

E5 - Despliegue de una Aplicación Web con Dockerfile

Álvaro Ronco Acebal y Guillermo Jesús Martín Pérez

E5 - Despliegue de una Aplicación Web con Dockerfile

Crear el directorio del proyecto

Crear los archivos del sitio web

Construir y ejecutar la imagen

Subir la imagen a Docker Hub

Descargar la imagen desde otra máquina

Eliminar todo

En este ejercicio, crearemos una imagen personalizada basada en PHP 7.4 con Apache, subiremos la imagen a Docker Hub y desplegaremos una aplicación web simple en un contenedor.

Crear el directorio del proyecto

Para organizar los archivos, creamos una carpeta y dentro de ella los archivos necesarios:

```
$mkdir mi-web  
$cd mi-web
```

Dentro de esta carpeta se deben incluir:

- Un archivo `index.html` con los nombres de los integrantes del grupo.
- Un archivo `styles.css` con un diseño básico.
- Un archivo `script.php` que ejecutará código en PHP.
- Un `Dockerfile` para la automatización del contenedor.

```
alvaro@alvaro-VirtualBox ~$ mkdir mi-web  
alvaro@alvaro-VirtualBox ~$ cd mi-web
```

Crear los archivos del sitio web

Archivo `index.php`:

```
<!DOCTYPE html>  
<html lang="es">  
  <head>  
    <meta charset="UTF-8" />  
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />  
    <title>Incluir PHP en HTML</title>  
  </head>  
  <body>  
    <h1>Hola, este es un ejemplo de incluir un archivo PHP</h1>  
  
    <?php
```

```
// Incluir el archivo PHP en el HTML
include 'script.php'; // O también podrías usar require

?>
</body>
</html>
```

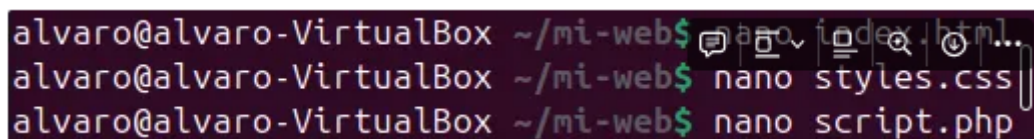
Archivo `styles.css`:

```
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
    text-align: center;
}
h1 {
    color: #333;
}
```

Archivo `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>";
?>
```

Todos creados y editados usando `nano`:



```
alvaro@alvaro-VirtualBox ~/mi-web$ nano index.html
alvaro@alvaro-VirtualBox ~/mi-web$ nano styles.css
alvaro@alvaro-VirtualBox ~/mi-web$ nano script.php
```

Finalmente creamos el archivo Docker con lo siguiente:

```
# Usar la imagen oficial de PHP con Apache
FROM php:7.4-apache

# Copiar los archivos del sitio web al directorio raíz del servidor
COPY . /var/www/html/

# Exponer el puerto 8000 para acceder a la aplicación
EXPOSE 8000

# Iniciar Apache en segundo plano
CMD ["apache2-foreground"]
```

Explicación:

- `FROM php:7.4-apache` → Usa la imagen oficial de PHP con Apache.

- `COPY . /var/www/html/` → Copia los archivos del sitio web al contenedor.
- `EXPOSE 8000` → Indica que el contenedor usará el puerto 8000.
- `CMD ["apache2-foreground"]` → Inicia Apache al arrancar el contenedor.

Construir y ejecutar la imagen

Construir la imagen personalizada, para ello ejecutamos el siguiente comando en la carpeta donde está el `Dockerfile`:

```
$docker build -t mi-web .
```

```
alvaro@alvaro-VirtualBox ~/mi-web$ docker build -t mi-web .
[+] Building 0.6s (7/7) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile              0.0s
=> == transferring dockerfile: 324B                             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
=> [internal] load build context                                0.1s
=> == transferring context: 1.07kB                               0.0s
=> [1/2] FROM docker.io/library/php:7.4-apache                 0.3s
=> [2/2] COPY . /var/www/html/                                  0.1s
=> exporting to image                                           0.1s
=> == exporting layers                                           0.0s
=> == writing image sha256:d8935ca38573a69c0c2cad75e45656bfc56f7907d5b8f 0.0s
=> == naming to docker.io/library/mi-web                        0.0s
```

Ejecutar el contenedor:

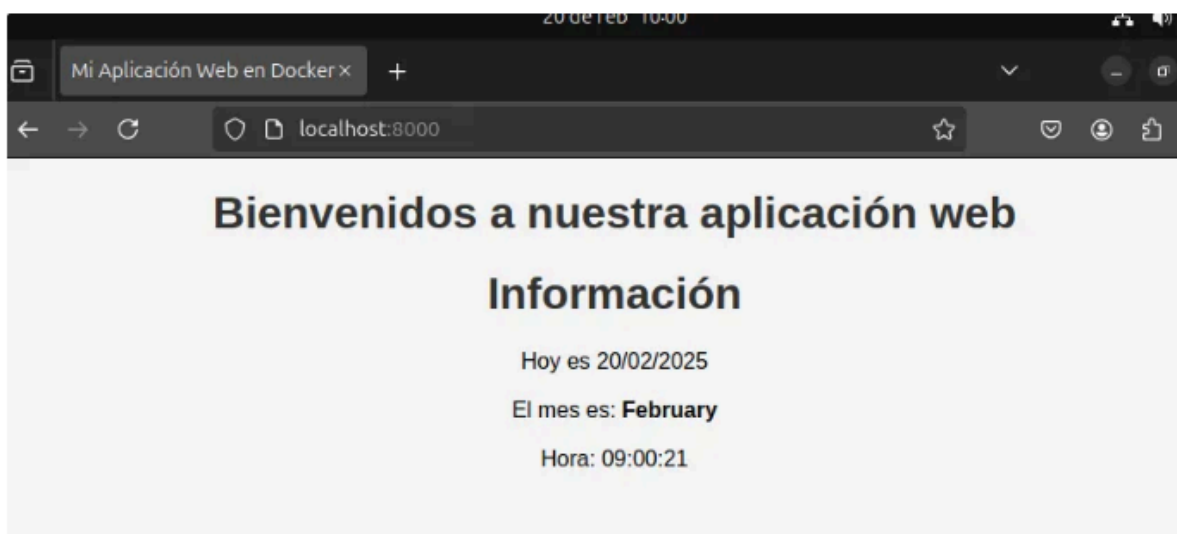
```
$docker run -d -p 8000:80 --name web mi-web
```

Explicación:

- `-p 8000:80` → Mapea el puerto 80 del contenedor al puerto 8000 del host.
- `--name web` → Nombra el contenedor como `web`.

```
alvaro@alvaro-VirtualBox ~/mi-web$ docker run -d -p 8000:80 --name web mi-web
6b628ef3d910fa07ec6d40773279b3bed34f3c05bdb206d3cb5cac3c5432cc8e
```

Abre un navegador y accede a <http://localhost:8000> para ver la página cargada.



Subir la imagen a Docker Hub

Para compartir la imagen con el equipo, la subimos a Docker Hub.

1. Iniciar sesión en Docker Hub:

```
$docker login
```

```
alvaro@alvaro-VirtualBox ~/mi-web$ docker login

USING WEB-BASED LOGIN
To sign in with credentials on the command line, use 'docker login -u <username> -p <password>'

Your one-time device confirmation code is: RJJH-CQBH
Press ENTER to open your browser or submit your device code here: https://login.docker.com/activate

Waiting for authentication in the browser...

WARNING! Your password will be stored unencrypted in /home/alvaro/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credential-stores

Login Succeeded
```

2. Etiquetar la imagen con el nombre del usuario en Docker Hub y subir la imagen al repositorio:

```
$docker tag mi-web alvarora62/mi-web:lastest
$docker push alvarora62/mi-web:lastest
```

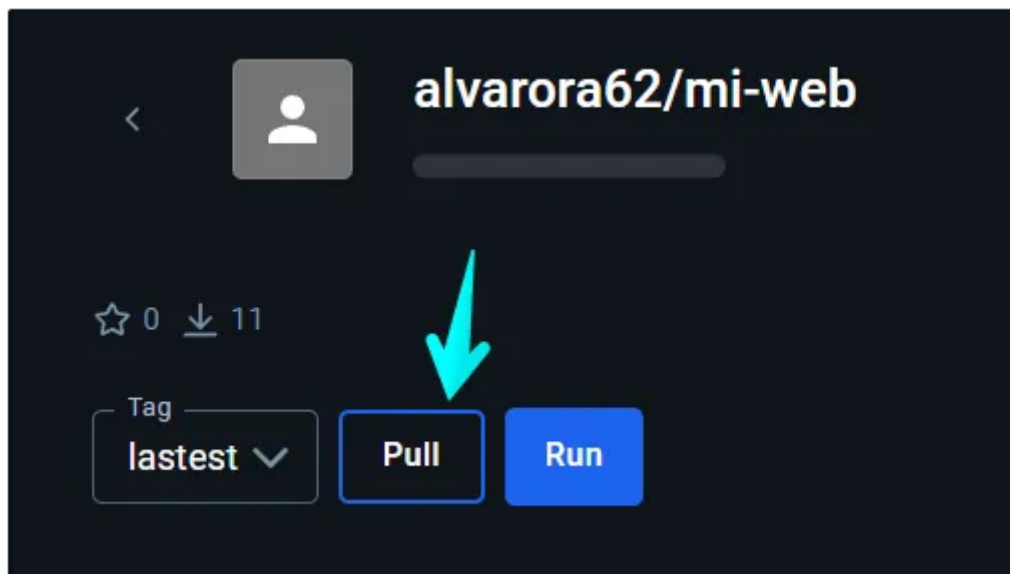
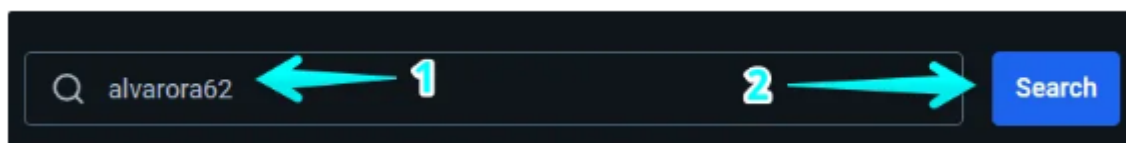
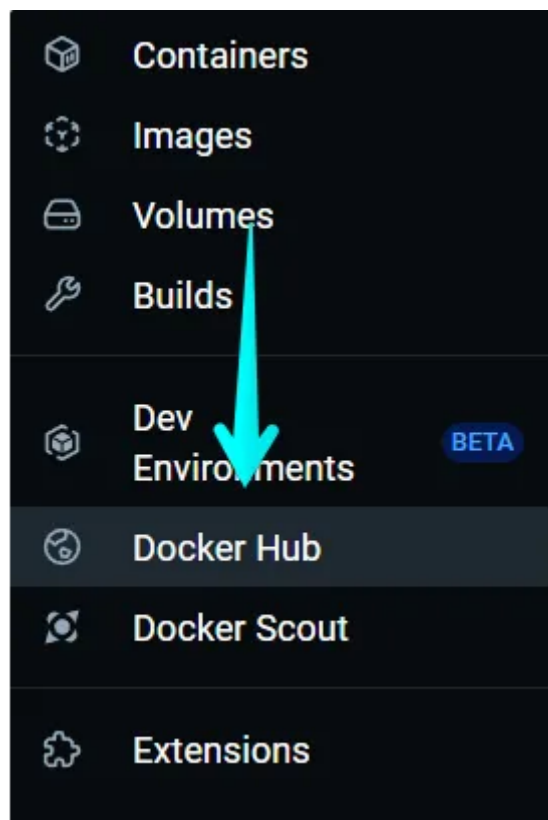
```

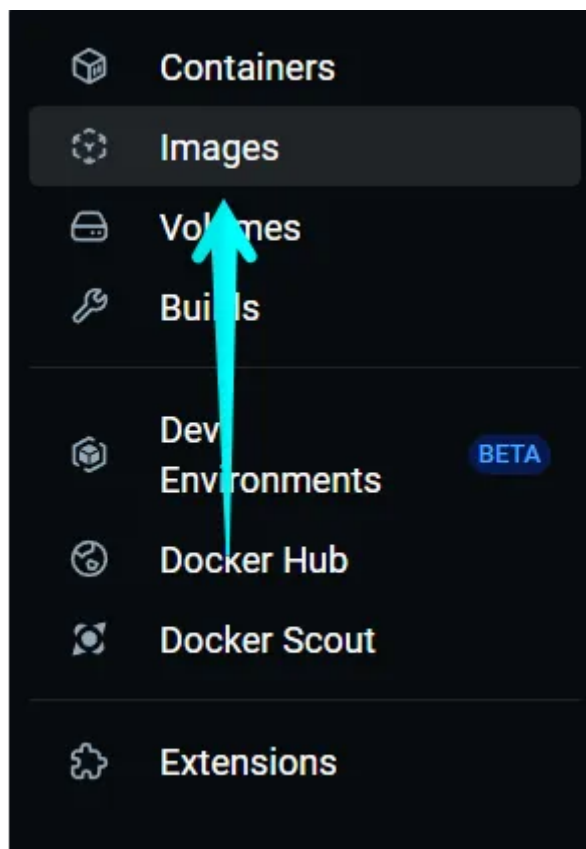
alvaro@alvaro-VirtualBox ~/mi-web$ docker build -t mi-web .
[+] Building 0.3s (7/7) FINISHED
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 324B 0.0s
=> [internal] load metadata for docker.io/library/php:7.4-apache 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [internal] load build context 0.0s
=> => transferring context: 489B 0.0s
=> CACHED [1/2] FROM docker.io/library/php:7.4-apache 0.0s
=> [2/2] COPY . /var/www/html/ 0.0s
=> exporting to image 0.1s
=> => exporting layers 0.0s
=> => writing image sha256:9478ad87592b7263140c82e4783c1fdfe9afd0722b7e7ac0d66e 0.0s
=> => naming to docker.io/library/mi-web 0.0s
alvaro@alvaro-VirtualBox ~/mi-web$ docker run -d -p 8000:80 --name web mi-web
af07618c1c0eb668acec65225987034bb42c9bc7cdb6b6284ac19087aa5b2203
alvaro@alvaro-VirtualBox ~/mi-web$ docker tag mi-web alvarora62/mi-web:lastest
alvaro@alvaro-VirtualBox ~/mi-web$ docker push alvarora62/mi-web:lastest
The push refers to repository [docker.io/alvarora62/mi-web]
eb25e435e4ea: Pushed
3d33242bf117: Layer already exists
529016396883: Layer already exists
5464bcc3f1c2: Layer already exists
28192e867e79: Layer already exists
d173e78df32e: Layer already exists
0be1ec4fbfdc: Layer already exists
30fa0c430434: Layer already exists
a538c5a6e4e0: Layer already exists
e5d40f64dcb4: Layer already exists
44148371c697: Layer already exists
797a7c0590e0: Layer already exists
f60117696410: Layer already exists
ec4a38999118: Layer already exists
latest: digest: sha256:917fcbd9bba7a9d0b9167c9faa9dcaa92467ce4be0a58fc91d3bd54959bb7e6
1 size: 3242
alvaro@alvaro-VirtualBox ~/mi-web$

```

Descargar la imagen desde otra máquina

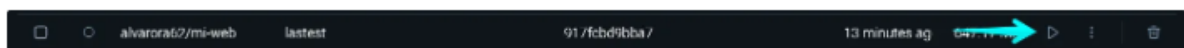
Para probar que la imagen se puede compartir, otro integrante del equipo sigue los siguientes pasos:





<input type="checkbox"/>	Name ↑	Tag	Image ID	Created	Size
<input type="checkbox"/>	nginx	stable	16341a41917d	14 days ago	12
<input type="checkbox"/>	alvarora62/mi-web	latest	917fcbd9bba7	10 minutes ago	64

Una vez descargada la imagen pasamos a crear un contenedor.





Run a new container

alvarora62/mi-web:lastest

Optional settings

Container name

mi-web-alvaro

A random name is generated if you do not provide one.

Ports

Enter "0" to assign randomly generated host ports.

Host port

8081

:80/tcp

Host port

:8000/tcp

Volumes

Host path

...

Container path

+

Environment variables

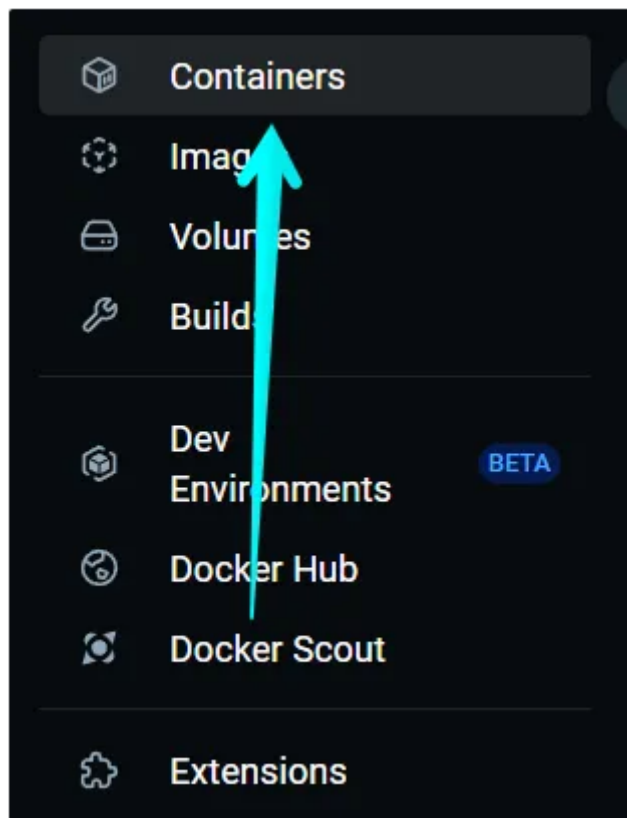
Variable

Value

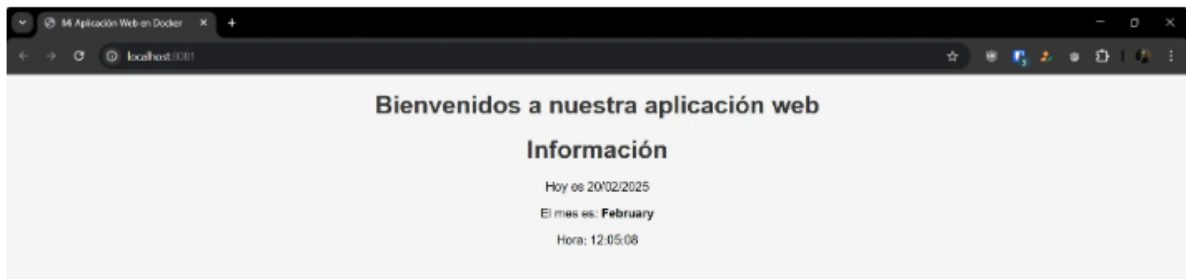
+

Cancel

Run



	Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
<input type="checkbox"/>	mi-sevidor	bb0a85890590	nginx:stable-alpine3.20-perl	8080:80	0%	4 hours ago	
<input checked="" type="checkbox"/>	mi-web-alvaro	7efa8bb1e16f	alvarora62/mi-web:latest	8081:80	0%	35 seconds ago	



Eliminar todo

Cuando terminemos, podemos limpiar el sistema:

```
$docker stop web
$docker rm web
$docker rmi usuario-dockerhub/mi-web
```

```
alvaro@alvaro-VirtualBox ~/mi-web$ docker stop web
web
alvaro@alvaro-VirtualBox ~/mi-web$ docker rm web
web
alvaro@alvaro-VirtualBox ~/mi-web$ docker rmi alvarora62/mi-web
```