**1. Install Git :**

Windows : *git bash : https://gitforwindows.org/*

Ubuntu : *sudo apt-get install git-core*

RedHat : *sudo yum install git-core*

**2. Config Set :**

Set a Git username:

*git config --global user.name "Ethans Institute"*

*git config --global user.email "deepak.kumar@ethans.com"*

*git config --list`*

**3. Need HELP?**

Open the help page in browser or in console

*git help <verb>*

*git <verb> --help*

**4. Initialize a repository from existing code.**

*mkdir LearnGit*

*cd LearnGit*

*git init* (will create .git directory which makes the LearnGit directory a tracked Repo)

*touch .project*

*touch test.pyc*

*touch calc.py*

*git status* (will show the untracked file before git commit)

*touch .gitignore* (to block some files in repo to not get commited to remote repo)

*vi .gitignore* (and add the files to be ignored)

.project

\*.pyc

*git status* (will not show the files included in .gitignore)

**5. Basic WorkFlow**

Working Dir Staging Area .git Dir(Repo)

| | |

|<---------------------Checkout the Project-----------------------|

| | |

|---------------Stage Fixes---------->| |

| | |

| |------------Commit------->|

Add File to staging area

*git add .gitignore*

*git add calc.py*

Remove File to staging Area

*git reset calc.py* (bring back calc.py to untracked stage)

*git reset* (will bring all staged file to untracked)

**6. Our First Commit**

*git add -A* (add All file to staging area)

*git commit -m "Initial Commit"*

*git status*

*git log*

**7. Cloning a Remote Repo**

*mkdir RemoteRepo*

*cd RemoteRepo*

*git clone git@github.com:deepak2717/MyScripts.git .*

**8. Viewing Information about remote repository**

*git remote -v*

---------Note-----------

git pull does a git fetch followed by a git merge

You can do a git fetch at any time to update your remote-tracking branches under refs/remotes/<remote>/.

This operation never changes any of your own local branches under refs/heads, and is safe to do without changing your working copy

A git pull is what you would do to bring a local branch up-to-date with its remote version, while also updating your other remote-tracking branches

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

*git branch -a*

**9. create a .gitignore and calc.py in the cloned repo.**

*touch .gitignore*

*touch calc.py*

write a code in calc.py

def add(x,y):

pass

def subtract(x,y):

pass

def multiply(x,y):

pass

def divide(x,y):

pass

def square(x,y):

pass

*git add -A*

make change in the calc.py

def multiply(x,y):

return x\*y

*git diff* (show the changes made in the files)

*git status* (show the modified state)

git commit -m "Added calc.py and worked on Mulitiply Function"

git status

git pull origin master

*git add -A* (all the chnages in file)

**10. Pushing the changes**

we should always pull first then push when there is parallel development is going on

*git pull origin master*

From github.com:deepak2717/MyScripts

\* branch master -> FETCH\_HEAD

Already up to date.

*git remote add origin git@github.com:deepak2717/MyScripts.git* (its get added when clone is done, if remote is not added we can use this command)

*git push -u origin master*

Counting objects: 4, done.

Delta compression using up to 4 threads.

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

Writing objects: 100% (4/4), 407 bytes | 203.00 KiB/s, done.

Total 4 (delta 1), reused 0 (delta 0)

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

To github.com:deepak2717/MyScripts.git

7257f74..6c795bb master -> master

Branch 'master' set up to track remote branch 'master' from 'origin'.

**11. Common Work Flow :**

a) Create a branch from desired feature

*git branch calc-divide* (creates the branch from master)

*git branch* (check the branch)

*git checkout calc-divide* ( move from master to the created branch)

b) Make changes to the calc.py in divide function

def divide(x,y):

return x/y

*git status* ( show the modified state in branch)

*git add -A*

*git commit -m “Divide Function”* ( only changed in the local branch, it has no effect in the local master and remote repository)

c) After commit push branch to remote

*git push -u origin calc-divide*

*git branch -a*

d) Merge a Branch

*git checkout master* ( switch to master as branch need to be merged with master)

*git pull origin master* (a good practice, to pull any new changes from remote to local, in our case we had not made any changes)

*git branch –merged* ( It shows with what all branched you have merged till now)

*git merge calc-divide*

*git push origin master*

**12. Deleting a Branch**

*git branch –merged* (to check the branch is merged with master, so we can delete the branch now)

*git branch -d calc-divide* (delete the branch calc-divide locally, not from the remote)

*git branch -a* (we can see the calc-divide branch is still in remote)

*git push origin –delete calc-divide* (delete the branch from remote)

**1. Git checkout to get the original state back :**

Create a branch subtract-feature

git branch subtract-feature

Made change to the subtract function being in master branch only and say you made the bad changes

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

def add(x,y):

pass

def subtract(x,y):

abcdefff

ghjkkkkkkss

def multiply(x,y):

pass

def divide(x,y):

pass

def square(x,y):

pass

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

git status

git diff ( shows the changes made)

git checkout calc.py

git status

git diff (no diffs means the file went to original state)

**2. Changing the wrong committed messages :**

Make a valid change to the subtract function

return x - y

*git status*

*git add -A*

*git commit -m “Completed multiply function” (A bad commit with wrong message)*

*git commit --amend -m “Completed subtract function” (using –amend to change the message)*

*git log*

**3. Git Cherry Picking**

Since we made changes of subtract in master but it was meant to be done on subtract feature. This is very common in day to day work

*git log ( we have to copy the first 6-7 character from the hash)*

*git checkout subtract-feature*

*git log*

*git chery-pick 1b818d3b (first 7 char. Of hash from the last commit on the master using git log)*

*git log*

**4. Git Reset**

Git reset are of three types : soft, mixed(default) and hard

*Let’s get back to master and reset it back to the state were we have not made the subtract feature changes in master*

*git log (and pick the hash of the commit older than the “completed subtract feature one”*

*git reset --soft 2e7520782 ( will take to stage where we see the new file if added and file in modified state)*

*git reset 2e7520782 ( it’s a mixed reset)*

*git reset --hard 2e7520782 ( be careful when using it, it will remove all the tracked file changes before commit for the passed hash id)*

**5. Getting rid of untracked files**

*git clean -df*  (will remove the untracked file)

git status

**6. Recovering from hard reset**

*git reflog* (add All file to staging area)

*git checkout <hash id> (will detatch the head till the hash id commit)*

*git branch backup (to create a backup branch from the head)*

*git branch (to see the branch created and currently its on detached head)*

*git checkout master ( return to master from detached head after taking the backup)*

*git checkout backup (go to branch backup)*

*git log*

**7. Git Revert**

*git reflog*

*git revert <hash-id> (the commit that you want to revert to without changing history)*

*git log (you will see a new revert hash created and old hashes are intact)*

git diff <old-hash> <new-hash>

**Working with Stash**

Lets say in working directory in master we have a calc.py that contains

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

def add(x,y):

pass

def subtract(x,y):

pass

def multiply(x,y):

pass

def divide(x,y):

pass

def square(x,y):

pass

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

*git branch add*

*git checkout add*

*make changes to add function : inplace of* ***pass*** *write* ***return x+y***

*git diff*

*git stash save “worked on add function”*

*git diff ( no changes will show up but for good)*

*git status*

*git stash list ( will show stash@{id}) you can switch branches work and come back*

*git stash apply stash@{id} (take you back to the changes)*

*git stash list ( will show the id again)*

*git checkout -- . ( going back to old change)*

*git stash list*

*git stash pop ( apply and drop the stash)*

*git stash list*

*git diff*

*Lets make change to rest of function*

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

def add(x,y):

return x + y

def subtract(x,y):

return x - y

def multiply(x,y):

return x \* y

def divide(x,y):

return x / y

*git stash save “Calc Functions”*

*git stash list*

*now lets add the square functions that we have not added*

def add(x,y):

pass

def subtract(x,y):

pass

def multiply(x,y):

pass

def divide(x,y):

pass

def square(x,y):

pass

*git diff*

*git stash save “added square function”*

*git stash list (you can see the calc function is pushe down to id 1 from 0 and square function takes id 0)*

*git stash drop stash@{id} (put the id that you want to drop, say we want to drop square fucnction change is that is 0)*

*git stash list*

*git stash clear ( will remove all the stashes. Be careful when running this, ensure if you want to get rid of all the stash changes)*

*git stash list*