



# Introducción

Docker Open Source

## **Docker Open Source:** Productos Docker disponibles bajo licencias de código abierto

- **Docker Open Source:**

- Docker Engine
  - Containers
  - CLI
  - Daemon
  - ...
- Docker Build
- Docker Compose

- **Docker Inc:**

- Docker Desktop
- Docker Hub
- Docker Scout
- ...

[Documentación Docker](#)

# Índice

1. Analogía del transporte marítimo
2. Docker Engine
  1. Contenedores
  2. Imágenes
  3. Volumen
  4. Networking
3. Docker Compose
4. Docker Build: Dockerfiles  
(=> Sesión 3)



# 1. Analogía

Transporte marítimo

# 1. Analogía del transporte marítimo

El término docker, en inglés, significa «estibador», que es la persona encargada de realizar la carga y descarga de un buque u otros medios de transporte

## **Transporte marítimo:**

- Normativa ISO, donde se establecen las medidas, tamaño y forma de los contenedores.
- Pueden ser transportados en cualquier embarcación que cumpla el estándar ISO.

## **Software:**

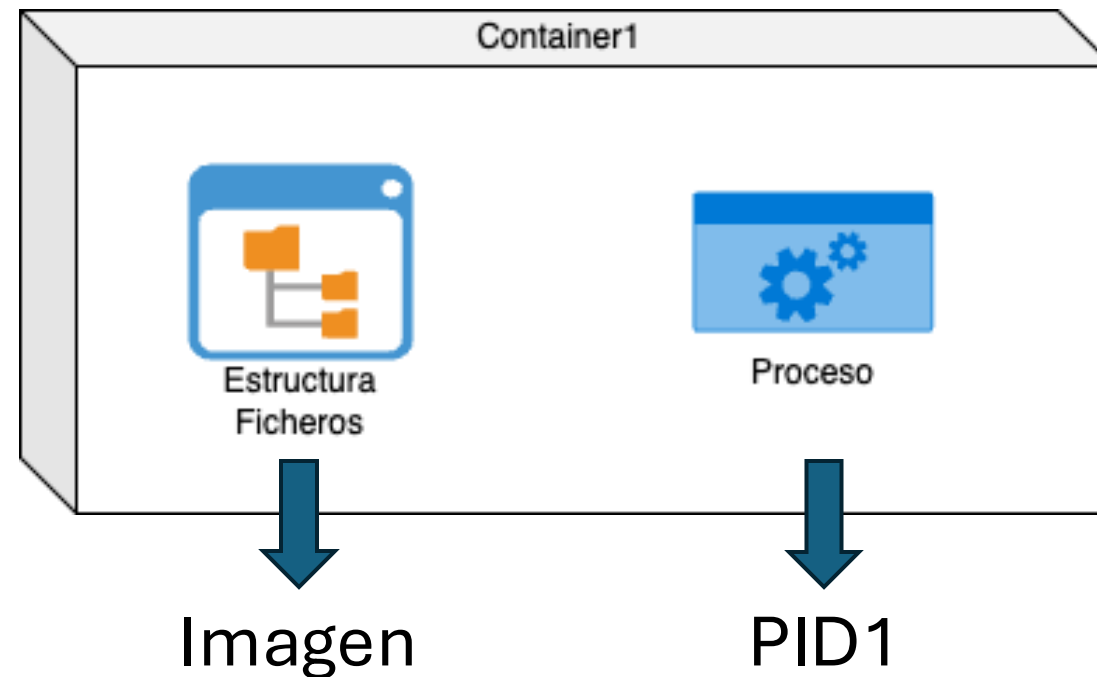
- Normativa OCI (Open Container Initiative).
- Pueden ser ejecutados en cualquier runtime que cumpla el estándar OCI.



## 2. Docker Engine

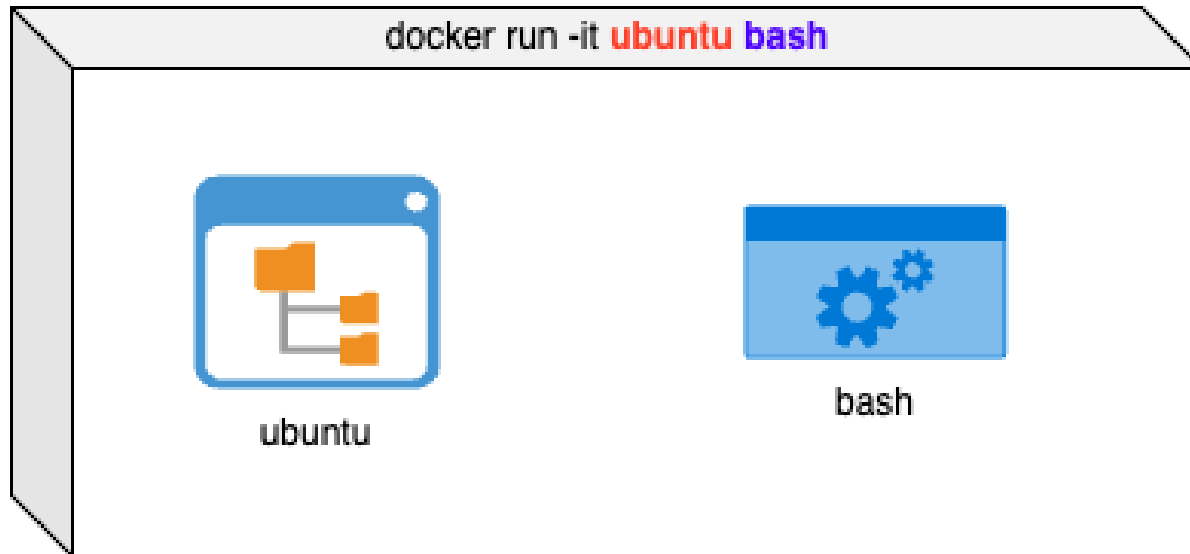
## 2. Docker Engine Contenedores

Es una forma de empaquetar **código y dependencias** de una aplicación en un formato estándar que **permita su ejecución**.



## 2. Docker Engine Contenedores

**\$ docker run -it ubuntu bash**



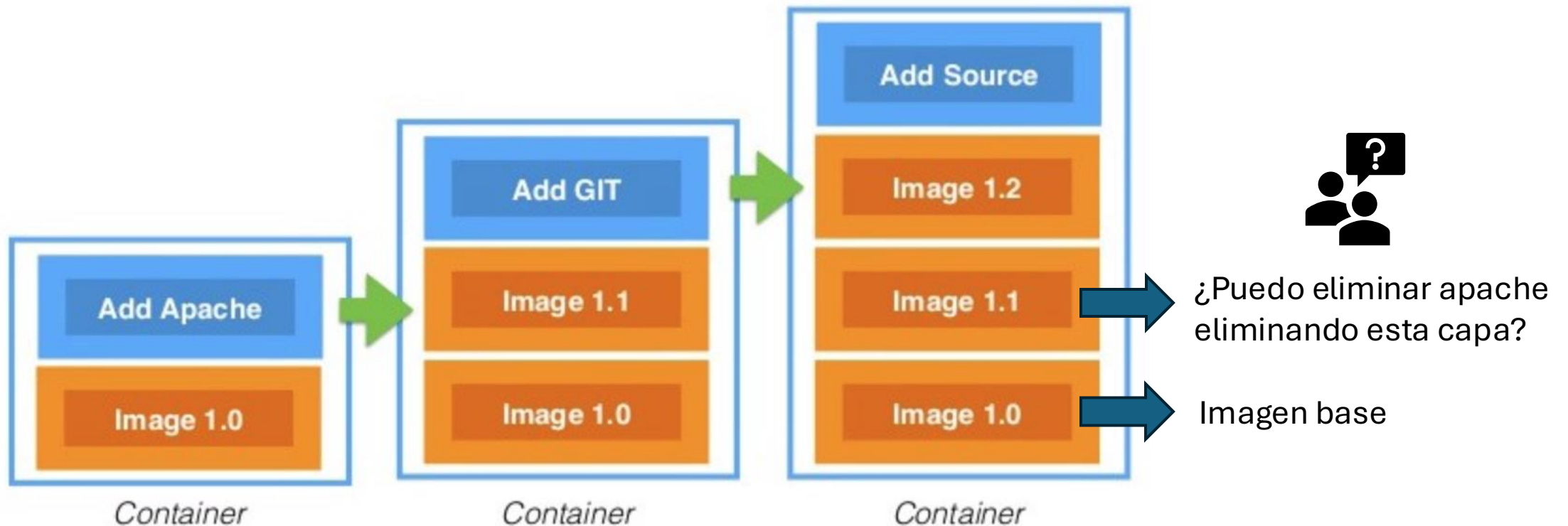
- Ficheros y binarios Ubuntu:
  - `/opt`
  - `/usr/local/bin`
  - `Chmod`, `apt`, ...
- PID1: `bash`
  - Al ser el PID1, si este proceso se detiene, el contenedor muere
  - Igual que con `systemd` en un SO Ubuntu



## 2. Docker Engine

### Imagen

Plantilla que contiene todo lo necesario para ejecutar una aplicación: código, config, bin ... Se compone de capas **inmutables** que representan cambios en la imagen.





# DOCKER IMAGE LAYERS

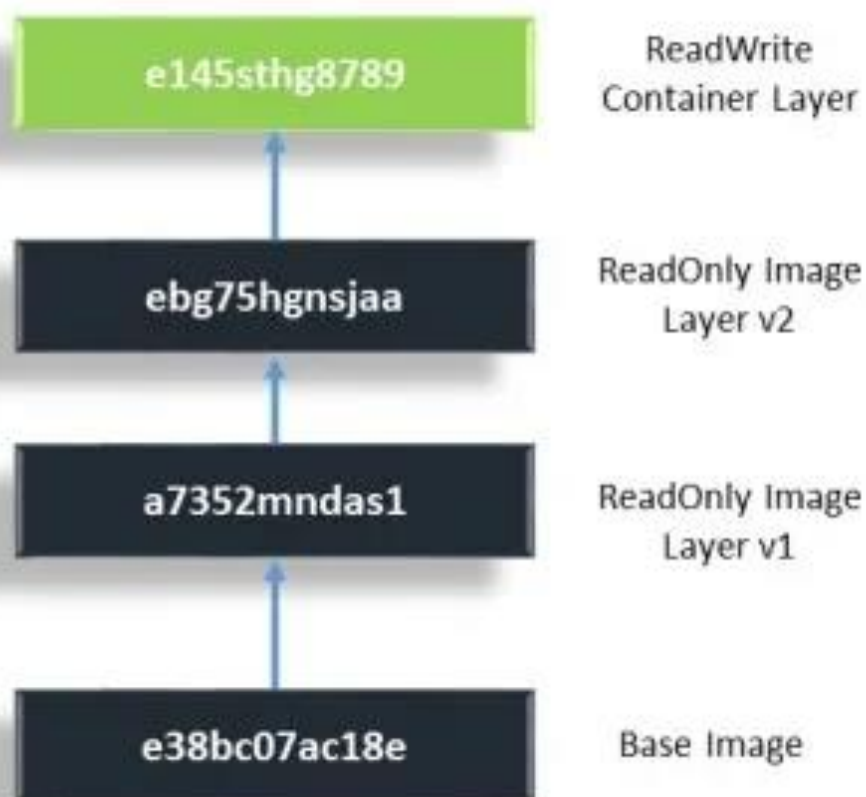
www.learnitguide.net

1. Started Image Layer 3 as a container and accessible by users

1. Started Image Layer v1 as a container.  
2. Installed and Configured https web server.  
3. Committed new layer v2

1. Started Base Image (**docker.io/centos**) as a container.  
2. Package Updated on Base Image using "yum update".  
3. Committed new layer v1

Pulled CentOS image from Docker Hub using docker pull command. **Repo: docker.io/centos**



## 2. Docker Engine

### Imagen


Hay diferentes formas de crear imágenes:

- Descargar de Docker Registry
- A partir de un contenedor existente
- A partir de un fichero Dockerfile

## 2. Docker Engine

### Imagen

Hay diferentes formas de crear imágenes:


- Descargar de Docker Registry 
- A partir de un contenedor existente
- A partir de un fichero Dockerfile

```
$ docker pull nginx:latest
```

## 2. Docker Engine

### Imagen

Hay diferentes formas de crear imágenes:

- Descargar de Docker Registry
- A partir de un contenedor existente 
- A partir de un fichero Dockerfile

```
~ docker run -it --name nginx_v1 nginx bash
```


```
~ root@75f87f84a091:/# echo "<h1>Hola</h1>"  
                        > /usr/share/nginx/html/index.html
```

```
~ root@75f87f84a091:/# exit
```

```
~ docker commit nginx_v1 jpaniorte/nginx:v1
```

## 2. Docker Engine Imagen

Hay diferentes formas de crear imágenes:

- Descargar de Docker Registry
- A partir de un contenedor existente
- A partir de un fichero Dockerfile 

~ docker build -t jpaniorte/nginx:v2 .

Especial atención al "."  
Indica el contexto del Dockerfile

#Dockerfile

```
FROM nginx:latest"
COPY index.html \
    /usr/share/nginx/html/index.html
```

Estructura del directorio

→ tree

```
.
├── Dockerfile
└── index.html
```

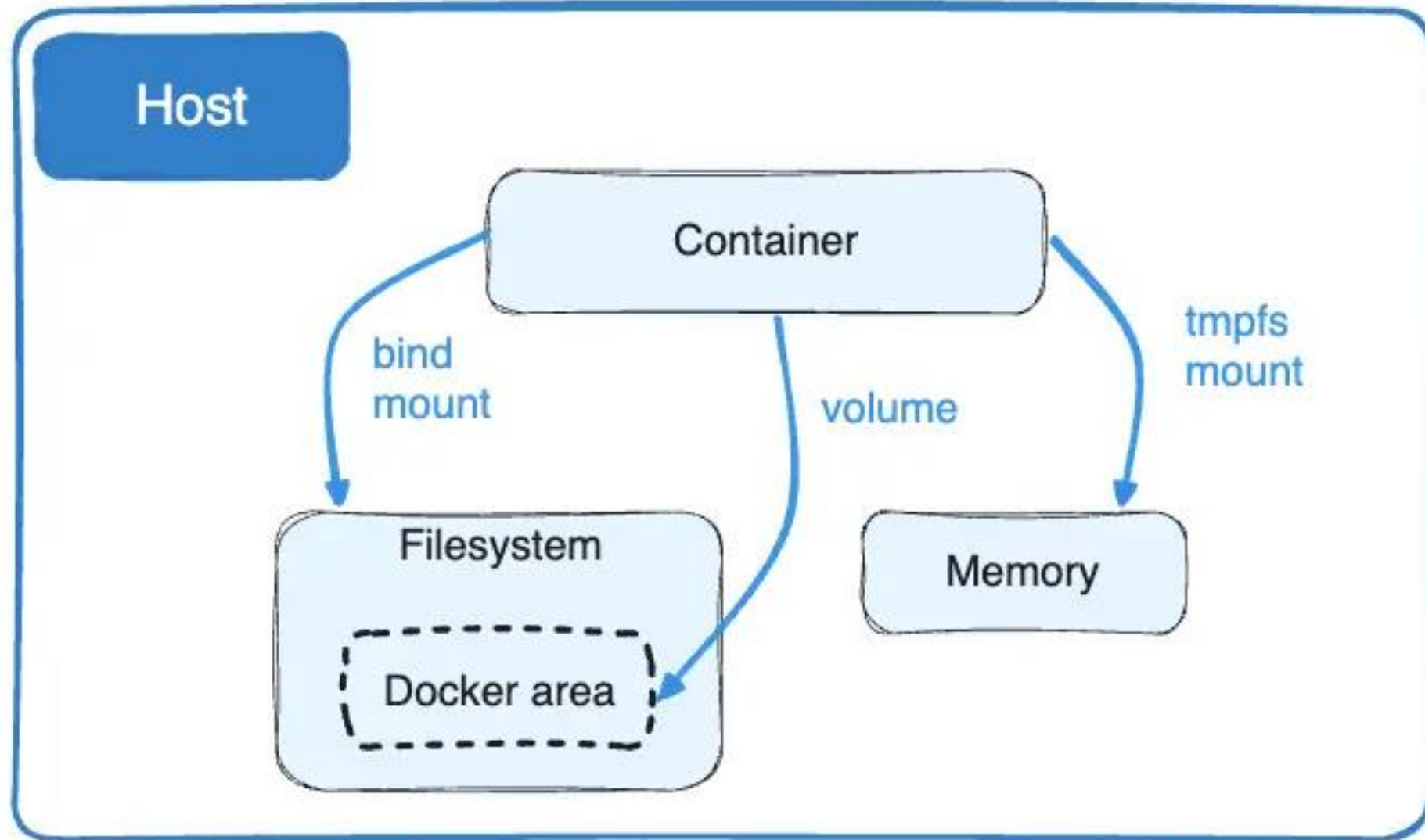
## 2. Docker Engine

### Volumen

Hay diferentes tipos de volúmenes en Docker:

- **Bind mount:** los ficheros son visibles en el Docker Host.
- **Volume:** Los ficheros no son visibles en el Docker Host
- **Tmpfs mount:** Los volúmenes se almacenan únicamente en la memoria del Docker Host y no se escriben nunca en el sistema de archivos.

## 2. Docker Engine Volumen





## 2. Docker Engine Networking

Hay diferentes tipos de redes en Docker:

- **bridge:** Los contenedores en esta red pueden comunicarse entre sí y tienen acceso a la red local de la máquina host.
- **host:** En esta red, los contenedores comparten la red del host, es decir, no tienen su propia dirección IP.
- **none:** Este tipo de red asigna a los contenedores una interfaz de red sin conexión: no tienen acceso a la red del host ni a otras redes.

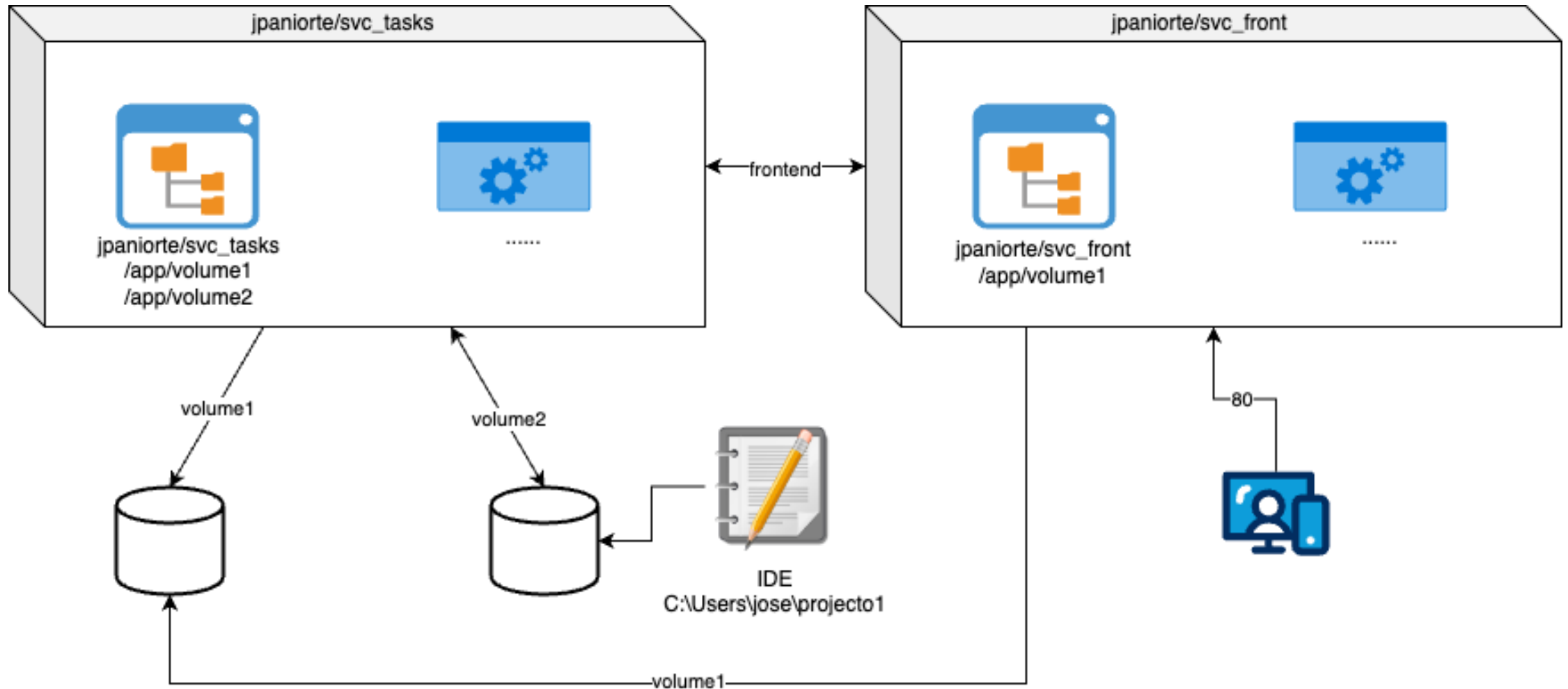
## 2. Docker Engine Networking

Hay diferentes tipos de redes en Docker:

- **overlay**: Utilizada para crear redes virtuales entre contenedores distribuidos en múltiples hosts Docker.
- **container**: Un contenedor puede compartir la red de otro contenedor.

## 2. Docker Engine

### Componentes: desarrollo local





# 3. Docker Compose

# 3. Docker Compose

## Casos de uso

- **Desarrollo:** Inicializa todos los contenedores con una instrucción: “docker compose up”
- Entornos para pruebas automatizada  
    docker compose up -d  
    ./run\_test  
    docker compose down
- Implementaciones en producción de host único: permite escalabilidad horizontal (múltiples instancias de un contenedor).

# 3. Docker Compose

## Ejemplo

services:

frontend:

image: example/webapp

ports: - "443:8043"

networks:

- front-tier

- back-tier

backend:

image: example/database

volumes:

- db-data:/etc/data

networks:

- back-tier

volumes:

db-data:

driver: flocker

driver\_opts:

size: "10GiB"

networks:

front-tier: {}

back-tier: {}