MCP SERVER | NO CODE | CLOUD HOSTED

Azure Cosmos DB Container MCP for AI Agents

Managing structured NoSQL data and document storage securely

Azure Cosmos DB Container MCP gives your AI agent secure access to one specific NoSQL database container. It lets you read, write, update, and query structured documents without any risk of accessing other parts of your cloud environment. This is critical for managing chat history, application state, or complex records securely.

F Quality Score 3.6/100

nosql | document-database | data-storage | scoped-access | cloud-database | json-storage

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/azure-cosmos-db-container

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.

Azure Cosmos DB Container MCP

4 tools available
Cloud-hosted on Vinkius

This MCP connects your AI client directly to a single Azure Cosmos DB Container. Think of it like giving your agent a dedicated safe deposit box for all structured data, and nothing else. You get the ability to manage documents—inserting new ones, pulling specific records by ID, updating old entries, or running complex searches across the entire container.

Because access is strictly scoped to this one container, you don't have to worry about your agent accidentally touching critical system databases or listing other resources. It's a surgical level of control that lets developers store everything from user profile data and chat histories to complex application state in a scalable NoSQL format. When you connect it through Vinkius, your AI client gets instant access to reliable document management without needing heavy backend engineering. This means your agent can handle complex data workflows right out of the box.

Core Capabilities

01 — Querying Document Data

Run custom SQL queries against the container to find documents based on specific criteria.

02 — Retrieving Documents by ID

Fetch a single document directly using its unique identifier and partition key.

03 — Creating New Records

Add brand new documents to the container, ensuring they are properly indexed with an ID and Partition Key.

04 — Deleting Documents

Permanently remove a document from the container using its specific ID and partition key.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/azure-cosmos-db-container** — connect your AI agent in three steps.

01 Your AI client connects to Vinkius, selecting this MCP for Azure Cosmos DB Container.

02 The agent uses natural language to specify whether it needs to read data (query), write new data (create), or delete an entry.

03 The system executes the precise database operation against the container and returns the resulting document(s) or a success confirmation.

The bottom line is, you tell your agent what data you need—whether it's a list of records or a single chat history—and it performs the exact action within the secure boundaries of this one database container.

Built For

This MCP is for backend developers, technical architects, and data integration engineers who need to give their AI agents reliable, scoped access to structured records. If you're tired of building complex boilerplate code just to save or retrieve a user record, this connects your agent directly to the source.

Backend Developer

Uses it to allow their agents to manage application state and persistent structured data without exposing the entire cloud environment.

Data Engineer

Connects it when building pipelines that require an AI agent to query or audit specific, isolated sets of records for testing or archival purposes.

DevOps Architect

Implements it to enforce the principle of least privilege, ensuring any integrated tool only has access to one designated data container.

What Changes When You Connect

01 Absolute containment means the agent can only interact with this single container. It won't list or access your other databases, keeping your production environment safe.

02 Need to save chat transcripts? Use `create_document` to reliably store session histories in a scalable NoSQL format for later retrieval by your AI client.

03 Running complex reports used to mean writing dedicated backend microservices. Now, you can use `query_documents` to run advanced SQL queries directly via your agent's natural language prompt.

04 Retrieving a single user profile is simple. The `get_document` tool lets your agent pull specific records instantly using the document ID, making data access fast and reliable.

05 It removes the risk of over-permissioning. Instead of giving global database read/write rights, you grant precise access to just this one container.

Real-World Applications

Archiving old user session data

A support manager needs to review a customer's interaction from six months ago. Instead of digging through logs, the agent uses `query_documents` with date ranges and specific IDs to pull all relevant chat history records in one query.

Updating user preferences after checkout

A marketing automation workflow needs to update a user's preferred communication method. The agent calls `get_document` first to verify the current record, then uses `create_document` or similar logic to write the updated data safely.

Cleaning up outdated records

The ops team runs a script that identifies and archives stale data. The agent executes `query_documents` for documents marked 'expired' and then uses `delete_document` to clean them up permanently, confirming the operation each time.

Building a knowledge base from unstructured inputs

A content editor needs to store newly drafted articles. The agent creates structured records using `create_document`, ensuring every article gets a unique ID and is immediately available for querying by other parts of the system.

Patterns to Avoid

Trying to list all databases

✗ AVOID

An agent tries to run a generic command like 'show me all your data sources' because it doesn't know which database holds what information.

✓ INSTEAD

Don't ask for everything. Use this MCP and tell the agent exactly what you need: 'Query the container for documents where status is pending.' This forces the action to stay within the single, secure scope.

Forgetting required keys

✗ AVOID

Attempting to retrieve a document without providing the necessary Partition Key or ID causes the operation to fail with a vague error message.

✓ INSTEAD

When requesting data, always specify the unique identifier. Use `get_document` and make sure you pass both the ID and any required partition keys for success.

Over-querying without filtering

✗ AVOID

Running a broad query like 'select * from c' that returns millions of records, overwhelming the agent and slowing down performance.

✓ INSTEAD

Always use `query_documents` and include specific WHERE clauses. Filter results by status, date range, or user ID to limit the scope and keep the response actionable.

The Right Fit

Use this MCP if your primary need is secure, isolated storage for structured data—think chat logs, user profiles, or inventory records. It's perfect when you must guarantee that an AI agent can only touch one specific container and nothing else in your cloud setup.

Don't use it if: 1) You need to manage multiple distinct database services (e.g., a separate SQL DB for transactions and Cosmos DB for documents). In that case, you'd need multiple MCPs. 2) Your data is truly unstructured and massive (like raw images or video files). Those require object storage solutions instead. This tool only manages the document metadata itself.

Azure Cosmos DB Container MCP: Managing Structured Data in Enterprise Systems

Today, when an agent needs to access persistent data, developers often have to build complex authorization layers. This involves granting permissions across multiple databases and containers just to read one small piece of information. You spend time managing roles and ensuring the AI client can't accidentally 'scope creep' into production systems.

With this MCP, you give your agent a single, tightly scoped superpower. It acts as if it only has eyes on one specific container. You get reliable document management—whether you're creating new records or running complex queries—without worrying about the security implications of global cloud access.

Azure Cosmos DB Container MCP: Querying Specific Document Records for AI Agents

Manually, retrieving data meant writing specific code paths for every single action: one function to get by ID, another to search by status, and yet another to write updates. This created repetitive, brittle code that was hard to maintain.

Now, the agent handles it all conversationally. You use `query_documents` to ask complex questions in plain English, and the MCP translates that into a robust SQL query against your container. The outcome is immediate, accurate data retrieval with zero boilerplate.

Azure Cosmos DB Container: 4 Operations for NoSQL Data Management

Use these four tools to read, write, update, or delete any document within the designated Azure Cosmos DB container.

#	TOOL	DESCRIPTION
01	`create_document`	Creates a new document record within the Cosmos DB Container.
02	`delete_document`	Permanently removes an existing document from the container.
03	`get_document`	Retrieves a specific, single document by its unique ID.
04	`query_documents`	Runs a full SQL query against the container to find multiple documents matching complex criteria.

See It in Action

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

U Find all user records from Q1 2023 where the status was 'pending review'.

Query Results: Pending Records

ID	USER NAME	STATUS
user_456	Jane Smith	pending review
user_789	Mike Jones	pending review

Action: Found 2 records ready for manual review. You should prioritize the record belonging to Jane Smith.

U I need to save a new chat history session for user ID 'client-101'.

Document Creation Successful

A new document was successfully created in the container. Here are the details:

- **ID:** `chat_20230612`
- **Partition Key:** `client-101`
- **Content Summary:** Initial inquiry regarding billing cycle changes.

This record is now available for future queries.

U What's the current status of document 'report_q2_23'? And please delete it if it's marked as archived.

Status Check: The document `report_q2_23` was found. Status: *archived*.

Action Taken: Because the status matched 'archived', I ran the deletion command. ✅ Document `report_q2_23` has been permanently removed from the container.

Frequently Asked Questions

01 **Can Azure Cosmos DB Container MCP handle chat history storage and retrieval?**

Yes, absolutely. You can use this MCP to treat chat histories as structured documents, creating a dedicated record for each session. This makes it easy for your AI agent to pull up long-term context when needed.

02 **Is the Azure Cosmos DB Container MCP safe regarding permissions?**

It is designed with extreme security in mind. The MCP only grants access to one specific container, meaning your agent cannot interact with or view any other databases you have running in Azure.

03 **How do I use this MCP if I need to search across multiple criteria?**

You must use the query function. Instead of asking for everything, phrase your request like: 'Find all documents where status is pending AND created before last month.' This allows precise filtering.

04 **Does Azure Cosmos DB Container MCP work with other cloud databases?**

No. This MCP is specifically built to manage one single, isolated container within the Azure Cosmos DB ecosystem. It will not connect to external or different database types.

05 **What if I need to update an existing document's data?**

You first retrieve the current document using its ID and then use the appropriate tool function to write the updated version back. This ensures you don't overwrite critical information accidentally.

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 `"azure-cosmos-db-container": { "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

Azure Cosmos DB Container 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 Azure Cosmos DB Container. 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	Azure Cosmos DB Container MCP
Server ID	`019e3869-b80d-7286-a03c-0004c3c6fe7f`
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/azure-cosmos-db-container.