MCP SERVER | NO CODE | CLOUD HOSTED

Gatus Health Dashboard MCP

Check System Status with Natural Language Queries

Gatus Health Dashboard MCP lets you monitor your entire service infrastructure using natural conversation. List every monitored endpoint, check its real-time health status, and pull performance metrics—all without leaving your AI agent. It turns complex dashboard checking into simple questions.

A+ Quality Score 100/100

service-health | uptime-monitoring | infrastructure-alerts | real-time-status | dashboard

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/gatus-health-dashboard

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.

Gatus (Health Dashboard) MCP

4 tools available
Cloud-hosted on Vinkius

You don't need to open a dozen tabs or run manual scripts just to see if your services are up. By connecting this MCP, your AI client becomes a 24/7 Site Reliability Engineering assistant. You can ask it to list all monitored endpoints and instantly get their current health status across the whole system. Need to know why a service is slow? Ask for performance statistics on that specific endpoint, or pull raw metrics data suitable for deep analysis. This capability means you stop reacting to alerts and start asking questions. The entire catalog of Vinkius makes it simple; connect once from your preferred AI client and get instant visibility into system health.

It's about getting immediate answers: What is the status? How fast was it last time? Did it fail five minutes ago? Your agent handles the complexity, giving you direct insight into service availability when you need it most.

Core Capabilities

01 — Check overall system health

List every monitored endpoint and instantly see its current status across your entire infrastructure.

02 — Inspect specific endpoint history

Drill down into a service to view recent results, historical statuses, and check for patterns of failure or success using its unique key.

03 — Analyze performance metrics

Get detailed statistics on an endpoint's speed and reliability, identifying potential latency issues before they become outages.

04 — Retrieve raw data streams

Access Prometheus-compatible metrics for highly customized reporting or deep technical analysis outside the standard dashboard view.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/gatus-health-dashboard** — connect your AI agent in three steps.

01 Subscribe to this MCP and provide your specific Gatus instance URL.

02 Connect your AI agent to Vinkius. The connection authenticates your access credentials for the monitoring service.

03 Ask your agent a question, like 'What is the status of the payment gateway?' Your agent calls the necessary tools and returns the structured data.

The bottom line is you get instant system health reports without ever having to manually interact with a web dashboard or run terminal commands.

Built For

This MCP is for ops engineers who are tired of clicking through multiple dashboards at 2 AM. It's essential for SRE teams that need deep performance metrics on demand, and product owners who require quick, high-level reports during an incident.

DevOps Engineer

Quickly audit the health of multiple services and retrieve raw metrics directly through natural language queries instead of leaving the terminal.

SRE Team Member

Investigate performance regressions or endpoint history by asking targeted questions about latency and uptime over time.

Product Owner

Get high-level, natural language status reports on system availability during an outage without needing to understand the underlying infrastructure metrics.

What Changes When You Connect

01 Instant Visibility: Instead of manually navigating a dashboard to see if Service X is up, ask your agent. It uses the `list_endpoints` tool to give you an immediate summary of all monitored services.

02 Pinpoint Failures: If something isn't working, don't guess. Use `get_endpoint_health` to check a specific service and see its most recent failure details instantly.

03 Diagnose Slowness: When latency spikes, use `get_endpoint_stats`. This tool shows you the average speed and 99th percentile data so you know exactly where performance is dropping.

04 Deep Dive Analysis: Need to write a report or build custom tooling? Use `get_metrics` to pull raw, Prometheus-compatible data that goes far beyond simple status checks.

05 Operational Efficiency: Your AI agent acts as the ultimate SRE assistant, eliminating the need for multiple context switches between dashboards and terminals.

Real-World Applications

Investigating an intermittent API outage

The Ops Engineer sees a ticket about fluctuating API availability. Instead of checking three different monitoring panels, they prompt their agent: 'What is the status and performance stats for the user-profile service?' The agent uses `get_endpoint_health` and then `get_endpoint_stats`, providing both current failure data and historical latency metrics in one response.

Pre-launch system readiness check

The QA lead needs to confirm every single dependency is green before deployment. They ask the agent to 'List all endpoints.' The agent uses `list_endpoints`, giving a definitive, real-time status report on the entire connected infrastructure.

Debugging slow payment processing

The Product Owner notices payment transactions are slowing down. They ask the agent to check performance for 'payment-gateway.' The agent uses `get_endpoint_stats`, confirming if the average latency is spiking and pinpointing exactly which metric changed.

Building custom compliance reports

The Security team needs raw data on all service availability over a period for auditing. They ask the agent to 'Export Prometheus metrics for core services.' The agent executes `get_metrics`, delivering the structured, technical payload required for external analysis.

Patterns to Avoid

Using the wrong tool for the job

✗ AVOID

Asking 'How fast is the API?' when you really need historical data. Just checking status (`get_endpoint_health`) only tells you if it's up or down *right now*.

✓ INSTEAD

To check speed and reliability, you must use `get_endpoint_stats`. This tool gives you latency metrics (average, 99th percentile) that show performance trends, not just current status.

Assuming raw data is simple

✗ AVOID

Trying to interpret a massive stream of raw metrics without knowing the format. It looks like garbage text.

✓ INSTEAD

If you need raw, deep-dive data for custom tools or analysis, use `get_metrics`. This tool provides structured Prometheus data ready for your specialized reporting systems.

Ignoring the scope of the check

✗ AVOID

Asking for 'all metrics' when you only care about one service. You get a huge, unmanageable wall of text.

✓ INSTEAD

Always specify which endpoint you need data on. Use `get_endpoint_stats` or `get_endpoint_health` and name the exact service key to keep the output focused.

The Right Fit

Use this MCP if your workflow requires checking system availability, latency, or historical performance metrics across multiple services in a conversational way. If you need to know 'Is it up?'—this works perfectly. However, don't use this if your only goal is simple notification delivery; for that, a dedicated alerting service might be

better suited. Also, if you are building a front-end dashboard and already have the data stream coming from another source, you may not need these tools. But if you are in an IDE or terminal environment and need to query live operational status without context switching, this is exactly what you want. It's your primary tool for real-time infrastructure insight.

The Dashboard Fatigue Problem

Today, checking if a multi-service application is healthy means logging into the monitoring portal, clicking on the main dashboard. Then you have to open tabs for 'Auth API,' 'Payment Gateway,' and 'User Profiles.' For every single service, you check the green light, copy down the status code, and then manually compare performance metrics like latency or error rates across all those individual views.

With this MCP, you simply ask your agent: 'What's the health of our core services?' Instead of clicking through a dozen screens, you get an immediate, consolidated report. The AI gathers that information using tools like `list_endpoints`, giving you one definitive answer instead of twenty clicks and ten minutes.

Get Status Checks with get_endpoint_health

Before, finding out the status of a specific component meant navigating to that service's dedicated dashboard page. If you were debugging an issue in another tool, you had to stop everything and manually check that other panel just to see if it was working.

Now, your agent handles that lookup instantly. You can ask for `get_endpoint_health` on demand, getting the most current status without disrupting your coding or analysis flow. It's always available when you need it.

Gatus Health Dashboard: 4 Tools

These tools let you query every aspect of your monitored system, from basic endpoint listing to deep performance metric retrieval.

#	TOOL	DESCRIPTION
01	`get_endpoint_health`	Retrieves the current health status and recent results for one specific service endpoint.
02	`get_endpoint_stats`	Pulls key performance statistics, like average latency, for a single designated endpoint.
03	`list_endpoints`	Provides a comprehensive list of all monitored services and their current operational status across the entire system.
04	`get_metrics`	Fetches raw, technical performance metrics data from Gatus for custom reporting and deep analysis.

See It in Action

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

U List all monitored endpoints and their current status.

I've retrieved the status for 8 endpoints. 7 are 'Healthy' and 1 ('payment-gateway') is currently 'Unhealthy'. Would you like me to investigate the 'payment-gateway' results?

U What is the health status of the 'core-api' endpoint?

The 'core-api' endpoint is currently Healthy. The last check was successful with a 200 OK response and a latency of 45ms. Recent history shows 100% uptime over the last 24 hours.

U Show me the performance statistics for 'database-service'.

Fetching stats for 'database-service'... Average latency is 12ms, with a 99th percentile of 85ms. There have been 0 failures in the last 500 requests.

Frequently Asked Questions

01 **How do I list all monitored endpoints using Gatus Health Dashboard MCP?**

You use the `list_endpoints` tool. This instantly gives you a comprehensive, real-time roster of every service connected to your monitoring system and its current status.

02 **Can I check performance statistics with get_endpoint_stats?**

Yes, using `get_endpoint_stats`, you can retrieve detailed metrics for a single endpoint. This shows more than just 'up' or 'down,' giving you average latency and reliability data.

03 **What is the difference between get_endpoint_health and get_metrics?**

Use `get_endpoint_health` for a simple, current status check. Use `get_metrics` when you need raw, structured Prometheus-compatible data for deep technical analysis or custom reporting.

04 **Does Gatus Health Dashboard MCP work with multiple services?**

Absolutely. The entire MCP is designed to monitor your full infrastructure stack. You can ask about a mix of services, and the agent will use multiple tools to gather all the necessary data.

05 **Can I check an endpoint that failed last week using get_endpoint_health?**

While `get_endpoint_health` gives recent results, for historical trends or deep dives into past failures, you should use `get_endpoint_stats`, which tracks performance over 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 `"gatus-health-dashboard": { "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

Gatus (Health Dashboard) 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 Gatus (Health Dashboard). 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	Gatus (Health Dashboard) MCP
Server ID	`019e389c-e4fa-730d-b998-65c987f911b8`
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/gatus-health-dashboard.