Introduzione a GIT

Corsi Linux Base 2018

Lorenzo Prosseda

lerokamut@gmail.com

10 ottobre 2018

POLITECNICO OPEN
unix LABS

Perché mantenere ogni versione

Quando lavoriamo su uno o più file apportando delle modifiche, accade spesso di volerne annullare qualcuna in particolare...

- scorciatoia `CTRL + Z`
- copiare i file aggiungendo un timestamp al nome
- usare un version control system

Perché mantenere ogni versione

Quando lavoriamo su uno o più file apportando delle modifiche, accade spesso di volerne annullare qualcuna in particolare...

- scorciatoia `CTRL + Z`
- copiare i file aggiungendo un timestamp al nome
- usare un version control system

Perché mantenere ogni versione

Un VCS mantiene una base di dati che conserva le modifiche effettuate sui file tracciati

- locale: i file tracciati risiedono sulla propria macchina
- centralizzato: i file tracciati risiedono solamente su un server centrale
- distribuito: la base di dati e i file tracciati risiedono sulle macchine di ciascun utente

Perché mantenere ogni versione

GIT è un VCS decentralizzato, che tiene traccia delle modifiche ai file tramite snapshot

- la maggior parte delle operazioni sono svolte in locale
- ogni modifica viene salvata con un controllo di integrità
- difficile «convincere» GIT a cancellare delle informazioni

Usare un repository GIT

Una directory i cui file siano tracciati da GIT è chiamata repository; per crearne una locale basta solo:

- spostarsi nella directory radice del progetto che si vuole tracciare
- `$ git init`
- ... tutto qui!

Usare un repository GIT

Una directory i cui file siano tracciati da GIT è chiamata repository; per crearne una locale basta solo:

- spostarsi nella directory radice del progetto che si vuole tracciare
- `$ git init`
- ... tutto qui!

Hey, se volessi configurare cose?

GIT salva le proprie impostazioni in tre aree del file system; possiamo modificarle col comando:

```
$ git config <--local|--global|--system>
```

Usare un repository GIT

Configurare GIT:

- `$ git config <scope> user.name <name>`
  - Imposta il nome utente che sarà associato alle modifiche
- `$ git config <scope> user.email <mail>`
  - Imposta anche una mail da associare a un nome utente
- `$ git config <scope> core.editor <editor>`
  - Scegli l'editor predefinito da usare con GIT

Usare un repository GIT

Configurare GIT:

- `$ git config <scope> user.name <name>`
  - Imposta il nome utente che sarà associato alle modifiche
- `$ git config <scope> user.email <mail>`
  - Imposta anche una mail da associare a un nome utente
- `$ git config <scope> core.editor <editor>`
  - Scegli l'editor predefinito da usare con GIT

Inoltre, può essere utile controllare l'output dei seguenti comandi:

- `$ git help <verb>`
  - Mostra una pagina di aiuto per una specifica funzione di GIT
- `$ git config --list`
  - Mostra una lista di impostazioni con i relativi valori

Usare un repository GIT

Se il repository fosse online?

- `$ git clone <repo-address>`

in questo modo verrà copiato sulla propria macchina anche lo storico delle modifiche di tutti i file del repository

Modifiche e staging area

Ho capito come creare il mio repository locale o come scaricarlo, ma come faccio a tenere traccia delle modifiche?

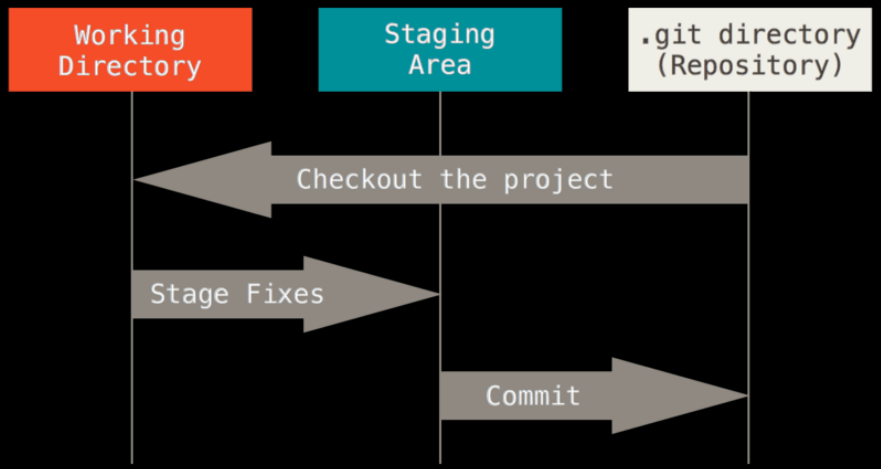
Stato dei file nel repository

GIT gestisce i file permettendo all'utente di assegnargli uno stato, nei modi seguenti:

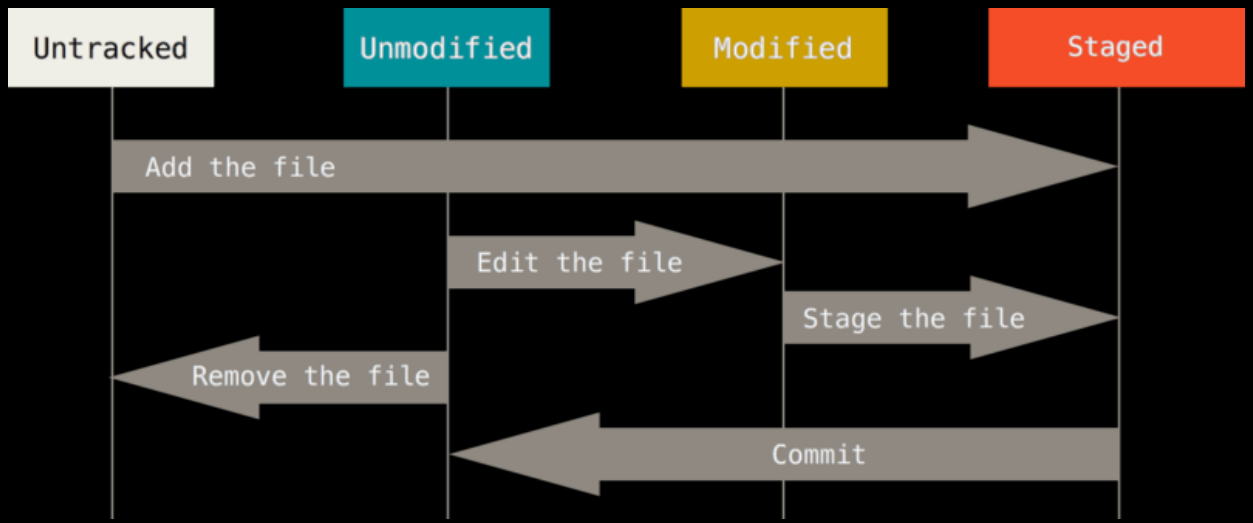
- staged: ho chiesto a GIT di «mettere da parte» lo snapshot di un file col comando `$ git add <file>`
- modified: ho chiesto a GIT di tracciare un file con `$ git add <file>`, dopodiché ho modificato quel file
- committed: ho chiesto a GIT di prendere il contenuto della staging area e di spostarlo nella cartella `.git`, tramite il comando `$ git commit`

Modifiche e staging area

Aree e stato dei file



Ciclo di vita di un file tracciato



Commit

Per salvare permanentemente gli snapshot dei file contenuti nella staging area, si usa il comando `$ git commit`

- per prima cosa si aprirà l'editor di testo per scrivere un messaggio *significativo* da associare al commit
- gli snapshot nella staging area saranno spostati nella cartella `.git`, e i file associati saranno marcati come «unmodified»

Possiamo controllare in ogni momento lo stato dei file tracciati e nella staging area

- `$ git status`: mostra lo stato dei file tracciati, indicando le modifiche e il contenuto della staging area (così sapremo cosa finirà nel commit)
- `$ git diff <--staged>`: mostra in modo specifico le differenze dei contenuti dei file «modified» ma non «staged» rispetto all'ultimo commit

Commit

Il mio editor genera millanta file di configurazione e log che non voglio includere nei commit: GIT permette di gestire una lista di esclusioni sotto forma di file di testo

File di esclusioni `.gitignore`

- si tratta di un file di testo che supporta la sintassi glob[1]
- va posto nella directory radice del repository
- GitHub mantiene una lista[2] di file `.gitignore` standard, adatti ai principali casi d'uso

[1] Glob Pattern [Wikipedia]
[2] Lista di gitignore [GitHub]

Commit

Poffarbacco! Per sbaglio ho incuso nel mio commit un file inutile, che non ho specificato nel `.gitignore`!

- prima rimuovo il file dal controllo di versione, rimuovendolo dalla staging area e dal disco, grazie al comando `$ git rm <file>`
- adesso uso la mia staging area attuale al posto dell'ultimo commit effettuato, tramite il comando `$ git commit --amend` (non creo un nuovo commit)
  - *Pro tip*: posso usare questo comando senza modificare la staging area, solamente per modificare il messaggio dell'ultimo commit

E se invece volessi solo rimuovere un file dalla staging area, ma non dal file system?

- uso il comando `$ git rm --cached <file>`

Commit

Come posso gestire e utilizzare la cronologia dei commit?

- tramie il comando `$ git log` posso consultare la cronologia dei commit relativi al repository in cui il comando viene invocato
- tramite il comando `$ git reset <sha-1>` posso riportare la staging area allo stesso stato del commit identificato dallo SHA-1 specificato
- tramite il comando `$ git revert <sha-1>` posso annullare le modifiche legate solamente al commit indicato dallo SHA-1 specificato

Annullare un commit indecente

Tramite il comando `$ git reset --hard <sha-1>` non solo la staging area, ma anche i file sul disco saranno sovrascritti da quelli salvati tramite il commit di riferimento; da usare con cautela!

Gestire repositoy remoti

GIT non era un VCS distribuito?

Gestire repositoy remoti

GIT non era un VCS distribuito?

Possiamo associare al nostro progetto uno o più repository remote scegliendo di caricare o scaricare dati da una o più di esse

- col comando `$ git remote -v` si ottiene una lista di repository remoti associati e i rispettivi URL
- col comando `$ git remote add <shortname> <url>` si aggiunge un repository remoto al progetto corrente
- col comando `$ git remote rm <shortname>` si rimuove un repository remoto associata

Tracking Branch

GIT usa in ogni repositoy degli aggeggi chiamati *branch*: se avete appena aggiunto un remote in cui volete caricare i commit, dovete dire a GIT di usarlo come upstream col comando

```
$ git push --set-upstream shortname branchname
```

Push e pull

Dopo aver associato dei repository remoti a un progetto, possiamo caricare o scaricare modifiche da e verso di essi:

- col comando `$ git pull <remote>` scarichiamo dal repository remoto i commit che non abbiamo in locale
- col comando `$ git push <remote>` carichiamo le modifiche al progetto sul repository remoto

Conflitti e merging

Nel caso in cui due commit contengano modifiche allo stesso file, bisogna scegliere manualmente la modifica da tenere prima di poter caricare o scaricare entrambi i commit nello stesso repository

Push e pull

Per mille schede madri: chi è quella testa di fagiolo che ha introdotto il bug?!

- Faccio pull - commit - push per caricare le modifiche che ho fatto sulla repository
- Quando eseguo di nuovo il codice il compilatore segnala un errore alla riga X, in un file che non ho toccato

Push e pull

Per mille schede madri: chi è quella testa di fagiolo che ha introdotto il bug?!

- Faccio pull - commit - push per caricare le modifiche che ho fatto sulla repository
- Quando eseguo di nuovo il codice il compilatore segnala un errore alla riga X, in un file che non ho toccato

GIT ha il comando per «incolpare»

Col comando `$ git blame <-L riga_ini,riga_fin> file` viene invocato lo strumento per visualizzare le annotazioni dei file: esso mostrerà esattamente a chi appartengono i commit che contengono le modifiche per ciascuna riga di un file (o un intervallo di righe specificato).

Push e pull

Facciamo tira e molla coi commit: **demo**!

- Creo un repository locale che vorrei condividere
- Clono un repository e inizio a lavorarci sopra

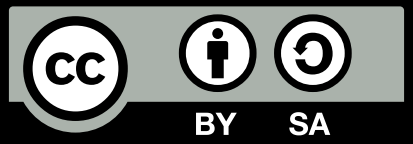
URL del repository che sto usando (dal GitLab del POuL):
`corsi/linuxBase/gtk-counter.git`

Fonti

- Manuale Pro GIT
- Slide corso GIT 2017 POUL
- Video corso GIT 2017 POUL
- Manuale Linux per GIT

Fine

Grazie per l'attenzione!



Queste slide sono licenziate Creative Commons Attribution-ShareAlike 4.0

`http://www.poul.org`