PRÁCTICA 1

Servidores web y almacenamiento

UNIVERSIDAD DE GRANADA

Titulación: Ingeniería Informática + ADE

FLORIN EMANUEL TODOR GLIGA

ÍNDICE

Tareas Básicas

B1. Configuración del Entorno:

Para ello se descarga docker, en mi caso desde ubuntu, sudo apt install docker, y también he descargado el docker desktop. Por otro lado, he añadido mi usuario al grupo de Docker, para que tenga los permisos necesarios para poder ejecutar docker sin necesidad de permisos de superusuario.

Por otro lado, se crea el directorio de web_florintodor y el archivo index.php dentro de dicho directorio, el cual contiene la información que tiene el readme de mi perfil de github.

B2. Creación del dockerfile

Para la parte básica he creado un dockerfile, para el cual he hecho de la imagen de la última versión de ubuntu, posteriormente he actualizado el sistema y he instalado las herramientas necesarias para poder realizar las pruebas necesarias entre los distintos servidores web.

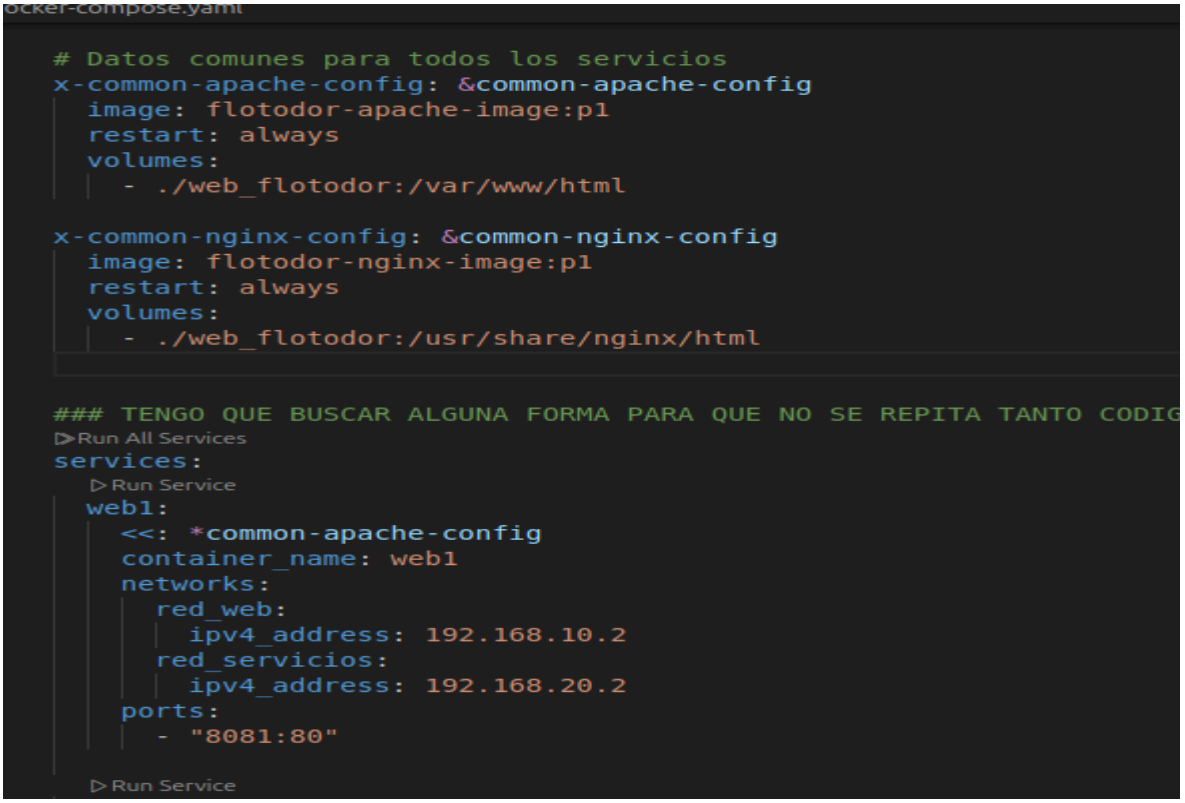
Por otro lado, he expuesto el puerto 80 y he ejecutado el servidor apache en segundo plano.

```
FROM ubuntu:latest

# Actualizamos los paquetes del sistema

RUN  apt-get update && apt-get upgrade -y

# Instalamos apache2 y php

RUN apt-get install apache2 php libapache2-mod-php -y

# Instalamos las herramientas de internet básicas (net-tools)

RUN  apt install net-tools -y
RUN  apt install iputils-ping -y

# Cambiamos la configuración de apache para poner mayor seguridad
COPY ./apache_config/.htaccess /var/www/html/.htaccess

### EJecución de servicios de la imagen

# Ejecutamos el servicio de apache2
CMD ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]

# Exponemos el puerto 80 (http)
EXPOSE 80
```

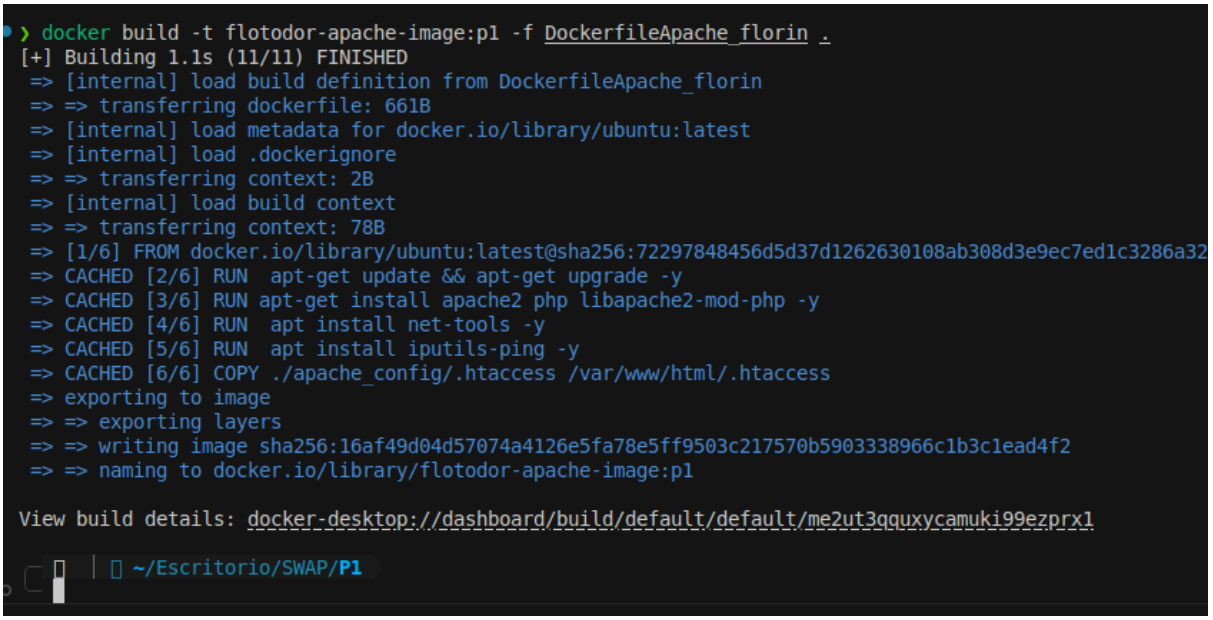
B3.Uso de Docker Compose:

Para ello, aunque posteriormente lo comente con más detalle en la parte avanzada, he creado configuraciones comunes por una parte para servidores apache y por otra parte para los servidores nginx.

Quitando esa parte, el resto de información es tal cual la que se nos solicita en el documento de la práctica 1. Es decir, declaración de los puertos, redes, etc.



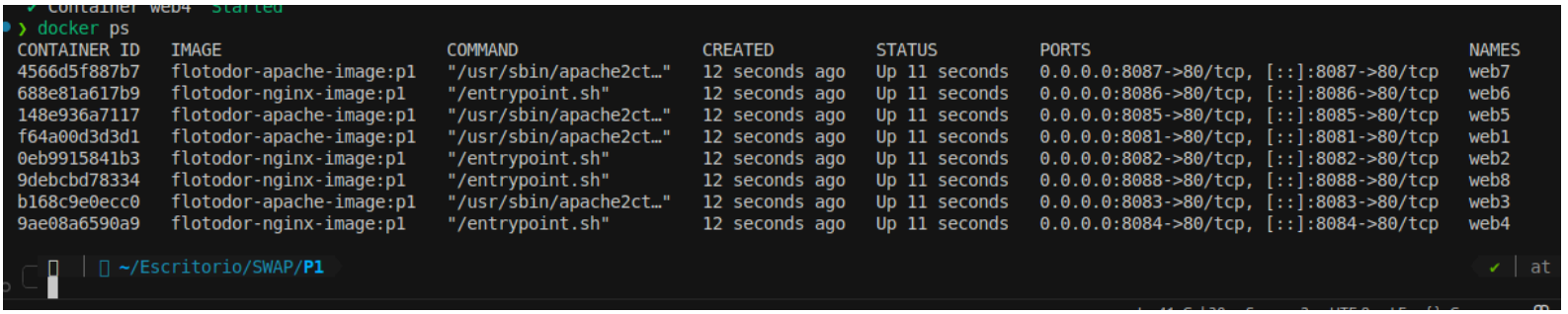
Creación de la imagen de apache



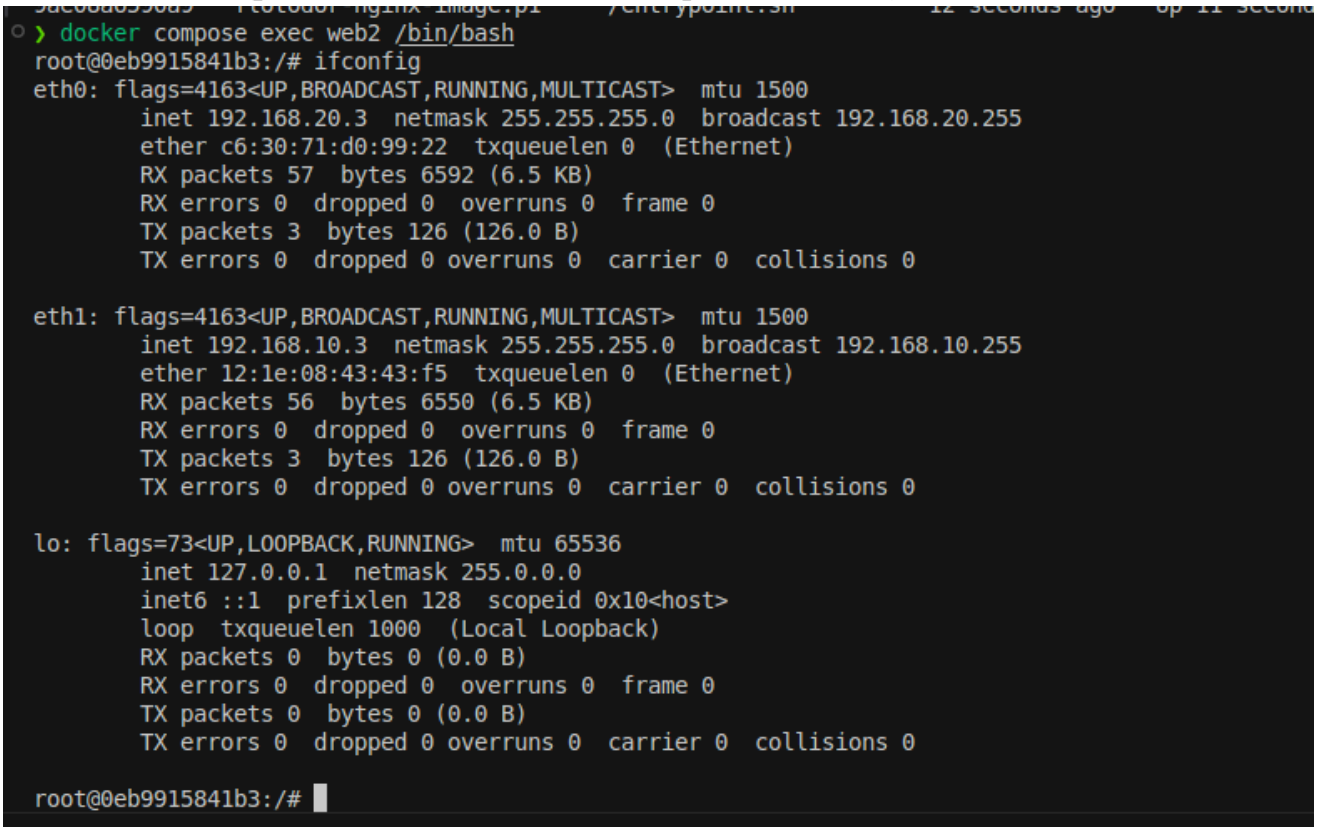
B4. Despliegue y verificación de Contenedores:

Para ello primero ejecuto el docker-compose creado y explicado anteriormente

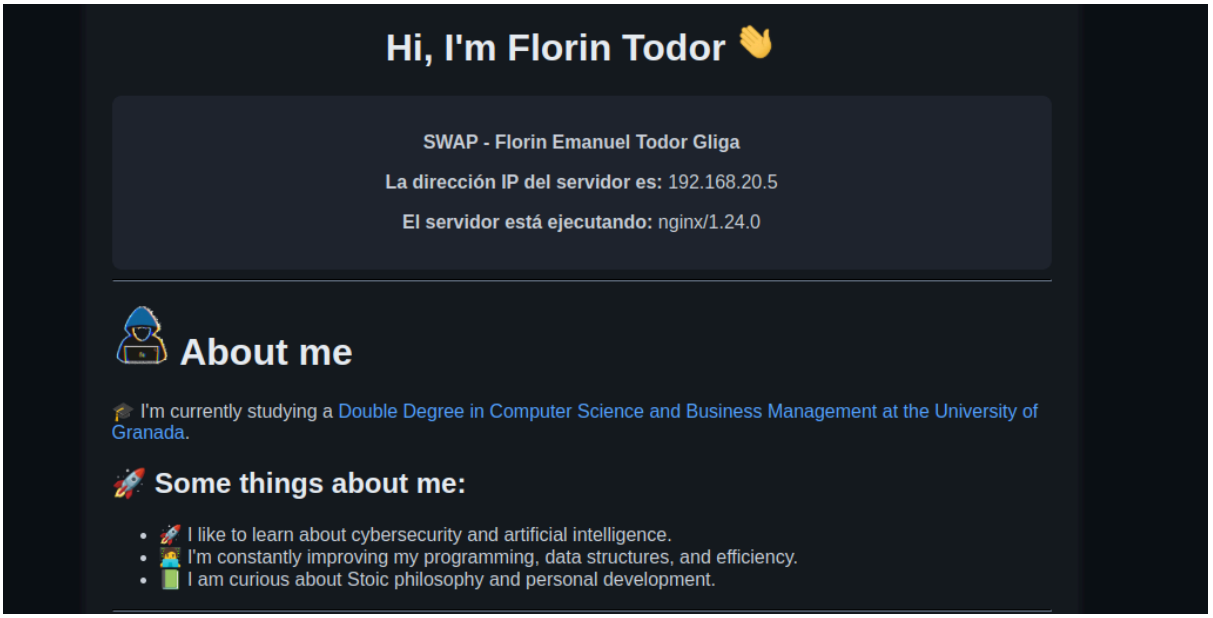
Posteriormente, muestro los contenedores creado donde podemos comprobar que los servidores impares son apache y los pares nginx, es decir, respecto al puerto 808x, siendo x un número par o impar.



Por último, entro en el contenedor de web2 y compruebo con las herramientas previamente descargadas, usando ifconfig, las redes de cada uno de los contenedores (pero solamente le muestro aquí la de web2)



Por último, accedo a la web web4, la cual tiene ip 192.168.20.5, para acceder a ella hacemos uso de localhost:8084.
En ella podemos ver la información solicitada y el tipo de servidor que se está ejecutando, es este caso es nginx.



Tareas Avanzadas

A1. Personalización del Dockerfile:

Respecto a esta tarea avanzada he decidido, junto a la ayuda de la IA, agregar configuraciones de seguridad para proteger así los servidores apache.

```
# Proteger archivos sensibles
<FilesMatch "(^\.ht|\.ini|\.log|\.sh|\.sql|\.bak|\.inc)">
    Order Allow,Deny
    Deny from all
</FilesMatch>

# Bloquear exploración de directorios
Options -Indexes

# Activar compresión gzip para reducir el tamaño de los archivos que se envian y acelerar la página
<IfModule mod_deflate.c>
    AddOutputFilterByType DEFLATE text/html text/plain text/xml text/css text/javascript application/javascript application/json
</IfModule>
```

Para ello, bloqueamos el acceso a archivos acabados en .ht, .log, etc. Por otro lado, bloqueamos la posibilidad de explorar los distintos directorios que forman parte de los archivos del servidor. Por último, activamos la compresión de gzip para así reducir el tamaño de los archivos que se envían por la red, para así acelerar la página.

Para este archivo (.htaccess) realizo una copia de dicho fichero en el dockerfile.

```
15
16    # Cambiamos la configuración de apache para poner mayor seguridad
17    COPY ./apache_config/.htaccess /var/www/html/.htaccess
18
```

A2. Creación de contendores con otros servidores web

Para esta tarea avanzada, he creado un nuevo dockerfile y una nueva imagen, para así crear contenedores nginx.

~/Escritorio/SWAP/P1/DockerfileNginx_florin

```
# Actualizamos los paquetes del sistema
RUN apt-get update && apt-get upgrade -y

# Instalamos Nginx, PHP y PHP-FPMS
RUN apt-get install nginx php php-fpm php-cli php-mysql php-mbstring php-xml php-curl php-gd php-zip php-bcmath -y

# Instalamos herramientas de red básicas
RUN apt install -y net-tools iputils-ping

# Eliminamos la configuración predeterminada de Nginx
RUN rm /etc/nginx/sites-enabled/default

# Copiamos nuestra nueva configuración personalizada
COPY ./Nginx_config/default /etc/nginx/sites-available/default
RUN ln -s /etc/nginx/sites-available/default /etc/nginx/sites-enabled/default

# Copiamos nuestro script de entrada
COPY ./entrypoint.sh /entrypoint.sh

# Expón el puerto 80
EXPOSE 80

# Ejecuta el script de entrada
CMD ["/entrypoint.sh"]
```

Como podemos ver, en este caso he tenido que cambiar varias partes en comparación con los servidores apache.

- Instalar php-fpm para poder ejecutar archivos php en el servidor nginx
- He cambiado la configuración del servidor, para primero colocar el index.php en /usr/share/nginx/html. Posteriormente he tenido que cambiar el archivo default que contiene la configuración del servidor para tener en cuenta el uso de php-fpm y la modificación que he comentado.

Nginx_config > default

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    root /usr/share/nginx/html;
    index index.php index.html;

    server_name _;

    location / {
        try_files $uri $uri/ =404;
    }

    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/run/php/php8.3-fpm.sock;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        include fastcgi_params;
    }

    location ~ /\. {
        deny all;
    }
}
```

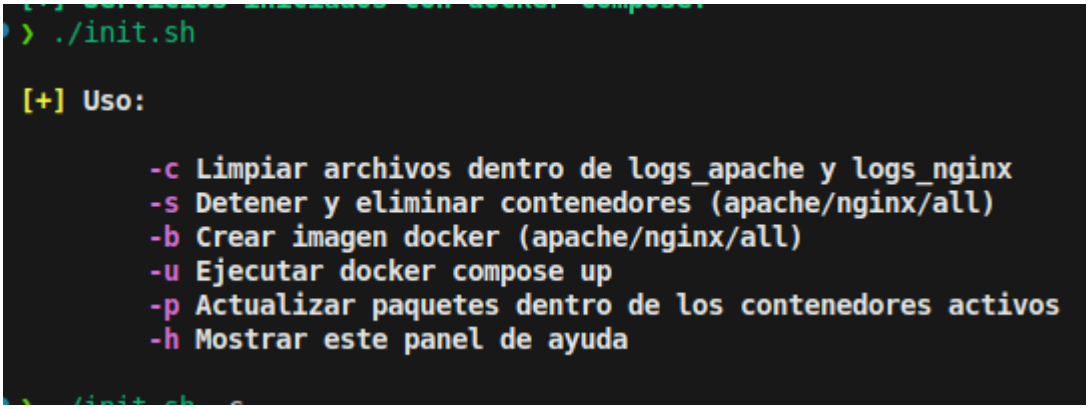
Por último, he creado un script para ejecutarlo con CMD, en dicho script ejecutamos tanto php-fpm como nginx en segundo plano. Esto se debe a que CMD no permite ejecutar ambos “comandos” a la vez.

Este script lo voy a usar también para la parte Monitoreo y Logging (aunque en esta captura no se muestra la información relevante para esas partes).

A4. Automatización con Scripts:

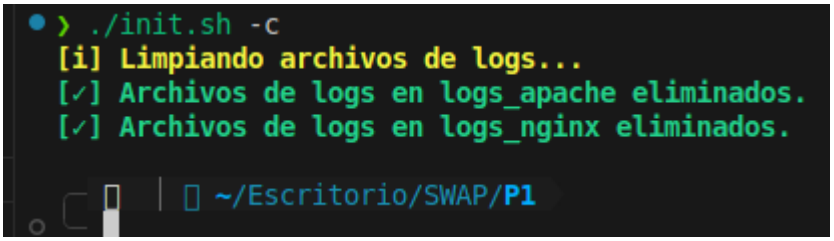
Para esta parte he desarrollado un script en Bash que me permite gestionar de forma centralizada tanto la creación, mantenimiento y automatización de los contenedores.

El script es “init.sh”



Como podemos ver en el menú de ayuda, he implementado 5 funciones.

La **primera función**, con parámetro **-c**, consiste en eliminar los logs de los monitoreos tanto de los contenedores apache como nginx.



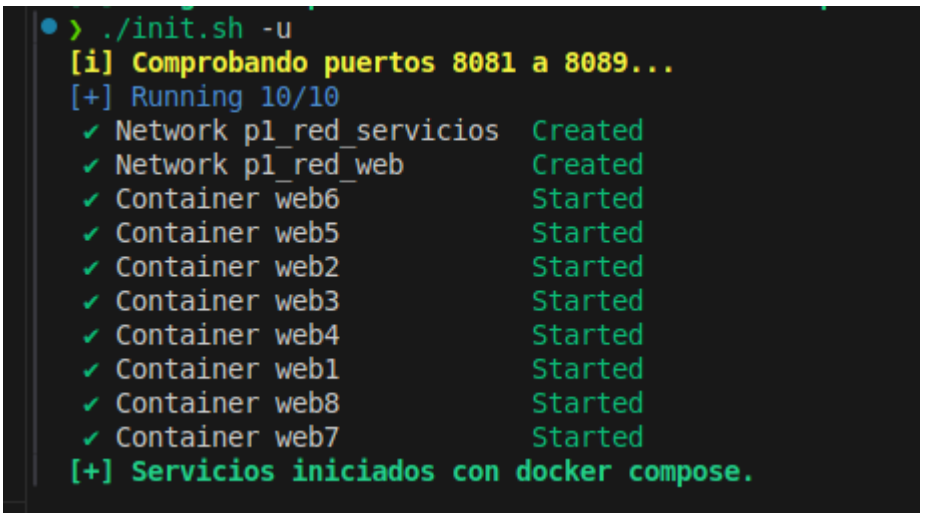
La **segunda función**, con parámetro **-s**, consiste en detener y eliminar los contenedores creados previamente, el usuario puede indicar si solo hacerlo con los servidores apache, nginx o ambos a la vez. Además, elimina las redes creadas (para evitar problemas futuros, como el que ya me ha ocurrido en el que se mostraba que estaban las ips y puertos ocupados por contenedores, pero no había ni un solo contenedor creado y mucho menos ejecutándose).

```
❯ ./init.sh -s all
[+] Contenedor web1 detenido y eliminado
[+] Contenedor web2 detenido y eliminado
[+] Contenedor web3 detenido y eliminado
[+] Contenedor web4 detenido y eliminado
[+] Contenedor web5 detenido y eliminado
[+] Contenedor web6 detenido y eliminado
[+] Contenedor web7 detenido y eliminado
[+] Contenedor web8 detenido y eliminado
p1_red_web
p1_red_servicios
[+] Redes Docker eliminadas.

 ~/Escritorio/SWAP/P1
```

La **tercera función**, con parámetro **-b**, consiste en eliminar las imágenes ya creadas, ya sea de nginx, apache o ambas a la vez. Posteriormente, las creamos de nuevo, esta funcionalidad la realizo para cuando hago modificaciones en los dockerfiles, para poder automatizar el proceso de borrado y creación.

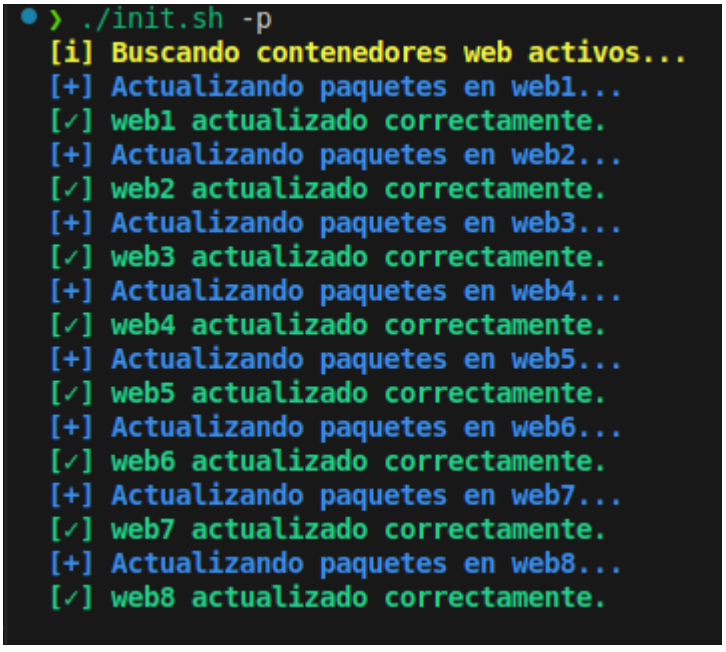
```
❯ ./init.sh -b apache
Untagged: flotodor-apache-image:p1
[+] Building 1.0s (11/11) FINISHED
 => [internal] load build definition from DockerfileApache_florin
 => => transferring dockerfile: 661B
 => [internal] load metadata for docker.io/library/ubuntu:latest
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load build context
 => => transferring context: 557B
 => [1/6] FROM docker.io/library/ubuntu:latest@sha256:72297848456d5d37d1262630108ab308d3e9ec7ed1c3286a32fe09
 => CACHED [2/6] RUN  apt-get update && apt-get upgrade -y
 => CACHED [3/6] RUN apt-get install apache2 php libapache2-mod-php -y
 => CACHED [4/6] RUN  apt install net-tools -y
 => CACHED [5/6] RUN  apt install iputils-ping -y
 => CACHED [6/6] COPY ./apache_config/.htaccess /var/www/html/.htaccess
 => exporting to image
 => => exporting layers
 => => writing image sha256:16af49d04d57074a4126e5fa78e5ff9503c217570b5903338966c1b3c1ead4f2
 => => naming to docker.io/library/flotodor-apache-image:p1

View build details: docker-desktop://dashboard/build/default/default/u3op5cft6vau0o76m26rb65in
[+] Imagen de Apache construida como flotodor-apache-image:p1
```

La **cuarta funcionalidad**, con parámetro **-u**, consiste únicamente en ejecutar el docker-compose.yaml. Para ello previamente comprueba si están los puertos que se van a utilizar para los servidores ocupados, en el caso de que lo estén, se cancela el compose up

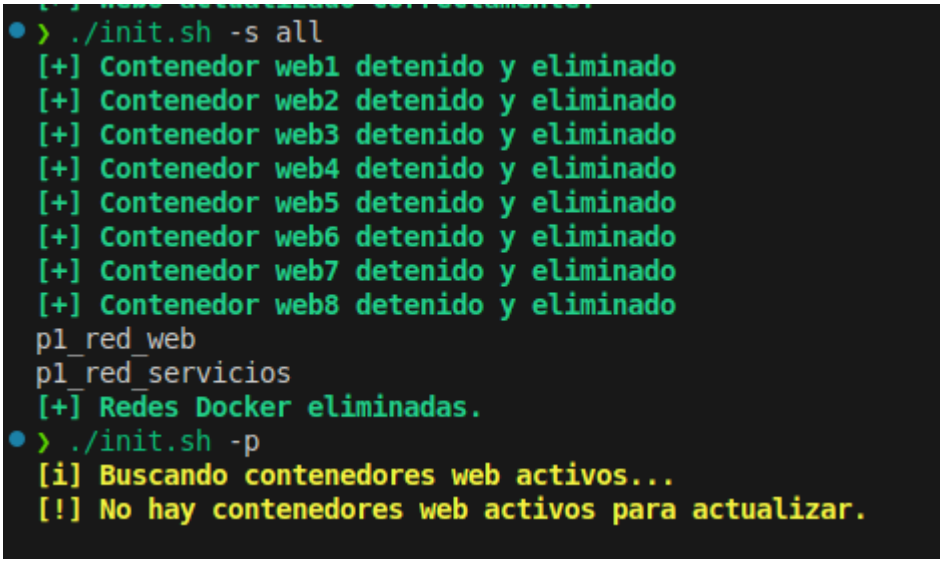


.

Por último, **la quinta funcionalidad**, con parámetro -**p**, consiste en acceder a todos los servidores web que estén activos y actualizar sus paquetes.



Otro ejemplo de ejecución:



A5. Monitoreo y Logging:

Para la parte de monitoreo he creado para los servidores apache un script en bash para monitorear:

- Procesos activos en apache
- Conexiones activas, haciendo uso de netstat
- Mostrar los cinco procesos que más están consumiendo los recursos del servidor

```bash
#!/bin/bash

CONTAINER_NAME="${SERVER_NAME:-apache_container}"
LOGFILE="/var/log/apache2/apache_monitor_${CONTAINER_NAME}.log"

while true; do
  echo "[+] Monitoreando Apache en $CONTAINER_NAME - $(date)" >> $LOGFILE
  echo "[+] Estado de Apache - $(date)" >> $LOGFILE

  echo "[*] Procesos Apache:" >> $LOGFILE
  ps aux | grep apache2 | grep -v grep >> $LOGFILE

  echo "[*] Conexiones activas (netstat):" >> $LOGFILE
  netstat -tuln >> $LOGFILE

  echo "[*] Top procesos por uso de memoria:" >> $LOGFILE
  ps aux --sort=-%mem | head -n 5 >> $LOGFILE

  echo "----------------------------" >> $LOGFILE
  sleep 30
done
```

Además del script, he creado un nuevo volumen para almacenar los logs con la información recogida del monitoreo.

```yaml
# Datos comunes para todos los servicios
x-common-apache-config: &common-apache-config
  image: flotodor-apache-image:p1
  restart: always
  volumes:
    - ./web_flotodor:/var/www/html
    - ./logs_apache:/var/log/apache2



x-common-nginx-config: &common-nginx-config
  image: flotodor-nginx-image:p1
  restart: always
  volumes:
    - ./web_flotodor:/usr/share/nginx/html
    - ./logs_nginx:/var/log/nginx
```

Dicho script se ejecuta cada treinta segundos, no sobrescribe la información que se encontraba anteriormente, sino que escribe dentro del fichero (He decidido dejarlo así por si ocurre algún error tener el registro en todo momento de que ha pasado).

Ejemplo de salida en los logs.

```
[+] Monitoreando Apache en web1 - Tue Mar 25 20:10:45 UTC 2025
[+] Estado de Apache - Tue Mar 25 20:10:45 UTC 2025
[*] Procesos Apache:
root          1  0.0  0.0   2800  1792 ?        Ss   20:02   0:00 /bin/sh /usr/sbin/apache2ctl -D FOREGROUND
root         14  0.0  0.1 203456 21452 ?        S    20:02   0:00 /usr/sbin/apache2 -D FOREGROUND
www-data     19  0.0  0.0 203708 14360 ?        S    20:02   0:00 /usr/sbin/apache2 -D FOREGROUND
www-data     20  0.0  0.0 203708 13856 ?        S    20:02   0:00 /usr/sbin/apache2 -D FOREGROUND
www-data     21  0.0  0.0 203488  7540 ?        S    20:02   0:00 /usr/sbin/apache2 -D FOREGROUND
www-data     22  0.0  0.0 203488  7464 ?        S    20:02   0:00 /usr/sbin/apache2 -D FOREGROUND
www-data     23  0.0  0.0 203488  7464 ?        S    20:02   0:00 /usr/sbin/apache2 -D FOREGROUND
www-data     24  0.0  0.0 203488  7412 ?        S    20:02   0:00 /usr/sbin/apache2 -D FOREGROUND
[*] Conexiones activas (netstat):
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.11:35615        0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
udp        0      0 127.0.0.11:60638        0.0.0.0:*
[*] Top procesos por uso de memoria:
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         14  0.0  0.1 203456 21452 ?        S    20:02   0:00 /usr/sbin/apache2 -D FOREGROUND
www-data     19  0.0  0.0 203708 14360 ?        S    20:02   0:00 /usr/sbin/apache2 -D FOREGROUND
www-data     20  0.0  0.0 203708 13856 ?        S    20:02   0:00 /usr/sbin/apache2 -D FOREGROUND
www-data     21  0.0  0.0 203488  7540 ?        S    20:02   0:00 /usr/sbin/apache2 -D FOREGROUND
```

Por otro lado, también he modificado el monitoreo para los servidores nginx, en este caso únicamente he modificado formato de los logs de acceso, es decir, en el momento que el servidor recibe una petición almacena una información de dicha solicitud y la muestra en el log con el siguiente formato.

```
log_format detailed_logs '[CLIENT $remote_addr] - $remote_user [$time_local] '
                '"$request" => $status ($body_bytes_sent bytes) '
                'Referer: "$http_referer" '
                'Agent: "$http_user_agent" '
                'RequestTime: $request_time sec';
```

Para ello, modifico el fichero nginx.conf de los contenedores nginx.

Ejemplo de salida de logs de nginx.

```
inx > access.log
[CLIENT 192.168.20.1] - - [25/Mar/2025:20:14:12 +0000] "GET / HTTP/1.1" => 200 (2638 bytes) Referer: "-" Agent: "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
[CLIENT 192.168.20.1] - - [25/Mar/2025:20:14:12 +0000] "GET / HTTP/1.1" => 200 (2638 bytes) Referer: "-" Agent: "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
[CLIENT 192.168.20.1] - - [25/Mar/2025:20:14:12 +0000] "GET / HTTP/1.1" => 200 (2638 bytes) Referer: "-" Agent: "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
[CLIENT 192.168.20.1] - - [25/Mar/2025:20:14:13 +0000] "GET / HTTP/1.1" => 200 (2638 bytes) Referer: "-" Agent: "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
[CLIENT 192.168.20.1] - - [25/Mar/2025:20:14:13 +0000] "GET / HTTP/1.1" => 200 (2638 bytes) Referer: "-" Agent: "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
```

Análisis propuesta IA:

ENLACE AL CHAT: https://chatgpt.com/share/67daeaf8-1c68-800d-982f-a20bcf95aa55

Para esta parte tengo que recalcar que he usado bastante la IA debido a que es una buena herramienta para facilitar y agilizar los procesos más básicos y para también agilizar la búsqueda de soluciones a errores comunes.

Sin embargo, aunque sea partidario de utilizarlo, da lugar a varios errores respecto a las implementaciones. Pero por eso tenemos que usarlo como herramienta.

En el principal problema que tuve con la IA fue el intento de crear los contenedores de cada servidor web haciendo uso de algún tipo de loop, sin embargo, aunque la IA me propusiera alguna solución , no funcionaban.
La IA con la constancia respecto a las preguntas me indicó que no se podía, sin embargo, recordé que en la asignatura usábamos archivos .yaml y que si se podían crear loops, pero al final me puse a investigar y no se permite los loops en docker en comparación con ansible debido a la falta de Ninja2 templates en docker (además de no tratarse de un lenguaje de automatización).
Al final, tuve que crear los contenedores de forma manual (podría haber creado un script en bash para crearlos directamente, pero ya tenía creado en sí el archivo y no vamos a usar más servidores webs).
Esta era la versión que me proponía :

yaml Copiar Editar

```yaml
version: '3.8'

services:
  {% for i in range(1, 9) %}
  web{{ i }}:
    image: flotodor-apache-image:p1
    container_name: web{{ i }}
    restart: always
    volumes:
      - .:/var/www/html
    networks:
      red_web:
        ipv4_address: 192.168.10.{{ i }}
      red_servicios:
        ipv4_address: 192.168.20.{{ i }}
    ports:
      - "808{{ i }}:80"
  {% endfor %}

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
```

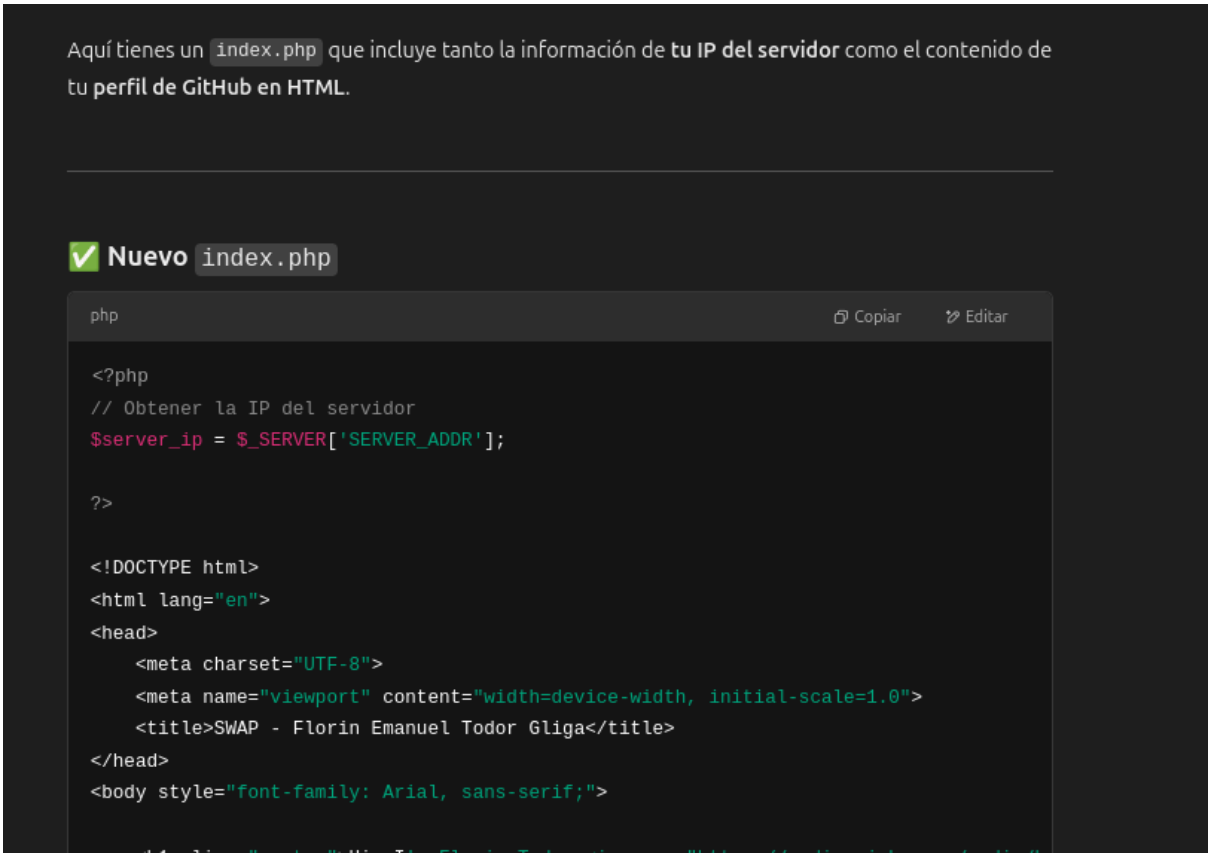
Por otro lado, respecto al mismo tema de antes, me ayudó a crear configuraciones comunes, esto es porque en el docker-compose.yaml al crear los servidores apache/nginx comparten los mismos volumenes e imágenes creadas a través de los dockerfiles.

Por ello, aunque no pude automatizar la creación de los 8 servidores, pude ahorrar algo de código.

```
1
2   # Datos comunes para todos los servicios
3   x-common-apache-config: &common-apache-config
4     image: flotodor-apache-image:p1
5     restart: always
6     volumes:
7       - ./web_flotodor:/var/www/html
8       - ./logs_apache:/var/log/apache2
9
0
1
2
3   x-common-nginx-config: &common-nginx-config
4     image: flotodor-nginx-image:p1
5     restart: always
6     volumes:
7       - ./web_flotodor:/usr/share/nginx/html
8       - ./logs_nginx:/var/log/nginx
```

También me propuso usar docker warms (para crear réplicas,) y docker,sin embargo decidí no usarlo.

Por otro lado, como he comentado anteriormente, el uso de la IA para tareas simples y básicas es de gran utilidad para agilizar procesos, por ejemplo, le pedí que pasara el readme de mi github al index.php de las páginas web y lo hizo en pocos segundos en comparación de haberlo hecho de forma manual (teniendo en cuenta la sintaxis de php).



Aquí tienes un `index.php` que incluye tanto la información de **tu IP del servidor** como el contenido de tu **perfil de GitHub en HTML**.

✅ **Nuevo** `index.php`

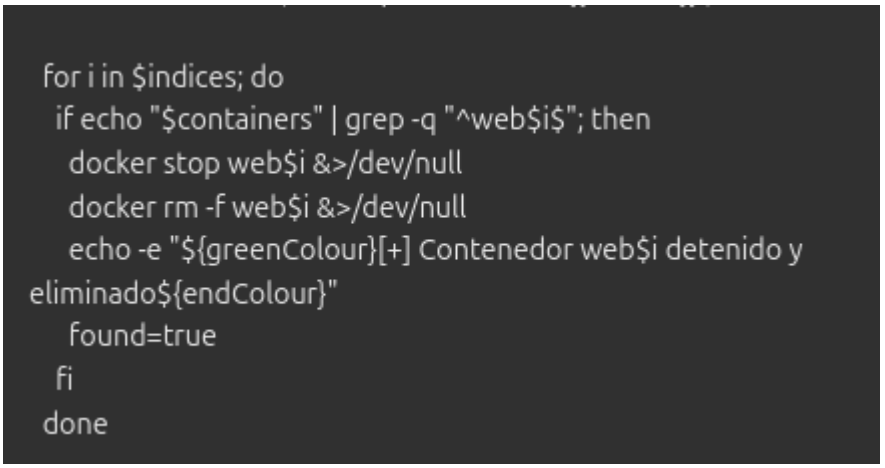
php — Copiar — Editar

```php
<?php
// Obtener la IP del servidor
$server_ip = $_SERVER['SERVER_ADDR'];

?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>SWAP - Florin Emanuel Todor Gliga</title>
</head>
<body style="font-family: Arial, sans-serif;">
```

Por otro lado, me ayudó con la creación de los servidores nginx, debido a que no sabía que era necesario el uso de php-fpm y otras librerías que tuve que instalar.

También tuve que corregirle algunos errores que tuvo, como usar funciones que no sirven para la ejecución de los servidores nginx, etc. (Esto lo comento porque por más que probaba lo que me proponía no funcionaba y tuve que tirar de la vieja confiable usando Stack OverFlow).

Por otro lado,tuve una “discusión” con la IA debido a que me propuso una función muy ineficiente respecto al tiempo.



```
for i in $indices; do
  if echo "$containers" | grep -q "^web$i$"; then
    docker stop web$i &>/dev/null
    docker rm -f web$i &>/dev/null
    echo -e "${greenColour}[+] Contenedor web$i detenido y
eliminado${endColour}"
    found=true
  fi
done
```

Respecto a este bucle de la función recorre contenedor por contenedor para pararlo y eliminarlo, cuando realmente sé desde el principio los “índices” de los servidores nginx y apache, **por lo que le propuse** que directamente use docker stop índes, docker rm indices. NO coloco directamente docker stop $(docker ps -aq) por si tengo otro contenedor que no quiero parar).

En resumen, aunque tenga algunos errores sigue siendo una buena herramienta, sin embargo hay que usarlo como herramienta y tener criterio propio. La IA me ha ayudado bastante para indicarme que podría realizar en cada apartado de las tareas, agilizar programación básica como son los archivos de dockerfile, docker-compose, etc. Además de ayudar en la gestión del script general que es “init.sh”.