MCP SERVER | NO CODE | CLOUD HOSTED

Porter PaaS MCP

Control your entire deployment stack from chat.

Porter PaaS lets you take full command of your Kubernetes infrastructure through natural conversation. Use your AI client to map organizational projects, check cluster health, manage environments, and force rollouts—all without opening a dashboard or running complex CLI commands. It's programmatic control over your entire deployed application stack.

A+ Quality Score 100/100

kubernetes | paas | container-orchestration | deployment-automation | cluster-management | infrastructure-as-code

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

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.

Porter PaaS MCP

10 tools available
Cloud-hosted on Vinkius

Managing complex cloud deployments usually means jumping between dashboards, remembering arcane command flags, and dealing with brittle YAML files. This MCP changes that. You connect your account once to Vinkius and give your AI client deep access to your Kubernetes infrastructure. Instead of running `kubectl` commands or navigating resource trees, you simply ask the question. Your agent handles the complex orchestration: listing out all organizational scopes, checking which environments are active, mapping web services, and even forcing a container mutation with a fresh image tag if something breaks.

It lets you audit an entire cluster's architecture—from high-level projects down to specific Helm charts behind your core databases. You can instruct the system to gracefully restart a hanging application pod or get precise metrics on CPU limits for a service, all from a single chat window. It's direct control over state, not just reading logs.

Core Capabilities

01 — Audit Organizational Scope

Retrieves the structural metadata that defines your major organizational projects.

02 — Map Cluster Boundaries

Inspects the core cloud credentials and definitions for an entire Kubernetes cluster.

03 — Inventory Deployed Services

Discovers all active applications, including those mapped to specific subdomains or custom routes.

04 — Force Application Rollbacks

Instructs the system to pull and deploy a specific image tag, overriding any current running container version.

05 — Manage Environments

Separates and lists out distinct isolation environments (like staging or pre-prod) within a single cluster.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to this MCP and provide your Porter API token.

02 Your AI client uses the credentials to access your cloud infrastructure data.

03 You interact using natural language, asking your agent to perform specific actions like listing projects or restarting pods.

The bottom line is you get programmatic control over complex cluster operations without ever leaving the chat window.

Built For

This MCP is for Ops Engineers who are tired of switching between dashboards and CLI tools. It's for Backend Developers needing quick, safe ways to roll back hotfixes. And it's perfect for Engineering Leads who need instant visibility into the entire resource map.

DevOps Engineer

Runs audits on cluster architectures or performs emergency restarts on services that are hanging up.

Backend Developer

Forcing a deployment of a specific image tag to test a hotfix, or rolling back an app if the new code breaks production.

Engineering Lead

Inspecting resource mapping and isolating distinct staging environments before greenlighting a major release.

What Changes When You Connect

01 Bypass dashboards and CLIs. Instead of running complex `kubectl` commands, you tell your agent to check the status or restart a pod, getting instant results in text form.

02 Safety during deployments is built-in. If an app crashes, use `restart_app` to cycle its replicas without having to modify the core code deployment tag first.

03 Full visibility across all resource types. Use `list_projects` and `get_project` to map out organizational boundaries instantly, giving you a high-level view of your infrastructure's scope.

04 Debugging hotfixes is fast. With `deploy_app_tag`, you can force an application to run a specific image tag immediately—perfect for emergency rollbacks without manual intervention.

05 Check everything required. You don't just see the web app; you use `list_helm_releases` to verify if crucial dependencies, like Postgres or Redis, actually installed correctly.

Real-World Applications

The weekend emergency rollback

A backend developer pushed a minor change that caused the main API to fail. They immediately use `deploy_app_tag` to force the application back to the stable image tag from last week, minimizing downtime and preventing manual effort.

Auditing architecture boundaries

An engineering lead needs to know how many separate testing environments exist before a major launch. They use `list_environments` to get an instant list of all isolated zones, confirming that the staging area is fully separated from production.

Troubleshooting a hanging worker

The queue processing service stops responding. An ops engineer uses `restart_app` on the specific worker app to force its pods to recycle, bringing the service back online without needing root access or SSHing into machines.

Mapping a new microservice

A team is launching a brand-new component. They use `list_apps` first to see what subdomains are currently active, then use `get_app` on the new service name to map out its exact resource requirements (CPU/RAM) before deployment.

Patterns to Avoid

Manual CLI Commands

✗ AVOID

Switching between your terminal, cloud console, and dashboard tabs just to confirm if the service is running on the correct environment.

✓ INSTEAD

Instead of running multiple `kubectl get` commands, ask your agent to call `list_environments`, then use `list_apps` to see what's deployed in that specific zone. It summarizes everything for you.

Forgetting Dependencies

✗ AVOID

The main application works, but the attached database service is failing because its Helm chart didn't install correctly.

✓ INSTEAD

Don't assume it worked. After deployment, call `list_helm_releases` to verify that every dependent third-party component finished installation successfully.

Debugging without context

✗ AVOID

Getting an error message but not knowing which cluster or project the failing service belongs to.

✓ INSTEAD

Start by calling `list_projects` and then use `get_project` with the resulting ID. This immediately scopes the problem, helping you target the right resources.

The Right Fit

Use this MCP if your primary need is deep operational control over a Kubernetes or PaaS deployment stack. You must be comfortable discussing concepts like namespaces, container tags, and resource quotas. If your job involves auditing cluster health, deploying hotfixes, or performing routine maintenance without opening a GUI, this is exactly what you want. Don't use it if you just need to retrieve simple, non-operational data; for instance, if you only needed general company metadata that isn't tied to an active cloud resource. For those cases, you might prefer a different MCP focused purely on data retrieval rather than state change.

The pain of the dashboard switcheroo

Today, fixing something usually means switching context five times: first to the main cloud console to check cluster status. Then into the specific application's dashboard to see logs. Next, you might open a separate terminal window just to run `kubectl` against the correct namespace. It's clicks, tabs, and copy-pasting—a terrible way to spend an hour.

With this MCP, that entire process collapses into one chat session. You tell your agent exactly what needs doing: 'The worker is hanging.' The agent calls the necessary functions like `restart_app`, handles the cloud authentication internally, and reports back the status change in plain language. It's just conversation.

Get instant deployment visibility with Porter PaaS.

Before this MCP, checking if a microservice was running on the correct version meant manually verifying container tags across multiple environments. You had to confirm the project ID via one tool and then check the application status in another.

Now, you ask your agent to look at the entire scope using `list_projects` and immediately get confirmation of all active apps, their associated environments, and resource mapping without ever leaving your chat window.

Porter PaaS: 10 Tools for Infrastructure Management

Use these functions to audit cluster status, deploy new versions, restart services, and map your entire application stack from a single chat interface.

#	TOOL	DESCRIPTION
01	`deploy_app_tag`	Forcefully updates an app's running container image to a specific version or tag, triggering a fresh deployment.
02	`get_app`	Retrieves deep metrics and resource limits for a specific deployed application.
03	`get_cluster`	Inspects the foundational cloud credentials used by a given Kubernetes cluster.
04	`get_project`	Pulls structural metadata related to an entire organizational project scope.
05	`list_apps`	Finds all running applications exposed under specific subdomains or custom target mappings.
06	`list_clusters`	Lists the available cloud definitions and boundaries that host your Kubernetes nodes.
07	`list_environments`	Extracts all distinct isolation environments (like staging or pre-prod) within a cluster.
08	`list_projects`	Retrieves the unique organizational identifiers for all major projects under your account.
09	`list_helm_releases`	Checks if third-party components, like databases or monitoring tools, successfully installed in a specific area.
10	`restart_app`	Instructs the system to cycle the application's container replicas without changing the underlying code.

See It in Action

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

U List all applications currently running in cluster ID 5 on the Production environment.

I found 4 applications bound to the cluster namespace. We have `core-api`, `cron-sync-worker`, `redis-cache`, and `web-frontend`. The core API is serving on a porter.run subdomain mapping. Would you like me to get architectural configs for `core-api`?

U The queue worker is completely hung. Please perform a forceful restart of the `async-worker` app.

Command received. I've sent the signal to recycle the pod replicas for `async-worker` across cluster 4. Kubernetes is now performing a rolling flush seamlessly. It will take roughly 15 seconds to stand out new instances and attach them.

U We just built a hotfix on main. Deploy the image tag `d83a1b1` strictly onto `portal-frontend`.

I've successfully forced the container mutation! The `portal-frontend` application was patched to image tag `d83a1b1`. The cloud runner is pulling the fresh digest and orchestrating a replacement. Check the frontend in 1 minute.

Frequently Asked Questions

01 **How do I use Porter PaaS MCP to check my cluster health?**

You can inspect the core credentials of a specific K8s Cluster by calling `get_cluster`. This gives you visibility into the foundational cloud definitions powering your deployments.

02 **Can I use Porter PaaS MCP to roll back an app version?**

Yes. Use the `deploy_app_tag` tool and provide the specific image tag or digest. This forces Kubernetes to pull that exact container version, overriding what's currently running.

03 **What if my service is hanging? How do I fix it with Porter PaaS MCP?**

Use `restart_app`. This instructs the system to cycle the application's pod replicas across the cluster. It's a non-disruptive way to clear connection leaks without changing the underlying code.

04 **Does Porter PaaS MCP help me see all my projects?**

Absolutely. Call `list_projects` to retrieve all the unique organizational IDs (the `projectId` arrays) that define your major operational scopes within AWS or GCP clusters.

05 **What is the difference between list_apps and list_clusters in Porter PaaS MCP?**

`list_clusters` gives you the boundaries of the physical cloud zones hosting your nodes. `list_apps` focuses on the logical layer, telling you which specific web services are running inside those clusters.

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 `"porter-paas": { "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

Porter PaaS 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 Porter PaaS. 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	Porter PaaS MCP
Server ID	`019d75f8-1a9e-71db-8f74-2dc8b72e10a4`
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/porter-paas.