

THE AI NOBODY IS SELLING YOU

THERE IS MORE THAN ONE WAY TO SAY AI

EXECUTIVE SUMMARY

Every enterprise running AI today receives a monthly bill that is hard to predict, and, in a growing number of cases, hard to justify. This is true for AI billed by consumption, and it is true in a different way for AI billed by seat, the flat, per user licenses many organizations already carry alongside their consumption based tools. Neither problem is a temporary rough edge. Both are direct consequences of how today's dominant AI commercial models are built.

A structural alternative exists. Instead of renting access to a model, an organization can own the AI platform itself, model included, on infrastructure it controls. This paper works through that alternative for an architect audience, in architectural terms: what problem it actually solves, what it requires to run, where its real limits are, and how it fits alongside the closed AI an organization is likely already using.

Cost is where this conversation almost always starts, so this paper starts there too. But it does not stop there, because in practice it never does.

THE PLAYING FIELD

Something has shifted in enterprise AI conversations over the past year. A question that used to come up as a footnote, whether an organization should consider running its own AI model instead of renting one, has become one of the most frequent questions architects and CIOs are asking.

It rarely starts as a question of principle. It starts as a practical one, and almost always from the same place: the monthly AI bill. That bill takes two different shapes depending on how the AI is licensed, and both shapes cause real problems, for different reasons. This paper sets out both problems in detail, then works through the alternative and everything that comes with it.

Once that first question is on the table, cost stops being the only thing architects want to talk about. The follow up questions come quickly. Where does the data actually go? What does European regulation expect from us? Are we making ourselves dependent on infrastructure we do not control, again? Could this be used to build something specific for one part of the business without redoing the integration work later? Is there something useful to do with hardware that already sits idle across the organization? Each of these gets its own section below?

TWO KINDS OF AI, AND WHY THE DIFFERENCE MATTERS

Most conversations about AI cost skip a basic fact: there is more than one way to run AI, and the different ways have fundamentally different cost behavior.

- **Closed AI.** This is what most people mean when they say AI today: tools such as Claude, ChatGPT, or Gemini, whether used directly or embedded inside another product. The organization does not own the model. It rents access to it from the company that built it, and either pays per unit of use or pays a fixed fee per user seat. The provider controls the pricing, the availability, and the schedule on which the model changes or is retired.

- **Open weight AI models.** These are AI models whose trained parameters can be downloaded and run on infrastructure the organization itself controls, on premises or in a cloud account it manages. A number of these models, from labs many architects have not heard of alongside well known names, now perform close to the same level as leading closed models for a large share of everyday tasks, coding included. There is no per use fee to the model's creator. The organization pays for the computing capacity to run the model itself, a cost that can be fixed and contracted in advance.

The rest of this paper is about what changes, in both directions, once an organization treats the second option as a real architectural choice rather than a research curiosity.

WHY THE CONSUMPTION BASED INVOICE IS UNPREDICTABLE, BY DESIGN

Closed AI providers that bill by consumption charge for tiny units of usage, commonly called tokens, that add up every time the AI is used. There is no equivalent of a fixed software license here, and no cap built into the architecture by default. A single workload, a misconfigured agent loop, a new project, or simply more people using the tool, can move the invoice by a large amount in a single month, without warning.

This is not an oversight the providers are working to fix. It is the mechanism by which their own, extremely capital intensive business survives. Training frontier models costs enormous sums. Consumption based pricing recovers that cost at scale, in proportion to the value delivered. A frontier AI provider offering a genuinely fixed price, unlimited usage enterprise contract would be working against its own financial model, and it is not a mainstream offering for that reason, regardless of how large the customer is.

Enterprises already know how to evaluate this kind of trade off, because they have made it before with other technology decisions. Faced with a fixed price license that costs more on average against a consumption based alternative that costs less on average but swings unpredictably, many organizations choose the fixed price anyway. Predictability has its own value, and finance teams are often willing to pay a premium for it.

THE OTHER HALF OF THE PROBLEM: SEAT BASED COST AND LOW USE

Consumption based billing is only one half of the AI cost picture. A large part of enterprise AI spend today does not come from token metering at all. It comes from seat based licenses, a fixed monthly fee per named user, of the kind sold alongside office and productivity software, including entry level AI assistant tiers such as a basic Copilot style license.

Seat based pricing does solve the unpredictability problem described above. The invoice is fixed and known in advance. But it introduces a different problem, and it is one we see consistently in our own conversations with customers and in usage data some customers have shared with us directly. In an organization of roughly a thousand people, actual regular use of a seat based AI assistant is often concentrated in a small part of that population, and a meaningful share of that

use turns out to be light, occasional, or the kind of task that does not clearly justify the cost of the license across everyone it was purchased for.

This is a return on license problem rather than a return on consumption problem, but it produces the same underlying frustration at board level: a monthly AI cost that is hard to defend against the value it visibly produces. A fixed invoice that nobody can point to concrete value for is not obviously better than a variable invoice that at least scales with actual use. Any serious cost conversation about AI has to hold both pictures at once, the unpredictable consumption bill and the underused seat bill, because most enterprises are already carrying some mix of both.

JUST TURN IT OFF IS NOT A STRATEGY

When AI spend escalates without warning, the most common first response is blunt: cap it, and when the cap is hit, shut it down. That reflex is understandable, and in the short term it is sometimes the only lever available. It is not, however, a workable long term operating model. It converts AI from a dependable capability into an intermittent one that disappears on an unpredictable date each month, forcing teams to plan around an outage that has nothing to do with their own performance.

Even this blunt instrument is harder to implement cleanly than it looks. Genuine, provider level hard spend limits are a fairly recent addition to the market, and even where they exist, requests already in progress are typically allowed to complete past the configured limit. A reliable stop requires an extra engineering layer, sitting in front of the provider, that blocks a request before it is ever sent. That is a solvable engineering problem, but even a perfectly built off switch still leaves the organization with an AI capability it can only run intermittently. The more useful question is whether that trade off is necessary at all.

IT IS NOT ONLY ABOUT COST

Once cost is on the table, in either of the two forms described above, several other questions tend to surface quickly in our conversations with architects. The five below are the ones that come up most often, roughly in the order they tend to appear.

DATA SOVEREIGNTY: KEEPING DATA INSIDE THE PERIMETER

When an application calls a closed model's API, the prompt and any retrieved context travel to infrastructure the provider operates, wherever that happens to be. For a large share of everyday tasks this is a manageable, contractually governed risk. For some categories of data it is a harder question regardless of the contract in place, simply because that data should not leave a defined perimeter at all.

Self hosting an open weight model on infrastructure the organization controls removes that question rather than mitigating it, because the data never has to leave the perimeter in the first place. This matters most for the categories architects already treat carefully: legal case files, HR records, health data, and anything covered by sector specific confidentiality rules. As one concrete example, the European Health Data Space regulation includes a requirement, under its Article 50, that secondary use of health data only happen inside a secure processing

environment, which makes the physical location and control of compute a compliance question rather than a preference for that sector.

THE REGULATORY TAILWIND

European regulation is not, by itself, forcing organizations toward open weight or self hosted AI. But several pieces of it are making the location and control of AI infrastructure a live procurement question rather than a background concern.

- **The EU AI Act.** Obligations for general purpose AI providers, including training data summaries and a published copyright policy, have applied since August 2025. Its enforcement powers and Article 50 transparency obligations became applicable on August 2, 2026, while a separate Digital Omnibus deferred the high risk system obligations to December 2027. These timelines move, so treat any specific date as a point in time rather than a permanent fact.
- **NIS2.** This directive requires essential and important entities to implement supply chain security measures and report significant incidents. Agentic AI platforms that automate critical business processes may fall within its scope as ICT service providers, depending on sector and on how the directive has been transposed into national law.

The honest reading is that none of these obligations require EU hosted or self hosted infrastructure on their own. What they do require is documented audit trails, human oversight, and demonstrable control over how a system reached a decision, and those are simply easier to build and defend when the infrastructure sits inside the organization's own perimeter. Owning the platform does not exempt an organization from any of these obligations. It puts the organization in a stronger position to demonstrate that it is meeting them.

VENDOR INDEPENDENCE AND CONTINUITY

Relying on a single AI vendor for a critical workflow carries the same kind of risk architects already manage for any other single point of failure: pricing changes, feature or model deprecation on the vendor's own schedule, and, in rare but real cases, a sudden change in availability for reasons entirely outside the customer's control.

A concrete, publicly disclosed example: in mid 2026, United States export control action led one major AI provider to temporarily suspend access to two of its models, with access restored several weeks later once the relevant controls were lifted. This is mentioned not to single out one provider, but because it is a real, documented instance of the general architectural risk this section is about. Access to a closed model can change for reasons that have nothing to do with the customer's contract or behavior.

An open weight model, once downloaded, belongs to the organization indefinitely. It does not disappear if a vendor's pricing, roadmap, or regulatory status changes. This argues for a deliberate multi model architecture, not for abandoning closed models. The sound engineering answer is the same one architects already apply to cloud providers and critical suppliers: build the integration layer so that no single vendor decision can halt every workflow at once, and test that fallback before it is needed.

DIVISION SPECIFIC ASSISTANTS: SAME INTEGRATION EFFORT, MORE CHOICE

A pattern we see often: one part of the business, legal is a typical example, wants its own assistant, linked to its own archive and its own way of working, rather than a generic company wide tool. Building that assistant requires a defined set of engineering work regardless of which model ends up behind it: retrieval against the right archive, access control, an integration layer to the systems that archive lives in, and a usable interface.

That is the useful insight for architecture planning. Once that integration layer is built well, in a model agnostic way, swapping in an open weight model instead of a closed one, or running both side by side for different tasks, becomes a relatively small decision made near the end of the project rather than a decision that has to be made upfront. The Model Context Protocol, the open standard for connecting AI systems to external tools, repositories, and data, was introduced by Anthropic but has since been adopted across the industry, including by competing model providers. It is explicitly designed to be model agnostic, which is what makes this late binding approach practical: a self hosted open weight model can connect to the same internal systems a closed model would use, through the same integration layer.

THE EDGE PATTERN: IDLE COMPUTE AS UNDERUSED CAPACITY

Most office laptops and desktops sit idle for a large part of the working day. Modern business hardware, including the NPUs now built into many recent laptops, is capable of running small to mid sized open weight models entirely on the device itself, with no round trip to a shared server or a cloud API.

We are currently running a proof of concept built around exactly this pattern: a lightweight, copilot style assistant installed as an app on each PC, using the machine's own compute rather than a shared server or a rented API. We are evaluating it with several customers at the moment, and the early results are encouraging specifically for lightweight, well bounded, day to day tasks. This is not a substitute for a properly resourced platform when the workload calls for a larger model or complex multi step orchestration, and hardware fleets are uneven across any real organization, so this works best as one deliberate tier in a wider architecture rather than a default for everything.

Because the device is already switched on and mostly idle for other reasons, this approach uses existing resources more efficiently rather than requiring new shared infrastructure to be provisioned. Some people will see a sustainability angle in that too. We think the efficiency argument, using what you already have, stands on its own without needing to lean on that framing, so we will leave it there.

OWN THE PLATFORM, NOT JUST THE MODEL

Everything in the previous section depends on the same underlying architectural move, so it is worth setting out in full. Instead of paying a closed provider per token, or per seat, the organization deploys an open weight model on infrastructure it controls: reserved cloud GPU capacity under a multi year commitment, dedicated on premises hardware, or, for the lightweight cases described above, existing endpoint hardware that is already part of the estate. The monthly cost becomes fixed and contracted, much like a traditional software license,

because the organization is now paying for capacity, not consumption, and not for a seat whether it is used or not.

This is also where the two cost problems from earlier in this paper meet. A shared, owned platform can absorb the kind of light, occasional use that does not justify a per seat license across a thousand people, because the marginal cost of one more internal user on shared infrastructure is small, while it also removes the unpredictable per token invoice for the heavier, high volume workloads. Owning the platform does not just fix one of the two problems. It changes the shape of both at once.

This is a bigger commitment than selecting a pricing tier, and architects evaluating it should go in with a clear picture of what it actually takes to run. Five components make up the platform.

- **Compute layer.** Reserved GPU capacity under a cloud commitment, dedicated on premises hardware, or existing endpoint hardware for the lightweight, on device cases.
- **Model layer.** A deliberately chosen, version pinned open weight model, left stable for a defined period rather than updated automatically. The choice of model, and the decision to freeze it, are both architecture decisions, not defaults.
- **Serving layer.** An inference engine exposing a standard API to existing developer tooling. Typical choices include vLLM or SGLang for server side deployment, and ONNX Runtime or a vendor specific NPU runtime for on device deployment.
- **Integration layer.** Model Context Protocol connections to internal and external systems, built to be model agnostic from the start, as described earlier in this paper.
- **Governance layer.** Access control, logging, and the compliance documentation described earlier, needed regardless of which model sits underneath.

Running this well is a genuine platform engineering capability, not a one time installation project. Neither a closed model nor an open one fixes a weak technical foundation. Both expose it, and a self hosted model is, if anything, less forgiving of a weak foundation than a closed API, because it adds a new operational responsibility on top of infrastructure that now has to carry it.

This is not a small undertaking, and it deserves more depth than a single section of this paper can give it. 45 Degrees has its own platform engineering framework, built specifically to structure this kind of decision, and we recently extended it with a dedicated AI plane.¹ Explaining that framework in full is a separate whitepaper in its own right, so we will not attempt it here.

THE HONEST CAPABILITY PICTURE

The obvious question for anyone evaluating this is whether an open weight model is actually good enough to replace a closed model for real work. As of today, for coding and general enterprise workloads specifically, the practical gap between well chosen open weight models and closed frontier models has narrowed substantially over the past two years. What remains is concentrated in complex, multi step agentic reasoning: choosing the right tool, chaining several calls together, and recovering cleanly from a failed step, rather than in raw coding ability or general knowledge.

¹ We are happy to discuss this framework in more detail. Contact us at info@45degrees.be.

A question we hear, close to word for word, in almost every one of these conversations is this: if a model gets me eighty percent of what a closed model gives me, that is probably good enough for most of my workload. The honest answer is that this understates the current picture. On several well known coding and reasoning benchmarks, today's strongest open weight models are not trailing far behind closed frontier models. They are close behind, sometimes level, and on specific benchmarks ahead, depending on the exact pair of models compared and the exact task being measured. So the realistic answer to the eighty percent question, for a meaningful part of most enterprise workloads, is usually that it does noticeably better than eighty percent.

That said, a benchmark score is not your workload. The only reliable way to know what a specific model will do for a specific task in your environment is to test it against that task, with your own data and your own definition of good enough. Public benchmark numbers, including any figures used in this paper, change from month to month and vary between sources. Treat them as a reason to shortlist a model for evaluation, never as a reason to skip the evaluation.

RISKS AND CONSTRAINTS WORTH DESIGNING AROUND

None of the above should be read as a case for open weight AI without qualification. A few risks are worth naming plainly.

- **License terms vary by model and by version.** Some of the strongest open weight coding models today originate outside Europe, and license terms range from fully permissive to more restrictive terms with usage conditions attached. Check the license of the exact model and version being deployed rather than assuming an entire family behaves the same way.
- **Patch cadence.** The serving stack, drivers, inference engine, and operating system, needs its own security patch cycle, generally faster than the model's own release cycle. This is ongoing operational work, not a one time setup task.
- **Compliance responsibility does not move.** EU AI Act obligations and other sector specific rules apply to the deploying organization regardless of whether the model is open or closed, or where it runs.
- **Fleet unevenness.** Where part of the architecture depends on end user hardware, device age and specification will vary across the organization, and the design needs a sensible fallback for hardware that cannot carry the workload.

THE REALISTIC ARCHITECTURE: HYBRID, NOT ALL OR NOTHING

Bringing all of the above together, the most defensible architecture is a deliberate split, not a wholesale replacement of closed AI. The bulk of well understood, high volume, routine work runs on the owned platform, potentially split further between shared infrastructure and on device compute depending on the task. The smaller, genuinely complex share of the workload, the multi step orchestration tasks described above, continues to run against a closed provider under a small, deliberately capped budget. The result is a predictable base cost with a small, contractually bounded variable exposure, built on an integration layer that never assumed a single vendor or a single model in the first place. That last point is the one worth carrying into any architecture review. The goal is not to pick a winner between open and closed. It is to build

an architecture where that choice can be made, and remade, per workload, without redoing the foundation each time.

WHERE TO START

- Classify current and planned AI workloads by volume and complexity, and separately, note which ones sit behind a seat based license and which sit behind consumption based pricing.
- Pick one well bounded, high volume workload as a pilot for the owned platform, rather than attempting a broad migration in one step.
- Verify the license and provenance of any candidate open weight model for the specific version being deployed, before it enters a business case.
- Build the integration and governance layers to be model agnostic from day one, using an open standard such as MCP, so the model choice stays a late, reversible decision.
- Treat on device and edge deployment as a distinct, deliberate tier for lightweight tasks, not as a default answer for every workload.
- Measure cost and quality against your own representative tasks, not against a published benchmark score alone.

CONCLUSION

The AI cost conversation that most organizations are having today is really two conversations wearing one invoice. One is about consumption that nobody can forecast. The other is about seats that nobody is quite using. Both point toward the same structural answer: an AI platform the organization owns, sized to its own workloads, sitting alongside the closed AI it will likely keep using for the harder, less common tasks. That is a bigger architectural commitment than choosing a pricing tier, and it comes with real operational responsibility that does not disappear just because the model is open. But for organizations running meaningful, sustained AI workloads, it is a real, technically credible option, worth evaluating on its own architectural merits.

None of this should be read as easy. Owning an AI platform is technically possible today in a way it was not two years ago, but possible is not the same as simple. It only works on top of a foundation that is already solid: consistent engineering practices, automated and governed pipelines, clear security, and reliable monitoring. Without that foundation, an owned platform will surface the same problems a rented one hides for a while longer, only sooner and more visibly. This is exactly the kind of undertaking where getting the sequence right matters more than getting the model choice right, and it is also exactly where organizations most often underestimate the work involved.

This is where 45 Degrees can help. We do not start this conversation with a model. We start it with the platform engineering foundation underneath it, the same foundation any serious AI strategy depends on, open or closed. Our platform engineering framework was built for exactly this kind of decision, and we recently extended it with a dedicated AI plane, precisely to support organizations making this move. If your foundation is not there yet, that is the right place to start the conversation, not the model. If it already is, we are glad to help you take the next step. Reach us at info@45degrees.be.