MCP SERVER | NO CODE | CLOUD HOSTED

Cronitor MCP for AI Agents

Manage Cron Jobs and Website Uptime Monitoring

Cronitor brings full visibility into your infrastructure's background jobs, heartbeats, and website uptime. Connect this MCP to instantly track performance metrics, manage alerts, or diagnose failures across any service without switching tabs. Your AI agent acts like a constant SRE assistant right in your chat window.

A+ Quality Score 98.33/100

cron-jobs | uptime-monitoring | heartbeats | incident-management | telemetry

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/cronitor-cron-monitoring

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.

Cronitor (Cron Monitoring) MCP

44 tools available
Cloud-hosted on Vinkius

You shouldn't have to jump between ten different dashboards just to see if your nightly backups ran, or why your primary API endpoint is suddenly flaky. This MCP lets you connect Cronitor directly to your preferred AI client, giving you full visibility into your entire infrastructure's health through natural conversation. You can ask your agent to check the status of a critical cron job, query performance metrics for specific sites, or even schedule maintenance windows when planned downtime hits.

This capability is huge. Instead of manually running checks, your AI agent handles the monitoring lifecycle: it monitors background jobs, tracks heartbeats, and analyzes site analytics—all from one place. Because this MCP is hosted on Vinkius, you connect once to access Cronitor alongside thousands of other developer tools, making it a single source for operational intelligence.

Core Capabilities

01 — Monitor Infrastructure Status

Set up, list, and update monitors for cron jobs, heartbeats, and website health checks across multiple services.

02 — Gather Performance Metrics

Query aggregated data and site analytics to understand latency, success rates, and overall system performance trends.

03 — Manage Incidents and Alerts

Track active issues, create maintenance windows, or manage status pages directly through your AI agent.

04 — Send Live Telemetry Pings

Send job state updates (like 'running', 'complete', or 'failed') to track execution lifecycles in real time.

05 — Organize Monitoring Assets

Group monitors, manage notification lists, and control which environments are being monitored.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/cronitor-cron-monitoring** — connect your AI agent in three steps.

01 Subscribe to the Cronitor MCP and provide your unique API key.

02 Connect this MCP to your AI client (like Claude or Cursor).

03 Use natural language prompts to ask about monitor statuses, query performance data, or trigger incident actions.

The bottom line is you talk to your agent like a teammate and get immediate operational status reports for your entire stack.

Built For

Anyone whose job involves keeping systems running 24/7 needs this. If you're tired of jumping between monitoring dashboards, these tools are for you. It’s especially critical for Ops Leads and Backend Developers who need to prove uptime or troubleshoot a flaky cron job before the customer even calls.

Site Reliability Engineer (SRE)

Managing status pages, grouping monitors by service line, and coordinating incident response actions like setting maintenance windows.

DevOps Engineer

Creating new monitors for specific cron jobs or heartbeats, reviewing aggregate performance metrics, and updating group settings.

Backend Developer

Sending direct telemetry pings from the code editor to confirm a job's execution state after committing changes.

What Changes When You Connect

01 Stop manually checking job statuses. Use the `list_monitors` tool to instantly check if a critical background process is healthy or failing.

02 Get immediate incident visibility using `list_issues`. Instead of searching through logs, your agent pulls up a live list of all active failures and incidents.

03 Control planned downtime without panic. Schedule maintenance windows with `create_maintenance_window` so alerts automatically silence during deployments.

04 Improve performance diagnostics by calling `get_metrics` to pull aggregated data on latency across different services in one prompt.

05 Maintain clean infrastructure by using `delete_monitor` or `delete_group` when a service is deprecated, keeping your monitoring setup accurate.

Real-World Applications

Investigating a sudden API failure

A backend developer notices an endpoint failing. They prompt their agent to check the status by calling `get_monitor` and then use `send_telemetry` to ping the service immediately, getting confirmation of the execution cycle right in the chat.

Preparing for planned deployment downtime

The SRE lead needs to deploy a major system update. They ask their agent to first use `create_maintenance_window` and then pause all related services using `pause_group`, ensuring no alerts fire during the rollout.

Auditing service performance trends

An operations lead needs proof of system stability. They ask their agent to use `get_aggregates` and `list_sites` to pull up historical data on site latency, providing a single dashboard view for stakeholders.

Scaling monitoring setup

A DevOps team needs to monitor five similar services. They prompt their agent to use the `clone_monitor` tool and then run `update_monitors` in bulk, setting up consistency across all new deployments.

Patterns to Avoid

Ignoring group management

✗ AVOID

A user creates 50 individual monitors for a single service line. When something breaks, they have to ask the agent about each one separately, which is slow and inefficient.

✓ INSTEAD

Group related services first by using `create_group`. Then, if anything fails, you can query or manage the status of all 50 through that single group name.

Manually managing alerts

✗ AVOID

A team member forgets to tell their agent they are deploying. The system sends dozens of 'Failure' alerts for every service, creating alert fatigue.

✓ INSTEAD

Before deployment, use `create_maintenance_window` and then leverage the group controls to pause all relevant monitors, silencing noise until the window closes.

Using outdated site data

✗ AVOID

A user tries to troubleshoot slow page load times but only has access to historical metrics that don't reflect current code changes.

✓ INSTEAD

Use `list_sites` to ensure your RUM sites are active, and then use the specific API calls like `create_site` or `update_site` to keep the monitoring data accurate for today's performance.

The Right Fit

You should use this MCP if you need a centralized view of system health, especially when coordinating multiple background jobs and user-facing services. It's perfect for teams that rely on constant uptime validation—think SREs or DevOps groups. You absolutely need it if your team uses cron jobs or has publicly visible service status pages.

Don't use this if you only need to monitor one isolated, non-critical endpoint; a simpler monitoring tool might suffice. Also, don't rely on this for deep code debugging (that's what an IDE is for); use it instead when the problem is 'Did the job run?' or 'Is the site slow right now?'. If you need to build a full CI/CD pipeline and want type-safe validation of inputs before sending commands, look into dedicated workflow tools.

Cronitor (Monitoring MCP) for Uptime Monitoring

Right now, checking if your critical background jobs finished requires logging into the cron dashboard. Then you check the heartbeat service in a second tab, and finally, you manually verify the site analytics on a third page. It's copy-pasting statuses across three different screens just to answer: 'Is everything running?'

With this MCP, your agent pulls all that information into one chat window. You ask it about job status or uptime, and it brings back comprehensive reports instantly, giving you the full picture without leaving your primary workflow.

Cronitor (Monitoring MCP) for Incident Management

When an issue hits, the typical process is a frantic search: which service failed? Is it active or already resolved? Did we schedule maintenance for this? You spend time sifting through tickets and status pages just to establish facts.

Now, your agent tracks everything. You can ask it to list all open incidents via `list_issues`, see if a maintenance window is scheduled, and even create a new incident report—all in one conversation thread.

Cronitor (Monitoring) MCP: 24 Tools for Uptime Monitoring

Use these tools to manage monitors, retrieve metrics, update status pages, and control the entire lifecycle of your infrastructure monitoring setup.

#	TOOL	DESCRIPTION
01	bulk_issues	Perform bulk actions on multiple reported issues at once.
02	clone_monitor	Create a new monitor using the settings from an existing one.
03	create_api_key	Generate and issue a new API key for system access.
04	create_status_page_component	Build individual sections or components that go onto your main status page.
05	create_environment	Define a new monitoring environment (e.g., staging, production).
06	create_group	Assemble related monitors into a single logical group.
07	create_issue	Log and report a new active incident or issue for tracking.
08	create_maintenance_window	Schedule an official window of planned downtime to prevent alerts from firing unnecessarily.
09	create_monitor	Set up a new monitor for a specific cron job, heartbeat, or website endpoint.
10	create_notification	Establish a list of recipients who need to be alerted when an issue occurs.
11	create_site	Set up a new Real User Monitoring (RUM) site for performance tracking.
12	create_status_page	Initialize and create an entirely new public status page.
13	delete_api_key	Revoke and delete a specific API key when it's no longer needed.
14	delete_status_page_component	Remove status page components that are outdated or incorrect.
15	delete_group	Delete an entire monitor group from the system.
16	delete_monitor	Take down and delete a specific, individual monitor.
17	delete_site	Remove a Real User Monitoring site when monitoring is complete.

#	TOOL	DESCRIPTION
18	delete_status_page	Archive and delete an entire status page instance.
19	get_aggregates	Retrieve rolled-up aggregates
20	get_metrics	Requires at least one field parameter. Retrieve performance metrics
21	get_monitor	Retrieve a specific monitor
22	get_site_errors	Get site errors
23	list_api_keys	View all API keys currently associated with the account.
24	list_status_page_components	Retrieve a list of existing components on status pages.
25	list_environments	View all defined monitoring environments (e.g., dev, prod).
26	list_groups	See a list of organized monitor groups.
27	list_issues	Retrieve a history of all open and closed incidents or issues.
28	list_monitors	Get an overview, or search for specific existing monitors.
29	list_notifications	List notification lists
30	list_sites	List RUM sites
31	list_status_pages	List status pages
32	pause_group	Bulk pause or resume a group of monitors
33	pause_monitor	Use 0 to resume. Pause or resume a monitor
34	query_site_analytics	Query site analytics
35	send_telemetry	link. Send telemetry events (pings, metrics)
36	update_api_key	Update an API key
37	update_status_page_component	Update status page components
38	update_environment	Update an environment
39	update_group	Update a monitor group
40	update_issue	Update an issue state
41	update_monitors	Create or update monitors in bulk

#	TOOL	DESCRIPTION
42	`update_notification`	Update a notification list
43	`update_site`	Update RUM sites
44	`update_status_page`	Update status pages

See It in Action

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

U What's the status of my nightly database backup job? And also, what were the site error rates yesterday?

System Health Check:

• **Database Backup (cron):** Status: **FAILED**. Last run: 2026-06-12. Error Code: `DB_TIMEOUT`. Action needed.
• **Website Errors (RUM):** Yesterday's aggregate site errors were 45/10,000 page views. This is a 3% increase over the weekly average.

U Can you pause all monitoring for our primary API services while we deploy new code?

Action Confirmed:

The 'Primary Services Group' has been successfully paused for the next four hours (until 14:00 UTC). All associated monitors are now suspended, and no alerts will be generated. You can resume monitoring by asking me to `resume group: Primary Services Group`.

U List all active monitors and tell me which ones are healthy but haven't reported in 12 hours.

I checked the list of 34 monitors. Here's what I found:

✅ ***Healthy/Stale:*** `legacy_heartbeat` *(Last ping: 15+ hrs).* Recommendation: Check if this service still exists.*
• ⚠️ **Needs Review:** `third_party_api` (Status: OK, but last metric pull was 12 hours ago). Monitor is fine, but data is stale.

Frequently Asked Questions

01 **How do I use the Cronitor MCP to check if my cron jobs are actually running?**

You simply ask your agent for a status report. It checks all defined monitors and tells you which job is healthy, failing, or paused. This saves you from manually checking multiple dashboards just to confirm execution.

02 **Can the Cronitor MCP help me manage my website downtime information?**

Yes. You can create a dedicated status page and use the MCP to post updates about ongoing incidents or scheduled maintenance windows, keeping your users informed automatically.

03 **What if I need to track performance metrics for different times of day using Cronitor?**

The MCP lets you query aggregated data over specific time ranges. You can pull P50 duration or success rates for the last 24 hours, giving you historical context on system load.

04 **Is Cronitor good for tracking API endpoints that fail intermittently?**

Absolutely. By setting up a dedicated monitor and using telemetry pings, your agent tracks the entire execution lifecycle—from running to failure—giving you precise data on intermittent issues.

05 **Does Cronitor help me organize my monitoring setup if I have many services?**

Yes. You can use monitor groups to logically cluster related jobs (like 'Payment Services' or 'User Auth'), letting your agent manage them all with one command.

06 **What is the difference between using Cronitor for job monitoring versus website uptime?**

Job monitoring tracks background processes that run on a schedule. Website uptime monitors track external-facing endpoints, ensuring the actual pages your users visit are loading correctly and quickly.

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 `"cronitor-cron-monitoring": { "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

Cronitor (Cron Monitoring) 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 Cronitor (Cron Monitoring). 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	Cronitor (Cron Monitoring) MCP
Server ID	`019e3880-dca5-7283-a160-3f30154add30`
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/cronitor-cron-monitoring.