

MCP SERVER

NO CODE

CLOUD HOSTED

# Deno Deploy MCP for AI Agents

Orchestrate edge computing deployments and review app logs by conversation

Deno Deploy MCP gives your AI agent control over your entire edge computing infrastructure. You can list, manage, and deploy serverless applications written in Deno, all through natural conversation with any compatible client.

**A+**

Quality Score 98.33/100

deno

serverless

edge-computing

deployment

javascript-runtime



# 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.

### 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.

### 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.

### 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**

Encryption at rest

**Ed25519**

PKI vault signatures

**24h TTL**

Ephemeral session keys

**V8 Isolate**

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.

### 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.

### 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.

### 06 — Honeypot Trap System

Phantom credentials are injected into isolated environments. If a honeypot 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.

**FinOps**

Cost tracking per tool, payload compression savings, budget optimization signals, and consumption trends.

**Firewall & DLP**

PII redaction activity, sensitive data protection counters, and security event timeline.

**Agent Activity**

Which AI clients are connecting, how often, and what they're doing — real-time session tracking.

**Tool Health**

Slowest and most error-prone tools, with actionable root-cause insights and performance baselines.

**Incident Log**

Error trends, failure rates, status-code breakdowns, and forensic audit trail access.

Get started at [cloud.vinkius.com](https://cloud.vinkius.com) — connect your AI agent in under 60 seconds.

# Deno Deploy MCP

15 tools available

Cloud-hosted on Vinkius

Managing an edge network is usually a CLI nightmare. This connection lets you talk to your Deno Deploy account using your preferred AI agent. Instead of running complex sequences of commands, you just ask your agent what you need done—whether it's listing every app in the organization or deploying a fresh revision.

For example, you can tell your agent to fetch details on an application, then immediately request its build logs for debugging. It handles the whole process, giving you visibility into everything from shared environment variables (layers) to custom domains. This makes it possible to run complex deployment and monitoring workflows without ever leaving your chat interface. By connecting this through Vinkius, you get a single point of access to orchestrate all your serverless applications.

---

## Core Capabilities

### 01 — Manage Applications

List all deployed applications within an organization or fetch deep configuration details for a specific app.

### 03 — Monitor Live Logs and Status

Stream build logs for new revisions or query historical application errors with filtering options.

### 05 — Inspect Organization Details

Get organization metadata, list associated custom domains, and view project-specific configurations.

### 02 — Control Deployment Cycles

Start new deployments (revisions) by uploading assets, or create project-specific deployments using the latest code base.

### 04 — Handle Infrastructure Layers

Create or update shared environment variables using layers, centralizing configs across multiple apps.

# One Click on Vinkius — From Prompt to Execution

Available at [vinkius.com/mcp/deno-deploy](https://vinkius.com/mcp/deno-deploy) — connect your AI agent in three steps.

- 01 First, subscribe to this MCP in Vinkius and provide your Deno Deploy Personal Access Token.
- 02 Next, prompt your AI agent with a clear request—for example, 'List all apps and show the logs for api-gateway'.
- 03 Finally, your agent executes the necessary tools, gathering real-time deployment status or historical log data directly into the chat.

The bottom line is you talk to your AI client like talking to a teammate, and it handles all the underlying API calls for Deno Deploy.

---

## Built For

This connection is built for Platform Engineers and DevOps who are tired of context-switching between their IDE, CLI, and dashboard. If you spend your day manually checking deployment status or piping logs to grep, this MCP saves hours.

### DevOps Engineer

Automating complex deployment pipelines and monitoring system health across multiple services without leaving the terminal or chat interface.

### Full-stack Developer

Quickly checking application logs or verifying a deployment status while debugging code in their IDE, instead of jumping into a separate log viewer.

### Platform Team Lead

Managing organizational resources, shared environment variables (layers), and domain configurations across many different projects efficiently.

---

## What Changes When You Connect

- 01 Stop manually checking deployment status. You can use `get_revision_progress` to stream real-time updates right in your chat interface.

- 
- 02 Debugging is faster than ever. Use `get_build_logs` or `get_app_logs` to query specific error types and timeframes without leaving your workflow.

---

  - 03 Centralize configurations. Create shared environment variables using `create_layer`, ensuring consistency across every app in the organization.

---

  - 04 Get a full picture of your infrastructure by listing all apps with `list_apps` and checking out custom domains via `list_domains`—all in one go.

---

  - 05 Manage complex projects efficiently. You can deploy an entire group using `create_project_deployment`, which is often faster than managing individual app deployments.
- 

---

## Real-World Applications

### A new feature deployment fails and needs debugging

A full-stack developer runs into a 500 error after pushing code. Instead of running `deno deploy --log` manually, they ask their agent to check the logs for that specific app using get_app_logs`, immediately spotting the database timeout error.`

### Need to update shared credentials across multiple apps

The team needs to change a database API key used by five different services. They use their agent to `create_layer` and push the updated variable once, knowing all associated applications will pick it up.`

### Platform team needs to audit all deployed services

A platform engineer needs a list of every service and its current status across 20 projects. They use their agent to run `list_apps` and then loop through the results, gathering metadata for an inventory report.`

### Checking if an old deployment was successful

A DevOps engineer needs confirmation on a revision from last week. Instead of guessing which ID to check, they ask their agent to `list_revisions` and then request the status using get_revision`.`

---

# Patterns to Avoid

---

## Assuming 'list apps' shows live logs

### X AVOID

The user runs a simple command like 'Show me the logs for my app.' and expects a real-time stream, but the tool only provides static metadata.

### ✓ INSTEAD

To view streaming or historical output, you must explicitly tell your agent to use ``get_build_logs`` (for compiling) or ``get_app_logs`` (for runtime errors).

---

## Deploying without shared config checks

### X AVOID

A developer deploys a new app that immediately fails because the environment variables are missing or incorrect, wasting time.

### ✓ INSTEAD

Before deploying, verify your infrastructure by listing and updating shared configurations using ``list_layers`` and then running ``create_layer`` to ensure all necessary keys are present.

---

## Using 'get app' for deployment status

### X AVOID

The user asks the agent to 'What is my deployment progress?' but only gets static metadata about the application, not the live build status.

### ✓ INSTEAD

To check if a revision is actually running or deploying, you must use ``get_revision_progress`` or request the specific status using ``get_revision``.

---

# The Right Fit

Use this MCP if your job involves routine operations on serverless infrastructure: managing deployments, monitoring logs, and updating shared environment variables. You'll want it when you need to automate tasks like creating a new revision ( `create_deployment` ) or auditing the entire organization's setup using `list_apps` . Don't use it if you are only building the core logic of your function; that happens in your IDE. If your primary goal is just viewing documentation, check out specialized documentation tools instead. This MCP focuses on operational control and real-time monitoring.

---

## Deno Deploy: Automating Edge Computing Deployment Pipelines with Deno Deploy

Right now, deploying updates to an edge function means a lot of clicking through command line interfaces. You have to run one command to deploy, another to check the status, and yet another to pull logs when things inevitably break. It's manual, slow, and makes debugging feel like detective work.

With this MCP, you simply instruct your agent to 'Deploy v2 of the auth service and check for errors.' The tool handles the sequence: it runs `create_deployment`, monitors progress with `get_revision_progress`, and if failures pop up, it automatically pulls logs via `get_app_logs` so you know exactly what broke. You get a full lifecycle report in one chat response.

---

## Deno Deploy: Managing Shared Variables and Domains with Deno Deploy

When your company has ten different microservices, updating a single credential—like the Stripe API key—is painful. You have to manually log into each service's configuration dashboard or run update commands for every single one.

This MCP fixes that by allowing you to manage shared environment variables using `create_layer`. By centralizing these keys and then telling your agent to update the layer, you propagate the change across all services instantly. It makes managing a complex, multi-app setup feel like flipping one switch.

---

# Deno Deploy: 15 Tools for Edge Computing Operations & Logging

Use these tools to manage application lifecycle, monitor deployment status, create layers, and query historical logs across your Deno Deploy infrastructure.

#	TOOL	DESCRIPTION
01	<code>create_app</code>	Sets up a brand new Deno Deploy application in your account.
02	<code>create_deployment</code>	Triggers a new deployment (revision) for an existing app, updating the live code.
03	<code>create_layer</code>	Creates a shared environment layer to hold variables used across multiple applications.
04	<code>create_project_deployment</code>	Deploys a new revision specifically for an entire project, rather than just one application.
05	<code>get_app_logs</code>	Queries historical logs for an application based on specific time frames and error levels.
06	<code>get_app</code>	Fetches detailed metadata and configuration settings for a single Deno Deploy app.
07	<code>get_build_logs</code>	Streams real-time build logs, showing the process as the application is compiled into a new revision.
08	<code>get_organization</code>	Retrieves high-level details about your entire Deno Deploy organization account.
09	<code>get_revision_progress</code>	Streams the progress of a deployment revision, showing its current status (e.g., uploading assets).
10	<code>get_revision</code>	Retrieves the current operational status and metadata for a specific deployed revision ID.
11	<code>list_apps</code>	Lists all applications in your organization, supporting filtering by labels to narrow down results.
12	<code>list_domains</code>	Retrieves a list of custom domains registered and associated with the organization.
13	<code>list_projects</code>	Shows all defined projects within your Deno Deploy account structure.

#	TOOL	DESCRIPTION
14	<code>list_revisions</code>	Lists every historical revision for a specific application, providing version history.
15	<code>update_layer</code>	Modifies existing shared environment layers to update variables without affecting other apps.

# See It in Action

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

**U** I need to know if my 'api-gateway' service is running correctly and what its latest build logs were.



Okay, I checked the status for api-gateway. It appears operational. Here are the last 10 entries from the build log:

TIMESTAMP	LEVEL	MESSAGE
2024-06-12T10:05Z	INFO	

Build started successfully.

| 2024-06-12T10:07Z | WARN | Deprecation warning on module X. Upgrade needed. |  
| 2024-06-12T10:10Z | SUCCESS | Deployment finished. App ready for traffic. |

Looks clean, but keep an eye on that deprecation warning.

**U** List all my current apps and tell me which ones are using custom domains.



## Deno Deploy Apps Overview

- `auth-service` : Running. Domain: `auth.yourdomain.com` .
- `web-frontend` : Deploying (65%). Domain: N/A.
- `payment-api` : Running. Domain: `payments.yourcompany.net` .

I also found the following custom domains registered to your account:

- `support.yourdomain.com`
- `dev.yourcompany.net`

## U Can you help me update the database URL for all my services?



Got it. I'm updating the shared environment layer now. This affects all apps drawing from this layer.

✅ **Success:** Layer 'global-env' updated.

• **Variable:** DATABASE\_URL

• **New Value:** postgresql://newuser:s3cur3p@prod-db.com/main

Please trigger a redeployment for critical apps like the payment API to ensure they pick up the new connection string.

---

## Frequently Asked Questions

---

### 01 How do I use Deno Deploy MCP if my deployment fails?

If your deployment breaks, you don't need to guess what went wrong. You can ask the agent for the logs and it will fetch real-time or historical build logs, showing exactly where the failure occurred.

---

### 02 Can Deno Deploy MCP help me manage multiple projects?

Absolutely. The MCP allows you to list all your defined projects and then target deployments or configuration changes across them using project-specific tools. It keeps your multi-app setup clean.

---

### 03 What if I need to change a variable that 10 different apps use?

Instead of logging into every app's dashboard, you manage shared variables through the 'layers' feature. You update the layer once, and your agent handles propagating that change across all dependent services.

---

### 04 Does Deno Deploy MCP only work for new deployments?

No. It gives you full historical visibility. You can list all past revisions and check the status or logs of any specific, older deployment ID.

---

### 05 Is this connection better than using the command line CLI?

Yes. The primary difference is conversation flow. Instead of running `list\_apps` then copying the name for `get\_app`, you just ask your agent to 'List apps and show me the details for X,' getting everything in a single, conversational step.







---

# Go Live in 60 Seconds

Get your connection token from [cloud.vinkius.com](https://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
 <b>Claude AI</b>	Profile → Customize → Connectors → "+" → Add custom connector → Paste endpoint
 <b>Cursor</b>	Settings → Features → MCP Servers → "+ Add New MCP Server" → Type: SSE → Paste endpoint
 <b>VS Code</b>	Ctrl/Cmd+Shift+P → "MCP: Add Server" → add <code>"deno-deploy": { "url": "..." }</code>
 <b>Windsurf</b>	MCP Settings → <code>mcp_settings.json</code> → Add endpoint URL
 <b>ChatGPT</b>	Settings → Tools & plugins → Add MCP server → Paste endpoint
 <b>Gemini</b>	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

# Deno Deploy 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](https://cloud.vinkius.com) →

[vinkius.com](https://vinkius.com) · [support@vinkius.com](mailto: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 Deno Deploy. 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	Deno Deploy MCP
Server ID	019e3889-117d-7374-82ab-f479994454c7
Platform	Vinkius Cloud for AI Agents
Endpoint	<a href="https://edge.vinkius.com/{token}/mcp">https://edge.vinkius.com/{token}/mcp</a>

### 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/deno-deploy](https://vinkius.com/mcp/deno-deploy).