



Red Hat AI 3.5 Upgrade: GPU Serving and Token Showback

Plan and build private AI compute with GPU Smith. We connect workload requirements with GPU hardware, networking, deployment and operating decisions so your infrastructure fits the work it must perform.

Published by GPU Smith · 2026-10-15 · red hat ai 3.5 upgrade · red hat ai · openshift ai · multi-tenant inference · gpu scheduling · token showback · tenant isolation · gpu cost allocation · upgrade readiness

Original article: <https://gpusmith.com/articles/en/red-hat-ai-3-5-upgrade-gate>



In this article

1. Executive Summary
 2. Introduction and Background
 3. Key Changes: Capability, Maturity, and Architecture
 4. Shared-GPU Control: Fairness, Priority, and Admission
 5. Implementation Considerations and Process Changes
 6. Token Showback: Meters, Attribution, and Internal Cost
 7. Reproducible Acceptance Lab
 8. Data Analysis and Evidence
 9. Implications and Future Directions
 10. Conclusion
-

Executive Summary

Upgrade Red Hat AI 3.5 when a supported capability closes a measured operating gap and passes a reproducible acceptance test. Pilot preview-dependent accounting. Wait when compatibility, recovery, or the required operating behavior remains unproven. Red Hat announced general availability on **September 9, 2026**, but maturity is capability-specific. (www.redhat.com) Its launch material identifies **EvalHub** as GA and **MaaS showback** as TP. (www.redhat.com) (www.redhat.com) Token consumption should therefore be evaluated separately from an internal financial charging decision.

For **shared-GPU inference**, define what fairness means before approving the upgrade. Equal dispatch turns do not necessarily produce equal resource use or user experience. Prompt processing and sequential generation are different phases, while primary serving research evaluates capacity against both first-token and subsequent-token constraints. Use tenant and priority-specific acceptance criteria, preserve background progress requirements, and include overload and endpoint failures in the lab. An open arrival model keeps offered arrivals independent of response completion; generator-side dropped iterations must be disclosed. (mlcommons.org) (arxiv.org) (grafana.com) (grafana.com)

For **tenant isolation**, distinguish namespace controls, separate control planes, virtual machines, and dedicated hardware. Kubernetes explicitly states that per-tenant control planes do not solve data-plane isolation. NVIDIA separately documents that time-sliced GPU replicas have no memory or fault isolation between replicas. Neither a tenant cluster nor an advertised GPU share establishes the complete inference boundary. Require evidence of worker placement, device attachment, storage, identity, and network controls appropriate to the workload. (kubernetes.io) (docs.nvidia.com)

For **token showback**, reconcile request identity, input and output accounting, restarts, and infrastructure allocation before using the data financially. OpenCost separates workload, idle, and overhead costs; FinOps permits several allocation methods rather than prescribing a universal internal token rate. (opencost.io) (framework.finops.org) A useful quantitative reference is MLCommons' benchmark-specific server limits of **2 seconds TTFT** and **100 milliseconds TPOT**, versus interactive limits of **0.5 seconds** and **30 milliseconds**.

(mlcommons.org) (mlcommons.org) These illustrate differentiated objectives, not Red Hat performance guarantees. The report supplies a proposed acceptance lab, isolation matrix, and local cost equations; it reports no original benchmark or billing-accuracy result.

Introduction and Background

A **Red Hat AI 3.5 upgrade** should begin with an operating problem: interactive requests waiting behind batch work, unclear tenant boundaries, or consumption that cannot be assigned to an accountable owner. Red Hat announced the release on **September 9, 2026**. (www.redhat.com) The portfolio announcement establishes availability; component documentation is needed to establish configuration and support boundaries.

This report uses **October 7, 2026** as its editorial reference date. Research verification occurred on **October 15, 2026**. Dated launch statements and subsequently retrieved, mutable documentation are distinguished throughout. Current documentation supplies implementation checks, not proof of an archived configuration on the publication date. No original performance benchmark, deployment result, customer experience, or billing-accuracy measurement is claimed.

The architecture explains why a version number alone cannot settle the decision. Upstream **llm-d** separates routing, an **InferencePool**, and model servers. The pool groups servers for the same base model, while the **Endpoint Picker (EPP)** selects destinations using operating metrics and policy. These are upstream architectural definitions, not an assertion that every current upstream feature ships in the supported product. (llm-d.ai) (llm-d.ai)

The **Gateway API Inference Extension** likewise separates endpoint selection from forwarding: the picker advises the gateway where to send a request. **vLLM**, the serving engine discussed in the measurement sources, exposes engine-level metrics; these represent a different observation point from the client and gateway. (gateway-api-inference-extension.sigs.k8s.io) (docs.vllm.ai)

An acceptance-based engineering perspective is appropriate here. GPU Smith's published method describes engagements against written acceptance criteria. This report applies that discipline to the upgrade decision, with proposed tests and explicit evidence requirements. (gpusmith.com)

The intended readers are platform engineers, private-AI operators, security architects, finance operations (**FinOps**) teams, and procurement leads. Their shared decision is whether the release closes a measured operating gap at an acceptable support and infrastructure cost.

Key Changes: Capability, Maturity, and Architecture

Separate product availability from feature maturity

General availability (GA) and **Technology Preview (TP)** must be evaluated per capability. Red Hat's launch blog identifies **EvalHub**, its model and agent evaluation toolkit, as GA, while **Model-as-a-Service (MaaS) showback** is TP. The blog's description of the latter as billing-grade metering is a vendor characterization, not an independently demonstrated accounting result. (www.redhat.com) (www.redhat.com)

Table 1 maps the release claims to acceptance evidence. Dependencies in the final column are checks to complete locally, not a complete support matrix.

Capability	Documented maturity or boundary	Operational decision	Acceptance evidence to retain
Priority-aware distributed inference	Current product release notes identify flow control as GA. (docs.redhat.com)	Evaluate where interactive and background traffic share model capacity.	Saved policy, authenticated priority assignment, saturation results, and queue outcomes.
MaaS token showback	Launch blog identifies TP. (www.redhat.com)	Pilot attribution before making it an internal financial dependency.	Token reconciliation, identity mapping, counter continuity, and exception ownership.
Controlled model deployment	Current release notes describe active-request completion during promotion and rollback. (docs.redhat.com)	Test model-level traffic changes separately from platform recovery.	Streaming completion records and a rehearsed return to the previous model deployment.

The distinction changes scope. A team can accept a supported serving capability while confining preview accounting to a pilot. Evaluation evidence should have its own acceptance criteria; a successful evaluation run does not establish scheduling behavior or financial reconciliation.

Keep the scheduling layers separate

Admission decides when a request may advance; endpoint scheduling decides where it goes. Red Hat explicitly distinguishes these functions. Upstream gateway architecture also separates the proxy from its endpoint-selection extension. (developers.redhat.com) (gateway-api-inference-extension.sigs.k8s.io)

For the upgrade dossier, draw separate boxes for workload resource admission, request flow control, endpoint choice, model execution, and telemetry. Attach the relevant installed component and configuration to each box. Avoid treating a GPU resource quota, an admitted request, and a completed response as equivalent accounting or scheduling events.

That separation matters for evidence. A namespace quota governs particular Kubernetes resource allocations, but Kubernetes cautions that quotas do not protect every shared resource, including network traffic. A model-pool acceptance test therefore needs to observe the actual serving path as well as the cluster's resource objects. (kubernetes.io) (llm-d.ai)

Shared-GPU Control: Fairness, Priority, and Admission

Fair share needs a written definition before testing. It might mean equal dispatch turns, reserved resource capacity, comparable latency, or a limit on how long background work waits. These are different contracts. Red Hat documents round-robin fairness as equal dispatch turns within a priority band; it does not describe equal GPU time per tenant. (docs.redhat.com)

Request size makes that distinction operationally important. Prompt processing and sequential token generation are separate phases in MLCommons' language-model benchmark description. DistServe research evaluates capacity against both first-token and subsequent-token constraints. Equal request counts therefore cannot, by themselves, demonstrate the intended user experience. (mlcommons.org) (arxiv.org)

The following are proposed acceptance requirements:

- **Tenant identity:** Derive the scheduling identity from an authenticated boundary and retain its mapping to the business owner.
- **Priority assignment:** Define who can request each class and how unauthorized assignments are handled.
- **Fairness objective:** Specify whether the contract concerns dispatch opportunities, resource reservation, or completed work.
- **Background progress:** Agree a maximum acceptable wait and a minimum useful service rate for low-priority work.
- **Admission outcome:** Record rejection, queue expiry, cancellation, and completion separately.
- **Capacity transition:** Repeat the same workload as endpoints enter and leave service.

Scheduling identity also needs a platform-specific configuration review. The OpenShift AI procedure documents objective assignment through gateway authentication policy and matching InferenceObjective resources. The third-party Kubernetes procedure requires the relevant fairness and objective headers to be supplied manually or by an authentication layer. Verify that clients cannot assign themselves an unauthorized class. (docs.redhat.com) (docs.redhat.com)

These are local requirements, not a list of vendor guarantees. Red Hat's documented default fairness policy is greedy and ignores tenant boundaries; its guidance identifies round-robin as the choice for equitable same-priority dispatch. Under strict priority ordering, sustained high-priority load can indefinitely defer lower-priority traffic. Policy configuration is therefore part of the acceptance evidence. (docs.redhat.com) (docs.redhat.com)

A useful test contains both unequal tenants in the same band and different bands competing for capacity. Run comparable request shapes first to isolate the policy effect, then repeat with long prompts, long outputs, and mixed streaming behavior. Save the offered traffic distribution, not just successful response counts. An open arrival model is useful because arrivals remain independent of previous request completion. (grafana.com)

Capacity evidence must include the generator. Grafana documents unsent iterations separately from server responses, and arrival-rate tests can drop iterations when virtual users are unavailable. A generator that cannot sustain its intended input does not establish how the serving system behaves at that input. (grafana.com) (grafana.com)

The decision rule is straightforward: accept shared serving when the configured policy meets the agreed tenant and workload objectives under contention. Neither a policy object's existence nor high aggregate token throughput is sufficient evidence.

Implementation Considerations and Process Changes

Select an isolation boundary before optimizing utilization

Kubernetes namespaces, role-based access control (**RBAC**), and network policies provide logical segmentation. Cluster-scoped resources and a shared data plane remain separate concerns. Kubernetes explicitly warns that per-tenant control planes do not solve data-plane isolation problems. (kubernetes.io) (kubernetes.io)

Table 2 compares architectural choices. Utilization and cost effects are planning considerations to measure locally, not quantified findings.

Boundary	What the evidence establishes	Boundary requiring separate proof	Proposed operating test
Namespace and RBAC	Roles and network policies are namespace-scoped. (kubernetes.io)	Cluster administration, shared nodes, network contention, and accelerator sharing.	Attempt cross-project access with tenant credentials and stress permitted shared resources.
Dedicated tenant control plane	Separate control planes leave data-plane isolation unresolved. (kubernetes.io)	Worker placement, GPU access, storage, and inference traffic.	Trace control-plane and inference-worker placement independently.
VM boundary	Hardware virtualization can add an isolation layer; Kata documents a dedicated guest kernel. (katacontainers.io) (katacontainers.io)	The selected accelerator attachment and supported product configuration.	Validate guest separation, device assignment, restart behavior, and resource contention.
Dedicated hardware	Kubernetes identifies dedicated hardware as an option where a VM boundary is insufficient. (kubernetes.io)	Network, storage, identity, and management infrastructure can still be shared.	Verify physical placement against the complete tenancy design and inventory.

The table should guide the threat model, not rank configurations by an assumed universal security score. Red Hat's hosted-control-plane documentation includes a VM-based option in which control-plane pods are pinned to dedicated VMs. That describes control-plane placement; it does not independently prove dedicated inference GPUs. (docs.redhat.com)

Accelerator sharing needs an additional decision. NVIDIA documents that time-sliced replicas lack memory and fault isolation between replicas, and that additional replicas do not guarantee proportional compute. These are documented mechanism boundaries, not claims about an observed deployment. Multi-Instance GPU (**MIG**), [time slicing](#), [passthrough](#), and virtual GPU configurations must be named explicitly in the build sheet. (docs.nvidia.com) (docs.nvidia.com)

Kata's own description identifies network, input/output, and memory isolation through the guest kernel. (katacontainers.io) A dedicated guest kernel is useful evidence of a virtualization boundary, but it is not evidence that a particular accelerator, runtime, or Red Hat AI combination is supported. The same caution applies to upstream gateway architecture: an architectural pattern does not establish a vendor support entitlement. (katacontainers.io) (gateway-api-inference-extension.sigs.k8s.io)

Build a version and recovery dossier

Current Red Hat documentation identifies **OpenShift AI 3.5** as an **Extended Update Support (EUS)** release and documents direct upgrades from **2.25** and **3.4**. (docs.redhat.com) For the documented in-place upgrade from **2.25**, a robust full-cluster backup is described as the only rollback mechanism. (docs.redhat.com) A model traffic rollback must not be substituted for this platform recovery procedure.

Before authorizing the change, retain:

- **Product inventory:** Exact installed product, operator channel, component images, serving runtime, and enabled capabilities.
- **Support evidence:** Dated confirmation of the intended OpenShift, Kubernetes, accelerator, driver, firmware, and runtime combination.
- **Entitlement evidence:** The contracted products, support term, preview exclusions, and support contacts.
- **Recovery evidence:** A restorable backup, dependency inventory, recovery owner, and rehearsal results.
- **Configuration evidence:** Exported scheduling objects, tenant mappings, monitoring rules, and gateway configuration.
- **Change evidence:** The completed component-specific migration checks and a signed acceptance record.

For migrations from the earlier preview, current release notes require updates to `InferenceObjective` and `EndpointPickerConfig`: the API group changes to `llm-d.ai`, metric prefixes change to `llm_d_epp_`, and saturation detection moves under `flowControl.saturationDetector`. Inspect the operator's `odh-upgrade-acks` gates as part of the change record. (docs.redhat.com)

These are procurement and engineering requirements. The research did not establish a complete entitlement or accelerator-support matrix. Accelerator vendors' own compatibility pages are useful inputs: AMD publishes release-specific system requirements, while Intel's hardware tables identify devices and driver-related support details. Neither source independently certifies the combined Red Hat deployment. (rocm.docs.amd.com) (dgpu-docs.intel.com)

Driver ownership also varies across environments. Microsoft documents AKS-managed installation and maintenance of the NVIDIA driver. AWS documents disabling GPU Operator installation of the driver and toolkit for its EKS-optimized AL2023 NVIDIA images because those components are included already. Use these examples to require an explicit ownership model, not to infer cross-platform product equivalence. (learn.microsoft.com) (docs.aws.amazon.com)

Treat disconnected operation as a complete dependency test

Red Hat's disconnected upgrade procedure includes mirroring cert-manager Operator images. (docs.redhat.com) It also documents migration from managed embedded Kueue to the separate operator, with **OpenShift 4.19 or later** required for that migration. (docs.redhat.com) This is a migration-specific prerequisite, not a demonstrated universal minimum for every deployment.

An offline model cache is only part of [disconnected readiness](#). Hugging Face's version-pinned documentation says offline mode uses cached files and errors when required files are missing; telemetry has a separate disabling setting. Verify the actual installed library version and all model artifacts rather than assuming a single environment variable covers the deployment. (huggingface.co) (huggingface.co)

The disconnected rehearsal should cover image pulls, model and tokenizer loading, certificates, identity dependencies, dashboards, evaluation assets, logs, and recovery. Perform it with external access actually unavailable and preserve the dependency inventory. Changes to accelerator partitioning deserve a

maintenance plan as well: NVIDIA documents that MIG reconfiguration requires GPU workloads to be stopped. (docs.nvidia.com)

Token Showback: Meters, Attribution, and Internal Cost

Showback presents attributed consumption and cost to an owner. For this report, **chargeback** means using that allocation to transfer an internal financial charge. The decision to charge is organizational; a token counter alone does not specify the allocation policy. FinOps guidance separates dedicated and shared costs and permits fixed, proportional, or proxy-based allocation. (framework.finops.org) (framework.finops.org)

The first acceptance question is what the meter counts. Upstream vLLM documents processed prompt-token counters. A dated OpenTelemetry demonstration describes token-consumption histograms differentiated by input and output. These establish useful observation patterns, not the exact fields or semantics of Red Hat's preview meter. (docs.vllm.ai) (opentelemetry.io)

The proposed attribution record should contain:

- **Owner:** An authenticated tenant identifier and a time-versioned business-unit mapping.
- **Workload:** Model identifier, deployment revision, endpoint, and serving environment.
- **Request:** A traceable request identifier and an explicit policy for retry relationships.
- **Usage:** Separate input, output, and supported cache-related measurements.
- **Outcome:** Completion, rejection, cancellation, timeout, and partial-stream status.
- **Time:** Measurement interval, timezone, ingestion time, and correction history.
- **Meter:** Counter source, software version, scrape health, and reset indicators.
- **Cost policy:** Rate version, shared-cost allocation rule, and reconciliation owner.

This schema is a proposed local contract. FinOps identifies account structures, tags, labels, and derived metadata as allocation tools. The identity key should survive the transition from request handling to organizational reporting; changing a team label should not silently rewrite historical ownership. (framework.finops.org)

Counter continuity needs an explicit test. vLLM counters can reset when the instance restarts. Reconcile interval increases across replicas and restarts, and distinguish an unobserved interval from genuine zero use. OpenTelemetry also cautions that a provider identifier can differ from the actual upstream provider when requests pass through another platform. A provider label is therefore insufficient as the sole tenant or ownership key. (docs.vllm.ai) (github.com)

Infrastructure reconciliation is a separate ledger. OpenCost distinguishes workload costs, idle costs, and overhead, with workload plus idle equaling asset cost. Its specification permits optional distribution of shared costs among tenants. Choose and publish the treatment of idle capacity, reservations, common services, and unallocated consumption. (opencost.io) (opencost.io)

Device telemetry cannot automatically fill every attribution gap. NVIDIA documents a limitation in associating [DCGM-Exporter metrics](#) with containers under GPU time slicing. Maintain a distinction between request-level token attribution, container resource attribution, and physical-device measurements. (docs.nvidia.com)

Privacy should also be explicit. OpenTelemetry's dated demonstration describes default exclusion of prompt content and tool arguments. Inspect actual instrumentation configuration and retain only the information needed for operations and reconciliation; observability should not accidentally expand the intended data-retention boundary. (opentelemetry.io)

Reproducible Acceptance Lab

The lab should compare a retained baseline and the proposed deployment using the same workload manifest. A baseline is a recorded configuration and trace, not an assumed historical result. Model-server, gateway, and client observations should remain distinguishable because endpoint selection and request forwarding are separate operations. (gateway-api-inference-extension.sigs.k8s.io) (llm-d.ai)

A reproducible manifest should capture:

- **Model:** Weights, tokenizer, revision, quantization, and relevant serving options.
- **Hardware:** Accelerator type, memory, partitioning mode, host placement, and fabric topology.
- **Software:** Image digests, driver and runtime versions, operators, and gateway policy.
- **Traffic:** Tenant mix, priority mix, prompt and output distributions, arrival pattern, and streaming mode.
- **Measurement:** Client clocks, histogram configuration, trace sampling, and collection gaps.
- **Recovery:** Endpoint changes, restart events, traffic promotion, and return-to-service procedure.

Network configuration belongs in the manifest. DistServe's primary research treats cluster bandwidth as a placement consideration for separating prompt processing and token generation. That supports recording the network; it does not establish a performance multiplier for this upgrade. (arxiv.org)

Exercise contention and failure deliberately

Start below observed contention, increase offered load until the queueing mechanism is engaged, and continue through a controlled overload period. This follows Red Hat's recommendation to reach measured saturation. Record client-side latency using a monotonic elapsed-time clock; Python documents that this clock is unaffected by system-clock updates. (developers.redhat.com) (docs.python.org) Its performance counter also includes time spent sleeping, a useful distinction from CPU execution time when measuring response delay. (docs.python.org)

Use an **open workload model** for the contention phase, with arrivals independent of completion. k6 documents constant-arrival-rate and ramping-arrival-rate executors for this purpose. (grafana.com) Grafana explains that a closed model waits for previous work, which can reduce offered demand as responses slow. Check unsent iterations before interpreting the server's admission or rejection counts. (grafana.com) (grafana.com)

Table 3 is a proposed queue and recovery test sheet. Every row must be evaluated by tenant and priority, with the same measurement definitions.

Test	Offered workload or event	Evidence to collect	Local acceptance decision
Same-band fairness	Unequal tenant arrival rates, first with comparable request shapes	Intended and sent arrivals, admissions, rejections, dispatch opportunities, completed work	Does the configured fairness objective hold?
Priority contention	Interactive requests with sustained background demand	Time to first token, output-token latency, queue age, background progress	Are interactive objectives and the agreed background bound both met?
Capacity change	Add and remove ready model endpoints during load	Ready endpoints, queue outcomes, errors, throughput, physical GPU utilization	Does service recover within the declared operating objective?
Picker outage	Interrupt endpoint-picking availability	Routing behavior, admission outcomes, per-class latency, policy restoration	Is the configured failure behavior acceptable?
Accounting continuity	Restart replicas and introduce retries, cancellations, and partial streams	Client records, meter deltas, resets, missing intervals, unmatched ownership	Can every accepted discrepancy be explained?
Promotion and recovery	Change model traffic and rehearse the separate platform recovery plan	Streaming completion, retained state, restoration time, compatibility checks	Are both recovery procedures demonstrably usable?

The table deliberately includes unsuccessful work. Grafana distinguishes generator-side unsent iterations, while upstream vLLM distinguishes engine-level timing metrics. Upstream vLLM defines TTFT as a histogram in seconds; verify that dashboards and client records use compatible units. (docs.vllm.ai) A completed-response dashboard alone cannot explain a discrepancy between intended demand and observed server throughput. (grafana.com) (docs.vllm.ai)

The picker-outage row is particularly consequential. Red Hat documents a default route directly to model servers when the picker is unavailable, bypassing flow-control protection. Test the desired failure policy and restoration behavior; do not infer that a running endpoint means its scheduling policy remains active. (docs.redhat.com)

Make the evidence statistically usable

Time to first token (TTFT) measures initial response delay; **time per output token (TPOT)** describes subsequent generation pace. The OpenTelemetry specification calculates server TPOT from duration after the first token and subsequent output-token count. Guard cases with insufficient output tokens when implementing the calculation. (github.com)

Report latency distributions by class. For **P95**, the ninety-fifth percentile, aggregate compatible histogram observations rather than averaging per-replica percentile values. Prometheus explicitly warns that averaging precomputed quantiles is statistically invalid and explains that histogram bucket width limits quantile accuracy. (prometheus.io) (prometheus.io)

Track **SLO-compliant goodput**, meaning useful work completed within the agreed **service-level objectives (SLOs)**. DistServe's research evaluates serving capacity against simultaneous TTFT and TPOT constraints. This provides a defensible measurement principle without importing its experimental performance results into a

different product stack. (arxiv.org)

Data Analysis and Evidence

The quantitative core of this decision should be measured locally. The research establishes benchmark definitions and cost-accounting methods, but does not establish a hardware-normalized Red Hat AI upgrade speedup, overhead percentage, tenant-isolation score, or billing-accuracy result.

Use benchmark thresholds as examples, not product promises

MLCommons' published **Llama3.1-8B** server scenario uses random arrivals with **TTFT at most 2 seconds** and **TPOT at most 100 milliseconds**. (mlcommons.org) Its interactive scenario applies tighter limits of **0.5 seconds** and **30 milliseconds**, respectively. (mlcommons.org) These belong to that benchmark's workload and scenario, not to Red Hat's support contract.

Calculated from those published limits, the interactive TTFT ceiling is **one quarter** of the server ceiling, and the TPOT ceiling is **three tenths**. (mlcommons.org) (mlcommons.org) The point is differentiated acceptance criteria: the same serving configuration must be assessed against the workload's actual responsiveness requirement. These ratios are arithmetic comparisons of benchmark thresholds, not observed performance improvements.

For the local results, publish the interval, sample count, workload distribution, and histogram settings alongside every percentile. Prometheus describes quantiles as estimates and notes their dependence on histogram resolution. A threshold decision close to a bucket boundary should disclose that measurement uncertainty. (prometheus.io) (prometheus.io)

A useful evidence package reports offered demand, demand actually sent, completed demand, and demand completed within the objective separately. k6's dropped-iteration metric identifies work never sent by the generator. DistServe's constraint-based capacity definition provides the complementary reason to distinguish total completions from compliant work. (grafana.com) (arxiv.org)

Reconcile usage and cost with explicit equations

The following is a **proposed local showback model**, not vendor billing:

Tenant showback = metered input tokens × internal input rate + metered output tokens × internal output rate + explicitly allocated infrastructure overhead.

Use customer-supplied rates in consistent units. If cache treatment changes the rate, create a separately reconciled category rather than silently counting it twice. If the token rates already recover all infrastructure spending, identify and prevent overlap with the overhead term. FinOps permits fixed, proportional, or proxy-based shared-cost allocation; the local policy must specify which method applies. (framework.finops.org)

Maintain a second calculation for the asset ledger:

Allocated resource cost = allocated resource amount × duration × hourly rate.

That follows OpenCost's resource-cost formulation. Reconcile allocated and idle resource costs back to assets, then handle additional cluster overhead explicitly. Keep dedicated and shared cost categories explicit when applying that allocation. (framework.finops.org) Token valuation and asset allocation are related views, not inherently identical totals. (opencost.io) (opencost.io)

For a team-level unit metric, divide the agreed period cost by the agreed count of useful operations. FinOps provides the analogous cost-per-API-call formulation and encourages generative-AI unit economics to move beyond tokens toward outcomes. Define what constitutes a completed assist, agent action, or resolved case before comparing teams. (www.finops.org) (www.finops.org)

No example currency rate is supplied because internal prices, ownership costs, reservations, and utilization are organization-specific. Procurement should request the full recurring and capital-cost inputs; the lab should supply the measured denominator.

Implications and Future Directions

The release changes the questions an upgrade review can ask. A priority policy can be tested against workload classes, and accounting can be tested against accountable ownership. The strongest investment case is a documented gap closed by reproducible evidence.

GPU Smith's stated sizing method begins with workload models and throughput targets. Applied here, that means retaining the workload definition through procurement, configuration, and acceptance, so an infrastructure quote and an operating result refer to the same demand. (gpusmith.com)

Upstream evolution remains relevant but requires version discipline. OpenTelemetry describes its generative-AI conventions as actively developing, and its official specification repository defines the instruments and attributes. That version review should include the engine's processed-prompt-token counter and restart behavior. (docs.vllm.ai) (docs.vllm.ai) Freeze the deployed schema in the evidence package and make changes explicit during subsequent upgrades. (opentelemetry.io) (github.com)

The procurement and request-for-proposal (**RFP**) questions should be concrete:

- **Support scope:** Which exact product, operator, platform, accelerator, and runtime combination is covered?
- **Capability maturity:** Which required functions are supported production capabilities and which remain previews?
- **Policy semantics:** What defines a tenant, a priority, an admission outcome, and acceptable background progress?
- **Isolation boundary:** Which control-plane, worker, device, network, and storage boundaries are actually provided?
- **Accounting scope:** What happens to cached input, retries, failed requests, cancellations, and partial output?
- **Recovery commitment:** What restoration procedure is supported for model changes and platform changes?
- **Disconnected operation:** Which images, artifacts, certificates, and services must be locally available?

- **Acceptance deliverable:** Which traces, configurations, measurements, and reconciliation records constitute sign-off?

These questions should produce dated evidence, not a generic feature checklist. AMD's release-specific compatibility matrix and Intel's device-oriented support tables illustrate why hardware identification must be precise. Hosted service documentation also differs in driver responsibility between AKS and EKS configurations. (rocm.docs.amd.com) (dgpu-docs.intel.com) (learn.microsoft.com) (docs.aws.amazon.com)

The future-facing choice is therefore an operating model: shared services with measured policy, or stronger separation where the tenancy requirement demands it. Kubernetes' data-plane warning remains applicable even when tenant control planes are separate. Evaluate the complete boundary and its measured capacity cost. (kubernetes.io)

Conclusion

The **Red Hat AI 3.5 upgrade decision** should be recorded as an acceptance outcome with three possible dispositions: upgrade, pilot, or wait. The evidence package should explain which operating gap justifies the change, which configuration addresses it, and which results demonstrate that the requirement has been met.

Upgrade when the required supported capabilities pass the local contention, access-control, recovery, and compatibility checks. Retain the baseline and accepted configuration so subsequent changes can be compared against an explicit contract.

Pilot selected capabilities when scheduling or attribution is promising but support scope, preview dependencies, or usage semantics remain unresolved. Restrict the pilot's operational and financial consequences, name an owner for each unresolved question, and define the evidence needed to leave the pilot.

Wait when the proposed release does not close a measured gap, when the needed configuration lacks dated support confirmation, or when recovery and disconnected dependencies have not been rehearsed. Waiting is a technical disposition that should identify the missing evidence and a condition for reassessment.

The final sign-off should keep resource fairness, request priority, tenant isolation, and cost attribution as separate judgments. It should also distinguish configuration from behavior, product maturity from local readiness, and token consumption from the full economics of infrastructure ownership. This produces a reviewable upgrade decision without relying on a generic release recap or an assumed performance benefit.

Source links

Links cited in this article, in order of first appearance. Full addresses are provided for readers using extracted text.

1. <https://www.redhat.com/en/about/press-releases/red-hat-puts-safety-and-observability-core-enterprise-ai-red-hat-ai-35>
2. <https://www.redhat.com/en/blog/red-hat-ai-35-scaling-and-governing-ai-agents-production>
3. <https://mlcommons.org/2025/09/small-llm-inference-5-1/>
4. <https://arxiv.org/abs/2401.09670>
5. <https://grafana.com/docs/k6/latest/using-k6/scenarios/concepts/open-vs-closed/>

6. <https://grafana.com/docs/k6/latest/using-k6/scenarios/concepts/dropped-iterations/>
7. <https://kubernetes.io/docs/concepts/security/multi-tenancy/>
8. <https://docs.nvidia.com/datacenter/cloud-native/gpu-operator/25.3.4/gpu-sharing.html>
9. <https://opencost.io/docs/specification/>
10. <https://framework.finops.org/framework/capabilities/allocation/>
11. <https://llm-d.ai/docs/architecture>
12. <https://gateway-api-inference-extension.sigs.k8s.io/>
13. <https://docs.vllm.ai/en/latest/design/metrics/>
14. <https://gpusmith.com/>
15. https://docs.redhat.com/en/documentation/red_hat_openshift_ai_self-managed/3.5/html/release_notes/new-features-and-enhancements_relnotes
16. <https://developers.redhat.com/articles/2026/08/27/llm-d-flow-control-priority-queuing-for-shared-gpu-inference>
17. https://docs.redhat.com/en/documentation/red_hat_openshift_ai_self-managed/3.5/html/deploy_models_using_distributed_inference_with_llm-d/managing-mixed-workloads-with-priority-queuing
18. https://docs.redhat.com/en/documentation/red_hat_ai_inference/3.5/html/manage_mixed_workloads_by_using_priority_queueing/managing-mixed-workloads-with-priority-queuing
19. <https://katacontainers.io/>
20. https://docs.redhat.com/en/documentation/openshift_container_platform/4.18/html/hosted_control_planes/networking-for-hosted-control-planes
21. <https://gpusmith.com/articles/en/mig-vs-mps-time-slicing-gpu-passthrough>
22. https://docs.redhat.com/en/documentation/red_hat_openshift_ai_self-managed/3.5/html/upgrading_openshift_ai_self-managed/overview-of-upgrading-openshift-ai-self-managed_upgrade
23. https://docs.redhat.com/en/documentation/red_hat_openshift_ai_self-managed/3.5/html/upgrading_openshift_ai_self-managed/requirements-for-upgrading-openshift-ai_upgrade
24. <https://rocm.docs.amd.com/en/docs-7.1.0/compatibility/compatibility-matrix.html>
25. <https://dgpu-docs.intel.com/overview/hardware-table.html>
26. <https://learn.microsoft.com/en-us/azure/aks/use-nvidia-gpu>
27. <https://docs.aws.amazon.com/eks/latest/userguide/ml-eks-optimized-ami.html>
28. https://docs.redhat.com/en/documentation/red_hat_openshift_ai_self-managed/3.5/html/upgrading_openshift_ai_self-managed_in_a_disconnected_environment/requirements-for-upgrading-openshift-ai_upgrade
29. <https://gpusmith.com/articles/en/air-gapped-llm-deployment-guide>
30. https://huggingface.co/docs/huggingface_hub/v0.24.5/en/package_reference/environment_variables
31. <https://docs.nvidia.com/datacenter/cloud-native/gpu-operator/latest/gpu-operator-mig.html>
32. <https://opentelemetry.io/blog/2026/genai-observability/>
33. <https://github.com/open-telemetry/semantic-conventions-genai/blob/main/docs/gen-ai/gen-ai-metrics.md>
34. <https://gpusmith.com/articles/en/dcgm-exporter-gpu-monitoring-metrics-alerts-triage>
35. <https://docs.python.org/3.14/library/time.html>
36. <https://prometheus.io/docs/practices/histograms/>
37. <https://www.finops.org/framework/capabilities/unit-economics/>

THE PUBLISHER

About GPU Smith

GPU Smith provides independent engineering and research for private AI infrastructure. We help technical buyers and operators reason about GPU workload sizing, hardware selection, procurement, deployment and inference operations, from the compute system to the networking, power and cooling requirements around it.

Start with the workload

Useful infrastructure decisions begin with the models, data, throughput, latency and operating constraints a team actually has. GPU Smith helps connect those requirements with system design and deployment choices. The goal is a privately controlled AI environment whose capacity and operational demands are understood before a purchasing decision.

Hardware and supplier research

Our public reference library covers GPUs, complete systems and networking components. The server-vendor directory and manufacturing research help teams investigate suppliers, compare options and follow sources behind technical claims. We distinguish manufacturer specifications from measured benchmarks and advertised capabilities from tested configurations. Reference pages are research resources, not a live inventory listing or a binding equipment quote.

Deployment and operations

GPU Smith's engineering scope includes infrastructure integration, networking, power, cooling and the practical operation of inference workloads. Our published articles explain the tradeoffs behind deployment, procurement and ongoing operations so teams can ask better questions and document decisions.

Work with GPU Smith

Explore the [hardware library](#), [GPU references](#), [networking references](#) and [vendor research](#). [Contact GPU Smith](#) with your workloads and deployment constraints to discuss a project and confirm current scope, pricing and availability.

Inclusion of a manufacturer or product in our research does not imply a partnership, certification or endorsement.

Website: <https://gpusmith.com>

This article is educational. Verify consequential medical, legal, financial, regulatory and technical decisions with appropriate professionals and the cited primary sources. Company services are described separately from the article's research findings.