Designers

Developers

Progettare, sviluppare ed erogare servizi pubblici digitali

Sviluppare in ottica cloud native: fornire un software come servizio SaaS

Fabio Bonelli
Esperto di Open Source

13 luglio 2021

DIPARTIMENTO PER LA TRASFORMAZIONE DIGITALE

AgID

FormezPA

Perché sviluppare in ottica cloud native

- **Riduzione dei costi di manutenzione**
  Nessuna manutenzione hardware e aggiornamenti software gestiti
- **Scalabilità e resilienza**
  Più semplice supportare carichi di lavoro importanti
- **Flessibilità**
  Le risorse possono essere allocate all'occorrenza
- **Modernità e interoperabilità**
  API e automatismi

Come arrivarci

- Container
- Microservizi
- 12 factor app

12 factor app

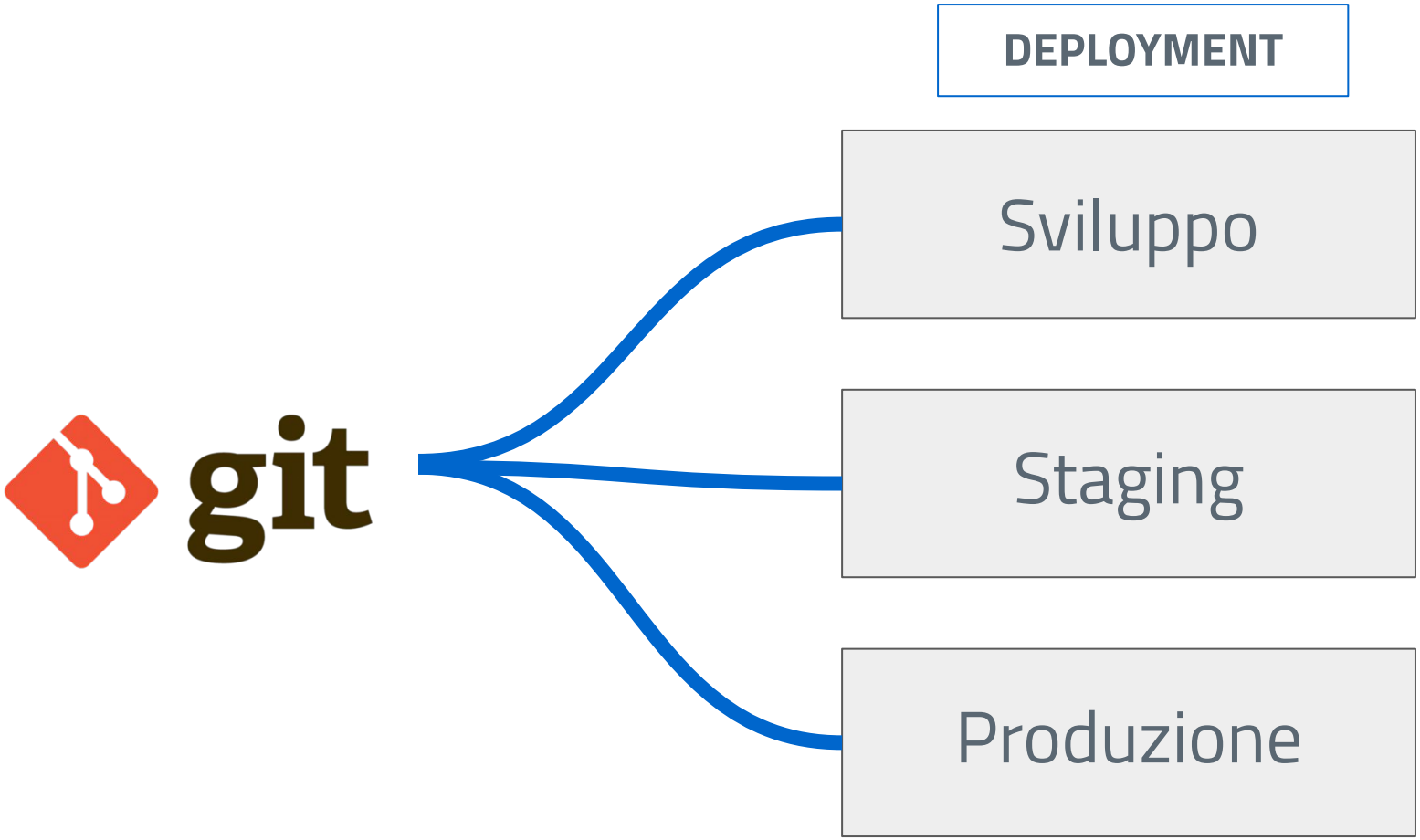
Una metodologia per sviluppare software cloud native

https://12factor.net/

Photo by Tanjir Ahmed Chowdhury on Unsplash

1. Codebase

Una codebase sotto controllo di versione, tanti deploy


DEPLOYMENT
Sviluppo
Staging
Produzione
git

2. Dipendenze

Dipendenze dichiarate e isolate

Photo by Mae Mu on Unsplash

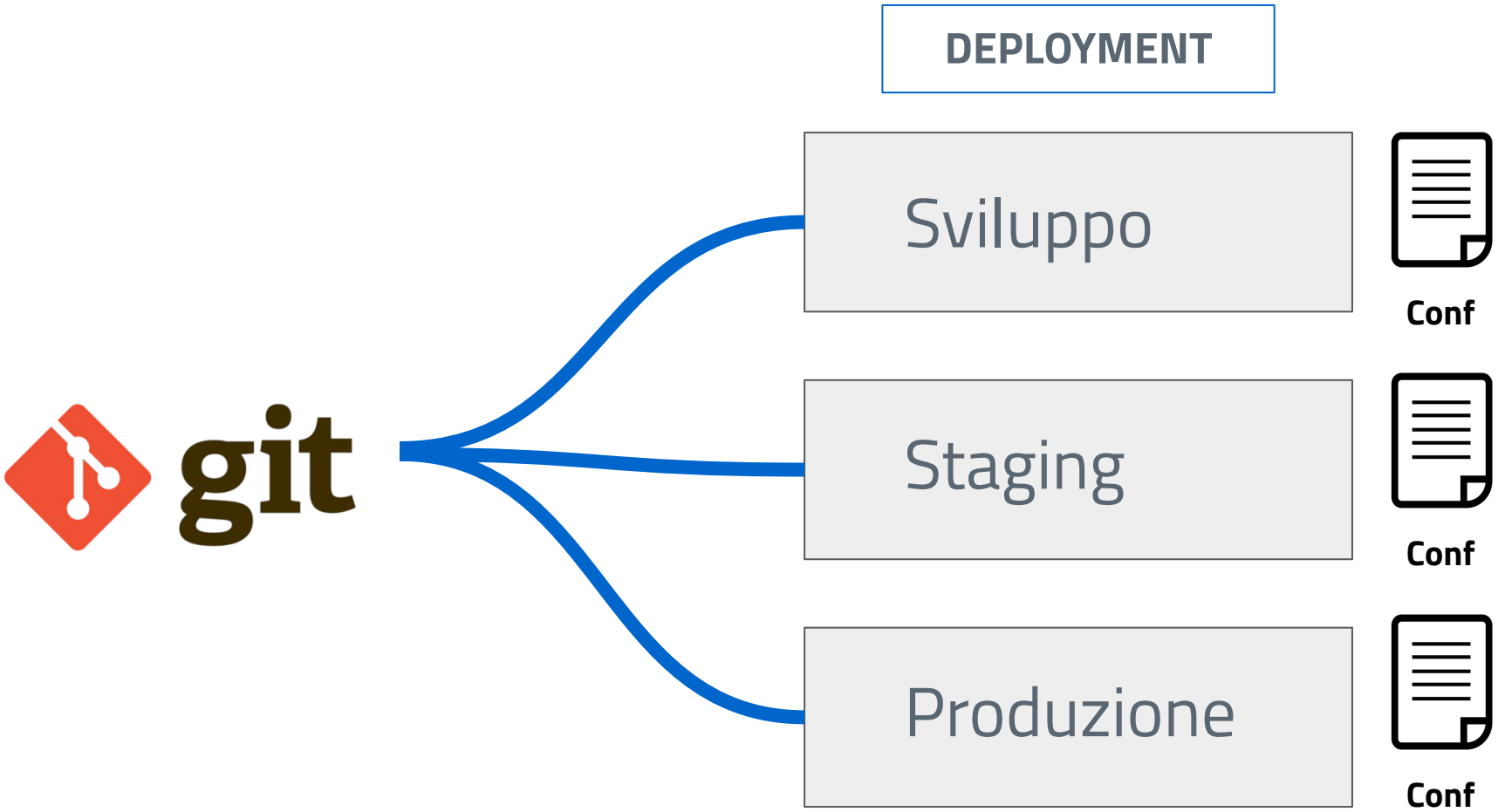
- requirements.txt (Python)
- package.json (npm)
- Dockerfile

Photo by Aaron Burden on Unsplash

3. Configurazione

Memorizza le informazioni di configurazione nell'ambiente

DIPARTIMENTO
PER LA TRASFORMAZIONE
DIGITALE


DEPLOYMENT
git
Sviluppo
Staging
Produzione
Conf
Conf
Conf

4. Backing service

Tratta i backing service come risorse

Photo by Gabriel Gusmao on Unsplash

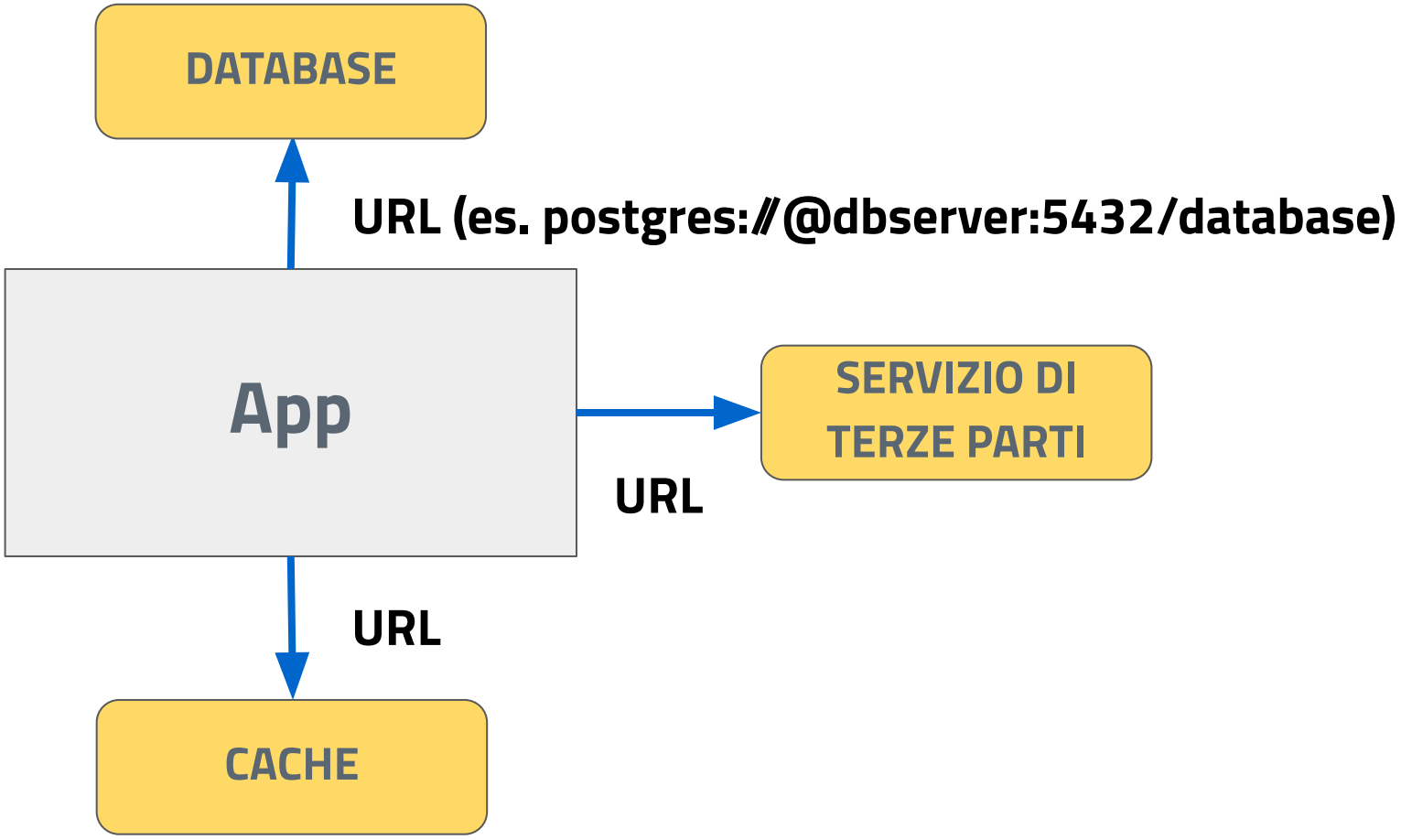
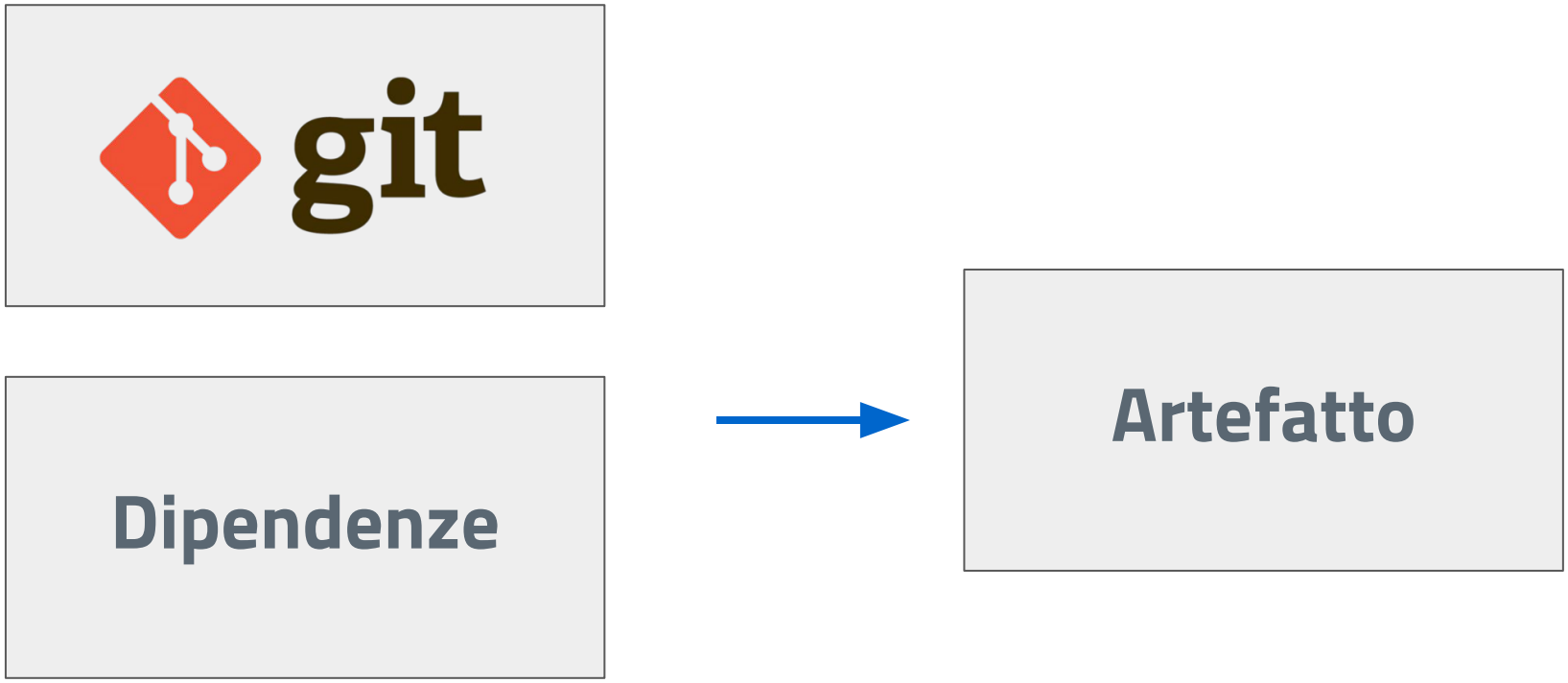

DATABASE
URL (es. postgres://@dbserver:5432/database)
App
SERVIZIO DI TERZE PARTI
URL
URL
CACHE

Photo by Stephanie Klepacki on Unsplash

5. Build, release, esecuzione

Separa in modo netto la fase di build dall'esecuzione

DIPARTIMENTO
PER LA TRASFORMAZIONE
DIGITALE



Build

Release

- Ogni release ha un id univoco di rilascio
- Qualsiasi modifica prevede una nuova release.

Esecuzione

6. Processi stateless

Esegui l'applicazione come uno o più processi stateless

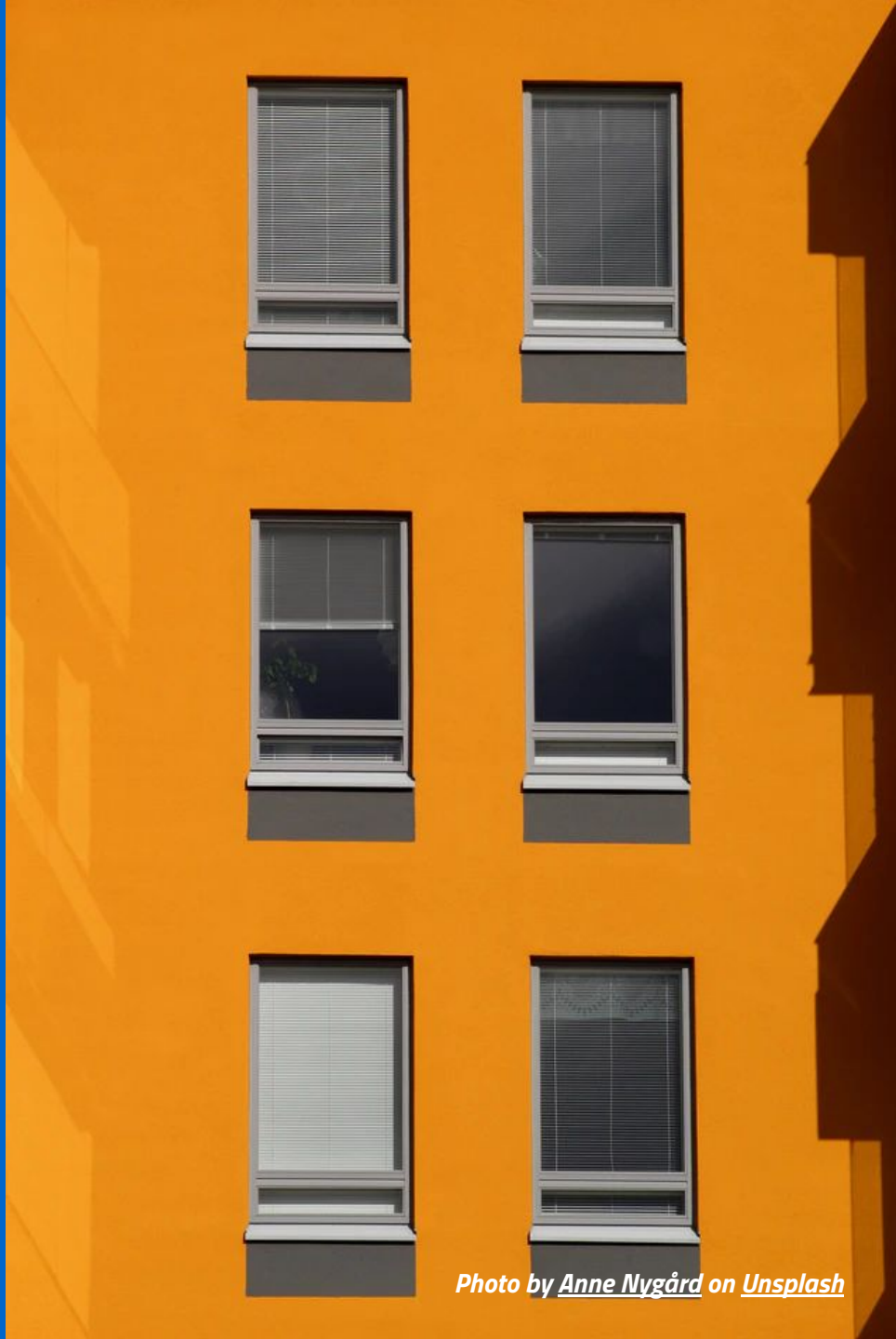


Photo by Anne Nygård on Unsplash

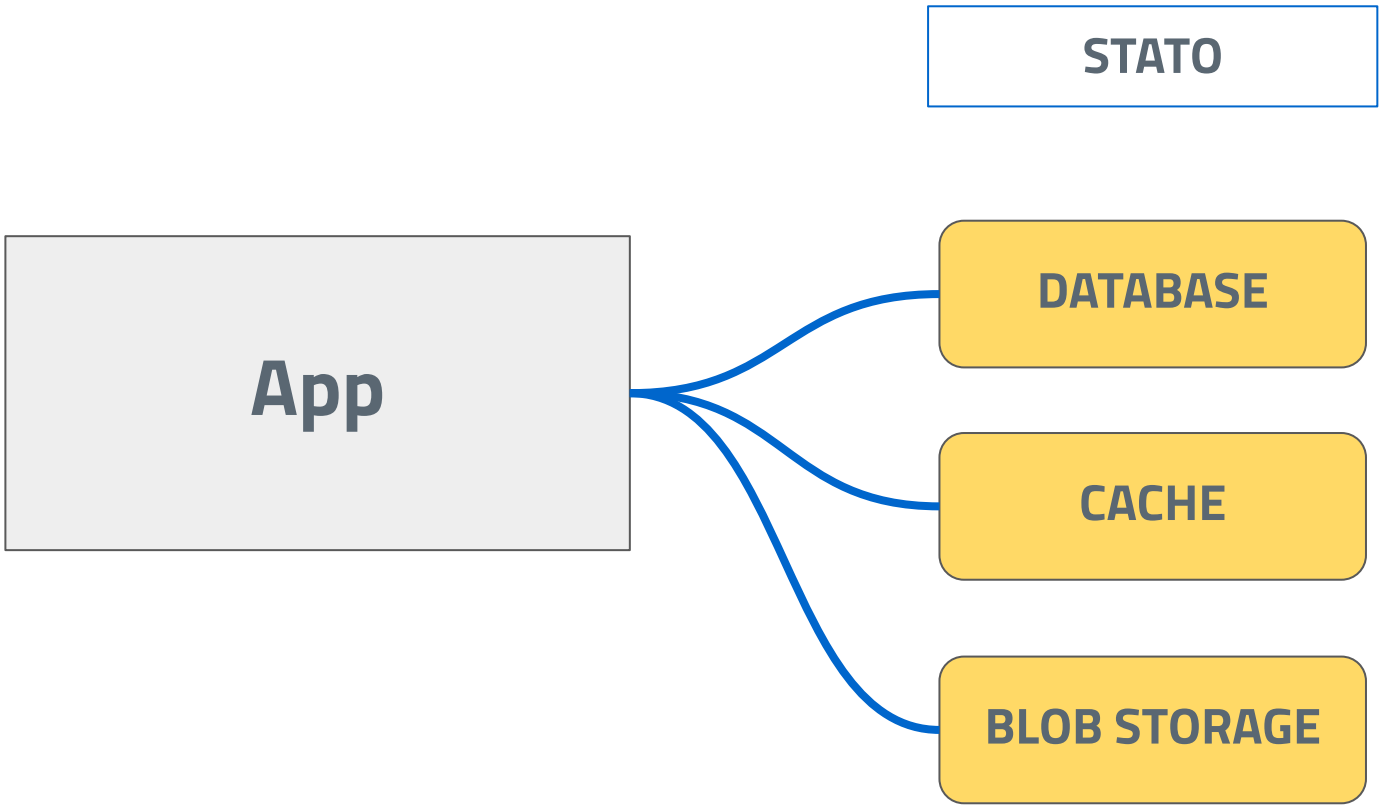
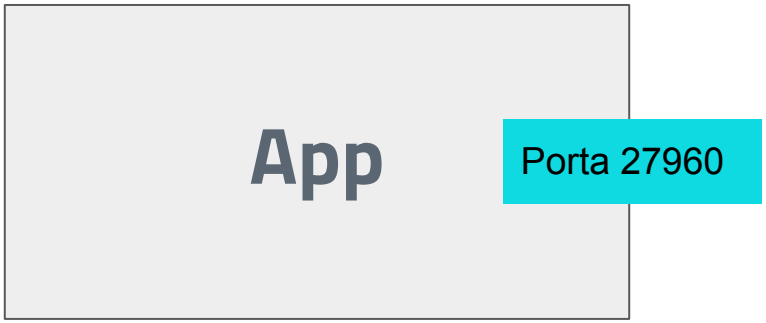

STATO
App
DATABASE
CACHE
BLOB STORAGE

Photo by Thom Milkovic on Unsplash

7. Binding delle porte

Esporta i servizi tramite binding delle porte

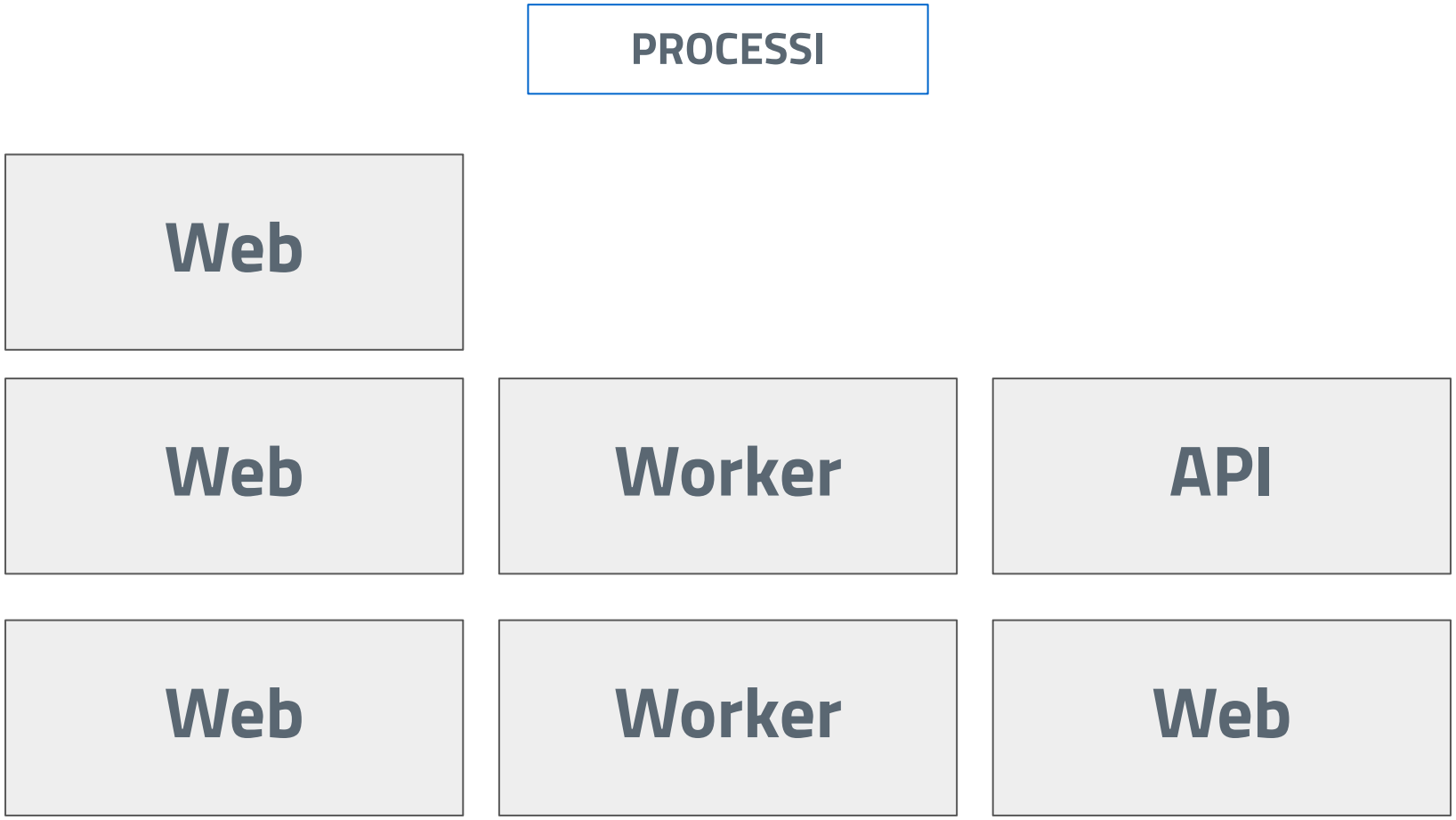
DIPARTIMENTO
PER LA TRASFORMAZIONE
DIGITALE


App
Porta 27960

- L’app è self-contained e non si affida a un altro servizio (es. web server)
- Può essere usata come backing service

8. Concorrenza

Scala attraverso il process model


PROCESSI
Web
Web
Worker
API
Web
Worker
Web

- La gestione dei processi è esterna all'applicazione
- Si può regolare il numero di processi a seconda della potenza dell'host
- Si può scalare a seconda del traffico

Photo by Paolo Chiabrando on Unsplash

9. Rilasciabilità

Massimizza la robustezza con avvii veloci e shutdown graduali

DIPARTIMENTO
PER LA TRASFORMAZIONE
DIGITALE

- Avvio e stop senza problemi
- Velocità di avvio
- Rimpiazzare processi in crash velocemente

10. Parità tra Sviluppo e Produzione

Mantieni lo sviluppo, staging e produzione simili il più possibile

Photo by Sharon McCutcheon on Unsplash

SVILUPPO

PRODUZIONE

App containerizzata

STAGING

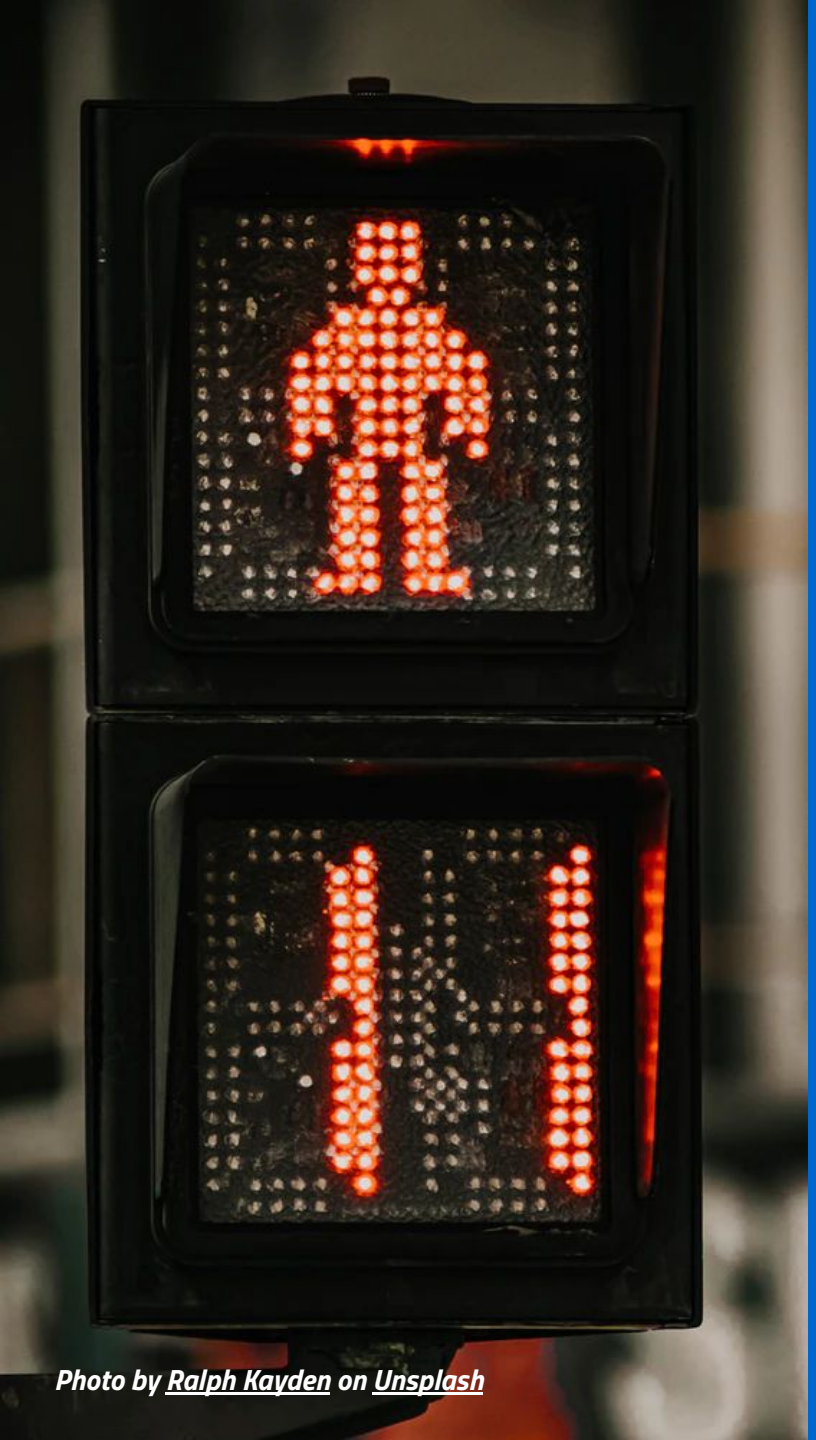


Photo by Ralph Kayden on Unsplash

11. Log

Tratta i log come stream di eventi

- La logica di logging è separata dall'applicazione
- L'applicazione scrive su stdout o stderr
- I log resistono ai crash dell'applicazione

12. Processi di amministrazione

Esegui i task di amministrazione come processi una tantum

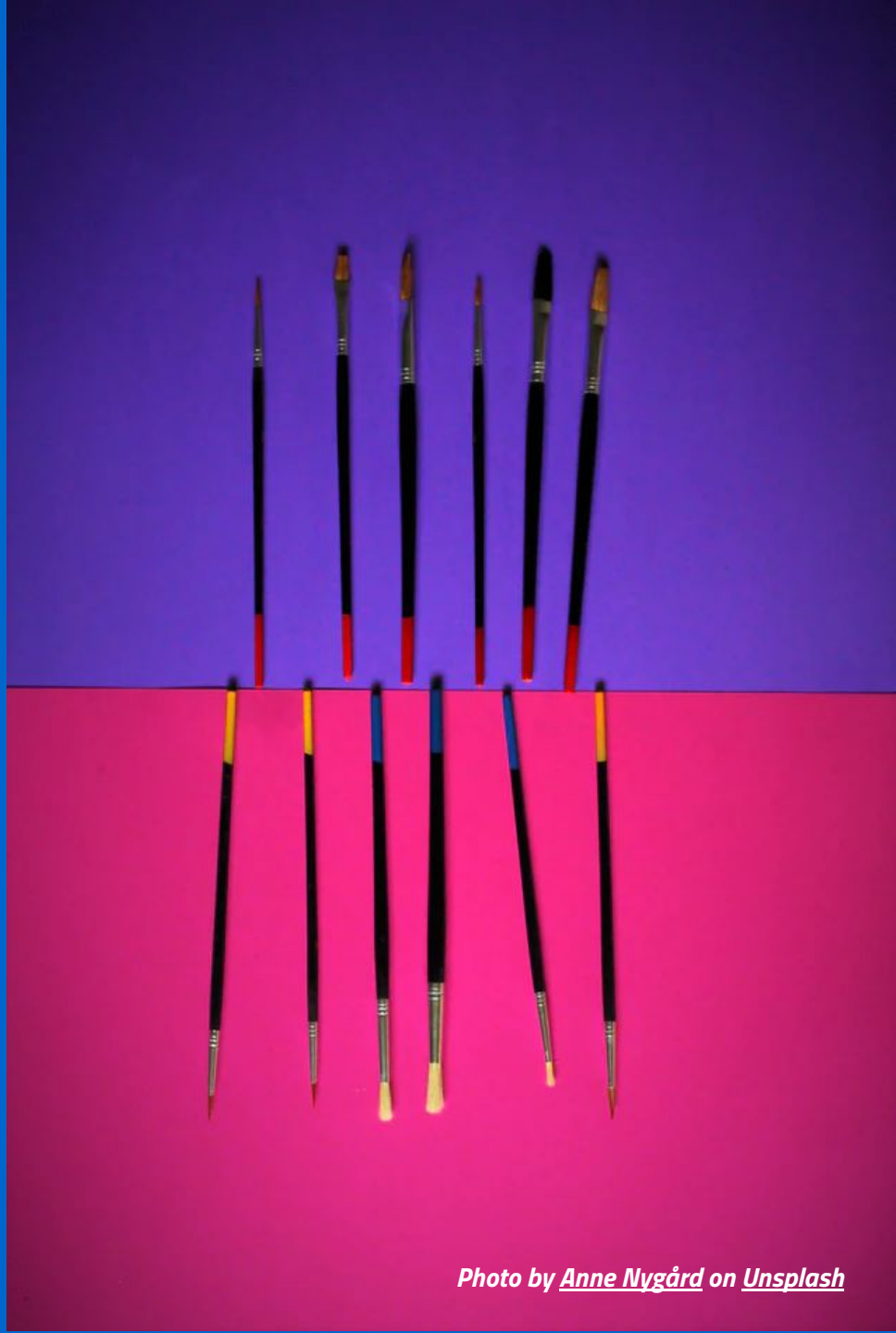


Photo by Anne Nygård on Unsplash

- Sono inclusi nella codebase
- Girano nello stesso ambiente dell’applicazione

Q&A

(ci sono domande?)

Designers

Developers

DEVELOPERS ITALIA

- **sito** developers.italia.it
- **mail** contatti@developers.italia.it
- **profilo** Twitter
- **chat** Slack Developers Italia

GRAZIE PER L'ATTENZIONE!

DIPARTIMENTO PER LA TRASFORMAZIONE DIGITALE

AgID

FormezPA