MCP SERVER | NO CODE | CLOUD HOSTED

Healthchecks.io MCP

Monitor job health without leaving your agent.

Healthchecks.io monitors scheduled background jobs and cron tasks, letting you track service uptime directly through your AI agent. Quickly list active checks, inspect ping histories, and pause or resume monitoring to debug failed infrastructure runs without leaving your chat window.

A+ Quality Score 100/100

cron-jobs | uptime-monitoring | background-tasks | alerting | infrastructure-monitoring

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/healthchecksio

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.

Healthchecks.io MCP

13 tools available
Cloud-hosted on Vinkius

When a critical back-end process fails at 3 AM, you don't want to jump between dashboards. This MCP connects Healthchecks.io directly to your workflow, giving your agent the visibility it needs to manage system health through plain conversation. You can ask your AI client to list all monitoring checks or inspect specific ping bodies to figure out exactly why a job failed. Need to temporarily stop alerts while you deploy? Your agent handles that too. This connection works with any MCP-compatible client and centralizes complex infrastructure auditing, making it one of the most powerful developer tools available in the Vinkius catalog.

Core Capabilities

01 — Manage Monitoring Checks

Create, read, update, or delete checks to define what services need monitoring.

02 — Review Ping History

Retrieve a list of recent pings and inspect the full payloads to debug successful or failed tasks.

03 — Track Status Changes

View historical status changes (flips) for any monitored service over time.

04 — Control Check State

Pause or resume an entire check to manage alerts during maintenance windows.

05 — Inspect Notifications

List all configured notification channels to confirm that team alerts are going to the right places.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to this MCP and provide your Healthchecks.io API Key.

02 Your AI agent connects using the key, granting it access to your monitoring accounts.

03 You interact with the system by asking your agent to perform actions like listing checks or retrieving ping bodies.

The bottom line is: you treat complex infrastructure auditing exactly like a chat command.

Built For

The DevOps Engineer who gets paged at 2 AM because a cron job failed. The SRE team that needs to audit monitoring coverage across dozens of services. Developers debugging background workers outside of local machine environments.

Site Reliability Engineer

Using this MCP, they automatically audit which services are monitored and check status flips without logging into the main dashboard.

DevOps Engineer

They ask their agent to list all checks for a service group or pause monitoring temporarily during a high-risk deployment window.

Software Developer

When debugging, they request the ping body for a failed task to understand exactly what data caused the back-end worker to crash.

What Changes When You Connect

01 Stop context switching: Instead of opening a browser, navigating to the dashboard, and filtering logs, you just tell your agent to list pings for a check. The data appears instantly in your chat history.

02 Pinpoint failure causes: If a job fails, you don't just get an 'error.' You ask your agent to retrieve the ping body, which gives you the exact payload that triggered the crash, saving minutes of painful debugging.

03 Control alerts on demand: Need to deploy? Use the `pause_check` tool through your agent. When the deployment is done, tell it to `resume_check`. It's simple, conversational control over critical infrastructure monitoring.

04 Audit coverage easily: You can use `list_checks` to get a full inventory of every scheduled task being monitored in a project, helping you audit compliance or spot blind spots quickly.

05 Visualize history: Beyond just knowing if it passed or failed, your agent lets you list flips, showing the precise timeline and number of times a service transitioned from 'up' to 'down', giving deep operational insights.

Real-World Applications

Debugging a Payment Gateway Sync Failure

A user asks their agent to check the payment sync job. The agent lists pings, finds the most recent failure, and then uses `get_ping_body` to confirm that the payload was missing the required transaction ID, solving the issue instantly.

Pre-Release Maintenance Window

An SRE team member tells their agent, 'We're deploying version 3.0; pause all checks related to user signups.' The agent uses `pause_check` for the specified checks, ensuring no false alerts interrupt the critical deployment process.

Inventory Check of Background Tasks

A new developer asks their agent, 'Show me every check related to user data cleanup.' The agent calls `list_checks` and filters the results by tags, giving them a complete inventory without manually browsing multiple dashboards.

Verifying Alert Channels

A team lead wants to confirm that critical alerts are going to Slack. They ask their agent to list integrations, confirming which notification channels are active and configured correctly before the next major release.

Patterns to Avoid

Manually checking job status in a UI

✗ AVOID

Opening the monitoring dashboard, clicking through multiple service groups, filtering by date range, and manually cross-referencing timestamps to determine if a task failed.

✓ INSTEAD

Instead, use your agent to `list_checks` for the relevant group, then ask it to retrieve specific pings or list flips. This is faster, searchable, and happens entirely within your conversation.

Forgetting to pause checks during deployment

✗ AVOID

Deploying a new version while monitoring active. The moment the background job fails due to code changes, an alert fires, causing unnecessary page-out calls.

✓ INSTEAD

Always ask your agent to use `pause_check` before starting major deployments and remember to call `resume_check` immediately after.

Not knowing what caused the failure

✗ AVOID

Seeing a 'failed' status on a job but having no idea if it was a credential issue, bad data format, or system timeout.

✓ INSTEAD

Don't just look at the status. Use your agent to `list_pings` and then specify `get_ping_body`. The body reveals the actual input data that failed.

The Right Fit

Use this MCP if your primary workflow involves auditing, managing, or debugging scheduled services (cron jobs) and background tasks. You need to know *why* a service failed, not just *that* it failed. If you regularly have to check 'Is the daily report job running?' or 'Why did the data sync fail last night?', this is for you.

Don't use this if your goal is simply logging simple API calls or storing ephemeral data; those are better handled by basic logging tools. Also, don't use it if you only need to trigger a single action once (like sending an email); then just use a messaging MCP. This tool is for comprehensive, historical service health management.

Debugging Background Jobs Used To Be a Dashboard Nightmare

Think about it: A critical background job fails at 3 AM. Your usual process involves logging into the monitoring dashboard, finding the correct project, locating the specific service group, and then sifting through pages of raw logs until you find the failure marker. You're clicking tabs, copying UUIDs, opening multiple views, and trying to piece together a timeline from scattered data points.

With this MCP, that whole process vanishes. You simply tell your agent, 'Show me why the payment sync failed.' It automatically finds the last pings, identifies the error status, and even pulls the exact ping body for you—all in one conversational response.

Healthchecks.io MCP Gives You Direct Job Visibility

You eliminate manual checks like listing pings or viewing status changes through separate UIs. Instead of clicking 'View History' and then filtering by date, you ask your agent for `list_flips`, and the data is immediately structured and ready to read.

This means you stop debugging in tabs and start solving problems with natural language. You get a single source of truth about job health that lives right where you are working.

Healthchecks.io: 13 Tools for Job Monitoring

These tools let your agent interact with every function of Healthchecks.io, allowing you to manage checks, view history, and debug infrastructure status through simple commands.

#	TOOL	DESCRIPTION
01	`create_check`	Creates a brand new monitoring check for an infrastructure component.
02	`delete_check`	Removes an existing monitoring check from the system.
03	`get_check`	Retrieves all details about a specific check using its unique ID or key.
04	`get_ping_body`	Fetches the raw data payload that was included with a specific ping.
05	`get_status`	Checks the overall health status of the entire Healthchecks.io monitoring service.
06	`list_badges`	Lists all predefined status badges available for displaying project health.
07	`list_checks`	Retrieves a list of all active monitoring checks, optionally filtered by tags or identifiers.
08	`list_flips`	Lists the historical record of status changes (up/down) for a specific check.
09	`list_integrations`	Shows all configured notification channels, such as Slack or email alerts.
10	`list_pings`	Provides a list of the most recent ping events associated with a given check.
11	`pause_check`	Temporarily suspends monitoring for a specified check to prevent false alerts.
12	`resume_check`	Restores active monitoring status for a previously paused check.
13	`update_check`	Modifies the settings or parameters of an already existing check.

See It in Action

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

U List all my active checks and their current status.

I've retrieved your checks. You have 3 active monitors: 'Daily Backup' (Up), 'Hourly Sync' (Up), and 'Cleanup Script' (Down). Would you like to see the pings for the 'Cleanup Script'?

U Show me the last 5 pings for check uuid '550e8400-e29b-41d4-a716-446655440000'.

Here are the last 5 pings for that check. The most recent was 10 minutes ago with a 'success' status. Ping #452 included a payload of 1.2KB. Do you want to inspect the body of any specific ping?

U Create a new check named 'Database Backup' with a 24-hour timeout.

I've created the 'Database Backup' check. The UUID is 'a1b2c3d4...'. You can now send pings to `https://hc-ping.com/a1b2c3d4...`. I've set the timeout to 86400 seconds as requested.

Frequently Asked Questions

01 **How do I list all my active checks using the Healthchecks.io MCP?**

You use the `list_checks` tool through your agent. This command pulls a comprehensive inventory of every monitoring check in your project, letting you see what's covered and what isn't.

02 **Can I pause alerts using Healthchecks.io MCP?**

Yes. You use the `pause_check` tool to temporarily suspend monitoring for any specific check. This is perfect for maintenance windows or controlled deployments when you don't want false alarms.

03 **What is the best way to debug a failed task with Healthchecks.io MCP?**

The most effective way is to first use `list_pings` and then request the raw data payload using `get_ping_body`. This shows you the exact input that caused the failure.

04 **Does Healthchecks.io MCP handle status history?**

It does. You can use the `list_flips` tool to track every single transition—from 'up' to 'down' and back again—giving you a full audit trail of service stability.

05 **How do I confirm that my team gets alerts for critical jobs?**

You use the `list_integrations` tool. This shows all configured notification channels, allowing you to verify which services are correctly alerting your team.

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 `"healthchecksio": { "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

Healthchecks.io 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 Healthchecks.io. 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	Healthchecks.io MCP
Server ID	`019e38a7-21fa-704d-9e08-d11daab65509`
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/healthchecksio.