DevOps

Definizione: Applicazione dei **Principi fondamentali** per un flusso di lavoro ottimizzato.

Scopo:

- **cultura** DevOps (unisce sviluppo e operazioni aziendali) è conoscere e integrare alcuni principi atti ad ottimizzare alcune fasi aziendali
- **lavorare** nel DevOps è attuare questi principi

I principi chiave della DevOps:

i. **Collaborazione e comunicazione:**
   - Bisogna rompere le barriere cercando di creare un ambiente di lavoro dove le informazioni vengono condivise liberamente

ii. **Integrazione e automazione:**
   - **Integrare** i processi di sviluppo permettendo di ridurre i tempi di sviluppo.
   - **Automatizzare** alcune fasi di creazione di codice, test, distribuzione e monitoraggio. Ad esempio, sapere ricreare le stesse configurazioni avendo la conoscenza di alcuni meccanismi.

iii. **Feedback continuo:**
   - L'**implementazione** di feedback permette di risolvere e identificare problemi in modo più efficiente ed efficace, rientra in questa casistica il fare dei test automatizzati, il monitoraggio del sistema e il feedback degli utenti.
   - Saper **gestire** il più possibile il progetto al quale si sta lavorando è indice delle conoscenze acquisite all'interno dello sviluppo.

iv. **Pensiero orientato al cliente:**
   - Le **esigenze del cliente** devono essere al **centro di ogni attività** e al **centro dell'operatività**.
   - **Il cliente fa parte del team**.

v. **Miglioramento continuo:**
   - E' un processo iterativo che richiede impegno costante per il miglioramento.
   - I team devono monitorare i processi e identificare le aree in cui possono essere ottimizzati.

vi. **Cultura del "Caos":**
   - I team devono essere in gradi di adattarsi rapidamente alle nuove sfide e apprendere dagli errori.
   - Lor-error history, versionamento evolutivo e sistemi di disaster recovery sono ottime tecniche.
   - E' importante creare degli End-point che possano permettere di correggere degli errori.

Pratiche per migliorare il flusso di lavoro del software:

1. **Sviluppo agile**:
   - Suddividere il lavoro in piccoli incrementi
   - Controllo di versione

2. **Continuous Integration (CI):**
    - Automatizzare l'integrazione del codice da parte di diversi sviluppatori in un repository centrale.
3. **Continuos Delivery:**
    - Automatizzare il processo di rilascio del software in produzione (Repository-cliente).
4. **Infrastruttura come codice:**
    - Definire l'infrastruttura IT utilizzando codice potendo manutenere elementi fisici digitalmente.
5. **Monitoraggio e alerting:**
    - Monitorare le prestazioni del sistema e creare un registro di prestazioni ed esiti (dimmi se succede qualcosa a livello di prestazioni)
6. **Feedback**

Pratiche parallele:

1. **Cultura e mindset AKA il regno degli HR:**
    - Non è solo un insieme di strumenti o pratiche, ma è un cambiamento culturale all'interno di un'organizzazione.
    - Sono in grado di creare un ambienti di lavoro in cui i team si sentono autorizzati a sperimentare e ad assumersi la responsabilità dei propri errori.
2. **La sicurezza AKA il regno dei test:**
    - Le pratiche DevOps devono essere progettate per integrare la sicurezza fin dalle prime fasi del processo di sviluppo software.
    - Attività come: scansione del codice statico, test di penetrazione e gestione delle vulnerabilità.

DevOps e OOP:

1. **Allineamento tra sviluppo e operazioni:**
    - Promuove la compartimentazione e la modularità del progetto sfruttando lo sviluppo e l'operatività multilayer.
2. **Migliore manutenibilità e scalabilità:**
    - Quanto facilmente siamo tutti allineati sul come costruiamo le cose per poi essere sulla stessa linea per quanto riguarda la manutenibilità e la produzione in serie.
3. **Test unitari e integrazione continua:**
    - Test unitario: si occupa della più piccola unità che viene testata
    - Test di integrazione: servono a capire e trovare gli errori tra le integrazioni del codice
4. **Riutilizzo del codice e principi SOLID:**
    - Insieme di elementi e caratteristiche adattate alla scrittura di un codice testabile e riutilizzabile.
    - Fondamentale dove è necessario distribuire frequentemente nuove versioni del software.
    - Facilita la creazione di componenti modulari.
5. Modellazione del dominio e separazione delle preoccupazioni:
    - Come gestiamo un problema da risolvere quando è più grande del singolo e quindi è un problema di logica.

- La divisione dei compiti deve rispettare le conoscenze dei singoli e adeguare le task in base alle possibilità di commettere errori

Che si fa con sta DEV OPS?

Jenkins:

- Sistema di automazione dei test e di build, configurato con Java.
- Utilizzando “Jenkinsfile” o pipeline script, è possibile definire passaggi complessi come la compilazione del codice, l’esecuzione dei test unitari e di integrazione, e il feedback automatico sullo stato del build.

Docker:

- Serve a creare containerizzazione delle Applicazioni Java che contengono un elemento e le sue librerie fondamentali.
- Posso mandare la stessa sezione a tutti quelli del Team e del Cliente

Kubernetes:

- Orchestratore di container

GIT e GITHUB:

- E’ il sistema matematico logico atto al versionamento e al controllo del codice distribuito.
- GitHub aggiunge altre funzionalità come branch, merge e commit.
- Consente lo sviluppo in parallelo e decentralizzato mantenendo un nucleo controllato

- **Git Hooks:**
  - **Sono personalizzazioni impostabili per automatizzare alcune funzioni** come: test, download di file, inviare notifiche a tutti, prevenire i commit indesiderati, imporre alcuni standard.

Maven/Gradle:

- Servono ad automatizzare le build e semplificare la compilazione, testing.
- Maven ti da la sicurezza che tutto funzioni gestendo la presenza delle librerie e dipendenze cercandole all’interno del progetto.

- Il problema più grande di MAVEN è proprio gestire le dipendenze che hai inserito, l’alternativa è un rosario (prego fra)

Selenium:

- E’ un framework di automazione dei test per applicazioni web e serve ad usare script che regolino le interazioni dell’utente.
- Crea un Browser per verificare il funzionamento della nostra app e in caso di fatal error riprova finché non si arrende e si auto-distrugge

Puppet/Chef/Ansible:

- Si usano per non configurare a mano i server ma viene automatizzato e noi dobbiamo semplicemente seguire una procedura guidata.

Kubernetes:

- Sistema di orchestrazione per Container che semplifica la gestione di applicazioni distribuite su larga scala.
- Un orchestratore è un software che automatizza la gestione di applicazione distribuite, in particolare si occupa di decidere quale container va a chi e dove.

Prometheus:

- E' un framework sistema di monitoraggio che raccoglie metriche dalle applicazioni e dall'infrastruttura.
- A livello di prestazioni è utilissimo soprattutto per capire come ottimizzare un processo, senza appesantire con il monitoraggio il calcolo del progetto.