

GIT-HUB



* Terminal :- It allows us to manipulate the ~~git~~ ^{file.} structure using commands.

* ls : Command that lists all the files of your current folder.

Q. How to maintain the history of the project?
and where is this entire history being stored?
like, create, make changes, or delete any file.

All of these history are stored in store in another folder that git provides us.

This is known as git repository and named as .git.

These files are hidden, so, how do we get this folder, where all the history is saved.

Just Command :-

: git init

and : ls -a

Here a :- means show me all the hidden files.

Any file that starts with '-' is hidden in the repository.

Now, Any change you made in this ~~git~~ repository, git will take the picture of it.

" : touch names.txt. "

To Create a New file.

: "git status."

This is a command, that tells that whether the changes you done here is in the history or not.

So, currently the changes are not saved here in the history, because they are untracked or not on the stage.

Ques How to stage the names.txt file?

Here is another command :-

[git add.] // Here : means everything in the current project dir. that is untracked there history not saved, put all the files in the staging area, so, that photo of them can be taken.

git add names.txt.

either you can use this command to stage the files individually

Now Command (git status) again, and you can see that, file names.txt is on the stage. and ready to take the picture.

So, We Command photographer, Hey! photographer please take their picture, so, that they are permanent-ly saved in the photo album or git history

So, Command for that :-

`git commit -m "names.txt file added"`

Here -m : (message) provide a message.

Now, there is no other person whose photos needs to be added

* Now, let's add something in names.txt file :-

So, Command for that

`vi names.txt`

Now, Typed by using "I" and make changes in text file and save and exit,

by Ctrl + C and (:q), Enter,

Now to open the edited file - ?

Command for that :-

`Cat names.txt.`

Ques How do we know that I modified names.txt.
= `git status.`

Here you will see :-

`modified: names.txt`

Now git add: - again
and again

git status

and you will see that names.txt is again staged.

If you want to unstage any files bcoz, their picture is already taken or save in the history then, Command for that -

git restore --staged names.txt.

Now you can click the photo of the modified file. names.txt or save into the history

So, Command -

git commit -m "names.txt file modified"

Ques What if you want to delete the file names.txt.
Command :

rm -rf names.txt.

Now → add the file again by ~~Commit~~ Command

git add names.txt

and make commit -

git commit -m "names.txt file deleted"

you can see the History of all the activities by ⁽⁵⁾
Command

git log

Ques What if you commit something by mistake, how to delete that commit?

Ans first of all, we cannot remove ~~just~~ that commit which is in the middle, because as you can see that each commit has a hash ID, so, each commit is built on top of each other, so, if you want to delete multiple commit or the top one commit, just copy the Hash ID of just next commit and
Command :-

git reset <Hash ID>

All the above commits of that Hash ID is removed

Now, you might be like, what happens to all the files that were modified or changed in previous commits?

They are in unstaged area

This is like people whose photograph was taken was be deleted, and they are again in the section of people whose photos has not been taken

(git status)

Here, we see that name.txt deleted,

Now, we want to store this file into stash area,

(stash area means)

I don't want to delete ^{or make} the changes in name.txt,
but also I want them to go back stage,
I will call these files whenever I need them
back.

Eg:- you want like, I wish there was a way where
I could just put all my work somewhere else
without making any commit or history for
the project, and
whenever I want that thing back I can get
that whenever I want.

Command for that is called git stash.

First:- Get the people you want and stage them first

So, (git add . or git add names.txt)

Second:- send them to Back stage.

3- Make another change -

(touch surnames.txt)

(git add surnames.txt)

(git status)

(7)

Here, the file names.txt has been renamed.

renamed: names.txt → surnames.txt → See like that

make changes —

like —

vi surnames.txt

edit →
Kushwaha

~
~
exit

touch houses.txt.
git add houses.txt

Now

git status →

renamed: names.txt → houses.txt
new file: surname.txt.

→ See like that

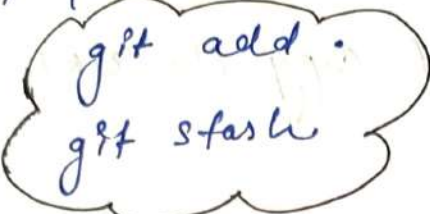
Now, you want them ~~add~~ to the backstage

← git stash →

Now, you want to call all the people or files who
were backstage on the stage.

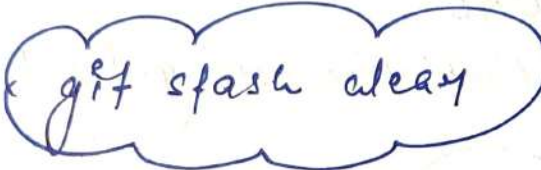
Command :- git stash pop.

Now, you want to send them back again — ②

So,  git add.
git stash

Now, those people who are Back staged, you want them to go away, you won't take their picture

Command:-

 git stash clear

Now, Go to Github ;

(9)

* and make a new repository and copy the URL of that repository :-

Command →

`git remote add origin <URL>`

remote : We are working with URL.

add : Adding new URL.

Origin : Name of URL that you are going to add.

⇒ `git remote -v` ⇐

This will show all the URLs that are attached with the present folder.

Now refresh the page of URL that you have copied there is no different, no any file named names.txt added into Community classroom repository.

So, if we want to add this names.txt file into your repository -

Command →

`git push origin master`

push: means push the changes,

Origin: To which URL you want to push

Branch: & which Branch you want to push

#. Now make random files and random commits ÷

10

like—

\$ touch hotel.txt.

\$ git add.

\$ git commit -m "hotel.txt added".

\$ touch rollno.txt.

\$ git add.

\$ git commit -m "rollno.txt added".

\$ vi rollno.txt. // add some roll no.s. & changes made.

\$ git status

* Here you will see.

Modified: rollno.txt.

\$ git add.

\$ git commit -m "rollno.modified".

\$ git log

→ Here you will ^{see} the 4-Commits.

* In which you can see that there is only one commit in the master branch, because that is the only thing I pushed.

Now, we want to push these new commits.

git push origin master.

Now all the files are added to your repository.

11 # How to Contribute to Open Source # On Git-Hub. ??

What we need to do, to make some changes to any
repo or contribute to any project?

Ans- So, if we are able to contribute or make changes
to any project directly, so how dangerous it
would be?

So, we don't have access to change it directly

How to resolve? →

You create the copy of that project in your own
account so, fork that account, and the
project will be copied to your own account.

So, now you can do whatever you want,

Now, that project which is in your account (added)
Copy its clone and remember any folder that starts
with your own account the name of that is
origin

So, Command:

`git clone <clone URL>`

Note:- From where you have forked the project is
known as Upstream URL by convention

So, Command:

`git remote add upstream <URL of main project>`

(12)

git remote -v

Here, now you can see, upstream and origin URLs, where, origin is your personal and upstream is from where you've forked it.

Now make some changes in the project and modify.

* And before committing?

make sure that you're not committing to the main branch, if yes, then, make another branch.

Command -

git branch <name>

Now switch to that branch -

git checkout <name>

Output :-

switched to branch <name> (seems like subs)

* Now add the modified files \Rightarrow and commit here \rightarrow

git add :

git commit -m "Shahma added a manage".

Now, if you want to make ~~change~~ these changes what you have done here into main project or into upstream branch, you have to first send "pull request"

So, first push these changes into you account first
which you have forked.

(13)

So,

Git push origin < Branch name >

Here \rightarrow Branch name :- Branch which you have
Created and switched to —

Note :- for every new feature, bug or anything you're
working on create a new pull request, and
from examples as you can see that if a branch
has already a pull request associated with it,
it will not allow you to add another pull request.
all the commits will be added to that pull request
only.

In simple language —

[1 pull request means 1 branch]

1 branch can open only one pull request,
if you want to open different pull request
for different features that you're working on for
the project which is also recommended for that
you want to open diff. pull request and since one
branch can associated to one pull request,
So, Hence, you'll be able to create diff. branches.

So, Any particular new thing you're working on create a new branch in your local folder make a pull request from that. 14
You can, have 10, 20, or 100 branches whatever you like, make sure your work is distributed, so, that it does not get mixed up.

How to delete Commits :-

1st - first add some file -
like -

touch names.txt

git add .

git commit -m "names.txt added"

git push origin shadun

Same branch

Now if you want to delete this commit -

So, Copy the Hash ID of Commit just below of it

So, git reset < Hash ID >

Now, Names.txt is unstaged,

git add .

git stash

Now, file names.txt is gone.

Afterwards, push this +

(15)

git push origin shahna -f

Query - f ?

Because the online repository contains a commit that my repository does not and commits are interlinked so, we'll have to push this using (-f)

How to fetch all the files to the main branch.
from shahna or whatever branch you've created?

* First checkout to main branch

So,

git checkout main

* Then fetch all +

git fetch --all --prune.

prune :- Deleted file.

Now, we've fetched all the changes.

Now, Reset the main branch of by origin to the main branch of upstream

(16)

`git reset --hard upstream/main`

Now my folder/fork that I have on my local system the main branch of that is exactly same as the one that I have in upstream,

Now to make online changes all of these -

`git push origin main`

PICK & SQUASH

What if you want to merge multiple commits into one commit:-

There is a command:-

`git rebase -i <Hash ID>`

-i - means interactive environment.

Now, all the commits above that Hash ID, we can pick and squash