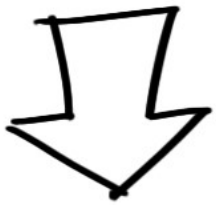
rogerdudler.github.io

git - la guida tascabile

5-7 minuti

Solamente una piccola guida per iniziare con git. Niente di complicato ;)

by Roger Dudler (translation by @stecb)
credits to @tfnico, @fhd, Namics
this guide in english, deutsch, español, français, indonesian, nederlands, polski, português, русский, türkçe, မြန်မာ, 日本語, 中文, 한국어



creazione di un nuovo repository

crea una nuova directory, entraci ed esegui

```
git init
```

per creare un nuovo repository git.

checkout di un repository

crea una copia di un repository locale eseguendo il comando

```
git clone /percorso/del/repository
```

usando invece un server remoto, il comando sarà

```
git clone nomeutente@host:/percorso/del/repository
```

aggiungere & validare

Puoi proporre modifiche (aggiungendole all'**Index**) usando

```
git add <nomedelfile>
```

```
git add *
```

Questo è il primo passo nel flusso di lavoro in git. Per validare queste modifiche fatte si usa

```
git commit -m "Messaggio per la commit"
```

Ora il file è correttamente nell'**HEAD**, ma non ancora nel repository remoto.

invio delle modifiche

Quello che hai cambiato ora è nell'**HEAD** della copia locale. Per inviare queste modifiche al repository remoto, esegui

```
git push origin master
```

Cambia *master* nel branch al quale vuoi inviare i cambiamenti.

Se non hai copiato un repository esistente, e vuoi connettere il tuo repository ad un server remoto, c'e' bisogno che tu lo aggiunga con

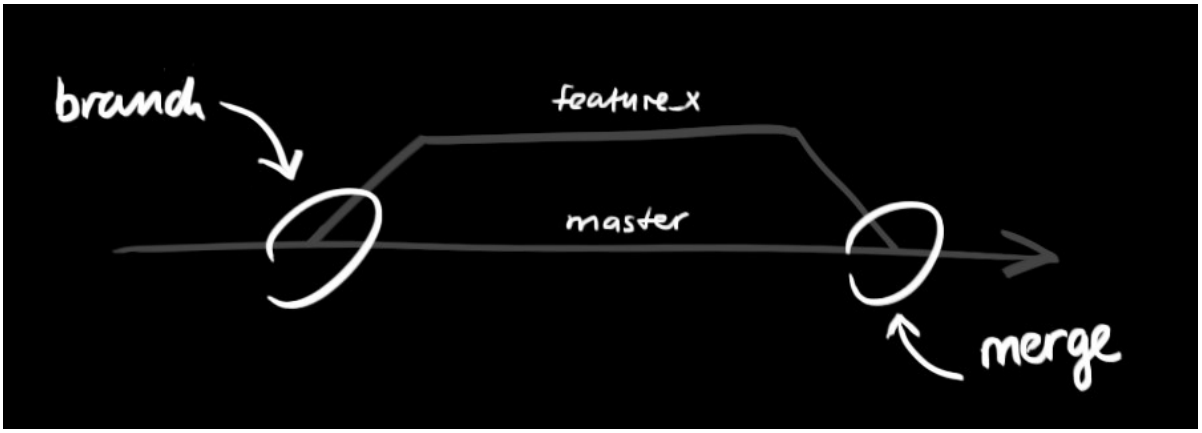
```
git remote add origin <server>
```

Ora sarai in grado di inviare le tue modifiche al server remoto specificato

branching

I branch ('ramificazioni') sono utilizzati per sviluppare features che sono isolate l'una dall'altra. Il branch *master* è quello di default quando crei un repository. Puoi usare altri branch per lo sviluppo ed

infine incorporarli ('merge') nel master branch una volta completati.



crea un nuovo branch chiamato "feature_x" e passa al nuovo branch usando

```
git checkout -b feature_x
```

ritorna di nuovo su master

```
git checkout master
```

e cancella il branch creato in precedenza

```
git branch -d feature_x
```

il branch non sarà disponibile agli altri fino a quando non verrà inviato al repository remoto

```
git push origin <branch>
```

aggiorna & incorpora

per aggiornare il tuo repository locale alla commit più recente, esegui

```
git pull
```

nella tua directory corrente per fare una *fetch* (recuperare) ed incorporare(*merge*) le modifiche fatte sul server remoto.
per incorporare un altro branch nel tuo branch attivo (ad esempio master), utilizza

```
git merge <branch>
```

in entrambi i casi git prova ad auto-incorporare le modifiche. Sfortunatamente, a volte questa procedura automatizzata non è possibile, ed in questo caso ci saranno dei *conflitti*. Sei tu il responsabile che sistemerà questi *conflitti* manualmente modificando i file che git mostrerà. Dopo aver cambiato questi files, dovrai marcarli come 'correttamente incorporati' tramite

```
git add <nomedelfile>
```

prima di immettere le modifiche, potrai anche visualizzarne un'anteprima eseguendo

```
git diff <branch_sorgente> <branch_target>
```

tags

È raccomandato creare dei tags nel caso in cui il software venga rilasciato. Questo è un concept già conosciuto, che esiste anche in SVN. Puoi creare un tag chiamato *1.0.0* eseguendo

```
git tag 1.0.0 1b2e1d63ff
```

la sequenza *1b2e1d63ff* sta per i primi 10 caratteri del commit che si vuol referenziare tramite questo tag. Puoi ottenere l'id della commit tramite

```
git log
```

puoi anche utilizzare meno caratteri per l'id della commit, basta che sia unico.

sostituire i cambiamenti locali

Nel caso tu abbia fatto qualcosa di sbagliato (ma non capita mai, sicuro ;) puoi sostituire i cambiamenti fatti in locale con il comando

```
git checkout -- <nomedelfile>
```

questo rimpiazza le modifiche nell'albero di lavoro con l'ultimo contenuto presente in HEAD. I cambiamenti fatti ed aggiunti

all'index, così come i nuovi files, verranno mantenuti.

Se vuoi in alternativa eliminare tutti i cambiamenti e commits fatti in locale, recupera l'ultima versione dal server e fai puntare il tuo master branch a quella versione in questo modo

```
git fetch origin
git reset --hard origin/master
```