Git Guide

Alexandru Andrei Colacel

11 gennaio 2024

Git

I file in Git possono essere in tre stati principali: modified (modificati), staged (in stage) e committed (committati).

- **Modificato** significa che il file è stato modificato, ma non è ancora stato committato nel database.
- In **stage** significa che hai contrassegnato un file, modificato nella versione corrente, perché venga inserito nello snapshot alla prossima commit.
- **Committato** significa che il file è registrato al sicuro nel database locale.

Aggiungere nome ed e-mail

- Aprire la linea di comando
- Settare il proprio user name digitando:

```
git config --global user.name "FIRST_NAME LAST_NAME"
```

- Settare la propria email:

```
git config --global user.email "MY_NAME@example.com"
```

Controllare nome ed e-mail

- Aprire la linea di comando
- Controllare il proprio user name digitando:

```
git config --global user.name
```

- Controllare la propria email:

```
git config --global user.email
```

Comandi

Inizializiamo una repository di una directory esistente

Se abbiamo una directory già esistente allora possiamo eseguire

```
$ git init
```

Questo crea una nuova subdirectory chiamata `.git` che contiente tutti gli eelmenti necessari ai file della repository. Ma a questo punto ancora nessun cambiamento del nostro progetto non è tracciato. Se vogliamo iniziare a controllare le versioni dei file esistenti dovremmo fare un `git add` specificando il file che vogliamo aggiungere.

```
$ git add *.c
$ git add LICENSE
$ git commit -m 'initial project version'
```

Committiamo i cambiamenti

Dobbiamo precisare che ai cambiamenti verranno aggiunti solo i file che abbiamo aggiunto con `git add`. Ora eseguiamo:

```
$ git commit
```

Ora per modificare il file di configurazione di git dobbiamo eseguire:

```
$ code ~/.gitconfig
```

Controllare lo status dei nostri files

Il comando principale da usare per determinare quali file sono stati aggiunti e per determinare lo stato nel quale si trovano bisogna usare il comando:

```
$ git status
```

Il comando `git log` visualizza tutti i commit nella cronologia di una repository. Eseguendo quindi:

```
$ git log
```

Visualizzeremo:

- Secure Hash Algorithm (SHA)
- autore
- data
- messaggio di commit

Come ripristinare un percorso

Se ci siamo resi conto che un certo `commit` non era necessario allora dobbiamo prendere la lista degli hash con il comando:

```
$ git log --oneline
    436ad25 (HEAD -> master) update
    0d014f0 Aggiunto nuovi file
```

e poi dobbiamo eseguire:

```
$ git checkout 0d014f0
```

Git Revert e Git Reset (soft, mixed, hard)

Permette di tornare indietro nel tempo. Il primo comando è:

```
$ git reset --mixed 0d014f0
    o
$ git reset --soft 0d014f0
    o
$ git reset --hard 0d014f0
```

Ignoriamo dei Files o Directory con Git

Bisogna creare un file `.gitignore` e inserire il path da ignorare.

Come gestire una repository su GitHub da CLI

Push

Creare una nuova repository:

```
$ git init
$ git add FILE.extension
$ git commit -m "first commit"
$ git remote add origin https://gitub.com/NOME_SU_GITHUB/NOME_PROGETTO.git
$ git push -u origin master
```

Aggiungere ad una repository esistente:

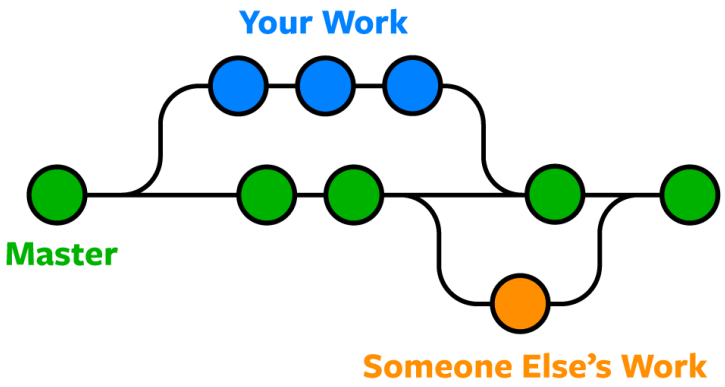
```
$ git remote add origin https://gitub.com/NOME_SU_GITHUB/NOME_PROGETTO.git
$ git push -u origin master
```

Pull

Il comando per scaricare gli aggiornamenti ai file sono:

```
$ git pull origin master
```

Cos'è un Branch



Creare un nuovo branch aiuta a organizzare il lavoro e permettere il lavoro di diversi utenti. Il comando per la creazione di un nuovo branch:

```
$ git branch NOME-BRANCH
```

Una volta effettuati tutti i cambiamenti dobbiamo fare il `commit` su questo branch e poi il `push`.

Eliminare da GitHub il branch bisogna eseguire:

```
$ git push origin --delete NOME_BRANCH
```

Ma per elimminarlo anche in locale bisogna prima fare lo switch al branch `master` con `git checkout master` e poi eliminare con:

```
$ git branch -D NOME_BRANCH_DA_ELIMINARE
```

Git merge

Una volta soddisfatti dei nostri cambiamenti possiamo fare lo switch nel branch `master` e poi eseguire il `merge`:

```
$ git checkout master
$ git merge NOME_BRANCH_CREATO
```

Si può proteggere un branch nelle impostazioni di GitHub.