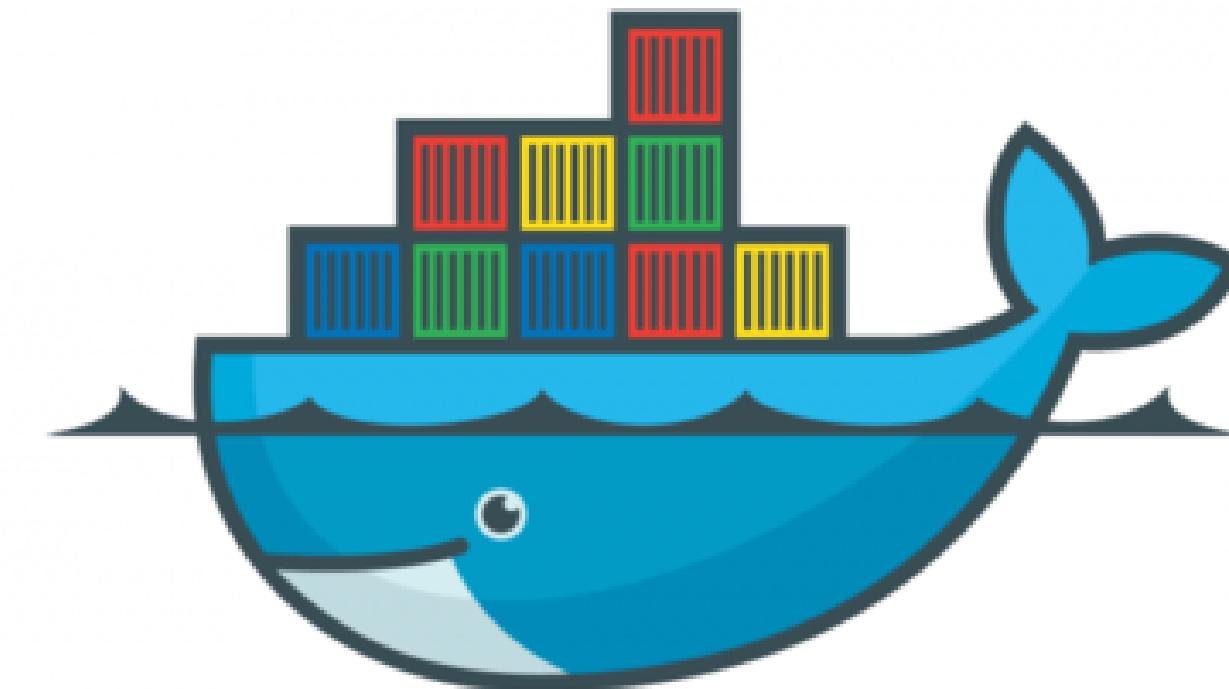


Empezando con Docker



1

Conceptos Docker

2

Configuración

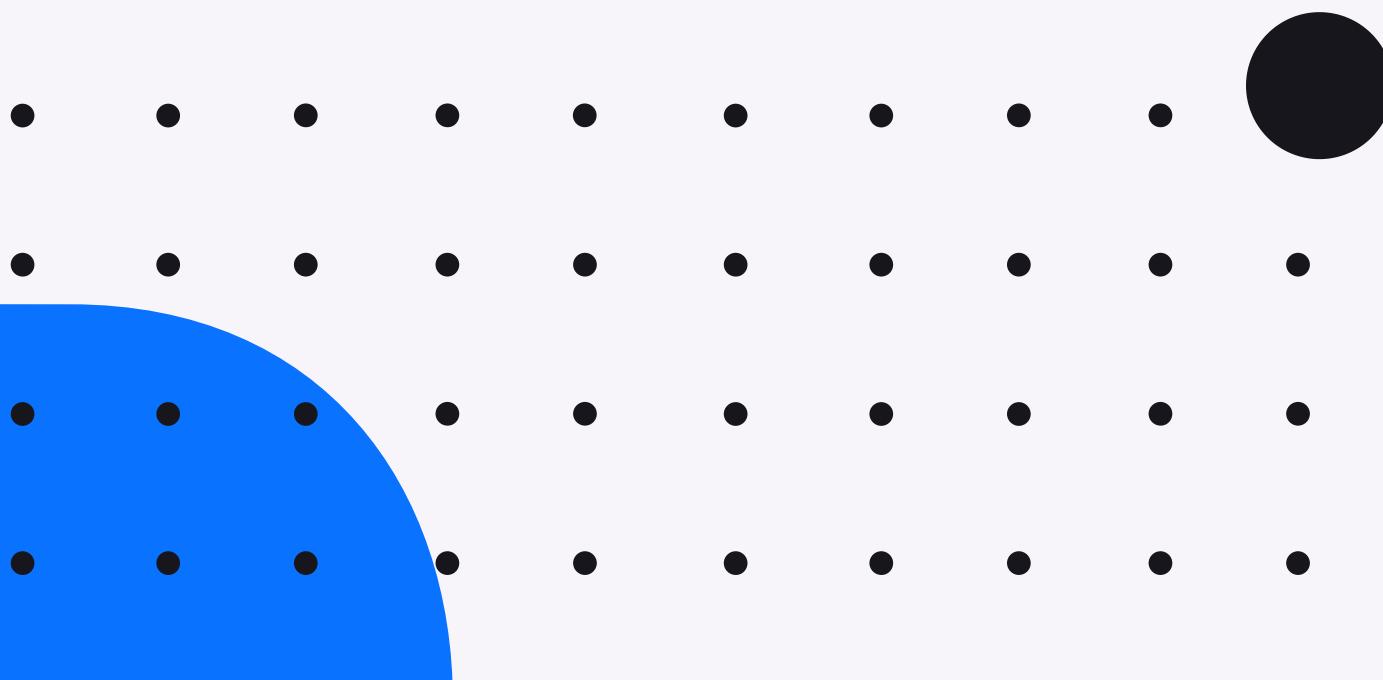
3

Comandos Docker

4

Ejemplos

Tabla de Contenido



1. Conceptos Docker

Docker es una plataforma para desarrolladores y administradores de sistemas para construir, compartir y ejecutar aplicaciones con contenedores

- Flexibles
- Ligeros
- Portables
- Desacoplados
- Escalables
- Seguros



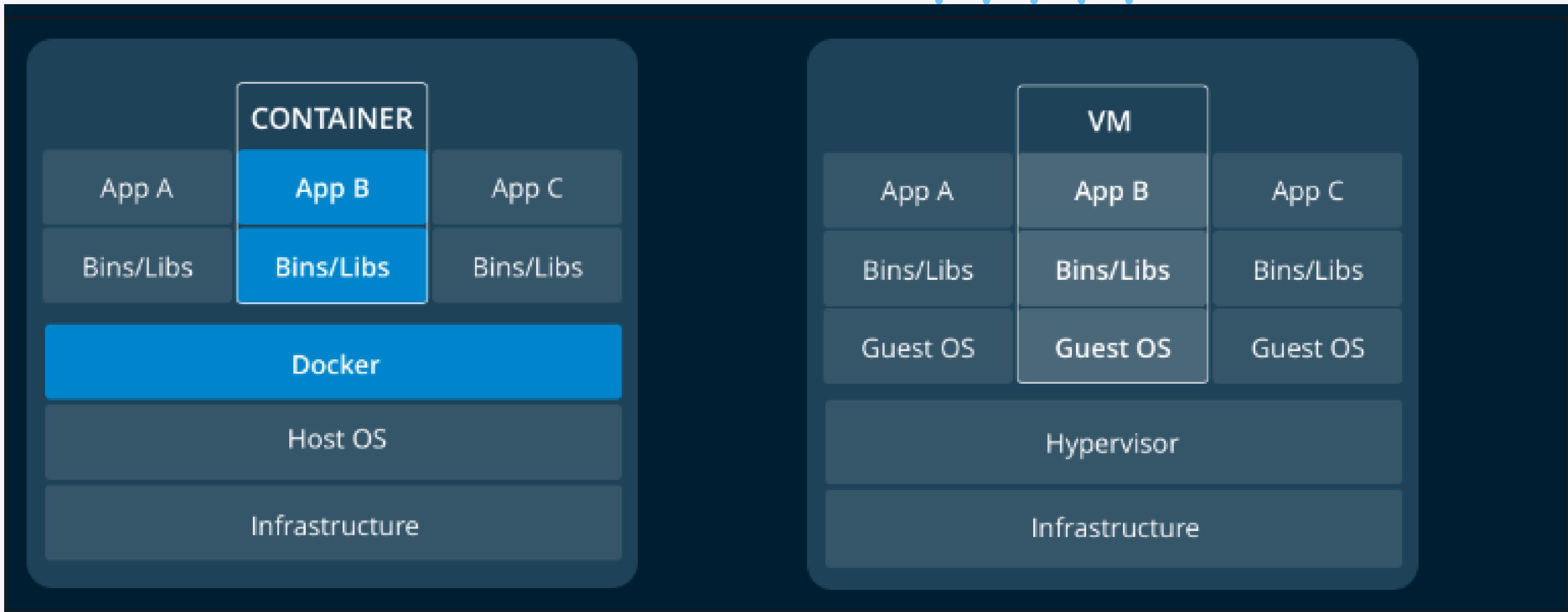
Imágenes

- Una imagen incluye todo lo necesario para ejecutar una aplicación: el código o binario, tiempos de ejecución, dependencias y cualquier otro objeto del sistema de archivos requerido.
- Las imágenes se construyen a través de capas (cada capa es un cambio en el sistema de archivos del contenedor)

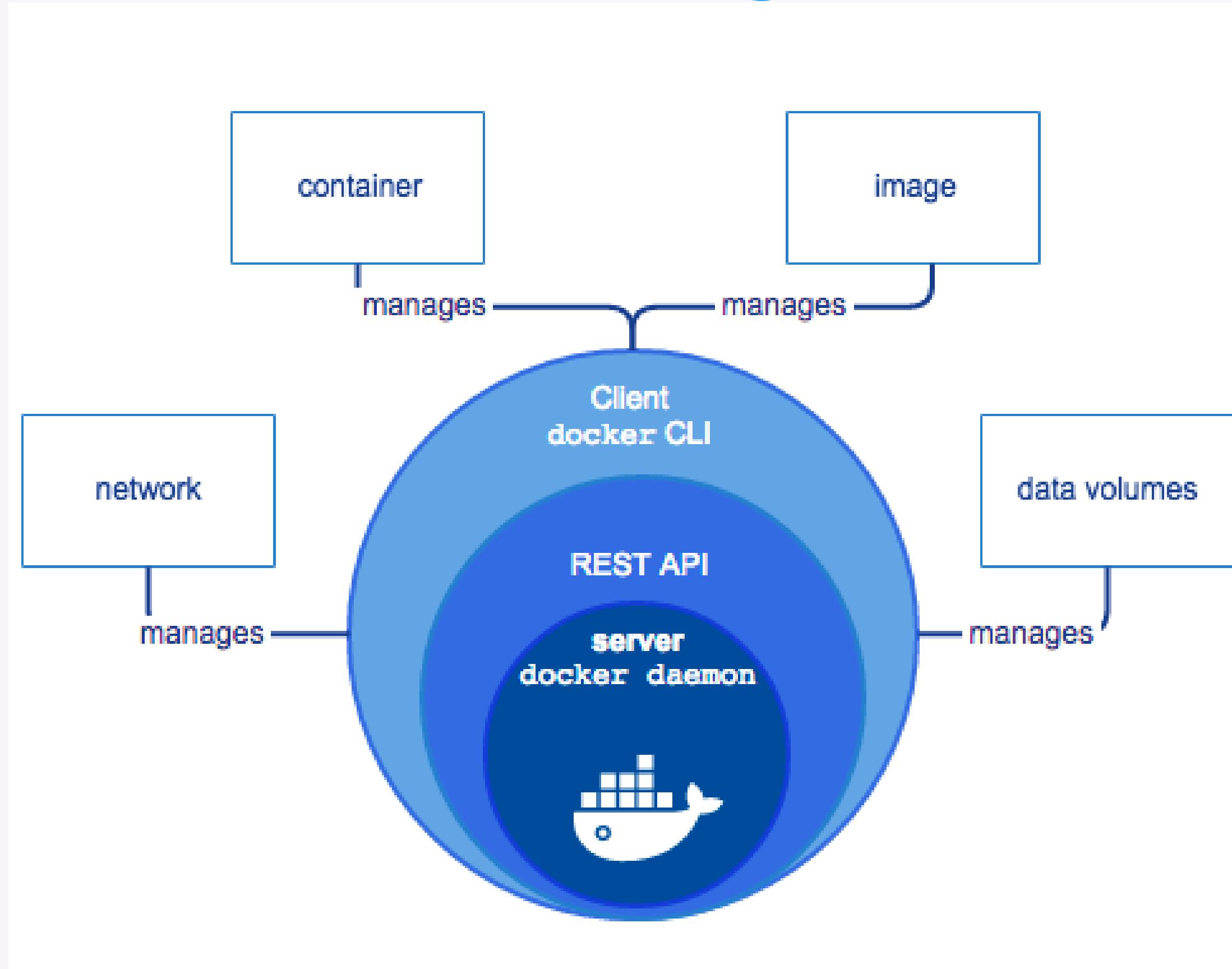
Contenedores

- no es más que un proceso en ejecución, con algunas características de encapsulación adicionales aplicadas para mantenerlo aislado del host y de otros contenedores.
- Cada contenedor interactúa con su propio sistema de archivos privado

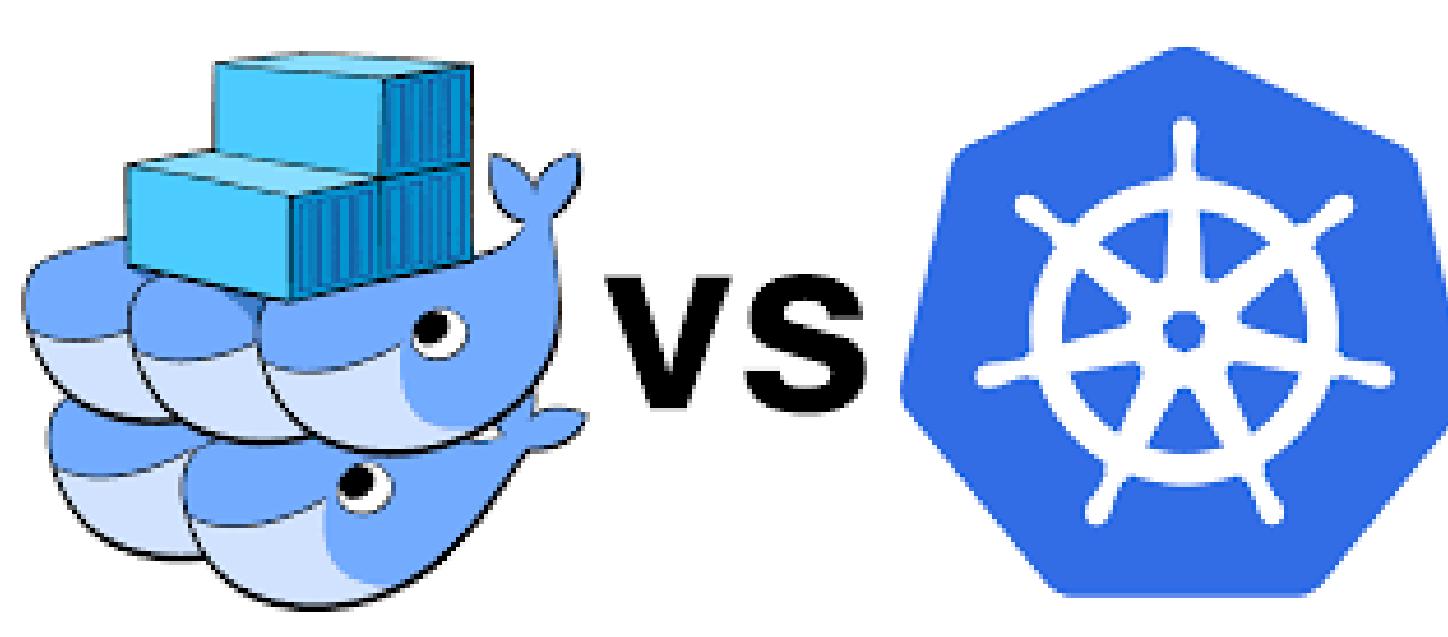
Funcionamiento



Docker Engine



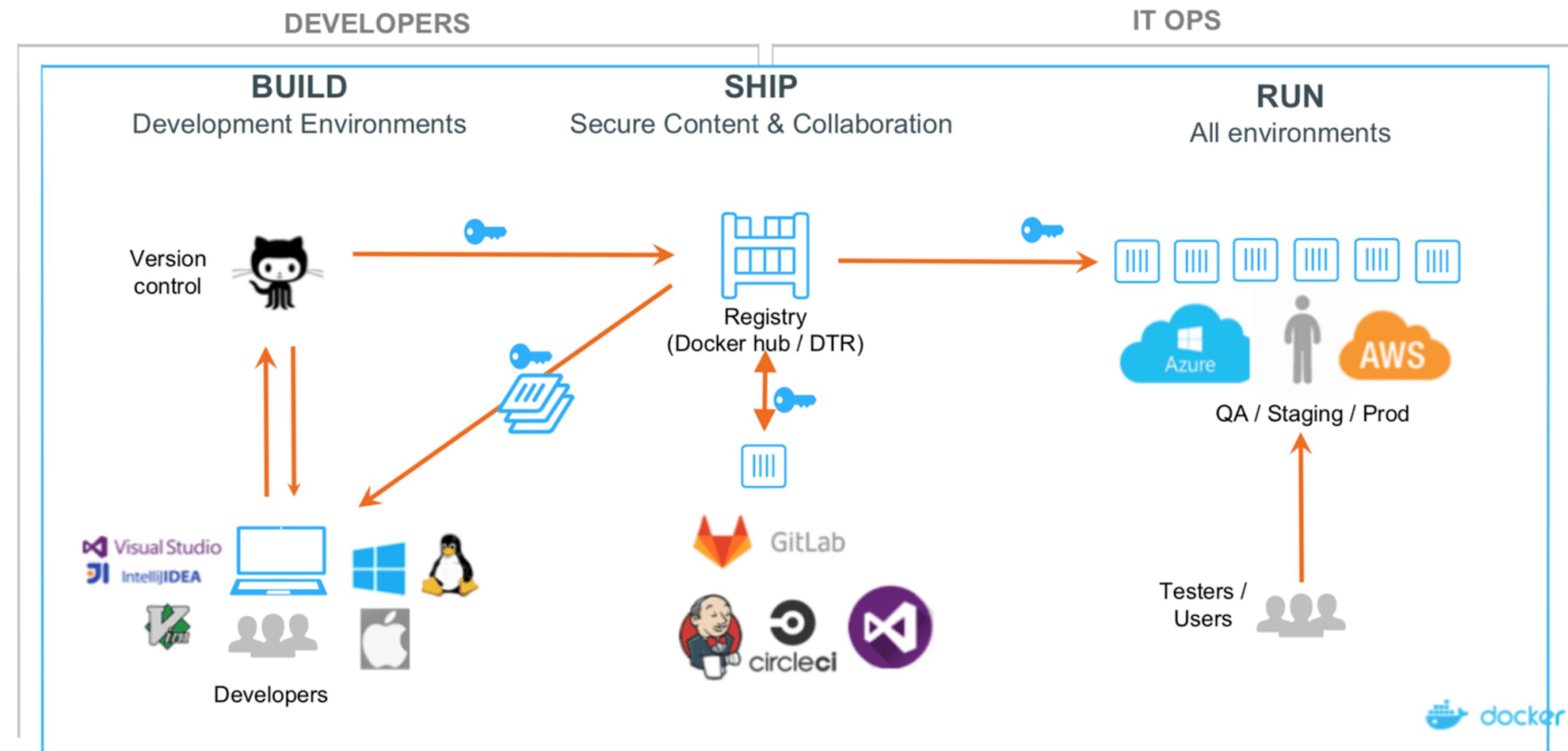
Orquestación



- La portabilidad y reproducibilidad de un proceso en contenedores significa que tenemos la oportunidad de mover y escalar nuestras aplicaciones en contenedores a través de nubes y centros de datos.
- Los contenedores garantizan efectivamente que esas aplicaciones se ejecutarán de la misma manera en cualquier lugar.
- Orquestadores.

Docker Development Workflow

Continuous Integration & Delivery Workflow



2. Configuración

- Descargar Docker Desktop.
- Crear Cuenta en Docker Hub

The screenshot shows the Docker Desktop website. At the top, there's a navigation bar with the Docker logo, links for "Why Docker?", "Products", "Developers", and "Company", a search bar labeled "Sign In", and a blue "Get Started" button. The main heading is "Docker Desktop and Desktop Enterprise" in large, bold, dark blue text. Below it, a sub-heading reads "The fastest way to securely build cloud-ready modern applications on your desktop." At the bottom, there are two buttons: "Download Desktop for Mac and Windows" and "Watch overview video".

- <https://www.docker.com/products/docker-desktop>
- (Opcional pero recomendado, instalar Visual Studio Code)

The screenshot shows the Docker Hub sign-in page. It features the Docker logo at the top right. The main title is "Welcome Back" in bold black text. Below it is a sub-instruction "Sign in with your Docker ID". There are two input fields: "Docker ID" and "Password", both with placeholder text. At the bottom is a large blue "Sign In" button. To its left are links for "Forgot Docker ID or Password?" and "Sign Up".

3. Comandos

- Docker info
- Docker images: Listar imágenes
- Docker ps: Listar contenedores
- Docker run <image> : Crear contenedor con base en una imagen

Ejemplo 1:

Docker run hello-world.

Ejemplo 2:

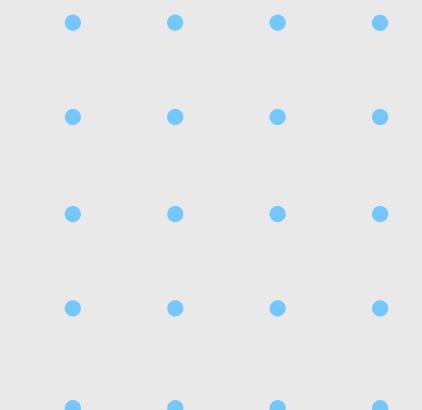
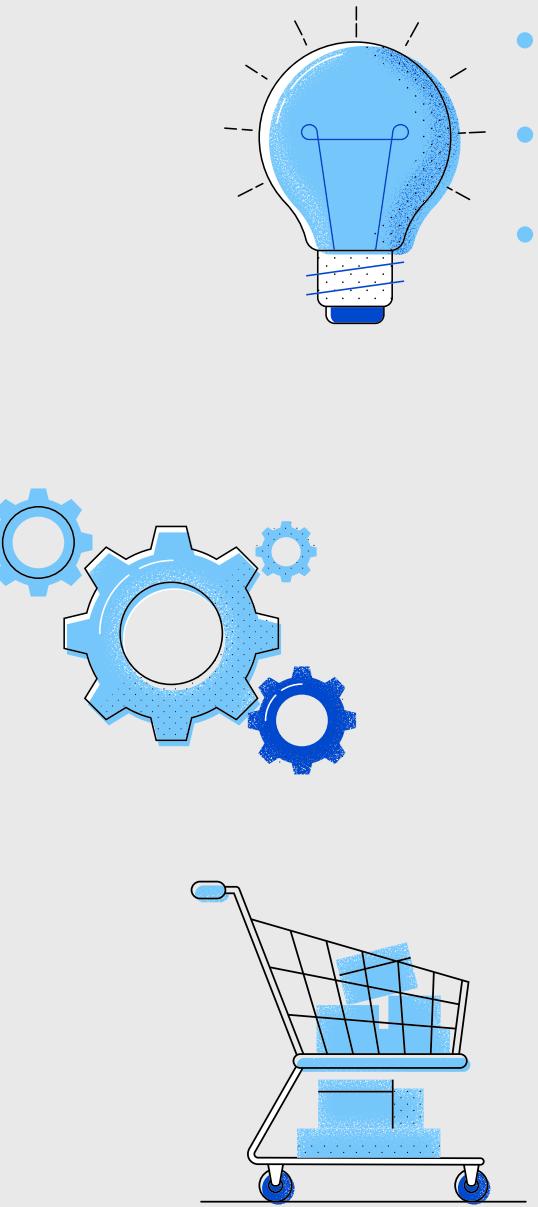
Docker pull Ubuntu

Docker run --name ubuntu -it ubuntu

Ejemplo 3:

Docker pull mysql

Docker pull node



Construcción de Imágenes

Manualmente.

- crear una carpeta <capacitacion-docker>
dentro, hacer pull al repo
- <https://github.com/Jccorzo/spring-convertir-numeros-arabicos.git>
- luego por consola: git checkout develop
- Docker run --name -p 8080:8080 ubuntu -it ubuntu
apt update && apt install openjdk-11-jdk -y
mkdir /home
- Docker cp build/libs/ArabigosARomanos-0.0.1-SNAPSHOT.jar ubuntu:/home/
- java -jar build/libs/ArabigosARomanos-0.0.1-SNAPSHOT.jar (dentro del contenedor)
(quedará el microservicio ejecutándose en el contenedor
pero disponible a través del navegador)
- docker commit ubuntu my_image (para crear nuestra propia imagen con la configuración anterior)

Mediante Dockerfile

FROM ubuntu

RUN apt-get update -y && apt-get install
openjdk-11-jdk -y

EXPOSE 8080

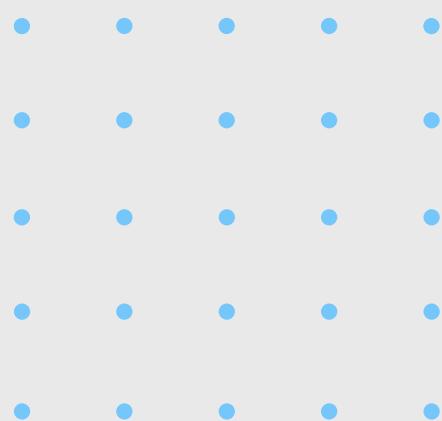
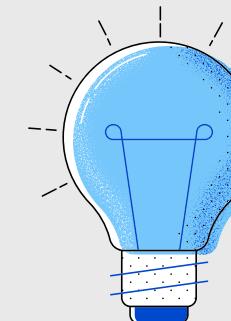
WORKDIR /home

RUN mkdir /home/spring-project

COPY build/libs/ArabigosARomanos-0.0.1-

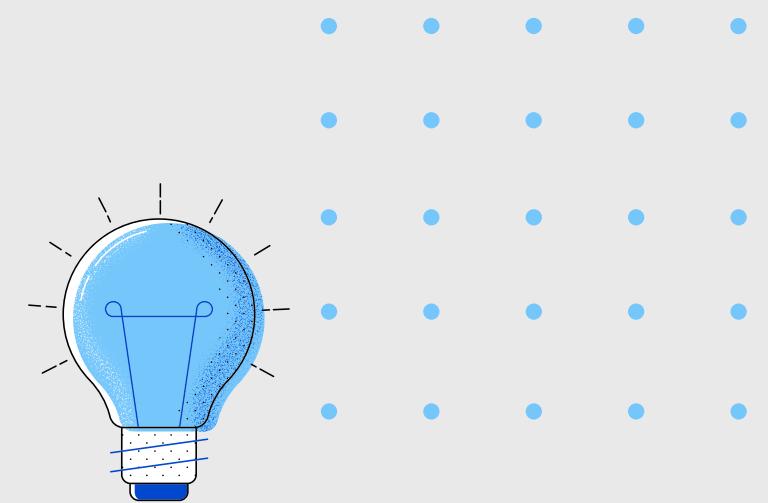
SNAPSHOT.jar /home/spring-project

CMD ["java", "-jar", "/home/spring-
project/ArabigosARomanos-0.0.1-
SNAPSHOT.jar"]



Redes Ejemplo

```
Docker run --name <nOMBRE1> -it ubuntu  
Docker run --name <nOMBRE2> -it ubuntu  
    apt update && apt install iputils-ping
```



```
Docker network create --driver bridge red2
```

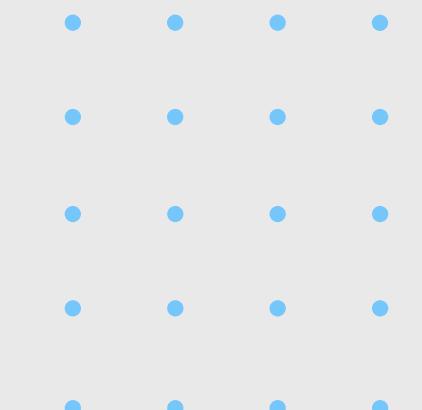
```
Docker network connect red2 <nOMBRE1>  
Docker network connect red2 <nOMBRE2>
```

```
Docker inspect <nOMBRE1> (obtener ip o  
hostname)
```

El propósito de los comandos ejecutados en el lado derecho es crear dos contenedores, que por defecto quedan aislados uno del otro. Pero al crear una nueva red y conectarlos a esta red, se puede evidenciar que ambos quedan dentro de la misma red.

En contenedor dos:

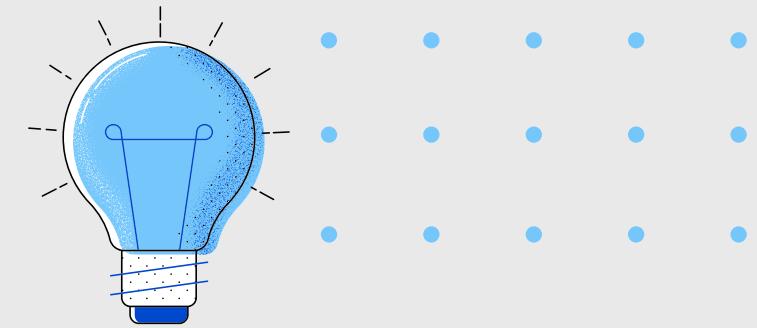
```
ping <ip o hostname del contenedor 1>
```



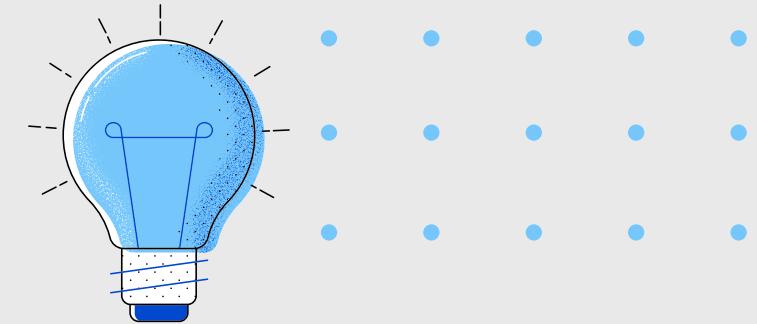
Volumes

- Los volumes es docker son el mecanismo para persistir información en las aplicaciones cuando se eliminan los contenedores y se crean otros nuevos.
- La información que genera, o que usa una app, se recomienda sea almacenada en volumenes y no directamente dentro de los contenedores.

Para el ejemplo, se ejecutará un contenedor con base en una imagen mongodb, se creará un registro y se guardará dentro de un volumen llamado "data" en la carpeta donde mongo guarda su información por defecto, se eliminará el contenedor, y se creará uno completamente nuevo, enlazandolo con el volumen creado anteriormente y verificando como en la db se encuentra la información creada en el volumen anterior.



Volumes Ejemplo



1. docker run --name db1 -it -p 27017:27017/tcp -v data:/data/db mongo:latest
2. docker exec -it db1 /bin/bash (esto nos deja dentro del contenedor creado)
3. Dentro del contenedor: ejecutamos "mongo" como comando y esto ejecuta la shell de mongo.
4. show dbs (para ver las bases de datos)
5. use dbprueba (para cambiar a esta nueva base de datos)
6. show collections (para ver que no hay collections creadas)
7. db.users.insert({ name: "usuario1" }) (para guardar un nuevo usuario)

Ahora abrimos una nueva consola para eliminar el contenedor con el comando "docker rm -f db1" y ejecutamos el primer comando nuevamente pero con otro nombre

-- docker run --name db2 -it -p 27017:27017/tcp -v data:/data/db mongo:latest

- repetimos el paso dos pero con el nombre del nuevo contenedor (db2), y repetimos hasta el paso 4.

De este modo veremos que la db (dbprueba) ya estará creada.

-repetimos el paso 5.

-Finalmente ejecutamos db.users.find() y veremos el registro que habiamos guardado anteriormente.

Probando así el uso de los volúmenes en bases de datos de una forma sencilla.

