

Notes Made By

- RITI KUMARI

~~Bit~~ → free ops distributed version control system - is handle everything from small to very large project with speed & efficiency

## Git / Github

Git → at what particular time which person made what particular changes (version control system)

Github → Online platform which allows us to host our git repository.

- \* Maintaining the history of project (commits basically)
- \* Sharing code with people around the world.

Red → line deleted

green → line added

Download Git → commandline

Terminal → Allows us to manipulate the file structure using commands.

~~ls~~ ls → list of projects (list)

mkdir → makes a folder (make directory)

hp> mkdir project



cd → change directory

hp> cd project

All of these histories are stored in a file called  
git folder.  
↳ (hidden files)

git init (initialise empty git repository)

git : (master) ls -a → show me all the hidden file

git : (master) ls .git → files inside git

To create a new file we can also use touch

project git : (master) touch names.txt

Q How do we know about the changes being made?

git : (master) \* git status

↳ tell us the changes

Untracked files → names.txt which are not in (whose history has not been saved yet)  
If we share my project with someone no one tells us about it.

To have these changes in our history

project git : (master) \* git add names.txt

git add .

↳ put all the files which are unsaved

project git:(master) \* git commit -m "name.txt  
file added"

commit  
the changes  
message

lets say we are wedding. People/guest take photo with couple.

Guest → Took photo → come out of stage → photo same in album.  
goes to stage whose photo is not taken.

(git history)

(untracked file)

project git:(master) vi names.txt

write it

it basically displays whatever is there in names.txt : x

project git:(master) cat names.txt

file

insert  
i  
esc  
esc in mode

exit

Now our file is modified

It shows the names

git init

git status

git add

git commit

To restore the modified names.txt to its original one we use

git restore --staged names.txt

git status

where can we see the entire history of the project

git log

Now delete the file

~~rm -rf name.txt~~

Now what if my file got deleted by mistake?

Each commit has a hash id they are hashed one above each other.

If you want to remove commit 3 copy commit 1

commit 3

commit 2

commit 1

defgh00c

git reset (hash id of commit 1)

Now the previous commits are in the untracked zone.

Put your project somewhere else without making any commits or history

git add .

git status

git touch surnames.txt

git add .

git status

If you don't want to commit the changes but want to save it for future use.

git stash

To send them to backstage

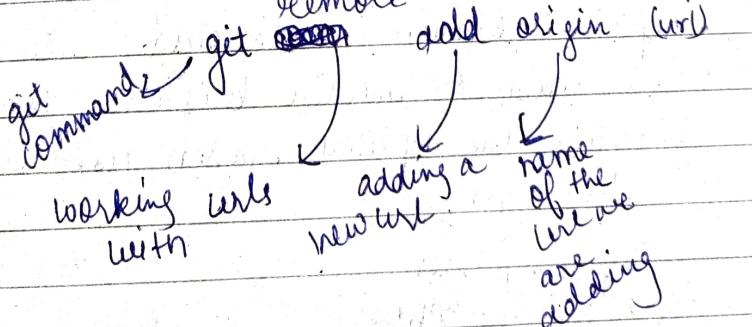
Now all the people from backstage come to the staging area

git stash pop

git stash clear → To permanently loose or let the commits gone.

Github

New repository → After creating got the url now



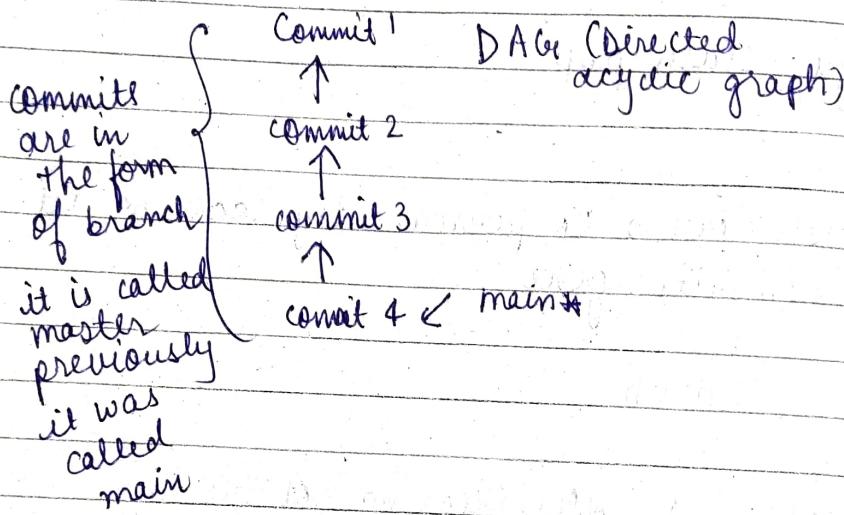
git remote -v → show all the url attached to folder

New url is connected to the folder

but changes is not shared so after refreshing  
we can't see the changes so we use  
git push origin master

↓  
push  
the  
changes

## branches in GitHub



Q. What is the use of a branch?

Ans: Whenever we are resolving a bug we create a separate branch - we should never commit in the main or master branch. Main branch is the default branch used by people.

ls: git

head → just a pointer  
which says hooks

Original → all the  
new index

config → commits info

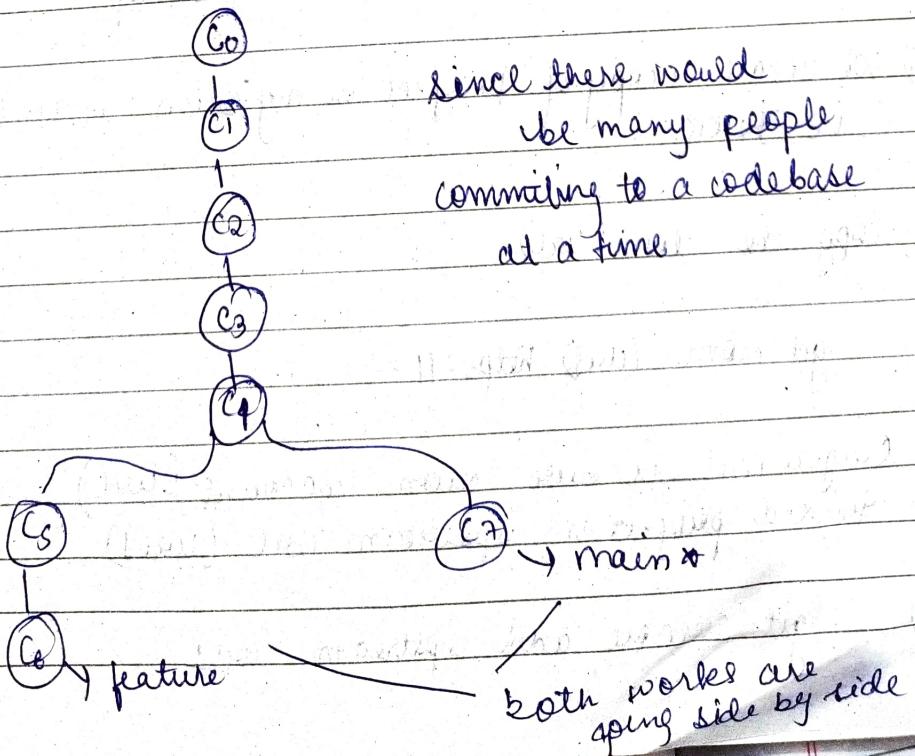
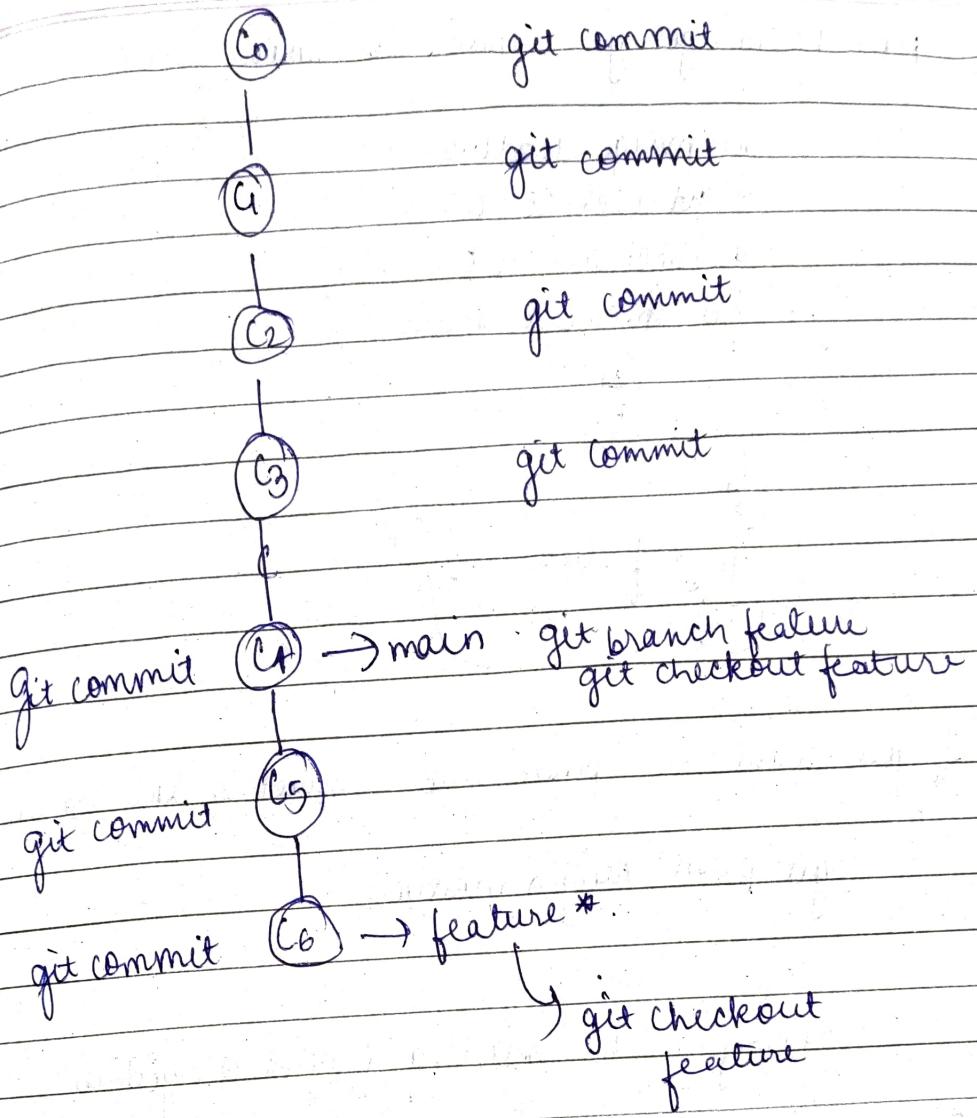
being made info

added info

to head info

logs  
objects  
refs

git branch feature



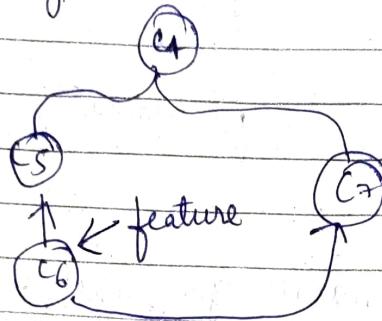
How to merge your code to main code

git merge feature

git checkout CS

git checkout CB.

git checkout main



If you want to push new commit to the branch

git push origin master

Working with existing project on github

To make some contributions to other repos

1) we create a copy of project in my own account  
i.e fork.

2) Copy the clone (url)

git clone (url) https://...

origin url is our own account (url)  
forked project is upstream url (url1)

3) git remote add upstream url1

Checkout → head will come to the new branch

4) Create Pull requests → when you create a copy & want it to be reflected in main project.

\* Lets create a new branch

git branch Riti

\* Move the head to the new branch

git checkout Riti

git add .

git commit -m "head changes"

git log

whatever code is have in Riti branch push it to the main branch

git push origin ~~Riti~~ Riti

Create a pull request

If a branch has a pr all other pr would go into commits i.e. why we don't commit in master branch

- Git - distributed version control system (software)
- Installation - Git windows install (git scm download)
- Git simply save the changes (Revert the change, know the date as well as version)
- Git Bash (terminal) - It works as same as Unix or Linux
- Whenever we use git for the first time we need to provide our email, password etc.
- So we need to write the command

\$ git config --global user.name Riti

\$ git config --global user.email ritikumarupadhyay24@gmail.com

→ To open visual studio run

\$ code (in the git-bash)

To get repo

init

clone

(from  
starting)

(get from server)

→ To initialise a repo : \$ git init

→ To see hidden files : \$ ls -lart

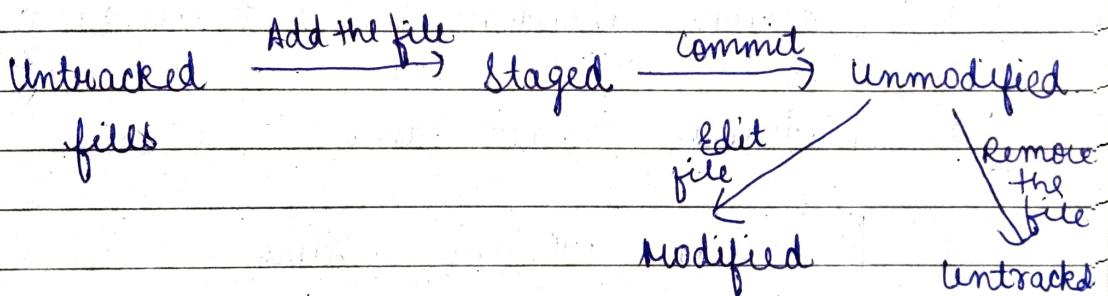
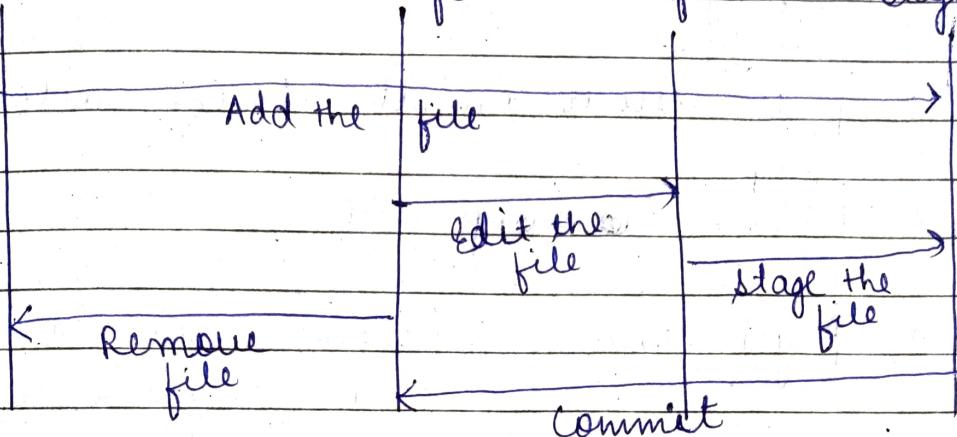
→ To see the status of files : \$ git -status

(git is not used  
↑ to track files)

(the area  
where  
commits  
reside)

Untracked      Unmodified      Modified

Staged

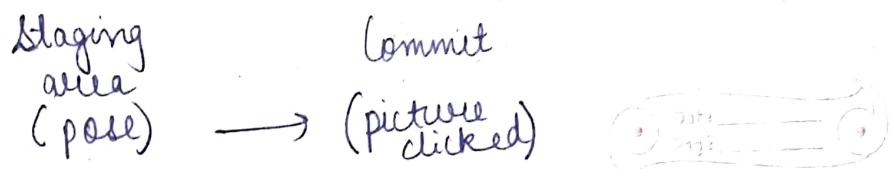


- **Untracked:** The files are not tracked by git

- **Staged:** We directly commit the file.

- **Modified :** To apply changes to file .

- **Unmodified :** We are ready to commit



- 1 → \$ git init
  - 2 → \$ git status
  - 3 → \$ git add index.html (file gone to staging area)
  - initial commit
  - 4 → \$ git commit
  - 5 → \$ i<sup>click</sup> initial commit (vim editor)
  - 6 → \$ esc + :wb . exit
  - 7 → \$ git status (working tree clean  
all changes are committed)
  - 8 → \$ touch about.html (touch → to make blank file)
  - 9 → \$ touch contact.html
  - 10 → \$ touch monuments.html
- Now 3 files are untracked, so to track all these files we would use
- 11 → \$ git add -A (to push all files to staging area)

When we modify contact.html.

↓

Staging area modified.

12) \$ git add -A

13) \$ git commit -m "Added more htmls" (If we don't want to work with vim editor)

14) \$ git status (working tree clean)

15) \$ git clear (to clear the terminal)

How to recover lost file on the last commit  
file

16) \$ git checkout contact.html (checkout → match with my last commit).

To restore all the files from last commit

17) \$ git checkout -f

18) \$ git log (what commits are done)

We need to filter out the log like last 5 commits

19) \$ git log -p -5

You can click q for quit

git diff - let us know the difference (working tree is compared with staging area)

20) \$ git -diff

21 → \$ git diff --staged (compares the last commit with staging area)

22 → \$ git checkout -f (now all the differences are restored)

How to skip staging area and fix it  
(If we want to work with particular file only & don't want to send them to staging area)

23 → \$ git commit -a -m "Skipped staging area"

24 → \$ ls (to list down all the files)

25 → \$ touch waste.html

26 → \$ git add -A

27 → \$ git commit -a -m "Added waste"

28 → \$ git rm waste.html (remove from working directory + staging area (hard disk))

29 → \$ git rm --cached waste.html (to only remove from staging area)

30 → \$ git status

31 → \$ git ls

32 → \$ git commit -a -m "Removing waste".



33 → \$ git log -p -2

34 → \$ git status -s (to get summarised status)

35 → \$ git touch .gitignore

gitignore - A file which we want to be untracked  
like pycache which are a waste. Push &  
pull process slows down

36 → \$ touch mylogs.log

• gitignore

37 → \$ git add -A

mylogs.log  
↳ ignoring  
the files  
which are  
present  
here)

/mylogs.log

only ignore the  
file where gitignore is  
present don't  
get into folder

\*.log → any file having extension log  
\*.cpp

38 → \$ git rm --cached logs/mylogs.log

39 → \$ git status

40 → \$ git add -A

Branches in Git

Branch → We make a new copy and work on it.  
(so that any changes won't affect or slow down the site)

Main branch → main branch by default  
(by default main  
branch is  
called main  
branch)

\$ git branch feature1

\$ git branch

% → feature1  
master

\$ git checkout feature1 (by default it is on  
master branch)  
So to move to feature1

How to merge 2 branches

\$ git status

\$ git add -A

\$ git commit -m "fixed extra bracket"

\$ git checkout master (to get back to master  
branch)

→ \$ git merge feature1

we can make as many branch as possible.

Make a branch & skip to that branch only

```
$ git checkout -b branch2
```

GitHub.

Hosting site for git repositories

Remote repository → we push our local repo in remote repos (It is a url where we would host our project).

```
$ git init
```

```
$ git add README.md
```

```
$ git commit -m "first commit"
```

```
$ git remote add origin https://github.com/udit2409/one.git
```

```
$ git remote
```

o/p → origin

```
$ git remote -v
```

o/p → https://github.com/udit2409/one.git  
https://github.com/udit2409/one.git

```
$ git push origin master
```

(push master branch at origin)

generate SSH & GPG keys (ie giving github acc. access to my PC)

To take read write access on our repository  
We deploy SSH keys in our github account

generate → Add a key   
a key

```
$ ssh-keygen -t rsa -b 4096 -c "ritikumaripadhyay  
@gmail.com"
```

Let the passphrase be blank.

```
$ eval $(ssh-agent -s)
```

```
$ ssh-add ~/.ssh/id_rsa
```

Adding a new SSH key to github account

```
$ cat ~/.ssh/id_rsa.pub
```

copy the content

Title : Riti personal content

Key : paste

to change  
url

```
$ git remote set-url origin git@github.com:  
riti2091/me
```

```
$ git remote -v
```

```
$ git push -u origin master
```

To push different branches.

\$ git checkout branch2

\$ git push -u origin branch2

pull request → A request to improve or make changes

How to clone a repository

- 1) Make a new folder.
- 2) open the new terminal.
- 3) Copy the url
- 4) git clone url foldername