Mastering Git & GitHub: From Zero to Collaboration

A Comprehensive Hands-on Course

Abderrahman Bouanani

ENSAA

October 12, 2025

Course Overview

- Understand version control concepts
- Master Git commands and workflows
- Collaborate effectively using GitHub
- Handle real-world scenarios
- Learn best practices and troubleshooting

Course Structure

1. Session 1: Version Control & Git Fundamentals
2. Session 2: Collaboration on GitHub
3. Session 3: Common Git Problems
4. Conclusion

Learning Objectives

By the end of this session, you will be able to:

- Explain the importance of version control
- Set up and configure Git
- Create and manage repositories
- Track changes with commits

Why Version Control?

Scenario: The "One Small Change" Problem

- Your program is working perfectly.
- You change "just one little thing"...
- Your program breaks.
- You try to change it back...
- Your program is still broken!

What is Version Control?

A system that records changes to files over time

Think of it as a **time machine** for your code!

Key Features

- **History**: See who changed what and when.
- **Backup**: Recover any previous version.
- **Collaboration**: Work with others seamlessly.

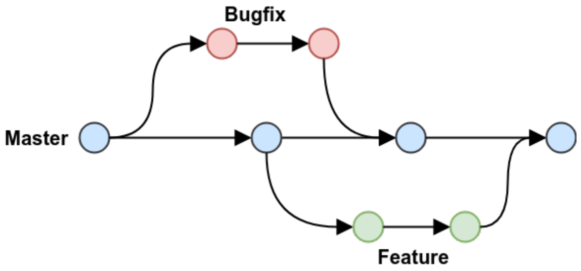
Analogy

Like "Track Changes" in a document, but supercharged for code.

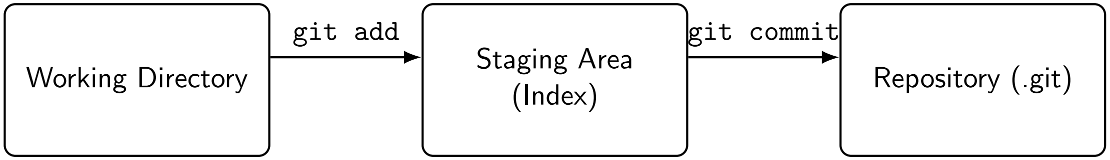
What is Git?

A distributed version control system

- Created by Linus Torvalds in 2005
- Initially developed for Linux kernel development
- Now the most widely used VCS in the world



Core Git Concepts: The Three Areas



- **Working Directory**: Files you actively modify.
- **Staging Area**: Draft of your next commit.
- **Repository**: Full history stored in .git.

Essential Git Commands

Basic Commands

`git init`	Initialize a new repository
`git add`	Stage changes for commit
`git commit`	Save changes to the repository
`git log`	View the commit history
`git status`	Check the status of your files

Useful Options

`git status -s`	Compact status view
`git log --oneline`	Compact log view
`git commit -am`	Add & commit in one step
`git diff`	See unstaged changes

Working with Branches

Scenario 2: The "Works on My Machine" Problem

- Your program worked well enough yesterday
- You made improvements last night...
- ...but haven't gotten them to work yet
- You need to turn in your program now

Working with Branches

What are Branches?

Branches are independent lines of development. They let you work on features or fixes without affecting main.

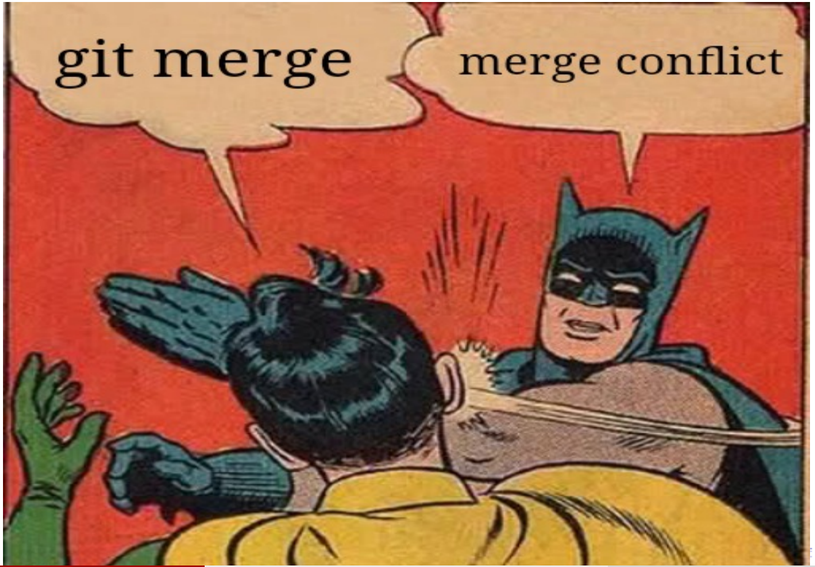
Branch Commands

`git branch`	List all branches
`git branch <name>`	Create a new branch
`git checkout <name>`	Switch branches
`git merge <name>`	Merge a branch

Branching Strategy

- **main**: Production code
- **develop**: Integration branch
- **feature/***: New features
- **hotfix/***: Urgent fixes

Merge Conflicts



How to Resolve Merge Conflicts

What is a Merge Conflict?

Happens when Git cannot automatically merge because two branches edit the **same line** or one deletes a file the other modified.

Step-by-Step Resolution

1. Open the conflicted file(s)
2. Look for conflict markers:
   - `<<<<<<< HEAD` (your changes)
   - `=======`
   - `>>>>>>> branch-name` (their changes)
3. Edit code to keep desired changes & remove markers
4. Save the file
5. Stage the resolved file: `git add <file>`

Example in a File

```
/* style.css */
.title {
<<<<<<< HEAD
  color: blue;
=======
  color: red;
>>>>>>> feature-new-color
}
```

Key Takeaways

What We've Learned

- Version control is essential for tracking changes
- Git provides a powerful way to manage project history
- Basic workflow: modify $\rightarrow$ stage $\rightarrow$ commit

Next Steps

- Practice basic Git commands
- Explore more Git features

End of Session 1

Next: Session 2 — Collaboration on GitHub

Session 2 Begins

Collaboration on GitHub

Session 2: What You'll Learn

GitHub Basics

- Creating repositories
- Pushing code
- Basic collaboration

[Space for GitHub screenshot]

Introduction to GitHub

What is GitHub?

- A cloud platform for hosting Git repositories
- A hub for developer collaboration
- The world's largest open-source community
- A professional portfolio for your work

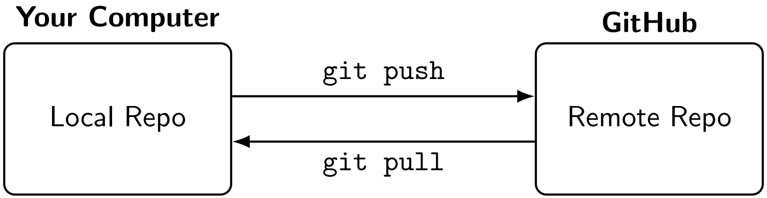
Key Features

- Code hosting & version control
- Issue tracking
- Pull Requests
- GitHub Actions (CI/CD)
- GitHub Pages

The Git + GitHub Workflow

Essential Remote Commands

`git clone <url>`	Download a repo from GitHub
`git push`	Upload committed changes
`git pull`	Download & merge changes
`git remote add origin <url>`	Connect local repo to GitHub



Golden Rule

Always `git pull` before `git push` to avoid unnecessary conflicts!

Code Reviews

Reviewing Code: What to Look For

- Bugs and edge cases
- Adherence to code style
- Opportunities for improvement
- Presence of tests
- Clear documentation

Giving Good Feedback

- Be constructive and kind
- Explain the 'why' behind suggestions
- Ask questions instead of making demands
- Offer help if needed

Pull Requests (PRs)

What is a Pull Request?

A way to propose and discuss changes before merging them into the main project. PRs are the heart of collaboration on GitHub.

The PR Workflow

1. Push your feature branch to GitHub
2. Click "New Pull Request" on GitHub
3. Describe your changes clearly
4. Request reviews from teammates
5. Discuss, make more changes if needed
6. Merge!

End of Session 2

Next: Session 3 — Common Git Problems

Session 3 Begins

Common Git Problems

Common Git Problems & Fixes

"I committed to the wrong branch!"

- **Scenario:** Accidentally committed on ‘main‘ instead of your feature branch
- **Fix:**
  1. Create the correct branch: `git branch feature-branch`
  2. Reset main back one commit: `git reset HEAD˜ --hard`
  3. Switch to your branch: `git checkout feature-branch`
  4. Commit is now safely on the correct branch

"I need to undo my last commit!"

- Keep changes but undo commit: `git reset HEAD˜`
- Permanently delete last commit: `git reset HEAD˜ --hard` (caution!)

Conclusion & Next Steps

What You've Learned

- Git fundamentals & version control
- Using branches for parallel development
- Collaborating on GitHub via Pull Requests
- Handling merge conflicts & common issues

Your Journey Forward

- Practice Git commands daily
- Start a personal project on GitHub
- Contribute to open-source projects

Thank You!

Questions or feedback?