MCP SERVER | NO CODE | CLOUD HOSTED

Browserbear MCP for AI Agents

Automate web scraping and visual testing workflows

Browserbear lets your AI agent handle complex browser tasks. It automates web scraping, runs visual monitoring checks, and executes multi-step workflows—all from natural conversation. You don't need to log into a dashboard; just tell your agent what to scrape or check.

A+ Quality Score 100/100

visual-monitoring | web-scraping | automation-tasks | screenshot-capture | form-automation | no-code

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

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.

Browserbear MCP

10 tools available
Cloud-hosted on Vinkius

Stop clicking through dashboards to monitor websites or gather data. This MCP connects your browser automation capabilities directly to your AI client. Now, you can use natural language instructions to orchestrate complex web scraping jobs and visual checks without ever leaving your chat window. Need to test if a login form broke? Just ask. Want to scrape product prices across ten competitor sites? Tell your agent. You get immediate access to all these powerful features through the Vinkius catalog, making it easier than ever to manage advanced automation from any MCP-compatible client. It's about running multi-step actions—like logging in, navigating three different pages, and then extracting specific data points—all via simple conversation.

Core Capabilities

01 — List available automation workflows

Retrieves a list of all saved browser automation tasks, giving you an overview of what the system can run.

02 — Create new automation tasks

Saves a multi-step process that your AI agent will execute later on demand.

03 — Execute and customize task runs

Triggers a specific saved task, allowing you to override parameters like the starting URL or form data for unique tests.

04 — Capture current web view visuals

Takes an immediate screenshot of any given URL at customizable dimensions for quick visual validation.

05 — Monitor job status and history

Checks the real-time progress of a running task or lists all completed automation runs for auditing purposes.

One Click on Vinkius — From Prompt to Execution

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

01 First, subscribe to this MCP and provide your Browserbear API Key.

02 Next, tell your AI client what you want to accomplish—for example, 'Take a screenshot of X URL at 1280×800' or 'Run task Y'.

03 Your agent executes the command, monitors the web action, and returns the resulting data, list, or visual confirmation.

The bottom line is that you use plain conversation to manage sophisticated browser automation jobs without needing manual API calls or dashboard logins.

Built For

This MCP is essential for QA Engineers who spend hours clicking through dashboards, Growth Marketers tracking competitors, and Developers who need complex web interactions integrated into their code base.

QA Engineer

You use this to quickly trigger visual regression tests across multiple URLs or check the results of a multi-step login flow without touching the manual testing dashboard.

Growth Marketer

You automate lead gathering campaigns by instructing your agent to scrape contact details from specific industry websites and compile them into structured lists.

Developer/Engineer

You integrate complex browser interactions, like form filling or data submission flows, directly into your coding environment using natural language commands.

What Changes When You Connect

01 Run multi-step tasks with full control. You can trigger a specific task, like 'Login Flow,' and dynamically override the starting URL or form data without manually accessing the dashboard.

02 Get instant visuals of any page. Instead of writing code to capture screenshots, you simply ask your agent to 'Take a screenshot' of a URL and get high-quality images back immediately.

03 Manage complexity with structure. You can use tools like 'list_tasks' to see all saved workflows, letting you build up complex testing sequences piece by piece using natural language.

04 Maintain full visibility into your work. Use the run history functions (like 'list_runs') to track every job executed and get detailed metadata about past automation attempts.

05 Save time on data collection. Rather than manually visiting multiple competitor sites, you can instruct your agent to scrape specific structured data points across a list of URLs.

Real-World Applications

Detecting broken user flows

A QA Engineer notices the checkout button isn't appearing on mobile. They tell their agent, 'Take a screenshot of the cart page at 400×600.' The agent runs the task and returns an image, confirming the visual bug instantly.

Competitor price tracking

A Marketing Manager needs to know if their top competitor changed pricing. They ask their agent to 'Run task' on a list of product pages, scraping the visible price for comparison across multiple sites in one go.

Auditing website performance

An engineer wants to see how a page looks after a major site redesign. They ask their agent to 'list_tasks' to select the visual check and run it, comparing the current output against saved historical screenshots.

Lead generation from niche sites

A user needs contact info for industry professionals. They instruct their agent to 'run task' on a specific directory site, overriding parameters to search by city, and scrape the resulting emails into a structured list.

Patterns to Avoid

Trying to hardcode scraping selectors

✗ AVOID

Writing brittle code that assumes an element's XPath will never change. When the website updates even slightly, your entire script breaks and you waste hours debugging.

✓ INSTEAD

Use this MCP instead. Tell your agent to 'create_task' with a high-level goal (e.g., 'scrape all visible product names') and let Browserbear handle the complex element targeting.

Manually tracking test cycles

✗ AVOID

Running five different tests, then going into the dashboard to manually copy IDs or check which one failed last week. It's tedious record-keeping.

✓ INSTEAD

Use 'list_runs' and 'get_task' through your agent. You get a summarized history of all runs and tasks in plain text format right where you are working.

Forgetting to check usage limits

✗ AVOID

Running dozens of tests back-to-back until the API hits rate limits, leaving your agent hanging and failing silently.

✓ INSTEAD

Always start by asking the agent to 'get_account_usage'. This confirms you have capacity before launching any large scraping or visual monitoring job.

The Right Fit

Use this MCP if your workflow requires interaction with a live browser, whether that's capturing visuals or extracting data from dynamically loaded content. It’s perfect for QA and marketing roles that deal with website state changes. However, don't use it just because you need to process static files; simple file transfers are better handled by general cloud storage tools. Also, if your entire job involves running tasks based on internal database records rather than external websites, this MCP won't help—you need a specialized CRM or ERP integration instead. If you only need basic API calls (like fetching user IDs), the dedicated API client is faster; but for anything that needs to 'see' a webpage, this is your tool.

Browserbear MCP: Automating Visual Regression Testing

Today, QA engineers spend hours manually navigating staging environments. They click through pages, comparing the current live view to last week's approved screenshot. This involves opening multiple browser tabs, taking screenshots one by one, and then manually uploading them into a tracking spreadsheet—a process that is slow, error-prone, and takes up entire afternoons.

With this MCP, you simply ask your agent to 'list_tasks' and run the pre-built visual check. The agent handles the login, navigation, and comparison in the background. You get an immediate pass/fail report with a detailed view of exactly which element changed—saving hours of manual clicking.

Browserbear MCP: Scaling Web Scraping for Market Research

Manual competitor monitoring means opening ten different tabs, logging into each site individually to find the pricing page, and then manually copying structured data like product names and SKU numbers into a single master sheet. This process is not only slow but guarantees human error every time.

Now, you instruct your agent with 'list_tasks' to scrape specific fields across multiple domains. The agent runs the sequence, collects all the target data points, and returns them as one clean, structured output ready for analysis.

Browserbear: 10 Tools for Web Automation and Scraping

Use these tools to create, execute, monitor, and track every aspect of your web automation workflow directly through your AI client.

#	TOOL	DESCRIPTION
01	`create_task`	Saves a new multi-step automation sequence that can be run later by the AI agent.
02	`delete_run`	Removes records of completed task runs from your history for cleanup.
03	`get_run`	Retrieves the current status and final results of a specific, running automation job.
04	`get_task`	Fetches all detailed metadata for one particular saved browser task.
05	`get_account_usage`	Checks and returns your current account usage statistics against established limits.
06	`list_projects`	Displays a list of all separate projects you have set up within the system.
07	`list_runs`	Gets a complete, paginated list of every recorded task run in your history.
08	`list_tasks`	Provides an overview and names of all saved browser automation tasks across all projects.
09	`run_task`	Initiates a run for a specified task, allowing you to override parameters like the target URL.
10	`take_screenshot`	Captures and returns an image of any given website URL at specific dimensions.

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 screenshot of our homepage right now, but I want it at 1920×1080 resolution.

Screenshot Run Initiated

- **Target URL:** `https://mycompany.com/`
- **Dimensions:** 1920×1080 pixels
- **Status:** Processing...

✅ **Complete!** The image is ready for review and attached below. Let me know if you need a different crop.

U Can you list all the tasks we have set up? I want to check on our scraping workflow.

Browser Automation Tasks Found:

Daily News Scraper: *(ID: task_1)* — Description:* Scrapes 5 articles from three major sources. **Last Run:** Yesterday.
Visual Check: *(ID: task_2)* — Description:* Takes screenshot of the checkout page. **Status:** Ready to run.
Login Flow: *(ID: task_3)* — Description:* Simulates user login and checks for success message.

U Run the Visual Check task, but change the target URL to our staging environment instead.

Task Run Triggered!

✅ **Confirmation:** Running 'Visual Check' (ID: task_2) with a dynamic override.
- **New Target URL:** `https://staging.mycompany.com/`
- **Monitoring Status:** Active... The agent is now capturing the view and will notify you when the results are ready.

Frequently Asked Questions

01 **How does Browserbear MCP help me scrape data from complex websites?**

This MCP allows your AI agent to run sophisticated web scraping jobs. You just tell it what data you need, and the agent executes the multi-step process—like logging in or navigating through several pages—to gather the structured information.

02 **Can Browserbear MCP check if my website design broke after an update?**

Yes. You can use this MCP to take high-quality screenshots of specific URLs and compare them against previous versions, instantly spotting visual bugs or layout shifts that happened during a redesign.

03 **I'm a QA Engineer; is Browserbear MCP good for visual testing?**

It's excellent. Instead of manually running tests on a dashboard, you can ask your agent to trigger the test run and get immediate status updates and visual confirmation right in your chat interface.

04 **What if I need to scrape data from different sources? Does Browserbear MCP support that?**

Absolutely. You build a sequence of tasks—for example, scraping three different competitor sites—and the agent runs them one after another, compiling all the results into one organized output for you.

05 **Does using Browserbear MCP require me to write any code?**

No. The entire process is managed through natural language conversation with your AI client. You just describe what needs to happen, and the agent handles the complex web interactions in the background.

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 `"browserbear": { "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

Browserbear 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 Browserbear. 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	Browserbear MCP
Server ID	`019d7564-77cf-7143-8c8a-20eeb310ba1e`
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/browserbear.