MCP SERVER | NO CODE | CLOUD HOSTED

Honeybadger MCP

Stop clicking dashboards. Start asking questions.

Honeybadger MCP lets you monitor application health, track errors, and check uptime without leaving your development environment. Connect it to any AI client to list projects, analyze error groups, see backtraces from individual failures, or audit recent deployments using natural conversation.

A+ Quality Score 100/100

error-tracking | uptime-monitoring | exception-handling | application-health | cron-monitoring

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/honeybadger-error-tracking

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.

Honeybadger (Error Tracking) MCP

10 tools available
Cloud-hosted on Vinkius

Need to figure out why the staging site is glitching? Instead of jumping through five different dashboards—checking logs here, looking at deployment history there, and cross-referencing team ownership somewhere else—just talk to your agent. This MCP connects directly to Honeybadger, giving you full visibility into your app's health status using only conversation.

Your agent can list all monitored projects for you, then drill down to query fault groups to understand class names and environment distribution across your entire setup. If a specific error group looks bad, you can ask it to pull details from individual error notices, pulling backtraces and request data instantly. You can also check recent deployments or even audit team roles responsible for different projects. It’s powerful stuff, and since Vinkius hosts this MCP, connecting it is simple: just link your preferred AI client and start asking questions.

Core Capabilities

01 — Check Project Status

List all monitored applications, showing basic details like language used or the count of unresolved faults.

02 — Diagnose Error Groups

Query aggregate error groups (faults) to understand common patterns, class names, and where they are occurring across your infrastructure.

03 — Inspect Specific Failures

Retrieve deep details on single error occurrences (notices), including backtraces, request data, and the server environment that failed.

04 — Manage Faults and Uptime

Mark specific errors as resolved or ignore them to keep your dashboards clean. You can also check site availability and track recent code deployments.

05 — Audit Team Members

List the team members assigned to a project, helping you understand who owns which part of the bug-fixing workflow.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/honeybadger-error-tracking** — connect your AI agent in three steps.

01 Subscribe to this MCP and enter your Honeybadger Personal Auth Token.

02 Your AI client connects to the service, making all error monitoring data available through natural conversation.

03 You ask your agent specific questions—like 'What was wrong with the checkout page yesterday?'—and it returns detailed, actionable answers.

The bottom line is you get a single source of truth for all your application health data without switching tabs or writing complex API calls.

Built For

This is for the SRE who's sick of jumping between dashboards at 2 AM, or the developer who just needs to see a backtrace without leaving their IDE. If your job involves figuring out *why* something broke in production, this MCP saves you hours.

Software Developer

Uses it to debug exceptions and inspect full backtraces for specific errors directly within their coding environment.

Site Reliability Engineer (SRE)

Monitors global uptime sites, tracks deployment histories, and audits fault groups to ensure infrastructure stability across all environments.

Engineering Lead

Audits unresolved fault counts and team assignments to optimize incident response protocols and assign ownership for critical bugs.

What Changes When You Connect

01 Pinpoint failures faster: Instead of manually searching through logs, you can ask your agent to list notices or get a notice for the exact backtrace needed. This drastically cuts down debugging time.

02 Understand code changes instantly: By listing deployments and checking site availability, you immediately know if a recent code push is responsible for an increase in faults.

03 Simplify triage workflows: Use the 'resolve_fault' tool or similar actions to mark issues as resolved right from your chat window. Your team stays focused only on what needs active work.

04 Get project context quickly: Need to know who owns a bug? List members gives you that ownership data, while list_projects shows the overall status of all monitored services.

05 Deep dive into every failure: Don't just see an error message. Use 'get_notice' to retrieve session context and request data for granular understanding of what triggered the fault.

Real-World Applications

The production bug needs immediate root cause analysis

A developer sees a spike in errors. They ask their agent to list faults, identify the top error group, and then use get_notice on that fault to retrieve the full backtrace and request data. They find it's related to a specific API call missing required parameters.

Investigating service degradation after an update

An SRE notices uptime dipping. They use list_deployments to see what code was pushed 3 hours ago, then check the fault data against that deployment to confirm the new revision caused the regression.

Auditing project ownership and responsibilities

An Engineering Lead needs to know who should fix a critical bug. They ask the agent to list_members for the specific project, confirming which team member was assigned responsibility for that module.

Clearing up stale error reports

A developer confirms a previously reported fault is fixed in the latest release. Instead of logging into the dashboard, they simply ask their agent to resolve_fault, keeping the error count accurate and clean.

Patterns to Avoid

Treating it like a ticketing system

✗ AVOID

A user assumes that using list_projects will also allow them to assign tickets or change statuses outside of Honeybadger's core fault management.

✓ INSTEAD

Remember this MCP is for technical monitoring. Use get_fault and resolve_fault to manage the state of errors within Honeybadger, but keep external ticket creation in your separate project management tool.

Ignoring deployment history

✗ AVOID

A developer sees a fault increase but assumes it's random. They waste time checking logs without knowing *when* the problem started.

✓ INSTEAD

Always start by using list_deployments. This tells you if a recent code change triggered the issue, which immediately narrows down where to look for fixes.

The Right Fit

Use this MCP if your primary pain point is understanding *why* and *when* an application failed in production. You need deep visibility into error backtraces, uptime metrics, and deployment history. If you are constantly copy-pasting stack traces from logs or jumping between a monitoring dashboard and Git to check commits, this tool is for you.

Don't use it if your needs are purely workflow management (e.g., updating customer records) or general knowledge retrieval that isn't tied to application performance data. For those tasks, look into other MCPs in the Vinkius catalog that specialize in CRM or documentation tools. This tool is narrowly focused on app health and runtime failure analysis only.

The Pain of Context Switching

Today, figuring out a complex error means jumping between five different places. You check the dashboard for an alert, then you open your Git client to see recent commits, you switch to a log viewer to pull the raw stack trace, and finally, you copy all that information into a ticket in Jira just to write it down. It's slow, tedious, and you lose context every time.

With this MCP, you ask your agent directly about the error. You tell it the fault ID, or even just 'the checkout failure from yesterday.' The agent pulls the backtrace, checks related deployments, and gives you a cohesive answer right in your chat window. All the data you need lives here.

Honeybadger MCP: Deep Error Analysis

You no longer have to manually list projects or query fault groups just to get an overview. The agent handles that orchestration for you, summarizing class names and environments so you know exactly where the problem is spread.

It’s not about getting data; it's about instant answers. You tell your agent 'Show me all unresolved faults in production,' and it gives you a clean list of what needs fixing, without you having to click through menus.

Honeybadger (Error Tracking) with 10 Tools

Use these tools to list projects, diagnose fault groups, retrieve detailed notices, and manage deployment status for your application's health.

#	TOOL	DESCRIPTION
01	`list_projects`	Lists all projects in Honeybadger, returning key details like the project name, ID, language used, and current error counts.
02	`get_project`	Retrieves complete information for a specific Honeybadger project when you provide its ID.
03	`list_faults`	Lists error groups (faults) within a project, detailing class names, common messages, and how many times they occurred.
04	`get_fault`	Gathers all details about one specific fault group, including its full history and scope.
05	`resolve_fault`	Marks a specified Honeybadger fault as resolved, removing it from the list of active issues.
06	`list_notices`	Lists individual error occurrences (notices) belonging to a specific fault group for review.
07	`get_notice`	Gets the full, detailed record of one single error notice, including backtraces and session context.
08	`list_sites`	Lists all external uptime monitoring sites associated with a project so you can check service availability.
09	`list_members`	Retrieves the full list of team members registered on a project, including their assigned roles.
10	`list_deployments`	Shows the most recent code deployments for a project, detailing who pushed the change and when it happened.

See It in Action

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

U List all unresolved faults in my 'production-backend' project

I've found 3 unresolved faults in 'production-backend': 'NoMethodError: undefined method [] for nil:NilClass', 'Timeout::Error', and 'ActiveRecord::RecordNotFound'. Which one would you like to inspect?

U Show me the details for fault ID 123456

Fault ID 123456 ('NoMethodError'): First seen 2 days ago, occurred 45 times in 'production'. Last noticed today at 10:30 AM. It's currently unassigned. Would you like to see the last notice details or the backtrace?

U List recent deployments for project ID 9876

I've listed the 3 most recent deployments for project 9876: Revision 'a1b2c3d' by Alex (production, 2 hours ago), Revision 'e5f6g7h' by Sarah (staging, yesterday), and Revision 'i9j0k1l' by Alex (production, 3 days ago).

Frequently Asked Questions

01 **How do I check if the error was caused by a recent deployment using Honeybadger MCP?**

You can use list_deployments first. This shows the revision history and who pushed what. Then, you cross-reference those dates with your fault data to see if an increase in errors aligns with a specific code change.

02 **Can I resolve faults using the Honeybadger MCP?**

Yes, you can use 'resolve_fault' to mark issues as fixed. This is useful when your team confirms that a critical bug has been addressed and should no longer be tracked.

03 **What information does get_notice provide for debugging?**

get_notice gives you the most granular data point: the full backtrace, request parameters, session context, and the exact server environment where that single error occurred. It's essential for root cause analysis.

04 **Does Honeybadger MCP help with uptime monitoring?**

Yes, it includes tools like list_sites to monitor site availability across different external endpoints. This lets you confirm if the failure is system-wide or limited to a specific service.

05 **Do I need developer credentials for Honeybadger MCP?**

You must provide your personal Auth Token when setting up the connection. The agent uses this token to authenticate and pull all the necessary project data on your behalf.

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 `"honeybadger-error-tracking": { "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

Honeybadger (Error Tracking) 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 →

INDEPENDENT PLATFORM DISCLAIMER

Vinkius is an independent platform and is not affiliated with, endorsed by, sponsored by, verified by, or otherwise authorized by Honeybadger (Error Tracking). 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	Honeybadger (Error Tracking) MCP
Server ID	`019d75b2-66d2-7018-854e-6707705623e0`
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/honeybadger-error-tracking.