MCP SERVER | NO CODE | CLOUD HOSTED

Argo Workflows MCP for AI Agents

Monitor Kubernetes Orchestrations and Pipeline Deployments

Argo Workflows lets your AI agent take full control of complex infrastructure orchestrations running on Kubernetes. You can query, list, and inspect every active pod, workflow template, or scheduled job without touching a command line. It's instant visibility into your entire deployment pipeline.

D Quality Score 55/100

kubernetes | workflow-automation | container-orchestration | ci-cd | infrastructure-as-code | job-scheduling

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

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 Workflows MCP

6 tools available
Cloud-hosted on Vinkius

Running large-scale container deployments means managing incredibly complicated dependency graphs. This MCP connects your AI agent directly to your Argo Workflows cluster, giving you natural language access to every running process and historical record. Instead of opening multiple dashboards or wrestling with `kubectl` commands, you simply ask your agent what's going on. Your agent can list all active jobs across different namespaces, dive deep into a failed pipeline's resource tree, or check if that critical nightly cron job actually ran this morning. When something breaks—and it always does—you don't waste time figuring out *where* to look; you just ask your AI client. The entire Vinkius catalog makes this possible, giving your agent the deep visibility needed to debug complex infrastructure patterns immediately.

Core Capabilities

01 — List all current workflows

Retrieve a list of workflow executions that are running, pending, or have recently finished within any specified Kubernetes namespace.

02 — Inspect specific workflow status

Get the detailed resource tree and operational status for a single Argo workflow instance to pinpoint exactly where a failure occurred.

03 — View reusable templates and scheduled jobs

List parameterized WorkflowTemplates, allowing you to see which job structures are available, and list all recurring CronWorkflows that handle scheduled tasks.

04 — Audit historical runs

Search through archived workflow records in the history database to understand past infrastructure patterns or identify old failure points.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to this MCP and provide your Argo Cluster Server URL along with a valid RBAC Bearer Token.

02 Connect your preferred AI client (like Claude, Cursor, or Windsurf) using the Vinkius platform.

03 Ask your agent a question—for example, 'What's wrong with the nightly ETL job?'—and get immediate status reports and failure details.

The bottom line is that you talk to your AI client like talking to a teammate; it handles all the complex Kubernetes API calls for you.

Built For

This MCP is built for DevOps and SRE teams who live in dashboards. If you spend too much time clicking between Argo UI, `kubectl`, and logs just to figure out why a deployment failed, this is for you. It gives your agent the single source of truth on your cluster's health.

DevOps Engineer

Debugging pipeline failures by running list_workflows or inspecting deep resource trees via get_workflow without leaving their chat window.

SRE (Site Reliability Engineer)

Quickly checking the overall health of the Argo server and querying historical archives to validate service uptime metrics.

Data Engineer

Monitoring complex, scheduled ETL workflows and analyzing which reusable templates are available for data cleanup or model training.

What Changes When You Connect

01 Eliminate dashboard hopping. Instead of opening multiple UIs to check status, your agent gives you a single, conversational answer about the health of any workflow.

02 Pinpoint failures faster using get_workflow. You get the precise resource tree that shows exactly which node or pod failed and why, saving hours of debugging time.

03 Manage recurring jobs easily. List cron_workflows to audit all scheduled tasks without manually checking the deployment manifest files for every service.

04 Understand your infrastructure history. list_archived_workflows lets you look back at past runs to understand patterns that might cause future failures.

05 Save keystrokes and time. Your AI agent handles complex API calls, turning difficult CLI commands into simple chat prompts.

Real-World Applications

The Nightly ETL Pipeline Failed

A data engineer asks their agent to check the 'data-pipeline' workflow. The agent uses get_workflow, identifies that a specific step failed with an S3 permission error, and tells them exactly which parameter needs fixing.

Auditing Compliance for Scheduled Jobs

An SRE needs to know if the billing report ran last month. They ask the agent to list_archived_workflows, quickly finding the historical record and confirming successful execution dates for compliance purposes.

Checking Available Job Blueprints

A developer asks what reusable templates are available. The agent uses list_workflow_templates, providing a clear list of parameterized options like 'model-training-tmpl' so the developer can start coding immediately.

Debugging a Pending Service Deployment

A DevOps engineer sees an active workflow stuck in 'pending'. They ask the agent to get_workflow, and it reports that the issue is with node resource allocation, giving them the exact error message needed for platform adjustments.

Patterns to Avoid

Relying solely on the Argo UI

✗ AVOID

Opening the web interface and manually clicking through multiple stages to find a failure point, wasting time switching tabs and reloading pages.

✓ INSTEAD

Use your agent to call get_workflow directly. Just asking your AI client for the deep inspection tree instantly surfaces the failure status without needing any manual UI interaction.

Using general Kubernetes commands

✗ AVOID

`kubectl describe pod` only gives basic container health, but doesn't provide the full context of the workflow template or scheduled run that spawned it.

✓ INSTEAD

Use your agent to list_workflows. This provides a higher-level view of the entire execution chain—the parent workflow and its associated resources—which is much more informative.

Forgetting job dependencies

✗ AVOID

Assuming that because one step passed, all previous steps ran correctly, leading to incomplete debugging when a later failure occurs.

✓ INSTEAD

Always use get_workflow for deep inspection. This mechanism shows the full dependency graph and status of *every* resource in the workflow execution tree.

The Right Fit

Use this MCP if your core problem is visibility into complex, scheduled infrastructure pipelines running on Kubernetes. You need to know *why* a deployment failed, not just *that* it failed. It's perfect for debugging state and auditing history using tools like get_workflow and list_archived_workflows.

Don't use this if you are only trying to manage basic container deployments (those should be handled by pure Kubernetes APIs). Also, don't rely on this MCP just because your AI client can run shell commands; it specifically reads the structured state from Argo Workflows. If you need simple notifications or single-resource status checks outside of a workflow context, an alternative general monitoring tool might suffice.

Argo Workflows and DevOps: Debugging Complex Kubernetes Pipeline Failures

Today, debugging a failed deployment means a nightmare of clicking. You open the Argo Web UI, check the active workflow status, then jump to logs, maybe run `kubectl` in another terminal window, and finally scroll through confusing resource trees trying to pinpoint where that single step broke. It's slow, it's manual, and you lose context between tabs.

With this MCP, your agent handles all of that complexity for you. You simply ask your AI client about the failed run, and it executes deep inspections via get_workflow, pulling together the resource tree status from across the entire cluster into one readable response. You walk away with a definitive answer, not just a trail of half-broken links.

Argo Workflows and SRE: Auditing Scheduled Jobs and Historical Data

Manually auditing scheduled jobs is a massive time sink. You have to track down which cron job runs when, where its definition lives, and whether the

Now, simply ask your agent to list_cron_workflows or list_archived_workflows. The MCP consolidates that historical knowledge instantly, giving you a

previous run succeeded months ago. It requires knowing dozens of specific resource names and APIs.

clean report on job schedules and past runs without needing to query the database manually.

6 Tools in the Argo Workflows MCP for Pipeline Debugging

Use these tools to list, fetch details, inspect templates, and audit every resource related to your Kubernetes workflows.

#	TOOL	DESCRIPTION
01	`list_workflows`	Retrieves a list of all active workflow executions running within the Kubernetes namespace.
02	`get_workflow`	Fetches the detailed resource tree and current status for any specific Argo workflow instance.
03	`list_workflow_templates`	Provides an inventory of reusable WorkflowTemplates that can be used to define new jobs.
04	`list_cron_workflows`	Lists all scheduled CronWorkflows, helping you see what recurring jobs are set up in the cluster.
05	`list_archived_workflows`	Searches and lists historical workflow records stored in the Argo history database for auditing purposes.
06	`get_server_info`	Retrieves basic operational information about the underlying Argo Workflows server itself.

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 the nightly billing process ran successfully yesterday.

Workflow Status Report: Billing Process

- **Job Name:** `billing-nightly`
- **Status:** ✅ SUCCESS
- **Completion Time:** 2026-06-11T04:00:15Z
- **Duration:** 18 minutes

The run completed successfully. All associated pods (pg_extract, s3_upload) reported a clean exit code 0.

U What's wrong with the 'user-sync' workflow? It's stuck in pending.

I checked the tree for `user-sync`. The failure is at node **Authentication Step**.

- **Error:** Timeout waiting for external service credential.
- **Code:** 401 Unauthorized (S3)
- **Action Needed:** Review the Service Account's IAM policy to ensure it has read access to the target bucket.

U List all reusable templates I can use for data processing.

Available Workflow Templates in Default Namespace:

1. `data-cleanup`: Requires S3 Bucket URI input.
2. `model-training-tmpl`: Accepts array of hyperparameters (e.g., learning_rate).
3. `slack-notification-hook`: Takes a target channel and message body.

Frequently Asked Questions

01 **How can the Argo Workflows MCP help me debug a broken deployment?**

It lets your agent check deep into the workflow's resource tree, showing you exactly which step failed and why. You get granular details about the failure—like an incorrect code or missing permission—without having to navigate complex UI screens.

02 **Does this MCP track historical data for Argo Workflows?**

Yes, it can list archived workflows. This is huge for compliance and auditing because you don't have to guess what happened last month; the agent finds that record for you.

03 **Is this better than using basic kubectl commands?**

Absolutely. While `kubectl` gives status, this MCP provides context. It connects all the pieces—the template definition, the scheduled run, and the live resource state—in one conversational answer.

04 **What if I need to see a list of all recurring jobs?**

You can use the MCP to list cron workflows. This gives you a clean inventory of every job that runs automatically, making it easy to audit schedules and check for orphaned or outdated tasks.

05 **Can I see what reusable templates are available in my cluster?**

Yes. The agent lists all the workflow templates defined in your namespace. This helps developers quickly find existing blueprints, saving them time writing boilerplate code.

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-workflows": { "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 Workflows 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 Workflows. 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 Workflows MCP
Server ID	`019d7552-0b30-71ee-8d2a-1094269b96f7`
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-workflows.

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