MCP SERVER | NO CODE | CLOUD HOSTED

Dagster MCP for AI Agents

Monitor and manage data pipelines, assets, and schedules in real-time

Dagster connects your data orchestration platform to any AI client, letting you manage complex pipelines and track data assets using natural conversation. Instead of clicking through dashboards or writing code just to check status, you ask your agent about job runs, dependency maps, or scheduled triggers. It's full control over your entire data stack, right from the chat window.

A+ Quality Score 100/100

data-orchestration | data-pipelines | workflow-automation | observability | data-engineering | asset-management

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

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.

Dagster MCP

6 tools available
Cloud-hosted on Vinkius

Your AI client can now talk directly to your Dagster instance, giving you granular control over your data workflows without opening a dashboard. You manage everything—from listing all available jobs to checking if a critical asset is fresh—all through natural language.

For example, you can ask the agent to list every configured job schedule or trace back which assets depend on raw customer data. This means deep visibility into your entire data mesh. The system aggregates this power and makes it accessible via Vinkius, giving you a single point of control regardless of whether you use Cursor, Claude, or any other compatible AI client.

It's less about learning a new UI and more about talking to your infrastructure the way you already talk to a teammate. You get immediate status updates on job runs, detailed logs for failures, and full audit trails for every piece of data that moves through your system.

Core Capabilities

01 — Review all available jobs

List the names and boundaries of every data pipeline job configured in your Dagster instance.

02 — Check historical job run status

Fetch a chronological list of recent job runs, allowing you to select specific runs for detailed status or execution logs.

03 — Map data asset dependencies

Enumerate all software-defined assets in your project to understand what data relies on which source.

04 — Audit automated triggers

List every configured job schedule and active sensor, verifying exactly how and when pipelines are supposed to run automatically.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to this MCP on Vinkius and provide your Dagster URL along with a valid User API Token.

02 Your AI client authenticates the connection, granting it read access across your data platform's metadata.

03 You issue a natural language command—like 'Show me all failed runs for the marketing job.'—and the agent retrieves and displays the specific results.

The bottom line is that you manage complex data operations using chat commands instead of navigating multiple dashboards.

Built For

This MCP is essential for any professional who spends time monitoring, troubleshooting, or auditing production data pipelines. If your job involves knowing if the right data landed in the right place at the right time, this connector saves you hours of UI clicking.

Data Engineer

Uses it to check pipeline health instantly by listing jobs or fetching run logs when a scheduled ETL job fails unexpectedly.

Analytics Engineer

Relies on this MCP to track data assets and verify data freshness in real-time, confirming that source tables have been correctly materialized.

Data Platform Manager (SRE)

Audits job schedules and sensor configurations across multiple organizational clusters to ensure all automation triggers are active and pointing to the correct endpoints.

What Changes When You Connect

01 Instant troubleshooting: Instead of opening the UI to check status, you can use `list_runs` or `get_run` with your agent to pull detailed logs for failed jobs immediately.

02 Dependency mapping: The `list_assets` tool allows you to see which data tables rely on others, making it easy to verify data lineage and pinpoint the source of stale metrics.

03 Complete automation audit: Use the MCP to list both job schedules (`list_schedules`) and active sensors (`list_sensors`), ensuring every automated trigger is configured correctly across all environments.

04 Holistic visibility: You can view all available pipelines by calling `list_jobs` in one chat command, giving you a single overview of your entire data platform boundary.

05 Deep operational insight: By querying the system via natural language, you gain immediate access to run status and asset details without needing specialized CLI commands or dashboard filters.

Real-World Applications

Investigating a failed ETL job

A data engineer notices the morning sales metrics are missing. They ask their agent, who uses `list_jobs` to find the 'daily-sales' pipeline, then calls `get_run` to check the last execution logs, confirming the failure was due to an upstream dependency.

Verifying data freshness for a report

An analytics engineer needs to know if their board dashboard is using current data. They ask about assets, and the agent uses `list_assets` to enumerate all required tables, allowing them to verify that the 'cleaned_customer' asset was recently materialized.

Auditing scheduled maintenance

A platform manager needs to ensure a backup pipeline runs exactly when expected. They ask the agent to list schedules, and it uses `list_schedules` to confirm that the 'daily-backup' job is correctly configured for 2:00 AM UTC.

Debugging unexpected triggers

An SRE finds a pipeline ran when no one should have triggered it. They ask the agent to list sensors, and it uses `list_sensors` to identify that an external event listener is running too broadly, allowing them to narrow down the scope.

Patterns to Avoid

Asking for raw API dumps

✗ AVOID

Writing a prompt like: 'Give me the full JSON payload for run ID 123 and all related asset IDs.' This is too verbose and often requires specific, technical parameters.

✓ INSTEAD

Simply ask your agent in natural language: 'What was the status of run ID 123?' The tool handles the complexity; you just get the answer.

Forgetting to check dependencies

✗ AVOID

Assuming that because a job ran successfully, all the data products it creates are ready for use. This ignores potential upstream failures.

✓ INSTEAD

Before trusting the results, ask your agent to `list_assets` and confirm the materialization status of every critical asset in the pipeline.

Only checking job names

✗ AVOID

Limiting your audit to just 'Show me all jobs.' This doesn't tell you if those jobs are actually scheduled or if they have dependencies.

✓ INSTEAD

Use multiple commands, like asking to `list_jobs` first, then running a separate query to `list_schedules` and `list_assets` for full coverage.

The Right Fit

You should use this MCP if your core job involves monitoring the lifecycle of structured data—things like ETL pipelines, materialized views, or scheduled reports. It's perfect when you need to verify *status* (did it run?) and *state* (is the data current?). Don't use it if your primary goal is writing new code or designing the pipeline logic itself; for that, you need a development environment. If you only want simple messaging or record creation without checking data state, look at general-purpose database MCPs instead.

Dagster MCP: Managing Data Pipeline Visibility and Jobs

Right now, keeping tabs on your company's data flows means jumping between dashboards. You open the Dagster UI to check if a job ran; you click into the run history to see logs; and then you might have to manually cross-reference assets to confirm the data is ready for consumption. It’s constant context switching.

With this MCP, your agent handles all that manual clicking. You ask a natural language question—like 'What was the status of yesterday's ML job?'—and it instantly executes `list_runs` and delivers a clean answer right in the chat.

Dagster MCP: Auditing Data Assets and Schedules

The biggest manual pain point is auditing. You have to manually check job schedules, then separately verify sensors, and finally confirm data dependencies using asset graphs—a tedious process prone to human error.

This MCP lets you query all those boundaries together. By asking the agent to list assets or audit scheduled triggers, you get a complete, cross-system view of your entire data platform in one prompt.

6 Tools for Dagster: Data Workflow Management

Use these tools in natural conversation to list jobs, check run status, map data dependencies, and audit automated schedules across your entire platform.

#	TOOL	DESCRIPTION
01	`list_jobs`	Lists every data pipeline job defined in your Dagster deployment.
02	`list_runs`	Retrieves a history of recent job executions to give you an overview of system activity.
03	`get_run`	Fetches deep details and status logs for one specific, identified run ID.
04	`list_assets`	Lists all software-defined data assets to map out physical dependencies within your project.
05	`list_schedules`	Retrieves a list of every scheduled job, showing when they are set to run next.
06	`list_sensors`	Lists active sensors that wait for external events before triggering a pipeline execution.

See It in Action

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

U What were the status and error logs for the 'finance' job yesterday?

Pipeline Run Audit: Finance ETL

- **Job:** `daily_financial_report`
- **Date Checked:** 2026-06-11
- **Last Run Status:** Failure (ID: 987)
- **Error Log Snippet:** Connection timeout detected while accessing the 'raw_ledger' source. Check credentials or network access.

> *Action required: Credentials need updating for the ledger connection.*

U Show me all critical data assets and if they are fresh right now.

Asset Lineage Report

The following 4 key assets were found:

ASSET NAME	LAST MATERIALIZATION	DEPENDENCIES	STATUS
`cleaned_user_profiles`	2 hours ago	`raw_auth_logs`	✅ Fresh
`monthly_sales_summary`	3 days ago	N/A	⚠️ Stale (Needs Run)
`product_inventory_view`	Never found	N/A	❌ Missing

Please run the pipeline for `monthly_sales_summary`.

U Which jobs are scheduled to run next week?

Scheduled Jobs Report

Here are all active schedules:

- `daily_etl_job` : Runs every day at 1:00 AM UTC.
- `weekly_ml_retrain` : Runs every Monday at 6:00 AM UTC. (Checks for external triggers).
- `hourly_sync` : Runs every hour on the hour, unless a sensor detects an event.

Frequently Asked Questions

01 **How do I check if my data pipelines are running correctly using Dagster MCP for AI Agents?**

You simply ask your agent about pipeline status. It uses the `list_runs` and `get_run` tools to pull historical execution logs, letting you instantly see if a job succeeded or where it failed.

02 **Can Dagster MCP help me track data dependencies across different tables?**

Yes. You can ask the agent to list all software-defined assets using `list_assets`. This shows you exactly which pieces of data rely on others, helping you map out your full data lineage.

03 **What if I need to know when my automated jobs are supposed to run?**

You can audit all automatic triggers. By asking the agent about schedules and sensors, it will list every configured job schedule and any external event listeners, giving you full visibility into automation.

04 **Does Dagster MCP only work if I have a complex setup?**

No. The tool is designed to talk to your existing Dagster instance (whether Plus or self-hosted). You just need the URL and API token, and you can start querying job boundaries right away.

05 **Is this MCP better than using a regular dashboard UI?**

For quick checks and troubleshooting, yes. It's faster because you don't have to navigate menus; you just ask the question in plain English and get the specific data result immediately.

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 `"dagster": { "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

Dagster 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 Dagster. 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	Dagster MCP
Server ID	`019d7581-1c1f-7235-9ae0-0353e1d0829e`
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/dagster.