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.

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 add -A (all the chnages in file)

++++++++++++++++++++++++++++++++++++++++++++++++++++++

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'.

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

`git clone /srv/repos /local/copy`

Create branches A-D to represent the features you're working on.

`git checkout -b A`

`git checkout -b B`

`git checkout -b C`

`git checkout -b D`

Feature B is very simple and you want to knock it out and get it into production ASAP. `git checkout B`

[work in editor]

`git commit -a`

[peer review]

`git format-patch`

`git send-email` [options will vary]

[peer gives you "thumbs up"]

`git checkout master`

`git merge B`

`git push`

Get rid of unnecessary branch B. `git branch -d B`

Stashing Temporary Work

Switch to branch A, write 100 lines of code `git checkout A`

Switch to branch C while waiting on a co-worker's advise for A

`git stash`

`git checkout C`

Work on C for a while, get advise from co-worker and resume work on A `git stash`

`git checkout A`

`git stash list`

`git stash apply [stash name]`