



UNIVERSIDAD DE GRANADA

Servidores Web de Altas Prestaciones

Práctica 1: Servidores Web y Almacenamiento

Curso 2023-2024
Grado en Ingeniería Informática
Alumno: Miguel Torres Alonso

Índice

Tareas Básicas - B1. Configuración del entorno.....	4
Instalar Docker y Docker Compose si aún no están instalados.....	4
Análisis propuesta IA.....	4
Consulta realizada:.....	4
Resultado proporcionado por IA:.....	4
Análisis detallado: [OBJ].....	4
Crear el directorio web_mtorres26 y el archivo index.php.....	5
Tareas Básicas - B2. Creación del Dockerfile.....	6
Escribir un Dockerfile DockerfileApache_mtorres26 que use una imagen base de Linux, e instale Apache, PHP y herramientas de red.....	6
Análisis propuesta IA.....	6
Consulta realizada:.....	6
Resultado proporcionado por IA:.....	7
Análisis detallado:.....	7
Tareas Básicas - B3. Uso de Docker Compose.....	8
Escribir un archivo docker-compose.yml que defina la construcción de la imagen mtorres26-apache-image:p1 y la creación de los 8 contenedores webX.....	8
Análisis propuesta IA.....	8
Consulta realizada:.....	8
Resultado proporcionado por IA:.....	9
Análisis detallado:.....	9
Tareas Básicas - B4. Despliegue y verificación de Contenedores.....	12
Ejecutar docker-compose up para iniciar los contenedores.....	12
Usar docker ps para asegurarse de que todos los contenedores están en ejecución.....	13
Verificar que cada contenedor tiene una dirección IP asignada en las redes red_web y red_servicios.....	13
Análisis propuesta IA.....	13
Consulta realizada:.....	13
Resultado proporcionado por IA:.....	13
Análisis detallado:.....	14
Web1.....	14
Web4.....	17
Web8.....	19
Tareas Básicas - B5. Pruebas Básicas.....	21
Acceder a la página web de cada contenedor usando su dirección IP y verificar que muestra la información correcta.....	21
Web1.....	21
Web2.....	22
Web3.....	22
Web4.....	22
Web5.....	23
Web6.....	23
Web7.....	24

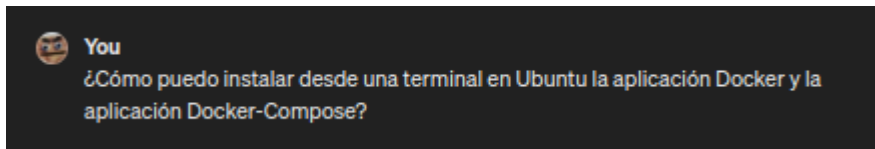
Web8.....	24
Tareas Avanzadas - A1. Personalización del Dockerfile.....	25
Modificar el Dockerfile para incluir configuraciones personalizadas de Apache o PHP..	25
Análisis propuesta IA.....	25
Consulta realizada:.....	25
Resultado proporcionado por IA:.....	25
Análisis detallado:.....	26
Tareas Avanzadas - A2. Creación de contenedores con otros servidores web.....	28
Crear contenedores con otros servidores web (nginx, lighttpd, etc.).....	28
Análisis propuesta IA.....	28
Consulta realizada:.....	28
Resultado proporcionado por IA:.....	28
Análisis detallado:.....	29
Consulta realizada:.....	30
Resultado proporcionado por IA:.....	30
Análisis detallado:.....	31

Tareas Básicas - B1. Configuración del entorno

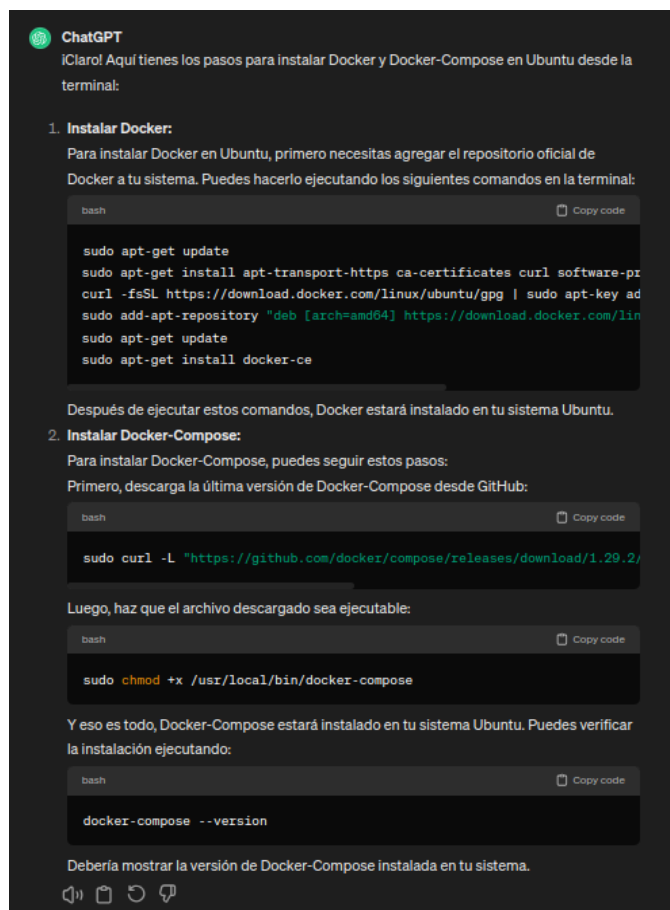
Instalar Docker y Docker Compose si aún no están instalados.

Análisis propuesta IA

Consulta realizada:



Resultado proporcionado por IA:



Análisis detallado:

La IA nos proporciona comandos algo distintos a los proporcionados por la [documentación oficial](#) de Docker.

Seguimos los pasos de la documentación oficial, que son los siguientes:

Unset

```
$ sudo apt-get update
$ sudo apt-get install ca-certificates curl
$ sudo install -m 0755 -d /etc/apt/keyrings
$ sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o
/etc/apt/keyrings/docker.asc
$ sudo chmod a+r /etc/apt/keyrings/docker.asc
```

Si todo ha ido bien ejecutamos los comandos siguientes:

Unset

```
$ echo "deb [arch=$(dpkg --print-architecture)
signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu $(. /etc/os-release && echo
"$VERSION_CODENAME") stable" | sudo tee /etc/apt/sources.list.d/docker.list
> /dev/null
$ sudo apt-get update
```

Por último, instalamos la última versión de los paquetes siguientes:

Unset

```
$ sudo apt-get install docker-ce docker-ce-cli containerd.io
docker-buildx-plugin docker-compose-plugin
```

Y ya tendríamos instalado Docker Engine y el plugin de Docker Compose. Hay que tener cuidado con las dos versiones de Docker Compose, la del plugin y la Standalone. Nosotros usaremos la del plugin que se llama con *'docker compose'* mientras que la otra se llama con *'docker-compose'*

Crear el directorio **web_mtorres26** y el archivo **index.php**

La creación del directorio es trivial, por tanto no necesitamos la IA generativa y utilizamos el comando básico:

Unset

```
mkdir web_mtorres26
```

Dentro de este directorio creamos el archivo "index.php" con la estructura que se nos pide: que muestre SWAP - mtorres26 y la dirección IP del servidor Apache.

Quedaría algo así:

```
<!DOCTYPE html>
<html lang="es">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>SWAP-mtorres26</title>
</head>

<body>
  <?php
    echo "<h1>SWAP - mtorres26</h1>";
    echo "<p>La dirección IP del servidor Apache es: ",
$_SERVER['SERVER_ADDR'], "</p>";
  ?>
</body>

</html>
```

Tareas Básicas - B2. Creación del Dockerfile

Escribir un Dockerfile **DockerfileApache_mtorres26** que use una imagen base de Linux, e instale Apache, PHP y herramientas de red.

Análisis propuesta IA

Consulta realizada:



You

Escribe el código necesario para crear un Dockerfile que use una imagen base de Linux e instale Apache, PHP y herramientas de red

Resultado proporcionado por IA:

 **ChatGPT**

Aquí tienes un ejemplo básico de un Dockerfile que utiliza una imagen base de Linux (en este caso, Ubuntu), instala Apache, PHP y algunas herramientas de red como curl e iputils:

```
Dockerfile
```

```
# Usa una imagen base de Ubuntu
FROM ubuntu:latest

# Actualiza el repositorio e instala Apache, PHP y herramientas de red
RUN apt-get update && \
    apt-get install -y apache2 php libapache2-mod-php curl iputils-ping && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*

# Expone el puerto 80 para Apache
EXPOSE 80

# Inicia Apache en el inicio del contenedor
CMD ["apachectl", "-D", "FOREGROUND"]
```

Copy code

Para construir la imagen Docker, coloca este código en un archivo llamado **Dockerfile** en un directorio y ejecuta el siguiente comando en la terminal:

```
bash
```

```
docker build -t mi_aplicacion_web .
```

Copy code

Este comando construirá una imagen Docker etiquetada como "mi_aplicacion_web". Luego, para ejecutar un contenedor basado en esta imagen, puedes usar el siguiente comando:

```
bash
```

```
docker run -d -p 8080:80 mi_aplicacion_web
```

Copy code

Este comando ejecutará un contenedor en segundo plano, mapeando el puerto 8080 de tu máquina local al puerto 80 del contenedor. Puedes acceder a tu aplicación web abriendo un navegador y visitando **http://localhost:8080**.

Análisis detallado:

La IA nos proporciona un Dockerfile con bastantes cosas necesarias para este apartado, pero vamos a utilizar una versión de Debian más ligera mediante el tag 'slim'. La construcción de la imagen la vamos a llevar a cabo en el próximo apartado con el docker-compose.yml.

Dockerfile:

```
Unset
# Usa una imagen base de Debian con una versión ligera
FROM debian:buster-slim

# Actualiza el repositorio e instala Apache, PHP y herramientas de red
RUN apt-get update && \
    apt-get install -y apache2 php libapache2-mod-php net-tools \
    iputils-ping && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*

# Expone el puerto 80 para Apache
EXPOSE 80

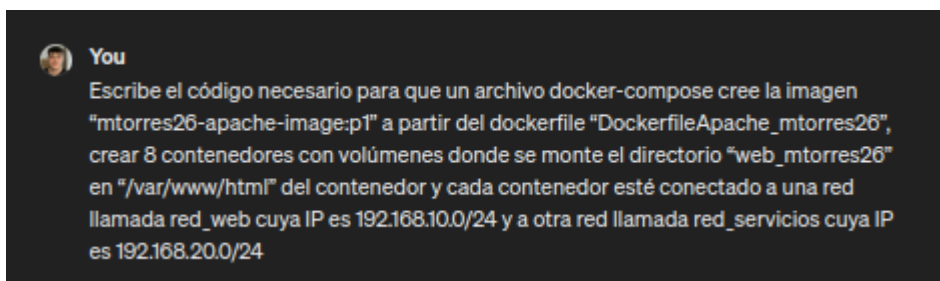
# Inicia Apache en el inicio del contenedor
CMD ["apachectl", "-D", "FOREGROUND"]
```

Tareas Básicas - B3. Uso de Docker Compose


Escribir un archivo **docker-compose.yml** que defina la construcción de la imagen **mtorres26-apache-image:p1** y la creación de los 8 contenedores **webX**.

Análisis propuesta IA

Consulta realizada:



Resultado proporcionado por IA:

 ChatGPT

Para lograr lo que has solicitado, necesitarás un Dockerfile específico llamado "DockerfileApache_mtorres26" para construir la imagen "mtorres26-apache-image:p1" y luego utilizarlo en el archivo "docker-compose.yml".

Primero, asegúrate de tener el archivo "DockerfileApache_mtorres26" en tu directorio de trabajo. Luego, puedes crear el archivo "docker-compose.yml" con el siguiente contenido:

```
yaml
version: '3'

services:
  apache:
    build:
      context: .
      dockerfile: DockerfileApache_mtorres26
    volumes:
      - ./web_mtorres26:/var/www/html
    networks:
      red_web:
        ipv4_address: 192.168.10.2
      red_servicios:
        ipv4_address: 192.168.20.2
    deploy:
      replicas: 8

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

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

En este archivo "docker-compose.yml", estamos construyendo la imagen "mtorres26-apache-image:p1" utilizando el Dockerfile "DockerfileApache_mtorres26" que está presente en el contexto actual (el mismo directorio donde se encuentra el archivo "docker-compose.yml"). Luego, se definen los volúmenes y las redes como lo has especificado en tu pregunta.

Análisis detallado:

- La IA nos da bastantes cosas bien salvo algunos aspectos, como por ejemplo la propiedad 'image' la cual sería 'mtorres26-apache-image:p1' y se creará mediante el Dockerfile del apartado anterior ('docker compose build'), sólo se indicará que se cree en el contenedor web1 pero los demás contenedores también podrán usarla.
- El montaje del volumen es correcto.
- Los contenedores 'webX' con X del 1 al 8 los vamos a crear manualmente introduciendo sus direcciones IP con respecto de la red_web y red_servicios manualmente.
- Elimino la opción 'driver:default' dentro del ipam de las redes, pues no es necesario.

- Añado la propiedad `container_name` a cada contenedor para ponerle el nombre que se nos indica en vez del que Docker usa como predeterminado (`p1-web1-1...` etc).

Nuestro fichero `docker-compose.yml` quedaría así:

```
Unset
services:
  web1:
    image: mtorres26-apache-image:p1
    container_name: web1
    build:
      context: .
      dockerfile: DockerfileApache_mtorres26
    volumes:
      - ./web_mtorres26:/var/www/html
    networks:
      red_web:
        ipv4_address: 192.168.10.2
      red_servicios:
        ipv4_address: 192.168.20.2

  web2:
    image: mtorres26-apache-image:p1
    container_name: web2
    volumes:
      - ./web_mtorres26:/var/www/html
    networks:
      red_web:
        ipv4_address: 192.168.10.3
      red_servicios:
        ipv4_address: 192.168.20.3

  web3:
    image: mtorres26-apache-image:p1
    container_name: web3
    volumes:
      - ./web_mtorres26:/var/www/html
    networks:
      red_web:
        ipv4_address: 192.168.10.4
      red_servicios:
        ipv4_address: 192.168.20.4

  web4:
    image: mtorres26-apache-image:p1
    container_name: web4
    volumes:
      - ./web_mtorres26:/var/www/html
```

```
networks:
  red_web:
    ipv4_address: 192.168.10.5
  red_servicios:
    ipv4_address: 192.168.20.5

web5:
  image: mtorres26-apache-image:p1
  container_name: web5
  volumes:
    - ./web_mtorres26:/var/www/html
  networks:
    red_web:
      ipv4_address: 192.168.10.6
    red_servicios:
      ipv4_address: 192.168.20.6

web6:
  image: mtorres26-apache-image:p1
  container_name: web6
  volumes:
    - ./web_mtorres26:/var/www/html
  networks:
    red_web:
      ipv4_address: 192.168.10.7
    red_servicios:
      ipv4_address: 192.168.20.7

web7:
  image: mtorres26-apache-image:p1
  container_name: web7
  volumes:
    - ./web_mtorres26:/var/www/html
  networks:
    red_web:
      ipv4_address: 192.168.10.8
    red_servicios:
      ipv4_address: 192.168.20.8

web8:
  image: mtorres26-apache-image:p1
  container_name: web8
  volumes:
    - ./web_mtorres26:/var/www/html
  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

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

```

Tareas Básicas - B4. Despliegue y verificación de Contenedores

Ejecutar **docker-compose up** para iniciar los contenedores.

Primero construimos la imagen que hemos especificado en el docker-compose.yml:

```

mtorres@mtorres:~/UGR_NATIVO/CuartoCurso/SegundoCuatri/SWAP/P1$ docker compose build
[+] Building 71.0s (10/10) FINISHED                                docker:default
=> [web1 internal] load build definition from DockerfileApache_mtorres26
=> transferring dockerfile: 745B
=> [web1 internal] load metadata for docker.io/library/debian:buster-slim
=> [web1 internal] load .dockerignore
=> transferring context: 2B
=> [web1 1/5] FROM docker.io/library/debian:buster-slim@sha256:4620251e86cea652ef2746c8abef2fcc3ea54abb5f014bfbf841159d805cd1fd
=> resolve docker.io/library/debian:buster-slim@sha256:4620251e86cea652ef2746c8abef2fcc3ea54abb5f014bfbf841159d805cd1fd
=> sha256:4620251e86cea652ef2746c8abef2fcc3ea54abb5f014bfbf841159d805cd1fd 984B / 984B
=> sha256:886b6d8710aa6fe54881beef0dc76af5966848638ddba94efdf1fe9aa3d2924c 529B / 529B
=> sha256:0b031cf4038f92451c8dbdaf26901cb6f66fc3774cbd454e83262f74bebb38cd 1.46kB / 1.46kB
=> sha256:4a21b857313366ff6d22f19e31c4300a81dce92b023b4ff6583ca167179804ac 27.19MB / 27.19MB
=> extracting sha256:4a21b857313366ff6d22f19e31c4300a81dce92b023b4ff6583ca167179804ac
=> [web1 internal] load build context
=> transferring context: 80.22kB
=> [web1 2/5] RUN apt-get update && apt-get install -y apache2 php libapache2-mod-php net-tools iputils-ping && apt-get clean &&
=> [web1 3/5] COPY apache2.conf /etc/apache2/apache2.conf
=> [web1 4/5] COPY php.ini /etc/php/7.3/apache2/php.ini
=> [web1 5/5] RUN a2enmod autoindex
=> [web1] exporting to image
=> exporting layers
=> writing image sha256:2da38afbd681e36ed0120e2ec56ebd54f365727e57bc3814b910f12d199fb16b
=> naming to docker.io/library/mtorres26-apache-image:p1

```

Ahora iniciamos contenedores:

```

mtorres@mtorres:~/UGR_NATIVO/CuartoCurso/SegundoCuatri/SWAP/P1$ docker compose up -d
[+] Running 8/10
! Network p1_red_servicios Created                                2.7s
! Network p1_red_web Created                                     2.6s
! Container web1 Started                                         2.3s
! Container web2 Started                                         1.7s
! Container web5 Started                                         2.3s
! Container web7 Started                                         2.5s
! Container web8 Started                                         2.0s
! Container web6 Started                                         2.3s
! Container web4 Started                                         1.8s
! Container web3 Started                                         1.8s

```

Los contenedores se han iniciado correctamente.

Usar **docker ps** para asegurarse de que todos los contenedores están en ejecución

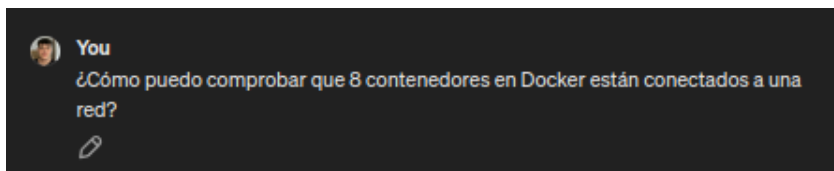
```
mtorres@mtorres:~/UGR_NATIVO/CuartoCurso/SegundoCuatrI/5WAP/P1$ docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED        STATUS        PORTS          NAMES
fcf9400cba7a   mtorres26-apache-image:p1  "apachectl -D FOREGR..." 19 seconds ago Up 17 seconds 80/tcp         web3
8a6ec97a2910   mtorres26-apache-image:p1  "apachectl -D FOREGR..." 19 seconds ago Up 17 seconds 80/tcp         web4
96f234f61dd9   mtorres26-apache-image:p1  "apachectl -D FOREGR..." 19 seconds ago Up 16 seconds 80/tcp         web7
b2fd7936ed93   mtorres26-apache-image:p1  "apachectl -D FOREGR..." 19 seconds ago Up 16 seconds 80/tcp         web8
a3720deed76a   mtorres26-apache-image:p1  "apachectl -D FOREGR..." 19 seconds ago Up 17 seconds 80/tcp         web2
0f16420bc06b   mtorres26-apache-image:p1  "apachectl -D FOREGR..." 19 seconds ago Up 16 seconds 80/tcp         web5
f9f46628f6d5   mtorres26-apache-image:p1  "apachectl -D FOREGR..." 19 seconds ago Up 16 seconds 80/tcp         web1
c37fea5b69d7   mtorres26-apache-image:p1  "apachectl -D FOREGR..." 19 seconds ago Up 16 seconds 80/tcp         web6
```

Efectivamente están en ejecución.

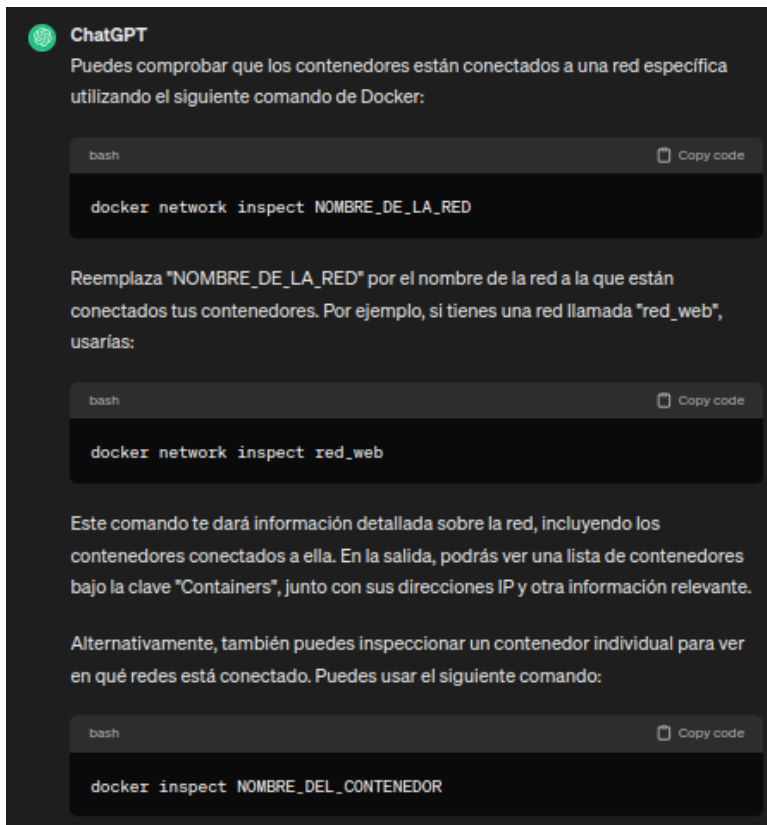
Verificar que cada contenedor tiene una dirección IP asignada en las redes **red_web** y **red_servicios**

Análisis propuesta IA

Consulta realizada:



Resultado proporcionado por IA:



Análisis detallado:

De la manera que nos propone la IA generativa se puede hacer la comprobación de las redes a las que está conectado cada contenedor, sin embargo voy a ejecutar varios contenedores en el terminal y ejecutar los siguientes comandos:

```
Unset
$ ifconfig
$ ping 192.168.10.X
$ ping 192.168.20.X
```

Donde X = un número del 2 al 9.

De esta manera comprobamos la configuración de las redes y la conectividad con los demás contenedores manualmente.

Adjunto capturas de pantalla de la comprobación en algunos contenedores:

Web1

```
mtorres@mtorres:~/UGR_NATIVO/CuartoCurso/SegundoCuatri/SWAP/P1$ docker compose exec web1 /bin/bash
root@60710cfaabc5:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.20.2 netmask 255.255.255.0 broadcast 192.168.20.255
    inet6 fe80::42:c0ff:fea8:1402 prefixlen 64 scopeid 0x20<link>
    ether 02:42:c0:a8:14:02 txqueuelen 0 (Ethernet)
    RX packets 76 bytes 13167 (12.8 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 8 bytes 656 (656.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.10.2 netmask 255.255.255.0 broadcast 192.168.10.255
    inet6 fe80::42:c0ff:fea8:a02 prefixlen 64 scopeid 0x20<link>
    ether 02:42:c0:a8:0a:02 txqueuelen 0 (Ethernet)
    RX packets 72 bytes 12490 (12.1 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 8 bytes 656 (656.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

```
root@60710cfaabc5:/# ping 192.168.10.3
PING 192.168.10.3 (192.168.10.3) 56(84) bytes of data.
64 bytes from 192.168.10.3: icmp_seq=1 ttl=64 time=0.190 ms
^C
--- 192.168.10.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.190/0.190/0.190/0.000 ms
root@60710cfaabc5:/# ping 192.168.10.4
PING 192.168.10.4 (192.168.10.4) 56(84) bytes of data.
64 bytes from 192.168.10.4: icmp_seq=1 ttl=64 time=0.082 ms
^C
--- 192.168.10.4 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.082/0.082/0.082/0.000 ms
root@60710cfaabc5:/# ping 192.168.10.5
PING 192.168.10.5 (192.168.10.5) 56(84) bytes of data.
64 bytes from 192.168.10.5: icmp_seq=1 ttl=64 time=0.064 ms
^C
--- 192.168.10.5 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.064/0.064/0.064/0.000 ms
root@60710cfaabc5:/# ping 192.168.10.6
PING 192.168.10.6 (192.168.10.6) 56(84) bytes of data.
64 bytes from 192.168.10.6: icmp_seq=1 ttl=64 time=0.161 ms
^C
--- 192.168.10.6 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.161/0.161/0.161/0.000 ms
```

```
root@60710cfaabc5:/# ping 192.168.10.7
PING 192.168.10.7 (192.168.10.7) 56(84) bytes of data.
64 bytes from 192.168.10.7: icmp_seq=1 ttl=64 time=0.064 ms
^C
--- 192.168.10.7 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.064/0.064/0.064/0.000 ms
root@60710cfaabc5:/# ping 192.168.10.8
PING 192.168.10.8 (192.168.10.8) 56(84) bytes of data.
64 bytes from 192.168.10.8: icmp_seq=1 ttl=64 time=0.075 ms
^C
--- 192.168.10.8 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.075/0.075/0.075/0.000 ms
root@60710cfaabc5:/# ping 192.168.10.9
PING 192.168.10.9 (192.168.10.9) 56(84) bytes of data.
64 bytes from 192.168.10.9: icmp_seq=1 ttl=64 time=0.074 ms
^C
--- 192.168.10.9 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.074/0.074/0.074/0.000 ms
```

```
root@60710cfaabc5:/# ping 192.168.20.3
PING 192.168.20.3 (192.168.20.3) 56(84) bytes of data.
64 bytes from 192.168.20.3: icmp_seq=1 ttl=64 time=0.152 ms
^C
--- 192.168.20.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.152/0.152/0.152/0.000 ms
root@60710cfaabc5:/# ping 192.168.20.4
PING 192.168.20.4 (192.168.20.4) 56(84) bytes of data.
64 bytes from 192.168.20.4: icmp_seq=1 ttl=64 time=0.127 ms
^C
--- 192.168.20.4 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.127/0.127/0.127/0.000 ms
root@60710cfaabc5:/# ping 192.168.20.5
PING 192.168.20.5 (192.168.20.5) 56(84) bytes of data.
64 bytes from 192.168.20.5: icmp_seq=1 ttl=64 time=0.298 ms
^C
--- 192.168.20.5 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.298/0.298/0.298/0.000 ms
root@60710cfaabc5:/# ping 192.168.20.6
PING 192.168.20.6 (192.168.20.6) 56(84) bytes of data.
64 bytes from 192.168.20.6: icmp_seq=1 ttl=64 time=0.111 ms
^C
--- 192.168.20.6 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.111/0.111/0.111/0.000 ms
```

```
root@60710cfaabc5:/# ping 192.168.20.7
PING 192.168.20.7 (192.168.20.7) 56(84) bytes of data.
64 bytes from 192.168.20.7: icmp_seq=1 ttl=64 time=0.148 ms
^C
--- 192.168.20.7 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.148/0.148/0.148/0.000 ms
root@60710cfaabc5:/# ping 192.168.20.8
PING 192.168.20.8 (192.168.20.8) 56(84) bytes of data.
64 bytes from 192.168.20.8: icmp_seq=1 ttl=64 time=0.112 ms
^C
--- 192.168.20.8 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.112/0.112/0.112/0.000 ms
root@60710cfaabc5:/# ping 192.168.20.9
PING 192.168.20.9 (192.168.20.9) 56(84) bytes of data.
64 bytes from 192.168.20.9: icmp_seq=1 ttl=64 time=0.143 ms
^C
--- 192.168.20.9 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.143/0.143/0.143/0.000 ms
```


Web8

```
mtorres@mtorres:~/UGR_NATIVO/CuartoCurso/SegundoCuatrI/SWAP/P1$ docker compose exec web8 /bin/bash
root@b195e7feb79a:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.20.9 netmask 255.255.255.0 broadcast 192.168.20.255
    inet6 fe80::42:c0ff:fea8:1409 prefixlen 64 scopeid 0x20<link>
    ether 02:42:c0:a8:14:09 txqueuelen 0 (Ethernet)
    RX packets 211 bytes 29449 (28.7 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 25 bytes 1734 (1.6 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.10.9 netmask 255.255.255.0 broadcast 192.168.10.255
    inet6 fe80::42:c0ff:fea8:a09 prefixlen 64 scopeid 0x20<link>
    ether 02:42:c0:a8:0a:09 txqueuelen 0 (Ethernet)
    RX packets 206 bytes 28546 (27.8 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 25 bytes 1734 (1.6 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

```
root@b195e7feb79a:/# ping -c 1 192.168.10.2
PING 192.168.10.2 (192.168.10.2) 56(84) bytes of data.
64 bytes from 192.168.10.2: icmp_seq=1 ttl=64 time=0.210 ms

--- 192.168.10.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.210/0.210/0.210/0.000 ms
root@b195e7feb79a:/#
root@b195e7feb79a:/#
root@b195e7feb79a:/# ping -c 1 192.168.10.3
PING 192.168.10.3 (192.168.10.3) 56(84) bytes of data.
64 bytes from 192.168.10.3: icmp_seq=1 ttl=64 time=0.190 ms

--- 192.168.10.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.190/0.190/0.190/0.000 ms
root@b195e7feb79a:/#
root@b195e7feb79a:/#
root@b195e7feb79a:/# ping -c 1 192.168.10.4
PING 192.168.10.4 (192.168.10.4) 56(84) bytes of data.
64 bytes from 192.168.10.4: icmp_seq=1 ttl=64 time=0.172 ms

--- 192.168.10.4 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.172/0.172/0.172/0.000 ms
root@b195e7feb79a:/#
root@b195e7feb79a:/#
root@b195e7feb79a:/# ping -c 1 192.168.10.5
PING 192.168.10.5 (192.168.10.5) 56(84) bytes of data.
64 bytes from 192.168.10.5: icmp_seq=1 ttl=64 time=0.131 ms

--- 192.168.10.5 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.131/0.131/0.131/0.000 ms
```

```
root@b195e7feb79a:/# ping -c 1 192.168.10.6
PING 192.168.10.6 (192.168.10.6) 56(84) bytes of data.
64 bytes from 192.168.10.6: icmp_seq=1 ttl=64 time=0.175 ms

--- 192.168.10.6 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.175/0.175/0.175/0.000 ms
root@b195e7feb79a:/#
root@b195e7feb79a:/#
root@b195e7feb79a:/# ping -c 1 192.168.10.7
PING 192.168.10.7 (192.168.10.7) 56(84) bytes of data.
64 bytes from 192.168.10.7: icmp_seq=1 ttl=64 time=0.144 ms

--- 192.168.10.7 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.144/0.144/0.144/0.000 ms
root@b195e7feb79a:/#
root@b195e7feb79a:/#
root@b195e7feb79a:/# ping -c 1 192.168.10.8
PING 192.168.10.8 (192.168.10.8) 56(84) bytes of data.
64 bytes from 192.168.10.8: icmp_seq=1 ttl=64 time=0.086 ms

--- 192.168.10.8 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.086/0.086/0.086/0.000 ms
```

```
root@b195e7feb79a:/# ping -c 1 192.168.20.2
PING 192.168.20.2 (192.168.20.2) 56(84) bytes of data.
64 bytes from 192.168.20.2: icmp_seq=1 ttl=64 time=0.156 ms

--- 192.168.20.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.156/0.156/0.156/0.000 ms
root@b195e7feb79a:/#
root@b195e7feb79a:/#
root@b195e7feb79a:/# ping -c 1 192.168.20.3
PING 192.168.20.3 (192.168.20.3) 56(84) bytes of data.
64 bytes from 192.168.20.3: icmp_seq=1 ttl=64 time=0.306 ms

--- 192.168.20.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.306/0.306/0.306/0.000 ms
root@b195e7feb79a:/#
root@b195e7feb79a:/#
root@b195e7feb79a:/# ping -c 1 192.168.20.4
PING 192.168.20.4 (192.168.20.4) 56(84) bytes of data.
64 bytes from 192.168.20.4: icmp_seq=1 ttl=64 time=0.176 ms

--- 192.168.20.4 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.176/0.176/0.176/0.000 ms
root@b195e7feb79a:/#
root@b195e7feb79a:/#
root@b195e7feb79a:/# ping -c 1 192.168.20.5
PING 192.168.20.5 (192.168.20.5) 56(84) bytes of data.
64 bytes from 192.168.20.5: icmp_seq=1 ttl=64 time=0.101 ms

--- 192.168.20.5 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.101/0.101/0.101/0.000 ms
```

```
root@b195e7feb79a:/# ping -c 1 192.168.20.6
PING 192.168.20.6 (192.168.20.6) 56(84) bytes of data.
64 bytes from 192.168.20.6: icmp_seq=1 ttl=64 time=0.179 ms

--- 192.168.20.6 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.179/0.179/0.179/0.000 ms
root@b195e7feb79a:/#
root@b195e7feb79a:/#
root@b195e7feb79a:/# ping -c 1 192.168.20.7
PING 192.168.20.7 (192.168.20.7) 56(84) bytes of data.
64 bytes from 192.168.20.7: icmp_seq=1 ttl=64 time=0.199 ms

--- 192.168.20.7 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.199/0.199/0.199/0.000 ms
root@b195e7feb79a:/#
root@b195e7feb79a:/#
root@b195e7feb79a:/# ping -c 1 192.168.20.8
PING 192.168.20.8 (192.168.20.8) 56(84) bytes of data.
64 bytes from 192.168.20.8: icmp_seq=1 ttl=64 time=0.205 ms

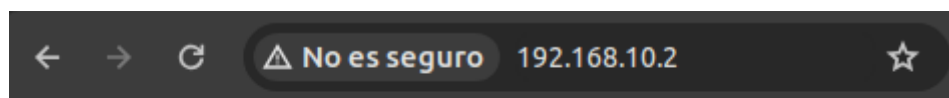
--- 192.168.20.8 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.205/0.205/0.205/0.000 ms
```

Tareas Básicas - B5. Pruebas Básicas

Acceder a la página web de cada contenedor usando su dirección IP y verificar que muestra la información correcta.

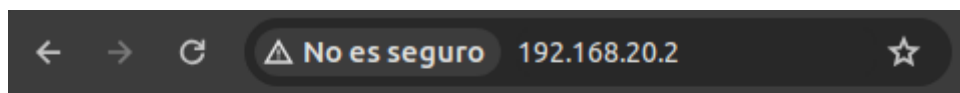
Para esta tarea no veo necesaria la consulta a la IA generativa, simplemente visitamos con el navegador dirección IP por dirección IP cada contenedor manualmente.

Web1



SWAP - mtorres26

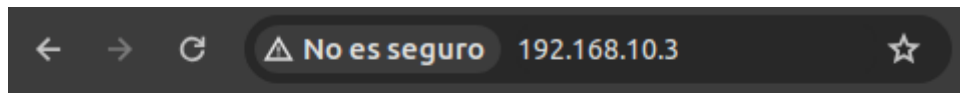
La dirección IP del servidor Apache es: 192.168.10.2



SWAP - mtorres26

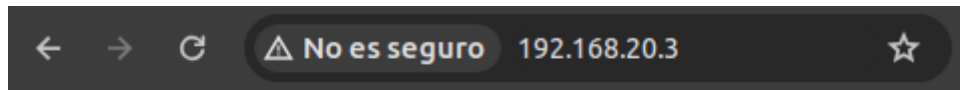
La dirección IP del servidor Apache es: 192.168.20.2

Web2



SWAP - mtorres26

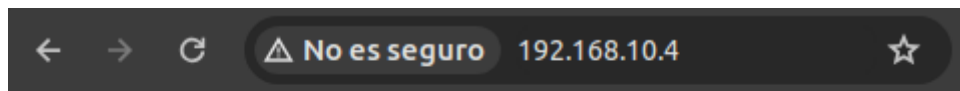
La dirección IP del servidor Apache es: 192.168.10.3



SWAP - mtorres26

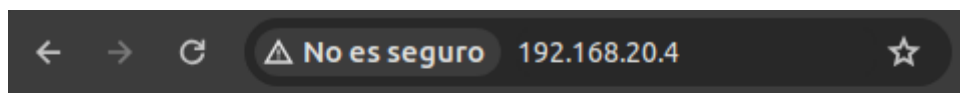
La dirección IP del servidor Apache es: 192.168.20.3

Web3



SWAP - mtorres26

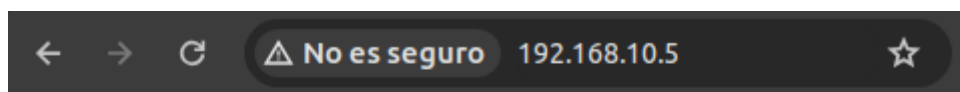
La dirección IP del servidor Apache es: 192.168.10.4



SWAP - mtorres26

La dirección IP del servidor Apache es: 192.168.20.4

Web4



SWAP - mtorres26

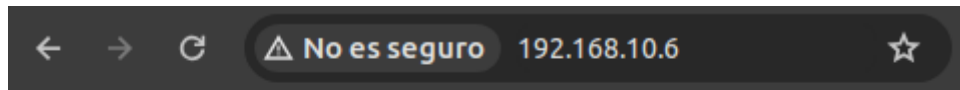
La dirección IP del servidor Apache es: 192.168.10.5



SWAP - mtorres26

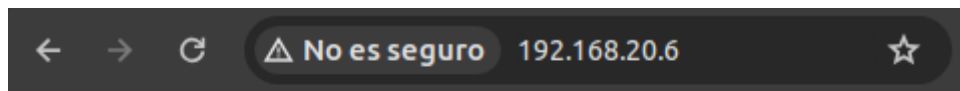
La dirección IP del servidor Apache es: 192.168.20.5

Web5



SWAP - mtorres26

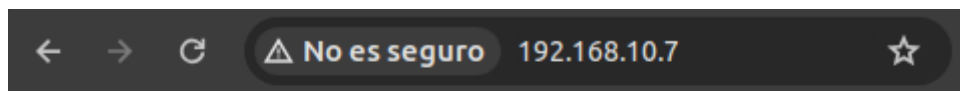
La dirección IP del servidor Apache es: 192.168.10.6



SWAP - mtorres26

La dirección IP del servidor Apache es: 192.168.20.6

Web6



SWAP - mtorres26

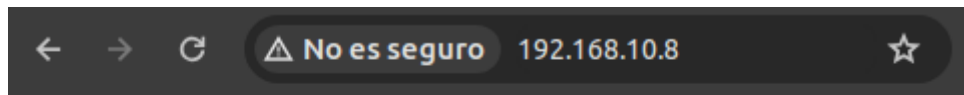
La dirección IP del servidor Apache es: 192.168.10.7



SWAP - mtorres26

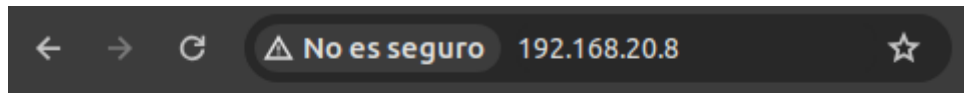
La dirección IP del servidor Apache es: 192.168.20.7

Web7



SWAP - mtorres26

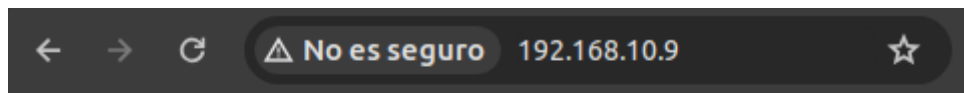
La dirección IP del servidor Apache es: 192.168.10.8



SWAP - mtorres26

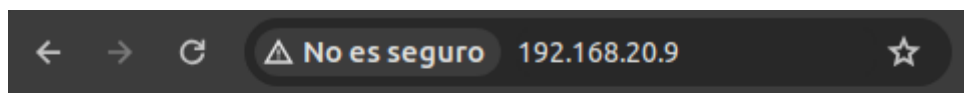
La dirección IP del servidor Apache es: 192.168.20.8

Web8



SWAP - mtorres26

La dirección IP del servidor Apache es: 192.168.10.9



SWAP - mtorres26

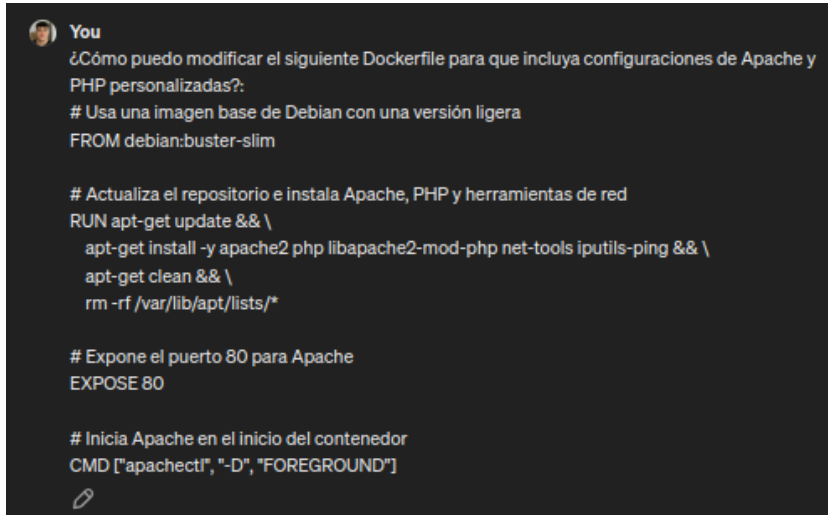
La dirección IP del servidor Apache es: 192.168.20.9

Tareas Avanzadas - A1. Personalización del Dockerfile

Modificar el Dockerfile para incluir configuraciones personalizadas de Apache o PHP

Análisis propuesta IA

Consulta realizada:



You

¿Cómo puedo modificar el siguiente Dockerfile para que incluya configuraciones de Apache y PHP personalizadas?:

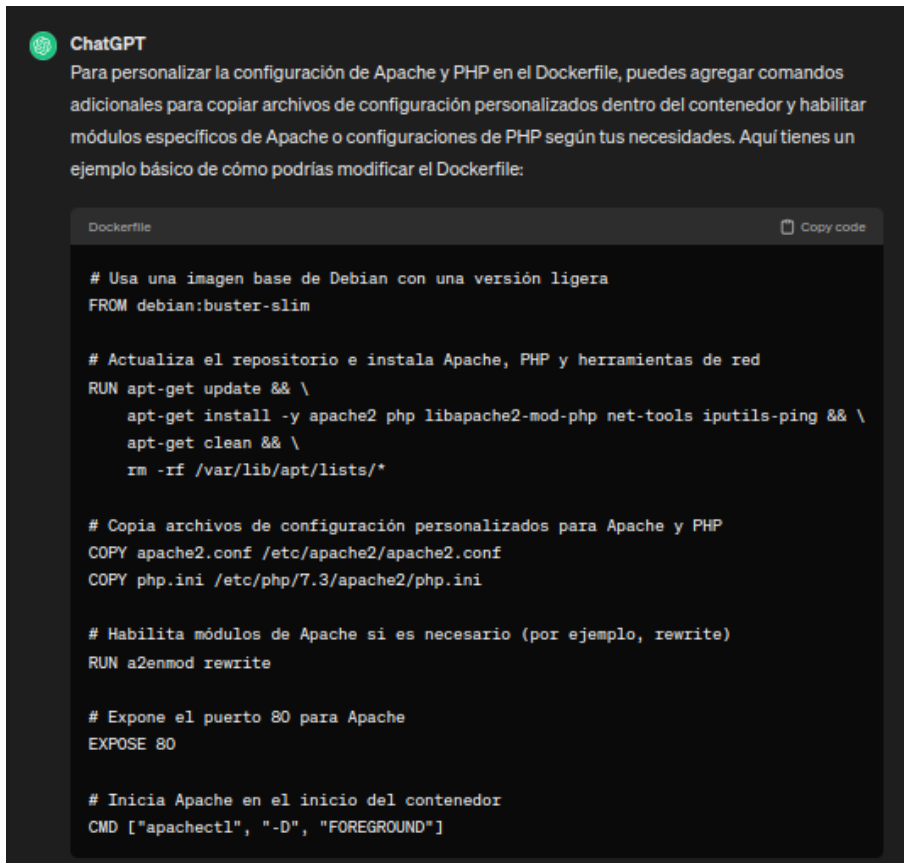
```
# Usa una imagen base de Debian con una versión ligera
FROM debian:buster-slim

# Actualiza el repositorio e instala Apache, PHP y herramientas de red
RUN apt-get update && \
    apt-get install -y apache2 php libapache2-mod-php net-tools iputils-ping && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*

# Expone el puerto 80 para Apache
EXPOSE 80

# Inicia Apache en el inicio del contenedor
CMD ["apachectl", "-D", "FOREGROUND"]
```

Resultado proporcionado por IA:



ChatGPT

Para personalizar la configuración de Apache y PHP en el Dockerfile, puedes agregar comandos adicionales para copiar archivos de configuración personalizados dentro del contenedor y habilitar módulos específicos de Apache o configuraciones de PHP según tus necesidades. Aquí tienes un ejemplo básico de cómo podrías modificar el Dockerfile:

```
Dockerfile

# Usa una imagen base de Debian con una versión ligera
FROM debian:buster-slim

# Actualiza el repositorio e instala Apache, PHP y herramientas de red
RUN apt-get update && \
    apt-get install -y apache2 php libapache2-mod-php net-tools iputils-ping && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*

# Copia archivos de configuración personalizados para Apache y PHP
COPY apache2.conf /etc/apache2/apache2.conf
COPY php.ini /etc/php/7.3/apache2/php.ini

# Habilita módulos de Apache si es necesario (por ejemplo, rewrite)
RUN a2enmod rewrite

# Expone el puerto 80 para Apache
EXPOSE 80

# Inicia Apache en el inicio del contenedor
CMD ["apachectl", "-D", "FOREGROUND"]
```

Análisis detallado:

La IA generativa nos da un buen ejemplo, sin embargo voy a cambiar que el módulo que habilite sea el autoindex, de modo que el Dockerfile, ahora llamado DockerfileApacheModificado_mtorres26 (temporalmente) queda así:

```
Unset
# Usa una imagen base de Debian con una versión ligera
FROM debian:buster-slim

# Actualiza el repositorio e instala Apache, PHP y herramientas de red
RUN apt-get update && \
    apt-get install -y apache2 php libapache2-mod-php net-tools iputils-ping
&& \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*

# Copia archivos de configuración personalizados para Apache y PHP
COPY apache2.conf /etc/apache2/apache2.conf
COPY php.ini /etc/php/7.3/apache2/php.ini

# Habilita módulos de Apache si es necesario (por ejemplo, autoindex)
RUN a2enmod autoindex

# Expone el puerto 80 para Apache
EXPOSE 80

# Inicia Apache en el inicio del contenedor
CMD ["apachectl", "-D", "FOREGROUND"]
```

Vamos a modificar nuestros ficheros apache2.conf y php.ini:

En el fichero apache2.conf vamos a modificar los siguientes parámetros por defecto:

```
Unset
LogLevel notice
KeepAliveTimeout 10
MaxKeepAliveRequests 200
Timeout 250
```

En el fichero php.ini vamos a modificar los siguientes parámetros por defecto:


```

Unset
expose_php = On
max_execution_time = 25
max_input_time = 50
log_errors = On
html_errors = On
error_prepend_string = "<span style='color: #ff0000'>"

```

Ejecutamos ‘docker compose build’ para generar la imagen con el Dockerfile.

```

mtorres@mtorres:~/UGR_NATIVO/CuartoCurso/SegundoCuatrI/SWAP/P1$ docker compose build
[+] Building 89.0s (10/10) FINISHED                                docker:default
=> [web1 internal] load build definition from DockerfileApacheModificado_mtorres26
=> transferring dockerfile: 755B
=> [web1 internal] load metadata for docker.io/library/debian:buster-slim
=> [web1 internal] load .dockerignore
=> transferring context: 2B
=> [web1 1/5] FROM docker.io/library/debian:buster-slim@sha256:4620251e86cea652ef2746c8abef2fcc3ea54abb5f014bfbf841159d805cd1fd
=> resolve docker.io/library/debian:buster-slim@sha256:4620251e86cea652ef2746c8abef2fcc3ea54abb5f014bfbf841159d805cd1fd
=> sha256:4620251e86cea652ef2746c8abef2fcc3ea54abb5f014bfbf841159d805cd1fd 984B / 984B
=> sha256:886b6d8718a3a6fe54881beef0dc76af5966848638ddbaf94efd1fe9aa3d2924c 529B / 529B
=> sha256:0b031cf4038f92451c8dbdaf26901cb6f66fc3774cbd454e83262f74bebb38cd 1.46kB / 1.46kB
=> sha256:4a21b857313366ff622f19e31c4300a81dce92b023b4ff6583ca167179804ac 27.19MB / 27.19MB
=> extracting sha256:4a21b857313366ff622f19e31c4300a81dce92b023b4ff6583ca167179804ac
=> [web1 internal] load build context
=> transferring context: 80.22kB
=> [web1 2/5] RUN apt-get update && apt-get install -y apache2 php libapache2-mod-php net-tools iputils-ping && apt-get clean &&
=> [web1 3/5] COPY apache2.conf /etc/apache2/apache2.conf
=> [web1 4/5] COPY php.ini /etc/php/7.3/apache2/php.ini
=> [web1 5/5] RUN a2enmod autoindex
=> [web1] exporting to image
=> exporting layers
=> writing image sha256:123021c66d432166147b763fc821a329cb21c880a52ac8a6f0f29ba7592dd5e7
=> naming to docker.io/library/mtorres26-apache-image:p1

```

En un contenedor cualquiera miramos el fichero apache2.conf para ver si nuestras configuraciones se han cargado bien:

```

root@bd9d9f06e7fb:/# cat /etc/apache2/apache2.conf | grep LogLevel
# LogLevel: Control the severity of messages logged to the error_log.
# "LogLevel info ssl:warn"
LogLevel notice
root@bd9d9f06e7fb:/# cat /etc/apache2/apache2.conf | grep LogLevel notice
grep: notice: No such file or directory
root@bd9d9f06e7fb:/# cat /etc/apache2/apache2.conf | grep LogLevel
# LogLevel: Control the severity of messages logged to the error_log.
# "LogLevel info ssl:warn"
LogLevel notice
root@bd9d9f06e7fb:/# cat /etc/apache2/apache2.conf | grep KeepAliveTimeout
# KeepAliveTimeout: Number of seconds to wait for the next request from the
KeepAliveTimeout 10
root@bd9d9f06e7fb:/# cat /etc/apache2/apache2.conf | grep MaxKeepAliveRequests
# MaxKeepAliveRequests: The maximum number of requests to allow
MaxKeepAliveRequests 200
root@bd9d9f06e7fb:/# cat /etc/apache2/apache2.conf | grep Timeout
# Timeout: The number of seconds before receives and sends time out.
Timeout 250

```

Lo mismo para el fichero php.ini:

```

mtorres@mtorres:~/UGR_NATIVO/CuartoCurso/SegundoCuatrI/SWAP/P1$ docker compose exec web1 /bin/bash
root@bd9d9f06e7fb:/# cat /etc/php/7.3/apache2/
conf.d/ php.ini
root@bd9d9f06e7fb:/# cat /etc/php/7.3/apache2/php.ini | grep expose_php
expose_php = On
root@bd9d9f06e7fb:/# cat /etc/php/7.3/apache2/php.ini | grep max_execution_time
max_execution_time = 25
root@bd9d9f06e7fb:/# cat /etc/php/7.3/apache2/php.ini | grep max_input_time
; max_input_time
max_input_time = 60
root@bd9d9f06e7fb:/# cat /etc/php/7.3/apache2/php.ini | grep log_errors
; log_errors
log_errors = On
root@bd9d9f06e7fb:/# cat /etc/php/7.3/apache2/php.ini | grep html_errors
html_errors = On
; If html_errors is set to On *and* docref_root is not empty, then PHP
root@bd9d9f06e7fb:/# cat /etc/php/7.3/apache2/php.ini | grep error_prepend_string
error_prepend_string = "<span style='color: #ff0000'>"
root@bd9d9f06e7fb:/#

```

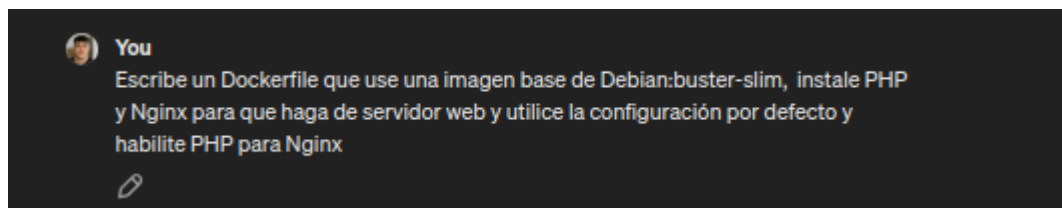
Tareas Avanzadas - A2. Creación de contenedores con otros servidores web

Crear contenedores con otros servidores web (nginx, lighttpd, etc.)

Análisis propuesta IA

Consulta realizada:

Voy a repetir la consulta realizada en el apartado B2 pero usando Nginx en lugar de Apache.



Resultado proporcionado por IA:



Análisis detallado:

Utilizamos la respuesta de la IA generativa pero vamos a añadirle configuración personalizada.

En el fichero 'mi_config.conf' vamos a realizar los siguientes cambios a la configuración por defecto:

```
Unset
index index.php index.html index.htm;
server_name mtorres26.org;
```

El archivo 'mi_config.conf' queda así:

```
Unset
server {
    listen 80;
    server_name mtorres26.org;

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

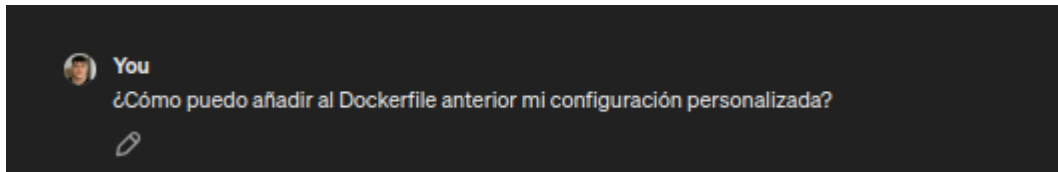
    location / {
        try_files $uri $uri/ /index.php?$query_string;
    }

    location ~ /\.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/var/run/php/php7.3-fpm.sock; # Ajusta el path y
la versión de PHP según tu configuración
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        include fastcgi_params;
    }

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

Ahora necesitamos añadir dicho archivo a la configuración.

Consulta realizada:



Resultado proporcionado por IA:

ChatGPT

Para añadir tu propia configuración personalizada a Nginx en el Dockerfile anterior, necesitas copiar tu archivo de configuración personalizado al contenedor y hacer que Nginx lo utilice en lugar del archivo de configuración por defecto. Aquí tienes una forma de hacerlo:

Supongamos que tienes un archivo de configuración personalizado llamado `mi_configuracion_personalizada.conf`. Aquí está cómo podrías modificar el Dockerfile para añadir tu configuración personalizada:

```
Dockerfile
```

```
# Usa una imagen base de Debian con una versión ligera
FROM debian:buster-slim

# Actualiza el repositorio e instala Nginx, PHP y herramientas de red
RUN apt-get update && \
    apt-get install -y nginx php-fpm php-mysql net-tools iputils-ping && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*

# Copia tu archivo de configuración personalizado a la ubicación adecuada
COPY mi_configuracion_personalizada.conf /etc/nginx/sites-available/

# Habilita tu archivo de configuración personalizado creando un enlace simbólico
RUN ln -s /etc/nginx/sites-available/mi_configuracion_personalizada.conf /etc/nginx/sites-enabled/

# Elimina el archivo de configuración predeterminado de Nginx
RUN rm /etc/nginx/sites-enabled/default

# Expone el puerto 80 para Nginx
EXPOSE 80

# Inicia Nginx y PHP-FPM en el inicio del contenedor
CMD service php7.3-fpm start && nginx -g "daemon off;"
```

En este Dockerfile modificado:

1. Se copia tu archivo de configuración personalizado (`mi_configuracion_personalizada.conf`) al directorio `/etc/nginx/sites-available/` dentro del contenedor.
2. Se crea un enlace simbólico desde el archivo de configuración personalizado al directorio `/etc/nginx/sites-enabled/`, que es donde Nginx busca los archivos de configuración activos.
3. Se elimina el archivo de configuración predeterminado de Nginx para evitar conflictos.
4. Se asegura de exponer el puerto 80 para que Nginx pueda recibir solicitudes.
5. Se inicia PHP-FPM y Nginx al iniciar el contenedor.

Análisis detallado:

Sólo vamos a copiar el fichero de 'mi_config.conf' para que sustituya a '/etc/nginx/sites-available/default', lo demás proporcionado por la IA no es necesario.

El DockerfileNginx_mtorres26 queda así:

```
Unset
# Usa una imagen base de Debian con una versión ligera
FROM debian:buster-slim

# Actualiza el repositorio e instala Nginx, PHP y herramientas de red
RUN apt-get update && \
    apt-get install -y nginx php-fpm php-mysql net-tools iputils-ping && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*

# Copia tu archivo de configuración personalizado a la ubicación adecuada en
el contenedor
COPY mi_config.conf /etc/nginx/sites-available/default

# Expone el puerto 80 para Nginx
EXPOSE 80

# Inicia Nginx y PHP-FPM en el inicio del contenedor
CMD service php7.3-fpm start && nginx -g "daemon off;"
```

Vemos cómo visitamos cualquier IP de un contenedor y si “inspeccionamos” la red con las herramientas del navegador nos sale el servidor Nginx:

The screenshot shows a web browser window with the address bar displaying '192.168.10.4'. The page title is 'SWAP - mtorres26' and the content indicates 'La dirección IP del servidor Apache es: 192.168.10.4'. The browser's developer tools are open to the 'Network' tab, showing a single request to 'http://192.168.10.4/'. The request details are as follows:

Request Details	
Request URL:	http://192.168.10.4/
Request Method:	GET
Status Code:	200 OK
Remote Address:	192.168.10.4:80
Referrer Policy:	strict-origin-when-cross-origin
Response Headers	
Connection:	keep-alive
Content-Encoding:	gzip
Content-Type:	text/html; charset=UTF-8
Date:	Tue, 09 Apr 2024 21:04:51 GMT
Server:	nginx/1.14.2
Transfer-Encoding:	chunked