Guida rapida per i comandi Git

Luca Morlino

26 febbraio 2021

Capitolo 1

Operazioni preliminari

- Per impostare un nome utente di default :

  git config --global[1] **user.name "YOUR NAME"**
  es. git config --global user.name "JOHN SMITH"

- Per specificare una mail di default:

  git config --global user.mail "your.email@provider"
  es. git config --global user.email johnsmith@example.com

- Impostazione carattere di new line:

  Dato l'uso di diversi sistemi operativi bisogna configurare git affinchè non reinterpreti il fine linea. Il fine linea va settato sull'IDE (Text File Encoding:UTF-8; New Text File Line Delimiter: Unix) mentre su git va disattivato.

  Per utenti Unix e Mac-OS:

  git config --global core.autocrlf input

  Per utenti Windows:

  git config --global core.autocrlf false

[1] il flag --global setta le impostazioni non per il repository corrente ma per qualunque nuovo repository

Capitolo 2

Primi passi: creare un repository e fare un commit

Spostarsi sulla cartella dove si vuole creare il repository e con il comando

git init

verrà creata una cartella .git (inizia con il punto poichè su Unix tali cartelle sono nascoste).

Verificare di essere sul branch master con il comando

git status

Cosa **VA** tracciato con git:

- Sorgenti
- Risorse
- Librerie
- File esterni quali README.md, file per la licenza, il file .project per facilitare un lavoro di un team che utilizza Eclipse

Cosa **NON VA** tracciato con git:

- Binari
- Documentazione rigenerabile dai sorgenti
- Archivi rigenerabili

Per non tracciare file e/o cartelle creare il file .gitignore con all'interno un elenco:

bin/
doc/
.log
.pdf

Per aggiungere le modifiche fatte su un file (compresa la cancellazione del file stesso) allo staging area lanciare:

git add PATH_TO_FILE

Nel caso di più file:

git add PATH_TO_FILE_1 PATH_TO_FILE_2 PATH_TO_FILE_3

Se ci si accorge di avere fatto un errore e di voler togliere uno o più file dallo stage usare il comando:

git reset PATH_TO_FILE

Nel caso di più file:

git reset PATH_TO_FILE_1 PATH_TO_FILE_2 PATH_TO_FILE_3

Una volta messo il file sulla staging area si potrà fare il commit:

git commit -m "QUI INSERIRE IL TESTO DEL COMMIT"

Riverificare lo stato del repository:

git status

GOOD PRACTICES:

- Molti commit
- Piccola dimensione
- Messaggio breve ma significativo

- Verificare continuamente lo stato del repository

Come visualizzare la storia dei commit:

Per visualizzare tutti i commit della linea di sviluppo corrente:

git log

Per visualizzare tutti i commit della linea di sviluppo corrente ma con una visualizzazione grafica:

git log --graph

Per visualizzare tutti i commit di tutti i branch con una visualizzazione grafica:

git log --all --graph

Capitolo 3

Navigare nel repository

Se vogliamo vedere le modifiche intercorse fra due commit usiamo il comando diff. Per riferirci ad un commit usiamo il suo codice hash (bastano le prime 7 cifre. Per ottenerlo cercarlo nello storico con git log)

Per mostrare le differenze fra il working tree e l'ultimo commit:

git diff

Per mostrare le differenze fra il working tree e un commit specifico:

git diff FROM

es. git diff 07388467

Per mostrare le differenze fra due commit specifici:

git diff FROM TO

es. git diff 0738847 7736421

Per semplificare l'accesso agli ultimi commit possiamo riferirci ad essi usando HEAD ∼N dove N è il numero quanti commit vogliamo spostarci all'indietro. Per esempio se vogliamo spostarci indietro di 3 commit lanciamo il comando:

git diff HEAD∼3

Per spostarsi invece fisicamente su di un commit si usa il comando checkout:

git checkout COMMITREF

Dove COMMITREF può essere:

- L'hash completo di un commit
- Almeno le prime 7 cifre di un commit
- HEAD∼2
- master
- il nome di un altro branch (ciò fa sì che ci si sposta alla fine di quel branch)

A quel punto si è in una modalità chiamata Detached Head (testa staccata). Ogni commit fatto in quel punto verrà scartato. Per tornare in modalità attached bisogna tornare alla HEAD del branch:

es. git checkout master

Con il comando checkout si possono anche recuperare le modifiche solo per alcuni file a mia scelta fatte in un certo commit:

git checkout COMMITREF -- FILENAME

Se per esempio voglio ripristinare lo stato del solo file pippo.txt allo stato che aveva 2 commit fa:

git checkout HEAD∼2 -- /pippo.txt

Capitolo 4

Creare un nuovo branch e unire due branch

Il sottocomando checkout consente anche di creare un nuovo branch e di spostarcisi automaticamente:

git checkout -b NEWBRANCHNAME

I commit seguenti saranno perciò aggiunti al nuovo branch.
Per visualizzare tutti i branch lanciare:

git branch

La HEAD identifica la posizione corrente all'interno della storia del repository. In una normale sessione la HEAD è posizionata alla fine del branch. Introduciamo il concetto di fusione di due branch. Il comando per ottenere ciò è merge.

Bisogna prima spostarsi sul branch in cui voler introdurre le modifiche e poi usare il comando:

git merge BRANCHNAME

Dove BRANCHNAME è il nome del branch che si vuole unire al branch su cui si è attualmente

Se non ci sono conflitti tutti i commit di BRANCHNAME vengono aggiunti al branch corrente.

Viene creato un nuovo commit di default (che è buona norma non modificare) con l'editor di default.

Eventualmente si può ora cancellare il branch "secondario" con:

git branch -d BRANCHNAME

Attenzione: viene cancellata solo l'associazione del nome a quel branch. Nello storico il branch è ancora visibile. Ora possiamo dare lo stesso nome ad un nuovo branch.