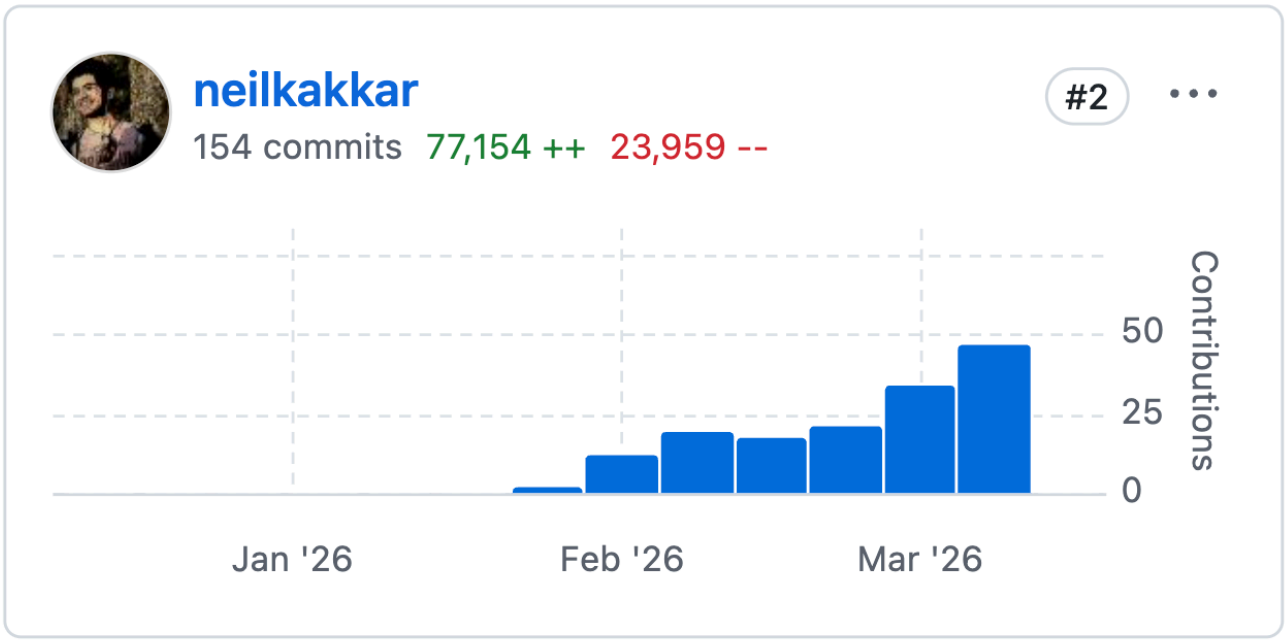
How I'm Productive with Claude Code

Mar 16, 2026 • Tech

It's been about 6 weeks since I joined Tano, and this is what my commit history looks like:



Commits are a terrible metric for output, but they're the most visible signal I have. Something real changed in how I work, and the commit count is a side effect.

So, what has changed?

Automating the grunt work

When I joined Tano, I was making every pull request by hand. Stage changes, write the commit message, craft the PR description, push, create the PR on GitHub. Standard process, it was *fine*.

It took me a while to realize this is grunt work. I was so used to doing it that I'd never questioned it.

That was the first real shift: I'm not the implementer anymore. I'm the manager of agents doing the implementation. And managers automate their team's grunt work.

Then I wrote my first Claude Code skill: `/git-pr`.[1]

It does everything I used to do, except it does it better. The PR descriptions are more thorough than what I'd write, because it reads the full diff and summarises the changes properly. I'd gotten so used to the drudgery that I'd stopped noticing it was drudgery.

The time saved matters, but the real unlock was the mental overhead removed. Every PR used to be a

Killing the wait

Reviewing changes had this annoying loop.

Preview changes locally, go away from what I'm working on, kill the dev server, restart it on the new branch, check it all works, review the code.

The server build took about a minute, which was agonisingly long when I was mid-context-switch. Long enough to break focus, too short to do anything useful.

I switched the build to SWC, and server restarts dropped to under a second. This sparked joy.

It sounds like a small change. It wasn't. Sub-second restarts mean you never leave the flow. Save a file, the server's already up, check the preview. There's no gap where your attention drifts. It's the difference between a conversation with awkward pauses and one that flows naturally.

Let Claude see what happens

Before this, I checked every UI change. Preview locally, eyeball it, decide if it matches what I expected. It worked, but it meant I was a bottleneck on every feature.

After the Chrome extension kept crashing, I switched to the preview feature in Claude Code. It lets the agent set up a preview, persist session data, and see how the UI actually looks.

I wired it into the workflow: a change isn't "done" until the agent has verified the UI itself. That meant I could delegate verification and only step in for final review — which also meant agents could run much longer without oversight. They'd catch their own mistakes. That mattered more than I realized at the time.

Parallel everything

Fast rebuilds and automated previews made another friction visible: I could only comfortably work on one thing at a time.

I was reviewing PRs from other agents and teammates. The workflow was painful: check out the PR branch on main, rebuild, test. But that would mess with my uncommitted changes. So I'd stash, checkout, rebuild, test, switch back, pop the stash. Or create a worktree manually, set it up, try to run the preview - only to find the ports clashing with my other running server.

Our app has a frontend and a backend, each needing its own port. Every worktree shared the same environment variables, so they'd all try to bind to the same ports. Running two things at once was a fight.

I built a system around this. Whenever a worktree is created, every server gets assigned ports from a unique range. No collisions. I could run ten previews simultaneously if I wanted.

only stop once they'd verified the UI themselves.

I'd be heavily involved in planning. Then I'd disappear until code review. Agents catching their own mistakes mattered a lot more with five running at once.

Reviewing got smoother too. No faffing around with setup. No rebuilding. No port conflicts. Just: read, verify, merge. Next.

It's the infrastructure, not the AI

My role has changed. I used to derive joy from figuring out a complicated problem, spending hours crafting the perfect UI. I still do that sometimes, but a lot less now. What's become more fun is building the infrastructure that makes the agents effective. Being a manager of a team of ten versus being a solo dev. And like any good manager, you get to claim credit for all the work your "team" does.

These aren't glamorous problems. They're plumbing. But plumbing determines whether you're in flow or wrestling your environment.

The highest-leverage work I've done at Tano hasn't been writing features. It's been building the infrastructure that turned a trickle of commits into a flood.

The loop

Each of these stages removed a different kind of friction:

1. `/git-pr` removed the friction of **formatting** - turning code changes into a presentable PR.
2. SWC removed the friction of **waiting** - the dead time between making a change and seeing it.
3. The preview removed the friction of **verifying changes** - I could quickly see what's happening.
4. The worktree system removed the friction of **context-switching** - juggling multiple streams of work without them colliding.

And each time I removed one, the next became visible. When PRs were effortless, I noticed I was wasting time on rebuilds. When rebuilds were instant, I noticed I couldn't run things in parallel. Classic theory of constraints — fix one, and the system immediately shows you the next one.

The nature of the work changed. I'm not "using a tool that writes code." I'm in a tight loop: kick off a task, the agent writes code, I check the preview, read the diff, give feedback or merge, kick off the next task. The feedback loop is so tight that there's no gap for my attention to leak out.

Building things is a different kind of fun now — it's so fast that the game becomes improving the speed. How much faster can I go? When the loop is tight enough, engineering becomes the entertainment.

1. It's /git-pr because our codebase CLAUDE.md says to use Graphite, but I prefer plain git. ↩

- How to simulate a broken database connection for testing in Django

Subscribe to the Idea Muse

One idea every few weeks designed to make you smarter. Join 1,500+ people cutting through the noise.
Read more

Email Address *

First Name *

Subscribe

Neil Kakkar
Write (Code). Create. Recurse.
neil@neilkakkar.com

I want to understand how the world works. This blog tracks my growth, the things I've learned, and how I'm leveraging them to do epic things.