MCP SERVER | NO CODE | CLOUD HOSTED

Typesense Vector Search MCP

Run semantic searches across indexed knowledge bases.

Typesense Vector Search lets your AI agent perform complex semantic searches and manage vector data entirely through conversation. Index documents, create new collections with specific schemas, and run combined text-filtering queries without writing a single line of API code.

A+ Quality Score 100/100

vector-search | semantic-search | rag | embedding-management | document-indexing

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

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 Vector Search MCP

6 tools available
Cloud-hosted on Vinkius

Connect this MCP to any compatible client to take autonomous control over your vector database. Instead of constructing CURL payloads or writing custom Python scripts for every query, you talk to your agent about what data you need. You can ask it to list all existing collections, create a new schema dataset with specific embedding structures, and immediately begin indexing documents by simply providing the JSON payload. The system handles the complex API calls in the background. This ability to manage vector storage—from creating schemas via `create_collection` to running advanced queries using `search_vectors`—means your agent becomes a full-time data engineer for your knowledge base. When you connect this through Vinkius, you get access to a robust set of tools that lets your AI client do all the heavy lifting on indexing and retrieval.

Core Capabilities

01 — Perform Semantic Search

Run vector similarity searches combined with text filters using `search_vectors`.

02 — Manage Collections

List all existing collections or retrieve the detailed schema for a specific collection using `list_vector_collections` and `get_collection_details`.

03 — Build New Data Schemas

Instantly provision new vector search datasets with custom schemas via `create_collection`.

04 — Index Documents

Add or update JSON documents in a collection using `index_document`, bypassing manual REST calls.

05 — Remove Data

Permanently delete specific records from a collection by ID using `delete_document`.

One Click on Vinkius — From Prompt to Execution

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

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

02 Instruct your AI agent to perform the desired data operation, like listing available collections or running a search query.

03 The agent executes the necessary tool calls through the connection, returning the requested data or confirmation status.

The bottom line is that you talk naturally to your agent, and it handles the entire back-end process of querying and managing your vector database for you.

Built For

This MCP is for people who build complex AI applications using proprietary data. It targets the Data Engineer who needs to manually ingest reference documents into a running collection, or the Backend Developer building RAG pipelines that require precise schema validation and multiple types of search.

Data Engineer

Needs to manually add missing knowledge base records or validate schemas for new data sources before they are used by the AI.

AI Application Builder

Builds conversational agents that need to perform advanced, filtered semantic searches over proprietary document sets at runtime.

Backend Developer

Writes and tests the core indexing logic for a service, needing an easy way to perform sanity checks on relevance scores or collection geometry.

What Changes When You Connect

01 You skip complex REST calls. Instead of writing code to `index_document`, you just ask your agent to update the data, and it handles the payload transfer automatically.

02 Advanced querying is simple. With `search_vectors`, you can combine natural language text filters with vector similarity results in a single chat command.

03 Schema management becomes conversational. Use `get_collection_details` to verify field geometries or `create_collection` to deploy a new, structured knowledge base instantly.

04 You maintain full control over your data lifecycle. Easily run `list_vector_collections` to map out every dataset you have, and use `delete_document` when a record is stale.

05 The entire process runs through your agent's chat window. You don't need dedicated terminal access or manual script execution for basic document mutations.

Real-World Applications

Updating Product Catalogs

A product manager wants to update pricing and descriptions across 50 old records in the inventory collection. Instead of writing a batch script, they prompt their agent: 'Update all products matching category X with the new JSON payload.' The agent executes `index_document` for each record automatically.

Debugging Retrieval Failures

A developer notices some search results are poor. They use `get_collection_details` to check the schema geometry, ensuring that their new field is mapped correctly and doesn't break the existing vector structure.

Building a New Knowledge Domain

A team needs a separate index for legal documents. They use `create_collection` to build the specialized schema, then run `list_vector_collections` to confirm the new dataset is ready before they start indexing.

Complex Research Queries

A researcher needs results about 'AI ethics' that were written specifically in 2023. They use `search_vectors`, combining a text filter ('year: 2023') with the vector query, getting highly relevant and narrow answers immediately.

Patterns to Avoid

Manual API scripting

✗ AVOID

Writing multi-step Python code or CURL commands every time you need to index a document, check a schema, or run a query.

✓ INSTEAD

Let your agent handle it. Use `index_document` for updates, `get_collection_details` for validation, and `search_vectors` for the final results—all in conversation.

Assuming schema compatibility

✗ AVOID

Running a search query against a collection that hasn't been properly initialized or whose field definitions were changed manually.

✓ INSTEAD

Always start by calling `list_vector_collections` and then use `get_collection_details` to confirm the exact geometry before making any changes.

Overwriting data blindly

✗ AVOID

Executing an update without confirming that the document ID exists or if the new JSON payload actually contains valid embedding vectors.

✓ INSTEAD

Use `get_collection_details` first. Then, run a test search using `search_vectors` before committing to any updates via `index_document`.

The Right Fit

Use this MCP if your core workflow involves semantic retrieval from complex, proprietary data and you need conversational control over the indexing process. This is essential for building advanced RAG systems where the search criteria must combine natural language filters with dense vector similarity searches. Don't use it if your needs are limited to simple key-value lookups (like finding a user by ID) or basic database operations that don't involve semantic understanding—a standard relational tool would be better. You need

this if you constantly have to manage schemas (`create_collection`) and ensure data fidelity across multiple, distinct knowledge domains.

The pain of manual vector management

Today, managing a semantic database means jumping between documentation, writing API wrappers in Python, and manually constructing complex JSON payloads for every single operation. If you need to index 50 documents, that's 50 separate calls. If you change the schema geometry, it’s another code deployment cycle just to confirm the fields are correct.

With this MCP connection, those manual steps vanish. You simply tell your agent what needs doing—like 'Add these records and make sure they fit the existing structure.' The system handles the API calls for `index_document` and validates the process without you ever touching a terminal command or writing a single line of boilerplate code.

Get full control with Typesense Vector Search

You no longer need to manually run `list_vector_collections` just to see what datasets exist, nor do you have to worry about schema mismatches. The agent surfaces the available collections and their definitions for you.

What's different now is that your AI client treats the entire vector database like a single API endpoint accessed via natural language. It’s immediate, conversational control over complex data infrastructure.

Typesense Vector Search: 6 Tools Available

These tools give your agent the power to read, write, and structure data within your vector database using conversational commands.

#	TOOL	DESCRIPTION
01	`search_vectors`	Runs a combined search that uses both text filters and numerical vector similarity queries.
02	`create_collection`	Builds a new search dataset by defining its specific required schema using JSON details.
03	`delete_document`	Removes a document from any collection, and this action cannot be reversed.
04	`get_collection_details`	Retrieves the full schema definition and metadata for a specific data collection.
05	`index_document`	Adds or updates an existing document in a collection by providing its name and JSON payload.
06	`list_vector_collections`	Fetches a list of every available vector collection within the current Typesense instance.

See It in Action

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

U List all active collections on this vector cluster. Do I have any collections initialized yet?

I've listed 2 active collections: 'customer_kb_index' (configured with a 1536-dimensional embedding schema) and 'products_inventory'. Need me to execute a vector search on either limit?

U I have an embedding snippet: [0.34, 0.42, 0.99...]. Delete the document carrying ID 'test-123' and re-index it using this JSON data on collection 'faqs'.

Document 'test-123' has been successfully wiped. I've seamlessly pushed the new JSON package into 'faqs', updating the embedding vectors as instructed.

U Explain the schema definitions used inside the 'products_inventory' collection.

The collection 'products_inventory' uses 4 strict fields: `product_id` (string), `name` (string), `popularity` (int32), and critically `embeddings` formulated as a `float[]` of 768 dimensions representing product semantics.

Frequently Asked Questions

01 **How do I check what collections are available using Typesense Vector Search MCP?**

You ask your agent to list all vector collections. The tool `list_vector_collections` will immediately provide a roster of every dataset configured on your instance.

02 **Can I create a new schema with the Typesense Vector Search MCP?**

Yes, you can use `create_collection`. You just need to supply the desired field structure as a JSON object, and the tool provisions the entirely new semantic dataset.

03 **What is the difference between using search_vectors and indexing documents?**

`search_vectors` reads data; it executes complex queries combining text filters with vector similarity. `index_document`, conversely, writes data by adding or updating a record in a collection.

04 **Is the action of deleting documents permanent using Typesense Vector Search MCP?**

Yes, the `delete_document` function permanently removes records by their ID. Treat this tool with care, as there is no undo feature for this operation.

05 **Does the Typesense Vector Search MCP support combined search queries?**

Absolutely. The `search_vectors` tool lets you combine traditional text filtering with vector similarity searches in one single request.

06 **Can the agent perform vector plus text-filtering search combined natively?**

Yes. Provide the agent with the collection name alongside the text payload and tell it the exact vector structure. It leverages internal filters querying natively and returns the nearest neighbors with exact accuracy scores.

07 **How do I make the AI create a semantic collection ready for embeddings (OpenAI 1536 dims)?**

Ask the agent to use 'create_collection'. Provide standard JSON declaring the name, the field structure, and explicitly define the `float[]` field tracking the 1536 dims length. The cluster will spin the framework up instantly.

08 **Can it delete problematic vectors holding bad geometry data manually?**

Absolutely. Supplying the explicit collection target and the item 'id' to the delete_document prompt securely wipes out all traces from the dataset. Use this sparingly as it can't be undone easily.

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-vector-search": { "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 Vector Search 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 Vector Search. 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 Vector Search MCP
Server ID	`019d7617-527a-71ad-a8a6-4ab8bb65c437`
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-vector-search.