GPU SMITH
COMPUTE ENGINEERING

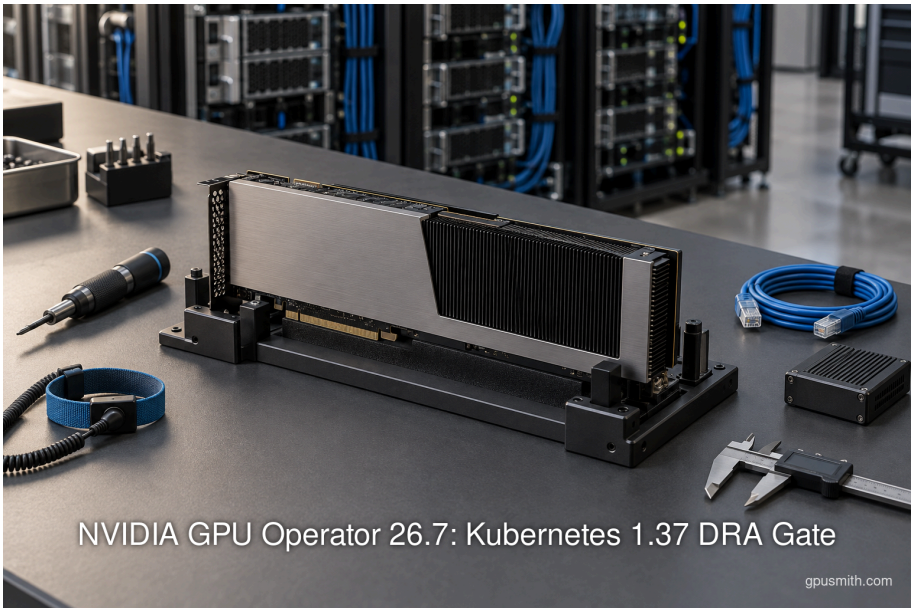
GPU SMITH / RESEARCH & PRACTICAL GUIDANCE

NVIDIA GPU Operator 26.7: Kubernetes 1.37 DRA Gate

> 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 · nvidia gpu operator 26.7 · kubernetes 1.37 · dynamic resource allocation · nvidia dra driver · gpu kubernetes · containerd · vfio passthrough · private gpu cluster · kubernetes upgrade

Original article: https://gpusmith.com/articles/en/gpu-operator-kubernetes-dra-upgrade-gate



In this article

Executive Summary

NVIDIA GPU Operator 26.7.0, released on **August 21, 2026**, is the first decision point in this guide, not an automatic upgrade target (github.com). It adds management of **DRA Driver for NVIDIA GPUs v0.5.0**, retains **NVIDIA Device Plugin v0.20.0**, validates upstream Kubernetes through **1.37**, and corrects the general minimum supported containerd version from 1.8 to **2.0** (docs.nvidia.com) (docs.nvidia.com) (docs.nvidia.com) (docs.nvidia.com). The practical recommendation for a private cluster on 26.3 or earlier is therefore two decisions: qualify the operator upgrade first, then qualify DRA separately.

Kubernetes 1.37 makes DRA-backed extended resources generally available and enabled by default, so existing requests such as `nvidia.com/gpu: 1` can keep their Pod shape only after a separate `DeviceClass` maps `spec.extendedResourceName` to that legacy name (kubernetes.io) (kubernetes.io) (docs.nvidia.com). That does not make every 1.37 DRA feature production-ready. The maturity and default state of each capability still need separate review, as Table 2 shows.

The hard gate is resource ownership. GPU Operator uses either a `ClusterPolicy` for the Device Plugin path or a `GPUCluster` for the managed DRA path, not both, and NVIDIA does not support an in-place migration between them (docs.nvidia.com) (docs.nvidia.com). Treat backend separation as a node-pool migration primitive, not proof that the two GPU Operator APIs can coexist.

The recommended production gate is a disposable DRA canary pool, explicit runtime and Container Device Interface checks, whole-GPU claim-lifecycle tests, then a drain and reschedule exercise. A VFIO test requires an explicitly accepted Alpha `PassthroughSupport` enablement, which is disabled by default, and must not be treated as a standard production path (docs.nvidia.com). Do not promote DRA until claim success, p95 allocation latency, NUMA placement, taint behavior, restarts, drains, failed reschedules, and rollback time all meet written local thresholds. The decision record should preserve exact inputs, attempted and successful claims, every failed reschedule, the measured rollback interval, named owners, and the expiry of accepted exceptions.

Introduction and Background

Dynamic Resource Allocation (DRA) moves specialized-device selection into Kubernetes APIs built around `DeviceClass`, `ResourceSlice`, `ResourceClaim`, and driver-mediated preparation. The older Device Plugin model advertises integer extended resources and lets the kubelet pass allocated devices into containers. DRA can express richer attributes, shareable capacity, topology, device health, and claim lifecycle. For a private GPU estate, that broader contract is useful only if it remains predictable through node replacement, runtime restart, workload restart, and rollback.

This report is a **version-locked acceptance guide** for platform engineers and site reliability engineers. It assumes the current production cluster is on GPU Operator 26.3 or earlier and uses extended resources, and that the target is GPU Operator 26.7 with Kubernetes 1.37. It is not an installation tutorial. The central question is whether 26.7 and DRA can pass a production gate with the actual GPU models, runtime configuration, topology, passthrough requirements, and disruption budgets of the cluster.

GPU Smith is adjacent to the products assessed here, not a GPU Operator, Kubernetes, or cloud-platform provider. Its published method describes integration and validation against written acceptance criteria, and its operations scope includes telemetry, failure-mode analysis, and operating procedures (gpusmith.com) (gpusmith.com). That posture informs the process below: vendor compatibility is an entry condition, while measured acceptance on representative hardware is the release decision.

Managed-service status must be checked separately. On the publication date, Google Kubernetes Engine described DRA device metadata publishing as beta, Azure Kubernetes Service scheduled 1.37 general availability for **October 2026**, Amazon Elastic Kubernetes Service listed versions only through 1.36, Red Hat OpenShift Container Platform 4.22 used Kubernetes 1.35, and DigitalOcean Kubernetes listed 1.36 as its newest supported release (docs.cloud.google.com) (learn.microsoft.com) (docs.redhat.com) (docs.digitalocean.com). A vendor matrix saying a combination is supportable is not evidence that a managed control plane currently offers it.

Key Changes

GPU Operator 26.7 changes the supported baseline

The final 26.7.0 artifact is a defined release, not a rolling `latest` target. NVIDIA NGC records its Helm chart at **125.25 KB** with an August 21 publication timestamp (catalog.ngc.nvidia.com). Pin the chart and image digests used in the canary rather than allowing tags to drift between test and promotion.

The component baseline includes the Device Plugin **v0.20.0**, DRA Driver **v0.5.0**, and supported driver branches **610.57.04, 595.91.07, 580.173.02, and 535.309.01**, with 595.91.07 marked recommended and default in the matrix (docs.nvidia.com). The DRA v0.5.0 release explicitly supports deployment through GPU Operator 26.7.0 (github.com). The managed `GPUCluster` workflow raises the effective DRA baseline further: Kubernetes **1.34.2 or later**, an NVIDIA driver **580 or later**, and a CDI-compatible runtime (docs.nvidia.com).

Containerd needs careful reading. NVIDIA validates **containerd 2.0 through 2.3**, but containerd itself recommends **2.3.0 or later or 2.4.0 or later** for Kubernetes 1.37 and says its matrix reflects the most thoroughly tested combinations (containerd.io) (containerd.io). The safe common set is therefore containerd 2.3.x, unless a distribution certifies another point release. Containerd 2.3 is also a long-term-support branch through **April 30, 2028**, and direct upgrades are supported between sequential long-term-support releases (github.com) (github.com).

Table 1 turns the vendor and upstream statements into a before-and-after prerequisite matrix.

Layer	26.3 or earlier baseline	26.7 qualification target	Acceptance interpretation
Kubernetes	Existing supported minor, commonly extended-resource Device Plugin	Upstream **1.37**, or a distribution release that explicitly ships it	Do not infer managed-service support from upstream status. AKS rollout can take up to **10 business days** across regions (learn.microsoft.com).
GPU allocation	Device Plugin via `ClusterPolicy`	DRA v0.5.0 via `GPUCluster`, or Device Plugin v0.20.0	Choose one operator-managed API model. No in-place `ClusterPolicy` to `GPUCluster` migration.

Layer	26.3 or earlier baseline	26.7 qualification target	Acceptance interpretation
NVIDIA driver	Site-specific supported branch	**580+** for managed DRA, matrix default **595.91.07**	Preserve the exact driver, firmware, GPU model, and kernel combination tested.
Runtime	Existing CRI runtime	NVIDIA supports containerd **2.0 to 2.3**; upstream recommendation favors **2.3+** for 1.37	Confirm Container Runtime Interface v1 and the distribution's supported point release.
Device injection	NVIDIA runtime integration, possibly CDI	CDI-compatible runtime and discoverable specifications	The CDI project uses `/etc/cdi` and `/var/run/cdi` as default directories (github.com).
Upgrade artifact	Existing Helm revision and CRDs	Pinned 26.7.0 chart, saved values, manifests, CRDs, image digests	Apply both ComputeDomain custom resource definitions before a `GPUCluster` upgrade, even if the feature is disabled (docs.nvidia.com)

The table exposes two different meanings of “compatible.” NVIDIA's range establishes a supported product envelope. The runtime project's recommendation identifies combinations most thoroughly tested with Kubernetes 1.37. A production gate should satisfy both where possible and record any distribution-specific deviation.

Kubernetes 1.37 makes a stable bridge, not a fully stable DRA stack

DRA-backed extended resources are the migration bridge. A workload can continue requesting a familiar extended resource while allocation comes from DRA. Upstream permits the same resource name to be provided by the Device Plugin on one node pool and DRA on another, but each node has only one backend for that name (kubernetes.io). This supports a controlled pool-by-pool transition if the operator topology can enforce it.

Kubernetes 1.37 standardizes a shared NUMA location, and NVIDIA DRA Driver v0.5.0 publishes standard NUMA attributes for GPUs and adds consumable-capacity sharing across namespaces (github.com) (github.com). Those features are useful for topology-aware and partitioned workloads, but each adds an acceptance dimension.

Table 2 identifies the Kubernetes 1.37 feature state that should govern enablement decisions.

Capability	1.37 state and default	Production decision
DRA extended resources	**Stable, enabled**	Use as the lowest-change workload bridge after node-pool validation.
ResourceClaim device status	**Stable, enabled**	Collect allocation, preparation, health, and deallocation evidence.
Device taints and tolerations	**Stable, locked on**	Test `NoSchedule` and `NoExecute`; `NoExecute` can evict Pods already using the device (kubernetes.io).
Workload or PodGroup ResourceClaims	**Beta, disabled**	Leave off for the first migration unless a demonstrated multi-Pod requirement justifies it.

Capability	1.37 state and default	Production decision
Device metadata in containers	NVIDIA `DeviceMetadata`: **Alpha, disabled by default**	Keep disabled unless its required Alpha dependency, `PassthroughSupport`, has been explicitly accepted and validated.
Consumable capacity	NVIDIA `ConsumableShares`: **Alpha, disabled by default**	Treat sharing as a separately accepted Alpha qualification track, not part of basic whole-GPU migration.
Fractional capacity range	**Beta, compatibility-sensitive default**	Set explicitly on API server and scheduler rather than relying on an inherited compatibility version (kubernetes.io).
Compatibility groups	**Alpha, disabled**	Do not make this a production dependency in the first migration.
Optional node operations	**Alpha, disabled**	Keep disabled in the initial canary.

The stable items support whole-GPU production allocation. The beta items need explicit ownership and rollback. The alpha items belong in an isolated experiment unless the organization accepts feature-gate churn and maintains the relevant control-plane configuration.

VFIO passthrough validation

DRA Driver v0.5.0 adds passthrough support for NVIDIA Grace and Blackwell systems, including VFIO variants (github.com).

Implementation Considerations and Process Changes

Choose the resource model before changing the cluster

Three operating models are available:

- **Remain on Device Plugin extended resources:** choose this when whole-GPU allocation is sufficient, existing scheduling is stable, or the target distribution has not shipped a qualified 1.37 release. GPU Operator 26.7 can still update the Device Plugin without enabling DRA.
- **Move a dedicated cluster to managed DRA:** choose this when richer claims, standard NUMA attributes, passthrough, or consumable capacity justify a new resource-management contract. This path uses `GPUCluster` and should be treated as a controlled migration, not an in-place toggle.
- **Stage coexistence at infrastructure boundaries:** keep a production Device Plugin cluster or pool while creating a separate DRA canary cluster or strictly isolated pool. Upstream supports per-node backend separation, and Amazon EKS likewise states that DRA drivers and device plugins for one device type must not run simultaneously (docs.aws.amazon.com). DigitalOcean imposes the same same-brand exclusion in its public-preview implementation (docs.digitalocean.com).

Do not describe the third option as ordinary GPU Operator coexistence. The operator's mutually exclusive custom resources remain the controlling constraint. A second cluster is the cleanest boundary. A single-cluster staged pool requires an architecture that prevents both operator-managed backends from claiming the same node and may fall outside the documented managed workflow.

Upgrade 26.7 before enabling DRA

The upgrade sequence should separate software change from allocation-model change:

1. **Inventory:** record Kubernetes, kernel, containerd, driver, firmware, GPU model, Multi-Instance GPU profile, chart, images, CRDs, `ClusterPolicy`, RuntimeClass, CDI directories, labels, taints, disruption budgets, and workload manifests.
2. **Resolve supported hops:** NVIDIA supports upgrades within a major release or to the next major release only (docs.nvidia.com). A cluster earlier than the immediately preceding major release needs explicit intermediate hops.
3. **Take control-plane rollback artifacts:** save an etcd snapshot from a live member and verify its hash and revision (etcd.io) (etcd.io). All restored members must use the same snapshot (etcd.io). Managed services need the provider's control-plane recovery procedure instead.
4. **Record Helm state:** save `helm history`, the deployed values, rendered manifests, chart artifact, and image digests. Helm history supplies the revisions needed to name a rollback target (helm.sh).
5. **Prepare CRDs:** apply the required ComputeDomain definitions before the upgrade. Helm does not install CRDs during upgrade or rollback, so a chart rollback alone cannot restore their previous schema (helm.sh).
6. **Canary the operator with Device Plugin mode unchanged:** upgrade a non-production cluster or representative node pool. Exercise existing workloads before introducing `GPUCluster` or ResourceClaims.
7. **Create a separate DRA canary:** reproduce production hardware, runtime, topology, admission, policy, and monitoring. Confirm CDI discovery. If legacy `nvidia.com/gpu` manifests must be retained, create and validate a separate `DeviceClass` with `spec.extendedResourceName` set to that value, and retain that object in the rollback artifacts (docs.nvidia.com). Containerd configuration changes require a runtime restart before taking effect (github.com).
8. **Promote only after acceptance:** require written evidence for every gate in the next section. Do not merge operator qualification and DRA qualification into one change window.

Kubernetes minor upgrades require draining Pods from a node before upgrading kubelet, and kubelet can lag a 1.37 API server at 1.36, 1.35, or 1.34 during a staged rollout (kubernetes.io) (kubernetes.io). Use that skew window to replace nodes in waves, not to leave a mixed fleet indefinitely.

Acceptance and Canary Gates

Acceptance should cross resource mode, workload type, and disruption behavior. A happy-path CUDA Pod is insufficient because it does not test topology, device health, tainting, passthrough, or lifecycle cleanup.

Table 3 is a canary worksheet. The “result” cells must contain measured values from the target environment before promotion.

Gate	Test and evidence	Pass criterion	Measured result
Extended-resource control	Schedule existing `nvidia.com/gpu` workload in Device Plugin mode before and after 26.7 upgrade	Same requested GPU count, device visibility, health, and completion	Not measured in this report
DRA whole GPU	Create, bind, prepare, use, and release ResourceClaims	**successful claims / attempted claims = 100%** in the planned canary set; zero stale allocations	Enter count and ratio
Allocation latency	Measure claim-handling histogram from request through preparation	Locally approved **p95**, with no regression beyond the predeclared budget	Enter p50, p95, p99
NUMA placement	Compare allocated GPU `resource.kubernetes.io/numaNode` with CPU and network locality	Placement matches the workload's topology policy in every topology test	Enter pass count
Taints and tolerations	Apply DRA `NoSchedule` and `NoExecute` device taints	Untolerated new Pods do not schedule; eviction matches the declared policy	Enter events and evictions
VFIO serial	Run container then VFIO and reverse after preparation completion	Zero ownership overlap, correct binding, clean release	Enter attempts and failures

| Restart | Restart workload and DRA node plugin separately | Prepared claims recover as designed; zero leaked assignments | Enter recovery time | | Drain and reschedule | Cordon, drain, replace or reboot, uncordon, reschedule | **failed reschedules = 0**; PDB respected; device health restored | Enter failures and duration | | Rollback | Revert chart, allocation model, runtime configuration, and canary nodes | **rollback time** within the recovery-time objective; prior workloads pass control suite | Enter minutes and checks |

The universal criteria are correctness and cleanup. Latency and rollback thresholds are local because neither Kubernetes nor Prometheus prescribes one p95 value for every GPU cluster. Kubernetes exposes a histogram for the duration of handling all ResourceClaims during Pod start and stop (kubernetes.io). Prometheus `histogram_quantile` can compute a requested quantile, and counter-safe aggregation applies `rate()` before aggregation (prometheus.io) (prometheus.io).

The test log should preserve:

- **Versions:** full control-plane, kubelet, chart, operator, driver, DRA driver, runtime, kernel, firmware, and BIOS identifiers.
- **Topology:** node, socket, NUMA node, Peripheral Component Interconnect address, IOMMU group, GPU universally unique identifier, Multi-Instance GPU profile, and network device.
- **Claims:** creation, allocation, preparation, Pod start, deallocation, and deletion timestamps plus selected devices.
- **Disruptions:** Pod restart, plugin restart, runtime restart, node reboot, cordon, drain, replacement, and control-plane upgrade.

- **Outcomes:** attempted claims, successful claims, allocation error class, p50, p95, p99, failed reschedules, leaked resources, evictions, and rollback minutes.
- **Decision:** named owner, accepted exceptions, expiry date, rollback trigger, and promotion approval.

ResourceClaim device status is stable and enabled in 1.37, and kubelet uses a **30-second** device-health timeout if the driver does not provide one (kubernetes.io). Include both the status object and kubelet events in the evidence package.

Data Analysis and Evidence

The quantitative evidence supports a gate, not a universal performance claim. The upstream DRA documentation reports a scale test with **100 nodes**, **720 long-lived Pods**, and **80 churn Pods**. It reports that kube-controller-manager settings as low as **75 queries per second** and a **150 burst** met the non-DRA metric targets in that scenario (kubernetes.io) (kubernetes.io). Those figures show that DRA has been exercised at material scale. They are not a latency guarantee for a GPU driver, a particular API-server datastore, or a cluster with different churn.

The report's local calculation model should be explicit:

- **Claim success rate** = successful claims divided by attempted claims, multiplied by 100.
- **p95 allocation latency** = the 0.95 quantile over the selected ResourceClaim-handling histogram and test interval.
- **Failed reschedule rate** = failed reschedules divided by drain-triggered reschedule attempts, multiplied by 100.
- **Rollback duration** = time from the rollback decision to successful completion of the entire pre-upgrade control suite.
- **Cleanup defect rate** = claims or device bindings remaining after their expected deletion divided by completed claim lifecycles.

Each numerator and denominator must be retained. A “99% successful” statement is unhelpful without knowing whether it represents 99 of 100 attempts or 9,900 of 10,000.

Distribution timing is another quantitative risk. GKE release notes describe DRA device metadata publishing as beta (docs.cloud.google.com). AKS scheduled general availability for October and warns that a new version can take **10 business days** to reach all regions. EKS supports DRA from Kubernetes **1.33 and later**, but its current version list did not yet expose 1.37 on the research date (docs.aws.amazon.com). Oracle Kubernetes Engine enables DRA APIs by default from Kubernetes **1.34**, while DigitalOcean labeled GPU DRA a public preview (docs.oracle.com) (docs.digitalocean.com). These differences are why upstream 1.37 status cannot stand in for distribution support.

Rollback also has measurable components. Helm can automatically roll back chart changes after a failed atomic upgrade, and its rollback command can wait for Pods, persistent volume claims, Services, and workload readiness (v3.helm.sh) (docs.helm.sh). Yet CRDs, node runtime configuration, driver state, and

control-plane data require separate artifacts. Etcd restoration creates a new logical cluster and, for Kubernetes, recommends a revision bump to invalidate stale controller caches (etcd.io) (etcd.io). A credible rollback-time measurement therefore ends only when the prior workload suite passes, not when `helm rollback` exits.

Implications and Future Directions

GPU Operator 26.7 and Kubernetes 1.37 make DRA more practical, but they shift operational responsibility from a single integer resource to a lifecycle spanning scheduler allocation, driver preparation, runtime injection, health, deallocation, and topology. The stable extended-resource bridge reduces application-manifest churn. It does not reduce the need to test each lifecycle transition.

For clusters that only need whole GPUs, upgrading GPU Operator while retaining the Device Plugin is a rational endpoint. It captures the newer component baseline without accepting a new resource API. For clusters that need claim status, cross-driver topology, passthrough, or sharing, DRA deserves a dedicated canary and a new operational runbook.

The support-lifecycle decision also matters. NVIDIA's policy moves the previous major release into deprecated support with patches limited to critical fixes when a new major version ships (docs.nvidia.com). Remaining indefinitely on 26.3 has a maintenance cost, but that cost does not justify combining the operator, Kubernetes, runtime, driver, CRDs, and allocation model into one change.

Future 1.37 features should be adopted according to maturity:

- **Use stable foundations now:** extended-resource mapping, device status, and device taints can enter production after local acceptance.
- **Pilot beta behavior deliberately:** workload claims, metadata publishing, and consumable capacity need owners, explicit flags, workload demand, and rollback tests.
- **Keep alpha dependencies isolated:** compatibility groups, optional node operations, derived attributes, and resource-availability visibility should not become hidden production requirements.
- **Recheck managed distributions:** availability, feature defaults, runtime versions, and support policies may lag or differ from upstream. GKE Rapid availability is not equivalent to a standard or stable channel, and a vendor's roadmap date is not regional availability.

The most durable architecture is one that can replace a DRA node pool without rewriting workloads, retain a Device Plugin fallback during a migration program, and reconstruct every feature gate and runtime setting from version-controlled configuration. Managed rollback semantics also differ: Amazon EKS permits a control-plane minor rollback only within **seven days** of an in-place upgrade (docs.aws.amazon.com).

Frequently Asked Questions (FAQs)

Does GPU Operator 26.7 require Kubernetes 1.37?

No. NVIDIA's Ubuntu 24.04 matrix spans Kubernetes **1.33 through 1.37**, while the managed DRA workflow requires Kubernetes 1.34.2 or later. Kubernetes 1.37 is relevant because it graduates DRA-backed extended resources and device-status capabilities, not because it is the only supported minor.

Can a cluster run the NVIDIA Device Plugin and DRA during migration?

Upstream Kubernetes permits one extended-resource name to use the Device Plugin on some nodes and DRA on others. GPU Operator's managed models are more restrictive: `ClusterPolicy` and `GPUCluster` are mutually exclusive. The supportable default is a separate DRA canary cluster, or a vendor-documented node-pool architecture that proves backend isolation.

Is containerd 2.0 enough?

It is the corrected NVIDIA minimum, but it is not the strongest 1.37 target. NVIDIA validates 2.0 through 2.3, while containerd recommends 2.3.0 or later or 2.4.0 or later for Kubernetes 1.37. Containerd 2.3.x is the clear overlap in the cited matrices.

Can existing `nvidia.com/gpu` manifests move to DRA unchanged?

Kubernetes 1.37's stable DRA-backed extended-resource support is designed for that bridge. For the managed `GPUCluster` path, retaining the legacy `nvidia.com/gpu` name requires a separate `DeviceClass` that sets `spec.extendedResourceName` to that value; create and validate that object in the canary and rollback prerequisites (docs.nvidia.com). The node's resource name must come from only one backend, and the scheduler, driver, runtime, and cleanup behavior still need acceptance testing.

What blocks production DRA enablement?

Any unsupported version combination, ambiguous backend ownership, stale allocation, incorrect NUMA placement, unplanned eviction, failed reschedule, or missed rollback objective should block promotion. A missing universal p95 threshold is not a blocker if the organization defines one from its service-level objective and baseline.

What is the rollback unit?

It is larger than the Helm release. The rollback unit includes chart revision, values, CRDs, operator custom resources, driver and runtime configuration, CDI specifications, node images, feature gates, control-plane state, and the prior workload control suite. Helm rolls back to the prior revision when revision zero is selected (helm.sh), but CRDs and external node state remain separate responsibilities.

Conclusion

GPU Operator 26.7 is a defensible upgrade target for Kubernetes 1.37, provided the cluster fits the documented operating-system, driver, runtime, and component matrices. DRA enablement is a second decision. The stable extended-resource bridge and device status reduce migration friction, while beta and alpha capabilities should remain explicitly gated.

For a cluster on 26.3 or earlier, the lowest-risk path is to inventory and snapshot, follow supported version hops, upgrade 26.7 with the Device Plugin behavior unchanged, validate existing workloads, and then create a separate DRA canary. Promotion requires complete claim lifecycles, topology correctness, deterministic taint behavior, successful restarts and drains, and a timed rollback.

Compatibility tables authorize testing. Only reproducible canary evidence authorizes production.

Source links

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

1. https://github.com/NVIDIA/gpu-operator/releases/tag/v26.7.0
2. https://docs.nvidia.com/datacenter/cloud-native/gpu-operator/26.7/dra-intro-install.html
3. https://docs.nvidia.com/datacenter/cloud-native/gpu-operator/26.7/release-notes.html
4. https://docs.nvidia.com/datacenter/cloud-native/gpu-operator/26.7/platform-support.html
5. https://kubernetes.io/blog/2026/09/03/kubernetes-v1-37-dra-updates/
6. https://kubernetes.io/docs/tasks/configure-pod-container/extended-resource/
7. https://gpusmith.com/articles/en/gpu-cluster-acceptance-testing-runbook
8. https://gpusmith.com/
9. https://docs.cloud.google.com/kubernetes-engine/docs/release-notes
10. https://learn.microsoft.com/en-us/azure/aks/supported-kubernetes-versions
11. https://docs.redhat.com/en/documentation/openshift_container_platform/4.22/html/release_notes/ocp-4-22-release-notes
12. https://docs.digitalocean.com/products/kubernetes/details/supported-releases/
13. https://catalog.ngc.nvidia.com/orgs/nvidia/-/helm-charts/gpu-operator/v26.7.0/version-history
14. https://github.com/kubernetes-sigs/dra-driver-nvidia-gpu/releases/tag/v0.5.0
15. https://containerd.io/releases/
16. https://github.com/containerd/containerd/blob/main/RELEASES.md
17. https://github.com/cncf-tags/container-device-interface
18. https://docs.nvidia.com/datacenter/cloud-native/gpu-operator/26.7/upgrade.html
19. https://kubernetes.io/docs/concepts/resource-management/dynamic-resource-allocation/dra-api/
20. https://kubernetes.io/docs/concepts/resource-management/dynamic-resource-allocation/device-taints/
21. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-scheduler/
22. https://docs.aws.amazon.com/eks/latest/userguide/device-management.html
23. https://docs.digitalocean.com/products/kubernetes/details/supported-gpus/

24. https://etcd.io/docs/v3.7/op-guide/recovery/
25. https://helm.sh/docs/helm/helm_history/
26. https://helm.sh/docs/topics/charts/
27. https://kubernetes.io/releases/version-skew-policy/
28. https://kubernetes.io/docs/reference/instrumentation/metrics/
29. https://prometheus.io/docs/prometheus/latest/querying/functions/
30. https://kubernetes.io/docs/concepts/resource-management/dynamic-resource-allocation/dra-observability/
31. https://kubernetes.io/docs/concepts/cluster-administration/dra/
32. https://docs.oracle.com/en-us/iaas/Content/ContEng/Tasks/contengmanagingspecializedhardwaredevices.htm
33. https://v3.helm.sh/docs/v3/helm/helm_upgrade/
34. https://docs.helm.sh/docs/v3/helm/helm_rollback/
35. https://docs.aws.amazon.com/eks/latest/userguide/kubernetes-versions.html
36. https://helm.sh/docs/helm/helm_rollback/

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