Git e Il versionamento

Python

La teoria dietro i sistemi di versionamento si basa sull'idea di migliorare la collaborazione e l'organizzazione nel ciclo di sviluppo del software. Questi sistemi consentono agli sviluppatori di:

- **Conservare la storia:** Ogni modifica ai file è tracciata insieme a dettagli come l'autore della modifica e la data.
- **Branching e merging:** Creare rami di sviluppo indipendenti (branches) per lavorare su nuove funzionalità o correzioni senza disturbare il flusso principale del progetto (main o master). Successivamente, i cambiamenti possono essere integrati (merged) nel ramo principale.
- **Gestire conflitti:** Risolvere automaticamente i cambiamenti conflittuali tra più contributi e aiutare gli sviluppatori a risolvere i conflitti manualmente quando necessario.

I sistemi di versionamento sono evoluti significativamente nel corso degli anni:

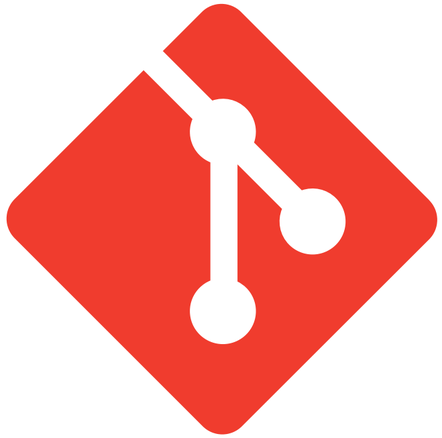
- **Sistemi di versionamento locali:** Il primo tipo di sistema di versionamento, come RCS (Revision Control System), memorizzava le modifiche in un database locale. Questo permetteva a un singolo sviluppatore di tenere traccia delle modifiche in modo semplice ma non supportava bene la collaborazione tra più utenti.

- **Sistemi di versionamento centralizzati (CVCS):** Sistemi come CVS (Concurrent Versions System) e successivamente Subversion (SVN) introducevano un modello in cui tutte le modifiche venivano memorizzate in un server centrale. Questo permetteva a più sviluppatori di lavorare insieme, ma presentava problemi di single point of failure e accessibilità limitata in assenza di connettività al server.

- **Sistemi di versionamento distribuiti (DVCS):** Con l'introduzione di Git (creato da Linus Torvalds per lo sviluppo del kernel Linux) e Mercurial, i sistemi di versionamento hanno adottato un approccio distribuito. Ogni sviluppatore ha una copia completa del repository, inclusa la storia completa delle modifiche, rendendo possibile lavorare in modo completamente autonomo e sincronizzare le modifiche con altri repository quando necessario.

- **Git è un sistema di controllo di versione distribuito (DVCS) molto diffuso, progettato per gestire progetti di ogni dimensione con velocità ed efficienza.**

- **Creato da Linus Torvalds nel 2005, Git è diventato lo standard per il versionamento nel mondo dello sviluppo software, specialmente per la gestione del codice sorgente.**



- **Git è basato su una teoria di snapshot, piuttosto che su differenze tra le versioni. Ogni volta che si salva lo stato del progetto (tramite il comando commit), Git prende uno snapshot di tutti i file che sono stati modificati dall'ultimo commit.**

- **Se un file non è cambiato, Git non salva una nuova copia, ma un link al file identico precedente.**

- **Una delle fondamenta teoriche di Git è il modello di dati che usa per gestire e immagazzinare la storia del progetto. Git considera i dati come un insieme di snapshot di un mini filesystem, con riferimenti che sono organizzati in una struttura ad albero.**

caratteristiche di Git

- **Distribuito: Ogni sviluppatore lavora con una copia locale dell'intero repository, non solo con l'ultima versione dei file. Questo significa che ogni clone di un repository Git funziona come un backup completo con tutte le funzionalità.**

- **Integrità dei dati: Git usa una combinazione di checksum SHA-1 per identificare e gestire i suoi oggetti (commits, file, directory, ecc.), il che garantisce l'integrità del codice sorgente e previene corruzione, manipolazione accidentale o intenzionale dei dati.**

- **Branching e merging veloci: Git gestisce in modo molto efficiente le operazioni di branch e merge. I branch sono leggeri (si tratta essenzialmente di riferimenti a un commit specifico) e il processo di cambio da un branch all'altro è molto rapido. Il merging è altrettanto veloce, rendendo possibile il flusso di lavoro con molteplici linee di sviluppo simultaneamente.**

Caratteristiche di Git

- **Stashing e stage area: Git offre un'area di "staging" dove i file possono essere formati per il prossimo commit, oltre alla possibilità di mettere da parte (stash) modifiche non ancora pronte per essere committate, mantenendo così il workspace pulito.**
- **Tagging: Git permette di taggare specifici commit con etichette stabili, utili per il rilascio di versioni del software (ad esempio, v1.0, v2.0 e così via).**
- **Velocità: Le operazioni in Git sono estremamente rapide, poiché la maggior parte delle operazioni ha bisogno solo di risorse locali, non di rete. Anche la gestione di grandi progetti è incredibilmente efficiente in termini di performance.**

GitHub è una piattaforma di hosting per il controllo di versione e la collaborazione, che permette agli sviluppatori di lavorare più efficacemente su progetti insieme.

Utilizza Git, il sistema di controllo di versione distribuito, per gestire i repository dei progetti.

Fondata nel 2008 da Tom Preston-Werner, Chris Wanstrath, e PJ Hyett, GitHub è diventata una delle piattaforme più popolari e ampiamente utilizzate per la collaborazione nel software development.

Principali Funzionalità di GitHub

Repository Hosting: GitHub permette agli utenti di caricare o clonare repository di codice sorgente. Questi repository sono accessibili agli sviluppatori di tutto il mondo e possono essere sia pubblici che privati.

Gestione di Branch e Merge: Attraverso l'interfaccia di GitHub, gli utenti possono facilmente creare branch e gestire i merge, facilitando i flussi di lavoro di sviluppo parallelo e la revisione del codice.

Pull Requests e Code Review: Una delle funzionalità più potenti di GitHub è la capacità di gestire "pull requests", che sono proposte o richieste di unione di un branch in un altro. Questo facilita le discussioni su specifiche modifiche con i colleghi prima di integrarle nel branch principale.

Principali Funzionalità di GitHub

Issue Tracking: GitHub offre un sistema di tracciamento degli issue integrato che permette ai team di tenere traccia dei bug, delle richieste di funzionalità e di altri compiti all'interno del progetto.

GitHub Actions: È un ambiente di CI/CD (Continuous Integration/Continuous Delivery) integrato che permette agli utenti di automatizzare test, build e deploy direttamente dalla piattaforma GitHub.

GitHub Pages: Permette agli utenti di ospitare gratuitamente siti web direttamente dai loro repository GitHub. È comunemente usato per ospitare la documentazione del progetto o siti web personali.

Branch
Un branch in Git è essenzialmente un puntatore mobile su uno degli snapshot (commits) nel tuo repository. Quando si lavora su una funzionalità, bug fix, o esperimento, creare un nuovo branch permette di lavorare isolatamente senza disturbare il flusso principale del progetto, comunemente rappresentato dal branch "main" o "master".

Fork
Un fork è una copia di un repository intero che viene ospitato nel tuo account GitHub. I fork sono utilizzati per copiare e modificare progetti senza influenzare il repository originale. Sono particolarmente utili in progetti open source, dove gli utenti possono contribuire al progetto originale attraverso i fork.

Merging
Dopo aver completato il lavoro nel branch, è possibile unire le modifiche al branch principale (ad esempio, main). Prima di fare il merge, è spesso necessario eseguire un git pull per aggiornare il tuo repository locale con le ultime modifiche del repository remoto, e risolvere eventuali conflitti.

- **Repository (Repo): Un archivio digitale dove vengono conservati e tracciati i file e la storia dei loro cambiamenti. È l'elemento base per tutto il lavoro in Git e GitHub.**
- **Commit: Un "salvataggio" o snapshot del tuo progetto in Git. Ogni commit registra le modifiche ai file e contiene un messaggio di commit che descrive cosa è stato cambiato.**
- **Branch: Una linea indipendente di sviluppo nel repository, utilizzata per lavorare su nuove funzionalità o bug fix senza disturbare il codice principale (main branch).**
- **Merge: L'azione di unire i cambiamenti da un branch a un altro, tipicamente per integrare le modifiche di una feature branch nella main branch dopo il completamento dello sviluppo.**
- **Fork: Una copia personale di un altro repository, ospitata nel tuo account GitHub. Viene utilizzato per proporre modifiche al progetto originale o per usarlo come punto di partenza per un nuovo progetto.**
- **Conflict: Si verifica quando più modifiche allo stesso pezzo di codice sono fatte in modi diversi su branch diversi. Questi devono essere risolti manualmente prima di poter completare un merge.**

- **Pull Request (PR): Una richiesta inviata ai gestori del repository originale per "tirare" le modifiche dal tuo fork o branch. È usata per discutere le modifiche proposte prima che vengano integrate nel progetto principale.**
- **Push: Il comando usato per caricare il contenuto del repository locale al repository remoto su GitHub. Questo permette di condividere le modifiche che hai fatto localmente con altri sviluppatori.**
- **Fetch: Un comando che ti permette di scaricare i cambiamenti da un repository remoto, ma senza integrarli nel tuo lavoro corrente. È utile per vedere cosa stanno facendo gli altri senza influenzare il tuo lavoro locale.**
- **Pull: Simile a fetch, ma anche integra (merge) i cambiamenti nel tuo branch attuale, aggiornando il tuo lavoro con le ultime modifiche dal repository remoto.**
- **Stash: Una funzione di Git che ti permette di temporaneamente mettere da parte le modifiche che hai fatto per poter lavorare su qualcos'altro. Puoi poi riapplicare quelle modifiche più tardi.**

GitHub Desktop è un'applicazione grafica che facilita la gestione di repository Git senza dover utilizzare la riga di comando.

È sviluppato da GitHub e offre un'interfaccia utente semplice e intuitiva, ideale per chi è alle prime armi con Git o preferisce un'interfaccia più visuale rispetto alla linea di comando.

GitHub Desktop è disponibile per Windows e macOS.

Come usare GitHub Desktop, per iniziare con GitHub Desktop, segui questi passaggi:

- **Download e Installazione: Scarica GitHub Desktop dal sito ufficiale di GitHub e installalo sul tuo sistema operativo.**

- **Configurazione dell'Account: Collega l'applicazione al tuo account GitHub (o GitHub Enterprise) seguendo le istruzioni fornite dall'applicazione.**

- **Clonazione di un Repository: Clona un repository esistente da GitHub o crea un nuovo repository locale attraverso l'interfaccia.**

- **Gestione del Codice: Usa l'applicazione per fare commit delle tue modifiche, gestire i branch, e sincronizzare le modifiche con il repository remoto.**


Buon MasterD a tutti