The Complete Git Command Guide — From Beginner to Pro

A Comprehensive Reference for Every Git Command, Workflow, and Example (Includes VS Code Integration)

Author: Mohit

🏠**Cover Page**

Tagline: *Learn Git like a pro — one command at a time.*

(3D Git logo or minimalist design placeholder)

🗾1. Introduction

What is Git?

Git is a distributed version control system that allows multiple developers to collaborate, track changes, and manage versions of code efficiently.

Why Git is Essential

Git helps manage versions, enables branching and merging for parallel development, and integrates seamlessly with platforms like GitHub and GitLab.

How Git Works

- **Local Repository:** Your personal working copy of the project.
- **Remote Repository:** A shared copy on a platform like GitHub.
- **Workflow:** Work locally ➔ commit ➔ push ➔ pull ➔ merge.

Installing Git

Windows: Download from git-scm.com **macOS:** `brew install git` **Linux:** `sudo apt install git`

(Include basic workflow diagram)

---

2. Git Configuration & Setup

Command	Description	Example
`git config`	Configure Git settings	`git config --global user.name "John Doe"`
`git config --list`	View settings	`git config --list`
`git help`	Get help	`git help commit`

Note: The `.gitconfig` file stores global settings like username, email, and aliases.

---

3. Starting a Project

Command	Description	Example
`git init`	Initialize new repo	`git init my-project`
`git clone`	Clone from remote	`git clone https://github.com/user/repo.git`

(Add visuals of folder structure post-init)

---

4. Basic File Operations

Command	Description	Example
`git add`	Stage files	`git add .`
`git status`	Show changes	`git status`
`git commit -m`	Commit staged files	`git commit -m "Initial commit"`
`git log`	View commit history	`git log --oneline`
`git diff`	Compare changes	`git diff`

🚀5. Branching & Merging

Command	Description	Example
`git branch`	List branches	`git branch`
`git branch <name>`	Create branch	`git branch feature-1`

Command	Description	Example
`git checkout`	Switch branch	`git checkout feature-1`
`git switch`	Switch or create	`git switch -c dev`
`git merge`	Merge branch	`git merge dev`
`git rebase`	Reapply commits	`git rebase main`

(Include branching workflow diagram)

---

6. Remote Repositories

Command	Description	Example
`git remote add`	Add remote repo	`git remote add origin https://github.com/user/repo.git`
`git push`	Upload commits	`git push origin main`
`git pull`	Fetch + merge	`git pull origin main`
`git fetch`	Download metadata	`git fetch origin`
`git remote -v`	List remotes	`git remote -v`

Supported Platforms: GitHub, GitLab, Bitbucket.

---

7. Undoing Changes & Recovery

Command	Description	Example
`git reset`	Unstage / undo	`git reset --soft HEAD~1`
`git revert`	Create reverse commit	`git revert <commit-id>`
`git restore`	Restore file	`git restore file.txt`
`git clean`	Remove untracked files	`git clean -f`

Caution: Use with care. Reset and clean may lead to data loss.

---

8. Stashing & Temporary Work

Command	Description	Example
`git stash`	Save uncommitted changes	`git stash`
`git stash pop`	Reapply stash	`git stash pop`
`git stash list`	Show stashes	`git stash list`

Use Case: When switching branches mid-development.

---

9. Tagging & Versioning

Command	Description	Example
`git tag`	List tags	`git tag`
`git tag -a`	Create annotated tag	`git tag -a v1.0 -m "Release v1.0"`
`git push origin --tags`	Push all tags	`git push origin --tags`

---

10. Collaboration & GitHub Workflow

- **Forks:** Duplicate repositories for personal use.
- **Pull Requests:** Request to merge changes.
- **Code Reviews:** Peer validation before merging.

Team Commands: `git fetch`, `git pull`, `git merge`, `git rebase`

(Include Git flow diagram and VS Code GUI steps)

---

11. Advanced Git Commands

Command	Description	Example
`git cherry-pick`	Apply commit	`git cherry-pick <commit>`
`git bisect`	Find bad commit	`git bisect start`
`git reflog`	Show all refs	`git reflog`
`git blame`	Show who changed lines	`git blame file.txt`

Command	Description	Example
`git submodule`	Manage submodules	`git submodule add <url>`
`git archive`	Create tar/zip	`git archive --format=zip HEAD > repo.zip`
`git gc`	Clean unnecessary files	`git gc`

🗾12. Git Shortcuts & Cheat Sheet

- `init → add → commit → push`
- `branch → checkout → merge → push`
- Quick reference tables and visual flows.

🚃13. Troubleshooting & Best Practices

- Common Git errors and their fixes.
- Use meaningful commit messages.
- Keep `.gitignore` up to date.
- Use aliases for frequent commands (`git config --global alias.co checkout`).

---

14. Appendix

- Glossary of Git terms.
- Links to official docs and tutorials.
- Author credits.

Design Notes

- Clean, professional layout.
- Syntax-highlighted code blocks.
- Visual tables with light borders.
- Minimalist icons and diagrams for clarity.