Git Commands Cheat Sheet

Essential Git commands for developers — Setup to advanced workflows

🚀 Setup & Config

COMMAND	DESCRIPTION
`git init`	Initialise new repository
`git clone <url>`	Clone remote repository
`git config --global user.name "…"`	Set username
`git config --global user.email "…"`	Set email
`git config --list`	Show all settings

📝 Basic Workflow

COMMAND	DESCRIPTION
`git status`	Show working tree status
`git add <file>`	Stage specific file
`git add .`	Stage all changes
`git commit -m "msg"`	Commit staged changes
`git commit -am "msg"`	Stage & commit tracked
`git diff`	Show unstaged changes
`git diff --staged`	Show staged changes
`git log --oneline`	Compact commit history
`git log --graph --all`	Visual branch history

🌿 Branching

COMMAND	DESCRIPTION
`git branch`	List all branches
`git branch <name>`	Create new branch
`git branch -d <name>`	Delete branch (safe)
`git branch -D <name>`	Force delete branch
`git checkout -b <name>`	Create & switch
`git switch <branch>`	Switch branch (modern)
`git switch -c <name>`	Create & switch (modern)
`git merge <branch>`	Merge into current
`git merge --no-ff <branch>`	Merge with merge commit

🔄 Common Workflow

```
git checkout -b feature/my-thing
→ make changes
git add . && git commit -m "Add feature"
git push -u origin feature/my-thing
→ open pull request → merge → delete branch
```

🌐 Remote Operations

COMMAND	DESCRIPTION
`git remote -v`	List remotes
`git remote add <name> <url>`	Add remote
`git fetch`	Download remote changes
`git fetch --prune`	Fetch & clean dead refs
`git pull`	Fetch & merge
`git pull --rebase`	Fetch & rebase
`git push`	Push to remote
`git push -u origin <branch>`	Push & set upstream
`git push --force-with-lease`	Safer force push

⏪ Undoing Changes

COMMAND	DESCRIPTION
`git restore <file>`	Discard changes
`git restore --staged <file>`	Unstage file
`git reset --soft HEAD~1`	Undo commit, keep staged
`git reset --mixed HEAD~1`	Undo commit, unstage
`git reset --hard HEAD~1`	Undo commit, discard all
`git revert <commit>`	Create undo commit
`git clean -fd`	Remove untracked files

📦 Stashing

COMMAND	DESCRIPTION
`git stash`	Stash current changes
`git stash save "msg"`	Stash with message
`git stash list`	List all stashes
`git stash pop`	Apply & remove latest
`git stash apply`	Apply without removing
`git stash drop`	Delete specific stash
`git stash clear`	Delete all stashes

📄 .gitignore Essentials

```
node_modules/ — Dependencies
.env — Secrets & API keys
*.log — Log files
.DS_Store — macOS junk
dist/ build/ — Build output
*.pyc __pycache__/ — Python cache
.vscode/ .idea/ — Editor settings
```

🔥 Advanced

COMMAND	DESCRIPTION
`git rebase <branch>`	Rebase current branch
`git rebase -i HEAD~3`	Interactive rebase
`git cherry-pick <commit>`	Apply specific commit
`git reflog`	Show reference log
`git bisect start`	Binary search for bug
`git tag <name>`	Create lightweight tag
`git tag -a <name> -m "…"`	Create annotated tag
`git submodule add <url>`	Add submodule
`git blame <file>`	Show who changed what

🆘 Emergency Commands

SCENARIO	FIX
Wrong branch	`reset --soft HEAD~1` → switch → commit
Undo last commit	`git reset --soft HEAD~1`
Deleted branch	`git reflog` → `checkout -b name <sha>`
Old file version	`git checkout <sha> -- <file>`
Pushed secrets	`reset --hard HEAD~1` → force push ⚠️
Merge conflicts	Edit files → `git add` → `git commit`

⚡ Aliases

```
[alias] in ~/.gitconfig
  st = status · co = checkout · br = branch
  ci = commit · unstage = reset HEAD --
  last = log -1 HEAD
  lg = log --graph --pretty=format:'%h -%d %s (%cr) <%an>' --abbrev-
commit
  undo = reset --soft HEAD~1
  amend = commit --amend --no-edit
```

Pro Tips

- Use `git status` constantly — it's your compass
- Commit early, commit often
- Write clear, descriptive commit messages
- Always pull before you push
- Never force push to shared branches
- Use `.gitignore` from day one
- `git reflog` can save your life
- Learn interactive rebase — it's a superpower
- Use `--force-with-lease` instead of `--force`