Git & Git-Hub

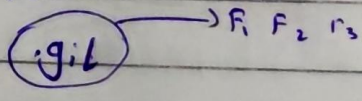
- easy recovery file
- who introduced an issue & whom?
- Roolback to previously working state.

Git ↓

GitHub → Website (For Hosting)
(git or Repostiy on tracking કરી શકાય?
GitHub માં)

Git :-

- Snapshot not difference



all History

- Almost every operation is local.
after that push in server (github)
- Git has intigrity.

1) add user name & email

- git config --global user.name "Amruta"
- user.email "amruta@gmail.com"

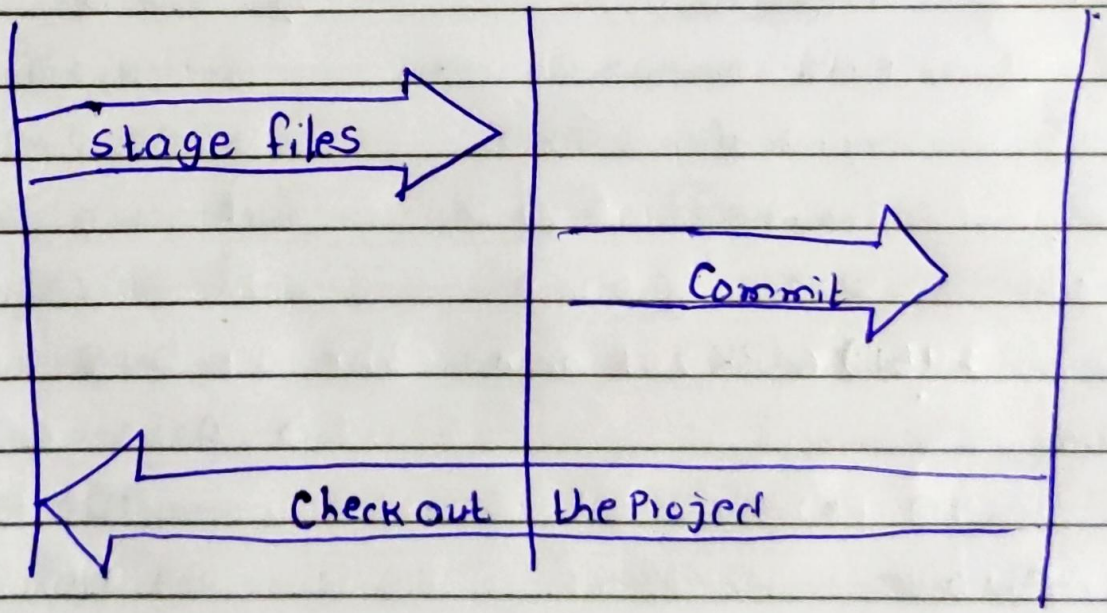
Show name & email

- git config user.name
- user.email

⇒ Git Stage Architecture

It is set of file ⇄ you want to take snapshot (commit/save file)

Working Directory | Staging Area | git directory (repository)



1) git init
(initialized empty git repo in particular Folder)

2) git status
(Show the status of folder)

3) git add -A
(add all files in staging Area)

4) git commit -m "Your Message"
(It take snapshot & save it)

5) git log
(To check which commits i was mad)

6) git add Filename_Name.txt
(add particular file in staging Area)

Danger
7) rm -rf .git
(To stop the tracking in Folder && all things are Delete from git)
* Contents not Delete from Folder. Only .git Folder Delete.

8) git clone https://github.com/tensflow/tensflow.git
(url of other Repo's on gitHub)
(To create a clone of Repo in Local machine)

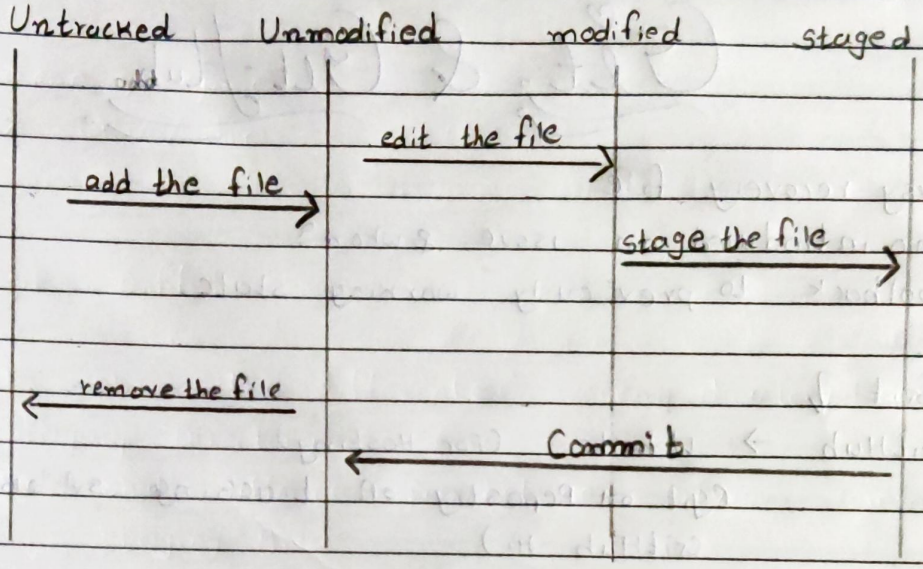
9) Pwd
(show current directory)

10) ls
(show all files in current directory)

11) cd (change directory)

12) (To exit from git log)
Press "Q"

File Status Life cycle



Track
કરવામાં આવે
છે

File modified
થઈ ગઈ

File ને staged
થઈ ગઈ &
unmodified પર
લાવી શકાય.

Git ignore :-

1) touch error.log
(create a file in which the errors of software are store)
(It is not imp file) (we want to this file is not shown in git status when file is update)

2) touch .gitignore
(create .gitignore file) (open file type → error.log)
(Now, error.log file is store in .gitignore) (so, when error.log file is change, we don't see changes in git status)

5

3) git status
(it show only gitignore is untrack)

4) git add -A
(add .gitignore in staging Area)

5) git commit -m "gitignore commit"

So,
જો આપણે .gitignore કરીને file બનાવી & એની અંદર બધી file નાખી દીધી તો એ file ignore થઈ જાય છે

-> માની લો કે કોઈ જ બધી error.log વાળી file Generate થઈ તો
.gitignore માં
↳ *.log લખવું એનો મતલબ જેટલા પણ .log વાળી file હશે એ ignore થઈ જશે.

-> કોઈ folder ને ignore કરવું હોય તો.
Ex folder Name: dir

```
├─ [folder] dir
├─ gitignore
├─ error.log
└─ Hello.txt
```

open .gitignore
type in 2nd line
=> dir/
save .gitignore (No dir વાળુ આખુ folder ignore થશે)

-> git by default folder ને ignore કરે જો એ Blank હોય.

->
```
├─ [folder] dir
├─ [folder] static
│   ├─ [folder] dir
│   └─ txt.txt
├─ Hello.txt
└─ .gitignore
```

-> આ Folder પણ ignore થઈ જશે
Because .ignore માં આપણે dir/ લખ્યું છે એનો મતલબ જેટલા પણ dir વાળા Folder હોય એ ignore થાય.

- બાકી, બહાર વાળુ dir Folder નેજ ignore કરવું હોયતો
  open .gitignore
  change dir/ —to→ /dir/ → save
  હવે, બાકી બહાર વાળા dir ને જ ignore કરશે.

- જો Particular folder ના અંદર ignore કરવું હોય તો Ex dir
  static ના અંદર વાળુ

open .git ignore
↳ static/ dir save

⟹ **Git Diff**

To show the differente between working directory and Staging Directory
or what's Differece ~~pre~~ between previous commit & Staging Area

→ git add -A
→ git status

written in green color {
modified: .gitignore
newfile: static/dir/text Document in dir.txt
newfile: static/textDoc.txt
}

(Above this file is now in staging Area)

(Now modifie static/textDoc.txt

(P.T.O)

(After changes) (Aysh live here)

→ git status

changes to be committed:

green color { modified: .gitignore
newfile: static/dir/text Document in dir.txt
newfile: static/textDoc.txt

changes not staged for commit:

modified: static/textDoc.txt (Red color)

1)→ git diff

(Show Difference working directory & staging Area)

diff --git a/static/textDoc.txt b/static/textDoc.txt

--- a/static/textDoc.txt (stage Area file version)

+++ b/static/textDoc.txt (working Dir. file version)

@@ -0,0 +1 @@

+Aaysh lives here!! (This is add) it is difference

\no new line at end of file

→ git add -A

→ git status

changes to be commited:

modified: .gitignore

newfile: static/dir/text Document in dir.txt

newfile: static/textDoc.txt

(now, there is no modified file)

2)→ git diff

(કશું show નહી થશે) (Because working Dir & Stage Area same છે)

⇒ (Now, Compare Staging Area with Previous commit)

1→ git diff --staged
(Show Difference between stagin Area & previous commit)

⇒ Skipping The staging Area

1) git commit -a -m "Direct commit"
(જો tracked file ને stage કરીને commit કરી દે છે)
(untracked file નહી મારા એ untrack જ રહેશે)

⇒ Moving and Renaming File :-

જો manually તમારી Direct file નું નામ Rename કરીએ
(without using git). & પછી પછી git status Run કરીએ
તો એ કહેશે તમે old નામ વાળી file Delete કરી દીધી &
નવા નામ ની file add કરી દીધી જે untrack file છે.

પછી new name file ને git add -A કરીને
git status જોઈએ ત્યારે એ કહેશે કે તમે
file નું નામ change કર્યું છે. old.txt → new.txt

After that you need to commit

→ Remove File (Delete File)

File ને Direct Delete કરીએ તો એ git status માં લાલ કલરથી deleted : thirt.txt (Red color) એ Delete file ને stage નથી કરે

optimise way

1) git rm File_Name.txt
(File Delete થઈ જશે & automatic આ changes Staged પણ થઈ જશે.)

→ 2) git commit -m "Remove This File"

→ Rename File # optimize way

2) git mv old.txt new.txt
(File ના નામ change થઈ જશે & automatic એ staging Area માં પણ થઈ જશે)

2) git commit -m "Rename This file"

⟹ Ignore tracked file :-

કોઈ tracked file ને Ignore કરવી હોય તો.

1) . .gitignore માં એ file નું name લખી save કરીએ
↳ track.txt

. git status
modified: .gitignore
. git commit -m "modifi gitignore"

હવે, જો તમે ignore માં નાખો એ file (track.txt) ને modified કરીએ.

- git status કરીએ તો હવે પણ એ file track વાળા માં જ બતાવશે

modified.txt

એટલે એ file હજુ પણ track થાય છે.
એને .gitignore માં નાખવાના બદલે પણ

- તો એ file ને untrack કરવી પડશે.

* 2) git rm --cached track.txt
(track.txt બધી file હટે untrack file પણ રહે & git ના working tree માંથી Dele પણ કરશે)
* Local machine માં તો રહેશે જ.

git status
↳ changes to be commited:
deleted: track.txt

git commit -m "Remove track.txt"

⟹ Viewing & Changing Commit in Git

1) rm -rf .git
(Git repository ને remove કરવા માટે)

1) git log
(show all commit's)

2) git log -p (show all commit with diff)
(દરેક commit વખતે જે changes થયા હોય તે જોવા મળે.) જેમકે કઈ line remove થઈ, કઈ line Add થઈ code માં.

3) git log -p -3 (show lates 3 commit with diff)
↳ can specify any number.

4) git log --stat
(દરેક commit વખતે કુલમાં કેટલા ફેરફાર થયા કે કેટલા + insertain થયા & કેટલા - Deletion થયા)

5) git log --pretty=oneline (Author & commit Msg સહ)
(દરેક commit 1 line માં જ દેખાઈ જશે.

6) git log --pretty=short
(shorts માં પણ commit દેખાઈ જશે)

7) git log --pretty=full (Author, commit & commit Msg સહ)
(આમાં Author કોણ છે? commit કોણ એ કર્યું? & commit ની Msg શું છે? એ પણ બતાવશે.)

Author → જેને File બનાવી છે એનું નામ આવશે.
Commit → જેને File ને change કર્યું છે એનું નામ આવશે.

Filter in git log

8) git log --since=2.days
(2 દીવસ માં કેટલા commits થયા હશે તે દેખાશે.)
1.months , 1.years , 1.weeks

9) git log --pretty=format:"%h --%an" (www.git-scm.com/docs/git-log)
(Format માં output જોવા માટે)
Where, %h = abbreviated commit hash
%an = author name.
%ae = author email

10) git commit --amend (જુના commit ને change કરવા)

કોઈ screen open થશે એમાં તમે તમારું commit પહેલા કોઈ commit કર્યું હોય એમા ફેરફાર કરી શકો છો & commit નો Msg પણ બદલી શકો છો.
Edit કરવા માટે → i → દબાવવું

એ screen માંથી બહાર આવવા માટે
↳ press 'esc'
↳ Type → :wq → Enter

43

⟹ Unstaging & Unmodifying Files In Git :-

1) git restore --staged first.txt
(first.txt file Staging Area માં છે (જે commit થવા રહ્યું છે) તે file ને unstage કરવા માટે)
git restore first.txt (Modify માંથી unmodify માં લાવવા)

* 2) git checkout -- first.txt
(Example:- first.txt માં "Hello world" લખીને commit કર્યું છે.
- પછી તે file માં કંઈ changes કર્યા "Hello world yellow world" આવું કરીને file ને save કરી દીધી
- હવે, પાછા આપણે જોઈતું હોય કે commit કર્યું ત્યારે file જેવી હતી એવી પર લાવવા માટે git checkout -- first.txt લખશું તો આખી file માં "Hello world" જ દેખાશે)

* 3) git checkout -f
(working tree ને પાછલા commit સાથે મળાવી દેશે) એટલે કે working tree પાછળના commit માં જેવું હતું એવું પર જશે. બધા changes જતા રહેશે ને પાછળના commit જેવું પર જશે.)

⟹ Working with Remote Repositories :-

Push	Pull (ખેંચવું)
↓	↓
code ને gitHub પર નાખી દેવો	GitHub પરથી code ને laptop માં નાખી દેવો.

1) git remote add origin https://github.com/Ayushp454/GitTutorial.git
(આ website નું નામ આપણે origin આપી દીધું & એને as a remote add કરી દીધું)

2) git remote
origin (remote ને add કરતી વખતે આપેલું નામ બતાવશે)

3) git remote -v
(કેટલા remote ની કઈ link પર push & pull કરવાનું છે તે બતાવશે.)

git push origin master

4) git push -u origin master
(Local machine of folder જેમાં git init કરેલું છે એ Folder નો origin પર gitHub પર push કરી દેશે)
(જેના વડે Data gitHub પર પણ દેખાઈ શકે)

(જો error આવે કે access નથી તો
go to → setting (gitHub setting) → SSH & GPG Keys
→ Title → જે નામ રાખવું હોય તે પછી → Key

=> Key માટે → open git bash → type →
ssh-keygen -t rsa -b 4096 -C "your_email@gmail.com"
↳ Enter → Enter → y

→ eval $(ssh-agent -s)
Agent pid 1688

→ ssh-add ~/.ssh/id_rsa
Identity added

→ tail ~/.ssh/id_rsa.pub
(જે કઈ પણ આવશે એને copy કરીને
Key માં નાખી દેવાનું

Add SSH Key

⟶ Setting Alias In Git :

→ Alias

mean write small command instead of biger command.

Example :

git status ની જગ્યા એ git st આપણે command run કરવા ઈચ્છા તો એના માટે.

1 → git config --global alias.st status

(એનો અર્થ કે જ્યારે હું git st લખું તો એ git status ને run કરે.)

એનો હું status ની જગ્યા પર st કરવા માગું છું.

Commit ની જગ્યા એ ci આવા માટે

2 → git config --global alias.ci commit

3 → unstage કરવા માટે

git restore --staged this.txt કરવા

git " " -- this.txt

⇓

→ git config --global alias.unstage 'restore --staged --'

git restore --staged -- this.txt

git unstage this.txt

Page :
Date : / /

4 Show last commit

→ git config --global alias.last 'log -p -1'

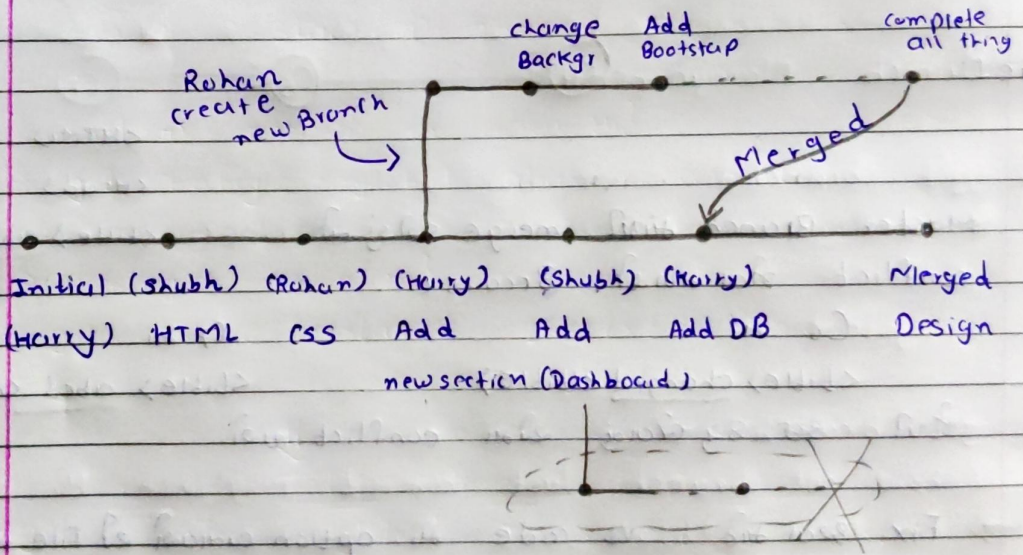
-(6)-

Creating & Switching Branches :-

(Master)

Master Branch એ કોઈપણ main & important branch હોય છે

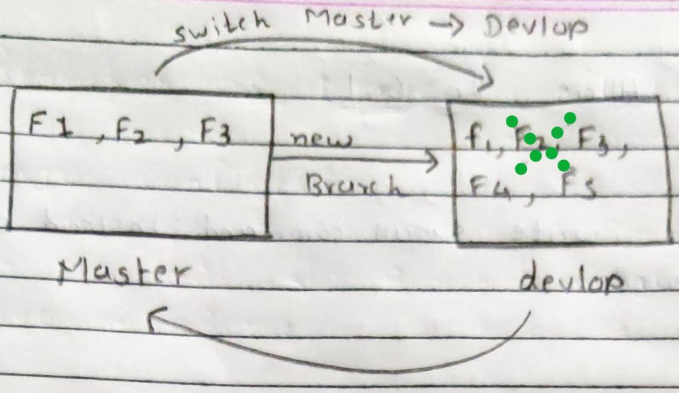
આ 3 એક project પર કામ કરવા માગેલ છે.



Harry also crete new branch But he is deleted because him work in this branch is not better than master.

→ આપણે Master Branch માંથી બધી વહી Branches કાઢીને parelly work કરી શકીએ છીએ અને અને last માં Merge કરી શકીએ છીએ. જે વધુ બહાર પાડી દો

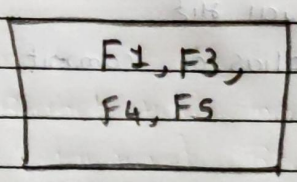
→ આપણે Branche માંથી પણ બીજી બધી Branches કાઢી શકીએ છીએ.



Branch switch કરવા કરી શકો?
Devlop to → Master

MERGE

After complete all work successfully in Devlop



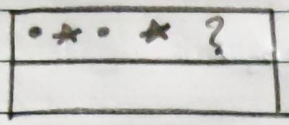
1) git checkout -b devlop
(નવી branch બનાવશે devlop નામની & એની અંદર ફાઈલ કરશે.

2) git checkout master
(devlop માંથી master branch માં switch કરી દેશે.)

3) git branch
(branch ની list બતાવશે & કઈ Branch માંથી હાલ તે green color માં બતાવશે.)

4) git merge devlop (Master માં રાખીને command run કરવો)
(Master branch માં Devlop Branch ને merge કરી દેશે.) | આ દરમિયાન conflicts પણ આવી શકે છે.

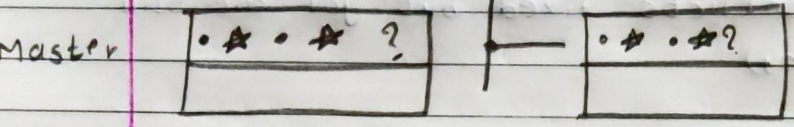
⟹ Branching & Merging a Production Grade Project :-



Master

This website in production

(Boss said Improve Design)



Master

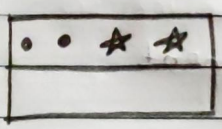
Devlop

(create new branch)

(Make changes)

checkout

git -b devlop



Devlop

(Now merge to Master)

→ go to Master branch

i) git checkout Master

→ Merge Master with devlop

ii) git merge devlop

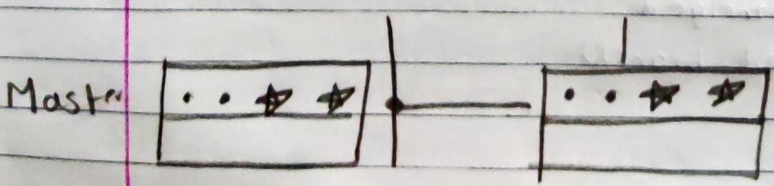
At merge any conflic can be come Then fix it

ii) git add .

iii) git commit

↳ esc

↳ :qw ↳



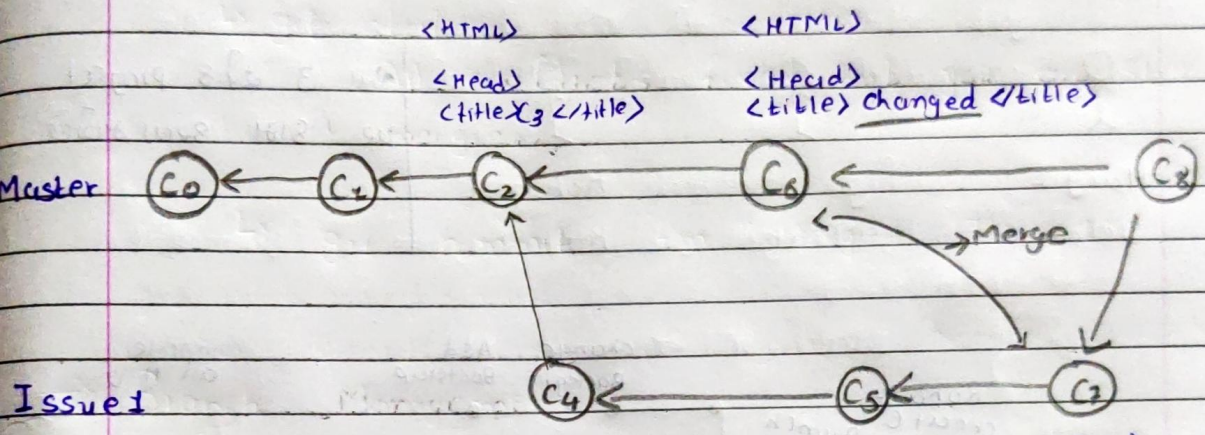
Master

Devlop

=> Branches & Merge Conflicts :-

→ Merge Conflicts :-

કારે કોઈ Branch ને merge કરી પણ git તેને Merge ના કરી શકે.



Master Branch માંથી merge કરીએ તો conflict બોલાવશે. Because

C6 માં
<title> changed </title>

C7 માં
<title> abcd </title>

જેથી કઈ કરવું બોલાવે conflict થશે.

=> Fix કરવા માટે vs code માં option બતાવશે કે File માં accept incoming change, accept current change, add both change.
આમાંથી select કરીને conflict દૂર કરી શકાય કા તો manually પણ કરી શકે છે.

```
<<<<<< Masle: index.html
======
>>>>>> Issue1: index.html
```

→ Fix કરવા માટે

↳ git status કરીશું તો unmerged paths:
both modified : index.html
બતાવશે.

↳ git add .
(staging area માં મોકલવું)

↳ git status
(All conflicts fixed but you are still merging)

↳ git commit -m "Merged Issue 1"

→ Branch Management

1) git branch
* master
devlop
system

2) git branch -v
(show all branch last commit Msg with Has code)
* master 93b13c "This will Fix"
devlop a93ef "This was Remaining"

3) git branch --merged
(show only already Merged branch)

4) git branch --no-merged
(Show non-merged branch)

5) git branch -d devlop
It will delete devlop branch if it is already merged.
otherwise
give error if devlop is not merged & Say: write! -D instead of -d

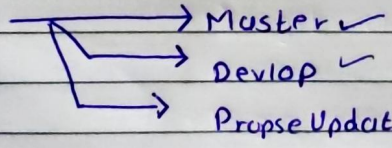
6) git branch -D devlop
It will delete devlop branch if it is not merged

⟹ Git Branching Work-Flow :-

Work Flow

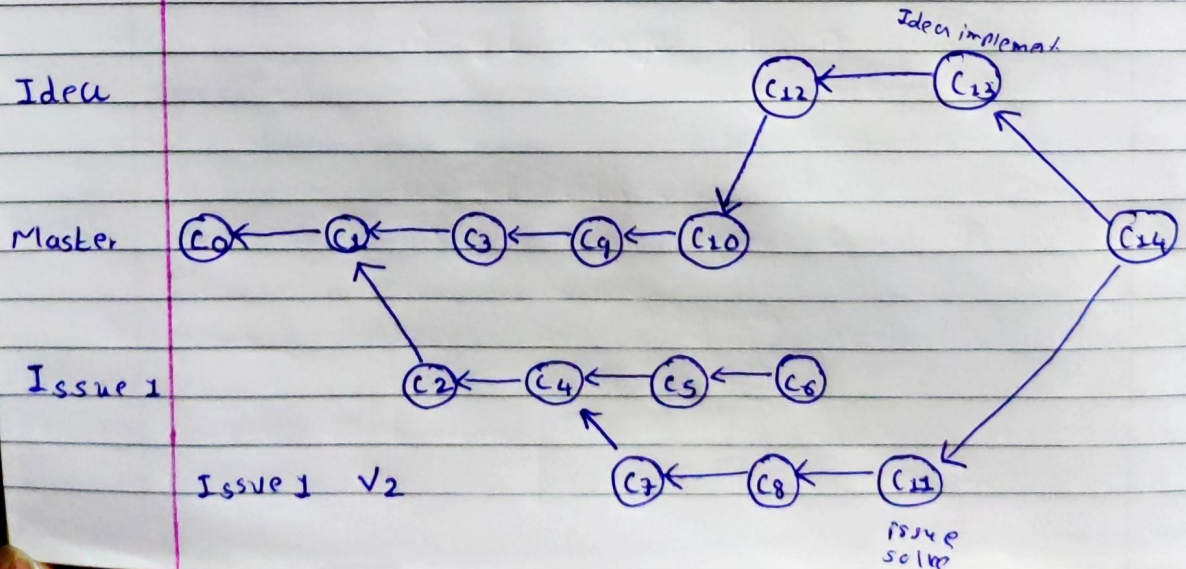
Long Running Branches

Topic Branches



→ Replace text with typed JS

Typed JS Int

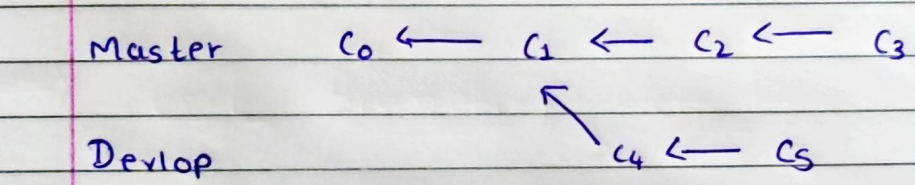


⟹ Publishing Git Branches :

How to push branches on server ?

Branches publish કરવા માટે તેને explicit publish કરવા પડે છે

Example :-



1) git push origin master
(આમાં Master Branch ને server પર મોકલો)

Devlop Branch ને push કરવા માટે આપણે Devlop Branch માં જવું પડશે તેને push કરો

2) git checkout Devlop

3) git push origin Devlop

NOTE:
Push કરતા પહેલા તે Branch ને commit કરી દેવા

optional
⟹ git push origin bugfix:mybugfix
Elaptop માં જે bugfix નામ branch છે તેને mybugfix નામ Remote side પર નવું tracking start કરીદો)

Page :
Date : / /

હવે Branch ને successfully merged કરી લીધું.

તેનો અર્થ છે laptop માંથી Delete કરશે તો Branch ને તો
એ મારી laptop માંથી જ Delete થશે remote માંથી નહીં.

git branch -d bugfix
(Delete branch in laptop)

git branch
* master

git push origin master
(તો પણ Remote server પર bugfix આવી branch દેખાશે)

4) git push -d origin bugfix
(Remote server માંથી bugfix આવી branch ને Delete
કરી દેશે.)