

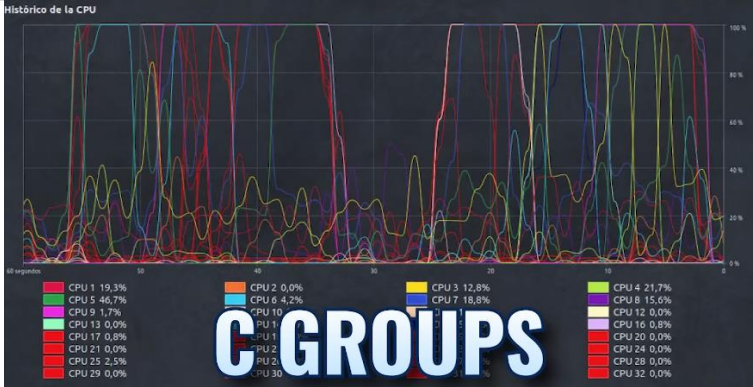
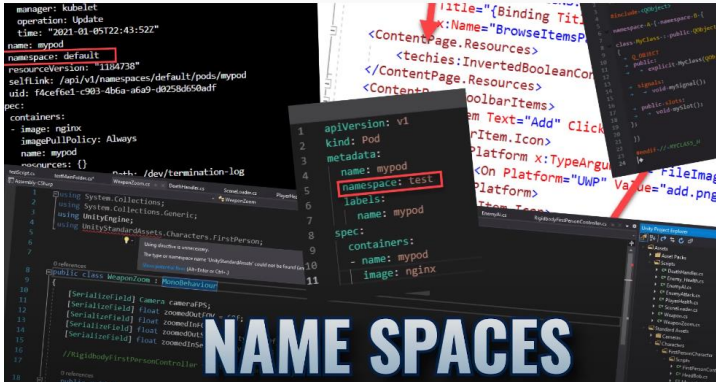
# CONTENEDORES ----CIBERSEGURIDAD



# Linux Containers



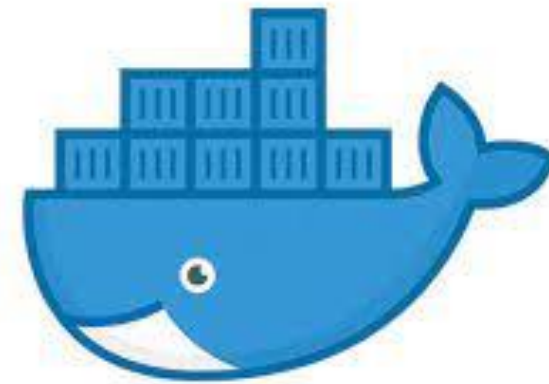
# LXC



# Contenedores

**Contenedor = namespaces + cgroups + chroot**

1. **Namespaces:** Vistas de los recursos del SO
2. **Cgroups:** Limitan y miden los recursos del SO
3. **Chroot:** Cambia el root directory de un proceso



# Contenedores

## Aisla con namespaces

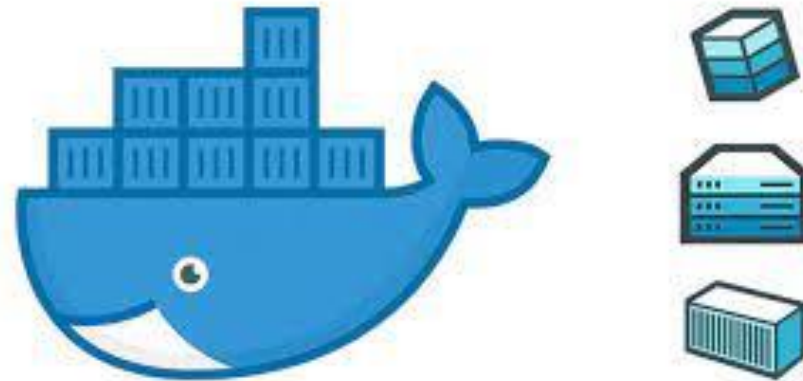
- PID, network, mount, IPC, users, etc.

## Limita los recursos con cgroups

## Se estandarizó el runtime

- runC con la iniciativa OCI
- o CoreOS está haciendo rocket
- o Hyper es otro (runV) con VMs y un “mini kernel”

## Docker es más que un runtime





# Portabilidad

Un contenedor Docker se puede **desplegar en cualquier otro sistema** (que soporte esta tecnología).



# Autosuficiencia

No contiene todo un sistema completo, sino únicamente aquellas librerías, archivos y configuraciones necesarias para desplegar las funcionalidades que contenga.



# Ligereza

```
lenovo@DESKTOP-MRE8HNL MINGW64 ~  
$ docker images  
REPOSITORY          TAG         IMAGE ID      CREATED        SIZE  
miho1a              latest     fb676eacdefe  4 hours ago   117MB  
ubuntu              latest     216c552ea5ba  3 days ago    77.8MB  
alpine/git          latest     692618a0d74d  5 weeks ago   43.4MB  
hello-world         latest     feb5d9fea6a5  12 months ago 13.3kB  
centos              latest     5d0da3dc9764  12 months ago 231MB  
mikesplain/openvas  9          f3e8ed228230  3 years ago   5.33GB  
mikesplain/openvas  latest     889967897c49  3 years ago   6.39GB
```

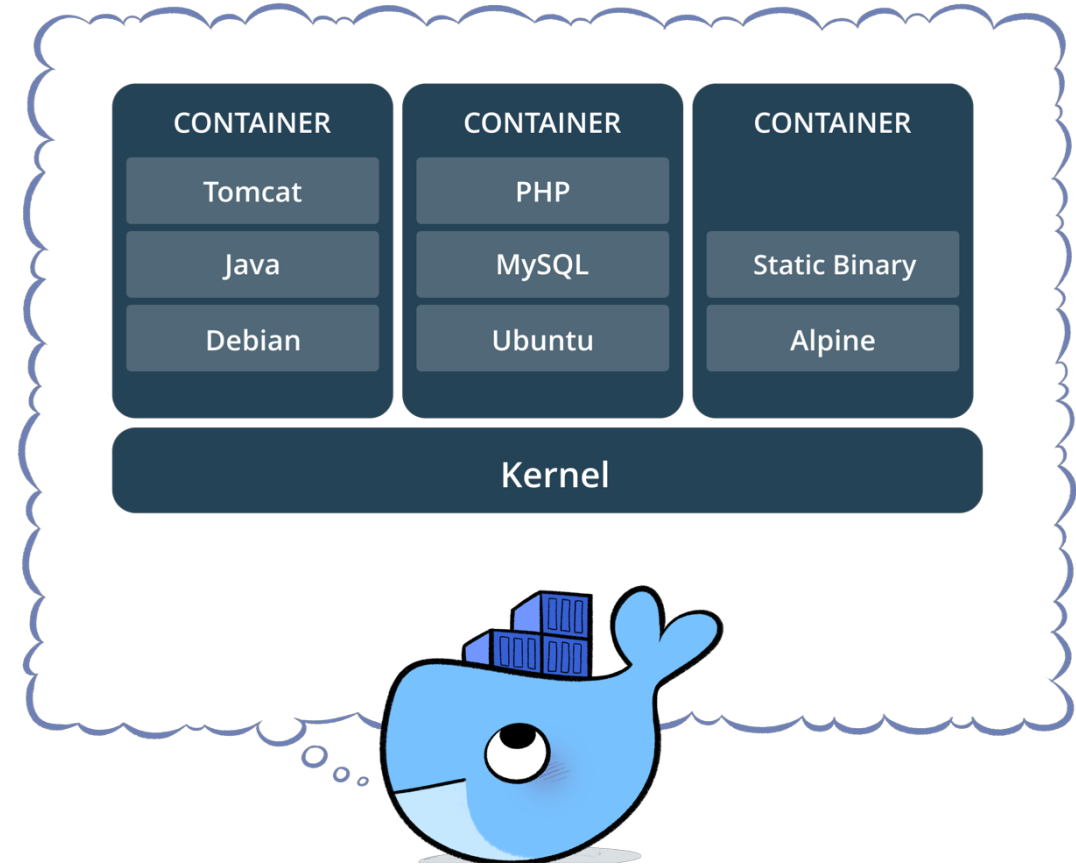


## Ventajas destacables.

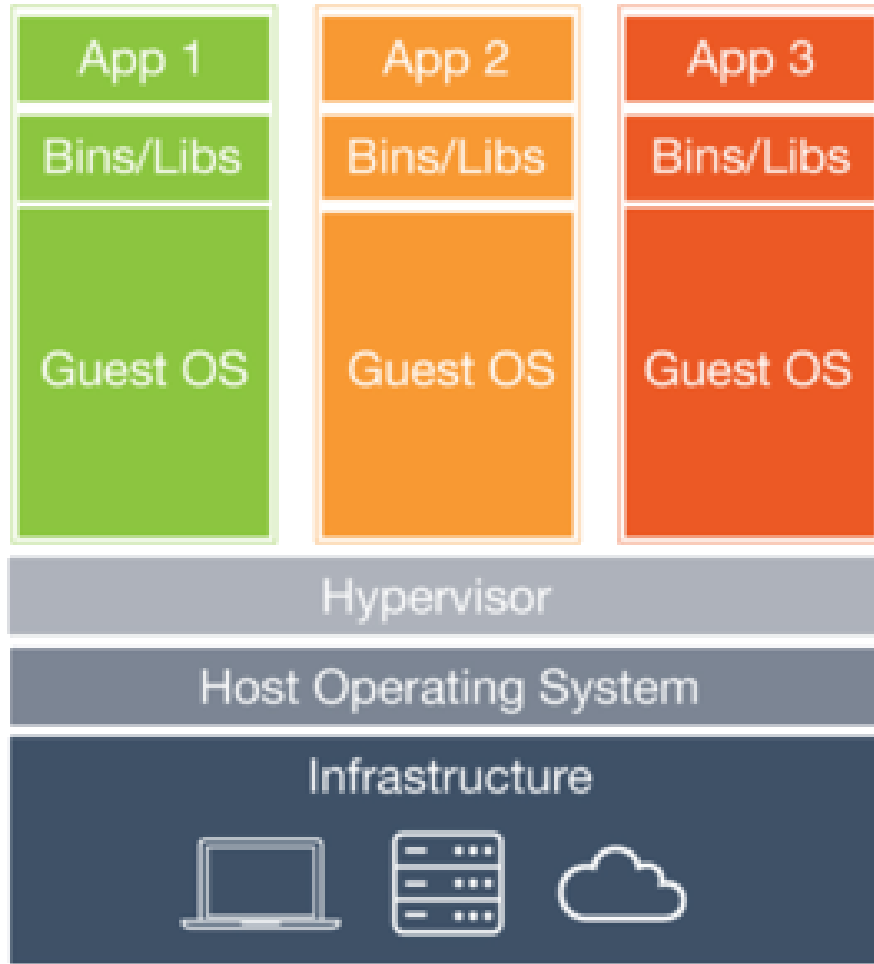
- Bajos recursos de hardware
- Entorno de trabajo aislado
- Rápido despliegue
- Múltiples entornos de desarrollo
- Reutilización
- Utilización de arquitecturas de microservicios

## ¿Qué es un contenedor?

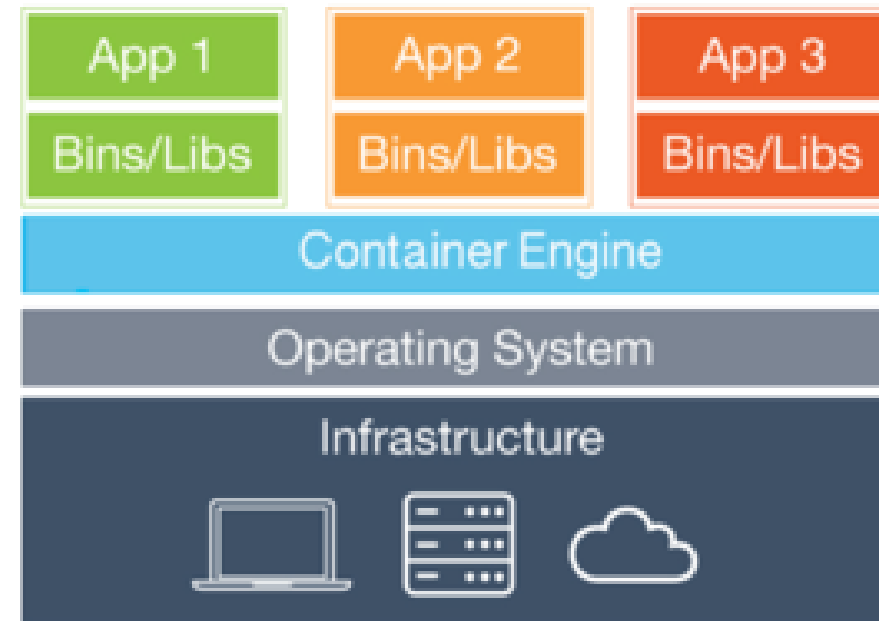
- Una forma de empaquetar software en un formato que incluye todo lo necesario para hacerlo funcionar y se ejecuta aislado del resto de la máquina



# Contenedores VS MV



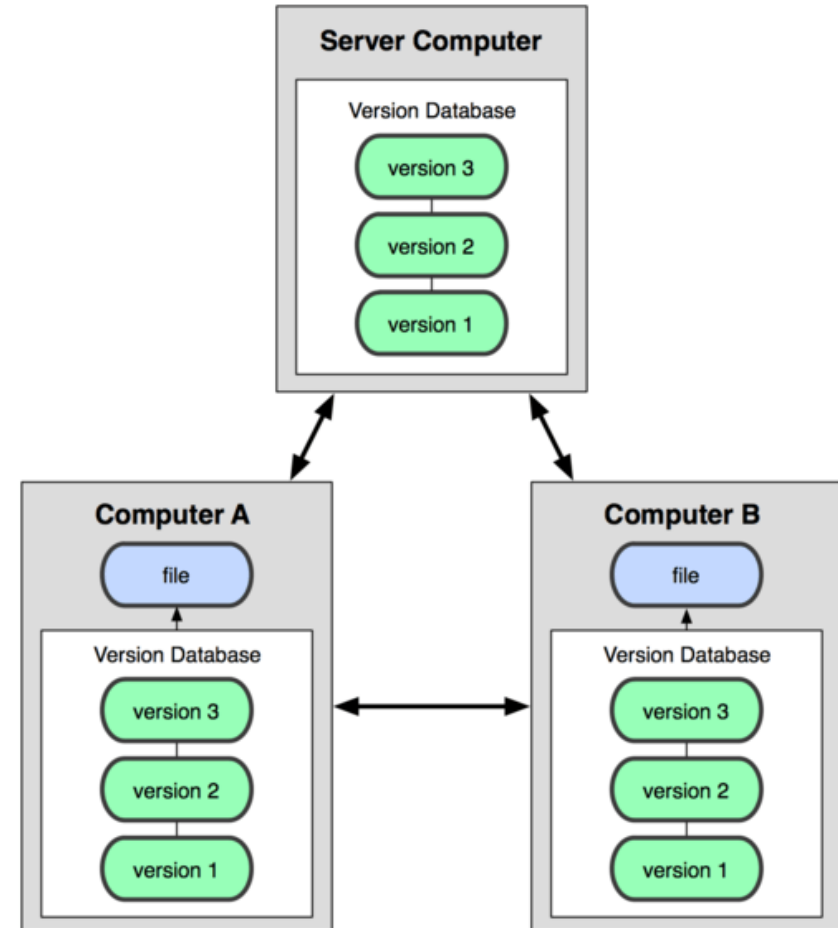
Hypervisor-based Virtualization



Container virtualization



# CONTEXTO





# Monolithic

User Interface

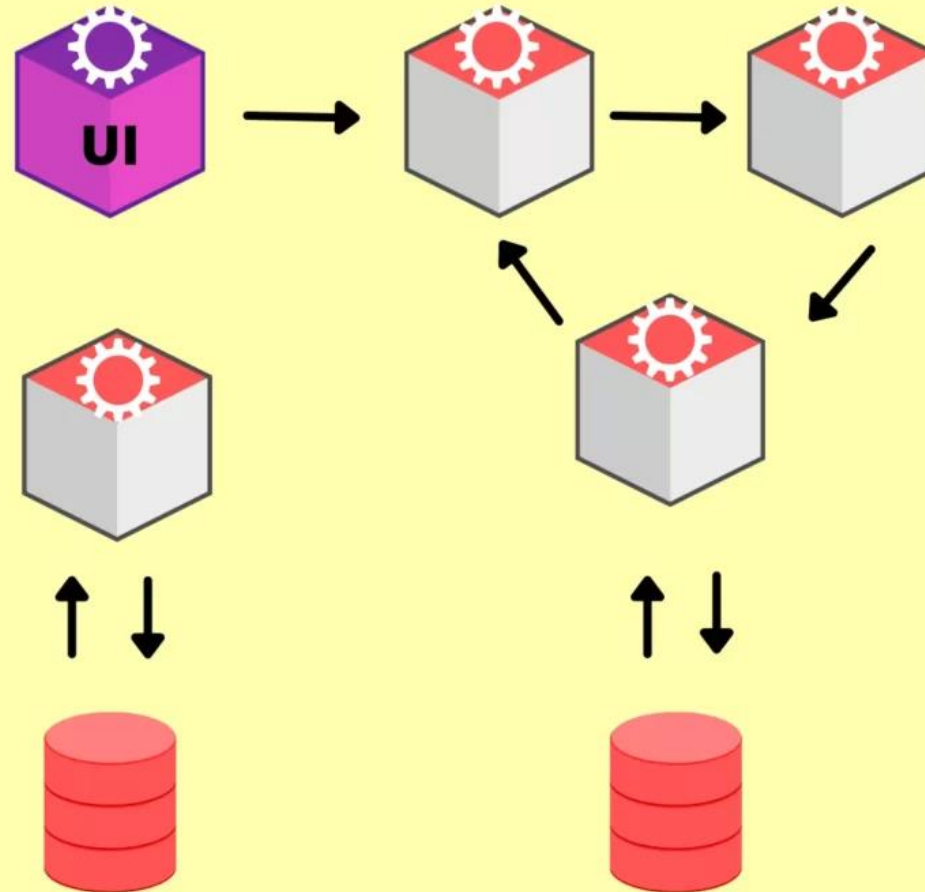
Business Logic

Data Access  
Layer



# Microservices

© DevOpsPod

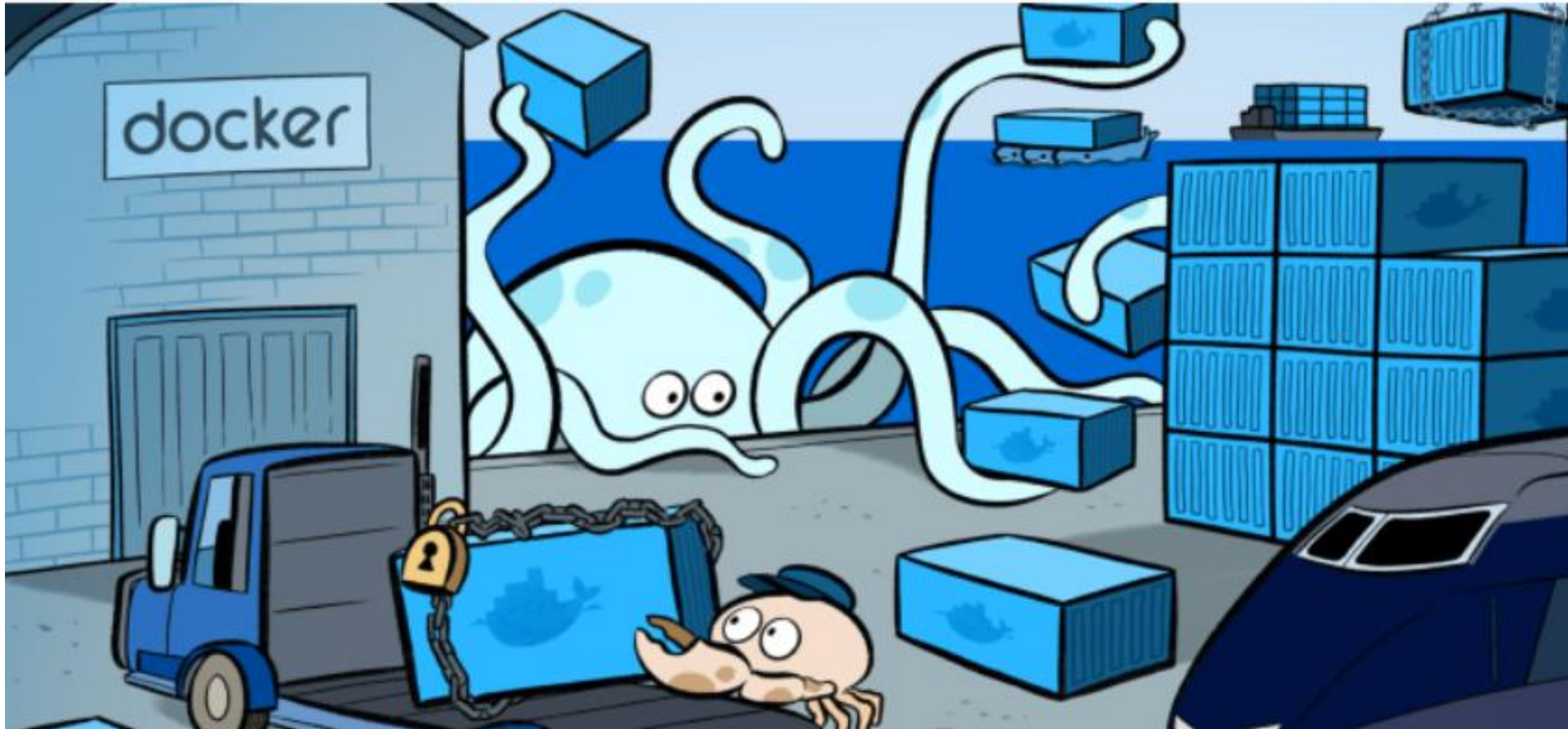




<https://github.com/Netflix/chaosmonkey>

# ¿Qué es Docker?

Software para la gestión de contenedores



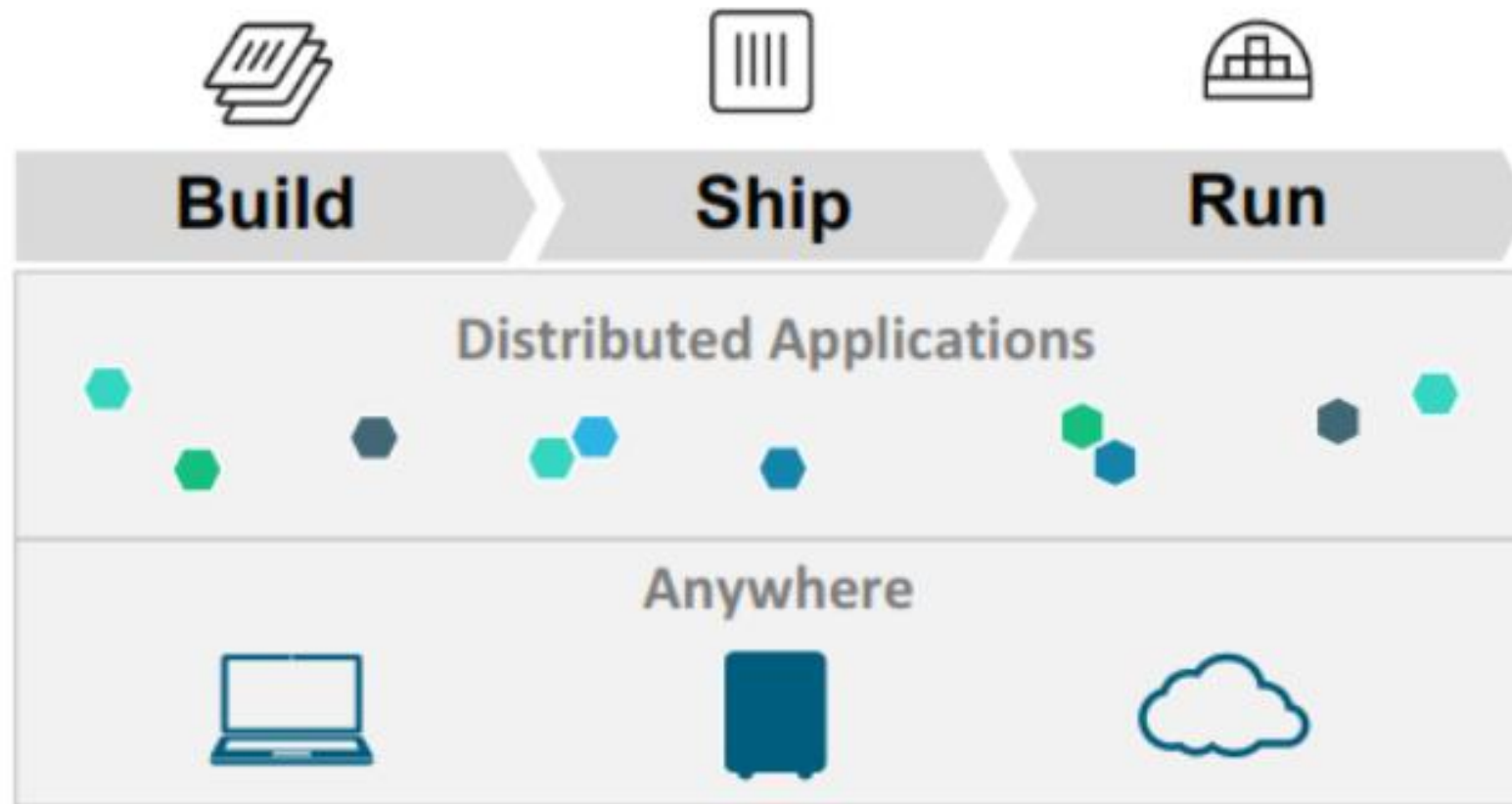




2014 (go)



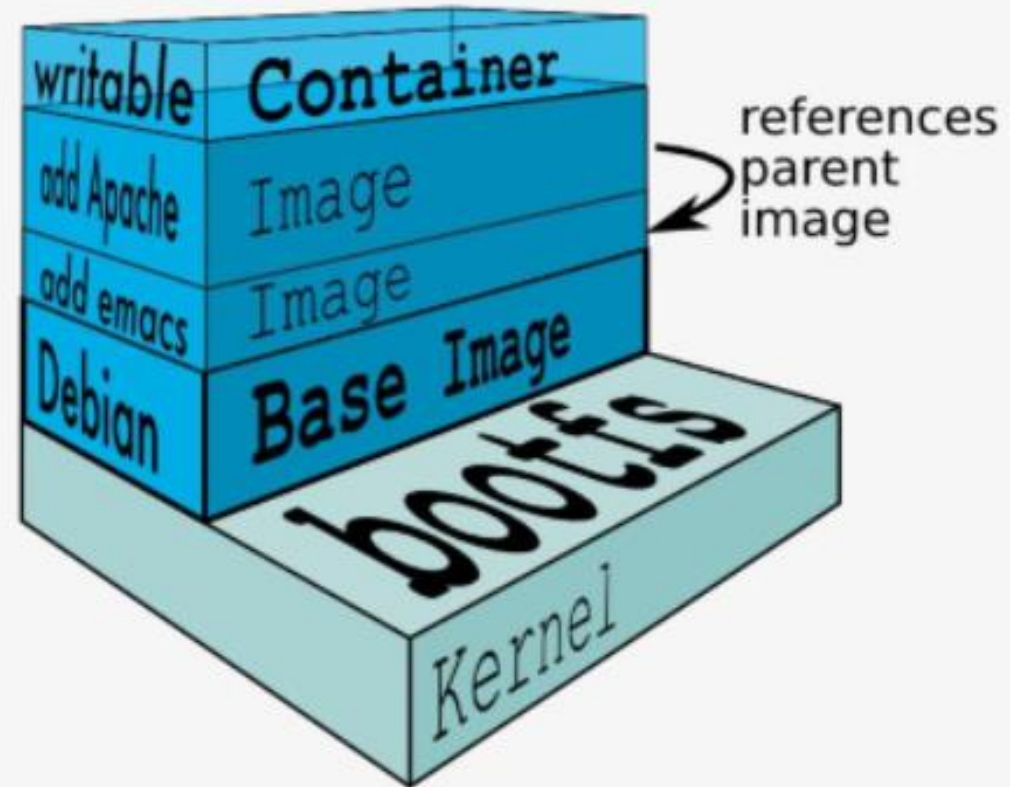
# La misión de Docker



*“Build, ship and run any app anywhere”*

# Contenedores e imágenes

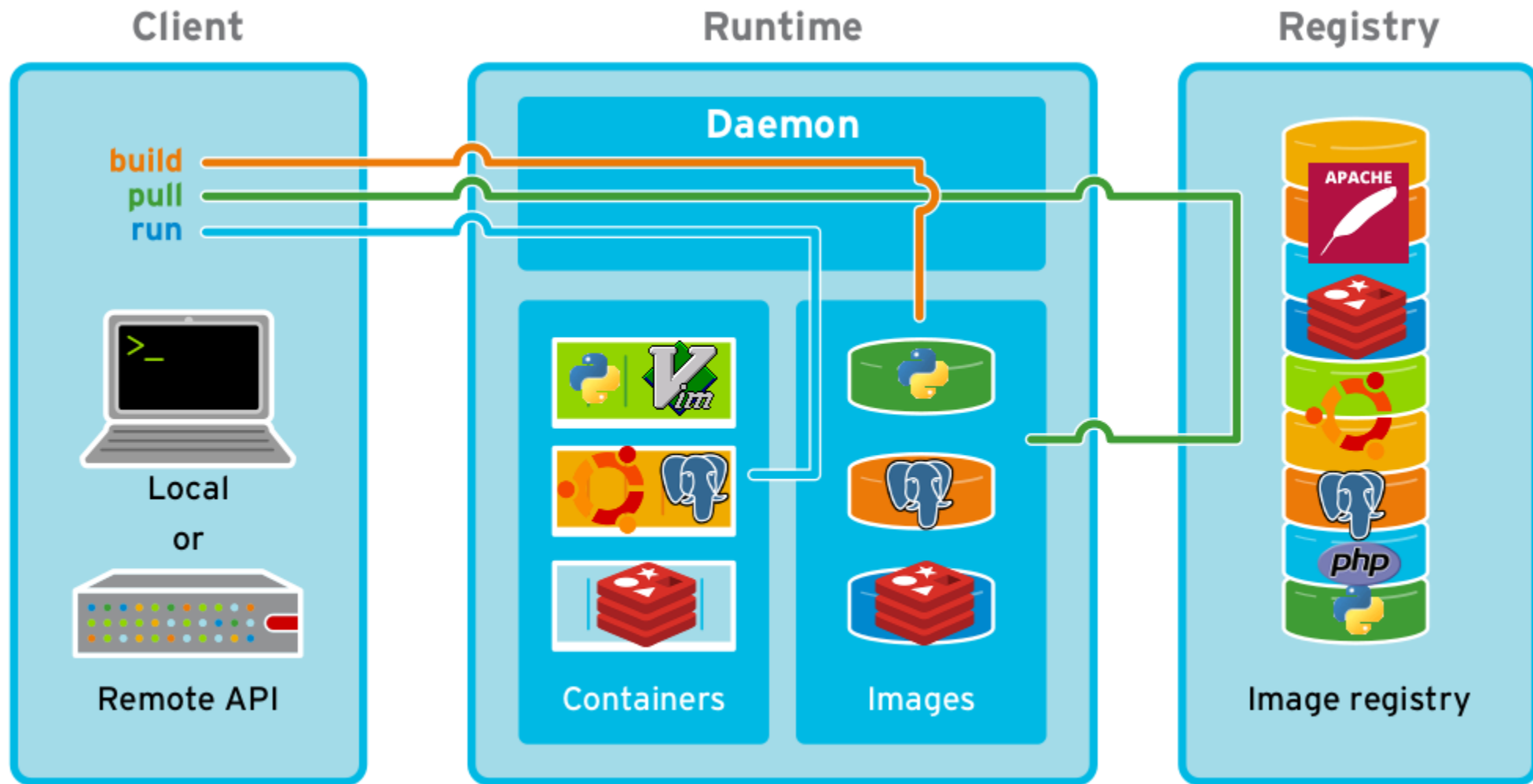
- Imágenes
  - Plantilla de sólo lectura para crear nuestros contenedores
  - Creadas por nosotros u otros usuarios de la comunidad
  - Se pueden guardar en un registro interno o público
- Contenedores
  - Aplicación aislada
  - Contiene todo lo necesario para ejecutar nuestra aplicación
  - Basados en una o más imágenes.



# Conceptos Fundamentales

- **Imagen:** Es un modelo de lo que se desea construir. Ejemplo: Srver LAMP (Ubuntu + Apache + php + MySQL).
- **Contenedor:** Instancia de una imagen. Puede tener varias copias de la misma imagen en ejecución.
- **Dockerfile:** Receta para crear una imagen. Los Dockerfiles contienen una sintaxis especial de Docker. Es un documento de texto que contiene todos los comandos que un usuario puede utilizar para ensamblar una imagen.
- **Commit:** Al igual que Git, los contenedores Docker ofrecen control de versiones. Puede guardar el estado de su contenedor Docker en cualquier momento **como una nueva imagen**, y añadiendo una **nueva capa**.
- **DockerHub / Image Registry:** Lugar donde la gente puede publicar imágenes de docker públicas (o privadas) para facilitar la colaboración y el intercambio.
- **Layer:** modificación de una imagen existente, representada por una instrucción en el Dockerfile. Las capas se aplican en secuencia a la imagen base para crear la imagen final.

# Arquitectura de Docker

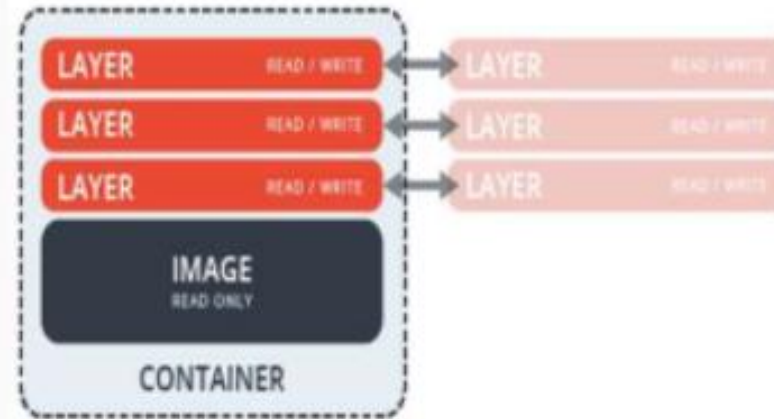


# Componentes arquitectura Docker

- 1.Docker daemon:** Es el motor principal de Docker que ejecuta los contenedores y gestiona los recursos del sistema.
- 2.Docker client:** Es la interfaz de línea de comandos que los desarrolladores utilizan para interactuar con el Docker daemon. Los comandos de Docker client se envían al Docker daemon para ejecutar acciones como crear, iniciar o detener contenedores.
- 3.Imágenes de Docker:** Son plantillas de archivos que contienen todo lo necesario para ejecutar una aplicación, incluyendo el código, las dependencias y la configuración. Las imágenes de Docker se utilizan para crear contenedores.
- 4.Contenedores de Docker:** Son instancias de imágenes de Docker que se ejecutan como procesos aislados en el sistema operativo host. Cada contenedor tiene su propio sistema de archivos y recursos, lo que permite que varias aplicaciones se ejecuten en el mismo host sin interferir entre sí.
- 5.Docker registry:** Es un repositorio centralizado de imágenes de Docker que permite a los desarrolladores compartir y distribuir sus imágenes de Docker. Docker Hub es el registro de imágenes de Docker más popular y contiene una gran cantidad de imágenes públicas y privadas.
- 6.Servicios de Docker:** Son una forma de definir y escalar aplicaciones compuestas por varios contenedores. Los servicios de Docker permiten que los contenedores se distribuyan y escalen automáticamente en diferentes hosts.

# Flujo en Docker

## Container



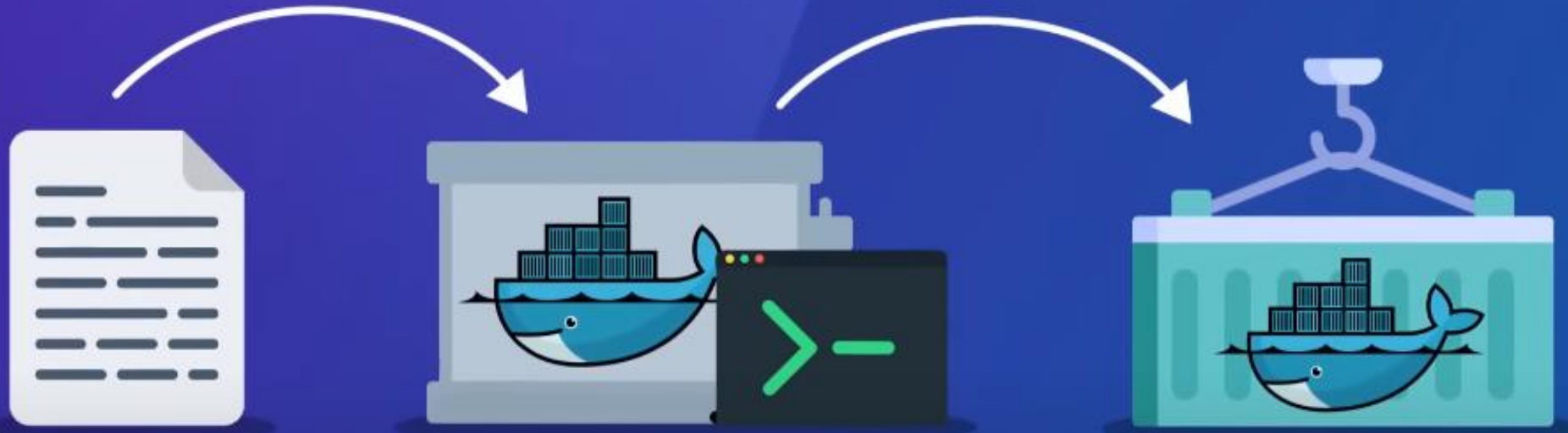
## Image



## Registry







**DOCKER FILE**

**DOCKER IMAGE**

**DOCKER CONTAINER**

FROM ...

COPY ...

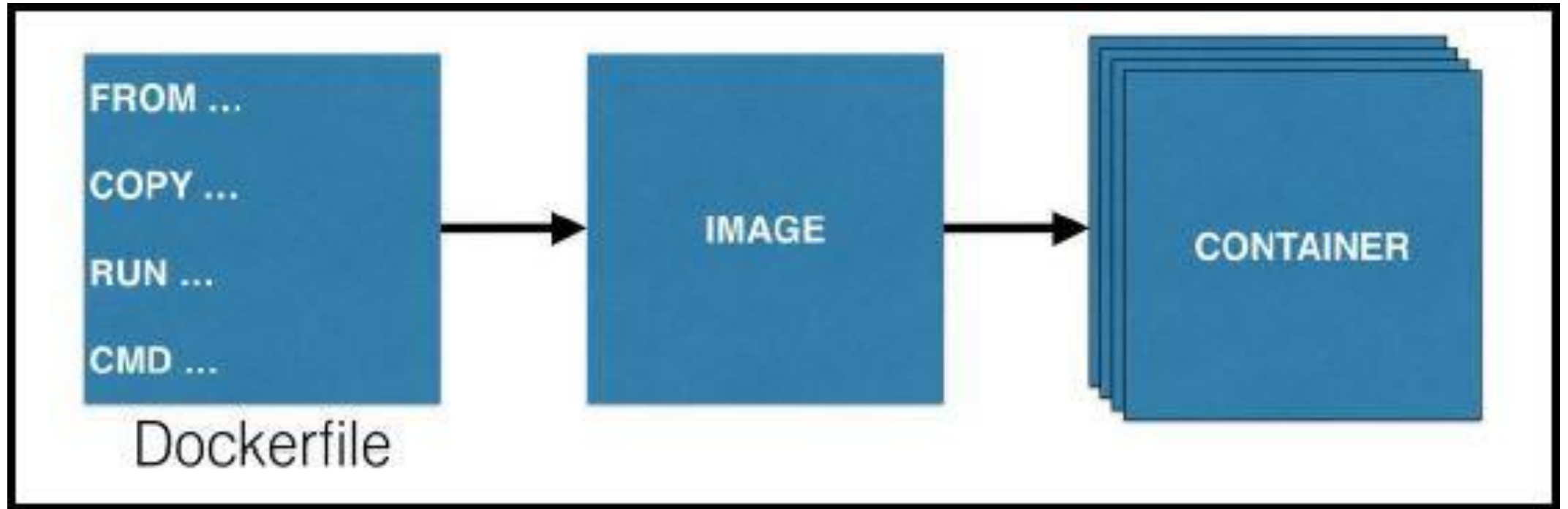
RUN ...

CMD ...

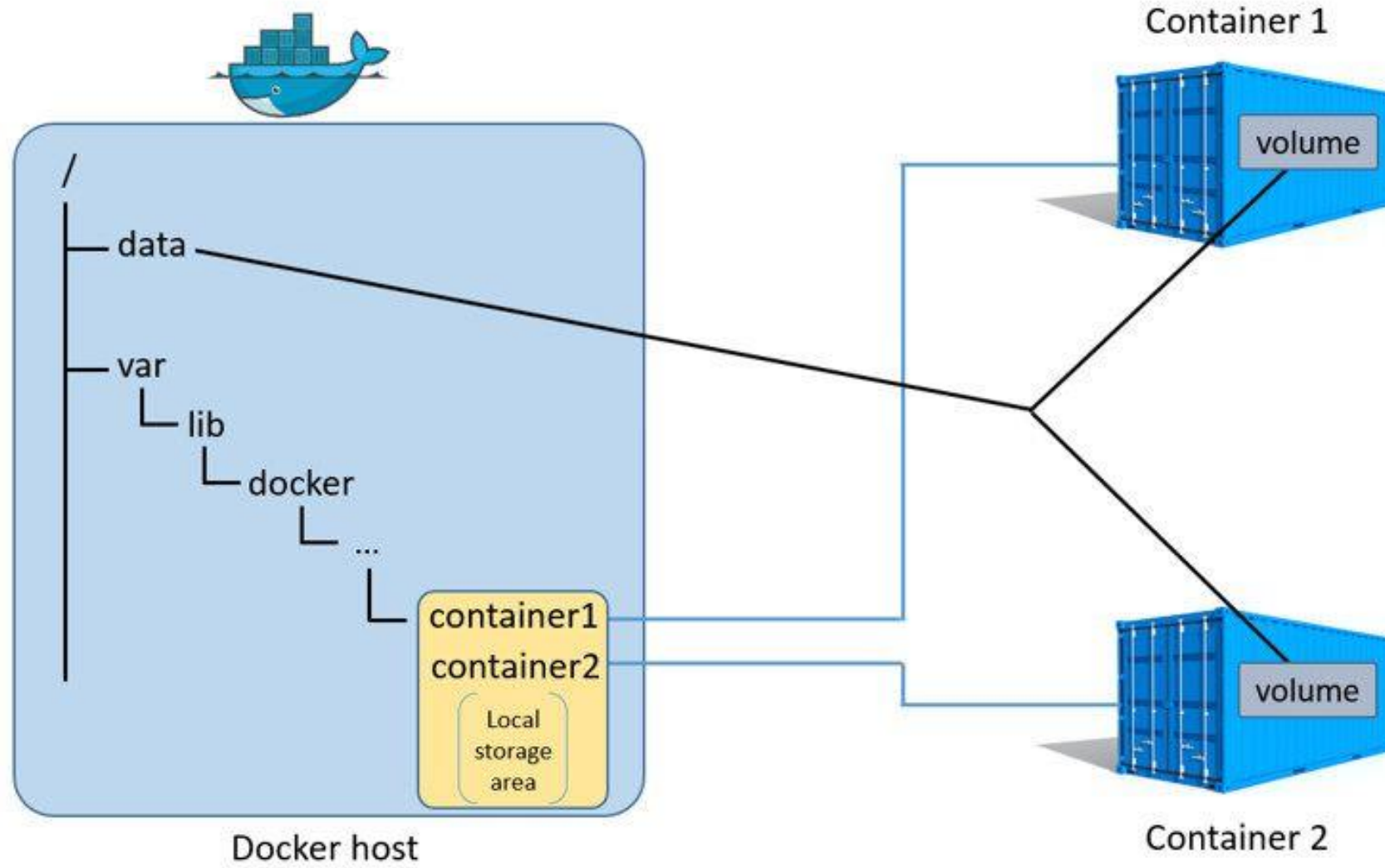
Dockerfile

IMAGE

CONTAINER



## Escritura en Docker





## Volúmenes


- Un directorio designado en el contenedor en el cual se persiste información independientemente del ciclo de vida del contenedor.
- Los cambios en un volumen son excluidos cuando se guarda una imagen
- La información se persiste aunque se elimine el contenedor
- Pueden estar mapeados a un directorio del host.
- Pueden compartirse entre contenedores

# DOCKERHUB

← → ↻ 🏠 [hub.docker.com/r/vulnerables/web-dvwa](https://hub.docker.com/r/vulnerables/web-dvwa) 🔍 🗑️ 📄 ⭐ 🏠 📺 📁 👤

 **docker hub** 🔍  Explore Repositories Organizations Help ▾ [Upgrade](#)  **jaidier** ▾

Explore > [vulnerables/web-dvwa](#)



## vulnerables/web-dvwa ☆

By [vulnerables](#) • Updated 4 years ago

Damn Vulnerable Web App (DVWA) is a PHP/MySQL web application that is damn vulnerable.

[Image](#)

[Overview](#) [Tags](#)


### Damn Vulnerable Web Application Docker container

docker pulls **8.4M** License [GPL](#)

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal

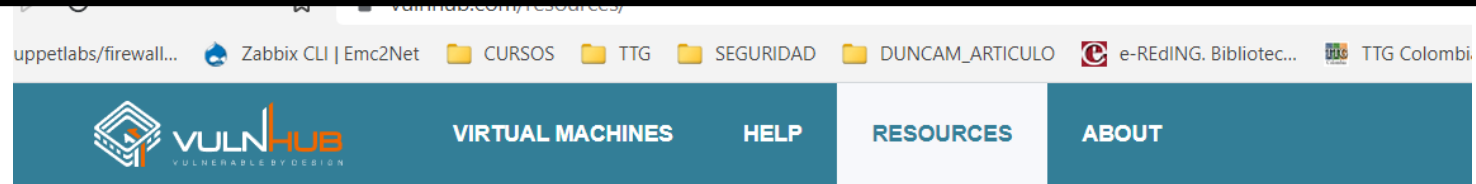
#### Docker Pull Command

```
docker pull vulnerables/web-dvwa
```



<https://hub.docker.com/r/vulnerables/web-dvwa>

# https://www.vulnhub.com/



## RESOURCES

### Building VMs

- DCAU7: [Guide to Building Vulnerable VMs](#)
- FalconSpy: [Creating Boxes for Vulnhub](#)
- Techorganic: [Creating a virtual machine hacking challenge](#)
- Donavan: [Building Vulnerable Machines: Part 1 — An Easy OSCP-like Machine](#)
- Donavan: [Building Vulnerable Machines: Part 2 — A TORMENT of a Journey](#)
- Donavan: [Building Vulnerable Machines: Part 3 — JOY is More Than One \(Machine\)](#)

### Community VulnHub Resources

- Google Doc: [NetSecFocus Trophy Room](#)
- GitHub: [Ignitetechnologies/CTF-Difficulty](#)
- GitHub: [Ignitetechnologies/Vulnhub-CTF-Writeups](#)
- GitHub: [Ignitetechnologies/Linux-Privilege-Escalation](#)
- GitHub: [Ignitetechnologies/Privilege-Escalation](#)
- GitHub: [Ignitetechnologies/Web-Application-Cheatsheet](#)

(Free) Virtual Networks (VPNs) + Custom Personal Targets



















# vulhub

vulhub / vulhub Public Sponsor Watch 554 Fork 3.7k

[Code](#) [Issues 11](#) [Pull requests 15](#) [Actions](#) [Projects 2](#) [Wiki](#) [Security](#) [Insights](#)

master 69 branches 0 tags Go to file Add file Code

 **phith0n** update the discord link ✓ 27b64cd 20 hours ago 🕒 1,784 commits


 .github	bump hadolint to v2.6.0	13 months ago
 activemq	fix lint for all Markdowns	15 months ago
 airflow	remove unused port exposing	5 months ago
 apereo-cas/4.1-rce	add Chinese introducing	2 years ago
 apisix/CVE-2020-13945	fixed for markdown lint	8 months ago
 appweb/CVE-2018-8715	fix lint for all Markdowns	15 months ago
 aria2/rce	fix lint for all Markdowns	15 months ago
 base	Updated opentsdb with the right urls to the packages and generating s...	27 days ago
 bash/CVE-2014-6271	renamed shellshock to <a href="#">CVE-2014-6271</a>	5 months ago
 celery/celery3_redis_unauth	added a workflow to check CRLF in the text files ( <a href="#">#295</a> )	13 months ago
 cgi/CVE-2016-5385	renamed shellshock to <a href="#">CVE-2014-6271</a>	5 months ago
 coldfusion	fix lint for all Markdowns	15 months ago
 confluence	fixed style	3 months ago
 couchdb	fix lint for all Markdowns	15 months ago
 discuz	fix lint for all Markdowns	15 months ago


**About**


Pre-Built Vulnerable Environments Based on Docker-Compose


[vulhub.org](#)


[docker](#) [dockerfile](#) [docker-compose](#) [vulnerability-environment](#) [vulhub](#)

 Readme


 MIT license

 11.8k stars

 554 watching

 3.7k forks


**Sponsor this project**

 **phith0n** Owen Gong

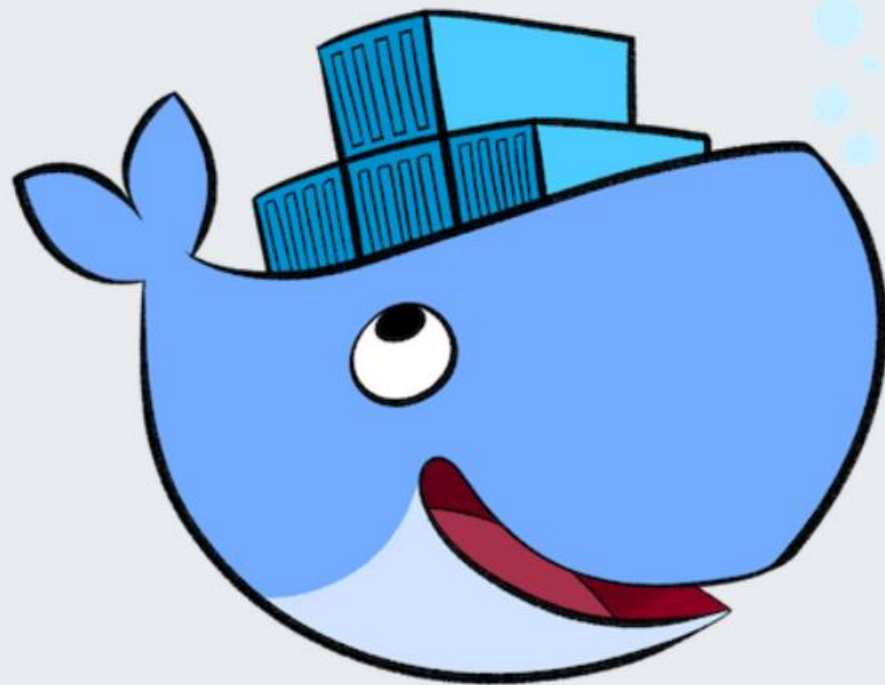
Sponsor

[Learn more about GitHub Sponsors](#)

**Contributors** 45



<https://github.com/vulhub/vulhub>



# Play with Docker

A simple, interactive and fun playground to learn Docker

Login ▼

## Comandos comunes

Comando	Descripción
docker version	Información sobre la instalación realizada
docker run	Crea un contenedor a partir de una imagen
docker ps	Lista de contenedores en funcionamiento
docker inspect	Información acerca de un contenedor o imagen
docker start/stop	Arranca o para un contenedor
docker rm/rmi	Borra un contenedor (rm) o borra una imagen (rmi)
docker cp	Copiar ficheros de dentro de un contenedor (lanzado)
docker exec	Ejecuta comandos dentro de un contenedor
docker logs	Nos da información (logs) acerca de un contenedor
docker stats	Estadísticas de los contenedores

# First Step!



A terminal window with a dark background and light-colored text. The text 'Hello world!' is displayed in a large, bold, monospace font. The background of the terminal has a subtle grid pattern.

es "Hello world!"

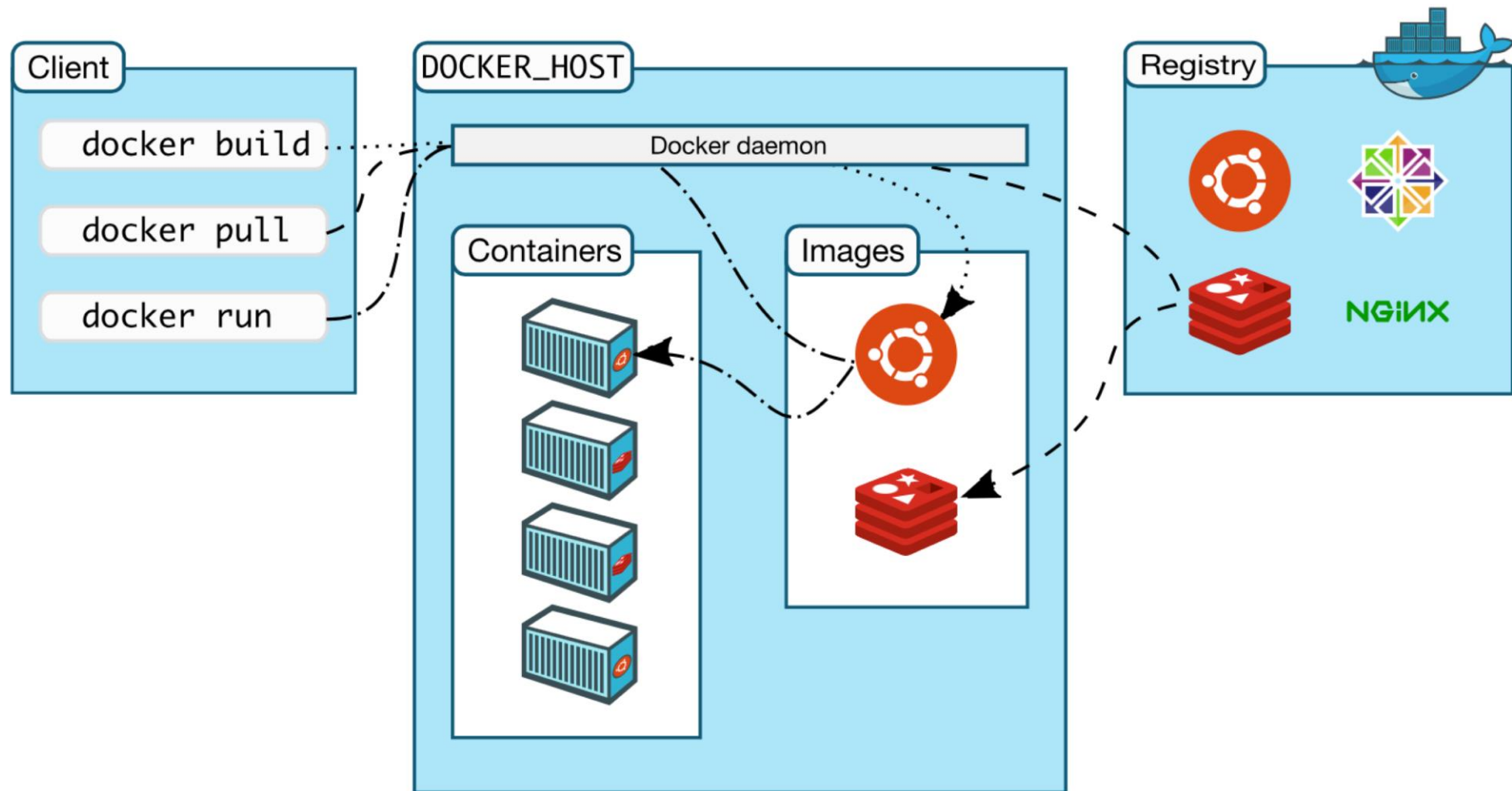
```
> docker run hello-world
```

Docker CLI

Lanzar una imagen

Nombre de la imagen

¿Qué pasó?





# Construcción de imágenes



Las imágenes son construidas a través del comando **docker build** y mediante la utilización de un fichero denominado **Dockerfile**

# Comandos utilizados en la construcción de Dockerfiles

- **FROM:** Inicia el sistema de ficheros y provee de un gestor de paquetes específico de la distribución empleada.
- **RUN:** ejecuta un comando.
- **VOLUME:** Define un volumen.
- **WORKDIR:** directorio de trabajo del contenedor
- **COPY <origen> <destino>:** copia ficheros dentro de la imagen.
- **EXPOSE:** Define puertos expuestos por contenedor.
- **ENTRYPOINT:** define el comando por *defecto* que ejecuta el contenedor al iniciar.
- **CMD:** Define distintos argumentos de la instrucción usada en el *entrypoint*.

# Ejemplo de Dockerfile

Este Dockerfile emplea una base de Python 3.9, instala las dependencias necesarias a partir de un archivo requirements.txt.

Expone el puerto 5000 para que la aplicación pueda ser accedida desde el exterior.

Finalmente, el comando CMD ejecuta la aplicación Python a través de app.py.

```
# Imagen base de Python
FROM python:3.9-slim-buster

# Directorio de trabajo en el contenedor
WORKDIR /app

# Copiar los archivos necesarios al contenedor
COPY requirements.txt .
COPY app.py .

# Instalar las dependencias especificadas en el archivo requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# Exponer el puerto en el que se ejecuta la aplicación
EXPOSE 5000

# Comando para iniciar la aplicación
CMD ["python", "app.py"]
```

**Reto.**

**Validar el anterior ejemplo**

### Otro ejemplo

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

Interactivo

Comando a ejecutar

### Otro más


Puerto local : Puerto contenedor

```
> docker run -p 8000:80 -d  
kitematic/hello-world-nginx
```

En segundo plano

## Otro más

```
> docker run -p 8010:80 -d -v  
/c/Users/me/Desktop/nginx_files:/website_files  
kitematic/hello-world-nginx
```



En Linux y Mac hay que poner la ruta completa.  
Si se usa Windows 10 hay que poner C:/Users...

Edita el index.html que ha aparecido en nginx\_files y prueba cómo se actualiza dinámicamente

# Dockerfile

**FROM debian:Jessie**

→ Define la imagen base de la que empezamos

**RUN apt-get update && \**  
**apt-get install -y nginx**

→ Corre un comando en el container

**COPY index.html /var/www/html/**  
**EXPOSE 80**

→ Copia algo del contexto al container

→ Dice que este container expone ese puerto

**CMD ["nginx", "-g", "daemon off;"]**

→ Qué correr al levantar el container



# Armando nuestra primer imagen



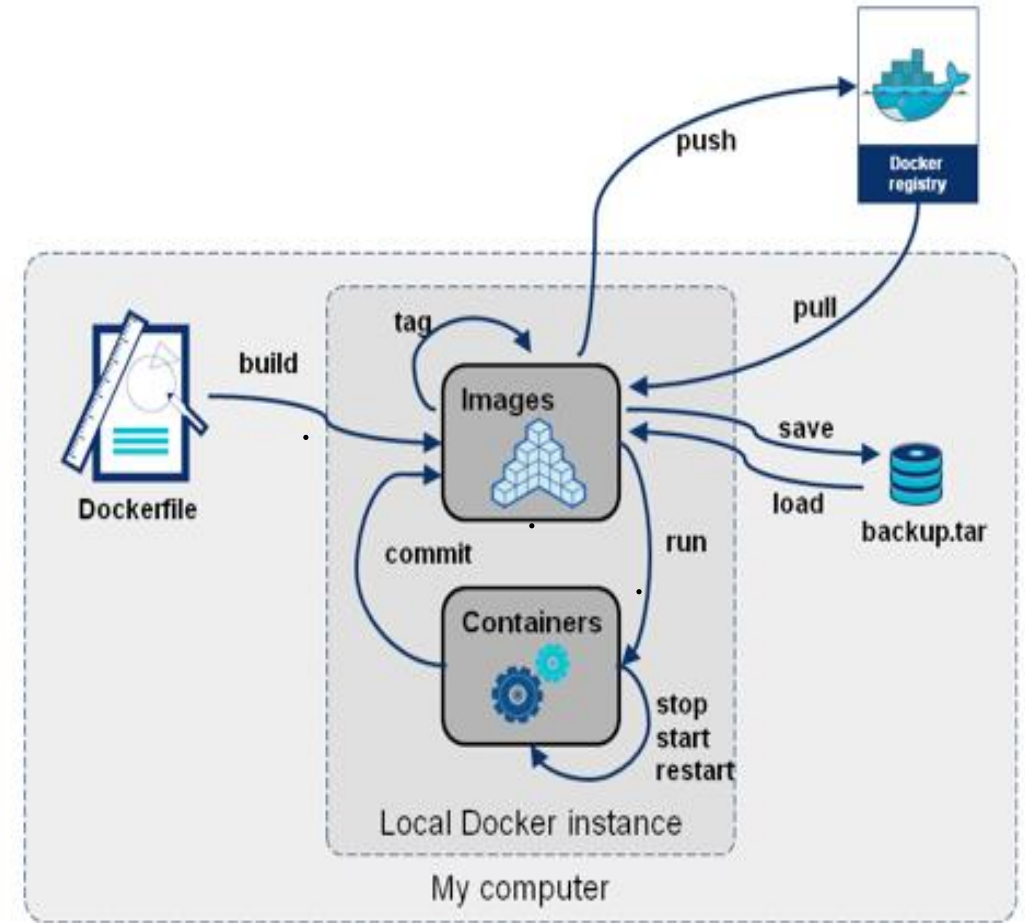
```
$ docker build -t workshop .  
$ docker run -P workshop  
$ docker ps -a
```

# CONTENERIZANDO APLICACIONES

Contenerizar ( dockerizar) es **implementar sobre un contenedor para empaquetar una aplicación** (software), para luego distribuirla y ejecutarla a través de los contenedores.

Beneficios:

- **Creación y distribución de código.**
- **Adopción de cultura DevOps**
- **Construcción de arquitecturas orientadas a microservicios.**



## ¿Qué queremos conseguir?

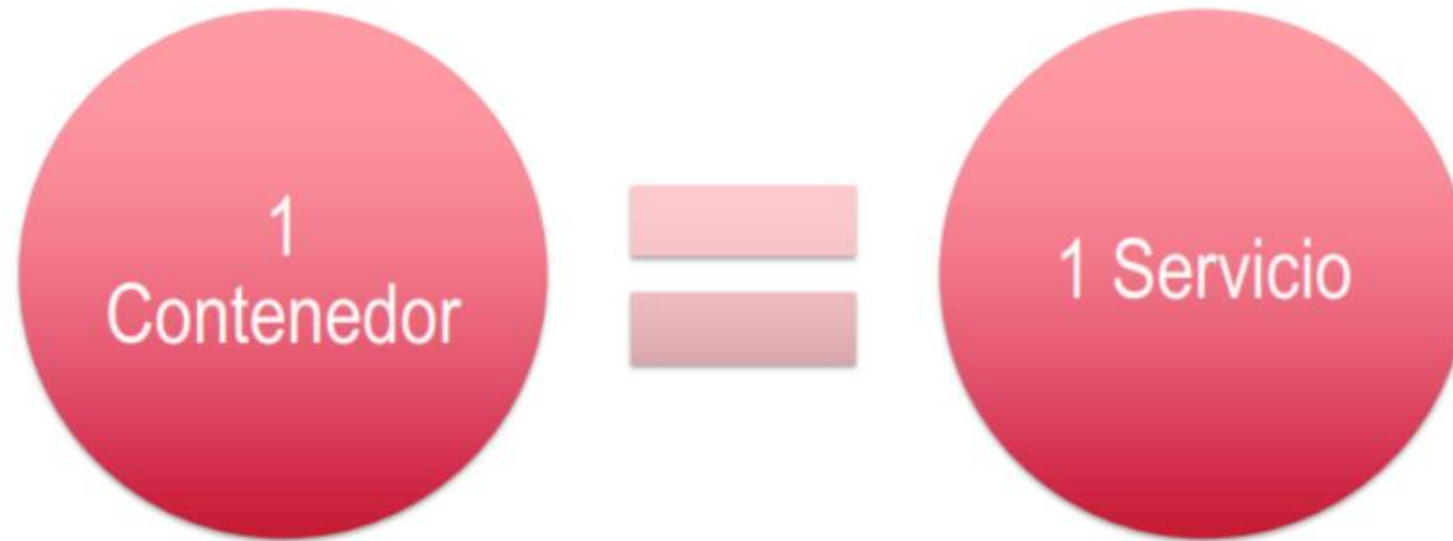
- Tener empaquetada nuestra aplicación y sus dependencias en una imagen para poder desplegarla donde queramos simplemente con

```
> docker run miAplicacion
```

## Pasos para Dockerizar una aplicación



# Principio de responsabilidad única





# Greenbone

[Greenbone Cloud Service TRIAL](#) [Greenbone Enterprise TRIAL](#) [Buy Here](#) [Contact](#) [Blog](#) [Germany](#)

[Products](#) [Cyber Resilience](#) [Customer Services](#) [About Gre](#)

## Supply Chains in Open-Source Software

Not everyone who uses open source software follows all the best practices. We help to avoid mistakes.

[Read more](#)

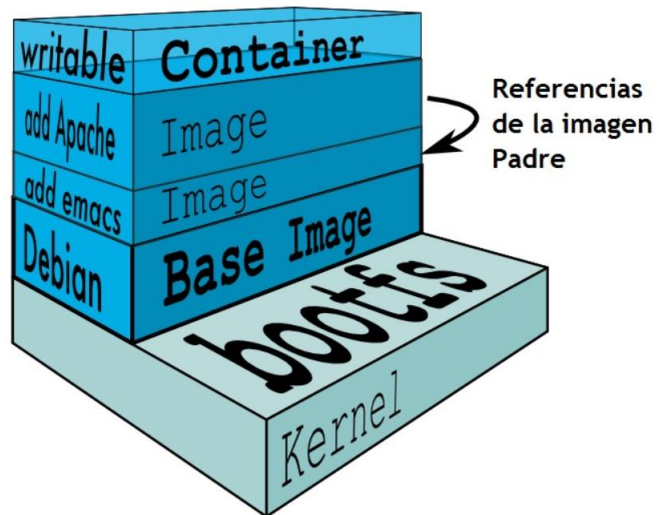


```

(root@kali)-[/home/kali/DOCKER]
# nano docker-compose.yml

(root@kali)-[/home/kali/DOCKER]
# docker-compose up -d
Creating network "docker_default" with the default driver
Pulling php (php:8-fpm)...
8-fpm: Pulling from library/php
7a6db449b51b: Pull complete
ad2afdb99a9d: Pull complete
dbc5aa907229: Extracting [=====] 64.62MB/91.6MB
82f252ab4ad1: Download complete
8d88582e93e0: Download complete
c07b7e794fcd: Download complete
ea981381696a: Download complete
9c67e25f918f: Download complete
6556af059564: Download complete
84d98907de8e: Download complete

```



root@kali: /home/kali/DOCKER

```

version: '3'
services:
  web:
    image: nginx:latest
    ports:
      - "8080:80"
    links:
      - php
  php:
    image: php:8-fpm

```

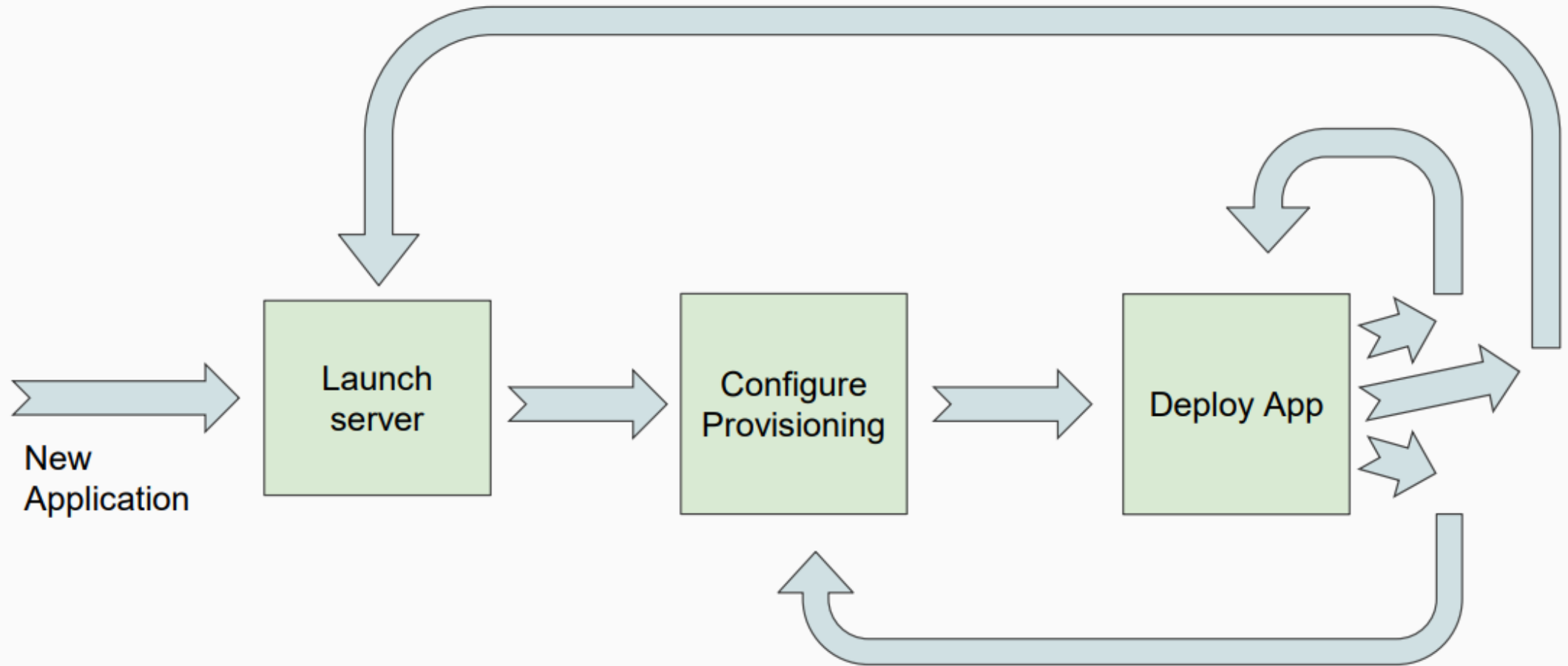


# Instalación de Docker sobre Kali Linux

```
kali@kali:~$ sudo apt update
kali@kali:~$
kali@kali:~$ sudo apt install -y docker.io
kali@kali:~$
kali@kali:~$ sudo systemctl enable docker --now
kali@kali:~$ apt install docker-compose
kali@kali:~$ sudo usermod -aG docker $USER
```

```
kali@kali:~$ docker pull docker.io/kalilinux/kali-rolling
kali@kali:~$
kali@kali:~$ docker run --tty --interactive kalilinux/kali-rolling
└─(root@e4ae79503654)-[/] └─# └─(root@e4ae79503654)-[/] └─# exit
kali@kali:~$
```

# Deployment tradicional



# Consecuencias de esta forma

- Servers mutables
  - Posibles diferencias entre ambientes y/o diferentes nodos
- Puede haber conflictos entre aplicaciones si usan el mismo servidor
  - En general lleva a un server por app y desperdicio de recursos
- El escalado suele ser “manual” o complicado de automatizar por completo
- Desarrolladores dependenden del sector de operaciones

# Deployment con containers

## Ventajas

- Devs más libertades e involucrados en la definición de infraestructura
- Menos dependencia con equipo de “Operaciones”
- Permite monitoreo de recursos más preciso
- Fácilmente adaptable a buenas prácticas como 12 factor app

## Desventajas

- Más cosas para manejar
  - Complicado hacerlo manualmente o con las herramientas tradicionales
- Otro layer de abstracción no trivial
- Dificulta el deployment de aplicaciones que mantienen estado

# Publicar imagen en Docker Hub

Un registro debe poseer la siguiente sintaxis para considerarse válido.

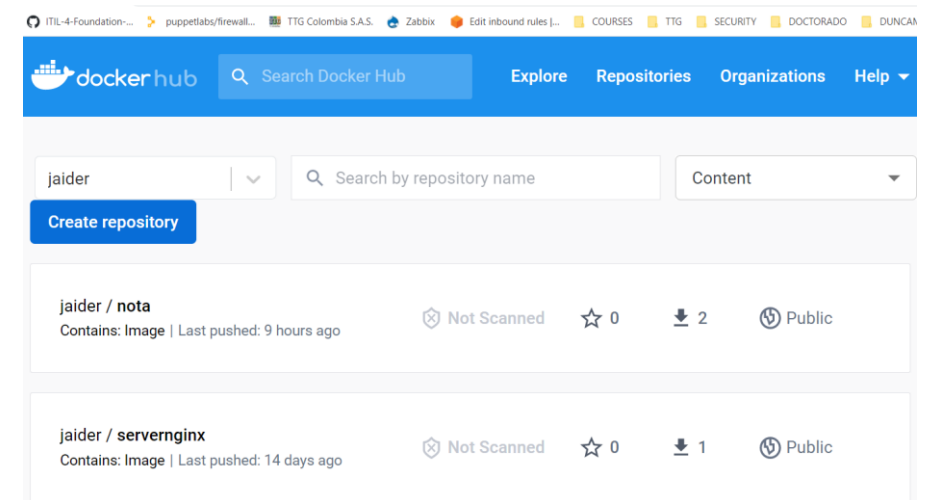
**nombre\_de\_usuario/nombre\_del\_repositorio:etiqueta**

Renombrar la imagen.

***\$docker tag ID\_imágen jaider/miapp:v1***

Push hacia dockerhub.

***\$docker push jaider/miapp:v1***



```
$ docker volume ls
DRIVER      VOLUME NAME
local       4ea5eeaea20756386dfd34d848fd87d93ed708d01be00f265091034021b3255c
local       5a2782f766f6c640970e9c5d072f76101d4e798ddc79a49ebe8f255c0c294e7d
local       98ac483d1e23219117a35d66e8362812f4b963609732e2be1a0b3f9764a1f624
local       684ab5c36b4f7dbde5c3686f44e9579682a64b720fe9803e77f0fcb234567a55
local       960f54776025c6482e5e403b87d3e407f78fdff3dd37619938349d1fe7d968ac
local       9861d99edb5659554ebd76d09bc84ac09bf6563cb8cd4553a136eada2207480b
local       d6eb6b5c61e0a389c11e2d245c7a6adf107c9d5a21bfd5b235afaf9dc71a1bc2
local       dafee0c634a8b564d8e9185c5a4ba44de95267acc5e105caace65a073e4de747
local       docker4drupal_codebase
local       e0f537dd3b8af99511535e4f2d59a00bf9e1d74cc9f2d93050ad4a7d3520aaee
local       f4850bb064a3efe7406c4d25e9393ab34b04b4b104e5c0adb4f80765edf85560
local       volume-awesome-compose-dreamy_hypatia
local       vsCodeServerVolume-awesome-compose-dreamy_hypatia-web
```

← Anónimos

← Nombrados

```
mongo-express
  image mongo-express
  ports
    80:8081
  environment
    ME_CONFIG_BASICAUTH_USERNAME "jaider"
    ME_CONFIG_BASICAUTH_PASSWORD "password"
  links
    mongo
mongo
  image mongo
  volumes
    mongo-data:/data/db
```



Preguntas?



Gracias...Totales.

# Recursos

- Cursos: – Laboratorios virtuales gratuitos: <http://training.play-with-docker.com/> – Cursos gratuitos oficiales: <http://training.docker.com/category/self-paced-online>
- Libros: – Docker Cookbook: <http://shop.oreilly.com/product/0636920036791.do> – Using Docker: <http://shop.oreilly.com/product/0636920035671.do> – Docker: Up & Running: <http://shop.oreilly.com/product/0636920036142.do>

Fuentes:

<https://1984.lsi.us.es/wiki-egc/images/egc/7/7c/Presentacion.pdf>

<https://es.slideshare.net/restorando-devs/nerdearla-2016-docker-workshop>