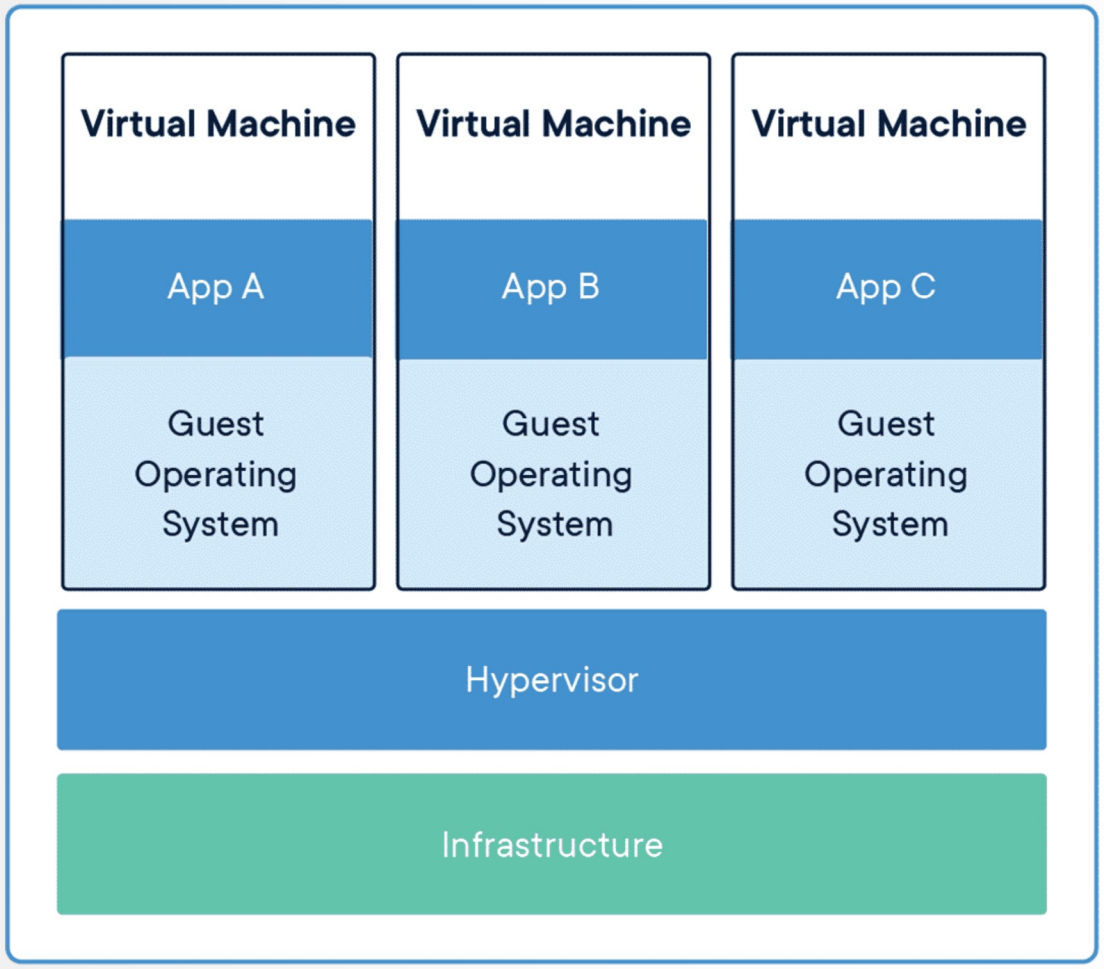
Linux Containers

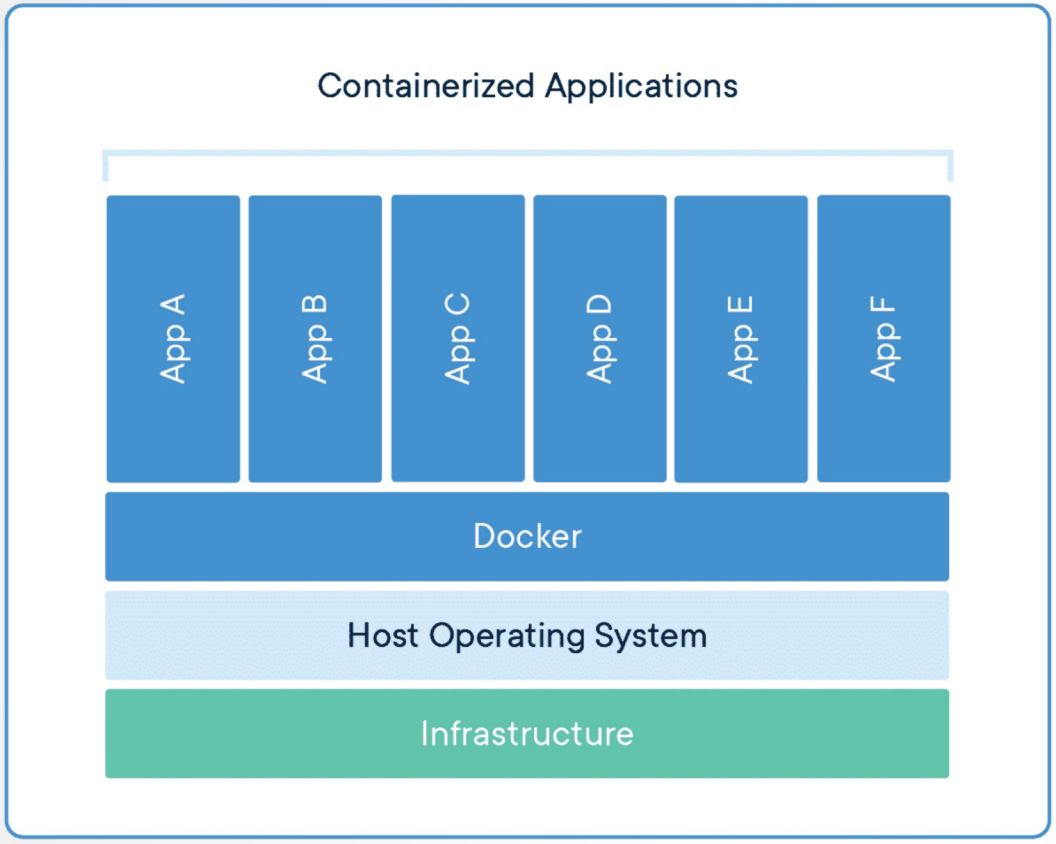
Cosa non è un container



VIRTUAL MACHINES

Le virtual machines (Vms) sono astrazioni di hardware fisico che, tramite un hypervisor, permettono di simulare più server in un unico server fisico. Ogni VM include una copia del sistema operativo, le applicazioni, i binari e librerie necessarie e il kernel. Richiedono decine di GB e possono essere relativamente lente ad avviarsi

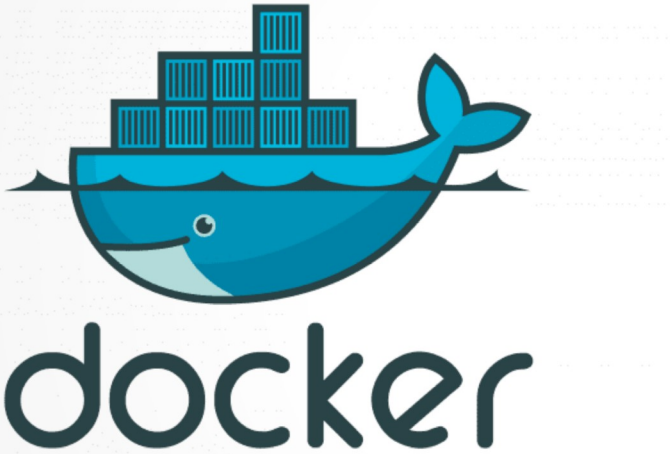
Cosa sono i container



CONTAINERS

I container sono un'astrazione a livello applicativo, che impacchetta insieme codice e dipendenze. E' possibile "far girare" più container all'interno dello stesso server e condividere il kernel del sistema operativo con altri container. All'interno dei container i processi vengono eseguiti in ambienti isolati tra di loro. I container richiedono meno spazio (fino a poche decine di MB) e meno risorse di una normale VM

Le tecnologie di “containerizzazione”



Caratteristiche comuni

Overhead e utilizzo risorse ridotto rispetto alle VM

I container non hanno bisogno di eseguire un sistema operativo aggiuntivo a quello già in esecuzione. I processi all’interno dei container vengono eseguiti come normali processi del kernel.

Caratteristiche comuni

Possibilità di montare volumi esterni al container

Utili con LXC, fondamentali con Docker, i volumi sono delle cartelle esterne alla root del container messe a disposizione, montate all'interno del container stesso. Esse possono risiedere in una cartella differente all'interno del filesystem gestito dall'host, un volume di un filesystem LVM/Btrfs/Zfs, un volume Ceph o tanto altro.

Caratteristiche comuni

Possibilità di limitare le risorse per container

I processi che vengono eseguiti all'interno del container possono aver accesso ad un sottoinsieme delle risorse totali del computer.

Alcuni esempi sono:

- Numero di core della CPU
- Quantità di memoria RAM
- Dimensione della root e dei volumi

Caratteristiche comuni

Deploy tramite immagine

Il deploy di un nuovo container avviene partendo da un’immagine di un container precompilata. Cambia la tecnologia ed il paradigma tra Docker ed LXC, ma il concetto di partenza rimane lo stesso (ne parleremo più avanti).

Caratteristiche comuni

Girano in ambiente Linux

Sia Docker che LXC per funzionare richiedono delle tecnologie implementate nel kernel Linux

Attenzione
A causa della natura intrinseca dei container, all'interno degli stessi possono essere eseguiti esclusivamente applicativi compilati per il sistema operativo Linux, poiché gestiti da un kernel Linux.

Caratteristiche comuni

Sono integrati come componente in altri progetti

Altri progetti usano queste tecnologie per erogare servizi aggiuntivi o per gestire i container più comodamente

Alcuni esempi:

- LXD, Proxmox per i container LXC
- Kubernetes, Portainer, docker-compose per i container Docker

Caratteristiche comuni

Possono essere privilegiati o non privilegiati

Un container non privilegiato subisce una rimappatura degli user e group ids. In altre parole l'utente root all'interno del container in realtà visto da fuori il container è un utente qualunque *senza privilegi*.

Questa è una caratteristica molto importante per quanto riguarda la sicurezza. Nel malaugurato caso in cui risultasse possibile ottenere i privilegi di root ed "evadere" dal container sfruttando una sua possibile falla di sicurezza, avrebbe comunque privilegi limitati.

Un container non privilegiato viene quindi considerato più sicuro di quello privilegiato, ma di contro non può eseguire alcune attività per cui sono necessari permessi speciali.

Differenze principali

Paradigma d’uso

I container Docker sono pensati per eseguire un singolo applicativo all’interno dello stesso. Nel caso l’erogazione di un servizio richiedesse l’esecuzione di più applicativi, questo paradigma richiederebbe l’avvio di più container distinti e collegati tra loro tramite reti interne al PC

LXC d’altro canto è strutturato in maniera più simile ad una VM classica. Viene eseguito un processo di init all’interno del container, il quale avvia e gestisce i vari servizi all’interno dello stesso, e all’interno del quale girano più applicazioni

Differenze principali

La persistenza delle modifiche

Le immagini di Docker sono pensate per non mutare nel tempo, o meglio, per tornare allo stato originale allo spegnimento del container.

Questo è il motivo che rende fondamentale l’esistenza dei volumi in ambiente Docker, poiché questi ultimi, generalmente, sono persistenti

Le immagini di LXC d’altro canto sono considerati come punto di partenza, come una nuova VM appena installata e pronta per il deploy del servizio desiderato.

I volumi in questo caso possono rendersi utili per per altri motivi.

Differenze principali

La struttura delle immagini

Quando viene compilata un’immagine di Docker il primo comando è generalmente il nome di un’altra immagine Docker da cui partire per aggiungere successivamente la proprie modifiche. In fase di download dell’immagine desiderata vengono a sua volta scaricate in maniera ricorsiva tutte le immagini dipendenti, e fuse in un unico rootfs tramite la tecnologia overlayfs.

Le immagini di LXC invece consistono in archivi (solitamente tar.gz) contenenti la struttura della rootfs che vengono estratti all’interno dello spazio messo a disposizione del container.

Differenze principali

Tutto e niente

Al giorno d’oggi molte delle funzionalità implementate da una tecnologia sono state implementate anche nell’altra tecnologia.

Quello che cambia e che **fa da differenza è la forma**, che si presta meglio a certi requisiti e meno ad altri.

Veniamo alla pratica