Git Commands Guide

# Table of Contents

1. git init
2. git add
3. git commit
4. git commit (single file)
5. git status
6. git log
7. git branch
8. git checkout
9. git merge
10. git remote
11. git push
12. git pull
13. git clone
14. git reset
15. git rebase
16. git stash
17. git fetch
18. git config
19. git rm
20. git mv
21. git tag
22. git cherry-pick
23. git revert
24. git archive
25. git describe
26. git diff
27. git log --oneline
28. git diff HEAD
29. git show
30. git reflog
31. git clean
32. git fsck

# Command: git init

Description: Initializes a new Git repository. Purpose: Create a Git repo in the current directory.

When to Use: When starting a new project or versioning an existing one.

Syntax: git init

Example:

mkdir my-new-project

cd my-new-project

git init

# Command: git add

Description: Stages changes (new/modified files) for committing. Purpose: Prepares files for a commit.

When to Use: After modifying or creating files.

Syntax(for single file): git add filename.extension

Syntax(for single folder): git add folderName/

Syntax(for all files in that directory): git add .

Example:

git add src/

git add index.html

git add .

# Command: git commit

Description: Saves staged changes to repository history. Purpose: Create a permanent snapshot.

When to Use: After staging, to record a milestone.

Syntax: git commit -m "Commit message"

Example:

git commit -m "Initial commit"

**Command: git commit (single file)** Description: Commits only a specified file. Purpose: Record a snapshot of a single file.

When to Use: When committing only selected changes. Syntax: git commit <file-path> -m "Commit message" Example:

git commit app.js -m "Update server config"

# Command: git status

Description: Displays working directory and staging area state. Purpose: See what is staged, modified, or untracked.

When to Use: Before committing, to check project status.

Syntax: git status

Example:

git status

# Command: git log

Description: Shows commit history. Purpose: Review project history.

When to Use: To view previous commits.

Syntax: git log git log --oneline

Example:

git log --oneline

# Command: git branch

Description: List, create, or delete branches. Purpose: Manage different work streams.

When to Use: While working on different features.

Syntax: git branch

git branch <branch-name> git branch -d <branch-name> Example:

git branch feature-login git branch -d old-feature

# Command: git checkout

Description: Switch branches or restore files. Purpose: Move between branches/commits.

Why/When to Use: To change context or work on different tasks. Syntax: git checkout <branch-name>

git checkout <commit-hash>

Example:

git checkout main

git checkout 39f3e38

# Command: git merge

Description: Merges two branches.

Purpose: Combine feature changes into main.

When to Use: After completing a feature branch. Syntax: git merge <branch-name>

Example:

git merge feature-login

# Command: git remote

Description: Manages remote repositories. Purpose: Connect to remote servers.

When to Use: To collaborate and push/pull code.

Syntax: git remote add <name> <url>

git remote -v Example:

git remote add origin https://github.com/user/repo.git git remote -v

# Command: git push

Description: Pushes commits to a remote.

Purpose: Update remote with local changes.

When to Use: To publish commits to a team.

Syntax: git push <remote> <branch>

git push -u <remote> <branch> Example:

git push origin main

# Command: git pull

Description: Fetches and merges remote changes.

Purpose: Update your local branch.

When to Use: To get the latest code. Syntax: git pull <remote> <branch> Example:

git pull origin main

**Command: git clone** Description: Clones a repository. Purpose: Copy an existing repo.

When to Use: To start working on a project.

Syntax: git clone <url>

Example:

git clone https://github.com/user/repo.git

# Command: git reset

Description: Unstages or undoes commits. Purpose: Rollback commits/staging.

When to Use: To undo mistakes.

Syntax: git reset <file>

git reset --hard Example:

git reset HEAD~1 git reset --hard

# Command: git rebase

Description: Reapplies commits on another base. Purpose: Clean up history.

When to Use: To make history linear.

Syntax: git rebase <branch> git rebase -i HEAD~N Example:

git rebase -i HEAD~3

# Command: git stash

Description: Temporarily saves changes. Purpose: Save progress without committing.

When to Use: When switching branches quickly.

Syntax: git stash git stash apply Example:

git stash

git stash apply

# Command: git fetch

Description: Fetch updates from remote without merging. Purpose: Review changes before applying.

When to Use: To keep track of remote updates. Syntax: git fetch <remote>

Example:

git fetch origin

# Command: git config

Description: Configure Git settings. Purpose: Set username, email, editor. When to Use: When setting up Git.

Syntax: git config --global user.name "Your Name"

git config --global user.email ["you@example.com"](mailto:you@example.com) Example:

git config --global user.name "John Doe"

git config --global user.email ["johndoe@gmail.com"](mailto:%22johndoe@gmail.com%22)

# Command: git rm

Description: Remove files from working tree and index. Purpose: Delete tracked files.

When to Use: When removing files.

Syntax: git rm <file>

Example:

git rm oldfile.txt

# Command: git mv

Description: Move or rename a file. Purpose: Track file movement.

When to Use: When refactoring files. Syntax: git mv <old> <new>

Example:

git mv oldname.txt newname.txt

# Command: git tag

Description: Create tags for commits. Purpose: Mark specific points in history. When to Use: When making releases. Syntax: git tag <tagname>

Example:

git tag v1.0.0

# Command: git cherry-pick

Description: Apply a commit from one branch to another. Purpose: Transfer specific commits.

When to Use: For isolated bug fixes.

Syntax: git cherry-pick <commit-hash>

Example:

git cherry-pick a1b2c3d

# Command: git revert

Description: Create a new commit that undoes a previous one. Purpose: Undo mistakes safely.

When to Use: Safer than git reset for public branches.

Syntax: git revert <commit-hash>

Example:

git revert a1b2c3d

# Command: git archive

Description: Create a tar or zip archive of files. Purpose: Package code.

When to Use: When creating a snapshot.

Syntax: git archive --format=zip HEAD > latest.zip Example:

git archive --format=tar HEAD > latest.tar

# Command: git describe

Description: Describe a commit using tags. Purpose: Human-readable names.

When to Use: Useful for releases.

Syntax: git describe

Example:

git describe --tags

# Command: git diff

Description: Shows changes between commits, commit and working directory, etc. Purpose: Compare changes.

When to Use: When you want to see the differences between two commits or a commit and the working tree.

Syntax: git diff Example:

git diff HEAD

# Command: git log --oneline

Description: Shows commits in a concise format. Purpose: Compact commit history.

When to Use: For quick view of commit history. Syntax: git log --oneline

Example:

git log --oneline

# Command: git diff HEAD

Description: Shows changes in the working directory compared to the latest commit. Purpose: Track changes not yet committed.

Why/When to Use: For reviewing all changes before committing. Syntax: git diff HEAD

Example: Example: git diff HEAD

# Command: git show

Description: Shows details of a commit. Purpose: Inspect a commit's contents.

When to Use: When you need to view detailed information of a commit.

Syntax: git show <commit-hash>

Example:

git show a1b2c3d

# Command: git reflog

Description: Show the history of references. Purpose: Track changes to the HEAD.

When to Use: To see the history of the HEAD pointer and find commits that are not in the current branch.

Syntax: git reflog Example:

git reflog

# Command: git clean

Description: Remove untracked files. Purpose: Clean up the working directory.

When to Use: To remove files that are not tracked.

Syntax: git clean -fd

Example:

git clean -fd

# Command: git fsck

Description: Verifies the integrity of the repository. Purpose: Checks for errors in the repository.

When to Use: To check for corruption or issues in the repository.

Syntax: git fsck Example:

git fsck

Git Command Options (Details)

1. -m: Provide commit message (git commit)
2. -d: Delete branch (git branch -d)
3. -v: Show detailed info (git remote -v)
4. --oneline: Short format log (git log)
5. --global: Apply settings globally (git config)
6. --hard: Force reset files and commits (git reset)
7. -i: Interactive mode (git rebase -i)
8. -u: Set upstream for branches (git push -u)
9. --format: Define output format for archive (git archive)
10. --tags: Include tags (git describe)
11. --all: Show all refs/branches (git branch --all)
12. --force: Force actions (git push --force, git clean -f)
13. -r: Remote branches (git branch -r)
14. --cached: Remove from index only, not working tree (git rm --cached)
15. --soft: Soft reset to move HEAD (git reset --soft)
16. --keep: Keep local cha