**Git is a version Control System** –

* Easily recover file
* Who introduced an issue and when
* Rollback to previously working state

**Histroy of Version Control System –** to overcome problem of keep track file we made

* **Local VCS** : Using DB to keep track file
* cons : fear of losing file if system fail and difficulty in collaboration of contributors
* Pros : keep track file and rollback
* **Centralized VCS** : A central Server to store and track file of different system
* Cons : Chance of server failure
* **Distributed VCS** : An enhanced Centralized VCS . Even in case of failure of Central Server contributors will have track of file with them

**Git** is Distributed VCS created by Linus Torvald. **Github** is hosting service for git repo.

**Features of Git** –

* Screeshots of each commit
* Almost every operation is local . Then we update in Centralized server according to our need.
* Provides integrity : Checksome can be used to check wether file has been disturbed or not.

**Git for Windows** – When we install we get

* Command Line Tool
* Git Bash(A Terminal like cmd or powershell)

**Way of Use**

* **Using Command line** to work in a directory . It takes linux based command line.
* **Gui**

**Setting name and email and some command -**

// it is - - before global

git status //tells wether you are in a git repo or not . Detects wether any changes are made in any file etc.

git config --global user.name “Kaivalya”

git config --global user.email “[kaivalyamanit@gmail.com](mailto:kaivalyamanit@gmail.com)”

git config --list //to see list of user and details

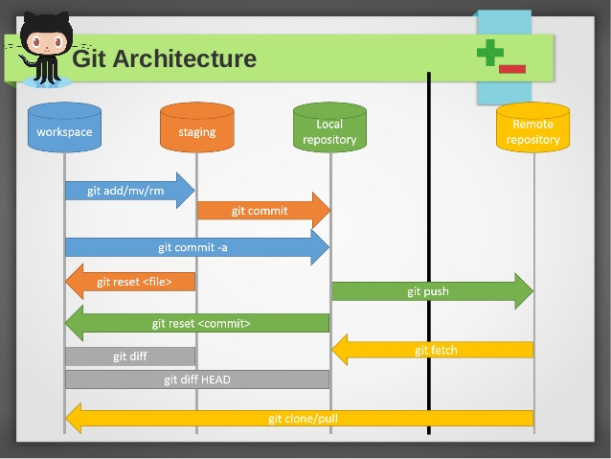
git config --global core.editor atom //to set editor

git config user.name //to see name

git config user.email // to see email

git config core.editor //to see editor

**Git Architecture –**



Git repo or git directory – different version of files are stored in .git folder in compressed form and it shows required version according to our command

**Lets Start –**

git init //initialize a git folder or repo . It can reinitialize the already exsisting git repo

git add --a //add all the file in that folder to staging area

git add . //add all the file in that repo to staging area

git commit -m “Initial Commit” //commit the file of staging area and displays the message in the quote

git commit –a –m //directly commit all the \*tarcked\* file without going to staging area

git commit - - amend // to change just previous commit

git log //to see what what has been commited like – author , hash , date , message of commit

git log –p //It shows git log along with diff

git log –p –number //will show “number” commits with above feature

git log - -stat // will show stats also

git log - - pretty=oneline // just write and see effect

git log - - pretty=short // just write and see effect

git log - - pretty=full //just write ans see effect

git log - -since=n.days //what has happened in last n days

git log - -since=n.weeks/months/years

git log - -pretty=format : “format and code” //just see documentation for various abbr used and you can

.//give desired format style to see the log in that way

git add <filename> //add the specified file name to statging area

git clone <link> <”name you want”> //To clone a git repo and give it a name you want

git diff // compares staging area with working repo

git diff –staged //compares staging area with just previous commit

git rm file\_or\_foldername //we can do deletion manually but then for changes we will have to stage

// and commit ourself. But this removes a file and move changes to staging area.

git mv old\_name new\_name //rename and move to staging area . mv is linux move commandu

git rm –cached file-name //Now git will not track the specified file and move it to staging area

git restore –staged filename //to unstage the file

git checkout –filename //to restore previous commit

* git ignores empty folders. It consider empty folder as nothing
* anything written inside .gitignore will be ignored by git
* Only untracked file will be ignored
* To ignore currently tracked file we have to ignore it explicitly . First untrack it and then move to gitignore
* .gitignore is like pattern searching . So explore it as you learn more
* ex1) t.log // will ignore t.log file
* ex2) \*.log //will ignore all .log file anywhere inside repo
* ex3) dir1/ //ignore all the dir1 folder
* ex4) /dir1/ //ignore only outer dir1 folder
* ex5) games/files/dir1 //ignore dir1 folder inside of game/files only.
* in this way explore more

***File Life Cycle***



I have a folder “Document” with 3 files – 1.txt , 2.txt , 3.txt . I made “Document” a git repo then all files in the folder were untracked . Now we add the files to staging area and tracking starts.

git add –a

So 1.txt ,2.txt ,3.txt are in staging area . While they are still in staging area I modify 2.txt then git will recognize the changes but it will not include it in 2.txt of staging area . So 2.txt is in modified state as well as in stagin area.

From staging area when we commit anything then that becomes unmodified . When we do any modification then it go to modified state and so on life cycle continues.