Git Guide

Tutorial Resources

- **Beginner Tutorial:** Git and GitHub for Beginners – Crash Course
- **Intermediate Tutorial:** Git Tutorial for Beginners: Learn Git in 1 Hour
- **Advanced Tutorial:** Advanced Git Tutorial

Further Reading

- **Managing Multiple Git Remotes:** Jigarius Blog
- **Corporate Git Workflows:** Git Workflow in Large Companies
- **Miscellaneous Git Tips:** Git Tips and Tricks

Personal Notes

As a Data Scientist, I've adopted **VIM** and **Nano** as my primary terminal editors. I'm also familiar with using **tags** for version-controlling deployed packages.

Over time, I've gained strong proficiency with Git. I implement best practices such as maintaining **linear history**, using **development and deployment branches**, and preferring **rebase over merge** to keep a clean commit tree. Although I do not have team-wide enforcement authority, I consistently advocate for and apply disciplined Git workflows.

I frequently use commands and tools such as `reflog`, `cherry-pick`, and `prune`—practices often overlooked by many, which unfortunately leads to messy repositories.

Git Basics

```
# List all files, including hidden
ls -la

# Initialize Git in current directory
git init

# Check file staging status
git status

# Add all files
git add .

# Add a specific file
git add index.html

# Show all branches (current branch marked with *)
git branch

# Rename current branch
git branch -M main

# Create and switch to a new branch
git checkout -b feature_branch

# Switch to an existing branch
git checkout main

# Make a commit
git commit -m "Short message" -m "Detailed description"

# View remote repositories
git remote -v

# View condensed commit history
```

```
git log --oneline
```

Intermediate Git

Pushing to Remote

Scenario 1: Pushing a local project to a new remote

```
git remote add origin <remote-URL>
git branch -M main
git push -u origin main
```

Scenario 2: Pushing changes after cloning

```
git push -u origin main
```

Feature Branch Workflow

Standard team process:

```
# Push your feature branch
git push -u origin feature_branch

# After pull request is merged remotely
git checkout main
git pull
git branch -D feature_branch
```

For personal projects:

```
git checkout main
git merge feature_branch
git branch -D feature_branch
```

Squashing commits during merge:

```
git checkout main
git merge --squash feature_branch
git branch -D feature_branch
```

Rebasing feature branch into main (preferred for cleaner history):

```
git checkout main
git rebase feature_branch
git branch -D feature_branch
```

Handling Merge Conflicts

```
git checkout main
git pull

git checkout feature_branch
git commit -am "Save current work"
git merge main
# Resolve conflicts manually
git commit -am "Resolved merge conflicts"
```

To abort a conflicted merge:

```
git merge --abort
```

Undoing Changes

1. **Unstage a specific file:**
2. `git reset index.html`
3. **Undo last commit (keep changes):**
4. `git reset HEAD~1`
5. **Undo to a specific commit:**
6. `git reset <commit-hash>`
7. **Hard reset (discard all changes):**
8. `git reset --hard <commit-hash>`

Perfect Commits and Selective Staging

Use `-p` to interactively choose which changes to stage:

```
git add -p index.html
```

Commit message format:

```
Short summary (max 50 chars)

Longer body explaining the context and reasoning.
```

Forking Workflow

Forks are personal copies of repositories. Use them to make changes independently and submit pull requests to the original repository.

Advanced Git

Interactive Rebase

Use for cleaning up local commit history (not for already pushed commits):

```
git rebase -i HEAD~3
# Use editor instructions to pick, squash, reword, etc.
```

Amend the last commit message:

```
git commit --amend -m "Updated commit message"
```

Cherry-Picking Commits

Useful when changes are made on the wrong branch:

```
git checkout feature_branch
git cherry-pick <commit-hash>

git checkout main
git reset --hard HEAD~1
```

Reflog

Use `git reflog` to view the history of HEAD movements—essential for recovering lost commits or rollbacks.