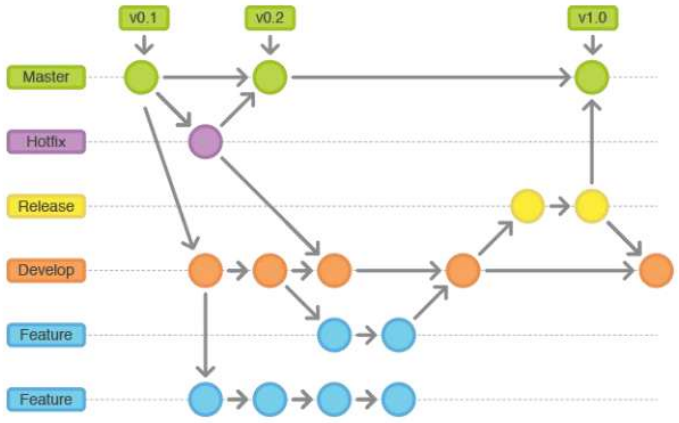
Lezione 22 Git

venerdì 17 novembre 2023 11:19

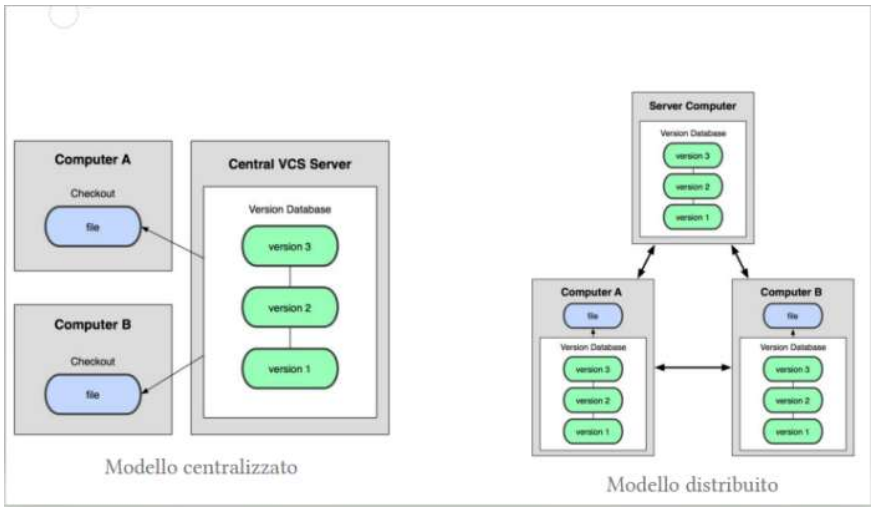
Git è uno strumento che ci permette di gestire il controllo delle versioni del codice . In generale lo sviluppo di una qualsiasi applicazione non è lineare : sia per il numero di persone che ci lavorano , sia per le versioni ecc ecc . In dettaglio lo sviluppo del software di solito è cosi :



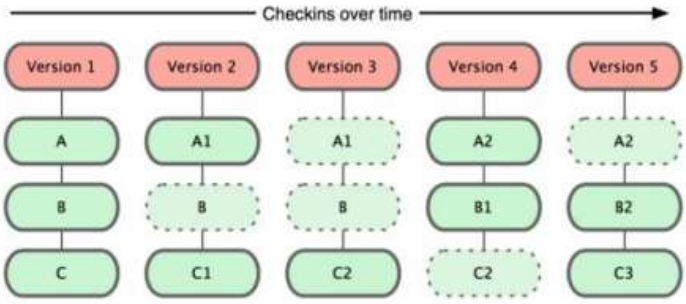
Le operazioni in generale che si fanno durante lo sviluppo del codice sono le seguenti :

Commit	"ho modificato la mia applicazione" : permette di inserire queste modifiche fatto in modo atomico e le rende permanenti a tutti gli utenti
File locking	Gestione atomica dei file : sistema semplice della concorrenza in quanto un solo utente può modificare quel file
Merge	Gli sviluppatori modificano lo stesso file , quindi attenzione alle modifiche se in stesse parte del file . Risoluzione manuale
Tagging	Uno specifico commit può essere associato ad un'etichetta

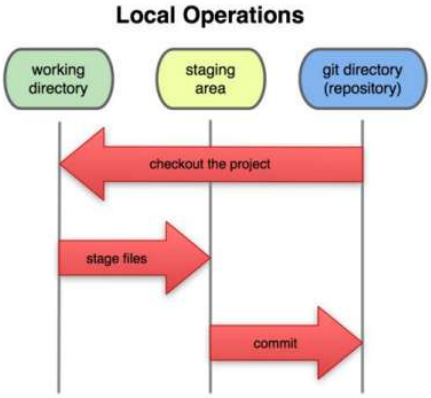
Come funziona il modello distribuito??

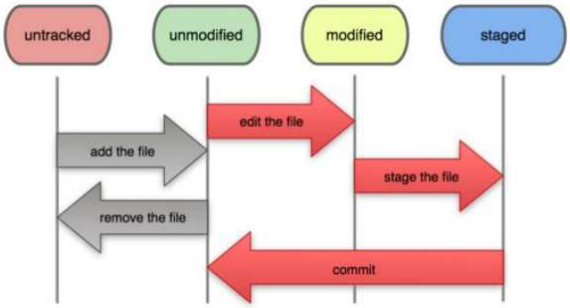


Da notare che nel modello centralizzato , ogni cambio/modifica del programma dovevano passare per il server centrale , mentre in quello distribuito non si deve passare per forza per quello centrale.
I file vengono salvati attraverso SNAPSHOTS : istantanee delle versioni (evoluzione nel tempo) -> possibile andare avanti ed indietro con le versioni :



Quindi in generale la vita di un programma parte dal repo git e mano mano si inviano file, avendo queste 3 arre di lavoro :





Vediamo ora i comandi principali :

1. git config --global user.name “Bugs Bunny"
    a. Imposto autore di commit
2. git config --global user.email bugs@example.com
    a. Imposto la mail di commit
3. git config -list
    a. Vedo tutti gli utenti
4. $ git clone <url> [local dir name]
    a. Copia di repository remoto con la storia
    b. Url è il link http ://
5. $ git init
    a. Creo repo nuovo
    b. $ git add README.md
        i. Inserisco un file
    c. $ git commit –m “initial project ve
        i. Inserisco il file nel repo con opportuno messaggio
6. $ git add README.txt hello.java
    a. Sposto il file da locale ad area di staging
    b. git commit –m “Fixing bug #22”
        i. Mando il file nel repo
7. $ git push origin main
    a. Pubblico su repository remoto i commit del repo locale
8. $ git pull origin main
    a. Recupero dati da repo condiviso

9. $ git remote -v
    a. Informazioni sul/sui repo remoti
10. $ git status
    a. Stato del repo locale
11. $ git diff
    a. Mostra cosa è stato modificato ma non ancora inserito in area di staging
12. $ git diff --cached
    a. Mostra differenze nell'area di staging
13. Si possono avere piùrami di lavoro : i **branch**
    a. $ git branch <nome>
        i. Crea un altro ramo di lavoro
    b. $ git branch
        i. Mostro le branch locali
    c. $ git checkout <nome>
        i. Passo da branch ad altro
    d. $ git checkout master+ git merge <name>
        i. Fondo insiemi i branch
14.

comando	descrizione
`git clone url [dir]`	copia un repository git in modo da potervi aggiungere qualcosa
`git add files`	aggiunge il contenuto dei file all'area di staging
`git commit`	registra un'istantanea dell'area di staging
`git status`	mostra lo stato dei file nella directory di lavoro e nell'area di staging
`git diff`	mostra la differenza tra ciò che è in staging e ciò che è modificato, ma non in staging.
`git help [command]`	ottieni informazioni di aiuto su un particolare comando
`git pull`	recupera da un deposito remoto e cerca di fonderlo con il ramo corrente
`git push`	spinge i nuovi rami e i dati in un repository remoto
altri: `init, reset, branch, checkout, merge, log, tag`	