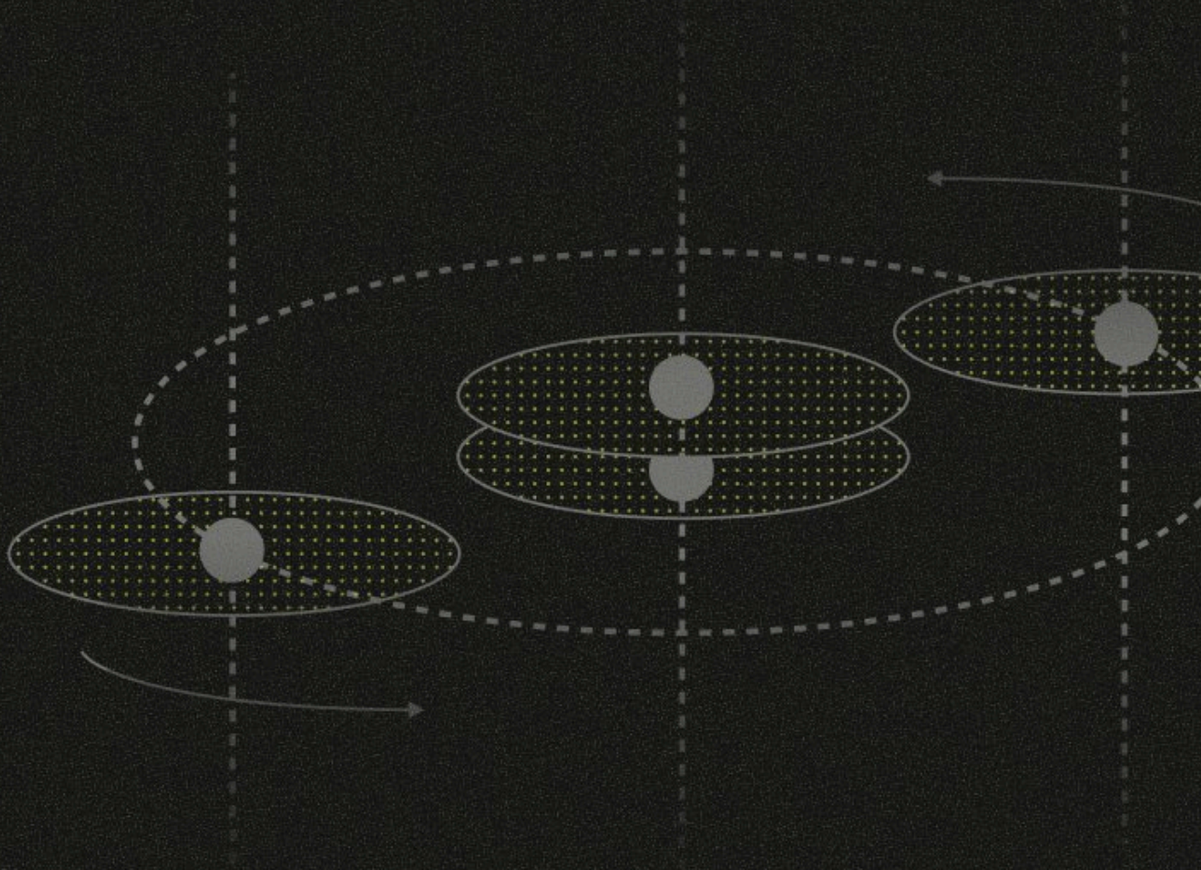
Build the Logic
Buy the Platform

SaaS is unbundling. What AI changes for enterprise software, what it doesn't, and what owning the rest really costs.



Kognitos

The short version

Every SaaS contract you ever signed bought two things in one box. The first was a platform: the hard, unglamorous engineering of running software reliably, securely, and at scale. The second was business logic: the vendor's best guess at how your process should work, exposed through settings, menus, and picklists.

The platform half was excellent. The logic half was never finished, and it couldn't be, because one product had to serve thousands of companies that each work differently. That is why the largest SaaS vendors grew developer ecosystems around customization: customers kept building the logic the product left out.

AI breaks the bundle. Coding agents make business logic fast and cheap to build, and the builders inside companies are right to want to own it. Logic is where a company's differentiation lives.

But AI does not make the platform cheap to own. It made writing code cheaper. It did not make running that code with SaaS-grade guarantees any cheaper: the on-call rotations, the security reviews, the audits, the upgrades, the scaling.

Build the logic. Buy the platform. And buy a platform that leaves the logic entirely to you.

At a glance

Key takeaways

01

SaaS was a bundle. A platform you couldn't build, wrapped around logic you couldn't control.

02

AI unbundles it. Business logic is now cheap to build, and it should be owned by the business.

03

The platform is still expensive. AI made code cheap. It didn't make owning a platform cheap.

04

Don't re-bundle. Point solutions and AI-flavored SaaS put your logic back in someone else's menus.

05

Buy the guarantees, own the logic. The right platform specifies everything except the part that's yours.

What SaaS really sold

Section 01 of 09

A platform you couldn't build, wrapped around logic you couldn't control

When a company bought SaaS, the most valuable thing it bought was everything it didn't have to hire for. Nobody had to staff cloud infrastructure, Kubernetes, deployment pipelines, scaling and autoscaling, reliability engineering, security, governance, performance tuning, audit logging, metrics, monitoring, or SLA reporting. The vendor did all of it, for every customer at once.

That was a genuinely good deal, and the SaaS industry, and the cloud providers underneath it, delivered on it.

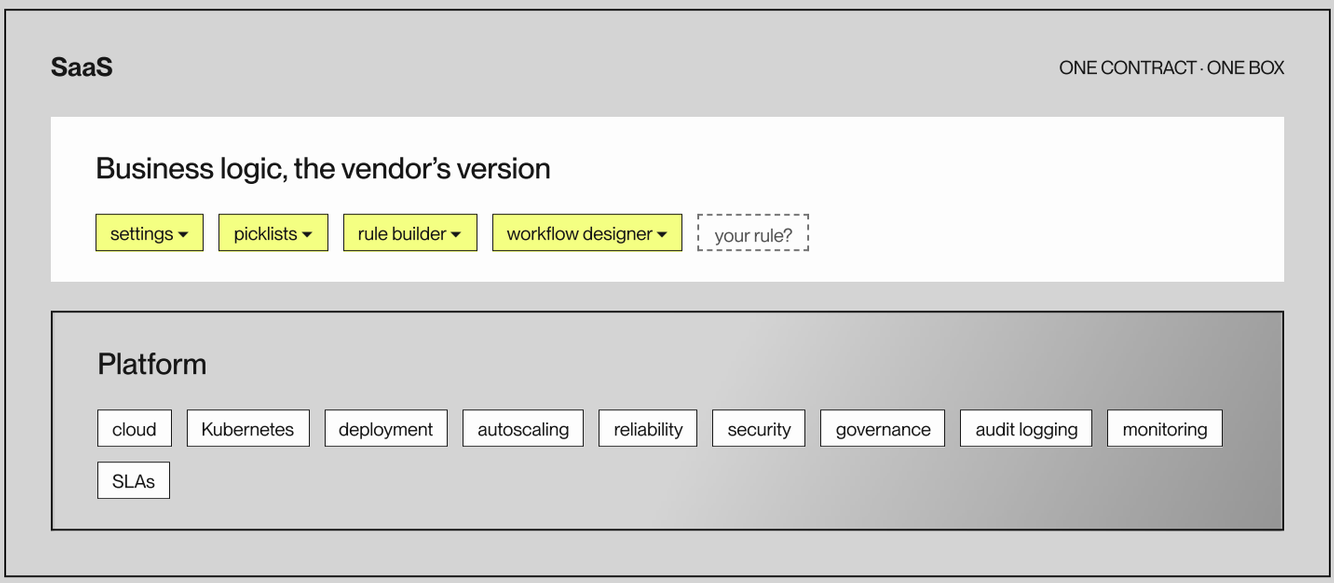


Figure 1. What a SaaS contract bought: an excellent platform, wrapped around business logic you could only configure.

The business logic was the other half of the box, and here the deal was weaker. A vendor had to express how thousands of different companies work through one product, so it offered configuration: fields, picklists, rule builders, workflow designers. Whatever the menus couldn't express was left to the customer.

Look at what grew up around the largest SaaS platforms: administrators, consultants, and developers whose job is to refine, configure, and build on top of the product. Salesforce's developer community is one of the most impressive in enterprise software, and it exists because the logic layer was incomplete. That isn't a criticism. It was the unavoidable cost of one product serving everyone.

SaaS was never just software you rented. It was a platform you didn't have to build, wrapped around logic you didn't get to control.

What AI changes

Section 02 of 09

AI made business logic cheap, and it should be yours

AI changes the economics of the logic layer. A builder inside a business can now describe what a process should do and have working logic in hours instead of a quarter. Coding agents turn intent into code, and the cost is counted in tokens rather than headcount.

So builders are asking a reasonable question: why accept a vendor's approximation of our process, configured through menus, when we can build exactly the logic we need? They are right to ask it. Business logic is where a company's differentiation lives. It encodes how you price, approve, reconcile, escalate, and decide. It should belong to you, be readable by you, and change when you decide it should.

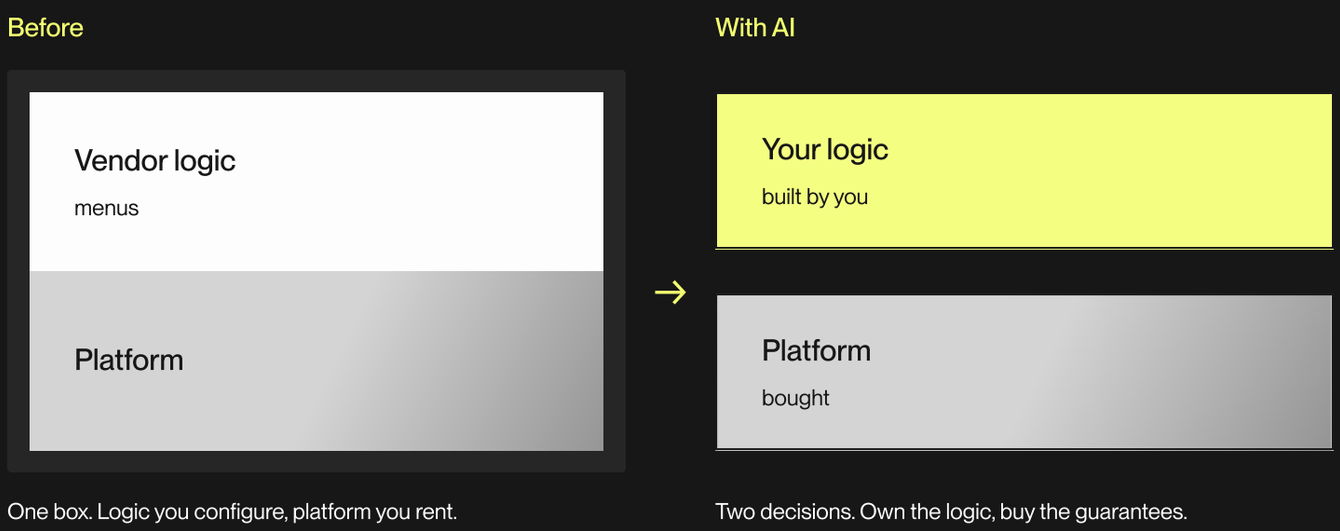


Figure 2. AI unbundles SaaS. The logic moves to the people who know the business; the platform stays something you buy.

This is what people mean when they say SaaS is dying. Software spending isn't falling, and products aren't disappearing. What is dying is the bundle: the assumption that to get a reliable platform you must also accept someone else's logic.

Don't re-bundle

The new wave of point solutions and AI-flavored SaaS bundles the same way. It encodes your logic in its own model and its own menus, now with a chat window on top. Trading one vendor's configuration screens for another's isn't owning your logic.

What AI doesn’t change

Section 03 of 09

AI made code cheap. It didn‘t make owning a platform cheap.

Here is the part the build-everything argument leaves out. The logic is only the visible part of a production system. Underneath it sits everything SaaS used to carry for you, and none of it went away because the code got cheaper to write.

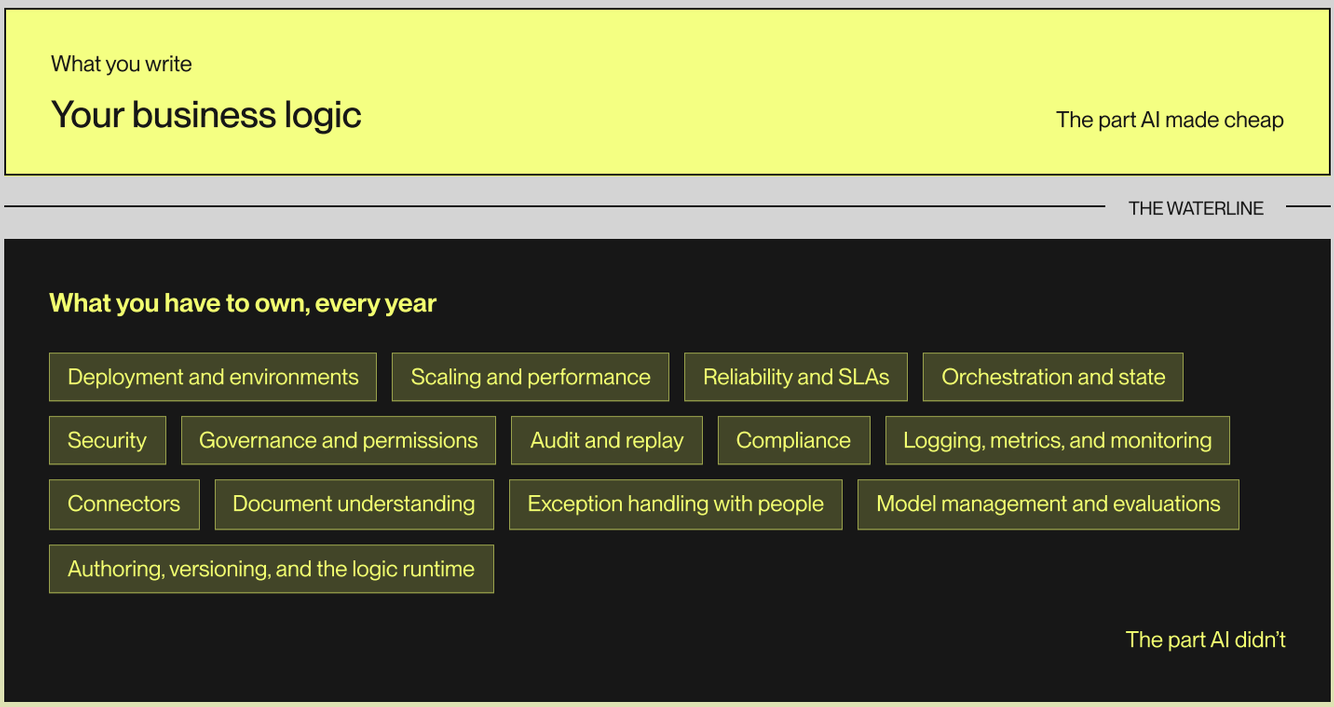


Figure 3. The logic is the visible part of a production system. Everything below the line is what SaaS used to carry for you.

Guarantee	What owning it means, year after year
Deployment and environments	Upgrades, cluster maintenance, release engineering
Scaling and performance	Capacity planning, performance regressions
Reliability and SLAs	24/7 on-call, incident reviews, SLA reporting
Orchestration and state	Stuck-run triage, migrations of in-flight state
Security	Patching, penetration tests, vulnerability response
Governance and permissions	Access reviews, policy changes
Audit and replay	Evidence requests, retention, auditor walkthroughs
Compliance	Annual audits, control testing, remediation
Logging, metrics, and monitoring	Alert tuning, observability costs
Connectors	API changes on every connected system
Document understanding	Model updates, new document types
Exception handling with people	Workbench operations, reviewer tooling
Model management and evaluations	Model upgrades, evaluation suites, inference cost control
Authoring, versioning, and the logic runtime	Language and runtime evolution

Coding agents can write much of this code. What they don't do is carry the pager at 2 a.m., sit through the SOC 2 audit, patch the next critical vulnerability, keep dozens of connectors working as dozens of vendors change their APIs, or explain to an auditor why a decision was made eighteen months ago. Those are the costs that compound, and they are people costs, not token costs.

The first version ships fast. That is the seduction. By the second year, the team that built a few workflows is running a platform: on-call rotations, upgrade backlogs, security questionnaires. The company has quietly become a SaaS vendor with one customer.

You can generate a platform's code in weeks.
You can't generate the years of operating it.

What it really costs

Section 04 of 09

The cost is a team, not a build

It is tempting to price a platform by what it takes to build. In the AI era that is the smaller number, and it keeps getting smaller. The number that matters is the team it takes to keep every guarantee running, every year, for as long as the logic on top of it runs.

Start with one guarantee

Take the easiest one to state: when a run stalls at 2 a.m., someone answers. Google's *Site Reliability Engineering* puts the minimum for a sustainable on-call rotation at eight engineers for a single-site team, so that each is on call one week a month, or six per site when two sites share the load.[4] That is one of the 14 guarantees above, and it only staffs the pager. Nobody has patched, audited, upgraded, or scaled anything yet.

8	6 + 6	1 of 14
Engineers, single-site rotation	Engineers across two sites	Guarantees staffed

Before you write a line of business logic,
the pager alone needs a rotation of eight.

What tokens don’t buy

None of these is a project with an end date. Each is a standing commitment, and together they are the platform team a company signs up for when it builds the platform as well as the logic. Your business logic isn’t on the list. That part was always going to be yours.

Audit cycles

SOC 2, penetration tests, and customer security reviews recur every year, and each one needs evidence.

Upgrades

Clusters, libraries, models, and connected systems all change underneath you.

Incident response

Finding, fixing, and explaining failures, with a timeline an auditor will accept.

Scale you haven’t seen yet

Quarter-end volume, a new region, a tenant ten times bigger than the last.

Three ways

Section 05 of 09

Three ways to get business logic into production

Every company that wants AI to run real processes ends up choosing one of three paths, whether it decides explicitly or not.

	Buy SaaS or point solutions	Build everything	Build the logic on a platform
Who owns the logic	The vendor	You	You
How logic is expressed	Configuration: menus, picklists, designers	Code your team maintains	Fully, in a form the business can read
Platform guarantees	Included	Yours to build and staff	Included
Who you need to hire	Admins and consultants	A platform team, on call	Builders who know the business
Year two	Workarounds around the menus	You are running a platform	You are changing logic
Where it goes wrong	Your process bends to the product	You become a SaaS vendor with one customer	A platform that quietly constrains the logic

Only the third lets builders build the part that matters without turning the company into a platform company. Its one failure mode is worth naming: a platform that claims to leave the logic to you but quietly constrains it. That is what the next section tests for.

The test

Section 06 of 09

A platform that leaves the logic unspecified

Not every platform passes. The test is simple to state: it should specify everything except your business logic, and constrain nothing about it.

Test 1 of 5

The logic is fully expressible, not configured.

If a rule can't be written because a menu doesn't offer it, the platform is SaaS in disguise.

Test 2 of 5

The logic is readable by the business.

Builders write it; finance, operations, and auditors can read it. At Kognitos this is English as Code.[3]

Test 3 of 5

SaaS-grade guarantees come underneath, by default.

Deployment, scale, security, governance, audit, monitoring, and SLAs, without the customer staffing them.

Test 4 of 5

It is deterministic where it matters.

AI reads and drafts; approved logic runs the same way every time, and the system stops and asks when the model isn't sure. See what deterministic AI means.[2]

Test 5 of 5

Exceptions are part of the platform.

When real work doesn't fit the logic, a person resolves it, and the answer becomes reviewed logic.

That clean separation is what Kognitos is built around. The platform carries every guarantee SaaS used to carry, and the only thing it leaves unspecified is the business logic you should own.

The platform you already own

Section 07 of 09

What about Copilot Studio and Power Automate?

Many builders already build in Microsoft's tools, and many CIOs will reasonably ask whether they already own the platform. Run them through the same test.

Kind of tool	Good at	Against the test
Low-code flow builders, such as Power Automate	Routine glue across Microsoft 365, approvals, notifications	The logic lives in the designer's boxes and expressions. That is configuration again, and it strains on exception-heavy, auditable logic.
Copilot-style agent builders, such as Copilot Studio	Conversation, Microsoft 365 grounding, fast agents	A model plans the control flow at run time. Copilot Studio's own architecture guidance recommends enforcing business invariants in deterministic logic as well.[1]
A platform for business logic	Rules that must run the same way, exceptions with people, audit and replay	Specifies everything except the logic, and constrains nothing about it

These aren't either-or choices. Copilot Studio can be where people ask, and Kognitos automations can be called from it, so the conversation lives in Copilot and the business logic runs, governed, in English as Code. For a detailed comparison, see Kognitos vs Power Automate.

The exception

Section 08 of 09

When building the platform yourself is the right call

There are honest cases for building the platform too. When the platform is your product. When you already run a platform team at scale, and one more guarantee costs you little. When a constraint no vendor can meet forces your hand.

Outside those cases, building the platform is a decision to enter a business you didn't plan to be in, and to stay in it for as long as the logic on top of it runs.

Before you build

Section 09 of 09

Five questions for CIOs and CFOs

01	Who will own this logic, and can the business read it?
02	Who is on call for it at 2 a.m., in year three?
03	When an auditor asks why a decision was made, how long does the answer take?
04	How many people does this become in year two?
05	When the logic needs to change next quarter, who changes it, and through what?

Conclusion

Build the logic. Buy the platform.

Builders are right: the logic is theirs, and AI lets them build it at a speed no SaaS roadmap will match. CIOs and CFOs are right too: the company shouldn't turn itself into a software vendor to get there.

Both are true at once if the line is drawn in the right place. **Own every rule that makes your business yours. Buy every guarantee that keeps it running.**

References

Evidence

[1] Candede, D. *Copilot Studio Architecture: When to Use Agents, Workflows, or Both.* On enforcing business invariants in deterministic logic as well as agent instructions.

[2] Kognitos. *What Is Deterministic AI?* Definition, the temperature-0 myth, and how to prove deterministic execution.

[3] Kognitos. *What Is English as Code?* Business logic written in plain English and executed as code.

[4] Spadaccini, A. "Being On-Call." Chapter 11 in Beyer, B., Jones, C., Petoff, J., and Murphy, N. R. (eds.), *Site Reliability Engineering: How Google Runs Production Systems.* O'Reilly, 2016. On the minimum team size for a sustainable on-call rotation.

Get started

Own the logic. Buy the guarantees.

Bring one process your builders want to own. We'll show the logic in English, running on a platform you don't have to staff.

Book a Demo at kognitos.com/book-a-demo