= Git Tutorial=

1. **git init ---> creates local repository:**

git config --list → **Global(available for all the projects on the system)**

2. **Most important config parameters to be set at global level :**

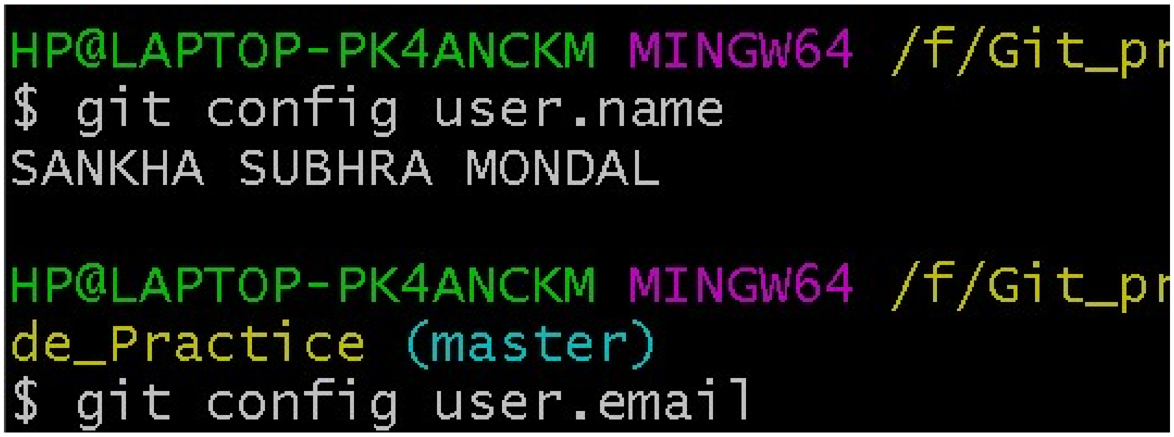
   >> git config --global user.name "<username>" ↩

   >> git config --global user.email "<email ID>" ↩

1. **Check local <username> & <email ID>:**

   >> git config --global user.name ↩

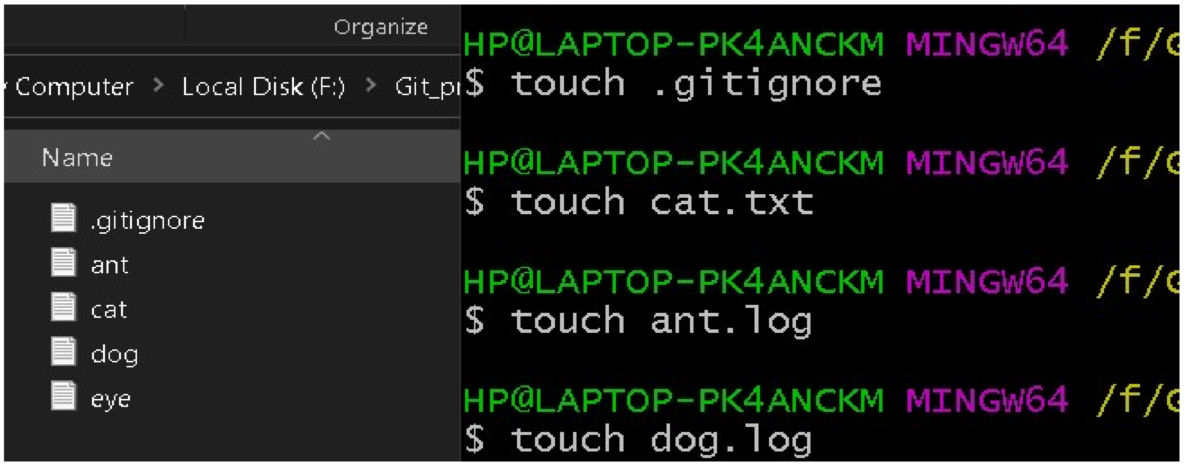
   >> git config --global user.email ↩



```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_pr
$ git config user.name
SANKHA SUBHRA MONDAL

HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_pr
de_Practice (master)
$ git config user.email
```

2. **To Create a file:**

   >> touch file.extension (e.g: Test.java, index.html)



```
HP@LAPTOP-PK4ANCKM MINGW64 /f/G
$ touch .gitignore

HP@LAPTOP-PK4ANCKM MINGW64 /f/G
$ touch cat.txt

HP@LAPTOP-PK4ANCKM MINGW64 /f/G
$ touch ant.log

HP@LAPTOP-PK4ANCKM MINGW64 /f/G
$ touch dog.log
```

3. To Create a file/

```
>> mkdir NAME-OF-YOUR-NEW-DIRECTORY
```

4. Go back one directory or file:

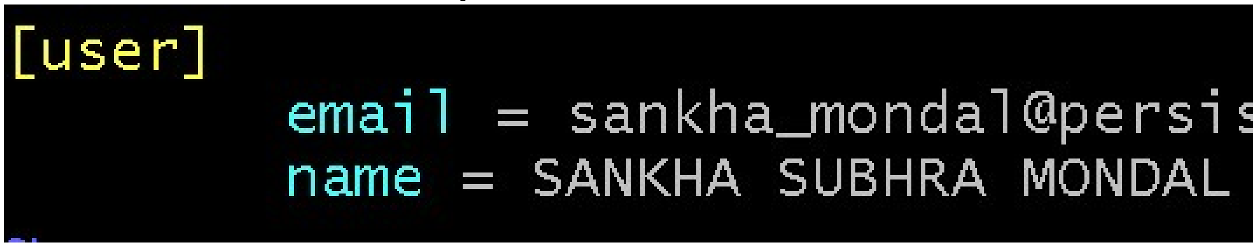
```
>> cd ../
```

5. Edit local <username> & <email ID>:

```
>> git config --global --edit ↩
      [user]
          email = sankha_mondal@persistent.com
          name = SANKHA SUBHRA MONDAL
```

After editing is done to exit press

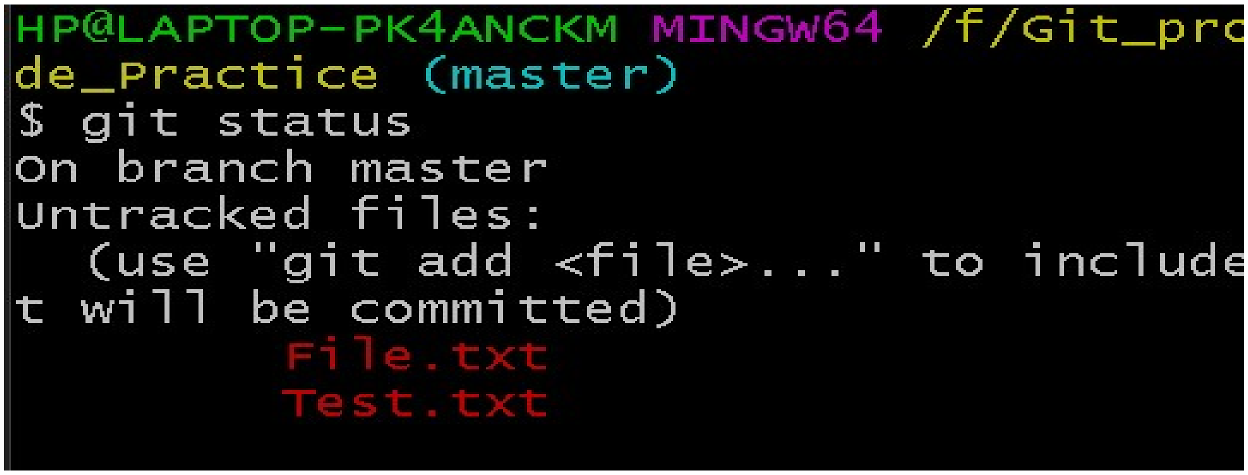
Esc ▶ : ▶ w ▶ q ▶ ↩



```
[user]
        email = sankha_mondal@persis
        name = SANKHA SUBHRA MONDAL
```

6. To Check current position:

```
>> git status ↩
```

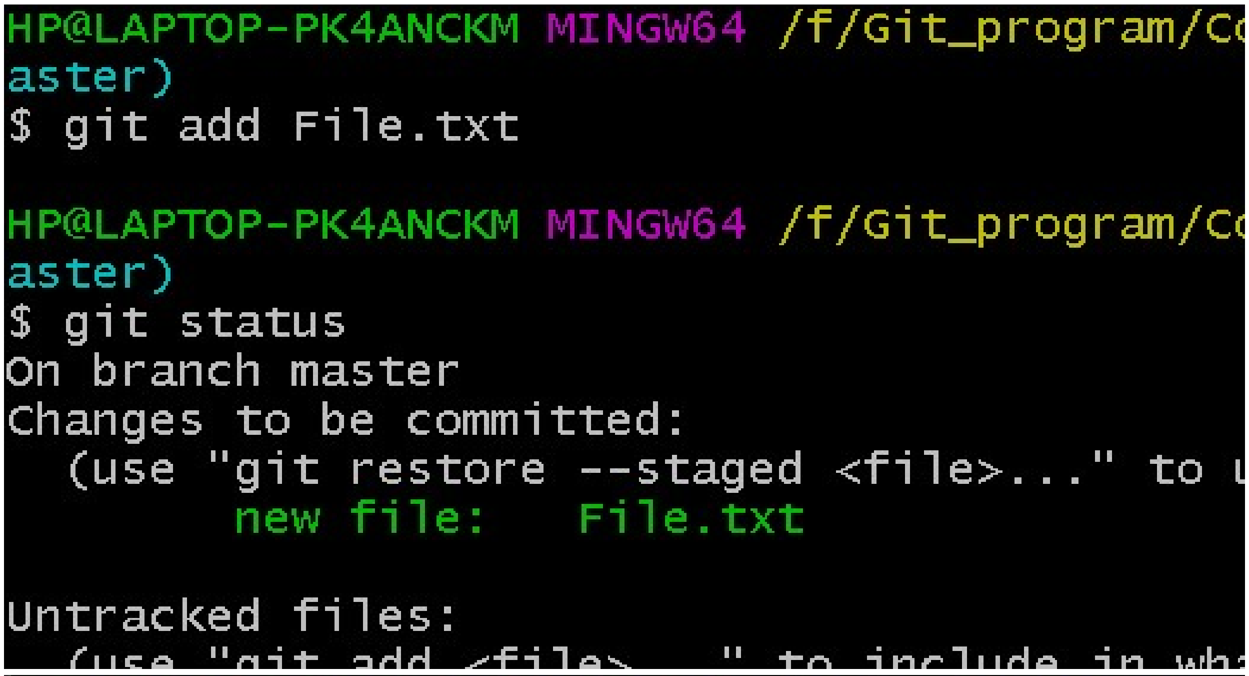


```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_pro
de_Practice (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include
t will be committed)
        File.txt
        Test.txt
```

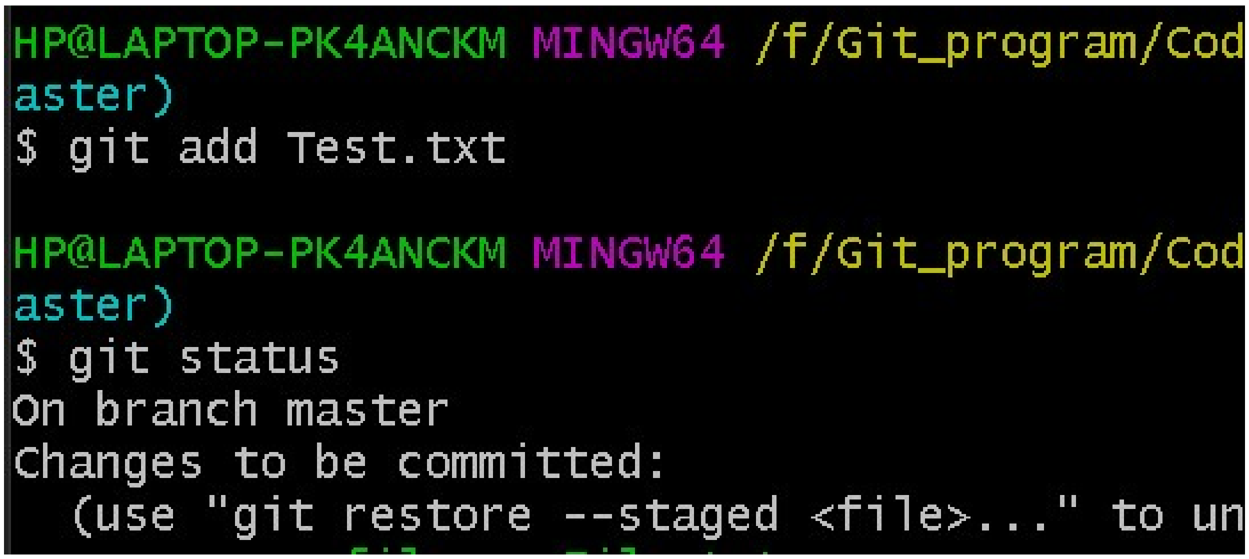
6. Add any <file> to Staging area(SA)/ Caching area/ Index area:

>> git add <file> ↩

Automatically added to Staging area



```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Co
aster)
$ git add File.txt

HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Co
aster)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to u
        new file:   File.txt

Untracked files:
  (use "git add <file>..." to include in wha
```

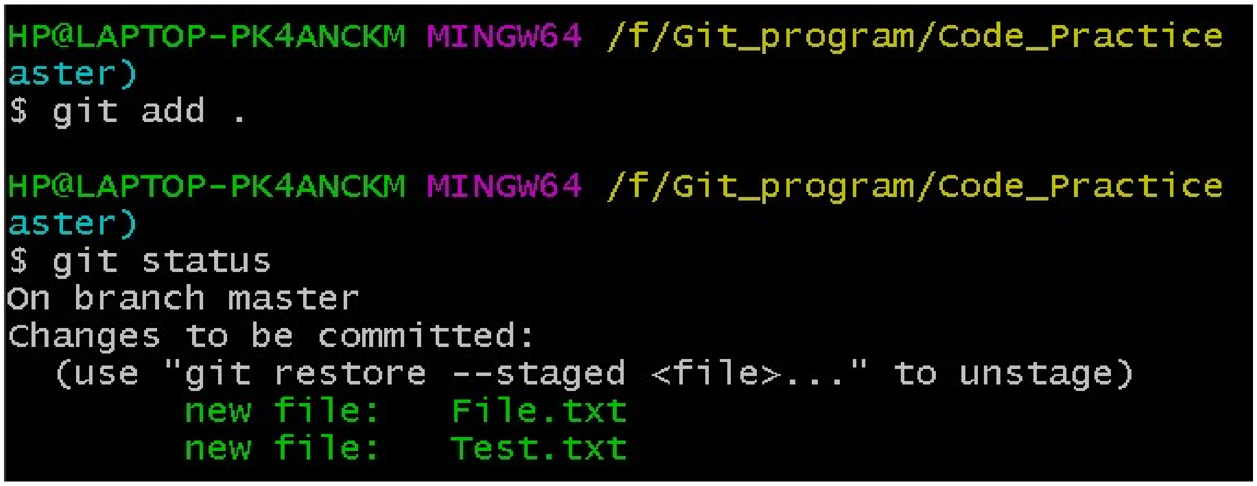


```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Cod
aster)
$ git add Test.txt

HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Cod
aster)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to un
```

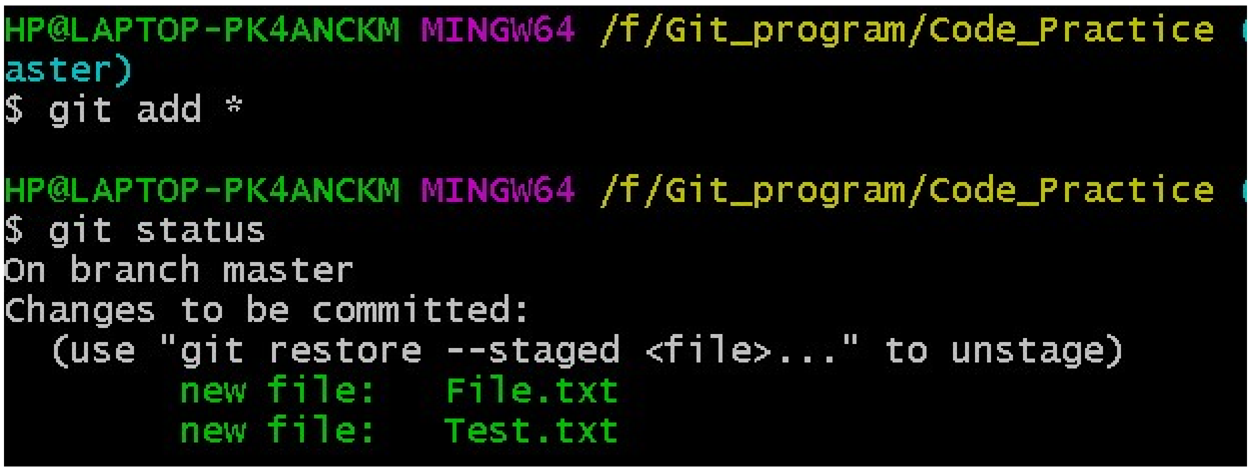
7. Add ALL <file> to Staging area(SA)/ Caching area/ Index area:

>> git add . ↩ or **git add *** ↩

Automatically ALL files added to Staging area



```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Code_Practice
aster)
$ git add .

HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Code_Practice
aster)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   File.txt
        new file:   Test.txt
```



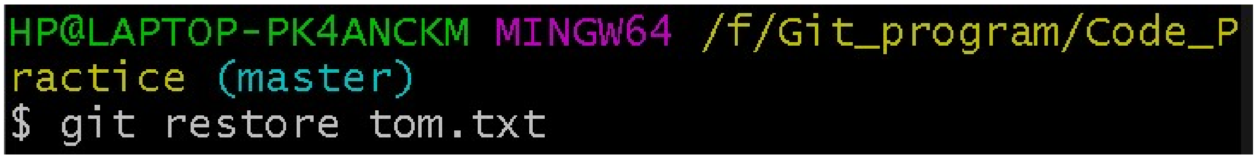
```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Code_Practice
aster)
$ git add *

HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Code_Practice
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   File.txt
        new file:   Test.txt
```

8. To discard changes in Working Directory(WD) before Staging:

>> git restore <file> ↩

I am tom. **Staged Condition**

I am dog. **Not Staged Condition**



```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Code_P
ractice (master)
$ git restore tom.txt
```

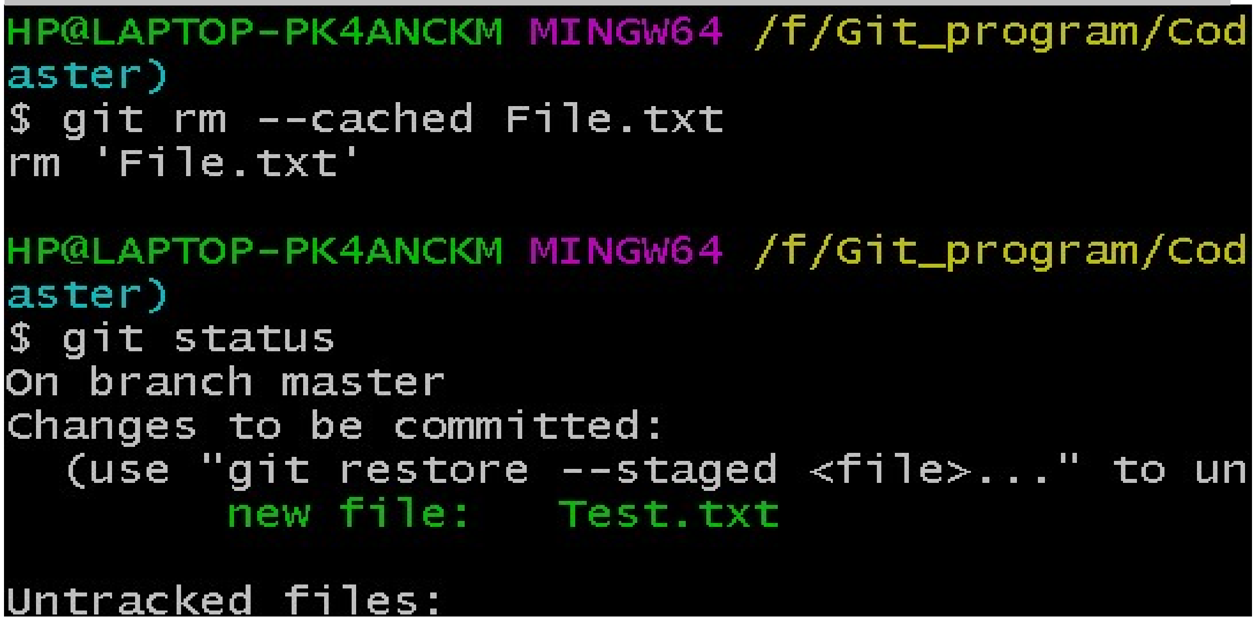
I am tom. **(Present)**

I am dog. **(Absence)**

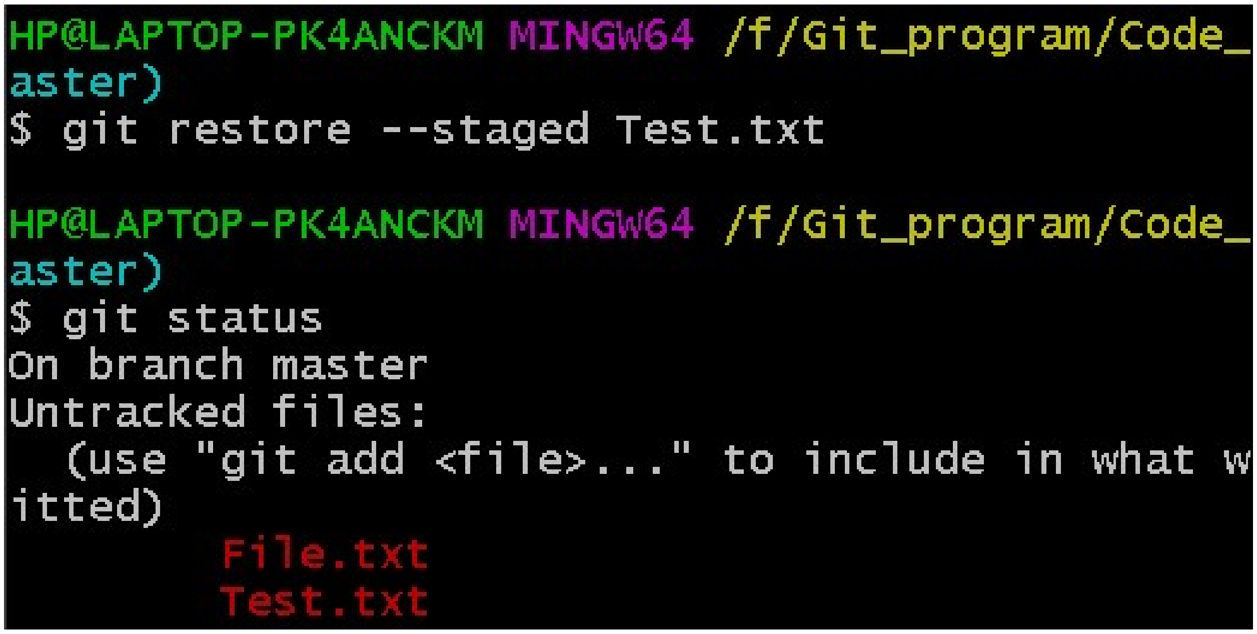
9. **Remove <file> from Staging area(SA)/ Caching area/ Index area to Unstage:**

git rm --cached <file> ↩ or **git restore --staged <file>** ↩

`rm 'File.txt'` **i.e remove the file from SA or make it untracked.**



```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Cod
aster)
$ git rm --cached File.txt
rm 'File.txt'

HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Cod
aster)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to un
        new file:   Test.txt

Untracked files:
```



```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Code_
aster)
$ git restore --staged Test.txt

HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Code_
aster)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what w
itted)
        File.txt
        Test.txt
```

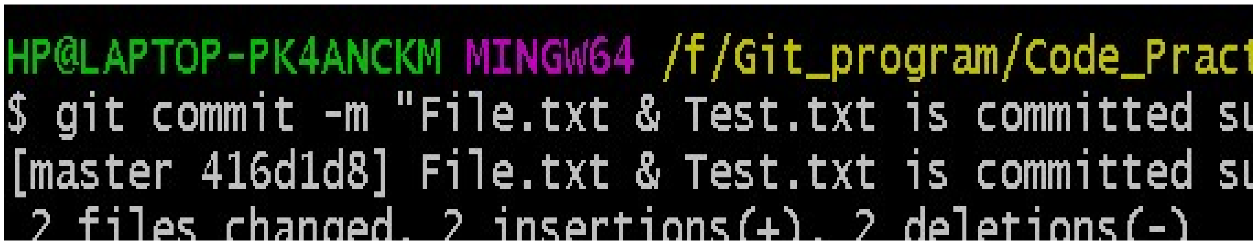
10. **What is the command to add all files and changes of the current folder to the staging environment of the Git repository?**

>> **git add --all**

11. Commit(save) the change to Local Reposatory:

>> **git commit -m "valid-msg"** ↩

e.g : git commit -m "File committed successfully"

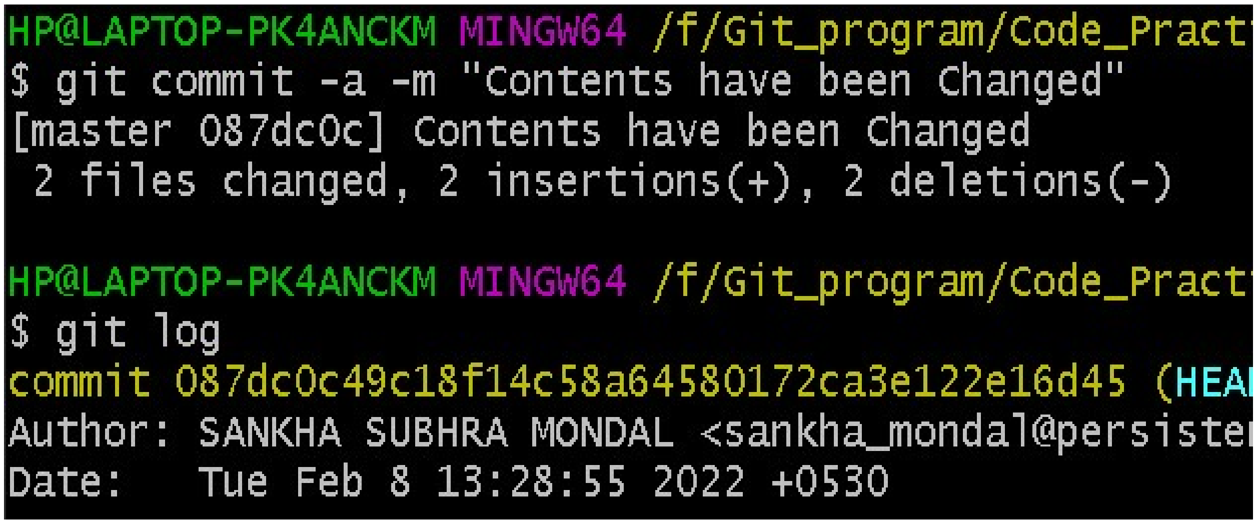


```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Code_Pract
$ git commit -m "File.txt & Test.txt is committed su
[master 416d1d8] File.txt & Test.txt is committed su
 2 files changed, 2 insertions(+), 2 deletions(-)
```

12. To add <file> to Staging area and Commit(save) the change to Local Repository TOGETHER (git add + git commit):

>> **git commit -a -m "msg"** ↩ **or** **git commit --am “msg”** ↩

Note: Don’t use this at the very first time. Use it when the file is already being tracked.



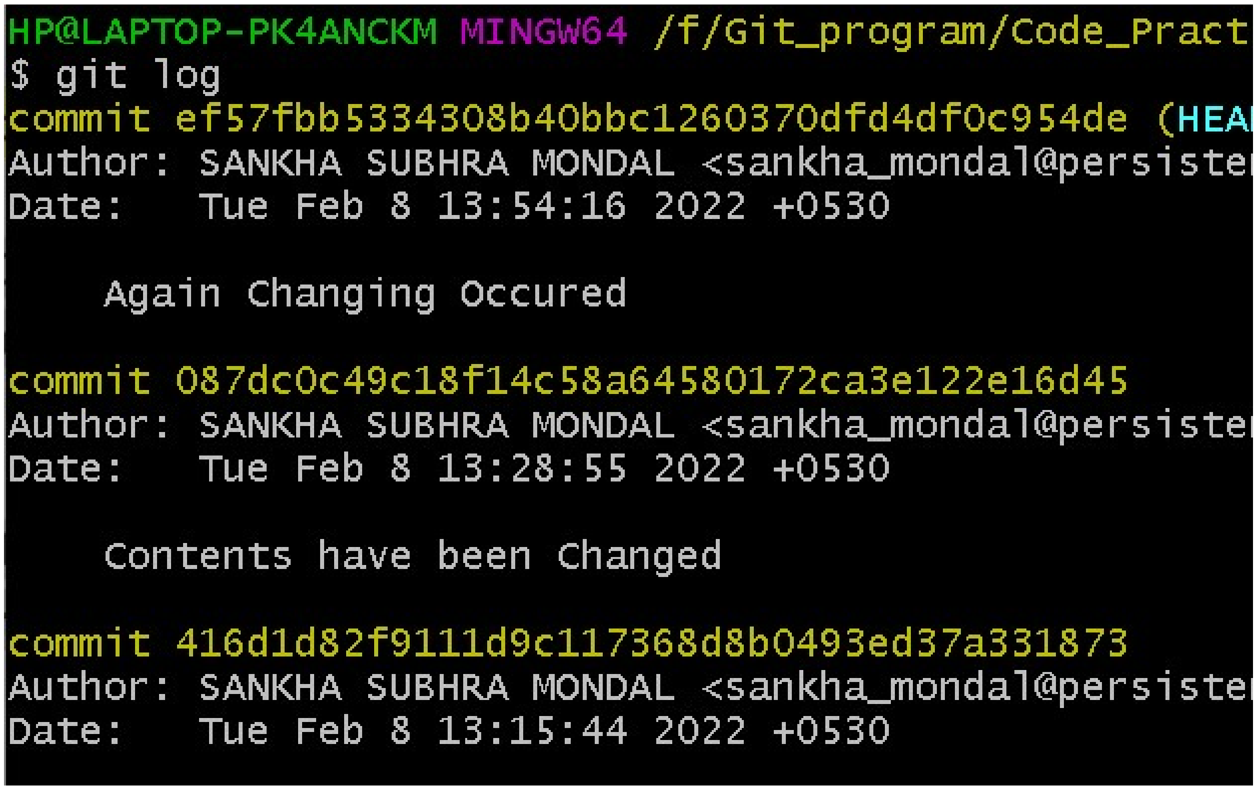
```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Code_Pract
$ git commit -a -m "Contents have been Changed"
[master 087dc0c] Contents have been Changed
 2 files changed, 2 insertions(+), 2 deletions(-)

HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Code_Pract
$ git log
commit 087dc0c49c18f14c58a64580172ca3e122e16d45 (HEA
Author: SANKHA SUBHRA MONDAL <sankha_mondal@persiste
Date:   Tue Feb 8 13:28:55 2022 +0530
```

13. updates the commit msg for latest commit:

>> **git commit --amend -m "msg"**

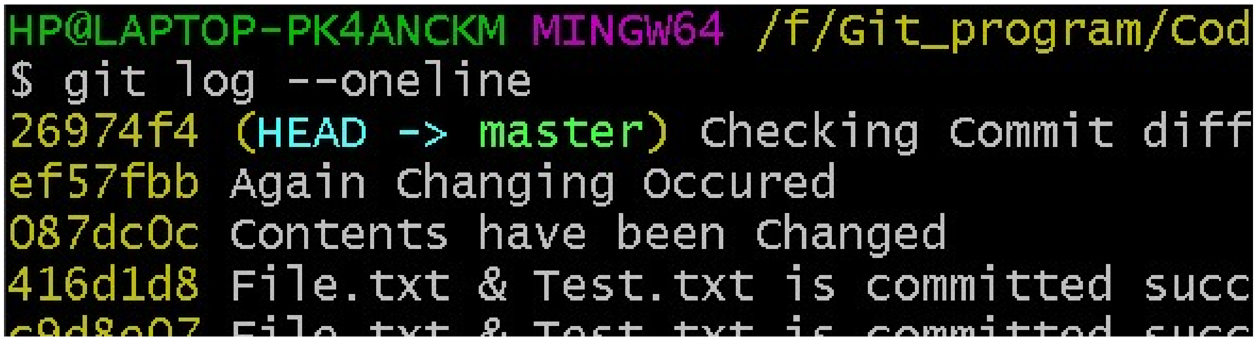
14. To Check the version History of Project :

>> **git log** ↩



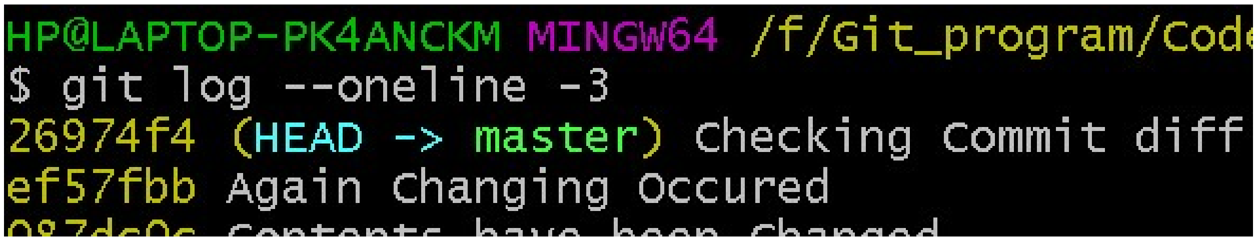
```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Code_Pract
$ git log
commit ef57fbb5334308b40bbc1260370dfd4df0c954de (HEA
Author: SANKHA SUBHRA MONDAL <sankha_mondal@persiste
Date:   Tue Feb 8 13:54:16 2022 +0530

    Again Changing Occured

commit 087dc0c49c18f14c58a64580172ca3e122e16d45
Author: SANKHA SUBHRA MONDAL <sankha_mondal@persiste
Date:   Tue Feb 8 13:28:55 2022 +0530

    Contents have been Changed

commit 416d1d82f9111d9c117368d8b0493ed37a331873
Author: SANKHA SUBHRA MONDAL <sankha_mondal@persiste
Date:   Tue Feb 8 13:15:44 2022 +0530
```

>> **git log --oneline** ↩ → to see only 7 **SHA** commit



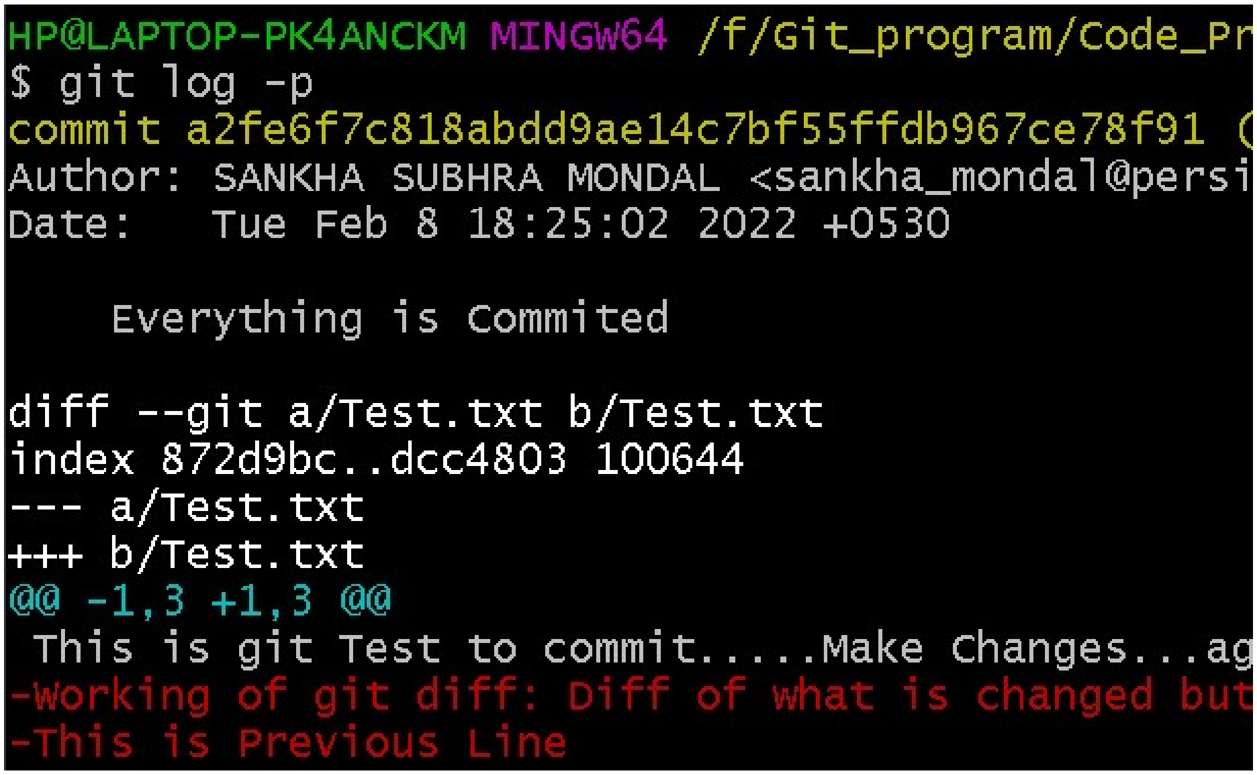
```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Cod
$ git log --oneline
26974f4 (HEAD -> master) Checking Commit diff
ef57fbb Again Changing Occured
087dc0c Contents have been Changed
416d1d8 File.txt & Test.txt is committed succ
```

>> **git log --oneline -3** ↩ → to see only latest 3 commit



```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Code
$ git log --oneline -3
26974f4 (HEAD -> master) Checking Commit diff
ef57fbb Again Changing Occured
```

>> git log -p ↩ → changes done for every commit (fNew Vs fOld)



```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Code_Pr
$ git log -p
commit a2fe6f7c818abdd9ae14c7bf55ffdb967ce78f91 (
Author: SANKHA SUBHRA MONDAL <sankha_mondal@persi
Date:   Tue Feb 8 18:25:02 2022 +0530

    Everything is Commited

diff --git a/Test.txt b/Test.txt
index 872d9bc..dcc4803 100644
--- a/Test.txt
+++ b/Test.txt
@@ -1,3 +1,3 @@
 This is git Test to commit.....Make Changes...ag
-Working of git diff: Diff of what is changed but
-This is Previous Line
```

>> git log -p -2 ↩ → changes done for latest 2 commit.

>> git log --grep="Initial" ↩ Commit done by Specific words

>> git log --author="authorname" ↩ → Commit done by Specific person

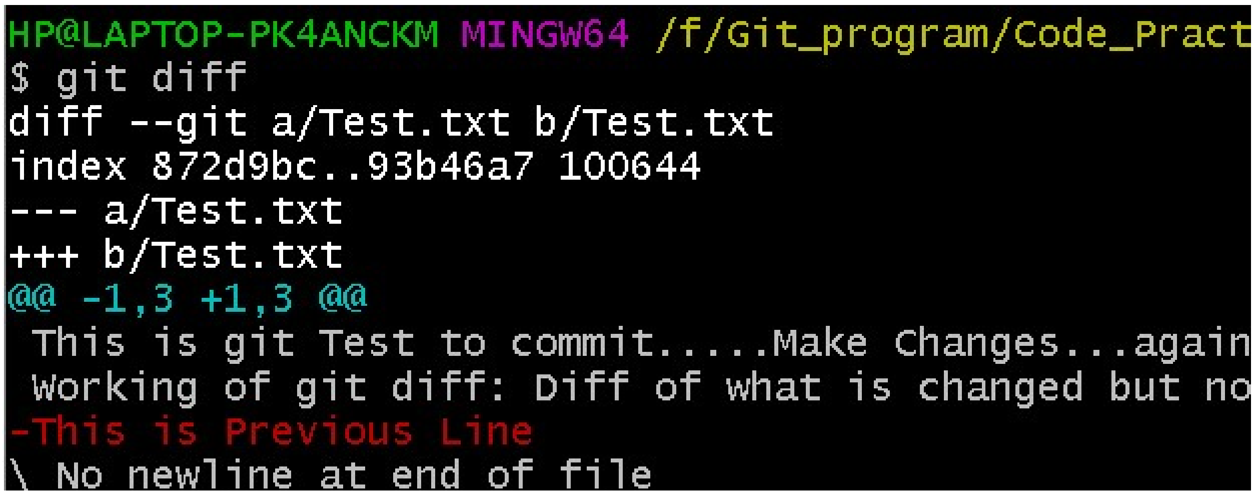
>> git log --since = "07/30/2021" ↩ → Commit done by date

>> git log --until = "07/30/2021" ↩ → Commit done by date

15. To get DIFFERENCE between Stages(WD, SA, Local Repo) of file:

i) Diff of what is changed but not staged

>> **git diff** → difference between WD and SA

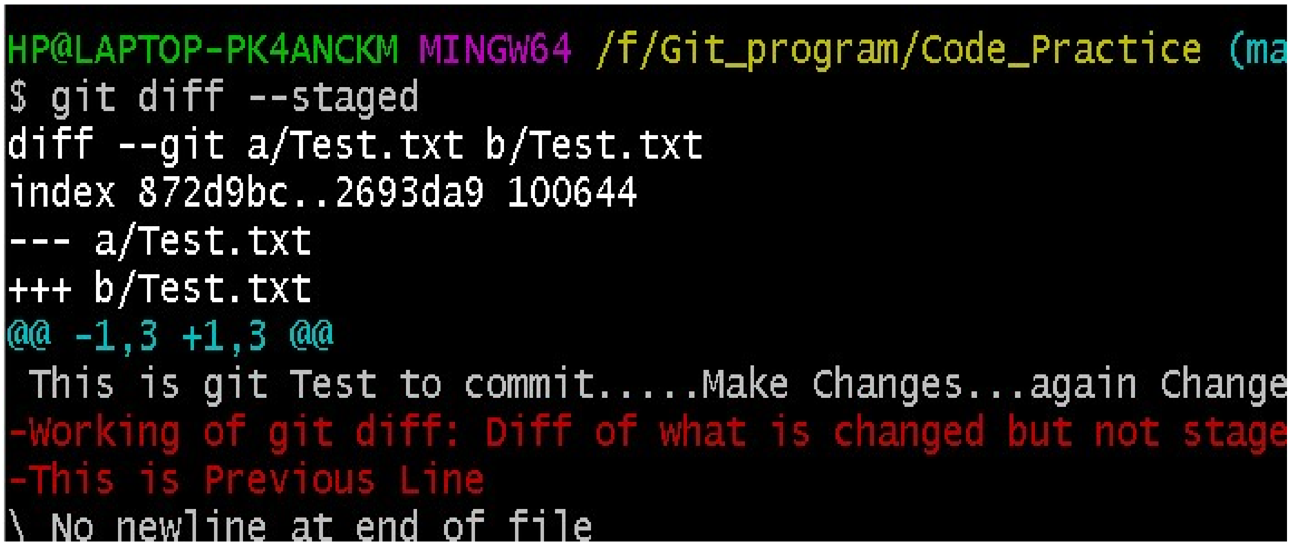


```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Code_Pract
$ git diff
diff --git a/Test.txt b/Test.txt
index 872d9bc..93b46a7 100644
--- a/Test.txt
+++ b/Test.txt
@@ -1,3 +1,3 @@
 This is git Test to commit.....Make Changes...again
 Working of git diff: Diff of what is changed but no
-This is Previous Line
\ No newline at end of file
```

ii) Diff of what is staged but not yet committed

>> **git diff HEAD** → difference between WD and LR (OR)

>> **git diff --staged** → difference between WD and LR



```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Code_Practice (ma
$ git diff --staged
diff --git a/Test.txt b/Test.txt
index 872d9bc..2693da9 100644
--- a/Test.txt
+++ b/Test.txt
@@ -1,3 +1,3 @@
 This is git Test to commit.....Make Changes...again Change
-Working of git diff: Diff of what is changed but not stage
-This is Previous Line
\ No newline at end of file
```

Case 1: Change in Working Directory(WD) :

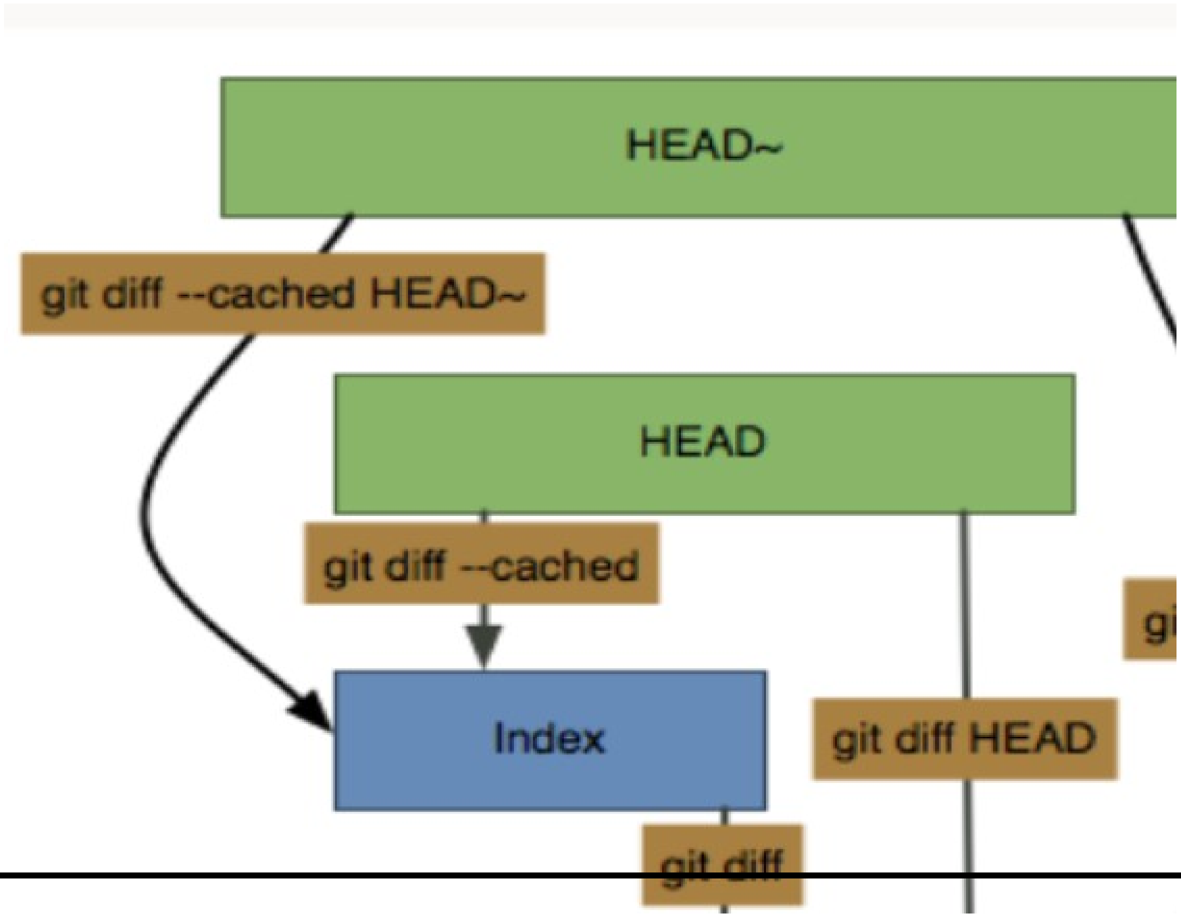
Command	WD	SA	LR	Result
git diff ↩	Change1	0	0	**Change1**
git diff HEAD ↩	Change1	0	0	**Change1**

Case 2: Change in Staging area (SA) :

Command	WD	SA	LR	Result
git diff ↩	Change1	Change1	0	**No Output**
git diff HEAD ↩	Change1	Change1	0	**Change1**
git diff HEAD ↩	Change1	Change2		**Change1 & Change2**

Case 3: Put change to SA and change in WD again :

Command	WD	SA	LR	Result
git diff ↩	Change1+Change2	Change1	0	**Change2**
git diff HEAD ↩	Change1+Change2	Change1	0	**Change1 & Change2**



16. Create File:

>> touch file

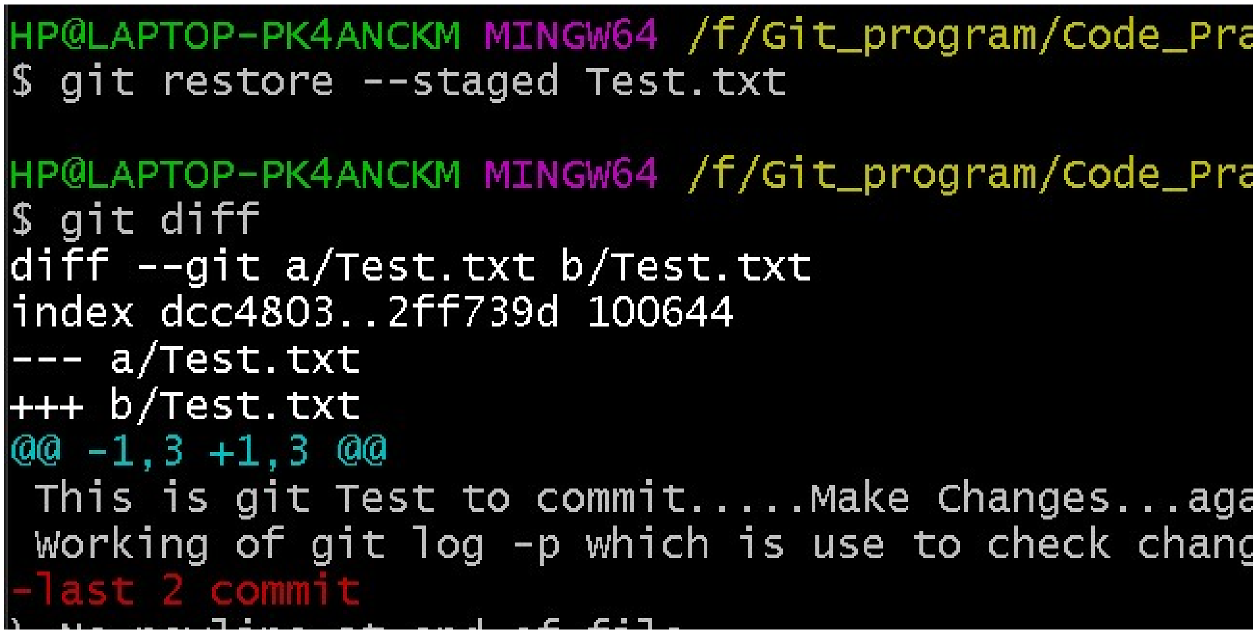
17. Rename File:

>> git mv oldFile newFile

18. Restoring/Undoing local/staged changes :

- **Restore the changes in Staging area(SA) from LR**

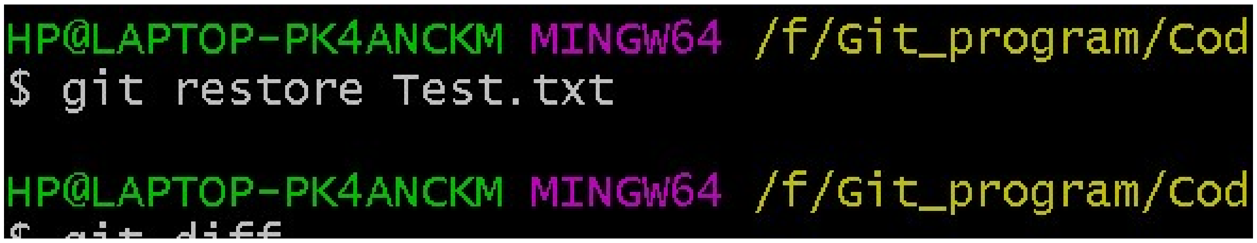
  git restore --staged <file> ↩



```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Code_Pra
$ git restore --staged Test.txt

HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Code_Pra
$ git diff
diff --git a/Test.txt b/Test.txt
index dcc4803..2ff739d 100644
--- a/Test.txt
+++ b/Test.txt
@@ -1,3 +1,3 @@
 This is git Test to commit.....Make Changes...aga
 Working of git log -p which is use to check chang
-last 2 commit
```

- **Restore changes in WD from LR**

  git restore <file> ↩

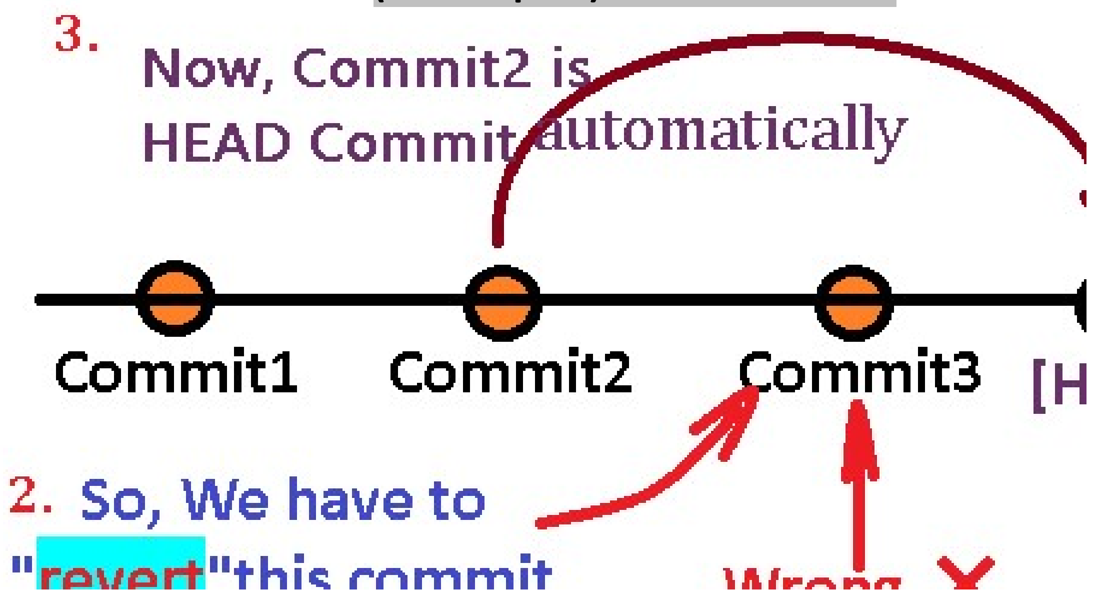


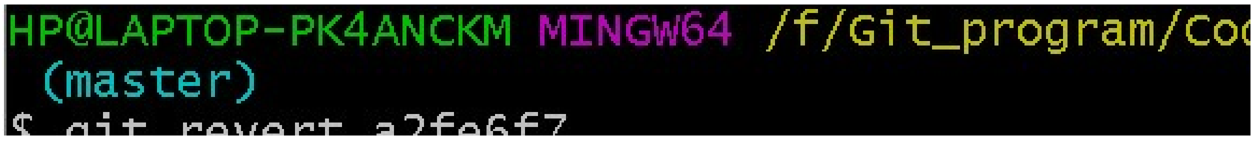
```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Cod
$ git restore Test.txt

HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Cod
```

19. Undoing of committed changes :

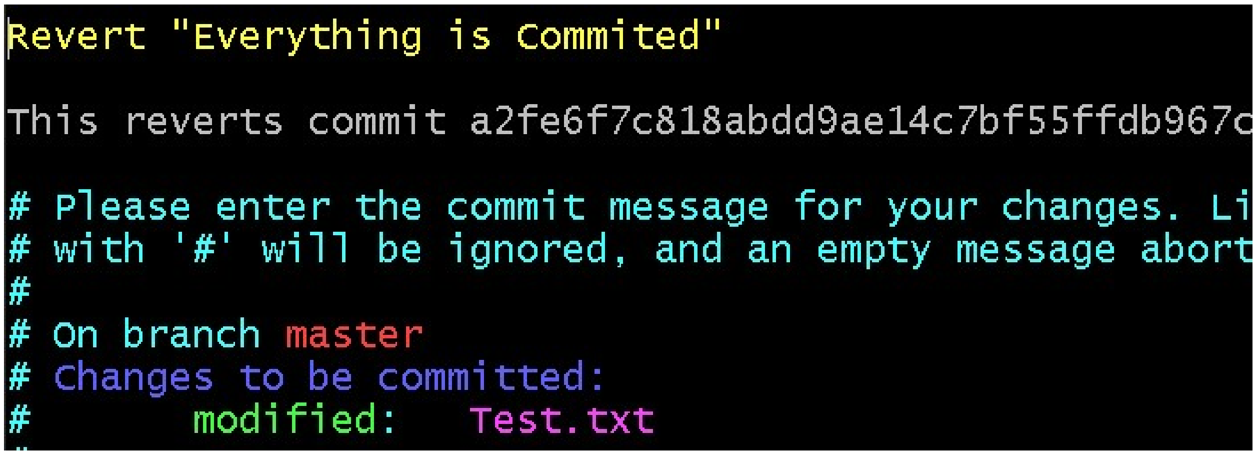
- **Safe way**

>> **git revert <SHA>** ↩ **i.e to ignore wrong commit, treat (before past)commit as HEAD**

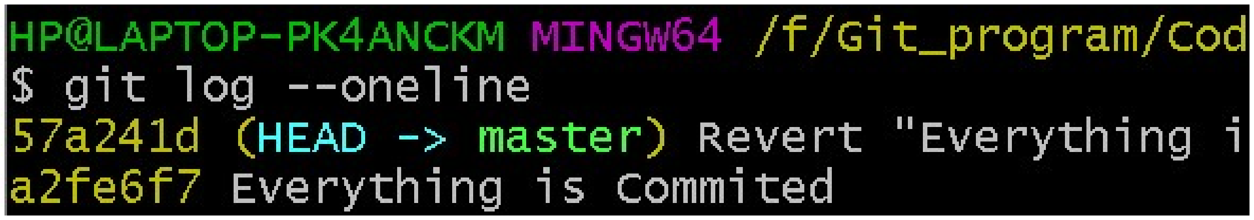

3.
Now, Commit2 is HEAD Commit automatically
Commit1
Commit2
Commit3
[H
2. So, We have to "revert"this commit
Wrong ✗



```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Coo
 (master)
$ git revert a2fe6f7
```



```
Revert "Everything is Commited"

This reverts commit a2fe6f7c818abdd9ae14c7bf55ffdb967c

# Please enter the commit message for your changes. Li
# with '#' will be ignored, and an empty message abort
#
# On branch master
# Changes to be committed:
#       modified:   Test.txt
```

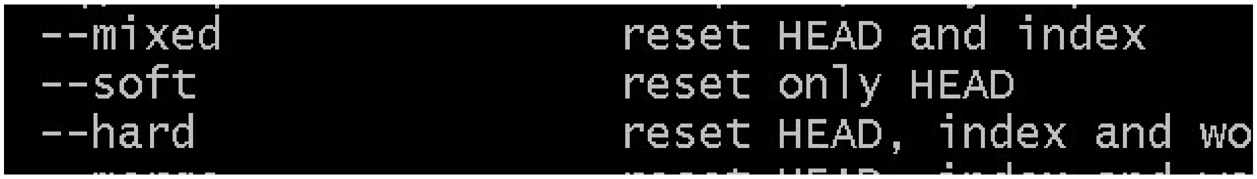


```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Cod
$ git log --oneline
57a241d (HEAD -> master) Revert "Everything i
a2fe6f7 Everything is Commited
```

- **UnSafe way**

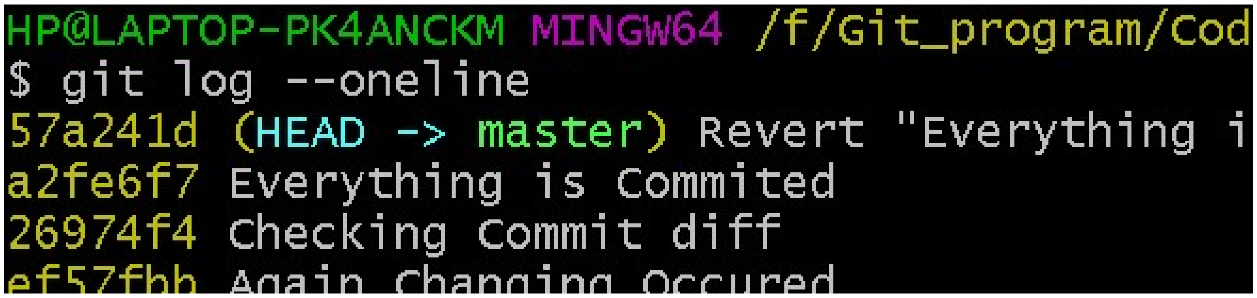
i) git reset --hard <SHA> ↩ To move HEAD pointer, but previous commit is deleted permanently.

Options:



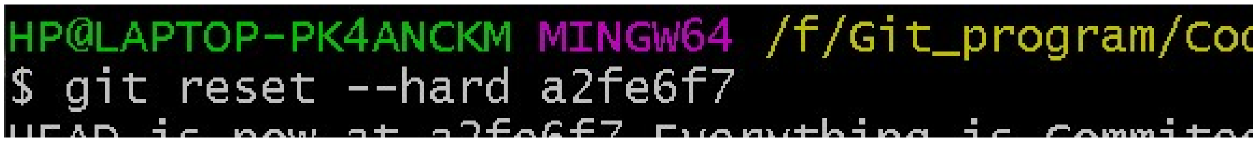
```
 --mixed                 reset HEAD and index
 --soft                  reset only HEAD
 --hard                  reset HEAD, index and wo
```

Previous :



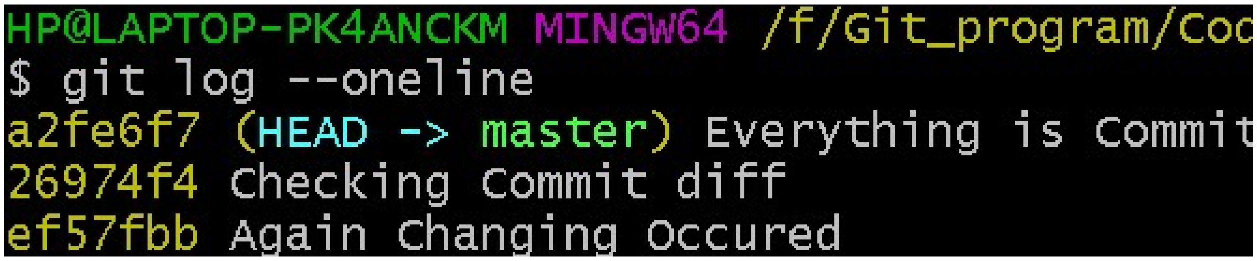
```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Cod
$ git log --oneline
57a241d (HEAD -> master) Revert "Everything i
a2fe6f7 Everything is Commited
26974f4 Checking Commit diff
ef57fbb Again Changing Occured
```

Performing Hard Reset, to move HEAD pointer from (57a241d) **to** (a2fe6f7)



```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Coc
$ git reset --hard a2fe6f7
```

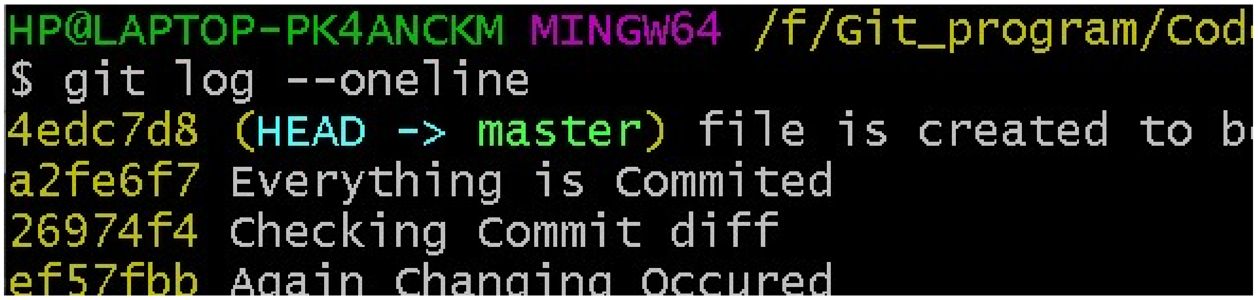
HEAD Pointer changed it's position (57a241d) **to** (a2fe6f7)



```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Coc
$ git log --oneline
a2fe6f7 (HEAD -> master) Everything is Commit
26974f4 Checking Commit diff
ef57fbb Again Changing Occured
```

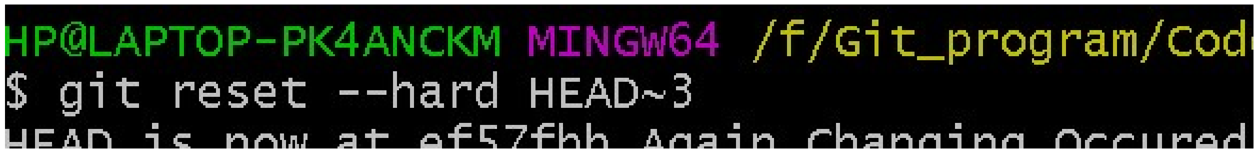
ii) git reset --hard~3 ↩ Permanently delete new commit & HEAD pointer is pointing to 4th Commit.

Previous :



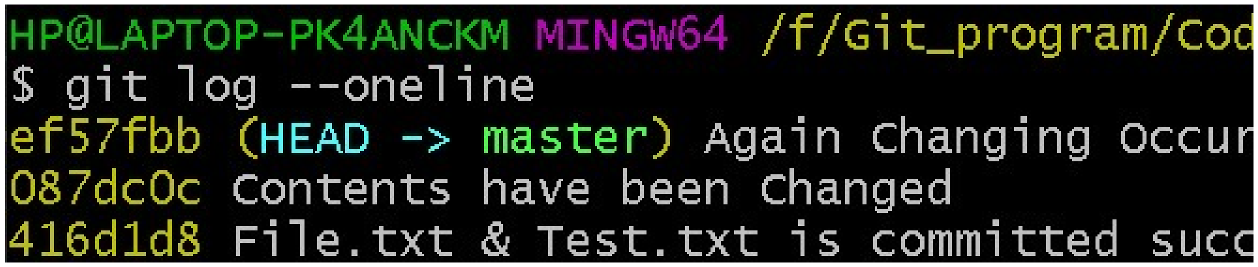
```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Cod
$ git log --oneline
4edc7d8 (HEAD -> master) file is created to b
a2fe6f7 Everything is Commited
26974f4 Checking Commit diff
ef57fbb Again Changing Occured
```

Performing Hard Reset~3, to DELETE first 3 Commit i.e (4edc7d8),(a2fe6f7), .. .



```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Cod
$ git reset --hard HEAD~3
HEAD is now at ef57fbb Again Changing Occured
```

HEAD Pointer changed it's position to 4th Commit :



```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Cod
$ git log --oneline
ef57fbb (HEAD -> master) Again Changing Occur
087dc0c Contents have been Changed
416d1d8 File.txt & Test.txt is committed succ
```

16. Delete Files :

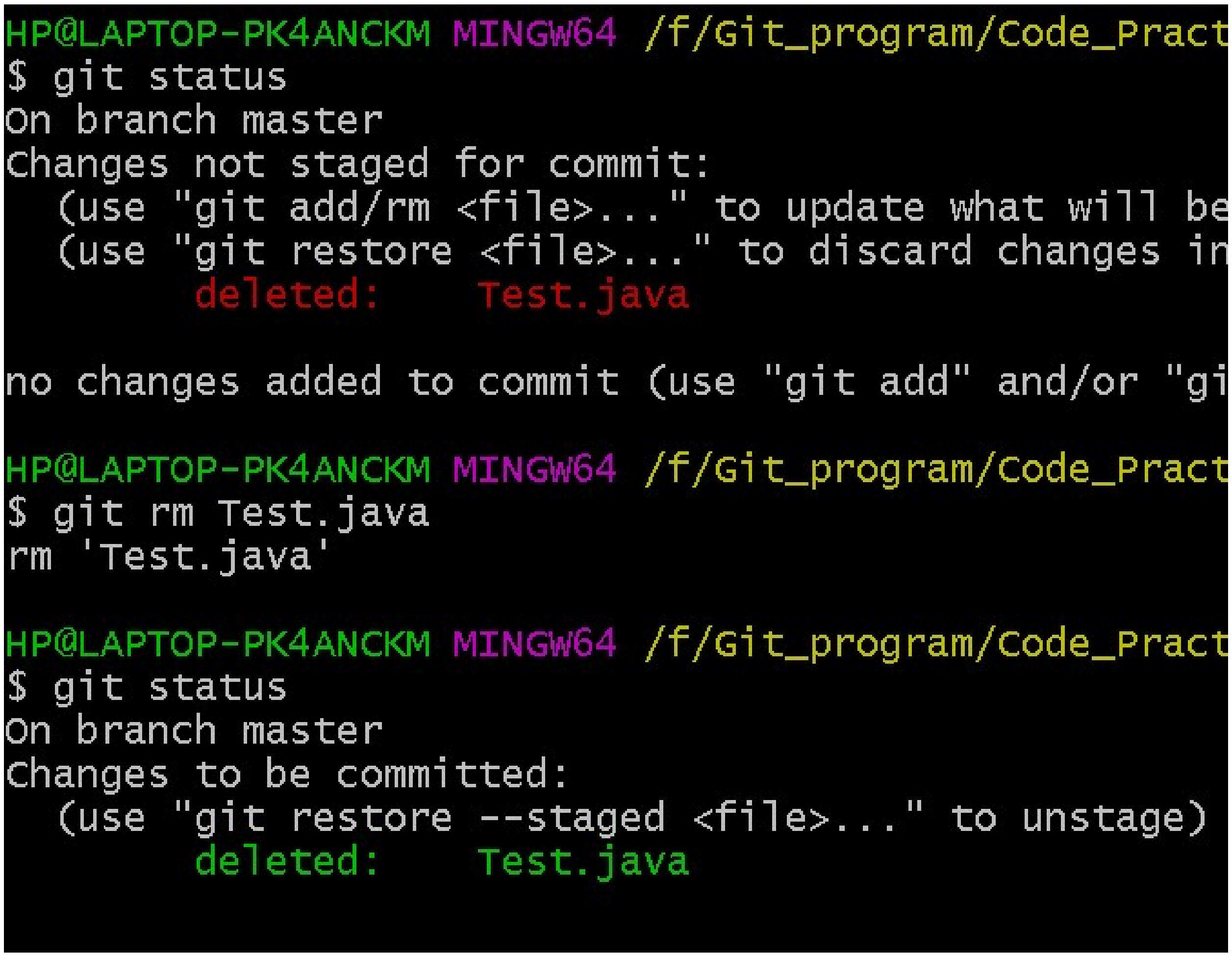
i) Untracked Files:

>> git clear -n ↩ (gives Warning first then)

>> git clear -f ↩ (Firstly Delete)

ii) Delete <file> from Folder which is committed earlier from Working Directory :

>> git rm <filename>

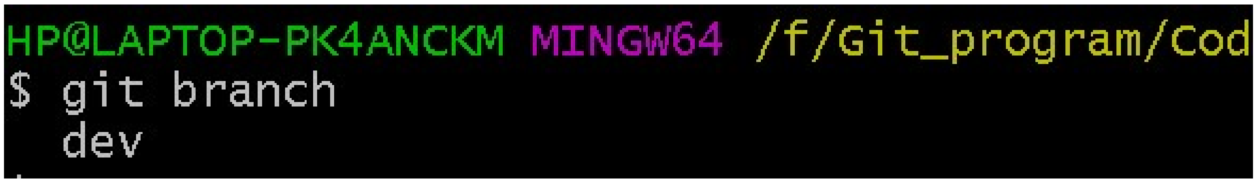


```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Code_Pract
$ git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be
  (use "git restore <file>..." to discard changes in
        deleted:    Test.java

no changes added to commit (use "git add" and/or "gi

HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Code_Pract
$ git rm Test.java
rm 'Test.java'

HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Code_Pract
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    Test.java
```

Branch

1. Check how many Branch are there..?

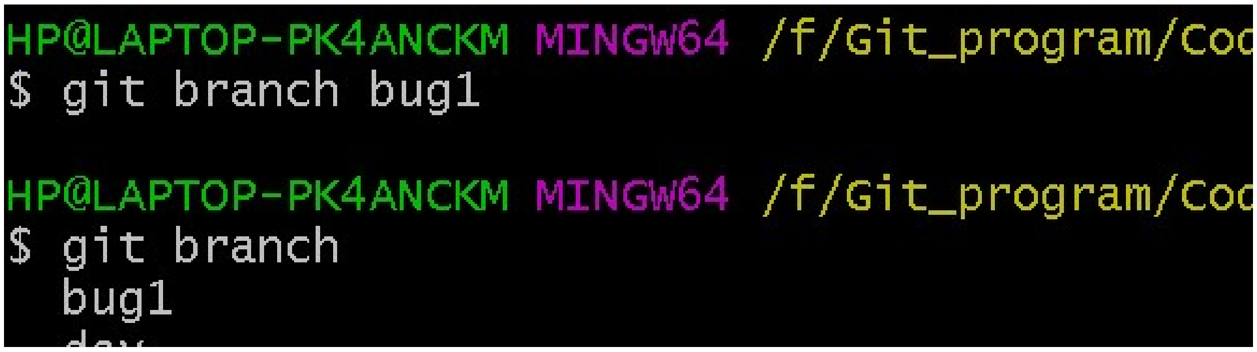
>> git branch



```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Cod
$ git branch
  dev
```

2. Creating a NEW Branch :

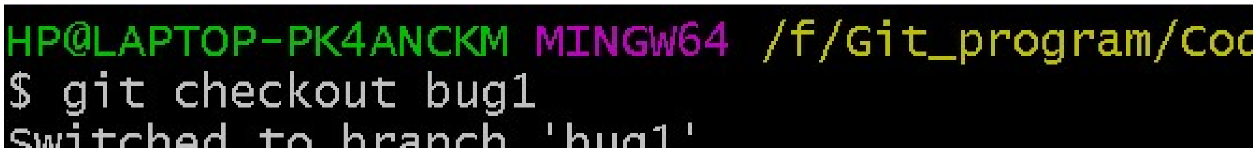
>> git branch branchName ↩



```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Coc
$ git branch bug1

HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Coc
$ git branch
  bug1
```

3. Switch to Another Branch :

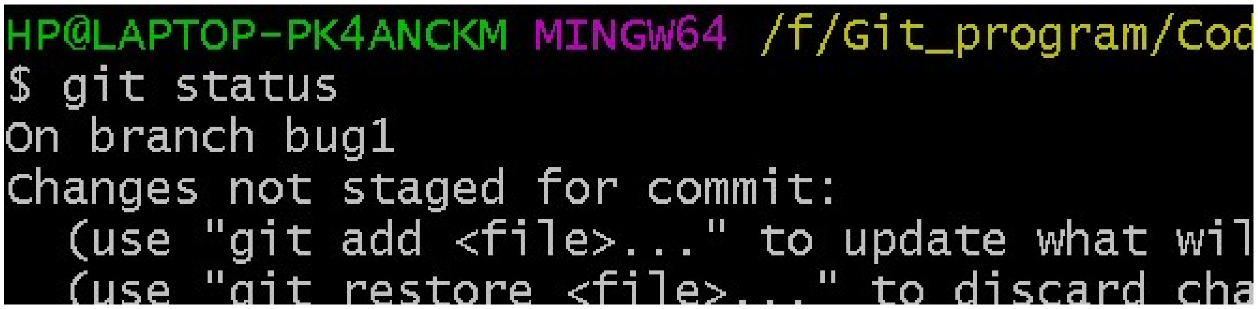
>> git checkout branchName



```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Coc
$ git checkout bug1
Switched to branch 'bug1'
```

4. To Check current position:

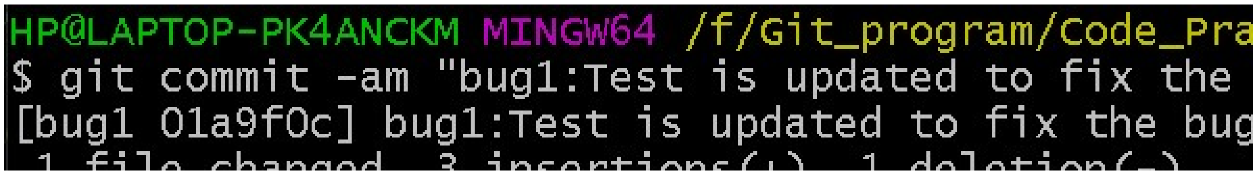
>> git status



```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Cod
$ git status
On branch bug1
Changes not staged for commit:
  (use "git add <file>..." to update what wil
  (use "git restore <file>..." to discard cha
```

5. To Staged & Commit the File Together :

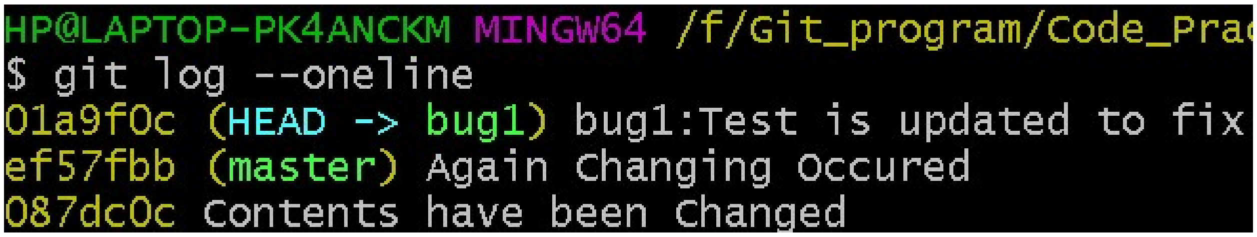
>> **git commit –am “message”**



```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Code_Pra
$ git commit -am "bug1:Test is updated to fix the
[bug1 01a9f0c] bug1:Test is updated to fix the bug
 1 file changed, 3 insertions(+), 1 deletion(-)
```

6. To Check History :

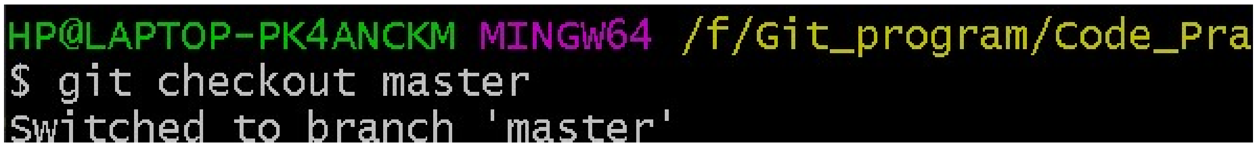
>> **git log --oneline**



```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Code_Prac
$ git log --oneline
01a9f0c (HEAD -> bug1) bug1:Test is updated to fix
ef57fbb (master) Again Changing Occured
087dc0c Contents have been Changed
```

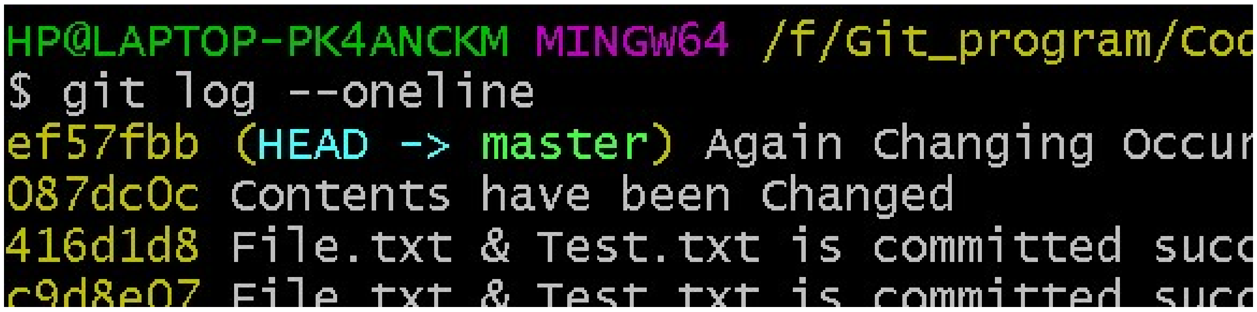
7. To MARGE Child Branch with Parent Branch :

i) We have to move the HEAD pointer to Parent Branch.



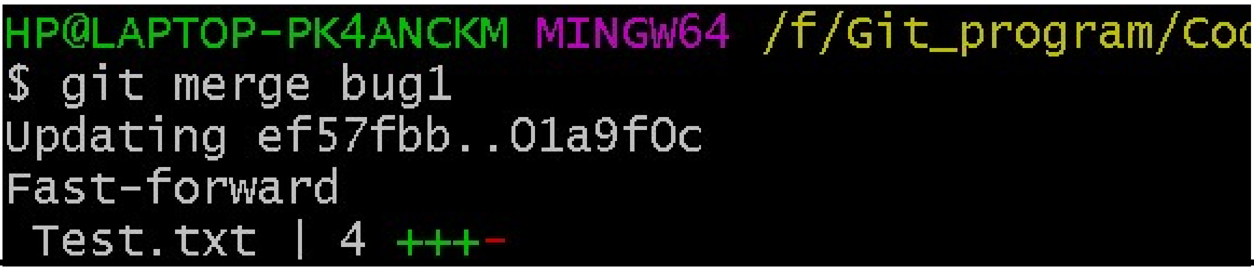
```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Code_Pra
$ git checkout master
Switched to branch 'master'
```

ii) Verify the Parent Branch is in Remote Repo Condition.



```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Coc
$ git log --oneline
ef57fbb (HEAD -> master) Again Changing Occur
087dc0c Contents have been Changed
416d1d8 File.txt & Test.txt is committed succ
c9d8e07 File.txt & Test.txt is committed succ
```

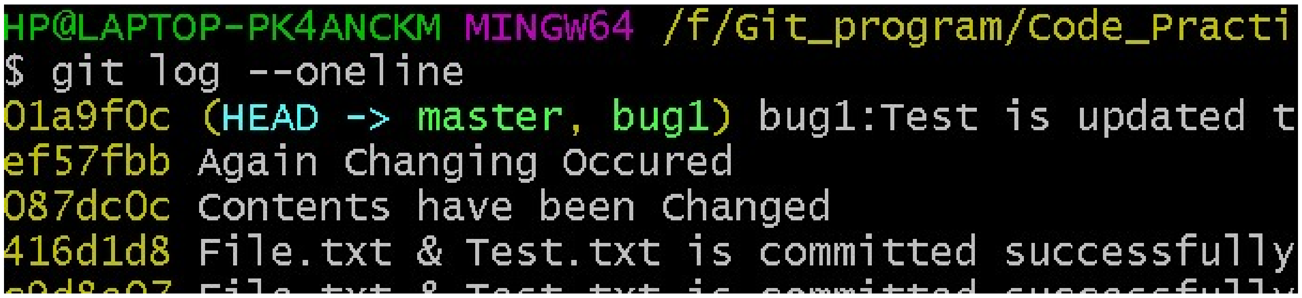
iii) Now MERGE with Parent Branch.



```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Coc
$ git merge bug1
Updating ef57fbb..01a9f0c
Fast-forward
 Test.txt | 4 +++-
```

8. To Check History AGAIN & see successfully Merged:

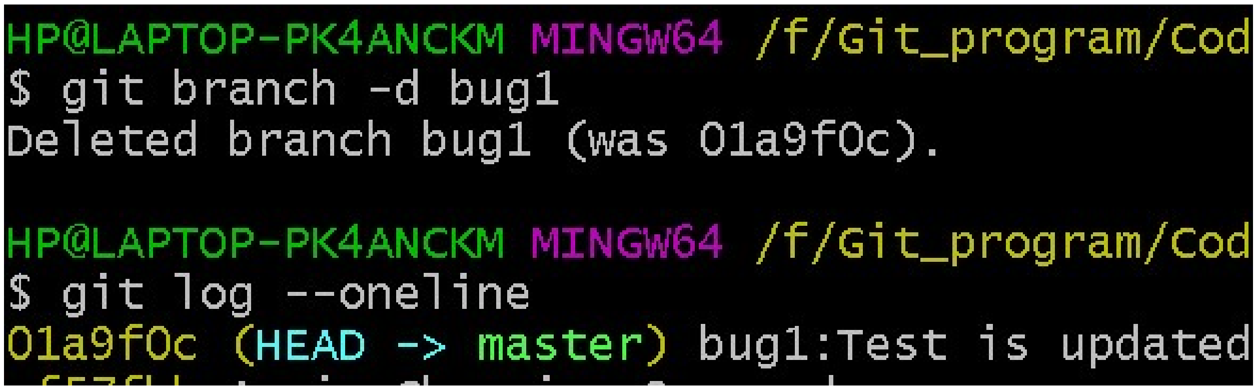
>> **git log --oneline**



```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Code_Practi
$ git log --oneline
01a9f0c (HEAD -> master, bug1) bug1:Test is updated t
ef57fbb Again Changing Occured
087dc0c Contents have been Changed
416d1d8 File.txt & Test.txt is committed successfully
```

9. Delete the Branch after Merging : (GOOD Practice)

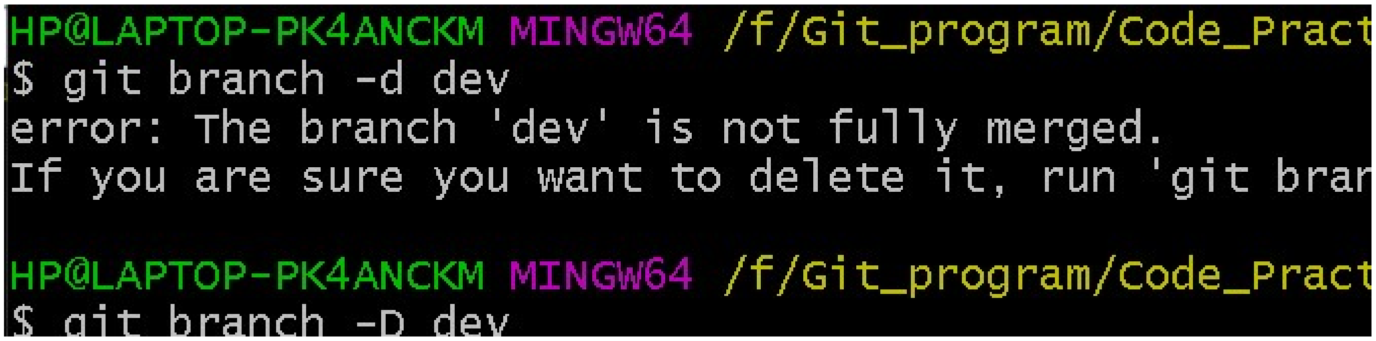
>> **git branch -d branchName**



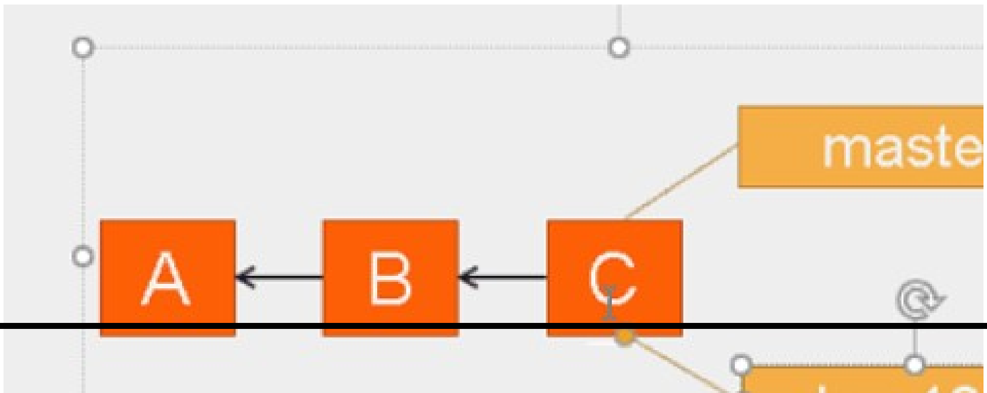
```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Cod
$ git branch -d bug1
Deleted branch bug1 (was 01a9f0c).

HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Cod
$ git log --oneline
01a9f0c (HEAD -> master) bug1:Test is updated
```

10. Delete a Branch Forcefully without Merged

>> **git branch -D branchName (Use Capital D)**

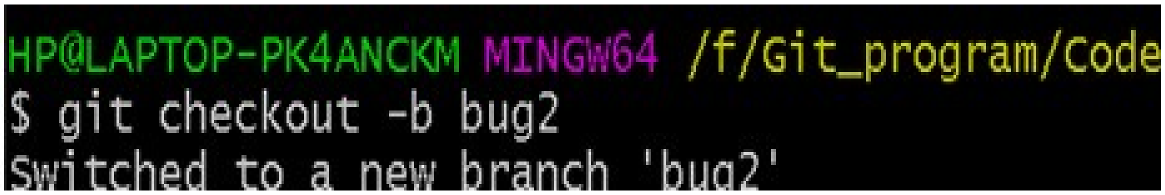


```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Code_Pract
$ git branch -d dev
error: The branch 'dev' is not fully merged.
If you are sure you want to delete it, run 'git bran

HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Code_Pract
$ git branch -D dev
```



11. Create a Branch + Switch in it :

>> **git checkout -b branchName**



```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Code
$ git checkout -b bug2
Switched to a new branch 'bug2'
```

12. Recover the deleted branch 'branchName' and rename it to 'newBranchName'

>> **git checkout -b branchName <SAH>** **(THEN)**

>> git branch -m branchName newBranchName

13. Rename a branch

>> **git branch -m <old> <new_branch_name>**

14. Differences between the current branch and the branch "new-email"

>> **git diff new-email**

Push

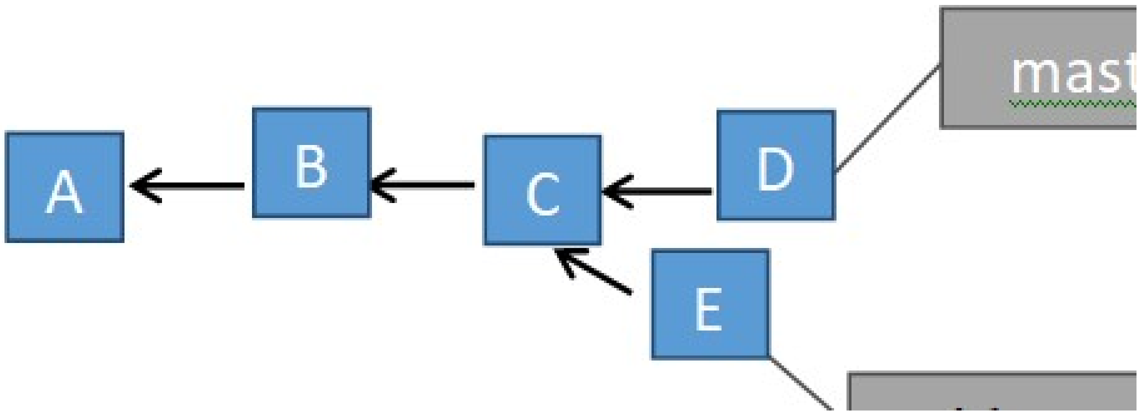
```
git push -u origin master
```

15. **Rebase Merge**

Q. Consider the following scenario and achieve the same to Rebase:-

-Create new branch ‘idea’ and make a new commit on ‘idea’ branch

- Now checkout to master and make a new commit on master



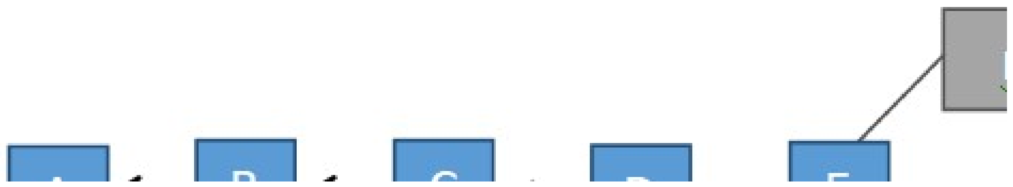
```
$ git checkout -b idea
Switched to a new branch 'idea'
```

```
$ git commit -am "idae:new change added:commit E"
[idea a0bb15a] idae:new change added:commit E
 1 file changed, 2 insertions(+)
```

```
$ git checkout master
Switched to branch 'master'
```

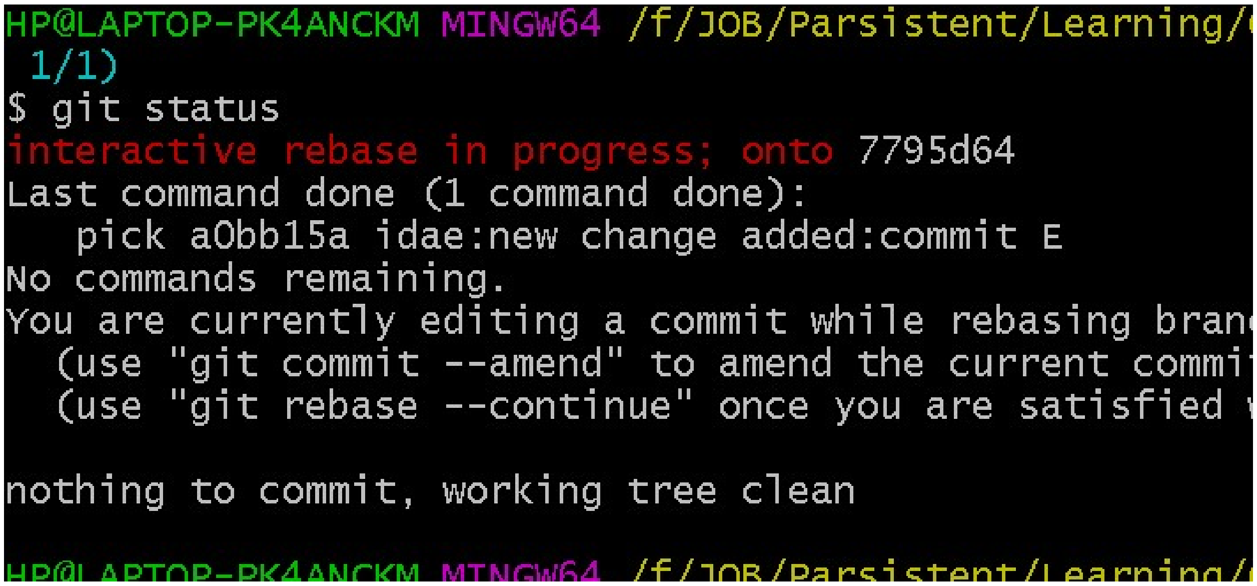
```
$ git commit -am "master:new change added:commit D"
[master 7795d64] master:new change added:commit D
 1 file changed, 2 insertions(+)
```

Now use Rebase to get the linear story line as follow



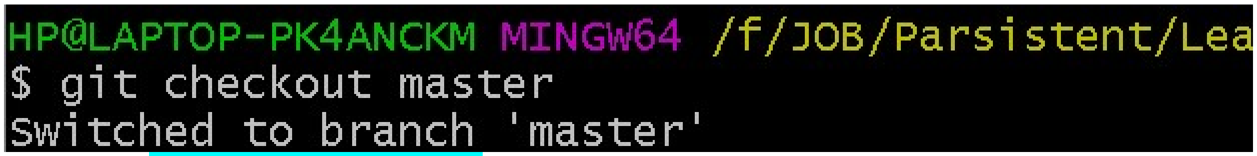
Steps: **$ git checkout idea**

$ git rebase master



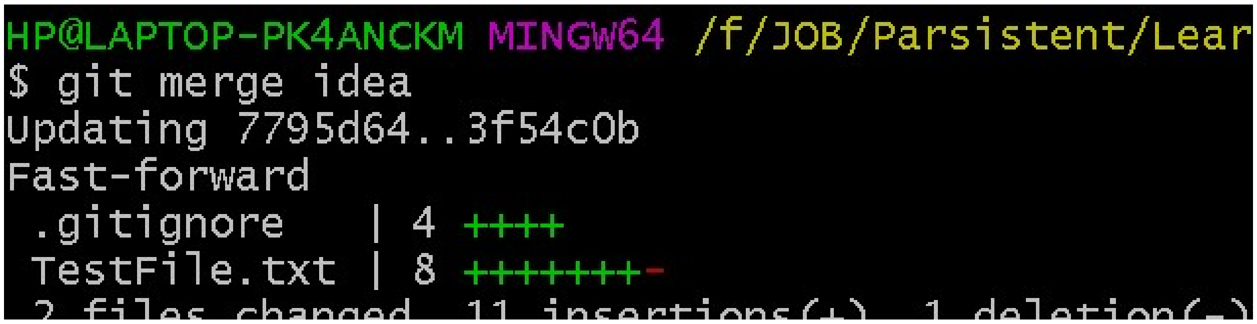
```
HP@LAPTOP-PK4ANCKM MINGW64 /f/JOB/Parsistent/Learning/(
 1/1)
$ git status
interactive rebase in progress; onto 7795d64
Last command done (1 command done):
   pick a0bb15a idae:new change added:commit E
No commands remaining.
You are currently editing a commit while rebasing bran
  (use "git commit --amend" to amend the current commi
  (use "git rebase --continue" once you are satisfied w

nothing to commit, working tree clean

HP@LAPTOP-PK4ANCKM MINGW64 /f/JOB/Parsistent/Learning/(
```

$ git checkout master



```
HP@LAPTOP-PK4ANCKM MINGW64 /f/JOB/Parsistent/Lea
$ git checkout master
Switched to branch 'master'
```

$ git merge idea



```
HP@LAPTOP-PK4ANCKM MINGW64 /f/JOB/Parsistent/Lear
$ git merge idea
Updating 7795d64..3f54c0b
Fast-forward
 .gitignore   | 4 ++++
 TestFile.txt | 8 +++++++-
 2 files changed, 11 insertions(+), 1 deletion(-)
```

$ git branch -d idea

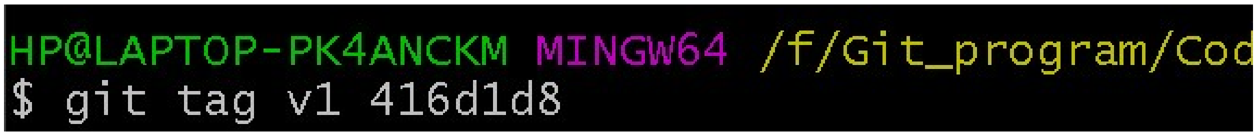
3 way Merging

`Check Day 2 Video at 2:30 hr

Git Tag

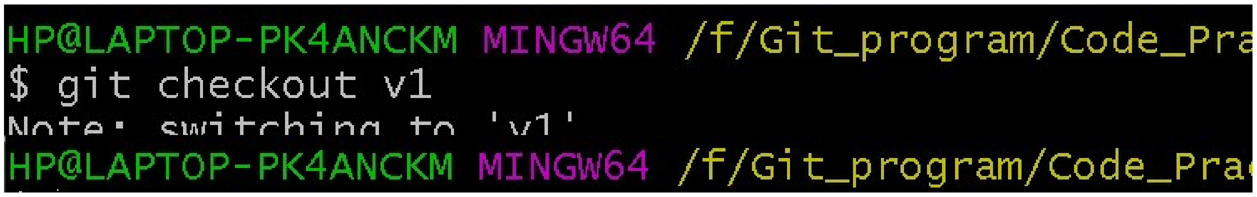
1. To give Tag name :

>> git tag tagName <SAH>



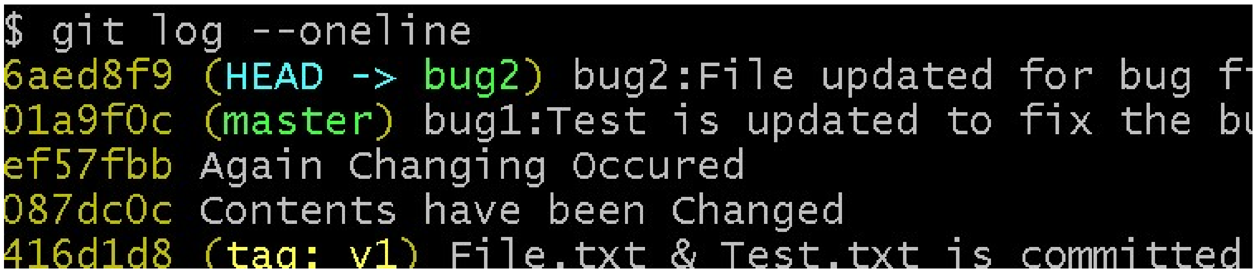
```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Cod
$ git tag v1 416d1d8
```

2. Tag Check Out:



```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Code_Pra
$ git checkout v1
Note: switching to 'v1'
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/Code_Pra
```

3. To check history :



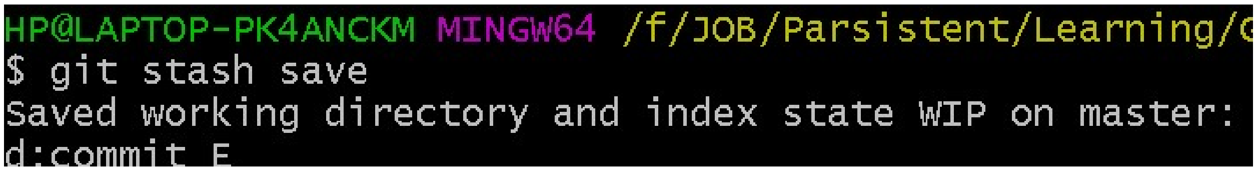
```
$ git log --oneline
6aed8f9 (HEAD -> bug2) bug2:File updated for bug f
01a9f0c (master) bug1:Test is updated to fix the bu
ef57fbb Again Changing Occured
087dc0c Contents have been Changed
416d1d8 (tag: v1) File.txt & Test.txt is committed
```

Git – Stash

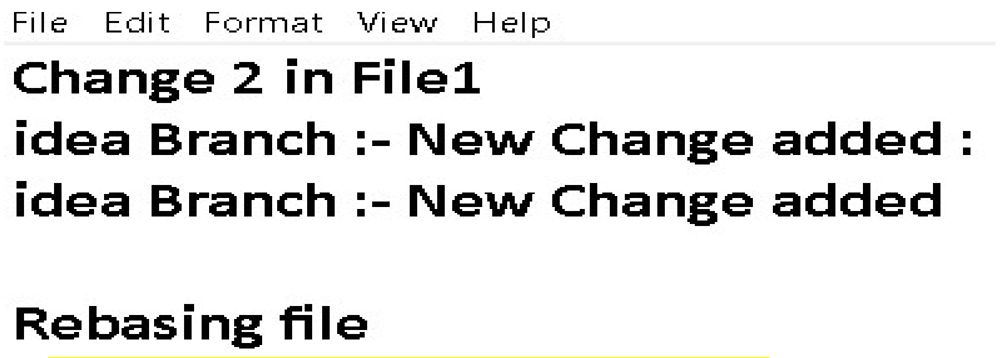
Git – Stash is very useful. It is needed only when you are working on a task but manager say you have to complete another task immediate-ely. So we have to switch your task from older one.

1. **To save my older task without commit : (Because you can't switch to other branch without commit)**

>> git stash save



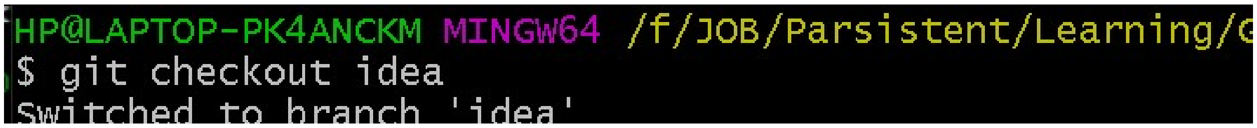
```
HP@LAPTOP-PK4ANCKM MINGW64 /f/JOB/Parsistent/Learning/G
$ git stash save
Saved working directory and index state WIP on master:
d:commit E
```



Change 2 in File1
idea Branch :- New Change added :
idea Branch :- New Change added

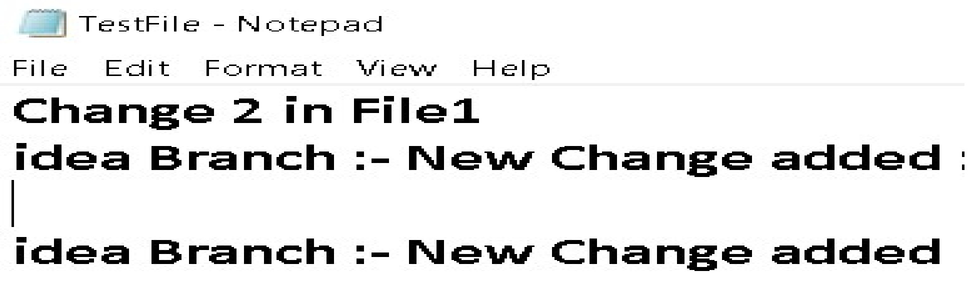
Rebasing file

2. **Now You Can Switch to other Branch :**



```
HP@LAPTOP-PK4ANCKM MINGW64 /f/JOB/Parsistent/Learning/G
$ git checkout idea
Switched to branch 'idea'
```

Now perform your new task, if you check the file in which you were previously work the texts are missing because you moved to another branch.

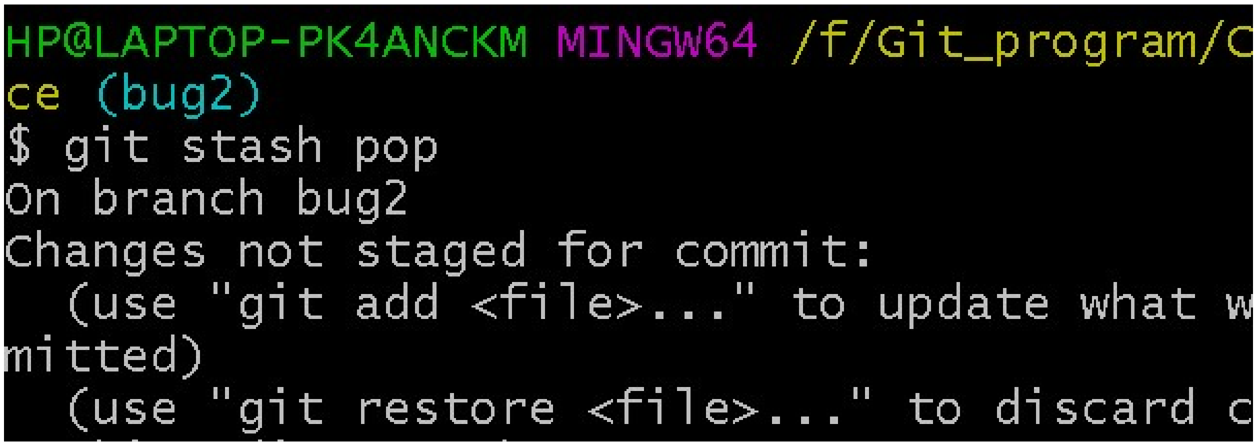


Change 2 in File1
idea Branch :- New Change added :

idea Branch :- New Change added

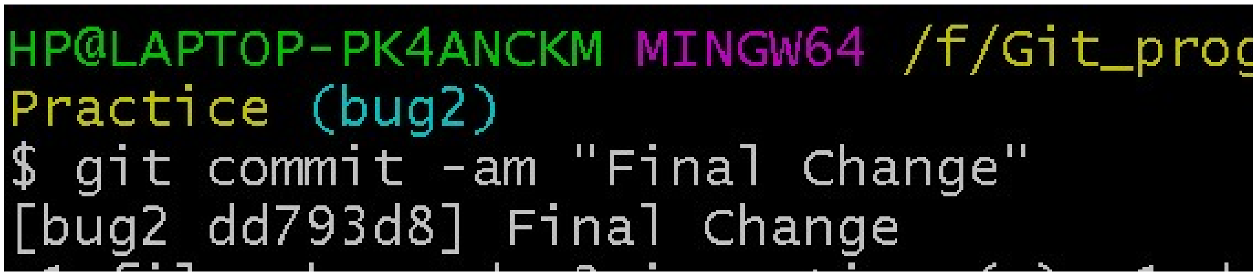
You can do that other task which your manage told to do
After that if you want to go with your own task then follow this steps below:

>> git stash pop (if you are sure you want your previous Edition)



```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_program/C
ce (bug2)
$ git stash pop
On branch bug2
Changes not staged for commit:
  (use "git add <file>..." to update what w
mitted)
  (use "git restore <file>..." to discard c
```

>> git commit –am “Final commit or anything” (To commit own work)



```
HP@LAPTOP-PK4ANCKM MINGW64 /f/Git_prog
Practice (bug2)
$ git commit -am "Final Change"
[bug2 dd793d8] Final Change
```

Which is the command to remove everything from the stash?
>> git stash clear

SHA - Secure Hash Algorithm - SHA-1
Fixed length hash = SHA(input data) --> data can be in bits/bytes/GBs/TBs

Properties SHA -

- Deterministic - For same input we always get same output, regardless of
  the OS, environment.
- Output should be fixed length.
- Avalanche Effect - For any minor change, entire hash value changes.
- Unique Value - Any 2 inputs should have same hash value

<u>Question & Answer</u>

1. What is the command to create a new branch named "new-email"?
   >> git branch new-email

2. Which one of these statements about a merge involving a merge commit is true?
   >> Git places the result of the merge into a new commit.

3. What does the git merge option -s abbreviate?
   >> --strategy

4. What is the option, when moving to a branch, to create the branch it if it does not exist?
   >> -b

5. What is the command to move to the branch named "new-email"?

>> git checkout new-email

6. What is a request to merge your branch into another branch called?

>> Pull request

7. When should you avoid rebasing a branch?

>> If you have shared the branch

8. In Git, a branch is?

>> A separate version of the main repository

9. What is the command to merge the current branch with the branch "new-email"?

>> git merge new-email

10. What is the command to show the differences between the current branch and the branch "new-email"?

>> git diff new-email

11. Which of these two statements makes git use the file from the conflicting commit you are merging from?

>> git checkout --theirs index.html

12. Find the command which is used to rename a branch

>> git branch -m old_branch_name new_branch_name

13. Which is correct about git checkout -b test_branch

>> Creates a new branch test_branch and switches to the test_branch

14. True or False?.git branch -D branch_name is used to delete the branch

>> True

15. Which is incorrect regarding branches.

>> Branches cannot be merged.

16. If --list is given, or if there are no non-option arguments, existing branches are listed

>> True

17. True or false? It is not possible to delete a branch once created

>> False

18. Which is true regarding Git fails to start the merge?

>> All of the listed options

19. git config --global merge.tool is used to set merge tool on project level

>> True

20. Which is true regarding git diff.

>> git diff is a multi-use Git command that when executed runs a diff function on Git data sources.

21. .git pull runs git fetch in the background and then git merge to merge the retrieved branch heads into current branch.

>> True

22. Which one of the following is not part of the data structure of a Git repository?

>> Body element

23. Consider if you as an individual working in an organization wants to see information regarding the commit or you Need to see who else checked in specific files, in that case, what command you would use to sort it out.
>> gitk

24. What's the opposite of git clone, instead of downloading your code from GitHub, uploads your changes and code back to GitHub?
>> git push

25. GitHub is the same as Git.
>> FALSE

26. Which of these terms best describes GitHub?
>> Web-Based Repository Hosting Service

27. How do you create a copy of a lab under your own GitHub account so that you can solve the lab?
>> Forking it via the GitHub interface

28. What's the git command that downloads your repository from GitHub to your computer?
>> git clone