MCP SERVER | NO CODE | CLOUD HOSTED

Fly.io MCP

Control edge compute and data state from chat.

Fly.io MCP lets your agent manage complex edge infrastructure and container orchestration through natural conversation. You can list apps, monitor machines globally, scale compute capacity dynamically, run remote commands inside containers, and handle persistent data volumes—all without touching the command line.

A+ Quality Score 100/100

edge-computing | container-orchestration | docker | serverless | infrastructure-as-code

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/flyio

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.

Fly.io MCP

10 tools available
Cloud-hosted on Vinkius

This connector connects your account to Fly.io, giving you full control over distributed edge computing resources via any MCP-compatible client. Forget logging into separate dashboards or writing complex CLI scripts just to check machine health across global regions. Instead, talk to your agent and get real-time status updates on everything from active container assignments to persistent storage volumes.

You can ask the system to list all app spaces belonging to an organization or identify specific dedicated IPv4/IPv6 IPs assigned by the cluster master. Need more compute power? Your agent provisions new, highly available Edge Machines to scale capacity instantly. If something goes wrong, you don't have to SSH in; your agent runs shell commands directly inside active machines through the hypervisor API. This means you can diagnose issues or run database migrations without manual intervention. All of this infrastructure control is managed seamlessly through Vinkius, making it one place for all your edge resources.

Core Capabilities

01 — Listing Infrastructure Components

You can list the app spaces belonging to an organization or retrieve a list of individual microVM endpoints and their specific physical placement regions.

02 — Managing Machine Lifecycle

The system allows you to start, stop, scale up, or terminate machines dynamically based on current operational needs.

03 — Auditing Runtime Status and IP Ranges

You can get the exhaustive runtime state of a machine, including its running image digest, or retrieve the operational baseline, such as anycast IPs and Wireguard network ranges for an app.

04 — Executing Remote Commands

Your agent injects and runs standard shell commands inside active machines, bypassing the need for manual SSH connections.

05 — Controlling Persistent Storage

You can list hardware NVMe Volumes attached to an app, ensuring stateful data like PostgreSQL or SQLite remains available even if compute instances fail.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to this MCP and provide your Fly.io API Token.

02 Connect your agent from any MCP-compatible client, giving it access to the Fly.io infrastructure layer.

03 Use natural language commands to perform actions like listing apps or running diagnostics on machines.

The bottom line is that you manage complex global edge computing resources using plain conversation instead of platform CLIs.

Built For

This MCP is essential for the Ops Engineer who gets tired of clicking through multiple dashboards just to check machine health across three different regions. It's built for people whose job is keeping critical, globally distributed services running 24/7.

Site Reliability Engineer (SRE)

Uses the MCP to monitor system execution logs and run diagnostic commands on active machines in real time, diagnosing failures without manual CLI interactions.

DevOps Engineer

Manages machine health by listing all app spaces or provisioning new highly available Edge Machines when capacity starts running low.

Backend Developer

Triggers restarts, runs database migrations, and verifies deployment digests directly from a chat interface rather than jumping into the terminal.

What Changes When You Connect

01 You manage the entire lifecycle of your infrastructure—from listing apps using `list_apps` to ensuring machines are running via `start_machine`. This eliminates context switching between multiple cloud tools.

02 Never manually SSH into a machine again. Use `exec_machine` to run diagnostic commands like `ps aux` directly from your agent, giving you immediate shell output without complex terminal setup.

03 Data persistence is guaranteed. By using `list_volumes`, you can audit attached hardware NVMe Volumes and know exactly where critical stateful data lives, independent of any single machine's uptime.

04 Scaling compute capacity becomes conversational. Instead of following a multi-step deployment process, your agent provisions new Edge Machines instantly with the `create_machine` tool.

05 You maintain visibility into global networking. The system can provide the operational baseline using `get_app`, identifying active Anycast IPs and Wireguard ranges in one query.

Real-World Applications

Debugging a service outage in a remote region

A backend developer notices high latency. They ask their agent to run `get_machine` on the primary web-api machine. The agent returns that the status is 'suspended.' The developer then instructs the agent to use `start_machine`, restoring the service and confirming connectivity via a follow-up command.

Verifying data integrity before deployment

A DevOps engineer needs to migrate user data. They first call `list_volumes` for the app, confirm the existence of the necessary NVMe storage. Then, they use `exec_machine` to run a specific database migration script inside the target machine.

Scaling up during peak traffic hours

The SRE team sees CPU spikes approaching limits. Instead of waiting for automated scaling policies, they instruct their agent to use `create_machine`, provisioning a new Edge Machine instantly in the busiest region and balancing load across the cluster.

Auditing network topology changes

A cloud architect needs to confirm which IPs are active for compliance. They run `get_app` on the service, receiving an immediate report detailing all assigned Anycast IP ranges and internal Wireguard assignments.

Patterns to Avoid

Mistaking local state for persistent data

✗ AVOID

Assuming that running a temporary script via `exec_machine` will save the results or changes, leading to data loss when the machine restarts.

✓ INSTEAD

Always verify statefulness by using `list_volumes` first. If you need data to survive reboots, confirm it's saved on an attached NVMe volume rather than just in the ephemeral container layer.

Manually listing everything for a full audit

✗ AVOID

Running separate commands for every machine and then having to manually compile a list of their statuses (started, stopped, suspended) into a spreadsheet.

✓ INSTEAD

Use `list_machines` followed by `get_machine` on the specific endpoints. This provides an immediate, compiled view of all execution states in one chat response.

Over-relying on single machine status

✗ AVOID

Assuming that if a primary machine is 'started,' the entire application is healthy and available to users.

✓ INSTEAD

Check `get_app` first. This tool provides the holistic operational baseline, confirming not only the machines but also the global Anycast IP routing assignments for the entire service.

The Right Fit

Use this MCP when your primary pain point is controlling infrastructure that lives across multiple physical locations (edge computing) and requires constant state management. You need to interact with a complex, containerized deployment layer without logging into a dedicated console or running boilerplate CLI scripts.

Don't use it if you just need simple messaging capabilities, file storage access outside of an app volume, or basic CRUD operations on user records; those are better handled by general API connectors. If your goal is simply to list all available apps and get a quick overview, `list_apps` is the starting point. But if you're doing more—like scaling capacity (`create_machine`) or running diagnostics (`exec_machine`)—this MCP gives you the necessary depth.

The manual toil of global infrastructure monitoring.

Right now, checking on your services means jumping through hoops. You open the cloud console for App A to check its Anycast IPs, then switch tabs to run a `kubectl get pods` command on the cluster master, and finally log into an SSH client just to run `docker logs` on one specific machine. It's slow, it's error-prone, and you lose track of whether you checked every single region.

With this MCP, you tell your agent what you need: 'What is the status of App A in all regions?' Your agent handles the complexity, synthesizing data from multiple endpoints into a single, conversational answer. You get immediate visibility without ever leaving the chat window.

Fly.io MCP delivers comprehensive control over machine lifecycle.

The need to manually manage machine uptime—whether it's restarting a stopped worker or scaling up during peak load—requires juggling multiple commands (`start_machine` , `create_machine`) and monitoring the outcomes of each one. This is tedious, especially when you're on call.

Now, your agent handles that entire workflow conversationally. You tell it to scale capacity, and it provisions the new machines, confirming their status immediately. It’s instant control over distributed compute.

Fly.io: 10 Infrastructure Tools

These tools give your agent the power to list apps, control machine lifecycles, execute shell commands, and manage persistent storage for complex edge infrastructure.

#	TOOL	DESCRIPTION
01	`create_machine`	This tool provisions a new, highly available Edge Machine within an existing App space.
02	`delete_machine`	It terminates and destroys a specific Fly Machine instance permanently.
03	`exec_machine`	This function injects and runs shell or Bash commands inside an active machine.
04	`get_app`	It retrieves the operational baseline state of a distinct Fly App, including its primary region and Anycast IPs.
05	`get_machine`	This fetches the exhaustive runtime status for a single machine, detailing its current execution state and image digest.
06	`list_apps`	It lists all App spaces belonging to an organization, identifying their container and storage components.
07	`list_machines`	This function retrieves unique identifiers and explicit placement regions for individual machines within an app.
08	`list_volumes`	It lists persistent hardware NVMe Volumes attached to an App, crucial for stateful data management.
09	`start_machine`	This tool boots up a Fly Machine that was previously stopped or suspended.
10	`stop_machine`	It gracefully halts an active, running internal machine instance.

See It in Action

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

U List all machines in my 'web-api' app

Retrieving machines... I found 2 machines in 'web-api': '918572b0' (Region: iad, Status: started) and 'e28654c1' (Region: gru, Status: stopped). Would you like me to start the stopped machine?

U Run 'ls -la /app' on machine '918572b0' in app 'web-api'

Command execution initiated! I've sent the request to machine 918572b0. Result: [Detailed directory listing showing files in /app folder]. No SSH required, I used the Fly Machine API synchronously.

U Show me the persistent volumes for 'web-api'

Retrieving volumes... I found 1 persistent NVMe volume: 'data_vol' (Size: 10GB, Region: iad). It is currently attached to machine 918572b0 and is residency on physical hardware node xyz.

Frequently Asked Questions

01 **How does Fly.io MCP handle data integrity?**

The system helps manage stateful applications by allowing you to list hardware NVMe Volumes using `list_volumes`, which keeps persistent data separate from the compute layer.

02 **Can I run diagnostics on a machine without SSH?**

Yes. You use the `exec_machine` tool, which injects and runs shell commands like `ls -la` directly inside an active machine via the hypervisor API, bypassing traditional SSH methods.

03 **What if I need to scale capacity quickly?**

Use `create_machine`. This tool provisions new highly available Edge Machines dynamically, letting you adjust horizontal scaling without waiting for a full platform deployment cycle.

04 **Does Fly.io MCP show me the network IPs?**

Yes. You can run `get_app` to retrieve the operational baseline, which includes identifying anycast assignments and internal Wireguard ranges assigned to your app.

05 **How do I shut down a machine safely?**

Use `stop_machine`. This tool gracefully halts a running Fly.io internal Machine instance, minimizing potential latency issues during idle cycles.

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 `"flyio": { "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

Fly.io 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 Fly.io. 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	Fly.io MCP
Server ID	`019d759c-a6d2-72d6-a076-39b0c01b1267`
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/flyio.