MCP SERVER | NO CODE | CLOUD HOSTED

Sauce Labs MCP

Monitor and manage your entire test pipeline status.

Sauce Labs MCP connects your entire UI/E2E test execution environment to your chat agent. Stop bouncing between CI dashboards and the Sauce Labs site just to check a build status or find an error log. This tool lets you monitor job results, diagnose failures, and manage concurrency limits—all right where you're chatting.

A+ Quality Score 98.33/100

automated-testing | ui-testing | e2e-testing | test-automation | pipeline-monitoring | quality-assurance

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/sauce-labs

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.

Sauce Labs MCP

11 tools available
Cloud-hosted on Vinkius

Connecting Sauce Labs via this MCP puts your entire test automation workflow directly into your chat agent. Instead of logging into the dashboard to check if a build passed or failed, you simply ask your AI client. You can quickly check the status of recent builds and drill down into specific job results, seeing pass/fail ratios instantly. Need to free up resources? You can programmatically stop hung jobs, freeing up test concurrency limits right away. Beyond that, you can review current account activity levels or browse supported OS and browser combinations for your next matrix variable. This capability is managed by Vinkius, making it simple to connect this critical infrastructure into any MCP-compatible client.

This means QA engineers get immediate access to failure logs without navigating complex menus, and DevOps teams know their concurrency limits before the test queue backs up.

Core Capabilities

01 — Check Build Status

You can retrieve details for specific builds or list recent automation runs to track overall stability.

02 — Diagnose Test Jobs

The agent pulls detailed information on individual test jobs, showing their status and results within a build.

03 — Manage Resources

You can check current concurrency limits or list active Sauce Connect tunnels to monitor resource usage.

04 — Stop Failing Tests

If a test job hangs, you can issue a command to stop it immediately, freeing up system resources.

05 — Verify Platform Support

The agent lists all supported OS and browser combinations (like Appium or WebDriver) for your tests.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to this MCP using your Sauce Labs credentials, providing your Username, Access Key, and operating Region.

02 Connect your preferred AI client (like Claude or Cursor) through the Vinkius catalog.

03 Ask your agent a question about test status, concurrency, or job results. It will run the necessary commands and provide the answer directly in the chat.

The bottom line is you get real-time insight into complex testing infrastructure without leaving your conversation window.

Built For

This MCP serves QA Engineers tired of context switching between dashboards and CI/CD tools. It's for DevOps staff who need to manually check concurrency limits at 2 AM, and developers needing quick verification of platform support when adding new test variables.

QA Engineer

Diagnosing a flaky Selenium or Playwright job by requesting the failure logs or video output directly in the chat.

DevOps Engineer

Checking active Sauce Connect tunnels and current concurrency limits to prevent test queues from backing up unexpectedly.

Software Developer

Verifying available browser/OS combinations on the fly when setting up a new matrix variable for a test suite.

What Changes When You Connect

01 Stop jumping between dashboards. Your agent pulls job details, build statuses, and error logs directly into the chat window, saving time and mental overhead.

02 Manage resources instantly. If a test hangs, use the `stop_job` capability to free up concurrency limits immediately, preventing your entire pipeline from backing up.

03 Stay ahead of capacity issues. Use `get_concurrency` or `list_tunnels` to check account usage before running high-volume tests, ensuring you never hit a hard limit unexpectedly.

04 Speed up diagnosis. Instead of manually checking logs for every failure, the agent gives you actionable build reports and error metadata right when you ask.

05 Broaden your testing scope. Need to test on an obscure OS/browser combo? Use `list_platforms` to verify support instantly without leaving your workflow.

Real-World Applications

The Build Failed, I Need Answers Now

A QA engineer sees a build fail and asks the agent for details. The agent checks the `list_builds` to find the latest run, then uses `get_build_jobs` to pinpoint which specific job failed, giving the exact failure context in seconds.

The Test Queue is Stuck

A DevOps engineer notices tests aren't running. They use the agent to check `list_tunnels` and find an abandoned connection. They then execute `stop_job` on the stale job ID, releasing capacity for the waiting queue.

Platform Support Check

A developer is adding a new test matrix variable but isn't sure if Chrome on Ubuntu is supported. They ask the agent to run `list_platforms`, instantly verifying compatibility before writing more code.

Checking Overall Health

Before starting a massive regression suite, an engineer asks about system capacity. The agent runs `get_activity` and `get_concurrency`, confirming they have enough parallel sessions available for the run.

Patterns to Avoid

Manual Dashboard Switching

✗ AVOID

The engineer has to copy a job ID from Jira, open the Sauce Labs dashboard in Chrome, navigate to Builds, find the specific test run, and then locate the error log within that page.

✓ INSTEAD

Just ask your agent. You tell it: 'Show me the failures for build ABC.' The agent handles finding the correct data using `get_build` or `list_jobs`, delivering the result instantly in the chat.

Ignoring Concurrency Limits

✗ AVOID

The team runs three large test suites concurrently, exceeding their allocated parallel sessions. All subsequent tests fail with a 'resource limit exceeded' error.

✓ INSTEAD

Always check capacity first by asking for the `get_concurrency` status. If limits are tight, you can use `stop_job` to terminate non-critical runs and free up space.

Guessing Platform Compatibility

✗ AVOID

A developer assumes a certain browser/OS is supported because it worked last quarter, only for the test run to fail later due to platform mismatch.

✓ INSTEAD

Use `list_platforms` before writing code. It provides a definitive list of all currently supported OS and browser combinations for your tests.

The Right Fit

Use this MCP if your primary need is real-time monitoring, status checks, or resource management related to automated UI/E2E testing. If you only need to write the test code itself, or manage user accounts outside of the build pipeline, this tool isn't enough. This MCP excels at telling you *what happened* (checking `get_build` results) and *why it couldn't run* (via `get_concurrency` reports). Don't use it if you need to write complex test scripts; that requires a dedicated code editor. But if you are in the diagnostic phase—the 'did it pass?' or 'why did it fail?' stage—this is the tool you need.

The Dashboard Jungle

Today, finding out why a test failed requires a lot of clicking. You jump from your CI dashboard to the Sauce Labs site. Then you navigate through builds, find job IDs, and finally click deep into logs just to see an error message. It's tedious, it takes five minutes minimum, and you lose context while switching tabs.

With this MCP, the entire process collapses into a single conversation turn. You ask your agent about build failures or concurrency usage, and it runs the necessary checks (like `list_builds` or `get_concurrency`) and delivers an immediate, actionable answer right in your chat.

Get Live Test Status with Sauce Labs MCP

You no longer need to manually check the status of active tunnels or verify if a job is stuck. The agent can track resources by listing `list_tunnels` and, crucially, you get immediate control by calling `stop_job` on rogue processes.

The difference now is speed and efficiency. You stop diagnosing infrastructure issues in separate tabs and start solving problems conversationally.

Sauce Labs MCP: 11 Tools for Testing Operations

These tools let you manage everything from checking account activity levels to listing all supported operating systems and browsers in the Sauce Labs environment.

#	TOOL	DESCRIPTION
01	`get_activity`	Retrieves metrics about your current Sauce Labs account activity level.
02	`get_build_jobs`	Lists every individual test job that ran within a specific build.
03	`get_build`	Pulls all the detailed results and metadata for one specific test build run.
04	`get_concurrency`	Retrieves your current account limits on parallel testing sessions.
05	`get_job`	Gathers all the detailed information for a single, specific test job ID.
06	`get_status`	Checks if the main Sauce Labs platform is currently available and operational.
07	`list_builds`	Provides a list of all your most recent automation test builds.
08	`list_jobs`	Lists the IDs and status of recent individual test jobs running on Sauce Labs.
09	`list_platforms`	Retrieves a comprehensive list of all supported operating systems and browsers for testing.
10	`list_tunnels`	Lists any active Sauce Connect tunnels currently running in your account.
11	`stop_job`	Sends a command to immediately terminate any test job that is stuck or failing.

See It in Action

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

U What is our current concurrency usage vs limit in Sauce Labs?

I checked your concurrency data. You are currently using 12 out of your 25 allowed parallel sessions. There are currently no queued jobs. You have plenty of capacity to spin up a new test run.

U Show me the jobs that failed in my last automation build.

In your most recent build ('Nightly Regression 2.4.1'), exactly 1 out of 45 jobs failed. Job ID: fbd198. The failure occurred on Chrome version 114 (Windows 10). Would you like the deep link to the video asset or the error log for this job?

U Stop the test job ID 4f4f391e0 because it's stuck.

I've successfully terminated Job ID 4f4f391e0. The status has changed to 'aborted' and the underlying machine node has been freed, releasing 1 concurrent session to the pool.

Frequently Asked Questions

01 **How do I check if my Sauce Labs account has enough capacity using the Sauce Labs MCP?**

You use the `get_concurrency` tool. It reports your current usage versus your total allowed parallel sessions, letting you know instantly if a test run will fail due to resource limits.

02 **I need help finding my latest failed job using Sauce Labs MCP.**

Use `list_builds` first to find the correct build ID. Then, request that specific build's details with `get_build`, and finally narrow it down by listing jobs using `get_build_jobs`.

03 **What is the purpose of `stop_job` in the Sauce Labs MCP?**

`stop_job` sends a command to immediately terminate any running test job that has become stuck or unresponsive, freeing up resources for other tests.

04 **Can I check which browsers are supported using Sauce Labs MCP?**

Yes, run the `list_platforms` tool. It gives you a comprehensive list of every OS and browser combination currently available in your account.

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 `"sauce-labs": { "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

Sauce Labs 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 Sauce Labs. 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	Sauce Labs MCP
Server ID	`019d7603-926f-7212-8a8c-fb7f93d8967d`
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/sauce-labs.