MCP SERVER | NO CODE | CLOUD HOSTED

Hookdeck MCP

Manage Webhooks and Event Flow via Chat

Hookdeck manages your webhook infrastructure directly through conversation, letting you treat event routing like a chat command. You can list, create, and update connections, sources, and destinations without leaving your agent interface. It gives you deep visibility into every piece of data flowing through your system.

F Quality Score 3.6/100

webhooks | event-driven | api-integration | event-routing | observability

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

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.

Hookdeck MCP

52 tools available
Cloud-hosted on Vinkius

Running an event-driven application means dealing with webhooks—and webhooks are notoriously fragile. This MCP connects to Hookdeck, giving you total control over your entire webhook pipeline using natural language commands. You can instantly check how many sources feed into your system or pause a connection during maintenance without ever logging into the dashboard. Need to debug a weird routing issue? Fetch specific metadata for any source or destination using unique IDs. The Vinkius catalog makes this power available, letting you manage complex event reliability right from your chat client. You'll handle everything from listing connections and sources to updating rules or even manually retrying failed events, all through simple conversation.

Core Capabilities

01 — Inspect Webhook Topology

List and retrieve details for every connection, source, destination, event, and request in your webhook setup.

02 — Control Data Flow State

Pause or resume specific connections, sources, or entire pipelines to control traffic during deployments or outages.

03 — Manage Infrastructure Assets

Create, update, and delete core components like connections, destinations, sources, and transformations programmatically.

04 — Debug and Audit Events

Get granular metrics on request volume, queue depth, or retrieve the full history of specific events and attempts for debugging.

05 — Force Data Recovery

Manually retry failed requests or events to ensure data integrity after a temporary failure.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to this MCP and provide your Hookdeck API Key.

02 Connect the key in your preferred AI client, giving your agent access to the full webhook control panel.

03 Tell your agent what you need—for example, 'Show me all connections that are paused' or 'Retry the last failed event for source XYZ'.

The bottom line is: You manage a complex, production-grade data pipeline using only conversation.

Built For

This MCP targets highly technical roles that spend time debugging API failures and managing critical infrastructure. It’s for the backend engineer who can't afford to be blocked by a simple dashboard click, or the DevOps team member needing to audit event flow during an incident.

Backend Engineer

They use this MCP to quickly verify connection statuses and toggle webhook traffic when deploying new features.

DevOps Engineer

They monitor the health of event-driven pipelines, checking metrics like request volume or queue depth immediately after a deployment.

Integration Specialist

They verify complex routing rules and connection counts across staging and production environments to ensure data integrity.

What Changes When You Connect

01 Avoid context switching. Instead of jumping into a web dashboard to check statuses, you just ask your agent for the connection list or current queue depth.

02 Maintain reliability during deployments. If you need to test an endpoint change, use 'pause_connection' and then 'unpause_connection' directly in conversation, zero clicks required.

03 Deep debugging access. Need to know why a request failed? Use 'get_request' or 'list_attempts' to pull the exact payload and error details instantly for root cause analysis.

04 Build resilient pipelines. You can create new components—like running 'create_source' or setting up complex data handling with 'create_transformation'—without writing setup scripts.

05 Control recovery efforts. If a critical event fails, you don't wait; you call 'retry_event' to force the system to reprocess the data immediately.

Real-World Applications

Debugging a Failed Payment Webhook

The payment webhook failed after an update. Instead of logging into Hookdeck and manually sifting through logs, you ask your agent to 'get_metrics_attempts' for the relevant connection ID. The agent instantly shows which attempts failed and if they are retriable using 'list_attempts'.

Pre-Deployment Traffic Control

A team is updating a destination endpoint. Before going live, you use your agent to call 'pause_connection' on the production route. This guarantees no data gets sent while you test, and then you run 'unpause_connection' when it's safe.

Auditing Data Flow Changes

An integration specialist needs to know if a specific source is still active after a merge. They ask your agent to count sources using 'count_sources', then use 'get_source' for the problematic ID, confirming all rules are correct before proceeding.

Replaying Test Data

You manually created a perfect test request bookmark. You tell your agent to 'trigger_bookmark'. This re-runs that exact payload through the entire pipeline—including any transformations—without having to copy/paste the data again.

Patterns to Avoid

Manual Dashboard Monitoring

✗ AVOID

Waiting for a support ticket when your app fails, then manually navigating to Hookdeck's dashboard to find failure metrics or connection statuses.

✓ INSTEAD

Ask your agent directly: 'Show me the queue depth for my primary destination' (using 'get_metrics_queue_depth') or 'List all connections and their current status' (using 'list_connections').

Blindly Re-enabling Services

✗ AVOID

A system went down, so an engineer quickly enables everything to see if it works. This often causes more problems because they don't know which specific source or connection needs fixing.

✓ INSTEAD

Always isolate the problem first. Use 'list_sources', then use 'disable_source' on all but one test source. Once isolated, you can precisely call 'enable_source' only when ready.

Confusing Statuses

✗ AVOID

Seeing a connection is marked as 'active' in the UI, but data still isn't flowing because a dependency was missed.

✓ INSTEAD

Don't trust the status alone. Use your agent to call 'get_connection' and compare its retrieved rules against what you expect. If needed, use 'update_connection' before enabling it.

The Right Fit

Use this MCP if your core workflow involves complex data routing, event processing, or API integrations that require constant monitoring and manual intervention. You need to control the *flow* of data—pausing it for maintenance, checking its status, or replaying failures. It's essential when debugging unreliable systems where the failure point is hard to trace.

Don't use this if you only need static CRUD operations (like creating a simple record in a database) or if your 'messaging' requirement is just sending a simple text notification; those are better handled by dedicated messaging tools. If all you need is basic logging, the monitoring metrics are helpful, but for full operational control, this MCP is required.

The Pain of Webhook Debugging

Today, when an event fails or a webhook connection breaks, you're trapped in a dashboard. You click into the Connections tab, then drill down to Sources, then check the Metrics panel. If you need to know why it failed last Tuesday, you have to filter by date, scroll through logs, and copy/paste IDs just to hand them off to a teammate.

With this MCP, that multi-step journey vanishes. You simply ask your agent: 'What was the status of the Stripe connection yesterday?' The agent pulls the specific metrics instantly. It's not about reading; it's about asking.

Hookdeck MCP Brings Control to Your Event Pipeline

The manual process of updating rules, toggling traffic, or fixing failures requires jumping between dozens of tabs and managing complex JSON payloads. You spend time figuring out *where* the control button is.

Now you just talk to your agent. Need to reroute data? Say so. Pause it for testing? Tell it. The power isn't in the buttons; it’s in the conversation, giving you operational command over every piece of event infrastructure.

Hookdeck MCP with 52 Tools

Manage every aspect of your webhook infrastructure—from listing all connections to manually retrying failed events—using these specialized tools.

#	TOOL	DESCRIPTION
01	`cancel_event`	Stops scheduled retries for a specific event.
02	`count_connections`	Returns the total number of active connections in your account.
03	`count_sources`	Counts the total number of webhook sources configured.
04	`create_bookmark`	Creates a reusable bookmark for a specific request payload.
05	`create_connection`	Establishes a new connection to route webhooks from one source to another destination.
06	`create_destination`	Sets up a new endpoint where incoming webhook data will be sent.
07	`create_issue_trigger`	Generates a new trigger that creates issues in an external system.
08	`create_source`	Adds a new source endpoint to the webhook network.
09	`create_transformation`	Builds a new data transformation rule to modify incoming payloads.
10	`delete_bookmark`	Removes an existing request bookmark.
11	`delete_connection`	Permanently deletes a specific connection route.
12	`delete_destination`	Deletes an entire destination endpoint configuration.
13	`delete_issue_trigger`	Removes an issue trigger from the webhook setup.
14	`delete_source`	Permanently removes a source endpoint.
15	`disable_connection`	Deactivates a connection, stopping event routing immediately.
16	`disable_source`	Disables an entire webhook source from sending data.
17	`enable_connection`	Restores normal operation for a previously disabled connection.
18	`enable_source`	Reactivates a previously disabled webhook source.
19	`get_attempt`	Retrieves details about a single, specific delivery attempt.

#	TOOL	DESCRIPTION
20	`get_connection`	Fetches all configuration details for a specified connection route.
21	`get_destination`	Retrieves the full configuration and status of a target destination.
22	`get_event`	Gets comprehensive details about a single processed event.
23	`get_issue_trigger`	Retrieves the configuration for a specific issue trigger.
24	`get_metrics_attempts`	Returns delivery attempt metrics, showing success and failure rates over time.
25	`get_metrics_events`	Provides statistics on event processing performance.
26	`get_metrics_queue_depth`	Shows the current number of pending items waiting for each destination to process.
27	`get_metrics_requests`	Delivers metrics on overall request volume and throughput.
28	`get_metrics_transformations`	Presents data on how transformations are performing, including execution time.
29	`get_request`	Retrieves the full payload and metadata for a specific webhook request.
30	`get_source`	Fetches all details about a specific source endpoint configuration.
31	`get_transformation`	Gets the full code and settings for a defined data transformation.
32	`list_attempts`	Retrieves a list of past delivery attempts, including status codes.
33	`list_bookmarks`	Lists all existing request bookmarks you've saved.
34	`list_connections`	Pulls a list of every active connection route by name and status.
35	`list_destinations`	Lists all configured destination endpoints.
36	`list_events`	Provides a paginated list of recent processed events.
37	`list_issue_triggers`	Displays a catalog of all issue triggers in your setup.
38	`list_requests`	Lists the most recent webhook requests received by the system.
39	`list_sources`	Provides a list of all configured source endpoints.
40	`list_transformations`	Lists all defined data transformation rules.
41	`pause_connection`	Stops event routing for a specific connection route immediately.

#	TOOL	DESCRIPTION
42	retry_event	Forces a manual retry of a single, failed event record.
43	retry_request	Attempts to resend data for a rejected or failed request payload.
44	test_transformation	Runs specific transformation code against sample data to test its logic.
45	trigger_bookmark	Replays a previously saved, bookmarked request payload through the system.
46	unpause_connection	Resumes event routing for a connection that was paused.
47	update_connection	Modifies an existing connection's rules or name without deleting it.
48	update_destination	Updates the configuration details, such as URL or rate limits, for a destination.
49	update_issue_trigger	Modifies the settings of an existing issue trigger.
50	update_source	Updates configuration details or credentials for a source endpoint.
51	update_transformation	Changes the actual code or logic of a data transformation rule.
52	upsert_connection	Creates a connection if it doesn't exist, or updates it by name if it does.

See It in Action

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

U List all my current Hookdeck connections.

I've retrieved your connections. You have 3 active routes: 'Shopify-to-Lambda' (ID: conn_123), 'Stripe-to-Analytics' (ID: conn_456), and 'GitHub-Webhooks' (ID: conn_789).

U Pause the connection with ID conn_456 immediately.

The connection 'Stripe-to-Analytics' (conn_456) has been successfully paused. Event routing is now suspended.

U How many webhook sources do I have?

You currently have a total of 5 sources configured in your Hookdeck project.

Frequently Asked Questions

01 **How do I list all active connections using the Hookdeck MCP?**

You ask your agent to use 'list_connections'. It will retrieve a clean, formatted list showing the name and current status of every connection route you have set up.

02 **Can I test my data transformations with the Hookdeck MCP?**

Yes. You call 'test_transformation', giving it sample input data. The agent runs your code against that payload and reports back exactly what the output will look like, confirming its logic before deployment.

03 **What is the difference between listing sources and listing connections with Hookdeck MCP?**

Listing sources shows you *where* data originates (the input). Listing connections shows you *how* that data moves from one source to a specific destination (the route).

04 **If I need to fix a failed webhook, should I use 'retry_event' or 'retry_request' via the Hookdeck MCP?**

Use 'get_metrics_attempts' first. If you are retrying a known, specific failure instance, use 'retry_event'. If it was a general payload rejection, use 'retry_request'.

05 **How many tools does the Hookdeck MCP have?**

The Hookdeck MCP provides access to over 50 specialized functions for managing every aspect of your webhook infrastructure.

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 `"hookdeck": { "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

Hookdeck 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 Hookdeck. 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	Hookdeck MCP
Server ID	`019e38a8-7df3-73b4-994c-4868da0e6f61`
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/hookdeck.