What is Git and GitHub? --- A Beginner's Guide

Key Concepts (In Simple Words):

Git:A tool that tracks changes in files, like version control.

GitHub:A website that hosts your Git repositories online.

Repository:A folder/project on GitHub that holds your files and history.

Commit:Saving your changes with a message.

Push:Sending your saved changes to GitHub.

Pull:Getting the latest version from GitHub.

Branch:A copy of your project to experiment on.

Merge:Combining changes from different branches.

What is GitHub?

GitHub is a website and cloud platform where people can store, manage, and share code. It works like Google Drive for code but includes powerful version control and collaboration features.

Imagine this:

You're building a project. Every time you change it, you want to:

- Save your work

- Go back to an earlier version if needed

- Work with others without messing up their work, GitHub lets you do all that.

Who Uses GitHub?

- Programmers

- Students

- Researchers

- Designers

- Companies and Teams

What Can You Do with GitHub?

- Save your code online

GitHub: A Beginner's Guide

- Share code with others

- See change history

- Show off work

- Contribute to open-source

Git and GitHub Command Cheat Sheet

1. Command: git init

Info: Initializes a new Git repository in your current directory.

When to Run: Use when starting a new project and want to track it with Git.

How to Run: Navigate to your project folder and run `git init`.

Example: C:\majrepos> git init

1. Command: git clone <repo-url>

Info: Clones an existing repository from GitHub or another remote.

When to Run: Use when you want to make a local copy of a remote repository.

How to Run: Run the command with the repo URL.

Example: git clone https://github.com/user/repo.git

1. Command: git add <file>

Info: Adds file(s) to the staging area.

When to Run: Use after editing/creating files and before committing.

How to Run: Run `git add filename` or `git add .` to add all.

Example: git add hello.txt

1. Command: git commit -m "message"

Info: Records changes in the repository with a message.

When to Run: Use after staging files with `git add`.

How to Run: Run `git commit -m "your message"`.

Example: git commit -m "Updated hello.txt with 4 lines"

1. Command: git status

Info: Shows the status of working directory and staging area.

When to Run: Use to see which changes are staged, unstaged, or untracked.

How to Run: Simply run the command.

Example: git status

1. Command: git push

Info: Uploads local repository content to a remote repository.

When to Run: Use after committing to send changes to GitHub.

How to Run: Run `git push`, or use `--set-upstream` if first time.

Example: git push --set-upstream origin main

1. Command: git pull

Info: Fetches and integrates changes from a remote repo.

When to Run: Use to get updates from the remote before pushing.

How to Run: Run `git pull origin main`.

Example: git pull origin main

1. Command: git remote -v

Info: Displays the URLs of the remotes.

When to Run: Use to check which remote repositories are connected.

How to Run: Run the command directly.

Example: git remote -v

1. Command: git log

Info: Shows the commit history.

When to Run: Use to see all previous commits.

How to Run: Run `git log` or `git log -1` for latest.

Example: git log -1

1. Command: git config --global user.name "Your Name"

Info: Sets your Git username globally.

When to Run: Use it once to set your identity.

How to Run: Run with your name in quotes.

Example: git config --global user.name "Mohammad Majeed"

Git + GitHub: Revision & Some More Commands

## git init

* 📘 what: Initializes a new Git repository in your local folder.
* 🕒 when: First time you start version control on a new project.
* 🧪 How:

git init

## git clone <url>

* 📘 What: Downloads a GitHub repo to your local machine.
* 🕒 When: You want to start working on a project that already exists on GitHub.
* 🧪 How:

git clone https://github.com/username/repo-name.git

## git add <file>

* 📘 What: Stages changes for commit.
* 🕒 When: After editing/creating a file and you're ready to include it in the next commit.
* 🧪 How:

git add hello.txt  
OR  
git add .

## git commit -m "message"

* 📘 What: Records the staged changes with a message.
* 🕒 When: After `git add`, to save your changes in Git.
* 🧪 How:

git commit -m "Added new features"

## git remote add origin <url>

* 📘 What: Links your local repo to a remote GitHub repo.
* 🕒 When: First time you're connecting your local repo to GitHub.
* 🧪 How:

git remote add origin https://github.com/username/repo-name.git

## git push

* 📘 What: Uploads commits from your local repo to GitHub.
* 🕒 When: After committing changes, to share them on GitHub.
* 🧪 How:

First time: git push --set-upstream origin main  
Next time: git push

## git pull

* 📘 What: Downloads and merges the latest changes from GitHub.
* 🕒 When: Before you start working — to get updates from others or GitHub.
* 🧪 How:

git pull origin main

## git status

* 📘 What: Shows which files are changed, staged, or committed.
* 🕒 When: Anytime to check what’s going on with your files.
* 🧪 How:

git status

## git log

* 📘 What: Shows the history of commits.
* 🕒 When: To see what changes were made and by whom.
* 🧪 How:

git log

## git reset

* 📘 What: Removes staged files from the staging area.
* 🕒 When: You accidentally ran `git add` but changed your mind.
* 🧪 How:

git reset

## git rm <file>

* 📘 What: Deletes a file and stages the removal.
* 🕒 When: You want to remove a file from both your repo and disk.
* 🧪 How:

git rm hello.txt

## git diff

* 📘 What: Shows changes in the working directory vs last commit.
* 🕒 When: To review what you changed before committing.
* 🧪 How:

git diff

## git merge

* 📘 What: Combines changes from one branch into another.
* 🕒 When: You're done working on a feature and want to merge it into `main`.
* 🧪 How:

git checkout main  
git merge feature-branch

## git branch

* 📘 What: Shows or manages branches.
* 🕒 When: Working on features separately or checking current branch.
* 🧪 How:

git branch  
 OR  
 git branch new-feature

## git checkout

* 📘 What: Switches to another branch.
* 🕒 When: You want to work on a different branch.
* 🧪 How:

git checkout main

## git stash

* 📘 What: Temporarily saves your uncommitted changes.
* 🕒 When: You want to switch branches but don’t want to commit yet.
* 🧪 How:

git stash

## git config

* 📘 What: Sets your Git username and email for commits.
* 🕒 When: First time setup.
* 🧪 How:

git config --global user.name "Majeed"  
git config --global user.email "majfarfaiz@gmail.com"

## git clean -f

* 📘 What: Removes untracked files (careful!).
* 🕒 When: Cleaning your working directory.
* 🧪 How:

git clean -f

## echo "text" > file.txt

* 📘 What: Creates or overwrites a file with text (Windows).
* 🕒 When: Anytime you need to create a new text file for testing or demo.
* 🧪 How:

echo "Hello, GitHub!" > hello.txt