Git Documentation

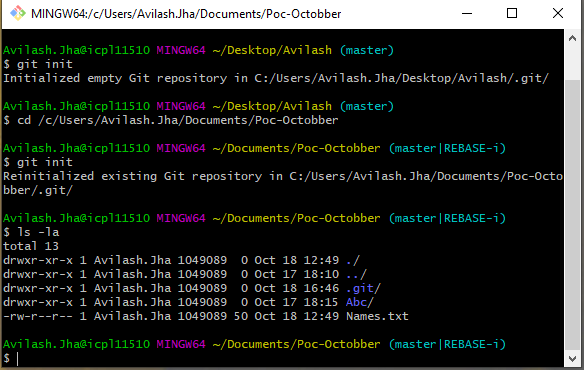
1. **git init** : It is used to create a new git repository or initialize a new empty repository. It create a .git sub-directory in current working directory. It contain metadata of git

Command:

git init : To initialize empty repositories or existing repository

ls -la : To check whether directory has .git file

Screenshot:

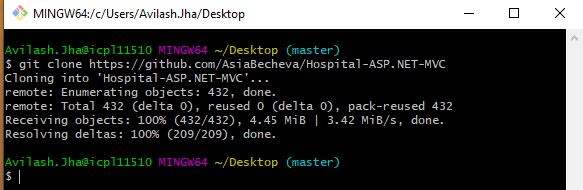


1. **git clone :** It isused to make copy of remote repository into local repository

Command:

git clone <url of repository>

Screenshot:



1. **git config** : It is used to set configuration at global or local project level. These configuration is saved in .gitconfigfile test file. It contain email and username.

Command :

git config –-global user.name “avilashj34” :- Set username

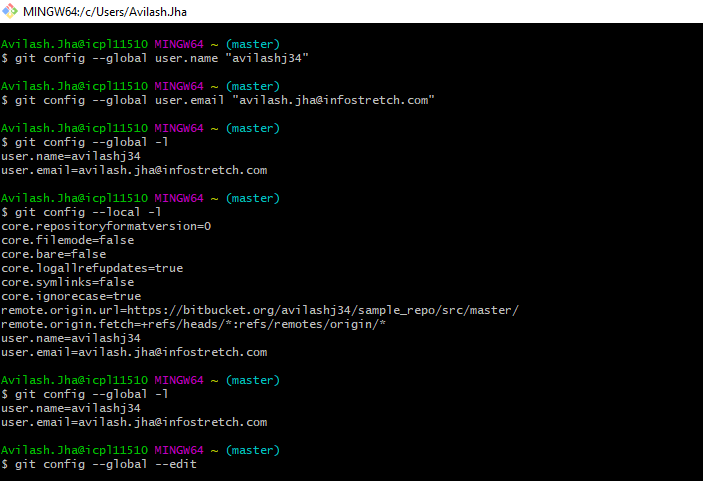
git config –-global user.email “[avilash.jha@infostretch.com](mailto:avilash.jha@infostretch.com)” : Set email

git config –-local -l :- Local User Info

git config –-global -l : Remote user Info

git config –-global –edit : Edit the username and email in Vim editor

Screenshot:



1. **git alias :-** It is used to create short-cut name of git command. It will not change the source command. All the alias will get save in .gitconfig file

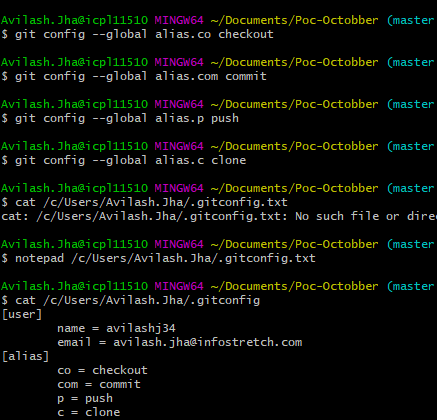
Command:-

git config –-global alias.<alias-name> <source-command-name> : To create alias of source command

cat <path of .gitconfig file> : To view data in command itself

Screenshot:-

The example of alias is attached in below screenshot . To view the alias use cat command followed by url of .gitconfig file



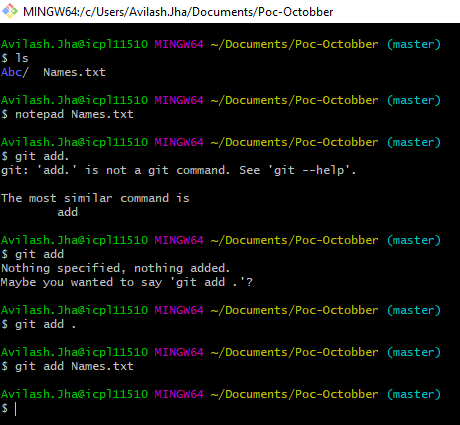
1. **git add :-** It is used to add file into staging area.

Command :-

git add . :- To add all file to staging area

git add <file-name> :- To add particular file to staging area.

Screenshot:



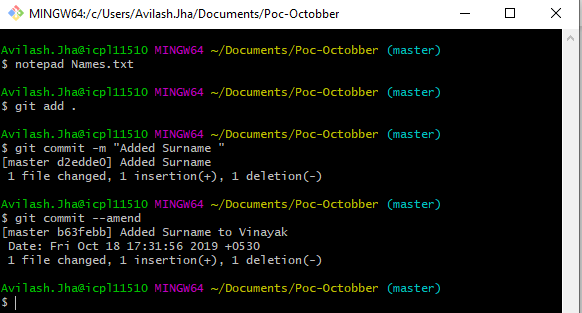
1. **git commit :-** It is used to save changes into local repository.

Command:

git commit --amend :- To edit last commit message

git commit -m “<commit-message>” :- To make commit to

Screenshot:



1. **git remote :-** It is used to perform operation on remote repository. To add new remote we use remote add command

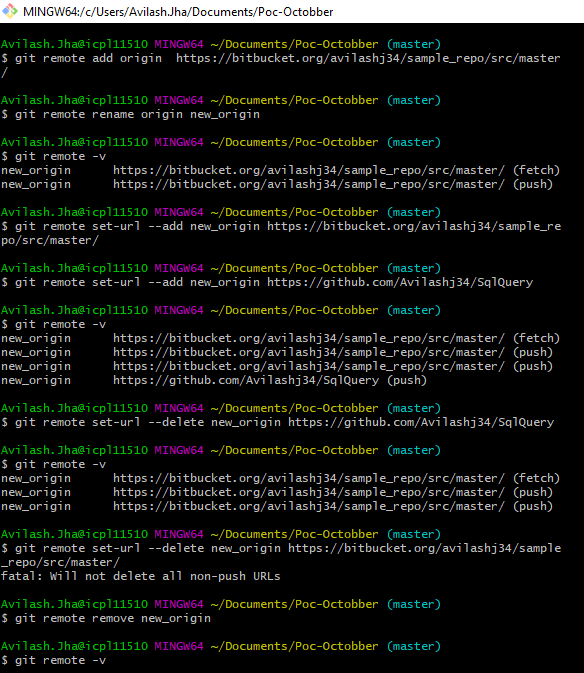
Command :-

git remote -v :-

git remote add <origin-name> <remote repo-url>

git remote rename <old-name> <new-name>

Screenshot:-



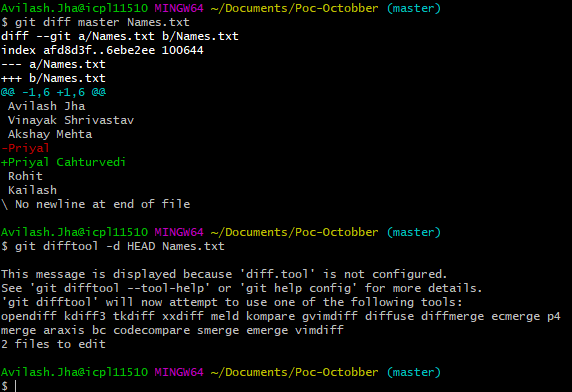
1. **git diff :-** It is used to see the changes between the file on local.

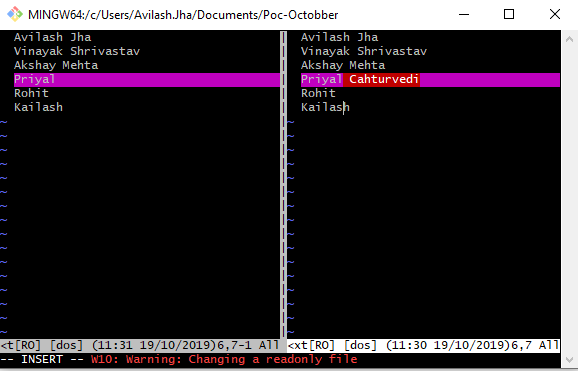
Command :-

git diff <file-name> :- This command is used to check the local difference made in the present file.

git difftool -d HEAD <file-name> : This command show comparison using VI slit window in terminal.

Screenshot:-



****

1. **git reset:-** It is used to undo local changes to the staging index and working directory. Reset is of four types –soft, --head, --mixed, --merge. It revert all the changes back to specific commit.

Command :-

git ls-files -s

git reset --soft HEAD^ :- This command is used to move head but doesn’t touch the staging area

git reset --hard HEAD^ :- This command adjust staging area and working tree to new HEAD and throw everything

git reset HEAD^ : This command use to undo to previous commit.

***To Solve Merge Conflict :-***

when merge conflict arises use this command to see diff in VIM :- git mergetool

Result will be in these format

|  |  |  |
| --- | --- | --- |
| LOCAL | BASE | REMOTE |
| MERGED | | |

LOCAL:- This is file from current branch

BASE :- This is how file look before both changes

REMOTE :- File you are merging into your branch

MERGED :- Merge result, This is what get saved in the repository

:diffg RE :- Get change from remote

:diffg BA :- Get change from base

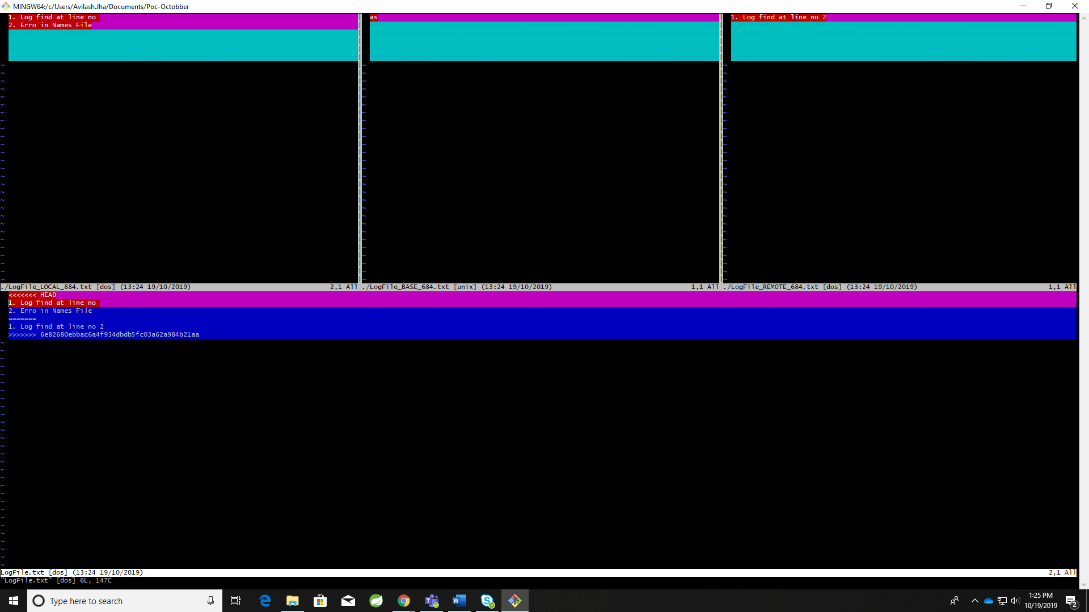
:diffg LO :- Get change from local

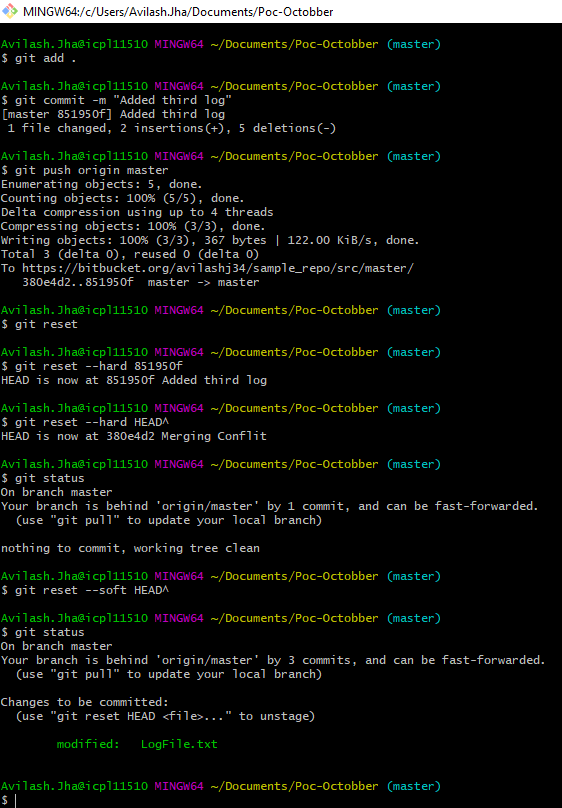
To save and exit from VI use :wqa

git commit -m “Message”

git clean :- To remove extra files (ex :- \*.orig)

Screenshot:-





1. **git rm :** This command is used to delete file from local as well as from remote repository

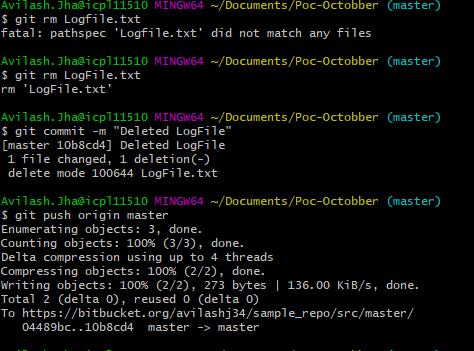
Command :-

1. git rm <File-Name>
2. git commit -m “<Commit- Message>”
3. git push <remote-name> <branch-name>

To remove file remotely use this command

1. git rm –cached <File-Name>
2. git commit -m “<Commit- Message>”
3. git push <remote-name> <branch-name>

Screenshot:-



1. **git log** :- It show history of all changes made to local and remote repository

Command :-

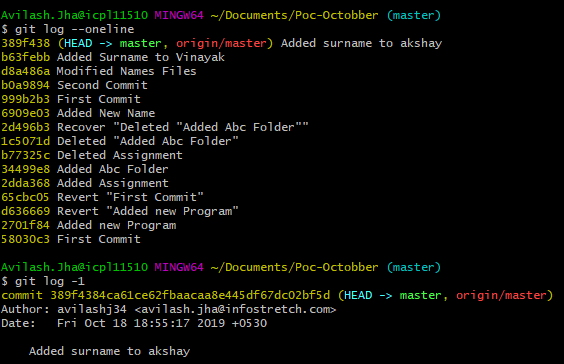
git log :-- It show all record of commited

git log :–oneline : It show history in oneline

git log -2 :- It show history of last 2 commited

git log --pretty=oneline –decorate :- Show all log with hash code

Screenshot:-

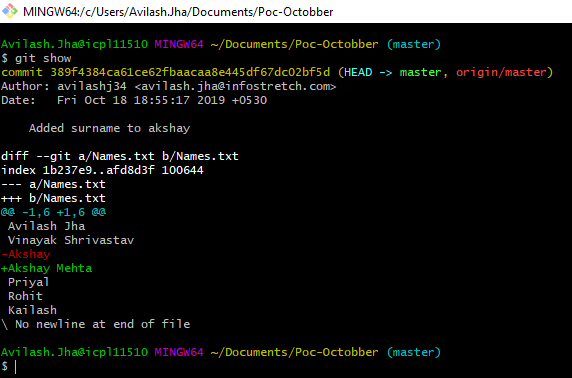


1. **git show :-** It show log message of last commit and what changes we made on that file(It mask red and green for clear representation).

Command :-

git show

Screenshot:-



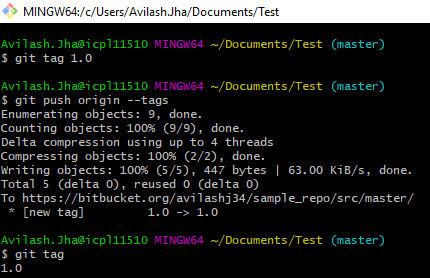
1. **git tag :-** It is used to capture a point in history used to mark version release. Tags are of two type lightweight and annotated.

Command :-

git tag <tag-name> :- Used to set tag

git tags :- List of all tag

Screenshot:-



1. **git branch :-** It is used to create branch. It represent independent line of development instead of copying from directory to directory git store branch as reference to commit

Command :-

git branch :- Show list of branch

git branch <branch-name> :- Create new branch

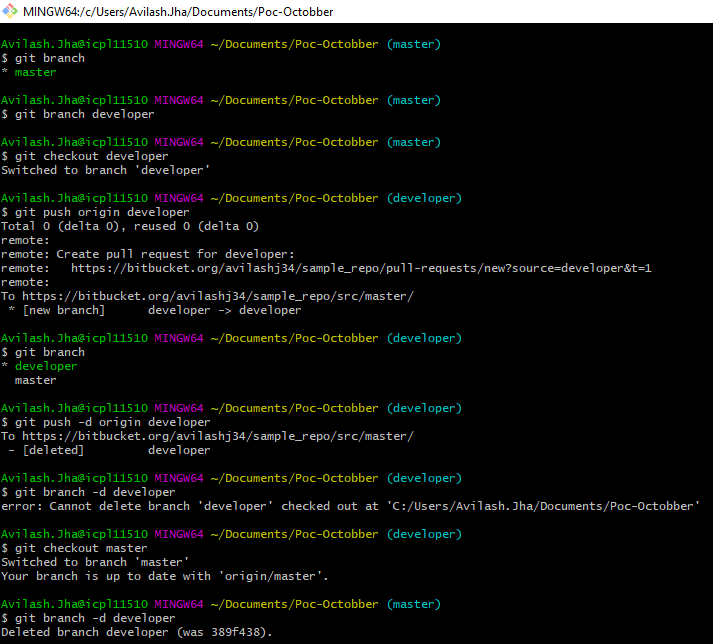
git checkout <branch-name>:- Switch to that branch

git push <remote-name> <branch-name> :- Push branch from local to remote i.e It create branch at remote repository

git push -d <remote-name> <branch-name> :- Delete branch from remote repository

git branch -d <branch-name> :- Delete branch from local repository

Screenshot:-

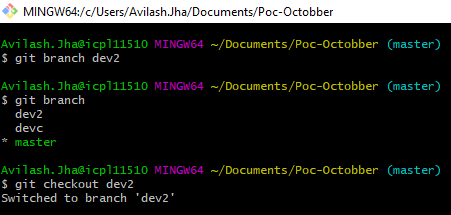


1. **git checkout :-** It is used to update working tree also used to switch branch.

Command :-

git checkout <branch-name>

Screenshot:-

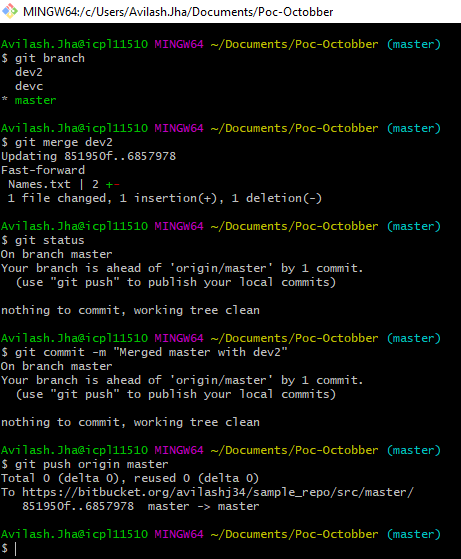


1. **git merge :-** It is used to inherit changes from another branch.

Command :-

git merge <branch-name>

Screenshot:-



1. **git push :-** It is used to push commited file from staging area to remote repository

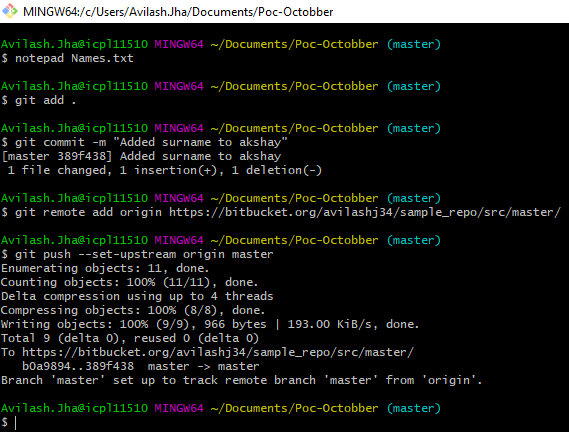
Command :-

git push -u origin <branch-name>

git push -f <name> <branch-name>

git push –set-upstream <name> <branch-name>

Screenshot:-



1. **git pull :-** It fetch and downld content from remote repository and update local repository. It is combination of two command git fetch and git merge. First it fetch all the content and then update it with current checkout branch(local version of repository).

Command :-

git pull <remote-repo>

1. **git stash:-** It is used to temporary stashes changes we made in local repository. When needed we can use this changes to commit to remote repository.

Command :-

git stash save “message” :- Used to save message temporary

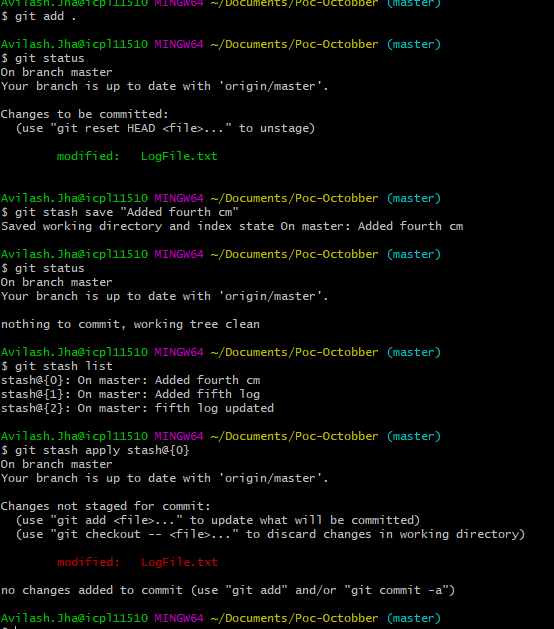
git stash list :- Used to view list of stash

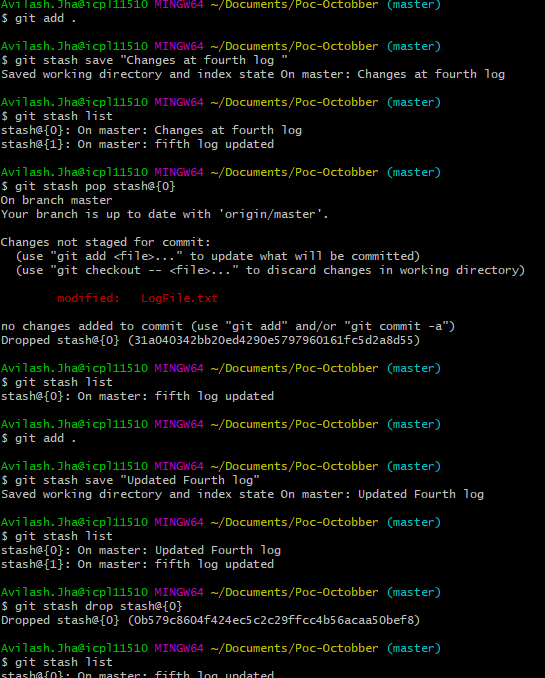
git stash apply <hashcode> :- Apply changes from stashes list.

git stash pop <hashcode> **OR** git stash pop :- apply the changes of choosed stash and drop it from stash list.

git stash drop <hashcode> :- Used to remove particular stash from stash list

Screenshot:-



****

1. **git rebase :-** It is used to integrate changes from one branch to another branch. Rewrite the project history by creating new commit.

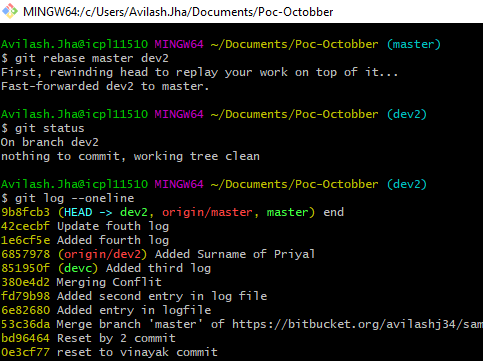
Command :-

git pull --rebase <remote-name> <branch-name>

git rebase <branch-name1> <branch-name2>

git rebase -i origin master

Screenshot:-

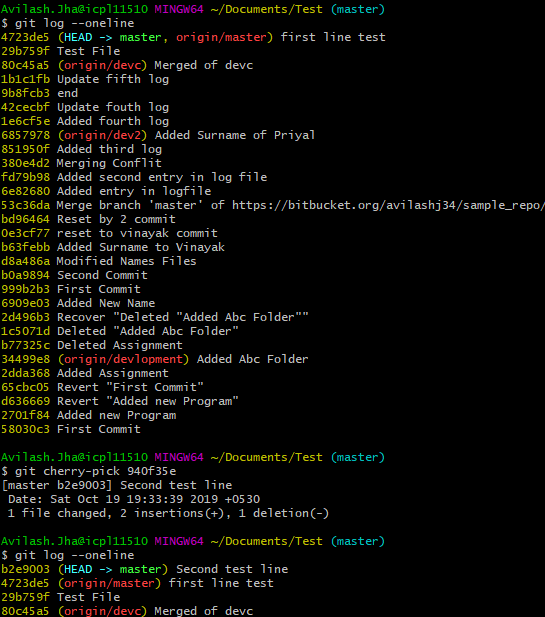


1. **git cherrypick :-** Instead of merging all commit from another branch it merge only particular commit from another branch.

Command :-

git cherry-pick <hash-code> : Copy commit message of that hashcode

Screenshot:-



1. **git revert :-** It is used for undoing changes of remote repository. It create a new commit that is opposite of an existing commit.

Command :

git revert <hash-code>

Screenshot :

