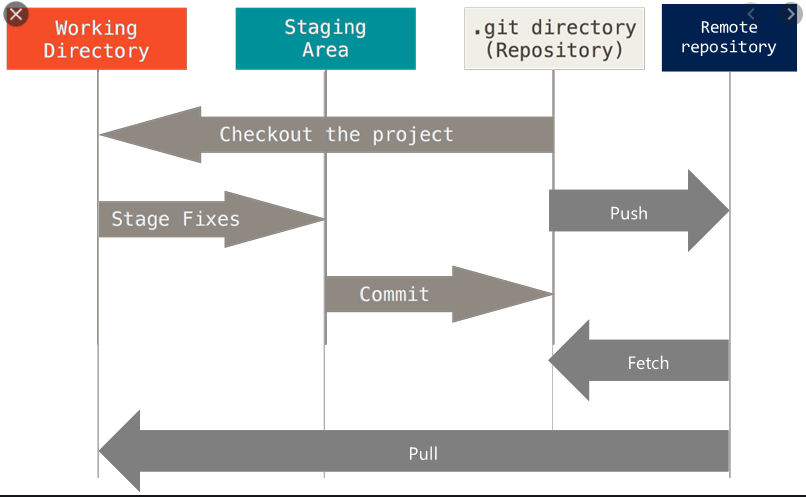
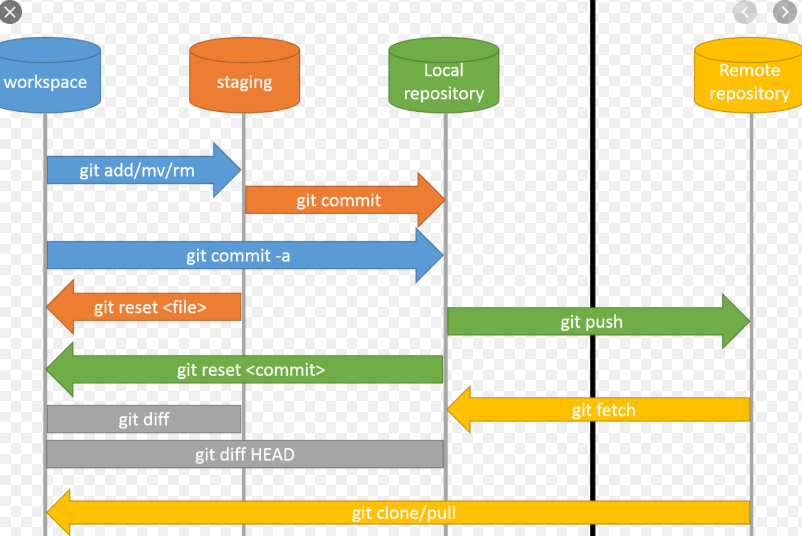
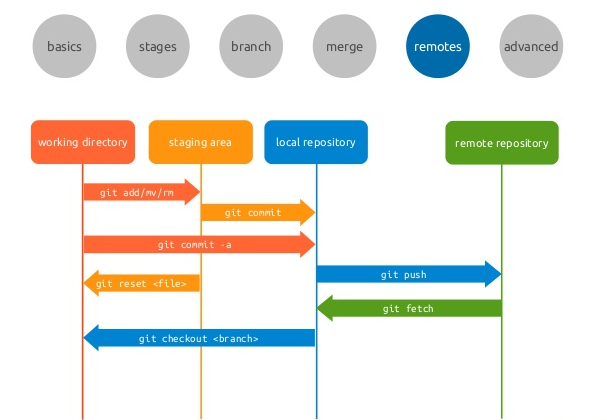
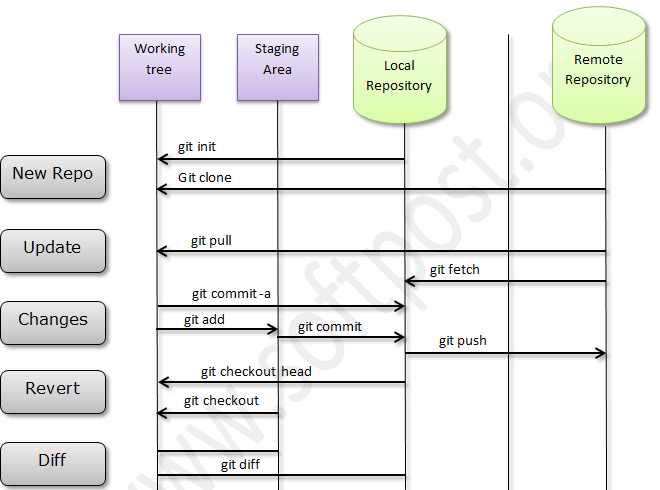
**SCHEMA FUNZIONAMENTO GIT**

Una serie di schemini che riassumono i comandi ed i flussi principali di git





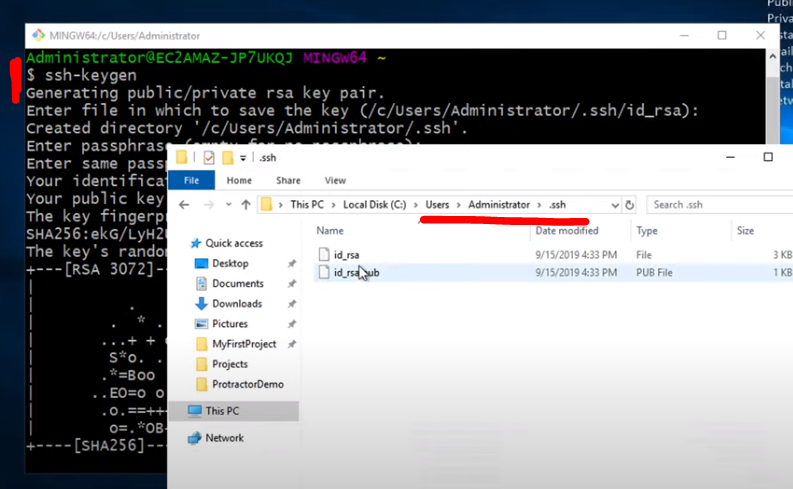




**COME CREARE CHIAVI SSH**

È possibile clonare un progetto utilizzando https e fornendo username e password o in alternativa SSH (secured shell) così possiamo connetterci senza fornire ogni volta username e password ma in quest’ultimo caso è necessario definire una chiave nella repositories remota al fine di supportare una comunicazione sicura.

Ssh-keygen

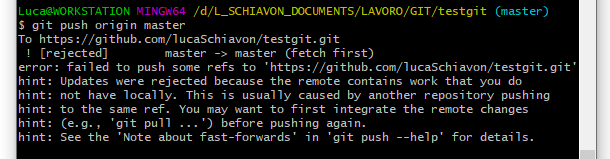


Viene creata una chiave pubblica e privata

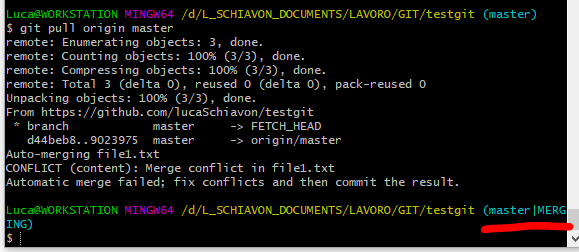
Poi dentro gitlab in setting ssh keys incollare la chiave pubblica copiata da rsa\_pub

**per risolvere un conflitto:**

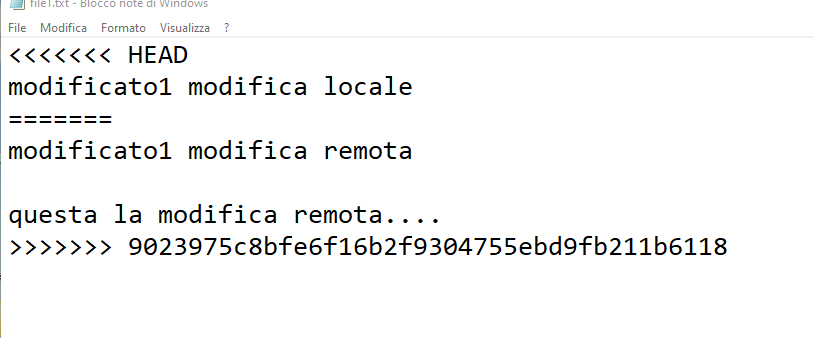
caso in cui ci siamo dimenticati di fare il pull e nel frattenpo qualcun altro ha modificato i file…



Faccio il pull da remoto, questo causa il merge del file modificato da remoto in quello locale modificato localmente:



Infatti guardiamo nel file:

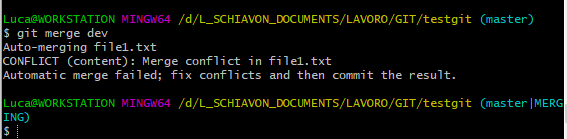


Notiamo che il commit remoto è stato mergiato con il commit locale (HEAD)

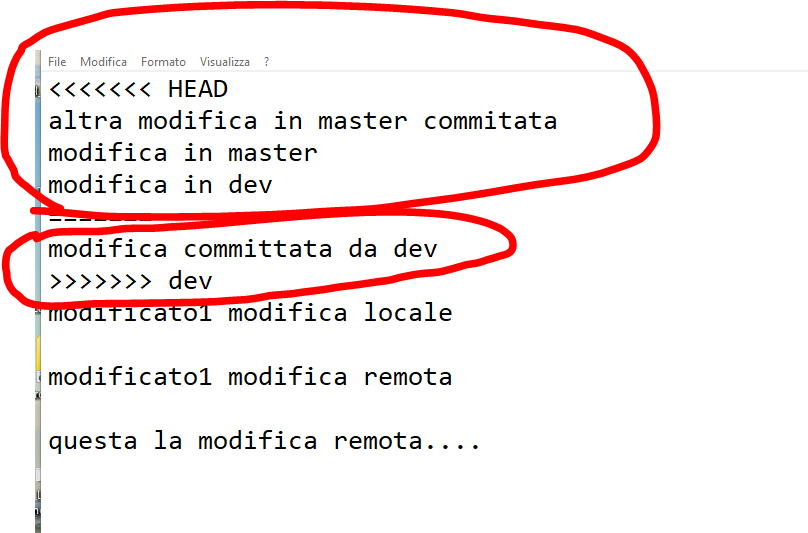
A questo punto sta a moi risolvere i conflitti cancellando cio che non vogliamo tenere e quindi, una volta risolta la cosa, aggiungiamo allo stage con add poi committiamo e quindi pushiamo.

Altro caso di conflitto

Il file1 del master viene modificato e committato, viene anche modificato il file 1 del dev e committato senza prima fare il merge del master nel dev, faccio il merge del dev dentro il master ed ottengo questo errore:

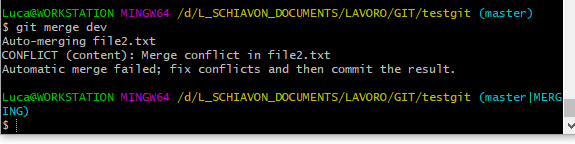


Entriamo nel file1 e:

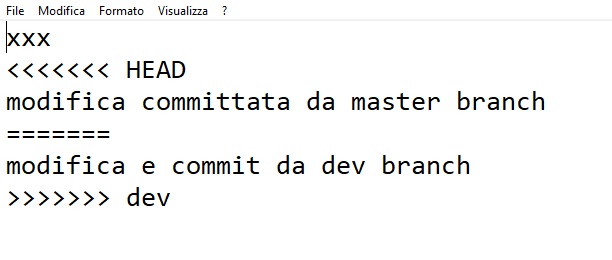


Anche qui dobbiamo risolvere i confitti noi decidendo cosa tenere e poi lanciando add commit, a questo punto potremo spostarci in dev (se non risolviamo il conflitto non possiamo)

Il file2 del dev viene modificato e committato, viene anche modificato il file 2 del master e committato senza prima fare il merge del dev nel master, faccio il merge del dev dentro il master ed ottengo questo errore:



Apro il file incriminato e risolvo manualmente il conflitto tenendo ad esempio entrambe le modifiche:



**COMANDI UTILI per affrontare situazioni particolari**

Nel caso ci fosse l’esigenza di vedere l’alberatura dei branch per un progetto da riga di comando:

git log --all --decorate --oneline --graph

nel caso ci trovassimo in situazione di conflitto per far vincere sovrascrivendo con il current branch il branch di destinazione ossia nel caso non si riuscisse più a passare da un branch ad un altro per problemi di files non allineati:

git branch -f otherbranch currentbranch

otherbranch (branch di destinazione)



Si noti che il comando non funziona se lanciato dal branch di destinazione:



se volessimo vedere le differenze tra due branch

questo comando mostra le differenze tra branch in caso di disallineamenti e non riuscendo più a fare il merge di un branche sull’altro:

git diff <branch\_sorgente> <branch\_target>

per tornare alla versione di un file prima dell’ultimo commit in caso si fossero fatti pasticci:

questo comando rimpiazza il file corrente riportandolo all’ultimo commit:

git reset head <nomefile>

per fare il rollback all’ultimo commit:

git reset --hard HEAD

git stash

git rebase