MCP SERVER | NO CODE | CLOUD HOSTED

Firestore Collection MCP

Give your agent a contained memory bank for structured data.

Google Firestore Collection MCP gives your AI agent a secure, dedicated NoSQL database for structured data storage. It lets your client perform precise operations like querying, creating new records, and updating existing documents within one specific Google Firestore collection. This is perfect for giving agents a safe place to track chat histories or process project states without touching critical cloud databases.

F Quality Score 3.6/100

nosql | document-database | data-persistence | structured-data | cloud-database

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/google-firestore-collection

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.

Google Firestore Collection MCP

3 tools available
Cloud-hosted on Vinkius

Your AI agent suddenly gets a dedicated memory bank. Instead of forcing it to store complex data in messy text blocks, this MCP gives your client the ability to manage structured information directly inside one Firestore collection. Think of it as a single, protected filing cabinet for all your project's temporary or persistent data.

It strips away dangerous global database permissions, giving your agent only surgical access to that specific spot. Your AI can safely read documents, write new ones, and modify fields in place—all without the risk of damaging other parts of your cloud setup. This controlled environment is huge for building reliable agents. When you connect this MCP via Vinkius, you get instant, contained database power for anything from storing chat threads to running complex workflow results.

It's a simple, scalable NoSQL connection that lets your agent behave like it has its own internal memory and state machine.

Core Capabilities

01 — Read data records

Your AI client reads an entire document or specific fields from the configured Firestore collection.

02 — Write new documents

The agent creates a brand-new record in the collection, assigning it all necessary structured information.

03 — Update existing records

Your AI client modifies specific fields within an already existing document without affecting other data points.

04 — Remove old documents

The agent deletes a targeted record from the collection when it's no longer needed.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/google-firestore-collection** — connect your AI agent in three steps.

01 You connect your AI client to this MCP via Vinkius, designating which specific Firestore collection the agent can interact with.

02 Your agent decides it needs data—maybe it needs a user's last five interactions or a project's current status. It calls the relevant tool (like getting or setting a document).

03 The MCP executes that call securely against the designated single collection, returning the structured data back to your AI client for immediate use.

The bottom line is you give your agent one controlled, secure endpoint where it can read, write, and modify specific records without any risk of collateral damage across your main databases.

Built For

This MCP is for the application architect or backend engineer who needs their AI agents to maintain persistent state. If you're tired of having to manually feed context data back into the prompt every time a user interacts, this connector gives your agent memory.

AI Solutions Architect

They use this MCP to build reliable prototypes by giving their agents a safe, dedicated place to store conversation history and application state.

Backend Engineer

They integrate it into existing Python or JavaScript services to allow AI workflows to persist results—like scoring models or user profiles—in a structured manner.

Data Scientist

They use it when an agent needs to process and save the output of a complex calculation, like storing model weights or simulation parameters for later review.

What Changes When You Connect

01 Safe Data Storage: The system locks the agent to one collection. It can’t accidentally query or mess with other, more critical production databases.

02 Full CRUD Operations: You gain full read, write, update, and delete capabilities (CRUD) using tools like `get_document`, ensuring your agent can manage data lifecycle completely.

03 Context Persistence: Agents can reliably save complex workflow results—like a long-form report's score or status—using `set_document` so the next user session starts with accurate context.

04 Structured Memory: Instead of dumping everything into one giant prompt, you store structured memories in Firestore. This keeps the data clean and easily searchable via `get_document`.

05 Clean Cleanup: When a document or chat history is finished, use `delete_document` to remove it completely, preventing unnecessary bloat and keeping your collection tidy.

Real-World Applications

Building multi-step forms

A user fills out a complex application over several days. Instead of losing the progress, the agent uses `set_document` to save the current draft state every time the user leaves a page, allowing for seamless pick-up later.

Tracking customer service chats

The agent handles a conversation. After it's resolved, it uses `set_document` to write a summary and resolution code into the collection, making that record immediately available for future support agents to review.

Managing project task boards

A team leader asks the agent to check the status of 'Task Alpha'. The agent uses `get_document` to pull the latest state from the document, confirming it's moved from 'In Review' to 'Complete'.

Cleaning up temporary data

After a background job runs and generates millions of log entries, the agent uses `delete_document` to purge old or temporary records that are no longer needed for analysis.

Patterns to Avoid

Using global database access

✗ AVOID

Trying to build an agent that can read from *all* your collections. This is a huge security risk, and if the agent gets confused, it could delete production data.

✓ INSTEAD

Use this MCP to limit scope. By using only `get_document` or `set_document`, you guarantee the AI operates solely within one specific collection, protecting everything else.

Over-relying on prompt context

✗ AVOID

Trying to make an agent remember details from 100 turns of conversation just by stuffing it into the prompt. Context windows are limited and expensive.

✓ INSTEAD

Use this MCP's tools, specifically `set_document` and `get_document`, to store those long-term memories in Firestore. The data is persistent outside the chat window.

Treating it like a general API wrapper

✗ AVOID

Assuming this MCP can handle relational queries, like 'Find all users who bought X and live near Y.' This is NoSQL, not SQL.

✓ INSTEAD

Focus on single-record operations. Use `get_document` to pull data by ID, or use `set_document` to update a user's specific profile fields.

The Right Fit

Use this MCP if your primary need is structured, persistent state management for an agent—specifically when you need the AI client to read, write, and modify records in one defined spot. Think of it as giving your agent its own secure hard drive. Don't use it if you need complex joins or relational queries across multiple tables; this MCP handles single-document operations (getting, setting, deleting). If you simply need a generic data visualization layer without the ability to write back changes, an analytics dashboard might be better. But if your workflow requires the AI to *change* what is stored, use this MCP.

The Pain of Ephemeral Agent Memory

Today, when a user interacts with an agent over several sessions, all the conversation history and resulting data—the key decisions, the final score, the temporary notes—must be kept in the prompt context. This is clunky; you're limited by token counts, and if the chat gets too long, the core information falls out of scope or costs a fortune to run.

With this MCP, your agent doesn't rely on its short-term memory (the conversation window). Instead, it uses tools like `set_document` to write critical context into Firestore. The data is saved externally, meaning you can retrieve exactly what you need later using `get_document`, no matter how many days pass.

Managing Documents with the Google Firestore Collection MCP

You eliminate the manual step of copying conversation summaries into a separate spreadsheet or manually re-running complex workflows just to save the output. The agent handles all the persistence steps automatically.

It's reliable, contained, and structured. You get true data ownership that keeps your AI workflows stable and scalable over time.

Google Firestore Collection: 3 Tools

These tools let you perform the fundamental operations needed to manage data persistence within a single Google Firestore collection.

#	TOOL	DESCRIPTION
01	`delete_document`	This tool removes an entire document from the specified Firestore collection by its ID.
02	`get_document`	It retrieves all the field data associated with a specific document ID within the configured collection.
03	`set_document`	This tool creates a new document or updates an existing one in the collection, allowing you to write structured JSON data.

See It in Action

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

U Get the document with ID 'task-99'.

I've retrieved document 'task-99'. The fields indicate its status is currently 'in-progress'.

U Save this workflow result to a new document 'result-123': {"status": "done", "score": 95}.

I've successfully written the JSON data to the document 'result-123' in the collection.

U Delete the temporary 'draft-01' document.

The document 'draft-01' has been deleted successfully.

Frequently Asked Questions

01 **Does Google Firestore Collection MCP support complex SQL joins?**

No, it is designed for NoSQL document operations. It manages individual records within one collection using tools like `get_document` and `set_document`, not relational joins.

02 **Is my data secure when I use the Google Firestore Collection MCP?**

Yes, security is paramount. This MCP limits your agent's access to a single collection only, preventing it from touching other sensitive parts of your cloud infrastructure.

03 **How do I delete old chat logs using the Google Firestore Collection MCP?**

You use the `delete_document` tool. You simply provide the unique document ID for the chat session, and the agent removes that entire record from the collection.

04 **Can this MCP store structured JSON data?**

Absolutely. The primary function of the MCP is to allow you to write rich, structured data—like workflow results or user profiles—using `set_document` into the NoSQL format.

05 **What if I need to update only one field in a document?**

You use the `set_document` tool. This allows you to target and modify specific fields within an existing record without overwriting all the other data points.

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 `"google-firestore-collection": { "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

Google Firestore Collection 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 Google Firestore Collection. 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	Google Firestore Collection MCP
Server ID	`019e38a2-a6e4-7309-8f2a-0180a5f41841`
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/google-firestore-collection.