MCP SERVER | NO CODE | CLOUD HOSTED

Gatling MCP

Automate Load Testing Management Via AI Chat

Gatling MCP manages complex performance testing cycles right from your AI client. You can list simulation scenarios, kick off new load runs, track Virtual User counts as they spike, and pull detailed metrics like request error rates—all through natural conversation. It lets you manage everything from team quotas to resource pools without touching a dashboard.

A+ Quality Score 100/100

load-testing | performance-testing | simulation | stress-testing | automation | ci-cd-integration

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

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.

Gatling MCP

10 tools available
Cloud-hosted on Vinkius

Performance testing used to mean opening the dedicated platform, navigating deep into menus just to start a test run or check if it failed halfway through. Now, your agent handles that whole workflow for you. You tell it what load scenario needs running—maybe 'Search-API-Performance'—and it triggers the job on Gatling Enterprise infrastructure. Then, instead of refreshing pages and hunting down run IDs, you simply ask it to track progress or pull detailed stats like total requests and error counts. It’s about taking full control of your high-scale load simulations using plain language. By connecting your account through Vinkius, you gain access to a complete set of tools that covers everything from auditing resource pools to stopping an overrunning test immediately. This capability lets QA engineers and DevOps teams manage the entire performance lifecycle in one chat window.

Core Capabilities

01 — Kick off load tests

Start new Gatling simulations on the Enterprise platform, getting a unique run ID back instantly.

02 — Monitor and audit runs

Track the progress of running or finished tests, including peak Virtual User counts and overall execution status.

03 — Stop runaway simulations

Immediately abort a live load test run to save system resources when something goes wrong.

04 — Review detailed metrics

Retrieve full test statistics, including request counts, error rates, and start/end times for deep analysis.

05 — Manage team capacity

List registered teams and check member counts or credit quotas to ensure you don't hit usage limits.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to this MCP on Vinkius and enter your Gatling Enterprise API Token.

02 Connect your AI client (like Cursor or Claude) to the MCP endpoint using that token.

03 Use natural language prompts with your agent to list simulations, start a test run, or request performance statistics.

The bottom line is you manage complex load testing workflows through simple chat commands rather than navigating multiple web dashboards.

Built For

This MCP serves QA Engineers and SRE teams who are tired of manual dashboard monitoring. If your job involves repeatedly launching, tracking, or auditing performance tests across multiple systems, this is for you.

QA Performance Engineer

They use the MCP to trigger load tests and monitor simulation results in real time without clicking through dozens of tabs.

Site Reliability Engineer (SRE)

They audit team quotas, list load generator pools, and verify capacity using natural language to plan scaling efforts.

DevOps Lead

They test and debug performance scenarios, checking request stats and managing artifacts across the entire deployment lifecycle.

What Changes When You Connect

01 Manage the entire test lifecycle without leaving your chat window. Need to start a run? Just ask your agent, and it triggers the job on Gatling Enterprise.

02 Stop wasting time checking dashboards. You can track progress or retrieve full details for any test run using the `get_run` tool, giving you immediate status updates.

03 Audit capacity easily. Check resource limits by listing load generator pools (`list_pools`) and verifying team quotas (`list_teams`)—all done with a simple prompt.

04 Control your environment instantly. If a test runs too long or hits an unexpected spike, use `abort_simulation` to shut it down immediately.

05 Deep dive into results. Beyond just pass/fail, you can get detailed stats for any run, checking request counts and error rates crucial for debugging.

Real-World Applications

Need to check if the new checkout flow is stable?

The QA engineer asks their agent: 'List simulations' to confirm the correct scenario ID, then runs `start_simulation` with that ID. Finally, they ask for stats on a specific run ID using `get_run` to verify the error rate is below 0.1%.

The deployment team needs to know if we have enough capacity.

A DevOps lead uses the agent to call `list_pools`. This confirms that the required regions are available and lists the current instance counts, allowing them to validate scaling readiness before a major release.

We suspect a test run is going rogue and consuming too many credits.

An Ops Manager immediately tells their agent: 'Abort that simulation.' This uses `abort_simulation` to cut power to the generators, preventing accidental credit overruns.

The team needs to plan for next quarter's peak traffic.

A Performance Architect uses the MCP to run a baseline test, then uses `list_teams` and `get_run` repeatedly to gather metrics and ensure the current allocated quotas can handle projected growth.

Patterns to Avoid

Treating it like a simple data lookup

✗ AVOID

Manually calling 'list simulations' only when you forget the IDs, or using the tool just to read names.

✓ INSTEAD

Use your agent to manage the entire flow. Start by listing all scenarios with `list_simulations`, then immediately pass that list to start a run via `start_simulation`—it's a connected workflow, not a series of isolated clicks.

Forgetting the context (which test are we talking about?)

✗ AVOID

Asking 'What were the stats?' without specifying which simulation or run ID.

✓ INSTEAD

Always reference IDs. If you want metrics, first use `list_runs` to find the specific run ID, then follow up with `get_run` and provide that unique identifier.

Running tests without checking resources

✗ AVOID

Launching a massive load test only to discover later that the allocated generator pool is already maxed out.

✓ INSTEAD

Before starting, check capacity first. Run `list_pools` and verify the instance count in your target region using this MCP.

The Right Fit

Use this MCP if your performance testing workflow involves continuous monitoring, state management (starting/stopping tests), or auditing system resources like team quotas and load generator capacity. If you need to check a single, static piece of data—like just listing the names of all available teams—this is perfect. However, don't use it if all you want is to view raw logs from an external source that isn't Gatling Enterprise. Also, this MCP cannot simulate user interaction; it simulates system load at the API level. If your primary need is checking a development database schema against test results, you might be better off with a specialized data validation tool instead.

Dashboard fatigue is real.

Right now, managing a performance run means jumping between tabs: the dashboard to kick it off, another tab to copy the unique run ID, and then constantly refreshing status pages until you see if it succeeded or failed. If you need to pause it, you have to find the 'stop' button somewhere buried in the settings.

With this MCP, you talk to your agent instead. You say, 'Start a test on Scenario X.' The job runs. When it finishes, you just ask, 'What were the results?' and the data shows up immediately. It turns 15 minutes of clicking into one conversation.

Gatling MCP: Full-Cycle Load Testing Management

You no longer need to manually check team quotas or verify available load generator pools before planning a major test. You can ask the agent to list all teams and then list the resource pools in one go, ensuring you're scoped correctly.

The difference is control. Instead of being limited by what buttons are visible on a dashboard, you use natural language commands like `abort_simulation` or `list_packages`, giving your testing process full, conversational command authority.

Gatling MCP: 10 Tools for Performance Testing

These tools let you perform every action needed to manage your load testing lifecycle, from listing simulations to retrieving detailed run metrics.

#	TOOL	DESCRIPTION
01	`list_simulations`	Lists all active load scenarios on Gatling Enterprise, providing their IDs, names, and associated teams.
02	`get_simulation`	Retrieves complete details for a specific Gatling simulation scenario.
03	`start_simulation`	Initiates a new performance test run on the Gatling Enterprise infrastructure and returns the unique run ID.
04	`abort_simulation`	Immediately halts any running Gatling simulation to manage resources or prevent overruns.
05	`list_runs`	Retrieves a list of past and active runs for a given simulation ID.
06	`get_run`	Fetches the complete details, status, and metrics for a specific test run.
07	`list_teams`	Lists every team registered within your Gatling Enterprise account.
08	`list_packages`	Lists all uploaded test packages or artifacts, helping you verify versions and upload dates.
09	`list_tokens`	Displays existing API tokens configured within Gatling Enterprise.
10	`list_pools`	Retrieves a list of available load generator pools, showing regions and instance counts for scaling checks.

See It in Action

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

U List all simulations on Gatling Enterprise

Retrieving simulations... I found 3 active scenarios: 'User-Login-Stress-Test', 'Search-API-Performance', and 'Checkout-Flow-Baseline'. Which one would you like to start or view history for?

U Start simulation 'abc-123'

Simulation run started! I've triggered 'abc-123' on the Enterprise infrastructure. Your run ID is 'run_xyz789'. I'll monitor the progress and notify you when the virtual user peak is reached.

U Show me the stats for run 'run_xyz789'

Retrieving stats for run_xyz789... The test achieved a peak of 500 Virtual Users. Total requests: 12,500. Error rate: 0.05%. Average response time: 120ms. Would you like a breakdown by request type?

Frequently Asked Questions

01 **How do I start a test run with the Gatling MCP?**

You use the `start_simulation` tool. You just need to tell your agent which simulation scenario you want to run, and it handles triggering the process.

02 **Can I check my team's credit usage with the Gatling MCP?**

Yes, you can use `list_teams` to view registered teams. This helps you monitor member counts and verify quotas before running large load tests.

03 **What if a test run is going too far? How do I stop it?**

If you need to halt an active simulation, use the `abort_simulation` tool. This stops the generators immediately, saving resources and preventing overruns.

04 **Does Gatling MCP only show success or failure? What about metrics?**

No, it provides deep metrics. After using `get_run`, you get full stats including total requests, error counts, and average response time for detailed debugging.

05 **How do I see what test packages are available to use?**

You can list your artifacts by calling `list_packages`. This shows the names, versions, and upload timestamps of all uploaded materials.

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 `"gatling": { "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

Gatling 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 Gatling. 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	Gatling MCP
Server ID	`019d75a2-ce69-73cd-beb3-ed3ab14cb6b3`
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/gatling.