

Actividad Evaluable 3 - Docker - EJERCICIO 3 - Imagen con Dockerfile

Tarea creada por Aida Montes Cabello para el módulo Despliegue de Aplicaciones Web.

Actividad Evaluable 3 - Docker - EJERCICIO 3 - Imagen con Dockerfile

Introducción

1. Creación estructura de archivos

Archivos `index.html`, `estilos.css` y `fecha.php`

Creación del archivo `Dockerfile`

2. Construcción de la imagen

3. Ejecución del contenedor

4. Verificación. Ver el sitio web en el navegador

5. Subida de la imagen a Docker Hub

6. Borrado de la imagen del Docker local

7. Operación `pull` de la imagen de Docker Hub

8. Creación y ejecución nuevo contenedor

Introducción

En este ejercicio lo que haremos será crear una imagen personalizada llamada `ejercicio3` con Dockerfile basada en `php:7.4-apache` y que será accesible desde el navegador en el puerto 8000. Esta imagen contendrá lo siguiente:

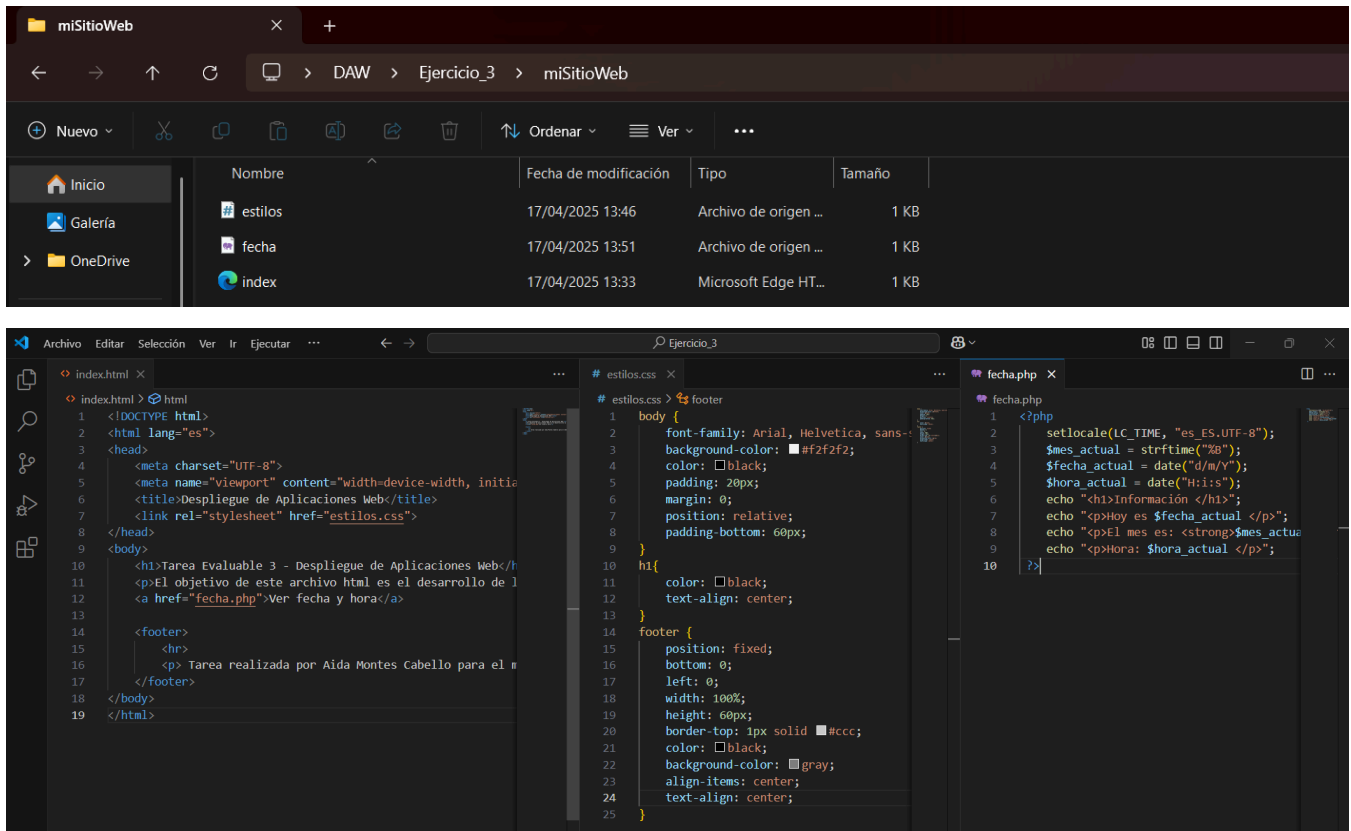
- Un sitio web en el que figurará nuestro nombre y tendrá dos archivos, un archivo `index.html` sencillo y un archivo `estilos.css`.
- Un script llamado `fecha.php`.

Posteriormente se subirá la imagen a nuestra cuenta de Docker Hub, la borraremos de nuestro Docker local, la volveremos a bajar y ejecutaremos un contenedor usando la imagen.

1. Creación estructura de archivos

Creamos un directorio llamado `Ejercicio_3` y, dentro de este, otro con el nombre `misitioweb`. En su interior, creamos los archivos `index.html` (con nuestro nombre y un enlace), `estilos.css` (para el diseño visual) y `fecha.php` (script).

Archivos `index.html`, `estilos.css` y `fecha.php`

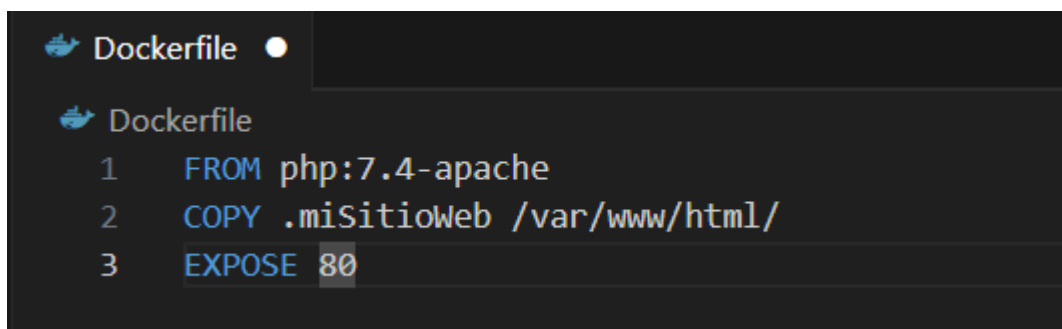


Creación del archivo `Dockerfile`

Este archivo le indicará a Docker cómo debe construir la imagen personalizada, es importante que el archivo no tenga extensión y se llame `Dockerfile`. El archivo debe contener lo siguiente:

```
FROM php:7.4-apache
COPY .misitioweb /var/www/html
EXPOSE 80
```

- Imagen base usando PHP con Apache (`php:7.4-apache`)
- Se copian al directorio raíz del servidor todos los archivos del proyecto del directorio `misitioweb`.
- Por defecto en Apache, se expone el puerto 80.



2. Construcción de la imagen

Para la creación de la imagen personalizada, abrimos la terminal integrada en Docker Desktop y nos situamos en la ruta donde está la carpeta `misitioweb`.

Terminal

```
PS C:\Users\Usuario\Desktop\DAW\Ejercicio_2> cd C:\Users\Usuario\Desktop\DAW\Ejercicio_3\miSitioWeb
PS C:\Users\Usuario\Desktop\DAW\Ejercicio_3\miSitioWeb> |
```

Una vez en la ruta, ejecutaremos el siguiente código para la construcción de la imagen:

```
docker build -t aidamontes/ejercicio3:v1 .
```

- `docker build`: este comando inicia la construcción de la imagen.
- `-t`: da un nombre y etiqueta a la imagen.
- `aidamontes/ejercicio3:v1`: nombre completo de la imagen. `aidamontes` es el nombre de usuario y lo utilizaremos para después poder subirla, `ejercicio3` es el nombre descriptivo de la imagen y `v1` sería la versión de la imagen.

Terminal

```
PS C:\Users\Usuario\Desktop\DAW\Ejercicio_3\miSitioWeb> docker build -t aidamontes/ejercicio3:v1 .
[+] Building 16.3s (8/8) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 90B
=> [internal] load metadata for docker.io/library/php:7.4-apache
=> [auth] library/php:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load build context
=> => transferring context: 1.68kB
=> [1/2] FROM docker.io/library/php:7.4-apache@sha256:c9d7e608f73832673479770d66aacc8100011ec751d1905ff63fae3fe2e0ca6d
=> => resolve docker.io/library/php:7.4-apache@sha256:c9d7e608f73832673479770d66aacc8100011ec751d1905ff63fae3fe2e0ca6d
=> => sha256:fe42347c4ecfc90333acd9cad13912387eea39d13827a25cfa78727fa5d200e9 514B / 514B
```

```
=> => sha256:80692ae2d067c8358112c56490a2a97f69ef395fd8f7662a31498644c9a813ef 246B / 246B
=> => sha256:d2c43c5efbc861f83ee6565c7102ca660d6f35e158324fbb042de5017e43afe8 10.20MB / 10.20MB
=> => sha256:d14eb2ed1e17ae00f5fcb44b0d562e2867c401c20372829e2cf443fc409342fa 10.76MB / 10.76MB
=> => sha256:66d98f73acb62e86c0c226f9eedcbc7eda305df0c1e171ca5caf81cb8b1c40cb 491B / 491B
=> => sha256:9b233e420ac7bbca645bb82c213029762acf1742400c076360dc303213c309d5 475B / 475B
=> => sha256:05e465aaa99a358add4acecdade8f39843089069f31fea0201533d3a09a98c9a 892B / 892B
=> => sha256:ab590b48ea476386dd7b07c34de9eff7cf2103c4668ade985fe31e59f15deee8 2.46kB / 2.46kB
=> => sha256:fb5a4c8af82f00730b7427e47bda7f76cea2e2b9aea421750bc9025aface98d8 270B / 270B
=> => sha256:25f85b498fd5bfc6cce951513219fe480850daba71e6e997741e984d18483971 19.25MB / 19.25MB
=> => sha256:156740b07ef8a632f9f7bea4e57e4ee5541ade376adf9169351a1265382e39de 91.63MB / 91.63MB
=> => sha256:c428f1a494230852524a2a5957cc5199c36c8b403305e0e877d580bd0ec9e763 226B / 226B
=> => sha256:a603fa5e3b4127f210503aaa6189abf6286ee5a73deeaab460f8f33ebc6b64e2 31.41MB / 31.41MB
=> => extracting sha256:a603fa5e3b4127f210503aaa6189abf6286ee5a73deeaab460f8f33ebc6b64e2
```

```
=> => extracting sha256:c428f1a494230852524a2a5957cc5199c36c8b403305e0e877d580bd0ec9e763 0.0s
=> => extracting sha256:156740b07ef8a632f9f7bea4e57e4ee5541ade376adf9169351a1265382e39de 2.7s
=> => extracting sha256:fb5a4c8af82f00730b7427e47bda7f76cea2e2b9aea421750bc9025aface98d8 0.0s
=> => extracting sha256:25f85b498fd5bfc6cce951513219fe480850daba71e6e997741e984d18483971 0.0s
=> => extracting sha256:9b233e420ac7bbca645bb82c213029762acf1742400c076360dc303213c309d5 0.5s
=> => extracting sha256:fe42347c4ecfc90333acd9cad13912387eea39d13827a25cfa78727fa5d200e9 0.0s
=> => extracting sha256:d14eb2ed1e17ae00f5fcb44b0d562e2867c401c20372829e2cf443fc409342fa 0.1s
=> => extracting sha256:66d98f73acb62e86c0c226f9eedcbc7eda305df0c1e171ca5caf81cb8b1c40cb 0.0s
=> => extracting sha256:d2c43c5efbc861f83ee6565c7102ca660d6f35e158324fbb042de5017e43afe8 0.4s
=> => extracting sha256:ab590b48ea476386dd7b07c34de9eff7cf2103c4668ade985fe31e59f15deee8 0.0s
=> => extracting sha256:80692ae2d067c8358112c56490a2a97f69ef395fd8f7662a31498644c9a813ef 0.0s
=> => exporting layers 0.1s
=> => exporting manifest sha256:f444b844698ab7deb52cbc8177f54a06a66d1d74319ac05ce3eb122d30cd6b32 0.0s
```

```
=> => exporting config sha256:7882edfe3b5ddac52797c122c61b902818ddf9ea2ae5a267d1b006404431361c 0.0s
=> => exporting attestation manifest sha256:198e6bc7d3364567ce5b203f78195ddc6e7a9298aad8ae925bafa04cbbc7e6fc 0.0s
=> => exporting manifest list sha256:6e9540b3de07fb85789aa5f81e936d5a336c96bd5b093dba48545ec5ad8961c 0.0s
=> => naming to docker.io/aidamontes/ejercicio3:v1 0.0s
=> => unpacking to docker.io/aidamontes/ejercicio3:v1 0.1s
```

View build details: [docker-desktop://dashboard/build/desktop-linux/desktop-linux/lfnrvyvk4co6hvj681wmqu33q](https://dashboard/build/desktop-linux/desktop-linux/lfnrvyvk4co6hvj681wmqu33q)

Comprobamos que la imagen se ha creado correctamente utilizando el código expuesto a continuación, el cual muestra todas las imágenes de Docker que se encuentran almacenadas de manera local en nuestra máquina.

```
docker images
```

Terminal

```
PS C:\Users\Usuario\Desktop\DAW\Ejercicio_3\miSitioWeb> docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
aidamontes/ejercicio3	v1	6e9540b3de07	9 minutes ago	647MB
hurlenko/filebrowser	latest	74c293e125b9	2 months ago	37.5MB
docker/disk-usage-extension	0.2.9	f4c95478a537	14 months ago	3.64MB
docker/welcome-to-docker	latest	eedaff45e3c7	17 months ago	29.5MB
portnavigator/port-navigator	1.1.0	1c3b1d792e15	18 months ago	235MB

3. Ejecución del contenedor

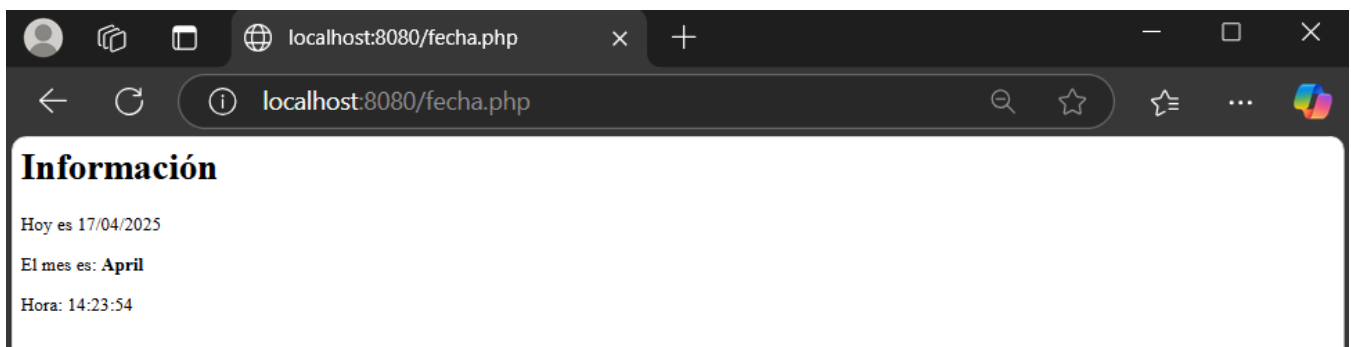
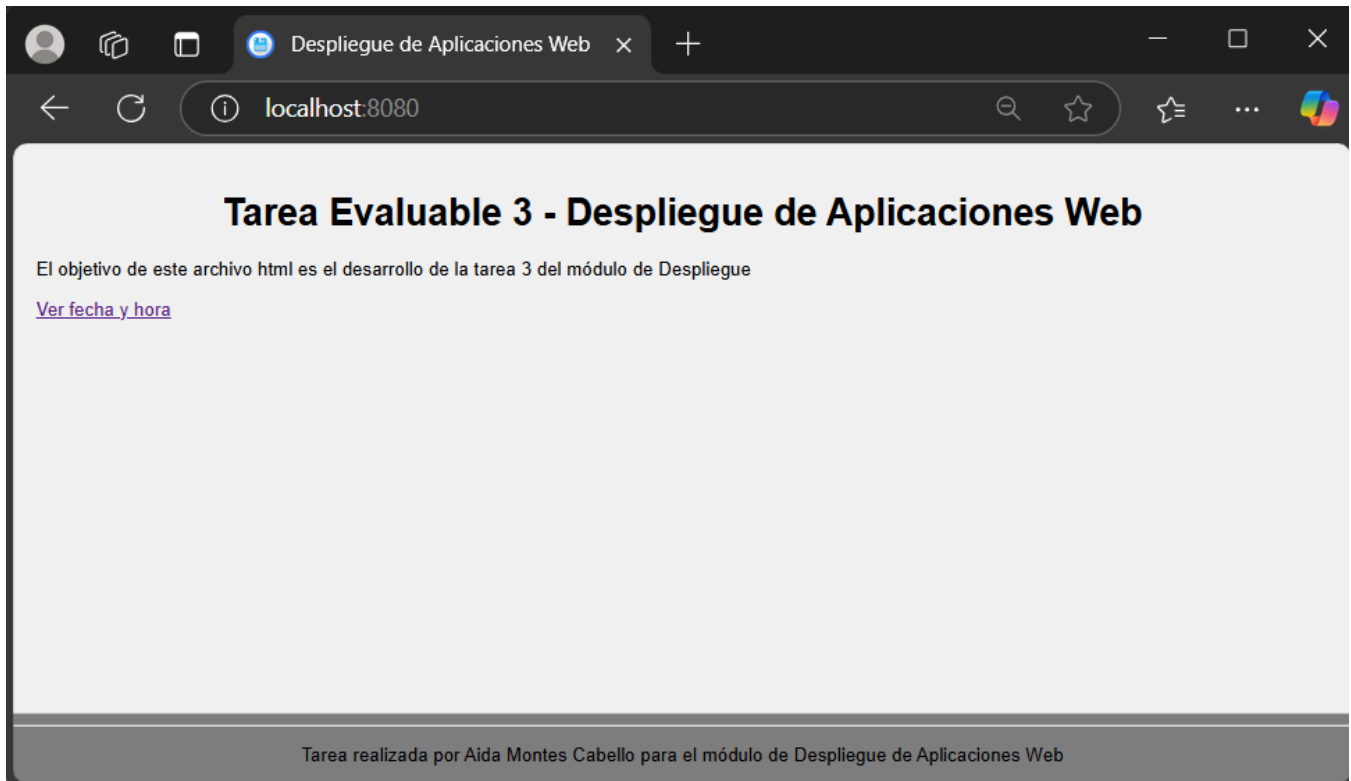
Lo primero que tenemos que hacer es crear el contenedor, para ello ejecutamos:

```
docker run -d -p 8080:80 --name ejercicio3 aidamontes/ejercicio3:v1
```

- `-d`: ejecuta el contenedor en segundo plano.
- `-p 8080:80`: redirige el puerto 80 del contenedor al 8080 de nuestra máquina, más adelante, accederemos desde el navegador con `localhost:8080`.
- `--name ejercicio3`: nombre que se le asigna al contenedor.
- `aidamontes/ejercicio3:v1`: imagen anteriormente creada.

4. Verificación. Ver el sitio web en el navegador

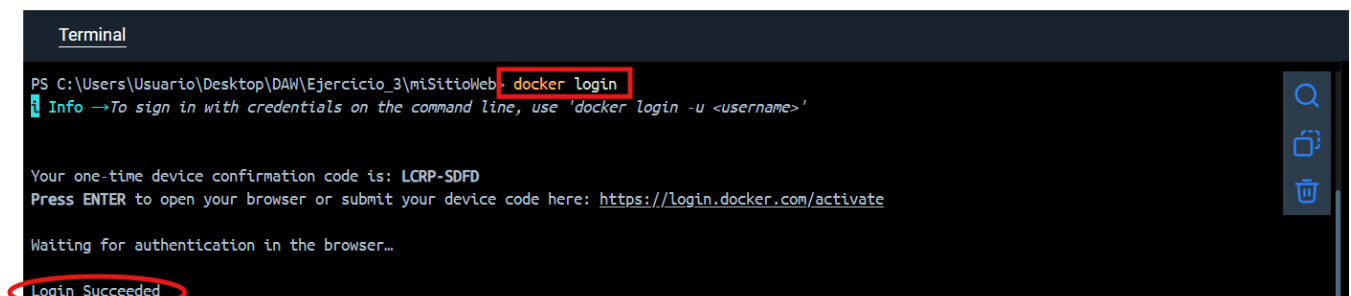
Para verificar que los pasos anteriores se han hecho de manera satisfactoria, vamos a acceder a `http://localhost:8080` desde nuestro navegador.



5. Subida de la imagen a Docker Hub

Lo primero que tenemos que hacer para subir la imagen es iniciar sesión en Docker Hub desde la terminal.

```
docker login
```



Vemos que el inicio de sesión ha sido exitoso, que se ha conectado correctamente a Docker Hub.

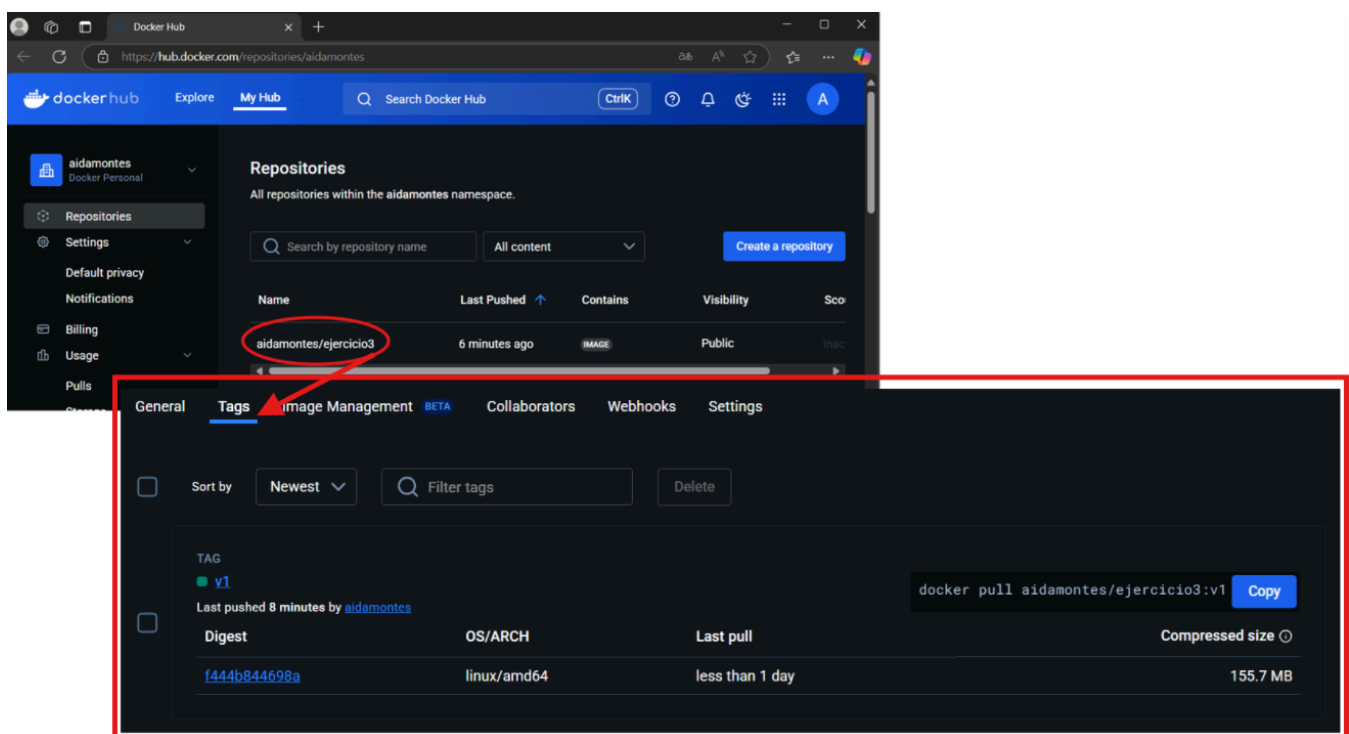
Ahora nos tocaría subir la imagen a Docker Hub haciendo un `push`.

```
docker push aidamontes/ejercicio3:v1
```

```
PS C:\Users\Usuario> docker push aidamontes/ejercicio3:v1
The push refers to repository [docker.io/aidamontes/ejercicio3]
9b233e420ac7: Layer already exists
156740b07ef8: Layer already exists
d2c43c5efbc8: Layer already exists
c428f1a49423: Layer already exists
fb5a4c8af82f: Layer already exists
25f85b498fd5: Layer already exists
ab590b48ea47: Layer already exists
80692ae2d067: Layer already exists
fe42347c4ecf: Layer already exists
7659e09d7b5f: Layer already exists
05e465aaa99a: Layer already exists
66d98f73acb6: Layer already exists
06291d61f2fe: Already exists
a603fa5e3b41: Layer already exists
d14eb2ed1e17: Layer already exists
v1: digest: sha256:6e9540b3de07fb85789aa5f81e936d5a3366c96bd5b093dba48545ec5ad8961c size: 856
```

En este caso en la imagen adjuntada, podemos ver que nos dice que ya existe porque ejecutamos el código, pero se nos cerró la terminal y no pudimos hacer captura de pantalla. Al ejecutarlo de nuevo, nos pone `Layer already exists / Already exists`, en un primer momento, cuando lo ejecutamos la primera vez nos traía `Pushed`.

Veamos si se ha subido correctamente, para ello iremos a `Repositories` de nuestro Docker Hub y pulsaremos sobre `aidamontes/ejercicio3`. Una vez dentro, iremos a la pestaña `Tags` y ahí podremos observar la etiqueta que hemos subido (`v1`), información sobre el tamaño y fecha en la que se subió entre otras cosas.



The screenshot shows the Docker Hub interface for the repository `aidamontes/ejercicio3`. The `Tags` tab is selected, displaying a list of tags. The tag `v1` is highlighted, showing its details:

Tag	Digest	OS/ARCH	Last pull	Compressed size
<code>v1</code>	<code>444b844698a</code>	<code>linux/amd64</code>	less than 1 day	155.7 MB

6. Borrado de la imagen del Docker local

Para eliminar la imagen local, iremos a la terminal y ejecutaremos:

```
docker rmi aidamontes/ejercicio3:v1
```

```
PS C:\Users\Usuario\Desktop\DAW\Ejercicio_3\miSitioWeb> docker rmi aidamontes/ejercicio3:v1
Untagged: aidamontes/ejercicio3:v1
Deleted: sha256:6e9540b3de07fb85789aa5f81e936d5a3366c96bd5b093dba48545ec5ad8961c
```

Comprobamos que se ha eliminado correctamente:

```
docker images
```

Terminal

```
PS C:\Users\Usuario\Desktop\DAW\Ejercicio_3\miSitioWeb> docker images
REPOSITORY              TAG          IMAGE ID      CREATED        SIZE
hurlenko/filebrowser    latest       74c293e125b9  2 months ago  37.5MB
docker/disk-usage-extension 0.2.9       f4c95478a537  14 months ago 3.64MB
docker/welcome-to-docker latest       eedaff45e3c7  17 months ago 29.5MB
portnavigator/port-navigator 1.1.0       1c3b1d792e15  18 months ago 235MB
```

7. Operación Pull de la imagen de Docker Hub

El comando Pull traerá la imagen a nuestro entorno local desde Docker Hub, para ello ejecutamos:

```
docker pull aidamontes/ejercicio3:v1
```

Terminal

```
PS C:\Users\Usuario\Desktop\DAW\Ejercicio_3\miSitioWeb> docker pull aidamontes/ejercicio3:v1
v1: Pulling from aidamontes/ejercicio3
7659e09d7b5f: Pull complete
Digest: sha256:6e9540b3de07fb85789aa5f81e936d5a3366c96bd5b093dba48545ec5ad8961c
Status: Downloaded newer image for aidamontes/ejercicio3:v1
docker.io/aidamontes/ejercicio3:v1
```

8. Creación y ejecución nuevo contenedor

En este apartado generaremos un contenedor a partir de la imagen creada y lo lanzaremos especificando un puerto diferente. Por lo tanto mapearemos el puerto 80 interno del contenedor al puerto 1234 del host, para poder acceder más adelante desde el navegador con `http://localhost:1234`. Ejecutamos para ello el siguiente comando:

```
docker run -d --name ejercicio3_v2 -p 1234:80 aidamontes/ejercicio3:v1
```

```
PS C:\Users\Usuario\Desktop\DAW\Ejercicio_3\miSitioWeb> docker run -d --name ejercicio3_v2 -p 1234:80 aidamontes/ejercicio3:v1
aea0356b5e49a5a36302ae2096f088d8825a6cdaef790c2a7f691e0ff9cf3487
```

Por último vamos a comprobar que accediendo a `http://localhost:1234` vemos nuestra página correctamente.

