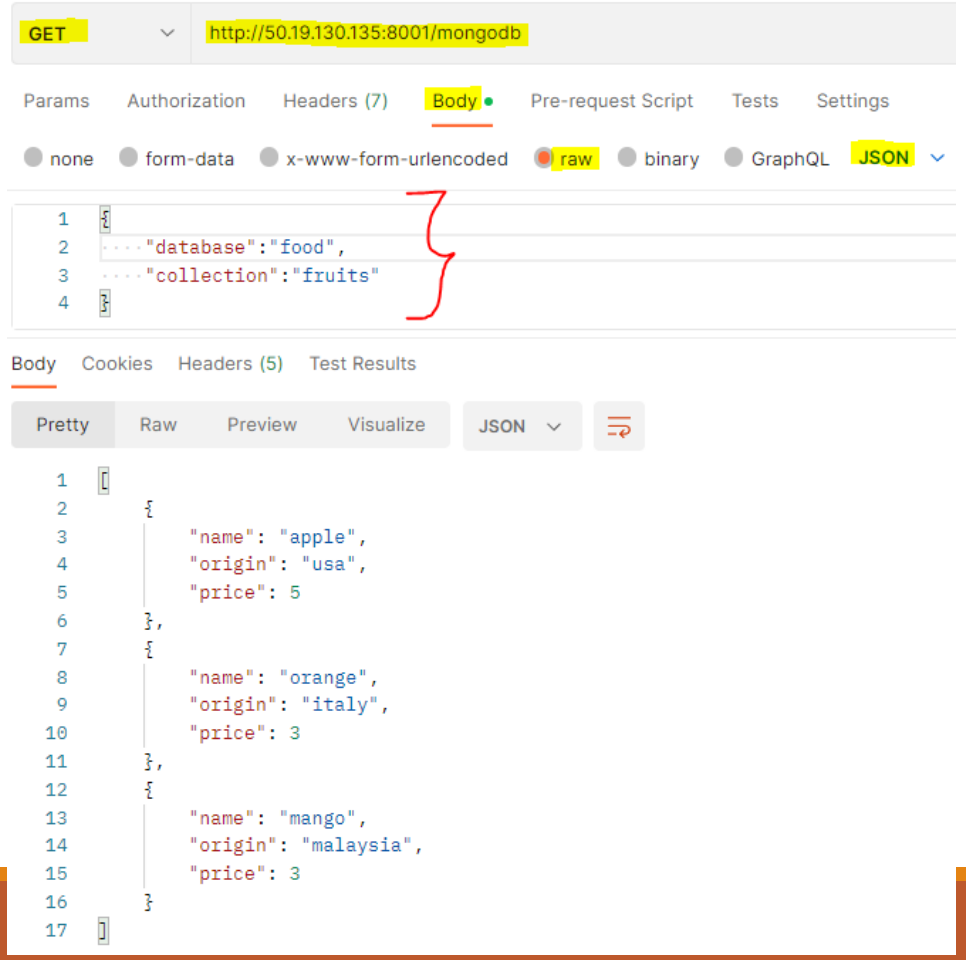
- **Paso 5:** Ejecute el contenedor y verifique logs:
  $ docker run -d --rm --name api-fruits_c -p 8001:8001 api-fruits
  $ docker logs api-fruits_c

- **Paso 6:** Abra el puerto 8001 en el grupo de seguridad de MV “MV Desarrollo”

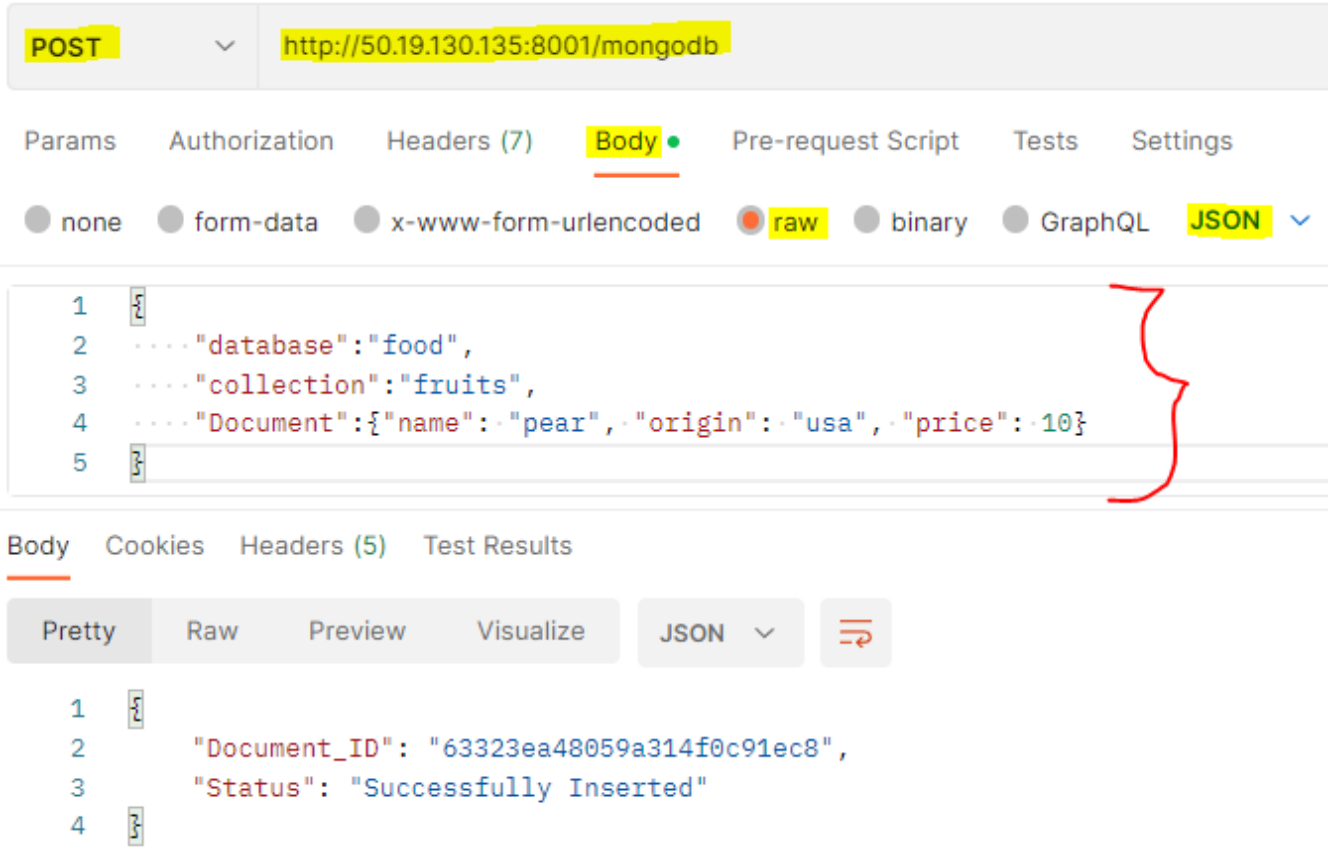
- **Paso 7:** Pruebe en postman el api-fruits con la IP de la MV “MV Desarrollo”

Ejercicio 2:
Crear contenedor de api-fruits con acceso a MongoDB en MV Desarrollo

Consultar frutas

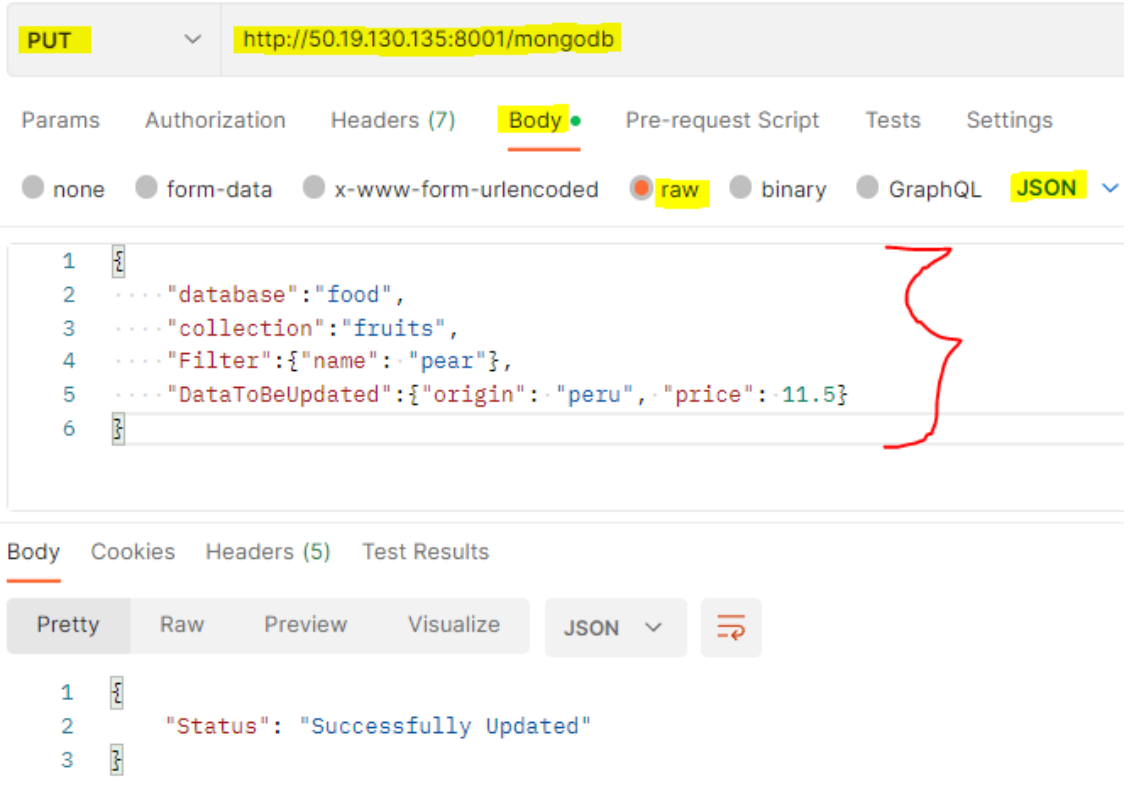


Crear fruta

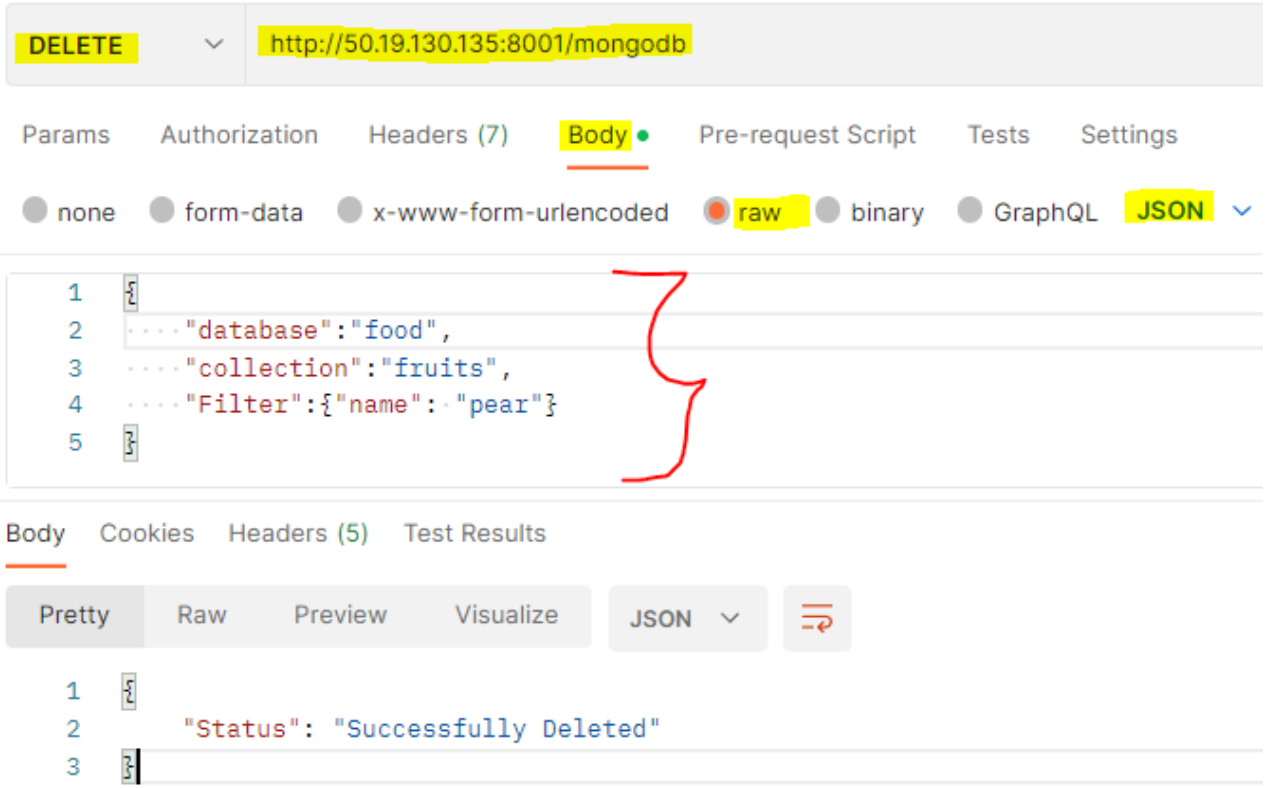


Ejercicio 2:
Crear contenedor de api-fruits con acceso a MongoDB en MV Desarrollo

Modificar fruta



Eliminar fruta



Contenido

Balanceador de Carga

1. Objetivo del taller 3
2. Ejercicio 1: Crear contenedor de MongoDB en MV Bases de datos
3. Ejercicio 2: Crear contenedor de api-fruits con acceso a MongoDB en MV Desarrollo
4. **Ejercicio 3: Desplegar contenedor api-fruits en 2 MV de producción**
5. Ejercicio 4: Configurar y probar Balanceador de Carga
6. Ejercicio 5: Diagrama de Arquitectura de Solución
7. Cierre

Ejercicio 3:
Desplegar contenedor api-fruits en 2 MV de producción

- **Paso 1:** Ingrese por ssh y ejecute el contenedor en las 2 MV de producción y verifique logs de ejecución:

  $ docker run -d --rm --name api-fruits_c -p 8001:8001 gcolchado/api-fruits
  $ docker logs api-fruits_c

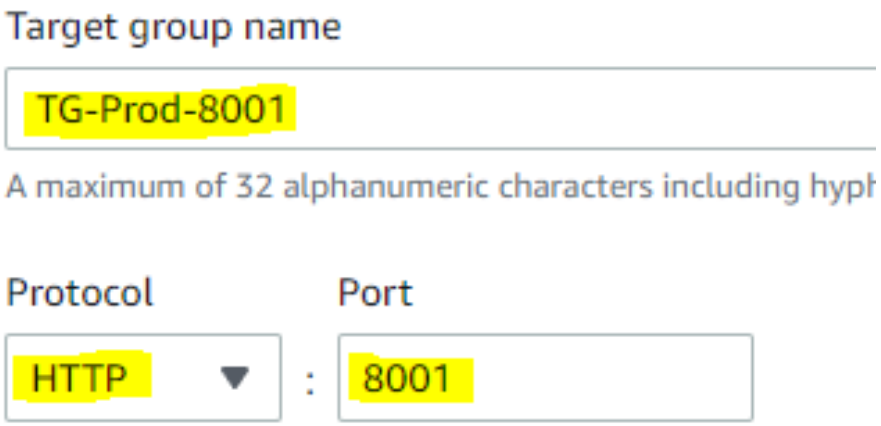
Contenido

Balanceador de Carga

1. Objetivo del taller 3
2. Ejercicio 1: Crear contenedor de MongoDB en MV Bases de datos
3. Ejercicio 2: Crear contenedor de api-fruits con acceso a MongoDB en MV Desarrollo
4. Ejercicio 3: Desplegar contenedor api-fruits en 2 MV de producción
5. **Ejercicio 4: Configurar y probar Balanceador de Carga**
6. Ejercicio 5: Diagrama de Arquitectura de Solución
7. Cierre

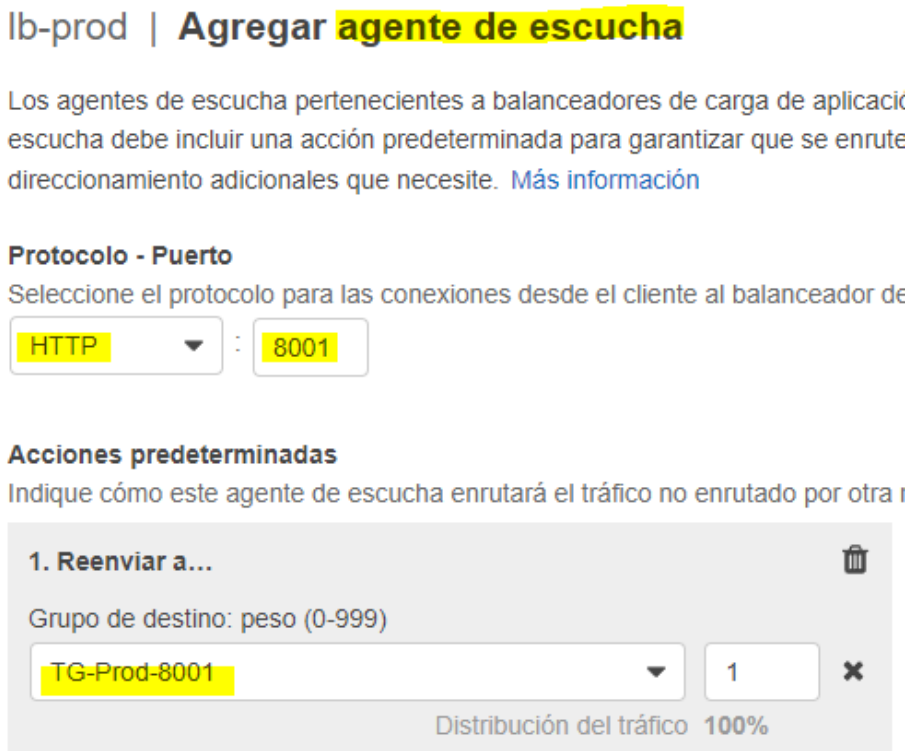
Ejercicio 4:
Configurar y probar Balanceador de Carga

- **Paso 1:** En grupo de seguridad “GS-Prod”, que usan las 2 MV de producción, abra puerto 8001
- **Paso 2:** Crear un Target Group con las 2 MV de producción para el puerto 8001



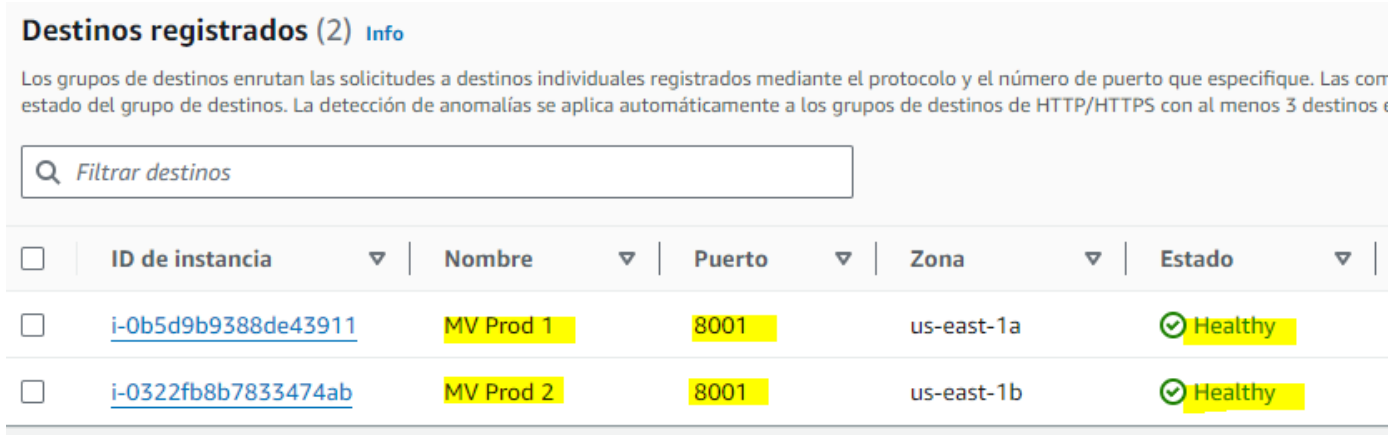
Ejercicio 4:
Configurar y probar Balanceador de Carga

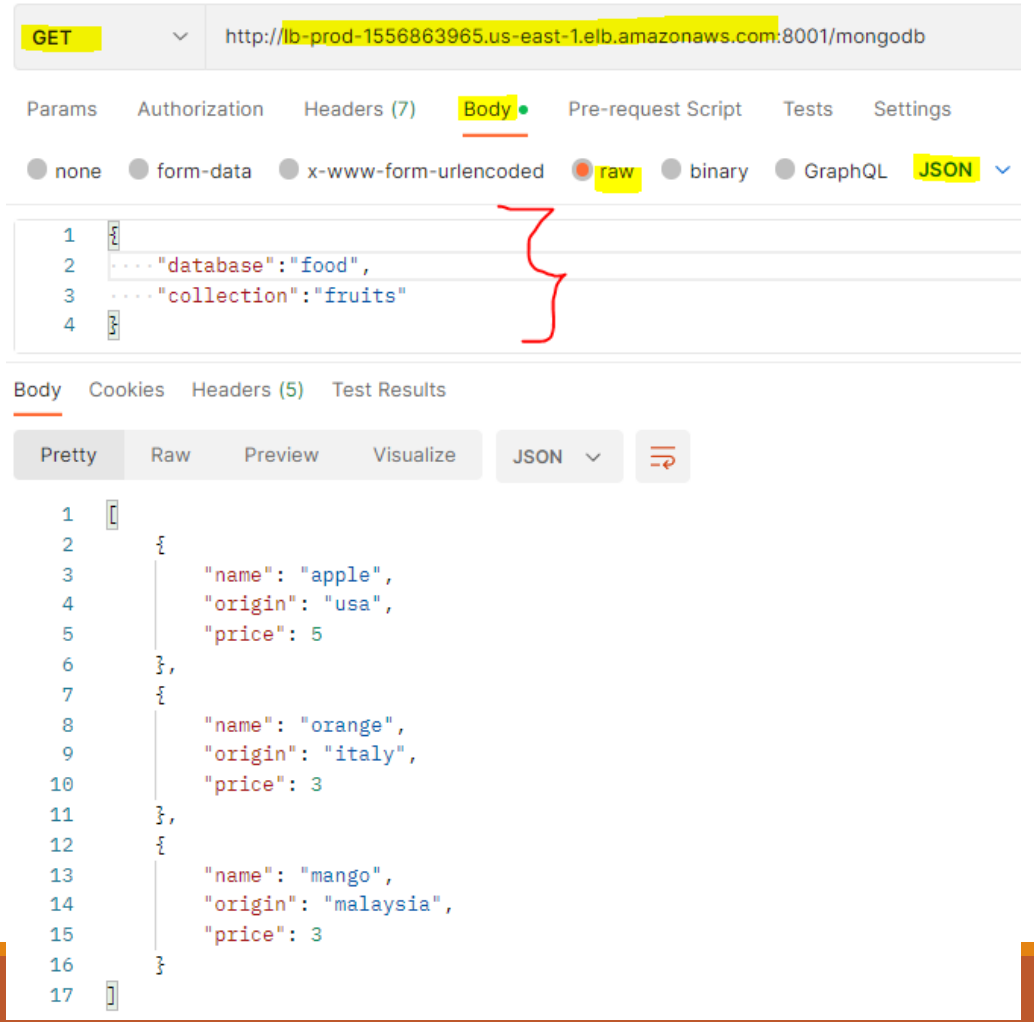
- **Paso 3:** Agregue un agente de escucha en el Balanceador de Carga



Ejercicio 4:
Configurar y probar Balanceador de Carga

- **Paso 4:** Verifique estén “Healthy” las 2 MV de producción y luego pruebe en postman con enlace de balanceador de carga varias veces.





- **Paso 5:** Detener la instancia “MV Prod 1” y probar
- **Paso 6:** Detener la instancia “MV Prod 2” y probar
- **Paso 7:** Iniciar la instancia “MV Prod 1”, ejecutar el contenedor y probar
  $ docker run -d --rm --name api-fruits_c -p 8001:8001 gcolchado/api-fruits
- **Paso 8:** Iniciar la instancia “MV Prod 2”, ejecutar el contenedor y probar
  $ docker run -d --rm --name api-fruits_c -p 8001:8001 gcolchado/api-fruits

Contenido

Balanceador de Carga

1. Objetivo del taller 3
2. Ejercicio 1: Crear contenedor de MongoDB en MV Bases de datos
3. Ejercicio 2: Crear contenedor de api-fruits con acceso a MongoDB en MV Desarrollo
4. Ejercicio 3: Desplegar contenedor api-fruits en 2 MV de producción
5. Ejercicio 4: Configurar y probar Balanceador de Carga
6. **Ejercicio 5: Diagrama de Arquitectura de Solución**
7. Cierre

Ejercicio 5:
Diagrama de Arquitectura de Solución

- Elabore en draw.io el Diagrama de Arquitectura de Solución del Api REST con acceso a base de datos MongoDB balanceada en carga usando el puerto 8001. Publique su diagrama en el padlet.

Contenido

Balanceador de Carga

1. Objetivo del taller 3
2. Ejercicio 1: Crear contenedor de MongoDB en MV Bases de datos
3. Ejercicio 2: Crear contenedor de api-fruits con acceso a MongoDB en MV Desarrollo
4. Ejercicio 3: Desplegar contenedor api-fruits en 2 MV de producción
5. Ejercicio 4: Configurar y probar Balanceador de Carga
6. Ejercicio 5: Diagrama de Arquitectura de Solución
7. **Cierre**

Cierre:
Balanceador de Carga - Qué aprendimos?

- Balanceo de Carga y Alta disponibilidad con Api REST con acceso a base de datos MongoDB (NoSQL)

Gracias

Elaborado por docente: Geraldo Colchado