Git and Github

1. Git -> It is a version control which allows us to store history of project. At a particular point of time which person made which changes and where in the project
2. Github -> It is platform which allows us to host our git repositories.
3. repository -> folder where all the changes are saved.
4. .git -> .git folder in our project is a hidden folder which maintains history of project.(create new file,make changes in file,delete file).
5. **git init** -> initialize an empty git repository
6. **git status** -> git status command tells us the changes which are not currently in the history of our project by highlighting it in red.
7. **git add .** -> everything in the current project directory,that is not having it's history (untracked),put all those file in a staging area.
8. **git commit -m "names.txt added"** -> Adding commits keep track of our progress and changes as we work. The Staging Environment has been committed to our repo, with the message: "names.txt added"
9. **git restore --staged names.txt** -> If we put names.txt in staging area by mistake,we can remove that file from the staging area without committing it in our repo using git restore command.
10. **git log** -> history of commits
11. **git reset hashid** -> git reset is the command we use when we want to move the repository back to a previous commit, discarding any changes made after that commit.We reset our repository back to the specific commit using git reset hash (*hash* being the first 7 characters of the commit hash we found in the log):
12. **git stash** -> The git stash command takes your uncommitted changes (both staged and unstaged), saves them away for later use.
13. **git stash pop** -> Popping your stash removes the changes from your stash and reapplies them to your staging area.
14. **git stash clear** -> changes which were not committed and in the stash,they are now cleared from stash and you cannot retrieve them back into staging area.
15. **git remote add origin url** -> creates connection between our local project and git repository.
16. **git remote -v** -> all the Url's attach to the folder
17. **git push origin master** -> push the changes from local repository to git repository master branch.
18. Branch structure - commits link to each other.
19. Whenever we are working on a new feature or any bug fix, always create a separate branch.
20. Never commit directly on main branch,because there might be some bugs which may affect the user.
21. **git branch feature** -> create a new branch feature.
22. **git checkout feature** -> all the commits from now will be made to feature branch. Head is pointing to feature branch.
23. **git merge feature** -> feature branch is now part of the main branch.
24. fork -> create a copy of a project in my own account. You cannot make changes directly to someone's account.
25. **git clone url** -> clone a project locally in your system.
26. upstream url -> from where have you forked the project
27. **git remote add upstream url** -> Setting a remote upstream and fetching it time to time makes sure

your forked repo is in sync with the original repo.

28. Pull request -> Pull requests are a mechanism for a developer to notify team members that they have completed a feature. Once their feature branch is ready, the developer files a pull request. This lets everybody involved know that they need to review the code and merge it into the main branch.
29. **git push origin branchname** - > push the changes to branch.
30. create a new branch for every feature or every bug fix so that you can create a new pull request.
31. **git push origin branchname -f** -> sometimes you have to force push because online git repository contains a commit that my local repository does not.
32. Sync Fork -> To keep main branch upstream and our fork repository main branch up-to date.
33. **git fetch --all --prune** -> fetch all the commits from upstream repository to our forked repository locally.
34. **git reset --hard upstream/main** -> reset my main branch of origin to my main branch of upstream.And them do git push origin main.
35. **git pull upstream main** -> does the same things pulls the changes from upstream main to our local forked repository and then do git push origin main so that upstream main branch and our forked repository main branch is up to date
36. **git rebase -i hashid** -> merge all the commits into one single commit.All the commit above it we can pick or squash.above 's' whichever pick you have merge those squash commits into one 'pick' commit.
37. Merge Conflicts -> I made a change in line no 3 and some other person also made change in line no 3,then git might get confused which change to merge.We have to go in pull request and manually solve those conflicts and tell git which change to merge.