# GIT, semplice e performante!

Sei un programmatore? Un webmaster? Aggiorni spesso il tuo sito?

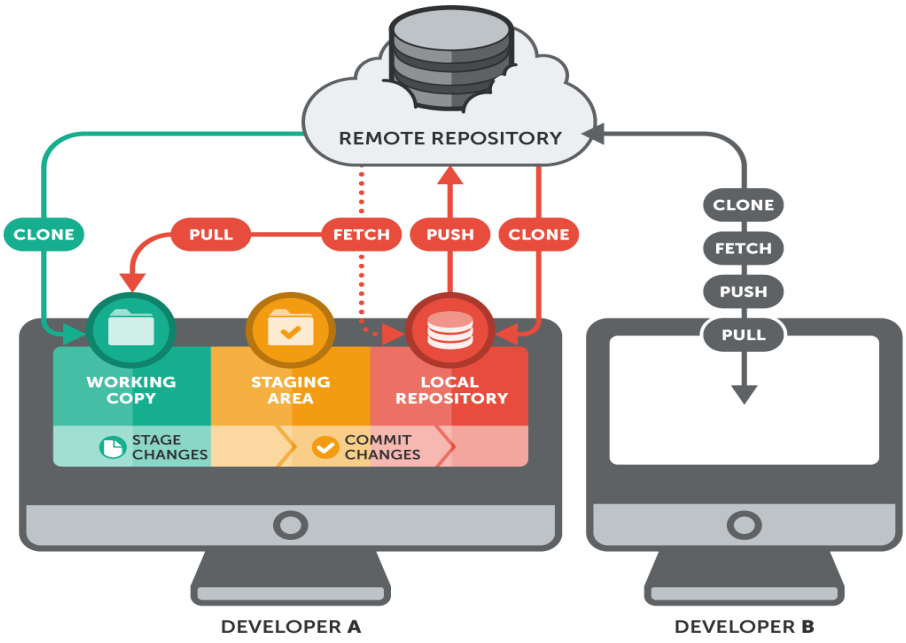
Hai bisogno di creare un progetto sul quale lavorare in gruppo mantenendo traccia delle modifiche e avendo la possibilità di recuperare diverse versioni di un file o dell'intero progetto?

Adesso su Netsons supporta anche Git!!!

## Cos'è Git?

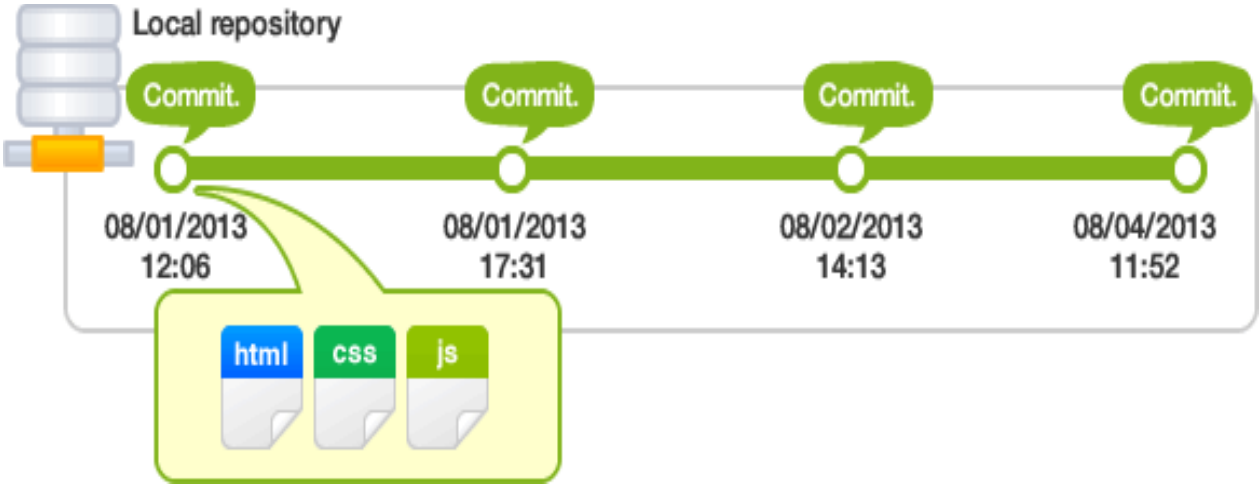
Git è un sistema di controllo di versione distribuito (Distributed Version Control Systems o DVCS) sviluppato da Linus Torvalds che sta diventando rapidamente uno dei più diffusi in circolazione.

In un sistema di controllo di versione distribuito, ciascun client fa anche da server per se stesso e possiede una copia locale del repository.




Git, considera i propri dati come una serie di istantanee di un mini filesystem. Ogni volta che l'utente effettua un commit, o salva lo stato del proprio progetto,

crea un'immagine di tutti i file presenti in quel momento, salvando un riferimento. Se alcuni file non sono stati modificati, GIT non li clona ma crea un collegamento agli stessi file della versione precedente.




Altro aspetto molto interessante è la **possibilità di lavorare anche off-line** con il server centrale. L'utente può lavorare sulla propria copia locale del repository e rendere pubbliche le modifiche quando il server torna online.

# Git step by step.

**Installazione**

Scarica git:

1. Su piattaforma Linux: https://git-scm.com/download/linux
2. Su piattaforma Windows: https://git-for-windows.github.io/
3. Su piattaforma OSX: https://code.google.com/archive/p/git-osx-installer/downloads

**Crea un nuovo repository**

Crea una cartella, entraci da linea di comando e lancia :

```
git init
```

**Crea un copia del repository**

Per creare una copia locale lancia il comando:

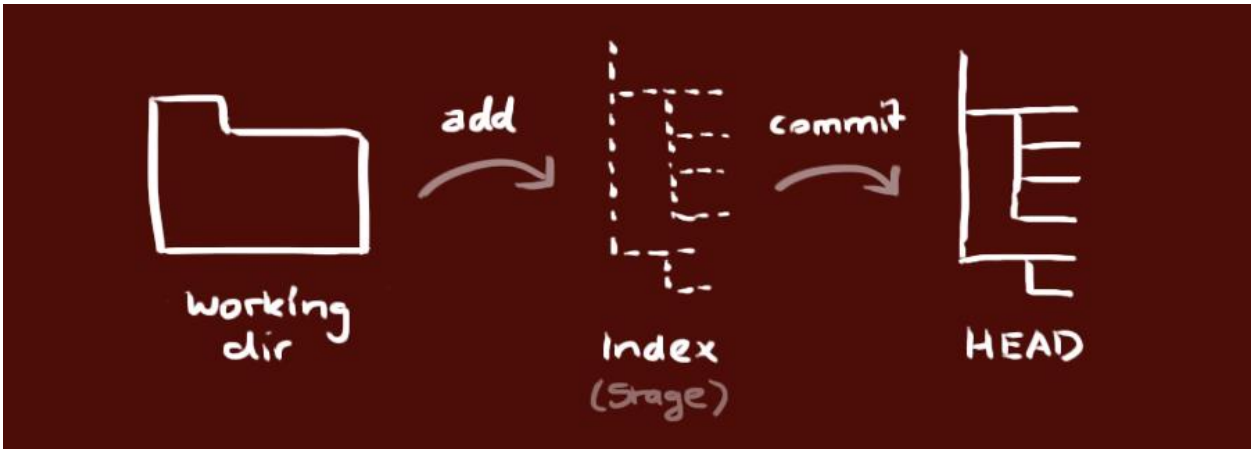
```
git clone /percorso/del/repository
```

Per usare un server remoto, invece, il comando sarà:

```
git clone nomeutente@host:/percorso/del/repository
```

**Ambiente di lavoro**

La tua copia locale del repository è composta da tre "alberi" mantenuti da git.
Il primo è la tua **Directory di lavoro** che contiene i files attuali.
Il secondo è l'**Index** che fa da spazio di transito per i files e per finire l'**HEAD** che punta all'ultimo commit fatto.



## Aggiunta e validazione

Puoi proporre modifiche (aggiungendole all'**Index**) usando :

```
git add <nomedelfile>
```

```
git add *
```

Questo è il primo passo nel flusso di lavoro in git. Per validare queste modifiche fatte si usa:

```
git commit -m "Messaggio per la commit"
```

Ora il file è correttamente nell'**HEAD**, ma non ancora nel repository remoto.

## Invio delle modifiche

Quello che hai cambiato ora è nell'**HEAD** della copia locale. Per inviare queste modifiche al repository remoto, esegui:

```
git push origin master
```

Cambia *master* nel branch al quale vuoi inviare i cambiamenti.

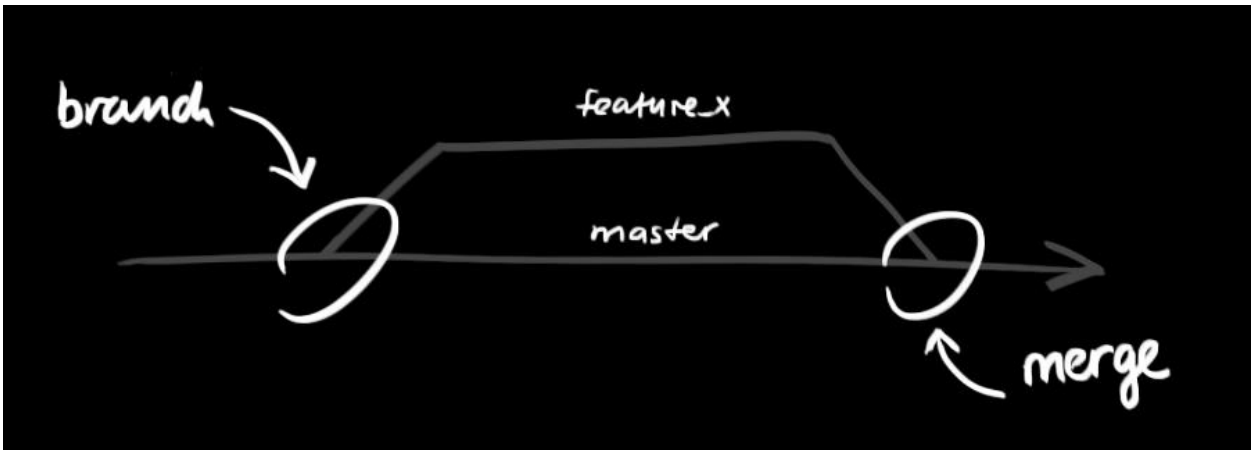
Se non hai copiato un repository esistente, e vuoi connettere il tuo repository ad un server remoto, c'e' bisogno che tu lo aggiunga con:

```
git remote add origin <server>
```

Ora sarai in grado di inviare le tue modifiche al server remoto specificato.

**Branching**

I branch ('ramificazioni') sono utilizzati per sviluppare features che sono isolate l'una dall'altra. Il branch master è quello di default quando crei un repository. Puoi usare altri branch per lo sviluppo ed infine incorporarli ('merge') nel master branch una volta completati.




crea un nuovo branch chiamato "feature_x" e passa al nuovo branch usando:

```
git checkout -b feature_x
```

ritorna di nuovo su master:

```
git checkout master
```

e cancella il branch creato in precedenza:

```
git branch -d feature_x
```

il branch non sarà disponibile agli altri fino a quando non verrà inviato al repository remoto:

```
git push origin <branch>
```

## Aggiorna e incorpora

Per aggiornare il tuo repository locale alla commit più recente, esegui:

```
git pull
```

nella tua directory corrente per fare una fetch (recuperare) ed incorporare (merge) le modifiche fatte sul server remoto.

Per incorporare un altro branch nel tuo branch attivo (ad esempio master), utilizza:

```
git merge <branch>
```

in entrambi i casi git prova ad auto-incorporare le modifiche. Sfortunatamente, a volte questa procedura automatizzata non è possibile, ed in questo caso ci saranno dei conflitti. Sei tu il responsabile che sistemerà questi conflitti manualmente modificando i file che git mostrerà. Dopo aver cambiato questi files, dovrai marcarli come 'correttamente incorporati' tramite:

```
git add <nomedelfile>
```

prima di immettere le modifiche, potrai anche visualizzarne un'anteprima eseguendo:

```
git diff <branch_sorgente>
```

## Tags

È raccomandato creare dei tags nel caso in cui il software venga rilasciato. Questo è un concept già conosciuto, che esiste anche in SVN. Puoi creare un tag chiamato 1.0.0 eseguendo:

```
git tag 1.0.0  1b2e1d63ff
```

la sequenza 1b2e1d63ff sta per i primi 10 caratteri del commit che si vuol referenziare tramite questo tag. Puoi ottenere l'id della commit tramite:

```
git log
```

puoi anche utilizzare meno caratteri per l'id della commit, basta che sia unico.

## Sostituire i cambiamenti locali

Nel caso tu abbia fatto qualcosa di sbagliato, puoi sostituire i cambiamenti fatti in locale con il comando:

```
git checkout -- <nomedelfile>
```

questo rimpiazza le modifiche nell'albero di lavoro con l'ultimo contenuto presente in HEAD. I cambiamenti fatti ed aggiunti all'index, così come i nuovi files, verranno mantenuti.

Se vuoi in alternativa eliminare tutti i cambiamenti e commits fatti in locale, recupera l'ultima versione dal server e fai puntare il tuo master branch a quella versione in questo modo:

```
git fetch origin
```

```
git reset --hard origin/master
```

## Suggerimenti utili

GUI (Interfaccia utente grafica) per git disponibile di default:

```
gitk
```

colora gli output di git:

```
git config color.ui true
```

mostra il log in una riga per commit:

```
git config format.pretty oneline
```

utilizza l'aggiunta interattiva:

```
git add –i
```

# Links e risorse

## clients grafici

- GitX (L) (OSX, open source) : http://gitx.laullon.com/
- Tower (OSX): http://www.git-tower.com/
- Source Tree (OSX, free): http://www.sourcetreeapp.com/
- GitHub per Mac (OSX, free): http://mac.github.com/
- GitBox (OSX): https://itunes.apple.com/gb/app/gitbox/id403388357?mt=12

## le guide

- Git Community Book: http://book.git-scm.com/
- Pro Git: http://progit.org/book/
- Think like a git: http://think-like-a-git.net/
- GitHub Help: http://help.github.com/
- A Visual Git Guide: http://marklodato.github.com/visual-git-guide/index-en.html