Complete Git Guide - Beginner to Advanced

This guide covers everything from basic to advanced Git usage. A few key points to remember:

Start with the basics:

Understanding repositories Basic commits Branching

Progress to intermediate concepts:

Remote repositories Merging strategies Resolving conflicts

Advanced topics for when you're comfortable:

Rebasing Cherry-picking Git hooks Submodules

1. Git Basics

Installation and Setup

```
# Install Git
# For Windows: Download from git-scm.com
# For Mac:
brew install git
# For Linux:
sudo apt-get install git

# Configure Git
git config --global user.name "Your Name"
git config --global user.email "your.email@example.com"
```

Essential Commands

```
# Initialize a repository
git init

# Check status
git status

# Add files to staging
git add filename    # Add specific file
git add .          # Add all files

# Commit changes
git commit -m "Your commit message"

# View commit history
git log
```

2. Branching and Merging

Branch Management

```
# Create new branch
git branch branch-name
```

```
# Switch to branch
git checkout branch-name
# or (newer method)
git switch branch-name

# Create and switch in one command
git checkout -b branch-name
git switch -c branch-name

# List branches
git branch          # Local branches
git branch -r       # Remote branches
git branch -a       # All branches

# Delete branch
git branch -d branch-name    # Safe delete
git branch -D branch-name    # Force delete
```

Merging

```
# Merge branch into current branch
git merge branch-name

# Handle merge conflicts
# Edit conflicted files manually, then:
git add <resolved-files>
git commit -m "Merge conflict resolved"
```

3. Remote Repositories

Working with Remotes

```
# Add remote repository
git remote add origin https://github.com/username/repo.git

# View remotes
git remote -v

# Push to remote
git push -u origin main    # First time
git push                   # Subsequent pushes

# Pull from remote
git pull origin main

# Clone repository
git clone https://github.com/username/repo.git
```

4. Advanced Git Operations

Stashing

```
# Save changes temporarily
git stash

# List stashes
git stash list
```

```
# Apply and remove latest stash
git stash pop

# Apply specific stash
git stash apply stash@{n}

# Drop stash
git stash drop stash@{n}

# Clear all stashes
git stash clear
```

Rebasing

```
# Rebase current branch onto another
git rebase main

# Interactive rebase
git rebase -i HEAD~3    # Rebase last 3 commits

# Continue after resolving conflicts
git rebase --continue

# Abort rebase
git rebase --abort
```

Cherry-picking

```
# Apply specific commit to current branch
git cherry-pick commit-hash

# Cherry-pick without committing
git cherry-pick -n commit-hash
```

5. History and Changes

Viewing History

```
# View commit history with graph
git log --graph --oneline --decorate

# View changes in commit
git show commit-hash

# View file history
git log -p filename

# View who changed what
git blame filename
```

Undoing Changes

```
# Discard changes in working directory
git restore filename
# or
git checkout -- filename

# Unstage files
git restore --staged filename
```

```
# or
git reset HEAD filename

# Reset to specific commit
git reset commit-hash        # Soft reset (keep changes)
git reset --hard commit-hash # Hard reset (discard changes)

# Revert commit
git revert commit-hash
```

6. Git Best Practices

Commit Messages

```
# Good commit message format:
git commit -m "feat: add user authentication

- Add login functionality
- Implement password hashing
- Add session management"
```

Branching Strategy

1. Main/Master: Production code
2. Develop: Integration branch
3. Feature branches: New features
4. Release branches: Version preparation
5. Hotfix branches: Production fixes

Workflow Example

```
# Start new feature
git checkout -b feature/user-auth develop

# Make changes and commit
git add .
git commit -m "feat: implement user login"

# Update with latest develop changes
git checkout develop
git pull
git checkout feature/user-auth
git rebase develop

# Merge feature when complete
git checkout develop
git merge --no-ff feature/user-auth
git push origin develop
```

7. Advanced Git Configurations

Git Aliases

```
# Add to ~/.gitconfig
[alias]
    st = status
    co = checkout
```

```
    br = branch
    ci = commit
    unstage = reset HEAD --
    last = log -1 HEAD
```

Git Hooks

Location: `.git/hooks/`

```
# pre-commit example
#!/bin/sh
npm test
if [ $? -ne 0 ]; then
    echo "Tests failed, commit aborted"
    exit 1
fi
```

8. Git Tools and Integration

Git Large File Storage (LFS)

```
# Install Git LFS
git lfs install

# Track large files
git lfs track "*.psd"

# Verify tracking
git lfs ls-files
```

GitHub CLI

```
# Create pull request
gh pr create

# View pull request
gh pr view

# Merge pull request
gh pr merge
```

9. Troubleshooting Common Issues

Fix Last Commit

```
# Amend commit message
git commit --amend -m "New message"

# Add forgotten files to last commit
git add forgotten-file
git commit --amend --no-edit
```

Clean Repository

```
# Remove untracked files
git clean -n    # Dry run
git clean -f    # Actually remove files
```

```
# Remove untracked files and directories
git clean -fd
```

Recover Lost Changes

```
# Find lost commits
git reflog

# Recover deleted branch
git checkout -b recovered-branch commit-hash
```

10. Advanced Topics

Submodules

```
# Add submodule
git submodule add https://github.com/user/repo

# Initialize submodules after cloning
git submodule init
git submodule update
```

Bisect

```
# Start bisect
git bisect start

# Mark current version as bad
git bisect bad

# Mark last known good version
git bisect good commit-hash

# After testing each version
git bisect good  # or
git bisect bad

# End bisect
git bisect reset
```