

Strumenti di gestione del ciclo di vita del *software*

Università degli studi di Padova
a.a. 2017/18

Laurea in Informatica

Corso di Ingegneria del Software

Presenta

Nicola Bertazzo

nicola.bertazzo@gmail.com

Giovedì 7 Dicembre 2017

> Sommario

1. Obiettivi
2. Processo
3. Esempio
4. Visione in dettaglio
5. Conclusioni

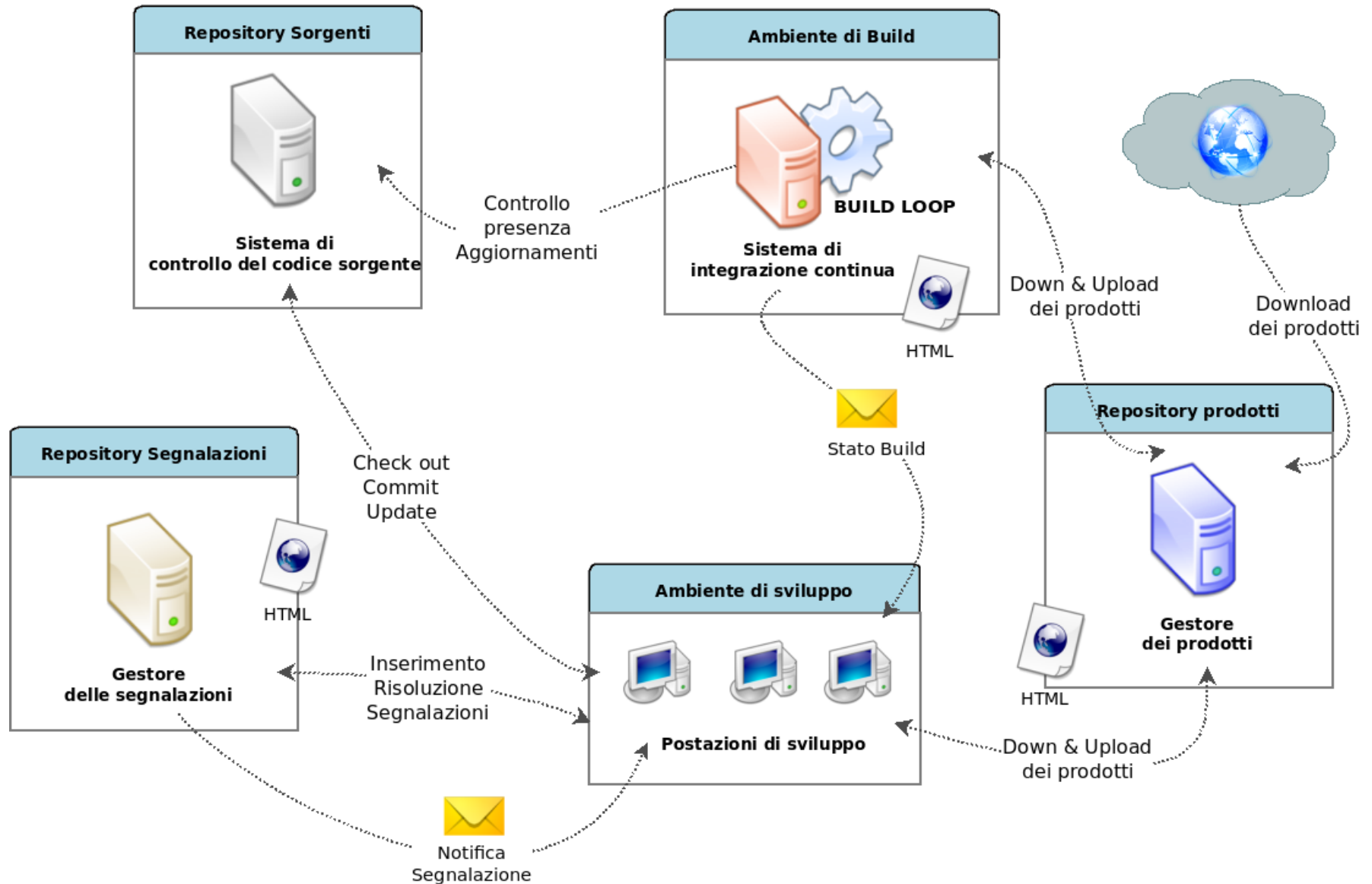
> Obiettivi

- Descrivere un processo di sviluppo e verifica del software **ripetibile**, con lo scopo di misurare e migliorare la qualità di prodotto
- Fornire un esempio dell'implementazione del processo, per lo sviluppo di un progetto **Java**, con prodotti *Free*

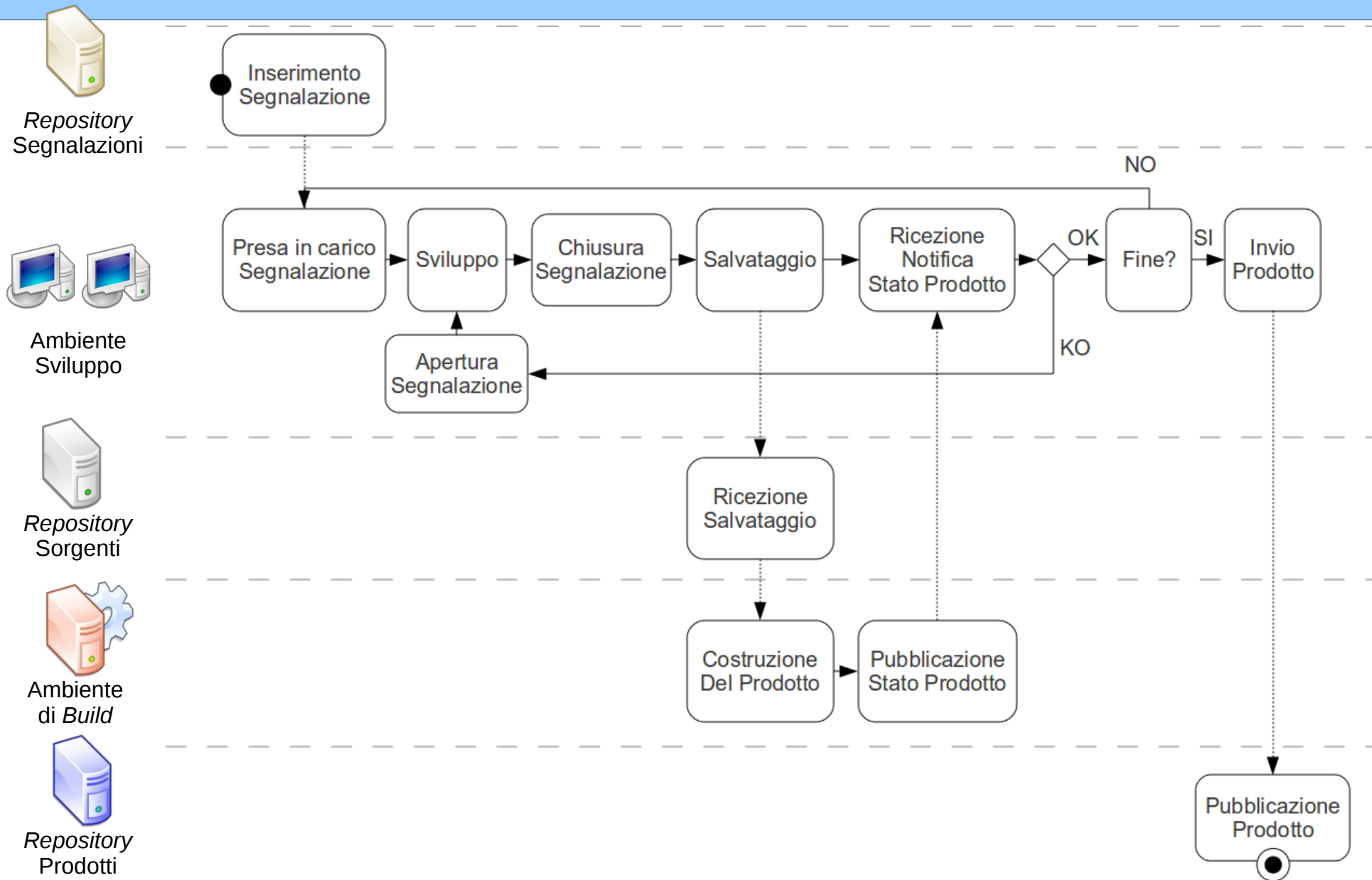


Processo di sviluppo

> Visione generale



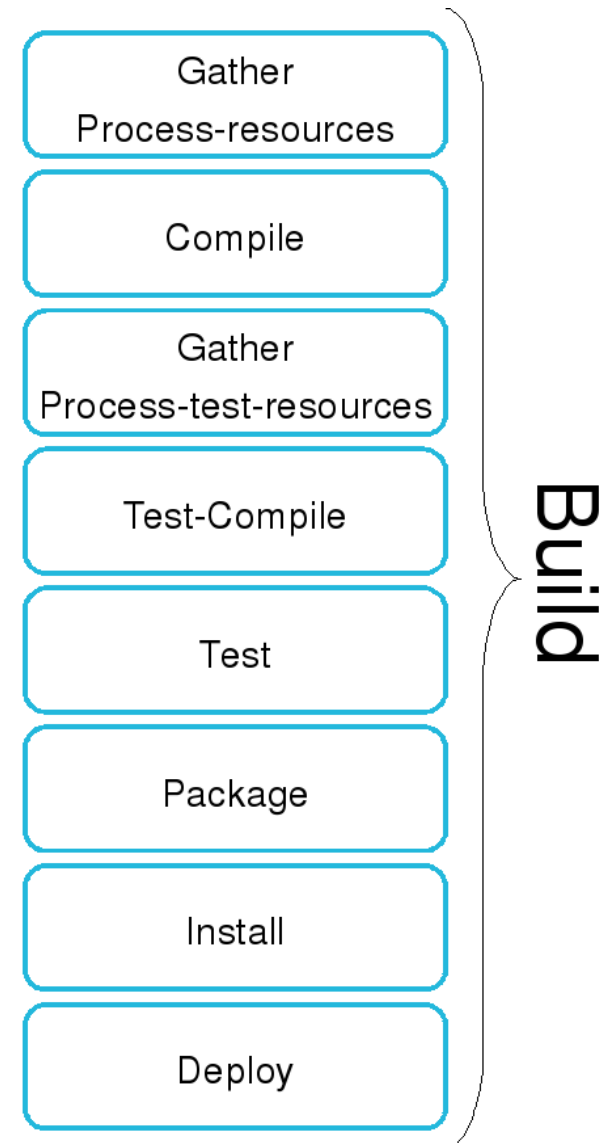
> Esempio



Visione in dettaglio

> Ambiente di sviluppo

- Luogo dove avviene l'attività di sviluppo e verifica del prodotto
- Vengono realizzati i test di unità
- Il processo di costruzione del prodotto deve essere automatico
- Il codice prodotto viene inviato frequentemente al *Repository* dei sorgenti
- Dopo la realizzazione di una funzionalità avvengono le altre attività di verifica



Ambiente di sviluppo

> Strumenti di verifica

Unità: Controllo del comportamento di ogni singolo oggetto in isolamento

- JUnit, TestNG [TDD]

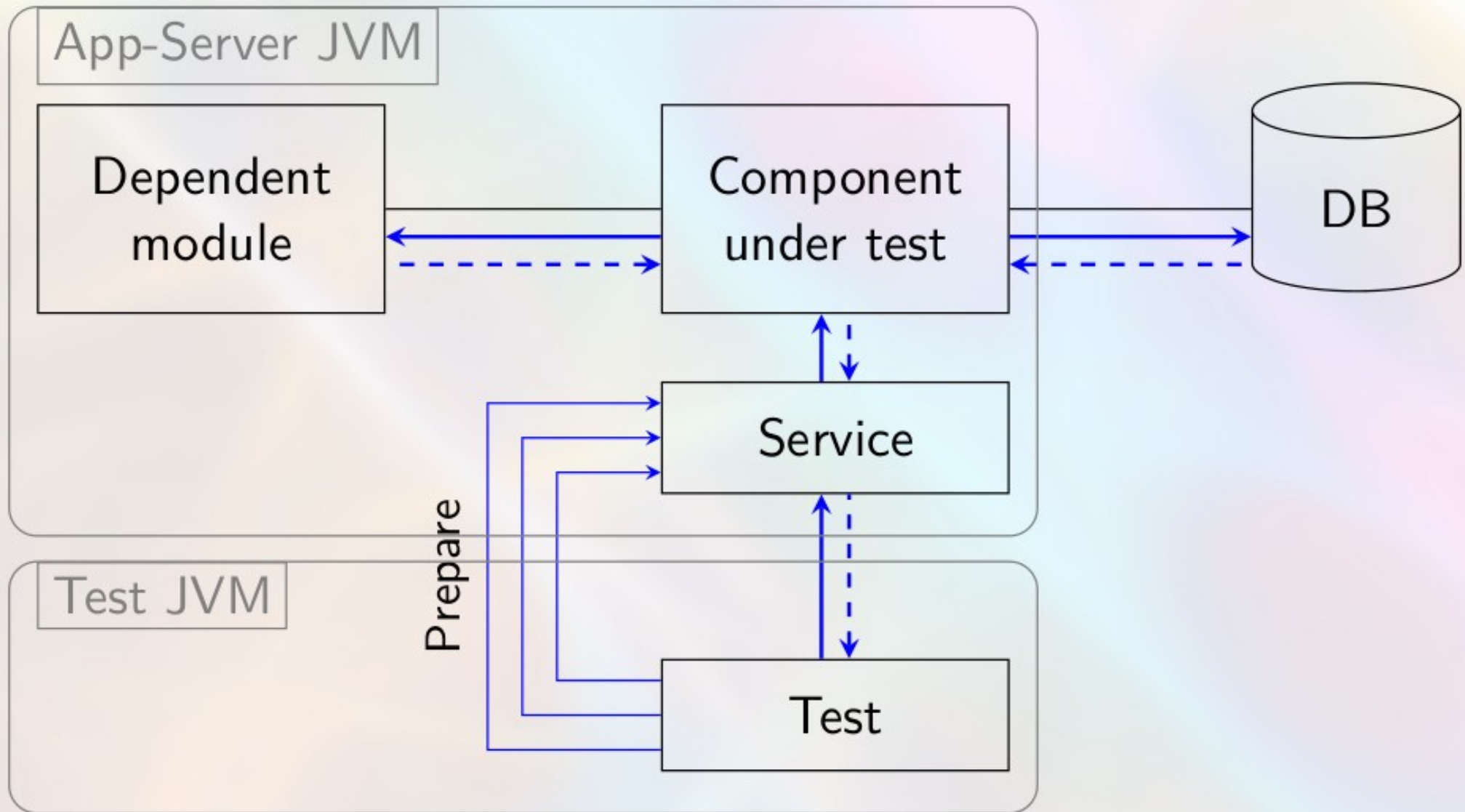
Integrazione: Verifica della collaborazione fra più oggetti nel formare un sottosistema, al fine di verificare il comportamento esterno e i contratti di interfaccia

Funzionali e verifica e validazione: Controllo del soddisfacimento dei requisiti funzionali

- Fitnesse, Selenium, Cucumber, Jbehave [BDD]

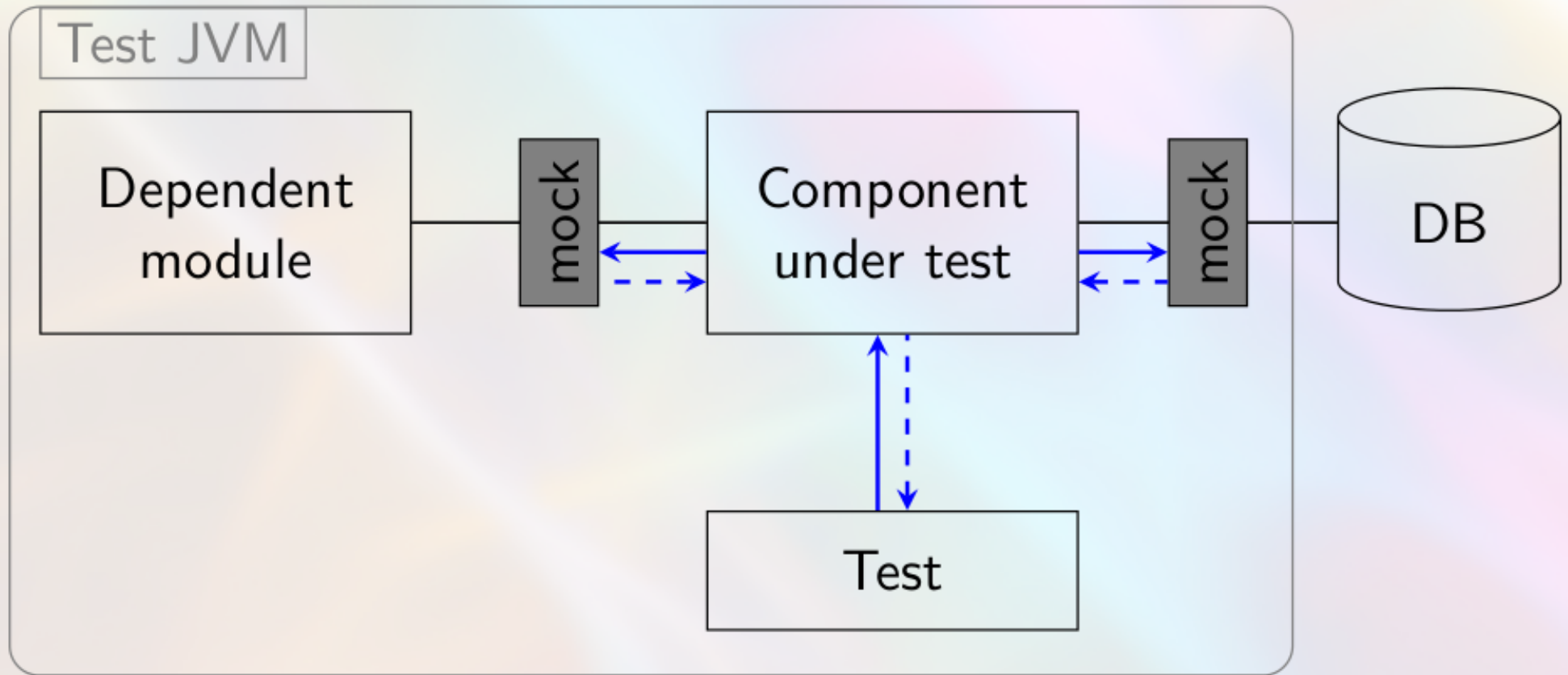
Ambiente di sviluppo

> Test funzionali



Ambiente di sviluppo

> Test di unità



Ambiente di sviluppo

> Strumenti di analisi del codice

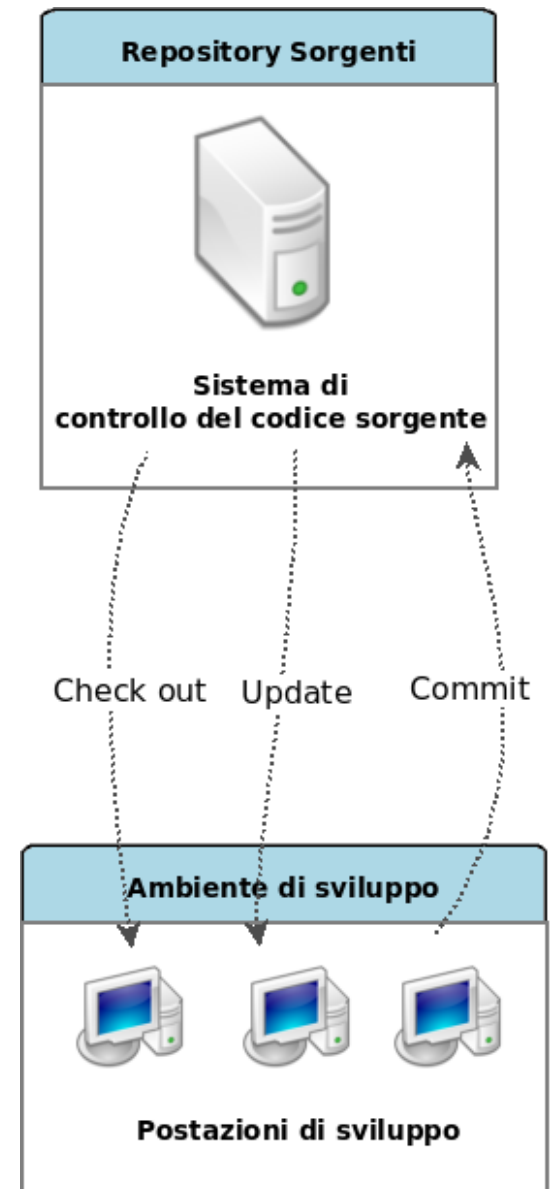
- Imporre il rispetto di convenzioni e stili
- Verificare la congruità della documentazione
- Controllare metriche ed indicatori (complessità ciclomatica, grafo delle dipendenze, numerosità delle linee di codice)
- Ricercare codice copiato in più punti
- Ricercare errori comuni nel codice
- Misurare la percentuale di codice testato
- Ricercare indicatori di parti incomplete (p. es. tag)



Visione in dettaglio

> *Repository* dei sorgenti

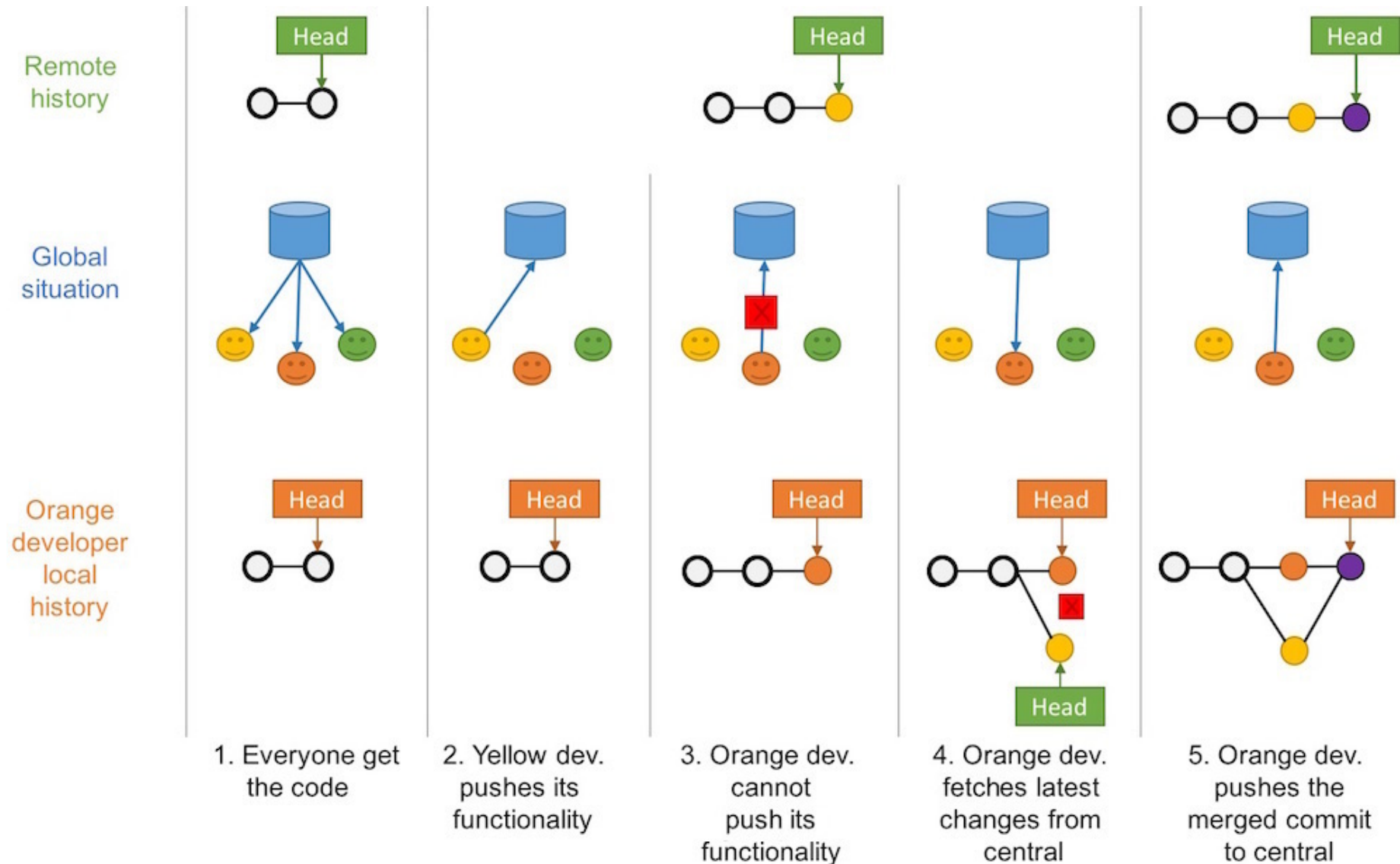
- Deposito unico di tutto il codice sorgente
- Accesso alla storia completa del software
- Documentazione delle modifiche:
 - Oggetti modificati
 - Data e ora
 - Autore
 - Commenti e motivazioni
- Possibilità di gestire più rami diversi di sviluppo contemporaneamente



<https://www.atlassian.com/git/tutorials/comparing-workflows>

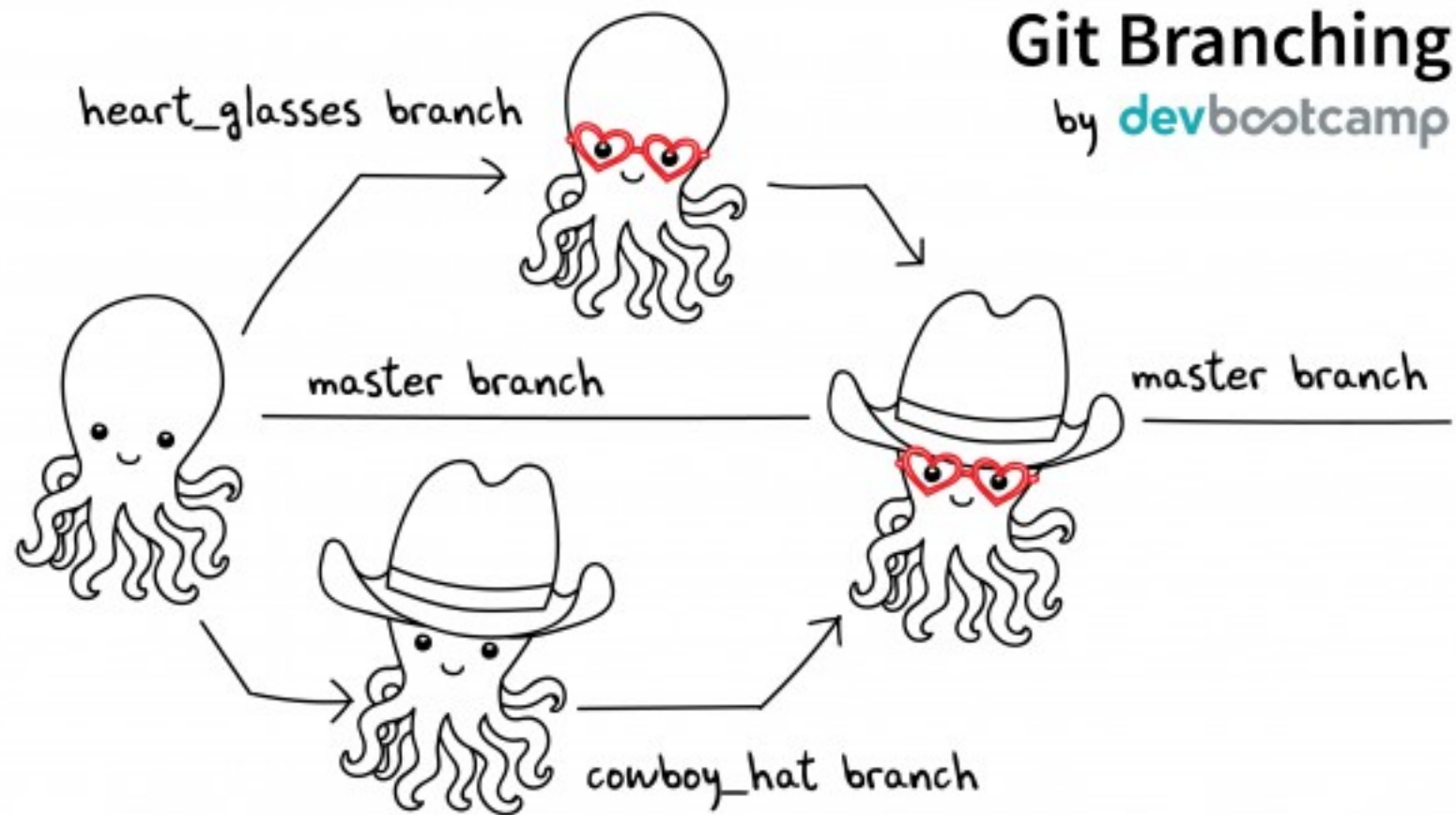
Repository dei sorgenti

> Workflow pattern: Centralized



Repository dei sorgenti

> Workflow pattern: Feature Branch

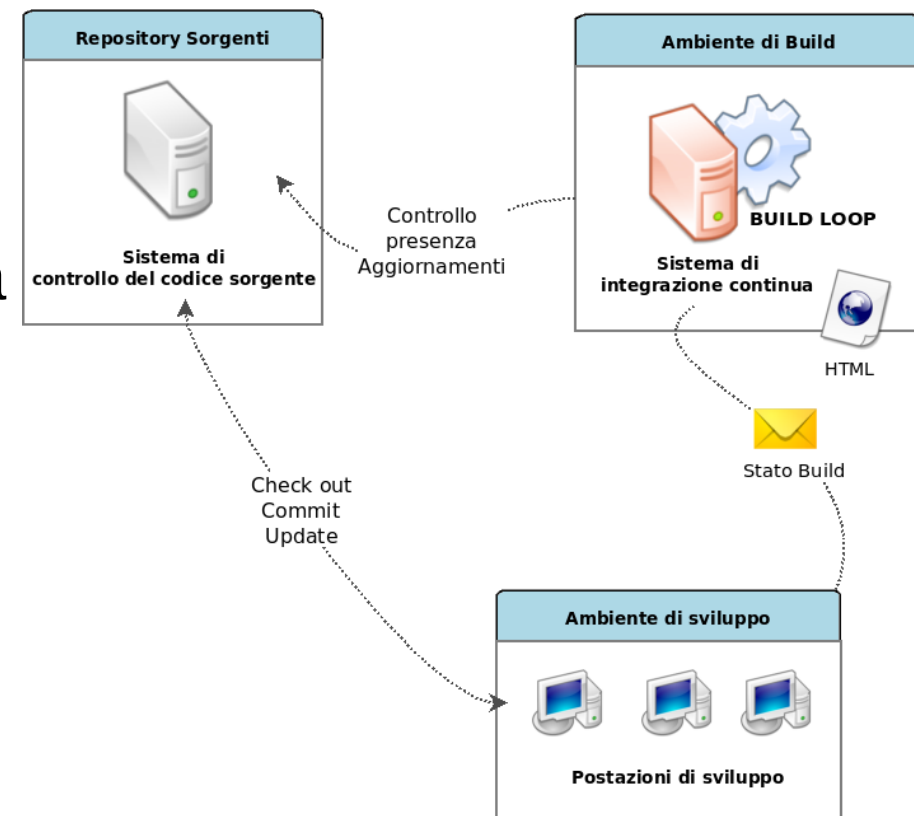


<https://www.atlassian.com/git/tutorials/comparing-workflows>

Visione in dettaglio

> Ambiente di *build*

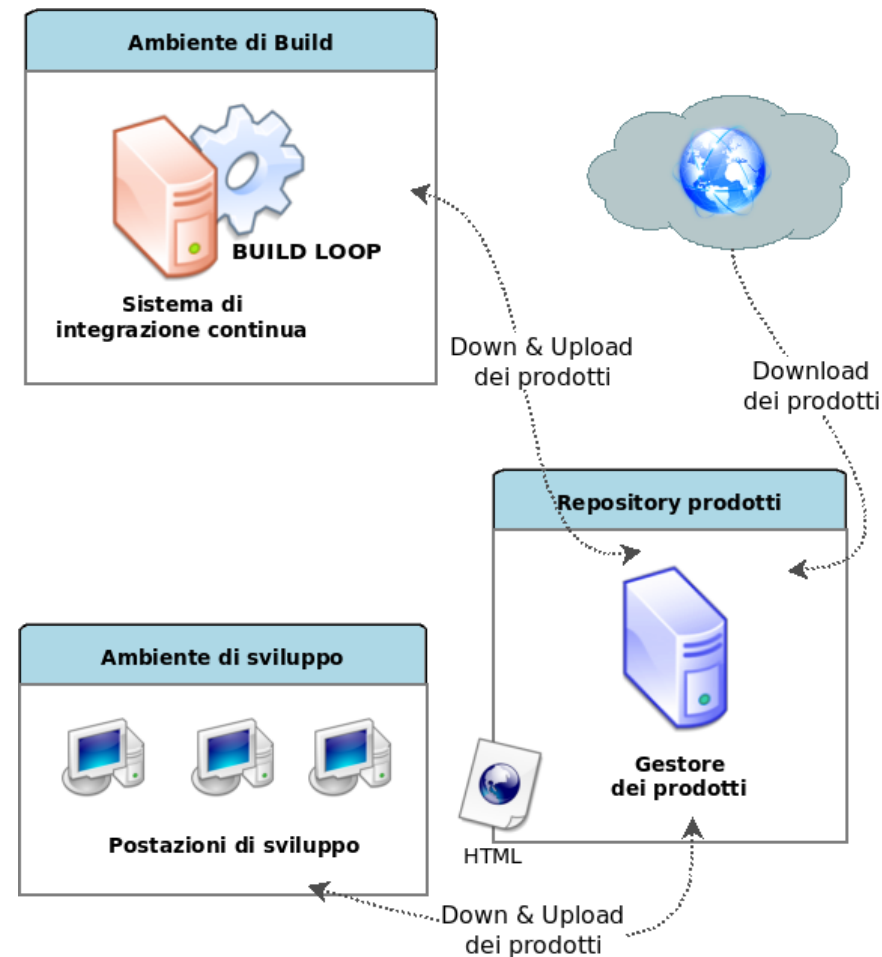
- Al completamento di un'attività viene costruito il prodotto
- Se il processo di costruzione fallisce l'attività non continua fino a che il prodotto non viene riparato
- Assicura la presenza di un prodotto consistente
- Il processo di costruzione deve essere automatico e ripetibile



Visione in dettaglio

> *Repository* prodotti

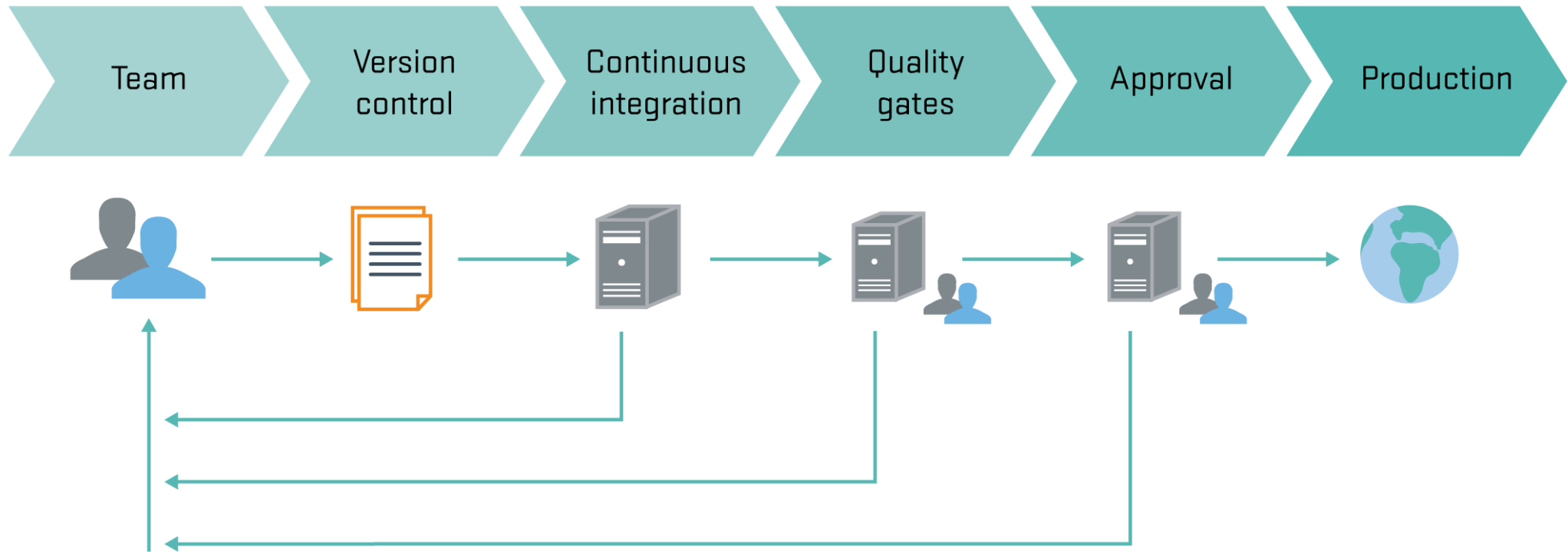
- Ambiente dove depositare e pubblicare i prodotti
- Agisce da intermediario per scaricare prodotti da depositi esterni
- Permette di effettuare ricerche e reperire informazioni riguardanti i prodotti
- Permette di gestire e associare permessi d'accesso sui prodotti



> Conclusioni

- Risultati garantiti nel lungo periodo
- Benefici incrementali
 - Si ottengono vantaggi da un'applicazione parziale o progressiva
- Tempo di avvio lungo
- Gestione complessa di molte componenti

> ... *Continuous Delivery*



> Continuous Delivery VS Deployment

CONTINUOUS DELIVERY



CONTINUOUS DEPLOYMENT



> Demo

