Git Hub

Remote

In Github le remote sono repository remote ovvero copie del progetto che si trovano su un server online e non sul computer in locale, le remote servono principalmente per salvare codice online e collaborare con altri sviluppatori e avere il progetto sincronizzato con gli altri. Quando si collega una repository locale a un remoto solitamente gli viene assegnato il nome origin per convenzione, le remote possono usare protocolli differenti per comunicare con Github come HTTPS o SSH, concludendo si può dire che è solo grazie alle remote che si riesce ad avere un versionamento distribuito e a collaborare con altri sviluppatori.

Issues

Le Issues sono ticket che servono per tenere traccia di:

- bug
- nuove funzionalità (feature request)
- to-do
- domande, discussioni tecniche o richieste di chiarimento
- miglioramenti alla documentazione

Si possono vedere come una lista di lavoro strutturata per il progetto, con titolo, descrizione, commenti, assegnatari, etichette e priorità.
NON hanno direttamente il codice al loro interno, ma sono collegate al codice tramite commit, branches e pull request.

→ se è qualcosa che richiede lavoro e vuoi che non venga dimenticato e sia visibile al team, crea una issue.

Una issue aiuta nello sviluppo software permettendo di organizzare un lavoro rendendo tutto ben visibile in un unico posto, in cui gestirlo direttamente.

Creare una issue:

- Apri repository
- Tab issues
- New issues
- Compila (titolo, descrizione, labels, assegnazione, se necessario milestone a cui assegnare)

Gestire una issue:

Pull Request

Le Pull Request sono uno strumento di Github che permette di proporre delle modifiche al codice originale di un repository, In pratica, uno sviluppatore lavora su un branch separato o su un fork, poi apre una pull request per confrontare le proprie modifiche con il codice esistente. La pull request mostra le differenze tra i file, i commit effettuati e consente agli altri membri del team di lasciare commenti o suggerimenti.
Questo strumento vince spesso utilizzato durante la revisione del codice perché permette di guardare le modifiche e fare commenti e miglioramenti.
Una volta approvata la pula request il codice verrà modificato tramite un merge tra un branch all'altro.

Best Practice per la Gestione di Repository in Git

Branch Protection:
cos'è
Per una repository condivisa (o progetti personali grandi), questa protezione protegge da:

- Push accidentali sui branch principali
- Force push (sovrascrivere la cronologia dei commit, cancellando il lavoro svolto)
- Eliminazione accidentale di branch
- Merge di codice non revisionato

Il branch protection usa una serie di regole che impediscono l'avvenire di queste situazioni, imponendo un workflow rigoroso.

Come funziona?
non permette a nessuno che non sia autorizzato di scavalcare il processo richiesto per il merge o per apportare modifiche ad un branch protetto.

Branching Strategy:
cos'è?
Una branching strategy definisce:

- quali tipi di branch creare e per quali scopi
- da quale branch creare un nuovo branch
- come e quando effettuare il merge

Entrambi consentono un'organizzazione fissa che rende il lavoro nel team uniforme e facilmente analizzabile.

.gitignore
cos'è
E' un file di testo nel quale sono presenti file che non vengono tracciati da git nonostante si trovino nella working directory.

A cosa serve
Serve per evitare che git esegua il commit:

- file di configurazione personali
- dati sensibili
- cartelle temporanee
- *in generale tutte le cose che non dovrebbero trovarsi in una cartella condivisa, e che servono solo agli scopi del singolo dev.*

Si colloca nella root directory, la radice del progetto. Così facendo .gitignore copre tutto il progetto

Github Actions

Cosa sono?
Github actions è una piattaforma di automazione integrata direttamente in GitHub:

- permette di automatizzare task ripetitive del ciclo di sviluppo
- esegue test automaticamente
- compila codice, esegue il deploy, pubblica pacchetti

Tutto automaticamente.

Come funzionano?