L'importanza di Continuous Integration e Continuous Deployment

di M. Valentino, Solutions Architect @ Corley - whitepaper 2020

Viviamo in un contesto storico in cui la tecnologia la fa da padrona e siamo ampiamente circondati da prodotti e servizi digitali.

La costante evoluzione e l'aumento della richiesta di prodotti e servizi hanno evidenziato sempre più la necessità di **nuovi modelli di sviluppo**, favorendo tante rivoluzioni informatiche e culturali come ad esempio la metodologia Agile o la nascita di figure professionali trasversali su diverse aree di sviluppo.

Al tempo stesso si è intensificata la necessità di nuove tecniche e nuovi strumenti di supporto per aiutare gli sviluppatori a **testare, integrare e distribuire** i loro prodotti in maniera più semplice e veloce.

Approcciarsi alla metodologia CI/CD consente agli sviluppatori di superare i normali problemi di integrazione e rilascio di aggiornamenti software

La CI/CD è un processo definito nato proprio in questo scenario di grande evoluzione e, al tempo stesso, grande richiesta di strumenti evoluti.

Che cos'è la CI/CD?

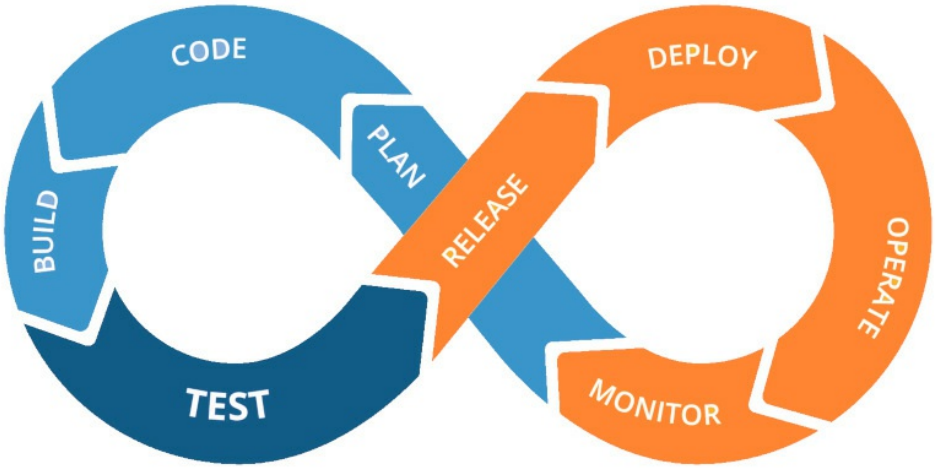
Per CI/CD, ovvero *Continuous Integration* e *Continuous Deployment*, si intende una metodologia per la distribuzione continua di prodotti e/o servizi mediante l'utilizzo di processi automatizzati e basata sui concetti di integrazione, distribuzione e deployment.

Approcciarsi alla metodologia CI/CD consente agli sviluppatori di superare i normali problemi di integrazione del codice che avvengono tipicamente negli scenari in cui lo sviluppo procede per vie parallele, e consente di distribuirne il risultato automaticamente.

Grazie ad essa è possibile concentrare le risorse disponibili principalmente sullo sviluppo del codice accelerando l'introduzione di nuove funzionalità e rendendo i cicli di rilascio del software più celeri e soprattutto automatizzati.

I significati dietro l'acronimo CI/CD

Le sigle CI e CD sono due acronimi dai vari significati. Tendenzialmente quando si parla di CI/CD e ci si riferisce all'acronimo CI si intende *Continuous Integration*, ovvero integrazione continua.



Essa è la parte del processo dedicata principalmente agli sviluppatori del codice e si occupa di effettuare l'integrazione, il test unitario e l'unificazione del codice all'interno di una codebase condivisa in modo automatico al fine di evidenziare o meno la corretta integrazione del nuovo codice nel repository.

L'acronimo CD ha invece un doppio significato ovvero **Continuous Delivery e/o Continuous Deployment** che indicano rispettivamente la distribuzione continua o il deployment continuo (inteso come rilascio in produzione).

Nel processo di distribuzione continua, in genere a seguito del processo di CI, avviene un processo di test di integrazione automatico del software al fine di ricercare eventuali bug ancora presenti e non manifestati nei test unitari e la produzione di un **artefatto per il rilascio**.

L'artefatto può assumere più forme come ad esempio uno o più repository appositi, un pacchetto di installazione o anche un semplice ZIP ma rappresenterà sempre il prodotto che sarà destinato al cliente finale.

Infine, nella distribuzione continua, il rilascio al cliente dell'artefatto prodotto è un'operazione da compiere manualmente ad opera di un team operativo e non deve essere automatizzato se non a seguito di un intervento manuale.

Il processo di deployment continuo differisce dal processo di distribuzione continua proprio per quest'ultimo passaggio poiché l'artefatto viene distribuito al cliente finale con un **processo automatizzato** rendendo non più necessario l'intervento manuale.

Abbiamo quindi un processo totalmente automatico dall'invio di nuovo codice da parte dello sviluppatore al rilascio di una nuova versione del software.

Quando conviene usare la CI/CD?

Come abbiamo precedentemente illustrato la metodologia di CI/CD si applica molto bene a contesti in cui vi è uno sviluppo continuativo di nuovo codice: ha quindi poco senso implementare questo processo in contesti privi di attività come ad esempio nel software legacy.

In uno scenario tipico, se adottata fin dall'inizio dello sviluppo, può risultare uno strumento estremamente utile e versatile per consolidare varie tematiche come il controllo del codice prodotto, il test, QA ed il monitoraggio dei cicli di rilascio sia applicativo che documentale.

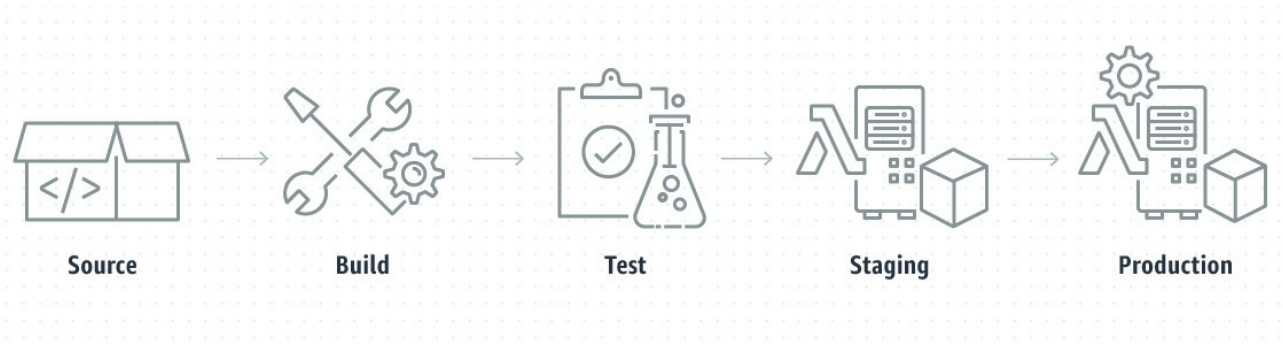
Ciò, come già citato, permetterà di **concentrarsi maggiormente sulla produzione di nuovo codice** anziché sul rilascio di esso massimizzando l'utilizzo delle risorse.

Quali sono i requisiti per utilizzare CI/CD?

Essendo la metodologia CI/CD un processo in cui si automatizzano le normali operazioni manuali è indispensabile costruire dei processi sia logici che infrastrutturali in cui ciò possa avvenire in modalità automatica.

Quasi sempre ciò prevede la costruzione di script in vari linguaggi come ad esempio bash o python che procedono ad effettuare tutte quelle operazioni che tipicamente verrebbero impartite manualmente da terminale. L'unica particolarità da tenere a mente in questo contesto è che essendo procedure automatiche non vi devono essere interrogazioni verso l'utente come messaggi di conferma o l'inserimento manuale di parametri.

I singoli script devono quindi essere in grado di **procedere in autonomia**.



Un altro aspetto da tenere a mente riguarda l'ambito infrastrutturale: i processi automatici devono avere i permessi e la capacità di pubblicare e/o effettuare rilasci sui vari ambienti in maniera sicura e controllata notificando eventuali anomalie o errori via mail, messaggi su chat e/o notifiche sugli smartphone al fine di rendere tempestivo l'intervento diretto del team competente.

Qualche scenario On Premise e Cloud Based

Ad oggi esistono moltissimi software e servizi online dedicati al mondo della CI/CD, tra essi spiccano nomi come **Jenkins**, **Travis CI**, **GitHub Actions** o i [vari servizi AWS (https://aws.amazon.com/it/getting-started/projects/set-up-ci-cd-pipeline/)](https://aws.amazon.com/it/getting-started/projects/set-up-ci-cd-pipeline/) come CodeBuild/CodePipeline.

La principale distinzione consiste nella scelta tra prodotti on premise o servizi managed.

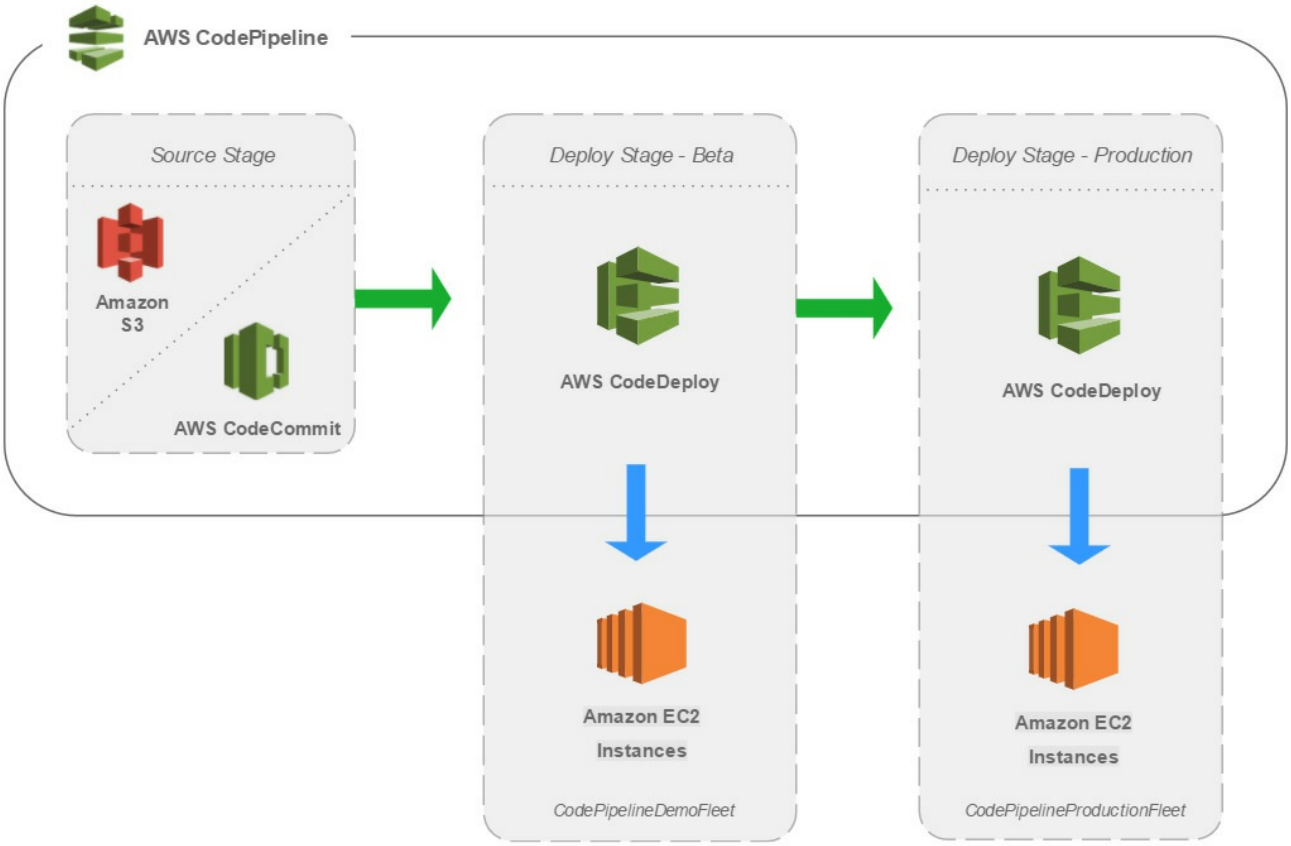
Nella famiglia dei prodotti *on premise* la scelta ricade quasi sempre su Jenkins grazie ai suoi tanti anni di attività, l'alto numero di plugin ed ad una community ampia ed attiva.

Discorso diverso invece per quanto riguarda l'ambito dei servizi managed, in quanto nel corso degli anni si stanno diffondendo un'alternanza di servizi gestiti che si differenziano per pochi punti chiave o per modelli economici differenti.

Tra essi non è possibile ignorare AWS con la suite di servizi della *famiglia Code*:

- CodeCommit: per la gestione di repository git
- CodeBuild: per la costruzione di artefatti
- CodeDeploy: per la distribuzione di artefatti

- CodePipeline: per la creazione di pipeline strutturate
- CodeStar: per avere un ambiente unificato di tutti i precedenti servizi



Questi strumenti interagiscono tra loro e permettono di creare strutture di CI/CD solide e *cost effective*.

Conclusione

In definitiva possiamo confermare quanto la metodologia di CI/CD sia uno degli strumenti di maggiore successo ed utilità nella produzione e gestione di prodotti e servizi digitali.

> *Nella famiglia dei prodotti on premise la scelta ricade quasi sempre su Jenkins, mentre su AWS sui prodotti Code*

Avere processi automatizzati significa essere più consapevoli di cosa si produce e di cosa si distribuisce e **permette di sviluppare più agevolmente** nuovi modelli di business. Modelli che godranno di maggiori risorse in quanto lo sviluppatore, non dovendo più concentrarsi anche sulle procedure di rilascio, avrà a disposizione più tempo per concentrarsi sullo sviluppo.

Da un lato avremo quindi un sensibile miglioramento della qualità del processo di rilascio, che diventerà più solido, coerente e sicuro, e dall'altro un'ottimizzazione dei tempi (e quindi delle risorse) necessarie a produrre e gestire i nostri software.