Git Basics

**# Yum install git** - Installation of GIT in Linux / Amazon/redhat

**Git – Operations & Commands**

Some of the basic operations in Git are:

---- GIT Configure Commands -------- >

**1.# git --version** --> version check

**2. # git init --> User Initialize**

**this will create .git the hidden folder , the Local repository of Git system -[.git the local repository]**

ex: /root/git/.git/ [/root/git/ - working diretory] -[.git/ local repository]

**-w/o initiate except version command nothing will work in git**

**-each folder in git need to initiate.**

[root@ip-172-31-2-120 senthil]# mkdir newgit

[root@ip-172-31-2-120 senthil]# cd newgit

[root@ip-172-31-2-120 newgit]# ls -lrta

total 0

drwxr-xr-x 6 root root 56 Jan 27 05:46 ..

drwxr-xr-x 2 root root 6 Jan 27 05:46 .

[root@ip-172-31-2-120 newgit**]# git init**

**Initialized empty Git repository in /root/senthil/newgit/.git/**

[root@ip-172-31-2-120 newgit]# ls -ltra

total 0

drwxr-xr-x 6 root root 56 Jan 27 05:46 ..

drwxr-xr-x 3 root root 18 Jan 27 05:46 .

drwxr-xr-x 7 root root 119 Jan 27 05:46 .git

[root@ip-172-31-2-120 newgit]# **cd .git**

[root@ip-172-31-2-120 .git]# ls -ltra

total 12

-rw-r--r-- 1 root root 73 Jan 27 05:46 description

drwxr-xr-x 2 root root 6 Jan 27 05:46 branches

drwxr-xr-x 3 root root 18 Jan 27 05:46 ..

drwxr-xr-x 2 root root 301 Jan 27 05:46 hooks

drwxr-xr-x 4 root root 31 Jan 27 05:46 refs

drwxr-xr-x 4 root root 30 Jan 27 05:46 objects

drwxr-xr-x 2 root root 21 Jan 27 05:46 info

-rw-r--r-- 1 root root 23 Jan 27 05:46 HEAD

-rw-r--r-- 1 root root 92 Jan 27 05:46 config

drwxr-xr-x 7 root root 119 Jan 27 05:46 .

[root@ip-172-31-2-120 .git]#

**Diff between Working Directory , Local repository?**

**Working Directory – The Developer working environment ex:/root/senthil/git**

**# git init --> User Initialize made one hidden local repository named as .git**

**this will create .git the hidden folder , the Local repository of Git system -[.git the local repository]**

**ex: /root/git/.git/ [/root/git/ - working diretory] -[.git/ local repository]**

[root@ip-172-31-2-120 newgit**]# pwd**

/root/senthil/newgit -------------------------> Working Directory/working copy

[root@ip-172-31-2-120 newgit]# cd .git

[root@ip-172-31-2-120 .git]# pwd

/root/senthil/newgit/.git --------------------> local repository

[root@ip-172-31-2-120 .git]#

**3. # Git config ------> Configure Git user details**

**<global is optional>** - all the folders inside the server has accessed by him responsible to that user.

non global - the only folders where he is working is accessed by him only response for him

# git config --global user.name "Senthil"

# git config --global user.email "bharathiSen@gmail.com"

# git config -l [-list config details]

#/--Reset User details-------------------------------------------------------------------------------------------------------------/

git config --global user.name "Your Name"

git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

**# git commit --amend --reset-author.**#/---------------------------------------------------------------------------------------------------------------/

**4.# git log** Log file --L # - detailed

**# git log --oneline** - log shows in short form of single line

5**.# git show** <commit id> - details about the log

---- GIT FLOW Commands -------- >

6**.# git status -s - comaparing two folders**

[root@ip-172-31-2-120 newgit**]# git status -s**

**?? Newtest**

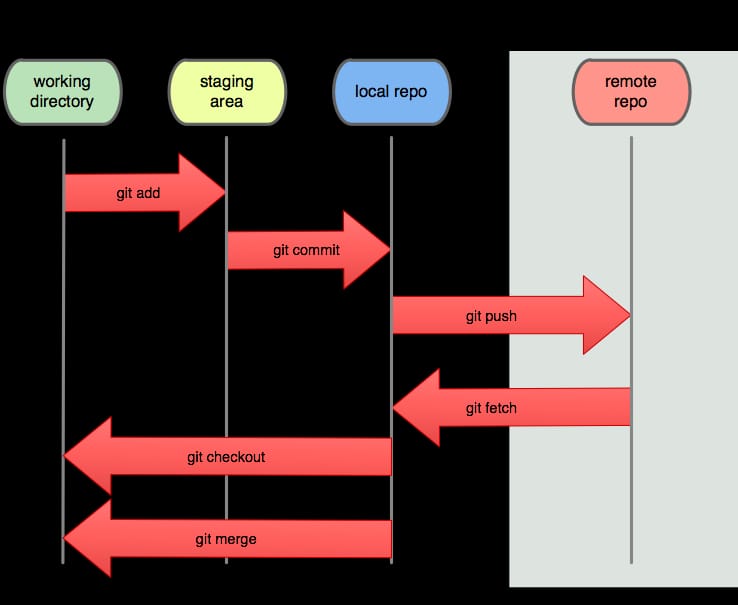
status with ?? - untracked file , not available in local repo

here we use the comparison between git working area & git local repo.

**M**- Modify - <RED> in clolour - the file is not yet staging.

**M**- Modigfy -<Green> in colour - The file is staging. meaned the file is added into stage position.

-------------------------------

-------------------------------

**# git add.**

A- adding into staging area

[root@ip-172-31-2-120 newgit**]# git status -s**

A Newtest

7**.# git status** -for detailed

8. **# git add** Add --> <filename> the developer file will move to staging area

- yet this stage no log hasbeen created -role back is possible

- non confirmed datas are stored to rework.

9**.# git commit -m <message>**Commit --> - moving the file from staging area to local repository

- from this stage there is log has created. if is logging there is no role back

10**. # git show 21d525e**- The Detailed information about the git commit

[root@ip-172-31-2-120 newgit]# **git show 21d525e**

commit 21d525eeaf72d5c0b30b7440bf81d50edd72c9b3 (HEAD -> master)

Author: Senthil <bharathiSen@gmail.com>

Date: Wed Jan 27 17:56:08 2021 +0000

Second Commit

diff --git a/Newtest b/Newtest

index d980039..491d9f1 100644

--- a/Newtest

+++ b/Newtest

@@ -1 +1,2 @@

New test - First Line

+Second Line

------- Remote repository commands(git hub - )------->

**# git remote** - remote repository

10**.# git remote add test** https://github.com/<remote repo address>

**#git remote add <local repo> <remote repo>**

Remote repository added with Local repository for file transferring to Git hub repository

EX:git remote add git2 https://github.com/Senthilkumar1974/Gitnew

GIT2 -->local repository to RR(git hub)

11 . **# Git Remote Remove** --- > to remove the git remote

[root@ip-172-31-2-120 newgit**]# git remote**

gitnew

gittynew

mygit

[root@ip-172-31-2-120 newgit]**# git remove** gittynew

git: 'remove' is not a git command. See **'git --help'**.

The most similar command is

remote

[root@ip-172-31-2-120 newgit**]# git remote rm** gittynew

[root@ip-172-31-2-120 newgit**]# git remote**

gitnew

mygit

[root@ip-172-31-2-120 newgit**]# git remote rm** mygit

[root@ip-172-31-2-120 newgit]#

**#** git rm --cached Newtest

git rm is used to remove a file from a Git repository. It is a convenience method that combines the effect of the default shell rm command with git add .

------------------------------------------------Rename ------------------------

**# git remote rename gittest gitremote**

**---------------------------------------------------------------------------------**

**12. #** **git push git2 master**

ex :***GIT2 -->local repository to git branch(git hub-->master)***

# git push gittynew master

Username for 'https://github.com': Senthilkumar1974

Password for 'https://Senthilkumar1974@github.com':

Enumerating objects: 6, done.

Counting objects: 100% (6/6), done.

Compressing objects: 100% (2/2), done.

Writing objects: 100% (6/6), 469 bytes | 469.00 KiB/s, done.

Total 6 (delta 0), reused 0 (delta 0)

To https://github.com/Senthilkumar1974/mygit

\* [new branch] master -> master

**# git push -u gitnew001 master –**

* git push -u origin master is used for pushing local content to GitHub.
* In the code, the origin is your default remote repository name and '-u' flag is upstream, which is equivalent to '-set-upstream.' and the master is the branch, name.upstream is the repository that we have cloned the project.
* *Fill in your GitHub username and password.*
* *Technically, the -u flag adds a tracking reference to the upstream server you're pushing to.*
* *What is necessary here is that this allows you to do a git pull without supplying any more arguments.*
* *For example, once you are doing a git push -u origin master, you'll be able to later call git pull and git will understand that you indeed meant git pull origin master.*
* *Otherwise, you'd have to type in the whole command.*

-------------------------------------------------

**13 # git fetch** git2 master

**# git fetch gitnew** - git fetch is the command to download from remote repository. but the latest changes happending the remote was not updated, means the updated / pushed data available.

[root@ip-172-31-3-77 git**]# git fetch gitnew**

remote: Enumerating objects: 5, done.

remote: Counting objects: 100% (5/5), done.

remote: Compressing objects: 100% (2/2), done.

remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0

Unpacking objects: 100% (3/3), done.

From https://github.com/Senthilkumar1974/gitclass

057d19e..6a11c25 master -> gitnew/master

**12. # Git Pull**

git pull is also the command to download from remote repositry with the latest changes made in the remote itself updated.

**# git pull gittynew master**

<remote> <branch>

[root@ip-172-31-2-120 newgit**]# git pull gittynew master**

remote: Enumerating objects: 5, done.

remote: Counting objects: 100% (5/5), done.

remote: Compressing objects: 100% (2/2), done.

remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0

Unpacking objects: 100% (3/3), done.

From https://github.com/Senthilkumar1974/mygit

\* branch master -> FETCH\_HEAD

21d525e..4da925f master -> gittynew/master

Updating 21d525e..4da925f

Fast-forward

Newtest | 1 +

1 file changed, 1 insertion(+)

----- new update

[root@ip-172-31-3-77 git]# git pull gitnew master

From https://github.com/Senthilkumar1974/gitclass

\* branch master -> FETCH\_HEAD

Updating 057d19e..6a11c25

Fast-forward

gittest | 1 +

1 file changed, 1 insertion(+)

[root@ip-172-31-3-77 git]# cat gittest

First Line --- Git Add

second line

-----

Third Line ---- git modify

------

git hub addition - fetch/pull

git hub addition - fetch/pull 2nd test updation

[root@ip-172-31-3-77 git]#

-------------------- git Clone ---------------------

**13.# git clone** is how you get a local copy of an existing repository to work on. ... git pull (or git fetch + git merge ) is how you update that local copy with new commits from the remote repository.

[root@ip-172-31-3-77 gitclass**]# git clone** https://github.com/Senthilkumar1974/gitclass.git

Cloning into 'gitclass'...

remote: Enumerating objects: 12, done.

remote: Counting objects: 100% (12/12), done.

remote: Compressing objects: 100% (7/7), done.

remote: Total 12 (delta 3), reused 5 (delta 1), pack-reused 0

Unpacking objects: 100% (12/12), done.

[root@ip-172-31-3-77 gitclass]# ls -la

total 4

drwxr-xr-x 4 root root 49 Jan 28 08:41 .

drwxr-xr-x 4 root root 49 Jan 28 08:40 ..

drwxr-xr-x 8 root root 163 Jan 28 08:40 .git

drwxr-xr-x 3 root root 33 Jan 28 08:41 gitclass

-rw-r--r-- 1 root root 157 Jan 28 08:40 gittest

**Branching & Merging**

**What is git branch ? when it will create in git operation ?**

Branches- git providing a **virtul environment** to work is called branch. Branches are the sub components of the main component, master. branching is for backup for master

*Git branch will create only* ***after the first commit*** *initiates*, a log file is created for that action and a branch is opened / created for that user.

Except the Master branch, other created sub branches are called ” **Feature branches**” common terminalogy

**Window Time :** Time to allot / fixed to finish a Task./ Downtime activity

/-----------------------------------------------------------------------------------------

**13 # git branch - branch list**

13**.# git branch** \*\*\* - branch details --- \*\*\*refer below / active branch where we stand

# git branch dev

//----

**# git branch**

dev

\* master - The \* denotes the active branch where we stand / resides now and the test with green & starred

**14. # git checkout <feature branch>** - switched to branch feature branch <dev>

Switched to feature branch - the feature branch virually created and switched over. the files what are available in master in now available in feature branch also.

**# git branch**

\* dev - now virtualy we standing on feature branch

Master

***How to find which branch you are ?***

***Using git branch command to find the branch , which is stared and colored as green. Want to work to other feature branch the git checkout command will use to change or move to the required feature brnach***

**Git Merging**

[root@ip-172-31-3-77 git**]# git log --oneline**

3a065a5 (HEAD -> master) fifth commit for -log3

0426854 fourth commit for -log2

ff95290 Third commit for -log1

6a11c25 (gitnew/master) Update gittest

057d19e Update gittest

b440e6a Second Commit - First modification

0fd24d2 Git First Commit

**Rule 1 ### During Merge the branch location should be always in master**

**Rule 2 ### Check the log before merge. To avoid conflicts**

[root@ip-172-31-3-77 git**]# git branch**

dev

\* master

**Scenario #1 – smooth merging.**

**15# git merge <feature branch>**

[root@ip-172-31-3-77 git**]# git merge dev**

Updating 3a065a5..7af8c48

Fast-forward

gittest | 3 ++-

1 file changed, 2 insertions(+), 1 deletion(-)

[root@ip-172-31-3-77 git]#

----------------------------------------------------------

[root@ip-172-31-3-77 git**]# git log** --oneline

7af8c48 (**HEAD -> master, dev**) Fourth Commit - branching/merging

3a065a5 fifth commit for -log3

0426854 fourth commit for -log2

ff95290 Third commit for -log1

6a11c25 (**gitnew/master**) Update gittest

057d19e Update gittest

b440e6a Second Commit - First modification

0fd24d2 Git First Commit

[root@ip-172-31-3-77 git]#

\*\*\*\*\*\* ----------------------------------

**Scenario #2 – conflict merging.**

# mkdir merge

#cd merge

# vi test

#ls -ltra

#git init – initiate folder to local epository

# ls -ltra

# clear

# git add .

# git status -**s**

# git commit -m "First commit - Merge conflict"

# git status -s

# clear

# history

**[root@ip-172-31-3-77 merge]#**

**[**root@ip-172-31-3-77 merge**]# git branch**

\* master

[root@ip-172-31-3-77 merge]# vi test --- edit the file

[root@ip-172-31-3-77 merge**]# git commit -a -m "second commit"**

**Is possible to commit the file without add ?**

**Ans : with out add there is not possible to commit a file, but commit with add is possible by using some parameter a. this commit with add is only possible working with modified file only. It will not work with files in untrack stage.**   
# **git commit -a -m "second commit**" / # **git commit -am "second commit**"

/ [master e0bcc2f] second commit

1 file changed, 2 insertions(+), 1 deletion(-)

[root@ip-172-31-3-77 merge]#

[root@ip-172-31-3-77 merge]# git branch

\* master

[root@ip-172-31-3-77 merge]# git branch dev -- New future branch created

[root@ip-172-31-3-77 merge]# git branch

dev

\* master

[root@ip-172-31-3-77 merge]# git checkout dev – move to feature branch

Switched to branch 'dev'

[root@ip-172-31-3-77 merge]# git branch

\* dev

master

[root@ip-172-31-3-77 merge]# vi test – edit the available file

[root@ip-172-31-3-77 merge]# git commit -am "Third commit" – commit with add file.

[dev d8ef7fe] Third commit

1 file changed, 1 insertion(+)

**Conflict creation**

*The another user directly edit the file and commit it into the master branch itself*

*Now the featured branch user try to add his file in the master faced conflict*

[root@ip-172-31-3-77 merge**]# git checkout master**

Switched to branch 'master'

[root@ip-172-31-3-77 merge**]# git branch**

dev

\* master

[root@ip-172-31-3-77 merge]# clear

[root@ip-172-31-3-77 merge]# ls -ltr

total 4

-rw-r--r-- 1 root root 44 Jan 28 20:11 test

[root@ip-172-31-3-77 merge]# vi test

[root@ip-172-31-3-77 merge]# git commit -am "3rd commit"

[master 216c272] 3rd commit

1 file changed, 1 insertion(+)

[root@ip-172-31-3-77 merge]# git log

commit 216c272140bfdf89b8ec8e80995164156cde0d58 (HEAD -> master)

Author: Senthil <bharathiSen@gmail.com>

Date: Thu Jan 28 20:14:18 2021 +0000

3rd commit

commit e0bcc2f47db5a28c85660966b8cb13ccd4b1c6c5

Author: Senthil <bharathiSen@gmail.com>

Date: Thu Jan 28 19:44:10 2021 +0000

second commit

commit 751adfcbdef50fd47ed3e24c37e79788f65e2a77

Author: Senthil <bharathiSen@gmail.com>

Date: Thu Jan 28 19:34:24 2021 +0000

First commit - Merge conflict

[root@ip-172-31-3-77 merge]#

[root@ip-172-31-3-77 merge**]# git merge dev**

**Auto-merging test**

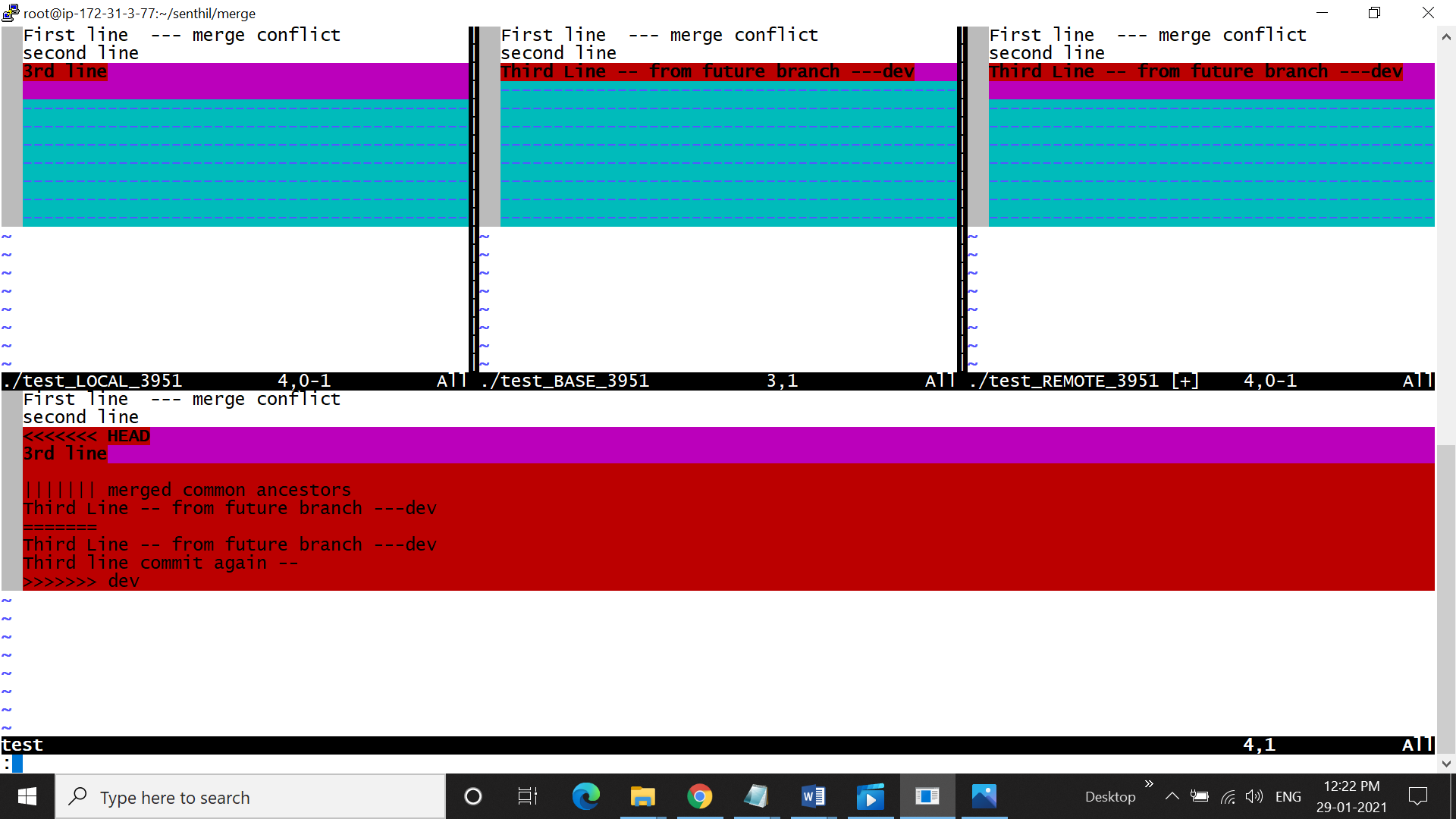
**CONFLICT (content): Merge conflict in test**

Automatic merge failed; fix conflicts and then commit the result.

**Merge tool used to merge the file -**

git config --global merge.tool vimdiff  
git config --global merge.conflictstyle diff3  
git config --global mergetool.prompt false

**#git mergetool**



* Move with in tool - **^W** twice to move next screen
* The mergetool allow to edit
* After finalise the correct code from the merge conflict.
* Come out to tool baseline use **: diffg <LO / RE /BA > -**LO for test local, RE for Remote, BA for base
* **To save files** :wq! / wqa – wq! Save one by one , wqa to save all

[root@ip-172-31-3-77 merge**]# git mergetool**

Merging:

Test  
Normal merge conflict for 'test':

{local}: modified file

{remote}: modified file

4 files to edit

[root@ip-172-31-3-77 merge]# cat test

First line --- merge conflict

second line

Third Line -- from future branch ---dev

**>>>> After merging**  
[root@ip-172-31-3-77 merge]# **git log --oneline**

31a0888 (HEAD -> master) commit 3rd agin

8c97804 3rd commit

216c272 3rd commit

d8ef7fe Third commit

e0bcc2f second commit

751adfc First commit - Merge conflict

[root@ip-172-31-3-77 merge**]# git status -s**

**M**  test – modified and add into staging area

?? test.orig – the conflict file before fix the merge conflict.

#### The Git Mergetool- > after merging files , it will add / move to staging area, so it also act as **STAGINGTOOL.**

**Git is a #Verssion Control System tool  
 #Sourcecode Managemt Tool  
 #Staging Tool – by using Mergetool  
### Git took backup before merge for future reference**

[root@ip-172-31-3-77 merge**]# cat test.orig**

First line --- merge conflict

second line

<<<<<<< HEAD

3rd line

||||||| merged common ancestors

Third Line -- from future branch ---dev

=======

Third Line -- from future branch ---dev

Third line commit again --

>>>>>>> dev

[root@ip-172-31-3-77 merge**]# rm -Rf test.orig**

rm: remove regular file ‘test.orig’? y

[root@ip-172-31-3-77 merge**]# git status -s**

M test

[root@ip-172-31-3-77 merge]#

**After complete the gitmerge operation to fix the git conflict , what will be the next tag.**

**Ans : git merge tool only add the corrected conflict file. So there is one commit is balance in the process.**

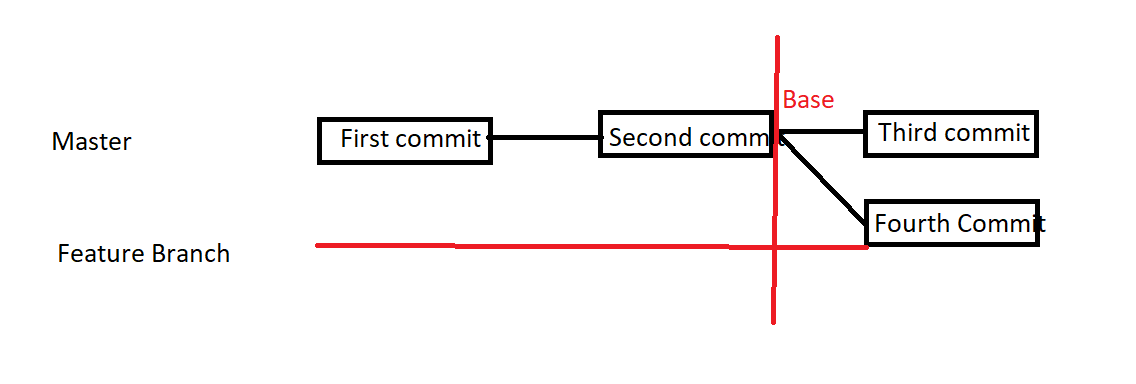
[root@ip-172-31-3-77 merge**]# git commit -m "final commit"**

[master 5129563] final commit

**Scenario #3 – Rebase..**

**Without merge conflict.**

**Rebase – repeated – changing the base**

****

* **Common base for both branches..**

[root@ip-172-31-3-77 senthil]# mkdir rebase

[root@ip-172-31-3-77 senthil]# cd rebase

[root@ip-172-31-3-77 rebase]# git init

Initialized empty Git repository in /root/senthil/rebase/.git/

[root@ip-172-31-3-77 rebase]# vi test – **new text file**

[root@ip-172-31-3-77 rebase]# git add .

[root@ip-172-31-3-77 rebase]# git commit -m "first commit" (Master branch)

[master (root-commit) b5b0115] first commit

1 file changed, 1 insertion(+)

create mode 100644 test

[root@ip-172-31-3-77 rebase]# vi test – **Text file modification**

[root@ip-172-31-3-77 rebase]# git commit -am "second commit"(Master branch)

[master d6145a7] second commit

1 file changed, 2 insertions(+)

[root@ip-172-31-3-77 rebase]# git log

**commit d6145a75a8a8b4132485e4355c413f2e0df2f681 (HEAD -> master)**

Author: Senthil <bharathiSen@gmail.com>

Date: Fri Jan 29 15:42:00 2021 +0000

second commit

commit b5b0115efff44589682c3d97c2d8131a50501671

Author: Senthil <bharathiSen@gmail.com>

Date: Fri Jan 29 15:41:20 2021 +0000

first commit

[root@ip-172-31-3-77 rebase]# git branch

\* master

[root@ip-172-31-3-77 rebase]# git branch dev - feature branch created

[root@ip-172-31-3-77 rebase]# git checkout dev - **Switched to branch 'dev'**

Switched to branch 'dev'

[root@ip-172-31-3-77 rebase**]# git branch**

**\* dev**

master

[root@ip-172-31-3-77 rebase]# git log

**commit d6145a75a8a8b4132485e4355c413f2e0df2f681 (HEAD -> dev, master)**

Author: Senthil <bharathiSen@gmail.com>

Date: Fri Jan 29 15:42:00 2021 +0000

second commit

commit b5b0115efff44589682c3d97c2d8131a50501671

Author: Senthil <bharathiSen@gmail.com>

Date: Fri Jan 29 15:41:20 2021 +0000

first commit

[root@ip-172-31-3-77 rebase]# vi test1 – **another one new text file created**

[root@ip-172-31-3-77 rebase]# ls -la

total 8

drwxr-xr-x 3 root root 43 Jan 29 15:45 .

drwxr-xr-x 6 root root 55 Jan 29 15:40 ..

drwxr-xr-x 8 root root 166 Jan 29 15:42 .git

-rw-r--r-- 1 root root 25 Jan 29 15:41 test ---two files available

-rw-r--r-- 1 root root 12 Jan 29 15:45 test1

[root@ip-172-31-3-77 rebase]# git status -s

?? test1

[root@ip-172-31-3-77 rebase]# git add .

[root@ip-172-31-3-77 rebase]# git status -s

A test1

[root@ip-172-31-3-77 rebase]# git commit -m "**fourth commit**"(Feature branch /dev)

[dev cbccc4e] fourth commit

1 file changed, 1 insertion(+)

create mode 100644 test1

[root@ip-172-31-3-77 rebase]# git log

**commit cbccc4e41c09f03f28f15b5908fde48f5cb76339 (HEAD -> dev)**

Author: Senthil <bharathiSen@gmail.com>

Date: Fri Jan 29 15:53:22 2021 +0000

fourth commit

commit d6145a75a8a8b4132485e4355c413f2e0df2f681 (master)

Author: Senthil <bharathiSen@gmail.com>

Date: Fri Jan 29 15:42:00 2021 +0000

second commit

commit b5b0115efff44589682c3d97c2d8131a50501671

Author: Senthil <bharathiSen@gmail.com>

Date: Fri Jan 29 15:41:20 2021 +0000

first commit

[root@ip-172-31-3-77 rebase]# git branch

git: 'bracnh' is not a git command. See 'git --help'.

The most similar command is

branch

[root@ip-172-31-3-77 rebase]# git brach

git: 'brach' is not a git command. See 'git --help'.

The most similar command is

Branch

[root@ip-172-31-3-77 rebase]# git branch

\* dev

master

[root@ip-172-31-3-77 rebase]# **git checkout master**

**Switched to branch 'master'**

[root@ip-172-31-3-77 rebase**]# git log**

**commit d6145a75a8a8b4132485e4355c413f2e0df2f681 (HEAD -> master)**

Author: Senthil <bharathiSen@gmail.com>

Date: Fri Jan 29 15:42:00 2021 +0000

second commit

commit b5b0115efff44589682c3d97c2d8131a50501671

Author: Senthil <bharathiSen@gmail.com>

Date: Fri Jan 29 15:41:20 2021 +0000

first commit

[root@ip-172-31-3-77 rebase]# vi test

[root@ip-172-31-3-77 rebase]# git add .

[root@ip-172-31-3-77 rebase]# git commit -m "**third line"(**Master branch)

[master f6278e5] third line

1 file changed, 1 insertion(+), 1 deletion(-)

[root@ip-172-31-3-77 rebase]# git log --oneline

f6278e5 (HEAD -> master) third line

d6145a7 second commit

b5b0115 first commit

[root@ip-172-31-3-77 rebase]# git merge dev

hint: Waiting for your editor to close the file...

[3]+ Stopped git merge dev

[root@ip-172-31-3-77 rebase]#

[root@ip-172-31-3-77 rebase]# git log --oneline

33baee7 (HEAD -> master) merge

f6278e5 third line

cbccc4e (dev) fourth commit

d6145a7 second commit

b5b0115 first commit

[root@ip-172-31-3-77 rebase]#

**Rules :Before Merge -  
#1. Standing in Master branch only to merge-check branch  
#2.Master branch – Log should be verify of conflicts.**

-

**Now** the situation :

**#git log --oneline**

8980b4e (HEAD -> master) -third commit

33baee7 merge

f6278e5 third line

cbccc4e (dev) fourth commit

d6145a7 second commit

b5b0115 first commit

**to remove unwanted commits from git** **gi <sha1-commit-id>**

[root@ip-172-31-3-77 rebase**]# git reset --hard f6278e5**

HEAD is now at f6278e5 third line

[root@ip-172-31-3-77 rebase]# git log --oneline

f6278e5 (HEAD -> master) third line

d6145a7 second commit

b5b0115 first commit

**unwanted commits removed…**

**using Rebase for merging files and commits**

**# Rule : using rebase , the branch should be always featured only.**

[root@ip-172-31-3-77 rebase**]# git rebase master**

**First, rewinding head to replay your work on top of it...**

Applying: fourth commit

[root@ip-172-31-3-77 rebase]# git log --oneline

9ba3293 (HEAD -> dev) fourth commit – commit from featured branch

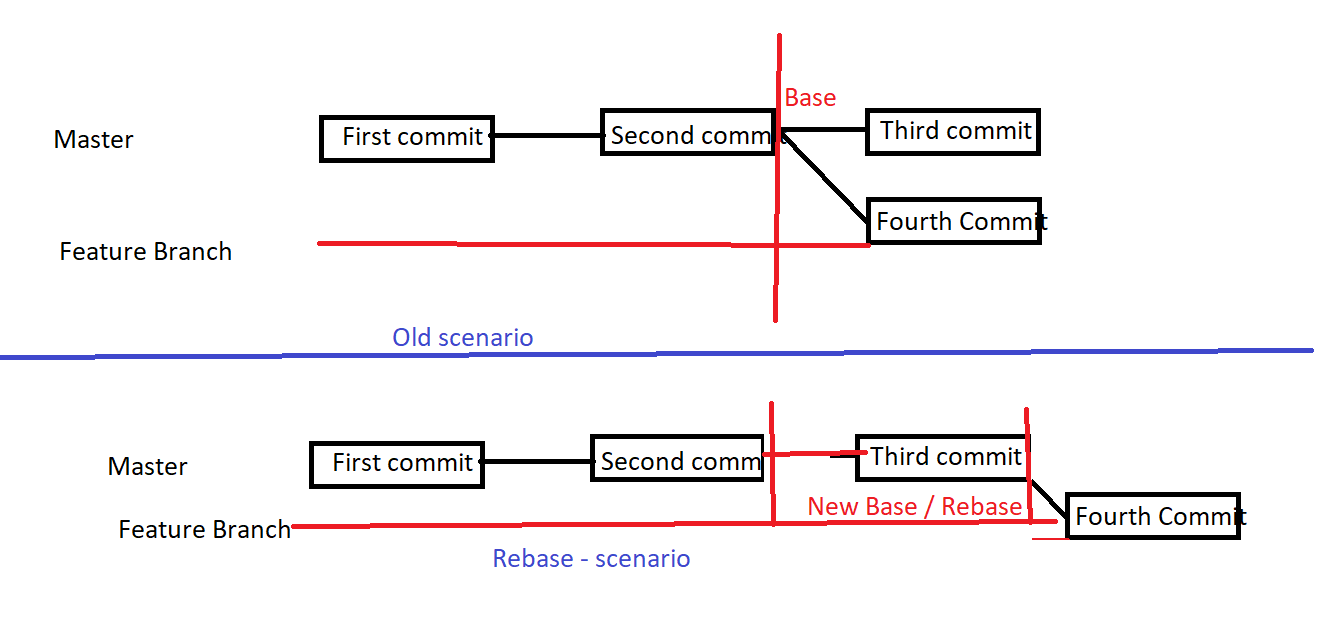
f6278e5 (master) third line --- Base….

d6145a7 second commit

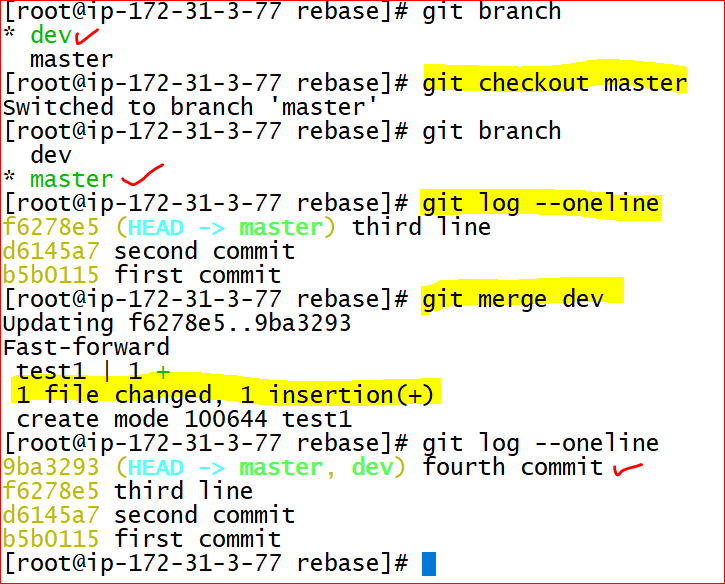
b5b0115 first commit

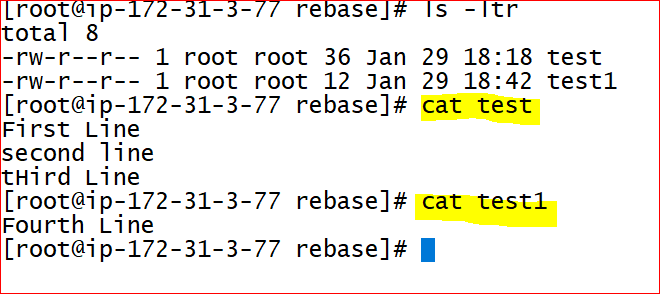
[root@ip-172-31-3-77 rebase]#

Now upto the commit third line is base now….base is rearranged…



Now the files in Master and Feature Branches are Merged.





* Master and Feature Branches are equal.
* **#### Git Status – difference between current working directory and the local repository**

**@@ Cherry Pick**

**Cherry pick is use to transfer / move a particular commit to another branch.**

**Rule # 1. The very next commit from the base only moved**

**Need to move all use merge , or select one by one except base.**

[root@ip-172-31-3-77 rebase]# cd ..

[root@ip-172-31-3-77 senthil]# mkdir cherry

[root@ip-172-31-3-77 senthil]#

[root@ip-172-31-3-77 senthil]# cd cherry

[root@ip-172-31-3-77 cherry]# git init

Initialized empty Git repository in /root/senthil/cherry/.git/

[root@ip-172-31-3-77 cherry]# ls -la

total 0

drwxr-xr-x 3 root root 18 Jan 29 20:00 .

drwxr-xr-x 7 root root 69 Jan 29 20:00 ..

drwxr-xr-x 7 root root 119 Jan 29 20:00 .git

[root@ip-172-31-3-77 cherry]# vi test

[root@ip-172-31-3-77 cherry]# git add.

git: 'add.' is not a git command. See 'git --help'.

The most similar command is

add

[root@ip-172-31-3-77 cherry]# git add .

[root@ip-172-31-3-77 cherry]# git commit -m ' first commit'

[master (root-commit) 7a69826] first commit

1 file changed, 1 insertion(+)

create mode 100644 test

[root@ip-172-31-3-77 cherry]# git log

commit 7a6982685343fe74b6feb78bfc01e815057fd3bd (HEAD -> master)

Author: Senthil <bharathiSen@gmail.com>

Date: Fri Jan 29 20:02:16 2021 +0000

first commit

[root@ip-172-31-3-77 cherry]#

[root@ip-172-31-3-77 cherry]# git branch

\* master

[root@ip-172-31-3-77 cherry]# git branch dev

[root@ip-172-31-3-77 cherry]# git checkout dev

Switched to branch 'dev'

[root@ip-172-31-3-77 cherry]# git branch

\* dev

master

[root@ip-172-31-3-77 cherry]# git log

commit 7a6982685343fe74b6feb78bfc01e815057fd3bd (HEAD -> dev, master)

Author: Senthil <bharathiSen@gmail.com>

Date: Fri Jan 29 20:02:16 2021 +0000

first commit

[root@ip-172-31-3-77 cherry]#

-------------------------------------------------------------------------------------

[root@ip-172-31-3-77 cherry]# git branch

\* dev

master

[root@ip-172-31-3-77 cherry]# vi test

[root@ip-172-31-3-77 cherry]# git commit -am "Second Commit"

[dev 9f3ddef] Second Commit

1 file changed, 1 insertion(+)

[root@ip-172-31-3-77 cherry]# git og

git: 'og' is not a git command. See 'git --help'.

The most similar command is

log

[root@ip-172-31-3-77 cherry]# git log

commit **9f3ddef2852a5ce9ef3f48b486fc7d0b1cc7adcd (HEAD -> dev)**

Author: Senthil <bharathiSen@gmail.com>

Date: Fri Jan 29 20:09:25 2021 +0000

Second Commit

commit 7a6982685343fe74b6feb78bfc01e815057fd3bd (master)

Author: Senthil <bharathiSen@gmail.com>

Date: Fri Jan 29 20:02:16 2021 +0000

first commit

[root@ip-172-31-3-77 cherry]#

------------------------------- Master –

[root@ip-172-31-3-77 cherry]# git checkout master

Switched to branch 'master'

[root@ip-172-31-3-77 cherry]# git branch

dev

\* master

[root@ip-172-31-3-77 cherry]# git cherry-pick **9f3ddef2852a5ce9ef3f48b486fc7d0b1cc7adcd**

[master **5f19134]** Second Commit

Date: Fri Jan 29 20:09:25 2021 +0000

1 file changed, 1 insertion(+)

[root@ip-172-31-3-77 cherry]#

# git cherry-pick is a git command to copy one commit from the feature branch to master branch.

Except the **unique commit-id** , nothing will change with in the two commits.  
**\*\*\* Commit – only copying, nothing move**

**Merge : Mastet 🡪 first commit Feature branch🡪 second commit**

**Third commit**

**Fourth commit**

**To merge all file # git merge Feature branch**

**To pick only a specific file that must be a very next file of the Base , we use Cherry-pick**

**# git cherry-pick <commit-id>.**

**Want to pick all files we should use one-by-one except final commit**

**5 final commit**

**4**

**3**

**2**

**1 base –first commit**

**# git cherry-pick <commit-id of 2>.**

**# git cherry-pick <commit-id of 3>.**

**# git cherry-pick <commit-id of 4>.**

-----------------------------------------------------------------------------

**delete a Git branch locally and remotely in this article.**

[root@ip-172-31-3-77 merge**]# git branch**

devnew

\* master

[root@ip-172-31-3-77 merge**]# git branch -d devnew**

Deleted branch devnew (was e0bcc2f).

[root@ip-172-31-3-77 merge]# git branch

\* master

// delete branch remotely

git push origin --delete remoteBranchName

----------------------------------------------------------------------------------------------------------------

[root@ip-172-31-3-77 merge]# git branch

[root@ip-172-31-2-120 git2]# pwd

/root/senthil/git2

[root@ip-172-31-2-120 git2]# ls -lrt

total 4

-rw-r--r-- 1 root root 169 Jan 19 05:08 gittest

[root@ip-172-31-2-120 git2]# git branch

\* dev

master

[root@ip-172-31-2-120 git2]# ls -lrt

total 4

-rw-r--r-- 1 root root 169 Jan 19 05:08 gittest

[root@ip-172-31-2-120 git2]# git log --oneline

8503936 (HEAD -> dev, git2/master, master) the third update

b618ced My second spell

62c7d59 My first commit in Gittest

[root@ip-172-31-2-120 git2]# cat gittest

Hi this is my test doc for gittetst....

hope doing well & pursue.

---------------------------------------

hi this is my second spel

----

this is test for fetch and push

[root@ip-172-31-2-120 git2]#

\*\*\*\*\*\* ----------------------------------

\*\*\*\* Merging Files ------

-- # Merging will do only in master branch

-- #After adding & commit files into featured branches the files will be merged with Master branch.

# gir merge --> the new commit what happend in the featured branch <ex.dev> will be copied and settled in master branch.

# git merge dev <featured branch >

Updating 8503936..9c0f6cb

Fast-forward

gittest | 2 ++

1 file changed, 2 insertions(+)

[root@ip-172-31-2-120 git2]#

-----------

now the master and featured branches are same.

Master = dev.

but the variable folder that created in github is not updated. here we use git2 as remotename.

#git push git2 master

remotename branchname

now the remote github reposiary also updated.'

[root@ip-172-31-2-120 git2]# git push git2 master

Username for 'https://github.com': Senthilkumar1974

Password for 'https://Senthilkumar1974@github.com':

Enumerating objects: 5, done.

Counting objects: 100% (5/5), done.

Compressing objects: 100% (2/2), done.

Writing objects: 100% (3/3), 298 bytes | 298.00 KiB/s, done.

Total 3 (delta 1), reused 0 (delta 0)

remote: Resolving deltas: 100% (1/1), completed with 1 local object.

To https://github.com/Senthilkumar1974/Gitnew

8503936..9c0f6cb master -> master

[root@ip-172-31-2-120 git2]#

--------------------------

# git commit -a -m / # git commit -am - will apply only for modification files only. not for untracked files.

[root@ip-172-31-2-120 sce2]# git commit -a -m " this is new sce-2 -1"

[master b66d6ba] this is new sce-2 -1

1 file changed, 1 insertion(+)

[root@ip-172-31-2-120 sce2]# git branch

\* master

[root@ip-172-31-2-120 sce2]#

---------- sce2----

###-- Merge Conflict state issuesin Git?

How to resolve the conflict?

# Ans : multiple developers working on a same file , without the proper notfication.

--> featured branch --dev

--- when try to merge a file from featured branch from the master , if there is no space/position to that paticular location in the master,

the conflict may occur.

>>> reason : the other user may add r occupied his file in the master brnc itself.

-->> to resolve the conflict we use a new tool Mergetool , which is not default in git, need to configure.

What can the conflict occur Merge tool used to merge the file -

git config --global merge.tool vimdiff

git config --global merge.conflictstyle diff3 - show three different conflict screens

git config --global mergetool.prompt false

[root@ip-172-31-2-120 sce2]# git log

commit 26840acfca9d380d685ed8485d642c933416de39 (HEAD -> master)

Author: Senthil <bharathiSen@gmail.com>

Date: Sat Jan 23 02:25:08 2021 +0000

3rd commit

commit b66d6ba11789c6cad2646c80dae716bc1c9cc66c

Author: Senthil <bharathiSen@gmail.com>

Date: Sat Jan 23 02:03:59 2021 +0000

this is new sce-2 -1

commit 4997ccf690448ea6d8a8dee5eec95a646c525216

Author: Senthil <bharathiSen@gmail.com>

Date: Fri Jan 22 19:52:20 2021 +0000

First commit - newtest

[root@ip-172-31-2-120 sce2]# git checkout dev

newtest: needs merge

error: you need to resolve your current index first

[root@ip-172-31-2-120 sce2]# git branch

dev

\* master

[root@ip-172-31-2-120 sce2]# cat newtest

<<<<<<< HEAD

# new test /senthil/sce2

Cat

Dog

Tiger

=======

# new test /senthil/sce2

Cat

Dog

fish

>>>>>>> dev

[root@ip-172-31-2-120 sce2]# git config --global merge.tool vimdiff

[root@ip-172-31-2-120 sce2]# git config --global merge.conflictstyle diff3

[root@ip-172-31-2-120 sce2]# git config --global mergetool.prompt false

[root@ip-172-31-2-120 sce2]# git mergetool

Merging:

newtest

Normal merge conflict for 'newtest':

{local}: modified file

{remote}: modified file

4 files to edit

[root@ip-172-31-2-120 sce2]#

## now the text is updated as need.

[root@ip-172-31-2-120 sce2]# cat newtest

<<<<<<< HEAD

# new test /senthil/sce2

Cat

Dog

fish

[root@ip-172-31-2-120 sce2]#

## now we check the working directory = local repository

# git status -s

M newtest - M in green

?? newtest.orig --?? in untracked

[root@ip-172-31-2-120 sce2]#

## M newtest - M in green is a modification file that made by mergetool, is a git tool. M means the file is staging / add from Working directory

its done by git Mergingtool which is using git , so the git is also called as STAGING TOOL

GIT- VCT, SCM also STAGING TOOL

##?? newtest.orig --?? in untracked - its a backup file of conflicted file, the mergingtool backeuped the file before merge for future reference. we may use it or not.

[root@ip-172-31-2-120 sce2]# git commit -m " final commit"

[master 57bf616] final commit

[root@ip-172-31-2-120 sce2]# git log

commit 57bf6165b1b675988fbbc74eef4755b470584565 (HEAD -> master)

Merge: 26840ac 5d00208

Author: Senthil <bharathiSen@gmail.com>

Date: Sat Jan 23 05:24:27 2021 +0000

final commit

commit 26840acfca9d380d685ed8485d642c933416de39

Author: Senthil <bharathiSen@gmail.com>

Date: Sat Jan 23 02:25:08 2021 +0000

3rd commit

commit 5d00208efaeb87cb4ef6cab8c10b5d0d9a292c8a (dev)

Author: Senthil <bharathiSen@gmail.com>

Date: Sat Jan 23 02:14:47 2021 +0000

third commit

commit b66d6ba11789c6cad2646c80dae716bc1c9cc66c

Author: Senthil <bharathiSen@gmail.com>

Date: Sat Jan 23 02:03:59 2021 +0000

this is new sce-2 -1

commit 4997ccf690448ea6d8a8dee5eec95a646c525216

Author: Senthil <bharathiSen@gmail.com>

Date: Fri Jan 22 19:52:20 2021 +0000

First commit - newtest

[root@ip-172-31-2-120 sce2]# git log --oneline

57bf616 (HEAD -> master) final commit

26840ac 3rd commit

5d00208 (dev) third commit

b66d6ba this is new sce-2 -1

4997ccf First commit - newtest

[root@ip-172-31-2-120 sce2]#

master -- head -- 5 commits

dev ---3 commits

/-------------------------------------------------------------------------------------

------- revert commands------->

# git reflog - look at the history of our files and status

# git fetch git2 master

From https://github.com/Senthilkumar1974/Gitnew

\* branch master -> FETCH\_HEAD

/-------------------------------------------------------------------------------------

===--------------------------------------------- NOTES ------------------------------------------------------------------------------------------

\*\*\* What is "HEAD" in Git?

When working with Git, only one branch can be checked out at a time - and this is what's called the "HEAD" branch. Often, this is also referred to as the "active" or "current" branch.

Git makes note of this current branch in a file located inside the Git repository, in .git/HEAD. (This is an internal file, so it should not be manually manipulated!)

If you wonder what exactly this file contains, you can simply have its contents printed on the command line:

$ cat .git/HEAD

ref: refs/heads/master

In this example case, a local branch named "master" is the current HEAD.

06:56 AM 19-01-2021

1 git --version

2 yum install git

3 git --version

4 mkdir senthil

5 cd senthil/

6 mkdir test

7 cd test

8 pwd

9 ls -lrt

10 ls -lrta

11 ls -lrth

12 ls -lrta

13 cleaf

14 clear

15 git init git init

16 git init

17 pwd

18 ls-lrta

19 ls -lrta

20 cd .git

21 ls -la

22 ls -lrta

23 cd ..

24 cd .git

25 pwd

26 clear

27 pwd

28 cd ..

29 pwd

30 git config --global user.name "Senthil"

31 git config --global user.email "bharathiSen@gmail.com"

32 git config -l

33 git log

34 vi test

35 cat test

36 clea

37 clear

38 ls -ltra

39 git status -s

40 git status

41 git log --l

42 clear

43 git status

44 git add test

45 git status -s

46 git reflog

47 first commit -m "My first commit in GIT"

48 git commit -m "My first commit in GIT"

49 git log

50 git reflog

51 ]# git reflog

52 clear

53 git reflog

54 git status -s

55 git status

56 vi test

57 git status

58 git status -s

59 git add test

60 git status -s

61 git commit -m "My second line commitT"

62 git log

63 git log --oneline

64 git show

65 git show 3738cfe

66 clear

67 git log --oneline

68 git show

69 cleafr

70 clear

71 git log --oneline

72 git show 3738cfe

73 git show

74 clear

75 git show

76 5d582f3clear

77 cat .git/HEAD

78 git branch

79 git remote add test https://github.com/Senthilkumar1974/GIT-course1

80 git remote

81 git push test master

82 ls -la

83 git push test master

84 git remote add test https://github.com/Senthilkumar1974/my-app

85 git remote add test https://github.com/Senthilkumar1974/my-app1

86 git remote add test https://github.com/Senthilkumar1974/GIT-course1

87 git push test master

88 clear

89 git remote add test https://github.com/Senthilkumar1974/Gittest.git

90 git remote add test https://github.com/Senthilkumar1974/Gittest

91 git push test master

92 git remote add gittest https://github.com/Senthilkumar1974/Gi

93 git push gittest master

94 git remote add gittest https://github.com/Senthilkumar1974/GIT-course1

95 git remote add gittest1 https://github.com/Senthilkumar1974/GIT-course1

96 git push gittest master

97 git remote add gittest1 https://github.com/Senthilkumar1974/GIT-course1

98 git remote add post1 https://github.com/Senthilkumar1974/giitet

99 git push post1 master

100 git remote add test https://github.com/Senthilkumar1974/Gittest

101 git push post1 master

102 git remote add test https://github.com/Senthilkumar1974/Gittest

103 clear

104 git remote add testpg https://github.com/Senthilkumar1974/Gittest

105 git push testpg master

106 git remote -v

107 git blame

108 git remote -v

109 git remote show

110 clear

111 get fetch

112 git fetch

113 git fetch Senthilkumar1974

114 /

115 Gittest

116 git fetch Senthilkumar1974/gittest

117 git fetch Senthilkumar1974/Gittest

118 git fetch Senthilkumar1974/Gittest/test

119 git fetch https://github.com/Senthilkumar1974/Gittest

120 clear

121 lrt

122 ls -lrt

123 pwd

124 ls -lrt

125 cat test

126 cd senthil

127 cd..

128 cd /

129 ls -lrth

130 exit

131 pwd

132 history

10:22 PM 19-01-2021