

Sviluppo software in gruppi di lavoro complessi¹

Mattia Monga

Dip. di Informatica
Università degli Studi di Milano, Italia
mattia.monga@unimi.it

Anno accademico 2024/25, II semestre

¹ 2025 M. Monga. Creative Commons Attribuzione — Condividi allo stesso modo 4.0 Internazionale.
<http://creativecommons.org/licenses/by-sa/4.0/deed.it>



Svigruppo

Monga

Continuous
Integration &
Delivery

Docker

Lezione XIII: Continuous integration & delivery



Svigruppo

Monga

Continuous
Integration &
Delivery

Docker

Configuration Management

Esplicitazione delle dipendenze

Test d'unità, d'integrazione, d'accettazione

Build automatizzate

Deployment simulato o automatizzato

↔ *Continuous Integration & Delivery/Deployment*



Svigruppo

Monga

Continuous
Integration &
Delivery

Docker

- Martin Fowler 2001, 2006, nuova versione del 2024 (uso delle *branch*!)
(<https://www.martinfowler.com/articles/continuousIntegration.html>)
- Tradizionalmente, l'integrazione è una delle parti più lunghe e rischiose dei progetti *software*
- CI $\rightsquigarrow$ integrazione continua (incrementale) per minimizzare il rischio di fallimento
 - *Reduced Deployment Risk*
 - *Believable Progress*
 - *User Feedback*



Svigruppo

Monga

Continuous
Integration &
Delivery

Docker

- 1 Lavoro su una copia locale sulla *macchina di sviluppo*
- 2 *Build* vs. *Compile*: oltre alla costruzione esecuzione di test
- 3 *build* funzionante sulla *macchina di sviluppo*
- 4 Caricamento sulla *macchina d'integrazione*
- 5 *build* funzionante sulla *macchina d'integrazione*

Almeno una volta al giorno.



Svigruppo

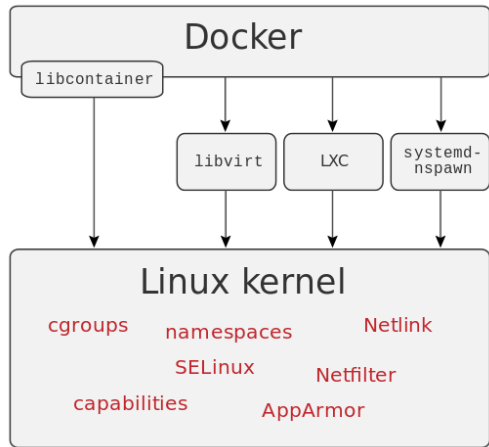
Monga

Continuous
Integration &
Delivery

Docker

Cambiato drasticamente negli ultimi 10 anni, grazie alla facilità di allestire **ambienti virtuali**

- Emulatori (QEmu/VMWare)
- “Meta-sistemi operativi” (*hypervisor*: Hyper-V, supporto *hardware* Intel-VT, AMD-V)
- *container*: forme di isolamento regolate dal sistema operativo che partiziona le risorse con politiche stringenti (UML, LXC, Docker)



- Sfrutta anche un *file system* a **layer**
- Le **immagini docker** (dati persistenti, costruite per *layer*) vengono istanziate in *container* di processi
- Le immagini si costruiscono con un Dockerfile

```
FROM python:3.13-slim-bookworm
LABEL maintainer="monga@di.unimi.it"
WORKDIR /app
ADD . /app
RUN pip install -r requirements.txt
EXPOSE 80
CMD ["python", "app.py"]
```



Svigruppo

Monga

Continuous
Integration &
Delivery

Docker

Approfondimento consigliato:

- Lezione di Massimo Santini su Docker:
https://www.youtube.com/watch?v=h_1xsPnwk0k
- <https://github.com/mapio/docker-tutorial>



Svigruppo

Monga

Continuous
Integration &
Delivery

Docker

Il *build* non dovrebbe impiegare più di 10 minuti, altrimenti si perde il *feedback* immediato. Che fare se è necessario più tempo?

- *Deployment pipeline*: il *build* è spezzato in più fasi, *commit build*, *slower tests build*, ecc.
- Il *commit build* impiega meno di 10 min, il resto parte poi



Svigruppo

Monga

Continuous
Integration &
Delivery

Docker

- Il sistema dove avviene l'**integrazione** dovrebbe essere il più possibile "simile" a quello di **produzione**
- Il *deployment* verso l'ambiente di produzione può essere automatizzato (garantendo così un maggior controllo sulla effettiva configurazione in uso!)