Git & GitHub

COMMAND MASTERY

Your pathway to version control excellence

BEGINNER

Foundation

Repository Setup

```
$  git init
```

Initialize a new Git repository in your project

```
$  git clone <repository-url>
```

Download and copy an existing repository

Identity Configuration

```
$  git config --global user.name "Your Name"
```

Set your global username for all repositories

```
$  git config --global user.email "you@example.com"
```

Configure your global email address

Essential Workflow

```
$  git status
```

Check the current state of your repository

```
$  git add <filename>
```

Stage specific files for the next commit

```
$  git add .
```

Stage all modified files at once

```
$  git commit -m "Your commit message"
```

Save your staged changes with a descriptive message

```
$  git push origin main
```

Upload your commits to the remote repository

```
$  git pull origin main
```

Download and merge changes from remote

> 💡 **PRO TIP**
>
> Always pull before pushing to prevent conflicts. Write meaningful commit messages that clearly describe what changed and why.

INTERMEDIATE

Branch Mastery

Branch Operations

```
$ git branch
```

List all local branches (shows current branch)*

```
$ git branch <branch-name>
```

Create a new branch from current position

```
$ git checkout <branch-name>
```

Switch to an existing branch

```
$ git checkout -b <branch-name>
```

Create and immediately switch to new branch

```
$ git merge <branch-name>
```

Merge specified branch into current branch

```
$ git branch -d <branch-name>
```

Safely delete a merged branch

Remote Management

```
$ git remote -v
```

Display all configured remote repositories

```
$ git remote add origin <url>
```

Connect your local repo to a remote repository

```
$ git fetch origin
```

Download remote changes without merging

```
$ git push -u origin <branch-name>
```

Push branch and set up tracking relationship

History Exploration

```
$ git log --oneline --graph
```

View commit history with visual branch structure

```
$ git diff
```

Show changes between working directory and staging

```
$ git show <commit-hash>
```

Display detailed information about a specific commit

> 💡 **PRO TIP**
>
> Use descriptive branch names with prefixes like "feature/user-login" or "bugfix/navigation-error" to keep your workflow organized and professional.

ADVANCED

Power User Techniques

Time Travel & Undo

```
$ git reset --soft HEAD~1
```

Undo last commit but keep changes staged

```
$ git reset --mixed HEAD~1
```

Undo last commit and unstage changes

```
$ git reset --hard HEAD~1
```

⚠️ *Permanently remove last commit and all changes*

```
$ git revert <commit-hash>
```

Create new commit that undoes previous changes

Stash Management

```
$ git stash
```

Temporarily save uncommitted work

```
$ git stash pop
```

Apply most recent stash and remove from stack

```
$ git stash list
```

Show all saved stashes with descriptions

Advanced Workflows

```
$ git rebase main
```

Reapply current branch commits on top of main

```
$ git rebase -i HEAD~3
```

Interactive rebase to edit last 3 commits

```
$ git cherry-pick <commit-hash>
```

Apply specific commit to current branch

Release Management

```
$ git tag -a v1.0.0 -m "Release version 1.0.0"
```

Create annotated tag for version release

```
$  git push origin --tags
```

Push all tags to remote repository

> 💡 **PRO TIP**
>
> Advanced commands can rewrite history and potentially lose work. Always create backups and work on feature branches when experimenting with these powerful tools.

PLATFORM

GitHub Integration

GitHub CLI Power

```
$ gh repo create <repository-name>
```

Create new repository directly on GitHub

```
$ gh pr create --title "Feature" --body "Description"
```

Create pull request with title and description

```
$ gh pr list --state open
```

List all open pull requests in repository

```
$ gh issue create --title "Bug report"
```

Create new issue with interactive prompts

Fork Workflow

```
$ git remote add upstream <original-repo-url>
```

Add reference to original repository

```
$ git fetch upstream
```

Download updates from original repository

```
$ git merge upstream/main
```

Integrate upstream changes into your fork

Quick Reference

WORKFLOW	COMMAND SEQUENCE
Start new feature	git checkout -b feature/awesome-feature
Save progress	git add . && git commit -m "Add awesome feature"
Update from main	git checkout main && git pull && git checkout feature/awesome-feature
Publish feature	git push -u origin feature/awesome-feature
Clean merge	git checkout main && git merge --no-ff feature/awesome-feature

Professional Guidelines

Commit Style: Use conventional commits (feat:, fix:, docs:, style:, refactor:)
Branch Strategy: Use GitFlow or GitHub Flow for team collaboration
Code Review: Always use pull requests for code integration
Security: Never commit sensitive data like passwords or API keys
Testing: Run tests before committing and ensure CI/CD passes

> 💡 **PRO TIP**
>
> Master Git through consistent practice. Start with basic workflows, gradually incorporate advanced techniques, and always prioritize team collaboration and code quality.