

# Docker en el entorno de desarrollo



# Alejandro Hernández (@aleherse)



Desarrollador de aplicaciones web, consultor y formador.

Trabajando actualmente en un juego web de estrategia por turnos ambientado en la antigua Grecia

+info: <http://about.me/aleherse>

# **¿Qué es Docker?**

**Software Open Source que  
automatiza el despliegue de  
aplicaciones dentro de  
contenedores**

# ¿Qué hay en un contenedor?

- Sistema de ficheros completo
- Cualquier cosa que pueda instalarse en linux
- Nuestra aplicación

# Contenedores vs Máquinas Virtuales

- Ejecución en espacio de usuario sobre el kernel del SO

# Contenedores vs Máquinas Virtuales

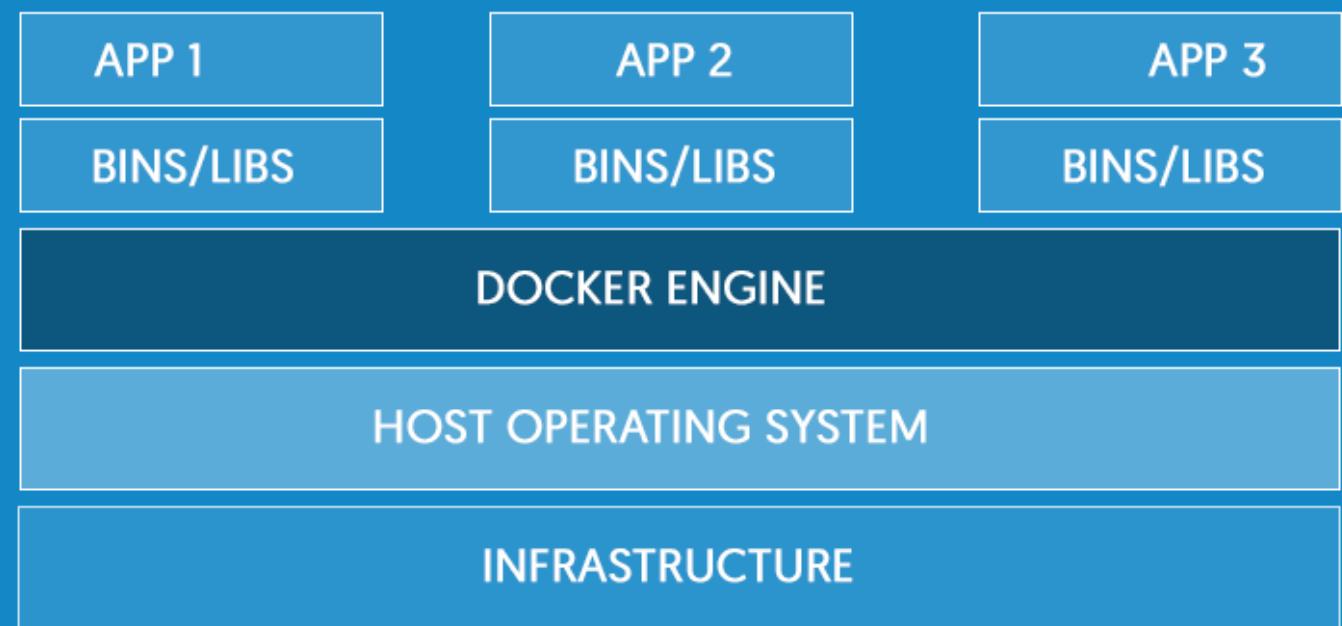
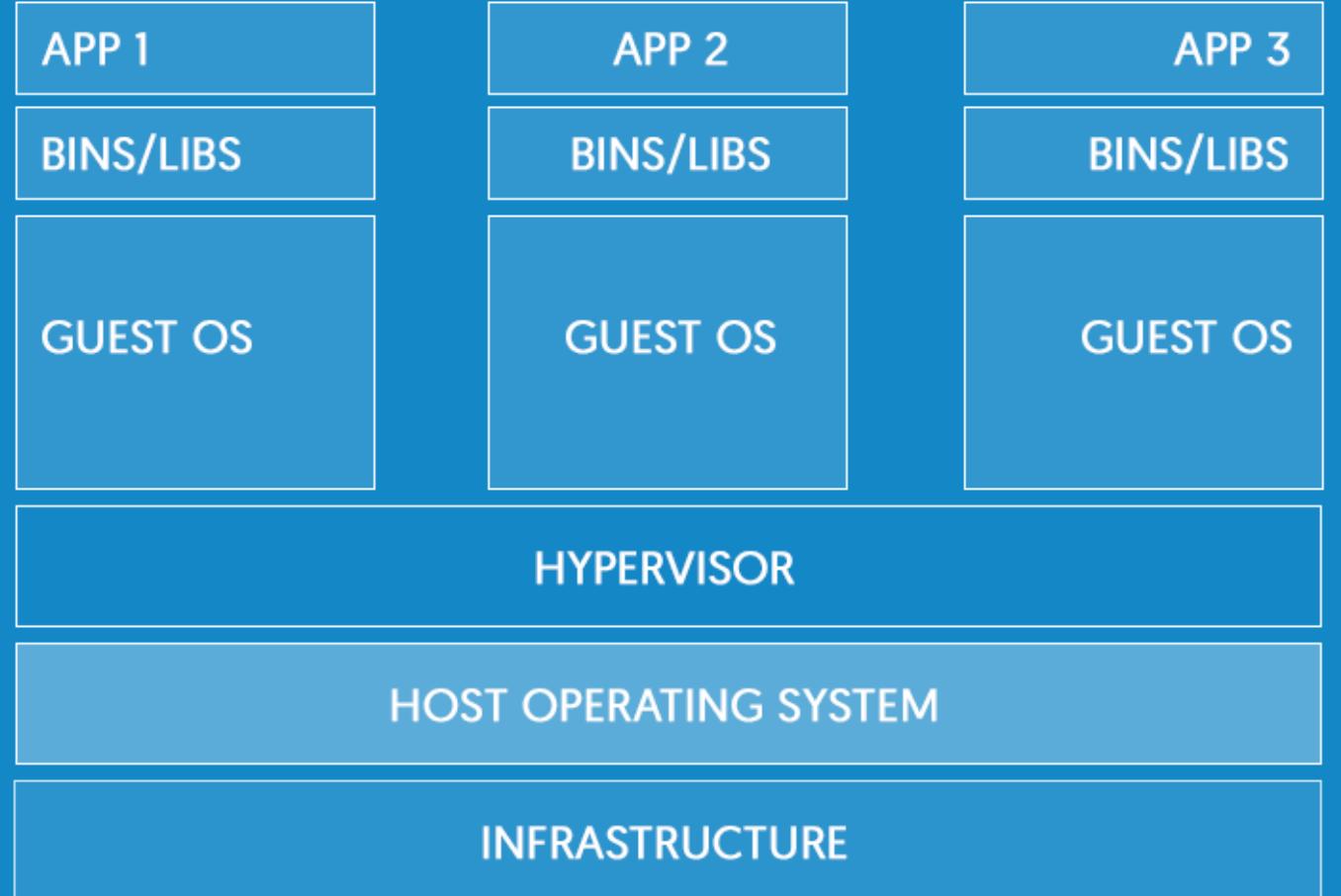
- Ejecución en espacio de usuario sobre el kernel del SO
- Menor sobrecarga de la máquina anfitrión

# Contenedores vs Máquinas Virtuales

- Ejecución en espacio de usuario sobre el kernel del SO
- Menor sobrecarga de la máquina anfitrión
- Permite multiples instancias aisladas

# Contenedores vs Máquinas Virtuales

- Ejecución en espacio de usuario sobre el kernel del SO
- Menor sobrecarga de la máquina anfitrión
- Permite multiples instancias aisladas
- Requiere un kernel de Linux



# Tecnología detrás de Docker

- Linux Kernel Control Groups (cgroups)

# Tecnología detrás de Docker

- Linux Kernel Control Groups (cgroups)
- Linux Kernel Namespaces

# Tecnología detrás de Docker

- Linux Kernel Control Groups (cgroups)
- Linux Kernel Namespaces
- Union Mount

# Tecnología detrás de Docker

- Linux Kernel Control Groups (cgroups)
- Linux Kernel Namespaces
- Union Mount
- Docker libcontainer

# Objetivos de Docker

- Forma fácil y rápida de modelar la realidad

# Objetivos de Docker

- Forma fácil y rápida de modelar la realidad
- Segregación lógica de tareas

# Objetivos de Docker

- Forma fácil y rápida de modelar la realidad
- Segregación lógica de tareas
- Ciclo de desarrollo eficiente y rápido

# Objetivos de Docker

- Forma fácil y rápida de modelar la realidad
- Segregación lógica de tareas
- Ciclo de desarrollo eficiente y rápido
- Promueve arquitectura orientada a servicios

# Componentes

- Docker Engine
- Docker Images
- Docker Containers
- Registries

# Instalación

Guía muy completa en la web

Existen aplicaciones para OSX y Windows

Puedes descargar y ejecutar un script para Linux

# Docker containers

- Un formato de imagen
- Un conjunto de operaciones estándar
- Un entorno de ejecución

**¡Manos a la obra!**

**Contenedores**

¿Funciona?

```
$ docker info
```

Ejecutemos nuestro primer contenedor

```
$ docker run -i -t ubuntu /bin/bash
```

Juguemos un poco

```
# hostname  
# ls -la  
# cat /etc/hosts  
# ps -aux  
# exit
```

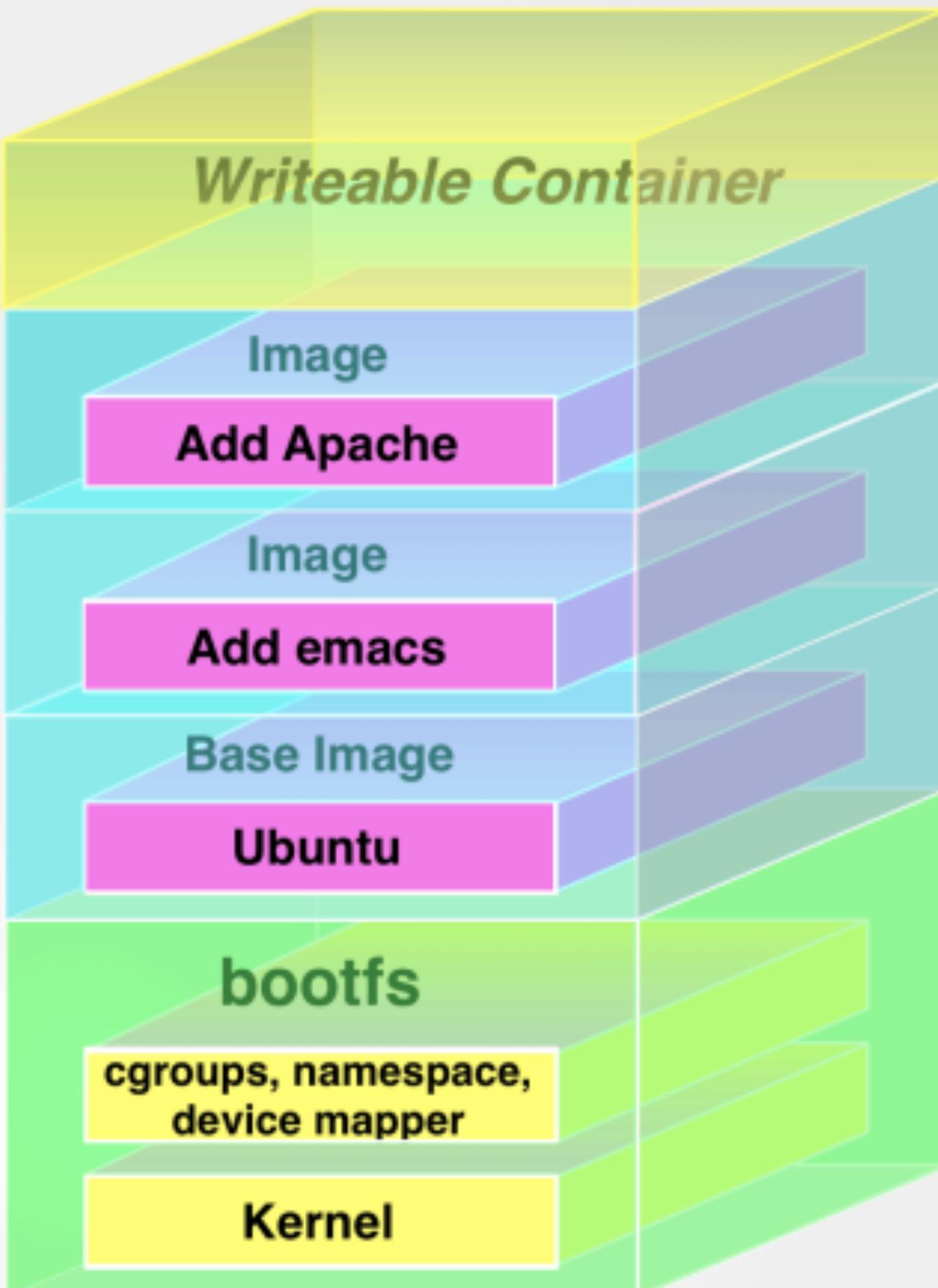
# ¿Qué ha pasado con nuestro contenedor?

```
$ docker ps -a  
$ docker start <container_name>  
$ docker ps  
$ docker attach <container_name>
```

## Ejecutar contenedores en segundo plano

```
$ docker run --name daemon -d ubuntu /bin/sh -c \
"while true; do echo hello world; sleep 1; done"
$ docker ps
```

```
$ docker logs -f deamon
[Ctrl-C]
$ docker exec -d daemon touch /new_file
$ docker exec -t -i daemon /bin/bash
# ls -la
[Ctrl-C]
$ docker stop daemon
$ docker rm daemon
```



# Docker Images

Base image contiene el sistema operativo mínimo (debian < 150mb)

Gracias a *union mount* docker superpone varios sistemas de ficheros de solo lectura

Añade al final un sistema de ficheros de lectura-escritura

**¡Manos a la obra!**

**Imágenes**

Repositorio de imágenes oficial:

<https://hub.docker.com/>

¿Cómo construir nuestra propia imagen?

Creamos un fichero Dockerfile conteniendo:

```
FROM ubuntu:16.04
```

```
RUN apt-get update; apt-get install -y nginx
```

```
RUN echo 'Soy un fichero dentro del contenedor' \  
> /var/www/html/index.html
```

```
EXPOSE 80
```

## Comandos relacionados con imágenes

```
$ docker build -t aleherse/static_web .
$ docker images
$ docker rmi aleherse/static_web
```

# Docker Compose

Inicia una serie de contenedores definidos en un fichero YAML

Incluido en Docker para OSX y Windows

Para usar en linux seguir los pasos de instalación de:

<https://github.com/docker/compose/releases>

**¡Manos a la obra!**

**Orquestación**

# Funcionalidades de nuestra aplicación de ejemplo

- Muestra un formulario con destinatario, asunto y contenido
- Envía el contenido del formulario en un correo electrónico
- Almacena los datos del correo con la fecha de envío
- Lista todo los correos que se han enviado

# ¿Qué software tenemos en producción?

- Aplicación PHP 7.1.0
- Servidor web Nginx 1.11.6
- Base de datos MySQL 5.7.16
- Servidor SMTP Postfix 3.5

# Aplicación PHP 7.1.0

```
FROM php:7.1.0-fpm
```

```
RUN apt-get update && apt-get install -yq git vim zip && \
    docker-php-ext-install mysqli pdo pdo_mysql && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*
```

```
ENV COMPOSER_ALLOW_SUPERUSER=1
```

```
...
```

```
COPY local.ini /usr/local/etc/php/conf.d/
```

```
WORKDIR /app
```

```
VOLUME /app
```

```
EXPOSE 9000
```

```
CMD ["php-fpm", "-F"]
```

## **Un inciso sobre los volúmenes**

- Comparten ficheros del anfitrión en el contenedor

## **Un inciso sobre los volúmenes**

- Comparten ficheros del anfitrión en el contenedor
- Se saltan el Union Mount

## **Un inciso sobre los volúmenes**

- Comparten ficheros del anfitrión en el contenedor
- Se saltan el Union Mount
- Puede compartirse y reutilizarse entre contenedores

## **Un inciso sobre los volúmenes**

- Comparten ficheros del anfitrión en el contenedor
- Se saltan el Union Mount
- Puede compartirse y reutilizarse entre contenedores
- Puede ser compartido sin estar en ejecución el contenedor

## **Un inciso sobre los volúmenes**

- Comparten ficheros del anfitrión en el contenedor
- Se saltan el Union Mount
- Puede compartirse y reutilizarse entre contenedores
- Puede ser compartido sin estar en ejecución el contenedor
- Existen hasta que no haya ningún contenedor que lo use

# Servidor web Nginx 1.11.6

```
FROM nginx:1.11.6
```

```
COPY nginx.conf /etc/nginx/
```

```
COPY web.conf /etc/nginx/conf.d/
```

```
RUN echo "upstream php-upstream { server app:9000; }" >  
/etc/nginx/conf.d/upstream.conf
```

```
VOLUME /app
```

```
EXPOSE 80
```

```
CMD ["nginx"]
```

## Fichero docker-compose.yml (1/2)

```
version: '2'  
services:  
    app:  
        build: docker/php  
        volumes:  
            - ./app:/app  
    server:  
        build: docker/nginx  
        ports:  
            - "80:80"  
        volumes_from:  
            - app  
    ...
```

## Fichero docker-compose.yml (2/2)

```
...  
db:  
  image: mysql:5.7.16  
  ports:  
    - "3306:3306"  
  environment:  
    MYSQL_ROOT_PASSWORD: root  
    MYSQL_DATABASE: web  
    MYSQL_USER: web  
    MYSQL_PASSWORD: password  
  volumes:  
    - ./docker/mysql/:/docker-entrypoint-initdb.d  
mailcatcher:  
  image: schickling/mailcatcher  
  ports:  
    - "1080:1080"
```

```
$ docker-compose build  
$ docker-compose up  
[Ctrl-C]  
$ docker-compose up -d
```

## Puntos de acceso desde el anfitrión

- Nginx `http://localhost/`
- Aplicación `http://local.dev/`
- Mailcatcher `http://local.dev:1080/`
- MySQL `localhost:3306`

**Compose** crea una red por defecto para la aplicación, cada servicio se une a esta red y son accesibles y detectable entre ellos usando el nombre del servicio

```
$ docker network ls
$ docker network inspect dockerfordevelopment_default
$ docker-compose exec app /bin/bash
# ping db
[Ctrl-C]
# ping server
[Ctrl-C]
```

```
$ docker-compose ps
$ docker-compose logs
$ docker-compose logs db
$ docker-compose run app /bin/bash
$ docker-compose stop
$ docker-compose start
$ docker-compose stop
$ docker-compose ps
$ docker-compose rm
$ docker-compose ps
```

# **¡Manos a la obra!**

## **Aplicación de ejemplo**

**<https://github.com/aleherse/docker-for-development>**

# Ventajas de Docker en desarrollo

- Mismo código, Sistema Operativo y aplicaciones

# Ventajas de Docker en desarrollo

- Mismo código, Sistema Operativo y aplicaciones
- Más ligero que una máquina virtual

# Ventajas de Docker en desarrollo

- Mismo código, Sistema Operativo y aplicaciones
- Más ligero que una máquina virtual
- Independiente de la máquina del desarrollador

# Ventajas de Docker en desarrollo

- Mismo código, Sistema Operativo y aplicaciones
- Más ligero que una máquina virtual
- Independiente de la máquina del desarrollador
- Es muy fácil cambiar versiones de software (PHP 5.6 a PHP 7)

# Ventajas de Docker en desarrollo

- Mismo código, Sistema Operativo y aplicaciones
- Más ligero que una máquina virtual
- Independiente de la máquina del desarrollador
- Es muy fácil cambiar de versión (PHP 5.6 a PHP 7)
- Es trivial reconstruir el entorno

# Ventajas de Docker en desarrollo

- Mismo código, Sistema Operativo y aplicaciones
- Más ligero que una máquina virtual
- Independiente de la máquina del desarrollador
- Es muy fácil cambiar de versión (PHP 5.6 a PHP 7)
- Es trivial reconstruir el entorno
- Si modificas el contenedor actualiza el Dockerfile

# ¿Por dónde continuar?

- Documentación oficial ([docs.docker.com](https://docs.docker.com))
- Imágenes en **Docker Hub** ([hub.docker.com](https://hub.docker.com))
- **The Docker book** ([dockerbook.com](https://dockerbook.com))
- Búsqueda de **Docker** en **YouTube**
- Crea contenedores para tus proyectos más sencillos
- Usar docker para testeo y producción

**¿Preguntas?**