MCP SERVER | NO CODE | CLOUD HOSTED

LoadNinja (Real-Browser Load Testing) MCP

Talk to your agent instead of clicking dashboards.

LoadNinja (Real-Browser Load Testing) MCP gives your AI agent full control over performance engineering. Manage complex load tests—from listing saved scenarios to triggering live, high-volume traffic runs—all through natural conversation. Get detailed metrics on throughput, peak user counts, and transaction times without ever touching a dashboard.

A+ Quality Score 98.33/100

load-testing | performance-engineering | browser-automation | scalability | stress-testing

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/loadninja-real-browser-load-testing

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.

LoadNinja (Real-Browser Load Testing) MCP

10 tools available
Cloud-hosted on Vinkius

Need to know if your website handles 10,000 users at checkout? Instead of spending hours navigating dashboards and manually exporting CSVs, you just talk to your agent. This MCP connects your AI client directly to LoadNinja's performance infrastructure. You can ask it to list all existing load scenarios or trigger a new test run instantly, specifying the virtual user count and duration in minutes. After the test finishes, you don't sift through raw data; you simply ask for an analysis of throughput, peak usage, and transaction timing. The agent handles the complexity, giving you actionable performance insights right where you work. By connecting this MCP via Vinkius, your entire team gains a single point of access to manage all load testing needs, whether you're running basic smoke tests or full-scale stress simulations.

Core Capabilities

01 — List test scenarios

Shows every saved performance test scenario you have set up in LoadNinja.

02 — Get specific scenario details

Retrieves the full configuration and target URL for any single load test scenario.

03 — Run a live load test

Starts a new performance run, specifying exactly how many virtual users and for what duration.

04 — List all past runs

Shows the status of every test execution to help you track if it's completed or still running.

05 — Analyze run summaries

Gathers high-level performance data, like average throughput and peak user counts, for finished tests.

06 — Get raw performance stats

Extracts the deep, underlying metrics and server response times to pinpoint specific bottlenecks.

07 — Stop active test runs

Immediately cancels a running load test to manage system resources or halt an error cycle.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/loadninja-real-browser-load-testing** — connect your AI agent in three steps.

01 Subscribe to this MCP and provide your LoadNinja API key credentials.

02 Your AI client uses these keys to connect directly to the LoadNinja performance platform.

03 You simply ask your agent a question, like 'Run the checkout scenario with 500 virtual users for 15 minutes,' and it executes the command.

The bottom line is that you manage complex, global-scale load testing by talking to your AI client instead of clicking through multiple dashboards.

Built For

Performance engineers and QA teams who are sick of context switching. If you spend more time exporting data and formatting reports than actually analyzing performance, this is for you.

Performance Engineer

Needs to trigger complex load scenarios across various browsers and analyze the resulting throughput metrics without leaving their terminal.

QA Automation Tester

Requires a quick way to check if a performance regression happened after a code deploy by monitoring test run completion status directly in the chat window.

DevOps Engineer

Needs to audit global infrastructure limits, checking available virtual user counts and physical data center locations before deploying a major service change.

What Changes When You Connect

01 Instant control: Instead of navigating complex dashboard menus, you simply tell the agent to 'Stop test run X.' This uses the `stop_test_run` tool, saving minutes of manual cleanup time.

02 Deep diagnostics: When a bottleneck pops up, don't guess. Use the `get_test_run_stats` tool to pull raw performance statistics and find exactly which server response time failed.

03 Global visibility: Need to know if you can scale internationally? The agent uses `list_locations` and `get_account` to show available physical data center points and your current VU limits instantly.

04 Full scenario management: Forget digging through folders. You use the `list_scenarios` tool to see every saved test configuration and `get_scenario` for its full details, all in one conversation.

05 Controlled execution: Running a new load test is simple. Just ask the agent to trigger the run with specified users using the `run_scenario` tool, and you're live immediately.

Real-World Applications

Checking for holiday traffic readiness

A QA tester needs to know if the checkout process can handle a major sales spike. They ask their agent: 'Run the Main-Store-Checkout scenario with 1000 virtual users for 30 minutes.' Once done, they use `get_test_run_stats` to verify that average throughput stayed above 50 requests/second.

Debugging a regional outage

A DevOps engineer notices latency spikes in Europe. They ask their agent to list all physical data center locations, see the European options, and then use `run_scenario` targeting that specific region to isolate the bottleneck.

Verifying account capacity

A team lead is preparing for a major launch. They first ask their agent to run `get_account` to verify their current subscription limits, ensuring they don't exceed the maximum virtual user count before starting any tests.

Quickly reviewing last week's metrics

A performance engineer needs a quick summary of the previous day's run without pulling up historical dashboards. They ask for the summaries using `list_test_runs` and then request analysis via `get_test_run`.

Patterns to Avoid

Manually comparing reports

✗ AVOID

Downloading five separate CSV files from different dashboards (e.g., 'Run A Throughput' vs. 'Run B Latency') and spending an hour correlating the P95 times in Excel.

✓ INSTEAD

Instead, ask your agent to execute two runs and then prompt it: 'Compare the raw performance stats for Run A and Run B.' The agent handles the comparison instantly using `get_test_run_stats`.

Assuming test status

✗ AVOID

Walking away from your computer after initiating a massive 60-minute load test, only to return later and find out if it finished or failed by clicking through the dashboard list.

✓ INSTEAD

Use `list_test_runs` first. The agent gives you the real-time status of all executions. You can then follow up with 'Show me the full details for run ID XYZ' using `get_test_run`.

Trying to test without context

✗ AVOID

Starting a high-volume test without knowing if you have enough resources or which browser type is best suited for the target audience.

✓ INSTEAD

Always check your limits first. Run `get_account` and then use `list_browsers` before initiating any load cycle to ensure global scalability.

The Right Fit

Use this MCP if your primary bottleneck is moving from *data collection* to *data insight*. If you are drowning in performance metrics, logs, or dashboard tabs, and you need a conversational layer on top of that data, this is perfect. It excels at the full lifecycle: list scenarios -> run test -> get stats. Don't use it if your problem is simple configuration management; for instance, if you just need to set up a new user account in LoadNinja, that's not what this MCP does. Stick to analyzing and controlling existing runs. If you only need to check available data center locations without running anything, `list_locations` works great, but the power is in the full test cycle.

The Dashboard Overload

Today, performance testing means opening a browser, finding the 'Scenarios' tab, manually selecting which test to run, and then clicking through five different tabs—Metrics, Logs, Summary, etc. You end up with multiple dashboards showing slightly different views of the same data, forcing you to copy-paste numbers into an external sheet just to draw a conclusion.

With this MCP, those manual clicks vanish. You simply tell your agent what test you need and how many users to simulate. When it's done, instead of opening five tabs, you ask one simple question—like 'What was the P95 transaction time?'—and the answer appears immediately in the chat.

LoadNinja (Real-Browser Load Testing) MCP

You no longer have to open a separate dashboard, remember which report ID you used last week, or manually check if your test is still running. The agent tracks the status for you and provides summaries of completed runs directly in the conversation.

It changes how fast you can respond to failure. You don't wait an hour to export data; you ask for `get_test_run_stats` and get the raw metrics instantly, allowing your team to fix bottlenecks much faster.

LoadNinja (Real-Browser Load Testing) 10 Tools

Use these tools to control every aspect of load testing, from listing saved scenarios to triggering complex performance simulations.

#	TOOL	DESCRIPTION
01	`list_scenarios`	Shows you every saved load test scenario in your account.
02	`get_scenario`	Retrieves the full configuration details, including target URLs, for one specific load test scenario.
03	`run_scenario`	Starts a new performance run by executing a saved scenario with specified virtual users and duration in minutes.
04	`list_test_runs`	Lists all previous test executions, showing their current status (running or completed).
05	`get_test_run`	Retrieves the basic details and summary information for a specific completed test run.
06	`get_test_run_stats`	Pulls the raw performance statistics, including server response times, for deep analysis of a single test run.
07	`stop_test_run`	Immediately and irreversibly cancels an active load test to prevent unnecessary resource usage.
08	`list_browsers`	Lists all explicit native browser types that can be used for load testing.
09	`list_locations`	Lists the physical data center locations available for global load injection testing.
10	`get_account`	Provides your account's current subscription details and strict maximum virtual user limits.

See It in Action

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

U List all load test scenarios in my LoadNinja account

I've found 4 scenarios: 'Main-Store-Checkout' (ID: scen-123), 'User-Login-Flow' (ID: scen-456), 'Inventory-Search-API' (ID: scen-789), and 'Mobile-App-Smoke' (ID: scen-012). Which one would you like to run or inspect?

U Run scenario 'scen-123' with 100 VUs for 10 minutes

Triggering load test scenario 'scen-123'... I've started the run with 100 virtual users for a duration of 10 minutes. New Run ID: 'run-98765'. I'll monitor the progress and notify you once the performance metrics are available.

U Show me the performance stats for the last completed run

Retrieving metrics for run ID 'run-13579'... The test achieved a peak of 250 VUs with an average throughput of 45 requests/sec. The P95 transaction duration was 1.2s. No major bottlenecks detected. Would you like to see the raw metrics breakdown?

Frequently Asked Questions

01 **How do I list all load test scenarios using LoadNinja (Real-Browser Load Testing) MCP?**

You ask the agent to 'list all load test scenarios.' The tool uses `list_scenarios` to show you every saved performance configuration in your account.

02 **Can I trigger a live test run with LoadNinja (Real-Browser Load Testing) MCP?**

Yes. You tell the agent exactly which scenario, how many virtual users, and for what duration you want to test by calling `run_scenario`.

03 **What is the difference between getting run details and raw stats with LoadNinja (Real-Browser Load Testing) MCP?**

Use `get_test_run` for high-level summaries like overall success. Use `get_test_run_stats` when you need deep, raw performance statistics to pinpoint a specific bottleneck.

04 **How do I check my account limits with LoadNinja (Real-Browser Load Testing) MCP?**

The agent runs the `get_account` tool. This instantly verifies your current subscription details and strict maximum virtual user counts so you don't over-allocate resources.

05 **If a test is running badly, how do I stop it with LoadNinja (Real-Browser Load Testing) MCP?**

Simply ask the agent to terminate the run. It executes `stop_test_run`, which immediately and irreversibly vaporizes the active load cycle.

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 `"loadninja-real-browser-load-testing": { "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

LoadNinja (Real-Browser Load Testing) 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 LoadNinja (Real-Browser Load Testing). 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	LoadNinja (Real-Browser Load Testing) MCP
Server ID	`019d75c9-5e92-72e7-8ba3-c2b020f2a232`
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/loadninja-real-browser-load-testing.