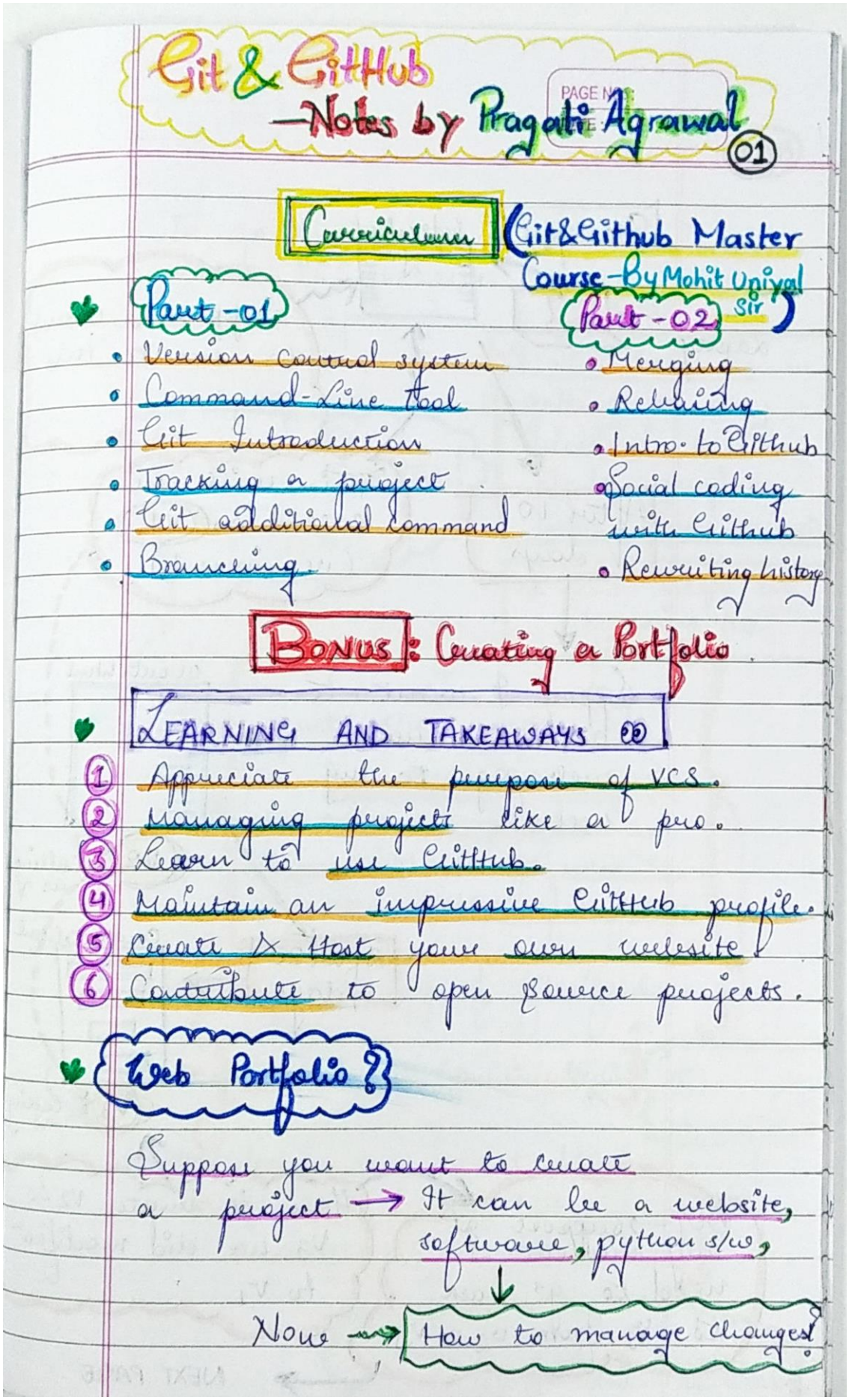
PART-01 Git&GitHub Notes



02

Example: index.html
Pragati

Laptop

Suppose I want to create this page

This will be called as V1 (version 1)

After 10 days

Suppose I want to add one more web page to my website

about.html

V2 calling as

After 20 days

contact.html

V3 calling as

Here to create V2 & V3 we did modification to V1

BUT → suppose at any moment I need to go back to V1 from V2 or V3

NEXT PAGE

PAGE NO.:
DATE: / /

03

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

(git)

✡ VCS helps us in getting to any desired version of the project from current version.

✡ VCS helps in tracking the files.

Version Control System

Def": VCS is a mgmt s/w responsible for managing changes to computer programs, documents, large websites, or other collec" of informa"

- Easy recovery of files/folders.
- Rollback to previous version.
- Informs us about Who, What, When, Why changes have been made.

Can see clone easily using VCS.

When time is a team of many people, such inf becomes critical.

PAGE NO. :
DATE : / /

04

History of VCS

TYPES

- Local VCS
- Centralized VCS
- Distributed 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 created (V1)

This can be saved in db with a timestamp

eg Will contain both date & time for V1

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

This whole data will be stored at the local system

PAGE NO. :
DATE : / /

05

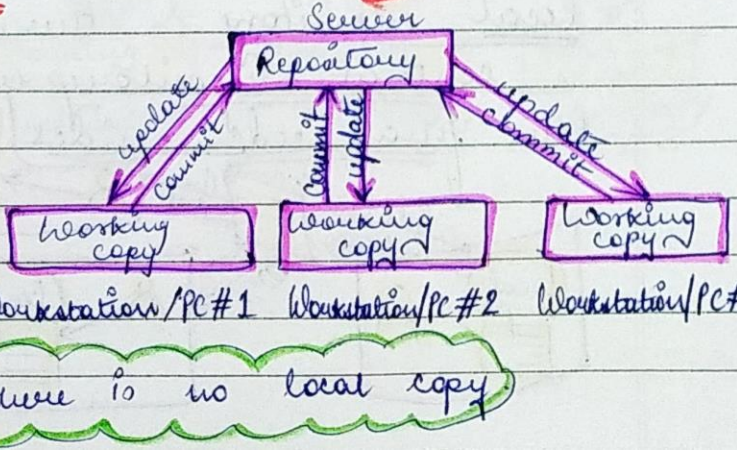
Disadvantage of Local VCS

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

Repository
Directory where our project resides?

- Can be a local folder in our computer (LOCAL REPOSITORY)
- Remote directory on a server (CENTRAL REPOSITORY) → ∴ Known as Centralized VCS

Centralized VCS
contains just one repository & each user gets to work on the same



If we there is no local copy

06

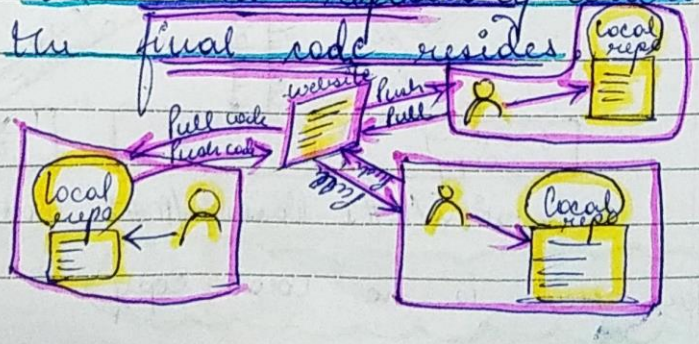
Disadvantages of Centralized VCS

- 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.



PAGE NO. :
DATE : / /

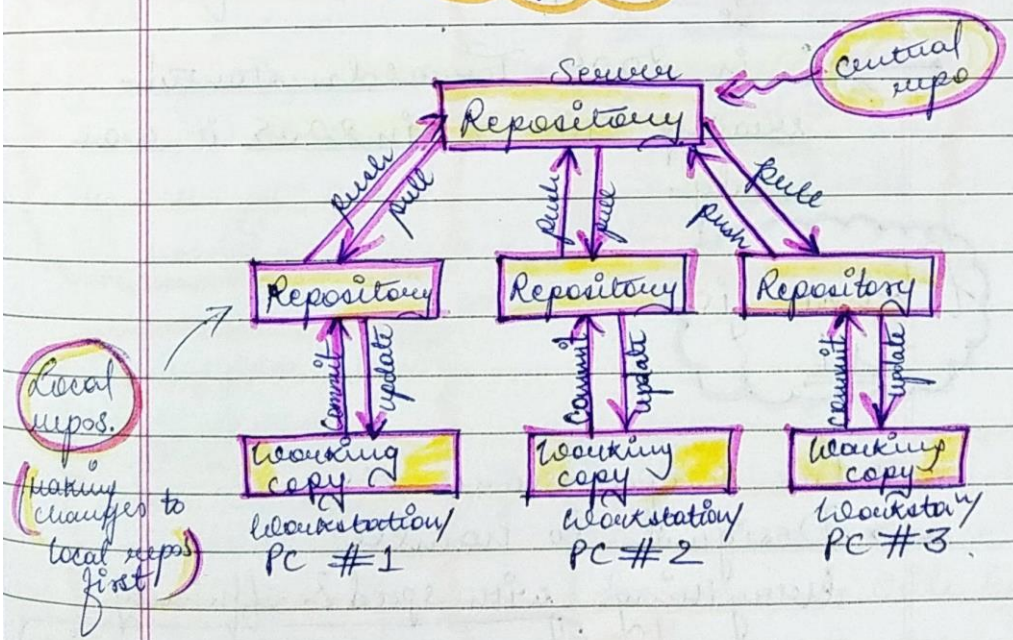
07

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

IMPORTANT

∴ Full Backup → eg: GIT

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 smaller 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 →

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

Now we have to add some changes (code) we are not 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.

PAGE NO. :
DATE : / /

Security / Integrity in Git explained

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

Speed

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

This language C is very close to hardware

Configure Git

```
1# to check git version
2 git --version
3
4
5#configuration of global variables
6 git config --global user.name "Pragati"
7 git config --global user.email "pragatinov@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 & here there is no GUI.

Because installing GUI takes a lot of space

The command pwd → 2 (Second screenshot)

Stands for Print Working Directory

Gives the present working directory

Directory in which Terminal (Git Bash) is opened.

eg: / (Users/Mohit)

- This Slash represents the root directory
- Inside root → there is a Users folder
- Inside users there is a folder Mohit

Currently I am in this directory

PAGE NO.:
DATE: / /

List Items

to do this use the command `ls`

Screenshots → 3

Some flags can be used along with this commands

- `ls -l` → will give all files & folders in list format
- `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.

clear cmd → To clear the bash screen when filled

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

- folders having dot before them (eg .abc) → these are hidden folders
- parent directory
- means current directory

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 write cd then it will directly take us to the home.

`cd ~` — tilde symbol

`cd /` — will take us to the root

NOTE: Even we can mention the complete path
eg: cd Users/mohit/Desktop

Create Files & Folders

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

PAGE NO.:
DATE: / /

(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 ~~this~~ remove files

→ `rmdir` cmd is to remove folders

About rm cmd:

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.
eg:
→ Creating file inside folder
eg: touch f4/abc.txt

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

rm f2/file.txt

Deleting the ~~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 →

PAGE NO.:
DATE: / /

To overcome this (delete folder containing something)

use the flag -R

eg: rm -R f2

mean recursively

Copy & Move

First let us create one file

(creating a file) mohit@Mohits-MacBook-Pro ~/Desktop/CLI> touch script.js

(creating a file inside folder) mohit@Mohits-MacBook-Pro ~/Desktop/CLI> touch f4/index.html

(creating a folder) mohit@Mohits-MacBook-Pro ~/Desktop/CLI> mkdir f1

OUR GOAL

GOAL 01 → We want copy the file script.js into f1 folder

→ cmd is cp for this

copy

PAGE NO.:

DATE: / /

we can give desired name here

we name to this file

```
cp script.js f1/script.js
```

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 folders having some content to current dir

folder f4 with content

cp: `cp -R f4 new`

to copy

recursively

renaming done to "new" & present in current directory

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

NEXT PAGE.

PAGE NO. :
DATE : / /

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

eg.

```
mv script.js
hello.js
```

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

Pasting it with new name at the same place

Cutting it from current dir

Git Introduction

NEXT PART →

Types of Git commands

- **Creating Repository** : git init
- **Making changes** (to repo.) : git add, git status,
  - cmd
  - to check the status
  - git commit,
  - to finally create the version when satisfied with the changes
- **Parallel Development**: Non linear / Branching
  - We will talk about git branch, git merge
- **Syncing Repository** : Sending code from our local repo to the central repo. (GitHub)
  - origin, remote, push, pull, fetch cmds.

We will also see how to use flags with all these cmds

NEXT PAGE →

PAGE NO.:
DATE: / /

(20)

THREE STAGE ARCHITECTURE

Here we are telling that what changes we need to store in the next commit

Modified
Working directory

Staged
Staging area

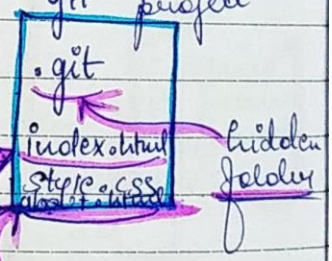
Committed
.git directory (Repository)

Checkout the project

Stage fixes

Commit
(Taking snapshot)

- Suppose there is a git project with a hidden .git folder



- I need to create my own website, a portfolio so adding index.html

In WORKING DIRECTORY

- Now we add about.html (suppose) to this project

Want to save this version

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)

- Initializing your own local git repository
- Clone a remote repository

Explanation

- Suppose you have created a website and it is successfully working.
- Now this is your friend who wants to contribute to the website
- In this case, he can simple clone your repo. instead of initializing his own new repo.

Currently, we will be talking about initializing the repo.

To do this, use the cmd

`git init`

This will create a hidden .git folder.

That's where git stores everything.

Here all versions are stored

PAGE NO. :
DATE : / /

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 →

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

STATUS AFTER INIT →

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

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

PAGE NO.:
DATE: / /

23

TRACKING YOUR PROJECT

→ Screenshots ↓ 5.

Here we will talk about how Git tracks the whole project

Each file in our working directory can be in one of 2 status → 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 .
```

Cmds for adding files & folders to the staging area

In Git Bash

Are the files & folders in current directory

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

COMMITING CHANGES

Screenshots 6

→ Moving files from staged area to the git folder is called a commit.

→ Making commits, creates version/ snapshots of our project.

Command

```
git commit -m "Your commit message"
```

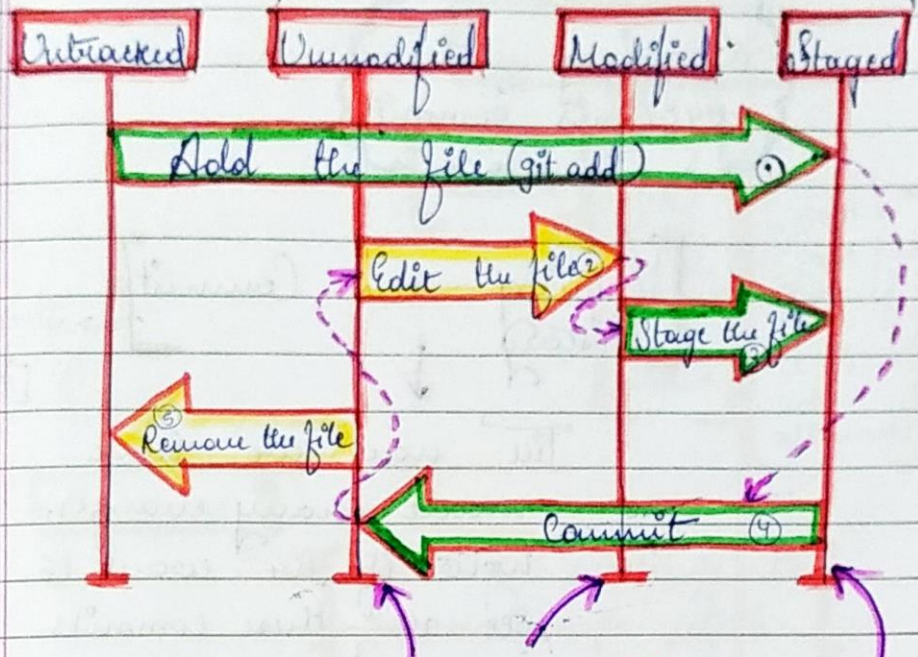
To understand about why any particular version is created, this msg is also written

This flag stands for our msg

Lifecycle of the Status of a file

Screenshots 7

Tracked file can be in any of these 3 stages



Even if we change a word, our tracked, unmodified file will move to modified state

Then if we again do git add, it will become staged

NOTE

If we add the file using git add <filename> & then modify it, it will show "modified" when its status will be checked using git status

Also will show → "changes not staged for commit"

i.e again after modification we need to do git add <filename>

Logging the previous commits

Viewing the Commit History

→ Screenshots 8

File 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 msgs quickly in single lines

Deleting a Git Repo

→ Screenshots 9

Dangerous cmd,

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

Command → rm -rf .git

- rm → to remove
- -r → delete all the files & folders in .git folder
- -f → means forceful (means even if it is not allowed)

git push → used for pushing to server 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"

PAGE NO. :
DATE : / /

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

↓

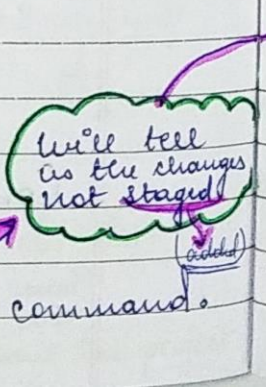
Now most of the time we don't make changes to the entire project but only to some particular file

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` command.



PAGE NO.:
DATE: / /

29

About flag --staged

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

git diff --staged

This will show even the staged changes

Git Ignore

Documentation

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)

To avoid them

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

regex (regular expression)

PAGE NO. :
DATE : / /

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 (.gitignore) 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 Exp^n)

Write *.log

- In .gitignore
- Covers all the names (any name)

Ignoring a folder

Put the name of the folder in .gitignore file