# Version control system

* It is a management software used to keep a track of the changes made in any kind of project
* It makes recovery of old file/version possible
* Rollback of previous versions
* Informs us about what and when changes were made and by whom.

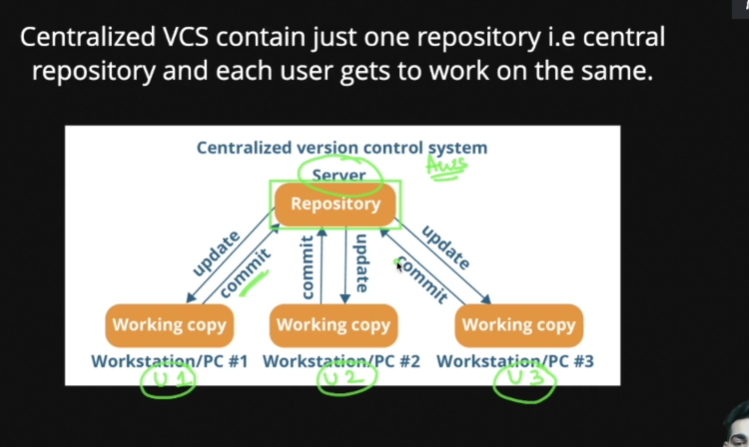
History of VCS

Local VCS

* Changes are stored in database along with the timestamp(date and time)
* Codes in local system
* Your project can be lost when your hard disk is corrupted

Centralised VCS

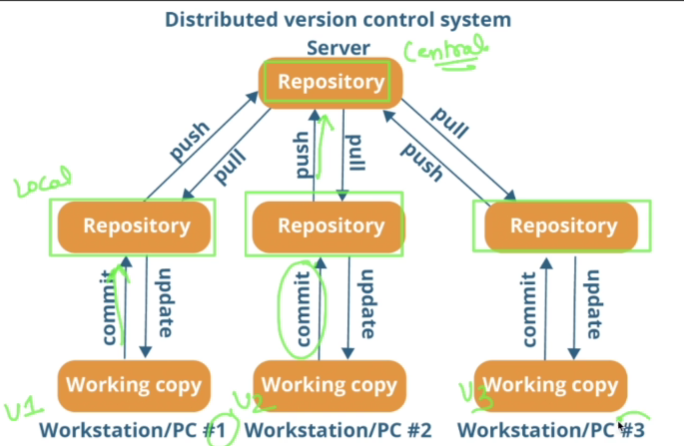
* Repository is a directory where your project resides.
* It can be in local system or remote directory on a server



* Cons- if the central repo goes down or if the hard disk of server gets corrupted.

Distributed VCS

* Contains multiple repo
* Each user has their own repo and there is a central repo where the final code resides



* Full back up is provided.
* Git is a example of distributed VCS

How git is created?

* Linus torvalds created it in 2005.
* They created git to maintain the versions for Linux.

What is git?

* Git is a free open source distributed VCS designed to handle everything with speed and efficiency
* Git stores snapshots of your projects(not differences).
* It actually stores that you deleted/updated but what changes is not present in snapshots
* If you want to know what changes you made then you have to roll back to that version.
* You can easily collaborate.
* Storing versions is easy.
* Analyse the code changes.
* Almost everything is locally; only at the time of push/pull we need internet connection.
* Non linear/ branching method
* Secure and integrity.(MD-5 algorithm is used)
* Git id written in C and it is very easily understand and interpreted by the computer.

Git bash commands

Command line tool

* doing all the stuff like copying file, deleting tool, creating tool, etc. using commands
* Pwd- tells us where actually we are.(print working directory), gives the path
* cd Desktop – changes your directory
* ls- lists all the files and folders
* ls –l used to list all the files with details like read write access and the owner and name
* ls –a tells us all the hidden folder as well.
* Ls –al gives us all the folders(hidden as well ) in the form of lists
* clear- used to clear all the written commands in the git bash
* git --version gives us the version of git we have
* git init is used to initialize the git there
* git config –global user.name “Aakanksha Jha”
* git config –global user.email “[jha.aakanksha111@gmail.com](mailto:jha.aakanksha111@gmail.com)”
* git config –list
* . refers to the current directory
* .. refers to parent directory
* Cd or cd ~ takes you directly to the home
* / refers to roots
* mkdir Name refers to creating new folder
* mkdir f1 f2 f3 will create three new folders
* mkdir –p f4/f5 it will create a new folder and inside that
* mkdir f2/f6 will create a new folder f6 under the existing f2 folder.
* Touch file.txt will create a new file.
* Cd .. will go to parent directory
* open file.txt opens the file
* rm file.txt used to delete file
* rm f2/file.txt to remove a file inside the folder
* rmdir f2 used to delete directory/folder which is empty
* rm ~R f2 use to delete folder/directory which is non empty (R stand for recursively)
* cp script.js f1/script.js is used to copy script to folder f2
* cp –R f4 new is used to copy f4 folder and make a new folder named as new in the same directory.
* mv index.html f1/index.html is used to move the file
* mv index.html about.html is used to rename but actually we are just moving it to the same directory with new name.

git commands

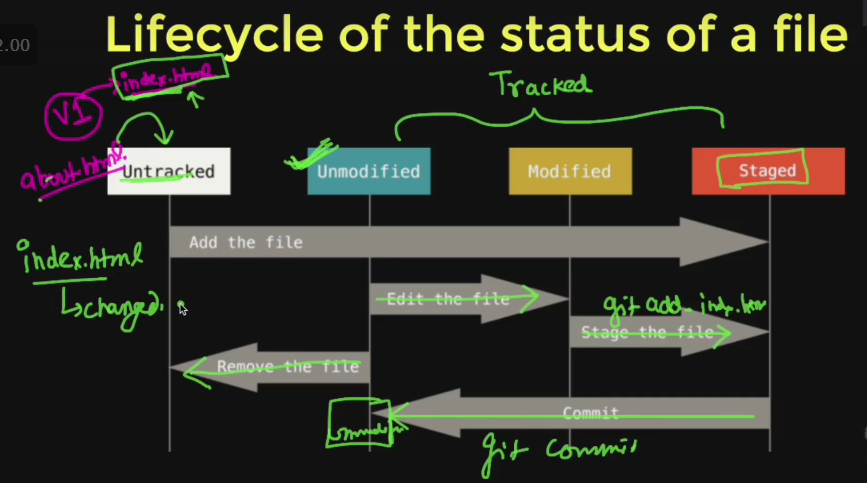
* **creating git repo- initialize or clone repo**
* **git init** is used to create a git repo
* **git status** is used to track the progress and changes.
* **git add index.html** to add all files
* **git add .** will add all the files in the current directory
* **git commit –m “Your commit message”** is used to commit the changes and make the file available for creating a version.(here –m is a flag which stands for message).
* **git log** is used to see the history of commits
* **git log –oneline** is used to get history with one liner details.
* Press q to exit from git log
* **rm –rf .git** is used to delete git repo (rm stands for remove, -r stands for recursively, and f stands for forcefully)
* **git commit –a –m “commit message**” will help you to add and commit together means skipping the staging area.(but initially when we create new file that time we need to do both things separately)
* **git diff** is used to know what is the actually difference in the two versions(the difference is between the version not added and existing version, so always use git diff before using add otherwise no changes will be observed)
* **git diff –staged** will tell you the difference you are going to commit, this is done even after adding files.
* **.gitignore** is file which contains the name of all files whose track you don’t want to keep)

Touch .gitignore

Then open the file and write all the name of the file you want to ignore

If you have many files having same extension and you want to ignore them all then use \*.log to ignore all files having extension .log

* To ignore folders write abc/ in the .gitignore file and the git won’t keep track of that file.
* **git rm -f abc/random.txt** will remove the file and it will be known to git as well otherwise you have to add and commit the changes you brought.(f for forcefully remove).
* **git mv index.html home.html** to rename /move the files using git and automatically stage them.
* **git rm –cached .DS\_store** is used when let say your git is already tracking the file but after that you added its name in .gitignore so you need to explicitly ignore or remove that file.
* **git restore –staged contact.html** to unstage the file
* **git checkout -- filename.txt** restore the previous version.
* **git checkout –f** is used to move to the previous commit losing all the newly modified files.
* **git config –global alias.st status** is used to create a short form like making a short form st for status
* **git config –global alias.unstage ‘restore –unstage** –‘ is replacement for long unstaging command ,usage is like git unstage file.txt



Branching

* Head is present at the last commit.
* **git branch** gives name of all the branches
* **git branch –v** gives the last commit /recent commit
* **git branch branch\_name** is used to create a new branch
* **git checkout branch\_name** is used to move to other branch.
* **git checkout –b branchname** is used to create and switch to new branch simultaneously.
* If we checkout without adding and commiting things in one branch to other then we’ll lose code.
* If we check the log then we can see one branch log at a time
* **git log --oneline --graph --all** is used to see all branch log at the same time
* **git branch –d branchname** is used to delete merge branch.
* **git branch –D branchname** is used to delete unmerged branch.
* We can’t delete the branch we are working currently on.

Merging