MCP SERVER | NO CODE | CLOUD HOSTED

Argo CD (GitOps) MCP for AI Agents

Automate Kubernetes deployments and check logs via natural conversation

Argo CD (GitOps) MCP connects your AI agent directly to your Kubernetes deployment pipeline. It lets you list applications, sync deployments, roll back failed versions, and pull real-time logs for any cluster or repository—all through natural conversation.

A+ Quality Score 100/100

kubernetes | gitops | continuous-deployment | argo-cd | k8s-management

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/argo-cd-gitops

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.

Argo CD (GitOps) MCP

13 tools available
Cloud-hosted on Vinkius

This connection gives your AI agent full control over complex GitOps workflows. Instead of logging into the console and running a dozen commands to check deployment status, you just talk to it. You can ask the MCP to list all deployed applications across multiple clusters, then tell it to sync any that are out of date. If something breaks, you don't have to leave your chat window; you pull real-time logs for the specific application and analyze them right alongside your deployment history. Furthermore, you manage the infrastructure itself—add or delete entire clusters, or inspect project groupings to understand who owns what resources. It's all managed through a single conversation interface, making complex cluster operations accessible from any MCP-compatible client connected via Vinkius.

Core Capabilities

01 — Check and list deployed applications

Retrieve a comprehensive overview of every application currently running in your Argo CD environment.

02 — Initiate deployments and sync changes

Force an update or synchronization on any specified application to match the desired state defined in Git.

03 — Revert failed deployments (Rollback)

Programmatically roll back a faulty application to a previously stable, known-good version.

04 — Analyze real-time logs

Fetch and analyze live container logs for specific applications without leaving your chat interface.

05 — Manage infrastructure components

Add, remove, or list entire Kubernetes clusters and Git repositories connected to Argo CD.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/argo-cd-gitops** — connect your AI agent in three steps.

01 Subscribe to the MCP and provide your specific Argo CD URL and authentication token.

02 Your AI agent authenticates with the service, mapping your conversational requests (e.g., 'sync staging-api') into structured API calls for deployment operations.

03 The MCP executes the required GitOps command on your cluster, returns the status (success/failure), and reports back details like log snippets or application state changes.

The bottom line is: you use natural language to execute complex infrastructure commands that usually require direct CLI access.

Built For

This MCP targets the DevOps Engineer who spends hours clicking through dashboards just to figure out why a deployment failed. It's for the SRE tired of context switching between chat, terminal, and monitoring tools. If your job involves keeping Kubernetes clusters running smoothly across multiple environments, this is built for you.

Site Reliability Engineer (SRE)

Investigating production outages by pulling application logs directly into the chat context to quickly pinpoint a failing container or connection timeout.

DevOps Engineer

Managing deployment lifecycles across multiple clusters, triggering sync operations and rolling back changes without writing boilerplate script code.

Software Developer

Testing their own application environments by creating new Argo CD applications or inspecting AppProjects to verify resource constraints before merging code.

What Changes When You Connect

01 Stop manually checking deployment status. Use the `list_applications` tool to get a full inventory of every running service in one prompt.

02 Debugging is faster than ever. Instead of SSHing into a box, use `get_application_logs` to pull live logs and find the root cause instantly within your agent chat.

03 Rollbacks are simple. If a feature breaks, simply tell the MCP to run `rollback_application`, directing Argo CD to revert to a stable commit state.

04 Infrastructure management is centralized. You can use `add_cluster` or `delete_cluster` without needing deep CLI knowledge, just by describing what you need done.

05 Better visibility means better governance. Use `list_projects` and `get_project` to understand which teams own which resources before making changes.

Real-World Applications

The staging environment deployment is broken.

A developer asks their agent, 'Show me why the API gateway isn't syncing.' The agent runs `get_application_logs`, identifies a database connection timeout error in the logs, and recommends checking the AppProject resource limits.

I need to quickly test rolling back an app.

An SRE needs to revert a change immediately. They prompt the agent to `rollback_application` on 'user-auth'. The MCP executes the rollback, confirms the new version is deploying, and reports success.

We need to onboard a brand new cluster.

The DevOps engineer asks the agent to add a new target environment. The agent runs `add_cluster`, validates connectivity, and makes the entire cluster available for deployment management.

I can't find out who owns this resource.

A developer uses the MCP to run `get_project` on a specific app. The agent returns details about the AppProject, showing ownership boundaries and required permissions.

Patterns to Avoid

Running manual cluster commands

✗ AVOID

Typing out complex `kubectl get pods -n namespace --selector label=value` commands repeatedly to check status across multiple environments.

✓ INSTEAD

Instead, ask the agent to run `list_applications`. It gathers all necessary deployment statuses and presents them in a single, digestible overview.

Forgetting which repo is linked

✗ AVOID

Getting confused about whether the 'backend-service' app uses the main git branch or the release tag for its source code.

✓ INSTEAD

Use `list_repositories` first. This shows all registered Git repos, helping you confirm exactly where the application definition lives before triggering a sync.

Overwriting stable configs

✗ AVOID

Manually forcing an update without checking if the target cluster is ready for the change, risking downtime.

✓ INSTEAD

Always start by running `get_project` to understand the resource constraints and permissions. Then use `sync_application` only after confirming readiness.

The Right Fit

Use this MCP if your pain point is context switching between CLI commands, dashboards, and chat interfaces when managing Kubernetes deployments. It's perfect for SREs who need to rapidly diagnose issues by pulling logs (`get_application_logs`) or roll back changes (`rollback_application`). Don't use it if you just need to manage simple secrets; those require a dedicated secrets manager MCP. Also, don't use it to write the initial manifest YAML—that's better handled by code generation tools. Use this when you need *execution* and *observability* across your entire GitOps lifecycle.

Argo CD (GitOps) MCP: Automating Kubernetes Deployment Status Checks

Right now, checking the health of a deployment means jumping through hoops. You might run `kubectl get pods` for one namespace, then switch to another terminal window to check logs with `kubectl logs`, and finally hit the Argo CD dashboard just to see if it says 'OutOfSync.' It's tedious, error-prone copy/pasting across three different tools.

With this MCP, you simply ask your agent: 'What is the status of the staging API?' The tool runs through all those checks—listing apps and fetching logs—and gives you a single report. You get immediate operational visibility without leaving your conversational flow.

Argo CD (GitOps) MCP: Controlling Cluster and Repository Operations

Managing the underlying infrastructure is usually separate from deployment management. Adding a new cluster or linking a source repository often requires dedicated, complex CLI commands that you have to remember and execute in sequence.

This MCP exposes `add_cluster` and `list_repositories`. You can tell your agent, 'We need to target Cluster Beta,' and it handles the necessary setup. It centralizes infrastructure control right where you manage applications.

Argo CD (GitOps): 13 Tools for Cluster Management

Use these tools to manage the full lifecycle of applications, clusters, and repositories within your Argo CD environment through natural language commands.

#	TOOL	DESCRIPTION
01	add_cluster	Adds a new, target Kubernetes cluster into your managed Argo CD environment.
02	add_repository	Registers a Git repository that Argo CD can monitor for application code changes.
03	get_application_logs	Retrieves the recent, real-time logs for a specified Argo CD application instance.
04	create_application	Sets up a brand new Argo CD application definition pointing to a specific Git source and target cluster.
05	create_project	Defines a logical AppProject grouping for better resource management and permission control within the system.
06	delete_cluster	Removes a specific Kubernetes cluster from Argo CD's list of managed deployment targets.
07	get_project	Fetches detailed information about an existing AppProject, including its members and constraints.
08	list_applications	Provides a list of every application currently tracked by Argo CD across all connected clusters.
09	list_clusters	Lists all Kubernetes clusters that are registered and manageable through this MCP connection.
10	list_projects	Displays a list of all defined AppProjects, allowing you to see resource boundaries.
11	list_repositories	Shows all Git repositories that Argo CD is configured to watch for deployments.
12	rollback_application	Triggers a controlled rollback of an application to its last known stable version.
13	sync_application	Forces the synchronization of an application, ensuring the live cluster matches the state defined in Git.

See It in Action

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

U Show me all the apps that are currently out of sync across my production environment.

Deployment Status Report

- `frontend-app` : OutOfSync (Needs Sync)
- `billing-api` : Healthy (Synced)
- `user-service` : OutOfSync (Requires Rollback)

I found two apps needing attention. Should I trigger a sync for `frontend-app` or rollback `user-service` ?

U The billing API is failing and the logs are showing connection errors.

Logs for billing-api (Last 50 lines)

```text
[2026-06-12T08:30:11Z] ERROR: Connection timeout to 'db-primary'. Check service endpoint.
[2026-06-12T08:30:15Z] WARN: Retrying connection attempt...
[2026-06-12T08:31:01Z] CRITICAL: Failed to establish connection. Status code 504.
```

The logs confirm a persistent database timeout issue.

U Add a new cluster called 'dev-sandbox' and ensure it can manage apps.

Adding cluster 'dev-sandbox'... **SUCCESS**. The cluster is now registered with Argo CD.

- **ID:** dev-sandbox (us-east-2)
- **Status:** Ready for deployment.
- **Next Step:** You can now use `create_application` to point a service here.

Frequently Asked Questions

01 **How does the Argo CD (GitOps) MCP help me with daily deployments?**

It lets you manage complex Kubernetes changes through simple conversation. You can list all apps, sync updates, or roll back faulty versions without touching the command line.

02 **Can I use this MCP to debug my applications when they crash?**

Absolutely. If an app fails, you ask the agent for logs (`get_application_logs`). It pulls real-time container output directly into your chat window so you can see exactly what went wrong.

03 **Is this MCP safe to use if I'm not a deep DevOps expert?**

Yes. The AI agent handles the complex commands for you. You give it natural language instructions, and the MCP executes them safely within your Argo CD environment.

04 **Does this MCP help me manage multiple clusters at once?**

Yes. It provides tools to list all connected clusters (`list_clusters`) and apply changes across different environments from a single chat session, keeping everything centralized.

05 **If I change my Git repo structure, can the MCP help me update Argo CD?**

Yes. You use tools like `add_repository` to register new sources and then run sync operations to ensure your applications reflect the new code base.

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 `"argo-cd-gitops": { "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

Argo CD (GitOps) 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 Argo CD (GitOps). 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	Argo CD (GitOps) MCP
Server ID	`019e3867-3be2-71d1-ad07-f47d3d20445b`
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/argo-cd-gitops.

© 2026 Vinkius · vinkius.com · support@vinkius.com