MCP SERVER | NO CODE | CLOUD HOSTED

Render MCP

Manage Your Entire Cloud Stack From Conversation.

Render MCP gives your agent direct control over your cloud infrastructure. Instead of opening the dashboard, you can use natural language prompts to list services, suspend compute resources to save costs, deploy hotfixes instantly from GitHub, or build brand-new backend services—all through conversation.

A+ Quality Score 100/100

cloud-hosting | deployment-automation | scaling | server-management | infrastructure-as-code | web-services

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

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

10 tools available
Cloud-hosted on Vinkius

Your AI client connects directly to Render's capabilities via this MCP. This changes how you manage your entire cloud stack; it turns standard chat into a powerful DevOps control center. You can ask your agent to inspect the status of every web endpoint, database, and cron job in your account. Need to save money? Tell it to suspend compute on inactive projects, or wake them up when needed. If a hotfix lands on GitHub, you don't need to click buttons; just prompt your AI client to trigger a full deployment for the service. You can even tell your agent to create brand-new services pointing to specific repository branches or completely delete obsolete staging environments. This level of infrastructure management is what makes Vinkius such a vital catalog, giving your agent deep operational control over complex systems.

Core Capabilities

01 — Assess current service status

Lists all active web apps, databases, and cron jobs to show their current running state.

02 — Control compute costs

Suspends or resumes services instantly, stopping billing cycles when the project isn't needed.

03 — Build new infrastructure

Automatically provisions entirely new cloud services linked to a specific GitHub repository branch.

04 — Force deployments and updates

Triggers an immediate, manual deployment for any service, even clearing the build cache if necessary.

05 — Review deployment history

Retrieves a chronological log of past deployment attempts for deep auditing.

06 — Remove resources

Permanently deletes specific services that are no longer required in the staging environment.

One Click on Vinkius — From Prompt to Execution

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

01 Install the Render platform extension module into your MCP connection.

02 Obtain and securely enter your personal Render API Key into the Vinkius configuration settings.

03 Use natural DevOps language in your chat, for example: "List my web services, then suspend the one named 'old-staging-app'."

The bottom line is you talk to your agent like a terminal command and it handles the complex API calls.

Built For

This MCP is for infrastructure engineers, backend developers, or technical founders who spend too much time clicking through dashboards. If manually managing deployments, scaling compute, or spinning up test environments eats into your workday, you need this.

DevOps Engineer

Manages service lifecycle by listing services and executing commands like suspend_service to control costs.

Backend Developer

Quickly spins up private background workers or new API endpoints for testing architectures without manual setup.

Startup Founder/CTO

Needs instant visibility into deployment history (list_deploys) and the ability to tear down old staging instances permanently via natural language prompts.

What Changes When You Connect

01 Save time on maintenance tasks. Instead of navigating multiple dashboards, you can ask your agent to list all services and immediately suspend costly non-production workers using `suspend_service`.

02 Control deployments without UI clicks. Need a hotfix deployed now? Simply tell your AI client to execute a fresh build pipeline via `trigger_deploy`, even clearing the previous cache.

03 Scale infrastructure on demand. Have an idea for a new service? Your agent provisions it automatically using `create_service`, pointing directly at the right GitHub repository branch.

04 Maintain cost control effortlessly. You can list all services and tell your agent to suspend compute usage, ensuring you only pay for what's actively running.

05 Audit deployments instantly. If something breaks, use your AI client to run `list_deploys` and get a clean history of the last few build attempts.

Real-World Applications

Debugging an intermittent staging failure

The founder noticed production was behaving strangely. Instead of guessing which resource was failing, they asked their agent to run `list_services` and check the status of all connected databases and web endpoints. The agent immediately flagged a specific service that needed attention.

Preparing for end-of-quarter cost savings

The ops engineer knows several staging environments are idle until next month. They prompt their agent to check the status of all non-production projects, then instruct it to `suspend_service` on every single one, stopping unnecessary billing overnight.

Testing a critical hotfix build

A developer pushes a fix that needs immediate testing. They ask their agent to force the deployment (`trigger_deploy`) and clear the cache for the main web app, ensuring they are testing against a completely clean code slate.

Building out a new microservice architecture

A team leader needs to spin up a whole new API worker. They tell their agent to use `create_service`, specifying the repository and branch, letting the AI handle the entire provisioning workflow.

Patterns to Avoid

Guessing which service is down

✗ AVOID

Typing vague commands like 'my app isn't working' or trying to manually recall a resource name from memory.

✓ INSTEAD

Start by running `list_services`. This shows every single web application, database, and cron job. Once you have the accurate list, target the specific service ID for further action.

Forgetting to clear cache on redeploy

✗ AVOID

Running a deployment trigger without telling your agent to bypass internal optimizations, leading to stale code or cached errors.

✓ INSTEAD

Always include instructions to force the build and clear the cache when calling `trigger_deploy`. This guarantees you test against a true clean slate.

Using the wrong branch source

✗ AVOID

Manually changing the Git branch setting in the UI, which is tedious and easy to misconfigure.

✓ INSTEAD

Tell your agent to run `update_service_branch` directly. You specify the service name and the desired branch label through plain language.

The Right Fit

Use this MCP if you need conversational control over the full lifecycle of cloud-hosted services, from provisioning (create_service) to decommissioning (delete_service). This is for operational management. Don't use it if your goal is merely code generation or writing documentation; those are better handled by specialized coding agents. If you just want to read a single service's status, `get_service` works. But if you need to manage *multiple* services—like listing them all or suspending groups of resources—you must use the collective commands like `list_services`. This MCP is your centralized command console; it's about action, not just information retrieval.

The Manual Pain Points of Cloud Ops

Right now, managing a growing cloud architecture means living in dashboards. You open the Render UI, click 'Services,' scroll through status checks, and if you need to pause billing on staging, you have to find that specific project and manually toggle it off. If something breaks, you're clicking tabs—checking deploy history here, then checking logs there.

With this MCP, you skip the whole UI process. You just tell your AI client: "List all services and suspend anything marked 'staging' except for my database." Your agent handles the navigation, the status checks, and the billing controls in a single conversational step.

Control Deployments with trigger_deploy

Before this, forcing an update meant logging into the service settings, selecting 'Deploy', picking the correct branch, and hoping nothing was missed. It was a multi-step process that always felt clunky.

Now you simply prompt your AI client: "Force deploy the main app and clear the cache." The agent executes `trigger_deploy` perfectly every time, giving you immediate control over your code's release cycle.

Render MCP: 10 Tools for Service Management

These tools allow your AI client to execute every core operation needed to build, deploy, and manage services within the Render cloud environment.

#	TOOL	DESCRIPTION
01	`create_service`	Creates a brand new Render service instance linked directly from a GitHub repository.
02	`delete_service`	Permanently removes an existing, unwanted Render service; this action cannot be undone.
03	`get_deploy`	Pulls specific details about a single recorded deployment run.
04	`get_service`	Retrieves full, detailed status information for one particular Render service.
05	`list_deploys`	Generates a list of all recent deployment attempts made to a specific service.
06	`list_services`	Lists every single resource in the account, including web apps, databases, and cron jobs.
07	`resume_service`	Restarts a service that was previously suspended, bringing it back online.
08	`suspend_service`	Stops a running service to halt compute usage and prevent billing charges.
09	`trigger_deploy`	Forces a manual rebuild and deployment of code for an existing service.
10	`update_service_branch`	Changes which specific branch in GitHub is used as the source for a running service.

See It in Action

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

U List my web services, then suspend the one named 'old-staging-app'.

I checked your services utilizing `list_services`. I found the matching instance ('old-staging-app', ID: srv-...a1s). By executing `suspend_service`, the application is now safely paused, terminating related compute billing operations momentarily until you need it again.

U Check the recent deployment history for my main front-end service (srv-xyz123).

Pulling data using the `list_deploys` interface on `srv-xyz123`, I can confirm the last 3 deployments occurred at [10:30 AM], [11:15 AM], and [1:00 PM]. The most recent deployment completed successfully and is live.

U Trigger a force deployment on service ID 'srv-backend88' and clear its build cache.

Deploy triggered using `trigger_deploy` with instructions pointing at `srv-backend88`. As requested, I bypassed internal optimizations establishing a true clean slate by clearing the previous build cache prior to cloning code.

Frequently Asked Questions

01 **Can I use Render MCP to check which services are running?**

Yes. Running list_services shows all connected resources—web apps, databases, and cron jobs—so you always know what's active in your account.

02 **How do I stop billing for a test environment using Render MCP?**

You use the suspend_service tool. You tell the agent to suspend the specific service name, which immediately halts compute usage and prevents related charges.

03 **What if I need to deploy code from an old branch?**

First, you must run update_service_branch to point the service to that historical branch. Then, use trigger_deploy to start the build process from that new source.

04 **Is delete_service permanent? Should I worry about it?**

Yes, this action is irreversible and permanently deletes the resource. Use it only when you are 100% certain you never need the service again.

05 **Can the AI clear the cache when triggering a deploy?**

Yes, absolutely. The tool `trigger_deploy` incorporates an optional variable explicitly created for cache management. You can command the agent: "Redeploy the web app named Node-Backend and bypass rendering cache."

06 **Which type of new services can the AI deploy using `create_service`?**

The MCP can provision and launch exactly three core resource forms utilizing GitHub repos: standard web services (`web_service`), private network-locked processes (`private_service`), and asynchronous task handlers (`background_worker`).

07 **Warning: Is there a confirmation before using `delete_service`?**

Since natural language agents can occasionally misinterpret parameters, invoking the text request explicitly will route straight to the Render API resulting in instantaneous destruction. Please ensure absolute clarity when pointing the AI logic toward deletion operations.

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": { "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	`019d75fe-6e27-707c-923b-7ed43046f89c`
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.