Capitolo 2 – Architettura di Docker

Per utilizzare Docker in modo efficace, è fondamentale comprendere la sua architettura e come interagiscono i vari componenti.

2.1 Componenti Principali

Docker Engine

- È il cuore di Docker.
- Si occupa di eseguire container e gestire immagini.
- Funziona come **server client-daemon**: la CLI comunica con il daemon per eseguire le operazioni.

Docker Daemon (`dockerd`)

- Processo in background che gestisce container, immagini, volumi e reti.
- Ascolta richieste dalla CLI o da API.
- Avvia container secondo le specifiche delle immagini.

Docker CLI (`docker`)

- Interfaccia a linea di comando per interagire con Docker.
- Permette operazioni come:
  - Creare, avviare e fermare container
  - Costruire e scaricare immagini
  - Gestire volumi e network

Docker Registries

- Repository per immagini Docker.
- **Docker Hub**: registry pubblico più diffuso.
- È possibile avere registry privati per sicurezza o ambienti interni.

2.2 Come Funzionano i Container

Un container è un processo isolato che condivide il kernel del sistema operativo host, ma ha:

- Filesystem isolato
- Network virtuale
- Volumi per persistenza dati
- Limiti risorse configurabili (CPU, memoria)

Vantaggi:

- Avvio immediato
- Minore overhead rispetto a una VM completa
- Facile da distribuire

2.3 Immagini Docker

- Un'immagine è un **modello di container immutabile**.
- Contiene:
  - Sistema operativo minimo
  - Applicazioni
  - Dipendenze
- Le immagini possono essere **layered**, permettendo riuso ed efficienza.
- Esempio di layer:
  - `ubuntu:22.04`
  - Installazione Python
  - Installazione librerie app
  - Codice applicativo

2.4 Networking Docker

- **Bridge network**: default per container sullo stesso host.
- **Host network**: il container usa direttamente la rete dell'host.

- **Overlay network**: per container distribuiti su più host (swarm o Kubernetes).

2.5 Volumi e Persistenza

- I container sono effimeri: i dati all’interno vengono persi se il container viene rimosso.
- **Volumi**: spazio persistente gestito da Docker.
- **Bind mount**: collegamento diretto a cartelle dell’host.

2.6 Flusso Operativo Base

1. La CLI invia comandi al Docker Daemon.
2. Il Daemon verifica immagini disponibili.
3. Se necessario, scarica l’immagine dal registry.
4. Avvia il container con filesystem, network e volumi configurati.
5. Il container esegue il processo richiesto in isolamento.

2.7 Conclusione

Comprendere l’architettura Docker permette di:

- Pianificare correttamente deployment e rete
- Gestire volumi e dati in sicurezza
- Ottimizzare immagini e risorse
- Evitare errori comuni di isolamento e rete

Nel prossimo capitolo vedremo come **installare e configurare Docker** su Linux, pronto per l’uso in produzione.

Revision #1
Created 2026-02-26 14:26:21 UTC by Pe
Updated 2026-02-26 14:26:45 UTC by Pe