MCP SERVER | NO CODE | CLOUD HOSTED

Better Stack MCP for AI Agents

Manage infrastructure uptime and incident response via conversation

Better Stack connects your infrastructure monitoring, incident management, and on-call coordination into a single conversation stream. It lets you talk to your service health data—checking if an HTTP endpoint is down, listing active incidents, or finding out who's covering the system this week—all directly from your AI client.

A+ Quality Score 100/100

uptime-monitoring | incident-response | sre | alerting | heartbeat-monitoring | infrastructure-monitoring

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

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.

Better Stack MCP

10 tools available
Cloud-hosted on Vinkius

Forget clicking through multiple dashboards just to check if your API gateway is up or who should be paged at 3 AM. This MCP connects your Better Stack account, letting you manage complex infrastructure monitoring and incident details using only natural conversation.

It lets you list all your HTTP monitors, verify the status of critical endpoints, and even pull historical records for post-mortems. Need to know who's responsible for system health this month? You can check current on-call schedules right away. Monitoring cron jobs via heartbeat logs or seeing if a public status page is accurate are simple commands. All these deep operational checks run through your AI client, giving you real-time context without ever leaving your workspace. By connecting it through the Vinkius catalog, you get instant access to this powerful monitoring layer alongside thousands of other tools.

It’s about turning a complex web of dashboards into simple questions.

Core Capabilities

01 — View and manage service uptime monitors

Create, read, update, or delete specific HTTP, Ping, or Keyword monitors to ensure service availability.

02 — Review current system incidents

Get real-time updates on active outages or pull detailed reports on past incidents for root cause analysis.

03 — Check team rotation schedules

List who is currently assigned to be on call, viewing rotations and coverage until a specific date.

04 — Verify scheduled job health

Retrieve logs for heartbeat monitors to track the status of recurring tasks or cron jobs.

05 — Check public service status pages

List and verify the current published status page details without logging into a separate dashboard.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to this MCP in your preferred AI client.

02 Enter your Better Stack API token to authenticate access.

03 Ask your agent specific questions, like 'List all monitors that are currently down,' and it pulls the live data.

The bottom line is you get conversational control over complex infrastructure health data.

Built For

This MCP is for DevOps engineers and SREs who hate context switching. If you spend your day jumping between monitoring dashboards, needing immediate incident status or on-call details during an outage, this tool cuts out the clicks.

Site Reliability Engineer (SRE)

Running post-mortems by retrieving historical reports for past incidents and checking if existing monitors need updating.

DevOps Engineer

Quickly verifying the health of background processes by listing heartbeat logs or making sure new services are covered with monitors.

Backend Developer

Confirming that a scheduled cron job ran successfully by checking its dedicated heartbeat status right from their coding environment.

What Changes When You Connect

01 Instant Incident Visibility: Get immediate, conversational updates on active incidents or historical reports using the `list_incidents` tool. No more manual dashboard dives.

02 Full Monitor Control: You can manage your entire monitoring suite—from listing all monitors to creating a new one (`create_monitor`) or updating settings with `update_monitor` — all via text prompts.

03 Clear Accountability: The `list_on_calls` tool tells you exactly who is responsible for system health, eliminating confusion during an outage and speeding up response times.

04 Background Task Health Check: Use `list_heartbeats` to monitor cron jobs. You can verify if recurring tasks are running on schedule without touching the scheduler UI.

05 Simplified Status Checks: The MCP allows you to check public status pages (`list_status_pages`) and system monitors in one go, consolidating your operational visibility.

Real-World Applications

Finding out why a core service is down

An engineer asks the agent, 'Show me all HTTP endpoints that are failing.' The agent uses `list_monitors` and identifies two critical services. This saves minutes of clicking through multiple dashboard tabs during an active outage.

Determining who to call after hours

A developer asks, 'Who is on-call for the database team this week?' The agent uses `list_on_calls` and provides Sarah Miller's name and rotation dates. This prevents unnecessary wake-up calls and ensures proper escalation.

Investigating a sporadic failure

A manager asks, 'What happened with the primary API last month?' The agent runs `list_incidents`, providing historical reports that help them pinpoint the root cause for reporting purposes.

Verifying scheduled cleanup tasks

An ops engineer needs to confirm if a nightly backup job ran successfully. They ask about heartbeats, and the agent uses `list_heartbeats` to confirm the last ping was successful within the expected window.

Patterns to Avoid

Searching for status by guessing endpoints

✗ AVOID

Trying to check monitor health by manually entering URLs into a general query box, often resulting in vague error messages or incomplete data.

✓ INSTEAD

First, use `list_monitors` to get the exact names and IDs of all services. Then, ask your agent to 'Get details for [Monitor Name]' using `get_monitor` for precise, accurate status checks.

Calling someone without checking coverage

✗ AVOID

An engineer calls a teammate at 10 PM about an outage only to find out the team was already on call or that the issue falls under a different rotation.

✓ INSTEAD

Always start by asking the agent to 'List on-call schedules' using `list_on_calls`. This ensures you talk to the right person immediately, saving time and effort.

Treating monitors as static definitions

✗ AVOID

Assuming a monitor endpoint is always correct without verifying its current status or if it needs adjustment (e.g., when moving services).

✓ INSTEAD

Before assuming anything, use `get_monitor` to pull the detailed specifications for existing monitors. If you need to change them, use `update_monitor` instead of manually editing.

The Right Fit

Use this MCP if your core job requires real-time visibility into operational uptime and incident status via a natural conversation. Specifically, it's perfect for quickly checking monitor health (`get_monitor`), listing recent incidents (`list_incidents`), or confirming on-call coverage (`list_on_calls`). Don't use this if you simply need to generate monitoring code from scratch; that requires general coding assistance. Also, don't rely on it solely for billing data or user management—it only handles infrastructure health. If your primary need is managing users, stick to a dedicated identity MCP instead.

Better Stack MCP: Solving Infrastructure Uptime Monitoring Pain

Today, checking the status of an entire service stack means context switching. You open your dashboard for API uptime; then you switch to a separate tab to see if the database heartbeat is firing; finally, you check a third tool just to confirm who owns the problem. It's manual clicking and copy-pasting across three or four different systems.

With this MCP, those steps disappear. You tell your agent, 'What's wrong with the service stack?' and it orchestrates checks on monitors, heartbeats, and incident logs instantly. The result is a single, actionable summary that tells you exactly what failed and where to start fixing it.

Better Stack MCP: Streamlining Incident Response Coordination

During an actual outage, the worst thing is trying to figure out who's responsible or if a service was already reported. You waste time searching through old tickets and checking multiple on-call calendars.

Now, you can ask your agent about incident history or current rotation using `list_on_calls` or `list_incidents`. This moves the conversation from 'Who should I talk to?' to 'Here's what we need to fix,' dramatically speeding up recovery.

Better Stack: 10 Tools for Uptime Monitoring & Incident Response

Use these tools to manage all aspects of your service health, from listing active monitors to retrieving detailed incident reports.

#	TOOL	DESCRIPTION
01	`create_monitor`	Set up a brand new uptime monitor for a service endpoint.
02	`delete_monitor`	Remove an existing, unused monitoring rule from your stack.
03	`get_incident`	Fetch specific details about one known incident ID.
04	`get_monitor`	Retrieve all technical specifications for a single, named monitor.
05	`list_heartbeats`	Show the status of multiple recurring tasks or automated jobs.
06	`list_incidents`	View a summary list of recent outages and service disruptions.
07	`list_monitors`	Pull a full inventory listing of every uptime monitor you run.
08	`list_on_calls`	See the current and upcoming on-call rotation schedule for your team.
09	`list_status_pages`	List all public status pages associated with your services.
10	`update_monitor`	Modify the settings, endpoints, or frequency of an existing monitor.

See It in Action

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

U What services are currently failing and who should I call about them?

🚨 **ALERT: Service Outage Detected** 🚨

- **Primary API Gateway:** Down (Last check: 5 minutes ago).
- **User Profile Endpoint:** Degraded performance.

I've checked the on-call schedule. For API issues, **Sarah Miller** is currently responsible until Friday. She can be contacted via Slack for immediate details.

U Show me all my monitors and tell me which ones need updating.

Here's an inventory of your 15 active monitors:

MONITOR NAME	STATUS	LAST CHECK	ACTION NEEDED?
Auth Gateway	OK	2 mins ago	No
Legacy API Ping	**DOWN**	4 hours ago	Yes (Needs update)
Billing Cron Job	OK	Just now	No

Please review the 'Legacy API Ping' monitor; its endpoint appears outdated and needs manual adjustment.

U What was the most recent major incident, and what were the root causes?

Incident Report: Core Service Disruption (2 days ago)

- **Status:** Resolved.
- **Impacted Services:** User Authentication, Payment Processing.
- **Root Cause:** A database connection pool exhaustion. The incident was triggered by an unexpected traffic spike during peak hours.

I recommend reviewing the `get_incident` report for full details on mitigation steps.

Frequently Asked Questions

01 **How does Better Stack MCP help me monitor my infrastructure uptime?**

It gives you conversational access to all your monitors. You can ask about specific endpoints, see a full list of services, or check if any are currently down without ever logging into the monitoring dashboard itself.

02 **Can Better Stack MCP tell me who is on call right now?**

Yes, it uses `list_on_calls` to show you current rotations and schedules. This means you never have to guess who owns an issue after hours; the agent tells you immediately.

03 **What kind of incidents can Better Stack MCP help me track?**

You can pull both live alerts on active outages and detailed historical reports for past disruptions. This is useful when reporting to management or doing a post-mortem analysis.

04 **Do I need to worry about cron job status with Better Stack MCP?**

No, you don't. The MCP can list heartbeats, letting you confirm if your recurring background tasks are running on schedule and that their pings are coming through as expected.

05 **How do I use the Better Stack MCP in my daily DevOps workflow?**

You start by asking questions about 'status' or 'schedules.' Instead of opening 5 tabs, you ask your agent to list monitors, check incidents, and verify on-call schedules all in one prompt.

06 **Is Better Stack MCP better than looking at my dashboard manually?**

It's faster and less error-prone. The MCP gives you a consolidated summary of the data you need right now, saving you the minutes spent navigating multiple views just to get an answer.

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 `"better-stack": { "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

Better Stack 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 →

INDEPENDENT PLATFORM DISCLAIMER

Vinkius is an independent platform and is not affiliated with, endorsed by, sponsored by, verified by, or otherwise authorized by Better Stack. 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	Better Stack MCP
Server ID	`019d755b-54fc-7376-be99-830bc16ae734`
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/better-stack.