GITHUB

GitHub is a website which is used for storing folders(repo) and manage them using **git** git is a tool which is like version control system that track changes and maintain record and save changes y commit option... (commit means save changes or taking screenshot of that moment so that a record can be maintained and used whenever required)

readme is a file which contain details like title (extension is readme.md which means mark down making changes in that can be done(using edit pencil button on readme after creating repo) like normal html then save commit and can add name at each change (A commit message is a short description of what changes you made in a commit.) (No. Committing does not automatically rename or change file extensions.))

GITBASH

now for windows install gitbash(Git Bash is a terminal (command-line tool) that lets you run Git commands on Windows.) then configure it using username and email of GitHub then now accessing GitHub using git in vscode terminal(git config --global user.name "Your Name", git config --global user.email "you@example.com" already done in my system)this is for maintaining git commit history with author and mail and can make mail pvt .. this config is global same for each repo but use only when want to change username and mail for separate repo then use same code for config as above but with diff username and mail that u want . config allows git how to behave as u want like commit changes like name and mail etc

1. What Git Bash is

Git Bash is a terminal program for Windows that gives you a Bash-like shell (Linux-style terminal) and includes Git commands.

When you install Git on Windows, Git Bash usually comes with it.

On Mac/Linux, Git is already accessible via the default terminal, so Git Bash is mainly for Windows.

2. What Git Bash lets you do

Run Git commands (git commit, git push, git config, etc.)

Run Linux-style terminal commands (ls, cd, mkdir) on Windows

Basically, it acts like a Linux terminal with Git installed

3. Running Git anywhere

After installing Git:

You can use Git Bash

You can also use Command Prompt (Windows) or Terminal (Mac/Linux)

Even VS Code integrated terminal works because it uses the Git installed on your system

In other words: Git Bash is just one way to run Git, but any terminal that recognizes Git commands works.

4. Key point

Tool Purpose

Git Bash Terminal with Git commands, Linux-like shell on Windows

VS Code Terminal Uses system-installed Git, can run Git commands inside VS Code

Command Prompt / Terminal System terminal, can run Git commands if Git is installed

💡 **Analogy:**

Git = the engine that manages code

Git Bash = a car you use to drive the engine on Windows

VS Code terminal = another car that can also drive the same engine

Exactly ✅ — once Git is installed (including Git Bash), configuration can be done anywhere a terminal can run Git commands

1. Where you can do Git configuration

You can run global or local Git configuration commands in any terminal that recognizes Git:

Git Bash (Windows) → the most common way for beginners

Command Prompt (Windows) → if Git is added to your PATH

PowerShell (Windows) → same as above

Terminal (Mac/Linux) → Git comes pre-installed on most systems

VS Code integrated terminal → it just uses the Git installed on your system

git config user.name does NOT have to be the same as your GitHub username.(user.name :- This is the name shown on your commits. It can be your real name, nickname, or anything you want.)

name.

3. Types of configuration

1. Global configuration

Applies to all repositories on your computer

Example:

git config --global user.name "Your Name"

git config --global user.email "your_email@example.com" ... this is once done for me not again needed for each repo but if u want then can read it is done once in gitbash terminal or

can do in vscode termial also.

2. Local (per-repo) configuration (it overwrites the global one)
Applies to only one repository

Example:
git config user.name "author name"
git config user.email "yournamecustom_email@example.com"

3. System configuration
Applies to all users on the computer (rarely used)

1. What is Configuration in Git
Configuration is like telling Git how you want it to behave.
It's settings you define so Git knows things like:
Your name and email (so commits are properly labeled)
Default editor for writing commit messages
Merge or push behavior
Colors for Git output, etc.
Think of it as setting up Git to recognize "you" and your preferences.

2. Why configuration is needed
Every commit in Git records who made it (name + email).
Without configuration, Git doesn't know your identity and may reject commits.
Configuration also makes Git work the way you want instead of using defaults.

Git = a notebook
Commit = a page you write
Configuration = signing the notebook and setting your preferences (so Git knows whose notebook it is and how to organize it)

There are two files 1. local - our system and 2. remote means which is on GitHub in form of repo

1st command of **git is clone** of remote files using
open vscode
then in terminal write (git clone httpslink) from remote repo code section (always do cloning through https) -- clone means creating a copy of repo on our system and can access all files of that repo like readme also

cd means change directory(folder) from main folder to that which is opened we got into our new folder that is being cloned also then clear and we got into our new folder that is being cloned

enter ls in terminal for getting list of all files in that repo
ls -Force = shows hidden files in terminal

all the folder which is tracked or used by git there is a .git file/folder always in that repo
.git means “this folder is a Git repository.” (.git denotes a Git repository; when connected to GitHub, it can be hosted as a GitHub repository.` 📁 Normal folder → ❌ not a repo 📁 Folder with .git → ✅ Git repository)

- 🔹 Types of Git repositories

1️⃣Local Git repository
Exists on your system,Created using git init,Works offline

2️⃣Remote Git repository
Hosted online (GitHub, GitLab, Bitbucket),Used for collaboration,Synced using push and pull

- 🔹 Why do we use Git repositories?

✅ Version control, ✅ Track who changed what, ✅ Work in teams safely, ✅ Backup of code, ✅ Experiment using branches

A Git repository is a version-controlled project folder tracked by Git.(Git maintains a hidden directory called .git to store the complete history and versions of the project.Git functions include add (stage changes), commit (save version), status (check state), push (send to GitHub), and pull (get updates).)

GitHub is an online platform that hosts Git repositories and helps developers store, share, and collaborate on code.

Git is the tool; GitHub is the platform that uses Git.(They work together but are not the same thing.)
Relationship in simple terms :-

Git manages versions locally on your computer
GitHub stores those Git repositories online
GitHub cannot work without Git
Git can work without GitHub

How they connect (step-by-step) (this is process of making local file to git then connect to gthub)
You create a project using Git (git init)
Git tracks changes and creates commits locally
You create a repository on GitHub
You connect local Git to GitHub using: git remote add origin <url>
You upload code using:git push
Others download changes using: git pull

vice versa process
When a repository is created first on GitHub, it is connected locally using git clone, which creates the project folder with a .git directory, and changes are managed using git add, git commit, git push, and git pull.

Key difference between the two flows
🅰️ **Local first → GitHub later**
Use: git init
Manually connect using git remote add

🅱️ **GitHub first → Local later**
Use: git clone
Connection is automatic

✅ Yes, every GitHub repository is a Git repository (there **is .git folder** in every repo of GitHub but cant see on GitHub directly for that we need to clone in local system) Because GitHub is mainly for hosting and collaboration, not for full development work so need to connect with local environment.

git status in terminal for showing status of code in repo

there are 4 status 1. **unmodified or unchanged**- which means nothing is modified or nothing need to commit
2**. modified** means its when u make some changes in files of repo in vs code like in readme file then show modified then for that there are two process 1. add that changes and then commit that changes..
and ready for commited its comes in the category of staged then make commit and then files status become unchanged or unmodified.
files are not tracked by git yet (resolved by adding it to repo)

now add & commit command are two important for overll process finished whether for modified or untracked data
1. **add** :- adds new or changed files in your working directory to the git staging areas(means ready for commit) like making a new index.html file and adding that then after adding again check status u will got file staged green then in same way git add README.md(if changes are made in readme file) or when there are multiple changes then we use **git add .** which means all (space between add and dot)

2. **commit command** :- which means record of the change]
syntax :- **git commit -m "some message**" (-m means message) and message means like add new button or fix bug like these..
in this only staged files get commited

(2 files changed, 12 insertions(+), 1 deletion(-)
create mode 100644 index.html
PS C:\Users\Subham Jha\Desktop\gitdemo\SubhamJhademo> git status
On branch main
Your branch is ahead of 'origin**/main' by 1 commit.**
(use "**git push**" to publish your local commits)) -- > something like this is appeared which means i make changes in 2 files and one saying like your branch is ahead of main by 1 commit (which means in our local system we become ahead by 1 commit as compare to remote system GitHub means now u unable to see new files on GitHub so for making available on remote system called GitHub we use push command as given in suggestion also)

push command
push :- upload local repo content to remote repo
syntax :- **git push origin main** (for the first time it ask for authorization do that) u can see total commits also on GitHub for that repo

git push mean normal pushing code from local to remote and origin means it is the name (When you clone a repo, Git names the remote origin instead of using long url it uses default

name origin (given to remote repository)) main means default main branch that we are working upon later study more branches

(**When you clone a repo → Git automatically names it origin,** You are free to rename it by Rename origin to something else **(git remote rename origin** github) now new name is GitHub or can put anything)
now comeback to orginal folder or get out of cloned repo use (**cd ..)** to get out then make (mkdir localrepo) means make new directory which name is localrepo
now local repo folder is created now cd localrepo then clear check for ls -Force nothing shows means its not a git repo we need to make it a git repo using command

1**. git init** in terminal then check using ls -force u will see .git file then make some change in that file like creating html and css files then check status again untracked then add using git add . then commited using (git commit -m "add initial files")

then for pushing this whole folder into new repository we are going to create a new repo in GitHub but not initializing readme now bcs for that we also need to create readme in our local system also so without making it on just create a new repo

then for adding use syntax **:- git remote add origin <-link->(**means This command connects your local Git repository to a remote repository (usually on GitHub).) (origin means assigning name origin by default(By convention, the main remote is called origin) then paste link of https from GitHub localrepo project which is latest created)

and now for verifying the remote :- we use syntax :- git remote -v (it will show u the link of project used on which we are going to perform push work (there is extra syntax :- git branch for checking branch name) and for changing branch name we use git branch -M newname)

now at final :- git push origin main

again making a readme file at local system in vs using extension README.md file name (and put # before writing for making text in h1 level ## for h2.. same on)
then just add to the system then commit then push again by same syntax as above ... if we are working on same repo and push many codes later then for make easing pushing we can create some shortcuts like (git push -u origin main) ---one time its means making as upstream for later just u need to write only git push from the next time

but always work as making first on GitHub then clone this way is good

workflow :- 1. GitHub repository 2. cloning 3. make changes 4. add 5. commit 6. push

GITHUB SYSTEM SETTINGS

Even if your repository is public, you can control what email shows up, but your commit messages and code changes will still be visible to everyone.

1. Private Repository Contributions
GitHub contribution graph (the green squares on your profile) can still track commits to private repositories, but only if you allow it.
By default, commits in private repos do not show publicly. You need to opt-in to display them in your graph.

2. How to Show Private Contributions
Go to GitHub → Settings → Profile → Contributions (or Profile settings).
Enable "Include private contributions on my profile".
After enabling:
Your contributions (green squares) will appear on your graph.
But the commits themselves remain private—no one can see code, repo name, or commit messages.
3. Key Takeaways
Private repo commits do not show publicly unless you make the repo public.
Contribution graph can still reflect activity without revealing sensitive info.
Public repos always show commits and code to everyone.

Deleting a repository removes the associated commits from your profile contribution graph. If you want to keep the green squares without keeping the repo public, you could instead:
Keep commits there, which still count on your graph if you opted in.

A commit is like a snapshot of your work in a repository.
Think of it as a "save point" in a game. Each commit records:
The changes you made (new code, edits, or deletions)
Who made the changes (your GitHub username and email)
When the changes were made

A commit message explaining what was done

A **contribution** is any activity that GitHub counts as part of your profile activity.
It's what makes the green squares appear on your GitHub profile.
Examples of contributions:
Commits to a repository (public or private, if you opt-in)
Opening issues
Creating pull requests
Reviewing code or merging pull requests
Important:
Every commit you make in a repo counts as a contribution.
But contributions also include other actions beyond commits.

Every GitHub account has a special repo like:
https://github.com/yourusername/yourusername
This repo is used to power your profile page (your README acts like a personal bio).
It cannot be made private, because GitHub uses it to display your profile publicly. (per account one special repo is allowed)

This is how i optimize GitHub using gprm

1. Why your username repo content shows on top
GitHub treats the repo named exactly like your username as your profile repository.
The README.md in this repo is displayed directly on your profile page, above your list of repositories.
appear immediately when someone visits your profile.

2. Things to remember
Public repos can always be made private (except if it's your username repo).
Private repos can't be made more private, obviously 😄
Username repo: cannot be made private, because GitHub uses it to show your profile

README publicly.can make but later change it using settings button then danger zone settings in navbar below...
can make any repo pinned or unpin from profile on clicking repo for private repositories click on repo section on right side not on profile

SUMMARIZED WAY FOR READY TO WORK TILL PUSH

1. **git clone https**
2. **ls**
3. **ls -Force**
4. **cd foldername**
5. **git status (unmodified, modified (1. add 2. commit, staged (commit), untracked(add))**
6. **git add filename / git add . (for all changes to be added once)**
7. **git commit -m ""some message"**
8. **when make commits in local repo after cloning then u will found after getting check git status ()Your branch is ahead of 'origin/main' by 1 commit.) this means now u also have to update that changes in github repo so we use git (git push origin main)**
9. **origin is the default name of remote repo and main is the branch and git push means send me commited changes to remote repo.....can change**

origin name by using (git remote rename origin newname)

10. command :- cd.. for get out to cloned repo or all

11.

| Command | Meaning |

| ---------- | ---------- |

| cd foldername | Go inside a folder |

| cd .. | Go one level up |

| cd | Go to home directory |

12.mkdir localrepo (name of new directory)

13. then cd localrepo

14. ls -Force

15. git init (then we get .git folder in it)

16. then make some chnage or edit files in local repo then add or commt all the changes

17. now u need to connects your local Git repository to a remote repository (usually on GitHub) : - by using (git remote add origin <-link->) then (git remove -v) for verification and for checking branch name (git branch) or for

chnaging branch name :- (git branch -M newname)(IN THIS FIRST NEED TO MAKE A GITHUB REPO THEN CONNECT TO IT)

18. then git push origin main

19. can make separate readme file for loaclrepo also & make chnages then add then commit and if working on same repo and need to make more pushs then use (git push -u origin main) this will make that localrepo upstream the from next time just direct (git push for that repo)

20. git push -f origin main (for forcefully push) as readme separately created in remote

21. Best practice:

✓ Add README in your local project folder

✓ Then push it

✓ Don’t create README separately on GitHub

This keeps everything clean.

22. Yes ✅ — GitHub allows HTML inside README.md. in remote and preffred markdown in local repo but can use html also

23. We can make commit and add in local repo before connecting to remote repo till that everything is tracked or saved in .git folder of local repo

THIS IS FIRST PART THANK YOU NEXT PART IS COMING SOON FOR BRANCHING AND PULL REQUEST ETC.

……SUBHAM JHA