MCP SERVER | NO CODE | CLOUD HOSTED

Telegram Bot Notifier MCP

Push alerts and status updates right to chat.

Telegram Bot Notifier MCP sends critical messages and alerts directly from your AI agent into specific Telegram chats, groups, or channels. This MCP gives any compatible client a secure, read-only way to push notifications—like build failures, status updates, or data reports—straight to your phone without needing complex webhooks or hosting setup.

A+ Quality Score 95.83/100

notifications | alerts | bot-api | messaging | real-time-alerts | mobile-notifications

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/telegram-bot-notifier

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.

Telegram Bot Notifier MCP

1 tools available
Cloud-hosted on Vinkius

Your AI agent needs more than just console output; it needs a reliable voice that hits people's phones. This MCP provides exactly that: a secure bridge to Telegram messaging. You simply provide the Bot token and the Chat ID, and your agent instantly gains the ability to drop alerts and messages into any designated chat. It’s designed for absolute containment—your agent can only push information; it cannot read private DMs or poll group history. This means you get notifications formatted with HTML or MarkdownV2—think bold text, code blocks, and links—so critical alerts don't look like plain garbage text. By connecting this through the Vinkius catalog, your AI client gets immediate access to a powerful messaging tool without any server management overhead. It’s the safest, simplest way to ensure that when an automated process fails or succeeds, someone on your team actually sees it right away.

Core Capabilities

01 — Send formatted alerts

The agent pushes messages containing HTML or MarkdownV2 formatting (bold, italics, code blocks) to a specified Telegram chat.

02 — Target specific chats

You direct the notification to an exact group chat, channel, or private user ID within Telegram.

03 — Handle automated status updates

The agent can deliver routine operational reports—like successful nightly builds or data pipeline completion notices—directly into a team communication channel.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/telegram-bot-notifier** — connect your AI agent in three steps.

01 You configure the MCP with your Bot Token and the specific Chat ID for the destination group or user.

02 Your AI agent calls the `send_telegram_message` tool, passing the text content and specifying if it needs HTML or MarkdownV2 formatting.

03 The MCP processes the request and delivers the formatted message instantly to Telegram, notifying your team.

The bottom line is you get an immediate, secure way for any automated process to talk to your team via a familiar messaging app.

Built For

This MCP is for the DevOps engineer who can't afford missed alerts at 2 AM. It's also for product managers and data scientists whose workflows depend on immediate feedback loops, needing to know when a script fails or an API call succeeds without manual dashboard refreshing.

DevOps Engineer

Needs to send automated build failure notifications or environment status reports directly into the team's dedicated chat channel.

Data Scientist

Uses it to push alerts when a data pipeline runs successfully, including summary metrics and links to generated dashboards.

System Administrator

Requires the ability to send critical system outage notices or maintenance window reminders immediately across multiple operational groups.

What Changes When You Connect

01 Instant visibility means less time checking dashboards. When a workflow finishes or breaks, the agent uses `send_telegram_message` to notify your team immediately in Telegram.

02 It’s built for safety. The MCP only allows messages to be pushed out; it never reads private DMs or polls chat history, so you don't sacrifice security for convenience.

03 Formatting is native. Don't stick with plain text. You can use `send_telegram_message` and specify HTML or MarkdownV2 to make your critical alerts look professional—with bold headers and code blocks.

04 No infrastructure headache. Forget setting up webhooks, managing long-polling daemons, or paying for a dedicated server just for notifications. This MCP handles the connection using only a Bot Token and Chat ID.

05 Supports all workflows. Whether you're running tests via an IDE agent or executing complex pipelines through a specialized client like Cursor or Windsurf, this MCP ensures results land where people look first.

Real-World Applications

The nightly build failed

A DevOps engineer's CI/CD pipeline breaks overnight. Instead of waiting for a dashboard email that might get lost, the agent automatically calls `send_telegram_message` to push an immediate, formatted alert into the #devops channel, including logs and the failure point.

Data quality check completed

A data scientist runs a complex validation job. Once it passes all checks, the agent uses this MCP to send a summary message containing key metrics (formatted in Markdown) directly into the project chat channel for immediate sign-off.

API rate limit hit

The automated integration attempts too many calls and hits an API quota. The agent immediately sends a formatted alert detailing which service failed and why, alerting the on-call team before they even notice the slowdown.

User onboarding sequence complete

A new user completes all setup steps. The workflow agent uses this MCP to send a congratulatory message, including links to documentation, directly into the dedicated client channel for follow-up.

Patterns to Avoid

Trying to read chat history

✗ AVOID

Assuming that because Telegram is a messaging service, the agent can be pointed at old messages or private DMs for context.

✓ INSTEAD

This MCP only supports pushing data. To get historical context, you must use a different tool designed specifically for reading message archives, as this one does not read your chat.

Relying on email notifications

✗ AVOID

Setting up every workflow to trigger an email notification when something happens, leading to high volume and missed alerts.

✓ INSTEAD

Use `send_telegram_message` instead. It delivers highly visible, formatted alerts right into a dedicated team channel where everyone is already looking.

Writing plain text only

✗ AVOID

Sending an urgent alert that just says 'FAIL' without any details or formatting, making it impossible to triage quickly.

✓ INSTEAD

Always specify the `parseMode` parameter when using `send_telegram_message`. Use MarkdownV2 for structured, readable alerts.

The Right Fit

Use this MCP if your primary requirement is reliable, immediate notification delivery from an automated process into a chat platform. You need to signal status changes (success/failure) or report data results without requiring continuous human monitoring of dashboards. Don't use it if you need the AI agent to *read* messages, poll for updates, or interact with user input other than what is being pushed by your workflow. For reading data, look at document indexer MCPs. If you only want to send simple plain text without

formatting, this works fine, but you lose the ability to make alerts clear and actionable.

Today's notification process feels scattered and delayed.

Right now, when a critical job finishes or fails, people have to check four different places: the CI/CD dashboard, an email inbox that might be buried under lunch memos, the dedicated Slack channel (if someone remembers to post it), and sometimes, they just wait for a pager alert. It's manual, slow, and you're always worried about missing the one alert that matters.

With this MCP, all those endpoints collapse into one reliable stream. Your agent uses `send_telegram_message` to deliver a single, formatted notification directly into your team chat. You get the information instantly, structured with bold text and clear failure codes—no more checking four tabs.

Send messages via Telegram Bot Notifier MCP

You eliminate the need to set up complex webhook listeners or manage dedicated notification microservices. You don't have to worry about different platforms requiring unique, bespoke APIs for simple status updates.

The result is an immediate communication layer that simply works across any MCP-compatible client, making your agent smarter by giving it a direct voice into the team conversation.

Telegram Bot Notifier: 1 Tool Available

This single tool allows you to send messages, status reports, and critical alerts through a dedicated bot connection in Telegram.

#	TOOL	DESCRIPTION
01	`send_telegram_message`	Sends a message, optionally using HTML or MarkdownV2 formatting, to a designated Telegram chat, group, or channel via your bot.

See It in Action

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

U Send a Telegram message saying the nightly build failed.

I've successfully sent the failure notification to your Telegram chat.

U Send an alert to Telegram. Format it in HTML with bold text for 'CRITICAL'.

The HTML formatted alert has been delivered to your Telegram chat.

Frequently Asked Questions

01 **Does Telegram Bot Notifier MCP read my private DMs?**

No. This MCP is strictly outbound and contained. It can only push messages; it cannot access your chat history, private direct messages, or read any content that hasn't been explicitly provided by the calling agent.

02 **How do I use send_telegram_message for a group of people?**

You simply need to provide the Chat ID of your designated Telegram group. As long as the Bot has permission in that group, the message will land there automatically.

03 **Can I format my alerts using send_telegram_message?**

Yes, absolutely. You can specify the `parseMode` parameter when sending the message to use either HTML or MarkdownV2 for rich formatting, making your alerts easy to read.

04 **Is this MCP reliable for mission-critical alerts?**

It is designed as a zero-friction bridge. Because it requires only a Bot Token and Chat ID, the deployment complexity drops away, making it ideal for systems that need to be highly dependable.

05 What if I want to send multiple alerts? Do I need multiple tools?

No. You just call `send_telegram_message` once per alert, passing the required content and formatting parameters in the single tool call.

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 `"telegram-bot-notifier": { "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

Telegram Bot Notifier 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 Telegram Bot Notifier. 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	Telegram Bot Notifier MCP
Server ID	`019e38f8-b058-70fb-936a-86577d989365`
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/telegram-bot-notifier.