

## Ejercicio 5 - Imagen con Dockerfile - Aplicación Web

Tarea Docker: Gerald Alexis Rueda Tejedo y Sara García Barbas

[Paso 1](#)

[Paso 2](#)

[Paso 3](#)

[Paso 4](#)

[Paso 5](#)

[Paso 6](#)

[Paso 7](#)

[Paso 8](#)

[Paso 9](#)

### Paso 1

Arranca un contenedor que ejecute una instancia de la imagen `php:7.4-apache`, que se llame web y que sea accesible desde un navegador en el puerto `8000`.

Primero vamos a crear un directorio donde vamos a almacenar la páginas de nuestro sitio web y luego vamos a hacer que el contenedor almacene los datos de `/var/www/html` en ese directorio con un bind mount.

```
mkdir mitisioweb
```

```
docker run -d --name web -v /home/gerald/mitisioweb:/var/www/html -p 8000
```

```
gerald@clientlinux ~$ mkdir mitisioweb
gerald@clientlinux ~$ docker run -d --name web -v /home/gerald/mitisioweb:/var/www/html -p 8000:80 php:7.4-apache
7ee0963919465c8648e6e9bc87d5ff93282d6fd1f0173217440884a1a3e88409
gerald@clientlinux ~$
```

## Paso 2

Coloca en el directorio raíz del servicio web ( `/var/www/html` ) un "sitio web" donde figure

el nombre de los componentes del grupo - el sitio deberá tener al menos un archivo

`index.html` y un archivo `.css`

Creamos el sitio web index.html

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Mi Sitio Web</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <h1>Bienvenidos a Nuestro Proyecto</h1>
  <p>Este sitio fue creado por Sara y Gerald.</p>
  <a href="script.php">Ver script PHP</a>
</body>
</html>
```

Con su CSS:

```
body {
    font-family: Arial, sans-serif;
    text-align: center;
    background-color: #cbf7a3;
}
h1 {
    color: #333;
}
```

Y el `script.php` :

```
<?php
    setlocale(LC_TIME, "es_ES.UTF-8");
    $mes_actual = strftime("%B");
    $fecha_actual = date("d/m/Y");
    $hora_actual = date("H:i:s");
    echo "<h1>Información</h1>";
    echo "<p>Hoy es $fecha_actual</p>";
    echo "<p>El mes es: <strong>$mes_actual</strong></p>";
    echo "<p>Hora: $hora_actual</p>";
?>
```

Copiamos los archivos a `/var/www/html` :

```
gerald@clientlinux ~/misitioweb$ docker cp /home/gerald/misitioweb/index.html web:/var/www/html
Successfully copied 2.05kB to web:/var/www/html
gerald@clientlinux ~/misitioweb$ docker cp /home/gerald/misitioweb/style.css web:/var/www/html
Successfully copied 2.05kB to web:/var/www/html
gerald@clientlinux ~/misitioweb$ docker cp /home/gerald/misitioweb/script.php web:/var/www/html
Successfully copied 2.05kB to web:/var/www/html
```

## Paso 3

Coloca en ese mismo directorio raíz el siguiente script `php`

```
<?php
    setlocale(LC_TIME, "es_ES.UTF-8");
    $mes_actual = strftime("%B");
    $fecha_actual = date("d/m/Y");
    $hora_actual = date("H:i:s");
    echo "<h1>Información</h1>";
```

```
echo "<p>Hoy es $fecha_actual</p>";  
echo "<p>El mes es: <strong>$mes_actual</strong></p>";  
echo "<p>Hora: $hora_actual</p>";  
?>
```

Ya lo hemos hecho en el paso anterior 🤖

## Paso 4

**Ver la salida del script y de la página `index` en el navegador**

Como hemos creado un bind mount debemos cambiar los permisos de nuestra carpeta:

```
sudo chown -R www-data:www-data /home/gerald/misitioweb/  
sudo chmod -R 777 /home/gerald/misitioweb/
```

```
gerald@clientelinux ~/misitioweb$ sudo chown -R www-data:www-data /home/gerald/misitioweb/  
gerald@clientelinux ~/misitioweb$ sudo chmod -R 777 /home/gerald/misitioweb/
```


Ya vemos nuestra página en el navegador:



## Paso 5

**Automatizar estas operaciones creando un fichero Dockerfile**

Creamos un archivo en el directorio `mistioweb` al que llamaremos `Dockerfile` y le añadiremos el siguiente contenido:



```
1 # Usar la imagen oficial de PHP con Apache
2 FROM php:7.4-apache
3
4 # Establecer el directorio de trabajo
5 WORKDIR /var/www/html
6
7 # Copiar el sitio web y el script PHP al contenedor
8 COPY . /var/www/html/
9
10 # Hacer que Apache pueda servir el contenido
11 RUN chown -R www-data:www-data /var/www/html
12
13 # Habilitar mod_rewrite para permitir URLs limpias si es necesario
14 RUN a2enmod rewrite
15
16 # Exponer el puerto 80 (Apache por defecto)
17 EXPOSE 80
```

▼ Este archivo hace lo siguiente:

- Usa la imagen base de PHP con Apache ( `php:7.4-apache` ).
- Establece el directorio de trabajo en `/var/www/html` (el directorio raíz de Apache).
- Copia todos los archivos desde el directorio local al contenedor.
- Cambia los permisos de los archivos para que Apache pueda acceder a ellos.
- Activa el módulo `rewrite` de Apache (por si lo necesitas más adelante para redirecciones o URLs limpias).
- Expone el puerto 80 para que Apache sea accesible.

## Paso 6

### Subir la imagen a la cuenta de Docker Hub

Abrimos una terminal en el directorio donde está el `Dockerfile` y ejecutamos:

```
docker build -t gesaraweb .
```

Este comando construye la imagen de Docker a partir del `Dockerfile` y la etiqueta como `gesaraweb`.

```
gerald@clientelinux ~/misitioweb$ docker build -t gesaraweb .
[+] Building 1.6s (10/10) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile              0.1s
=> => transferring dockerfile: 489B                             0.0s
=> [internal] load metadata for docker.io/library/php:7.4-apache 0.0s
=> [internal] load .dockerignore                                0.0s
=> => transferring context: 2B                                    0.0s
=> [1/5] FROM docker.io/library/php:7.4-apache                 0.3s
=> [internal] load build context                                0.1s
=> => transferring context: 1.47kB                               0.0s
=> [2/5] WORKDIR /var/www/html                                  0.1s
=> [3/5] COPY . /var/www/html/                                  0.1s
=> [4/5] RUN chown -R www-data:www-data /var/www/html          0.3s
=> [5/5] RUN a2enmod rewrite                                    0.4s
=> exporting to image                                           0.2s
=> => exporting layers                                          0.1s
=> => writing image sha256:bdc8cca7f1b28ac8f12c9f829a32cb03a76e9e3e277ca 0.0s
=> => naming to docker.io/library/gesaraweb                    0.0s
gerald@clientelinux ~/misitioweb$
```

## Paso 7

Ahora eliminaremos el contenedor `web` original. Luego crearemos un nuevo contenedor con la imagen que acabamos de crear para comprobar que funciona correctamente:

1. Eliminamos el contenedor con el siguiente comando:

```
docker rm -f web
```

```
gerald@clientelinux ~/misitioweb$ docker rm -f web
web
gerald@clientelinux ~/misitioweb$
```

2. Creamos el contenedor con la imagen

```
docker run -d --name web -p 8000:80 gesaraweb
```

```
gerald@clientelinux ~/misitioweb$ docker run -d --name web -p 8000:80 gesaraweb
b466bb85eb44360d14146eb83d8cd29447dbf496b0ac54c68872ccfcffbc663
gerald@clientelinux ~/misitioweb$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
b466bb85eb44	gesaraweb	"docker-php-entrypo..."	11 seconds ago	Up 10 seconds	0.0.0.0:8000->80/tcp, [::]:8000->80/tcp	web
6e05ebba2af8	debian:latest	"/bin/bash -c 'apt u..."	About an hour ago	Up About an hour		htop_container

3. Comprobamos desde el navegador que vemos nuestro sitio web:



## Paso 8

**Subir a Docker Hub la imagen creada.**

Nos logueamos en Docker Hub para subir la imagen que creamos:

```
docker login -u gerald1995
```

```
gerald@clientelinux ~/misitioweb$ docker login -u gerald1995
Password:
WARNING! Your password will be stored unencrypted in /home/gerald/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credential-stores

Login Succeeded
gerald@clientelinux ~/misitioweb$
```

Ahora le ponemos un nombre de etiqueta a la imagen (requisito de Docker Hub para poder subir la imagen) y luego ya procedemos a subirla:

```
docker tag gesaraweb gerald1995/gesaraweb:latest
docker push gerald1995/gesaraweb:latest
```

```

gerald@clientelinux ~/misitioweb$ docker tag gesaraweb gerald1995/gesaraweb:latest
gerald@clientelinux ~/misitioweb$ docker push gerald1995/gesaraweb:latest
The push refers to repository [docker.io/gerald1995/gesaraweb]
fa63f00770e4: Pushed
96342e5347d4: Pushed
d445aef72b0b: Pushed
5f70bf18a086: Pushed
3d33242bf117: Pushed
529016396883: Mounted from library/php
5464bcc3f1c2: Mounted from library/php
28192e867e79: Mounted from library/php
d173e78df32e: Mounted from library/php
0be1ec4fbfdc: Mounted from library/php
30fa0c430434: Mounted from library/php
a538c5a6e4e0: Mounted from library/php
e5d40f64dcb4: Mounted from library/php
44148371c697: Mounted from library/php
797a7c0590e0: Mounted from library/php
f60117696410: Mounted from library/php
ec4a38999118: Mounted from library/php
latest: digest: sha256:22a63a4a69274a27e58a3e0f0b3851d785a199de2aff6bda09c6bffa8f4ea5c6
gerald@clientelinux ~/misitioweb$

```

Ahora comprobamos que la imagen ya aparece en el repositorio de Docker Hub personal:

Name	Last Pushed	Contains	Visibility	Scout
gerald1995/gesaraweb	4 minutes ago	IMAGE	Public	Inactive

## Paso 9

Ahora desde la máquina virtual de la otra persona (**cliente@clienteLinux**), descargamos la imagen que habíamos creado anteriormente:

```
docker pull gerald1995/gesaraweb:latest
```



```
cliente@clienteLinux ~$ docker pull gerald1995/gesaraweb:latest
latest: Pulling from gerald1995/gesaraweb
a603fa5e3b41: Pull complete
c428f1a49423: Pull complete
156740b07ef8: Pull complete
fb5a4c8af82f: Pull complete
25f85b498fd5: Pull complete
9b233e420ac7: Pull complete
fe42347c4ecf: Pull complete
d14eb2ed1e17: Pull complete
66d98f73acb6: Pull complete
d2c43c5efbc8: Pull complete
ab590b48ea47: Pull complete
80692ae2d067: Pull complete
05e465aaa99a: Pull complete
4f4fb700ef54: Pull complete
f500754c873f: Pull complete
b7911daaf0e1: Pull complete
5de8d6e2db23: Pull complete
Digest: sha256:22a63a4a69274a27e58a3e0f0b3851d785a199de2aff6bda09c6bffa8f4ea5c6
Status: Downloaded newer image for gerald1995/gesaraweb:latest
docker.io/gerald1995/gesaraweb:latest
cliente@clienteLinux ~$
```

Creamos el contenedor con la imagen:

```
docker run -d --name nuestrositioweb -p 8000:80 gerald1995/gesaraweb:late
```

```
cliente@clienteLinux ~$ docker run -d --name nuestrositioweb -p 8000:80 gerald1995/gesaraweb:late
st
6429f50c3fa1f7756f81c674ce147bbee7073bb1cce674625853badb128b5009
```

Y comprobamos desde el navegador que vemos la página web desde la máquina virtual de la otra persona (**cliente@clienteLinux**):



