Professional GitHub Workflow Guide (Agile Solo Project)

Overview

- This guide explains a professional GitHub workflow using feature branches, dev, and main branches.
- It's designed for solo developers building backend projects with Agile methodology.
- You build features in isolation, merge them step-by-step, and deploy from the main branch.

Step-by-Step Workflow

- 1. Start from the dev branch:
-    git checkout dev
-
- 2. Create a new feature branch:
-    git checkout -b feature/your-feature-name
-
- 3. Build your feature (write code + tests).
-    Use clear, meaningful commits:
-    git commit -m 'Implement X feature with unit tests'
-
- 4. Push your feature branch to GitHub:
-    git push origin feature/your-feature-name
-
- 5. Open a pull request on GitHub (base: dev, compare: feature).
-    Describe what you did, why you did it, and what to review.
-
- 6. Review your own PR (solo dev).
-    Confirm tests pass, logic is clean.
-    Click 'Merge' into dev.
-
- 7. Repeat this for every feature in your sprint (auth, project model, etc).
-
- 8. When sprint ends, open a PR from dev to main.
-    Title: Sprint Summary
-    Description: List merged features + test summary.

- Click 'Merge' into main.
-
- 9. Delete merged feature branches to keep your repo clean:
- git branch -d feature/your-feature-name
- git push origin --delete feature/your-feature-name

Optional Advanced Features (Add Later)

- 1. GitHub Actions CI/CD: Automatically test and deploy code when PRs are created or merged.
- 2. PR Checklists: Use PR templates with steps like: tested, reviewed, passed CI, etc.
- 3. Branch Protection Rules: Prevent direct pushes to main/dev, require PR review.
- 4. GitHub Releases: Tag your main branch with release versions (v1.0.0, etc).
- 5. Multiple Contributors: Let collaborators create feature branches and open PRs.
- 6. Automated Linting/Formatting: Run tools on every push to enforce code quality.