

P1 Servidores Web y Almacenamiento

Mario Rodriguez Lopez

March 2024

Índice

B1. CONFIGURACIÓN DEL ENTORNO	2
B2. CREACIÓN DEL DOCKERFILE	2
B3. USO DE DOCKER COMPOSE	4
B4. DESPLIEGUE Y VERIFICACIÓN DE CONTENEDORES	8
B5. PRUEBAS BÁSICAS	10
A1. PERSONALIZACIÓN DEL DOCKERFILE	11
A2. CREACIÓN DE CONTENEDORES CON OTROS SERVIDORES WEB	13
A3. GESTIÓN AVANZADA DE REDES	16
A4. AUTOMATIZACIÓN CON SCRIPTS	18
A5. MONITOREO Y LOGGING	23

B1. CONFIGURACIÓN DEL ENTORNO

Siguiendo la P0 de contenerización se cumple lo que nos pide en la configuración del entorno. Además creamos el directorio y el fichero index.php como sigue:

```
mkdir web_MarioRgzLpz
```

```
1 <?php
2     $server_ip = $_SERVER[ 'SERVER_ADDR' ];
3     echo "SWAP-Mario Rodriguez Lopez";
4     echo "La direccion IP del servidor Apache es: " . $server_ip;
5 ?>
```

index.php

Para esta sección no he usado IA puesto que la configuración base es simple y se puede lograr siguiendo el guión y la creación del index.php es también muy simple teniendo solo que obtener el ip mediante la variable de servidor de php.

B2. CREACIÓN DEL DOCKERFILE

B2.1. Propuesta final

Aquí muestro la propuesta de DockerFile final para la tarea B2 aunque luego la modifiqué para evitar tener que hacer los dos comandos run y simplemente copiar el fichero de configuración apache. Esto lo muestro en la tarea A1. **DockerFile creado:**

```
# Usar una imagen base de Linux
FROM alpine:3.15.0

# Instalar Apache, PHP y herramientas de red (en este caso tenemos ping por defecto para
# comprobar conectividad)
RUN apk add \
    bash \
    php8-apache2 \
    php8 \
    apache2

# Habilitar el acceso de .htaccess para sobrescribir la configuración
RUN sed -i 's|    AllowOverride None|    AllowOverride All|g' /etc/apache2/httpd.conf
# Actualiza el directorio index para agregar archivos php
RUN sed -i 's|DirectoryIndex index.html|DirectoryIndex index.php index.html|g' /etc/
    apache2/httpd.conf

# Exponer el puerto 80 para Apache
EXPOSE 80

# Copia el archivo index.php al directorio raíz de Apache
COPY ./web_MarioRgzLpz/index.php /var/www/localhost/htdocs

# Comando predeterminado para iniciar Apache en primer plano
CMD [ "httpd", "-D", "FOREGROUND" ]
```

B2.2. Análisis propuesta IA

Pregunta a ChatGPT:

Crear un archivo Dockerfile en la raíz del proyecto llamado DockerfileApache_usuarioUGR.

-Usar una imagen base de Linux, instalar Apache, PHP y herramientas de red para comprobar conectividad entre máquinas.

-Ademas añadir un archivo index.php que nos permita ver en local dicho archivo a traves de cualquier buscador web.

Respuesta de ChatGPT

```
# Usar una imagen base de Linux
FROM ubuntu:latest

# Actualizar el repositorio e instalar Apache, PHP y herramientas de red
RUN apt-get update && \
    apt-get install -y apache2 php

# Exponer el puerto 80 para Apache
EXPOSE 80

# Copia el archivo index.php al directorio raiz de Apache
COPY index.php /var/www/html/

# Comando predeterminado para iniciar Apache en primer plano
CMD [ "apache2ctl", "-D", "FOREGROUND" ]
```

Análisis y cambios

La propuesta de ChatGPT es válida y nos valdría para nuestras prácticas, sin embargo como vamos a necesitar muchos contenedores cuanto más reducido sea el espacio de la imagen a usar mejor. Es por esto que yo he decidido usar una imagen de alpine linux en lugar de ubuntu, ya que esta es mucho más ligera y nos permite realizar lo mismo que la imagen de ubuntu.

He cambiado por tanto la imagen base del DockerFile y tambien la forma en la que se instala es distinta para alpine ya que usa el gestor apk en lugar de apt. Además no tenemos el directorio /var/www/html si no que se usa /var/www/localhost/htdocs por tanto debemos modificar esto para lograr lo que queremos. Es necesario instalar bash en alpine ya que no viene por defecto. De esta forma nos queda una imagen que pesa menos de 50 MB mientras que usando la de ubuntu obtenemos una con unos 240 MB suponiendo un gran aumento.

Además es necesario modificar la configuración de apache si queremos que, al acceder a la IP de nuestro contenedor, se muestre el fichero index.php que estamos copiando y no el que apache tiene por defecto que es el index.html. En esta parte lo modifiko de manera manual usando RUN sed aunque luego en la tarea avanzada A1 modifiko el fichero de configuración directamente para permitir configuraciones personalizadas.

B3. USO DE DOCKER COMPOSE

B3.1. Propuesta final

Archivo docker compose creado:

```
version: '4.0'

services:
  web1:
    image: mariorgzlpz--apache--image:p1
    volumes:
      - ./web_MarioRgzLpz:/var/www/localhost/htdocs
      - ./configsphp:/etc/php8/conf.d
    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:p1
    volumes:
      - ./web_MarioRgzLpz:/var/www/localhost/htdocs
      - ./configsphp:/etc/php8/conf.d
    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:p1
    volumes:
      - ./web_MarioRgzLpz:/var/www/localhost/htdocs
      - ./configsphp:/etc/php8/conf.d
    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:p1
    volumes:
      - ./web_MarioRgzLpz:/var/www/localhost/htdocs
      - ./configsphp:/etc/php8/conf.d
    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:p1
    volumes:
      - ./web_MarioRgzLpz:/var/www/localhost/htdocs
      - ./configsphp:/etc/php8/conf.d
    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:p1
    volumes:
      - ./web_MarioRgzLpz:/var/www/localhost/htdocs
      - ./configsphp:/etc/php8/conf.d
```

```

    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:p1
  volumes:
    - ./web_MarioRgzLpz:/var/www/localhost/htdocs
    - ./configsphp:/etc/php8/conf.d
  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:p1
  volumes:
    - ./web_MarioRgzLpz:/var/www/localhost/htdocs
    - ./configsphp:/etc/php8/conf.d
  container_name: web8
  networks:
    red_web:
      ipv4_address: 192.168.10.9
    red_servicios:
      ipv4_address: 192.168.20.9

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

```

B3.2. Análisis propuesta IA

Pregunta a ChatGPT:

Crear una imagen llamada usuarioUGR-apache-image:p1 a partir del Dockerfile DockerfileApache.usuarioUGR.

-Crear 8 contenedores llamados webX, donde X es un número de 1 a 8 con volúmenes donde se monte el directorio “web_usuarioUGR” en el directorio raíz de Apache en el contenedor.

-Añadir las dos redes al contenedor, una red llamada red_web con dirección 192.168.10.0/24 y otra red llamada red_servicios con dirección 192.168.20.0/24.

Debes usar docker compose para crear los contenedores.

Respuesta de ChatGPT

```
version: '3'

services:
  web1:
    image: usuarioUGR-apache-image:p1
    volumes:
      - ./web_usuarioUGR:/usr/local/apache2/htdocs
    networks:
      - red_web
      - red_servicios

  web2:
    image: usuarioUGR-apache-image:p1
    volumes:
      - ./web_usuarioUGR:/usr/local/apache2/htdocs
    networks:
      - red_web
      - red_servicios

  web3:
    image: usuarioUGR-apache-image:p1
    volumes:
      - ./web_usuarioUGR:/usr/local/apache2/htdocs
    networks:
      - red_web
      - red_servicios

  web4:
    image: usuarioUGR-apache-image:p1
    volumes:
      - ./web_usuarioUGR:/usr/local/apache2/htdocs
    networks:
      - red_web
      - red_servicios

  web5:
    image: usuarioUGR-apache-image:p1
    volumes:
      - ./web_usuarioUGR:/usr/local/apache2/htdocs
    networks:
      - red_web
      - red_servicios

  web6:
    image: usuarioUGR-apache-image:p1
    volumes:
      - ./web_usuarioUGR:/usr/local/apache2/htdocs
    networks:
      - red_web
      - red_servicios

  web7:
    image: usuarioUGR-apache-image:p1
    volumes:
      - ./web_usuarioUGR:/usr/local/apache2/htdocs
    networks:
      - red_web
      - red_servicios
```

```

web8:
  image: usuarioUGR-apache-image:p1
  volumes:
    - ./web_usuarioUGR:/usr/local/apache2/htdocs
  networks:
    - red_web
    - red_servicios

networks:
  red_web:
    ipam:
      driver: default
      config:
        - subnet: 192.168.10.0/24

  red_servicios:
    ipam:
      driver: default
      config:
        - subnet: 192.168.20.0/24

```

Análisis y cambios

La propuesta de ChatGPT podría valer pero deberíamos de hacer varios ajustes ya que por defecto no se asignarían las IPs correctamente al no ir creandose los contenedores en orden. Para solucionar esto yo he asignado a cada contenedor directamente la IP que le corresponde, aunque esto no es muy escalable ya que deberíamos de asignar IPs cada vez que añadimos un nuevo contenedor web. Para solucionar esto he creado un script que de manera automática te crea el docker compose pasandole el numero de contenedores que queremos. Además por la forma en la que asigna las IPs docker no siempre se asignaran todas las IPs de manera correcta y deberemos añadir de manera manual la IP que no sea correcta. En la tarea avanzada A4 muestro una posible solución para esto.

También he añadido que el driver de las networks sea el bridge (aunque por defecto ya es este) para poder modificarlo si quisieramos en el futuro.

Además ya he puesto el volumen con el que cargaremos el fichero de configuración de php que usamos en la tarea A1.

B4. DESPLIEGUE Y VERIFICACIÓN DE CONTENEDORES

-Lanzar los contenedores

Usamos el comando docker compose up para lanzar nuestros contenedores siguiendo el archivo docker-compose creado en el apartado anterior.

```
parrot: ~/.../SWAP/P1
-- docker compose up
[*] Running 8/0
✓ Container web1 Created 0.0s
✓ Container web2 Created 0.0s
✓ Container web3 Created 0.0s
✓ Container web4 Created 0.0s
✓ Container web5 Created 0.0s
✓ Container web6 Created 0.0s
✓ Container web7 Created 0.0s
✓ Container web8 Created 0.0s
Attaching to web1, web2, web3, web4, web5, web6, web7, web8
```

-Verificar la ejecucion

Usamos el comando docker ps para ver si nuestros contenedores estan correctamente creados y ejecutandose.

```
parrot: ~/.../DGIIM/SWAP
-- docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS        NAMES
d4e0c57aa518   mariorgzlpz-apache-image:p1        "httpd -D FOREGROUND"   20 minutes ago Up 5 minutes  80/tcp       web8
9e9e92d11476   mariorgzlpz-apache-image:p1        "httpd -D FOREGROUND"   20 minutes ago Up 5 minutes  80/tcp       web7
496bb5f9781c   mariorgzlpz-apache-image:p1        "httpd -D FOREGROUND"   20 minutes ago Up 5 minutes  80/tcp       web6
b6137e6c27b9   mariorgzlpz-apache-image:p1        "httpd -D FOREGROUND"   20 minutes ago Up 5 minutes  80/tcp       web5
78c8124c1497   mariorgzlpz-apache-image:p1        "httpd -D FOREGROUND"   20 minutes ago Up 5 minutes  80/tcp       web4
a77401bf543a   mariorgzlpz-apache-image:p1        "httpd -D FOREGROUND"   20 minutes ago Up 5 minutes  80/tcp       web3
b184c980b45d   mariorgzlpz-apache-image:p1        "httpd -D FOREGROUND"   20 minutes ago Up 5 minutes  80/tcp       web2
0beb37a2eeba   mariorgzlpz-apache-image:p1        "httpd -D FOREGROUND"   20 minutes ago Up 5 minutes  80/tcp       web1
```

-Comprobar la asignacion de IPs de cada contenedor

Para esto he creado un script que muestro y explico en la tarea avanzada A4, aqui solo mostrare la salida de dicho script y que nos enseña si la asignación ha sido correcta o no.

```
_omb_util_prompt_command_hook - Parrot Terminal
File Edit View Search Terminal Help

parrot: ~/.../P1/scripts
-- ./comprobar_IPs.sh
Direcciones IP de los contenedores:
-----
Contenedor web1 - IP de la red web: 192.168.10.2
Contenedor web2 - IP de la red web:
Contenedor web3 - IP de la red web: 192.168.10.4
Contenedor web4 - IP de la red web: 192.168.10.5
Contenedor web5 - IP de la red web: 192.168.10.6
Contenedor web6 - IP de la red web: 192.168.10.7
Contenedor web7 - IP de la red web: 192.168.10.8
Contenedor web8 - IP de la red web: 192.168.10.9
Contenedor web1 - IP de la red de servicios: 192.168.20.2
Contenedor web2 - IP de la red de servicios: 192.168.20.3
Contenedor web3 - IP de la red de servicios: 192.168.20.4
Contenedor web4 - IP de la red de servicios: 192.168.20.5
Contenedor web5 - IP de la red de servicios: 192.168.20.6
Contenedor web6 - IP de la red de servicios: 192.168.20.7
Contenedor web7 - IP de la red de servicios: 192.168.20.8
Contenedor web8 - IP de la red de servicios: 192.168.20.9

parrot: ~/.../P1/scripts
```

Aqui haciendo uso del script para asignar correctamente las IPs y mostrando el resultado por pantalla.


```
_omb_util_prompt_command_hook - Parrot Terminal
File Edit View Search Terminal Help

parrot: ~/.../P1/scripts
- ./ajustar_IPs.sh
Verificando y ajustando direcciones IP de los contenedores:
-----
Proceso completado.

parrot: ~/.../P1/scripts
- ./comprobar_IPs.sh
Direcciones IP de los contenedores:
-----
Contenedor web1 - IP de la red web: 192.168.10.2
Contenedor web2 - IP de la red web: 192.168.10.3
Contenedor web3 - IP de la red web: 192.168.10.4
Contenedor web4 - IP de la red web: 192.168.10.5
Contenedor web5 - IP de la red web: 192.168.10.6
Contenedor web6 - IP de la red web: 192.168.10.7
Contenedor web7 - IP de la red web: 192.168.10.8
Contenedor web8 - IP de la red web: 192.168.10.9
Contenedor web1 - IP de la red de servicios: 192.168.20.2
Contenedor web2 - IP de la red de servicios: 192.168.20.3
Contenedor web3 - IP de la red de servicios: 192.168.20.4
Contenedor web4 - IP de la red de servicios: 192.168.20.5
Contenedor web5 - IP de la red de servicios: 192.168.20.6
Contenedor web6 - IP de la red de servicios: 192.168.20.7
Contenedor web7 - IP de la red de servicios: 192.168.20.8
Contenedor web8 - IP de la red de servicios: 192.168.20.9

parrot: ~/.../P1/scripts
```

-Comprobar conectividad entre contenedores

Para esto he creado un script que muestro y explico en la tarea avanzada A4, aqui solo mostraré la salida de dicho script y que nos dice si hay conectividad o no entre dos contenedores.

Puesto que probar la conectividad de todos los contenedores produciría una salida muy grande, basta con comprobar que las asignaciones de ip son correctas y probaremos la conectividad entre el contenedor 1 y el contenedor 2 y podremos suponer que si hay conectividad entre esos dos en el resto debe haber conectividad (por defecto sin definir reglas de enrutamiento ni bloquear IPs específicas)

```
parrot: ~/.../P1/scripts
- ./conectividad.sh
Ping desde el contenedor 1 al contenedor 2 (IP 1):
PING 192.168.10.3 (192.168.10.3): 56 data bytes
64 bytes from 192.168.10.3: seq=0 ttl=64 time=0.178 ms
64 bytes from 192.168.10.3: seq=1 ttl=64 time=0.075 ms

--- 192.168.10.3 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.075/0.126/0.178 ms
Ping desde el contenedor 1 al contenedor 2 (IP 2):
PING 192.168.20.3 (192.168.20.3): 56 data bytes
64 bytes from 192.168.20.3: seq=0 ttl=64 time=0.138 ms
64 bytes from 192.168.20.3: seq=1 ttl=64 time=0.131 ms

--- 192.168.20.3 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.131/0.134/0.138 ms
Ping desde el contenedor 2 al contenedor 1 (IP 1):
PING 192.168.10.2 (192.168.10.2): 56 data bytes
64 bytes from 192.168.10.2: seq=0 ttl=64 time=0.127 ms
64 bytes from 192.168.10.2: seq=1 ttl=64 time=0.062 ms

--- 192.168.10.2 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.062/0.094/0.127 ms
Ping desde el contenedor 2 al contenedor 1 (IP 2):
PING 192.168.20.2 (192.168.20.2): 56 data bytes
64 bytes from 192.168.20.2: seq=0 ttl=64 time=0.075 ms
64 bytes from 192.168.20.2: seq=1 ttl=64 time=0.153 ms

--- 192.168.20.2 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.075/0.114/0.153 ms
```

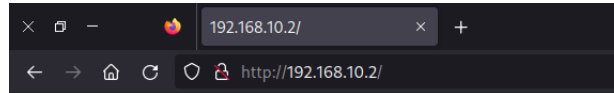
B4.1. Análisis propuesta IA

Para evitar redundancia el uso de la inteligencia artificial se ve reflejado en los scripts y que muestro en la tarea avanzada A4 por tanto aqui solo muestro la salida del script.

B5. PRUEBAS BÁSICAS

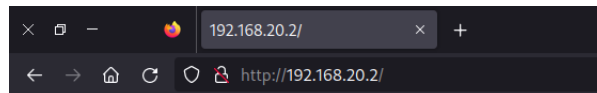
En este apartado no he usado inteligencia artificial ni ningún script puesto que lo que debemos hacer es ingresar las IPs directamente en el navegador y comprobar que la información sea correcta.

Para evitar ser repetitivo solo muestro por pantalla 3 de los 8 contenedores que son los dos primeros y el ultimo y viendo que las IPs son correctas por el apartado anterior es claro que van a ser todas correctas.



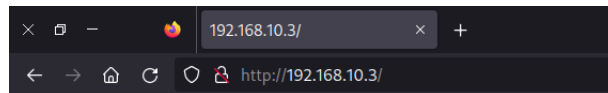
SWAP-Mario Rodriguez Lopez

La direccion IP del servidor Apache es: 192.168.10.2



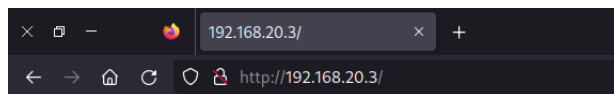
SWAP-Mario Rodriguez Lopez

La direccion IP del servidor Apache es: 192.168.20.2



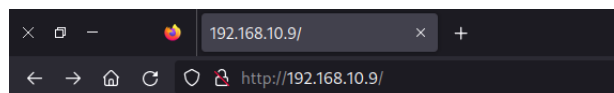
SWAP-Mario Rodriguez Lopez

La direccion IP del servidor Apache es: 192.168.10.3



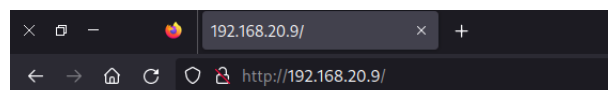
SWAP-Mario Rodriguez Lopez

La direccion IP del servidor Apache es: 192.168.20.3



SWAP-Mario Rodriguez Lopez

La direccion IP del servidor Apache es: 192.168.10.9



SWAP-Mario Rodriguez Lopez

La direccion IP del servidor Apache es: 192.168.20.9

A1. PERSONALIZACIÓN DEL DOCKERFILE

A1.1. Configuraciones realizadas

Lo principal para poder cargar las configuraciones es añadir lo que hara que tengamos los ficheros de configuracion en nuestros contenedores. Para las configuraciones de php he usado un volumen mientras que para las configuraciones de apache he decidido modificarlo estáticamente (porque el volumen me estaba dando problemas).

Añado el volumen a docker compose

```
#Todo lo anterior
volumes:
  # Montamos el directorio para las configuraciones de php
  - ./configsphp:/etc/php8/conf.d
```

Modifíco el DockerFile para copiar el fichero de configuración apache

```
# Usar una imagen base de Linux
FROM alpine:3.15.0

# Instalar Apache, PHP y herramientas de red (en este caso tenemos ping por defecto para
comprobar conectividad)
RUN apk add --no-cache \
    bash \
    php8-apache2 \
    php8 \
    apache2

COPY configsapache/httpd.conf /etc/apache2/httpd.conf

# Habilitar el acceso de .htaccess para sobrescribir la configuracion
# RUN sed -i 's|    AllowOverride None|    AllowOverride All|g' /etc/apache2/httpd.conf
# Actualiza el directorio index para agregar archivos php
# RUN sed -i 's|DirectoryIndex index.html|DirectoryIndex index.php index.html|g' /etc/
    apache2/httpd.conf

# Exponer el puerto 80 para Apache
EXPOSE 80

# Comando predeterminado para iniciar Apache en primer plano
CMD ["httpd", "-D", "FOREGROUND"]
```

Archivo apache-config.conf creado: Para evitar crear 2 capas más al crear la imagen desde el DockerFile podemos añadir lo que tenemos en el RUN sed a nuestro archivo de configuración. Dejo reflejado en el fichero de configuración como se hace esto además de como se puede modificar el puerto en el que escucha y el nombre del servidor. Debemos tener el archivo httpd.conf y modificar las líneas siguientes:

```
# Configurar el nombre del servidor(Con esto evitamos el mensaje de docker a la hora de
crear los contenedores)
ServerName localhost

# Configurar el puerto en el que escucha el servidor(80 por defecto)
# Listen 80

# Habilitar el acceso de .htaccess para sobrescribir la configuracion(para que podamos
ver los ficheros)
<Directory /var/www/localhost/htdocs>
    AllowOverride All
</Directory>

# Definir un archivo de indice predeterminado(para que se muestre el php)
DirectoryIndex index.php index.html
```

Archivo php.ini creado: Puesto que el php de las practicas va a ser muy básico no seria necesario modificar nada, pero aqui añado algunas configuraciones bastante comunes basandome en el archivo que me ha proporcionado ChatGPT.

```

; Deshabilitar la exposicion de la version de PHP en encabezados HTTP
expose_php = Off

; Habilitar la visualizacion de errores en pantalla (Nos sirve para depurar)
display_errors = On

; Evita que en las direcciones aparezcan los IDs de las sesiones
session.use_trans_sid = Off

; Establecer limite de tiempo de ejecucion de scripts a 30 segundos
max_execution_time = 30

; Aumentar el limite de memoria permitido para scripts a 128 megabytes
memory_limit = 128M

; Activar el registro de errores de PHP
error_reporting = E_ALL & ~E_NOTICE

; Establecer la zona horaria a UTC
date.timezone = "UTC"

```

A1.2. Análisis propuesta IA

Preguntas a ChatGPT:

Pregunta1: Modificar el Dockerfile para incluir configuraciones personalizadas de Apache o PHP.

Respuesta:

```

# Usar una imagen base de Linux con Apache y PHP
FROM php:8.0-apache

# Instalar herramientas adicionales si es necesario
RUN apt-get update && apt-get install -y \
    # Herramientas adicionales aqui si es necesario \
    && rm -rf /var/lib/apt/lists/*

# Copiar archivos de configuracion personalizados de Apache
COPY apache-config.conf /etc/apache2/sites-available/000-default.conf

# Copiar archivos de configuracion personalizados de PHP
COPY php.ini /usr/local/etc/php/php.ini

# Exponer el puerto 80 para Apache
EXPOSE 80

# Copia el archivo index.php al directorio raiz de Apache
COPY ./web/MarioRgzLpz/index.php /var/www/html/index.php

# Comando predeterminado para iniciar Apache en primer plano
CMD ["apache2-foreground"]

```

Pregunta2: Proporcioname dos configuraciones personalizadas una de php y otra de apache

Respuesta:

```

; Ejemplo de configuracion personalizada de PHP
; Deshabilitar la exposicion de la version de PHP en encabezados HTTP
expose_php = Off

; Establecer limite de tiempo de ejecucion de scripts a 30 segundos
max_execution_time = 30

; Aumentar el limite de memoria permitido para scripts a 128 megabytes
memory_limit = 128M

; Activar el registro de errores de PHP
error_reporting = E_ALL & ~E_NOTICE

; Establecer la zona horaria a UTC
date.timezone = "UTC"

```

php.ini

```
# Ejemplo de configuracion personalizada de Apache
# Deshabilitar el listado de directorios
<Directory /var/www/html>
    Options -Indexes
</Directory>
# Configurar el nombre del servidor
ServerName localhost
# Habilitar el acceso de .htaccess para sobrescribir la configuracion
<Directory /var/www/html>
    AllowOverride All
</Directory>
# Definir un archivo de indice predeterminado
DirectoryIndex index.php index.html

# Establecer el formato de registro personalizado
LogFormat "%h %l %u %t \"%r\" \"%s %b \"%{Referer}i \"%{User-agent}i\" combined
CustomLog /var/log/apache2/access.log combined
ErrorLog /var/log/apache2/error.log
```

apache-config.conf

Análisis y cambios La primera respuesta que nos proporciona ChatGPT no es recomendable puesto que nos esta modificando la imagen base de nuestro DockerFile, algo que no se debería de hacer ya que se ha usado dicha imagen por razones de optimización. La forma en la que se añaden los archivos de configuración es correcta si queremos que sean configuraciones estáticas. Yo he decidido hacer una estática (la configuración de apache) ya que me estaba dando muchos problemas al cargar nuevas configuraciones mediante un volumen. Al ser estática nos obliga a rehacer la imagen cada vez que modifiquemos la configuración. Tambien he hecho una dinámica(la de php) a la que añado un volumen en el directorio donde se añaden las configuraciones de php de manera que para que surta efecto bastará con relanzar los contenedores con el docker compose.

La segunda respuesta si es correcta porque como no hemos especificado las configuraciones nos ha dado dos configuraciones personalizadas tanto de php como de apache. Usando estas como base yo he modificado las configuraciones para realizar lo que estabamos añadiendo mediante el comando RUN sed y a parte tambien modificar ciertas cosas más convenientes y que se explican en cada archivo de configuración.

A2. CREACIÓN DE CONTENEDORES CON OTROS SERVIDORES WEB

A2.1. Propuesta final

He decidido hacer un servidor web con nginx para lo cual he creado una imagen con otro DockerFile que muestro aquí debajo, además de modificar la configuración base de nginx para poder visualizar nuestro index.php.

DockerFile creado:

```
# Utilizar la imagen base de Alpine
FROM alpine:3.15.0

# Instalar Nginx y PHP
RUN apk --update --no-cache add nginx php8 php8-fpm bash

# Copiar la configuracion de Nginx
COPY configsnginx/nginx.conf /etc/nginx/nginx.conf

# Crear el directorio para el socket de PHP-FPM
RUN mkdir -p /run/php

# Copiar el archivo PHP de ejemplo
COPY web_MarioRgzLpz/index.php /var/www/localhost/htdocs/index.php

# Exponer el puerto 80
EXPOSE 80

# Comando para iniciar Nginx y PHP-FPM
CMD nginx && php-fpm8 -F
```

Fichero de configuración de nginx:

```
worker_processes 1;

events {
    worker_connections 1024;
}

http {
    server {
        listen 80;
        server_name localhost;

        root /var/www/html;
        index index.php index.html index.htm;

        location / {
            try_files $uri $uri/ /index.php?$args;
        }

        location ~ \.php$ {
            fastcgi_pass 127.0.0.1:9000;
            fastcgi_index index.php;
            include fastcgi_params;
            fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
            fastcgi_param PATH_INFO $fastcgi_path_info;
        }
    }
}
```

A2.2. Análisis propuesta IA

Pregunta a ChatGPT:

Teniendo este DockerFile como base: (Mi DockerFile creado en B2) Creame otro DockerFile en el que el servidor web sea nginx en lugar de apache.

Respuesta de ChatGPT

```
# Utilizar una imagen base de Linux
FROM alpine:3.15.0

# Instalar Nginx y PHP-FPM
RUN apk add --no-cache \
    nginx \
    php8 \
    php8-fpm \
    php8-mysql \
    php8-json \
    php8-openssl \
    ...Unas 20 instalaciones mas de php8 con servicios...

# Configurar Nginx
COPY nginx.conf /etc/nginx/nginx.conf
COPY default.conf /etc/nginx/conf.d/default.conf

# Configurar PHP-FPM
COPY php-fpm.conf /etc/php8/php-fpm.conf
COPY www.conf /etc/php8/php-fpm.d/www.conf

# Crear directorio para los archivos del sitio web
RUN mkdir -p /var/www/html

# Exponer el puerto 80 para Nginx
EXPOSE 80

# Comando predeterminado para iniciar Nginx y PHP-FPM
CMD ["nginx", "-g", "daemon off;"]
```

También me proporciona los fichero de configuración de nginx y php-fpm.

```
user nginx;
worker_processes auto;
error_log /var/log/nginx/error.log;
pid /run/nginx.pid;

events {
    worker_connections 1024;
}

http {
    include /etc/nginx/mime.types;
    default_type application/octet-stream;
    log_format main '$remote_addr - $remote_user [$time_local] "$request" '
        '$status $body_bytes_sent "$http_referer" '
        '"$http_user_agent" "$http_x_forwarded_for"';
    access_log /var/log/nginx/access.log main;
    sendfile on;
    keepalive_timeout 65;
    include /etc/nginx/conf.d/*.conf;
}
```

nginx.conf

```
server {
    listen 80;
    server_name localhost;
    root /var/www/html;
    index index.php index.html index.htm;

    location / {
        try_files $uri $uri/ /index.php?$query_string;
    }

    location ~ /\.php$ {
        include fastcgi_params;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        fastcgi_pass unix:/run/php/php8.0-fpm.sock;
        fastcgi_index index.php;
    }

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

default.conf

```
[global]
pid = /run/php/php8.0-fpm.pid
error_log = /var/log/php8.0-fpm.log

[www]
user = nginx
group = nginx
listen = /run/php/php8.0-fpm.sock
listen.owner = nginx
listen.group = nginx
```

php-fpm.conf

```
[www]
user = nginx
group = nginx
pm = dynamic
listen = /run/php/php8.0-fpm.sock
listen.owner = nginx
listen.group = nginx
```

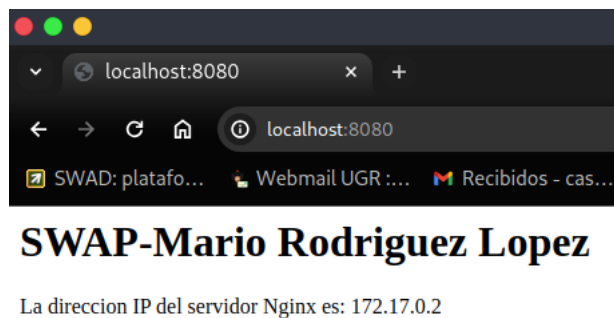
Análisis y cambios El DockerFile que me proporciona copia las configuraciones correctamente, aunque instala muchos paquetes de php8 que yo no quería haciendo mi imagen mas pesada de lo que la necesito. Por esta razón y que la solución de ChatGPT no me funcionaba al final opte por seguir un tutorial que había en una pagina web [Tutorial nginx-php](#) en el que simplemente añadía la configuración de nginx y modificando el parámetro **fastcgi_pass unix:/run/php/php8.0-fpm.sock**; por **fastcgi_pass 127.0.0.1:9000**; que es donde en el fichero de configuracion por defecto de php-fpm esta escuchando (esto es posible verlo haciendo ifconfig).

Ya solo faltaría crear la imagen con el DockerFile final y crear el contenedor usando

```
1 docker build -t mariorgzlp-nginx-image -f DockerfileNginx_MarioRgzLpz .
2 docker run -d -p 8080:80 (o el puerto que queramos) mariorgzlpz-nginx-image
```

Con esto ya podremos ver nuestra pagina index.php con el servidor nginx en localhost:8080 o el puerto en el que hayamos decidido escuchar cuando creamos el contenedor con docker run.

Lo que nos muestra es esto (He cambiado el nombre de Apache a Nginx del fichero index.php inicial)



La IP que nos sale es la que ha asignado docker de manera automática al contenedor creado.

A3. GESTIÓN AVANZADA DE REDES

Para esto simplemente he aprovechado el script de conectividad para realizar algunas configuraciones de red mediante docker disconnect y iptables y comprobar si sigue existiendo conectividad. Como ejemplo desconecto los contenedores web1 de la red web y web2 de la red servicios y compruebo si sigue habiendo conectividad entre los contenedores. Nos arroja el siguiente resultado: (Los comandos usados se muestran por pantalla)

```
parrot: ~/.../P1/scripts
$ docker network disconnect p1_red_web web1
$ docker network disconnect p1_red_servicios web2
parrot: ~/.../P1/scripts
$ ./conectividad.sh
Ping desde el contenedor 1 al contenedor 2 (IP 1):
PING 192.168.10.3 (192.168.10.3): 56 data bytes

Redes asignadas con Docker
--- 192.168.10.3 ping statistics ---
2 packets transmitted, 0 packets received, 100% packet loss
Ping desde el contenedor 1 al contenedor 2 (IP 2):
PING 192.168.20.3 (192.168.20.3): 56 data bytes

Alfaya App Recomendados
--- 192.168.20.3 ping statistics ---
2 packets transmitted, 0 packets received, 100% packet loss
Ping desde el contenedor 2 al contenedor 1 (IP 1):
PING 192.168.10.2 (192.168.10.2): 56 data bytes

Docker Redes Asignadas
--- 192.168.10.2 ping statistics ---
2 packets transmitted, 0 packets received, 100% packet loss
Ping desde el contenedor 2 al contenedor 1 (IP 2):
PING 192.168.20.2 (192.168.20.2): 56 data bytes

Alfaya App Recomendados
--- 192.168.20.2 ping statistics ---
2 packets transmitted, 0 packets received, 100% packet loss
```


Esto nos indica que hemos perdido toda conectividad entre ambos contenedores. Ahora configuro la red para crear un puente en las interfaces y así poder crear conectividad entre la IP de la red servicios que tiene el contenedor web1 y la IP de la red web que tiene el contenedor web2. (Ya que ambos contenedores los hemos desconectado antes de las otras redes usando docker disconnect)

Primero con ifconfig averiguo los nombres de las interfaces y luego empleo iptables para configurar la red:

```
parrot: ~/.../P1/scripts
└─ ifconfig
br-20a672edb016: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.10.1 netmask 255.255.255.0 broadcast 192.168.10.255
    inet6 fe80::42:1c:ff:fe9d:af1 prefixlen 64 scopeid 0x20<link>
    ether 02:42:1c:9d:0a:f1 txqueuelen 0 (Ethernet)
    RX packets 108 bytes 6816 (6.6 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 75 bytes 11534 (11.2 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

br-ab9dda9d197f: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.20.1 netmask 255.255.255.0 broadcast 192.168.20.255
    inet6 fe80::42:94:ff:fe1c:45:ff prefixlen 64 scopeid 0x20<link>
    ether 02:42:94:1c:45:ff txqueuelen 0 (Ethernet)
    RX packets 108 bytes 6816 (6.6 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 76 bytes 11644 (11.3 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

parrot: ~/.../P1/scripts
└─ sudo iptables -I FORWARD -i br-20a672edb016 -o br-ab9dda9d197f -j ACCEPT
[sudo] password for mariorl:

parrot: ~/.../P1/scripts
└─ sudo iptables -I FORWARD -i br-ab9dda9d197f -o br-20a672edb016 -j ACCEPT
```

Así, después de usar los comandos anteriores comprobamos la conectividad:

```
parrot: ~/.../P1/scripts
└─ ./conectividad.sh
Ping desde el contenedor 1 al contenedor 2 (IP 1):
PING 192.168.10.3 (192.168.10.3): 56 data bytes
64 bytes from 192.168.10.3: seq=0 ttl=63 time=0.114 ms
64 bytes from 192.168.10.3: seq=1 ttl=63 time=0.079 ms

--- 192.168.10.3 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.079/0.096/0.114 ms
Ping desde el contenedor 1 al contenedor 2 (IP 2):
PING 192.168.20.3 (192.168.20.3): 56 data bytes

--- 192.168.20.3 ping statistics ---
2 packets transmitted, 0 packets received, 100% packet loss
Ping desde el contenedor 2 al contenedor 1 (IP 1):
PING 192.168.10.2 (192.168.10.2): 56 data bytes

--- 192.168.10.2 ping statistics ---
2 packets transmitted, 0 packets received, 100% packet loss
Ping desde el contenedor 2 al contenedor 1 (IP 2):
PING 192.168.20.2 (192.168.20.2): 56 data bytes
64 bytes from 192.168.20.2: seq=0 ttl=63 time=0.103 ms
64 bytes from 192.168.20.2: seq=1 ttl=63 time=0.092 ms

--- 192.168.20.2 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.092/0.097/0.103 ms
```

Como vemos hemos recuperado la conectividad entre la IP de la red servicios del contenedor web1 y la IP de la red web del contenedor web2 que era lo que buscábamos.

A4. AUTOMATIZACIÓN CON SCRIPTS

He decidido crear varios scripts de manera que me ayuden a realizar tareas necesarias de otras partes.

A4.1. Comprobacion de IPs

He creado este script para comprobar si la asignación de IPs ha sido correcta para cada contenedor. Lo que hace es obtiene la ID de cada red y a partir de eso con el comando docker inspect pasandole la id de cada red podremos obtener para cada contenedor la IP en dicha red.

A4.1.1 Script final

```
#!/bin/bash

# Nombre de las redes de los contenedores web
RED_WEB="red_web"

# Nombre de la red de los servicios
RED_SERVICIOS="red_servicios"

# Funcion para obtener el ID completo de una red dado un fragmento de su nombre
obtener_id_red() {
    local fragmento_nombre=$1
    local network_name=$(docker network ls --filter name=${fragmento_nombre} --format "{{.Name}}")
    network_id=$(docker network inspect $network_name --format "{{.Id}}")
    echo "$network_id"
}

# Obtener el ID completo de la red web
NETWORK_ID_WEB=$(obtener_id_red "$RED_WEB")

# Obtener el ID completo de la red de servicios
NETWORK_ID_SERVICIOS=$(obtener_id_red "$RED_SERVICIOS")

# Funcion para obtener la IP de un contenedor en la red especificada por su ID
obtener_ip() {
    docker inspect -f '{{range .NetworkSettings.Networks}}{{if eq .NetworkID "'$1'"}}{{.IPAddress}}{{end}}{{end}}' "$2"
}

echo "Direcciones IP de los contenedores:"
echo "_____ "

# Iterar sobre los contenedores web
for i in {1..8}; do
    CONTAINER_NAME="web$i"
    IP_WEB=$(obtener_ip "$NETWORK_ID_WEB" "$CONTAINER_NAME")
    echo "Contenedor $CONTAINER_NAME - IP de la red web: $IP_WEB"
done

# Iterar sobre los contenedores de servicios
for i in {1..8}; do
    CONTAINER_NAME="web$i"
    IP_SERVICIOS=$(obtener_ip "$NETWORK_ID_SERVICIOS" "$CONTAINER_NAME")
    echo "Contenedor $CONTAINER_NAME - IP de la red de servicios: $IP_SERVICIOS"
done
```

A4.1.2 Analisis IA

Preguntas:

1. Tengo este docker compose(paso el docker compose anterior) que me va a generar 8 contenedores cada uno con unas IPs especificas. Necesito un script bash que me proporcione las IPs de cada contenedor para ver si se ha realizado la asignacion de manera correcta.
2. Quiero poder filtrar por un trozo del nombre de la red para poder obtener el ID completo de la red.

Respuesta1 ChatGPT:

```
#!/bin/bash

# Nombre de la red de los contenedores web
RED_WEB="red_web"

# Nombre de la red de los servicios
RED_SERVICIOS="red_servicios"

# Funcion para obtener la IP de un contenedor
obtener_ip() {
    docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' "$1"
}

echo "Direcciones IP de los contenedores:"
echo "_____ "

# Iterar sobre los contenedores web
for i in {1..8}; do
    CONTAINER_NAME="web$i"
    IP_WEB=$(obtener_ip "$CONTAINER_NAME")
    echo "Contenedor $CONTAINER_NAME - IP de la red web: $IP_WEB"
done

# Iterar sobre los contenedores de servicios
for i in {1..8}; do
    CONTAINER_NAME="web$i"
    IP_SERVICIOS=$(obtener_ip "$CONTAINER_NAME")
    echo "Contenedor $CONTAINER_NAME - IP de la red de servicios: $IP_SERVICIOS"
done
```

Respuesta2 ChatGPT:

```
# Ejecutar el comando para obtener la lista de redes y filtrar por el nombre especifico
NETWORK_NAME=$(docker network ls --filter name=red_web --format '{{.Name}}')

# Inspeccionar la red para obtener su ID completo
NETWORK_ID=$(docker network inspect $NETWORK_NAME --format '{{.Id}}')

# Imprimir el ID completo de la red
echo "El ID completo de la red $NETWORK_NAME es: $NETWORK_ID"
```

Analisis

El primer script proporcionado no era correcto del todo porque obtenia las IPs asignadas a cada contenedor y como cada contenedor tiene dos IPs, sacaba las IPs concatenadas y de manera erronea. Para solucionar esto he pensado en obtener el IP del contenedor en una red especifica a partir del ID de la red y usando docker inspect. Esto me lo soluciona ChatGPT bien. Tras esto ya solo falta usar una busqueda de IP en la funcion obtener_ip y que filtre por el ID de la red. De esta forma me queda el script comprobar_IPs que es el que muestro al principio y que genera la salida que se ve en la tarea basica B4 en la parte de comprobacion de IPs.

A4.2. Arreglar IPs

Como ya hemos dicho antes por la forma en la que docker asigna las IPs se produzcan algunos fallos aun asignando IPs de manera manual en el docker compose. Por esta razon usaremos una script para modificar las IPs que no sean correctas.

A4.2.1 Script final

El script final es el mismo que me ha dado ChatGPT porque hace la funcion que yo le pido de manera correcta.

A4.2.2 Analisis IA

Pregunta a ChatGPT:

Tengo este docker compose (paso el docker compose anterior) cuando lo ejecuto no se me colocan siempre todas las IPs de manera correcta. Tengo este script: (le paso el script de comprobar IPs anterior). Necesito un script que se encargue de ver si en la red web la ip asignada al contenedor webX es 192.168.10.(X+1) y si no lo es que lo modifique y le asigne esa IP, y para la red servicios que la ip asignada al contenedor

webX sea 192.168.20.(X+1) y si no lo es que lo modifique. Tambien puede ser que el contenedor no este conectado a la red en especifico, en ese caso que lo conecte.

Respuesta ChatGPT:

```
#!/bin/bash

# Nombres de las redes
RED_WEB="red_web"
RED_SERVICIOS="red_servicios"

# Funcion para obtener el ID completo de una red dado un fragmento de su nombre
obtener_id_red() {
    local fragmento_nombre=$1
    local network_name=$(docker network ls --filter name=${fragmento_nombre} --format "{{.Name}}")
    network_id=$(docker network inspect $network_name --format "{{.Id}}")
    echo "$network_id"
}

# Obtener el ID completo de la red web
NETWORK_ID_WEB=$(obtener_id_red "$RED_WEB")

# Obtener el ID completo de la red de servicios
NETWORK_ID_SERVICIOS=$(obtener_id_red "$RED_SERVICIOS")

# Funcion para obtener la IP de un contenedor en la red especificada por su ID
obtener_ip() {
    docker inspect -f '{{range .NetworkSettings.Networks}}{{if eq .NetworkID "'$1'"}}{{.IPAddress}}{{end}}{{end}}' "$2"
}

# Funcion para asignar la IP y conectar el contenedor a la red especificada
asignar_ip_y_conectar() {
    local network_id=$1
    local container_name=$2
    local desired_ip=$3

    # Verificar si el contenedor esta conectado a la red
    if [[ $(docker network inspect -f '{{.Containers}}' $network_id | grep -c $container_name) -eq 0 ]]; then
        docker network connect $network_id $container_name
    fi

    # Obtener la IP actual del contenedor
    current_ip=$(obtener_ip "$network_id" "$container_name")

    # Verificar si la IP actual no coincide con la IP deseada
    if [[ "$current_ip" != "$desired_ip" ]]; then
        echo "Ajustando IP para el contenedor $container_name a $desired_ip"
        docker network disconnect $network_id $container_name
        docker network connect --ip $desired_ip $network_id $container_name
    fi
}

echo "Verificando y ajustando direcciones IP de los contenedores:"
echo "-----"

# Iterar sobre los contenedores web
for i in {1..8}; do
    CONTAINER_NAME="web$i"
    IP_WEB="192.168.10.$((i+1))"
    asignar_ip_y_conectar "$NETWORK_ID_WEB" "$CONTAINER_NAME" "$IP_WEB"
done

# Iterar sobre los contenedores de servicios
for i in {1..8}; do
    CONTAINER_NAME="web$i"
    IP_SERVICIOS="192.168.20.$((i+1))"
    asignar_ip_y_conectar "$NETWORK_ID_SERVICIOS" "$CONTAINER_NAME" "$IP_SERVICIOS"
done

echo "Proceso completado."
```

Analisis

En este caso ChatGPT da una respuesta correcta muy rápido y con solo una pregunta ya que le he dado bastante contexto y por eso no es necesario modificar nada. El funcionamiento lo muestro en la tarea básica B4.

A4.3. Verificación de conectividad entre contenedores

Para no ser repetitivo el script solo muestra la conectividad entre el contenedor 1 y 2 de manera recíproca entre sus distintas IPs.

A4.3.1 Script final

El script final es el mismo que el que me proporciona ChatGpt.

A4.3.2 Analisis IA

Pregunta a ChatGPT:

Proporcioname un script que desde dentro del contenedor haciendo uso de docker exec, utilice ping para comprobar la conectividad desde el contenedor 1 hasta el contenedor 2, y también desde el contenedor 2 al 1 (Le paso el docker compose para que vea como se crean las IPs)

Respuesta ChatGPT:

```
#!/bin/bash

# Definir las direcciones IP de los contenedores
contenedor1_ip="192.168.10.2"
contenedor2_ip="192.168.10.3"

# Realizar ping desde el contenedor 1 al contenedor 2
echo "Ping desde el contenedor 1 al contenedor 2:"
docker exec web1 ping -c 4 $contenedor2_ip

# Realizar ping desde el contenedor 2 al contenedor 1
echo "Ping desde el contenedor 2 al contenedor 1:"
docker exec web2 ping -c 4 $contenedor1_ip
```

Análisis

Como podemos comprobar el script proporcionado es bastante completo, realiza la tarea que le pido y está bien estructurado. El funcionamiento de este script lo enseñé en la tarea básica B4.

A4.4. Automatización docker compose

A4.4.1 Script final

El script final es el mismo que me ha dado ChatGPT porque hace la función que yo le pido de manera correcta.

A4.4.2 Analisis IA

Pregunta:

Creame un script que me genere el docker compose de manera automática recibiendo un parámetro que será el número de contenedores a crear. Se deben realizar las comprobaciones necesarias a los parámetros.

Respuesta:

```
#!/bin/bash

# Verificar si se proporciona un numero de contenedores como argumento
if [ $# -ne 1 ]; then
    echo "Uso: $0 <numero_de_contenedores>"
    exit 1
fi

num_contenedores=$1

# Verificar si el numero de contenedores es un entero positivo
if ! [[ $num_contenedores =~ ^[1-9][0-9]*$ ]]; then
    echo "El numero de contenedores debe ser un entero positivo."
    exit 1
fi
```

```

fi

# Generar el contenido del docker-compose.yaml
cat > docker-compose.yaml <<EOF
version: '3.8'

services:
EOF

# Agregar cada servicio al docker-compose.yaml
for ((i = 1; i <= num_contenedores; i++)); do
    cat >> docker-compose.yaml <<EOF
    web${i}:
        image: mariorgzlpz-apache-image:pl
        volumes:
            - ./web_MarioRgzLpz:/var/www/localhost/htdocs
            - ./configsphp:/etc/php8/conf.d
        container_name: web${i}
        networks:
            red_web:
                ipv4_address: 192.168.10.$((1 + i))
            red_servicios:
                ipv4_address: 192.168.20.$((1 + i))
EOF
done

cat >> docker-compose.yaml <<EOF
networks:
    red_web:
        driver: bridge
        ipam:
            driver: default
            config:
                - subnet: 192.168.10.0/24
                  gateway: 192.168.10.1
    red_servicios:
        driver: bridge
        ipam:
            driver: default
            config:
                - subnet: 192.168.20.0/24
                  gateway: 192.168.20.1
EOF

echo "docker-compose.yaml generado con $num_contenedores contenedores."

```

A4.5. Instalación de paquetes

Necesitaba añadir paquetes a los contenedores y puesto que no iba a ser permanente su uso en lugar de modificar el DockerFile he decidido usar un script para instalarle el paquete que se pase como parametro.

A4.5.1 Script final

El script final es el mismo que me ha dado ChatGPT porque hace la función que yo le pido de manera correcta.

A4.5.2 Analisis IA

Pregunta:

Dame un script que instale en todos los contenedores webX que tengo el paquete cuyo nombre yo le paso como parametro, teniendo en cuenta que uso alpine linux y tengo bash instalado.

Respuestas:

```

#!/bin/bash

package_name="$1"

if [ -z "$package_name" ]; then
    echo "Usage: $0 <package_name>"
    exit 1

```

```

fi

# Obtener una lista de todos los contenedores web
containers=$(docker ps --filter "name=web" --format "{{.Names}}")

# Iterar sobre cada contenedor y ejecutar el comando de instalacion
for container in $containers; do
    echo "Instalando $package_name en el contenedor $container..."
    docker exec "$container" /bin/bash -c "apk update && apk add $package_name"
    if [ $? -eq 0 ]; then
        echo "Instalacion exitosa en el contenedor $container."
    else
        echo "No se pudo instalar $package_name en el contenedor $container."
    fi
done

```

A4.6. Obtención de loggins

Como en la tarea A5 nos piden que comprobemos logging, he decidido usar un script para mostrar los logs de cada contenedor. Esto se puede hacer tambien con docker compose logs pero usando el script queda más claro.

A4.6.1 Script final

El script final es el mismo que me ha dado ChatGPT porque hace la función que yo le pido de manera correcta.

A4.6.2 Analisis IA

Pregunta:

Dame un script que saque por pantalla los logs de los contenedores cuyo nombre es webX.

Respuestas:

```

#!/bin/bash

# Ruta absoluta para el archivo de salida
output_file="logcontenedores.txt"

# Obtener una lista de los contenedores que comienzan con "web"
contenedores=$(docker ps --filter "name=web" --format "{{.Names}}")

# Iterar sobre cada contenedor y mostrar los logs de dicho contenedor
for contenedor in $contenedores; do
    logs=$(docker logs -t $contenedor)
    echo "Logs para el contenedor $contenedor: $logs"
done

```

A5. MONITOREO Y LOGGING

A5.1 Configuración de las herramientas

Para obtener logging podemos hacer uso de docker logs nombre_contenedor que nos proporciona los logs del contenedor que le estamos pasando.

Para esto he empleado un script que muestra por pantalla los logs de todos los contenedores web.

Para borrar los logs de todos los contenedores empleamos el siguiente comando (extraído de stackoverflow):

```
1 sudo sh -c "truncate -s 0 /var/lib/docker/containers/*/*-json.log"
```

Otra manera de borrar los logs sería configurando el archivo daemon.json y dar una configuración específica para los logs como un tamaño máximo y que se borren cada cierto tiempo. No he realizado esto porque habría que crear de nuevo los contenedores para que se aplique dicha configuración ademas de reiniciar el servicio de docker.

Para monitoreo he decidido instalar netstat y hacer uso de las funcionalidades que tiene como las conexiones activas, puertos... Lo he instalado usando la script que he creado y que me permite instalar paquetes en todos los contenedores sin necesidad de ir uno a uno o de rehacer la imagen del DockerFile.

A5.2 Uso de las herramientas

Logs

He comenzado mostrando los logs de mis contenedores, viendo que al crearlos una vez por no tener un ServerName en el fichero de configuración de apache nos saltará que el nombre puesto es la IP en la red_servicios(si se le ha conseguido poner una IP). Además también se ve que al relanzar los contenedores otra vez con docker compose sucede lo mismo y vuelve a almacenar la misma advertencia. Yo no modifico el ServerName porque así se puede comprobar si ha habido algún problema en la asignación de IPs y ya poder usar las scripts asociadas a arreglar las IPs. Aquí también podríamos ver si ha habido algún problema con instalación de paquetes en el contenedor.

Logs de todos los contenedores después de lanzarlos dos veces.

```
amb_uti_prompt_command_hook - Parrot Terminal
barrot: ~/.../P1/scripts
./obtener_logs.sh
2024-04-08T08:22:17.499891744Z AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 192.168.20.5. Set the 'ServerName' directive globally to suppress this message
2024-04-08T08:22:22.816217673Z AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 192.168.20.5. Set the 'ServerName' directive globally to suppress this message
Logs para el contenedor web4:
2024-04-07T16:02:17.755001479Z AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 192.168.20.6. Set the 'ServerName' directive globally to suppress this message
2024-04-08T08:22:22.665731036Z AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 192.168.20.6. Set the 'ServerName' directive globally to suppress this message
Logs para el contenedor web5:
2024-04-07T16:02:17.639057240Z AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 192.168.20.2. Set the 'ServerName' directive globally to suppress this message
2024-04-08T08:22:22.769215352Z AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 192.168.20.2. Set the 'ServerName' directive globally to suppress this message
Logs para el contenedor web1:
2024-04-07T16:02:17.374787535Z AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 192.168.20.9. Set the 'ServerName' directive globally to suppress this message
2024-04-08T08:22:22.166935980Z AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 192.168.20.9. Set the 'ServerName' directive globally to suppress this message
Logs para el contenedor web8:
2024-04-07T16:02:17.744032291Z AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 192.168.20.3. Set the 'ServerName' directive globally to suppress this message
2024-04-08T08:22:21.775031803Z AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 192.168.20.3. Set the 'ServerName' directive globally to suppress this message
Logs para el contenedor web2:
2024-04-07T16:02:16.925022373Z AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 192.168.20.7. Set the 'ServerName' directive globally to suppress this message
2024-04-08T08:22:22.605582178Z AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 192.168.20.7. Set the 'ServerName' directive globally to suppress this message
Logs para el contenedor web6:
2024-04-07T16:02:16.239295003Z AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 192.168.10.4. Set the 'ServerName' directive globally to suppress this message
2024-04-08T08:22:22.606999280Z AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 192.168.20.4. Set the 'ServerName' directive globally to suppress this message
Logs para el contenedor web3:
2024-04-07T16:02:16.875165959Z AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 192.168.20.8. Set the 'ServerName' directive globally to suppress this message
2024-04-08T08:22:21.773508912Z AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 192.168.20.8. Set the 'ServerName' directive globally to suppress this message
```


Logs de los contenedores después de borrar

```
parrot: ~/.../P1/scripts
$ sudo sh -c "truncate -s 0 /var/lib/docker/containers/*/*-json.log"
[sudo] password for mariorl:
parrot: ~/.../P1/scripts
$ ./obtener_logs.sh
Logs para el contenedor web4:
Logs para el contenedor web5:
Logs para el contenedor web1:
Logs para el contenedor web8:
Logs para el contenedor web2:
Logs para el contenedor web6:
Logs para el contenedor web3:
Logs para el contenedor web7:
```

Monitoring

Para netstat simplemente he ejecutado los comandos básicos dentro de un contenedor para ver las salidas que me da. Aquí muestro la instalación, que es exitosa en los 8 contenedores.

```
parrot: ~/.../P1/scripts
$ ./instalar_paquete.sh net-tools
Instalando net-tools en el contenedor web4...
fetch https://dl-cdn.alpinelinux.org/alpine/v3.15/main/x86_64/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.15/community/x86_64/APKINDEX.tar.gz
v3.15.11-6-g2ae4ac5ddb8 [https://dl-cdn.alpinelinux.org/alpine/v3.15/main]
v3.15.10-165-g73b5f3b6fbc [https://dl-cdn.alpinelinux.org/alpine/v3.15/community]
OK: 15866 distinct packages available
OK: 48 MiB in 34 packages
Instalación exitosa en el contenedor web4...
Instalando net-tools en el contenedor web5...
fetch https://dl-cdn.alpinelinux.org/alpine/v3.15/main/x86_64/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.15/community/x86_64/APKINDEX.tar.gz
v3.15.11-6-g2ae4ac5ddb8 [https://dl-cdn.alpinelinux.org/alpine/v3.15/main]
v3.15.10-165-g73b5f3b6fbc [https://dl-cdn.alpinelinux.org/alpine/v3.15/community]
OK: 15866 distinct packages available
OK: 48 MiB in 34 packages
Instalación exitosa en el contenedor web5...
Instalando net-tools en el contenedor web1...
fetch https://dl-cdn.alpinelinux.org/alpine/v3.15/main/x86_64/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.15/community/x86_64/APKINDEX.tar.gz
v3.15.11-6-g2ae4ac5ddb8 [https://dl-cdn.alpinelinux.org/alpine/v3.15/main]
v3.15.10-165-g73b5f3b6fbc [https://dl-cdn.alpinelinux.org/alpine/v3.15/community]
OK: 15866 distinct packages available
OK: 48 MiB in 34 packages
Instalación exitosa en el contenedor web1.
```

Ejecuto una terminal dentro del contenedor web1 con

```
1 docker exec -it web1 bash
```

Y ya muestro los resultados de ejecutar varios comandos con netstat:

```
parrot: ~/.../P1/scripts
$ docker exec -it web1 bash
bash-5.1# netstat --help
usage: netstat [-vWenNcF] [-A<af>] -r netstat {-V|--version|-h|--help}
netstat [-vWenNcF] [-A<af>] [-i] [-cN] [-M] [-S] [-t] [-u] [-w] [-x] [-y] [-z]
netstat [-vWenNcF] [-A<af>] [-i] [-cN] [-M] [-S] [-t] [-u] [-w] [-x] [-y] [-z]

-r, --route display routing table
-i, --interfaces display interface table
-g, --groups display multicast group memberships
-s, --statistics display networking statistics (like SNMP)
-M, --masquerade display masqueraded connections

-v, --verbose be verbose
-W, --wide don't truncate IP addresses
-n, --numeric don't resolve names
--numeric-hosts don't resolve host names
--numeric-ports don't resolve port names
--numeric-users don't resolve user names
-N, --symbolic resolve hardware names
-e, --extend display other/more information
-p, --programs display PID/Program name for sockets
-o, --timers display timers
-c, --continuous continuous listing

-l, --listening display listening server sockets
-a, --all display all sockets (default: connected)
-F, --fib display Forwarding Information Base (default)
-C, --cache display routing cache instead of FIB

<Socket>={-t|--tcp} {-u|--udp} {-U|--udplite} {-S|--sctp} {-w|--raw}
{-x|--unix} --ax25 --ipx --netrom
-AF=Use '-6|-4' or '-A <af>' or '--<af>'; default: inet
List of possible address families (which support routing):
inet (DARPA Internet) inet6 (IPv6) netrom (AMPR NET/ROM)
```

```

bash-5.1# netstat -a
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.11:39213        0.0.0.0:*               LISTEN
tcp6       0      0 [::]:80                [::]:*                  LISTEN
udp        0      0 127.0.0.11:42402       0.0.0.0:*               LISTEN
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State      I-Node  Path
bash-5.1# netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags       MSS  Window  irtt  Iface
default         192.168.20.1   0.0.0.0        UG          0 0          0     eth0
192.168.10.0    0.0.0.0        255.255.255.0  U           0 0          0     eth1
192.168.20.0    0.0.0.0        255.255.255.0  U           0 0          0     eth0
bash-5.1# netstat -i
Kernel Interface table
Iface    MTU     RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0     1500    0      0      0      0        0      0      0      0    BMRU
eth1     1500    0      0      0      0        0      0      0      0    BMRU
lo       65536   0      0      0      0        0      0      0      0    LRU

```

```

bash-5.1# netstat -s
ip:
1178 total packets received
0 forwarded
0 incoming packets discarded
896 incoming packets delivered
615 requests sent out
icmp:
0 ICMP messages received
0 input ICMP message failed.
ICMP input histogram:
0 ICMP messages sent
0 ICMP messages failed
ICMP output histogram:
Tcp:
1 active connections openings
0 passive connection openings
0 failed connection attempts
0 connection resets received
0 connections established
869 segments received
588 segments send out
0 segments retransmitted
0 bad segments received.
3 resets sent
Udp:
27 packets received
0 packets to unknown port received.
0 packet receive errors
27 packets sent
0 receive buffer errors
0 send buffer errors
Udplite:
Problem while parsing /proc/net/netstat
bash-5.1#

```

Comandos usados:

- netstat -s: Recupera estadísticas sobre los protocolos de red importantes como TCP, IP o UDP.
- netstat -a: Muestra todos los puertos activos.
- netstat -i: Muestra el menú de resumen de netstat.
- netstat -r: Muestra la tabla de enrutamiento IP.

A5.1 Análisis IA

La IA en este apartado esta reflejada en la creación de los scripts que para evitar ser repetitivo aparece solo en el apartado A4 y aqui muestro solo lo que proporciona cada script que he usado.