

AICOST.AI

CLOUDINTELLIGENCE.AI LLC

Irvine, California ~ Austin, Texas

Cost and governance for AI workflows in the enterprise

Agentic loops. RAG. Fine-tuning. Voice. Evaluation. Human review. Physical AI.

Cost and governance decision engines that model the full economics of an AI workload, turn the approved decision into enforceable policy, and prove the outcome against it.

WITH AICOST.AI

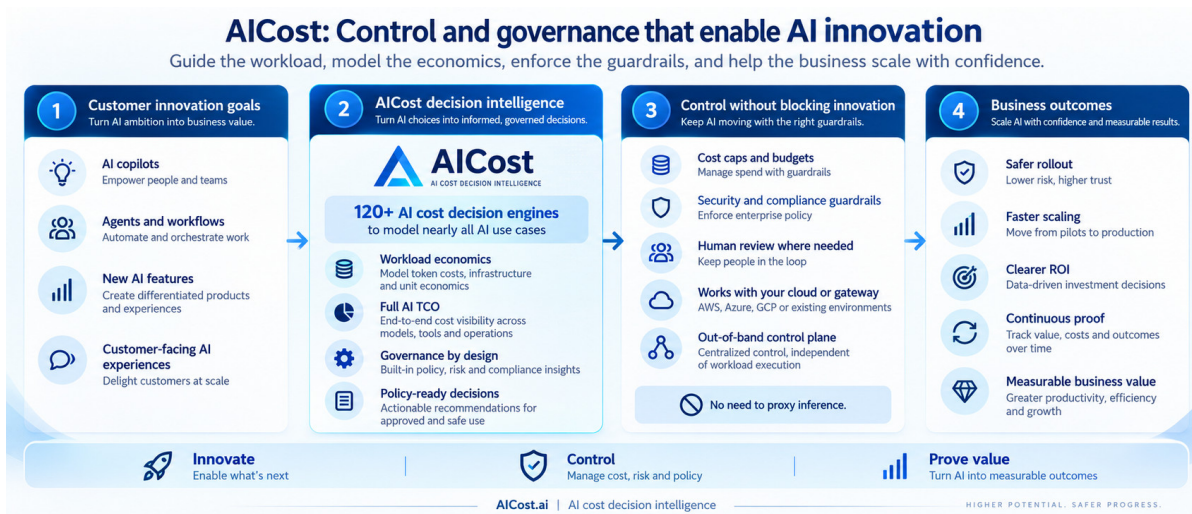
**Innovate with AI in confidence across your
enterprise, with governance and cost
controls from day one**

1. Why we built AICost.ai

Subu Vdaygiri, Founder, AICost.ai

“AI cost governance has to begin before deployment, not after the bill arrives. AICost starts with the business workflow and the AI workload an enterprise wants to put into production or scale. We help determine which models, routes and architectures are eligible, model the full economics including inference, evaluation, security, observability, human review and governance, and turn approved decisions into policy that can be enforced through the customer existing cloud, gateway or CI/CD environment.

Once the workload is live, AICost reconciles actual cost and operating evidence against the approved baseline, identifies drift, and keeps the economic and governance picture current. The enterprise retains control of its runtime and makes the final decisions. AICost provides the decision intelligence, evidence and continuous control loop.”



“We built AICost around AI workload economics from the ground up, rather than treating AI as another cloud-spend category with token accounting added. From my experience working on AWS, Azure and GCP cloud cost optimization with enterprise customers over the past decade, AI is a different class of workload. Usage can be less predictable, models and data can drift, agents can create variable execution paths, and enterprises must manage multiple providers, rapidly evolving models, frameworks and tools.

That means the economics cannot stop at token costs. Successful production AI also carries the cost of evaluation, observability, security, human oversight, data pipelines, model maintenance, integration and governance. We have built more than 120 live cost and governance decision engines to model these decisions across a broad range of AI workloads. Those engines can be composed into customer-specific workflows and connected to policy controls through MCP, APIs, CI/CD and existing cloud or gateway environments. Our goal is to help enterprises move from pilot to production and scale with the economics, controls and evidence required to make sound business decisions.”

Subu Vdaygiri | Founder, AICost.ai | CloudIntelligence.ai LLC

2. Twelve operating principles

What AICost assumes about AI workloads, and how each assumption changes a decision

“For us, AI cost management is not about finding the cheapest model. It is about finding the architecture, model mix and operating controls that allow the workload to deliver its business outcome at the right economics and an acceptable level of risk.”

Subu Vdaygiri, Founder, AICost.ai

#	Principle	What it means in practice	The claim
1	Workload first	Start from the use case, not the cloud bill. Model user behaviour, quality needs, latency targets and compliance boundaries. A support assistant, a coding agent and marketing content have different economics	Built for AI workloads
2	Full AI TCO	Include reasoning, context, embeddings, vector database, OCR, evaluation, observability, human review, guardrails, integration, retraining and maintenance. A support bot cost per resolved case includes retrieval, generation, review and refresh	Beyond token costs
3	Plan before spend exists	Forecast pilot, production and scale before the first invoice. Estimate development cost for five hundred developers before the rollout rather than after it	Plan before rollout
4	Cost and governance together	Governance starts on day one. Model eligibility, human oversight and audit needs are part of the workload design. A regulated workload may require approved models, residency rules and human review	Governance by design
5	Deterministic decision engines	Reproducible cost models that are auditable and machine callable. The same inputs yield the same output, assumptions and evidence, and can be re-run with pinned pricing	Auditable decision logic
6	Policy-ready outputs	Decisions become operational guardrails: budget caps, model allowlists, route rules, turn limits and approval gates. An agent cost cap per task with a fallback to a cheaper approved model	Operationalize decisions

#	Principle	What it means in practice	The claim
7	Out-of-band control plane	Guide and govern without sitting in the inference path. Apply policy through LiteLLM, Portkey, Bedrock, Azure AI, Vertex AI or direct APIs	Works with your stack
8	Compose, enforce, prove	Chain engines into workflows, enforce selected controls at runtime, then prove outcomes with actuals and evidence. Token estimator plus router plus cost wall plus proof is a full operating loop	Full operating loop
9	Forecast to actual	Close the loop between modelled economics and real usage. Reconcile bills, telemetry, attribution and showback against the original plan, by team, project, customer, model or provider	From forecast to proof
10	Vertical and function aware	Departments and industries need different economics and controls. A healthcare prior-authorisation agent, marketing video generation and an engineering copilot should not be modelled the same way	Economics by context
11	Partner with the customer	Workflows, costs and governance are intertwined with each customer business processes, data and operating model. A regulated claims process is co-designed rather than configured	Partnership-led delivery
12	Customise where it matters	Use the platform as a base, then tailor engines, parameters, policy packs and runtime rules to the customer requirements, including custom engines with customer-specific variables	Configured to fit you

What these add up to for a customer

Better architecture choices, a safer pilot-to-production rollout because governance is designed in rather than retrofitted, clearer ROI and unit economics, and continuous governance and proof.

12 Core AICost Philosophies

How AICost differs from traditional FinOps for AI workloads

Control the economics and governance of AI workloads from pilot to production.



The economic decision is not just which model is cheapest. It is which architecture and model mix satisfies the workload's **quality, latency, compliance, and scale** requirements at the best economics.

AI TCO includes build, run, evaluate, govern, maintain, observe, and remediate. Beyond token costs.

1

Workload first



Start with the AI use case, not only the cloud bill. Model the workload, user behavior, quality needs, latency targets, and compliance boundaries.

Example:

Customer support RAG, coding agents, or marketing content all have different economics.



Built for AI workloads

2

Full AI TCO



Model more than inference token costs. Include reasoning, context, embeddings, vector DB, OCR, evals, observability, HITL, guardrails, integration, retraining, and maintenance.

Example:

A support bot cost per resolved case includes retrieval, generation, review, and refresh.



Beyond token costs

3

Plan before spend exists



Forecast pilot, production, and scale before the first invoice arrives. Use workload assumptions to estimate TCO, ROI, and variance ranges.

Example:

Estimate development cost for 500 developers using AI coding tools before rollout.



Plan before rollout

4

Cost and governance together



Governance starts on day one. Security, privacy, model eligibility, human oversight, and audit needs are part of the workload design.

Example:

A PHI or PII workload may require approved models, residency rules, and human review.



Governance by design

5

Deterministic decision engines



Use reproducible cost decision models that are auditable and machine callable. The same inputs should yield the same output, assumptions, and evidence.

Example:

A token estimator or agent loop engine can be re-run with pinned pricing and identical logic.



Auditable decision logic

6

Policy ready outputs



Turn decisions into operational guardrails such as budget caps, model allowlists, route rules, turn limits, token limits, and approval gates.

Example:

Set an agent cost cap per task and fall back to a cheaper approved model when needed.



Operationalize decisions

7

Out of band control plane



AICost can guide and govern without sitting in the inference path. It can stay vendor neutral and export decisions into existing runtime systems.

Example:

Apply policies through LiteLLM, Portkey, Bedrock, Azure AI, Vertex AI, or direct APIs.



Works with your stack

8

Compose, enforce, prove



Chain engines into workflows, enforce selected controls in runtime, and then prove outcomes with actuals and evidence.

Example:

Token estimator plus router plus CostWall plus CostProof creates a full operating loop.



Full operating loop

9

Forecast to actual



Close the loop between modeled economics and real usage. Reconcile bills, telemetry, attribution, and showback against the original plan.

Example:

Compare forecast vs actual by team, project, customer, model, or provider.



From forecast to proof

10

Vertical and function aware



Different departments and industries need different economics and controls. AICost adapts to marketing, engineering, healthcare, finance, legal, and support workflows.

Example:

Healthcare prior authorization agents, marketing video generation, and engineering copilots should not be modeled the same way.



Economics by context

11

Partner with the customer



AICost is not only generic SaaS. Workflows, costs, and governance are intertwined with each customer's business processes, data, and operating model.

Example:

Co-design a workflow for a regulated claims process or a secure enterprise coding environment.



Partnership-led delivery

12

Customize where it matters



Use the platform as a base, then tailor engines, parameters, policy packs, and MCP runtime rules to the customer's unique requirements.

Example:

Create a custom engine with customer specific variables and policies for their approval flow, budget rules, and governance needs.



Configured to fit you

What this means for customers



Better architecture choices

Choose the right model mix for quality, latency, and cost.



Safer pilot to production rollout

Reduce risk with built-in governance and controls.



Clearer ROI and unit economics

Make data-driven investment decisions.



Continuous governance and proof

Reconcile, show value, and keep improving.

AICost.ai

AI cost decision intelligence

Compose. Enforce. Prove.

Real AI outcomes
start with better decisions.

3. Why cost and governance start together

The same workload decisions that shape cost also determine risk, policy, quality and proof

The two are usually sequenced: build the thing, watch the spend, add governance when somebody asks. That order is what produces a surprise bill, a risk discovered after deployment, and a governance document that is stale the week it is signed.

Cost-only approach	Cost and governance from day one
Build first, spend appears later, governance added after launch, then surprise bills and an audit scramble	Model the workload, add governance requirements, set runtime policy, operate in production, prove with actuals and evidence
Misses full AI TCO	Plans before the first invoice
Finds risk after deployment	Builds controls into the design
Turns governance into a stale document	Creates a continuous operating loop

Five reasons they cannot be separated

Reason	Why
AI cost starts before runtime	Forecasts are needed before historical spend exists
Governance is part of TCO	Security, compliance, human review, evaluation and observability all add real cost
Cheaper is not always acceptable	Savings must respect quality, privacy and risk constraints
Policies prevent runaway spend	A monthly alert cannot stop a bad agent loop in time
Evidence beats assumptions	Actuals and audit trails prove whether savings and controls worked

The same variables appear in both views

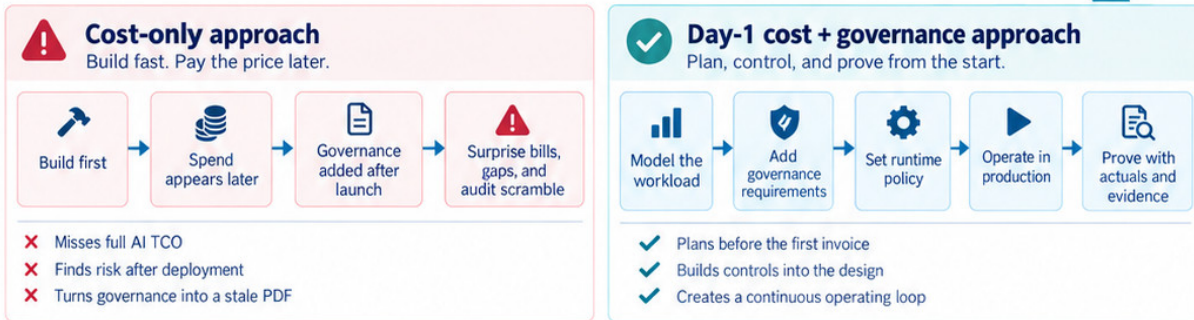
An agent loop has a cost view of turns per task, tokens per turn, tasks per day and runaway probability. It has a governance view of maximum turns, maximum cost per task, fallback model, pause or kill, and the human review threshold. **They are the same variables.** Modelling them once and using them for both is why the policy is consistent with the economics rather than negotiated against them.

Why AI Cost and Governance Must Start Together on Day 1

AI TCO is not just tokens. The same workload decisions that shape cost also determine risk, policy, quality, and proof.

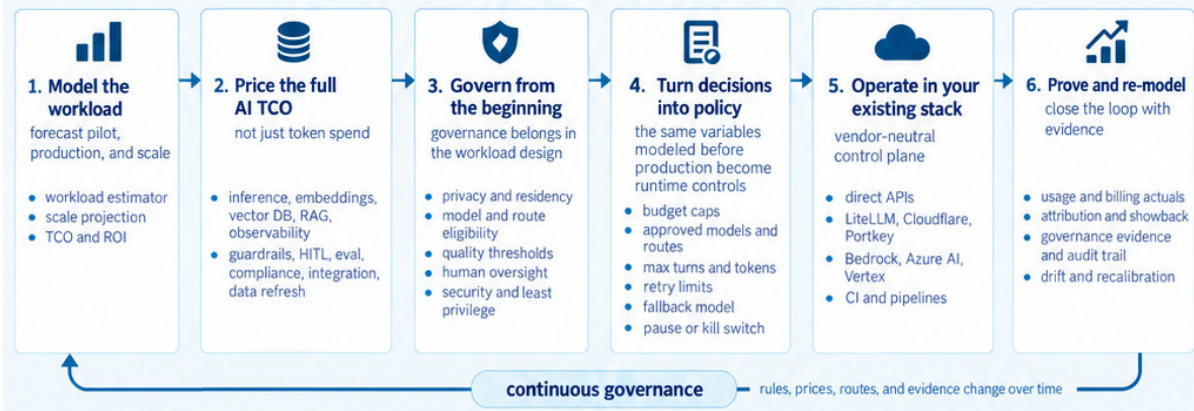
Plan → Govern → Operate → Prove → Re-model

REAL
DECISIONS.
LOWER RISK.
HIGHER IMPACT.



The integrated lifecycle: from design to evidence

The same workload decisions shape cost, governance, and runtime behavior.



Why they must go together



Simple examples



AICost.ai
PLAN | GOVERN | OPERATE | PROVE

123+
decision engines

20
governance modules

247
controls

 Decision logic → runtime policy → CostProof and governance evidence

 From pilot to production, cost and governance work best as one system.

AICost.ai | AI ECONOMICS AND GOVERNANCE FOR REAL-WORLD AI.

HIGHER VALUE. SAFER AI. BRIGHTER TOMORROW.

4. How it runs, and what each part of the organisation gets

An operating rhythm rather than a reporting tool

The central question an enterprise is trying to answer is a single one with five parts: **what may we run, what should we run, what will the governed workload cost at scale, what controls must exist, and can we prove the approved state is still true after go-live.** Today those five parts are usually answered by different people at different times, which is where late rework and unpriced obligations come from.

The loop the engines support



Each stage feeds the next. Re-modelling is scheduled rather than triggered by a surprise, because prices, models and volumes all move.

The full picture is on the next page. Every stage below is backed by the engines, the policy output and the evidence trail described in the rest of this document.

What each role in the decision gets

An AI cost and governance decision usually needs four groups to agree. Each of them wants something different from the same underlying model.

Role	What they get
AI and architecture	A common method for deciding which workloads are worth building, which routes are eligible, and what architecture is economically sustainable at the volumes expected
FinOps and finance	A bridge from provider spend to workload-level forecast, unit economics, the governance premium, showback and realised savings
Engineering and platform	Explainable thresholds that can be promoted into gateway, CI, cloud or application policy without forcing a new inference path
Governance and security	Evidence that policy, route eligibility, human oversight and quality requirements were designed before production and remained in place afterwards
The business owner	A unit that means something in their terms: cost per successful engineering task, cost per resolved support case, cost per contract reviewed. Not cost per token

The unit is the workload, not the model or the bill

A workload carries an owner, a business purpose, a department, a lifecycle state, an architecture, a data class, a risk tier and a business KPI. Cost, policy and evidence all attach to that same object. This is what allows a coding agent, a support assistant and a contract review workflow to appear in one total despite having almost nothing in common as cost shapes.

Where enterprises usually run into difficulty

Failure	What it looks like
Planning	A business case built on token price alone, which omits retrieval, review, evaluation and governance
Pilot to production	A pilot that worked at one team scale and had no tags, so production cannot be attributed or compared

Failure	What it looks like
Scale	Unit cost that looked fine per developer and does not hold at fifty, because usage profile changes with headcount
Operations	A policy agreed in a document with no enforcement point, so drift is only visible in the next invoice

AICost.ai: From AI Workload Knowledge to Runtime Policy and Proof

Learn → Model → Decide → Policy → Operate → Measure → Re-model

Vendor-neutral AI economics and governance decision layer

PLAN
GOVERN
OPERATE
PROVE

MORE THAN
CALCULATORS.
A COMPLETE
DECISION LAYER
FOR REAL-WORLD
AI.

FOR PEOPLE
Explore, model
and make decisions
with interactive
guides.

FROM INSIGHTS
TO ACTION
Compose decision
chains for your
organization.

TURN DECISIONS
INTO POLICY
Translate analysis
into enforceable
guardrails.

RUN ANYWHERE
Work with your
existing AI stack.
Vendor-neutral
by design.

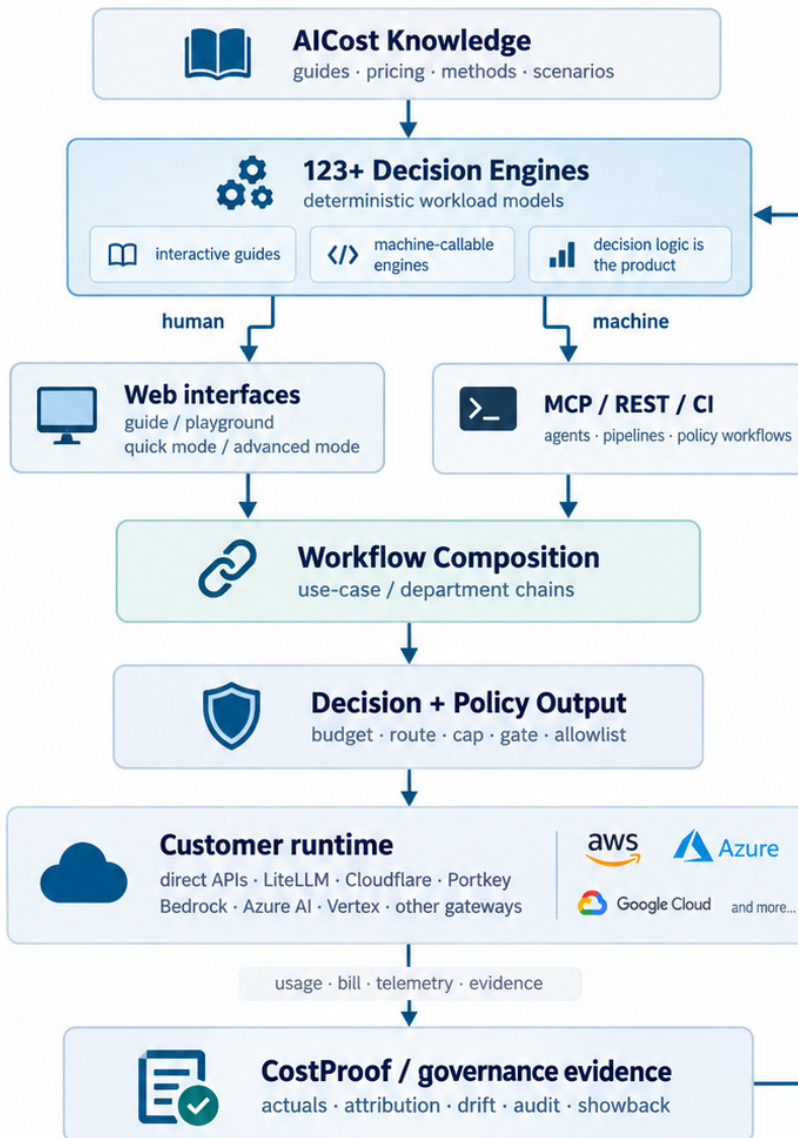
PROVE VALUE
Demonstrate
outcomes with
audit-ready
evidence.

SAME
WORKLOADS.
BETTER
DECISIONS.
LOWER RISK.
HIGHER IMPACT.

FOR SYSTEMS
Integrate into agents,
pipelines and
enterprise workflows.

re-model /
re-policy

CONTINUOUSLY
IMPROVE
Feed real usage
and outcomes back
into better models
and policy.



Example decision chains

REAL ENGINES. REAL DECISIONS. REAL IMPACT.



Example: Token Estimator Engine

token estimator → cost calculator → scale projection

Inputs: requests, input/output tokens, model rate

Outputs: cost/request, monthly budget, approved model



Example: Agent Loop Engine

agent-loop → context compaction → retry cost → multi-model router

Inputs: turns/task, tokens/turn, tasks/day, runaway probability

Policy: max turns, cost cap/case, fallback model, pause/kill



Model economics
before runtime spend exists

Plan with confidence. Avoid surprises.



Turn decisions into
enforceable policy

From analysis to action.



Prove outcomes with
actuals and governance evidence

Audit-ready. Business-ready.

AICost.ai | PLAN DECIDE OPERATE PROVE

WORKLOADS TODAY. A SMARTER TOMORROW.

5. The platform: six pillars, one engine set

Same engines, many interfaces, from planning through to proof

Pillar	What it does	A concrete example
Clarity see and attribute	Understand where AI spend is going, what is driving it, and who is responsible	Detect fourteen shadow AI tools in use, attribute the monthly spend to teams, and flag one tool training on your data
Optimize reduce	Find lower-cost options without compromising quality, latency or compliance	Switch non-critical chat workloads to a smaller model via policy, at materially lower cost with no measured quality drop
Forecast plan and scale	Model AI costs in ranges rather than false-precision points, across teams, use cases and growth scenarios	Forecast a coding assistant for five hundred developers as an annual range, with seat plus agent usage and overage scenarios
Governance risk and control	Apply policy, compliance and risk controls to AI workloads by design	Block personal data from reaching public models, require approval for high-risk use cases, and map to the control framework
CostWall enforce	Turn decisions into enforceable guardrails that can block, cap, route or downgrade in real time	Set a per-user daily cap for agent tools. If exceeded, route to a cheaper model or require approval
CostProof prove	Reconcile modelled against actual spend, show ROI, produce audit evidence and demonstrate value	Show actual against forecast for the quarter, attribute costs to customers and projects, export an audit report for the board

Why the first three are not enough on their own

Clarity, Optimize and Forecast are what most tools in this category offer, and they are necessary. They are also entirely advisory: each produces a recommendation somebody has to act on. Governance, CostWall and CostProof are what make the recommendation stick, stop a workload that ignores it, and prove afterwards whether it worked. The first three sections that follow cover the advisory pillars; sections 19 through 21 cover the three that enforce and prove.

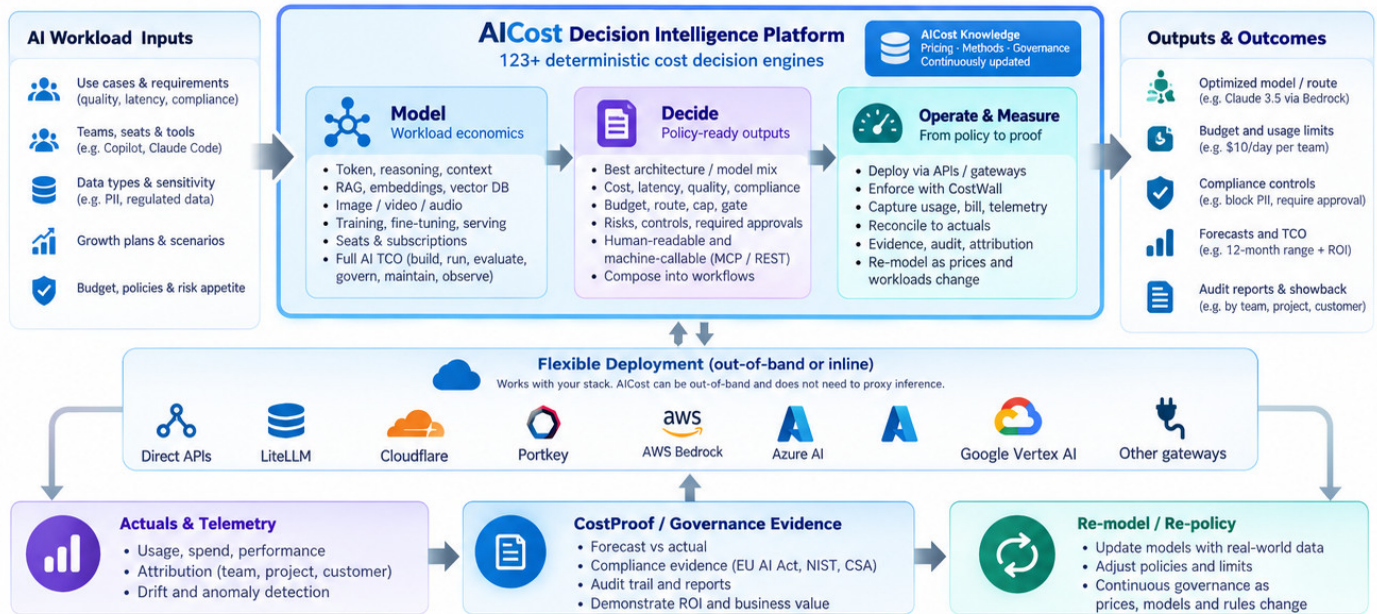
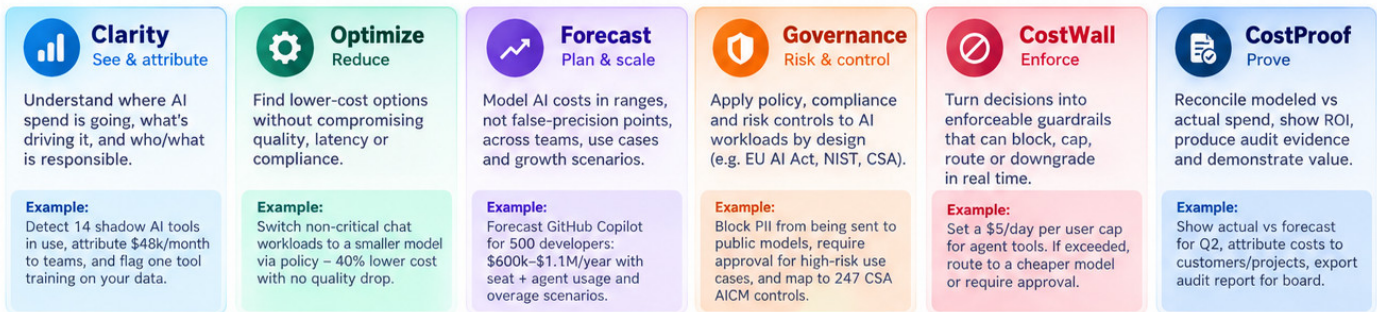


AI Cost Decision Intelligence Platform

Control the economics and governance of AI workloads from pilot to production.

DETERMINISTIC MODELS | POLICY-READY | VENDOR NEUTRAL | FROM PLANNING TO PROOF

SAME ENGINES.
MANY INTERFACES.
REAL OUTCOMES.



Why cost and governance must go together from day 1



AICost.ai COMPOSE · ENFORCE · PROVE | The AI economics control plane for the real world.

COST AND GOVERNANCE BY DESIGN.
PILOT TO PRODUCTION. AT SCALE.

6. Clarity: finding the spike, then preventing the next one

The first of three jobs. Each engine answers one question in plain terms.

The engines cover the range of AI workloads an enterprise runs. They do three jobs. The first is clarity: establishing where the money went and which part of the workload sent it there.

When a bill spikes, these are the usual suspects

Engine	What it tells you
Business AI Bill Diagnose	Takes the bill and names the drivers, rather than leaving you to read a CSV
Agent Loop Cost	What one agent run costs when the turn count is variable, and where the runaway risk sits
Agentic Workflow Cost	Monthly burn across coding assistants, which is usually several vendors at once
Context Window Cost	At what document size the cost per call doubles. The cliff most teams cross without noticing
Reasoning Token Cost	What thinking tokens add on models that bill for them separately
Retry Cost	What failed-and-retried calls quietly cost each month. Billed in full, invisible in most reporting
Token Estimator	Converts a described workload into a token count, so a forecast has a real basis
Token Reduction	Where prompt and context volume can be cut before any model change is considered
AI Margin Calculator	Gross margin on an AI-powered product line once true inference cost is counted

EXAMPLE 1 Last month doubled and nobody can say why.

Chain: Business AI Bill Diagnose → Agent Loop Cost → Context Window Cost → Retry Cost

What comes back: Four candidate causes, priced rather than argued: agentic sessions replacing chat calls, context growth across long sessions, a document size that crossed the cliff, and retries nobody counted.

These engines diagnose. The engines that prevent a repeat are in the next two sections.

Preventing the next one

Diagnosis is retrospective by nature. Prevention is where the same engines are used forward: the ceiling comes from Agent Loop Cost and the variance reserve, the routing rule from the eligibility gates, and the alert threshold from the forecast. Section 20 covers how those become enforced rather than advisory.

7. Optimise: cutting spend without moving the problem elsewhere

The second job. Every lever priced against your shape, and gated on quality.

Engine	What it tells you
Prompt Cache ROI	Whether caching saves money or costs money here, and the break-even hit rate by vendor
Multi-Model Router	What routing easy queries to a cheap model and hard ones to the flagship is worth
Cheapest Model Finder	The cheapest model that still meets the capability requirement, not the cheapest model
Stacked Savings	What the levers are worth together. They compound on what remains, they do not add
Context Compaction	What compaction keeps after its own cost and the re-derivation the agent pays back
RAG Pipeline Cost	The full retrieval stack on one screen: ingest, storage, query and generation
Fine-Tuning Cost	Training plus inference against the base model, and where the break-even volume falls
Self-Host Breakeven	Open weights on rented GPUs against the API, with real operating costs included
Open-Source Model Finder	Which open-weight model fits, and what hosting it actually costs
Chinese vs Western Models	A like-for-like comparison where the rate cards differ by region and host
Vector DB Cost	The retrieval layer, which is usually priced separately and forgotten
Guardrail and Security Cost	What runtime safety screening adds as a percentage of the token bill
LLM Observability Cost	The tracing bill by span, which grows fastest on agentic workloads
Buy vs Build	An outcome-priced vendor against building it, with engineering cost included

EXAMPLE 2 We need 30 percent out of the support assistant without breaking it.

Chain: Prompt Cache ROI → RAG Pipeline Cost → Multi-Model Router → Stacked Savings → Eval Gate

What comes back: Each lever priced separately, then combined correctly, then checked. The eval gate is the last block because a cheaper model that escalates more often has moved cost to the reviewer rather than removed it.

Moving cost from the token bill to the review queue is the most common false saving in this work.

8. Forecast and plan: costing what has not happened yet

The third job, and the one retrospective tooling cannot do at all.

Engine	What it tells you
Scale Projection	What scaling ten times actually costs, including the tier transitions on the way
AI Budget Planner	A defensible budget line for a workload that does not exist yet
Quarterly Spend Forecaster	Where the quarter lands on current trajectory, and when
Overage Forecaster	Whether a plan allowance will be breached, and in which month
Agentic Variance Reserve	The reserve an agent workload needs, because the tail matters more than the mean
Agentic TCO and ROI Builder	All the cost layers of an agent programme rolled into one verdict with payback
AI Seat ROI	Whether a seat licence returns its cost, per person rather than on average
Support Deflection ROI	Cost per deflected ticket, which is the metric support leaders already report
ROI Quick Check	A first-pass business case in the units the business line uses
Annual AI Cost Forecaster	The twelve-month view, with the assumptions visible rather than embedded
AI Cost Diligence	Whether a stated AI cost survives vendor-exact pricing, with a verdict
GPU Cost per Inference	Self-hosted economics at a given utilisation, for the build-or-rent decision

EXAMPLE 3 We want to put an agent in front of 400 support staff next year. What do we budget?

Chain: [Token Estimator](#) → [Scale Projection](#) → [Agentic Variance Reserve](#) → [AI Budget Planner](#)

What comes back: A budget figure with a reserve stated separately, plus the month the plan allowance would be breached if adoption runs ahead of forecast.

The reserve is priced separately because agent workloads exceed budgets through variance rather than through the average.

The decision engines make no model calls

An enterprise evaluating "AI cost intelligence" reasonably assumes another language model is generating the estimates. The decision-engine runtime makes no model calls at all. The same inputs produce the same answer every time, which is what makes a figure reproducible for an auditor and safe to put in a policy. Language models are used elsewhere in the product, for drafting and input suggestion, but not inside the engines that produce the numbers.

What every engine returns

Each answer carries the cost per unit, the source of every rate used and the date it was captured, the assumptions made, and which engine to call next. That last field is how chains are built: an engine names its own successors, so the sequences in the next section are assembled rather than programmed.

1. An AI workload is not a token bill

Q1 What is AICost | Q2 Why does an enterprise need it

Token cost is the largest and most visible driver of AI spend, and it deserves the attention it gets. Getting the model choice, the caching and the routing right is where the biggest single savings come from, and several sections of this document are about exactly that. The position taken here is narrower: token cost alone does not tell you whether a project will succeed at scale. A retrieval assistant over internal documents has eight cost layers, and the decisions that determine

whether it reaches full production are spread across all of them.

Layer of a single RAG workflow	What actually drives the cost	Who prices it today
Corpus onboarding	One-time embedding and indexing of the document set	Nobody. It lands in a cloud bill as undifferentiated compute
Freshness	Re-embedding as documents change, forever	Nobody
Retrieval shape	Chunk size and retrieved context dominate tokens per call	Nobody. It looks like "more tokens"
Inference	The visible part. Cache hit rate is the dominant variable	Every FinOps tool
Retries	Failed and retried calls, billed in full	Nobody
Guardrails	Prompt-injection and PII screening on every call	Nobody
Evaluation	A golden set scored every release to catch silent regressions	Nobody
Human review	Reviewer time on escalated or sampled answers. Often the largest line	Nobody

The position this document argues

Token cost is critical and it is well served by existing tools. What is not served is the rest of the picture: a realistic total cost of ownership, an ROI a business owner will stand behind, and governance that holds when the workload goes from a pilot to serving large customers. **Those three are what decide whether an AI project reaches full production at all.** Most enterprises are good at the token question and are stopped by the other three.

Why this changes as a programme matures

Because the question changes as the programme matures, and three of the four stages need an answer that retrospective tooling cannot give.

The four questions, in the order an enterprise actually meets them



A retrospective FinOps tool answers the second. The first, third and fourth need engines that model, gate and evidence.

9. What is covered: the whole estate, not the chat bill

Every column below is a workload type with its own cost shape and its own engines

An enterprise AI estate is not one thing being billed one way. Consumption can be seat-based, token-based, request-based, GPU-based, media-based, credit-based or capacity-based, often simultaneously and often inside the same business unit. The engine set is organised around that reality.

Domain	What sits inside it	Representative engines
Agentic and coding	Agent loops, multi-turn workflows, coding assistants, tool calls, retries, context growth	Agent Loop Cost, Agentic Workflow Cost, Agentic TCO and ROI, Agentic Variance Reserve, Retry Cost, Context Compaction
Retrieval and knowledge	Corpus onboarding, OCR and parsing, embeddings, vector storage, hybrid retrieval, refresh	RAG Pipeline, Corpus Onboarding, Data Freshness, Chunking Optimizer, Vector DB Cost, OCR Parsing Cost, Multimodal RAG Stack
Voice and media	Speech to text, text to speech, session duration, interrupt waste, image and video generation	Voice Agent Stack, Image Generation Cost, Video Generation Cost
Training and MLOps	Training runs, fine-tuning, GPU hours, spot risk, serving, ongoing model maintenance	Training Run Cost, Fine-Tuning Cost, GPU Spot Savings, Self-Host Breakeven, Inference Serving Cost, Model Maintenance Cost
Evaluation and quality	Golden sets, LLM-as-judge, human review, hallucination cost, benchmark runs	Eval Benchmark Cost, Eval and Anti-Hallucination Cost, HITL Review Cost, Eval Gate
Platform and operations	Guardrails, observability and tracing, integration engineering, developer tooling	Guardrail and Security Cost, LLM Observability Cost, Integration Cost, Dev Stack Calculator
Seats and subscriptions	Per-seat licences, plan tiers, allowances, overage, free-tier limits	AI Seat ROI, Subscription Picker, Annual against Monthly, Free Tier Checker, Plan Overage Calculator
Model and route selection	Model families, open weights, regional rate cards, routers and gateways	Cheapest Model, Model Family Comparison, Open-Source Model Finder, Chinese against Western Models, Multi-Model Router, Router and Gateway Comparison
Cloud and placement	Bedrock, Vertex, Foundry, direct APIs, regions, reserved and provisioned capacity	Cloud Workload TCO, Cross-Cloud Placement, Provisioned Capacity Optimizer, Region Cost Map, Gateway Stage Advisor
Governance and assurance	Eligibility, residency, controls, evidence, exposure, the cost of compliance	Jurisdiction Risk Scorer, Compliance Cost, Governance Cost Pack, Client AI Exposure Audit, Cost Firewall
Business case and diligence	ROI, payback, TCO, build against buy, investor-grade cost claims	ROI Quick Check, TCO and ROI Builder, Buy against Build, Support Deflection ROI, AI Cost Diligence, GPU Cost per Inference
Physical and emerging AI	Humanoid labour, robot fleets, teleoperation corpora, per-episode economics	Humanoid Labor ROI, Robot Training-Data Cost, AI Robot Fleet Cost, Robot Buy against RaaS

Why the breadth is the point rather than a feature list

A large enterprise will have workloads in most of those rows at once, at different stages of maturity, owned by different people. Pricing them with one model produces a number that is wrong in several directions simultaneously. Pricing each with the engines built for its shape, against one shared rate catalogue and one ledger, produces a total that survives being taken apart in a board meeting.

10. Situations enterprises are actually in, and what to run

Cost. Each row is a real pattern reported across 2026, and the engines that address it.

The industry data for 2026 is consistent. The FinOps Foundation found **73 percent of enterprises reported AI costs exceeding original projections**, while the average enterprise AI budget rose from about \$1.2 million in 2024 to roughly \$7 million. Blended token prices fell over the same period, from roughly \$18.40 to \$6.07 per million. Unit price down, bills up, which means the problem was never mainly price.

The situation you recognise	What is usually driving it	What to run
"We are three times over our annual token budget and it is only April."	The deployment decision preceded the cost model. Nothing priced the workload before it shipped	Scale Projection, AI Budget Planner, Agentic Variance Reserve
"Our coding agent bill tripled and nobody changed anything."	Agentic sessions replaced chat calls. One 2026 audit across thirty engineering teams attributed 62 percent of the inference bill to repeated context alone	Agent Loop Cost, Context Window Cost, Prompt Cache ROI
"One developer costs twenty times another developer."	Usage profile, not headcount. Heavy agentic users and light chat users share a line item	Per-key attribution, AI Seat ROI, Subscription Picker
"We capped every developer at a fixed monthly amount."	A blunt cap stops the work as well as the overspend. It is the response when there is no downgrade path	Cost Firewall with a soft cap and downgrade before the hard cap
"We revoked the licences after enabling them."	Seat cost was visible, usage cost was not, and the two were never modelled together	AI Seat ROI, Agentic Workflow Cost, Subscription Picker
"A \$1 task somehow cost \$20."	Reasoning models bill for thinking tokens the user never sees	Reasoning Token Cost, Model Family Comparison
"The prompt just loops until it gives up."	No turn ceiling and no circuit breaker. The runaway tail can be a third of the monthly figure	Agent Loop Cost, Agentic Envelope, Cost Firewall
"Our bill is fine but the pilot numbers were nothing like this."	The pilot ran at one team size. Cost per user is not linear as the profile changes with scale	Scale Projection, then the three-team-size presets in the calculator
"We have dashboards but the spend keeps growing."	Reporting a leak is not fixing it. The dominant variable is design decisions made daily by engineers	Multi-Model Router, Stacked Savings, CI cost gate
"We cannot attribute a dollar to a prompt, a tool call or a retry."	In a 2026 survey, 85 percent planned LLM observability and 8 percent had implemented it	Ledger attribution by team, workload, key and agent
"We cannot prove the ROI, so the next project will not get budget."	Only 23 percent of AI projects launched in the past year were fully deployed and meeting their ROI objectives	Support Deflection ROI, Agentic TCO and ROI Builder, ROI Quick Check

11. Situations enterprises are actually in: governance

The same format. What it looks like, what is behind it, and what to run.

Governance failures in 2026 have a common root: AI use is distributed and the inventory is not. A Forrester study found **65 percent of compliance executives cite shadow AI as the leading reason for governance gaps**, and 60 percent of IT leaders said they lack visibility into which generative AI tools their workforce is using while spending on AI-native applications rose 108 percent year on year.

The situation you recognise	What is usually driving it	What to run
"We do not know which AI tools our teams are using."	Procurement cycles take weeks and a team can adopt a tool in an afternoon. Banning it moves usage rather than stopping it	Shadow AI exposure findings, with the spend attached
"We cannot classify AI systems we do not know exist."	Inventory built from a survey ages immediately. Inventory built from spend does not	AI Inventory Board, built from metered spend
"Legal asked whether we can actually deploy this, and nobody could answer."	Eligibility was never derived. It lives informally, or in a document older than the model list	The eleven questions, six gates, and the elimination trace
"Different teams need different models and we cannot police that."	A single corporate policy is wrong for at least one team, so it gets ignored by all of them	Policy engine per team, compiled into one gateway config
"Our policy is a document. Enforcement is a reminder."	No enforcement point exists between the policy and the API call	policy_patch into the gateway, or workspace caps on direct vendor keys
"We can show the policy but not that it was followed."	Audit trails assembled after the fact rather than generated as a by-product of operations	Hash-chained ledger, oversight log, recertification record
"A model was swapped and nobody re-ran the evaluation."	Evaluation currency is not tracked, so a production model drifts away from the evidence behind it	Eval Gate with staleness reporting, which blocks an unevaluated swap
"The board is asking where AI sits in our critical processes."	Roughly 39 percent of Fortune 100 boards have explicit AI oversight mechanisms, and the rest are being asked the same questions anyway	Inventory with risk tier and owner, plus the priced control register
"Compliance wants a residency guarantee and engineering says it is expensive."	Nobody has priced the control, so the debate is about principle rather than about money	Jurisdiction Risk Scorer, Compliance Cost, Governance Cost Pack
"Our governance costs more than we budgeted and we cannot see why."	Human review, evaluation, guardrails and oversight are real spend that sits outside the token bill	HITL Review Cost, Eval Benchmark Cost, Guardrail and Security Cost

The pattern across both tables

In nearly every row the cost problem and the governance problem are the same problem seen from two directions. An unregistered workload is a governance gap and an unattributed line on the bill. A model swapped without evaluation is a compliance exposure and an unverified saving. Which is why the engines, the policy and the ledger sit on one workload object rather than in three separate systems.

12. Two ways to use any of it

The same engines, reached either by a person or by a machine

	As a calculator or playground	As an MCP chain
Who uses it	A finance manager, an architect, a team lead	An agent, an IDE assistant, a CI job, your own application
When	Deciding something: a budget, a ceiling, an architecture, a model choice	Continuously, or at a point in a pipeline
What it looks like	A page with three or four sliders that move the result, and a breakdown behind the number	A tool call that returns a typed answer with cost per unit, provenance and the next engine to call
Typical use	Quick mode at idea stage, advanced mode once the parameters are measured	Cost gate in CI, a nightly reconciliation, an agent checking its own budget before a long run
Output	A figure you can defend, and the inputs it rests on	A figure a downstream engine can use arithmetically, or a policy patch

The same question, asked both ways

EXAMPLE A A team lead, before committing to an architecture

Chain: Opens the Agent Loop Cost playground, moves turns per task and tokens per turn, and sees the runaway tail separate out from the base cost

What comes back: A ceiling they can justify, and the reason a ceiling is needed at all.

Takes about five minutes and requires no integration.

EXAMPLE B The same workload, six weeks later, running in production

Chain: CI calls the engine on every merge → compares against the agreed envelope → fails the build if the change would breach it

What comes back: The same arithmetic, applied automatically, with the result in the ledger.

No second model of the workload. The number the team lead agreed to is the number being enforced.

Why both surfaces matter for adoption

A finance manager will not call an API, and an agent cannot use a slider. Most tools in this space pick one audience. Running both against one engine set means the figure a person agreed to in a meeting and the figure enforced in production are the same figure, produced by the same code, with the same provenance attached.

13. Combining engines: the chain is assembled, not programmed

Q3 How is it different, part two

Because each engine names what to call next, a chain is put together rather than built. The same blocks appear in different orders for different workflows, and a workflow nobody has costed before gets a chain in a working session rather than a place in a roadmap.

■ registry engine ■ workflow tool ■ platform capability ■ quality gate ■ your system ■ outcome

EXAMPLE 4 Two engineering teams want two different models. Can we allow that and still get one bill?

Chain: [model-eligibility](#) → [policy_patch](#) → [ledger](#)

What comes back: Platform engineering is allowed Claude Fable for long refactors; the feature teams are allowed Sonnet and nothing above it. Both rules land in one gateway config, and both teams appear as separate lines in the same monthly total.

Three blocks. This is usually the first question an enterprise asks.

EXAMPLE 5 One developer went from \$29 a month to \$750. Which developer, and what changed?

Chain: [per-key attribution](#) → [Agent Loop Cost](#) → [Context Window Cost](#) → [Reasoning Token Cost](#)

What comes back: The ledger names the developer and the agent. The chain shows whether the cause was agentic sessions replacing chat calls, context growth across long sessions, or reasoning tokens on a model that charges for them.

Published 2026 benchmarks put coding assistants around \$13 per developer per active day, with agentic tasks consuming roughly a thousand times the tokens of code chat. Tenfold individual variance is normal, which is why attribution has to be per key.

EXAMPLE 6 Should these developers be on a subscription or on metered API billing?

Chain: [AI Seat ROI](#) → [Subscription Picker](#) → [per-developer ledger](#)

What comes back: A per-developer split. Heavy agentic users are cheaper on a capped subscription; light users are cheaper on metered API. The answer is a list of names, not one policy for everyone.

Industry reporting puts a fully used premium subscription near \$200 a month against \$600 to \$1,500 on API billing for the same usage.

EXAMPLE 7 Marketing wants a cheap model. Legal wants one that survives an audit. Same company.

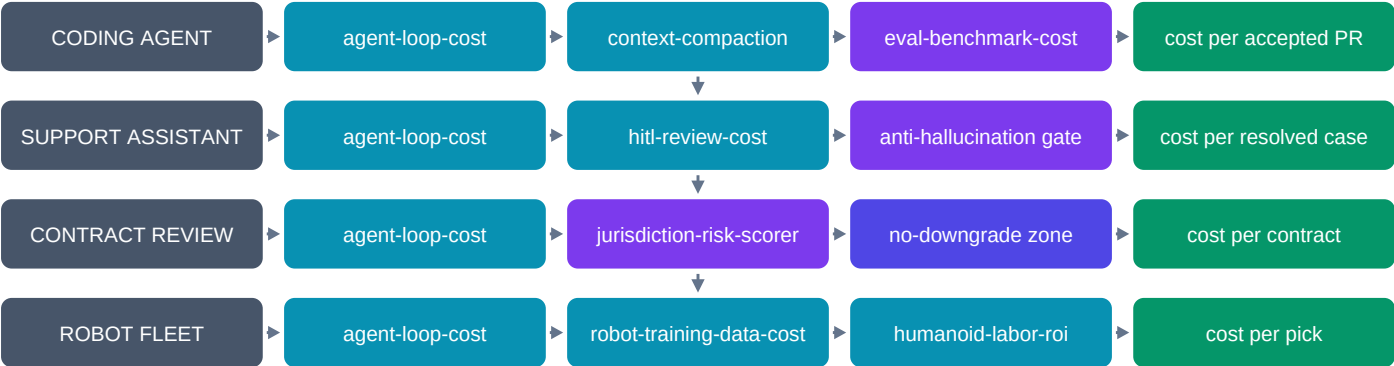
Chain: [data-class question](#) → [six eligibility gates](#) → [two templates](#) → [one policy file](#)

What comes back: Marketing gets the widest eligible set optimised purely for cost. Legal gets a narrow corridor of certified surfaces with retention contractually off. Both derive from the same eleven questions, so both are defensible to the same auditor.

This is the policy engine, covered in full in section 19.

One engine, four different chains

The clearest demonstration of composition is a single engine appearing in unrelated workflows. Agent Loop Cost prices turns, context growth and runaway risk. That same arithmetic is needed by four workloads that otherwise have nothing in common, and in each one it sits next to different neighbours and feeds a different decision.



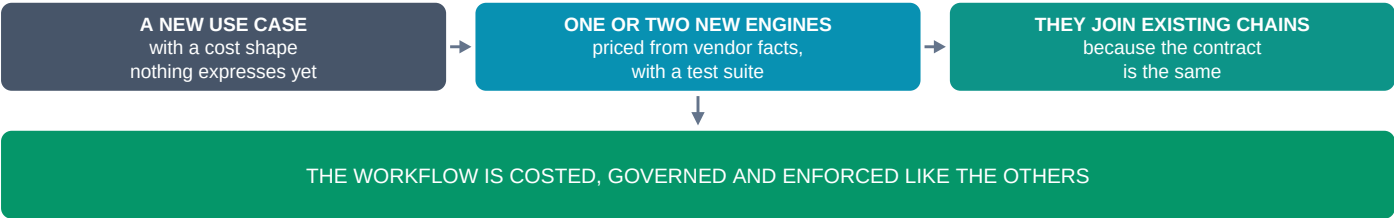
The same engine, four workloads, four different units of business value. Nothing about the engine changes; what changes is what it is chained to and what gate sits after it.

Why this is the useful property rather than the engine count

An enterprise does not need a hundred and twenty-three engines for one workload. It needs the right six or seven, arranged in the right order, with the correct gate in the middle. The engine set is a vocabulary; the chain is the sentence. That is why a workflow nobody has costed before can be handled in a working session rather than a release.

Adding a capability that did not exist last week

This is not an annual problem. New AI capabilities land weekly, and each one can bring a cost shape that nothing in an existing taxonomy expresses. Voice agents brought interrupt waste. Agentic coding brought unbounded turn counts. Physical AI brought per-episode economics with no token in them at all. A platform with fixed dimensions has to wait for a release before it can model any of these, and new engines are added here continuously for exactly that reason.



Voice agents, agentic coding and physical AI were each added this way, and the set keeps growing. None of them required the chains around them to change.

Capability that arrived recently	The cost shape it brought	Engines added
Agentic coding assistants	Unbounded turn counts and context that grows across a session	Agent Loop Cost, Agentic Variance Reserve, Context Compaction
Voice agents	Speech in, speech out, and waste when a caller interrupts mid-response	Voice Agent Stack, with interrupt-waste modelling
Reasoning models	A separately billed thinking step that does not appear in the response	Reasoning Token Cost
Physical AI and robotics	Per-episode rather than per-token economics, plus a demonstration corpus	Humanoid Labor ROI, Robot Training-Data Cost, Robot Fleet Cost, Buy against RaaS

What to ask any vendor in this space

Not how many things it measures today, but what happens when your teams adopt something next quarter that has not been priced yet. A fixed schema answers with a roadmap date. Adding an engine takes a wave of work and leaves the existing chains unchanged, because the contract between engines stays the same.

What this means for an enterprise with many workflows

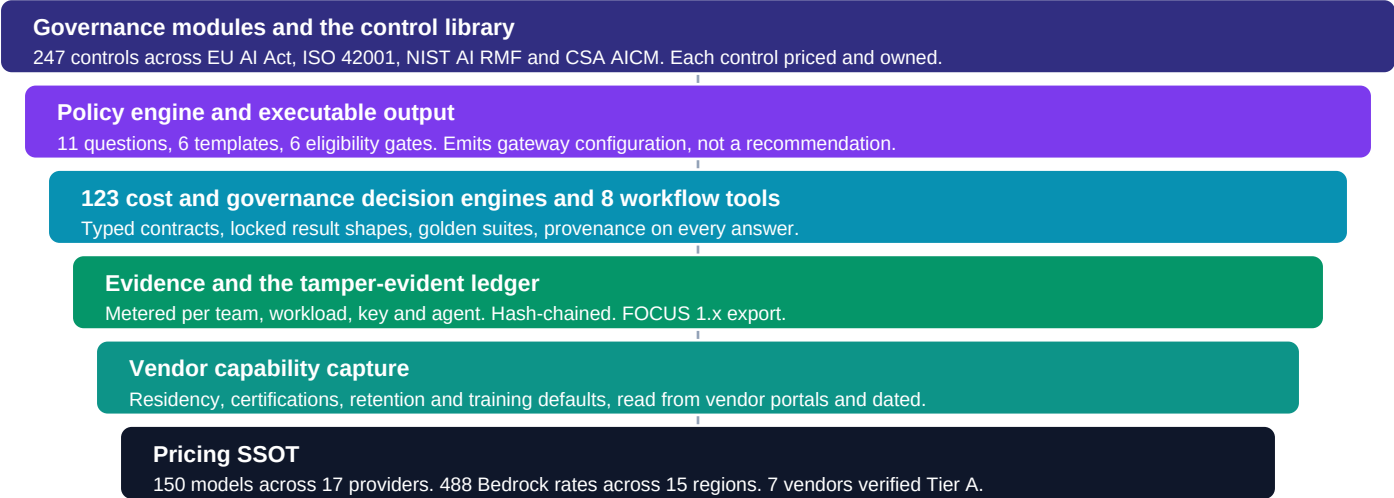
A large organisation is rarely running one kind of AI. It has coding assistants in engineering, a retrieval assistant in support, something summarising documents in legal, an experiment in marketing and at least one team trying agents. These have almost nothing in common as cost shapes. Pricing them with one model produces a number that is wrong in six different directions at once. Pricing each with its own chain, against one shared rate catalogue and one ledger, produces a total that survives being taken apart.

14. The single source of truth, and what keeps it true

Every engine answer is arithmetic on these rates. A wrong rate is invisible in the output.

If a base rate is wrong, every figure derived from it is confidently wrong and no reviewer can detect it by reading the report. The pricing catalogue is therefore treated as a product with its own test suite rather than as reference data, and governance is built upward from it.

GOVERNANCE (top) rests on EVIDENCE which rests on the PRICING SSOT (base)



The evidence standard

A price enters the catalogue only when it has been read from the vendor own pricing page. Not from an aggregator, not from a marketplace, not from a search result. Aggregators copy one another, so two of them agreeing is not corroboration. In one verification sweep, four prices sourced from converging secondary sources were each found to contradict the vendor page.

Twenty-three automated checks, every morning

The catalogue is audited daily by two layers before anyone uses it. These are not linting rules; each exists because a specific class of defect reached the catalogue at least once.

Check class	What it catches	Why it exists
Duplicate identity	The same model present twice under different spellings, so a lookup returns an arbitrary one	A duplicated entry once caused a model to be quoted at five times its real price
Reference disagreement	A catalogue price that differs from an independent reference beyond tolerance	Catches a mistyped rate and a vendor repricing on the same check. Disagreements are waived only with dated evidence
Retirement and resurrection	A retired model reappearing because a staging file still lists it behind a deprecation banner	Retired rows became eligible again once. The veto now runs at the writer, not only at the gate
Scheduled change tracking	An announced price rise landing unapplied, or a withdrawn one being applied anyway	A vendor announced an increase, then cancelled it. Applying it from memory would have overcharged by half
Catalogue size floor	The catalogue shrinking without an approved reason	An unattended job once removed dozens of entries and no alarm fired, because nothing was watching the count

Check class	What it catches	Why it exists
Provenance integrity	A row losing the field that records whose price it is	Stripping that field did not fail a check, it silently switched one off
Regional and routing coherence	Batch not at the expected ratio, routing modes mislabelled, end-of-life paths still priced	One parser fault mislabelled geographic rows as batch rows, hiding a real residency premium

What this discipline is worth to a CFO

Every savings claim, every showback line and every compliance-premium figure in a quarterly review is arithmetic on these rates. The daily checks, the as-of dating and the marked inferences exist so that the number being signed has a traceable origin and a date. The standing position is deliberately modest: the catalogue does not claim to be current, it claims to be dated, checked and honest about the gap, and it says so on every page that carries a price.

15. Costing work before it runs

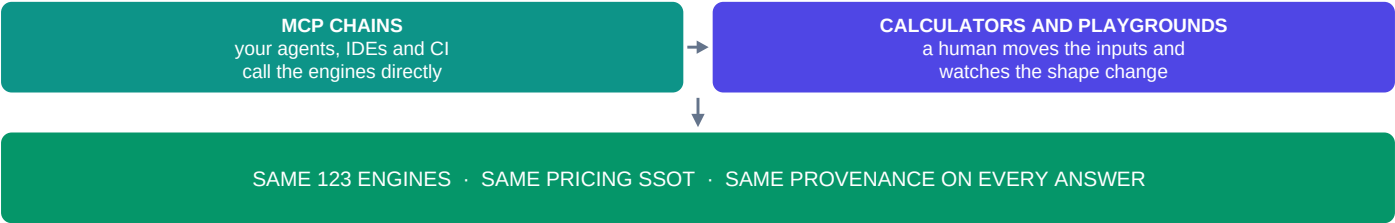
What it does, when you use it, and what you get

Most FinOps tools are historical databases. They read bills, traces and usage exports, then allocate and project from what already happened. Their own documentation describes forecasting from historical usage patterns. That works well for spend that exists, and it cannot answer a question about spend that does not exist yet.

AICost models a workload from its description: model, token shape, volume, cache rate, batch share and review rate, priced against the current rate catalogue. That produces a figure before anything is built, which is when most of the expensive decisions are made.

The decision	When it is made	Retrospective FinOps tool	AICost
Build this agent platform or not	Before any spend exists	Silent	Cost per run, with the variance reserve
Frontier or mid-tier for this team	Before deployment	Silent	The delta, derived from live rates
Fine-tune or keep prompting	Before the training run	Silent	Break-even across four cost terms
Self-host or call the API	Before commitment	Silent	Break-even with real ops costs included
Bedrock, Vertex, Foundry or direct	Before contract	Silent	Four-lane TCO at your shape
Add a human review step	Before design is fixed	Silent	Cost per reviewed item, and whether it pays

Two surfaces, one engine set



An engineer calls the engine from Cursor or a CI job. A finance manager uses the same engine through a calculator. The arithmetic is identical.

Two things that keep a forward estimate grounded

Connecting a cloud account means the model starts from your actual last six months rather than from entered figures. And each forecast is compared against actual spend afterwards, with the accuracy reported as a percentage on the status endpoint.

16. Integrate into the workflows you already run

Q4 What does it connect to | Q5 What data does it require

Q4 What does it connect to?

The same answer in every scenario: **your model calls never route through us**. The MCP layer is out-of-band advisory and audit. What you get in each lane is the same pair of outcomes, daily cloud and AI cost visibility, and daily optimisation recommendations from engines running in your pipeline.

AWS Bedrock full connector One-click CloudFormation creates a read-only cross-account role in your account with a per-customer External ID. Model invocation logging puts token-level JSONL in near real-time, so you do not wait for CUR. Daily: by model, by workload, by agent, tokens plus cache, modelled against billed drift.	Azure AI Foundry cost export Schedule the Cost Management daily export and forward it to the inbox. Azure Monitor token metrics per deployment fill the intraday view. PTU showback allocated by utilisation share. Daily: by deployment, workload and agent, with the pay-as-you-go against PTU split.	GCP Vertex AI billing export Gemini and Model Garden priced before you ship, plus daily actuals from the BigQuery billing export. Real-time through the modelled lane when your app emits usage metadata. Daily: by model, by workload, by agent, with labels honoured.
Direct vendor APIs richest lever surface Anthropic, OpenAI, Google, Mistral and others. Caching, batch and routing engines apply at list prices, which is where the largest levers live. Two paths for daily cost: the usage block your app already receives on every response, or pull the vendor usage API into the inbox.	LLM gateways fully automated LiteLLM, Portkey, Helicone and others. Policy flows out as configuration; actuals flow back through the connector continuously. No exports, no cron, no waiting, and still zero inference through us.	Your pipelines and assistants four doors Claude Code and Desktop by MCP, Codex CLI and IDE by config, your own application by REST, and CI or nightly jobs by run-batch. OAuth for humans in assistants, bearer keys for apps and pipelines. Nothing is degraded by using a key.

Three lanes, and why an enterprise needs all three

Cost data arrives at different speeds and with different authority. Treating them as one number is why forecasts and invoices disagree without explanation. AICost keeps them separate and reconciles them, so the question is always answerable: what should this have cost, what does telemetry say it is costing now, and what did the provider actually bill.

Lane	Where it comes from	What it is for
Modelled	The architecture and the rate catalogue, before anything runs	A baseline that exists before the first invoice, which is the only lane available for a new agent or a pilot with no history
Log-valued	Invocation telemetry: tokens, cache behaviour, caller identity	Intraday visibility and abnormal-behaviour detection, without needing prompt content
Billed	The provider billing export, such as AWS CUR	Financial truth, including discounts and billing mechanics that a modelled rate cannot reproduce exactly

The cloud provider stays the source of financial truth

AICost does not attempt to replace the billing export or re-derive what a provider charged. The modelled lane says what the workload should cost by design, the log lane says what it is doing today, and the billed lane remains authoritative. The value is in the comparison between them, which is where a drifting workload or an unexpected discount shows up.

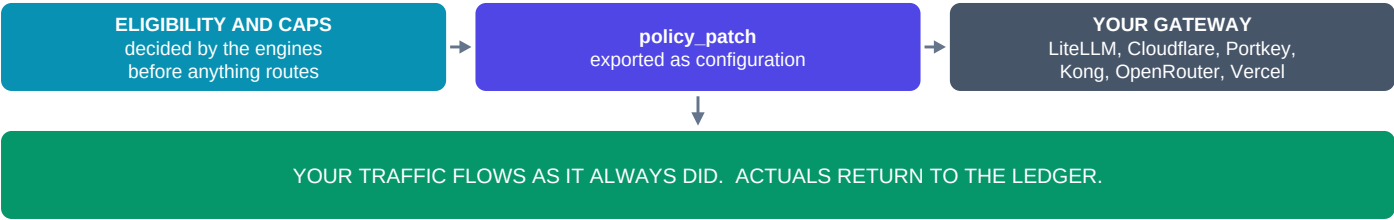
Rolling up across an enterprise



Every level also shows an unattributed residual. Hiding the gap would make the roll-up look complete when it is not; showing it is what makes the number usable.

Deploy into the gateway you already run

We are the **vendor-neutral decision layer that runs before the routing layer**. The engines decide which models are permitted, what they cost and what the caps are, and that decision is exported as configuration into whichever gateway your team already operates. Anything a target cannot express natively is preserved under **x_aicost_policy**, so nothing is silently dropped.



Helicone traces answer what happened per request; the envelope answers what it should have cost and why. The two reconcile in the ledger.

Q5 What data does it require?

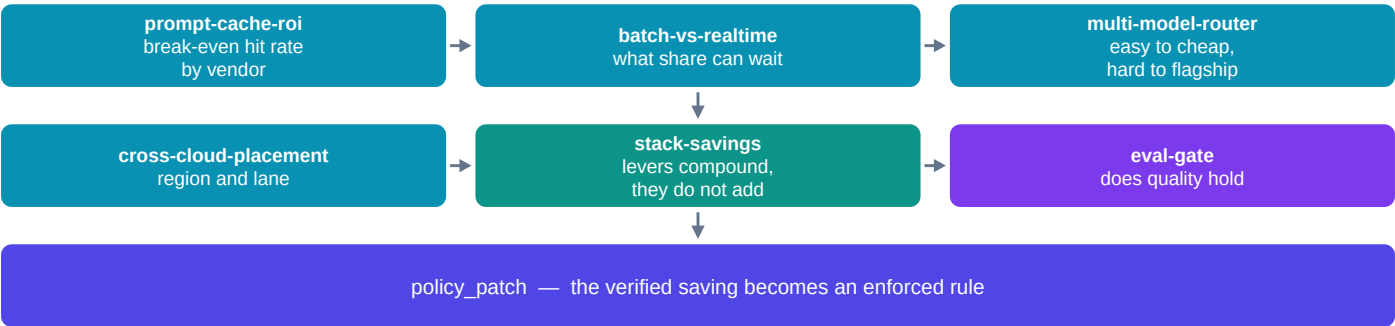
Required	Why it is needed	Not required
Read-only cost and usage	The retrospective half: what was actually spent	Prompts, completions or any customer content
A team-scoped key convention	The team dimension must exist at the moment spend is recorded. It cannot be added afterwards	Any change to how your teams build or deploy
A workload description	Model, token shape, volume, cache and batch share, review rate	Production access of any kind
An owner per workload	Governance needs an accountable person, not a system	A completed inventory. The inventory is an output, not an input

17. How cost comes down, and how it stays down

Q6 How does it reduce costs

Four structural levers, each priced against your real shape before it is claimed, each checked by a quality gate, then compiled into a rule so the saving does not decay when attention moves elsewhere.

The optimise chain, and why the fifth and sixth blocks matter most



Routing 45 percent plus caching 30 percent is not 75 percent. Each lever only works on what the last one left behind.

The gateway question, decided per team rather than per company

A gateway earns its fee when traffic is unpredictable: model switching during the day, failover, trialling a new model without a deployment. It earns nothing on a workload that has called the same model with the same shape for six months. Most enterprises pay the same rate for both.

	Static workload	Dynamic workload
Shape	One model, stable volume, months at a time	Shifts between models, bursty, experimental
Right path	Direct vendor API on a team-scoped key	Gateway or router, where the fee is earned
Why	A percentage fee on a fixed pattern buys nothing	Failover, trialling and per-key budgets are worth real money

What the fee actually is, stated so it survives scrutiny

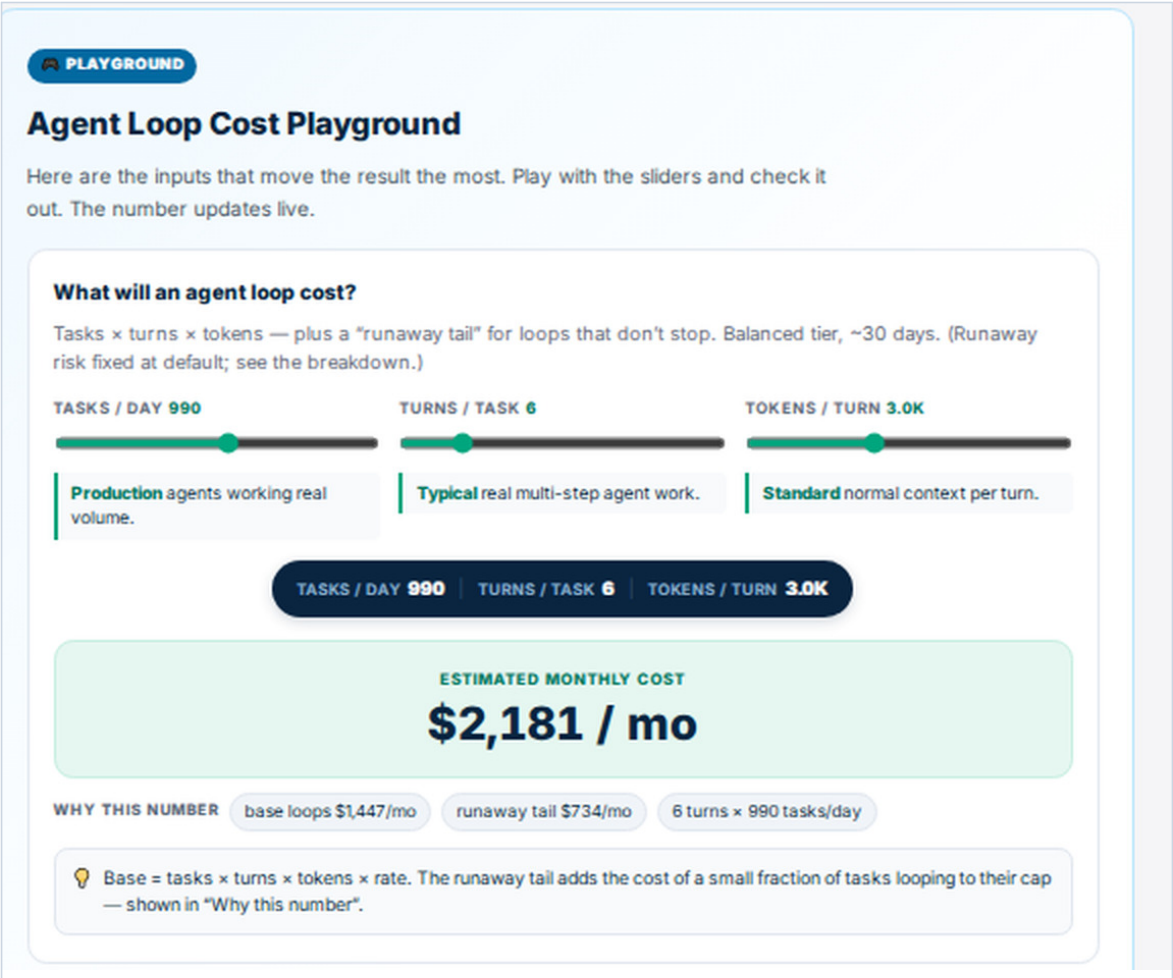
Router and gateway fees sit around 5 to 5.5 percent of routed spend, and one major gateway charges 5 percent unified billing. Bring-your-own-key tiers waive much of this below a monthly threshold, so at modest volume there is little to recover. At enterprise volume the fee is real. Where 60 to 70 percent of traffic is static and moves to direct keys, the recoverable portion is on the order of **3 to 4 percent of total inference spend**, before the engineering cost of managing separate keys and losing automatic failover. On a 200,000 dollar per month programme that is roughly 80,000 to 95,000 dollars a year. Material, and smaller than caching, batch and routing, which are 10 to 40 percent levers.

Being neither a gateway nor a model vendor, AICost has no incentive in either direction. The policy engine is free to conclude that one team should call Anthropic directly on a scoped key while another team on the same floor should route through the gateway.

18. Calculators and playgrounds: where a policy gets designed

Some policies are a tag and a model list. Others have to be explored first.

Not every policy can be written from first principles. A simple one can: marketing carries a marketing tag and may use this list of models. A more consequential one, such as the ceiling on an agent loop, depends on parameters nobody knows until they have moved them and watched the number respond. Every engine therefore has a human surface as well as a machine one.



The Agent Loop Cost playground. Three sliders, a live monthly figure, and a breakdown separating the base cost from the runaway tail.

What the playground is actually for

That screen is where somebody decides what the ceiling should be, because it shows which input moves the number and by how much. In this configuration the base loop cost is \$1,447 a month and the runaway tail adds a further \$734, which is the cost of a small fraction of tasks looping until they hit their cap. About a third of the monthly cost comes from the tail rather than from the intended work, which is the figure a ceiling is set against.

What the surface gives a person	Why a machine call alone does not
Only the inputs that move the result	Each engine exposes the parameters that matter, so nobody tunes something that will not change the answer

What the surface gives a person	Why a machine call alone does not
A live number as an input moves	Sensitivity becomes visible. You can see which parameter justifies an architecture change and which does not
A breakdown behind the figure	Base against tail, cheapest against priciest. The decision is usually in the decomposition, not the total
Real scenarios as presets	Solo developer, five-person team, fifty-person team. A starting point near your shape, then adjusted
The same engine underneath	What was explored in a browser is the same arithmetic an agent calls, and the same figure that lands in the policy

19. The policy engine

Q7 How does it handle governance. This is the section that ties the rest together.

Everything in the previous sections produces a decision. The policy engine is what turns those decisions into a rule that holds when nobody is watching. It answers one question for each workload, **which models may this team use, at what ceiling, on which route**, and it answers it as configuration rather than as advice.

Two kinds of policy, and two ways to write one

Most policies in an enterprise are simple, and they are written directly. A minority are consequential enough that the right answer has to be found before it can be stated, and those start in the calculators from the previous section.

	Simple: written directly	Consequential: explored first
Example	Marketing carries team=marketing, data_class=public, and may use the economy tier. Dev team XYZ carries team=xyz and may use Sonnet but not the frontier model	What ceiling should an agent loop have? What cache hit rate does the support assistant need to justify its budget? Should these fifty developers be on subscriptions or metered keys?
How it is decided	The data class and the attestation determine it. The eleven questions reach an answer in three steps	A person opens the relevant calculator, moves the inputs that matter, and reads the sensitivity before choosing a number
What it produces	A tag, a model allowlist and a budget, compiled into the gateway config	The same, plus a reasoned figure for the ceiling and a record of the parameters it was based on
Who does it	Set once by whoever owns the function, reviewed at the quarterly recertification	The team that owns the workload, because they are the ones who know what a turn count or a cache rate realistically looks like in their system

Why a person is involved at this step

An engine can compute what a ceiling would cost at any given value. It cannot know that this particular research agent typically takes four to eight turns and occasionally forty-seven. That knowledge sits with the team running it, and the calculator exists to let them apply it against real rates rather than guess. Once the number is chosen, it stops being a human decision: it is compiled into the policy and enforced on every call thereafter.

Eleven questions, and why each one exists

The tree is ordered so a low-sensitivity workload reaches an answer in three questions and a regulated one earns the long path. Every question is present because a captured vendor fact makes it decisive, and each carries the provenance of that fact.

#	Question	The fact that makes it decisive
1	What data will this workload touch	Eligibility is scored against a data class, never in the abstract. Everything branches here
2	Does the organisation hold an attestation an undocumented flow would break	The exposure is the attestation. A SOC 2 auditor asks where AI data is processed, and an incomplete answer produces a qualified opinion
3	What plan or surface is in use today	The most common trap. Consumer and business tiers of the same vendor differ on coverage
4	Which cloud agreements are already held	A Bedrock, Vertex or Foundry agreement changes which routes are contractually available
5	Does policy forbid shared inference infrastructure	Single tenancy eliminates the hosted route at every closed vendor captured

#	Question	The fact that makes it decisive
6	Where must processing stay	Region availability is not uniform. Some models are unavailable under a strict residency rule at any price
7	Must the vendor be contractually barred from training	Vendor defaults differ by plan and by surface, and they change
8	What kind of workload is it	Batch, interactive and agentic have different cost shapes and different ceilings
9	Does it answer while someone waits	Rules out the batch tier and some regions
10	What accelerator memory could be provided	Opens or closes the open-weight branch
11	What routes model traffic today	Determines whether the output is advisory or directly executable

A trap the engine fires early, as a concrete illustration

If a team answers that they are on a business Team tier and the workload touches health or financial data, the engine stops before it prices anything and says so: at both closed vendors captured, the Team tier is excluded from the business associate agreement. The plan has to change before any model choice matters. That finding arrives in question three rather than in an audit eighteen months later, and it is the kind of thing a spreadsheet of model prices cannot contain.

Six gates, applied in order



Price is last on purpose. A model that fails residency is not expensive, it is ineligible, so pricing it would be misleading. Eligibility is scored per ROUTE: the same model on Bedrock

Six templates, selected by the answers

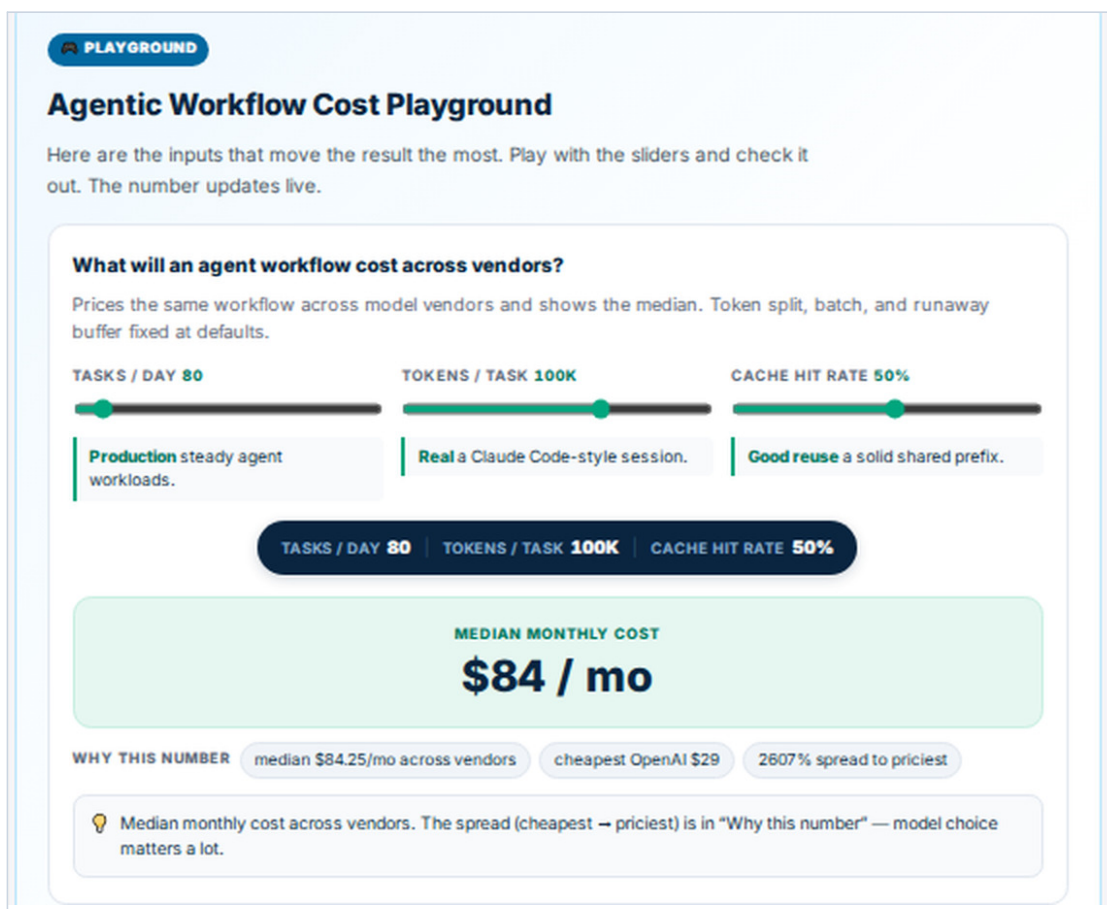
Template	Selected when	Posture
Open, cost optimised	Data class is public	Any vendor, any route. Optimise purely for cost and latency
Internal business	Data class is internal	Named vendors only, deny by default, training off
Regulated personal data	Data class is regulated	Agreement in place, residency constrained, training contractually off
Health or financial, agreement required	Health data plus a matching attestation	Narrow corridor. Only covered surfaces reachable, with a kill switch among the required controls
Export-controlled or defence	Export data class or attestation	Deny by default. Nothing leaves the permitted boundary
Self-host or open weight	Sufficient accelerator capacity available	Open-weight branch, priced against the hosted alternative

Two modes, because certainty arrives gradually

Every calculator has a quick mode and an advanced mode. Quick mode asks three or four fields and a named scenario, with sourced typical values, for someone who knows the use case but not yet the engineering parameters. Advanced mode adds token and context detail, tool overhead, caching, model choice, retries and volume, for an architect working from measured figures. The same engine serves both, so a workload can be costed at idea stage and re-costed with real numbers later without changing tools or losing comparability.

Designing the harder policies before writing them

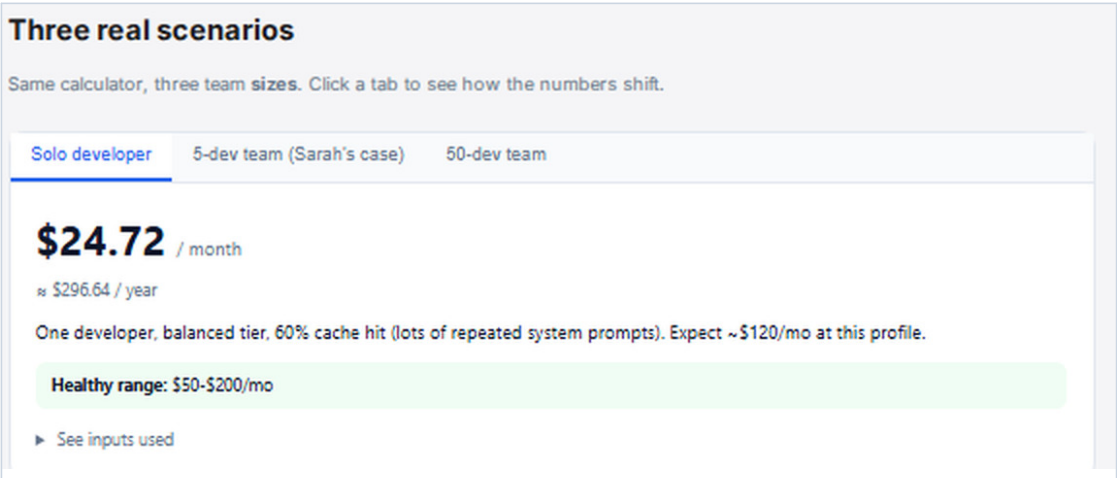
For a workload where the right ceiling is not obvious, the sequence is explore, then decide, then compile. The playground answers what the cost is sensitive to; the policy engine answers what is permitted; the export turns both into configuration.



The Agentic Workflow Cost playground, priced across model vendors.

What the vendor comparison shows

The same workflow, at the same volume and token shape, costs about \$29 a month at the cheapest vendor and more than twenty-six times that at the most expensive. Model choice is usually the largest single variable in a workload. This screen shows the range so the choice can be made with the cost visible, and the result becomes the allowlist for that team.



The same calculator at three team sizes. Cost per developer is not linear, so the usage profile matters as much as the headcount.

What gets exported, and where it goes

The output is never a bare verdict. It is the template chosen, the models eligible per route, the compiled policy, the elimination trace with a plain-English reason for every model ruled out, and the capture date of every vendor fact relied upon. The compiled part is then exported to whatever your teams actually use, and that varies by team inside one company.

Direct vendor APIs Anthropic, OpenAI, Google For teams that call the vendor directly with no gateway in the path. The policy becomes workspace and project-level spend limits, per-key allowlists and admin-console budget caps. The engines tell you which keys, which models and which ceilings; the vendor console enforces them.	LiteLLM self-hosted gateway policy_patch compiles to model_list, fallbacks, budgets, cache and retry config. The ledger pulls spend back from its database, so forecasts against actual closes on its own.	Cloudflare AI Gateway edge gateway Compiles to gateway config: caching, rate limits, fallback order. Their edge guardrails and DLP run in front; our caps and routing policy ride in the same config.
Portkey / Kong gateway Policy export to gateway config with guardrails at every tier on their side. Our eligibility gates decide which models the config is permitted to contain.	OpenRouter / Vercel gateway and router We run before it: eligibility and budget policy decide the permitted model set, and the router moves the traffic across that set. Billing exports return to the ledger.	Coding assistants Claude Code, Copilot, Cursor, Codex Model and repository controls, seat tiers and per-developer budgets on the admin side; the engines decide who belongs on a subscription, who belongs on metered API, and what each team may select.

What gets measured after the policy is live

Unit economics	Quality	Control	Value
Cost per successful task, per accepted pull request, per resolved case	Evaluation pass rate and task success after routing changed	Breaker, downgrade and policy coverage across the estate	Throughput, and whether high-value usage was preserved

The four together are what stop cost control becoming usage suppression. A programme that cut spend by reducing output has not optimised anything, and only the second and fourth columns will show it.

Governance controls are priced

A residency requirement, a zero-retention requirement and a certified-surface requirement each add cost, and the engines calculate how much. The output is a monthly figure per control at current volumes, what would have to change to remove it, and the obligation that requires it.

Teams with no gateway at all

A large share of enterprise AI spend runs on direct vendor keys with no gateway, because that is what shipped first. The policy engine still applies. It produces the allowlist, the per-key budget and the ceiling, and those are set in the vendor console instead of a gateway config. Enforcement is coarser on that path: a vendor console can cap a workspace, but it cannot downgrade a model mid-session the way the cost wall can.

20. From policy to enforcement: the graduated ladder

What the policy does when a team crosses a line, in six graduated steps

A policy that only records a breach is a report. The engines compile the policy into an enforcement point that acts, and it acts gradually. The first four steps keep work flowing and only remove cost; the fifth asks a person; the sixth stops. The downgrade and hard-stop behaviours were proven on live traffic on 14 July 2026, with spend attributed per model in the ledger.

WARN	Cost velocity or context growth crosses a threshold	recorded and notified Nothing is restricted. The condition is on the record.
CONSTRAIN	The workload is above plan but the work is still legitimate	cache, reuse, compact context Cost is removed before capability is.
DOWNGRADE	Routine substeps can move to a cheaper eligible route	routed_to_economy Only to a route that passed eligibility. Work continues.
GATE	The cheaper route must still meet the accepted evaluation	quality gate Fails the eval, returns to the approved route or escalates.
APPROVE	Spend beyond the envelope needs a person to accept it	requires approval A human decision, recorded, rather than a silent overrun.
BREAK	The loop continues outside the hard envelope	pause or kill The smallest useful scope: run, agent, workload or team.

No-downgrade zones

Not every task should be made cheaper under pressure. A contract review, a production incident response or a regulated decision can be marked as a no-downgrade zone, which means the ladder skips the economy route entirely and goes from constrain to approve. A blunt cap treats a legitimate spike and a pathological loop identically; a graduated ladder with protected zones does not.

What the downgrade step does

When spend crosses the soft cap the wall does not block; it **changes the model** and work carries on at a fraction of the cost. On the proving run the hard cap took considerably longer to arrive than it otherwise would, because the cheaper model stretched the same budget much further. The ledger recorded both models, so the downgrade is visible in the data rather than asserted.

EXAMPLE A An agentic refactor runs overnight and nobody is watching.

Chain: agentic-envelope → soft cap → routed_to_economy → hard cap → ledger

What comes back: The loop is downgraded at the soft cap and stopped at the ceiling. By morning the ledger shows which model served which calls and at what point the downgrade happened.

The break step is for the other case: something is wrong at 2am and nobody yet knows what. It halts the smallest useful scope and will not restart itself because a counter reset.

The ledger is the proof

What it records	Why it matters
Model, team, workload, key, agent and dollars, per call	Showback and chargeback become a by-product of the control, not something reconstructed from an invoice
Enforcement events, refused and downgraded calls	What the ceiling prevented becomes a reportable line, not an anecdote
Hash-chained, signed entries	Tamper-evident. An auditor can test it rather than take it on trust
Forecast against actual, scored as a percentage	The forward model is graded, including when the grade is unflattering
FOCUS 1.x export	Lands in the FinOps tooling already in place, in the standard format

21. One enterprise, one rule set, eleven policies

What the whole governed estate looks like on a single page

Every AI workload, its data class, what it may use, what it may spend, what stops it, and where that is enforced. All derived from the same eleven questions, so any line can be challenged and the reasoning shown.

Org unit and workload	Data class	Eligible routes	Ceiling	Enforced at
Platform engineering agentic refactor	Source code	Frontier eligible, geo routing	Per-run envelope, turn limit	Gateway
Feature teams code assist	Source code	Mid-tier only, no frontier	Per-key daily budget	Gateway
Data and ML offline evaluation	Internal	Batch tier only, economy models	Monthly budget	Gateway
Solution engineers client document analysis	Client confidential	Residency-gated, zero retention	Per-workload cap, no downgrade	Gateway plus vendor console
Customer support RAG assistant	Internal business	Economy primary, mid-tier on escalation	Daily soft and hard cap	Gateway
Support quality evaluation pipeline	Internal business	Batch tier, judge model fixed	Monthly budget	Direct key
Marketing content generation	Public	Widest eligible set, cost optimised	Seat plan, overage alert	Vendor console
Legal contract review	Regulated	Certified surfaces only, agreement in place	No-downgrade zone, approval above cap	Gateway plus approval
Finance analysis assistant	Confidential	Mid-tier, retention contractually off	Monthly budget	Gateway
HR policy assistant	Personal data	Narrow allowlist, human oversight logged	Per-workload cap	Gateway plus oversight log
Innovation agent experiments	Internal	Any eligible route, time-boxed	Hard monthly ceiling, auto-expiry	Gateway

What a manager should take from that table

Observation	Why it matters
Eleven workloads, eleven different policies, one derivation	Nobody negotiated with anybody. Each policy follows from the data class and the attestation, which is why they can differ without being arbitrary
Three different enforcement points appear	Gateway, vendor console and approval workflow. The estate is governed as it actually is, not as it would be if every team used the same infrastructure
Two workloads carry a no-downgrade zone	Legal and client-facing work is protected from cost optimisation. Cost control that degrades a regulated output is not a saving
Every line has a ceiling	Including the innovation team, whose ceiling is time-boxed with automatic expiry rather than depending on somebody remembering to revoke it
The whole table is one file	It compiles to gateway configuration, versioned and diffed before it loads. Changing a data class regenerates the affected lines

The same estate, seen from the board

Boards in 2026 are asking a consistent set of questions, and answering them requires artefact-level evidence rather than assurance: an inventory, risk classifications, audit trails and human oversight records. Each question below is answered by something the estate produces as a by-product of operating, not by a document written for the meeting.

The board asks	What answers it	Where it comes from
Where is AI in our critical processes?	The inventory, with risk tier and owner on every workload	Built from metered spend, so an unregistered workload appears as an unattributed line
Who may use what, and who decided?	The policy estate and the elimination trace behind each line	Derived from the eleven questions, versioned, with the vendor facts dated
What stops a workload going wrong?	The ceiling, the graduated ladder and the no-downgrade zones	Compiled into gateway configuration and enforced on every call
What is our exposure to shadow AI?	Spend appearing from tools nobody approved	Exposure findings with the dollar figure attached
What does our governance cost?	The priced control register and the compliance premium	Calculated at current volumes rather than estimated once
Can we prove the approved state still holds?	The hash-chained journal, oversight log and recertification record	Accumulated continuously, testable by an auditor

What makes this different from a governance report

A report describes a state at a point in time. This is generated from the live model on demand, so the answer given to a board in October and the answer given to an auditor in January come from the same source and can be compared. Where they differ, the journal shows what changed and when.

22. What a pilot looks like, and how success is measured

Q8 What happens in a pilot | Q9 How will we measure success

Q8 What happens in a pilot?

Phase	What we do	What you do	Elapsed
0 Exchange	Issue organisation and tenant identifiers, MCP keys, and the connector or export mapping for your lanes	Tell us which lanes you run, give read-only access	Day 1
1 Baseline	Run the visibility chain across the workloads in scope. Produce the first attributed number	Name an owner per workload	Week 1
2 Attribution	Map teams to workloads, keys to agents, tags to business context	Adopt the key convention	Week 1 to 2
3 Policy	Run the policy engine per team. Produce the elimination traces and the first policy patch	Review the traces and challenge them	One session
4 Enforcement	Deploy the wall. Set soft caps, hard caps and the emergency stop per workload	Agree the ceilings and the downgrade ladder	Week 2 to 3
5 Optimise	Run the optimise chain. Each lever verified, and gated on quality, before it is claimed	Approve the swaps that pass the eval gate	Week 3 onward
6 Govern	Control register priced, journal running, review pack generated from the live model	Adopt the policy, sit the review	Quarterly

Q9 How will we measure success?

Five criteria, all objective, all verifiable by your own team without taking our word for any of them.

Criterion	The evidence	How you verify it independently
A per-team cost number that did not exist before	Ledger rows with team, workload and agent attached	Compare the roll-up against your vendor invoices
At least one structural saving identified	A cache, batch, routing or placement lever with the arithmetic shown	Re-run the lever yourself against the vendor rate card
A policy patch loaded by your gateway	The diff your security team reviewed, and the config in place	Your own change control
A working enforcement event	A downgrade or a refused call in the ledger, with its tier	Trigger it deliberately in a test workload
Forecast accuracy scored	Forecast against actual, reported as a percentage	The status endpoint, which reports the figure on every run

Why these five and not a savings percentage

Each of the five either happened or did not. A number exists or it does not. A call was refused or it was not. A forecast was inside its stated accuracy or it was not. All five can be checked by your team against your own systems.

23. What enterprise deployment looks like

Q10 How does this scale beyond the pilot

Dimension	Pilot	Enterprise deployment
Scope	One or two business lines, a handful of workloads	Every business line, every AI workload, one ledger
Tenancy	A single organisation identifier	Organisation and sub-tenant per business line, with roll-up
Connectors	One cloud lane plus direct keys	Every lane you run, each reconciled against the others
Chains	The visibility chain and one optimise chain	A chain per workflow family, composed for your estate
Policy	One or two team policies	A policy per team from one rule set, recompiled as vendors change
Enforcement	Caps on the pilot workloads	Ceilings per team, per workload and per key, with the emergency stop tested
Governance	Control register for the pilot scope	Full control set, quarterly recertification, incident clock, signed journal
Reporting	A first attributed number	Showback or chargeback per cost centre, FOCUS export into your FinOps tool
Operation	We run it alongside your team	Either we continue to run it, or your team does with us in support

What changes in your environment, and what does not

Changes	Does not change
Keys become team-scoped rather than shared per environment	Your inference path. Traffic still flows application to gateway to vendor
Your gateway loads a policy file you have reviewed and diffed	How your teams build, deploy or write code
A read-only role exists in your cloud account, revocable at any time	Any access to prompts, completions or customer data. There is none
Cost becomes a dimension in engineering conversations, not only finance ones	Your existing FinOps tooling, which receives the FOCUS export

This is not a replacement programme

FinOps platforms hold cloud financial actuals. GRC platforms hold controls and attestations. Gateways hold request-path routing and enforcement. Observability and evaluation platforms hold runtime traces and quality. Each is valuable and none of them owns the complete workload economic decision, which is the gap AICost fills.

System you already run	What it keeps doing	What AICost adds
FinOps or cloud cost platform	Cloud financial actuals, allocation, budgets	The workload model, the forward economics, and a FOCUS export back into it
GRC or risk platform	Controls, attestations, audit workflow	Route eligibility derived per workload, and the cost of each control
Gateway or router	Routing the request, failover, rate limits	The eligibility and budget decision that runs before routing, exported as config
Observability or eval platform	Traces, latency, quality scores	What the traced call should have cost, and whether quality held after a change

What adoption actually requires

No rip-and-replace. Keep your current runtime, billing exports, identity controls, risk platform and observability stack. Start with one or two workloads, reuse the tags and billing exports you already have, and add only the identity or evidence fields that are missing.

A clear division of responsibility

Stays with you	Sits with AICost
The inference path. Requests flow application to gateway to vendor as they do today	The decision that runs before routing: what is eligible, at what ceiling, on which route
Cloud accounts, identity, and who may grant access	A read-only role in those accounts, scoped to cost and metadata, revocable by you
The gateway, its configuration and its change control	A policy file you review and diff before it loads
The decision on what a ceiling should be	The arithmetic that shows what each ceiling would cost and what it would stop
Your systems of record for finance, risk and observability	The workload model that connects them, and the FOCUS export back

24. Summary: the ten questions, answered

Each answer with the section that covers it in full.

Question	Answer	Section
What is AICost?	A cost and governance layer for enterprise AI workflows. 123 cost and governance decision engines that combine into a chain built for each workflow, reachable as MCP chains and as calculators	1, 2, 3
Why do we need it?	Because an AI workload has six to eight cost layers and most tooling prices one of them, so the number a CFO receives is accurate about a minority of the cost	1
How is it different?	The category is retrospective and prices inference. This is prospective, prices the whole workflow, gates on quality, then enforces a ceiling and produces the evidence	2, 3, 5
What does it connect to?	AWS and Bedrock by one-click CloudFormation. Azure and Foundry by cost export. GCP by billing export. Direct vendor APIs. LiteLLM, Cloudflare, Portkey, Kong, OpenRouter, Vercel, Helicone	6
What data does it require?	Read-only cost and usage, a team-scoped key convention, a workload description, and an owner per workload. No prompts, no completions, no customer content	6
How does it reduce costs?	Caching, batch, routing and placement, each priced against your shape and gated on quality, then compiled into an enforced rule. Plus roughly 3 to 4 percent of inference spend on static traffic	7, 10
How does it handle governance?	The assessment is the engine, so it is re-run rather than re-written. 247 controls across four frameworks, each priced. Eligibility derived per route. Evidence in a hash-chained journal	9, 14
What happens in a pilot?	Seven phases, roughly three weeks to first enforcement, from connector exchange to a priced control register	16
How will we measure success?	Five objective criteria your own team can verify, including a deliberately triggered enforcement event and a forecast accuracy percentage reported whether it flatters us or not	16
What will enterprise deployment look like?	Every lane, every business line, one ledger, a chain per workflow family, a policy per team from one rule set, and either we operate it or your team does	16

Prepared 12 September 2026 by CloudIntelligence.ai LLC, Irvine, CA and Austin, TX. Scenario figures are illustrative of enterprise structure and do not represent any customer actual spend.

Appendix. Example scenarios: AICost in specific enterprise workflows

Four worked cases, each assembled from the blocks described earlier

The scenarios that follow are not product modes. Each is a chain composed from the engines, gates and enforcement points already described, arranged for a workflow with its own cost shape and its own governance obligation. They are included so the method can be checked against a situation you recognise rather than accepted in the abstract.

Scenario	The workflow	What it demonstrates
One	Distributed engineering running agentic loops	Four teams needing opposite policies from one rule set, and how a runaway loop is bounded
Two	Customer support with retrieval, fine-tuning and human review	Why a cost chain that stops at the token bill recommends changes that raise total cost
Three	Many pilots becoming production across functions	How tags established at pilot time make the twentieth use case cost what the second one did
Four	Continuous governance	Controls evaluated against live data on a schedule, with evidence accumulating as a by-product

How to read these

In each case the engines come first and the limit comes last. A ceiling set without knowing which cost driver it protects is arbitrary, and an engineering organisation will treat it that way. A ceiling traced to a specific driver, with the quality condition that permits the optimisation stated alongside it, is a policy the same organisation will approve and deploy.

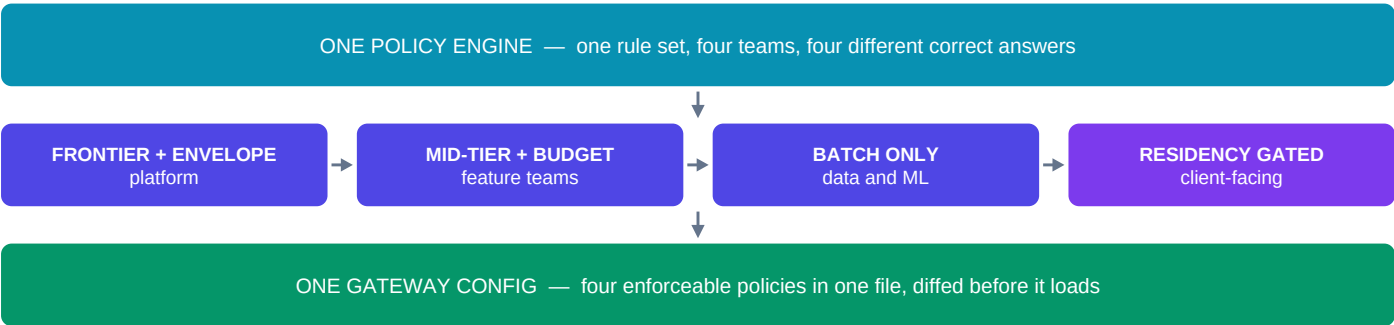
A1. Scenario one: distributed engineering running agentic loops

The runaway risk, and two teams that need opposite policies

The objective is not to give everyone the cheapest model. It is to spend premium intelligence where it creates engineering value, and contain low-value or runaway consumption everywhere else. Those are different goals and they produce different policies.

A distributed engineering organisation is the hardest case for a single corporate policy, because the teams have genuinely different needs and both answers must be defensible to the same auditor.

Team	Work	Why the obvious policy is wrong for them	Compiled policy
Platform and core services	Long agentic refactors across large repositories. Multi-hour loops, deep context	A mid-tier model loops, fails and costs more in aggregate than the frontier model that finishes	Frontier eligible. Per-run envelope and ceiling
Product feature teams	Code assist, review, test generation. Ten times the call volume, bounded scope	Frontier capability here is pure waste, at ten times the volume	Mid-tier only. Per-key daily budget, eval gate on swaps
Data and ML engineering	Offline evaluation, synthetic data, enrichment. Latency irrelevant	Realtime pricing on work that could wait is the most common silent overspend	Batch-only route. Monthly budget
Client-facing solution engineers	Client commercial data in prompts. Obligations vary by engagement	A cheaper model that fails residency is not cheaper, it is ineligible	Jurisdiction and retention gates before price



The teams never negotiate with each other. The engine resolves the differences and the gateway enforces them.

The full engineering chain, and what each engine contributes

Engine	What it contributes
AI Seat ROI	Whether paid seat capacity is creating enough developer value to justify the licence
API against subscription break-even	Where seat economics and agent economics cross over, per developer population
Agent Loop Cost	Turns, context growth, tool-result accumulation and runaway tail risk per task
Context Compaction	How much cost comes out by reducing accumulated context, before changing model quality
Prompt Cache ROI	Repeated prefix, system and tool context, and where cache economics actually matter
Multi-Model Router	Not the cheapest available model. The cheapest eligible model that meets the quality threshold
Eval Benchmark Cost	The cost of proving a cheaper route still meets the engineering acceptance threshold

Engine	What it contributes
Stacked Savings	Caching, context reduction and routing combined in sequence, so savings are not double counted
Agentic Variance Reserve	A forecast range rather than a false-precision point, for workloads whose turns vary
Cost Firewall	Compiles the accepted result into model corridors, task budgets, turn and context limits, fallback rules and circuit-breaker behaviour

Why the chain starts with cost structure rather than with a limit

Every engine above runs before any runtime restriction is imposed. A limit set without knowing which lever it protects is arbitrary, and an engineering organisation will treat it that way. A limit that can be traced to a specific cost driver, with the quality condition that allows the optimisation stated alongside it, is a policy the same organisation will approve, version and deploy.

The runaway, bounded

An agent loop with unbounded turns is the most common cause of a surprise invoice, and the reason is structural: cost grows with context as the loop accumulates history, so the last turn is far more expensive than the first. Four blocks bound it.



If a loop misbehaves at 2am it is constrained, then downgraded, then stopped, and every step is in the ledger by morning.

And the routing decision, for this scenario specifically

Feature-team code assist is the textbook static workload: one model, stable shape, high volume, for months. Moving that share to direct vendor keys recovers most of the gateway fee on it. Platform engineering, which switches models and needs failover during long refactors, should stay on the gateway and pay the fee, because it is buying something real. Same organisation, same policy engine, two different answers.

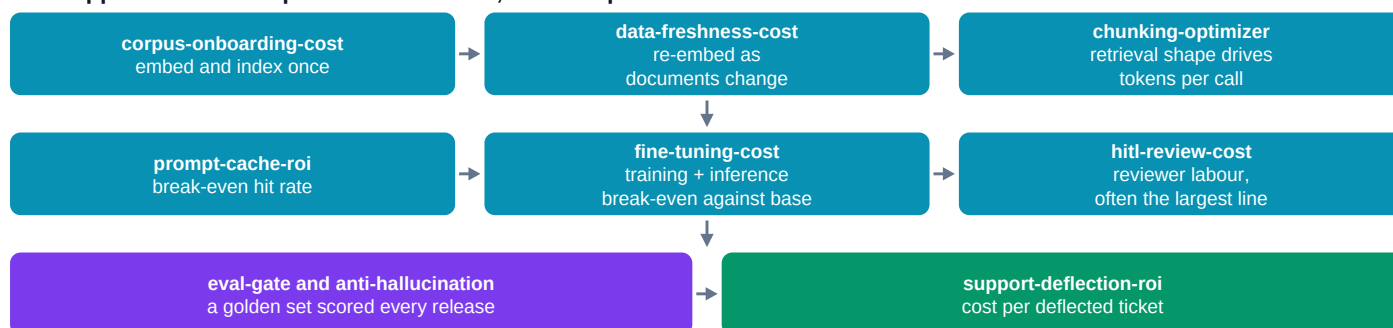
A2. Scenario two: customer support with RAG, fine-tuning and human review

Where the token bill is the smallest line in the workflow

The customer does not experience tokens. The customer experiences a resolved case, a voice interaction or an escalation, so the economic model has to cover the whole outcome rather than the generation step inside it.

A production customer-experience platform can combine corpus onboarding, OCR or parsing, embeddings, vector storage, hybrid retrieval, context assembly, model generation, tool calls, agent loops, speech-to-text, text-to-speech, guardrails, evaluation, observability, human escalation and ongoing data refresh. **A pilot hides most of these**, because volume is low and the team is still focused on answer quality.

The support chain: cost per deflected ticket, not cost per token



Deflection rate is the metric support leaders already report, so the number needs no translation to be usable.

The decision this chain usually settles

Fine-tune a smaller model, or keep prompting a larger one? It is an arithmetic question with four terms that are rarely counted together: the training run, the custom-model storage and hourly capacity charges which are billed separately from base-model inference on every major cloud, the inference delta, and the human review rate at each quality level. **A cheaper model that escalates more often moves cost from the token bill to the reviewer**, which looks like a saving in a FinOps dashboard and is not one.

The failure this chain is designed to catch

Optimising only the model price can miss the dominant cost driver entirely. Human escalation, oversized retrieval context, frequent re-embedding, long voice sessions, repeated agent turns or expensive evaluation may each matter more than the base model rate. A local saving that moves cost into human escalation, quality failures or downstream operations is not a saving, and only a chain that prices the whole outcome will show that.

Why the anti-hallucination gate sits inside the cost chain rather than beside it

Continuous evaluation costs money: a golden set scored every release by an LLM as judge, plus sampled human review. It is tempting to treat that as a quality budget and keep it out of the cost model. But it is precisely what determines whether a cheaper model is admissible, so a cost chain that excludes it can only ever recommend the cheap option. The gate belongs in the chain because the saving is not real until quality has held.

A3. Scenario three: many pilots becoming production

Different functions, different models, different tags, one accountable number

Most enterprises do not begin with a platform decision. They begin with four or five pilots in different functions, each choosing a model for its own reasons, each with its own key. The governance problem appears at the moment those pilots become production and somebody asks for the total.

Function	Pilot shape	Model posture	Tags carried from day one
Marketing	Copy generation and campaign drafting, public data only	Cheapest capable tier. Frontier is not justified by the work	team=marketing, data_class=public, cost_centre
Customer support	RAG assistant over internal knowledge	Economy primary, escalation to mid-tier	team=support, workload=deflection, agent
Engineering	Code assist and agentic refactor	Split by sub-team, frontier through to mid-tier	team=platform or feature, agent, repo
Legal and compliance	Contract review over regulated material	Narrowest allowlist, certified surfaces only	team=legal, data_class=regulated, residency
Finance	Analysis over commercial data	Mid-tier, zero retention required	team=finance, data_class=confidential

The same path for every function, which is what makes it scale



A pilot that carried a team tag and a workload tag from day one becomes a production line item without rework.

How this scales

Because tags are established at pilot time and policy is derived from the same eleven questions for every function, adding the twentieth use case costs roughly what the second one cost. The alternative, which is how most enterprises arrive at this point, is twenty separately negotiated arrangements that nobody can roll up and no auditor can test.

Marketing is a useful contrast, because it needs almost none of this. Public data, low sensitivity, no residency question, and its value sits in seat and subscription decisions rather than token routing. Including it in the same framework costs nothing, and it means the function with no governance burden still appears in the same total as the one carrying the heaviest.

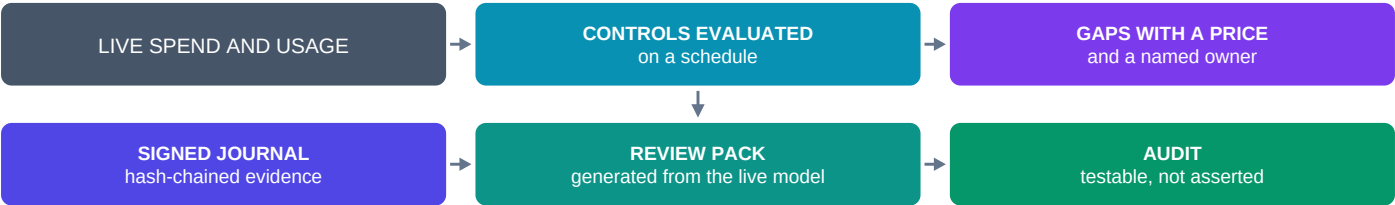
A4. Scenario four: continuous governance

A running process with accumulating evidence, not an annual document

Assessment documents go stale the week they are signed. A model is swapped, a vendor changes a retention default, a team ships an agent nobody registered, and the document no longer describes the organisation. Continuous governance means controls are evaluated against live data on a schedule, and the evidence accumulates as a by-product of the work rather than as a project before an audit.

Control theme	What changes between reviews	How it stays current
AI inventory and ownership	Teams ship new use cases weekly	Inventory built from metered spend, so an unregistered workload shows up as an unattributed line
Model and route eligibility	Vendors add, retire and reprice models continuously	Re-derived against a dated capability capture, per route rather than per model
Vendor attestations	Retention and training defaults differ by plan and surface, and change	Captured from vendor pages with the date they were read, not remembered
Evaluation currency	The evaluation behind a production model ages quietly	The eval gate reports staleness, and a swap without a current eval is blocked
Human oversight	Consequential decisions need a person, and proof one acted	Oversight log entries in the signed journal
Incident lifecycle	A reportable incident starts a regulatory clock	Article 73 clock with reporting deadlines attached
Access recertification	People change teams; keys outlive them	Quarterly recertification against live key usage rather than against a roster
Shadow AI	Spend appears from tools nobody approved	Exposure findings with the spend attached, which is what gets the meeting
Cost of the control set	Volumes change, so the compliance premium changes	Priced continuously rather than estimated once

The governance loop



247 controls across EU AI Act, ISO 42001, NIST AI RMF and CSA AICM. Each one priced, owned and evidenced.

AICOST.AI

CLOUDINTELLIGENCE.AI LLC

What AICost does, in twelve lines

- | | | |
|----|-------------------------------------|--|
| 01 | Workload first | Start from the AI use case, not the cloud bill |
| 02 | Full AI TCO | Inference, retrieval, evaluation, security, human review, governance |
| 03 | Plan before spend exists | Forecast pilot, production and scale before the first invoice |
| 04 | Cost and governance together | Eligibility, oversight and audit are part of the design |
| 05 | Deterministic engines | Same inputs, same answer, auditable and machine callable |
| 06 | Policy-ready outputs | Budget caps, allowlists, route rules, turn limits, approval gates |
| 07 | Out-of-band control plane | We never proxy your inference |
| 08 | Compose, enforce, prove | Chain engines, enforce at runtime, prove with actuals |
| 09 | Forecast to actual | Reconcile bills, telemetry and showback against the plan |
| 10 | Vertical and function aware | Marketing, engineering, support and legal are modelled differently |
| 11 | Partner with the customer | Workflows and governance co-designed, not configured |
| 12 | Customise where it matters | Custom engines, policy packs and runtime rules |
-

**Innovate with AI in confidence across your enterprise,
with governance and cost controls from day one**

IRVINE, CALIFORNIA

949 378-7860

AUSTIN, TEXAS

469 951-8383