***GitHub & Git Notes***

**General Points**

1. ***– git add .*** = will track files and changes in git.
2. ***– git add <file-name>*** = change the tracking stage of specific mentioned file.
3. ***– git commit –m “<commit-message>”*** = save your files in git.
4. ***– git push origin <branch-name>*** = makes the code on your local devices upload to GitHub.
5. ***– git push –u origin main =*** -u here says that whenever we use only “git push” the git will automatically upload to origin main as **“-u”** work is to assign default branch to push your code when no branch is mentioned. It goes not affect the basic way to push or pull your code. –u sets a default upstream.
6. **Origin** = it is a naming convention pointing to original repository. It refer to the default remote repo connected to local repo or where it was cloned. I guess the local repo is treated as copy of remote repo.
7. ***– git pull origin <branch-name>*** = pulls changes from the branch on remote site with similar name as requested branch name of the origin remote repo regardless of the current branch tracking configuration.
8. ***– git remote add origin <remote-repo-URL>*** = the local repo and remote repo which is on GitHub connects with this command.
9. ***– git branch –M main =*** renames the default branch to main which may be called master. *–M* represent –force or –move.
10. ***– git status =*** check the status of tracked files and commits.
11. ***– git checkout <branch-name>*** = moves the attention of git to mentioned branch.
12. ***– git checkout –b <branch-name>*** = creates a new branch of that specified name and switch the attention to the newly created one.
13. ***– git clone <repo-URL>*** = will clone any remote on GitHub to your local machine. But if you want to make a copy of other’s repo then fork it repository using your account then clone the forked repo in you local machine and make changes. You can create a pull request to the original owner of that repo and ask him to review the pull request, if the change suites him you can **COLLABORATE** or become a **CONTRIBUTER** on the original git repo.
14. ***– git remote –v*** =helps to see the URL your remote repo which your local sends the data.
15. ***{git pointer on<branch\_1>}: -- git diff <branch-2***> = to check the difference of files and there data between two separate branches.
16. ***{git pointer on<branch\_1>}: -- git merge <branch-2>*** = to merge the two different branches offline locally.
17. ***– git commit –am “<commit-message>”*** = -am stands for add modified. With this –am prompt we can commit a modified file without changing its stage because it is been tracked but git already. –am will not work for files who are newly introduced as its not modified file as tracking is important for git to do its job.
18. ***– git diff*** = to check the difference in the old and new code.
19. ***– git reset <file-name>| --git reset*** = unstage the staged file.
20. ***–git reset HEAD~1*** = this command will uncommit and unstage the previous commit by one step. I guess HEAD points to list of commits but from bottom to top.
21. ***– git reset –hard <alphanumeric-commit-id>*** = Totally go back in time and reflect the changes till that commit id. Now HEAD points to that <alpha-commit-ID> as its now the latest one.
22. ***– git branch <new-branch-name>*** = will create new branch
23. ***– git switch <branch-name>=*** sole purpose of switching to other branches.
24. ***– git branch –m <new-branch-name>*** = will rename the branch HEAD is pointing too.
25. ***– git branch –m <old-branch-name> <new-branch-name>*** = will change the branch name of a branch who is not pointed by HEAD at the current moment.
26. ***– git branch --track <local-branch-name> origin/<remote-branch -name>*** = if there is a branch in remote repository(GitHub) and you want to bring that remote branch to the local branch with having <newly-created-local-branch> tracking the <origin/remote-branch> used this command line.
27. ***– git checkout --track origin<remote-branch-name>*** = the purpose is same as above snippet but with checkout even we do not specify the branch name git will copy the name as it is form remote repo. In both case HEAD will be pointing to newly created branch.
28. ***– git branch –v*** = see the list of branched you have.
29. *Ahead* in GitHub means you have some commit in your local repo that remote repo does not have.
30. *Behind* in GitHub means there are some commits in you remote branches that you have not pulled in your local repo.
31. ***– git branch –d <branch-name>*** =deletes the branch on *local* machine, cleaning the branch which will be not be used is a good practice. Deleting the current HEAD branch is not possible.
32. ***– git push origin --delete <branch-name>*** = will delete the branch that is on *remote* while you type this command line in local desktop.
33. ***– git log <branch-1-name>..<branch-2-name>*** = this will compare <branch-1> to <branch-2> and tell the number & details of commits that are different between two.
34. ***– git log origin/<branch-of-remote-repo>..<branch-of-local-repo>*** = this will compare the branch commit status between remote and local branch.
35. ***– git log --oneline*** = The git log --oneline command is used to display a simplified version of the commit history in Git
36. ***– git merge <branch-A-name>*** (meanwhile HEAD points to branch-B) = to have updates of branch-A into branch.
37. **Shift + :** = will return the command line to normal, use this when u have a merge conflict and you have done you changes into the HEAD branch and commit it (a page will open about merge conflict) use this syntax to go back to normal cmd and it will automatically commit + merge the changes you were doing.
38. Git command reference repo: <https://github.com/joshnh/Git-Commands>
39. Directories/working directories are your current state of project.
40. When we do **<git add .>** it kind of bundles the changes into one and that bundled space is referred as **Staging Area.**
41. Changes in working directories -> moved in staging area -> commit + saved in local repo -> push in remote repo.
42. How to remove a file from staging area and also remove from the folder (delete) ***-git rm –f <file.name>***
43. Unstage a file = ***git rm –cache <file-name>***
44. ***– git revert*** = rollback to the previous commit
45. ***– git revert <commit-Id>*** = revert the changes that are made in that commit.
46. ***– git log <commit-Id>*** = to go back to the commit and check the changes details.
47. ***– git reset –hard <commit-ID*>** = make the Head point to the mentioned commit-ID and delete all the progress of the commit history.
48. – git rm –r –cache: when a .gitingnore is introduced manually this will reconfigure the tracking system and from now onward files mentioned in .gitignore will be ignored.

**::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::**

**Q&A**

* What is publish branch in Git hub? And why when I pushed by code via VS code the same did not reflected in Git hub website and after I published the branch the changes reflected!
* Do people make a branch for update, bug fix etc. do the changes in the code deploy on GitHub, get merged with main branch and then delete that very branch, is this practice is good?