MCP SERVER | NO CODE | CLOUD HOSTED

Daytona MCP for AI Agents

Automating Cloud-Based Development Environment Provisioning

Daytona manages ephemeral development environments through an MCP. It lets your AI agent orchestrate cloud sandboxes by creating, resizing, stopping, and starting entire dev workspaces on demand. Manage full environment lifecycles—from initial setup to deep debugging—all conversationally.

A+ Quality Score 98.33/100

sandboxes | dev-environments | workspace-automation | ephemeral-infrastructure | cloud-development

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/daytona-dev-workspaces

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.

Daytona (Dev Workspaces) MCP

28 tools available
Cloud-hosted on Vinkius

Need to spin up a fresh coding environment without leaving your chat or IDE? This MCP connects your development workflow directly to Daytona's infrastructure control plane. It lets you treat complex cloud management tasks like simple conversations with your AI agent.

You can provision standardized, temporary sandboxes instantly, giving your team consistent environments for testing and debugging. Need more power? You tell the agent, and it dynamically resizes resources like vCPU or RAM to match your workload requirements. If a sandbox gets into an error state mid-test, you don't restart from scratch; you simply ask the agent to recover it. The process includes full control over persistent storage—you can create volumes, manage snapshots for later use, and even fork existing environments if you need a specific baseline for testing. Because this MCP is hosted on Vinkius, your AI client connects once to access this tool alongside thousands of other enterprise capabilities.

Core Capabilities

01 — Provisioning and Scaling Sandboxes

Create new sandboxes, start stopped environments, stop running ones, delete old workspaces, resize resources, or fork existing instances.

02 — Snapshotting and Restoration

Take point-in-time backups of an environment's state using `create_snapshot`, and then use those snapshots to restore the workspace later via `activate_snapshot`.

03 — Persistent Storage Management

Create, list, retrieve, and delete volumes (`create_volume`, `get_volume`) ensuring that critical data persists even after the sandbox is terminated.

04 — Authentication Key Control

Manage all necessary API access tokens by listing, generating (`create_api_key`), or deleting keys to maintain a clean security posture.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/daytona-dev-workspaces** — connect your AI agent in three steps.

01 Subscribe to this MCP on Vinkius and provide your Daytona API Key.

02 Connect your preferred AI client (Claude, Cursor, etc.) to the Vinkius platform.

03 Ask your agent natural language questions like, 'Create a new sandbox with 4GB of RAM for Node.js testing' or 'What are my active development environments?'

The bottom line is you manage complex dev infrastructure using simple conversation.

Built For

This MCP solves the problem of environment drift and manual resource management for technical roles. It targets engineers who spend too much time clicking through dashboards or waiting for slow, manually provisioned testing sandboxes.

DevOps Engineer

Automating the spin-up, teardown, and scaling of dozens of test environments required for continuous integration pipelines.

Software Developer

Spinning up isolated coding environments on demand to test a feature without impacting their main local setup or requiring manual infrastructure tickets.

QA Engineer

Reproducing complex bugs by quickly creating clean, dedicated sandboxes from known-good snapshots for detailed debugging sessions.

What Changes When You Connect

01 Instant debugging environments: Quickly spin up dedicated sandboxes, allowing your agent to run isolated tests without manual infrastructure provisioning.

02 State preservation: Never lose progress. Use `create_snapshot` and `activate_snapshot` to save an environment's state right before a risky code change or test suite execution.

03 Resource optimization: Stop over-provisioning. The agent can automatically manage your compute by calling `resize_sandbox` when workloads increase, saving costs.

04 Zero-downtime debugging: If a sandbox fails mid-test, the agent doesn't quit; it uses `recover_sandbox` to bring the environment back online quickly.

05 Full visibility into infrastructure: Use `list_volumes` and `list_snapshots` to get an immediate inventory of all your long-term data stores and backups.

Real-World Applications

Debugging a production bug in a safe sandbox

A QA engineer discovers a bug only reproducible on the staging environment. Instead of filing an infrastructure ticket, they ask their agent to `fork_sandbox` from the last known good build and run diagnostic steps immediately.

Scaling up for seasonal load testing

A development team knows holiday traffic will spike resource usage. They instruct their agent to `list_sandboxes`, identify underpowered ones, and then use the `resize_sandbox` tool across the board before the peak period.

Reproducing a failure from last month

A developer is debugging an intermittent issue. They ask their agent to find snapshots taken around the time of the error and use `activate_snapshot` to restore a perfect, historical testing environment.

Cleaning up old development clutter

An ops engineer needs to decommission several old test workspaces. They ask their agent to `list_sandboxes`, identify the targets, and then run `delete_sandbox` on them in bulk, ensuring nothing gets missed.

Patterns to Avoid

Treating sandboxes as disposable

✗ AVOID

The developer manually deletes a sandbox without realizing it held critical, unbacked-up data needed for the next test phase.

✓ INSTEAD

Instead of deleting, ask your agent to `create_snapshot` first. This saves the environment's state before you delete or modify resources.

Ignoring resource limits

✗ AVOID

The CI/CD pipeline attempts a massive data load test but crashes because the sandbox only had 1 CPU and 2GB of RAM.

✓ INSTEAD

Use the `resize_sandbox` tool to dynamically increase resources (e.g., to 8 vCPU, 16GB RAM) before running high-load tests.

Mixing up data backups

✗ AVOID

A developer uses an old volume ID and tries to restore the wrong dataset, leading to corrupted test results.

✓ INSTEAD

Always use `get_volume` or `list_volumes` first. This confirms you have the correct resource ID before attempting any restoration.

The Right Fit

Use this MCP if your development process requires programmatic control over complex, ephemeral infrastructure like sandboxes and volumes. You need to manage the full lifecycle—creation, scaling (`resize_sandbox`), snapshotting, and eventual deletion. Don't use it if you only need basic file storage access; for that, a simple cloud storage connector is better. If your primary goal is just managing user identities or permissions, look at an identity management MCP instead. However, if your workflow involves testing code in isolated, controlled environments, Daytona provides the deep operational tools necessary to manage every resource component.

Daytona (Dev Workspaces) MCP for AI Agents: Orchestrating Sandbox Lifecycles

Currently, managing test environments is a nightmare of clicks. You have to jump between the cloud console, write up tickets, wait for allocation, and then manually scale resources whenever your test suite demands more RAM or CPU. This process slows down development dramatically.

With this MCP, you simply tell your agent what you need—'Give me a sandbox with 8 vCPU and enough memory to run the database.' The environment provisions instantly, ready for work. You get immediate control over the entire dev stack without leaving the chat interface.

Daytona (Dev Workspaces) MCP for AI Agents: Managing Persistent Data Volumes

Manually backing up or managing data volumes means tracking IDs and ensuring that when a sandbox is deleted, the critical persistent storage remains safe. This often involves copy-pasting resource identifiers across multiple dashboards.

Now, your agent handles it all. You can ask to `create_volume` or `list_volumes`, knowing you have full control over your long-term data assets separate from the temporary compute instances. It keeps your infrastructure clean and auditable.

28 Tools in the Daytona (Dev Workspaces) MCP for Sandboxes & Volumes Management

Use these tools to fully automate your development environment lifecycle, from creating new sandboxes to managing persistent storage volumes and API keys.

#	TOOL	DESCRIPTION
01	`activate_snapshot`	Activate a snapshot
02	`archive_sandbox`	Archive a sandbox
03	`create_api_key`	Create a new Daytona API key
04	`create_sandbox`	Create a new Daytona sandbox
05	`create_snapshot`	Create a new snapshot
06	`create_volume`	Create a new volume
07	`deactivate_snapshot`	Deactivate a snapshot
08	`delete_api_key`	Delete an API key by name
09	`delete_sandbox`	Delete a sandbox
10	`delete_snapshot`	Delete a snapshot
11	`delete_volume`	Delete a volume
12	`fork_sandbox`	Fork an existing sandbox
13	`get_api_key`	Get details of a specific API key by name
14	`get_current_api_key`	Get details of the currently authenticated API key
15	`get_sandbox_preview_url`	Get a signed preview URL for a specific port on a sandbox
16	`get_sandbox`	Get details of a specific sandbox
17	`get_snapshot`	Get details of a specific snapshot
18	`get_volume_by_name`	Get details of a specific volume by name
19	`get_volume`	Get details of a specific volume by ID

#	TOOL	DESCRIPTION
20	`list_api_keys`	List Daytona API keys
21	`list_sandboxes_paginated`	List all Daytona sandboxes (paginated)
22	`list_sandboxes`	List all Daytona sandboxes
23	`list_snapshots`	List all Daytona snapshots
24	`list_volumes`	List all Daytona volumes
25	`recover_sandbox`	Recover a sandbox from an error state
26	`resize_sandbox`	Resize sandbox resources
27	`start_sandbox`	Start a stopped sandbox
28	`stop_sandbox`	Stop a running sandbox

See It in Action

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

U I need a new environment to test the payment gateway. Can you create one with high memory?

Sandbox Provisioning Status:

✅ Sandbox ID: `dev-payment-03` created successfully.

- **CPU:** 4 vCPUs (Increased from default)
- **RAM:** 16GB
- **Disk:** 50GB
- **Status:** Running

I recommend running a test suite immediately. Remember to `create_snapshot` before you make major changes.

U The API key for the staging environment seems to be revoked. Can you check my keys?

API Key Inventory

- **Key Name:** `staging-api`
- **Status:** Active
- **Created By:** Agent (You)
- **Last Used:** 2 hours ago

I also see an expired key, `legacy-test`, which I recommend deleting to keep your security profile clean. Let me know if you want me to delete it.

U We found a bug in the old user profile service from last week. Can we restore that environment?

Snapshot Restoration Plan:

- **Target Snapshot:** `user-profile-20260515`
- **Status Check:** The snapshot is valid and contains necessary resources.
- **Action:** I am now initiating the restoration process. This will give you a clean, stable environment exactly as it was on May 15th.

Frequently Asked Questions

01 **How do I manage development sandboxes with the Daytona MCP for AI Agents?**

You control everything through conversation. You can ask your agent to create a new sandbox, scale its CPU resources up or down, and even take backups (snapshots) of it when you're ready to test something risky.

02 **Can the Daytona MCP for AI Agents help me debug old code?**

Yes. Instead of trying to replicate a bug manually, your agent can restore an environment from a previous snapshot—say, one from last week—so you have the exact conditions needed for debugging.

03 **What is the best way to keep my test data safe using Daytona MCP for AI Agents?**

Use persistent volumes. This keeps your core application data separate from the temporary compute environment, guaranteeing that your data survives even if you delete and recreate the sandbox.

04 **Is the Daytona MCP for AI Agents useful for large teams?**

Absolutely. It allows different team members to work in isolated sandboxes without interfering with each other's setup, ensuring everyone has a clean workspace on demand.

05 **How do I know if my current sandbox is properly configured?**

You can ask your agent to get the full details of the sandbox. It will return all metadata—CPU, RAM, disk size, and network status—so you always know exactly what resources are allocated.

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 `"daytona-dev-workspaces": { "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

Daytona (Dev Workspaces) 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 Daytona (Dev Workspaces). 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	Daytona (Dev Workspaces) MCP
Server ID	`019e3887-5aee-70bc-bcdc-e6ca00659153`
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/daytona-dev-workspaces.