

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)

Activamos el Docker:

```
usuario@pps:~$ sudo systemctl enable docker
[sudo] contraseña para usuario:
Synchronizing state of docker.service with SysV service script with /lib/systemd/sysem
emd-sysv-install.
```

Empezamos la actividad:

```
usuario@pps:~$ sudo systemctl start docker
usuario@pps:~$ sudo usermod -aG docker $USER
usuario@pps:~$ mkdir -p ~/docker/php-app/app
usuario@pps:~$ cd ~/docker/php-app
```

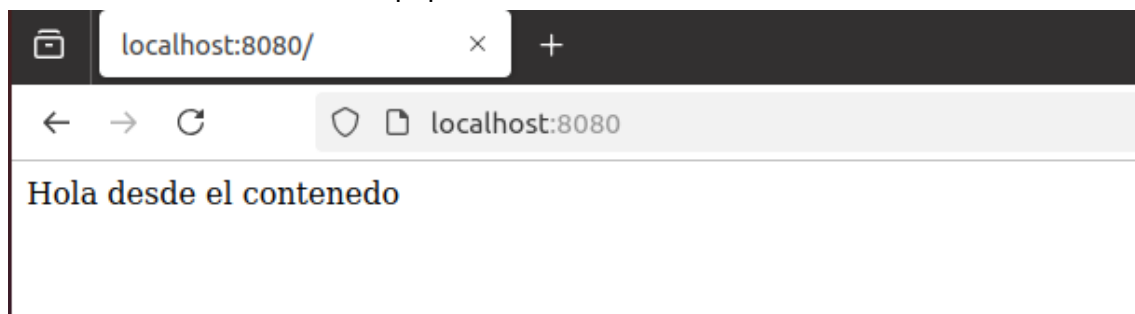
Creamos el dockerfile y también el php:

```
usuario@pps:~/docker/php-app$ echo "<?php echo 'Hola desde el contenedo'; ?>" > app/index.php
usuario@pps:~/docker/php-app$ nano Dockerfile
usuario@pps:~/docker/php-app$ docker build -t php-app .
[+] Building 0.0s (0/1)
[+] Building 34.8s (8/8) FINISHED
docker:default
docker:default
```

Ejecutamos el contenedor:

```
usuario@pps:~/docker/php-app$ docker run -d --name php-app-container -p 8080:80 php-app
42b2d839bd3a453de343ec8fb8a15420ee8ee386afde0feaedef25bb560ada890
```

Y desde localhost veremos el php:



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

Creamos un nuevo directorio:

```
usuario@pps:~/docker/php-app$ mkdir -p ~/docker/ubuntu-php
usuario@pps:~/docker/php-app$ cd ~/docker/ubuntu-php
```

Creamos el Dockerfile:

```
FROM ubuntu:22.04

ENV DEBIAN_FRONTEND=noninteractive

ENV DEBIAN_FRONTEND=noninteractive
ENV TZ=Europe/Madrid

RUN apt-get update && \
    apt-get install -y apache2 php git && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*

RUN rm -rf /var/www/html/* && \
    git clone https://github.com/heroku/php-getting-started.git /var/www/html

EXPOSE 80

CMD ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]
```

En este docker file cogemos así y creamos la imagen de ubuntu, php, apache y la app descargada, todo seguido.

Ahora construimos la imagen:

```
usuario@pps:~/docker/ubuntu-php$ nano Dockerfile
usuario@pps:~/docker/ubuntu-php$ docker build -t ubuntu-php-app .
[+] Building 2.1s (7/7) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile             0.0s
=> => transferring dockerfile: 448B                             0.0s
=> [internal] load metadata for docker.io/library/ubuntu:22.04 0.5s
=> [internal] load .dockerignore                               0.0s
=> => transferring context: 2B                                    0.0s
=> [1/3] FROM docker.io/library/ubuntu:22.04@sha256:67cadaff1dca187079fce41360d5a7eb 0.0s
=> CACHED [2/3] RUN apt-get update && apt-get install -y apache2 php git && 0.0s
=> [3/3] RUN rm -rf /var/www/html/* && git clone https://github.com/heroku/php-g 1.1s
=> exporting to image                                           0.5s
=> => exporting layers                                           0.5s
=> => writing image sha256:b5ef4219294796cde690f40fd6053234d406b5f46b1fea89319bbe5c5 0.0s
=> => naming to docker.io/library/ubuntu-php-app               0.0s
```






Ahora ejecutamos el contenedor:

```
usuario@pps:~/docker/ubuntu-php$ nano Dockerfile
usuario@pps:~/docker/ubuntu-php$ docker run -d --name ubuntu-app-container -p 8081:80 ubuntu
8040f6b1d809bed291ee122cf8c983938bd27cd2d94228688257e0efb58e5d9a
```

Ahora entramos al localhost y vemos que ha funcionado:

[←](#) [→](#) [↻](#) [🔒](#) [📄](#) localhost:8081/views/

Index of /views

Name	Last modified	Size	Description
 Parent Directory		-	
 header.html	2025-05-07 17:31	500	
 index.twig	2025-05-07 17:31	3.8K	
 layout.html	2025-05-07 17:31	156	
 nav.html	2025-05-07 17:31	4.3K	

Apache/2.4.52 (Ubuntu) Server at localhost Port 8081



[Parent Directory](#)



[header.html](#)

2025-05-07 17:31 500



[index.twig](#)

2025-05-07 17:31 3.8K



[layout.html](#)

2025-05-07 17:31 156

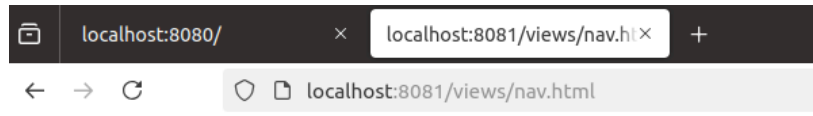


[nav.html](#)

2025-05-07 17:31 4.3K

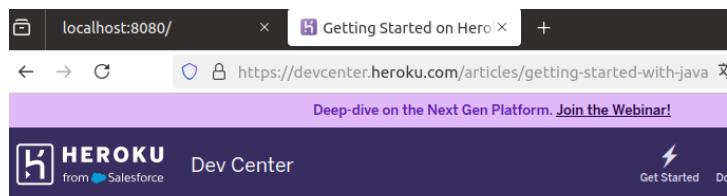
Apache/2.4.52 (Ubuntu) Server at localhost Port 8081

Vemos que la app funciona, ya que es simplemente una app html que no tiene base de datos:



- [Home](#)
- [How Heroku Works](#)
- [Getting Started Guides \(Cedar\)](#)
 - [Getting Started on Heroku with Ruby](#)
 - [Getting Started on Heroku with Node.js](#)
 - [Getting Started on Heroku with PHP](#)
 - [Getting Started on Heroku with Python](#)
 - [Getting Started on Heroku with Java](#)
 - [Getting Started on Heroku with Clojure](#)
 - [Getting Started on Heroku with Scala](#)
 - [Getting Started on Heroku with Go](#)
 - [Getting Started on Heroku with .NET](#)
 - [Getting Started on Heroku with Heroku Connect](#)
- [Getting Started Guides \(Fir\)](#)
 - [Getting Started on Heroku Fir with Ruby](#)
 - [Getting Started on Heroku Fir with Node.js](#)
 - [Getting Started on Heroku Fir with PHP](#)
 - [Getting Started on Heroku Fir with Python](#)
 - [Getting Started on Heroku Fir with Java \(Maven\)](#)
 - [Getting Started on Heroku Fir with Java \(Gradle\)](#)
 - [Getting Started on Heroku Fir with Clojure](#)
 - [Getting Started on Heroku Fir with Scala](#)
 - [Getting Started on Heroku Fir with Go](#)
 - [Getting Started on Heroku Fir with .NET](#)
 - [Getting Started on Heroku with Heroku Connect](#)
- [Heroku Dev Center](#)

Si pulsamos en Getting started:



Getting Started on Heroku
with Java

Ahora paramos y borramos el docker:

```
usuario@pps:~/docker/ubuntu-php$ docker stop ubuntu-app-container
ubuntu-app-container
usuario@pps:~/docker/ubuntu-php$ docker rm ubuntu-app-container
ubuntu-app-container
```

Ahora vamos a emplear el comando para lanzar 20 contenedores de esa segunda imagen en puertos distintos.

```
usuario@pps:~/docker/ubuntu-php$ for i in $(seq 1 20); do port=$((800 + i)); docker run
-d --name web$i -p $port:80 ubuntu-php-app; done
a2daf7ce0d72d56c2a79279f6060e31dcd4c1cedf33e26ed26111879e8c411
7d772b4d0bbcb2dd9b9d7a6eaf26b20408b2fbf65ebe5226dc2d2a5be200b4c93
7ccf5f2089a13d3a8873b325639da3582add5bf32c250f7e8675aa51c340e74d
b1c3bbd9b1c743928ea0fccde5d6bad0d1f398b5cb657e015738ceee6029506d
5aba67ff02c3e5a14dfef06d7511b3e0fa5fbde1c909d64da27f3fda55edc851
0dc75d04421e36893577a942a2961c5278c6957ba173828316f3dfd69491d9c7
4a2e2e5929982e310b38d5babe40a55cb00f74f71a10a2cf45e73eb1bba1a55b
a18f600ce083c5a2d91619593033e03d983d98396a01908799f6ff838f7b7c1c
24ad38a124ec5d9179f0992bf8eab07a2e967b29f38c75bd37999f1e040d5fda
ab955a7be033e9ca9b8829d2b16da31586cdc0c1b357302e0affbfff6d8b39507
ec28dabd8cf8b03e36e5875720b9dd9dfb5007e4dd9625e778e886a24e58a8d8
654bb0e1e9d7e23fd882c0ae968aee98cd9915d845adebcd2cdf369911085b10
aca766ae2b712d1f363c6d6dc97258a9d31b0b8f06e6c283e721aede4c4e1960
7876abc15ff4a730165e461cda91341eebf752709617204fa9e782bcce26b35b
3a8ac8acf59bd61dcf5a35f5dc34742aea16d18454604e2dc942688f0762758a5
d0f63f94ac616787a3002e66f3614458b873c67b250f330dc7607b23b4311f0b
577e56b08842d822e3fd50e8483dbfb11f30bd16495fa5538ceaa1619661f4bd
3267a9d8f639177904a69ac562a24bfe1049af7dfeaab73a05c9e2dfb497ec19
3b4c251f218964c62cb4adb3a1e3a2814b32617fb18e4237ce770bb31b7d5dd6
6f94ca57f6211c9bb3bd129bfff4571230ab73247efc84cb2678f1efbd37329cd
```








Esto cargará los 20 puertos, por ejemplo en este caso si vamos al puerto 801 se verá.

7 de may 17:52

localhost:8080/ Index of /

localhost:801

Index of /

	Name	Last modified	Size	Description
	Procfile	2025-05-07 17:31	29	
	README.md	2025-05-07 17:31	2.4K	
	app.json	2025-05-07 17:31	199	
	composer.json	2025-05-07 17:31	308	
	composer.lock	2025-05-07 17:31	57K	
	views/	2025-05-07 17:31	-	
	web/	2025-05-07 17:31	-	

Apache/2.4.52 (Ubuntu) Server at localhost Port 801