MCP SERVER | NO CODE | CLOUD HOSTED

ntfy (Push Notifications) MCP

Get instant alerts on your phone or desktop.

ntfy (Push Notifications) lets your AI agent send and retrieve real-time alerts across different topics directly to any device. Send targeted messages with custom titles, priorities, tags, or schedule future notifications using just natural conversation.

F Quality Score 12.97/100

push-notifications | pub-sub | real-time-alerts | http-api | cross-platform | messaging

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/ntfy-push-notifications

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.

ntfy (Push Notifications) MCP

2 tools available
Cloud-hosted on Vinkius

This MCP connects your chat agent to a robust pub/sub service, letting you manage critical system alerts and development updates in real time. You can publish specific push notifications to dedicated topics, automatically creating the channel if it doesn't exist. These messages support rich formatting like Markdown, plus you can customize everything—setting priority levels, adding tags, or including clickable links. If you need historical context, your agent can also pull cached logs from a topic so you know what happened minutes ago without checking a separate dashboard. Plus, you don't have to be online now; you can even schedule notifications for the future. Connecting this through Vinkius means your AI client handles all the complexity, bridging natural language requests with reliable push delivery across phones and desktops.

Core Capabilities

01 — Send immediate alerts

You send a targeted notification to a specific topic using defined titles, priorities, tags, or rich content.

02 — Check historical message logs

Your agent retrieves cached messages from a topic, allowing you to review past alerts and system events instantly.

03 — Schedule future reminders

You set up a notification to send automatically at a specified time later on.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/ntfy-push-notifications** — connect your AI agent in three steps.

01 Subscribe this MCP and provide your ntfy instance URL (e.g., https://ntfy.sh) and any access token.

02 Your AI agent interprets a natural language request, determining if it needs to send an alert or check logs for a specific topic.

03 The system sends the structured command—whether publishing a message or polling cached data—and delivers the resulting real-time notification or log list.

The bottom line is your AI agent handles connecting to the service, structuring the payload (priority, tags, content), and delivering the alert to your desired endpoint.

Built For

This MCP serves anyone whose job involves reacting to asynchronous system events. It's for the DevOps engineer tired of constantly checking dashboard metrics or the developer who needs immediate, actionable debugging information delivered right to their phone.

DevOps Engineer

Automating deployment alerts and monitoring status updates so they arrive instantly on your mobile device without needing a dedicated chat window.

Software Developer

Sending complex debugging output or tracking long-running build process completion messages directly from the terminal to review later.

System Administrator

Creating custom workflows that notify a small team about critical system changes, like database backups finishing successfully.

What Changes When You Connect

01 Immediate visibility: You don't have to keep tabs open. Send a notification, and it pops up right on your device when the event happens.

02 Contextual history: Need to know what triggered an alert? Use the `poll_messages` tool to check cached logs from a topic, giving you instant context without manual effort.

03 Structured communication: You control more than just the text. Publish notifications with custom titles, priority levels (like high or medium), and specific tags for easy filtering.

04 Advanced reliability: Send rich content that includes clickable URLs, custom icons, and files, making alerts actionable right away.

05 Time management: Forget to check on something? Use the scheduling features to set a reminder now that arrives automatically at a time you designate later.

Real-World Applications

Monitoring Critical Deployments

A DevOps engineer finishes a major deployment. Instead of waiting for an email, they ask their AI agent to use the `publish_message` tool, sending a high-priority notification to the 'production-alerts' topic. They get an instant phone alert saying, 'Deployment successful,' letting them know immediately without checking Slack or monitoring dashboards.

Debugging CI/CD Pipelines

A developer needs to track a long build process. They ask their agent to monitor the 'build-logs' topic and use `poll_messages` every few minutes. The agent reports finding specific log messages, like 'Build #402 failed at step 5,' letting them know exactly where the failure occurred.

Team Reminders for Standups

A manager needs to remind a team about a recurring meeting. They use their agent to schedule a notification using `publish_message` for 'meeting-reminders' in one hour, ensuring every relevant person gets the alert exactly when they need it.

System Health Checks

A system admin wants to know if the database backup completed successfully. They ask their agent to send a message via `publish_message` with a specific checkmark tag and high priority, guaranteeing that even if they are away from their desk, the critical status is delivered.

Patterns to Avoid

Over-relying on simple text messages

✗ AVOID

Manually checking multiple web dashboards and relying only on plain text alerts that lack context or priority.

✓ INSTEAD

Use this MCP to send structured notifications. With `publish_message`, you can add specific titles, priorities, and rich content, so the alert tells you everything you need right away.

Forgetting historical context

✗ AVOID

An issue pops up, but nobody remembers what happened five minutes ago because they only saw the initial error message.

✓ INSTEAD

Don't just wait for new alerts. Use `poll_messages` to retrieve cached logs from the topic. This instantly gives you a history of events leading up to the current problem.

Ignoring time constraints

✗ AVOID

Needing a reminder, but forgetting to set an alarm or write it down until it's too late.

✓ INSTEAD

Use the scheduling feature. Your agent handles setting up `publish_message` for a future date and time, making sure you get the alert exactly when planned.

The Right Fit

You should use this MCP if your core problem is about *asynchronous communication*—meaning critical status updates happen outside of a direct conversation or immediate workflow. This is perfect for alerts: 'The build finished,' or 'The database hit 90% capacity.' Don't use it, however, if you simply need to pass data between two connected systems that are always active; in those cases, an API webhook might be better. Use this MCP when the communication needs to bridge a system event (like a successful backup) and a human user who is potentially disconnected or busy with other tasks. If your goal is just to store structured data for later retrieval by your agent, you might need a different database connector instead of one focused on real-time messaging.

The Headache of Missing Critical Status Updates

Right now, status updates are scattered. You have to check the CI/CD dashboard for build failures, then switch tabs to Slack to see deployment messages, and finally open your phone's email app for database reports. It’s a constant cycle of clicking through different interfaces just to piece together if everything is green or red.

With this MCP, you tell your agent the condition—like 'Notify me when the production service hits 95% CPU.' The system handles the listening and delivery. You get one clean, actionable notification pushed directly to your device, no matter what application you're currently focused on.

Getting Real-Time Alerts with ntfy (Push Notifications)

You don't have to worry about setting up webhooks or maintaining multiple integrations. You simply tell your agent, 'Alert me when X happens.' The MCP handles routing the message, applying tags like 'CRITICAL,' and ensuring it gets delivered using `publish_message`.

The result is a single source of truth for system status. Your focus stays on solving problems, not managing where the alerts are coming from.

ntfy (Push Notifications) 2 Tools

These tools allow you to programmatically publish new notifications or retrieve historical logs from specific alert topics.

#	TOOL	DESCRIPTION
01	`poll_messages`	Retrieves previous alert messages from a specified topic so you can see the history of events.
02	`publish_message`	Sends an immediate push notification to any designated ntfy topic, creating it if needed.

See It in Action

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

U Send a high priority notification to the 'server-alerts' topic saying 'Database backup completed successfully' with a checkmark tag.

I've published the message to the 'server-alerts' topic with priority 4 and the 'white_check_mark' tag. The notification has been sent successfully.

U Poll the last 5 minutes of messages from the 'dev-updates' topic.

I've retrieved the cached messages for 'dev-updates' from the last 5 minutes. I found 2 messages: 'Build #402 started' and 'Build #402 finished'.

U Schedule a notification to the 'reminders' topic in 1 hour saying 'Time for the standup meeting!'.

I've scheduled your notification for the 'reminders' topic. It will be delivered in approximately 1 hour.

Frequently Asked Questions

01 **How does ntfy (Push Notifications) MCP handle message history?**

It uses the `poll_messages` tool to retrieve cached messages. This lets you look back at previous alerts on a specific topic without needing a dedicated logging dashboard.

02 **Can I send notifications with custom tags using ntfy (Push Notifications) MCP?**

Yes, when publishing a message, you can add structured metadata like tags. This lets your team filter alerts by type—for example, only showing 'Database' errors.

03 **Is the ntfy (Push Notifications) MCP suitable for scheduled reminders?**

Absolutely. You can schedule a notification to arrive at any future time using this MCP, perfect for automated meeting reminders or follow-up alerts.

04 **Does ntfy (Push Notifications) MCP support different priorities?**

Yes, you control the urgency of the message by setting a priority level. This helps your team visually distinguish between routine updates and critical failures instantly.

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 `"ntfy-push-notifications": { "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

ntfy (Push Notifications) 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 ntfy (Push Notifications). 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	ntfy (Push Notifications) MCP
Server ID	`019e38ca-5790-73b2-88af-1423f902dffc`
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/ntfy-push-notifications.

© 2026 Vinkius · vinkius.com · support@vinkius.com