MCP SERVER | NO CODE | CLOUD HOSTED

Apache APISIX MCP for AI Agents

Orchestrate Microservices Traffic Management and API Routing

The Apache APISIX MCP lets your AI agent manage complex cloud-native APIs through natural language. You configure routes, update services, and monitor traffic flow without needing to use cURL or navigating a dashboard. It gives you full control over high-performance microservices architecture directly from your chat client.

A+ Quality Score 98.33/100

api-gateway | traffic-management | cloud-native | load-balancing | microservices

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

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.

Apache APISIX MCP

50 tools available
Cloud-hosted on Vinkius

Managing an API gateway usually means jumping between dashboards, running complex CLI commands, and wrestling with massive JSON blocks. This MCP changes that entirely. Instead of learning the intricacies of every configuration parameter, you just tell your AI agent what needs to happen—like 'Route all `/v2/users` traffic through the new load balancer' or 'Increase rate limiting for the mobile client.' It handles the underlying API calls to update everything from routes and services to SSL certificates. You use this MCP via Vinkius, connecting it once to any compatible AI client, giving your agent immediate access to managing your entire microservices infrastructure. The result is that you can orchestrate high-performance traffic management using nothing but plain conversation.

Core Capabilities

01 — Discovering Gateway Configuration

Retrieve metadata dumps for all configured routes, services, upstreams, and plugins to audit the current state of your gateway.

02 — Managing API Traffic Paths

List, create, or delete specific routing rules (Routes) that direct client requests to the correct backend service.

03 — Updating Backend Endpoints

Set up new Upstream configurations for load balancing and managing node health checks across your application cluster.

04 — Controlling User Access and Limits

List, create, or delete API Consumers and Consumer Groups to manage authentication keys and enforce per-user rate limits.

05 — Monitoring System Health

Check the operational health status of specific upstream nodes or entire gateway resources in real time.

One Click on Vinkius — From Prompt to Execution

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

01 First, you subscribe to this MCP and provide your APISIX Admin URL along with an API Admin Key.

02 Next, you interact with the system using natural language prompts in your AI client, asking it to perform infrastructure changes or retrievals (e.g., 'Create a new route for payments').

03 Finally, the agent executes the necessary low-level calls, updating resources like routes and upstreams directly on your gateway.

The bottom line is that you treat complex API Gateway management like a conversation, not like an engineering task requiring dozens of manual clicks or CLI commands.

Built For

This MCP is for DevOps engineers and SRE teams who spend too much time navigating complex dashboards just to check if traffic flows correctly. It's for backend developers who need to test and deploy new API routes without leaving their IDE. If your job involves managing microservices architecture, you need this.

DevOps Engineer

Auditing routing rules or checking the health status of upstreams across multiple clusters from a single chat window.

SRE Team Member

Troubleshooting live traffic flow issues, such as consumer access errors or load balancing failures, using natural language queries.

Backend Developer

Creating and testing new service abstractions or API routes for a feature build without needing to spin up local environment variables or use the dashboard GUI.

What Changes When You Connect

01 Instead of building complex cURL commands, you ask your agent to configure a new route or service. For example, asking the agent to create an endpoint for `/api/v2/orders` handles all the JSON updates automatically.

02 You can instantly audit your infrastructure's health. Running `get_control_healthcheck` lets you know if all upstream nodes are up before deploying any code changes.

03 Manage user access without touching rate limits or keys. Using tools like `list_consumers` and `put_consumer` allows you to provision new client credentials in minutes via conversation.

04 Update complex configurations for services or plugins using natural language prompts, bypassing the need to manually edit raw YAML or JSON files.

05 The MCP lets you differentiate between L7 (HTTP) routes and basic network layer (L4) traffic flow by managing both `list_routes` and `list_stream_routes` seamlessly.

Real-World Applications

Hotfix Routing for a Broken Service

A payment service endpoint fails unexpectedly. Instead of manually logging into the dashboard, you ask your agent to list all upstreams, identify the failing node, and then trigger `put_upstream` to redirect traffic to a backup cluster immediately.

Onboarding a New Client Application

A new mobile app requires specific rate limits. You ask your agent to list consumer groups, verify the right group exists, and then use `put_consumer` to create the account and assign appropriate access.

Auditing Gateway Complexity Before Migration

Before migrating microservices, you need a full map. You prompt your agent to run `dump_control_routes`, `dump_control_services`, and `dump_control_upstreams` to get a comprehensive, structured data dump for documentation.

Scaling Out a Core API Endpoint

The user service is hitting capacity. You ask your agent to update the upstreams by running `put_upstream`, adding several new node IPs and adjusting the load balancing weight, all in one go.

Patterns to Avoid

Trying to manage configs via CLI

✗ AVOID

Running dozens of cURL commands for every minor change (e.g., `curl -X PUT ...` followed by a massive JSON payload). This is slow and error-prone.

✓ INSTEAD

Use your AI client to talk directly to the MCP. Instead of typing out the whole payload, you just tell it: 'Update the rate limit for Consumer X to 100 requests per minute.' The agent handles the complex API call.

Mixing up L7 and L4 traffic rules

✗ AVOID

Attempting to manage basic network flow (L4) using HTTP route definitions, leading to misrouted packets or service unavailability.

✓ INSTEAD

Recognize the difference. Use dedicated commands like `list_stream_routes` and `put_stream_route` when you need raw packet handling, not just standard web requests.

Ignoring system health checks

✗ AVOID

Deploying a new route without verifying that all backend nodes are actually online first. This results in immediate 503 errors.

✓ INSTEAD

Always start by running `get_control_healthcheck` or `list_upstreams` to confirm every single target node is reporting as healthy before you deploy anything.

The Right Fit

Use this MCP if your job requires managing a complex, multi-layered API Gateway and your primary bottleneck is the manual overhead of configuration. If you routinely copy JSON payloads or run long sequences of CLI commands to provision routes, services, or consumers, this is for you. Don't use it if you only need basic logging; that requires a different monitoring tool. Also, don't rely on it for writing application business logic—it manages the *plumbing*. If your goal is simply reading configuration data, you might just need to list resources like `list_routes` or `dump_control_services`. But if you need to change them, this MCP provides the necessary control.

APISIX MCP for AI Agents: Simplifying Microservices Traffic Management

Today, updating a single service endpoint requires navigating multiple admin panels. You might have to copy an Upstream's IP range into a new Route definition, manually adjusting weights and checking the consumer key in a separate tab. It’s a tedious cycle of clicking, copying JSON, and hoping you didn't miss a comma.

With this MCP, you simply tell your agent: 'Update the upstream for the payment service to include 10.0.0.5 on port 80.' The agent handles finding the correct resource ID, building the payload, and sending the update request. You get immediate, reliable configuration changes without ever touching a dashboard.

APISIX MCP for AI Agents: Mastering API Consumer Control

Before this tool, managing client access was a headache of spreadsheets and manual key generation. You'd have to check if the user belonged to the right group, verify their current rate limit, and then manually issue a new credential set.

Now, you just ask your agent: 'Create a consumer for Acme Corp with limited access.' The MCP handles listing groups, creating the consumer record, setting initial limits, and ensuring everything links up correctly. Access control becomes conversational.

Apache APISIX: 29 Tools for Microservices Traffic Management

Use these tools to list, create, delete, or retrieve every component of your API Gateway, from routes and services to consumer keys and upstreams.

#	TOOL	DESCRIPTION
01	dump_control_discovery	Retrieves a memory dump of all discovered service endpoints configured in the gateway.
02	dump_control_plugin_metadatas	Outputs metadata for every plugin currently registered and used by the Control API.
03	dump_control_routes	Provides a full list of all existing, configured Routes within the control plane.
04	dump_control_services	Outputs details for every Service entity registered across your microservice mesh.
05	dump_control_upstreams	Retrieves a dump of all Upstream configurations, showing where traffic is currently being directed.
06	get_consumer_group	Fetches the details for one specific Consumer Group using its unique ID.
07	get_consumer	Looks up and returns a specific API Consumer record based on their username.
08	get_control_healthcheck	Checks the overall health status of all connected upstream nodes for immediate failure detection.
09	put_upstream	Creates a brand new Upstream configuration or modifies an existing one with updated node details.
10	reload_control_plugins	Triggers a hot reload of all configured plugins, ensuring changes take effect without downtime.
11	trigger_control_gc	Forces a full garbage collection cycle in the underlying HTTP Lua VM to free up memory.
12	get_control_resource_healthcheck	Checks and reports on the specific health status of any named gateway resource you point it to.
13	get_control_schema	Retrieves the JSON schemas for all available resources and plugins, useful for validation.

#	TOOL	DESCRIPTION
14	get_global_rule	Fetches the full details of a specific Global Rule by its unique identifier.
15	get_plugin_config	Retrieves all configuration settings for a specified plugin using its ID.
16	get_proto	Fetches the details of a specific Protocol Buffer (Proto) definition by ID.
17	get_route	Retrieves all configuration details for a single Route using its unique identifier.
18	get_service	Fetches the complete definition of an API Service, including plugins and upstreams.
19	get_ssl	Retrieves the details for a specific SSL certificate used by the gateway.
20	get_stream_route	Fetches the configuration of a Stream Route, which handles Layer 4 (L4) traffic routing.
21	get_upstream	Retrieves all information about a specific Upstream cluster using its unique ID.
22	list_consumer_groups	Lists every Consumer Group currently defined in your gateway's configuration.
23	list_consumers	Retrieves a list of all API Consumers, showing who has been granted access.
24	list_global_rules	Lists every Global Rule in the gateway, allowing you to audit cross-cutting policies.
25	list_plugin_configs	Provides a list of all Plugin Configurations currently deployed and active.
26	list_protos	Lists every Protocol Buffer (Proto) definition available in the system.
27	list_routes	Retrieves a list of all active Routes configured on the API Gateway.
28	list_services	Lists every Service entity, providing an overview of your microservice endpoints.
29	list_ssls	Provides a list of all SSL certificates currently managed by the gateway.

#	TOOL	DESCRIPTION
30	list_stream_routes	Lists all Stream Routes, which control basic network layer (L4) traffic routing.
31	list_upstreams	Retrieves a list of every Upstream cluster defined, showing potential backend targets.
32	put_consumer_group	Create or update an APISIX Consumer Group
33	put_consumer	Create or update an APISIX Consumer
34	put_global_rule	Create or update an APISIX Global Rule
35	put_plugin_config	Create or update an APISIX Plugin Config
36	put_proto	Create or update an APISIX Proto
37	put_route	Create or update an APISIX Route
38	put_service	Create or update an APISIX Service
39	put_ssl	Create or update an APISIX SSL certificate
40	put_stream_route	Create or update an APISIX Stream Route
41	delete_consumer_group	Delete an APISIX Consumer Group
42	delete_consumer	Delete an APISIX Consumer
43	delete_global_rule	Delete an APISIX Global Rule
44	delete_plugin_config	Delete an APISIX Plugin Config
45	delete_proto	Delete an APISIX Proto
46	delete_route	Delete an APISIX Route
47	delete_service	Delete an APISIX Service
48	delete_ssl	Delete an APISIX SSL certificate
49	delete_stream_route	Delete an APISIX Stream Route
50	delete_upstream	Delete an APISIX Upstream

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 API routes that point to the user authentication service.

Found 3 Routes for Authentication:

- `api/auth/v1` : Active, uses SSL cert ID `abc-123`. Matches header `X-Auth-Key`.
- `internal/auth` : Active, reserved for internal IPs. Uses rate limit of 500/min.
- `webhooks/user_event` : Active, requires Consumer Group `WEBHOOKS`. Last updated 2 hours ago.

U What is the current health status of all back-end nodes for the inventory API?

Gateway Health Check Status:

Inventory Upstream: *Healthy. Nodes:* `10.0.1.2:8080` *(Weight 3),* `10.0.1.3:8080` *(Weight 3).* Status: OK.*
Payment Upstream: *Warning. Node:* `10.0.2.1:8080`. Status: Degraded*. Last check failed due to timeout.

U I need to increase the rate limit for the 'mobile-client' consumer group.

Consumer Group Rate Limit Update:

The `mobile-client` Consumer Group was successfully updated. The new limits are:

- **Rate Limit:** 250 requests per minute (increased from 100/min).
- **Burst Limit:** 50 requests.
- **Active Consumers:** 3 (Includes `user-mobile-abc`, etc.).

Please remember to redeploy the gateway for this change to take effect.

Frequently Asked Questions

01 **How can I use the Apache APISIX MCP for AI Agents to manage my API routes?**

You manage your API routes conversationally. Instead of running complex commands, you tell your agent what path needs routing—like 'Send all `/billing` traffic to Cluster B.' The agent handles creating or updating the necessary route definitions automatically.

02 **Does this MCP help with microservices load balancing?**

Yes. You can manage upstreams and load balancing rules directly through the MCP. You tell your agent which nodes need to be added or removed from a cluster, and it updates the upstream configuration for you.

03 **What if I need to check if my gateway is running correctly?**

You can use this MCP to run health checks. You simply ask your agent to check the status of any specific resource or all upstreams, giving you an immediate report on which parts are failing.

04 **Can I change API consumer access rules using the Apache APISIX MCP for AI Agents?**

Absolutely. The MCP lets your agent manage user authentication by listing and creating consumers or even entire consumer groups, allowing you to define who can talk to which services.

05 **Is this better than using the command line interface (CLI)?**

It's far less cumbersome. The CLI requires perfect syntax and remembering resource IDs. With the MCP, your AI agent understands intent—you just tell it what you want done, regardless of how many parameters are involved.

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 `"apache-apisix": { "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

Apache APISIX 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 Apache APISIX. 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	Apache APISIX MCP
Server ID	`019e3865-555e-7153-afc7-56e7f5990ab6`
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/apache-apisix.