Guida rapida per i comandi Git

Luca Morlino

3 marzo 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 dal punto in cui ci si trova 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.

4.1 Risolvere i merge conflict

Nel caso in cui due linee di sviluppo abbiano modificato concorrentemente un file nello stesso punto, il merge non è banale da effettuare ma occorre risolvere manualmente un merge conflict. Una volta modificati i file che confliggono occorre aggiungerli allo stage e poi fare il commit (il commit di default viene auto-generato).

Ricordarsi di tracciare le cose corrette. Non i file rigenerabili, in quanto provocano merge conflict difficili da risolvere.

Capitolo 5

Git in rete

5.1 Clonare un repository

Per importare una copia di un repository esistente lanciare il comando:

git clone URI localfolder

Il comando scarica l'intera storia del repository conservato in URI all'interno di localfolder(che deve essere vuoto) che diventa un nuovo repository Git. Se localfolder non viene segnalato allora Git crea una cartella con l'ultimo nome del path.

5.2 Repository remoto

E' utile ricordare un indirizzo con un nome simbolico. Per conoscere i repository remoti configurati usare il comando:

git remote -v

Per aggiungerne uno nuovo:

git remote add name url

es. git remote add serverAziendale www.github.com/pippo/repo

Per cancellarne uno esistente:

git remote rm name

Ogni branch sulla nostra macchina può essere configurato per avere un upstream. In pratica il branch locale è la copia del branch di un repository remoto. Per settare ciò usare il comando:

git branch -u remoteName/remoteBranch

D'ora in poi il branch locale corrente fara' push e pull dal Branch remoteBranch.

Nel caso il repository sia stato clonato e non inizializzato un riferimento al repository remoto di origine viene automaticamente inserito e configurato come upstream per tutti i branch col nome di origin.

Al momento del clone Git scaricherà solo il branch primario. Per sapere quali sono tutti i branch in remoto usare il comando:

git branch -a

Qualora si voglia importare in locale un branch remoto per lavorarci occorre creare un "tracking branch" locale:

git checkout -b localBranchName remoteName/remoteBranchName

es. git checkout -b miaLineaSvliuppo origin/developBranch

5.3 Aggiornamento dei nuovi commit

Vogliamo sapere se su un certo branch ci sono dei nuovi commit fatti da altri utenti. Il sottocomando fetch ci permette di aggiornare le informazioni:

git fetch URI branchName

oppure

git fetch remoteName branchName

oppure qualora ci sia un branch upstream configurato

git fetch

Dopo aver fatto la fetch occorre fare la merge con "git merge" e rislovere gli eventuali conflitti Il sottocomando:

git pull

permette di fare fetch e merge in un colpo solo. Quindi perchè fare la fetch? Qualora si volesse vedere gli aggiornamenti di un branch senza effettuare poi il merge.

La sintassi è la stessa del comando fetch.

git pull URI branchName

oppure

git pull remoteName branchName

oppure qualora ci sia un branch upstream configurato

git pull

Il comando speculare a pull è push. Permette di "spingere" i nuovi commit su un branch se non ci sono conflitti.

La sintassi è la stessa del comando pull. La push va a buon fine solo se c'è corrispondenza tra tutti i commmit sul locale e sul remoto.

git push URI branchName

oppure

git push remoteName branchName

oppure qualora ci sia un branch upstream configurato

git push