# 🧭 GitHub Workflow Cheatsheet

A concise guide for managing a **professional Git workflow** using **VS Code's built-in Git tools** — no terminal required.

---

## 🚀 Branching Model

### 💈 Main branch rules

- `main` = production / stable code.
- Never commit directly to `main`.
- Protect it on GitHub (Settings → Branches → Protection Rules).

### 💈 Create new branches for all work

- Click branch name (bottom left in VS Code) → `Create new branch`
- Use clear naming conventions:
- `feature/<name>` – for new features
- `fix/<name>` – for bug fixes
- `chore/<name>` – for cleanup or config changes

**Example:**

```
feature/qdrant-upsert
fix/timeout-error
chore/update-readme
```

---

## 🏡 Daily Development Loop

1. **Pull latest main**

2. In VS Code: Source Control → "Pull" or "Sync Changes".

3. **Create a new branch from main**

4. Branch → `Create New Branch` → choose descriptive name.

5. **Work on your code**

6. Add/modify files as needed.

7. **Stage only relevant files**
8. Review changes in *Source Control* panel.
9. Click `+` to stage specific files.
10. **Commit small, clear changes**
11. Use Conventional Commit style:
    - `feat: add upload_points batching`
    - `fix: handle write timeout`
    - `chore: update .gitignore`
12. **Push your branch**
13. First push → "Publish Branch".
14. **Open a Pull Request (PR)**
15. In VS Code or on GitHub.
16. Optional: mark as *Draft* while still developing.
17. **Merge via PR**
18. Prefer **Squash & Merge** for a clean history.
19. Delete the branch after merging.
20. **Return to main and pull again.**

---

## 🚉 Keeping Branches Fresh

- Regularly **pull main** and **rebase/merge** into your branch before PR.
- In VS Code: Branch menu → "Rebase current branch onto main".

---

## 🚨 Hotfixes

- Create a quick branch: `fix/<short-desc>` from `main`.
- Make the fix → commit → push → PR → merge → delete branch.

---

## 🛖Useful Git Hygiene

`.gitignore`

Create once in project root:

```
.venv/
.env
*.env
__pycache__/
.ipynb_checkpoints/
.DS_Store
.pytest_cache/
.mypy_cache/
```

### Secrets

- Store API keys in `.env`, **never** commit them.
- Add `.env` to `.gitignore`.

### Stash WIP

If you need to switch tasks quickly: - Source Control → `⋯` → `Stash Changes` - Later: `Apply Stash`

---

## 🛖Versioning (Optional)

Use **semantic versioning**: - `feat` → bump **minor** (`v0.2.0` → `v0.3.0`) - `fix` → bump **patch** (`v0.2.0` → `v0.2.1`)

Tag releases:

```
v0.1.0
v0.2.0
```

---

## TL;DR Flow

```
1️⃣ Pull main
2️⃣ Create new branch (feature/...)
3️⃣ Code → Stage → Commit small
4️⃣ Push branch
```

```
5️⃣ Open PR → Review → Squash & Merge
6️⃣ Delete branch
7️⃣ Pull main again
```

---

**Pro tip:** Think of GitHub like a *conversation about code*. Branches are your discussion topics, commits are your sentences, and PRs are the review meeting.