MCP SERVER | NO CODE | CLOUD HOSTED

Cilium (eBPF Networking) MCP for AI Agents

Monitor and manage Kubernetes node networking status with eBPF visibility

Cilium MCP provides natural language control over your Kubernetes eBPF networking stack. Connect it to your AI agent to inspect cluster nodes, monitor daemon health, and manage network endpoints without writing complex CLI commands. It's how you get deep visibility into container connectivity and security policies through simple conversation.

A+ Quality Score 100/100

ebpf | kubernetes-networking | cilium | network-security | cluster-management

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/cilium-ebpf-networking

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.

Cilium (eBPF Networking) MCP

6 tools available
Cloud-hosted on Vinkius

This MCP connects directly to your Cilium agent, letting your AI client talk to the core of your Kubernetes networking layer. Instead of running a dozen different `kubectl` commands just to audit cluster health, you ask your agent what's wrong with things. It can pull detailed status reports on every node, check if the main daemon is healthy, and even help you manage network endpoints to secure container traffic.

When you connect this MCP via Vinkius, you get access through any compatible AI client. You'll use natural language to list nodes, inspect configurations, or create new connectivity points for services. It simplifies complex networking management into simple dialogue. This means platform engineers and security teams can validate network policies and troubleshoot connectivity issues faster than ever before.

Core Capabilities

01 — Audit Cluster Node Status

Retrieves detailed information about every node currently known to the Cilium agent.

02 — Check Daemon Health and Connectivity

Determines the operational status of the Cilium daemon, container runtime, datastore, and Hubble connection.

03 — Manage Core Networking Settings

Allows inspection and modification of key daemon configuration options and datapath modes on demand.

04 — Inspect Network Connectivity Points

Looks up the status, assigned IP, and labels for specific networking endpoints.

05 — Establish New Endpoints

Creates new network connectivity endpoints to enforce container security boundaries.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/cilium-ebpf-networking** — connect your AI agent in three steps.

01 Subscribe to this MCP and provide your Cilium API URL credentials.

02 Select your preferred AI client (Claude, Cursor, etc.) within the Vinkius Marketplace.

03 Ask your agent a question—like 'What is the health of the networking daemon?' or 'List all nodes'—and it executes the required checks.

The bottom line is that you talk to your AI agent like you're talking to an ops team member, and it runs the necessary commands behind the scenes.

Built For

Platform engineers who spend too much time running repetitive CLI health checks. It's for security teams needing real-time network policy validation, or SREs troubleshooting complex connectivity issues at 2 AM.

DevOps Engineer

Uses this MCP to quickly audit the overall cluster health and node status across environments without juggling multiple command lines.

Site Reliability Engineer (SRE)

Queries daemon configurations and connectivity metrics to pinpoint exactly why a microservice lost network access or is running slowly.

Security Analyst

Inspects endpoint labels and states to verify that the intended network policy enforcement is active across all containers.

What Changes When You Connect

01 Instantly audit cluster health. You can ask the agent to check the daemon's operational status using `get_healthz` —no need for a dashboard deep dive.

02 Manage network endpoints conversationally. Use `create_endpoint` and `get_endpoint` to define and inspect container connectivity points, which is critical for policy enforcement.

03 Simplify complex debugging. Need to know why traffic isn't flowing? Query daemon settings or run the agent using `get_config` to see exactly what rules are in place.

04 Streamline node visibility. Instead of running multiple commands, use `get_cluster_nodes` to get a comprehensive list and status report for every machine in your cluster.

05 Fine-tune network policy on the fly. If you need to change a setting, simply request it via `patch_config`, making configuration management much less error-prone.

Real-World Applications

Diagnosing intermittent pod connectivity issues

A user asks the agent, 'Why can't service X talk to Y?' The agent runs checks using `get_healthz` and then inspects endpoint labels with `get_endpoint`. It reports that a required label is missing on the destination node, solving the issue in minutes.

Onboarding a new cluster segment

A platform engineer runs 'list all nodes' using `get_cluster_nodes` to confirm every machine is online. They then use `patch_config` to apply baseline networking rules before deploying services.

Security audit of network policy enforcement

A security team member asks the agent to list all active endpoints and their assigned IP addresses using a combination of tools. This confirms that every critical service has the correct, tightly scoped connectivity boundaries enforced.

Scaling up services requiring new routes

A developer needs a new connection point for a feature flag rollout. They simply ask the agent to establish it using `create_endpoint`, rather than manually writing and applying network manifests.

Patterns to Avoid

Treating networking like simple DNS lookups

✗ AVOID

A user tries to 'check if 10.24.1.5 is reachable.' This only confirms basic network reachability, ignoring the complex eBPF policy layer and security context that Cilium manages.

✓ INSTEAD

To check actual connectivity status and policies, ask your agent to inspect an endpoint by ID using `get_endpoint` or run a full health audit with `get_healthz`. This verifies both reachability *and* policy compliance.

Manual configuration updates via YAML files

✗ AVOID

The user writes, 'I need to patch the datapath mode on node 3.' They then have to manually edit and apply a large, complex YAML file that might contain syntax errors.

✓ INSTEAD

Instead of editing manifests, ask your agent to modify settings using `patch_config`. You describe *what* you want to change in plain language, and the MCP handles generating and applying the correct configuration patch.

Running checks without context

✗ AVOID

A user runs 'list all nodes' but doesn't know which node is actually experiencing service failure. The output is too big to parse manually.

✓ INSTEAD

First, use `get_healthz` to confirm the overall system health. Then, ask the agent to filter and compare node details using `get_cluster_nodes`, focusing only on nodes in a specific availability zone or cluster segment.

The Right Fit

Use this MCP if your primary pain point is translating complex Kubernetes networking knowledge into actionable steps that don't require constant context switching between CLI tools. It’s ideal when you need to audit, view, and modify network policy endpoints or node configurations without writing YAML files. However, don't rely on it for deep-level kernel debugging or custom eBPF program development—that still requires specialized tooling. If your goal is simply viewing basic IP addresses, a standard `kubectl get pods` might suffice; but if you need to understand the *policy* governing those IPs, this MCP is essential.

Cilium (eBPF Networking) MCP: Auditing Kubernetes Network Policies

Right now, auditing network policy status means running a dozen different `kubectl` commands. You check nodes with one command, then use another to list endpoints, and run yet a third just to see the daemon's current configuration—all while manually cross-referencing timestamps and output blocks.

With this MCP, you simply ask your agent to 'Show me the full policy status for node X.' It aggregates that information using tools like `get_cluster_nodes` and `get_endpoint`, delivering a single, cohesive report in plain language. You get visibility without the command-line headache.

Cilium (eBPF Networking) MCP: Troubleshooting Daemon Health

Troubleshooting networking usually involves logging into a machine, checking service logs, and manually verifying the daemon's health status by running multiple checks. It’s tedious, slow work that often requires escalating to specialized network engineers.

This MCP centralizes all diagnostics. A single prompt triggers checks using `get_healthz` and allows you to inspect configuration details with `get_config`. You get immediate confirmation on whether the networking stack is green or if a specific setting needs adjustment.

Cilium (eBPF Networking) MCP: 6 Tools for Network Management

These tools let your agent perform every major networking operation, from listing cluster nodes to creating new connectivity points and modifying the daemon configuration.

#	TOOL	DESCRIPTION
01	`create_endpoint`	Creates a new, managed networking connectivity point for specific services.
02	`get_config`	Retrieves the current detailed configuration settings for the Cilium daemon.
03	`get_endpoint`	Looks up the status and details of a specific networking endpoint using its ID.
04	`get_healthz`	Checks and reports the overall operational health status of the Cilium daemon.
05	`get_cluster_nodes`	Gathers and lists information about all known cluster nodes managed by the agent.
06	`patch_config`	Modifies existing daemon configuration settings directly through natural language input.

See It in Action

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

U Can you list all nodes in the cluster and tell me their current IP ranges?

Cluster Node Report

NODE NAME	STATUS	PRIMARY CIDR RANGE	LAST CHECK
node-101	Ready	10.244.0.0/16	minutes ago
node-102	Degraded	10.244.1.0/16	hours ago (Needs Attention)
node-103	Ready	10.244.2.0/16	minutes ago

Note: Node 102 shows degraded status due to potential datastore sync issues.

U I think my frontend service connectivity broke. Can you check the endpoint details for me?

Endpoint Details Report

🔍 **ID:** `cilium-local:99`

- **State:** ✅ Ready
- **Assigned IP:** 10.244.5.10
- **Labels Found:** app=frontend, env=prod
- **Last Seen:** seconds ago

The endpoint is active and appears correctly labeled for production traffic.

U What's the current configuration for datapath mode? I need to verify it.

Cilium Daemon Configuration

- **Datapath Mode:** eBPF (Active)
- **Fallback Mode:** Kernel Bridge (Inactive)
- **Max Connections:** 50,000
- **Scope:** Global

Current configuration confirms that the system is running in native eBPF mode. This means all network policy enforcement happens at the kernel level.

Frequently Asked Questions

01 **How does the Cilium (eBPF Networking) MCP help me troubleshoot my Kubernetes networking?**

It turns complex, multi-step CLI commands into simple conversations. You ask your agent about a connectivity issue—for example, 'Why is service X unreachable?'—and it runs multiple underlying checks to give you a clear diagnosis of the network policy failure or node status.

02 **Do I need to be an expert in eBPF networking to use this MCP?**

No. This MCP lets you interact with complex concepts using plain English. You tell your agent what you want to check—like 'the node status' or 'daemon health'—and it handles the technical execution for you.

03 **Can I use the Cilium (eBPF Networking) MCP to scale new services?**

Yes. If your service needs a dedicated network connection point, you can ask the agent to create one using `create_endpoint`. This ensures the new service gets proper security policy enforcement immediately.

04 **What if I need to change a setting on my cluster? Can this MCP do it?**

You can. Instead of manually editing configuration files, you describe the desired change (e.g., 'Increase max connections') and the agent uses tools like `patch_config` to apply the modification safely.

05 **Is this MCP better than just using standard kubectl commands?**

It's better for speed and scope. Standard commands are single-purpose; this MCP aggregates data from multiple sources—node status, daemon health, endpoints, and config—into one chat response, saving massive amounts of time.

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 `"cilium-ebpf-networking": { "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

Cilium (eBPF Networking) 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 Cilium (eBPF Networking). 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	Cilium (eBPF Networking) MCP
Server ID	`019e3877-1278-72c6-9ffb-12cfc2cdf355`
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/cilium-ebpf-networking.