🧭 Git & GitHub Tutorial

🧩 1. What is Git and Git Hub

Many people mix these up, but they serve different purposes. Here's the breakdown 👇

🧠 What Is Git?

Git is a **version control system** — a tool that tracks changes in your files (usually code).
It lets you:

- Keep a history of your work
- Revert to earlier versions
- Manage multiple branches
- Work locally without the internet

Where it runs: on your **local computer**

Example commands:

```
git init
git add .
git commit -m "Initial commit"
```

Analogy:
Git is the **engine** that powers version tracking.

☁️ What Is GitHub?

GitHub is a **cloud-based platform** that hosts Git repositories online.
It allows you to:

- Store your code on the web
- Share projects with others
- Collaborate using issues, pull requests, and branches
- Contribute to open-source projects

Where it runs: on the **internet (GitHub servers)**

Analogy:
GitHub is like **Google Drive for Git projects**, with teamwork features built in.

⚖️ Git vs GitHub Comparison

Feature	Git	GitHub
Type	Version control software	Online hosting platform
Purpose	Track and manage changes	Host and collaborate on Git repositories
Works offline?	✅ Yes	❌ No
Installed on	Your computer	Website / cloud
Owner	Open-source (Linus Torvalds)	Microsoft
Example command	`git commit -m "message"`	`git push origin main`
Used for	Version control	Collaboration and storage

💡 In Short

> **Git** = the technology (local version control)
> **GitHub** = the platform (remote hosting and collaboration for Git projects)

🧱 2. Prerequisites

Make sure you have:

- Git installed
- A GitHub account
- Your name and email configured in Git (only needed once):

```
git config --global user.name "Your Name"
git config --global user.email "your@email.com"
```

🗂️ 3. Create a Local Project Folder

In your terminal (or PowerShell on Windows) (or Cygwin in Windows)

```
mkdir my-project
cd my-project
git init
```

✅ This creates an empty Git repository (a hidden `.git` folder).

🧰 Common Git Commands and Their Meanings

A quick reference for everyday Git usage.

🔧 Setup & Configuration

Command	Description
`git config --global user.name "Your Name"`	Sets your global Git username.
`git config --global user.email "you@example.com"`	Sets your global Git email address.
`git config --list`	Displays all current Git configuration settings.

🏗️ Creating & Initializing

Command	Description
`git init`	Initializes a new Git repository in the current folder.
`git clone <url>`	Makes a copy of a remote repository on your local machine.

📂 Tracking Files

Command	Description
`git status`	Shows the current state of the repository (changes, untracked files, etc.).
`git add <file>`	Stages a specific file for the next commit.
`git add .`	Stages all modified and new files.
`git rm <file>`	Removes a file from the repository and staging area.

💾 Saving Changes

Command	Description
`git commit -m "message"`	Records the staged changes with a descriptive message.
`git commit -am "message"`	Stages and commits all tracked files in one step.
`git log`	Displays the commit history.
`git diff`	Shows the differences between modified files and the last commit.

🌿 Branching & Merging

Command	Description

Command	Description
`git branch`	Lists all local branches.
`git branch <name>`	Creates a new branch.
`git checkout <branch>`	Switches to the specified branch.
`git switch <branch>`	Modern alternative to `git checkout`.
`git merge <branch>`	Merges another branch into the current branch.
`git branch -d <branch>`	Deletes a branch.

🚀 Working with Remotes

Command	Description
`git remote -v`	Lists remote repositories.
`git remote add origin <url>`	Links your local repo to a remote one.
`git push -u origin main`	Pushes commits to the remote `main` branch and sets it as default.
`git push`	Uploads local commits to the remote repository.
`git pull`	Fetches and merges changes from the remote repository.
`git fetch`	Downloads commits, branches, and files from a remote repository (without merging).

🧹 Undoing & Fixing Mistakes

Command	Description
`git restore <file>`	Discards changes in a file (before commit).
`git reset <file>`	Unstages a file but keeps changes.
`git reset --hard`	Resets the repository to the last commit (⚠ erases changes).
`git revert <commit>`	Creates a new commit that undoes a specific previous commit.

📦 Tags & Versions

Command	Description
`git tag`	Lists all tags.
`git tag <tagname>`	Creates a new tag.

Command	Description
`git push origin <tagname>`	Pushes a specific tag to the remote repository.
`git push origin --tags`	Pushes all tags to the remote.

🕵️ Viewing History

Command	Description
`git show`	Shows details of a specific commit.
`git log --oneline --graph`	Displays the commit history in a simplified graphical format.
`git blame <file>`	Shows who made changes to each line of a file.

💡 Tip

You can use:

```
git help <command>
## 📝 3. Add a File and Commit It

Create a new file and commit it:

```bash
echo "# My First Git Project" > README.md
git add README.md
git commit -m "Initial commit"
```

✅ You now have one file committed locally.

🧰 Common Git Commands and Their Meanings

A quick reference for everyday Git usage.

🔧 Setup & Configuration

Command	Description
`git config --global user.name "Your Name"`	Sets your global Git username.
`git config --global user.email "you@example.com"`	Sets your global Git email address.
`git config --list`	Displays all current Git configuration settings.
```

🏗️ Creating & Initializing

Command	Description
`git init`	Initializes a new Git repository in the current folder.
`git clone <url>`	Makes a copy of a remote repository on your local machine.

📂 Tracking Files

Command	Description
`git status`	Shows the current state of the repository (changes, untracked files, etc.).
`git add <file>`	Stages a specific file for the next commit.
`git add .`	Stages all modified and new files.
`git rm <file>`	Removes a file from the repository and staging area.

💾 Saving Changes

Command	Description
`git commit -m "message"`	Records the staged changes with a descriptive message.
`git commit -am "message"`	Stages and commits all tracked files in one step.
`git log`	Displays the commit history.
`git diff`	Shows the differences between modified files and the last commit.

🌿 Branching & Merging

Command	Description
`git branch`	Lists all local branches.
`git branch <name>`	Creates a new branch.
`git checkout <branch>`	Switches to the specified branch.
`git switch <branch>`	Modern alternative to `git checkout`.
`git merge <branch>`	Merges another branch into the current branch.
`git branch -d <branch>`	Deletes a branch.

🚀 Working with Remotes

Command	Description
`git remote -v`	Lists remote repositories.
`git remote add origin <url>`	Links your local repo to a remote one.
`git push -u origin main`	Pushes commits to the remote `main` branch and sets it as default.
`git push`	Uploads local commits to the remote repository.
`git pull`	Fetches and merges changes from the remote repository.
`git fetch`	Downloads commits, branches, and files from a remote repository (without merging).

🧹 Undoing & Fixing Mistakes

Command	Description
`git restore <file>`	Discards changes in a file (before commit).
`git reset <file>`	Unstages a file but keeps changes.
`git reset --hard`	Resets the repository to the last commit (⚠ erases changes).
`git revert <commit>`	Creates a new commit that undoes a specific previous commit.

📦 Tags & Versions

Command	Description
`git tag`	Lists all tags.
`git tag <tagname>`	Creates a new tag.
`git push origin <tagname>`	Pushes a specific tag to the remote repository.
`git push origin --tags`	Pushes all tags to the remote.

🕵️ Viewing History

Command	Description
`git show`	Shows details of a specific commit.
`git log --oneline --graph`	Displays the commit history in a simplified graphical format.
`git blame <file>`	Shows who made changes to each line of a file.

💡 Tip

You can use:

```
git help <command>
```