MCP SERVER | NO CODE | CLOUD HOSTED

PagerDuty Incident Trigger MCP

Instantly declare system failures for on-call teams.

PagerDuty Incident Trigger MCP lets your AI agent immediately declare and escalate system failures to on-call engineers. This single-purpose connector uses PagerDuty's Events API V2, giving agents the safe authority to signal critical issues without granting access to sensitive incident history or scheduling tools.

F Quality Score 3.6/100

incident-response | on-call | alerting | automation | events-api | site-reliability

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/pagerduty-incident-trigger

SCAN TO VIEW

The connectivity layer between AI and the world's software.

Vinkius sits between AI and every application. All communication passes through Vinkius Cloud via the Model Context Protocol (MCP) — with governance, observability, and security at every layer.

Your AI Connections Run Through Vinkius Cloud

The world's largest managed MCP catalog

Vinkius is the connectivity layer where AI connects 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.

PagerDuty Incident Trigger MCP

1 tools available
Cloud-hosted on Vinkius

Need to give your AI agent a way to actually wake up an engineer? That’s what this MCP does. It provides a surgical connection that lets any compatible client invoke immediate alerts in PagerDuty. The tool is designed with zero-trust principles: it only sends out the alarm, nothing else. You can use your agent to evaluate system anomalies and instantly declare an issue—providing a summary, source, and severity level—to get attention on the phone.

This approach means you don't have to build a massive integration that tries to manage schedules or read through old tickets. It’s pure escalation power. By using this MCP via Vinkius, your agent gains the immediate ability to push an event directly into PagerDuty. This is how you give AI agents critical incident response capabilities without giving them access to sensitive operational data. You simply tell it what went wrong, and PagerDuty handles the rest.

Core Capabilities

01 — Declare System Failures

Your agent can instantly signal an issue in PagerDuty by providing a clear summary, source, and severity level to alert the on-call team.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/pagerduty-incident-trigger** — connect your AI agent in three steps.

01 Instruct your AI agent to identify a system anomaly that needs immediate attention.

02 Your agent uses this MCP's tool to format the required details: a summary, the source of the issue, and whether it's critical, error, or warning level.

03 The MCP executes the call, sending the event directly into PagerDuty's system, triggering the automated on-call escalation process.

The bottom line is that your agent can use context from a log file or dashboard to turn an observed error into a pagable incident alert in minutes.

Built For

This MCP targets Site Reliability Engineers (SREs), DevOps teams, and Incident Commanders. It's for the ops engineer who gets tired of manually checking dashboards at 2 AM just to realize an alert needs to be escalated somewhere else. You need a way to turn raw system context into actionable emergency calls.

Site Reliability Engineer (SRE)

You use this MCP when automated monitoring detects a pattern of failure, and you need your agent to immediately escalate that specific alert type without human intervention.

DevOps Engineer

Your team uses this to connect newly deployed service metrics directly to the incident response process, ensuring critical failures wake up the right people instantly.

Incident Commander

You rely on this when multiple systems are failing and you need your agent to consolidate logs into a single, actionable alert for the entire team.

What Changes When You Connect

01 Immediate Escalation: Your agent declares a 'Critical' state with one command. PagerDuty takes care of the rest, ensuring the right person gets paged instantly when an anomaly is detected.

02 Zero Scope Creep: This MCP only triggers alerts. It never reads your incident history or modifies on-call schedules, giving you maximum safety and minimal risk exposure.

03 Simplified Setup: You avoid complex OAuth flows or massive SDKs. Just supplying a single Integration Key allows the agent to push events directly without hassle.

04 Contextual Alerting: The tool requires specific inputs—a summary, source, and severity. This forces your agent to provide structured context with every alert it sends out.

05 Direct Automation: You can connect this MCP to any client (Claude, Cursor, etc.) to build automated pipelines that move from log parsing straight into incident declaration.

Real-World Applications

Monitoring Tool Failure

A monitoring system detects a 90% failure rate on the payment gateway. Instead of an engineer manually logging into PagerDuty, they prompt their agent: 'The payments are failing.' The agent uses `trigger_pagerduty_incident` to send a critical alert immediately, bypassing manual dashboard checks.

Post-Deployment Validation

A new microservice deployment finishes its smoke tests and reports all services are nominal. The automation pipeline needs confirmation that the on-call team is aware of the successful deployment state. It uses `trigger_pagerduty_incident` with a low severity ('info') to signal completion.

Log Analysis Escalation

An agent processes thousands of lines of log data and finds repeated database connection failures in the logs' source. It automatically calls `trigger_pagerduty_incident`, summarizing the failure pattern and marking it as an 'error' to initiate investigation.

Scheduled Maintenance Alert

A scheduled maintenance script runs successfully but requires human acknowledgment of a known, non-critical state change. The agent sends a warning alert using `trigger_pagerduty_incident` so the team knows to check the system soon.

Patterns to Avoid

Giving agents full API access

✗ AVOID

Connecting an AI agent with read/write permissions across all PagerDuty APIs. This allows it to delete incidents, modify schedules, or dump sensitive operational data.

✓ INSTEAD

Only use this MCP's tool: `trigger_pagerduty_incident`. It limits the agent's power solely to declaring a state of emergency (an alert), keeping its capabilities contained and safe.

Using general purpose webhooks

✗ AVOID

Sending raw JSON payloads or generic HTTP POST requests. These lack the structured data required by PagerDuty, often failing to trigger proper alerts.

✓ INSTEAD

Always use `trigger_pagerduty_incident`. It wraps your input (summary, source, severity) into the exact format PagerDuty expects for a reliable alert.

Relying on manual copy/paste

✗ AVOID

An engineer manually reading an error message from a terminal and then logging into PagerDuty to type out what happened. This introduces delays and human errors.

✓ INSTEAD

Let your agent handle this. The agent consumes the raw source data and passes it directly to `trigger_pagerduty_incident`, automating the entire alert lifecycle.

The Right Fit

Use this MCP if your primary need is for an AI client to translate a detected system failure or anomaly into an immediate, structured alarm that pages an on-call team. It's the dedicated 'fire alarm' button for automation.

Don't use it if you need your agent to read historical incident data

(you'd need a separate read-only query tool), modify who is on call, or resolve tickets. This MCP has absolute containment; its only job is signaling danger. If your workflow requires reading past context before alerting, this specific MCP won't cut it—you'll need a different type of connector to fetch that data first.

Alerting teams today feels like playing phone tag with dashboards.

When an issue pops up, what happens? You copy the error code from the terminal. Then you open your incident management tool. You paste the summary into a ticket, manually set the severity level (Critical? Warning?), and finally, you have to figure out which channel or person needs to be alerted next. It's slow, it involves too many clicks, and something always gets missed.

With this MCP, the process shrinks down to one step. Your agent reads the error code, determines the severity, and uses `trigger_pagerduty_incident`. The system handles the rest—the escalation path, the paging, and the notification blast—all automatically.

Triggering Incidents with PagerDuty Incident Trigger MCP

You ditch the manual workflow of copy-pasting logs into a ticket and then remembering to hit 'escalate.' The agent automatically captures the source data, summarizes the failure context, and sends it all in one structured payload.

What's different now is that your AI agent moves from being a log reader to an active responder. It doesn't just report; it initiates action. Period.

PagerDuty Incident Trigger: 1 Tool

These tools allow your AI client to send structured alerts directly into PagerDuty, automating incident declaration when systems fail.

#	TOOL	DESCRIPTION
01	`trigger_pagerduty_incident`	Sends an immediate incident event to PagerDuty using the Events API V2, requiring only a clear summary and source of the issue. The severity defaults to critical.

See It in Action

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

U **Trigger a critical PagerDuty incident for the payment-gateway saying transactions are failing.**

I've successfully triggered the critical incident in PagerDuty.

Frequently Asked Questions

01 **How does the PagerDuty Incident Trigger MCP work with my existing monitoring systems?**

Your monitoring system generates data, and your agent reads that data. The agent's job is to call `trigger_pagerduty_incident` when a threshold is crossed, ensuring the alert gets into PagerDuty correctly.

02 **Can I use the PagerDuty Incident Trigger MCP to view old incident history?**

No. This MCP is designed for absolute containment and only supports triggering new incidents. It cannot read or query any previous alert data, keeping your system secure.

03 **What severity levels can I use with trigger_pagerduty_incident?**

You can specify 'critical', 'error', 'warning', or 'info'. If you don't specify a level, the tool defaults to critical, which is usually safest for automated alerting.

04 **Does PagerDuty Incident Trigger MCP require me to use Python?**

No. You connect this MCP via any compatible client (Cursor, Claude, etc.). The agent's code just needs to call the tool; it doesn't dictate your underlying programming language.

05 **Is PagerDuty Incident Trigger MCP secure?**

Yes. Its one-way nature means the AI client can only send data out (trigger alerts) and has zero permissions to read or modify existing records in PagerDuty.

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 `"pagerduty-incident-trigger": { "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

PagerDuty Incident Trigger 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 PagerDuty Incident Trigger. 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	PagerDuty Incident Trigger MCP
Server ID	`019e38d1-dace-70d2-bfa6-163547418905`
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/pagerduty-incident-trigger.