PAGE NO.:
DATE: / /

- Why git ? free

→ - version control system. (VCS)

- proj.zip → proj-final-.zip → ..... → .....
  2GB 2GB

- easy file recovery

- who introduced issue and when

(Rollback to previously working state)

(HDD ✗ fail)

* Local VCS → System failure and can rollback. ✓

* Centralized VCS →

✗ server failure ∴ no rollback

* Distibute VCS = each one has complete backup.

- Github = (hosting service)

- Git =

① Capture snapshot to track proj., not differences — Creates .git folder —

- fetch latest version (pull).

- almost every operation is local.

- Git has integrity. (get checksum of file). at sender / receiver same

- Git generally only adds data.
- Git made by Linux towards.

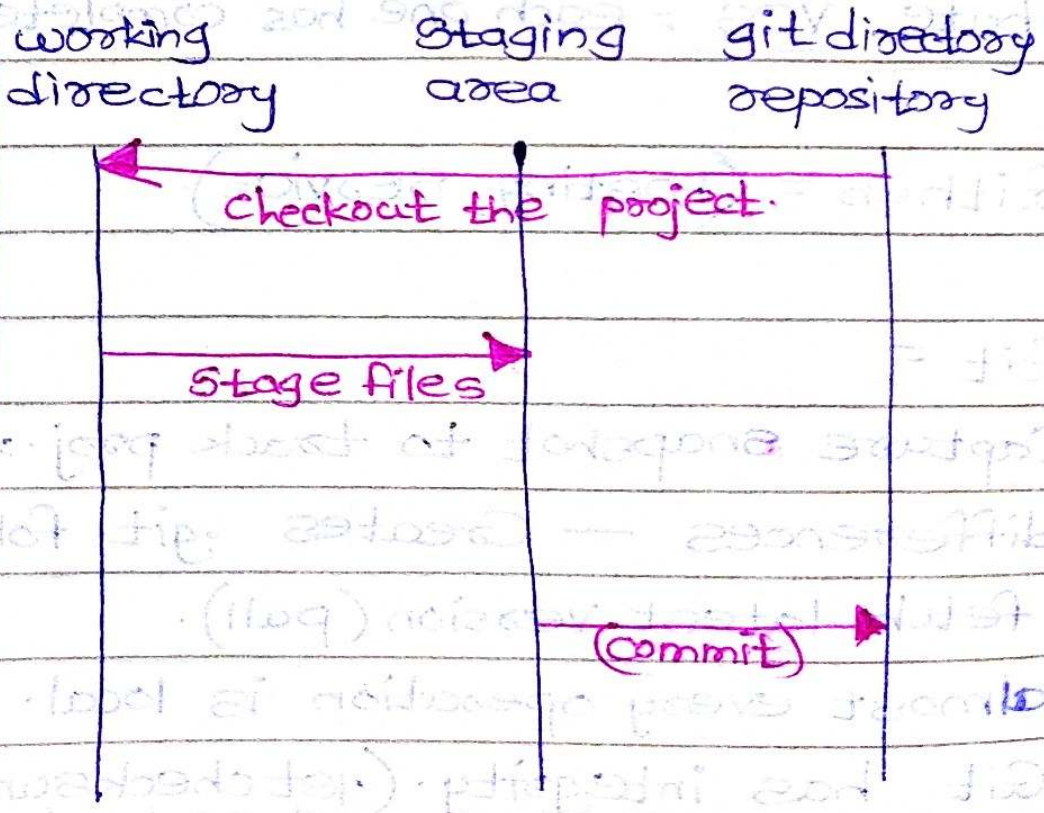
PAGE NO.:
DATE: / /

■ Downloading Git = ① Git CMD => git command
② Git Bash = (like linux)
③ Git GUI =

① User setup in Git =
- Create "Git Repo" folder in D: drive.
- right click inside folder to open Git CMD.

> git config --global user.name "Aishwarya"
> git config --global user.email "-.com"
> git config --list

* 3 Stage architecture of Git =



index.html ✓ stage it

engine.js error

index.css ✓ stage it

PAGE NO.: Commit

DATE: / /

Staging = goto further commit.

.git = compressed files.

*Track Git Project =

- Create .txt, .excel, .db files in "Git Repo" folder
- Goto Git Bash,

> git status = not a git repository.

> git init = initialize empty git repo

> git status = untracked files present

> git add --a = add all files in staging area

> git status = changes to be commited

> git commit -m "Initial commit" =

Commit done, 3 files changed

> git status = nothing to commit, working tree clean.

(snapshot taken successfully)

> git log = to list down all commits.

(hash created)

(Now, change the content of txt : file)

> git status = it will detect modified file

Modify data excel file. (now 2 files modified)

> git add first.txt =

Stage first.txt for commit.
Only.

> git commit -m = 1 file changed,
2 insertions,
1 deletion(-)

* Cloning Remote Git Repository from GitHub =

> rm -rf .git = deletes remove .git (total repo) folder

> git clone github-tensorflow-link tensorflow-repo
name of folder.

> pwd >ls >cd

> git add --a

> git commit -m "Changed readme"

* this is how, now this tensorflow version history and folder is yours.

* we can see (git log) who made changes & when.

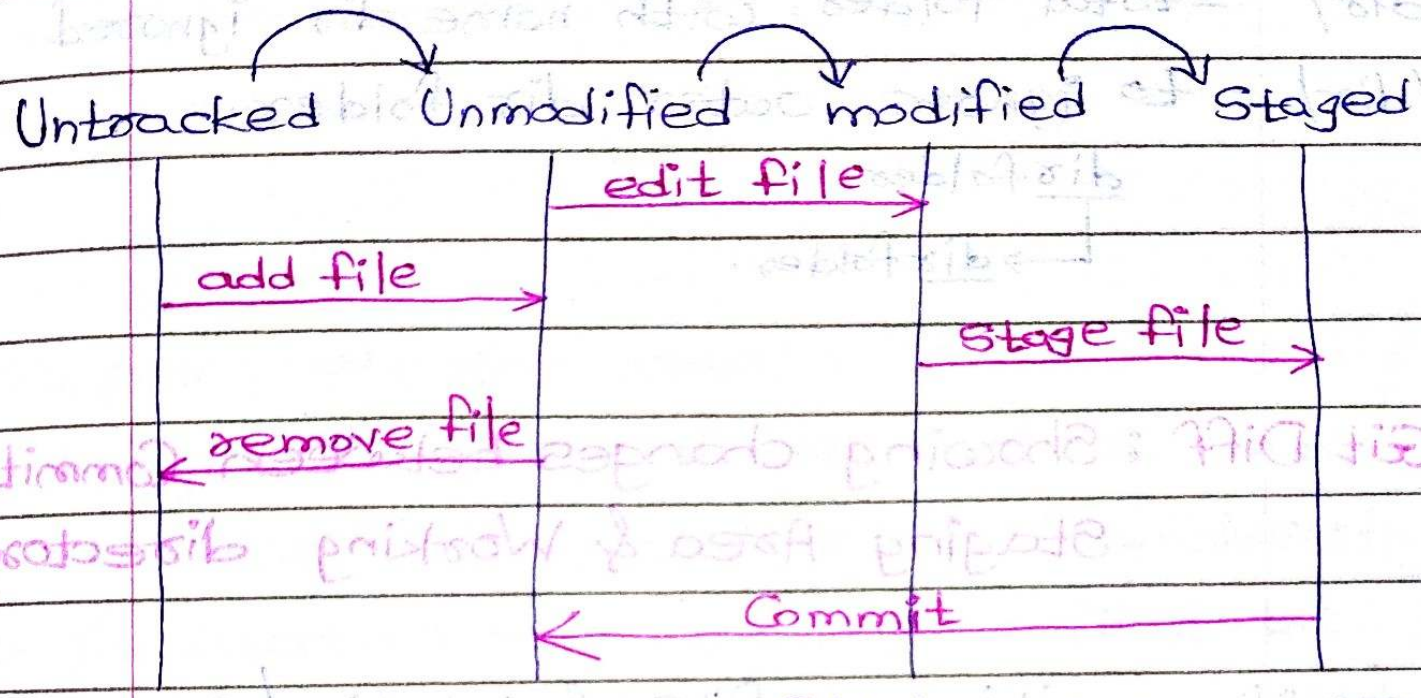
* File Status Lifecycle =

~~tensorflow link~~

PAGE NO.:

DATE: / /

Repository → file1 f2 f3 f4 f5 f6 .git



git add --a → Staging

git commit -m "Changed" → edit → Unmodified

* .gitignore = Ignoring files in Git =

> touch error.log = creates new file

> touch .gitignore = files added in .gitignore are ignored. Just add error.log in ↑ file.

> git add --a

> git add .

> git commit -m "Added gitignore"

* If we add
```
*.log
dir/
static/dir
```
then it ignore all files with .log extension

.gitignore file

* dir/ = total folder with name "dir" ignored.
* /dir/ = to ignore outer dir folder.

dir folder
└→ dir folder.

* Git Diff : Showing changes between Commits / Staging Area & Working directory.

- modify = gitignore file & 'static/f1.txt.
> git add . ⟶ in staging area both file
- modify f1.txt
> git status ⟶ modify = f1.txt
Staging = gitignore
f1.txt
> git diff ⟶ Compare working area with staging area (f1.txt present in both areas)
It will return added & removed lines of files
> git add . ⟶
> git diff ⟶ staging & working area is same.

It will return nothing.

> git diff --staged = PAGE NO.: DATE:

* It compares previous commit with current staging area.

* It will show added/removed content in modified_files. (+ , -).

> git status

>

* Skipping staging area =

- change first.txt files content and
- add new file second.txt in Git-Folder

> git status → modified : first.txt
untracked : second.txt

> git commit -a -m "Direct commit" =
to skip staging (commit all tracked files)

> git status → untracked : second.txt
not included in commit.

> git add second.txt

> git status

> git commit -a -m "added second"

> git log. → to exit type "q"

* Moving and Renaming files in Git =

- If we rename file1.txt to third.txt.

> git status = deleted : file1.txt
Untracked : third.txt

> git add . =

> git status = renamed : file1.txt -> third.txt

> git commit -m "rename file1 to third" =

delete and stage ✓

> git rm third.txt = remove third.txt in folder

> git status = deleted : third.txt

> git commit -m = "removed useless file"

> git mv first.txt first_renamed.txt =
file is renamed and added in stage area.

> git status = renamed first → first_renam

> git commit -m "renamed"

added in .gitignore.

* If we modify "db" file and add & commit.
* Now git status → shows modified: db
* even after commit, it is showing modified.

* we have to tell gitignore that I don't want to track db.accdb file.

PAGE NO.

DATE :

∴ to untrack file →

> git rm --cached db.accdb

→ stop tracking this file.

> git status = deleted: accdb

* Now, if we modify db.accdb, does not show.

* If u wanna track again then remove file name from .gitignore file.

* Git Log: Viewing & Changing Commits in Git =

> git status → nothing to commit

> rm -rf .git → repository totally removed

> git status → not a git repository

> git clone https://github.com/pandas-dev.git myPanda

> git status → not a git repository

* Go to myPanda folder and open GitBash =

> git log → all commits displayed

press "q" to exit

* If we want to know, what changes made in file = (diff ✓)

> git log -p ⇒ all changes shown

> git log -p -3 ⟶ latest 3 changes shown

> git log --stat ⟶ short summary of changes

> git log --pretty=oneline

⟶ all commits information in one line.

> git log --pretty=short

> git log --pretty=full ✓ author & commiter.

> git log --since=2.days

⟶ It will show changes made in last 2 days

> git log --since=2.weeks

> git log --since=2.months

> git log --since=2.years

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

* Open website git-scm useful options for git log format

- %an = author name
- %ae = author email.

? git commit --amend command =

- to modify most recent commit.

- It lets you combine staged changes with previous commit instead of creating entirely new commit

PAGE NO.:
DATE: / /

- Also used to edit previous commit message without changing its ~~timestamp~~ snapshot.

> git commit --amend =
→ editor will open
- "i" to open and insert
- Escape
- ":wq" to exit.
> git log -p -1 →

★ Unstaging & Unmodifying Files in Git =
> git status → no get repo.
> git init → empty repo initialized
> git status → Untracked files.
> git add. → staging files
> git commit -m "Initial Commit" →
> git status → git created repo

- modify 1 file. Stage it.
- git add first_renamed.txt

Now, it is staged.

∴ to Unstage = match to prev. commit =

> git restore --staged first_renamed.txt →

∴ to Unmodify = to match prev. commit

> git checkout -- first_renamed.txt

> git status

> git add .gitignore

> git checkout -- .gitignore

> git status → nothing happened

Want to merge working directory to last commit =

→ Commit first (Carefully)

> git commit -m "this"

* modify - .gitignore
  - first_renamed.txt

> git checkout -f = lose all changes and

> git status = clean working directory

we will get file as in last commit

(All modified changed will be removed).

Git: Working with Remote Repositories

- Go to GitHub site & create new repository.
- public ✓

① Pull = get git repo from github to our system
② Push = push int remote repo Github of yours.

> git remote add origin https://github//Aishwary-gsgs/Demo-repo.git ⟶ Add URL which has name as "origin"

> git remote ⟶ origin
> git remote -v ⟶ (pull) https://--link
(push) https://--link
> git push -u origin main ⟶ No permission /authority to do changes

- Goto Accnt settings ⟶ SSH & GPG Keys

- Generate SSH key = search on google
- After creating key with command
> ssh-keygen -t rsa -b 4096 -C "aish@gmail.com"
- Search = Adding new SSH key to Github accnt.

> eval $(ssh-agent -s) → Agent pid 1281
> ssh-add ~/.ssh/id_rsa
> tail ~/.ssh/id_rsa.pub → Copy key. and paste in github created repo. SSH key creation page.
> git push -u origin master → up to date.

Now if we add file.txt in Git_Folder (D-drive) then that file will also be added in github website repository. after → ① git add .
② git commit
③ git push -u origin master

Setting Alias in Git =

> git config --global alias.st status
> git st

* One file modified →
> git add .
> git st
> git config --global alias.unstage 'restore --staged --'
> git unstage this.txt =

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

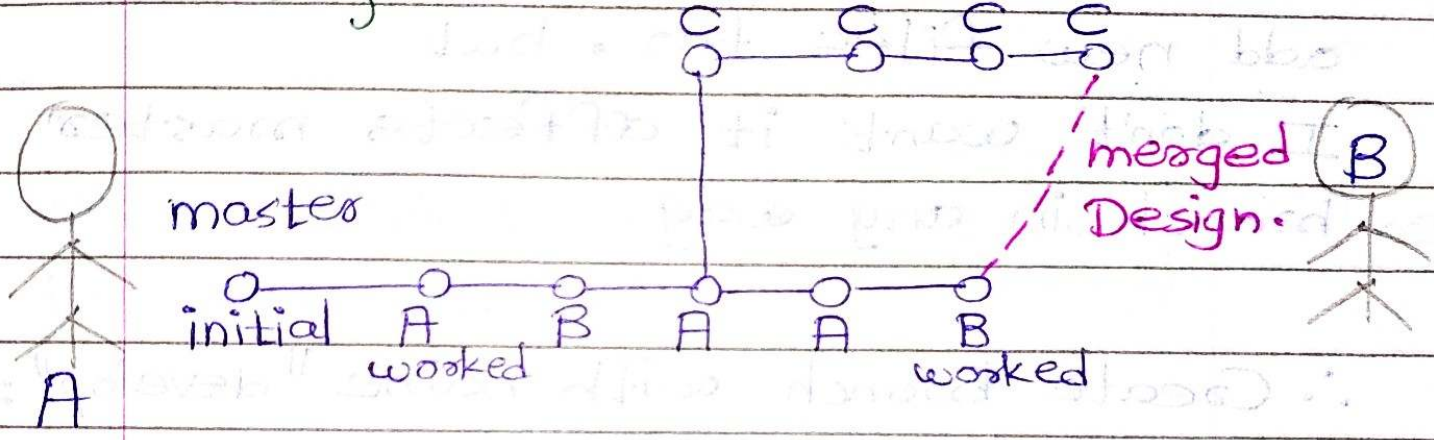
> git last

PAGE NO.:

DATE: / /

* If you need some command, over and over again, then make alias of it.

Creating & Switching branches in Git=



- branch is created
- if u want, u can merge.
- if u don't want then delete.
- Branch is version of the repository that diverges from main working project.
- Git project can have more than one branch.
- branches are pointer to snapshot of changes.

* When we add new feature, or when we fix a bug, we make new branch to summarize changes.

* Git Master Branch =

* If I want to delete many files and add new files too, but I don't want it affects master branch in any way.

∴ Create branch with name "develop" :-

> git checkout -b develop = creates new branch and switch to it new branch.

* Now do whatever changes you want, add, delete files.

> git status = 1 delete dir/ untracked files

> git add . =

> git commit -m "beautified structure" =

Now, to change branch,
Goto master branch=

PAGE NO:
DATE : / /

> git checkout master =

> git checkout develop = switch branch.
> git branch = to list all branches.

* We can use Git Bash / Vs Code cmd line to work with git.

* Download & enable "Live Server" on VsCode.

* Use bootstrap to create webpage easily.

```
> git add .
> git commit -m " "
> git checkout branch2
```

① Master
② cheoutCleanUp } merge.

> git checkout master =

> git status =

> git merge toyCleanUp =

PAGE NO.:

DATE: / /

⟶ It will show conflicts in VSCode; choose one of them.

(Accept incoming change)

> git status =

⟶ You have unmerged paths

> git add . =

> git commit = escape :wq to save.

Resolving Merge Conflicts (example)

Master branch-1

Issue1 branch-2

<<< = Conflict resolution master

> git branch
> git checkout -b issue 1
> git add -a -m "Commit4"

PAGE NO.:
DATE: / /

> git commit -a -m "Commit5"
> git commit -a -m "C7"
> git status
> git checkout master
> git commit -a -m "C6"
> git merge issue 1 = Coflicts solve.
> git add index.html
> git commit -m "merged issue 1"

> git log = "q" to exit

> git checkout -b issue 2
> git add .
> git commit -m "Did some gdbd"
> git checkout master

* 3 branches.

> git branch -V

master	Commit_hash	Commit_msg
develop	23#9431	"merged"

> git branch --merged = issue1 ✓
* master

> git branch --no-merged = issue2 ✗

* we have merged issue 1, so we can delete it.

* We have not merged "issue2".
If we try to delete issue2 give error. (Use -D)

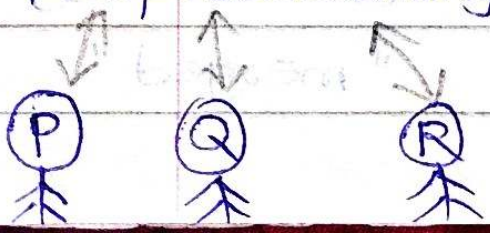
> git branch -d issue2 = gives error
> git branch -D issue2 = deletes branch

Git Branches = A to Z.

- branching workflow

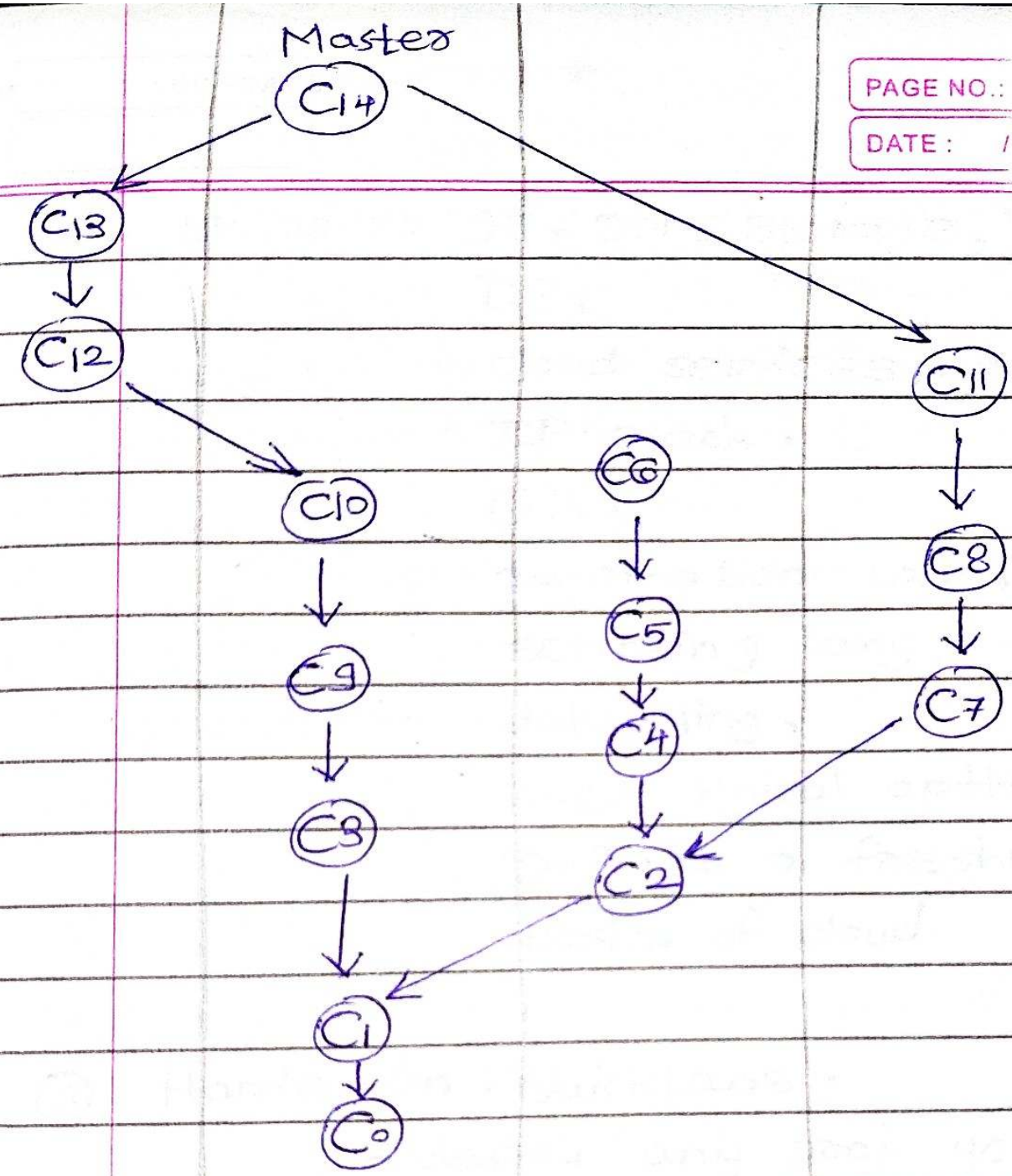
① Long Running Branches
- master
- develop
- proposed_changes

② Topic Branches
- typedJS Integration



PAGE NO.:

DATE : / /



Idea master issue1 issue2