॥ जय श्री राम ॥

twitter:- @vinayakstwt ॥ ॐ ॥

Page No.: YOUVA

Date:

GIT AND GITHUB

Before getting into what is Git and Github, let us understand first that why is git around?

You must be using Chrome or any other browsers to do your research, visit websites, etc. but the Chrome your using now did not looked the same way when it was made. Developers keep on developing it and upgrading it. Suppose developer added a feature, but later want to discard it. He/she cannot do it if he don't have files of previous version saved. This lead to development of Git.

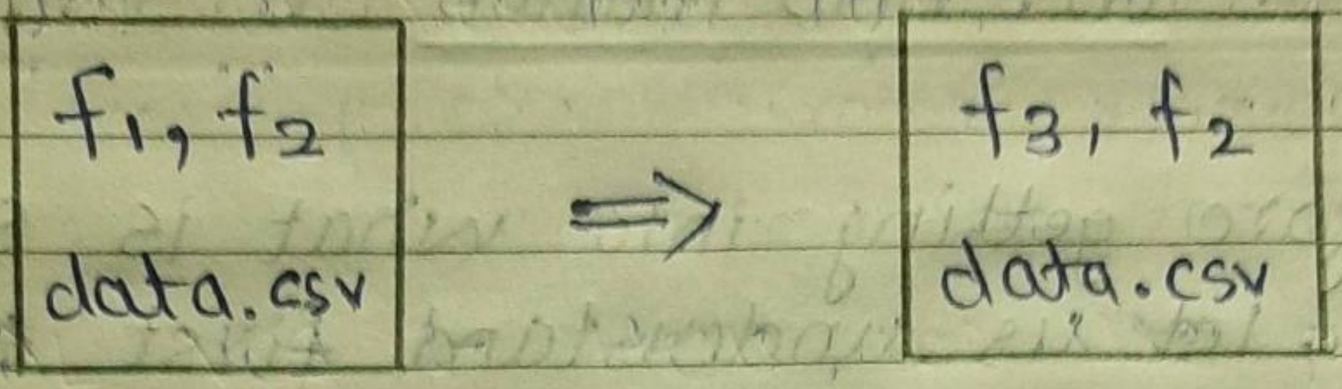
- So what is Git?

↳ Git is a version control system. It keeps track of all the files. Using git one can:-

i) easily recover file

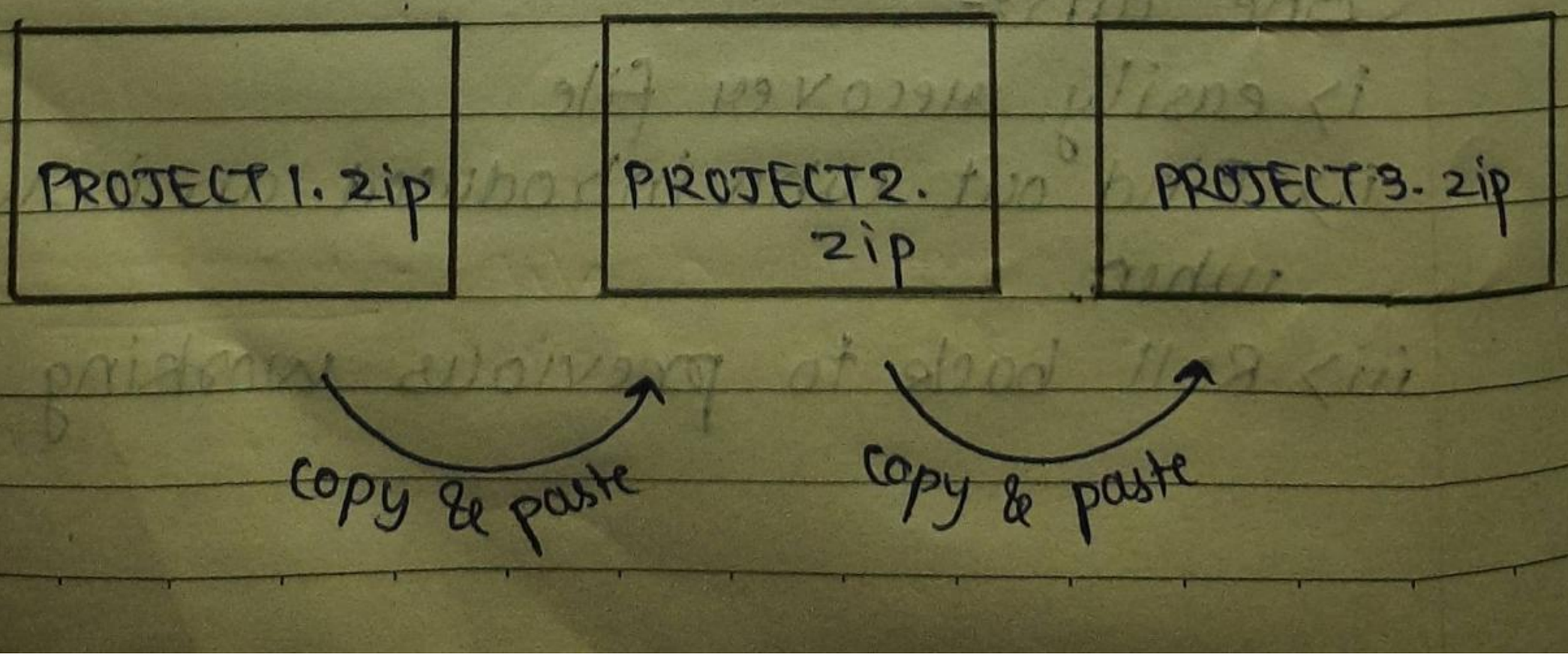
ii) Find out who introduced an issue and when.

iii) Roll back to previous working state.



. Suppose I made a app having version 1 containing files f_1 and f_2 and data.csv which I upgraded to app version 2 and made changes in f_1 and called it f_3. It is very important that I have version 1 files saved so that if anything bad happens. (eg. f_3 not working properly) I can roll back to version 1.

• But still, is git really required?
↳ No!



If I have a project I can copy & paste the enitire project every time I upgrade it. This method is not so good because it consumes spaces. It will require lot of storage if my project is very big.

Morever, it also does not give us information about who made changes and when. Even if we add data to remind its change time, this method is not good because git gives time stamp in seconds i.e. in which second what changes have taken place.

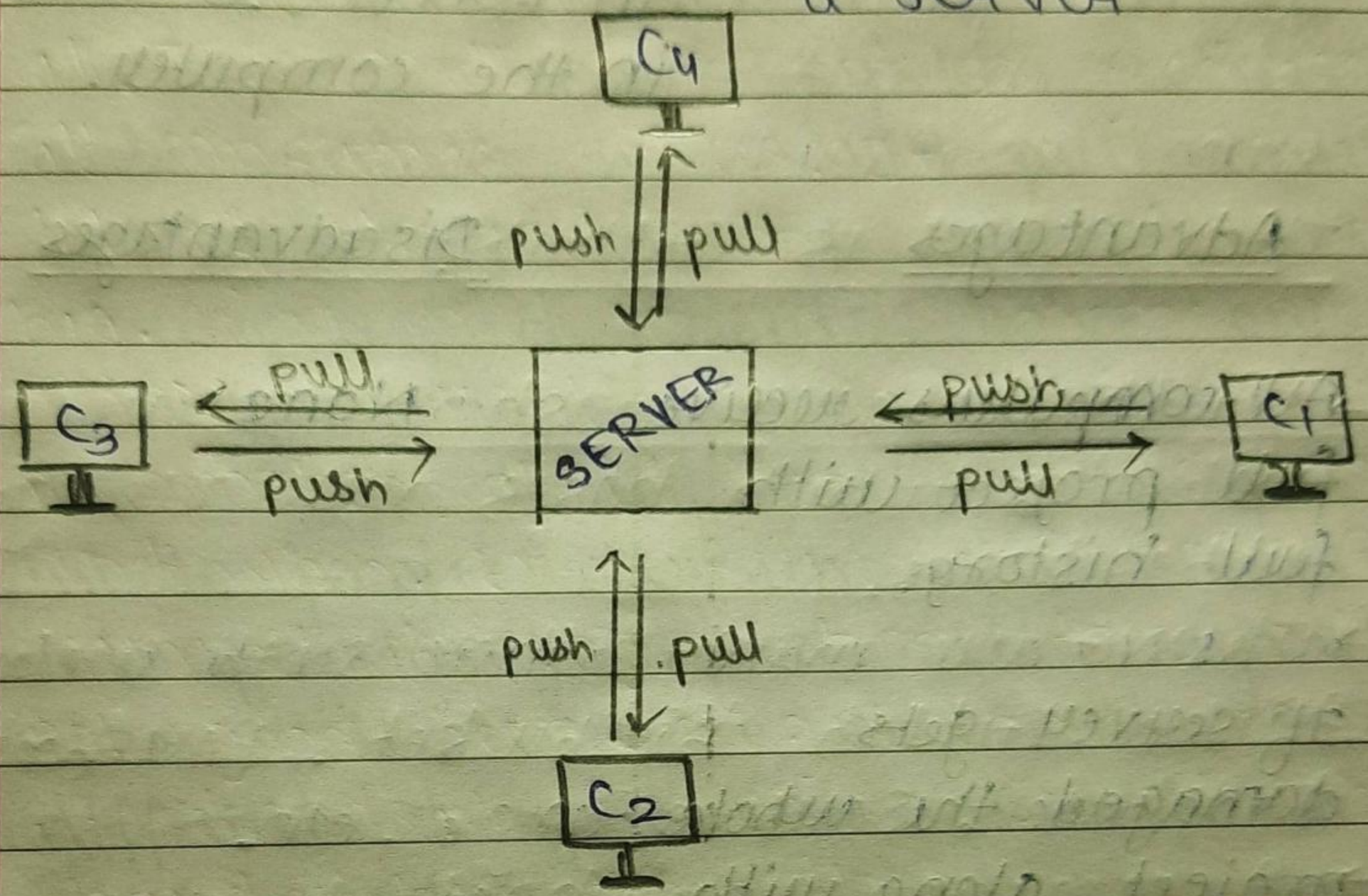
So I depends on you, if you have large storage and money to buy hundred hard disk don't use git and also you cannot send full project along with version history to your teammates. So better understand git and make life simple.

HISTORY OF VERSION CONTROL SYSTEM (VCS)

I] LOCAL VCS :- Here we use databases to keep track of files.

ADVANTAGES	DISADVANTAGES
1) We can roll back to previous version	1) All changes remain in computer only.
2) Can track files	2) If computer / hard drive is damaged all data is lost
3) No stress of pushing or pulling data from server	3) Lot of storage is required.

II] CENTRALISED VCS :- Here data is stored in a server



ADVANTAGES	DISADVANTAGES
1) If computer gets damaged he/she can recover files easily.	1) The copy stored in centralised server is considered to be final.
2) Many developers can work at same time connecting to the server	2) If server is damaged all data is lost
	3) Some files can be recovered from PC but cannot roll back

III] DISTRIBUTED VCS :- Here files are stored in server as well as in the computer.

Advantages	Disadvantages
1) All computers receive full project with full history.	None
2) If server gets damaged the whole project along with previous versions are in computer safe	
3) can roll back (complete backup)	

*CONDITION :- We need smart system which does not occupy large disk space.

Written :- @vinayakctwt

Page No. 7 YOUVA
Date :

If we have SCRS project, in that we don't have source code (it is in KBS) we need smart system which will ignore and will pull only source code and if any file is changed, it should save only that change.

It should not copy & paste whole project which will require lot of storage need. It's repository size will increase and will become diffiult to give full backup to users when needed.

Who created Git and what is the story behind it?

→ Linus Torvalds created Git who was also the creator and main developer of the Linux kernel.

→ In 1991-2002, Linux (OS) development was done in pacthes and archive files. There was no VCS.

→ Bitkeeper VCS offered VCS for free to Linux development

→ But, later removed free of charge status.

→ So, Linus Torvalds created Git which provided free VCS.

- Then what is Github?

↳ Github is a hosting service, a website, which hosts various git repositories.

↳ There are many other such websites like Gitlab, Bitbucket, Perforce, Codebase, SourceForge, etc.

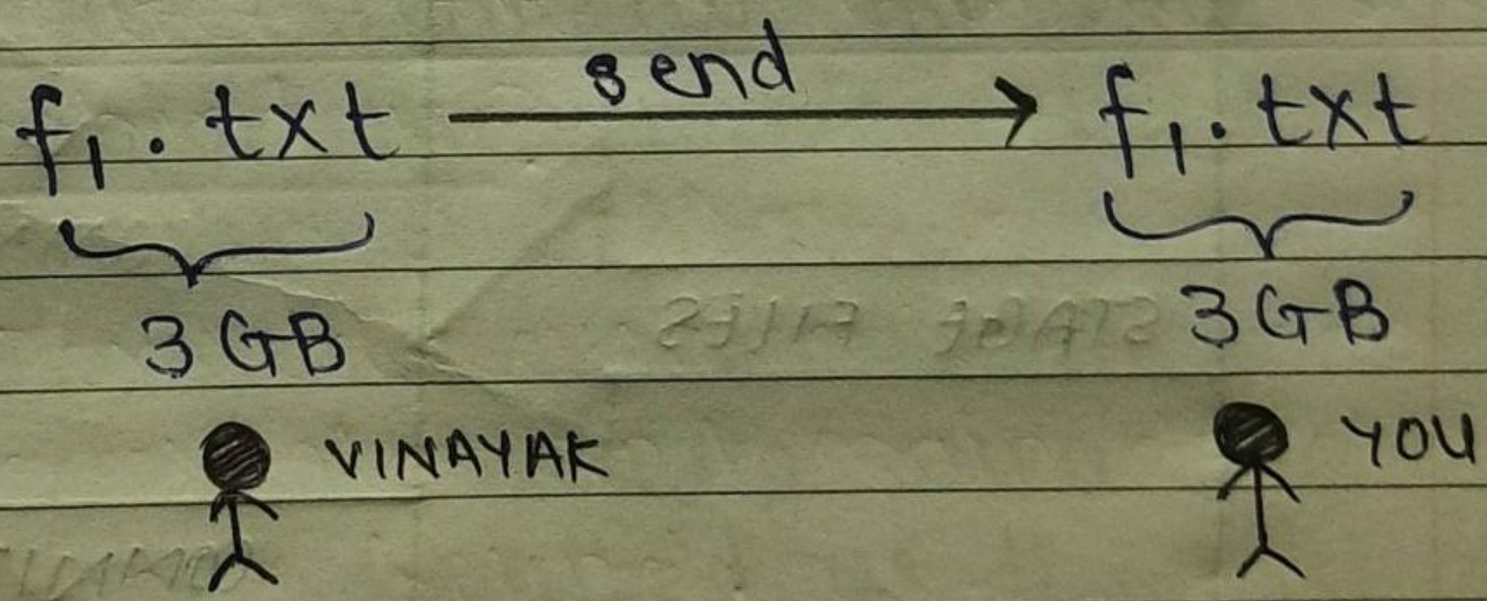
↳ A company can host its own VCS for its product.

twitter:- @vinayakstwt

M T W T F S S
Page No.: 9
Date: YOUVA

Git : Features

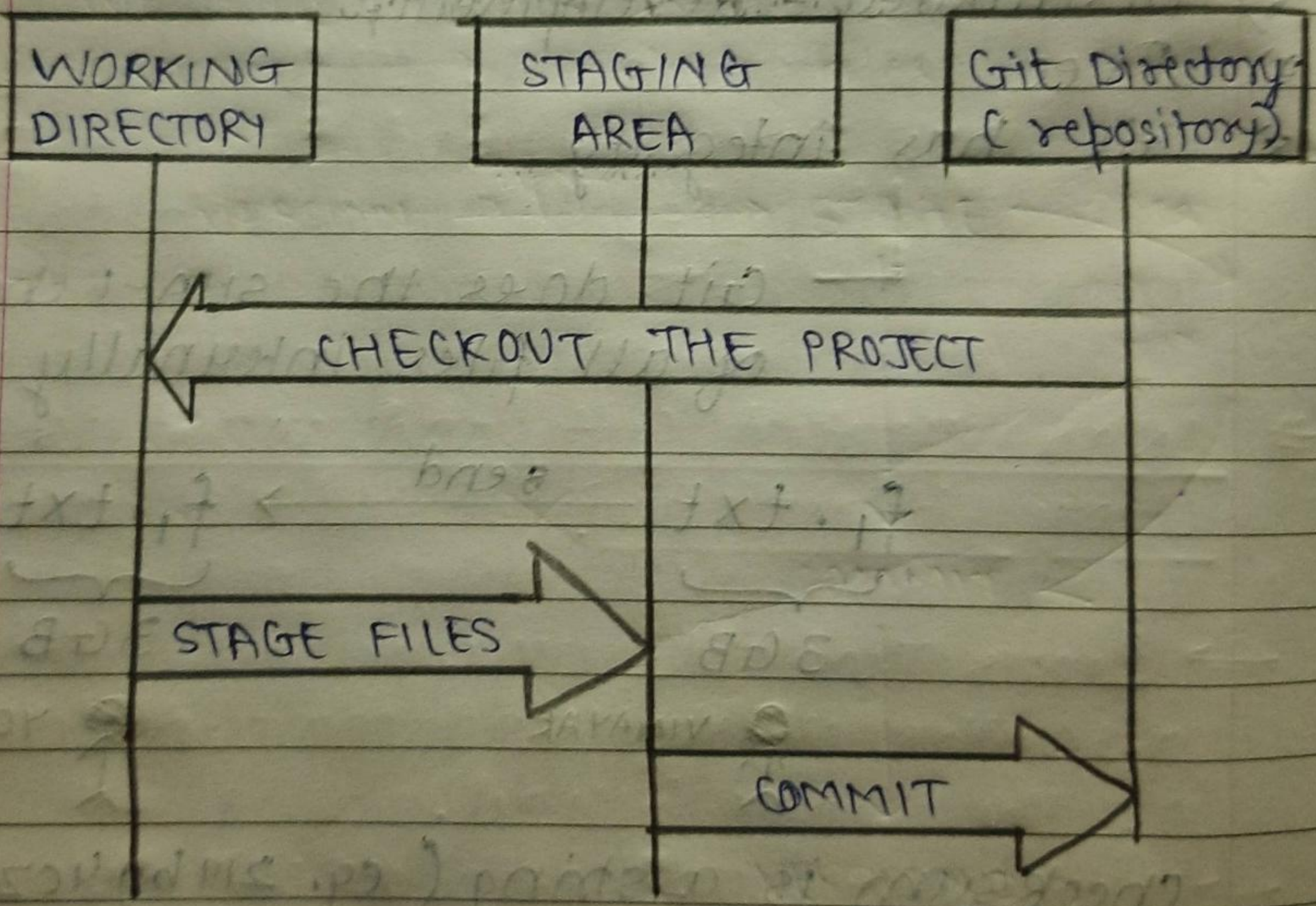
1) Tracks history
2) Creates backup
3) Almost every operation is local
   (exception :- pull, push operations)
4) Scalable
5) supports non-linear development
6) Distributed development.
7) Free
8) Git has integrity
   - Git does the SHA-1 checksum of all files internally



checksum is a string (eg. 219b94cZf6122). Every file has unique file. If anything is changed in files checksum is changed. If I send you f1.txt our checksum should match, if not they it has been changed in b/w. Git takes care that nothing is altered in between. ♥

INSTALL GIT in your laptop from its official website only. If you type git scm you will get official website.

Git : Three stage architecture



twitter :- @vinayakstwt

M T W T F S S
Page No.: 11
Date: YOUVA

DON'T GET AFRAID SEEING THE DIAGRAM!
IT'S EASY LET ME EXPLAIN 😊

WORKING DIRECTORY :- The working directory/ tree consists of files that you are currently working on. In this directory you view & modify files.

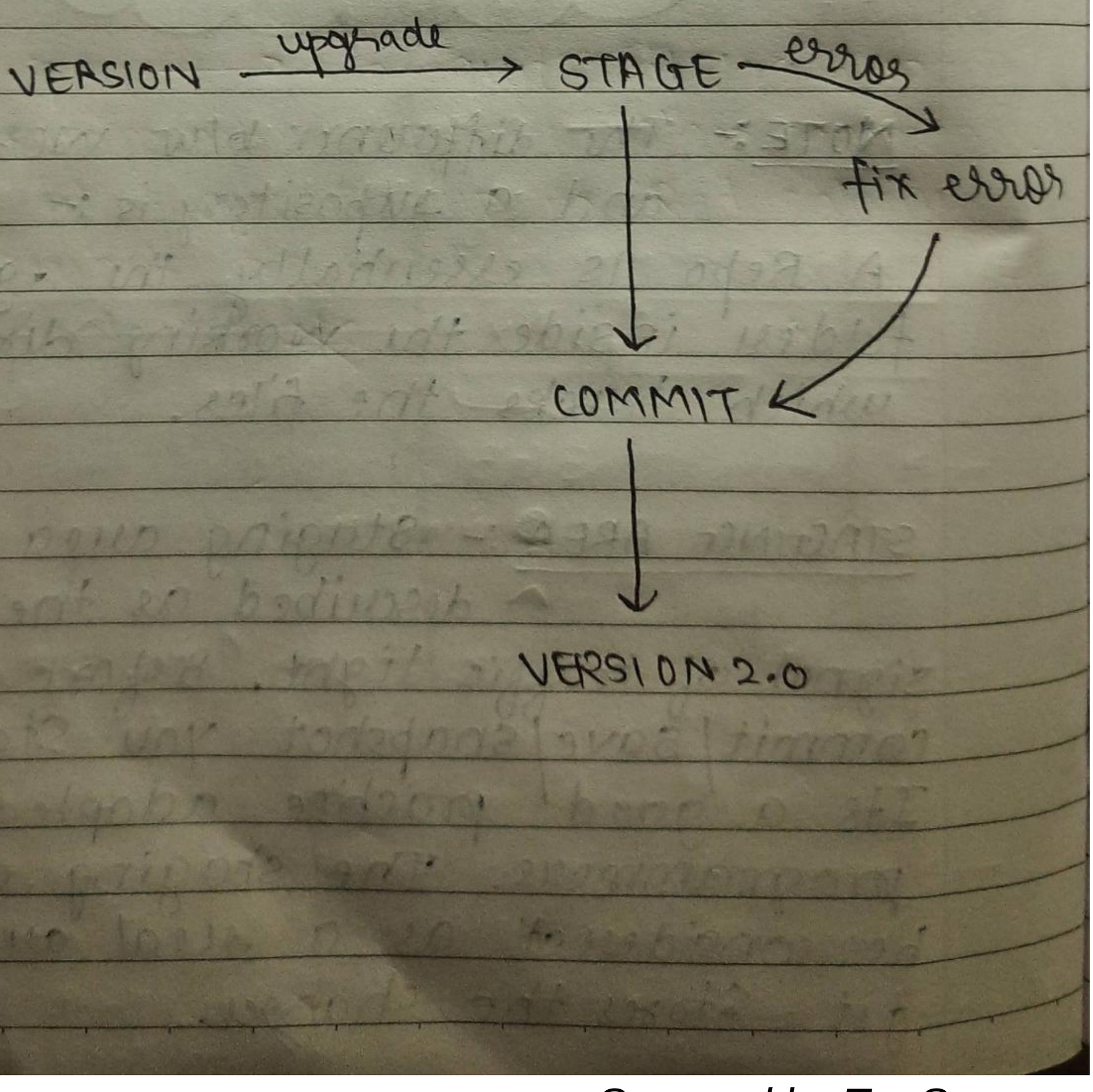
DIRECTORY = FOLDER

NOTE :- The difference b/w working directory and a repository is :-
A Repo is essentially the .git hidden folder inside the working directory which tracks the files.

STAGING AREA :- Staging area can be described as the yellow signal of traffic light. Before you can commit/save/snapshot you stage them. Its a good practice adopted by programmers. The staging area can be considered as a real area where git stores the changes.

GIT DIRECTORY:- Git directory means .git folder which stores all the data (current & previous commits/ saved files) and tracks them and is hidden hence cannot be seen inside working directory / folder.

WE FOLLOW THIS SEQUENCE:-



twitter:- @vinayakstwt

M T W T F S
Page No.: 13
Date: TO JVA

GIT COMMANDS (BASICS)

1) pwd ⟶ present working directory

2) cd ⟶ to change directory

3) git status ⟶ To check whether the working direct-ory is git repo or not

4) git init ⟶ initializ-ed git repo

5) git add --a or git add . ⟶ this command will stage all files, basically will start tracking

6) git commit -m "msg" ⟶ It will give message of what is committed.

7) git log ⟶ To check who has done commit & when. You will also get hash

[Hash → Hashes are what enable git to share data efficiently b/w repositories. If two files are same, their hashes are guaranteed to be same]

8) git add filename.type ⟶ It will track/ stage only that file.

9) rm -rf .git ⟶ To delete .git repo and to lose all tracking

10) git clone <url> <filenameyou wish>

↳ ⟶ To clone repo from github ❤

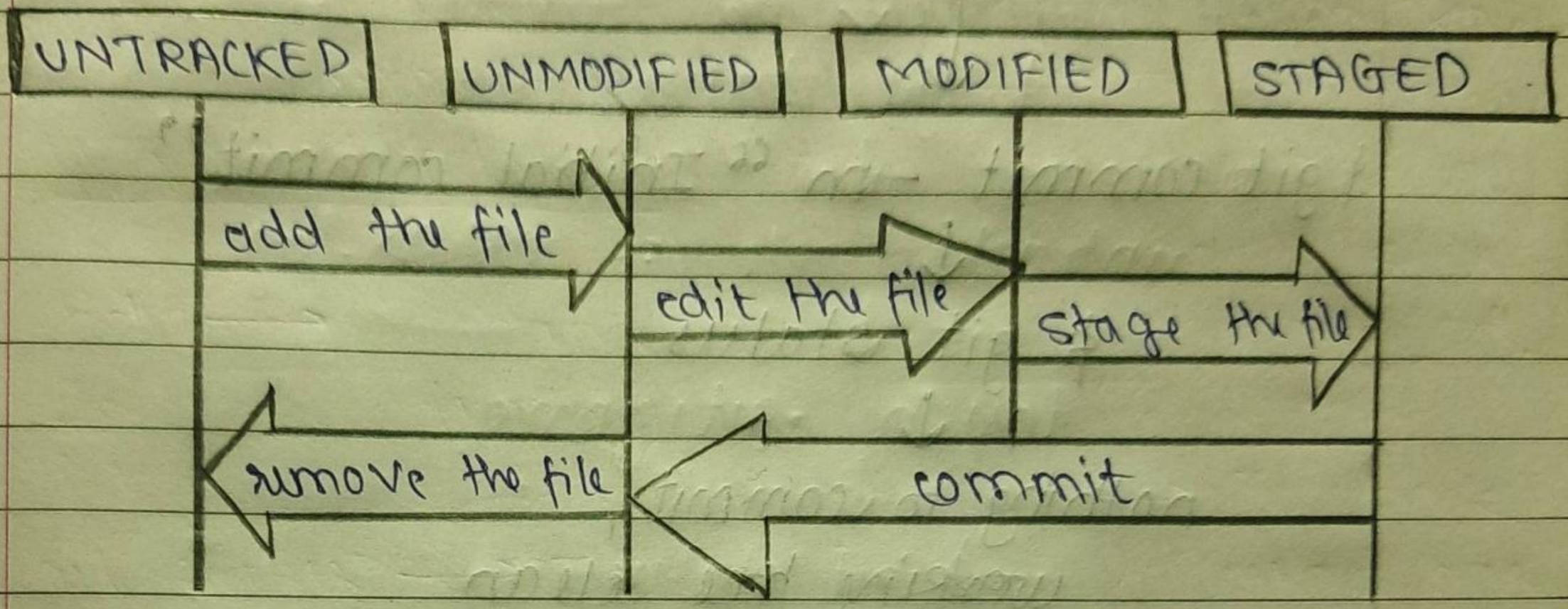
11) ls ⟶ to list files

12) q ⟶ to quit git log & many other commands.

twitter :- @vinayakstwt

Page No.: 15 YOUVA
Date:

FILE STATUS LIFECYCLE



suppose I have project contain files f1, f2 & temp.css

↓

$ git status

↓

It will say no git repo

↓

$ git init

↓

Initialized my project into git repo but is still untracked

↓

$ git add --a

↓

Project is now tracking and is unmodified

↓

suppose I changed f1 to f1'

↓

$ git status

↓

modified f1 (In red colour)

↓

$git add f1'

↓

modified f1' (In green colour)

f,' is now staged also and tracked also

↓

$ git commit -m "Initial commit"

↓

$ git status

↓

nothing to commit,
working tree clean

!

* {Remmember untracked / unstaged files are shown in Red colour while tracked / staged in green colour} *

twitter:- @vinayakstwt

Page No.: 19
Date: YOUVA

GIT IGNORE : HOW TO IGNORE FILES IN GIT

I have a project having files
first.txt, second.txt

↓

$ git status

↓

not a git repo

↓

$ git init

↓

initialized git repo
my project is untracked git repo

↓

$ git status

↓

shows untracked files
first.txt
second.txt
→ (Both in red colour)

↓

$ git add --a

↓

shows tracked files
first.txt
second.txt
→ (Both in green colour)

↓

touch error.log ... [touch command generates file]

↓

$ git status

↓

untracked file : error.log

↓

touch .gitignore

↓

$ git status

↓

untracked file : error.log
.gitignore

↓

Now open the project and open .gitignore file. Type error.log in it and save and closed

↓

$ git status

↓

untracked file : .gitignore ... [Does not show error.log file!!]

↓

$ git add .

↓

$ git status

↓

tracked all files
including .gitignore

↓

If I modify ~~error~~.log
and save it

↓

$ git status

↓

It will not show
me modified error.log
file in Red colour.
Infact it will not even
mention its name.
It will show working
tree clean

If there are many .log wala files
then type '*.log' in .gitignore and
all .log ones will be ignored.

If you wish to ignore directory / folder then write '[illegible]/' in the .gitignore file but remember it ignores folder if its empty.

twitter :- @vinayakstart

M T W T F S S
Page No.: 21
Date: YOUVA

Git Diff: This command shows changes b/w commits/staging area & working directory.

Suppose I have project with files one.txt, two.txt

↓

$ git status

↓

untracked one.txt, two.txt

↓

$ git add .

↓

all files started tracking

↓

Then I open suppose one.txt file and typed 'Hello' and modified it

↓

$ git status

↓

modified: one.txt

↓

$ git diff

↓

It will show me what changes I have made!

In short,

git diff ⟶ Compares working directory with staging area

git diff --staged ⟶ compares staging area and previous commit.

SKIPPING THE STAGING AREA

Suppose I have a project with only one file hello.txt

↓

$ git status

↓

not a git repo

↓

$ git init

↓

$ git status

↓

untracked : hello.txt

↓

$ git add .

↓

tracked : hello.txt

↓

Now I add another file hello1.txt and modify hello.txt

↓

$ git status

↓

modified : hello.txt

untracked : hello1.txt

↓

$ git commit -a -m "Direct commit"

↓

[command to skip staging area]

untracked : hello1.txt

[Remember only tracked files are committed and untracked files are not committed]

↓

$ git add hello1.txt

↓

$ git status

↓

working tree clean

How to rename/move and delete files using git commands

I have a git repository having files one.txt, second.txt all tracked

↓

$ git status

↓

working tree clean

↓

Now I open folder and rename one.txt to third.txt

↓

$ git status

↓

deleted :- one.txt
added :- third.txt

[But I had renamed one.txt not deleted, so git does not understand this]

↓

$ git add --a

↓

renamed: one.txt → third.txt

[Now, git understood that
I had renamed]

↓

git commit -m "rename one.txt
to third.txt"

↓ ↓

$ git status

↓

working tree clean.

DELETION:

Suppose I have a file
one.txt in my repo

↓

$ git rm one.txt

↓

removed one.txt

↓

$ git status

↓

deleted: one.txt [Here git stages the file on its own]

Renaming:

If I have a file named
first.txt

↓

$ git mv first.txt first-renamed.txt

↓

File in the folder is renamed!

↓

$ git status

↓

renamed : first.txt → first_renamed.txt

[Here it stages file by default]

twitter :- @vinayakstwt

Page No.: 27 YOUVA
Date:

UNTRACKING

Suppose I have files F_1, F_2 and want to ignore F_1. So I put it into .gitignore file

↓

$ git status

↓

modified .gitignore

↓

$ git add .

↓

$ git commit -m "changed .gitignore"

↓

Now I add something in F_1

↓

$ git status

↓

modified : F_1 { You may think why it is showing this since I had put it in gitignore. It is because It was tracking it from start. We have to untrack it}

↓

$ git rm --cached F1

↓

$ git status

↓

deleted : F1

{ It is not deleted only untracked but git shows deleted msg?

↓

$ git commit -m "removed F1"

↓

$ git status

↓

working tree clean

↓

If I add again somthing into F1

↓

$ git status

↓

working tree clean

↓

Now If I remove f1 from .gitignore file

↓

$ git status

↓

untracked file : f1
modified : .gitignore

Viewing and changing commits in git using Git commands

git log is the command to view commits in gitbash.

$ git log -p ⟶ along with commit shows what is removed and added

$ git log -p -2 ⟶ 2 commit & shows what is added & removed

{ you can replace 2 with 3, 4, 5, 6, n and will show that many commits }

$ git log --stat ⟶ shows in short how many lines are added and removed

$ git log --pretty=one line ⟶ shows commit in one line

$ git log --pretty=short ⟶ shows commit in short

$ git log --pretty=full ⟶ shows commit in long form

$ git log --since=2 days ⟶ filter according to time
{ Instead of days you can add 'months' & 'years' }

twitter:- @vinayakstust

Page No.: 31 YOUVA
Date:

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

h = hash

author name

{ visit website, search "git scm useful options for git log format" }

$ git log --pretty=format:"%h--%ae"

hash

author email

If you want to amend your previous commit message then,

$ git commit --amend

It will open an editor

i= to edit in editor

type → escape :wq + enter = to exit

Git commands to unstage and unmodify files in gitbash

I have a file first.txt
in my project
↓
$ git status
↓
not a git repo
↓
$git init
↓
$ git status
↓
No commits yet
↓
$ git add
↓
$ git commit -m "Initial commit"
↓
$ git status
↓
working tree clean
↓
Now we modify first.txt

twitter:- @vinayakstwt

↓

$ git status

↓

modified : first.txt

↓

git add first.txt

↓

modified first.txt ... [staged]

↓

git restore --staged first.txt
— (to unstage)

↓

$ git status

↓

modified : first.txt ... (unstaged)

Now suppose I delete my entire content in first.txt

↓

$ git status

↓

modified : first.txt ... (I have deleted but it shows modified)

↓

git checkout -- first.txt

↓

my entire content will be recovered!

But if I modified it, staged it and then commanded
git checkout -- first.txt
it won't work.

twitter:- @vinayakstut

M T W T F S S
Page No: 35
Date:
YOUVA

If you have modified 50 files and want to go to previous commit use

$ git checkout -f

What is Alias ?

Alias means instead of writing big command write it in short form

For eg. $ git status

↓

working tree clean

↓

git st

↓

Git will not understand this

↓

$ git config --global Alias.st 'status'

↓

$ git st

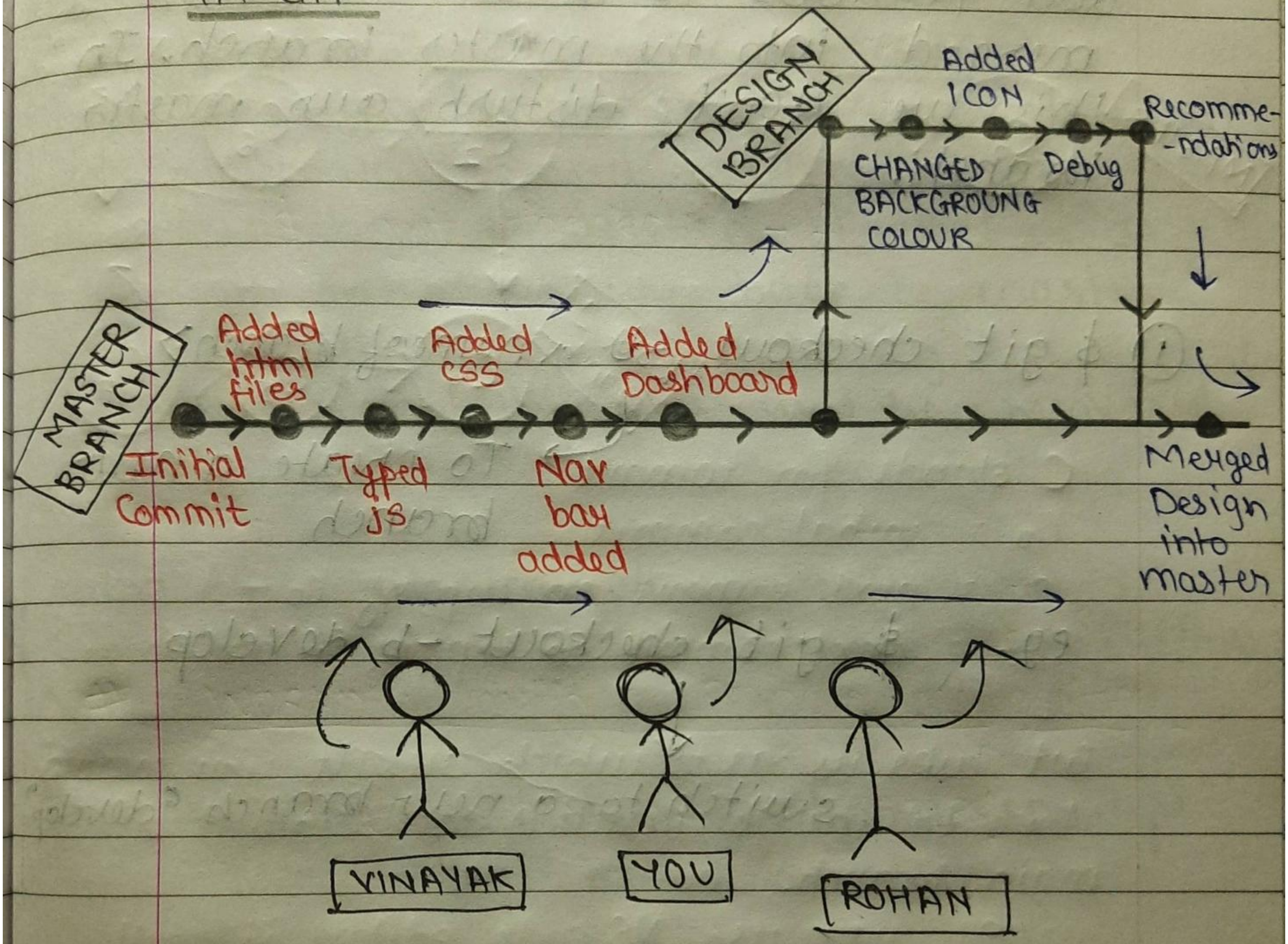
↓

working tree clean

twitter:- @vinayakstwt

M T W T F S S
Page No.: 37
Date:
YOUVA

HOW TO CREATE BRANCHES AND SWITCH BRANCHES IN GIT



Suppose we are working on a e-commerce website. The main website which users are seeing is the master branch. If we try to add features on master branch without creating an different branch and it doesn't work then it can affect income & our brand status.

Therefore, we create an branch and add features to it which is later on merged into the master branch. In this we don't disturb our master branch.

① $ git checkout -b <nameof branch>

→ To create a new branch

eg. $ git checkout -b develop

↓

switch to a new branch 'develop'

② $ git checkout master ⟶ To return back to master branch

twitter:- @vinayakstwt

suppose your master branch contains files F1, F2, F3, F4 and temp.csv and you command "$ git checkout -b develop" and switched to new branch and deleted F1 & F2. If you ~~return~~ return back to master branch using command 'git checkout master' all the files will be recovered.

(3) $ git branch ⟶ To show all branches out there

(4) If you want to merge branch 'Develop' into 'Master' branch

Make sure your in master branch

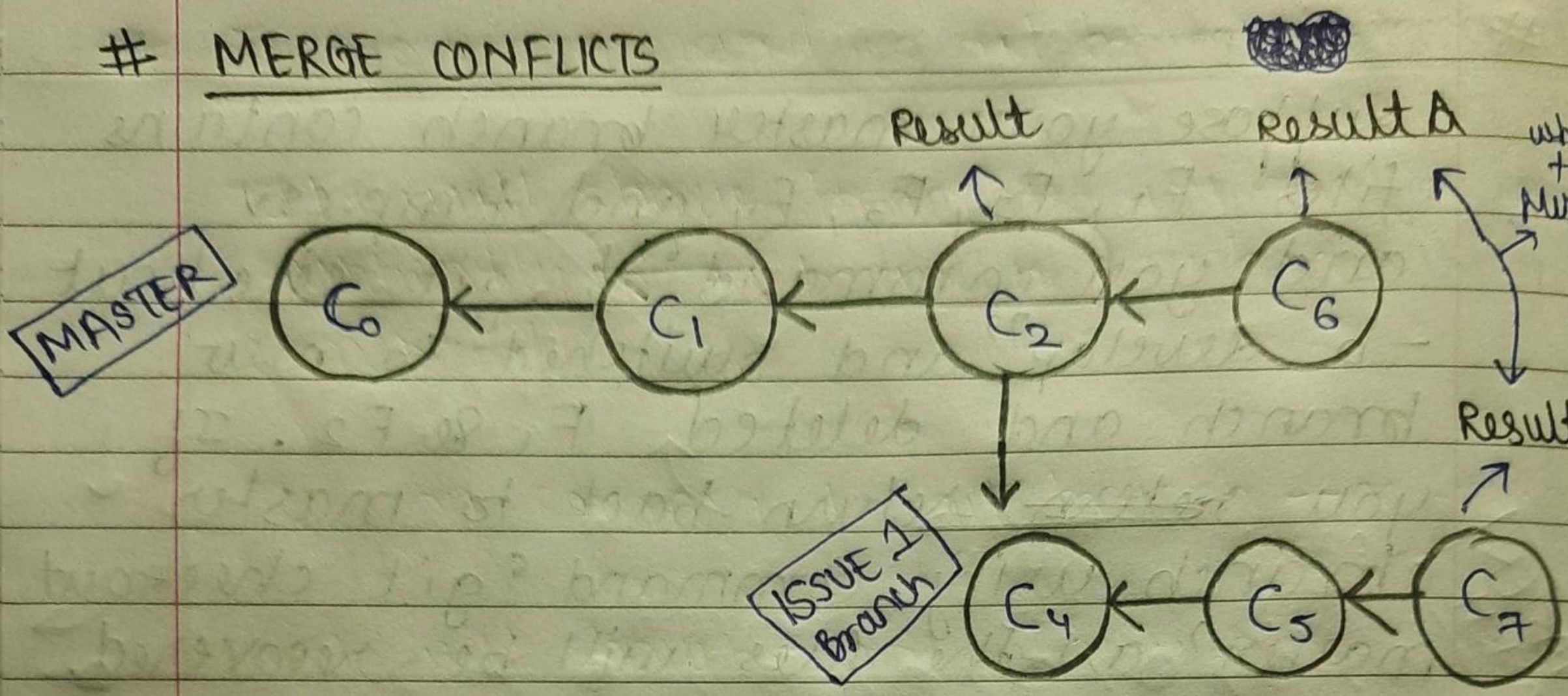
↓

$ git merge Develop

↓

Whoopee merged!

MERGE CONFLICTS



C stands for commit
C_0 = initial commit
C_1 = next commit on top of C_0
C_2 = next commit on top of C_1 and so on

Git asks us as to which result you would love to merge Result A or B into the main branch.

If you use VS code editor, it will mark two choices with conflict resolution markers which looks like :-

"<<<<<<"

Obviously, in real time projects you will have two or more outcomes and you can do merging using git command :-

$ git add filename.type

[Just make sure that while merging you are in master branch and applying this git command there]

HOW TO MANAGE BRANCHES : AN ADVICE FROM A SR. ENGINEER

If you command git branch

↓

* master ← main branch
develop
system
xyz...
} other branches depending on project

In git making branches is not expensive, it does not require storage, it only saves only changes and leaves a pointer on previous commit.

If you command :- git branch -v

* master	commit hash	commit msg
develop	97b1g72zx	"Fixed bug"

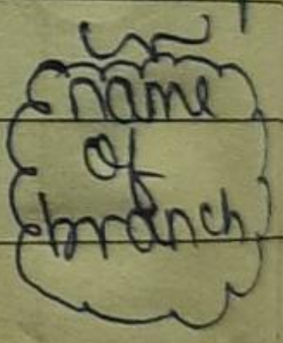
If you command git branch --merged

It will show branches you are currently in + branches you had merged

If you command git branch --no-merged

It will show branchus you have not merged.

If you command git branch -d develop



It will first give error to confirm that you are not deleting by mistakely and will ask you to type capital.

∴ git branch -D develop

Then it will delete branch!

twitter:- @vinayakstut

M T W T F S S
Page No.: 43
Date: YOUVA

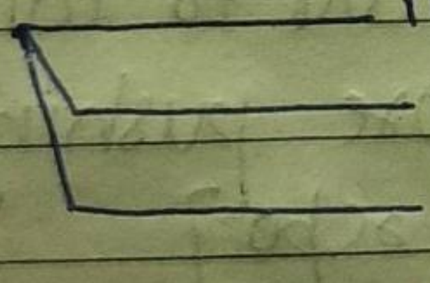
BRANCHING WORKFLOW

BRANCHING WORKFLOW

LONG RUNNING BRANCHES

This branches as name indicates are long. They exists till the project exists.

for example:-



master march will contain main code, in develop your development will be done & same with design, they all are long running & keep upgrading project.

TOPIC BRANCHES

It is a short lived branch that you create and use for a single particular feature or related work. They are typically lightweight branch that you create locally and that have a name that is meaning-full to you.

PUSHING GIT BRANCHES IN REMOTE REPOSITORIES

To do this, go to the branch you want to push.

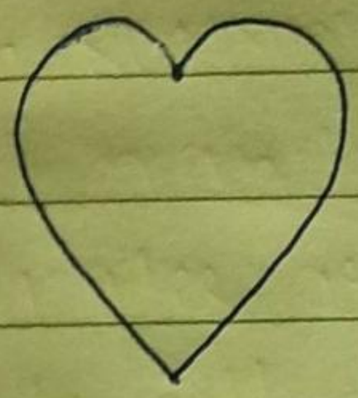
For example :- If I want to push develop branch into remote repo I should be in develop branch (If yours not use git command git checkout develop) and then to push it use command :-

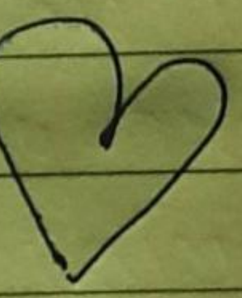
```
git push origin develop
```

[Remember make sure to commit & check git status before pushing your branch into remote repo]

If you want to name develop branch new in your local system to newdevelop in remote repository, use command :-

```
git push origin develop : newdevelop
```



Hope it adds
Value to the
community 

FOLLOW ME ON TWITTER :- @vinayakstwt

FIND MY BLOG ON HASHNODE :- vinayakpanchal.hashnode
.dev

KEEP CONTRIBUTING, KEE GROWING!