

The AI Agent Cost Playbook

Cut your AI agent bill without losing quality. Work out what your agent will cost and where to cut, in one afternoon. Includes a spreadsheet model.

```
cost/month = tasks × steps
              × (tokens_in × price_in + tokens_out × price_out)
              + tools
```

WORKED EXAMPLE FROM CHAPTER 1 (list prices, 30 Oct 2026)

	\$1,238 · baseline coding agent
	\$639 with prompt caching
	≈\$215 caching + routing

What's inside

Before you start	3
The 12 words you need, and how to use this playbook.	
01 • The agent cost equation	5
Where the money goes, why input dominates, and how to measure your own agent.	
02 • What drives token consumption	8
Nine drivers, an estimation formula, a worksheet and a fix for each.	
03 • Benchmarks by workload	11
Four common agents priced on 12 models, with official prices.	
04 • The 7 levers, ranked	14
Caching, routing, pruning, batch, output limits, fewer steps, cheaper models.	
05 • Decide before you generate	17
Cheap classifier models (Jev) as a gate in front of your LLM.	
06 • Tooling costs	20
Observability, web data, sandboxes, vector DBs, search: free tiers and price jumps.	
07 • Budgets and alerts	22
Per-task ceilings, step limits, spend alerts and a weekly cost review.	
08 • Case study: a coding agent, step by step	24
Applying the levers in order, with every number shown.	
09 • Using the spreadsheet	26
Inputs, scenarios, the driver worksheet and the price table.	
Final checklist & price sources	28
Everything to do, in order, plus where every price came from.	

Comes with **agent-cost-model.xlsx** (Excel and Google Sheets). Prices are official list prices checked on 30 Oct 2026; providers change them often, so re-check before big decisions. Free updates for 12 months.

The 12 words you need

You don't need to be a machine-learning engineer to cut your agent's bill. You need a handful of terms and a calculator. Each term below is used throughout the playbook, so it's worth five minutes now.

Token: the unit AI models read and write, and the unit you pay for. A token is a piece of a word: in English, 1 token $\approx \frac{3}{4}$ of a word, so 1,000 tokens $\approx$ 750 words $\approx$ 1.5 pages. Code and non-English text use more tokens per word. Each model family has its own "tokenizer", so the same text can be a different number of tokens on different models (Anthropic notes its newer models produce about 30% more tokens for the same text).

Price per 1M tokens: providers quote prices "per million tokens" (written /1M or /MTok). \$2 per 1M input tokens means 1,000 tokens cost \$0.002. Always convert to *cost per task* before you compare.

Input tokens vs output tokens: input is everything you send to the model (instructions, chat history, documents, tool results); output is what it writes back. Output is usually 4–8× more expensive per token, but agents send far more input, so input is usually the bigger line.

Context (context window): everything the model sees in one request. The "context window" is the maximum size, e.g. 1M tokens. A bigger window is not free: you pay for every token you actually put in it, on every request.

System prompt: the fixed instructions at the top of each request ("You are a support agent..."). It is re-sent every time, so a 3,000-token system prompt costs 3,000 tokens per call.

Agent, step and task: an *agent* is a model in a loop that can use tools. Each trip round the loop is a *step* (one API call). A *task* is the whole job the user asked for, e.g. "fix this bug", which might take 15 steps.

Tool call: when the model asks your code to do something (search, read a file, run tests). The tool's result is pasted into the context as input tokens for the next step, which is why big tool results are expensive.

Prompt caching: the provider remembers the start of your prompt (the "prefix") for a few minutes. When the next request starts with exactly the same text, those tokens are billed at a big discount, e.g. GPT-6.1 Sol charges \$0.10 instead of \$2.00 per 1M for cached input (–95%). Writing to the cache can cost a little extra.

Routing: sending each step to the cheapest model that can do it well, e.g. a small model for "read this file" and a big model for "design the fix".

Batch API: a cheaper lane (usually –50%) for work that can wait minutes to hours, like nightly reports or bulk classification.

Reasoning ("thinking") tokens: some models think before answering. Those hidden tokens are billed as output (Google says so explicitly: "output price including thinking tokens"). They can be the hidden reason a bill is high.

Usage log: every API response includes a `usage` block with the exact token counts for that call. This is your most important cost data. Save it.

How to use this playbook

1. **Read Chapter 1 and do its exercise.** You will know your real cost per task in about 30 minutes.
2. **Use Chapter 2 to find your biggest driver,** then jump to the matching lever in Chapter 4.
3. **Change one thing at a time,** re-measure, and keep a short log of before/after cost *and* quality.
4. **Set the guard rails in Chapter 7** before you scale up.

All prices are official list prices in USD, standard tier, checked on 30 Oct 2026 (sources on the last page). Workload sizes in examples are stated assumptions, not measurements of any real company. Your numbers will differ: that's what the exercises are for.

The agent cost equation

Most people look at a model's price and multiply. Agents don't work that way: the bill depends on how many times the loop runs and how much text is re-sent each time. This chapter gives you the one formula you need and shows you how to fill it in with your own numbers.

1.1 The formula, in plain words

```
cost per step = input tokens × input price + output tokens × output price
cost per task = sum of all steps in the task
cost per month = cost per task × tasks per month (+ tool costs, Chapter 6)
```

Prices are per 1M tokens, so divide by 1,000,000. Example: one step with 20,000 input tokens and 1,500 output tokens on GPT-6.1 Sol (\$2 in / \$10 out) costs $20,000 \times 2 / 1M + 1,500 \times 10 / 1M = \$0.04 + \$0.015 = \textbf{\$0.055}$. Small, until you multiply it.

1.2 A worked example: a coding agent

Assume a coding agent that handles **50 tasks a day** (1,500 a month), each taking **15 steps**, with about **20,000 tokens of context in** and **1,500 tokens out** per step.

Line	Tokens / month	Price / 1M	Cost / month
Input (context)	450,000,000	\$2.00	\$900.00
Output	33,750,000	\$10.00	\$337.50
Total · GPT-6.1 Sol			\$1,237.50

Input is **73%** of the bill even though output is 5× more expensive per token. Why? Because every step re-sends the instructions, the conversation so far and every tool result. The model writes 1,500 tokens but reads 20,000.

1.3 Lever 1: pay less for the context you re-send

Most of those 20,000 tokens are the same on every step: system prompt, tool definitions, earlier messages. With **prompt caching**, repeated text is billed at the cached rate. For GPT-6.1 Sol that's \$0.10 instead of \$2.00 per 1M (OpenAI pricing page, 30 Oct 2026). If 70% of input is a cache hit:

```
input = $900 × (0.30 + 0.70 × 0.10/2.00) = $900 × 0.335 = $301.50
total = $301.50 + $337.50 = $639.00 (-48%)
```

OpenAI also lists a cache *write* price (\$2.50 per 1M for a 30-minute write on Sol); if your prefix is reused many times within the window, the write cost is a rounding error, so we leave it out here. Chapter 4 shows how to get high hit rates.

1.4 Lever 2: route the routine steps

Many steps are routine: open a file, run the tests, reformat. If 70% of steps go to Claude Haiku 5.5 (\$0.10 in / \$0.50 out / \$0.01 cached, for prompts up to 100K tokens) and the rest stay on Sol, both with 70% caching:

```
Sol 30% of steps    = $191.70
Haiku 70% of steps = $23.47
total               = $215.17  (-83% vs baseline)
```

Watch out: caches are per model. If you switch models mid-task, the new model starts with an empty cache. Route at the level of whole sub-tasks where you can, not single steps.

Rule of thumb. Before you compare model prices, measure *input tokens per step × steps per task*. If that is more than 10× your output, caching and context pruning will save more than switching models.

1.5 Three mistakes that make forecasts 2–5× too low

1. **Ignoring retries.** A failed step re-runs with the full context. 10% retries ≈ +10% cost, more if whole tasks restart.
2. **Forgetting tool payloads.** One web page is ~2,500 tokens; a research PDF can be ~125,000 (Anthropic's own estimates). These land in the input of *every later step*.
3. **Assuming context stays flat.** Context grows each step, so step 15 is much bigger than step 1. Use the *average* step size from real logs, not the first step.

EXERCISE • MEASURE YOUR REAL COST PER TASK (30 MIN)

1. Pick 10–20 recent, typical tasks your agent ran (or run them now).
2. For every API call, save the `usage` block: input tokens, cached input tokens, output tokens. Most frameworks log it; otherwise print `response.usage`.
3. Per task, add up: total input, cached input, output and number of calls (steps).
4. Multiply by your model's price from its official pricing page. Write down cost per task (average and the most expensive one).
5. Multiply the average by tasks per month. Compare with your provider's billing page: if they differ by more than 20%, something (retries, a background job) is missing from your logs.

WHAT YOU LEARNED

- Agent cost = steps × (input × price + output × price) × tasks, plus tools.
- Input usually dominates because the context is re-sent every step.
- Caching and routing cut the example from \$1,238 to about \$215 without changing the main model for the hard steps.

TRY THIS NOW (≈15 MIN)

Open the spreadsheet, enter your tasks/month, steps/task, average input and output per step and your model. Note your baseline monthly cost: every later chapter is measured against it.

FREE SAMPLE ENDS HERE

What's in the rest of the playbook

You've read the glossary and Chapter 1. The full playbook (29 pages + spreadsheet) continues step by step:

- **02 • What drives token consumption:** nine drivers, a worksheet and a fix for each.
- **03 • Benchmarks by workload:** four common agents priced on 12 models.
- **04 • The 7 levers, ranked:** caching, routing, pruning, batch, output limits, fewer steps, cheaper models, each with the maths shown.
- **05 • Decide before you generate:** cheap classifier gates in front of your LLM.
- **06 • Tooling costs:** free tiers and price jumps for observability, web data, sandboxes and vector DBs.
- **07 • Budgets and alerts:** step limits, cost ceilings and a 15-minute weekly review.
- **08 • Case study:** a coding agent, lever by lever, every number shown.
- **09 • Using the spreadsheet** and a final checklist.

GET THE FULL PLAYBOOK

\$29 + tax (tax calculated at checkout) · PDF + Excel/Google Sheets model · 14-day money-back guarantee.

<https://agentcosthub.com/playbook/> · questions: hello@agentcosthub.com

Written with help from AI and checked by a human. Prices are official list prices checked 30 Oct 2026.