MCP SERVER | NO CODE | CLOUD HOSTED

Flipt MCP

Manage feature flags and controlled rollouts via chat.

Flipt lets you manage feature flags and controlled rollouts by talking to your AI agent. Use this MCP to list namespaces, check flag statuses, define user segments, and create percentage-based distributions for progressive delivery.

A+ Quality Score 100/100

feature-flags | rollouts | targeting | ab-testing | configuration-management

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

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.

Flipt MCP

9 tools available
Cloud-hosted on Vinkius

Managing features isn't just about flipping a switch; it's about knowing exactly who sees what, and when. Connect your Flipt instance via Vinkius and let your AI agent handle the complex logic of progressive delivery. You can inspect feature flags across different namespaces to see your current setup. Need to target only internal QA users? The MCP lets you retrieve segments and define constraints for precise rollouts. Want to test a new UI with 5% of your user base? It handles creating distributions and rules for canary deployments. If you need to return two completely different values based on logic, create flag variants. This entire process—from auditing environment structures by listing namespaces to setting up complex rollout percentages—happens entirely through natural conversation with your agent.

Core Capabilities

01 — Audit environments and flags

List every available namespace and check the status of all feature flags within a given scope.

02 — Define user subsets for testing

Retrieve existing segments or create new constraints to limit who sees a specific feature, ensuring targeted rollouts.

03 — Control release percentages

Set up rules and distributions to manage controlled releases, like rolling out a feature to 1% before 20%.

04 — Manage flag variations

Create specific variants for flags so they can return different values based on defined evaluation logic.

05 — View system architecture details

List authentication tokens and namespaces to understand the entire environment's structure and access levels.

One Click on Vinkius — From Prompt to Execution

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

01 First, subscribe to this MCP and provide your Flipt URL along with a valid API token.

02 Next, tell your AI client what you need—for example, 'List the rules for the dark mode flag in production.'

03 Finally, the agent executes the tool calls, retrieving structured data about flags, rollouts, or segments that you can then act on conversationally.

The bottom line is that your AI client translates complex feature management commands into simple natural language chat.

Built For

This MCP is essential for DevOps Engineers and Product Managers who get frustrated by manually navigating multiple dashboards just to confirm a release state. It's also perfect for developers who need to test feature logic without leaving their IDE.

DevOps Engineer

You use this MCP to quickly audit rollout configurations and namespace partitions, eliminating the need to jump between your terminal and dashboard tools.

Product Manager

You check the status of feature flags and segments to verify release progress across different user groups without filing a ticket or waiting for engineering access.

Software Developer

You create variants and constraints directly while coding, allowing you to test specific flag logic against defined rules before committing code.

What Changes When You Connect

01 Audit your entire environment instantly. Use list_namespaces to see every partition, then list_flags to audit the status of features across all environments without clicking through a dashboard.

02 Precision targeting saves time. Instead of broad changes, use list_segments and create_constraint to define rules that only apply to specific user subsets, like beta testers or internal staff.

03 Control deployments with confidence. When you need controlled rollouts, the MCP lets you list_rollouts and create_distribution to manage percentage-based releases, ensuring minimal risk when deploying new code.

04 Build complex logic easily. Use create_variant to define multiple possible outcomes for a single flag, allowing your application to behave differently based on user attributes or system rules.

05 Keep track of access. If you're unsure about permissions, use list_tokens and list_namespaces to map out exactly what credentials exist and where they apply.

Real-World Applications

Verifying a controlled rollout for the new checkout flow

A product manager asks their agent: 'Show me the current rules for the checkout flag.' The agent calls list_rules and list_rollouts, confirming that the feature is correctly set to 5% distribution only for users in the QA segment.

Onboarding a new environment partition

A DevOps engineer starts by calling list_namespaces. After seeing 'staging-v3' isn't listed, they ask the agent to check its structure, using list_flags and list_tokens to verify access is ready.

Testing a new UI variant for a specific user group

A developer needs to test a different header layout. They use create_variant to define 'header-v2' and then call create_constraint, ensuring only users with the 'premium' segment can see it.

Debugging unexpected feature behavior

A QA tester reports a bug in production. The agent uses list_rules to trace the flag logic, pinpointing that an incorrectly set distribution is causing some users to bypass the required checks.

Patterns to Avoid

Assuming full visibility

✗ AVOID

A user tries to check a feature flag's status and only sees the name, but doesn't know if it's active or just defined.

✓ INSTEAD

Don't just list flags; always follow up by listing_rollouts for that specific flag. This confirms whether any actual release distribution is currently applied.

Overwriting existing logic

✗ AVOID

A developer manually changes a rollout percentage in the UI, but forgets to update related constraints.

✓ INSTEAD

Instead of manual edits, use create_distribution through your agent. This ensures any changes to feature visibility are logged and tied directly to defined rules.

Missing context on environment state

✗ AVOID

A team thinks a flag is off in 'staging,' but the system still shows it's active because of an old token.

✓ INSTEAD

Always start by listing_namespaces to confirm you are operating in the correct environment, and then list_tokens to verify your agent has access only to necessary credentials.

The Right Fit

Use this MCP if your core problem is controlling *who* sees a feature and *how* it rolls out (e.g., 5% of users, only internal staff). It's built for progressive delivery management, making sure features are toggled safely across complex environments.

Don't use this if you simply need to store a simple configuration value that affects the whole application globally—a standard key-value database might be better. Also, don't use it just to send messages; that requires a dedicated messaging tool. If your goal is merely to audit user profiles, listing_segments helps, but you'll still need another system for actual user data retrieval.

The headache of feature parity checks across environments

Right now, checking if 'dark mode' is enabled for the staging environment requires a mess of manual clicks: hopping into the dashboard, finding the right namespace partition, clicking through multiple tabs to verify flags, and then cross-referencing release rules. You spend minutes just confirming what should be obvious.

With this MCP, you talk to your agent like you're talking to a coworker. You ask about feature status, and it instantly pulls all that data together. It gives you the full picture of flag visibility, distributions, and segments in one conversational reply.

Flipt MCP: Controlled Feature Rollout Management

The biggest time sink is cross-referencing segment definitions with rollout rules. You have to check if the right user group (segment) has been assigned a constraint, and then verify that a distribution exists for that rule.

This MCP automates that detective work. It lets you list_segments and create_constraint together, giving you immediate certainty about who gets access and under what percentage rules.

Flipt: 9 Tools for Release Management

Use these tools to manage the full lifecycle of your features, from listing environments to creating complex user-specific rollout rules.

#	TOOL	DESCRIPTION
01	`create_constraint`	Defines a specific rule or boundary for a user segment.
02	`create_distribution`	Sets the percentage or scope for how widely a feature rollout is applied.
03	`create_variant`	Creates a new, distinct value that a flag can return based on logic.
04	`list_flags`	Retrieves a list of all feature flags within a specified namespace.
05	`list_namespaces`	Lists every available operational partition or environment name in your Flipt instance.
06	`list_rollouts`	Gets the current list of percentage-based rollouts applied to a specific flag.
07	`list_rules`	Retrieves the logic rules that govern how flags are evaluated for a given feature.
08	`list_segments`	Lists existing user groups or segments within your configured namespaces.
09	`list_tokens`	Shows all stored authentication tokens, helping you understand the system's access levels.

See It in Action

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

U List all namespaces in my Flipt instance.

I've retrieved the namespaces. You currently have: 'default' (Global resources) and 'production-env' (Production flags). Which one would you like to explore?

U Show me all the feature flags in the 'default' namespace.

Fetching flags for 'default'... I found 3 flags: 'new-onboarding-flow', 'dark-mode-beta', and 'api-v2-enabled'. Would you like to see the variants for any of these?

U Create a new variant called 'v2-test' for the 'header-experiment' flag in the 'default' namespace.

Variant 'v2-test' has been successfully created for the 'header-experiment' flag. It is now available for use in your rules and distributions.

Frequently Asked Questions

01 **How do I see all the environments available in Flipt using the MCP?**

You call list_namespaces. This command pulls a complete inventory of every namespace your instance manages, helping you know exactly where to look for flags.

02 **Can Flipt help me test feature logic with different values?**

Yes, use create_variant. This allows you to define multiple potential outcomes for a single flag so your application can behave differently based on the rules.

03 **What is the difference between segments and namespaces in Flipt MCP?**

Namespaces define large partitions (like 'staging' or 'production'). Segments are smaller, targeted groups of users *within* a namespace, used for precise rollouts.

04 **How do I increase the rollout percentage safely with Flipt?**

To manage gradual increases, you call create_distribution. This allows you to adjust the specific rule's distribution from 1% up to 50%, controlling risk.

05 **Does Flipt MCP only work for simple on/off switches?**

No. It handles complex logic, allowing you to list_rules and define rules that depend on multiple user attributes or conditions.

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 `"flipt": { "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

Flipt 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 Flipt. 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	Flipt MCP
Server ID	`019e3898-c0b4-7078-81ae-7c0155e2d867`
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/flipt.