

MCP SERVER

NO CODE

CLOUD HOSTED

Vertex AI Vector Search MCP

Semantic Search Across Billions of Embeddings

Vertex AI Vector Search brings Google's massive vector matching power directly into your agent. You can search billions of semantic embeddings and manage complex index endpoints without leaving your chat window or IDE. This MCP lets you find related data by meaning, not just keywords, giving your LLM context based on deep geometric similarity calculations.

A+ Quality Score 100/100

vector-search

embeddings

nearest-neighbor

semantic-matching

cloud-infrastructure

low-latency



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.

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.

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.

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

Encryption at rest

Ed25519

PKI vault signatures

24h TTL

Ephemeral session keys

V8 Isolate

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.

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.

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.

06 — Honeypot Trap System

Phantom credentials are injected into isolated environments. If a honeypot 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.

FinOps

Cost tracking per tool, payload compression savings, budget optimization signals, and consumption trends.

Firewall & DLP

PII redaction activity, sensitive data protection counters, and security event timeline.

Agent Activity

Which AI clients are connecting, how often, and what they're doing — real-time session tracking.

Tool Health

Slowest and most error-prone tools, with actionable root-cause insights and performance baselines.

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.

Vertex AI Vector Search MCP

6 tools available

Cloud-hosted on Vinkius

Need to pull information that goes beyond simple keyword matching? Vertex AI Vector Search connects the power of Google Cloud's massive vector database directly to your agent. Instead of digging through console dashboards or writing complex API calls, you prompt your client and it handles the search. It takes a query—whether it's a float array or text—and finds the most semantically similar data points across billions of records in low latency. You can also manage your infrastructure on the fly: ask to list all active vector indexes or check if an index is properly exposed for production traffic. This ability to administer and search massive datasets conversationally makes it indispensable, especially when connected through the Vinkius catalog.

Core Capabilities

01 — Find nearest neighbors

Execute a vector similarity search using a query array against specific index endpoints to locate highly related data IDs.

03 — Check index configuration

Retrieve detailed metadata and current setup information for any single, specific vector index.

05 — Verify active endpoints

List all network endpoints that expose indexed data, confirming which indexes are currently ready to receive production search traffic.

02 — List all indexes

View every vector index defined within your entire Google Cloud project for an overview of available datasets.

04 — Track deployment jobs

Monitor the status of multi-terabyte index builds or updates by listing long-running operational tasks in your cloud queue.

One Click on Vinkius — From Prompt to Execution

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

- 01** First, you point your agent toward the specific index or endpoint ID and provide the query vector (a JSON array of floating-point numbers).
- 02** Next, the MCP executes a nearest neighbor lookup against the target Google Cloud resource, comparing the query vector to millions of stored embeddings.
- 03** Finally, your agent receives a list of top matching data IDs along with their calculated distance scores, showing how semantically close they are.

The bottom line is you get deep contextual knowledge from massive datasets without ever writing boilerplate search code.

Built For

This MCP is for the ML Ops engineer who needs to check index health and deployment status without logging into a dashboard, or the Data Scientist who needs to test new float arrays against production endpoints instantly.

MLOps Engineer

Checks the progress of multi-hour index deployments using `list_vector_operations` and verifies endpoint availability via `list_index_endpoints`.

RAG Data Scientist

Pushes experimental float arrays directly into production endpoints using `search_nearest_neighbors` to gauge proximity precision on the fly before full deployment.

Backend Architect

Verifies complex vector database configurations, shard counts, and node health by calling `get_index_details` across organization-wide deployments.

What Changes When You Connect

-
- 01** Stop manually checking console logs. You can use `list_vector_operations` to monitor multi-terabyte index build progress right through chat.

 - 02** Instantly test your data structures. The `search_nearest_neighbors` tool lets you push experimental float arrays into production endpoints without writing boilerplate code.

 - 03** Know what's live and ready for traffic. Use `list_index_endpoints` to confirm which underlying deployed index versions are actually receiving requests.

 - 04** Get a full picture of your data assets. Running `list_vector_indexes` gives you an immediate inventory of every vector index in the entire project.

 - 05** Avoid configuration surprises. Calling `get_index_details` provides all the necessary metadata and setup info for any specific index, confirming its dimensionality and current state.
-

Real-World Applications

Finding Context in a Massive Document Vault

A data scientist needs to check if their new document chunk is semantically similar to existing records. They simply prompt their agent: 'Find the top 5 matches for this vector.' The agent uses `search_nearest_neighbors`, returning relevant IDs and proximity scores instantly.

Verifying Database Readiness

A backend architect needs to know if a specific index is ready for production traffic across multiple regions. They use `list_index_endpoints`, which confirms exactly which deployed index iteration is currently exposed and accepting requests.

Checking Infrastructure Health Before Launch

An MLOps engineer is deploying a new product catalog index. Instead of navigating the console, they ask their agent to run `list_vector_operations`. The agent replies with the status and ETA, confirming the deployment job is still active.

Inventorying All Available Data Sources

A developer starts a new project and needs to know what vector databases exist in the cloud. They run `list_vector_indexes`, immediately getting an overview of all available core indexes they can point their agent toward.

Patterns to Avoid

Using keyword search for context

✗ AVOID

The user asks the agent to 'Find documents mentioning quantum computing,' but those documents are stored using conceptual embeddings, and basic keyword searching fails entirely.

✓ INSTEAD

You must use a vector similarity tool. Provide your query vector array and call `search_nearest_neighbors`. This finds documents based on meaning, not just matching words.

Assuming index status

✗ AVOID

A user assumes an index is available for querying because it exists in the list of all indexes, but it hasn't been deployed to a live endpoint yet.

✓ INSTEAD

Always verify availability first. Use `list_index_endpoints` before calling `search_nearest_neighbors`. This confirms the data is actually exposed and ready.

Ignoring job progress

✗ AVOID

A user assumes a massive index build finished just because it started, but the process failed silently or stalled in the background.

✓ INSTEAD

Check the operational status. Use `list_vector_operations` to review the persistent task timeline and ensure the build completed successfully before using any search tools.

The Right Fit

Use this MCP if your core need is semantic understanding—matching ideas, concepts, or relationships rather than matching exact words. If you are building a Retrieval Augmented Generation (RAG) system, you *must* use vector search. This tool excels at pinpointing the nearest neighbors across billions of embeddings using `search_nearest_neighbors`.

Don't use this if: 1) You just need to perform simple CRUD operations (creating or updating records); those require a different type of connector. 2) Your data is structured in basic tables and you only need filtered results based on exact text matches; a standard database query tool will be faster and simpler.

If you are dealing with high-dimensional, conceptual embeddings, this MCP gives you the power to manage the infrastructure (`list_vector_indexes`, `get_index_details`) *and* run the search in one place.

The Headache of Multi-Dashboard Data Discovery

Today, if your team needs to understand how closely related two documents are based on concept—not just keywords—you have a manual nightmare. You're forced to jump between the Cloud Console, checking index status in one tab, verifying endpoints in another, and finally running the search job in a third window. It's slow, brittle, and requires constant context switching.

With this MCP, your agent handles all that complexity for you. You simply ask it to find semantic matches by providing the necessary query vector. The system coordinates the lookups across endpoints and indexes automatically, giving you immediate results without ever touching a console dashboard.

Accessing Deep Context with Vertex AI Vector Search

The major manual steps that disappear are the tedious checks: listing all available vector indexes

Now, your agent manages the entire data pipeline conversationally. You get direct, actionable search

(list_vector_indexes), confirming which endpoints are live (list_index_endpoints), and verifying if a deployment job is complete (list_vector_operations). These tasks used to require dedicated navigation.

results using search_nearest_neighbors—a massive reduction in friction that lets you focus entirely on insights.

Vertex AI Vector Search MCP: 6 Tools

Use these six tools to perform deep semantic searches, list indexes, check configurations, and monitor long-running vector operations in Google Cloud.

#	TOOL	DESCRIPTION
01	<code>get_index_details</code>	Retrieves the specific metadata and configuration settings for a single vector index.
02	<code>list_deployed_indexes</code>	Lists all vector indexes that have been successfully deployed to an active search endpoint.
03	<code>list_index_endpoints</code>	Retrieves a list of every index endpoint configured within the current project.
04	<code>list_vector_indexes</code>	Lists all vector indexes that exist in the entire Google Cloud project scope.
05	<code>list_vector_operations</code>	Shows a timeline of long-running tasks, like index builds or updates, currently running in the cloud queue.
06	<code>search_nearest_neighbors</code>	Performs a precise vector similarity search by inputting an endpoint ID, index ID, and a query vector array.

See It in Action

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

U List all our active vector indexes on the current GCP project.



I've scanned your infrastructure on Vertex AI. You currently have 2 core vector indexes active:

- `product_catalog_dense` (ID: 4858..., Dimensions: 768)
- `document_vault_prod` (ID: 3959..., Dimensions: 1536)

Would you like me to inspect endpoints verifying which of these are currently exposed for search?

U Check for any long-running vector deployment operations currently uncompleted.



I've queried the Vertex AI Operations log.

There is currently `1` pending operation:

- **Operation ID:** `74803923`
- **Target:** Deploying index `document_vault_prod` to endpoint `112xyz`.
- **Status:** `RUNNING` (Approx 45% completion, started 22 minutes ago).

No errors or stalling reported on your cluster. I can poll this periodically if you wish.

U Find the 3 nearest neighbors mapping to endpoint '39xl' array index ID 'dep_30' using vector [-0.2, 0.5, 0.0].



I've passed the floating-point vector directly to the Vertex instance. The semantic similarities fetched from `dep_30` are:

1. Base ID `prod-281x` (Distance: 0.12)
2. Base ID `prod-994y` (Distance: 0.17)
3. Base ID `prod-110m` (Distance: 0.31)

This indicates extremely tight clustering to item `#281x` in the embedding space.

Frequently Asked Questions

01 How do I check if an index is ready to be searched with Vertex AI Vector Search?

You must first use `list_index_endpoints`. This tool lists all active network endpoints, confirming which specific underlying deployed index iterations are currently receiving production traffic and can accept search queries.

02 What is the difference between listing indexes and checking endpoint details with Vertex AI Vector Search?

`list_vector_indexes` gives you a list of all existing datasets in the project. `list_index_endpoints`, however, tells you which of those indices are currently set up to be used for live search traffic.

03 Can I monitor index build progress using Vertex AI Vector Search?

Yes, use `list_vector_operations`. This tool lets your agent query the cloud queue and review persistent long-running task timelines, letting you know if a multi-terabyte build is still running or has failed.

04 Do I need to manually write API calls for semantic search with Vertex AI Vector Search?

No. Your agent handles the complex JSON formatting and endpoint calling. You just provide the query vector, and the MCP executes the `search_nearest_neighbors` call.

05 What should I do if my index configuration seems wrong? (Vertex AI Vector Search)







Run `get_index_details` for that specific index. This retrieves all metadata and configuration details, allowing you to verify its dimensionality or operational settings without guessing.

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 <code>"vertex-ai-vector-search": { "url": "..." }</code>
 Windsurf	MCP Settings → <code>mcp_settings.json</code> → 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

Vertex AI 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](https://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 Vertex AI 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	Vertex AI Vector Search MCP
Server ID	019d761b-f59f-71ac-8e1b-89f201a24403
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/vertex-ai-vector-search.