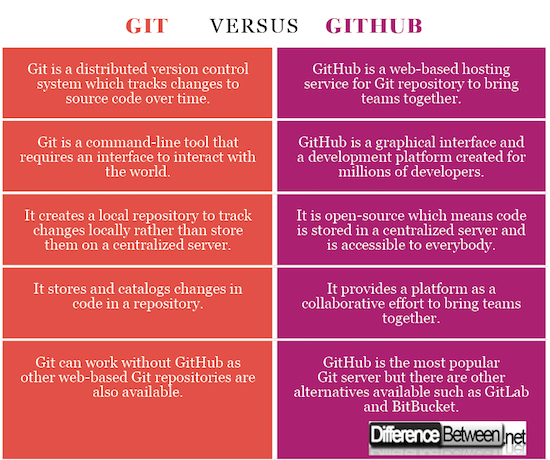
Git and GitHub



Command to make a directory : mkdir directory\_name the if I want to switch into that directory command is :directory\_name

[Using the git pull Command in Git](https://www.delftstack.com/howto/git/difference-of-git-clone-and-git-pull/#using-the-git-pull-command-in-git)

We use the git pull command to get updates from remote to local ones. The command will update the files in your local repository with the files in the remote repository.

Common use options of git pull:

1. git pull <remote>: Get the latest changes from a remote repository and copy them to a local repository.
2. git pull --no-commit: Does not overwrite your commit history when merging.
3. git pull --rebase: Merge a remote branch with a local branch.

Difference between git init and git clone :

There is a difference between git init and git clone. The git init command will initialize and empty the local repository.

 Use git clone --bare to get a copy of the remote repository without a working directory, meaning you cannot change the project.

How can we make changes to our on the git hub :

Step 1 : create a working directory by : open git bash from the file which I want to make it working directory or I can make my own directory by :mkdir directory\_name then change into that directory by :cd new\_directory\_name

Step 2 : git init

Step 3 : git pull https\_link

Step 4 :make the changes which you want . then staged it by :git add --a and the commit that changes

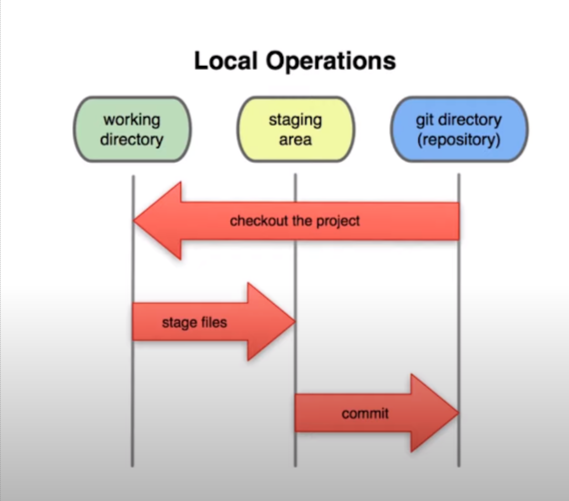
Step 5:git remote add origin https\_link

Step 6 :git push -u origin branch\_name

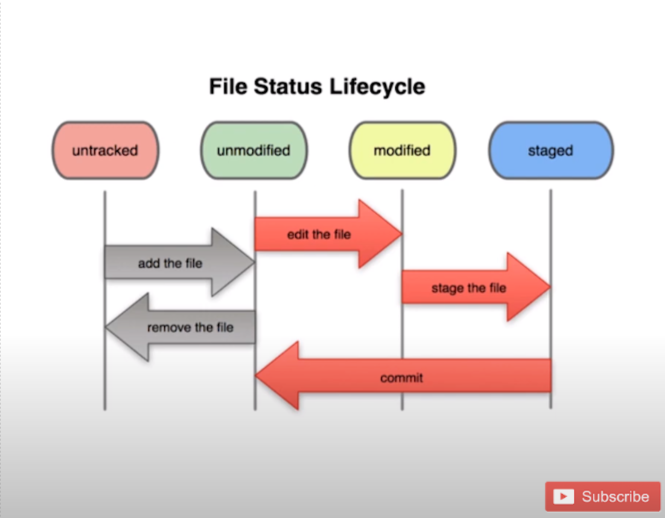
1. $ pwd :(print working directory) :give information about the path of file .
2. $ git config –global user.name “name” : use for setting name .
3. $ git config –global user.email “email” :use to set the email of user .
4. $ git config user.name : going to give the name of the user.
5. $ git config user.email : going to give the email of the user .
6. $ git config --list : going to give the list of all .
7. $ cd /location : use to set the location .

Cd=change directory

8 . $ ls : give all the file present in present working directories .



GIT STAGES :





1. $ git status : it give status whether the we are in git repositories or not .

2. $ git init : it is use to initialize git repositories

3. $ git add --a: adding every file in staging area . or we can use $ git add .

3. $ git comit -m “initial comit” : it is use to initial comit of file we have made .

4. git add filename.extension : it use to add a particular file in staging area .

Ex : $ git add newtxt.txt

5. $ git log : it will going all the information of changes we made . Actually it track the changes we make in our git reposirites.

Note : to exist the git log : we need to press q .

6. rm –rf .git : use to delte the reposerities that we have created with the $git init .

Note : above written command is : 1.rm 2. -rf 3. .git;

7.$ git clone : use to clone the git apis , github reposeritries .

🡺 Git ignores

If we want to ignore some file then we can create a .gitigonre file and put all the file which we want to ignore in it .

Step 1 🡪 We create a ignore file with : $ touch .ignore

Step 2 🡪 put the file we want to ignore in it .(with its name )

Example :

We create a error file with : touch error.log (.log is used to create a file ) and we want to not to include it in the git directories the we can ignore it with by creating a another file : $ touch .gitignore , in that gitignore file I need to write the name of file in it like error.log ;

Note : in order to ignore many file of same extension like(.log file ) we can do this with by writing

\*.log in the .gitignore file .

* Ignoring outer file and inner file :

In .gitignore we write : dir/ will ignore all the dir folder but /dir/ ignore only outer folder

\*\* comparing working directory with stagging area

* $ git diff : compare the working directory(modified docs) with the staging area(docs which is about to commit )

\*\* Comparing staging area with the commit

Use command : $ git diff --staged

🡺 skipping staging area :

(directly commit all the tracked file by git without staging . it will going to commit all the tracked modified or unmodified file but not not to do any thing with the untracked file (new file which is not add to git directory )

Command : git commit -a -m “message” .

12. removing file from the directory .

$ rm file\_name.extension

13. renaming a file :

$ mv oldfile\_name.extension newfile\_name.extension

14. stop tracking a particular file :

$ git rm --cached file\_name\_extension

COMMIT OPERATION :

1. after commit if we want to see which file is removed or which is added :

Command :git log -p

Or to see for n commit command : git log -p -n ;

Where n=integer for past n commit .

2. command : $ git log --stat : it will going to give a small comparssion of what is added or what is removed .

3. $ git log --pretty=oneline

Or $ git log --pretty =short

Or $ git log --pretty=full

4. applying time filter on git log :

Command : $ git log –since=2.days

5. Make a amend in previous commite :

Setp 1 :select a commit you want to make changes by : $ git log -p -n;

Step 2 : make changes you want

Step 3 : then enter command : $ git commit -- amend , it will take us to a new editor ,so I can make change which I want

Step 4 : To exist from that editor : 1 .enter i 2. Enter :wq press enter

🡺 ustaging file :(to modified)

Conversion : staged to modified

Command : git restore --staged file\_name.extension

🡺 moving to previous commit from modified

Conversion : modified to previous commit

Command : git checkout -- file\_name.extension

Note : for group changes command : git checkout -f

🡺 Work with remote repository :

Step 1 : create a reposiritry on github

Step 2 : put all the command one by one in git bash editior :

Commands are :

git remote add origin https://github.com/AjajAlam1399/new\_reposiritries.git

git branch -M main

git push -u origin main

setp 3 : if we make any changes the we want to push changes on the reposirtries the we need to do :

git push origin main

note : in order to view the current git remote :

command : git remote -v

* way to changes the current remote repositories :
* command : $ git remote set-url orgin https\_url\_of\_new\_repositries

🡺 ALIAS :

IT USED TO CONVERT BIG COMMAND INTO SMALL COMMAND :

SYNTAX :

$ git config --global alias.shortcut\_command\_name ‘ orginal command--‘

Example : to create a alias for unstaging file

git config --global alias.unstaged ‘restore --unstaged .’

git ustaged file\_name\_extension

🡺 Branching in git (creating or switching to other branch ) :

Meaning of branching in git :

Branching means creating other branch and we can make changes without disturbing the existing one .

Command :

i) Git checkout -b newbranchname (for creating and switching to new branch )

ii) git checkout switching\_branch\_name(switch to existing other branch ))

iii) git branch (give the name of all the branch )

iv) git branch -v (give the information about last commit of each branch )

v ) git branch --merged (give the list of merge branch with the current branch)

vi) git branch --no-merged (give the list of not merged branch of the current branch)

vii) for deleting a branch

git branch -d branch\_name (it will going to give an error if the branch is not merged at all )

for delting not merged branch the command is :

git branch -D branch\_name

🡺 note : Before switching to other branch always try to staged and commit the changes that you have made .

🡺 MERGING TWO BRANCH:

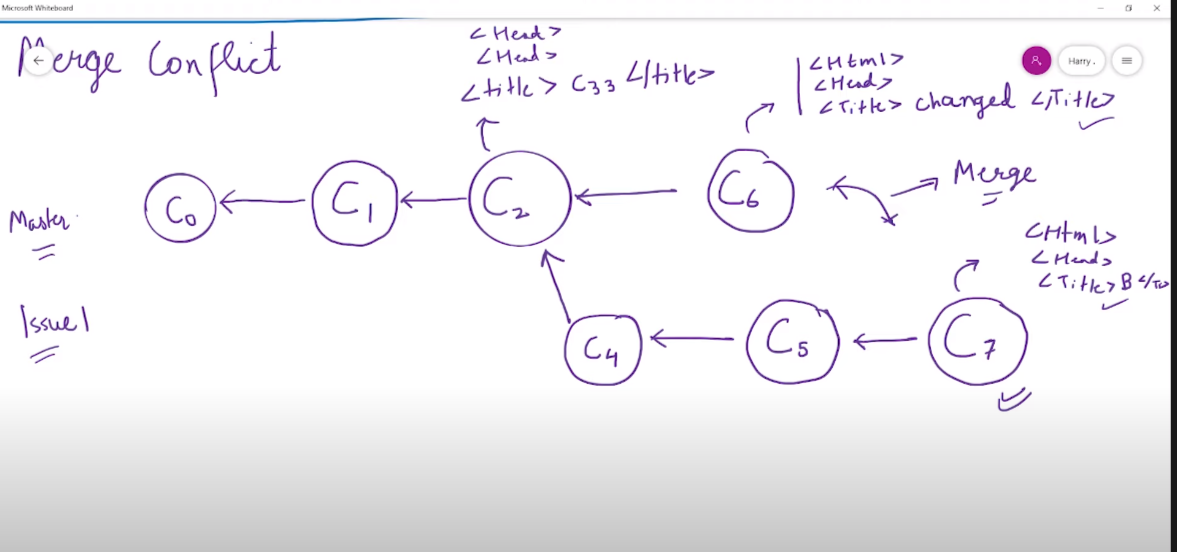
Go to master or other branch you want to merge them .

Commmad :

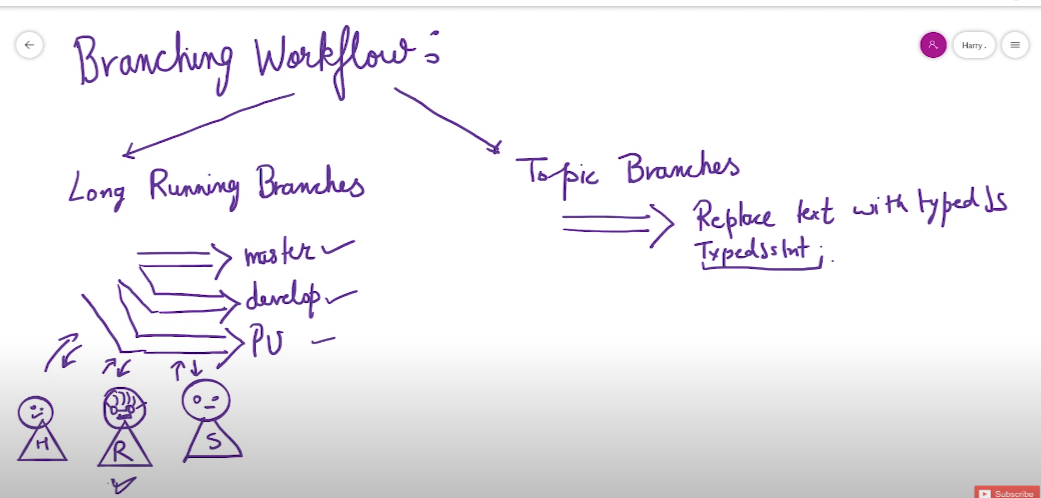
git merge branch\_name

and the commit the changes

🡺 Merge conflict :



Branching workflow :



🡺 pushing git branches on remote repositories

After adding or creating remote repositories

Command : git push origin branch\_name

Ex: git push origin developer

Note : recommended things : I if want to push developer branch then It is prefer to be in the developer branch .

🡺 Deleting a branch on the remote repositories :

Command :

git push -d origin branch\_name

==> we have error like this :

master -> master (fetch first)

then we need to do this :

git fetch origin master:tmp

git rebase tmp

git push origin HEAD:master

git branch -D tmp