Kimi K3 vs the Frontier: A Private, Contamination-Resistant Coding Evaluation

Actual Intelligence Labs

17 July 2026

Abstract

Kimi K3 launched on 16 July 2026, and the reflex is to open a leaderboard and read off a rank. That rank says almost nothing about how a model handles *your* task, on *your* data, under *your* constraints. Following Chip Huyen's case for evaluation-driven development, we skip the leaderboard and run K3 head-to-head against Claude Fable 5 and GPT-5.6-Sol on nine hand-authored, contamination-resistant coding tasks, scoring quality, cost, and latency ourselves with a blind dual judge. Across 194 runs (40 for K3), all three models solve these tasks at a near-ceiling pass rate, so the story is efficiency, not correctness. K3 lands as the **cheapest per task at list price** ($0.14 vs Fable's $0.44 and Sol's $1.62), with judged quality in the middle of the pack (8.77/10). Along the way we quantify a cost driver leaderboards never show: a fixed ∼**30k-token "scaffold tax"** that an agent harness imposes on every model call, which dominates the bill on short tasks. We publish the method, the confounds, and the exact statistics behind every number.

1 Why not just read the leaderboard

Chip Huyen's argument for evaluation-driven development (EDD) starts from a property of foundation models that breaks classic testing: their outputs are probabilistic, not deterministic. The same prompt can produce different answers, so a binary pass/fail unit test stops being a clean signal; you need evaluation that scores a distribution of behavior rather than asserting one correct string.

Her critique of leaderboards follows from there. Public benchmarks such as MMLU are a "bird's-eye view": useful for a first-pass sanity check, useless as a final selection criterion. They suffer contamination (test items leak into training data) and overfitting (teams tune toward the benchmark). A leaderboard rank is a summary statistic over *someone else's* task distribution — not performance on your task, your data, or your constraints. Her verdict is blunt: both "vibe checks" and leaderboards fail as selection criteria, one because it is unsystematic and the other because it measures the wrong thing.

Her prescription is to define "good" first, then run a funnel: (1) a hard-constraint gate on license, deployment language, hardware, and privacy that removes models you cannot use regardless of quality; (2) public benchmarks for a cheap first cut; and (3) your own private, domain-specific evaluation, judged side-by-side on quality, cost, and latency, driven by data rather than impression. This paper is our step 3 for K3.[1]

2 The subject: Kimi K3

We attach a confidence flag to every external claim, because not trusting the vendor's numbers is the entire point of the exercise.

[1] Source: Huyen, *AI Engineering* (O'Reilly, 2025), evaluation chapters. Our phrasing of the "hard-constraint gate" is our framing of her argument, not a direct quotation.

Confirmed (Moonshot `/v1/models` probe + Simon Willison + MarkTechPost): released 16 July 2026, open weights promised by 27 July; a 2.8-trillion-parameter open Mixture-of-Experts using "Kimi Delta Attention"; model id `kimi-k3`; context length 1,048,576 (1M). Reasoning is *max-only* (`valid_efforts: ["max"]`, `supports_thinking_type: "only"`) — there is no effort ladder to sweep. **Pricing** (confirmed, Moonshot): input $0.30/M cache-hit, $3.00/M cache-miss; output $15.00/M. **Third-party** (Artificial Analysis via Willison, medium confidence): long-horizon Elo ∼1547, behind only Fable 5; $0.94/task; #1 on a frontend-code arena at launch. We deliberately do not lean on Moonshot's own benchmark table.

3 Method: the gauntlet

The evaluation is nine hand-authored, contamination-resistant tasks — novel repos we wrote, not public benchmark problems that could sit in a training set:

- **Tier 1** — a seeded bug with a failing test (4 tasks).
- **Tier 2** — a ticket-sized feature whose tests reference functions that do not yet exist (4 tasks).
- **Tier 3** — a greenfield CLI built from scratch against a hidden acceptance suite (1 task).

Languages are Python and TypeScript. "Done" is defined by us, not the model: we run the task's `verify.sh` ourselves and exit 0 means pass. The model never grades its own work. Per run we capture pass/fail, tokens in/out (plus a list-price estimate), wall-clock time, turns, and lines changed (an over-engineering proxy). Runs are interleaved (seed 1337) and executed one at a time with resumable JSONL state. "Harness on/off" is whether the model gets our seven-rule benchmark harness: repo=spec, done=command-output, WIP=1, state-in-files, checker≠worker, clean-exit, fix-the-harness-not-the-model.

Blind dual judge. Quality on passing runs is scored by two judges, Claude Opus 4.8 and GPT-5.6, each rating the diff 0–10 on four axes — correctness beyond the visible tests, simplicity/minimal-diff, idiomatic style, and spec-adherence — with no model identity in the judge prompt. Judging cross-vendor and blind neutralizes the self-grading objection.

How we ran K3, and the confound we disclose. K3 has no native agent CLI that fits our stack. Codex 0.144 dropped Chat-Completions wire support and is Responses-only, while Moonshot's endpoint returns 404 on `/v1/responses` (we verified this). So we drove K3 through **Claude Code pointed at Moonshot's Anthropic-compatible endpoint** (base URL `api.moonshot.ai/anthropic`, model `kimi-k3`). The upside is that K3 and Fable then run inside the *same* agent scaffold, so the Fable-vs-K3 comparison isolates the model cleanly. The cost: GPT-5.6-Sol runs inside Codex, a different scaffold. This is a "how well does each model do the job inside a capable agent" study, not a pure model-in-a-vacuum study. We flag it rather than hide it.

Cost definition. Unless stated otherwise, *cost per task* is the mean over *every* run of that model in our matrix (all efforts, bare and harnessed), priced at published list rates: Fable and Opus $15/$75 per M in/out, Sol $10/$40, K3 $3/$15 (cache-miss). Fable and Sol run on a subscription, so their dollar figures are estimates; only K3 is billed for real. Cache-hit input for K3 is $0.30/M, so its real bill trends below the figure we report.

4 Results

Across 194 runs (193 passing, 193 judged), all three models clear these tasks at a near-ceiling pass rate (Table 1). K3 passed 39 of 40; its single miss (`t2-py-a`, rep 1) was a CLI transport error — zero tokens, wall timeout — not a wrong answer, and its other two reps both passed. Pass rate is therefore a near-constant here; the interesting variation is in cost, quality, and latency.

model	effort	n	pass	pass rate	avg. quality /10
Fable 5	medium	40	40	100%	8.42
Fable 5	high	24	24	100%	8.99
Kimi K3	**max**	**40**	**39**	**98%**	**8.77**
GPT-5.6-Sol	low	40	40	100%	9.04
GPT-5.6-Sol	medium	24	24	100%	9.26
GPT-5.6-Sol	high	24	24	100%	9.36

Table 1: Pass rate and blind-judge quality by model and effort. A T3-only "hybrid" arm ($n = 2$) is omitted here.

4.1 The cost–quality frontier

Figure 1 is the headline. K3 is the cheapest model on test by a wide margin — \$0.14/task against Fable's \$0.44 and Sol's \$1.62 — while its judged quality (8.77/10) sits squarely between the other two. The result is counterintuitive on tokens alone: K3 spends *roughly twice* Fable's tokens per solved task (39k vs 21k, Fig. 2), yet costs about a third as much, because Moonshot's per-token price is roughly 5× below the Opus/GPT tier. Token count is not the bill; price × tokens is, and K3 wins that product.

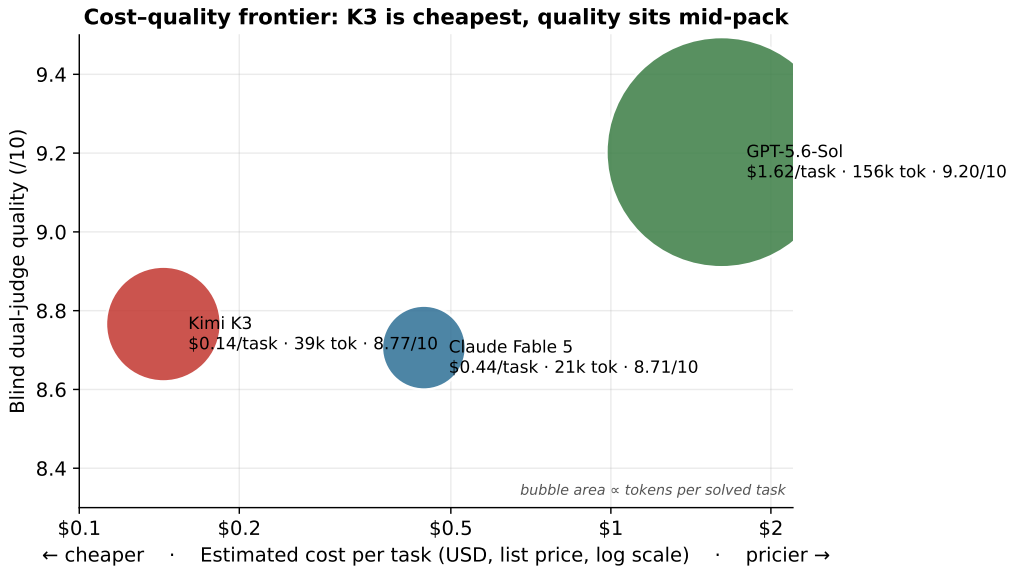

Figure 1: Cost–quality frontier. Horizontal axis is estimated list-price cost per task on a log scale (cheaper to the left); vertical axis is blind dual-judge quality. Bubble area is proportional to tokens per solved task. K3 occupies the cheap corner; Sol buys ∼0.4 quality points at ∼11× the cost and ∼4× the tokens.

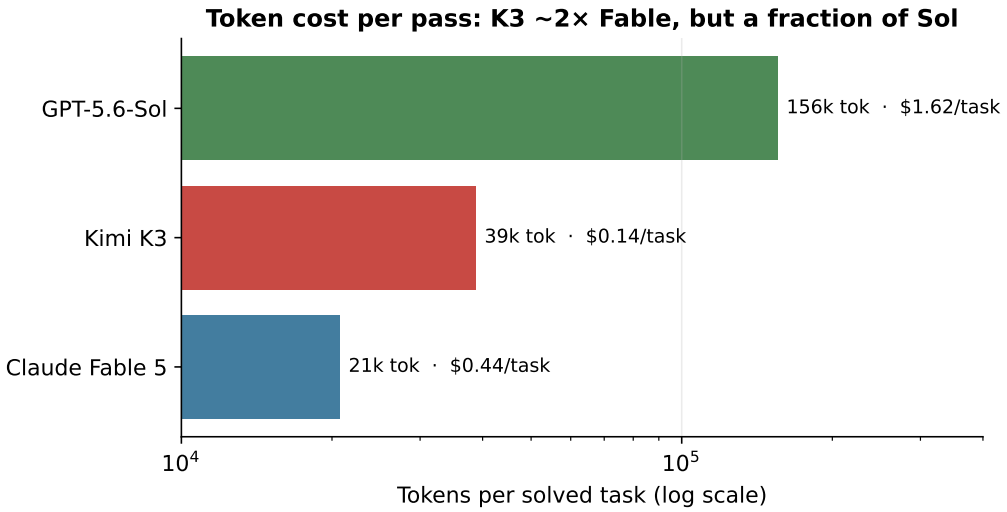

Figure 2: Tokens per solved task (log scale). K3 spends about twice Fable's tokens and roughly a quarter of Sol's. Dollar figures are list-price estimates under the cost definition in §3.

4.2 Quality by axis

Broken out by judge axis (Fig. 3), K3 slots between Fable and Sol on every dimension — edging Fable on correctness (8.82 vs 8.74) and idiomatic style (8.73 vs 8.47), trailing Sol across the board (Sol leads on spec-adherence at 9.48). All three cluster inside about 0.7 points. Given that the two judges themselves agree only to within ±0.56 points on average (§6), these axis gaps are directional, not decisive.

4.3 Latency and the harness delta

K3 is the slowest model on wall-clock (median 89s vs Fable's 64s and Sol's 54s), which is expected: it is max-only, with no lower-effort setting to trade down, and Claude Code capped its usable context at 200k rather than the advertised 1M. The harness helped K3 the most in relative terms — it closed K3's one bare-arm failure (bare 96% → harnessed 100%) for a modest +6k tokens per run — while imposing no pass-rate change but a steep +72k-token cost on Sol. The K3 effect rests on a single discordant pair (McNemar exact $p = 1.00$): suggestive, not significant.

5 The input floor, in plain terms

The single most transferable finding has nothing to do with which model "won." It is that *per-task cost is a property of your harness as much as of the model's sticker price*, and we can put a number on it.

Every time you hand a coding model a task through an agent like Claude Code, the agent prepends a large fixed preamble before the model sees a single line of your code: the system prompt (who the agent is, its rules) plus the full JSON schema for every tool the model may call — read a file, edit a file, run a shell command, and so on. In our setup that preamble alone is about 29–30 thousand tokens. We measured it directly: asking K3 to reply with the single word "OK" cost **29,091 input tokens**.

Think of it as a cover charge at the door. You pay it to get in, whether you order a one-line bug fix or a from-scratch CLI. Figure 4 shows it: nearly every real K3 call lands at 30–43k input

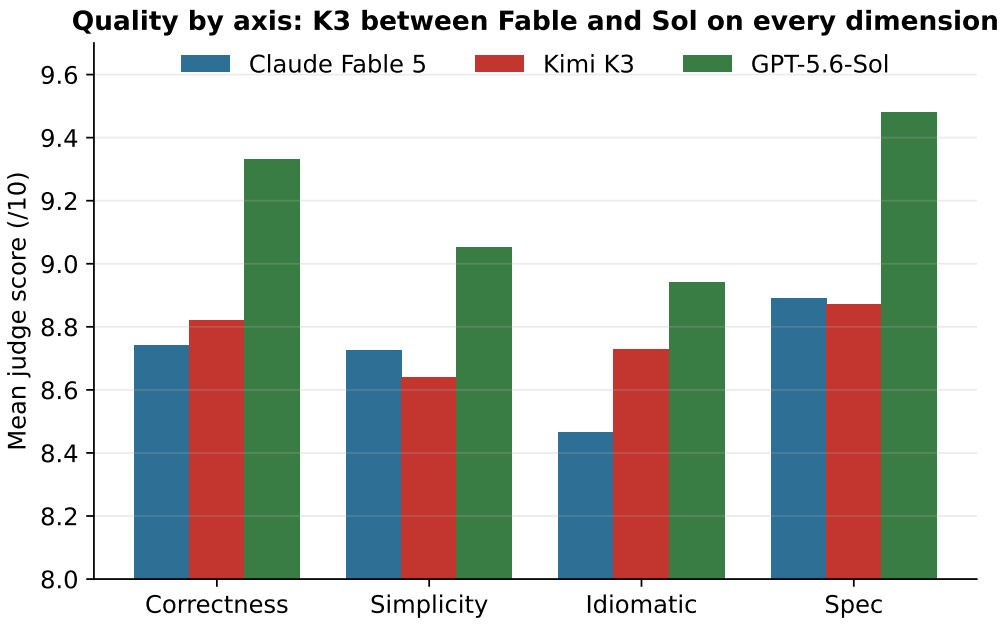

Figure 3: Mean blind-judge score by axis, pooled across both judges. K3 is a consistent middle: never worst, never best.

tokens (median 38k), stacked on top of that ∼29k floor, no matter how little the model writes back (the horizontal axis). Fable, running through the same Claude Code scaffold, sits far below — a reminder that the floor is partly the scaffold and partly how much history each model carries forward. (Two K3 runs dip below 15k: those are heavy cache-hit calls, where Moonshot served most of the prompt from cache and billed less. They are the exception, and we keep them in the plot rather than trim them.)

Two practical consequences follow. First, on *short* tasks — a one-line fix where the model writes a few hundred output tokens — that fixed preamble is most of the input bill. The model's efficiency barely matters; the harness does. On a large task the same overhead is rounding error. Second, this is exactly the quantity a leaderboard cannot show you, because a leaderboard never runs the model inside your real agent with your real tools. The same K3 that looks cheap here would cost more under a heavier scaffold and less under a leaner one — and its advertised 1M context was silently clipped to 200k by the harness, not by the model. If you are selecting a model to deploy, measure it inside the agent you will ship.

6 Statistical rigor

Every test below is exact (Wilson intervals, McNemar exact, sign-flip permutation), rendered with raw counts rather than a bare p-value.

- **The cost gap is real.** A sign-flip permutation test on per-task median log-tokens (Fable minus Sol, paired by task) gives all nine tasks favoring Fable, two-sided exact $p = 0.0039$. K3 sits below both on list-price cost per task.
- **The judges are trustworthy.** Over 178 dual-judged runs, Claude and GPT-5.6 correlate at Pearson $r = 0.69$ with a mean absolute gap of 0.56 points on the 0–10 scale; **94% of runs agree within ±1 point**.

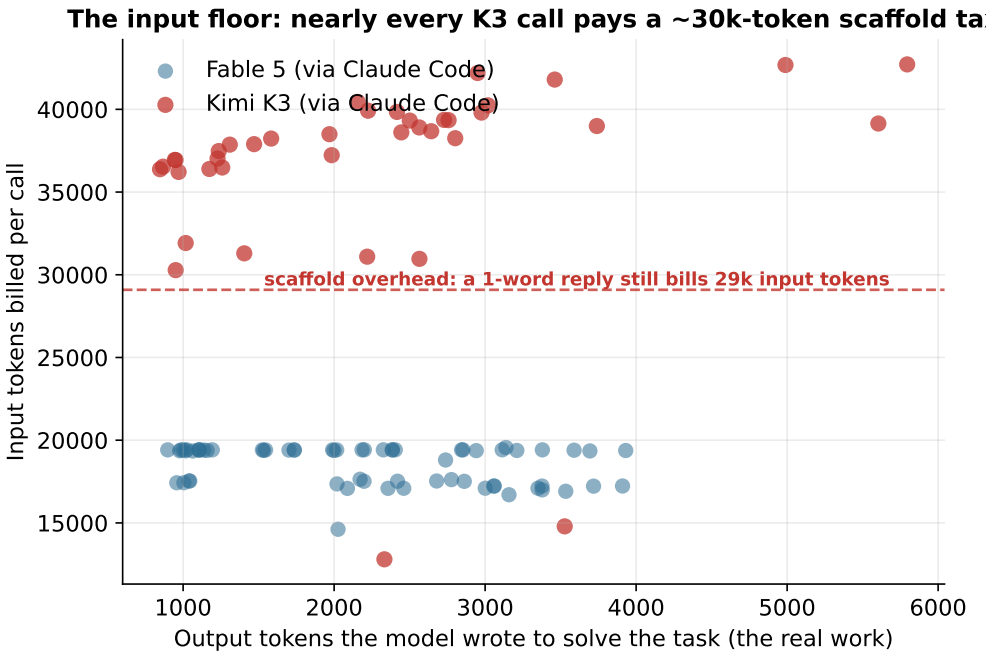

Figure 4: The input floor. Each point is one passing run: horizontal position is the output tokens the model wrote to solve the task, vertical is the input tokens billed for that call. Nearly all K3 calls sit well above the measured ∼29k scaffold-overhead line, largely independent of how much output the task required.

- **We are honest about power.** With per-arm $n \approx 24$, the design detects only pass-rate gaps of ≈ 37 percentage points at 80% power. Every model here passes near 100%, so we say "no detectable pass-rate difference," never "they are equal."

7 Limitations and threats to validity

- **Scaffold confound.** Fable and K3 run in Claude Code; Sol runs in Codex. Cross-scaffold comparisons (including turn counts, where Codex reports differently) are representative of real deployment, not pure model isolation. Fable-vs-K3 is clean; anything involving Sol carries the scaffold caveat.
- **K3 is max-only.** There is no effort ladder to sweep for it, unlike the others, so its latency and cost reflect its single available setting.
- **Small** n, especially Tier 3.
- **Effort mix.** Our pooled cost figure averages over each model's available efforts, which differ (Sol spans low/medium/high; K3 is max-only). The K3-cheapest and Sol-most-expensive ordering is robust to this, but the exact Fable/Sol ratio shifts with the effort mix you assume.
- **List-price dollars are estimates.** Subscription models are billed differently; only K3 is billed for real.
- **Contamination** cannot be fully disproven, though the tasks are novel repos we authored.
- **Judge bias.** The two judges are frontier models with their own biases, even blinded.
- **Statistics scope.** The pairwise McNemar and permutation tests compare only Fable vs Sol; K3 appears in every descriptive result but was not retrofitted into those two exact tests.

8 Conclusion

On our nine contamination-resistant coding tasks, Kimi K3 is strong and cheap: it solves what the frontier solves, at judged quality between Fable 5 and GPT-5.6-Sol, for the lowest list-price cost per task of the three, because its per-token price more than offsets its heavier token appetite. Whether it is the right choice for a given deployment depends on latency tolerance (it is the slowest), on the confounds above, and on the most portable lesson here: the harness you run a model inside can move its per-task cost more than the choice of model does. Measure the model in the agent you will ship, define "good" yourself, and treat the leaderboard as a first cut rather than a verdict.

Data and method. Runner, tasks, and JSONL results live in our `model-gauntlet` repository; figures regenerate from the run JSONL via `render_figs.py`. External numbers carry confidence flags; unverified figures are excluded from claims. Sources: Huyen, *AI Engineering* (O'Reilly 2025); Kimi K3 — Moonshot platform docs, a live `/v1/models` probe, Simon Willison (2026-07-16), and MarkTechPost; Artificial Analysis third-party Elo/cost (medium confidence).