IES Miguel Romero Esteo

Curso 2025/26

Unidad 1 - Arquitectura Docker

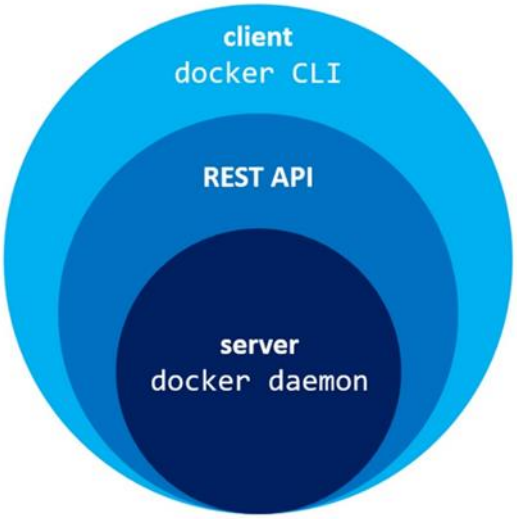
Unidad 1 - Arquitectura de Docker

ÍNDICE

1.- Arquitectura cliente-servidor
2.- API Rest de Docker
3.- Docker CLI
4.- Buscando imágenes con Docker CLI
5.- Buscar imágenes en Docker Hub (web)

1.- Arquitectura cliente-servidor

Docker se podría describir como una aplicación cliente-servidor. El demonio es el servidor y la CLI es uno de los muchos clientes (incluyendo clientes gráficos como Docker Desktop. También existen varios clientes de terceros. La siguiente gráfica muestra la arquitectura servidor cliente comentada

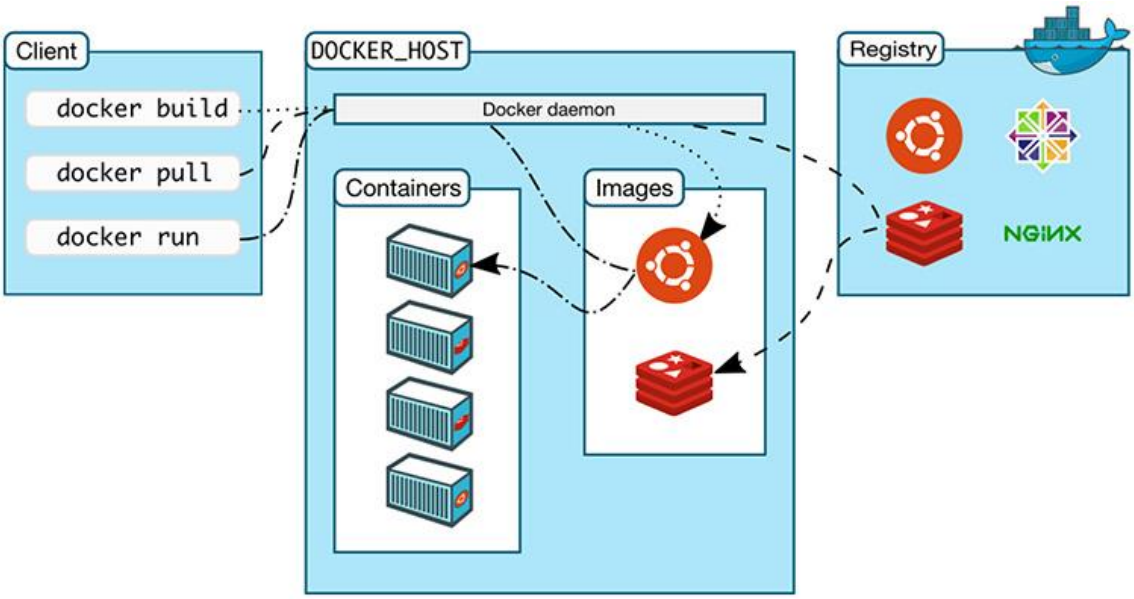


El demonio de Docker es un servicio que se ejecuta en el sistema operativo host . Actualmente solo funciona en Linux porque depende de varias características del kernel de Linux, pero también existen varias maneras de ejecutar Docker en macOS y Windows.

El propio demonio de Docker expone una API REST . Desde aquí, diversas herramientas pueden comunicarse con el demonio a través de esta API.

La herramienta más extendida es Docker CLI . Es una herramienta de línea de comandos que permite comunicarse con el demonio de Docker. Al instalar Docker, se obtienen tanto el demonio de Docker como las herramientas Docker CLI juntas.

.Docker CLI se conecta al Deamon de acuerdo con el siguiente esquema:



- **Primero, tenemos el cliente a la izquierda** , donde ejecutamos varios comandos de Docker. El cliente puede estar instalado en tu portátil con Windows, macOS o un servidor con Linux, no importa.

- **A continuación, tenemos el host de Docker** . Normalmente se le conoce como el servidor que ejecuta el demonio de Docker. Tiene sentido, ¿verdad? Es el host que ejecuta el demonio de Docker.

- **Por último, tenemos el registro,** que también forma parte del ecosistema Docker, pero por ahora puedes ignorarlo.

2.- API Rest de Docker

Una API REST (Representational State Transfer) es una interfaz que permite que dos aplicaciones se comuniquen entre sí a través de peticiones HTTP, de forma estructurada y estandarizada.

En lugar de comunicarse directamente, las aplicaciones usan la API como intermediaria.

Una API REST funciona mediante:

- URLs (endpoints)
- Métodos HTTP
- Mensajes en formato JSON

Los métodos más comunes son:

- GET → obtener información
- POST → crear recursos
- PUT → modificar recursos

- DELETE → eliminar recursos

Una API REST se caracteriza por:

- Ser sin estado (stateless): cada petición es independiente.
- Usar protocolos estándar (HTTP).
- Separar cliente y servidor.
- Ser escalable.
- Usar formatos de datos ligeros como JSON.
- Docker tiene una API REST que permite controlar Docker de forma programática, es decir, sin usar directamente la terminal.
- Cuando usamos Docker CLI o Docker Desktop, en realidad están enviando peticiones a la API REST de Docker.

Con la API REST de Docker se puede:

- Crear y eliminar contenedores
- Iniciar y detener contenedores
- Listar imágenes
- Subir y descargar imágenes
- Gestionar redes y volúmenes

Aquí tienes un ejemplo explicativo del uso de la API Rest

Cuando escribimos:

docker ps

Ocurre lo siguiente:

1. Docker CLI envía una petición GET a la API REST de Docker.
2. La API consulta el Docker Engine.
3. El motor responde con la lista de contenedores.
4. Docker CLI muestra el resultado al usuario.

3.- Docker CLI

La interfaz de línea de comandos de Docker (Docker CLI) es una herramienta robusta que permite interactuar con contenedores Docker y gestionar diferentes aspectos del ecosistema de contenedores directamente desde la línea de comandos. Con la CLI, se pueden gestionar eficientemente tareas como crear, iniciar, detener y eliminar contenedores, así como gestionar imágenes, redes y volúmenes de contenedores. La CLI de Docker ofrece una opción potente y flexible para gestionar contenedores y sus recursos.

Con Docker CLI podemos:

- Descargar imágenes desde Docker Hub.
- Crear y ejecutar contenedores.
- Ver contenedores activos o detenidos.

- Crear imágenes propias.
- Gestionar redes y volúmenes.
- Parar, reiniciar o eliminar contenedores.

Arquitectura Docker CLI

1. Recibe el comando del usuario.
2. Lo envía al Docker Engine (el motor de Docker).
3. El motor ejecuta la acción solicitada.

El usuario no trabaja directamente con el motor, sino a través del CLI.

Ejemplos de comandos básicos

- docker pull imagen → descarga una imagen.
- docker run imagen → crea y ejecuta un contenedor.
- docker ps → muestra los contenedores en ejecución.
- docker images → lista las imágenes disponibles.
- docker stop contenedor → detiene un contenedor.

Ventajas de Docker CLI

- Es rápido y potente.
- Permite automatizar tareas con scripts.
- Es igual en Windows, Linux y macOS.
- Es el método más usado en entornos profesionales.

4.- Buscando imágenes con Docker CLI

En docker tenemos dos formas de buscar imágenes del repositorio de docker hub. La primera es desde docker CLI con el comando

docker search alpine

Esto te mostrará:

- NAME → nombre de la imagen
- DESCRIPTION → descripción
- STARS → popularidad
- OFFICIAL → si es imagen oficial
- AUTOMATED → si se construye automáticamente

☞ Recomendación: prioriza imágenes con OFFICIAL = [OK].

El resultado sería

NAME	DESCRIPTION	STARS	OFFICIAL
alpine	A minimal Docker image based on Linux…	11437	[OK]
alpine/git	A simple git container running in alpine li…	249	

alpine/socat Run socat command in alpine container 115
alpine/helm Auto-trigger docker build for kubernetes hel… 69

Filtros

Podemos simplificar el proceso de búsqueda si utilizamos filtros. La sintaxis es

--filter clave=valor

O, de forma abreviada

-f clave=valor

Quedando la búsqueda como:

docker search --filter “Mi-Filtro” nombre_imagen

a) **is-official**

Muestra solo imágenes oficiales de Docker.

docker search --filter "is-official=true" nginx

Valores posibles:
- true
- false

☞ Recomendado siempre para entornos educativos o producción.

b) **stars**

Filtra por número mínimo de estrellas.

docker search --filter "stars=100" python

Esto mostrará imágenes con 100 estrellas o más.

c**) is-automated**

Muestra imágenes que se construyen automáticamente desde un repositorio.

docker search --filter "is-automated=true" ubuntu

Valores posibles:

true
false

Puedes usar varios filtros a la vez:

docker search --filter "is-official=true" --filter "stars=50" node

Algunos ejemplos con filtros son

docker search --filter "is-official=true" nginx

Busca sólo imágenes oficiales.

docker search --filter "stars=100" Python

Popularidad (mínimo número de estrellas)

Una vez encontrada podemos descargar la imagen con

docker pull nombre_imagen

Ejemplo:

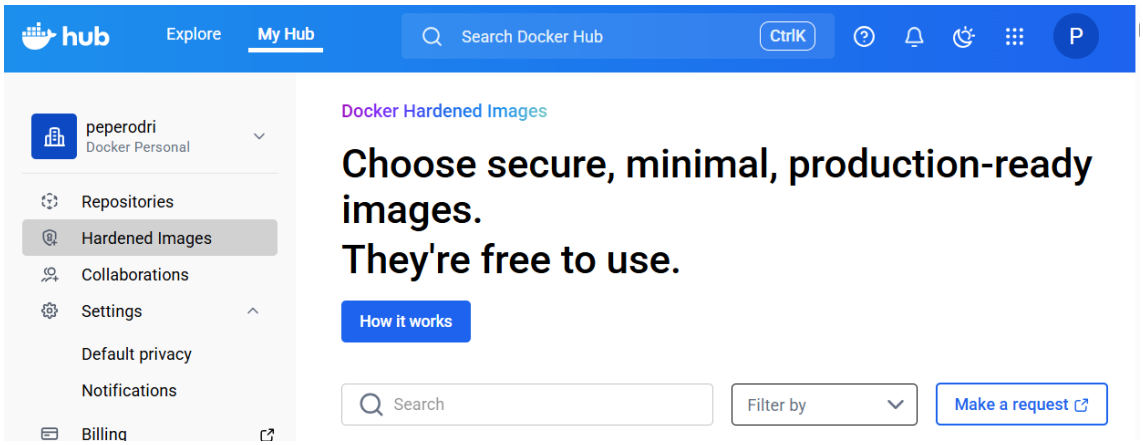
docker pull python:3.12

Si no indicas versión, Docker descarga latest (no siempre recomendable en producción).

5.- Buscar imágenes en Docker Hub (web)

Ve a ☞ https://hub.docker.com

1. Escribe el nombre de la imagen (ej. mysql, node, python)
2. Comprueba:
    - Que sea Official Image
    - Etiquetas disponibles (tags)
    - Fecha de actualización



Además de escribir el nombre se nos permite definir filtros. En primer lugar no pregunta por el tipo de ajuste.

🛡 FIPS (Federal Information Processing Standards)

FIPS es un conjunto de estándares oficiales del gobierno de EE. UU. para seguridad criptográfica. Define cómo debe hacerse la criptografía, no cómo configurar un sistema entero.

Principalmente regula una serie de parámetros tales como

- Algoritmos de cifrado (AES, RSA, SHA…)
- Gestión de claves
- Módulos criptográficos (software / hardware)

Ejemplo muy común:

- FIPS 140-2 / 140-3 → certificación de módulos criptográficos

Se usa principalmente en

- Sistemas que manejan información sensible
- Entornos gubernamentales o regulados
- Software que necesita cifrado certificado

Ejemplo práctico

Un sistema “FIPS-compliant”:

- Usa OpenSSL compilado en modo FIPS
- Solo permite algoritmos aprobados
- Bloquea MD5, SHA-1, etc.

La otra opción es

🛍️ STIG (Security Technical Implementation Guide)

Un STIG es una guía de endurecimiento (hardening) publicada por el DoD (Departamento de Defensa de EE. UU.). Define cómo configurar un sistema para que sea seguro.

Regula principalmente:

- Usuarios y contraseñas
- Servicios permitidos
- Permisos de archivos
- Logging y auditoría
- Red, firewall, SSH, sudo, etc.

Y se usa para

- Sistemas operativos
- Servidores
- Bases de datos
- Contenedores
- Kubernetes

Ejemplo:

- Red Hat Enterprise Linux STIG
- Docker STIG
- Kubernetes STIG

La diferencia entre ambas opciones es

Aspecto	FIPS	STIG
Tipo	Estándar	Guía técnica
Enfoque	Criptografía	Configuración del sistema
Quién lo define	NIST	DoD
Qué asegura	Cifrado correcto	Sistema endurecido
Afecta a	Algoritmos y librerías	Usuarios, servicios, red
Se puede auditar	Sí	Sí
Obliga a cifrado	✓	✗
Obliga a hardening	✗	✓

Ambos métodos no son excluyentes, de hecho:

- Un sistema STIG-compliant suele exigir FIPS
- Pero un sistema FIPS-compliant NO garantiza que cumpla STIG

Ejemplo real:

“Servidor RHEL configurado según STIG y con OpenSSL en modo FIPS”

Resumen en una frase

FIPS se ocupa del *cómo se cifra*
STIG se ocupa del *cómo se configura*

HELM CHART

Un Helm Chart es un paquete de Kubernetes que define cómo desplegar una aplicación en un clúster Kubernetes.

Incluye:

- Deployments
- Services
- ConfigMaps
- Volumes
- Variables de entorno
- Versiones y dependencias

Docker Hub ya no aloja solo imágenes Docker, también muestra:

- 🐋 Docker Images
- ⚓ Helm Charts

Por eso en los filtros puedes elegir:

- Images
- Helm Charts

❗ Diferencia clave

Docker Image	Helm Chart
Contiene una app + SO mínimo	Define cómo desplegar la app
Se usa con `docker run`	Se usa con `helm install`
Vale para Docker / Podman	Requiere Kubernetes
Ideal para prácticas básicas	Ideal para orquestación

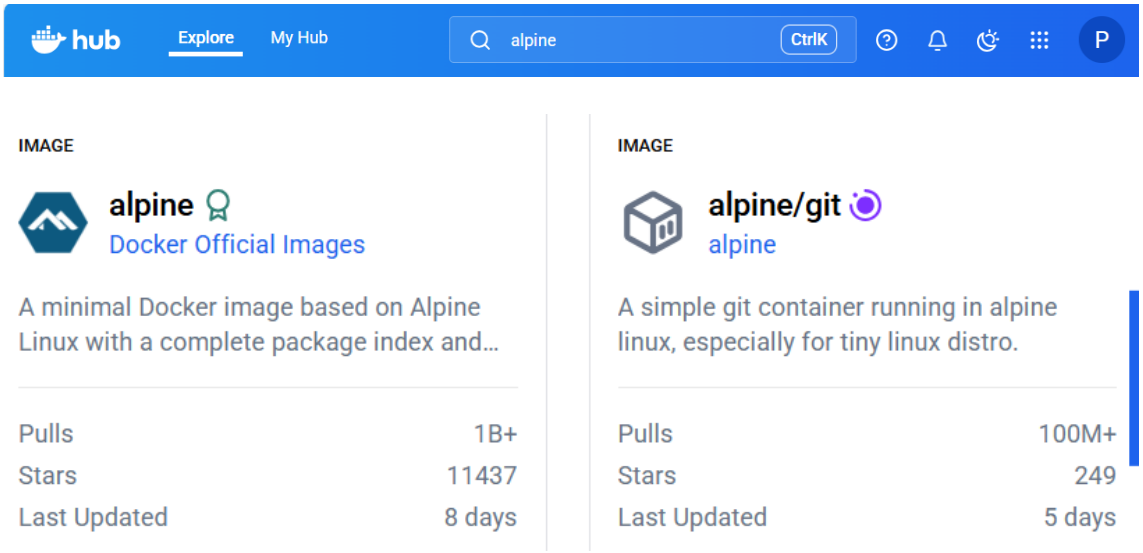
Ejemplo práctico

Imagen Docker
docker pull nginx
docker run -p 8080:80 nginx
Helm Chart
helm install mi-nginx bitnami/nginx

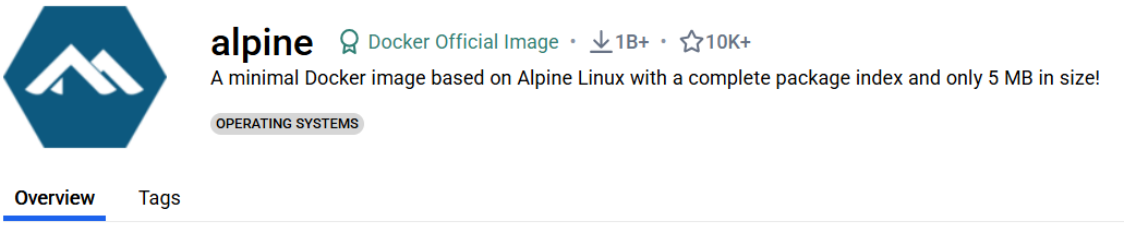
Aquí:

- el chart usa una o varias imágenes Docker
- pero añade toda la configuración de Kubernetes

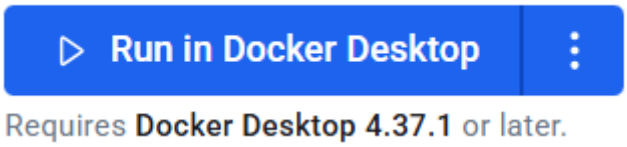
Una vez definidos los filtros obtenemos los resultados de la búsqueda.



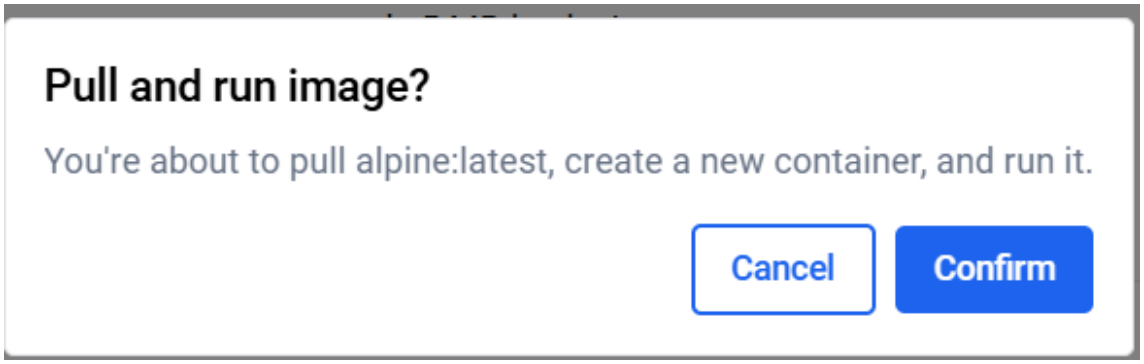
Podemos obtener información de cualquiera de las imágenes mostradas simplemente pulsando sobre ella

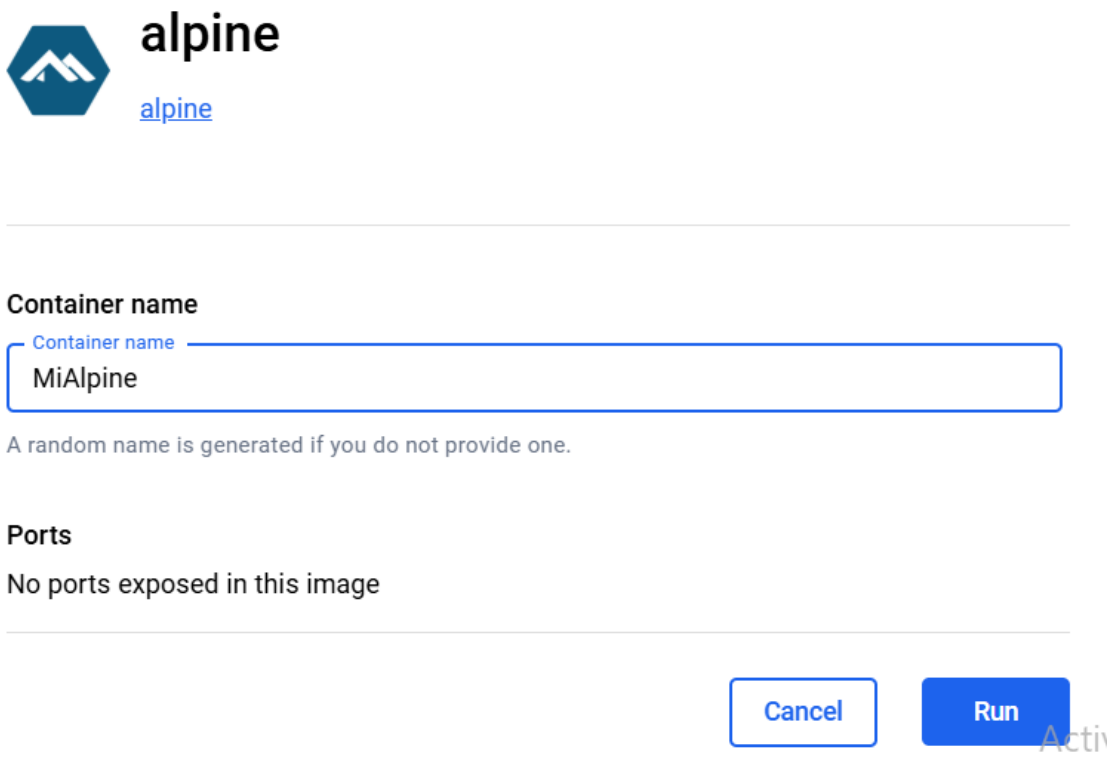


Entre esta información podemos encontrar la opción de cargarla en nuestro Docker Desktop

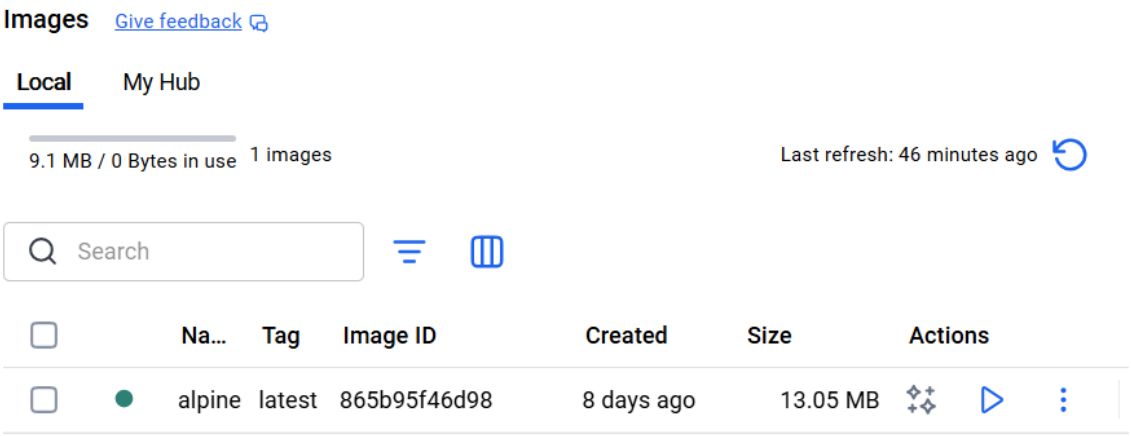


En Docker Desktop tendrás la siguiente información:





Y ya tendremos la imagen en nuestro repositorio local



Y lanzada en un contenedor

Containers

Give feedback

Container CPU usage

No containers are running.

Container memory usage

No containers are running.

Show charts

Search

Only show running containers

		Name	Container ID	Image	Pc	Actions
☐	○	MiAlpine	855164001d84	alpine:lates		