# ***( Git {local,desktop work})***

## # 1- Extensions for git in vs code

* Git Blame by wade anderson
* Git autoconfig by skykoserhiy
* Github theme by github.com

## # 2- Basic setup for first time setup git user in vsCode(skip this step if you already done it)

* Git config - -global user.name “Syed Muhamamd Umar” => (set vs code username)
* Git config - -global user.email “[ssyedmuhammadumar.developer@gmail.com](mailto:ssyedmuhammadumar.developer@gmail.com)” => (for set vs code user email)
* Git config - - global - e => (to edit the username and email)

## # Note: Git for Local management (all history,commit,add, manage locally, it don't have any type of link with internet/github )

* **Folder Structure**

.Parent folder

.Client folder

.Server folder

Scenario1: (if git init in parent folder so both client and server folder also tracked)

Scenario2: (if git init in client folder or server folder so each folder will be tracked at one)

Resolve: (let's suppose if you initialize client but after that boss want that to add server to or whole parent so how do we do that? )

## # 3- Initialization and local history (for local/desktop, there is no relation with internet and github)

Setps:

* 1- Git init => (tell the git to track this specific folder)
* 2- Git add FileOne => ( for specific one file & git know there is a FileOne in parent folder)
* 3- Gig add . => (for all files and folder in the parent folder )
* 4- git commit -m “changes one done, 1 may ka hisab”

git commit -m “changes two done, 5 jan ka hisab” => (store each commit with customize message )

* Note: git log - -oneLine (check all commit history and what we did previously with unique id of each and current status

## # 5 - Advance (GitIgnore)

* 1- create file with name of in parent folder is, .gitignore => (to add those files which we don't want to tracked by git)
* 2- write down those files names in git ignore file => (write the name of al those sensitive files, which we don't want to let the tracked by git)
* 3- .gitignore file > write( package.json, .env, more)

## # 6- Checking/Status (only for local/desktop, there is no link with internet and github)

* git status -s => (check stages of the all files like, which one is tracked, which is committed, and which untracked, tell that before and after commit history )
* git log => (check all commit history from beginning to today)
* git log - -oneline =>
* Git log - - oneline - - graph =>

## # 7- Branchind (Super Advance)

* Create copy of the main code that is called branching

Flow of the branching:

* I write 32 line of code in our static website project file
* I want that hassan create header so i give the 32 lines of codes file, copy to hassan then hassan add 100 lines of header code , so i say hassan you code is fine lets give me back that so i do merge, then now our code become 132 lines of code and then i say hassan that copy from yours side
* And the flow goes on, one copy send to hassan if hassan feature done so i merge back to original project , then same one copy of the same project from the original send to ali to add new feature if this feature done from the ali side so i merge back to our original code and
* Original 30line code > hassan 30 + 100 > merge ⇒ Original 130
* Original 130 lines of code > ali 40 + 130 > merge ⇒ original 170
* Now Original is 170