Grid Dynamics white paper

# Governing the variable economics of enterprise AI

www.griddynamics.com

Grid Dynamics

# Contents



**Contributing authors:**

## Nikita Ivanov

VP of Technology/AI | Grid Dynamics

## Kingshuk Banerjee

Account Manager (India) | Grid Dynamics

# Executive summary

An AI initiative can beat every adoption target and still destroy its own margin.

Every prompt, model call, tool invocation, retry, and agent decision adds a variable cost that grows with usage. What looks affordable in a pilot can become a material P&L exposure within months as adoption expands, tasks become more complex, premium models enter the mix, and agents gain autonomy.

This paper explains how to prevent that outcome.

AI FinOps connects consumption to the business result it produces. It attributes cost by workflow, user, model, and outcome, allowing leaders to see which AI use cases generate value, which require redesign, and which should stop consuming budget.

The paper covers:

- Why AI economics differ from fixed-cost software and conventional cloud consumption;
- The four drivers of AI cost volatility: adoption, task complexity, model choice, and agent autonomy;
- The lessons Cloud FinOps offers enterprise AI programs;
- How to build AI run costs, quality controls, human review, and risk into the same financial model as expected benefits;
- The platform capabilities that control cost at its source: context optimization, intelligent routing, and state-preserving orchestration.

***The goal is simple: scale AI use with a defensible connection between every dollar consumed and the business outcome it delivered.***

# AI adoption is shifting to AI usage discipline

The first phase of enterprise AI was about access. Companies handed engineers copilots, coding assistants, and agentic tools and told them to experiment.

That phase is ending.

Across the Fortune 1000 and smaller firms alike, organizations are introducing token budgets, usage limits, approval thresholds, and model-access policies. It marks a shift from experimentation to financial discipline.

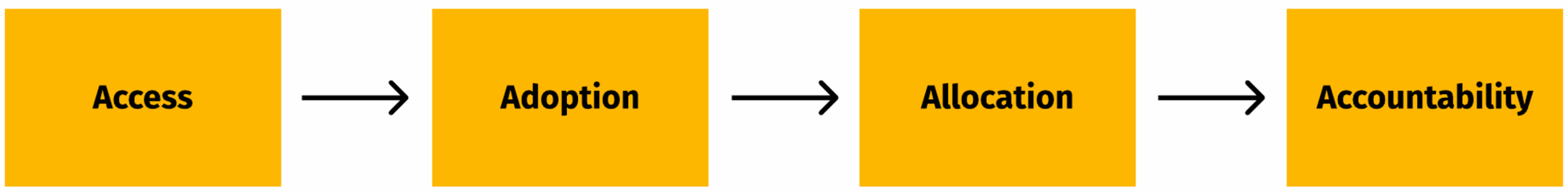
Uber is the clearest public example. The company burned through its entire 2026 AI budget in four months.[1] They responded by introducing a $1,500-per-employee monthly cap on tools like Claude Code and Cursor, tracked through an internal dashboard with approval required for any increase. President and COO Andrew Macdonald put it plainly on the Rapid Response podcast:

> ***"If you're not actually able to draw a direct line to how many useful features and functionality you're shipping to your users, that trade becomes harder to justify."***

Notice how fast the question changed from “how do we get more people using AI?” to “how do we know AI spend is creating enough value to justify it?”

Companies making this shift are routing consumption toward the workflows that pay for themselves, and cutting off the duplicative, low-value use that doesn't.

That path is predictable once you've seen it once: enterprises grant access to AI tools, usage spreads into real adoption, leadership starts allocating that spend toward the workflows worth funding, and the results eventually have to answer to the value they created.




Most of the Fortune 1000 has already cleared access and adoption. The harder work now is allocation and accountability: deciding where AI spend belongs and proving it earned its keep.

AI is now material enough to the P&L that it belongs to the CFO's operating model as much as the CIO's, because for the first time, technology cost tracks *usage* rather than *ownership*, and the bill moves every time someone runs a prompt.

# Why AI costs behave differently

Every technology wave changed the unit enterprises paid for. Infrastructure-era buyers paid for hardware capacity. Software-era buyers paid for licenses and seats. Cloud-era buyers paid for compute, storage, and network consumption.

AI moves the unit again. You're now paying for computational reasoning and, increasingly, for automated action. Consumption is metered in input tokens, cached input tokens, output tokens, model calls, search calls, tool calls, code-interpreter or container sessions, retrieval calls, agent steps, retries, evaluation calls, human-review loops, and orchestration overhead.

## What actually gets metered

AI consumption isn't one number. It gets tracked at three different levels, from the smallest unit up.

| Token-level<br>Metered on every request | Call-level<br>Metered per action taken | Workflow-level<br>Metered across the full run |
|---|---|---|
| Input tokens | Model calls | Code-interpreter or container sessions |
| Cached input tokens | Search calls | Agent steps |
| Output tokens | Tool calls | Retries |
| | Retrieval calls | Evaluation calls |
| | | Human-review loops |
| | | Orchestration overhead |

*Thirteen meters, one bill, and none of them show up on a per-seat license.*

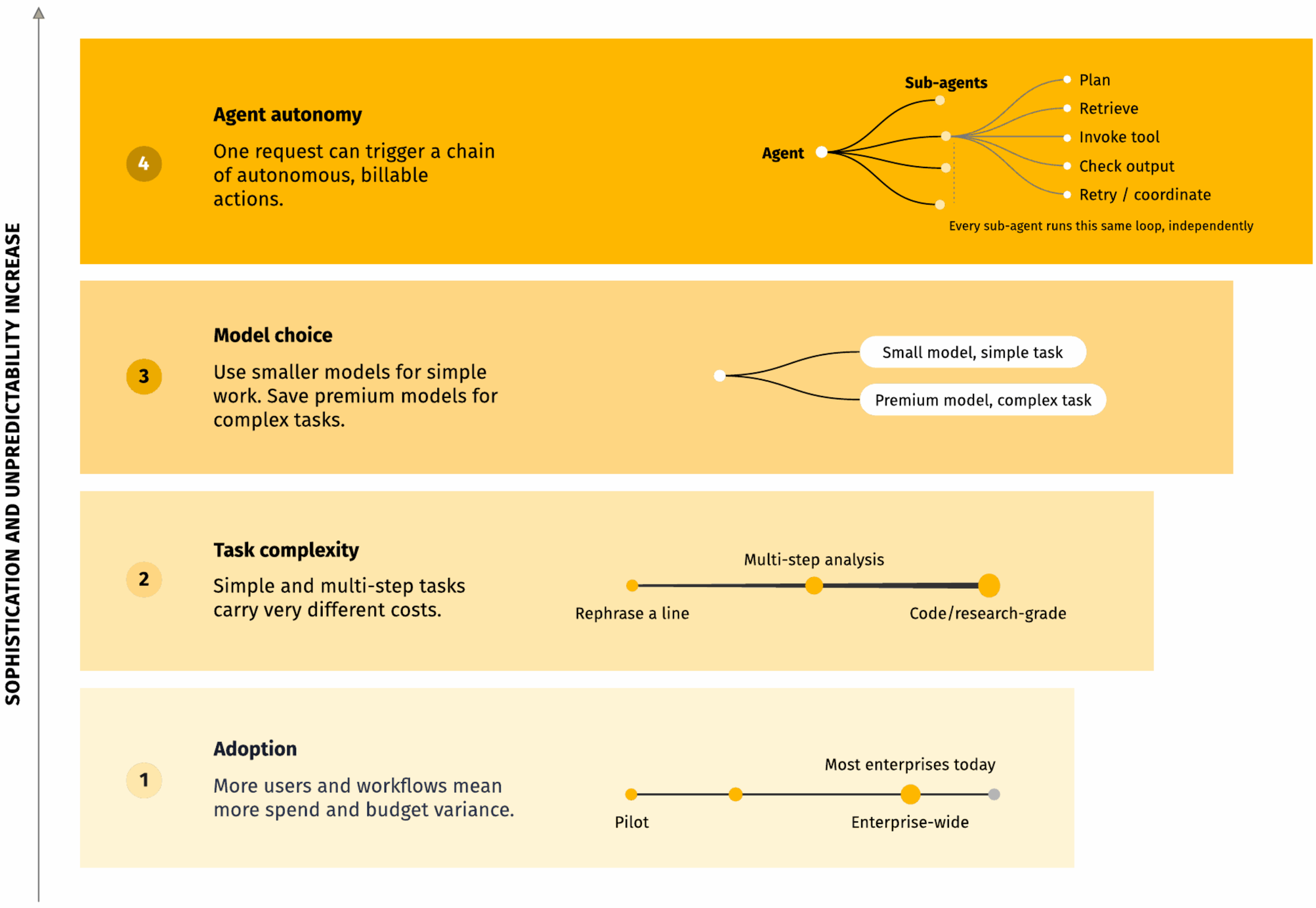
Vendor pricing already reflects this. API providers price input, cached input, and output tokens separately; web search is billed per thousand calls; containers are billed by session and memory. Azure OpenAI's quota system runs on tokens-per-minute and requests-per-minute limits, and Microsoft's own documentation acknowledges customers use those limits specifically to manage billing exposure.[2] AWS Bedrock's invocation logs expose input and output token counts for the same reason: enterprises need that data for cost attribution, as well as performance monitoring.[3] Google's Gemini API runs the same logic through requests-per-minute, tokens-per-minute, and requests-per-day limits, and its documentation is explicit that spend-based limits exist to protect against unexpected charges and to manage load.[4]

AI cost behaves as a usage-indexed operating variable, and agentic AI makes it worse by letting software generate its own additional consumption. A user starts one task, and the agent decides, on its own, to retrieve more information, query another system, call a more expensive model, retry a failed step, break the task into subtasks and route them to sub-agents that make those same calls independently, ask a second agent to check its work, or escalate to a human. Two sub-agents can end up fetching the same data instead of splitting it, and one can rerun an entire workflow because a single step looked uncertain.

One request becomes a chain of cost-bearing events that nobody explicitly authorized, each one adding its own tokens, memory, and model calls.

## Four factors drive cost variability

Each factor adds its own layer of unpredictability. Adoption is the most settled. Agent autonomy, at the top, is the least bounded and the hardest to forecast.




*Read bottom to top: each factor sits on top of the one below it. Adoption is largely resolved for most enterprises today. Task complexity and model choice are manageable with the right routing discipline. Agent autonomy is the newest and least bounded factor, and the one most likely to produce cost nobody explicitly authorized.*

Architecture decides how hard those four factors bite.

Take AI-assisted coding as an example. A chat-based coding agent sits outside the editor and has to rebuild project state, file trees, diffs, build logs, and prior conversation history on every single turn. That reconstruction compounds in multi-agent setups where each sub-agent replays the same context. An IDE-centric agent skips that. Embedded in the editor, it already has the open buffers, symbol graphs, and test results it needs, so it sends only the delta relevant to the current edit. Same task, same model tier, and one architecture burns tokens per turn while the other burns tokens per edit.

## How coding tool architecture changes the token bill

Chat-based agents rebuild context on every turn. IDE-centric agents share it instead.

| **Chat-based agents**<br>Operate outside the editor | **IDE-centric agents**<br>Embedded directly in the editor |
|---|---|
| Reconstructs file trees, diffs, and build logs on every turn | Reads open buffers, syntax trees, and symbol graphs natively |
| Replays prior conversation history back into the model | Sends only the deltas relevant to the current edit |
| Favors large, monolithic code generations | Multiple agent roles share one live semantic state |
| Multi-agent plans duplicate the same context across agents | Token cost scales with edits made, not codebase size |
| **RESULT**<br>Token thrash: the same code gets re-ingested for every incremental change, and context duplication across agents drives N× token amplification. | **RESULT**<br>Token efficiency through shared state. Cost moves from a project-scale problem to a keystroke-scale one. |

*Context architecture is cost architecture.*

Context architecture is cost architecture, and the same logic applies to every workflow in this paper, code included.

This is why AI economics can't be reduced to token price alone. The unit that actually matters is the completed business workflow. **The question enterprises are learning to ask is “what did this workflow cost end to end, and did it create proportionate value?”**

# Cloud FinOps was the dress rehearsal

Traditional IT economics followed a simple sequence: buy, own, use. A company buys 1,000 servers for $5 million. Whether those servers run at 40% or 80% utilization, most of that cost is already spent. AI economics runs the opposite sequence: use, then pay. Every API call, generated token, inference request, GPU-hour, or agent workflow step can add cost the moment it happens.

The early cloud conversation was dominated by one question: why is the bill climbing so fast? Companies had treated cloud as a cheaper substitute for owned infrastructure. They learned, the hard way, that it was a different economic model entirely, one that annual infrastructure budgets weren't built to manage.

That reckoning produced Cloud FinOps, giving enterprises a way to govern variable infrastructure spend by connecting engineering decisions, usage data, financial accountability, and business value. Its controls, from tagging and allocation to anomaly detection and unit-cost reporting, made cloud consumption visible and manageable. AI FinOps applies that discipline to reasoning and autonomous action. It traces consumption through the workflow itself, connecting spend to the model, user, process, output, and business value it produces.

## Cloud FinOps vs. AI FinOps

Same governance instinct, applied to a completely different unit of consumption.

| **Cloud FinOps**<br>Governs what you provisioned | **AI FinOps**<br>Governs what you actually did |
|---|---|
| Workload tagging | Spend visibility by business unit, workflow, model, user, and customer |
| Cost allocation | Agent steps per execution |
| Showback and chargeback | Retry and failure rates |
| Rightsizing | Human-review rates |
| Reserved capacity | Cost per accepted output |
| Autoscaling controls | Cost per completed workflow |
| Budget thresholds | **Value generated per dollar of AI consumption** |
| Anomaly detection | |
| Unit-cost reporting | |

*Cloud FinOps tracks what you provisioned. AI FinOps tracks what you did with it, call by call, retry by retry.*

Cloud FinOps earned its keep at the point of purchase, in how a project got funded in the first place, as much as in how it got monitored afterward. **AI FinOps has to work the same way, starting with the business case.**

# Building an AI-native business case

AI's P&L impact is real, but it's not a balance-sheet story in most cases. Token and API spend generally flows through the income statement as operating expense or cost of revenue. Balance-sheet impact only shows up if you're capitalizing platform development, buying GPU assets, or committing to infrastructure you'll depreciate later.

The managerial problem is P&L volatility: you can know your revenue forecast and still not know the AI consumption intensity required to deliver it.

Most AI business cases still use single-point assumptions for cost and productivity, as if AI behaved like a fixed subscription or a fixed delivery team. This is where the model breaks. A traditional business case says: “we'll spend $X, generate $Y in productivity, and earn $Z in return.”

> ***An AI-native business case has to ask a harder question: at what level of adoption, model mix, agent autonomy, retry rate, output quality, and human review does this initiative stay margin-accretive?***

That means modeling the AI run-cost line explicitly in your net present value (NPV) math, not folding it into a generic technology budget:

```
None
```

```
AI Project NPV =
Σ [
 Revenue uplift
+ Cost takeout
+ Risk reduction
+ Working-capital or cycle-time benefit
- AI run cost
- Integration and platform cost
- Governance, monitoring, evaluation cost
- Human review and escalation cost
- Error, rework, compliance, and exception cost
] / (1 + discount rate)^t
- Initial investment
```

The AI run-cost line needs its own formula:

```
None
```

```
AI run cost =
Σ workflows [
executions
× average agent steps
× model route mix
× (input tokens × input-token price
+ cached tokens × cached-token price
+ output tokens × output-token price)
+ search calls
+ tool calls
+ retrieval calls
+ container/session cost
+ logging/evaluation cost
+ retry/rework cost
+ human-review cost]
```

The revenue line can be contracted, budgeted, or forecast with reasonable confidence. The AI consumption line is behavioral. It moves with how people and agents actually use the system, and that's exactly what CFOs are starting to ask about:

- What's the cost per completed workflow, agentic or not?
- What's the cost per accepted output?
- What's the cost per resolved ticket, generated test, completed claims review, or approved transaction?
- What's the P90 and P95 consumption exposure (the 90th- and 95th-percentile cost scenarios), alongside the average?
- What happens if agent loops double?
- What happens if the model mix shifts from a cheap model to a premium one?
- What happens if output quality demands more human review, not less?
- What happens if adoption succeeds and consumption grows faster than the plan assumed?

These are exactly the questions that go unanswered when nobody is watching, and **three companies show what that looks like in practice** →

# Three companies, three different AI cost problems

Uber's $1,500 cap is one entry in a wider pattern, and the shape of the problem changes from company to company.

> ***Walmart built an internal coding tool called Code Puppy and found employees were repeatedly asking it the same questions, duplicating effort and cost. Global CTO and chief development officer Suresh Kumar pushed for smarter usage: "you don't need to keep asking Code Puppy the exact same question."*** [5]

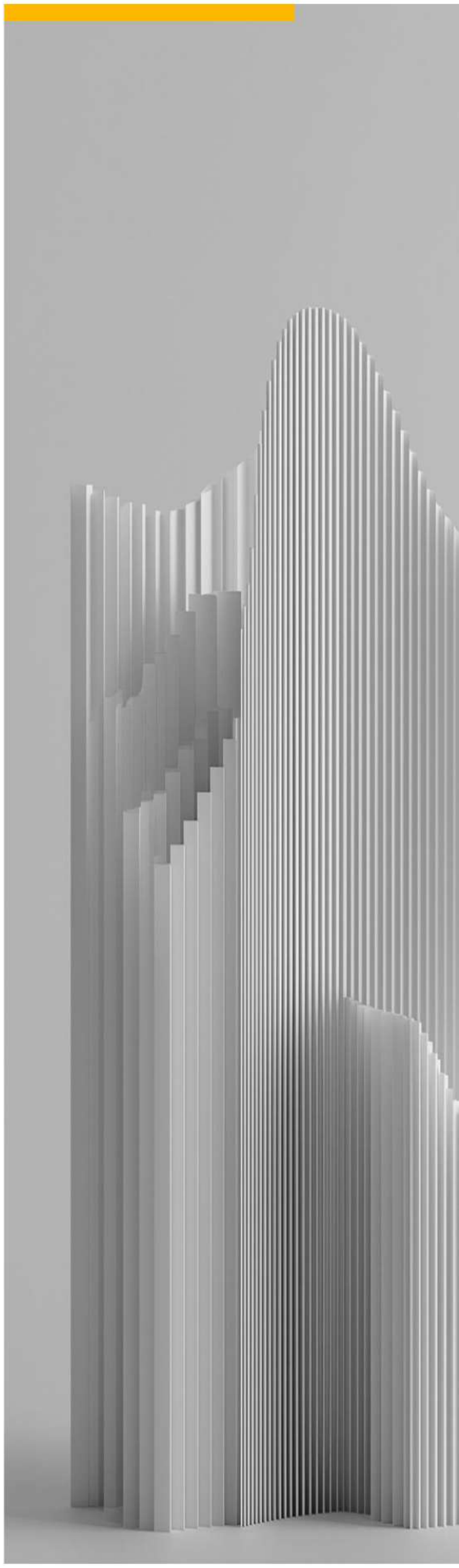
> ***Pega's COO and CFO Ken Stillwell called reasoning-token cost a "black box": "the fact that you cannot predict what it's going to cost to use AI" is what slows adoption down.*** [6]

> ***Meta shows what happens with no governance at all. An employee built an internal leaderboard, nicknamed "Claudeonomics," tracking token consumption across the workforce. In one 30-day window, it logged more than 60 trillion tokens, with the top user averaging 281 billion tokens, an estimated $1.4 million in consumption from one person. Neither Mark Zuckerberg nor CTO Andrew Bosworth ranked in the top 250. The dashboard came down within two days of going public.*** [7]

Three companies, three different symptoms of the same underlying disease: consumption without a governing model behind it.

Deloitte's 2026 research on AI cost, risk, and ROI puts it plainly: "AI is different. Usage-based pricing rises with every drafted email and every summarized meeting." Tech budgets earmarked for AI are projected to climb from 8% to 13% over the next two years, yet only 17% of finance leaders say AI investments have already delivered clear, measurable value.[8] That gap is what's pushing CFOs from enthusiasm to scrutiny.

Uber had the compute, Walmart had the tool, Pega had the access, and Meta had a working product. Each one was missing a governing model connecting spend to outcome, and building that connection is a platform requirement.



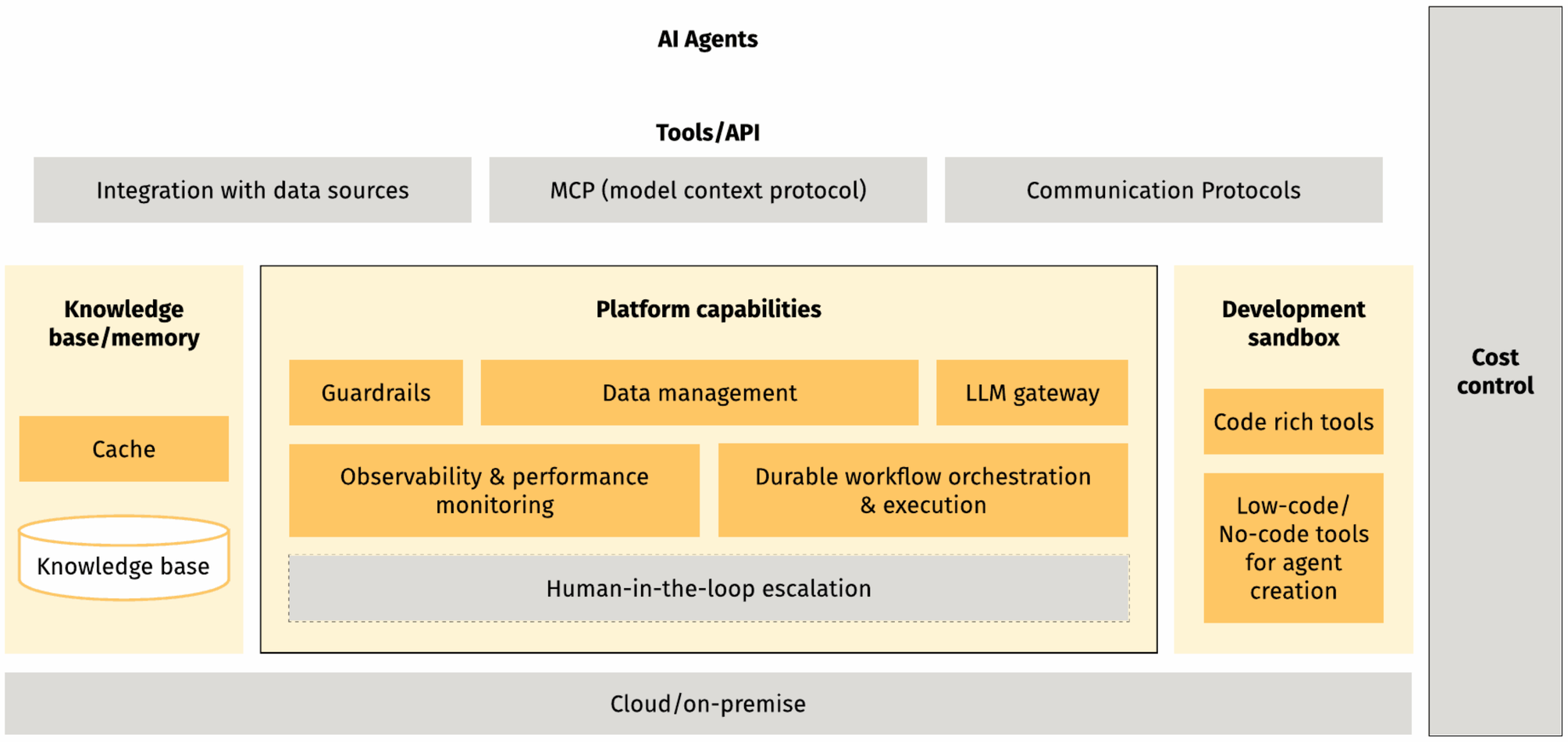
# Architectural requirements for cost governance

That requirement starts earlier than most teams expect. AI FinOps needs visibility after consumption happens, but durable cost control begins in the architecture itself: how a workflow carries context, selects a model, and holds state determines how much compute it burns before a finance team ever sees a charge.

As enterprise agent adoption grows, governance has to scale with it. Gartner puts current AI agent deployment at 17% of organizations, with 42% planning to deploy within the next 12 months and another 22% within the year after.[9]

That's a narrow window to get governance right before adoption outpaces it, which is where the Access → Adoption → Allocation → Accountability ladder from earlier in this paper matters most: cost controls built into the architecture are what turn allocation and accountability from a spreadsheet exercise into something the system enforces on its own.

## Key components of a governed AI platform




*Key components of a governed agentic platform*

Three architectural decisions determine most of that cost before it ever shows up on a bill: how a workflow **carries context**, how it **routes work** to a model, and how it **holds state** while it waits.

## Context architecture

Every workflow needs enterprise context: product data, policies, codebases, contracts, customer records, inventory, pricing, exceptions, and operational rules. That context grounds an agent's decisions and determines whether it produces useful work in a real business process.

It also determines cost, through the same mechanism described earlier: agents that reconstruct context from scratch at every step pay to reprocess the same information repeatedly, and that waste compounds when multiple agents retrieve or replay overlapping context independently.

The same fix applies well beyond coding agents. **A cost-governed platform treats context as a managed resource:** shared context services, semantic caching, retrieval controls, and task-specific context windows cut duplicated token processing while preserving what an agent actually needs.

**Rosetta, Grid Dynamics' open-source governance and context layer built with Anthropic Claude Code**, applies this to coding agents specifically, giving them access to a shared, structured context so they work from the same architectural and codebase knowledge across tasks.

A claims agent, a service agent, or a supply chain agent needs the same discipline: only the policy, customer data, transaction history, and system state relevant to the decision at hand. Limiting context to the active task controls cost, improves response time, and makes agent behavior easier to evaluate.

## Intelligent model routing

Model selection has a direct effect on unit economics. A routing layer assigns each task to the model, tool, deterministic rule, retrieval system, or human reviewer that fits the reasoning, accuracy, latency, and risk the task actually requires.

- **Straightforward classification, extraction, and summarization** can often run on smaller models.
- **Higher-capability models handle complex reasoning, synthesis, code generation, and high-risk decisions**. Retrieval resolves many information requests without heavy model reasoning.
- **Deterministic rules** handle known policy conditions.
- **Human escalation** remains the right call for decisions involving material risk, ambiguity, or customer impact.

**Routing policies need to account for the full cost of the workflow:** token consumption, tool calls, agent steps, retry rates, output quality, human-review requirements, latency targets, and business value. That's what creates a measurable basis for cost-per-completed-workflow and cost-per-accepted-output.

**Deployment location** belongs in the same routing decision. Hosted models produce variable API costs that track consumption. Open-source models running on enterprise infrastructure shift the economics toward GPU utilization, throughput, latency, and capacity thresholds, and cost shifts from linear to stepwise: flat until utilization crosses a threshold, then expensive because it triggers new capacity.

The effect varies by workflow:

- **Latency-sensitive work like autocomplete** and inline edits is hardest to serve well on-premises.
- **Chat, refactoring, and codebase Q&A** tolerate the added latency and run cost-effectively on-premises.
- **Multi-step, agentic coding** is token-intensive on hosted models and capacity-intensive on-premises, which tends to encourage more retries and deeper planning loops.

**An LLM operations layer** brings that decision into daily operations: multi-provider connectivity, load balancing, failover, guardrail filtering, semantic caching, and token-level cost tracking, so the cost-per-workflow number stays a continuously tracked, running metric. That's also where the budgets, caps, showback, chargeback, and anomaly detection that cloud FinOps built for infrastructure spend now apply to model spend, priced per workflow instead of per server.

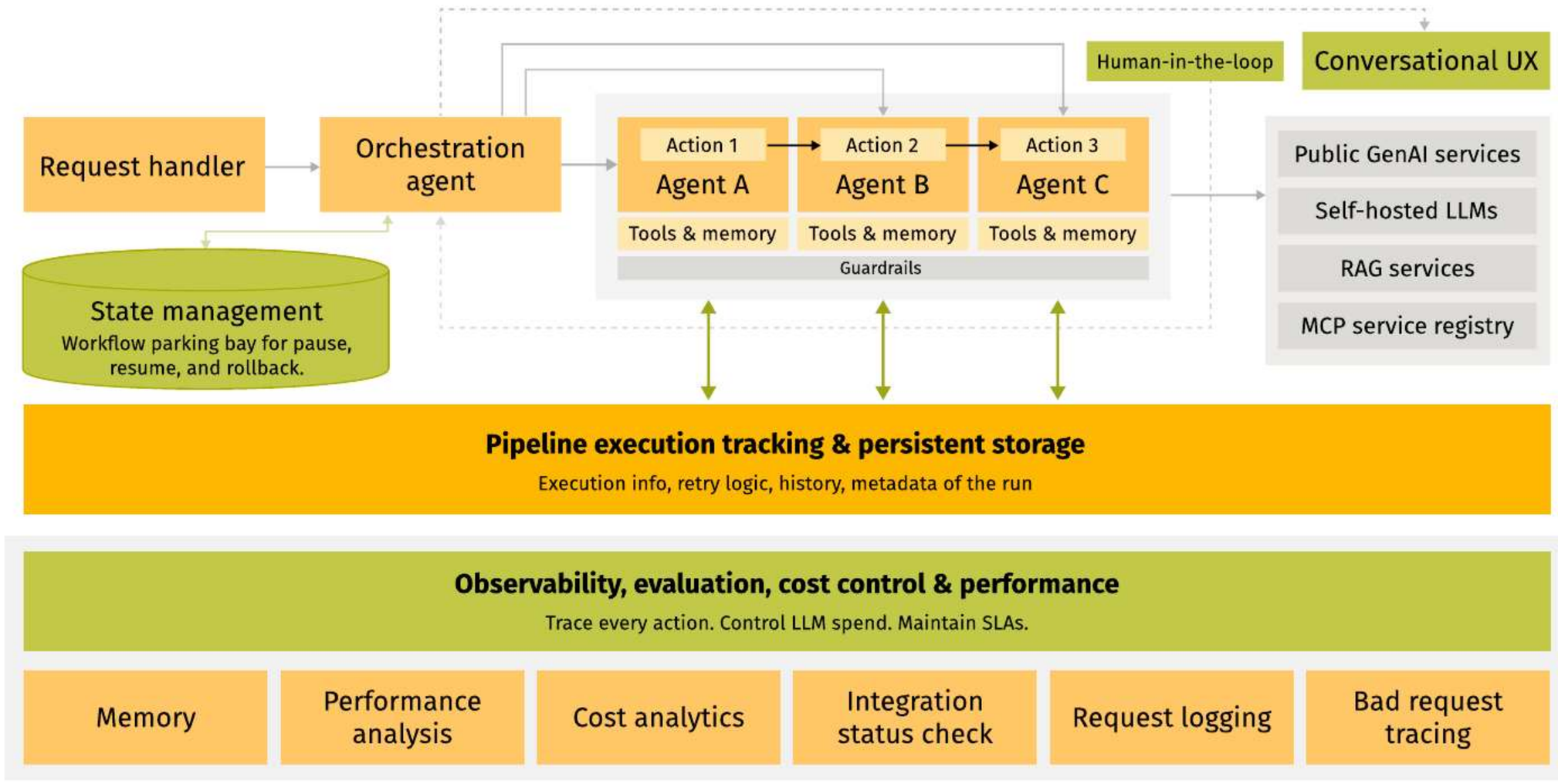
## State preserving architecture

Agentic workflows chain together model calls, retrieval, tool use, approvals, system updates, retries, exception handling, and human review. Their cost profile depends on how the orchestration layer manages state across all of that.

**A state-preserving execution engine** records a workflow's progress and pauses it safely at an approval, external dependency, or exception. While paused, the workflow consumes zero compute. When work resumes, the engine restores the state and continues from the relevant step, instead of keeping the full runtime, network connections, and LLM context active for every pending request, which is what a traditional architecture does.

That pause-and-resume model also creates an auditable execution history: every model invocation, tool call, retry, escalation, and outcome ties back to a single workflow execution. That history is what makes cost attribution possible, and it supports the metrics this section keeps returning to: cost per workflow, cost per accepted output, cost by department or project, agent-step volume, failure and retry rates, human-review rates, and value generated per dollar of AI consumption.

**A durable-execution platform** operationalizes this through failure-tolerant state management, human-in-the-loop workflows, cost alerting, and integrations with enterprise systems like Salesforce, SAP, and Informatica, so agents work against real proprietary data while teams keep visibility into cost, quality, policy adherence, and business outcomes.

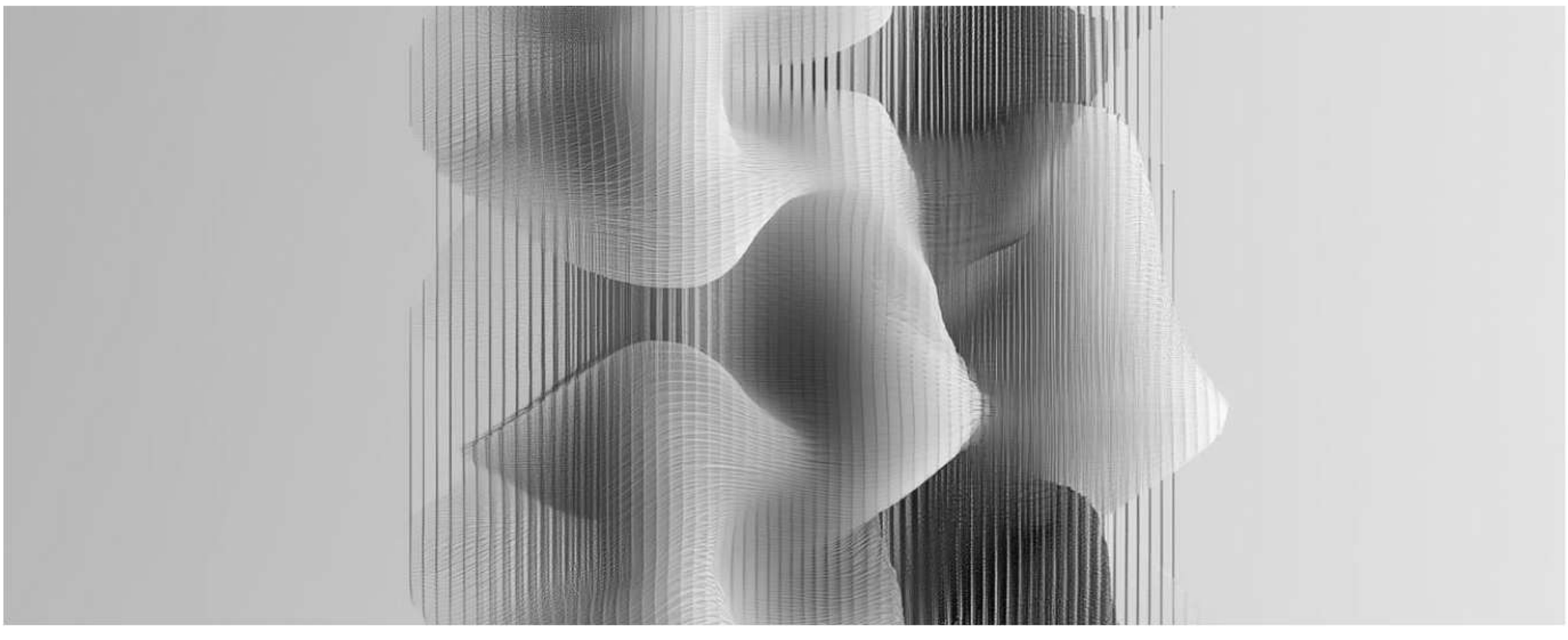



*A representative agentic architecture, showing the orchestration, guardrail, and execution chain*

**A spec-driven AI SDLC platform** applies the same discipline earlier in the delivery lifecycle: formal specifications, standards enforcement, sandboxed agent environments, and validator-driven loops constrain agent behavior before code reaches production, cutting rework, wasted execution cycles, and compliance exceptions across everything the enterprise eventually runs on top of it.

Same discipline, applied at different points in the same pipeline.

The architecture is what makes AI cost governable at the source: by controlling the context agents process, the resources they invoke, and the execution paths they follow.



# The results

**90% faster knowledge access, and a 30-50% cut in manual agent deployment effort** for a Fortune 500 manufacturer building deep research agents across 100+ plants and 5,000 daily users.

**READ CASE STUDY →**

**$9M-$14M in estimated annual savings, with analysis cycles cut from 4-6 weeks to a few hours** for a global payments leader modernizing operations with a Temporal-powered multi-agent platform.

**READ CASE STUDY →**

**3 days of AI-first work delivered 9 weeks of value** for a major health insurance SaaS provider working under strict compliance constraints.

**READ CASE STUDY →**

**Half the team, 3x faster delivery, and a 65% infrastructure cost reduction** for a leading auto parts retailer modernizing a legacy replenishment system across thousands of stores.

**$7K total AI spend for 3 engineers to rebuild seven automation services from undocumented COBOL, delivering over 240K+ lines of code in 6 months** for a Fortune 500 home improvement retailer.

**+30% productivity gain across the engineering organization** for a large digital banking platform modernizing a legacy codebase with no documentation and no clear owners.

These results happen when context, routing, and orchestration are governed from the start. The enterprises that win can show a defensible line from every dollar of AI consumption to a business outcome. Spending the most on AI, or restricting it hardest, doesn't guarantee that on its own.

# Where this leaves you

> ***AI changed what you're actually paying for: reasoning and automated action, priced by consumption instead of access. Cloud computing went through the same shift. The enterprises that handled it well understood their unit economics early enough to scale consumption without losing control of quality, accountability, or margin.***

The next phase of enterprise AI will be judged by one question: what did your tokens actually buy?

If you're building the business case for your next AI initiative, the AI run-cost formula above is a reasonable place to start, before the budget commitment, not after. Grid Dynamics AI ROI consulting works through exactly this kind of unit-economics assessment with clients evaluating AI-native delivery. If that's useful to you, let's talk.

# References

[1] Uber's AI budget cap: TechCrunch, "Uber caps employee AI spending after blowing through budget in four months" (June 2026); Bloomberg, "Uber Caps Usage of AI Tools Like Claude Code to Manage Costs" (June 2026). Uber COO quote: Fortune, "Uber's COO says it's getting harder to justify the company's AI spend: 'That link is not there yet'" (May 2026), reporting his comments on the Rapid Response podcast.

[2] Microsoft Learn, "Manage Azure OpenAI in Microsoft Foundry Models quota."

[3] Amazon Bedrock documentation, "Monitor model invocation using CloudWatch Logs and Amazon S3".

[4] Google, "Rate limits" (Gemini API documentation)

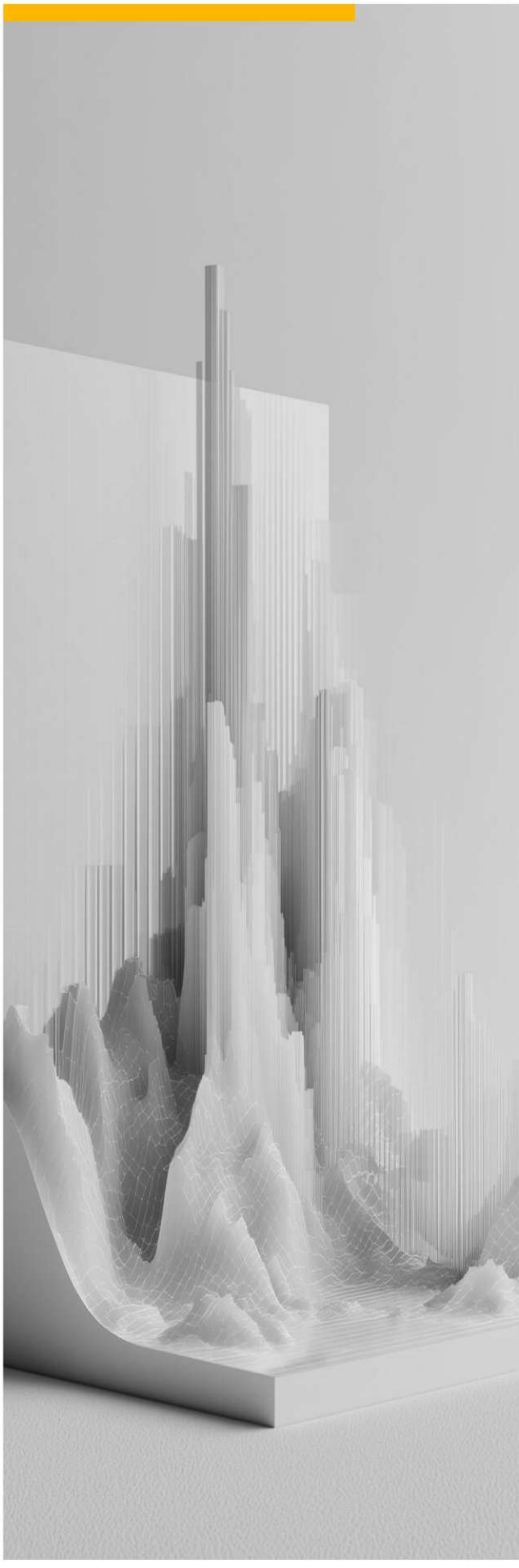
[5] Modern Retail, "Walmart executives see the promise of AI, but also the costs."

[6] AI Business, "As AI Spending Climbs, Enterprises Get Serious About Token Costs."

[7] Fortune, reporting on Meta's internal AI token-usage leaderboard, "Claudeonomics" (April 2026).

[8] Deloitte, "As AI enters a new phase, CFOs gather useful intelligence" (2026 CFO Insights on AI cost, risk, and ROI).

[9] Gartner, "2026 Hype Cycle for Agentic AI | Gartner" (April, 2026)

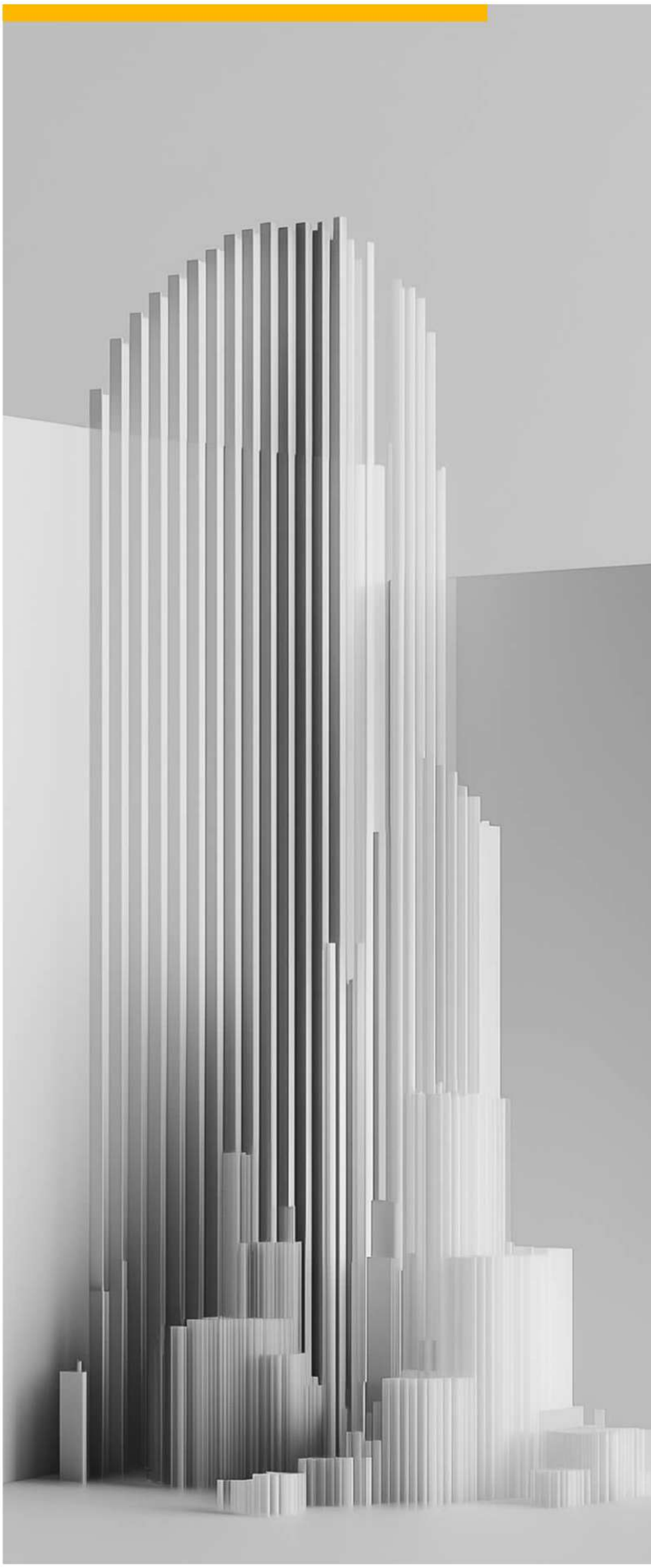


# About Grid Dynamics

Grid Dynamics (Nasdaq: GDYN) is a premier AI transformation partner to the Fortune 1000. We combine deep AI expertise with proven enterprise-scale delivery to help clients identify where to invest and build AI systems that drive growth, improve efficiency, and deliver measurable outcomes.

Our proprietary GAIN (Grid Dynamics AI Native) platform suite accelerates this transformation with domain-specific solutions—purpose-engineered for scale, security, and ROI across areas including agentic commerce, risk and compliance, and physical AI. We bring nearly a decade of enterprise AI leadership, supported by continuous investment in data, cloud, and platform engineering.

Founded in 2006 and headquartered in Silicon Valley, Grid Dynamics operates globally with offices across the Americas, Europe, and India. Learn more at griddynamics.com and follow us on LinkedIn.



**LET'S TALK AI TOKEN ECONOMICS →**

# Grid Dynamics

trusted engineering partner for digital transformation


**Grid Dynamics Holdings, Inc.**

6101 Bollinger Canyon Road, Suite 465,

San Ramon, CA 94583

www.griddynamics.com