MCP SERVER | NO CODE | CLOUD HOSTED

FireHydrant MCP

Manage Outages and Service Dependencies in Chat

FireHydrant connects your incident management system to any AI agent, letting you handle complex outages through natural conversation. Instead of clicking through dashboards, you can ask your agent to list active incidents, declare new sev-2 alerts, check service dependencies, and update the timeline—all from your chat interface.

A+ Quality Score 100/100

incident-management | site-reliability | service-catalog | incident-response | on-call | post-mortem

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

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.

FireHydrant MCP

12 tools available
Cloud-hosted on Vinkius

FireHydrant lets you manage major service disruptions without ever leaving your chat window. It connects directly to your existing incident management platform, giving your AI client a full view of what's broken, who needs to know, and why it happened.

When an outage hits, time is critical. You can simply ask to list all currently active incidents or check the status of a specific service. Need to start a new response? Just tell your agent to create an incident, specifying severity levels, and get it logged immediately. The system keeps track of everything: who's on the team, what changes might have caused this mess, and every note added to the timeline.

This kind of operational intelligence usually lives in separate dashboards, requiring you to switch between tabs just to understand dependencies or assign personnel. Using your AI client through Vinkius means all that data—service catalogs, responder teams, runbooks, even post-incident reviews—comes together conversationally. You talk to it like a teammate and get actionable operational steps back.

Core Capabilities

01 — Track active incidents

Retrieve lists of current or past service outages using list_incidents.

02 — Declare new incidents

Quickly log a new incident with create_incident, assigning severity levels and initial details.

03 — Check service status and dependencies

Understand the impact of an outage by checking all defined services using list_services or getting specific data with get_service.

04 — Manage responder teams

Find out which personnel are available and assigned to handle a given crisis using list_teams or get_team.

05 — Update incident records

Add status notes or formal updates to the timeline of an existing incident with add_incident_note or update_incident.

06 — Review historical data and changes

Gather context by listing change events, viewing runbooks, or retrieving past retrospectives using list_change_events, list_runbooks, or list_retrospectives.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to this MCP via Vinkius and plug in your FireHydrant API Key.

02 Start talking to it through any AI client; just ask it to perform an operational task, like listing all current incidents or getting service details.

03 Your agent uses the underlying tools to fetch accurate data and delivers a conversational summary of the findings.

The bottom line is you get access to complex incident management data using simple chat commands instead of jumping through multiple web forms and dashboards.

Built For

This MCP is built for Site Reliability Engineers (SREs) and DevOps teams who are tired of clicking through seven different tabs just to understand the scope of a single outage. It's for anyone whose job involves coordinating complex, high-stakes system responses under pressure.

Site Reliability Engineer

You use this MCP to quickly declare incidents and fetch service details during a major outage, bypassing manual dashboard navigation.

Incident Commander

When things go sideways, you use it to automate gathering team information and post updates to the timeline without manually copying data.

DevOps Engineer

You leverage this MCP to review recent change events or check active runbooks immediately after an alert triggers, speeding up diagnosis.

What Changes When You Connect

01 You stop hunting for data. Instead of manually checking multiple dashboards, you can use the `list_incidents` tool to get a comprehensive overview of every active outage instantly.

02 Team coordination becomes automatic. Need to know who's available? The MCP lets you call `list_teams` and confirm that the right people are assigned before calling for help.

03 Incident documentation is fast. Instead of copying status updates into separate tickets, use `add_incident_note` to post everything directly to the incident timeline.

04 Service impact analysis is immediate. By using `list_services`, you don't just see a service is down; you see what other systems depend on it.

05 Post-mortem learning gets easier. You can call `list_retrospectives` and retrieve historical deep dives, ensuring every incident leads to better procedures.

Real-World Applications

The critical system is down, but I don't know what failed.

An agent asks: 'What happened?' The MCP runs `list_change_events` and immediately provides a list of recent infrastructure changes that might have triggered the outage. This cuts hours off initial diagnosis.

I need to start documenting this major incident right now.

The Incident Commander asks: 'Start logging sev-1 for Payment Gateway.' The agent uses `create_incident` and automatically notifies the relevant responder teams, getting the process started in seconds.

I need to update my team on the current status of the outage.

The manager asks: 'What's the latest?' The agent runs `get_incident` and pulls all recorded notes and updates, summarizing them for a stakeholder meeting without needing to log into the platform.

We need to find out if two services are related.

The engineer asks: 'What depends on the authentication service?' The agent runs `get_service` and lists all downstream dependencies, preventing them from missing a critical link during recovery.

Patterns to Avoid

Treating it like a knowledge base search.

✗ AVOID

Asking the system to 'Tell me about database latency.' This treats the incident platform like Google, giving vague, non-actionable answers.

✓ INSTEAD

Instead, use `list_incidents` to find an active alert ID, then ask: 'Get details for inc_456.' This forces the agent to retrieve structured data and actionable facts.

Manual status updates via copy/paste.

✗ AVOID

Copying a finding from Slack into the incident platform's notes section, risking human error or formatting issues.

✓ INSTEAD

Use `add_incident_note` to post your update directly through the chat. This ensures the note is formatted correctly and instantly visible in the timeline.

Ignoring the service catalog entirely.

✗ AVOID

Focusing only on the immediate failing component without understanding its impact, potentially missing a critical upstream dependency failure.

✓ INSTEAD

Always start by running `list_services` to map out dependencies. This gives you the full operational picture before jumping into remediation.

The Right Fit

Use this MCP if your core problem is coordinating complex, multi-step technical operations—like incident response or service dependency mapping. If your workflow requires tracking who does what, when, and why during a crisis, this MCP is for you. You need the ability to read past incidents (`list_retrospectives`), identify potential causes (`list_change_events`), and update real-time status across multiple teams. Don't use it if you simply need to store information; that’s a document management tool. Also, don't use it if your goal is just general knowledge retrieval; for specific troubleshooting guides, look for dedicated runbook tools or documentation repositories. If the action requires manipulating state (creating an incident, updating a record), this MCP handles it.

The Pain of Outage Coordination

Right now, when your service goes down, you're juggling five different tabs: the main alert dashboard, the runbook wiki, the team chat, the dependency map, and a separate ticketing system. You have to copy findings from one screen into another, manually updating status fields like 'Investigating' or 'Mitigated.' It's a cycle of switching context, clicking through menus, and doing repetitive data entry.

With this MCP, you speak naturally to your agent. Instead of juggling tabs, you simply ask: 'What do we know about the Payment Gateway outage?' The system doesn't just give you a link; it runs `get_incident` and delivers all the current status updates, team assignments, and related service details in one clear conversation. You get answers, not links.

FireHydrant MCP: Command Incident Response

The manual steps that disappear include running a search across five different systems to compile an incident summary; manually checking the status of every dependent service; and writing out repetitive 'Update' notes multiple times. These are hours spent on coordination, not fixing.

Now, you get a single point of truth. You use this MCP to declare incidents with `create_incident` or update them using `update_incident`. The difference is that your entire operational history—the investigation, the fix, and the post-mortem — lives right where you are talking.

FireHydrant: 12 Tools for Incident Ops

These tools let you automate every part of the operational lifecycle, from logging a new service disruption to retrieving detailed historical records.

#	TOOL	DESCRIPTION
01	`add_incident_note`	Adds a specific note directly to the timeline of an existing incident record.
02	`create_incident`	Initiates and logs a brand new service incident, setting its initial status and severity.
03	`get_incident`	Retrieves complete, detailed information on one specific incident by its ID.
04	`get_service`	Fetches detailed operational data for a single service from the catalog.
05	`get_team`	Pulls specific details about an assigned responder team.
06	`list_change_events`	Provides a list of infrastructure or code changes that occurred recently, useful for root cause analysis.
07	`list_incidents`	Retrieves a summary list of all current and historical incidents logged in the system.
08	`list_retrospectives`	Lists past post-incident reviews, helping you learn from previous outages.
09	`list_runbooks`	Shows a list of active runbooks available for use when responding to an incident type.
10	`list_services`	Lists every service defined in your catalog, allowing you to understand dependencies at a glance.
11	`list_teams`	Retrieves a list of all available responder teams and their current status.
12	`update_incident`	Modifies key fields or the overall status of an existing incident record.

See It in Action

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

U List all currently active incidents in FireHydrant.

Checking for active incidents... I found 2 ongoing incidents: 'Database Latency Issue' (sev-2) and 'Payment Gateway Timeout' (sev-1). Would you like the details for any of these?

U Declare a new sev-2 incident: 'Redis Connection Spikes'.

Incident declared! I've created a new sev-2 incident named 'Redis Connection Spikes'. The incident ID is 'inc_123'. Responder teams are being notified.

U Add a note to incident 'inc_123': 'Investigating potential cache invalidation issue'.

Note added! Your update has been successfully posted to the timeline for incident 'inc_123'. Responders will see this investigation update immediately.

Frequently Asked Questions

01 **Can FireHydrant MCP list all current outages?**

Yes. You use `list_incidents` to fetch a summary of every active and historical incident in the system, giving you a quick operational overview.

02 **How do I check what services are affected by an outage using FireHydrant MCP?**

You call `list_services`. This tool gives you the full service catalog, allowing you to map out dependencies and understand the total scope of impact.

03 **Does FireHydrant MCP help assign people during an incident?**

Yes. Use `list_teams` or `get_team` to see which responder teams are available, ensuring you assign the right experts immediately when declaring an event with `create_incident`.

04 **I need to record a finding from the investigation; can FireHydrant MCP do that?**

Absolutely. Use `add_incident_note` to post your findings directly to the incident timeline, making sure every responder sees your update immediately.

05 **Can I retrieve information about past outages with FireHydrant MCP?**

Yes. You can use `list_retrospectives` to view previous post-incident reviews and lessons learned from similar system failures.

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 `"firehydrant": { "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

FireHydrant 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 FireHydrant. 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	FireHydrant MCP
Server ID	`019d759a-08f0-71ae-92ae-56ed2bb97c78`
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/firehydrant.