1 - Configuration Management

Il **Configuration Management (CM)** è un processo sistematico (*generale*) che garantisce la coerenza e la tracciabilità di tutti gli elementi di un sistema software durante l'intero ciclo di vita. Ti spiego come si collega al deployment.

Configuration Management: Concetti Fondamentali

Il CM ha **due obiettivi principali**:

- **Riproducibilità**: essere in grado di creare qualsiasi ambiente in modo completamente automatizzato, sapendo che ambienti diversi creati dalla stessa configurazione sono identici
- **Tracciabilità**: determinare rapidamente e precisamente le versioni di ogni dipendenza usata per creare un ambiente, e confrontare versioni diverse

I 4 Processi del CM Software

1. **Configuration Identification**: identificare gli attributi funzionali e fisici del software
2. **Configuration Control**: controllo sistematico delle modifiche
3. **Configuration Status Accounting**: tracciamento delle modifiche
4. **Configuration Audits**: verifica che il software rilasciato contenga tutte le funzionalità pianificate

Configuration Items (CI)

I **Configuration Items** sono unità di configurazione gestibili individualmente: computer, router, server, software. Il **Configuration Management Database (CMDB)** traccia tutti i CI e le loro relazioni (esempio: "il server A ospita il servizio B").

Collegamento con il Deployment

Il CM è **prerequisito essenziale** per Continuous Delivery perché:

1. Infrastructure as Code

- Le configurazioni degli ambienti sono gestite tramite codice (Chef, Puppet, Ansible)
- Stesso script di deployment per tutti gli ambienti (sviluppo, test, produzione)
- Separazione tra codice applicativo e configurazioni specifiche dell'ambiente

2. Principio "Deploy the Same Way to Every Environment"

- **Stesso processo** di deployment per sviluppo, test e produzione
- **Script versionate** nel VCS insieme al codice
- **Configurazioni separate** per ogni ambiente ma stesso meccanismo

3. Supporto alla Deployment Pipeline

Il CM garantisce che ogni stage della pipeline operi su ambienti consistenti:

- **Commit Stage**: build e test di unità
- **Acceptance Stage**: test funzionali in ambiente simile alla produzione
- **Production Stage**: deployment automatico con stessa configurazione testata

4. "Deploy into a Copy of Production"

Per garantire successo del deployment, l'ambiente di test deve avere:

- Stessa configurazione di rete
- Stesso sistema operativo
- Stesso stack applicativo
- Dati in stato consistente

Strumenti Modern CM

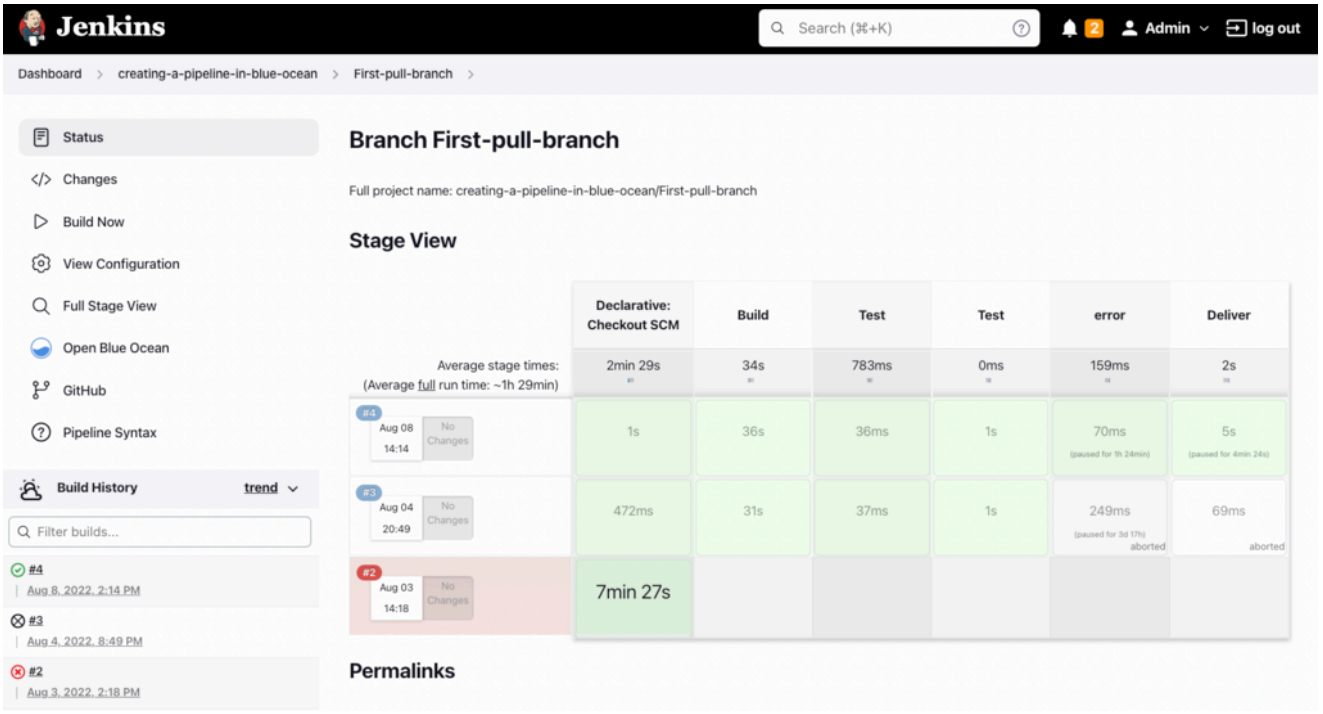
Tradizionali: Chef, Puppet per configurazione server **Cloud**: AWS CloudFormation, Terraform per Infrastructure as Code **Container**: Docker, Kubernetes per gestione applicazioni

Benefici per il Deployment

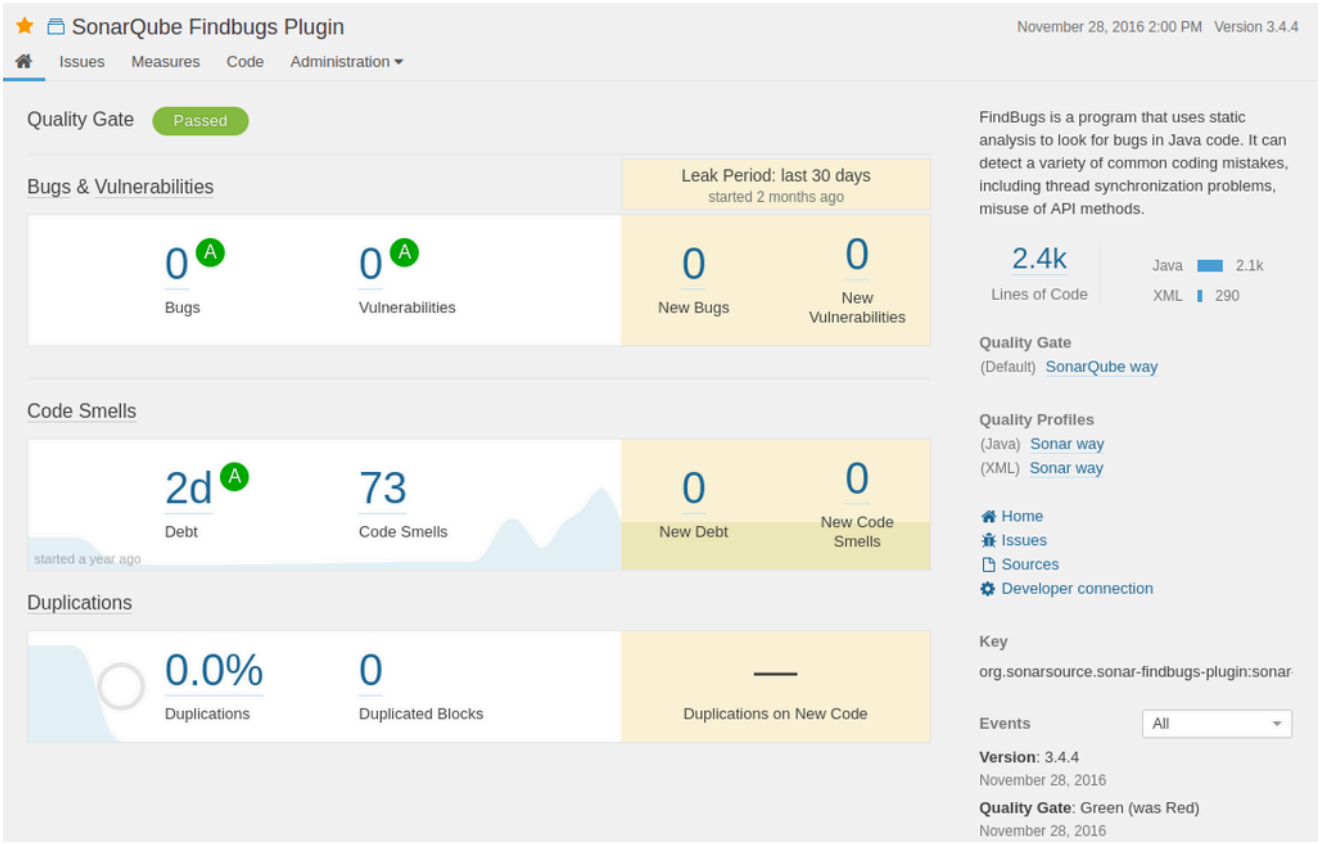
- **Controllo accurato** dell'infrastruttura IT
- **Maggiore affidabilità** dei rilasci
- **Debugging facilitato** in caso di problemi
- **Compliance** e tracciabilità delle modifiche
- **Rollback rapido** a configurazioni precedenti

Il CM trasforma la gestione dell'infrastruttura da processo manuale e soggetto a errori in un processo automatizzato, versionate e riproducibile, rendendo possibile la Continuous Delivery.

2. Jenkins - Esempio di CI



3 - SonarQube - Analisi statica

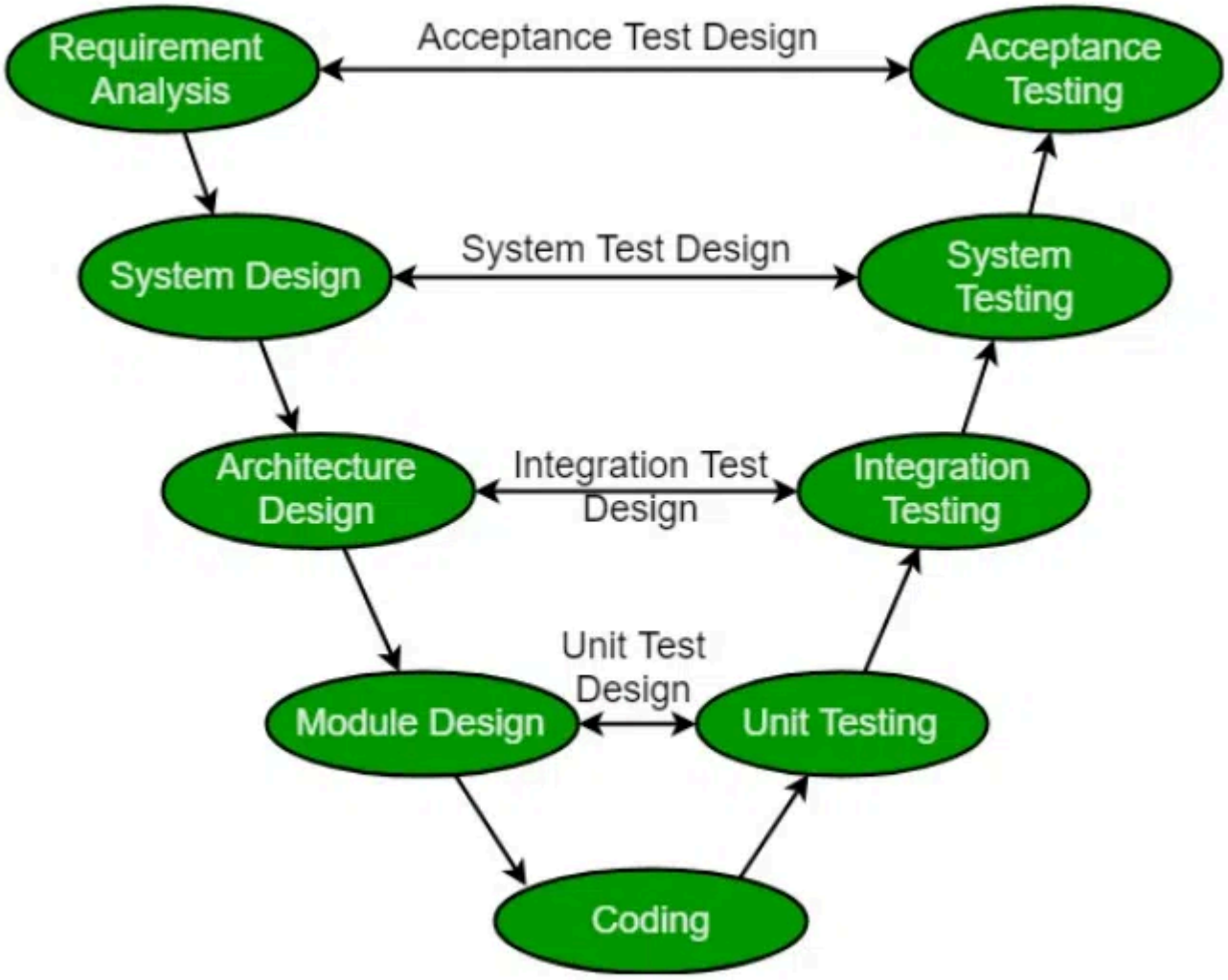


4 - A-TRIP

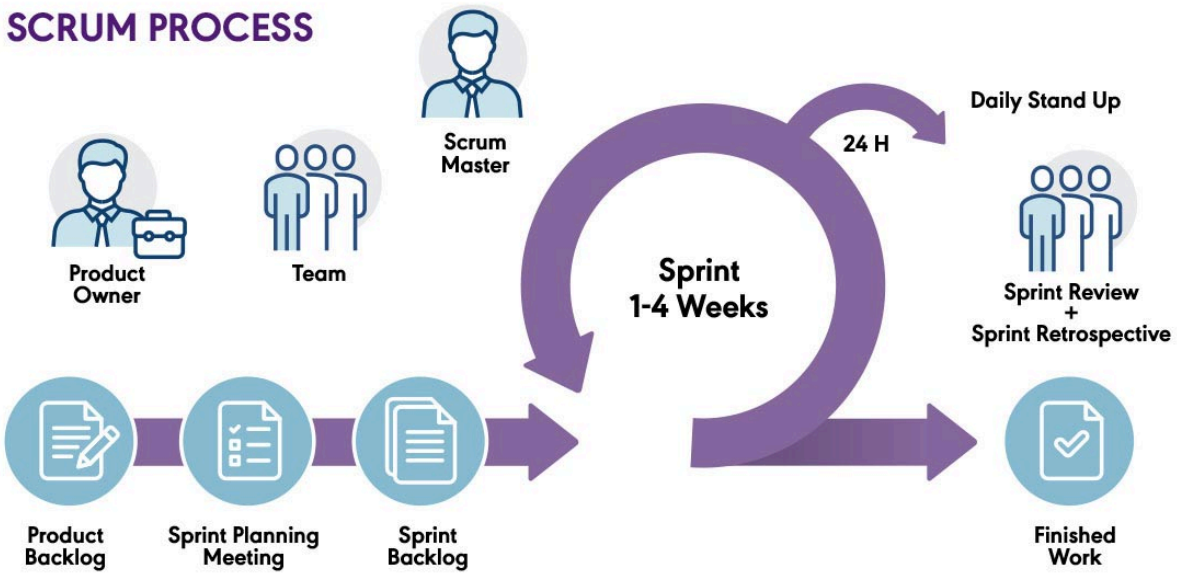
- Automatic = A
- Thorough = T = Accurato
- R = Repeatable
- I = Independent
- P = Professional

SUT = System Under Test

5 - V-Model



5 - Scrum (Mischia)



Storico nello Scrum - Backlog

- Definition of Done = Tutto fatto rispetto alle *user story*
- Acceptance Criteria = Test di accettazione (collaudo)

Bonus

Ripasso (per alcune parti):

- https://ale958.github.io/Sweky/Introduzione.html