MCP SERVER | NO CODE | CLOUD HOSTED

Render MCP

Control your entire PaaS stack with conversation.

Render MCP connects your AI agent directly to your PaaS infrastructure, letting you manage services, deployments, and scaling from conversation. Instantly list all microservices, trigger cache-clearing deploys, check live logs, or suspend worker processes without touching a dashboard.

A+ Quality Score 100/100

paas | web-hosting | deployment-automation | cdn | serverless | infrastructure-management

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

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.

Render MCP

9 tools available
Cloud-hosted on Vinkius

Need to manage complex web services but hate clicking through dashboards? This MCP connects your AI agent straight into your Render account. You can treat your entire infrastructure—everything from static sites and databases to core APIs—like one single service that you talk to. Want to see what's running? Just ask, and the agent lists every microservice on your portfolio. Need a fix for an outage? Command it to list current deployments or check live logs instantly. Scale resources up or down with simple commands, suspending workers when they aren't needed or scaling them out right before peak traffic hits. This makes managing edge infrastructure feel like talking to a teammate who already has access and knows the system inside and out. When you connect this MCP via Vinkius, your agent gains total visibility into every piece of code running on your platform.

Core Capabilities

01 — Inventorying Services

List all active services, including web apps, databases, and static sites, across your entire Render account.

02 — Controlling Deployments

Safely kick off new deployments for specific services, or trigger a 'Clear Cache' build to ensure clean code testing.

03 — Managing Scaling State

Scale worker instances horizontally or instantly suspend and resume background workers based on demand.

04 — Checking Network Config

Verify custom domains attached to services and retrieve hidden environment variables for debugging.

05 — Monitoring Deployment History

Retrieve a full list of past deployments, allowing you to check logs or roll back to stable versions.

One Click on Vinkius — From Prompt to Execution

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

01 First, subscribe to this MCP and provide your Render API key.

02 Next, connect the credential to any AI-compatible client (Claude, Cursor, etc.)

03 Finally, tell your agent what needs fixing—like 'Scale the Core API up' or 'List all services'.

The bottom line is you command infrastructure changes using natural language prompts instead of clicking through menus.

Built For

This MCP solves problems for Ops Engineers, Full-Stack Developers, and Incident Responders who spend too much time navigating complex dashboards. It gives them direct command access to the platform's core functions.

DevOps Engineer

Manages scaling rules and routine deployment checks directly from their terminal without having to open a web UI.

Full-Stack Developer

Fetches staging environment variables on the fly while they are actively coding or debugging in their IDE.

Incident Responder

Quickly lists and verifies active deployment logs to diagnose and fix downtime states immediately.

What Changes When You Connect

01 Stop clicking through tabs. You can use list_services to pull up every microservice, static site, and database in a single command, giving you a full system map instantly.

02 Fix broken builds fast. Instead of guessing which deployment failed, you can check the history using list_deploys and verify logs without leaving your chat window.

03 Save money on downtime. Suspend services when they aren't needed for the weekend, then resume_service with a single command to cut costs immediately.

04 Test clean code easily. You don't need to wait for background processes; you can force a fresh build using create_deploy, even telling it to clear the cache.

05 Access sensitive settings instantly. Need to verify environment variables? list_env_vars gets those details without manual navigation through multiple security menus.

Real-World Applications

Troubleshooting a production API outage

The agent detects that the Core-API is failing. The engineer asks it to list_deploys, checks the last three builds, and realizes the cache was wrong. They then tell the MCP to run create_deploy with 'clear cache' enabled, solving the issue instantly.

Preparing for a low-traffic holiday period

The DevOps engineer tells their agent to suspend_service on all non-critical background workers and scale_service down on the main API worker. This cuts costs immediately, and they will resume everything when needed.

Debugging a staging environment variable

While coding in an unfamiliar module, the developer asks for the list_env_vars for that specific service. The MCP instantly provides the required keys, preventing them from having to ask a teammate for documentation.

Onboarding a new team member

A manager uses list_services to get an immediate inventory of all components—backend services, databases, and landing pages. This gives the new hire a complete picture without needing training on every single system.

Patterns to Avoid

Ignoring deployment history

✗ AVOID

A developer manually restarts a service because it's slow, but doesn't know if the latest code push was actually bad or if the issue is resource-related.

✓ INSTEAD

First, use list_deploys to review the build timeline. Then, check the logs for that specific deployment ID before making any changes.

Forgetting cache control

✗ AVOID

Triggering a new deploy (create_deploy) without specifying 'clear cache', which results in rolling out old, cached code and failing silently.

✓ INSTEAD

Always append the 'Clear Cache' flag to your create_deploy command when testing major changes.

Treating scaling as a single action

✗ AVOID

Trying to scale up the API workers, but forgetting that some services must be resumed first (resume_service) before they can accept traffic.

✓ INSTEAD

Check service status with get_service. If it's suspended, use resume_service before attempting scale_service.

The Right Fit

Use this MCP if your primary pain point is managing the operational state of multiple cloud services (PaaS). Specifically, you need to list every service via list_services, trigger controlled deployments using create_deploy, or adjust resource consumption by suspending/scaling. Don't use it if all you need is simple code generation or writing documentation; those are separate agent tasks. If your problem is purely about database query optimization (e.g., complex SQL joins), you should look for a specialized data-tool MCP instead.

Dashboard fatigue kills productivity.

Today, updating an infrastructure component means logging into Render, finding the right service ID, clicking through deployment tabs to check logs, then maybe jumping over to another tab just to scale up workers. It's a dozen clicks across four different screens, and you lose context trying to copy/paste IDs.

With this MCP, you tell your agent what needs doing: 'Scale the Core-API now.' The agent handles the entire sequence—checking service details, determining if suspension is needed, and executing the scale_service command—and gives you a simple confirmation. You get immediate operational control without leaving your conversation.

The Render MCP delivers total infrastructure visibility.

You don't have to manually run list_services every morning just to confirm everything is still running. You can ask the agent, and it immediately presents your full service portfolio. This saves you ten minutes of clicking through multiple dashboards.

It means your AI client isn't just a chatbot; it's an active control panel for your entire PaaS stack. It gives you immediate command access to every deployment lifecycle.

Render Alternative: 9 Tools for Cloud Operations

Use these tools to manage every aspect of your PaaS stack, including listing all services, triggering deploys, and adjusting resource scaling.

#	TOOL	DESCRIPTION
01	`create_deploy`	Triggers a new code deployment for a specific service.
02	`get_service`	Retrieves detailed information about a single Render service.
03	`list_custom_domains`	Lists all custom domains associated with a given service.
04	`list_deploys`	Retrieves the history of deployments for a specific service.
05	`list_env_vars`	Lists all environment variables configured for a service.
06	`list_services`	Queries and returns a comprehensive list of all services running on the account.
07	`resume_service`	Brings a suspended Render service back online.
08	`scale_service`	Increases or decreases the number of running instances for a service that supports scaling.
09	`suspend_service`	Temporarily shuts down and suspends a Render service to save resources.

See It in Action

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

U List all running backend and logic services on my Render account.

I've pulled the core array. You have 3 primary services active: 'Vinkius-Engine' (srv-124, running Postgres), 'Core-API' (srv-239, running Node), and 'Static-Landing' (srv-884, running Next.js). Shall we scale the Core API?

U My recent deploy on the API just failed. Start a fresh deploy now and clear the cache.

Understood. Firing deploy sequence for Service ID `srv-239`. The clear cache flag has been successfully appended. Deploy `dep-ab294c` has been initialized and is transitioning into 'build' phase.

U Scale our main API worker down to zero instances temporarily to cut costs for the weekend.

Action dispatched logically. Scaling service `srv-workers-8a` down to `0` instances. Operations have halted on the target. Let me know when you desire to reinstate operations.

Frequently Asked Questions

01 **How do I list all my services using the Render MCP?**

You ask the agent to run list_services. The MCP queries your entire account and returns a complete array of every microservice, web service, and database running under your account.

02 **Can I force a cache-clearing deploy with Render MCP?**

Yes, you can use the create_deploy tool. By specifying the 'Clear Cache' flag in your request to create_deploy, you ensure that the new deployment uses the freshest code and avoids old cached assets.

03 **What is the difference between suspend_service and scale_service?**

Suspension (suspend_service) completely halts a service's operation to save costs. Scaling (scale_service) adjusts the number of running instances while keeping the service operational.

04 **Can I check environment variables with Render MCP?**

Absolutely. The list_env_vars tool lets you query and retrieve all hidden environment variables for any specific service, which is crucial for debugging connections or keys.

05 **Does Render MCP help me debug a failed deployment?**

Yes. You can use list_deploys to view the history of deployments and get the necessary logs to pinpoint exactly when and why a specific version failed.

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 `"render-alternative": { "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

Render 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 Render. 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	Render MCP
Server ID	`019d8477-5a8d-706d-b9ab-fc65c60448ff`
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/render-alternative.