MCP SERVER | NO CODE | CLOUD HOSTED

Opsgenie MCP

Control alerts, incidents, and on-call status.

Opsgenie helps you automate your incident response. Connect it to any agent and manage alerts, track who is on-call, and coordinate major outages conversationally. Instead of clicking through dashboards during a crisis, simply ask your AI client to acknowledge an alert, check the current rotation schedule, or create a full incident report.

A+ Quality Score 100/100

alerting | on-call | incident-response | sre | monitoring | workflow-automation

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

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.

Opsgenie MCP

11 tools available
Cloud-hosted on Vinkius

Running an SRE team means living in a constant cycle of alerts and escalations. This MCP gives you control over that entire process directly from your preferred agent. You can talk to it like talking to a teammate: 'What's wrong with the database?' or 'Who owns this incident right now?'.

It lets you handle everything from simple alert management—like adding notes and closing tickets—to coordinating major incidents with defined priority levels. Need to know who is covering the primary schedule next week? Just ask. You can list all schedules, check specific on-call personnel, or query historical data across both alerts and incidents. When you connect it via Vinkius, your agent gains immediate access to this powerful operational visibility. It turns a stressful, multi-tab manual process into simple conversation.

Core Capabilities

01 — Acknowledge an alert

Mark an open alert as seen so the rest of the team knows you're working on it.

02 — Add notes to alerts and incidents

Log technical details or investigation steps directly into the activity log for full audit history.

03 — Close an alert or incident

Formally mark a problem as resolved once troubleshooting is complete.

04 — Create new alerts and incidents

Manually trigger the process for an outage when automated systems fail to do so.

05 — Check on-call status

Find out who is currently assigned to handle a specific schedule or service.

06 — List schedules and alerts

View comprehensive lists of all team rotations, active alerts, and past incidents for auditing.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to this MCP and provide your Opsgenie API Key (GenieKey).

02 Optionally, specify the region you operate in—US or EU.

03 Start talking to it through your AI client: 'List all open P1 alerts' or 'Who is on-call for SRE?'

The bottom line is that your agent talks directly to Opsgenie, retrieving and manipulating real incident data without you ever leaving your chat window.

Built For

This MCP is built for the people running production systems. It's for the ops engineer who spends too much time clicking through dashboards at 2 a.m., and for the incident manager who needs to coordinate resources instantly during an outage.

SRE Engineer

Uses this to acknowledge alerts, add technical notes while investigating issues in their IDE, or check if a service is currently flapping.

Incident Manager

Relies on it to create major incidents quickly and verify the current on-call roster during high-pressure outages.

Support Team Lead

Checks if a reported issue is already an active alert or incident, directing users to the right resource immediately.

What Changes When You Connect

01 Stop hunting for information. You can ask your agent to check the current rotation schedule or list all open P1 alerts instantly, eliminating manual dashboard navigation.

02 Maintain a perfect record of events by using 'add_note' directly through your AI client. Every action and piece of investigation data stays linked to the right alert or incident.

03 Accelerate response time. Instead of finding the contact email, simply ask your agent who is on-call for a given schedule via 'get_who_is_on_call', routing issues immediately.

04 Streamline outage recovery. After fixing something, you can use 'close_alert' and 'create_incident' to formally document the timeline, keeping audit logs clean and accurate.

05 Get full visibility into past events. Use 'list_alerts' or 'list_incidents' to query historical data instantly when performing a root cause analysis.

Real-World Applications

The Support Ticket Triage

A support agent receives a ticket about payment failures. Instead of reading the internal runbook, they ask their agent to check for active alerts. The agent runs 'list_alerts', finds an open P1 alert, and tells the agent: 'There's already an incident open; I'll add a note linking this ticket.' This saves 15 minutes of searching.

The Critical Outage Response

An API service fails at 3 AM. The on-call engineer asks their agent: 'Who is responsible for the auth gateway?' The agent runs 'get_who_is_on_call' and provides the name instantly, allowing the engineer to start communicating immediately.

Post-Mortem Documentation

The team finished an outage. Instead of copying details from three different dashboards, the Incident Manager asks their agent to 'get_incident' details and list all related alerts. The agent compiles a clean summary for the post-mortem report.

Proactive Monitoring

A junior engineer wants to check if their service is covered by an alert. They ask: 'Are there any open alerts for the payment processing queue?' The agent runs 'list_alerts' and confirms that no issues are currently flagged, giving them peace of mind.

Patterns to Avoid

Checking dashboards manually

✗ AVOID

A user spends 10 minutes opening the Opsgenie UI, clicking through tabs, and copying ticket IDs to determine if an issue is already known.

✓ INSTEAD

Ask your agent directly: 'List all open P2 alerts related to payment.' This uses 'list_alerts' instantly and gives you a clean list without leaving your conversation.

Forgetting the context

✗ AVOID

An engineer fixes an issue but forgets to document the fix, leading to confusion during the next shift handover.

✓ INSTEAD

After fixing it, use 'add_note' immediately in your agent: 'Root cause was X; fixed by Y.' This ensures the historical record is accurate and complete.

Creating alerts twice

✗ AVOID

Two different people notice an issue and both manually trigger a new alert, creating confusion about which one is primary.

✓ INSTEAD

The first person who sees it should use 'create_incident' to establish the main incident record. Others then reference that ID when using 'add_note'.

The Right Fit

Use this MCP if your core problem is *alert visibility* and *on-call coordination*. You need a single source of truth for who owns what, and whether or not an alert exists right now. This MCP excels at reading the current state (listing alerts via 'list_alerts' or checking staff availability via 'get_who_is_on_call'). Don't use this if you primarily manage internal team communication; while you can add notes, it isn't a full ticketing system like Jira. If your main goal is generating reports from structured data that requires complex joins across multiple databases (e.g., combining Opsgenie with GitHub commits), consider a dedicated workflow automation tool. But for real-time incident status and person accountability, this is the right choice.

The stress of manual incident checks

Today, when something breaks, you're forced into a multi-tab nightmare. You jump between your monitoring dashboard, the on-call roster spreadsheet, and the alert history page. If you need to know who is covering the East Coast schedule right now, you spend time cross-referencing names and shift times. It’s slow, it's tedious, and you risk missing critical context.

With this MCP connected through Vinkius, that process vanishes. You just ask your agent: 'Who is on-call for the East Coast?' The answer comes back immediately, linking directly to the relevant people and schedules. It turns hours of clicking into a single question.

Opsgenie MCP gives you instant incident coordination

Manual processes often require logging every action—acknowledging, updating status, adding notes—across multiple interfaces. This makes the final post-mortem report a massive headache of copy-pasting and data reconciliation.

Now, your agent handles it all. You acknowledge the alert, add the technical details using 'add_note', create an incident record with 'create_incident', and close everything out—all in one conversation thread. The entire audit trail is seamless.

Opsgenie: 11 Tools for Incident Response

These tools let you programmatically control your incident response process. You can acknowledge alerts, check who is on-call, and manage the entire lifecycle of an outage.

#	TOOL	DESCRIPTION
01	`acknowledge_alert`	Marks a specific alert as acknowledged so that response time metrics are accurate.
02	`add_note`	Attaches textual information or investigation findings to an existing alert record.
03	`close_alert`	Closes a specific alert when the root cause has been resolved and verified.
04	`create_alert`	Generates an entirely new Opsgenie alert based on specified criteria.
05	`create_incident`	Initiates a major incident record with defined priority and scope.
06	`get_alert`	Retrieves all current details for one specified alert ID.
07	`get_incident`	Pulls the full history and status of a specific major incident.
08	`get_who_is_on_call`	Checks the current on-call schedule to identify the responsible team member or individual.
09	`list_alerts`	Gives a comprehensive list of all currently open and historical alerts in the system.
10	`list_incidents`	Lists multiple major incidents that have occurred or are currently active for review.
11	`list_schedules`	Displays all defined on-call rotation schedules and their next changes.

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 open P1 alerts.

I've found 3 open P1 alerts. 1. 'Database latency high' (ID: 4930), 2. 'API Gateway 5xx errors' (ID: 4931), 3. 'Payment service down' (ID: 4932).

U Acknowledge alert 4930 and add a note saying 'I am investigating the connection pool limits'.

Alert 4930 has been acknowledged and the note has been added to the activity log. You are now the owner of this alert.

U Who is currently on-call for the 'SRE-Primary' schedule?

The current on-call responder for 'SRE-Primary' is Sarah Miller (sarah.miller@example.com). Her shift ends at 08:00 AM UTC tomorrow.

Frequently Asked Questions

01 **How do I check if an alert has already been acknowledged using Opsgenie MCP?**

You can use the 'get_alert' tool to pull all details for a specific alert ID. This data includes the activity log, which will confirm who owns it and when it was last updated.

02 **Can I list all active P1 alerts using Opsgenie MCP?**

Yes, you can use 'list_alerts' to pull a comprehensive list of open issues. You simply need to phrase your query to filter for the severity level and status you care about.

03 **Is Opsgenie MCP better than just using the native Opsgenie UI?**

The key difference is context switching. Instead of leaving your IDE or chat app, your agent runs the commands for you. It brings the dashboard data directly into natural conversation.

04 **What information does 'get_who_is_on_call' provide?**

It gives you the name and contact details of the current on-call responder for a specific schedule, plus when their shift ends. This is crucial for routing issues correctly.

05 **How do I create an incident using Opsgenie MCP?**

You use the 'create_incident' tool by providing the necessary scope and priority details. The agent handles forming the record, making sure it gets logged correctly in the system.

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 `"opsgenie": { "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

Opsgenie 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 Opsgenie. 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	Opsgenie MCP
Server ID	`019d75ea-779e-701b-89b0-8744ecfb54f5`
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/opsgenie.