**Branch**

Una branch è una copia della repository principale master o di un altra branch. Creare branch (rami) risulta essere utile quando si vuole lavorare su delle versioni individuali del progetto senza influenzare quelle condivise con altre persone o le altre versioni. Le modifiche riguardanti una branch non sono condivise con le altre. Per trasmettere i contenuti di una branch (quindi le modifiche fatto rispetto ad un’altra versione), è possibile unire le varie branch mediante un comando denominato merge

Per creare un ramo si usa il comando

git branch <nomeBranch>

Per esempio

git branch MyBranch

Per far si che la cartella di lavoro si riferisca al nuovo branch si usa il comando

git checkout <nomeBranch>

Esempio

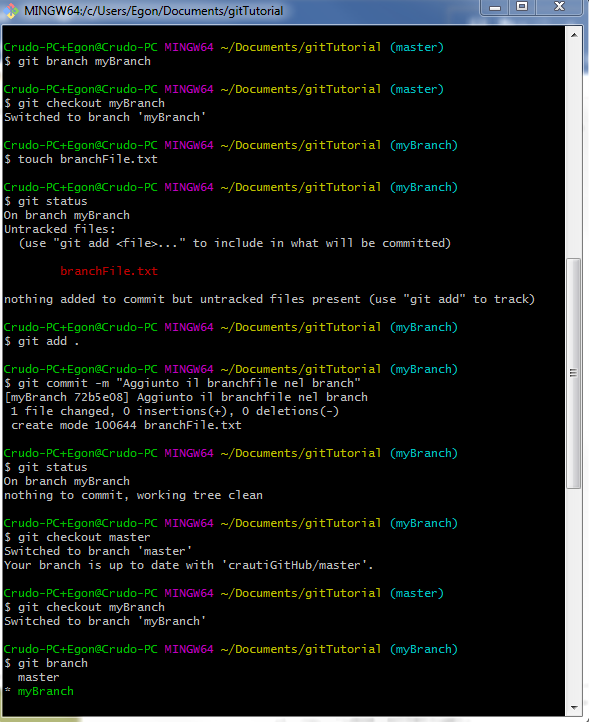
git checkout MyBranch

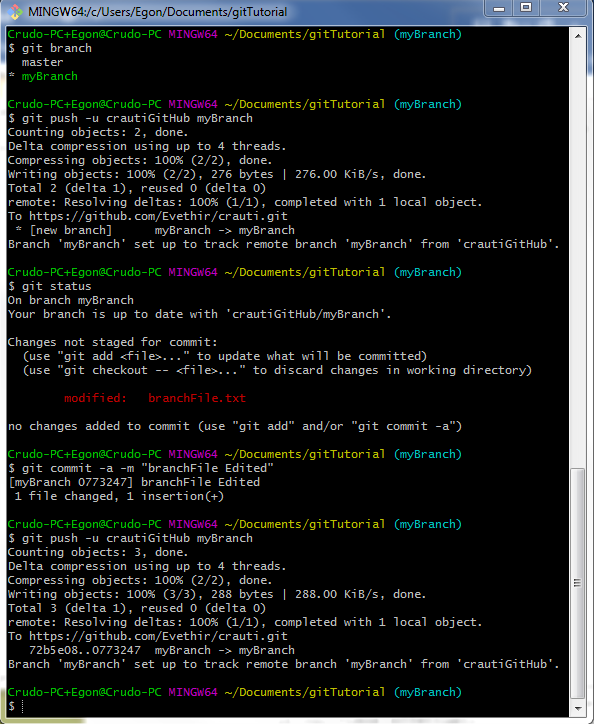
Per creare e passare direttamente alla nuova branch si fa

git checkout -b <nomeBranch>

Per eliminare il branch invece

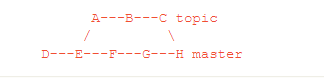
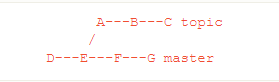
git branch -d <nomeBranch>

Una volta creati i file, perché siano validi al branch bisogno eseguire la procedura di committaggio solita. Una volta che un file viene committato in un branch, se si passa ad un altro ramo, la cartella di lavoro non vedrà più i file dell’altro branch, ma i propri, i quali possono essere versioni differenti o file diversi.



Git Merge

Merge copia i commit di un dato branch in un’altro, quindi le varie modifiche, sia come modifiche file esistenti che nuovi file, verranno riportati nel branch dell’head.



Considerando di avere l’head puntato sul master, al commit “E” viene creato il branch topic che vede i vari commit A,B,C. All’ultimo commit si avrà l’unione dei branch mediante la funzione merge, che ripeterà tutti i commit effettuati sul branch topic da quanto è stato creato (punto E) sino al punto in cui viene effettuato il merge.

Il comando per fare il merge è

git merge <nome\_branch>

Nel caso sopra, considerando di essere nel master, il comando sarà

git merge topic

In caso di conflitti è possibile annullare il merge mediante il comando

Git merge –abort

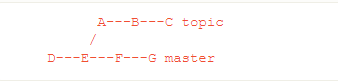
E git cercherà di ricostruire lo stato precedente dei branch

La peculiarità del merge è il fatto che non altera in alcun modo la linea temporale dei vari commit ed è più facile risalire agli autori di questi ultimi mediante il braching.

Rebase

Rebase è un metodo che unisce i vari commint tra branch ridefinendo la linea temporale dei commit su uno dei due branch (solitamente un master).

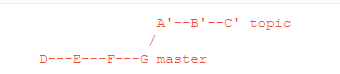
Consideriamo il seguente commit workflow



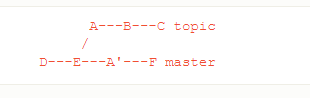
Al punto E, il branch master genera il branch topic che avrà i proprio commit personali.

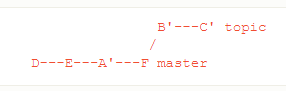
Effettuare un rebase vuol dire traslare l’istante di generazione di un branch sull’ultimo commit del branch obiettivo, in modo tale da riscrivere la storia e far si che le modifiche effettuate sul branch obiettivo siano trasportate sul branch di lavoro.

Si ipotizza quindi di fare un rebase del branch topic sul master, il grafo sarà del tipo



Se il master avesse condiviso un commit uguale al branch topic, questo topic sarebbe stato eliminato dai commit di topic





Si può notare quindi una ricollocazione della cronologia dei commit, il che può risultare svantaggioso in quanto il rebase può offuscare l’autore delle varie modifiche effettuate sul progetto, pertanto si consiglia l’uso del merge per progetti condivisi, mentre il rebase per progetti privati.

Per richiamare il comando rebase bisogna avere il puntatore dell’header puntato sul branch da riallocare (quindi router nell’esempio) e performare il comando git rebase sul branch dove avviene il ricollocamente (ossia il master)

git rebase master

(il che è invertito rispetto a merge)