*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*GIT and GIT HUB\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

**/\* GIT \*/**

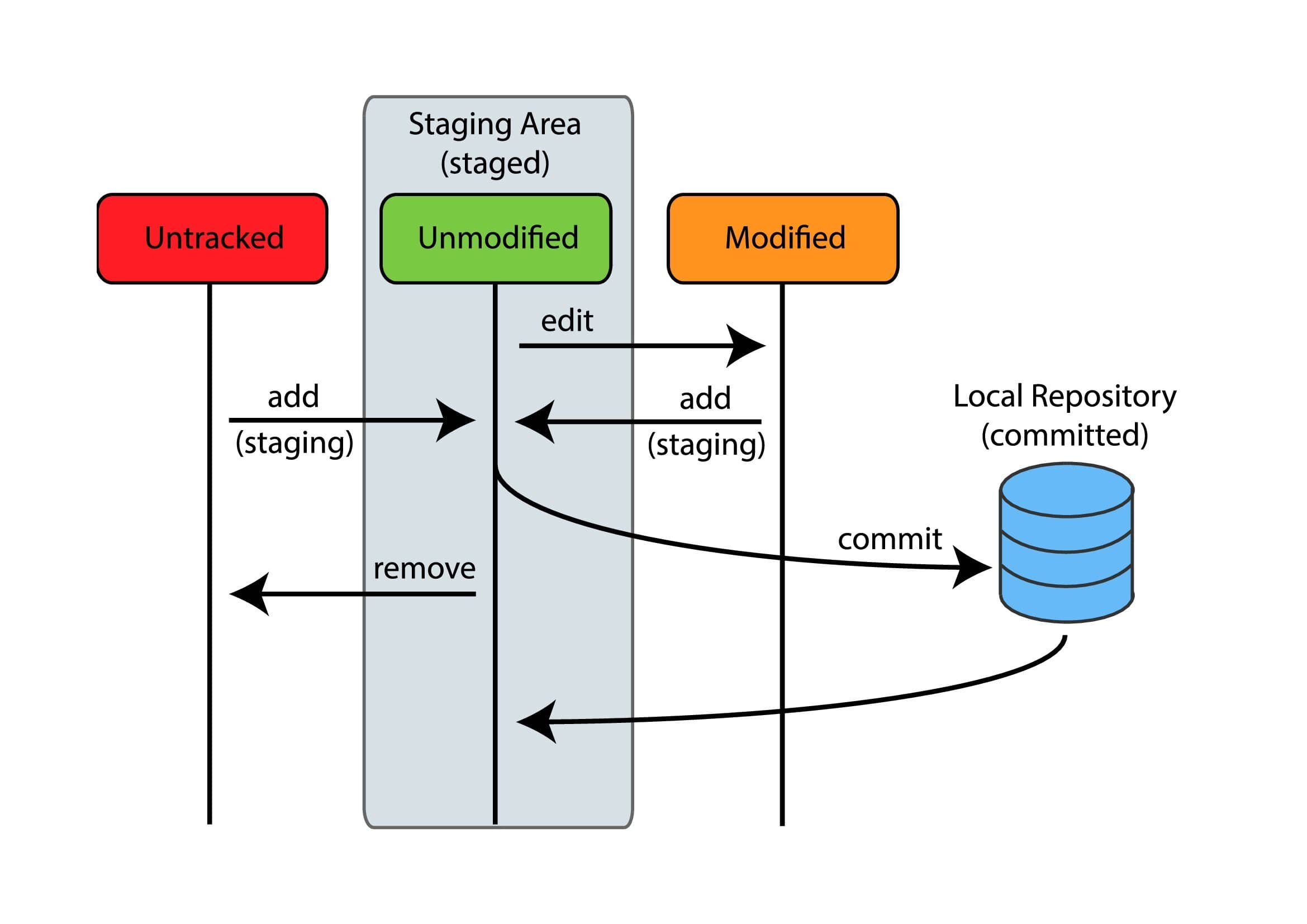
//Git is a version control tool (software) to track the changes in the source code.

**/\* GIT HUB \*/**

//GitHub is a web-based cloud service to host your source code (Git repositories). It is a centralized system.

//Git doesn’t require GitHub but GitHub requires Git.

// working directory---(git add) --->staging area----(git commit) ---->commit history-----(git push) ----->finally store into server



**/\* git setup \*/**

git config --global user.name "naveen"

git config --global user.email "naveenaravapalli09@gmail.com"

git config --global colour.ui auto

git config -l //to know info--

**/\* git init \*/**

/\* --only for new projects we use this step \*/

//create directory.

//open the directory using terminal.

//now we need to Initialized Git repository.

git init . //it will create. git folder (hidden one)

**/\* git add \*/**

//create some files

git status //to see status

git add <filename> //to add the files

git rm --cached <file name> //to remove file

git add . //it add all files in current dir.

git add -A // it will add all files if you are in another directory in same directory

**/\* git alias \*/**

// we can create commands shortcut for own

got config --global alias.<own cmd> "original cmd"//example -- git config --global alias.allcommits "log --oneline --graph --all"

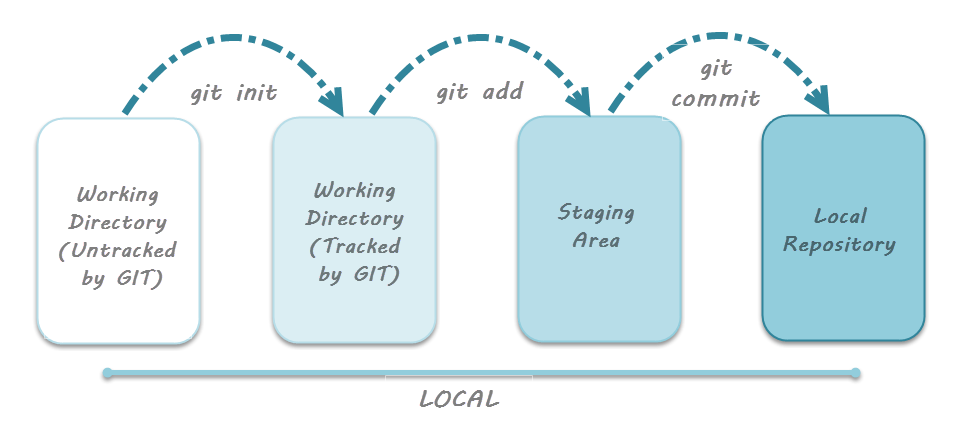
**/\* git commits \*/**

git commit //for direct commit

git commit -v //verbose it will give every change

git commit -m "any mesage type here" //single line message

git commit -am "any message"//it will do adding and commit at atime



//to see changes

**/\* git log \*/**

//copy commit hash (random numbers)

git show <past hash here> //will see the changes

git diff //give the differences current one and committed one

git diff --staged // to give diff of staged files

//if any changes done will see the changes and will follow the git add and git commit steps

git restore <file name>// to discard the changes

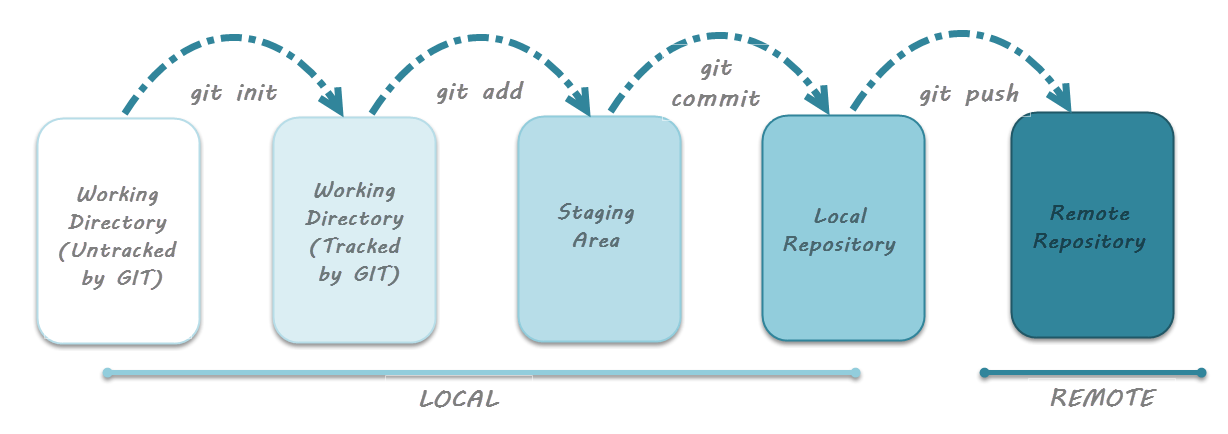
git commit --amend -m "message type here" //(--amend) it will replace previous message

**/\* push an existing repository from the command line \*/**

git remote add origin git@github.com:naveenaravapalli/learnin-git.git //to connect my git repo

git branch -M main //for changing branch master to main

git push -u origin main



//here we get error we first setup ssh our GitHub account

**/\* generate ssh key \*/**

//Goto git hub profile select SSH and GPG keys and follow the instructions to generate SSH keys link. there you follow commands based on your machine(windows/Linux/mac).

//here i am giving link for ssh key generation

https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent

//after ssh key generation successfully then try to push the code.

git push -u origin main //for next time you can use only git push

//now you can see local files in your GitHub account

**/\* git pull \*/**

// you will get files and changes from git to your local machine

git pull //

**/\* git clone \*/**

// git clone <ssh address> // to get the dir from git hub to local.

**/\* Branches \*/**



//here we have so many commits for each commit it will generate hash (random numbers (will see using git log command)) all these commits are inside what we called branch.

//while pushing files into git will see the main branch and in internet master branch, we changed master to main branch (git branch -M main) both same. these are default branches.

// A branch represents an independent line of development.

// best practice was created separate branch then do couple of commits and merge to main branch.

**/\* working with branches \*/**

git branch //to know working branch

git branch -r //for remote branches

git branch -a //all branches

git branch <new branch name> //to create new branch

git checkout <branch name> //to switch the branch

git checkout -b <branch name> // at a time create branch and switching to newly created branch

git branch -d <branch name> // to delete the branch (you should not at delete branch to delete)

//newly created branch files will not reflect too main

// for that we need to merge the branches

git merge <branch name> //we should at main branch then merge the branch to main after we can delete

git branch -d <branch name> //to delete after merge

// Better do pull request in git hub

// for that select pull request option on git hub then select pull request then pic the branch give the description for changes and pull request.

//we have reviewers to pic them to review the changes. Like some settings are there you can set according to work.

// after review click merge pull request. we can configure through settings.

//after delete the branch.

// now Goto main branch you will see the merged branch files.

// Goto local machine you can see the changes

//first, we full

git pull

git log --oneline //to see the changes better way

**/\* Dealing with conflicts \*/**

//if you avoid conflicts every time you first pull then do the changes then push.

// if in case conflicts are raised you first solve then then push.

// you can do manually or using terminal.

**/\* git rebase \*/**

// if you working on one branch and you master done changes so we need to take those changes and add our changes on top of it this is what rebase allow to do.

git pull --rebase origin main // or use this command git pull -r origin main

//you will see conflicts you should resolve. at a time, you can solve one conflict so follow below commands and slove all conflicts.

git add .

git rebase --continue

// after solving the conflicts, you should push by forcefully

git push -f

// now you can see the main branch changes to your branch then you can merge by the pull request. after you can delete branch.

**/\* git stashing \*/**

// it just delete the updated data means move that data to recycle bin

git stash or git stash <file name>

git stash list //it will all stashes and it will assign numbering also like stash@{1}

git stash apply //it apply latest stash that means it will work like stack(fifo)

git stash drop //it will delete stash data from bin

git stash apply <stash value (example -- stash@{1})> //it will give particular stash and apply

git stash pop// it will apply and drop the stash

git stash -u // it will stash the untracked (before add all files are untracked) files also

**/\* creating branch from stashing \*/**

// that means it will create branch where we stash was done

git stash branch <branch name>//it will all stash apply to here

**/\* git clean \*/**

// be careful we can nor recover deleted files

git clean -f// only untracked files was deleted

git clean -f -d // to delete directory's

git clean -f -d -x// it will delete .gitignore files also

git clean -f -n// it will give to be deleted files

**/\* tagging \*/**

//to mark the particular commit

git tag -l // to list all tags

git tag <tag name > // to create tag for latest commit.

// it was lite weight tag it does not any description.

// we can give description also using annoyed tag.

git tag -a <tag name> -m "description related to tag"

//if you want to see the changes of different tags you can follow below commands

git diff <tag name1> <tag name2>

git tag -d <tag name> // to delete the tag

// we can give tags to old commits also.

git tag -a <tag name> <commit id> -m "description anything"

// you can change tag place also by deleting and assign new tag place.

//you can directly update tag forcefully by using below cmd.

git tag -a <tag name> -f <commit id> -m "description anything"

//we can push the particular tag also

git push origin <branch> <tag name>

git push origin <branch> --tags // it will push all tags to remote location

git push -d origin <tag name> // to delete tags on remote location