MCP SERVER | NO CODE | CLOUD HOSTED

Nhost MCP

Manage Authentication, Storage, and Users in One Place

Nhost MCP manages all your backend needs—user sign-ins, profile lookups, file uploads, and session control—directly from any AI agent. Connect this MCP to instantly handle authentication flows (email/password, OTP, magic links), manage user accounts, and interact with cloud storage without leaving your development environment.

A+ Quality Score 98.33/100

authentication | backend-as-a-service | cloud-storage | user-management | serverless

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

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.

Nhost MCP

15 tools available
Cloud-hosted on Vinkius

This connector lets you build complex applications by giving your AI client direct access to Nhost's core services. Instead of writing boilerplate API calls every time a user signs up or needs their profile checked, your agent simply uses this MCP. You can programmatically manage everything from registering new users with `signup_email_password` to refreshing expired tokens using the `refresh_token`. Need to handle files? Your agent handles uploads via `upload_file`, and you can even retrieve necessary file metadata or download content through specialized tools like `get_file`. Because Vinkius hosts this MCP, your AI client gets all these backend capabilities in one place. This lets you test entire user journeys—like an anonymous login followed by a profile update—entirely within your development workflow.

Core Capabilities

01 — Manage User Authentication

Your agent can sign users into the system using email and password, one-time passwords (OTP), or magic links.

02 — Handle Account Registration

You can create new user accounts by providing an email and password combination, or via OTP methods.

03 — Retrieve User Profiles

The MCP fetches current user account details, allowing you to check statuses or retrieve specific profile data.

04 — Control Sessions and Tokens

You can generate new access tokens using a refresh token, or securely end active user sessions across all devices.

05 — Perform File Storage Operations

Your agent manages file uploads to your Nhost buckets and allows you to delete files when they are no longer needed.

One Click on Vinkius — From Prompt to Execution

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

01 First, subscribe to this MCP on Vinkius and provide the necessary Nhost Auth and Storage URLs from your project dashboard.

02 Next, tell your AI agent what action you need. For example, 'Sign in user X with Y password' or 'Upload image Z'.

03 The agent executes the required tool call, interacting directly with Nhost to perform the authentication, storage, or profile management task.

The bottom line is that you get a single point of control for all your user and file backend operations without writing any low-level HTTP requests.

Built For

This MCP is built for the full-stack developer who gets frustrated having to switch between their IDE, the terminal, and a web browser just to test an auth flow. It's essential for the DevOps engineer automating user provisioning or the support team needing quick account status checks.

Full-Stack Developer

You use this MCP to quickly prototype and test complete authentication flows, like signing up a new user using `signup_email_password` and immediately calling `get_user` to validate the profile data.

DevOps Engineer

You automate routine tasks, such as mass user provisioning or triggering password resets by invoking `reset_password` without manual intervention.

Support Technician

When a customer reports an issue, you use the MCP to verify their account status and confirm if session tokens need refreshing via `refresh_token`.

What Changes When You Connect

01 Test full user journeys instantly. You can simulate a sign-in using `signin_email_password`, then immediately check the profile status with `get_user`—all without changing tabs.

02 Handle complex credentialing. Don't just reset passwords; your agent can initiate that process by calling `reset_password` and guide you through the full flow.

03 Simplify file handling. Whether uploading new assets via `upload_file` or retrieving content with `get_file`, all storage operations are channeled through one consistent interface.

04 Improve session reliability. If a user's token expires, your agent can automatically call `refresh_token` to extend access without disrupting the workflow.

05 Streamline onboarding. You can register users using multiple paths—from simple anonymous login via `signin_anonymous` to formal registration with `signup_otp_email`.

Real-World Applications

The Onboarding Flow Test

A developer needs to verify that a new user sign-up works end-to-end. They prompt their agent: 'First, register user Jane Doe using email and password.' The agent uses `signup_email_password`, then follows up by calling `get_user` to confirm the profile was created correctly.

The File Management Audit

A DevOps team needs to clean up old assets. They ask their agent to list all files and delete a specific image using `delete_file`, followed by generating a secure, temporary link for review with `get_file_presigned_url`.

The Session Recovery

A support agent needs to check why a user's app suddenly logged them out. They use the MCP to call `refresh_token`, and if that fails, they can escalate by calling `signout` to confirm token invalidation.

The Data Ingestion Pipeline

A data scientist needs to upload a large configuration file. They instruct their agent to use the `upload_file` tool with base64 content, ensuring the file metadata is captured instantly for subsequent processing.

Patterns to Avoid

Treating it like a generic database CRUD service

✗ AVOID

Trying to use `get_user` just to fetch names and emails, when the tool actually provides detailed profile information tied specifically to Nhost's authentication context.

✓ INSTEAD

Always rely on the specific tools. If you need basic user info, use `get_user`. If you are onboarding a completely new account, start with `signup_email_password` first.

Ignoring session control

✗ AVOID

A developer signs in and then just assumes the connection is good for later testing. They fail to call `signout`, leaving stale tokens that might cause false-positive errors.

✓ INSTEAD

Always wrap up your test cycle by calling `signout` to invalidate refresh tokens, ensuring a clean state for the next test run.

Mixing file retrieval methods

✗ AVOID

Attempting to download a private file using a simple URL copy-paste. The connection will fail because it lacks proper authorization.

✓ INSTEAD

To access any protected asset, you must first call `get_file_presigned_url` to generate the temporary, authorized link for your agent to use.

The Right Fit

Use this MCP if your application logic relies on a unified source of truth for user identity and file storage. If managing user accounts (sign-up, sign-in, token refresh) or handling cloud assets is part of your core feature set, you need this connector. It abstracts away the complexity of Nhost's various endpoints. Don't use this if you only need to interact with a single domain—for instance, if you just need to read public data without authentication, another category tool might suffice. Also, don't rely on it for database querying; while it manages user records, specific data reads are better handled by dedicated data access tools.

Managing Backend Services Used To Be a Multi-Tab Headache

Think about the usual workflow: you log into your IDE to code. When an auth bug pops up, you have to open a separate browser tab to Nhost's dashboard. You check user credentials there. Then, if you need to test file upload limits, you switch yet another tab. It's constant context switching and copy-pasting of URLs.

Now, everything is in one place. Your AI agent handles the entire lifecycle—from initiating `signin_otp_email` to uploading a new image with `upload_file`. You stay focused on writing code while your agent manages the messy backend details for you.

Nhost MCP Gives You Full Control Over User Authentication

Before this, every change to a user's status—like changing their email or resetting a password—required specific API calls, managing tokens manually, and handling failure states across multiple endpoints.

Now your agent executes these actions in natural language. Need to reset credentials? Just prompt for it; the MCP handles the complexity of `reset_password` and guides you through the secure flow.

Nhost MCP: 15 Tools for Backend Ops

These tools let you manage every part of your backend stack, including authentication flows, user profiles, secure file storage, and session control.

#	TOOL	DESCRIPTION
01	`change_email`	Requests that a user's email address be updated, requiring elevated credentials.
02	`delete_file`	Permanently removes a specific file from your Nhost storage buckets.
03	`get_file_presigned_url`	Generates a temporary, secure link that allows controlled access to a private file.
04	`get_file`	Downloads the full content of an existing file, optionally applying image transformations.
05	`get_user`	Retrieves the profile information for the user currently signed into the system.
06	`refresh_token`	Generates a new JWT access token using an existing, valid refresh token.
07	`reset_password`	Initiates the process of sending a password reset email to the user's registered address.
08	`signin_anonymous`	Creates a new user session without requiring any credentials or personal information.
09	`signin_email_password`	Authenticates a user by submitting their email address and current password.
10	`signin_otp_email`	Starts the login process using an email-based one-time password (OTP).
11	`signin_passwordless_email`	Initiates a secure, click-through login by sending a magic link to the user's email.
12	`signout`	Invalidates all associated refresh tokens and ends the current user session completely.
13	`signup_email_password`	Registers a brand new user account using an email address and chosen password.
14	`signup_otp_email`	Creates a new user account by verifying their identity through an OTP sent to their email.

#	TOOL	DESCRIPTION
15	upload_file	Uploads file content, whether text or base64 encoded, and returns its associated metadata.

See It in Action

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

U Sign in user dev@example.com with password 'securePass123'.

I've initiated the sign-in process for dev@example.com using `signin_email_password`. The session has been successfully established.

U Show me the profile details for the current authenticated user.

Fetching profile... Using `get_user`, I've retrieved the account details: User ID 'usr_7721', Email 'dev@example.com', and the account is currently active.

U Upload a file named 'config.json' with content '{"theme": "dark"}'.

Processing upload... I've used `upload_file` to save 'config.json' to your Nhost Storage. The file is now available in your default bucket.

Frequently Asked Questions

01 **How do I start a session using Nhost MCP?**

You can start sessions in several ways. For basic testing, use `signin_email_password`. If you need maximum security, try `signin_passwordless_email` to generate a magic link.

02 **Can I check user details with Nhost MCP?**

Yes, the `get_user` tool lets your agent retrieve the current authenticated user's profile information, giving you immediate access to key account data.

03 **What if I need to delete a file using Nhost MCP?**

You use the `delete_file` tool. Your agent will handle the permanent removal of the specified asset from your storage bucket, making sure it's gone completely.

04 **How does Nhost MCP help with token management?**

Token management is handled by two primary tools: `refresh_token` extends an expired session, and `signout` ensures the user is logged out everywhere.

05 **Do I need specific credentials to use Nhost MCP?**

Yes. You must subscribe to this MCP with your Nhost Auth and Storage URLs provided by your project dashboard for it to function correctly.

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 `"nhost": { "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

Nhost 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 Nhost. 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	Nhost MCP
Server ID	`019e38c8-3fed-7102-867a-3d2d0b2d1ba1`
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/nhost.