Imagen con Dockerfile y aplicación web

> Documento realizado por Roberto Delgado Sánchez - Alumno de Despliegue de Aplicaciones Web - DAW

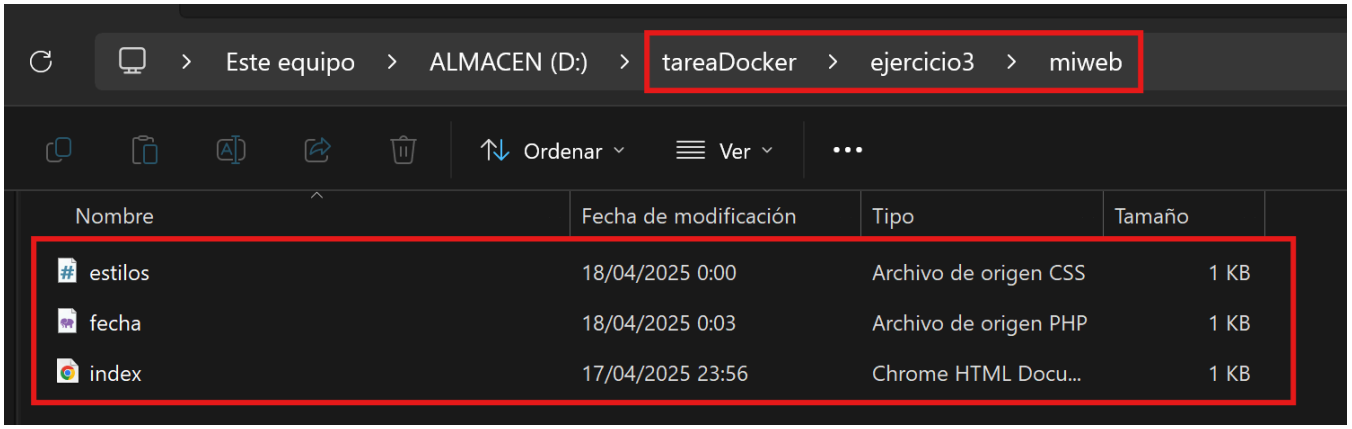
1. Enunciado

En este ejercicio vamos a abordar la creación de una imagen con `Dockerfile`, fichero de **Docker** que nos permite crear una imagen a partir de otra ya creada y modificarla a nuestro gusto con el objetivo de que pueda ser redistribuida posteriormente.

En nuestro caso partiremos de la imagen `php:7.4-apache` a la que le añadiremos un sitio web y un script **PHP** para posteriormente crear otra imagen que subiremos a nuestra cuenta de **Docker Hub**.

2. Creación de la web

Se nos pide que creemos un sitio web sencillo en el que figure nuestro nombre y que debe constar al menos de un archivo `index.html` y de una hoja de estilos. Ambos ficheros los crearemos dentro de una carpeta que se llamará `miweb`:



index.html

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>DAW - Ejercicio 3</title>
    <link rel="stylesheet" href="estilos.css">
</head>
<body>
    <h1>Despliegue de Aplicaciones Web</h1>
    <h2>CIFP La Laboral ---- 2024-2025</h2>
    <h3>Tarea realizada por Roberto Delgado Sánchez</h3>
    <h3>egl33817@educastur.es</h3>
</body>
</html>
```

estilos.css

```css
body
{
    display: flex;
    flex-direction: column;
    align-items: center;
    font-family: Arial, Helvetica, sans-serif;
    background-color: rgb(128, 175, 190);
    color: rgb(23, 42, 42)
}
```

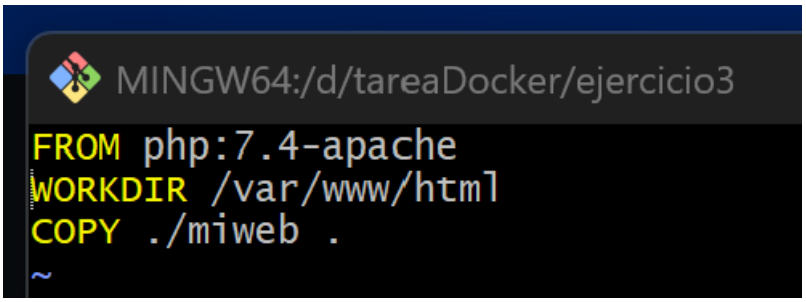
fecha.php

```php
<?php
    setlocale(category: LC_TIME,locales: "es_ES.UTF-8");
    $mes_actual = strftime(format: "%B");
    $fecha_actual = date(format: "d/m/Y");
    $hora_actual = date(format: "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>";
?>
```

3. Creación del fichero Dockerfile y de la imagen

El contenido del fichero `Dockerfile` es el siguiente:

- `FROM` : imagen que usaremos como base para crear la nuestra (`php:7.4-apache` en nuestro caso).
- `WORKDIR` : nos movemos al directorio `/var/www/html` .
- `COPY` : copiamos el contenido de la carpeta `miweb` en el directorio al que nos movimos antes.



Para crear la imagen deseada a partir del fichero `Dockerfile` antes citado usamos el siguiente comando:

```
$ docker built -t egl33817/ejercicio3:v1 .
```

significando cada elemento del mismo lo siguiente:

- `docker build` : comando para crear una imagen a partir de lo especificado en un archivo `Dockerfile` .
- `-t` : opción para darle un nombre a la imagen que vamos a crear.
- `egl33817/ejercicio3` : el nombre de la imagen.
- `v1` : versión de la imagen.
- `.` : ruta en la que están tanto el archivo `Dockerfile` como los archivos necesarios para construir la imagen, es decir, lo que se conoce como contexto en el mundo **Docker**. El punto hace referencia a la carpeta actual.

El resultado de la ejecución de este comando se puede ver a continuación:

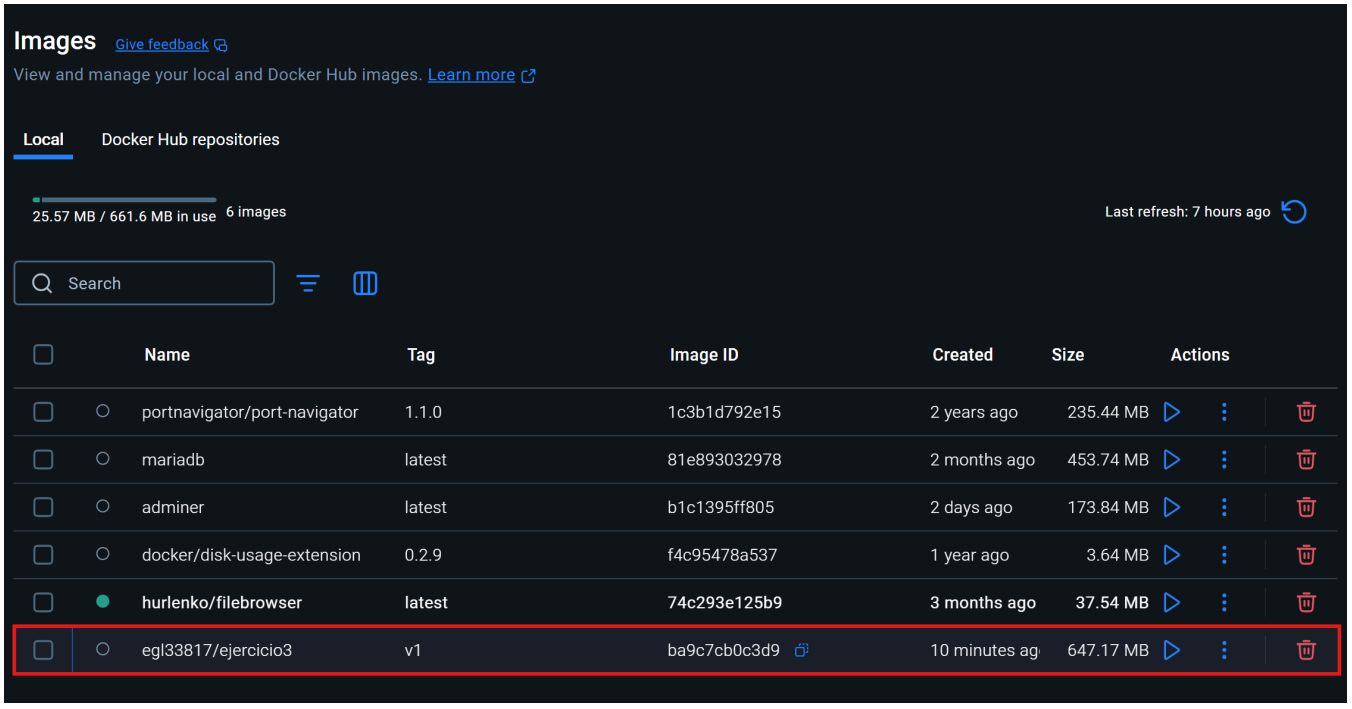
```
raber@SurfacePro9 MINGW64 /d/tareaDocker/ejercicio3 (ejercicio3)
$ docker build -t egl33817/ejercicio3:v1 .
#0 building with "desktop-linux" instance using docker driver

#1 [internal] load build definition from Dockerfile
#1 transferring dockerfile: 94B 0.0s done
#1 DONE 0.1s

#2 [internal] load metadata for docker.io/library/php:7.4-apache
#2 ...

#3 [auth] library/php:pull token for registry-1.docker.io
#3 DONE 0.0s

#2 [internal] load metadata for docker.io/library/php:7.4-apache
#2 DONE 2.3s

#4 [internal] load .dockerignore
#4 transferring context: 2B 0.0s done
#4 DONE 0.1s

#5 [internal] load build context
#5 transferring context: 1.15kB 0.0s done
#5 DONE 0.1s

#6 [1/3] FROM docker.io/library/php:7.4-apache@sha256:c9d7e608f73832673479770d66aacc8100011ec751d1905ff63fae3fe2e0ca6d
#6 resolve docker.io/library/php:7.4-apache@sha256:c9d7e608f73832673479770d66aacc8100011ec751d1905ff63fae3fe2e0ca6d 0.1s done
#6 DONE 0.6s

#6 [1/3] FROM docker.io/library/php:7.4-apache@sha256:c9d7e608f73832673479770d66aacc8100011ec751d1905ff63fae3fe2e0ca6d
#6 sha256:05e465aaa99a358add4acecdade8f39843089069f31fea0201533d3a09a98c9a 0B / 892B 0.2s
#6 sha256:05e465aaa99a358add4acecdade8f39843089069f31fea0201533d3a09a98c9a 892B / 892B 0.2s done
#6 sha256:80692ae2d067c8358112c56490a2a97f69ef395fd8f7662a31498644c9a813ef 246B / 246B 0.3s done
#6 sha256:ab590b48ea476386dd7b07c34de9eff7cf2103c4668ade985fe31e59f15deee8 0B / 2.46kB 0.2s
#6 sha256:d2c43c5efbc861f83ee6565c7102ca660d6f35e158324fbb042de5017e43afe8 0B / 10.20MB 0.2s
#6 sha256:ab590b48ea476386dd7b07c34de9eff7cf2103c4668ade985fe31e59f15deee8 2.46kB / 2.46kB 0.4s done
#6 sha256:66d98f73acb62e86c0c226f9eedcbc7eda305df0c1e171ca5caf81cb8b1c40cb 0B / 491B 0.2s
#6 sha256:66d98f73acb62e86c0c226f9eedcbc7eda305df0c1e171ca5caf81cb8b1c40cb 491B / 491B 0.3s done
#6 sha256:d14eb2ed1e17ae00f5fcb44b0d562e2867c401c20372829e2cf443fc409342fa 0B / 10.76MB 0.2s
#6 sha256:fe42347c4ecfc90333acd9cad13912387eea39d13827a25cfa78727fa5d200e9 514B / 514B 0.2s done
#6 sha256:9b233e420ac7bbca645bb82c213029762acf1742400c076360dc303213c309d5 475B / 475B 0.2s
#6 sha256:d2c43c5efbc861f83ee6565c7102ca660d6f35e158324fbb042de5017e43afe8 3.15MB / 10.20MB 0.6s
#6 sha256:d14eb2ed1e17ae00f5fcb44b0d562e2867c401c20372829e2cf443fc409342fa 2.10MB / 10.76MB 0.5s
#6 sha256:9b233e420ac7bbca645bb82c213029762acf1742400c076360dc303213c309d5 475B / 475B 0.2s done
#6 sha256:d2c43c5efbc861f83ee6565c7102ca660d6f35e158324fbb042de5017e43afe8 5.24MB / 10.20MB 0.8s
#6 sha256:d14eb2ed1e17ae00f5fcb44b0d562e2867c401c20372829e2cf443fc409342fa 4.19MB / 10.76MB 0.6s
#6 sha256:25f85b498fd5bfc6cce951513219fe480850daba71e6e997741e984d18483971 0B / 19.25MB 0.2s
#6 sha256:fb5a4c8af82f00730b7427e47bda7f76cea2e2b9aea421750bc9025aface98d8 270B / 270B 0.2s done
#6 sha256:d2c43c5efbc861f83ee6565c7102ca660d6f35e158324fbb042de5017e43afe8 7.34MB / 10.20MB 0.9s
#6 sha256:d14eb2ed1e17ae00f5fcb44b0d562e2867c401c20372829e2cf443fc409342fa 5.24MB / 10.76MB 0.8s
#6 sha256:25f85b498fd5bfc6cce951513219fe480850daba71e6e997741e984d18483971 1.05MB / 19.25MB 0.3s
#6 sha256:156740b07ef8a632f9f7bea4e57e4ee5541ade376adf9169351a1265382e39de 0B / 91.63MB 0.2s
#6 sha256:d2c43c5efbc861f83ee6565c7102ca660d6f35e158324fbb042de5017e43afe8 10.20MB / 10.20MB 1.1s done
```

```
#6 sha256:d2c43c5efbc861f83ee6565c7102ca660d6f35e158324fbb042de5017e43afe8 10.20MB / 10.20MB 1.1s done
#6 sha256:25f85b498fd5bfc6cce951513219fe480850daba71e6e997741e984d18483971 3.15MB / 19.25MB 0.5s
#6 sha256:d14eb2ed1e17ae00f5fcb44b0d562e2867c401c20372829e2cf443fc409342fa 7.34MB / 10.76MB 0.9s
#6 sha256:25f85b498fd5bfc6cce951513219fe480850daba71e6e997741e984d18483971 5.24MB / 19.25MB 0.6s
#6 sha256:d14eb2ed1e17ae00f5fcb44b0d562e2867c401c20372829e2cf443fc409342fa 10.76MB / 10.76MB 1.0s done
#6 sha256:25f85b498fd5bfc6cce951513219fe480850daba71e6e997741e984d18483971 8.39MB / 19.25MB 0.8s
#6 sha256:c428f1a494230852524a2a5957cc5199c36c8b403305e0e877d580bd0ec9e763 226B / 226B 0.2s done
#6 sha256:25f85b498fd5bfc6cce951513219fe480850daba71e6e997741e984d18483971 10.49MB / 19.25MB 0.9s
#6 sha256:a603fa5e3b4127f210503aaa6189abf6286ee5a73deeaab460f8f33ebc6b64e2 0B / 31.41MB 0.2s
#6 sha256:25f85b498fd5bfc6cce951513219fe480850daba71e6e997741e984d18483971 12.58MB / 19.25MB 1.1s
#6 sha256:156740b07ef8a632f9f7bea4e57e4ee5541ade376adf9169351a1265382e39de 5.24MB / 91.63MB 0.9s
#6 sha256:25f85b498fd5bfc6cce951513219fe480850daba71e6e997741e984d18483971 14.68MB / 19.25MB 1.2s
#6 sha256:a603fa5e3b4127f210503aaa6189abf6286ee5a73deeaab460f8f33ebc6b64e2 3.15MB / 31.41MB 0.5s
#6 sha256:25f85b498fd5bfc6cce951513219fe480850daba71e6e997741e984d18483971 16.78MB / 19.25MB 1.4s
#6 sha256:a603fa5e3b4127f210503aaa6189abf6286ee5a73deeaab460f8f33ebc6b64e2 5.24MB / 31.41MB 0.6s
#6 sha256:25f85b498fd5bfc6cce951513219fe480850daba71e6e997741e984d18483971 19.25MB / 19.25MB 1.6s done
#6 sha256:a603fa5e3b4127f210503aaa6189abf6286ee5a73deeaab460f8f33ebc6b64e2 7.34MB / 31.41MB 0.8s
#6 sha256:156740b07ef8a632f9f7bea4e57e4ee5541ade376adf9169351a1265382e39de 13.63MB / 91.63MB 1.5s
#6 sha256:a603fa5e3b4127f210503aaa6189abf6286ee5a73deeaab460f8f33ebc6b64e2 9.44MB / 31.41MB 0.9s
#6 sha256:a603fa5e3b4127f210503aaa6189abf6286ee5a73deeaab460f8f33ebc6b64e2 12.58MB / 31.41MB 1.1s
#6 sha256:156740b07ef8a632f9f7bea4e57e4ee5541ade376adf9169351a1265382e39de 18.87MB / 91.63MB 1.8s
#6 sha256:a603fa5e3b4127f210503aaa6189abf6286ee5a73deeaab460f8f33ebc6b64e2 15.73MB / 31.41MB 1.2s
#6 sha256:a603fa5e3b4127f210503aaa6189abf6286ee5a73deeaab460f8f33ebc6b64e2 17.83MB / 31.41MB 1.4s
#6 sha256:156740b07ef8a632f9f7bea4e57e4ee5541ade376adf9169351a1265382e39de 25.17MB / 91.63MB 2.1s
#6 sha256:a603fa5e3b4127f210503aaa6189abf6286ee5a73deeaab460f8f33ebc6b64e2 20.97MB / 31.41MB 1.5s
#6 sha256:a603fa5e3b4127f210503aaa6189abf6286ee5a73deeaab460f8f33ebc6b64e2 23.07MB / 31.41MB 1.7s
#6 sha256:156740b07ef8a632f9f7bea4e57e4ee5541ade376adf9169351a1265382e39de 30.41MB / 91.63MB 2.4s
#6 sha256:a603fa5e3b4127f210503aaa6189abf6286ee5a73deeaab460f8f33ebc6b64e2 26.21MB / 31.41MB 1.8s
#6 sha256:a603fa5e3b4127f210503aaa6189abf6286ee5a73deeaab460f8f33ebc6b64e2 29.36MB / 31.41MB 2.0s
#6 sha256:156740b07ef8a632f9f7bea4e57e4ee5541ade376adf9169351a1265382e39de 38.80MB / 91.63MB 2.7s
#6 sha256:a603fa5e3b4127f210503aaa6189abf6286ee5a73deeaab460f8f33ebc6b64e2 31.41MB / 31.41MB 2.1s done
#6 extracting sha256:a603fa5e3b4127f210503aaa6189abf6286ee5a73deeaab460f8f33ebc6b64e2
#6 sha256:156740b07ef8a632f9f7bea4e57e4ee5541ade376adf9169351a1265382e39de 48.23MB / 91.63MB 3.0s
#6 sha256:156740b07ef8a632f9f7bea4e57e4ee5541ade376adf9169351a1265382e39de 57.67MB / 91.63MB 3.3s
#6 sha256:156740b07ef8a632f9f7bea4e57e4ee5541ade376adf9169351a1265382e39de 67.11MB / 91.63MB 3.6s
#6 sha256:156740b07ef8a632f9f7bea4e57e4ee5541ade376adf9169351a1265382e39de 72.35MB / 91.63MB 3.8s
#6 sha256:156740b07ef8a632f9f7bea4e57e4ee5541ade376adf9169351a1265382e39de 81.79MB / 91.63MB 4.1s
#6 extracting sha256:a603fa5e3b4127f210503aaa6189abf6286ee5a73deeaab460f8f33ebc6b64e2 1.6s done
#6 sha256:156740b07ef8a632f9f7bea4e57e4ee5541ade376adf9169351a1265382e39de 87.03MB / 91.63MB 4.2s
#6 extracting sha256:c428f1a494230852524a2a5957cc5199c36c8b403305e0e877d580bd0ec9e763 0.0s done
#6 sha256:156740b07ef8a632f9f7bea4e57e4ee5541ade376adf9169351a1265382e39de 91.63MB / 91.63MB 4.4s
#6 sha256:156740b07ef8a632f9f7bea4e57e4ee5541ade376adf9169351a1265382e39de 91.63MB / 91.63MB 4.4s done
#6 extracting sha256:156740b07ef8a632f9f7bea4e57e4ee5541ade376adf9169351a1265382e39de
#6 extracting sha256:156740b07ef8a632f9f7bea4e57e4ee5541ade376adf9169351a1265382e39de 2.0s done
#6 DONE 7.8s

#6 [1/3] FROM docker.io/library/php:7.4-apache@sha256:c9d7e608f73832673479770d66aacc8100011ec751d1905ff63fae3fe2e0ca6d
#6 extracting sha256:fb5a4c8af82f00730b7427e47bda7f76cea2e2b9aea421750bc9025aface98d8 0.0s done
#6 extracting sha256:25f85b498fd5bfc6cce951513219fe480850daba71e6e997741e984d18483971
#6 extracting sha256:25f85b498fd5bfc6cce951513219fe480850daba71e6e997741e984d18483971 0.4s done
#6 DONE 8.2s

#6 [1/3] FROM docker.io/library/php:7.4-apache@sha256:c9d7e608f73832673479770d66aacc8100011ec751d1905ff63fae3fe2e0ca6d
#6 extracting sha256:9b233e420ac7bbca645bb82c213029762acf1742400c076360dc303213c309d5 0.0s done
#6 extracting sha256:fe42347c4ecfc90333acd9cad13912387eea39d13827a25cfa78727fa5d200e9 0.0s done
```

```
#6 extracting sha256:d14eb2ed1e17ae00f5fcb44b0d562e2867c401c20372829e2cf443fc409342fa 0.1s done
#6 DONE 8.3s

#6 [1/3] FROM docker.io/library/php:7.4-apache@sha256:c9d7e608f73832673479770d66aacc8100011ec751d1905ff63fae3fe2e0ca6d
#6 extracting sha256:66d98f73acb62e86c0c226f9eedcbc7eda305df0c1e171ca5caf81cb8b1c40cb 0.0s done
#6 extracting sha256:d2c43c5efbc861f83ee6565c7102ca660d6f35e158324fbb042de5017e43afe8
#6 extracting sha256:d2c43c5efbc861f83ee6565c7102ca660d6f35e158324fbb042de5017e43afe8 0.3s done
#6 DONE 8.7s

#6 [1/3] FROM docker.io/library/php:7.4-apache@sha256:c9d7e608f73832673479770d66aacc8100011ec751d1905ff63fae3fe2e0ca6d
#6 extracting sha256:ab590b48ea476386dd7b07c34de9eff7cf2103c4668ade985fe31e59f15deee8 0.0s done
#6 extracting sha256:80692ae2d067c8358112c56490a2a97f69ef395fd8f7662a31498644c9a813ef 0.0s done
#6 extracting sha256:05e465aaa99a358add4acecdade8f39843089069f31fea0201533d3a09a98c9a 0.0s done
#6 DONE 8.8s

#7 [2/3] WORKDIR /var/www/html
#7 DONE 1.6s

#8 [3/3] COPY ./miweb .
#8 DONE 0.1s

#9 exporting to image
#9 exporting layers
#9 exporting layers 0.2s done
#9 exporting manifest sha256:3f569b38b0052239169d3b1177bf681a09ebf6db8a4cdca7bda21cdae41e9669 0.0s done
#9 exporting config sha256:345854fccdc8da738759e60c6e88a8a67b5bc698f09845cd1b9dfe3bfa88b561 0.0s done
#9 exporting attestation manifest sha256:166eabdc32dd2c6fd028ee50b977e72fbb880316088d8166b97ea55cecf2e11f 0.0s done
#9 exporting manifest list sha256:ba9c7cb0c3d9f5e9d7826ebde16cf8f2ce2a6f05faad54df3666bafba942d2e7 0.0s done
#9 naming to docker.io/egl33817/ejercicio3:v1
#9 naming to docker.io/egl33817/ejercicio3:v1 done
#9 unpacking to docker.io/egl33817/ejercicio3:v1 0.1s done
#9 DONE 0.4s
```

En la última imagen podemos ver el detalle de la ejecución de las instrucciones que habíamos definido en el archivo `Dockerfile`.

Si ahora nos vamos al **Docker Desktop**, podemos ver que tenemos una nueva imagen cuyo nombre es el que pusimos en el comando `docker build`:

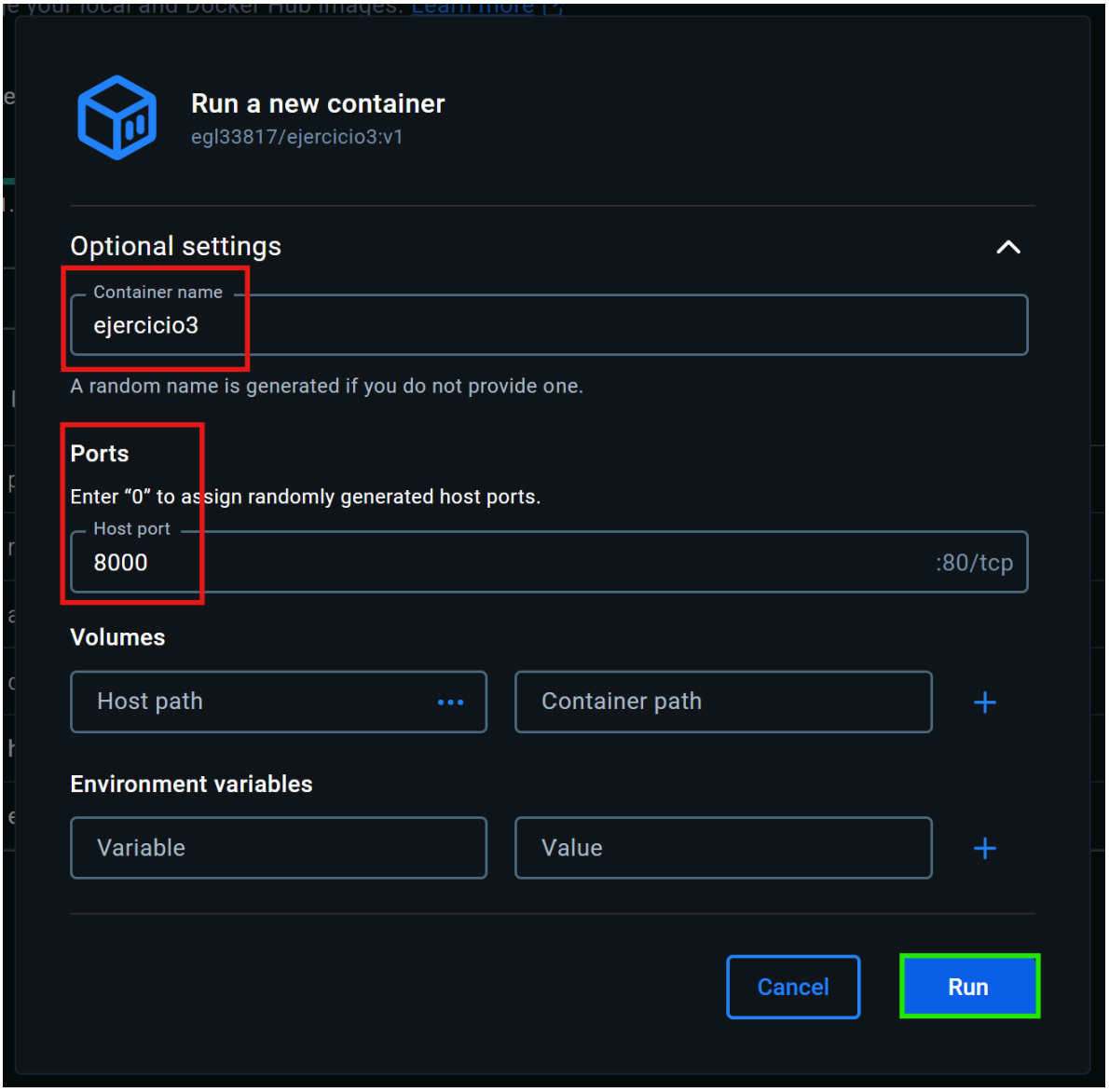


4. Creación de un contenedor a partir de la imagen generada

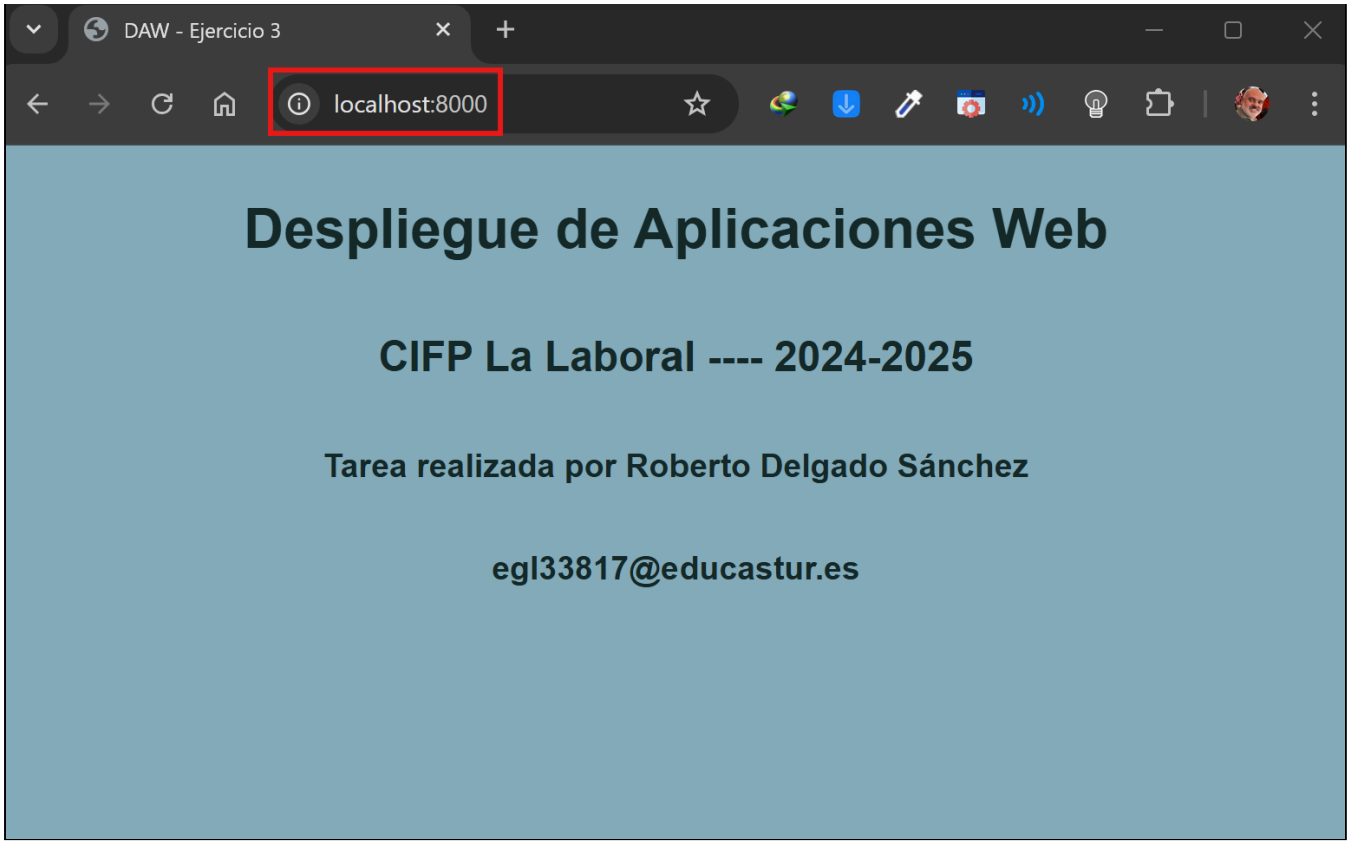
Vamos a probar la imagen creada generando un contenedor a partir de la misma con los siguientes parámetros que se nos especifican en el enunciado del ejercicio:

- debe llamarse `ejercicio3`.
- debe ser accesible desde un navegador en el puerto `8000`.

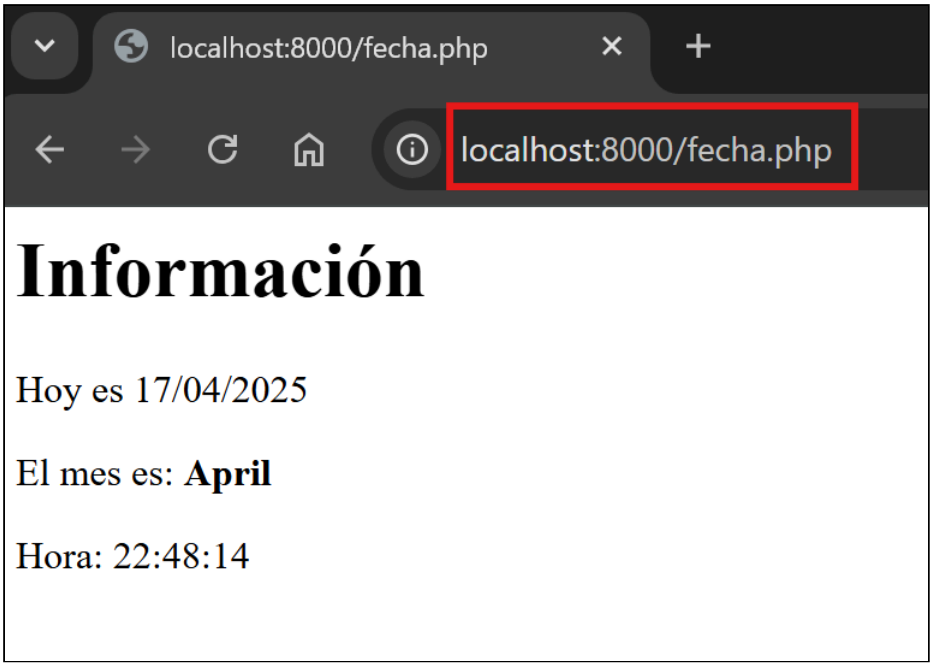
Creamos el contenedor y lo ponemos en ejecución:



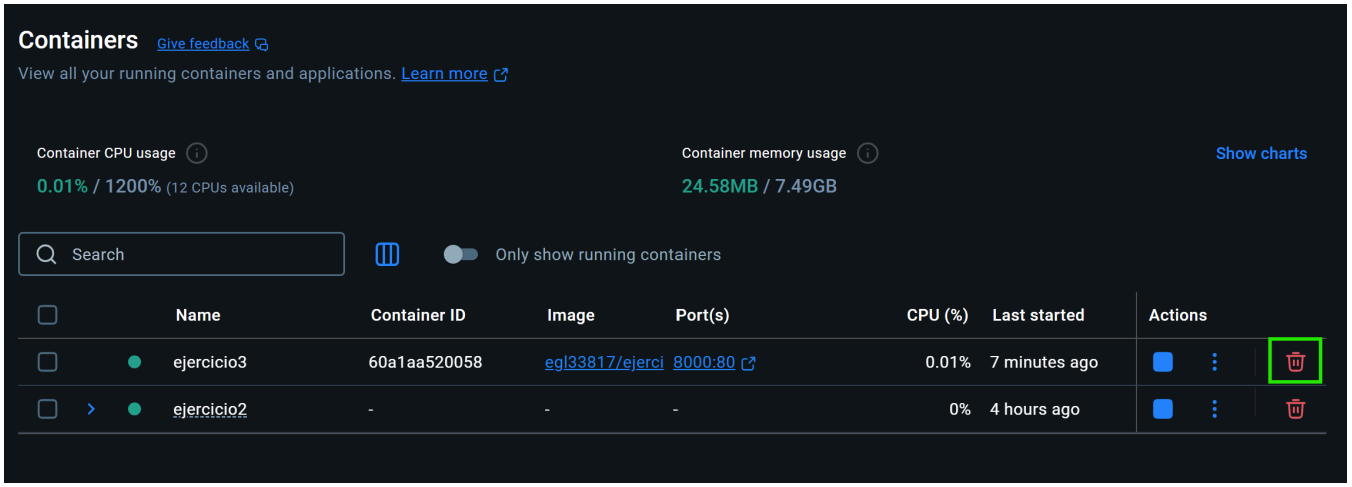
Una vez arrancado nos conectamos al mismo con un navegador:

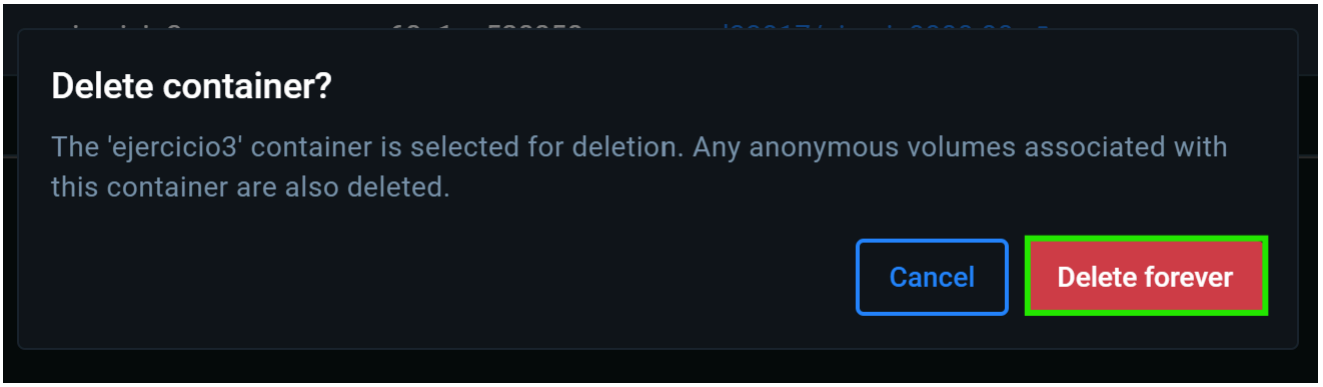


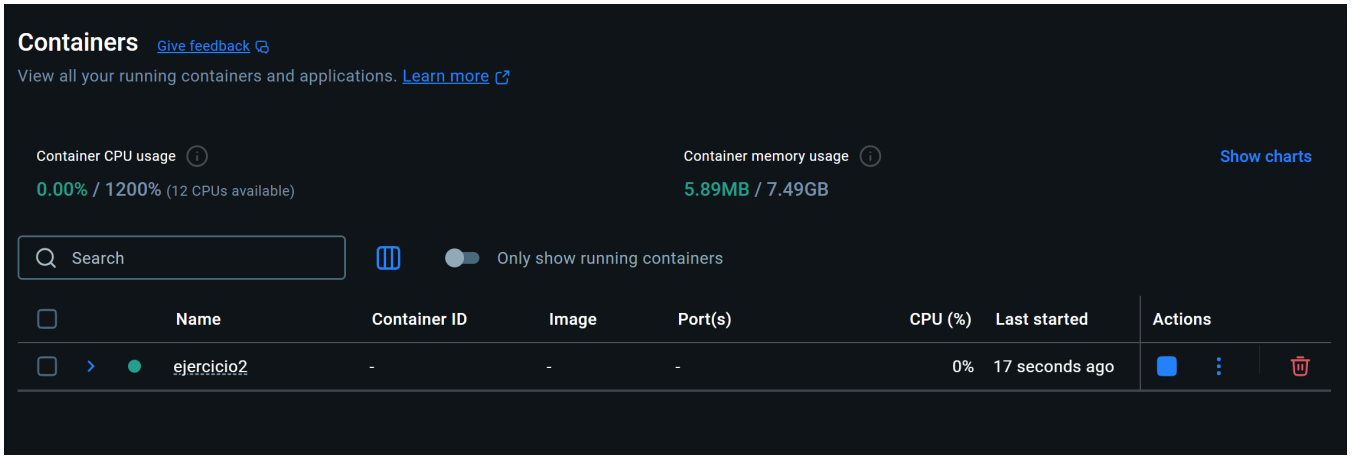
Y comprobamos que el script **PHP** se ejecuta correctamente:



Una vez comprobado que el contenedor funciona lo borramos:







5. Subir la imagen a Docker Hub

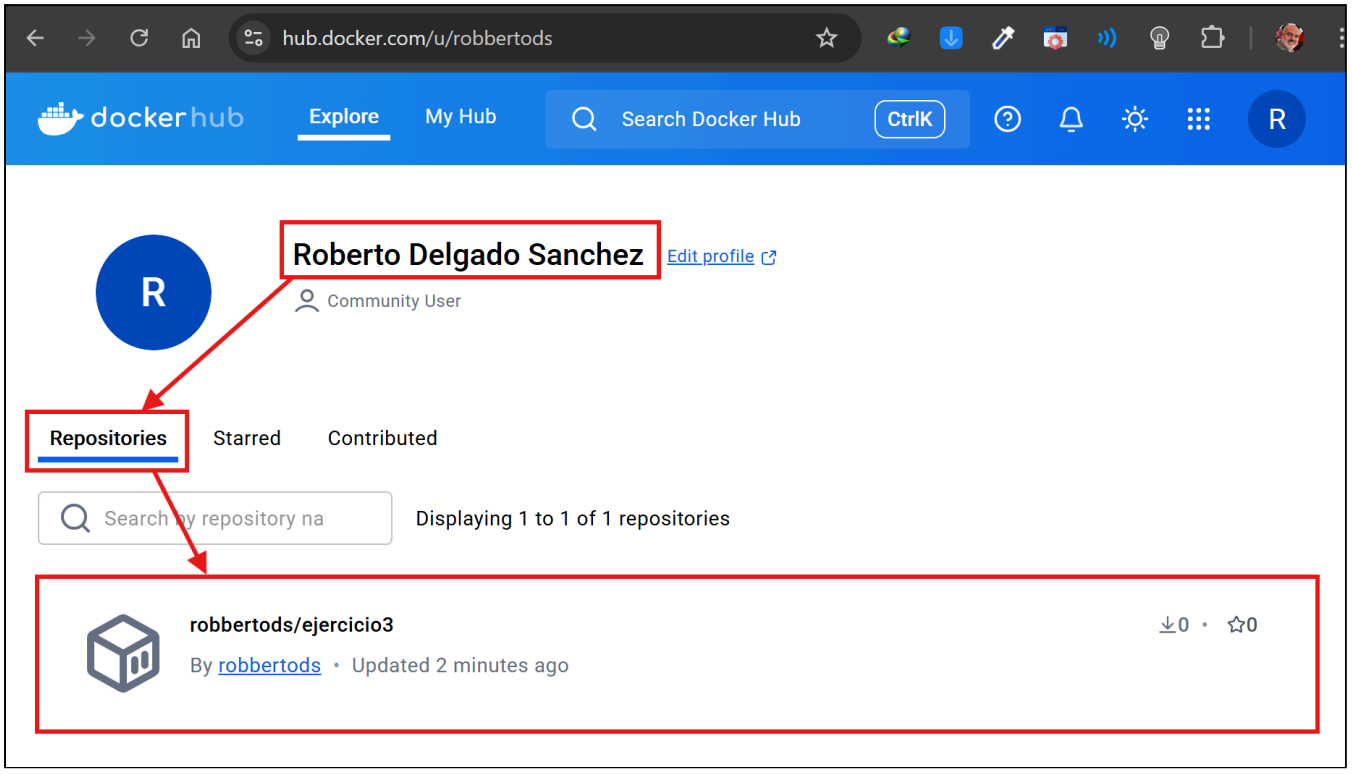
Antes de subir la imagen al repositorio he tenido que añadir una etiqueta a la misma, ya que al subirla me daba un error de permisos. Los comandos utilizados para añadir esa etiqueta y subirla a **Docker Hub** son:

```
$ docker tag ba9c7cb0c3d9f5e9d7826ebde16cf8f2ce2a6f05faad54df3666bafba942d2e7
robbertods/ejercicio3:version1.0
$ docker push robbertods/ejercicio3:version1.0
```

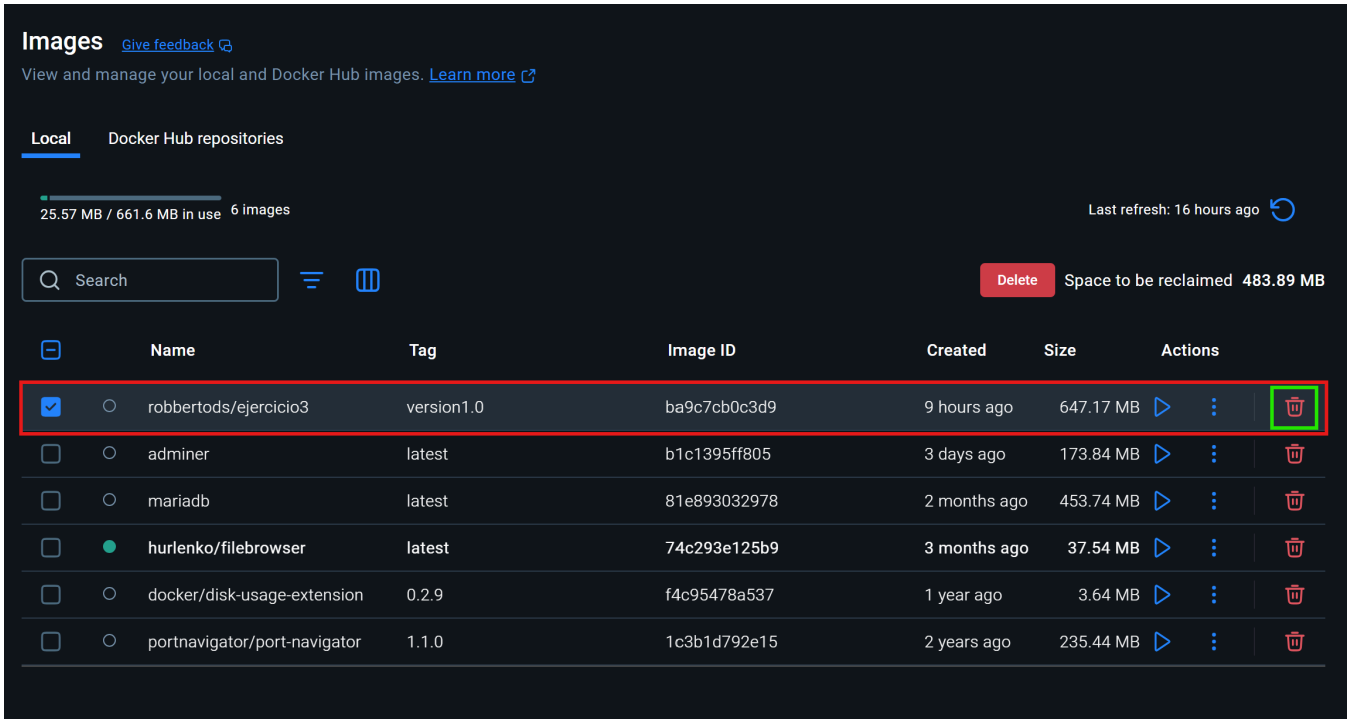
```
$ docker tag ba9c7cb0c3d9f5e9d7826ebde16cf8f2ce2a6f05faad54df3666bafba942d2e7 robbertods/ejercicio3:version1.0

$ docker push robbertods/ejercicio3:version1.0
The push refers to repository [docker.io/robbertods/ejercicio3]
fe42347c4ecf: Waiting
25f85b498fd5: Waiting
a603fa5e3b41: Waiting
ab590b48ea47: Waiting
66d98f73acb6: Waiting
```

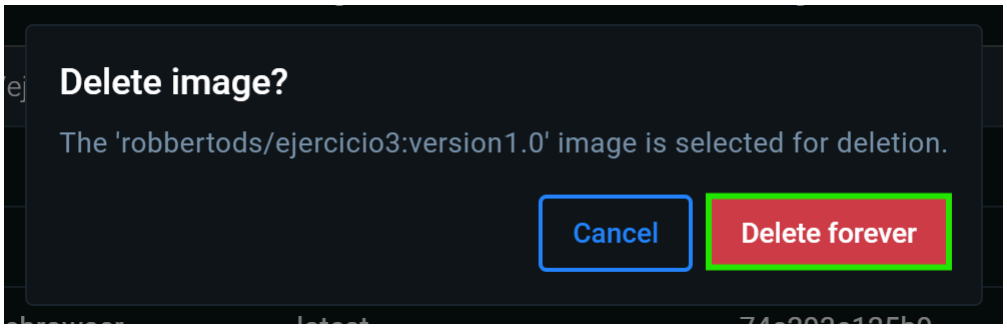
```
66d98f73acb6: Pushed
fe42347c4ecf: Pushed
ab590b48ea47: Pushed
fb5a4c8af82f: Pushed
c428f1a49423: Pushed
80692ae2d067: Pushed
9b233e420ac7: Pushed
bd6b7634fa55: Pushed
bd5c196d140f: Pushed
05e465aaa99a: Pushed
d2c43c5efbc8: Pushed
d14eb2ed1e17: Pushed
25f85b498fd5: Pushed
a603fa5e3b41: Pushed
156740b07ef8: Pushed
version1.0: digest: sha256:ba9c7cb0c3d9f5e9d7826ebde16cf8f2ce2a6f05faad54df3666bafba942d2e7 size: 856
```

Una vez finalizado el proceso, podemos ver la imagen en nuestro repositorio personal:

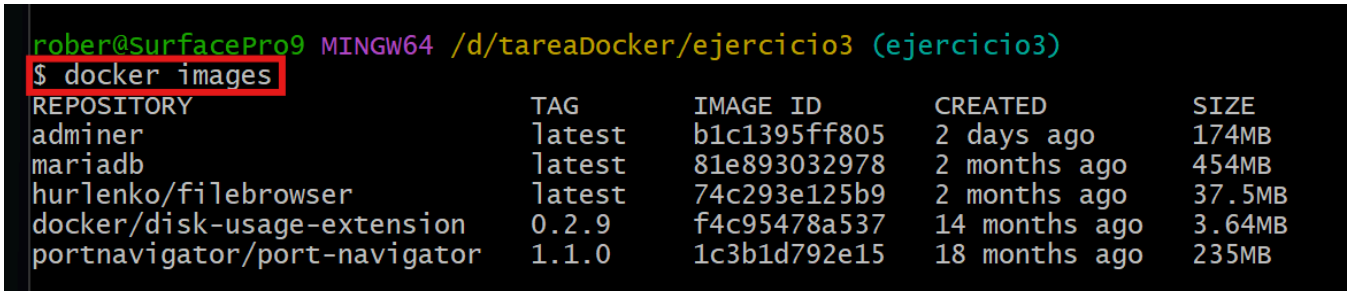


Ahora borramos la imagen que tenemos en nuestra máquina local desde **Docker Desktop**:

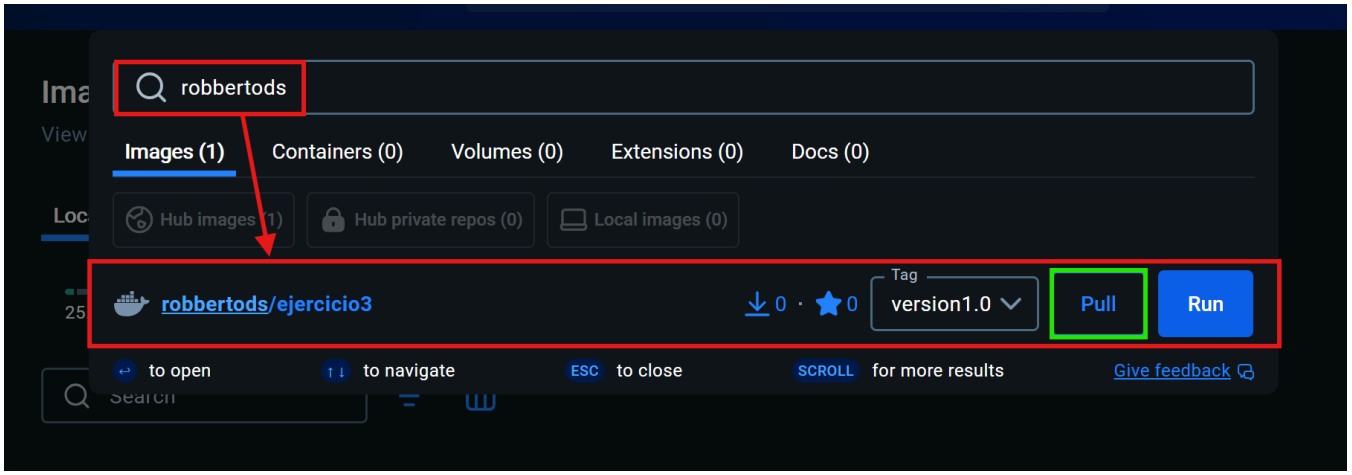


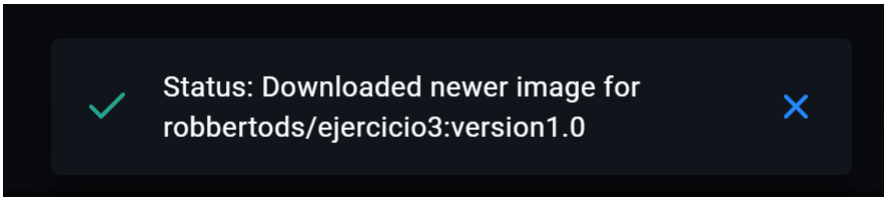


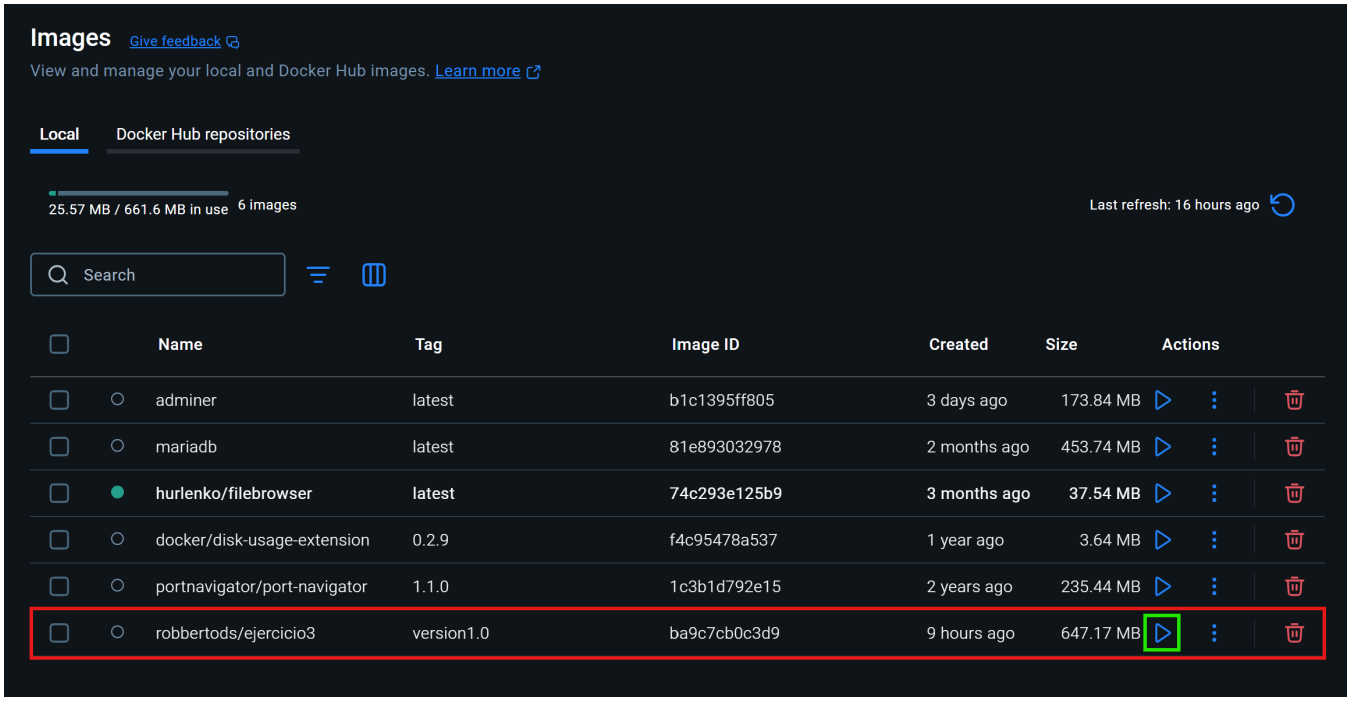
Comprobamos que ya no existe:



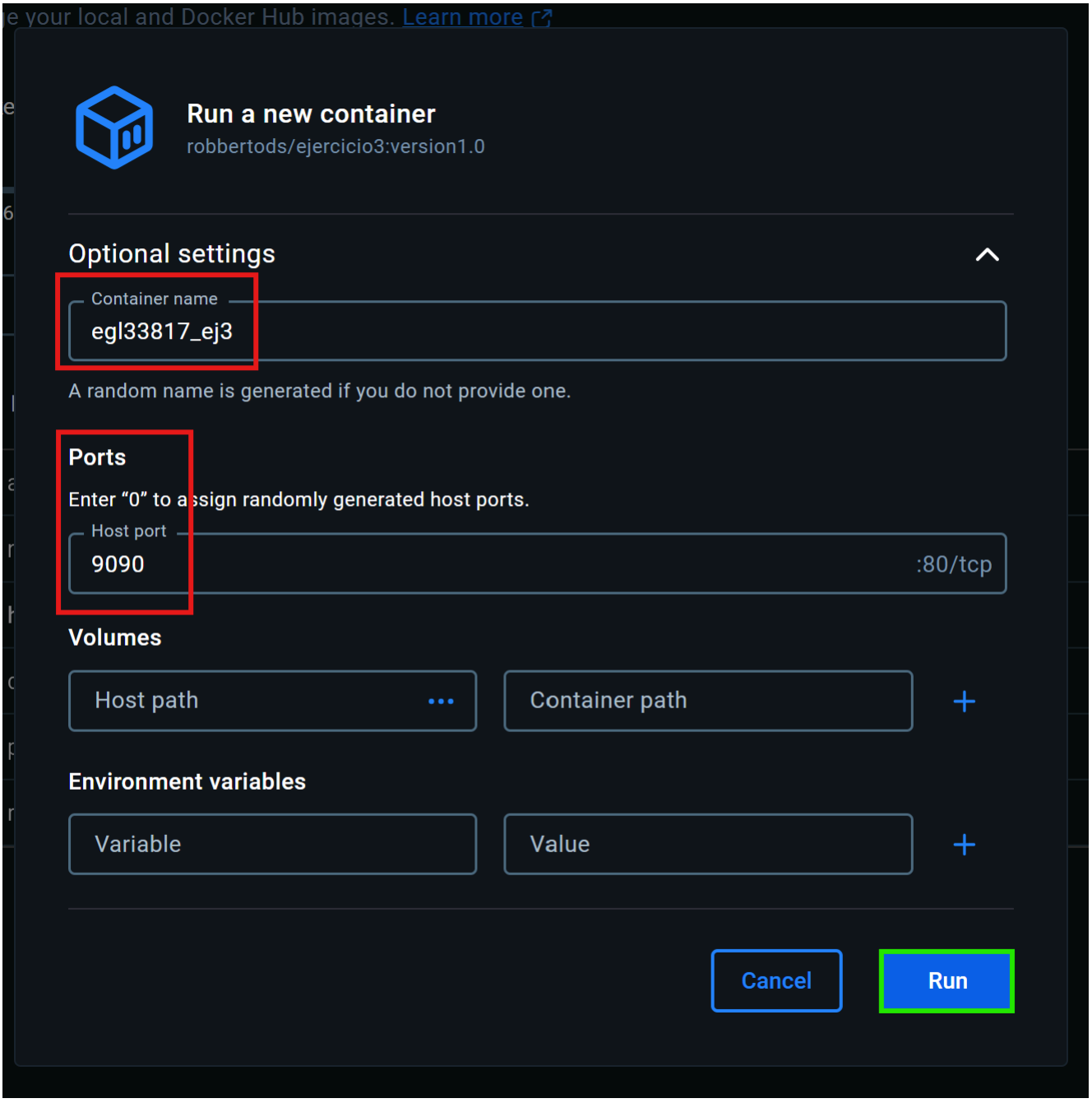
Y a continuación la descargamos desde Docker Desktop:



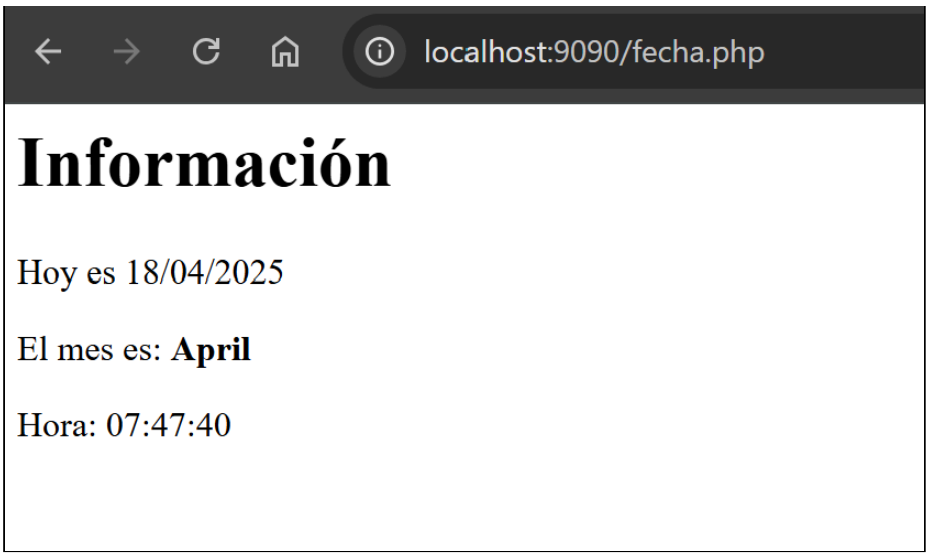




La ponemos en marcha en el puerto `9090` por ejemplo:



Y comprobamos que todo funciona correctamente:



Y así finaliza este tercer y último ejercicio de esta práctica.