pwd ls cd

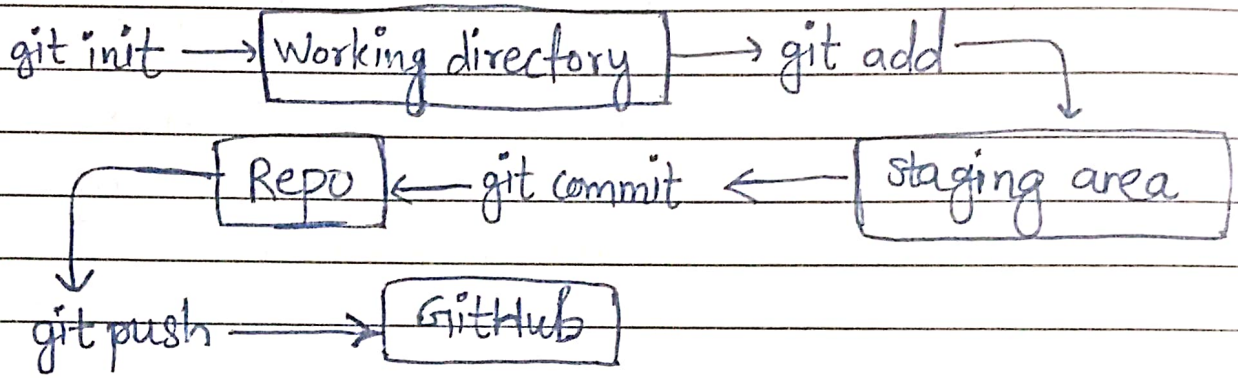
Git → version control
↳ software

github → service

git --version
git status
git init

git config --global user.email "@gm..."
" --name "Suyog"

write → add → commit



git add .
git add <file1> <file2>

git commit -m "First commit"

git log
git log --oneline

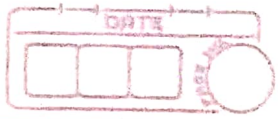
Atomic Commit (Present tense & Imperative)

.gitignore .env .gitkeep ⇒ to track empty parent folder

git branch

git branch bug-fix ⇒ create new branch but remains in old branch

git graph tool 



git switch bug-fix ⟹ now current branch is bug-fix
↳ git checkout bug-fix

git switch -c dark-mode ⇒ create new branch of name dark-mode & now current branch is dark-mode
↳ git checkout -b dark-mode

To come back

git checkout <hash commit> ⇒ Snapshot of that commit

git checkout <commit hash> -- <file name>
↳ Restore given file changes to provided commit.
(only that one file remains same as old)

↳ git checkout branch-name

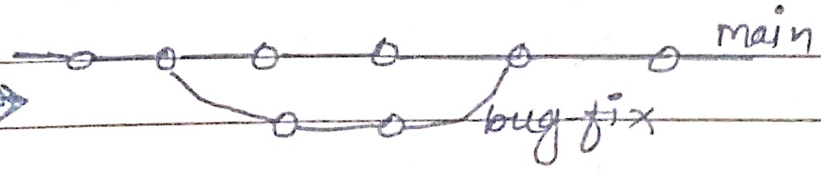
* Come to main branch and then merge the sub branch

fast forward merge

git checkout ~~main~~ master
git merge bug-fix

* Not fast-~~for~~ forward merge

Merge conflict ↳



If conflict occur during merge then add & commit

Rename a branch
git branch -m <old-branch-name> <new-b-n>

delete branch
git branch -d <branch-name>

checkout a branch
git checkout <branch-name>

List of all branch
git branch

* Git diff :- is an informative command that shows the diff. beth 2 commits.

How to read the diffs
a → File A and b → File B
--- Indicates the file A
+++ Indicates the file B
@@ indicate the line number

(here file A & B are same but diff. versions)

git diff ⇒ we have to ~~use~~ provide some more options to show changes.

git diff --staged
↳ shows the changes beth your last commit & the staging area (ie. changes that are staged & ready to be committed)

git diff <branch-name-one> <branch-name-two>
or → ——— || ——— .. <branch-name-two>

~~git diff~~

* Comparing specific commits

git diff <commit-hash-one> <commit-hash-two>

* Git stash :- (git stash)

stash is a way to save your changes in a temporary location. It is useful when you want to make changes to a file but don't want to commit them yet. You can then come back to the file later & apply the changes.

Naming the stash :- git stash save "work in progress"

View the stash list :- git stash list

Apply the stash :- git stash apply

Apply specific stash :- git stash apply (stash@{0}) ← name of stash

Apply & droop the stash :- git stash pop

this command applies the stash & droops it from the stash list.

Drop the stash :- git stash drop

Applying stash to a specific branch :-

git stash apply stash@{0} <branch-name>

clearing the stash :- git stash clear

→ mostly used in upgrade the version

Git tags :-

Tags are a way to mark a specific point in your repository. They are useful when you want to remember a specific version of your code or when you want to refer to a specific commit. Tags are like sticky notes that you can attach to your commits.

Creating a tag :- git tag <tag-name>

(attached to a current commit)

Create an annotated tag :- git tag -a <tag-name> -m "Release 1.0"

→ create an annotated tag with the specific name & message.

List all tags :- git tag

Tagging a specific commit :- git tag <tag-name> <commit-hash>

Push tags to remote repository :-

git push origin <tag-name>

Delete a tag :- git tag -d <tag-name>

Delete tag on remote repository :-

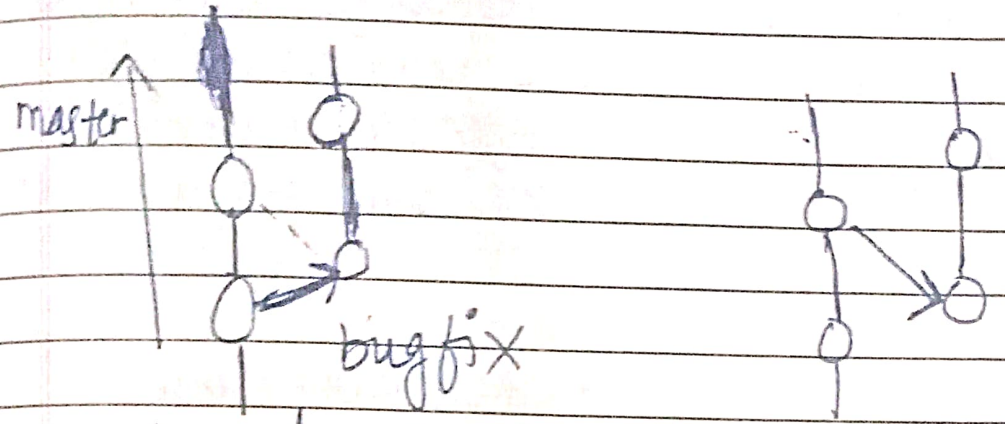
git push origin :<tag-name>

It rewrite the history so we should avoid.

If someone says to you that rebase the branch means merge the branch

* Rebase in git :-

used to change the base of a branch.



* It reduces one merge commit
* rebase always do by standing in sub branch
* git rebase master → current branch is bug-fix

Git reflog :- This will show you the history of commits. git reflog

find specific commit → git reflog <commit-hash>

Recover lost commits or changes :-

git reflog <commit-hash>
git reset --hard <commit-hash>

or you can use HEAD@{n} to reset to the nth commit before the one you want to reset to.

git reflog <commit-hash>
git reset --hard HEAD@{1}

GitHub :-