MCP SERVER | NO CODE | CLOUD HOSTED

Permit.io MCP

Govern access rules without writing code.

Permit.io lets you manage application authorization and access control policies conversationally. It handles complex rules—like checking if a user can read a document based on their department or role, or defining relationships between resources. Use it to build robust, fine-grained permission layers without writing code.

F Quality Score 3.6/100

authorization | rbac | rebac | abac | policy-as-code | access-control

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/permitio

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.

Permit.io MCP

18 tools available
Cloud-hosted on Vinkius

Stop building custom backend endpoints just to check basic permissions. This MCP connects your authorization layer to any AI agent, letting you govern access rules using plain language prompts. You can define roles and resources dynamically, whether you're setting up a brand new feature or auditing existing policies. Need to know if 'admin@company.com' can delete a file in the production environment? Your agent handles that check instantly. It also lets you provision users and tenants directly into your authorization system for testing or setup. When you connect this MCP via Vinkius, you get immediate access to industry-standard tools like AuthZen evaluation, meaning you don’t have to worry about vendor lock-in when checking permissions across different services. You manage the entire schema—creating roles, defining resources, and mapping complex relationships—all through your agent's conversation.

Core Capabilities

01 — Evaluate access rights

The system checks if a specific user has permission to perform an action on a designated resource.

02 — Define and organize structure

You can create new resources, define roles, or map relationships between existing data objects.

03 — Manage user and tenant accounts

The MCP lets you provision users or tenants in bulk, keeping your authorization environment up-to-date.

04 — Update role assignments

You can assign permissions to a whole group (a role) or give a specific user a role within their tenant.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to the MCP and enter your Permit.io API Key, optionally including your PDP URL.

02 Tell your agent what you want to do—for example, 'Check if this user can access resource X.'

03 The system runs the check against your defined policies and sends back a definitive answer: permitted or denied.

The bottom line is, you treat complex access control logic like talking to an expert security engineer who lives inside your agent.

Built For

This MCP targets developers and platform engineers whose jobs involve building or auditing secure, multi-tenant applications. If you're tired of writing boilerplate code just to manage basic permissions checks, this is for you.

Platform Engineer

They use the MCP to set up bulk user and tenant creation or define new relationship structures (ReBAC) so that application services can trust the access layer.

Security Architect

They audit existing policies conversationally, using tools like authzen_bulk_evaluations to verify if all user roles adhere to least-privilege principles across a large number of resources.

Backend Developer

They test complex authorization logic quickly during development cycles, running checks like check_permission without leaving their IDE or writing temporary API calls.

What Changes When You Connect

01 You gain instant policy evaluation via check_permission, allowing your agent to answer complex questions like 'Can X do Y?' in real-time, eliminating the need for custom permission microservices.

02 The MCP manages full authorization schema definition. You can use create_resource and create_role to build out entirely new protected features just by defining their rules, not writing code.

03 Managing user data is easy: Use bulk_create_users or bulk_create_tenants to provision thousands of accounts in a single conversational step, drastically cutting setup time for large deployments.

04 Complex relationships are handled with tools like create_relation and bulk_relationship_tuples. You can model ownership and hierarchical permissions (ReBAC) directly through your agent's prompts.

05 The system supports standardized AuthZen evaluation tools. This means the policies you define today will work reliably, even if you switch underlying access control technologies later on.

Real-World Applications

Auditing a new client portal

A security engineer needs to ensure that only premium users can view the 'advanced analytics' resource. Instead of manually checking database tables, they prompt their agent: 'Check if any user without the 'premium_client' role can access the advanced analytics.' The agent uses check_permission and returns a definitive audit report.

Onboarding a massive client base

A platform team needs to set up 5,000 new tenants and assign them default 'read-only' roles. They use bulk_create_tenants followed by bulk_assign_roles, automating the setup process that would normally take days of manual scripting.

Implementing ownership rules

A developer is building a document management system where only the creator should be able to delete a file. They use create_relation and then check_permission to enforce this complex, resource-specific rule set without writing custom database triggers.

Testing role changes quickly

A product manager wants to see if giving 'junior' users the 'project:read' permission breaks anything. They use assign_permissions_to_role and then run authzen_bulk_evaluations on test accounts before committing any code.

Patterns to Avoid

Hardcoding permissions in service logic

✗ AVOID

A developer writes `if user.is_admin or resource.owner == user.id:` inside the core business logic of every single endpoint, leading to massive code duplication and maintenance nightmares.

✓ INSTEAD

Instead, use check_permission. Define the ownership rule once using create_relation, then let your agent call check_permission at the entry point for all protected endpoints.

Managing users via database scripts

✗ AVOID

Having to run complex SQL scripts every time a new client signs up or an admin needs to update user roles across multiple tables.

✓ INSTEAD

Use bulk_create_users and bulk_assign_roles. The MCP handles the underlying data structure updates, letting you manage identity purely through natural language conversation.

Skipping role definitions

✗ AVOID

Assigning permissions directly to individual users instead of grouping them into roles, which makes auditing impossible and means a policy change requires updating hundreds of records.

✓ INSTEAD

First, use create_role to define the group (e.g., 'Editor'). Then, assign all necessary rights using assign_permissions_to_role.

The Right Fit

Use this MCP if your core problem is governance: determining *who* can do *what*. If you are building a multi-tenant application or an API that needs fine-grained access control (like checking ownership, department rules, or resource type limits), this is the right tool. Don't use it if your primary need is simply data retrieval; for fetching unstructured text or simple records, use a general database connector instead. If you only need to check permissions once and never change them, that might work, but if you need to manage roles, resources, and relationships over time, this MCP provides the full lifecycle management required.

The headache of managing access rights in complex apps

Today, figuring out who can see what usually means digging through a dozen dashboards: checking user roles on one screen, cross-referencing resource ownership on another, and running separate database queries just to validate if the request is even allowed. You end up spending hours debugging authorization failures that should have been simple.

With this MCP, you ditch the dashboard maze. Your agent handles it all in a single prompt: 'Does user X have permission Y for resource Z?' The system returns an instant, definitive yes or no answer, letting you move on to building features instead of policing access.

Permit.io MCP gives you complete control over your authorization layer

You eliminate manual schema updates by using create_resource and create_role to define new protected areas in plain language. You no longer have to wait for a backend engineer to write and deploy the necessary code every time a product feature needs access control.

The result is an authorization layer that evolves at the speed of your product requirements, not the speed of your deployment pipeline.

Permit.io: 18 Tools for Authorization Management

These tools allow you to programmatically manage every aspect of your application's access control, from creating users to evaluating the most complex permissions.

#	TOOL	DESCRIPTION
01	`assign_permissions_to_role`	Attaches specific permissions to an existing group role.
02	`assign_role_to_user`	Gives a user a defined role within their tenant.
03	`authzen_access_evaluation`	Performs standardized checks to validate if a subject has access to an action on a resource.
04	`authzen_action_search`	Searches available actions that can be applied in the authorization schema.
05	`authzen_bulk_evaluations`	Runs multiple access evaluations at once for large-scale policy testing.
06	`authzen_resource_search`	Searches available resources that can be protected by policies.
07	`authzen_subject_search`	Searches existing user subjects for standardized policy checking.
08	`bulk_assign_roles`	Assigns roles to many users in a single batch operation (up to 2000).
09	`bulk_create_tenants`	Creates multiple new isolated tenants at once (up to 2000).
10	`bulk_create_users`	Creates many user accounts in a batch process (up to 3000).
11	`bulk_relationship_tuples`	Adds multiple relationship links between resources simultaneously (up to 1000).
12	`check_permission`	Determines if a user can execute a specific action on a resource.
13	`create_relation`	Establishes a complex relationship link between two resources.
14	`create_relationship_tuple`	Creates a single, specific relationship link between resources.
15	`create_resource`	Adds a brand new type of resource into the system schema for protection.
16	`create_role`	Defines a new group role that can be assigned to users.
17	`create_tenant`	Creates a new isolated organizational environment (tenant).

#	TOOL	DESCRIPTION
18	create_user	Adds a brand new user account into the system.

See It in Action

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

U Check if user 'admin@company.com' is permitted to 'delete' the 'server' resource in tenant 'production'.

I've evaluated the permission. The user 'admin@company.com' is **permitted** to perform the 'delete' action on the 'server' resource in the 'production' tenant.

U Create a new resource 'document' with actions 'read', 'write', and 'share' in project 'my-app' environment 'dev'.

I have successfully created the 'document' resource with the specified actions in your 'dev' environment.

U Assign the permissions 'document:read' and 'document:write' to the 'editor' role in project 'my-app' environment 'dev'.

The 'editor' role has been updated. It now has 'read' and 'write' permissions for the 'document' resource.

Frequently Asked Questions

01 **How does Permit.io MCP handle bulk user creation?**

You use bulk_create_users to add thousands of accounts in a single conversational command, saving massive amounts of manual scripting time for platform teams.

02 **Can I check permissions without writing code using the Permit.io MCP?**

Yes, you prompt your agent with an access query, and it uses check_permission to evaluate the rule against your entire defined policy structure in real-time.

03 **What is ReBAC and how do I set it up with Permit.io MCP?**

ReBAC (Relationship-Based Access Control) handles ownership, meaning access depends on relationships between resources. You use create_relation or bulk_relationship_tuples to define these complex links.

04 **Does the MCP support multiple client types for governance?**

Yes, it works with any MCP-compatible client, letting you govern your permissions whether you're working in VS Code, Cursor, or directly through a terminal agent.

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 `"permitio": { "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

Permit.io 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 Permit.io. 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	Permit.io MCP
Server ID	`019e38d5-d2ec-73fe-9e28-4211ee9fde91`
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/permitio.