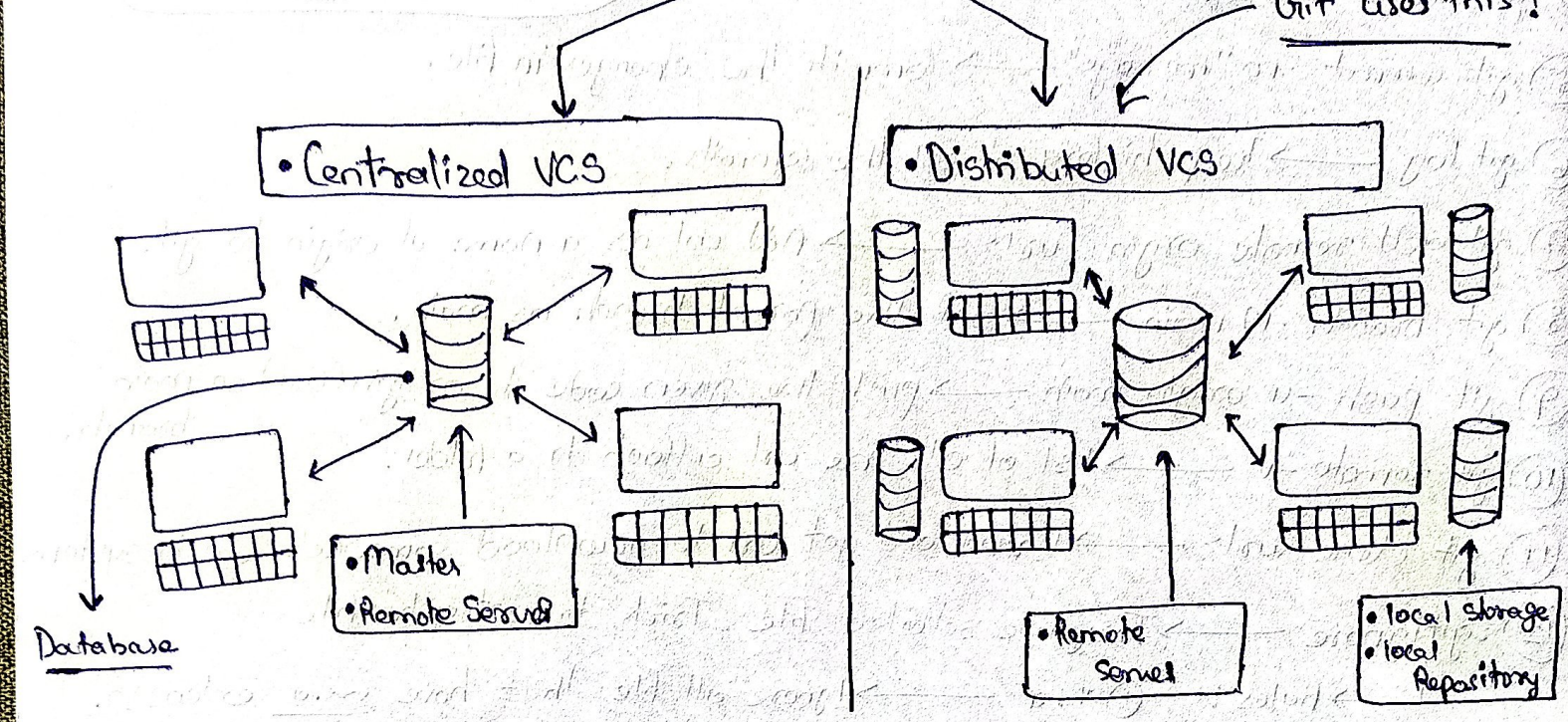
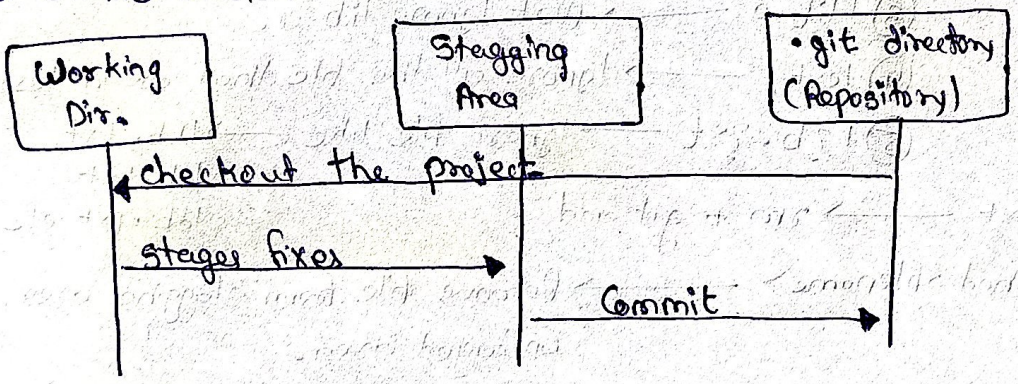
Hello Git!

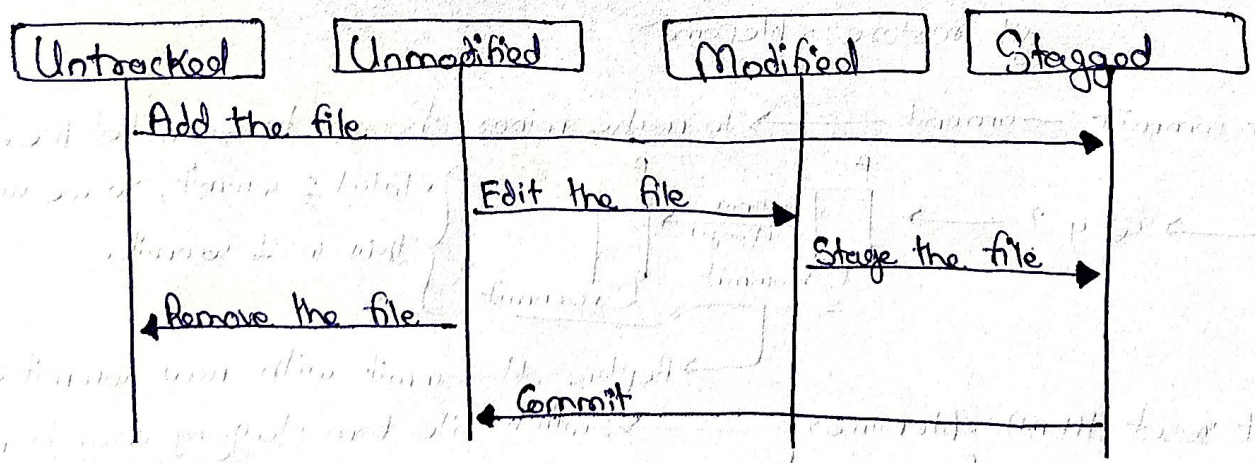
→ VCS ⟶ Version Control System : It is a system which records changes to a file or folder.



3 Stages of file in GIT :—

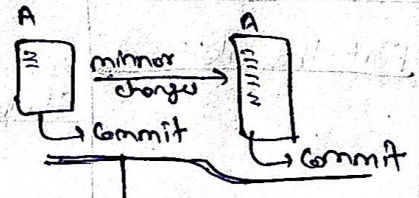


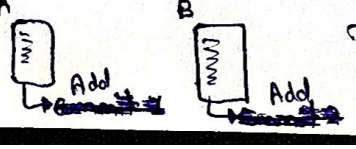
lifecycle of git files :—



Git commands :—

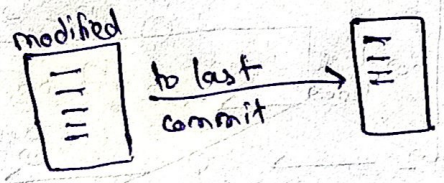
> **Hint box :—**
> - mkdir → make directory
> - ls → ls of all file
> - ls -a → ls of all file including hidden file
> - touch file_name → create new file.

1. git init → Initilize the git in folder.
2. git status → Give the status
3. git add file_name → Add file to tracked file.
4. git add . → Add all file
5. git commit -m "message" → Commits the changes in file.
6. git log → Keep history of all the commits.
7. git add remote origin <url> → Add url as a name of origin to git.
8. git branch -M main → Set the present branch as main.
9. git push -u origin main → push the given code to origin (url) on main branch.
10. git remote -v → list of all the url attach to a folder.
11. git clone <url> → forked and get url to download someone's code in system.
12. .gitignore → To ignore selected file. Trick to untrack file.
    - Rules :—
      1. *.a → Ignore all file that have .a extension.
      2. *.[oa] → Ignore all file that have *.a, *.o.
      3. !lib.a → Not Ignore lib.a.
      4. test/ → Ignore all the file that are in test folder.
      5. !?b.?xt → Ignore file like → lkb.txt, lib.txt ldb.pxt etc.
13. git rm hello.txt → rm + git add
14. git rm --cached <filename> → Remove file from stagging area, back to unstaged area.
15. To restore! →
    git restore --staged <filename>
    git restore <filename>
16. git commit, --amend → To make minor changes to commited file.
    - why ? →
      

      - Total 2 commit, so we can do this in 1 commit.
    - Replace old commit with new commit.
17. git reset HEAD <filename> → Switch file from stagging area to modified.
    - why? →
      

      - Both file are ~~untracked~~ ~~committed~~ added ~~separately~~ but if we want to commit one then we use.
      - Git reset to rescue !

(18) git checkout -- <filename> ⟶ If you don't want changes and go back to last commit. (modified ⟶ unmodified)

↳ why? ⟶ 

If you don't want changes.

Note :— ~~Perserve~~ Permanently deleted hence we use git stash.

(19) git stash ⟶ Undo changes and saved to local memory.

↳ git stash pop ⟶ Get undo changes back to your code.

↳ git stash clear ⟶ clear up all the stash.

(20) git restore <filename> ⟶ (1) staged ⟶ modified.
(2) modified ⟶ unmodified.

New command that replaces git checkout & git reset.

(21) Aliasing ⟶ (2nd name) git config --global alias.custom_name <cmd_to_replace>

↳ Eg:— git config --global alias.stage 'add'.

(22) git branch <branchname> ⟶ To create a branch

(23) git checkout <branchname> ⟶ Take you to that branch

git checkout -b <branchname>

(24) git merge <branchname> ⟶ Merge all files to the checkout useli branch.

(25) git branch ⟶ list of all branches.

(26) git branch -d <branchname> ⟶ After merging branch to main. Delete that branch using this command.

(27) git log --all --decorate --oneline --graph ⟶ To see all branches visually.

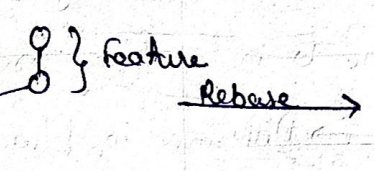
(28) git fetch ⟶ (1) git fetch origin
(2) git merge

OR (29) To keep up to date.

(29) git pull upstream main ⟶ Commit all the upstream changes to your local system.

(30) git remote add upstream "url" ⟶ Add url form which you forked.

(31) git rebase ⟶ apply on base

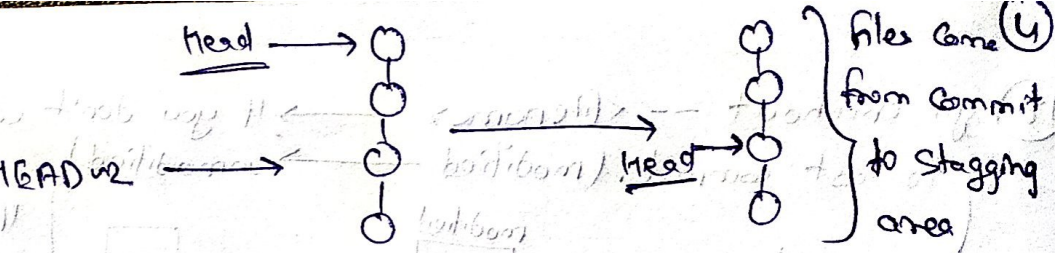


(1) git rebase main
(2) git rebase -i main

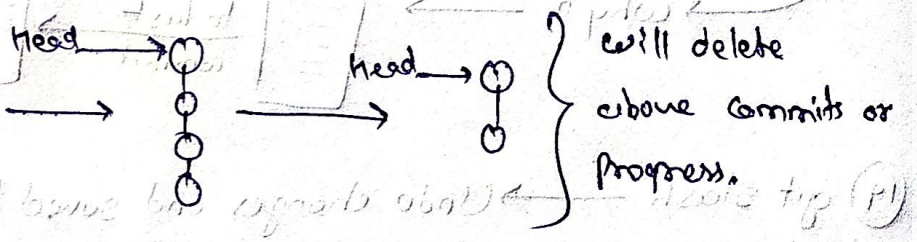
Note!— Don't use when you are contributing open source. This will create mess!
⟶ Use to rewrite the commit history.

(32) git reset :—

(A) git reset --soft HEAD~2



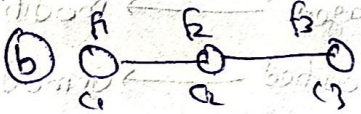
(B) git reset --hard HEAD~2



(33) git revert <commit id> ⟶ what ? ⟶ und(commit, merge).

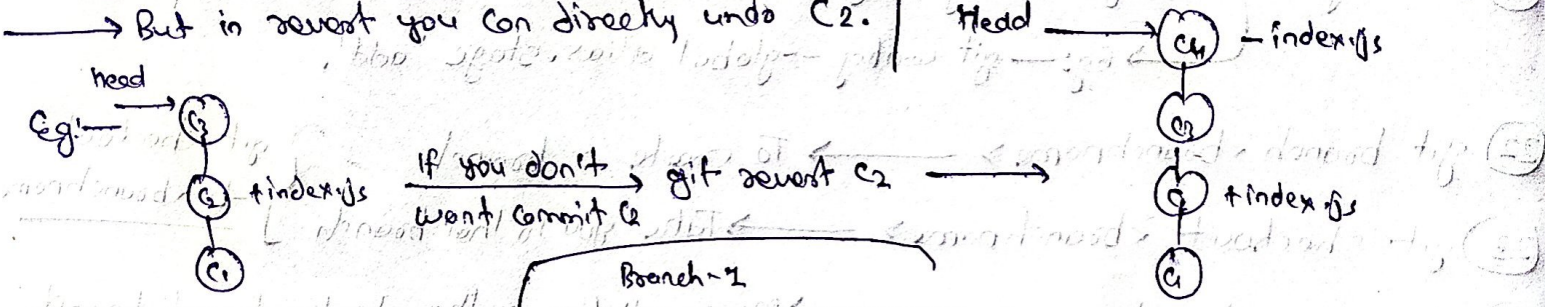
↳ why ? ⟶ (a) Does not re write history.

(b)

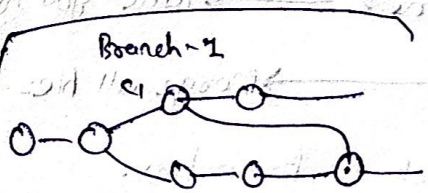


If you want to undo C2 then by using ~~revert~~ reset firstly you have to undo C3 & then C2.

⟶ But in revert you can directly undo C2.



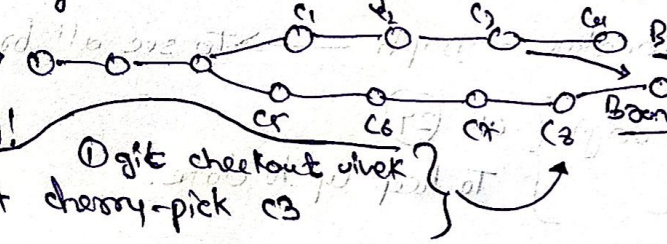
(34) git cherry-pick ⟶



⟶ Can make copy of C1 commit of branch 1 to branch 2.

↳ why ⟶ (1) Undoing ⟶ When you commit to different branch. Then to undo it.

(2) team ⟶



⟶ Can make copy of someone code to your branch.

(3) Bug resolved!

① git checkout vivek

② git cherry-pick C3

(35) git reflog ⟶ shows the history of action in the repo.

(36) git rebase -i log_code ⟶ Use to merge all the above commits to one commit.

(37) git diff <filename> ⟶ Shows changes in a given file.

↳ --staged ⟶ If file is in stagging area.

(38) git diff <branch1>..<branch2> ⟶ Difference in b/w two given branch.