Git

- Distributed source control system
    - not required to be decentralized.
- Massively scales
- Open source
- Developed for linux project requirements.
- Most operations are local
- Very fast
- Active community.
- Most popular DVCS, VCS.

- Key concepts -
- Repository contains files, history, config managed by git.
- 3 states of Git.
    - Working directory
    - Staging area - pre commit holding area
    - Commit - Git Repository (history)
- Remote repository (Git hub)
- Master branching.

- Basic Git workflow → (add, commit, pull & push).

- Comparisons → In order to compare 2 branches easily, we have to use the "git diff" command and provide the branch named separated by dots.

~~[illegible]~~
~~Merging~~
~~[illegible]~~

- Branching → It allows each developer to branch out from the original code base and isolate their work from others. It also helps git to easily merge versions later on.
"git branch ~~i~~issues"

- Merging → It is a procedure to connect the forked history. It joins 2 or more development history together. It facilitates you to take the data created by git branch and integrate them into a single branch. It will associate a series of commits into one unified history. Generally, it is used to combine 2 branches.
~~G~~ "git merge"

- Rebasing → It ~~is~~ is the process to reapply commits on top of another base trip. It is used to apply a sequence of commits from distinct branches into a final commit. It is an alternative of git merge command. It is a linear process of merging.
It is referred to as the process of moving or combining a sequence of commits to a new base commit. It is very beneficial and it visualized the process in the environment of a feature branching workflow.
It is good to rebase your branch before merging it.

"$ git rebase <branch name>" | "$ git skip"
"$ git status" | "$ git checkout master"
"$ git rebase --continue"

Stashing → Sometimes you want to switch the branches, but you are working on an ~~to~~ incomplete part of your current project. You don't want to make a commit of half-done work. It allows you to do so. The git stash command enables you to switch branches without committing the current branch.
Generally it means "store something safely in a hidden place". The sense in git is also the same for stash. Git temporarily saves your data safely without committing.

$ git stadus	Git stadus
$ git stash	Git stash
$ git stash save "<stashing message>"	Git stash save
$ git stash list	→ Git stash list
$ git stash apply	Git stash apply
$ git stash apply <stash id>	
$ git stash show	Git stash changes
$ git stash show -p	
$ git stash pop	→ Git stash pop
$ git stash drop	
$ git stash drop <stash id>	Git stash drop
$ git stash drop stash@{1}	
$ git stash clear	→ Git stash clear
$ git stash branch <Branch Name>	→ Git stash branch

- Tagging → Tags make a point as a specific point in git history. Tags are used to mark a commit stage as relevant. We can tag a commit for future reference. Primarily, it is used to mark a project's initial point like V1.1.

Tags ~~make a~~ are much like branches, and they do not change once initiated. We can have any no. of tags on a branch or diff. branches. ~~The below~~

Types:-

- Annotated tag | `$ git tag <tag name> -m "<tag msg`
- Light-weighted tag | `$ git tag <tag name>`
  "`$ git tag projectv1.0`"

```
$ git checkout <Branch Name>
$ git tag <tag name>
$ git tag projectv1.0
```
→ Git create tag

```
$ git tag
$ git tag show <tagname>
$ git tag show project v1.0
$ git tag -l "<pattern>"
$ git tag -l "pro"
```
→ Git list tag

```
$ git push origin <tagname>
$ git push origin --tags
$ git push --tags
```
→ Git push tag

"$ git tag --d <tagname>
OR
"$ git tag --delete <tagname>"
"$ git tag --d project v1.0"

Git delete tag

Delete a remote tag

"$ git push origin -d <tag name>"
OR
"$ git push origin --delete <tag name>"

Delete multiple tag

"$ git tag -d <tag1><tag2>"
"$ git push origin -d <tag1><tag2>"

$ git checkout -b <new branch name><tag name>

Git checkout tags