GPU SMITH
COMPUTE ENGINEERING

GPU SMITH / RESEARCH & PRACTICAL GUIDANCE

ROCm 10 Upgrade Guide: Compatibility and Rollback

> 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 · 2026-09-19 · rocm 10 upgrade guide · rocm 10 compatibility · rocm migration · amd instinct · mi300 · mi350 · gpu infrastructure · production deployment · rollback planning

Original article: https://gpusmith.com/articles/en/rocm-10-production-upgrade-runbook

In this article

Executive Summary

ROCm 10.0.0 is not a routine user-space refresh. AMD dated the Core SDK release **August 26, 2026** (rocm.docs.amd.com), one day before its production-focused announcement highlighted validated vLLM and SGLang containers, Python wheels, and modular packages (www.amd.com). For operators, the consequential change is the number of independently moving layers: GPU architecture, host operating system and kernel, amdgpu driver, OEM firmware, virtualization mode, Core SDK packages, framework build, serving engine, model revision, and orchestration policy. AMD explicitly says **amdgpu 31.50.0** is compatible with ROCm 10.0.0 (instinct.docs.amd.com), but compatibility is a tuple, not a single version claim.

The go or no-go decision should therefore begin with an immutable current-state bill of materials and a candidate matrix. ROCm 10 maps **MI350 to gfx950, MI300 to gfx942, and MI200 to gfx90a** (rocm.docs.amd.com). Firmware floors differ by board: for example, AMD lists **01.26.01.03 or later for MI355X** (rocm.docs.amd.com). A green result exists only where every local field intersects an officially documented row.

Qualification should test application behavior, not merely installation. The release changes package names and paths, replaces ROCm SMI with AMD SMI, and advances AMD SMI to a breaking **27.0.0** major version (github.com). One MI350X issue documents **9 to 25 percent** lower Hugging Face training throughput in a specified AOTriton configuration (rocm.docs.amd.com). These are test inputs, not universal predictions.

The recommended production path is **freeze, intersect, canary, expand, and retain a separately qualified rollback image**. Pin containers by digest because Docker supports content-specific pulls (docs.docker.com); drain nodes without bypassing disruption budgets (kubernetes.io); measure correctness, cold start, tail latency, throughput, memory pressure, collectives, profiling, recovery, and soak behavior; and obtain OEM confirmation before any firmware downgrade. GPU Smith is an adjacent independent infrastructure advisor, not a ROCm vendor. Its published method centers written acceptance criteria and as-built records (gpusmith.com), which is the appropriate posture for this decision.

Introduction and Background

ROCm 10 arrives as several deliverables that can be upgraded together or separately. The **Core SDK** contains compiler, runtime, libraries, and tools. The **kernel driver and firmware bundle** establish device support below user space. Framework wheels and containers package PyTorch, JAX, vLLM, or SGLang around particular ROCm builds. The AI ecosystem pages then provide recipes and validated versions. AMD's overview lists **PyTorch 2.11.0 through 2.13.0, JAX 0.10.0 through 0.11.0, vLLM 0.27.0, and SGLang 0.5.15**, while warning that actual support varies by GPU (rocm.docs.amd.com).

That distinction matters commercially and operationally. A technical buyer may see "ROCm 10 supported" on a proposal, yet the quoted system may use an unlisted guest kernel, an older OEM firmware bundle, a mutable container tag, or a model path never exercised by the vendor. The upgrade record must preserve the exact tuple that was tested. Docker notes that digest pinning prevents automatic updates, including security updates (docs.docker.com); that is precisely why security patch review and application qualification must be separate controlled events.

The objective is not to prove ROCm 10 is universally faster. No public result can substitute for a local workload's model, precision, context length, batch pattern, concurrency, topology, and service-level objective. The objective is to produce evidence that a specific candidate is supported, correct, observable, performant enough, recoverable, and reversible within an organization-defined change window. NIST change guidance calls for testing both security and functional effects before approval (nvlpubs.nist.gov), then verifying that the approved change was implemented correctly (nvlpubs.nist.gov).

In search-intent terms, this is a **ROCm 10 compatibility matrix**, **ROCm 10 known issues** triage, **ROCm 10 migration guide**, **ROCm 10 system requirements** check, **ROCm 10 production deployment** plan, **ROCm 10 upgrade checklist**, **ROCm 10 rollback procedure**, and **ROCm 10 validation tests** protocol in one change-control workflow.

Key Changes in ROCm 10.0.0

A new package and component boundary

ROCm 10 consolidates legacy library packages. For example, the transition guide groups hipBLAS, rocBLAS, hipBLASLt, and hipSPARSELt under `amdrocm-blas`; it also changes prefixes and paths (rocm.docs.amd.com). This can break configuration management, container build stages, license or software bill of materials scanners, hard-coded paths, and package-presence monitoring even when application source is unchanged.

The operational response is a **BOM diff**, not a package upgrade command. Record package name, version, repository snapshot, file path, checksum, owning team, and dependent service for current and candidate states. TheRock notes that JAX plugin and PJRT package names encode their ROCm major version (github.com), illustrating why a Core SDK pin does not pin the full machine-learning environment.

Monitoring and developer interfaces move

ROCm SMI is removed in favor of AMD SMI, while the ROCm Validation Suite provides tests, benchmarks, and qualification tools (github.com). TransferBench separately records support for the AMD SMI **27.0.0** ABI change associated with ROCm 10 (github.com). Any exporter, health daemon, automation wrapper, or dynamically linked utility that expects the former command output, Python field, API type, or library ABI needs an explicit compatibility test.

AMD SMI's standard summary includes GPU status and running processes (rocm.docs.amd.com). Capture that output before and after the canary, alongside PCI identifiers, firmware, temperatures, power, error counters, partition state, and process ownership. Prometheus recommends recording total attempts with failures so a ratio can be calculated (prometheus.io); gauges suit state such as free memory or temperature (prometheus.io).

Validated frameworks are starting points

AMD publishes a ROCm 10 vLLM image tagged for **vLLM 0.27.0** (rocm.docs.amd.com) and an SGLang image tagged for **SGLang 0.5.15** (rocm.docs.amd.com). A tag is not an immutable identity. Resolve it to a registry digest, archive the manifest, and record base-image and wheel hashes. OCI annotations can carry the

packaged software's source-control revision (github.com), while SLSA provenance records where, when, and how an artifact was produced (slsa.dev).

Pre-canary impact questions should include:

- **Packages:** Which old names, repositories, and paths appear in automation or policy?
- **Linkage:** Which services load AMD SMI, HIP, RCCL, or profiler shared objects directly?
- **Bindings:** Which Python modules or fields are parsed by exporters and diagnostics?
- **Images:** Which base tags are mutable, and what are their resolved digests?
- **Profiles:** Which stored profiler datasets must remain readable after the upgrade?
- **Models:** Which model and custom-code revisions are actually approved?
- **Ownership:** Who can accept, mitigate, or block each detected incompatibility?

Build the Compatibility and Qualification Matrix

Freeze the current-state bill of materials

Inventory each production pool before changing it. The record should be machine generated where possible and reviewed by a human owner. At minimum, capture:

- **Hardware identity:** server SKU, GPU SKU, serial number, PCI address, fabric adapter, and topology.
- **Compilation target:** Record the exact LLVM target emitted by the versioned compatibility matrix, and retain the compiler command line with the build artifact.
- **Host:** distribution release, exact kernel, boot parameters, IOMMU state, and security modules.
- **Low-level stack:** amdgpu package, loaded module, OEM firmware or PLDM bundle, and partition mode.
- **User space:** every ROCm package, library SONAME, wheel, environment variable, and custom build flag.
- **Workload:** framework, engine, container digest, model revision, tokenizer, precision, and quantization.
- **Platform:** Kubernetes version, device plugin, scheduler rules, runtime, storage, and network fabric.
- **Virtualization:** bare metal, hypervisor build, passthrough or SR-IOV, guest OS, virtual functions, SPX/DPX/CPX, and NPS mode.

A CycloneDX record can represent a complete inventory of first-party and third-party components (cyclonedx.org); CISA similarly defines a software bill of materials as a formal record of component details and supply-chain relationships (www.cisa.gov). Store the current and candidate records together so approval examines a diff, not two disconnected inventories.

Resolve the full support intersection

Table 1 is a template populated with selected official ROCm 10.0.0 facts. "Verify" means the cited documentation does not, by itself, prove the rest of a local tuple.

Fleet	LLVM target	Illustrative documented host tuple	Driver and firmware checkpoint	Virtualization checkpoint	Local status
MI350 series	`gfx950`	Ubuntu 26.04 with GA 7.0 (rocm.docs.amd.com)	Confirm exact amdgpu and board-specific OEM firmware	AMD's MI350P guide says ESXi 9.1 was validated (instinct.docs.amd.com)	Verify
MI300 series	`gfx942`	Ubuntu 24.04.4 with GA 6.8 (rocm.docs.amd.com)	Confirm exact amdgpu and board-specific OEM firmware	MI300X passthrough is listed for ESXi 8 U3 with Ubuntu 24.04 or 22.04 guests (rocm.docs.amd.com)	Verify
MI200 series	`gfx90a`	RHEL 10.2 with kernel 6.12.0-211 (rocm.docs.amd.com)	Confirm exact amdgpu and board-specific OEM firmware	Do not infer MI300 or MI350 partition support; source the exact MI210, MI250, or MI250X row	Verify

The table is deliberately incomplete. An operator must add server model, exact amdgpu build, firmware, hypervisor, guest, partition mode, framework, image digest, and evidence URL for every pool. The release notes link SR-IOV configurations to **GIM 9.2.0.K** (rocm.docs.amd.com), and the GIM release validates different host and guest combinations by GPU. For example, its MI300X rows name Ubuntu 22.04 kernel 6.8 and Ubuntu 24.04 kernel 6.14 hosts (github.com). Generic "KVM supported" language is insufficient.

Qualification must preserve equally specific evidence below and above the matrix. RVS recommends GPU-specific rather than default test configurations (github.com); AMD's driver prerequisites describe DKMS building for installed kernels (instinct.docs.amd.com); the MI350P guide defines DirectPath I/O as PCI passthrough (instinct.docs.amd.com); and GIM publishes a distinct **RHEL 9.4 kernel 5.14.0-427** host row (github.com).

Use a deterministic decision rule

Classify each row as **green**, **amber**, or **red**:

- **Green:** Every field matches an official supported tuple, and every local acceptance test passes.
- **Amber:** Documentation is silent, an exception is temporary, or a workaround is required. Name the accepting owner and expiry date.
- **Red:** A field conflicts with the matrix, a required test fails, rollback is unqualified, or evidence is missing.
- **No inference:** Support for one GPU, guest, partition, framework, or model never transfers automatically to another.

This rule prevents marketing language from becoming production evidence. It also makes later minor-release review efficient: rerun only rows whose upstream facts or local artifacts changed.

Implementation Considerations and Process Changes

Choose delivery mode for rollback behavior

AMD documents **five installation methods** (rocm.docs.amd.com). The operational choice is about ownership, isolation, offline reproducibility, and reversal, not installer convenience.

Table 2 compares the documented modes with the controls an enterprise runbook should add.

Method	Documented scope or fit	Change-control strengths	Rollback and air-gap questions
Native package manager	System-wide DEB or RPM with dependency tracking (rocm.docs.amd.com)	Fits repository snapshots, configuration management, and package inventory	Mirror exact repositories, retain package set and signing metadata, test downgrade transactions and reboot
Runfile	Installs ROCm and can include the AMD GPU driver (rocm.docs.amd.com)	Self-contained artifact, selective components, custom location	Dependencies must exist before offline use (rocm.docs.amd.com); runfile does not remove native-package installs
Tarball	Raw files without installation configuration steps (github.com)	Easy immutable directory switch and artifact caching	Operator owns dependencies, paths, updates, removal, and environment isolation
Python wheels	ROCm libraries in a virtual environment for Python-only workflows (rocm.docs.amd.com)	Lockfile and wheel-hash reproducibility	Does not by itself reverse host driver, firmware, or separately distributed framework artifacts
amdgpu-install	Documented for Radeon and Ryzen in the comparison table	Bundles broad selection and post-install logic for its stated targets	Do not select for an Instinct fleet merely because the name includes amdgpu

The best fit is often two-layered: an immutable host image or snapshotted native repository for driver and system components, plus digest-pinned workload containers. Ubuntu describes rolling out package upgrades in stages across a large set (ubuntu.com). Historical-snapshot users should also check the latest available version (ubuntu.com).

Convert changes and known issues into tests

Build an impact register with component or API, old behavior, new behavior, affected repository or service, detection test, mitigation, owner, and decision. High-priority ROCm 10 items include:

- **AMD SMI:** Relink native consumers, retest Python bindings, parsers, exporters, and alerts against 27.0.0 (github.com).
- **Package consolidation:** Diff manifests, paths, build stages, allowlists, and vulnerability-scanner identities.
- **Profiler data:** Re-profile retained baselines because older workload directories are incompatible with the new `sysinfo` schema (rocm.docs.amd.com).
- **RCCL observability:** Test `ncclBroadcast` visibility because improved AllGatherV support affects the NCCL profiler path (rocm.docs.amd.com).
- **MI350X training:** Reproduce the exact Hugging Face model, AOTriton, precision, and kernel selection before accepting the documented workaround.
- **Hardware scoping:** Do not turn Radeon or Ryzen issues into Instinct blockers. Match the product family stated in each issue before assigning severity to an Instinct pool.

Execute the canary protocol

Use the same requests, datasets, seeds, model weights, and traffic shape against the current and candidate stack. Hugging Face recommends pinning custom model code to a specific revision (huggingface.co). PyTorch deterministic algorithms aim to return the same output on the same software and hardware (docs.pytorch.org), but PyTorch cautions that this setting alone does not guarantee reproducibility (docs.pytorch.org).

Run tests in this order:

1. **Health baseline:** Enumerate devices, topology, firmware, partition state, temperatures, power, errors, and running processes.
2. **Correctness:** Compare output tokens, logits or task metrics within a documented tolerance, plus deterministic unit and integration tests.
3. **Cold start:** Clear engine, kernel, model, and compilation caches; measure image pull, process start, model load, compilation, and first response.
4. **Warm service:** Measure throughput, time to first token, inter-token latency, and p50, p95, and p99 end-to-end latency by concurrency.
5. **Memory pressure:** Sweep batch, context, KV cache, sequence mix, and allocation churn through the planned operating envelope.
6. **Collectives:** Run RCCL tests and the application's real tensor or pipeline parallel path across the actual topology.
7. **Observability:** Confirm metrics, logs, traces, profiler capture, alert routing, and dashboards against candidate field names.
8. **Recovery:** Kill workers, restart pods, recycle a device, reboot a node, and verify scheduler and checkpoint behavior within policy.
9. **Soak:** Hold realistic mixed traffic long enough to expose thermal, memory, queueing, and error-counter trends.

vLLM warns that repeated serving tests can reuse prefix-cache prompts and inflate throughput (docs.vllm.ai); its sweep mode resets server caches between runs (docs.vllm.ai). SGLang's benchmark reports throughput, time to first token, inter-token latency, and per-request end-to-end latency (github.com). JAX users must separate first-run compilation cost and block asynchronous dispatch before stopping the timer (docs.jax.dev) (docs.jax.dev).

Roll out and roll back without hidden state

Start with one representative node per materially different tuple, not one convenient node for the whole estate. Kubernetes drain marks a node unschedulable (kubernetes.io) and waits for graceful termination (kubernetes.io). A PodDisruptionBudget limits simultaneous unavailability for replicated applications (kubernetes.io). Include scheduler capacity, storage detach, checkpoint, and request-drain time in the window.

The rollback package should contain:

- **Prior host artifact:** image, packages, repository metadata, keys, and checksums.
- **Prior workload artifact:** container digest, manifest, wheelhouse, model and tokenizer revision.

- **Configuration:** orchestration manifests, device-plugin settings, environment, secrets references, and dashboards.
- **Low-level plan:** OEM-approved firmware and driver downgrade or forward-recovery procedure.
- **Measured timing:** transfer, drain, uninstall or switch, reboot, health check, and service validation ranges.
- **Abort triggers:** correctness failure, tail-latency breach, throughput floor, GPU reset, error-counter threshold, telemetry gap, or recovery overrun.

Do not promise that user-space rollback makes driver or firmware rollback safe. AMD says OEMs or infrastructure providers distribute firmware packages (rocm.docs.amd.com). Obtain written support for the exact server, firmware, driver, and direction of change before the window.

Data Analysis and Evidence

The quantitative core is a **paired local comparison**, not a borrowed vendor percentage. *Table 3* defines the minimum results sheet. Each row should contain measured values for both current and candidate stacks, plus a predeclared threshold and decision.

Dimension	Required segmentation	Measured fields	Decision evidence
Correctness	Model revision, precision, quantization, prompt or dataset	Exact-match or task metric, tolerance, divergent cases	Pass only within the approved tolerance and seed policy
Serving	Model, context, input/output length, concurrency, cache state	Time to first token, inter-token latency, p50/p95/p99 latency, requests/s, tokens/s	Compare paired runs and confidence intervals; disclose warm or cold state
Training	Model, global and micro batch, sequence, optimizer, parallel strategy	Samples/s or tokens/s, step time, loss trajectory, HBM peak, job time	Treat every applicable documented regression issue as a scoped reproduction test, not an expected result
Collectives	Topology, ranks, message size, operation	Algorithm and bus bandwidth, latency, errors, timeout rate	PyTorch `all_reduce` should leave tensors bitwise identical across processes (docs.pytorch.org)
Reliability	Failure mode, duration, traffic mix	GPU errors, resets, retries, failed requests or jobs, recovery time	Zero unexplained resets; error ratio stays within declared service objective
Resources	Node type and workload phase	HBM, host RAM, CPU, network, storage, power, temperature	No sustained threshold breach or hidden capacity regression

The table separates inputs from results, preventing a throughput number from circulating without its configuration. It also forces measurement of failures as a ratio rather than an isolated count. For JAX, persistent compiled programs can survive on disk (docs.jax.dev), so cache deletion or preservation must be part of the recorded test condition.

Two simple calculations drive rollout planning. First:

Available production capacity during change = total qualified capacity minus nodes in canary or change minus reserved failure capacity.

All three inputs come from the operator. Capacity may be expressed as qualified nodes, GPUs, or measured workload units, but the units must remain consistent. The rollout batch is acceptable only if available capacity stays above forecast demand plus the organization's headroom policy.

Second, calculate rollback time as a measured range:

Rollback duration = drain + artifact transfer + stack switch or reinstall + reboot + health validation + workload validation.

Report the median and worst observed rehearsal, then set the change-window abort point early enough to finish the upper-bound rollback. NIST treats resumption time as organization-defined rather than universal (nvlpubs.nist.gov). Any article or vendor claim promising a generic recovery time would omit the dominant local variables.

The approval evidence pack should preserve:

- **Support matrix:** every tuple, source URL, source date, reviewer, and status.
- **BOM diff:** packages, SONAMEs, images, digests, wheels, firmware, models, and configuration.
- **Provenance:** signatures, attestations, source revision, SBOM, repository snapshot, and artifact hashes.
- **Impact register:** breaking changes, applicable known issues, tests, mitigations, owners, and expiry dates.
- **Raw results:** commands, datasets, seeds, timestamps, metrics, logs, traces, and profiler outputs.
- **Decision record:** thresholds, exceptions, capacity model, rollout batches, abort triggers, and approvers.
- **Rollback proof:** cached artifacts, OEM position, rehearsal logs, measured duration range, and operator checklist.

Cosign verifies by default that a signed payload digest matches the checked container (docs.sigstore.dev). That makes signature verification useful, but it does not prove workload correctness or performance. Supply-chain identity and runtime qualification are complementary controls.

Evidence crosswalk for production approval

The following crosswalk prevents a single green benchmark from standing in for the whole decision. Each line connects an approval question to independently preserved evidence:

- **Change boundary:** Record what was authorized, how the artifact was produced, and which components it contains. NIST calls for security and functional impact testing (nvlpubs.nist.gov); SLSA defines production provenance (slsa.dev); CISA defines the SBOM record (www.cisa.gov); and CycloneDX supplies a component inventory model (cyclonedx.org).
- **Image identity:** Preserve the pulled digest, source revision, signature result, and update decision. Docker supports digest pulls (docs.docker.com) but warns that pinning suppresses automatic updates (docs.docker.com); OCI defines a revision annotation (github.com); and Cosign checks digest correspondence (docs.sigstore.dev).

- **Safe node withdrawal:** Preserve cordon time, eviction results, disruption-budget status, and capacity headroom. Drain makes the node unschedulable (kubernetes.io), waits for graceful termination (kubernetes.io), and can bypass disruption checks if forced (kubernetes.io); staged package snapshots support controlled rollout (ubuntu.com).
- **Offline completeness:** Preserve repository snapshots, dependencies, keys, wheelhouse, images, model files, and current security deltas. Ubuntu supports staged package rollout (ubuntu.com) while urging snapshot users to check the latest available version (ubuntu.com); CycloneDX covers first- and third-party components (cyclonedx.org); and CISA describes component relationships (www.cisa.gov).
- **Metric integrity:** Preserve numerator, denominator, state gauges, cache condition, and workload configuration. Prometheus recommends a total-attempt metric with failure counts (prometheus.io) and uses gauges for state snapshots (prometheus.io); vLLM warns of prefix-cache inflation (docs.vllm.ai) and provides cache-reset sweeps (docs.vllm.ai).
- **Correctness boundary:** Preserve hardware and software identity, model revision, deterministic settings, seeds, tolerance, and divergent cases. PyTorch defines deterministic output for the same software and hardware (docs.pytorch.org) but says that switch alone is insufficient (docs.pytorch.org); completed `all_reduce` tensors should be bitwise identical (docs.pytorch.org); and Hugging Face recommends a specific code revision (huggingface.co).
- **Timing validity:** Preserve compile state, synchronization point, and persistent-cache state. JAX says first execution is slower (docs.jax.dev), requires `block_until_ready()` for completed timing (docs.jax.dev), and can persist compiled programs (docs.jax.dev); vLLM identifies cached-prompt inflation (docs.vllm.ai).
- **Serving comparability:** Preserve request distribution and report latency components, not only aggregate throughput. SGLang measures throughput, time to first token, inter-token latency, and end-to-end latency (github.com); vLLM defines per-request time per output token without the first token (docs.vllm.ai) and resets caches between sweeps (docs.vllm.ai); Prometheus gauges retain point-in-time state context (prometheus.io).
- **Collective validation:** Preserve rank count, tensor shapes, operation, message-size sweep, topology, and GPU-specific test configuration. PyTorch expects bitwise-identical `all_reduce` outputs (docs.pytorch.org) but notes that coalesced collectives lack individual shape checking across nodes (docs.pytorch.org); RVS describes a test and qualification collection (github.com) and recommends GPU-specific configurations (github.com).
- **Hardware identity:** Preserve accelerator family, architecture target, driver construction, and container digest. AMD associates MI350 with fourth-generation CDNA (www.amd.com), MI300 with CDNA 3 (www.amd.com), and MI200 with CDNA 2 (www.amd.com); AMD's driver prerequisites say DKMS builds the module for installed kernels (instinct.docs.amd.com).
- **Virtualization identity:** Preserve assignment method, validated hypervisor, host kernel, guest release, and virtual-function layout. AMD describes MI350P DirectPath I/O as PCI passthrough (instinct.docs.amd.com) and validates ESXi 9.1 (instinct.docs.amd.com); GIM documents MI300X Ubuntu hosts (github.com) and a RHEL 9.4 kernel 5.14.0-427 host row (github.com).
- **Approval and handoff:** Preserve the implemented-change check, recovery objective, written acceptance criteria, and operating record. NIST requires post-implementation verification (nvlpubs.nist.gov) and makes recovery timing organization-defined (nvlpubs.nist.gov); GPU Smith states

that work is delivered against written acceptance criteria (gpusmith.com) and identifies telemetry and operating procedures as infrastructure disciplines (gpusmith.com).

Implications and Future Directions

ROCm 10's modularity can improve change isolation only when operators preserve boundaries. Separate Core SDK, driver, firmware, framework, engine, and model artifacts allow a team to identify which layer changed and to qualify smaller deltas. The opposite outcome is also possible: independently updated wheels, tags, and host packages can create combinations no one tested. Immutability, provenance, and a support-intersection matrix determine which outcome prevails.

Recurring review should be event-driven. Trigger a focused requalification when AMD edits the versioned release record, an OEM publishes a firmware bundle, a security patch changes a pinned base, a framework or engine image digest changes, a model revision moves, or orchestration changes device exposure. Do not silently absorb mutable tags. At the same time, digest pinning should not become permanent patch avoidance, because pinned images do not automatically receive security updates.

GPU Smith's adjacent-advisor perspective is appropriately centered on acceptance evidence, capacity, telemetry, and operating procedures. Its site describes capacity planning, telemetry, failure-mode analysis, and operating procedures for existing facilities (gpusmith.com). For a private-AI owner, the durable asset is not a one-time install. It is a reproducible evidence set that lets SRE, security, platform, application, procurement, and OEM stakeholders reach the same decision.

Frequently Asked Questions (FAQs)

Is ROCm 10.0.0 compatible with MI200, MI300, and MI350?

All three families appear in the official matrix, but that fact is not sufficient for a production yes. Confirm the exact SKU, LLVM target, operating system and kernel, amdgpu build, firmware, virtualization and partition mode, guest, framework, and container. AMD identifies MI350 as fourth-generation CDNA (www.amd.com), MI300 as CDNA 3 (www.amd.com), and MI200 as the family underlying CDNA 2 (www.amd.com).

What are the main ROCm 10 breaking changes?

The operationally broad changes are consolidated package names and paths, replacement of ROCm SMI with AMD SMI, the AMD SMI 27.0.0 ABI change (github.com), removed bandwidth tooling, changed profiler data compatibility, and framework-specific API or behavior changes. The correct inventory is repository-specific: search code, images, automation, dashboards, and native linkage, then map every hit to a test and owner.

Which ROCm 10 known issues should block deployment?

Only applicable issues should influence the decision. Match each issue's hardware, software, feature, and failure mode to the local tuple. A correctness failure, unexplained reset, missing telemetry, or breached service objective should block expansion. A documented issue outside the local GPU family should not. Each accepted workaround needs an owner and retest trigger.

Should production use vLLM or SGLang containers from AMD?

They are useful qualification starting points because AMD publishes validated version recipes, but the operator must resolve the tag to a digest and test the actual model, precision, context, concurrency, cache state, and topology. The SGLang and vLLM benchmark metrics help structure the measurement, not replace it.

What is the safest ROCm 10 rollback procedure?

There is no universal command. Preserve the prior host and workload artifacts, rehearse the exact reverse path, measure every phase, and define abort triggers. Treat driver and firmware separately from user space. Any firmware or driver downgrade needs server-OEM and AMD confirmation for that exact tuple.

How should an air-gapped cluster upgrade?

Mirror or stage every package, dependency, image by digest, wheel, model, signature, key, SBOM, and documentation snapshot before the window. The runfile supports offline use after dependencies are installed, and its dependencies can use an air-gapped source or mirror (rocm.docs.amd.com). Verify the offline bundle on an isolated staging node, including signature, checksum, install, reboot, validation, and rollback.

What belongs in the final production approval?

Include the support-intersection matrix, signed BOM diff, source snapshots, issue register, raw canary results, exception approvals, capacity calculation, rollout batches, abort thresholds, OEM downgrade position, rollback rehearsal, and measured recovery range. Approval should identify the exact artifact digests and the date after which upstream changes require review.

Conclusion

The defensible ROCm 10 upgrade is a controlled evidence program, not a generic installation tutorial. Freeze the current state, construct the complete support tuple, choose a delivery method whose reversal is understood, translate each breaking change and applicable known issue into a test, and canary every materially different hardware and deployment class.

Production approval should require four results: **documented support, workload correctness, acceptable measured service behavior, and a rehearsed rollback within the local change window**. The MI200, MI300, and MI350 families cannot share a single inherited verdict because their architecture targets, operating-system rows, firmware, and virtualization combinations differ. Likewise, a validated framework version or vendor container tag does not prove a particular model, digest, or traffic profile.

The lasting output is the evidence pack: versioned sources, immutable artifacts, support matrix, BOM diff, impact register, raw test results, capacity model, rollout record, and rollback proof. That package turns later security patches and minor releases into bounded reviews rather than fresh investigations. It also gives engineering, SRE, security, procurement, and OEM stakeholders a common basis for a go, conditional go, or no-go decision.

Source links

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

1. https://rocm.docs.amd.com/en/docs-10.0.0/about/release-notes.html
2. https://www.amd.com/en/blogs/2026/amd-rocm-10-a-simpler-path-to-production-ai-on-amd.html
3. https://instinct.docs.amd.com/projects/amdgpu-docs/en/develop/compatibility/compatibility-matrix.html
4. https://rocm.docs.amd.com/en/docs-10.0.0/compatibility/compatibility-matrix.html
5. https://github.com/ROCm/TransferBench/releases
6. https://docs.docker.com/reference/cli/docker/image/pull/
7. https://kubernetes.io/docs/reference/kubectl/generated/kubectl_drain/
8. https://gpusmith.com/
9. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-128.pdf
10. https://rocm.docs.amd.com/en/docs-10.0.0/about/transition-guide-TheRock.html
11. https://github.com/ROCm/TheRock/blob/main/RELEASES.md
12. https://github.com/ROCm/ROCmValidationSuite
13. https://rocm.docs.amd.com/projects/amdsmi/en/develop/how-to/amdsmi-cli-tool.html
14. https://prometheus.io/docs/practices/instrumentation/
15. https://rocm.docs.amd.com/projects/ai-ecosystem/en/latest/inference/vllm.html
16. https://rocm.docs.amd.com/projects/ai-ecosystem/en/latest/inference/sglang.html
17. https://github.com/opencontainers/image-spec/blob/main/annotations.md
18. https://slsa.dev/spec/v1.2/provenance
19. https://cyclonedx.org/specification/overview/
20. https://www.cisa.gov/topics/cyber-threats-and-advisories/sbom/sbomresourceslibrary
21. https://instinct.docs.amd.com/projects/amdgpu-docs/en/latest/virtualization/esxi/mi350p/index.html
22. https://github.com/amd/MxGPU-Virtualization/releases/tag/9.2.0.K
23. https://instinct.docs.amd.com/projects/amdgpu-docs/en/latest/install/detailed-install/prerequisites.html
24. https://gpusmith.com/articles/en/amd-mi350p-retrofit-guide
25. https://rocm.docs.amd.com/en/docs-10.0.0/install/rocm.html
26. https://rocm.docs.amd.com/en/docs-10.0.0/install/rocm-runfile-installer.html
27. https://ubuntu.com/server/docs/how-to/software/snapshot-service/
28. https://rocm.docs.amd.com/en/docs-10.0.0/release/changelog.html
29. https://huggingface.co/docs/transformers//models
30. https://docs.pytorch.org/docs/main/generated/torch.use_deterministic_algorithms.html
31. https://docs.vllm.ai/en/latest/benchmarking/cli/
32. https://github.com/sgl-project/sglang/blob/main/docs/docs/developer_guide/bench_serving.mdx
33. https://docs.jax.dev/en/latest/201/profiling.html
34. https://kubernetes.io/docs/concepts/workloads/pods/disruptions/

35. https://docs.pytorch.org/docs/stable/distributed
36. https://docs.jax.dev/en/latest/501/compilation-cache.html
37. https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-34r1.pdf
38. https://docs.sigstore.dev/cosign/verifying/verify/
39. https://www.amd.com/en/products/accelerators/instinct/mi350.html
40. https://www.amd.com/en/products/accelerators/instinct/mi300.html
41. https://www.amd.com/en/technologies/cdna.html

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