Riassunto comandi latex

Matteo Mistri

26 ottobre 2017

Indice

Capitolo 1

Teroia

Ho tre livelli:

- Working Directory
- Staging Area
- git directory/Repository

Con lo stage dei file passo da WD a SA, con il commit da SA a Repository
Con il checkout passo da Repository a WD.

Capitolo 2

Creazione repository

Creo una cartella in locale, apro la finestra del terminale e digitare

```
git init
```

si creerà una cartella .git, che contiene file di sistema. Dopodiche copio i file che voglio nella directory e uso il comando

```
git add nomefile
```

posso sostituire il nome file con *.estensione per includere tutti i files con la estensione indicata

2.1 Commit

Per esegiore il commit delle modifiche, utilizzo il comando

```
commit -m 'Messaggio della commit'
```

Se aggiungo l'opzione -a, committo tutto quello che è stato modificato, anche se non è staged. Evita il git add *

2.2 Reset

Per annullare l'ultima commit(non modifica files locali) uso:

```
git reset
```

Per ripristinare i file locali a una commit precedente, ma mantenendo lo storico delle commit precedenti, uso:

```
//Individuo l'hash del commit a cui voglio tornare
git log
git revert –no-commit 46bdfb576c377ac7..HEAD
git commit -m "Messaggio nuova commit"
```

Per ripristinare i file locali a una commit precedente ed eliminare lo storico delle commit successive a quella ripristinata, uso:

```
//Individuo l'hash del commit a cui voglio tornare
git log
git reset –hard 46bdfb576c377ac7..
git commit -m "Messaggio nuova commit"
```

2.3 Clone

Per copiare un intero progetto dal server che sul pc non ho, uso:

```
git clone url
```

Questo collega la cartella con la repository con la cartella del progetto

2.4 Stato files

I file possono essere in 4 stati:

- Untracked

- Unmodified
- Modified
- Staged

Se aggiungo un file con add, diventa staged. Dopo una commit il file passa da staged a unmodified. Dopo una modifica entra nel Modified e dopo lo stage, passa da modified a staged.
Per rimuovere un file uso git rm nomefile
Il comdando:

```
git status
```

ritorna lo stato di file nella cartella Per aggiungere nuovi files uso

```
git add nomefile
```

Allo stesso modo per mettere il file nella Stage Area utilizzo

```
git add nomefile
```

2.5 Gitignore

Se creo un file chiamato .gitignore posso specificare dei file da ignorare completamente. Esempio, ignorare i files che hanno estensione o oppure a e ignorare i files che iniziano con :

```
*.[oa]
*~
```

Altro esempio:

```
# no .a files
*.a
# but do track lib.a, even though you're ignoring .a files above
!lib.a
# only ignore the TODO file in the current directory, not subdir/TO-
DO
/TODO
# ignore all files in the build/ directory
build/
# ignore doc/notes.txt, but not doc/server/arch.txt
doc/*.txt
```

2.6 Collegamento con GitLab

Se ho fatto il clone, è gia collegtao con la repository di gitlab Uso il comando:

```
git remote add nomeprogetto link
git push nomeprogetto master
```

per collegare la cartella con il progetto di gitlab.

2.7 Pull/Fetch

L'istruzione fetch scarica eventuali nuove versioni dal server, senza modificare quelle presenti.
L'istruzione pull scarica le nuove versioni dal server e esegue la merge con i file locali

2.8 Branch

Le branch sono gestite da git in modo trasparente. Sono degli snapshot dei files che vengono switchati a seconda del branch su cui stiamo lavorando. Il comando

```
git checkout nomebranch
```

permette di cambiare la branch corrente. Per creare una nuova branch, prima di nomebranch utilizzo l'opzione -b. Per eliminare un branch utilizzo l'opzione -d.
La branch di default è master. Devo riconfigurare il push come consigliato nel nuovo branch.