Version Control with Git

Track, Collaborate, and Manage Code (anything)

Bipin C

September 5, 2025

Why Version Control?

- Problems without version control:
  - Overwriting files (e.g., Assignment_final_v2_reallyfinal.docx)
  - Hard to track changes
  - Difficult collaboration
- Git solves these problems:
  - Complete history of changes
  - Easy collaboration
  - Branching and merging

History of Git

- Created in 2005 by Linus Torvalds (creator of Linux kernel)
- Designed after disputes with proprietary version control (BitKeeper)
- Key goals:
  - Speed
  - Simple design
  - Strong support for non-linear development (branches)
  - Distributed architecture
  - Integrity of source code

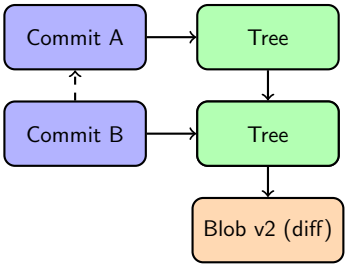
What is Git?

- Distributed version control system
- Tracks changes in files
- Works locally and with remote repositories (GitHub, GitLab, Bitbucket)

How Git Stores Diffs

Git does not store full copies of files for each commit. It stores objects:

- Blobs (file contents)
- Trees (directory structure)
- Commits (metadata + pointers)



Resources

- Official Git docs: `https://git-scm.com/docs`
- Pro Git book: `https://git-scm.com/book`
- Git cheat sheet

Installing Git

- Linux, macOS, Windows
- Verify installation: `git --version`
- Configure:
  - `git config --global user.name "Your Name"`
  - `git config --global user.email "you@example.com"`

Basic Workflow

Working Directory → Staging Area → Repository

- `git init`
- `git status`
- `git add`
- `git commit`

Creating a Repository

- New project: `git init`
- Clone existing: `git clone <url>`

Making Changes

- Edit files
- `git status`
- `git diff`
- `git add filename`
- `git commit -m "message"`

Branches & Merging

- Branch = independent line of development
- `git branch`
- `git checkout -b feature`
- `git merge`

Working with Remotes

- `git remote add origin <url>`
- `git push`
- `git pull`
- `git fetch`

Undoing Changes

- `git restore`
- `git reset`
- `git revert`

GitHub / GitLab

- Host repositories
- Pull requests / merge requests
- Collaboration workflows

Best Practices

- Write meaningful commit messages
- Use `.gitignore`
- Commit often, in small chunks
- Branching strategy (feature branches, main vs dev)

Hands-On Exercise

1. Lets edit this article
2. `git clone git@github.com:vu3bpn/Gitworkshop.git`
3. make edits
4. `git commit -a -m "commit message"`
5. `git push origin master`

Resources

- Official Git docs: `https://git-scm.com/docs`
- Pro Git book: `https://git-scm.com/book`
- Git Guide `https://github.com/git-guides`
- Git cheat sheet

Q&A / Wrap-Up

- What Git is
- Why it matters
- Basic workflow
- Encourage practice with real projects