AI Infrastructure and SRE: Running Models in Production

Julien Laurenceau | Published 2026-09-30 | Updated 2026-10-22

https://pepitedata.com/ai-infrastructure-and-sre/

PepiteData | LinkedIn: Julien Laurenceau | X: @Julien_Lau



AI has entered the operations loop from both sides. Models take production responsibilities (agents with tool access, pipelines with write paths), and production data flows through models (prompts, documents, code, telemetry). The resulting surface is not just a model: it is a model, a router, a provider, a terms-of-service document and a behavior that changes over time without a deploy. This guide maps how to run that surface like infrastructure, with one shared metric: **cost per correctly solved task**. It collects what the AI posts on this blog established across model economics, local inference and production control; each section states the mechanism and links to the post that proves it.

The one-line version: **efficiency is not raw power, quiet failure is the default, and verification is the method, not a step**. The guide has two sub-fils: model economics (when the cheaper model is the right model) and SRE practice (the AI supply chain as a production dependency).

Two movements, one surface

Start from the dependency map, not the leaderboard. When an AI system holds a production role, every hop in the chain can fail quietly: the router rewrites your request, the provider swaps weights behind a stable name, the terms silently change what happens to your data, and the model's output was never a stable contract to begin with.

That chain is what an audit has to cover, and it is what auditing the AI supply chain like a production dependency treats in full: inventory and provenance, data path, tool-call control, output contracts, behavior verification, failure paths and pipeline sizing.

The metric that makes the two sub-fils one conversation is cost per correctly solved task: not price per token, not peak score, but what a unit of actually-finished work costs. In a pipeline the same idea becomes cost per million records, concurrency arithmetic and replay cost. Once the metric is named, model choice and operations review stop being separate meetings.



Review every hop: data handling, routing, provider terms, model versions and behavior. Audit scope includes both the request and response paths.

Model economics: when the cheaper model is the right model

The frontier premium buys exploration and time, not tokens. A leader optimizes toward an absolute target and must explore: architecture, data mix, recipe, reward design. A challenger optimizes a distance to a published target and can exploit, kill bad runs early, and distill at roughly an order of magnitude less compute for a comparable score on the narrow tasks that matter.

Why leading costs ten times more is the cost asymmetry, and what actually works in distillation is the engineering path that follows from it: execution-verified traces, a narrow verifiable domain, and three named traps (you cannot copy hidden reasoning, the teacher is a single-source vendor, and a distilled model ships without the safety behavior).

Arrangement matters as much as choice. A panel of models with a judge-synthesizer can beat the best single model on deep research, and a budget panel can land within a point of the top frontier at half the cost per task, with the caveat that judge selection swings absolute scores heavily. The cost case for model fusion is the measurement. And when comparing single models, separate sharply on cost per correctly solved task rather than peak scores: a previous-generation configuration can beat a flagship entry tier. Efficiency is not raw power reads the leaderboard that way. Distillation mechanics themselves (soft labels, feature matching, inherited habits) live in how big models teach small models.

SRE practice: the AI supply chain is a production dependency

Treat a model like a database: vendor, version, failure mode, bill. The operational conditions around any model choice are three. First, evaluation is a gate with a holdout: quality, agent reliability, operational cost and data-provider risk, measured on your own workload with fixed acceptance rules, because a demo passes on vibes and fails in the harness. Testing a free model before production is the worked protocol.

Second, the serving regime changed: streaming moved the capacity-planning unit from request latency to time-to-first-token, token throughput and concurrent streams, and the headline vendor number is a peak while the production panel is the reality to size against. Real-time AI streaming is a product trend, not a benchmark. Third, the dependency itself moves: providers update weights, filters and prompts without changelogs, so version discipline, resolved model IDs recorded per row and behavior alarms are part of the runbook.

Quiet failure: what changes when the component is a model

The failure taxonomy of an AI dependency is drift and contract violation, not crash. The model name stays the same and the weights change. The refusal rate collapses while the benchmark score holds. The output violates the JSON contract it honored yesterday.

A distilled model inherits traits nobody taught. A judge swings the evaluation 10 to 25 points. None of these raise a page. The production requirement follows directly: pin and record versions, keep a behavioral holdout that runs on a schedule, alarm on deviation from a rolling baseline (a change of 15 points is a real event), and treat every model upgrade as a migration with replay against the holdout before cutover.



The endpoint kept its name. The behavior changed. Replay a fixed holdout and monitor deviations when evaluating a provider update. Conceptual illustration.

Verification and judgment: the gate is the method

Across every post in this set, the same discipline appears under different names. Distillation works when every training trace passes deterministic tests before it counts (the execution gate is the method, not a filter). Model panels are judged against fixed criteria with contamination checks. A free or anonymous

provider is admitted only through a paired holdout with pre-agreed acceptance rules and manual inspection of failures.

Production models ship behind a canary, a fixed evaluation set and a drift alarm. And judgment work keeps a second opinion the team actually owns, because outsourced judgment inherits the vendor's biases along with its answers. The human or the owned system still decides three things: what counts as solved, when behavior has changed, and what the exit path is.

Local or frontier: owning the critical path

Local inference is an operations decision before it is an ideology. You own weight control, offline capability, data boundaries and stable behavior; you pay in memory and engineering time. The real blocker is memory, not quality: a modest quantized model plus its key-value cache at long context can exceed the weights themselves, which is the arithmetic the local LLM case works through while making the judgment-work argument (keep an uncensored second opinion you actually own).

Batch workloads are genuinely cheap on a consumer GPU: transcribing a hundred hours of podcasts in an evening is one measured point, at about thirty-three times real time, with the same idempotence and provenance discipline a platform pipeline needs.

The decision rule: put the differentiating judgment work where behavior is stable and inspectable, rent the commodity frontier for narrow verifiable high-volume tasks while its economics hold, and revisit when the rented part changes (price, terms, weights, availability). One policy for both is the expensive mistake.

The production checklist

What to hold before an AI dependency carries a production role: a four-part evaluation (task quality, agent reliability, operational cost, data-provider risk) on a paired holdout with acceptance rules; a dependency review covering router and provider data paths, retention, terms and exit options; a fail-closed policy gate at the tool-call boundary. Output contracts validated in the consuming harness; version discipline with the resolved model recorded per row.

Behavior alarms against a rolling baseline; pipeline arithmetic (records to calls to tokens, cost per million records, concurrency, replay cost). And a sizing plan built on the production panel (P50 and P90 of time-to-first-token and token throughput), not the vendor peak. When a platform decision rests on these numbers, a data infrastructure audit or a focused Expert Call is where I pressure-test them.

FAQ

What is cost per correctly solved task?

It is the total cost of producing one unit of work that passes your acceptance tests: tokens, retries, judge calls and failures included, divided by the tasks actually solved. It is the right metric for production model choice because it separates efficiency from raw capability: a cheaper model that solves more of your tasks per euro is the better system, whatever the leaderboard says.

Why not just use the biggest model?

Because the frontier premium pays for exploration and time advantage, and your workload rarely needs either. On narrow, verifiable, high-volume tasks a distilled local model or a budget panel can match frontier quality at a fraction of the cost per solved task. Keep the frontier where the task is differentiating, ambiguous or safety-critical.

What is an AI supply chain audit?

It is the same review you would run on any critical dependency, pointed at the model chain: where prompts and data go (router, provider, retention, terms), which model version actually serves each row, what the tool-call layer may do (with a fail-closed gate), whether output contracts hold in the harness, how behavior is alarmed, and what the exit options are if a provider changes or disappears.

When is a local model the right model?

When you need behavior stability, data control or an uncensored second opinion more than you need frontier breadth, and when the workload is narrow and verifiable enough to distill or fine-tune. Check the memory arithmetic first: key-value cache at your context length can exceed the weights. For batch processing of owned data, local inference on a consumer GPU is often simply cheaper.

Related posts

- After Coldcard: access dependencies in AI-assisted security review
- Your AI supply chain is a production dependency. Audit it like one.
- Efficiency is not raw power: meet GPT-6-Astra and the DeepSWE reality
- Model fusion beats the frontier model: the cost case
- Distilling frontier reasoning into a local model: what actually works
- How big models teach small models, and why it looks like school
- The frontier model has no teacher: why leading costs ten times more
- Local LLMs, political bias, and why llama.cpp still matters
- Transcribe 100 hours of podcasts with whisper.cpp
- Ox Alpha review: test a free AI model before production
- Real-time AI streaming is a product trend, not a benchmark

If an AI dependency is about to carry a real workload, or already shows drift you cannot explain, the data infrastructure audit covers the reliability and cost sides together. A focused Expert Call is the usual starting point.

PDF snapshot of the public article. The online version is authoritative; links remain clickable.