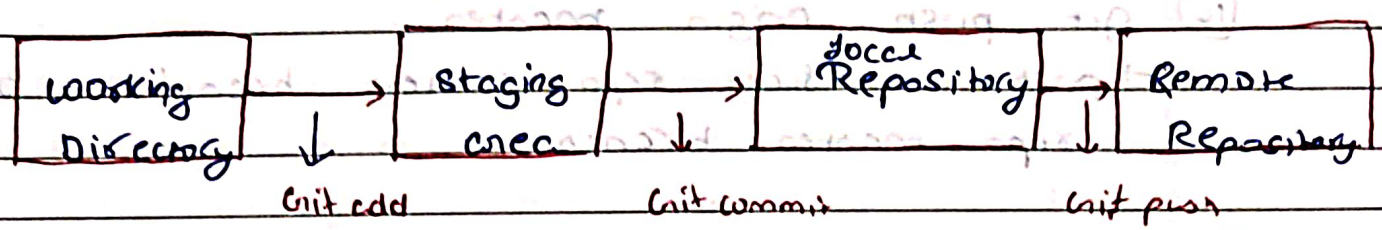
Git

① Git init / rm -rf .git
→ Initialize git in project → Remove Git from project

② Git Status
→ current status of git



③ Git add . / Git add <file name>
→ working area to staging area.

④ Git restore --staged . / git restore --staged <filename>
→ Staging area to working area.

⑤ git Commit -m "message"
→ Staging area to local repository.

⑥* Git Remote

⑥ Git remote
→ show all remote repository.

⑦ git remote -v
→ show all remote repository with URL()

(8) git remote add origin url( )
→ add remote repository

(9) git remote remove origin
→ Remove remote repository.

* Push & pull

(10) git push origin master.
→ push ~~git~~ local repo current branch to remote repo master branch.

(11) git pull origin master
→ pull from remote repo branch master to current branch.

* Git collaboration
* Git issue
* Git pull request

* Git hash & git blob
→ 40 character hash is created

(12) git cat-file -p <40 characters/hash>
→ Read

(13) git cat-file -t <40 characters/hash>
→ type of file

* Git Stash

→ Kind of locker, we can dump some code & use whenever needed

→ Follo to FILO (First In Last Out)

→ Why? → not to loose code, nor to push code.

(14) Git stash list

→ Show stash list

(15) Git stash push -m "message"

→ Staging area to stash

→ working area to stash

→ Untracked files does not go to stash.

(16) Git stash --include-untracked

→ Untracked files to stash

(17) Git stash apply

→ Restore last change

(18) Git stash apply stash@{}

→ Restore stash@{} code

(19) Git stash pop

→ Restore last change

→ Delete last change.

(20) Git stash pop stash@{}

→ Restore stash@{}

→ Delete stash@{}

(21) Git stash drop
→ Delete last stash

(22) Git stash drop stash@{3}
→ Delete stash@{}

(23) Git stash clean
→ Clean stash

* Git gignore v/s	Git stash
• gitignore	• stash
• not to loose code not to push code	• nor to loose code not to push code
• line of code is applicable in code	• line of code is not applicable in code

*GitBranch
→ It is individual line of development.
→ All branches are isolated from each other.
→ We can imagine, while creating new branch, the earliest code is copied in this branch.

(24) Git branch
→ Get list of branch.

(25) Git checkout (branch name)
→ To switch branch

(26) Git branch <branch name>
→ To create branch

(27) Git ~~bran~~ checkout -b <branch name>
→ To create branch
→ To switch branch.

(28) Git branch -m "new name"
→ Rename current branch name

(29) Git branch -m <old name> <new name>
→ Rename old branch name.

(30) Git merge <branch name>
→ Merge any branch to current branch

(31) Git branch -d <branch name>
→ Before deleting branch it ensure, it is merged

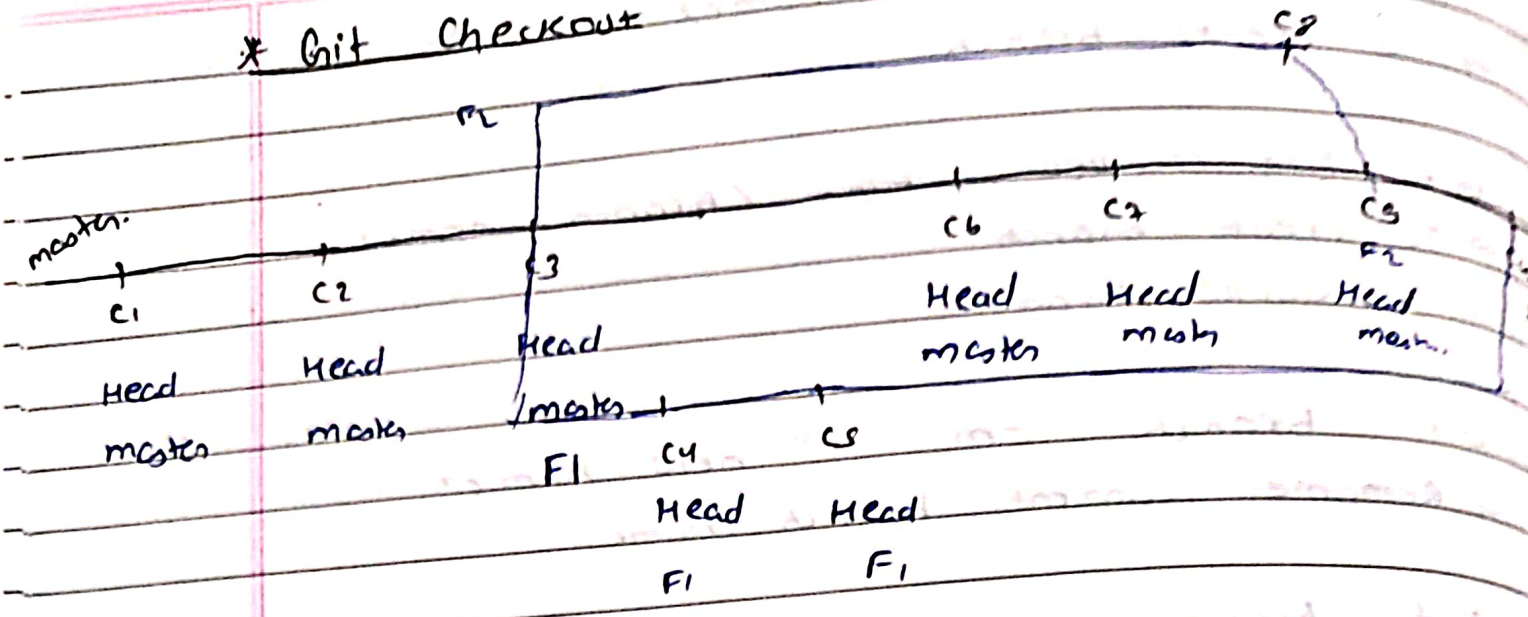
(32) Git branch -D <branch name>
→ It directly delete branch.

(33) Git branch -r
→ Show branch which is connected ~~to current~~ remote repo.

(34) Git branch -a
→ Display all branch

(35) Git branch -- show - current
→ Show current branch.

* Git Checkout



① Create a branch by moving on any previous commit
→ git checkout Head HEAD~1 / <commit hash> / tag
→ git branch f1

* Git Revert

(36) Git revert HEAD
→ Revert Change
→ Commit

(37) Git revert HEAD~1 / tag / <commit hash>
→ Revert that commit
→ Commit.

(38) Git revert -n HEAD
→ Revert Change
→ Uncommitted.

0) Revert by & by bork.

git revert HEAD

git revert HEAD~2

→ counting starts with zero.

* Git Reset

(39) Git reset -- soft HEAD~1 / <commit hash> / tag

→ Bring commit to staged area

→ Staged area to staged area

→ Working area to working area.

(40) Git reset -- mixed HEAD~1 / <commit hash> / tag

→ Bring commit to working area

→ Staged area to working area

→ Working area to working area.

(41) Git reset -- hard HEAD~1 / <commit hash> / tag

→ Clear commit

→ clear staging area

→ clear working area.

* Git tag

(42) Git tag -a <tag name> -m "new tag"

commit (head, branch, tag: t1)

Use→ checkout, reset { tag, HEAD~1, <commit hash>}

(43) Git ~~tag~~ push origin master -- force

→ Push forcefully.