THE FLAT-RATE ERA IS ENDING

Lock-In, Then *Meter*

The build versus buy decision for AI inference after the subscription era, and the realistic point at which self hosting actually pays.

Cody Champion, PhD
AI Decision Science · Claude Certified Architect

A WHITE PAPER
JUNE 2026
EVIDENCE-BACKED

BOTTOM LINE UP FRONT

The shift from flat subscriptions to token billing is spreading across the AI tooling market, and it is gathering speed. GitHub Copilot moved 4.7 million paid subscribers onto a meter on 1 June 2026, after Cursor and Windsurf had already gone the same way. Wanting out of that is reasonable, but the first lever most teams reach for, owning GPUs, is rarely the right one. What protects you is **switchability**: building on portable interfaces against a market of interchangeable open weight providers, so that no single vendor can pin you in place and then raise the price. On cost alone, **managed open model inference wins in most situations**, because a serious provider keeps its hardware far busier than you ever will. And if you do decide to own infrastructure outright, the economics only work past roughly **one hundred billion tokens a month with a dedicated platform team**, since the people, not the silicon, are the thing you are really buying. Buy hardware when sovereignty or scale calls for it, never as a way to dodge the meter.

01 The pattern is no longer speculative

For about three years, the flat monthly AI subscription looked permanent, in the way you stop noticing the price of something you use every day. It was always a subsidy dressed up as a product, and the disguise is coming off in public. The order of events is easy to follow. Cursor switched from request based to credit based billing in June 2025 and took a wave of backlash for it. Windsurf reworked its pricing twice over the following year. Then came the move that more or less settled the argument.

On 1 June 2026, GitHub Copilot retired its flat rate plans and Premium requests, and put every paid tier onto token metered AI Credits. Completions stayed free. Everything agentic, the chat and the multi step work, now draws down credits at model API rates. The numbers developers traded in the days afterward were rough but alarming. Heavy users projected increases of ten to fifty times the old cost. One person reported a twenty nine dollar plan turning into something closer to seven hundred and fifty, and another watched fifty dollars become three thousand.

None of this was kept quiet. GitHub's own product leadership pointed to weekly compute costs that had nearly doubled since January 2026, and admitted that a few heavy requests could already cost more than a whole monthly plan. Greed is the wrong word for it. What is actually happening is a market settling onto its real unit economics, and every enterprise AI product still living on a subsidy will go through the same adjustment eventually.

02 Why this is structural, not a betrayal

Flat pricing made sense when the product was autocomplete, where generating one more suggestion cost almost nothing. Agentic tools broke that arrangement. A single instruction can now load a large context, fire off a dozen tool calls, and run several rounds of reasoning, which means the cost tracks usage almost one for one. Anything that is priced like compute eventually gets billed like compute. The only open questions are when the subsidy runs out, and whether you see the change coming or get blindsided by it.

The lock in works the same way whether anyone planned it or not. You sign customers up on a flat rate, give them time to wire their workflows and habits and quiet little dependencies into the tool, and then, once leaving has become genuinely painful, you switch on the meter. Whether that sequence was a deliberate strategy or just gravity makes no difference to the person paying the bill. By the time the price moves, they are already in too deep to do much about it.

> *A meter only becomes dangerous once you have lost the ability to walk away from it.*

03 Reframe the risk you are actually carrying

Most of the anxiety here bundles two different problems together: being charged per token, and being unable to leave. Per token pricing on its own is fine, and for plenty of workloads it works out cheaper than a flat plan ever did. What hurts is the pairing of a meter with a workload you cannot move. A vendor only has a hold on you for as long as you are stuck with them.

So the engineering question worth asking has more to do with exit speed than with self reliance. How quickly, and how cheaply, could you hand this workload to someone else? A workload you can repoint to a different provider in an afternoon is one no vendor can squeeze, and at that point the meter stops being a threat at all. Running your own hardware is one route to that position. It also happens to be the most expensive route, and close to the last one you should reach for.

04 The four layer inference stack

Self hosting gets talked about as a single yes or no choice, when it is really a spectrum. The four layers below run from least control to most. Every step up the ladder hands you more independence, and asks for more operational work in return.

LAYER	WHAT IT IS	CONTROL	OPS BURDEN	LOCK-IN RISK
1. Frontier API `Claude, GPT, Gemini`	Best in class quality, called over the wire	Low	None	**High** if you depend on one model's behavior
2. Managed open API `DeepInfra, Together, Fireworks, Groq, OpenRouter`	Open weight models served by many vendors behind a common API	Low	None	**Low**, same weights, many hosts
3. Rented GPU `CoreWeave, Lambda, RunPod`	You operate the serving stack on rented silicon	High	Heavy	Low external, high internal complexity
4. Owned hardware `On-prem or colo`	You buy and run the metal	Total	Heaviest	None external, full capital and staffing

Layer two is the one most teams skip past, and it deserves far more attention than it gets. Managed open model inference hands you almost all of the lock in protection that self hosting promises, with none of the operational weight. The same DeepSeek, Llama, or Qwen weights sit behind half a dozen interchangeable providers, all of them speaking the same OpenAI compatible API. That crowd of ready substitutes is exactly what takes a vendor's pricing power away.

05 Capability in 2026: the gap has mostly closed

The argument for committing to one frontier vendor used to rest on a clear quality gap. That gap has shrunk fast. Open weight releases now arrive monthly, and for most production work they have caught up. DeepSeek V4 Pro matches the closed frontier on agentic coding and trails Claude Opus by only a handful of points on SWE-Bench Verified, while Kimi K2.6, GLM-5.1, and Qwen3.5 all land in frontier territory on reasoning and code. On the independent leaderboards, open models now hold more of the top spots than the closed ones do.

MODEL	LICENSE	SWE-BENCH VERIFIED	GPQA DIAMOND	NOTES
DeepSeek V4 Pro	`MIT`	`~80%+`	`~85%+`	Ties closed frontier on agentic coding
GLM-5.1	`MIT`	`77.8%`	`~94%*`	Cleanest license, strong all rounder
Qwen3.5 (Reasoning)	`Apache-2.0`	`~strong`	`~89%`	Balanced, permissive, broad sizes
Kimi K2.6	`Open`	`~88.7†`	`high`	Tops a neutral open index
Claude Opus 4.x	`Closed`	`frontier`	`frontier`	Leads head to head coding arenas

Figures are representative and drawn from public benchmark aggregators; standings shift monthly. *Reasoning variant. †Blended coding score, not SWE-Bench. Treat as directional, not as a procurement spec.

Where frontier still earns its premium

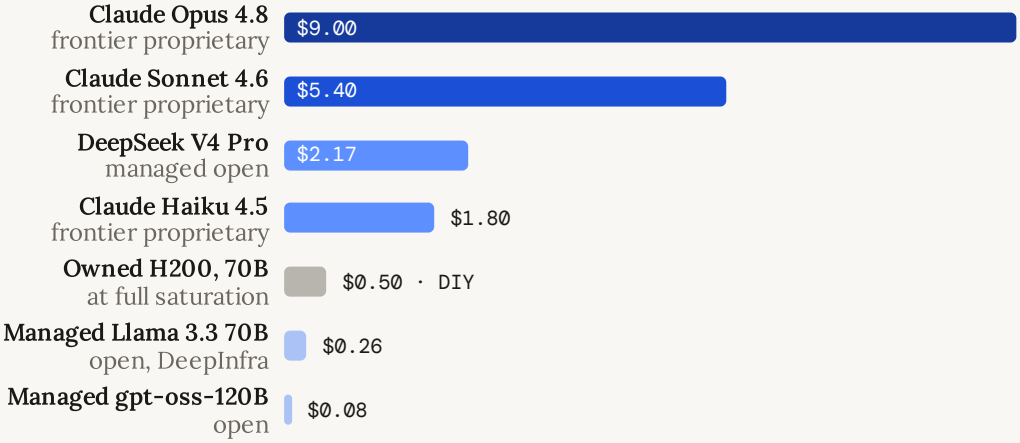
What is left of the gap shows up in the hardest slice of the work, maybe ten to twenty percent of it. Long horizon agents that have to stay coherent across many steps. The most demanding reasoning. The genuinely gnarly code. The sensible architecture routes around the problem instead of picking a single tier for everything: the bulk of the volume goes to cheap open models, and a frontier API stays in reserve for that difficult tail. One decision, and you have lowered your bill and capped your exposure to any single vendor at the same time.

06 The cost reality nobody prices correctly

The usual mistake is to set a GPU's hourly rate next to an API's per token price as though the two numbers were comparable. They are not. The only measure that means anything is cost per million tokens at the utilization you will actually reach, and utilization is where most build it yourself spreadsheets quietly come apart.

A GPU you rent or own costs the same whether it is pinned at full load or sitting idle overnight. A single reserved H200 runs about two thousand dollars a month and will comfortably serve a 70B class model, but once you account for the load you can realistically keep on it, the effective price settles somewhere near forty to fifty eight cents per million tokens. Managed providers serve that same class of model for twenty three to forty cents. They get there by batching across many tenants at once, a level of utilization a single team almost never matches on its own. So for the ordinary open models that make up most of your traffic, the provider usually beats your own rack on price.

REPRESENTATIVE BLENDED COST PER MILLION TOKENS · 4:1 INPUT:OUTPUT MIX · USD



SOURCE: PUBLIC PROVIDER PRICE LISTS, JUNE 2026. OWNED FIGURE ASSUMES NEAR-SATURATION AND EXCLUDES THE PLATFORM TEAM.

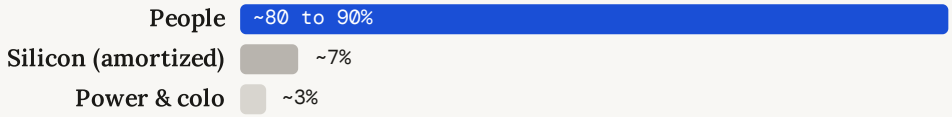
07 The break-even, with people in the model

Most build versus buy analyses leave out the largest number on the page, which is the team. Standing up open models in production is not something you deploy once and walk away from. It is a platform you have to keep running: serving infrastructure like vLLM or SGLang, autoscaling, quantization, an evaluation harness, regular model

refreshes, observability, and someone on call when it breaks at two in the morning. In practice that is two to four engineers. In a market like Dublin or the US, their fully loaded salaries make the cost of the GPUs look like a rounding error.

OWNING TWO H200S, MONTHLY ALL-IN	COST / MONTH
Capital, ~$80k to $100k amortized over 3 years	$2,200 to $2,800
Power and colo, Western European rates	$700 to $1,200
Platform staff, 2 FTE fully loaded	**$25,000 to $40,000**
Total	**$30,000 to $44,000**

WHAT DOMINATES THE BUILD COST OF OWNED INFERENCE



TWO-H200 OWNED SCENARIO, FULLY LOADED. PEOPLE COST DOMINATES AT EVERY REALISTIC VOLUME.

At a managed blended rate near thirty cents per million tokens, that thirty four thousand dollars a month would buy you something like one hundred and ten billion tokens from a provider. So the break even on cost alone sits up there, and even reaching it assumes you keep the hardware saturated the whole time. Fall short of that, and you are paying a full platform team to lose money against an API. Which is really the lesson worth carrying out of this section. Owning hardware is a decision about sovereignty or sheer scale, and only very rarely a decision about saving money.

THE THREE THRESHOLDS

1. **Frontier API to tiered routing on managed open models.** Stand up a router, send the bulk of volume to open weights, keep a frontier model for the hard tail. Near zero regret, and the real hedge against the meter.
   TRIGGER: STEADY PRODUCTION VOLUME + TASKS AN OPEN MODEL HANDLES WELL

2. **Managed open API to rented dedicated GPU.** Move when a constraint binds, or when sustained utilization would keep a dedicated GPU busy past roughly sixty to seventy percent. Below that, managed wins on cost.
   TRIGGER: DATA RESIDENCY, LATENCY OR THROUGHPUT SLAS, PROPRIETARY FINE-TUNES, HIGH STEADY LOAD

3. **Rented to owned hardware and a platform team.** Justified by near 24/7 utilization sustained for 18 months or more, a regulatory mandate for on-prem, and the ability to staff the team. For nearly everyone else, this is a sovereignty decision, not a budget one.
   TRIGGER: ~100B+ TOKENS / MONTH SUSTAINED, OR A HARD COMPLIANCE MANDATE

08 Decision framework

IF YOUR PRIMARY CONSTRAINT IS…	DEFAULT TO
Unpredictable volume, broad task mix	**Tiered routing:** managed frontier + managed open behind one gateway
Cost at scale on commodity tasks	**Managed open**, multi provider, switchable on price
The hard tail: agents, frontier reasoning	**Keep a frontier API** for that slice only
Data residency, air gap, regulated data	**Rented dedicated** or **owned**
A proprietary fine-tune as a moat	**Rented dedicated**, scale to owned if volume justifies
Strategic independence and insurance	**Portable tiered stack now**, owned only past the scale break-even

09 The hedge that actually works

Substitutability is something you engineer into the system from the beginning. No procurement clause hands it to you after the fact. The checklist that gets you there is short, and it belongs in the architecture rather than bolted on once everything is already wired to one vendor.

- Build against an **OpenAI compatible interface** so providers and models are swappable without rewrites.
- Put a **gateway in front** (LiteLLM, OpenRouter, or your own) to route, fail over, and price shop across providers.
- Keep prompts and tool definitions **model agnostic**, so you are not leaning on one model's quirks.
- Maintain an **evaluation harness** so a candidate replacement can be qualified in hours rather than weeks.
- Pin a frontier model for the hard tail, but keep it **behind the same interface** as everything else.
- Track **cost per token by workflow**, so you see the meter coming before it surprises you.

Get this right and you end up in a comfortable spot. The per token bill is still there, and that part was always fair enough. What changes is that you can leave the moment the numbers stop making sense, and being able to leave is worth a great deal more than any single line on the invoice.

10 Bottom line

The trouble was never that AI started costing money by the token. It is what a meter can do to you once you have lost the ability to walk away, and that is a problem you can design out long before it bites. Build for substitutability and the whole lock in then meter routine simply stops working on you. There are real reasons to run your own infrastructure. Sovereignty, hard latency guarantees, a model you have fine tuned into a genuine advantage. When one of those applies, do it. Owning the metal outright is narrower still, worth the trouble only when the scale is enormous and the team already exists. For most of what most companies actually run, a portable stack that spans a few providers and tiers will cost less, move faster when you need it to, and leave nobody in a position to reprice you after you have committed. That last property is the one to keep your eye on.

◆

METHODOLOGY & SOURCES. Pricing and benchmark figures are drawn from public provider price lists and independent aggregators as of June 2026, including Anthropic, GitHub, DeepInfra, Together AI, Fireworks, CoreWeave, Lambda, Spheron, and benchmark trackers (Artificial Analysis, Vellum, LLM-Stats, BenchLM). Frontier API rates: Claude Opus 4.8 at $5 / $25, Sonnet 4.6 at $3 / $15, Haiku 4.5 at $1 / $5 per million input / output tokens. All figures are representative estimates for decision framing. Per token economics, model standings, and GPU rates shift monthly. Validate against your own workload and a current price check before committing capital or headcount. This document is analysis, not financial or procurement advice.