Git Cheatsheet

Configurazione di un repository

`$ git init <directory>` Crea un **repository** (o più brevemente repo) locale (ometti <directory> per inizializzare la directory corrente come repository git)

`$ git clone <url>` Scarica una repo remota

```
$ git config --global user.name “Mario Rossi”
$ git config --global user.email mariorossi@gmail.com
```
Imposta il nome e l'e-mail che verranno allegati ai tuoi **commit** e **tag**

Salvataggio delle modifiche

`$ git add <file>` Aggiunge il file nello **staging**

`$ git add .` Aggingi nello stage tutti i file

`$ git add *` Aggingi nello stage tutti i file (tranne quelli di configurazione)

`$ git commit -m “commit msg”` Aggiunge tutte le modifiche apportate ai file tracciati e esegui il commit

`$ git commit -a` Fa la commit di tutti i file in staging

`$ git commit --amend -m “commit msg”` Permette di cambiare il messaggio dell'ultima commit

`$ git diff` Mostra i cambiamenti nei file non i staging.
Per applicare i cambiamenti ai file in staging aggiungere il flag --cached

`$ git diff commit1_ID commit2_ID` Mostra i cambiamenti tra due commit

Annullamento di commit e modifiche

`$ git mv <existing-path> <new-path>` Muove (e/o rinomina) un file e il suo stage

`$ git rm <file>` Rimuove un file dalla directory in cui sta lavorando e dalla staging area, in seguito inserisce nello stage i file rimossi

`$ git rm --cached <file>` Rimuove il file solo dall'area di staging

`$ git checkout <commit_ID>` Sposta su una commit precedente (solo lettura) per effettuare dei controlli delle modifiche

`$ git revert <commit_ID>` Crea un nuovo commit, annullando le modifiche di un commit specifico

`$ git reset <commit_ID>` Torna indietro di un commit specifico e cancella tutti i commit avanti a lui

Stashing

`$ git stash` Inserisce i file modificati nello stash

`$ git stash pop` Riprende i file dallo stash

.gitignore

Git vede ogni file della tua repo in uno di questi modi: tracked (in stage o committato), untracked e ignored.
I file ignorati sono in genere artefatti di compilazione e file generati dalla macchina che possono essere derivati dall'origine del repository o altrimenti non dovrebbero essere sottoposti a commit.

Alcuni esempi comuni sono:

- cache delle dipendenze
- codice compilato
- output di una build
- file generati in fase di esecuzione
- file di sistema nascosti
- file di configurazione IDE personali

Controllare una repo

`$ git status` Mostra la lista dei file nuovi o modificati non ancora committati

`$ git log --oneline` Mostra la cronologia dei commit, con i rispettivi ID (con il flag --oneline le informazioni sono condensate in una riga)

`$ git log --oneline --decorate` Il flag --decorate fa in modo che git log mostri tutti i riferimenti (ad esempio rami, tag, ecc.) che puntano a ciascun commit.

`$ git log --graph --oneline --decorate` L'opzione --graph disegna un grafico ASCII che rappresenta la struttura del ramo della cronologia dei commit.

`$ git tag <tag-name>` Aggiunge un tag al commit corrente (spesso utilizzato per il rilascio della nuova versione). Un tag è come un ramo che non cambia. A differenza dei rami, i tag, dopo essere stati creati, non hanno ulteriore cronologia dei commit.

`$ git blame <file>` Mostra la lista dei metadati legati all'autore di un file committato.

Branches

Collaborating

`$ git branch` Elenca tutti i branch locali. Aggiungi il flag -r per mostrare tutti i branch remoti, -a per tutti i branch

`$ git branch <nuovo-branch>` Crea un nuovo branch

`$ git checkout <nuovo-branch>` Sposta in un altro branch e aggiorna la directory in cui stai lavorando

`$ git checkout -b <nuovo-branch>` Crea un nuovo branch e spostati su di esso

`$ git branch -d <branch>` Cancella un branch mergiato

`$ git branch -D <branch>` Cancella un branch, se già mergiato o no

Syncrionizing

Collaborating

`$ git remote add <alias> <url>` Aggiunge una repo remota

`$ git remote` Mostra tutte le connessioni remote. Aggiugere il flag -v per vedere gli url

`$ git remote remove <alias>` Rimuove una connessione

`$ git remote rename <old> <new>` Rinomina una connessione

`$ git fetch <remote>` Effettua il fetch (scarica le modifiche modifiche) tutti i branch da una repo remota (senza merge)

`$ git fetch <remote> <branch>` Effettua il fetch di un branch specifico

`$ git pull` Effettua il fetch della copia remota della repo del branch corrente, e esegue il merge

`$ git pull -rebase <remote>` Sposta le modifiche locali (rebase) all'inizio delle nuove modifiche fatte alla repo remota (per pulire lo storico)

`$ git push <remote> <branch>` Carica il contenuto locale nella repo remota e nel branch indicato

Merging

Collaborating

`$ git merge a` Mergia (fonde) il branch a nel branch b. Aggiungere il flag --no-ff per un no-fast-farward merge

`$ git merge --squash a` Merge & squash di tutti i commit in un unico nuovo commit

@dianabernabei