

Studio, analisi e progettazione di un prototipo di Infrastructure as Code

Relatore

Prof. Andrea Omicini

Presentata da

Alessandro Mazzoli

Sessione II

Anno Accademico 2014-2015

Background

- **Virtualization & Containerization**
- **Cloud Computing**
- **Configuration Management**

PaaS

Infrastruttura come agglomerato di software e dati

SaaS

Orchestration

Service Oriented Architecture

Infrastructure as Code(I)

- L'infrastruttura può e deve essere trattata come codice
- Gli sviluppatori che trattano l'infrastruttura dovrebbero conoscere le stesse metodologie e pratiche dei classici sviluppatori

Nuove tecnologie vecchie abitudini

- **script non idempotenti**
- **task manuali ripetitivi**
- **configurazione di servizi ad hoc**

Infrastructure as Code(2)

Infrastructure as Code è un approccio basato sulla Cloud Era nel costruire ed automatizzare una infrastruttura dinamica.

Tratta l'infrastruttura, i servizi e gli strumenti come parti di un sistema software.

Infrastruttura dinamica

- **riproducibile in ogni sua parte**
- **consistente**
- **ripetibile**
- **disponibile**
- **versionabile**
- **autodocumentabile**
- **flessibile ed incrementabile**

Prototipo di Infrastructure as Code

L'obiettivo della tesi consiste nell'applicazione dei principi cardine dell'Infrastructure as Code, di alcune delle metodologie legate allo sviluppo software e strumenti di version e di Configuration Management ad un progetto **Symfony** in corso.

Il prototipo si lega in parte all'approccio Lean minimizzando gli sprechi rilevati lungo il processo di produzione del software dalla fase di sviluppo fino al cliente finale riprendendo i concetti della Continuous Delivery.

Analisi del flusso di sviluppo software aziendale

Osservando il flusso di sviluppo del software interno è stato possibile sintetizzare un set di ambienti isolati: *development, testing, staging e production.*

Step di development

Ideato utilizza un template custom per lo sviluppo web combinando Vagrant ed Ansible.

Considerato Symfony quale PHP web framework,

l'ambiente sarà un classico stack LAMP

(Linux Apache2 MySQL PHP)

Step di testing(I)

La Continuous Integration è una pratica già adottata in Ideato, ad ogni modifica applicata al CVS, il server di Continuous Integration lancia una corrispettiva build.

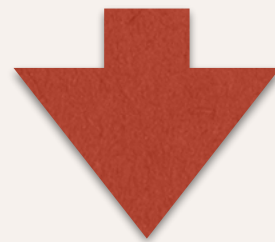
Step di testing(2)

Ideato presenta un'ampia gamma di tecnologie all'interno dei suoi progetti quindi è presumibile realizzare un sistema di

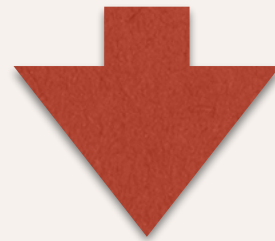
Continuous Integration non dedicato solamente al singolo tipo di progetto.

Step di testing(3)

Come eliminare la dipendenza dalle varie versioni dei componenti all'interno del server di CI?



CONTAINERIZATION



DOCKER



Transizione testing-staging-production

La Continuous Delivery che è parte integrante e alla base del prototipo di Infrastructure as Code, si concentra sulla riduzione degli sforzi per portare, da questo "deliver", il software in produzione.

Ridefinizione del template di sviluppo(I)

Durante lo step di development è stato analizzato lo stack Vagrant e il template, tuttavia quest'ultimo è adatto per ora solo ad un ambiente di sviluppo.

Ridefinizione del template di sviluppo(2)

Le tre tematiche riscontrate sono state:

- eterogeneità delle distribuzioni software
(Debian, RedHat)
- regole di security ed ottimizzazioni
- mantenere funzionante l'attuale template

Ridefinizione del template di sviluppo(3)

**La soluzione è stata sfruttare la
potenzialità del CVS per realizzare un
branch detto experimental o di testing
in cui fosse possibile applicare modifiche
senza compromettere la versione
corrente nel branch detta di master**

Eterogeneità delle distribuzioni

L'eterogeneità delle distribuzioni può essere gestita in due possibili modi:

- realizzare duplici Ansible roles, ognuno per ciascuna distribuzione
- strutturare i playbooks per gestire entrambe le distribuzioni

Il template avrà solo tre roles Ansible coerenti con lo stack LAMP :

- **ideato.common**
- **ideato.webserver**, role relativo a PHP e Apache2
- **ideato.database.mysql**, role relativo a MySQL

contenenti sia le specifiche per Debian che per RedHat

- name: include OS-specific variables

include_vars: "{{ ansible_os_family }}.yml"

-

name: CentOS Package requirements

yum: name="{{ ideato_common_packages }}" state=present update_cache=yes

when: ansible_distribution == 'CentOS'

-

name: Ubuntu Package requirements

apt: name="{{ ideato_common_packages }}" state=present update_cache=yes

when: ansible_distribution == 'Ubuntu'

—prod.yml—



facts



variables



conditionals



providers

Verso la deployment pipeline(I)

La separazione degli ambienti seguirà l'approccio suddividendoli in playbooks differenti: `dev.yml` e `prod.yml`.

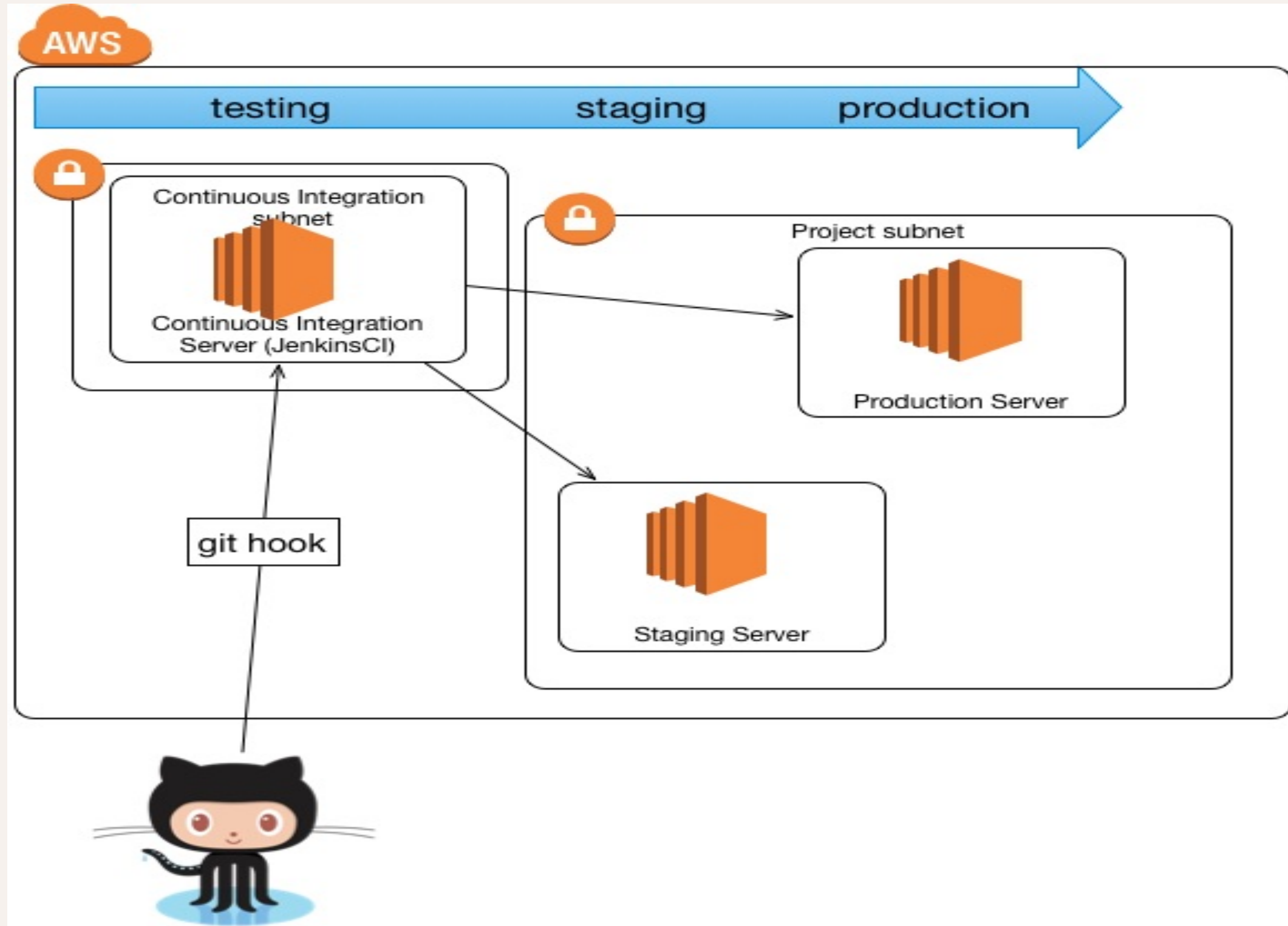
La scelta di non definire anche un playbook per l'ambiente di staging, è dovuta alla best practice di tendere verso ambienti production-like.

**Grazie alla ridefinizione sia del
playbook principale sia dei ruoli
relativi ai servizi, ora è possibile
effettuare il provisioning dei servizi
non solo verso l'ambiente di
sviluppo ma anche verso i successivi
ambienti, fondando le basi per futuri
sviluppi legati alla deployment
pipeline e alla Continuous Delivery.**

Deployment pipeline

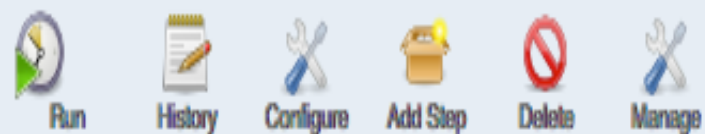
La seconda parte del protipo riguarda la realizzazione di una deployment pipeline fra i vari ambienti.

Dopo aver configurato il template, è stata realizzata un'ipotetica infrastruttura su **Amazon Web Services**.



Jenkins CI server & Build Pipeline Plugin

Build Pipeline: Project Pipeline



Considerazioni Finali

Il prototipo realizza pienamente i concetti della Continuous Delivery e della Lean Production, rendendo il deploy verso i vari ambienti un "non evento" attenuando sensibilmente i tempi di attesa e gli sprechi.

Il prototipo garantisce agli sviluppatori la facoltà di poter effettuare il setup degli ambienti o il deploy dell'applicazione in completa autonomia senza dover chiedere l'assistenza dal personale operativo.

Il prototipo tuttavia non rende possibile l'abolizione totale degli sprechi.

Il provisioning del sistema operativo, richiede tuttora l'assistenza del personale operativo.

Un futuro sviluppo sarà quello di integrare concetti quali di:

- **Dynamic Inventory**
- **Service Discovery**
- **TDD (Test Driven Development)**
- **TDI (Test Driven Infrastructure)**

Grazie per l'attenzione!