MCP SERVER | NO CODE | CLOUD HOSTED

Sumo Logic MCP

Diagnose system failures by querying logs directly.

Sumo Logic connects your AI client directly to enterprise log data. Run complex security searches, monitor data ingestion pipelines, and check account usage metrics—all from a single chat window. It lets you diagnose system issues by querying diagnostic logs or checking collector status without ever opening the web console.

A+ Quality Score 100/100

log-analysis | security-monitoring | incident-response | data-ingestion | system-observability

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

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.

Sumo Logic MCP

9 tools available
Cloud-hosted on Vinkius

When your systems throw an error, you can't afford to jump between dashboards just to find root causes. This MCP gives your AI agent direct access to massive streams of security and operational data. You tell your agent what to look for—like 'all timeouts in the last two hours'—and it handles the complex queries needed to track down those specific events.

Beyond searching logs, you can check how your system is collecting data by listing out all connected collectors or checking billing usage right from the command line. It also lets you manage who has access and what alerts are running via webhooks. Because this integration lives in Vinkius, your AI client gets instant access to all these deep operational tools, allowing you to automate log analysis organically without needing complex dashboard integrations.

Core Capabilities

01 — Run targeted log searches

Start a detailed search query on your logs and wait for the results to appear.

02 — Track live job status

Check if a complex or lengthy log search is still processing or if it finished successfully.

03 — Retrieve final incident details

Pull the actual list of logs and event records once a search job has completed.

04 — Audit data sources

List all configured data collectors to verify where your system is gathering telemetry.

05 — Review account access controls

See who the users are and what security roles they possess within the Sumo Logic environment.

06 — Check operational alerts

View which external systems are configured to receive automated alerts via webhooks.

One Click on Vinkius — From Prompt to Execution

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

01 First, enable the Sumo Logic MCP integration module in your Vinkius environment and authenticate using your `SUMO_ACCESS_ID` and `SUMO_ACCESS_KEY`.

02 Next, instruct your AI client naturally: 'Find all high-priority security errors spanning the last day.'

03 Your agent executes the search job, provides a Job ID for tracking, and then retrieves the final logs once the status confirms completion.

The bottom line is you treat log analysis like a conversation instead of navigating complex web interfaces.

Built For

This MCP is for platform engineers and security analysts who get frustrated having to switch between terminals, dashboards, and ticketing systems just to figure out what broke. If your job involves tracing an incident from a single log line back through user permissions and alert configurations, this tool saves you hours of clicking.

Site Reliability Engineer (SRE)

Uses the MCP to validate data ingest loads by listing collectors and checking billing usage without leaving their terminal.

Security Operations Analyst

Queries logs using `create_search_job` to trace unauthorized access attempts or suspicious activity across historical records.

DevOps Engineer

Runs deep log searches against production clusters to find root causes, then confirms necessary alert webhooks are active for remediation.

What Changes When You Connect

01 Instantly locate root causes. Instead of manually building complex queries in a web UI, you just ask your agent to find specific errors using `create_search_job` and get the answers immediately.

02 Eliminate dashboard hopping. You can check account usage metrics via `get_account_billing`, verify which users exist (`list_account_users`), and see active alerts—all without switching tabs or applications.

03 Verify data pipelines easily. Use `list_collectors` to get a full map of your telemetry sources, then drill down with `get_collector_details` if something looks wrong.

04 Manage security compliance quickly. You can check all configured alert webhooks using `list_active_webhooks`, ensuring critical systems like PagerDuty are still connected and firing alerts.

05 Streamline incident response. If an error occurs, your agent runs the search (`create_search_job`), waits for confirmation (`get_search_status`), and delivers the final log data (`get_search_results`)—all in one flow.

Real-World Applications

A service keeps failing intermittently, but the logs are too massive to sift through.

The SRE asks their agent to run a targeted search job on 'connection refused' errors over the last 48 hours using `create_search_job`. The agent tracks the status with `get_search_status` and returns all specific failure timestamps, allowing the engineer to narrow down the failing microservice IP address.

The security team suspects an unauthorized user account is active.

An analyst tells their agent to list all users via `list_account_users` and then immediately checks the roles using `list_account_roles`. This confirms if a service account has excessive permissions, speeding up compliance audits.

The billing department needs to confirm what data sources are contributing to high usage.

Instead of downloading complex reports, the agent uses `get_account_billing` to pull current usage metrics and then cross-references that with a list of active collectors found via `list_collectors`.

A new alert system needs integration, but nobody knows where the webhooks are configured.

The ops engineer prompts their agent to list all active webhooks using `list_active_webhooks`. This instantly provides a checklist of every external service currently receiving automated alerts.

Patterns to Avoid

Trying to manually combine searches

✗ AVOID

Opening the dashboard, running Search A; copying the results. Then opening a second tab and running Search B, and then trying to compare them in Excel.

✓ INSTEAD

Just ask your agent to run both queries sequentially: 'First, run search job X for auth failures, then check webhooks using `list_active_webhooks`.' The agent handles the complex orchestration.

Forgetting to wait for results

✗ AVOID

Telling your AI client to find all logs and immediately assuming it has the final data, leading to incomplete or partial responses.

✓ INSTEAD

Always tell your agent to track the job status first. Use `create_search_job`, then confirm completion with `get_search_status` before calling `get_search_results`.

Assuming access permissions

✗ AVOID

Trying to fix a network issue without knowing who has admin rights or what the current security roles are.

✓ INSTEAD

Always check the account structure first. Use `list_account_roles` and then `list_account_users` to understand the system's permission boundaries before making changes.

The Right Fit

Use this MCP if your primary need is deep, diagnostic querying of historical log data or auditing internal infrastructure components. If you are an SRE who needs to check collector status (`list_collectors`), confirm alert endpoints (`list_active_webhooks`), and run complex searches for failure patterns, this is the tool for you.

However, don't use this if your only goal is basic trend spotting or long-term capacity planning. For simple charts showing CPU utilization over six months, stick to Sumo Logic's native dashboarding tools; they are built specifically for that visualization. This MCP shines when you need the raw data and the ability to run sophisticated searches on demand via natural language prompts.

Finding root causes used to feel like a digital scavenger hunt.

Today, diagnosing an incident means opening half a dozen browser tabs. You jump from the dashboard to check billing limits, switch to the user management console to see who has access, and then finally run the actual log search on a separate screen. If you miss one step or copy-paste the wrong parameter, your troubleshooting time triples.

With this MCP, all that complexity disappears into a single conversation. You tell your agent what's broken—say, 'Why did the service fail at 2 pm?'—and it automatically runs the necessary checks: it finds the collector details, starts the search job, and delivers the full logs. The result is immediate context.

Get a complete view of your security posture with Sumo Logic's tools.

You no longer have to manually list out every connected system, check role permissions for compliance reports, and separately review all external alert settings. The agent consolidates this information by using `list_collectors`, running searches with `create_search_job`, and checking webhooks via `list_active_webhooks`—all in one flow.

This MCP doesn't just provide data; it gives you a cohesive operational picture, letting you confirm system health and user access simultaneously. That's the difference between debugging and truly understanding.

Sumo Logic with 9 Tools

These tools let you programmatically interact with your entire Sumo Logic environment, allowing your AI agent to manage searches, audit accounts, and monitor data flow.

#	TOOL	DESCRIPTION
01	`list_account_users`	Gets a list of every registered user account in the Sumo Logic system.
02	`list_active_webhooks`	Shows all external systems that are currently configured to receive alerts from your logs.
03	`create_search_job`	Starts a new, time-bound log search job and provides an ID for tracking its progress.
04	`get_account_billing`	Retrieves detailed metrics on your account's usage and billing data.
05	`get_collector_details`	Fetches specific setup details for a single configured data collector.
06	`get_search_results`	Pulls the actual log entries and incident reports after a search job has successfully finished.
07	`get_search_status`	Checks the current status of any active or pending search job using its ID.
08	`list_collectors`	Provides a comprehensive list of all data collection sources configured in your account.
09	`list_account_roles`	Lists the various security roles that define user permissions across the entire account.

See It in Action

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

U Fetch all account users along with active local integration webhooks sequentially properly.

Authenticating logic cleanly verifying parameters safely accurately...
Account Operatives Located (`list_account_users` **):**
- `Sarah Connor (s.connor@sumo.io)` - Admin
- `John Ops (j.ops@sumo.io)` - Observer

External Endpoints Mapped Systematically (`list_active_webhooks` **):**
- `PagerDuty Alerts` (Status: Active)
- `Slack #sec-ops channel` (Status: Active)

Successfully executed read tasks seamlessly inherently securely clearly effectively.

U Create a new search job tracking 'auth_failure' errors over the last 24 hours.

Initiating search constraints flawlessly...
Executed `create_search_job` cleanly, allocating Job ID `881A-9XF` securely.
Actively parsing `get_search_status` locally:
- Check 1: Processing dynamically.
- Check 2: Resolution Finalized accurately.

Status natively secured actively.

U Retrieve the exact search results from the active job ID once the asynchronous monitoring reports completion.

Retrieving output systematically and faithfully...
Log Occurrences Captured via `get_search_results` **natively**:
- Incident 08:00 AM: IP 10.4.5.1 failed authentication structurally.
- Incident 11:30 AM: IP 192.168.1.10 unauthorized access attempt.

Search results retrieved seamlessly.

Frequently Asked Questions

01 **How do I use Sumo Logic to find billing metrics with the MCP?**

You ask your agent directly for usage data, and it uses `get_account_billing` to pull your current consumption metrics. This avoids having to navigate the dedicated billing section of the console.

02 **Can I use Sumo Logic to check if a specific user exists?**

Yes, you ask for all users, and the agent uses `list_account_users` to provide a list. This lets you audit who has access without manual searching.

03 **How do I run a search job and ensure I get the results from Sumo Logic?**

You first use `create_search_job`. Then, tell your agent to check the status using `get_search_status` until it's complete. Finally, you call `get_search_results`.

04 **Does Sumo Logic help me monitor data sources?**

Yes. You can list all configured collectors with `list_collectors`, and if needed, get granular setup details for one source using `get_collector_details`.

05 **What is the best way to check alert endpoints in Sumo Logic?**

The agent can list every configured webhook endpoint for you using `list_active_webhooks`, giving you a quick audit of all external integrations.

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 `"sumo-logic": { "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

Sumo Logic 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 Sumo Logic. 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	Sumo Logic MCP
Server ID	`019d760e-a8f9-7280-aac7-5092032dc45c`
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/sumo-logic.