

GIT Tutorial

(Beginners)



Using GitLab & Source Tree

Step by step hands on guide on git.

Author : Bhabesh

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



Preface

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 familiarise themselves with source control management using Git.

Most of the cases beginners worried about the command line interfaces specially who are not familiar 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. However 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

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

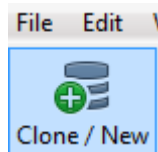
Preface	0
Terms	2
Table of Contents.....	3
Getting Started.....	4
01. Setting up a repository (Local)	4
Steps.....	4
02.Setting up a repository (Remote)	5
Steps.....	5
03.Configuration of Repository.....	6
Steps.....	6
Command.....	7
04.Saving changes.....	7
Git Add	7
05.Git Commit.....	9
06.Viewing Old Commits.....	11
Steps.....	11
Undoing Changes	14
07.Revert.....	14
08.Reset	17
Collaborating.....	23
Syncing	23
11.Remote.....	23
12.Push.....	26
13.Fetch	28
13.Pull	30
Using Branch	31
Branch	31

Getting Started

Command

`git init`

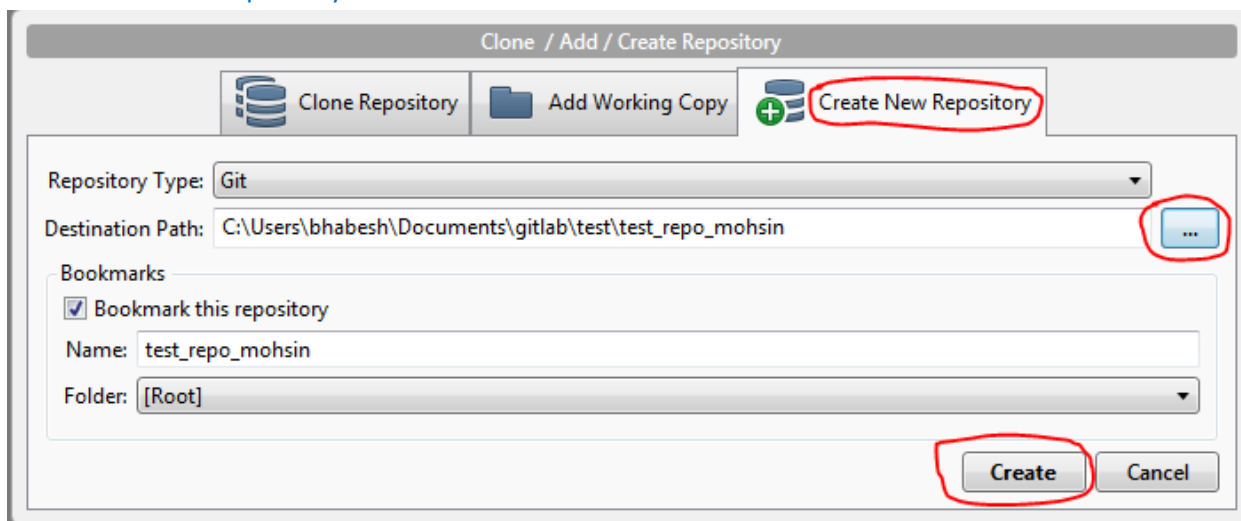
01. Setting up a repository (Local)



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

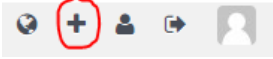
Destination Path : <Folder where you want your local repo>

Bookmark this repository: check the box

Click on **Create**

02.Setting up a repository (Remote)

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

Command

```
git clone
```

Project name

Namespace

☒ Customize repository name?

☐ Import existing repository?

Description (optional)

Visibility Level (?)


- ☒ **Private**
Project access must be granted explicitly for each user.
- ☐ **Internal**
The project can be cloned by any logged in user.
- ☐ **Public**
The project can be cloned without any authentication.

Need a group for several dependent projects?

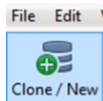
4. Click **Create Project**
5. Click http and copy the url

Issues 0 Wiki Settings

SSH HTTP

6. In your local machine select Source Tree 

7. Click Clone / New



8. Paste the url in at the Source Path / URL at the dialogue box and then click **Clone**

Clone / Add / Create Repository

☒ Clone Repository ☐ Add Working Copy ☐ Create New Repository

Source Path / URL:

Repository Type: ☒ This is a Git repository

Destination Path:

Advanced Options

Bookmarks

☒ Bookmark this repository

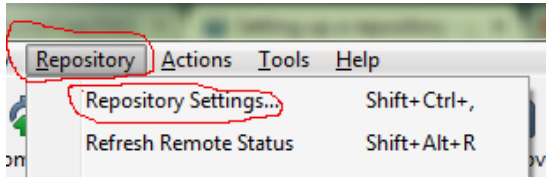
Name:

Folder:

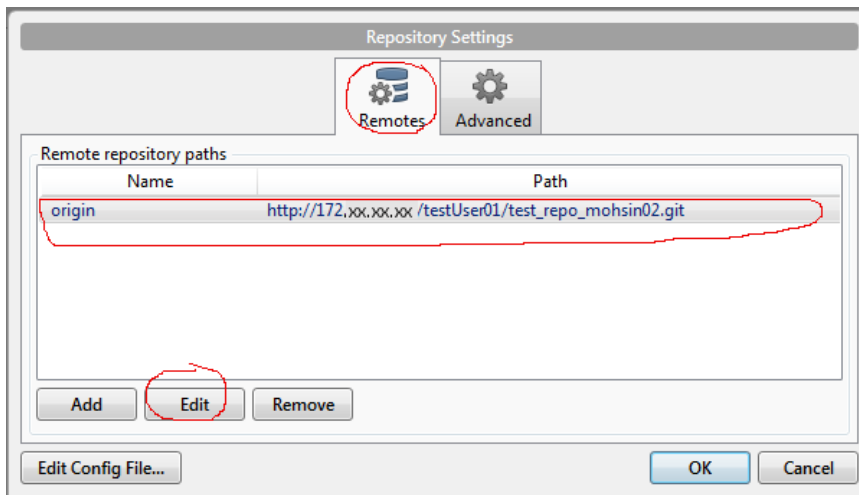
03.Configuration of Repository

Steps

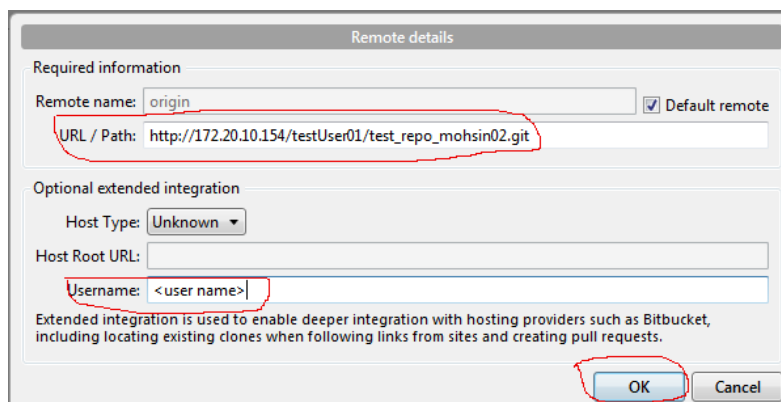
1. Select the repository using double click at the source tree
2. Click **Repository/Repository Settings..**



3. Click
 - a) **Remote** tab ,
 - b) Select the existing (required) row from **Remote repository paths**
 - c) Click **Edit** button



4. Make sure that
 - URL/Path** : valid git url
 - Username** : git user name (active directory user name)Click **Ok**



Command

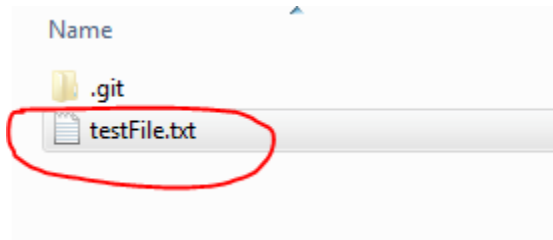
git config

04.Saving changes

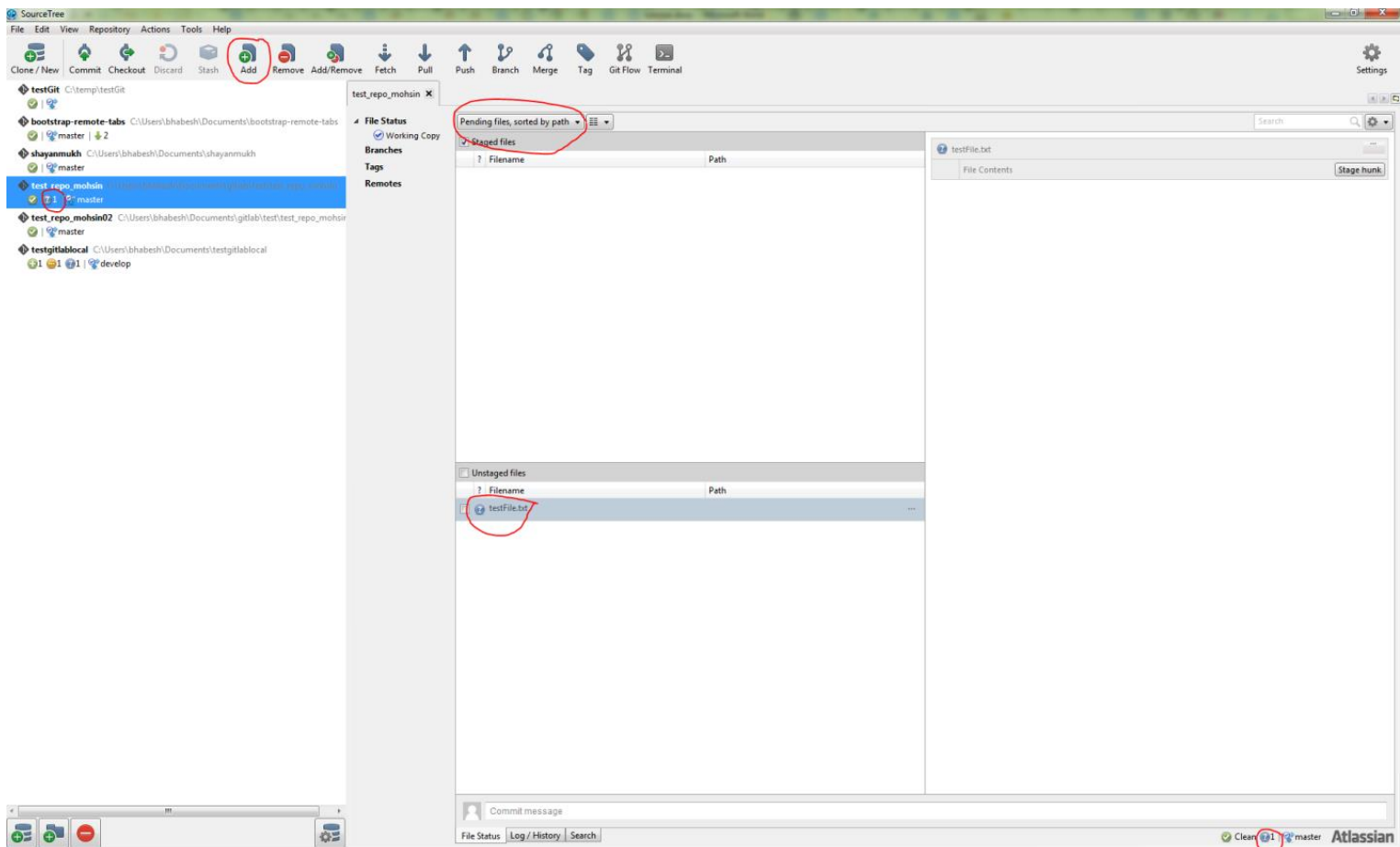
Git Add

Steps

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



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

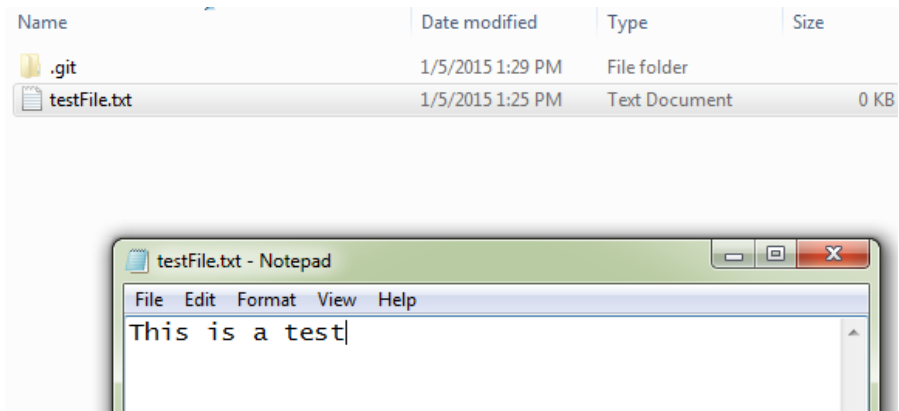


4. File will be added at staging

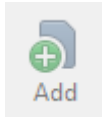
Command

git add

5. Open the file and add some text



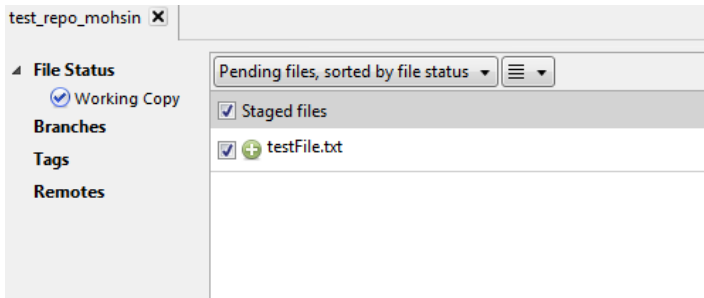
- 6. File will again visible at 'Unstaged Files' area with **yellow** color
- 7. Select the file and click add (if needed)
- 8. Changes will be saved at staging area



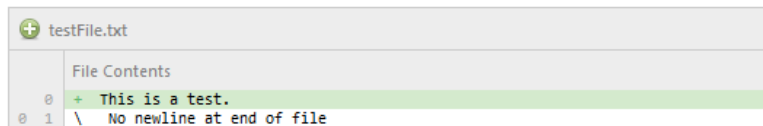
05.Git Commit

Steps

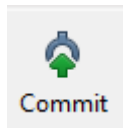
1. Select file/s from 'Staged Files'



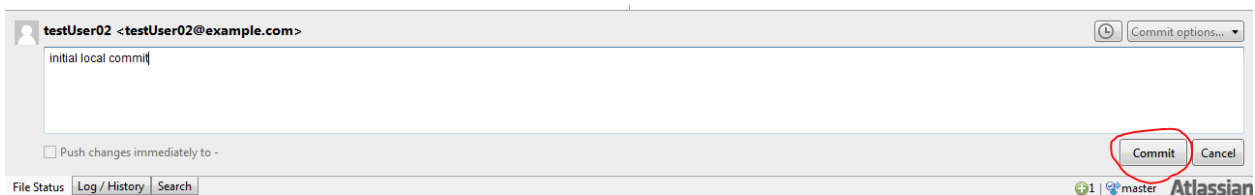
2. Check the changes



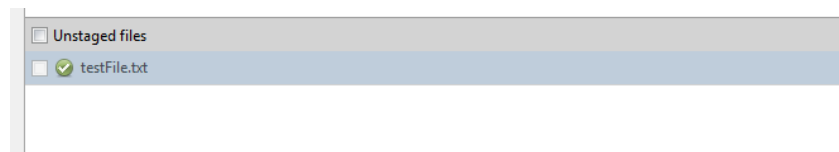
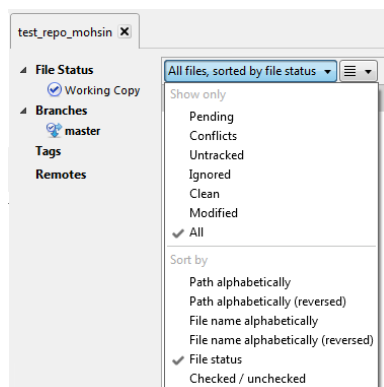
3. Click on commit



4. Give some comment/justification of commit in commit dialogue box and click **commit** button



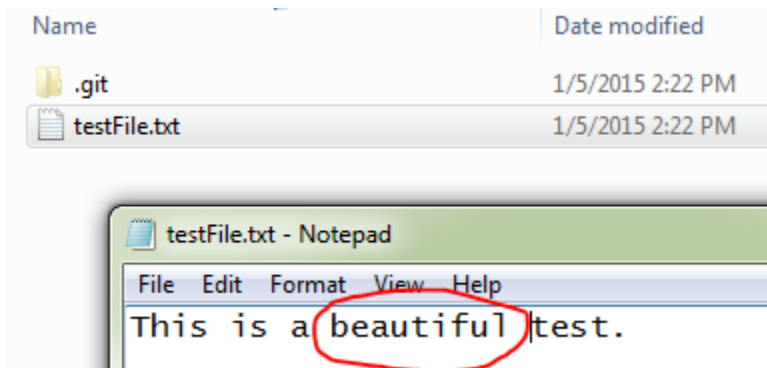
5. File will be committed. It can be viewed as below :



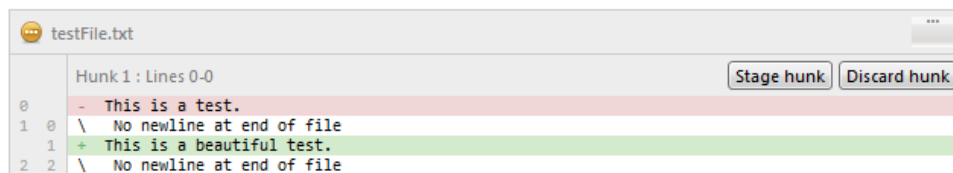
Command

git commit

6. Modify some text

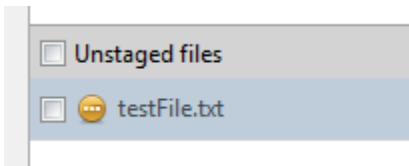


7. In SourceTree the changes are like this



8. It is possible to stage only the portion or full file

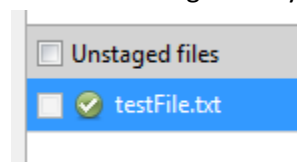
9. Click the file at Unstage area ('git add' command will be applied) .



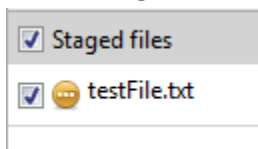
10. After add file will be staged and file color will change in two places

a. In 'Unstaged files' area

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



b. And in 'Staged files' area will change from green to yellow (with icon changes)

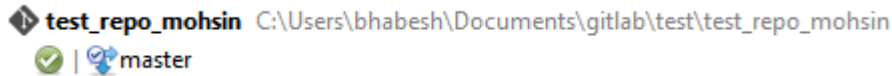


11. Select the file and commit. File will be committed locally.

06.Viewing Old Commits

Steps

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



Command

git checkout

3. See the commit log

All Branches

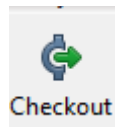
Show Remote Branches

Date Order

Jump to:


Graph	Description	Date	Author	Commit
<div> <div>master</div> <div>file modified</div> </div>		5 Jan 2015 14:42	testUser02 <testUs	9ff98ab
<div> <div></div> <div>initial local commit</div> </div>		5 Jan 2015 14:06	testUser02 <testUs	fc863e3


4. Click on **checkout**



5. Select the particular commit

Checkout

 Checkout Existing



 Checkout New Branch

All Branches

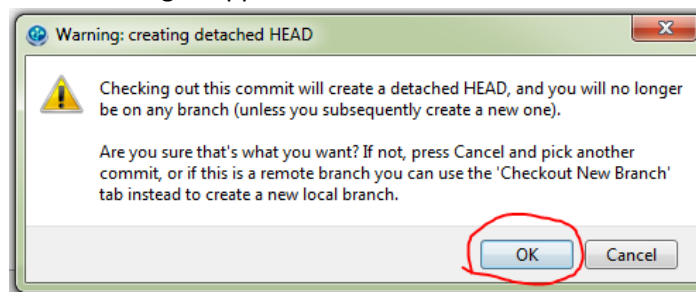
Show Remote Branches

Date Order

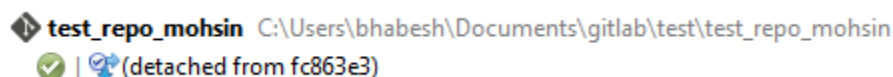
Jump to:

Graph	Description	Date	Author	Commit
 master	file modified	5 Jan 2015 14:42	testUser02 <testUs	9ff98ab
	initial local commit	5 Jan 2015 14:06	testUser02 <testUs	fc863e3

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

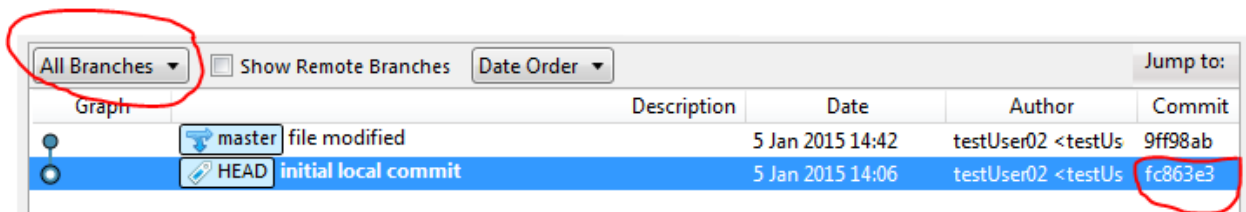


7. BookMark pane looks something similar

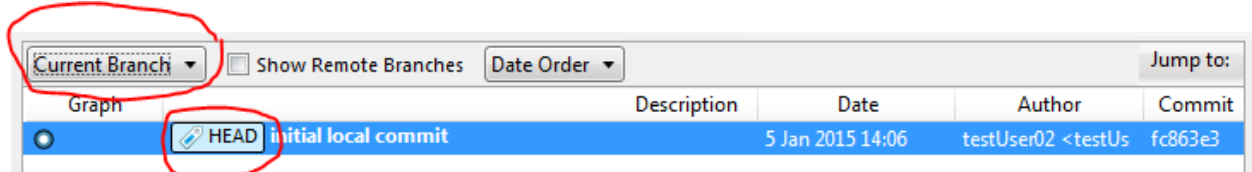
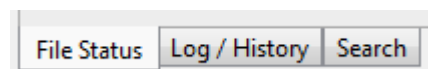


8. Right side pane for log :

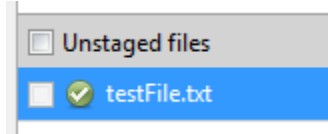
a. For all Branch



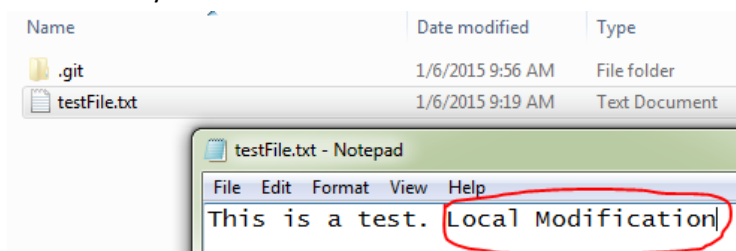
b. For Current Branch

9. Click on **File Status**

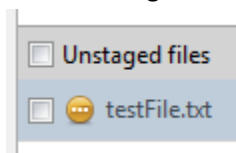
10. Unstaged files listing



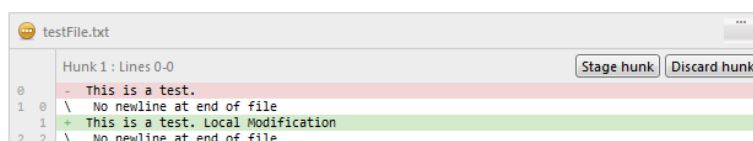
11. Open the file and modify



12. Unstaged files listing

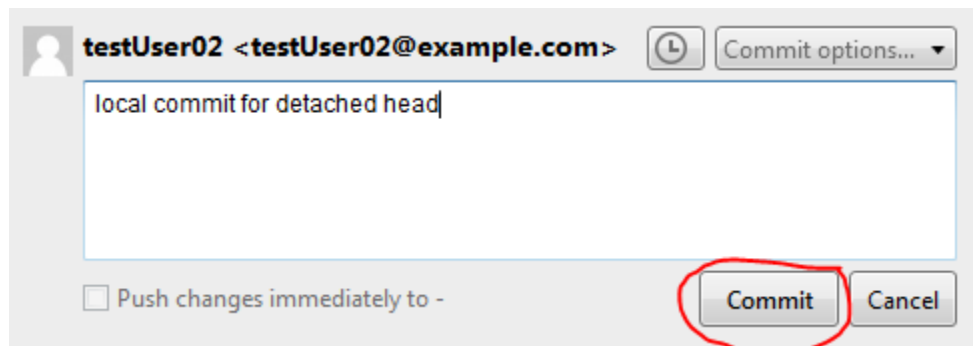
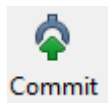


13. Changes in file



14. Check the file in **Unstaged files** pane to add to staged area

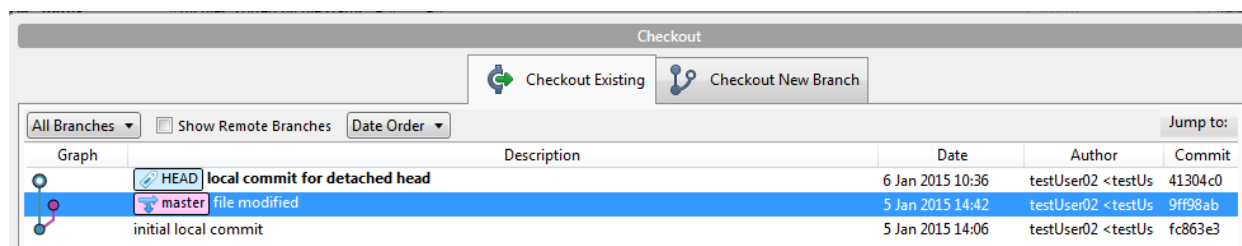
15. Click on **Commit**



16. Click on **CheckOut**



17. Select **Master**



18. Click **OK**

19. You will find your earlier work is not modified.

20. It proves checkout an earlier commit will not disturb the present master branch.

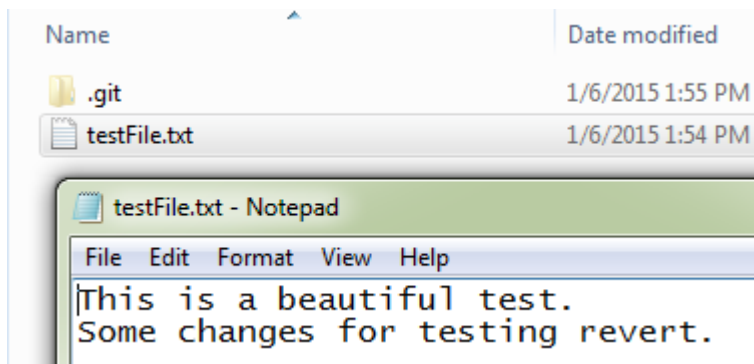
21. However this is not the case of checkout a file.

Undoing Changes

07.Revert

Steps

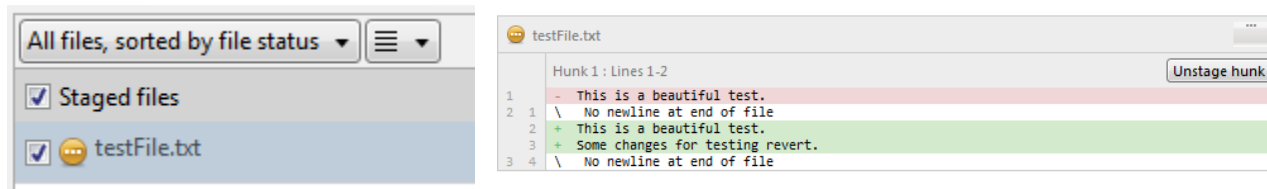
1. Open the file and do some changes



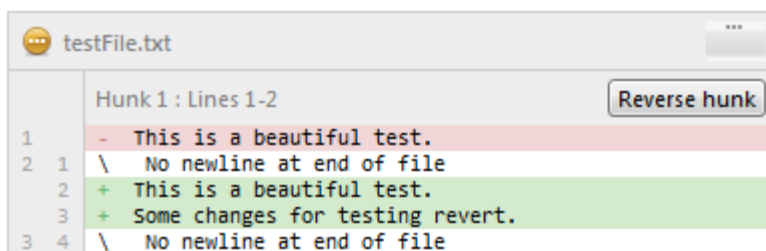
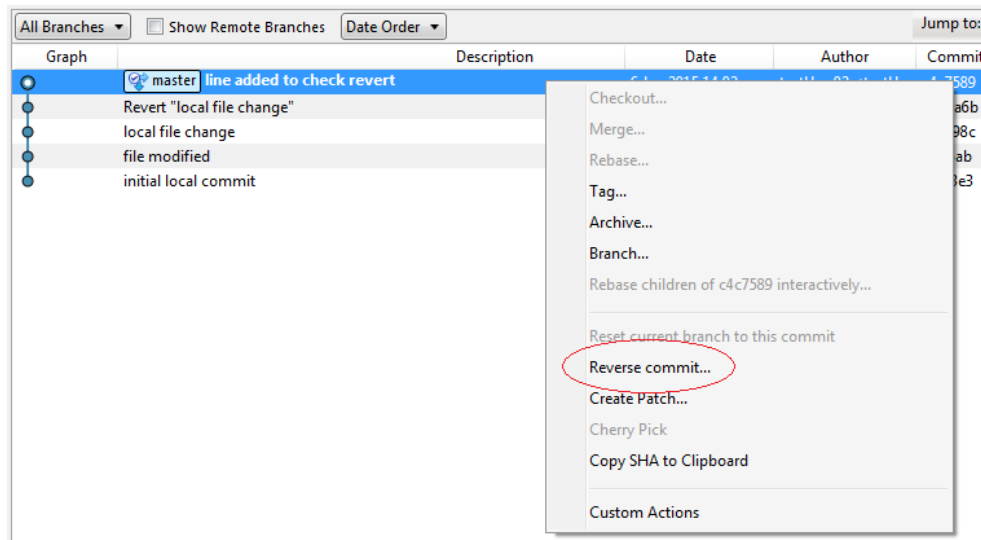
Command

git revert

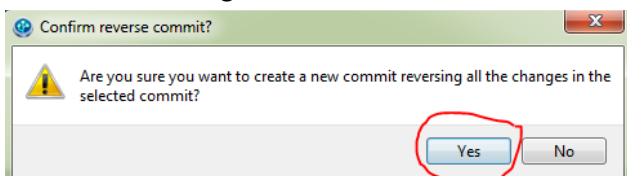
2. Commit it










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



4. Click **Yes** on Dialogue box



5. The contents will be earlier commit stage

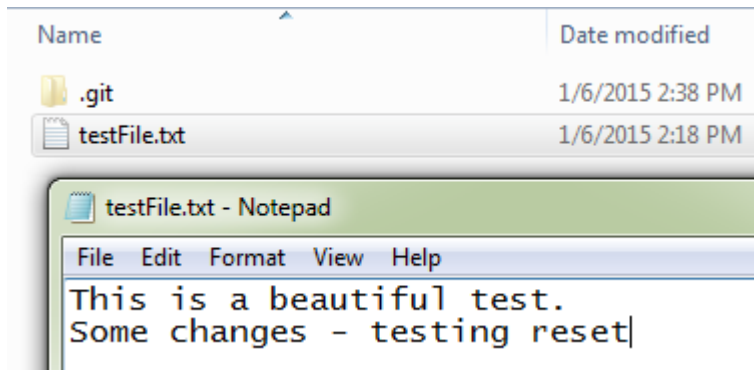
All Branches ▾		<input type="checkbox"/> Show Remote Branches	Date Order ▾	Jump to:	
Graph		Description	Date	Author	Commit
	 master	Revert "line added to check revert"	6 Jan 2015 14:18	testUser02 <testUs	97af81b
		line added to check revert	6 Jan 2015 14:02	testUser02 <testUs	c4c7589
		Revert "local file change"	6 Jan 2015 12:56	testUser02 <testUs	466da6b
		local file change	6 Jan 2015 12:54	testUser02 <testUs	021298c
		file modified	5 Jan 2015 14:42	testUser02 <testUs	9ff98ab
		initial local commit	5 Jan 2015 14:06	testUser02 <testUs	fc863e3

testFile.txt		
Hunk 1 : Lines 0-0		Reverse hunk
0	-	This is a beautiful test.
1	-	Some changes for testing revert.
2	0	\ No newline at end of file
1	+	This is a beautiful test.
3	2	\ No newline at end of file

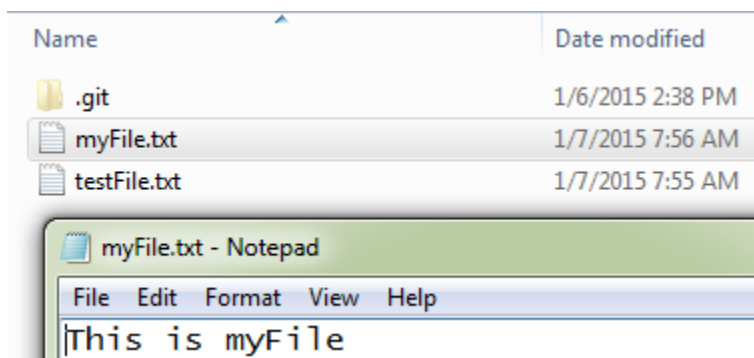
08.Reset

Steps

1. Modify testFile.txt



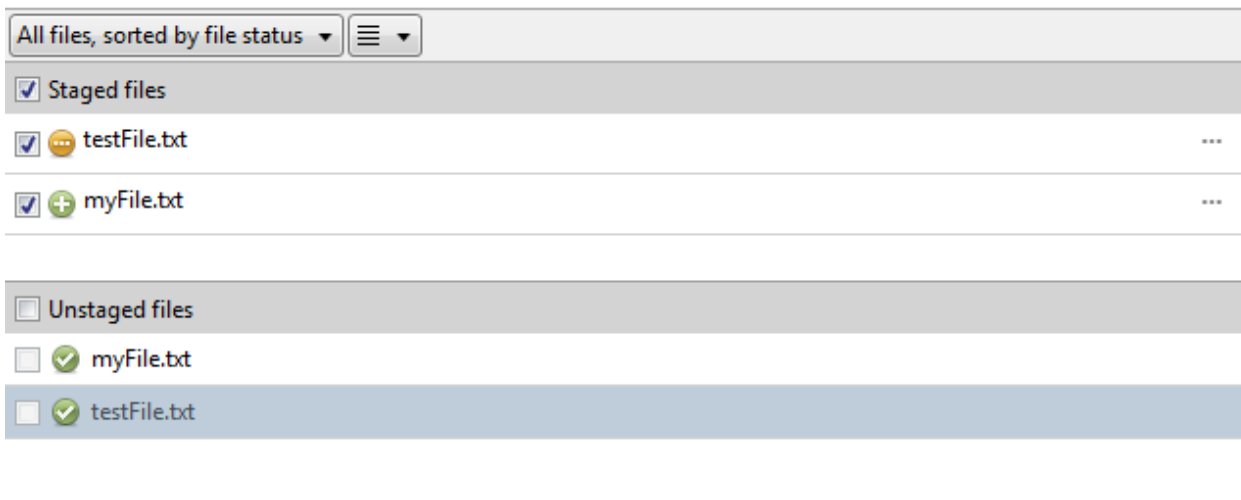
2. Add another File



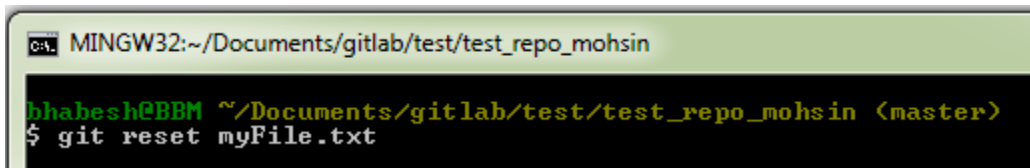
Command

```
git reset
```

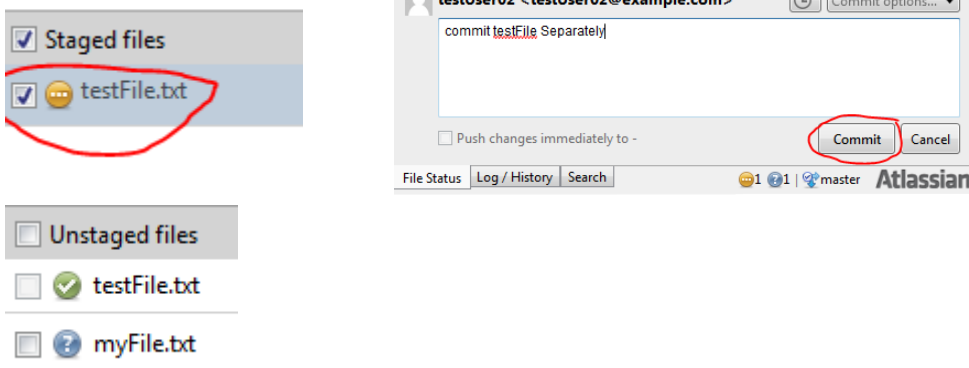
3. Add for staging



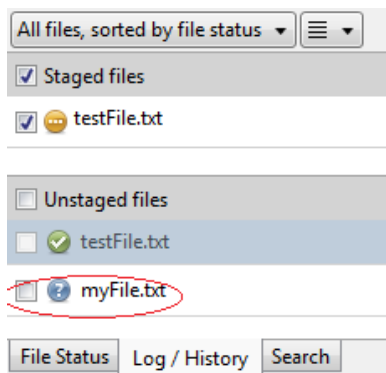
4. Reset the file (e.g. myFile.txt) .. It will be unstaged



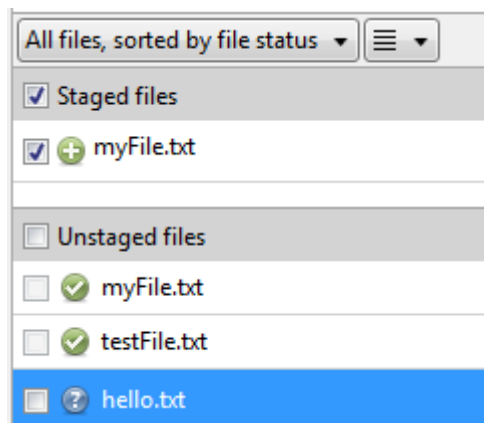
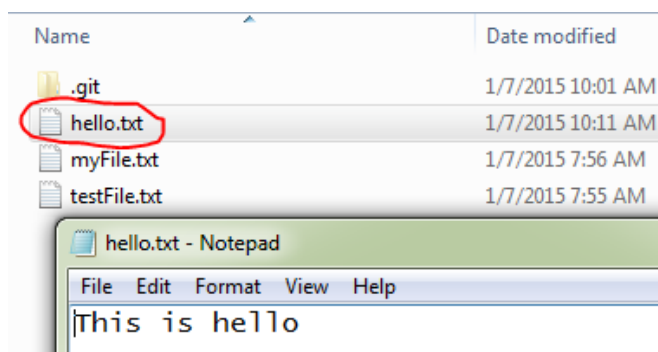
5. Commit Files seperately



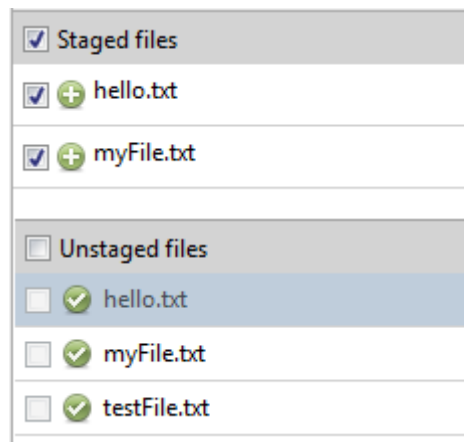
6. Add the other file from unstaging area



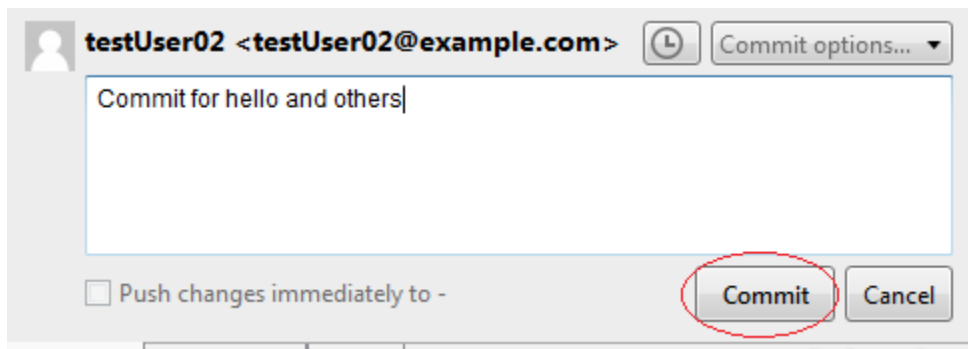
7. Create another file (e.g. hello.txt)



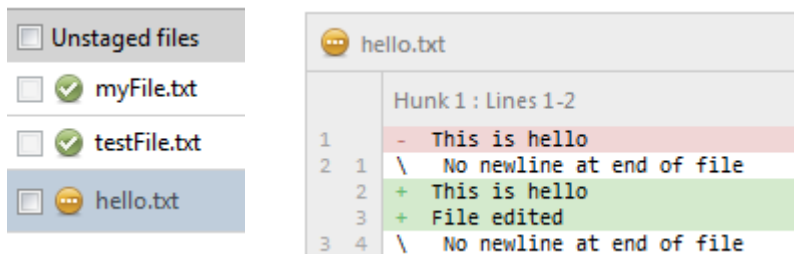
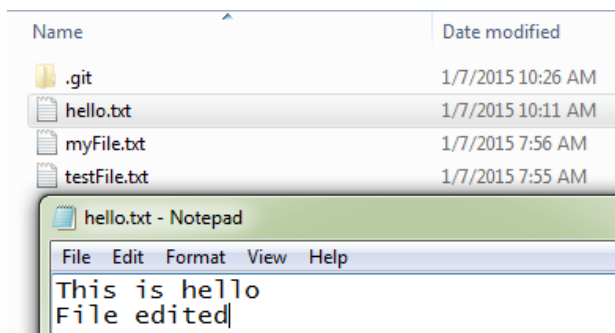
8. Now Add the file (from Unstaged to staged): Click



9. Commit :



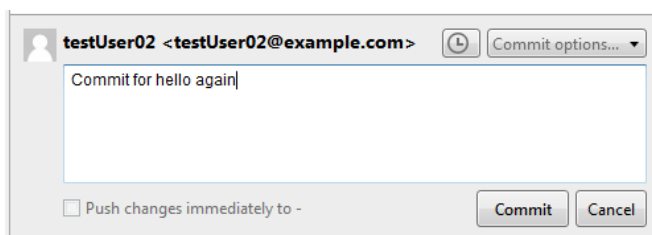
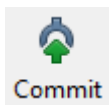
10. Edit the newly committed file (e.g. hello.txt) again :













11. Add the file to staging area



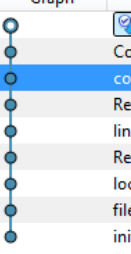









12. Commit :



13. Commit logs :

All Branches ▾		Show Remote Branches	Date Order ▾	Jump to:	
Graph		Description	Date	Author	Commit
	 master	Commit for hello again	7 Jan 2015 10:41	testUser02 <testUs	228f178
		Commit for hello and others	7 Jan 2015 10:25	testUser02 <testUs	9b27528
		commit testFile Separately	7 Jan 2015 9:56	testUser02 <testUs	ce58483
		Revert "line added to check revert"	6 Jan 2015 14:18	testUser02 <testUs	97af81b
		line added to check revert	6 Jan 2015 14:02	testUser02 <testUs	c4c7589
		Revert "local file change"	6 Jan 2015 12:56	testUser02 <testUs	466da6b
		local file change	6 Jan 2015 12:54	testUser02 <testUs	021298c
		file modified	5 Jan 2015 14:42	testUser02 <testUs	9ff98ab
		initial local commit	5 Jan 2015 14:06	testUser02 <testUs	fc863e3

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

All Branches ▾		Show Remote Branches	Date Order ▾	Jump to:	
Graph		Description	Date	Author	Comm
	 master	Commit for hello again	7 Jan 2015 10:41	testUser02 <testUs	228f178
		Commit for hello and others	7 Jan 2015 10:25	testUser02 <testUs	9b27528
		commit testFile Separately	7 Jan 2015 9:56	testUser02 <testUs	ce58483
		Revert "line added to check revert"	6 Jan 2015 14:18	testUser02 <testUs	97af81b
		line added to check revert	6 Jan 2015 14:02	testUser02 <testUs	c4c7589
		Revert "local file change"	6 Jan 2015 12:56	testUser02 <testUs	466da6b
		local file change	6 Jan 2015 12:54	testUser02 <testUs	021298c
		file modified	5 Jan 2015 14:42	testUser02 <testUs	9ff98ab
		initial local commit	5 Jan 2015 14:06	testUser02 <testUs	fc863e3

- Checkout...
- Merge...
- Rebase...
- Tag...
- Archive...
- Branch...
- Rebase children of ce58483 interactively...
- Reset current branch to this commit**
- Reverse commit...
- Create Patch...
- Cherry Pick
- Copy SHA to Clipboard
- Custom Actions

15. Click **OK** to Warning notice

Reset to Commit...

Are you sure you want to move the branch pointer?

Reset branch: master

To commit: ce58483: commit testFile Separately

Using mode: Hard - discard all working copy changes

OK Cancel









Warning: destructive operation

Using the 'Hard' reset mode will discard all your local changes in the working copy and the index. Are you sure this is what you want?

Yes No

16. Commit Logs : (addition and modification lost)

Current status lost

All Branches ▾		<input type="checkbox"/> Show Remote Branches	Date Order ▾	Jump to:	
Graph	Description	Date	Author	Commit	
  master	commit testFile Separately	7 Jan 2015 9:56	testUser02 <testUs	ce58483	
	Revert "line added to check revert"	6 Jan 2015 14:18	testUser02 <testUs	97af81b	
	line added to check revert	6 Jan 2015 14:02	testUser02 <testUs	c4c7589	
	Revert "local file change"	6 Jan 2015 12:56	testUser02 <testUs	466da6b	
	local file change	6 Jan 2015 12:54	testUser02 <testUs	021298c	
	file modified	5 Jan 2015 14:42	testUser02 <testUs	9ff98ab	
	initial local commit	5 Jan 2015 14:06	testUser02 <testUs	fc863e3	

Collaborating

Syncing

11.Remote

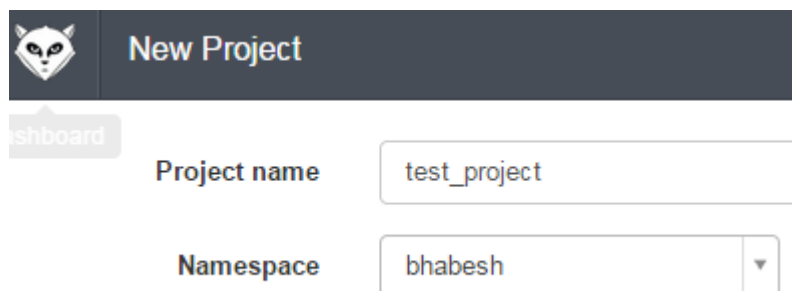
Command

`git 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



New Project

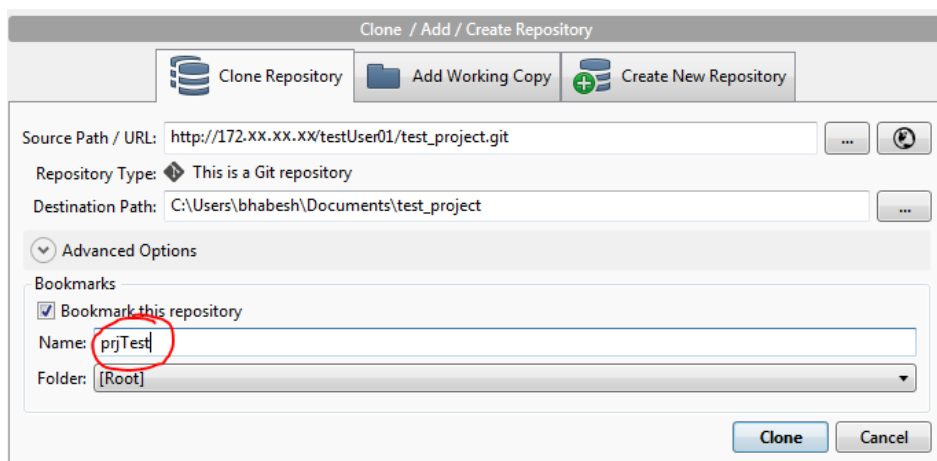
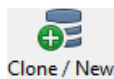
Project name: test_project

Namespace: bhabesh

2. Copy the url

SSH HTTP `http://172.xx.xx.xx/testUser01/test_project.git`

3. Click **Clone/New**



Clone / Add / Create Repository

Clone Repository Add Working Copy Create New Repository

Source Path / URL: `http://172.xx.xx.xx/testUser01/test_project.git`

Repository Type: This is a Git repository

Destination Path: `C:\Users\bhabesh\Documents\test_project`

Advanced Options

Bookmarks

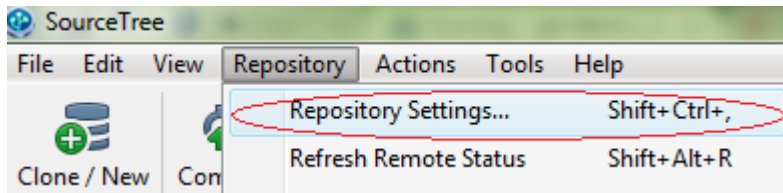
☒ Bookmark this repository

Name: `prjTest`

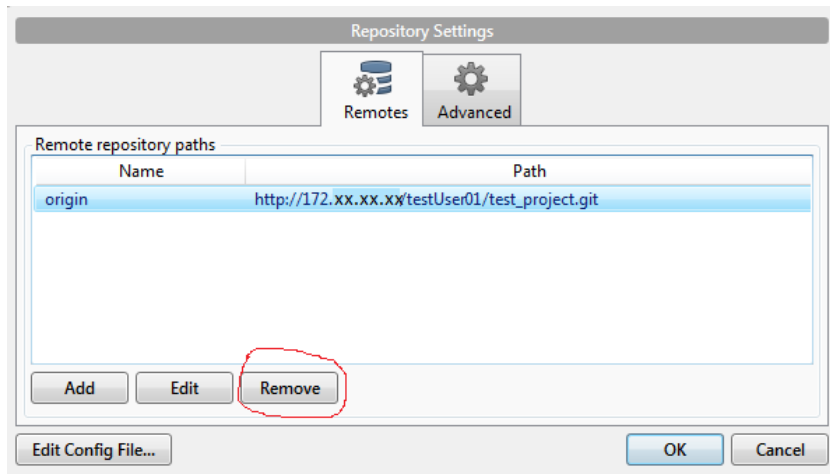
Folder: [Root]

Clone Cancel

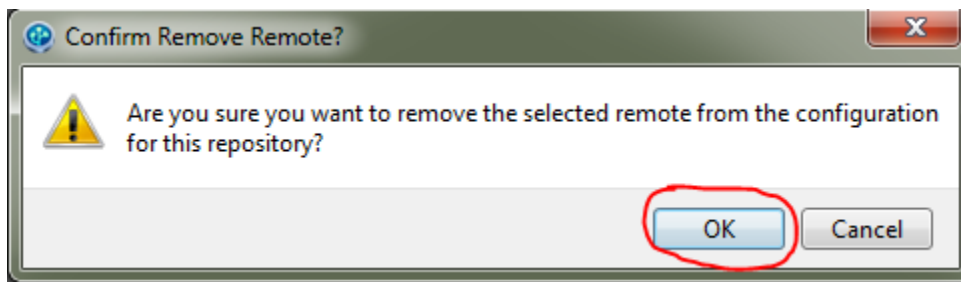
4. Click **Repository / Repository Settings...**



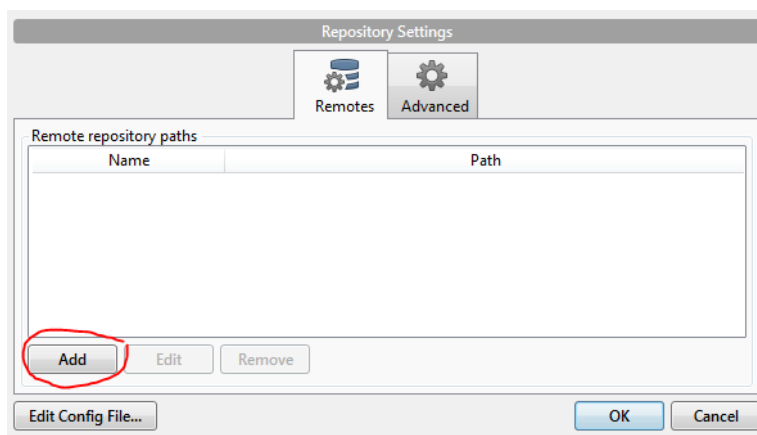
5. Repository settings view as below :



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



7. Click **Add**



8. Fill it accordingly

Remote details

Required information

Remote name: **prjTest** ☐ Default remote

URL / Path: **http://172.xx.xx.xx/testUser01/test_project.git**

Optional extended integration

Host Type: Unknown

Host Root URL:

Username:

Extended integration is used to enable deeper integration with hosting providers such as Bitbucket, including locating existing clones when following links from sites and creating pull requests.

OK Cancel

9. Repository settings should look something similar :

Repository Settings

Remotes Advanced

Remote repository paths

Name	Path
prjTest	http://172.xx.xx.xx/testUser01/test_project.git

Add Edit Remove

Edit Config File... **OK** Cancel

10. Click **Ok**

12.Push

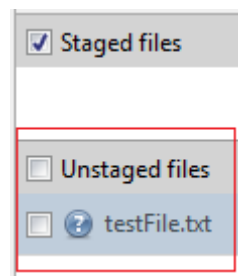
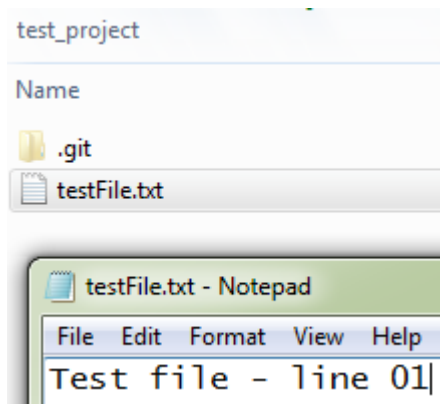
Transfer commits from local repository to a remote repo.

Command

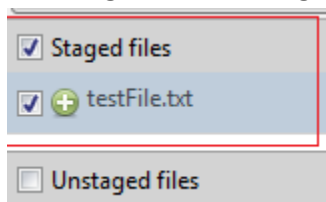
`git push`

Steps

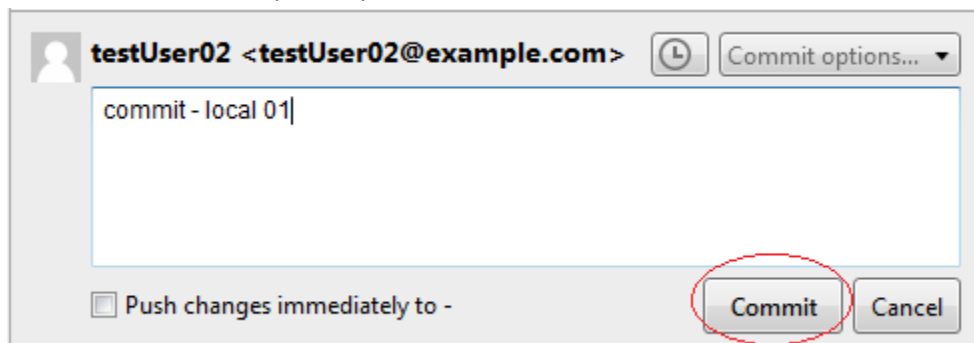
1. Create a file (e.g. **testFile.txt**) in your project (e.g. **test_project**)



2. Add the file from 'Unstaged files' to 'Staged files' using **Add**



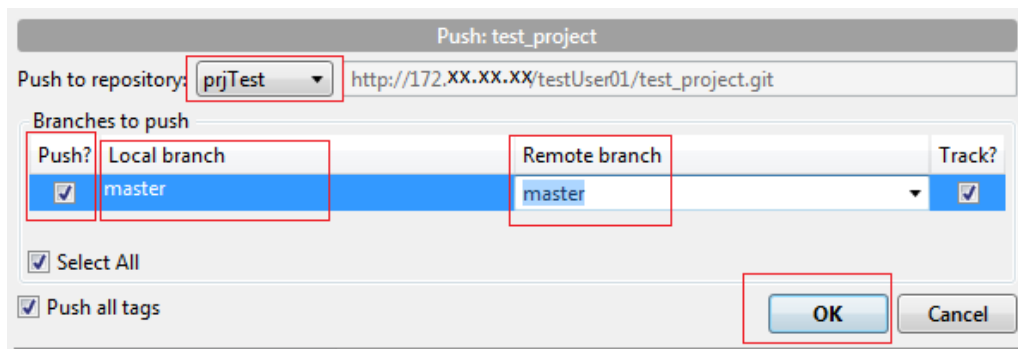
3. Commit to the local repository



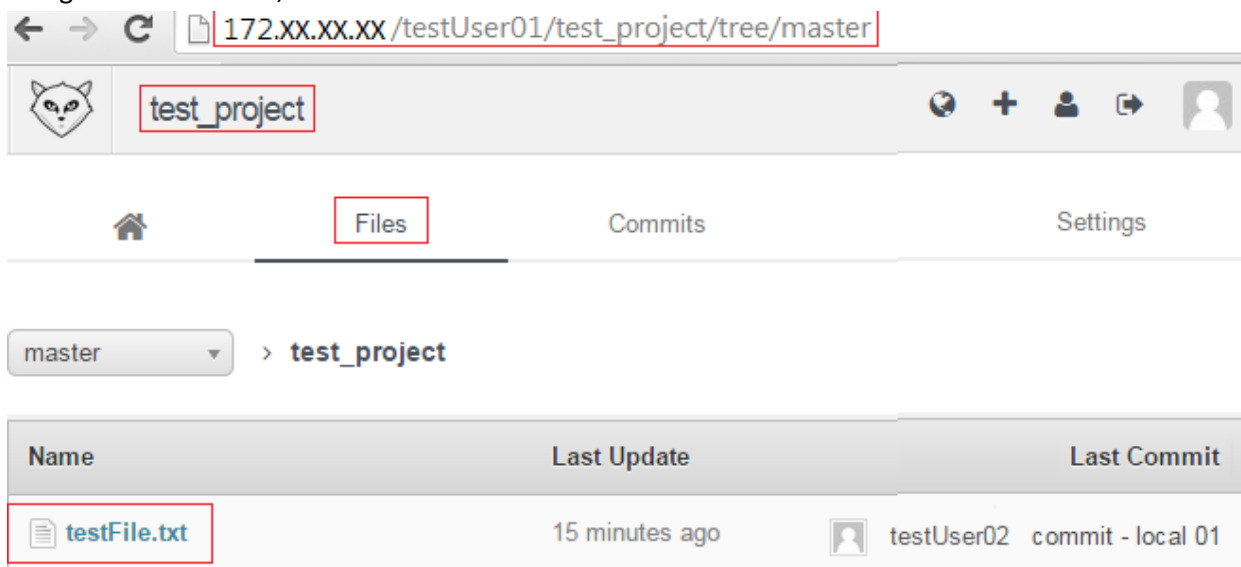
4. Click 'Push'



5. In push dialogue box make sure you select (type) **master** as your remote and local branch



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



- 7.

13.Fetch

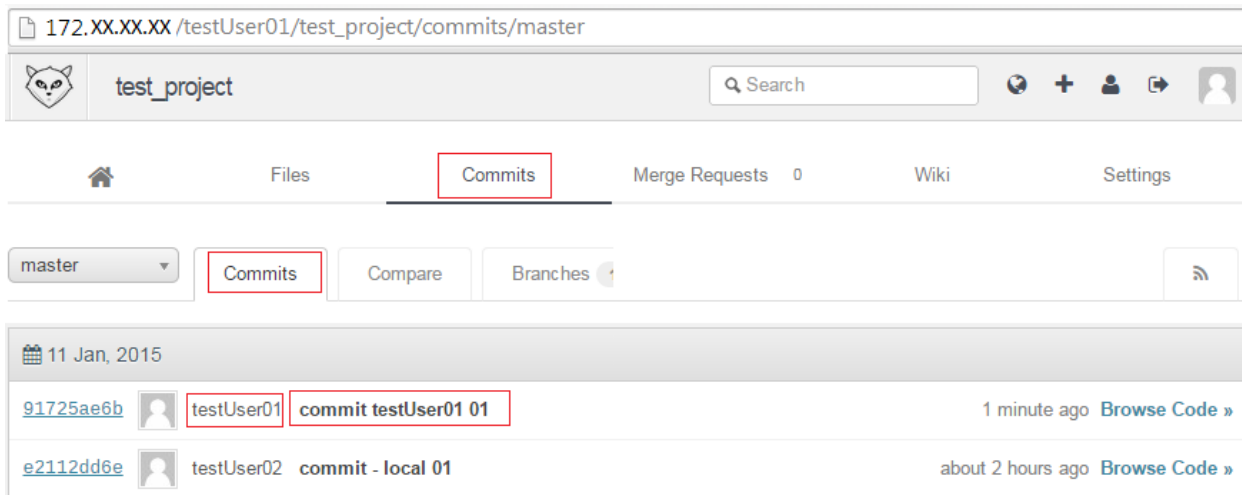
This command imports commits from a remote repository into the local repository.

Command

git fetch

Steps

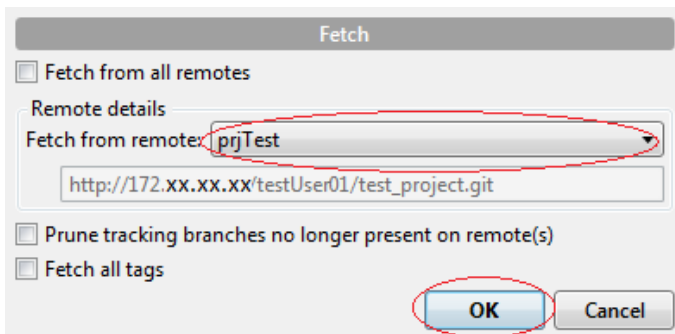
1. Let some **other user** modify the file , commit, and then push the changes to the remote repository (irrespective of any tool)



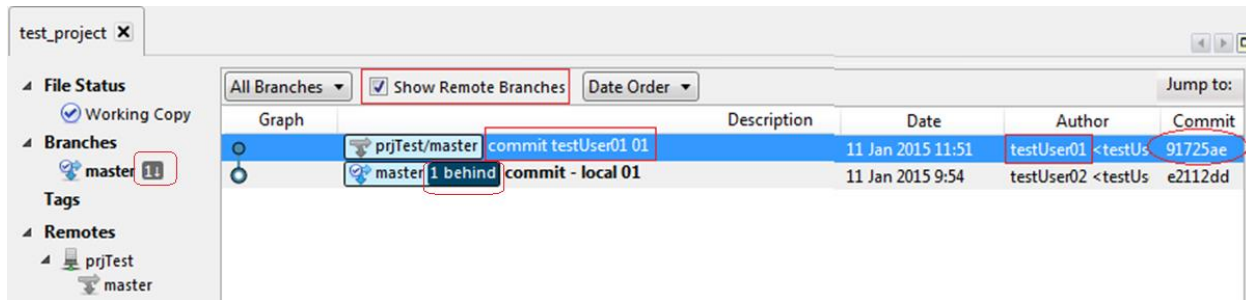
2. Click on **Fetch**



3. Fetch dialogue box will appear



4. 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

Merging upstream changes into local repository
(It's a combined command of **git fetch** and **git merge**)

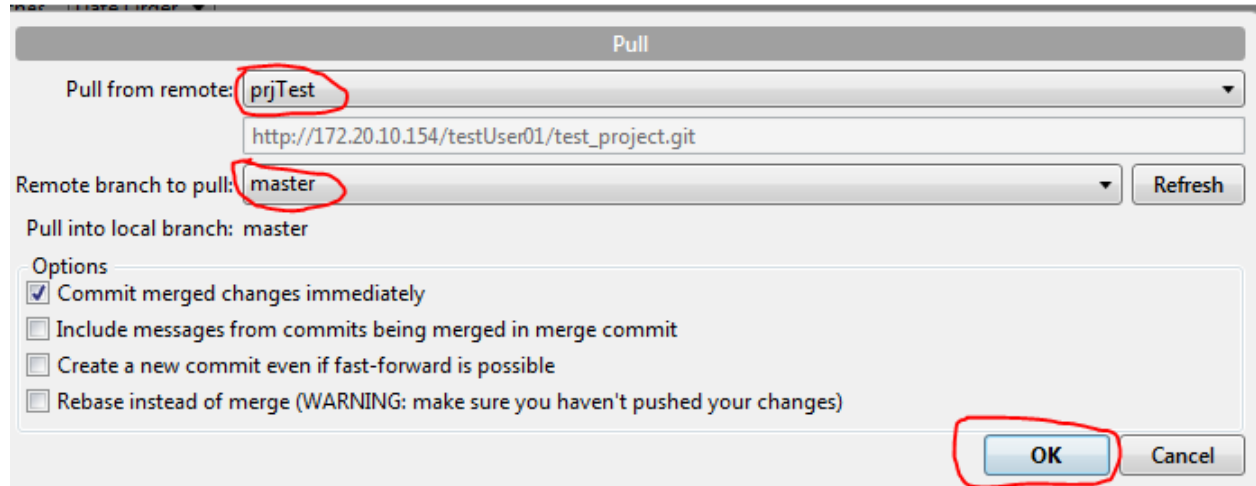
Steps

1. Click 'Pull'

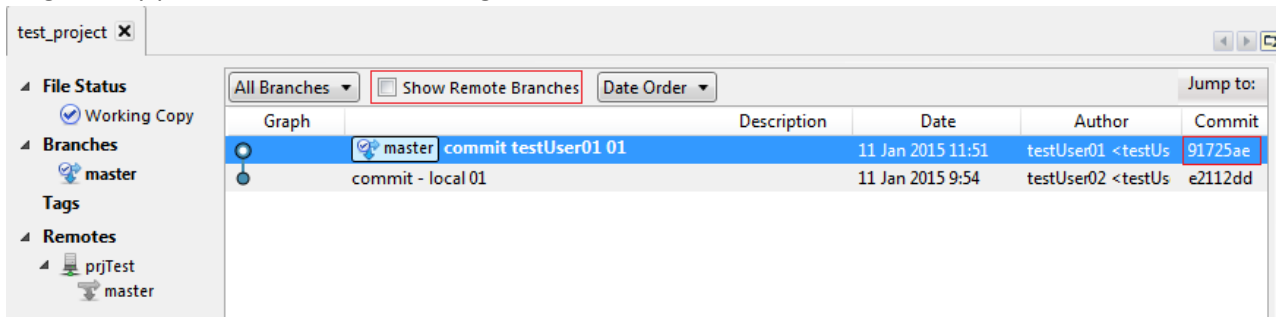


(Check the numeric digit above . It shows changes in remote)

2. Click ok in the dialogue box



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



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

Command

git pull

Using Branch

Branch

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

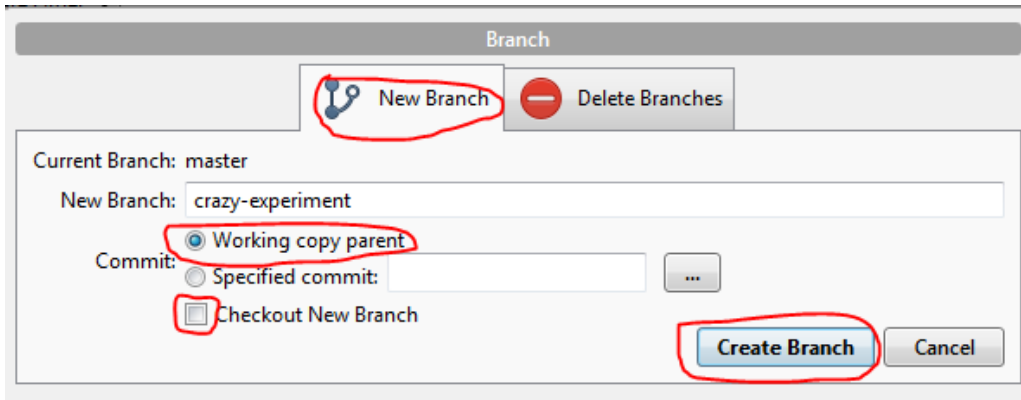
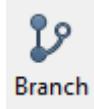
Command

`git branch`

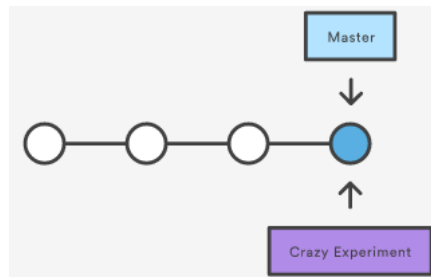
Create a Branch

Steps

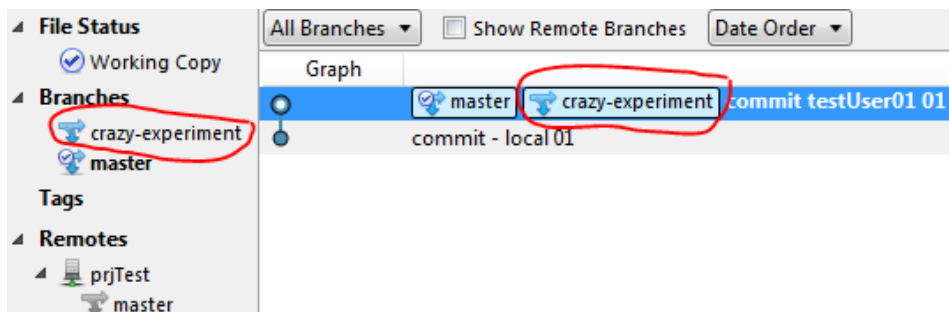
1. Click **Branch**
2. Write a branch name (e.g. crazy-experiment). Uncheck the **Checkout New Branch** checkbox



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



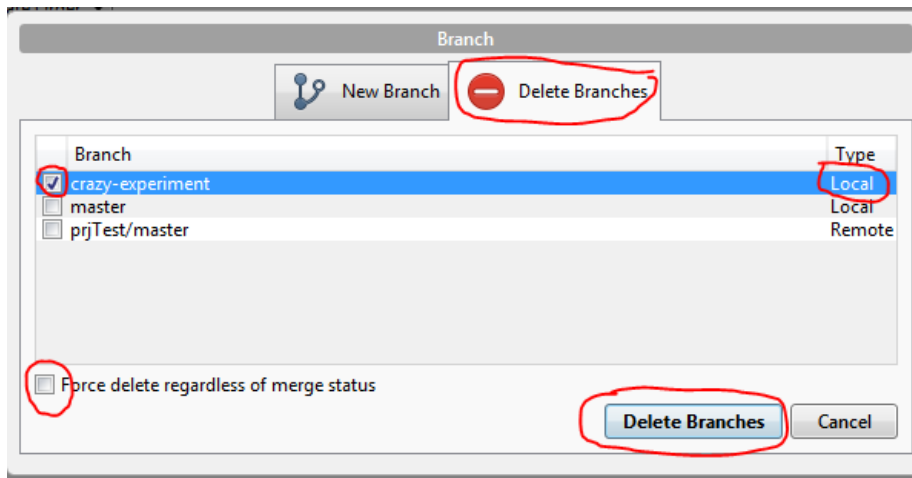
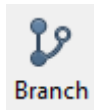
4. To working with this branch one should use the **checkout**



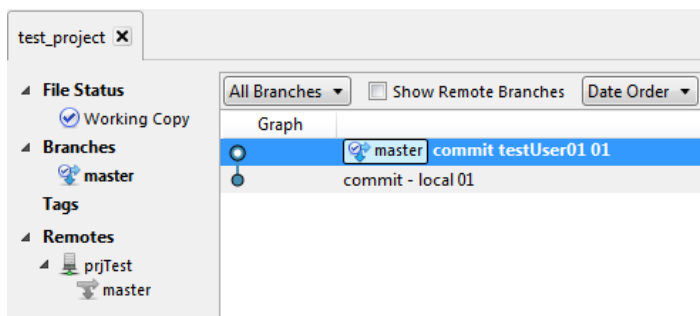
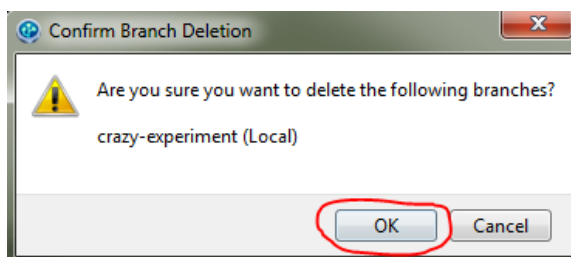
Delete a Branch

Steps

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



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



Checkout

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

Command

git checkout


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

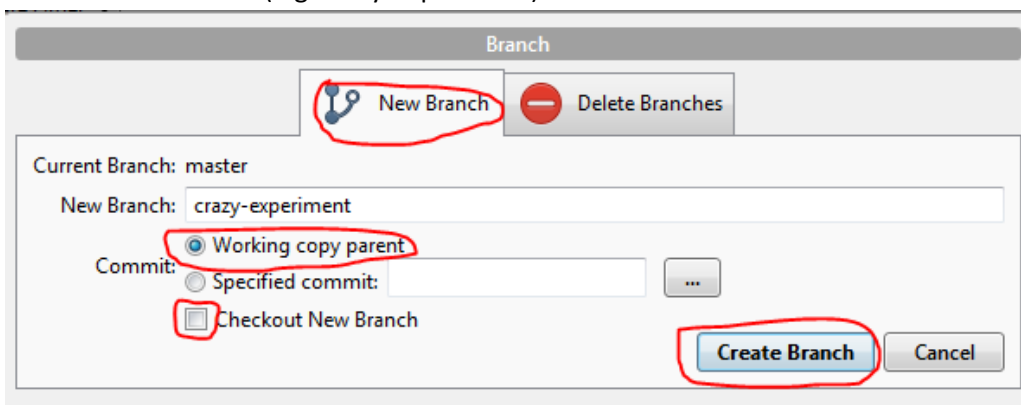
- Using of Checkout **after** creating a Branch
- Creation of Branch at the time of **Checkout**

Let's discuss both the steps

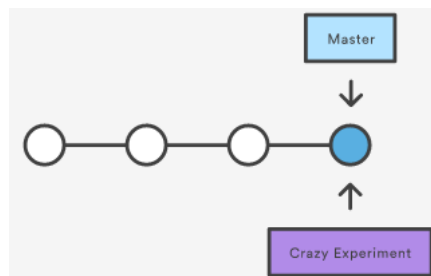
A. Using of Checkout **after** creating a Branch

Steps

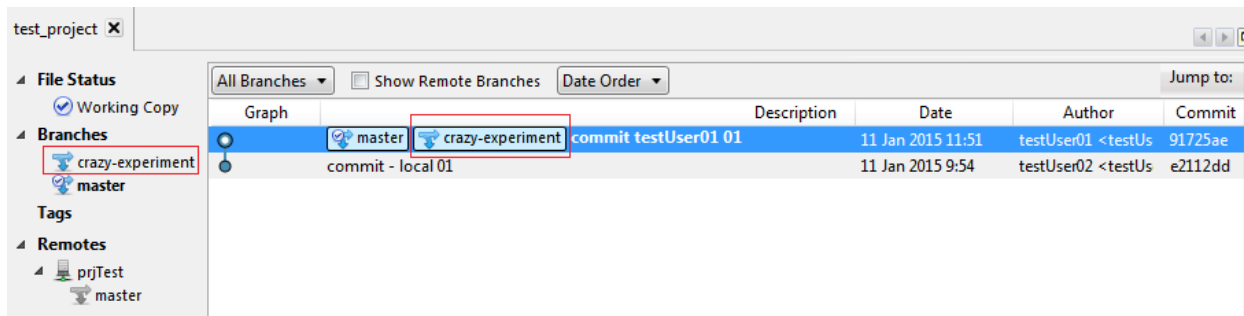
- Click **Branch** 
- Write a branch name (e.g. crazy-experiment). Uncheck the **Checkout New Branch** checkbox



- A branch with pointer to working copy will create.



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

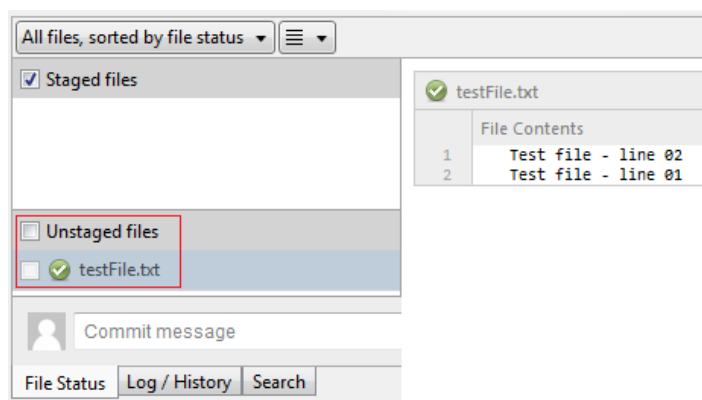


5. Different views

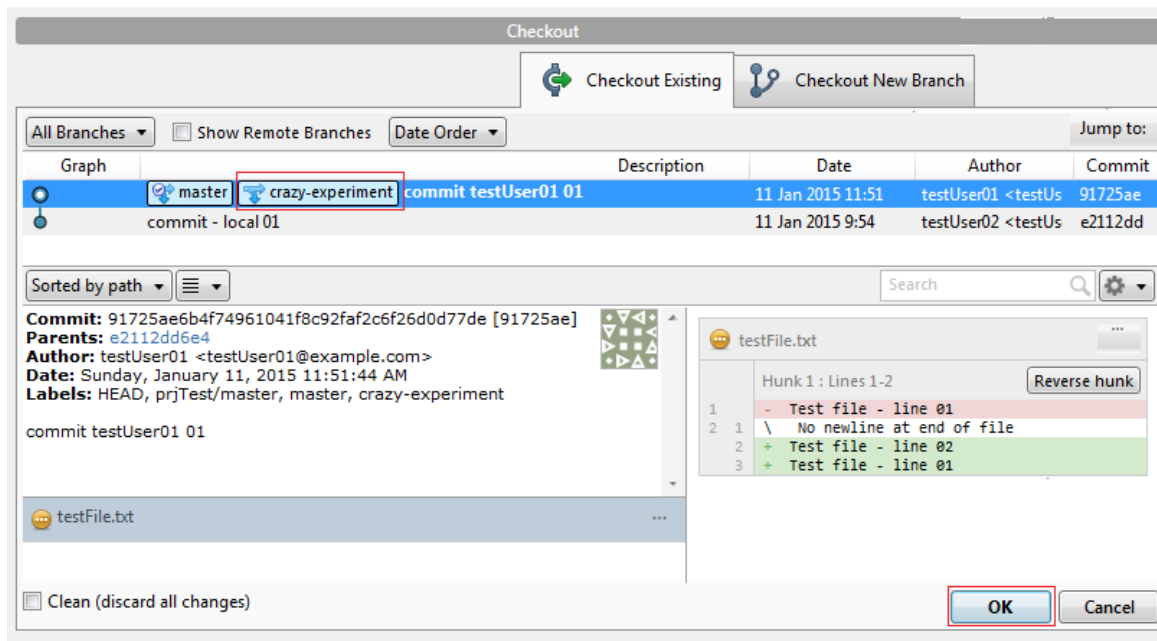
a. **Log / History**



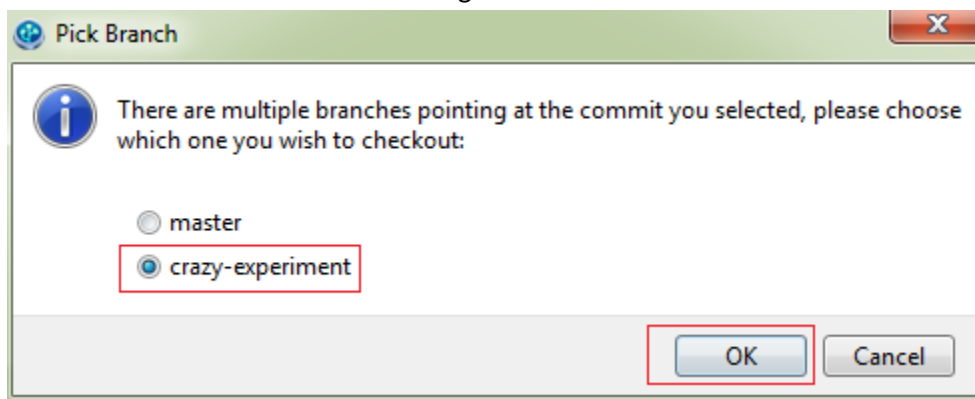
b. **File Status**





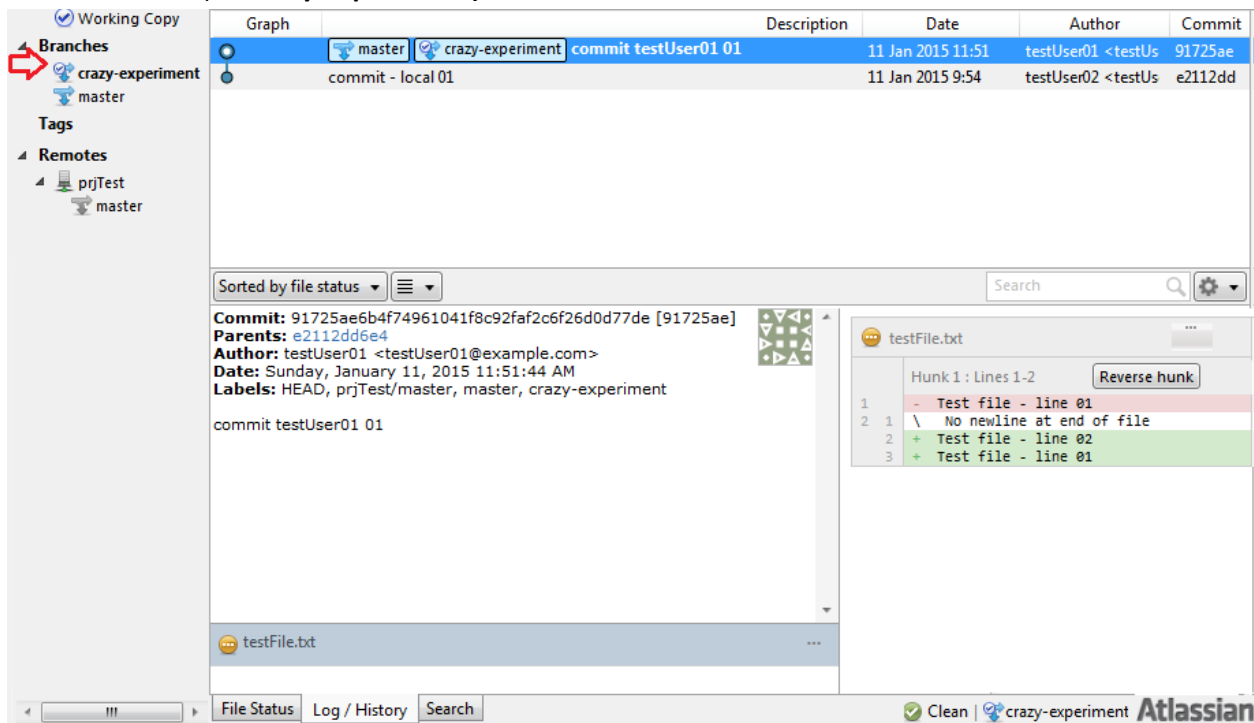
6. Click on **Checkout**






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



8. After **checkout** the little arrow (with check mark)  (i.e. ) will be shifted to the new branch (i.e. **crazy-experiment**)



The screenshot displays the Git GUI interface. On the left sidebar, under 'Working Copy', the 'crazy-experiment' branch is selected with a red arrow icon. The main area shows a commit graph with two commits: 'commit testUser01 01' (blue) and 'commit - local 01' (grey). The commit details for '91725ae6b4f74961041f8c92faf2c6f26d0d77de [91725ae]' are shown, including the author 'testUser01 <testUser01@example.com>' and the date 'Sunday, January 11, 2015 11:51:44 AM'. The file status section shows 'testFile.txt' with a diff view. The diff shows a hunk with three lines: a red line '- Test file - line 01', a green line '+ Test file - line 02', and a green line '+ Test file - line 01'. The bottom status bar indicates 'Clean' and 'crazy-experiment'.

Graph	Description	Date	Author	Commit
 master  crazy-experiment	commit testUser01 01	11 Jan 2015 11:51	testUser01 <testUs	91725ae
 commit - local 01		11 Jan 2015 9:54	testUser02 <testUs	e2112dd

Sorted by file status

Commit: 91725ae6b4f74961041f8c92faf2c6f26d0d77de [91725ae]
Parents: e2112dd6e4
Author: testUser01 <testUser01@example.com>
Date: Sunday, January 11, 2015 11:51:44 AM
Labels: HEAD, prjTest/master, master, crazy-experiment
commit testUser01 01

testFile.txt

Hunk 1 : Lines 1-2
Reverse hunk

```
1 - Test file - line 01
2 1 \ No newline at end of file
2 + Test file - line 02
3 + Test file - line 01
```

File Status Log / History Search

Clean | crazy-experiment

B. Creation of Branch at the time of Checkout

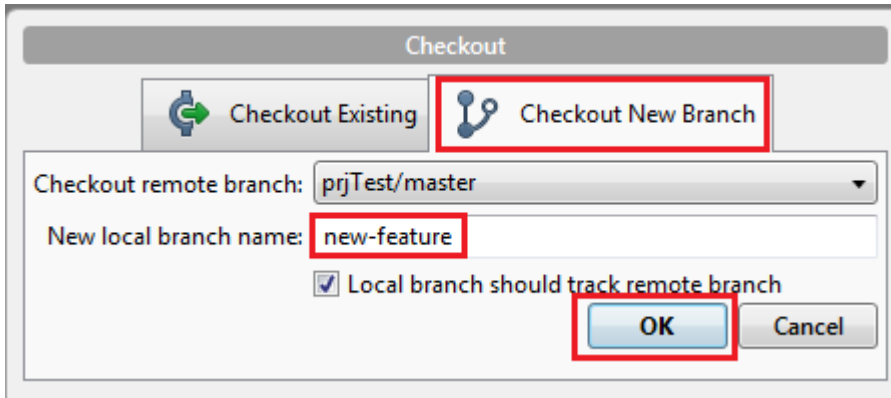
Steps

1. Click **Checkout**

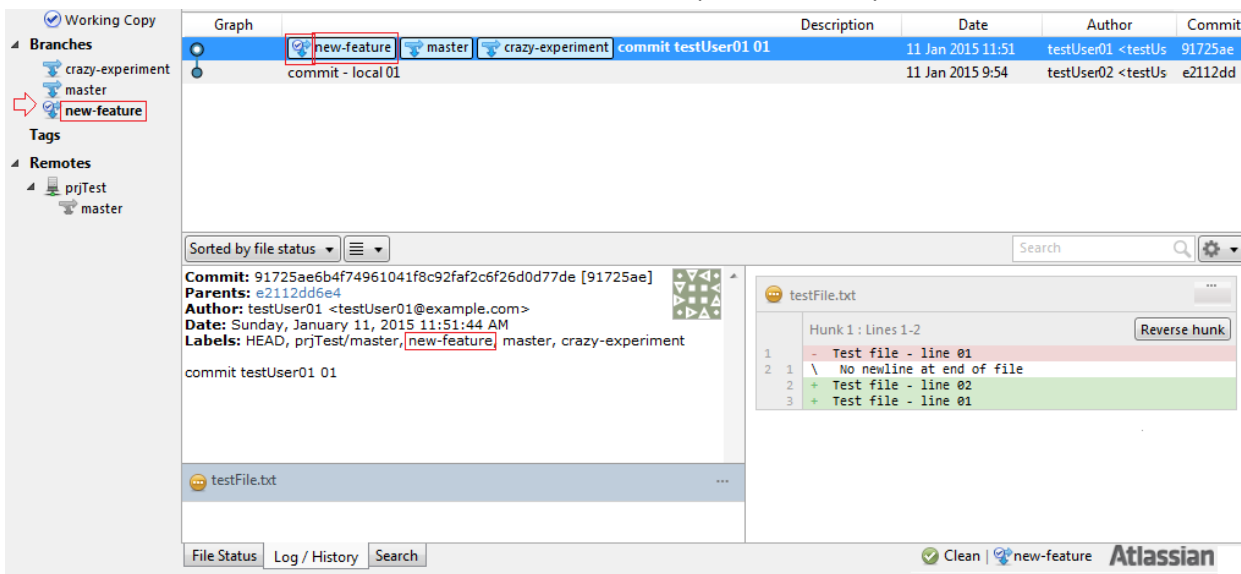


Checkout

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



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



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.

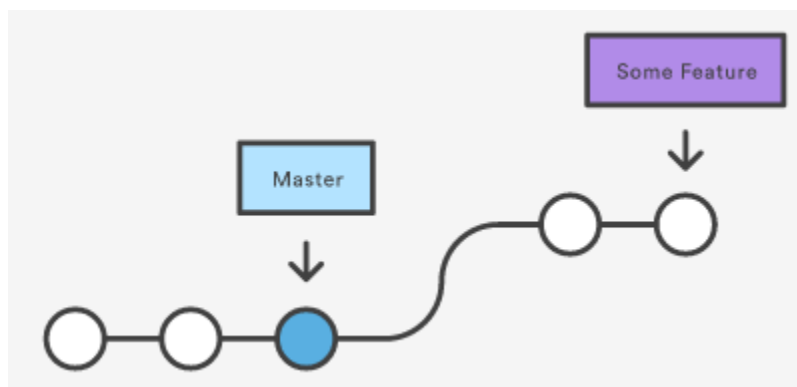
Command

git merge

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

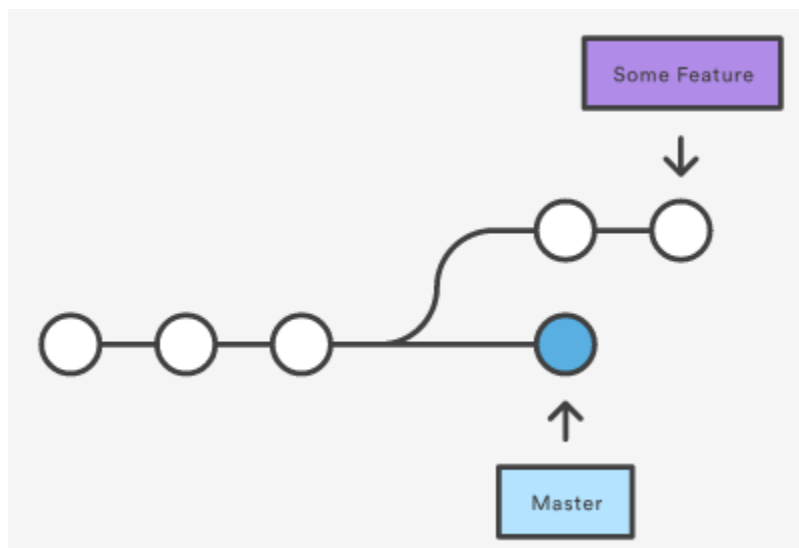


3-Way-Merge

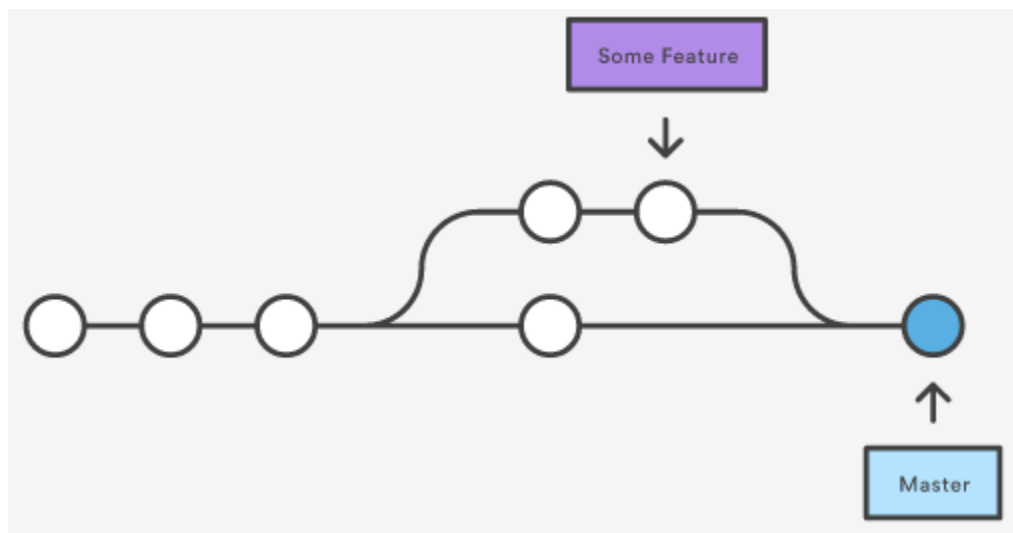
a fast-forward merge is not possible if the branches have diverged.

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

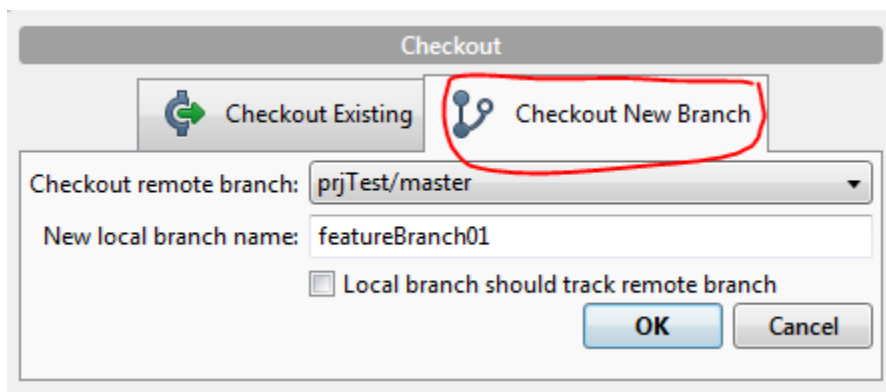
Steps

Start a new feature



1. Click **Checkout**

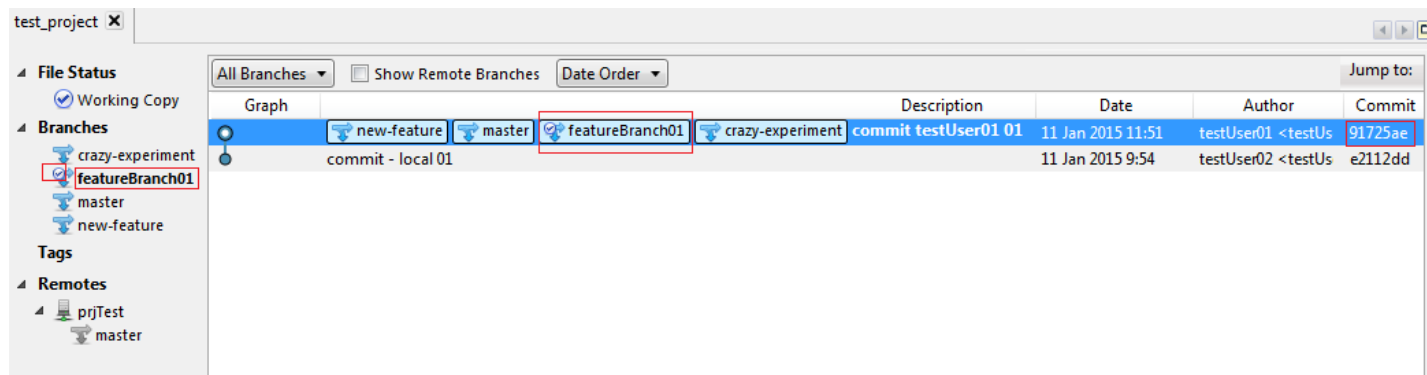
Click **Checkout New Branch** tab



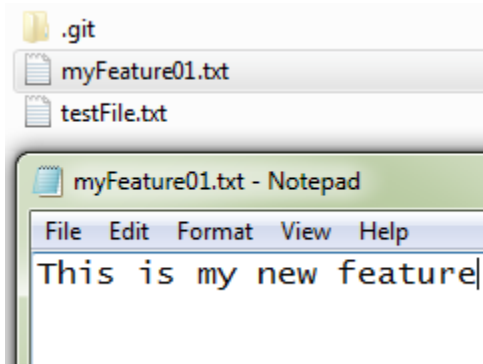
Give a suitable name of the branch (e.g. **featureBranch01**)

Click **OK**

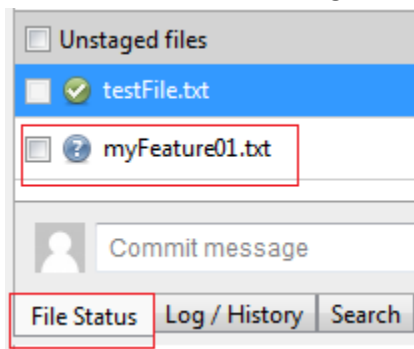
2. Newly created branch looks like below :



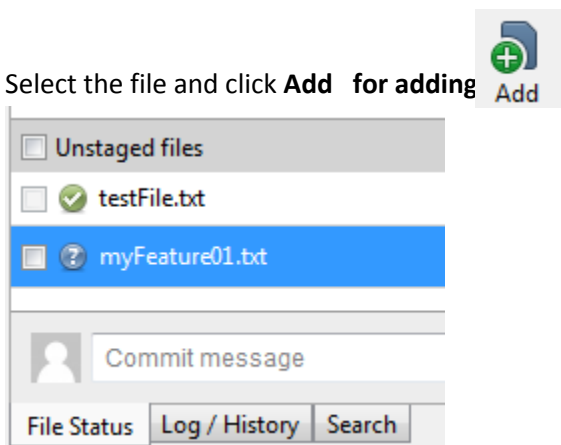
3. Create a file



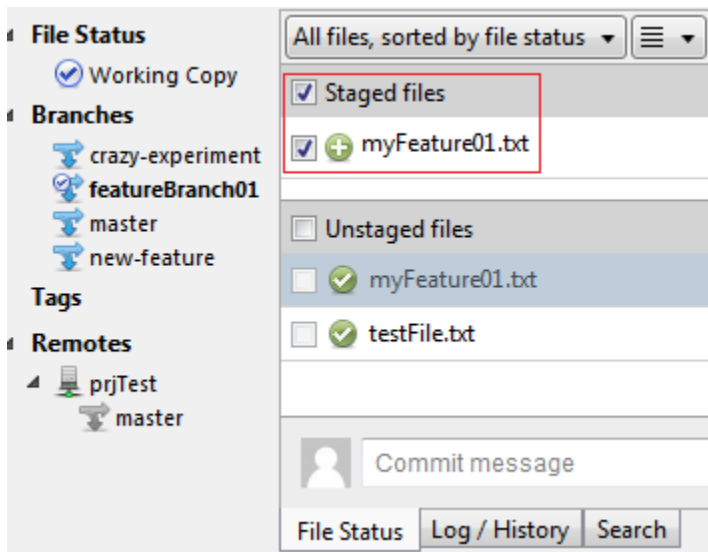
4. Newly created file will be in unstaged area



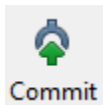
5. Select the file and click **Add** for adding



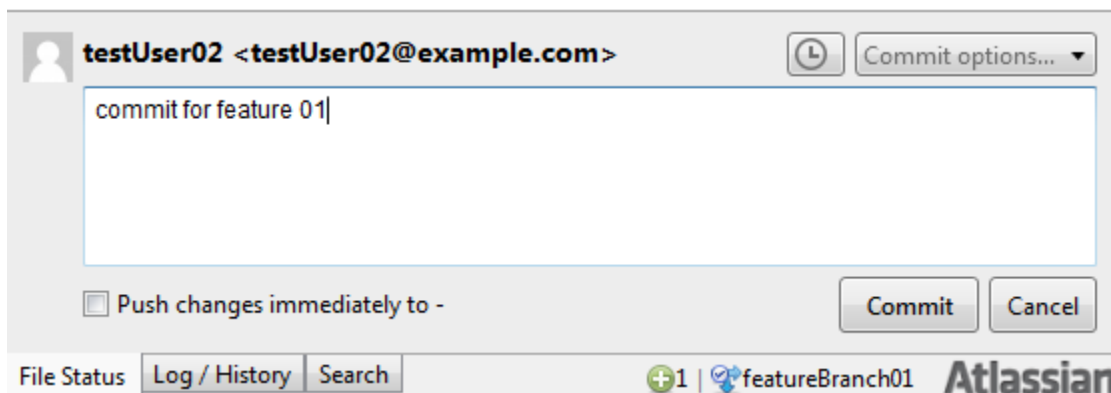
6. File will be added to the staging area



7. Click on **Commit**

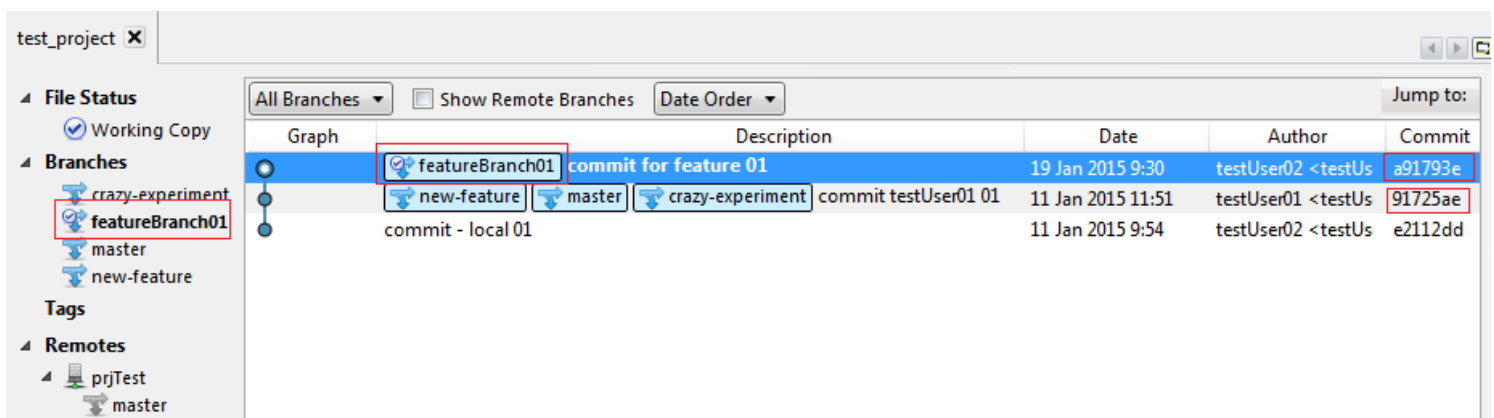


8. Write comment into the comment box

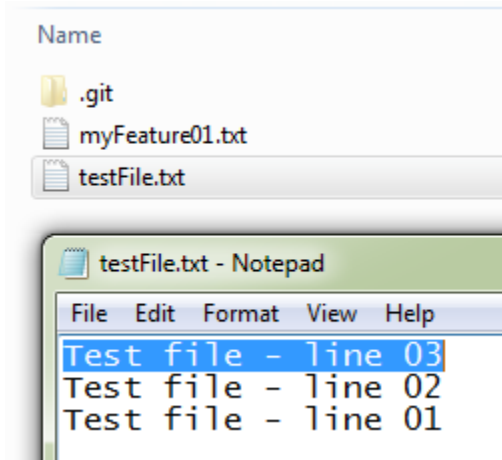


Click **Commit**

9. Log/History looks something given below

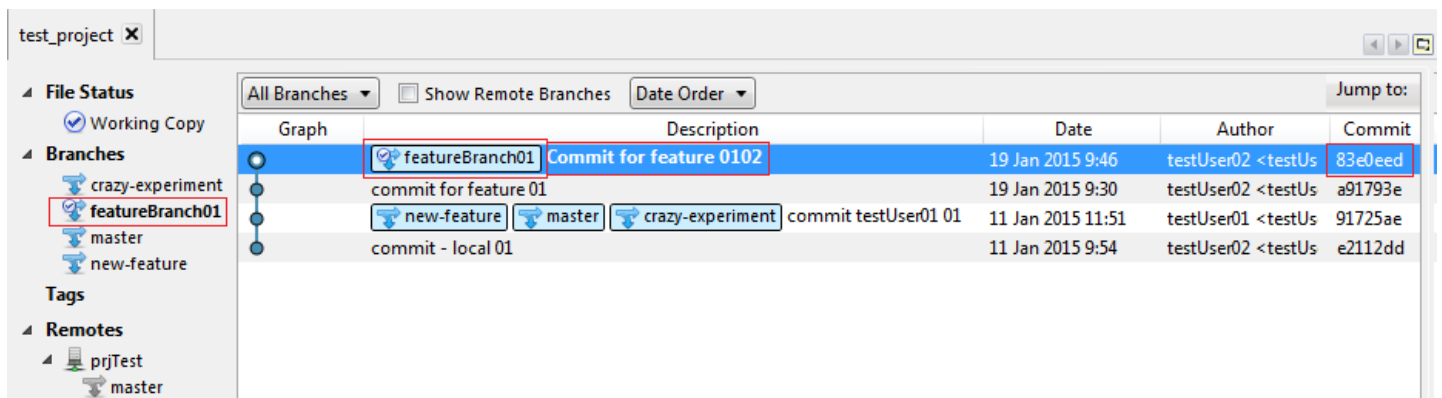


10. Modify some other file



11. Add the file and commit it as described before

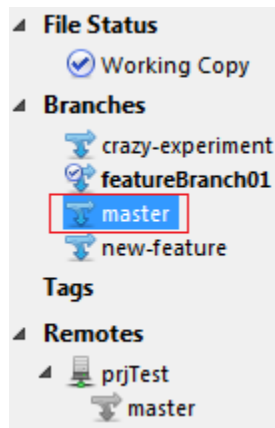
12. Log / History looks something like :



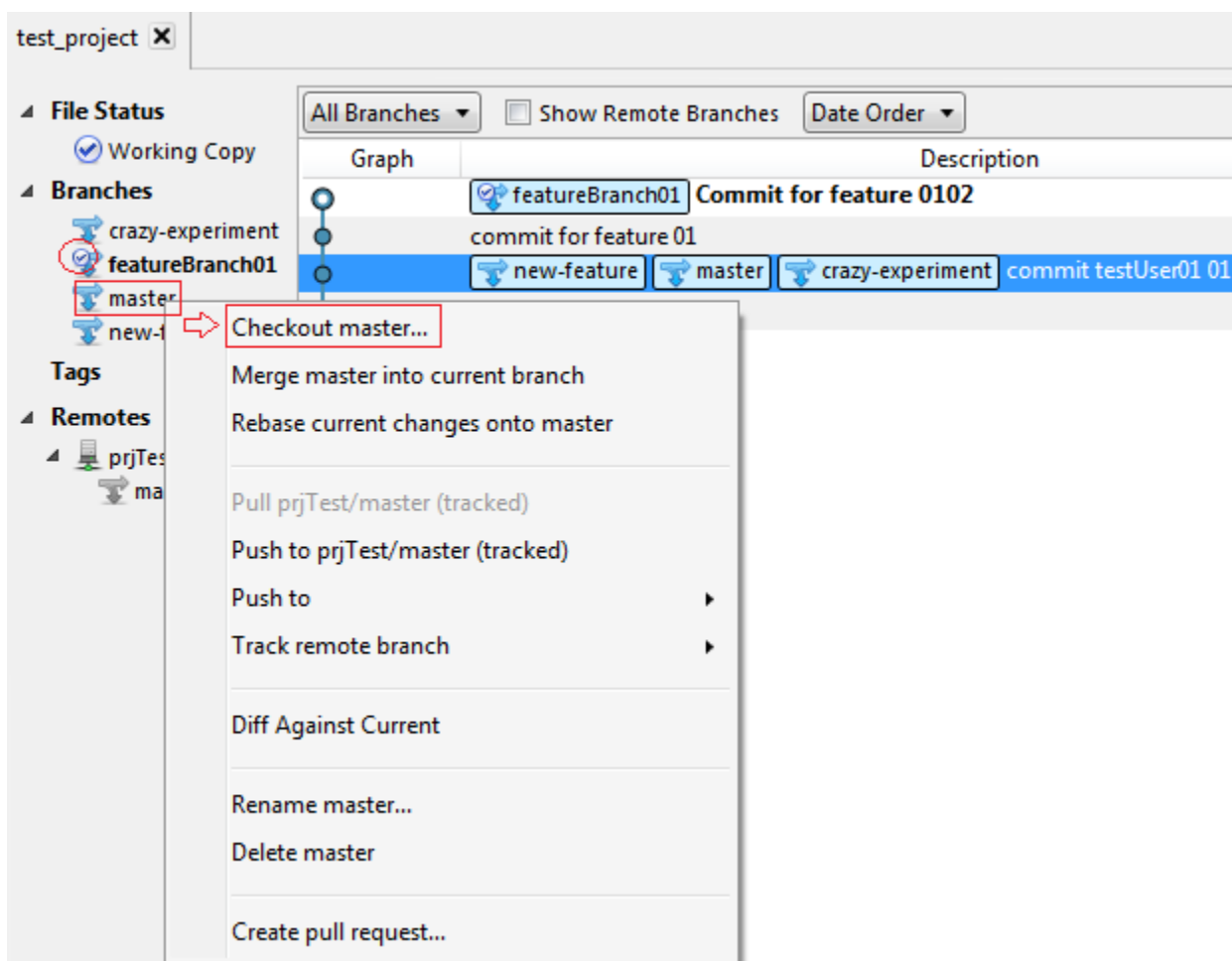
Merge in the new-feature branch

13. Checkout master

Select **Master** from Branches

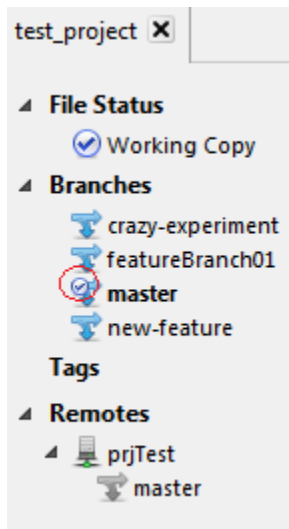


Right click after selecting master

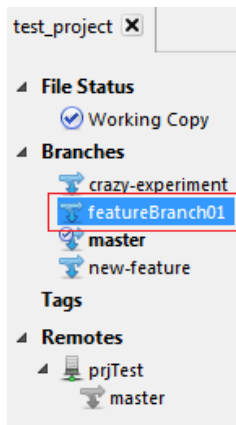


Click **Checkout master**

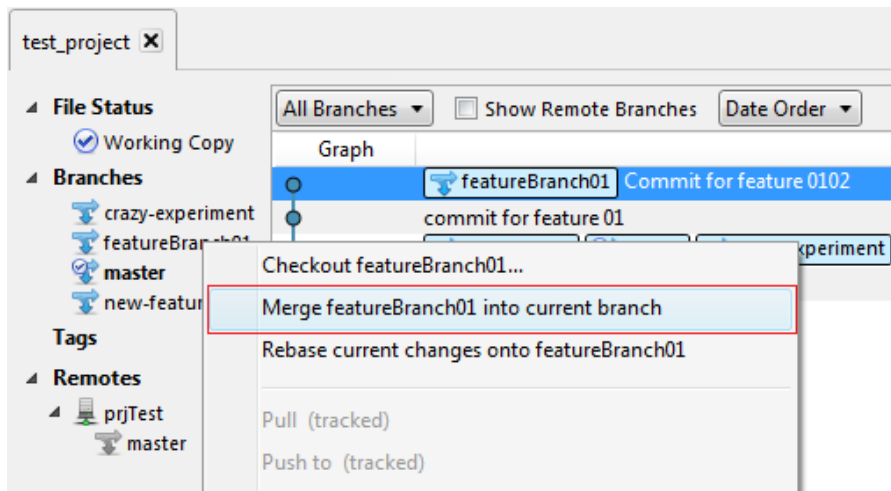
Branch view are shown below (mind the little arrow – shifted to master)



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

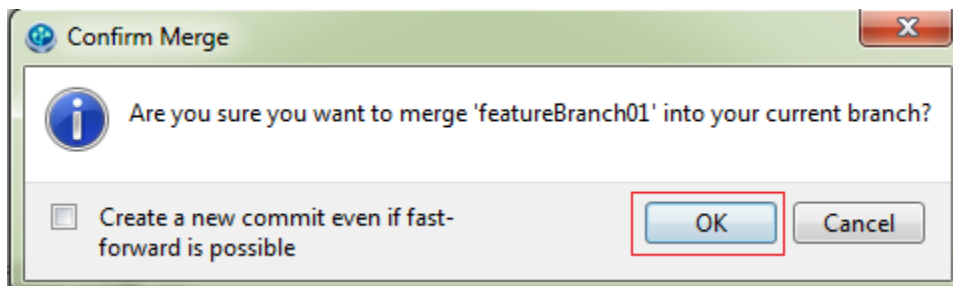


15. Right Click on selected branch

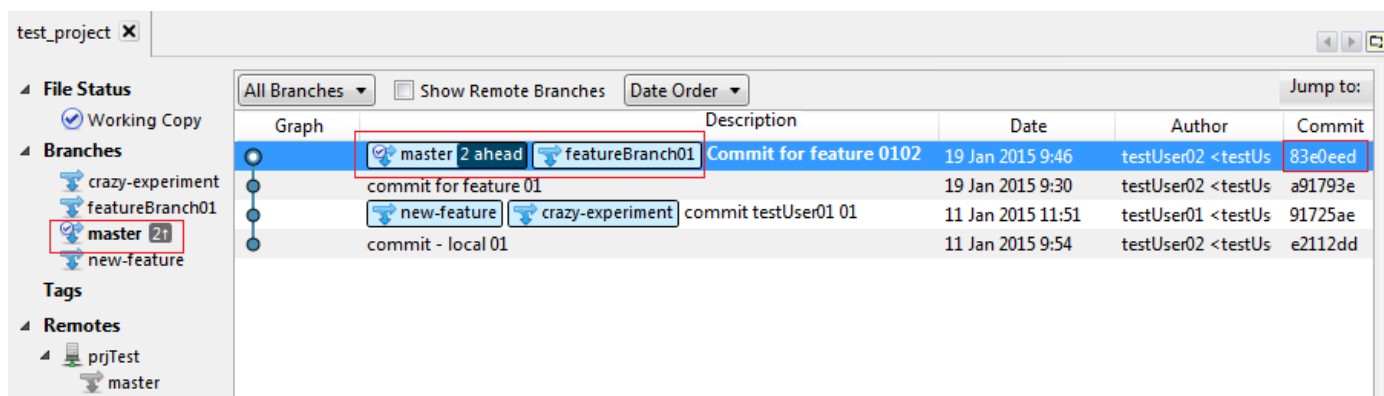


16. Click on the second option from the pull down menu i.e. **Merge <branch name> into current branch**

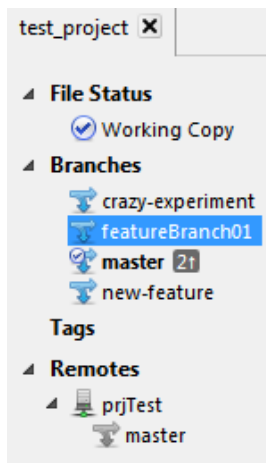
17. Click **OK** to the **Confirm Merge** dialogue box



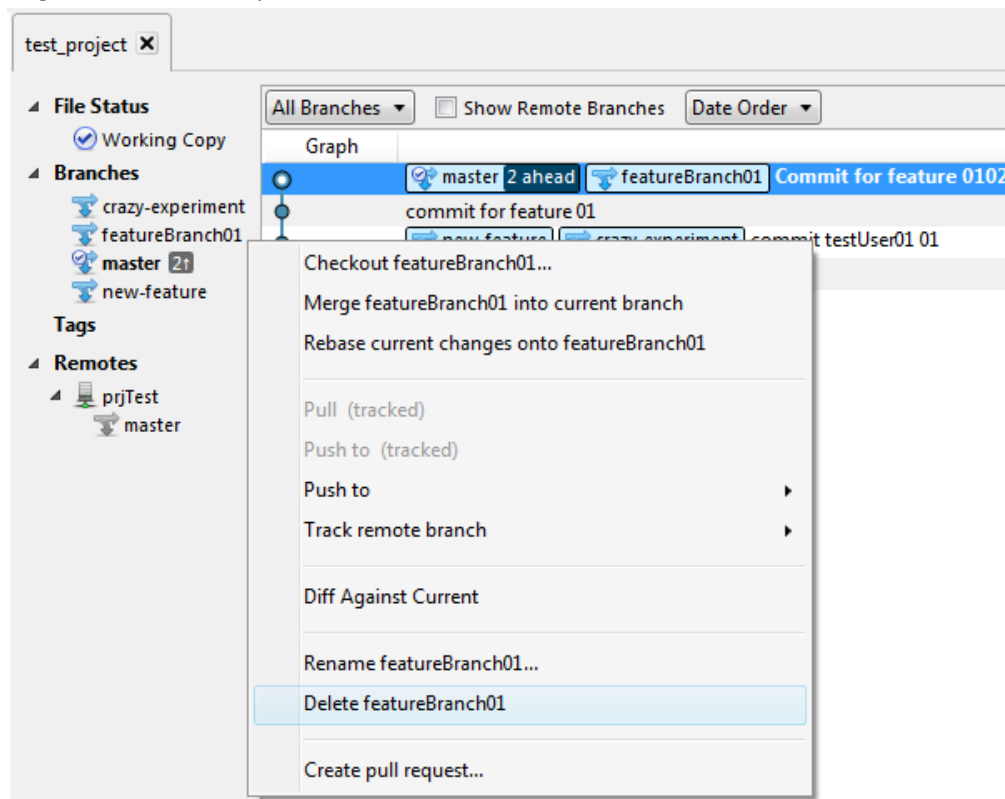
18. Log / History view will be something as shown below :



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

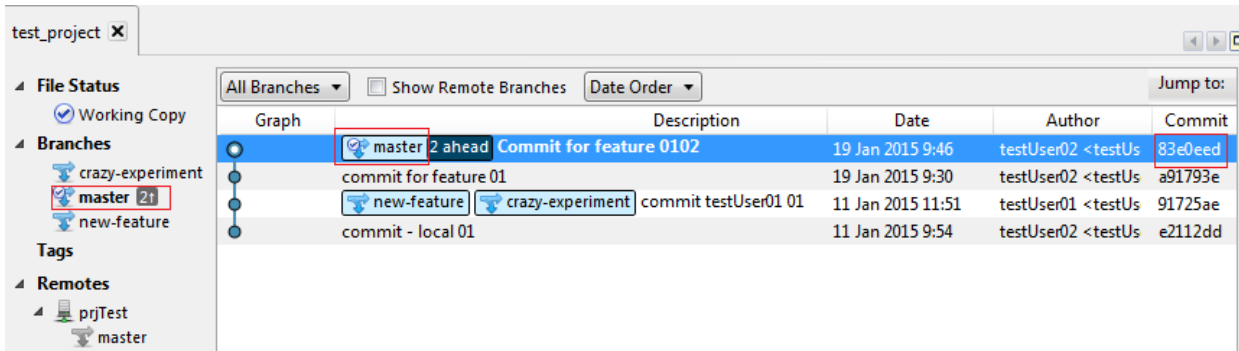


20. Right click and from pull down menu click on **Delete <branch name i.e. featureBranch01>**



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

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



The screenshot shows the Git commit history view in SourceTree. The left sidebar displays the 'File Status' and 'Branches' sections. The 'Branches' section shows 'crazy-experiment', 'master' (with a red box around it and '2 ahead' next to it), and 'new-feature'. The 'Tags' section is empty. The 'Remotes' section shows 'prjTest' and 'master'. The main area displays a table of commits, sorted by date. The top commit is highlighted in blue and has its hash '83e0eed' boxed in red. The table has columns for 'Graph', 'Description', 'Date', 'Author', and 'Commit'.

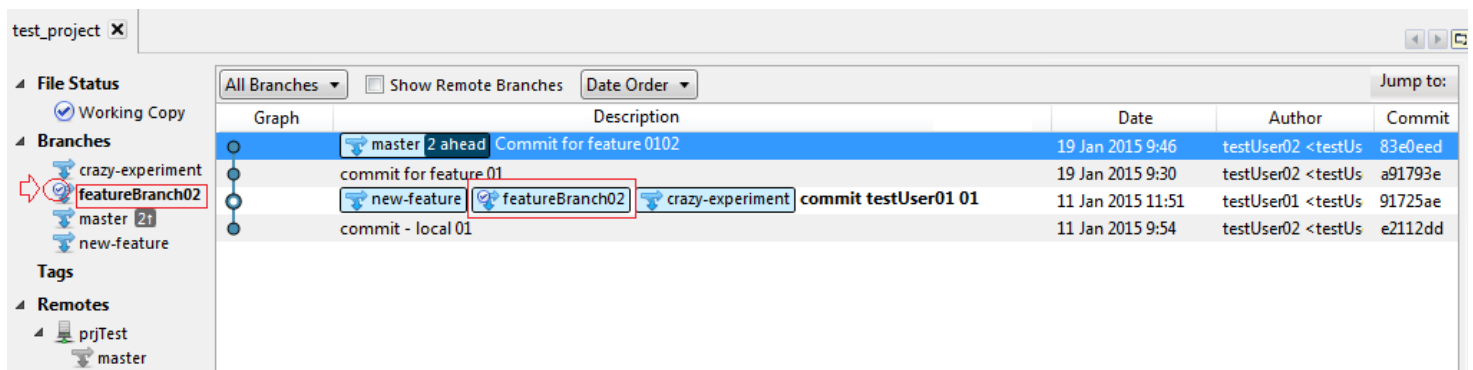
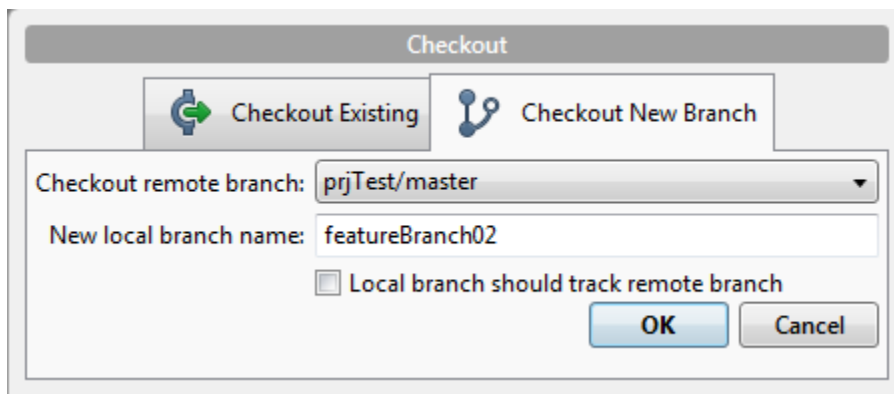
Graph	Description	Date	Author	Commit
master 2 ahead	Commit for feature 0102	19 Jan 2015 9:46	testUser02 <testUs	83e0eed
	commit for feature 01	19 Jan 2015 9:30	testUser02 <testUs	a91793e
new-feature crazy-experiment	commit testUser01 01	11 Jan 2015 11:51	testUser01 <testUs	91725ae
	commit - local 01	11 Jan 2015 9:54	testUser02 <testUs	e2112dd

3-Way Merge

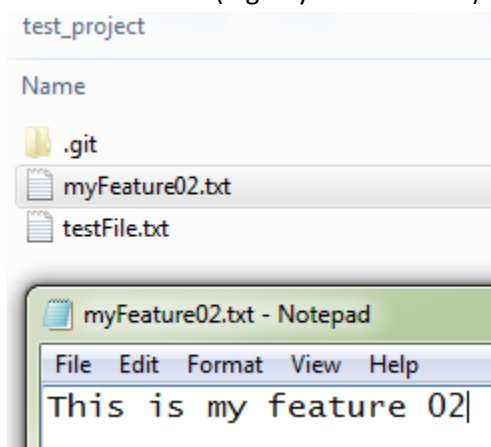
Steps

Start new feature

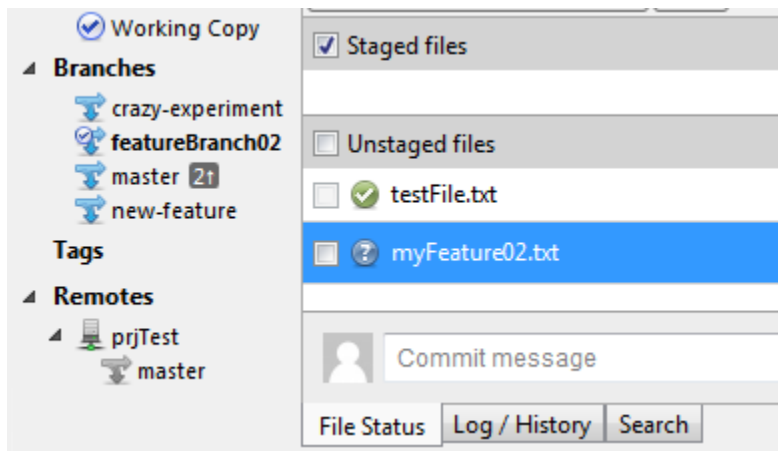
1. Create a new branch as discussed earlier (e.g. featureBranch02) using checkout



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



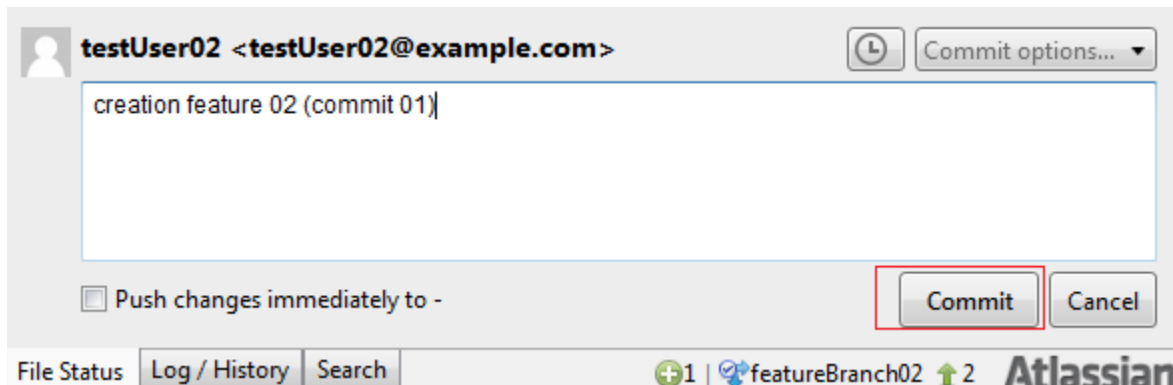
3. File will be Unstaged area



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



5. Commit the file using **Commit**



6. Log / History view

The screenshot shows the Git GUI interface. On the left, the 'Working Copy' section shows the 'featureBranch02' branch selected. The main area displays a list of commits in a table format:

Graph	Description	Date	Author	Commit
featureBranch02	creation feature 02 (commit 01)	20 Jan 2015 9:13	testUser02 <testUs	81be820
master 2 ahead	Commit for feature 0102	19 Jan 2015 9:46	testUser02 <testUs	83e0eed
	commit for feature 01	19 Jan 2015 9:30	testUser02 <testUs	a91793e
new-feature	commit testUser01 01	11 Jan 2015 11:51	testUser01 <testUs	91725ae
	commit - local 01	11 Jan 2015 9:54	testUser02 <testUs	e2112dd

Below the table, the selected commit (81be820) is detailed:

- Commit:** 81be8205d699383bd17e263338ae53c07084a3b6 [81be820]
- Parents:** 91725ae6b4
- Author:** testUser02 <testUser02@example.com>
- Date:** Tuesday, January 20, 2015 9:13:25 AM
- Labels:** HEAD, featureBranch02
- Description:** creation feature 02 (commit 01)

On the right, the file 'myFeature02.txt' is shown with its contents:

```

+ This is my feature 02
\ No newline at end of file

```

7. Modify the newly created text file (e.g. myFeature02.txt)

The screenshot shows a file explorer window for 'test_project'. The files listed are '.git', 'myFeature02.txt', and 'testFile.txt'. Below the file explorer, a Notepad window titled 'myFeature02.txt - Notepad' is open, showing the following text:

```

This is my feature 02
I am developing new feature

```

8. Add the file using Add



test_project X

File Status

Working Copy

Branches

- crazy-experiment
- featureBranch02**
- master 21
- new-feature

Tags

Remotes

- prjTest
- master

All files, sorted by file status

☒ Staged files

- ☒ myFeature02.txt

☐ Unstaged files

- ☒ myFeature02.txt
- ☒ testFile.txt

testUser02 <testUser02@example.com>

Commit options...

Developing feature 02

☐ Push changes immediately to -

Commit **Cancel**

File Status Log / History Search

1 | featureBranch02 2 | **Atlassian**

myFeature02.txt

Hunk 1 : Lines 1-2

Unstage hunk

```

1 - This is my feature 02
2 \ No newline at end of file
3 + This is my feature 02
4 + I am developing new feature
5 \ No newline at end of file

```

9. Commit the changes

After commit the Log/History view

File Status

Working Copy

Branches

- crazy-experiment
- featureBranch02**
- master 21
- new-feature

Tags

Remotes

- prjTest
- master

All Branches

☐ Show Remote Branches

Date Order

Jump to:

Graph	Description	Date	Author	Commit
featureBranch02	Developing feature 02	20 Jan 2015 10:50	testUser02 <testUs	7aec64d
	creation feature 02 (commit 01)	20 Jan 2015 9:13	testUser02 <testUs	81be820
master 2 ahead	Commit for feature 0102	19 Jan 2015 9:46	testUser02 <testUs	83e0eed
	commit for feature 01	19 Jan 2015 9:30	testUser02 <testUs	a91793e
new-feature crazy-experiment	commit testUser01 01	11 Jan 2015 11:51	testUser01 <testUs	91725ae
	commit - local 01	11 Jan 2015 9:54	testUser02 <testUs	e2112dd

Sorted by file status

Commit: 81be8205d699383bd17e263338ae53c07084a3b6 [81be820]

Parents: 91725ae6b4

Author: testUser02 <testUser02@example.com>

Date: Tuesday, January 20, 2015 9:13:25 AM

creation feature 02 (commit 01)

myFeature02.txt

File Status Log / History Search

Search

myFeature02.txt

File Contents

Reverse hunk

```

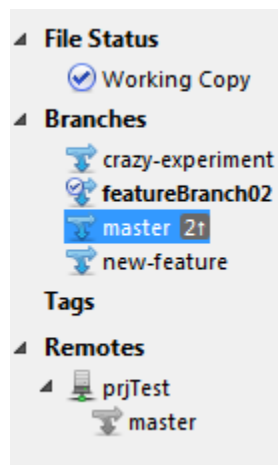
0 + This is my feature 02
0 1 \ No newline at end of file

```

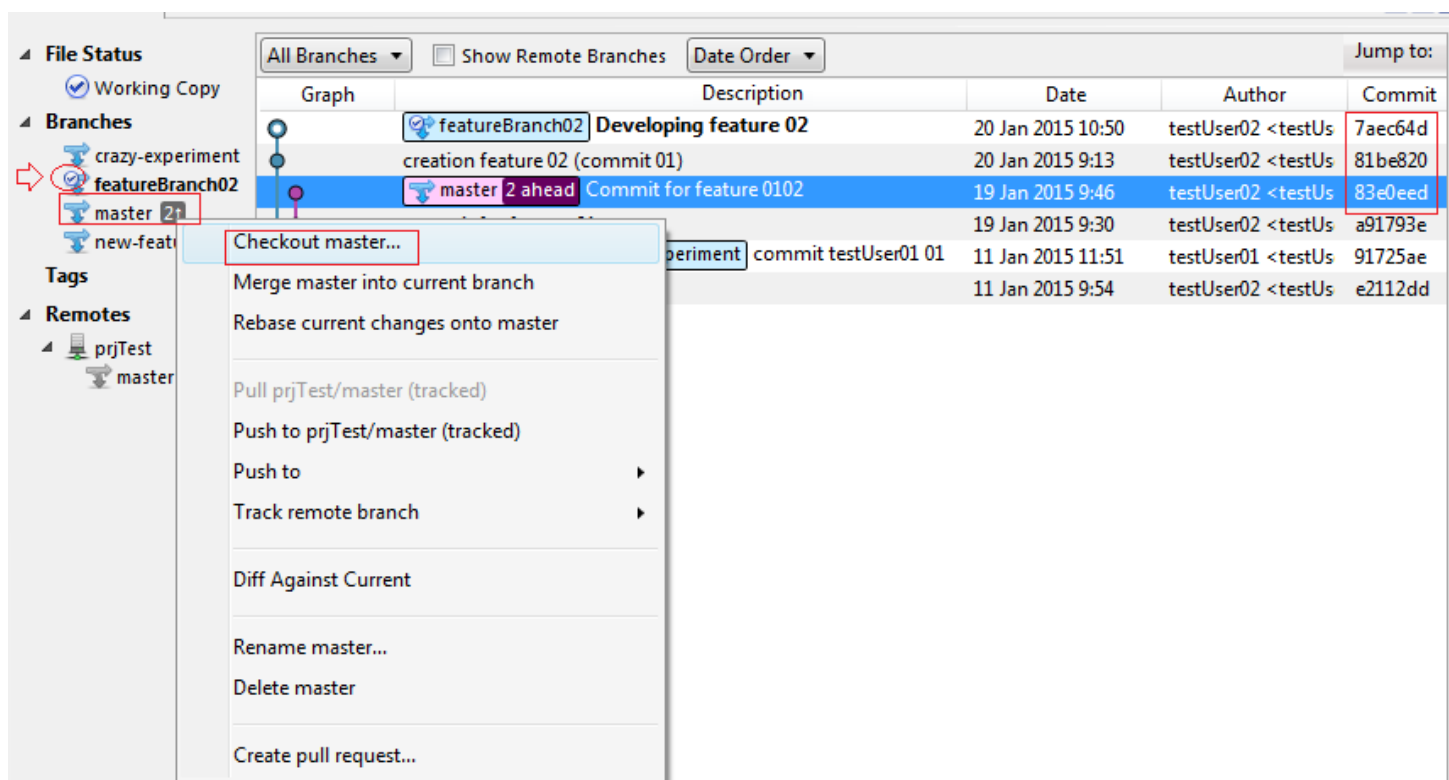
Merge in the new-feature branch

10. Point to **master**

a. Select **Master**



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



c. Log / History view

The screenshot displays the Git Log/History view in SourceTree. The left sidebar shows the file status with branches like crazy-experiment, featureBranch02, master (2 ahead), and new-feature. The main panel shows a list of commits sorted by date. The selected commit is 83e0eed, titled 'Commit for feature 0102', by testUser02. The commit details panel shows the commit hash, parents, author, date, and labels. The diff view for testFile.txt shows a hunk with lines 1-3, including additions and deletions.

Graph	Description	Date	Author	Commit
featureBranch02	Developing feature 02	20 Jan 2015 10:50	testUser02 <testUs	7aec64d
	creation feature 02 (commit 01)	20 Jan 2015 9:13	testUser02 <testUs	81be820
master 2 ahead	Commit for feature 0102	19 Jan 2015 9:46	testUser02 <testUs	83e0eed
	commit for feature 01	19 Jan 2015 9:30	testUser02 <testUs	a91793e
new-feature	commit testUser01 01	11 Jan 2015 11:51	testUser01 <testUs	91725ae
	commit - local 01	11 Jan 2015 9:54	testUser02 <testUs	e2112dd

Commit: 83e0eed0dd14882ea14822b42561b15b67265507 [83e0eed]
 Parents: a91793e617
 Author: testUser02 <testUser02@example.com>
 Date: Monday, January 19, 2015 9:46:17 AM
 Labels: HEAD, master
 Commit for feature 0102

testFile.txt

Hunk 1 : Lines 1-3
 1 + Test file - line 03
 1 2 Test file - line 02
 2 - Test file - line 01
 3 + Test file - line 01
 3 4 \ No newline at end of file

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

The screenshot shows a file explorer window for the test_project directory. It contains files .git, myFeature01.txt, and testFile.txt. Below the file explorer, a Notepad window is open, editing testFile.txt. The text in the Notepad window is as follows:

```

Test file - line 04
Test file - line 03
Test file - line 02
Test file - line 01
  
```

12. Add the changes using **Add**

13. Commit the master using **commit**



testUser02 <testUser02@example.com> Commit options...

Finishing feature 02

☐ Push changes immediately to prjTest/master

Commit **Cancel**

File Status Log / History Search 1 master 2 **Atlassian**

14. Log / History status

File Status Working Copy

Branches

- crazy-experiment
- featureBranch02
- master** 31
- new-feature

Tags

Remotes

- prjTest
- master

All Branches ☐ Show Remote Branches Date Order

Graph	Description	Date	Author	Commit
master 3 ahead	Finishing feature 02	20 Jan 2015 11:51	testUser02 <testUs	7684a89
featureBranch02	Developing feature 02	20 Jan 2015 10:50	testUser02 <testUs	7aec64d
	creation feature 02 (commit 01)	20 Jan 2015 9:13	testUser02 <testUs	81be820
	Commit for feature 0102	19 Jan 2015 9:46	testUser02 <testUs	83e0eed
	commit for feature 01	19 Jan 2015 9:30	testUser02 <testUs	a91793e
new-feature crazy-experiment	commit testUser01 01	11 Jan 2015 11:51	testUser01 <testUs	91725ae
	commit - local 01	11 Jan 2015 9:54	testUser02 <testUs	e2112dd

Sorted by file status

Commit: 7684a89a25a2fc02c5769e98f2101846b303ce56 [7684a89]

Parents: 83e0eed0dd

Author: testUser02 <testUser02@example.com>

Date: Tuesday, January 20, 2015 11:51:48 AM

Labels: HEAD, master

Finishing feature 02

testFile.txt

File Status Log / History Search

testFile.txt

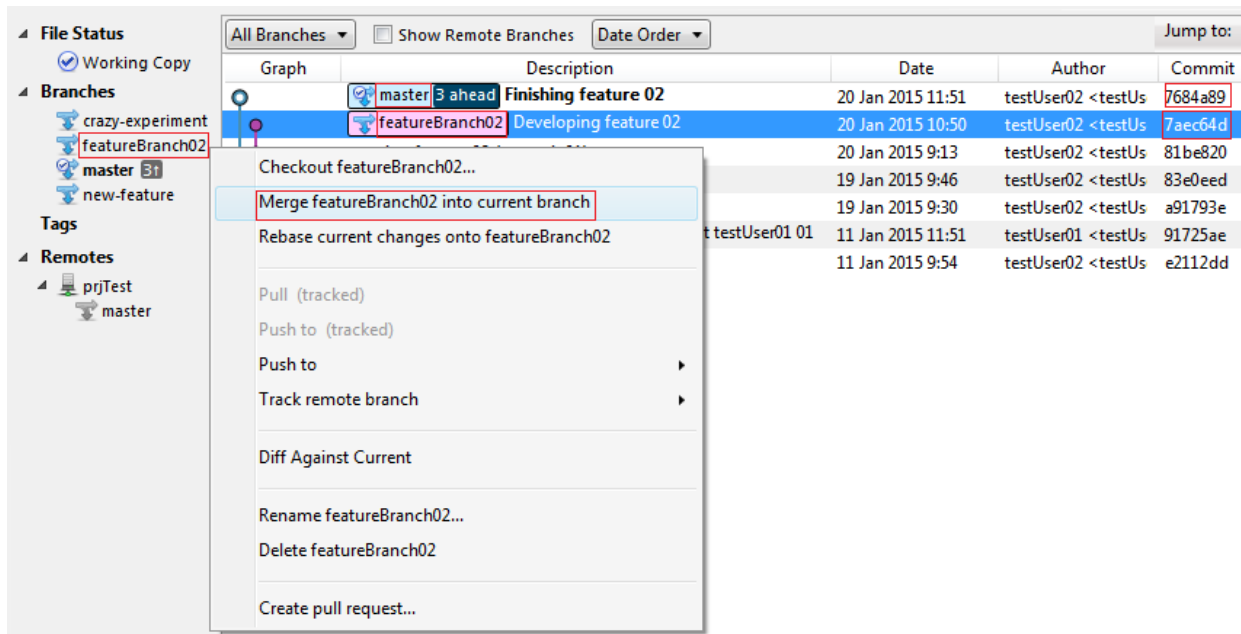
Hunk 1 : Lines 1-4 **Reverse hunk**

```

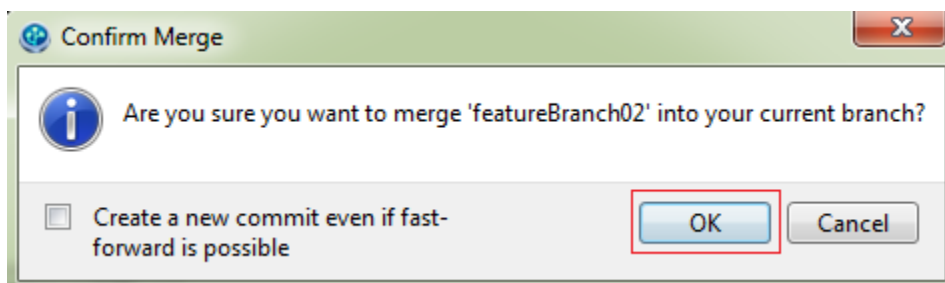
1 + Test file - line 04
1 2 Test file - line 03
2 3 Test file - line 02
3 4 Test file - line 01
4 5 \ No newline at end of file

```


- Merge with earlier created branch (e.g. featureBranch02)
Select the required branch (e.g. featureBranch02)
15. Right click and from pull down menu select **Merge <branch name> into current branch** (e.g. here branch name used is featureBrach02)



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



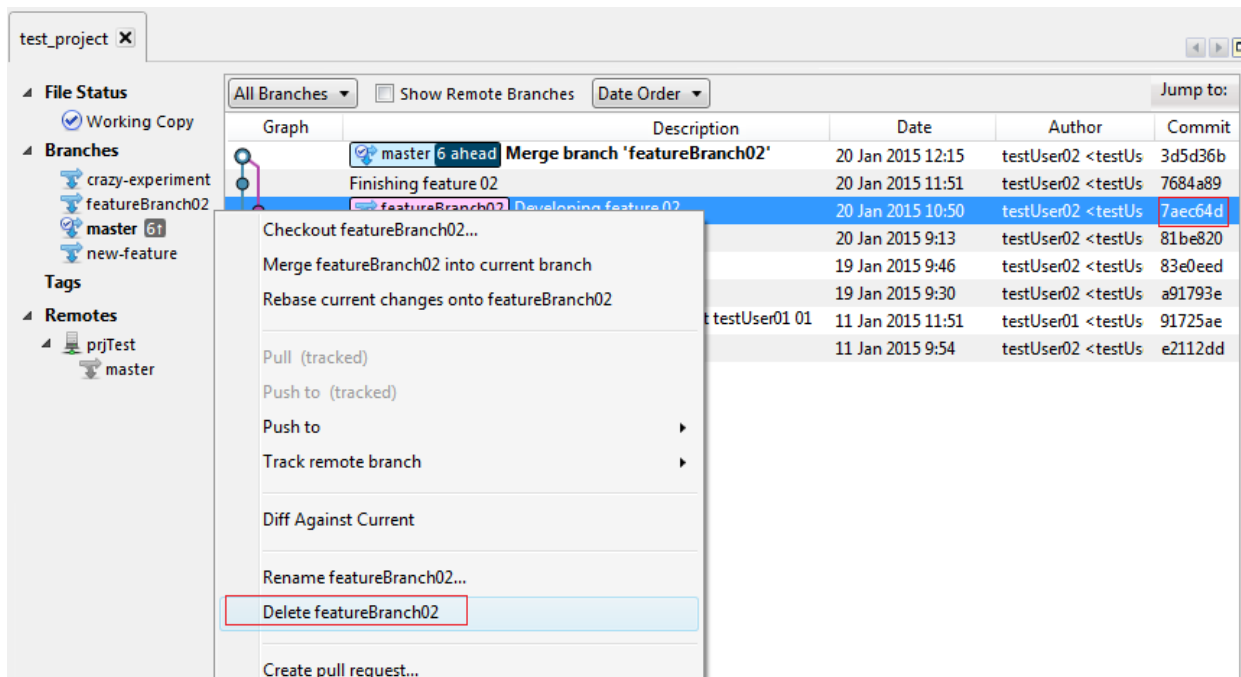
17. Log/History View

Graph	Description	Date	Author	Commit
	master 6 ahead Merge branch 'featureBranch02'	20 Jan 2015 12:15	testUser02 <testUs	3d5d36b
	Finishing feature 02	20 Jan 2015 11:51	testUser02 <testUs	7684a89
	featureBranch02 Developing feature 02	20 Jan 2015 10:50	testUser02 <testUs	7aec64d
	creation feature 02 (commit 01)	20 Jan 2015 9:13	testUser02 <testUs	81be820
	Commit for feature 0102	19 Jan 2015 9:46	testUser02 <testUs	83e0eed
	commit for feature 01	19 Jan 2015 9:30	testUser02 <testUs	a91793e
	new-feature crazy-experiment commit testUser01 01	11 Jan 2015 11:51	testUser01 <testUs	91725ae
	commit - local 01	11 Jan 2015 9:54	testUser02 <testUs	e2112dd

18. You might delete the branch you created earlier (e.g. featureBranch02)

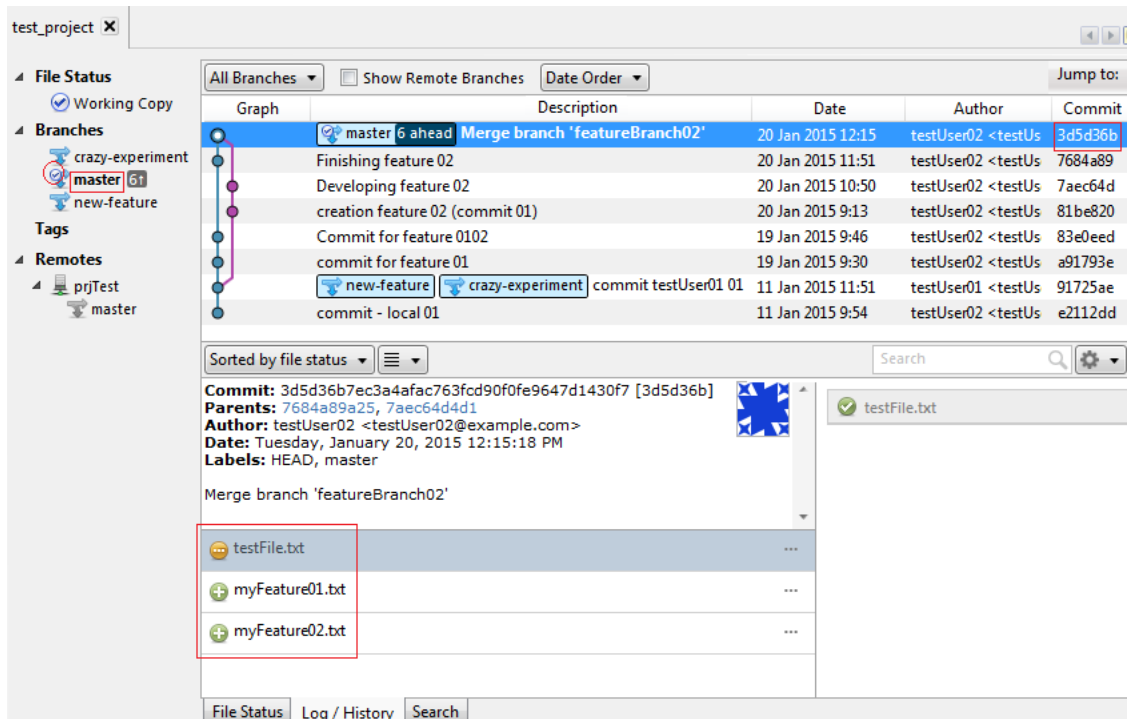
19. Select the branch to be deleted (e.g. featureBranch02)

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



21. Click on **OK** to confirm branch deletion

22. 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/>