DRUPAL CONTINUOUS INTEGRATION

Parte I - Introduzione

Continuous Integration

La “Continuous Integration” è una pratica di sviluppo software nella quale i membri di un team **integrano il proprio lavoro di frequente**, spesso con cadenza almeno giornaliera, effettuando diverse integrazioni ogni giorno. **Ogni integrazione è verificata** da una build automatica e dai suoi test, in modo da **diagnosticare gli errori di integrazioni il più velocemente possibile**.

– Martin Fowler

Continuous Integration
BENEFICI

- automazione di task ripetitivi quali build, QA e deploy

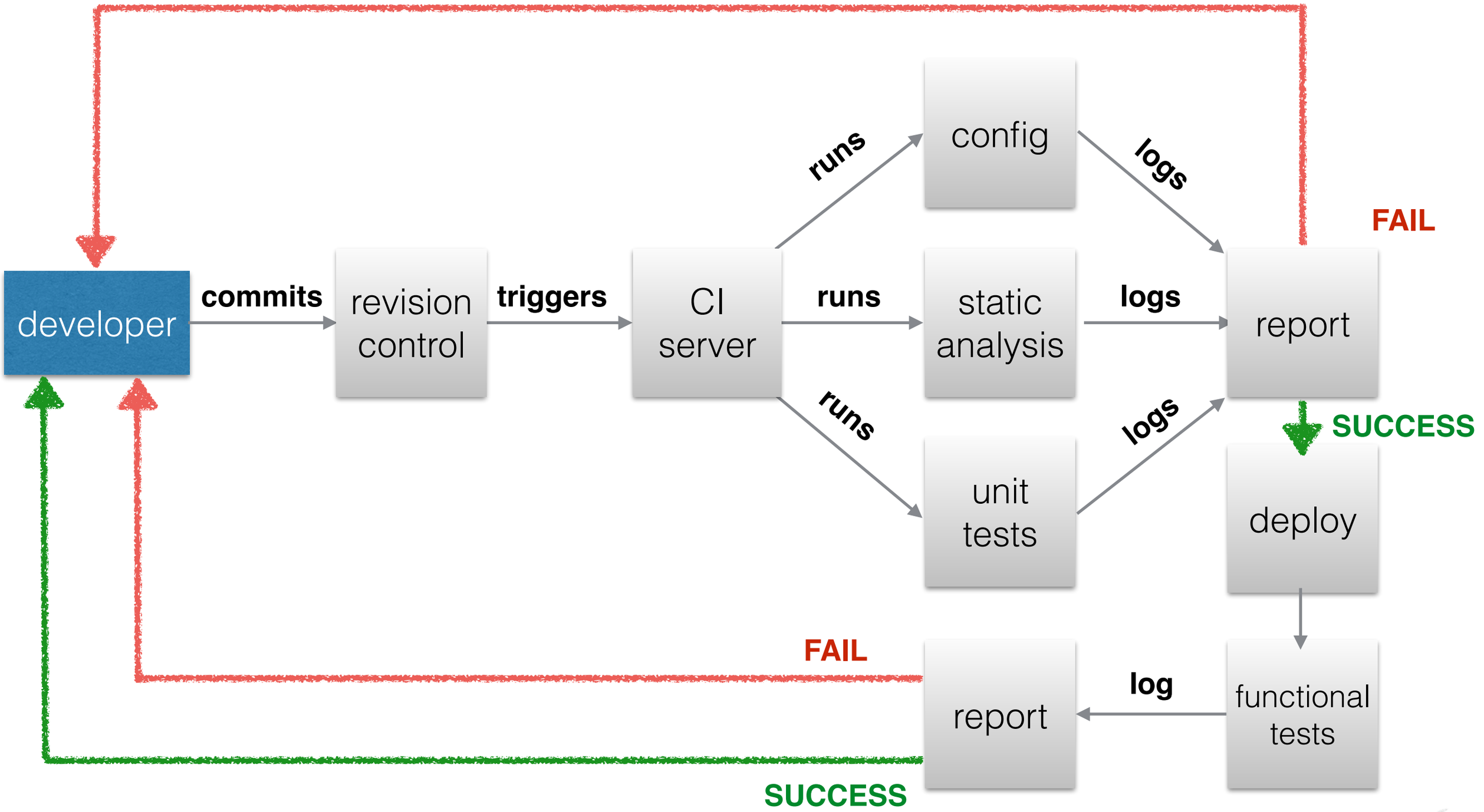
- integrazioni piccole e frequenti

- ciclo di feedback veloce

- deploy tracciati ed effettuabili in qualsiasi momento

Continuous Integration

PROCESSO DI BUILD



Continuous Integration
TOOLS

- SCM: git, svn, cvs, …
- CI SERVER: jenkins
- BUILD TOOL: phing
- DRUPAL AUTOMATION: drush
- DEPENDENCIES MANAGEMENT: drush make & composer
- DRUPAL CDD TOOLS: features & migrate
- TESTING: PHPUnit & Behat

Continuous Integration
DRUPAL - I

Contrariamente a quanto accade quando si utilizzano framework tradizionali, con Drupal è complicato e laborioso ottenere e mantenere il ciclo di Continuous Integration a causa dell'**alto accoppiamento tra configurazione e dati all'interno del database**.

Continuous Integration
DRUPAL - II

Per ovviare a questo problema, è necessario adottare una **metodologia di lavoro sistematica e disciplinata**, facendo ricorso a strumenti che permettano di **trasformare la configurazione in codice**, **trattare agevolmente i contenuti** ed **aggiornare in maniera incrementale lo schema della base dati**.

Continuous Integration
FLUSSO DI LAVORO - I

Il flusso di lavoro di uno sviluppatore consiste, oltre che nell'implementazione delle funzionalità richieste, anche nella **corretta, precisa e completa esportazione di tutti gli elementi di configurazione** necessari alla ricostruzione della funzionalità stessa.

Moduli chiave: **Features** e **Strongarm**.

Continuous Integration
FLUSSO DI LAVORO - II

È spesso necessario approntare contenuti predefiniti che possono variare durante lo sviluppo ed essere ampliati una volta che il sito viene pubblicato. È dunque necessario procedere all'**implementazione tramite codice di tutte le procedure di importazione**, al fine di poter gestire al meglio e replicare in qualsiasi momento ogni operazione.

Modulo chiave: **Migrate**.

Continuous Integration
FLUSSO DI LAVORO - III

Una volta terminata un'attività di sviluppo "atomica" quale un bugfix o la creazione di una nuova funzionalità, **è necessario creare un commit ed inviarlo al repository principale** in modo che possa essere integrato, testato e rilasciato in automatico sull'ambiente di sviluppo.

Strumenti chiave: **git/svn**.

Continuous Integration
FLUSSO DI LAVORO - IV

E’ buona pratica scrivere test automatici durante il processo di sviluppo al fine di **verificare che il lavoro effettuato sia funzionante** ed **assicurarsi che future modifiche** (specialmente se effettuate da altre persone) **non ne compromettano la validità**, creando regressioni e potenzialmente danni ai clienti o agli utenti.

Strumenti chiave: **PHPUnit**, **Behat**.

Q & A

Continuous Integration
ESERCIZI

1. Quali sono le ragioni principali per le quali è conveniente instaurare un sistema di Continuous Integration?

2. In quale punto del processo di build vengono eseguiti i test unitari ed in quale quelli funzionali? Per quale motivo?

3. Perché è importante scrivere test?

4. Quali benefici comporta l'utilizzo di un sistema di controllo di versione del codice?

5. Per quale motivo Drupal è particolarmente complesso da portare nel processo di Continuous Integration?

6. Quali sono gli strumenti che permettono di superare i limiti intrinseci di Drupal in modo da poterlo portare sotto Continuous Integration?