MCP SERVER | NO CODE | CLOUD HOSTED

Tingyun / 听云 MCP

Query App Performance and Metrics Via Conversation

Tingyun / 听云 connects your AI client to a complete Application Performance Monitoring (APM) system. This MCP lets you talk to your entire performance stack—applications, alerts, metrics, and dependencies—using natural language. Instead of logging into multiple dashboards, your agent instantly lists monitored apps, checks real-time health summaries, and retrieves specific metric data points just by asking a question. It turns complex site reliability work into a simple conversation.

A+ Quality Score 100/100

apm | performance-metrics | incident-response | application-monitoring | real-time-alerts | digital-experience

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

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.

Tingyun / 听云 MCP

10 tools available
Cloud-hosted on Vinkius

Tingyun / 听云 gives your AI client control over your whole digital performance stack. You don't need to navigate the monitoring console; you just tell your agent what you want to know, and it handles the rest. Need to know why latency spiked on checkout? Ask your agent. It can instantly list monitored applications for you. Want to check if a database connection is slowing things down? Just ask for dependencies. The tool lets you browse active alerts immediately or query specific metric data points to find anomalies. Everything from external service calls to frontend user experience metrics (RUM) gets organized into simple, conversational answers. By connecting this MCP via Vinkius, your agent acts like a real-time Site Reliability Engineer on demand, keeping your system performance accurate and responsive without you ever leaving your chat window.

Core Capabilities

01 — List monitored applications

Retrieves the names and general health status of all applications being tracked by Tingyun.

02 — Get application health summaries

Pulls a quick performance report, including average response time and error rates, for a specific service.

03 — Check active alerts

Lists all current performance warnings or critical incidents that require immediate attention.

04 — Query system metrics

Fetches precise, historical data points for any measured metric you specify (e.g., CPU usage, latency).

05 — Audit service dependencies

Lists all connected databases and external services that an application relies on.

06 — Review user experience data

Browses Real User Monitoring (RUM) applications to audit how the frontend performs for actual end-users.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to this MCP and provide your Tingyun API Key and Secret Key.

02 Connect your preferred AI client (like Claude or Cursor) to Vinkius, giving it permission to access the data.

03 Ask a natural language question, such as 'What is the performance summary for the user service?' Your agent translates that request into the necessary tool calls and returns the live data.

The bottom line is you get instant visibility into complex system health using nothing but plain conversation.

Built For

This MCP is for SREs, DevOps Engineers, and technical analysts who are tired of switching between dashboards, running multiple scripts, and manually cross-referencing data points at 3 AM. If your job involves figuring out *why* something broke in a distributed system, this is what you need.

Site Reliability Engineer (SRE)

Automates incident response by checking active alerts and querying detailed metric data points to pinpoint the failure source immediately.

DevOps Engineer

Conducts routine system audits, listing application instances and external service calls to ensure architectural dependencies are healthy.

Technical Analyst

Performs bottleneck analysis by retrieving specific metric data for comparison and reviewing historical performance trends.

What Changes When You Connect

01 Eliminate dashboard hopping. Instead of opening five tabs to check the 'Checkout Service' summary, you ask your agent directly for the `get_app_summary` and get a single, unified answer.

02 Stop guessing where performance issues come from. You can use `list_external_services` to see exactly which third-party API call is causing latency spikes, cutting down debugging time dramatically.

03 Stay ahead of downtime with instant awareness. The agent checks for critical alerts using `list_alerts`, giving you immediate notification on 'High Latency' incidents before users complain.

04 Understand the full scope of an app. You don't just get a summary; you can run `list_databases` and `list_applications` to map out every dependency, helping pinpoint system bottlenecks.

05 Focus on user reality. By using `list_browser_apps`, your agent doesn't just tell you the API is slow; it shows you how the actual customer sees it across different browsers.

Real-World Applications

Investigating a sudden spike in checkout errors.

The agent first uses `list_applications` to confirm 'Checkout Service' is monitored. Next, it calls `get_app_summary`. The summary shows the error rate jumped 10x yesterday. You then ask for dependencies, and the tool reveals that one external service call added latency, identifying the root cause instantly.

Auditing compliance for system health.

An engineering manager needs to know if all core services are monitored properly. The agent uses `list_applications` and then calls `list_alert_policies`. This provides a comprehensive, auditable list of what is covered by monitoring rules.

Diagnosing slow frontend performance for international users.

The team suspects the mobile experience is poor. Instead of manually checking browser console logs, the agent uses `list_browser_apps` to pull Real User Monitoring data and compare global performance metrics.

Mapping a complex system's dependencies.

A new developer needs to understand how 'User API' works. They ask the agent, which uses `list_databases` and `list_external_services`, to generate a map of all critical connections, preventing accidental breakage.

Patterns to Avoid

Checking metrics manually

✗ AVOID

Opening the Tingyun console, clicking on 'User API', navigating to 'Latency Metrics', then finding and copying the average response time into a spreadsheet.

✓ INSTEAD

Just ask your agent: 'What was the average response time for User API over the last hour?' The MCP uses `get_metrics` to get the number instantly without any manual clicks or copy-pasting.

Forgetting system boundaries

✗ AVOID

Assuming that because a service is up, it means all its dependencies are also healthy. You check only the main app dashboard and miss an underlying database issue.

✓ INSTEAD

Always ask your agent to run `list_databases` and cross-reference any reported issues against those dependency lists to ensure nothing is being missed.

Querying without context

✗ AVOID

Asking 'What's wrong?' into the monitoring tool. It returns hundreds of alerts, forcing you to manually figure out which ones matter.

✓ INSTEAD

Be specific: 'Show me any Critical alerts for Payment Service related to high latency.' This uses `list_alerts` and filters the noise down to actionable items.

The Right Fit

Use this MCP if your primary pain point is translating complex, multi-layered monitoring data into simple answers. You need an agent that can act as a conversational layer over dozens of dashboards—for example, needing to correlate `list_external_services` failures with current `get_app_summary` metrics and active alerts from `list_alerts`. Don't use this if you just need to set up basic monitoring; for configuration tasks, use the native Tingyun console. If your goal is purely data visualization without querying—for example, building a specific custom graph in another tool—you may only need to extract raw metrics using `get_metrics` and handle the graphing elsewhere.

The Pain of Monitoring: Juggling Dashboards All Day

Right now, figuring out why an app is slow means logging into a dozen different dashboards. You check the main performance overview, see a red warning flag, then have to manually click through dependency trees to find out if it’s the database or an external API call. Then you copy that failure code and paste it into your incident ticket.

With this MCP, you simply ask your agent: 'Why is the Checkout Service slow?' The system automatically runs checks on application summaries, external service calls, and database dependencies, compiling a full diagnosis back to you in seconds. You get the answer, not just links to more dashboards.

Tingyun / 听云: Instant Deep Dives with Conversation

The ability to map out dependencies and monitor user experience was always siloed. You'd run `list_applications` for the backend, then open a separate view for RUM data using `list_browser_apps`. If you wanted to correlate those two findings, it took hours of cross-referencing.

Now, your agent correlates all these views instantly. It tells you exactly what's wrong with the frontend user experience *because* an external service call failed, providing one unified diagnosis every time.

Tingyun / 听云: 10 Tools for Observability Ops

These tools give your agent the ability to query every aspect of your performance stack—from listing applications to checking live metrics and alerts.

#	TOOL	DESCRIPTION
01	`get_account_info`	Retrieves general metadata about the Tingyun account setup.
02	`get_app_summary`	Provides a high-level performance summary for a specific application.
03	`get_metrics`	Allows querying precise, customizable metric data based on timeframes and dimensions.
04	`list_alerts`	Shows all currently active performance alerts and their associated policies.
05	`list_app_instances`	Lists every running instance of an application to identify geographic or deployment issues.
06	`list_applications`	Retrieves a full list of all Application Performance Monitoring (APM) applications being monitored.
07	`list_browser_apps`	Lists the Real User Monitoring (RUM) applications to audit frontend performance data.
08	`list_databases`	Retrieves a list of all databases monitored for connectivity and query performance issues.
09	`list_external_services`	Lists every external service call an application makes, tracking latency and failure rates.
10	`list_alert_policies`	Shows the predefined rules used to trigger performance alerts within the system.

See It in Action

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

U List all applications monitored by Tingyun.

I've retrieved your monitored applications. You have 12 APM applications, including 'Checkout Service', 'User API', and 'Auth Gateway'. Which one would you like to check the summary for?

U Show me the performance summary for application ID 12345.

I've retrieved the summary for app 12345. Average response time is 150ms, error rate is 0.05%, and Apdex score is 0.95. Would you like to see the database dependency list?

U Check for any critical alerts in Tingyun from today.

I've checked for alerts. There is 1 Critical alert for 'High Latency' on the 'Payment Service' and 2 Warnings regarding disk usage. Should I retrieve the alert detail for the high latency incident?

Frequently Asked Questions

01 **How does Tingyun / 听云 MCP help with performance audits?**

It lets you run comprehensive audits by listing all monitored applications and services. You can use `list_applications` combined with checking application summaries to ensure every part of your stack is accounted for.

02 **Can I track frontend issues using Tingyun / 听云 MCP?**

Yes, you can check Real User Monitoring (RUM) data by listing browser apps. Using `list_browser_apps` lets your agent audit how the actual end-user views your application on different devices.

03 **What if I need to know about database changes?**

You can use the MCP's tools to list all monitored databases. This ensures you have a complete inventory and visibility into any potential connectivity or query slowdowns.

04 **Is Tingyun / 听云 MCP good for SRE teams?**

Absolutely. It’s built for incident response, allowing your agent to list active alerts and quickly retrieve detailed metric data points when an issue arises.

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 `"tingyun": { "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

Tingyun / 听云 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 Tingyun / 听云. 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	Tingyun / 听云 MCP
Server ID	`019d848e-b962-7162-8d5f-7d0e1c1c7c2b`
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/tingyun.