**Eseential Git Commands:**

1. Encrypted message has been decrypted by “$$” algorithmically after every letter with SSH.
2. Asymmetric Cryptography Algorithm-> 1. RSA(Rivest-Shamir-Adleman) & 2.DSA(Digital-Signature-Algorithm). Others-[Elliptic Curve Cryptography](https://www.ssl2buy.com/wiki/ecc-algorithm-to-enhance-security-with-better-key-strength) (ECC) or Elliptic Curve Digital Signature Algorithm (ECDSA) id\_rsa & id\_rsa.pub are the keys respective for private and public keys and ssh-keygen-t is the main algorithm to generate of ssh.
3. Command->git: to verify git is installed or not.
4. Command->git –version: for checking git version on desktop.
5. Command-> git help: for help with other commands.
6. Command->git config –list: to check configuration of git on desktop.
7. Command->git config –global user.name “….”: to enter username.
8. Command->git config –global user.email “….”: to enter email.
9. Command->cd “full-path of directory or folder”: to set up project folder.
10. Command->git init: to specify git in project folder.
11. Command->git status: to check folder status for repository.
12. Command->touch <file-name>: to create new file.
13. Command->ls: to check all files in project folder & repository.
14. Command->git add <file-name>: to add files from project folder to staged repository.
15. Command->git commit –m “your message”: to specify commit for entire snapshot.
16. Command->git log: to show all commits history.
17. Command->git diff: to check difference between working folder and last repository.
18. Command->git rm <file-name>: to remove a file form working directory.
19. Command->git ls -files: to check all files in directory.
20. Command->git rm –r <directory-or-subdirectoru-name>: to delete directory in repository
21. Command->git tag <tag-name>: to create tag name.
22. Command->git tag : to show tag names.
23. Command->git tag --list: to create tag name list.
24. Command->git tag -l “initial letter\*”: to show all starting tag with initial letter.
25. Command->git tag -a <new-tag-name> -m “message”: to create new tag with message for new commit.
26. Command->git show <tag-name>: to show all details of this particular tag.
27. Command->git tag --delete <tag-name>”: to delete a tag details.
28. Command-> vi <file-name>: to create file in staging area working or modifying directly.
29. Command-> git restore file-name: to restore file from staged are to project directory.
30. Command-> git reset HEAD <file-name>: to show untracked files.
31. Command-> git reset --hard HEAD :going back to HEAD.
32. Command-> git reset --hard HEAD^ :going back to the commit before HEAD.
33. Command-> git reset --hard HEAD~1 :equivalent to "^"
34. Command-> git reset --hard HEAD~2 :going back two commits before HEAD
35. Command-> cat <file-name>: to show file details.
36. Command-> git remote: to show origin on github.
37. Command-> git remote -v: to show origin links or urls github.
38. Command-> cd “full-path-directory-change-with-/-to-each-folder”->git push origin master: to push all files to github from git.
39. Command->ssh-keygen -t rsa –b 4096 –c ”your e-mail”: for ssh keygeneration.
40. Command->eval $(ssh-agent -s): to activate rsa.
41. Command->ssh-add username\_id\_rsa: activate ssh.
42. Command->clip < username\_Id\_rsa.pub: to create ssh in gthub.
43. Command->ssh –v: to make sure : Make sure you have an SSH client installed by running
44. Command->ls -a ~/.ssh: Look at keys already generated under the "~/.ssh" directory
45. Command->ssh-keygen: Generate a new key-pair
46. Command-> All details-

Add your newly generated SSH key to the "ssh-agent" 1. Make sure ssh-agent is in fact enabled :1. eval "$(ssh-agent - s)" ,2. Add the SSH key to the "ssh-agent" :1. ssh-add ~/.ssh/id\_rsa

1. Command-> cat ~/.ssh/id\_rsa.pub: COPY the public key content
2. Go to "Settings" --> SSH and GPG keys --> New SSH key. PASTE the public key content which was copied previously.
3. Raw-button helps to copy entire code of a particular file. Blame-button shows all the writers who modify the code of a particular file. History-button shows all history of commits.
4. The git commit is a process with command, used to create a snapshot of the staged changes along a timeline of a git projects history.
5. Issues-button describes issues of a file or all files for a particular repository with the helping of Label-button which describes the enhancement, urgency etc. with different colored symbols.
6. “.gitignore ” describes to avoid all the specific files of particular extensions with a subdirectory, named with command “build/ \*.txt or other extension”.Command->touch .gitignore.
7. Fork-button helps to copy separately another project or repository to own GITHUB profile to modify or to change anything.
8. Pull Request-Button helps to main author of a “forked” project and repository to find any changes of his/her projects.
9. Command-> git clone then paste the clone link with https link from github page with respective repository code option: to clone remote repository to local.
10. Command->git branch “put your branch name”: to create branch localy.
11. Branches allow you to develop features, fix bugs, or safely experiment with new ideas in a contained area of your repository to isolate development work without affecting other branches in the repository. A branch is essentially is a unique set of code changes with a unique name.
12. Command-> git checkout branch-name: to check another branch.
13. Command-> git branch: to check new branch.
14. Git workflow diagram: untracked by git->tracked by git(unchanged)->tracked by git(changed)->staged->committed.
15. SHA-I:”Any Number” is specified unique number for unique commit.
16. Command->git commit –a -m “your message”: to skipping staging area.
17. Command->git checkout –b new-branch-name: switch to branch feature.
18. Command->git branch new-branch-name: to create new branch feature.
19. Command->git checkout branch-name: to checkout new branch feature.
20. Command->git merge branch-name: to merge new branch with “master” branch.
21. Command->git branch –d branch-name: to delete a branch.
22. Command->git log:: to see entire commits list.
23. Command->git log -1: to check last commit.
24. Command->git commit –m “merged from <present commit> and resolved a conflict in <destination-file-name>”: to specify of merging different branches and files.
25. Conflict is considered as the decision of git with essential commands for changing content of a file in master or other new branch, merging of one or more different branches to master branch.
26. In merging process of two branches, always use the command-> git add file-name.
27. Create gitlab account with github or other options, setting your projects and solve the issues according to instructions on gitlab.
28. Generate ssh keys with project folder by gitbash with above commands then paste the key to gitlab project repository.
29. Command->git clone gitlab-project-details-link: to clone the projects from gitlab to gitbash.
30. Command->git remote: to access origin of master branch.
31. Command->git remote –v: to fetch and push project from gitlab.
32. Command-> git push origin master: to push origin of master branch for gitlab.
33. “SVN”-Sub Version is remote version control tool and it is a centralized VCS where as other category of VCS is decentralized or distributed VCS.
34. Command-> git status --short or -s: to short of details for modified or new created files in current branch. (??) And (m) symbols in front of files are describing untracked stage and modified stage of those files respectively.
35. Command-> git help status: to take help to show status of current branch entirely if everything is going right.
36. Go to this link to find other topics-https://git-scm.com/docs/git-status as well as serach also other page in google or similar search engine.
37. Command->git log –p or –patch: to show entire commits list with all files details.
38. Command->git log -p -3: to specify the last nth commits details according to put number like “3”.
39. Command->git log –since=nth-number.weeks/days:/months: to show week or day or month basis commits.
40. Command->git log –stat: to show entire commits details with modification, deletion, creation of files.
41. Command->git config –global alias.<alias-command(nickname of command)> <actual-command>: to create nickname or short name of commands.
42. Command->git config –global alias.unstage ‘rest HEAD --’: to create unstage command for files.
43. Command-> git log -1 HEAD: to show nth number of commit details.
44. Command-> git ls-files: to find all tracked files.
45. Command->ls: to show all files in current branch.
46. Command-> rm <file-name>: to remove specific file.
47. Command-> git ls-files: to show all files with .gitignore file.
48. Command-> rm \*.extension-name(like .txt): to remove files with specific extensions.
49. Command-> git rm file-name: to remove tracked file.
50. Command-> git clean –fx: to remove all untracked files with .gitignore file.
51. Command-> git clean –fd: to remove all directories.
52. Command-> git clean –d –n: to warm us what will be removed before deleting it.
53. Command->git clean –f: to remove all known files.
54. Command->git clean –x –f: to remove all ignored files and directories.
55. Command-> git clean –d –i: to show all interactive interface before deleting files.
56. Command->mv <file-old-name> <file-new-name>:to rename file name.
57. Command->git rm <file-name>: to remove specific file.

.