MCP SERVER | NO CODE | CLOUD HOSTED

# Grafana k6 Cloud MCP

## Analyze performance metrics under load, conversationally.

Grafana k6 Cloud MCP lets you manage your entire performance testing lifecycle through natural conversation with your AI agent. You can list existing load tests, start new runs, stop active sessions instantly, and retrieve detailed metrics like latency, throughput, and error rates. It also checks if your application meets specific Service Level Objectives (SLOs), all without navigating multiple dashboards.

A+ Quality Score 98.33/100

load-testing | performance-engineering | stress-testing | test-automation | cloud-infrastructure | metrics

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/grafana-k6-cloud-load-testing

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.

# Grafana k6 Cloud (Load Testing) MCP

10 tools available
Cloud-hosted on Vinkius

Managing performance testing used to mean clicking through a dozen dashboards just to get the numbers you needed. Now, your AI agent handles it. This MCP connects directly to your k6 Cloud account, letting you run complex load tests and then ask questions about the results—like, "Did the checkout flow fail when we hit 500 virtual users?" You can start a test, wait for it to finish, and immediately pull back aggregated metrics on response time or failure rates. It's all conversational.

It gives you full control over your testing process, from listing projects across different organizations to getting the raw data needed for audits. When working with Vinkius in your AI client, you get access to this specialized set of tools alongside thousands of others, meaning performance analysis is just one query away. You don't need to jump between ten different tabs; you just talk to your agent.

## Core Capabilities

**01 — List and manage load tests**

Retrieves a list of all available k6 Cloud test names, IDs, scripts, and their last run statuses.

**02 — Control test execution runs**

Starts new test sessions or immediately halts running tests to manage resources.

**03 — Analyze performance metrics**

Pulls aggregated data for completed runs, showing things like average response time, throughput, and total errors.

**04 — Verify Service Level compliance**

Checks detailed threshold results to confirm if your application meets defined SLOs after a test run.

**05 — Monitor ongoing sessions**

Polls running tests for their current status, letting you track transitions like QUEUED or RUNNING.

## One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/grafana-k6-cloud-load-testing** — connect your AI agent in three steps.

01 Subscribe to this MCP and provide your k6 Cloud API Token.

02 Connect the credentials to any MCP-compatible AI client (Claude, Cursor, etc.).

03 Ask your agent a direct question like, "Start a load test for Project X" or "What were the latency metrics?"

The bottom line is you use natural language commands instead of API calls and web forms to manage your entire performance testing workflow.

## Built For

This MCP is built for engineers who spend too much time clicking through dashboards just to find a single metric. It's for the SRE who needs to quickly verify if a recent deployment caused latency spikes, or the QA engineer auditing months of test run histories.

**Performance Engineer**

Triggers specific load tests and uses the agent to immediately analyze metrics and audit thresholds.

**Site Reliability Engineer (SRE)**

Monitors application health by checking performance regressions after a deployment, using tools like get_run_metrics.

**QA Automation Specialist**

Audits historical test run histories and generates reports on overall system reliability across multiple environments efficiently.

## What Changes When You Connect

01 You instantly get a full picture of your test environment by using list_organizations and list_projects. Instead of manually navigating complex folder structures, you ask the agent to show you all available projects across multiple organizations.

02 Stop wasting time clicking through status tabs. You can start a new test run with start_test_run and monitor its progress in real-time without leaving your chat window until it finishes running.

03 Need proof that your application meets SLOs? Don't guess. Using get_run_thresholds, you ask the agent for precise results to verify if performance requirements were met after a major release.

04 Analyzing failure patterns is fast. Instead of looking at raw logs, calling get_run_metrics gives you aggregated data on error rates and average response times—all in plain language format.

05 Control your testing costs by using stop_test_run to abort an active session instantly if the test goes wrong or hits a resource limit too soon.

## Real-World Applications

**Investigating Post-Deployment Regression**

An SRE notices high latency in production. They ask their agent to list_tests for the 'API Checkout' suite, start a new test run with a moderate load, and then use get_run_metrics to instantly compare the average response time against historical baselines.

**Auditing Compliance History**

A QA Automation specialist needs proof that the payment gateway passed compliance checks last quarter. They ask the agent to list_runs for the 'Payment Gateway' project and then use get_run_thresholds on specific run IDs to compile a comprehensive audit report.

**Simulating Peak Traffic Spikes**

A Performance Engineer wants to see how the system handles Black Friday traffic. They ask to start_test_run, targeting 1000 VUs over 3 minutes, and then monitor its status using get_run to ensure it completes without error.

**Managing Complex Environments**

A DevOps team is testing three different microservice versions. They use list_organizations first, then list_projects for each, allowing the agent to manage and track results across disparate environments simultaneously.

# Patterns to Avoid

### Over-relying on a single dashboard view

✗ AVOID

Trying to figure out if a test passed or failed by just looking at the main run summary page, which often only shows partial data.

✓ INSTEAD

To get definitive proof, first use list_runs to find the specific ID. Then, you must call get_run_metrics for general performance and follow up with get_run_thresholds to confirm SLO compliance.

### Manually comparing multiple test runs

✗ AVOID

Having to copy-paste data points (like P95 latency) from several different run dashboards into a spreadsheet for comparison.

✓ INSTEAD

Instead, list_runs identifies the candidates. Then, ask get_run_metrics sequentially across those IDs and have your agent compile a clean, comparable table in conversation.

### Forgetting to stop tests

✗ AVOID

Starting a high-volume load test run and forgetting to cancel it when the initial data is gathered, wasting API quota and compute resources.

✓ INSTEAD

Always use start_test_run only when needed. Once you have the required metrics, use stop_test_run immediately to manage your account's computational costs.

## The Right Fit

Use this MCP if your primary goal is performance engineering, load testing, and verifying Service Level Objectives (SLOs). If you need to know how many users a system can handle before breaking or what the average response time was during peak traffic, this is for you. Don't use it if your problem is network debugging—this tool won't tell you why a server went offline due to a firewall block. You would need a dedicated infrastructure monitoring MCP for that. Also, don't use it if you only want general application usage metrics; stick to get_run_metrics for load-specific data.

## Debugging performance is tedious when you rely on dashboards alone.

Today, checking system performance means jumping between the k6 Cloud dashboard, opening your spreadsheet program, copying metrics like 'Average Latency' and 'Error Rate,' and manually comparing those numbers across different run IDs. It’s a cycle of clicking, copying, and pasting that eats up hours.

With this MCP, you simply talk to your agent. You tell it what test you ran, and it pulls the aggregated performance metrics for you, presenting clean data points in text format right where you are working.

## Grafana k6 Cloud MCP Provides Actionable Load Test Insights

You eliminate the need to manually list all test runs and then run separate queries for metrics. Instead, your agent handles the entire sequence: it lists the tests via list_tests, finds the most recent successful one using list_runs, and then executes get_run_metrics.

The result is immediate clarity. You get performance data—not just raw numbers, but direct answers about whether or not your application passed its defined SLOs.

# Grafana k6 Cloud (Load Testing) with 10 Tools

Use these tools to manage the full lifecycle of load testing: list, start, stop runs, and retrieve detailed performance metrics for analysis.

| # | TOOL | DESCRIPTION |
|---|---|---|
| 01 | `list_tests` | Lists all load tests available in your k6 Cloud account, showing their status and ID. |
| 02 | `get_test` | Retrieves the complete script configuration and details for a specific k6 test. |
| 03 | `list_runs` | Lists all past and current test runs, showing their status (like QUEUED or FINISHED), duration, and VUs. |
| 04 | `get_run` | Gets full details about a specific k6 test run, including metadata. |
| 05 | `start_test_run` | Initiates a brand new load test run on the k6 Cloud infrastructure, returning a unique ID for tracking. |
| 06 | `stop_test_run` | Stops an active k6 Cloud test session immediately when you need to save resources or abort testing. |
| 07 | `list_organizations` | Provides a list of all organizations under your k6 account, along with member counts. |
| 08 | `list_projects` | Lists all testing projects within a specific organization. |
| 09 | `get_run_metrics` | Calculates and returns detailed performance metrics for any completed test run, like latency or data received. |
| 10 | `get_run_thresholds` | Retrieves the specific evaluation results to confirm if a finished test passed or failed against defined SLOs. |

# 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 load tests in k6 Cloud

I've found 4 load tests: 'API Stress Test' (ID: 123), 'Checkout Flow' (ID: 456), 'Inventory Spike' (ID: 789), and 'Frontend Smoke' (ID: 012). Which one would you like to check the last run status for?

**U** Show me the performance metrics for the last run of the 'API Stress Test'

Retrieving metrics for run ID 'run-98765'... Average response time: 142ms, P95: 285ms, Failed requests: 0.2%, Max VUs: 500. All metrics are within acceptable ranges. Would you like me to check the threshold details?

**U** Start a new test run for 'Checkout Flow' (ID: 456)

Trigging new run for 'Checkout Flow'... Done. New Run ID is 'run-13579'. It's currently in the QUEUED state. I'll let you know when the first VUs start injecting traffic.

# Frequently Asked Questions

01 **How do I check the status of a running test with Grafana k6 Cloud MCP?**

You use list_runs to see all active and completed test sessions. This allows you to monitor if the run is QUEUED, RUNNING, or FINISHED without guessing.

02 **Can I start a new load test run using Grafana k6 Cloud MCP?**

Yes, calling start_test_run immediately kicks off a new session. The agent returns the unique ID you need to track it and manage resources later.

03 **What kind of metrics does get_run_metrics provide for Grafana k6 Cloud?**

This tool provides crucial performance data, including average response time (latency), throughput, VUs used, and the count of failed requests.

04 **Does Grafana k6 Cloud MCP help me check SLOs?**

Yes, using get_run_thresholds lets you verify if your application met pre-set Service Level Objectives. It's a direct compliance check for any given test run.

05 **Do I need to know the k6 script code to use Grafana k6 Cloud MCP?**

No, you don't. You only interact with the agent using plain English commands; the MCP handles the underlying connection and execution of the scripts.

# 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 `"grafana-k6-cloud-load-testing": { "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

# Grafana k6 Cloud (Load Testing) 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 Grafana k6 Cloud (Load Testing). 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 | Grafana k6 Cloud (Load Testing) MCP |
| Server ID | `019d75be-f178-711d-bf25-88256f131864` |
| 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/grafana-k6-cloud-load-testing.