

Contenedores vs. máquinas virtuales.

Las dos aíslan aplicaciones para que no interfieran entre sí. La diferencia está en **qué comparten**: una máquina virtual lleva su propio sistema operativo; un contenedor usa el del anfitrión.

1 · La arquitectura, capa por capa



2 · Qué hace cada capa

1

Infraestructura

El servidor físico: CPU, memoria, disco y red

EN PALABRAS SENCILLAS

Es la computadora real donde todo se ejecuta. Las dos soluciones empiezan aquí: sin hardware no hay nada que repartir.

EXPLICACIÓN TÉCNICA

Hardware x86-64 o ARM con extensiones de virtualización (Intel VT-x / AMD-V) cuando se usan máquinas virtuales. En la nube, esta capa la administra el proveedor (AWS, Azure, Google Cloud).

VM

Hipervisor

Solo en máquinas virtuales

EN PALABRAS SENCILLAS

Es el «administrador» que divide un servidor en varias computadoras virtuales y le da a cada una su parte de procesador, memoria y disco.

EXPLICACIÓN TÉCNICA

Tipo 1 (bare-metal: VMware ESXi, Proxmox/KVM, Hyper-V) se instala directo sobre el hardware; tipo 2 (VirtualBox, VMware Workstation) corre sobre un sistema operativo. Emula o paravirtualiza dispositivos para cada VM.

EJEMPLOS

VMware ESXi

Proxmox VE

Hyper-V

KVM

VirtualBox



Sistema operativo invitado

Uno completo por cada máquina virtual

EN PALABRAS SENCILLAS

Cada máquina virtual instala su propio Windows o Linux, como si fuera una computadora nueva. Por eso pesan gigabytes y tardan en arrancar.

EXPLICACIÓN TÉCNICA

Cada VM tiene su propio kernel, drivers y servicios del sistema. Esto da un aislamiento muy fuerte y permite mezclar sistemas distintos (Windows junto a Linux), a cambio de más consumo de memoria y disco.



Sistema operativo anfitrión

Solo en contenedores: uno para todos

EN PALABRAS SENCILLAS

Todos los contenedores usan el mismo sistema operativo del servidor. No hay que instalar uno por cada aplicación, por eso son tan ligeros.

EXPLICACIÓN TÉCNICA

Los contenedores comparten el kernel del anfitrión (normalmente Linux). El aislamiento lo dan funciones del propio kernel: *namespaces* (qué ve cada proceso) y *cgroups* (cuántos recursos puede usar).



Motor de contenedores

Docker, Podman o containerd

EN PALABRAS SENCILLAS

Es el programa que crea, arranca y detiene los contenedores a partir de una «imagen», que es como una receta con todo lo que la app necesita.

EXPLICACIÓN TÉCNICA

Descarga imágenes de un registro (Docker Hub), las monta en capas de solo lectura y agrega una capa escribible por contenedor. Gestiona redes, volúmenes y el ciclo de vida mediante el runtime OCI (runc).

EJEMPLOS

Docker

Podman

containerd

CRI-O

2

Librerías y aplicación

Lo que de verdad quieres ejecutar

EN PALABRAS SENCILLAS

Tu aplicación y todo lo que necesita para funcionar (por ejemplo, PHP con sus extensiones). En ambos casos viajan juntas, así no chocan con otras apps.

EXPLICACIÓN TÉCNICA

En una VM se instalan dentro del SO invitado; en un contenedor van empaquetadas en la imagen. Por eso un contenedor «funciona igual en tu computadora que en el servidor».

3 · En la práctica

Aspecto	Máquina virtual	Contenedor
Arranque	Minutos (inicia un SO completo)	Segundos (inicia un proceso)
Tamaño típico	Gigabytes	Megabytes
Aislamiento	Muy fuerte (núcleo propio)	Por proceso (núcleo común)

Aspecto	Máquina virtual	Contenedor
Sistemas distintos	Sí: Windows y Linux en el mismo servidor	Comparten el núcleo del anfitrión
Densidad	Decenas por servidor	Cientos por servidor
Portabilidad	Imágenes de disco grandes	Imagen ligera que corre igual en cualquier lado
Ideal para	Otros sistemas operativos, laboratorios, software heredado	Aplicaciones web, microservicios, CI/CD

4 · ¿Cuándo conviene cada uno?

VM Usa una máquina virtual cuando...

- Necesitas otro sistema operativo (Windows Server en un equipo Linux).
- Armas un laboratorio de redes o seguridad (GNS3, Kali Linux).
- El software requiere su propio kernel o drivers.
- Buscas el máximo aislamiento entre clientes o equipos.

CT Usa contenedores cuando...

- Desarrollas apps web (por ejemplo, Laravel + MySQL con Docker Compose).
- Quieres que el proyecto funcione igual en todas las computadoras.
- Divides un sistema en microservicios que escalan por separado.
- Automatizas pruebas y despliegues (CI/CD).

5 · Para recordar



No compiten: se complementan. En la nube es muy común que los contenedores corran *dentro* de una máquina virtual: la VM aporta el aislamiento fuerte y los contenedores, la ligereza y la portabilidad.