Contextual GitHub Workflows with Cursor Commands

Jack Taylor

Cursor Meetup

19th of November, 2025

Introduction

- At a medium-sized company with many projects
- In a team of eight developers
- Mainly full stack web applications

Question

Have you had to follow up on or call about a code review, or pull request, in the last two weeks?

Problem

- GitHub’s bug and issue tracking was underutilised
- Some pull requests took days or weeks to approve
- Context and information was missing
- Delays increased merge conflicts and slowed feature development
- Sometimes lack of documentation/teammate availability could cause problems

Solution

- I’ve used Cursor Commands to speed up issue and pull request creation
- They utilise our team’s templating and repository standards
- Automating approximately ten minutes worth of writing each time

Demo

Structure

- Any text after the command is passed as context
- Breaks down problem into individual steps
- Git Commands are already hardcoded to keep model on track
- Utilises diffs, commit messages, and branch information for even more context
- GitHub markdown formatting encouraged

Design Choices

- GitHub MCP Server was used because of development restrictions
- Cursor Rules weren’t as clean as commands/team commands
- Cursor Commands were most consistently working
- Security of MCP is easier to audit/control than CLI, and responses are made for LLMs

Results

- Developers report higher satisfaction due to less time writing
- Information and context in pull requests and issues higher
- Average number of comments on issue or pull request has significantly decreased

Future Extensions

- Launching a coding agent after issue creation
- Writing documentation after creating pull request
- Checking GitHub Actions status and fixing bugs based on CI/CD pipeline results

Questions

- Any questions?

Thank You