**List of Useful Git Commands**

**1. git help**

Take help from the Git help section for different commands and other errors.

git help

**2. git config**

To set the basic configurations on Git like your name and email.

git config

**3. git config –-global user.name “ ”**

Sets configuration values for your user name on git.

git config –-global user.name “Ashish Madaan”

**4. git config –-global user.email ” “**

Sets configuration values for your user email on git.

git config –-global user.email ashishmadaan6@gmail.com

**5. git config –-global color.ui**

To see different colors on the command line for different outputs.

git config –-global color.ui true

**6. mkdir**

Create a directory if not created initially.

mkdir store

**7. cd**

To go inside the directory and work on its contents.

cd store

**8. git init**

To create a local git repository for us in our store folder. This will help to manage the git commands for that particular repository.

git init

**9. git status**

To see what’s changed since the last commit. It shows all the files that have been added and modified and are ready to be committed and files that are untracked.

git status

**10. git add Readme.txt**

To add a file Readme.txt to the staging area to track its changes.

git add Readme.txt

**11. git commit -m “ ”**

To commit our changes(taking a snapshot) and provide a message to remember for future reference.

git commit -m “Created a Readme.txt”

**12. git log**

To check the history of commits for our reference.

git log

**13. git add**

To add a specific list of files to the staging area.

git add

**14. git add –all**

To add all files of the current directory to the staging area.

git add --all

**15. git add \*.txt**

To add all text files of the current directory to the staging area.

git add \*.txt

**16. git add docs/\*.txt**

To add all text files of a particular directory(docs) to the staging area.

git add docs/\*.txt

**17. git add docs/**

To add all files in a particular directory(docs) to the staging area.

git add docs/

**18. git add “\*.txt”**

To add text files of the entire project to the staging area.

git add “\*.txt”

**19. git diff**

To figure out what changes you made since the last commit.

git diff

**20. git reset head license**

To undo the staging of the file that was added in the staging area.

git reset head license

**21. git checkout –license**

To Blow away all changes since the last commit of the file.

git checkout –license

**22. git commit -a -m “ ”**

To add any of our tracked files to the staging area and commit them by providing a message to remember.

git commit -a -m “Readme.md”

**23. git reset –soft HEAD^**

To undo the last commit and bring the file to the staging area.

git reset –soft HEAD^

**24. git reset –hard HEAD^**

To undo the last commit and remove the file from the staging area as well(In case we went horribly wrong).

git reset –hard HEAD^

**25. git reset –hard HEAD^^**

To undo the last 2 commits and all changes.

git reset –hard HEAD^^

**26. git remote add origin**

These commands make a bookmark which signifies that this particular remote refers to this URL. This remote will be used to pull any content from the directory and push our local content to the global server.

git remote add origin https://github.com/madaan123/MyAlgorithms.git

**27. git remote add <address>**

To add new remotes to our local repository for a particular git address.

git remote add <address> 

**28. git remove rm**

To remove a remote from our local repository.

git remove rm

**29. git push -u origin master**

To push all the contents of our local repository that belong to the master branch to the server(Global repository).

git push -u origin master

**30. git clone https://github.com/madaan123/MyAlgorithms.git**

To clone or make a local copy of the global repository in your system   
(git clone command downloads the repository and creates a remote named origin which can be checked by the command – git remote -v).

git clone https://github.com/madaan123/MyAlgorithms.git

**31. git branch Testing**

To create a new branch named Testing.

git branch Testing

**32. git branch**

To see all the branches present and current branches that we are working on.

git branch

**33. git checkout Testing**

To switch to branch Testing from the master branch.

git checkout Testing

**34. ls**

To see directories and files in the current directory.

ls

**35. ls -la**

To see hidden directories and files within the current directory.

ls -la

**36. git merge Testing**

To merge the Testing branch with the master branch.

git merge Testing

**37. git branch -d Testing**

To delete the Testing branch.

git branch -d Testing

**38. git checkout -b admin**

To create a new branch admin and set it as the current branch.

git checkout -b admin

**39. git branch -r**

To look at all the remote branches.

git branch -r

**40. git branch -D Testing**

To forcefully delete a branch without making commits.

git branch -D Testing

**41. git tag**

To see the list of available tags.

git tag

**42. git checkout v0.0.1**

To set the current tag to v0.0.1.

git checkout v0.0.1

**43. git tag -a v0.0.3 -m “version 0.0.3”**

To create a new tag.

git tag -a v0.0.3 -m “version 0.0.3”

**44. git push –tags**

To push the tags to the remote repository.

git push –tags

**45. git fetch**

To fetch down any changes from the global repository to the current repository.

git fetch

**46. git stash**

To move staged files to the stash area which is present in the staging area.

git stash

**47. git stash pop**

To get back the files that are present in the stash area.

git stash pop

**48. git stash clear**

To clear the stash folder.

git stash clear

**49. git rebase**

Three tasks are performed by git rebase

1. Move all changes to master which are not in origin/master to a temporary area.
2. Run all origin master commits.
3. Run all commits in the temporary area on top of our master one at a time, so it avoids merge commits.

git rebase

**50. git –version**

used to show the current version of Git

git –version