Video - 1

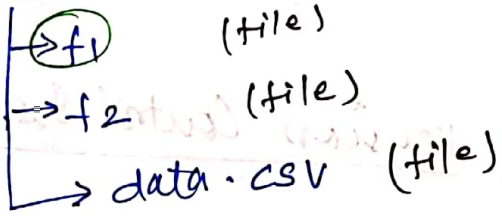
What is git-hub / git & why do we need it ?

* why is Git ?

⇒ version control system

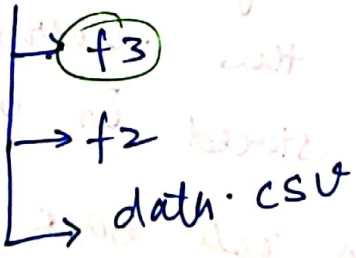
Eg:-

AI desktop Assistant (folder) (version 1)



⇓ after evolution

AI desktop Assistant (version 2)



→ Git is a version control system.

→ Helps in keeping track record of our file.

uses

→ easy file recovery

→ who introduced an issues and when

→ Roll back to previously working state

History of version Control System (VCS)

→ (1) Local VCS → Data Base (DB) to keep track of files

↓↓

all files in your computer

→ not in the server, internet, cloud

→ all file stored in your hard-disk

→ we can roll back here

→ if we lose our PC, we'll lost all our programs.

Track files
↓
Kaun si file kab kab change hui
↓
simple software

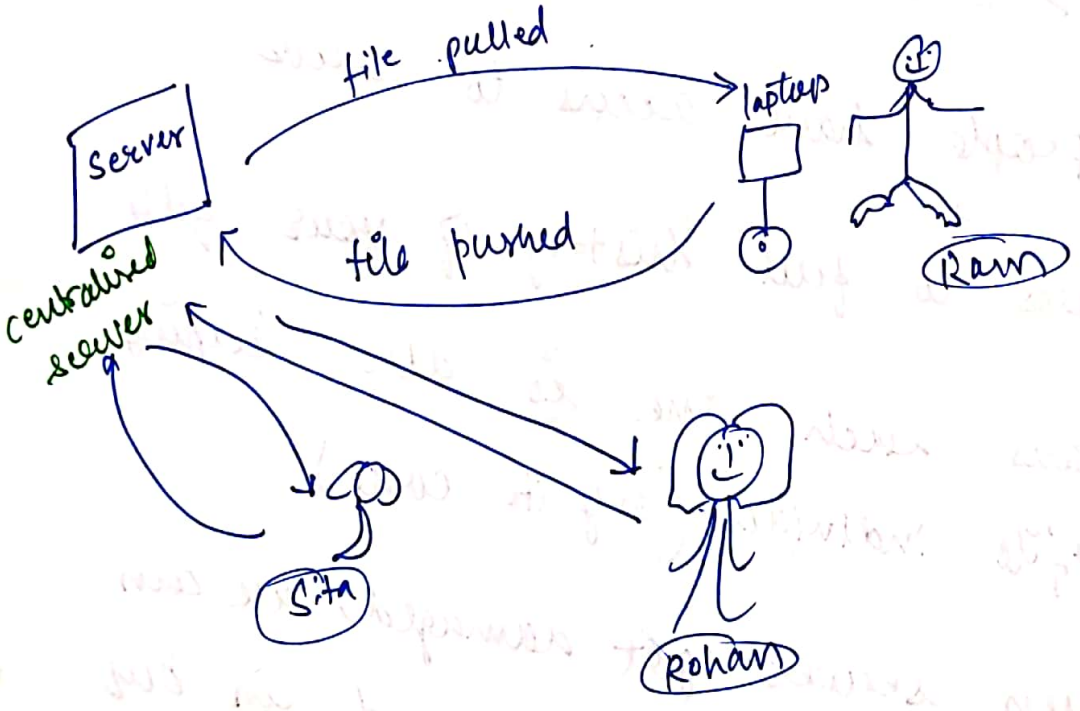
Local VCS

pros → can track files & roll back

cons → if we lose our harddisk, everything is lost

---

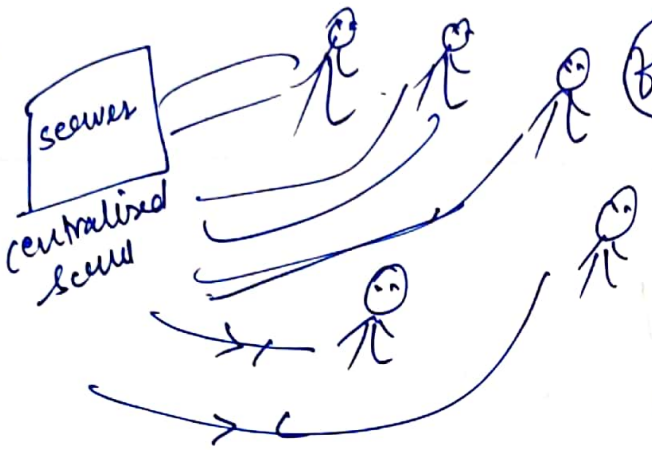
② centralised VCS



⇒ if anyone's Pc is damaged, then they can get the files from centralised server.

⇒ But if server is damaged, we can't roll back our file.

③ Distributed VCS

→ 

→ all people have access to server
+
access to full history of your file

→ it means each one is also keeping the same file individual (only its code).

→ if even server got damaged; we can rebuild it using code saved in our individual PCs.

→ Eg: (gpt)

→ if one is pulling file from server; he/she getting full back of the file.

Qus: How was Git Created ?

Linus Torvalds

2002 → Bitkeeper VCS
↓
2002–2005

2005 → free of charge status was revoked

then he develop Git VCS :

GIT → software (free)

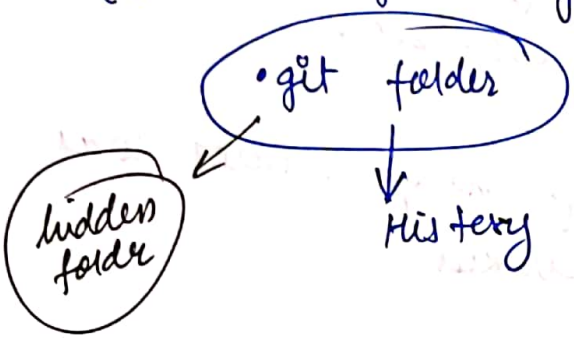
GITHUB → it is a website jo git ki repository ko host करता है (tracking ...)

Alternate :- GITHLAB , BITBUCKET

Git : Features

⇒ (1) Captures snapshot not differences

(if we changed any file, it capture snapshot



⇒ (2) Almost every operation is local.

(no need to push in the server,

just work in your PC

&

then in remote git depority me (CS) me

push kar sakte hai apne

depository

repository

Centralised server

Local

↓

⇒ no internet

no outside network

working in your PC

Remote

↓

⇒ accessing from

internet (any network/

server.

⇒ (3) Git has Integrity.

checksum

↓

value that represents the no. of bits in a transmission message and used by IT professionals to detect high-level errors within data transmission.

Git Repository

↓

tracks & saves the history of all changes made to the files in a Git Project.

It saves this data in a directory called .git, also known as repository folder.

⇒ if checksum of any file changes then it means file has been changed (updated)

⇒ Git → calculate checksum Internally

→ And check that koi .git folder me file ko change na kare.

⇒ (9) git generally only adds data

Video-2

Installing Git + Initial Setup

⇒ Open Google

↓

Type : git install

↓

click :⇒ git-scm.com (official)

↓

windows

↓

64-bit (standalone-installer)

↓

open softwar (to install

↓

Nesxt next

(no need to change defaut settings)

Git Bash → terminal

type: git + enter ↵

⇒ Git Bash is just a terminal just like windows powershell.

Git Installation

- Git command line tool
- Git Bash (terminal program)

⇒ command: cd (change directory)

⇒ terminal works in any directory

eg: local disk C
local disk D

⇒ command: pwd (present working directory)

⇒ It takes path as input of Linux type.

① pwd

→ /c/Users/rahul

② cd Desktop

↓

if any desktop folder is present in

/c/Users/rahul

↓

it will take us to that folder

③ cd "/c"

↓ we'll come in c folder

Now,

create any folder

↓

Right click

↓

open git bash here

↓

type :⇒ git config --global user.name "Rahul" + ↵

(name configured)

⇒ Press ↑ arrow key → phele wala command aa jayega.

type: git config --global user.email "rahul.com" (email configure)

type: git config --list

(configure editer)

type: git config --global core.editor emacs/vim

↓

type any 1

type: git config user.name

(to see your name)

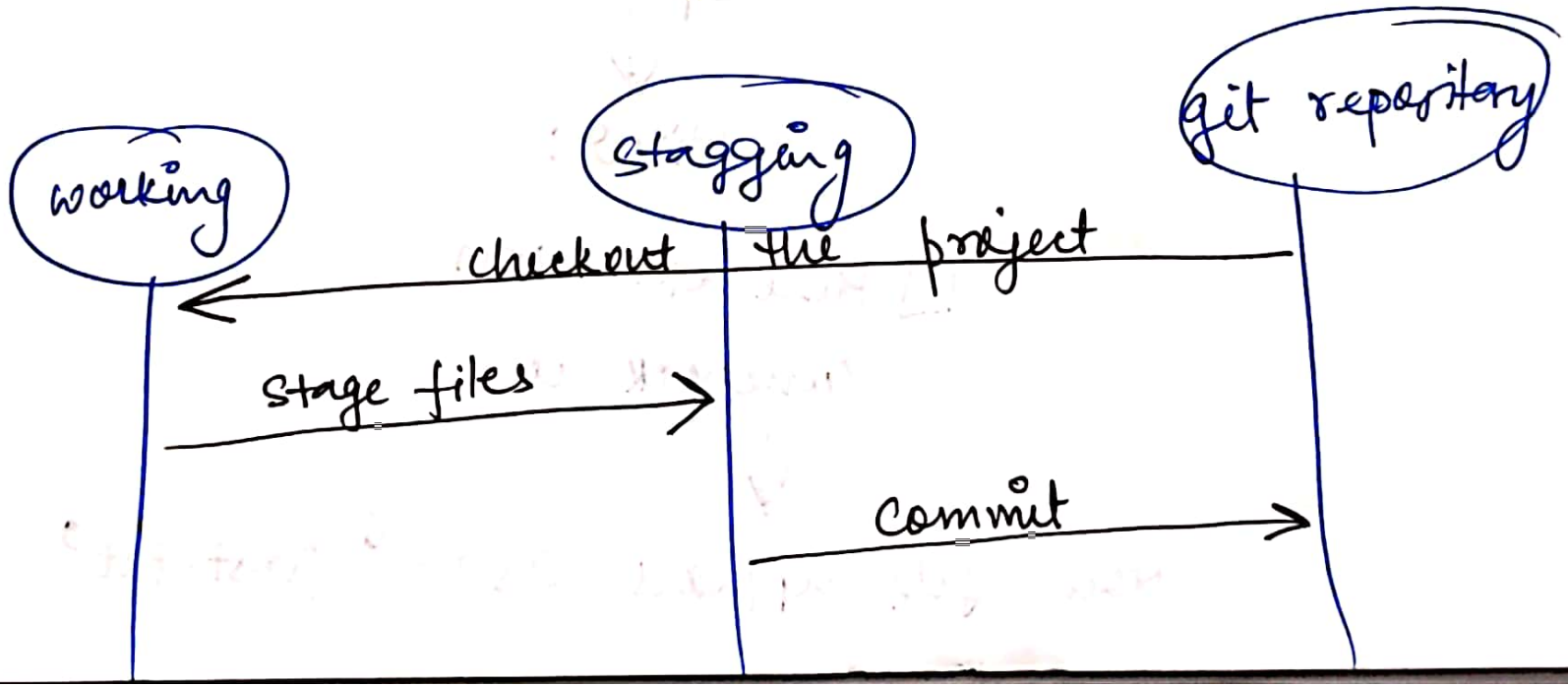
git config user.email

(to see your email)

Video - 3

3 Stages Architecture

1. Working directory
2. Staging Area
3. git directory (repository)



* git directory ⇒ (.git) → folder

↓

it stores all the hidden file

Video - 4

Create any folder

↓

open : that folder

↓

Create text file

new → text

↓

name : first ~~.txt~~

↓

select that text file :

↓

view : Ribbon

↓

options

↓

View :

☐ Hide extension

(uncheck it)

↓

Now file appears as :⇒ 'first.txt'

↓

Open : first.txt

↓

write :→ any Text
(version 1 of Project)

↓

Now, Right click : New → microsoft Access

↓

db (name)

↓

Right click : New → excel document
(my excel) name

Shift + right click in this folder space

↓

menu :→ ~~Open powershell window here~~

click : Git Bash Here

1st command in Git Bash :

$ git status ↵

fatal: not a git repository

↓

this shows it is not a git repository

(we are not inside any git repository
nor it is a git repository)

if not a git respositery.

#Creating a git repositery

① $ git init ↵

⇓

output : Initialised empty Git repository

② $ git status ↵

⇓ on branch master

output : No commits yet

⇒ (nothing added to commit but untracked files present)

branch → by default : master

⇓ main branch

⇒ Untracked files % - - - -

To add all the files in that folder

③ $ git add --a ↵

⇓ 2 times "-"

--a ⇒ all ⇒ all files in Stagging Area

④ $ git status ↵

⇒ u'll see all the files

(5) $ git commit -m "Initial Commit" ↵

⇒ we want to commit & add mess & finish it off.

⇒ $ git commit -m :⇒ opens a editor

⇒ 1st: sari files ko stage kia
2nd: commit those files

(6) $ git status ↵

⇒ on branch master
nothing to commit, working tree clean

(7) $ git log ↵

⇒ To know what we have commited
↓
"Initial Commit"

Now,

open your text file from the folder and add more contain to it by typing, then:

(8) $ git status ↵

⇒ modified : first.txt

Now,
Add few columns in your excel file
then save that excel file

(9) $ git status ↵

⇓

modified : first.txt
" : myexcel.xlsx

⇒ no changes added to commit

⇒ Now, we want to put : first.txt in stagging area & not to myexcel.xlsx

Since we want to work more with excel.

~~(10) $ git status ↵~~

(10) $ git add first.txt ↵

(11) $ git status ↵

⇓

Changes to be committed ⇒ first.txt

Changes not staged for commit

⇓

myexcel.xlsx

(12) $ git commit -m "changed first.txt and added better design" ↵

⇒ Changes it will showed.

(13) $ git status ↵

⇒ Changes not staged for commit:

⇒ myexcel.xlsx

Creating Our Project in Git

$ git init : folder ki git banaya (reinitialise git repository)

$ git status

$ git add -a : sare change stage ho jayega

only for our viewing purpose ← $ git status

$ git commit -m "message" : → to commit your file

$ git status

$ git log : ⇒ To see sare commit

↓

hamne kya kya change kia wo sara details aa gaya (in the form of log file)

Video-5

Cloning a Remote Git Repository from Github

Open Google :

type : github website tensorflow

↓

Open tensorflow site

https://github.com/tensorflow

↓

Click : "clone or download"

↓

Copy the Url

↓

Open prev. wala git folder

↓

folder name

delete the content of any folder

Note :

* $ git rm -rf .git ↵

↓

it will delete .git repository folder

$ git status ↵

↓

not a git repository

Open Git Bush

↓

$ git clone (paste that Url) ↵

→ paste the Url : (Shift + Insert) : shortcut
or directly (Ctrl+V)

⇓

after enter ↓ tensor flow floder create

it will create a directory : tensor flow
and all the directories of tensor
flow will be pulled here

↓

it will take few minutes

↓

after completion :
you'll see a folder : tensor flow

↓ open

all its version
history are there
inside it.

⇓

Open Readme in notepad

↓

↓

remove: 'l' from tensor flow

↓

Close this notepad

come back to git bash

* $ git status ↵

Output ⇒ not a git repository

* $ pwd (present working directory)

note

* pwd → present working directory
  ls → list all the content
  cd → change directory

* $ cd tensorflow/ ↵

* $ pwd ↵

* $ ls ↵

↳ all content inside tensorflow

* $ git status ↵

↳ modified : README.md

* $ git add --a {adding it to staging area

* $ git status

↳ changes committed

* $ git commit -m "Charged readme" ↵

* $ git status ↵

⇓

working tree clear

↳ Your branch is ahead of 'origin/master' by 1 commit

* $ git log ↵ ⇒ to see kaun sa changes किये - किये {31}

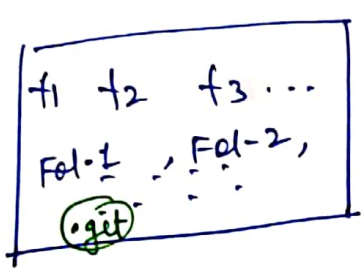
⇒ to quit it

q + ↵

Video - 6

File Status Life-cycle

Directory



} repository

f = file

⇓

is pure folder ko jise hum track karte hai, use hi repository kehte hai.

where
f : file
fol : folder

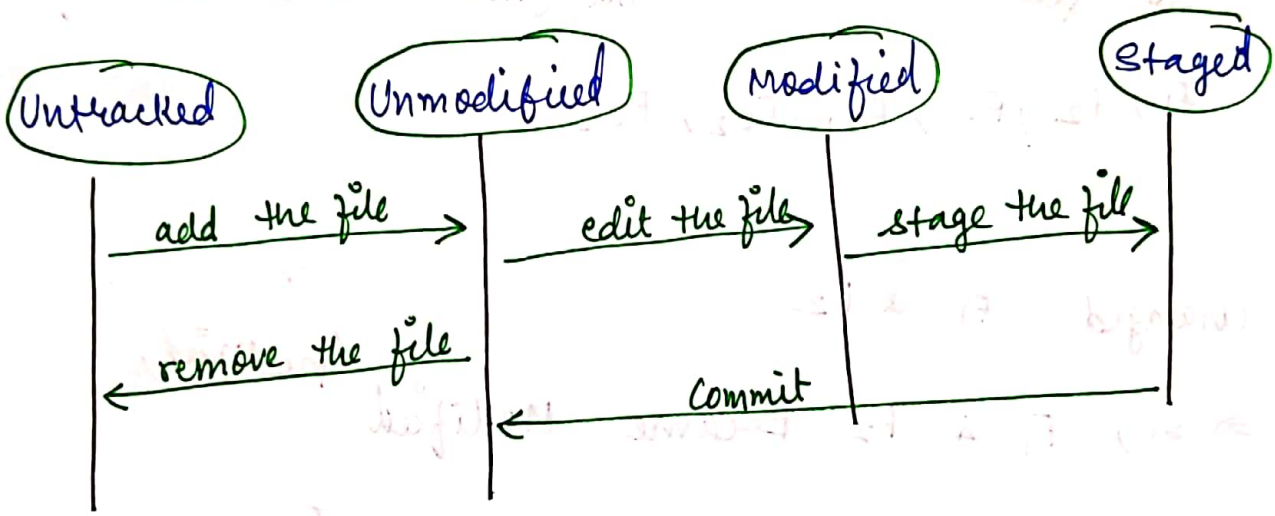
.git folder :⇒
* Hidden
* it contain all information of version control
* jaise jaise we give command of git, then it keep giving files (files : jo bhi version magge)

* Eg: 24 march ko file kaise dikte the if commit was done on 24 march, it will show.

• .git ⇒ file ugalta hai aapki working directory me

* When we start to track our project, an empty repository is created. (it contains nothing) ⇒ contains : untrack files only

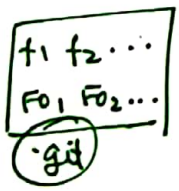
File Status Life-cycle



1) * $ git init ⇒ isko git repository bna do

↓

empty git repository will be created



2) Initially ; files : f1, f2 ... ; folders : Fo1, Fo2 ....

are Untracked. (by default initially)

(Untracked)

(3) $ git add (file name)

⇒ to add our file in our stagging area

⇒ now, we'll come to unmodified section

or

$ git add --a ⇒ (to all all files & folder)

⇓

will come to stagging Area

⇓

All files become 'Unmodified'

(Unmodified) ⇓ now tracking the file

F1, F2, F3, Fo1, Fo2, Fo3

(4) Changed F1 & F2

⇒ So; F1 & F2 became Modified (modified)

⇒ F1 & F2 are modified bhi hai

+ Unmodified bhi hai

(⇒ Staged bhi hai
+ Modified bhi hai)

⇒ git add --a [sare file ko staet kia track karna] ⇒ i.e., Stagging area me

dal do, then we have changed F_1 & F_2 - files.

F_1 & F_2 ka wo version jo staging area me data tha, wo remove nahi hogi (wo as it is rahega) and usi ke saath "F_1 & F_2" modified me bhi dikhayega

⇒ Koi bhi file change kia to wo modified me aa jata hai and uske bad; use add karna padta hai to put it in staging area again

⇓

$ git add (F_1 & F_2) ⇒ Staged !!

now,

file status life-cycle in P.C.

Open folder (practise wala)

↓

right click : git bash here

↓

$ git status (confirm whether it is git repository or not)

$ git init (to make it a git repository)

$ git status (git repository created but not tracking any files)

⇒ Untracked stage pe hoo abhi

$ git add --a (to add all the file to git repository)

$ git status (Changes to be commited)

⇒ ab files staged ho gayi

taken 1st time start kia folder KO track karna isliye we say it :

Unmodified stage

now, all these files are in staggeing area.

↓

Now, Change first.txt

$ git status (we'll see first.txt will be present in both stagging-area + modified area)

new file : first.txt → stagging area
modified : first.txt → modified area

⇒ dono jagh bzc phele maine stagging me data, then we have changed our file

stagging area: commit karne ke liye ready

↓

jaise hi we write git commit.

they'll go in commit

↓

even if we modify them, the file that were supposed to go in Commit they'll go in commit section.

(next snapshot)

↓

If you have changed first.txt, but git will ignore them. Git will say I have kept two 3 in staging area, only these 3 will go in commit area

↓

$ git add first.txt

↓

$ git status (new wali first.txt will be staged now in the staging area)

Now, if we commit now wali first.txt will go there along with others file

Video - 7

Ignoring files in Git

Open that folder :- practise folder

↓

right click : git bush here

↓

$ git status (3 files we'll in stagging area)

$ git commit -m "git tutorial 7"

$ git status (working tree clean)

⇓

(working directory clean)

* $ touch error.log (it will create a file with name → error.log

↓

add something or write something to it

⇒ we only change our code in any software, so we want to ignore our log file.

But as a part of software, log file is generated.
But we want to ignore this log file.
whenever this log file gets updated, we don't want our git repository to get affected bzc of it

$ git status
(Untracked files : error.log)

Now, we want to make our tree clean and don't bother about log file

$ touch .gitignore (creates gitignore file in our folder)

$ git status
(Untracked : .gitignore
error.log

Now,
open .gitignore file from our folder
↓
write :⇒ error.log
↓
(ctrl + s)

$ git status (untracked : .gitignore)
⇓
error.log gets automatically ignored by git

git has ignored : "error.log"

since we have written error.log in out .gitignore folder.

$ git add --a (add the files to stagging area)

or $ git add .

$ git status (new file : .gitignore)

↓

.gitignore come into stagging area

$ git commit -m "Added gitignore"

(commit ho gaya file)

#Note :#

Even if we change our error.log file then also git won't be affected by it Error.log is present in .gitignore

$ git status

(working tree clean)

Note :- Create .gitignore file & put all the uncessary file which you want git to ignore them just by writing

name of that file in .gitignore file -

Note :-

Now, we have copied that error.log file 7 times.

$ git status

Untrack: copy(2).log
copy(3).log
:
copy(7).log

→ But we don't want to see it

open .gitignore file from folder

↓

write: *.log → any .log file will come here

↓

$ get status

(modified : .gitignore)

⇒ all .log file will get ignored by get

Now,

Create a folder : "dir" ⇒ name

↓

put some .log file in "dir" folder

⇒ there might be some folder (not necessary only files will be there)

⇓

Both (file + folder) present

* $ git status

(untracked file : dir/)

*Now if we want to ignore this "dir" folder

↓

open .gitignore folder

↓

dir/

↓

then it will ignore the content of

dir_folder

$ git status

(modified : .gitignore)

Create new folder (directory) ⇒ "static" name

↓

in static folder → create dir folder

↓

create a notepad file

↓

name = " this is a text doc .txt "

* $ git status

(modified .gitignore)

it will not show about this new file bzc we have ignore " dr/ "

Now,

create again a text file in static folder → tex.txt

* git by default ignores blank folder

static folder blank

↓

bzc 'dr' it contains already present in .gitignore

* $ git status
(Untrack: static/)

* $ git add --a
* $ git commit -m "added static"
* $ git status
(working tree clean)

* if we have changed/added content in static/dir/.txt file

* $ git status
(working tree clean)
bzc it is ignored by git

.gitignore
- dir/ → dir ka folder
- /dir/ → sirf outer wala dir folder ko ignore

* $ git status
(Untrack: static/dir/

* add this to .gitignore:

```
*.log
dir/
```

*$ git status
(working tree clean)

Note:-

*.log → all .log files
/dir/ → Bahar wala dir
static/dir → static ke andar wala dir

Video - 8

Showing Changes b/w Commits, Stagging Area & Working Directory

Continue with the same folder :⇒

* $ git status → changes not stagged
  (modified : .gitignore ; static/tex.txt)

* $ git add . (add all file to stagging area)

* $ git status
  (changes to be committed
  modified : .gitignore ; static/tex.txt)

if we have changed some file
↓
then it will show in both :
(Stagging area + modified section)
↓
open static → tex.txt
↓
add some content to it
↓ ctrl + save (S)

* $ git status

changes to be commited :

staging area { modified : → .gitignore , static/tex-txt

changes not staged for commit :

modified section { modified : static/tex-txt

⇒ ek bar hamne staging area me file staged kar di ⇒ commit ki tayari (to take snapshot of the file)

even if we edit then the edditted wala file in go to modified section But, the staggeng area will remain same.

But if we give command

git add --a

⇓

then both : static/tex-txt will get merged.

* $ git diff (compares working directory to stagging area)

⇓

stagging area
<-- a/static/tex.txt] 2 version of static/tex.txt
++ b/static/tex.txt

working directory
@@ -1 +1 @@
-asdasd. → lines removed from file ⇒ stagging Area
\no newline at the end of file
+harry bhai → lines added to file ⇒ working directory
\no newline at the end of file

---

* $ git add .

* $ git ~~diff~~ status
(staging + working ⇒ same)
green colour :-> stagging Area

* $ git diff
(no output :
bzc everything is in stagging Area)

* $ git diff -- staged [git diff --staged]
(compares the prev. commit to stagging area)

prev. commit ⇒ "added commit"

⇓

① -dir/ ⇒ removed → prev. commit

+/dir/ ⇒ add
+ static/dir ⇒ add] present staging Area

② tex.txt

↓

+ harry Bhai → add kia phele kuch na tha. time

+ $ git status

(changes to commit:
modified: .gitignore, static/tex.text)

Video - 9

Skipping the Stagging Area

Open your git practice folder

↓

open: git Bash here

↓

* $ git status
* $ git add .
* $ git commit -m "This is a good commit"
  + comint text → write information regarding what you have changed
* $ git status (working tree clean)

⇓

come Back to folder

↓

change first.txt

↓

write : this is changed now

⇓

* $ git status

(modified : first-txt)

changes not stagged for Commit

⇓

add: second.txt in the folder

* $ git status

× changes not stagged

⇒ modified : first-txt

⇒ Untracked : second.txt

* $ git commit -a -m "Direct Commit"

(to commit our file directly without staggeng)

* $ git status

Untracked : second.txt

→ tracked file ko commit

→ untracked file ko specifically add karna hoga commit me

* $ git add second.txt

* $ git status

(changes to commit:
new file: second.txt)

* $ git commit -a -m "added second"

(2nd.txt directly gets committed without staging)

* $ git status

(working tree clean)

* $ git log

(save commits so we will shown here)

use arrow key to move up-down

⇓

Press "q" to ⇒ exit it

Video - 10

Moving & Renaming Files in Git

* $ git status
(working tree clean)

⇓

Rename : second.txt as third.txt

* $ git status
(deleted : 2nd.txt
Untracked : 3rd.txt)

⇒ git ko lag raha aapne file dilete ki then dosri file add ki

But we have just renamed our file :

* $ git add. (staging area)

* $ git status
(renamed : second.txt → third.txt)

* $ git commit -m "renamed 2nd to 3rd"

Now delete third.txt (from folder)

↓

$ git status

(deleted : third.txt)

Now,

do (ctrl + z) in folder to bring back third.txt

* $ git status

(working tree clean)

* $ git rm third.txt → to delete/remove any file

(rm 'third.txt')

⇒ rm = remove

* $ git status

(deleted : third.txt)

⇓

⇒ removed the file + stagged it

⇒ if deleted manually ; then after delte we have to staged it.

* $ git commit -m "removed bad file 3rd.txt"

↓

message can be any but generally it written such that

it should comment/ specify about it-

→ to rename any file

* $ git mv first.txt first_renamed.txt

* $ git status

(renamed : 1st.txt → first_rena.txt)

mv = ~~rename~~ move

↓

it will rename + put it in staging area

↓

basically first.txt ko first_renamed.txt me move kar do

⇓

[renamed + staged]

* $ git commit -m "renamed"

* $ git status

(working tree clean)

New open :⇒ .gitignore file (from folder)

⇓

*.log
/dir/
static/dir
db.accdb ⇒ add (write)

⇓

* $ git status
(modified : .gitignore)

* $ git add .

* $ git commit -m "changed gitignore"

⇓

open db.accdb file

↓

Create : table

↓

write something

↓

(ctrl + S)

↓

close it

* $ git status

(modified : db.accdb)

⇓

Locha: ideally, git should not track db.accdb bcz we have kept it in .gitignore file

Lekin: we were before hand tracking this file i.e., before putting it in .gitignore ; jab humne apne .gitignore ko change kia, tab bhi wo track hi ho rahi. To stop track of it, we have to explicitially untrack it.

→ to stop track of any file

* $ git rm --cached db.accdb

(stop tracking of db.accdb)

* $ git status

(deleted : db.accdb)

⇓

actually me delte nahi kri, me have untracked it

* $ git commit -m "removed db.accdb"

* $ git status
(working tree clean)

⇓

open file: db.accdb

↓

add text in other table

↓

ctrl+S → close this file

* $ git status
(working tree clean)

↳ we have said git not to track this file :⇒ db.accdb

→ Also it won't show in untrack bzc ye .gitignore me hai

⇓

Now)
remove db.accdb from .gitignore

⇓

* $ git status

(modified : .gitignore
Untracked : db.accdb)

Video - 11

Viewing & Changing Commits in Git

cw to git practice folder

↓

open git bash here

↓

$ git status (modified .. , untracked ...)

$ git add . (to add all file to stagging area)

$ git commit -m "for tutorial" (commit)

$ git status (working tree clean)

$ rm -rf_.git → to delete a git repository

↓

just space

$ git status (not a git repository)

↓

open google : pandas github

↓

github panda page opens up

↓

click : clone or download

↓

copy the url (ctrl + c)

↓

$ git clone (ctrl+v) : mypanda

folder name

(to clone any software code from internet)

remote :- internet (github)

1. receive
2. resolve
3. update

* $ git status (not a repository)

⇒ git clone se ek naya folder bana, wo ek git repository hai (i.e., my panda - git repository hai)

→ my panda ka bahar wale git repository nahi hai

⇓

mypanda folder is a git repository

⇓

open mypanda folder

↓

right click : git bush here

* $ git status
(working area clean)
↓
it's git repository

(Your branch is up to date with 'origin/master'

* $ git log
(to see all the commit)
↓
press 'q' to exit

To see what has been changed in the file [diff:- difference → changes]

* $ git log -p
(along with commit, what thing has been removed from it)

⇒ kaun si line add ya remove hui hai
↓
press 'q' to exit it

* $ git log -p 2 → jitna no. → utna commit
(it will show 2 commit along with their diff)
only

* $ git log --stat
[to give summary of all the commit in short]

* $ git log --pretty=oneline
(all commit in one line
i.e., each commit in 1 single line)
↓
press 'q' to quit/exit

* $ git log --pretty=short
(show all the commit in very short :⇒
it will show:
commit →
author →
message
} no extra detail

→ used to search any particular commit in any program

* $ git log --pretty=full
(give more detail than prev. one i.e., =short wale se)

Author :- jisne file banai (gave birth to file)
Commit :- jisne file me change kia

↓

press 'q' to exit

* $ git log --since=2.days
(last 2 din me jo kaam hua usko wo show karega)

* $ git log --since=2.weeks

* $ git log --since=2.months

* $ git log --since=2.years

* $ git log --pretty=format:"%h--%an"

an = author name
h = abbreviated commit hash

Open google : type ⇒ git scm useful options for git log format

↓

git log documentation

↓

scroll :⇒ pretty =format :

↓↓

" %h -- %an "

↓↓

abbreviated hash commit — author name

Similarly :-

ae ⇒ author email

for more such command see in this git documentation

↓↓

⇒ to apply filter in git log

To change | ammend Our Commit

* $ git status
* $ git log -p -1 (1 commit show karega)

press 'q' to exit

↓

open folder of mypanda

↓

↓

Authors.md - notepad file

↓

add / write :⇒ this is a change

↓

ctrl + S

↓

* $ git status

(modified : AUTHORS.md)

↓

we want to merge all the commit to Simon Hawkins

↓

* git commit -amend

(to change our commit)

↓

An Editor will open up

↓

it will contain the same message as written by Simon Hawkins

⇓

top line edit !

Harry bhai ka msg + LN : remove simplejson (#29169)

To edit in this editor

↓

Press 'i' on your keyboard ⇒ to edit (write)

Press "escape" + ":wq" ↵

↓

exit from this editor

* $ git log -p -1

(Now, you'll see commit message has been changes/amended as done in the editor)

* $ git status

(modified : AUTORS.md)

* $ git add .

(to stage all the files)

* $ git status

(working tree clean)

* $ git commit --amend

↓

An editor will opens up

↓

Press 'i' from keyboard

↓

'I' cursor will blink

↓

type : added now Harry Bhai ka msg + LN - - - -

↓

Press " escape " + " : " + wq ↵

↓

it will be then added to our commit.

* $ git log -p -1

space hai

(you'll see commit will be amended)

Video - 12

Unstaging & Unmodifying Files in Git

⇒ delete mypanda folder

↓

in git practise folder

↓

open: git bush here

$ git status

(not a repository bzc we have deleted git repository previously)

$ git init (create git repository)

$ git status

(Untrack files: - - -)

$ git add .

$ git commit -m "Initial Commit"

$ git status (working tree clean)

↓

Now, change any file

for eg: first-renamed.txt (open)

↓

write/add something

↓

* $ git status

(modified : first_renamed.txt)

* $ git add first_renamed.txt ⇒ (to put file in stagging area)

* $ git status (changes to commit)

⇓

if our file is in stagging area then we need to commit it.

⇓

But now if we want to unstage this file in git :⇒

⇓

* $ git restore --staged first_renamed.txt

* $ git status

(changes not staged for commit

modified :- file.txt)

→ Our file : first_renamed.txt is unstaged now from git

⇓

Open first_renamed.txt file from folder.

↓

delete all its content

↓

type : (abcdef . . .) anything

↓

Ctrl + S ⟶ close this file

* $ git status

(modified : first_renamed.txt)

Note :- we remembered now, we have deleted very imp. information of our file

⇓

we want to get back our original .txt file with prev. content

⇓

we need to restore our file

⇓

we need to run a command to unmodify it & match with the prev. commit

* $ git checkout -- first_renamed.txt

* $ git status

(working tree clean)

⇓

Now reopen that text file :

first_renamed.txt

↓

we'll see that prev. content has come into it.

⇒ "Basically :⇒

jo bhi is file ka phicla commit tha usse hamne isi match karwa dii"

* $ git checkout -- .gitignore
* $ git status
(modified .gitignore)

* $ git commit -m "this"
* $ git status
(working tree clean)

↓

change : first-remaned.txt } file
.gitignore

* $ git status
(modified: 2 file)

* $ git checkout -f] → to remove present commit for all the modified file i.e., any no. of file
→ & match them with prev. commit

* $ git status
(working tree clean)

* $ git checkout -f } → all modifications are removed & we come to clean working tree & the file is matched to prev. commits.

Video - 13

Working with Remote Repositories

Remote Repositories :⇒ ⇒ two repositories jo kisi bhi hosting website or server par rakhi hai

⇒ from there we can pull or push them

Open google : → type : Github

↓

Open official github site

↓

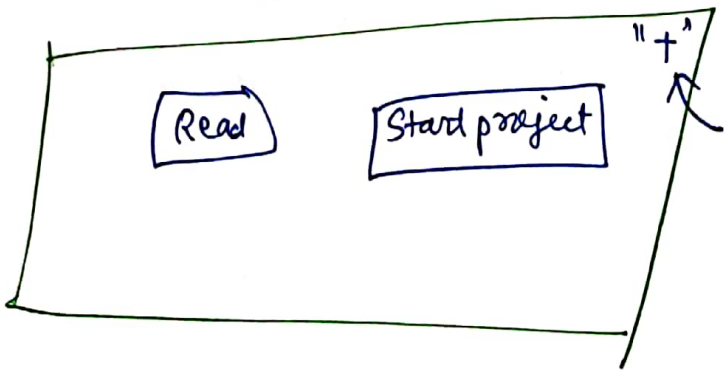
Create an Account

↓

Login in Github

↓

You'll see a screen



click here

↓

New repository

↓

Create a new Repository

↓

fill : Repository name :⇒ Git Tutorial Demo

Owner : Your Name/Account will be shown

Description :-

This is a repository for explaning it on code with Harry.

☑ Public (easy to pull or push)

☐ Private

☐ Create repository

⇓

Open our git practice folder

↓

Open : git bush there

↓

Git Bash

* $ git status (working tree clean)

* $ git remote ⇒ creating a remote here

↓

git is a distributed VCSyster

↓

a server where we can push our code

and any user can pull it along with its code + its history

⇓

so if we pull any code from github to our system],

us code ke sath usme kya changes hue (history) will come into our system

⇒ push :⇒ Only incremental thing will get pushed

⇒ code ko remote repositories me daalna

$ git remote

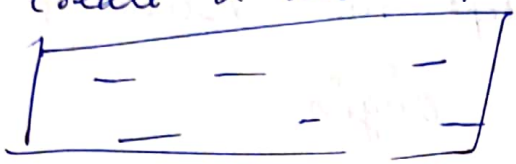
(create remote : ek website or ek repositories jo internet par hai

⇓

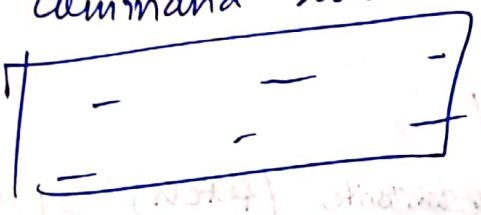
github is a website jo git repositories ko host karta hai

On the github website where you have just created a new repositories :-

create a new repository command line



push an existing repository from the command line



** But we have already made a repository in our pc, so we just need to push it

* To paste in gitbash :⇒ (shift + insert)

Syntax :

* $ git remote add origin (url name (we can change it)) git@github.com:CodeWithHarry/GitTutorialDemo.git

(CodeWithHarry → Username; GitTutorialDemo.git → filename)

⇒ we have kept name of the website as "origin"

* $ git remote

(origin)

↳ we have 1 remote

we have kept the name & the url as the origin

⇓

jaha par hamari git repository hai

* $ git remote -V

origin :- ~~rejected~~ (fetch) → pull

origin :- ~~rejected~~ (push) → push

⇓

abhi ke liye dono same rahege

Now, we have to push our changes :-

* $ git push -u origin master

→ Permission denied

→ Could not read from remote repository

→ make sure correct access rights and the repository exits

⇒ To avoid everyone to push, it's requires a permission

⇓

Go to your git account settings

↓

SSH & GPG Keys

↓

click : New SSH Key

↓

Title : Harry's Personal Computer

Key :

Note :-

Type on google : ssh keys

↓

open : Configure SSH key based secure authentication

↓

ssh keys github

↓

Generating a new ssh key

↓

↓

open your git bush

↓

* $ ssh-keygen -t rsa -b 4096 -C "harry

@code.withharry.com"

email

(Generating public/private rsa key pair)

Enter file in which to save the key: ↵

↓

Overwrite (y/n)? y ↵

↓

Enter passphrase ( ): ↵ basically abhi empty rakha hai

Enter same passphrase again: ↵

* $ eval $(ssh-agent -s)

(Agent pid 1688)

* $ ssh-add ~/.ssh/id_rsa

(identity added)

* $ tail ~/.ssh/id_rsa.pub

( ) → copy this text

↓

Paste it in git website

↓

Under key (SSH keys)

↓

Click : Add

now we have added this key

* $ git push -u origin master ⏎

↓

now it will push the code into the git repository

↓↓

Now, go back to your github account

settings ↓↓

Github : Home

↓

You'll see all the codes along with the commit there

↓

it will show 1 Commit [∵ only 2 had been created]

↓

Now ; aur log isme kaam kar sakte hai ; even they can push or pull

* $ git status
(working tree clean)

↓

create a next text file in practise folder (this.txt)

↓

add any content i.e., write something

↓

ctrl+s → close this file

↓

* $ git status
(untrack file :)

* $ git add . (add your file to staging area)

* $ git commit -m "Added this.txt"

* $ git push -u origin master

↓

Go to git hub website account

↓

refresh the website

↓

After refresh, this.txt will be seen

11)

Koi doosra insaan isi repository ko pull / push karke aapke saath kaam kar sakta hai

Video - 14

Setting Alias In Git

* $ git status
  (working tree clean)

* Alias ⇒ Kisi bade command ki jagah ek chota command likhna

* $ git st
  (not a git command)

* $ git config --global alias.st status
  (giving git command that I'm going to use st in place of status in upcoming command)

* $ git st
  (working tree clean)

* $ git config --global alias.ci commit
(Use : ci inplace of commit)

* $ git ci
(nothing to commit, working tree clean)

TO unstaged any file

~~$ git restore --staged~~

* $ git config --global alias.unstage 'restore --staged'

* $ git unstage this.txt

created an alias such that jo bhi unstage ke baad likho waise => "restore --staged --" ke baad aa jaye.

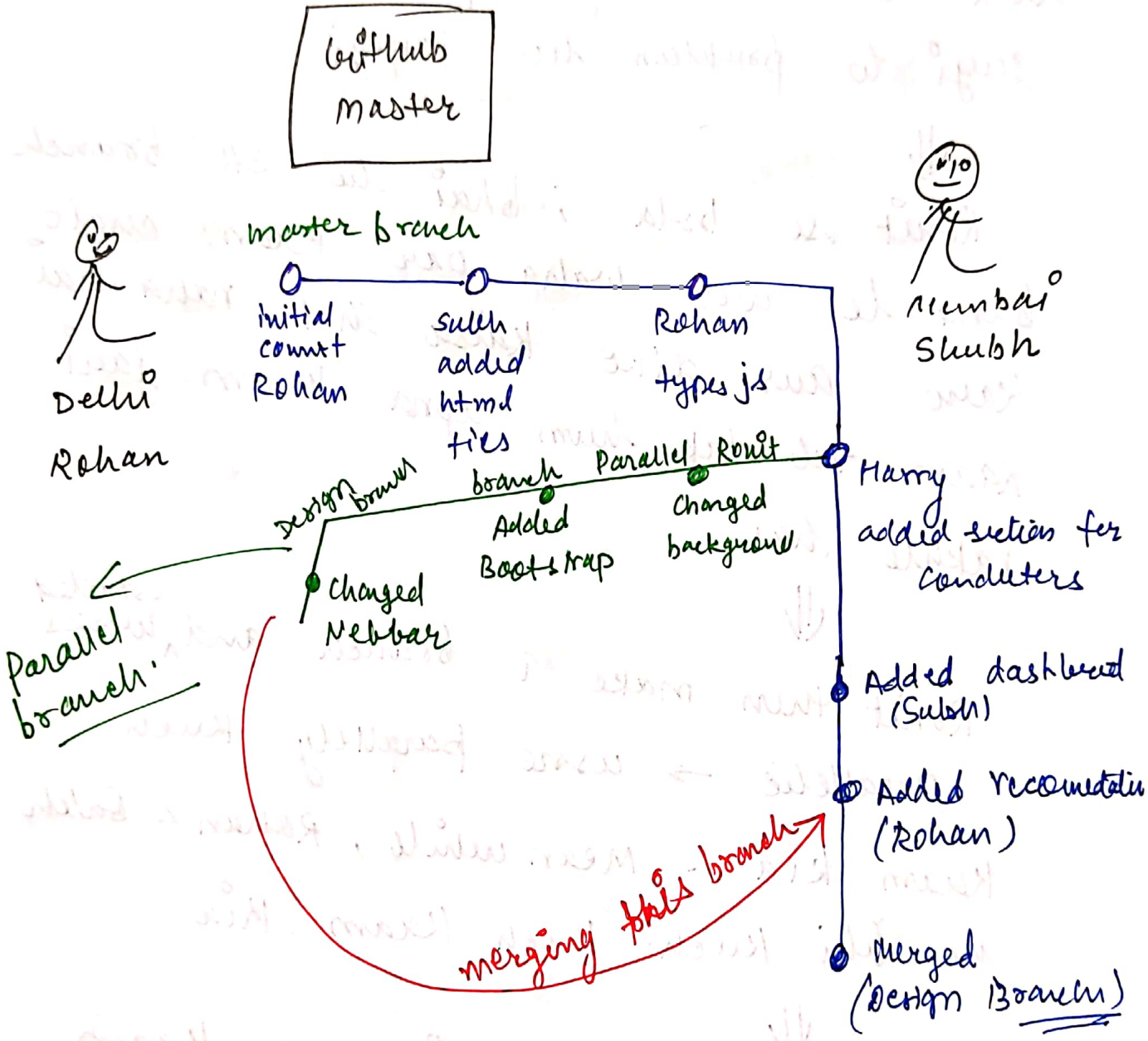
To create alias

* $ git config --global alias.last 'log -p -1'
show the last commit -1.

* $ git last ↵

Video - 15

Creating & Switching Branches in Git



Arati:-

Rohit ko ek idea aaya, purani wali functionality ko rook kar kuch aur karna chata hoo

↓

lakin Rohan aur Subh ne socha agar kuch

gaubar hua to dikkat hai, but we can roll back, but hum abhi production me chal rahe hai; hamari website abhi kachi ho chuki hai agar down ho gayi to problem ho jayegi

⇓

Rohit se bola; bhai tu ek branch bana le aur waha par kaam chalo. Karo aur dikho kaisa chal raha hai kaam tab tak hum apna kaam jari rakhte hai

⇓

Rohit then make a branch and started working parallelly → usne kuch kaam kia. Meanwhile, Rohan & Saleh ne bhi kuch-kuch kaam kia

⇓

Rohit ne bola bhai mera kaam ready hai; check it

⇓

Rohan & Saleh liked his work.

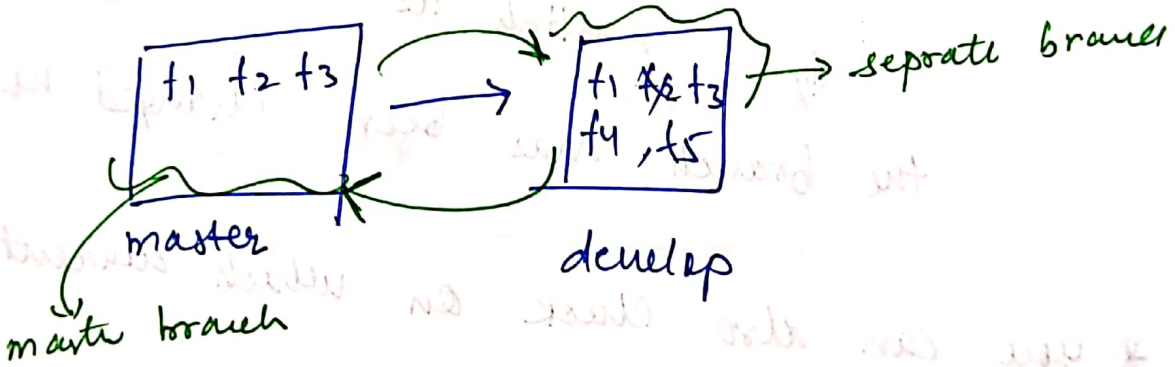
Now, they want to merge Rohit branch to their

* master branch ke saath kiliear nahi kartl. Make 1000 parallel branch. Bzc if master branch gets affected bad then website will go down.

* make a new branch if it works well then merged it with the master branch otherwise leave it (that branch)

* we can make infinite no. of branches

* when we are in master branch



* we can even make a new branch from branch instead of from master branch. and we can even push into remote.

Open git practise folder

↓

open: git bush here

⇓

* $ git status
(modified : - - -)

* $ git restore this.txt (to discard changes in working directory)

* $ git status
(working tree clean)

* $ git checkout -b develop → new branch name
(create a new branch develop & switch us to their)

⇓

the branch has been changed now

* you can also check on which current branch you are by seeing :⇒
above "$" sign.

Now,

* delete all the content inside of the dir folder and rename this folder as "directory"

⇒ Create more folder: static ;
garbage ; templates ;
important files ;

↓

now put: excel & .db file in imp file fold
: put other files in garbage

only: .gitignore, this.txt, this-remind.txt

* $ git status

{ deleted : → db.accdb ,
myexcel.xlsx

Untrack : → important files/ }

⇓

Create any .txt file in directory folder
with something

⇓

* $ git status

{ deleted : → db.accdb
: → myexcel.xlsx

Untrack : → directory/
important files/ }

*$ git add.

* $ git status
(changes to be committed)

* $ git commit -m "Beautified structure"

Suppose :- these changes has been done by some employee ; now Boss says to him to delete 1 log file.
But currently we don't have log file in folder.
log file to purani master branch wali folder me hai.

Switch Our Branch

* $ git checkout master
(switched to branch master

your branch → 'origin/master'

everything gets changed here
but garbage gayi nahi bzc garbage hamara ".gitignore" me chali gayi.

=> garbage gayi nahi kyki garbage to ander sari files .gitignore (log file) me thi

=> .gitignore me jo bhi file hoti hai usko koi bhi fanak nahi padta hai

Now,

⇓

Remove everything from .gitignore file

⇓

* $ rm -rf .git
(remove this git repository)

[Recreating the git repository]

* $ git init
* $ git add .
* $ git commit -m "Initial Commit"
* $ git status
(working tree clean)

⇓

Change first_renamed.txt

* $ git status
(modified)

* $ git add.
* $ git commit -m "change first renamed"
* $ git status
(working tree clean)

⇓

Change ~~this~~ ⇒ this.txt
⇒ first_renamed.txt

directly commit skipping staging area

* $ git commit -a -m "Changed"
* $ git checkout -b develop
(switch to new branch & created a new branch = develop)

⇓

delete :⇒ excel, dbaccdb, this.txt, first_renamed.txt

* $ git commit -a -m "Changed"
* $ git status
(working tree clean)

now change / switch to master branch :-

* $ git checkout 'master'

now we are back into our old state => we will see our deleted file in that same folder

Note:-

(1) $ git checkout -b name of branch

creating a new branch & switch to it

(2) $ git checkout master

switching to master branch

(3) $ git checkout develop

switch to develop branch

(4) $ git branch

develop
master

} all the branch will come here

Video – 16

Branching & Merging a Production Grade Project

Open google :→ vs code

↓

download + install VS code

↓

Open git practise folder

↓

open: git bush here

↓

$ rm -rf.git
(to delete this git repository)

↓

$ get status
(not a git-repository)

↓

close this git bush window

Now,

create a new folder = "My Website"

↓

right click

↓

Open with VS Code

↓

Terminal : tab on the top

↓

click : new terminal

↓

we can also use git command in this terminal

(but we generally prefer - "git bash")

⇓

add a file to v.s. code

↓

index.html

↓

type : "!" ⏎

↓

it will automatically complete the boiler plate

Now :

⇒ google : getbootstrap.com

(library of html & css)

↓

we can create very beautiful website very easily

↓

Copy → starter template under documentation section

↓

paste it in V.S. code

↓

(ctrl+s)

Now install the extensions : in VS code

↓↓

Live Seever ((·))

↓

install

↓

Enable

↓

You'll see at the bottom right

Go live button (if don't get press right click on the VS code coding area)

↓

click : Go live

↓

Your page will be open into your browser

↓

It will only show "Hellow Word"

↓

Go to components section in the Bootstrap page/window

↓

Go to Navbar from the Right

↓

Copy any Navbar (1st/2nd wala)

↓

paste it in place of hellow world

↓

then go to ur created website, you can see Navbar in your created website

Now: come back to our git bush

⇓

Note

* [& create git repository do this inside mywebsite folder] *

* $ git init

[normal directory ko → git project bana diya]

* $ git status

(untrack : index.html)

* $ git add.
* $ git commit -m "Initial commit"

Now we have taken its snapshot

* $ git status
(working tree clean)

Now we want to do some changes but this is running in the production (website is running & people are using it so we don't want to do anything that will shut this website down for a period of time)

⇓

so, parallely we'll create a branch to do some changes in it

* $ git checkout -b toycleanup
(branch name)

(created a new branch & switched to it)

Now, we are in new branch, we can do whatever things we want

bad master branch contains its snap-shot of originally website

↓

add many things to our website code of html

$ git status

* $ git add.

* $ git commit -m "broken features"

* $ git checkout master

↓

after checkout master we'll reach to same state of our website which was before amending it.

↓

To about our link

↓

v.s. code html

↓

link → about (line no : 26)

↓

(ctrl + S)

* $ git add.

* $ git commit -m "Added About"

* $ git checkout feycleanup
(switch to this branch)

↳ Jaha par is branch me chora wahi par milega sab kuch

↓ do some changes in html code.

Line 23 : >Home < span clas

Line 26 : >About US<

Line 33 : >Our Goals<

Line 34 : >Read about our goals<

Line 30 : Inplace of dropdown → Goals

Line 40 : >Know us more <

Line 45 : >search<

↓
(Ctrl + S)

⇒ Now open your website tab

⇒ line 15 : > Codex With Harry <

*$ git status

* $ git add .

* $ git commit -m "fixed the structure"

* $ git checkout master

after this our website will switch to master state before wali

* $ git branch

* $ git checkout trycleanup

website will switch

Now, Our Boss has given some suggestion and after that suggestion merge this to master branch.

suggestion :⇒ Goals

spelling sahi karo

& then merge it to master branch

Correct those spelling in VS code html code

* $ git commit -a -m "Made Changes to dropdown"

* $ git checkout master (switch to master branch)

* $

note:- * clean state aane par hi branch checkout karo *

* $ git merge tryCleanup

(merge failed → merged conflicts)

↓

jo changes - master & tryCleanup me hua usme conflicts hai

↓

* Go to VS. Code html code *

see line : 26 & 30.

Choose any 1

↓

Below line 25:

Click : Accept incoming change

* $ git status (unmerged paths)

* $ git add .

* $ git commit

↓

editor window opens up

↓

Press "i" to write [merge branch 'tryCleanup'

↓

Press "escape" + ":" + wq ⏎ (to exit)

↓

Now, our master branch will look like new final wala

⇒ Basic merge conflict ko merge kia

Merge Conflict :⇒ jab master wala change hua, then other branch change hua

conflict: user ko ask karta which one is imp for you to keep

↓↓

Merge Conflict

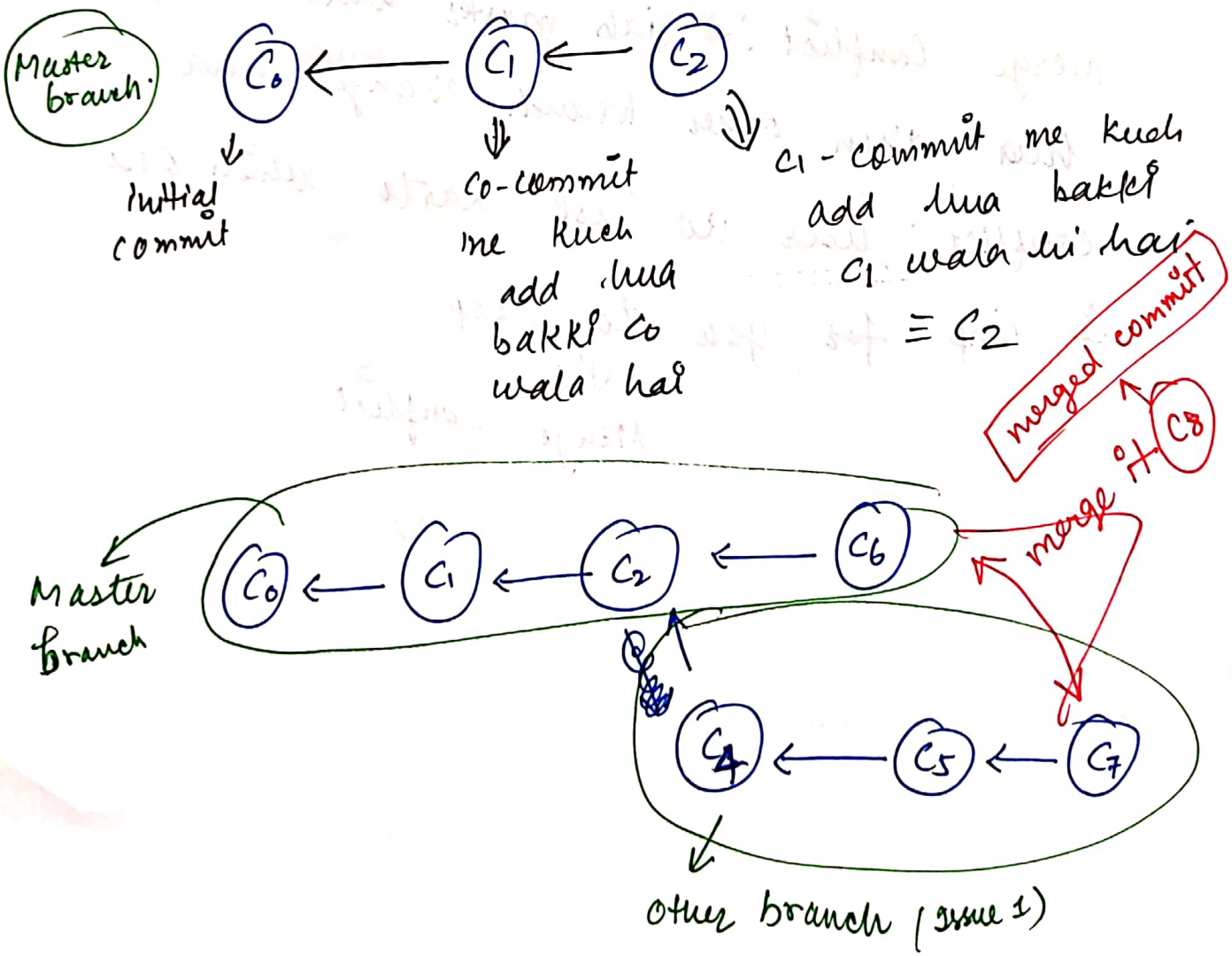
Video - 17

Resolving Merge Conflicts

Merge Conflict

⇒ aapne merge ki koi branch but git ne merge nahi karpaya

C = Commit



⇒ Issue 1 branch resolved a bug that the program/website was facing it.

⇒ Now we want to merge issue 1 branch to master branch

* Note :- * Let's take Ex:

C2 ⇒
<Html>
<Head>
<title> C33 </title>

C6 ⇒
<Html>
<Head>
<title> changed </title>

C7 ⇒
<Html>
<Head>
<Title> B</title>

⇒ it will say title kaun sa loon, bzc title dono me change hua hai then our git will get confused which title to take then there arises "Merge Conflict".

(!)
it will notify the user & ask user to handle it by themself

<<<<< : conflict resolution markers

<<<< <<Head : index.html

=====

>>>>> issue 1 : index.html

* Dono me se kaun sa wale ka title chalye bas wo ye pouch raha hai

to mark merge conflict

($ git add index.html)

Branch Management

$ git branch (it will show all our branches)

⇒ jis branch me hum rahege us branch ke aage star lag jayega

* master → if currently we are in master branch

① $ git branch (show all the branch)

* master
develop
system

we can create any no. of branches in git

↓

git doesn't need space for branch creation

↓

it just save the changes & put the pointer to prev. work

② $ git branch -v
(Show last commit of all the branch)

*master commit hash commit message
develop 9763c "this will fix"

③ $ git branch --merged → already merged branch
(show all the branches which we have merged)

$ git branch --no-merged
↓
not already merged branches

④ $ git branch -d develop

⇓

deleting any branch

⇓

(gives error if develop is not merged)

⇓

if u want to delete this branch
→ d KO capital Karke do

(5) * $ git branch -D develop → delete this branch even if you don't merged it

⇓

no error & branch gets deleted

Practical Demonstration

Open git practise folder

↓

right click : open git bush

↓

* $ git status (not a repository)

* $ git init
* $ git status
* $ git add .
* $ git commit -m "Initial Commit"

⇓

Right click git practise folder

↓

open with VS Code

New file Create

↓

index.html

↓

Place "!" ↵

↓

emit abbreviation

↓

auto complete few Syntax of html

↓

line 7: This is my initial title

Now come to git Bush

$ git status

(Untrack: index.html)

$ git add .

$ git commit -m "Added index.html"

$ git status

(working tree clean)

↓↓

Now go to VS code

line no:→ 7:⇒ "This is my changed title

⇓

Back to git-Bush

⇓

$ git status

$ git add .

$ git commit -m "Added title 1"

$ git status (clean)

⇓

Back to V.S Code

Line no: 7 → This is my changed title 2

⇓

Back to git-Bush

$ git status

$ git add .

$ git commit -m "Added title 2"

$ git status (clean)

⇓

~~Back to VS~~

'Git Bush"

⇓

$ git checkout -b issue1

(Switched to new branch :- 'issue 1')

⇓

⇓

V.S. Code

line 7: ⇒ This is my issue1 title 2 - C4

Git bush

⇓

skipping now staggering Area

~~$ git add~~

$ git commit -a -m "this is C4"

⇓

V.S. Code

line 7: ⇒ This is my issue1 title 3 - C5

⇓

$ git commit -a -m "this is C5"

⇓

V.S code

line 7: ⇒ This is my issue1 title 4 - C7

⇓

$ git commit -a -m "this is C7"

⇓ git

$ git status

(clean)

$ git checkout master

(switch to branch 'master')

⇓

index.html → master branch me aa jayega

⇓

VS - Code

line 7 :⇒ This is my changed title 3 - C6

$ git commit -a -m "this is C6"

Merge issue1 to master branch.

⇓

$ git merge issue1

(merge conflict)

C6 & C7 me diff. title hai

⇓

Kaun sa rakhna hai batado

⇓

V.S. Code

⇓

Click :

Accept Current : 'C6'

Accept Incoming : 'C7'

$ git status

(both modified → unmerged path)

⇓

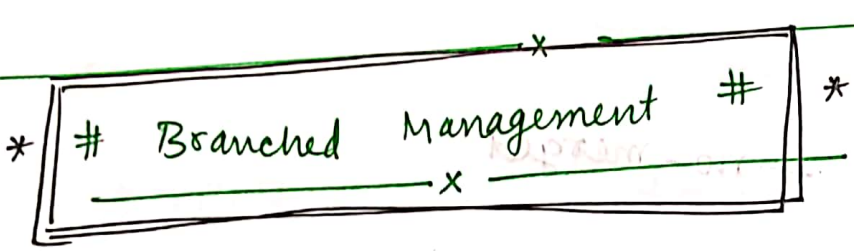
$ git add index.html

$ git status (not committed)

$ git commit -m "Merged issue1"

$ git log (see all the commit)

↓ 'q' to exit from git log.



* $ git branch (to see all the branch)

* $ git checkout -b issue2
(switch to new branch - 'issue2'

* $ git add .

* $ git commit -m "did some gadbad"

* $ git checkout master
(switch back to branch 'master')

* $ git status (clean tree)

* $ git branch
issue1
issue2
* master

* $ git branch -v

issue 1 865d6aa this is c7
issue 2 e3d2406 did some gad bad
* master 6092c46 Merged issue 1

* $ git branch --merged

issue1
* master

* $ git branch --no-merged

issue2

* $ git branch -d issue2
(error bzc nee have not merged issue2)

* $ git branch -D issue2
(to delete this branch)

* $ git branch -d issue1
(deleted branch issue1)

* $ git branch

(issue 2 , * master)

[[* Creating branch in git is not costly.]]

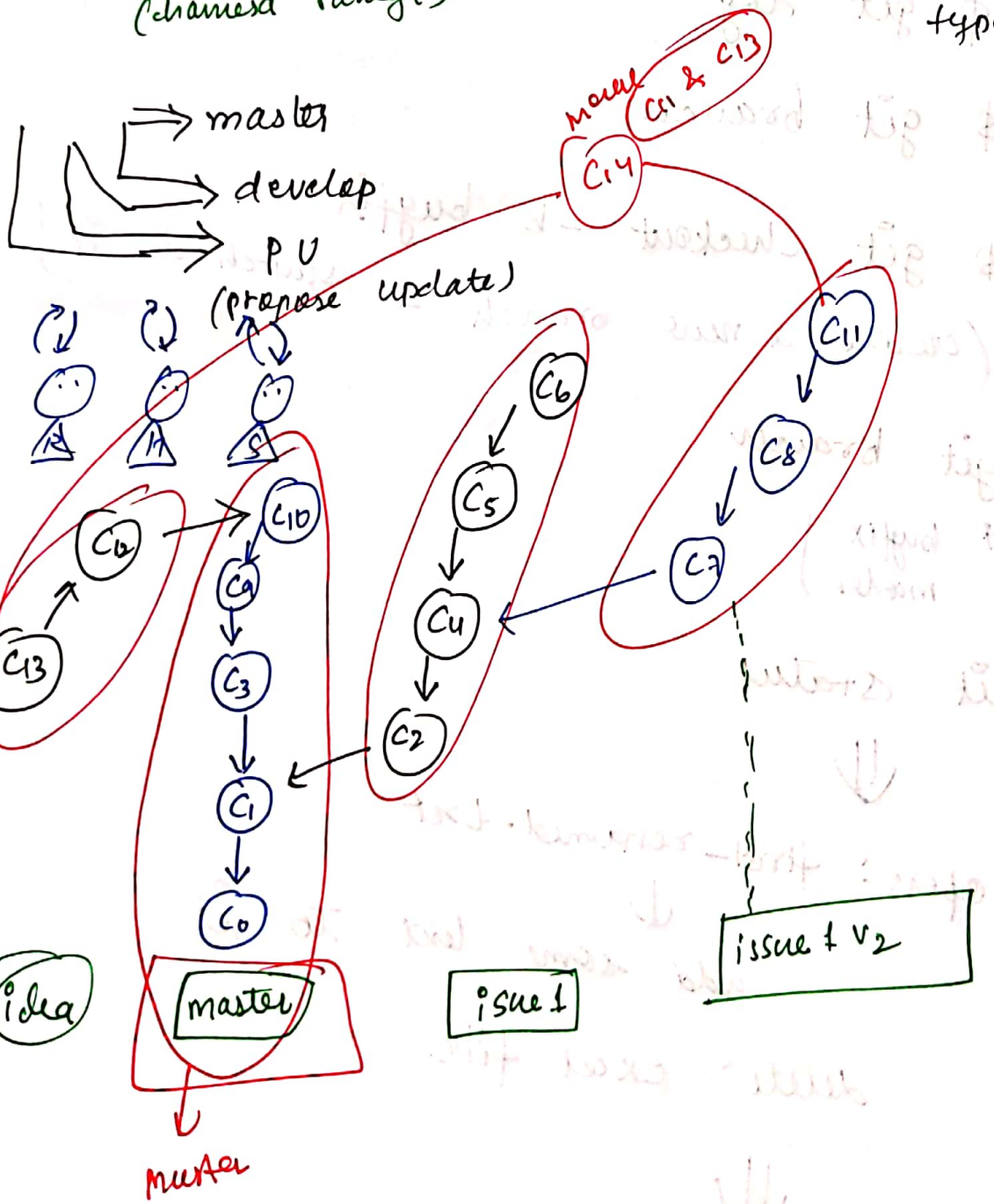
Video - 18

Branching Workflow in Production

Branching Workflow

1.) long Running Branch
(chanesa rahegi)

2) Topic Branches
(short-live)
=> Replace text with types JS



Video - 19

Pushing Git Branches to Remote Repositories

Open git practics folder

↓

* $ git status
* $ git log
* $ git branch
* $ git checkout -b bugfix
  (create a new branch & switch to it)
* git branch
  (* bugfix
  master)
* git status

⇓

open: first-renamed.txt

↓

add some text to it

delete: excel file

⇓

* $ git status
(modified & deleted)

* $ git add .
* $ git commit -m "fixed the bug"
* $ git status
* $ git checkout master
* $ git branch
(bugfix
* master)

* $ git remote
* $ git remote -v
(kuch aaya nahi matlab remote set nahi hai)

⇓
go to our git hub site
⇓
login your account
⇓
click : "+"
⇓
create a new repository

Rep. name : ~~sample~~ demo repository
Discription : This is a demo repository
⇓

click : creating repository

⇓

copy the command from

Push an existing repository

⇓

* $ git remote add origin - (Shift + Insert or right click + paste)

* $ git remote (origin)

* $ git remote -v

origin - - - → fetch

origin - - - → push

* $ git push -u origin master

(To push our code into repository)

⇓

login : username :

pass :

⇓

then it will push your content to your git repository.

refresh your git site

you'll see all your commit under 'Home' section

Click : Branch master

not show branch

Note :- while pushing your content it will not push your branch

for branch we have to explicitly push them.

* $ git status (tree clean)
* $ git checkout bugfix
* $ git push origin bugfix (pushing this branch explicitly)

Now, again refresh your git site

⇓

You'll see 2 branches there

Click : 2 branch

↓

Click : compare

⇓

you can see diff b/w them

Note:- ① jis branch ko push kar rahe hai aap usi branch me rahe (its recommended)

② Before push :- Keep your working directory clean

* $ git status
(tree clean)

* $ git push origin bugfix : mybugfix
(to change the name of bugfix to mybugfix)

⇓

go to git site

⇓

⇓
Reload it
⇓
You'll see 3 branch there

* $ git branch
  * bugfix
    master

remote me track karo
* mybugfix branch ko *

* $ git checkout master
* $ git branch
* $ git branch -d bugfix
  (error → not merged)
* $ git merge bugfix
  (merged if no merge conflict)
* $ git branch -d bugfix
* $ git branch
  (master)
* $ git push origin master
⇓
website :→ reload
* $ git push -d origin bugfix
* $ git push -d origin mybugfix

delete them