MCP SERVER | NO CODE | CLOUD HOSTED

ElectricSQL MCP

Stream Live Postgres Data into Your AI Agent's Context

ElectricSQL Sync Engine MCP brings real-time data from Postgres directly to your AI client. It lets you fetch structured subsets of database information, track changes over time, and query live application data right within your chat window.

A+ Quality Score 100/100

postgres | real-time-sync | data-streaming | incremental-updates | http-api | database-sync

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/electricsql-sync-engine

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.

ElectricSQL (Sync Engine) MCP

2 tools available
Cloud-hosted on Vinkius

This MCP connects your AI agent straight into a Postgres database, letting it stream live data without needing complex backend setups. You can ask questions about specific tables or monitor system state in real-time. It handles initial data dumps—getting the whole picture at once—and then keeps you updated as records change, using efficient log offsets for incremental syncs. Need to query a niche group of records? Use advanced filtering and subset snapshots, letting your agent handle complex 'WHERE' clauses that would normally break a URL. Because Vinkius hosts this MCP, connecting it is simple: link up once from any compatible client, and you immediately gain access to live database context for analysis or debugging.

Core Capabilities

01 — Initial Data Snapshots

Fetch a complete, historical snapshot of a data table using an initial sync call.

02 — Complex Subset Queries

Query specific groups of records—for example, only 'pending' orders in the last month—using advanced filtering methods.

03 — Real-time Data Streaming

Keep your agent continuously updated as data changes in the underlying Postgres database via long-polling or SSE.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/electricsql-sync-engine** — connect your AI agent in three steps.

01 Subscribe to this MCP and provide your specific Electric Service URL, pointing it toward your live Postgres instance.

02 Your AI client initiates a data request using the appropriate method—either requesting an initial sync or setting up a continuous stream.

03 The agent receives structured, filtered data (a 'shape') directly into its context, allowing it to answer questions based on the current database state.

The bottom line is that your AI client sees your Postgres data as if it were local memory, giving you live application insight without writing a single integration script.

Built For

This MCP is for developers and engineers who can't afford to waste time building fragile, custom APIs just to give their AI agent data access. It’s perfect for anyone needing live system state—from debugging production issues to letting product teams monitor real-time user activity.

Backend Developer

Building internal tools where the AI needs immediate, verifiable access to application records (e.g., 'What are all the users who signed up in the last hour?').

Data Engineer

Testing data pipelines or debugging complex ETL processes by letting the agent inspect specific database 'shapes' and sync subsets.

Product Manager

Monitoring real-time user activity or system health directly within a chat interface without needing to open up multiple dashboards.

What Changes When You Connect

01 You get live data streams. Instead of polling a database every few seconds, the agent stays connected and gets instant updates whenever records change.

02 Complex filtering is simple. You don't have to write complicated API endpoints for niche queries; you just tell the agent what subset of data you need.

03 It handles massive datasets efficiently. The built-in support for limits and offsets means you can browse huge tables without hitting memory or URL length restrictions.

04 Use get_shape for initial dumps. When you start a project, you can pull an entire table snapshot quickly using this tool to establish context immediately.

05 The whole process avoids complex plumbing. You connect your AI agent through Vinkius, and the data sync handles itself, letting you focus on logic, not infrastructure.

Real-World Applications

Debugging a User Flow Issue

A backend developer notices a user reported an incorrect order total. Instead of running manual SQL queries across multiple tables, they ask their agent to use post_shape to query the 'orders' and 'line_items' tables for that specific user ID. The agent returns all relevant data shapes, confirming where the calculation failed.

Monitoring System Health

A product manager needs to see if a new feature is causing errors in real-time. They set up a live sync on the 'error_logs' table. As soon as an error is written to Postgres, the agent immediately alerts them and provides the full context of the failure.

Analyzing Campaign Leads

A marketing team wants to see all leads from a specific zip code who haven't opened an email in 30 days. They use post_shape, defining complex WHERE clauses for both geography and activity status. The agent returns only the actionable subset of contacts.

Initial Project Context Build

A new data analyst needs to understand a legacy system's structure. Using get_shape with an offset=-1, they pull a full snapshot of the 'user_profiles' table, giving their agent enough context to answer structural questions immediately.

Patterns to Avoid

Manual API Calls for State Checks

✗ AVOID

Manually writing and maintaining dozens of endpoint calls (e.g., `/api/user-status`, `/api/inventory?id=123`) to check the current state of a system.

✓ INSTEAD

Connect this MCP instead. Use the agent's built-in data streaming capabilities or post_shape to query the underlying Postgres tables directly. The agent handles the API structure for you.

Relying on Cached Data

✗ AVOID

Building an AI workflow that relies on a data cache refreshed every hour, meaning critical information is always outdated when the user asks about it.

✓ INSTEAD

Use this MCP's real-time sync capabilities. It keeps your agent long-polling against Postgres, ensuring any change in the database is instantly visible to your query.

Ignoring Data Structure

✗ AVOID

Trying to pass an entire, unfiltered Postgres table into the chat context because it 'might contain what you need.' This overloads the agent and introduces noise.

✓ INSTEAD

Use post_shape. Define specific subsets of data using precise filtering (WHERE clauses) and column selection. This keeps the information clean and actionable for your agent.

The Right Fit

Use this MCP if your core problem is getting live, structured application state—data that lives in a database—into an AI workflow without building complex middleware. You need to monitor changes or run specific queries against tables like 'orders' or 'user_logs.'

Don't use it if you only need to process unstructured files (like

PDFs) or interact with simple APIs that don't involve relational data. For those cases, a file ingestion MCP is better.

If your goal is simply to run a single SQL query and get one result without needing real-time context, a standalone database querying tool might suffice. But if you need the agent to *know* what changed over time, this MCP's streaming capabilities are necessary.

Database state used to be hard to access for AI

Today, if your AI needs to know something—like the current inventory count or a user's latest activity—you usually have to write dedicated API endpoints. That means building an endpoint just for 'user status,' another one for 'order history,' and then linking them all up so they don't time out when data changes.

With this MCP, you bypass that whole process. You connect your agent once, point it at the Postgres database, and the system handles the complexity of fetching, filtering, and streaming live data shapes directly into your chat context. Your agent just knows how to ask.

ElectricSQL Sync Engine MCP gives you real-time data insight

The manual steps that disappear are writing the complex backend plumbing and managing state across multiple services. You eliminate the need for dedicated 'status check' endpoints, simplifying your entire architecture.

Now, your AI agent can treat the database as a living source of truth. It doesn't just answer based on what it was trained on; it answers based on exactly what's happening right now.

ElectricSQL (Sync Engine) with 2 Tools

These tools let you programmatically retrieve data from Postgres by fetching initial snapshots or submitting highly detailed queries for specific subsets of records.

#	TOOL	DESCRIPTION
01	`get_shape`	Use this tool to pull an initial snapshot of data from Postgres using a standard GET request.
02	`post_shape`	Send complex queries and sync subsets of data out of Postgres by submitting them via POST.

See It in Action

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

U Sync the 'public.items' table from the beginning using get_shape.

I'm initiating the initial sync for 'public.items' with offset -1. I've retrieved 150 rows. The shape handle is 'items-handle-xyz'. Would you like to see the data or wait for live updates?

U Use post_shape to get the first 10 rows of 'orders' where status is 'pending', ordered by date.

Querying 'orders' via POST... I found 10 pending orders. The most recent is Order #502 from today. I've used the limit and order_by parameters as requested.

U Start a live sync for the 'messages' table to watch for new entries.

Live sync enabled for 'messages'. I am now long-polling the ElectricSQL engine. I will notify you immediately when new rows are detected in the database.

Frequently Asked Questions

01 **How does ElectricSQL Sync Engine MCP handle large datasets?**

It supports pagination and limits, allowing you to efficiently browse massive tables without overloading the system. This prevents errors related to dataset size.

02 **Is this good for continuous monitoring?**

Yes. You can enable live streaming via long-polling or SSE, meaning your agent updates instantly whenever new rows are added to a table like 'messages'.

03 **Do I need to write SQL code to use the MCP?**

No. While the underlying mechanism uses SQL concepts (shapes and WHERE clauses), you interact with it naturally by simply asking your agent questions about the data.

04 **What is the difference between get_shape and post_shape?**

Use get_shape for initial, basic snapshots. Use post_shape when you need to send complex filtering parameters or detailed subsets of data that exceed simple GET request limits.

05 **Can I sync multiple tables at once using ElectricSQL Sync Engine MCP?**

Yes, you can define and manage shapes across several different Postgres tables within the same agent workflow to get a holistic view of your application state.

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 `"electricsql-sync-engine": { "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

ElectricSQL (Sync Engine) 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 ElectricSQL (Sync Engine). 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	ElectricSQL (Sync Engine) MCP
Server ID	`019e388f-b8d8-70dd-990e-603c981098be`
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/electricsql-sync-engine.