Luca Cabibbo

Architettura dei Sistemi Software

Introduzione alla delivery del software e a DevOps

Imagine a world where product owners, Development, QA, IT Operations, and Infosec work together, not only to help each other, but also to ensure that the overall organization succeeds.

Gene Kim, Jez Humble, Patrick Debois, and John Willis

dispensa asw610

ottobre 2024

- Riferimenti

- Luca Cabibbo. **Architettura del Software**: Strutture e Qualità. Edizioni Efesto, 2021.
  - Capitolo 34, Introduzione alla delivery del software e a DevOps
- Humble, J. and Farley, D. **Continuous Delivery**: Reliable Software Releases through Build, Test, and Deployment Automation. Addison-Wesley, 2010.
- Bass, L., Weber, I., and Zhu, L. **DevOps**: A Software Architect's Perspective. Addison-Wesley, 2015.
- Kim, G., Humble, J., Debois, P., and Willis, J. **The DevOps Handbook**: How to Create World-Class Agility, Reliability, and Security in Technology Organizations. IT Revolution, 2016.
- Nygard, M. **Release It!**: Design and Deploy Production-Ready Software, second edition. Pragmatic Bookshelf, 2018.

- Obiettivi e argomenti

- Obiettivi
  - introdurre la delivery del software e discutere la sua importanza
  - presentare DevOps
  - introdurre gli ambienti di esecuzione e le piattaforme
  - motivare i successivi capitoli di questa parte del corso
- Argomenti
  - introduzione
  - introduzione alla delivery del software
  - DevOps
  - ambienti
  - piattaforme
  - discussione

* Introduzione

Il rilascio del software può essere di due tipi: il team di sviluppo ha terminato il suo lavoro e quindi si può installare sui server e poi consentire l'accesso al software su questo/i server da parte dei clienti. Si parla di rilascio sia quando si parla del primo rilascio (prima del quale i clienti NON stanno usando il sistema) ma anche quando si parla di aggiornamento (i clienti stanno già utilizzando il sistema che va aggiornato)

- Questa parte del corso affronta la *delivery* (o *rilascio*) del software – ovvero, la consegna di un sistema software (o di una sua nuova versione) ai suoi utenti finali, affinché lo possano effettivamente utilizzare
  - la delivery è un aspetto architetturalmente significativo, perché ha impatto su diverse qualità del software – come la modificabilità, la disponibilità e la scalabilità
    - pertanto è importante comprendere come l'architettura del software possa sostenere la delivery del software
  - gli interessi principali sono gli ambienti di esecuzione (vista di deployment) e il processo di delivery (vista operational)
  - la delivery del software viene presentata nello spirito di *DevOps* – un movimento culturale per aiutare le organizzazioni a migliorare i loro cicli di rilascio del software

Quando si parla di tempo per il rilascio di un sistema si parla di tre componenti: costo e tempo del lavoro degli sviluppatori, costo e tempo della verifica, costo e tempo del rilascio. Con rilascio si intende dal momento in cui il team di sviluppo dice "il lavoro è finito e verificato" al momento in cui l'utente può effettivamente utilizzare il sistema/il sistema aggiornato. Questo tempo è VARIABILE.

* Introduzione alla delivery del software

- La **delivery** (*rilascio*, *consegna* o *distribuzione*) di un sistema (o di un servizio) software è l'attività di rilascio e consegna del software (o di una sua nuova versione) ai suoi utenti finali

O di una sua parte

  - ha inizio quando il team di sviluppo effettua il commit del codice sorgente di una nuova versione del software
  - si conclude quando gli utenti possono effettivamente utilizzare questa nuova versione del software

Importanza della delivery del software

- La delivery del software è un'attività importante
  - lo sviluppo di nuovo software non produce nessun valore finché il software non è stato rilasciato agli utenti

Il lavoro degli sviluppatori non ha valore finché non viene rilasciato e diventa quindi utilizzabile dai clienti

  - la capacità di effettuare rilasci affidabili, rapidi e frequenti sostiene l'innovazione e può offrire un vantaggio competitivo

La delivery del software sostiene l'innovazione, e questo perché il modo migliore per trovare la versione migliore di un codice e quindi di un sistema è quello di PROVARE. Si rilasciano varie versioni, per tempi brevi, in modo diversificato, e quindi si sperimenta e si traggono conclusioni che poi guidano i futuri sviluppi.

    - l'innovazione richiede di saper formulare delle ipotesi di business – ma anche di poterle verificare prima di attuarle
    - effettuare esperimenti è fondamentale, per convalidare le ipotesi formulate e per apprendere le preferenze degli utenti
    - anche la rapidità e la frequenza degli esperimenti sono importanti
  - avere tempi di rilascio brevi è importante anche ai fini della modificabilità e della disponibilità
  - è importante anche effettuare i rilasci in modo affidabile

Importanza della delivery del software

- La delivery del software è un'attività rischiosa
  - il rilascio di nuovo software costituisce uno dei passi più delicati nel ciclo di vita dello sviluppo del software
    Può provocare interruzione di servizio, corruzione di dati ecc
  - infatti, un errore commesso in un rilascio può avere conseguenze più o meno gravi – sulle funzionalità o sulla disponibilità del sistema
    - ad es., nel 2012 il rilascio errato di un aggiornamento del software è costato alla Knight Capital 170 mila dollari al secondo per 45 minuti circa – oltre 440 milioni di dollari
      - l'errore non era nell'aggiornamento del software – ma nel modo in cui è stato effettuato il rilascio dell'aggiornamento
    - anche l'incidente CrowdStrike del 2024 è riconducibile ad un errore in un aggiornamento di un antivirus per Microsoft Windows – con un danno finanziario complessivo di circa 5.4 miliardi di dollari

Processo di delivery del software

- La delivery del software è un processo complesso, che richiede di svolgere numerosi passi e azioni
  - quattro categorie principali di attività
    Non sono quattro attività, le attività possono essere migliaia
    - *build* (*costruzione*) – la compilazione e l'assemblaggio del codice in un formato adatto per l'installazione
    - *deployment* (*installazione*) – l'installazione del software in un ambiente di esecuzione – può richiedere anche il *provisioning* (*preparazione*) dell'ambiente
    - *test* (*verifica*) – l'esecuzione di test per verificare le funzionalità e le qualità del software
    - *release* (*rilascio*) – l'effettivo rilascio del software agli utenti, nell'ambiente di produzione

È di solito un'attivita diversa dalle precedenti, potenzialmente più breve. Ad esempio se eseguo le prime 4 categorie di attività su un nodo in quel momento inattivo, le attività di rilascio faranno in modo da sostituire il nodo "aggiornato" con quello correntemente in uso, ridirezionando quindi il traffico di richieste verso quello aggiornato che diventerà il main

Processo di delivery del software

- La delivery del software è un processo complesso, che richiede di svolgere numerosi passi e azioni
  - molte organizzazioni svolgono queste attività in modo manuale
    - il processo di delivery è lungo, complicato e soggetto a errori (è un anti-pattern)
    - queste organizzazioni vivono i rilasci del software come un incubo – per questo, effettuano i rilasci poco frequentemente

Processo di delivery del software

- La delivery del software è un processo complesso, che richiede di svolgere numerosi passi e azioni
  - molte altre organizzazioni eseguono oggi la delivery del software in modo automatizzato
    - il rilascio del software viene effettuato con un click – in modo ripetibile, rapido, frequente e affidabile
    - questo può essere desiderabile, per diversi motivi
      - per ridurre i rischi associati alla delivery
      - per aumentare il valore di business del software
      - per supportare agilità, disponibilità e scalabilità

Indica una estrema modificabilità del software

* DevOps

Developers + Operators
Devs sono pagati di più se implementano più funzionalità, Ops sono pagati di più se mantengono alta la disponibilità.
Hanno (avevano) quindi obiettivi contrastanti, innovazione e stabilità. DevOps come movimento ha l'obiettivo di rendere comuni gli obiettivi, e quindi di far lavorare in sincronia devs, ops e responsabili di business dell'applicazione.

- La delivery automatizzata del software è un interesse primario del movimento DevOps

DevOps non ha una definizione ufficiale, vediamo quella di un libro che ha a cuore però architettura del software e quindi QUALITÀ

- **DevOps** è un insieme di pratiche che hanno lo scopo di ridurre il tempo tra quando viene effettuato il commit di un cambiamento di un sistema software e quando il cambiamento viene effettivamente rilasciato in produzione, garantendo allo stesso tempo un'alta qualità [Bass et al.]
  - questa definizione "tecnica" collega DevOps alla delivery del software e alle qualità del software
- Oltre agli aspetti tecnici, DevOps è anche interessato alla capacità di innovare e di aumentare il valore di business del software
  - questo richiede dei cambiamenti nelle organizzazioni non solo tecnici ma anche culturali

Necessari anche per aspetti tecnici (ad esempio fare i test automatizzati)

DevOps

Altra definizione presa da una rivista che pubblicizza la presenza di devops nel mondo

- **DevOps** è un insieme di pratiche e di valori culturali per aiutare le organizzazioni di tutte le dimensioni a migliorare i loro cicli di rilascio del software, la qualità del software, la sicurezza e la capacità di ottenere dei feedback rapidi sullo sviluppo di un prodotto software [State of DevOps Report, 2017]
  - è un movimento culturale che enfatizza la collaborazione e la comunicazione tra le professionalità Dev e Ops
    - e con i responsabili di business
  - sostiene l'automazione e il monitoraggio di tutti i passi del processo di delivery del software – in modo che la delivery possa avvenire in modo più rapido, più frequente e più affidabile
    - e che sia strettamente allineata con gli obiettivi di business delle organizzazioni

DevOps

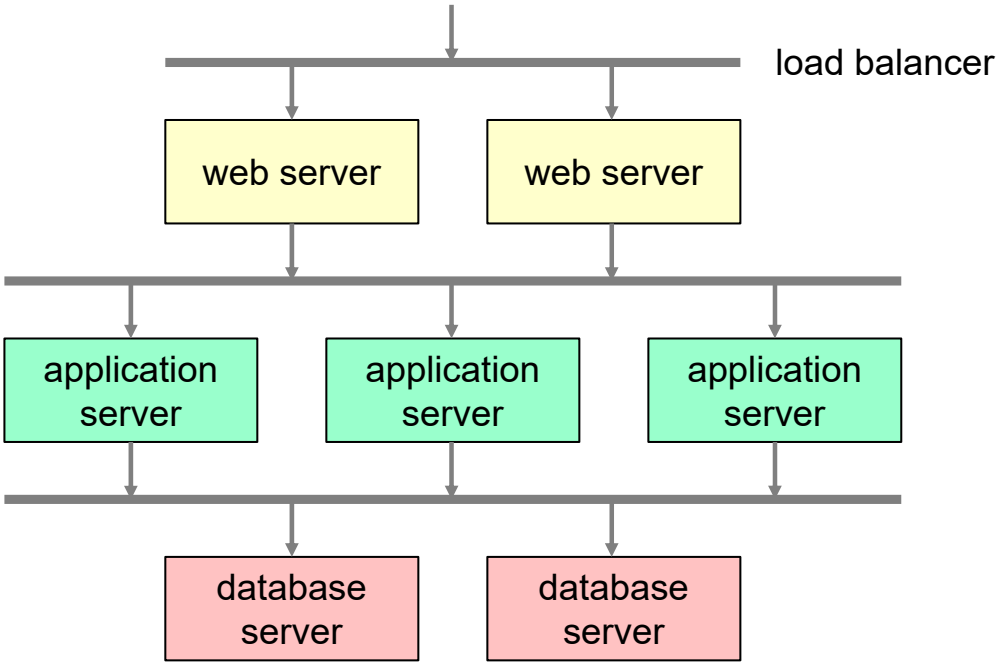
- La principale pratica tecnica DevOps è la *Continuous Delivery* (CD), per automatizzare i rilasci del software
  - ma sono importanti anche i cambiamenti culturali
    - lo sviluppo e il rilascio del software sono un unico processo
      Non sono due cose distinte
    - l'adozione dello sviluppo agile – team piccoli e agili, con cicli di sviluppo incrementali rapidi, su piccoli insiemi di requisiti, che realizzano software "pronto" al rilascio
    - l'adozione di strumenti di automazione (del controllo di versione, dei test, degli ambienti, dei deployment)
    - l'adozione di un'architettura software debolmente accoppiata
    - la collaborazione tra Dev, Ops e il business – e i cambiamenti culturali affinché il business sia in grado di sfruttare le possibilità offerte dalla CD

*Ambienti

Risorse hardware: computer, reti, dischi
Risorse software: sistemi operativi, middleware ecc
Configurazioni: in un so quali sono i ruoli, i permessi ecc

- Un *ambiente* di esecuzione per un sistema software comprende le risorse di calcolo (hardware e software) necessarie per poter eseguire il sistema software, insieme alle loro configurazioni
  - ad es., per eseguire un'applicazione web si potrebbero utilizzare più nodi – un web server, un application server e un database server – collegati in rete e replicati opportunamente
  - queste risorse computazionali possono essere fisiche o virtuali – ed essere collocate in un proprio data center o nel cloud
  - la definizione e la configurazione degli ambienti può avvenire in modo automatizzato – tramite l'utilizzo di opportuni strumenti

Ambiente: un esempio



Ambienti e configurazioni

Configurazione dei nodi

application server	
applications/ services	app/service configuration
middleware	middleware configuration
OS	OS configuration
hardware	

database server	
database	database configuration
OS	OS configuration
hardware	

Un sistema software, tanti ambienti

- Un sistema software richiede in genere più ambienti
  - l'*ambiente di produzione* — Ambiente in cui viene eseguito il software che viene utilizzato direttamente dagli utenti finali
  - l'ambiente di sviluppo — Ambiente in cui viene eseguito il software utilizzato solo dagli sviluppatori
  - uno o più ambienti di test — Possono essere più di uno, ad esempio uno dedicato al testing solo da parte degli sviluppatori, un ambiente di beta test in cui gli utenti possono loggarsi e fare test in modo autonomo, questo secondo ambiente separato dal primo
  - i diversi ambienti per un sistema software (tranne l'ambiente di sviluppo) sono di solito simili tra loro – e sono simili all'ambiente di produzione

Opzioni per le risorse di calcolo

- Ci sono numerose opzioni per le risorse di calcolo che costituiscono un ambiente
  - opzioni per i nodi
    - server fisici
    - macchine virtuali
    - container
  - opzioni di localizzazione
    - in un proprio data center (on premises)
    - nel cloud (cloud computing)
  - ulteriori opzioni
    - piattaforme
    - elaborazione serverless ("senza server")

Gestione di ambienti

- La delivery di un sistema software – in particolare, il suo deployment – può richiedere anche la preparazione dell'ambiente
  - nel *provisioning* di un ambiente, l'hardware (fisico o virtuale) viene acquisito e configurato, e in ogni nodo viene installato e configurato lo stack software richiesto (OS e middleware)

Cioè il software TRANNE la parte di software di cui si sta facendo deployment, che non fa parte del provisioning ma del deployment stesso

  - il provisioning e il deployment possono essere svolti in modo automatizzato, grazie all'utilizzo di strumenti per la gestione delle configurazioni e degli ambienti, basati su un approccio di tipo *infrastructure as code*

* Piattaforme

Le piattaforme consentono di semplificare lo sviluppo e l'esecuzione di applicazioni software; tuttavia c'è un prezzo. In particolare ciascuna piattaforma è pensata per UNA certa architettura e quindi ogni volta che decido di utilizzare una piattaforma la mia architettura deve essere conforme a quella piattaforma. Ad esempio se uso Java EE è considerata una piattaforma, quindi mi semplifica lo sviluppo di applicazioni web, però quella applicazione deve essere fatta in un certo modo.

- Le piattaforme hanno lo scopo di semplificare lo sviluppo e l'esecuzione di applicazioni software – ma richiedono che le applicazioni abbiano un'architettura specifica richiesta dalla piattaforma
  - usando alcune piattaforme potrebbe non essere necessario occuparsi direttamente dell'ambiente di esecuzione per l'applicazione
  - in questi casi, piuttosto, è la piattaforma che potrebbe occuparsi, in modo trasparente, della definizione e della gestione dell'ambiente di esecuzione per l'applicazione
  - esempi di piattaforme
    - Java EE e Microsoft .NET
    - i servizi di piattaforma (PaaS) nel cloud

Piattaforme

- Una *piattaforma* è un ecosistema di risorse per implementare ed eseguire applicazioni software
  - due tipi principali di risorse
    - un insieme di strumenti software per sviluppare applicazioni
    - un ambiente runtime per eseguire queste applicazioni
  - una piattaforma può semplificare lo sviluppo e l'esecuzione delle applicazioni – ma le applicazioni devono avere l'architettura richiesta dalla piattaforma

Piattaforme

- Alcune piattaforme moderne (come quelle per il cloud) consentono di rilasciare ed eseguire le applicazioni in un cluster di computer
  - in questi casi
    - la piattaforma definisce delle astrazioni (Indipendenti quindi) dalle risorse computazionali (distribuite) sottostanti e presenta un modello di programmazione più amichevole
    - a runtime, gestisce le risorse del cluster e vi schedula i processi – inoltre garantisce che le diverse parti di un'applicazione lavorino in modo coerente
  - alcune piattaforme consentono anche di rilasciare ed eseguire un'applicazione, in modo flessibile, sia in un cloud pubblico che in un proprio data center privato

* Discussione

- Questa parte del corso affronta i seguenti argomenti
  - ambienti di esecuzione e piattaforme
    - con riferimento a diversi tipi di infrastrutture
      - computer fisici, macchine virtuali e container
      - in un proprio data center (on premises) o nel cloud (cloud computing)
    - provisioning degli ambienti e strumenti per la gestione di ambienti
  - delivery del software
    - Continuous Delivery
    - in particolare, la delivery (lato server) di sistemi software distribuiti
  - questi aspetti sono rilevanti per molti sistemi distribuiti – ed essenziali nell'architettura a microservizi e nell'architettura nativa per il cloud