MCP SERVER | NO CODE | CLOUD HOSTED

Sanity MCP

Query, write, and manage your entire content lake via AI.

Sanity MCP gives your AI agent total control over your Content Lake. Execute powerful queries against structured data, manage document records, and handle media assets without ever leaving your chat window. Stop context switching between your CMS interface and an API client; your agent now acts as a native power-user for all Sanity content.

A+ Quality Score 100/100

groq-queries | structured-content | document-management | schema-modeling | content-lake | api-driven

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

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.

Sanity MCP

10 tools available
Cloud-hosted on Vinkius

This MCP connects any compatible AI client directly to your entire Sanity Content Lake. You can manage every aspect of your structured data—from creating brand new records to running deep queries across thousands of documents.

Want to find all posts written by a specific author who mention 'AI'? Your agent handles that with one command, executing complex graph-relational object queries (GROQ) that pull exactly the content you need. Need to update 50 product descriptions simultaneously? You can patch multiple fields on existing documents or even run raw mutations for tricky transactions.

It’s more than just reading data; your agent becomes a full CMS power-user, capable of listing every schema type in your project and managing all uploaded media assets. If you're looking for the best way to connect structured content sources to AI workflows, Vinkius hosts this MCP, giving you one connection point to manage complex document structures.

This means developers can prototype migrations instantly, editors can perform bulk updates without leaving their chat, and product managers get instant reports on published versus draft content.

Core Capabilities

01 — Querying structured data

Fetch specific pieces of content by running powerful queries against your document schema.

02 — Creating and updating records

Build new documents or modify existing fields on any type of record in the CMS.

03 — Managing media files

Browse, list, and handle all images and file assets uploaded to your project.

04 — Auditing content structure

List every unique document type available in the dataset or count how many records exist for a given type.

05 — Running raw mutations

Execute complex, multi-document transactions that require advanced API calls.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to this MCP and input your Sanity Project ID, Dataset name, and API Token.

02 Your AI agent automatically connects the credentials and makes all document types available for querying and modification.

03 Tell your AI client what you need—for example, 'List all unique content types' or 'Update post 123 to say it is published.' — and watch the actions happen.

The bottom line is that once connected, your agent treats your CMS like a built-in database, letting you talk to its data instead of writing API calls.

Built For

This is for the developer who hates context switching between their IDE and the web UI. It's for the editor who needs to bulk update metadata across hundreds of posts fast. If your job involves reading, writing, or reporting on structured content in a headless CMS, this MCP saves you hours.

Content Editor

Needs to perform large-scale updates, like changing the taxonomy or updating metadata across dozens of draft posts without exporting data to Excel.

Software Developer

Uses the MCP to prototype complex GROQ queries or run quick schema audits before writing production integration code.

Product Manager

Needs real-time content statistics, such as counting published versus draft documents for a specific content type, without needing dedicated analytics access.

What Changes When You Connect

01 You eliminate context switching. Instead of jumping between the CMS UI and an API playground to test queries, you just ask your agent what you need in plain English. The agent handles the complex GROQ execution for you.

02 Content updates get faster. Use tools like patch_cms_document to update specific fields on existing records—like changing a status or updating a date—without having to recreate the entire document payload every time.

03 Schema discovery is instant. Never wonder what content types exist again; just ask your agent to list unique schema types, and it gives you an immediate inventory of everything in your Content Lake.

04 Media management simplifies. You can browse all images and file assets using list_media_assets right from the chat, treating media files like data points rather than just visual clutter.

05 Data integrity is maintained. For complex tasks that involve multiple documents (like creating a new post and simultaneously updating its author's profile), you use run_raw_mutation to keep everything consistent.

Real-World Applications

Finding all outdated content

A Product Manager needs to find every product listing that hasn't been updated in six months. They prompt their agent, and it executes a run_groq_query targeting the 'product' type and filtering by the last update date. The result is a clean list of IDs ready for review.

Mass publishing content

A Content Editor has 20 articles that are finished but still marked as drafts. They ask the agent to patch_cms_document for all 20 records, setting the 'isPublished' field to true in one go, bypassing manual UI clicks.

Auditing content structure changes

A Developer needs to know if a new schema type was added since yesterday. They ask the agent to list_unique_schema_types and compare that output against their local records, instantly verifying the CMS structure.

Creating related articles automatically

A user wants to create a new case study document based on an old one. They ask the agent to get_document_details for the original post's data and then use those fields to run a create_cms_document, saving the structured copy.

Patterns to Avoid

Treating content like files

✗ AVOID

A user tries to paste a whole JSON blob into the chat and expects it to update 50 records at once. This won't work because simple text input doesn't contain the necessary structured instructions for bulk changes.

✓ INSTEAD

For mass updates, you must use patch_cms_document or run_raw_mutation. For instance, if you need to change a field on all documents of type 'product', start by running count_entity_nodes to confirm how many records are affected.

Over-relying on the UI

✗ AVOID

The user manually navigates through 10 different tabs and forms in the CMS interface just to retrieve a list of unique types.

✓ INSTEAD

Instead, prompt your agent directly: 'list all unique schema types.' This uses list_unique_schema_types and gives you an immediate, clean list without any clicking or navigating.

Assuming data is simple

✗ AVOID

A user tries to write a query but forgets that the author's name needs to be linked across two different document types. The resulting query fails because it doesn't account for references.

✓ INSTEAD

You must use run_groq_query and structure your request using specific field joins, like `*[_type == 'post'] | connectedTo('author')`. This ensures the agent writes a complex, interconnected query.

The Right Fit

Use this MCP if your goal is to automate data access or write content logic. You need an intermediary layer that can understand natural language commands and translate them into precise, structured CMS operations (like querying relationships between

documents). Don't use it if you just want to read a single page of content; the native CMS UI works fine for reading. Also, don't use this if your primary task is basic file hosting—use dedicated asset management tools for that. However, if your workflow requires 'read data -> process logic -> write new data,' then this MCP is essential because it allows your agent to perform all those steps sequentially using the available tools like run_groq_query and create_cms_document.

The Pain of Copying Content Data

Today, if you need data from a content lake—say, finding all product descriptions that reference an old policy document—you're stuck. You have to jump into the CMS UI, manually filter by dates, click through category dropdowns, and then copy-paste chunks of text or IDs into a spreadsheet just to start your analysis.

With this MCP, you simply ask your agent: 'Show me all product listings that link to policy document 456.' The agent executes the necessary query and gives you a clean, structured output directly in the chat. You get immediate answers without leaving your workflow.

Using Sanity's Structured Data for Automation

Manual content changes are slow and risky. To update 50 items, you might have to open 50 individual records, find the right field, manually change the status flag, and hit save—a process that takes hours and invites human error.

Now, you can tell your agent to patch_cms_document for all those items in one go. The command executes the necessary updates across dozens of records instantly. It's reliable automation.

Sanity MCP – 10 Tools Available

Use these specialized tools to automate nearly every task in your Sanity Content Lake, from running complex queries to managing documents and files.

#	TOOL	DESCRIPTION
01	`list_typed_documents`	Lists all available document types within your Sanity project.
02	`count_entity_nodes`	Counts the total number of documents belonging to a specific type.
03	`list_unique_schema_types`	Retrieves a full list of every unique document structure defined in your dataset.
04	`create_cms_document`	Builds and saves a brand new content document by specifying its type and attributes.
05	`get_document_details`	Pulls all the specific data fields for one single document using its unique ID.
06	`list_media_assets`	Displays a list of every image and file asset that has been uploaded to your project.
07	`patch_cms_document`	Updates specific fields on an existing document without affecting any other data points.
08	`run_groq_query`	Executes powerful, complex graph-relational object queries to fetch precise content subsets.
09	`run_raw_mutation`	Runs advanced, raw mutations for highly complex, multi-document data transactions.
10	`wipe_cms_document`	Permanently deletes a document from the CMS; this action is irreversible.

See It in Action

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

U List all unique schema types in my dataset.

I checked your schema types. Your dataset contains the following 5 types: `post`, `author`, `category`, `product`, and `siteSettings`. Would you like to know how many documents exist for a specific type?

U Write a GROQ query to find the 3 most recently updated authors.

I executed the GROQ query `*[_type == 'author'] | order(_updatedAt desc)[0...3]`. The most recently updated authors are: 1. Sarah Chen, 2. Marcus Rivera, 3. Alex Nguyen. Should I retrieve their full biographical details?

U Update the post with ID 'drafts.1234' and set its published state to true.

I've successfully patched document 'drafts.1234'. The field `isPublished` is now set to `true`. The change is immediately live in your Content Lake.

Frequently Asked Questions

01 **How does Sanity MCP handle complex queries?**

It handles them by allowing your agent to execute GROQ (Graph-Relational Object Queries). This means you can ask questions that require linking data across multiple different content types, not just simple field searches.

02 **Can Sanity MCP delete content permanently?**

Yes. It provides the wipe_cms_document tool to permanently remove records from your CMS. Be warned: this action is irreversible and should only be used when you are absolutely sure.

03 **Does the Sanity MCP help with media assets?**

Absolutely. You can list all image and file assets using the list_media_assets tool, letting your agent manage or reference these files directly within your content workflows.

04 **Is this better than just using the Sanity API directly?**

It's much easier. You don't need to write boilerplate API calls; you just talk naturally to your agent, and it translates those instructions into the correct tool commands for you.

05 **What if I want to create a new document with specific attributes?**

You use the create_cms_document tool. You specify the desired schema type and provide all the necessary data points in a simple, structured format that the agent handles.

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 `"sanity": { "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

Sanity 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 Sanity. 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	Sanity MCP
Server ID	`019d7603-0e9e-7390-8bd7-2b34d5edcc2e`
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/sanity.