MCP SERVER | NO CODE | CLOUD HOSTED

# Checkly MCP for AI Agents

## Monitor API Uptime and Web App Performance Metrics

Checkly lets your AI agent take full control of application monitoring and synthetic testing. You can track API uptime, view detailed performance metrics for web apps, and manually trigger checks—all through natural conversation with no dashboard required.

F Quality Score 3.6/100

api-monitoring | synthetic-testing | uptime-tracking | end-to-end-testing | performance-monitoring

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

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.

# Checkly MCP

8 tools available
Cloud-hosted on Vinkius

Need to know if your APIs are up? Checkly connects your monitoring stack directly into any AI client. Instead of logging into a separate dashboard every time you need an update, you just ask your agent. It instantly retrieves the status of all your API and browser monitors. You can audit alert configurations (for Slack or PagerDuty) without clicking through menus, check historical performance metrics for specific flows, or even force a manual run to verify system health on demand.

This capability means your team gets immediate visibility into application reliability right where they're already working. If you manage infrastructure monitoring, this MCP is essential for keeping your development process fast and responsive. It works alongside the full catalog of integrations found on Vinkius, making it a single point of truth for uptime and performance data.

## Core Capabilities

**01 — View all monitors and group details**

List every API or browser monitor you have configured and see which groups they belong to.

**02 — Get detailed check information**

Retrieve specific data points for any individual monitor, including configuration details.

**03 — Check performance metrics over time**

Pull historical response times and performance scores for a specified check to spot trends or regressions.

**04 — Manually force a test run**

Instantly trigger any monitor to execute its full sequence of tests, verifying current system health on demand.

**05 — Audit alert channels**

List every external channel (like Slack or Email) that is currently configured to receive alerts when a check fails.

## One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to this MCP and provide your Checkly API Key and Account ID.

02 Connect the credentials within any compatible AI client (Claude, Cursor, etc.).

03 Ask your agent a natural language question like, 'What's the status of my checkout flow?' or 'List all failing monitors.' The agent then executes the necessary checks and delivers the results.

The bottom line is, you manage complex monitoring tasks using simple conversation instead of navigating multiple web interfaces.

## Built For

This MCP is for Ops Engineers, SREs, and QA Automation specialists. It targets the pain point of context switching—the constant need to jump between dashboards (Checkly, Jira, Slack) just to verify if an API or web flow is broken.

**DevOps Engineer**

Runs manual checks on critical APIs after deployment and audits alert channels without leaving their terminal.

**Site Reliability Engineer (SRE)**

Reviews historical performance metrics for major services to prove system stability or pinpoint gradual degradation.

**Backend Developer**

Verifies API health and gets immediate feedback on check results straight from their chat interface during development cycles.

## What Changes When You Connect

01 Stop context switching. Instead of opening multiple dashboards to check status, you ask your agent to list all monitors or get specific performance data instantly.

02 Prove system stability with historical metrics. Use the `get_check_performance_metrics` tool to retrieve average response times and identify slow-down trends without manual chart digging.

03 Validate changes on demand. Need an immediate status check? The agent executes a test run using `trigger_check_run`, giving you real-time feedback on system health right in the chat.

04 Maintain visibility across your stack. Easily audit every configured alert channel with `list_checkly_alert_channels` to ensure nothing is missed when things break.

05 Understand your setup at a glance. Quickly use `list_checkly_checks` and `list_check_groups` to get an overview of all monitoring assets without clicking through the UI.

## Real-World Applications

**Debugging a new API endpoint**

A developer just merged a payment service change. Instead of manually waiting for the scheduled check, they ask their agent to run an immediate test via `trigger_check_run`. The agent reports back that the transaction monitor failed and provides detailed error messages.

**Quarterly SRE audit**

The SRE needs proof of uptime. They prompt the agent for performance metrics on the main checkout flow (`get_check_performance_metrics`). The agent compiles a 30-day report showing consistent low latency, passing the quarterly review.

**Onboarding new team members**

A junior engineer needs to know what's being monitored. They ask the agent to list all active monitors (`list_checkly_checks`), getting a complete overview of every API and browser check in seconds.

**Reviewing background tasks**

The team suspects a nightly job failed. Instead of checking logs, they ask the agent to list all heartbeats (`list_checkly_heartbeats`), confirming if the critical database cleanup cron job ran successfully within its expected window.

# Patterns to Avoid

### Forgetting historical context

✗ AVOID

A developer only checks the current status of a monitor, assuming 'green' means everything is fine. They miss that performance has been trending downward for weeks.

✓ INSTEAD

Always ask your agent to retrieve performance metrics using `get_check_performance_metrics`. This gives you the necessary historical context to spot gradual degradation before it becomes an outage.

### Over-relying on scheduled checks

✗ AVOID

Waiting for a critical deployment check to run at its next interval, which could be hours away. This leaves a gap in immediate validation.

✓ INSTEAD

When validating a change, use the `trigger_check_run` tool. It forces an instant test execution, giving you real-time verification that mimics manual testing without delay.

### Assuming one monitor covers everything

✗ AVOID

Checking only the 'API Gateway' monitor and assuming all internal services are fine. The actual failure might be in a secondary, unlisted service.

✓ INSTEAD

Run `list_checkly_checks` first to get a full inventory of all monitors. Then use that list to audit every necessary component.

## The Right Fit

Use this MCP if your team needs immediate, conversational access to monitoring data, especially when you need to validate changes or check historical trends without opening the dashboard. This is ideal for SREs and DevOps who live in chat tools. Don't use it if your primary goal is managing billing settings or user permissions; those tasks belong to dedicated account management systems. If all you need is a simple list of available monitors, `list_checkly_checks` handles that efficiently. However, remember this MCP doesn't provide remediation—it only reports the problem using tools like `get_check_performance_metrics`. You still have to fix the underlying code.

## Checkly Monitoring: Verifying API Uptime and Performance

Today, checking application health requires a painful ritual. Developers must copy credentials, log into Checkly's dashboard, navigate to different monitor groups, and then manually check the status of every single endpoint or flow. If you need an immediate status update during a deployment window, you waste time clicking through tabs just to answer one question: Is it working?

With this MCP, that whole process vanishes. You simply ask your agent for the 'Checkout Flow' status. The system executes the check and returns a clear pass/fail result immediately in the chat. It turns multi-step dashboard checks into a single conversational query.

## Checkly Monitoring: Tracking Background Jobs and Alerts

Manually tracking background tasks is tedious, especially when you have multiple cron jobs or scheduled database cleanups. You'd have to remember which job was supposed to run, check its status page, and then cross-reference that against the alert settings to make sure notifications were even configured for failure.

Now, you can ask your agent to list all heartbeat monitors ( `list_checkly_heartbeats` ) and simultaneously audit which external channels are responsible for alerts. This gives you a complete picture of reliability—the job status plus who gets notified when it fails.

# Checkly: 8 Tools for API Monitoring and Uptime Testing

Use these tools to list monitors, check specific performance metrics, trigger immediate test runs, and audit the entire monitoring setup.

| # | TOOL | DESCRIPTION |
|---|---|---|
| 01 | `get_checkly_account_info` | Retrieves core organizational metadata for the Checkly account. |
| 02 | `get_check_performance_metrics` | Pulls historical performance data and response times for a specific monitor. |
| 03 | `get_check_details` | Fetches detailed configuration information about an individual check or monitor. |
| 04 | `list_checkly_alert_channels` | Lists all active external channels, such as Slack and PagerDuty, receiving alerts for the account. |
| 05 | `list_checkly_checks` | Retrieves a list of every configured API and browser monitor in the system. |
| 06 | `list_check_groups` | Lists the logical groupings that contain multiple related checks or monitors. |
| 07 | `list_checkly_heartbeats` | Shows all configured heartbeat (cron) jobs and their last run status. |
| 08 | `trigger_check_run` | Forces a specified check to execute immediately, simulating an instant system test. |

# See It in Action

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

**U** Show me the performance history for the main API endpoint over the last 7 days.

**API Performance Report: /api/main**

| METRIC | LAST 24 HRS | WEEK AVG | TREND |
|---|---|---|---|
| Average Response Time | 350ms | 380ms | ↑ (Slight Increase) |
| Peak Latency | 1.1s | 800ms | Stable |
| Uptime | 100% | 99.9% | Excellent |

*Note: Performance dipped slightly on Tuesday afternoon, coinciding with the database migration.*

**U** List all monitors and tell me if any are failing right now.

**Current Monitor Status**

- ✅ Main API Gateway: Passing (Last run 5m ago)
- ⚠️ User Auth Service: **Failing**. Last error: '401 Unauthorized'. Check details for required credentials.
- ✅ Shopping Cart Flow: Passing (Last run 2m ago)

I recommend investigating the User Auth Service failure immediately.

U What are my alert channels and what's the status of the nightly report sync?

**Alert Channels Configured:**

- Slack: #devops-alerts (Active)
- Email: sres@company.com (Active)
- PagerDuty: Primary (Active)

**Heartbeat Status:** The 'Nightly Report Sync' ran 1 hour ago and is reported as **Active**. Everything looks nominal.

# Frequently Asked Questions

01 **How can Checkly MCP help me monitor my API uptime?**

Checkly lets your agent check the current status of any monitored endpoint instantly. You don't have to open a dashboard; you just ask, and it tells you if the API is up or if there was an error.

02 **Does Checkly MCP track historical performance data?**

Yes, it does. You can ask for detailed performance metrics on specific checks to see how fast your APIs were over days or weeks, helping you find slow spots before they break completely.

03 **What if I need to manually test a new feature right now?**

You can use the MCP to trigger an immediate check run. This forces the system to run tests on demand, giving you real-time proof that your latest changes are working correctly.

04 **Is Checkly MCP better than just using my CI/CD pipeline for checks?**

While CI/CD pipelines are great for code testing, this MCP handles the continuous monitoring of live production endpoints and web flows. It gives you visibility into how the entire application performs in a real-world environment.

05 **How do I check if all my alert systems are set up correctly?**

The MCP allows you to list every configured alert channel, including Slack and PagerDuty. You can audit these settings from your chat interface to ensure that failure notifications go to the right people.

# 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 `"checkly": { "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

# Checkly 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 Checkly. 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 | Checkly MCP |
| Server ID | `019d756e-1b49-7084-83e8-cc3e01376864` |
| 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/checkly.