GIT Tutorial (Beginners)

Bhabesh Bhabani Mukhopadhyay

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 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*
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.

Table of Contents

Command

git init

Getting Started

File Edit \

Clone / New

01. Setting up a   repository (Local)

Steps

1. **Create a Directory** at your window machine test_repo_<your_name> (Example : test_repo_mohsin)
2. Now in **Source Tree** click Clone/ New

Click Create New Repository

Clone / Add / Create Repository

Clone Repository   Add Working Copy   Create New Repository

Repository Type: Git

Destination Path: C:\Users\bhabesh\Documents\gitlab\test\test_repo_mohsin

Bookmarks
☑ Bookmark this repository

Name: test_repo_mohsin

Folder: [Root]

Create   Cancel
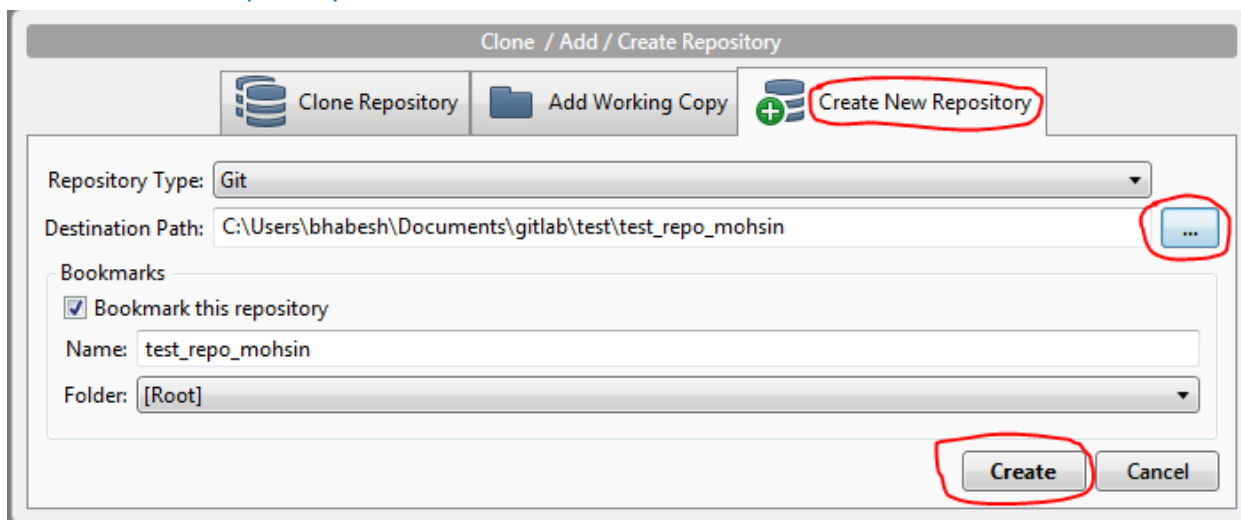
Select

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)
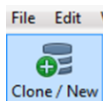2. At dashboard, Click
3. provide a justified name e.g. test_repo_<your name>02

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

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**

03.Configuration of Repository

Command

git config

Steps

1. Select the repository using double click at the source tree
2. Click **Repository/Repository Settings..**
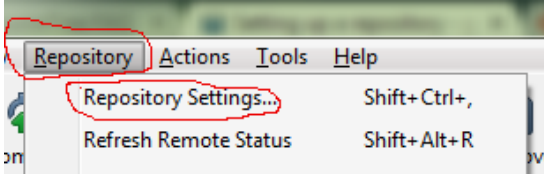
3. Click
   a) **Remotes** tab ,
   b) Select the existing (required) row from **Remote repository paths**
   c) Click **Edit** button
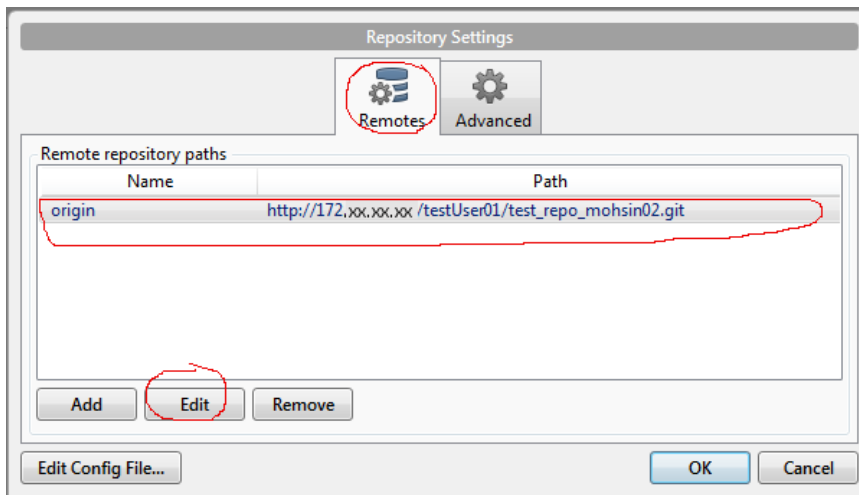
4. Make sure that
   URL/Path : valid git url
   Username : git user name (e.g. active directory user name)
   Click **Ok** (Click **Ok** again to Exit from dialogue box)
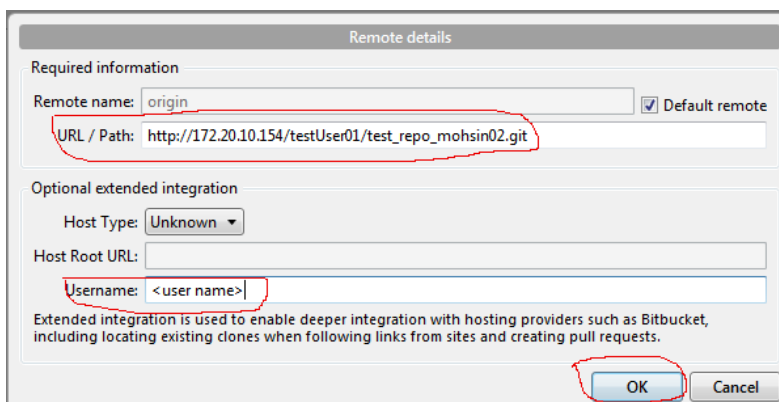
04.Saving changes

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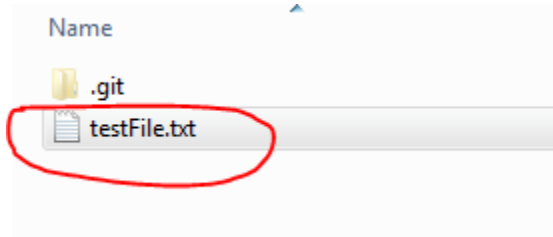
2. View changes in **SourceTree** ui
3. Click on the file (**purple** color) (e.g. testFile.txt) at 'Unstaged files' at right side of book mark pane
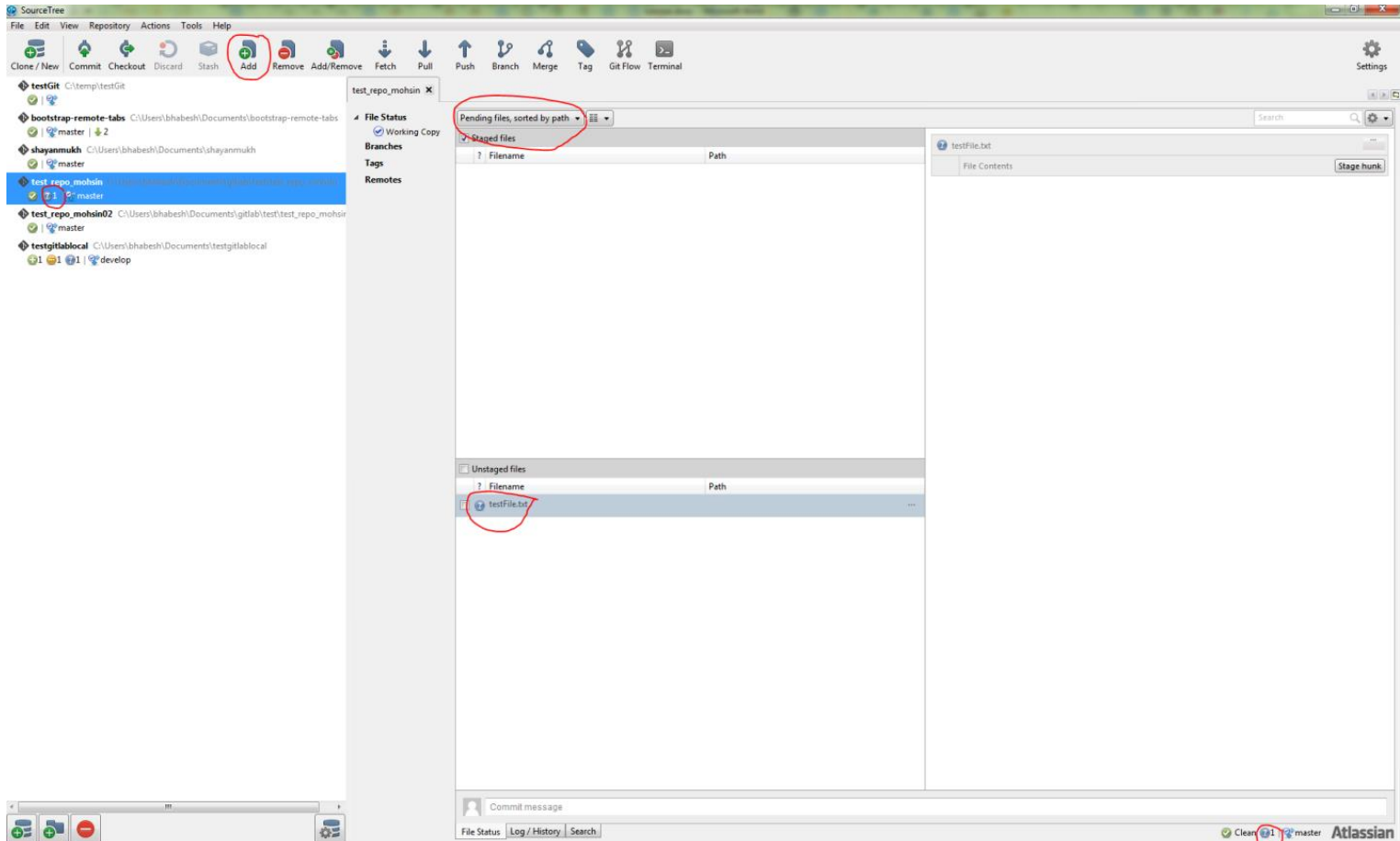
4. File will be added at staging

5. Open the file and add some text

Name	Date modified	Type	Size
.git	1/5/2015 1:29 PM	File folder	
testFile.txt	1/5/2015 1:25 PM	Text Document	0 KB

testFile.txt - Notepad

File   Edit   Format   View   Help

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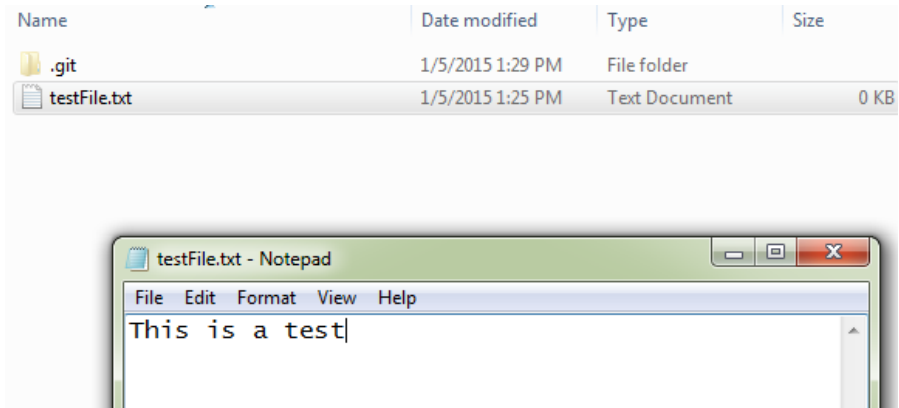
6. File will again visible at 'Unstaged Files' area with **yellow** color
7. Select the file and click add (if needed)
8. Changes will be saved at staging area

Add
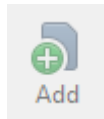
05.Git Commit

Steps

1. Select file/s from 'Staged Files'
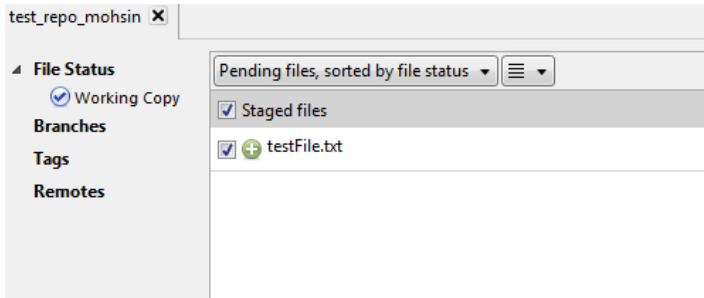
Command

git commit

2. Check the changes
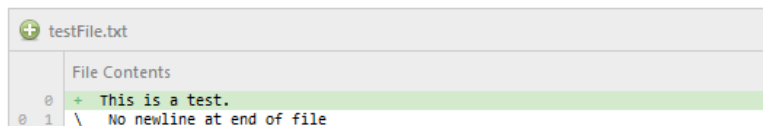
3. Click on commit
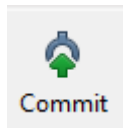
4. Give some comment/justification of commit in commit dialogue box and click **commit** button
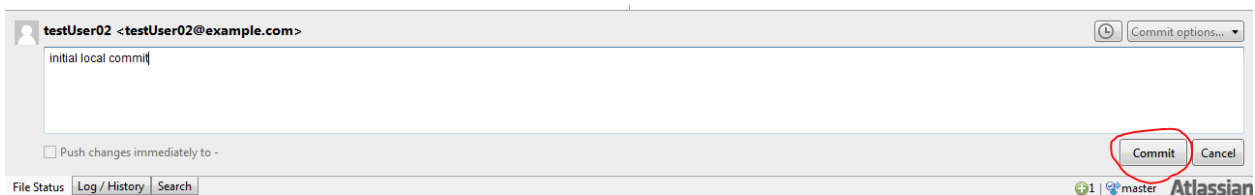
5. File will be committed. It can be viewed as below :

6. Modify some text

Name	Date modified
.git	1/5/2015 2:22 PM
testFile.txt	1/5/2015 2:22 PM

testFile.txt - Notepad

File Edit Format View Help
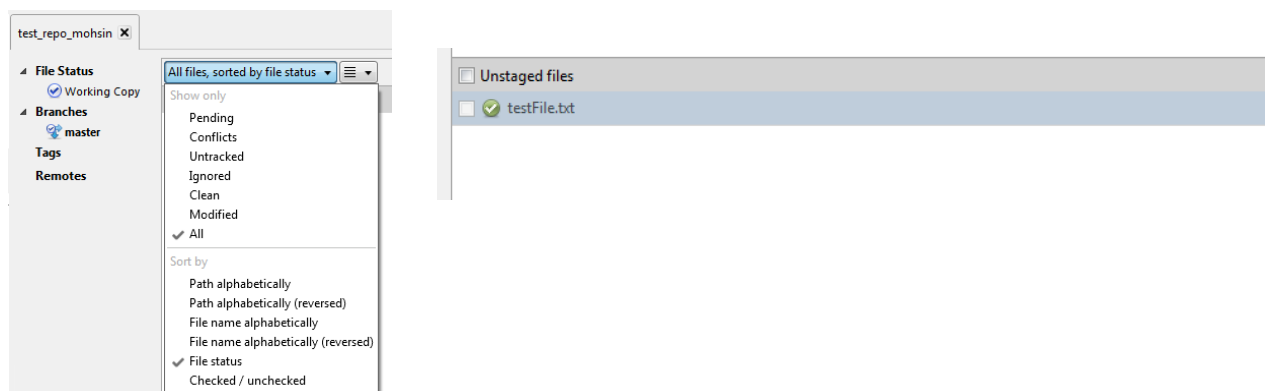
This is a beautiful test.
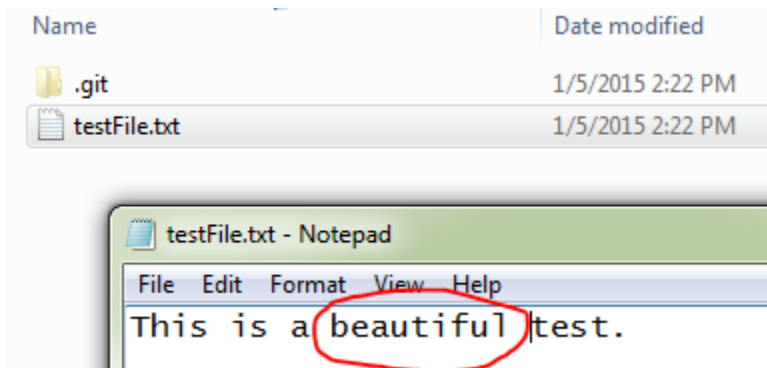
7. In SourceTree the changes are like this

testFile.txt

Hunk 1 : Lines 0-0   [Stage hunk] [Discard hunk]

```
0      -  This is a test.
1  0   \    No newline at end of file
   1   +  This is a beautiful test.
2  2   \    No newline at end of file
```
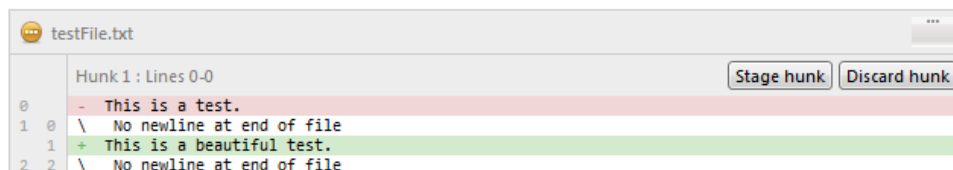
8. It is possible to stage only the portion or full file
9. Click the file at Unstage area ('**git add**' command will be applied) .

☐ Unstaged files
☐ 😐 testFile.txt
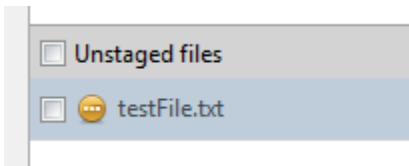
10. After add file will be staged and file color will change in two places
    a. In '**Unstaged files**' area
              Color will change from yellow to green (with icon changes to check)

☐ Unstaged files
☐ ✓ testFile.txt
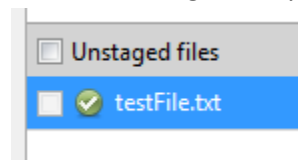
    b. And in '**Staged files**' area will change from green to yellow (with icon changes)

☑ Staged files
☑ 😐 testFile.txt
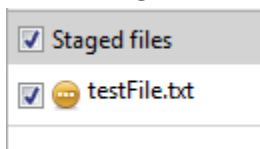
11. Select the file and commit. File will be committed locally.

06.Viewing Old Commits

Steps

Command

git checkout

1. Click on log view

File Status | Log / History | Search

2. Book Mark Pane looks something similar

test_repo_mohsin C:\Users\bhabesh\Documents\gitlab\test\test_repo_mohsin
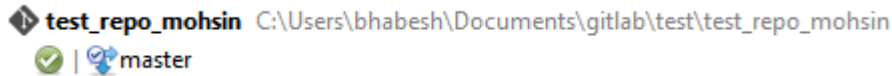
✅ | 🌿 master

3. See the commit log

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

All Branches ▼ ☐ Show Remote Branches   Date Order ▼   Jump to:

4. Click on **checkout**

🔄
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**

⚠ Warning: creating detached HEAD

⚠ Checking out this commit will create a detached HEAD, and you will no longer be on any branch (unless you subsequently create a new one).

Are you sure that's what you want? If not, press Cancel and pick another commit, or if this is a remote branch you can use the 'Checkout New Branch' tab instead to create a new local branch.

OK | Cancel

7. BookMark pane looks something similar

test_repo_mohsin C:\Users\bhabesh\Documents\gitlab\test\test_repo_mohsin

✅ | 🌿 (detached from fc863e3)

8. Right side pane for log :
    a. For all Branch
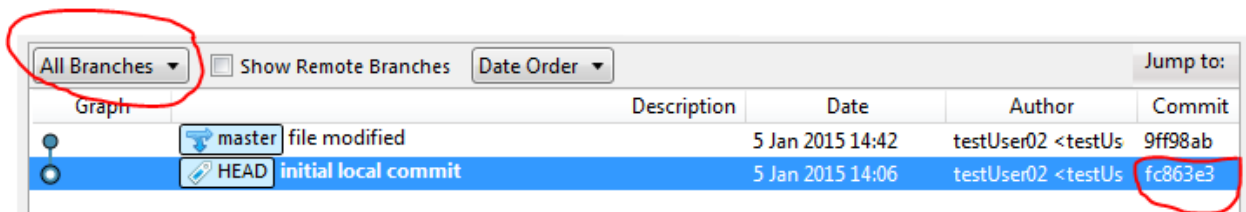
    b. For Current Branch
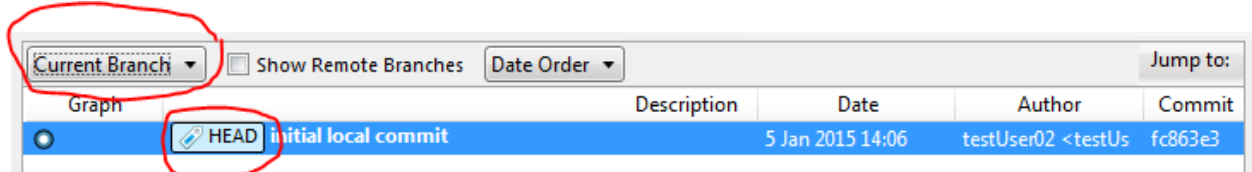
9. Click on **File Status**
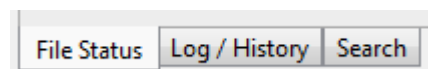
10. **Unstaged files** listing
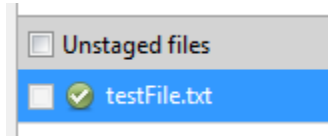
11. Open the file and modify
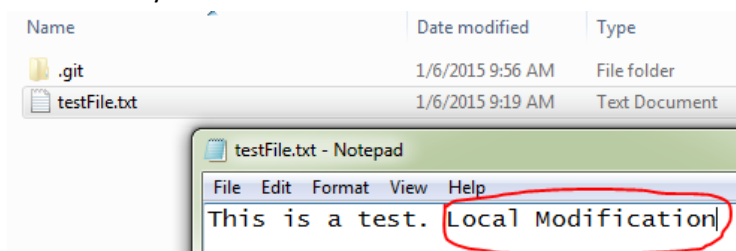
12. **Unstaged files** listing
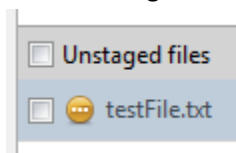
13. Changes in file
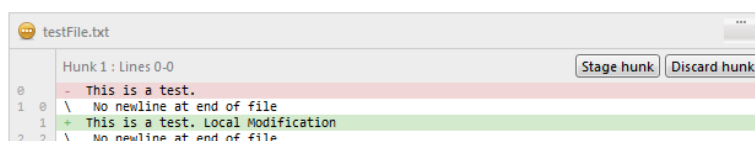
14. Check the file in **Unstaged files** pane to add to staged area
15. Click on **Commit**
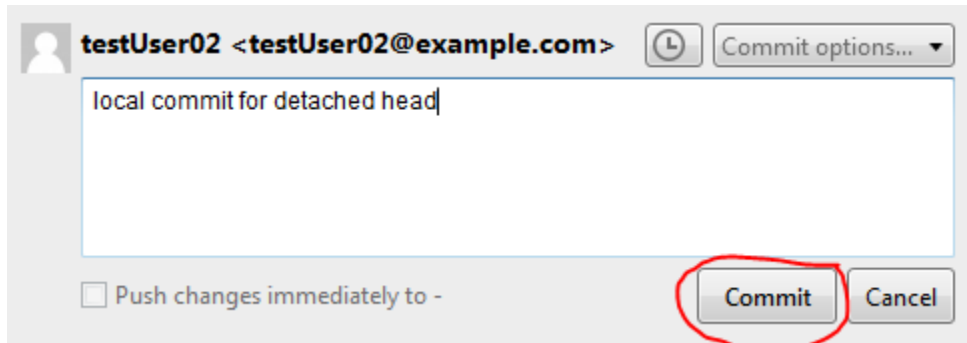
16. Click on **CheckOut**

17. Select **Master**
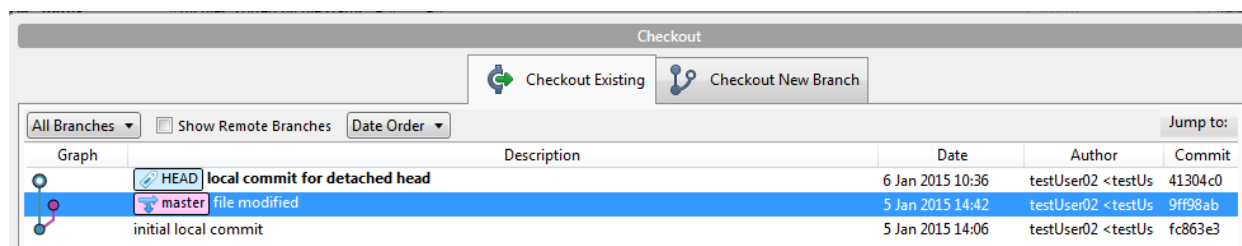
18. Click **OK**
19. You will find your earlier work is not modified.
20. It proves checkout an earlier commit will not disturb the present master branch.
21. However this is not the case of checkout a file.

Undoing Changes

07. Revert

Steps

1. Open the file and do some changes

Name	Date modified
.git	1/6/2015 1:55 PM
testFile.txt	1/6/2015 1:54 PM

testFile.txt - Notepad

File Edit Format View Help

This is a beautiful test.
Some changes for testing revert.

2. Commit it

All files, sorted by file status ▼

☑ Staged files

☑ 😶 testFile.txt

😶 testFile.txt

Hunk 1 : Lines 1-2   Unstage hunk

1   -   This is a beautiful test.
2   1   \   No newline at end of file
    2   +   This is a beautiful test.
    3   +   Some changes for testing revert.
3   4   \   No newline at end of file

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

line added to check revert

☐ Push changes immediately to -   Commit   Cancel

3. Select the recent commit , right click and select **Reverse commit**
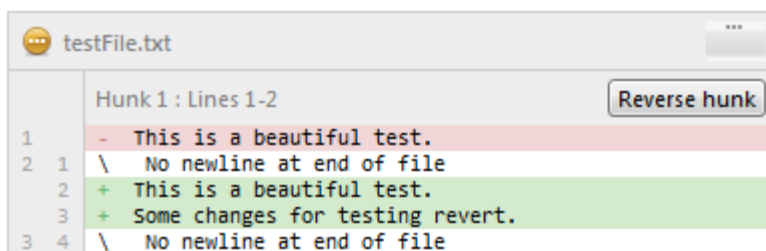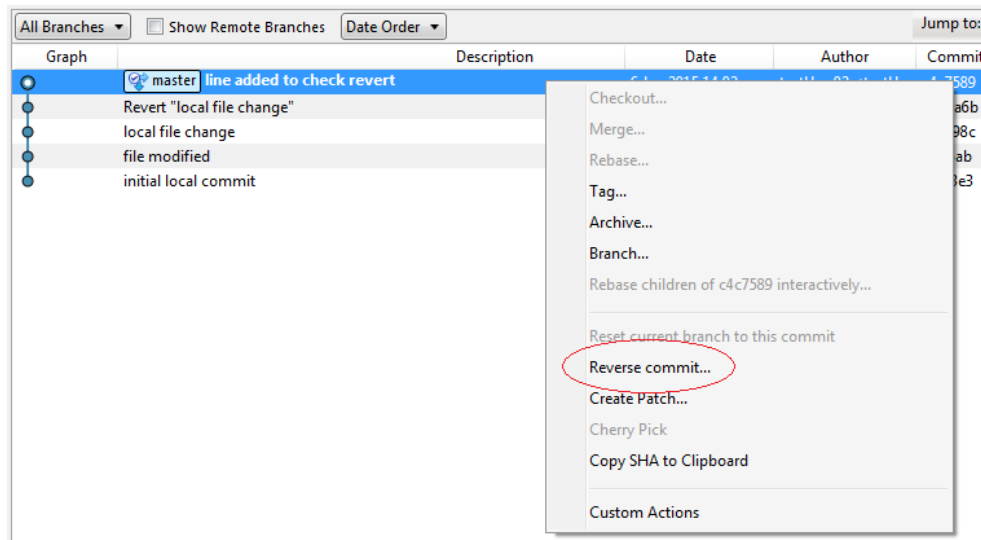
4. Click **Yes** on Dialogue box
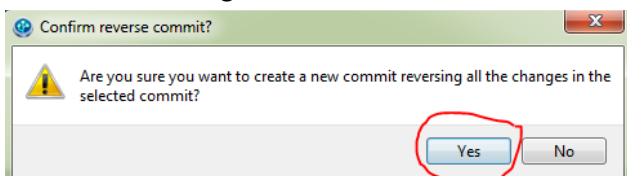
5. The contents will be earlier commit stage

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
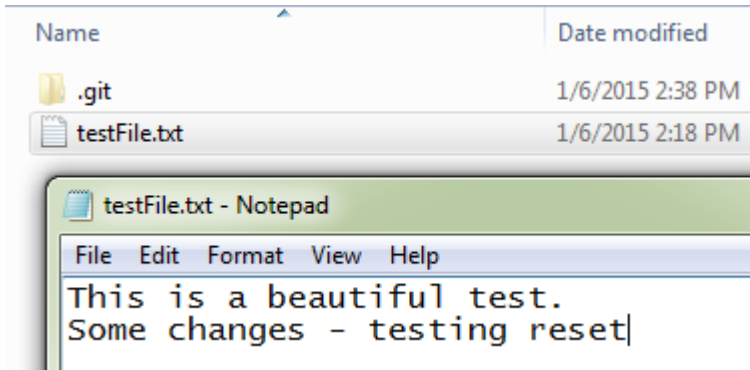
2. Add another File
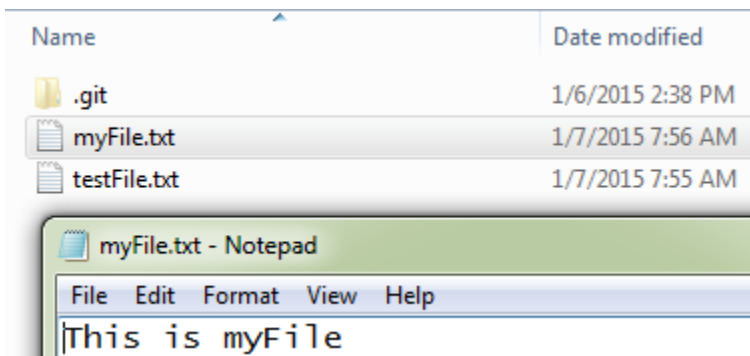
3. Add for staging
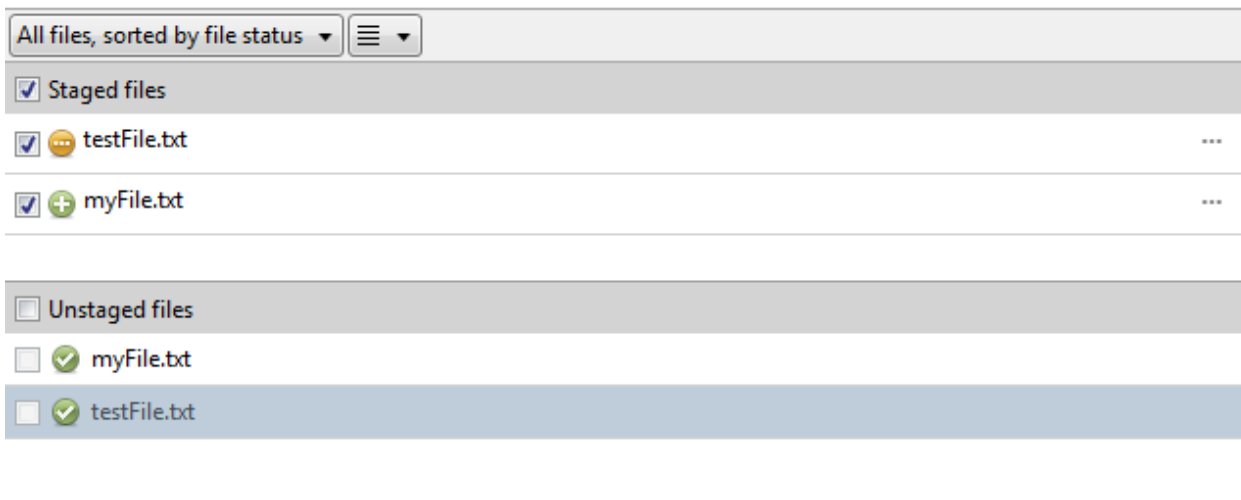
4. Reset the file (e.g. myFile.txt) .. **It will be unstaged**
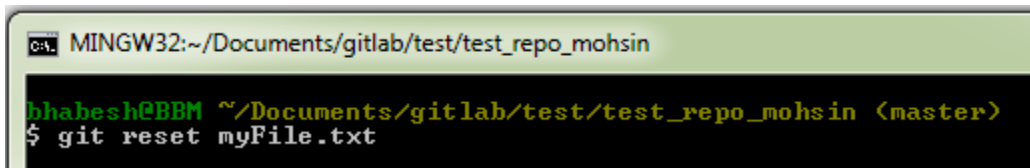
5. Commit Files seperately
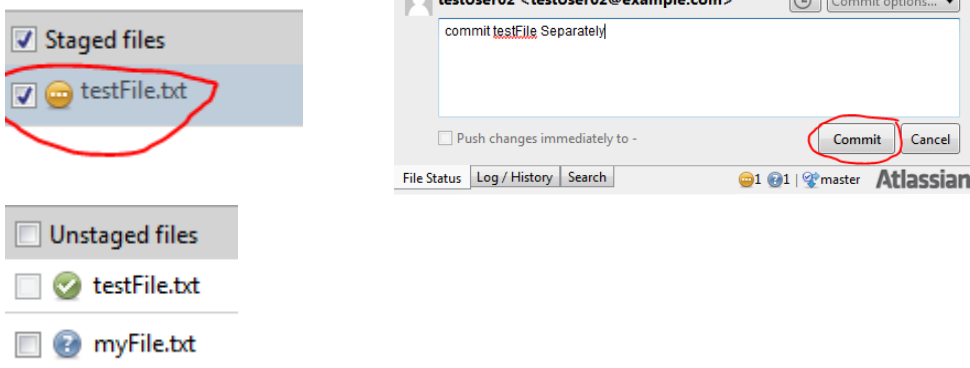
6. Add the other file from unstaging area
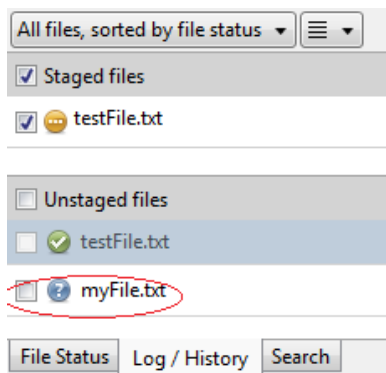
7. Create another file (e.g. hello.txt)
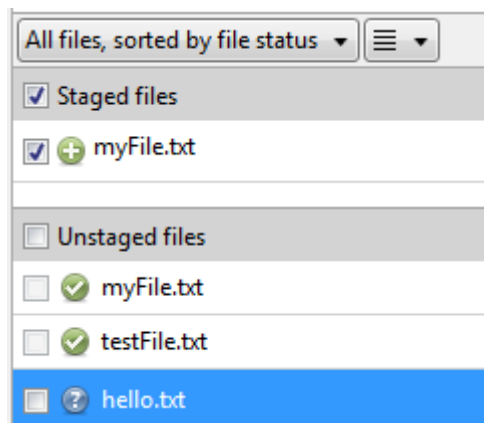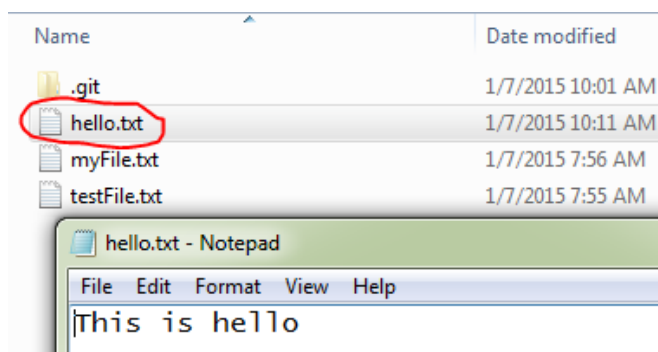
8. Now Add the file (from Upstaged to staged) :   Click

9. Commit :

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

Commit for hello and others

☐ Push changes immediately to -   Commit   Cancel
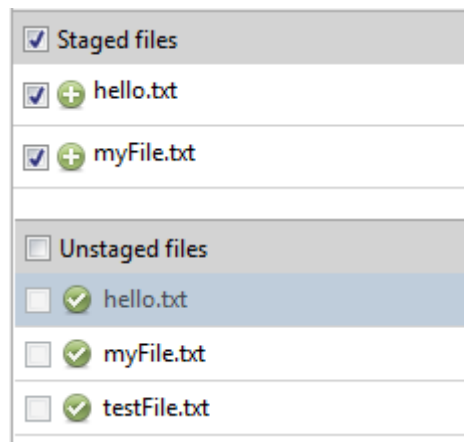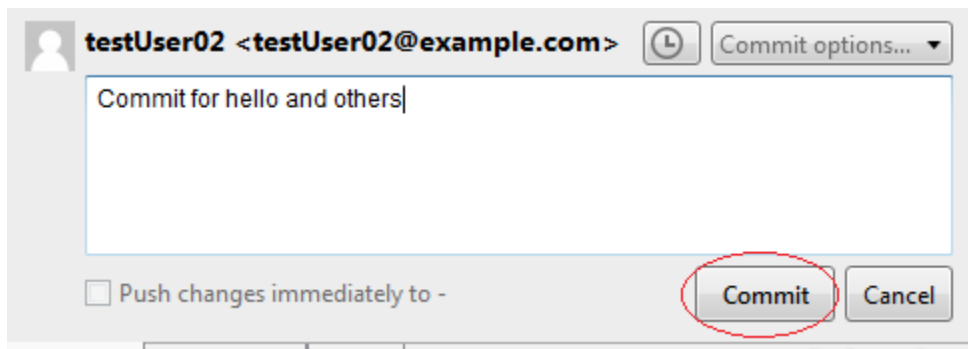
10. Edit the newly committed file (e.g. hello.txt) again :

Name	Date modified
.git	1/7/2015 10:26 AM
hello.txt	1/7/2015 10:11 AM
myFile.txt	1/7/2015 7:56 AM
testFile.txt	1/7/2015 7:55 AM

hello.txt - Notepad

File Edit Format View Help

This is hello
File edited

☐ Unstaged files   😐 hello.txt

☐ ✅ myFile.txt   Hunk 1 : Lines 1-2

☐ ✅ testFile.txt   1   -   This is hello
                           2 1 \   No newline at end of file
☐ 😐 hello.txt   2 +   This is hello
                           3 +   File edited
                        3 4 \   No newline at end of file
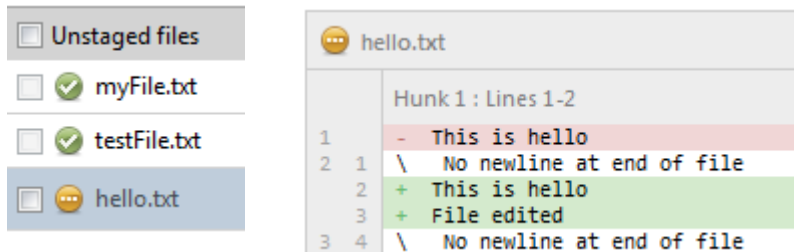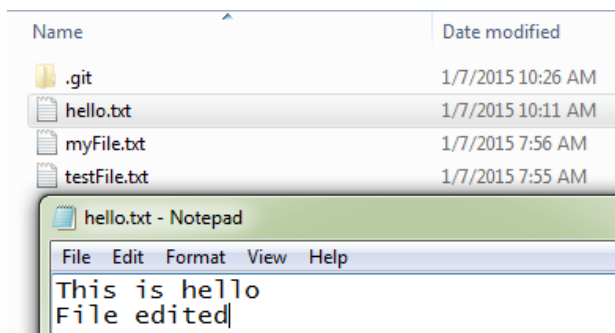
11.   Add the file to staging a 🗋⊕
                              Add

12.   Commit : 🔼 Commit

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

Commit for hello again

☐ Push changes immediately to -   Commit   Cancel

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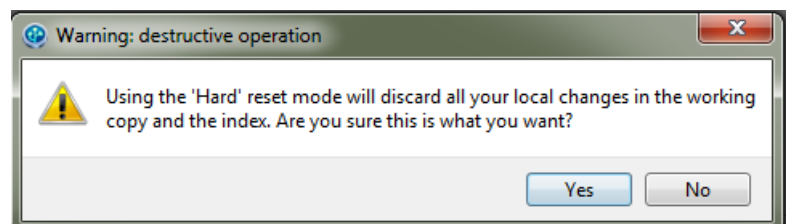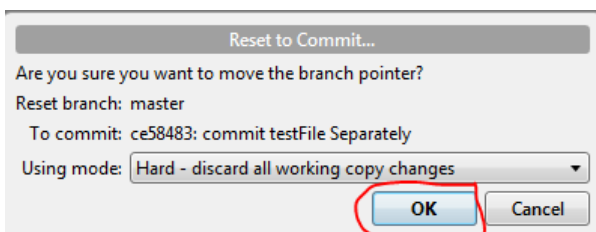
16. Commit Logs : (**addition and modification lost**)

Current status lost

Collaborating

Syncing

09.Remote
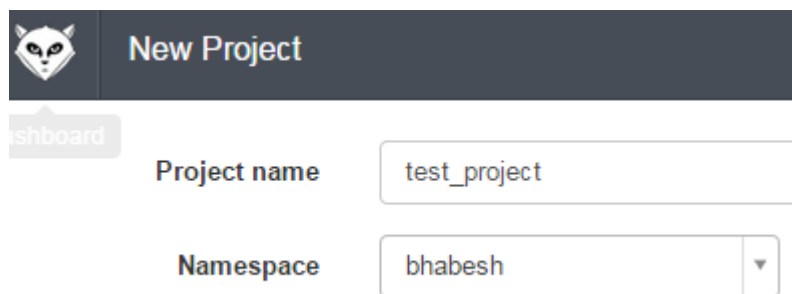
Command

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
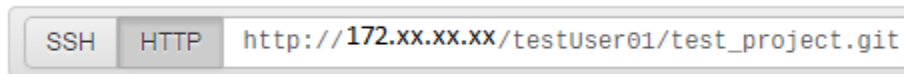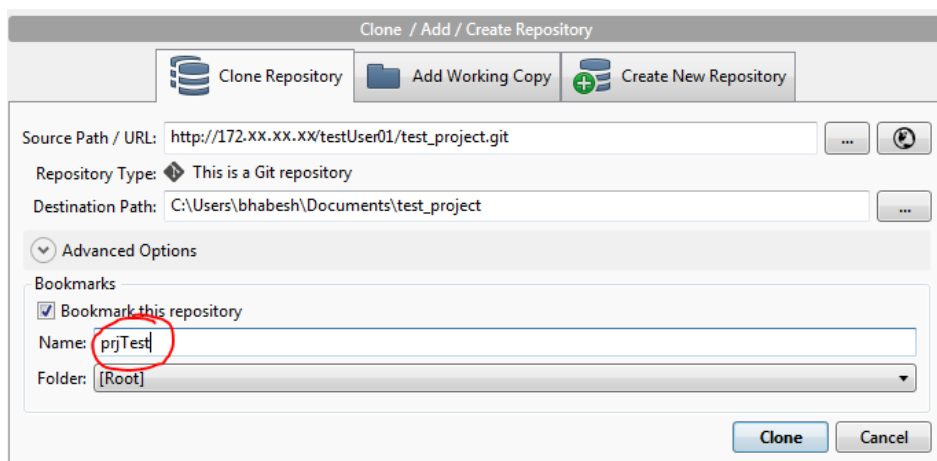
2. Copy the **url**

3. Click **Clone/New**
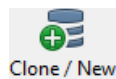
4. Click **Repository / Repository Settings...**
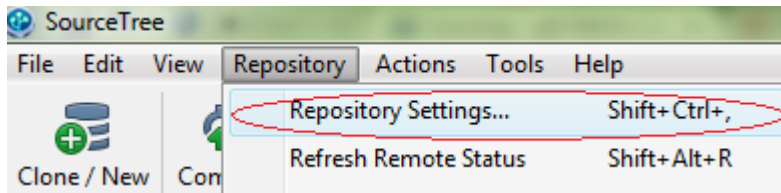
5. Repository settings view as below :

6. Remove the entry **origin** (to understand the concept for the time being)
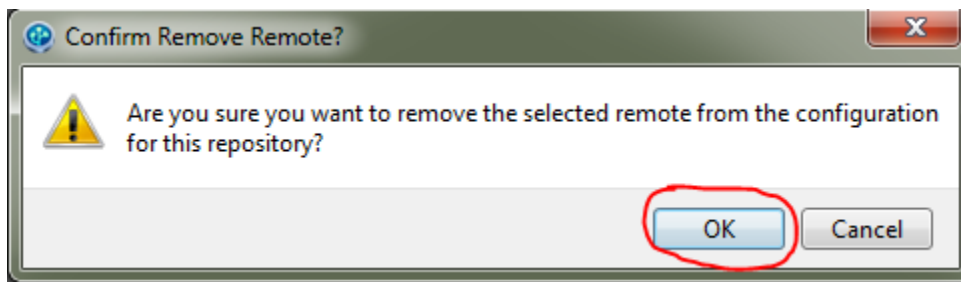
7. Click **Add**
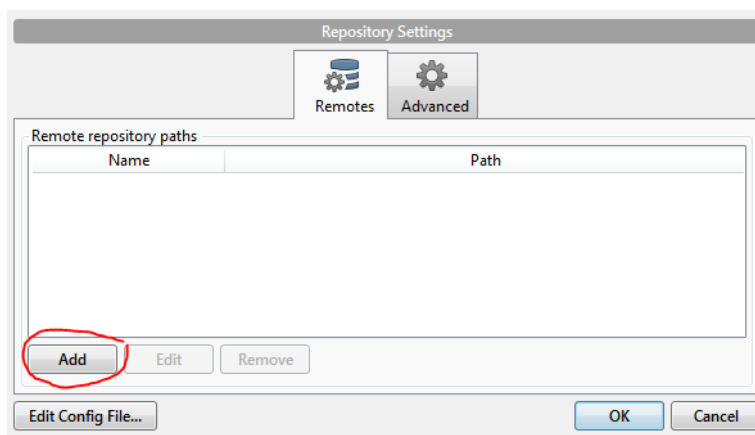
8. Fill it accordingly

9. Repository settings should look something similar :

10. Click **Ok**

10.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**)

test_project

Name

.git

testFile.txt

testFile.txt - Notepad

File   Edit   Format   View   Help

Test file – line 01

☑ Staged files

☐ Unstaged files

☐ ❓ testFile.txt
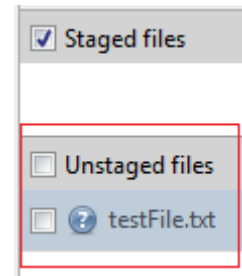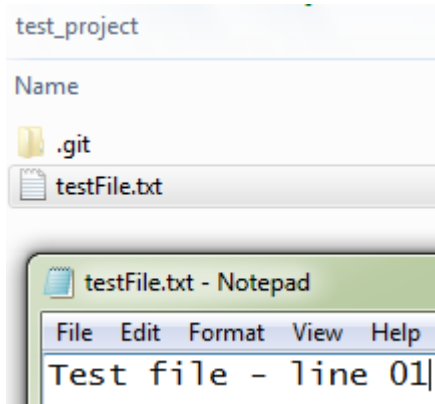
2. Add the file from '**Unstaged files**' to '**Staged files**' using **Add**

☑ Staged files

☑ ⊕ testFile.txt

☐ Unstaged files

⊕ Add
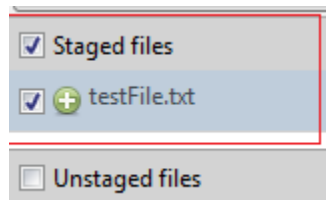
3. Commit to the local repository

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

commit - local 01

☐ Push changes immediately to -   Commit   Cancel
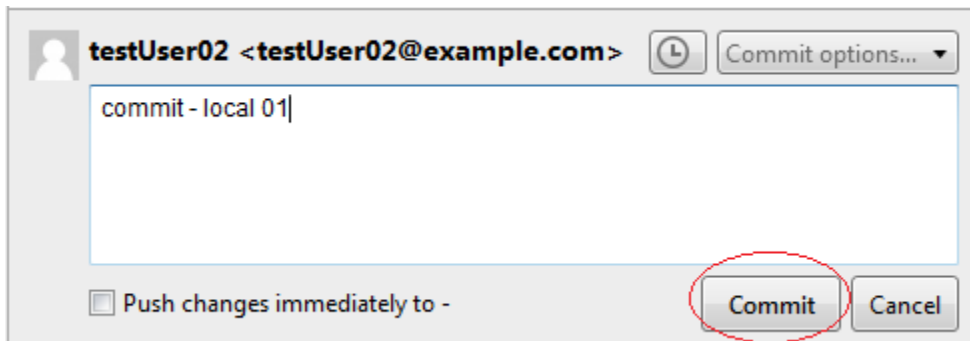
4. Click '**Push**'

↑ Push

5. In push dialogue box make sure you select (type) **master** as your remote and local branch
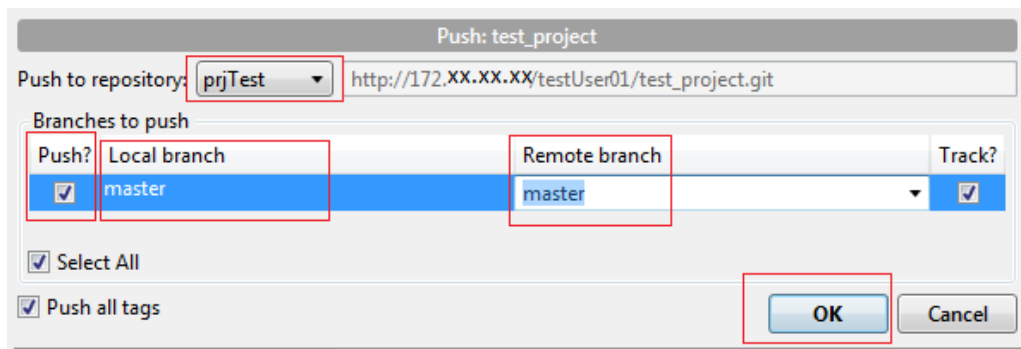
6. Using GitLab interface, we can see the remote commit as shown below :
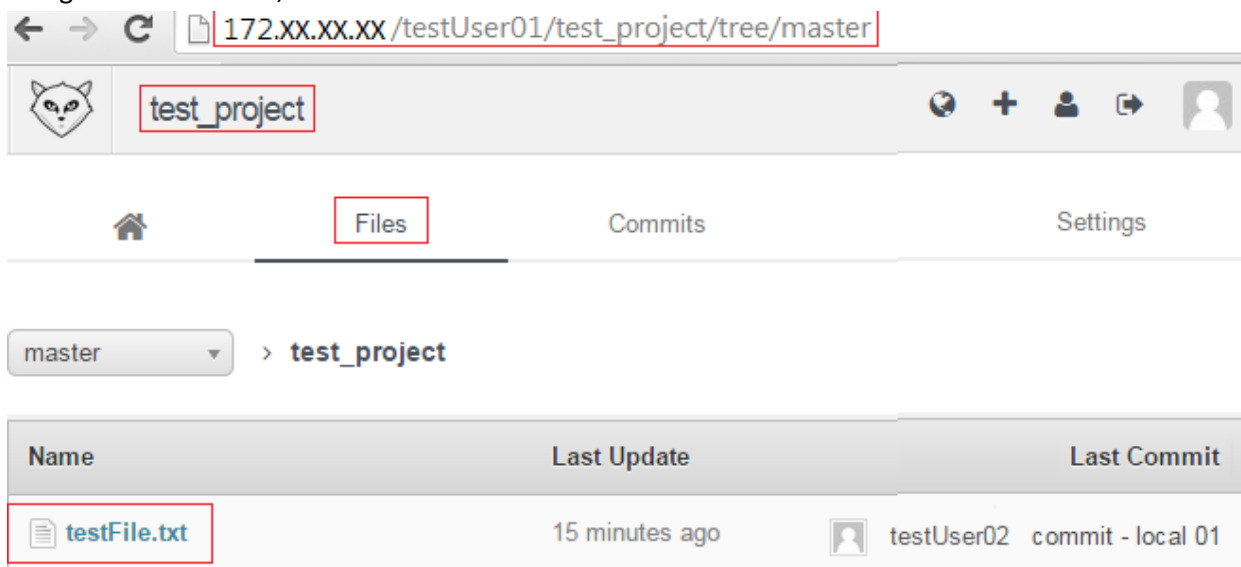
7.

11.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)
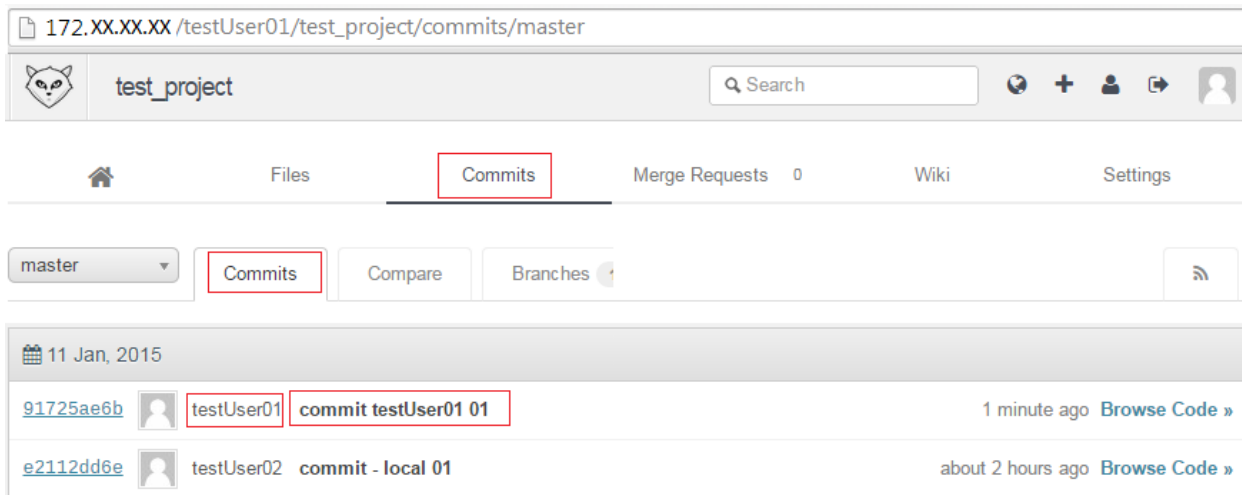
2. Click on **Fetch**

3. Fetch dialogue box will appear
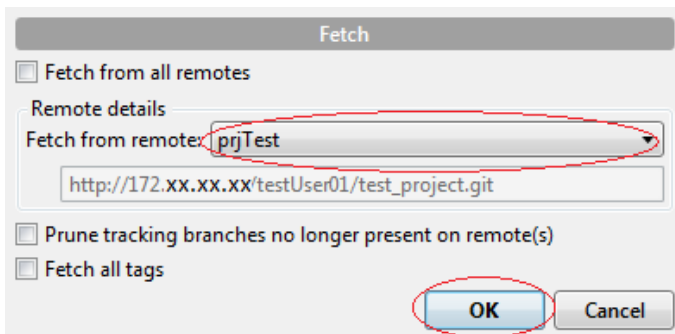
4. Right side pane looks something like below :
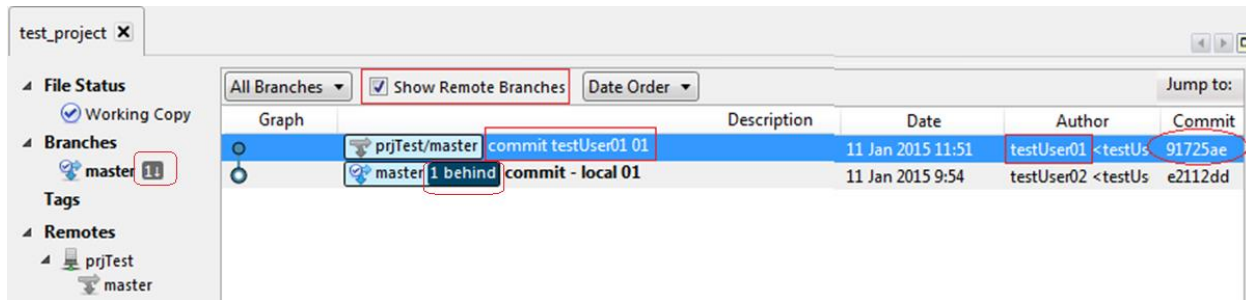
Till date the modified file is not in local repository but the **fetch** brings the commit from remote repository to local.

12.Pull

Merging upstream changes into local repository

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

Command

git pull

Steps

1. Click '**Pull** '

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

2. Click ok in the dialogue box
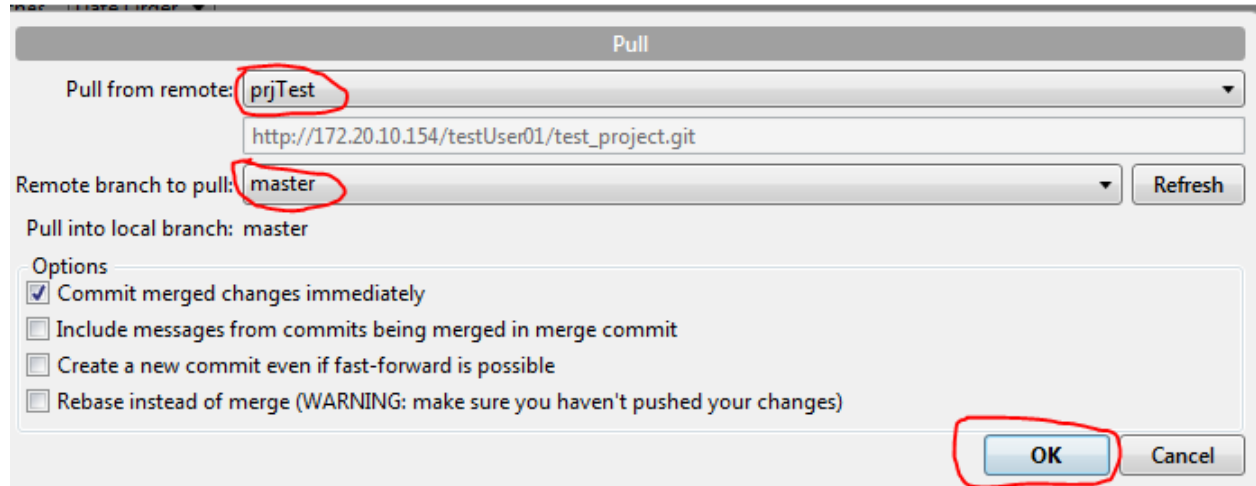
3. Log/history pane should look something like below :
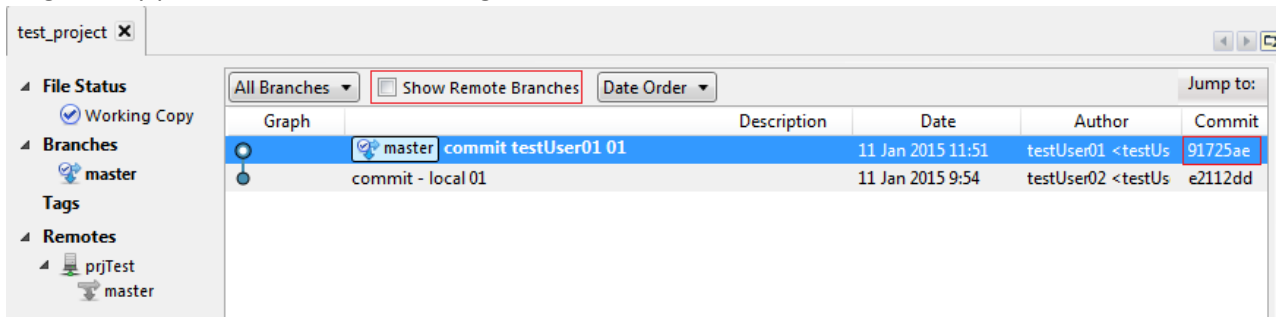
Now the modified file from the remote repository is in local along with the commit

Using Branch

13. 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
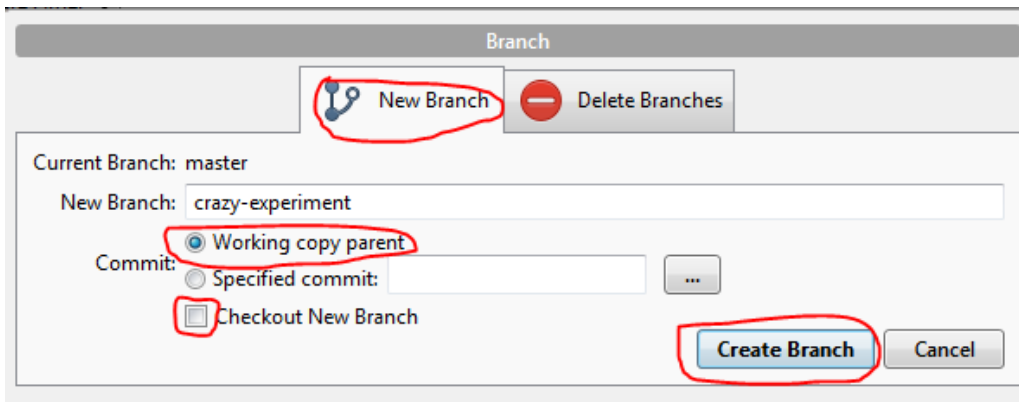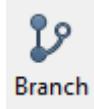
3. A branch with pointer to working copy will create.
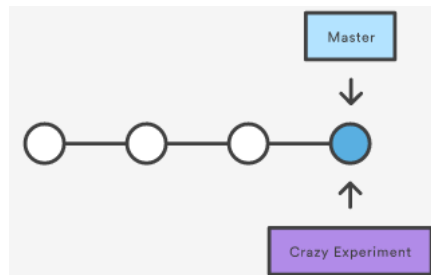
4. To working with this branch one should use the **checkout**
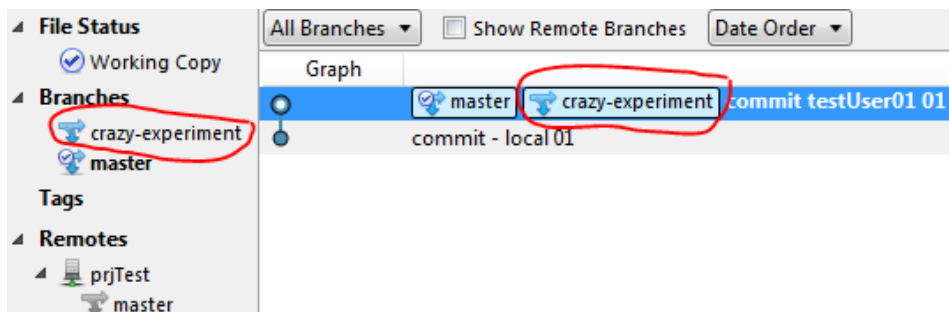
Delete a Branch

Steps

1. Click **Branch**

2. Select the branch to be deleted (e.g. crazy-experiment)
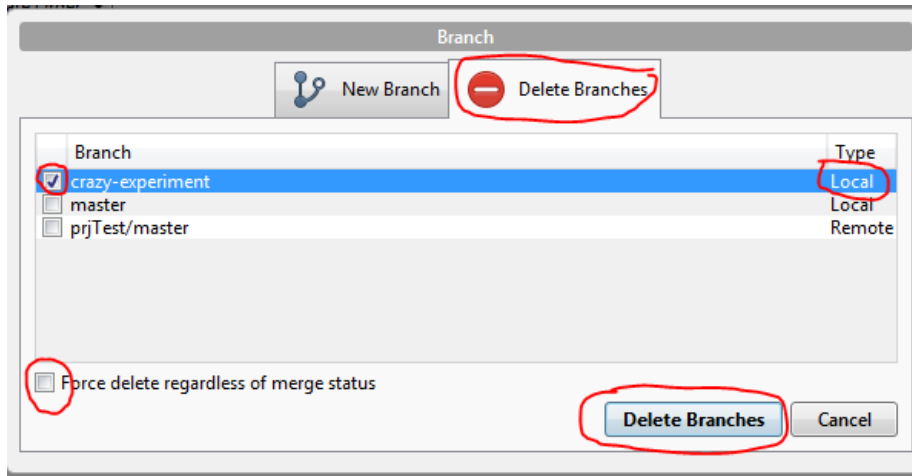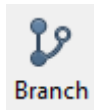
3. Make sure about the type of the branch (local/remote)
4. If the branch is not merged then will get error. Force delete enables to delete the branch and no concern with merge status.
5. Click **Delete Branches**

6. Click **OK** button
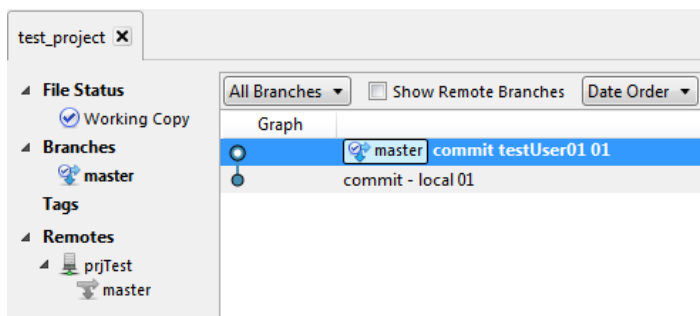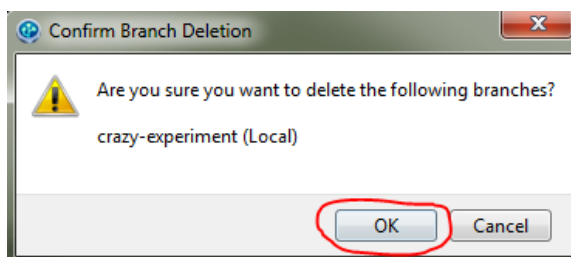
Checkout

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

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

A. Using of Checkout **after** creating a Branch
B. Creation of Branch at the time of **Checkout**

Let's discuss both the steps

Command

gitcheckout

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

Steps
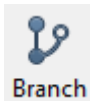
1. Click **Branch**

   Branch

2. Write a branch name (e.g. crazy-experiment). Uncheck the **Checkout New Branch** checkbox
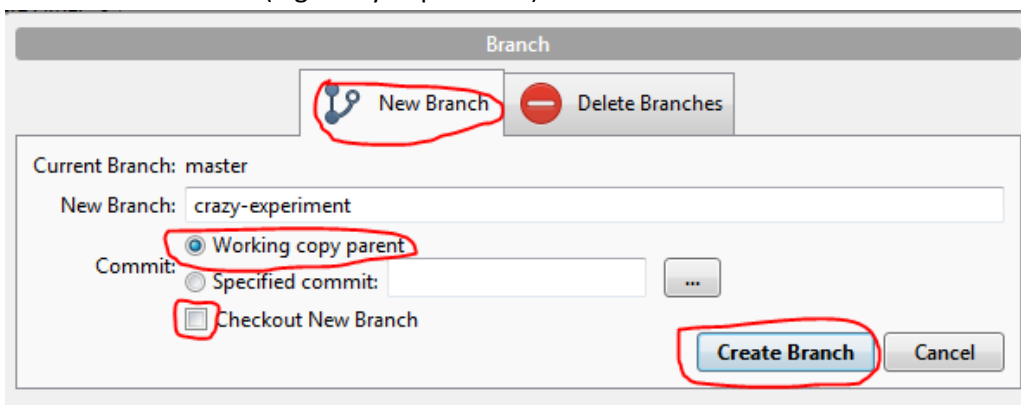
3. A branch with pointer to working copy will create.
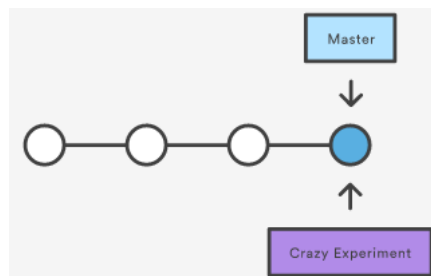
4. To working with this branch one should use the **checkout**
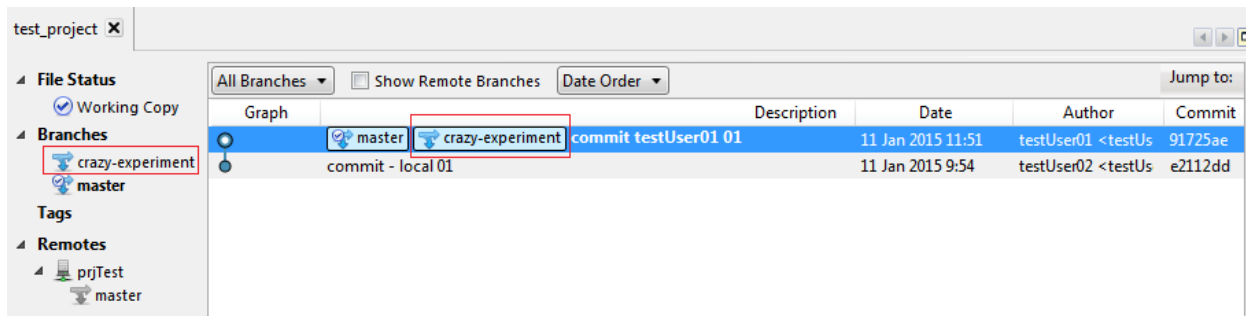
5. Different views
   a. **Log / History**

   b. **File Status**
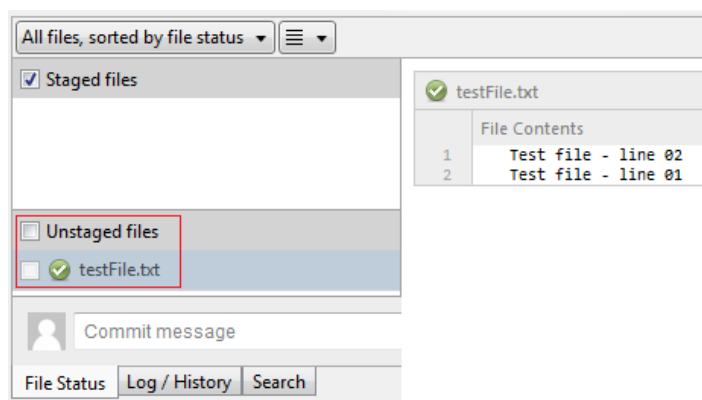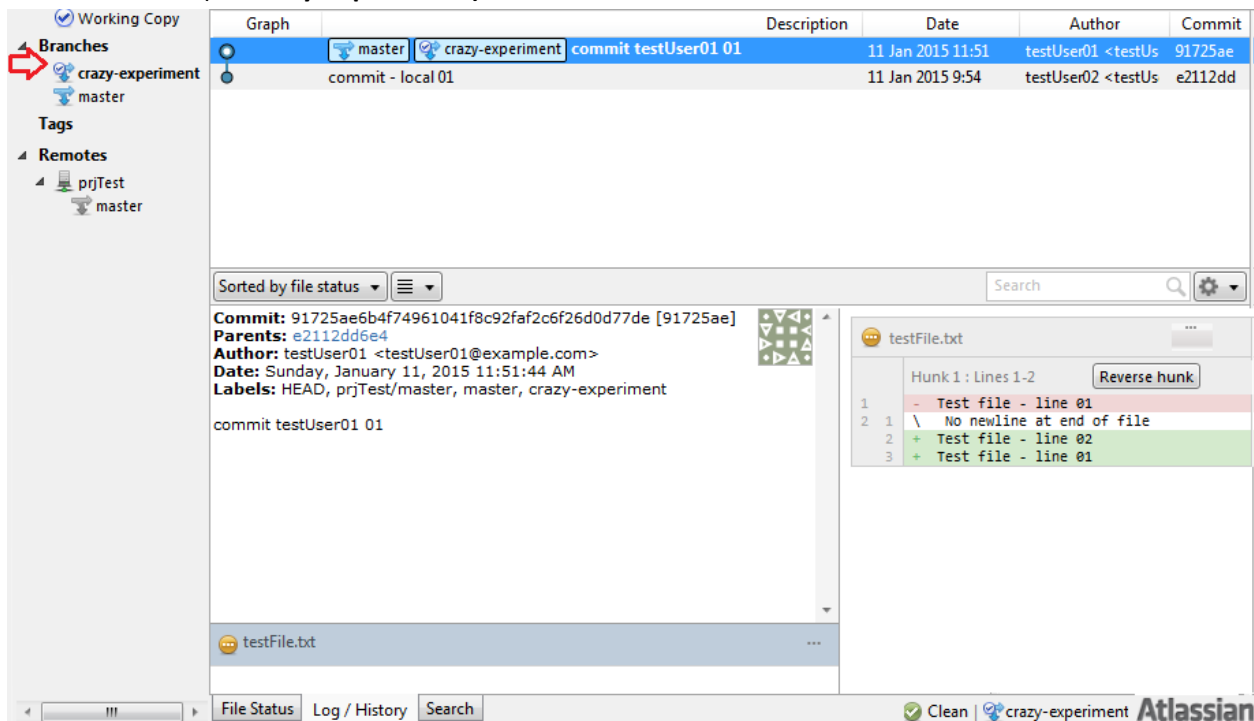
6. Click on **Checkout**
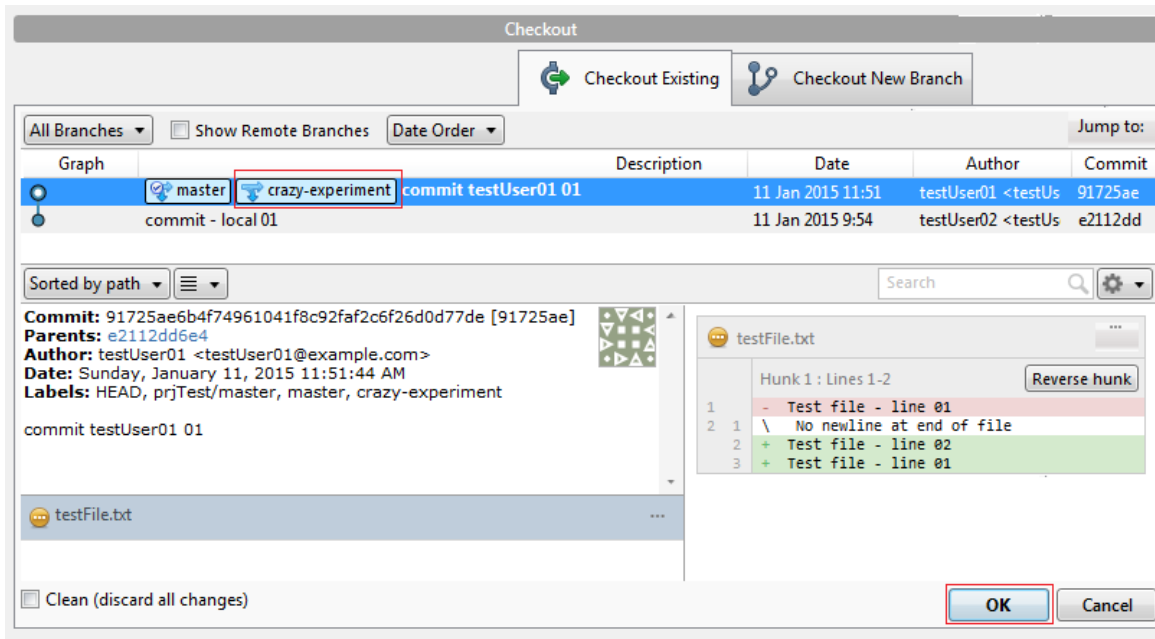
7. Select the branch at **Pick Branch** dialogue and click **OK**
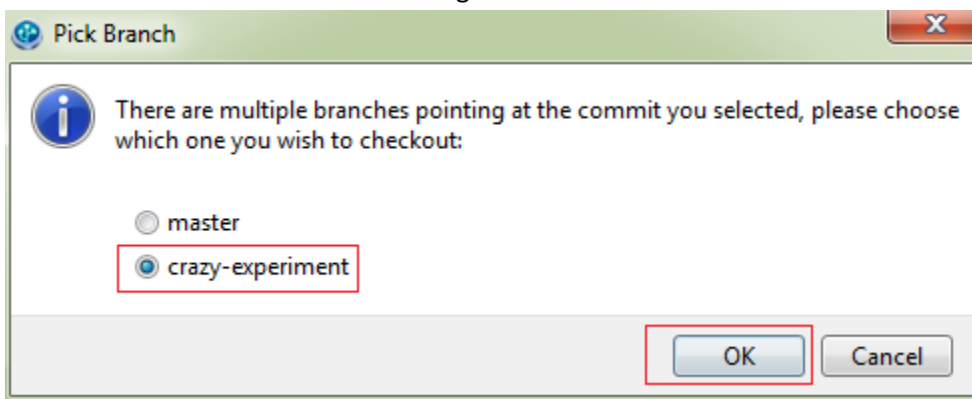
8. After **checkout** the little arrow (with check mark) ![icon] (i.e. ![icon]) will be shifted to the new branch (i.e. **crazy-experiment**)

B. Creation of *Branch* **at the time** of *Checkout*

Steps

1. Click **Checkout**

   Give a new local branch name (e.g. **new-feature**)
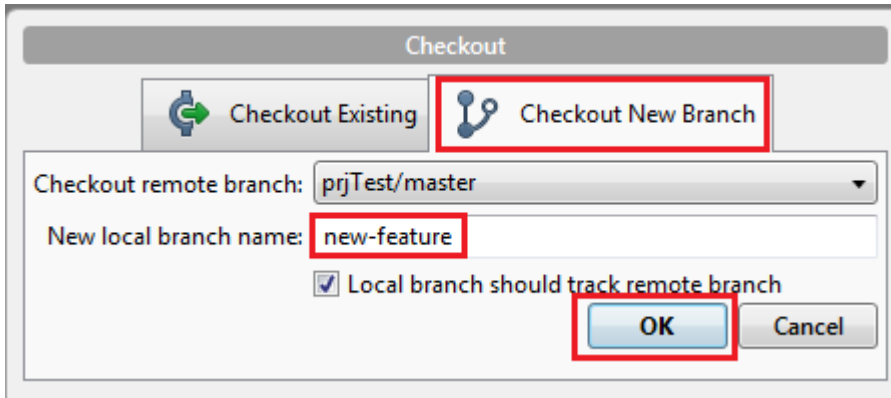
2. A new branch will be created and set. The little arrow will point the newly created branch

Merge

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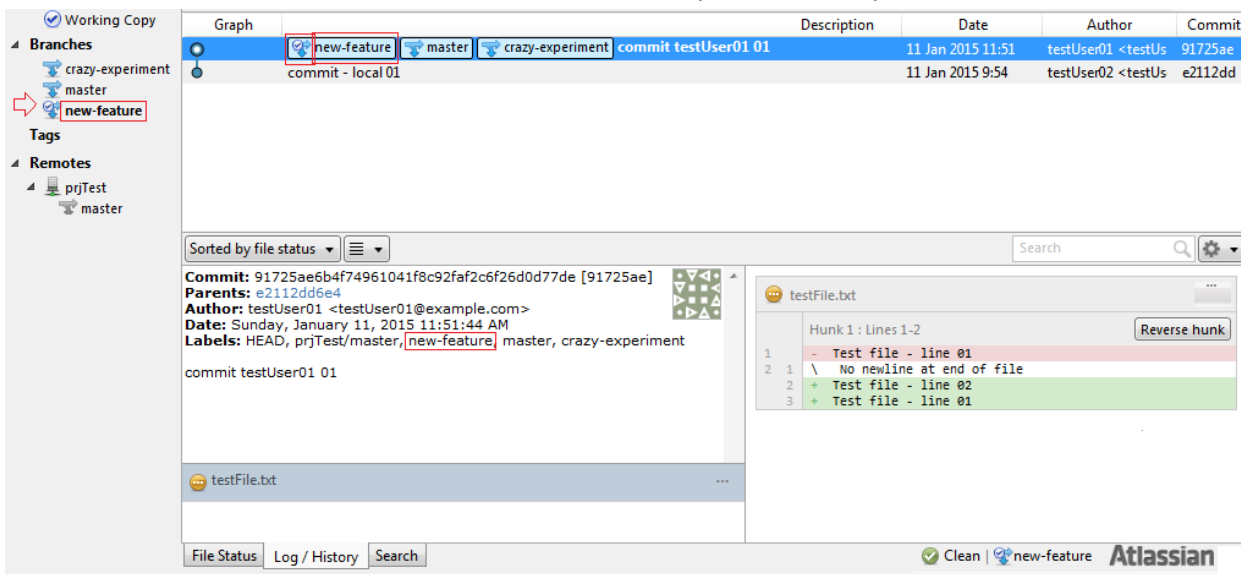
The **git merge** command lets you take the independent lines of development created by git branch and integrate them into a single branch.

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
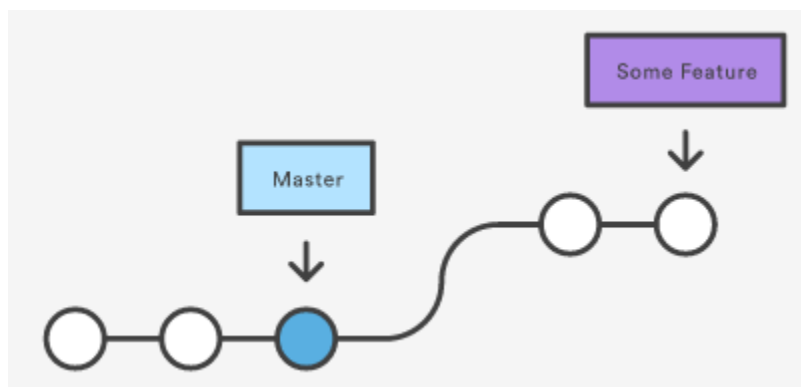
After merging
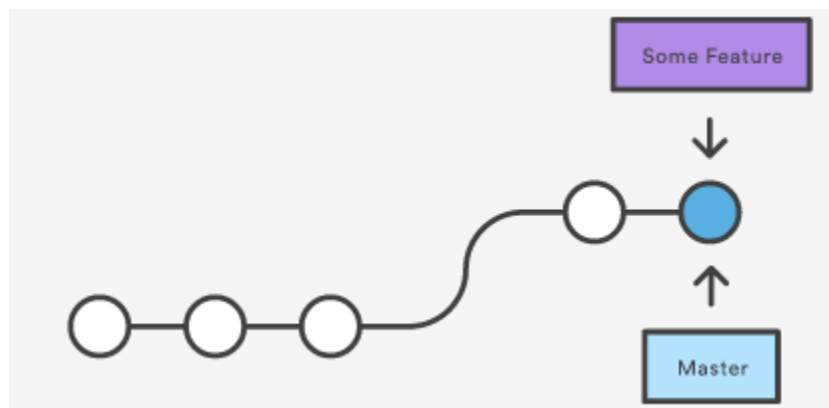
3-Way-Merge

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
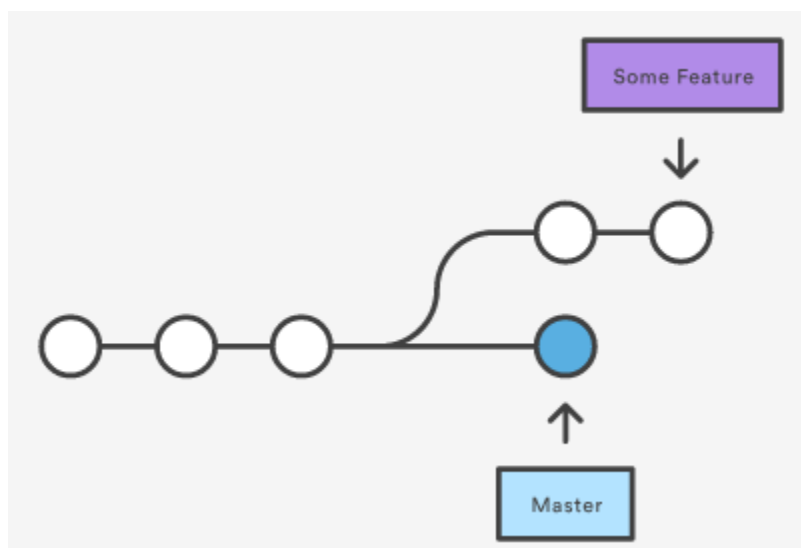
After merging
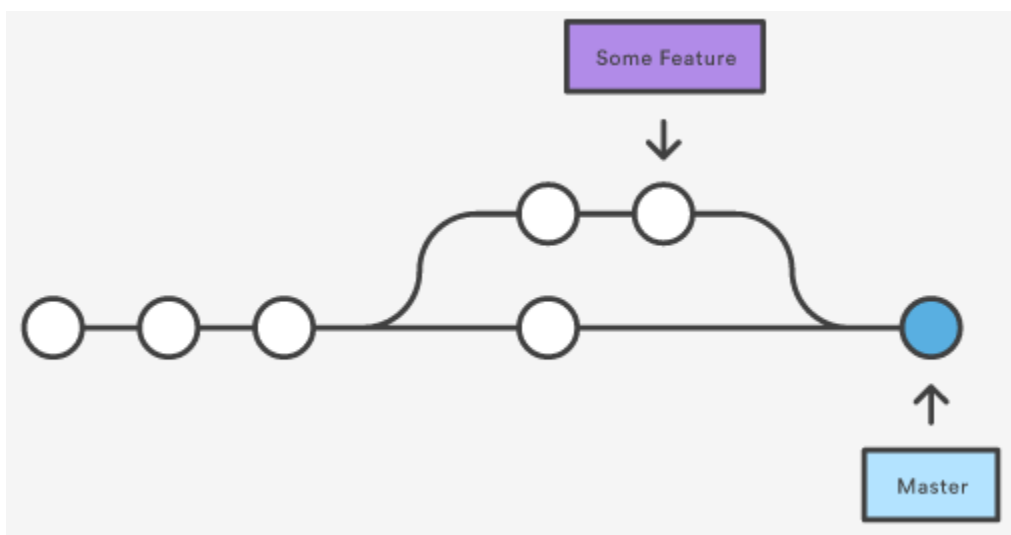
Fast-Forward Merge

Steps

Start a new feature

1. Click **Checkout**

   Click **Checkout New Branch** tab
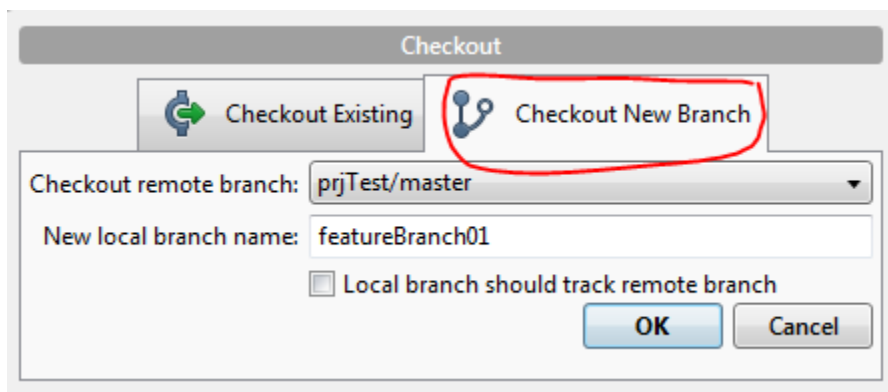
   Give a suitable name of the branch (e.g. **featureBranch01**)
   Click **OK**

2. Newly created branch looks like below :
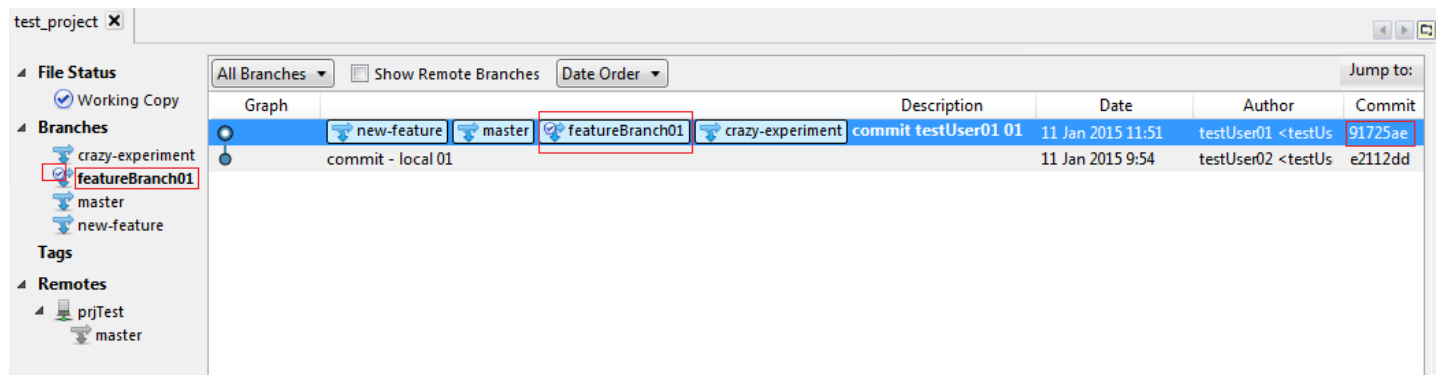
3. Create a file
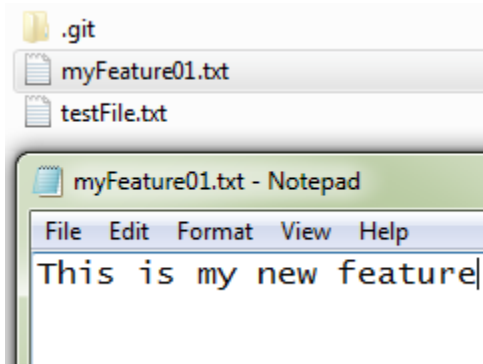
4. Newly created file will be in unstaged area
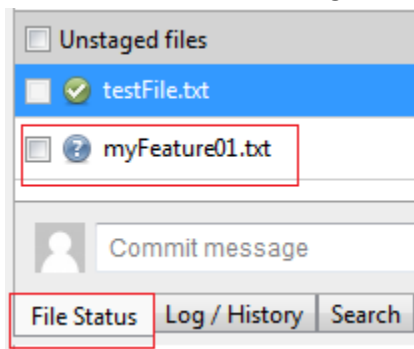
5. Select the file and click **Add**   for adding
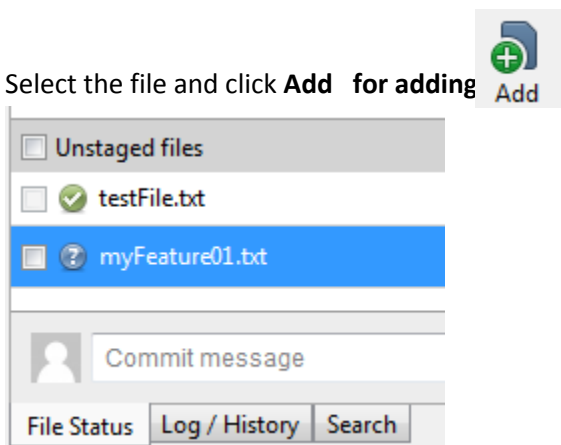
6. File will be added to the staging area
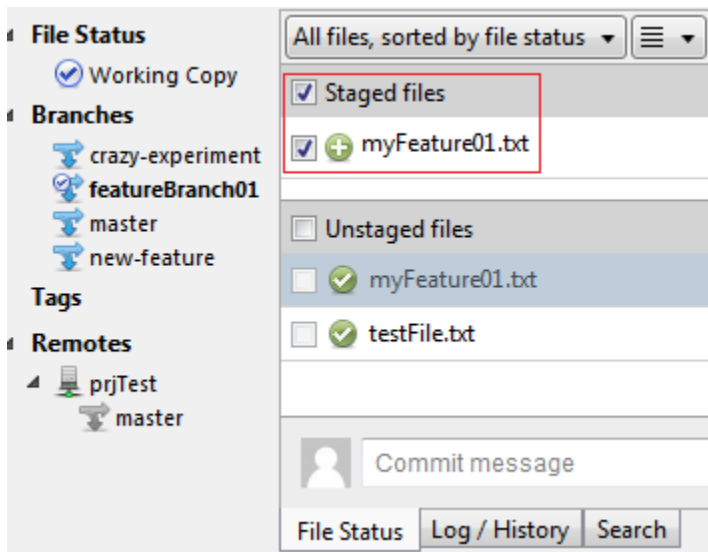
7. Click on **Commit**
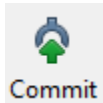
8. Write comment into the comment box
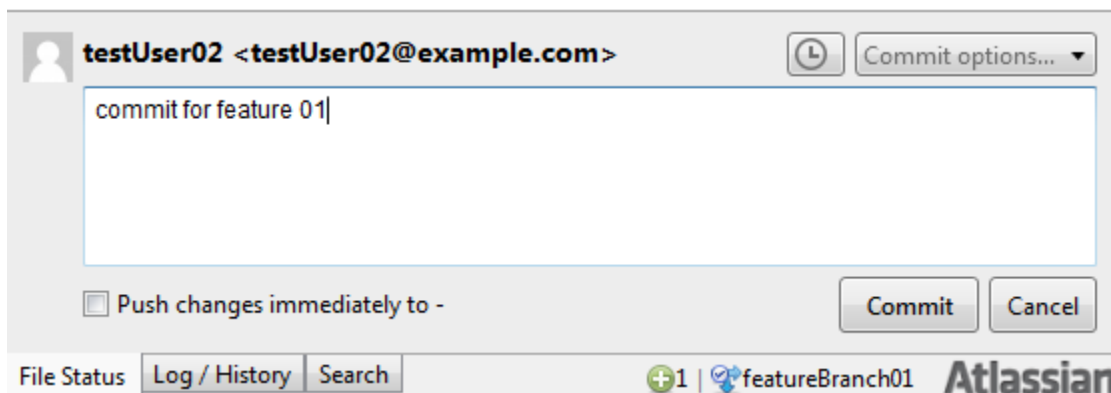
Click **Commit**

9. Log/History looks something given below
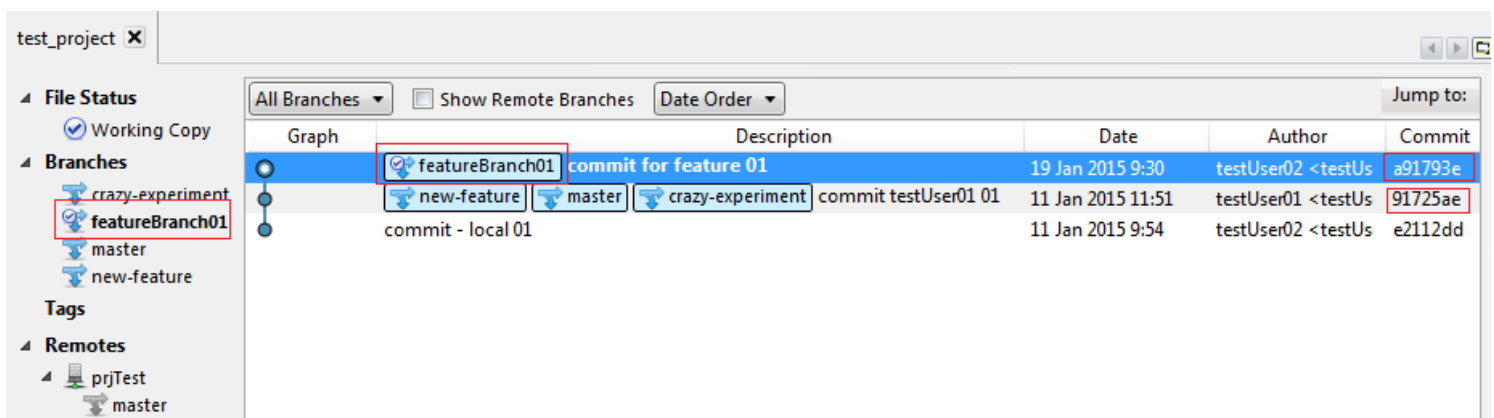
10. Modify some other file
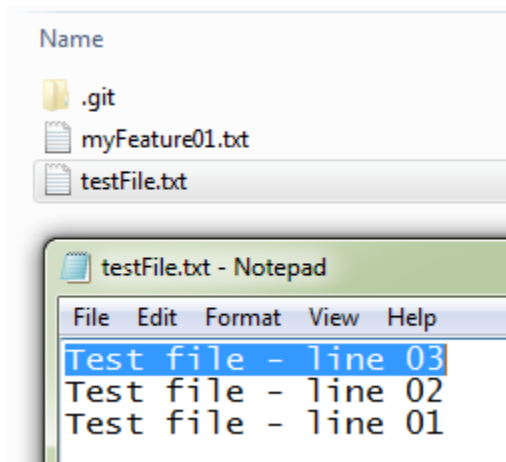
11. Add the file and commit it as described before

12. Log / History looks something like :
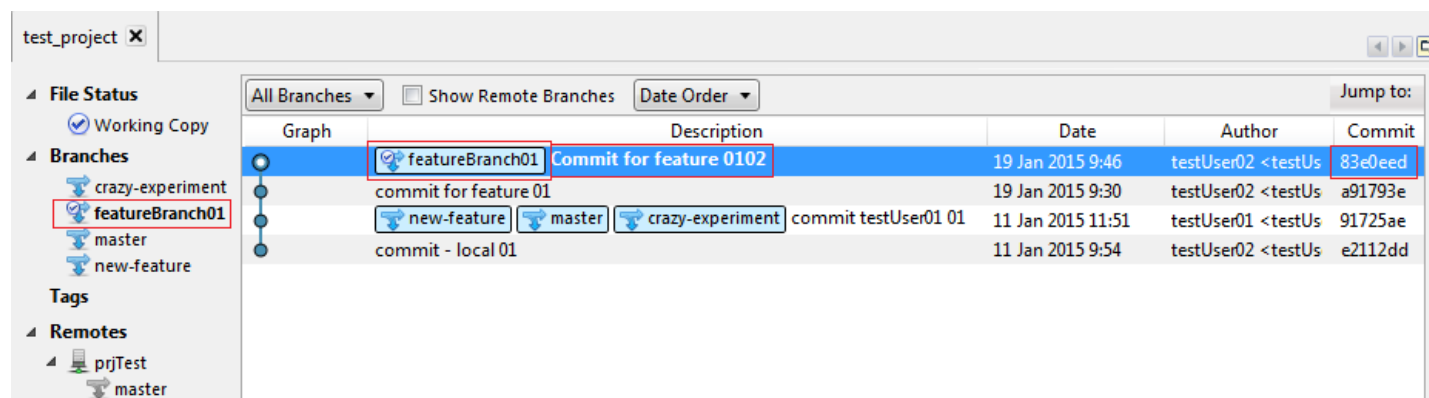
Merge in the new-feature branch

13. Checkout master

Select **Master** from Branches
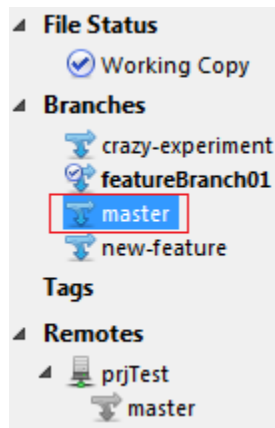
Right click after selecting master
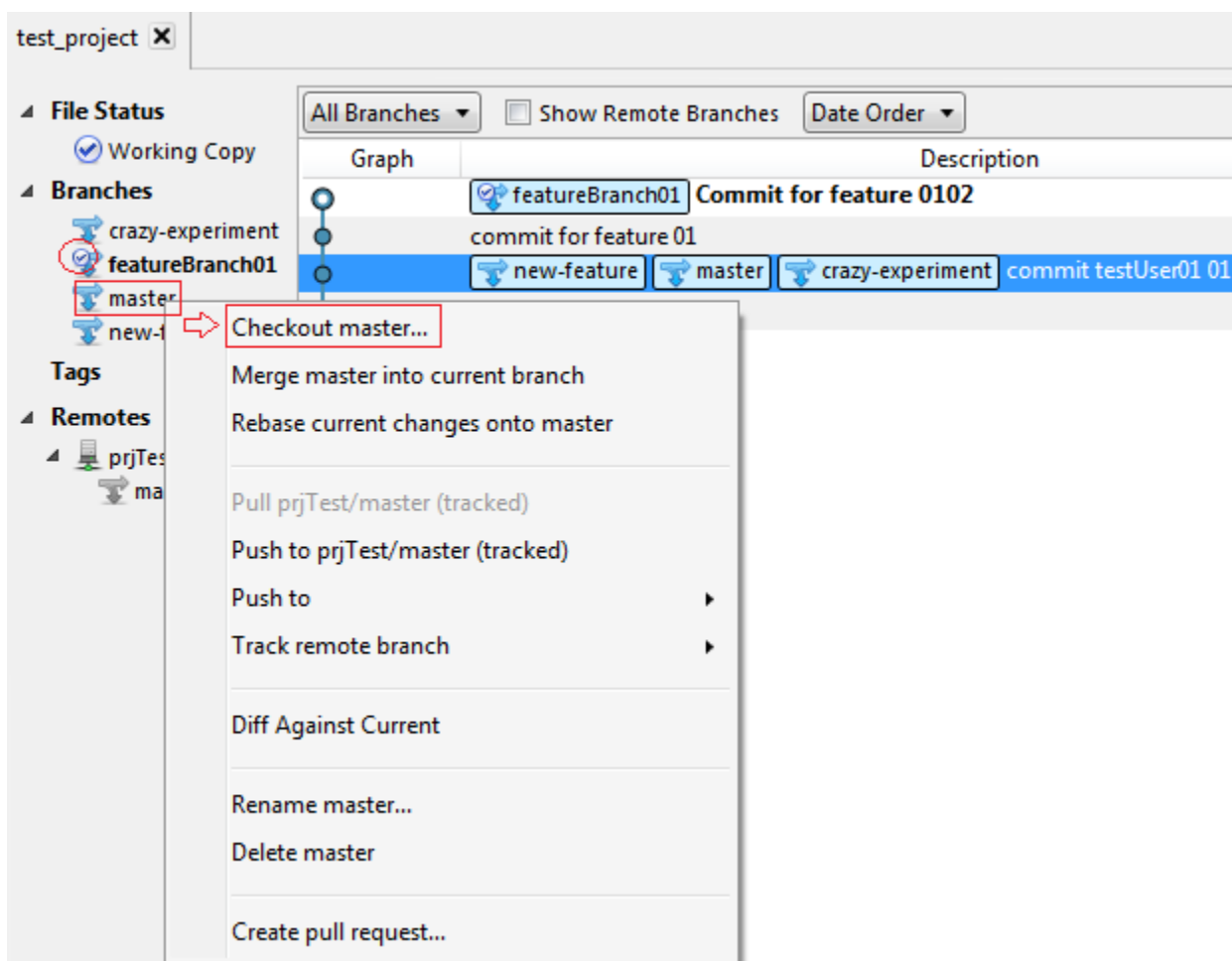
Click **Checkout master**

Branch view are shown below (mind the little arrow – shifted to master)
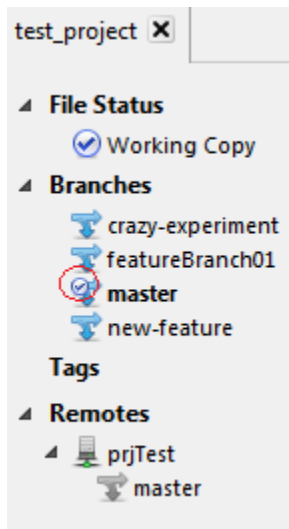
14. Select the **branch** you want to merge with master
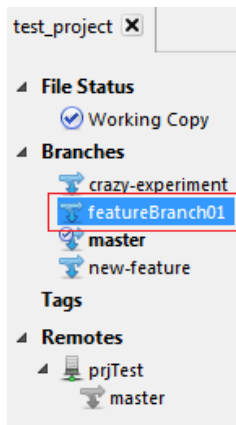
15. Right Click on selected branch
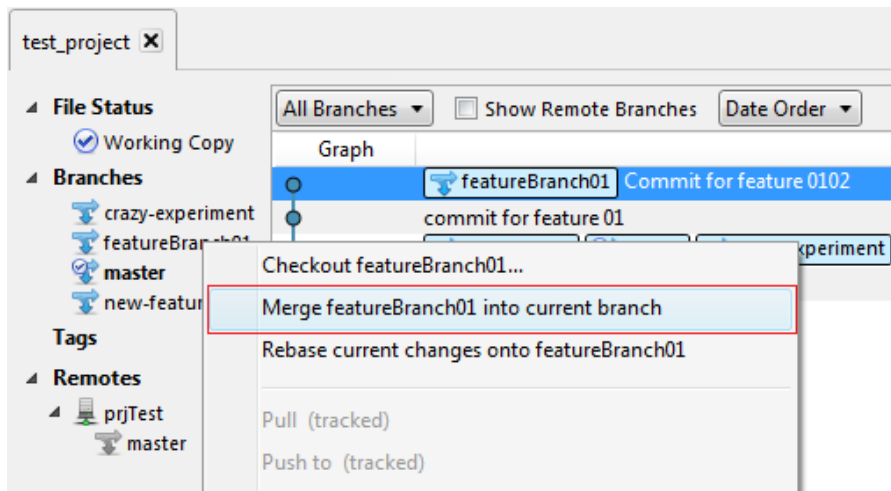
16. Click on the second option from the pull down menu i.e. **Merge <branch name> into current branch**

17. Click **OK** to the **Confirm Merge** dialogue box
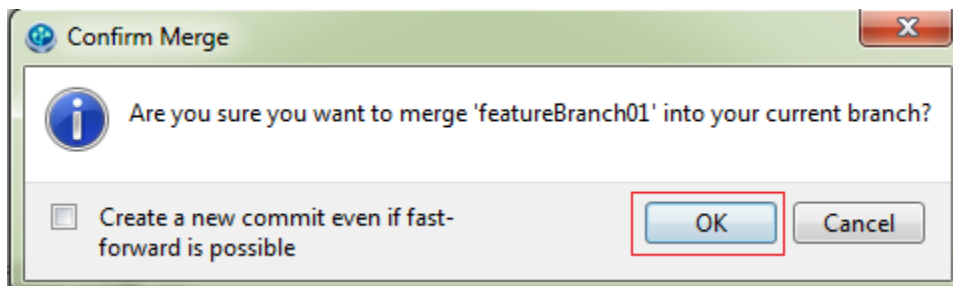
18. Log / History view will be something as shown below :
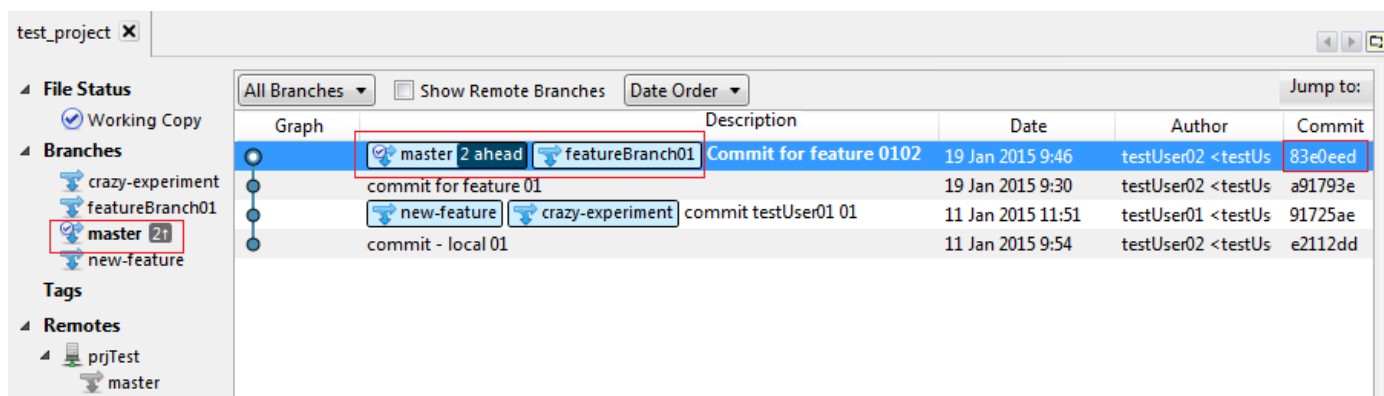
19. Now select the newly created branch (i.e. **featureBranch01**)
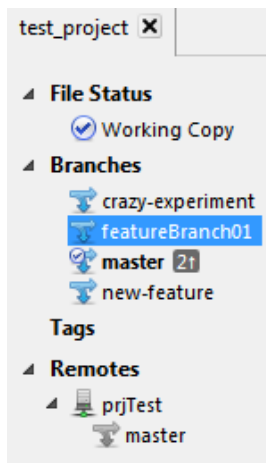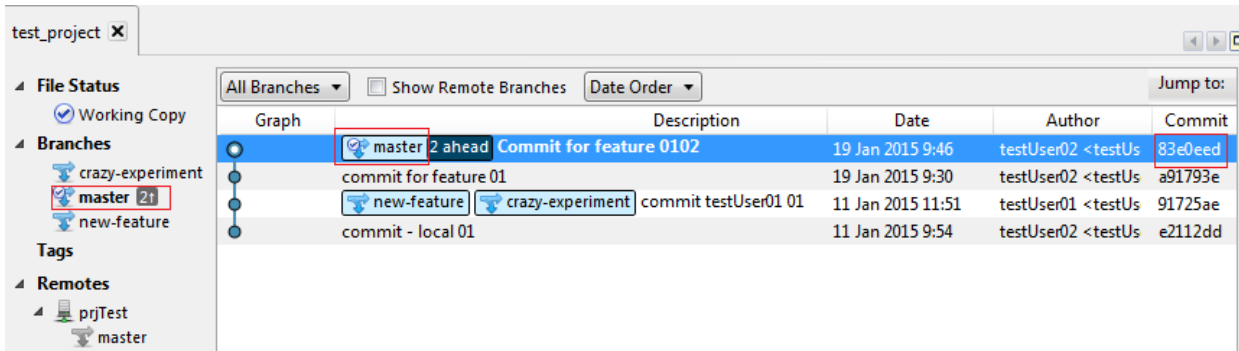
20. Right click and from pull down menu click on **Delete <branch name i.e. featureBranch01>**
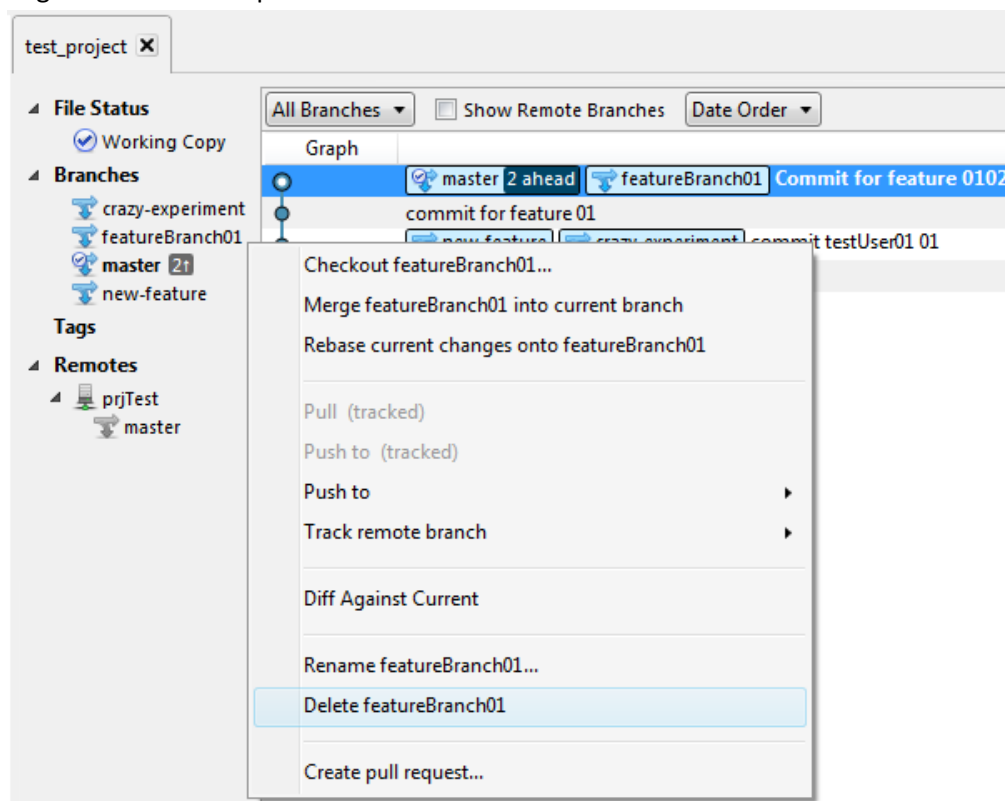
21. Now click **OK** to confirm the delete

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

3-Way Merge

Steps

<u>Start new feature</u>

1. Create a new branch as discussed earlier (e.g. featureBranch02) using checkout
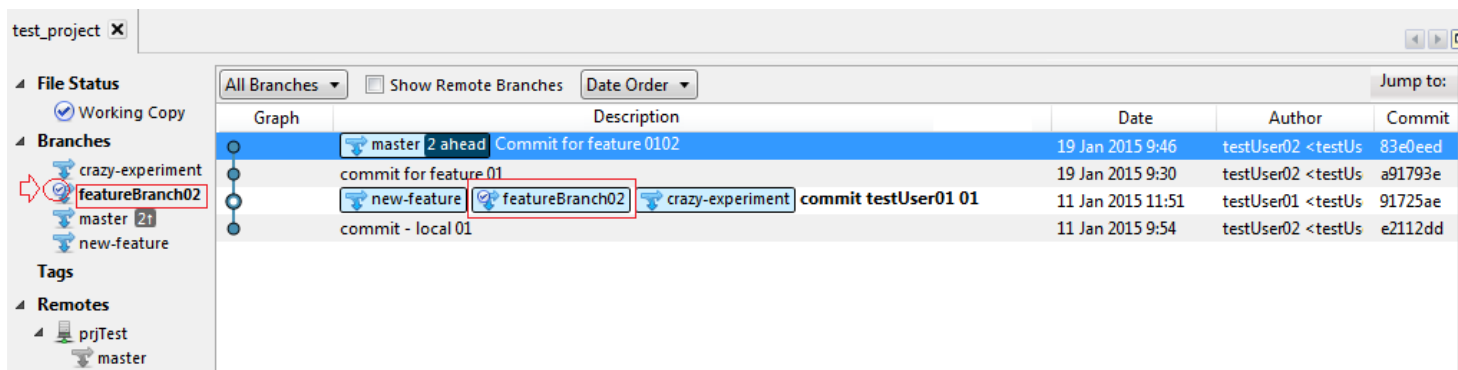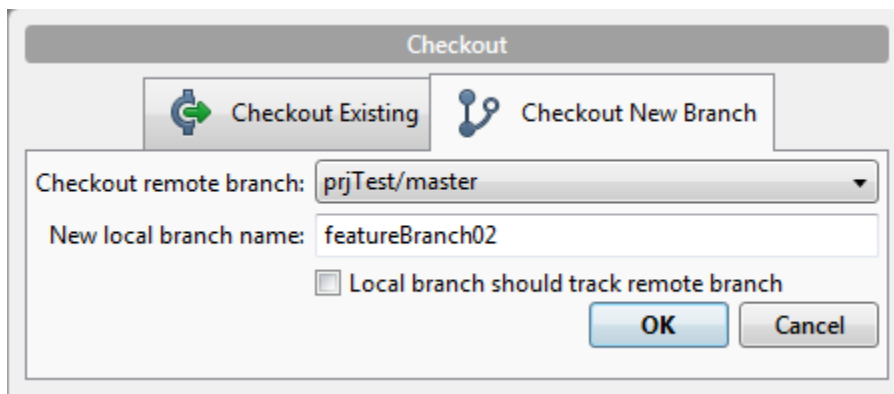
2. Create a new file (e.g. myFeature02.txt)
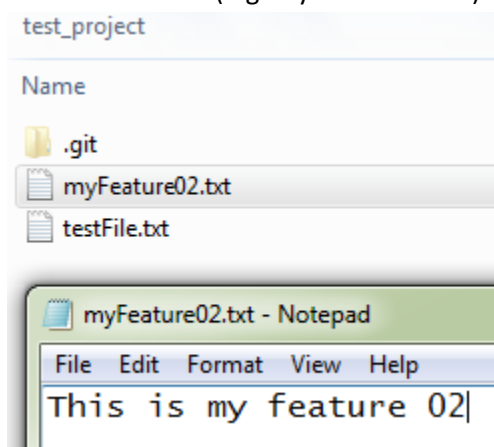
3. File will be Unstaged area
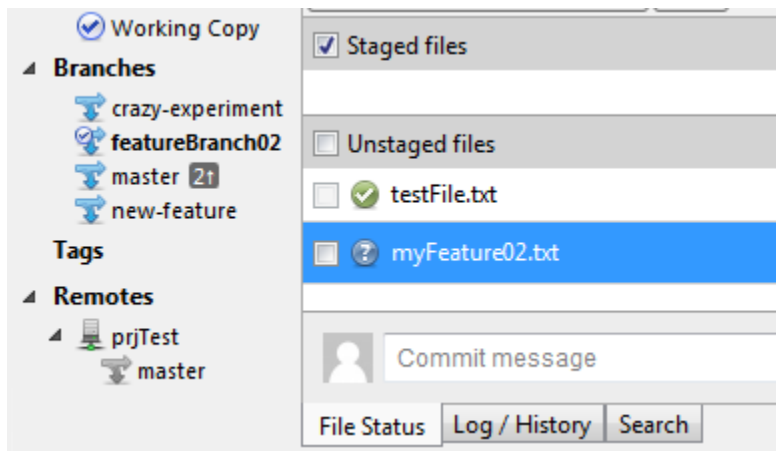
4. Add the file to staged area using **Add**

5. Commit the file using

6. Log / History view

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

8. Add the file using Add

9. Commit the changes

After commit the Log/History view

Merge in the new-feature branch

10. Point to **master**
    a. Select **Master**
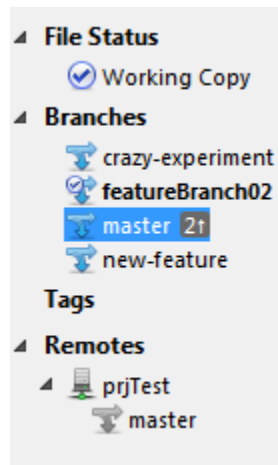
    b. Right Click; and Click **Check Out Master** from the pull down menu

c. Log / History view

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

12. Add the changes using **Add**
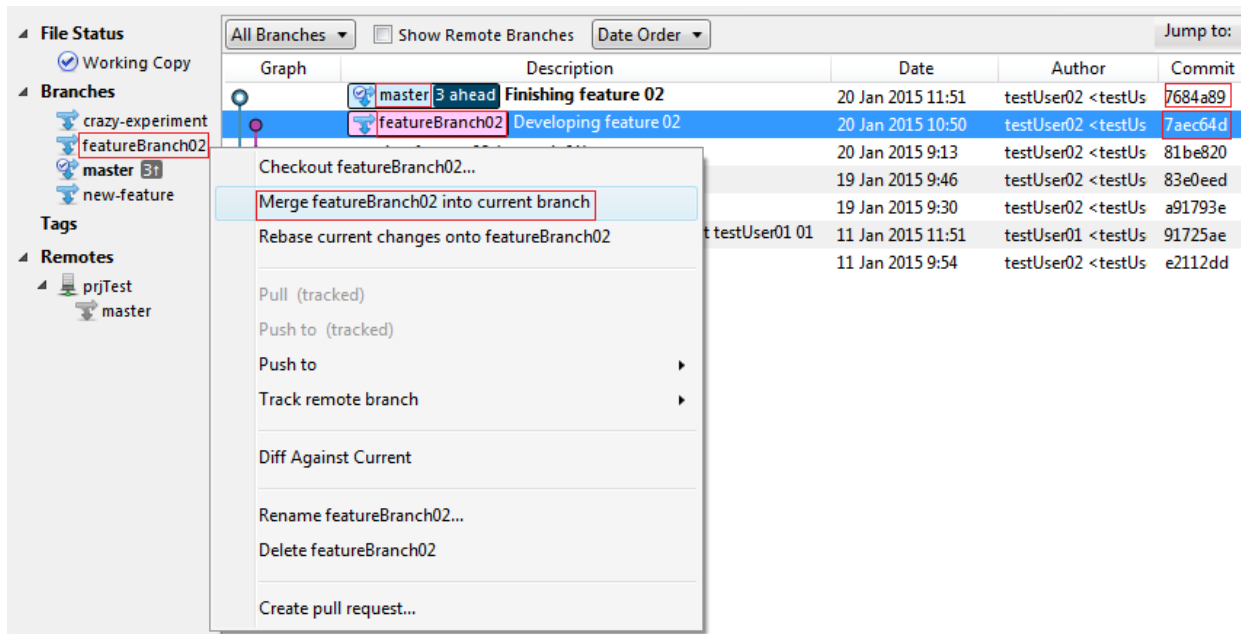13. Commit the master using **commit**
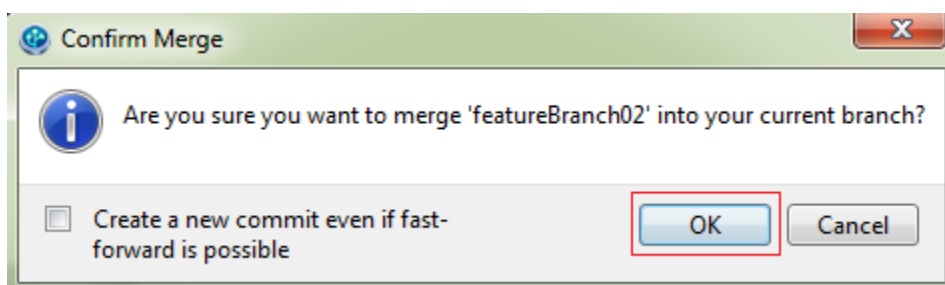

14. Log / History status

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)

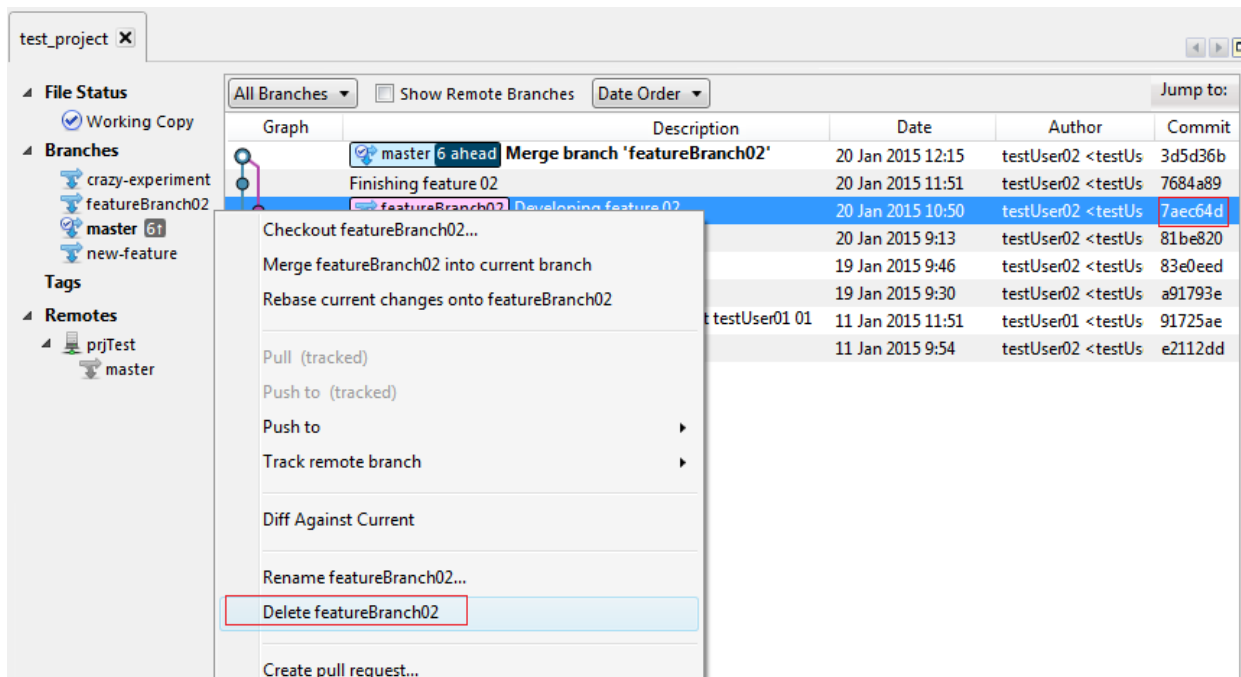16. Click on **OK** on Confirm Message dialogue

17. Log/History View

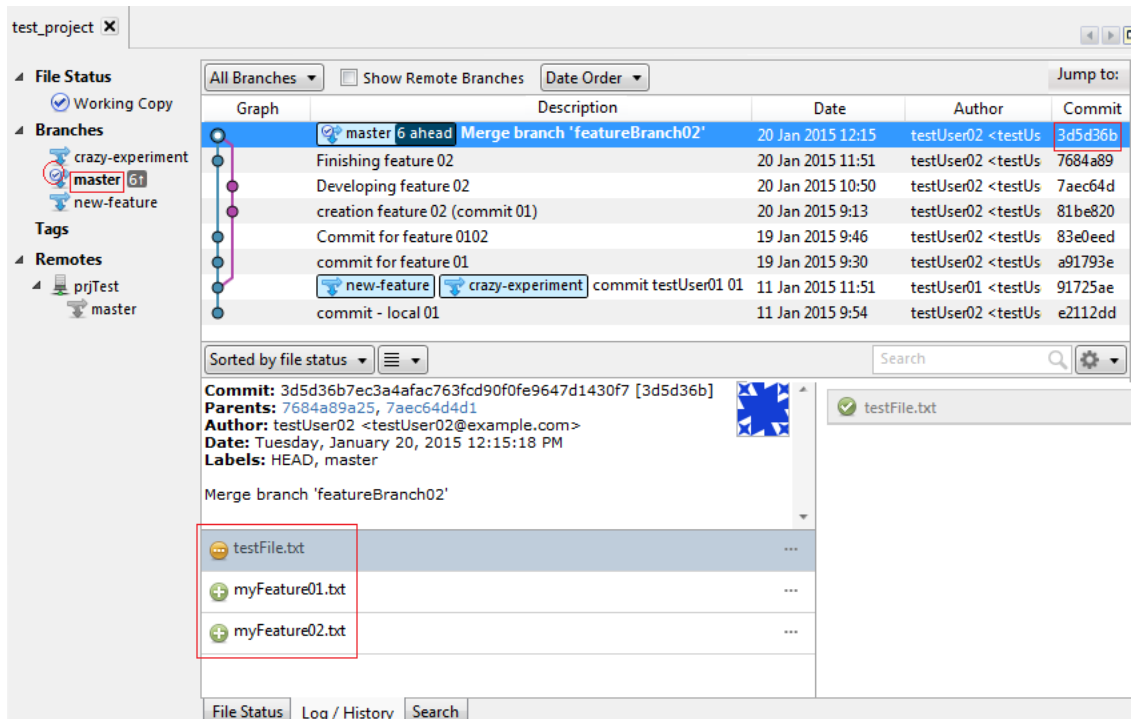18. You might delete the branch you created earlier (e.g. featureBranch02)
19. Select the branch to be deleted (e.g. featureBranch02)

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/