MCP SERVER | NO CODE | CLOUD HOSTED

LibreChat MCP

Control your self-hosted agents directly from any client.

LibreChat MCP connects your self-hosted AI instances to any agent client. It lets you manage custom agents, list all available models, and generate chat completions using an OpenAI-compatible interface for your private LLM setups.

A+ Quality Score 100/100

llm-orchestration | chat-interface | self-hosted | model-management | ai-agents | unified-api

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

SCAN TO VIEW

The infrastructure that powers AI agents in the real world.

Vinkius connects AI to the world's software through secure, enterprise-grade infrastructure — enabling real-world execution at scale, built on the Model Context Protocol (MCP).

Your AI Connections Run Through Vinkius Cloud

The world's largest managed MCP catalog

Vinkius is the cloud infrastructure where AI agents connect 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.

LibreChat MCP

4 tools available
Cloud-hosted on Vinkius

Connecting your own LibreChat environment through this MCP gives your AI client direct control over your private language model ecosystem. Instead of relying on a single provider's features, you can treat your self-hosted agents like any other tool right inside Claude or Cursor. You first use the `login` tool to authenticate with your credentials and grab necessary access tokens. Once authenticated, you can list everything available using `list_models`. After that, you send requests via `chat_completions` to run chats against specific agents. Need structured data? The `open_responses` tool handles that too. This makes building complex agent workflows simple, letting your AI client talk directly to the models you set up yourself. It's a solid way to centralize access to multiple private LLMs through one secure interface on Vinkius.

Core Capabilities

01 — List Available Models

You can ask the MCP for a full list of every agent and model configured in your LibreChat instance.

02 — Run Chat Completions

The MCP sends messages to specific agents, generating responses that mimic standard chat completion APIs.

03 — Generate Structured Responses

You prompt the agent for output and receive a highly structured response using the Open Responses API format.

04 — Manage Authentication

The MCP allows you to log in directly with your email and password, obtaining required access tokens without needing static API keys.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to the LibreChat MCP and provide your Base URL. If you don't have an API key, use the `login` tool with your email and password.

02 Use `list_models` to confirm that all desired agents are visible to your AI client.

03 Send a prompt using either `chat_completions` or `open_responses`, targeting the specific agent ID you need.

The bottom line is, it gives your AI client programmatic access to run and manage every model in your self-hosted LibreChat environment.

Built For

This MCP is for the developer or engineer who runs their own private LLM infrastructure. If you spend time connecting different APIs just to keep your models running, this is for you.

AI Engineer

You use the MCP to connect complex, self-hosted agent systems into automated IDE workflows or production code.

DevOps Team Lead

You monitor and query model configurations across multiple private environments, ensuring consistent access for your teams.

Power User/Researcher

You need to centralize access to several niche or experimental LLMs without having dozens of API keys scattered everywhere.

What Changes When You Connect

01 You get centralized control over multiple private LLMs. Instead of managing separate API keys for every model, you connect once to the LibreChat MCP and gain access to everything.

02 The `chat_completions` tool lets you run complex chats against agents using a standard OpenAI-compatible interface, making integration painless for your developers.

03 Authentication is easier with `login`. You can log in directly using credentials (email/password) instead of relying solely on static API keys.

04 Structured data output is guaranteed via the `open_responses` tool. If you need JSON or a specific schema from an agent, this ensures your response format is consistent every time.

05 The `list_models` capability instantly shows you what agents are available, saving you the headache of guessing which IDs to use in your workflows.

Real-World Applications

Building an internal research assistant

A researcher needs their agent client to talk to five different specialized models. Instead of configuring five separate API endpoints, they connect the LibreChat MCP, run `list_models` to verify all agents are online, and then use `chat_completions` to route queries dynamically based on the prompt topic.

Automating compliance reporting

A DevOps team needs an agent to summarize operational data into a strict JSON format. They connect via LibreChat MCP and use `open_responses`, forcing the model output into a predictable, machine-readable structure that can be automatically ingested by another system.

Integrating legacy LLMs

A developer has an older, specialized agent running on custom hardware. They connect it through LibreChat MCP and expose its functionality via `chat_completions`, allowing modern tools like Cursor to interact with the old system without code changes.

Testing new agents quickly

A power user wants to test an experimental agent before deploying it widely. They first use `login` for quick authentication, then run `list_models` to find the specific ID, and finally execute a few test prompts using `chat_completions`.

Patterns to Avoid

Using single-purpose APIs

✗ AVOID

Trying to connect your agent client only through an 'OpenAI Completions' wrapper when you actually need access to specialized, self-hosted agents.

✓ INSTEAD

Use the LibreChat MCP. It provides a unified interface that accepts custom, private LLMs and exposes them through both `chat_completions` for chat flow and `open_responses` for structured data.

Hardcoding API keys

✗ AVOID

Storing static API keys in your client configuration file, which is a major security risk if the repo gets compromised.

✓ INSTEAD

Start with the `login` tool. It lets you authenticate securely using your email and password to obtain temporary access tokens for your session.

Assuming model availability

✗ AVOID

Writing code that calls an agent ID (`agent_xyz`) without first verifying if that agent is actually configured or online in the LibreChat instance.

✓ INSTEAD

Always run `list_models` first. This ensures your agent client knows exactly which models are currently available and ready to take requests.

The Right Fit

Use this MCP if your primary pain point is managing a diverse, self-hosted collection of AI agents or LLMs that aren't connected via one single service. This connector lets you treat multiple private systems (like custom internal tools) as one cohesive unit for your agent client. Don't use it if all the models you need are already hosted under a single commercial endpoint with no customization required, because then a generic API wrapper will suffice. If you only need basic chat completion and don't care about structured output or managing multiple agents, this MCP gives you too much power—and that's usually what you want.

The pain of model sprawl

Today, getting your agent client to talk to different private models is a nightmare. You have to jump between five different dashboards, copy-pasting keys and unique endpoints every time you want to test something new or run a report.

With this MCP, you plug it in once. Your agent client talks directly to the LibreChat MCP. You simply list your agents using `list_models`, then send all your prompts through one unified API call, regardless of which underlying model actually answers.

LibreChat MCP: Accessing Every Model

You eliminate the manual steps of finding agent IDs, managing disparate keys, and writing wrapper code for every unique LLM endpoint you use.

The result is simple: your agents feel like they live in one place. You get instant, programmatic access to your entire private model catalog.

LibreChat: 4 Tools for Agent Operations

These tools let you manage authentication, list available agents, send chat prompts, and force structured data output directly from your self-hosted LibreChat environment.

#	TOOL	DESCRIPTION
01	`chat_completions`	Creates a chat completion using the Agents API, allowing you to send prompts and get model responses.
02	`list_models`	Retrieves a list of all available models and agents configured in your LibreChat setup.
03	`login`	Authenticates you with your credentials, retrieving necessary access and refresh tokens for the session.
04	`open_responses`	Generates a structured AI output using the Open Responses API specification.

See It in Action

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

U List all available agents in my LibreChat instance.

I've retrieved the available agents. You have access to: 'Research-Assistant' (ID: agent_123), 'Code-Interpreter' (ID: agent_456), and 'General-Chat' (ID: agent_789).

U Login to LibreChat using my credentials.

Authentication successful. I have obtained the access and refresh tokens for your session. You can now proceed with querying models or creating completions.

U Ask agent_123 to summarize the latest trends in AI.

Sending request to agent_123... The agent responded: 'The current trends in AI include the rise of multi-modal models, autonomous agentic workflows, and increased focus on efficient small language models (SLMs)...'

Frequently Asked Questions

01 **How do I get started with the LibreChat MCP?**

You start by subscribing to the MCP and running the `login` tool using your credentials. This authenticates you and gives your agent client the necessary tokens for subsequent calls.

02 **Can I use chat_completions with agents that aren't OpenAI-compatible?**

Yes, that is the point of this MCP. It provides an OpenAI-compatible interface layer over your self-hosted LibreChat environment, making diverse models appear uniform to your client.

03 **Is there a way to check which agents are active in my LibreChat instance?**

Use the `list_models` tool. It provides an immediate overview of all available agents and model configurations, saving you from guessing IDs.

04 **What if I need the output data to be perfectly formatted JSON?**

For guaranteed structure, use the `open_responses` tool. This API is specifically designed to force your agent's response into a predictable format that downstream systems can read.

05 **Does this MCP require me to modify my core AI client code?**

No. You connect the LibreChat MCP through your existing compatible client (Claude, Cursor, etc.). The tool handles the complex routing and API translation for you.

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 `"librechat": { "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

LibreChat 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 LibreChat. 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	LibreChat MCP
Server ID	`019e38b7-ad28-7040-9c37-38d0f1a714b2`
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/librechat.