MCP SERVER | NO CODE | CLOUD HOSTED

TestRail MCP

Query full project health via natural conversation.

TestRail MCP connects your conversational AI directly to your entire test suite. Instead of clicking through dashboards, you query project status, review manual steps for specific cases, and analyze run metrics using plain language commands. It brings deep QA insights straight into your agent.

A+ Quality Score 100/100

qa-testing | test-suites | test-cases | release-management | test-automation | quality-assurance

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

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.

TestRail MCP

10 tools available
Cloud-hosted on Vinkius

This connector lets you get full visibility into complex quality assurance projects without ever opening the TestRail web interface. You can ask your AI client to pull down a list of all active test projects or find specific sections within massive repositories, giving you an immediate architectural overview.

It tracks progress across multiple dimensions: you can check upcoming release milestones or see exactly which tests passed and failed in any given run. Need details on how a manual case works? You just ask for the steps, preconditions, and validation targets. This capability lets developers pull reproduction instructions directly into their IDE to start fixing bugs immediately.

All these deep QA metrics are accessible through natural conversation. Vinkius hosts this MCP, giving your agent access to TestRail's full history so you can manage project status and analyze test runs—all without switching tabs.

Core Capabilities

01 — List all available projects

Retrieves a list of every test project on the instance, providing essential IDs for further queries.

02 — Get project and section structure

Lists all sections (folders) or retrieves detailed information about specific TestRail projects.

03 — Check milestones and deadlines

Interrogates upcoming QA release milestones for a given project, keeping you updated on timelines.

04 — Review test case steps

Retrieves the full details, preconditions, and step-by-step logic for any specific manual test case ID.

05 — Analyze active test runs

Generates an immediate summary of a specific test run, showing total tests executed, passed counts, failures, and blocked items.

06 — List all contained tests in a run

Pulls down the list of individual tests that were included in a particular test run.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to this MCP and provide your TestRail Base URL, along with an API Key.

02 Your AI client executes natural language commands (e.g., 'What's the status of Project X?').

03 The connector gathers test run data, project structures, or case details and returns a clean summary directly to your chat.

The bottom line is you get real-time QA metrics without opening the TestRail web application.

Built For

Test Leads, Software Developers, and Automation Engineers. This connector helps anyone frustrated by jumping between multiple dashboards just to answer a simple status question.

Software Developer

Needs to pull down the exact manual reproduction steps from a failing test case directly inside their IDE when debugging.

QA Test Lead

Must quickly summarize project health, checking upcoming milestones and overall run metrics without deep-diving into complex dashboard tables.

Automation Engineer

Uses this to analyze the properties of existing test cases so they can efficiently convert manual steps into code for e2e automation.

What Changes When You Connect

01 You instantly get summaries of active Test Runs. Instead of digging through dashboard tables, you ask for the status and know exactly how many tests passed or failed.

02 Need to understand a bug? You can retrieve the step-by-step logic and preconditions for any test case ID directly into your chat window.

03 Never manually navigate project folders again. You can list all sections or even query the full suite architecture, getting a clear map of your repository structure.

04 Keep track of deadlines without logging in. Listing project milestones gives you an immediate overview of upcoming QA release dates and targets.

05 For automation engineers, this lets you analyze test case properties to pull down structured data that can be used to write code-based e2e tests quickly.

Real-World Applications

Diagnosing a failing build

A developer sees a bug report and asks their agent, 'What are the steps for Test Case 1285?' The connector runs get_test_case_details and provides all manual preconditions and exact steps needed to replicate the failure without leaving their terminal.

Getting project scope overview

A QA Lead asks, 'What projects are active right now?' The agent uses list_test_projects, providing a clean list of all IDs and names. This helps them decide if they need to check milestones or test suites for that specific ID.

Checking release readiness

A Test Lead needs to know the status before a major launch. They ask, 'What are the upcoming project milestones?' The agent uses list_project_milestones and provides a timeline summary, letting them prioritize which runs need immediate attention.

Analyzing run failure patterns

An engineer wants to know if a recent build was stable. They ask for the status of Run ID 403. The agent uses get_test_run_details and outputs a clear breakdown: Total Executed, Passed, Failed, and Blocked.

Patterns to Avoid

Treating it like a simple database query

✗ AVOID

Asking the agent only for 'test cases' without specifying a project or suite. The response is generic and unusable.

✓ INSTEAD

Always narrow your scope. First, use list_test_projects to get the ID, then use list_test_suites with that ID to pinpoint exactly what you need.

Ignoring project structure

✗ AVOID

Trying to ask about a specific test case without knowing which section it belongs to. The agent fails because context is missing.

✓ INSTEAD

Run list_project_sections first. This gives you the folder hierarchy, allowing you to specify the correct location for any detailed query.

Over-relying on full details

✗ AVOID

Requesting all data from every test run in a project—a massive dump of raw JSON that's impossible to read.

✓ INSTEAD

Instead, use get_test_run_details for an immediate summary. If you need more detail, list_run_tests provides the focused list of individual tests.

The Right Fit

Use this MCP if your primary pain point is context switching and manually navigating complex QA dashboards. You should use it when you need to *read* structured data—like reproduction steps from get_test_case_details, or metrics from get_test_run_details—via natural conversation.

Don't use this if your goal is to modify the test structure itself (e.g., adding a new project or editing preconditions). This connector is read-only for reporting and analysis. If you just need a simple list of all available projects, start with list_test_projects; it's the essential starting point for almost every query.

The pain of checking test coverage status

Today, figuring out project health means logging into TestRail, clicking on the Project Overview dashboard. You might then need to jump to the Milestone tab for dates, and finally open a specific Run ID just to see if tests failed or passed. It's three separate screens and at least five clicks every time you check status.

With this MCP, the process is simple: you ask your agent, 'What's the latest on Project X?' The connector compiles data from project milestones, test run details, and section lists into one single response in your chat. You get the answer instantly, without leaving your workflow.

Get TestRail Case Details with this MCP

Manually finding reproduction steps requires knowing the exact test case ID and navigating deep into a specific project section. You have to open that case, scroll past the description, and copy-paste the preconditions and step sequence.

Now, you simply ask your agent for 'the manual steps for Test Case 1285.' The connector uses get_test_case_details and delivers the entire structured playbook—preconditions, steps, expected results—in a clean format ready to use.

TestRail with 10 Tools

Use these tools to query every aspect of your QA data, from listing projects to getting deep details on individual test cases.

#	TOOL	DESCRIPTION
01	`get_test_case_details`	Pulls all data, including preconditions and detailed steps, for a single test case ID.
02	`get_test_project_details`	Retrieves comprehensive information about a specific TestRail project.
03	`get_test_run_details`	Fetches the overall status and metrics for a specified test run ID.
04	`list_test_cases`	Generates a list of all available test cases within a project, optionally filtered by suite name.
05	`list_project_milestones`	Lists every major milestone associated with a given TestRail project.
06	`list_test_projects`	Outputs all test projects available on the instance, which is critical for targeting other resources.
07	`list_test_runs`	Lists all completed or active test runs within a selected project.
08	`list_project_sections`	Outputs the folder hierarchy, listing every section available inside a project.
09	`list_test_suites`	Lists all major test suites within a specific project boundary.
10	`list_run_tests`	Outputs the list of individual tests that were executed during a particular test run.

See It in Action

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

U What active TestRail projects are available in this instance?

Found 2 active TestRail projects:
- `ID: 15` | "Native Mobile Apps (Q2)" | Uses standard multi-suite mode.
- `ID: 29` | "Core Payment Backend" | Single suite repository.

Should I list milestones or test suites under Project ID 15?

U Get the manual preconditions and test steps for Test Case 1285.

Case ID `1285` : **OAuth Registration Gateway Failure**
> **Priority:** CRITICAL_HIGH | **Estimate:** 5m
> **Preconditions:** Server must be isolated via VPN routing rules.

Steps:
1. Call initialization endpoint bypassing token assignment. (Expected Result: API triggers `ERR401` gracefully without cascading stack traces).
2. Force brute-force retries mimicking scripts. (Expected Result: IP locks completely after exactly 5th try).

Can I translate these rules over towards a framework structure?

U Return exact status summary for Test Run ID 403.

Metrics recovered for Test Run `#403: Sanity Validation Build.v45` :
- **Total Executed Tests:** 80
- **Passed:** 71 ✅
- **Failed (Action Required):** 6 ❌
- **Blocked (Dependency issues):** 3 ⏸️

This run has notable defect traces. Would you like me to extract the exact test lists behind the 6 failed triggers?

Frequently Asked Questions

01 **How do I find all active projects using TestRail MCP?**

You list all available test projects by calling the list_test_projects tool. This gives you a full inventory of IDs, allowing you to target any other resource on the instance.

02 **Can I check upcoming deadlines with TestRail MCP?**

Yes, use the list_project_milestones tool. It pulls all scheduled milestones for a project ID, helping you keep track of QA release timelines without opening the web app.

03 **What is the difference between listing tests and test runs?**

list_test_runs lists all available execution records (the 'container'). list_run_tests uses one of those run IDs to pull the specific individual tests that were executed in that session.

04 **Can TestRail MCP help with bug replication?**

Absolutely. Use get_test_case_details on a case ID. This retrieves the complete manual playbook, including preconditions and step-by-step instructions needed for developers to reproduce any issue.

05 **Does TestRail MCP support folder navigation?**

Yes. You can use list_project_sections to pull down the full section list (folder hierarchy) from within a project, giving you a clear visual map of your test repository.

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 `"testrail": { "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

TestRail 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 TestRail. 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	TestRail MCP
Server ID	`019d7612-2985-733c-8326-56b1ba657c18`
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/testrail.