MCP SERVER | NO CODE | CLOUD HOSTED

Conductor MCP for AI Agents

Orchestrate Complex Microservice Workflows from Natural Language

Conductor (Netflix OSS) MCP gives your AI client control over complex, multi-step business processes. It lets you define, execute, monitor, and manage entire workflows for microservices directly through natural conversation. Stop clicking around dashboards; start orchestrating critical systems from your agent.

A+ Quality Score 98.33/100

workflow-orchestration | microservices | netflix-conductor | automation | process-management

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

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.

Conductor (Netflix OSS) MCP

49 tools available
Cloud-hosted on Vinkius

Running distributed systems means managing dependencies that change constantly. This MCP connects your AI client to a powerful workflow engine, letting you treat complex process orchestration like simple chat commands. Instead of jumping between dedicated UIs to check task status or define new processes, you tell your agent what needs doing. You can list existing workflows and their versions, validate definitions before deployment, start new runs asynchronously, or even pause an entire sequence if something breaks down. If managing distributed tasks feels overwhelming, remember that Vinkius hosts thousands of MCPs, putting all enterprise systems into one place for your AI client to manage.

Core Capabilities

01 — Manage Workflow Definitions

List, get, create, update, and validate the structure of entire workflow definitions before they run.

02 — Execute Workflows on Demand

Trigger new instances of any defined workflow immediately, or start a specific task within a running process.

03 — Monitor Running Processes

Get the current state of active workflows, check queue depth for specific tasks, and retrieve detailed logs for individual steps.

04 — Control Workflow State

Pause, resume, retry failed tasks, bulk restart, or terminate entire workflow sequences at runtime.

05 — Batch Operations

Perform large-scale actions like updating definitions or managing multiple workflows with a single command.

One Click on Vinkius — From Prompt to Execution

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

01 Connect your AI client to this MCP and provide the Conductor Server URL.

02 Ask your agent to perform a high-level task, like 'check the status of user onboarding workflows.'

03 The MCP translates that request into specific function calls, retrieving the data or executing the necessary actions for you.

The bottom line is, it takes complex system operations and turns them into simple conversational commands.

Built For

This MCP is built for highly technical roles that deal with the plumbing of enterprise systems. If you spend time monitoring dashboards or manually updating YAML files to keep processes running, this tool saves hours of painful clicking.

DevOps Engineer

Running bulk commands like 'retry all failed payment workflows' without SSHing into a cluster.

Backend Developer

Triggering and validating test workflow definitions directly from the IDE to ensure new microservices integrate correctly.

System Architect

Querying the structure of complex processes, like listing all event handlers or getting task logs for troubleshooting design flaws.

What Changes When You Connect

01 Instead of manually listing every workflow version, use 'get_workflow_names_and_versions' to see all blueprints across your system at a glance.

02 Need to troubleshoot? Use 'get_task_logs' or check the state with 'get_workflow' to instantly understand exactly where and why a process failed.

03 Avoid manual deployments. With 'update_workflow_definitions', you can push changes to dozens of definitions in one go, instead of creating them individually.

04 When debugging a failure, don't restart everything. Use 'rerun_workflow' or 'retry_workflow' to target only the failed step for surgical fixes.

05 Save time managing state with bulk operations like 'bulk_pause' and 'bulk_resume', controlling entire sets of processes without touching a dashboard.

Real-World Applications

Onboarding a New Client Account

A new user signs up, triggering an onboarding workflow. The agent uses 'start_workflow' to kick off the process and monitors it with 'get_running_workflows', ensuring that profile creation, email verification, and initial billing setup all complete successfully.

Handling a Failed Payment Batch

A massive batch of payments fails due to an API timeout. Instead of manually checking each record, the agent uses 'bulk_retry' across 50 different transaction workflows and then checks the results with 'get_task_logs'.

Updating Core Business Logic

The billing department changes how tax is calculated. A developer uses 'validate_workflow_definition' to test the new JSON structure before committing, then uses 'update_workflow_definitions' to deploy the change across all related services.

Responding to a System Incident

A critical queue backs up. The architect asks the agent to check the depth with 'get_queue_size', then uses 'requeue_tasks' and 'bulk_resume' to clear the backlog and resume normal operations.

Patterns to Avoid

Over-relying on UI buttons

✗ AVOID

A developer manually clicks through 15 different tabs in the Conductor GUI just to check the status of three related processes and get their logs.

✓ INSTEAD

Instead, ask your agent to use 'get_workflow' for specific IDs or 'get_correlated_workflows' to see all linked statuses in one query.

Missing Definition Validation

✗ AVOID

A developer writes a new workflow definition and immediately saves it, only to find out later that the JSON structure was invalid when runtime fails unexpectedly.

✓ INSTEAD

Always run 'validate_workflow_definition' first. This checks the syntax without requiring you to commit or save any actual changes.

The Right Fit

Use this MCP if your pain point is managing processes that are inherently multi-step, stateful, and depend on multiple microservices completing tasks in order. If you need visibility into *how* data flows through a system—if you care about the sequence, dependencies, or failure points—this tool is essential. Don't use it if your workflow is simple (e.g., 'send email'). For single actions like that, simpler messaging integrations suffice. However, if you are building anything complex, involving multiple queue checks ('get_queue_size') or needing to track history across retries ('retry_workflow'), this MCP handles the full lifecycle management.

Conductor (Netflix OSS) MCP: Managing Distributed Workflow Blueprints

Today, updating a core business process means opening dozens of dashboards. You have to check the task logs in one place, validate the JSON

With this MCP, you describe the desired change—like 'update billing workflow V3'—and get immediate results. Your agent handles the

structure in another, and then manually update the definition version elsewhere just to deploy a small fix.

validation via 'validate_workflow_definition' and executes the necessary updates using 'update_workflow_definitions'. The process is conversational; the outcome is a live system fix.

Conductor (Netflix OSS) MCP: Monitoring Microservice Execution States

When an incident hits, you waste time figuring out which workflow instance failed and why. You have to run multiple searches just to gather the current state of all involved tasks and queues.

Now, simply ask your agent to 'get_running_workflows' or 'get_correlated_workflows'. It returns a consolidated view of everything active, allowing you to immediately use tools like 'pause_workflow' before the issue escalates.

49 Tools for Conductor (Netflix OSS) Workflow Management

Use these functions to manage workflow blueprints, monitor task queues, and control the entire lifecycle of distributed processes via natural conversation.

#	TOOL	DESCRIPTION
01	add_task_log	Adds a specific log entry to track activity within any given task step.
02	bulk_pause	Pauses multiple workflows simultaneously, stopping them from executing further.
03	bulk_remove	Deletes several workflow definitions or instances in a single operation.
04	bulk_restart	Forces multiple workflows to restart their execution from the beginning.
05	bulk_resume	Unpauses several workflows, allowing them to continue processing where they left off.
06	bulk_retry	Attempts to re-run a selection of failed or stalled workflows in batches.
07	bulk_search	Searches for workflow definitions using a list of specific IDs.
08	bulk_terminate	Stops and removes multiple running or defined workflows at once.
09	create_event_handler	Sets up a new event handler to listen for specific system events.
10	create_task_definitions	Creates one or more reusable definitions for microservice tasks.
11	create_workflow_definition	Defines a brand new, complex workflow structure.
12	delete_event_handler	Removes an existing event handler from the system.
13	delete_task_definition	Deletes a reusable task definition that is no longer needed.
14	delete_workflow_definition	Permanently removes an entire workflow blueprint from the system.

#	TOOL	DESCRIPTION
15	execute_workflow	Runs a full workflow execution immediately and waits for the result (synchronous).
16	get_all_queues	Retrieves the number of pending tasks waiting across all defined queues.
17	get_correlated_workflows	Finds workflows that are linked together using a common correlation ID.
18	get_event_handlers	Lists all active event handlers currently registered in the system.
19	get_queue_size	Checks how many tasks are waiting for a specific task type to run.
20	get_running_workflows	Retrieves the IDs of all workflows currently in an active execution state, grouped by their type.
21	get_task_definition	Fetches the details for a specific task definition using its name.
22	get_task_definitions	Retrieves a list of all available and defined microservice tasks.
23	get_task_logs	Fetches the detailed historical logs for a particular task execution.
24	get_workflow_definition	Retrieves the full structure and details of one specific workflow definition by name.
25	get_workflow_definitions	Gets a comprehensive list of all defined workflows available for execution.
26	get_workflow_names_and_versions	Lists every workflow name and all versions that have been created for it.
27	get_workflow_tasks	Shows the sequence of tasks required for a given running workflow instance.
28	get_workflow	Retrieves the complete status and history of a single, specific workflow run by its ID.
29	pause_workflow	Temporarily stops a specified running or defined workflow instance.
30	poll_batch_tasks	Checks the status of multiple queued tasks over time using long polling.
31	poll_task	Checks the current status of a single, specific task execution over time.

#	TOOL	DESCRIPTION
32	remove_workflow	Permanently deletes a workflow from the system registry.
33	requeue_tasks	Forces pending tasks back into the queue to be processed again.
34	rerun_workflow	Re-runs a workflow starting from a specific task that failed or needs revalidation.
35	restart_workflow	Resets and restarts an entire workflow execution back to its initial step.
36	resume_workflow	Continues a workflow that was previously paused, allowing it to complete naturally.
37	retry_workflow	Attempts to re-run only the most recently failed task in a sequence.
38	search_workflows	Searches for workflows using basic criteria and returns a summary of the results.
39	search_workflows_v2	Performs a detailed search across all workflows, returning full object details.
40	skip_task	Bypasses a specific task step in an active workflow instance.
41	start_workflow	Triggers a new workflow execution without waiting for immediate results (asynchronous).
42	terminate_workflow	Immediately stops and cancels a running workflow instance.
43	update_event_handler	Modifies the rules or targets of an existing event handler.
44	update_task_by_ref	Updates a task using its internal reference name for precise control.
45	update_task_definition	Modifies the operational details of a specific microservice task definition.
46	update_task	Updates the result data for a completed task step.
47	update_task_v2	Updates a task and automatically polls to find the next available task in the sequence.
48	update_workflow_definitions	Creates or updates multiple workflow definitions simultaneously (batch management).
49	validate_workflow_definition	Checks a workflow's JSON structure for errors without actually saving it to the system.

See It in Action

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

U Can you list all available workflows and tell me which ones need updating?

Workflow Definitions Found:

- `UserOnboarding` : v1.2 (Needs review)
- `PaymentProcessing` : v3.0 (Last updated 4 hours ago)
- `InventoryCheck` : v5.1 (Active/Stable)

We found three primary workflows. Version `user_onboarding:v1.2` is marked for review because the 'billing' task has been deprecated in favor of a new service. I recommend checking that definition.

U I need to check the logs for the payment processing workflow that ran yesterday.

Here are the key details for your request:

- **Workflow ID:** `8f2b3c4d-5e6f...`
- **Status:** Completed Successfully
- **Last Task Run:** `CreditCardValidation`

Log Snippet: *The task completed in 1.2s. Key output:* Transaction approved for $99.99.*

(Full log retrieval is available upon request.)

U Can we pause all the user onboarding workflows that are currently running?

Action: Pausing Workflows
Target: All active `UserOnboarding` instances.

✅ Successfully paused 14 workflow instances.

- The process will stop immediately at its current step.
- You can resume them later by asking to 'resume the onboarding workflows'.

Frequently Asked Questions

01 **How does Conductor (Netflix OSS) MCP help me troubleshoot a broken workflow?**

It gives you deep visibility into running processes. Instead of guessing, you can ask for the current state using 'get_workflow' or drill down with 'get_task_logs' to pinpoint exactly which step failed and why.

02 **Can I run a new workflow without having to manually trigger it?**

Yes. You can instruct your agent to use the 'start_workflow' function, kicking off complex processes asynchronously so you don't have to wait for immediate results.

03 **Is Conductor (Netflix OSS) MCP good for updating multiple workflows at once?**

It's excellent for bulk changes. Use 'update_workflow_definitions' or 'bulk_terminate' when you need to apply the same change, like version bumping or stopping old processes, across dozens of definitions quickly.

04 **What if a task fails midway through? Can I fix it?**

Absolutely. You don't have to restart everything. Use 'rerun_workflow' or 'retry_workflow' to target only the failed step, saving you time and computational resources.

05 **How do I ensure my new workflow definitions are correct before deploying them?**

You can use 'validate_workflow_definition'. This checks your JSON structure for errors without actually saving or changing anything in the live system, making deployments much safer.

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 `"conductor-netflix-oss": { "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

Conductor (Netflix OSS) 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 Conductor (Netflix OSS). 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	Conductor (Netflix OSS) MCP
Server ID	`019e387c-682b-71a1-958e-72fb873d8139`
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/conductor-netflix-oss.