* First create a github account then you get username and password and create a git repository(helloworld) in github account .
* Now install git in system through terminal(command line )
  + - sudo apt-get install git
* Now initialise git in system

$ git init

* Now configure git repository in local (in system)

$ git config --global user.name “username”

* + $git config –global user.name “Akbar Saleem”

$git config –global user.email [email@xxx.com](mailto:email@xxx.com)

$git config –global user.email akbarsaleem402@gmail.com

* now clone our git repository to local (in pc) by using our git repository link

$ git clone <https://github.com/akbarsaleemt/helloworld.git>

* now move to that local repository

$cd helloworld

* copy the files which we want to upload to git into this helloworld folder
* now create a branch
* $ git branch branchname
* $git branch hello
* now move to the particular branch
* $ git checkout branchname
* $git checkout hello
* now add files to git by using following command

$ git add . (. means all files will added ti git else give file name instead of .)

* now commit the details of files loaded

$ git commit -m “commit message added herein quotes”

* now push the content (files added) to branch

$git push origin -u branchname

$git push origin -u hello

* now merge branch to master .movr from branch to master

$ git merge branchname

$git merge hello

$git push -u origin(now the branch reflected to master)

* Deleting branch

first move out from that branch to another branch or master branch.

$git branch -d branchname

$git branch -d hello

* If we modify any file in our repositry on server then to reflect it in local we pull it.

Move to that changed branch and pull there

$git checkout branchname

$git pull

* git diff
* git log(Tom performs the clone operation and finds a new file string.c. He wants to know who added this file to the repository and for what purpose, so, he executes the **git log** command.)
* git status
* git tags
* git revert
* git reset --hard

git revert [--[no-]edit] [-n] [-m parent-number] [-s] [-S[<keyid>]] <commit>…​

git revert --continue

git revert --quit

git revert --abort

Given one or more existing commits, revert the changes that the related patches introduce, and record some new commits that record them. This requires your working tree to be clean (no modifications from the HEAD commit).

Note: git revert is used to record some new commits to reverse the effect of some earlier commits (often only a faulty one). If you want to throw away all uncommitted changes in your working directory, you should see [git-reset[1]](https://git-scm.com/docs/git-reset), particularly the --hard option. If you want to extract specific files as they were in another commit, you should see [git-checkout[1]](https://git-scm.com/docs/git-checkout), specifically the git checkout <commit> -- <filename> syntax. Take care with these alternatives as both will discard uncommitted changes in your working directory.

* git status
* git log --oneline
* git reset
* git revert commitid(resets previous content of files)