MCP SERVER | NO CODE | CLOUD HOSTED

GetBlock (Web3 RPC Provider) MCP

Analyze block states and track crypto assets in conversation.

GetBlock (Web3 RPC Provider) gives your AI agent direct, high-performance access to over 50 major blockchain networks. Query everything from Ethereum and Solana balances to raw Bitcoin transaction details using specialized tools like `eth_get_balance` or `btc_getblockchaininfo`. It lets you read the entire state of a chain simply by asking your client a question.

F Quality Score 3.6/100

rpc-nodes | web3 | blockchain-infrastructure | ethereum | solana | crypto-data

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/getblock-web3-rpc-provider

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.

GetBlock (Web3 RPC Provider) MCP

26 tools available
Cloud-hosted on Vinkius

Need to understand what's happening on-chain? This MCP connects your AI agent directly to dozens of blockchain protocols, giving it real-time data feeds for Ethereum, Solana, Bitcoin, and many others. Instead of building complex API wrappers or juggling multiple node endpoints, you just let your client ask the questions. It handles all the heavy lifting, whether you need a simple balance check (`sol_get_balance`) or deep debugging traces on a smart contract. Because Vinkius hosts this MCP, you connect once and instantly gain access to these specialized crypto tools, making complex blockchain data available through natural conversation.

Core Capabilities

01 — Check account balances across multiple chains

Determine the native coin or token holdings for any address on supported networks using specific balance retrieval tools.

02 — Inspect block and network metadata

Fetch current block numbers, difficulty metrics, and general chain status information from various blockchains.

03 — Analyze transaction history in depth

Retrieve full details for transactions by hash, including receipts, gas estimates (`eth_estimate_gas`), or detailed execution traces.

04 — Build and submit raw Bitcoin transactions

Create partially signed Bitcoin transaction formats (PSBT) and submit them as raw hex-encoded data to the network for processing.

05 — Execute generic RPC calls

Run virtually any standard JSON-RPC method supported by the GetBlock nodes using the `rpc_call` tool.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/getblock-web3-rpc-provider** — connect your AI agent in three steps.

01 Subscribe to this MCP and provide your unique GetBlock Access Token (API Key) to your AI client.

02 Your agent identifies the blockchain network and data points needed, then executes specific tools like `eth_get_balance` or `sol_get_account_info`.

03 The MCP processes the request through the high-performance RPC nodes and returns structured, actionable JSON data that your agent can interpret.

The bottom line is you get reliable, multi-chain blockchain intelligence delivered directly into your AI workflow without writing a single wrapper function.

Built For

Anyone who spends time looking at transaction logs or needing to validate on-chain data. It's for the DeFi analyst who needs real-time balances, the Web3 developer stuck debugging smart contract calls, or the research engineer tracking network metrics across multiple chains.

Web3 Developer

Debugging a failed transaction requires running `debug_trace_transaction` and checking gas limits using `eth_estimate_gas`, all without leaving the terminal.

Crypto Analyst

Pulling historical data on Bitcoin network health, like getting the latest block count (`btc_getblockcount`) or overall blockchain info (`btc_getblockchaininfo`), for quarterly reports.

DeFi Researcher

Monitoring wallet stability by checking token balances across different chains using `sol_get_balance` and comparing them to Ethereum holdings via `eth_accounts`.

What Changes When You Connect

01 Eliminate manual data fetching. Instead of opening 5 different tabs to check Ethereum, Solana, and Bitcoin stats, you ask your agent directly for the `eth_get_balance` or `sol_get_balance`, and it handles the complexity.

02 Deep contract inspection becomes easy. If a transaction fails, don't guess why. Use `debug_trace_transaction` to get the full execution path and pinpoint the exact line of faulty code.

03 Stay ahead with real-time block data. Get the latest Ethereum block height using `eth_block_number` or track Bitcoin progress instantly via `btc_getbestblockhash`, making your reports current.

04 Build complex transaction logic without writing boilerplate API calls. Use the versatile `rpc_call` tool to execute any standard JSON-RPC method, giving you maximum flexibility for niche queries.

05 Handle raw data submissions accurately. Need to move funds or create a specialized BTC transfer? Use `btc_createpsbt` and then submit it with `btc_sendrawtransaction` through the MCP.

Real-World Applications

Auditing a DeFi contract interaction

A user needs to know if a specific smart contract call succeeded or failed. They ask their agent, which uses `debug_trace_transaction` on the transaction hash, providing a full breakdown of state changes and gas usage.

Comparing cross-chain asset values

A researcher needs to compare an address's ETH balance versus its SOL balance. The agent calls both `eth_get_balance` and `sol_get_balance`, returning a unified view of the assets across different chains.

Verifying blockchain network health

An analyst needs quick, current stats on Bitcoin's overall size. The agent calls `btc_getblockchaininfo` to get difficulty and block count instantly, rather than navigating multiple explorer dashboards.

Modeling a new token transfer

A developer wants to see if sending funds will cost too much gas before writing code. They use `eth_estimate_gas` first, confirming the budget before running any actual transaction tools like `sol_send_transaction`.

Patterns to Avoid

Calling specific endpoint APIs

✗ AVOID

Manually writing code to call Ethereum's balance API, then having to repeat the entire function for Solana, and again for Bitcoin. It's a mess of redundant boilerplate.

✓ INSTEAD

Use this MCP. Your agent handles the multi-chain complexity by calling simple tools like `eth_get_balance` or `sol_get_balance`, unifying the data retrieval process.

Only checking basic transaction status

✗ AVOID

Just confirming a transaction was submitted, but not knowing if it ran out of gas or failed mid-execution. You're left guessing why the transfer stopped.

✓ INSTEAD

Use `debug_trace_transaction` to get the complete execution trace and understand exactly where the failure occurred on any supported chain.

Using generic HTTP requests for RPC

✗ AVOID

Building a catch-all JSON-RPC wrapper that requires you to manually map every single function name (e.g., `eth_get_block` vs `btc_get_block`). It breaks when the network updates.

✓ INSTEAD

Rely on the pre-built, dedicated tools like `eth_get_block_number` and the powerful fallback of `rpc_call`, which supports any standard JSON-RPC method.

The Right Fit

Use this MCP if your workflow requires reading state data from multiple disparate blockchain sources (Ethereum, Solana, Bitcoin, etc.)—especially when you need to debug complex interactions or compare balances across chains. The tools like `eth_get_balance` and `sol_get_balance` are perfect for that multi-chain comparison. Don't use it if all your needs fit within a single type of database query (e.g., only querying relational data, not blockchain state). If

you just need to read simple text or manage local records, an internal knowledge base connector works better. However, if the source of truth is 'on-chain,' this MCP's depth—from `btc_getblockchaininfo` all the way down to transaction debugging with `debug_trace_transaction`—makes it necessary.

Blockchain data used to be a nightmare to stitch together.

Today, getting a full picture of assets requires jumping between dozens of web interfaces. You check ETH on Etherscan, then switch tabs to Solana Explorer for SOL balances, and finally open a separate terminal just to view the latest Bitcoin block height. It's clicking through multiple dashboards and copy-pasting identifiers until you get dizzy.

With this MCP, you just ask your agent: 'What is the total value across my ETH, SOL, and BTC holdings?' The system handles the necessary calls—like `eth_get_balance` and `sol_get_balance`—and gives you one clean answer. No tab switching required.

GetBlock (Web3 RPC Provider) MCP delivers granular, actionable data.

You no longer have to write separate API handlers for every blockchain network or transaction type. Instead of building custom logic to trace a call's path, you simply invoke `debug_trace_transaction` and get the entire sequence of events right away.

The difference is that your agent doesn't just *tell* you data; it retrieves validated, structured state information across dozens of networks instantly. It changes how fast you can research and build.

GetBlock (Web3 RPC Provider) - 26 Tools

These tools allow your agent to execute specific read/write operations across major blockchains like Ethereum, Solana, and Bitcoin.

#	TOOL	DESCRIPTION
01	`btc_analyzepsbt`	Analyzes a Partially Signed Bitcoin Transaction (PSBT) to provide necessary information.
02	`btc_createpsbt`	Generates the structured data needed for a partially signed Bitcoin transaction.
03	`btc_getbestblockhash`	Fetches the hash identifier of the current, most advanced block on the Bitcoin chain.
04	`btc_getblockcount`	Retrieves the total number of blocks recorded in the longest Bitcoin blockchain history.
05	`btc_getblockchaininfo`	Gathers comprehensive status details about the entire Bitcoin network structure and state.
06	`btc_sendrawtransaction`	Submits raw, hex-encoded transaction data directly to a local node and the wider network.
07	`debug_trace_block_by_number`	Provides a full execution trace for every single transaction contained within a specified block number.
08	`debug_trace_call`	Simulates and provides an execution trace when calling a function (eth_call) on the blockchain context.
09	`debug_trace_transaction`	Generates a complete, step-by-step execution path for any given transaction signature.
10	`eth_accounts`	Lists all public wallet addresses that are currently associated with your client's credentials.
11	`eth_block_number`	Gets the number of the most recently confirmed block on the Ethereum network.
12	`eth_call`	Runs a function call immediately without spending funds or creating an actual transaction.
13	`eth_estimate_gas`	Calculates the precise amount of gas needed for a proposed Ethereum transaction to complete successfully.

#	TOOL	DESCRIPTION
14	eth_get_balance	Returns the current Ether balance associated with a specific Ethereum wallet address.
15	eth_get_block_by_hash	Fetches all information related to an entire block using its unique hash identifier.
16	eth_get_block_by_number	Retrieves the detailed data for a specific Ethereum block number.
17	eth_get_code	Pulls the compiled smart contract code associated with any given address.
18	eth_get_transaction_by_hash	Retrieves all metadata for a transaction when you know its unique hash identifier.
19	eth_get_transaction_receipt	Gets the official receipt from an Ethereum transaction, confirming execution details and status.
20	rpc_call	Allows you to execute any general JSON-RPC method supported by the GetBlock nodes directly through your agent.
21	sol_get_account_info	Retrieves all associated account metadata for a specific public key on the Solana network.
22	sol_get_balance	Checks the available native SOL balance for any given Solana wallet address.
23	sol_get_latest_blockhash	Gets the current block hash required to validate transactions on the Solana ledger.
24	sol_get_token_account_balance	Determines the token balance for a specific SPL Token account address on Solana.
25	sol_get_transaction	Retrieves full details and confirmation status for a submitted Solana transaction signature.
26	sol_send_transaction	Submits a signed transaction to the Solana cluster, initiating its processing across the network.

See It in Action

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

U What is the current Ethereum block height?

The most recent Ethereum block number is 19283746 (0x1263FBA).

U Check the ETH balance for address 0x742d35Cc6634C0532925a3b844Bc454e4438f44e.

The balance for that address is 1.25 ETH.

U Get the latest Bitcoin blockchain info.

I've retrieved the Bitcoin network status: Current blocks: 832104, Difficulty: 75.73T, Network Hashrate: 580.21 EH/s.

Frequently Asked Questions

01 **How do I check an Ethereum balance using GetBlock (Web3 RPC Provider) MCP?**

You use the `eth_get_balance` tool. You simply provide the target address, and the MCP returns the current Ether balance in a readable format.

02 **Can I debug smart contracts with GetBlock (Web3 RPC Provider) MCP?**

Yes, you can. Use `debug_trace_transaction` to get a complete execution trace for any transaction hash, showing exactly how the contract ran and where it might have failed.

03 **What is the difference between `eth_call` and sending a real transaction?**

`eth_call` executes a message immediately without spending gas or creating a permanent record. It's purely for checking contract logic, while tools like `sol_send_transaction` actually push funds to the network.

04 **Does GetBlock (Web3 RPC Provider) MCP support Bitcoin transactions?**

Yes. You can use specialized Bitcoin tools such as `btc_createpsbt` to build transactions and then submit them with `btc_sendrawtransaction`.

05 **What if I need a function not listed in the tools? Can GetBlock (Web3 RPC Provider) MCP handle it?**

The `rpc_call` tool acts as a catch-all. If you know the standard JSON-RPC method, you can execute virtually any other supported command through this single tool.

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 `"getblock-web3-rpc-provider": { "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

GetBlock (Web3 RPC Provider) 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 GetBlock (Web3 RPC Provider). 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	GetBlock (Web3 RPC Provider) MCP
Server ID	`019e389e-21e3-714e-a447-a08fa031b5fe`
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/getblock-web3-rpc-provider.