MCP SERVER | NO CODE | CLOUD HOSTED

Caddy Server MCP for AI Agents

Automating Web Infrastructure Management and Proxy Troubleshooting

The Caddy Server MCP lets you manage your entire web stack directly through chat. Use it to update complex configurations, check real-time proxy health, and handle SSL certificate chains without touching a terminal or writing boilerplate code.

A+ Quality Score 98.33/100

web-server | reverse-proxy | api-gateway | ssl-certificates | automation

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

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.

Caddy Server MCP

13 tools available
Cloud-hosted on Vinkius

This MCP gives your AI client direct control over the inner workings of a running Caddy reverse proxy instance. Instead of logging into the API dashboard or wrestling with YAML files, you simply tell your agent what needs to happen—update a path, check an upstream endpoint, or rotate a certificate. You can load new server configurations using various formats, making it easy to adapt complex setups on the fly. It also lets you analyze existing structures by converting Caddyfile text into raw JSON for easier programmatic handling. Need performance data? Get Prometheus-style metrics and monitor specific backends' health status instantly. All this functionality is managed through a single connection point in the Vinkius catalog, meaning once your AI agent connects to Vinkius, it has access to all web infrastructure tools, not just this one.

Core Capabilities

01 — Apply Configuration Changes

Load entire server configurations or make specific adjustments by appending new rules or replacing existing values within the Caddy setup.

02 — Analyze and Adapt Configs

Convert standard Caddyfile text into structured JSON format for review, without actually applying any changes to the live environment.

03 — Check Proxy Status

Get real-time health reports on all configured proxy upstreams and backends, identifying immediate performance bottlenecks or timeouts.

04 — Inspect Security Certificates

View details about the Certificate Authority (CA) and retrieve specific certificate chains for managed domains to ensure security compliance.

05 — Monitor Performance Metrics

Access structured, time-series data metrics that show request traffic volume and general server performance over time.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to this MCP on Vinkius and provide your Caddy Admin API URL (like http://localhost:2019).

02 Connect your AI client, giving it the necessary permissions to execute configuration commands.

03 Use natural conversation to tell your agent exactly what change you need—for instance, 'Check if my primary upstream is timing out,' or 'Update this path's headers.' The MCP handles the rest.

The bottom line is that you manage complex web infrastructure changes and diagnostics entirely through conversational prompts.

Built For

This tool is for senior engineers—SREs, DevOps specialists, and backend developers. You're the person who gets paged at 3 AM because a certificate expired or an upstream cluster failed. You need immediate diagnostic access to web services without jumping between monitoring dashboards and shell sessions.

Site Reliability Engineer (SRE)

Retrieving live metrics, checking PKI status, and ensuring the system can gracefully handle a failed upstream cluster.

DevOps Engineer

Automating configuration updates for staging environments or deploying new routing rules across multiple services quickly.

Backend Web Developer

Testing and adapting Caddyfile syntax during local development before committing changes to the main branch.

What Changes When You Connect

01 Stop manually copying configuration blocks. You can use `load_config` or `replace_config` to update entire server settings, whether they're in JSON or Caddyfile format.

02 Diagnose failures faster by checking the real-time health of your backend services using `get_upstreams`, immediately seeing if a proxy target is timing out or down.

03 Simplify security audits. Instead of logging into multiple dashboards, use `get_pki_ca_certs` and `get_pki_ca` to gather all necessary certificate chain information in one prompt.

04 Eliminate guesswork during development. Use `adapt_config` to convert Caddyfile snippets to JSON for review before you ever hit 'apply'.

05 Monitor performance without writing monitoring queries. Simply call `get_metrics` and get Prometheus-style data on traffic volume and request counts instantly.

Real-World Applications

Troubleshooting a Broken API Endpoint

The agent detects that `/api/v2` is returning 503 errors. You prompt it to check the upstream status, which reveals a timeout on one node in the backend cluster. The agent then retrieves the full metrics data so you can determine if the issue is intermittent or sustained.

Implementing a New Service Route

A new microservice needs routing at `/user-data`. You provide the service's Caddyfile snippet and ask the agent to adapt it. After reviewing the JSON structure, you instruct the agent to load the new configuration, activating the route immediately.

Emergency Certificate Rotation

The primary domain certificate is expiring in three hours. You ask the agent for the current PKI CA details and the necessary certificate chain for validation, allowing you to prepare the renewal process instantly.

Cleaning Up Old Routes

A legacy API path needs to be deprecated. Instead of manually finding the old route in the config, you ask the agent to read the full configuration and then use `delete_config` to remove the specific entry.

Patterns to Avoid

Trying to patch a single line

✗ AVOID

A user tries to manually modify one variable in the live config, risking syntax errors that could take down the whole service.

✓ INSTEAD

First, use `get_config` to pull the relevant section of the config into your agent's context. Then, use `insert_config` or `replace_config` with precise instructions on where and what to modify.

Confusing JSON keys

✗ AVOID

The user modifies a key using an incorrect path name (e.g., typing 'server' when the correct path is 'http_servers').

✓ INSTEAD

Run `get_config_by_id` first to confirm the exact structure and ID of the element you need. This prevents misconfigurations before running any updates.

Ignoring format compatibility

✗ AVOID

The user tries to feed a raw Caddyfile snippet into an API endpoint expecting pure JSON, causing failure.

✓ INSTEAD

Always run the `adapt_config` tool first. This validates the Caddyfile and converts it safely to JSON, giving you confidence in the data structure before applying changes.

The Right Fit

Use this MCP if your workflow requires frequent, complex modifications of web server routing rules or security certificates based on natural language prompts. It excels when diagnostics—like checking upstreams or pulling metrics—are necessary immediately after a change. Don't use it if you only need to read static logs; for that, standard log aggregation tools are better suited. Also, don't rely on this MCP to manage credentials storage; it deals with the *configuration* of the proxy, not the secret keys themselves. If your primary goal is just simple key-value pair updates without structural dependencies, a basic data store might be simpler.

Caddy Server MCP: Automating Web Proxy Configuration Updates

Managing web routing rules manually is a nightmare. You're constantly jumping between your code editor and the administrative dashboard, clicking through tabs just to load a configuration file or check if a single path is broken because of an upstream failure.

With this MCP, you treat the entire Caddy setup as data accessible via conversation. Instead of navigating menus, you ask your agent to `load_config` with your updated settings; the changes are applied instantly and reliably.

Caddy Server MCP: Monitoring Proxy Health and SSL Certificates

Today, checking site reliability means pulling up multiple dashboards: one for metrics, one for upstream status, and another for certificate expiry dates. This process is slow and prone to missing critical alerts.

Now, you can ask your agent to run diagnostics across the board—it checks `get_upstreams` for immediate health issues and uses tools like `get_pki_ca_certs` to confirm security compliance, all in one go. It makes site assurance proactive.

13 Tools in the Caddy Server MCP for Configuration Management

Use these tools to read, write, modify, and diagnose every aspect of your web server's configuration structure through conversational commands.

#	TOOL	DESCRIPTION
01	append_config	Adds or overwrites an element in a specific section of the overall Caddy configuration object.
02	delete_config	Removes a specified piece of data from any named path within the Caddy configuration structure.
03	get_config_by_id	Retrieves a specific, identifiable object or section of the full configuration using its unique ID field.
04	adapt_config	Converts Caddyfile text into valid JSON format, allowing you to review the structure without applying changes.
05	get_config	Exports the current configuration data for a specific path or provides the full server configuration as JSON.
06	get_metrics	Pulls performance and traffic metrics from the server, formatted in Prometheus standard text.
07	get_pki_ca_certs	Returns the full certificate chain details for a particular Certificate Authority (CA).
08	get_pki_ca	Provides information about a specific PKI application CA used by your web service.
09	get_upstreams	Checks and reports the current operational status of all configured proxy upstreams.
10	insert_config	Creates a new object or inserts data into an array at a precisely defined index in the configuration.
11	load_config	Sets or replaces the entire active Caddy server configuration using either native JSON or Caddyfile text.
12	replace_config	Strictly overwrites an existing object or array element in the Caddy configuration without adding new data.
13	stop_server	Shuts down the running Caddy process cleanly and safely, exiting the service.

See It in Action

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

U I need to check the status of my main API cluster upstreams right now.

Checking proxy health...

Status Report: api-cluster

ENDPOINT	STATUS
/v1/users	**Healthy**
/v2/payments	*Timeout (Retry)*

The `/v2/payments` cluster is reporting intermittent timeouts. I recommend checking the full metrics for that upstream to see if it's a load issue or an actual service failure.

U How do I convert this Caddyfile snippet into JSON format?

Here is the adapted structure based on your input:

```json
{
"servers": [
{
"site": "example.com",
"handler": {
"reverse_proxy": "localhost:8080"
}
}
]
}
```

Does this JSON structure look correct for what you need?

U What is the full configuration and can I delete the old /api/v1 path?

I've retrieved the current config. It contains 3 HTTP servers and 5 routes.

Action: To remove `/api/v1`, I used the `delete_config` tool. The old routing rule has been removed from the active configuration, leaving only the newer paths intact.

Frequently Asked Questions

01 **How do I use the Caddy Server MCP to fix a broken reverse proxy link?**

You tell your agent which path is failing. It will run diagnostics, checking `get_upstreams` to pinpoint if the issue is with the target backend or the routing itself. You can then ask it to update the config using tools like `append_config`.

02 **Can I use Caddy Server MCP to manage SSL certificates?**

Yes, absolutely. The MCP handles certificate management for you. You can query details about the CA using `get_pki_ca`, and retrieve specific chains with `get_pki_ca_certs` whenever you need proof of compliance.

03 **What is the easiest way to apply a large configuration change?**

If you have a full, working config file, use `load_config`. You can feed it either pure JSON or the text version (Caddyfile), and the MCP will update your live services instantly.

04 **Does Caddy Server MCP help with performance monitoring?**

It does. Instead of logging into a separate metrics system, you prompt the agent to get Prometheus-style data via `get_metrics`. This gives you immediate visibility into request volume and server load.

05 **Is this MCP only for testing configurations?**

No. While it's great for development (using `adapt_config`), its main power is live, production management. You use it to modify or monitor real-world, actively running web services.

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 `"caddy-server": { "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

Caddy Server 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 Caddy Server. 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	Caddy Server MCP
Server ID	`019e3872-f2a2-7061-9831-5306d1035c99`
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/caddy-server.