# Git — Detailed Guide (Comprehensive Reference)

A practical, detailed guide to Git: commands, explanations, examples, and diagrams.

## Table of contents

1. Introduction & philosophy
2. Installation & initial configuration
3. Repository lifecycle (init/clone)
4. Basic workflow (edit → stage → commit)
5. Branching and branching strategies
6. Remote repositories & collaboration (push/pull/fetch)
7. Merging, rebasing, and resolving conflicts
8. Undoing mistakes: reset, revert, restore
9. Stashing, shelving, and cleaning
10. Tagging and releases
11. Advanced: hooks, submodules, bisect, reflog, gc
12. Troubleshooting & best practices
13. Cheat sheet & quick reference
14. Glossary

## 1. Introduction & philosophy

Git is a distributed version control system focused on speed, data integrity, and support for nonlinear workflows. Think of commits as snapshots, branches as pointers to commits, and remotes as references to other repositories.

## 2. Installation & initial configuration

git config --global user.name "Your Name"  
git config --global user.email "you@example.com"  
git config --list

## 3. Repository lifecycle

git init  
git clone <url>  
git clone <url> <folder\_name>

## 4. Basic workflow

git status  
git add <file>  
git add .  
git commit -m "Commit message"

## 5. Branching

git branch  
git branch <branch\_name>  
git checkout <branch\_name>  
git switch <branch\_name>  
git checkout -b <branch\_name>  
git switch -c <branch\_name>

## 6. Remotes

git remote -v  
git remote add origin <url>  
git fetch origin  
git pull origin main  
git push origin main  
git push -u origin main

## 7. Merging & rebasing

git merge <branch>  
git rebase <branch>  
git rebase -i <branch>

Conflict resolution:

git add <file>  
git merge --continue  
git rebase --continue

## 8. Undoing changes

git restore <file>  
git restore --staged <file>  
git reset --soft <commit>  
git reset --hard <commit>  
git revert <commit>

## 9. Stashing & cleaning

git stash  
git stash list  
git stash apply  
git stash pop  
git stash drop

Cleaning:

git clean -n  
git clean -f  
git clean -fd

## 10. Tagging

git tag  
git tag v1.0.0  
git tag -a v1.0.0 -m "Release 1.0"  
git push origin v1.0.0  
git push origin --tags

## 11. Advanced commands

git blame <file> # Show who last modified each line  
git bisect start # Begin binary search for bad commit  
git bisect bad   
git bisect good <commit>  
git reflog # View HEAD history and recover commits  
git gc # Garbage collection  
git fsck # Verify repository integrity  
git submodule add <url> path # Add a submodule  
git submodule update --init --recursive

## 12. Troubleshooting & best practices

* Write descriptive commit messages.
* Keep branches small and focused.
* Avoid force pushing to shared branches unless coordinated.

## 13. Cheat sheet

git status; git add .; git commit -m "msg"  
git switch -c feature-x  
git fetch origin; git pull  
git push -u origin feature-x  
git restore file; git reset --hard HEAD~1  
git tag -a v1.0 -m "release"; git push origin v1.0  
git blame file; git clean -fd

## 14. Glossary

* **HEAD**: current branch pointer
* **Index**: staging area
* **SHA**: commit hash
* **Fast-forward**: merge without creating a merge commit