# Complete Git and GitHub Tutorial

⇒ GitHub - platform which host repo
- Interface

⇒ Git - verson control system

⇒ Red - deleted - code has been deleted.
⇒ Green - Added - new code has been added.
⇒ git init - initialize the repository
⇒ ls -a - lis/show all the directory, hidden also
⇒ ls .git - - show .git folder
⇒ touch nk.txt - create a file nk.txt
⇒ git add nk.txt → git add . (All files)
⇒ git status - show the status of branch
⇒ git commit -m "nk.txt file added" - commit the file to Repo
⇒ cat nk.txt - show on terminal
⇒ git add .
⇒ Remove from Stage
→ git restore --staged nk.txt | git rm --cached nkI.txt

changes made in stage area will be removed.

⇒ All History
→ git log - show all history

→ rm -rf name.txt

⇒ Unstage comments/changes

→ git reset hasid

⇒ First Add — Add/Store some features for future.
→ git add .
→ git status

→ don't want to commit, also do no want to [illegible] Lose.

Date ___/___/___

→ git stash : move to some other Area from stage

⇒ ~~Come-back to~~ Un-stage Area
→ git stash pop

→ git stash clear → remove unstaged comments/files.

connect local folder to remote repository with URL.

(GPROJECT) git remote add origin https://github.com/nitish-Kumar/Devops

→ so our GPROJECT (local) folder is ~~add~~ connected to remote Devops with name of link 'origin'.

⇒ to see all connections
→ git remote -v // see all linked repo info.

→ to push your changes

→ git push origin master
to which URL to push — origin
to which branch you want to push. — master

→ Branch
- git branch feature_01 // new branch is created and head is pointing there.
- git commit

→ git checkout feature_01 // come out of feature_01 branch to main branch.

→ merge to main branch
→ git merge feature_01.

pageNo-3

Date __/__/__

now push it to remote repository

git ~~remote~~ push origin master.

→ git ~~remote~~ push origin master.

copy project in your Account

→ fork
//make copy of the project in your account at GitHub.

// nitish-Kumar / copied-project

copy to local room ← If we clone, name will be origin.

⇒ git clone URL

Upstream → - From where you have forked the project

⇒ git remote add upstream Forked URL address

⇒ One pull request should be related to, one branch (PR request).

⇒ new pull request, new branch.

⇒ Now remove something from local folder and try to push this changes on repository, you will not be able to, so

in local
→ git reset has_id
→ git add .

Force - push    commits are interlinked

⇒ git push origin nitish -f
↑
branch

⇒ Go back to previous commit or delete the history from log, put previous has_id and do:

git reset has_id ;    it will restore to last point and above it all files will be unstaged, all comments will be unstaged.

⇒ when changs done in staged area need to be moved

First on the stage > git add.
> git Status

> Go to backstage
> git stash    // file will be moved from local directory also.

Result: working directory clean, changes done in staged area moved to some other place. in unstage area.

Bring back those changes from unstaged to [illegible] local directory.
> git stash pop

Completely Remove

1. stage    git add.

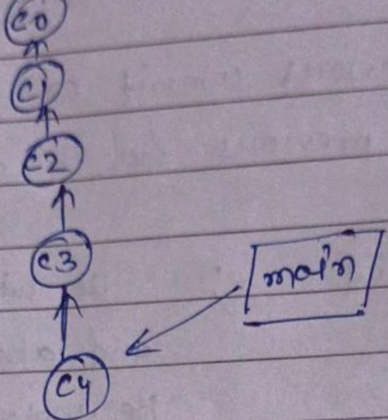
2.    git stash.

3.    git stash clear.    (changes gone permanently)

Git-Hub

→ personal account URL will be by-default referred as origin

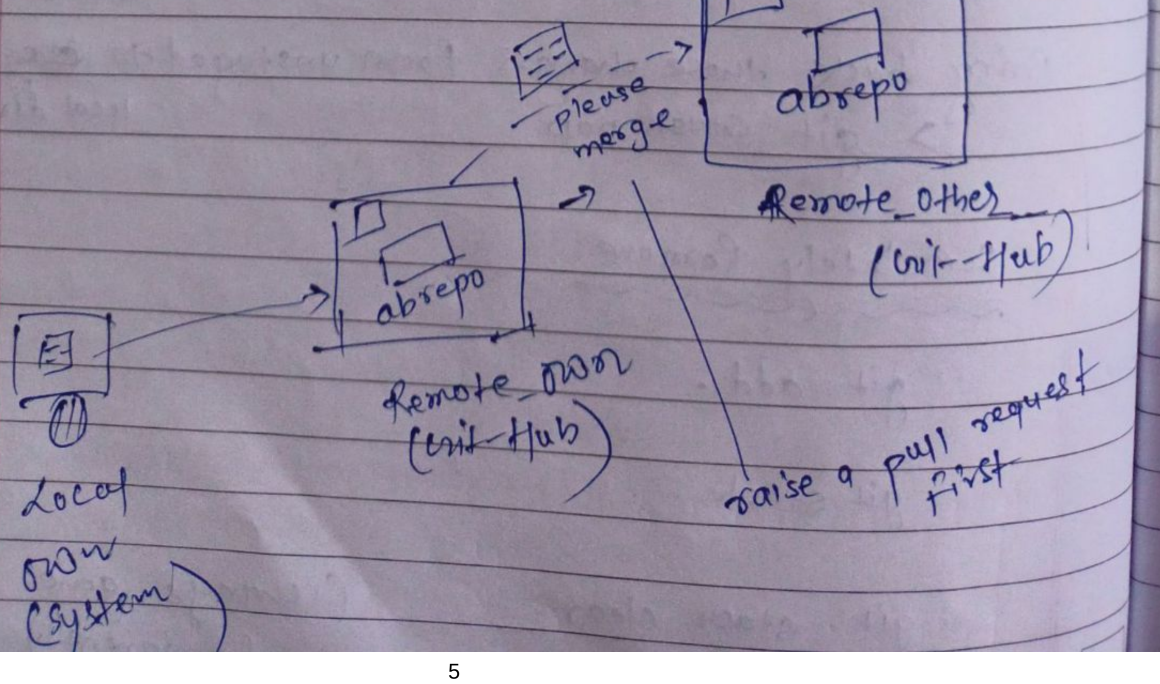
→ Branch : master - previously it's name
main - new name



→ Head

commit → (Head → master , origin/master )

if on top, nothing, there in local git repo, which is not present in remote repo.



Page No - 6

⇒ git push origin nitish

$\underset{\text{which URL}}{\uparrow}$ origin, $\underset{\text{which branch from local system}}{\uparrow}$ nitish

→ base repository: Kunal-Kushwaha   base: main ← head repository : nk10008/comm..   compare : nitish

head repository — copy or Fork of any ~~individual~~ person → has a branch nitish, this branch contains the changes in green below.

- requesting please merge these chanages to main branch of the project.

- when you click on $\boxed{\text{Create pull request}}$

Remote other owner will get a notification to merge.

100 differen things on one pull request
 - difficult

⇒ 1 - Branch can open one pull request

⇒ if you want to remove newest changes, just copy below hash and do ↓

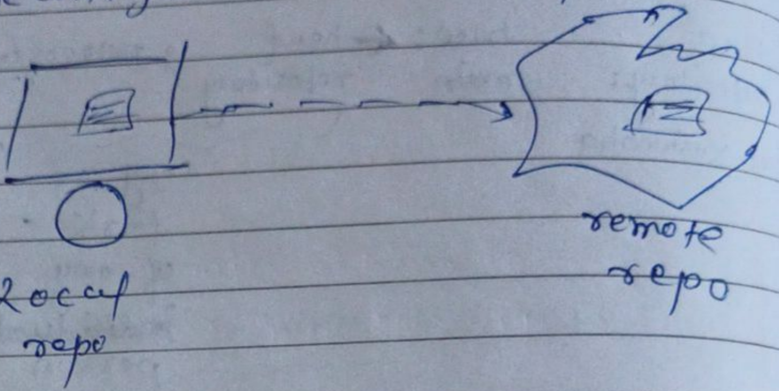
```
-> git reset hash_id
-> git status
-> git add .
-> git stash
```

Page NO - 7

→ If there is difference in your Local and remote (Git-hub) then Force push.

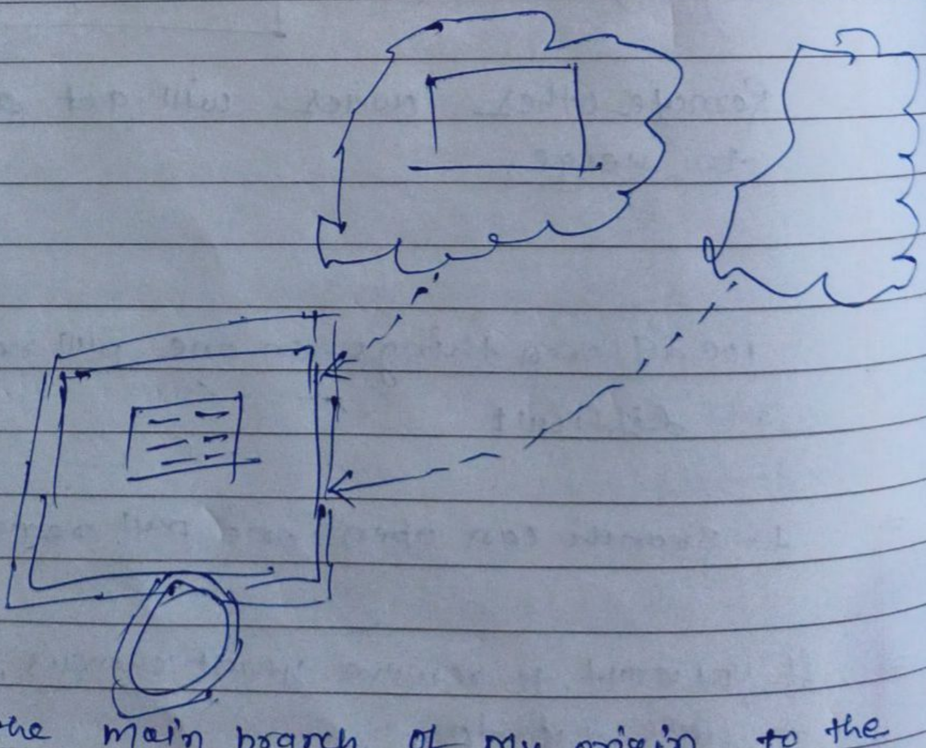
→ git push origin kunal -f
(nitish)

Now same changes will be at both place



→ Sync upstream and your Forked project from Upstream

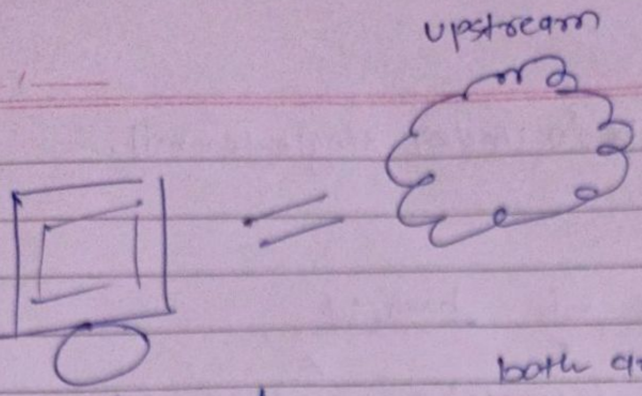
1. > git fetch --all --prune
it will fetch changes from
both places 1. your own remote repo (Git-hub)
2. Upstream Repo
to you local git Repo on System



2. Reset the main branch of My origin, to the main branch of upstream.

git reset --hard upstream/main.

Date __/__/____



both are now same.

Local forked
Upstream project

8. Update you forked branch on Git-Hub

> git push origin main

$$\begin{tikzcd} \text{origin} \arrow[d, leftarrow] & \text{main} \arrow[d, leftarrow] \\ \text{which URL} & \text{which branch} \end{tikzcd}$$

Above steps in One - Command:

> git pull upstream main.
  (it sync with upstream to local)

After that do synch you forked project on Git-Hub.

> git push origin main

Pick & Squash

1/ git branch temp
git checkout temp

make sure main branch is updated.

Squash commits in One single commit

- touch nk1.txt git aad .; git commit -m "first commit"
- touch nk2.txt git add .; git commit -m "Second commit"
- touch nk3.txt git add .; git commit -m "third commit"
- touch nk4.txt git add. ; git commit -m "Fourth commit"

Merge all commit above in single commit

I. trick one

1 reset to last commit, befor above commits done
2. All above commits will be in Unstaged Area.
3. then do git add . , git commit -m " "

page No - 9

Date __/__/__

Now all will be in under single commit.

<u>tricks - 2</u>

```
git rebase  -i      hash_id
             ↑          ↑
      interactive    ~~hash~~ id
      environment    before all above commits.
```

Now all the commits will appear in nano env editor and now you can either pick them up or squash them.

```
pick 37fb85  fcommit     |  →                         ↑   |  →
pick 8f50eff fcommit     |    S ————                      |  S
pick 5225f65 fcommit     |    S ————                      |  S
pick 3de5c65 fcommit     |    S ————                      |  P ---
                              all three will be
                              squashed to above
                              commit
                                                          one these
                                                          two to
                                                          above one
```

~~step-x~~ press ctrl+X, press Y, press $\boxed{\text{enter}}$

now, you will write message about all.

(want to, go to back)

→ git reset --hard hash_id

(Head -> temp, upstream/main, origin/main, main)

all of them at same commit.

Page No.