GIT MODULE

* ls **=** Use to list down the content.

ls –l **=** for detail output

ls –al **=** for hidden file

ls –rl **=** for reverse order,

ls –lt **=** for time

ls –lh **=** for human ridable.

* touch **||** vi **||** cat **||** echo **||** nano **=** Use to create file.
* git init(initialization) **=** To create a local git repository.
* git status **=** It’s simply tell us git aware your file or not.

git status –v **=** It will give detail output.

git status –s **=** It will give short information.

* git add **=** Use to bring a file from non-staging area to staging area.

git add . **=** Use to add all non-empty file into staging area.

* git commit **=** Use to track the file.

Git commit –m **=** Commit without opening the editor.

* git log **=** To see the commits.
* rm –rf **=** Use to remove the local git repository.
* git init --initial-branch **=** Use to create new branch name as per our choice.
* git rm --cached **=** Use to bring the file from staging area to non-staging area.

(undo of git add)

* git commit –a –m **=** If we modified in track file then we can directly commit the file this way.
* git commit --amend **=** Use to add commit massage in the existing commit without creating new commit id.
* git commit --amend --no-edit **=** Use to pick up your last commit massage.
* git commit –s –m **=** Use to commit with signed-off massage.
* git commit --allow-empty -m **=** Use to make empty/dummy commit.
* git log –n 2 **=** Use to see last 2 commits only.
* git log --pretty=short/one-line **=** Use to see the logs in short.
* git log --pretty=full/fuller **=** Use to see the logs in detail form.
* git log –p **=** Use to see detail information of each log.

* git log --pretty=format: % h **=** Use to see only hash code values.

% ae **=** Use to see author email id.

% h% s **=** Use to see hash code as well as massage also.

% an **=** Use to see author name.

* git log --since = “1week ago” **=** All logs happen in last 1 week.

“Yesterday” **=** All logs happen in yesterday.

“1month ago” **=** All logs happen in last 1 month ago.

“2024-04-10” --until= “2024-05-10” = All logs happen in this date.

* git log --author = “yash” **=** Use to see commit done by the specific author.
* git log --grep = “modified” **=** Use to search the file which include the word modified.
* git reset --soft (paste commit id) **=** Changes will have brought into the staging area
* git diff --staged **=** Use to see the changes happen in logs.
* git reset --mixed (paste commit id) **=** Changes will have brought into the non-staging area.
* git reset --hard (paste commit id) **=** Changes will have been deleted permanently.
* git revert (paste commit id) **=** Commit that happen by mistake it will undo the commit by creating new commit Without affecting the history of a branch.
* git checkout (paste commit id) **=** To see the changes of reverting.
* git branch **=** Gives the information about all the local branches present in the folder.
* git branch **‘main’ =** To create a new branch main.
* git checkout (branch name) **=** Use to move from one branch to another.
* git branch –D (branch name) **=** Use the delete the branch.
* git switch **=** Use to switch from one branch to another.
* git merge **=** Use to share the commit from one branch to another.
* git cherry --pick (paste commit id) **=** Use to pick individual commit from one branch and you can take those commit and it is going to be kept in the respective branches.
  + Following Commands are used to push our project on the git lab.
* git remote **=** It will give you the information of the remote repository.
* git remote –v **=** It will give you detail output.
* git remote add origin (paste url of remote repository) **=** Use to add remote repository location to our local git repository.
* git push –u origin master **=** Use to create a master branch on our remote repository and then push the code of our local repository to the remote repository.
* git branch **=** It will show you only local branches.
* git branch –r **=** It will show you only remote tracking branches.
* git clone (paste the url of project) **=** Whenever we have to work on any other readymade project at that time git clone use to download those project into our local repository and then we can work on it.
* git fetch **=** Use to download the changes from your remote repository to local repository.
* git pull **=** It is nothing but the combination of the two commands Git fetch & Git merge. (Whenever remote repository having changes and to add those changes to our local repository at that time git pull command take in picture.)
* git fetch –prune **=** If we delete remote tracking branch on the github then we have to delete this branch from the local also at that situation this command will do this work.
* git fetch **=** Use to update the changes of remote tracking branches.
* git stash **=** If we have done some changes in some files then we don’t want to commit it and in that situation if we want to change the branch at that time git stash command will use.
* **NOTE –**
* If the name of any file/folder starts with (**.**) then it is hidden file/folder.
* File name is showing in red colour = untracked file (i.e. - Git is not aware about this file)
* File name is showing in green colour = tracked file (i.e. - Git is aware about the content of the file)
* Whenever you create a local git repo then by default branch name:(**Master**) will created.
* Git does not track empty folder.
* When Head is pointing towards the 2 branches then the both branches are in sync.
* If two persons are doing changes in the same file afterword’s at time of merging there is an issue will create of merging conflict, to resolve this issue open the file and remove the stuff which we are not added, that’s it. (Merge conflict solve only once)
* Origin = Is the name given to your remote repository.