Controllo di versione

Il controllo di versione (detto anche controllo di versione distribuito) è un sistema che agevola la gestione di versioni multiple di un insieme di informazioni, come i documenti digitali o il codice sorgente dei linguaggi di programmazione.

Viene usato principalmente nello sviluppo di progetti ingegneristici o informatici per gestire la continua evoluzione dei file, tenendo traccia delle varie fasi dell'evoluzione di ogni file del progetto.

Il controllo di versione è normalmente basato su architettura client-server, questa architettura offre intrinsecamente vantaggi in termini di efficienza, flessibilità e scalabilità.

Il controllo di versione ha molti vantaggi, come:

- evitare la sovrascrittura o la perdita di documenti importanti
- individuare facilmente le differenze tra le versioni dei file
- tenere traccia di chi ha modificato i file e quando
- collaborare con altri sviluppatori in modo efficiente e sicuro
- ripristinare una versione precedente del file in caso di errori o problemi
- lavorare in modo produttivo anche senza connessione a internet
- condividere le modifiche all'interno del gruppo di lavoro in modo flessibile e scalabile
- avere una copia locale del repository che funge da backup

GIT

GIT è un software di controllo versione distribuito utilizzabile da interfaccia a riga di comando, anche se esistono numerose interfacce grafiche, soprattutto per sistemi operativi Windows.

GIT lavora con i repository, che sono le collezioni di file e cartelle che compongono il progetto.

Un repository GIT è una directory che contiene tutti i file e la documentazione di un progetto gestito con il controllo di version, il clone locale del progetto è un repository completo che permette di lavorare offlin, sincronizzando le modifiche con il repository remoto quando è necessario.

Flusso di lavoro

Un flusso di lavoro è una modalità di gestione dei progetti, pensata per sfruttare al meglio il controllo di versione in modo da lavorare in modo coerente e produttivo.

In GIT non esiste un flusso di lavoro unico, ma è possibile creare flussi personalizzati a seconda delle esigenze e delle modalità di lavoro del team.

Alcuni esempi di flussi di lavoro GIT sono:

- Flusso di lavoro base: si usa solo il ramo master per fare i commit e la distribuzione.
- Flusso di lavoro a rami: si usano più rami per separare le funzionalità e i bugfix.
- Flusso di lavoro Gitflow: si usano due rami principali (master e develop) e dei rami ausiliari (feature, release e hotfix) per gestire le diverse fasi dello sviluppo.

Per scegliere il flusso di lavoro più adatto ad un progetto, devono essere presi in considerazione alcuni fattori come la dimensione del team, la frequenza delle modifiche, la qualità del codice e il tipo di distribuzione.

Principali comandi GIT

Il protocollo GIT prevede molti comandi, alcuni di uso molto comune (durante il lavoro si utilizzano più volte al giorno), altri di uso molto raro.

I principali comandi GIT sono:

- git clone <URL del repository>: per creare la copia locale di un repository remoto.
- git status: per mostrare lo stato della working directory ed elencare i file modificati, aggiunti e cancellati.
- git add <nome del file>: per aggiungere i file modificati o nuovi all'area di staging, che fa da spazio di transito per i file.
- git commit -m "<messaggio di commit>": per registrare le modifiche dei file nell'area di staging nel repository locale con un messaggio descrittivo. NB: in GIT il messaggio è obbligatorio, in altri protocolli è opzionale.
- git push: per inviare le modifiche dal repository locale al repository remoto.
- git pull: per scaricare le modifiche dal repository remoto al repository locale e integrarle con le proprie modifiche.
- git branch <nome del branch>: per elencare, creare o eliminare i rami del repository, che sono delle linee di sviluppo indipendenti.
- git merge <nome del branch>: per riunire un ramo di sviluppo in quello principale