MCP SERVER | NO CODE | CLOUD HOSTED

Hookdeck (Webhook Gateway) MCP

Control and debug every event flow via conversation.

Hookdeck (Webhook Gateway) provides full control over your event-driven architecture. Manage webhook sources, connections, and destinations directly from any AI client, allowing you to monitor the flow of data and debug failures without touching code or switching dashboards.

A+ Quality Score 98.33/100

webhooks | event-gateway | api-management | event-delivery | infrastructure

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

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 (Webhook Gateway) MCP

75 tools available
Cloud-hosted on Vinkius

Stop juggling multiple consoles just to manage a single failing webhook. This MCP connects your entire webhooks infrastructure—the sources, connections, and final destinations—to your agent using natural conversation. You can list all existing connections and immediately pause them if traffic spikes or maintenance is needed. Need to debug? Your AI client retrieves raw event bodies or lets you check the performance metrics for request volume and failure grouping. It even handles complex tasks like retrying rejected requests in bulk, keeping your data pipeline running smoothly. If managing webhook reliability feels too complicated for a single dashboard, Vinkius puts all this power into one place. You use your agent to treat it like having an instant DevOps engineer dedicated solely to your event flow.

Core Capabilities

01 — Audit and Monitor Event Flow

Get counts for sources, connections, destinations, and transformations, providing a quick health check of the entire webhook setup.

02 — Control Connection Status

Enable or disable any connection instantly, or pause it to mark events as held without deleting the configuration.

03 — Manage Data Components

Create, update, and delete core infrastructure elements like sources, destinations, connections, and transformations.

04 — Debug and Retry Events

Retrieve specific events or request raw body data. You can also trigger bulk retries for both rejected requests and failed event batches.

05 — Track Infrastructure Changes

View historical records, including delivery attempt patterns and metrics on processing backlogs to understand system performance over time.

One Click on Vinkius — From Prompt to Execution

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

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

02 Your AI client accesses the connection controls via conversation, allowing you to list sources or destinations.

03 You instruct your agent to perform an action—like disabling a source or retrieving event metrics—and the system executes it immediately.

The bottom line is that instead of navigating complex web UIs, you talk directly to your webhook infrastructure and get results.

Built For

This MCP serves backend engineers who need to debug failing event routes fast. It's perfect for DevOps teams needing automated control during deployments, or support staff troubleshooting client integrations under pressure.

Backend Engineer

You use this MCP to quickly check connection status and retrieve raw data bodies when a webhook fails in production.

DevOps/SRE

Your job is automating failover. You use the controls here to pause or unpause entire connections during planned maintenance windows.

Support Engineer

When a client reports an issue, you check source configurations and connection statuses to isolate whether the problem is upstream or internal.

What Changes When You Connect

01 Debug failures instantly. Instead of manually checking logs, you ask your agent to retrieve the raw body data for a failed event or request, giving you immediate context.

02 Manage status without code changes. You don't need to write scripts to pause an entire connection during maintenance; just tell your AI client to 'disable the connection.'

03 Handle backlogs efficiently. Use tools like bulk_retry_events and list_attempts to recover large batches of failed events, minimizing data loss in one command.

04 Gain full visibility into performance. Check metrics like get_metrics_queue_depth or count active connections to understand your system's health at a glance.

05 Simplify component management. You can create new sources or destinations and link them together by running commands that manage the entire lifecycle, from creation (create_source) to deletion (delete_source).

Real-World Applications

Debugging a payment webhook failure

A user reports a failed payment event. Instead of manually digging through logs, the agent calls get_event and get_event_raw to pull the exact payload and show why the destination rejected it. They then use retry_event to resend the data.

Preparing for a scheduled outage

The SRE team needs to update an endpoint. Before they touch anything, the agent runs list_connections and uses disable_connection on all related webhooks. This prevents any unexpected events from hitting the outdated system.

Analyzing systemic performance degradation

A product manager notices a spike in errors. The agent calls get_metrics_failures_by_issue, which immediately groups failures by issue type, telling them exactly where the data pipeline is breaking down.

Onboarding a new third-party integration

The backend engineer needs to add a new webhook source. They use list_sources to check for conflicts and then run create_source, ensuring the new origin is correctly set up before linking it to any connections.

Patterns to Avoid

Using generic logging tools

✗ AVOID

Searching through a general log management system for 'webhook error' results in thousands of lines, making it impossible to isolate the specific failing payload or connection ID.

✓ INSTEAD

Use your MCP to run get_event and get_source. This gives you direct access to the exact data payload and source configuration that failed, skipping hours of log sifting.

Manual dashboard switching

✗ AVOID

To pause a connection, an engineer must jump from the main connections tab to the settings panel, click 'pause,' then switch back to verify the status. This process is slow and error-prone.

✓ INSTEAD

Keep everything in your IDE or terminal. Simply instruct your agent: 'Pause the Shopify-Sync connection.' The MCP handles the state change instantly.

Ignoring data history

✗ AVOID

A system fails and is fixed, but you don't know how many events were lost in the gap. You only see the current status, not the backlog.

✓ INSTEAD

Always check list_attempts or get_metrics_queue_depth after an outage. This confirms if there was a processing backlog that needs to be addressed using bulk_retry_events.

The Right Fit

Use this MCP if your problem is managing the flow, status, and failure recovery of asynchronous webhook events. You need programmatic control over every component: sources, connections, transformations, and destinations. Don't use it if you only want to view general application logs; that requires a dedicated logging tool. If you just need to move data from Point A to Point B without complex monitoring or retry logic, a simple integration connector might suffice. However, if your business depends on reliability—meaning knowing *why* an event failed and having the ability to programmatically retry it—this MCP is essential. It lets you manage the entire lifecycle, from listing sources (list_sources) to executing bulk retries (bulk_retry_events).

The webhooks are your company's nervous system, and managing them feels like a constant chore.

Today, keeping your data pipelines running means jumping between three or four different dashboards. You check the source status on one tab, verify the connection rules on another, then finally go to a third dashboard just to see if events actually hit the destination. If something breaks, you're manually copying IDs and pasting them into separate troubleshooting forms.

With this MCP, all that complexity collapses into conversation. You tell your agent, 'The Stripe feed is failing,' and it instantly pulls up the source configuration, checks the connection status, retrieves the raw event body, and tells you exactly which component broke—all without leaving your terminal.

Mastering Event Control with Hookdeck (Webhook Gateway) MCP

The specific manual steps that disappear include checking connection status via list_connections, manually enabling a source with enable_source, and then needing to check metrics separately using get_metrics_events. All these actions become single commands.

Now you treat your entire event architecture like code: you audit it, you modify it, and you debug it—all from one place. You finally have full, conversational control over the flow of data.

Hookdeck (Webhook Gateway) 30 Tools

These tools let you list, create, update, delete, and monitor every single piece of infrastructure that handles your webhooks.

#	TOOL	DESCRIPTION
01	`bulk_cancel_events`	Bulk cancel events
02	`bulk_retry_events`	Bulk retry events based on a query
03	`bulk_retry_requests`	Bulk retry rejected requests
04	`cancel_bulk_retry_events`	Cancel an ongoing bulk retry
05	`cancel_event`	Cancel a scheduled retry for an event
06	`count_connections`	Count connections
07	`count_destinations`	Count destinations
08	`count_issues`	Get issue count
09	`count_sources`	Count sources
10	`count_transformations`	Count transformations
11	`create_bookmark`	Create a bookmark
12	`create_connection`	Can include inline source/destination. Create a connection
13	`create_destination`	Create a destination
14	`create_issue_trigger`	Create an issue trigger
15	`create_source`	Create a source
16	`create_transformation`	Create a transformation
17	`delete_bookmark`	Delete a bookmark
18	`delete_connection`	Permanently removes a specific webhook connection from your gateway.
19	`delete_destination`	Permanently removes a destination from the webhook network.

#	TOOL	DESCRIPTION
20	delete_issue	Dismiss an issue
21	delete_issue_trigger	Delete an issue trigger
22	delete_source	Removes a webhook source entirely from your system.
23	delete_transformation	Delete a transformation
24	disable_connection	Stops all events for a connection immediately, marking it as inactive without deletion.
25	disable_destination	Disable a destination
26	disable_source	Disable a source
27	enable_connection	Restores full event flow and activity to a previously disabled connection.
28	enable_destination	Enable a destination
29	enable_source	Enable a source
30	get_attempt	Retrieve a specific attempt
31	get_bookmark	Retrieve a bookmark
32	get_bulk_retry_events	Get status of a bulk retry
33	get_connection	Retrieves the detailed configuration and current status of a specific connection.
34	get_destination	Retrieves the configuration details for a specific destination.
35	get_event_raw	Retrieves the unparsed, raw body data for an event payload.
36	get_event	Retrieves the full data payload for a single event that passed through the gateway.
37	get_issue	Retrieve an issue
38	get_issue_trigger	Retrieve an issue trigger
39	get_metrics_attempts	Delivery attempt pattern metrics
40	get_metrics_events_by_issue	Failures grouped by issue metrics
41	get_metrics_events	Gathers metrics showing event processing rates and overall success percentages.

#	TOOL	DESCRIPTION
42	`get_metrics_queue_depth`	Monitor processing backlogs
43	`get_metrics_requests`	Request volume and status metrics
44	`get_metrics_transformations`	Transformation performance metrics
45	`get_request_events`	Retrieve events generated by a request
46	`get_request_ignored_events`	Retrieve ignored events for a request
47	`get_request_raw`	Get request raw body data
48	`get_request`	Retrieve a specific request
49	`get_source`	Gets comprehensive details and configurations for one specific source.
50	`get_transformation_execution`	Get a specific transformation execution
51	`list_attempts`	Retrieves a list of past delivery attempt records for tracking purposes.
52	`list_bookmarks`	Retrieve bookmarks
53	`list_connections`	Lists all established connections within your webhook gateway.
54	`list_destinations`	Shows all endpoints that receive the processed webhooks (the final destinations).
55	`list_events`	Fetches a list of events that have been processed by your system.
56	`list_issue_triggers`	Retrieve issue triggers
57	`list_issues`	Lists all recorded issues or failures within the webhook pipeline.
58	`list_requests`	Retrieve requests
59	`list_sources`	Retrieves a list of all connected webhook sources (the origins of the data).
60	`list_transformation_executions`	Get transformation executions
61	`list_transformations`	Retrieve transformations
62	`pause_connection`	Events will be marked as HOLD. Pause a connection
63	`publish_event`	Send events directly to Hookdeck

#	TOOL	DESCRIPTION
64	retry_event	Manually attempts to process a specific, failed event again.
65	retry_request	Retry a rejected request
66	test_transformation	Test a transformation
67	trigger_bookmark	Replay the bookmarked request
68	unpause_connection	Unpause a connection
69	update_bookmark	Update a bookmark
70	update_connection	Modifies settings for an existing connection, like changing its rules or parameters.
71	update_destination	Makes changes to an existing destination, like updating its required format or endpoint.
72	update_issue	Update issue status
73	update_issue_trigger	Update an issue trigger
74	update_source	Modifies settings on a source, such as changing its credentials or endpoint URL.
75	update_transformation	Update a transformation

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 and their status.

I've retrieved your connections. You have 3 active connections: 'Shopify-Sync' (Enabled), 'Stripe-Webhooks' (Paused), and 'Auth0-Events' (Enabled). Would you like more details on any of these?

U Disable the source with ID 'src_123abc'.

I have successfully disabled the source 'src_123abc'. It will no longer accept incoming webhook events until re-enabled.

U Show me the details for connection 'conn_xyz789'.

Fetching details for 'conn_xyz789'... This connection routes from Source 'GitHub-Repo' to Destination 'Vercel-Deploy'. It currently has 2 active rules and is in 'Enabled' status.

Frequently Asked Questions

01 **How do I check if a webhook connection is active using Hookdeck (Webhook Gateway) MCP?**

You use list_connections to see all connections. The status column tells you if it's currently enabled or paused, giving you immediate operational visibility.

02 **Can I fix a failed webhook event with the Hookdeck (Webhook Gateway) MCP?**

Yes, you can. After retrieving an event using get_event, you run retry_event to manually reprocess it through the gateway without any code changes.

03 **What if I need to stop a connection temporarily for maintenance? Do I delete it?**

No, don't delete it. Use disable_connection; this stops events immediately but keeps all your rules and configurations intact, ready to be re-enabled later.

04 **Does the Hookdeck (Webhook Gateway) MCP help with billing or usage metrics?**

The MCP provides detailed operational data. You can use get_metrics_events to see success rates and process volume, which helps you track usage patterns for optimization.

05 **How do I update a source's API key using Hookdeck (Webhook Gateway) MCP?**

You retrieve the current details with get_source first. Then, you use the update_source tool to apply the new credentials or configuration parameters.

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-webhook-gateway": { "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 (Webhook Gateway) 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 (Webhook Gateway). 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 (Webhook Gateway) MCP
Server ID	`019e38a8-c3e2-71cc-875f-e1edba49662d`
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-webhook-gateway.