MCP SERVER | NO CODE | CLOUD HOSTED

# NVIDIA NIM MCP

## Govern Hardware Limits and ML Metrics

NVIDIA NIM MCP connects your AI agent directly to physical hardware metrics, giving you deep visibility into GPU usage and LLM performance. You can check container health, track memory limits, pull real-time resource statistics via Prometheus endpoints, and manage model scaling—all without logging into a dashboard. It gives the ops engineer total command over their ML infrastructure.

A+ Quality Score 100/100

mlops | gpu-telemetry | container-management | hardware-profiling | resource-monitoring | infrastructure-limits

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/nvidia-nim

SCAN TO VIEW

# The connectivity layer between AI and the world's software.

Vinkius sits between AI and every application. All communication passes through Vinkius Cloud via the Model Context Protocol (MCP) — with governance, observability, and security at every layer.

# Your AI Connections Run Through Vinkius Cloud

The world's largest managed MCP catalog

Vinkius is the connectivity layer where AI connects to the software your business already runs. We handle the hosting, the security, the credentials, the uptime — you get agents that actually do things.

We operate the world's largest managed MCP catalog. Major SaaS platforms, CRMs, databases, and cloud providers — running, monitored, production-ready. This MCP server is hosted and maintained by the Vinkius Cloud for AI Agents.

> *The agent doesn't manage credentials, doesn't manage uptime, doesn't manage security. Vinkius does.*
>
> — Architecture principle

## Four Pillars of the Vinkius Runtime

**01 — Security by design**

Credentials stay encrypted at rest via AES-256. The AI agent never touches raw keys — they're injected into a sandboxed V8 isolate at runtime. Actions are logged, and connections have an emergency kill switch.

**02 — Built on MCP Fusion**

This MCP server was built with **MCP Fusion**, the open-source framework (Apache 2.0) that powers the entire Vinkius catalog. Schema-as-firewall strips undeclared fields, compiled PII redaction runs at zero overhead, and cryptographic lockfiles produce git-diffable audit trails.

**03 — Deterministic observability**

Eight immutable metrics per endpoint: request volume, p95 latency, error rate, active connections, cost attribution. A live payload feed logs every tool call with mutation detection.

**04 — Autonomous operations**

Servers are deployed, monitored, and patched autonomously. New capabilities and security patches ship weekly. Zero-downtime deployments ensure continuous availability across all managed MCP servers.

| AES-256 | Ed25519 | 24h TTL | V8 Isolate |
|---|---|---|---|
| Encryption at rest | PKI vault signatures | Ephemeral session keys | Sandboxed execution |

## One Token. Instant Access.

Every MCP server on Vinkius is accessed through a **Connection Token**. Tokens are generated in the cloud dashboard and produce a unique MCP endpoint URL. Paste this URL into any MCP-compatible client — no SDK required.

A single token can serve **multiple AI clients simultaneously**, or you can issue separate tokens per client for granular access control. Each token tracks its own request count, last activity timestamp, and can be individually enabled or revoked.

MCP ENDPOINT

```
https://edge.vinkius.com/{token}/mcp
```

Claude · Cursor · VS Code · Windsurf · Grok · Gemini

## Security Is the Architecture

Security in Vinkius is not a feature — it's the foundation of the runtime. The gateway enforces multiple independent protection layers between AI agents and third-party APIs.

**01 — Ed25519 PKI Vault**

Every workspace has an Ed25519 Master Key. Session keys are generated ephemerally (24h TTL) and signed by the Master Key. Credentials never leave the vault boundary.

**02 — V8 Isolate Sandboxing**

Tool code runs inside isolated-vm V8 isolates with 64 MB memory caps and per-request timeouts. No filesystem access, no network access except through the SSRF-guarded fetch bridge.

**03 — SSRF Guard**

All outbound HTTP requests are DNS-resolved and validated before execution. Private IP ranges (10.x, 172.16-31.x, 192.168.x, AWS metadata 169.254.x) are blocked at the network layer.

**04 — DLP & PII Redaction**

A ResponseGuard pipeline intercepts every tool response. Configurable redaction patterns strip sensitive fields (emails, SSNs, card numbers) before data reaches the AI agent.

**05 — Cryptographic Audit Trail**

Every request is signed into a SHA-256 hash chain with Ed25519 signatures. Events form a tamper-proof, SIEM-exportable forensic record.

**06 — Honeytoken Trap System**

Phantom credentials are injected into isolated environments. If a honeytoken is used outside Vinkius infrastructure, the server is quarantined instantly.

# Emergency Kill Switch

EU AI Act Art. 14(1) Compliant

The kill switch is an **emergency halt** mechanism — not a simple toggle. When triggered, it executes three actions atomically:

**01 — Server deactivated**

The MCP server is immediately taken offline across the entire cluster.

**02 — All tokens revoked**

Every connection token is invalidated. Total lockout — reconnection blocked until new tokens are issued.

**03 — WebSocket connections killed**

Active connections terminated via Redis pubsub broadcast. Propagates to every runtime node in the cluster.

## Full Visibility. Zero Guesswork.

The Vinkius cloud dashboard includes a full MCP Governance suite — real-time analytics and security controls for production AI operations.

**Control Plane**

KPI dashboard with request volume, latency, success rate, token consumption, and AI-generated operational briefings.

**Agent Activity**

Which AI clients are connecting, how often, and what they're doing — real-time session tracking.

**FinOps**

Cost tracking per tool, payload compression savings, budget optimization signals, and consumption trends.

**Tool Health**

Slowest and most error-prone tools, with actionable root-cause insights and performance baselines.

**Firewall & DLP**

PII redaction activity, sensitive data protection counters, and security event timeline.

**Incident Log**

Error trends, failure rates, status-code breakdowns, and forensic audit trail access.

> Get started at **cloud.vinkius.com** — connect your AI agent in under 60 seconds.

# NVIDIA NIM MCP

8 tools available
Cloud-hosted on Vinkius

This MCP lets your agent talk directly to complex physical hardware running AI workloads. Instead of relying on high-level dashboards that mask the actual bottlenecks, you gain direct control over monitoring and resource management for NVIDIA containers. You can ask your agent to check if a model has finished loading or pull raw performance numbers from Prometheus endpoints. The system allows you to map exactly what's loaded onto the GPU and even scale the entire infrastructure up or down with simple commands. It’s like giving your AI client root access to the machine's core stats. If managing this complexity feels overwhelming, remember that Vinkius hosts this MCP so your agent can connect once and get access to all these critical hardware tools.

## Core Capabilities

**01 — Check container health status**

Determines if the physical host container orchestrator is running and responsive using liveness probes.

**02 — Verify model readiness**

Confirms whether the GPU inference layers have successfully loaded all required model artifacts for use.

**03 — Extract hardware resource usage**

Gathers specific details on allocated memory and topological limits mapped onto the NIM proxy.

**04 — Pull performance metrics data**

Fetches raw, actionable scaling metrics directly from Prometheus endpoints attached to the orchestrator.

**05 — Audit active models deployed**

Lists all currently loaded large language models (LLMs) that are available for inference targets on the backend array.

**06 — Adjust resource scaling**

Changes the number of hardware replicas assigned to the proxy, allowing you to scale execution layers up or down automatically.

## One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/nvidia-nim** — connect your AI agent in three steps.

01 Your agent targets the local instance by specifying the `NVIDIA_NIM_URL` in the prompt.

02 The system passes native proxy queries that explore hardware latencies using specific Prometheus endpoints.

03 The MCP maps and executes the necessary hardware limits, returning diagnostic error codes or status reports.

The bottom line is that your agent gets a direct data stream into the physical performance layer of your AI infrastructure.

## Built For

This MCP is for MLOps Engineers and Infrastructure Admins who are tired of guessing why their LLM inference keeps failing. If you spend too much time clicking through separate monitoring dashboards just to piece together a single picture of GPU usage, this tool is mandatory.

**MLOps Engineer**

Uses the MCP to run continuous checks on container health and pulls raw metrics data when diagnosing latency spikes or scaling issues.

**Hardware Proxy Admin**

Manages model deployment by listing active LLMs and adjusting replication counts (`nim_scale_replicas`) based on traffic load.

**Infrastructure Integrator**

Validates the physical bounds of the entire stack, checking GPU memory variables (`nim_get_gpu_status`) before any new model deployment.

## What Changes When You Connect

01 Instant Model Inventory: Use `nim_list_models` to get an immediate, clean dump of every LLM target running on your system. You don't have to guess what models are active.

02 Deep Health Checks: Quickly verify the entire stack with dedicated calls like `nim_check_health_live` or confirming readiness using `nim_check_health_ready`. This is faster than waiting for a dashboard widget to load.

03 Performance Benchmarking: Access raw, structured data by running `nim_get_metrics`. This lets you pull Prometheus hardware scaling metrics needed for true performance analysis.

04 Resource Visibility: Know exactly what's consuming memory. `nim_get_gpu_status` provides a clear breakdown of GPU topological limits and allocated memory variables.

05 Operational Stability: When traffic spikes, don't panic. Use `nim_scale_replicas` to dynamically adjust resources, ensuring your models stay online without manual intervention.

## Real-World Applications

**Diagnosing a sudden performance drop**

The agent detects high latency and runs `nim_get_metrics`. The output shows that GPU utilization is maxed out, pointing the engineer immediately to insufficient resources. They then use `nim_scale_replicas` to allocate more capacity.

**Validating model deployment**

Before launching a new feature, an admin uses `nim_get_metadata` to verify that the foundational configuration bounds are correctly set. They then run `nim_check_health_ready` to ensure all required artifacts loaded properly.

**Troubleshooting container failures**

The agent fails to connect, so the engineer runs `nim_get_container_logs` and uses `nim_list_models` simultaneously. The logs reveal a permission error, while the model list confirms the correct models were supposed to be running.

# Patterns to Avoid

### Checking system status manually

✗ AVOID

The user opens the terminal and has to run multiple `nvidia-smi` commands, cross-reference them with a separate dashboard, and then try to synthesize a single report on GPU memory.

✓ INSTEAD

Instead, let your agent use `nim_get_gpu_status` for an immediate snapshot of GPU limits, followed by `nim_get_metrics` to pull the full Prometheus dataset. This gives you all the data points in one actionable query.

### Guessing resource needs

✗ AVOID

The team manually guesses that doubling the replicas is enough for a traffic increase, leading to over-provisioning or under-scaling.

✓ INSTEAD

Use `nim_get_metadata` first. This reveals the current foundational bounds and metrics. Then use `nim_scale_replicas` with data-driven logic instead of gut feeling.

## The Right Fit

You must use this MCP if your core problem is determining *why* an AI workload failed or slowed down, and that failure is tied to underlying hardware capacity, container orchestration, or resource allocation. This isn't for general API calls; it's deep system diagnostics. Don't use this if you just need to send a message or read simple application data—you need a messaging or database MCP instead. If you only care about seeing the model names, `nim_list_models` is sufficient, but if you also need to check that the whole machine is actually healthy and ready for work, you must use both `nim_check_health_live` and `nim_get_metrics` together.

# The Pain of Dashboard Overload

Today, figuring out why your LLM inference is slow feels like playing detective with a dozen separate dashboards. You jump between the container logs

With this MCP, you skip all the clicking. You tell your agent what you need—say, 'Show me the current resource limits and how many LLMs are

tab, the Prometheus graph, and the GPU stats panel. You copy-paste numbers from one dashboard into a spreadsheet just to see if the memory usage matches the reported throughput.

loaded'—and it calls `nim_get_gpu_status` and `nim_list_models`. The agent returns a single, synthesized answer, giving you instant answers without touching a dashboard.

## NVIDIA NIM MCP: Get Hardware Metrics & Control

Gone are the days of manually cross-referencing `nvidia-smi` output with Prometheus charts. You can now ask your agent to execute a full audit, using tools like `nim_get_metrics` and `nim_get_metadata` in one go.

The difference is control. You don't just view metrics; you use them. Your agent doesn't stop at reporting low memory—it can trigger the fix by calling `nim_scale_replicas`. That’s the operational power you get.

# NVIDIA NIM: 8 Tools for Infrastructure Control

Use these tools to govern hardware limits, extract raw performance metrics, and manage the scaling of AI container deployments.

| # | TOOL | DESCRIPTION |
|---|---|---|
| 01 | `nim_check_health_live` | Runs a liveness check to see if the physical host container orchestrator is running and responsive. |
| 02 | `nim_check_health_ready` | Confirms that the GPU inference layers have finished loading all necessary model artifacts for use. |
| 03 | `nim_get_container_logs` | Retrieves execution parameters and standard output logs from the container orchestrator layer. |
| 04 | `nim_get_gpu_status` | Reads and formats active hardware memory variables, showing you the GPU's topological limits. |
| 05 | `nim_get_metadata` | Pulls core engine execution metrics, mapping out the foundational configuration bounds currently loaded. |
| 06 | `nim_get_metrics` | Extracts comprehensive hardware scaling and performance metrics directly from Prometheus endpoints attached to NIM. |
| 07 | `nim_list_models` | Dumps a list of all active LLMs that are allocated as inference targets on the backend array. |
| 08 | `nim_scale_replicas` | Automatically adjusts the number of hardware replicas, scaling the execution layers up or down dynamically. |

# See It in Action

Real prompts you can use once this MCP is connected to your AI agent through Vinkius Cloud.

**U** Analyze container limits executing active native probes mapped on the physical server to check explicit liveness natively securely.

Parsed logically evaluating native NIM bound ( `check_health_live` ). Inference container executed successfully actively returning 200 HTTP cleanly bounding constraints accurately.

**U** Dump active LLM targets explicitly listing matrices isolating natively loaded models natively secure.

Tunnel explicitly active limit targets mapped isolating model targets safely ( `list_models` ). Extracted cleanly formatting 'meta/llama3-8b-instruct' dynamically actively routing safely natively.

**U** Extract explicit proxy hardware telemetry strictly extracting native GPU metrics logically evaluating bounds attached to the docker bounds natively.

Execution telemetry directly extracted natively utilizing metric parameters securely matching `get_metrics` . Parsed arrays successfully formatting structural mappings mapping explicitly memory bounds efficiently.

# Frequently Asked Questions

01 **How do I check if my NIM container is alive using nim_check_health_live?**

You invoke `nim_check_health_live` to run a liveness probe. This checks the physical host orchestrator's status, telling you immediately if the core service layer is responsive or down.

02 **Does nim_get_gpu_status show total memory or used memory?**

It shows both the topological limits and the currently allocated memory parameters. This allows you to calculate available headroom, which is crucial for capacity planning.

03 **What should I use if I need detailed performance data? Is nim_get_metrics correct?**

Yes, `nim_get_metrics` is the right tool. It pulls Prometheus-formatted hardware scaling metrics directly from the orchestrator, giving you raw, quantitative data points.

04 **If I increase traffic, how do I manage capacity with nim_scale_replicas?**

You call `nim_scale_replicas` and provide the desired replica count. The MCP handles the dynamic orchestration of scaling the execution layers up or down safely.

05 **What is the difference between nim_list_models and nim_get_metadata?**

Use `nim_list_models` for a simple, clean dump of which LLMs are loaded. Use `nim_get_metadata` to pull deeper information about the foundational configuration bounds themselves.

# Go Live in 60 Seconds

Get your connection token from **cloud.vinkius.com**, then paste the endpoint URL into any MCP-compatible client.

YOUR MCP ENDPOINT

```
https://edge.vinkius.com/[TOKEN]/mcp
```

| CLIENT | WHERE TO CONFIGURE |
|---|---|
| Claude AI | Profile → Customize → Connectors → "+" → Add custom connector → Paste endpoint |
| Cursor | Settings → Features → MCP Servers → "+ Add New MCP Server" → Type: SSE → Paste endpoint |
| VS Code | Ctrl/Cmd+Shift+P → "MCP: Add Server" → add `"nvidia-nim": { "url": "…" }` |
| Windsurf | MCP Settings → `mcp_settings.json` → Add endpoint URL |
| ChatGPT | Settings → Tools & plugins → Add MCP server → Paste endpoint |
| Gemini | Extensions → Add MCP Server → Paste endpoint URL |

ASK AN AI ABOUT THIS

Let your preferred AI explain this MCP server

- Ask ChatGPT
- Ask Claude
- Ask Perplexity
- Ask Gemini
- Ask Grok

READY TO CONNECT

# NVIDIA NIM is live on Vinkius Cloud.

Get your connection token, paste it into your AI agent, and start building. No SDK. No deployment. Just results.

**Start at cloud.vinkius.com →**

vinkius.com · support@vinkius.com

## INDEPENDENT PLATFORM DISCLAIMER

Vinkius is an independent platform and is not affiliated with, endorsed by, sponsored by, verified by, or otherwise authorized by NVIDIA NIM. All third-party trademarks, logos, and brand names are the property of their respective owners. Their use in this document is strictly for informational purposes to identify service compatibility and interoperability.

## DOCUMENT INFORMATION

| | |
|---|---|
| Generated | June 2026 |
| MCP Server | NVIDIA NIM MCP |
| Server ID | `019d75e1-524a-72aa-954d-9d9dff56be4b` |
| Platform | Vinkius Cloud for AI Agents |
| Endpoint | `https://edge.vinkius.com/{token}/mcp` |

## LICENSE & USAGE

This document is generated automatically by the Vinkius PDF Engine. Content reflects the MCP server configuration at the time of generation and may change as updates are deployed. For the most current information, visit vinkius.com/mcp/nvidia-nim.

© 2026 Vinkius · vinkius.com · support@vinkius.com