MCP SERVER | NO CODE | CLOUD HOSTED

Meilisearch MCP

Automate Indexing and Search Operations

Meilisearch MCP lets you automate your entire search engine lifecycle. Connect it to any AI client and manage everything from creating indexes and updating documents in bulk, to running complex searches across thousands of records. You control the data flow without ever touching a dashboard.

A+ Quality Score 98.33/100

search-engine | indexing | full-text-search | meilisearch | api-management

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/meilisearch

SCAN TO VIEW

The infrastructure that powers AI agents in the real world.

Vinkius connects AI to the world's software through secure, enterprise-grade infrastructure — enabling real-world execution at scale, built on the Model Context Protocol (MCP).

Your AI Connections Run Through Vinkius Cloud

The world's largest managed MCP catalog

Vinkius is the cloud infrastructure where AI agents connect 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.

Meilisearch MCP

44 tools available
Cloud-hosted on Vinkius

Need to keep your search database fresh? This MCP connects your Meilisearch instance directly to your agent. Instead of logging into a dedicated dashboard or writing boilerplate scripts just to check if your data is indexed, you tell your AI client what needs doing. You can list all existing indexes and manage their settings, running tasks like atomic swaps for zero-downtime updates. Want to clean up old data? Your agent handles complex deletion operations, whether it's removing a single document by ID or filtering out entire batches of records based on specific criteria. The ability to pull metadata details helps you monitor the engine's health in real time. It's powerful management for developers and content teams alike, letting you perform heavy data engineering tasks right where your AI client is working, making it one of the most critical tools available in the Vinkius catalog.

Core Capabilities

01 — Index Management

You can list all indexes and create new ones, or swap multiple indexes atomically for zero-downtime deployments.

02 — Document Data Operations

Add, update, or replace large batches of documents, or get the details for a single document by its ID.

03 — Advanced Deletion and Cleanup

Remove data using granular methods, such as deleting all records in an index, removing specific matching groups, or cleaning up old tasks.

04 — Complex Search Queries

Run powerful searches against your content, including multi-query operations and finding documents similar to a starting ID.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to this MCP and provide your Meilisearch Instance URL along with an API Key.

02 Your AI client authenticates and connects the credentials, making the search engine available as a toolset.

03 You prompt your agent (e.g., 'Find all documents in the 'products' index that were modified last week') and it executes the complex data retrieval.

The bottom line is you get to manage mission-critical search infrastructure using natural language prompts through your preferred AI client.

Built For

This MCP is essential for developers and data engineers who spend too much time in web dashboards just to run basic indexing checks. If you're constantly dealing with structured, large-scale content or complex search requirements, this tool saves hours of manual API calls.

Data Engineer

Automating index maintenance, triggering snapshots, and running full document synchronization tasks across multiple indexes.

Software Developer

Integrating search functionality directly into code generation or testing workflows without needing to switch contexts to a separate admin tool.

Content Manager

Quickly verifying if specific content has been correctly indexed and checking the metadata status of an entire collection.

What Changes When You Connect

01 Avoid manual UI clicks. Instead of navigating dashboards to create a new index, your agent runs the `create_index` tool directly when prompted.

02 Eliminate downtime risks. Use the `swap_indexes` function to transition between versions instantly, ensuring zero disruption to live users.

03 Handle massive data cleanup. Don't manually delete records; use `delete_documents_by_filter` to remove large groups of outdated content based on complex criteria.

04 Speed up development cycles. Need to test search results? Use the `multi_search` tool to run several queries in one call, saving API overhead.

05 Maintain data integrity. You can use `get_index_stats` and `get_settings` to verify that your index is configured exactly right before deploying changes.

Real-World Applications

Refreshing a Product Catalog

A content manager needs to update the product search results after adding 500 new items. They ask their agent, 'Add these 500 JSON documents to the 'products' index.' The agent uses `add_documents`, handling the entire batch upload and queueing process instantly.

Debugging Search Failures

A developer finds that search results are missing fields. They prompt their agent, asking it to 'Get the metadata for the 'movies' index.' The agent uses `get_index` and reports back exactly what settings need changing.

Migrating Search Versions

A data engineer finishes a major content update. Instead of manually reconfiguring the live system, they prompt their agent to 'Swap indexes from v2_staging to v2_live.' The agent executes `swap_indexes` atomically.

Finding Related Content

A user searches for a specific article. To guide them better, the agent uses `similar_documents`, finding and listing three related articles that are contextually close to the original search term.

Patterns to Avoid

Deleting data piecemeal

✗ AVOID

Trying to delete old records by running 10 separate commands for 10 different date ranges. This is slow, error-prone, and misses complex filter logic.

✓ INSTEAD

Use `delete_documents_by_filter`. You pass one single instruction—like 'Remove all documents where status=archived AND created_at < last_year'—and it handles the bulk cleanup.

Ignoring index health

✗ AVOID

Assuming that just because a document was added, it is searchable. The system might be running on outdated settings or an old version.

✓ INSTEAD

Always run `get_health` first. If the health check passes, then you proceed with data operations like `add_documents`. It confirms everything is ready.

Overwriting all content

✗ AVOID

Running a basic 'delete all' command without checking if new documents were added since the last run.

✓ INSTEAD

Check first with `list_indexes` and `get_index`. If you confirm no one is using the index, then consider running `delete_all_documents`. Otherwise, stick to targeted updates.

The Right Fit

Use this MCP if your core problem involves managing complex, structured search data that needs constant indexing and manipulation. This tool shines when you need to run bulk operations —like using `add_documents` or `swap_indexes` —without writing custom CLI scripts. Don't use it if all you need is a simple database read (e.g., 'What is the user's name?'). For those cases, a general data retrieval tool is better. If your task involves advanced API management (like managing multiple credentials or keys), then this MCP provides the necessary granular control through tools like `list_keys` and `get_key`. It's built for operations engineers who treat their search engine as mission-critical infrastructure.

Managing a Live Search Engine Used to Be a Pain

If you're running a content site, the old process is a mess of tabs and clicking. You update data in one system, then have to log into your search dashboard, manually checking indexes and running separate commands just to see if everything synced correctly. It takes half an hour just to verify a single deployment.

With this MCP, you skip the UI entirely. Your agent handles the complexity. Instead of clicking through settings menus, you ask for it: 'Check the health and statistics.' You get immediate, actionable status reports that let you move forward without friction.

Meilisearch MCP Gives You Full Control Over Indexing

You don't have to rely on automated cron jobs or complex pipelines just for simple changes. If you need to change how documents are identified, `update_index` handles it. If the whole search structure needs a refresh, `create_snapshot` captures that moment perfectly.

The result is full operational command via chat. You manage the entire lifecycle—from creation (`create_index`) to daily operations and eventual deletion (`delete_index`)—all through simple prompts.

Meilisearch MCP: 44 Tools for Indexing and Search

Use these tools to handle every part of the Meilisearch workflow. From simple document retrieval to complex index swaps, you can manage your search data entirely through your AI client.

#	TOOL	DESCRIPTION
01	add_documents	Adds or replaces one or more documents within a specified index.
02	cancel_tasks	Stops any pending background tasks that haven't finished processing.
03	chat_completion	Requests conversational text generation from a specific workspace.
04	configure_experimental_features	Turns experimental search features on or off for better control.
05	create_dump	Starts the process of creating a full data dump of the index content.
06	create_index	Builds and initializes an entirely new search index from scratch.
07	create_key	Generates a brand new API key for secure access to the system.
08	create_snapshot	Triggers the creation of an immediate, point-in-time backup of the index data.
09	delete_all_documents	Wipes out every single document currently stored in a given index.
10	delete_document	Removes one specific document when you know its unique ID.
11	delete_documents_batch	Deletes multiple documents simultaneously by providing a list of IDs.
12	delete_documents_by_filter	Cleans up data by removing all documents that match complex search criteria.
13	delete_dynamic_search_rule	Removes a specific, customized rule used to enhance search functionality.
14	delete_index	Completely removes an entire index and all its associated data.

#	TOOL	DESCRIPTION
15	delete_key	Deletes a specific API key, revoking its access permissions.
16	delete_tasks	Cleans up records of tasks that have already completed processing.
17	get_batch	Retrieves detailed status information for a specific data batch job.
18	get_document	Fetches the full details of one document using its unique ID.
19	get_health	Checks and reports the overall operational status of the entire Meilisearch instance.
20	get_index_stats	Retrieves detailed performance statistics for one specific index.
21	get_index	Gets all the metadata and configuration details for a specified index.
22	get_key	Retrieves the full details and status of an existing API key.
23	get_settings	Fetches all current configuration settings used by a particular index.
24	get_stats	Returns global statistics, including data size and metrics for every index.
25	get_task	Retrieves the current status and details of a specific background processing task.
26	get_version	Checks which version of the Meilisearch instance is currently running.
27	list_batches	Retrieves a list of all data processing batches that have run or are pending.
28	list_chats	Lists the names and IDs of existing chat workspaces configured in the system.
29	list_documents	Fetches a list of document metadata from an index without retrieving full content.
30	list_dynamic_search_rules	Lists all custom rules that modify how search queries are processed for an index.
31	list_experimental_features	Shows the status (enabled or disabled) of various experimental features available in Meilisearch.

#	TOOL	DESCRIPTION
32	list_indexes	Provides a list of every index currently managed by the system.
33	list_keys	Displays all API keys associated with this Meilisearch instance.
34	list_tasks	Gets a comprehensive list of background tasks that are currently running or waiting to run.
35	multi_search	Executes several distinct search queries in one single, efficient API call.
36	reset_settings	Resets all configuration settings for a given index back to their factory defaults.
37	search_documents	Performs a standard text search across the content of documents within an index.
38	set_dynamic_search_rule	Creates or updates a custom rule to modify how certain data fields are searched.
39	similar_documents	Finds other documents whose content is highly related to the text of a specified document ID.
40	swap_indexes	Swaps multiple indexes at once, ensuring the switch happens instantly and without downtime.
41	update_documents	Makes partial updates to documents, changing only specific fields instead of replacing the whole record.
42	update_index	Changes the primary key used by an index, allowing you to migrate data identifiers safely.
43	update_key	Modifies the name or description of an existing API key for better tracking.
44	update_settings	Changes specific operational settings, such as pagination limits or indexing rules, for a given index.

See It in Action

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

U List all my Meilisearch indexes and their primary keys.

I've retrieved your indexes. You have 3 active indexes: 'movies' (primary key: 'id'), 'books' (primary key: 'isbn'), and 'products' (primary key: 'sku').

U Add these three product documents to the 'products' index: [JSON data].

I've initiated the document addition for the 'products' index. The task has been enqueued with UID 452. You can check the status to confirm when the indexing is complete.

U Get the document with ID 'prod_99' from the 'products' index, but only show the 'name' and 'price' fields.

Here is the requested document from 'products': { "name": "Wireless Headphones", "price": 99.99 }.

Frequently Asked Questions

01 **How do I check if my Meilisearch index is working correctly using Meilisearch MCP?**

Run the `get_health` tool. This immediately checks the overall operational status of your entire search engine instance, telling you if everything is green or if something needs attention.

02 **Can I update my documents without losing the old data? Meilisearch MCP?**

Yes, use `update_documents`. This tool performs partial updates, meaning it only changes specific fields on existing records instead of replacing the whole document.

03 **What’s the difference between deleting documents and listing them with Meilisearch MCP?**

Listing (`list_documents`) just shows you what's there. Deleting involves actually removing data, which can be done individually with `delete_document` or by filtering.

04 **How do I handle a major search version update safely? Meilisearch MCP?**

The safest way is to use `swap_indexes`. This tool allows you to transition between two versions of your index atomically, ensuring zero downtime for users.

05 **Does Meilisearch MCP support running multiple searches at once?**

Yes. Use the `multi_search` tool to combine several different search queries into a single call, which is much faster and more efficient than calling each one separately.

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 `"meilisearch": { "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

Meilisearch 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 Meilisearch. 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	Meilisearch MCP
Server ID	`019e38c0-872f-719f-bbd2-3df4b2c03fcd`
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/meilisearch.