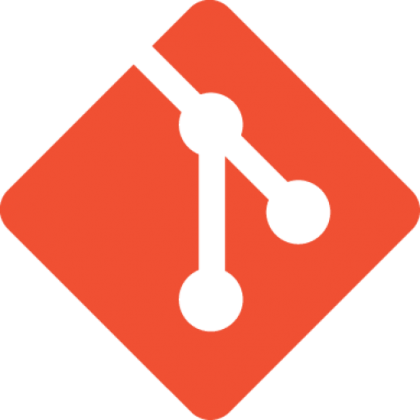
GIT Tutorial (Beginners) 

# Preface

**Author : Bhabesh**

**http://om.linkedin.com/in/bhabeshm**

Using GitLab & Source Tree

Step by step hands on guide on git.

There are lots of materials available for learning GIT. So mine is nothing new invention but it might be rather handy to people who first want to familarise themselvs with source control management using Git.

Most of the cases beginers worried about the command line interfaces specially who are not familier well with linux systems. Although git’s real power lies on command line. But to start git from command line for a newbie might not be a good idea.

To find a workaround , I found couple of great tools around. They all are good . I choose ‘Source Tree’ from Altassian. Howerver others are also found good fit.

Git can be hosted in linux or windows server. I was trying to find something like github but a community edition so that interested people will host on their own server as well. I found GitLab then. But the constraint is on linux. It’s not available on windows till. I consider the system administrators will understand this. However if any window lover want to go with the GitLab, there’s still a chance to fit. In turnkeylinux I found stack for gitlab on virtual (specially for vmware). Though turnkey linux’s virtual for GitLab is bit old, but hope users will survive and try to upgrade them on their own.

**Email : [*bhabesh.m@gmail.com*](mailto:bhabesh.m@gmail.com)

Web : <http://om.linkedin.com/in/bhabeshm>

# Terms

All representations and information contained in this document have been compiled to the best of my knowledge and carefully tested. However, mistakes cannot be ruled out completely. The authors assume no responsibility or liability resulting in any way from the use of this material or parts of it or from any violation of the rights of third parties.

Reproduction of trade marks, service marks and similar monikers in this document, even if not specially marked, does not imply the stipulation that these may be freely usable according to trade mark protection laws. All trade marks are used

without a warranty of free usability and may be registered trade marks of third parties.

This document is published under the “Creative Commons-BY-NC-ND 3.0 Unported” licence. You may copy and distribute it and make it publically available as long as the following conditions are met:

Attribution : You must make clear that this document is a product of the author.

No commercial use You may not use this document for commercial purposes

(contact me if you want to use this document commercially).

No derivatives You may not alter, transform, or build upon this document (contact

us if necessary).

The full legal license grant can be found at <https://creativecommons.org/licenses/by/3.0/us/legalcode>

# Table of Contents

[Preface 0](#_Toc413329231)

[Terms 2](#_Toc413329232)

[Table of Contents 3](#_Toc413329233)

[Getting Started 4](#_Toc413329234)

[01. Setting up a repository (Local) 4](#_Toc413329235)

[Steps 4](#_Toc413329236)

[02.Setting up a repository (Remote) 5](#_Toc413329237)

[Steps 5](#_Toc413329238)

[03.Configuration of Repository 6](#_Toc413329239)

[Steps 6](#_Toc413329240)

[Command 7](file:///C:\Users\bhabesh\Documents\gitlab\cis_presentation\tutorial_git.docx#_Toc413329241)

[04.Saving changes 7](#_Toc413329242)

[Git Add 7](#_Toc413329243)

[05.Git Commit 9](#_Toc413329244)

[06.Viewing Old Commits 11](#_Toc413329245)

[Steps 11](#_Toc413329246)

[Undoing Changes 14](#_Toc413329247)

[07.Revert 14](#_Toc413329248)

[08.Reset 17](#_Toc413329249)

[Collaborating 23](#_Toc413329250)

[Syncing 23](#_Toc413329251)

[11.Remote 23](#_Toc413329252)

[12.Push 26](#_Toc413329253)

[13.Fetch 28](#_Toc413329254)

[13.Pull 30](#_Toc413329255)

[Using Branch 31](#_Toc413329256)

[Branch 31](#_Toc413329257)

# Getting Started

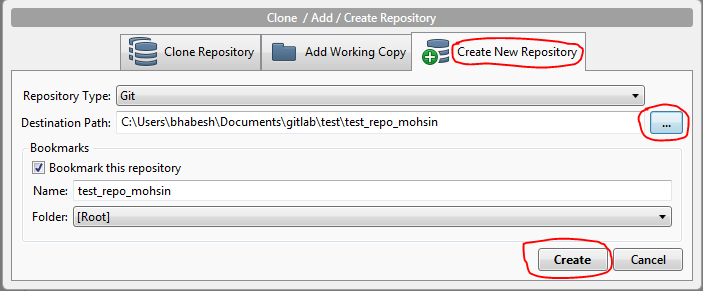
**Command**  
git init

## 01. Setting up a repository (Local)

### Steps

1. **Create a Directory** at your window machine test\_repo\_<your\_name> (Example : test\_repo\_mohsin )
2. Click Clone/ New

Click Create New Repository



Select

*Repository Type*: Git  
*Destination Path* : <Folder where you want your local repo>

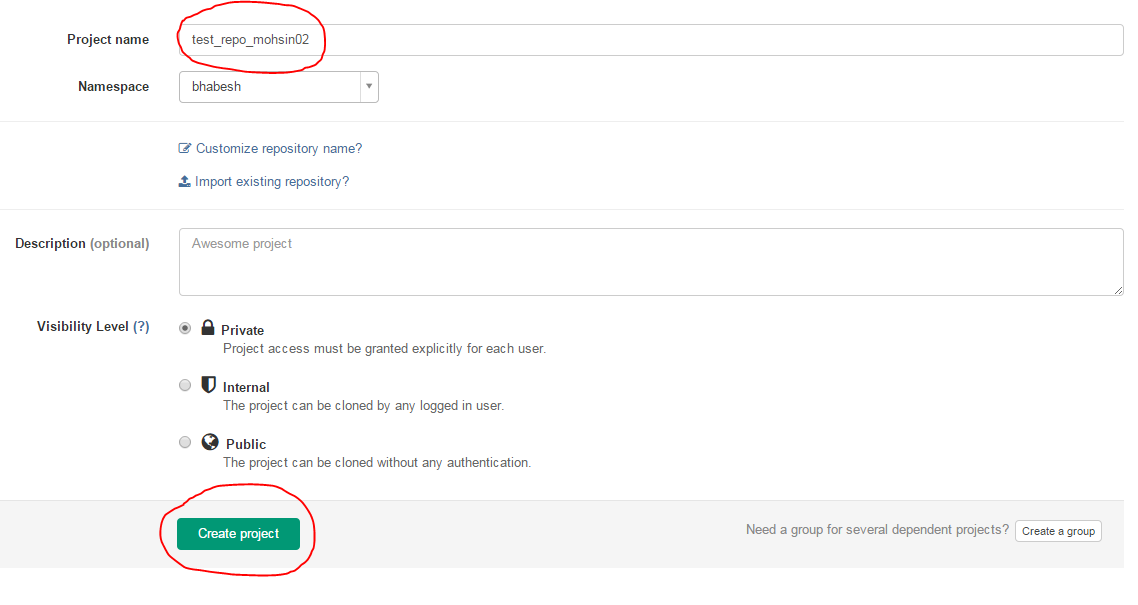
*Bookmark this repository*: check the box

Click on **Create**

## 02.Setting up a repository (Remote)

**Command**  
git clone

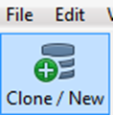
### Steps

1. login to GitLab repository (e.g. http://172.30.14.42)
2. At dashboard, Click
3. provide a justified name e.g. test\_repo\_<your name>02

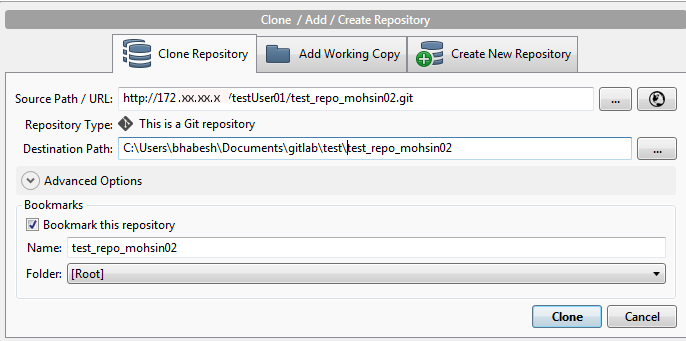
1. Click **Create Project**
2. Click http and copy the url



1. In your local machine select Source Tree



1. Click Clone / New
2. Paste the url in at the Source Path / URL at the dialogue box and then click **Clone**

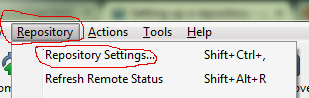


## 03.Configuration of Repository

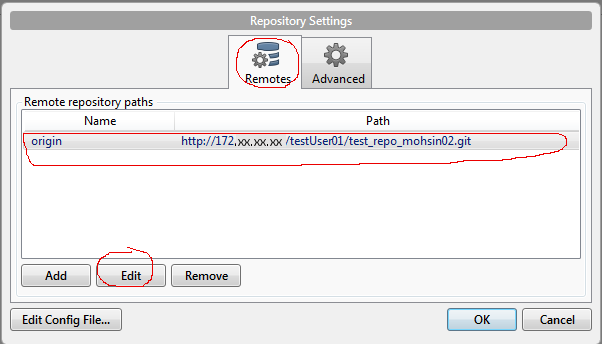
**Command**  
git config

### Steps

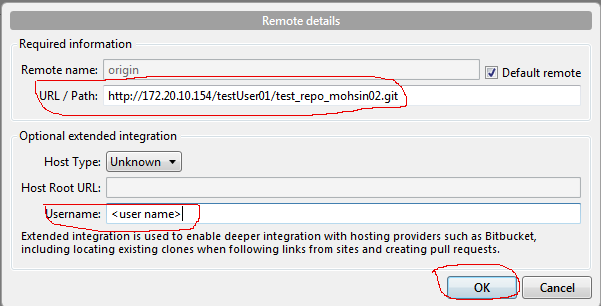
1. Select the repository using double click at the source tree
2. Click **Repository/Repository Settings..**



1. Click
2. **Remotes** tab ,
3. Select the existing (required) row from **Remote repository paths**
4. Click **Edit** button



1. Make sure that   
   **URL/Path** : valid git url   
   **Username** : git user name (e.g. active directory user name)   
   Click **Ok** (Click **Ok** again to Exit from dialogue box)



## 04.Saving changes

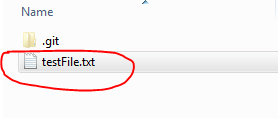
Command  
git add

### Git Add

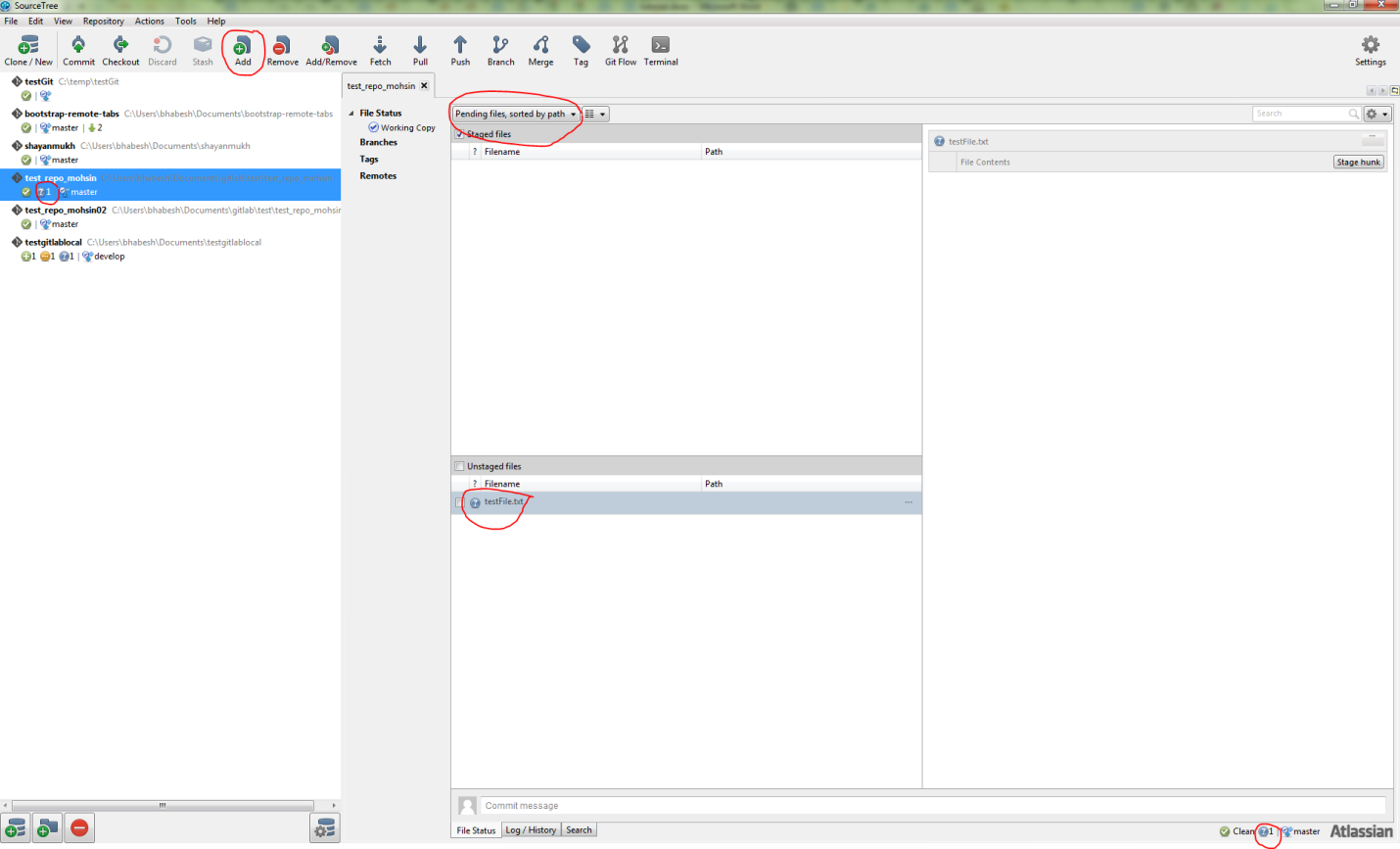
#### Steps

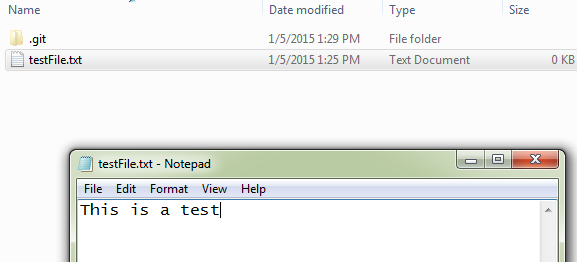
To avoid confusion you can **select** the repository you just cloned from remote

1. Create a file (e.g. testFile.txt) in your repository directory



1. View changes in **SourceTree** ui
2. Click on the file (**purple** color) (e.g. testFile.txt) at ‘Unstaged files’ at right side of book mark pane



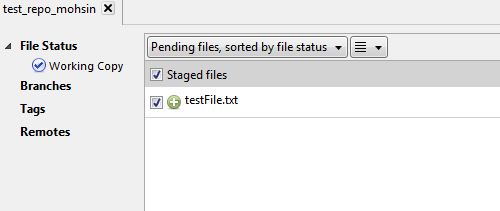
1. File will be added at staging
2. Open the file and add some text
3. File will again visible at ‘Unstaged Files’ area with **yellow** color
4. Select the file and click add (if needed)
5. Changes will be saved at staging area

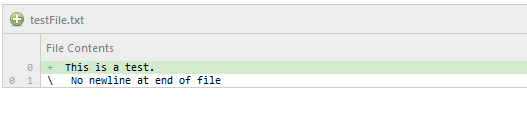
### 05.Git Commit

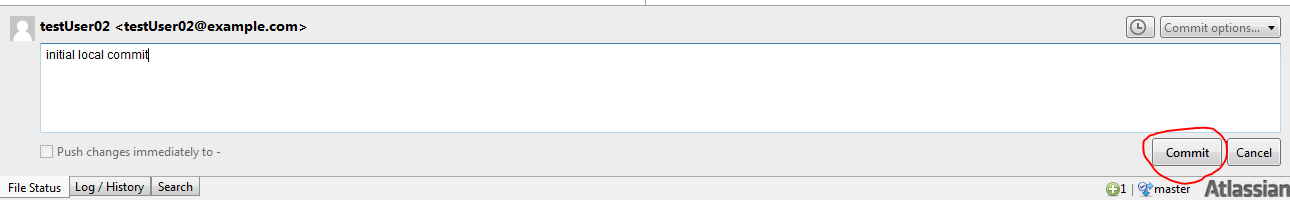
Command  
git commit

#### Steps

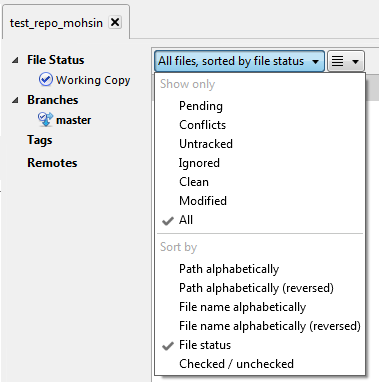
1. Select file/s from ‘Staged Files’

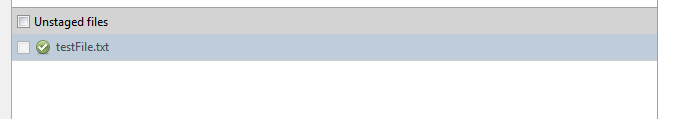


1. Check the changes
2. Click on commit
3. Give some comment/justification of commit in commit dialogue box and click **commit** button

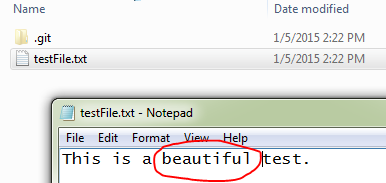


1. File will be committed. It can be viewed as below :

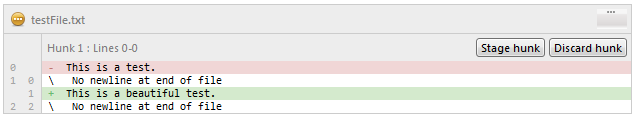




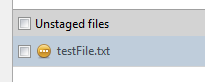
1. Modify some text



1. In SourceTree the changes are like this

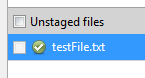


1. It is possible to stage only the portion or full file
2. Click the file at Unstage area (‘**git add**’ command will be applied) .

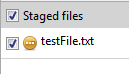


1. After add file will be staged and file color will change in two places
   1. In ‘**Unstaged files**’ area

Color will change from yellow to green (with icon changes to check)



* 1. And in ‘**Staged files**’ area will change from green to yellow (with icon changes)



1. Select the file and commit. File will be committed locally.

## 06.Viewing Old Commits

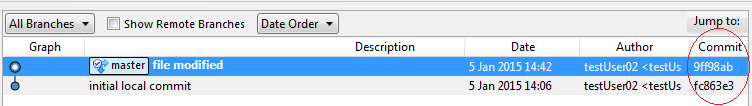
Command  
git checkout

### Steps

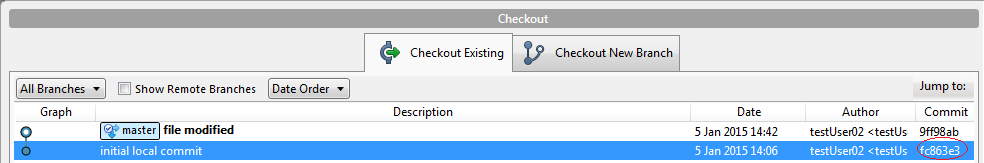
1. Click on log view
2. Book Mark Pane looks something similar



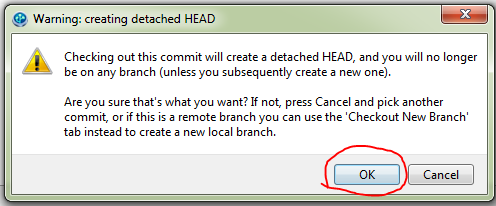
1. See the commit log



1. Click on **checkout**
2. Select the particular commit



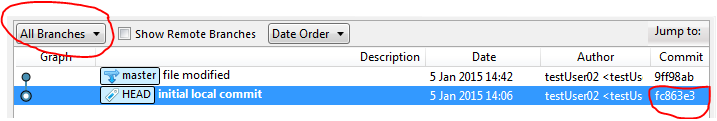
1. Creation of detached local branch dialogue appears. Click **OK**



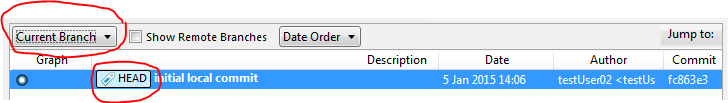
1. BookMark pane looks something similar



1. Right side pane for log :
   1. For all Branch



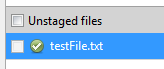
* 1. For Current Branch



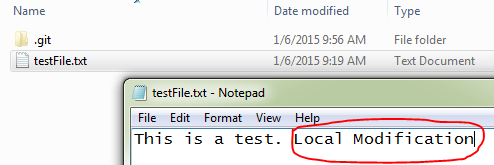
1. Click on **File Status**



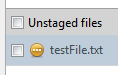
1. **Unstaged files** listing

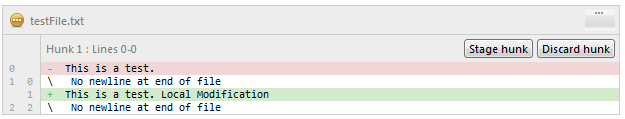


1. Open the file and modify

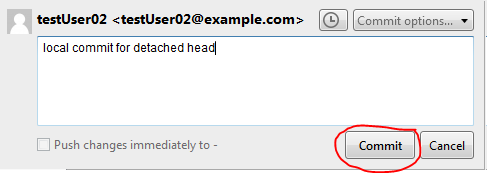


1. **Unstaged files** listing



1. Changes in file
2. Check the file in **Unstaged files** pane to add to staged area
3. Click on **Commit**

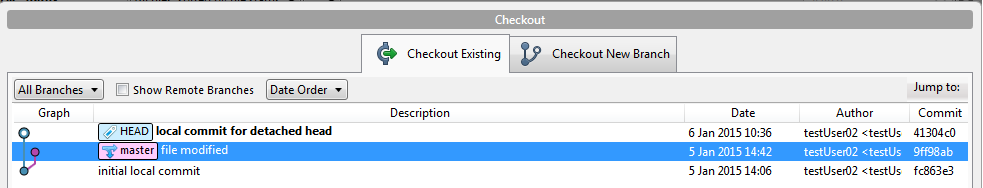




1. Click on **CheckOut**



1. Select **Master**



1. Click **OK**
2. You will find your earlier work is not modified.
3. It proves checkout an earlier commit will not disturb the present master branch.
4. However this is not the case of checkout a file.

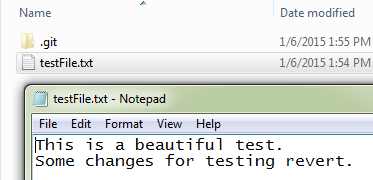
## Undoing Changes

Command  
git revert

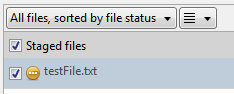
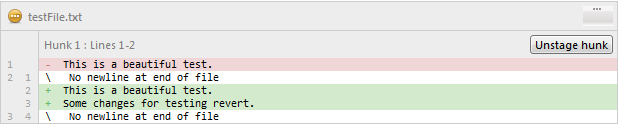
### 07.Revert

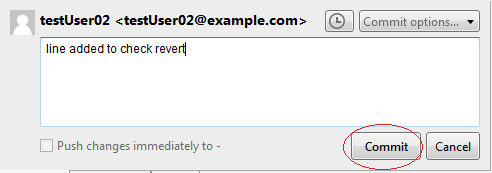
#### Steps

1. Open the file and do some changes

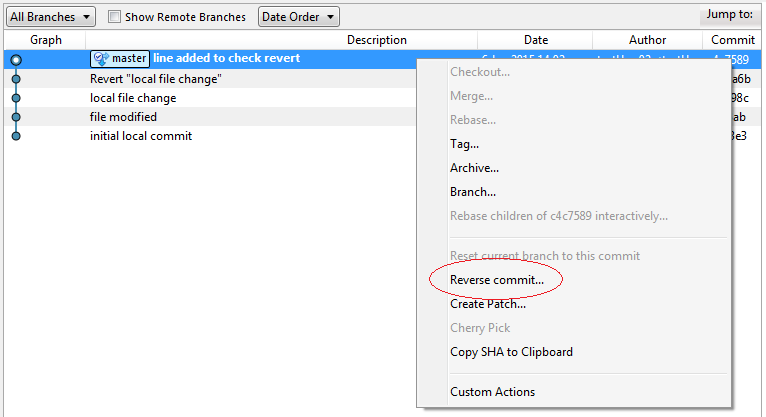


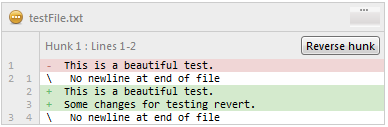
1. Commit it



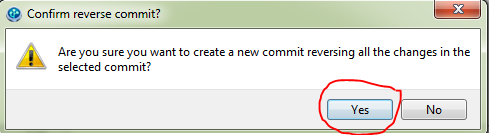


1. Select the recent commit , right click and select **Reverse commit**

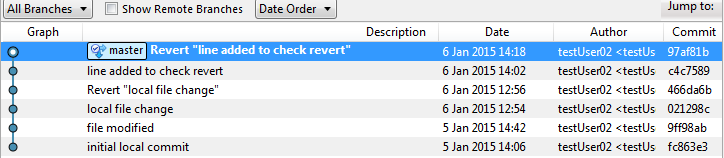


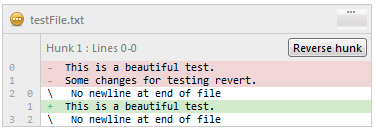


1. Click **Yes** on Dialogue box



1. The contents will be earlier commit stage



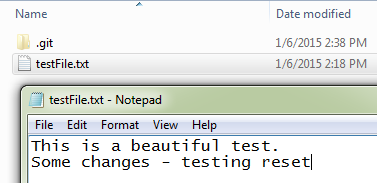


### 08.Reset

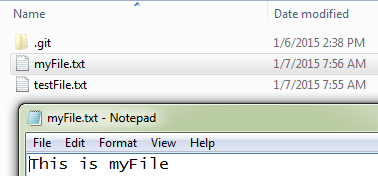
Command  
git reset

#### Steps

1. Modify testFile.txt



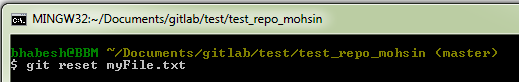
1. Add another File

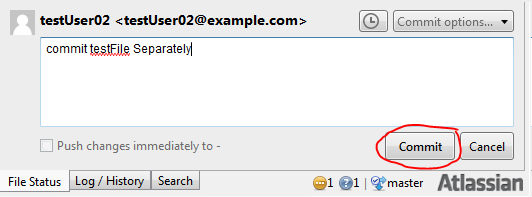


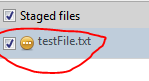
1. Add for staging

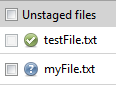


1. Reset the file (e.g. myFile.txt) .. **It will be unstaged**

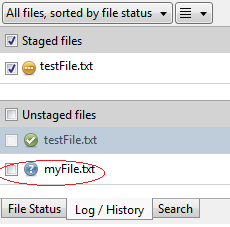


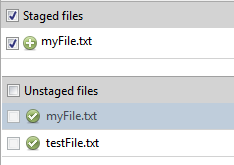
1. Commit Files seperately

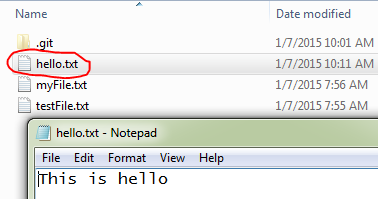
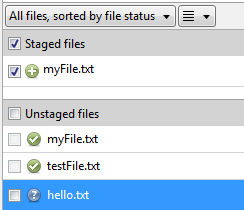






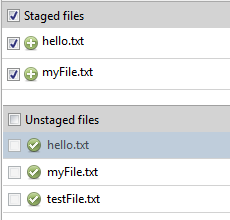
1. Add the other file from unstaging area



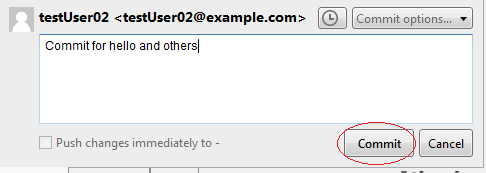
1. Create another file (e.g. hello.txt)



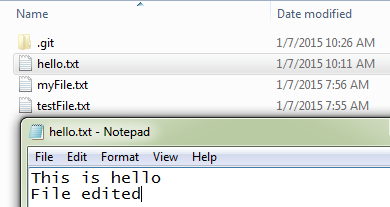
1. Now Add the file (from Upstaged to staged) : Click

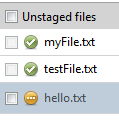
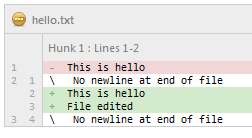


1. Commit :

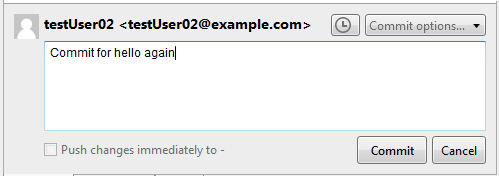


1. Edit the newly committed file (e.g. hello.txt) again :

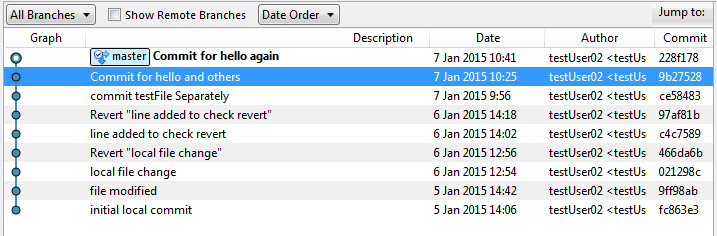


   
11. Add the file to staging area.

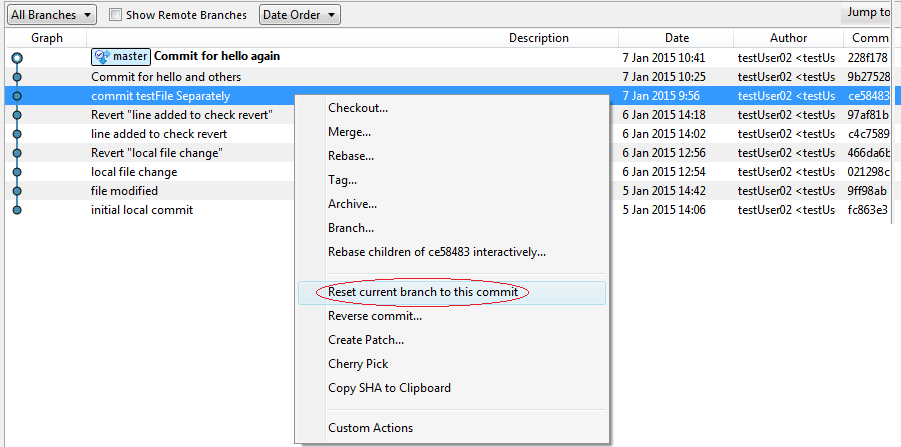
12. Commit : 



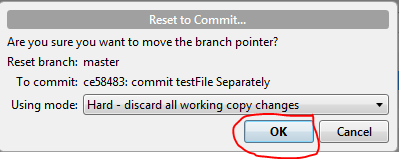
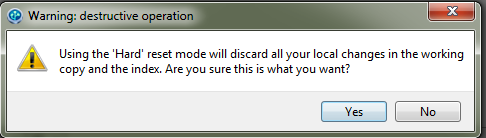
13. Commit logs :



14. To Reset right click particular commit (e.g. two commits before) and select (**Reset current branch to this commit**)

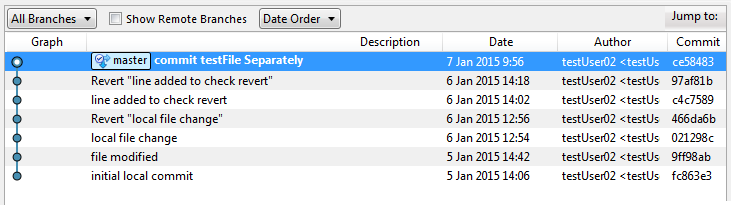


15. Click **OK**  to Warning notice



16. Commit Logs : (**addition and modification lost**)

Current status lost



# Collaborating

Command  
git remote

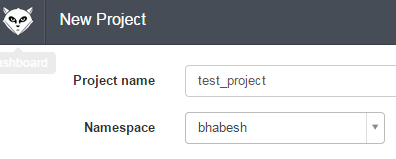
## Syncing

### 11.Remote

Remote connections are more like bookmarks rather than direct links into other repositories

#### Steps

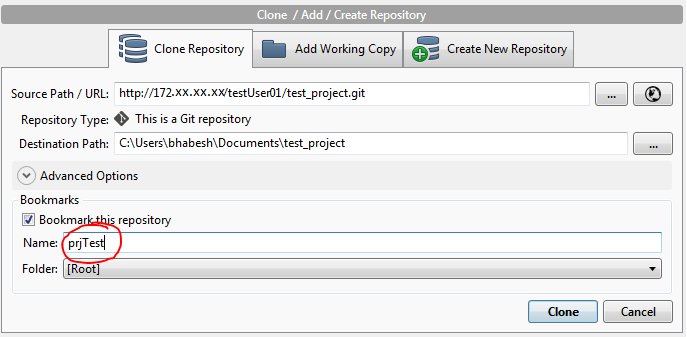
1. Create a project (similar to Section 02. ) (e.g. **test\_project**) from hosted gitLab



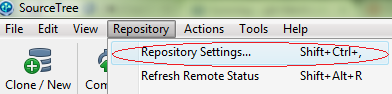
1. Copy the **url**



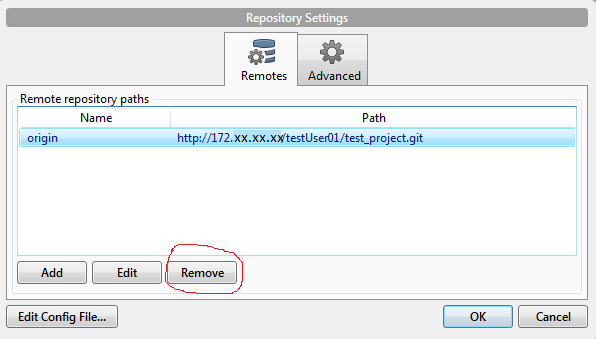
1. Click **Clone/New**



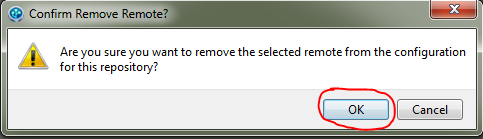
1. Click **Repository / Repository Settings…**



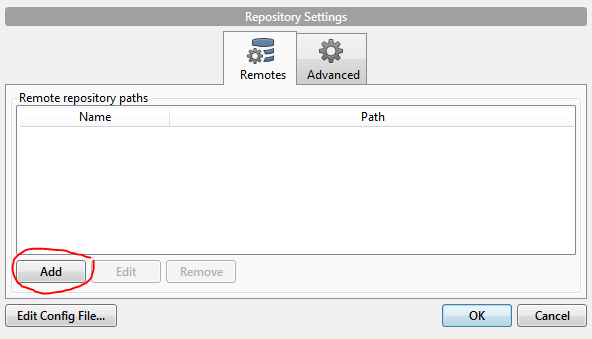
1. Repository settings view as below :



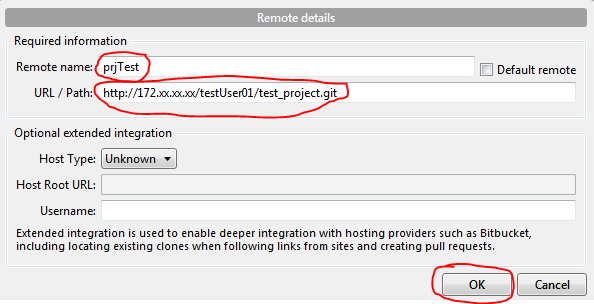
1. Remove the entry **origin** (to understand the concept for the time being)



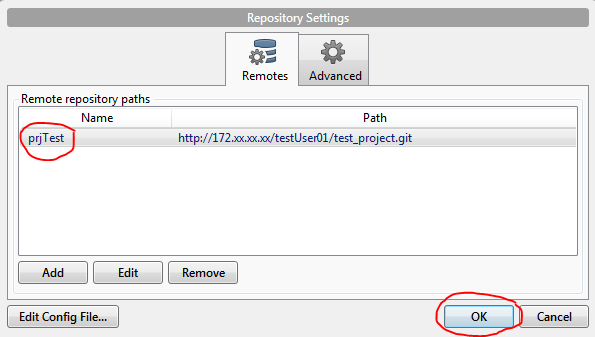
1. Click **Add**



1. Fill it accordingly



1. Repository settings should look something similar :



1. Click **Ok**

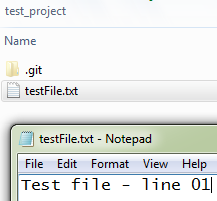
### 12.Push

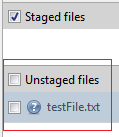
Command  
git push

Transfer commits from local repository to a remote repo.

#### Steps

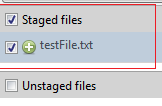
1. Create a file (e.g. **testFile.txt**) in your project (e.g. **test\_project**)



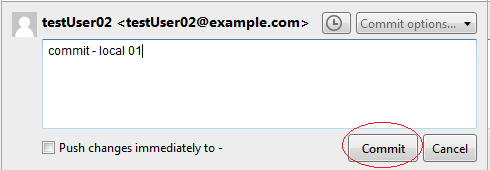




1. Add the file from ‘**Unstaged files’** to ‘**Staged files’** using **Add**

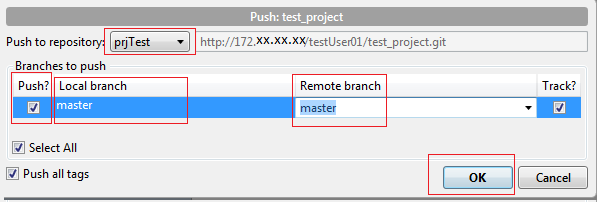


1. Commit to the local repository

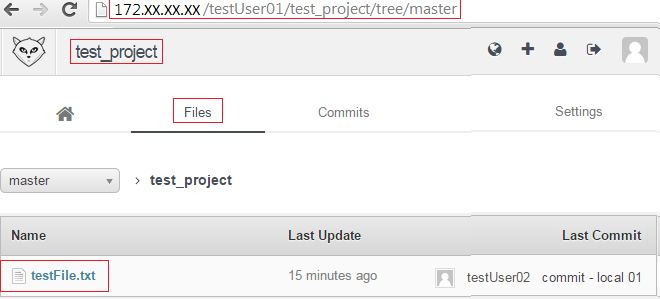




1. Click ‘**Push’**
2. In push dialogue box make sure you select (type) **master** as your remote and local branch



1. Using GitLab interface, we can see the remote commit as shown below :



### 

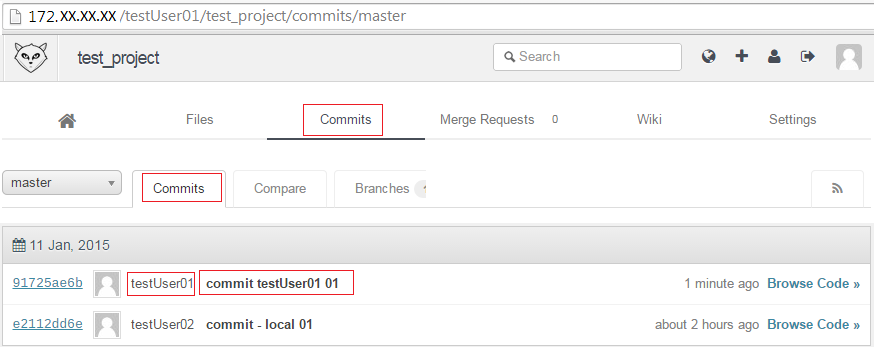
### 13.Fetch

Command  
git fetch

This command imports commits from a remote repository into the local repository.

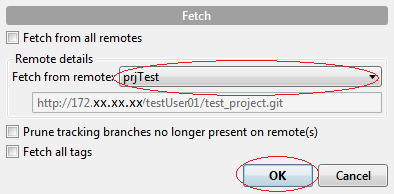
#### Steps

1. Let some **other user** modify the file , commit, and then push the changes to the remote repository (irrespective of any tool)

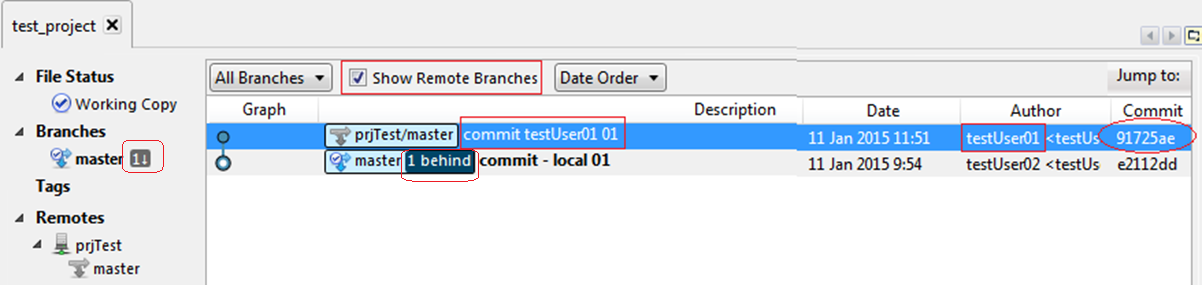




1. Click on **Fetch**
2. Fetch dialogue box will appear



1. Right side pane looks something like below :



Till date the modified file is not in local repository but the **fetch** brings the commit from remote repository to local.

### 13.Pull

Command  
git pull

Merging upstream changes into local repository

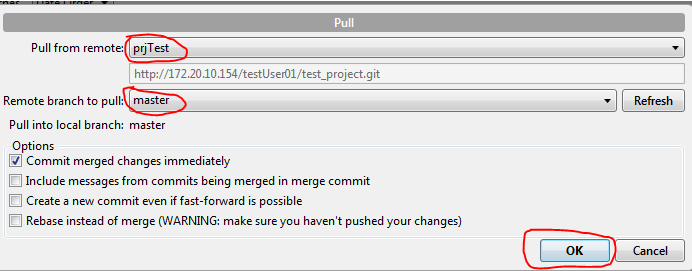
*(It’s a combined command of* ***git fetch*** *and* ***git merge****)*

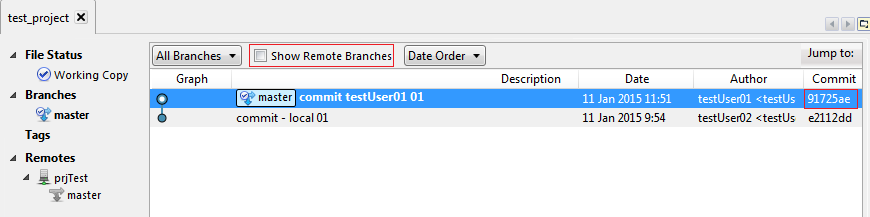
#### Steps

1. Click ‘**Pull ‘**

*(Check the numeric degit above . It shows changes in remote)*

1. Click ok in the dialogue box



1. Log/history pane should look something like below :

Now the modified file from the remote repository is in local along with the commit

## Using Branch

Command  
git branch

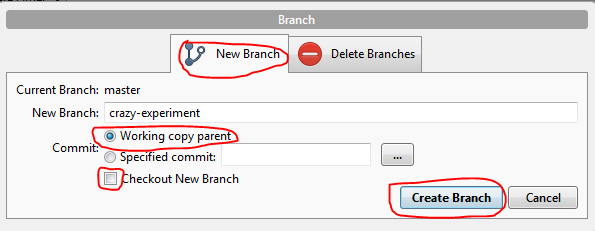
### Branch

A branch represents an independent line of development. Branches serve as an abstraction for the edit/stage/commit process

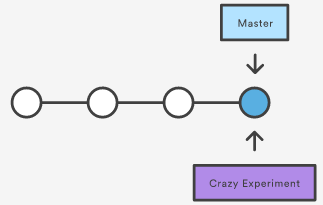
#### Create a Branch

##### Steps

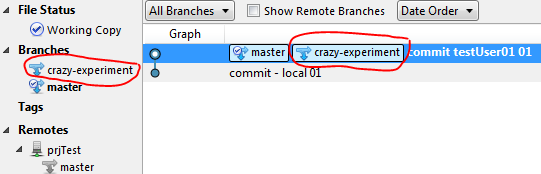
1. Click **Branch**
2. Write a branch name (e.g. crazy-experiment). Uncheck the **Checkout New Branch** checkbox



1. A branch with pointer to working copy will create.



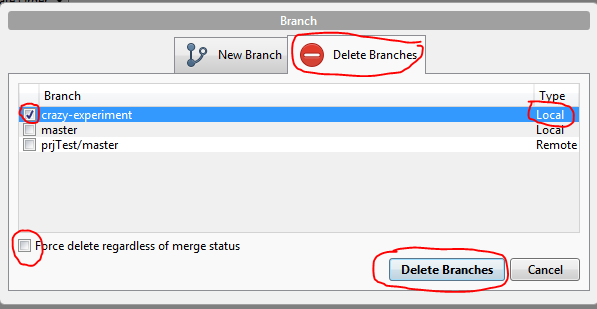
1. To working with this branch one should use the **checkout**



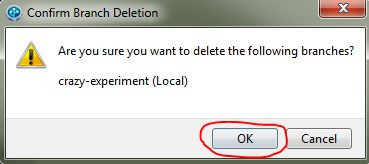
#### Delete a Branch

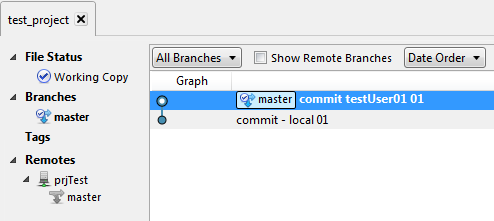
##### Steps

1. Click **Branch**
2. Select the branch to be deleted (e.g. crazy-experiment)



1. Make sure about the type of the branch (local/remote)
2. If the branch is not merged then will get error. Force delete enables to delete the branch and no concern with merge status.
3. Click **Delete Branches**
4. Click **OK** button





#### Checkout

Command  
gitcheckout

The **git checkout** command lets you navigate between the branches created by **git branch**

There is two way to use **checkout** in this regards

1. Using of Checkout **after** creating a Branch
2. Creation of Branch at the time of **Checkout**

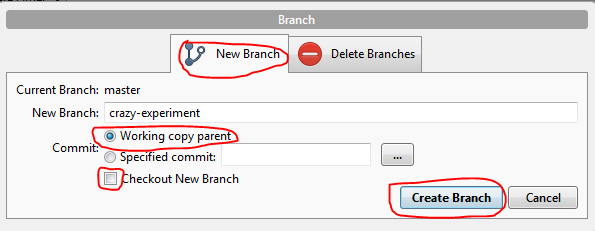
Let’s discuss both the steps

##### Using of Checkout **after** creating a Branch

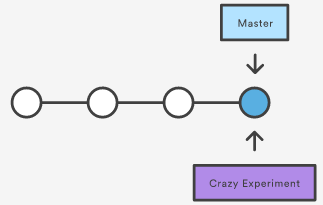
###### Steps



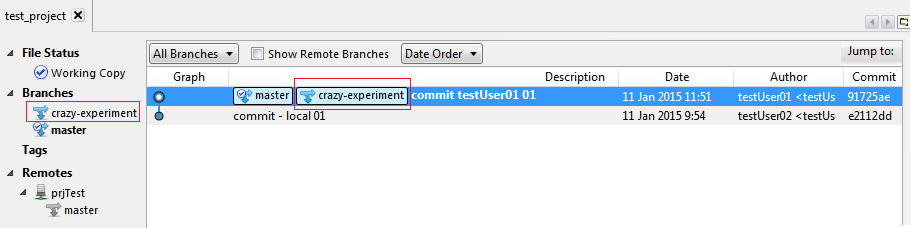
1. Click **Branch**
2. Write a branch name (e.g. crazy-experiment). Uncheck the **Checkout New Branch** checkbox



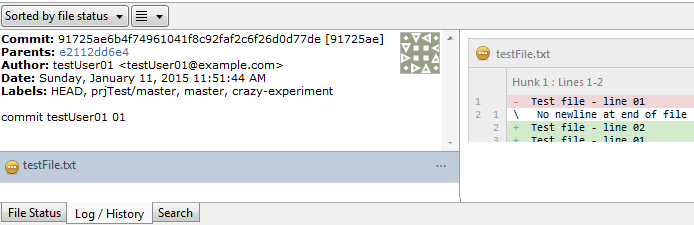
1. A branch with pointer to working copy will create.



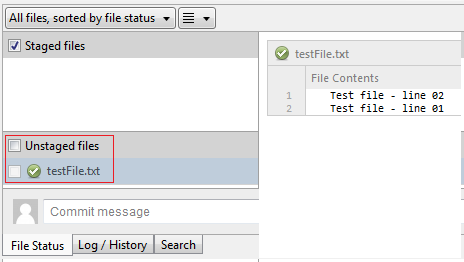
1. To working with this branch one should use the **checkout**



1. Different views
   1. **Log / History**

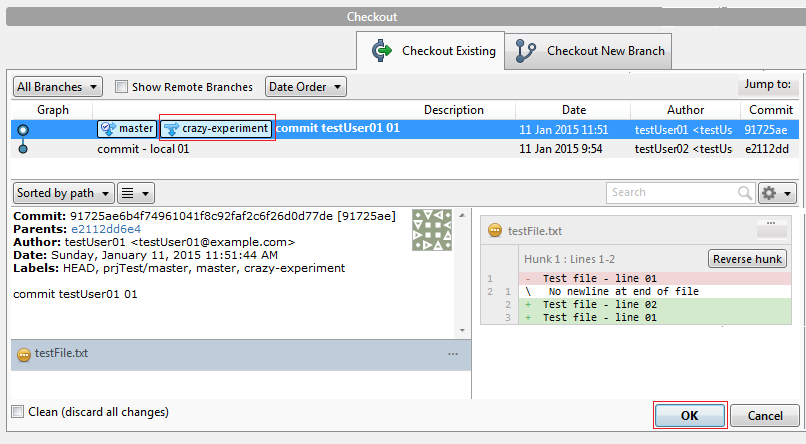


* 1. **File Status**

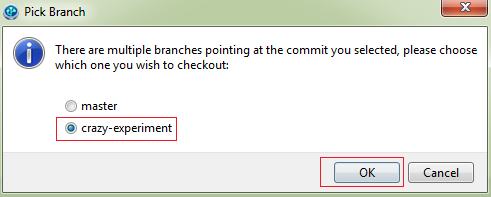




1. Click on **Checkout**

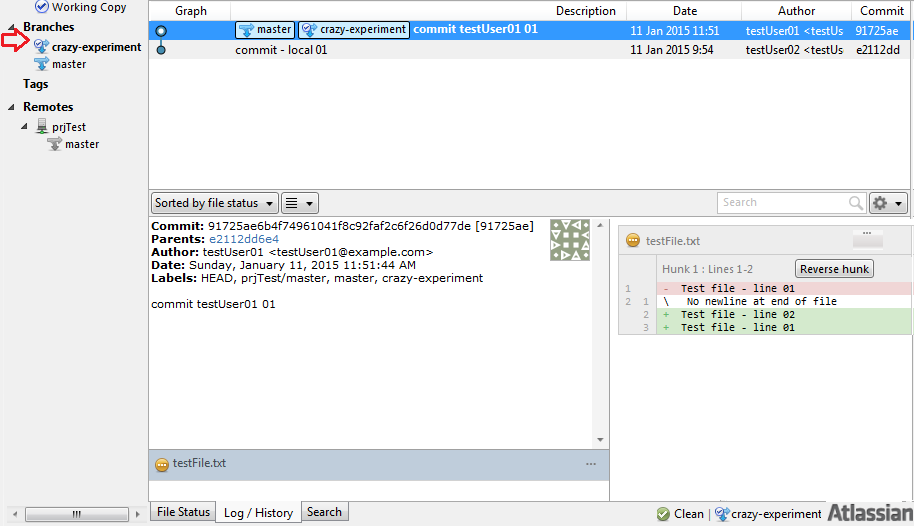


1. Select the branch at **Pick Branch** dialogue and click **OK**



1. After **checkout** the little arrow (with check mark) (i.e. ) will be shifted to

the new branch (i.e. **crazy-experiment )**

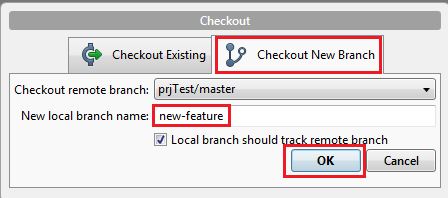


##### Creation of *Branch* **at the time** of *Checkout*

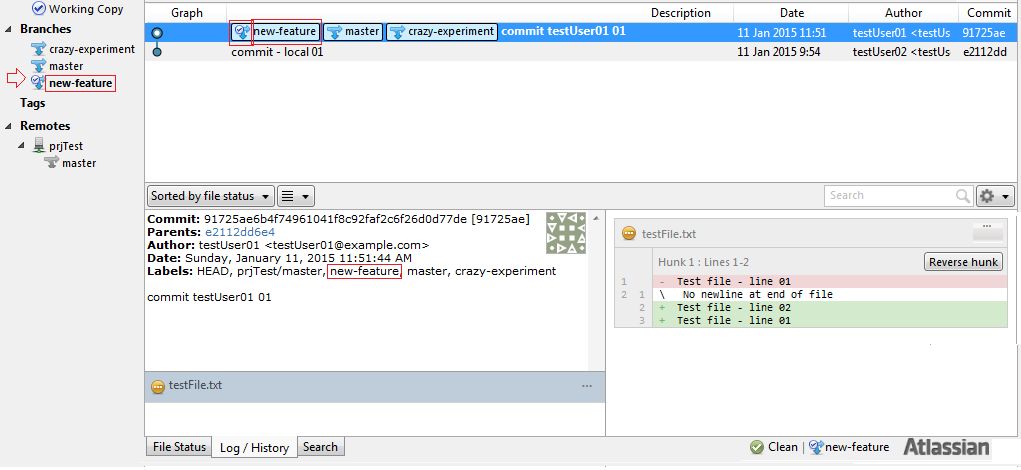
###### Steps

1. Click **Checkout**

Give a new local branch name (e.g. **new-feature**)



1. A new branch will be created and set. The little arrow will point the newly created branch



#### Merge

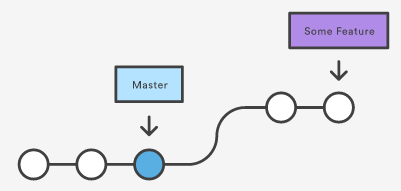
Command  
git merge

**Merging** is Git's way of putting a forked history back together again.   
  
The **git merge** command lets you take the independent lines of development created by git branch and integrate them into a single branch.

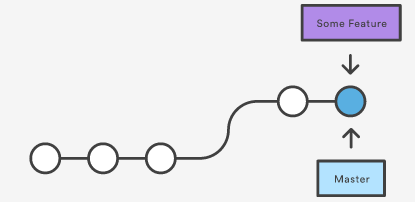
A **fast-forward merge** can occur when there is a linear path from the current branch tip to the target branch.

Instead of “actually” merging the branches, all Git has to do to ***integrate*** the histories is move (i.e., “fast forward”) the **current branch** tip up to the **target branch** tip. This effectively combines the histories, since all of the commits reachable from the target branch are now available through the current one.

Before merging



After merging

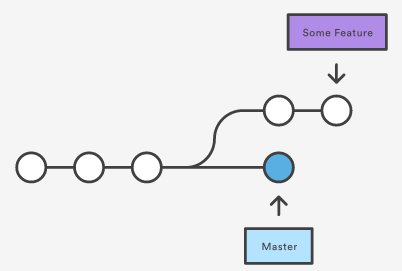


**3-Way-Merge**

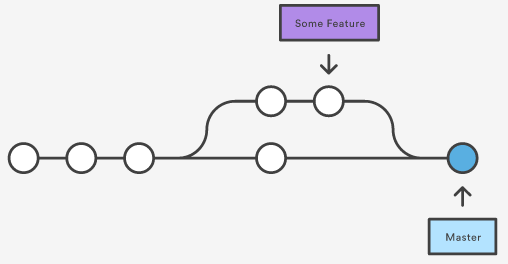
a fast-forward merge is not possible if the branches have diverged.

When there is not a linear path to the target branch, Git has no choice but to combine them via a **3-way merge**. 3-way merges use a dedicated commit to tie together the two histories. The nomenclature comes from the fact that Git uses three commits to generate the merge commit: the two branch tips and their common ancestor.

Before merging



After merging



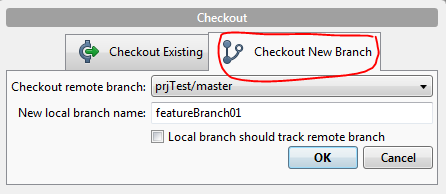
##### Fast-Forward Merge

###### Steps

***Start a new feature***

1. Click **Checkout** 

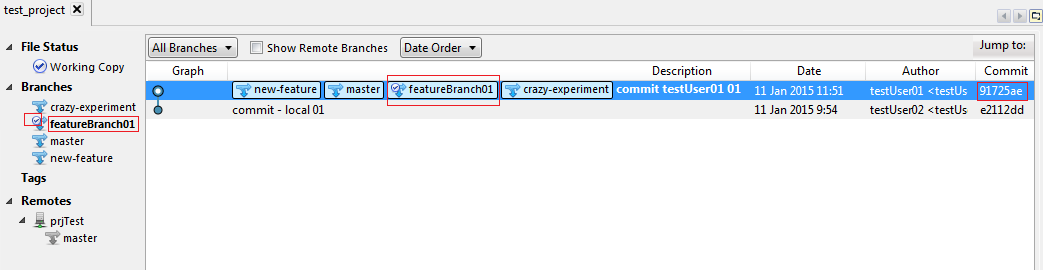
Click **Checkout New Branch** tab



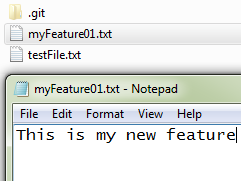
Give a suitable name of the branch (e.g. **featureBranch01 )**

Click **OK**

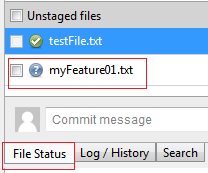
1. Newly created branch looks like below :



1. Create a file

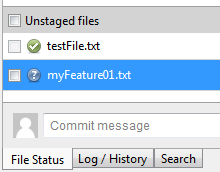


1. Newly created file will be in unstaged area

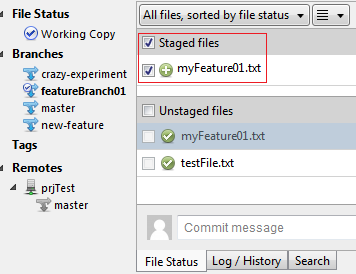




1. Select the file and click **Add for adding the** file

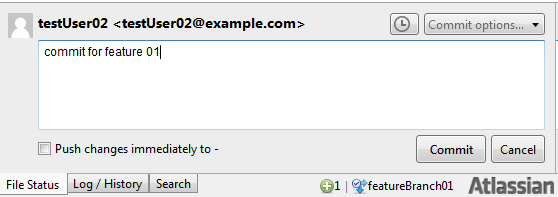


1. File will be added to the staging area



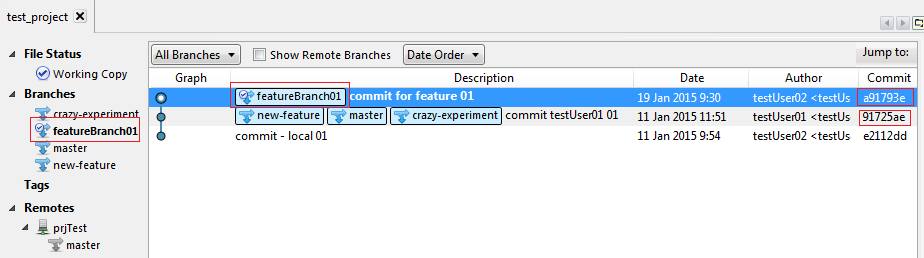


1. Click on **Commit**
2. Write comment into the comment box

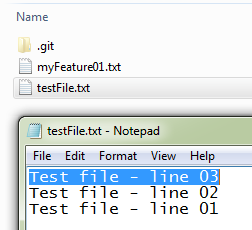


Click **Commit**

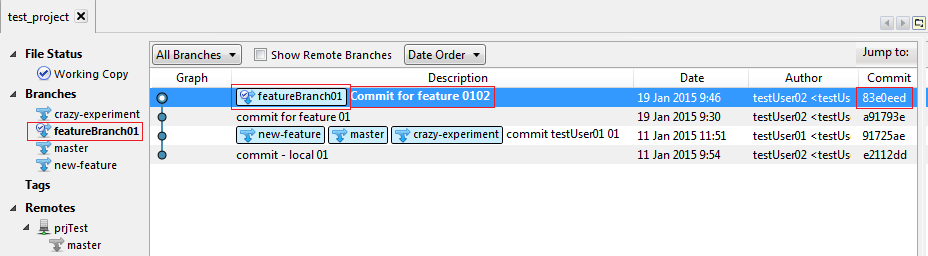
1. Log/History looks something given below



1. Modify some other file



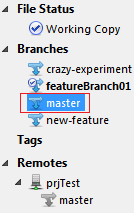
1. Add the file and commit it as described before
2. Log / History looks something like :



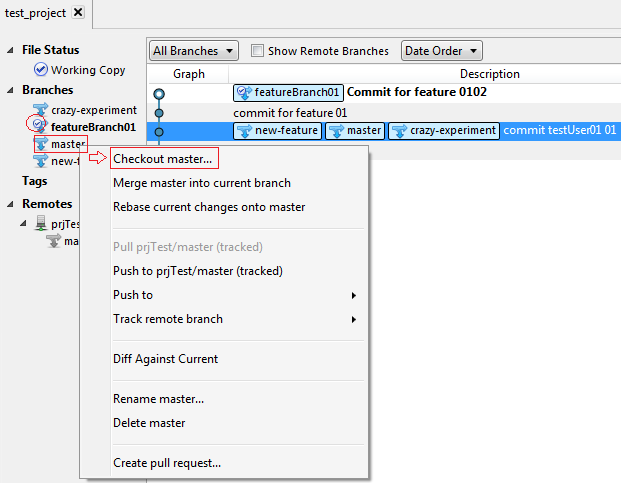
**Merge in the new-feature branch**

1. Checkout master

Select **Master** from Branches

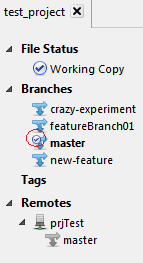


Right click after selecting master

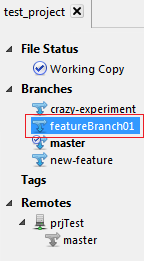


Click **Checkout master**

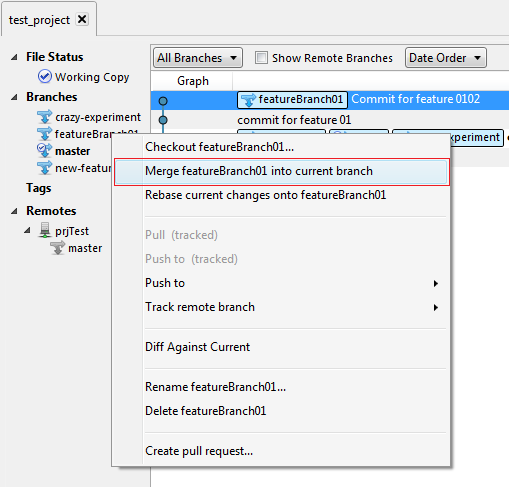
Branch view are shown below (mind the little arrow – shifted to master)



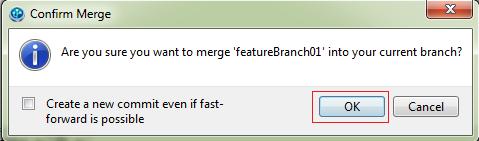
1. Select the **branch** you want to merge with master



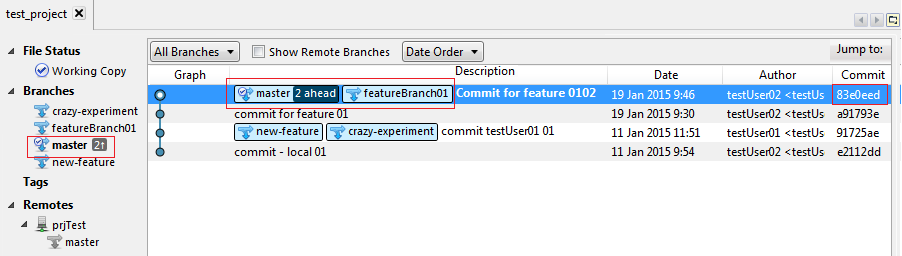
1. Right Click on selected branch



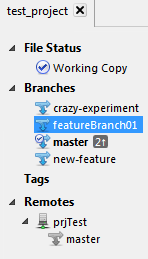
1. Click on the second option from the pull down menu i.e. **Merge <branch name> into current branch**
2. Click **OK** to the **Confirm Merge** dialogue box



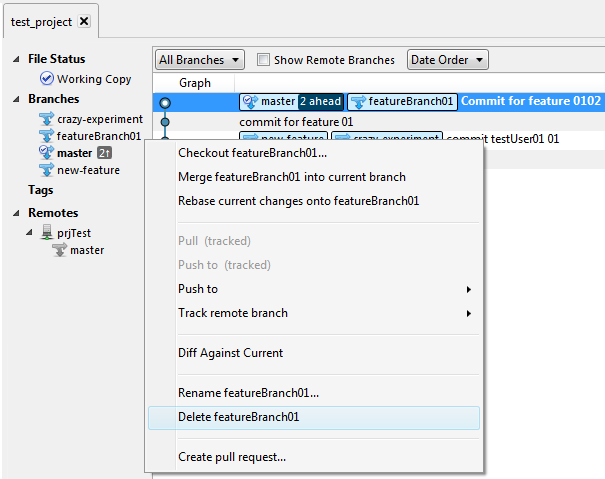
1. Log / History view will be something as shown below :



1. Now select the newly created branch (i.e. **featureBranch01**)

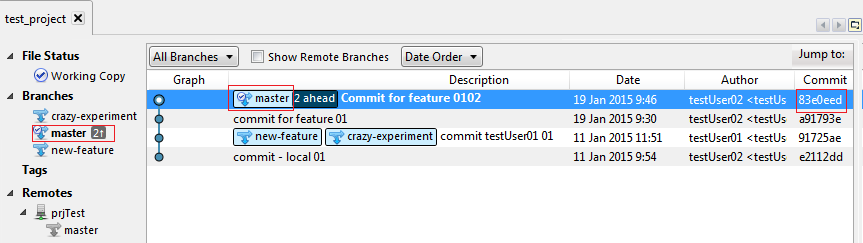


1. Right click and from pull down menu click on **Delete <branch name** i.e. featureBranch01**>**



1. Now click **OK** to confirm the delete

After delete Log / History view will be look something like this

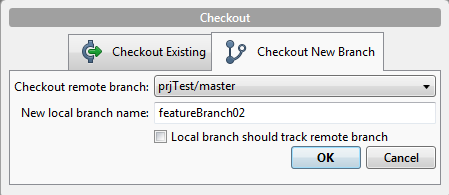


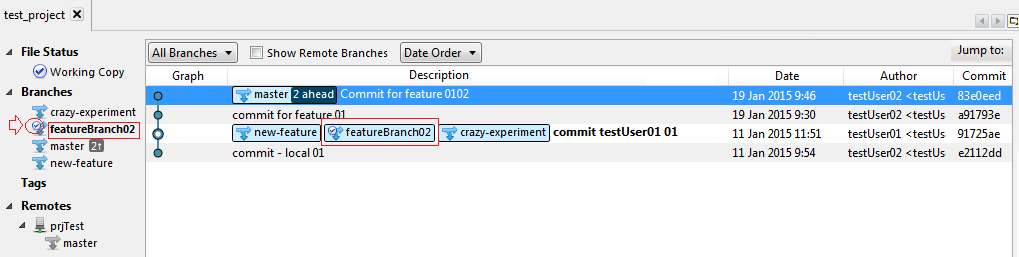
##### 3-Way Merge

###### Steps

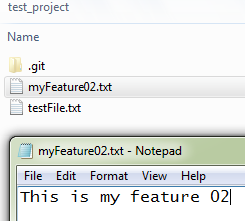
***Start new feature***

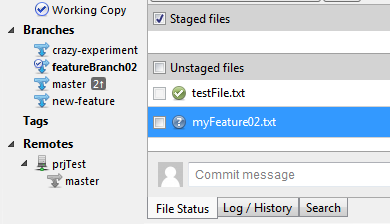
1. Create a new branch as discussed earlier (e.g. featureBranch02) using checkout





1. Create a new file (e.g. myFeature02.txt)

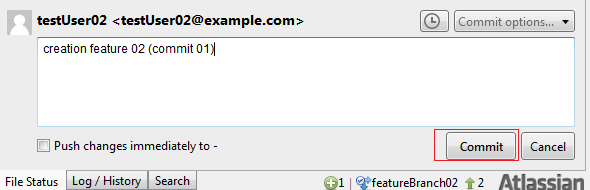


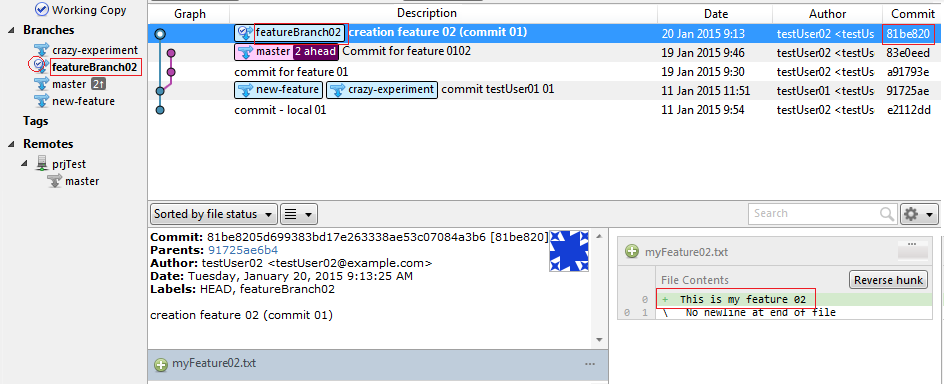
1. File will be Unstaged area

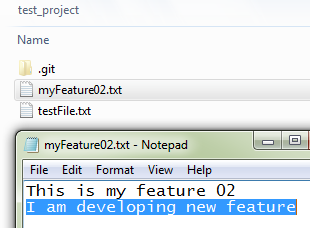


1. Add the file to staged area using **Add**

1. Commit the file using

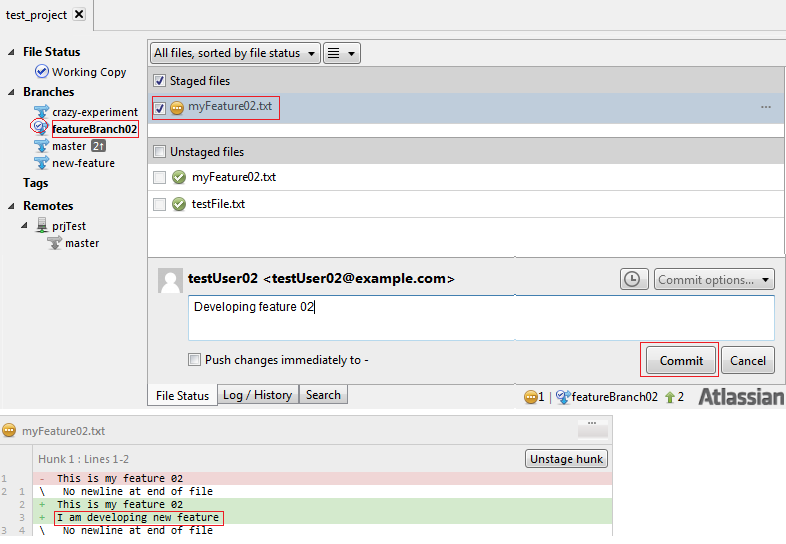


1. Log / History view
2. Modify the newly created text file (e.g. myFeature02.txt)



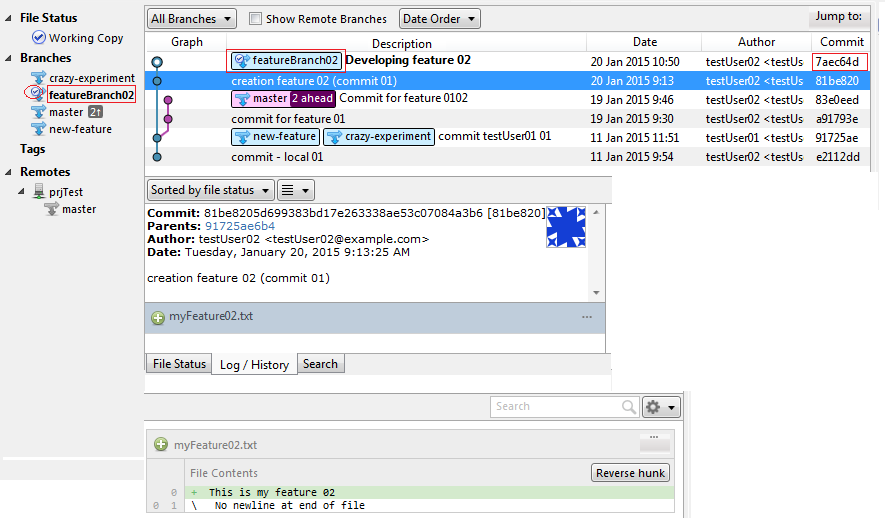


1. Add the file using Add



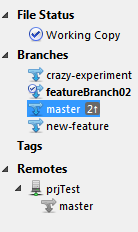
1. Commit the changes

After commit the Log/History view

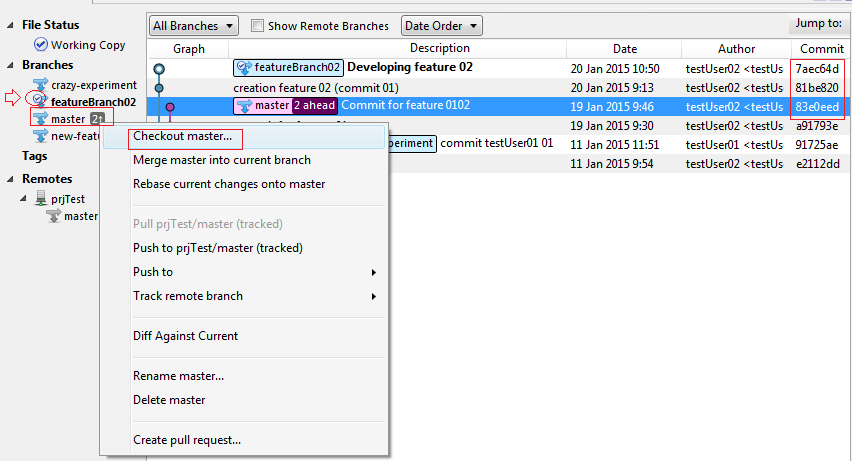


**Merge in the new-feature branch**

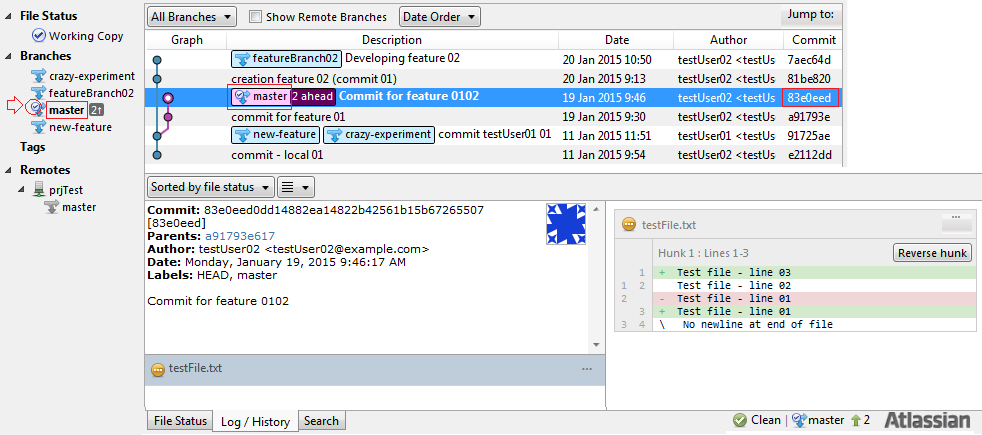
1. Point to **master**
   1. Select **Master**



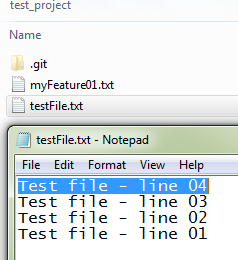
* 1. Right Click; and Click **Check Out Master** from the pull down menu



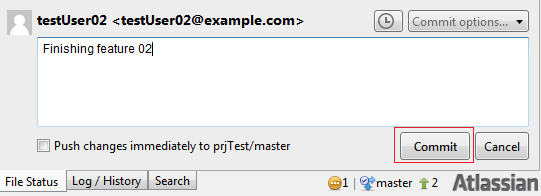
* 1. Log / History view



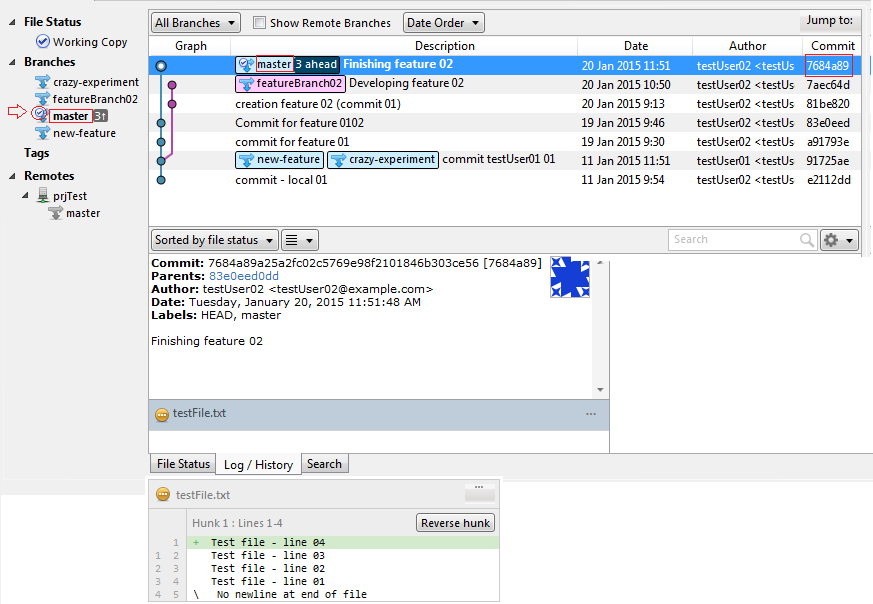
1. Save some changes in any file (e.g. testFile.txt)



1. Add the changes using **Add**
2. Commit the master using **commit**



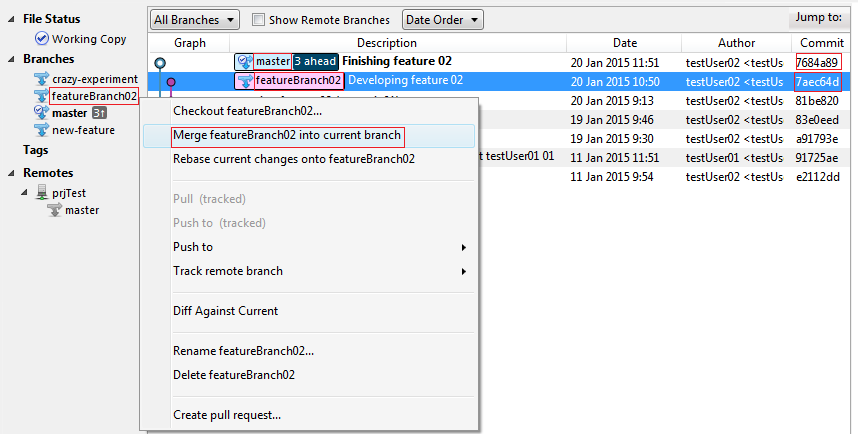
1. Log / History status



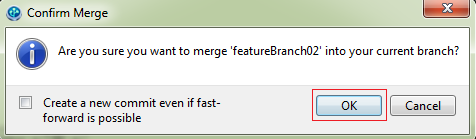
Merge with earlier created branch ( e.g. featureBranch02)

Select the required branch ( e.g. featureBranch02)

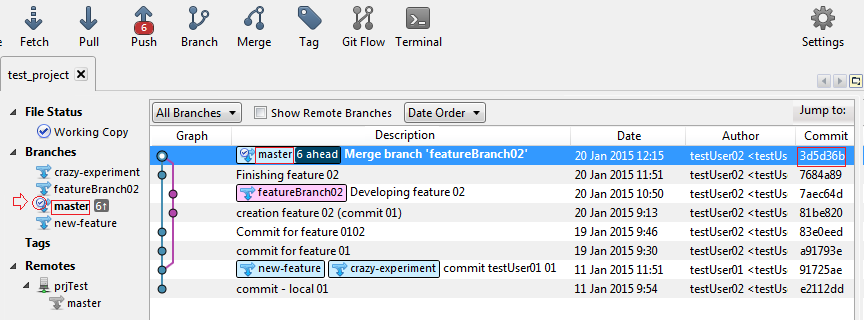
1. Right click and from pull down menu select **Merge <branch name> into current branch** (e.g. here branch name used is featureBrach02)



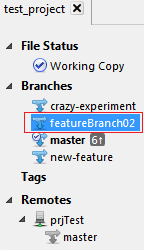
1. Click on **OK** on Confirm Message dialogue



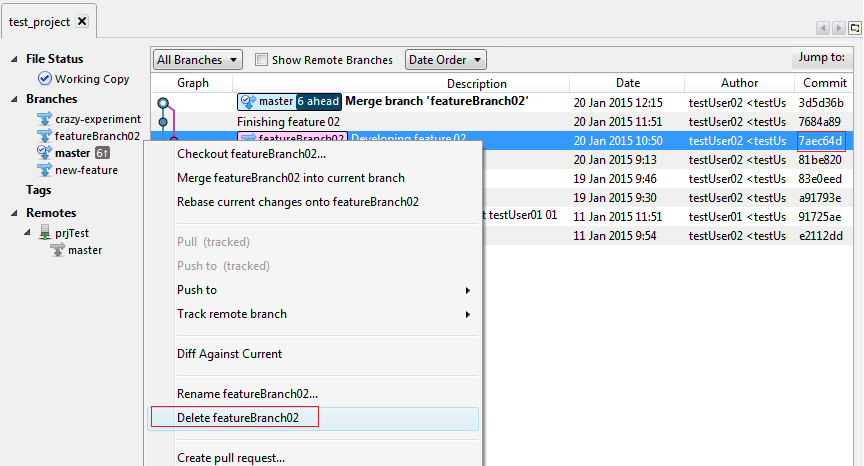
1. Log/History View



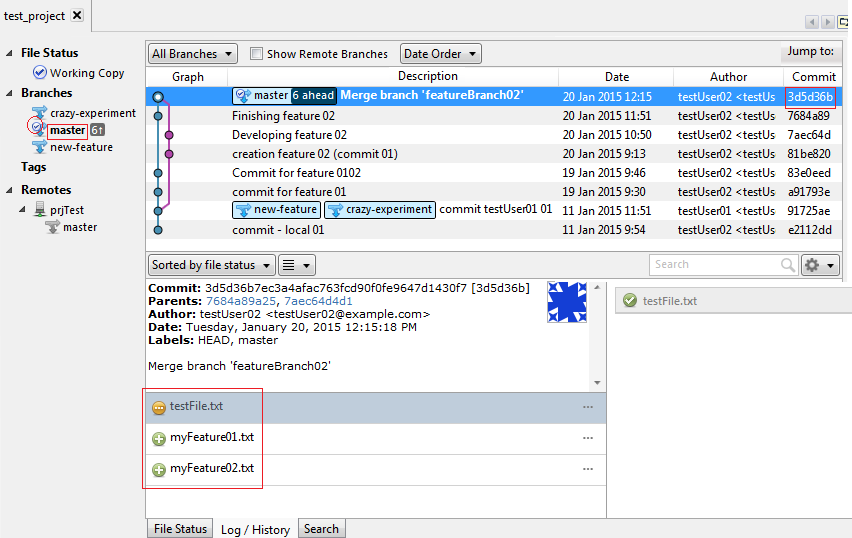
1. You might delete the branch you created earlier (e.g. featureBranch02)
2. Select the branch to be deleted (e.g. featureBranch02)



1. Right click and from pull down menu select **Delete <branch name>** (e.g. branch name for this example is featureBranch02)



1. Click on **OK** to confirm branch deletion
2. Log / History View



References :

1. <https://www.atlassian.com/git/tutorials/>
2. <http://git-scm.com/>
3. <http://www.sourcetreeapp.com/>
4. <https://about.gitlab.com/>