Cosa sono e come usare i Container

Marco Abbadini marco.abbadini@unibg.it

Outline

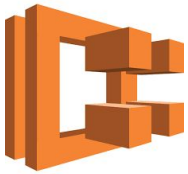
- Perché usare i container
- Cosa sono i container?
- Virtual Machine vs Container
- Principali comandi di Docker
- Sviluppare applicazioni con i container
- Costruire la propria immagine di un container
- Eseguire molteplici container con Docker Compose

Perché seguire questo intervento?

- I casi d'uso dei container continuano ad aumentare
- Negli ultimi due anni il numero di container è raddoppiato
- I container sono la tecnologia fondante dei maggiori servizi cloud

Azure Kubernetes Service

Google Kubernetes Engine



Amazon Elastic Kubernetes Service

- Molte tecnologie popolari sono anche tra le immagini disponibili su Docker Hub più utilizzate

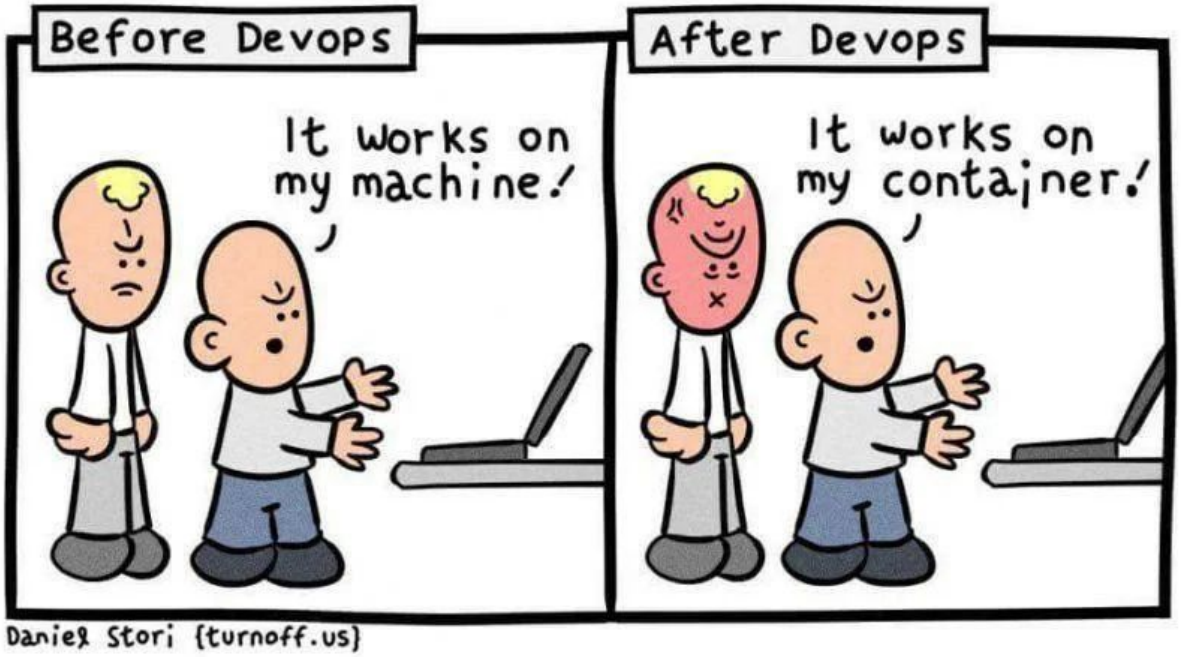
- (Motivazione bonus): poter scrivere nel CV "**Fluent with Docker**"

Perché usare i container?

- “**Sulla mia macchina funziona**”
- Configurazioni diverse in fase di sviluppo
- Configurazioni diverse in fase di rilascio
- Necessità di deployment veloce ed affidabile
- Separazione delle responsabilità developers-devops
- Controllo sulle risorse utilizzate dall’applicativo containerizzato

Perché non usare i container?

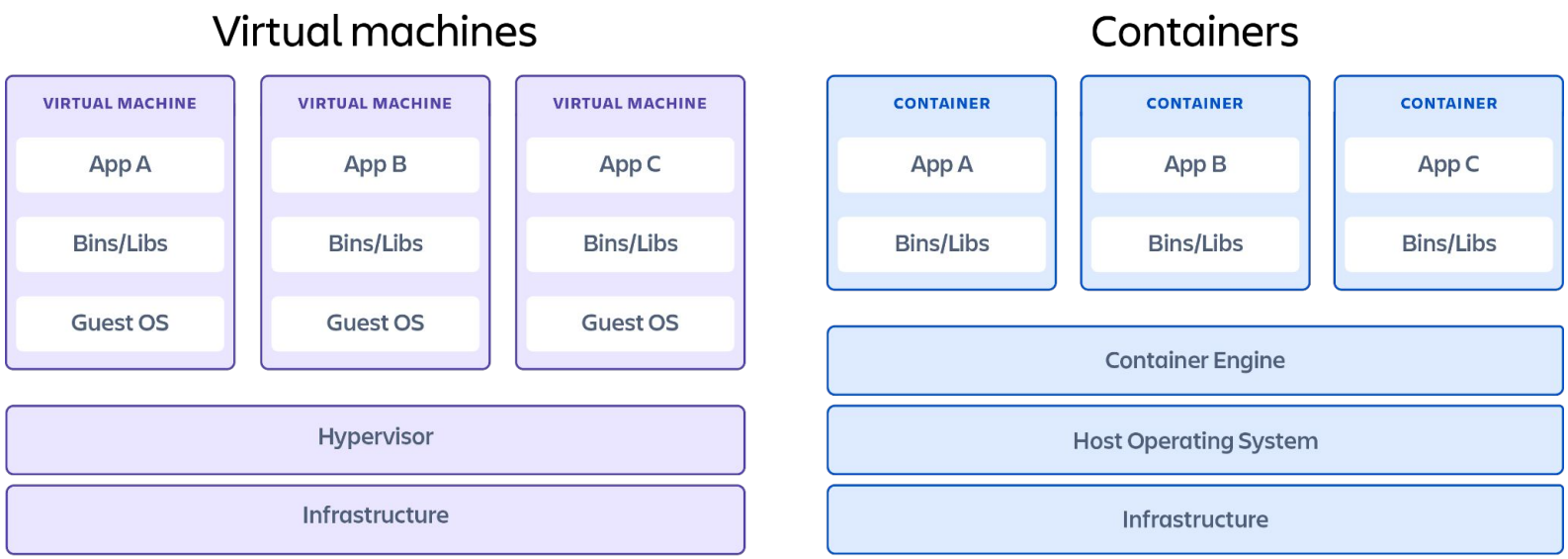
- Isolamento non forte come quello ottenuto mediante altre tecnologie
- Presenza di un forte legame tra l'applicazione ed interfacce specifiche del kernel sottostante
- Il tempo di start-up dell'applicativo non è un fattore critico
- L'applicazione opera in ambienti ben definiti e stabili
- Non si vuole condividere alcun tipo di risorsa software con altre applicazioni



Cosa sono i container?

- Pacchetti di software
- Contengono tutti gli elementi necessari all’esecuzione del software (runtime, librerie e utility di sistema)
- Creati mediante la virtualizzazione del sistema operativo
- Semplici processi con una visibilità “limitata” di quello che è il sistema host

Virtual Machine vs Container



- Pesanti
- Performance limitate
- Ogni VM esegue il proprio sistema operativo
- Virtualizzazione a livello HW
- Avvio in minuti
- Alloca la memoria richiesta
- Isolamento completo

- Leggeri
- Performance native
- I container condividono lo stesso sistema operativo
- Virtualizzazione a livello OS
- Avvio in millisecondi
- Richiede meno memoria
- Isolamento a livello di processo

Tecnologie alla base dei container

Esistevano da prima dell'avvento dei container, ma le loro dirette interfacce non sono di facile utilizzo

- **CGroups**: limitazione e isolamento dell'utilizzo delle risorse di sistema (memoria allocabile, tempo di CPU, I/O, numero di processi, etc.)
- **Chroot**: cambio della **root** del file system percepita dai processi
- **Linux namespaces**: limitazione di ciò che è visibile nel sistema (mount, processi, rete, utenti, etc.)

NOTA: Ne vengono usate anche altre per garantire maggiore sicurezza del sistema ospite (Linux capabilities, seccomp) e isolamento tra i container (AppArmor, SELinux)

Open Container Initiative (OCI)

The Open Container Initiative (OCI) is a project, with the purpose of creating open industry standards around container formats and runtime

- Container **Image**

  The image manifest contains metadata about the contents and dependencies of the image including one or more file system serialization archives that will be unpacked to make up the final runnable file system

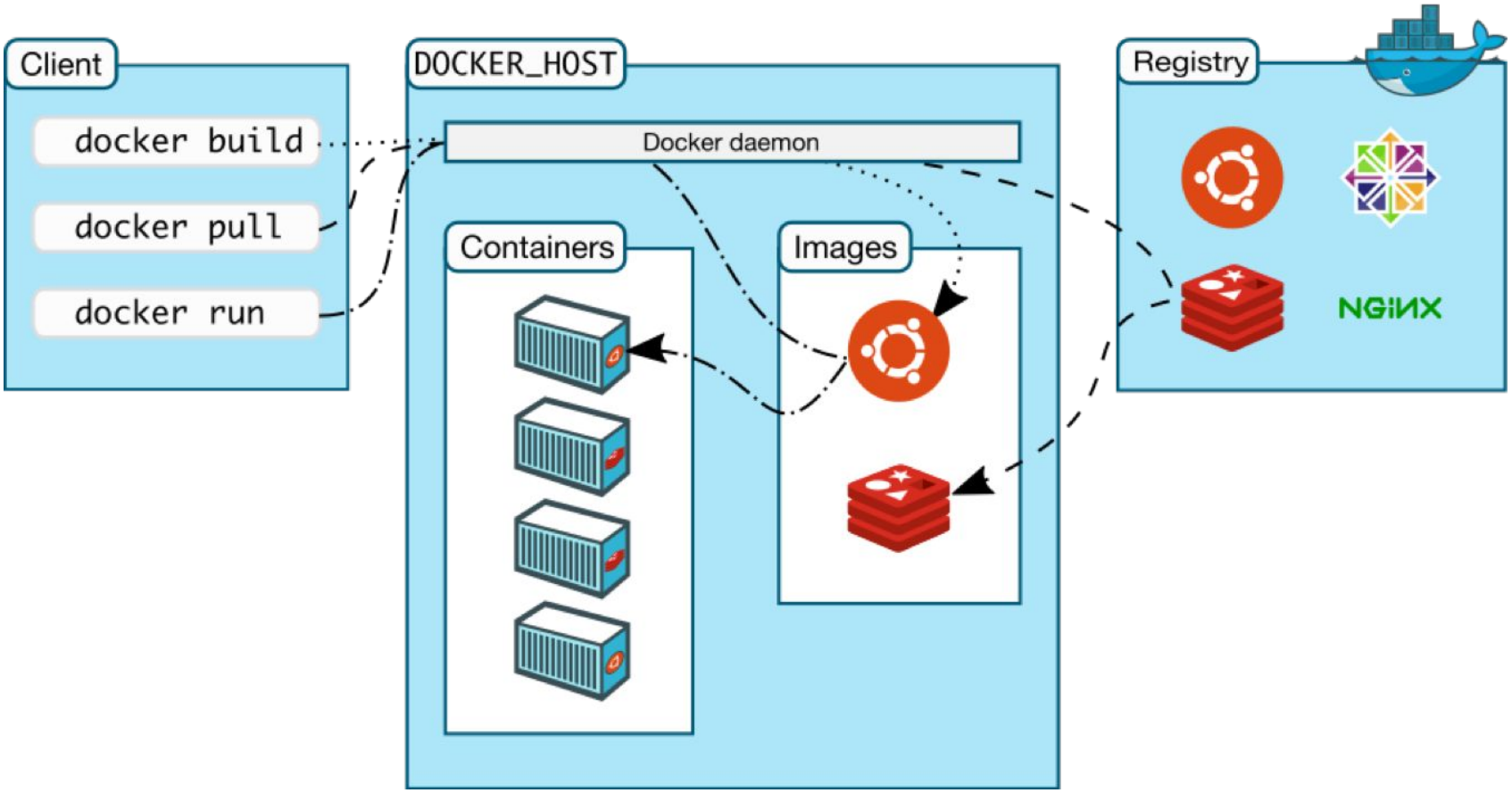
  Build tools: BuildKit, Buildah

- Container **Runtime**

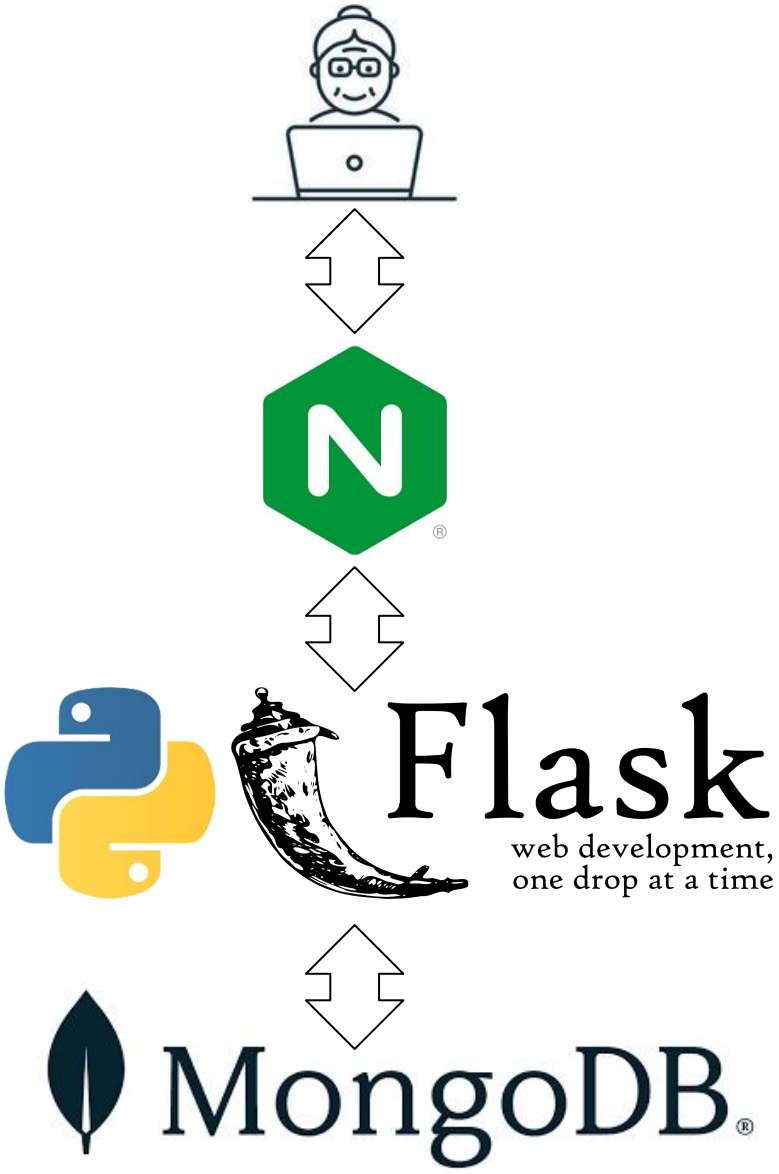
  An implementation would download the image, unpack it into a file system bundle, and then run it

  Runtimes: Runc, Kata containers, gVisor, Firecracker

Docker


Client
docker build
docker pull
docker run
DOCKER_HOST
Docker daemon
Containers
Images
Registry
NGINX

Applicazione dimostrativa



Il nostro client vuole visualizzare il contenuto del nostro blog

Le richieste vengono mandate al nostro HTTP server il quale risolve le richieste di contenuto statico e inoltra le richieste dei servizi al server applicativo

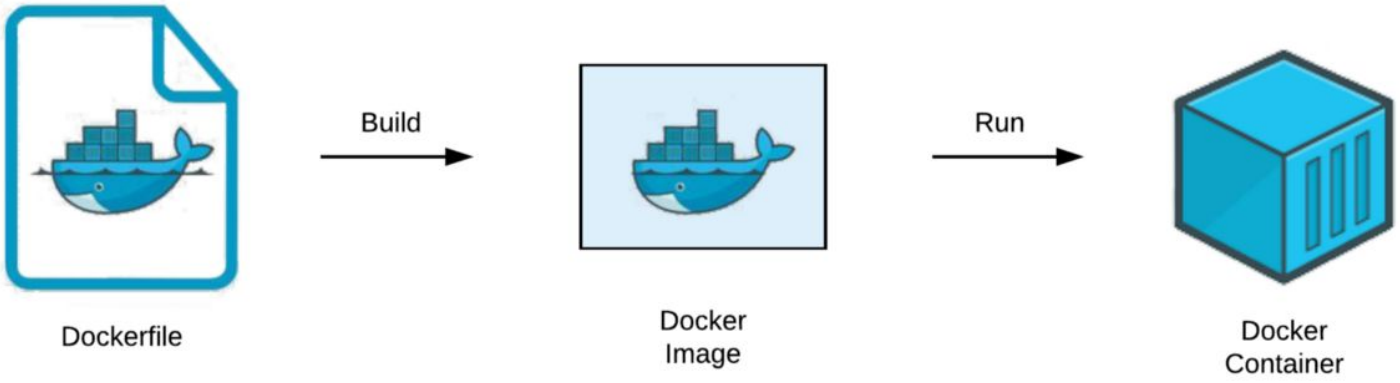
Il server applicativo al fine di rispondere alle richieste dinamiche interroga la base di dati

Il database di documenti risolve le interrogazioni e risponde al server applicativo

Costruire proprie immagini dei container

Docker può costruire immagini leggendo le istruzioni di configurazione elencate in un **Dockerfile**

Un Dockerfile è un file di testo che contiene tutti i comandi necessari all'utente per assemblare l'immagine

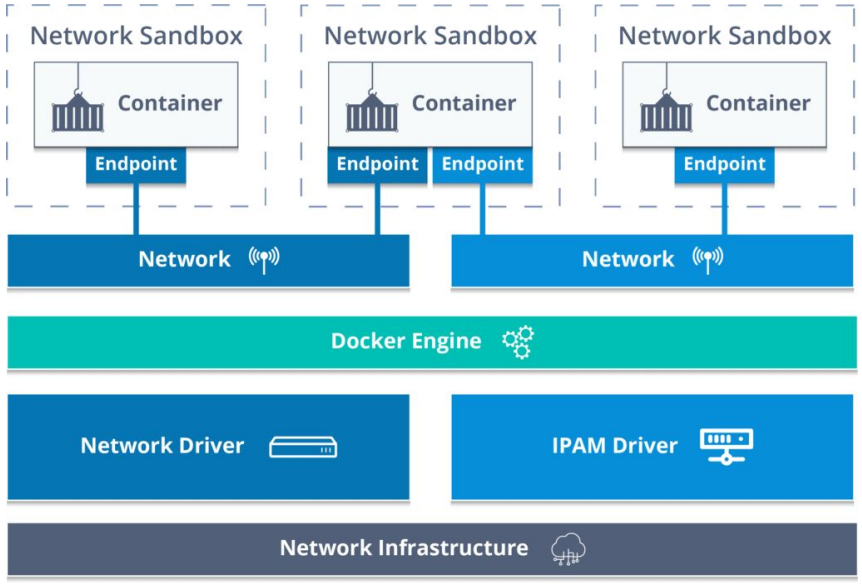


Configurare la rete tra più container

La forza dei container, e in generale delle architetture a microservizi, è che questi possono interagire tra loro

Tipologie

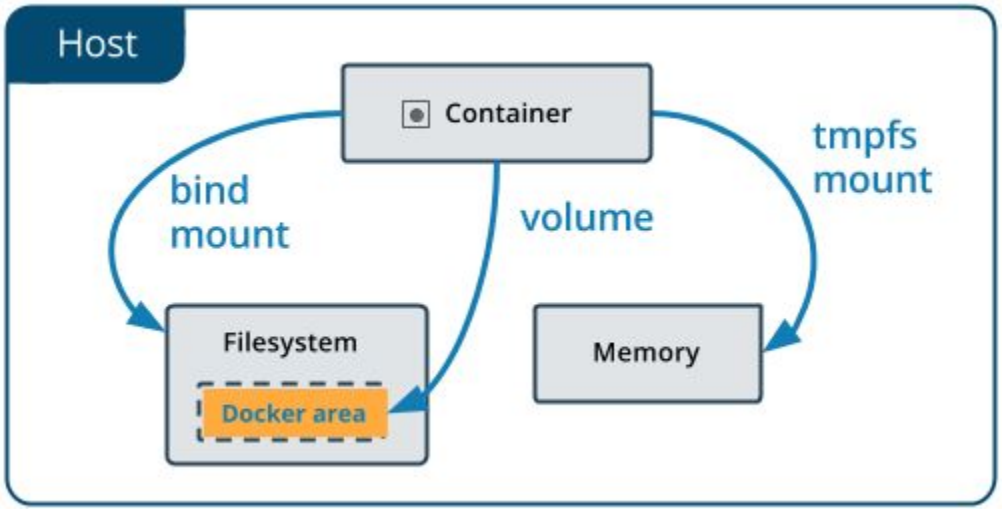
- **bridge** (default): le reti bridge sono usate quando le applicazioni eseguite all'interno di container hanno bisogno di comunicare
- **host**: rimuove l'isolamento di rete tra il container e l'host, il container può quindi usare la rete dell'host direttamente
- **none**: disabilita ogni forma di networking



Ne esistono anche di altre, ma vengono principalmente utilizzate quando i container sono distribuiti su diversi host

Configurare l’uso dei volumi

I volumi sono il meccanismo preferito per la persistenza dei dati generati dai container



Tipologie

- Host: memorizzato sull’host in una location specifica
- Anonimo: memorizzato in una location gestita da Docker
- Nominato: equivalente al volume anonimo, ma identificabile con il nome


USE CONTAINERS THEY SAID...
RENA
MONROVIA
DEVS CAN DEPLOY THEIR
APPLICATIONS THEMSELVES THEY SAID...
imgflip.com

Automatizzare l’esecuzione dei container

Compose è un tool per la definizione e l’esecuzione di applicazioni composte da multipli container

Utilizza un YAML file per configurare i servizi dell’applicazione e con un singolo comando permette di creare e avviare tutti i servizi dalla configurazione definita

Caratteristiche

- Isolamento di environment multipli in un singolo host
- Preserva i dati presenti nei volumi durante la creazione dei container
- Ricrea i container solo se questi sono cambiati
- La configurazione supporta l’uso di variabili permettendo quindi il supporto diversi environment


KUBERNETES

Un’alternativa nota

- Scoperta dei servizi e bilanciamento del carico
- Orchestrazione dello storage
- Rollout e rollback automatizzati
- Ottimizzazione dei carichi
- Self-healing
- Gestione di informazioni sensibili e della configurazione

Per maggiori informazioni, la pagina ufficiale di Kubernetes:

https://kubernetes.io/it/docs/concepts/overview/what-is-kubernetes/

HAPPY SAILING