

Ejercicio 3 - imagen con Dockerfile - Aplicación web

Documento que resuelve el ejercicio 3 de la Tarea 3- Despliegue de Aplicaciones Web

Ejercicio 3 - imagen con Dockerfile - Aplicación web

Bloque de código con el Dockerfile

Creación de la nueva imagen

Creación del contenedor inicial

Acceso al navegador con la página `html` y con el script `php`

Subir la imagen creada a mi cuenta en Docker Hub

Para la realización de este ejercicio es necesario tener una cuenta en Docker Hub.

Se puede resolver utilizando comandos, o Docker Desktop, o combinando ambos.

Enunciado:

Necesitamos un fichero **Dockerfile** que automatice las siguientes operaciones para crear una imagen que contenga un servidor con un sitio web y un script php. Características de la imagen:

- Usa un contenedor que ejecute una instancia de la imagen `php:7.4-apache` que se llame `ejercicio3` y que sea accesible desde un navegador en el puerto 8000.
- Coloca en el directorio raíz del servicio web (`/var/www/html`) un "sitio web" donde figure tu nombre -el sitio deberá tener al menos un archivo `index.html` sencillo y un archivo `.css`
- Coloca en ese mismo directorio raíz el siguiente script `php`, llámalo `fecha.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>";
?>
```

- Ver la salida del script `fecha.php` y de la página `index.html` en el navegador.
- Una vez creada la imagen, súbela a tu cuenta de Docker Hub:
 - Borra la imagen de tu Docker local.
 - Baja ('pull') de tu cuenta la imagen que acabas de subir.
 - Muestra las imágenes que tienes.
 - Ejecuta un contenedor usando esa imagen.

Resolución:

Bloque de código con el Dockerfile

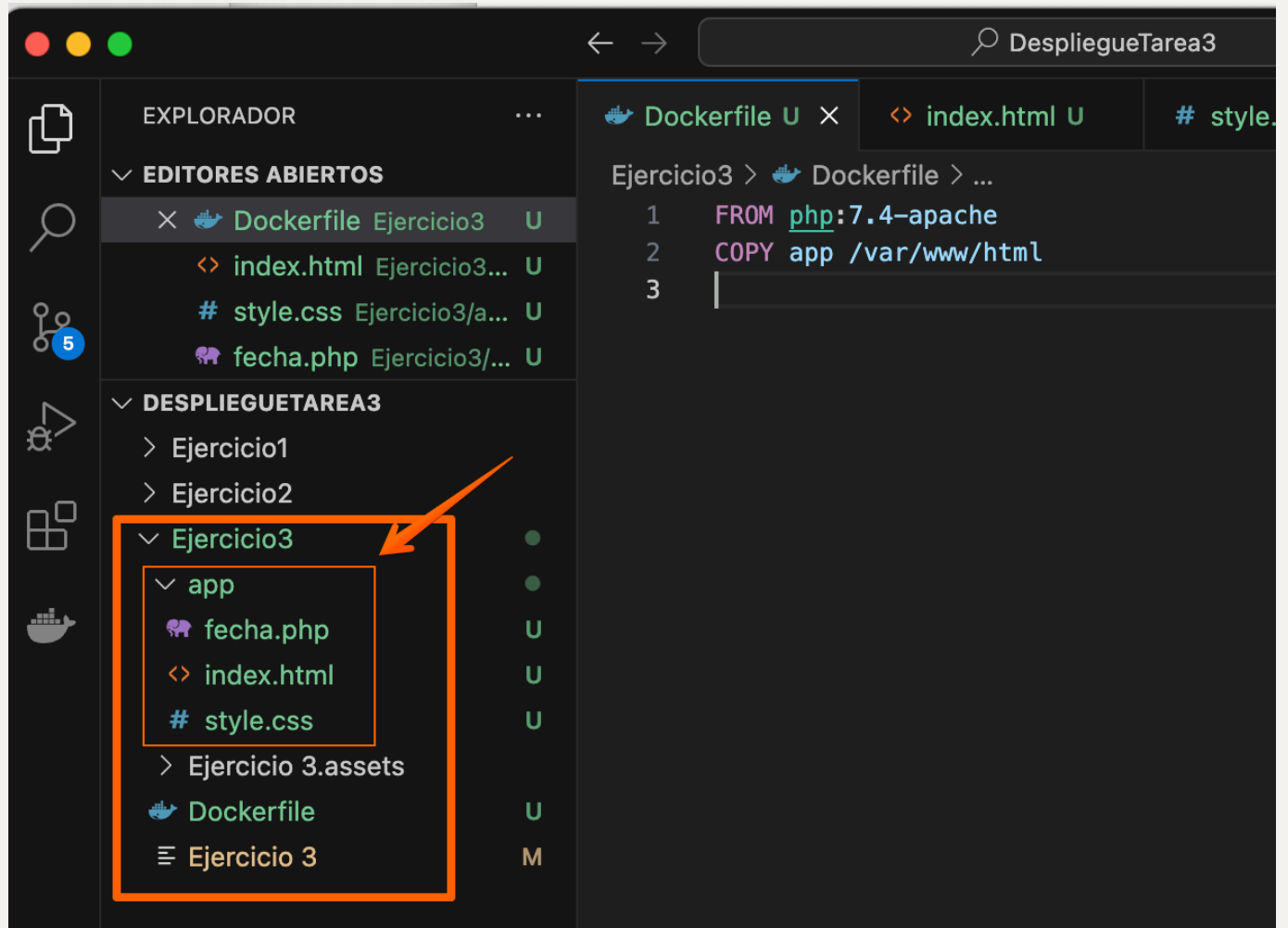
La imagen `php:7.4-apache` tiene ya instalado el servidor web y PHP.

Además, no es necesario indicar el `CMD` ya que por defecto el contenedor creado a partir de esta imagen ejecutará el mismo proceso que la imagen base, es decir, la ejecución del servidor web.

El código es:

```
FROM php:7.4-apache
COPY app /var/www/html
```

En `app/` están colgados los archivos `index.html`, `style.css` y `fecha.php`, de forma que el contenido de ese directorio se copiará al `/var/www/html/`



Creación de la nueva imagen

La nueva imagen sería:

```
docker build -t pfany/newImage .
```

Explicación:

- `docker build` -> inicia la construcción de la imagen.
- `-t pfany/newImage .` -> indica el nombre de la imagen y que se cree en el directorio actual donde está el archivo Dockerfile (en `Ejercicio3`).

Las capturas del proceso:

```
Ejercicio3 -- -zsh -- 125x29

fanyperalta@MacBook-Air-de-Fany Ejercicio3 % ls
Dockerfile      Ejercicio 3      Ejercicio 3.assets  app
fanyperalta@MacBook-Air-de-Fany Ejercicio3 % docker build -t pfany/newImage .
[+] Building 0.0s (0/0)                                docker:desktop-linux
ERROR: invalid tag "pfany/newImage": repository name must be lowercase
fanyperalta@MacBook-Air-de-Fany Ejercicio3 % docker build -t pfany/new-image .
[+] Building 7.8s (8/8) FINISHED                        docker:desktop-linux
=> [internal] load build definition from Dockerfile      0.0s
=> => transferring dockerfile: 117B                      0.0s
=> [internal] load metadata for docker.io/library/php:7.4-apache 2.2s
=> [auth] library/php:pull token for registry-1.docker.io 0.0s
=> [internal] load .dockerignore                        0.0s
=> => transferring context: 2B                          0.0s
=> [internal] load build context                        0.0s
=> => transferring context: 1.13kB                       0.0s
=> [1/2] FROM docker.io/library/php:7.4-apache@sha256:c9d7e608f73832673479770d66aacc8100011ec751d1905ff63fae3fe2e0ca6 5.3s
=> => resolve docker.io/library/php:7.4-apache@sha256:c9d7e608f73832673479770d66aacc8100011ec751d1905ff63fae3fe2e0ca6 0.0s
=> => sha256:3a667e35f6ca703bb8fed26fbcdb5dc8c6819a82112f6c4da8dd9441e6c222c3 891B / 891B 0.2s
=> => sha256:a0a2e1d76bfef0e3b37afb36c4e1ff0398f1f92988f2b1cacc63e953fd4e203d 244B / 244B 0.3s
=> => sha256:dc1c2b16275e20d87f9de7355eda30f5d6a84fea15e01fdac4600fa3d0f7a45a 2.46kB / 2.46kB 0.5s
=> => sha256:988fd8f2e2a7cea48d53220bcfd81139bcd83c3d7390a6e621f0490748a0762 10.00MB / 10.00MB 0.7s
=> => sha256:60f3998c5ba63dc3318a94cdcd5edb313ald9f325e5be9fef1e36a83db40ab1c 493B / 493B 0.3s
=> => sha256:ac2e4aa724ec9f16deec3d76b47c10c6416bdc26682dcbd436831f789ef66ab5 10.76MB / 10.76MB 0.9s
=> => sha256:e63b292a06a12240a013c7c7fc0ccc4117d2e75937b72ec7aa64b77333157619 510B / 510B 0.2s
=> => sha256:c5ba13601c856716cb7971ee60f01f0138236c05cadff15fd387bd5308610f98 475B / 475B 0.5s
=> => sha256:48bbb37a166bb998d4d6855dd70df915916376afc2a7eb0646a4453bbd43e8d6 19.17MB / 19.17MB 0.7s
=> => sha256:946bbe7211684bcd10ae8486cd441fc15f18504b0795elbbc3b0687b492b215e 269B / 269B 1.0s
=> => sha256:52c4af7a39c39eedcaf52194d52f333a71017ffe436a4e3725ebbb10f91baccf 86.93MB / 86.93MB 2.5s

=> => sha256:826c69643efc7a2fa413b41d83553f587c092a532d2d6582de3caf323e79a005 226B / 226B 0.2s
=> => sha256:f3ac85625e767ee0ec42b5a2ef93880251cd973b86f77124c4ed39bccd2f8bf9 30.06MB / 30.06MB 1.4s
=> => extracting sha256:f3ac85625e767ee0ec42b5a2ef93880251cd973b86f77124c4ed39bccd2f8bf9 0.5s
=> => extracting sha256:826c69643efc7a2fa413b41d83553f587c092a532d2d6582de3caf323e79a005 0.0s
=> => extracting sha256:52c4af7a39c39eedcaf52194d52f333a71017ffe436a4e3725ebbb10f91baccf 1.1s
=> => extracting sha256:946bbe7211684bcd10ae8486cd441fc15f18504b0795elbbc3b0687b492b215e 0.0s
=> => extracting sha256:48bbb37a166bb998d4d6855dd70df915916376afc2a7eb0646a4453bbd43e8d6 0.4s
=> => extracting sha256:c5ba13601c856716cb7971ee60f01f0138236c05cadff15fd387bd5308610f98 0.0s
=> => extracting sha256:e63b292a06a12240a013c7c7fc0ccc4117d2e75937b72ec7aa64b77333157619 0.0s
=> => extracting sha256:ac2e4aa724ec9f16deec3d76b47c10c6416bdc26682dcbd436831f789ef66ab5 0.0s
=> => extracting sha256:60f3998c5ba63dc3318a94cdcd5edb313ald9f325e5be9fef1e36a83db40ab1c 0.0s
=> => extracting sha256:988fd8f2e2a7cea48d53220bcfd81139bcd83c3d7390a6e621f0490748a0762 0.1s
=> => extracting sha256:dc1c2b16275e20d87f9de7355eda30f5d6a84fea15e01fdac4600fa3d0f7a45a 0.0s
=> => extracting sha256:a0a2e1d76bfef0e3b37afb36c4e1ff0398f1f92988f2b1cacc63e953fd4e203d 0.0s
=> => extracting sha256:3a667e35f6ca703bb8fed26fbcdb5dc8c6819a82112f6c4da8dd9441e6c222c3 0.0s
=> [2/2] COPY app /var/www/html 0.1s
=> => exporting to image 0.1s
=> => exporting layers 0.0s
=> => exporting manifest sha256:08bc4beb25a881d21adde8e314bdf3d4f89e2e202d1f807302159b70d4efcaea 0.0s
=> => exporting config sha256:5bcb3e5b9b40ff75d4ad290f1253b79b58455d74d7f20a42e29a23e3eadf7095 0.0s
=> => exporting attestation manifest sha256:0fb8882970c633f9eaf434cbaf424569fc5a8cd378f93f22cf6ba05b46e29023 0.0s
=> => exporting manifest list sha256:1260464a0479c0197cb8acb24699dccc5b0aab238438f9655c6a1a32a3728b6b 0.0s
=> => naming to docker.io/pfany/new-image:latest 0.0s
=> => unpacking to docker.io/pfany/new-image:latest 0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/z8v9x89k8xv728vbb6ostx91t
fanyperalta@MacBook-Air-de-Fany Ejercicio3 % docker images
REPOSITORY          TAG               IMAGE ID          CREATED           SIZE
pfany/new-image      latest            1260464a0479     9 seconds ago    611MB
mariadb              latest            310d29fbb581     6 weeks ago      491MB
phpmyadmin            latest            6cb0a7146734     2 months ago     819MB
fanyperalta@MacBook-Air-de-Fany Ejercicio3 %
```

Así pues, el nombre de la imagen es `pfany/new-image`, ya que Docker no permite nombres de imágenes con mayúsculas.

Creación del contenedor inicial

El contenedor llamado `ejercicio3` será accesible desde un navegador en el puerto 8000, con lo que a continuación de crear la imagen, creamos el contenedor llamado `ejercicio3` utilizando la imagen `pfany/newImage`

```
docker run -d -p 8000:80 --name ejercicio3 pfany/new-image
```

Al indicar `-p 8000:80` se hace el mapeo de puertos: mapeo el puerto 8000 del host al puerto 80 del contenedor (donde Apache escucha por defecto).

El resultado es el siguiente:

```
Ejercicio3 — zsh — 125x32
fanyperalta@MacBook-Air-de-Fany Ejercicio3 % docker run -d -p 8000:80 --name ejercicio3 pfany/new-image
4eee76ddf76a876773acf264ccdf8980b019a013029a88c6dc4698a1ec727af1
fanyperalta@MacBook-Air-de-Fany Ejercicio3 %
```

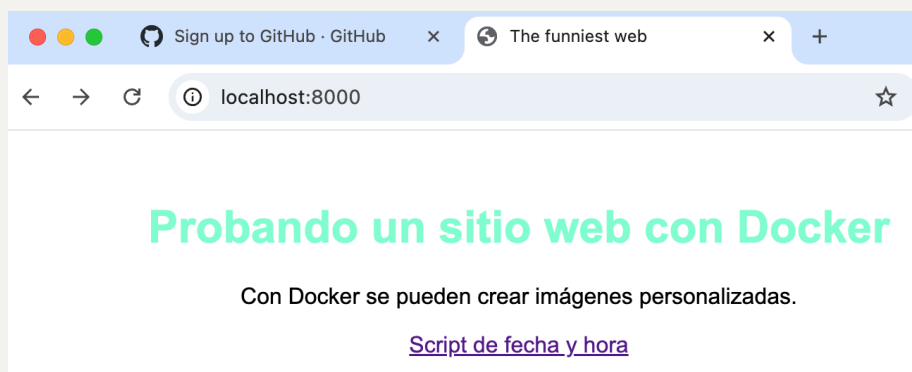
```
fanyperalta@MacBook-Air-de-Fany Ejercicio3 % docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
4eee76ddf76a ejercicio3	pfany/new-image	"docker-php-entrypoi..."	About a minute ago	Up About a minute	0.0.0.0:8000->80/tcp
640960cea9ba phpmyadmin_tarea3	phpmyadmin:latest	"/docker-entrypoint..."	4 days ago	Exited (0) 3 days ago	
6c80290c4cc0 mariadb_tarea3	mariadb:latest	"docker-entrypoint.s..."	4 days ago	Exited (0) 3 days ago	

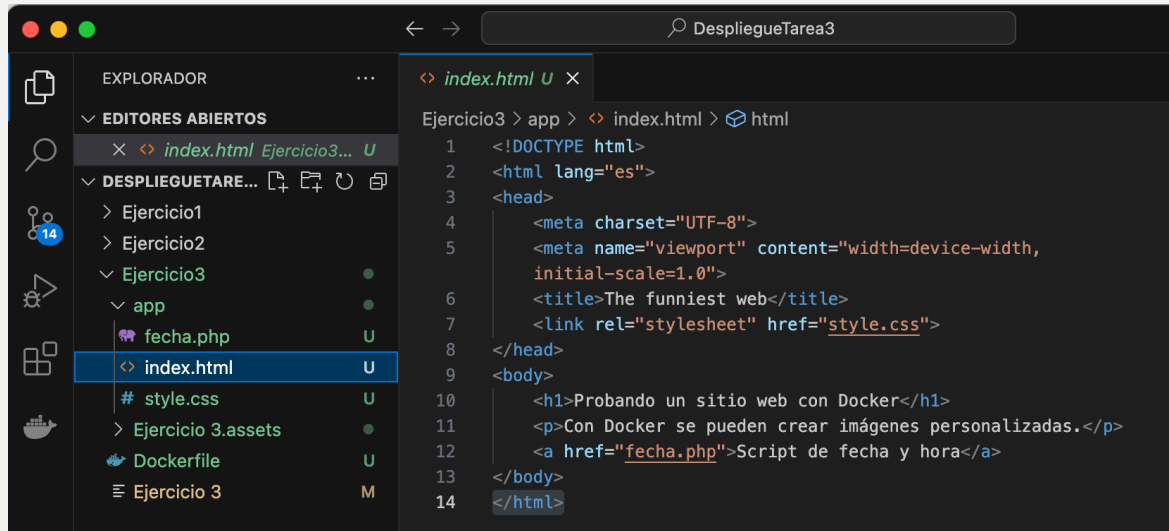
```
fanyperalta@MacBook-Air-de-Fany Ejercicio3 %
```

Acceso al navegador con la página `html` y con el script `php`

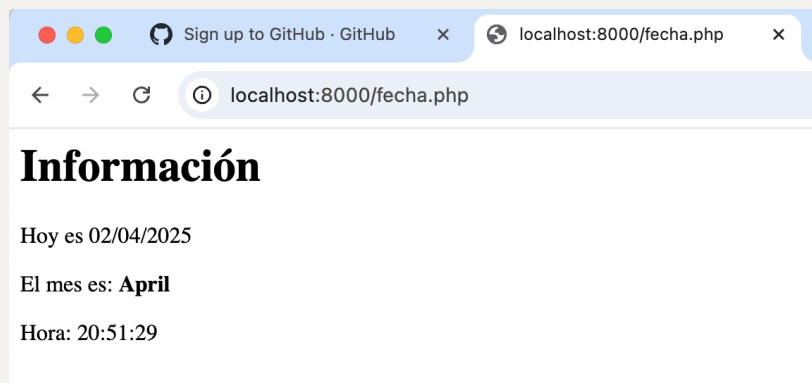
Accedo al navegador con el contenedor inicial:



que corresponde con el `index.html` muy básico creado:



Al clicar en `Script de fecha y hora`, el resultado es:



Subir la imagen creada a mi cuenta en Docker Hub