MCP SERVER | NO CODE | CLOUD HOSTED

Pinecone MCP for AI Agents

Manage your vector database and query embeddings using natural language.

Pinecone MCP lets your AI agent talk directly to your vector database. Query embeddings, manage collections, and pull performance stats through natural conversation. It turns your vector store into an accessible knowledge base for your agent without you having to write custom scripts.

A+ Quality Score 100/100

semantic-search | vector-embeddings | knowledge-graph | high-performance | ai-infrastructure

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JULY 2026
vinkius.com/mcp/pinecone

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.

Pinecone MCP

7 tools available
Cloud-hosted on Vinkius

This MCP gives your AI client direct access to your Pinecone environment. Instead of jumping between your terminal and the Pinecone console to check on your data, you can just ask your agent to do it for you. You can have your agent look up specific vector details, check how full your pods are, or clean up old data on the fly. It makes managing a complex vector space feel like a conversation rather than a series of API calls. Whether you're building a RAG system or just need to audit your storage, this connection puts the power of your vector store right into your chat interface. You can find this and thousands of other tools in the Vinkius catalog to build out your entire AI stack.

Core Capabilities

01 — Find similar vectors

Search your database to find the most relevant embeddings based on your input.

02 — Fetch specific data

Pull precise vector records using their unique IDs.

03 — View index list

See every index currently existing in your environment at once.

04 — Check index settings

View the configuration details and topology parameters of a specific index.

05 — Monitor usage stats

Pull real-time health checks and capacity limits for your pods.

06 — List collections

See all snapshot arrays and grouped data in your space.

07 — Remove vectors

Delete specific data points from an index to free up space.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to the Pinecone MCP in the Vinkius marketplace.

02 Add your Pinecone API key to your AI client configuration.

03 Ask your agent to run queries or pull stats on your vector stores.

The bottom line is you get a conversational interface for your vector database.

Built For

For the AI engineer who's tired of writing boilerplate Python scripts to test RAG relevance or the data custodian who needs to audit vector counts across multiple environments quickly.

AI/ML Engineer

Testing how well semantic chunks are being retrieved during RAG development without writing test scripts.

Data Custodian

Auditing storage limits and cleaning up old vectors across production indexes via terminal prompts.

Agent Builder

Building dynamic knowledge retrieval systems that query vector stores on demand.

What Changes When You Connect

01 Check RAG relevance quickly. Use query_vectors to see if your agent is pulling the right context.

02 Monitor storage without the console. Use get_index_stats to see your pod capacity in real time.

03 Track your snapshots. Use list_collections to see all your grouped data arrays.

04 See your whole environment. Use list_indexes to get a bird's eye view of your vector store.

05 Faster debugging. You can check your topology parameters without hunting through menus.

06 Easier cleanup. You can wipe specific user data or test records with a single command.

Real-World Applications

Testing RAG retrieval

An engineer asks the agent to find similar vectors for a query to see if the retrieved context is relevant for the prompt.

Configuration audit

An architect asks the agent to describe the index configuration to verify the mathematical dimensions and pod settings.

Targeted retrieval

A developer uses fetch_vectors to pull specific data points by their unique IDs for a precise data check.

Capacity planning

A data custodian asks for the vector count stats to see if they need to scale their production environment.

Data cleanup

A developer tells the agent to remove all records belonging to a specific user ID to comply with a deletion request.

Patterns to Avoid

Manual script fatigue

✗ AVOID

Writing a Python script every time you want to check a vector count or pod status.

✓ INSTEAD

Use get_index_stats directly in your chat to get real-time answers without leaving your workspace.

Guessing index status

✗ AVOID

Not knowing which pods are full or what the topology parameters are.

✓ INSTEAD

Use list_indexes and describe_index to see the full environment structure at a glance.

Hardcoding IDs

✗ AVOID

Trying to remember every vector ID for manual retrieval.

✓ INSTEAD

Use query_vectors to find the right data first, then use fetch_vectors if you need specific details.

The Right Fit

Use this if you need a way to interact with Pinecone via your AI client without writing extra glue code. It's perfect for rapid prototyping of RAG systems or for quick administrative tasks like auditing and cleanup. Don't use it if you need to perform complex, high-volume batch processing of millions of vectors at once. For heavy-duty ETL pipelines, you should stick to the standard Pinecone SDK or a dedicated data engineering tool. This MCP is for conversational interaction and quick management tasks.

Pinecone MCP: Stop wasting time on manual vector database audits

You're currently jumping between your IDE, a web browser, and the Pinecone console. You have to copy IDs, check pod capacities, and write one-off

With this MCP, you just ask your agent to check the stats or find a specific vector. You get the answer in the same window where you're already

Python scripts just to see if your semantic search is actually returning the right chunks. It's a context-switching nightmare that kills your flow.

building your app. It turns a five-minute chore into a five-second conversation.

Pinecone MCP: Scale your RAG infrastructure with conversational vector queries

Managing multiple collections and index configurations usually involves keeping a mental map of every parameter and snapshot. If you need to delete data or check dimensions, you're digging through nested menus and documentation.

You can now list every index or describe a specific configuration with a single prompt. You get a clear, organized view of your infrastructure without ever leaving your workspace.

Pinecone MCP: 7 Tools for Vector Database Management

Use these tools to query, fetch, and manage your Pinecone vector store via your AI client.

#	TOOL	DESCRIPTION
01	delete_vectors	Removes specific vectors from an index to free up space or clear data. This is useful for maintaining storage hygiene.
02	describe_index	Shows the configuration details and topology of a specific index. Use this to verify your pod settings and dimensions.
03	fetch_vectors	Retrieves specific vectors using their unique IDs for precise lookups. It's perfect for pinpointing exact records.
04	get_index_stats	Pulls real-time health checks and capacity limits for your pods. This helps you monitor your storage usage at a glance.
05	list_collections	Lists all index collections to help you organize snapshot arrays. Use it to see your grouped data structures.
06	list_indexes	Shows every index currently existing in your Pinecone environment. It gives you a clear view of your entire vector store.
07	query_vectors	Finds and returns the most similar vectors and metadata for a given input. This is the core tool for semantic search.

See It in Action

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

U Check the vector count stats for the index named `document-embeddings`.

Index Stats: `document-embeddings`

METRIC	VALUE
Vector Count	45,920
Pod Capacity	90% Full
Dimensions	1536

The index is approaching capacity. You might want to consider scaling your pods soon.

U Delete all vectors belonging to the user ID 'auth-abc123' namespace.

✅ **Deletion Successful**

I've wiped all vectors associated with the `auth-abc123` namespace from your index. The space has been cleared.

U List all existing collections created in my Pinecone environment.

You have the following active collections:

- `backup-q1-2026` (Snapshot)
- `knowledge-base-staging` (Snapshot)

Let me know if you need more details on any of these.

Frequently Asked Questions

01 **Can I use the Pinecone MCP to manage my vector database?**

Yes, this MCP connects your AI client directly to Pinecone. You can query data, check stats, and manage your indexes using natural language.

02 **How do I check my vector capacity with Pinecone?**

You can simply ask your agent to pull the usage stats. It will check your pod capacity and vector counts in real time.

03 **Can my AI agent delete vectors for me?**

Yes, you can give your agent the command to delete specific vectors. This is great for clearing out test data or handling user deletion requests.

04 **Is the Pinecone MCP good for testing RAG?**

It's perfect for RAG. You can ask your agent to run queries and see what context it retrieves, helping you debug relevance without writing scripts.

05 **How do I see all my indexes in Pinecone?**

Just ask your agent to list your indexes. It will return a list of all the indexes currently in your environment.

06 **Can I use this to check my index configurations?**

Yes, your agent can describe any specific index to show you its configuration, topology, and other parameters instantly.

07 **Can the AI execute raw vector similarity searches?**

Yes, absolutely. Once you supply the raw semantic embedding coordinates (normally a float array generated previously), the LLM can funnel it through the `query_vectors` tool. The Pinecone DB will process this and return the top-K closest vector matches along with embedded metadata.

08 **How do I check my remaining vector storage capacity?**

It's extremely simple. Just ask the connected AI agent to 'Get the index stats'. It will internally call `get_index_stats` against the specified index namespace, returning total vector count and physical dimensionality limits to your chat window.

09 **Is it safe to delete vectors dynamically using the chat terminal?**

Yes, but with standard precautions. The `delete_vectors` tool operates exactly as the official SDK. As long as you maintain clear contextual scopes and ID filtering in your prompts, the execution is purely deterministic and secure.

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 `"pinecone": { "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

Pinecone 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 Pinecone. 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	July 2026
MCP Server	Pinecone MCP
Server ID	`019d75f3-2b6b-72d6-bba4-f5773344ccd9`
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/pinecone.