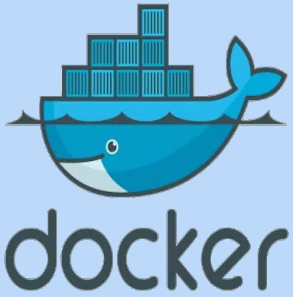


Taller de contenedores Docker

Mario Humberto Tiburcio Zúñiga

Instituto Tecnológico de Zacatepec 2025

Objetivo

Introducir a los participantes en los conceptos fundamentales de Docker, permitiéndoles crear y gestionar contenedores para el desarrollo y despliegue de aplicaciones.

1. Introducción

Conceptos de Docker

2. Instalación de Docker Engine

Puesta en marcha del ambiente de trabajo.

3. Creación de imágenes y Docker hub

Registrar una cuenta en Docker Hub y utilizar comandos para creación y/o obtención de imágenes.

4. Creación y manipulación de contenedores.

Conocer comandos Docker para crear y manipular contenedores.

5. Ejercicios prácticos.

Realizar ejercicios creando contenedores a partir de imágenes, realizar el despliegue y prueba de ejecución de las aplicaciones contenidas.

Recursos didácticos

- Computadoras del laboratorio conectadas en red.
- Computadora personal con Virtual Box
- Imagen de Ubuntu con Docker instalado.
- Proyector

Evaluación

1. Asistencia 30%

Hoja de asistencia

2. Práctica 50%

Lista de cotejo de actividades realizadas

3. Teoría 20%

Examen en línea

Registrarse al curso:

http://tiburcio.mx

http://tiburcio.cdmon.org/loginmaterias.php

¿Qué es Docker?

- Plataforma para ejecutar aplicaciones dentro de contenedores.
- Permite portabilidad y consistencia del entorno en desarrollo, pruebas y producción.
- Más liviano que una máquina virtual y evita errores típicos de diferencias entre entornos.

¿Qué es un Contenedor?

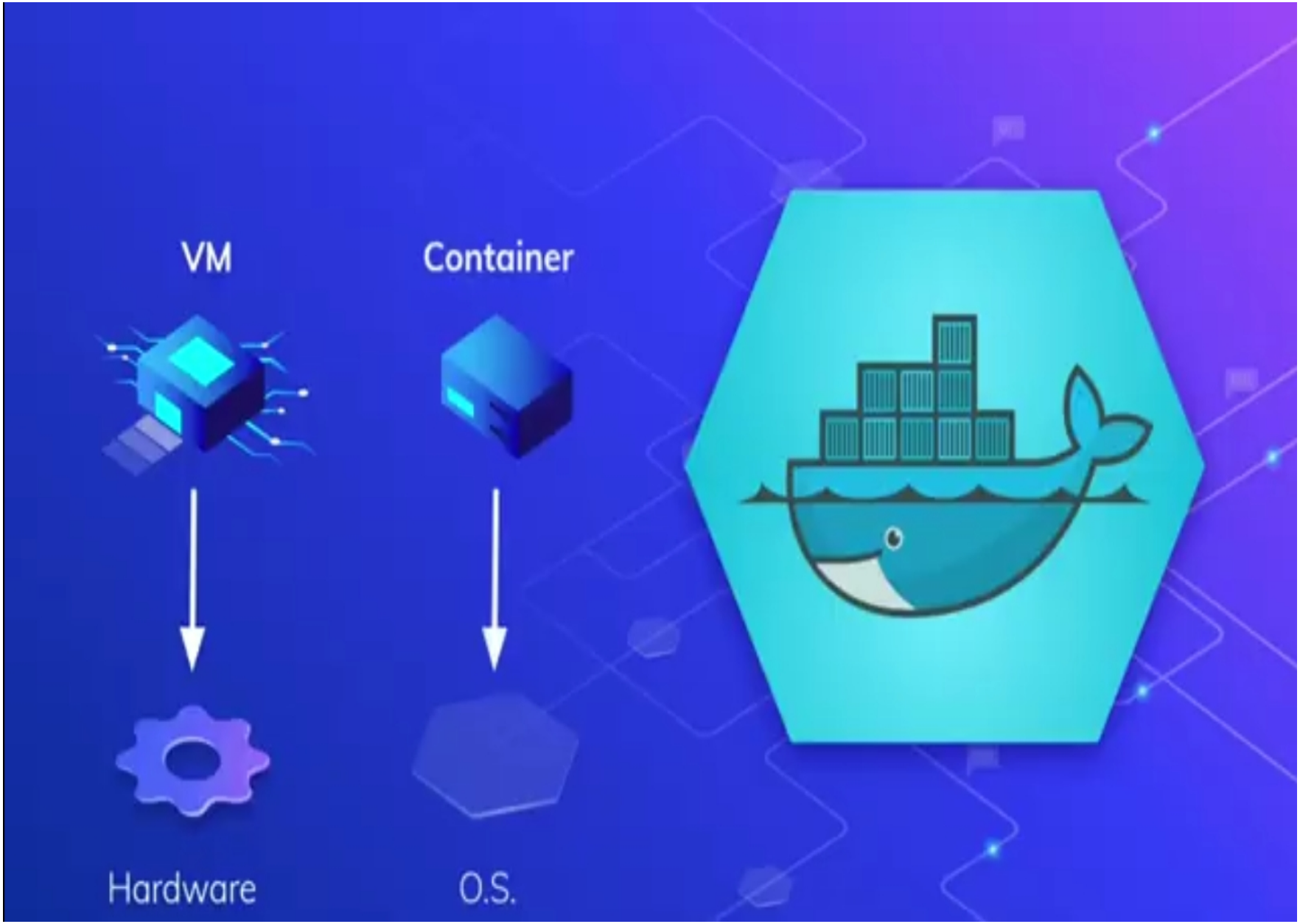
- Unidad aislada que incluye la aplicación, sus dependencias y configuraciones.
- Se ejecuta de forma consistente sin importar el sistema operativo del host.
- Equivale a una caja que lleva todo lo necesario para que la app funcione.

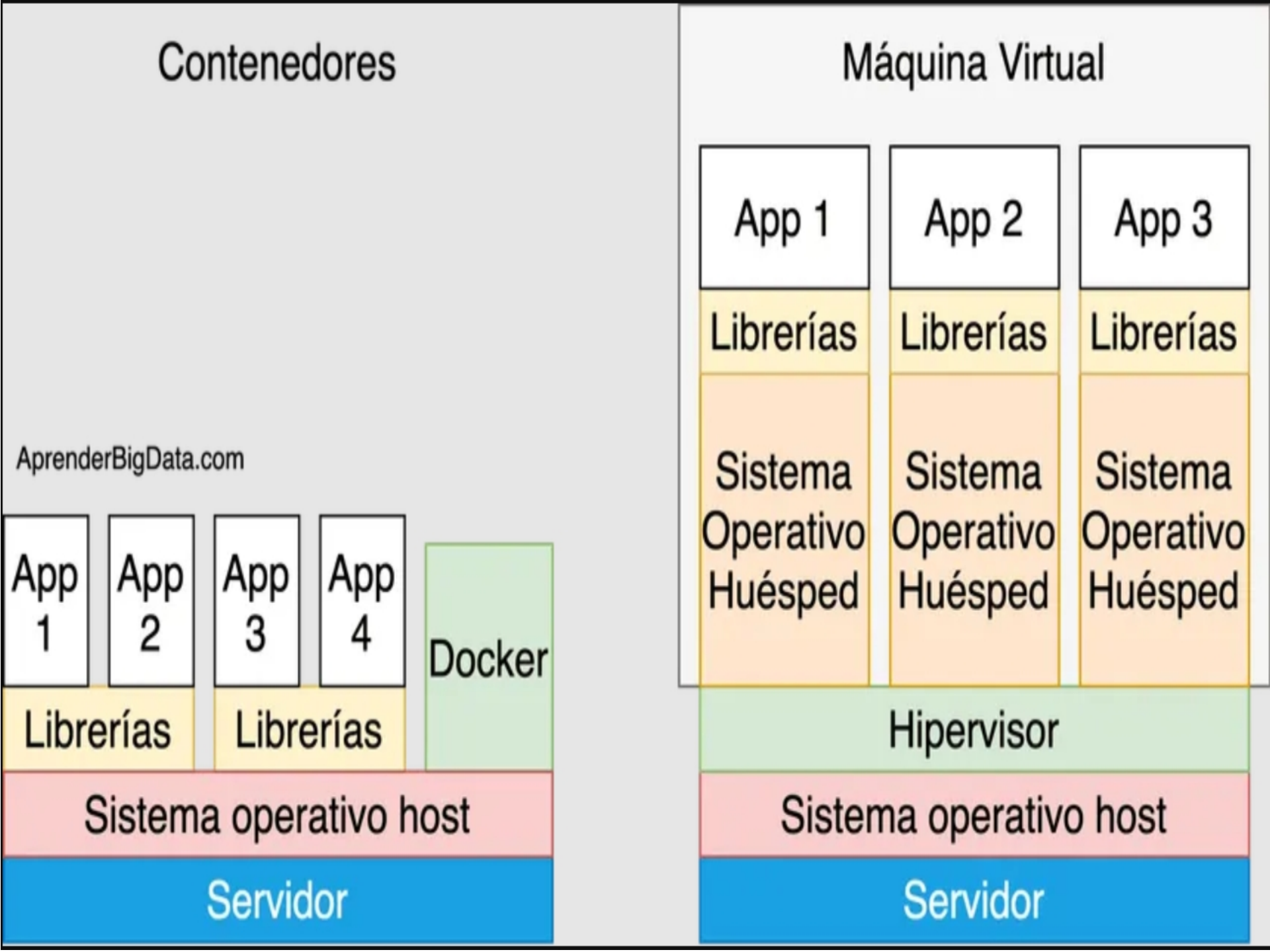
Contenedores vs Máquinas Virtuales

- Contenedor: Ligero, comparte SO del host, ejecución rápida.
- VM: Pesada, tiene su propio SO, mayor consumo de recursos.


VM
App A
App B
App C
Bins/Libs
Bins/Libs
Bins/Libs
Guest OS
Guest OS
Guest OS
Hypervisor
Infrastructure


CONTAINER
App A
App B
App C
Bins/Libs
Bins/Libs
Bins/Libs
Docker
Host OS
Infrastructure


VM
Container
Hardware
O.S.


Contenedores
Máquina Virtual
App 1
App 2
App 3
Librerías
Librerías
Librerías
Sistema Operativo Huésped
Sistema Operativo Huésped
Sistema Operativo Huésped
AprenderBigData.com
App 1
App 2
App 3
App 4
Docker
Librerías
Librerías
Hipervisor
Sistema operativo host
Sistema operativo host
Servidor
Servidor

¿Por Qué Docker?

- Evita el clásico “en mi máquina sí funciona”.
- Despliegue rápido y reproducible.
- Uso eficiente de recursos comparado con máquinas virtuales.
- Facilita arquitecturas de microservicios.
- Ideal para automatización CI/CD.

La automatización CI/CD se refiere a la práctica de automatizar el proceso completo de integración continua (CI) y entrega/despliegue continuo (CD) de software. Es un pilar fundamental del desarrollo ágil y DevOps.

¿Qué significa realmente DevOps?

Antes:

Desarrolladores escribían código.

Operaciones lo desplegaban y mantenían.

Muchas veces había fricción: código que "funciona en desarrollo" pero falla en producción.

Con DevOps:

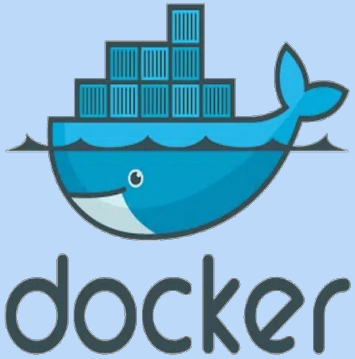
Se rompen esas barreras.

Ambos equipos trabajan juntos, usan herramientas compartidas, y automatizan procesos desde la codificación hasta el despliegue.

Aspecto	Agile (Desarrollo Ágil)	DevOps
Foco principal	Colaboración entre desarrolladores y clientes para entregar software funcional frecuentemente.	Colaboración entre desarrollo y operaciones para automatizar, desplegar y mantener el software de forma continua.
En qué parte del ciclo interviene	Desde la planificación hasta la entrega del código listo.	Desde la entrega del código hasta su despliegue y operación en producción.
Periodicidad	Iteraciones (sprints) cortas de 1-4 semanas.	Ciclo de vida continuo, con integraciones y despliegues frecuentes.
Objetivo	Responder rápidamente a cambios en las necesidades del cliente.	Hacer que esos cambios lleguen a producción de forma segura y rápida.

Analogía Visual

- Docker es como un contenedor de transporte:
- - Lleva diferentes cargas sin importar el barco.
- - Cada contenedor tiene lo necesario, sellado y aislado.
- - La ballena simboliza transportar software de forma segura y eficiente.



- El logotipo oficial es una ballena que transporta contenedores apilados en su lomo. Esta imagen es una metáfora visual muy clara de lo que hace Docker.

- La ballena representa a Docker, el motor que transporta y ejecuta aplicaciones.

- Los contenedores sobre la ballena simbolizan las aplicaciones y sus dependencias, que Docker lleva consigo de forma ligera y eficiente, como una especie de “barco de software”.

- **La idea detrás del símbolo es comunicar que Docker facilita el transporte de contenedores de manera sencilla y robusta, como si llevaras tus aplicaciones sobre una ballena viajando sin complicaciones entre distintos entornos (desarrollo, pruebas, producción, etc.).**

Problema Común sin Docker

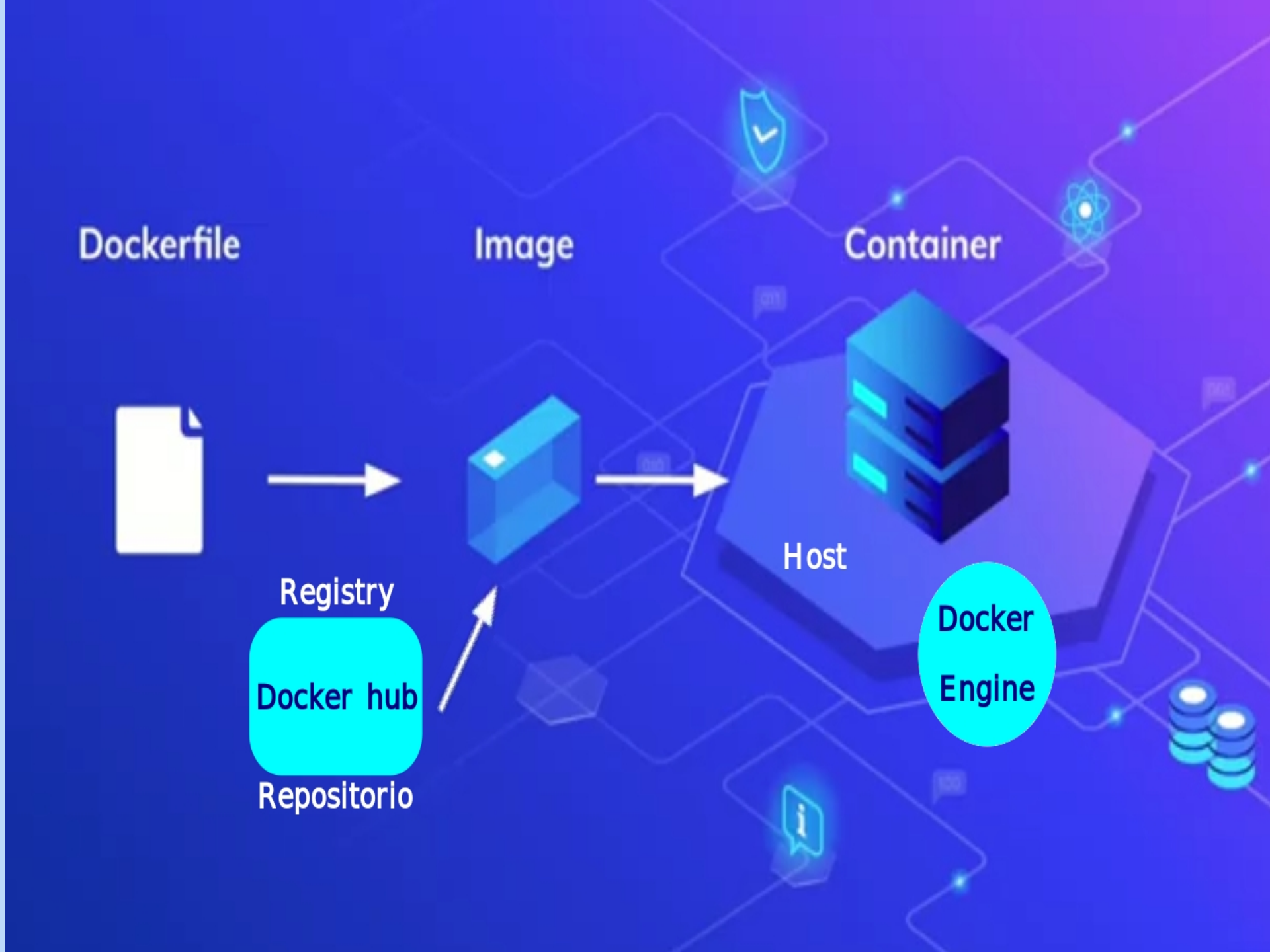
- App desarrollada en Python 3.10 con PostgreSQL 13 falla en producción.
- Producción tiene versiones distintas: Python 3.6, PostgreSQL 12.
- Con Docker: todo el entorno viaja junto → entornos idénticos.

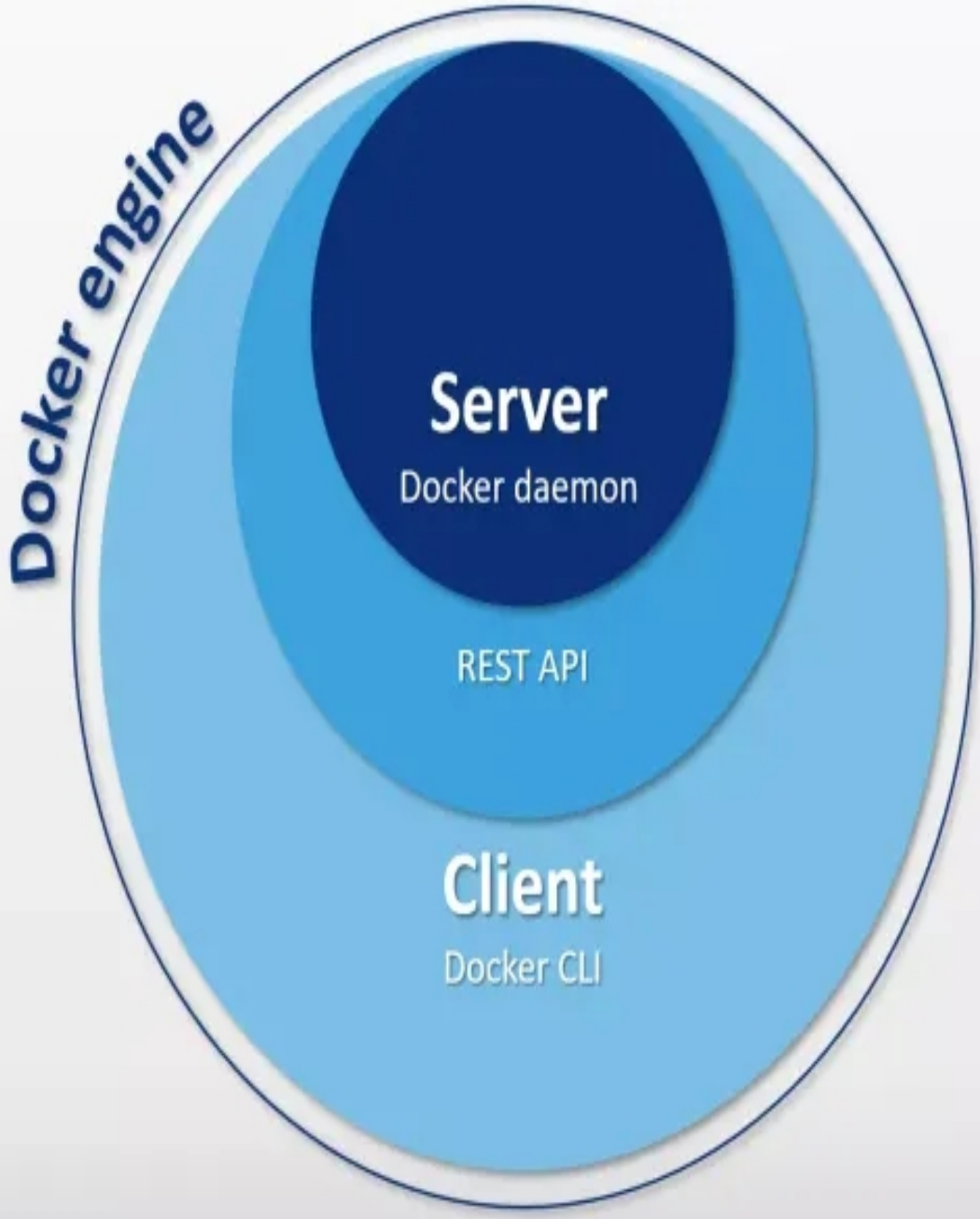
Cómo Docker Resuelve el Problema

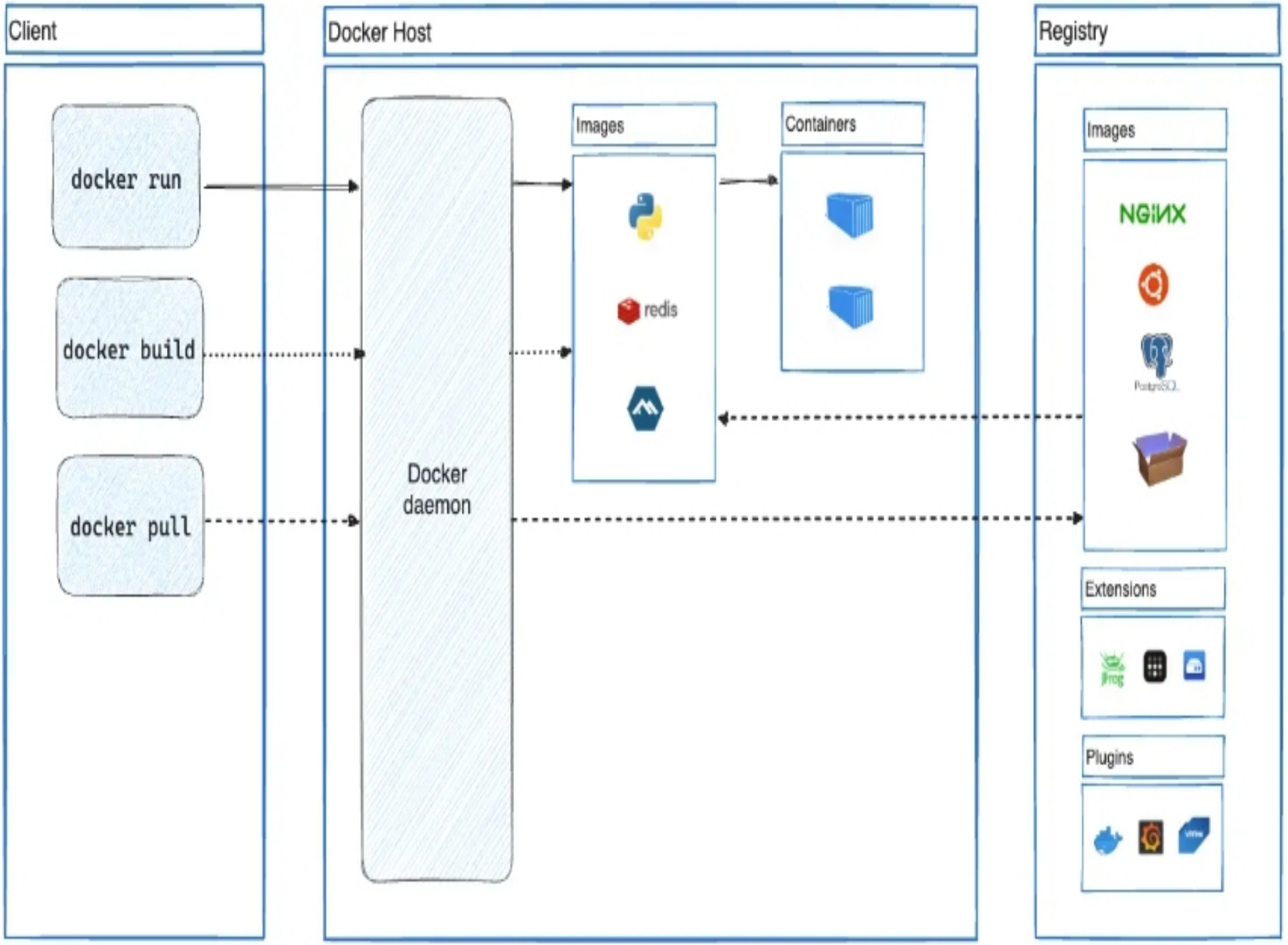
- Empaqueta app + dependencias en una imagen.
- Ejecuta la imagen como un contenedor aislado.
- Repite el proceso en cualquier entorno.
- Escalabilidad sencilla (1, 10, 100 contenedores).

Componentes Clave de Docker

- Docker Engine: Daemon + CLI + API.
- Contenedor: Entorno aislado de ejecución.
- Imagen: Plantilla para crear contenedores.
- Volumen: Almacenamiento persistente.
- Registry: Repositorio para compartir imágenes (Docker Hub).


Dockerfile
Image
Container
Host
Registry
Docker hub
Repositorio
Docker
Engine


Docker engine
Server
Docker daemon
REST API
Client
Docker CLI


Client
Docker Host
Registry
docker run
docker build
docker pull
Docker daemon
Images
Containers
redis
Images
NGINX
PostgreSQL
Extensions
Plugins

dockerhub

Search Docker Hub

Docker's curated GenAI catalog

Everything you need to build, scale, and deploy AI with ease.

View catalog now

Generative AI

AI Models

MCP Servers

Trusted content

Spotlight

CLOUD DEVELOPMENT

Build up to 39x faster with Docker Build Cloud

Introducing Docker Build Cloud: A new solution to speed up build times and improve developer productivity

AI/ML DEVELOPMENT

LLM everywhere: Docker and Hugging Face

Set up a local development environment for Hugging Face with Docker

SOFTWARE S

Take act

Bridge th needs

Docker en Diferentes Sistemas

- Linux: Funciona directamente.
- Windows/macOS: Requiere WSL 2 o una VM con Linux.
- Contenedores Windows: Requieren kernel de Windows.

¿Puedo correr un contenedor Windows en Linux ?

No. Por incompatibilidad de kernel:

- Los contenedores comparten el kernel del sistema operativo host.
- Un contenedor de Windows necesita el kernel de Windows, y Linux no lo tiene.
- No es como una máquina virtual que emula otro sistema.

¿Qué sí puedes hacer?

- Usar una máquina virtual con Windows sobre Linux (ej. con VirtualBox o VMware) y dentro de ella correr contenedores Windows.
- Usar contenedores Linux multiplataforma, que sí corren en Linux, Windows (con WSL2) y macOS (mediante una VM).

- Recomendación

  De ser posible, usa imágenes basadas en Linux (como ubuntu, node, mongo, etc.), ya que son más portables y compatibles.

Recomendación Final

- Usa contenedores Linux siempre que sea posible.
- Más portables, eficientes y ampliamente soportados.

Frases

- “La ballena lleva tu app.”
- “Contenedores ligeros, software potente.”
- “Tu código, en lomo de ballena.”