Git Commands & Workflows Complete Guide

Introduction

Git is a distributed version control system designed to track changes in your files and collaborate with others efficiently. This guide covers all essential Git commands with practical examples and real-world workflows.

---

1. Getting Started with Git

1.1 Installation & Configuration

Check Git Version:
git --version

Configure Git (Global Setup):
git config --global user.name "Your Name"
git config --global user.email "your.email@example.com"

View Configuration:
git config --list
git config --global --list

Example:
$ git config --global user.name "John Developer"
$ git config --global user.email "john@example.com"
$ git config --list
user.name=John Developer
user.email=john@example.com

1.2 Create a New Repository

Initialize a Repository Locally:
git init

Initialize in a Specific Directory:
git init my-project
cd my-project

Example:
$ mkdir my-app
$ cd my-app
$ git init
Initialized empty Git repository in /Users/john/my-app/.git/

---

2. Basic Git Workflow

2.1 Understanding Git Areas

[Working Directory] --git add--> [Staging Area] --git commit--> [Repository]

- **Working Directory**: Where you modify files
- **Staging Area**: Where you prepare changes before committing
- **Repository**: Where committed history is stored

2.2 Check Repository Status

Check Current Status:
git status

Example Output:
$ git status
On branch main

Changes not staged for commit:
(use "git add <file>..." to update what will be committed)
(use "git restore <file>..." to discard changes in working directory)
modified: app.py
modified: config.py

Untracked files:
(use "git add <file>..." to include in what will be committed)
README.md
data.txt

2.3 Stage Changes

Add a Specific File:
git add filename.txt

Add All Changes:
git add .
git add --all

Add Changes Interactively:
git add -p

Example:
$ git add README.md
$ git add app.py config.py
$ git add .

2.4 Commit Changes

Commit with Message:
git commit -m "Your commit message"

Commit All Tracked Files:
git commit -am "Your commit message"

Commit with Detailed Message:
git commit -m "Title" -m "Detailed description here"

Example:
$ git commit -m "Add user authentication feature"
$ git commit -am "Fix login bug and update documentation"
$ git commit -m "Add dashboard" -m "Implements new dashboard with analytics"

2.5 View Commit History

View Commit Log:
git log

View Oneline Log:
git log --oneline

View Last N Commits:
git log -n 5

View Commit with Changes:
git log -p

View Commit Graph:
git log --graph --oneline --all

Example:
$ git log --oneline
a3f5k2e (HEAD -> main) Add user authentication feature
b2k9w3r Update configuration file
c1j8v4n Initial project setup

$ git log -n 2
commit a3f5k2e7f8b9c0d1e2f3g4h5i
Author: John Developer john@example.com
Date: Fri Nov 22 10:30:00 2024 +0530

> Add user authentication feature

commit b2k9w3r6s7t8u9v0w1x2y3z
Author: John Developer john@example.com
Date: Fri Nov 22 09:15:00 2024 +0530

Update configuration file

---

3. Remote Repository Management

3.1 Add Remote Repository

Add Remote:
git remote add origin https://github.com/username/repo.git

List Remotes:
git remote
git remote -v

Remove Remote:
git remote remove origin

Rename Remote:
git remote rename origin upstream

Example:
$ git remote add origin https://github.com/john-dev/my-app.git
$ git remote -v
origin https://github.com/john-dev/my-app.git (fetch)
origin https://github.com/john-dev/my-app.git (push)

$ git remote rename origin upstream

3.2 Clone Repository

Clone Repository:
git clone https://github.com/username/repo.git

Clone into Specific Directory:
git clone https://github.com/username/repo.git my-folder

Example:
$ git clone https://github.com/django/django.git
$ git clone https://github.com/torvalds/linux.git linux-kernel
Cloning into 'linux-kernel'...
remote: Enumerating objects: 5000, done.
remote: Counting objects: 100% (5000/5000), done.

3.3 Push & Pull

Push to Remote:
git push origin main

Push All Branches:
git push origin --all

Push Tags:
git push origin --tags

Pull from Remote:
git pull origin main

Pull with Rebase:
git pull --rebase origin main

Fetch Only (No Merge):
git fetch origin

Example:
$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
To github.com:john-dev/my-app.git
a3f5k2e..b2k9w3r main -> main

$ git pull origin main
From github.com:john-dev/my-app.git

- branch main -> FETCH_HEAD
  Already up to date.

---

4. Branch Management

4.1 Create & Switch Branches

List Local Branches:
git branch

List Remote Branches:
git branch -r

List All Branches:
git branch -a

Create New Branch:
git branch feature/user-auth

Switch to Branch:
git checkout feature/user-auth

Create & Switch in One Command:
git checkout -b feature/user-auth

Create and Switch (Modern Syntax):
git switch feature/user-auth
git switch -c feature/user-auth

Example:
$ git branch
feature/dashboard

- main
  development

$ git checkout -b feature/user-auth
Switched to a new branch 'feature/user-auth'

$ git branch

- feature/user-auth
  feature/dashboard
  main

4.2 Delete Branches

Delete Local Branch (Safe):
git branch -d feature/user-auth

Force Delete Branch:
git branch -D feature/user-auth

Delete Remote Branch:
git push origin --delete feature/user-auth

Example:
$ git branch -d feature/user-auth
Deleted branch feature/user-auth (was a3f5k2e).

$ git push origin --delete feature/user-auth
To github.com:john-dev/my-app.git

- [deleted] feature/user-auth

4.3 Rename Branches

Rename Current Branch:
git branch -m new-branch-name

Rename Specific Branch:
git branch -m old-name new-name

Example:
$ git branch -m feature/auth feature/authentication
$ git branch
feature/authentication
main

---

5. Merging Branches

5.1 Merge Branches

Merge Branch into Current Branch:
git merge feature/user-auth

Merge with Commit Message:
git merge feature/user-auth -m "Merge user authentication feature"

Merge with No-FF (No Fast-Forward):
git merge --no-ff feature/user-auth

Example Workflow:
$ git checkout main
Switched to branch 'main'

$ git merge feature/user-auth
Updating a3f5k2e..b2k9w3r
Fast-forward
app.py | 50 ++++++++++++++++++++++++++++++++++++++
1 file changed, 50 insertions(+)

$ git log --oneline
b2k9w3r (HEAD -> main) Add user authentication feature
a3f5k2e Initial setup

5.2 Resolve Merge Conflicts

When Conflicts Occur:
$ git merge feature/branch
CONFLICT (content): Merge conflict in file.txt
Automatic merge failed; fix conflicts and then commit the result.

$ git status
On branch main
You have unmerged paths.
(use "git add <file>..." to mark resolution)
both modified: file.txt

Resolve Conflicts:
Edit the conflicted files to resolve issues, then:

git add resolved-file.txt
git commit -m "Resolve merge conflict in file.txt"

Example Conflict Content:
<<<<<<< HEAD
function main() {
console.log("Version A");
}

function main() {
console.log("Version B");
}

feature/branch

Choose and edit to:
function main() {
console.log("Final merged version");
}

Then commit:
$ git add file.txt
$ git commit -m "Resolve merge conflict"

---

6. Advanced Commands

6.1 Rebase

Interactive Rebase (Last 3 Commits):
git rebase -i HEAD~3

Rebase Current Branch onto Another:
git rebase main

Continue After Conflict:
git rebase --continue

Abort Rebase:
git rebase --abort

Example:
$ git rebase main
First, rewinding head to replay your commits onto main...
Applying: Add user profile feature
Applying: Update documentation

6.2 Cherry-Pick

Apply Specific Commit to Current Branch:
git cherry-pick commit-hash

Cherry-Pick Multiple Commits:
git cherry-pick hash1 hash2 hash3

Example:
$ git log --oneline
a3f5k2e Add feature X
b2k9w3r Fix bug Y

$ git checkout main
$ git cherry-pick a3f5k2e
[main f4g6l3p] Add feature X
1 file changed, 20 insertions(+)

6.3 Stash

Stash Changes:
git stash
git stash save "Work in progress"

List Stashes:
git stash list

Apply Latest Stash:
git stash apply

Apply Specific Stash:
git stash apply stash@{0}

Pop Stash (Apply and Remove):
git stash pop

Delete Stash:
git stash drop stash@{0}

Example:
$ git stash
Saved working directory and index state WIP on main: a3f5k2e Add feature

$ git stash list
stash@{0}: WIP on main: a3f5k2e Add feature
stash@{1}: WIP on main: b2k9w3r Fix bug

$ git stash pop
On branch main
Changes not staged for commit:
modified: app.py
modified: config.py

6.4 Reset

Soft Reset (Keep Changes Staged):
git reset --soft HEAD~1

Mixed Reset (Unstage Changes):
git reset HEAD~1

Hard Reset (Discard All Changes):
git reset --hard HEAD~1

Reset Specific File:
git reset HEAD filename.txt

Example:
$ git log --oneline
a3f5k2e (HEAD -> main) Wrong commit message
b2k9w3r Previous commit

$ git reset --soft HEAD~1
$ git commit -m "Correct commit message"

$ git log --oneline
c4h7m4f (HEAD -> main) Correct commit message
b2k9w3r Previous commit

6.5 Revert

Revert Specific Commit:
git revert commit-hash

Revert without Committing:
git revert commit-hash --no-commit

Example:
$ git log --oneline
a3f5k2e Add buggy feature
b2k9w3r Working commit

$ git revert a3f5k2e
[main c4h7m4f] Revert "Add buggy feature"
1 file changed, 10 deletions(-)

6.6 Diff

Show Changes in Working Directory:
git diff

Show Staged Changes:
git diff --staged

Compare Two Branches:
git diff branch1 branch2

Compare Two Commits:
git diff commit1 commit2

Example:
$ git diff
diff --git a/app.py b/app.py
index a3f5k2e..c4h7m4f 100644
--- a/app.py

▶ b/app.py