MCP SERVER | NO CODE | CLOUD HOSTED

Testim MCP

Manage branches and check E2E results from chat.

Testim connects your end-to-end testing suite directly into your AI agent. You can manage code branches, trigger complex test runs, and analyze execution logs—all using natural conversation. This MCP lets you run full regression tests or check specific development branch health without ever opening the Testim GUI.

A+ Quality Score 100/100

automated-testing | end-to-end-testing | test-automation | qa-automation | execution-logs | test-suite

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

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.

Testim MCP

10 tools available
Cloud-hosted on Vinkius

Imagine running complex quality assurance checks just by talking to your agent. This MCP brings automated E2E testing orchestration right into your chat window, letting you manage your entire test suite via conversation. Instead of navigating dashboards and clicking through multiple tabs to check logs or initiate a build, you ask your AI client to handle it all.

It lets you quickly view parallel development efforts by listing project branches, run dynamic batches based on labels, and pull the pass/fail status for any execution ID immediately after deployment. If you're working with different types of systems, Vinkius hosts this MCP in its catalog, making sure your agent can access all the necessary test details to give you a clear picture of quality. You get granular control over everything from triggering specific tests to merging branches without leaving your editor.

Core Capabilities

01 — Manage Project Branches

List, create, and merge automated test development branches directly through your agent.

02 — Run Test Batches

Trigger full test runs by specifying defined test plans, entire suites, or filtering by specific labels.

03 — Check Execution Diagnostics

Retrieve the status and detailed errors for any completed test run using a unique execution ID.

04 — Inspect Test Inventory

View all available tests in your project or retrieve full details on a specific automated test script.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to this MCP and provide your unique Testim Project ID and API Key.

02 Your agent connects the credentials, allowing it to access all testing tools. You then give a natural language command like 'Run tests on the latest branch'.

03 The MCP executes the required actions (like listing branches or triggering a run) and returns structured data—the pass/fail status, logs, or merge options—directly to your chat.

The bottom line is that you use natural language prompts to control complex CI/CD testing pipelines without touching any GUI.

Built For

This MCP is for QA Engineers and Release Managers who are tired of context switching between their IDE, the test dashboard, and Slack. It's for developers who need to verify E2E stability without leaving their coding environment.

QA Engineer

Triggers full regression test suites via chat as soon as a new testing environment is ready for review.

Frontend Developer

Runs specific E2E test scripts directly inside their IDE when they write code, getting immediate feedback on stability.

Release Manager

Rapidly audits the health of multiple test plans and retrieves execution IDs to confirm green lights before authorizing a production deployment.

What Changes When You Connect

01 Stop copying logs. By calling get_execution_results, you instantly pull the pass/fail status for any test run ID directly into your conversation thread.

02 Don't switch tools to see what's happening in development. Use list_project_branches or create_project_branch to manage and compare parallel code efforts without opening the Testim GUI.

03 Run exactly what you need, nothing more. Instead of running everything, use run_tests_by_label to trigger tests that match specific functional requirements (e.g., 'checkout' or 'payment').

04 Get deep insight into your test scripts using get_test_details. You can review all the technical steps and setup for any single automated script right away.

05 Verify deployment safety with run_test_plan. Triggering a full plan ensures that a critical set of tests runs together, providing confidence before release.

Real-World Applications

Pre-Deployment Safety Check

A Release Manager needs to know if the staging environment is stable. They ask their agent to run_test_plan 'Nightly Regression'. When the results come back, they use get_execution_results with the returned ID to confirm that every critical test passed before hitting the deploy button.

Fixing a Bug on a Feature Branch

A Frontend Developer has written code for a new feature branch. Instead of committing and opening a full pipeline, they use run_specific_test to validate only the critical paths. This gives immediate feedback without interrupting the main development line.

Comparing Development Lines

A QA Engineer needs to see if two separate feature branches conflict. They first list_project_branches to confirm both exist, then use merge_project_branch to simulate and validate how one branch incorporates changes from the other.

Investigating a Failure

A developer sees an error log in chat. They ask their agent to get_test_details for that failing test ID. This quickly pulls up the full script details, allowing them to diagnose if the failure is due to code or setup.

Patterns to Avoid

Trying to check everything manually

✗ AVOID

Copying dozens of individual test IDs and pasting them into a dashboard filter, then checking each one's status in separate tabs.

✓ INSTEAD

Instead, use list_tests to get an inventory, or trigger runs using run_test_suite for the entire group. Then, retrieve all results at once with get_execution_results.

Forgetting branch context

✗ AVOID

Running a test intended for a feature branch against the main development line and getting misleading results.

✓ INSTEAD

Always call list_project_branches first. Then, specify the target environment using create_project_branch or run_specific_test to ensure you're testing on the right isolation point.

Assuming full coverage

✗ AVOID

Running a general test suite and assuming that means all edge cases are covered, leading to unexpected production failures.

✓ INSTEAD

Don't just run everything. Use run_tests_by_label to focus the tests only on the specific functional area you changed, like 'auth-flow' or 'payment'.

The Right Fit

Use this MCP if your core pain point is coordinating and tracking end-to-end test status across multiple development branches. You need a centralized way to run tests (run_test_plan, run_test_suite) and get immediate diagnostic feedback on results or failure points. Don't use it if you just need to view raw application logs; that requires different log analysis tools. If your goal is merely project setup or user management, look for a dedicated identity MCP instead.

The Manual Test Status Check

Today, checking the health of your application means opening the Testim GUI. You navigate to the run history, filter by date, find the correct branch, and then manually check if the test plan ran successfully. If you want details on a failure, you have to copy an ID and paste it into another troubleshooting dashboard, leading to endless tabs and fragmented information.

With this MCP, that whole process vanishes. Your agent handles the complexity. You simply ask your AI client to 'check the status of the nightly regression tests.' The system executes run_test_plan, pulls the result using get_execution_results, and delivers a clean summary right where you are working.

Testim MCP: Branch Management

Before making any changes, developers used to have to manually create new branches in the GUI or risk overwriting work by merging directly. If a test failed due to an environment issue rather than code, tracking which branch was responsible was a multi-step headache.

Now you can use list_project_branches to see every active development effort at a glance. Need to start testing a fix? Use create_project_branch and run your tests against that clean slate. It makes complex parallel development predictable.

Testim: 10 Tools for Automated Testing

Use these tools to interact with all Testim functions—from listing test inventories to managing development branches and retrieving execution diagnostics.

#	TOOL	DESCRIPTION
01	`create_project_branch`	Creates a brand new, isolated development branch for testing changes.
02	`get_execution_results`	Pulls the final status and detailed outcome report from a specific test run ID.
03	`get_test_details`	Retrieves comprehensive information about one particular automated Testim test script.
04	`list_project_branches`	Shows a list of all active, parallel development branches within the project.
05	`list_tests`	Provides an inventory listing of every automated test defined in your Testim project.
06	`merge_project_branch`	Merges changes from one development branch into another target branch (like 'master').
07	`run_tests_by_label`	Triggers a complete test run that only includes tests matching specific, defined labels.
08	`run_test_plan`	Starts an execution for an entire predefined set of linked test plans.
09	`run_test_suite`	Initiates a run that includes every single test within a specified, grouped test suite.
10	`run_specific_test`	Starts an immediate test execution for one specific test script on a chosen branch.

See It in Action

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

U List all branches available in my Testim project.

Here are 3 branches active in your project:
- `master` (Default, ID: MSTR)
- `feature/new-checkout-flow` (ID: FTR104)
- `bugfix/login-captcha` (ID: BUG44)

Would you like to trigger test suites on any specific branch or merge one?

U Trigger a plan run for "Nightly-Regression".

Test plan 'Nightly-Regression' successfully triggered.
Execution ID: `EXC-558291a`
The cloud runners are executing your plan. You can ask me to fetch the execution results and logs by providing this ID later.

U Retrieve the execution diagnostic summary for ID EXC-999333.

Execution `EXC-999333` has concluded.
Status: **FAILED** ❌
Error Details: At Step 4 (Verify Cart Element). Element `#checkout-btn` could not be located on the DOM within the 10000ms timeout threshold window. Screenshot payload available in dashboard.

Frequently Asked Questions

01 **How do I check the status of a test run using Testim MCP?**

You use get_execution_results by providing the unique execution ID. This tool pulls the final status, whether it passed or failed, and gives you the error details immediately.

02 **Can I trigger tests for only a specific area using Testim MCP?**

Yes, use run_tests_by_label. You just tell your agent which labels to target (e.g., 'payment' or 'user-login'), and the system runs only those relevant tests.

03 **What is the difference between running a suite and running a plan with Testim MCP?**

A test suite runs all scripts within one defined group of tests. A test plan coordinates multiple suites or sets of tests, giving you broader coverage across different feature areas.

04 **Do I need to use the GUI if I want to manage my branches with Testim MCP?**

No. You can list_project_branches and even merge changes using the MCP tools directly through your agent, removing the need to open the graphical user interface entirely.

05 **Where do I find details about a specific test script with Testim MCP?**

Use get_test_details. This tool provides all technical information on a single test script, helping you understand its prerequisites and expected behavior.

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 `"testim": { "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

Testim 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 Testim. 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	Testim MCP
Server ID	`019d7611-c6ea-7337-b3bd-ac2f24757932`
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/testim.