MCP SERVER | NO CODE | CLOUD HOSTED

Railway MCP

Control your entire cloud stack from conversation.

Railway MCP connects your AI agent directly to your live cloud infrastructure. Use it to manage projects, trigger deployments, restart services, and pull environment variables—all from your chat terminal without needing to open a web dashboard.

A+ Quality Score 100/100

deployment | cloud-hosting | environment-variables | infrastructure-as-code | serverless | automation

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

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.

Railway MCP

10 tools available
Cloud-hosted on Vinkius

You can run core DevOps tasks right through your agent's conversational interface. Instead of opening multiple browser tabs or running complex CLI commands every time you need an update, your AI client handles the plumbing. You can ask it to list all projects available on your account or pull up a service's runtime config instantly. Need to verify if a recent code push succeeded? Just ask for deployment history and get the status back. If a container is acting up, triggering a restart is as simple as asking. This level of deep access lets you manage everything from project creation to sensitive configuration variables without ever leaving your chat window. It's exactly what Vinkius delivers when it connects you to powerful services like Railway.

Core Capabilities

01 — Manage Project Lifecycles

Create or retrieve details for specific cloud projects across your account.

02 — Review Build Status and History

View a project's full deployment history, checking build statuses and rollout logs to ensure stability.

03 — Troubleshoot Live Services

Get the current runtime configuration for services or restart an unhealthy container instance on demand.

04 — Configure Environment Variables

Securely read, update, or sync sensitive environment keys across different service instances.

05 — Initiate Code Deployments

Force a new deployment run for any connected service immediately after writing code.

One Click on Vinkius — From Prompt to Execution

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

01 Enable the MCP integration within your agent client and provide your Railway API Token.

02 Give your AI client a simple prompt, like 'Show me all projects' or 'Restart service X'.

03 The agent calls the necessary tool, reads the live data from Railway, and provides you with a concise summary.

The bottom line is that it eliminates context switching by bringing your entire operational dashboard into natural conversation.

Built For

Anyone who spends their days jumping between dashboards, terminals, and Jira tickets will need this. It's for the Ops Engineer tired of clicking through five different consoles at 2 a.m., and the Fullstack Developer who just wants to push code without typing out three dozen CLI commands.

DevOps Engineer

They use this MCP constantly to verify environment variables, check deployment health using list_deployments, or quickly restart containers via restart_service.

Fullstack Developer

After wrapping up a feature branch, they trigger_deploy the code and then monitor the resulting logs and service instances without leaving their IDE.

System Administrator

They use this to audit existing projects using list_projects or run routine connectivity checks by getting service instance details.

What Changes When You Connect

01 Forget jumping through dashboards. You can check the status of a build and get deployment history using list_deployments, all in one prompt.

02 Need to cycle containers because of intermittent bugs? Instead of navigating menus, just ask the agent to run restart_service for that service name.

03 Project setup is fast. Use create_project or get_project to manage your core infrastructure details without ever touching a web form.

04 Secrets management gets easier. You can list_variables to check which configuration keys are active on any given service instance, saving you from guessing.

05 Deployment workflows accelerate. When development wraps up, simply trigger_deploy the changes and track the outcome—all through your chat interface.

Real-World Applications

Investigating a Broken Production Build

A developer noticed the staging environment was failing after a push. Instead of manually checking logs, they ask their agent to list_deployments for that service and confirm if the latest build status is 'FAILURE'. The agent immediately diagnoses the issue.

Scaling Up a Service After Success

A system administrator notices high latency on an old container. They ask the agent to get_service_instances, verify resource usage, and then run restart_service to cycle out the problematic instance immediately.

Setting Up a New Microservice

A fullstack developer needs a new backend service. They first ask list_projects to see what exists, use create_project for the new one, and then get_service_instances to verify its initial state.

Confirming Credentials Before Launch

Before pushing code live, a DevOps engineer uses list_variables on the target service. This confirms that all necessary API keys are present and correctly configured before they trigger_deploy the final version.

Patterns to Avoid

Manual Dashboard Checks

✗ AVOID

Logging into the Railway console, navigating to 'Deployments', finding the specific service, then checking the logs and status manually. This takes five minutes.

✓ INSTEAD

Just ask your agent to list_deployments for that service name. The agent handles all the navigation and data retrieval in seconds.

Guessing Variables

✗ AVOID

Assuming a required database URI is set correctly because it worked last month, only to find out later during runtime that the environment variable was never updated.

✓ INSTEAD

Use list_variables. It forces an audit of all active configuration keys for that service instance, giving you the definitive source of truth.

Over-relying on CLI Commands

✗ AVOID

Running `railway deploy` in a terminal and then having to switch contexts to check if it actually succeeded or what the error was.

✓ INSTEAD

Use trigger_deploy, and immediately follow up with list_deployments. You get confirmation of the run *and* the status update without leaving your chat.

The Right Fit

You should use this MCP if your primary workflow involves managing the state or configuration of live cloud infrastructure—things like deployments, services, or environment variables. If you need to check 'what is running' or 'did that code actually make it live,' this toolset is mandatory. However, don't use this if your goal is pure application logic development; if you just want to write a new feature or refactor existing code without deploying it, stick to your usual coding tools. If you need data from an external source—say, fetching customer records from a CRM—this MCP won't help; you'll need a different connection type.

Use this when the problem is 'How do I programmatically audit or change my infrastructure?' Don't use it if the problem is 'I need to write an email.'

The Operational Dashboard Overload

Today, keeping track of a deployment can be a nightmare. You have to open the cloud dashboard, navigate through project settings, click into service logs, and then check the deployment history page. It's a cycle of tabs, clicks, and copy-pasting status codes just to confirm if your code made it live.

With this MCP, that process vanishes. You simply ask your agent to list_deployments for the target service. The AI pulls all the necessary status information—build success, rollout logs, current environment details—and hands it back in plain language. It’s instant confirmation.

Project and Service Management with Railway MCP

The most tedious parts are the initial setup checks. Do you need to know what projects exist? You have to go to a list view. Is a service running properly? You're stuck checking the 'Instance Details'. And if it's broken, restarting requires going back and forth between menus.

Now, your agent handles all those manual steps. With tools like get_project or list_projects, you can audit your entire setup from one chat window. It’s not just reading; it’s controlling the state of your whole cloud stack.

Railway: 10 Tools for DevOps Operations

These ten tools give you programmatic control over every aspect of your Railway infrastructure—from creating new projects to managing runtime variables.

#	TOOL	DESCRIPTION
01	`create_project`	Creates a brand new cloud project within your Railway account.
02	`delete_project`	Permanently removes an entire cloud project. Be careful; this action cannot be undone.
03	`get_project`	Retrieves detailed information about a single, specific cloud project.
04	`get_service_instances`	Fetches the current runtime configuration and details for an operational service.
05	`list_deployments`	Displays a list of all past and present deployments for a given project, environment, and service.
06	`list_projects`	Retrieves a comprehensive list of every Railway project your token has access to.
07	`list_variables`	Lists all current environment variables associated with a service, helping you check configuration keys.
08	`restart_service`	Forcibly restarts a running service instance when it's behaving poorly or needs a fresh container cycle.
09	`trigger_deploy`	Initiates and queues up an entirely new deployment run for the specified service.
10	`whoami`	Retrieves your personal profile information associated with the Railway API token.

See It in Action

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

U List all active projects on my Railway account.

I retrieved your projects using `list_projects`. You currently have two active projects: 'ECommerce Backend' (ID: prj_xyz123) and 'Auth Service' (ID: prj_abc789).

U Restart the deployment for the ECommerce Backend service.

I executed `restart_service`. The service 'ECommerce Backend' successfully began recycling its containers. You can track the current startup health via the logs.

U Has the latest Production build finished yet?

Using the `get_deployments` query, I found the latest deployment status. The Production build on service 'sv_222' successfully completed 4 minutes ago and is currently in a 'SUCCESS' state.

Frequently Asked Questions

01 **How do I use Railway MCP to see all my projects?**

You run the list_projects tool. This tells you every project linked to your account, which is a great first step before working on any single service.

02 **Can I restart a service using the Railway MCP? If so, how?**

Yes, use the restart_service tool. Just provide the name of the running service instance, and the agent handles cycling its containers for you.

03 **What if my latest deployment failed? Should I use list_deployments?**

Absolutely. Use list_deployments. This tool gives you a full timeline of build attempts and provides key details about where the rollout stopped, helping pinpoint the failure point.

04 **Does Railway MCP let me change environment variables?**

Yes, it allows management via list_variables. You can check what values are currently set for a service instance before making any changes.

05 **Is the delete_project action irreversible with the Railway MCP?**

The listing specifies that delete_project is an irreversible action, so always confirm you're deleting the correct project before running it.

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 `"railway": { "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

Railway 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 Railway. 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	Railway MCP
Server ID	`019d75fc-5c1f-7056-9bd5-1a8a44204e9c`
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/railway.