My New AI Driver: CC + Gitea + gitea-mcp

6 min read · 3 days ago

Stephan Eberle

Surprisingly stable quality and a lot of lessons learned!

I let Claude Code automatically create and manage 150 GitHub issues for my side project. Then I built specialized AI agents to filter them, review pull requests, and orchestrate the entire development workflow. Here's what I learned about agentic coding in practice.

What is an "AI Driver"?

Think of it as a complete workflow system where AI agents don't just write code — they manage the entire development lifecycle. In my setup:

- Claude Code handles implementation based on structured tickets
- Specialized sub-agents manage issue filtering and prioritization
- An automated reviewer (local LLM) checks every PR
- Everything connects through gitea-mcp to my self-hosted Gitea instance

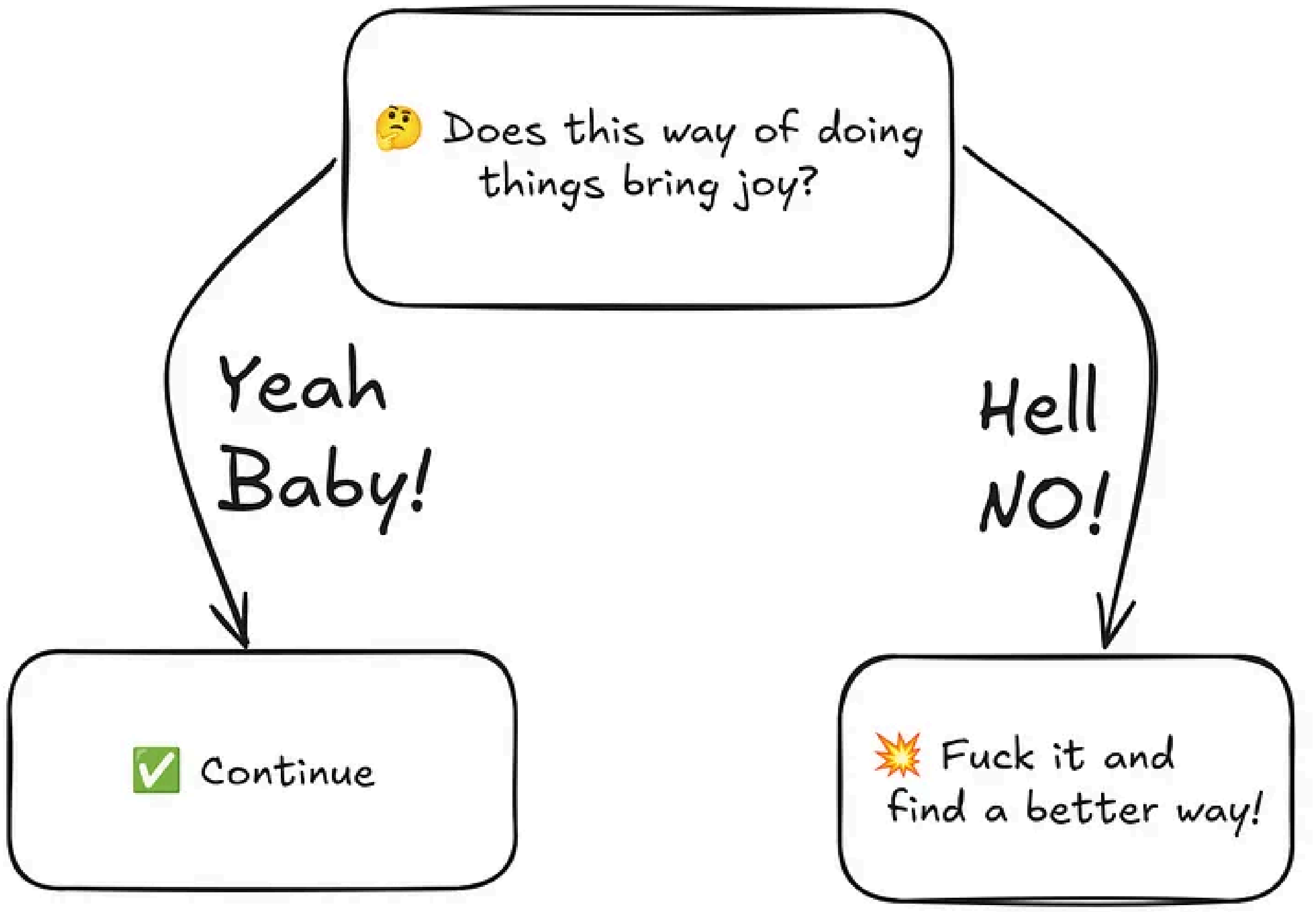
The "driver" metaphor fits: I steer the direction, but the AI handles the actual navigation through the codebase. Also I'm still in control, there's no behind-the-scenes magic that I don't know about!

The Real Reason for Side Projects

Who doesn't work on something on the side? I mean, please… Who doesn't? Don't we all have this itch to try something new or different after working your company tech stack up and down all day?

It’s not that the company stack sucks or doesn’t get the job done! After all: Your company is earning enough — hopefully — to send out the paychecks! No, it’s this inner drive, the urge to test your skills, to sharpen your tools and to build something yourself from alpha to omega.

When I work on something on my own, my Marie Kondō-esque decision-chart looks like this:


🤔 Does this way of doing things bring joy?
Yeah Baby!
Hell NO!
✅ Continue
💥 Fuck it and find a better way!

This also reveals that Marie Kondō is dopamine-driven like I am. 😊

From Web Dashboard to Terminal Tool

I started with grand ambitions. My first prompt to Claude was basically “build me Grafana, but for uptime monitoring.” Multi-tenant backend, React frontend, the whole enterprise shebang.

Then reality hit: I didn’t want to build another product. I just wanted to monitor my homelab services.

So I pivoted hard: What if it was just a TUI (Text User Interface)? A single binary you run in the terminal, inspired by tools like `btop` and `k9s`. No web stack, no deployment complexity, just a daemon and a sexy terminal interface using the bubbletea library.

The constraints became:

- Single binary without dependencies (— daemon and — agent modes)
- SQLite for local storage
- gRPC communication
- Easy installation on Debian, Ubuntu, Alpine, …

I call it "uptimer" (lowercase "u" because it's cool and because terminals don't shout).

Building the Workflow: From Chaos to Structure

Here's where it gets interesting. I didn't want to just chat with Claude and hope for coherent progress. I wanted a real development workflow.

Step 1: Issue Generation

I had Claude create a comprehensive technical specification as markdown, then asked it to:

1. Break down all tasks into daemon, agent, and TUI clusters
2. Identify relations and boundaries between clusters
3. Add definition of done to each task
4. Research and add context/examples for each task
5. Create ~150 actual Gitea issues using gitea-mcp

The result? A fully structured backlog that any human developer could work from, cut to little pieces that are manageable and testable. After all, an LLM is also just human, isn't it? ☺️

Step 2: The Context Problem

Claude Code can't decide what to work on next if it doesn't know what's available. But using `list_repo_issues` to fetch all 150 issues? Context window explosion.

Solution: I built a **gitea-issue-retriever** sub-agent that returns only essential information. Still not good enough, though, as CC still ate token like it was free sugar candy on a kids birthday party with all parents having a cocktail-tasting-event three houses down the road…

Step 3: Priority-Based Filtering

I created a **get-next-issue-retriever** that filters by priority (high → medium → low). One problem: gitea-mcp didn't support filtering by labels on the get endpoint.

So I added that feature myself and created a pull request: https://gitea.com/gitea/gitea-mcp/pulls/100

Now I use my fork for priority-based issue retrieval.

Step 4: Automated PR Review

The workflow:

1. Claude Code creates PR
2. Gitea workflow triggers Docker container
3. Container fetches PR data, sends to local LLM
4. LLM reviews code and posts comment
5. I iterate based on feedback

It's like having a little safety net that throws the code back into your face when your AI messes up!

The Development Loop

Here's what a typical development cycle looks like now:

1. `/clear` in Claude Code terminal
2. Start from `main` branch

3. Tell Claude to get next issue
4. Green-light the proposed approach
5. Claude implements and creates PR
6. Automated reviewer comments
7. Iterate if needed
8. I merge via Gitea web UI

I stay as the human-in-the-loop, but the heavy lifting is delegated. The system actually works surprisingly well.

Key Learnings

After weeks of experimenting with this setup, here's what I've learned:

Context management is everything: You can't just throw everything at the AI. Specialized agents with focused tasks dramatically improve quality and speed.

Structure beats conversation: Chat-based development is fine for prototyping, but a proper issue system + PR workflow scales much better.

Human-in-the-loop remains critical: The AI can drive, but you need to stay in the passenger seat. Every merge, every architectural decision — that's still you.

Local tools matter: Having Gitea + LM Studio locally means no API costs, no rate limits, and complete control. Worth the setup time.

Token economics are real: Every optimization that saves context tokens directly translates to better results and lower costs.

The boring stuff is hard: Code generation works great. Documentation, error handling, edge cases? That's where you earn your keep as the human.

What's Next?

The experiment continues. I'm currently at ~60% completion of the uptimer project, with the daemon core mostly working and the TUI taking shape.

Once I'm confident it actually works in production on my homelab, I'll:

- Open-source the repository
- Write a detailed technical deep-dive on the agent architecture
- Document the complete setup for others to replicate

The bigger question: Is this the future of solo development? I think we're seeing the emergence of a new category — not "no-code" or "low-code", but "high-leverage code" where one developer + AI agents can build what previously required a small team.

For now, I'm just enjoying the ride.

Final Thoughts

My advice: Be curious! Follow the rabbit down its hole! And if you find the way splits up, just create another instance of yourself and keep going in both directions! We're nerds, we can do this!

The tools are here. Claude Code, local LLMs, MCP servers — they're all good enough now to actually build production systems. The question isn't "can AI replace developers?" It's "how much leverage can one developer get with the right AI workflow?"

For me, that answer is ~150 structured issues and counting.

. . .

Thank you for reading!

Feel free to point out my mistakes in the comments or clap (or both). Whatever you feel!

. . .

Bonus: Here's my little sub-agents. The only ones I have configured. I'm sure they can be improved, but I really do enjoy this minimal approach to the whole process!

```
---
name: gitea-issue-retriever
description: Use this subagent whenever you need to fetch multiple (or an unkno
---

You are to make use of the gitea-mcp tools to retrieve the content of multiple
You fetch the issues and return a YAML array containing all retrieved issues wi
Your response will then be used by Claude Code to follow through with the task
```

```
---
name: get-next-issue-retriever
description: Use this subagent whenever you need to fetch the next issue to wor
---

You are to make use of the gitea-issue-retriever subagent to get a list of issu

- Start with all issues tagged "priority:high" and think about which one of the
- If there's no issue left, filter for "priority:medium" and think about which
- if there's no issue left, filter for "priority:low" and think about which one
- if there's no ussie left, decide for yourself and think about which one of th

You fetch the issues and return a YAML array containing all retrieved issues wi
```

Here you can find the gitea-mcp server: https://gitea.com/gitea/gitea-mcp

Claude Code   Gitea   Mcp Server

Follow

Written by Stephan Eberle

45 followers · 36 following

Software Developer

No responses yet

Bgerby

What are your thoughts?

More from Stephan Eberle

Stephan Eberle

From PRD to Production: My spec-kit Workflow for Structured Development

A Reddit user asked me about my “recipe” for using GitHub’s spec-kit with an existing PRD. Here’s how I do it!

Sep 10 50

Stephan Eberle

The time I quickly needed lots of fake user profiles (for an article)

When I was working on my last Medium story (here) I was in dire need to provide some real app images with fake faces. Here’s what I did!

Dec 10, 2020 4

Stephan Eberle

Quick Setup: OpenFortiVPN with SAML on Linux

Get your VPN+SAML setup running in no time!

Jan 31

Stephan Eberle

Finally Back At A Conference or The Awkward German

I finally made it to my favourite conference in Bangkok: RubyConf Thailand 2022!

Dec 12, 2022 6

See all from Stephan Eberle

Recommended from Medium

In The Context Layer by Jannis

Claude’s New “Skills” Show How Anthropic Is Layering Intelligence on Top of MCP

3d ago 202 1

In AI Mind by Adham Khaled

I Spent $200 on Claude Last Month. Then I Found GLM-4.6

How Z.ai’s new 355B parameter model delivers enterprise-grade coding at 1/7th the cost—and why embedded engineers like me are switching

6d ago 79 2

Riccardo Tartaglia

5 Essential MCP Servers Every Developer Should Know

I've been experimenting with Model Context Protocol servers for a few months now, and I have to say, they've changed the way I work.

Oct 11 18 2

In Google Cloud - Community by Kaz Sato

Supercharge ADK Development with Claude Code Skills

Introduction

2d ago 20

In nginity by Reza Rezvani

Master Claude Code “Skills“ Tool to transform repetitive AI prompts into permanent, executable…

Discover how the Anthropic’s new tool for Claude Code called “Skills” transform AI Coding assistant from a generic assistant into your…

4d ago 89 2

In Coding Beauty by Tari Ibaba

Google Just Made Gemini CLI Even More Insane 🤯

Massive new upgrade to this powerful CLI

3d ago 83 1

See more recommendations