Git Pro: Beginners Guide to Git and GitHub

Sayan Ghosh

July 16, 2025

IC&SR, IIT Madras

Agenda

- Core Git Concepts
- GitHub Integration
- Branching & Merging
- Collaboration: Pull Requests
- Automation with GitHub Actions
- Makefiles for Automation
- Command Line Essentials
- Real-World Workflow
- Wrap-up & Q&A

Introduction

Why Version Control?

- Keep track of changes in your work
- Collaborate safely without overwriting each other’s files
- Example: Version chaos vs Git timeline

Core Git Concepts

What is Git? What is GitHub?

- Git: Local version control system
- GitHub: Remote hosting and collaboration

Key Concepts

- Repository, Working Directory, Staging Area
- Commits & History

Three Stages of Git

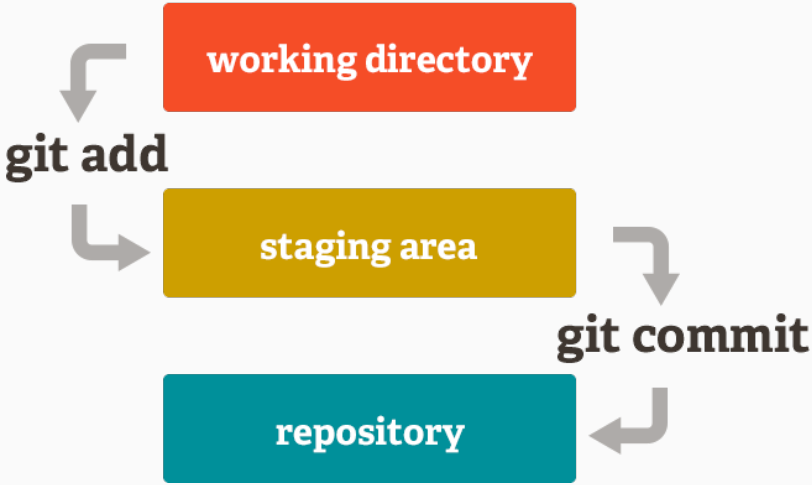

Figure 1: Three Stages of Git

Essential Commands

- `git init`, `git add`, `git commit`
- `git status`, `git log`

GitHub Integration

Local to Remote

- Add remote: `git remote add origin`
- Push changes: `git push`
- Clone: `git clone`

Demo: Connect Local Repo

1. Create repo on GitHub
2. Link remote
3. Push first commit

Branching and Merging

Why Use Branches?

- Work on new features safely
- Resolve conflicts systematically
- Keep track of different development lines

Branches Can Get Messy

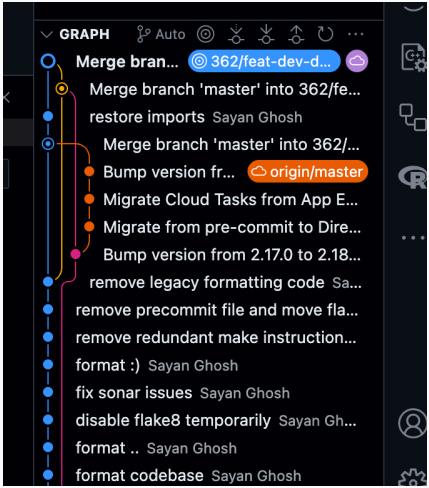

Figure 2: Branches can get messy!

Basic Commands

- `git branch`, `git checkout -b`
- `git merge`
- Resolving merge conflicts

Collaboration

Issues

- Track bugs, feature requests
- Assign to team members
- Link issues to commits

Pull Requests

- Review and discuss code before merging
- Assign reviewers, track changes

Git Flow vs GitHub Flow

- Git Flow: Develop, feature, release, hotfix branches
- GitHub Flow: Simple, single main branch with feature branches

Interactive Group Task: GitHub Collaboration

1. Form groups of 3–4 members.
2. One member creates a new GitHub repository for the group.
3. Add all teammates as collaborators with `Write` access.
4. Clone the shared repo to your local machine.
5. Create issues for features to add.
6. Create branches referring to the issues.
7. Add or edit files.
8. Push your branch and open a Pull Request (PR).
9. Review each other's PRs and merge them.
10. Intentionally create and resolve at least one merge conflict.

Tips: Use clear commit messages, meaningful branch names, and communicate with your team!

Automation

Intro to CI/CD

- Continuous Integration & Continuous Deployment
- Automate testing, linting, deployment

GitHub Actions

- Workflows defined in YAML
- Trigger on push/pull request
- Example: Auto lint on push

Makefiles

Why Makefiles?

- Automate repeated tasks: build, test, clean
- Simple syntax: `target: dependencies`

Makefiles with GitHub Actions

- Call Makefile tasks in your workflow
- Example: `make test`

Command Line Essentials

Linux Commands

- `ls`, `cd`, `cat`, `grep`, `chmod`
- Helpful for day-to-day developer tasks

Putting It All Together

Real-World Scenario

1. Clone repo → branch → commit → PR → review → merge
2. CI/CD runs automatically

Wrap-up

Key Takeaways

- Git for version control
- GitHub for collaboration
- Automate with Actions and Makefiles

Resources

- GitHub Learning Lab
- Official docs and cheat sheets
- Practice on real projects!

URL Submission & Feedback

- Submit your GitHub repo URL and your feedback for this workshop.
- Use the form: `https://forms.gle/my9gXm8ETkcqy9Up9`
- or scan the QR Code:

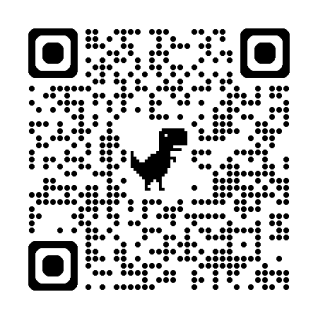

Figure 3: Scan to submit your URL and feedback!

Thank You!

Questions?

Contact: sayan@study.iitm.ac.in