Fable 5 is the best model ever released. Here's how I'm using it without going broke.

A short note on the workflow I've been living in all week, and the bit of reasoning behind it that's quietly saving me a fortune in tokens.

So Fable 5 is out. If you've been near my feed today you already know where I land on it: I think it's the most capable thing the public has ever been handed. Same Mythos family that had governments nervous earlier this year, and you can feel it. Top of nearly every benchmark. The catch is the price. It's roughly twice what Opus 4.8 runs, and if you just point it at everything all day, you'll hit your limits by lunch and have no idea where they went.

So I stopped doing that. Here's the thing nobody really says out loud: maybe 20% of your work actually *needs* the smartest model in the room. The other 80% is grunt work and a cheaper model handles it fine. Once that clicked, my whole setup changed. This next bit is the part I keep coming back to.

The 10 / 80 / 10 rule (this is the one)

Here's the thinking before I give you the steps, because the steps only make sense if you get this first. Fable is brilliant at exactly two things in a build: thinking through the plan before you start, and catching what's broken at the end. Those are the moments where a smarter model genuinely changes the outcome. Everything in between is just typing out code you've already decided on, and that's where tokens go to die if you're running the expensive model. So the whole rule is built around one idea: spend Fable only where its brain actually pays off, and never on the grunt work.

That gives you three phases, and I switch models on purpose between them:

First 10%, plan it in Fable. Before a single line gets written, I sit with Fable and architect the thing properly. Data models, file structure, the edge cases I know will bite me later. This is where its planning really shows, and getting it right means the next 80% basically writes itself.

Middle 80%, build it in Opus 4.8. The grunt work. Components, wiring, boilerplate, the boring refactors. There is no reason to burn premium tokens having a genius model type out code you've already planned. Opus does it just as well for half the cost, and this is the bulk of the job.

Last 10%, review it in Fable. Then I bring Fable back for the final pass to make sure the thing was actually executed properly. The nasty bugs, the logic that looks fine but isn't, the security review. This is the other moment its brain earns the spend.

So look at what you've actually done. You've taken the two best parts of Fable, the planning and the final review, and used it only there. The expensive, repetitive middle never touches it. You get the smartest model in the room making the decisions that matter and checking the work at the end, without paying premium rates to watch it type. That's the entire point. Best parts of Fable, none of the wasted tokens.

> *Plan in Fable. Build in Opus. Review in Fable. You're paying for its brain, not its typing.*

Where each one lives in my day

Outside of coding it's the same logic. Fable comes out when being wrong is expensive: business strategy, the big calls, anything mission-critical, the deep coding and refinement work. The hard stuff I'd happily pay double to get right the first time, basically.

Opus 4.8 is my daily driver for everything else, which is most of it. Creative work, throwing ideas around, the bulk of my chatting, first drafts, quick research, and all the execution once a plan already exists. I open Opus by default and only reach for Fable when something actually deserves it. Honestly that one habit is the whole difference between torching your usage and not.

Quick heads-up so it doesn't catch you out: in high-risk areas like cyber, bio and chemistry, Fable just blocks and falls back to Opus 4.8. That's by design, not a glitch, the guardrails doing their job. And if you saw "Mythos" thrown around today, that's the locked-down partner-only one. Fable 5 is the version you and I can actually use.

Give the 10/80/10 thing a go on your next build. Simplest change I've made in months and it's saved me more than anything fancy ever did. Let me know how it lands.

Miles