MCP SERVER | NO CODE | CLOUD HOSTED

Orkes Conductor MCP

See the state of any workflow, instantly.

Orkes Conductor connects your agent directly to complex workflow engines, giving you full visibility into microservice processes. It lets you list entire workflow definitions, track running instances across services, and search through historical execution data. Stop opening dozens of dashboards; ask your AI client everything about your system's operational state in one go.

A+ Quality Score 100/100

orchestration | microservices | workflow-engine | task-scheduling | execution-monitoring

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

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.

Orkes Conductor MCP

6 tools available
Cloud-hosted on Vinkius

You're dealing with systems where a single user action triggers a dozen backend steps—payment processing, inventory updates, notifications. Tracking if that whole chain worked is usually a nightmare of clicking through different monitoring UIs. This MCP lets your AI client bypass the dashboards entirely. It connects directly to your orchestration layer, giving you immediate read access to workflow definitions, currently running instances, and the granular history of any task execution.

Need to know why an order failed last week? You can search across all historical runs using powerful queries. Want to see if a process is stuck right now? You list active workflows by name and inspect their current state. If you're building complex agentic applications, connecting this MCP via Vinkius gives your agent the single source of truth it needs to debug or audit multi-step business logic without needing dedicated API calls for every piece of data.

Core Capabilities

01 — View all defined workflows

List and inspect every registered workflow definition, including their versions and underlying task structures.

02 — Find active process runs

Get a list of currently running workflow instances by filtering them based on the name they belong to.

03 — Deeply inspect execution failures

Retrieve detailed state information for any specific workflow run, showing task-by-task history and error messages.

04 — Search historical runs quickly

Perform broad searches across all past workflow executions using customizable query filters like status or correlation ID.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to the Orkes Conductor MCP and provide your required Access Key, Secret, and Base URL.

02 Your AI agent uses these credentials to authenticate against your orchestration cluster.

03 The agent then executes specific commands—like searching or listing definitions—and returns structured data directly to your chat interface.

The bottom line is you get a conversational window into the guts of your distributed system, without needing any terminal commands or UI clicks.

Built For

Platform engineers and DevOps teams who spend their mornings staring at cryptic dashboard alerts. If you're tired of manually cross-referencing monitoring tools to find out why a complex process stalled, this MCP is for you.

DevOps Engineer

Uses the agent to search execution history across multiple services during an incident response, quickly identifying failure patterns without logging into the main dashboard.

Platform Architect

Inspects workflow graphs and task definitions to understand dependencies or validate changes before deploying a new service integration.

Backend Developer

Monitors active, running workflows to check the real-time progress of complex background tasks or test failure scenarios against development environments.

What Changes When You Connect

01 Stop guessing where a process broke. You use get_execution to pull deep-dive trace histories for any run, telling you exactly which task failed and why.

02 Audit complex systems faster than ever. The search_workflows tool lets you query months of historical data using filters like 'failed' or specific correlation IDs.

03 Understand your architecture without reading documentation. By listing all registered workflow definitions, architects can quickly map out the entire system flow.

04 Monitor real-time operations with minimal effort. list_running gives you a quick snapshot of every active instance for rapid operational checks.

05 Validate new services before deployment. You can inspect task definitions to ensure your microservice outputs match what the workflow expects.

Real-World Applications

The Order Failure Mystery

A user asks their agent, 'Why did order #XYZ fail last Tuesday?' The agent runs search_workflows, filters by ID and date, and uses get_execution to pinpoint that the payment task timed out at 14:23 UTC. The fix is immediate.

Debugging a New Feature

A developer wants to know how the new 'premium user' path works. They ask the agent to list_workflow_defs and get_workflow_def for that specific workflow, inspecting the branching logic without touching the staging environment.

Daily Operations Check

The ops team needs to know if any critical ETL pipelines are running. They use list_running, filtering by 'data-pipeline', and instantly see 3 active instances currently processing data.

Understanding Dependencies

A new architect joins the project and asks, 'What processes rely on user onboarding?' The agent runs list_task_defs and inspects related workflow definitions to map out dependencies for them.

Patterns to Avoid

Treating it like a simple API call

✗ AVOID

Just calling get_workflow_def with a name, but not specifying the version or scope. You get an incomplete picture of what's actually live.

✓ INSTEAD

Always start by listing all registered workflow definitions using list_workflow_defs to confirm the correct name and active version before trying to retrieve a specific definition.

Checking status in multiple places

✗ AVOID

Logging into the dashboard, then checking the terminal logs, then running a separate query—all just for one failure event.

✓ INSTEAD

Use search_workflows to combine all that data. Filter by 'status: failed' and retrieve the ID. Then use get_execution with that ID to see the full trace in one place.

Ignoring task limitations

✗ AVOID

Assuming a workflow can do anything just because it's running. It might fail because a required task definition is missing or outdated.

✓ INSTEAD

Always run list_task_defs first. Verify that all necessary components are registered and available before attempting to monitor any workflow runs.

The Right Fit

Use this MCP if your business logic relies on multi-step, asynchronous processes (microservices) and you need visibility into the *state* of those processes, not just their success/failure. You need to answer questions like 'What happened at step 4?' or 'Which runs are currently stuck?'. Don't use this if you only need to look up simple static data (e.g., a user list). For simple data retrieval, your agent should hit a dedicated CRUD API. But when the process itself is the key piece of information—the *flow* and its history—this MCP is required.

Debugging complex workflows used to mean dashboard fatigue.

When something went wrong in a multi-service workflow, you were forced into a tedious ritual: log into the monitoring UI for Service A, copy the correlation ID; switch tabs and check the logs for Service B using that same ID; then jump to the dedicated error management dashboard just to see if it timed out. It took fifteen minutes of clicking through disparate screens.

Now, your agent connects directly to the orchestration layer. You ask a single question —'What happened with order #123?'—and instead of links and blank fields, you get an immediate summary covering definitions, running instances, and the full execution trace for every service involved.

Orkes Conductor MCP gives your agent a single source of workflow truth.

You no longer have to manually correlate timestamps or jump between five different logging tabs. The ability to list all registered workflow definitions and inspect task definitions means you understand the architecture just by asking questions, not by reading dense diagrams.

The system moves from being a collection of siloed services monitored separately, to one cohesive flow that your agent can audit conversationally.

Orkes Conductor: 6 Tools for Workflow Management

Use these tools to query workflow states, retrieve process definitions, list active runs, and search through historical orchestration data.

#	TOOL	DESCRIPTION
01	`list_workflow_defs`	Lists all registered overarching workflow definitions available in your system.
02	`get_workflow_def`	Retrieves the full definition schema for a specific named workflow.
03	`list_task_defs`	Lists all individual task definitions that can be used within your workflows.
04	`list_running`	Provides a list of currently active workflow instances, allowing you to monitor what's executing right now.
05	`get_execution`	Fetches detailed state information for one specific workflow execution run.
06	`search_workflows`	Performs an advanced search across all historical workflow executions using filters like status or ID.

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 registered workflow definitions.

Found 12 workflow definitions. 1) 'order-processing' v3 — 8 tasks, including payment validation and shipping triggers. 2) 'user-onboarding' v2 — 5 tasks with email verification fork. 3) 'data-pipeline-daily' v1 — 4 tasks for ETL orchestration. Want me to inspect the graph of any of these?

U Are there any failed workflows in the last 24 hours?

Found 3 failed executions in the last 24 hours. 1) order-processing (ID: abc-123) — failed at 'payment-validation' task at 14:23 UTC. Error: 'Gateway timeout'. 2) user-onboarding (ID: def-456) — failed at 'send-welcome-email' at 16:45. Error: 'SMTP connection refused'. 3) data-pipeline-daily (ID: ghi-789) — timed out at 'extract-transform' at 03:12. Want me to get the full execution trace for any?

U How many instances of the order-processing workflow are currently running?

8 instances of 'order-processing' currently running. 5 are at the 'payment-validation' stage, 2 at 'shipping-trigger', and 1 at 'inventory-check'. Average execution time so far: 4.2 seconds. No instances are stuck or timed out. Want me to inspect one?

Frequently Asked Questions

01 **How does Orkes Conductor MCP help me debug failed workflows?**

It provides deep state details via get_execution. You don't just see 'failed'; you see the exact task that threw an exception and the error message, letting you know exactly what needs fixing.

02 **Can Orkes Conductor MCP track running services in real time?**

Yes. By using list_running, your agent pulls a live count of active instances for any specific workflow name. This is crucial for monitoring capacity and immediate status checks.

03 **Does Orkes Conductor MCP help me map out my system architecture?**

Absolutely. You can list_workflow_defs to see every major process, and get_workflow_def to inspect the task-by-task graphs, helping architects understand dependencies.

04 **What if I need to check history from last month? Does Orkes Conductor MCP support that?**

Yes. The search_workflows tool allows you to perform powerful searches across all historical executions using filters like status, type, or a correlation ID.

05 **Is this just for viewing data, or can it trigger workflows?**

This MCP is read-only. It's designed solely for monitoring and auditing; it lets your agent query definitions and states but cannot initiate new workflow runs itself.

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 `"orkes-conductor": { "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

Orkes Conductor 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 Orkes Conductor. 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	Orkes Conductor MCP
Server ID	`019d75ec-a510-70ae-a014-b050985fe6e9`
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/orkes-conductor.