🚀 Beginner’s Guide to Command Line & Git (With Simple Explanations)

⚠️ **Important Note (Read First)**
The commands below are written for **Linux / macOS terminals** (and **WSL or Git Bash on Windows**).

If you run these commands directly in **Windows Command Prompt (cmd)** or **PowerShell**, some commands **will not work** or will behave differently.

✅ **Windows users:** Use **Git Bash**, **WSL (Ubuntu)**, or **VS Code Terminal** for best results.

🖥️ Basic Command Line (CLI) Commands

These commands help you **navigate and manage files/folders** using the terminal.

Command	What it does
`mkdir folder_name`	Creates a new folder
`ls`	Lists files and folders in the current directory
`ls -a`	Shows all files, including hidden ones (like `.git`)
`cd folder_name`	Moves into a folder
`cd ..`	Moves one level back
`touch file.txt`	Creates a new file
`cat file.txt`	Displays file contents in the terminal

📌 These commands are the **foundation** for working with Git.

🧱 Initialising a Git Repository

These commands set up Git and show the current state of your project.

Command	Purpose
`git init`	Starts Git tracking in the current folder
`git status`	Shows modified, staged, and untracked files

📁 `git init` creates a hidden `.git` folder that stores your project history.

🎭 Staging & Committing (Git Concept Explained Simply)

Think of Git like a **photo album**:

- Files = people
- Staging area = stage
- Commit = taking a photo

Command	What it does
`git add file.txt`	Adds a file to the staging area
`git add .`	Stages all changes
`git commit -m "message"`	Saves a snapshot of staged changes
`git restore --staged file.txt`	Removes a file from staging

📌 Only **staged files** are included in a commit.

🕰️ Viewing History & Undoing Changes

These commands help you **review and fix mistakes**.

Command	Use
`git log`	Shows commit history
`git reset commit_hash`	Moves back to a previous commit
`git reset --hard commit_hash`	Deletes all changes after that commit ⚠️
`git stash`	Temporarily saves uncommitted changes
`git stash pop`	Restores stashed changes

⚠️ **Be careful with `--hard`** — it permanently deletes work.

🌿 Branching & Merging

Branches let you work on features **without affecting the main code**.

Command	Function
`git branch branch_name`	Creates a new branch
`git checkout branch_name`	Switches to a branch
`git checkout -b branch_name`	Creates & switches in one step
`git merge branch_name`	Merges another branch into current
`git rebase -i commit_hash`	Cleans commit history (squash commits)

📌 Branching is **mandatory** in professional workflows.

🌍 Working with Remote Repositories (GitHub)

These commands connect your local project to GitHub.

Command	Purpose
`git clone url`	Downloads a GitHub repository
`git remote add origin url`	Connects local repo to GitHub
`git remote -v`	Shows linked remotes
`git push origin branch_name`	Uploads commits to GitHub
`git fetch origin`	Downloads updates without merging
`git pull origin branch_name`	Fetches + merges changes

📌 `git pull = git fetch + git merge`

🧠 Pro Tips for Beginners

✔ Always run `git status` before committing
✔ Write **clear commit messages**
✔ Create a new branch for every change
✔ Never work directly on `main`
✔ Use Git Bash / WSL on Windows