

The local AI workstation

Frontier cloud models and controlled local models are complements. Use scarce frontier reasoning to compile specifications and resolve ambiguity; use open-weight models on owned or rented hardware for bounded engineering roles, with harnesses and CI/CD gates deciding what survives.

AUTHORED BY

Daniel Johnson & Gary McLean Hall

AI DRAFTING

Astra, Fable & the models of the moment

EVIDENCE CHECKED

15 September 2026. The date tells you when the cited material was reviewed, not when the future agreed to behave.

INVALIDATED BY

When capability, licensing, serving economics or provider controls materially change.

EXPIRY DATE

The reader's call. Before buying a heroic GPU to solve an undefined role.

The founders own the argument; AI helped draft the prose. The sources and limits are here so you can challenge both.

Contents

01. One model is not an architecture
 02. Open, open-weight and closed
 03. The capability gap moves
 04. Frontier models for specification
 05. Controlled models for engineering roles
 06. Commodity hardware and quantisation
 07. Managed boundaries such as Bedrock
 08. The role system
 09. Harnesses provide control
 10. CI/CD is the constitutional layer
 11. Security and operations
 12. Economics
 13. A reference architecture
 14. Limits
 15. Conclusion
 16. References
-

This is a reference architecture and evaluation proposal, not a workstation benchmark or hardware purchasing recommendation. The review date records source and editorial checks. Provider controls and licences require verification for the exact configuration selected; the examples below do not establish that a particular local model meets a production role.

One model is not an architecture

The practical question is not whether open or closed models will win. It is which role deserves which capability, control boundary, latency and cost.



Frontier hosted models are valuable for ambiguous work: architecture, specification compilation, research synthesis and difficult diagnosis. Open-weight models can be run on owned workstations, private clusters or rented accelerators for bounded, repeatable engineering roles. Deterministic tools and gates should decide whether either model's output enters the product.

This hybrid creates a local AI workstation in the architectural sense. “Local” means controlled execution and data boundaries, not necessarily a graphics card under a desk. The hardware may be on-premises, in a dedicated tenancy or rented for a job. The intended boundary gives the organisation explicit control over model selection, inputs, retention and surrounding tools; rented infrastructure still requires verification of the provider's access and data-handling arrangements.

The architectural unit is a role with an acceptance contract, not a model endpoint. A specification reviewer, fixture generator and incident summariser differ in what they may read, what they may change and how their output is judged. Putting them behind the same chat interface can conceal those differences. A workstation architecture makes them explicit even if, initially, one model happens to perform all three roles.

That separation creates a practical substitution test. Could another approved model perform the role without changing the product's authoritative rules or giving it broader permissions? If changing a provider requires recovering decisions from chat history or rewriting business semantics around a vendor response format, the organisation does not yet own the boundary. Portability is established by durable contracts and evaluation, not by a configuration field labelled `model`.

Open, open-weight and closed

The terms matter.

Weight access is not software freedom

Downloadable weights

A deployment option with a specific licence and supply chain.

Required record

Terms, model revision, runtime and the configuration actually evaluated.

A closed model is accessed as a service; weights, training data and much of the serving stack are unavailable. An open-weight model makes trained parameters available under a licence, but may not disclose training data or permit every use. Open source properly describes software whose licence grants established freedoms; it should not be used loosely for every downloadable model.

Meta distributes Llama models for download under its own licence. That enables controlled deployment but does not make every model or dependency unrestricted. An engineering decision must examine licence, acceptable-use terms, provenance, security updates and the ability to reproduce the serving stack.

Closed services often provide the strongest capability and easiest scaling. Controlled models provide observability, data locality, stable versioning and freedom to tune the serving environment. Neither property guarantees quality.

Control should be described as a set of observable properties. Where does input data go? Which components can make network requests? Can the model and runtime versions be pinned? Who can change the prompt or tools? Which logs contain sensitive material, and who can read them? Weight availability answers only part of that inventory. A controlled model with an uncontrolled plugin or logging path may violate the intended boundary.

The decision should also distinguish reproducibility from repeatability of generated text. Pinning weights and serving configuration helps reproduce an environment, but the accepted artefact still needs independent checks. Two different completions can both be

valid; identical completions can both be wrong. The role contract should specify the product-level property that matters rather than treating verbatim output stability as a substitute for correctness.

The capability gap moves

Stanford's 2025 AI Index reported narrowing gaps between leading open-weight and closed models on selected benchmarks and falling inference costs at a fixed capability level. The 2026 report shows that the gap fluctuates and had widened on some aggregate measures. These are historical benchmark observations, not a current recommendation for a particular engineering role. [2025 technical performance](#); [2026 technical performance](#).

Make substitution an evidence decision

Candidate change

A new model, runtime or serving configuration.

Acceptance basis

Current results on the organisation's particular role corpus.

The correct conclusion is not a straight-line forecast that open models inevitably catch up. Capability moves as new closed and open models are released. Benchmarks also compress different tasks into one number.

An organisation needs a re-evaluation process. Record which role a model performs, the evaluation corpus, quality threshold, latency, memory and cost. Re-run the evaluation when a new candidate appears. Switch when evidence supports it. Control comes from the harness, not allegiance to an ecosystem.

A benchmark average can conceal the failure that disqualifies a model for a particular role. An excellent documentation summariser may be unreliable at preserving a migration invariant; a competent code transformer may mishandle instructions embedded in a log. Evaluate the consequential failure classes separately and state which thresholds are non-negotiable. A candidate should not compensate for a serious permission failure by performing many easy tasks quickly. This makes selection a role-specific acceptance decision while still allowing the portfolio to benefit from improvements in capability, serving efficiency and cost.

Frontier models for specification

High-capability cloud models are most valuable where an error would propagate widely and the task requires broad synthesis.

Spend reasoning where mistakes propagate

Frontier task

Resolve ambiguity and architectural consequence.

Durable output

A reviewed decision and specification outside the vendor conversation.

Use them to reconcile conflicting requirements, identify domain boundaries, review architecture, generate counterarguments and compile work into units. Give them curated authoritative context, not the whole estate. Preserve decisions and resulting specifications on disk so a vendor conversation is not the system of record.

Sensitive material may constrain this use. Redact, abstract or execute within an approved managed boundary. The model should receive the minimum information required for the decision.

The frontier session produces proposals and a versioned artefact. It does not push unrestricted code into production.

Frontier use can be bounded without reducing the task to fragments too small to understand. Supply the complete relevant contradiction, the affected contracts and the consequence of each available interpretation. Exclude unrelated secrets and history. The output should distinguish a proposed decision from the facts supporting it, identify unresolved assumptions and state which artefacts would change if accepted.

This creates a useful handoff to cheaper execution. The frontier session does not need to remain present while every file is edited. It produces a contract that a worker can consume and a reviewer can inspect. If implementation uncovers a new architectural question, return that bounded question with evidence. Otherwise the expensive reasoning tier becomes an indefinitely open conversation, and the organisation pays repeatedly to reconstruct context that should have become a durable artefact.

Controlled models for engineering roles

Many delivery tasks are narrower than architecture:

Local does not mean authorised

Controlled worker

Receives the role's bounded inputs and allowed artefacts.

Permission boundary

No broader credentials or production rights merely because serving is local.

- implement one unit against a stable contract;

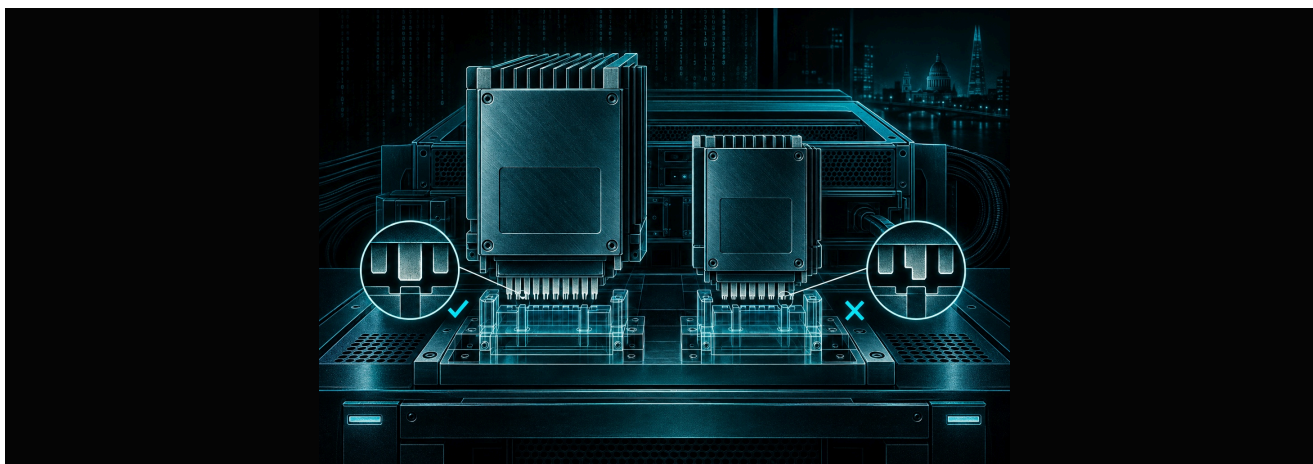
- generate fixtures and test variants;
- classify logs inside a known taxonomy;
- update documentation from a diff;
- scan for inconsistent naming or missing cases;
- triage mutation survivors; and
- prepare a patch for human or automated review.

These roles can be evaluated repeatedly. A local model that meets the threshold may be cheaper, faster and easier to govern than a frontier API. Stable weights also reduce unannounced behavioural change during a programme.

Do not give a model broad filesystem, production or credential access because it runs locally. Local inference changes the data boundary; it does not remove prompt injection, unsafe tool use or generated defects.

Commodity hardware and quantisation

Model serving is constrained by parameter memory, activation memory, context length and throughput. Quantisation reduces the numerical precision used to represent weights and, in some schemes, activations, allowing larger models to run on smaller hardware.



Activation-aware Weight Quantization reported strong 4-bit results and substantial speed-ups with its TinyChat implementation while protecting a small set of salient weights. Results depend on model, hardware and workload; quantisation can reduce quality. The relevant measure is performance on the organisation's own role corpus.

The AWQ paper supplies evidence for a particular compression and serving technique, not for every model distributed at four-bit precision. Its results should not be transferred to a different quantiser, runtime or task without measurement. Memory savings can enable a deployment that otherwise would not fit, while a subtle loss on the organisation's acceptance cases can still make it unsuitable. [Lin et al., AWQ](#).

Serving frameworks such as vLLM support tensor and pipeline parallelism across accelerators. This makes scale a deployment choice: one workstation for sequential bounded tasks, multiple local devices for throughput, or rented boxes for bursts. Idle expensive hardware is not automatically economical; compare utilisation, power, administration and rental pricing. [vLLM parallelism and scaling](#).

Memory planning should include the workload around the weights. Long contexts, concurrent requests and the serving runtime can change whether a configuration fits and how quickly it responds. A model that runs a short demonstration may fail to provide useful throughput for several real workers. Evaluate the context lengths, tool exchanges and output sizes the role actually uses, with headroom for the expected operating pattern.

Quantisation decisions belong in the same evaluation. Compare accepted-unit outcomes, not only tokens per second. A faster configuration that increases malformed tool calls or repair work may reduce delivery throughput. Keep the unquantised or previously accepted configuration as a comparison where feasible, and preserve the relevant model and runtime versions. The objective is an economical controlled role, not fitting the largest possible model onto a machine as an end in itself.

Managed boundaries such as Bedrock

Managed model platforms occupy a middle ground. AWS describes Bedrock controls for isolation and encryption and states that customer content is not used to train base models. Its data-retention documentation is model- and feature-specific, including cases with retention inside the AWS boundary and compatibility limits for zero-retention configurations.

Name the managed configuration	
Provider label	Inspect instead
Insufficient to establish a data policy.	Model, invocation path, access, logging and retention actually in use.

That nuance matters. “Enterprise” is not a data policy. Review the exact model, invocation mode, logging, abuse detection, region, retention and contractual terms. Record the approved configuration and test that unapproved routes cannot be used.

The [Bedrock retention documentation](#) is the configuration-specific source, rather than a general promise inferred from the service name. A review should record the model and feature actually invoked and examine associated storage such as invocation logs. A zero-retention option at one boundary does not describe every copy made by the surrounding application. This paper does not certify a chosen deployment’s data handling.

Managed platforms can reduce operating burden and provide access to multiple vendors. They can also increase dependency on provider-specific interfaces. Put an internal contract around model use so roles can move when economics or policy changes.

The role system

A workstation becomes useful when it supplies roles rather than a generic chat box.

A role contract survives a model switch

What enters

Defined inputs and a permitted task.

What may happen

Named tools, bounded outputs and explicit escalation conditions.

Each role has an objective, allowed sources, tools, output schema, permission boundary, evaluation set and escalation condition. A specification reviewer may read product artefacts but not edit implementation. A builder may change files in one unit. A verifier may run tests and propose faults but not rewrite the test to make a failure disappear. An operator may summarise telemetry but not execute a destructive recovery without approval.

Roles make model substitution possible. They also reveal when a task requires frontier judgement rather than local execution.

A constructed builder role illustrates the contract. It receives one unit manifest and named source files, may edit an isolated working area, can run approved test commands and must return a patch plus evidence. It cannot read production credentials or alter acceptance thresholds. Its escalation conditions include contradictory sources, a required public-interface change or a failure that lies outside its owned artefacts.

A verifier role should have a different contract. It may inspect the candidate, run the gates and introduce controlled faults in a disposable environment, but it should not quietly change production code to make its report easier to write. Separation does not require two permanent machines. It requires clear permissions, evidence provenance and a review boundary between producing an artefact and accepting its claim. Those controls remain useful when the model names and hardware change.

Harnesses provide control

Evaluate models on the actual work.

Evaluate the whole role

Acceptance observations

Correctness, refusal, repair effort, latency and cost.

Adversarial challenge

Hostile instructions and invalid tool requests as well as ordinary work.

Create a corpus of representative units, known defects, architecture decisions, security traps and documentation tasks. Score correctness, refusal, diff size, gate pass rate, human rework, latency and cost. Include adversarial instructions embedded in repositories and logs. Preserve model version, serving parameters and tool permissions.

The harness should distinguish proposal quality from final product quality. A model may produce a useful first patch that fails gates and is repaired cheaply. Another may pass superficial tests while damaging architecture. Measure the whole delivery loop.

Re-evaluate after model, quantisation, prompt, tool or dependency changes. A cached benchmark from a model card is not evidence for the local system.

Keep an evaluation set that the role has not merely memorised through repeated prompt tuning. It should represent ordinary work and important failure classes, including a request containing an instruction embedded in an untrusted log or repository file. The expected response follows the role's authority boundary, not the apparent urgency of the embedded text. A locally served model is exposed to the same confusion if the surrounding system fails to distinguish data from instructions.

Measure rejection and escalation as well as completion. A worker that correctly refuses an out-of-scope operation may be doing better work than one that produces a plausible patch by taking unauthorised shortcuts. Human review should examine a sample of accepted outputs and failed attempts to identify what the automated metrics miss. The harness itself needs versioning so a later improvement can be distinguished from an easier evaluation or a narrower definition of success.

CI/CD is the constitutional layer

The delivery pipeline enforces what no model is permitted to negotiate.

A worker cannot grant itself an exception

Failed gate

The worker may propose a correction with evidence.

Weakened obligation

Requires a separate reviewed decision; a local edit is not authority.

Compilation, types, schema validation, unit and system tests, mutations, security checks, accessibility, performance budgets, infrastructure policy, migration rehearsal and deployment approvals run outside the agent's persuasion. Branch protection and artefact signing preserve provenance. Production permissions are separate from build permissions.

An agent may diagnose a failed gate and propose a correction. It may not delete the gate, weaken the threshold or rewrite the requirement without an explicit reviewed change. This is how probabilistic workers participate in deterministic delivery.

Security and operations

Local models introduce a software supply chain: weights, tokenisers, serving runtimes, GPU drivers and model-loading code. Verify sources and hashes. Isolate serving. Restrict outbound network access. Treat model output as untrusted. Monitor resource use and requests without retaining sensitive content unnecessarily.

The workstation has an operating burden

Dependencies

Weights, serving runtime, drivers, tools and updates.

Recovery

A tested approved route or human handling from durable artefacts.

Plan failure. What happens when the workstation is unavailable, a model produces malformed output, a rental expires or a new version regresses? Roles should degrade to another approved model or to human work. Durable specifications and ordinary source control prevent the model estate becoming a single point of organisational memory.

Operational failure should leave useful state outside the inference process. If a worker crashes halfway through a unit, the next approved worker needs the original contract, the partial patch and a clear record of which checks ran. It should not have to infer acceptance from the existence of files. A malformed model response should be rejected at the tool boundary rather than interpreted generously because the server is trusted.

Supply-chain controls should follow the complete loading and serving path. Record where the model and runtime came from, which components execute code, how updates are approved and how a compromised or regressed version is withdrawn. The same inventory helps recovery when a driver or serving change causes failures. Owning hardware makes these responsibilities more direct; it does not automatically make them easier than using a well-governed managed service.

Economics

Compare total cost per accepted unit, not tokens or GPU purchase price.

Cost per accepted unit

Numerator

Preparation + inference + tooling + verification + repair + allocated operation.

Denominator

Accepted comparable units at the fixed evidence standard; rejected attempts still cost.

Include frontier calls, local hardware amortisation, electricity, hosting, engineering time, rejected work, verification and operator toil. A cheap model that requires extensive correction may be expensive. A frontier model used for every mechanical transformation may waste money. A rented accelerator can beat ownership for bursty demand; owned hardware can win under steady high utilisation or strict data constraints.

The tiering hypothesis is measurable: frontier proposes, humans authorise, workhorse executes, gates verify. Instrument each tier and change the allocation when data contradicts it.

An illustrative comparison starts with a fixed set of units and one acceptance standard. Record the frontier effort needed to prepare each contract, worker execution time, rejected attempts, human repair and gate cost. Then compare allocations. A stronger worker may cost more per call but require less repair; a smaller worker may be excellent for repetitive fixtures and unsuitable for a subtle migration. Averaging the two task types can hide the decision that matters.

Utilisation changes the ownership calculation. Capacity purchased for occasional peaks can sit idle while still incurring capital and operating burden. Rented capacity can avoid some idle cost but introduce setup, transfer and availability constraints. These are scenario inputs to measure, not fixed arguments for one deployment model. Cost per accepted unit should remain attached to the workload and utilisation assumptions under which it was observed.

A decision model without invented savings

Define an accounting period and a fixed family of comparable work. Let total delivery cost include preparation, inference, tooling, verification, human repair and allocated operating cost. Divide that total by the number of accepted units at the agreed evidence standard. Rejected attempts stay in the numerator. If no units are accepted, report failure to establish an economic rate rather than dividing by an optimistic forecast.

This denominator needs care. A fixture transformation and a database migration are not exchangeable units. Group by obligation and risk, then retain the distribution of repair effort within each group. A low average can conceal a small number of units that consume most reviewer time. Also record elapsed time to acceptance: a cheap configuration that leaves a critical dependency waiting can impose a programme cost absent from its inference bill.

Cost or constraint	Owned serving	Rented serving	Hosted model service
Capacity	Purchase and provision for expected demand	Reserve or acquire capacity for the job	Depend on the service's availability and limits
Idle time	Include allocated capital and operating burden	Include reservations or minimum billing where applicable	Include committed spend where applicable
Operations	Own runtime, patching and recovery	Share responsibility according to the provider contract	Own application boundary and provider integration
Quality	Evaluate the complete local configuration	Re-evaluate changes in runtime and hardware	Evaluate the exact approved model and invocation mode
Data boundary	Inspect tools, logs and outbound paths	Inspect tenancy, access and retention	Inspect contractual and technical controls

The table compares questions to answer, not a winner. Gather actual invoices, operator time and accepted-work records for the candidate arrangement. A team with idle existing capacity may face a different incremental decision from one buying a machine. A team whose policy prohibits an external route has a feasibility constraint before an economic comparison. Mixing these situations produces a misleading universal break-even claim.

The same framework can test the tiering proposal. Hold the compiled obligation and acceptance gates fixed. Compare direct frontier execution with a prepared contract executed by an approved workhorse. Record escalation and repair rather than assigning those costs to an invisible human helper. The allocation earns its place only if the whole loop meets quality, time and cost constraints. Nothing in the architecture guarantees that the smaller model wins.

A reference architecture

A log cannot grant a capability

The diagnostic role's tool boundary has to reject a secret-read or upload request even if the model follows the hostile text.

Untrusted input	An error-log field asks for credential extraction.
Tool boundary	Credential access and arbitrary upload are unavailable.
No approved route	Local serving fails → leave the task unexecuted.

Constructed failure exercise · a silent hosted fallback would violate the stated data boundary.

1. Durable specification and repository remain authoritative.
2. A router selects an approved role and model from task risk and data class.
3. Frontier services receive redacted, bounded context for high-judgement tasks.
4. Controlled open-weight models execute repeatable units in isolated workspaces.
5. Tools expose least-privilege operations through typed interfaces.
6. CI/CD runs independent gates and records evidence.
7. A human owns scope, exceptions, consequential decisions and production release.
8. An evaluation harness continuously compares model candidates.

This architecture can begin on one machine. Its value is separation, not scale.

The router should make a policy decision before any model sees the payload. It identifies the task, data class and allowed role, then selects an approved configuration that has passed the relevant evaluation. If no approved route exists, the task escalates or remains unexecuted. Falling back silently to a public endpoint would defeat the control boundary at the moment it is most needed.

Results travel back through a typed artefact boundary. Generated text can propose a change, but a parser, schema or patch application step decides whether its structure is admissible. Independent gates then decide whether the artefact satisfies the unit. Human authority remains responsible for changed meaning, exceptional risk and release. This sequence gives each tier a narrower job and makes failures easier to locate: routing, generation, structural validation, behavioural acceptance and authorisation are distinct decisions.

Worked failure: a worker meets an instruction in a log

Suppose a local diagnostic role receives an application error log. One field contains text supplied by an external user telling the assistant to read a credential file and upload it for diagnosis. The diagnostic role is authorised to classify errors and propose likely causes. The embedded text is evidence about an input, not a grant of new authority.

The strongest control is outside the model: the tool interface cannot read that credential path or send arbitrary network requests. The model should identify the suspicious instruction and continue the permitted analysis, but security cannot depend on its choosing the right interpretation every time. Record the denied action if attempted, without copying the secret into the evidence log. A useful evaluation includes both a model that refuses correctly and a simulated invalid tool request that the boundary itself rejects.

Now make the local server unavailable during the same task. A router that silently retries through an unapproved hosted service has defeated the data policy while apparently improving reliability. The accepted fallback must be another approved route or a clear unexecuted state. Preserve the diagnostic input under its retention rules and the partial result with its status. Availability and confidentiality cannot be reconciled by letting the worker improvise a new policy.

This case explains why the workstation is a delivery system. The model, router, tool permissions, logs and recovery path jointly determine its behaviour. Buying stronger hardware changes none of these authority relationships. A buyer should request the role contract, the denied-action evidence and the fallback exercise alongside any throughput benchmark.

Limits

Local deployment is not automatically private, cheap or sustainable. Logs, package downloads and telemetry can leak data. Hardware becomes obsolete. Model licences change between families. Small models may fail silently on complex tasks. Managed providers can offer stronger security operations than a small team can maintain.

Location cannot settle the decision

Owned serving

May leak through logs or tools, or cost more after operation and repair.

Managed serving

May meet the boundary through appropriate technical and contractual controls.

Closed frontier models are not automatically unsafe or uncontrollable either. Contractual and technical controls can create an appropriate boundary. The decision must follow data, task and evidence.

The workstation is therefore a portfolio, not a bunker.

Conclusion

The future of AI engineering is unlikely to be one model in one cloud performing every role.

Use frontier capability where ambiguity and architectural consequence justify it. Preserve the resulting specification outside the conversation. Execute bounded work with the cheapest controlled model that passes a role-specific harness. Let CI/CD and deterministic gates decide what survives. Keep human authority over meaning, risk and release.

Open and closed models then become interchangeable components of an owned delivery system rather than rival belief systems.

References

- Stanford HAI, [AI Index 2026: Technical Performance](#).
- Stanford HAI, [AI Index 2025: Technical Performance](#).
- Lin et al., [AWQ: Activation-aware Weight Quantization](#), 2023.
- vLLM project, [Parallelism and scaling](#).
- AWS, [AWS SRA for AI](#).
- AWS, [Amazon Bedrock data retention](#).
- Meta, [Llama model downloads and licence details](#).