MCP SERVER | NO CODE | CLOUD HOSTED

Upstash Redis MCP

Manage data persistence via conversation.

Upstash Redis lets your AI agent treat a serverless key-value database like an extension of your workflow. You can manage data—reading values, setting expiration times, deleting old keys, and incrementing counters—all by talking to your agent via natural chat or IDE terminal.

A+ Quality Score 98.33/100

key-value-store | caching | serverless-database | data-persistence | database-administration | latency-optimization

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/upstash-redis

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.

Upstash Redis MCP

7 tools available
Cloud-hosted on Vinkius

Stop context switching just to check a cache value. This MCP securely connects your AI client directly to your Upstash Redis database. Instead of jumping into a separate GUI tool, you talk to your agent and ask it to perform data operations against your live store. Your agent gets the technical ability to act like a real database administrator, letting you scan active keys or read raw datastore strings right from your chat window.

Whether you need to inject new values with specific lifespans or safely track operational counters, this connection handles it all. Because Vinkius hosts this MCP, your agent gets immediate access to the full suite of Redis tools. You can ask it to check the Time-To-Live on a caching profile or audit key patterns without writing boilerplate code.

Core Capabilities

01 — Retrieve stored data values

The MCP fetches the exact string configuration of any specified key.

02 — Inject and expire new keys

You can instruct the agent to write a new value, optionally setting how long it remains active before expiring automatically.

03 — Scan for specific data patterns

The MCP scans across your database using pattern matching to find relevant keys and inspect their structures or lifespans.

04 — Manage numeric counters

You can safely increase or decrease numerical keys, perfect for counting rate limits or tracking user sessions in real time.

05 — Check key metadata and TTL

The MCP retrieves the data type and remaining Time-to-Live duration configured for any given key.

One Click on Vinkius — From Prompt to Execution

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

01 First, you authorize the Upstash Redis MCP component in your agent's extensions.

02 Next, you provide the necessary REST URL and REST Token connected to your database from the Upstash console.

03 Finally, you prompt your AI client naturally—for example, 'List all active keys matching session:*'—and your agent executes the required command.

The bottom line is that using natural language, you treat managing your key-value store like another conversation with your agent.

Built For

This MCP is for the backend developer tired of context switching between their IDE and a separate GUI client. It's for the DevOps engineer who needs to check rate limits on live systems instantly, or the system architect auditing key lifecycles as they write code.

Backend Developer

Checks cache hits and clears outdated session states directly through conversation, without launching specialized database clients.

DevOps Engineer

Inspects rate-limiting structures or safely increments manual counter matrices during live site debugging incidents on the fly.

System Architect

Audits temporary key lifespans and tests new eviction behaviors immediately while writing operational code logic for service deployment.

What Changes When You Connect

01 Immediate debugging access: Instead of opening a specialized desktop database client, you can use the `get` tool to verify cache hits or check key status directly through your agent chat.

02 Safe operational counting: Use the `increment` tool to track things like rate limits or user attempts without having to run manual counter scripts in an external terminal.

03 Auditability: The `list_keys` tool lets you quickly scan for keys matching a pattern, helping you audit which sessions or resources are currently active in your system.

04 Clean state management: Need to remove stale data? The `delete` tool allows you to clear specific cache fragments or outdated session states instantly when prompted.

05 Time-to-Live control: You can set new values using the `set` tool and dictate exactly how long they must live, ensuring your system automatically cleans up temporary data.

Real-World Applications

Debugging a broken feature cache

A backend developer suspects cached product data is wrong. Instead of logging into RedisInsight, they ask their agent to run `get_key_info` on the key 'cache:product_header' to verify its type and check the remaining Time-to-Live before assuming it's expired.

Implementing a simple rate limiter

A DevOps engineer needs to enforce a 10x per minute limit. They prompt their agent to use `increment` on a user key, checking that the count doesn't exceed 10 before allowing the request through.

Auditing active sessions

A system architect needs to know which session tokens are still alive. They ask their agent to use `list_keys` with a pattern like 'session:*', immediately giving them visibility into all active users without manual scanning.

Cleaning up expired data

A developer finishes a test run and needs to wipe out temporary mock parameters. They instruct their agent using the `delete` tool, specifying the exact key prefix they want removed from the database.

Patterns to Avoid

Trying complex queries

✗ AVOID

Attempting to ask the agent for a full SQL-like join or filter across multiple data fields, like 'Show me all users whose status is active and who logged in yesterday'.

✓ INSTEAD

This MCP handles key-value operations only. If you need relational joins, use a dedicated database connector instead. Stick to simple lookups using `get`.

Ignoring TTL constraints

✗ AVOID

Using the `set` tool without specifying an expiration time on temporary data, leading to permanent cache bloat and memory issues.

✓ INSTEAD

Always pair the `set` command with a Time-To-Live (TTL) value. This ensures your keys automatically expire when they are no longer needed.

Running broad scans

✗ AVOID

Asking to scan all keys using a pattern like '*' on a massive production database, which can cause performance degradation or timeouts.

✓ INSTEAD

Always narrow your scope. Use the `list_keys` tool with specific patterns (e.g., 'session:*' or 'cache:product*') to target only the data you need.

The Right Fit

Use this MCP if your core problem is managing transient, non-relational state—caching objects, tracking counters, or managing session identifiers. If you can describe the required operation as 'read a value,' 'write a value,' or 'increment a count,' this tool works for you. Don't use it if you need to perform complex JOINs across multiple tables (that requires an SQL-type connector). Also, don't rely on this for primary data storage; treat the data here as volatile cache information only. If your goal is structured reporting or relationship mapping, look into a relational database MCP instead.

Dealing with Redis means switching apps and running commands manually.

Today, if you want to check a cache key's status, you have to stop what you're doing in your main IDE, open up a separate database GUI like RedisInsight, connect credentials, type the command into the CLI, hit enter, and then read the output. It's constant context switching that kills flow.

With this MCP connected through Vinkius, that entire manual process disappears. You keep your focus on your code or chat window, and simply ask your agent to check key metadata or retrieve a value. The result appears instantly, right where you are working.

The Upstash Redis MCP gives you direct control over data state.

Instead of manual commands, your agent now handles the complexity. You can simply prompt it to increment a counter or delete an entire cache structure using tools like `increment` and `delete`. It manages the syntax and execution for you.

You get back time and focus. The database becomes another functional layer in your conversation, allowing development teams to debug complex data flows without ever leaving their primary workspace.

Upstash Redis MCP with 7 Tools

These tools let you read values, delete keys, increment counters, and audit your serverless database state through natural conversation.

#	TOOL	DESCRIPTION
01	`delete`	Removes one or more specified keys entirely from the Redis store.
02	`get`	Retrieves the current string value associated with a single key.
03	`increment`	Adjusts a numeric counter at a specific key by increasing or decreasing its value.
04	`get_key_info`	Provides details on the data type and remaining expiration time of a given key.
05	`list_keys`	Scans for keys that match a specified pattern across your database.
06	`ping`	Verifies that the Redis instance is online and responsive.
07	`set`	Writes a string value to a key, allowing you to optionally set an expiration time in seconds.

See It in Action

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

U List all active Redis keys associated with app sessions.

I invoked the `list_keys` query with the pattern 'session:*'. Found 3 active keys residing in your environment.

U Check the Time-To-Live duration limit configured for the 'cache:product_header' key.

I checked the key info via `get_key_info`. The 'cache:product_header' value is a standard String and currently holds a remaining Time-to-Live format of exactly 3600 seconds.

U Delete all caching strings tagged as 'user_193_avatar' from the database immediately.

Using the `delete` tool, I have entirely removed the 'user_193_avatar' cache key structure from the Upstash datastore instance. Future calls seeking this key limit will return null entries until regenerated.

Frequently Asked Questions

01 **How do I check if the 'Upstash Redis' key is still active?**

Use `get_key_info` to check the status. This tool reports both the data type and the remaining Time-to-Live (TTL) for any specific key, letting you know exactly how long it will last.

02 **Can I use Upstash Redis MCP to count things?**

Yes, that's what `increment` is for. It safely increases or decreases a numeric counter at a specified key, making it perfect for tracking usage metrics or rate limits.

03 **What if I want to clear out old session keys using Upstash Redis MCP?**

You can run `list_keys` with the relevant pattern (e.g., 'session:*') to find them, and then use the `delete` tool to remove the entire set of keys you need gone.

04 **Does Upstash Redis MCP support setting expiration times?**

Yes, when using the `set` tool, you can include an optional Time-To-Live (TTL) value in seconds. This ensures temporary data automatically cleans itself up later.

05 **Is Upstash Redis MCP only for simple key lookups?**

No. While it handles basic key/value operations, you can also use `list_keys` to perform pattern-based scans across large sections of your database, making audits much easier.

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 `"upstash-redis": { "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

Upstash Redis 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 Upstash Redis. 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	Upstash Redis MCP
Server ID	`019d75fd-786b-7187-8098-1f160290aa79`
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/upstash-redis.