MCP SERVER | NO CODE | CLOUD HOSTED

BlazeMeter MCP for AI Agents

Automate continuous performance testing and monitor API throughput metrics

The BlazeMeter MCP automates continuous performance testing by letting your AI agent manage cloud load tests, workspaces, and metrics directly through chat. It lets you trigger stress tests, analyze throughput KPIs like p90/p99, and safely shut down runaway connections—all without switching dashboards.

A+ Quality Score 100/100

performance-testing | load-testing | stress-testing | continuous-testing | qa-automation | infrastructure-monitoring

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

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.

BlazeMeter MCP

10 tools available
Cloud-hosted on Vinkius

Managing large-scale performance tests used to mean context switching: jumping between your CI/CD pipeline, the testing console, and a metrics dashboard. Now, you can keep everything in one place. This MCP connects BlazeMeter's full suite of enterprise load testing capabilities directly to any AI client. Your agent handles everything from listing available workspaces and projects to executing complex load tests against your target APIs. It reads live run data, giving you critical throughput reports (p90/p99 KPIs) instantly. Need to stop a test immediately? You can force shut down runaway connections with a single command. If you're already using the Vinkius catalog for other services, adding BlazeMeter centralizes your entire DevOps toolchain under one roof.

It lets SREs run rapid baseline tests and QA teams verify JMeter structures, all while staying in their natural language environment.

Core Capabilities

01 — Inventorying Testing Resources

List all available workspaces, projects, and structural user metadata within the BlazeMeter platform.

02 — Executing Load Tests

Start cloud-based performance tests using configured JMeter definitions to simulate real-world load on your system.

03 — Monitoring Live Test Runs

Query the operational health of active master runs and retrieve precise throughput reports, including p90 and p99 metrics.

04 — Managing Master Connections

Enumerate attached structured rules and check the status of gateway run validations for critical systems.

05 — Stopping Active Tests

Forcefully shut down active cloud connections or runaway master runs to protect your source architecture during testing.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to this MCP and input your BlazeMeter Key ID and Secret credentials.

02 Connect your preferred AI client (Claude, Cursor, etc.) to the Vinkius catalog using these stored credentials.

03 Issue natural language commands through your agent, allowing it to execute complex load generation validations directly against the platform.

The bottom line is that your AI agent translates high-level instructions into precise API calls, giving you immediate control over your entire performance testing lifecycle.

Built For

This MCP is built for technical roles who need deep visibility and active control over infrastructure health. It's perfect for the SRE who gets frustrated switching between dashboards to check p99 latency, or the QA engineer who needs to trigger a test run mid-CI/CD pipeline without leaving their IDE.

Site Reliability Engineer (SRE)

Runs rapid baseline tests and monitors dynamic health metrics like p99 throughput, preventing downtime before it hits production.

QA Automation Team Lead

Queries active JMeter testing structures securely via chat during continuous integration cycles to validate system performance.

DevOps Engineer

Abruptly stops rogue master runs or runaway cloud connections using natural language, safely isolating operational networks when things go wrong.

What Changes When You Connect

01 Stop switching tabs. Instead of jumping between dashboards to check load results, your agent manages everything from workspaces (using `list_workspaces`) to live run monitoring.

02 Gain granular control over test environments. Use the MCP to list projects (`list_projects`) and retrieve detailed configurations for specific tests (`get_test`) before running them.

03 Respond instantly to failures. If a test runs wild, you can execute emergency shutdown controls using `stop_master` without manual intervention or complex API calls.

04 Get immediate visibility into performance health. Querying active master records via `list_masters` and checking gateway run status with `get_master` provides instant operational insights.

05 Eliminate guesswork during testing. You can trigger a new execution (`start_test`) and immediately query the resulting report data using `get_report`, getting p90/p99 KPIs right in your chat.

Real-World Applications

A critical API endpoint is slowing down during peak usage.

The SRE asks the agent to run a stress test on the payment gateway. The agent uses `start_test` and then queries the resulting data with `get_report`, immediately showing that p99 latency spikes above acceptable limits, identifying the bottleneck for the team.

Need to quickly validate a new microservice deployment.

The QA Automation Team uses the agent to list all available test structures (`list_tests`). They select the correct definition and trigger an execution. The system confirms the run is active, allowing them to proceed with confidence in the CI/CD cycle.

A performance test run gets out of control.

During a large-scale deployment simulation, the connection runs wild and threatens stability. Instead of scrambling for the dashboard, the DevOps Engineer tells the agent to use `stop_master`, safely isolating the network and preventing an outage.

Need to audit who has access to test environments.

A manager needs a full list of users with testing permissions. The agent uses `get_user` to pull active user records, allowing them to quickly verify the roles and account metadata across the platform.

Patterns to Avoid

Manually managing IDs for test runs

✗ AVOID

Trying to track every Workspace ID or Project ID by copy-pasting from multiple dashboards, which is slow and error-prone.

✓ INSTEAD

First, use `list_workspaces` to see all available environments. Then, use `list_projects` on a specific workspace ID to get the exact project name and ID needed for the next step.

Forgetting test definitions are linked

✗ AVOID

Running a load test without first verifying which JMeter definition is attached to the target environment, resulting in an incomplete or invalid run.

✓ INSTEAD

Always check available tests by calling `list_tests` and then use `get_test` on the specific ID. This confirms the full configuration before you hit 'run'.

Stopping the wrong connection

✗ AVOID

Issuing a general shutdown command that stops all active testing, including benign background monitoring runs.

✓ INSTEAD

Before stopping anything, always use `list_masters` to enumerate attached structured rules. Then, issue the precise stop command using `stop_master` on the specific master ID you intend to shut down.

The Right Fit

You should connect this MCP if your primary workflow involves running complex, multi-stage performance tests and requires real-time visibility into load metrics (p90/p99 KPIs). It's essential when you need to automate the entire cycle: listing resources, triggering a run (`start_test`), monitoring status (`get_master`), and safely shutting down (`stop_master`). Don't use this if your only goal is simple uptime checks or viewing static logs. For basic log retrieval

that doesn't involve performance metrics, you might find an alternative logging MCP better suited than one focused on heavy load generation.

BlazeMeter MCP for AI Agents: Streamlining Performance Test Management

Right now, running a full-stack performance test is a pain. You have to manually jump between the BlazeMeter web interface and your chat window just to check status updates or list available projects. Copying IDs, finding the right workspace, and verifying credentials across multiple tabs wastes time and introduces human error.

With this MCP, you simply tell your agent what you need—for example, 'Check all active performance tests for the checkout flow.' The agent handles the listing of workspaces, locating the correct project, and even retrieving detailed configuration data. You get a clean, actionable summary right where you are working.

BlazeMeter MCP for AI Agents: Monitoring Live Load Metrics

Monitoring performance live is tedious. You spend time refreshing dashboards to see if the p90 or p99 metrics are spiking, and you have no easy way to safely halt a runaway test that threatens your production environment.

This MCP gives you direct control over those critical moments. Your agent can monitor active master runs and force a shutdown instantly using `stop_master`. You get immediate confidence that the infrastructure is protected while you analyze the performance data.

BlazeMeter: 10 Tools for Load Test Management & Monitoring

Use these tools to list resources, start load tests, check master status, and retrieve detailed performance reports from BlazeMeter.

#	TOOL	DESCRIPTION
01	`list_workspaces`	Identifies and lists all bounded workspaces within the BlazeMeter platform.
02	`list_projects`	Extracts a list of projects that are bound to a specific workspace.
03	`list_tests`	Provides a comprehensive JSON payload listing all available tests.
04	`get_test`	Retrieves the full configuration for an active test limit within the vault.
05	`start_test`	Initiates a new, irreversible load generation validation run using specified metrics.
06	`list_masters`	Enumerates all attached structured rules that export active master records.
07	`get_master`	Runs an automated validation check to determine the status of a specific gateway run.
08	`stop_master`	Sends a native Gateway shutdown logic command to identify and stop active master arrays.
09	`get_report`	Inspects deep internal data arrays to mitigate specific plan math reports.
10	`get_user`	Identifies and retrieves the active user records associated with the platform.

See It in Action

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

U I need to run a stress test on my main user API. Can you start it?

Performance Test Status

- **Test ID:** 987654 (UserAuth)
- **Status:** Running (Active Master: m-11223)
- **Throughput:** Stable, currently at 120 RPS.

I've initiated the load test. I recommend monitoring the p99 KPI for spikes over the next hour.

U List all my active workspaces and projects so I know what tests are available.

Here is a list of your current testing resources:

Workspaces Found:

- `Development_API` (ID: 123456)
- `Pre-Production_Staging` (ID: 789012)

Under `Development_API`, the following projects are configured:

- `Core Backend Stress`
- `User Profile Load Test`

U Stop that master test right now. It's running too long.

⚠️ **EMERGENCY SHUTDOWN INITIATED** ⚠️

Connection to Master ID `m-11223` has been successfully severed and scaled down. All active load generation for this session is terminated immediately. You can now review the final metrics report in your chat history.

Frequently Asked Questions

01 **How do I use BlazeMeter MCP to test my API performance?**

You tell the agent which endpoint you want to stress-test. The MCP triggers a cloud load run, simulating real user traffic. You get immediate access to throughput reports and latency metrics like p90/p99 right in your chat window.

02 **Can I use BlazeMeter MCP if my test runs go wrong?**

Yes. If a test gets out of control, the agent can execute emergency stop controls using natural language. This safely shuts down runaway connections and protects your network architecture instantly.

03 **Does BlazeMeter MCP handle multiple environments?**

Absolutely. You can list all available workspaces and projects first. Then, you direct the agent to run a specific test on any environment—from development to pre-production—without switching context.

04 **What metrics does BlazeMeter MCP give me?**

You get precise operational health data. This includes throughput reports and critical KPIs like p90 (the 90th percentile) and p99 (the 99th percentile), which tell you how the system performs under heavy load.

05 **Is BlazeMeter MCP better than using a standalone dashboard?**

Yes. It keeps your entire testing process—from listing resources to executing tests—in one place with your AI client. You don't need to leave your conversation or IDE.

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 `"blazemeter": { "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

BlazeMeter 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 BlazeMeter. 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	BlazeMeter MCP
Server ID	`019d755e-4295-7117-963e-b24ed4eb1581`
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/blazemeter.