MCP SERVER | NO CODE | CLOUD HOSTED

Tenderly MCP

Test Ethereum Logic Without Spending Gas

Tenderly (Ethereum Dev Platform) MCP connects your AI agent directly to Ethereum development tools. Simulate transactions and full transaction bundles without spending gas or touching live funds. You can build private virtual test networks, monitor specific on-chain events in real time, and validate complex contract logic before deployment.

A+ Quality Score 100/100

smart-contracts | ethereum | transaction-simulation | web3-debugging | testnet | blockchain-monitoring

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/tenderly-ethereum-dev-platform

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.

Tenderly (Ethereum Dev Platform) MCP

4 tools available
Cloud-hosted on Vinkius

Need to debug smart contracts? This MCP lets you simulate any Ethereum action—from a single token transfer to a massive, multi-step DeFi protocol interaction—without spending gas or committing actual assets. Your agent handles the heavy lifting, allowing you to test code logic in an environment that feels exactly like mainnet.

Want to test your dApp against live production conditions? You can spin up private virtual networks that mimic Mainnet at any specific block height. If you're working with complicated contract interactions, you can also simulate entire bundles of transactions together just to see how they play out. Plus, set up advanced alerts for specific method calls or state changes, keeping you instantly informed about critical blockchain activity. It’s a powerful debugging suite that connects easily through Vinkius, so connecting your favorite AI client is straightforward.

Core Capabilities

01 — Test transaction outcomes

Simulate single transactions or complex groupings of actions without spending any real gas.

02 — Build private test environments

Create and manage virtual replicas of the main Ethereum network for isolated testing.

03 — Monitor blockchain activity

Set up alerts to get real-time notifications when specific contract events happen on the chain.

04 — Validate multi-step workflows

Run sequences of transactions together to see how complex protocols interact.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/tenderly-ethereum-dev-platform** — connect your AI agent in three steps.

01 First, subscribe to this MCP and provide your specific Tenderly access credentials: the Access Key, Account Slug, and Project Slug.

02 Next, tell your AI client exactly what you need. Do you want to simulate a transfer? Should you set up an alert for a state change? Or maybe create a virtual test network?

03 The MCP executes the request on Tenderly and returns the results—whether that's a successful simulation report or confirmation that your private TestNet is ready.

The bottom line is: you talk to your agent using natural language, and it handles all the complex blockchain setup and execution for you.

Built For

Smart Contract Engineers who hate writing boilerplate test scripts. DeFi Researchers who need to model highly specific, multi-step protocol interactions. Web3 DevOps teams managing production monitoring.

Smart Contract Engineer

Debugging a payment contract means simulating thousands of possible inputs and verifying every edge case before deployment.

DeFi Researcher

Analyzing how three different protocols interact requires running multi-step transaction bundles to trace funds accurately.

Web3 DevOps Engineer

Monitoring a live production contract means setting up alerts for specific method calls, ensuring immediate notification when things go wrong.

What Changes When You Connect

01 Stop wasting time on basic testing. Use `simulate_transaction` to run dry simulations of any transfer or call, confirming contract behavior before you deploy anything.

02 Need a clean slate for development? With `create_virtual_testnet`, your agent builds private mainnet replicas so you can test against custom configurations without affecting production data.

03 Forget manually checking logs. Use `create_alert` to monitor specific method calls or state changes, getting instant notification the moment an event happens on-chain.

04 Complex DeFi logic demands more than one step. Running a full sequence via `simulate_bundle` lets you test multi-protocol interactions as if they happened live.

05 Your agent manages all this complexity across multiple tools, giving you full visibility into transaction lifecycles from a single prompt.

Real-World Applications

Debugging an approval flow

A smart contract engineer needs to verify if an ERC20 token will successfully approve funds for a new router address. They ask their agent to `simulate_transaction` on the specific addresses and data, instantly seeing the resulting Approval event without using real test tokens.

Testing a new protocol version

A DeFi researcher is building out a concept that needs Mainnet conditions but shouldn't touch live funds. They instruct their agent to `create_virtual_testnet` forked from the current Mainnet block, giving them a safe space to test complex interactions.

Monitoring for fraud

A Web3 DevOps team suspects a key contract is being misused. They ask their agent to `create_alert` specifically watching for unauthorized value transfers or state changes on that contract, getting immediate alerts 24/7.

Validating multi-signature governance

A team needs to ensure a sequence of three different committee votes works correctly. They use `simulate_bundle` to run the entire flow in one go, ensuring that the transactions don't conflict and finalize successfully.

Patterns to Avoid

Manual contract testing

✗ AVOID

Writing dozens of separate test files for every minor change. This is slow, repetitive, and often misses complex interaction bugs.

✓ INSTEAD

Instead, let your agent use `simulate_transaction` or `simulate_bundle`. You define the desired outcome once, and the MCP runs the dry test multiple times to cover all variables.

Ignoring dependencies

✗ AVOID

Testing a protocol's final step without considering how the previous two steps set up the required state or tokens.

✓ INSTEAD

Always use `simulate_bundle`. Grouping transactions allows you to validate the entire dependency chain in one single, accurate run.

The Right Fit

Use this MCP if your primary workflow involves deep contract debugging, testing complex state changes, or monitoring live blockchain activity. If you need to know *what happens* when a transaction runs—without actually spending gas—this is the tool. You'll use it for single-point validation (`simulate_transaction`), multi-step logic checking (`simulate_bundle`), or continuous oversight (`create_alert`). Don't use this if you just need to read historical data; for that, look at a simple block explorer API connector. Also, don't use it if your goal is writing the contract itself; this MCP only tests what already exists.

Debugging Ethereum Contracts Used To Be Painful

Today, testing a smart contract means setting up local forks, manually calling functions one by one across multiple terminal windows, and constantly worrying about which gas token you'll accidentally spend on the wrong test. You copy data from one

Now, your agent handles all that mess. By connecting this MCP, you simply tell your AI client what needs testing—whether it's a single transfer or a whole suite of interactions. The result is an

dashboard into a spreadsheet just to track state changes.

instant report showing the outcome and gas consumption, letting you focus purely on the logic.

Simulate Transactions with Tenderly MCP

You no longer need to deploy a full local fork just to check if an approval function works. You don't have to spend time managing complex test accounts or worrying about gas costs for minor checks.

With this MCP, you initiate the dry run from your chat interface and get immediate feedback on state changes. It’s that simple.

Tenderly (Ethereum Dev Platform) MCP with 4 Tools

These four tools let you fully debug, simulate, and monitor Ethereum contract behavior using natural language commands.

#	TOOL	DESCRIPTION
01	`create_alert`	Set up real-time monitoring to notify you whenever a defined blockchain event occurs on Tenderly.
02	`create_virtual_testnet`	Instantly spin up a private, isolated copy of the Ethereum network for controlled development testing.
03	`simulate_bundle`	Run and evaluate multiple transactions as a single group to test complex workflow dependencies.
04	`simulate_transaction`	Perform a dry run of any single transaction, showing you the resulting state changes without spending gas.

See It in Action

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

U Simulate a transaction on Ethereum Mainnet from 0xd8dA6BF26964aF9D7eEd9e03E53415D37aA96045 to 0x7a250d5630B4cF539739dF2C5dAcb4c659F2488D with data 0x095ea7b3...

I've simulated the transaction. The simulation was successful, consuming 45,000 gas. It would result in an ERC20 Approval event for the Uniswap V2 Router. Would you like to see the full state changes?

U Create a Virtual TestNet named 'Staging-Fork' forked from Mainnet at the latest block.

Virtual TestNet 'Staging-Fork' has been created successfully. You can now access it via the generated RPC URL. The Chain ID is set to your specified virtual ID.

U Set up an alert to notify me whenever the 'transfer' method is called on contract 0x123...

I've created the alert for contract 0x123... monitoring the 'transfer' method. You will receive notifications based on your Tenderly project settings whenever this event is detected on-chain.

Frequently Asked Questions

01 **Can I use Tenderly (Ethereum Dev Platform) MCP to test transactions?**

Yes, you can simulate any single transaction or a group of them using `simulate_transaction` or `simulate_bundle`. This runs the test without spending actual gas.

02 **What is a Virtual TestNet in Tenderly (Ethereum Dev Platform) MCP?**

A Virtual TestNet lets you create a private, temporary copy of the Ethereum network. You can use this environment to test your dApp against production-like conditions safely.

03 **Does Tenderly (Ethereum Dev Platform) MCP require actual funds?**

No. The entire purpose of this MCP is simulation and monitoring, so you never risk spending real assets or gas fees while developing.

04 **How do I monitor live events with Tenderly (Ethereum Dev Platform) MCP?**

You use the `create_alert` tool. You specify exactly what event or state change you want to track, and your agent handles setting up the persistent monitoring.

05 **Is Tenderly (Ethereum Dev Platform) MCP compatible with all AI clients?**

Yes, because it's an open standard through Vinkius, any MCP-compatible client like Cursor or Claude can connect to and use its tools.

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 `"tenderly-ethereum-dev-platform": { "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

Tenderly (Ethereum Dev Platform) 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 Tenderly (Ethereum Dev Platform). 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	Tenderly (Ethereum Dev Platform) MCP
Server ID	`019e38f8-f322-72a6-82c6-88d0ff795ba8`
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/tenderly-ethereum-dev-platform.