MCP SERVER | NO CODE | CLOUD HOSTED

Nomad MCP

Control Your Workloads Via Conversation

HashiCorp Nomad MCP connects your AI client directly to your cluster. This lets you manage complex workloads, check node health, and track deployments using natural conversation. You stop clicking through dashboards; you just ask your agent what's going on with your infrastructure.

A+ Quality Score 100/100

workload-orchestration | container-management | cluster-monitoring | deployment | infrastructure-as-code

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

SCAN TO VIEW

The infrastructure that powers AI agents in the real world.

Vinkius connects AI to the world's software through secure, enterprise-grade infrastructure — enabling real-world execution at scale, built on the Model Context Protocol (MCP).

Your AI Connections Run Through Vinkius Cloud

The world's largest managed MCP catalog

Vinkius is the cloud infrastructure where AI agents connect 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.

HashiCorp Nomad MCP

10 tools available
Cloud-hosted on Vinkius

You need visibility into a constantly moving target: your production cluster. Instead of juggling the Nomad UI to find out if an allocation succeeded or why a node is showing degraded health, this MCP lets you talk to your orchestration layer. Your AI client interprets your request and runs the necessary checks against your live environment. You can list all running jobs and instantly see their configurations, monitor resource usage across every client node, or even manage rollbacks by failing an underperforming deployment. It's about operational control without context switching. By connecting to this MCP through Vinkius, you give your agent a single pane of glass for infrastructure management—whether it's from Cursor in your IDE or Claude on your desktop. You get direct access to the state of every job and every service running right now.

Core Capabilities

01 — Check Cluster Status

List all registered client nodes and retrieve current resource usage metrics.

02 — Inspect Workloads

Retrieve detailed information about running task allocations, including specific task details.

03 — Review Job Definitions

Fetch the complete configuration and current status for any registered job type.

04 — Monitor Deployments

Track progress on rolling updates or get specific details about past deployments.

05 — Control Rollbacks

Manually promote a successful deployment or fail an underperforming one to trigger rollbacks.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to this MCP and provide your Nomad Address and optional ACL Token.

02 Connect your preferred AI client (like Cursor or Claude) to the Vinkius catalog.

03 Tell your agent what you need—for example, 'Show me all nodes that are down'—and it executes the query directly against your cluster.

The bottom line is: Your AI client turns complex infrastructure APIs into simple conversation prompts.

Built For

This MCP is for the ops engineer who's tired of clicking through dashboards at 2am. It targets anyone whose job requires constant, precise interaction with a live, high-stakes infrastructure state.

SRE Engineer

Needs to quickly check cluster health or validate complex job statuses without opening the Nomad UI and navigating multiple tabs.

DevOps Automation Specialist

Manages service rollouts by monitoring deployment progress, promoting successful versions, or triggering failovers on bad ones.

Infrastructure Manager

Automates the retrieval of node and workload data needed for compliance reporting and performance audits.

What Changes When You Connect

01 Stop opening the Nomad UI just to check status. You can ask your agent for a list of jobs or node details and get an immediate, structured answer.

02 Need to know what's running? Use `list_allocations` to see every active task instance without navigating through multiple dashboards.

03 Handling bad deployments is simple. Instead of manually clicking a rollback button, you can tell your agent to use `fail_deployment` and trigger the necessary recovery.

04 Get deep insights by using `get_job` or `get_node` with unique IDs. This gives you metadata for specific components that general listing tools miss.

05 The process of deployment management is now conversational. You can follow progress via `list_deployments` and then manually move things forward using `promote_deployment` when ready.

Real-World Applications

Investigating a failing service during peak hours

A developer notices high error rates. They ask their agent to check the cluster status, which immediately uses `list_nodes` to flag two nodes as unreachable. The agent then runs `get_node` on those specific IDs and identifies a resource saturation issue, giving them the exact fix location.

Auditing compliance for infrastructure changes

An infra manager needs to report on which services were deployed last week. They use their agent to run `list_deployments` and then pull detailed records using `get_deployment` IDs, compiling a ready-made audit trail without manual data export.

Mid-rollout correction

A deployment is stuck at 50% completion. The engineer tells the agent to check the status (`list_jobs`), determines which phase failed, and then uses `promote_deployment` on the problematic version ID to force the update forward.

Troubleshooting a single bad task

A service is intermittently failing. Instead of looking at the general job status, the agent runs `list_allocations`, finds the specific allocation instance, and uses `get_allocation` to read the exact error logs for that one task.

Patterns to Avoid

Treating it like a general monitoring tool

✗ AVOID

Thinking you can use this MCP just to see basic CPU load across your whole cloud provider, or check application logs outside of Nomad.

✓ INSTEAD

This is strictly for workload orchestration. If you need generalized metrics from AWS CloudWatch or Azure Monitor, those are external API tools. Use `list_nodes` only for Nomad-specific status.

Trying to write complex shell scripts

✗ AVOID

Writing a complicated Bash script with nested loops and error handling to iterate through job IDs and check their status manually.

✓ INSTEAD

Let your agent handle the looping. Instead of scripting, just ask: 'List all jobs that are currently in an unstable state.' The agent runs `list_jobs` internally and filters the results for you.

Ignoring unique identifiers

✗ AVOID

Asking for 'the latest deployment' without specifying which service or environment, leading to vague, massive data dumps.

✓ INSTEAD

Be precise. Use `get_deployment` and specify the exact ID or name you want details on. For example: 'Get job details for API-Gateway using ID XYZ.'

The Right Fit

Use this MCP when your core problem is understanding the *state* of running services in a container orchestration cluster like Nomad. You need to know who's running, what they are configured to do, and if they passed their health checks. If you can ask 'What is the current status of my job X?'—this is for you. Don't use it if your problem is pure data transformation (like mapping JSON fields), or if you need to interact with non-container services (like a database write). For those cases, look for dedicated API connector tools in the Vinkius catalog. If you just want to read basic system metrics unrelated to job lifecycle management, an infrastructure monitoring tool will be better.

The Headache of Cluster Health Checks

Today, checking if a deployment went sideways means opening the Nomad UI. You click into the cluster view, then select the job name, then look at allocations to see if they succeeded or failed. If you need resource usage details, you might have to switch tabs and copy-paste IDs from one dashboard widget to another just to verify the status.

With this MCP, that friction vanishes. You simply ask your agent for the cluster state. It runs `list_nodes` and immediately tells you exactly which nodes are healthy and what their resource usage is. The information you need—the who, the where, and the how much—is delivered in a single, conversational answer.

Job & Node Management

The manual process of tracking deployments involves constantly cross-referencing multiple dashboard views. You have to check `list_jobs` for the type, then drill down into specific deployment records using `get_deployment`, and finally verify the task status with `get_allocation`. This takes minutes of clicking and copy/pasting.

Now you get immediate actionability. Your agent consolidates all that data. You can ask it to check a service's job details, or even tell it to use `fail_deployment` if things go wrong—all without touching the UI. It's instant control.

HashiCorp Nomad: 10 Tools for Cluster Management

These tools allow you to programmatically interact with every aspect of your cluster state, from listing job definitions to managing specific allocations.

#	TOOL	DESCRIPTION
01	`fail_deployment`	Marks a deployment as failed, typically to initiate an automated rollback process.
02	`get_allocation`	Retrieves specific operational details for a single task allocation instance.
03	`get_deployment`	Fetches detailed information about a particular deployment cycle.
04	`get_job`	Gets specific configuration and status details for a registered job type.
05	`get_node`	Retrieves detailed resource usage and operational status for a single cluster node.
06	`list_allocations`	Generates a list of all currently running task allocations across the cluster.
07	`list_deployments`	Provides an overview and history of recent deployment activities.
08	`list_jobs`	Lists every registered job type within your Nomad cluster.
09	`list_nodes`	Lists all client nodes connected to the cluster, showing their current status.
10	`promote_deployment`	Manually advances a deployment cycle to move it to a higher operational state.

See It in Action

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

U List all active jobs in the 'production' namespace.

Retrieving jobs... In the 'production' namespace, I found 5 active jobs: 'API-Gateway' (Service), 'Redis-Cluster' (Service), and 3 batch processing jobs. Would you like the detailed status for the API Gateway?

U Check the status of all client nodes in the cluster.

Fetching cluster nodes... You have 12 registered client nodes. 10 are 'ready' and healthy, while 2 are currently 'down' or unreachable. Shall I retrieve the resource usage for the healthy nodes?

U Promote the deployment with ID 'dep-98765'.

Promotion initiated... The deployment 'dep-98765' has been successfully promoted. The rolling update will now continue to the remaining allocations. I'll monitor the status for you.

Frequently Asked Questions

01 **How do I list all running services with HashiCorp Nomad MCP?**

You run 'list jobs.' This tool gives you an overview of every job type registered in your cluster, providing the necessary context to know what workloads are available.

02 **Can I check the health of a specific node using HashiCorp Nomad MCP?**

Yes. You use 'get node' and provide the unique node ID. This fetches detailed resource usage metrics, letting you pinpoint exactly why that machine might be struggling.

03 **What does promoting a deployment with HashiCorp Nomad MCP actually do?**

The 'promote deployment' tool advances a running job cycle to the next stage. You use it when a canary release has proven stable and you want to move the entire service up.

04 **Is this MCP used for pure coding tasks or infrastructure?**

This is purely for infrastructure control and monitoring. Use it to check job status, list nodes, and manage deployments; don't use it to write application logic.

05 **Which tool should I use if I only want task details?**

Use 'get allocation.' This provides the most granular data point—the specific operational status of a single, running task instance.

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 `"hashicorp-nomad": { "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

HashiCorp Nomad 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 HashiCorp Nomad. 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	HashiCorp Nomad MCP
Server ID	`019d75df-2922-71d2-91fb-560edb5fcfdf`
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/hashicorp-nomad.