MCP SERVER | NO CODE | CLOUD HOSTED

# Pulumi MCP

## Audit, track, and manage your cloud state via conversation.

Pulumi connects your agent directly to your infrastructure state. Use this MCP to list organizations, manage stacks (create, delete), track deployment history, inspect outputs like IPs and URLs, and add tags—all via natural conversation.

A+ Quality Score 98.33/100

infrastructure-as-code | cloud-deployment | automation | stack-management | cloud-native | api-integration

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

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.

# Pulumi MCP

11 tools available
Cloud-hosted on Vinkius

You can connect your agent to Pulumi and take full control of your cloud infrastructure from a chat window. Instead of logging into the console to check status or grab an endpoint URL, you just ask for it. Your AI client acts like a dedicated DevOps engineer right in your chat interface, giving you immediate visibility into everything. Need to know which stacks exist? You can list them instantly. Did the last deployment succeed? Check the history and see exactly what changed. The power of Vinkius makes this possible; your agent accesses this full catalog of infrastructure tools so you don't have to switch context or copy-paste commands anymore.

## Core Capabilities

**01 — View infrastructure structure**

The MCP allows you to list all stacks within an organization, giving you a quick map of your deployed environments.

**02 — Audit deployment history**

You can review the full stack update history for any project, seeing which resources changed and who triggered the deployment.

**03 — Retrieve live resource details**

Get specific information about a stack or organization, such as member lists or key configuration settings.

**04 — Access exported outputs**

Fetch critical values from the latest deployment, including API URLs, IP addresses, and resource IDs needed for other services.

**05 — Manage stack metadata**

Set or list custom tags on stacks (like environment=prod or team=platform) to keep your infrastructure organized and searchable.

## One Click on Vinkius — From Prompt to Execution

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

01 First, subscribe to the Pulumi MCP on Vinkius and enter your Pulumi Access Token.

02 Second, prompt your agent with a natural language request (e.g., 'Show me all dev stacks in the finance project').

03 Third, your AI client executes the necessary tool calls and returns structured data—like stack names or deployment status—directly into the chat.

The bottom line is you get to manage complex cloud operations using simple conversation instead of clicking through multiple dashboards.

## Built For

This MCP is for platform engineers and DevOps specialists who are tired of context switching. If your job involves checking deployment status, auditing resources, or finding a specific endpoint URL after a build, this saves you hours of repetitive clicking.

**DevOps Engineer**

Uses the MCP to audit resource changes and track deployment success/failure patterns without opening the Pulumi console.

**Platform Architect**

Manages stack tags across multiple organizations, ensuring environments are correctly categorized for cost-center tracking or compliance audits.

**Software Developer**

Requests exported endpoints and connection strings from the latest deployment to integrate into local development code.

## What Changes When You Connect

01 Stop jumping between tabs. You can list all stacks or audit deployment history instantly by asking your agent, eliminating manual dashboard checks.

02 Never lose an endpoint URL again. Use the MCP to get_stack_outputs and pull critical IPs, URLs, and connection strings from any successful deployment in one go.

03 Keep track of who did what and when. Reviewing the full stack update history via list_deployments gives you a non-repudiable audit trail for compliance or debugging.

04 Better organization means better governance. You can use set_stack_tag to apply mandatory metadata (like cost centers) across dozens of environments at once.

05 Get context fast. Instead of digging into documentation, simply ask the MCP to get_organization details and immediately understand your infrastructure scope.

## Real-World Applications

**Debugging a failed deployment**

A developer notices an endpoint is down. They ask their agent to list_deployments for the 'api' stack, find the failure version, and then use get_stack_outputs to see what resources were provisioned right before the crash.

**Onboarding a new team member**

A platform architect needs to show a new hire all available environments. They simply ask the agent to list_stacks, which returns every stack name and its last update time without needing manual navigation through project folders.

**Preparing for compliance audit**

The ops engineer must prove that all production stacks are correctly tagged. They use list_stack_tags to confirm tags like 'environment=prod' and then set_stack_tag if any critical tag is missing.

**Finding a database connection string**

A backend developer needs the DB endpoint for staging. Instead of querying the console, they ask the agent to get_stack_outputs on the 'staging' stack and instantly receive the required `db_endpoint`.

# Patterns to Avoid

### Manual Console Checking

✗ AVOID

A user has to log into the Pulumi web console, navigate to the specific project, select the correct stack, and then click 'History' to see a timeline of changes.

✓ INSTEAD

Instead, ask your agent to list_deployments or get_stack for the target environment. The MCP pulls that audit data directly without you ever opening the dedicated web console.

### Copying Outputs

✗ AVOID

After a successful deployment, a user manually navigates to the 'Outputs' tab and copies five different URLs or resource IDs into a spreadsheet.

✓ INSTEAD

Ask your agent to get_stack_outputs. The MCP collects all exported values—URLs, IPs, etc.—and presents them cleanly for immediate use.

### Forgetting Organization Scope

✗ AVOID

A user is unsure if a stack belongs to the 'finance' or 'hr' project and opens multiple console tabs trying to locate it.

✓ INSTEAD

First, run list_stacks. This tool shows every stack in your entire organization, giving you instant visibility into which team owns what.

## The Right Fit

Use this MCP if your job requires programmatic interaction with the state of deployed cloud infrastructure. Specifically, if you need to audit history (list_deployments), manage metadata (set_stack_tag), or retrieve runtime data (get_stack_outputs) without manual clicking, this is for you. Don't use it if your goal is simply writing the initial code; that requires a dedicated IaC editor. Also, don't use it if you just need to list resources in an IDE—that's better handled by type-safe validation tools. This MCP handles the *state* and *history*; it doesn't write the code itself.

## Checking Infrastructure Status is a Time Sink

Right now, checking if your staging environment stack is ready means logging into multiple web consoles. You click on the 'Stacks' tab to see names, then you click on a specific name to check status, and finally, you might have to navigate three different tabs—History, Outputs, Tags—just to gather a single picture of what happened.

With this MCP, your agent does all that work for you. You just ask: 'What's the status of the staging stack?' And the system aggregates the deployment history, current outputs, and resource count into one clean response. It's immediate context.

## Pulumi MCP Gives You Total State Visibility

The manual steps that vanish include opening tabs to check tags, switching windows to view deployment logs, and manually copying endpoints from the output page. These are all repetitive tasks with zero value-add.

Now, your agent combines these checks into one conversation. You get a single, comprehensive answer about resource provisioning and status. It’s not just faster; it changes how you operate.

# Pulumi: 11 Tools for Infrastructure Ops

Use these tools to manage the entire lifecycle of your cloud infrastructure, from listing environments to retrieving live resource details.

| # | TOOL | DESCRIPTION |
|---|---|---|
| 01 | `create_stack` | This tool creates a new, isolated instance of your infrastructure program for testing or staging environments. |
| 02 | `delete_stack` | It permanently deletes an entire stack and all resources associated with it. Use this with caution. |
| 03 | `get_current_user` | This tool verifies your identity by retrieving the GitHub login, email, and avatar URL for API logging purposes. |
| 04 | `get_deployment` | Retrieves specific details about a single deployment using its version number and stack name. |
| 05 | `get_organization` | Fetches high-level information and settings for an entire Pulumi organization by its slug. |
| 06 | `get_stack_outputs` | This tool collects all values your program exported during the last deployment, such as URLs or resource IDs. |
| 07 | `get_stack` | Provides detailed information about a specific stack, including its status and core configuration. |
| 08 | `list_deployments` | You can see the complete history of deployments for a stack, tracking who ran it and if it succeeded or failed. |
| 09 | `list_stack_tags` | Lists all key-value metadata labels currently applied to a specific stack. |
| 10 | `list_stacks` | Retrieves a list of every isolated stack within your Pulumi organization, showing their last update time and resource count. |
| 11 | `set_stack_tag` | This tool applies or updates organizational tags (like setting the environment to 'production') on any given stack. |

# See It in Action

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

**U** Show me all stacks in my organization.

I found 6 stacks in your organization: 'my-infra/dev' (last updated 2h ago, 24 resources), 'my-infra/staging' (last updated yesterday, 24 resources), 'my-infra/prod' (last updated 3 days ago, 24 resources), 'networking/prod' (last updated last week, 8 resources), 'database/dev' and 'database/prod' (both created but never updated).

**U** What was the result of the latest deployment to my-infra/prod?

The latest deployment (v12) to my-infra/prod succeeded 3 days ago. It created 2 resources (new S3 bucket and CloudFront distribution), updated 1 resource (EC2 instance type changed) and left 21 unchanged. The deployment took 4 minutes and was triggered by user alice.

**U** Show me the exported outputs from the prod stack.

The prod stack exports the following outputs: `api_url: https://api.example.com`, `cdn_url: https://cdn.example.com`, `db_endpoint: postgres://db.example.com:5432`, `s3_bucket_arn: arn:aws:s3:::my-prod-bucket` and `vpc_id: vpc-0abc123`. Would you like details on any of these?

# Frequently Asked Questions

01 **How do I check if my Pulumi stack is configured correctly using the Pulumi MCP?**

You use list_stack_tags to see what metadata labels are applied. You can also get_organization details to verify settings at the top level.

02 **Can I find out which user ran a specific deployment on my stack using Pulumi MCP?**

Yes, list_deployments shows the full history for your stack and includes the name of the user who triggered each deployment event.

03 **What is the difference between get_stack and get_organization in the Pulumi MCP?**

get_organization provides details about the top-level container (the whole company setup), while get_stack gives you granular info on one isolated environment within that organization.

04 **Does the Pulumi MCP let me create a new infrastructure stack?**

Yes, the create_stack tool allows your agent to provision a brand-new, isolated instance of your infrastructure program for testing or staging use.

05 **How do I retrieve exposed URLs from my deployed services using Pulumi MCP?**

You call get_stack_outputs. This tool collects all values the code exported during deployment, including IPs and external-facing URLs.

# 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 `"pulumi": { "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

# Pulumi 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 Pulumi. 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 | Pulumi MCP |
| Server ID | `019d8472-4ff7-720c-8caf-a89c8ea7eb7f` |
| 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/pulumi.