Git Commands Reference Guide

## Complete Command Reference for Developers

This comprehensive guide provides essential Git commands organized by category for easy reference and learning. Git is the most popular version control system used by developers worldwide.

# Table of Contents

* 1. Setup and Configuration
* 2. Repository Initialization
* 3. Basic Git Workflow
* 4. Branching and Merging
* 5. Remote Repository Operations
* 6. Advanced Commands
* 7. Git Stash Commands
* 8. Git Rebase Commands
* 9. History and Information Commands
* 10. Undoing Changes
* 11. Tagging
* 12. Useful Tips and Best Practices

# 1. Setup and Configuration

Before using Git, you need to configure your identity and preferences.

## Initial Setup Commands

|  |  |  |
| --- | --- | --- |
| Command | Description | Example |
| git config --global user.name "Your Name" | Set your Git username globally | git config --global user.name "John Doe" |
| git config --global user.email "email@example.com" | Set your Git email globally | git config --global user.email "john@example.com" |
| git config --global color.ui auto | Enable colored output | git config --global color.ui auto |
| git config --list | List all Git configuration settings | git config --list |
| git config --get user.name | Get specific configuration value | git config --get user.name |
| git --version | Show Git version | git --version |

# 2. Repository Initialization

Commands to create new repositories or clone existing ones.

## Creating and Cloning Repositories

|  |  |  |
| --- | --- | --- |
| Command | Description | Example |
| git init | Initialize a new Git repository | git init |
| git init <directory> | Initialize Git repository in specific directory | git init my-project |
| git clone <url> | Clone a remote repository | git clone https://github.com/user/repo.git |
| git clone --branch <branch> <url> | Clone specific branch | git clone --branch develop https://github.com/user/repo.git |

# 3. Basic Git Workflow

Core commands used in daily Git operations - the foundation of version control.

## Core Commands for Daily Use

|  |  |  |
| --- | --- | --- |
| Command | Description | Example |
| git status | Show status of working directory | git status |
| git add <file> | Add specific file to staging area | git add index.html |
| git add . | Add all changes to staging area | git add . |
| git add --all | Add all changes including deletions | git add --all |
| git add -p | Add changes interactively | git add -p |
| git commit -m "message" | Commit staged changes with message | git commit -m "Add new feature" |
| git commit -a -m "message" | Commit all tracked files with message | git commit -a -m "Fix bug" |
| git commit --amend -m "new message" | Amend last commit message | git commit --amend -m "Updated message" |
| git diff | Show changes in working directory | git diff |
| git diff --staged | Show staged changes | git diff --staged |
| git diff HEAD | Show all changes since last commit | git diff HEAD |

# 4. Branching and Merging

Git branches allow parallel development. These commands help you manage different lines of development.

## Branch Management Commands

|  |  |  |
| --- | --- | --- |
| Command | Description | Example |
| git branch | List all local branches | git branch |
| git branch -a | List all branches (local and remote) | git branch -a |
| git branch -r | List remote branches | git branch -r |
| git branch <name> | Create new branch | git branch feature-login |
| git branch -d <name> | Delete merged branch | git branch -d feature-login |
| git branch -D <name> | Force delete branch | git branch -D feature-login |
| git branch -m <old> <new> | Rename branch | git branch -m old-name new-name |
| git checkout <branch> | Switch to branch | git checkout main |
| git checkout -b <name> | Create and switch to new branch | git checkout -b feature-signup |
| git switch <branch> | Switch to branch (newer command) | git switch main |
| git switch -c <name> | Create and switch to new branch | git switch -c feature-signup |
| git merge <branch> | Merge branch into current branch | git merge feature-login |
| git merge --no-ff <branch> | Merge with merge commit | git merge --no-ff feature-login |

# 5. Remote Repository Operations

Commands for working with remote repositories - essential for collaboration.

## Working with Remote Repositories

|  |  |  |
| --- | --- | --- |
| Command | Description | Example |
| git remote | List remote repositories | git remote |
| git remote -v | List remotes with URLs | git remote -v |
| git remote add <name> <url> | Add new remote | git remote add origin https://github.com/user/repo.git |
| git remote remove <name> | Remove remote | git remote remove origin |
| git remote rename <old> <new> | Rename remote | git remote rename origin upstream |
| git fetch | Fetch changes from remote | git fetch |
| git fetch <remote> | Fetch from specific remote | git fetch origin |
| git pull | Fetch and merge changes | git pull |
| git pull <remote> <branch> | Pull from specific remote/branch | git pull origin main |
| git pull --rebase | Pull with rebase instead of merge | git pull --rebase |
| git push | Push commits to remote | git push |
| git push <remote> <branch> | Push to specific remote/branch | git push origin main |
| git push -u <remote> <branch> | Push and set upstream | git push -u origin feature-branch |
| git push --all | Push all branches | git push --all |
| git push --force | Force push (use with caution) | git push --force |

# 6. Advanced Commands

Powerful commands for complex Git operations and repository maintenance.

## Powerful Commands for Complex Operations

|  |  |  |
| --- | --- | --- |
| Command | Description | Example |
| git cherry-pick <commit> | Apply specific commit to current branch | git cherry-pick a1b2c3d |
| git bisect start | Start binary search for bug | git bisect start |
| git bisect good <commit> | Mark commit as good | git bisect good HEAD~5 |
| git bisect bad <commit> | Mark commit as bad | git bisect bad HEAD |
| git bisect reset | End bisect session | git bisect reset |
| git blame <file> | Show who changed each line | git blame index.html |
| git grep <pattern> | Search for pattern in files | git grep "function" |
| git reflog | Show reference logs | git reflog |
| git clean -n | Show untracked files to be removed | git clean -n |
| git clean -f | Remove untracked files | git clean -f |
| git clean -fd | Remove untracked files and directories | git clean -fd |

# 7. Git Stash Commands

Temporarily save work in progress without committing. Useful when switching contexts quickly.

## Temporarily Save Work in Progress

|  |  |  |
| --- | --- | --- |
| Command | Description | Example |
| git stash | Stash current changes | git stash |
| git stash push -m "message" | Stash with custom message | git stash push -m "WIP: new feature" |
| git stash -u | Stash including untracked files | git stash -u |
| git stash -a | Stash all files including ignored | git stash -a |
| git stash list | List all stashes | git stash list |
| git stash show | Show summary of stash | git stash show |
| git stash show -p | Show detailed changes in stash | git stash show -p |
| git stash pop | Apply and remove latest stash | git stash pop |
| git stash pop stash@{2} | Apply specific stash | git stash pop stash@{2} |
| git stash apply | Apply stash without removing it | git stash apply |
| git stash apply stash@{1} | Apply specific stash without removing | git stash apply stash@{1} |
| git stash drop | Delete latest stash | git stash drop |
| git stash drop stash@{2} | Delete specific stash | git stash drop stash@{2} |
| git stash clear | Delete all stashes | git stash clear |
| git stash branch <name> | Create branch from stash | git stash branch temp-work |

# 8. Git Rebase Commands

Rewrite commit history and maintain linear development. Powerful but use with caution on shared branches.

## Rewriting History and Linear Development

|  |  |  |
| --- | --- | --- |
| Command | Description | Example |
| git rebase <branch> | Rebase current branch onto another | git rebase main |
| git rebase -i <commit> | Interactive rebase | git rebase -i HEAD~3 |
| git rebase -i --root | Interactive rebase from root | git rebase -i --root |
| git rebase --onto <new-base> <old-base> <branch> | Rebase onto different base | git rebase --onto main feature~5 feature |
| git rebase --continue | Continue rebase after resolving conflicts | git rebase --continue |
| git rebase --abort | Abort rebase and return to original state | git rebase --abort |
| git rebase --skip | Skip current commit during rebase | git rebase --skip |
| git pull --rebase | Pull with rebase instead of merge | git pull --rebase |

## Interactive Rebase Actions

* pick - Use commit as-is
* reword - Change commit message
* edit - Stop to amend commit
* squash - Combine with previous commit
* fixup - Combine with previous commit, discard message
* drop - Remove commit entirely

# 9. History and Information Commands

Explore repository history and get detailed information about commits and changes.

## Exploring Repository History

|  |  |  |
| --- | --- | --- |
| Command | Description | Example |
| git log | Show commit history | git log |
| git log --oneline | Show condensed commit history | git log --oneline |
| git log --graph | Show commits with ASCII graph | git log --graph |
| git log --oneline --graph --decorate | Detailed visual log | git log --oneline --graph --decorate |
| git log -n 5 | Show last 5 commits | git log -n 5 |
| git log --since="2023-01-01" | Show commits since date | git log --since="2023-01-01" |
| git log --author="John" | Show commits by author | git log --author="John" |
| git log --grep="fix" | Search commits by message | git log --grep="fix" |
| git log <file> | Show history for specific file | git log index.html |
| git log --follow <file> | Follow file renames | git log --follow index.html |
| git show <commit> | Show commit details | git show a1b2c3d |
| git show HEAD | Show latest commit details | git show HEAD |
| git shortlog | Summarize commits by author | git shortlog |

# 10. Undoing Changes

Commands to revert, reset, and undo changes safely. Be careful with destructive operations.

## Reverting and Resetting Commands

|  |  |  |
| --- | --- | --- |
| Command | Description | Example |
| git restore <file> | Restore file to last committed state | git restore index.html |
| git restore --staged <file> | Unstage file | git restore --staged index.html |
| git checkout -- <file> | Restore file (older syntax) | git checkout -- index.html |
| git reset <file> | Unstage file | git reset index.html |
| git reset | Unstage all files | git reset |
| git reset --soft HEAD~1 | Undo last commit, keep changes staged | git reset --soft HEAD~1 |
| git reset --mixed HEAD~1 | Undo last commit, unstage changes | git reset --mixed HEAD~1 |
| git reset --hard HEAD~1 | Undo last commit, discard changes | git reset --hard HEAD~1 |
| git revert <commit> | Create commit that undoes another commit | git revert a1b2c3d |
| git revert HEAD | Revert last commit | git revert HEAD |
| git revert --no-commit <commit> | Revert without auto-commit | git revert --no-commit a1b2c3d |

# 11. Tagging

Create and manage tags to mark specific points in history, typically for releases.

## Creating and Managing Tags

|  |  |  |
| --- | --- | --- |
| Command | Description | Example |
| git tag | List all tags | git tag |
| git tag <name> | Create lightweight tag | git tag v1.0 |
| git tag -a <name> -m "message" | Create annotated tag | git tag -a v1.0 -m "Version 1.0 release" |
| git tag <name> <commit> | Tag specific commit | git tag v1.0 a1b2c3d |
| git tag -d <name> | Delete local tag | git tag -d v1.0 |
| git push <remote> <tag> | Push specific tag | git push origin v1.0 |
| git push <remote> --tags | Push all tags | git push origin --tags |
| git push <remote> --delete <tag> | Delete remote tag | git push origin --delete v1.0 |
| git show <tag> | Show tag information | git show v1.0 |

# 12. Useful Tips and Best Practices

## Git Aliases (Add to ~/.gitconfig)

[alias]  
 st = status  
 co = checkout  
 br = branch  
 ci = commit  
 cp = cherry-pick  
 lg = log --oneline --graph --decorate  
 unstage = reset HEAD --  
 last = log -1 HEAD  
 visual = !gitk

## Common Git Workflows

### Feature Branch Workflow

1. git checkout main
2. git pull origin main
3. git checkout -b feature/new-feature
4. Make changes and commit
5. git push -u origin feature/new-feature
6. Create pull request
7. git checkout main
8. git pull origin main
9. git branch -d feature/new-feature

### Hotfix Workflow

1. git checkout main
2. git checkout -b hotfix/urgent-fix
3. Make fix and commit
4. git checkout main
5. git merge hotfix/urgent-fix
6. git push origin main
7. git branch -d hotfix/urgent-fix

## Best Practices

* Commit often with meaningful messages
* Use present tense in commit messages ("Add feature" not "Added feature")
* Keep commits focused on single changes
* Use branches for features and experiments
* Review changes before committing with git diff
* Use git status frequently to understand repository state
* Test code before pushing to shared branches
* Use .gitignore to exclude unnecessary files
* Regular backups by pushing to remote repositories

## Commit Message Conventions

* feat: new feature
* fix: bug fix
* docs: documentation changes
* style: formatting changes
* refactor: code restructuring
* test: adding tests
* chore: maintenance tasks

## Emergency Commands

* git reflog - Find lost commits
* git fsck --full - Check repository integrity
* git gc - Cleanup unnecessary files
* git stash - Quickly save work in progress
* git reset --hard HEAD - Discard all changes (be careful!)

This Git Commands Reference Guide covers the most essential Git commands for developers at all levels. Keep this handy for quick reference during your development workflow.