MCP SERVER | NO CODE | CLOUD HOSTED

Typesense Cloud MCP

Diagnose search performance without logging into a terminal.

Typesense Cloud MCP lets your AI agent manage and debug your fast search infrastructure directly through chat. Check cluster health, track real-time performance metrics, list API keys, or run complex multi-searches across multiple collections without ever touching a terminal command line. It gives you full operational visibility into every part of your distributed search stack.

A+ Quality Score 100/100

search-engine | cluster-monitoring | latency-tracking | api-key-management | federated-search

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

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.

Typesense Cloud MCP

6 tools available
Cloud-hosted on Vinkius

This MCP connects your Typesense Cloud endpoint to any AI agent, giving you hands-on control over your lightning-fast search infrastructure right from your chat window. Instead of digging through confusing CURL outputs or complex dashboard views to diagnose slow searches, you simply ask your agent what's wrong. You can verify if all nodes are online and ingesting data smoothly, measure latency spikes against real usage logs, or even execute multi-search commands across several collections simultaneously. When you connect this MCP via Vinkius, you get a single pane of glass for cluster forensics. This eliminates the need for deep knowledge of terminal diagnostics; you just delegate those complex operational checks directly to your highly capable AI.

Core Capabilities

01 — Check overall cluster status

Determines if all nodes are running and accessible, confirming uninterrupted data ingestion.

02 — Retrieve performance measurements

Gathers real-time metrics like usage logs, active search workloads, and resource consumption patterns.

03 — Run complex searches across multiple sources

Sends a single request to perform simultaneous multi-searches against several defined collections.

04 — Manage virtual collection names

Lists all aliases that abstract the concrete structure of your search data.

05 — Audit configured access keys

Retrieves a list of every API key currently set up for the cluster.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to this MCP and provide your active Typesense Host URL along with your specific API Key.

02 Connect your agent client (like Cursor or Claude) to the Vinkius catalog using these credentials.

03 Ask your AI client a question, like 'What's the current search latency?' The agent executes the necessary tool call and gives you the direct answer.

The bottom line is that you get immediate answers about your search cluster status without needing to write or execute complex command-line code.

Built For

This is for the DevOps engineer who gets tired of clicking through three different monitoring dashboards just to check if a simple endpoint is degraded. It's for the Search Engineer who needs to validate multi-search endpoints before they even hit the UI layer, and any Sysadmin who needs to audit alias structures across test and production environments instantly.

DevOps Engineer

They use this MCP to trigger on-the-spot node health interrogations, pulling metrics like latency peaks when a deployment is causing unexpected performance dips.

Search Engineer

They validate complex multi-search endpoints logically using the agent before mapping them into user-facing search features.

Site Reliability Engineer (SRE)

They evaluate lingering alias structures dynamically across different environments to ensure consistency and prevent unintended data exposure.

What Changes When You Connect

01 Quickly diagnose failures: Instead of manually checking status codes, use the `get_cluster_health` tool to instantly confirm if your nodes are online and fully operational.

02 Pinpoint slow spots: The `get_cluster_metrics` tool provides usage logs and latency thresholds, letting you see exactly when and why performance degrades.

03 Complex querying made easy: Don't write multi-search logic; just run `execute_multi_search` to query multiple collections simultaneously in one go.

04 Maintain clean architecture: Use `list_collection_aliases` to track which virtual names are pointing where, keeping your data structure auditable and clear.

05 Secure access management: The `list_api_keys` tool gives you a single source of truth for auditing every key used against the cluster.

Real-World Applications

The search results are suddenly slow after deployment

A developer asks their agent to check the performance metrics, running `get_cluster_metrics`. The agent reports that the 99th percentile spike hit 88ms, immediately pointing out a specific bottleneck in the resource usage before any code needs to be rolled back.

I need to validate three different data sets for an A/B test

A search engineer uses `execute_multi_search` to hit three distinct collections (e-commerce, global docs, and archived content) in one API call. This confirms the query structure works across all required sources before any UI development begins.

We think a key might have been compromised

A sysadmin asks the agent to run `list_api_keys`. The tool returns an exhaustive list, allowing them to immediately audit and revoke specific keys without manually navigating the cloud console.

We need a full inventory of our search data sources

A DBA runs the agent command for `list_collections` and receives a clean list of all active collections. They can then use `list_collection_aliases` to understand how those virtual names are exposed publicly.

Patterns to Avoid

Treating it like a basic search tool

✗ AVOID

Trying to debug performance issues by only running simple searches or checking one collection at a time.

✓ INSTEAD

To properly diagnose degradation, run `get_cluster_metrics` first. Then, use `execute_multi_search` and analyze the results against the observed latency spikes.

Ignoring alias structures

✗ AVOID

Assuming that because two collections share similar data types, they are pointing to the same underlying source.

✓ INSTEAD

Always run `list_collection_aliases` when you suspect a naming conflict or need to know which virtual name maps to which physical resource.

Ignoring security audit requirements

✗ AVOID

Not knowing exactly how many API keys exist, leading to potential over-privileging of service accounts.

✓ INSTEAD

The first step in any security review is running `list_api_keys` to get a complete and accurate inventory of all active credentials.

The Right Fit

Use this MCP if your primary problem is operational visibility: you need to know *why* your search engine is slow, or you need to validate complex read operations across multiple sources. If you are constantly checking health status (`get_cluster_health`) or pulling performance logs (`get_cluster_metrics`), this tool saves time. Don't use it if you only need basic CRUD operations on the data itself; that's a separate database connector. Also, don't rely solely on `list_collections` —always follow up with `list_collection_aliases` to understand the public interface of your data.

Debugging search latency used to mean logging into a terminal and running dozens of commands.

Today, figuring out why a search query is suddenly slow means juggling multiple dashboards. You copy performance metrics from one tab, cross-reference API key usage in another, then run separate diagnostic checks for every collection you suspect. It’s tedious, manual labor that wastes time and makes diagnosing the root cause a guessing game.

With this MCP, you simply ask your agent to check the cluster status or pull performance metrics. The system runs the necessary diagnostics—like calling `get_cluster_metrics` —and gives you a plain-language report on latency spikes and resource usage immediately.

Get deep visibility using Typesense Cloud MCP

The manual steps that vanish are the repetitive health checks, the need to manually list credentials, and having to piece together multi-search requirements across different documentation tabs. It’s a massive time sink.

Now you can delegate those complex forensics directly to your agent. You get instant operational control over your entire search infrastructure.

Typesense Cloud MCP: 6 Tools for Search Infrastructure

These tools allow you to perform deep diagnostics on your Typesense search engine, covering everything from basic health checks to complex federated searching.

#	TOOL	DESCRIPTION
01	`get_cluster_health`	Checks the operational health status of your entire Typesense search cluster.
02	`get_cluster_metrics`	Retrieves detailed performance and usage metrics for the Typesense cluster, including resource consumption.
03	`list_collection_aliases`	Lists all virtual aliases that map to your real search data collections.
04	`list_collections`	Lists every active search collection housed within the cloud cluster.
05	`list_api_keys`	Shows a full list of all API keys configured and used by the Typesense cluster.
06	`execute_multi_search`	Runs several different, complex search requests in one single API call using JSON arrays.

See It in Action

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

U Check the cluster health to verify Typesense is up in London.

The cluster responded successfully in 12ms. Status is 'ok'. Your Typesense nodes are fully operational.

U List all active collections inside this database environment.

You have 3 core collections currently structured: 1. 'global_docs' (150,000 docs) 2. 'commerce_catalog' (2,000 docs) and 3. 'test_index_v1' (0 docs). Need the exact schema for the e-commerce one?

U Fetch the performance metrics of the cluster and tell me if response times are above 100ms.

Pulling cluster metrics... The search_latency median is currently at 14ms globally. The 99th percentile spike reached 88ms. Fortunately, there are no endpoints dropping past 100ms. CPU stands at an idle 12% usage.

Frequently Asked Questions

01 **How do I check the overall health of my Typesense cluster using the Typesense Cloud MCP?**

Run the `get_cluster_health` tool. This immediately verifies if all nodes are online and reachable, telling you right away if there's a systemic outage or just a minor hiccup.

02 **Can I use Typesense Cloud MCP to test multiple collections at once?**

Yes, the `execute_multi_search` tool allows you to send one single request that targets and searches across several different data collections simultaneously. This is perfect for A/B testing.

03 **What should I run if my search performance seems inconsistent?**

Start by using `get_cluster_metrics`. This tool pulls usage logs, active workloads, and latency thresholds to help you pinpoint if the issue is CPU load or network throttling.

04 **How do I find out what my virtual data names are?**

Use `list_collection_aliases` to see all the abstract aliases. This tells you which virtual name clients should use, even if the underlying collection structure changes.

05 **Is there a way to audit who has access via Typesense Cloud MCP?**

Run `list_api_keys`. This tool provides an inventory of every API key currently configured for your cluster, which is essential for security audits and managing permissions.

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 `"typesense-cloud": { "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

Typesense Cloud 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 Typesense Cloud. 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	Typesense Cloud MCP
Server ID	`019d7617-3204-715d-aa70-6b9a7dfc1d80`
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/typesense-cloud.