Git & GitHub Mega Cheat Sheet

1. Git Configuration (One-Time Setup)

Command	Example	Explanation
`git config --global user.name`	`git config --global user.name "Alice"`	Set your name for all local Git use
`git config --global user.email`	`git config --global user.email "a@mail.com"`	Set your email
`git config --list`	`git config --list`	View current Git configuration

2. Repository Basics

Command	Example	Explanation
`git init`	`git init`	Initialize a new local Git repo in your folder
`git clone <url>`	`git clone https://github.com/user/repo.git`	Copy a GitHub repo locally
`git remote add origin <url>`	`git remote add origin <url>`	Link your repo to a remote (GitHub) location
`git remote -v`	`git remote -v`	List remote URLs set for this repo

3. Working with Changes

Command	Example	Explanation
`git status`	`git status`	See which files are changed/staged
`git add <file>`	`git add readme.md`	Stage specific file for commit
`git add .`	`git add .`	Stage all changed files
`git commit -m "msg"`	`git commit -m "Initial commit"`	Commit staged changes locally

4. Daily Workflow (Step-By-Step)

A. Starting a New Repo & Pushing to GitHub

1. **Initialize locally:**

```
git init
```

2. **Add files and commit:**

```
git add .
git commit -m "Initial commit"
```

3. **Create remote repo on GitHub (web UI), then link:**

```
git remote add origin https://github.com/username/repo.git
git branch -M main               # Optional: set main branch as 'main'
git push -u origin main
```

B. Clone Existing Repo and Contribute

1. **Clone:**

```
git clone https://github.com/username/repo.git
cd repo
```

2. **Create a new branch for changes:**

```
git checkout -b feature-branch
```

3. **Work, then stage and commit:**

```
git add .
git commit -m "describe your work"
```

4. **Push and sync branch:**

```
git push -u origin feature-branch
```

5. **On GitHub, create a Pull Request (PR) for your branch to `main`.**

5. Branching & Merging

Command	Example	Explanation
`git branch`	`git branch`	List all branches

`git branch <branch>`	`git branch dev`	Create a new branch
`git checkout <branch>`	`git checkout dev`	Switch to branch
`git checkout -b <branch>`	`git checkout -b dev`	Create & switch to new branch
`git merge <branch>`	`git merge dev`	Merge dev into current branch
`git branch -d <branch>`	`git branch -d dev`	Delete branch (locally, if merged)

6. Pulling, Pushing & Keeping Synced

Command	Example	Explanation
`git push origin <branch>`	`git push origin main`	Push changes to GitHub (remote)
`git pull origin <branch>`	`git pull origin main`	Fetch and merge new changes from remote
`git fetch origin`	`git fetch origin`	Fetch changes without merging

7. Viewing and Undoing History

Command	Example	Explanation
`git log`	`git log --oneline`	Show commit history (short form)
`git diff`	`git diff`	See unstaged file differences
`git diff --staged`	`git diff --staged`	Show staged changes
`git reset HEAD <file>`	`git reset HEAD file.py`	Unstage a file
`git checkout -- <file>`	`git checkout -- myscript.py`	Discard local changes in that file
`git revert <commit>`	`git revert abc1234`	Undo a commit (preserve history)
`git reset --hard <commit>`	`git reset --hard HEAD~1`	Hard reset to previous commit (dangerous!)

8. Advanced Workflow: Stashing & Collaboration

Command	Example	Explanation

`git stash`	`git stash`	Save local uncommitted changes temporarily
`git stash pop`	`git stash pop`	Restore last stashed changes
`git stash list`	`git stash list`	View stashed changes
Open PR on GitHub (Web)	(GitHub UI)	Invite project maintainers to review/merge

9. Cleaning Up

Command	Example	Explanation
`git rm <file>`	`git rm data.csv`	Remove a file from working dir and staging area
`git mv <old> <new>`	`git mv old.py new.py`	Move/rename a file inside the repo
`git clean -f`	`git clean -f`	Remove untracked files from working directory

10. Troubleshooting & Help

- **Show help for any command:**

```
git <command> --help       # Example: git commit --help
```

- **Abort a problematic merge:**

```
git merge --abort
```

- **View remote branch info:**

```
git branch -r
```

Typical GitHub Feature Branch Workflow Example

1. Clone repo

```
git clone <repo_url>
cd repo_name
```

2. Create new branch

```
git checkout -b my-feature
```

3. Make and commit changes

```
# Edit files
git add .
git commit -m "Added awesome feature"
```

4. Push to GitHub

```
git push -u origin my-feature
```

5. Go to GitHub and open a Pull Request
6. After PR is merged, update local `main`:

```
git checkout main
git pull origin main
```

7. Delete feature branch (optional):

```
git branch -d my-feature
git push origin --delete my-feature
```

Quick Reference Table

Purpose	Command
Init new repo	`git init`
Clone repo	`git clone <url>`
Stage changes	`git add .` or `git add <file>`
Commit	`git commit -m "msg"`
Push to GitHub	`git push origin <branch>`
Pull from GitHub	`git pull origin <branch>`
Create branch	`git checkout -b <branch>`
Merge branch	`git merge <branch>`
See status	`git status`
See commit history	`git log --oneline`
Discard changes	`git checkout -- <file>`
Unstage a file	`git reset HEAD <file>`
Remove file from repo	`git rm <file>`