**Commit**

Si considera ora di inserire nella cartella “gitTutorial” un file di testo di nome “recensione.txt”. Nonostante il file sia nella cartella, non è ancora stato salvato nella reposy di Git.

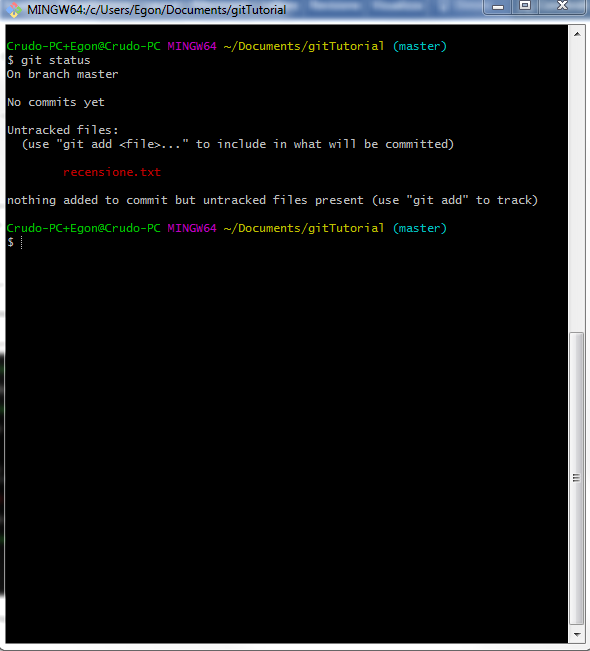
In Git si hanno 3 layer di lavoro:

* La **directory** dove vengono salvati istantaneamente i file con cui si sta lavorando, detti *working copy*
* **Index**: è una lista transitoria sulla quale verranno salvate le “proposte” da validare, ossia i file modificati le cui modifiche devono essere validate. Index è detta anche “*staging area*”
* **Head**: è il ramo attuale di lavoro. Ogni ramo corrisponde ad una versione del progetto, si ha sempre un ramo “master” che è la versione radice del progetto da cui si avranno altri rami/nodi relativi a versioni differenti. Prima di poter validare un file è necessario aggiungerlo all’Index

Il comando

git status

permette di elencare lo stato degli elementi contenuti nella cartella



Nell’immagine si notano due cose:

* È presente il file *recensione.txt* in rosso, ciò implica che il file non è stato ancora aggiunto all’Index
* La stringa “Crudo-PC……” contiene le informazioni riguardo la directory a cui punta git e il ramo (branch) di lavoro, in questo caso il brach è il master, ossia il ramo radice dal quale sarà possibile creare ulteriori branch. Ogni qualvolta che un repository viene realizzata, si partirà dal branch master

Per aggiungere il file nell’Index si possono usare i seguenti comandi:

git add recensione.txt ----------> si aggiunge solo il file “recensione.txt”

git add . ---------> si aggiungono tutti i file contenuti nella cartella puntata

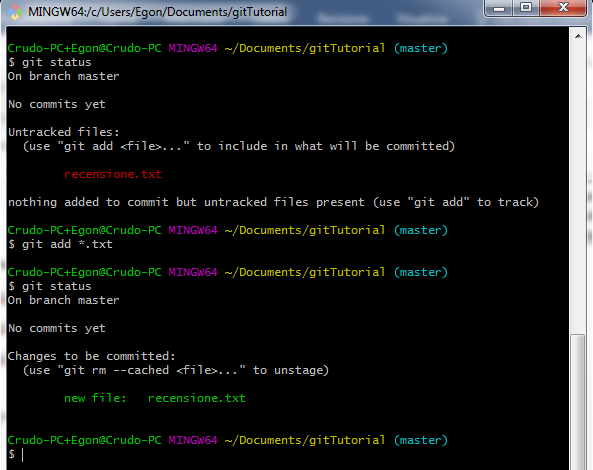
git add \*.txt ------> si aggiungono tutti i file di tipo .txt

Per rimuovere uno o più file dallo staging area

git reset HEAD <nome File> ---------> rimuove un file specifico dall’index

git reset HEAD -----> rimuove tutti i file nell’index

ri-eseguendo il comando git status si avrà il seguente messaggio



Il file adesso viene riportato con il colore verde, indice del fatto che è disposto nell’Index. Ora si può validare la modifica (in quanto alla creazione il reposy di git associato alla cartella è vuoto) mediante il comando

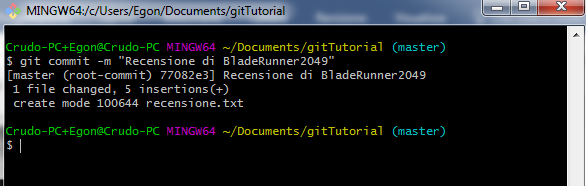
git commit -m “<messaggio personalizzato>”

oppure semplicemente con

git commit

In questo caso verranno visualizzati dei messaggi che ci ricorderanno di aggiungere un messaggio al commit, per fare ciò si preme “I” (i), si inserisce il messaggio (che verrà riportato in giallo), si preme “Esc” e poi per uscire confermando le modifiche si immette “wq”. Le stringhe precedute dal cancelletto risultano essere dei commenti

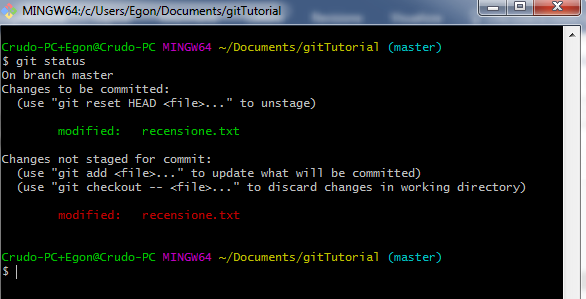
Ciò che fa commit è essenzialmente “fotografare”(salvare) lo stato dell’head in quel dato istante, analogamente a quando viene scattata una foto istantanea. Il messaggio del commit risulta essere un riferimento per risalire al commit desiderato!



Ripetendo il comando git status adesso non uscirà nessun file nella directory in stato “transitorio”, ossia da cui si può proporre una modifica ed eventualmente validarla.

Per modificare file nella repository basterà quindi modificare il file, aggiungerlo all’index e poi committarlo!

Si considera ora di modificare il file recensione.txt e di aggiungerlo alla lista, prima di eseguire il commit si modifica ulteriormente il file e si esegue git status, l’output sarà quindi



Il file verde sarà quindi il file modificato nell’index, quindi quello proposto per la validazione, mentre quello rosso sarà quello appena modificato e non ancora aggiunto all’index!

Se si committa il file quindi, alla repository verrà aggiunto la versione riportata in verde, quindi quella precedente all’ultima modifica, mentre l’ultima versione sarà ancora “fuori” dalla lista index.

E’ possibile effettuare l’aggiunta all’index e il commit in un unico passaggio mediante il comando

git commit -a -m “ messaggio dell’add e commit insieme”

Bisogna fare attenzione a questa funzione, perché prenderà tutte le working copy dei file nella cartella e li committerà al repository, quindi quando si ha a che fare con tanti file può essere rischioso nel caso non si voglia committare dei file in particolare!

Per rimuovere un file dal repository si esegue il comando

Git rm --cached <nome\_file>

Per vedere le differenze tra i file istantaneamente in uso e quelli presenti nell’index o committati, si usa il comandi

get diff

oppure se si vuole vedere le differenze solo con un singolo file

get diff <nome\_file>

NB: nel nome del file è richiesto anche il formato del file in questione!

Per vedere la cronologia dei commit si usa il comando

git log

nel quale sono riportati anche gli autori del commit, il messaggio , l’ora e il giorno.

E’ possibile filtrare la cronologia, per esempio per autore mediante

git log --author =”<nome autore>”

esempio

git log --author = “Egon Gamper”