MCP SERVER | NO CODE | CLOUD HOSTED

Google Cloud Logging Stream MCP

Find the root cause of any production error.

Google Cloud Logging Stream gives your AI agent secure, scoped access to query logs using Google Cloud Logging. It's built for observability: analyze app errors, track traffic spikes, and monitor infrastructure health without ever needing global GCP permissions. You use this MCP when you need deep, filtered insights into structured cloud log data.

A+ Quality Score 100/100

log-aggregation | observability | troubleshooting | cloud-monitoring | data-analysis

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/google-cloud-logging-stream

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.

Google Cloud Logging Stream MCP

1 tools available
Cloud-hosted on Vinkius

Running a modern application means generating mountains of log data. The problem is that accessing those logs often requires granting massive, dangerous global credentials. This MCP changes that by giving your agent one specific superpower: the ability to run scoped queries directly on Google Cloud Logging.

It lets your AI client safely troubleshoot complex problems. For instance, instead of manually checking dozens of dashboards for an error, you can ask it to find all logs matching a certain severity or filter for transactions related to a specific user ID in the JSON payload. It reads and searches log entries from the configured stream, letting you focus solely on what matters.

This kind of focused access is critical. Through Vinkius, your agent connects once and gets this logging capability alongside hundreds of others. You can analyze operational data—like figuring out why a payment webhook failed or pinpointing when a service hit its usage limit—all without ever exposing the entire cloud environment to the AI.

Core Capabilities

01 — Query by specific filters

You tell it exactly what to look for, limiting the log search to defined criteria like severity or resource type.

02 — Parse structured data payloads

The agent extracts meaningful pieces of information from JSON embedded within individual log entries.

03 — Search historical logs by user ID

You can pinpoint every action taken by a specific user across potentially millions of log records.

04 — Monitor real-time error patterns

It streams the newest entries, allowing you to monitor for repeating warnings or critical failures as they happen.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/google-cloud-logging-stream** — connect your AI agent in three steps.

01 You instruct your agent what specific logs you need—for example, 'all errors from the payment service last hour'.

02 The MCP sends a precisely scoped query to Google Cloud Logging using advanced filtering syntax.

03 Your agent receives clean log entries and structured data that it can then analyze or summarize for you.

The bottom line is: instead of reading raw, overwhelming streams of text, your agent delivers an analysis focused only on the events you care about.

Built For

This MCP is for the Site Reliability Engineer (SRE) who has to find a single root cause across dozens of microservices at 2 AM, or the Backend Developer who needs to debug an obscure transaction failure in production. If your job involves finding 'needle in haystack' errors, you need this.

Site Reliability Engineer (SRE)

They use it to find transient infrastructure issues or performance bottlenecks by querying log entries for specific time windows and resource IDs.

Backend Developer

They test a new feature's failure path by filtering logs to isolate the exact API call that failed, checking both request and payload data.

DevOps Engineer

They monitor system health across multiple services by running batch queries on error severity or specific service names.

What Changes When You Connect

01 Pinpoint failures instantly. Use stream_logs to filter only for critical severity levels, ignoring routine 'info' logs and immediately focusing on actual errors.

02 Understand user flow failure points. By searching JSON payloads with stream_logs, you can track a specific user ID across multiple service calls until the point of failure.

03 Analyze performance bottlenecks. You don't need to eyeball charts; simply ask your agent to count log entries over a time range to quantify traffic spikes or sudden drops in activity.

04 Reduce credential risk. Since this MCP only grants scoped query permissions, you can get full observability without giving away global cloud access—a massive security win.

05 Target complex data structures. The tool allows searching deep into the JSON payload of log entries, letting you filter by keys that standard search functions miss.

Real-World Applications

Debugging a Payment Failure

A user reports a payment failure. Instead of manually checking multiple services, your agent uses stream_logs to apply filters for 'payment' and 'severity>=ERROR'. It quickly finds the specific log entry showing a database connection timeout.

Investigating High Latency

The system is slow. You instruct your agent to use stream_logs to analyze log entries across all services, filtering by high-volume requests in the last hour. The resulting data points directly to a sudden increase in logging verbosity causing resource exhaustion.

Security Audit of User Actions

You need to confirm who changed a record. Your agent uses stream_logs, filtering on `jsonPayload.userId="user_123"`, and retrieves every associated action, confirming the exact time stamp and method used.

Tracking Service Degradation

A service starts failing sporadically. You set up a prompt asking your agent to continuously run stream_logs, monitoring for patterns of 'warning' logs that increase in frequency over 15 minutes, giving you an early alert.

Patterns to Avoid

Over-permissioning the AI

✗ AVOID

Assuming the agent needs global read access to all cloud services just because it's 'observability'. This is a massive security risk.

✓ INSTEAD

Use this MCP. Its design limits your agent to scoped querying, meaning you only expose log reading capability on specific resources, not your entire infrastructure.

Vague Natural Language Prompts

✗ AVOID

Simply telling the AI 'Show me what went wrong.' This results in overwhelming data that requires manual sifting and guessing.

✓ INSTEAD

Be surgical. Use stream_logs to apply advanced filters, specifying `severity>=ERROR` or filtering by a known resource name to narrow the search immediately.

Ignoring Structured Data

✗ AVOID

Trying to find a user ID only using plain text searching, which fails if the ID is nested deep inside a JSON structure.

✓ INSTEAD

Leverage stream_logs' ability to parse JSON payloads. Filter directly on `jsonPayload.userId` to guarantee you hit the specific data point.

The Right Fit

Use this MCP when your primary need is operational debugging, auditing, or monitoring live service metrics derived from log files. If you need to read structured logs and perform deep filtering based on severity or JSON keys, this is exactly what you want. Don't use it if you are trying to write reports that summarize business transactions (use a database connector instead). You also shouldn't use it if your goal is data warehousing—this tool streams real-time operational data; for long-term, queryable datasets, look at dedicated BigQuery integrations.

The Pain of Log Diving

Right now, finding the root cause of a bug feels like forensic archaeology. You jump between Cloud Logging, your monitoring dashboard, and maybe an incident management tool. You spend time setting up filters—copying complex filter syntax into one tab, then opening another tab to view the results, only to realize you forgot to adjust the timeframe.

With this MCP, that process shrinks down to a single conversation with your agent. You just tell it: 'Find me all failed payment attempts from user X in the last two hours.' The agent executes the complex `stream_logs` query and hands you a clean, actionable list of results.

Stream Logs for Surgical Observability

The manual steps that disappear are the context switching, the repetitive copy-pasting of filter strings, and the sheer mental effort of synthesizing data from disparate dashboards. You no longer need to be a log query expert; you just need to know what question to ask.

This MCP doesn't just give you logs; it gives your agent surgical focus on those logs. It turns hours spent clicking through tabs into minutes spent reviewing an intelligent summary.

Google Cloud Logging Stream: 1 Tool

With this single tool, you can read and search log entries across your configured Google Cloud Log stream using advanced filtering syntax.

#	TOOL	DESCRIPTION
01	`stream_logs`	Reads and searches log entries from Google Cloud Logging, letting you apply advanced filters like minimum severity levels or JSON key matches.

See It in Action

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

U Fetch the last 100 log entries from our configured log stream.

I've retrieved the latest 100 entries. They are mostly standard info-level logs, but I noticed a warning around 10:14 AM.

U Stream logs filtering only for 'severity>=ERROR'.

I found 3 matching error logs. The most recent one indicates a database connection timeout.

U Search the logs for the user ID 'user_8819' in the JSON payload.

Applying the filter `jsonPayload.userId="user_8819"`, I found the specific event where the user triggered the payment webhook.

Frequently Asked Questions

01 **How do I use Google Cloud Logging Stream with user IDs?**

You can search for specific users by applying filters to the JSON payload. Use stream_logs and include a filter like `jsonPayload.userId="user_123"` to isolate all events related to that ID.

02 **Is Google Cloud Logging Stream safe for my production environment?**

Yes, it's designed with security in mind. It provides scoped access, meaning the agent only gets permission to query specific log resources you define, not global permissions across your entire cloud account.

03 **Can I filter by time range using stream_logs?**

Absolutely. You can always refine your search by adding explicit time constraints to your advanced GCP Logging filters, limiting the scope to a specific window of activity.

04 **Does Google Cloud Logging Stream handle different log types?**

It handles all standard Cloud Logging syntaxes. You can filter based on severity (e.g., `severity>=WARNING`) or target specific resource types within the logs.

05 **What is the best way to find recurring errors using stream_logs?**

Run a query over a large time sample, filtering by `severity=ERROR`, and then ask your agent to analyze the resulting payloads for common patterns or repeated error messages.

06 **Why limit the agent to a single Log Name?**

To enforce zero-trust security. An autonomous AI agent debugging an application shouldn't have access to read your organization's entire audit log history, IAM logs, or logs from other unrelated services.

07 **Can I use advanced GCP Log queries?**

Yes! You can pass any standard GCP Logging filter (e.g., `textPayload:"Exception"` or `jsonPayload.status="500"`) via the `filter` argument. The server automatically merges your filter with the strict `logName` restriction.

08 **How are the results ordered?**

Results are always returned in descending order (`timestamp desc`), meaning the AI agent gets the most recent logs first, which is ideal for real-time debugging.

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 `"google-cloud-logging-stream": { "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

Google Cloud Logging Stream 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 Google Cloud Logging Stream. 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	Google Cloud Logging Stream MCP
Server ID	`019e38a1-9bd6-7136-b51f-e43a0519a605`
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/google-cloud-logging-stream.