MCP SERVER | NO CODE | CLOUD HOSTED

LaunchDarkly MCP

Check flag states across environments instantly.

LaunchDarkly MCP connects your AI client directly to feature flag systems, environments, and experiment data. You manage complex deployments by simply asking questions—checking if a new button is visible only in beta users' accounts or tracking how many people are engaging with the latest UI change.

A+ Quality Score 100/100

feature-flags | deployment-strategy | experimentation | release-management | environment-variables | ci-cd-integration

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

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.

LaunchDarkly MCP

9 tools available
Cloud-hosted on Vinkius

Feature flags complicate releases; they’re what let developers roll out code piece-by-piece instead of flipping everything on at once. This MCP lets you talk to your LaunchDarkly platform and manage all those complex rules conversationally. You can check specific feature flag statuses across multiple environments, inspect which projects are active, or even pull historical audit logs to see who changed what and when. Need to know if the new payment flow is live in Production but only for 10% of users? Your agent handles that query instantly. It's all managed through Vinkius, giving your AI client one place to access crucial deployment intelligence. You simply ask your agent to list available environments or check specific metrics without ever touching a dashboard.

Core Capabilities

01 — Check flag status across multiple environments

The MCP retrieves the current on/off state of any feature flag within specified workspaces, like staging or production.

02 — List and inspect all project configurations

You can pull a list of every LaunchDarkly project associated with your account to understand your deployment scope.

03 — Examine experiment tracking metrics

The system gathers data on specific user engagement metrics, allowing you to evaluate A/B test performance directly from the chat.

04 — Identify all active workspaces

It lists every environment (development, staging, production) connected to your main account for comprehensive oversight.

05 — View historical deployment changes

You can retrieve full audit log entries detailing who made changes and when they happened across the entire platform.

One Click on Vinkius — From Prompt to Execution

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

01 Install this MCP locally and supply your LaunchDarkly API token key.

02 Connect your AI client to the Vinkius catalog using the credentials.

03 Use natural language commands to request specific flag statuses, environment lists, or project metrics.

The bottom line is you stop clicking through dashboards and start asking questions about your feature flags and deployments.

Built For

This MCP is essential for DevOps Engineers who hate manually checking release status, Product Managers needing instant confirmation on A/B test results, or Fullstack Developers whose jobs depend on knowing if a newly pushed flag actually went live.

DevOps Engineer

You use this to monitor release rollouts and check deployment flags without needing to open multiple dashboards.

Product Manager

You confirm A/B test flag statuses immediately, letting you report on user engagement metrics from a single conversation.

Fullstack Developer

You check if your newly pushed feature flag is live in the correct environment before merging code.

What Changes When You Connect

01 Cut out the dashboard hopping. Instead of clicking through multiple tabs to check a feature status, you just ask your agent and get the answer immediately, whether it's in Staging or Production.

02 Audit every change without searching logs for hours. Use list_audit_logs to pull an entire history, letting you see who made which flag modification and exactly when it happened.

03 Evaluate A/B tests right away. By calling list_metrics and get_metric, you stop guessing about user behavior and start basing decisions on real data.

04 Know your boundaries. You can use list_projects to map out every single deployment silo, making sure nothing gets accidentally missed during a release cycle.

05 Get immediate status checks. Whether you need the current state of a feature flag via get_feature_flag or just want to see all connected workspaces using list_environments, it's instant.

Real-World Applications

The Production Flag Check

A Product Manager needs to know if the new checkout UI flag is active for beta users in Production. They ask their agent: 'Check the status of the checkout-beta-ui flag.' The MCP replies with the exact rollout percentage and environment details, saving them a 15-minute manual check.

The Post-Incident Investigation

A DevOps Engineer notices a bug and needs to know when the problematic configuration was deployed. They call list_audit_logs, retrieving a timeline that shows exactly which user changed the setting and what time it happened.

The Environment Map

A Fullstack Developer joins a new project and needs to know all available deployment targets. They ask for 'all connected workspaces,' triggering list_environments, instantly mapping out Staging, QA, and Beta nodes.

Comparing Feature Performance

The team is debating two different onboarding flows (A vs B). Instead of manually pulling graphs, they ask the agent to check 'onboarding-flow-metrics,' getting real-time data via list_metrics.

Patterns to Avoid

Assuming a flag is live

✗ AVOID

A developer sees the code merged and assumes the feature flag must be ON in Production. They waste time manually logging into dashboards, only to find no status indicator.

✓ INSTEAD

Instead of guessing, use get_feature_flag combined with list_environments. Your agent confirms the exact state (ON/OFF) for the specific environment you care about.

Ignoring deployment history

✗ AVOID

A bug pops up and no one remembers who changed the flag last week, so troubleshooting stalls while people hunt through Jira tickets.

✓ INSTEAD

Run list_audit_logs. This shows a clean timeline of every modification, telling you exactly when and by whom the setting was altered.

Focusing on code instead of flags

✗ AVOID

A developer pushes code and assumes it's ready for release. They forget that the flag might be accidentally disabled in the target environment.

✓ INSTEAD

Before deploying, always use get_environment to confirm the readiness status, followed by checking the specific feature flag with get_feature_flag.

The Right Fit

Use this MCP if your team's release process involves managing flags, multiple environments (Dev, Staging, Production), and continuous A/B testing. You need to know *why* a feature is visible or invisible, not just that the code exists. This connector excels at status reporting, auditing changes via list_audit_logs, and retrieving performance data through list_metrics.

Don't use this if your primary pain point is writing code logic or fixing deployment pipelines (you need CI/CD tools for that). If you only ever deal with a single environment and never change flags, then maybe the complexity isn't worth it. But if you operate in any modern product team managing releases, this MCP is critical because it gives your agent a complete view of all projects, environments, and flag statuses.

The manual headache of release coordination

Right now, checking if a new feature is ready for launch feels like an archaeological dig. You jump from the main dashboard to the environment selector, then drill down into project settings, and finally find the specific flag status you need. If you're tracking A/B tests, you have to switch tabs again just to pull up the metric graphs. It’s slow, it requires constant context switching, and one missed click can delay a major rollout.

With this MCP, all that manual checking vanishes. You talk to your agent once—saying, 'What's the status of the checkout flag in Production?'—and you get a single, definitive answer, no clicks required.

LaunchDarkly provides instant visibility into deployment status

You eliminate jumping between project lists and environment selectors. You don't have to copy and paste flag names across three different screens just to confirm if the feature is live in QA, Staging, and Production.

Now you can use your agent to check any combination of flags or environments instantly. It cuts out the manual review process entirely, giving your team immediate confidence in every release.

LaunchDarkly with 9 Tools

Use these tools to list projects, check specific flags, retrieve metrics, and view audit logs for comprehensive deployment oversight.

#	TOOL	DESCRIPTION
01	`get_environment`	Fetches specific operational details about one particular workspace or environment.
02	`get_feature_flag`	Provides deep, detailed information about one single feature flag by its name or ID.
03	`get_metric`	Retrieves detailed data for a specific metric, helping you evaluate test performance.
04	`get_project`	Fetches general configuration details for a specific LaunchDarkly project.
05	`list_audit_logs`	Retrieves an account-wide history of changes, showing who modified settings and when.
06	`list_environments`	Pulls every active environment (like Dev, Staging, or Production) within the current project.
07	`list_feature_flags`	Retrieves a comprehensive list of all feature flags defined in a specific project.
08	`list_metrics`	Gets a list of all available experimentation metrics within the current project scope.
09	`list_projects`	Retrieves a list of all available LaunchDarkly projects attached to your account.

See It in Action

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

U Check if the newly implemented dark mode feature flag is switched on in Production.

Searching the Production workspace... The UI Dark Mode feature toggle is currently set to ON targeting 50% of the active user base.

U Turn off the experimental flag targeting our staging environment immediately.

Executing shutdown on experimental flag. The flag has been returned to its OFF (kill switch) state in Staging.

U List all active environments linked to our main workspace project.

Scanning project environments... I found 3 connected clusters: Staging, Production, and Beta Testing Node.

Frequently Asked Questions

01 **How do I list all available deployments using LaunchDarkly MCP?**

You use list_projects to get an overview of all connected projects. Then, call list_environments to see the specific deployment targets (like Staging or Production) for that project.

02 **Can I check flag status without knowing the exact flag name with LaunchDarkly MCP?**

First, use list_feature_flags to retrieve a comprehensive list of all available flags. Then, you can provide the specific name or ID when calling get_feature_flag.

03 **Does LaunchDarkly MCP help with debugging who changed what?**

Yes, running list_audit_logs retrieves a full history of every change made across your account. This is crucial for understanding the sequence of events that led to an issue.

04 **Is LaunchDarkly MCP only useful for Production environments?**

No, it works with all environments. You can use list_environments and get_environment to check status in Dev, Staging, or any other workspace you need to validate.

05 **How does LaunchDarkly MCP help me evaluate A/B tests?**

You call list_metrics and then get_metric. This pulls specific user engagement data for the test flags, letting you see if one version is performing better than another.

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 `"launchdarkly": { "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

LaunchDarkly 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 LaunchDarkly. 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	LaunchDarkly MCP
Server ID	`019d75c5-1fe9-70b2-a46d-4b13bfa22517`
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/launchdarkly.