

MCP SERVER

NO CODE

CLOUD HOSTED

Applitoools MCP for AI Agents

Visual Regression Testing and UI Quality Assurance

Applitoools brings AI-powered visual testing directly to your workflow. It lets your agent manage entire visual regression pipelines—checking test batches, reviewing UI differences, and handling baselines—all from your preferred client without opening a dashboard.

A+ Quality Score 98.33/100

visual-testing

regression-testing

ui-validation

automated-testing

quality-assurance

batch-processing



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.

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.

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.

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

Encryption at rest

Ed25519

PKI vault signatures

24h TTL

Ephemeral session keys

V8 Isolate

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.

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.

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.

06 — Honeypot Trap System

Phantom credentials are injected into isolated environments. If a honeypot 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.

FinOps

Cost tracking per tool, payload compression savings, budget optimization signals, and consumption trends.

Firewall & DLP

PII redaction activity, sensitive data protection counters, and security event timeline.

Agent Activity

Which AI clients are connecting, how often, and what they're doing — real-time session tracking.

Tool Health

Slowest and most error-prone tools, with actionable root-cause insights and performance baselines.

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.

Applitools MCP

10 tools available

Cloud-hosted on Vinkius

Need to validate frontend changes visually without logging into the Applitools console? This MCP connects your AI agent directly to your entire visual testing suite. You can ask it to analyze active test batches for aggregated status updates or drill down into specific sessions to spot failed images and browser discrepancies. It's built to handle complex tasks like listing 'golden' graphical baselines, checking branch-specific states, or even deleting outdated records to keep your workspace clean. Because Vinkius hosts this MCP, you connect once from any compatible client, giving your agent access to all these visual validation tools in one place. Your AI agent handles the heavy lifting, letting you validate changes and assess testing health dynamically right where you're working.

Core Capabilities

01 — Get status summaries for test batches

Retrieves quick counts of passed, failed, or unresolved sessions across an entire Applitools batch.

03 — Manage visual baselines by branch

Lists or deletes graphical baselines tied specifically to a Git branch, ensuring you only check the correct version history.

05 — Inspect full test batch details

Gathers all information about a given test batch, including its overall status and associated sessions.

02 — Analyze detailed session failures

Drills down into specific test sessions to identify pixel drift and browser-specific differences between runs.

04 — List all configured baselines

Retrieves IDs and names of all existing visual baselines for an application.

06 — Validate API connectivity

Runs a quick check to confirm the Applitools API key is working before triggering any tests.

One Click on Vinkius — From Prompt to Execution

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

- 01 Subscribe to this MCP and provide your Applitools API Key.
- 02 Your AI agent uses the connection to query test batches for summary statistics or detailed session failures.
- 03 You get actionable reports showing exactly where UI changes occurred, letting you validate frontend code without context switching.

The bottom line is that your agent treats Applitools like a native tool, allowing it to read and write data directly from the testing platform via natural conversation.

Built For

This MCP is for QA Automation Engineers who are sick of jumping between dashboards. It's also critical for Frontend Developers who need immediate visual confirmation during PR reviews, and Engineering Managers who just need high-level failure stats to sign off on a release.

QA Automation Engineer

Uses the agent to summarize unresolved test batches and pinpoint exact OS/browser combinations that failed testing.

Frontend Developer

Verifies branch-specific visual baselines during a Pull Request without having to switch focus from their code editor.

Engineering Manager

Pulls high-level batch statistics quickly, ensuring that all release criteria are visually met before sign-off.

What Changes When You Connect

- 01 Check test batch health instantly. Instead of navigating dashboards, ask your agent to use `get_batch_stats` for immediate passed/failed counts.

-
- 02 Pinpoint visual bugs exactly. Use `list_results` or `get_session` to inspect specific failed step images and browser differences.

 - 03 Keep baselines clean. Easily manage outdated assets by running `delete_baseLine` or deleting old batches with `delete_batch` when they're no longer needed.

 - 04 Target branch validation. Use `list_branch_baselines` to confirm the visual state of a feature branch without polluting your main baseline set.

 - 05 Verify connectivity upfront. Run `validate_key` first; it confirms your API key works before you waste time on actual testing cycles.
-

Real-World Applications

Investigating an unstable release candidate

The QA Engineer notices test failures across multiple browsers. They ask the agent to list all recent batches, identify the failing ones using `list_batches`, and then dive into specific sessions via `get_session` to understand if it's a cross-browser alignment issue.

Cleaning up test environments

The Engineering Manager realizes old testing data is cluttering the system. They ask the agent to list all baselines via `list_baselines`, identify outdated entries, and authorize their deletion using `delete_baseline`.

PR review for design changes

The Frontend Developer commits code related to a new header component. Before merging, they ask the agent to list baselines specific to their feature branch using `list_branch_baselines` and confirm that no unintended visual regressions were introduced.

Determining overall test readiness

The team needs a quick health check for the entire application. They ask the agent to get summary statistics (`get_batch_stats`) for the latest main branch batch, confirming that all critical paths have passed.

Patterns to Avoid

Treating Applitools like a database

✗ AVOID

Trying to manually copy and paste IDs or data from the web interface into your agent's prompt, which is slow and prone to human error.

✓ INSTEAD

Instead, ask your agent to execute ``list_batches`` first. Then, tell it to get the full details of a specific batch using ``get_batch``, letting the tool handle all the ID lookups.

Assuming baselines are always current

✗ AVOID

Running tests and assuming that because the test passed, the visual baseline is accurate without checking for necessary updates.

✓ INSTEAD

Always run ``list_branch_baselines`` when merging. If the feature changed significantly, ask the agent to review if those specific baselines are outdated before proceeding.

Ignoring test history management

✗ AVOID

Letting old, irrelevant test batches pile up and cluttering the platform view over time.

✓ INSTEAD

Periodically ask your agent to run ``list_batches`` and then use ``delete_batch`` on any completed or obsolete cycles. This keeps the workspace focused.

The Right Fit

Use this MCP if you need to validate UI changes visually across multiple browsers, operating systems, or specific feature branches without opening a dedicated testing dashboard. It's perfect for QA Engineers who need summarized status reports (like using `get_batch_stats`) and Developers who want immediate visual confirmation during PRs (using `list_branch_baselines`). Don't use this if your only goal is to manage simple text-based data, or if you just need a basic API key check. For that, the simpler `validate_key` tool suffices. If you need access to other types of testing results—say, performance metrics unrelated to visuals—you'll need a different category of MCP.

Applitools Visual Testing for UI Quality Assurance

Right now, checking visual consistency is a manual nightmare. You have to run the tests in the platform, then open dozens of reports, click through failures across Chrome and Safari, and copy screenshots just to prove that one button alignment drifted by 12 pixels.

With this MCP, you simply ask your agent to inspect test batches for visual diffs. It gathers the failure details, pinpoints the exact step, and tells you precisely which browsers caused the mismatch. You get actionable reports without ever touching a dashboard.

Applitools Visual Testing for Frontend Development Workflow

Before this MCP, verifying changes to a specific feature branch meant manually checking its baselines and hoping nothing broke. It was slow, risky, and required constant context switching between the IDE and the testing portal.

Now, you just ask your agent to list baselines for that exact branch. The result gives you immediate visibility into what changed and whether those visual assets are ready for merge. You keep development focused on code.

Applitools: 10 Tools for Visual Testing and QA Validation

These tools allow your agent to perform specific actions like listing, deleting, or retrieving detailed reports on Applitools test batches and baselines.

#	TOOL	DESCRIPTION
01	<code>list_baselines</code>	Lists the IDs and names of visual baselines for a specific application.
02	<code>get_batch_stats</code>	Provides summary counts (passed, failed, unresolved) for an entire Applitools test batch.
03	<code>list_batches</code>	Lists all active test batches and their current statuses like Passed or Failed.
04	<code>list_branch_baselines</code>	Retrieves visual baselines tied to a specific Git branch name.
05	<code>delete_baseline</code>	Removes an Applitools test baseline when it becomes outdated or irrelevant.
06	<code>delete_batch</code>	Permanently removes an entire Applitools test batch; use this only if you know it's safe to delete.
07	<code>get_batch</code>	Retrieves full, granular details for a specific Applitools batch ID.
08	<code>list_results</code>	Lists all individual test results contained within a specified Applitools batch.
09	<code>get_session</code>	Gets deep details about one particular test session by providing the necessary batch and session IDs.
10	<code>validate_key</code>	Checks your Applitools API key to confirm connectivity before running any visual tests.

See It in Action

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

U List the most recent failed test batches and give me a summary of the failures.



Visual Test Batch Summary

The following 3 batches were recently processed:

- V2 Checkout Redesign (ID: b_991x) - **Unresolved**. Status shows 4 failed sessions out of 10 total.
- Login Dropdown Fix (ID: b_773z) - **Failed**. Contains 2 critical failures across Safari/iOS.
- Marketing Footer (ID: b_882y) - Passed. Looks clean.

I recommend we focus on the unresolved issues in the Checkout Redesign batch.

U What are all the baselines set up for the 'feature/dark-mode' branch?



Baselines for feature/dark-mode

I found 2 active baselines tied to this branch:

1. **ID:** bs_4ab (Name: Nav Bar Base) - Created 3 days ago.
2. **ID:** bs_5cd (Name: Mobile Hamburger Dark) - Created 3 days ago.

These look like they haven't been updated recently. Should I delete them if you confirm the feature is stable?

U Can you check the full details for batch b_991x?



Details: V2 Checkout Redesign (ID: b_991x)

- **Status:** Unresolved
- **Total Sessions:** 10
- **Failures Detected:** 4 sessions.
- **Primary Issue Area:** 'Payment Modal' step.
- **Mismatch Details:** Applitools AI flagged a significant 12% pixel drift in button alignment when comparing Chrome 114 (Win) vs Safari 16 (iOS).

Frequently Asked Questions

01 How does the Applitools MCP help me check visual bugs without opening the dashboard?

You can ask your agent to run reports and analyze test results directly. The agent pulls failure data, shows you pixel drift percentages, and identifies the exact step where the UI deviated from the expected baseline.

02 Can I use Applitools MCP to manage my different feature branches?

Yes. You can ask the agent to list baselines tied specifically to a branch name (like 'feature/dark-mode'). This lets you validate that changes only affect their specific code path, keeping your main baseline clean.

03 What if I need to delete an old test batch or baseline?

The MCP gives you tools to manage clutter. You can list all baselines and then authorize the agent to delete outdated ones using ``delete_baseline``, keeping your platform tidy.

04 Does Applitools MCP only work on web browsers?

No, it handles visual comparisons across multiple browser and OS combinations. It's designed to spot UI inconsistencies regardless of the underlying client environment you test against.

05 How do I get a quick summary of my recent test run status?

Instead of sifting through huge reports, ask your agent to get batch statistics. It instantly summarizes results for all active batches with simple counts of Passed, Failed, or Unresolved.

06 What is the difference between listing baselines and getting branch baselines?

Listing all baselines gives you a global view of every baseline. Listing branch baselines narrows that focus down, showing only the visual assets tied to one specific Git development branch.

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 `"applitools": { "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

Applertools 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](https://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 Applitools. 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	Applitools MCP
Server ID	019d7550-b318-71ae-89d7-c19f61e7c50a
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/applitools.