

Git & GitHub
Notes by Pragati Agrawal

PAGE NO. 01

Curriculum (Git & GitHub Master Course - By Mohit Uppal Sir)

Part - 01

- Version Control System
- Command-Line Tool
- Git Introduction
- Tracking a project
- Git Additional command
- Branching

Part - 02

- Merging
- Relining
- Intro to GitHub
- Social coding with GitHub
- Rewriting history

Bonus: Creating a Portfolio

LEARNING AND TAKEAWAYS

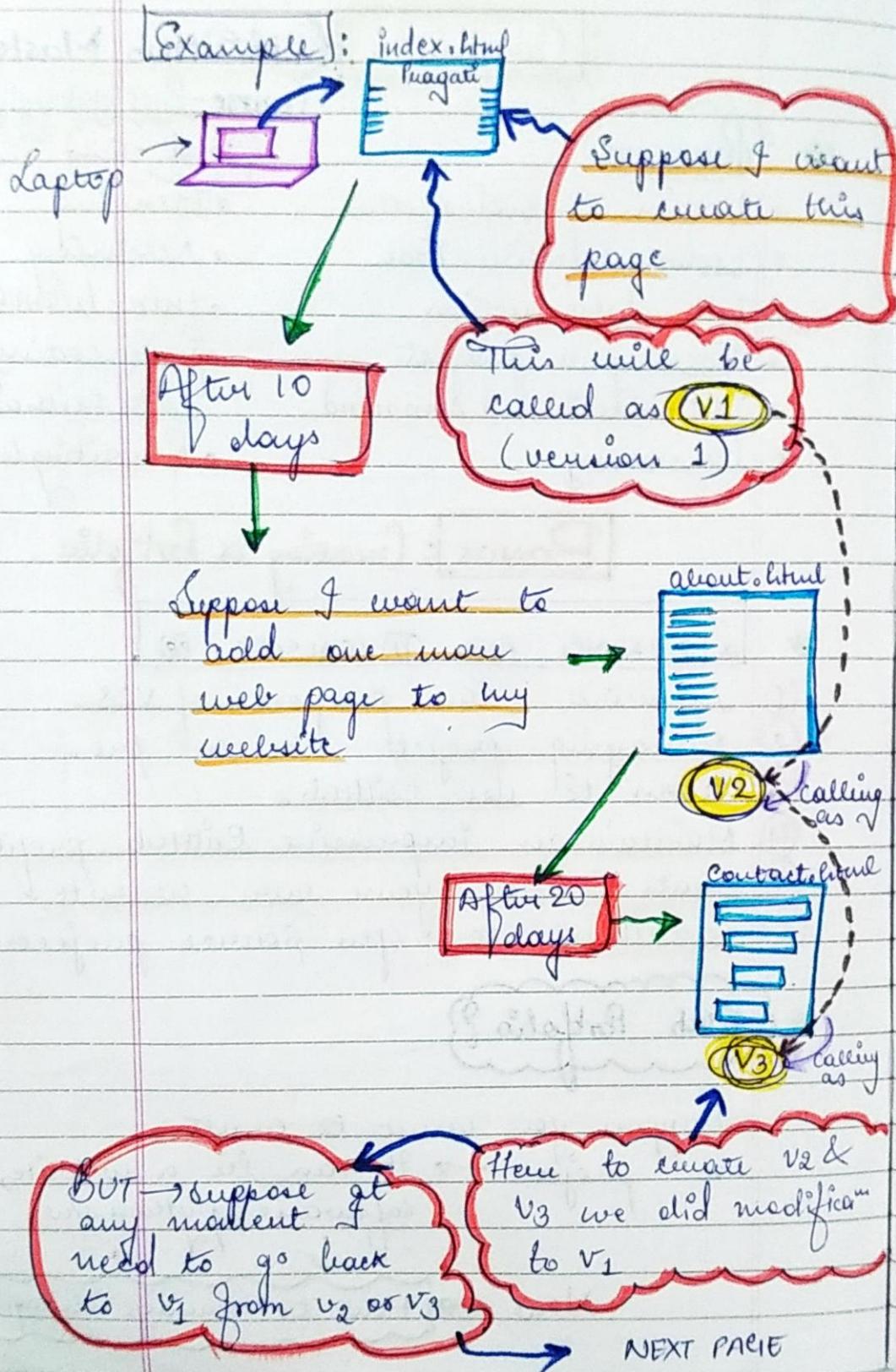
- ① Appreciate the purpose of VCS.
- ② Managing projects like a pro.
- ③ Learn to use GitHub.
- ④ Maintain an impressive GitHub profile.
- ⑤ Create & Host your own website.
- ⑥ Contribute to open source projects.

Web Portfolios?

Suppose you want to create a project → It can be a website, software, python s/w,

Now → How to manage changes?

02



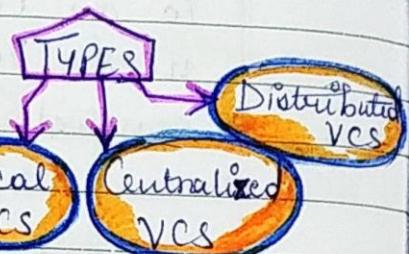
To do this, we need VCS (version control system).

Version Control System

- Defn: VCS is a system that is responsible for managing changes to computer programs, documents, large websites, or other collections of information.
 - Easy recovery of files/folders.
 - Rollback to previous version.
 - Informs us about who, what, when, why changes have been made.
- (git)
- * VCS helps us in getting to any desired version of the project from current version.
 - * VCS helps in tracking the files.
- Can be stored easily using VCS?
- When there is a team of many people, such info becomes critical.

04

History of VCS



- Changes are stored in database, along with the timestamp
- Code is in local system

About Local VCS

Example

Suppose we have a database



& we have committed V1.

This can be saved in db with a timestamp

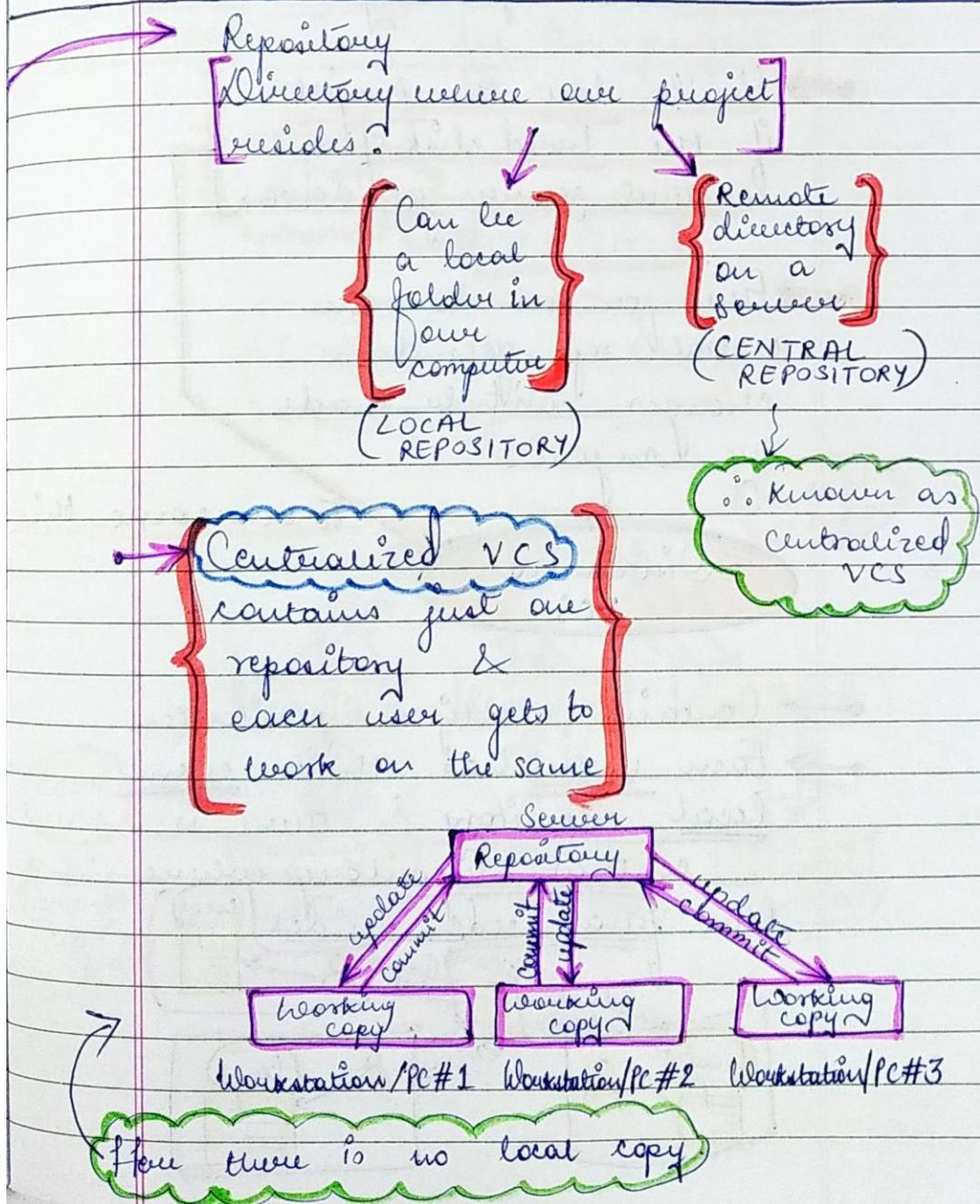
V1	10 th Jan 2021
V2	25 th Jan 2021
V3	5 th Feb 2021

e.g. will contain both date & time for V1

This whole data will be stored at the local system

Disadvantage of Local VCS

Project can be lost, if ~~our~~ our hard-disk is corrupted.



06

Disadvantages of Centralized VCS

→ Everyone gets to work on one repository

→ You'll lose the project if the hard disk of the central server goes down.

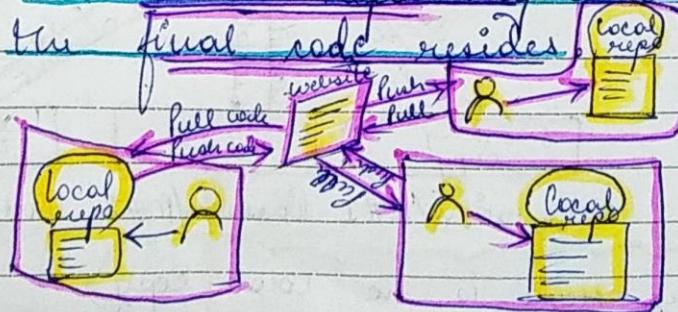
→ Even for an hr the central repo goes down, changes can't be made by anyone.

To overcome this

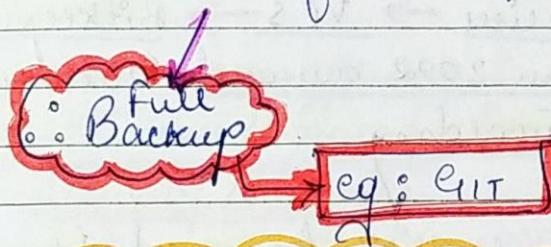
Distributed VCS

→ Contains multiple repositories.

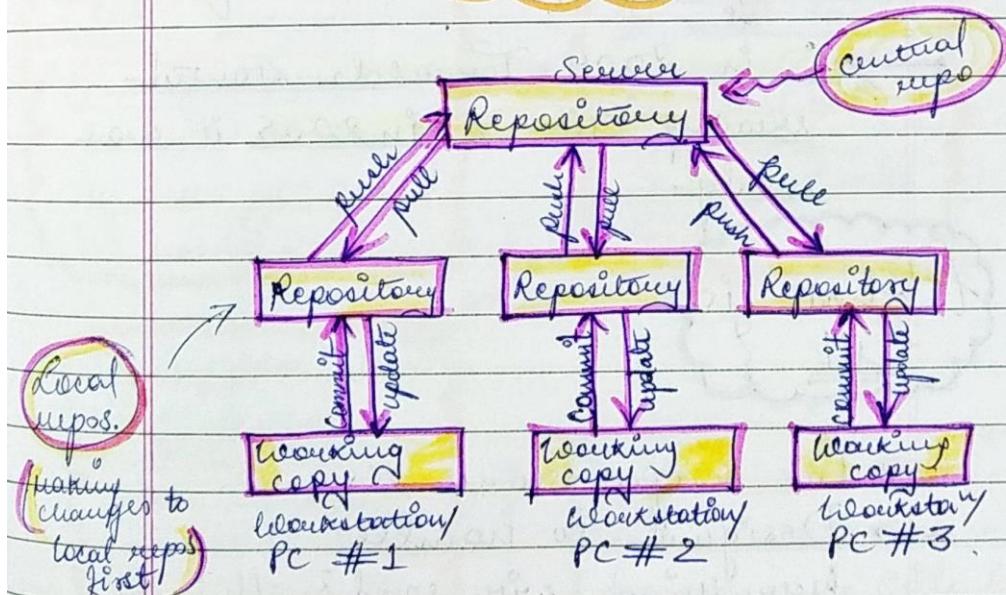
→ Each user has their own local repository & there is a central repository where the final code resides.



→ Here if suppose 100 programmers are working on the project then all the 100 programmers will have their own complete **IMPORTANT** code for the project



Distributed Version Control System



08

How is Git
created?

- Created in 2005 by Linus Torvalds.
(creator of Linux kernel)
- Earlier → VCS → Bitkeeper: VCS
from 2002 onwards was used by
Torvalds.
- But conflicts happened b/w Torvalds
& Bitkeeper founders. They were
not providing Bitkeeper for free.
- ∵ in 2004 Torvalds started
creating Git & in 2005 it was
ready.

What
is
Git?

- Free & open-source VCS.
- Designed to handle
everything with speed & efficiency.
From small to large projects
- Stores snapshots of our projects
(not differences)

Git Features

- Collaboration
- Storing versions
- Analyse the code changes (*Changes are made by whom & when*)
- Distributed
- Almost everything is locally present
- Non-linear / Branching (*Most imp. feature*)
- Security / Integrity
- Speed

NEXT PAGE →

Explaining:

Suppose there is a main branch (our Amazon website) seen by everyone around the world.

Now we have to add some changes (code). We have not been sure whether it will work.

If suppose this new feature works perfectly fine then it is good but in case if it is opposite then we can destroy this newly created branch.

In a nutshell, we can create new branches & side by side work on them.

(10)

Security / Integrity in Git explained

→ Git uses SHA-1. It uses Hash. There is an algo. MD5 for Scanning used in Git.

Speed

→ Since it (Git) is written in C, it is very fast.

Configure Git

This language C is very close to hardware



```

1 # to check git version
2 git --version
3
4
5 # configuration of global variables
6 git config --global user.name "Puqat"
7 git config --global user.email "puqatinov@gmail.com"
  
```

in Git Bash

(See the screenshot → [1])

Introduction & PWD

* COMMAND LINE TOOL/INTERFACE (CLI)

→ Here to do everything
we need commands

* Need of CLI

All the servers are generally
on the Linux Operating machine
& there is no GUI.

Second screenshot

because installing
GUI takes a
lot of space

* The command [pwd]

→ [2] Stands for Print Working Directory

[Gives the present
working directory]

eg: /Users/mohit

→ Directory in
which Terminal
(Crt Bash) is
opened.

- This slash represents
the root directory

- Inside root → there is
a users folder

- Inside users
there is a
folder Mohit

Currently in
users directory

(72)

List Items

to do this we
the command `ls`

Screenshots

3

Some flags can be
used along with this
commands

`ls -l`

will give all
files & folders
in list format

Clear cmd

To clean
the bash
screen when
filled

`ls -a`

will help us
see all the
hidden files
& folders as
well.

`ls -al`

{ all hidden files &
folders in the
form of a list }

Here this will
show a [single dot] (.)
& [double dots] (..)

Folders having
dot before
them
(e.g. .abc)

parent
directory

means
current
directory

these are
hidden folders

Change Directory

(eg) `cd ./Desktop`

{ will change the present working directory to Desktop }

`cd ..`

{ two dots, to get back to the previous directory (means one step backwards) }

`cd`

{ If we only execute cd then it will directly take us to the home }

tilde symbol

`cd ~`

NOTE: Even we can mention the complete path

e.g. `cd users/nikhil/Desktop`

{ will take us to the root }

Create Files & Folders

`mkdir` cmd is used to create a new folder/directory.

(14)

To create multiple folders at the same time eg: `mkdir f1 f2 f3`

To create a folder & inside it create another folder at the same time [eg: `f4 & f5` inside `f4`]

Then do this

`mkdir -p f4/f5`

this flag is to be used

* To create new files

use `touch` command

eg: `touch file.txt`

open `file.txt`

will open the file created (`file.txt`)

Remove files & folders

`rm` cmd. is used to ~~do~~ this remove files

`rmdir` cmd. is used to remove folders

eg: `rm file.txt`

will delete this file and it doesn't matter whether there is any content in this file or not



Deleting the content of file present in other folder while being present in some any folder.

Deg:

Creating file inside folder
eg: `touch f4/abcd.txt`

mohit@Mohits-MacBook-Pro:~/Desktop/CLI >

`rm f2/file.txt`

Deleting the content of file.txt present in f2

current dir.

NOTE :

If we try to delete the folder directly that contains something, it will not get deleted

eg: If f2 is not empty & we do `rmdir f2` [will show error] →

16

→ to overcome
this (delete folder
containing something)

use the flag -R

e.g. `rm -R f2`

means
recursively

Copy & Move

First let us create
one file

(Creating a file)
`touch script.js`

(Creating a file
inside folder)
`touch p4/index.html`

(Creating a folder)
`mkdir f1`

[OUR GOAL]

GOAL 01

We want copy one
file `script.js` into `f1`
folder

and is
your term
`cp`

(17)

`cp script.js`

copy
PAGES : :

DATE :

we give
desired
value
here

we have to
give file
name here

17

[inside folder f1
with the name
script.js]

GOAL 02

Copying the file present in other
folder (f4) to the current
folder

Cmd :

`cp f4/index.html
index.html`

Here we
can do
renaming
of the
copied
file as
well.

GOAL 03

Copying
multiple
folders
having
some
content
to current
dir

folders
with
content

`cp -R f4 new`

to copy
recursively

NOTE : `cp` is for
copy-paste.
To do cut-paste
use `mv`

Renaming
alone to
"new" &
present in
current
directory

NEXT PAGE

(18)

GOAL - 01

Suppose there is a file `index.html` & we want to move it to folder `f1`

cmd:

`mv index.html
f1/index.html`

GOAL - 02

Renaming the file

e.g.

`mv script.js
hello.js`

Cutting the file & pasting it in the same directory with diff name

Cutting it from current dir
Pasting it with new name at the same place

Bit Introduction

NEXT PAGE →

Types of Git commands

Creating Repository : git init

Making changes : git add,
(to repo.) git status,

cmd

to check the status

git commit,

We will also see how to use flags with all these cmd.

Parallel Development:

Non linear/ Branching

{ We will talk about git branch,
git merge }

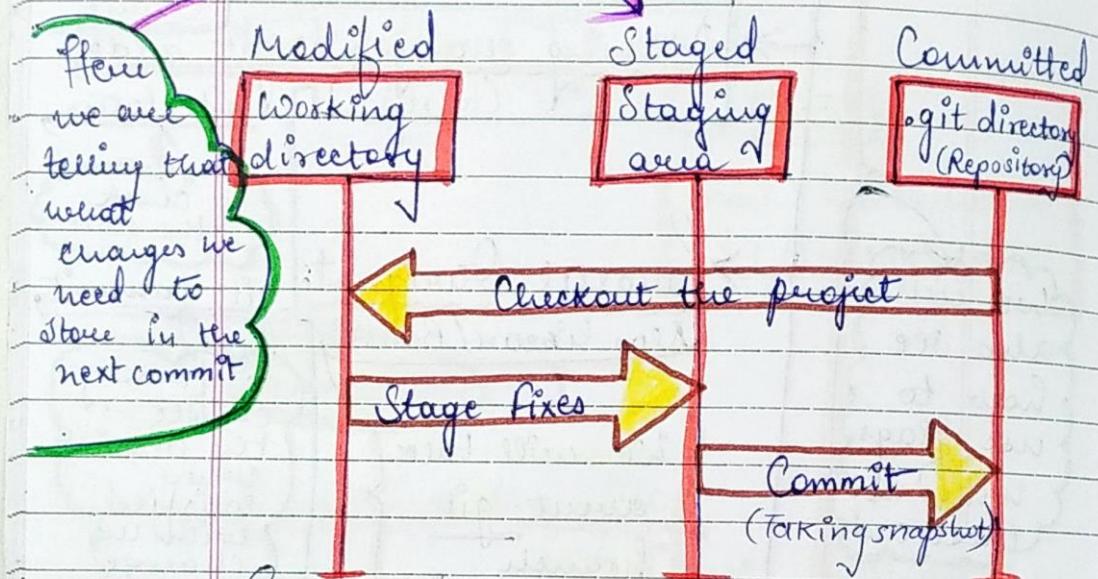
{ to finally create the version when satisfied with the change }

Synchronizing Repository : Sending code from our local repo to the central repo. (GitHub)

{ origin, remote, push,
pull, fetch cmd. }

20

THREE STAGE ARCHITECTURE



- Suppose there is a git project with a hidden .git folder
- I need to reset my own website, or portfolio so adding index.html
- In WORKING DIRECTORY
- Want to save this version
- Now we add about.html (suppose) to this project
- This is not same as prev. version now

So, first we need to stage these changes.

INITIALISING GIT REPOSITORY → Screenshot

4

Creating a Git Repository (ways)

[Initialising your own local git repository]

[Clone a remote repository]

Explanation

Suppose you have created a website and it is successfully working.

Currently, we will be talking about initializing the repo.

Now there is your friend who wants to contribute to the website.

To do this, use the cmd

git init

In this case, he can simply clone your repo instead of initializing his own new repo.

This will create a hidden .git folder.

Here all versions are stored

That's where git stores everything.

(24)

git status

To track files & folders
(Attaching screenshots.)

Practical

We first checked the status of the folder in which git bash was opened & it showed "fatal, no tracking".

STATUS BEFORE INIT**STATUS AFTER INIT**

After that when we initialized the repository and checked the status, it showed "On Branch master", "No commits yet" & also

This means now any changes happening to this folder are under the track of git.

it shows "Untracked files".

means it is telling us that we have to first put them to staging area & then commit.

TRACKING YOUR PROJECT

→ Screenhots



{ Here we will talk
about how Git
tracks the whole project.

Each file in our
working directory can
be in one of 2 states

TRACKED

UNTRACKED

Tracked files
are those
that Git knows
about

An file in
our working
directory that
is not in
our last
snapshot
& is not
in staging
area

- 1 # for one specific file
- 2 git add <filename>
- 3
- 4
- 5 # for all files & folders
- 6 git add .

Cards:
for
adding
files
& folders to
the staging area

In Git Bash

All the
files &
folders
in current
directory

It will now
allow us to
commit the
changes made
(will stage
our file)

24

COMMITTING CHANGES

Screenshots

6

- Moving files from staged area of to the git folder is called a commit.
- Making commits, creates version/snapshots of our project.

Command

git commit -m
"Your commit message"

To understand about why any particular version is created, this msg is also written

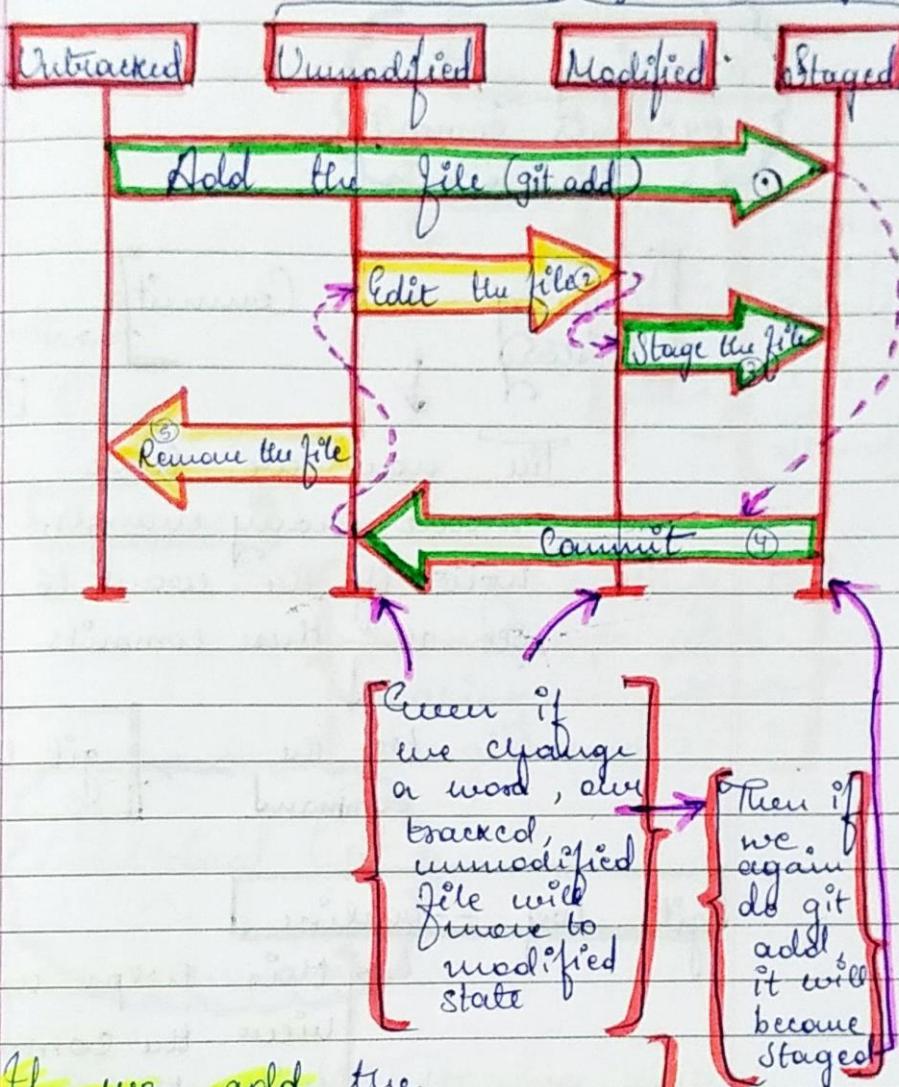
This flag stands for our msg

Screenshot

7

Lifecycle of the Status of a file

Tracked file can be
in any of the 3 stages



NOTE

If we add the file using `git add <filename>` then if we modify it, it will show "modified" when its status will be checked using `git status`. Also will show "changes not staged for commit".

Again after modification, we need to do `git add <filename>`

Logging the previous commits

[Viewing the Commit History]

→ Screenshots
8

Till now we have made many changes, now if we want to see all these commits

Use the command → `git log`

`git log --oneline`

this helps us to view the commits with the msg quickly in single lines

Deleting a Git Repo

→ Screenshots
9

Dangerous cmd,



We should not do this (use the cmd) until & unless we really don't want the project

Command →

`rm -rf .git`

[to remove]

[delete all the files & folders in .git folder]

[if means Forceful (means even if it is not allowed)]

`git push`

{ used for pushing to servers repo after making a commit to git repo }

Additional Git Commands



Skipping the staging area

Area where all our files are ready for getting committed.

Command → `git commit -a -m`

"Commit Message"

(28)

NOTE When for the first time we have initialized the repo, we can't skip the adding, means we have to do git add all



{ Now most of the time we don't make changes to the entire project but only to some particular file }

So if we want to skip the adding & directly want to commit

→ Command → `git commit -a
-m "One line
added in about.html"`

Screenshots → [10]

Git Difference
Check

→ Screenshots → [11]

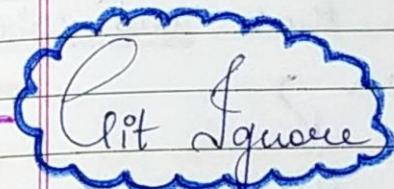
If you want to know exactly what you have changed, not just which files were changed - use `git diff`

will tell us the changes not staged

(added)

command.

Simple `git diff` is used to check the differences in case the changes are already staged, it will not show anything, so to view this use the flag `--staged`.



dimensions
[12]

Often times we don't want git to keep track of some specific autogenerated files like logs, or build files. .DS_Store etc...

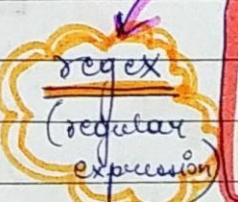
(In mac)

`git diff --staged`

This will show even the staged changes.

To avoid them

You can list files or a pattern for files in .gitignore file.



(30)

Steps to make git ignore the files

- ① first create .gitignore → touch .gitignore
- ② Open .gitignore in VS code
- ③ Write names of all files & folders in this file you don't want to be tracked & save it.

[Suppose we want to ignore all the files that end with .log, then]

Use Regex (Regular Expⁿ)

→ Write *.log

(In .gitignore) [Covers all the names (any name)]

Ignoring a folder

→ Put the name of the folder in .gitignore file