

Docker - Introducción

Containers & Imágenes

¿Que es Docker?

Software de **virtualización** que **facilita el desarrollo y el despliegue de aplicaciones**.



¿Como funciona?

Ejecuta una aplicación dentro de un “**contenedor**” (container) junto con todas las dependencias, configuraciones y herramientas de sistema y de entorno necesarias.

Es decir, **empaqueta la aplicación junto con todo lo que necesita para funcionar.**

Estas imágenes o contenedores son **portables**; fáciles de distribuir y desplegar.



¿Porque facilita el trabajo?

Los contenedores ofrecen un **entorno aislado**, ya preconfigurado con todo lo necesario para trabajar con la aplicación a desarrollar.

Es decir, un contenedor es **independiente del sistema operativo y de los servicios de la máquina donde se está ejecutando**. En caso de que se requiera un nuevo servicio, se arranca como otro contenedor que trabaje en conjunto, o se añade al empaquetado.

Con esto, se logra **estandarizar el proceso de preparar los servicios** necesarios de forma local, **permitiendo centrarse en el desarrollo del programa**.



¿Porque facilita el despliegue?

Los contenedores ofrecen abstracción sobre las dependencias y configuraciones necesarias para el código a ejecutar en el servidor, dado que ya viene todo preparado: **no se requieren dependencias o configurar específicas en el servidor por cada aplicación.**

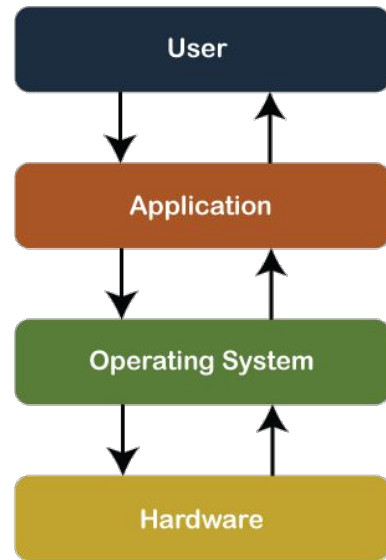


Docker vs VirtualBox (u otras máquinas virtuales)

Si simplificamos el funcionamiento de un Sistema Operativo, podemos separarlo en dos capas principales:

- Capa de **aplicación** - donde se ejecutan las aplicaciones (software)
- Kernel (**core**) - interactúa entre los componentes hardware y software

Por debajo del Kernel tendremos todo el **hardware** del equipo.




Docker vs VirtualBox (u otras máquinas virtuales)

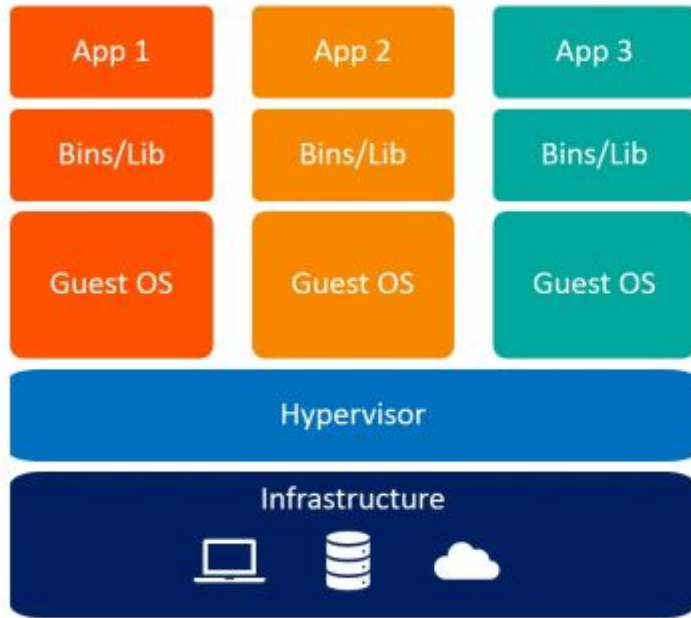
Las **máquinas virtuales** convencionales (como las que se montan con VirtualBox) **virtualiza el SO completamente; la capa de aplicación Y Kernel.**

- Para negociar con el SO Hosts, las máquinas virtuales incluyen una capa intermedia denominada “Hypervisor”.

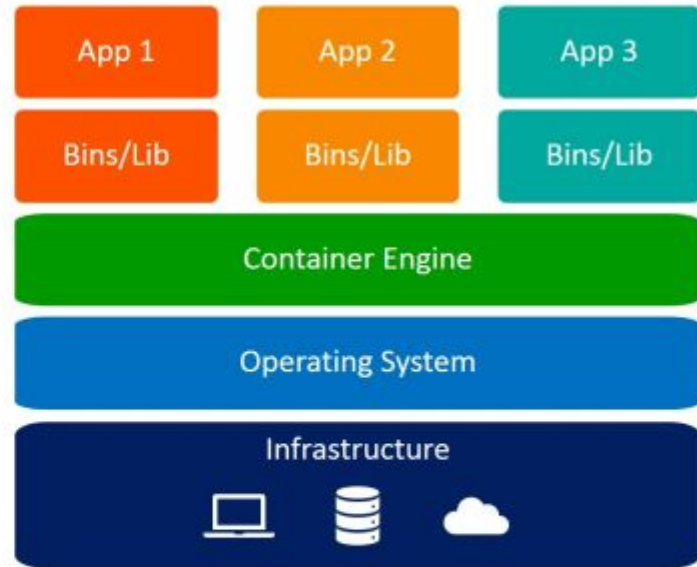
Docker virtualiza **únicamente la capa de aplicación**. En esta capa vendrá los servicios y las aplicaciones indicadas, todo ya instalado en el contenedor.

- Para que esta capa de aplicación virtualizada sea compatible con todos los SO, Docker también añade una capa de “Hypervisor”, que consiste en una distribución de linux ligera (parte del motor/daemon).
- 

Docker vs VirtualBox (u otras máquinas virtuales)




Virtual Machines



Containers

Docker vs VirtualBox (u otras máquinas virtuales)

Al comparar ambas virtualizaciones tendremos que:

- **El tamaño** de los contenedores o imágenes **de Docker es mucho menor**. Una imagen de Docker puede ocupar algunos MBs, mientras que las máquinas virtuales ocupan GBs.
 - **Arrancar contenedores de Docker es mucho más rápido** que arrancar máquinas virtuales.
 - Técnicamente, **Docker solo es compatible con distribuciones de linux**, mientras que las máquinas virtuales son compatibles con todos los SOs.
- 

Docker vs VirtualBox (u otras máquinas virtuales)


El problema de la compatibilidad se produce porque los contenedores de Docker se basan en linux. Originalmente, Docker fue desarrollado para este SO.

Con **Docker Desktop** se añade la capa de “Hypervisor”, que consiste en una distribución de linux ligera. Esto permite que se ejecuten contenedores en windows o MacOS sin problemas.



Docker Desktop

La instalación de Docker Desktop incluye:

- **Motor Docker** - hace posible la virtualización de contenedores
 - Se trata de un servidor con un demonio llamado "dockerd"
 - **Cliente CLI Docker** - Command Line Interface "docker"
 - Permite interactuar con el servidor de Docker vía comandos
 - **Cliente GUI** - Graphical User Interface
 - Permite interactuar con el servidor de Docker vía ventana/app (user friendly)
 - **Docker Build** - Para crear contenedores; herramienta de empaquetado
 - **Docker Compose** - Para configurar contenedores con múltiples aplicaciones
 - Docker Content Trust - Para verificar el origen de una imagen de Docker
 - **Kubernetes** - Automatizar despliegue de aplicaciones
 - Credential Helper - Suite para mantener credenciales de docker seguros.
 - Extensiones
- 

docker desktop

Q Search

📖 ⚙️ ⋮ 👤

Docker Desktop

Containers

Images

Volumes

Builds

Dev Environments

Docker Scout

Extensions

Extensions ...

+ Add extensions

Containers [Give feedback](#)

Container CPU usage

0.22% / 600%

(6 cores allocated)































Container memory usage

17.24MB / 7.56 CB

Show charts 📊

Q Search

☰ ☐ Only show running

<input type="checkbox"/>	Name	Image	Status	Port(s)	Last started	CPU %	Actions
<input type="checkbox"/>	<div> splashy-whale 43218bjf3411 </div>	free-willy:latest	Exited (255)	-	1 hour ago	0%	  
<input type="checkbox"/>	<div> barnacle-bob 70398a3bdf55 </div>	bikini-bottom:latest	Exited	-	2 hours ago	0.22%	  
<input type="checkbox"/>	<div> yarr-matey 87324y65ff32 </div>	blue-beard:latest	Exited (255)	-	4 hours ago	0%	  
<input type="checkbox"/>	<div> shrimp-clamtastic 34655h3ulf43 </div>	cocktail-sauce:latest	Exited	-	10 hours ago	0%	  
<input type="checkbox"/>	<div> oswald-west 57438h2gff89 </div>	cannon-beach:latest	Exited (255)	-	1 day ago	0%	  
<input type="checkbox"/>	<div> endless-summer 5823aa342f5i </div>	surfs-up:latest	Exited (255)	-	1 day ago	0%	  

Docker engine running

...

RAM 4.78 GB CPU 0.41% Disk 50.22 GB avail. of 58.37 GB

✓ Connected to Hub

v4.17.0

Images vs Containers

La imagen de Docker es el conjunto o paquete de la app al completo:

- Artefacto de aplicación ejecutado (como un .jar o .war)
- No solo tiene la aplicación a ejecutar; también incluye:
 - Opcionalmente; el código de la app (.java)
 - Entorno de desarrollo completamente configurado (Linux, node.js, npm...)
 - Variables de entorno, directorios necesarios, etc (estructura de carpetas)



Images vs Containers

El **contenedor** de Docker es **una instancia en ejecución de una imagen** de Docker.

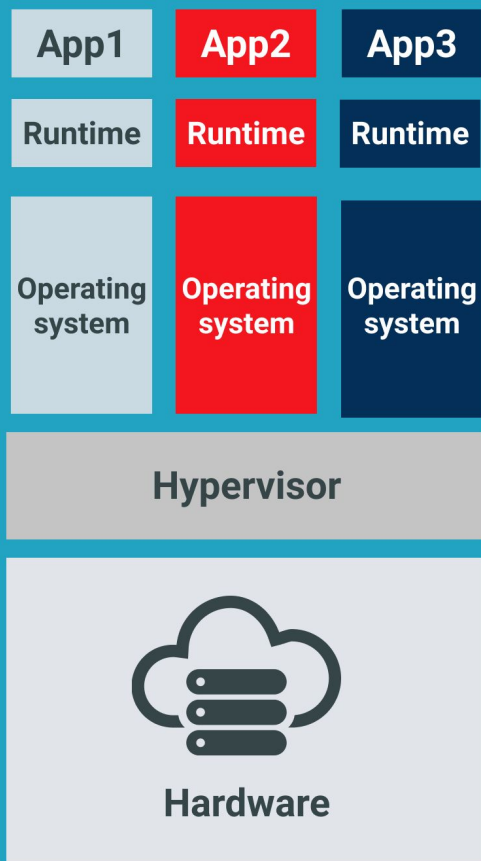
Es el programa desarrollado en funcionamiento. Cuando se inicia un contenedor es cuando se crea todo el entorno definido en la imagen.

En esencia; **un contenedor es una instancia de una imagen.**

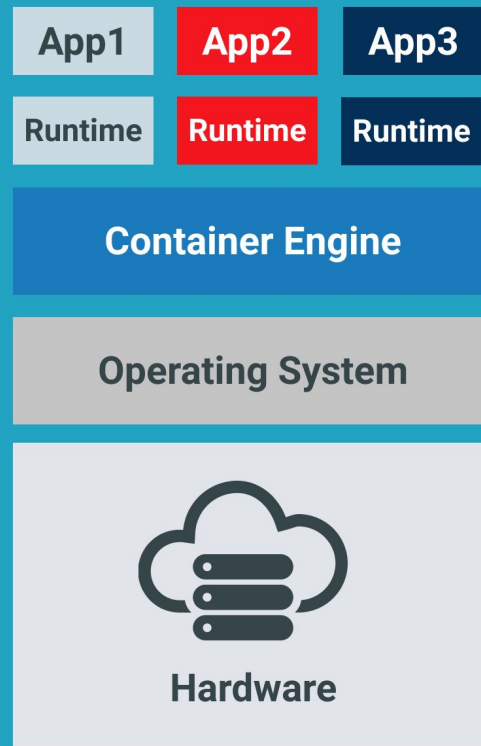
Es posible ejecutar múltiples instancias de la misma imagen (múltiples contenedores).



Machine Virtualization



Containers



App1 App2 App3

Runtime Runtime Runtime

Container Engine

Operating System



Hardware



App1 App2 App3

node GO .NET Core

 docker

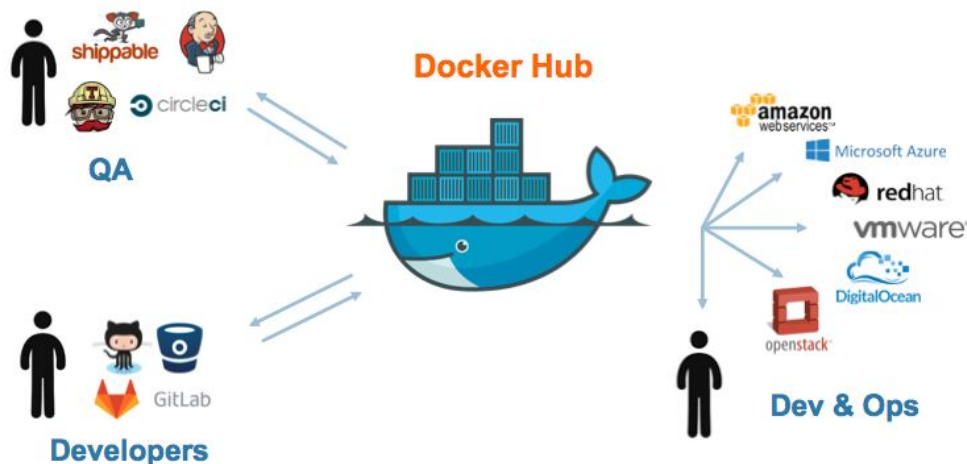
 Host OS


Hardware

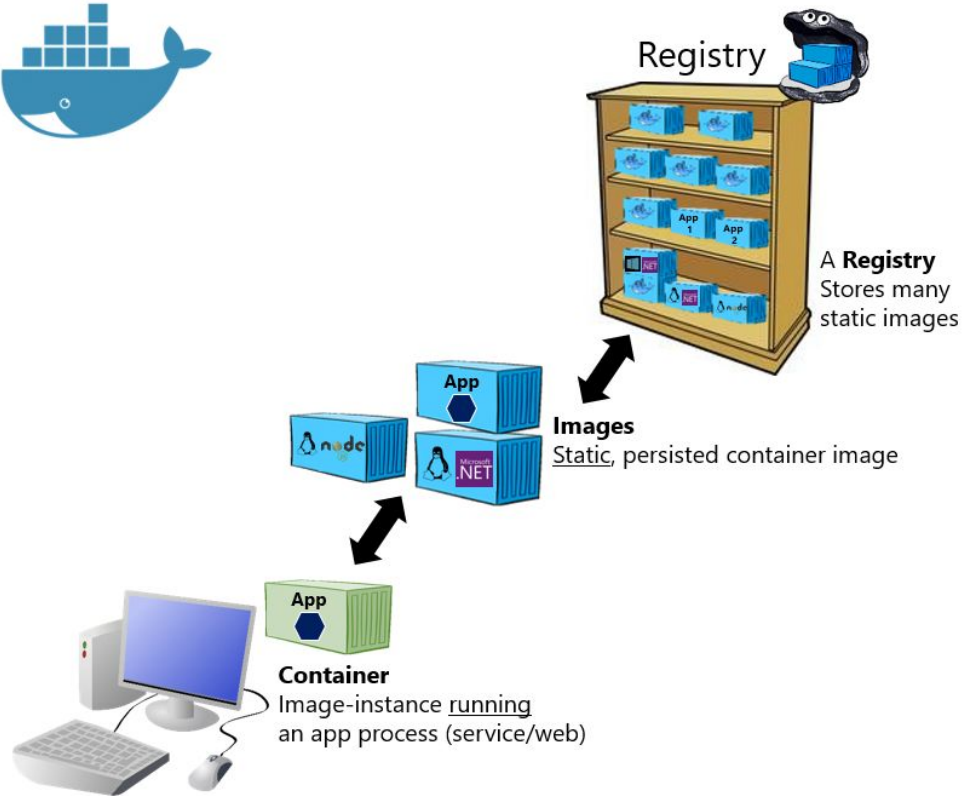
Registros de Docker (Docker Registries)

Un registro de Docker es un **almacén en cloud** para la distribución de imágenes de Docker.

¡Ver Docker Hub; el almacén oficial de Docker! Ofrece imágenes oficiales de aplicaciones.



Basic taxonomy in Docker



Hosted Docker Registry

Docker Trusted Registry on-prem.

On-premises

('n' private organizations)

Docker Hub Registry

Docker Trusted Registry on-cloud

Azure Container Registry

AWS Container Registry

Google Container Registry

Quay Registry

Other Cloud

Public Cloud

(specific vendors)

Registry vs Repository

- Docker Registry es un servicio que provee almacenamiento para imágenes.
- Puede ser hospedado por terceros o por uno mismo.
- Dentro de docker registry es posible tener múltiples repositorios.

En conclusión, dentro de un Registro de Docker, cada aplicación tendrá su propio repositorio, y dentro de dicho repositorio se almacenarán diferentes versiones de imágenes de esa aplicación.

