GIT Commands!!

Pattern :- Sr.no : command // meaning

1. git init // it includes the file in the repository

2. git add --a // add files to staging area

3. git add . // add files to staging area

4. git status // check the status of the files in the repository

5. pwd // present working directory

6. ls // list : shows the list of files in repository

7. cd // change directory

8. git add _file_name_with_extension // add a file to the staging
area eg. git add file1.txt
and can be done with multiple files

for eg.

git add file1.txt file2.txt file3.txt

9. touch _file_name // makes a folder

10. .gitignore // the names written in this folder that files will get ignored

For whole extensions : *.txt

For whole folder : /bin

11. git clone _url // to copy a repository from github

12. git commit -m "_ur_msg" // this commits the files which are in the staging area.

13. rm -rf .git // this deletes the current repository.

14. git diff // compares two files which are changed after committing

15. git diff --staged // compared the committed file and the last staged file.

16. git commit -a -m "message" // this skips the process of adding files to the staging area and then commits . It

directly commits the files i.e the tracked files and the untracked files won't get committed.

17. git rm _file_name // removes or deletes the file in the repository.

18. git mv _name1 _name2 // this is for renaming the file here name1 is original file and name2 is the name of renamed file

19. git log -p // show all the changes made in the commits

20. git log -2 // shows 2 commits

21. git log -p -2 // shows the changes of 2 commits.

22. git log --stat // gives the overview of the commit changes

23. git log --pretty=oneline // gives the commits in one line format

24. git log --pretty=short // gives the commits in short format

25. git log --since=2.(time_period eg.days) // shows the commit made in 2 days (can be done for days , weeks , months and years).

26. git log --pretty=format:"%h -- %an" // this helps in formatting the commits in required format. Like here it will show:

Address -- AuthorName

//%an for author name

//%ae for author email

For more formatting styles:
(https://git-scm.com/docs/git-log)

27. git commit --amend // it overlaps the commit and a vim editor opens and to edit on the editor press "i" and to exit press "Escape" and then ":wq" .

28. git restore --staged _filename // this helps for unstaging a file in the repository

29. git checkout -- _filename // this helps you in restoring the data of your previous project.

(This is for specific file)

30. git checkout -f // this helps in removing the changes you made in the files
    (This is for all files present in repos.)

31. git remote add _name _url // push an existing repository from the command line

32. git remote // check the remote

33. git remote -v // it shows the url for fetching and pushing.

34. To make an SSH key :
(https://docs.github.com/en/github/authenticating-to-github/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent)

35. To push the changes you have in the repository for that the commands are shown when you make a new repository.
For reference:
git remote add origin git@github.com:AyushSama/dfj.git
git branch -M main
git push -u origin main

36. git config --global alias._name _ur command
// this helps in forming an alias of a required command which can be used frequently.

37. git checkout -b _newbranchname // this will help in making a new branch of required name

38. git checkout _branchname // this will help us to switch into branches.

39. git merge _branchname // this will merge the new branch with your master branch (And vice-versa.)

40. git branch -v // shows the branches with commit hashes and commit messages

41. git branch -d _branchname // this deletes a branch which isn't required.. In case you haven't merged this branch then it will ask you to confirm again .

42. git branch -D _branchname // here capital d is used so if in case you haven't merged this branch with master then too it will delete the branch without giving any error.
(Not recommended)

43. To solve the merge conflicts you have to choose one of the outputs from the two branches and then stage them then commit it already gets merged.

44. git push // it pushes the code or the repo to the github .

45. git push origin _branchname // if you want to push the branch to the github then use this command.

46. git push origin _branchname:new_branchname // this will create a new branch identical to the branch in the remote that is github bt it will not be created in git bash

Eg: git push origin Ayush:Mickey

Here the branch named Mickey will be created identical to Ayush in remote bt won't be created in git bash i.e the terminal.

(This is strongly not recommended!)

47. git push -d origin _branchname // this will delete the branch in the remote that is github .

Note: If you want to push any branch then push that branch while being in that branch.

By: Ayush A. Yadav