MCP SERVER | NO CODE | CLOUD HOSTED

ParseHub MCP

Control Web Scraping Runs via Chat Conversation

ParseHub connects advanced cloud scraping jobs directly into your AI workflow. List configured projects, dispatch headless runs, check crawler status in real time, and pull structured datasets via chat commands. Stop managing web scrapers through separate dashboards; control complex data collection right where you write.

A+ Quality Score 98.33/100

data-extraction | headless-browser | web-crawling | json-output | cloud-scraping | automation-workflows

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

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.

ParseHub MCP

10 tools available
Cloud-hosted on Vinkius

Web scraping used to mean logging into a dedicated dashboard, setting up parameters, hitting 'run,' then waiting for emails or refreshing pages until the data finally appeared. Now, you can manage that entire process inside your chat agent. This MCP lets you treat web crawling like any other function call. You can list all your existing projects—including their start URLs and templates. Need new data? Just dispatch a run job on command, specifying which project to use or even overriding the default starting URL. The system tracks everything, telling you if the job is queued or running. When it’s done, you don't just get a 'Success' message; you pull down secure, structured JSON arrays containing all the scraped payloads, ready for your agent to process.

Core Capabilities

01 — List configured projects

View every web scraping project saved in your account, including their unique tokens and template details.

02 — Start a data extraction run

Tell the MCP to trigger a new headless scrape job for any specified project.

03 — Target custom URLs

Start a scraping run that focuses on specific pages, bypassing the default starting URL for a project.

04 — Check run status and progress

Get real-time updates on whether a scheduled scrape is queued, running, or if it has completed successfully.

05 — Download extracted data payload

Retrieve the final structured JSON data from any completed scraping run for immediate use.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to this MCP and provide your ParseHub API key.

02 Ask your agent to list available projects, or specify a custom URL, so it can identify the correct job parameters.

03 Once you confirm the run details, the MCP executes the scrape. You then use subsequent commands to track status until the data is ready for extraction.

The bottom line is, your agent handles the entire sequence: setup, execution, monitoring, and final data retrieval from a single conversation thread.

Built For

Data Engineers who are tired of switching between scraping tools and their primary workflow. Research Analysts needing to run academic paper extractors without manual dashboard interaction. Marketing Intelligence pros who need immediate competitor pricing data.

Data Engineer

Trigger cloud scraping logic securely, then pipe the extracted JSON datasets directly into a subsequent processing tool or database connection.

Research Analyst

Kick off academic paper extractors via chat and wait for confirmation before having the agent digest the resulting structured data.

Market Intelligence Specialist

Fetch completed scrapers tracking competitor pricing logic, ensuring that JSON arrays are immediately available for comparative analysis.

What Changes When You Connect

01 You don't have to switch between the ParseHub dashboard and your agent. You trigger, monitor, and retrieve data—all within one chat session.

02 Need fresh data fast? Use `get_last_ready_data` to grab the absolute latest payload without having to track a specific run token first.

03 When you need to scrape different pages using the same template (like product categories), use `run_project_with_url`. It changes only the start page, not your extraction rules.

04 The system keeps track of everything. Use `get_run_details` to check if a job is queued or running without needing to refresh an external web app.

05 You can clean up old jobs and manage costs by using tools like `cancel_run` or permanently removing data with `delete_run`.

Real-World Applications

Monitoring Competitor Pricing Changes

A market analyst needs to know if a competitor changed its pricing structure. They ask the agent to run an extractor on the main product page, wait for `get_run_details` to confirm completion, and then use `get_run_data` to pull the structured JSON of all price points.

Processing a Batch of Articles

A research team has 50 articles on different websites. Instead of running 50 jobs manually, they ask the agent to use `run_project_with_url` for each unique URL, then collect all the resulting structured data into one payload.

Auditing Historical Scrapes

A data engineer needs proof of what was scraped last month. They ask the agent to `list_runs`, find a specific run ID, and confirm its contents using `get_run_data` before moving on.

Stopping an Overdue Job

A job gets stuck in an infinite loop. The user uses the agent to check the status via `get_run_details`, determines it's stalled, and immediately calls `cancel_run` to free up resources.

Patterns to Avoid

Assuming data is ready.

✗ AVOID

The user asks the agent for the final JSON payload right after running a job. The agent fails because the run status is still 'queued' or 'running', and `get_run_data` cannot be called yet.

✓ INSTEAD

Always check the progress first. Use `get_run_details` to monitor the job until the system confirms it is complete. Only then should you use `get_run_data`.

Ignoring project scope.

✗ AVOID

The user wants to scrape data from a new site but uses the default run command, which only targets the original project's starting URL and template.

✓ INSTEAD

If you need to target a completely different page or set of pages while keeping the same scraping rules, use `run_project_with_url`. This overrides the default start address.

Overwriting data accidentally.

✗ AVOID

A user repeatedly runs jobs without cleaning up old results, leading to a massive storage quota bill and confusion about which data is current.

✓ INSTEAD

Use `list_runs` first to identify the specific historical run you need. When finished with an old job, use `delete_run` to permanently free up that stored payload.

The Right Fit

Use this MCP if your primary goal is automated, multi-step web data extraction and structured JSON output. You're working with content on the public internet—like product pages, competitor sites, or academic journals—and you need to run complex, headless browser scraping jobs without leaving your AI chat interface. This is a full lifecycle tool: it lets you list projects, manage runs, check status (`get_run_details`), and finally pull the data (`get_run_data`).

Don't use this if:
1. You are extracting data from a database (use a dedicated SQL/NoSQL connector).
2. You just need to send a simple message or write text (use a messaging MCP).
3. You only need to validate the format of data you already have in

memory. For pure schema validation, use a type-safe tool like Pydantic AI instead.

The Grind: Web Scraping Used To Be a Juggling Act

Today, getting structured data from a website means opening the scraping dashboard in one tab. You have to manually configure the starting URL, hit run, then switch tabs every ten minutes to check if it's still running. If you need to change the target site, you restart the whole process and repeat those clicks.

With this MCP, your agent handles the entire cycle. You tell it what you need, and it manages the complex headless browser automation in the background. The result? Clean, structured data payloads appear directly for your agent to use—no dashboard refreshing required.

ParseHub: Structured Data Extraction Via Conversation

Manual steps that disappear include navigating project tokens, monitoring status codes across different UIs, and manually downloading ZIP files just to get a JSON array. These are all abstracted away.

Now you simply command the action. You use `run_project` to kick off the job, check it with `get_run_details`, and when ready, pull the exact data using `get_run_data`. It's one simple conversation.

ParseHub with 10 Tools

These tools let you manage the entire lifecycle of web scraping: listing projects, starting runs, tracking progress, and retrieving final, clean data payloads.

#	TOOL	DESCRIPTION
01	cancel_run	Stops a running or queued scrape job to free up cloud resources and prevent unnecessary charges.
02	delete_run	Permanently removes old scraping run history and associated data, helping you clean up your account storage quota.
03	get_project	Retrieves the full configuration details for a specific web scraping project token.
04	get_run_data	Downloads the final, structured JSON payload from a run only after it has been confirmed as complete and data-ready.
05	get_run_details	Checks the current status of a specific scrape job to determine if it's waiting in queue, running, or finished.
06	get_last_ready_data	Immediately fetches the latest completed data for a project without needing to track individual run tokens first.
07	list_projects	Lists all available web scraping projects in your account, providing unique tokens and status information.
08	list_runs	Provides a historical record of every run for a project, useful for auditing or finding specific past data points.
09	run_project	Initiates a new scrape job using the default start URL and template configured in an existing project.
10	run_project_with_url	Starts a scraping run targeting a specific, custom web address while maintaining all of the project's original extraction rules.

See It in Action

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

U Fetch the list of scrape projects I have on my ParseHub account.

Found 3 mapping projects in your account. 1: 'Pricing Monitor' (Token: t9zx... , runs: 4). 2: 'Real Estate Leads' (Token: t4ax...). 3: 'Competitor Blogs'. Want me to launch any of these into a run?

U Start a new run for project 't9zx...' and check its status.

Dispatched action `run_project` for target `t9zx...`. Wait, pulling status logic: The cloud system confirmed it. Run ID `run_k1l`. Current Status is 'queued'. The parser will begin explicitly extracting domains soon.

U Extract the finished data JSON payload from run ID 'run_k1l'.

Pulled exact constraints running `get_run_data`. The JSON indicates 40 records containing titles and strict price fields. Would you like me to process this data or format it differently?

Frequently Asked Questions

01 **How do I start a scrape if I want to use different pages?**

You use the `run_project_with_url` tool. This lets you target custom URLs while keeping all of your project's original scraping rules and template definitions intact.

02 **Can ParseHub MCP list what projects I already have?**

Yes, use the `list_projects` tool. It shows every web scraping project you've set up, giving you the unique tokens needed for subsequent commands.

03 **What if my scrape job fails? Can I stop it?**

You can monitor the status using `get_run_details`. If it's stalled or taking too long, use the `cancel_run` tool to safely stop the operation and free up resources.

04 **How do I get data from a run that finished yesterday?**

First, you should `list_runs` to find the specific ID. Once you have the ID for a completed job, use `get_run_data` to pull down the structured JSON payload.

05 **Do I need an API key for ParseHub MCP?**

Yep. You must subscribe and provide your ParseHub API Key during setup so the agent can authenticate and manage cloud scraping jobs on your behalf.

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 `"parsehub": { "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

ParseHub 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 ParseHub. 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	ParseHub MCP
Server ID	`019d75ef-66f9-73b1-9183-e89e904e7d83`
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/parsehub.

© 2026 Vinkius · vinkius.com · support@vinkius.com