Git commands:

2. Git status: shows the status of working directory

3. npp <filename>: to create a file

4. Git add: to add a file from working directory to git’s staging area

5. Git commit -m “<message>” : commiting the file that was added in the staging area with a message.

6. Git rm -rf : to forcefully and recursively delete a folder in repository.

7. Git add . : to add all the file in the working directory to the staging area.\

8. Git commit : opens up the editor to write the commit message.

9. Git show and git log : to show all the pevious commits and messages.

10. Git commit -am : first add the updated file to the staging area and proceeds to commit also.

11. Git reset HEAD <file> : to unstage thechanges made to a file. i.e, if a file is added using git add , to reset the changes.

12.Git checkout -- <file> : to reset the changes entirely i.e. to get the previous safe version of file.

13. Git log –oneline –graph –decorate –all : (--oneline provides a lot of info in oneline), (--graph provide asterick based graph denoting branching hierarchy), (--decorate to tell us which commits are part of which branches), (--all to provide the history for all the branches that are available in this repository).

14. git config –global alias.<alias name> “<command without git prefix>” : to create an alias.

15. Git hist -- <filename> : to show the history regarding only that file.

16. Git mv <old file name> <new file name> : effectively renames the file.

17. Git rm <filename> : delete a file and also tracks the deletion.

18. Git add -u : handles all the deletion in the working directory and adds them.

19.Git add -A : handles all the insertions and deletions.

20.Git diff <commit hash> <hash> :

21.Git difftool <commit hash> <commit hash> :

22.Git branch : displays all the branch

23.Git checkout -b <branch name> : make a new branch

24.Git checkout <branchname> : switch to that branch

25.Git merge <otherbranchname>: merges <otherbranch> with current branch

26.Git branch -d <branchname> : delete that branch label obviously after merging

27.Git branch -D <branchname> : delete a branch even if it is not merged with master branch but will also delete any changes made in that branch.

28.Two types of tags in git: simple (less – info), annotated(more-info)

29.Git tag <tagname>: marks current head commit as tag.

30.Git tag –list: list all the tags.

31.Git tag -d <tagname> : delete the tag

32.Git tag -a <tagname> -m “<message>” : annotated tag at the current commit

33.Git show <tagname> : shows the tag and commit details.

34.Git stash : Often, when you’ve been working on part of your project, things are in a messy state and you want to switch branches for a bit to work on something else. The problem is, you don’t want to do a commit of half-done work just so you can get back to this point later. The answer to this issue is the git stashcommand.

35.Git stash pop : do two thing at a time, i.e. git stash apply and drop .applies the last stash and drop the last stash from the stack.

36. Git reset <commit\_id> --soft/mixed/hard

37.Git reflog : shows all the different actions done in the repository.

38.Git reset –hard/mixed/soft : resets the id from reflog.

39. git remote -v : displays the remote repository url

40. git remote add origin <*remote repository URL>:* Sets the new remote

*41.*git push -u origin master –tags : pushes the exisiting repository to remote repository

42. git pull --rebase origin master : consists of two comands fetch and merge. Can be destructive, advisable to use fetch first.

43. git fetch: fetches the remote repository and see if any changes.

44. git clone <remote repository URL>: clones the repository

45. git clone <remote repository URL> <folder-name>: clones the repository