git - la guida tascabile

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

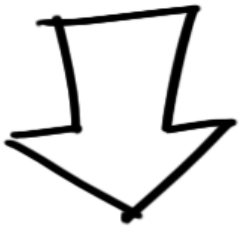
Tweet

by Roger Dudler (translation by @stecb)

credits to @tfnico, @fhd, Namics

ı english, deutsch, español, français, indonesian, nederlands, polski, português, русск

မြန်မာ, 日本語, 中文, 한국어



installazione

Scarica git per OSX

Scarica git per Windows

Scarica git for Linux

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

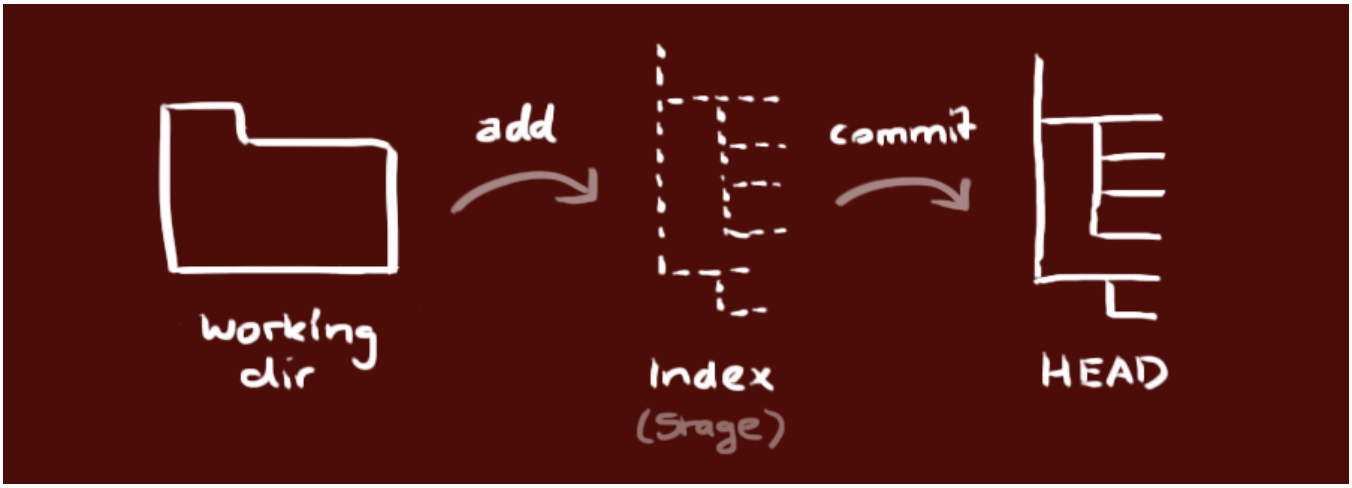
```
git clone /percorso/del/repository
```

usando invece un server remoto, il comando sarà

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

ambiente di lavoro

la tua copia locale del repository è composta da tre "alberi" mantenuti da git. Il primo è la tua `Directory di lavoro` che contiene i files attuali. Il secondo è l' `Index` che fa da spazio di transito per i files e per finire l' `HEAD` che punta all'ultimo commit fatto.



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

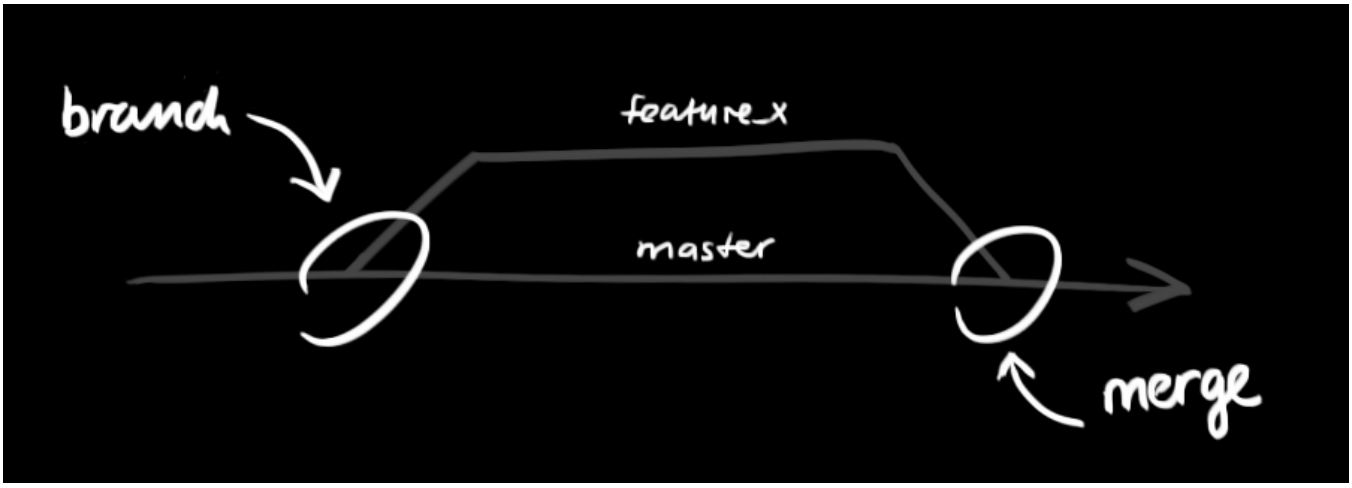
```
git remote add origin <server>
```

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
```

suggerimenti utili

GUI (Interfaccia utente grafica) per git disponibile di default

```
gitk
```

colora gli output di git

```
git config color.ui true
```

mostra il log in una riga per commit

```
git config format.pretty oneline
```

utilizza l'aggiunta interattiva

```
git add -i
```

links & risorse

clients grafici

GitX (L) (OSX, open source)
Tower (OSX)
Source Tree (OSX, free)
GitHub per Mac (OSX, free)
GitBox (OSX)

le guide

Git Community Book
Pro Git
Think like a git
GitHub Help
A Visual Git Guide

commenti

CiccioFormaggio • a year ago
Grazie a te bnon verrò licenziato, la fai una guida galattica per gittisti?
3 • Reply • Share ›

Federico Pasinetti • 2 years ago
Ho scaricato in locale il penultimo commit del mio progetto, come posso renderlo master? se lo committo mi dice che tutto è già aggiornato e non me lo fa fare. Io invece vorrei commitare questa versione come fosse la master date che l'ultimo commit che ho fatto ha dei bug.
2 • Reply • Share ›

Giuseppe Filomena • 2 years ago
bellissima guida semplice e utile, per non parlare dei font. :D Grazie
• Reply • Share ›

Lorenzo Istorn Neri • 2 years ago
Grazie a te passerò l'esame (spero) hahahahah :D
• Reply • Share ›

Sandro • 2 years ago
Complimenti, è una bellissima guida pratica.
• Reply • Share ›

Gianluca • 2 years ago
Ma l'invio delle modifiche consiste nel fare una pull request? Cioè se lancio il comando "git push origin master" l'admin del progetto se le ritiene corrette mi conferma le modifiche?
• Reply • Share ›

Giuseppe • 3 years ago
Guida molto interessante,

ma come posso fare a mergere due lavori che ho sullo stesso pc, e poi inviare le modifiche al repository principale?
• Reply • Share ›

Fabrizio • 3 years ago
Semplice ed essenziale! Grazie!
• Reply • Share ›

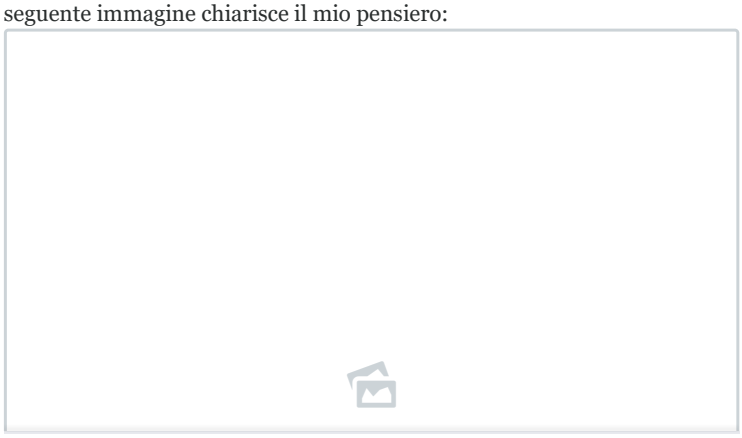
Avatar This comment was deleted.

No-N00bs-Allowed ➚ Guest • 3 years ago
magari piantatela voi di avere l'arroganza di pensare che tutto sia semplice e immediato. La guida e' abbastanza semplice, chiara, e immediata. Se poi l'utente che la legge trova questa guida complicata, un bel "man git" e passa la paura. Se invece non si capisce nemmeno cosa sia "man" allora cambiate mestiere ;)
PS: *o scrivi una guida
3 • Reply • Share ›

Avatar This comment was deleted.

Supermenn ➚ Guest • 3 years ago

Chiara per chi già conosce git e magari voleva scrivere una guida, ma chi vorrebbe imparare da zero si trova davanti ad un elenco di comandi talvolta spiegati in un gergo incomprensibile.
Non è solo problema di questa guida ma un po' di tutto il mondo git; la seguente immagine chiarisce il mio pensiero:



see more

4 ^ | v • Reply • Share ›

Avatar This comment was deleted.

Supermenn ➚ Guest • 3 years ago

Comunque in breve, per far vedere al mondo le tue modifiche:
1. "presenti" le tue modifiche alla tua copia locale del repositorium (è invisibile, si chiama "origin"):
add
2. Aggiungi le tue modifiche alla tua copia locale (origin)
commit -m "commento"
-m + "commento" SONO OBBLIGATORI, visto che se non li aggiungi ti apre un file dove scrivere il "commento"

Ancora il mondo non sa assolutamente nulla dei tuoi file. Tutto è nella tua origine, per cui dai il comando
push → ramoPubblico

che farà sapere al mondo le tue modifiche. Anche se bisognerà fare il merge (che in SVN è ovviamente pull)

Note
1: quando Linus T. ha scelto i nomi per i comandi, forse era troppo stanco e comunque non ci ha pensato. Confondono chi conosce sistemi simili.
2: In SVN+Tortoise per fare tutti quei comandi, basta cliccare sul/i file modificato/i o la cartella e selezionare "commit"
3: nei comandi che ti ho dato la sintassi non è completa perché io ho dei file dove copincollo e ora non li ho sotto mano. In questo senso è utile questa guida.

1 ^ | v • Reply • Share ›

cristianpozzessere ➚ Supermenn • 3 years ago

Grazie mille sto usando smart git ,che ha anche una gui ,non ha bisogno di installazione.comunque mi sono messo a leggermi la guida ufficiale...alla fine ho sottolineato solo le parti interessate..naturalmente cosi facendo al primo problema avrò delle noie,sicchè ho richiesto il libro.così da tenermelo sempre sottomano. Grazie ancora per la rispostaa.
se vuoi dare un occhiata al mio lavoro lo trovi qui :Lila-HD icon theme : https://github.com/ilnanny/...

• Reply • Share ›

Supermenn ➚ Guest • 3 years ago

Ma git ti serve se lavori con altri. Se il progetto è piccolo, prova SVN che è più semplice e se usi Windows, installa l'applicazione TortoiseSVN e fai tutto col mouse. Esiste anche Tortoise per GIT ma può far fare strafalcioni, rischi di far danno. È utile ma non sostituisce la riga di comando, e se non usi la console non imparerai mai git.
Se invece vuoi/devi usare GIT, la guida più facile che ho trovato è quella del partito pirata. Non so cosa intendi per facile, GIT è un sistema molto complicato, anche molto potente, ma è difficile anche fare le cose semplici. Dopo che l'avrai capito, allora ti sembrerà facile. Ma fino a quel punto, il rischio di far danni è sempre dietro l'angolo, e rischi di perdere il 40% del tempo a capire come usare git, e il 60% per programmare.
Riassumendo: prova SVN (che io ho imparato DA SOLO in 3 minuti grazie a Tortoise, che si integra nell'esploratore di Windows), se invece devi usare GIT hai la guida del partito pirata ma usa anche altre fonti, inoltre assicurati di capire

see more

2 • Reply • Share ›

Digodago ➚ Supermenn • a year ago

Credo che chiunque utilizzi un prodotto (o piu' in generale una cosa) che non conosce, utilizzandolo male, puo' fare danni, ma non è colpa del prodotto, ma sempre dell'utilizzatore.

• Reply • Share ›

Supermenn ➚ Digodago • a year ago

Nei quiz della scuola guida, quando nelle affermazioni c'è scritto "sempre, mai ... " si può chiudere gli occhi e segnarle false.

Il prodotto in questione è stato implementato in 2 settimane. Lo pubblicizzano così, eppure è una cosa molto negativa che Linus non abbia chiesto pareri per rendere più semplice la cosa.
Lui sarà ormai inglese di madrelingua ed ha dimenticato il finlandese, ma non c'è bisogno di inventare una nuova parola per ogni operazione. Per esempio le operazioni che fanno cose opposte potrebbero essere rese in modo logico, esempio stupido:
git -b ramo1 //creo ramo1
git undo -b ramo1 //cancello ramo1
la parola undo potrebbe essere riciclata per molti comandi; invece no, ogni operazione ha una parola diversa. Il finlandese è una lingua agglutinante e usa prefissi per creare nuove parole, se Linus non lo avesse dimenticato avrebbe creato un sistema simile, cioè l'uso di parti "riciclabili" come undo nel mio esempio, che si imparano una volta e si usano senza dover imparare una parola nuova, un poco come gli affissi del finlandese.
Ma ha voluto fare tutto in due settimane e vantarsene senza una revisione critica che avrebbe reso più semplice il prodotto.

4 • Reply • Share ›

cristianpozzessere ➚ Supermenn • 3 years ago

in realtà uso solo Debian Gnu/Linux (dallaversione 4.0 "Etch" del 2007-) di conseguenza il terminale , per fare la maggior parte delle operazioni (abitudine) ma non avevo voglia di imparare git per intero anche perchè la guida è lunga ...e io divento vecchio prima di caricare un pacchetto

di icone heheh.Comunque alla fine ho letto un pò di ciò che mi interessa per far sì che anche gli altri possano partecipare al piccolo progetto e magari farlo diventare più corposo.
Grazie comunque per la risposta.
Reply • Share ›

Avatar This comment was deleted.

Supermenn ➚ Guest • 3 years ago
Le stesse critiche ci sono nella versione in altre lingue.

E se la critica è costruttiva aiuta a sviluppare il progetto con ciò che manca.
Reply • Share ›

Supermenn • 3 years ago
Manca una cosa fondamentale, qui si suppone che io voglia creare un file e poi mandarlo. E se ho un file che ho clonato (da un repo già esistente) e voglio inviare le modifiche affinché siano visibili al mondo, a quale punto della guida devo agganciarmi?
(si potrebbe precisare la traduzione, es. branch = ramo; ramificazione = branching; to branch = ramificare -- ottimo invece merge = incorporare, non mi era mai venuto un termine adatto)
Reply • Share ›

Afro Games ➚ Supermenn • 3 years ago
devi innanzitutto forkare la repo sul tuo profilo github, poi la cloni in locale, e poi fai un commit alla tua repo forkata...
non puoi fare commit sulla repo di un altro ;)
Reply • Share ›

Supermenn ➚ Afro Games • 3 years ago
forkare ?? tradotto in git sarebbe?
Reply • Share ›

Avatar This comment was deleted.

Supermenn ➚ Guest • 3 years ago
cioè il comando:
git fork blablabla
vabbè ho capito, inutile insistere
Reply • Share ›

Francesco • 4 years ago
Una domanda .Installato GitHub windows in locale posso aggiungere e validare delle modifiche o devo farlo utilizzando un server remoto?
Reply • Share ›

antonio vangi • 4 years ago
Ottima guida, complimenti.
1 Reply • Share ›

Luca Corna • 4 years ago
Una volta fatto il tag (git tag 1.0 23y8w2f4rj per esempio), è necessario fare il push:

git push origin 1.0 (nel caso di server remoto)
Reply • Share ›

stefano • 4 years ago
Una guida ECCEZIONALE! :)
Grazie
1 Reply • Share ›

Flavio • 4 years ago
Questa guida forse e' utile a chi git lo conosce gia'. Io sto cercando di imparare a usarlo, e non mi e' servita a niente.
2 ^ | v • Reply • Share ›

Avatar This comment was deleted.

Afro Games ➚ Guest • 3 years ago
Giustissimo ;)
^ | v • Reply • Share ›

Gian Marco • 4 years ago
Qualcuno saprebbe consigliarmi una guida più semplice per una persona che non sa assolutamente nulla su git?
^ | v • Reply • Share ›

Admin ➚ Gian Marco • 4 years ago
Vero io non so' nulla e non ho capito nulla la colpa e' del admin che non vuole che gli atri ci capiscano :-(
^ | v • Reply • Share ›

Afro Games ➚ Admin • 3 years ago
no, semplicemente è che devi sapere che cos'è il controllo versione
^ | v • Reply • Share ›

Antonio Scigliano • 4 years ago
Ottima guida
1 ^ | v • Reply • Share ›

ProceduraGuidata79 • 4 years ago
Odio git, odio la mente perversa che l'ha inventato. Arrivo da Subversion e capisco benissimo i vantaggi teorici di un DVCS, però usare git mi fa rimpiangere ogni svantaggio ed ogni limitazione di SVN...
1 ^ | v • Reply • Share ›

Supermenn ➚ ProceduraGuidata79 • 3 years ago
Prova TortoiseGit, dovrebbe essere simile a TortoiseSVN ma non l'ho provato.
Git l'ha inventato lo stesso che ha inventato il kernel linux (Linus Torvalds), quindi la semplicità non è il suo forte...
^ | v • Reply • Share ›

ProceduraGuidata79 ➚ Supermenn • 3 years ago
TortoiseGit lo uso su Windows, purtroppo su Linux non c'è una GUI decente...
^ | v • Reply • Share ›

Mush • 5 years ago
è un memorandum non una guida. Se si conosce GIT è senz'altro utile. Non siate severi: esistono molte guide in rete.
1 ^ | v • Reply • Share ›

Luca ➚ Mush • 4 years ago
Hai qualche consiglio per neofiti? Questi sono strumenti scritti da NERD per.... NERD ;-) Pare che se non si fanno le cose complicate, non si è contenti.
^ | v • Reply • Share ›

Egg1234 • 5 years ago
Non è una guida adatta a neofiti. Almeno spiegate com'è composto un ambiente di sviluppo in modo da far capire cosa serve per realizzarlo e come gestirlo! La cosa che in tanti non capite e che date per scontato che chi legge sappia a priori di cosa si parla. Non tutti hanno un'informatura sull'argomento. Per Cinque!

cosa si parla. Non tutti hanno un' infarinatura sull argomento. Per Giove!
10 • Reply • Share ›

Afro Games ➚ Egg1234 • 3 years ago
qui si cerca di spiegare i comandi di git, un ambiente di sviluppo è tutta un altra cosa ;)
• Reply • Share ›

bcclsn • 5 years ago
perché clonando il mio repo mi scarica anche la cartella .git che dovrebbe invece essere nascosta e non nel repo? da web non la vedo ma se clono il repo, viene scaricata...
• Reply • Share ›

siscia ➚ bcclsn • 5 years ago
La cartella .git ti serve. Altrimenti come fai ad usare git in locale (esempio, cambiare repo oppure mandare le modifiche che fai in remoto ?).

Comunque non sono certo che venga scaricata dal repo, penso sia creata da git stesso in locale, ma non sono certo.
1 • Reply • Share ›

bcclsn ➚ siscia • 5 years ago
si, so che mi serve xD però scaricando il mio repo mi scarica anche la cartella... che nel repo non vedo.
idem se scarico la cartella da un pc "diverso". la cartella .git viene creata sempre... ma se accedo da web non la vedo.
se scarico i repo di altri ragazzi, non vedo nessuna cartella .git.
quindi: dove sbaglio?

per intenderci: quella cartella .git contenuta in dotfiles non dovrebbe esserci. io non l'ho uppata/inclusa nel repo e soprattutto, se vado su https://github.com/username... non la visualizzo.
why?! :grrrr:
• Reply • Share ›

siscia ➚ bcclsn • 5 years ago
Ciao, non sono certo di aver capito, comunque stai con e e seguimi.

https://gist.github.com/sis...

Non vedi nessuna cartella .git su github, immagino siano nascote (non ne sono certo però.)

Anche: http://stackoverflow.com/qu...
1 • Reply • Share ›

bcclsn ➚ siscia • 5 years ago
ok, risolto... ricordavo male io :(
grazie mille per l'aiuto e il tempo dedicatomi :)
• Reply • Share ›

alexforti • 5 years ago

bella questa idea di una guida come sequenza di "finte" slides... oltre naturalmente al contenuto, ineccepibile

1 ^ | v • Reply • Share ›

Load more comments