MCP SERVER | NO CODE | CLOUD HOSTED

Nile MCP

Manage Multi-Tenant Data Isolation & Metrics

Nile (PostgreSQL for Multi-Tenant Apps) provides natural language control over complex, isolated database environments. Use this MCP to manage B2B tenant isolation, provision new data boundaries on demand, and pull real-time performance metrics without writing SQL or clicking dashboards.

A+ Quality Score 100/100

postgresql | multi-tenancy | saas-infrastructure | tenant-sharding | serverless-db | database-metrics

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/nile-postgresql-for-multi-tenant-apps

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.

Nile (PostgreSQL for Multi-Tenant Apps) MCP

6 tools available
Cloud-hosted on Vinkius

Managing multi-tenant databases is usually a pain point involving constant dashboard switching and brittle shell scripts. This MCP lets your AI client talk directly to your PostgreSQL infrastructure through Nile. You control everything from high-level database configuration to the individual user access rights within specific tenants, all via conversation.

Need to spin up a new department's isolated data space? Just ask. Need to know if storage is running low or if connections are spiking? Ask that too. Your agent handles the complexity of mapping those requests into precise actions against your virtual boundaries. It gives you full visibility into who can access what, and exactly how much compute power each tenant uses. Connecting this MCP through Vinkius means you get to manage all your complex database needs—from provisioning new tenants to auditing user access—all from one place in any compatible AI client.

Core Capabilities

01 — Inventory Databases

List all high-level tenant-aware PostgreSQL databases provisioned on Nile for immediate context.

02 — Monitor Performance Metrics

Pull exact operational numbers showing database strain, active connections, and compute utilization over time.

03 — Provision Tenant Boundaries

Instantly create highly isolated virtual tenant boundaries within an existing database to support new B2B clients.

04 — Audit User Access

Enumerate all globally tracked users and map their identities to specific internal tenants for access control checks.

05 — Inspect Tenant Data Isolation

Query the strict virtual boundaries that split tenant data, allowing deep inspection of exact SaaS IDs before querying records.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/nile-postgresql-for-multi-tenant-apps** — connect your AI agent in three steps.

01 Subscribe to this MCP and provide your specific Nile API URL and API Key.

02 Connect your AI agent (like Claude or Cursor) through the Vinkius marketplace using these credentials.

03 Ask a natural language question, like 'What are the active connections for my production environment?' Your agent executes the necessary tools and returns structured data.

The bottom line is that you manage your entire multi-tenant infrastructure by simply talking to it.

Built For

Database Engineers who hate running manual scripts, or SaaS Architects who need instant visibility into tenant isolation. If managing complex data boundaries across multiple clients is part of your job, you need this.

SaaS Developer

Needs to provision new B2B tenants and verify database isolation quickly without writing manual CLI scripts.

Database Engineer

Consistently monitors storage exhaustion levels and active connection counts across dozens of different Nile databases for performance alerts.

Backend Architect

Manages user identity mappings and efficiently shards tenant-specific data to maintain high scalability standards while auditing access control.

What Changes When You Connect

01 Provisioning Tenants: Instead of writing a provisioning script, you simply ask the agent to create a new isolated tenant boundary using `create_tenant`, instantly segmenting data for a new client.

02 Performance Monitoring: You get immediate visibility into database health by requesting operational metrics via `get_metrics`. It shows true active connections and storage usage without needing to jump through monitoring dashboards.

03 Auditing Access: The combination of `list_users` and `list_tenants` lets you audit user access patterns. Your agent correlates global identities to specific virtual tenants, ensuring strict access control compliance.

04 Deep Configuration Checks: Need to know the exact state of a production cluster? Use `get_database` to pull detailed configuration JSON directly into your chat window for quick validation.

05 Data Discovery: Before running deep data checks, use `list_databases` to map out all potential targets. This helps you scope exactly which tenant-aware PostgreSQL databases need attention.

Real-World Applications

Onboarding a New Enterprise Client

The architect needs to add 'Global Corp' as a new, isolated client in the main production database. They ask their agent to execute `create_tenant` with the name 'Global-Corp'. The MCP provisions the virtual boundary and confirms the ID, making the process instant and fully auditable.

Investigating Performance Bottlenecks

The database engineer notices slow query times. They ask their agent to run `get_metrics` on the affected cluster. The MCP returns current connection counts, storage exhaustion layers, and compute utilization instantly, pinpointing if the issue is resource-based.

Verifying Data Segmentation

A compliance officer needs to confirm that Client A's data can never leak into Client B's space. They use `list_tenants` and then prompt the agent to query the virtual boundaries, ensuring the segmentation is strictly enforced at the database level.

Mapping User Permissions

The security team needs a full picture of who has access. They run `list_users`, which gives them a master list of global users. They then use `list_tenants` to verify that each user is only associated with the tenants they are authorized for.

Patterns to Avoid

Assuming Global Visibility

✗ AVOID

A developer might try to run a general query against the whole database, believing it will filter by tenant automatically. This often results in massive data dumps or outright errors because they didn't specify the isolation boundary.

✓ INSTEAD

Always use `list_tenants` first to identify the specific virtual boundaries you need. Then, use natural language prompts that target a single tenant ID before running any data inspection.

Missing Performance Context

✗ AVOID

A user sees slow query times and assumes it's a code issue, spending hours debugging application logic when the problem is actually resource exhaustion.

✓ INSTEAD

Before troubleshooting code, always check `get_metrics`. This tool provides hard data on active connections and compute utilization, telling you immediately if you need to scale up resources instead of fixing logic.

Bypassing Tenant Setup

✗ AVOID

A new client is added manually by a sysadmin without proper isolation setup. The application starts failing because the data structure isn't ready for multi-tenancy.

✓ INSTEAD

When onboarding any new B2B client, always use `create_tenant`. This guarantees that the virtual boundary is provisioned correctly and isolated before any data enters.

The Right Fit

Use this MCP when your core challenge revolves around managing complex boundaries. If you are dealing with a SaaS architecture where multiple distinct clients (tenants) share a single underlying PostgreSQL cluster, this tool is essential. It gives you the ability to provision new tenant boundaries (`create_tenant`), audit user access across those tenants (`list_users`), and monitor resource usage per segment (`get_metrics`).

Don't use this if your primary need is simple CRUD operations on a single database or if you don't care about the isolation between clients. If all you need is to run a basic SQL query against one known table, a generic database connector will suffice. However, if that 'basic' query needs to respect complex virtual boundaries and

track usage across dozens of different client segments, this Nile MCP is what you need.

The Pain of Dashboard Jumps

Today, checking the health of a multi-tenant system means jumping between three or four dashboards: one for connection counts, one for storage usage, and another for user activity. You spend minutes clicking through tabs, manually correlating a high active connection count with an unknown tenant ID that's causing the spike.

With this MCP, you just ask your agent to check performance. It pulls all three metrics—connections, storage, and compute utilization—and gives you one clear answer in plain chat text. You don't click; you talk.

Nile (PostgreSQL for Multi-Tenant Apps) MCP: Instant Visibility

Manual processes like listing all tenants involve running `list_tenants` on the correct database, then having to manually cross-reference that list with a separate user management system to see who owns which tenant. It's slow and error-prone.

Now, you ask your agent to correlate identities and boundaries. The MCP handles both `list_users` and `list_tenants` in one sequence, giving you an immediate, accurate map of ownership that used to take a junior engineer half a day.

Nile (PostgreSQL for Multi-Tenant Apps) - 6 Tools

Use these tools to provision tenants, track performance metrics, map database structures, and manage user identity within your complex multi-tenant environment.

#	TOOL	DESCRIPTION
01	list_databases	Lists all top-level, tenant-aware PostgreSQL databases configured on Nile for querying.
02	get_database	Retrieves detailed configuration and current state information for any specific Nile database endpoint.
03	list_tenants	Lists all active, virtualized B2B tenants currently residing within a specified Nile database.
04	create_tenant	Provisions a new, highly isolated virtual tenant boundary inside the existing database structure.
05	list_users	Retrieves an enumeration of all globally tracked users who have access to various segments of tenant data.
06	get_metrics	Pulls precise operational performance numbers that track the overall strain and utilization of the database.

See It in Action

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

U List all active databases in my Nile account

I've retrieved your Nile databases. Highlights include 'production-v1' (Status: Active), 'staging-cluster', and 'analytics-db'. Would you like to see the configuration details or metrics for the production database?

U Create a new isolated tenant called 'Acme-Corp' in the 'production-v1' database

Provisioning virtual tenant boundary... Success. I've created 'Acme-Corp' (ID: ten-12345) inside your production database. This tenant is now logically isolated and ready for data sharding. Would you like me to verify the initial metrics for this new boundary?

U Show me the compute and storage metrics for database 'production-v1'

Retrieving performance metrics... For 'production-v1', you currently have 45 active connections out of a 100 limit. Storage usage is at 2.4 GB (12% of cap). Compute utilization averaged 15% over the last hour. Everything looks healthy. Would you like a list of active users?

Frequently Asked Questions

01 **How does the Nile MCP help with data isolation?**

The MCP provides tools like `create_tenant` and `list_tenants` which provision and manage highly isolated virtual boundaries (shards) inside Postgres, ensuring one tenant's actions don't affect another.

02 **Do I need to manually write SQL when using the Nile MCP?**

No. The goal is conversation-based control. You tell your agent what you want—like 'Show me storage exhaustion metrics'—and it handles the underlying database calls for you.

03 **Can I check connection limits using the get_metrics tool?**

Yes, `get_metrics` pulls operational data that tracks active connections. It tells you how many connections are currently open against your defined usage limits.

04 **What if I have a new client and need to add them? Should I use the Nile MCP?**

Yes, always use the MCP's `create_tenant` tool. This ensures the new B2B tenant is provisioned with proper virtual boundaries and isolation from day one.

05 **Does the Nile MCP help me audit who can access my data?**

Absolutely. You use `list_users` to see all global users, and then you can cross-reference that list against tenants using `list_tenants` for a full security picture.

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 `"nile-postgresql-for-multi-tenant-apps": { "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

Nile (PostgreSQL for Multi-Tenant Apps) 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

DOCUMENT INFORMATION

Generated	June 2026
MCP Server	Nile (PostgreSQL for Multi-Tenant Apps) MCP
Server ID	`019d75dd-9403-709b-949a-f75144e1306b`
Platform	Vinkius Cloud for AI Agents
Endpoint	`https://edge.vinkius.com/{token}/mcp`

LICENSE & USAGE