TEST / CONTROL / AI / MULTIPLIED

SUMMARY REPORT · COMMERCIAL AI BILLING

Paying for AI

Three ways to buy it, four discounts that stack, and five ways to bill it back to the teams that spent it.

3 BUYING MODELS / 4 STACKING DISCOUNTS / 5 ATTRIBUTION OPTIONS

COMMERCIAL DEPLOYMENT · NO CUI · PREPARED 5 SEPTEMBER 2026

Alexander's Test & Control Solutions

Huntsville, Alabama · (256) 341-7917 · Billy@AlexanderTestControl.com

alexandertestcontrol.com

SECTION 1

Three ways to buy it

Almost every argument about AI cost in a commercial business is really an argument about which of these three you bought, because they fail in completely different ways.

Model	You pay for	Fails when
Per seat	A licence per person per month, flat. Copilot, ChatGPT Enterprise, Gemini for Workspace and the rest	Half your seats go unused and you keep paying. There is no usage signal in the bill, so you cannot tell who needs it
Per token	Input and output tokens actually consumed through the API	Nobody owns the bill. A single unbounded agent loop can spend a month's budget over a weekend
Reserved capacity	Throughput units bought in advance, billed whether used or not	You commit before you know your load curve and run at thirty percent utilisation for a year

Most organisations of any size end up with all three, which is fine. What is not fine is buying the second one without building the attribution in section 4 at the same time, because token billing with no attribution is the only one of the three that can surprise you.

> **The per seat trap that finance does not see**
> A per seat licence is predictable, which is why finance likes it, and it is the reason nobody measures anything. At roughly $19 to $60 per user per month, a hundred person company is spending $23,000 to $72,000 a year with no usage data attached to any of it. Run an API pilot alongside for one quarter and you will usually find that a third of your seats generate almost all the value, a third generate some, and a third have not opened the tool since onboarding. That is not an argument against seats. It is an argument for knowing which third is which before you renew.

SECTION 2

How token billing actually works

Four separate rates, not one. Most cost surprises come from a team that modelled the first rate and ignored the other three.

Rate	What it covers	Typical relationship to input
Input	Everything you send: system prompt, retrieved context, conversation history, tool definitions	The baseline
Output	Everything the model generates, including reasoning tokens on models that produce them	Four to five times the input rate
Cached input	Repeated prefix content served from cache	Ten percent of the input rate on current flagship models
Cache write	The one time cost of putting a prefix into cache	About 125 percent of the input rate

Two things people get wrong

First, output is the expensive side and it is the side you control least. A model asked an open question writes an essay. The same model asked for a JSON object with four fields writes four fields. Constraining output format is a cost control, not just an engineering preference.

Second, the tool definitions in an agentic workflow are input tokens on every single call. A large tool surface is a fixed tax on every request in the conversation, and it is invisible in most cost dashboards because it looks like ordinary input.

SECTION 3

Four discounts, and they stack

Discount	Saving	What it costs you	Right for
Prompt caching	**Up to 90 percent on the cached portion**	A 25 percent premium on the write, and you must structure the prompt so the stable part comes first	Anything with a long fixed preamble: a system prompt, a policy document, a code base, a product catalogue
Batch processing	**50 percent across the board**	Results come back within 24 hours rather than immediately	Overnight document processing, bulk classification, report generation, embedding a corpus
Model right sizing	**Often 80 to 95 percent**	Real evaluation work to prove the smaller model is good enough	Extraction, classification, routing, summarisation. Most production traffic, in other words
Committed use	**Varies by provider and volume**	A commitment you have to actually consume	Steady predictable load only, and only after a quarter of measurement

The stacking arithmetic, because it is larger than people expect

Take a nightly job that summarises 2,000 documents against a fixed 6,000 token instruction preamble. Run naively on a flagship model at list price, call it $340 a month. Cache the preamble and the repeated portion drops to a tenth. Send it as a batch and the whole thing halves again. Move the extraction step to a small model and the remaining input cost falls by roughly another order of magnitude. The same workload lands between $8 and $20 a month. Nothing about the output changed. The three decisions took an afternoon.

This is the single highest return work in commercial AI adoption and it is almost never done, because it is unglamorous and nobody owns it. It is also the reason a cost review is usually a better first engagement than a new build.

SECTION 4

Five ways to bill it back to teams

This is the part organisations postpone and then regret. Attribution has to be designed in at the start, because none of these options are retroactive. Cost allocation tags in particular only tag spend incurred after you activate them.

Option 1 · One key, no attribution

The default state. A single API key in an environment variable, one line on the corporate card, and no way to answer who spent it. Fine for a pilot with one team. Indefensible past about $2,000 a month, because at that point somebody in finance will ask and there will be no answer.

Option 2 · Provider native projects and workspaces

Every major provider now offers a project or workspace construct with its own keys, its own spend view and often its own limits. Zero infrastructure, works on day one, and it is genuinely enough for many companies.

Gives you	Does not give you
A separate key per team, spend visible per project, usually a hard or soft limit per project	Cross provider rollup, per user attribution inside a team, model level policy, or any single place to look when you use more than one provider

The ceiling is real: the moment you use two providers you have two consoles, two exports and a spreadsheet reconciling them.

Option 3 · Cloud native tagging

If you buy your inference through a hyperscaler, the cloud's own cost machinery does the work and the spend lands in the same tooling as the rest of your bill. On AWS this is the strongest of the five for a company already running there.

Mechanism	How it attributes	Note
Bedrock application inference profiles	A tagged wrapper around a foundation model. Route each team's calls through its own profile ARN and the tag flows to Cost Explorer and the Cost and Usage Report	No extra charge, standard token rates. Tags take up to 24 hours to appear and are not retroactive
Bedrock cost allocation by IAM user and role	Attributes spend to the calling principal directly, without needing a profile per team	Added April 2026. Useful where one application serves many teams under distinct roles
Azure OpenAI behind API Management	A subscription key per team, with usage and quota enforced at the gateway	You are running APIM, which is its own operational commitment

Option 4 · An LLM gateway with virtual keys

A proxy in front of every provider. Each team, and if you want each person, gets a virtual key. The gateway prices every request from the provider's own rate card, writes it to a database, enforces the budget before the spend happens rather than reporting it afterwards, and gives you one place to look across every model you use.

This is the option that scales, and it is the subject of the companion report on doing it inside AWS GovCloud. The control surface is deep. Taking LiteLLM as the reference implementation, budgets can be set at seven levels at once:

Level	Set via	Parameter
Global proxy	`config.yaml`	`max_budget, budget_duration`
Team	`/team/new, /team/update`	`max_budget`
Team member	`/team/member_update`	`max_budget_in_team`
Internal user	`/user/new`	`max_budget`
Virtual key	`/key/generate`	`max_budget`
Per model, per key or user	`enterprise feature`	`model_max_budget`
End customer	`/budget/new, /customer/new`	`user identifier on the request`

Rate limits sit on the same hierarchy through tpm_limit, rpm_limit and max_parallel_requests. Budget periods are set with budget_duration and accept values like 30s, 30m, 30h and 30d. Left unset, a budget never resets, which is a foot gun worth knowing about.

Option 5 · Stay on seats

Entirely legitimate and frequently correct. If your use is individual productivity rather than product features, per seat licensing needs no attribution machinery at all because the allocation is the headcount. The discipline you still need is a quarterly usage review so you are not renewing seats nobody opens.

SECTION 5

Choosing between them

	Setup effort	Cross provider	Enforces or reports	Right when
1. Single key	None	No	Neither	Pilot, one team, under $2,000 a month
2. Provider projects	Hours	No	Reports, sometimes limits	Two to five teams, one provider
3. Cloud tagging	Days	Within one cloud	Reports	You already buy inference through AWS or Azure and want it in the same bill as everything else
4. LLM gateway	One to three weeks	Yes	Enforces before spend	Multiple providers, real chargeback, or any regulated environment
5. Seats	None	n/a	Neither	Individual productivity rather than product features

The honest sequence for a commercial business
Start at option 2 because it costs you an afternoon. Move to option 3 if you are already committed to one hyperscaler and want the spend in your existing cost tooling. Move to option 4 when you have more than one provider, when a team needs a hard ceiling rather than a monthly surprise, or when someone needs to answer an auditor. Do not build a gateway for three engineers and one provider. Do not run five teams and two providers on a shared key.

SECTION 6

Chargeback or showback

Having the numbers is not the same as doing anything with them, and the choice between the two changes behaviour more than any technical control.

	What it is	What it does to behaviour
Showback	Each team sees its own spend. The cost stays central	Mild. Teams look at the number once, feel briefly responsible, and carry on. Good for the first two quarters while you learn the shape of the load
Chargeback	The spend lands on the team's own budget line	Strong, and not always in the direction you want. Teams optimise hard, which is good, and some quietly stop using AI for work where it would have paid off, which is not. Introduce it only once your attribution is trustworthy, because a wrong chargeback number destroys the programme's credibility in one meeting

The practical middle: showback for two quarters, then chargeback only on spend above a per team floor. The floor keeps ordinary experimentation free and puts the pressure on the workloads big enough to be worth engineering.

SECTION 7

What it actually saves, and what the evidence really says

This section is deliberately unflattering, because every vendor document you have read on this subject cites the same two favourable studies and none of the unfavourable ones.

Finding	Study	What it actually measured
55.8 percent faster task completion	GitHub Copilot controlled experiment	A single scoped task, implementing an HTTP server in JavaScript, by developers new to the problem. Completion fell from 2h41m to 1h11m and success rose from 70 to 78 percent. Real, and narrow
19 percent slower	METR randomized controlled trial, July 2025	16 experienced open source developers, 246 tasks, on repositories they already knew well. They were slower with AI, and predicted they had been 20 percent faster. METR revised the experiment design in February 2026
66 percent spend more time fixing almost right code	Stack Overflow Developer Survey 2025	Self reported, across 51 percent of professional developers who use AI daily
Review time up as much as 91 percent	Published DORA-aligned analyses, 2025	High AI adoption raises pull request volume and shifts the bottleneck to review. Velocity moved. Throughput did not

The synthesis, which is more useful than either headline

AI helps most where the person is least expert: unfamiliar languages, boilerplate, first drafts, code in a stack they touch twice a year, and every kind of document nobody enjoys writing. It helps least, and can actively hurt, where the person is deeply expert in a codebase they have lived in for years, because reviewing plausible wrong output costs more than writing correct output from memory. Any business case that assumes a flat percentage gain across every engineer is wrong in both directions at once. Target the unfamiliar work.

Where the savings are actually reliable

Setting the engineering argument aside, the returns that hold up under measurement are duller and more dependable: first draft document generation, meeting summarisation, retrieval over internal knowledge so people stop asking colleagues questions the wiki already answers, classification and routing of inbound work, and data cleanup. None of these are exciting. All of them are measurable, all are low risk, and none require anybody to trust generated code.

Measure before you start. Baseline the hours in the target workflow, in writing, before the tool arrives. Without that number the ROI conversation twelve months later is two people exchanging opinions.

SECTION 8

The governance that has to arrive with the billing

Cost control and governance are the same project, and the gateway that gives you one gives you the other. If you are building attribution anyway, take the rest of it while the work is open.

Control	Why it belongs here
Written acceptable use policy and an approved tool list	Governance first, tools second, training third. Without it your people are already pasting company data into consumer chatbots, and the billing work will show you exactly how much
Data classification before tool selection	What data may touch which platform is the first question. It is cheaper to answer it before the contract than after the incident
Per team model allow lists	Not every team needs the frontier model. Restricting the model list per virtual key is simultaneously a cost control and a risk control
Human in the loop where output is customer facing or contractual	Non negotiable. Put it in the policy, not in people's judgement
Logging that can answer who asked what, when	You will need this for an incident, for a customer questionnaire, and eventually for an auditor. Retrofitting it is painful

If any of your customers are defense contractors, or you intend to pursue that work, building these habits now shortens the later compliance path considerably. The companion report covers what changes when CUI enters the picture.

SECTION 9

Sources

What	Where
Prompt caching at 10 percent of input rate, 25 percent write premium, 5 minute default TTL; batch at 50 percent	Anthropic and OpenAI published pricing documentation, 2026
Bedrock application inference profiles, cost allocation tags, no additional charge, 24 hour tag latency, not retroactive	docs.aws.amazon.com/bedrock/latest/userguide/cost-mgmt-application-inference-profiles.html
Bedrock cost allocation by IAM user and role, April 2026	aws.amazon.com/about-aws/whats-new/2026/04/bedrock-iam-cost-allocation
LiteLLM budget hierarchy, rate limit parameters, budget_duration values	docs.litellm.ai/docs/proxy/users
55.8 percent faster task completion, 2h41m to 1h11m, 70 to 78 percent success	Peng et al., The Impact of AI on Developer Productivity: Evidence from GitHub Copilot, arXiv 2302.06590
19 percent slowdown for experienced developers on familiar repositories; design revision February 2026	METR, Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity, arXiv 2507.09089, and metr.org
51 percent daily AI use, 66 percent report more time fixing almost right code	Stack Overflow Developer Survey 2025

What	Where
Review time increases under high AI adoption	Published DORA-aligned industry analyses, 2025 and 2026
Enterprise foundation model API revenue and AI investment figures	Industry analyses of 2025 spend, cited in AI gateway market reviews, 2026

What is measured here and what is judgement

Measured: the discount percentages, the LiteLLM parameter names and budget hierarchy, the Bedrock attribution mechanisms and their limitations, and every study result in section 7. Judgement: the stacking arithmetic in section 3, the $2,000 a month threshold in section 4, the showback then chargeback sequence in section 6, and the synthesis in section 7. The arithmetic is reproducible from published rate cards. The thresholds are opinions formed from doing this, and you should argue with them.