Choosing Models and Providers

There Is No Best Model: why selection is a routing decision - and why the choosing capability matters more than the choice

LinkedIn and kjerside.com companion essay | Companion 3 of 3 | 29 July 2026

Every buying conversation about AI eventually arrives at the same question: which model should we use? It sounds like a procurement question with a procurement-shaped answer - evaluate, pick the winner, sign. The question assumes a ranking. The ranking does not exist.

Leaderboards certainly exist. But a benchmark measures specified tasks under specified conditions at a moment that has already passed. It says little about your documents, languages, latency, data constraints or cost per accepted outcome.

There is a quieter and more expensive problem. Foundation-model markets move faster than planning cycles: names, prices and benchmark positions change within weeks, while the decisions that bind an organisation persist for years. A process built around finding the current winner is built around the most perishable fact in the stack.

The first companion separated AI into capabilities. The second followed the path from model to governed system. This one asks how such a system should choose its changing inputs. The answer is not a permanent vendor decision. It is a standing capability to constrain, evaluate, route, monitor and exit.

There is no best model. There are only best fits.

The model choice is temporary. The choosing capability is permanent.

> *The model choice is temporary. The choosing capability is permanent.*

1. Why the ranking intuition fails

Models differ along many axes at once: architecture, modality, context span, reasoning-time compute, openness, deployment path and configurability. A single ranking requires collapsing those axes into one score, and the weights used to collapse them are themselves the decision. Best is not a property of the model. It is a property of the match.

Two models can each be best: one for a multilingual diligence memo, another for classifying a million documents overnight. A benchmark that averages across those tasks produces a number that describes neither. HELM makes the point formally: evaluation is scenario-specific and multi-metric, and rankings change with the tasks and desiderata included.[1]

Providers differ at a higher level too. The market is better read as a set of architectures: an integrated frontier and agent platform, a model-first multi-

cloud platform, a hyperscaler-native ecosystem, a European hybrid/open-weight platform and a sovereign domain stack. The practical difference is how much of the system each asks the customer to outsource. A dated snapshot is in Appendix A.

No composite score solves the problem. Sovereignty versus cloud integration, managed convenience versus exit, peak quality versus unit economics: those weights are the decision. The correct weights for a marketing assistant and a defence supplier are not different preferences over the same question. They are different questions.

2. Separate the five things you are actually buying

The word model collapses five separate purchase decisions: the model configuration, the delivery channel, the provider platform, the operating envelope and the enterprise control plane. Each layer can be outsourced or kept, and each choice moves data, cost, risk and control differently.

The reproducible unit of evaluation is the model configuration; the launch name alone is not a test condition. And the same named model may arrive through different delivery channels. The channel is not plumbing: it can change identity, region, processor chain, retention, price and deprecation clock.

Decision layer	What it includes	Why it changes the purchase
Model configuration	Family, tier, pinned snapshot, reasoning effort, tools and instructions.	The reproducible unit of evaluation, pricing and change control.
Delivery channel	First-party API, hyperscaler endpoint, SaaS application or dedicated/private runtime.	Changes identity, region, processor chain, retention, SLA, price and lifecycle.
Provider platform	State and memory, files and retrieval, connectors, tool execution, agents, evaluations, traces and guardrails.	Saves integration effort, while concentrating data, workflow semantics and lock-in.
Operating envelope	Contracting entity, DPA, storage and inference location, retention, support access and deprecation.	Defines legal exposure, continuity and what can actually be verified.
Enterprise control plane	Identity, classification, routing, budgets, approvals, evidence and incident response.	The durable layer that should remain customer-owned and provider-independent.

Every layer outsourced saves integration effort. It also moves data, operating knowledge and workflow semantics into provider control. That is the first major reframing: a provider decision is not a comparison of engines. It is a decision about where the boundary sits between supplier capability and institutional control.

You are not choosing a model. You are choosing an outsourcing boundary.

3. Set constraints before comparing capability

> *You are not choosing a model. You are choosing an outsourcing boundary.*

Before any capability comparison, define the envelope in which the work is allowed to run: consequence of error, data sensitivity, legal restrictions, geography, latency, continuity, volume and the authority the system may exercise. A provider that cannot satisfy the envelope is not second-best. It is out of scope.

Data residency is necessary but not sufficient. A European region setting may govern storage or inference location; it does not by itself resolve the contracting entity, controller-processor roles, subprocessors, support access, failover or governing jurisdiction. Residency has two dimensions - storage and inference - and contracts should specify both.

Zero retention is likewise feature-specific: a model call, stored conversation, vector store, cache, connector and support log can each follow a different policy. Contractual confidentiality and legal privilege are related but not identical; high-stakes professional use needs legal approval as well as technical control.

Self-hosting changes the location of responsibility, not its existence. Open weights move provenance, licence compliance, serving, patching, security and incident response inward, and at modest volumes hosted open-weight endpoints may still beat self-run hardware once idle capacity and operations are counted. Local deployment should be justified by sovereignty, latency, economics or offline requirements - not by the belief that local automatically means secure.

A European region is an address, not a guarantee.

4. Evaluate configurations on your work

Only after the constraints are set should capability be compared, and the comparison must run on the organisation's own work, not on public benchmarks. The unit under test is a pinned configuration - snapshot, endpoint, reasoning budget, instructions and tools - because a family name alone cannot be reproduced when the provider changes the serving layer.

Generic benchmarks can screen candidates; they cannot certify a workflow. Quality needs a threshold, not a beauty contest: the objective is the least expensive approved configuration that reliably clears the workload's quality bar. And price per token is often the wrong denominator; a cheap model that retries more or needs more correction can cost more per accepted outcome.

Question	What to test
Does it produce an acceptable result?	Representative tasks and difficult edge cases.

Does it fail safely?	Missing evidence, conflicting instructions, tool failure and out-of-scope requests.
Can the result be reproduced?	Exact configuration, inputs, instructions, tools and output schema.
What does success cost?	Cost per accepted outcome, including retries, excess context and correction.
What does error mean?	Consequence-weighted scoring rather than one average error rate.

A small error in classification and a small error in an irreversible transaction are not the same event. The scoring system must reflect what failure means, not merely how often it occurs.

Evaluate the configuration. Price the accepted outcome. Weight the consequence.

5. Build a portfolio and route by workload

The durable production pattern is a portfolio, not a default provider. Different workload classes have different binding constraints, so work is routed: consequence and sensitivity first, then the least expensive approved configuration that clears the quality threshold.

Managed and open-weight are not opposites - open-weight models can be consumed as managed services, and frontier models can orchestrate smaller models as tools, while deterministic software validates the things that need no language model at all. The question is not which family wins, but how much of each the portfolio needs.

Workload class	What decides	Typical destination
Complex synthesis, challenge and long-horizon analysis	Consequence and quality dominate cost.	Frontier managed models, with evidence, challenge and review attached.
High-volume narrow tasks: classification, extraction, first-pass triage	Unit economics, once the quality threshold is met.	Open-weight models, hosted or self-hosted.
Confidential, privileged or regulated content	Data sensitivity, processing location and retention.	Approved regional or isolated endpoint, or local open-weight deployment.
Interactive, latency-critical steps	Responsiveness at acceptable quality.	Fast, lower-cost model tiers or local models.
Offline or fully controlled environments	Connectivity and control requirements.	Self-hosted open-weight models.

Routing research has shown, under benchmark conditions, that choosing dynamically between stronger and weaker models can improve the cost-quality trade-off.[2] That evidence does not prove a universal enterprise policy; it proves the mechanism is real. The production policy still has to be learned from the organisation's own workload.

fastland consult implements this logic as product architecture. *fastland* builds and owns the application and platform. *taonta* uses it to run consulting engagements. The client selects the permitted envelope - a declared provider, EU-constrained or local path where supported - and the application routes within that envelope, records the choice and carries the evidence forward. The product is not loyal to a lab; the consulting firm remains accountable for the engagement.

The house blend will change as capability, economics and regulation change. That is the point. The durable feature is not today's list of models. It is the ability to apply constraints, compare configurations, route work and explain what ran where.

Route by consequence, not by logo.

6. Keep the control plane provider-independent

Provider-native state, tools, agent runtimes, evaluations and traces reduce time to production, but they move lock-in upward from API syntax into stored state and workflow semantics. The enterprise control plane - identity, classification, routing policy, budgets, approvals, evaluation records, evidence and release decisions - should remain customer-owned.

Provider platforms can sit beneath that control plane; they should not become the only place where the organisation knows how its work runs.

Tool access deserves the same discipline. Every tool a model may call is a privileged integration. Credentials should be scoped to the smallest action set, irreversible steps should require approval, and rollback should be designed before authority is granted. Natural-language instructions are not security boundaries.

Portability has three levels. Interface portability means another endpoint can be called. Operational portability means identity, logging, policy and incident controls still work. Semantic portability means the destination model produces an acceptable result on realistic evaluations at realistic volume. Only the third makes exit commercially real.

A common API is not an exit plan. A tested workload is.

7. Make choosing a continuous capability

> *A common API is not an exit plan. A tested workload is.*

Choosing does not end at procurement, because providers version, replace and retire models on their own schedules.[3-5] A production system needs pinned configurations, regression evaluations, release gates, fallback paths and a defined deprecation process: a deadline for replacing a deprecated configuration before the provider sets it for you.

Evaluation is the foundation. Not a one-off bake-off, but a standing capability: representative tasks, consequence-weighted scoring, regression sets that run whenever a model, prompt, tool or policy changes, and release gates before production. NIST's work on deployed AI systems reinforces the lifecycle point: pre-release evaluations cannot capture every behaviour that appears in real use.[6]

Evaluation converts routing from opinion into a system. It is also what makes exit real. A firm that can measure a workload on a second model can move it. A firm that cannot measure has a preference, not an option.

Four market movements reinforce this architecture: capability is commoditising unevenly, so reasoning becomes a budget; providers are moving up the stack, so lock-in moves into state and workflow; agentic capability shifts risk from answers to actions, so authority becomes a design variable; and hybrid portfolios become normal, so portability must be proven, not asserted. Appendix A records the provider snapshot; Appendix B keeps the questions that survive it.

None of the durable assets depreciates when a new model ships. The model choice loses value the day it is made; the choosing capability compounds. The problem with "which model should we use?" is not that models are interchangeable - they differ along every axis - but that it asks for a permanent answer to a perishable fact. The better question: which configuration fits this workload, inside these constraints, at this consequence - and can the answer be revisited when the facts move?

AI is not one thing. The model is not the system. And there is no best model - only a system that keeps choosing well.

Appendix A: Provider architectures - snapshot as of 12 July 2026

The core essay argues that provider choice is an architecture and outsourcing-boundary decision, not a model ranking. This appendix records five provider architectures as they stood on 12 July 2026, based on official provider documentation. It is deliberately the fastest-dating page in this corpus. Positions and constraints should be revalidated before any procurement decision, and provider statements are treated as claims about their own products rather than independent performance verification.

Provider and architecture	What the provider tends to bundle	Where dependence or burden sits
OpenAI - integrated frontier and agent platform	Managed model tiers plus provider-native tools, state, orchestration and multi-agent execution; strong fit for fast-moving knowledge work.	Dependency concentrates in native state, tools, traces and a rapidly evolving managed-service control matrix.
Anthropic - model-first, multi-cloud platform	Pinned model snapshots and agentic capability through Anthropic and major cloud control planes; unusually broad channel choice.	Region, retention, identity, pricing and lifecycle can differ by channel; closed weights.
Google - hyperscaler-native AI ecosystem	Models beside enterprise data, search, IAM, agents, sessions, memory, evaluation and observability.	Breadth creates configuration and processor-surface complexity; dependency can extend across cloud services.
Mistral AI - European hybrid and open-weight platform	Managed models and Studio plus open-weight, dedicated, hybrid and self-hosted paths.	More deployment optionality, but more customer responsibility for licences, serving and operations as control increases.
Aleph Alpha - sovereign domain stack	An end-to-end sovereign suite integrating open and proprietary models, applications, development and operating layers.	A more bespoke programme and smaller public ecosystem; value depends on institutional and domain fit.

The products in the table will change. The architectures will change more slowly. The durable question is how much model, state, tooling, deployment and governance each provider asks the customer to outsource.

Appendix B: The durable checklist

The facts change; the questions survive. This checklist turns the essay's argument into a quarterly control cycle and extends the model-level controls in the second companion toward deployment, governance and exit.

Question	Why it matters
Which family and tier clears the quality threshold for each workload now?	Capability and price move quarterly; the thresholds should not move with them.
What does each model's licence actually permit?	Open is a spectrum; production caps, use clauses and jurisdiction terms are legal questions.
Which snapshot, endpoint and reasoning configuration is each workload pinned to?	A model name is not a reproducible test condition; the configuration is what evaluation certifies.

Which parts of the system are provider-owned: model only, or state, retrieval, tools, orchestration, evaluations and logs?	Integration savings and semantic lock-in are created in the same places.
Which legal entity signs the DPA, and which transfer mechanism covers each endpoint?	Article 28 applies regardless of geography; Chapter V machinery applies outside the EEA - and both must be kept current.
Where is data stored, and where is inference performed?	Residency has two dimensions, plus failover; contracts should specify all of it.
What is retained, by which endpoint, feature and tool?	Zero retention is feature-specific; grounding, state and logging paths create separate records.
Who can access content, including support staff and subprocessors?	A region setting does not resolve the processor chain or access exposure.
What may the system do without a human, and what evidence is kept?	Agentic capability moves risk from answers to actions; authority should narrow as consequence rises.
Has each critical workload actually been moved to a second model, and at what cost?	Portability that has not been tested is a hope, not an exit option.
Does self-hosting still pay at current volumes and prices?	Idle capacity and operations often reverse the apparent economics.

Selected evidence and references

1. Liang et al., Holistic Evaluation of Language Models, Stanford Center for Research on Foundation Models, 2022; HELM is maintained as a living evaluation framework.

2. Ong et al., RouteLLM: Learning to Route LLMs with Preference Data, 2024.

3. OpenAI, API deprecations documentation, accessed 24 July 2026.

4. Anthropic, model deprecations documentation, accessed 24 July 2026.

5. Google Cloud, model versions and lifecycle documentation, accessed 24 July 2026.

6. Rao et al., NIST, Challenges to the Monitoring of Deployed AI Systems, NIST AI 800-4, March 2026.

7. Regulation (EU) 2024/1689 (the EU AI Act): obligations follow the intended purpose and use of the system, not the model brand. Regulation (EU) 2016/679 (GDPR), Articles 25 and 28 and Chapter V (Articles 44-49): data protection by design, the controller-processor relationship and the international-transfer machinery underlying section 3.

8. Official provider documentation for Appendix A: OpenAI, Anthropic, Google Cloud, Mistral and Aleph Alpha product and platform documentation, accessed

12 July 2026. The appendix synthesises those sources for architecture comparison. Provider statements are claims about their own products, not independent performance verification, and the comparison does not certify provider performance, availability or contractual guarantees.

Evidence and inference. HELM supports scenario- and metric-specific evaluation; RouteLLM demonstrates a cost-quality routing mechanism under benchmark conditions; official provider pages document active versioning and retirement; and NIST supports post-deployment monitoring. The conclusions about standing routing, tested fallback and provider exit are the author's operating inference from that evidence, not claims made by the providers.

Disclosure

This article was AI-assisted. The argument, judgement and conclusions are my own.

Jesper Kjerside is co-founder of *taonta*, an AI-native advisory firm, and *fastland*, the software company behind it.