MCP SERVER | NO CODE | CLOUD HOSTED

Node-RED MCP

Control flows, nodes, and system health via conversation.

Node-RED MCP lets you control complex, event-driven workflows directly through your AI agent. Manage entire IoT pipelines and low-code applications without leaving the chat window. You can check system diagnostics, update flow logic, or install new node modules just by asking your agent.

A+ Quality Score 98.33/100

low-code | event-driven | workflow-automation | flow-management | system-diagnostics | api-orchestration

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/node-red

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.

Node-RED MCP

11 tools available
Cloud-hosted on Vinkius

Running automated systems means dealing with constantly changing flows, dependencies, and runtime errors. This MCP gives you a single connection point to manage those complexities in Node-RED. Instead of jumping between the editor GUI, terminal logs, and configuration files, you talk to your AI client. Your agent handles reading existing flow configurations, updating nodes, or checking system health. When you connect this capability via Vinkius, it becomes part of a larger catalog of tools, letting you orchestrate much more than just Node-RED. You can ask your agent to troubleshoot a memory spike and get the live diagnostics report instantly. This means complex, mission-critical automation is now controllable through simple conversation.

Core Capabilities

01 — Manage full flow configurations

Create new workflows, delete unused tabs, retrieve existing flows, or modify running logic.

02 — Handle node dependencies

List all installed modules, install missing packages, and remove obsolete nodes from your system.

03 — Check system health metrics

Get immediate diagnostics on the Node.js version, operating system details, and current memory usage of the runtime environment.

04 — Retrieve runtime settings

Fetch the active operational settings to understand what constraints or variables your workflow is running under.

One Click on Vinkius — From Prompt to Execution

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

01 First, subscribe to this MCP and provide your Node-RED Base URL and Access Token.

02 Next, authorize your preferred AI client—Claude, Cursor, or any other MCP-compatible agent—to access the connection.

03 Finally, tell your agent what you need. You can ask it to 'show me all active flows' or 'install the weather node,' and it executes the command instantly.

The bottom line is that your AI client routes complex flow and system commands through this MCP, giving you full control over the entire Node-RED environment from a single chat interface.

Built For

This is for the DevOps Engineer who needs to check runtime diagnostics at 3 AM without SSHing into multiple servers. It's for the Automation Developer tired of context switching between three different tools just to update a single flow, and for the IoT specialist managing edge devices remotely.

DevOps Engineer

Checks system diagnostics and runtime settings across multiple Node-RED instances without logging into individual machines.

IoT Developer

Monitors, modifies, and deploys edge computing flows directly through the conversation interface, eliminating terminal time sinks.

Automation Engineer

Deploys new flow logic or manages node dependencies by simply describing the change to their agent.

What Changes When You Connect

01 You check the overall state of your environment using get_diagnostics. You instantly know if a memory spike is due to OS limits or an internal application leak, without manually checking server logs.

02 Updating complex logic used to require multiple manual steps: copying JSON definitions and pasting them into different dashboards. Now you can use update_flow and tell your agent exactly what change needs implementing.

03 Dependency management gets simple. Instead of remembering the npm command or searching documentation, you just ask your agent to install a missing piece using install_node.

04 The ability to see everything with get_flows is huge. You immediately know which automation pathways exist and can use get_flow on any specific tab without navigating away from your chat interface.

05 You manage the entire lifecycle of an application: list dependencies with get_nodes, add new ones with install_node, and remove them later with remove_node—all conversationally controlled.

Real-World Applications

The critical flow failure

A smart home automation fails after a firmware update. Instead of logging into the server to check logs, you ask your agent to get_diagnostics. You see Node.js is running an outdated version and need to upgrade. Your agent reports the current setup (get_settings), allowing you to quickly determine the necessary patch.

Adding a new data source

You need your IoT dashboard to pull weather data, but the node isn't installed. You tell your agent, and it uses install_node. It confirms the installation, and then you use get_nodes to verify the module is now available for building the flow.

Auditing an old project

A new team member needs to understand a complex workflow built months ago. You ask your agent to get_flows, which lists all active tabs. You then use get_flow on specific IDs to show the entire structure without having to manually recreate or navigate the whole thing.

Refactoring an obsolete module

A project is winding down and you need to clean up dependencies. Instead of searching your file system, you use get_nodes to list everything installed. You then instruct the agent to remove_node for outdated modules.

Patterns to Avoid

Manual GUI Navigation

✗ AVOID

Logging into Node-RED via a web browser, clicking through menus, manually checking dependencies in different tabs, and then switching back to the terminal for diagnostics.

✓ INSTEAD

Use your agent. Ask it to get_diagnostics first. Then use get_flows to see all paths. Finally, ask it to install_node if anything is missing.

Vague Configuration Changes

✗ AVOID

Telling a teammate, 'Hey, can you just change the flow?' This forces them to manually find the correct tab, understand the JSON structure, and then apply the changes.

✓ INSTEAD

Use the specific tools. Tell your agent: 'Please use get_flow on ID X to review it, and then update_flow with this new logic.' It handles the specifics.

Unverified Dependencies

✗ AVOID

Assuming a necessary node is installed because you used that module last week. You only find out later that the dependency was removed or never properly set up.

✓ INSTEAD

Always start by calling get_nodes to verify every single required module is present before building or running any flow.

The Right Fit

Use this MCP if your primary bottleneck is context switching between configuration tools and live monitoring. You need an agent that can manage the entire lifecycle of a low-code application—from initial setup (add_flow) to runtime maintenance (get_diagnostics). Don't use it if you only need to read static data; for simple reads,

just asking 'what are my flows?' is enough. However, if your job requires *making* changes or confirming system health before making those changes, this MCP is essential because it gives your agent the necessary write access (update_flow, install_node) and read capabilities (get_diagnostics). If you only need to monitor a single endpoint without changing its configuration, other monitoring tools might suffice. But for deep flow control, stick with this.

The headache of manual workflow maintenance

Today, managing complex automation means constant context switching. You start in the Node-RED editor to build a flow; you jump to the terminal to check if the required node is installed and what memory usage looks like. Then you might switch to a dashboard view just to see the current operational settings before you even try to save a change.

With this MCP, your agent handles all those jumps. You tell it, 'Check the system health and update the data logging flow.' It runs get_diagnostics for you, verifies dependencies using get_nodes, and executes the necessary updates via update_flow. Everything happens in one conversation.

Node-RED MCP: Full Control Over Your Automation

You stop manually running `npm list` every time you suspect a dependency issue, and you stop copy-pasting flow structures between local files and the editor. Instead, your agent executes get_nodes and then uses install_node or remove_node when needed.

Your workflow logic is now fully controllable via chat commands. You don't just monitor; you actively manage the entire stack—from OS diagnostics to individual node dependencies.

Node-RED MCP with 11 Tools

These tools give your agent the ability to perform every action needed to build, monitor, and debug a Node-RED workflow environment.

#	TOOL	DESCRIPTION
01	`add_flow`	Creates a brand-new, empty workflow tab within your Node-RED instance.
02	`delete_flow`	Removes an entire flow configuration (tab) from the active workspace.
03	`get_diagnostics`	Retrieves a detailed report on system diagnostics, including OS version and memory usage.
04	`get_flow`	Pulls the configuration details for one specific workflow tab you want to inspect.
05	`get_flows`	Lists all currently active flow configurations available in your Node-RED environment.
06	`get_nodes`	Retrieves a list of every node module that is currently installed on the system.
07	`install_node`	Adds and installs a specified new node module package into your environment.
08	`remove_node`	Deletes an installed node module package from the system to clean up dependencies.
09	`set_flows`	Sets or switches the active flow configuration that Node-RED is currently focused on.
10	`get_settings`	Fetches all current runtime operational settings for the entire Node-RED instance.
11	`update_flow`	Modifies and saves changes to a specific, existing flow configuration tab.

See It in Action

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

U Show me all active flows in my Node-RED instance.

I've retrieved your flows. You have 3 main tabs: 'Home Automation' (ID: f123), 'Data Logger' (ID: f456), and 'Alerts' (ID: f789). Would you like to inspect a specific one?

U Check the system diagnostics and memory usage.

Diagnostics report: Node.js v18.16.0, OS: Linux. Memory usage is currently at 150MB of 512MB allocated. The system is healthy.

U Install the 'node-red-node-email' module.

Starting installation of 'node-red-node-email'... The module has been successfully installed and is now available in your palette.

Frequently Asked Questions

01 **How do I check if my Node-RED instance has enough memory using the Node-RED MCP?**

You run get_diagnostics. This tool provides a full report, including live details on OS and current memory usage, letting you know immediately if resources are constrained.

02 **Can I see all my existing workflows using the Node-RED MCP?**

Yes, use get_flows. This tool lists every active flow configuration available in your workspace, giving you a clear overview of your automation paths.

03 **What if I need to add a new node module? Which tool should I use with the Node-RED MCP?**

You use install_node. Simply tell your agent what package name needs adding, and it handles the installation process for you.

04 **How do I update an old flow without losing data using the Node-RED MCP?**

You can use update_flow. You provide the specific ID of the flow you want to change, and your agent applies the modifications while keeping the existing structure intact.

05 **Does the Node-RED MCP let me see what settings my instance is running?**

Yes, run get_settings. This tool fetches all current runtime operational parameters for the entire system, so you know exactly how your application behaves out of the box.

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 `"node-red": { "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

Node-RED 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 Node-RED. 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	Node-RED MCP
Server ID	`019e38c9-40ea-71e7-ab60-7936df3e2d0d`
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/node-red.