Git tuts:

1. git status: to get the status of current git repo
2. git init: to create a new git repo
3. git add --a: changes would be come at staging area now the changes are ready to commit
4. git commit –m “any message”
5. git log: to know which commits we have done

now, suppose we edit **two** file which we commit previously.

1. git status
2. git add first.txt: we add first file only to staging area, and it is ready to commit now
3. git commit –m “changed first.txt only”
4. **rm –rf .git: delete a git repo**
5. pwd : present working directory, in which directory we are working on
6. ls: list the content in present working directory
7. cd: change directory, to go in a specific folder
8. to ignore a file

* touch .gitignore: write the name of file in ‘.gitignore’ which we want to ignore[if we want to ignore all log files then we have to write ‘\*.log’ in ‘.gitignore’ file.
* Now git status, git add --a, git commit –m “message”, git status

1. Compare working directory and staging area
2. git commit –a –m “message”: staged and commit all untracked files or we can skip staging area using this command
3. git restore --staged -- copy.txt: //unstage a staged file
4. git restore copy.txt: //send the untracked file to last commit
5. git rm filename: //delete the file (for ex delete.txt) and auomatically stage it we only need to run commit
6. git mv from.txt to.txt : rename a file automatically add to the staging are we need to commit it only.
7. Suppose we have a file name rename.txt, we add it in .gitignore and commit the .gitignore now we make some changes rename.txt and run ‘git status’ technically rename.txt should not show any changes to commit,

But it ask us to stage the file, this is happen because we need to remove rename.txt from the git tracking system

So we have to use ‘**git rm --cached filename.txt**’ and commit it.

1. ‘git checkout -- notepad.txt’ to revert the changes in ‘notepad.txt’ file, it will work before commit(i.e in staging or untracked phase).
2. ‘git checkout –f’ to revert changes in all files, it will work before commit(i.e in staging or not staged/untracked phase).
3. **PUSH IN GIT**
4. git remote add origin <https://github.com/ashishsingh17034/gittuts.git>: (gittuts is the name of repo and origin refer to ‘<https://github.com/ashishsingh17034/gittuts.git>’ we can change origin to an other name)
5. $ git remote: it gives us ‘origin’ because we add the name of repo as origin
6. git remote –v: it give the output as

origin https://github.com/ashishsingh17034/gittuts.git (fetch)

origin https://github.com/ashishsingh17034/gittuts.git (push)

1. $ git push -u origin master:(It push our code to repo)
2. Git push origin branchName: (It push our code to repo)
3. Git pull origin branchName: (It pull our code in repo, if we work at night and someone work on that branch in day so we have to take pull of that branch in our local system to get updated with the changes)
4. git checkout -b developashish: //**Create a new branch in git(branch name is developashish)**
5. git checkout master: //switch branch to the master branch
6. git branch: // to show all our branches
7. git merge developashish: //we are on ‘master’ branch and we want to merege the changes of ‘developashish’ branch to ‘master’ branch **we can do it from frontend**]
8. git clone <https://github.com/ashishsingh17034/gittuts.git> ://to clone a repo
9. git branch –d branchname : delete a branch in bash not in remote
10. git push branch –d branchname: delete a branch from remote