MCP SERVER | NO CODE | CLOUD HOSTED

OpenSearch Vector MCP

Manage your entire vector store via conversation.

OpenSearch Vector MCP lets your AI client treat OpenSearch like a true vector database. You can create k-NN indexes for cosine similarity and manage the entire embedding workflow through conversation. Run complex similarity searches, upsert document embeddings with metadata, or inspect index health without writing any `curl` commands.

A+ Quality Score 100/100

vector-database | k-nn | search-engine | embeddings | indexing

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

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.

OpenSearch Vector MCP

6 tools available
Cloud-hosted on Vinkius

Need to run semantic searches on your knowledge base? This MCP connects OpenSearch directly to your AI client, turning it into a powerful vector store. You don't have to leave your chat window to perform complex database operations. Your agent can now execute k-Nearest Neighbors queries against any index, retrieving documents based on conceptual similarity rather than keywords.

It handles the full lifecycle of vector data. Need to start fresh? You can provision new k-NN indexes optimized for specific dimensions and similarities. Later, when you have content ready, your agent will upsert those vectors with associated metadata. The whole process—from checking an index's current count to running a deep similarity search—is accessible via natural conversation. By connecting this MCP through the Vinkius catalog, you get immediate access to robust vector management for your entire suite of AI applications.

Core Capabilities

01 — Search similar documents

Run k-Nearest Neighbors queries against an index using a provided embedding array to find conceptually related data.

02 — Manage indexes

List all existing OpenSearch indexes and retrieve detailed configuration settings for any specific index.

03 — Create vector indexes

Provision new k-NN indexes, setting them up with required dimensions and cosine similarity optimization.

04 — Add document embeddings

Insert or update a single vector document directly into the index along with its metadata.

05 — Remove documents

Delete specific vector documents from the embedding space using their unique identifier.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to this MCP and provide your OpenSearch Host, Username, and Password credentials.

02 Your AI client authenticates with the connection details, making all vector data operations available in a conversational context.

03 You instruct your agent to perform an action—like creating an index or running a search—and it executes the query against your live cluster.

The bottom line is that you manage and query your entire vector store through simple conversation, bypassing complex command-line interfaces.

Built For

ML engineers who need to test similarity queries against production embeddings without writing `curl` commands. RAG developers building retrieval pipelines that require stable index management. Data teams tired of switching between chat and Kibana dashboards for basic health checks.

RAG Developer

Indexes context documents using the MCP's create index tool, then uses search to retrieve relevant passages for generative pipelines.

ML Engineer

Tests similarity queries against production embeddings by providing a dense float vector array and running a k-NN search.

Data Architect

Inspects index health, checks document counts, or provisions new indexes using the list_indexes tool instead of writing complex API calls.

What Changes When You Connect

01 Run k-Nearest Neighbors searches without leaving your chat. Provide a dense float vector and let the MCP perform a similarity query using the search tool.

02 Avoid manual dashboard navigation. Use list_indexes to see all cluster indexes, check their health status, and get document counts instantly.

03 Build reliable RAG pipelines by provisioning new k-NN indexes with create_index, configuring specific dimensions for cosine similarity.

04 Keep your data clean and current. Index a single vector document using index_document or delete outdated records with delete_document.

05 Understand exactly what you're dealing with. Get detailed index settings and mappings for any cluster component by calling get_index.

Real-World Applications

Troubleshooting a knowledge base

A data team member needs to know which indexes exist before starting work. They simply ask the agent, and it uses list_indexes to provide an immediate overview of all available vector stores.

Building a new feature store

An ML engineer wants to test embeddings for customer feedback. Instead of writing boilerplate code, they instruct the agent to use create_index to provision a dedicated 1536-dimensional k-NN index.

Retrieving context in real time

A developer wants to answer a complex question using RAG. They pass the query embedding to the agent, which uses search to find the top 5 most similar documents from the production knowledge base index.

Cleaning up stale data

The team needs to remove old user profile embeddings that are no longer relevant. They tell the agent to use delete_document, referencing the specific document IDs for a clean sweep of obsolete vectors.

Patterns to Avoid

Manual API calls

✗ AVOID

Opening Swagger UI or writing complex cURL commands just to check if an index exists or what its dimension is.

✓ INSTEAD

Just ask your agent. Use list_indexes to see every index, and then use get_index for specific details—all in conversation.

Confusing vector dimensions

✗ AVOID

Attempting to run a search using an embedding that has the wrong dimensionality (e.g., 768 when the index expects 1536).

✓ INSTEAD

When creating or updating embeddings, ensure you use create_index first to verify and set up the correct vector dimensions for your specific data type.

Over-relying on dashboards

✗ AVOID

Failing to notice an index is deprecated or has a health warning because they only check the Kibana dashboard.

✓ INSTEAD

Use list_indexes and get_index. These tools report current health status and detailed configurations directly through your agent's response.

The Right Fit

Use this MCP if your primary need is to manage, store, or query structured vector embeddings within an existing OpenSearch cluster. This includes running k-NN searches for semantic similarity (search tool) or performing full index lifecycle management (create_index, list_indexes). Don't use it if you simply need to search unstructured text fields without embedding generation; those require a standard search engine MCP instead. Also, don't use it if your data lives in a completely different vector store type; this is strictly for OpenSearch KNN functionality.

Managing Vector Data Used To Be a Command-Line Nightmare

Every time you need to check an index's health, verify its dimension count, or provision a new space for embeddings, you used to jump through hoops. That meant writing complex `curl` commands, managing JSON payloads, and jumping between the OpenSearch dashboard and your chat window just to get basic status updates.

Now, you tell your agent what you need—like checking all available indexes with list_indexes. The system handles the API calls in the background and gives you a clean, conversational summary of document counts and health status. It’s immediate.

OpenSearch Vector MCP: Direct Similarity Search

Before this MCP, executing a conceptual search meant preparing the embedding vector array yourself and sending it in a highly structured payload. If your top-K limit was wrong or the index name changed, the whole query failed.

Now you just need to ask for it. Your agent handles the precision required for k-NN searches, making sure the correct index is targeted and the search parameters are perfect every time.

OpenSearch Vector with 6 Tools

Use these tools to create, read, update, delete, and search vector indexes directly from your AI agent.

#	TOOL	DESCRIPTION
01	search	Executes a K-Nearest Neighbors search to find documents conceptually similar to a provided vector embedding.
02	list_indexes	Retrieves a list of all explicitly created indexes residing on the OpenSearch cluster.
03	get_index	Fetches detailed mapping and settings for a specific OpenSearch index name.
04	index_document	Inserts or updates a single vector document directly into the OpenSearch KNN index, preserving its metadata.
05	delete_document	Removes an entire vector document from the designated OpenSearch embedding space using its ID.
06	create_index	Sets up a new, native OpenSearch KNN index optimized for receiving and storing vector embeddings.

See It in Action

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

U List all vector indexes in my OpenSearch cluster.

Found 4 indexes. 1) 'product-embeddings' — 45,230 docs, green health, 768 dimensions. 2) 'support-tickets' — 12,800 docs, green, 1536 dimensions. 3) 'knowledge-base' — 3,450 docs, green, 384 dimensions. 4) 'test-index' — 50 docs, yellow, 768 dimensions. Want me to inspect the mapping of any of these?

U Find the 5 most similar documents to this embedding in the knowledge-base index.

Top 5 results from 'knowledge-base' (cosine similarity): 1) 'API Rate Limiting Guide' — score: 0.94. 2) 'Authentication Best Practices' — score: 0.89. 3) 'Error Handling Documentation' — score: 0.87. 4) 'SDK Migration Guide v3' — score: 0.83. 5) 'Webhook Configuration' — score: 0.81. Want me to pull the full content of the top result?

U Create a new k-NN index called 'customer-feedback' with 1536 dimensions.

Index 'customer-feedback' created successfully. Configuration: k-NN enabled, 1536 dimensions, cosine similarity, NMSLIB engine. The index is empty and ready for document ingestion. Want me to upsert a test document to verify the mapping?

Frequently Asked Questions

01 **How do I start with OpenSearch Vector MCP?**

Start by subscribing to this MCP and providing your OpenSearch credentials. Once connected, you can immediately use list_indexes to see what indexes are available in your cluster.

02 **What is the difference between search and index_document using OpenSearch Vector MCP?**

The 'search' tool reads data: it takes an embedding and finds similar documents. The 'index_document' tool writes data: it takes an embedding and saves it to the cluster.

03 **Can I create a new index with OpenSearch Vector MCP?**

Yes, you use the create_index tool. You specify if you want k-NN enabled and what vector dimensions (like 768 or 1536) the index needs.

04 **How do I find out about an existing OpenSearch index?**

Use get_index. This tool retrieves the full mapping, settings, and engine configuration for any specific index you point it toward.

05 **Does OpenSearch Vector MCP only handle text searches?**

No, this MCP is specifically designed for vector data. It executes k-NN searches on dense float vectors (embeddings), making it ideal for semantic similarity tasks.

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 `"opensearch-vector": { "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

OpenSearch Vector 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 OpenSearch Vector. 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	OpenSearch Vector MCP
Server ID	`019d75e9-e793-739b-9899-3ac45e85b9c3`
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/opensearch-vector.