
📘 Git & GitHub Roadmap

A Beginner-to-Intermediate Learning Path

⚠️ Important Note (Before You Start)

The commands in this roadmap are intended for **Linux-based terminals**
(Linux, macOS, **WSL**, or **Git Bash on Windows**).

🔴 **Windows Command Prompt (CMD) / PowerShell** may not support all commands.
✅ Recommended environments:

- Git Bash
- WSL (Ubuntu)
- VS Code Integrated Terminal

🧭 Roadmap Overview

This roadmap is divided into **6 progressive stages**:

1. Command Line Fundamentals
2. Git Basics & Repository Setup
3. Staging & Committing Workflow
4. History Management & Undoing Changes
5. Branching & Collaboration
6. Working with GitHub (Remote Repositories)

Each stage builds on the previous one.

🟢 Stage 1: Command Line (CLI) Fundamentals

Before learning Git, it is essential to understand basic terminal commands used to manage files and directories.

Essential CLI Commands

Command	Description
mkdir folder_name	Creates a new directory
ls	Lists files and folders
ls -a	Shows hidden files (e.g. .git)
cd folder_name	Moves into a directory
cd ..	Moves back one directory
touch file.txt	Creates a new file
cat file.txt	Displays file contents

📌 These commands form the **foundation** of Git usage.

🟡 Stage 2: Initialising Git & Checking Status

This stage introduces Git and how it tracks project changes.

Git Setup Commands

Command	Purpose
git init	Initializes a Git repository
git status	Displays current project state

📁 git init creates a hidden .git directory which stores the complete version history.

🔵 Stage 3: Staging & Committing Changes

Git follows a **two-step save process**:

1. Stage files
2. Commit changes

Core Commands

Command	Function
git add file.txt	Stages a specific file

Command	Function
git add .	Stages all changes
git commit -m "message"	Saves a snapshot of changes
git restore --staged file.txt	Removes file from staging

🧠 Conceptual Analogy

- Files → Guests
- Staging area → Stage
- Commit → Photograph
- Git history → Photo album

🟣 Stage 4: Viewing History & Undoing Changes

This stage helps you **review past work and recover from mistakes**.

History & Recovery Commands

Command	Description
git log	Shows commit history
git reset commit_hash	Moves HEAD to previous commit
git reset --hard commit_hash	Deletes all later changes ⚠️
git stash	Temporarily saves uncommitted work
git stash pop	Restores stashed changes

⚠️ Use --hard carefully — changes cannot be recovered.

🟠 Stage 5: Branching & Merging

Branches allow developers to work independently without affecting the main codebase.

Branch Management Commands

Command	Purpose
git branch branch_name	Creates a new branch

Command	Purpose
git checkout branch_name	Switches branches
git checkout -b branch_name	Create & switch branch
git merge branch_name	Merges branch into current
git rebase -i commit_hash	Rewrites commit history

📌 Professional workflows **always use branches**.

🔴 Stage 6: Working with GitHub (Remote Repositories)

This stage connects local Git repositories with GitHub.

Remote Repository Commands

Command	Function
git clone url	Downloads a GitHub repository
git remote add origin url	Links repo to GitHub
git remote -v	Lists remote URLs
git push origin branch_name	Uploads commits
git fetch origin	Downloads updates
git pull origin branch_name	Fetch + merge updates

🧠 **Key Concept**
git pull = git fetch + git merge

🧩 Complete Git Workflow Summary

Edit files

↓

git status

↓

git add .

↓

git commit -m "message"

↓

git push origin branch_name

🎯 Learning Outcomes

After completing this roadmap, you will be able to:

- Use Linux terminal confidently
- Track code changes using Git
- Work with branches professionally
- Undo mistakes safely
- Collaborate using GitHub
- Follow real-world DevOps workflows