GIT and GITHUB → Architecture and commands Cheat Sheet

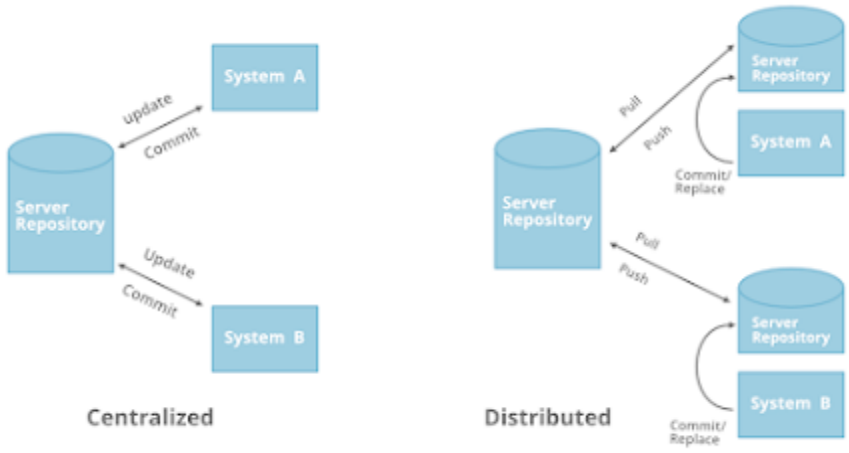
— — — — — — — -Introduction of git — — — — --- —

First of all git and github are two different thing github is a service which provide repository or folder for your code and git is a software configuration or source code management tool or to manage the different versions of your code

- GIT and GITHUB are two different things

➢ Source code management tools are of two types →

1. **Centralized version control System (CVCS)**
2. **Distributed version control System (DVCS)**



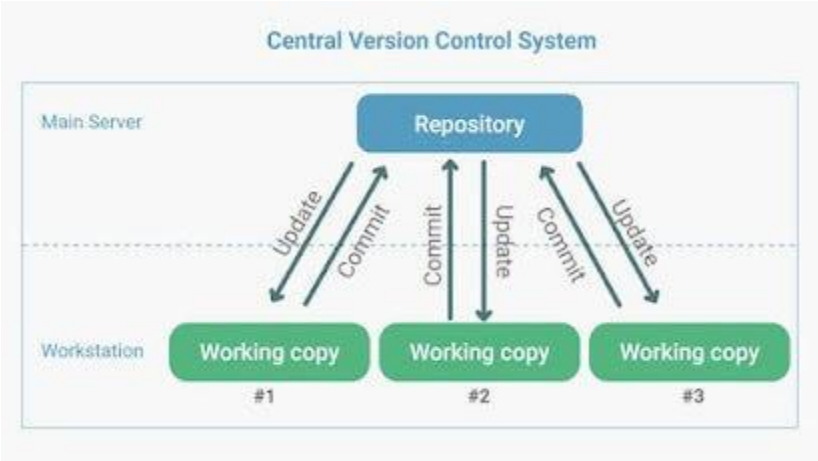
Problems with CVCS:-

1. There is no copy of code available

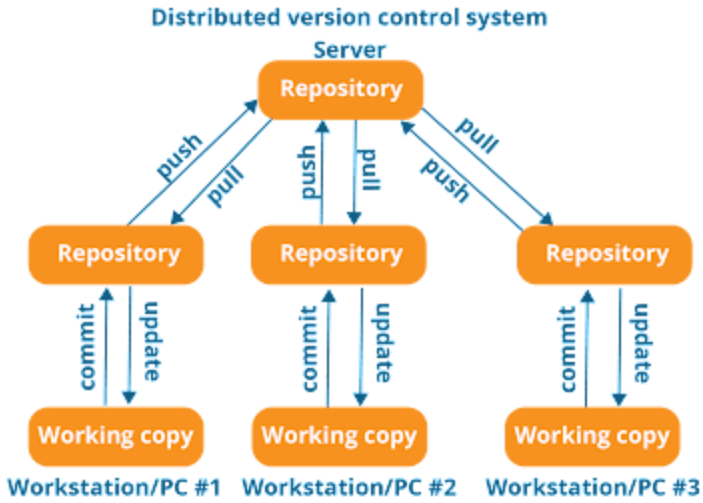
2. If repository being down than all collapsed

3. As there is no copy of code available and you need an internet connection all the time you try to connect to repository

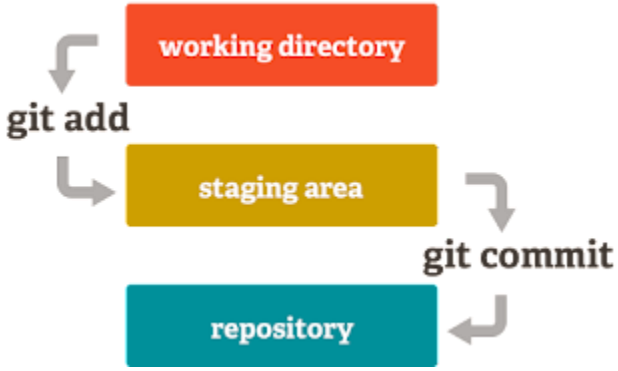
4. Since everything is centralized if control server gets failed you will lose your data



Git is used to overcome all these problems. Git is a dvcs tool which means the code written by you is in distributed form or there is a local repo is present in your device where you commit your code before push it on github which means if your github or central repository being down your code is available to work now your code is distributed or more copy of your code is available.



Architecture of Git -:



Three parts of git folder →

1. **Working directory** — — it is a part where all your code is present

2. **Staging area** — — it is that part where you can add file or code from working directory that is need to be committed or add to local repository

3. **Local repository** — — here you can add your final code which you can upload on central server or github

Before working — —

1. First of all we need to install git using " apt install git" command in linux you can also use it in windows

2. Next you will make a folder where you are working with git

3. Now run git init command to initialize a folder as git folder

4. The folder is now divided in 3 parts run ls -a for check

Commands to push your code on github from local System →

1. make a folder → mkdir <directory name>
2. git init → To initialize git personal directory
3. git --version
4. git config — global user.name "your name"
5. git config — global user. Email "your email"
6. git status → Now you are present in working directory make a file or add it to staging area using the command below

7. git add . → It adds all un tracked files which are in the PWD or present working path
8. git commit -m “your message” file name → To add your code in local repo
9. git commit -am filename or . (PWD)→To add and commit the any file by using single command. But here the thing is this command only applicable or used to the git tracked files and folders only
10. git log → Give info who is commit the code
11. git log --oneline --all –graph →To see the all Commit ids with in graphical view.
12. git show <commit id> → Information about the commit
13. git remote add origin <git repo url> → Where the code is uploaded/Creates the destination path
14. git push -u origin master → Then you need to enter your github id and password (PAT) to push the code on particular repo
15. git pull -u origin master → It pulls or cloned or copied all files and folders from remote repository master branch
16. git clone → To clone a repo
17. git add → To add the file from working directory/area to staging area

18. git commit → To add the file from staging area to Local Repository
19. git status → To see the Status
20. git restore --staged → To pull back file from staging area to working directory/area
21. git ls-files → to see the list of files tracked by git
22. git log → it Will show the log details of commits
23. git log --oneline → it Will show the log details of commits in short
24. git log --oneline --graph → it Will show the log details of commits in short and graphical way
25. git show <commit id> → it will show the details of commit id
26. git diff <file name>→ it will show the difference between committed and non-committed files
27. git restore <file name>→This will delete the recently added data of a file
28. git checkout <file name> →Undoing the local changes that have not been committed.
29. git revert commit id --no-edit →Undoing a specific commit (That has been pushed)

30. git reset --soft HEAD~1 or 2 (which commit data you want to edit that commit id number) →To undo or resetting the committed data.

31. git reset commit id →To undo or resetting the committed data directly

32. git reset --hard commit id →To undo or resetting the committed data and removing after created committed data along with its commit ids

33. git branch → This will show the list of branches and present working branch

34. git branch <branch name> → To create a new branch

35. git checkout or switch <branch name> → To Switch from one branch to another branch

36. git checkout -b <branch name> →To create new branch and switch directly into that created new branch

37. git merge <branch name> → To add the coded files or folders from one branch to another branch (it creates the extra commits or commit ids.)

38. git rebase <branch name> →To add the coded files or folders from one branch to another branch (it reduces the creation of extra or a greater number of commits or commit ids compare to merge)

39. git rebase --continue → To rebase or merge the conflicted files

40. git push --all → It pushes all files of all branches
41. git push origin <branch name> → It pushes all files of particular Branch
42. git fetch → it will only fetch the details of any one of the branch files
43. git pull → it will download the files of repo from Central GITHUB to LOCAL Server
44. git pull origin <branch name> → To pull a Particular Branch files
45. git init → To create a Local repo or reinitialize the existing repo
46. git branch –m <branch name> → To change the name of existing branch
47. git config --global user.name “enter your user name (configured username)” → To Create username of git
48. git config --global user.email “enter your email id (configured email id)” → To Create email id of git
49. git config --list → To list out the number of users of git
50. git clone -b branch name https url link of the repository →To clone or pull or download the particular branch coded files and folders
51. git commit -am filename or . (PWD) → To moving the git tracked files from working directory to Local repository (git add and git commit) with the single command

52. git branch -a → To see the complete details of branches
53. git stash → To hide the git tracked files
54. git stash -a → To hide git tracked and un tracked files
55. git stash list → To list out the hidden files in stash
56. git stash apply → To pull back or backup the hidden files from the git stash as a copy
57. git stash drop → To delete or remove the stashed or hidden files copies
58. git stash pop → To pull back or backup the hidden files from the git stash without copy
59. git stash apply stash@{0} or stash@{1} → To pull back or backup the particular hidden files from the git stash as a copy based up on stash que list (stash@{0} or stash@{1})
60. git stash pop stash@{0} or stash@{1} → To pull back or backup the particular hidden files from the git stash without copy based up on stash que list (stash@{0} or stash@{1})