Docker

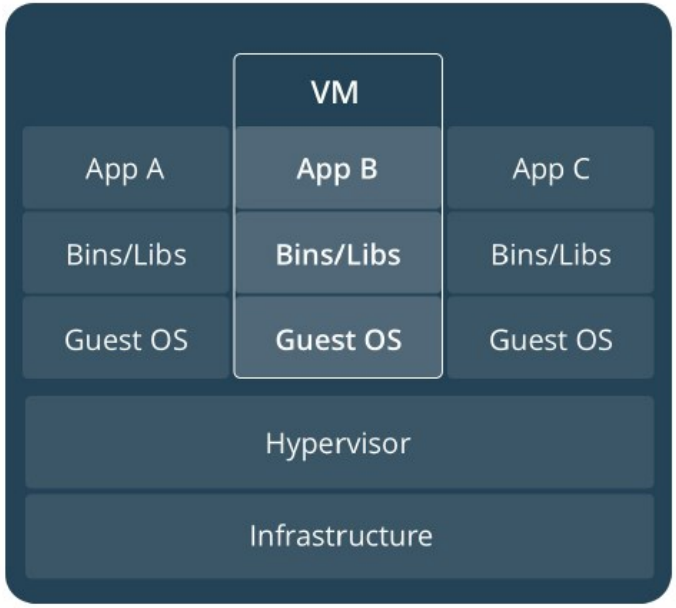
Ingegneria del software

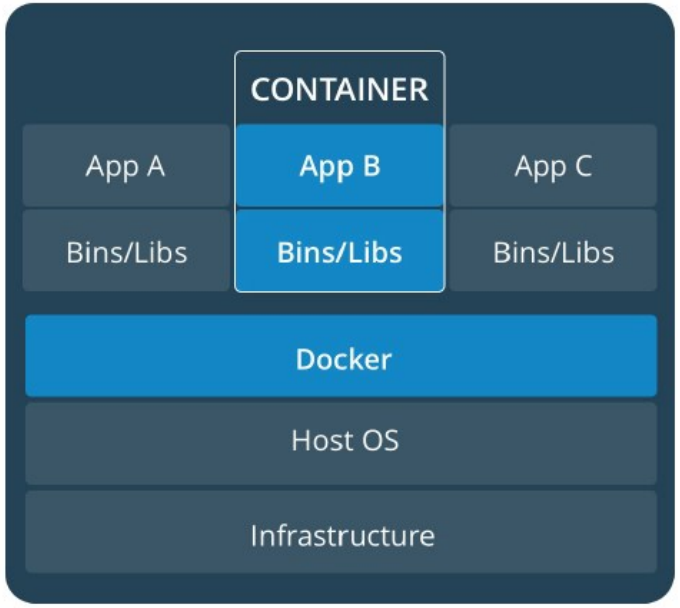
Vincenzo Bonnici
Corso di Laurea in Informatica
Dipartimento di Scienze Matematiche, Fisiche e Informatiche
Università degli Studi di Parma

2025-2026

Container vs macchine virtuali

- Una VM (virtual machine) ha hardware emulato e ospita un intero sistema operativo (guest), che è separato dal sistema operativo host.
-
- Un container non emula alcun hardware e condivide il kernel del sistema operativo con l'host → meno isolamento, più efficienza





Docker

- **Docker** è uno strumento per la creazione, la gestione e l'orchestrazione di **container** di applicazioni

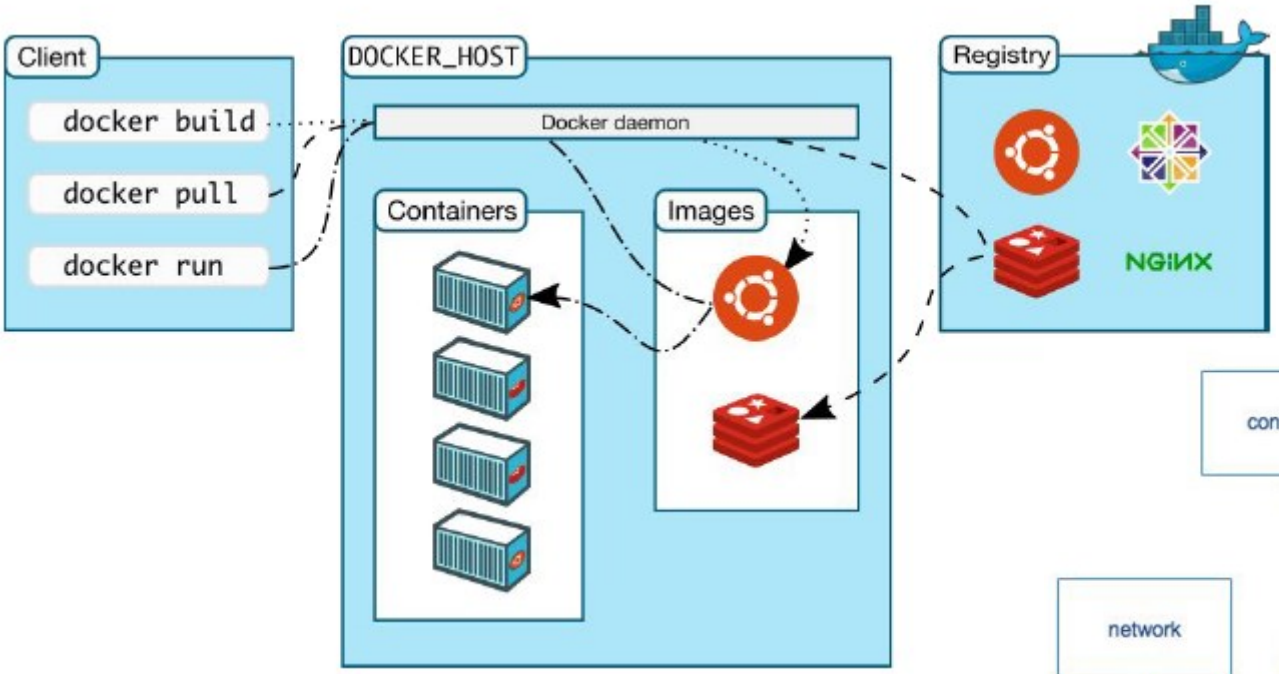
- L'obiettivo di Docker è quello di **ottimizzare il processo di sviluppo, test, consegna e distribuzione delle applicazioni**, mediante la creazione di pacchetti di ogni componente dell'applicazione in un contenitore separato

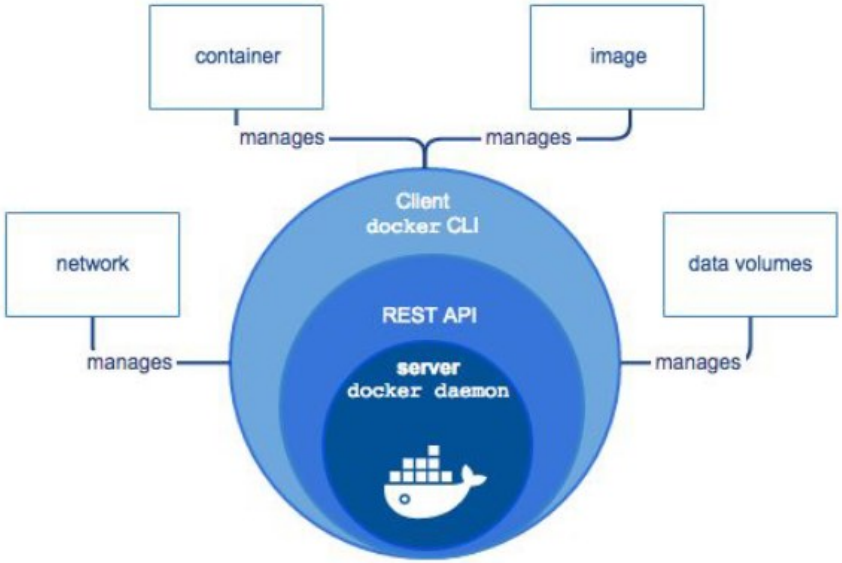
- Progettato per gli sviluppatori di software:
- Si occupa di tutte le fasi coinvolte nello **sviluppo del software**

- Il passaggio dalle macchine virtuali ai contenitori Docker non è immediato
  - Potrebbe essere necessario modificare l'applicazione o la sua configurazione
  - La radice del problema è che l'ambiente di esecuzione di un container Docker normalmente non è un sistema UNIX completo

Docker: architettura

- **Architettura client-server** per la gestione di immagini, container, volumi e reti virtuali
  - Client e server possono essere eseguiti su computer diversi
  - L'architettura è simile a libvirt, ma con più funzionalità, inclusa la capacità di interagire con un registro di immagini (https://hub.docker.com/)





Docker: componenti

Immagine

- Un'immagine è un **modello portatile**, accessibile in modalità di **sola lettura**, che contiene tutte le **istruzioni** necessarie per creare un contenitore Docker
- Contiene tutte le **dipendenze** necessarie per un componente software (codice o binario, run-time, librerie, file di configurazione, variabili d'ambiente, ...)
- Un'immagine può essere **definita da un archivio** del file system (tarball)
- Oppure può essere definito **estendendo un'immagine precedente** con un elenco di istruzioni specificate in un file di testo (noto come Dockerfile)

Registro Docker

- **Database** per archiviare in modo efficiente le **immagini**
- I registri possono essere pubblici (come DockerHub) o privati per un'organizzazione

Contenitore (Container)

- Un contenitore Docker è un'**istanza eseguibile** di un'immagine Docker
- Definito dall'**immagine** e dalla c**onfigurazione** specificata al momento della creazione
- Un container può essere creato, avviato, arrestato, migrato, distrutto, connesso a una o più reti virtuali, associato a uno o più volumi di dati...
- Il container è l'**unità di sviluppo**, test, consegna e distribuzione dell'applicazione, presupponendo che Docker venga utilizzato come supporto operativo
- Qualsiasi modifica al file system visibile a un contenitore non si riflette sull'immagine (l'immagine è di **sola lettura**)
- E' possibile definire in che misura un contenitore è **isolato dall'host**
  - Accesso al file system host e ai dispositivi speciali, limitazioni all'allocazione della memoria e all'utilizzo della CPU.

Docker: componenti

Rete (Network)

- **Reti virtuali**, implementate per mezzo di switch virtuali e iptables
- Le reti **bridge** limitano la connettività ai contenitori su un singolo host
- Le reti **overlay** consentono la connettività dei contenitori tra diversi host
  - In genere utilizzando l'incapsulamento VXLAN

Volume

- Un volume è una **directory** che può essere associata a uno o più contenitori
- La sua durata è indipendente dai contenitori che lo utilizzano
- Utilizzato per **condividere lo storage** tra diversi container, o comunque storage che può **sopravvivere al container** dell'utente

Docker: componenti

Servizio

- Un servizio Docker è un **set di contenitori** che sono **repliche** della stessa immagine e che insieme forniscono un servizio con carico bilanciato
- I servizi vengono utilizzati per distribuire i contenitori «in produzione»
- Un servizio può essere aumentato o ridotto a seconda del carico di input

---

Pila (**Stack**)

- Un **insieme di servizi interdipendenti** che interagiscono per implementare un'applicazione completa:
  - Es: Un'applicazione web per la condivisione di immagini potrebbe essere costituita da (i) un servizio per l'archiviazione e la ricerca di immagini; (ii) un servizio di interfaccia web per gli utenti; e (iii) un servizio per codificare/decodificare immagini

Dockerfile

- Un file di testo che contiene una **ricetta per creare un'immagine**
- Un'immagine dovrebbe essere un componente ben definito e contenere **solo il software effettivamente necessario per un'attività ben definita**

```
# Start from an official image with the Python runtime
FROM python:2.7-slim
# Set the container current working directory (PWD) to “/chess”
WORKDIR /chess
# Copy files from current host directory to the /chess directory in the container
ADD . /chess
# Install some packages
RUN apt-get update && apt-get install -y libcgroup acl
# Flag that the software inside this image listens on port 9000
EXPOSE 9000
# Define an environment variable
ENV PLAYER Ghost
# Specify the command to be run inside the container when it’s started
CMD ["python", "./chess.py"]
```

Docker: comandi

Abilitare l'utente ad usare Docker

$ sudo usermod -aG docker ${YOUR_USERNAME}

Verificare che Docker sia attivo e funzionante:

$ sudo systemctl status docker
[...]
Active: active (running) since ...
[...]

Ricerca in un registro (ad esempio, quello pubblico predefinito)

[user@host ~]$ docker search nginx

- L'output mostra un elenco di immagini che corrispondono alla parola chiave
- Le immagini sono ordinate in base al numero decrescente di voti dati dagli utenti Docker
- Prima le immagini più popolari

Docker: comandi

Gestire le immagini

- Elencare le immagini esistenti

```
[user@host ~]$ docker image ls
```

- Importare un'immagine da un registro

```
[user@host ~]$ docker image pull base/archlinux
```

- Rimuovere un'immagine (localmente)

```
[user@host ~]$ docker image rm ubuntu
```

- Visualizzare informazioni dettagliate su un'immagine

```
[user@host ~]$ docker image inspect ubuntu
```

- Rimuovi le immagini inutilizzate

```
[user@host ~]$ docker image prune
```

Docker: comandi

Costruire una immagine a partire da un Dockerfile

- Spostarsi nella directory contenente il Dockerfile

```
[user@host ~]$ cd /path/to/dockerfiledir
```

- Costruisci un'immagine dal Dockerfile nella directory corrente, dandogli un nome (tag) "myimg”

```
[user@host ~]$ docker build -t myimg .
```

- La nuova immagine verrà memorizzata insieme alle altre già disponibili sull'host

- Ogni Dockerfile è normalmente archiviato in una directory separata
  - Il nome del file deve essere "Dockerfile"

Creare e lanciare container

- Creare un contenitore e avviarlo (all'interno dello stesso comando)

```
[user@host ~]$ docker run -it --name ubu1 ubuntu /bin/bash
```

  - L'argomento ubuntu si riferisce al **nome di un'immagine** disponibile
  - L'argomento /bin/bash specifica il **comando che deve essere eseguito** dal contenitore
  - Se presente, sovrascrive il comando specificato all'interno dell'immagine (CMD)
  - L'opzione -t specifica **l'allocazione di un terminale**
  - L'opzione -i specifica che il **comando è interattivo** (è una shell)
  - L'opzione -d viene utilizzata per eseguire il contenitore in **background**

- È possibile creare e avviare con comandi separati:

```
[user@host ~]$ docker create -it --name ubu1 ubuntu /bin/bash
[user@host ~]$ docker start -i ubu1
```

Pubblicare le porte esposte

- Un'immagine può esporre una porta TCP/UDP tramite la direttiva EXPOSE nel Dockerfile

- Quando viene avviato un container, è possibile mappare ogni porta esposta a una porta host, per abilitare l'accesso dalla rete esterna host

- Questa mappatura viene specificata tramite l'opzione -p dei comandi run o create
  - Es: Avviare un contenitore del server Web che espone la porta 80, mappandola sulla **porta 8000 dell'host**
  - [user@host ~]$ docker run -p 8000:80 apache /usr/bin/apacheserv --daemon

Docker: comandi

Gestire i container

- Mostra tutti i contenitori in esecuzione

[user@host ~]$ docker ps

- Mostra tutti i contenitori (in qualsiasi stato)

[user@host ~]$ docker ps –a

- Include i contenitori che non sono attualmente in esecuzione
  - Riavviare un contenitore (specificato in base al nome o all'ID)

[user@host ~]$ docker restart ubu1

- Arrestare un contenitore

[user@host ~]$ docker stop ubu1

- Rimuovere un contenitore

[user@host ~]$ docker rm ubu1

- Altri comandi: uccidi, ispeziona, metti in pausa, riprendi, ...
- Comandi equivalenti in forma canonica:

  [user@host ~]$ docker container COMMAND

Gestire i volumi

- Creare un volume chiamato "myvol"

```
[user@host ~]$ docker volume create myvol
```

- Rimuovi "myvol"

```
[user@host ~]$ docker volume rm myvol
```

- Mostra l'elenco dei volumi disponibili nell'host

```
[user@host ~]$ docker volume ls
```

- Eseguire un contenitore, rendendo disponibile il contenuto del volume "myvol" nel percorso /mntvol all'interno del contenitore

```
[user@host ~]$ docker run -v myvol:/mntvol -it ubuntu /bin/bash
```

- Eseguire un contenitore, rendendo disponibile il contenuto della directory host "/home/user/tmp" nel percorso /mnt all'interno del contenitore

```
[user@host ~]$ docker run -v /home/user/tmp:/mnt -it ubuntu /bin/bash
```

«Data volume containers»

- È possibile creare volumi senza nome, associati implicitamente a un contenitore (ma comunque indipendenti dalla durata del contenitore)
    - Es: crea un volume senza nome, rendendolo disponibile in "/myvol"

```
[user@host ~]$ docker run -v /myvol --name ubu1 -it ubuntu /bin/bash
```

    - In realtà, un nome di volume viene assegnato automaticamente

- I volumi senza nome possono essere collegati ad altri contenitori
    - Es: Avvia un contenitore importando volumi da un altro contenitore chiamato ubu1

```
[user@host ~]$ docker run --volumes-from ubu1 --name ubu2 -it ubuntu /bin/bash
```

    - I volumi importati vengono montati nel file system ubu2 agli stessi punti di montaggio utilizzati all'interno di ubu1

- Un contenitore come ubu1 è chiamato "Contenitore del volume di dati"