GPU SMITH
COMPUTE ENGINEERING

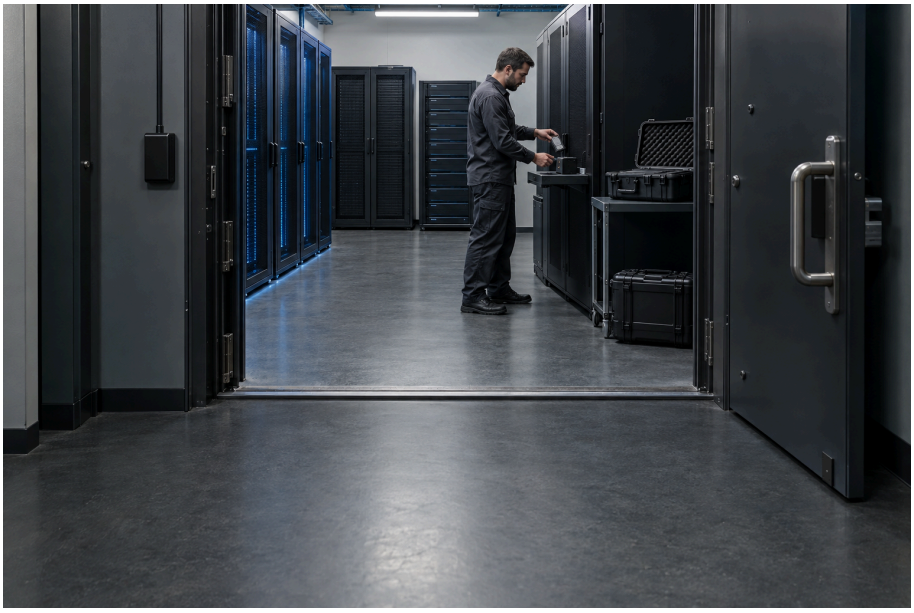
GPU SMITH / RESEARCH & PRACTICAL GUIDANCE

Air-Gapped NVIDIA NIM Upgrade Runbook for 2026

> 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 · air-gapped nvidia nim upgrade runbook · offline nvidia nim upgrade · nvidia nim · air-gapped deployment · kubernetes · gpu operator · model cache · private registry · artifact integrity

Original article: https://gpusmith.com/articles/en/air-gapped-nim-upgrade-runbook



In this article

1. Executive Summary
2. Introduction and Background
3. Key Changes
4. Release-Specific Bill of Materials
5. Connected Staging and Controlled Transfer
6. Implementation Considerations and Process Changes
7. Data Analysis and Evidence
8. Offline Proof, Upgrade, and Rollback
9. Implications and Future Directions
10. Frequently Asked Questions (FAQs)
11. Conclusion

Executive Summary

An **air-gapped NVIDIA NIM upgrade** is not just a container pull performed somewhere else. It is a controlled release transaction whose inputs include the NIM image, model assets, selected profile, Helm chart, rendered Kubernetes objects, every auxiliary image, platform dependencies, configuration, certificates, and a retained rollback set. NVIDIA describes the underlying deployment as a **two-phase workflow**, connected preparation followed by disconnected operation (docs.nvidia.com). The practical upgrade unit is therefore a versioned bill of materials, not a tag and not an informal list of files.

The most important branching decision is the model delivery mode. A model-specific NIM has a bundled manifest but not bundled weights (docs.nvidia.com). A model-free release can instead use a mounted local path, while a remote model URI depends on a runtime manifest generated during connected preparation and retained in the same persistent cache. Losing that cache can force regeneration, which fails in a strict air gap (docs.nvidia.com).

Capacity approval should use measurements from the candidate release. Keep separate values for compressed transfer bytes, expanded staging bytes, private-registry consumption, model or PersistentVolumeClaim storage, node image cache, import scratch, and the retained current release. OCI descriptors expose raw content size in bytes (github.com), but shared layers mean that adding image virtual sizes can overstate actual disk use (docs.docker.com). No universal bundle size or compression ratio should be approved in advance.

Go or no-go evidence should prove completeness, integrity, compatibility, and reversibility. Render the exact chart and values, inventory normal containers, init containers, sidecars, hooks, tests, and GPU Operator operands, then mirror every required artifact. Inside the boundary, apply the approved egress-denial policy, restart from imported assets, verify readiness, list models, run a deterministic smoke request, and retain logs. NVIDIA recommends default-deny egress for stronger validation (docs.nvidia.com), but Kubernetes notes that default-deny egress also blocks Domain Name System traffic unless explicitly allowed (kubernetes.io).

Readiness is necessary, not performance qualification. Rollback must restore both workload objects and the matching model/cache state because a Kubernetes Deployment rollback covers only the Pod template (kubernetes.io).

Introduction and Background

An **air-gapped NVIDIA NIM upgrade runbook** answers a narrower and more operational question than a first-time offline installation guide: can a particular release cross the boundary completely, fit in every storage tier, start without external services or credentials, produce repeatable evidence, and return safely to the prior state? The audience is the joint change authority made up of platform engineering, security, model operations, and service ownership. Each group signs a different part of the same release record.

The upgrade boundary is broader than **NVIDIA NIM** itself. The rendered workload can refer to images from the primary container, init containers, extra init containers, sidecars, chart tests, and hooks. NVIDIA expressly calls out those optional image sources (docs.nvidia.com). Kubernetes independently confirms that init containers can have images distinct from the application containers (kubernetes.io). The platform layer can add GPU drivers, toolkit components, device plugins, validators, discovery agents, and Dynamic Resource Allocation components.

This report treats the **release bill of materials**, or BOM, as the control plane for the whole change. The BOM connects upstream identity to transfer evidence, destination identity, local capacity, test results, and rollback retention. That framing is consistent with the adjacent engineering perspective of GPU Smith, which describes disconnected systems in terms of a local model registry and offline update paths (gpusmith.com) and describes chain of custody through serialized inventory and acceptance records (gpusmith.com). Those are useful engineering principles, not substitutes for the operator's own security policy.

Key Changes

Freeze a deployment shape, not merely a version

The change record should identify **one exact deployment shape** before download begins. Record the NIM image type and version, model source, model and served names, profile identifier, tensor-parallel requirements, chart version, values file hash, target namespace, registry mapping, Kubernetes version, GPU Operator version, driver branch, worker operating system and kernel, storage classes, certificate chain, and current rollback release.

The model mode changes the artifact graph:

- **Model-specific NIM:** preserve the image, its built-in model manifest, the chosen profile, and either its populated cache or the output of `create-model-store`. NVIDIA describes the latter as extracting files from a cached profile (docs.nvidia.com).
- **Model-free with local path:** preserve the exact model directory and mount mapping. The container has no fixed model manifest (docs.nvidia.com).

- **Model-free with remote URI prepared while connected:** preserve the persistent cache containing the generated runtime manifest. Subsequent starts reuse the cached manifest from that volume (docs.nvidia.com).
- **Profile selection:** pin the profile identifier rather than allowing hardware-dependent auto-selection during the change window. Hardware requirements are model-specific. For example, NVIDIA lists **24 GB** as the minimum GPU memory for one Llama 3.1 8B Instruct configuration (docs.nvidia.com). That is an example, not a default for other models or profiles.

The freeze should also state what does not change. A NIM-only release should not silently become a driver, kernel, storage, certificate, or GPU Operator upgrade. If one of those dependencies must change, include it explicitly, test the combined sequence, and define an intermediate recovery point.

Define the evidence model before download

A recurring release needs evidence that remains meaningful after names and locations change. The following crosswalk makes the distinction between a source label, a content identity, an approved integrity method and an operational test explicit.

- **Container identity:** keep the source repository and tag for traceability, but make the manifest digest the content key. Kubernetes describes a digest as the identifier for a specific image version (kubernetes.io); OCI makes that digest required in a descriptor (github.com); the Distribution API defines it as a collision-resistant hash of bytes (distribution.github.io); Docker supports selecting the exact version by digest (docs.docker.com).
- **Transport integrity:** verify the received object, not merely the filename. OCI recommends checking content against the descriptor digest (github.com); a registry verifies uploaded bytes against the supplied digest (distribution.github.io); HTTP Content-Digest is defined for message-content integrity (www.rfc-editor.org); Cosign verifies the signed digest against the container (docs.sigstore.dev).
- **Chart evidence:** retain the original chart, provenance material when used, approved values and rendered manifests. Helm supports fetching charts for inspection or repackaging (helm.sh); its provenance tools address package integrity and origin (docs.helm.sh); local rendering fakes in-cluster lookups (docs.helm.sh); a release manifest includes resources from dependent charts (helm.sh).
- **Workload coverage:** inspect ordinary containers, init containers, tests, probes and storage as separate dependency surfaces. Init containers can use distinct images (kubernetes.io); chart tests are defined by the installed chart (docs.helm.sh); readiness controls Service traffic (kubernetes.io); claim-to-volume binding is one-to-one (kubernetes.io).
- **Provenance policy:** record the mechanism and the policy decision separately. Sigstore can package signed material for portable use (docs.sigstore.dev); SLSA verification covers the provenance-envelope signature (slsa.dev) and comparison of subject digest to artifact (slsa.dev); consumers still form their own expectations (slsa.dev).

- **SBOM policy:** require the format, fields, issuer, delivery point and acceptance result that organizational policy names. SPDX associates hashes with integrity checking (spdx.github.io) and provides a property for recording the integrity method (spdx.github.io); CycloneDX models component inventory (cyclonedx.org) and direct or transitive dependency relationships (cyclonedx.org).
- **Archive accounting:** distinguish an archive from expanded and imported storage. Docker save emits a tarred repository stream (docs.docker.com); the archive includes parent layers, tags and versions (docs.docker.com); load restores both images and tags (docs.docker.com); OCI records raw content size in bytes (github.com).
- **Registry migration:** prove that copied content and accepted attachments arrive together. ORAS can restore into an internal registry without internet access (oras.land), preserve tags, manifests, layers and referrers (oras.land), and recommends including referrers for supply-chain artifacts (oras.land); Harbor can replicate matching artifacts with associated Cosign signatures (goharbor.io).
- **Custody evidence:** keep the approved protection method, media owner and handoff record. NIST identifies cryptography or locked containers as transport protections (nvlpubs.nist.gov), connects transport with authorized personnel and activity records (nvlpubs.nist.gov), and calls for an identifiable owner of removable media (nvlpubs.nist.gov); CISA describes SBOM minimum elements as a baseline (www.cisa.gov).
- **Offline proof:** record both denied and permitted paths. Kubernetes permits traffic when no policy selects a Pod (kubernetes.io); default-deny egress also blocks DNS (kubernetes.io); readiness gates Service traffic (kubernetes.io); image pull secrets are namespace-scoped (kubernetes.io).
- **Recovery scope:** link workload state to retained data and artifacts. Kubernetes rollback covers only the Deployment Pod template (kubernetes.io); persistent claim binding remains one-to-one (kubernetes.io); ORAS can preserve registry referrers (oras.land); Docker load restores image tags but not the cluster data state (docs.docker.com).

Together, those records create four independently checkable layers: content identity and transport integrity (distribution.github.io) (www.rfc-editor.org) (github.com); optional signature and provenance evidence selected by policy (docs.sigstore.dev) (docs.sigstore.dev) (slsa.dev) (spdx.github.io); component inventory and registry-transfer completeness (cyclonedx.org) (oras.land) (goharbor.io); and accountable handling governed by the organization (nvlpubs.nist.gov) (www.cisa.gov) (helm.sh). The crosswalk states what each record proves and prevents a digest, signature, SBOM or successful health check from being treated as proof of everything else.

Replace example inventories with rendered evidence

An **NIM container bundle inventory** must be derived from the actual release. Pull the pinned chart for inspection, archive the original package, render it with the approved values, and preserve the rendered output. Helm documents that `helm pull` is designed for inspection, modification, or repackaging (helm.sh). Local rendering is valuable but not identical to an API-server decision because in-cluster lookups are faked locally (docs.helm.sh).

Inventory both the rendered candidate and the currently installed release. Helm's release manifest includes resources generated by dependent charts (helm.sh), while chart tests are defined by the installed chart and may reference additional images (docs.helm.sh). Compare image references, service accounts, secrets, ConfigMaps, persistent volumes, jobs, hooks, Custom Resource Definitions, and network dependencies.

Promote immutable identity and evidence

Mutable tags remain useful labels, but the approval record should add **content digests** wherever the tooling supports them. Kubernetes states that an image digest uniquely identifies a particular image version (kubernetes.io). The Open Container Initiative, or OCI, descriptor makes the digest a required identifier (github.com) and recommends verifying content obtained through untrusted sources against that digest (github.com).

Digest recording is directly useful for reproducibility. Signature verification, provenance policy, Software Bill of Materials, or SBOM, acceptance, and removable-media controls are **organization-specific controls** unless the selected NVIDIA workflow says otherwise. Cosign can verify that the signed digest matches the container (docs.sigstore.dev), and SLSA verification compares the artifact digest with the provenance subject (slsa.dev). Neither fact means NVIDIA mandates those controls for every NIM transfer.

Release-Specific Bill of Materials

The release BOM should be machine-readable and reviewable as a table. Every row needs a stable source locator, candidate version or tag, immutable digest when available, measured sizes, destination, verification method and result, and retention decision. _`Table 1`_ shows the minimum schema and the discovery source for each class.

Artifact role	How to discover it	Required release record	Offline destination and proof
NIM runtime image	Resolve the approved image from pinned chart values and deployment shape.	Source name, tag, manifest digest, platforms, license reference, SBOM or provenance reference, archive bytes and imported bytes.	Private repository, rewritten reference, post-import digest, pull result.
Model profile, weights, tokenizer and manifest	Derive from NIM mode, model source and pinned profile.	Model identifier and revision, profile hash, cache or model-store path, runtime manifest hash, file count and measured bytes.	PVC or model path, ownership, mount mode, manifest presence, offline start result.
Helm chart and values	Save the original package, chart metadata, approved values, rendered manifests and diffs. Helm provenance can verify package integrity and origin (docs.helm.sh).	Chart version, package digest, provenance result if policy requires it, values hash, rendered manifest hash.	Local chart store or OCI registry, installed release record, render comparison.
Auxiliary workload images	Parse all container and init-container arrays, chart hooks, tests, enabled sidecars and extra-init settings.	Source and destination digest per image, enabling value, workload object and purpose.	Private registry presence and a pull test from a target node.

Artifact role	How to discover it	Required release record	Offline destination and proof
GPU platform images	Read the chosen GPU Operator component matrix. NVIDIA calls it the list of operands and default operand versions (docs.nvidia.com).	Operator, driver, toolkit, validator, discovery, plugin and optional DRA image identities, plus OS/kernel compatibility.	Mirrored paths, node pull proof, Operator pod state, scheduler resource proof.
Configuration, trust and licenses	Enumerate ConfigMaps, Secrets references, service accounts, image pull secrets, certificate bundles, license and policy records.	File hash, owner, scope, secret identifier without secret value, expiry and change ticket.	Approved namespace or secret store, certificate trust test and access-control result.
Rollback set	Capture the current installed manifest, values, image digests, cache/model snapshot and compatible platform state before mutation.	Recovery point, retention expiry, restore owner, trigger and maximum outage from the change plan.	Offline registry and storage presence, restore rehearsal evidence.

The table is deliberately release-specific. The same nominal NIM version can produce a different BOM when values enable tests, sidecars, LoRA adapters, a different model mode, or a different GPU platform. A PVC is also not interchangeable with the workload declaration. Kubernetes defines PersistentVolumeClaim to PersistentVolume binding as one-to-one (kubernetes.io), so the record must identify the actual claim, volume, snapshot, capacity and restore behavior.

_`Table 2`_ is the dependency coverage check. It prevents a visually complete image list from missing short-lived or platform-owned artifacts.

Rendered component	Artifact or dependency to cover	Evidence before transfer	Evidence inside air gap
Deployment, StatefulSet or Pod	Every `containers[].image`, commands, environment references, volumes and probes	Image digest list, rendered object hash, configuration source	Pull success, Ready condition, readiness response
Init and extra-init containers	Every `initContainers[].image`, mounted input and completion condition	Enabled-values mapping and per-image digest	Completed status and retained logs
Sidecars	Image, ports, volumes, certificates and destinations	Network and storage dependency map	Ready status and allowed-destination proof
Helm hooks and tests	Job image, hook annotation, test inputs and cleanup policy	Rendered hook/test objects and image digests	Test execution and logs; the image must exist even if the job is brief
GPU Operator and operands	Operator plus release-specific driver, toolkit, plugin, validator, discovery and optional DRA images	Component matrix, OS/kernel/driver compatibility, mirrored digests	Operator, operand and validator state plus scheduler resource proof
Model storage	Weights, tokenizers, profile data, cache and runtime manifest	Complete file inventory, digest set, file count and measured bytes	Mounted path, profile match, readable files and restart proof

Rendered component	Artifact or dependency to cover	Evidence before transfer	Evidence inside air gap
Registry, DNS and trust	Registry endpoint, certificate chain, namespace pull secret and any approved cluster DNS path	Name resolution plan, certificate bundle and secret reference	Node and Pod pull test, TLS validation, DNS behavior under policy

The coverage table should be generated from evidence, then reviewed by humans. Registry pull credentials and model-access credentials belong in separate records. Kubernetes requires image pull secrets to exist in the Pod's namespace (kubernetes.io). The disconnected release should retain only credentials required for internal services.

Connected Staging and Controlled Transfer

Build and verify on the connected side

The connected staging host should be treated as a reproducible build environment. Its job is to fetch the approved inputs, turn model inputs into the selected offline representation, record immutable identities, and create a sealed transfer set.

- **Validate the release worksheet:** reject unresolved tags, unapproved sources, absent licenses, missing destination mappings, or an undefined rollback release.
- **Pull and record artifacts:** fetch exact charts, images, model revisions, certificates and platform components. Record source locators before rewriting names.
- **Prepare model state:** run the release-matched NIM image to populate the selected profile's cache or create the model store. Preserve the generated runtime manifest when the workflow depends on it.
- **Render candidate resources:** render with the exact values intended for the disconnected cluster. Extract every image and external hostname, then reconcile both lists to approved destinations.
- **Collect integrity evidence:** compute organization-approved hashes, record OCI digests, retrieve accepted signatures, provenance and SBOMs where policy requires them. SPDX describes a hash as commonly used for integrity checking (spdx.github.io).
- **Archive without hiding structure:** preserve the BOM, chart, values, rendered manifests, model/cache tree, image archives, verification material and a top-level transfer manifest as separate named components.
- **Measure after construction:** record archive bytes, expanded bytes, registry import results and file counts. Do not infer actual use from nominal image sizes.

NVIDIA Mission Control offers a useful parallel pattern: its air-gap CLI reads a manifest and downloads charts, images and files into a portable bundle (docs.nvidia.com). This does not make that tool the NIM upgrade mechanism, but it illustrates the right release-manifest discipline.

Apply the organization's transfer controls

NVIDIA permits transfer through an allowed channel. The organization decides what **allowed** means. Keep policy-specific approval, media identity, custodian, handoff time, seal or case identifier, source hash, destination hash, sanitization state and exception record in the custody log. NIST notes that transport protection can use cryptography, locked containers, or both (nvlpubs.nist.gov) and associates accountable transport with authorized personnel and retained activity records (nvlpubs.nist.gov). These are external control references, not claims about NVIDIA requirements.

If supply-chain artifacts are part of the approved evidence set, keep them with the objects they describe. CycloneDX can represent first-party and third-party component inventory (cyclonedx.org); Cosign can save signed material as a portable Sigstore bundle instead of uploading it (docs.sigstore.dev). CISA describes its SBOM minimum elements as a baseline rather than a new federal requirement (www.cisa.gov).

Import and rewrite deterministically

On the disconnected side, verify the outer bundle before import, then each inner artifact as it reaches its final store. The registry specification says a registry verifies uploaded content against the supplied digest (distribution.github.io). Re-query the destination manifest digest rather than assuming the push retained the expected identity.

- **Trust first:** install the approved private-registry certificate chain on nodes and clients before testing imports.
- **Import by immutable inventory:** load archives or copy OCI content, preserving platform manifests, tags used by the chart and accepted supply-chain attachments.
- **Rewrite once:** maintain a source-to-destination map and generate chart values from it. Retain both names and the verified digest in the release record.
- **Keep secrets separate:** the bundle may contain secret references, but it should not carry connected-side NGC or Hugging Face credentials into runtime. NVIDIA explicitly says not to set those credentials in the air-gapped phase (docs.nvidia.com).
- **Preserve referrers when required:** ORAS export and restore can preserve tags, manifests, layers and referrers (oras.land); omitting referrers can strand signatures or attestations.
- **Reconcile the result:** every BOM row ends with a destination, post-import identifier, measured size and verification status. Missing or changed rows are a stop condition.

Implementation Considerations and Process Changes

Preflight inside the boundary

Preflight should run before the maintenance window while the current release is still healthy. It verifies that the candidate can start using only internal resources.

- **Registry reachability:** pull every candidate digest from representative worker nodes, not only from an administrator workstation.
- **Certificate trust:** validate the registry and internal service chains from both node runtime and Pod context.
- **Image completeness:** query every main, init, sidecar, hook, test and GPU platform image listed in the BOM.
- **Model path and PVC:** verify claim binding, mount access, required files, profile identifier, ownership and free space.
- **GPU compatibility:** match Operator, operand, driver, operating system, kernel, CUDA-facing runtime and selected NIM profile. Precompiled driver support is limited to NVIDIA's listed OS, kernel and driver combinations (docs.nvidia.com).
- **Scheduler visibility:** confirm allocatable GPU resources through the Kubernetes API and a release-matched validation workload.
- **Resources and quotas:** compare requested GPUs, CPU, memory, shared memory, ephemeral storage and PVC capacity with node availability and namespace quotas.
- **Network dependencies:** identify permitted private registry, cluster DNS, monitoring, logging and identity destinations. Record what the proof policy will deny and allow.
- **Credential absence:** inspect the rendered candidate and runtime secret references for connected-side NGC or Hugging Face credentials.
- **Rollback availability:** perform the same reachability checks for the retained current images and restore point.

GPU Operator changes deserve their own gate. NVIDIA supports Operator upgrades within a major release or to the next major release (docs.nvidia.com). Helm does not automatically upgrade existing Custom Resource Definitions, so the selected Operator path must account for CRDs (docs.nvidia.com). An Operator hook can also introduce an image dependency because NVIDIA's hooks use the Operator image itself (docs.nvidia.com).

Execute, observe and decide

Choose a canary or maintenance-window strategy based on the service's available capacity and state semantics. A canary is useful only if the cluster can host old and new workloads without oversubscribing GPUs or storage and if traffic can be separated. Otherwise, use an explicit stop, preserve, upgrade, validate, release sequence.

The go or no-go worksheet should include:

- **Decision:** proceed, hold, or roll back.
- **Trigger:** a measurable condition such as image pull failure, profile mismatch, readiness timeout, deterministic smoke mismatch, error-rate threshold, or cache incompatibility.

- **Owner:** the named role authorized to declare the condition.
- **Evidence:** command output, Kubernetes object snapshot, request record, digest comparison or storage measurement.
- **Time bound:** the maximum outage and the latest instant at which rollback must begin, supplied by the approved change plan.
- **Recovery action:** chart restore, workload rollback, cache or model snapshot restoration, registry reference reversion, or platform-state restoration.

Readiness should be interpreted narrowly. NVIDIA's NIM Kubernetes example expects **HTTP 200** from the readiness endpoint (docs.nvidia.com), and Kubernetes removes non-ready Pods from Service traffic (kubernetes.io). Neither proves latency, throughput, concurrency, output quality or sustained thermal behavior.

Data Analysis and Evidence

Capacity planning should expose assumptions rather than compress them into one storage number. _`Table 3`_ is the quantitative worksheet for **GPU capacity planning for NIM** and bundle logistics. Every input is measured for the frozen release; the reserve and retention factors are explicit policy inputs.

Capacity pool or calculation	Measured inputs	Calculation and acceptance evidence
Transfer archive	Final archive bytes plus detached verification material	Sum actual file lengths after bundle construction; record filesystem command output and bundle hash.
Expanded staging	Extracted chart, model/cache, image archives, SBOMs, manifests and logs	Measure the fully expanded staging tree. Do not substitute compressed bytes.
Import scratch	Peak temporary files used by the chosen registry and import tools	Measure peak free-space change during a representative import; add an approved reserve.
Private registry	Candidate manifests and layers, retained current layers, attachments and registry metadata	Measure repository consumption after import. Shared layers can reduce network and storage use (docs.docker.com).
Model/PVC	Candidate model store or cache, runtime manifest, current rollback set, adapters and filesystem overhead	Candidate measured bytes + retained current bytes + policy reserve. Record snapshot and restore space separately.
Node image cache	Per-node candidate layers, currently pinned layers and eviction headroom	Measure on each targeted node class after a representative pull. Do not multiply virtual image size blindly.
GPU allocation	Profile GPU count, GPU memory requirement, replicas, canary overlap and platform reservation	Confirm requested custom resources; Kubernetes exposes GPUs through a schedulable resource such as `nvidia.com/gpu` (kubernetes.io).
Transfer duration	Measured bundle bytes, sustained approved-channel bits per second and operational overhead factor	Base seconds = bytes × 8 / sustained bits per second. Multiply by the locally observed overhead factor, then add custody and verification time.

Capacity pool or calculation	Measured inputs	Calculation and acceptance evidence
Retained capacity	Current release, candidate release, required older recovery points and evidence retention	Required retained capacity = sum of measured retained sets + explicit safety reserve.

The table separates values that often get conflated. A Docker save operation creates a tarred repository stream (docs.docker.com), and load restores images and tags (docs.docker.com). Those archive bytes are not automatically equal to destination registry bytes or node cache bytes. OCI content size is an exact property of a descriptor, but registries and filesystems add metadata and can share blobs.

For a worked method, suppose the worksheet contains symbols rather than published universal values: **B** is the final bundle bytes, **R** is the measured sustained channel rate in bits per second, and **O** is the observed operational multiplier. The transport estimate is `(B × 8 / R) × O`. The team should derive **R** from a representative transfer on the approved path and **O** from handling, scan, verification, import and custody timings. A marketing link rate is not a measured sustained rate.

Quantitative evidence should also cover change behavior. NVIDIA documents a default GPU driver upgrade policy of one parallel node and at most **25%** unavailable when the custom resource omits a policy (docs.nvidia.com). That is a product default, not an outage budget. The change board should replace defaults with an explicit setting when the service's resilience model demands it.

Finally, preserve enough numbers to reproduce the decision: counts of BOM rows and unresolved rows, bytes per storage pool, image and model digest counts, GPU and replica requirements, preflight duration, startup time, readiness time, smoke-test response hash, rollback duration and retained evidence size. These are release measurements, not vendor benchmarks.

Offline Proof, Upgrade, and Rollback

Prove the disconnected path

The offline test should be conducted with the same policy and destinations intended for production, not with temporary broad access. Kubernetes allows all Pod ingress and egress when no NetworkPolicy selects them (kubernetes.io), so the absence of observed calls is not proof of enforced isolation.

Use this **verify NIM deployment offline** sequence:

- **Apply the policy:** record the egress-denial object, allowed internal destinations, selected Pods, cluster network plugin and enforcement time.
- **Prove necessary internal access:** confirm DNS if allowed, private registry, storage, logging and monitoring paths. A blanket deny that blocks DNS can create a misleading application failure.
- **Restart from imported assets:** delete or restart the candidate Pod after policy enforcement so image, manifest and model resolution are exercised under the intended conditions.
- **Check object state:** capture events, init completion, container state, volume mounts, GPU allocation and readiness.

- **Check the service:** query health, list models and issue the fixed smoke request. NVIDIA's validation sequence covers health, model listing and inference (docs.nvidia.com).
- **Record determinism boundaries:** save request body hash, parameters, response hash or normalized output, model name, profile and timestamps. NVIDIA's documented deterministic mode is profile-dependent, so release support must be rechecked (docs.nvidia.com).
- **Inspect credentials:** show that external registry and model tokens are absent from the rendered configuration and runtime environment.
- **Retain failure evidence too:** events and logs from a failed attempt can identify a missing artifact or hidden dependency. Correct the BOM, rebuild the candidate and rerun from a clean state.

The validation record should identify policy applied, allowed destinations, Pod restart identity, readiness result, model-list result, request hash, response record, logs and credential scan result. Treat this as functional acceptance only. Performance qualification needs its own load shape, duration, concurrency, latency percentiles, throughput measure and thermal or power criteria.

Roll back the complete state

A safe rollback reverses the release transaction, not just the Deployment. Restore the prior chart and values, container references, ConfigMaps, Secrets references, model/cache snapshot, persistent manifest, profile and any coupled platform state. Verify that the rollback artifacts are locally reachable before beginning the candidate upgrade.

- **Trigger promptly:** use the approved readiness deadline, smoke mismatch, repeated restart, GPU incompatibility, storage error or policy breach as an objective trigger.
- **Stop mutation:** preserve candidate logs and object snapshots before they are overwritten, provided this does not delay the recovery objective.
- **Restore workload declaration:** use the retained chart and values or the approved manifest snapshot.
- **Restore data state:** remount or recover the prior model store/cache and runtime manifest. Do not assume the candidate left the cache backward compatible.
- **Restore platform state if coupled:** drivers preinstalled on hosts are outside GPU Operator driver management (docs.nvidia.com), which changes the recovery procedure.
- **Revalidate offline:** repeat readiness, model listing and the prior deterministic smoke record under the same egress policy.
- **Close the record:** capture elapsed rollback time, restored digests, recovered data point, evidence location and remaining cleanup actions.

The recurrent **air-gapped NIM upgrade checklist** should therefore have paired candidate and rollback columns. An artifact present only for the candidate is not enough. A rollback set whose registry manifest, model files or certificate chain has expired is not a recovery plan.

Implications and Future Directions

Air-gapped operations benefit from treating manifests and evidence as products of the release pipeline. NVIDIA's Mission Control workflow verifies transferred bundles by recomputing SHA256 checksums against a bundle manifest (docs.nvidia.com). A NIM program can apply the same pattern with its own tooling: generate the BOM from rendered resources, attach model and platform records, verify after transfer, then bind test evidence back to the same release identifier.

OCI-native movement can reduce format translation. ORAS documents restoring a connected-side backup into a registry without internet access (oras.land), and Harbor replication can include associated Cosign signatures (goharbor.io). Operators should validate these capabilities against their chosen versions, because artifact referrer behavior and registry support evolve.

The durable improvement is organizational: make release evidence reusable. The security team defines accepted media, digest, signature, provenance and SBOM policies. Platform engineering produces rendered inventories and compatibility evidence. Model operations owns model identity, profile and smoke semantics. Service ownership defines outage and rollback triggers. The change board evaluates one reconciled record instead of four partial checklists.

Automation should remain inspectable. A future pipeline can fail on unresolved image tags, an unknown external hostname, a missing destination digest, an unapproved credential variable, insufficient measured capacity, or absent rollback artifacts. It should still emit the underlying chart, values, manifests, source-to-destination mapping and measurements so an approver can reproduce the decision.

Frequently Asked Questions (FAQs)

What is the difference between NVIDIA NIM air-gapped deployment and an offline upgrade?

An **NVIDIA NIM air-gapped deployment** explains how to prepare assets while connected and run without internet access. An **offline NVIDIA NIM upgrade** repeats that workflow for a specific new release while preserving the running release, proving compatibility, measuring capacity, recording custody and retaining a tested rollback set.

What must be included in a NVIDIA NIM offline installation bundle?

A **NVIDIA NIM offline installation** bundle should include the pinned NIM image, the chosen model store or cache and manifest, chart package, approved values, rendered objects, every auxiliary image, required GPU platform images, configuration and certificate material, licenses and accepted supply-chain evidence, plus the BOM and verification manifest. NVIDIA's core rule is that every image referenced by the downloaded chart must be available to the disconnected cluster (docs.nvidia.com).

What are the NVIDIA NIM model cache requirements?

The **NVIDIA NIM model cache requirements** depend on deployment mode. A model-specific NIM needs a populated profile cache or model store. A model-free NIM using a local path serves that mounted directory. A model-free NIM prepared from a remote URI needs the persistent cache that contains its generated runtime manifest. In all cases, measure the actual candidate, keep the mount path stable and test a restart under egress denial.

How should artifact integrity be verified?

Record source and destination digests and use the verification mechanisms approved by the organization. OCI digests identify content, registry uploads can be checked against supplied digests, and optional signatures, provenance and SBOMs can be retained according to policy. SLSA explicitly allows consumers to form their own artifact expectations (slsa.dev). Do not describe an internal signature or removable-media rule as an NVIDIA mandate unless the selected NVIDIA release documentation says so.

When is the upgrade ready for production?

It is ready when the release BOM reconciles without missing rows, preflight passes, measured capacity includes candidate and rollback sets, the imported digests match, the NIM Pod restarts under the approved egress policy, readiness and model listing succeed, the deterministic smoke record meets its acceptance rule, external credentials are absent, and rollback remains executable within the approved outage window.

Conclusion

A reliable air-gapped NIM upgrade is a chain of evidence. Freeze the complete deployment shape, derive the dependency inventory from the exact chart and values, prepare the correct model representation, measure each storage pool, transfer through the approved channel, verify at the destination and prove runtime behavior under enforced egress denial.

The recurring unit of work is the release BOM. It connects the NIM image and model profile to charts, auxiliary images, GPU platform components, configuration, trust, capacity, custody, smoke results and rollback retention. That connection prevents the most common operational omissions: an unseen init image, a lost runtime manifest, an incompatible driver path, an underestimated import peak, a connected-side credential left in values, or a rollback that restores only a Pod template.

The change board should approve measured facts, not generic bundle estimates. Functional readiness, offline proof and performance qualification are separate decisions. With paired candidate and rollback records, the organization can repeat the process for each NIM release, identify precisely what changed and retain a defensible account of what crossed the boundary, what ran, and how service can be restored.

Source links

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

1. https://docs.nvidia.com/nim/large-language-models/latest/deployment/air-gap-deployment.html
2. https://github.com/opencontainers/image-spec/blob/main/descriptor.md?plain=1

3. https://docs.docker.com/engine/storage/drivers/
4. https://kubernetes.io/docs/concepts/services-networking/network-policies/
5. https://kubernetes.io/docs/concepts/workloads/controllers/deployment/
6. https://kubernetes.io/docs/concepts/workloads/pods/init-containers/
7. https://gpusmith.com/
8. https://gpusmith.com/articles/en/llm-inference-hardware-sizing-guide
9. https://docs.nvidia.com/nim/large-language-models/latest/get-started/prerequisites.html
10. https://kubernetes.io/docs/concepts/containers/images/
11. https://distribution.github.io/distribution/spec/api/
12. https://docs.docker.com/reference/cli/docker/image/pull/
13. https://www.rfc-editor.org/info/rfc9530/
14. https://docs.sigstore.dev/cosign/verifying/verify/
15. https://helm.sh/docs/v3/helm/helm_pull/
16. https://docs.helm.sh/docs/v3/topics/provenance/
17. https://docs.helm.sh/docs/v3/helm/helm_template/
18. https://helm.sh/docs/v3/helm/helm_get_manifest/
19. https://docs.helm.sh/docs/v3/helm/helm_test/
20. https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes/
21. https://kubernetes.io/docs/concepts/storage/persistent-volumes/
22. https://docs.sigstore.dev/cosign/signing/signing_with_containers/
23. https://slsa.dev/spec/v1.2/verifying-artifacts
24. https://spdx.github.io/spdx-spec/v3.0.1/model/Core/Classes/Hash/
25. https://spdx.github.io/spdx-spec/v3.0.1/model/Core/Properties/verifiedUsing/
26. https://cyclonedx.org/specification/overview/
27. https://docs.docker.com/reference/cli/docker/image/save/
28. https://docs.docker.com/reference/cli/docker/image/load/
29. https://oras.land/docs/how_to_guides/backup-restore/
30. https://goharbor.io/docs/edge/administration/configuring-replication/
31. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/800-171r3/NIST.SP.800-171r3.html
32. https://www.cisa.gov/sites/default/files/2025-08/2025_CISA_SBOM_Minimum_Elements.pdf
33. https://docs.nvidia.com/datacenter/cloud-native/gpu-operator/latest/platform-support.html
34. https://docs.nvidia.com/mission-control/docs/nmc-software-installation-guide/2.3.1/airgapped-installation/overview.html
35. https://docs.nvidia.com/datacenter/cloud-native/gpu-operator/latest/precompiled-drivers.html
36. https://gpusmith.com/articles/en/gpu-cluster-acceptance-testing-runbook
37. https://docs.nvidia.com/datacenter/cloud-native/gpu-operator/latest/upgrade.html
38. https://docs.nvidia.com/nim/large-language-models/2.0.0/deployment/kubernetes-deployment/helm-k8s.html
39. https://kubernetes.io/docs/tasks/manage-gpus/scheduling-gpus/

40. https://docs.nvidia.com/datacenter/cloud-native/gpu-operator/latest/gpu-driver-upgrades.html
41. https://docs.nvidia.com/nim/large-language-models/1.15.0/deterministic-mode.html
42. https://docs.nvidia.com/mission-control/docs/nmc-software-installation-guide/2.3.1/airgapped-installation/downloading-artifacts.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

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.