MCP SERVER | NO CODE | CLOUD HOSTED

Qovery MCP

Manage deployments and environments conversationally

Qovery brings Kubernetes and cloud deployment management right into your chat client. List every environment, check application health, trigger zero-downtime restarts, or deploy a precise Git commit SHA—all without leaving your coding workflow.

A+ Quality Score 100/100

kubernetes | deployment-automation | infrastructure-management | microservices | cloud-ops

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

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.

Qovery MCP

10 tools available
Cloud-hosted on Vinkius

Stop context switching between your IDE and the cloud dashboard just to run basic devops tasks. This MCP connects your Qovery infrastructure directly into any AI client. Instead of navigating through multiple tabs to map out your Organization structure, check specific Project details, or verify an Environment's current status, you talk to your agent. You can ask it to list all applications in a staging environment, grab the replica count for a microservice, or restart pods with a single command. When you need a hotfix deployed—say, testing a new Git commit SHA against development sandboxes—it handles that targeted deployment instantly. Because this MCP lives on Vinkius, your agent gets access to thousands of other tools too. It's pure infrastructure control, conversationalized.

Core Capabilities

01 — Map all organizational deployments

List and navigate through every Qovery Organization, Project, and Environment associated with your account.

02 — Inspect application health

Retrieve real-time status details for specific applications, including replica counts and auto-scaling settings.

03 — Cycle running pods

Execute a zero-downtime rolling restart on an application to refresh variables or cycle out old Kubernetes pods.

04 — Deploy specific commits

Force a targeted deployment of any precise Git commit SHA, perfect for localized hotfixes or feature testing.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to this MCP and supply your Qovery Organization API Token.

02 Your AI agent uses the token to read the structure of your cloud deployments (Organizations, Projects, Environments).

03 You prompt the agent with an action (e.g., 'Restart the Payment Gateway app in staging'), and it executes the command directly against Qovery.

The bottom line is you control your production environment using natural chat conversation, not complex CLI commands or web dashboards.

Built For

The platform engineer who hates context switching between their terminal and the cloud dashboard. It's for anyone whose job requires constant visibility into multiple staging environments, needing to verify a microservice is properly scaled and healthy without leaving their IDE.

Platform Engineer

Checking cluster configurations or verifying staging environment status while writing infrastructure-as-code.

Full-stack Developer

Pushing a quick fix to a branch, copying the commit SHA, and telling their agent to deploy it immediately to the dev sandbox for testing.

Engineering Lead

Checking if the latest mission-critical app is correctly scaled and healthy across multiple availability zones after a major release.

What Changes When You Connect

01 Stop clicking through dashboards. Instead of manually navigating to check application states, your agent gets the details instantly using `get_application`.

02 No more guessing if a fix deployed correctly. You can issue a fast-track deploy of any specific Git commit SHA directly via `deploy_application`.

03 Need to cycle pods for variable changes? Running an app restart is simple: just use `restart_application` and watch the agent handle the zero-downtime process.

04 Mapping your infrastructure used to be a multi-step nightmare. Now, you can list all organizations (`list_organizations`) and projects (`list_projects`) in a single chat query.

05 Your team saves time by eliminating context switching. You stay in your IDE while performing critical actions like listing environments (`list_environments`) or checking project details (`get_project`).

06 This MCP gives you full visibility, allowing you to verify the scaling and health of mission-critical apps across all zones without leaving your desk.

Real-World Applications

The Hotfix Scenario

A developer finds a bug in production. Instead of SSHing into a server or manually triggering a deployment, they copy the commit SHA and ask their agent to use `deploy_application` on the specific app in the Production environment. It's instant verification.

The Scaling Check

An engineering lead needs to confirm if the Payment Gateway is properly scaled after a peak traffic event. They ask their agent, which uses `list_applications` and `get_application`, to report on replica counts across all active environments.

The Environment Audit

A platform engineer needs to build a map of the company's services. They prompt their agent, which first uses `list_organizations` and then cascades down through `list_projects`, providing a full structural overview.

The Variable Refresh

A service needs its environment variables refreshed without downtime. The engineer simply tells the agent to run a restart using `restart_application` on the affected microservice, and it handles the rolling update automatically.

Patterns to Avoid

Manual Dashboard Refreshing

✗ AVOID

Opening the Qovery web dashboard multiple times to check if an environment status has updated or if a pod is stuck in 'pending' state.

✓ INSTEAD

Instead, ask your agent to use `get_environment` or `list_applications`. It fetches the current, definitive status and reports it back immediately.

Copying CLI Commands

✗ AVOID

Finding a deployment command in documentation, copying it into a terminal, and hoping all environment variables are set correctly.

✓ INSTEAD

Use the `deploy_application` tool. You just give your agent the commit SHA and tell it where to deploy; it handles the rest of the complex infrastructure logic.

Confusing Scope

✗ AVOID

Trying to list projects across unrelated organizations, leading to fragmented results or needing multiple API calls.

✓ INSTEAD

Start by listing all available organizations using `list_organizations`. Then, drill down project-by-project using `list_projects` to stay focused.

The Right Fit

Use this MCP if your current workflow requires constant interaction with cloud deployment details—checking status, restarting services, or triggering deployments across multiple environments. If you spend more than five minutes clicking through dashboards or writing complex CLI scripts just to get a basic status report, this is for you. Don't use it, however, if you only need general architectural knowledge; simply reading the documentation might suffice. Also, don't use it if your primary goal is code generation; that's better handled by pure coding assistants. This MCP is purely about operations and state management.

The Status Quo: Jumping Between Tabs to Check App Health

Today, checking if an application is healthy means jumping between the Qovery dashboard, the Kubernetes logs, and sometimes even a separate monitoring tool. You click on the Project, then select the Environment, then drill down to the Application's replica count, hoping you haven't missed any warning lights or stale status messages.

With this MCP, all that data is available in chat. Your agent pulls the exact application details using `get_application` and shows you the health report immediately. You get a single source of truth without ever leaving your conversation.

Qovery MCP: Deployments on Demand

The manual process for deploying a hotfix is painful: find the commit SHA, remember which branch it belongs to, navigate to the right environment, and click 'Deploy'. One wrong click or forgotten variable breaks your flow.

Now you just tell your agent to deploy using `deploy_application`. It handles the targeted deployment of that specific Git commit SHA. The control is immediate, accurate, and conversational.

Qovery: Deployment & Environment Ops (9 Tools)

Use these tools to list organizations, fetch application details, manage environment states, trigger deployments, and restart services across Qovery.

#	TOOL	DESCRIPTION
01	`get_environment`	Retrieves specific details and status information for one Qovery environment.
02	`get_organization`	Retrieves details for a specific Qovery organization
03	`list_organizations`	Retrieves a list of all Qovery organizations connected to the provided API token.
04	`list_projects`	Lists all projects that belong within a specific Qovery organization.
05	`restart_application`	Performs an immediate, zero-downtime rolling restart of the target application's pods.
06	`list_applications`	Shows all running applications deployed inside a specified environment.
07	`list_environments`	Lists all active environments (like Production or Staging) within a given project.
08	`deploy_application`	Triggers a fast deployment using a precise Git commit SHA to update the running code.
09	`get_application`	Fetches deep details about a single Qovery application instance.
10	`get_project`	Retrieves comprehensive details for one specific Qovery project.

See It in Action

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

U List all Qovery projects and tell me how many there are.

Using the `list_projects` tool on your configured organization. I retrieved 3 active projects:
1. 'E-commerce Monolith' (ID: prj-abc123x)
2. 'Payment Gateway' (ID: prj-xyz789y)
3. 'Internal Tools' (ID: prj-def456z)
Would you like to list the environments inside 'Payment Gateway'?

U Check the health and limits of the application in my staging environment.

I checked the apps inside environment 'env-stg987'. Found 'web-frontend'.
Limits:
- CPU: 2000m
- RAM: 1024MB
- Replicas: 2 (Auto-scaling off)
It is currently showing as 'HEALTHY' and actively running.

U Deploy commit 7a8f9b2 to the backend application immediately.

Sure, I triggered the deployment of commit SHA `7a8f9b2` for the application `backend-api` (id: app-bkn456). The deployment status is now 'DEPLOYING'. We can check back on this in a few minutes.

Frequently Asked Questions

01 **How do I list all my Qovery organizations with the Qovery MCP?**

You use the `list_organizations` tool to get a full catalog of all connected organizations. This is your first step when mapping out your entire infrastructure.

02 **Can I restart an application using the restart_application tool?**

Yes, running `restart_application` triggers a zero-downtime rolling restart across the specified pods and environment. It's perfect for refreshing variables without service interruption.

03 **What information does get_project provide about my Qovery project?**

The `get_project` tool retrieves all key details about a specific project, including its associated environments and applications. It gives you the full context for that development silo.

04 **How do I deploy a specific version using the Qovery MCP?**

Use `deploy_application` and pass in the precise Git commit SHA. This ensures you are deploying exactly what was tested, eliminating guesswork from manual deployments.

05 **Does this MCP help with checking environment status?**

Absolutely. You can use `get_environment` to retrieve detailed status checks for any specific environment in your project, confirming its readiness for deployment.

06 **How do I securely obtain my Qovery API Token?**

Sign in to your Qovery Console. Navigate to your **Organization Settings**, then to the **API Tokens** section. Click **Generate Token** (or Add). Give it a brief name, select the desired roles, and click Create. Copy the static string immediately as it won't be shown again, and paste it to authenticate.

07 **Can it restart specific microservices?**

Yes. Once you identify the `app_id` using the list components tools, you can instruct your agent to `restart_application`. This triggers a rolling restart exactly as if you clicked 'Restart' on the console. Traffic is routed seamlessly while pods re-initialize.

08 **What does deploy specific Git commit do?**

Normally, Qovery auto-deploys a branch. With `deploy_application` you can force Qovery to pull a specific commit ID (SHA) and deploy it immediately to an environment. This is perfect for hotfixes, effectively circumventing prolonged CI loops while ensuring zero downtime.

09 Is this tool safe to run on production?

Yes, but with caveats. Standard queries (like listing environments and getting stats) are entirely read-only. However, tools like `restart` and `deploy` are mutating operations. Always make sure you instruct your agent precisely and maintain manual approval checkpoints before executing deployment functions.

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 `"qovery": { "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

Qovery 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 Qovery. 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	Qovery MCP
Server ID	`019d75fb-5ff8-70bf-b45b-2d8e52fd1da4`
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/qovery.