GIT & GITHUB COMPLETE MASTERY GUIDE

Beginner to Advanced | Practical | Interview Ready This book-style documentation is designed to make you fully confident in Git and GitHub. It covers concepts, commands, workflows, real-world practices, and professional tips.

1. Introduction to Version Control

Version control is a system that records changes to files over time. It allows developers to track history, collaborate with others, and safely experiment with new ideas.

Before version control, developers used manual backups like final_v1, final_v2, which was inefficient and error-prone.

Git is a distributed version control system created by Linus Torvalds in 2005. Every developer has a full copy of the repository, including history.

2. Centralized vs Distributed Version Control

Centralized systems like SVN have a single server that stores all versions. If the server fails, collaboration stops.

Distributed systems like Git allow every developer to have a full copy of the repository. This makes Git fast, secure, and reliable.

Git enables offline work and powerful branching.

3. Installing Git and Initial Configuration

Git can be installed on Windows, Linux, and macOS.

After installation, configure your username and email. These details are stored in commit history.

Git configuration has three levels: system, global, and local.

4. Understanding Git Architecture

Git has three main areas: Working Directory, Staging Area, and Repository.

The working directory contains your actual project files.

The staging area is where changes are prepared before committing.

The repository stores committed snapshots.

5. Creating Your First Git Repository

A repository is created using git init.

Git tracks changes only after files are added to the staging area.

The .git folder contains all Git metadata and history.

6. Git File States

Files in Git can be untracked, modified, staged, or committed.

Understanding file states helps prevent accidental commits.

git status is one of the most important commands.

7. Essential Git Commands Explained

git add moves changes to the staging area.

git commit creates a snapshot with a message.

git log shows commit history.

git diff compares changes between states.

8. Writing Professional Commit Messages

A good commit message explains what and why.

Use present tense and keep messages concise.

Professional teams enforce commit standards.

9. Branching Fundamentals

Branches allow parallel development.

The main branch represents stable code.

Feature branches isolate work.

10. Merging and Merge Conflicts

Merging combines changes from different branches.

Conflicts occur when Git cannot decide which change to keep.

Understanding conflict resolution is essential.

11. Git Reset, Revert, and Checkout

git reset rewrites history.

git revert safely undoes commits.

git checkout switches branches and restores files.

12. Git Stash and Temporary Work

Stashing allows you to save unfinished work temporarily.

It is useful when switching branches quickly.

13. Introduction to GitHub

GitHub is a platform for hosting Git repositories.

It enables collaboration, code review, and open-source contribution.

14. Creating and Managing GitHub Repositories

Repositories can be public or private.

README.md explains the project.

Licenses define usage rights.

15. Connecting Local Repository to GitHub

Remote repositories are added using git remote.

git push uploads commits.

git pull downloads and merges changes.

16. Forking and Cloning

Forking creates a personal copy of a repository.

Cloning downloads repositories to your system.

17. Pull Requests and Code Review

Pull requests propose changes.

They enable discussion and review.

Professional teams rely heavily on PRs.

18. GitHub Issues and Project Boards

Issues track bugs and features.

Project boards organize work visually.

19. Advanced Git Concepts

Rebasing creates linear history.

Cherry-pick applies specific commits.

Tags mark releases.

20. GitHub Actions and Automation

GitHub Actions automate testing and deployment.

CI/CD improves code quality and speed.

Congratulations! You have completed a full beginner-to-advanced guide on Git and GitHub. You are now ready for real-world projects, internships, and professional development.