**💥 Git Infinity Gauntlet: The Ultimate Cheat Sheet**

**🧱 1. Configuration & Setup**

# Global identity setup

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

git config --global user.email "your@email.com"

# Editor

git config --global core.editor "code --wait"

# View configuration

git config --list

# Alias setup

git config --global alias.co checkout

git config --global alias.st status

git config --global alias.cm commit

**🌱 2. Repository Lifecycle**

# Initialize repository

git init

# Clone existing repo

git clone <url>

# Create new branch

git branch <name>

# Rename branch

git branch -m <old> <new>

# Delete local branch

git branch -d <branch>

# Delete remote branch

git push origin --delete <branch>

# Check current branch

git branch

# List all branches (remote + local)

git branch -a

**📂 3. Staging & Committing**

# Track new files

git add <file>

git add . # all changes

git add -A # add all (even deletions)

# Unstage file

git reset <file>

# Commit

git commit -m "message"

git commit -am "quick commit" # adds + commits

# Amend last commit

git commit --amend

# Partial commit

git add -p

**🌍 4. Remotes & Synchronization**

# Add remote

git remote add origin <url>

# Show remotes

git remote -v

# Remove remote

git remote remove origin

# Rename remote

git remote rename origin upstream

# Push code

git push origin main

# Pull latest changes

git pull origin main

# Fetch only

git fetch origin

# Push new branch

git push -u origin <branch>

**🌳 5. Branching & Merging**

# Switch branches

git checkout <branch>

git switch <branch>

# Create + switch

git checkout -b <new-branch>

git switch -c <new-branch>

# Merge

git merge <branch>

# Abort merge conflict

git merge --abort

**🧼 6. Rewriting History**

# Rebase

git rebase <branch>

# Rebase interactively

git rebase -i HEAD~3

# Skip or continue rebase

git rebase --skip

git rebase --continue

# Squash commits

git rebase -i HEAD~n

# Reset to previous commit

git reset --hard HEAD~1

# Reset softly (keep changes)

git reset --soft HEAD~1

# Reset to specific commit

git reset --hard <commit-hash>

**🧳 7. Stash Commands**

# Save changes temporarily

git stash

# List stashes

git stash list

# Apply latest stash

git stash apply

# Apply + delete

git stash pop

# Delete a stash

git stash drop stash@{0}

# Apply specific stash

git stash apply stash@{2}

**🕵️ 8. Tracking & Diffing**

# File history

git log

git log --oneline

git log --graph --all --decorate

# Check status

git status

# Difference between commits

git diff

git diff <commit1> <commit2>

# Show commit details

git show <commit>

# Blame line-by-line

git blame <file>

**📦 9. Tagging**

# Create tag

git tag v1.0.0

# Annotated tag

git tag -a v1.0.0 -m "Release v1.0.0"

# Push tags

git push origin --tags

# Delete tag

git tag -d v1.0.0

git push origin :refs/tags/v1.0.0

**🧬 10. Cherry-Pick & Revert**

# Pick specific commit

git cherry-pick <commit-hash>

# Revert commit (create opposite commit)

git revert <commit-hash>

# Revert a range

git revert HEAD~2..HEAD

**🔬 11. Advanced Tools**

# Search commit message

git log --grep="keyword"

# Find bug (binary search)

git bisect start

git bisect bad

git bisect good <commit>

git bisect reset

# Clean untracked files

git clean -fd

# Archive repo

git archive --format=zip --output=repo.zip main

**🧠 12. Internals & Debug**

# Inspect objects

git cat-file -p HEAD

# Tree structure

git ls-tree HEAD

# Low-level log

git reflog

# View Git object types

git cat-file -t <hash>

**🚀 13. Open Source & GitHub Flow**

# Fork > Clone > Branch

git checkout -b fix/issue-101

# Link upstream

git remote add upstream https://github.com/original/repo.git

git fetch upstream

git rebase upstream/main

# Push PR

git push origin fix/issue-101

**🔄 14. Submodules & Monorepo**

# Add submodule

git submodule add <repo-url> path/to/dir

# Init/update submodules

git submodule init

git submodule update

# Pull with submodules

git pull --recurse-submodules

# Clone with submodules

git clone --recurse-submodules <repo>

**🧰 15. Git Hooks**

Located in .git/hooks

# Sample hooks:

pre-commit

commit-msg

post-merge

Use [Husky](https://typicode.github.io/husky/) for JavaScript projects:

npx husky-init && npm install

npx husky add .husky/pre-commit "npm test"

**🧪 16. CI/CD with GitHub Actions**

.github/workflows/ci.yml example:

name: CI

on: [push, pull\_request]

jobs:

build:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v2

- run: npm install

- run: npm test

**🧮 17. Semantic Commit Format (Pro Practice)**

feat: Add new expense component

fix: Resolve calculation bug

docs: Update README

style: Format code

refactor: Simplify split logic

test: Add test cases

chore: Lint config updates

**💀 18. Dangerous Commands (Use With Caution)**

# Delete everything untracked

git clean -fdx

# Force push (overwrites history!)

git push -f

# Hard reset (undo commits permanently)

git reset --hard HEAD~1

**📌 Final Tips**

* Use git reflog to rescue lost commits.
* Use git bisect to find breaking changes.
* Always push with -u when setting up a new remote branch.
* Automate with pre-commit hooks and GitHub Actions.
* Clean history before pull requests: squash and rebase.
* Use branches for every task — never commit directly to main.