Github

Comandi base

git init

inizializza la cartella git (storico salvataggi file)

git status

mostra lo stato dei file all'interno del progetto

git add

aggiunge i file all'interno della repo git.
git add . : aggiunge tutti i file.
git add <nomefile> : aggiunge il file <nomefile> alla repo git (è possibile indicare più file da aggiungere).
git add *.estensione : aggiunge tutti i file con estensione .estensione alla repo git.

git commit -m "Titolo"

serve ad eseguire il commit e a dare un titolo
(posso aggiungere una descrizione con git commit -m "Titolo" -m "descrizione")

git restore --staged <nomefile>

rimuove il file <nomefile> dalla repo git

git log

mostra i conti eseguiti fino a questo momento e le loro informazioni (id, titolo, descrizione, autore, data)
Si possono utilizzare opzioni quali --oneline per delle informazioni più compatte, --reverse per vederle in ordine inverso

git commit --amend

modifica titolo/descrizione ultimo commit

git reset

serve ad eliminare i commit

git reset --soft : riporta lo stato alla staging area precedente al commit.
git reset --mixed : riporta lo stato preccedente alla staging area (default).
git reset --hard : cancella sia il commit che eventuali file aggiunti non presenti ad un commit precedente dalla working directory.

N.B. vicino al comando reset oltre alla modalità si deve indicare **HEAD** + uno tra **<id-commit>/^/~**

<id commit> riporta lo stato al commit di cui è stato indicato l'id.
^ torna indietro di un commit per ciascun ^ presente.
~#commit torna indietro di un numero di commit pari a quello indicato dal numero insetiro.

ESEMPIO: ***git reset --soft HEAD^^*** → torna indietro di 2 commit riportando i file non prensenti al commit in questione alla staging area

Gitignore

touch .gitignore

crea un file .gitignore che ignora i file e le cartelle contenuti in esso dalla funzione di add e di conseguenza commit.
È possibile aprire il file .gitignore con un editor di testo, scrivere i nomi dei file e/o delle cartelle che si desidera ignorare e salvare il file, poi fare un add e un commit.
All'interno del file .gitignore è possibile scrivere il **nome del file/cartella** che si desidera ignorare, o in alternativa ***.estensione** per rimuovere tutti i file con l'estensione indicata.
!<nomefile> permette di rendere visibile un file la cui estensione è indicata all'interno del file .gitignore di cui però si vuole mantenere la visibilità.
È possibile inserire commenti a linea singola all'interno del file '.gitignore' iniziando la riga con il carattere **#**

Branch

git branch

mostra i branch esistenti nel progetto. Il branch su cui ci troviamo al momento corrente è indicato da un *

git branch <nomebranch>

serve per creare nuovi branch.

git checkout <nomebranch>

sposta il puntatore HEAD verso il branch **<nomebranch>**, quindi in parole povere serve a spostarsi tra i vari brach.

Git checkout -b <nomebranch>

creau un nuovo branch **<nomebranch>** e sposta il puntatore HEAD sullo stesso.

git merge <nomebranch>

serve ad unire il branch corrente con il branch **<nomebranch>**

git branch --merged

serve a vedere quali branch sono stati uniti.

git branch -d <nomebranch>

serve ad eliminare il branch **<nomebranch>**. Funziona solo se il branch è stato unito ad un altro branch

git branch -D <nomebranch>

serve ad eliminare il branch <nomebranch>.

git branch --abort

serve ad interrompere il merge pendente tra due branch.

Remote server

git remote add <nomeServer> <urlServer>

serve per inserire l'indirizzo di un server remoto a cui effettuare push e pull

git remote

Fornisce la lista dei nomi dei server remoti registrati in precedenza.

git remote -v

fornisce la lista dei nomi e degli url dei server remoti registrati in precedenza.

git remote show <nomeServer>

Serve ad ispezionare la configurazione del server **<nomeServer>**

git remote rename <nomeServer> <nuovoNomeServer>

serve a rinominare il server **<nomeServer>** in **<nuovoNomeServer>**

git remote remove <nomeServer>

serve ad eliminare il server.

Sincronizzazione

git push -u <nomeServer> <nomeBranch>

serve per fare l'upload del nostro codice ad un server remoto. L'opzione -u serve a specificare nella configurazione un server remoto di default.

git fetch

sincronizza le informazioni della repository remota nella repository locale.
Dopo è necessario effettuare il comando git merge per unire i commit.

git pull

sincronizza le informazioni della repository remota nella repository locale unendo automaticamente i commit

git clone <URL_Server>

clona il remote server **<URL_server>** in locale, mantenendo tutti i commit e le modifiche nel tempo.
Il comando clone accetta anche alcune opzioni:
git clone <URL_Server> <nomeCartella> clona il server in locale assegnando il nome **<nomeCartella>** alla repository appena clonata.

Comandi utili bash

cat <nomefile>

legge il file **<nomefile>** e ne stampa il contenuto a schermo

vim <nomefile>

apre il file in un editor di testo da terminale.
Cliccando **i** inizia la fase di inserimento in cui si può scrivere ciò che si vuole al documento.
Cliccando **Esc** e digitando **:wq** si salva il contenuto e si chiude il documento.