FIELD GUIDE · JULY 2026

THE FABLE 5 COST PLAYBOOK 10 MOVES.

Fable 5 leaves your plan on July 19. After that it is $10 / $50 per Mtok — exactly 2× Opus 4.8, the highest rate Anthropic lists. Here is how to keep the frontier and lose the bill.

Deep — Senior SRE, 10+ yrs. Independent AI safety researcher.
Every number in here is sourced to Anthropic docs or a named benchmark. Nothing is estimated.

WHAT ACTUALLY CHANGES.

Through **July 19, 2026, 11:59:59 PM PT**, Fable 5 is included on Pro, Max, Team and premium Enterprise seats — capped at **50% of your weekly usage limit**. After that it stops drawing from your plan entirely and bills through **prepaid usage credits at API rates**.

Anthropic has moved this deadline twice already: July 7 → July 12 → July 19. It frames the change as a capacity problem, not a pricing strategy, and says Fable will return to subscriptions "as soon as capacity allows." No date has been attached to that. Plan for the meter; be pleasantly surprised if it lifts.

THE RATE CARD

MODEL	INPUT /MTOK	OUTPUT /MTOK	VS FABLE
Fable 5	$10.00	$50.00	1.0×
Opus 4.8	$5.00	$25.00	0.5×
Sonnet 5 (intro, to Aug 31)	$2.00	$10.00	0.2×
Haiku 4.5	$1.00	$5.00	0.1×
Fable 5 cached read	$1.00	—	0.1×
Fable 5 batch	$5.00	$25.00	0.5×

Every Claude model shares Fable's 1:5 input-to-output ratio, so a model's share of the input price and the output price are identical. Sonnet 5 moves to $3 / $15 after August 31, 2026.

> **READ THIS TWICE —** if you do not enable credits before the window closes, Fable access does not degrade or fall back. It **stops**. Mid-session, mid-week, wherever the pool empties. Opus 4.8, Sonnet and Haiku stay inside your normal plan limits and are unaffected.

BEFORE THE TEN MOVES: THE RULE.

Every move in this playbook is downstream of one question. Not *"is Fable 5 the best model"* — it usually is. The question is **"does this task need the best model?"**

Most agentic work is a little hard reasoning wrapped in a lot of mechanical execution. Planning an approach or catching a real bug in a diff is where Fable earns $50/Mtok. Renaming symbols across forty files, running the suite, reading logs — that is not frontier work, and you are paying frontier rates for it every turn you let it happen.

PAY FLAGSHIP RATES ONLY FOR THIS

```
[x] architecture decisions with expensive blast radius
[x] gnarly debugging where wrong guesses cost hours
[x] high-stakes review of code about to ship
[x] deep research synthesis across many sources
[x] long-horizon autonomous runs (hours, not minutes)

[ ] renaming, formatting, boilerplate
[ ] running tests and reading the output
[ ] file exploration and grep-style search
[ ] writing unit tests from existing code
[ ] anything you could hand a junior with a spec
```

The counter-intuitive part: on genuinely long, deliberate work Fable can be **cheaper** than the model with the lower sticker. A frontier physics lab reported Fable 5 was its strongest model "while using a third of the reasoning tokens." One-third the tokens at twice the per-token price is two-thirds the effective cost. A spreadsheet suite found Fable beat Opus 4.8 at **every** effort level with fewer turns, finishing 25–30% faster.

> **SO** — do not blanket-ban Fable. Ban it from the 80% of your session that is typing. That is what the next ten pages are.

01 DROP THE EFFORT

THE MOVE

```
> /effort medium

  low  · subagents, classification, lookups
  medium · the cost-saving step-down
  high · DEFAULT — complex reasoning
  xhigh · hardest coding + agentic work
```

Thinking tokens bill as **output**, at $50/Mtok. Effort is the single dial that governs how many of them you buy. Anthropic's docs are direct about it: lower effort settings on Fable 5 "still perform well and often exceed xhigh performance on prior models." Fable at medium routinely beats Opus 4.8 at its ceiling. Dropping a level on a task that already works costs you nothing.

> **THE TRAP —** Max effort "adds significant cost for relatively small quality gains" and can push the model into overthinking on structured-output tasks. Also: effort silently resets when you switch models mid-session. Re-check it after every `/model`.

02 ARCHITECT, NOT EXECUTOR

THE MOVE

```
> Plan in plan mode (Shift+Tab) on Fable
> Write the plan to a markdown spec
> Sonnet 5 executes the spec
```

Fable writes the architecture; a model at 0.2× the price does the typing. Same plan, one-fifth the execution bill. Claude Code has this wired in natively — set your model to `opusplan` and plan mode runs on the bigger model while execution stays on Sonnet, no manual switching.

> **THE TRAP —** Plan mode is also the cheapest bug-prevention you own. The most expensive failure mode in agentic coding is not an expensive model — it is twenty confident minutes spent going the wrong way.

03 MAKE IT WRITE LESS

THE HONEST MATH

```
Skills: Caveman (JuliusBrussee) · Ponytail (DietrichGebert)

  claimed  → 65–75% token cut
  measured → 65% of PROSE tokens
  prose    → ~6% of a 100K session
  real     → ~4–5% off your bill
```

These are good skills and you should install them. But run the arithmetic the benchmark tables skip: Caveman compresses only the conversational prose Claude writes back to you. In a 100K-token session that slice is roughly 6,000 tokens. Cutting 65% of it saves ~4,000 tokens — about **4%** of the session, not 75%. The repo's own README carries an honest-number warning.

> **THE TRAP —** The headline numbers come from single-shot prompts where the bare-model baseline pads its answer with prose and options. That is a conversational-baseline artifact, not a session saving. The real compounding win is the companion memory-compression tool — a leaner CLAUDE.md is input you stop paying for on *every* future session.

04 CHEAP MODELS DO THE READING

THE MOVE

```
# .claude/agents/explorer.md
---
name: explorer
description: Maps structure, finds files, traces deps.
model: haiku
tools: Read, Grep, Glob
---
Return a structured summary under 50 lines.
Do not read files that are not relevant.
```

Subagents run in **their own context window**. Haiku reads forty files and burns forty files worth of tokens at $1/$5 — and Fable only ever sees the returned summary. The expensive context never touches the exploration. Anthropic's own cost docs say it plainly: for simple subagent tasks, specify `model: haiku`.

> **THE TRAP —** Subagents **inherit your session model by default**. If you are on Fable at xhigh, every explorer you spawn is a Fable-at-xhigh explorer. This is the most expensive default in the product. Pin it per-agent, or globally with `CLAUDE_CODE_SUBAGENT_MODEL`.

05 FABLE ON ADVISOR DUTY

THE MOVE

```
> /advisor   # pick the advisor model

Executor (Sonnet/Haiku) runs the task end-to-end.
Advisor never calls tools, never talks to you. It
reads the shared context and returns exactly one
of: a plan, a correction, or a stop signal.

API: tools=[{"type":"advisor_20260301", "max_uses":3}]
     anthropic-beta: advisor-tool-2026-03-01
```

This inverts the usual orchestrator pattern: the cheap model drives and escalates instead of the expensive model decomposing and delegating. Frontier reasoning applies only where the executor is stuck. Anthropic's measured numbers: Sonnet + Opus advisor scored **74.8% vs 72.1%** solo on SWE-bench Multilingual, at **11.9% less** than Opus solo. Haiku + Opus advisor on BrowseComp: **41.2% vs 19.7%** — more than double — trailing Sonnet solo by 29% at **85% less cost** per task.

> **THE TRAP —** It costs less because the advisor emits only 400–700 tokens per consult while the executor makes fewer wrong turns. But every consult re-bills the **`full shared context`** at the advisor's rate. Uncapped on a 200K context, ten consults is 2M advisor input tokens and you have made it worse. `max_uses` is not optional — it is the cost control.

06 THE 90% DISCOUNT YOU OWN

THE MOVE

```
Fable 5 input: $10.00 /Mtok   cached: $1.00 /Mtok

The cache keys off a STABLE PREFIX, in order:
  1. tools   2. system prompt   3. conversation

Change anything early → everything after it
re-reads at full price.
```

Claude Code turns prompt caching on for you. Your entire job is not breaking it. This matters more on Fable than on any other model because the whole prefix is re-sent every turn — a 200K context is $2.00 uncached and $0.20 cached. Cache writes carry a ~25% premium, so the math favours caching anything reused more than twice.

> **THE TRAP —** The easiest way to nuke it mid-task: **`add or remove an MCP server`**. That changes the tool set at position 1 in the prefix, invalidating everything behind it. Same for editing CLAUDE.md mid-session. Freeze tools and system prompt before a long run, not during it.

07 KILL THE CONTEXT TAX

THE MOVE

```
> /context    # what is eating the window
> /mcp        # disable what you do not use
> /usage      # spend by skill / subagent / plugin / MCP

Prefer CLI over MCP where one exists:
  gh · aws · gcloud · sentry-cli
```

Every active MCP server ships schema into context. You pay for that preamble on every turn whether or not you ever call the server. Anthropic's docs recommend CLI tools over MCP equivalents for exactly this reason — a CLI adds **no per-tool listing**; Claude just runs the command. Run `/context` once and you will find things you forgot you installed.

> **THE TRAP —** MCP tool definitions are deferred by default now — only tool *names* enter context until Claude actually uses one. That helps, but it is not free, and it does not save you from a server whose tools Claude does decide to call. Disable, do not just ignore.

08 STOP RE-SENDING THE TRANSCRIPT

THE MOVE

```
> Agent writes state to progress.md
> Each turn reads the SUMMARY, not the history
> /clear between unrelated tasks
> /compact Focus on API changes and test output

# trigger compaction earlier than the ~95% default
CLAUDE_AUTOCOMPACT_PCT_OVERRIDE=75
```

Resend the full transcript every turn and per-turn input cost grows **quadratically with session length**. File-based state keeps it flat. Fable is unusually good at this: in the Slay the Spire benchmark, file-based memory improved Fable 5's performance **3× more** than it improved Opus 4.8's. The model is built to work from its own notes — let it.

> **THE TRAP —** `/compact` is not free. Mechanically it sends the entire conversation to the model as input and replaces the history with the summary — you pay full input price to re-feed a prefix you already bought, and the summary bills as output. Compaction is damage control. External state is prevention.

09 BATCH WHAT CAN WAIT

THE MOVE

```
Batch API — 50% off BOTH rates:

  Fable 5 standard: $10 / $50
  Fable 5 batch:    $ 5 / $25

Results inside 24h. Same model. Same tokens.
```

There is no quality difference and no catch. The only thing the other 50% buys you is a response before you lose patience. Migrations, eval sweeps, doc generation, nightly analytics, data enrichment, backfills — none of that needs to be interactive, and most teams route all of it through the synchronous endpoint out of habit.

> **THE TRAP —** Batch stacks with caching. It does not stack with a human waiting on the output, so do not batch anything in your inner loop — you will just wait and still pay.

10 CAP THE METER FIRST

THE MOVE

```
claude.ai → Settings → Usage → enable credits
→ set a spend limit · optional auto-reload

  one pool: claude.ai + Claude Code share it
  top-ups cap at $2,000 / day
  mobile subscribers must enable on WEB

# measure your real baseline first
npx ccusage@latest
```

Do this before you run a single agent, not after. Credits drain from one balance across the web app and Claude Code, so a runaway loop in the terminal empties the same pool your chat uses. `/usage` attributes recent spend to individual skills, subagents, plugins and MCP servers — that is your hit list.

> **THE TRAP —** A runaway agentic loop at $50/Mtok output is not a thing you want to meet on a bill. Set the limit while it is a number on a settings page instead of a number on an invoice.

WHAT EACH MOVE ACTUALLY SAVES.

Ranked by real impact on a real bill, not by how good the number looks in a benchmark table. If you only do three things, do the first three.

#	MOVE	REAL SAVING	EFFORT
04	**Pin subagents to Haiku/Sonnet** the most expensive default in the product	large	1 line
01	**Drop effort to medium** thinking tokens bill at $50/Mtok	large	1 command
06	**Do not break prompt caching** $10 → $1 per Mtok on repeated context	up to 90% on input	discipline
02	Plan on Fable, execute on Sonnet 5	~5× on execution	workflow
09	Batch API for anything async	exactly 50%	1 param
08	External state file + /clear	kills quadratic growth	workflow
05	Advisor strategy, capped	11.9% vs Opus solo 85% w/ Haiku exec	config
07	Prune MCP servers	per-turn overhead	5 min
10	Spend cap	$0 — it is insurance	2 min
03	Caveman / Ponytail	~4–5% (not 75%)	1 install

TWO THINGS NOBODY MENTIONS

Refusals are free. Fable 5 ships safety classifiers. When one declines, the Messages API returns `stop_reason: "refusal"` as a **200**, not an error — and you are not billed for a request refused before any output. Retry on Opus 4.8 and **fallback credit** refunds the prompt-cache cost of switching, so you do not pay it twice. Your code should check the stop reason instead of assuming every 200 is a billed answer.

Fable is a Covered Model. Mythos-class carries a 30-day data-retention requirement with no zero-data-retention option. If you hold ZDR agreements, raise it in procurement before you deploy Fable on sensitive data — or run hybrid: sensitive traffic on Opus/Sonnet under ZDR, everything else on Fable.

THE 15-MINUTE SETUP.

Everything above, collapsed into the order you should actually do it in. Fifteen minutes, once.

[1] Enable usage credits and **set a spend limit**. Web app only. Two minutes.

[2] Run `npx ccusage@latest`. Get your real baseline before you change anything.

[3] Run `/context`, then `/mcp`. Disable every server not in active use.

[4] Pin subagents: `model: haiku` in frontmatter, or `CLAUDE_CODE_SUBAGENT_MODEL` once.

[5] Run `/effort`. Set medium. Step up only when a task actually fails.

[6] Freeze tools + CLAUDE.md before long runs. Never add an MCP server mid-task.

[7] Route one async job to the Batch API this week. Bank the flat 50%.

[8] Re-run ccusage in seven days. Compare. Keep what moved the number.

THIS IS WHAT I DO EVERY WEEK AT RESEARCHAUDIO.IO

One AI research breakdown, every week, for engineers who want the sourced version instead of the thread version. Every claim traced to the paper or the doc it came from — same standard as this playbook.

→ **researchaudio.io**

Follow **@deeponai** for the short version. Send this to whoever on your team owns the bill.

Sources: Anthropic platform docs (effort, pricing, Fable 5 / Mythos 5 introduction, refusals & fallback), Claude Code docs (manage costs, model configuration), Anthropic's advisor strategy announcement and its published benchmark tables, Anthropic's Sonnet 5 launch post, and the Caveman repository's own benchmark disclosures. Rates and dates verified July 15, 2026 — Anthropic has moved this deadline twice, so check the live support page before you plan around it.