

Docker

1. Crea un Dockerfile que partiendo de una imagen PHP genera una imagen que:

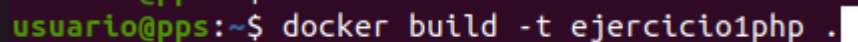
1. Copia una aplicación en PHP a un directorio del contenedor. Esta aplicación se debe copiar directamente desde un directorio del anfitrión. Para facilitar las cosas, debe de ser una aplicación sencilla que no emplee bases de datos (ya que si no también habría que instalar un MySQL).

- Creamos un fichero Dockerfile que contiene lo siguiente:
- Partimos de una imagen php con una versión 8,2 de apache.
- Establecemos el directorio de trabajo de nuestra máquina
- Actualizamos el contenedor
- Copiamos el directorio donde tenemos nuestro programa php previamente descargado y lo pegamos en el directorio html del contenedor
- Indicamos que abra el puerto 80
- Ejecutamos por consola que apache se ejecute en primer plano.



```
usuario@pps: ~  
usuario@pps: ~/app  
GNU nano 4.8 Dockerfile  
FROM php:8.2-apache  
# Establecemos el directorio de trabajo  
WORKDIR /home/usuario  
# Actualizamos la máquina  
RUN apt-get update -y  
# Copiamos el fichero php al directorio por defecto de apache  
COPY app/ /var/www/html/  
# Indicamos el puerto de escucha  
EXPOSE 80  
# Iniciamos el apache  
CMD ["apache2-foreground"]
```

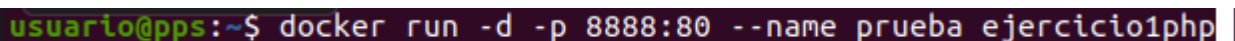
Una vez creado, lo lanzamos/construimos → `docker build -t ejercicio1php .`
El nombre lo inventamos y el punto es para que lo copie en el directorio actual.
Ojo, el nombre no puede tener mayúsculas.



```
usuario@pps:~$ docker build -t ejercicio1php .
```

Por último, arrancamos nuestro contenedor en el puerto que queramos: puerto establecido en el fichero Dockerfile el nombre lo inventamos de nuevo y seleccionamos el fichero nombrado en el paso anterior.

`docker run -d -p 8888:80 --name prueba ejercicio1php`



```
usuario@pps:~$ docker run -d -p 8888:80 --name prueba ejercicio1php
```

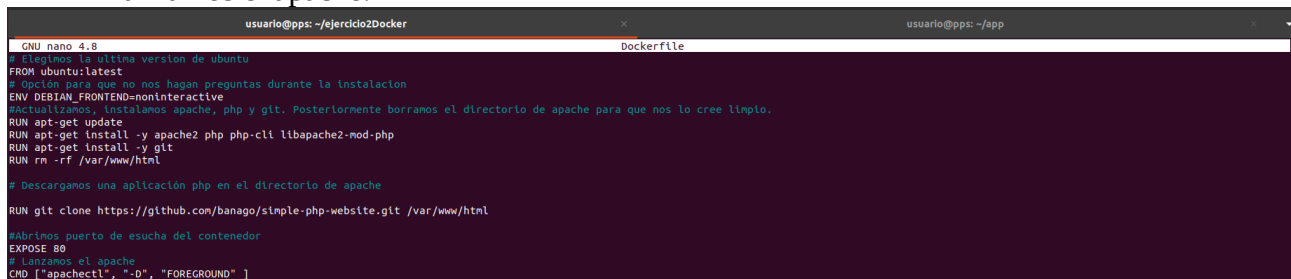
2. Crea un Dockerfile que partiendo de una imagen Ubuntu genera una imagen que:
 1. Instala Apache, de forma que se exponga el puerto 80.
 2. Instala PHP.
3. Copia una aplicación web en PHP al directorio de Apache que expone las páginas web. Esta aplicación se debe descargar automáticamente mediante algún comando como *git clone* o *curl*. Para facilitar las cosas, debe de ser una aplicación sencilla que no emplee bases de datos (ya que si no también habría que instalar un MySQL).

Creamos el Dockerfile → nano Dockerfile

```
usuario@pps:~/ejercicio2Docker$ nano Dockerfile
```

En el fichero definimos lo siguiente:

- Usamos la última versión de ubuntu.
- Actualizamos e instalamos los servicios necesarios (php, apache y gitclone).
- Usamos el comando git clone para obtener en este caso una página php.
- Utilizamos el puerto 80 del contenedor.
- Lanzamos el apache.

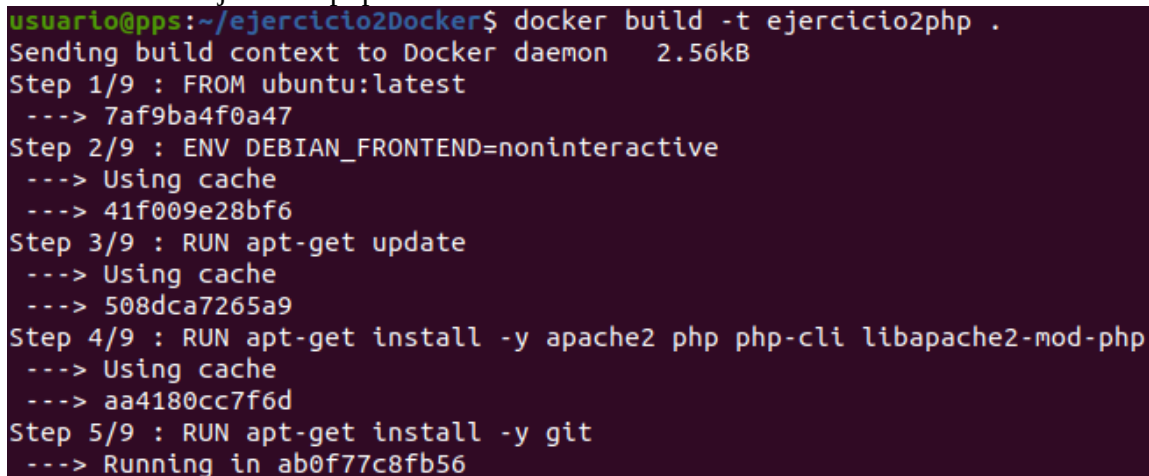


```
GNU nano 4.8 Dockerfile
# Elegimos la última versión de ubuntu
FROM ubuntu:latest
# Opción para que no nos hagan preguntas durante la instalación
ENV DEBIAN_FRONTEND=noninteractive
# Actualizamos, instalamos apache, php y git. Posteriormente borramos el directorio de apache para que nos lo cree limpio.
RUN apt-get update
RUN apt-get install -y apache2 php php-cli libapache2-mod-php
RUN apt-get install -y git
RUN rm -rf /var/www/html

# Descargamos una aplicación php en el directorio de apache
RUN git clone https://github.com/banago/simple-php-website.git /var/www/html

# Abrimos puerto de escucha del contenedor
EXPOSE 80
# Lanzamos el apache
CMD ["apachectl", "-D", "FOREGROUND"]
```

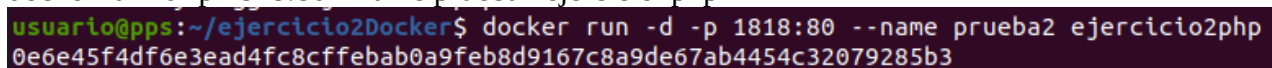
Procedemos a montarlo llamándolo ejercicio2php y lo guardamos en el directorio actual.
docker build -t ejercicio2php .



```
usuario@pps:~/ejercicio2Docker$ docker build -t ejercicio2php .
Sending build context to Docker daemon 2.56kB
Step 1/9 : FROM ubuntu:latest
--> 7af9ba4f0a47
Step 2/9 : ENV DEBIAN_FRONTEND=noninteractive
--> Using cache
--> 41f009e28bf6
Step 3/9 : RUN apt-get update
--> Using cache
--> 508dca7265a9
Step 4/9 : RUN apt-get install -y apache2 php php-cli libapache2-mod-php
--> Using cache
--> aa4180cc7f6d
Step 5/9 : RUN apt-get install -y git
--> Running in ab0f77c8fb56
```

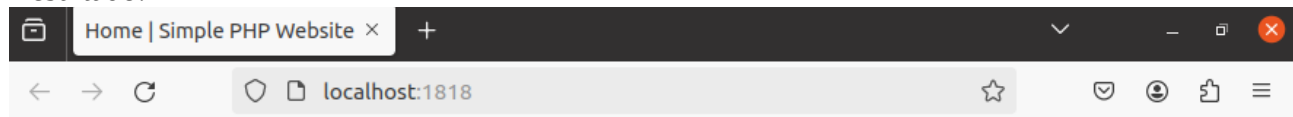
Lo corremos en el puerto 1818 de nuestra máquina y en el 80 del contenedor.

docker run -d -p 1818:80 --name prueba2 ejercicio2php



```
usuario@pps:~/ejercicio2Docker$ docker run -d -p 1818:80 --name prueba2 ejercicio2php
0e6e45f4df6e3ead4fc8cffeabab0a9feb8d9167c8a9de67ab4454c32079285b3
```

Resultado:



Simple PHP Website

[Home](#) | [About Us](#) | [Products](#) | [Contact](#)

Home

This is **home** page. Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

It is a long established fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using Lorem Ipsum is that it has a more-or-less normal distribution of letters, as opposed to using 'Content here, content here', making it look like readable English. Many desktop publishing packages and web page editors now use Lorem Ipsum as their default model text, and a search for 'lorem ipsum' will uncover many web sites still in their infancy. Various versions have evolved over the years, sometimes by accident, sometimes on purpose (injected humour and the like).

3. Crea un contenedor para cada una de esas imágenes y verifica que funciona. Para y borra dicho contenedor.

Listamos los dockers → `docker ps`

```
usuario@pps:~/ejercicio2Docker$ docker ps
```

CONTAINER ID	IMAGE	COMMAND NAMES	CREATED	STATUS	PORTS
0e6e45f4df6e	ejercicio2php	"apachectl -D FOREGR..."	19 minutes ago	Up 19 minutes	0.0.0.0
:1818->80/tcp, :::1818->80/tcp		prueba2			
7cafc7dbf9be	ejercicio1php	"docker-php-entrypoi..."	2 hours ago	Up 2 hours	0.0.0.0
:8888->80/tcp, :::8888->80/tcp		prueba			

Procedemos a pausar el docker del ejercicio 1 → `docker stop 7cafc7dbf9be`

```
usuario@pps:~/ejercicio2Docker$ docker stop 7cafc7dbf9be
7cafc7dbf9be
usuario@pps:~/ejercicio2Docker$ docker ps
```

CONTAINER ID	IMAGE	COMMAND NAMES	CREATED	STATUS	PORTS
0e6e45f4df6e	ejercicio2php	"apachectl -D FOREGR..."	25 minutes ago	Up 25 minutes	0.0.0.0
:1818->80/tcp, :::1818->80/tcp		prueba2			

```
usuario@pps:~/ejercicio2Docker$
```

Por último, lo borramos → `docker rm 7cafc7dbf9be`

```
usuario@pps:~/ejercicio2Docker$ docker rm 7cafc7dbf9be
7cafc7dbf9be
usuario@pps:~/ejercicio2Docker$ docker ps -a | grep 7cafc7dbf9be
usuario@pps:~/ejercicio2Docker$
```

4. Emplea un comando para lanzar 20 contenedores de la segunda imagen, cada uno mapeado en un puerto distinto del anfitrión. Cuando veas que funcionan, para y borra dichos contenedores.

Ejecutamos el comando `for` para hacer un bucle que cree 20 máquinas en distintos puertos usando el contenedor del segundo ejercicio.

En el comando indicamos que para la variable `i` se recorre 20 veces, se suma al puerto 8080 y se añade como campo después del nombre del contenedor.

`for i in {1..20}; do docker run -d -p $((8080+$i)):80 --name contenedor$i ejercicio2php; done`

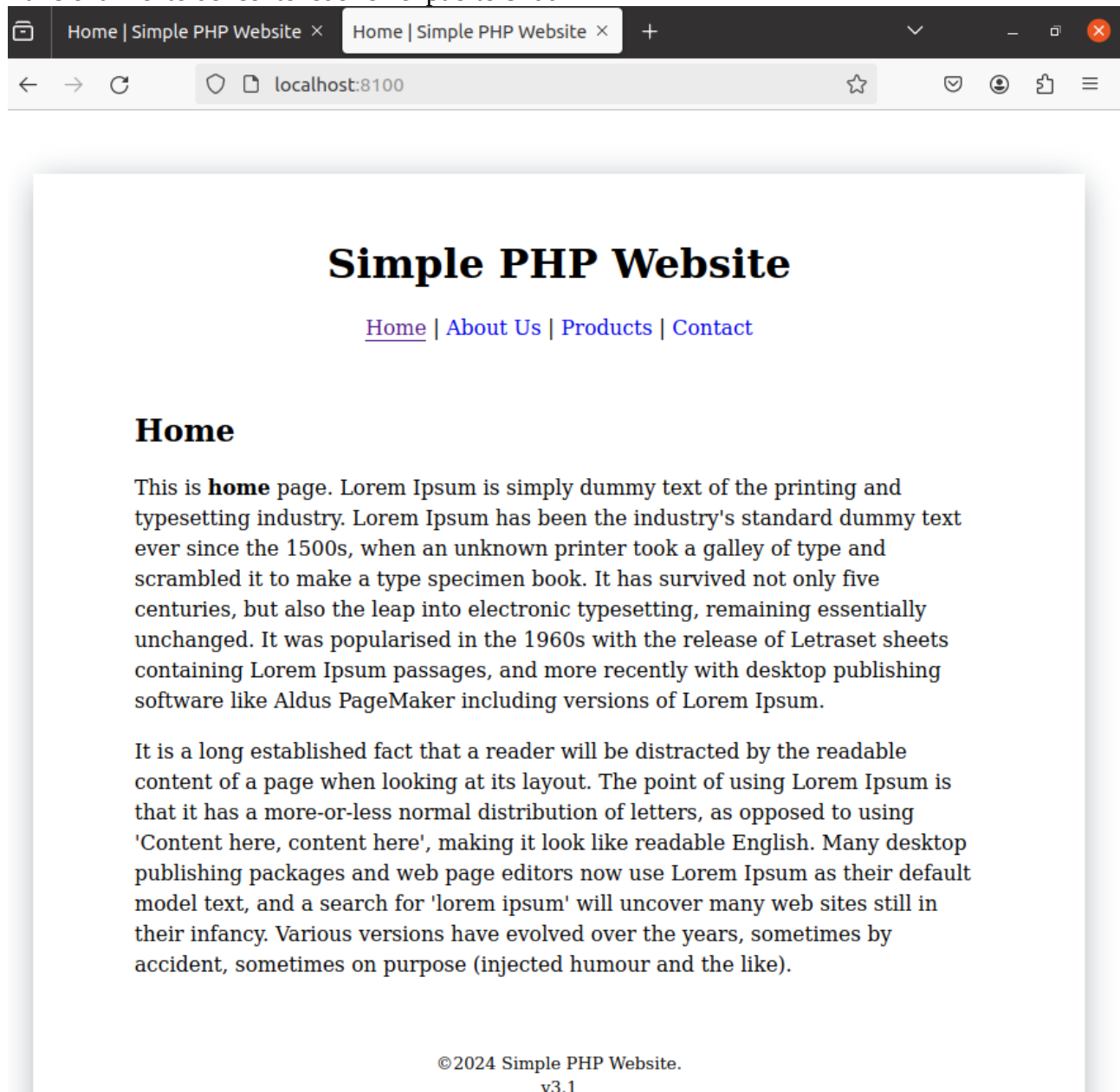
```
usuario@pps:~/ejercicio2Docker$ for i in {1..20}; do docker run -d -p $((8080+$i)):80 --name con
tenedor$i ejercicio2php; done
170e7f5c8b6a4e439361d27ac69e8fb93f4a4175f929184983ca0fe539bb4445
33fa6f13783ed6777f8788b01f2726cc88d883f40beffd51248be37bf85e166d
0ca38d73a886eda4c294d393725e95872555b100750caba76560134305dd0406
887ddb7ae245e1f77415000be3ee19af144c5012ede90b4b83f723efe651c251
905eb013fde7d2ec10ff007018a582e015e8b4363dfd704237887c4ab67c0407
7240f5234a9a5706cd3b513c02ca8773b60c081aeceb6931193c3a113bc669d1
9c9fa9dd9658810c21891c65a49067e744dd000b76f285876374ea1bab25db6b
c3947b6f9b891b09fc46569734a21d4dd0221c383ea7e524b5f506be03cc6f57
921e69b08b7821570a6f9a2856defbd1fb953ad01c163bfbf9be2519356172ea
2401f7d858458cfc23d15bf0aabfa7c95e5e078fee6b8abfca0a5f7fe73ae613
a32ea60f6a9142183799f9d1645d58ccf5dfae796ae0341702091cf8c080787e
3f6ec6825ad48652562e8b28466880fd6f31adb31003eb113619f6ea31eb6d82
d2f450963c2cffa42d64c2df5ef0564ebfe9e81e7c3d33f676d4cf32ea6f306b
4baa9706125541bd3e860d1ad5ba0a03a546cf71543012311c73cc6b50eb7cde
d7adbe0e7e737d98702144b19b17a0a74751187e57cfec173c27e4872b36a84f8
a028e5ddfa95ca71be481fbaeb88088cf7781b646224e20cc9b9daf9e98d02ec9
636cdbee3b3cf9e5858e49edfe6e53ca8fe2475655744f6709ae9f130f4c4632
9e6d980ee011995d94324bca8ea8abf081909e10c812a359e9555a62eaa56cf2
e804cce9498f6b79c68cf565bde1f25e68ec2ba86539d466f3e25d586de0aaac
e6ee79ee36232394908a3315bd818a335c2b8c65fbf27c3238d10ff3136038c8
```

Ejecutamos un docker ps para visualizar que las máquinas están en ejecución.

docker ps

```
usuario@pps:~/ejercicio2Docker$ docker ps
```

CONTAINER ID	IMAGE	COMMAND NAMES	CREATED	STATUS	PORTS
e6ee79ee3623	ejercicio2php	"apachectl -D FOREGR..."	6 seconds ago	Up 5 seconds	0.0.0.0
:8100->80/tcp, :::8100->80/tcp		contenedor20			
e804cce9498f	ejercicio2php	"apachectl -D FOREGR..."	6 seconds ago	Up 5 seconds	0.0.0.0
:8099->80/tcp, :::8099->80/tcp		contenedor19			
9e6d980ee011	ejercicio2php	"apachectl -D FOREGR..."	7 seconds ago	Up 6 seconds	0.0.0.0
:8098->80/tcp, :::8098->80/tcp		contenedor18			
636cdbee3b3c	ejercicio2php	"apachectl -D FOREGR..."	7 seconds ago	Up 6 seconds	0.0.0.0
:8097->80/tcp, :::8097->80/tcp		contenedor17			
a028e5ddfa95	ejercicio2php	"apachectl -D FOREGR..."	8 seconds ago	Up 7 seconds	0.0.0.0
:8096->80/tcp, :::8096->80/tcp		contenedor16			
d7adbe07e737	ejercicio2php	"apachectl -D FOREGR..."	8 seconds ago	Up 7 seconds	0.0.0.0
:8095->80/tcp, :::8095->80/tcp		contenedor15			
4baa97061255	ejercicio2php	"apachectl -D FOREGR..."	9 seconds ago	Up 7 seconds	0.0.0.0
:8094->80/tcp, :::8094->80/tcp		contenedor14			
d2f08b1c2c	ejercicio2php	"apachectl -D FOREGR..."	9 seconds ago	Up 8 seconds	0.0.0.0
:8093->80/tcp, :::8093->80/tcp		contenedor13			
3f6ec6825ad4	ejercicio2php	"apachectl -D FOREGR..."	10 seconds ago	Up 8 seconds	0.0.0.0
:8092->80/tcp, :::8092->80/tcp		contenedor12			
a32ea60f6a91	ejercicio2php	"apachectl -D FOREGR..."	10 seconds ago	Up 9 seconds	0.0.0.0
:8091->80/tcp, :::8091->80/tcp		contenedor11			
2401f7d85845	ejercicio2php	"apachectl -D FOREGR..."	10 seconds ago	Up 9 seconds	0.0.0.0
:8090->80/tcp, :::8090->80/tcp		contenedor10			
921e69b08b78	ejercicio2php	"apachectl -D FOREGR..."	11 seconds ago	Up 10 seconds	0.0.0.0
:8089->80/tcp, :::8089->80/tcp		contenedor9			
c3947b6f9b89	ejercicio2php	"apachectl -D FOREGR..."	11 seconds ago	Up 10 seconds	0.0.0.0
:8088->80/tcp, :::8088->80/tcp		contenedor8			



Paramos los contenedores antes de borrarlos aprovechando el script anterior, si miramos con docker ps -a vemos que están ocultos pero no están en funcionamiento.

```
for i in {1..20}; do docker stop contenedor$i; done
```

```
usuario@pps:~/ejercicio2Docker$ for i in {1..20}; do docker stop contenedor$i; done
contenedor1
contenedor2
contenedor3
contenedor4
contenedor5
contenedor6
contenedor7
contenedor8
contenedor9
contenedor10
contenedor11
contenedor12
contenedor13
contenedor14
contenedor15
contenedor16
contenedor17
contenedor18
contenedor19
contenedor20
usuario@pps:~/ejercicio2Docker$ ps
  PID TTY          TIME CMD
  2259 pts/0    00:00:00 bash
  3196 pts/0    00:00:00 ps
usuario@pps:~/ejercicio2Docker$ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED    STATUS    PORTS    NAMES
usuario@pps:~/ejercicio2Docker$ docker ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED    STATUS    PORTS    NAMES
TUS
e6ee79ee3623   ejercicio2php          "apachectl -D FOREGR..." 21 hours ago    Exited
ted (137)
e804cce9498f   ejercicio2php          "apachectl -D FOREGR..." 21 hours ago    Exited
ted (137)
```

Borramos los contenedores volviendo a aprovechar el script.

```
for i in {1..20}; do docker rm contenedor$i; done
```

Una vez borrados ya no nos aparecen con el comando `docker ps -a`

```
usuario@pps:~/ejercicio2Docker$ for i in {1..20}; do docker rm contenedor$i; done
contenedor1
contenedor2
contenedor3
contenedor4
contenedor5
contenedor6
contenedor7
contenedor8
contenedor9
contenedor10
contenedor11
contenedor12
contenedor13
contenedor14
contenedor15
contenedor16
contenedor17
contenedor18
contenedor19
contenedor20
usuario@pps:~/ejercicio2Docker$ docker ps -a
```

CONTAINER ID	IMAGE	PORTS	NAMES	COMMAND	CREATED	STATUS
0e6e45f4df6e	ejercicio2php			"apachectl -D FOREGR..."	22 hours ago	Exited
ted (137)	21 hours ago		prueba2			
b8ce1fc62b7f	d8d05495e144			"/bin/sh -c 'git clo..."	22 hours ago	Exited
ted (127)	22 hours ago		infallible_shtern			
9a873327e2fc	7f3d83319815			"/bin/sh -c 'git clo..."	22 hours ago	Exited
ted (127)	22 hours ago		goofy_noether			
d8da2cd6e8cc	7f3d83319815			"/bin/sh -c 'https:/..."	22 hours ago	Exited
ted (127)	22 hours ago		exciting_nightingale			
316ea9610a8a	de96209cca6c			"/bin/sh -c 'https:/..."	22 hours ago	Exited
ted (127)	22 hours ago		optimistic_goldstine			
d55d11e37a65	de96209cca6c			"/bin/sh -c 'apt-get..."	22 hours ago	Exited

Git

Mantén un repositorio de Git público con:

- Memoria de la práctica.
- El fichero Dockerfile.

Haz un *commit* y un *push* del proyecto cada vez que superes uno de los ítems de la sección anterior.