MCP SERVER | NO CODE | CLOUD HOSTED

Mabl MCP

Control E2E Test Runs & Analysis

Mabl MCP gives your AI client full control over complex E2E testing and quality orchestration. You can list all monitored applications and environments, trigger test plans across staging or production, monitor live runs, and deep-dive into failure details to pinpoint bugs instantly. It turns manual QA checklists into conversational commands.

A+ Quality Score 100/100

test-automation | e2e-testing | quality-assurance | low-code | failure-analysis | ci-cd-integration

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/mabl-ai-powered-test-automation

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.

Mabl (AI-Powered Test Automation) MCP

10 tools available
Cloud-hosted on Vinkius

This connector lets you treat your entire test suite like a conversation with your agent. Instead of navigating through complex UIs, you simply tell it what needs testing. You can list all available applications and then trigger specific test plans—say, the 'Checkout Regression' plan on the staging environment. Your agent monitors the run in real time, providing pass/fail rates and duration summaries as it progresses. If something breaks, you don't just get a red status; you get detailed failure analysis. The system extracts precise error reasons, diagnostic insights, and even visual screenshots for rapid debugging. This capability is part of Vinkius's massive catalog, giving your AI client deep control over your quality checks directly from your workspace.

Core Capabilities

01 — Identify all applications

Retrieves a list of every web and API application monitored by Mabl.

02 — Manage test environments

Lists configured testing environments along with their associated variables for context-aware testing.

03 — Execute test plans

Triggers a full, automated run of any defined test plan across specific deployment stages.

04 — Monitor live runs

Tracks the real-time status and retrieves detailed outcome summaries for ongoing test executions.

05 — Analyze failure reports

Deep-dives into failed test runs to extract exact failure reasons, diagnostic text, and screenshots.

06 — Discover plan labels

Lists the organizational tags used to group and select specific testing suites.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/mabl-ai-powered-test-automation** — connect your AI agent in three steps.

01 Subscribe to this MCP and enter your Mabl API Key into your agent.

02 Directly ask your AI client to list applications or environments it can manage.

03 Tell the agent which test plan to run, what environment to use, and wait for the failure analysis report.

The bottom line is you get full, conversational control over complex E2E testing workflows without opening a browser tab.

Built For

QA Automation Engineers who are sick of copy-pasting test plan IDs and sifting through endless dashboards. Software Developers needing instant UI regression reports to verify builds before merging code. DevOps teams responsible for auditing CI/CD pipeline quality gates.

QA Automation Engineer

Uses the MCP to trigger complex test plans and analyze failure diagnostics conversationally, without leaving their IDE.

Software Developer

Verifies build quality by retrieving session screenshots and detailed outcomes for UI regressions directly from their workspace.

DevOps Engineer

Audits CI/CD test results across multiple Mabl projects, monitoring environment health and associated variables efficiently.

What Changes When You Connect

01 Stop manually cross-referencing build logs and test dashboards. By using `mb.get_execution`, you get a single, comprehensive report that includes pass/fail rates, duration, and detailed failure analysis.

02 Never guess which environment to target again. Run `mb.list_envs` first, then tell your agent exactly where the plan needs to execute—staging or production—ensuring the right variables are loaded.

03 Reduce debugging time from hours to minutes. The MCP can extract precise root causes and visual screenshots for failed tests using the results of a run, bypassing vague error messages.

04 Manage complexity by organizing your suite. Use `mb.list_labels` to see all available tags and then trigger specific groups of tests using the `mb.trigger_plan` tool.

05 Keep track of what's running without switching tabs. You can use `mb.list_executions` for a quick status check, or `mb.get_execution` for a deep dive into any single test run history.

Real-World Applications

Finding the source of an intermittent bug

A developer notices a UI regression but can't reproduce it locally. They ask their agent to find all applications using `mb.list_apps`, select the target app, and then trigger a run on the staging environment using `mb.trigger_plan`. The resulting failure analysis provides screenshots showing the exact visual overlay causing the button click failure.

Auditing a release candidate build

A DevOps engineer needs to confirm that all critical components passed testing before deployment. They ask their agent to list all test plans using `mb.list_plans`, select the 'Smoke Test' plan, and then use `mb.get_execution` on the last run ID to verify the overall pass rate.

Validating API changes in a new environment

A backend team updates an API endpoint and needs QA to validate it quickly. They first list all environments using `mb.list_envs` to confirm the correct staging URL, then use `mb.trigger_plan` on the dedicated 'API Health Check' plan.

Reviewing test coverage gaps

A QA engineer wants to know what tests are grouped together. They list all available labels using `mb.list_labels` and then use `mb.get_app` for the main checkout app to confirm which specific API endpoints were monitored.

Patterns to Avoid

Manually triggering tests in a GUI

✗ AVOID

Opening the Mabl dashboard, navigating to 'Test Plans', selecting the target environment from the dropdown, and hitting 'Run'. This process is slow and requires multiple clicks.

✓ INSTEAD

Instead of clicking through menus, simply tell your agent: 'Trigger the Checkout Regression plan on Staging.' The MCP handles the entire execution sequence using `mb.trigger_plan`.

Relying on vague error messages

✗ AVOID

The test fails and the log only says, 'Element not found.' This forces a manual investigation into console logs or network traffic.

✓ INSTEAD

Use `mb.get_execution` to pull up the detailed failure analysis. The MCP extracts diagnostic insights, often pinpointing *why* the element was missing (e.g., due to an unexpected visual overlay).

Forgetting which environment is active

✗ AVOID

A developer mistakenly runs a test intended for staging against production data because they didn't check the current context.

✓ INSTEAD

Always start by listing environments using `mb.list_envs` to verify the available contexts, then specify the exact target when you tell your agent to run the plan.

The Right Fit

Use this MCP if your primary pain point is orchestrating and analyzing complex, end-to-end user journey tests that touch multiple systems. If you need to trigger a full test suite (like 'Checkout Regression') across defined environments (staging/production) and then get a deep, actionable report on *why* it failed—this is for you. Don't use this if you only write simple unit tests or if your testing involves manual data entry that can't be scripted. For simple environment checks, you might just need `mb.list_envs`. But if the goal is to verify application quality and failure diagnostics conversationally, this MCP is what you need.

Debugging test failures used to mean clicking through five different tabs.

Today, when a regression occurs, your workflow breaks. You open the Mabl dashboard, find the specific run ID, click into it, and scroll through endless logs trying to determine if the failure was because of bad code or just a temporary UI glitch. Then you copy parts of that log into Slack, asking a teammate to interpret it.

With this MCP, you simply ask your agent: 'What went wrong with the last checkout run?' The system runs `mb.get_execution` and returns a full analysis package—a precise error reason, diagnostic text, and a screenshot showing exactly what failed. You get answers, not log snippets.

Get instant quality verification with Mabl MCP.

The manual steps that disappear are the need to switch between monitoring tools, manually listing environments, and trying to piece together a failure narrative from disparate sources. You don't have to remember which test plan is linked to which label or environment variable.

Now, you control the entire quality pipeline through natural language commands. It’s not just an API connection; it’s instant operational visibility.

Mabl (AI-Powered Test Automation) MCP: 10 Tools

These tools let you programmatically interact with every part of your Mabl account, from listing applications to deep failure analysis.

#	TOOL	DESCRIPTION
01	`mb.list_apps`	Lists all applications monitored by Mabl, including details for web, mobile, and API testing.
02	`mb.get_app`	Retrieves full information on a specific Mabl application, listing its targets (web URLs plus API base URLs) and labels.
03	`mb.list_envs`	Lists all testing environments available in Mabl, returning their names, IDs, and defined variables.
04	`mb.list_plans`	Returns a list of all test plans (run policies), showing their status, triggers, and linked applications or environments.
05	`mb.trigger_plan`	Starts an immediate run of a specific Mabl plan via deployment event trigger.
06	`mb.list_executions`	Lists the most recent test execution results, giving a quick overview of outcomes.
07	`mb.get_execution`	Retrieves comprehensive details for one specific test run, including status, duration, and failure analysis reports.
08	`mb.list_tests`	Lists all available API tests configured within Mabl.
09	`mb.list_labels`	Returns a list of organizational labels (tags) used to group and manage test plans.
10	`mb.workspace_info`	Gets general metadata about the Mabl workspace, including organization details.

See It in Action

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

U List all my Mabl test plans

I've found 4 test plans: 'Daily-Smoke-Test' (ID: plan-123), 'Checkout-Regression' (ID: plan-456), 'Login-Flow-Validation' (ID: plan-789), and 'API-Health-Check' (ID: plan-012). Which one would you like to trigger or inspect?

U Trigger the 'Checkout-Regression' plan in the Staging environment

Triggering test plan 'Checkout-Regression'... I've successfully started the run in your Staging environment. I'll monitor the progress and notify you once the execution is complete. Would you like the initial status report?

U Show me the details for the last failed execution

Retrieving the most recent failed execution... Run ID 'exec-98765' failed 12 minutes ago during the 'Add to Cart' step. The AI analysis suggests a 'Button not clickable' error due to a visual overlay. I can provide the direct screenshot link if you'd like.

Frequently Asked Questions

01 **How do I list all my Mabl applications using the Mabl MCP?**

You use `mb.list_apps`. This tool gives you a complete catalog of every web, mobile, and API application that is currently monitored by your account.

02 **Can I run a test plan on different environments using mb.trigger_plan?**

Yes, `mb.trigger_plan` allows you to execute specific plans across defined deployment stages like staging or production, ensuring the correct context for your tests.

03 **What information does mb.get_execution provide about a failed test run?**

It gives full details of the execution, including the status, duration, pass/fail rates, and critical failure analysis reports with diagnostic insights and screenshots.

04 **How do I check what environments are available for testing?**

You run `mb.list_envs`. This function returns all defined environment names, unique IDs, and any associated variables needed by the test plan.

05 **Does the Mabl MCP help me find out which tests belong to a specific group?**

Yes, you can use `mb.list_labels` to view all organizational labels (tags). You can then reference these labels when triggering test plans to select grouped suites.

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 `"mabl-ai-powered-test-automation": { "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

Mabl (AI-Powered Test Automation) 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 Mabl (AI-Powered Test Automation). 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	Mabl (AI-Powered Test Automation) MCP
Server ID	`019d75cb-75b2-7053-9079-00cc5cf7768e`
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/mabl-ai-powered-test-automation.