

P2 Balanceo de Carga

Mario Rodriguez Lopez

March 2024

Índice

B1. PREPARACIÓN DEL ENTORNO DE TRABAJO	2
B2. CONFIGURACIÓN DE NGINX COMO BALANCEADOR DE CARGA	2
B3. IMPLEMENTACIÓN DEL ESCENARIO DE NGINX CON DOCKER COMPOSE	3
B4. VERIFICACIÓN Y PRUEBAS DEL ESCENARIO DE NGINX	5
B5. CONFIGURACIÓN DE HAPROXY COMO BALANCEADOR DE CARGA	6
B6. IMPLEMENTACION DEL ESCENARIO DE HAPROXY CON DOCKER COMPOSE	7
B7. VERIFICACION Y PRUEBAS DEL ESCENARIO DE HAPROXY	10
A1. CONFIGURACIONES AVANZADAS DE NGINX	11
A2. CONFIGURACIONES AVANZADAS DE HAPROXY	12
A3. EXPERIMENTACIÓN CON DIFERENTES BALANCEADORES DE CARGA	13
A4. INVESTIGACIÓN Y PRUEBAS DE TOLERANCIA A FALLOS	22
A5. AUTOMATIZACIÓN DE ESCALADO DEL ESCENARIO	24

B1. PREPARACIÓN DEL ENTORNO DE TRABAJO

Creamos los directorios como sigue:

```
mkdir P2-MarioRgzLpz-nginx
mkdir P2-MarioRgzLpz-haproxy
```

No he usado IA para realizar esto puesto que es muy simple.

B2. CONFIGURACIÓN DE NGINX COMO BALANCEADOR DE CARGA

B2.1. Propuesta final DockerFile, configuracion nginx y comandos docker

```
#Imagen base
FROM nginx:latest

# Copia el archivo de configuracion personalizado
COPY nginx.conf /etc/nginx/nginx.conf
```

Para crear la imagen uso el siguiente comando

```
docker build -t mariorgzlpz-nginx-image:p2 -f DockerFileNginx .
```

```
events {
    worker_connections 1024;
}

http {
    upstream backend_MarioRgzLpz {
        #round-robin
        server 192.168.10.2;
        server 192.168.10.3;
        server 192.168.10.4;
        server 192.168.10.5;
        server 192.168.10.6;
        server 192.168.10.7;
        server 192.168.10.8;
        server 192.168.10.9;
    }
    server {
        listen 80;
        server_name nginx_MarioRgzLpz;
        access_log /var/log/nginx/nginx_MarioRgzLpz.access_log;
        error_log /var/log/nginx/nginx_MarioRgzLpz.error_log;
        location / {
            proxy_pass http://backend_MarioRgzLpz;
            proxy_set_header Cookie $http_cookie;
            proxy_hide_header Set-Cookie;
        }
        location /estadisticas_MarioRgzLpz {
            stub_status on;
        }
    }
}
```

Para lanzar el balanceador uso el siguiente comando

```
docker run -d -p 80:80 --network p1_red_web --name balanceador-nginx mariorgzlpz-nginx-image:p2
```

B2.2. Análisis propuesta IA

Preguntas a ChatGPT:

-Tengo 8 contenedores conectados a una red llamada red_web y que muestran en las IPs de la red_web un fichero index.php. Quiero crear con el siguiente DockerFile (le paso el DockerFile) y la siguiente configuración nginx (le paso la configuración del guión) un balanceador de carga de manera que pueda acceder a localhost desde el navegador y vaya mandando peticiones a cada uno de los contenedores de la red_web en función del algoritmo de balanceo.

-Me aparece que el puerto 80 esta en uso

-Quiero que en el localhost nginx actue como balanceador haciendo que se vayan mostrando los distintos contenedores

Respuestas de ChatGPT

-Corrección de la configuración de Nginx: En tu archivo de configuración de Nginx, la sección http debe ir dentro del archivo nginx.conf, no en un archivo separado. Además, faltan algunos detalles en la configuración, como especificar el tipo de balanceo de carga (round-robin) y definir la directiva server dentro del bloque upstream y los events.

-Detener el servicio que está utilizando el puerto 80: Puedes detener el servicio que actualmente está utilizando el puerto 80 en tu sistema. Por ejemplo, si Apache u otro servidor web está en ejecución, puedes detenerlo temporalmente mientras pruebas tu contenedor de Nginx.

-Es importante tener en cuenta que necesitas asegurarte de que los contenedores de Apache estén en la misma red de Docker que el contenedor de Nginx para que puedan comunicarse entre sí.

Análisis

He tenido que realizar bastantes preguntas a ChatGPT para que me proporcione la información que necesitaba. Al principio me decía los errores del archivo de configuración (como que el algoritmo round-robin debe aparecer comentado ya que no es una directiva de la configuración de nginx, además también es necesario añadir los events en la configuración). Después usando el comando para lanzar el contenedor una vez que tenía listo el DockerFile y la configuración me aparecía un error diciendo que el puerto 80 estaba en uso y arreglar eso simplemente tenía que parar el servicio que estaba usando ese puerto que en mi caso era nginx. Tras esto se me quedaba colgado en la dirección http://localhost debido a que no había conectado el balanceador a la red y gracias a una respuesta de ChatGPT me doy cuenta del fallo. Al final tras todas las configuraciones me funciona de manera correcta y al acceder a localhost y ir recargando se va cambiando la IP que aparece lo que significa que está balanceando la carga entre los contenedores. Como el algoritmo por defecto es round-robin simplemente va cambiando de un contenedor al siguiente. Además por un problema de Docker la asignación de IPs no siempre es correcta por lo que tengo que seguir usando el script de la P1 que me ajusta las IPs de manera automática.

B3. IMPLEMENTACIÓN DEL ESCENARIO DE NGINX CON DOCKER COMPOSE

B3.1. Propuesta final

Uso el mismo archivo que me proporciona ChatGPT, añadiéndole el puerto.

B3.2. Análisis propuesta IA

Preguntas a ChatGPT:

-A partir de este docker compose (docker compose de la p1) y este DockerFile (dockerfile del apartado anterior) dame otro docker compose con los mismos 8 contenedores y redes, que añada un servicio llamado balanceador-nginx que use la imagen mariorgzlpz-nginx-image:p2 creada a partir del DockerFile y que se conecte a la red web en la dirección IP 192.168.10.50 para funcionar como balanceador de carga, además se debe crear un volumen para montar el archivo nginx.conf en el contenedor en /etc/nginx/nginx.conf y el balanceador debe depender de los servicios webX para garantizar un orden correcto en el despliegue.

Respuestas de ChatGPT

```
version: '4.0'

services:
  web1:
    image: mariorgzlpz-apache-image:p2
    volumes:
```

```

    - ./web_MarioRgzLpz:/var/www/localhost/htdocs
  container_name: web1
  networks:
    red_web:
      ipv4_address: 192.168.10.2
    red_servicios:
      ipv4_address: 192.168.20.2

web2:
  image: mariorgzlpz--apache--image:p2
  volumes:
    - ./web_MarioRgzLpz:/var/www/localhost/htdocs
  container_name: web2
  networks:
    red_web:
      ipv4_address: 192.168.10.3
    red_servicios:
      ipv4_address: 192.168.20.3

web3:
  image: mariorgzlpz--apache--image:p2
  volumes:
    - ./web_MarioRgzLpz:/var/www/localhost/htdocs
  container_name: web3
  networks:
    red_web:
      ipv4_address: 192.168.10.4
    red_servicios:
      ipv4_address: 192.168.20.4

web4:
  image: mariorgzlpz--apache--image:p2
  volumes:
    - ./web_MarioRgzLpz:/var/www/localhost/htdocs
  container_name: web4
  networks:
    red_web:
      ipv4_address: 192.168.10.5
    red_servicios:
      ipv4_address: 192.168.20.5

web5:
  image: mariorgzlpz--apache--image:p2
  volumes:
    - ./web_MarioRgzLpz:/var/www/localhost/htdocs
  container_name: web5
  networks:
    red_web:
      ipv4_address: 192.168.10.6
    red_servicios:
      ipv4_address: 192.168.20.6

web6:
  image: mariorgzlpz--apache--image:p2
  volumes:
    - ./web_MarioRgzLpz:/var/www/localhost/htdocs
  container_name: web6
  networks:
    red_web:
      ipv4_address: 192.168.10.7
    red_servicios:
      ipv4_address: 192.168.20.7

web7:
  image: mariorgzlpz--apache--image:p2
  volumes:
    - ./web_MarioRgzLpz:/var/www/localhost/htdocs
  container_name: web7
  networks:
    red_web:
      ipv4_address: 192.168.10.8
    red_servicios:
      ipv4_address: 192.168.20.8

```

```

web8:
  image: mariorgzlpz-apache-image:p2
  volumes:
    - ./web_MarioRgzLpz:/var/www/localhost/htdocs
  container_name: web8
  networks:
    red_web:
      ipv4_address: 192.168.10.9
    red_servicios:
      ipv4_address: 192.168.20.9

balanceador-nginx:
  image: mariorgzlpz-nginx-image:p2
  volumes:
    - ./nginx.conf:/etc/nginx/nginx.conf
  container_name: balanceador-nginx
  ports:
    - "80:80"
  networks:
    red_web:
      ipv4_address: 192.168.10.50
  depends_on:
    - web1
    - web2
    - web3
    - web4
    - web5
    - web6
    - web7
    - web8

networks:
  red_web:
    driver: bridge
    ipam:
      config:
        - subnet: 192.168.10.0/24
          gateway: 192.168.10.1
  red_servicios:
    driver: bridge
    ipam:
      config:
        - subnet: 192.168.20.0/24
          gateway: 192.168.20.1

```

Análisis

El docker compose generado es correcto simplemente falta ponerle que el puerto sea el 80:80 para poder verlo en la direccion localhost, modificar la imagen de apache de la version p1 a la p2 y quitar el volumen montado para la configuración de php que no es necesario, además de modificar el DockerFile para que no se copie la configuración puesto que ya tenemos el volumen montado.

B4. VERIFICACIÓN Y PRUEBAS DEL ESCENARIO DE NGINX

Verificamos que todo funcione correcto con el docker compose y con docker ps mostramos que todos los contenedores esten activos

```

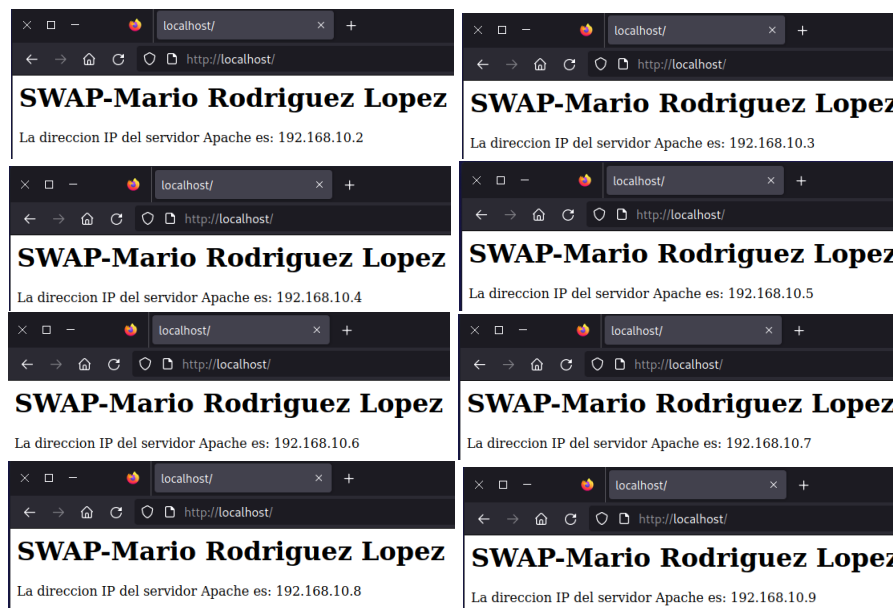
parrot: ~/.../P2-MarioRgzLpz-nginx
-- docker compose up
[+] Running 11/10
✔ Network p2-mariorgzlpz-nginx_red_web      Created
✔ Network p2-mariorgzlpz-nginx_red_servicios Created
✔ Container web6                            Created
✔ Container web7                            Created
✔ Container web5                            Created
✔ Container web3                            Created
✔ Container web1                            Created
✔ Container web8                            Created
✔ Container web2                            Created
✔ Container web4                            Created
✔ Container balanceador-nginx              Created
Attaching to balanceador-nginx, web1, web2, web3, web4, web5, web6, web7, web8

```

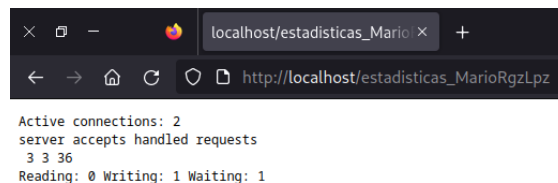
```
barrot: ~/.../P1/scripts
~ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
p54a5d58ea40	mariorgzlpz-nginx-image:p2	"/docker-entrypoint..."	47 seconds ago	Up 44 seconds	0.0.0.0:80->80/tcp, :::80->80/tcp	balanceador-nginx
558e9458b185	mariorgzlpz-apache-image:p1	"httpd -D FOREGROUND"	47 seconds ago	Up 45 seconds	80/tcp	web7
5fae0d201468	mariorgzlpz-apache-image:p1	"httpd -D FOREGROUND"	47 seconds ago	Up 44 seconds	80/tcp	web3
1575f18976fe	mariorgzlpz-apache-image:p1	"httpd -D FOREGROUND"	47 seconds ago	Up 44 seconds	80/tcp	web5
ac7193121878	mariorgzlpz-apache-image:p1	"httpd -D FOREGROUND"	47 seconds ago	Up 45 seconds	80/tcp	web8
87dea6967357	mariorgzlpz-apache-image:p1	"httpd -D FOREGROUND"	47 seconds ago	Up 45 seconds	80/tcp	web2
4f3021c18cd1	mariorgzlpz-apache-image:p1	"httpd -D FOREGROUND"	47 seconds ago	Up 44 seconds	80/tcp	web4
e5c3389ee95b	mariorgzlpz-apache-image:p1	"httpd -D FOREGROUND"	47 seconds ago	Up 44 seconds	80/tcp	web1
25a5f11daa40	mariorgzlpz-apache-image:p1	"httpd -D FOREGROUND"	47 seconds ago	Up 45 seconds	80/tcp	web6

Comprobamos que todas las IPs se muestran correctamente y ordenadas mediante el algoritmo Round-Robin (el que tengo por defecto) y que simplemente ira mandando peticiones empezando por la primera IP que hemos puesto en el fichero de configuración nginx hasta llegar a la ultima, de manera que la siguiente petición se enviaría de nuevo a la primera IP y asi de manera sucesiva.



También comprobamos que las estadísticas se vean de manera correcta.



Para este apartado no he usado IA puesto que era solo realizar comprobaciones.

B5. CONFIGURACIÓN DE HAPROXY COMO BALANCEADOR DE CARGA

B5.1. Propuesta final DockerFile, configuracion haproxy y comandos docker

```
#Imagen base
FROM haproxy:latest

# Copia el archivo de configuracion personalizado
COPY haproxy.cfg /usr/local/etc/haproxy/haproxy.cfg
```

Para crear la imagen uso el siguiente comando

```
docker build -t mariorgzlpz-haproxy-image:p2 -f DockerFileHaproxy .
```

```

global
    stats socket /var/lib/haproxy/stats

defaults
    mode http

frontend frontend_MarioRgzLpz
    bind *:80
    default_backend backend_MarioRgzLpz

frontend stats
    mode http
    bind *:9000
    stats enable
    stats uri /estadisticas_mariorgzlpz
    stats refresh 10s
    stats realm HAProxy\ Statistics
    stats auth mariorgzlpz:SWAP1234

backend backend_MarioRgzLpz
    server web1 192.168.10.2:80 maxconn 32 check
    server web2 192.168.10.3:80 maxconn 32 check
    server web3 192.168.10.4:80 maxconn 32 check
    server web4 192.168.10.5:80 maxconn 32 check
    server web5 192.168.10.6:80 maxconn 32 check
    server web6 192.168.10.7:80 maxconn 32 check
    server web7 192.168.10.8:80 maxconn 32 check
    server web8 192.168.10.9:80 maxconn 32 check

```

Para lanzar el balanceador uso el siguiente comando

```
docker run -d -p 80:80 -p 9000:9000 --network p1_red_web --name balanceador-haproxy mariorgzlpz-haproxy
```

B5.2. Análisis propuesta IA

Preguntas a ChatGPT:

- Tengo el siguiente DockerFile (Dockerfile de haproxy) y la siguiente configuración (le paso la configuración del guion). Cuando creo la imagen y hago docker run para crear el contenedor y lanzarlo se cae al instante.
- Me va pero solo si accedo a localhost de nuevo no me basta con recargar la página.
- Me aparece error en la dirección donde deben verse las estadísticas estadísticas.
- En las estadísticas no me aparecen los contenedores en verde, me aparecen como not checked.

Respuestas de ChatGPT

- Parece que tu fichero de configuración de HAProxy tiene algunos fallos. Me dice que el nombre del frontend está mal.
- Añadir al frontend el mode http.
- Comprobar la configuración de haproxy, comprobar los puertos... (Ninguna de estas soluciones era la mía, simplemente tenía que abrir el puerto 9000 con el docker run)
- Esto podría ser porque no se ha configurado ninguna verificación de salud en el archivo de configuración de HAProxy (haproxy.cfg) o porque la configuración de las verificaciones de salud no está activa o definida correctamente.

Análisis

He tenido que realizar algunas consultas a ChatGPT porque me estaba fallando al crear el contenedor de haproxy y era debido a que faltaba un guion en frontend. También gracias a haber realizado el de nginx sabía solucionar los errores de puertos y de red al lanzar el contenedor de haproxy. Una vez montado me funcionaba pero no hacía lo que quería que era balancear la carga y esto era debido a que no había configurado el modo a http. También no me estaba funcionando las estadísticas y esto era debido a que no estaba abriendo el puerto 9000. Esto no lo detectaba ChatGPT y me lo solucionó el profesor.

B6. IMPLEMENTACION DEL ESCENARIO DE HAPROXY CON DOCKER COMPOSE

t

B6.1. Propuesta final

Uso el mismo archivo que me proporciona ChatGPT, simplemente añadiendo el puerto que necesito para las estadísticas.

B6.2. Análisis propuesta IA

Preguntas a ChatGPT:

-A partir de este docker compose (docker compose de nginx) y este DockerFile (dockerfile del apartado anterior) dame otro docker compose con los mismos 8 contenedores y redes, que modifique el balanceador para que sea balanceador-haproxy en lugar de balanceador-nginx y use la imagen haproxy.

Respuestas de ChatGPT

```
version: '4.0'

services:
  web1:
    image: mariorgzlpz--apache--image:p2
    volumes:
      - ./web_MarioRgzLpz:/var/www/localhost/htdocs
    container_name: web1
    networks:
      red_web:
        ipv4_address: 192.168.10.2
      red_servicios:
        ipv4_address: 192.168.20.2

  web2:
    image: mariorgzlpz--apache--image:p2
    volumes:
      - ./web_MarioRgzLpz:/var/www/localhost/htdocs
    container_name: web2
    networks:
      red_web:
        ipv4_address: 192.168.10.3
      red_servicios:
        ipv4_address: 192.168.20.3

  web3:
    image: mariorgzlpz--apache--image:p2
    volumes:
      - ./web_MarioRgzLpz:/var/www/localhost/htdocs
    container_name: web3
    networks:
      red_web:
        ipv4_address: 192.168.10.4
      red_servicios:
        ipv4_address: 192.168.20.4

  web4:
    image: mariorgzlpz--apache--image:p2
    volumes:
      - ./web_MarioRgzLpz:/var/www/localhost/htdocs
    container_name: web4
    networks:
      red_web:
        ipv4_address: 192.168.10.5
      red_servicios:
        ipv4_address: 192.168.20.5

  web5:
    image: mariorgzlpz--apache--image:p2
    volumes:
      - ./web_MarioRgzLpz:/var/www/localhost/htdocs
    container_name: web5
    networks:
      red_web:
        ipv4_address: 192.168.10.6
      red_servicios:
        ipv4_address: 192.168.20.6

  web6:
```



```

image: mariorgzlpz-apache-image:p2
volumes:
  - ./web_MarioRgzLpz:/var/www/localhost/htdocs
container_name: web6
networks:
  red_web:
    ipv4_address: 192.168.10.7
  red_servicios:
    ipv4_address: 192.168.20.7

web7:
image: mariorgzlpz-apache-image:p2
volumes:
  - ./web_MarioRgzLpz:/var/www/localhost/htdocs
container_name: web7
networks:
  red_web:
    ipv4_address: 192.168.10.8
  red_servicios:
    ipv4_address: 192.168.20.8

web8:
image: mariorgzlpz-apache-image:p2
volumes:
  - ./web_MarioRgzLpz:/var/www/localhost/htdocs
container_name: web8
networks:
  red_web:
    ipv4_address: 192.168.10.9
  red_servicios:
    ipv4_address: 192.168.20.9

balanceador-haproxy:
image: haproxy:latest
volumes:
  - ./haproxy.cfg:/usr/local/etc/haproxy/haproxy.cfg
container_name: balanceador-haproxy
ports:
  - 80:80
  # Anadido por mi
  - 9000:9000
networks:
  red_web:
    ipv4_address: 192.168.10.50
depends_on:
  - web1
  - web2
  - web3
  - web4
  - web5
  - web6
  - web7
  - web8

networks:
  red_web:
    driver: bridge
    ipam:
      config:
        - subnet: 192.168.10.0/24
          gateway: 192.168.10.1
  red_servicios:
    driver: bridge
    ipam:
      config:
        - subnet: 192.168.20.0/24
          gateway: 192.168.20.1

```

Análisis

El docker compose generado es correcto puesto que proporcionando el anterior es fácil deducir como tiene que ser el nuevo, lo único que falta es añadir el segundo puerto donde voy a mostrar las estadísticas y ya está listo para realizar las pruebas.

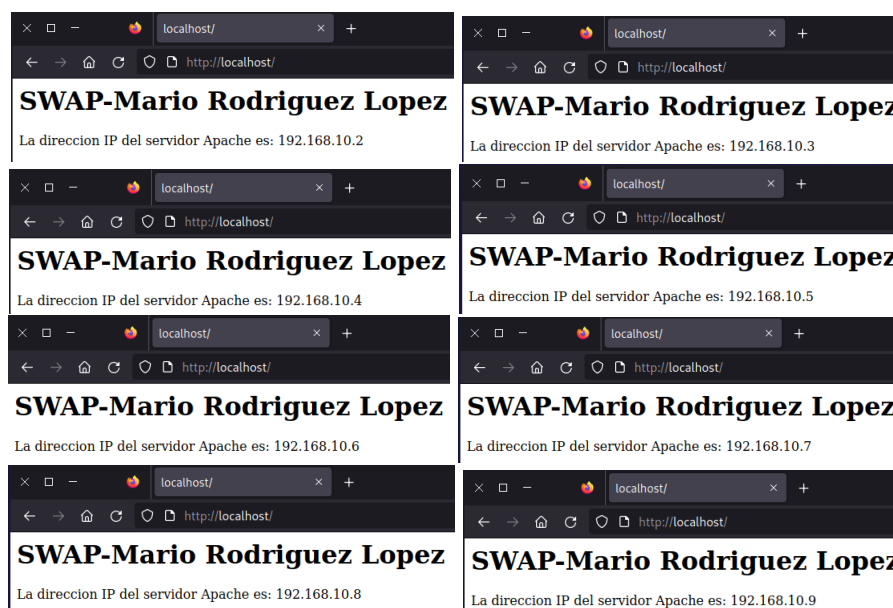
B7. VERIFICACION Y PRUEBAS DEL ESCENARIO DE HA-PROXY

Verificamos que todo funcione correcto con el docker compose y con docker ps mostramos que todos los contenedores esten activos

```
parrot: ~/.../P2/P2-MarioRgzLpz-nginx
- docker compose up
[+] Running 11/10
✓ Network p2-mariorgzlpz-nginx_red_web      Created
✓ Network p2-mariorgzlpz-nginx_red_servicios Created
✓ Container web6                            Created
✓ Container web7                            Created
✓ Container web5                            Created
✓ Container web3                            Created
✓ Container web1                            Created
✓ Container web8                            Created
✓ Container web2                            Created
✓ Container web4                            Created
✓ Container balanceador-nginx              Created
Attaching to balanceador-nginx, web1, web2, web3, web4, web5, web6, web7, web8

parrot: ~/.../P1/scripts
- docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS                               NAMES
b54a5d58ea40   mariorgzlpz-nginx-image:p2         "/docker-entrypoint..." 47 seconds ago Up 44 seconds 0.0.0.0:80->80/tcp, :::80->80/tcp balanceador-nginx
558e9458b185   mariorgzlpz-apache-image:p1        "httpd -D FOREGROUND"    47 seconds ago Up 45 seconds 80/tcp                               web7
5fae0d201468   mariorgzlpz-apache-image:p1        "httpd -D FOREGROUND"    47 seconds ago Up 44 seconds 80/tcp                               web3
1575f18976fe   mariorgzlpz-apache-image:p1        "httpd -D FOREGROUND"    47 seconds ago Up 44 seconds 80/tcp                               web5
ac7193121878   mariorgzlpz-apache-image:p1        "httpd -D FOREGROUND"    47 seconds ago Up 45 seconds 80/tcp                               web8
87dea6967357   mariorgzlpz-apache-image:p1        "httpd -D FOREGROUND"    47 seconds ago Up 45 seconds 80/tcp                               web2
4f3021c18cd1   mariorgzlpz-apache-image:p1        "httpd -D FOREGROUND"    47 seconds ago Up 44 seconds 80/tcp                               web4
e5c3389ee95b   mariorgzlpz-apache-image:p1        "httpd -D FOREGROUND"    47 seconds ago Up 44 seconds 80/tcp                               web1
25a5f11daa40   mariorgzlpz-apache-image:p1        "httpd -D FOREGROUND"    47 seconds ago Up 45 seconds 80/tcp                               web6
```

Comprobamos que todas las IPs se muestran correctamente y ordenadas mediante el algoritmo Round-Robin que es el que he configurado y que simplemente ira mandando peticiones empezando por la primera IP que hemos puesto en el fichero de configuración nginx hasta llegar a la ultima, de manera que la siguiente petición se enviaría de nuevo a la primera IP y así de manera sucesiva.



También comprobamos que las estadísticas se vean de manera correcta.

[←](#)
[→](#)
[🏠](#)
[↻](#)
[🔒](#)
[📄](#)
http://localhost:9000/estadisticas_mariorgzlpz

[🌟](#)
[∞](#)
[🔒](#)
[🔧](#)
[📁](#)
[☰](#)

HAProxy version 2.9.7-5742051, released 2024/04/05

Statistics Report for pid 8

> General process information

pid = 8 (process #1, nbproc = 1, nbthread = 16)
uptime = 0d 0h16m51s; warnings = 0
system limits: memmax = unlimited; ulimit-n = 1048576
maxconn = 1048576; **maxconn** = 524248; **reached** = 0; **maxpipes** = 0
 current conns = 1; current pipes = 0/0; conn rate = 1/sec; bit rate = 0.000 kbps
 Running tasks: 0/53 (0 niced); idle = 100 %

active UP

active UP, going down

active DOWN, going up

active or backup DOWN

active or backup DOWN for maintenance (MAINT)

active or backup SOFT STOPPED for maintenance

backup UP

backup UP, going down

backup DOWN, going up

not checked

for maintenance (MAINT)

Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

Display option:

- Scope:
- Hide "DOWN" servers
- Disable refresh
- Refresh now
- CSV export
- JSON export (schema)

External resources:

- Primary site
- Updates (v2.9)
- Online manual

frontend_MarioRgzLpz

	Queue			Session rate			Sessions				Bytes				Denied		Errors			Warnings		Server											
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntime	Thrtle			
Frontend	0	1	-	0	1	1	524 248		1				1 361	682	0	0	0	0	0	0	0	0	OPEN										

stats

	Queue			Session rate			Sessions				Bytes				Denied		Errors			Warnings		Server											
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntime	Thrtle			
Frontend	1	2	-	1	13	1	524 248		133				88 909	4 617 184	0	0	0	0	0	0	0	0	OPEN										

backend_MarioRgzLpz

	Queue			Session rate			Sessions				Bytes				Denied		Errors			Warnings		Server									
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntime	Thrtle	
web1	0	0	-	0	0	1	0	1	32	1	11m36s	718 255	0	0	0	0	0	0	0	0	0	0	16m51s UP	L4OK in 0ms	1/1	Y	-	0	0	0s	-
web2	0	0	-	0	1	1	0	1	32	1	11m36s	643 427	0	0	0	0	0	0	0	0	0	0	16m51s UP	L4OK in 0ms	1/1	Y	-	0	0	0s	-
web3	0	0	-	0	0	0	0	0	32	0	0	?	0	0	0	0	0	0	0	0	0	0	16m51s UP	L4OK in 0ms	1/1	Y	-	0	0	0s	-
web4	0	0	-	0	0	0	0	0	32	0	0	?	0	0	0	0	0	0	0	0	0	0	16m51s UP	L4OK in 0ms	1/1	Y	-	0	0	0s	-
web5	0	0	-	0	0	0	0	0	32	0	0	?	0	0	0	0	0	0	0	0	0	0	16m51s UP	L4OK in 0ms	1/1	Y	-	0	0	0s	-
web6	0	0	-	0	0	0	0	0	32	0	0	?	0	0	0	0	0	0	0	0	0	0	16m51s UP	L4OK in 0ms	1/1	Y	-	0	0	0s	-
web7	0	0	-	0	0	0	0	0	32	0	0	?	0	0	0	0	0	0	0	0	0	0	16m51s UP	L4OK in 0ms	1/1	Y	-	0	0	0s	-
web8	0	0	-	0	0	0	0	0	32	0	0	?	0	0	0	0	0	0	0	0	0	0	16m51s UP	L4OK in 0ms	1/1	Y	-	0	0	0s	-
Backend	0	0	-	0	2	1	52 425		2	2	11m36s	1																			

Para este apartado no he usado IA puesto que era solo realizar comprobaciones.

A1. CONFIGURACIONES AVANZADAS DE NGINX

A1.1. Propuesta final configuración usando least_conn

Mi propopuesta es la misma que la de ChatGPT.

Muestro un enlace probando que la configuración funciona y balancea de manera correcta siguiendo los pesos.

Link a drive

A1.2. Análisis propuesta IA

Preguntas a ChatGPT:

Esta es mi configuración de nginx (le paso la configuracion anterior) proporcioname una configuración que en lugar de funcionar con round robin normal funcione con pesos y de manera ponderada.

Respuestas de ChatGPT

```
events {
    worker_connections 1024;
}

http {
    upstream backend_MarioRgzLpz {
        # Pesos modificados por mi
        server 192.168.10.2 weight=1;
        server 192.168.10.3 weight=1;
        server 192.168.10.4 weight=8;
        server 192.168.10.5 weight=1;
        server 192.168.10.6 weight=1;
        server 192.168.10.7 weight=1;
        server 192.168.10.8 weight=8;
        server 192.168.10.9 weight=1;
    }

    server {
        listen 80;
        server_name nginx_MarioRgzLpz;
        access_log /var/log/nginx/nginx_MarioRgzLpz.access_log;
```

```

error_log /var/log/nginx/nginx_MarioRgzLpz.error_log;

location / {
    proxy_pass http://backend.MarioRgzLpz;
    proxy_set_header Cookie $http_cookie;
    proxy_hide_header Set-Cookie;
}

location /estadisticas_MarioRgzLpz {
    stub_status on;
}
}

```

Análisis

La propuesta que me da ChatGPT para la configuración es válida y me funciona, lo único que he modificado ha sido el peso de los servidores para que sea mas notable a la hora de las comprobaciones que estamos usando dicho algoritmo de balanceo.

A2. CONFIGURACIONES AVANZADAS DE HAPROXY

A2.1. Propuesta final configuración usando pesos

La propuesta es la misma que la de ChatGPT.

En este caso no muestro el funcionamiento puesto que es igual que el de nginx y en el apartado de verificación de errores muestro algo del balanceo avanzado con haproxy.

A2.2. Análisis propuesta IA

Preguntas a ChatGPT:

Esta es mi configuración de haproxy (le paso la configuración anterior) proporcioname una configuración que en lugar de funcionar con round robin normal funcione con pesos y de manera ponderada.

Respuestas de ChatGPT

```

global
    stats socket /var/lib/haproxy/stats

defaults
    mode http

frontend frontend_MarioRgzLpz
    bind *:80
    default_backend backend_MarioRgzLpz

frontend stats
    mode http
    bind *:9000
    stats enable
    stats uri /estadisticas_mariorgzlpz
    stats refresh 10s
    stats realm HAProxy\ Statistics
    stats auth mariorgzlpz:SWAP1234

backend backend_MarioRgzLpz
    mode http
    #Pesos modificados por mi
    server web1 192.168.10.2:80 weight 1 maxconn 32 check
    server web2 192.168.10.3:80 weight 8 maxconn 32 check
    server web3 192.168.10.4:80 weight 1 maxconn 32 check
    server web4 192.168.10.5:80 weight 1 maxconn 32 check
    server web5 192.168.10.6:80 weight 8 maxconn 32 check
    server web6 192.168.10.7:80 weight 1 maxconn 32 check
    server web7 192.168.10.8:80 weight 1 maxconn 32 check
    server web8 192.168.10.9:80 weight 1 maxconn 32 check

```

Análisis

La propuesta que me da ChatGPT para la configuración es válida y me funciona, lo único que he modificado ha sido el peso de los servidores para que sea mas notable a la hora de las comprobaciones que estamos usando dicho algoritmo de balanceo.

A3. EXPERIMENTACIÓN CON DIFERENTES BALANCEADORES DE CARGA

A3.1. Balanceo usando traefik

A3.1.1 Propuesta final realizada

Nada de lo que me proporciona ChatGPT me ha servido por lo que he optado por realizar la configuración de traefik usando Youtube [Traefik tutorial](#) y la documentación oficial de traefik [Traefik doc](#). Con esto he configurado Traefik como sigue:

DockerFile de traefik y creacion de la imagen.

```
#Imagen base
FROM traefik:latest
```

```
docker build -t mariorgzlpz-traefik-image:p2 -f DockerFileTraefik .
```

Docker compose final:

```
version: '4.0'

services:
  web1:
    image: mariorgzlpz-apache-image:p2
    volumes:
      - ./web_MarioRgzLpz:/var/www/localhost/htdocs
    container_name: web1
    networks:
      red_web:
        ipv4_address: 192.168.10.2
      red_servicios:
        ipv4_address: 192.168.20.2
    depends_on:
      - balanceador-traefik

  web2:
    image: mariorgzlpz-apache-image:p2
    volumes:
      - ./web_MarioRgzLpz:/var/www/localhost/htdocs
    container_name: web2
    networks:
      red_web:
        ipv4_address: 192.168.10.3
      red_servicios:
        ipv4_address: 192.168.20.3
    depends_on:
      - balanceador-traefik

  web3:
    image: mariorgzlpz-apache-image:p2
    volumes:
      - ./web_MarioRgzLpz:/var/www/localhost/htdocs
    container_name: web3
    labels:
      - "traefik.http.routers.web3.rule=Host('localhost')"
    networks:
      red_web:
        ipv4_address: 192.168.10.4
      red_servicios:
        ipv4_address: 192.168.20.4
    depends_on:
      - balanceador-traefik

  web4:
    image: mariorgzlpz-apache-image:p2
```

```

volumes:
  - ./web_MarioRgzLpz:/var/www/localhost/htdocs
container_name: web4
labels:
  - "traefik.http.routers.web4.rule=Host('localhost')"
networks:
  red_web:
    ipv4_address: 192.168.10.5
  red_servicios:
    ipv4_address: 192.168.20.5
depends_on:
  - balanceador-traefik

web5:
  image: mariorgzlpz-apache-image:p2
  volumes:
    - ./web_MarioRgzLpz:/var/www/localhost/htdocs
  container_name: web5
  networks:
    red_web:
      ipv4_address: 192.168.10.6
    red_servicios:
      ipv4_address: 192.168.20.6
  depends_on:
    - balanceador-traefik

web6:
  image: mariorgzlpz-apache-image:p2
  volumes:
    - ./web_MarioRgzLpz:/var/www/localhost/htdocs
  container_name: web6
  networks:
    red_web:
      ipv4_address: 192.168.10.7
    red_servicios:
      ipv4_address: 192.168.20.7
  depends_on:
    - balanceador-traefik

web7:
  image: mariorgzlpz-apache-image:p2
  volumes:
    - ./web_MarioRgzLpz:/var/www/localhost/htdocs
  container_name: web7
  networks:
    red_web:
      ipv4_address: 192.168.10.8
    red_servicios:
      ipv4_address: 192.168.20.8
  depends_on:
    - balanceador-traefik

web8:
  image: mariorgzlpz-apache-image:p2
  volumes:
    - ./web_MarioRgzLpz:/var/www/localhost/htdocs
  container_name: web8
  networks:
    red_web:
      ipv4_address: 192.168.10.9
    red_servicios:
      ipv4_address: 192.168.20.9
  depends_on:
    - balanceador-traefik

balanceador-traefik:
  image: mariorgzlpz-traefik-image:p2
  ports:
    - "80:80"
    - "8080:8080" # Para el dashboard de Traefik
  volumes:
    - "/var/run/docker.sock:/var/run/docker.sock:ro"

```

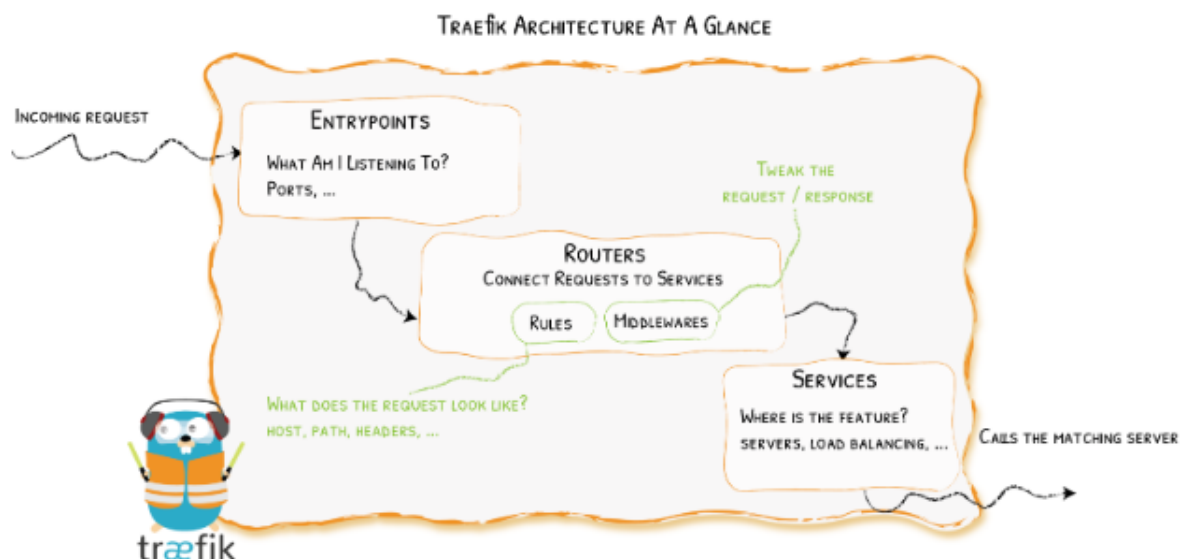
```

- "/dynamic.yml:/etc/traefik/conf.d/dynamic.yml"
- "/traefik.yml:/etc/traefik/traefik.yml"
container_name: balanceador-traefik
networks:
  red_web:
    ipv4_address: 192.168.10.50

networks:
  red_web:
    driver: bridge
    ipam:
      config:
        - subnet: 192.168.10.0/24
          gateway: 192.168.10.1
  red_servicios:
    driver: bridge
    ipam:
      config:
        - subnet: 192.168.20.0/24
          gateway: 192.168.20.1

```

Configuraciones de traefik. Para esto he seguido la foto que nos da la pagina sobre lo que vamos a configurar y que es cada cosa.



Todo esto lo realizamos en dos ficheros uno es el que se carga por defecto llamado traefik.yml y el otro será el que indicamos como provider en el mismo fichero traefik.yml que es la configuración dynamic.yml. El fichero traefik.yml me ha quedado como sigue:

```

api:
  insecure: true # Permite mostrar el dashboard de Traefik en HTTP

entryPoints:
  web:
    address: ":80" # Puerto por el que escucha Traefik

providers:
  file:
    directory: /etc/traefik/conf.d # Directorio donde se encuentran los ficheros de
    configuracion dinamica
    watch: true # Habilita la deteccion de cambios en los ficheros de configuracion
    dinamica

```

Y la configuracion del fichero dynamic.yml me ha quedado como sigue:

```

http:
  routers: # Routers configuration
    Router-1: # Llamo a mi router Router-1
      rule: "Host('localhost')" # Defino la regla de mi router, que sera en localhost es
      decir donde se encuentra mi balanceador de carga

```

```

service: "web-service" # Especifico el servicio que va a utilizar mi router

services: # Services configuration
web-service: # Llamo a mi servicio web-service
loadBalancer: # Defino el balanceador de carga
  servers: # Defino los servidores que tendra mi balanceador de carga
    - url: "http://web1:80" # Defino las urls de los contenedores que se encuentran en mi red
    - url: "http://web2:80"
    - url: "http://web3:80"
    - url: "http://web4:80"
    - url: "http://web5:80"
    - url: "http://web6:80"
    - url: "http://web7:80"
    - url: "http://web8:80"

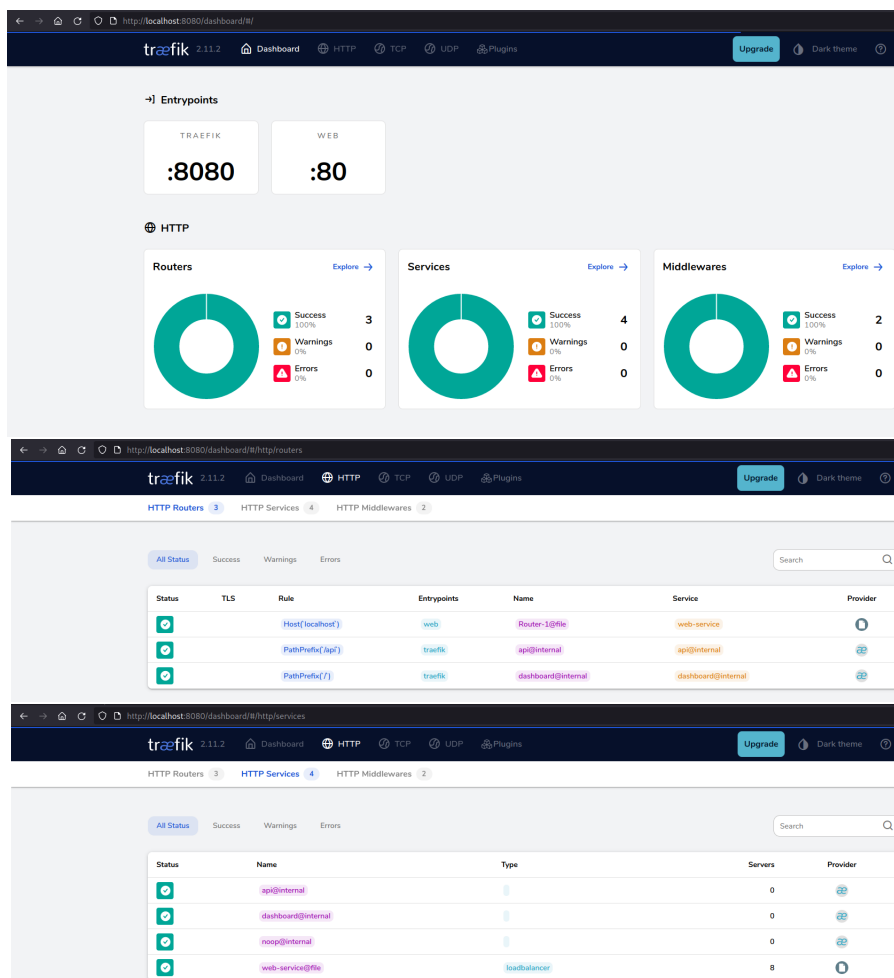
```

Con estos ficheros de configuracion simplemente levantando todos los contenedores podremos ver como en localhost vemos los servidores web y actuando como balanceador (Como no hemos definido uno especifico sera round-robin y además como los contenedores no se crean en orden iran apareciendo uno a uno con el orden de creación). Además en localhost:8080 tendremos el dashboard en el que podemos ver los routers y servicios que tenemos.

Muestra de localhost:



Muestra del dashboard



A3.1.2 Análisis propuesta IA

Preguntas a ChatGPT:

-Tengo el siguiente docker compose (docker compose de nginx) quiero en lugar de usar nginx como balanceador, usar traefik y que en localhost en el puerto 80 funcione como balanceador de carga y que en el puerto 8080 este el dashboard con las estadísticas.

-Me aparece error 404 al ingresar tanto a localhost como a las estadísticas.

- Proporciono los logs del balanceador:

docker logs balanceador-traefik

time="2024-04-27T11:08:47Z" level=info msg= "Configuration loaded from flags."

Respuestas de ChatGPT

—Para lograr lo que deseas, necesitas hacer algunos cambios en tu archivo Docker Compose para integrar Traefik como el balanceador de carga y configurar el dashboard en el puerto 8080. Aquí te muestro cómo podrías hacerlo:

Agrega el servicio de Traefik al archivo Docker Compose y actualiza la configuración de tus servicios web para que Traefik pueda descubrirlos y enrutar el tráfico:

```
services:
  traefik:
    image: traefik:v2.5
    container_name: traefik
    command:
      - "--api.insecure=true"
      - "--providers.docker=true"
      - "--providers.docker.exposedbydefault=false"
      - "--entrypoints.web.address=:80"
      - "--entrypoints.dashboard.address=:8080"
    ports:
      - "80:80"
      - "8080:8080"
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
  web1:
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.web1.rule=Host('web1.example.com')"
      - "traefik.http.services.web1.loadbalancer.server.port=80"
  web2:
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.web2.rule=Host('web2.example.com')"
      - "traefik.http.services.web2.loadbalancer.server.port=80"
# Repite para el resto de servicios web...
```

—Si estás viendo un error 404 tanto en localhost:80 como en las estadísticas en localhost:8080, podría haber varias razones para esto:

Configuración de los servicios web en Traefik: Asegúrate de que los servicios web estén etiquetados correctamente en el archivo Docker Compose para que Traefik los pueda descubrir y enrutar el tráfico correctamente. Revisa las etiquetas traefik.enable, traefik.http.routers, y traefik.http.services para cada servicio web.

Configuración de los nombres de host: Verifica que estés accediendo a los servicios web utilizando los nombres de host correctos. En la configuración proporcionada, los servicios web están configurados para ser accedidos a través de nombres de host como web1.example.com, web2.example.com, etc. Asegúrate de que estés accediendo a ellos utilizando estos nombres de host o cambia la configuración a tu localhost.

Conflictos de puertos: Asegúrate de que ningún otro servicio esté utilizando los puertos 80 y 8080 en tu máquina local. Si hay conflictos de puertos, Traefik no podrá iniciar correctamente y verás errores 404.

Errores de sintaxis en la configuración de Traefik: Revisa cuidadosamente la configuración del servicio Traefik en tu archivo Docker Compose para asegurarte de que no haya errores de sintaxis que puedan causar que Traefik no funcione correctamente.

Logs de Traefik: Verifica los registros de Traefik para obtener más información sobre cualquier error que pueda estar ocurriendo. Puedes acceder a los registros utilizando el comando docker logs traefik.

—El mensaje Configuration loaded from flags indica que Traefik se ha iniciado correctamente y ha cargado su configuración desde los flags proporcionados en el comando de inicio.

Dado que Traefik se ha iniciado sin problemas y no hay errores evidentes en los registros proporcionados, es posible que el problema resida en la configuración de los servicios web o en la forma en que estás

accediendo a ellos.

Aquí hay algunas acciones adicionales que podrías tomar para investigar más a fondo el problema:

Verificar la configuración de los servicios web: Revisa la configuración de cada servicio web en tu archivo Docker Compose para asegurarte de que estén etiquetados correctamente y que Traefik los esté descubriendo correctamente. Asegúrate de que los servicios web estén etiquetados con `traefik.enable=true`, `traefik.http.routers`, y `traefik.http.services` de manera adecuada.

Comprobar los nombres de host y puertos: Verifica que estás accediendo a los servicios web utilizando los nombres de host y puertos correctos. Si has configurado los servicios web para ser accedidos a través de nombres de host como `web1.example.com`, asegúrate de que estás accediendo a ellos utilizando estos nombres de host.

Revisar los logs de los servicios web: Verifica los logs de los contenedores de los servicios web para ver si están recibiendo tráfico y si hay algún error específico que pueda estar causando el error 404.

Reiniciar los servicios web: Intenta reiniciar los servicios web y Traefik para ver si eso resuelve el problema. Si después de realizar estas acciones aún no puedes resolver el problema, sería útil revisar más detenidamente la configuración de tus servicios web y los logs específicos de los contenedores para identificar la causa del error 404.

Analisis

Como vemos en este caso ChatGPT no ha sido la solución para lo que nosotros pedíamos puesto que traefik es algo bastante moderno y del cual ChatGPT no tiene tanta información, dandonos una configuración bastante engorrosa y no siendo de tanta ayuda. Simplemente siguiendo el tutorial de la documentación, adaptandolo a nuestro caso y entendiendo que hace cada cosa logramos lo que queremos de manera más rápida.

A3.2. Balanceo usando Gobetween

A3.2.1 Propuesta final realizada

Nada de lo que me proporciona ChatGPT me ha servido por lo que he optado por realizar la configuración de traefik usando la documentación oficial de gobetween [Gobetween doc](#). Con esto y viendo la imagen que hay de gobetween en docker hub he configurado como sigue

DockerFile de gobetween y creación de la imagen.

```
# Utiliza la imagen base de gobetween
FROM yyyar/gobetween:latest
```

```
docker build -t mariorgzlpz-gobetween-image:p2 -f DockerFileGobetween .
```

Docker compose final:

```
version: '4.0'

services:
  web1:
    image: mariorgzlpz-apache-image:p2
    volumes:
      - ./web_MarioRgzLpz:/var/www/localhost/htdocs
    container_name: web1
    networks:
      red_web:
        ipv4_address: 192.168.10.2
      red_servicios:
        ipv4_address: 192.168.20.2

  web2:
    image: mariorgzlpz-apache-image:p2
    volumes:
      - ./web_MarioRgzLpz:/var/www/localhost/htdocs
    container_name: web2
    networks:
      red_web:
        ipv4_address: 192.168.10.3
      red_servicios:
        ipv4_address: 192.168.20.3
```

```

web3:
  image: mariorgzlpz--apache--image:p2
  volumes:
    - ./web_MarioRgzLpz:/var/www/localhost/htdocs
  container_name: web3
  networks:
    red_web:
      ipv4_address: 192.168.10.4
    red_servicios:
      ipv4_address: 192.168.20.4

web4:
  image: mariorgzlpz--apache--image:p2
  volumes:
    - ./web_MarioRgzLpz:/var/www/localhost/htdocs
  container_name: web4
  networks:
    red_web:
      ipv4_address: 192.168.10.5
    red_servicios:
      ipv4_address: 192.168.20.5

web5:
  image: mariorgzlpz--apache--image:p2
  volumes:
    - ./web_MarioRgzLpz:/var/www/localhost/htdocs
  container_name: web5
  networks:
    red_web:
      ipv4_address: 192.168.10.6
    red_servicios:
      ipv4_address: 192.168.20.6

web6:
  image: mariorgzlpz--apache--image:p2
  volumes:
    - ./web_MarioRgzLpz:/var/www/localhost/htdocs
  container_name: web6
  networks:
    red_web:
      ipv4_address: 192.168.10.7
    red_servicios:
      ipv4_address: 192.168.20.7

web7:
  image: mariorgzlpz--apache--image:p2
  volumes:
    - ./web_MarioRgzLpz:/var/www/localhost/htdocs
  container_name: web7
  networks:
    red_web:
      ipv4_address: 192.168.10.8
    red_servicios:
      ipv4_address: 192.168.20.8

web8:
  image: mariorgzlpz--apache--image:p2
  volumes:
    - ./web_MarioRgzLpz:/var/www/localhost/htdocs
  container_name: web8
  networks:
    red_web:
      ipv4_address: 192.168.10.9
    red_servicios:
      ipv4_address: 192.168.20.9

```

```

balanceador-gobetween:
  image: mariorgzlpz-gobetween-image:p2
  container_name: balanceador-gobetween
  volumes:
    - ./gobetween.toml:/etc/gobetween/conf/gobetween.toml
  ports:
    - "80:80"
  networks:
    red_web:
      ipv4_address: 192.168.10.50
  depends_on:
    - web1
    - web2
    - web3
    - web4
    - web5
    - web6
    - web7
    - web8

networks:
  red_web:
    driver: bridge
    ipam:
      config:
        - subnet: 192.168.10.0/24
          gateway: 192.168.10.1
  red_servicios:
    driver: bridge
    ipam:
      config:
        - subnet: 192.168.20.0/24
          gateway: 192.168.20.1

```

Fichero de configuración gobetween.toml para el cual tengo un volumen. Es el mismo fichero

```

[servers.sample]
bind = "0.0.0.0:80"
protocol = "tcp"
balance = "roundrobin"

max_connections = 10000
client_idle_timeout = "10s"
backend_idle_timeout = "10s"
backend_connection_timeout = "20s"

[servers.sample.discovery]
kind = "static"
static_list = [
  "192.168.10.2:80",
  "192.168.10.3:80",
  "192.168.10.4:80",
  "192.168.10.5:80",
  "192.168.10.6:80",
  "192.168.10.7:80",
  "192.168.10.8:80",
  "192.168.10.9:80"
]

[servers.sample.healthcheck]
fails = 1
passes = 1
interval = "2s"
timeout = "1s"
kind = "ping"
ping_timeout_duration = "500ms"

```

Con este fichero de configuración, que es igual que el de la documentación pero modificando las IPs, los puertos y los tiempos, ya me funcionaba gobetween en el localhost como balanceador.

A3.1.2 Análisis propuesta IA

Preguntas a ChatGPT:

-Tengo el siguiente docker compose (docker compose de nginx) quiero en lugar de usar nginx como balanceador, usar gobetween y que en localhost en el puerto 80 funcione como balanceador de carga.

Respuestas de ChatGPT

-Para usar GoBetween como balanceador de carga en lugar de Nginx en tu entorno Docker Compose, primero necesitarás hacer algunos cambios en tu archivo docker-compose.yml y luego configurar GoBetween para que funcione como balanceador de carga.

Aquí tienes un ejemplo de cómo podría ser tu archivo docker-compose.yml modificado para usar GoBetween:

```
version: '3.8' # Cambiado a la version 3.8 para compatibilidad con GoBetween

services:
  web1:
    image: mariorgzlpz-apache-image:p2
    volumes:
      - ./web_MarioRgzLpz:/var/www/localhost/htdocs
    container_name: web1
    networks:
      red_web:
        ipv4_address: 192.168.10.2
      red_servicios:
        ipv4_address: 192.168.20.2

  web2:
    image: mariorgzlpz-apache-image:p2
    volumes:
      - ./web_MarioRgzLpz:/var/www/localhost/htdocs
    container_name: web2
    networks:
      red_web:
        ipv4_address: 192.168.10.3
      red_servicios:
        ipv4_address: 192.168.20.3

# Resto de servicios web...

gobetween:
  image: 'gobetween/gobetween:v0.9.0' # Imagen de GoBetween
  command: ["gobetween", "-c", "/etc/gobetween/gobetween.toml"] # Comando para
  ejecutar GoBetween con una configuracion personalizada
  volumes:
    - ./gobetween.toml:/etc/gobetween/gobetween.toml # Archivo de configuracion de
  GoBetween
  container_name: gobetween
  ports:
    - "80:80" # Puerto expuesto para el trafico HTTP
  networks:
    - red_web

networks:
  red_web:
    driver: bridge
    ipam:
      config:
        - subnet: 192.168.10.0/24
  red_servicios:
    driver: bridge
    ipam:
      config:
        - subnet: 192.168.20.0/24
```

Luego, necesitarás crear un archivo de configuración gobetween.toml para GoBetween. Aquí hay un ejemplo básico:

```
# gobetween.toml

[gobetween]
debug = false

[[tcp]]
addr = ":80" # Puerto que escucha GoBetween
protocol = "http"
```

```
[[tcp.services]]
balance = "roundrobin" # Algoritmo de balanceo de carga
[[tcp.services.backends]]
host = "192.168.10.2:80" # Direccion IP y puerto del primer contenedor web
weight = 1 # Peso relativo de este backend en el balanceo de carga

[[tcp.services.backends]]
host = "192.168.10.3:80" # Direccion IP y puerto del segundo contenedor web
weight = 1 # Peso relativo de este backend en el balanceo de carga

# Repite estas secciones para cada contenedor web adicional
```

Análisis

Viendo que lo que me proporcionaba ChatGPT no me funcionaba decidí que era más rápido buscar directamente en foros o en documentación y simplemente seguí los pasos de la documentación cambiando algunas cosas como el puerto del localhost y añadir las IPs de mis servidores web. Con esto y creando el servicio desde el docker compose(añadiéndole el volumen y la IP de la red) ya me permite acceder a localhost y ver como redirecciona la carga.

A4. INVESTIGACIÓN Y PRUEBAS DE TOLERANCIA A FALLAS

A4.1 Propuesta realizada y pruebas

He pensado en hacerlo en haproxy puesto que me permite ver las estadísticas de manera gráfica y así ver el estado de los contenedores en todo momento. Uso la configuracion avanzada con la cual el balanceo funciona mediante pesos.

Lo primero es lo mismo que obtengo en la verificación de haproxy de la tarea básica B7 por lo tanto no lo vuelvo a mostrar.

Comencemos parando alguno de los contenedores y viendo lo que sucede.

docker stop web1

HAProxy version 2.9.7-5742051, released 2024/04/05

Statistics Report for pid 8

> General process information

<

HAProxy version 2.9.7-5742051, released 2024/04/05

Statistics Report for pid 8

> General process information

pid = 8 (process 11, rtlibc = 1, threaded = 16) uptime = 0d 0h05m29s, warnings = 0 system limits: memmax = unlimited, ulimit-s = 1048576 maxsock = 1048576, maxconn = 524288, reached = 0, maxpipes = 0 current cores = 1, current pipes = 0, conn rate = 0/sec, bit rate = 0.000 kbps Running tasks: 0/52 (0 ticks), idle = 100 %		<div>active UP, going down active DOWN, going up active or backup DOWN active or backup UP for maintenance (MAINT) active or backup SOFT STOPPED for maintenance</div>		<div>Out Req Resp Req Resp Req Resp Req Resp</div>		<div>backlog UP backlog DOWN, going down backlog DOWN, going up (not checked) backlog or backup DOWN for maintenance (MAINT) backlog or backup SOFT STOPPED for maintenance</div>		<div>Errors Conn Resp Retr Redis</div>		<div>Warnings Status LastChk</div>		<div>Server Wght Act Bck Chk Dwn Downtime Thrtle</div>		<div>Display options: Scope: • Errors all • Warnings all • Status • Details • CPU usage • JSON export (schema)</div>		<div>External resources: • Errors all • Warnings all • Status • Details • CPU usage • JSON export (schema)</div>																			
Frontend Main00000000																																			
		Queue		Session rate		Sessions		Bytes		Denied		Errors		Warnings		Status		LastChk		Wght		Act		Bck		Chk		Dwn		Downtime		Thrtle			
Frontend		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	
Frontend		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Stats		Queue	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	
Frontend		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Backend Main00000000																																			
		Queue		Session rate		Sessions		Bytes		Denied		Errors		Warnings		Status		LastChk		Server		Wght		Act		Bck		Chk		Dwn		Downtime		Thrtle	
web1		0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
web2		0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
web3		0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
web4		0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
web5		0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
web6		0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
web7		0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
web8		0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Backend		0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
Backend		0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Como vemos nos aparece que web1 esta caido. Sin embargo si accedemos a localhost nos encontramos con que sigue funcionando perfectamente, aunque como solo nos falta un contenedor esto no es tan notable.

```
docker stop web2
```

Statistics Report for pid 8

➤ General process information

pid = 8 (process #1, ebrpcn = 1, offsetted = 16)

uptime = 0.000000 seconds = 0.7

system brkde: memmax = unlimited, limit = 1048576

active = 0.000000, swapon = 0.000000, reached = 0, swappages = 0

current cores = 1, current pages = 0.0, core rate = 0.0sec rate = 0.000 klops

Running tasks: OSG (0 vnodes), idle = 100 %

active UP

active UP, going down

active UP, going up

active or backup DOWN

active or backup DOWN for maintenance (MAINT)

active or backup STOP STOPPED for maintenance

NOT "UP" UP with load-balancing disabled

backup UP

backup UP, going down

backup DOWN, going up

just checked

not checked

STOPPED

Display option:

Scope:

Hide DOWN events

Exclude status

Refresh rate

CSV export

JSON export (scheme)

External resources:

Process logs

Logfiles (2/2)

Online manual

Frontend_Monitoring.gd

Cur	Queue	Max	Limit	Session rate	Max	Limit	Cur	Max	Limit	Sessions	Total	LaTot	Last	In	Bytes	Out	Denied	Req	Resp	Req	Errors	Conn	Resp	Retr	Warnings	Reids	Status	LastChk	Wght	Act	Server	Box	Chk	Down	Downtime	Thrtle
Frontend				0	1	-	0	4	1	524 248	1				5 985	2 295	0	0	0	0	0	0	0	0				OPEN								

init

Cur	Queue	Max	Limit	Session rate	Max	Limit	Cur	Max	Limit	Sessions	Total	LaTot	Last	In	Bytes	Out	Denied	Req	Resp	Req	Errors	Conn	Resp	Retr	Warnings	Reids	Status	LastChk	Wght	Act	Server	Box	Chk	Down	Downtime	Thrtle
Frontend				0	1	-	1	7	7	524 248	5				32 281	1 460 318	0	0	0	0	0	0	0	0				OPEN								

Backend_Monitoring.gd

Cur	Queue	Max	Limit	Session rate	Max	Limit	Cur	Max	Limit	Sessions	Total	LaTot	Last	In	Bytes	Out	Denied	Req	Resp	Req	Errors	Conn	Resp	Retr	Warnings	Reids	Status	LastChk	Wght	Act	Server	Box	Chk	Down	Downtime	Thrtle
web1	0	0	-	0	0	0	0	0	32	0	32	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	5m50s DOWN	14TOUT in 2003ms	11	Y	-	6	2	6m14s	-	
web2	0	0	-	0	0	3	0	1	32	4	4	5m50s	2 660	1 000	0	0	0	0	0	0	0	0	0	0	0	7s DOWN	7s DOWN	6m8	Y	-	3	1	7s	-		
web3	0	0	-	0	1	1	32	5m50s	660	250	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	140K in 2003ms	11	Y	-	0	0	0	0s	-		
web4	0	0	-	0	0	0	0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6m16s UP	140K in 0ms	11	Y	-	0	0	0s	-		
web5	0	0	-	0	1	1	32	4	4	5m50s	2 660	1 000	0	0	0	0	0	0	0	0	0	0	0	0	0	6m16s UP	140K in 0ms	11	Y	-	0	0	0s	-		
web6	0	0	-	0	0	0	0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6m16s UP	140K in 0ms	11	Y	-	0	0	0s	-		
web7	0	0	-	0	0	0	0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6m16s UP	140K in 0ms	11	Y	-	0	0	0s	-		
web8	0	0	-	0	0	0	0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6m16s UP	140K in 0ms	11	Y	-	0	0	0s	-		
Backend	0	0	-	0	0	0	0	524 476	0	0	5m50s	5 985	2 295	0	0	0	0	0	0	0	0	0	0	0	0	6m16s UP	140K in 0ms	11	Y	-	0	0	0s	-		

Link a drive

```
docker start web1
```

```
docker start web2
```

Y viendo en las estadísticas de haproxy vemos lo siguiente

Statistics Report for pid 8

[illegible]

A4.2 Análisis propuesta IA

Pruebas de tolerancia a fallos apagando intencionadamente instancias de servidor web para observar la reacción y la reasignación de carga de los balanceadores.

Ejecutar el entorno Docker Compose: Asegúrate de tener Docker instalado en tu sistema y luego ejecuta tu Docker Compose como lo haces normalmente.

```
docker compose up
```

23

```
docker ps
```

Apagar intencionadamente un contenedor: Elige uno de los contenedores web y apágalo manualmente. Puedes hacerlo usando el comando `docker stop` seguido del nombre del contenedor.

```
docker stop nombre_del_contenedor
```

Por ejemplo, si quieres apagar `web1`, ejecutarías:

Verificar la reacción del balanceador: Después de apagar un contenedor, observa cómo reacciona el balanceador. Debería redistribuir automáticamente la carga a los contenedores web restantes que aún están en funcionamiento. Puedes verificar esto accediendo a través del balanceador de carga a la dirección IP `192.168.10.50` en un navegador web y viendo si los sitios web aún están disponibles y si se han redistribuido las solicitudes entre los contenedores web que siguen en línea.

Repetir el proceso: Repite los pasos 3 y 4 para apagar otros contenedores web y observar cómo reacciona el balanceador en cada caso.

Restaurar los contenedores (opcional): Después de realizar tus pruebas de tolerancia a fallos, puedes volver a encender los contenedores que apagaste manualmente.

```
docker start nombre_del_contenedor
```

Analisis He realizado lo que me ha dicho ChatGPT porque me ha parecido algo similar a lo que podría ocurrir como escenario de fallo de un servidor en el cual se nos cae o se estropea alguno en concreto y como actuaría el balanceador en ese caso.

A5. AUTOMATIZACIÓN DE ESCALADO DEL ESCENARIO

Los pasos a seguir para probar el escalado son:

Levantar solo 4 contenedores al principio (mínimo vamos a tener 3 activos siempre) para ver si es necesario añadir más o quitar alguno, esto lo haces usando el siguiente comando `docker`:

```
docker compose up -d balanceador-nginx cadvisor web1 web2 web3 web4
```

Tras esto obtenemos las métricas que se guardaran en el archivo `datos.json`:

```
python3 datos.py
```

Con los datos ya podremos usar el script de escalado.

```
python3 logica_escalado.py
```

Aquí muestro los resultados de algunas pruebas:

```
parrot: ~/.../P2/Escalado
● → python3 datos.py
Datos guardados en: ./data/datos.json
```

```
parrot: ~/.../P2/Escalado
● → python3 logica_escalado.py
Media de memoria: 12040192.0
Media de uso de CPU: 45094000.0
Verificando 4 contenedores...
Contenedor web1:
  - Memoria: 11341824
  - Uso de CPU total: 44167000
Umbral de memoria por arriba: 12642201.6
Umbral de CPU por arriba: 47348700.0
Umbral de memoria por debajo: 11919790.08
Umbral de CPU por debajo: 44643060.0
El contenedor web1 no supera el umbral de memoria o uso de CPU. Parando el ultimo contenedor.
[+] Stopping 1/1
  ✓ Container web4   Stopped
[+] Restarting 1/1
  ✓ Container balanceador-nginx Started
```



```

parrot: ~/.../P2/Escalado
• → python3 logica_escalado.py
Media de memoria: 21231616.0
Media de uso de CPU: 75408750.0
Verificando 4 contenedores...
Contenedor web2:
  - Memoria: 21241856
  - Uso de CPU total: 81455000
El contenedor web2 supera el umbral de memoria o uso de CPU. Añadiendo un nuevo contenedor.
[+] Running 1/1
✓ Container web5 Started
[+] Restarting 1/1
✓ Container balanceador-nginx Started

parrot: ~/.../P2/Escalado
→ python3 logica_escalado.py
Verificando 3 contenedores...
Contenedor web1:
  - Memoria: 23154688
  - Uso de CPU total: 88149000
Umbral de memoria por arriba: 23627530.24
Umbral de CPU por arriba: 91516440.0
Umbral de memoria por debajo: 22932602.88
Umbral de CPU por debajo: 88824780.0
El contenedor web1 no supera el umbral de memoria o uso de CPU. Parando el ultimo contenedor.
Ya no se pueden parar más contenedores, minimo 3 contenedores activos

```

Funciona correctamente bajando los contenedores y modificando la configuracion de nginx y todo de manera correcta.

A5.1 Extracción de métricas con cAdvisor

A5.1.1 Propuesta final

Para realizar el escalado he decidido usar la imagen de nginx y exportar las metricas usando cAdvisor. El docker compose me queda como sigue:

```

version: '4.0'

services:
  web1:
    image: mariorgzlpz--apache--image:p2
    volumes:
      - ./web_MarioRgzLpz:/var/www/localhost/htdocs
    container_name: web1
    networks:
      red_web:
        ipv4_address: 192.168.10.2
      red_servicios:
        ipv4_address: 192.168.20.2

  web2:
    image: mariorgzlpz--apache--image:p2
    volumes:
      - ./web_MarioRgzLpz:/var/www/localhost/htdocs
    container_name: web2
    networks:
      red_web:
        ipv4_address: 192.168.10.3
      red_servicios:
        ipv4_address: 192.168.20.3

  web3:
    image: mariorgzlpz--apache--image:p2
    volumes:

```

```

    - ./web_MarioRgzLpz:/var/www/localhost/htdocs
  container_name: web3
  networks:
    red_web:
      ipv4_address: 192.168.10.4
    red_servicios:
      ipv4_address: 192.168.20.4

web4:
  image: mariorgzlpz--apache--image:p2
  volumes:
    - ./web_MarioRgzLpz:/var/www/localhost/htdocs
  container_name: web4
  networks:
    red_web:
      ipv4_address: 192.168.10.5
    red_servicios:
      ipv4_address: 192.168.20.5

web5:
  image: mariorgzlpz--apache--image:p2
  volumes:
    - ./web_MarioRgzLpz:/var/www/localhost/htdocs
  container_name: web5
  networks:
    red_web:
      ipv4_address: 192.168.10.6
    red_servicios:
      ipv4_address: 192.168.20.6

web6:
  image: mariorgzlpz--apache--image:p2
  volumes:
    - ./web_MarioRgzLpz:/var/www/localhost/htdocs
  container_name: web6
  networks:
    red_web:
      ipv4_address: 192.168.10.7
    red_servicios:
      ipv4_address: 192.168.20.7

web7:
  image: mariorgzlpz--apache--image:p2
  volumes:
    - ./web_MarioRgzLpz:/var/www/localhost/htdocs
  container_name: web7
  networks:
    red_web:
      ipv4_address: 192.168.10.8
    red_servicios:
      ipv4_address: 192.168.20.8

web8:
  image: mariorgzlpz--apache--image:p2
  volumes:
    - ./web_MarioRgzLpz:/var/www/localhost/htdocs
  container_name: web8
  networks:
    red_web:
      ipv4_address: 192.168.10.9
    red_servicios:
      ipv4_address: 192.168.20.9

balanceador--nginx:
  image: mariorgzlpz--nginx--image:p2
  volumes:
    - ./nginx.conf:/etc/nginx/nginx.conf
  container_name: balanceador--nginx
  ports:
    - 80:80
  networks:
    red_web:
      ipv4_address: 192.168.10.50

```

```

cadvisor:
  image: gcr.io/cadvisor/cadvisor:latest
  container_name: cadvisor
  ports:
    - "8080:8080"
  volumes:
    - /:/rootfs:ro
    - /var/run:/var/run:rw
    - /sys:/sys:ro
    - /var/lib/docker:/var/lib/docker:ro
  depends_on:
    - balanceador-nginx

networks:
  red_web:
    driver: bridge
    ipam:
      config:
        - subnet: 192.168.10.0/24
          gateway: 192.168.10.1
  red_servicios:
    driver: bridge
    ipam:
      config:
        - subnet: 192.168.20.0/24
          gateway: 192.168.20.1

```

Y la configuracion de nginx inicial es la siguiente:

```

events {
  worker_connections 1024;
}

http {
  upstream backend_MarioRgzLpz {

    server 192.168.10.2 weight=1;
    server 192.168.10.3 weight=1;
    server 192.168.10.4 weight=1;
    server 192.168.10.5 weight=1;
  }

  server {
    listen 80;
    server_name nginx_MarioRgzLpz;
    access_log /var/log/nginx/nginx_MarioRgzLpz.access_log;
    error_log /var/log/nginx/nginx_MarioRgzLpz.error_log;

    location / {
      proxy_pass http://backend_MarioRgzLpz;
      proxy_set_header Cookie $http_cookie;
      proxy_hide_header Set-Cookie;
    }

    location /estadisticas_MarioRgzLpz {
      stub_status on;
    }
  }
}

```

A5.1.2 Análisis propuesta IA

Preguntas a ChatGPT —Tengo el siguiente docker compose (docker compose nginx) y quiero exportar las metricas de los contendores usando cAdvisor

—No me funciona, me sale lo siguiente en los logs:

```
docker logs cadvisor
F0429 09:38:11.766240          1 cadvisor.go:146] Failed to create a Container Manager:
mountpoint for cpu not found
```

Respuestas ChatGPT —Para agregar cAdvisor a tu configuración de Docker Compose para extraer métricas, puedes agregarlo como un nuevo servicio en tu archivo docker-compose.yml. Aquí te muestro cómo puedes modificar tu archivo para incluir cAdvisor:

```
version: '4.0'

services:
  web1:
    image: mariorgzlpz--apache--image:p2
    volumes:
      - ./web_MarioRgzLpz:/var/www/localhost/htdocs
    container_name: web1
    networks:
      red_web:
        ipv4_address: 192.168.10.2
      red_servicios:
        ipv4_address: 192.168.20.2

# Otros servicios web (web2, web3, ..., web8) aqui...

balanceador--nginx:
  image: mariorgzlpz--nginx--image:p2
  volumes:
    - ./nginx.conf:/etc/nginx/nginx.conf
  container_name: balanceador--nginx
  ports:
    - 80:80
  networks:
    red_web:
      ipv4_address: 192.168.10.50
  depends_on:
    - web1
    - web2
    - web3
    - web4
    - web5
    - web6
    - web7
    - web8

cadvisor:
  image: google/cadvisor:latest
  container_name: cadvisor
  ports:
    - "8080:8080"
  volumes:
    - /:/rootfs:ro
    - /var/run:/var/run:rw
    - /sys:/sys:ro
    - /var/lib/docker:/var/lib/docker:ro

networks:
  red_web:
    driver: bridge
    ipam:
      config:
        - subnet: 192.168.10.0/24
          gateway: 192.168.10.1
  red_servicios:
    driver: bridge
    ipam:
      config:
        - subnet: 192.168.20.0/24
```

gateway: 192.168.20.1

—El error que estás viendo indica que cAdvisor está teniendo problemas para encontrar el punto de montaje para las métricas de CPU en tu sistema. Este error puede ocurrir si el sistema en el que estás ejecutando Docker no tiene habilitadas las métricas de CPU.

Para solucionar este problema, necesitas habilitar las métricas de CPU en el sistema host y reiniciar cAdvisor. Aquí hay algunos pasos que puedes seguir para resolver este problema:

Análisis La propuesta de ChatGPT es correcta pero el porque me da error no era correcto. Lo unico que he hecho es mirar la imagen en DockerHub donde aparece que google/cAdvisor esta deprecated y te recomienda usar otra imagen. Usando dicha imagen me funcionaba todo bien y me quedaba como sigue.



A5.2 Scripts para escalar servicios

Mi idea es hacer un script que al ejecutarse pare o arranque contenedores en función del uso de memoria y cpu que estan haciendo los contenedores activos. Además quiero que elimine o añada la IP del contenedor que baja o sube de la configuración de nginx para evitar que se hagan accesos a esas IPs.

A5.2.1 Propuesta final

Siguiendo lo primero que me dice ChatGPT no logro ver nada. Por lo que directamente decido consultar la documentación oficial para no perder tiempo. Aquí veo que para obtener las metricas de docker necesito acceder a la siguiente url: <http://localhost:8080/api/v1.2/docker/>.

Tomando el script que me pasa ChatGPT la primera vez no funciona, esto era debido a que no estaba obteniendo bien los datos, haciendo unas modificaciones a este script llego al siguiente script que si funciona y me proporciona los datos de mis contenedores web almacenados en un archivo con formato json para más tarde aplicar la lógica de escalado. El siguiente archivo python es datos.py:

```
import json
import requests

def obtener_datos_contenedores():
    # Hacer solicitud HTTP a cAdvisor
    url = "http://localhost:8080/api/v1.2/docker"
    response = requests.get(url)

    # Verificar si la solicitud fue exitosa
    if response.status_code == 200:
        data = response.json()
        contenedores_webX = []

        # Filtrar contenedores webX y obtener memoria y uso total de CPU
        for contenedor_id, datos_contenedor in data.items():
            nombre = datos_contenedor.get("aliases", [""])[0]
            if "web" in nombre:
                uso_cpu_total = datos_contenedor.get("stats", []) [0].get("cpu", {}).get("usage", {}).get("total", 0)
                memoria = datos_contenedor.get("stats", []) [0].get("memory", {}).get("usage", 0)

                contenedor = {
                    "nombre_contenedor": nombre,
                    "contenedor_id": datos_contenedor.get("id"),
                    "memoria": memoria,
                    "uso_cpu_total": uso_cpu_total
                }
                contenedores_webX.append(contenedor)

        return contenedores_webX
    else:
        print("Error al obtener datos de cAdvisor:", response.status_code)
        return []

# Obtener datos
contenedores_webX = obtener_datos_contenedores()

# Guardar datos en un archivo JSON
nombre_archivo = "./data/datos.json"
with open(nombre_archivo, "w") as archivo:
    json.dump(contenedores_webX, archivo, indent=4)

print("Datos guardados en:", nombre_archivo)
```

Para el escalado he decidido hacer que dependa de la media del uso de memoria y cpu de los contenedores que hay activos, asi si alguno de los contenedores tiene un uso anormal por encima o por debajo de la media se aplicara el escalado o desescalado de los servicios, parando o lanzando servicios en función del caso. Esto lo hago con el siguiente archivo:

```
import json
import os
import subprocess

def leer_datos_contenedores(nombre_archivo):
    with open(nombre_archivo, "r") as archivo:
        return json.load(archivo)

def calcular_medias(contenedores):
    memoria_total = sum(contenedor["memoria"] for contenedor in contenedores)
    cpu_total = sum(contenedor["uso_cpu_total"] for contenedor in contenedores)
    num_contenedores = len(contenedores)
```

```

memoria_media = memoria_total / num_contenedores
cpu_media = cpu_total / num_contenedores
return memoria_media, cpu_media

def modificar_configuracion_nginx(num_contenedor):
    # Leer configuraci3n actual
    with open("nginx.conf", "r") as archivo:
        lineas = archivo.readlines()

    # Agregar nueva IP al upstream
    # Encontrar la ltima lnea con el patr3n 'server 192.168.10.X'
    ultima_linea_index = -1
    for i, linea in enumerate(reversed(lineas)):
        if "server 192.168.10." in linea:
            ultima_linea_index = len(lineas) - 1 - i
            break

    # Aadir la nueva IP justo despu3s de la ltima lnea encontrada
    nueva_linea = f"server 192.168.10.{num_contenedor} weight=1;\n"
    lineas.insert(ultima_linea_index + 1, nueva_linea)

    # Escribir nueva configuraci3n
    with open("nginx.conf", "w") as archivo:
        archivo.writelines(lineas)

def eliminar_configuracion_nginx():
    # Leer configuraci3n actual
    with open("nginx.conf", "r") as archivo:
        lineas = archivo.readlines()

    # Encontrar la ltima lnea con el patr3n 'server 192.168.10.X'
    ultima_linea_index = -1
    for i, linea in enumerate(reversed(lineas)):
        if "server 192.168.10." in linea:
            ultima_linea_index = len(lineas) - 1 - i
            break

    # Eliminar la ltima lnea encontrada
    if ultima_linea_index != -1:
        del lineas[ultima_linea_index]

    # Escribir nueva configuraci3n
    with open("nginx.conf", "w") as archivo:
        archivo.writelines(lineas)
    else:
        print("No se encontr3 ninguna lnea con la IP 192.168.10.")

def recargar_contenedor_nginx():
    # Recargar el contenedor Nginx utilizando Docker Compose
    subprocess.run(["docker", "compose", "restart", "balanceador-nginx"])

def lanzar_siguiente_contenedor(num_contenedor):
    if(num_contenedor <= 8):
        # Lanzar el siguiente contenedor utilizando Docker Compose
        nombre = f"web{num_contenedor + 1}"
        subprocess.run(["docker", "compose", "up", "-d", nombre])
    else:
        print("Ampliando el nmero de contenedores en la red de balanceo")
        subprocess.run(["bash", "crear_docker-compose.sh", num_contenedor + 1])
        subprocess.run(["docker", "compose", "up", "-d", nombre])

def parar_contenedor(num_contenedor):
    if(num_contenedor > 3):
        # Lanzar el siguiente contenedor utilizando Docker Compose
        nombre = f"web{num_contenedor}"
        subprocess.run(["docker", "compose", "stop", nombre])
    else:
        print("Ya no se pueden parar m3s contenedores, minimo 3 contenedores activos")

# Leer datos de los contenedores desde el archivo JSON
contenedores = leer_datos_contenedores("./data/datos.json")

```

```

# Calcular medias de memoria y uso de CPU
memoria_media, cpu_media = calcular_medias(contenedores)

# Verificar si algun contenedor supera los umbrales
num_contenedores = len(contenedores)
print(f"Verificando {num_contenedores} contenedores...")
for contenedor in contenedores:
    print(f"Contenedor {contenedor['nombre_contenedor']}:")
    print(f"  - Memoria: {contenedor['memoria']}")
    print(f"  - Uso de CPU total: {contenedor['uso_cpu_total']}")
    print("Umbral de memoria por arriba:", memoria_media*1.02)
    print("Umbral de CPU por arriba:", cpu_media*1.02)
    print("Umbral de memoria por debajo:", memoria_media*0.99)
    print("Umbral de CPU por debajo:", cpu_media*0.99)
    if contenedor["memoria"] > memoria_media*1.02 or contenedor["uso_cpu_total"] >
cpu_media*1.02:
        print(f"El contenedor {contenedor['nombre_contenedor']} supera el umbral de
memoria o uso de CPU. A adiando un nuevo contenedor.")
        # Modificar la configuracion de Nginx y recargar
        lanzar_siguiente_contenedor(num_contenedores)
        modificar_configuracion_nginx(num_contenedores + 1)
        recargar_contenedor_nginx()
        # Lanzar el siguiente contenedor

        break # Solo modificamos/recargamos para el primer contenedor que supera el
umbral
    elif contenedor["memoria"] < memoria_media*0.99 or contenedor["uso_cpu_total"] <
cpu_media*0.99:
        print(f"El contenedor {contenedor['nombre_contenedor']} no supera el umbral de
memoria o uso de CPU. Parando el ultimo contenedor.")
        # Detener el contenedor
        parar_contenedor(num_contenedores)
        if (num_contenedores > 3):
            eliminar_configuracion_nginx()
            recargar_contenedor_nginx()
        break # Solo detenemos el primer contenedor que no supera el umbral

```

Lo que hace es primero comprueba contenedor a contenedor si superan o estan por debajo del umbral establecido y si es asi lanzará o parará el siguiente contenedor, modificará el fichero de configuración de nginx y recargará el balanceador.

A5.2.2 Análisis propuesta IA

Como puedo obtener datos a partir de metricas de cAdvisor

Preguntas a ChatGPT

Pregunta 1: Como puedo obtener datos a partir de metricas de cAdvisor

Pregunta 2: Estos son algunos de los datos que obtengo

```

{
  "/system.slice/docker-0dbfdf409b255ee789516ca67d64a70f59cf9
6db550acea03440e62521ad6a33.scope": {
    "id": "0dbfdf409b255ee789516ca67d64a70f59cf96db550acea03440e62521ad6a33",
    "name": "/system.slice/docker-0dbfdf409b255ee789516ca67d64a7
0f59cf96db550acea03440e62521ad6a33.scope",
    "alias": [
      "web4",
      "0dbfdf409b255ee789516ca67d64a70f59cf96db550acea03440e62521ad6a33"
    ],
    "namespace": "docker",
    "spec": {
      "creation_time": "2024-04-29T09:40:05.946151012Z",
      "labels": {
        "com.docker.compose.config-hash":
"0257f0efd5bf2b763d570b2d3bc658a3c1d0429874a40170c5e386cd8a03f237",
        "com.docker.compose.container-number": "1",
        "com.docker.compose.depends_on": "",
        "com.docker.compose.image":

```



```

    "sha256:5080a0199fa573dc34e9834768e51c2064c78a26458e64383b566f9abf19e48e",
    "com.docker.compose.oneoff": "False",
    "com.docker.compose.project": "escalado",
    "com.docker.compose.project.config_files":
    "/home/mariorl/Desktop/DGIIM/SWAP/P2/Escalado/docker-compose.yml",
    "com.docker.compose.project.working_dir":
    "/home/mariorl/Desktop/DGIIM/SWAP/P2/Escalado",
    "com.docker.compose.service": "web4",
    "com.docker.compose.version": "2.24.6"
  },
  "has_cpu": true,
  "cpu": {
    "limit": 0,
    "max_limit": 0,
    "mask": "0-15"
  },
  "has_memory": true,
  "memory": {
    "limit": 16108408832
  },
  "has_hugetlb": true,
  "has_network": true,
  "has_processes": true,
  "processes": {
    "limit": 18325
  },
  "has_filesystem": true,
  "has_diskio": true,
  "has_custom_metrics": false,
  "image": "mariorgzlpz-apache-image:p2"
},
"stats": [
  {
    "timestamp": "2024-04-30T08:29:11.488948218Z",
    "cpu": {
      "usage": {
        "total": 292890000,
        "user": 95245000,
        "system": 197644000
      },
      "cfs": {
        "periods": 0,
        "throttled_periods": 0,
        "throttled_time": 0
      },
      "schedstat": {
        "run_time": 0,
        "runqueue_time": 0,
        "run_periods": 0
      },
      "load_average": 0
    },
    "diskio": {
      "io_service_bytes": [
        {
          "device": "/dev/sda",
          "major": 8,
          "minor": 0,

```

```

        "stats": {
            "dbytes": 0,
            "dios": 0,
            "read": 14012416,
            "rios": 97,
            "wios": 0,
            "write": 0
        }
    }
]
},
"memory": {
    "usage": 23109632,
    "max_usage": 0,
    "cache": 0,
    "rss": 0,
    "swap": 0,
    "mapped_file": 0,
    "working_set": 22179840,
    "failcnt": 0,
    "container_data": {
        "pgfault": 2959,
        "pgmajfault": 27,
        "numa_stats": {
            }
        }
    }
}

```

Quiero un script en python que obtenga de esta url <http://localhost:8080/api/v1.2/docker/> los datos y que me almacene para cada contenedor webX el nombre, el id, la memoria usada y la cpu total usada.

Pregunta 3: Ahora quiero realizar un script python para escalado teniendo en cuenta los datos que guardo en formato json. Para esto quiero comprobar el estado de los contenedores y ver si alguno supera por cierto umbral tanto la media de memoria como la media de uso de cpu de todos los contenedores. Los datos se almacenan como sigue:

```

[
  {
    "nombre_contenedor": "web2",
    "contenedor_id": "4512b8f5ce9633101ce40470c6007c90c41fe68bee0d0ee6ec8f6e02dcef942",
    "memoria": 21950464,
    "uso_cpu_total": 75177000
  },
  {
    "nombre_contenedor": "web4",
    "contenedor_id": "b50e5fee24b84356d805cb9764fc0d78092f997b17a72fc4290815995a995107",
    "memoria": 22237184,
    "uso_cpu_total": 73702000
  },
  {
    "nombre_contenedor": "web1",
    "contenedor_id": "e4053f3cb141f4662585b62c37942025496fea79ba326bfdc573f96d6a7fe996",
    "memoria": 22618112,
    "uso_cpu_total": 78458000
  },
  {
    "nombre_contenedor": "web3",
    "contenedor_id": "e1e6c4275cb4ad6609352f0ad254bbf23a37c802e75cdf28daf70d3daba2e53",
    "memoria": 22233088,
    "uso_cpu_total": 76236000
  }
]

```

```

    }
]

```

Y lo que quiero hacer es modificar el fichero de configuracion de nginx que es el siguiente:

```

events {
    worker_connections 1024;
}

http {
    upstream backend_MarioRgzLpz {

        server 192.168.10.2 weight=1;
        server 192.168.10.3 weight=1;
        server 192.168.10.4 weight=1;
        server 192.168.10.5 weight=1;
    }

    server {
        listen 80;
        server_name nginx_MarioRgzLpz;
        access_log /var/log/nginx/nginx_MarioRgzLpz.access_log;
        error_log /var/log/nginx/nginx_MarioRgzLpz.error_log;

        location / {
            proxy_pass http://backend_MarioRgzLpz;
            proxy_set_header Cookie $http_cookie;
            proxy_hide_header Set-Cookie;
        }

        location /estadisticas_MarioRgzLpz {
            stub_status on;
        }
    }
}

```

y añadir aqui la IP del siguiente contenedor que será igual que la ultima pero sumando uno al final de la IP, ademas una vez modificada la configuracion debo recargar el contenedor con el balanceador y tambien tengo que lanzar el siguiente contenedor.

Pregunta 4: Para recargar el contenedor con el balanceador nginx y lanzar o bajar el nuevo contenedor debo usar docker compose up -d y el nombre del balanceador o del contenedor

Pregunta 5: El siguiente contenedor depende del numero de contenedores actuales

Pregunta 6: Ahora quiero el caso en el que la memoria del contenedor sea menor que el umbral o el uso de la cpu sea menor que el umbral y que en lugar de lanzar un contenedor lo baje

Pregunta 7: Quiero un script que realice 100 acceso a la direccion 192.168.10.2

Respuestas ChatGPT

Respuesta 1: Para obtener datos a partir de métricas de cAdvisor (Container Advisor), que es una herramienta que proporciona información sobre el rendimiento de los contenedores en sistemas Docker, puedes seguir estos pasos:

Accede a cAdvisor: Primero, necesitas acceder a la interfaz de cAdvisor. Puedes hacerlo a través de la interfaz web de cAdvisor, que por lo general está disponible en `http://IP_del_host:8080`.

Explora las métricas disponibles: Una vez dentro de la interfaz de cAdvisor, podrás explorar las diferentes métricas disponibles para los contenedores que estás monitoreando. cAdvisor ofrece una amplia variedad de métricas, que incluyen el uso de CPU, memoria, red, disco, entre otras.

Utiliza la API de cAdvisor: Además de la interfaz web, cAdvisor también proporciona una API HTTP que puedes utilizar para obtener datos programáticamente. Puedes hacer solicitudes HTTP a la API de cAdvisor para obtener métricas específicas en formatos como JSON o protobuf.

Consulta la documentación de la API: Para comprender completamente cómo utilizar la API de cAdvisor, te recomendaría consultar la documentación oficial de cAdvisor, donde encontrarás información detallada sobre los endpoints disponibles y cómo utilizarlos.

Implementa la integración en tu sistema: Una vez que estés familiarizado con las métricas disponibles y cómo acceder a ellas a través de la API de cAdvisor, puedes integrar la obtención de datos en tu sistema o aplicación. Esto podría implicar escribir scripts o desarrollar una integración más profunda dependiendo de tus necesidades específica

Respuesta 2: Aquí tienes un script python

```
import requests

def obtener_datos_contenedores_webX():
    # Hacer solicitud HTTP a cAdvisor
    url = "http://localhost:8080/api/v1.2/docker"
    response = requests.get(url)

    # Verificar si la solicitud fue exitosa
    if response.status_code == 200:
        data = response.json()
        contenedores_webX = []

        # Filtrar contenedores webX y obtener memoria y uso total de CPU
        for contenedor_id, datos_contenedor in data.items():
            if "webX" in datos_contenedor.get("alias", ""):
                memoria = datos_contenedor.get("memory", {}).get("usage", 0)
                uso_cpu_total = datos_contenedor.get("cpu", {}).get("usage", {}).get("total", 0)

                contenedor = {
                    "contenedor_id": contenedor_id,
                    "memoria": memoria,
                    "uso_cpu_total": uso_cpu_total
                }
                contenedores_webX.append(contenedor)

        return contenedores_webX
    else:
        print("Error al obtener datos de cAdvisor:", response.status_code)
        return []

# Ejemplo de uso
contenedores_webX = obtener_datos_contenedores_webX()
for contenedor in contenedores_webX:
    print("ID del contenedor:", contenedor["contenedor_id"])
    print("Memoria:", contenedor["memoria"])
    print("Uso total de CPU:", contenedor["uso_cpu_total"])
    print()
```

Respuesta 2: Entendido. Para alcanzar este objetivo, necesitamos los siguientes pasos:

Leer los datos de los contenedores desde el archivo JSON.

Calcular la media de la memoria y el uso total de CPU de todos los contenedores.

Verificar si algún contenedor supera ciertos umbrales de media para la memoria y el uso de CPU.

Si algún contenedor supera los umbrales, modificar la configuración de Nginx para agregar una nueva dirección IP de contenedor al upstream. Recargar la configuración de Nginx para aplicar los cambios.

Lanzar el siguiente contenedor.

Aquí tienes un script de Python que realiza estas acciones:

```
import json
import os
import subprocess

def leer_datos_contenedores(nombre_archivo):
    with open(nombre_archivo, "r") as archivo:
        return json.load(archivo)

def calcular_medias(contenedores):
    memoria_total = sum(contenedor["memoria"] for contenedor in contenedores)
    cpu_total = sum(contenedor["uso_cpu_total"] for contenedor in contenedores)
    num_contenedores = len(contenedores)
    memoria_media = memoria_total / num_contenedores
    cpu_media = cpu_total / num_contenedores
```

```

    return memoria_media, cpu_media

def modificar_configuracion_nginx(ip):
    # Leer configuracion actual
    with open("nginx.conf", "r") as archivo:
        configuracion = archivo.read()

    # Agregar nueva IP al upstream
    nueva_linea = f"server {ip} weight=1;\n"
    nueva_configuracion = configuracion.replace("}", f"{nueva_linea}    }}")

    # Escribir nueva configuracion
    with open("nginx.conf", "w") as archivo:
        archivo.write(nueva_configuracion)

def recargar_nginx():
    # Recargar Nginx
    subprocess.run(["nginx", "-s", "reload"])

def lanzar_siguiente_contenedor():
    # Comando para lanzar el siguiente contenedor (reemplaza con el comando real)
    # Ejemplo: subprocess.run(["docker", "run", "-d", "nombre_imagen"])
    pass

# Leer datos de los contenedores desde el archivo JSON
contenedores = leer_datos_contenedores("datos_contenedores.json")

# Calcular medias de memoria y uso de CPU
memoria_media, cpu_media = calcular_medias(contenedores)
print("Media de memoria:", memoria_media)
print("Media de uso de CPU:", cpu_media)

# Umbrales para la memoria y el uso de CPU
umbral_memoria = 25000000 # ejemplo de umbral de memoria
umbral_cpu = 80000000 # ejemplo de umbral de uso de CPU

# Verificar si algun contenedor supera los umbrales
for contenedor in contenedores:
    if contenedor["memoria"] > umbral_memoria or contenedor["uso_cpu_total"] > umbral_cpu:
        print(f"El contenedor {contenedor['nombre_contenedor']} supera el umbral de memoria o uso de CPU.")
        # Modificar la configuracion de Nginx y recargar
        nueva_ip = "192.168.10.6" # ejemplo de nueva IP para el siguiente contenedor
        modificar_configuracion_nginx(nueva_ip)
        recargar_nginx()
        # Lanzar el siguiente contenedor
        lanzar_siguiente_contenedor()
        break # Solo modificamos/recargamos para el primer contenedor que supera el umbral

```

Respuesta 4: Debes usar esto:

```
subprocess.run(["docker", "compose", "up", "-d"])
```

Respuesta 5:

```
nombre_siguiente_contenedor = f"web{num_contenedores + 1}"
```

Respuesta 6:

```

elif contenedor["memoria"] < umbral_memoria or contenedor["uso_cpu_total"] <
umbral_cpu:
    print(f"El contenedor {contenedor['nombre_contenedor']} no supera el umbral de memoria o uso de CPU. Deteniendo contenedor.")
    # Detener el contenedor actual
    detener_contenedor(contenedor["nombre_contenedor"])
    break # Solo detenemos el primer contenedor que no supera el umbral

```

Respuesta 7:

```
import requests
```

```
def realizar_solicitudes():
```

```

url = "http://192.168.10.2"
for i in range(100):
    response = requests.get(url)
    if response.status_code == 200:
        print(f"Solicitud {i+1}: La solicitud fue exitosa")
    else:
        print(f"Solicitud {i+1}: Error al realizar la solicitud. Código de estado: {
            response.status_code}")

# Llamar a la función para realizar las solicitudes
realizar_solicitudes()

```

Análisis

Siguiendo lo que me dice ChatGPT veo en la documentación de la API lo que yo busco y a partir de aquí le hago consultas para hacer los scripts que me hacen falta los cuales no me funcionaban de primeras pero realizando algunas pruebas y cambios logro hacer que se almacenen las métricas de los contenedores web activos.

Obteniendo de manera correcta las métricas ya simplemente le pregunto acerca de cómo realizar el script para el escalado el cual de primeras no es perfecto pero haciendo varias preguntas más y modificando algunas cosas logro que funcione.

A5.3 Automatización del script

A5.3.1 Propuesta final

He seguido lo que me ha dicho ChatGPT. Aquí muestro como queda:

```

GNU nano 7.2 /tmp/crontab.w9F73d/crontab
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m. every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
*/2 * * * /usr/bin/python3 /home/marior1/Desktop/DGIIM/SWAP/P2/Escalado/datos.py
*/2 * * * /usr/bin/python3 /home/marior1/Desktop/DGIIM/SWAP/P2/Escalado/escalado.py
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow command
line 22/25 (88%), col 70/70 (100%), char 1023/1056 (96%)
^H Help ^G Read File ^R Replace ^V Paste ^C Go To Line ^Y Redo ^M-6 Copy
^X Exit ^E Where Is ^K Cut ^J Execute ^U Undo ^M-A Set Mark ^M-] To Bracket

```

A5.3.2 Análisis propuesta IA

Preguntas ChatGPT

Tengo dos scripts de python llamados datos.py y escalado.py, quiero crear un crontab que ejecute primero datos.py y luego escalado.py de manera automática cada dos minutos

Respuestas ChatGPT Para configurar un cronjob que ejecute los scripts datos.py y escalado.py cada dos minutos, primero necesitas asegurarte de que los scripts sean ejecutables y estén en una ubicación accesible.

Asumiendo que los scripts están ubicados en el mismo directorio y son ejecutables, aquí está cómo puedes configurar el cronjob:

Abre una terminal.

Ejecuta el comando `crontab -e` para abrir el archivo crontab en el editor predeterminado.

Agrega las siguientes líneas al final del archivo:

```

*/2 * * * * /usr/bin/python3 /ruta/al/datos.py
*/2 * * * * /usr/bin/python3 /ruta/al/escalado.py

```