MCP SERVER | NO CODE | CLOUD HOSTED

Prometheus MCP

Ask about system health, don't build dashboards.

Prometheus MCP lets you talk to your monitoring system. Instead of building dashboards or running complex PromQL in a terminal, just ask your agent about service health, historical trends, and resource usage. It gives you instant access to time-series data analysis from right inside your chat window.

F Quality Score 3.6/100

prometheus | promql | metrics | observability | monitoring | sre

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

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.

Prometheus MCP

14 tools available
Cloud-hosted on Vinkius

Running an infrastructure check used to mean jumping between Grafana, the command line, and documentation pages just to answer one simple question. Now, connect your Prometheus instance via this MCP, and treat your monitoring stack like a conversation. You simply ask your agent about performance—whether you need to know the average CPU usage over the last hour or if a specific service is currently failing. It handles the complex PromQL required for instant queries and historical range data retrieval automatically.

This connection doesn't just display numbers; it translates raw metrics into actionable insights using natural language. If you're working with other monitoring systems, you'll appreciate this focused approach to observability. By connecting through Vinkius, your agent accesses a dedicated stream of system metrics and configurations, letting you bypass manual dashboard building entirely. You get the power of an SRE or DevOps engineer talking directly to you.

Core Capabilities

01 — Analyze historical performance trends

Retrieve complex metric expressions over specific time windows using PromQL.

02 — Find current system status

Get instant readings of metrics at a single point in time, like checking if a target is currently up or down.

03 — Inspect data structure and labels

Discover all available labels for a metric or retrieve metadata to understand what the units and types are.

04 — Manage time-series data (Admin)

Perform administrative tasks like creating snapshots of current metrics or cleaning up old, deleted data entries.

One Click on Vinkius — From Prompt to Execution

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

01 You subscribe to the MCP and provide your Prometheus server URL and any necessary authentication tokens.

02 Your AI client recognizes the connection and lets you ask questions about system health or performance metrics in plain English.

03 The agent translates your request into the correct PromQL query, executes it against the live data, and presents a clear, conversational answer.

The bottom line is that your monitoring data becomes available through natural language prompts, eliminating the need for manual dashboard construction.

Built For

The ops engineer who's tired of clicking through dashboards at 2 am. This MCP is essential for SRE teams and platform developers needing instant metric access without leaving their primary chat or coding environment.

SRE Engineer

Instantly troubleshoot incidents by running live queries against metrics and checking system configurations without switching tabs.

DevOps Developer

Verify service performance, check resource consumption patterns, and audit infrastructure health directly from their IDE or terminal chat.

Platform Architect

Automate the creation of infrastructure health reports by querying metrics and checking monitoring stack configurations via natural language prompts.

What Changes When You Connect

01 Stop context switching. Instead of navigating to a separate dashboard tool just to check latency, you ask your agent in the chat, and it executes the necessary query instantly. You get answers without leaving your workspace.

02 Understand metrics deeply using `get_metadata`. If you aren't sure what 'http_requests_total' measures or if it's a counter or gauge, run this tool to get clear documentation on units and types.

03 Audit system health with status tools. Use the MCP to check the loaded configuration (`get_status_config`) or review runtime information (`get_status_runtimeinfo`) without needing SSH access to the server.

04 Troubleshoot historical issues using `query_range`. Need to know if CPU usage spiked last Tuesday? Specify a time window and get the full trend data back in plain English.

05 Administer your monitoring stack safely. If you need to clean up old, unneeded metrics or create a specific snapshot of current data, use tools like `clean_tombstones` or `create_snapshot` through simple prompts.

Real-World Applications

Diagnosing a sudden latency spike

The agent needs to know why requests slowed down. The user asks, 'Show me the average request duration over the last 15 minutes.' The MCP executes `query_range`, providing a clear trend graph and identifying the exact time period when performance dropped.

Verifying service readiness during deployment

Before deploying code, the developer asks, 'Is the user authentication endpoint currently returning 200 OK?' The MCP runs `query` to check the current status of that specific metric, confirming system availability immediately.

Investigating resource leak patterns

The team needs to determine if memory usage is slowly creeping up. They ask for a historical trend over three days using `query_range`, allowing the agent to analyze the data and pinpoint potential memory leaks.

Checking monitoring stack integrity

A platform team member needs to verify if the Prometheus configuration has changed. They use the MCP's status tools (like `get_status_config`) to pull the YAML settings and confirm compliance with internal standards.

Patterns to Avoid

Treating it like a dashboard.

✗ AVOID

The user tries to visualize every single metric on one screen, leading to information overload and slow loading times across multiple tabs.

✓ INSTEAD

Use the MCP to ask targeted questions. Instead of viewing everything, use `get_labels` first to understand what data points exist, then follow up with a focused query like `query` for 'up' status.

Manually writing PromQL.

✗ AVOID

The user spends 10 minutes looking up the correct syntax for calculating averages or filtering by labels across documentation sites.

✓ INSTEAD

Describe your need to your agent in plain English. The MCP translates that natural language request into complex PromQL and executes it using `query_range`.

Ignoring admin limitations.

✗ AVOID

A user tries to delete data series but doesn't realize the system requires explicit permission to modify stored metrics.

✓ INSTEAD

The MCP handles permissions. If you need to clean up old data, use `delete_series`, knowing the tool itself validates that administrative access is required.

The Right Fit

Use this MCP if your primary bottleneck is translating a complex question about system metrics into executable queries. You should connect here when you need immediate answers—'What happened?' or 'Is it working right now?' The goal is conversational querying. Don't use this if your job requires deep, interactive data manipulation in a visual editor, like adjusting dashboard panels in Grafana. For purely visualization-based analysis, dedicated BI tools are better. If you only need to know *what* metrics are available

without running queries, then simply reviewing the Prometheus web UI is sufficient; this MCP adds the crucial layer of natural language access and status checks like `get_status_buildinfo` that a basic viewer doesn't provide.

The headache of context switching when troubleshooting outages

Today, finding out why service XYZ is slow means opening the monitoring dashboard, running a query to check CPU usage, then opening the logging tool to see errors, and finally opening a separate terminal just to verify network connectivity. You spend more time clicking tabs and copying API endpoints than actually diagnosing.

With this MCP, you stop jumping between apps. Your agent acts as your expert co-pilot. You simply ask it, 'Check latency on service XYZ over the last hour.' It handles all the necessary querying, data fetching, and status checks for you, delivering a single, clear answer right in the chat.

Prometheus MCP: Direct Metric Access

You eliminate the need to manually recall complex PromQL syntax or track which dashboard panel shows which metric. No more hunting through dozens of dropdown menus just to find a specific time series.

The difference is control and speed. You get instant, conversational access to your most critical operational data—whether checking current status with `query` or reviewing historical trends using `query_range`. It's immediate answers, every time.

Prometheus MCP with 14 Tools

These tools give your agent direct access to every operational function in Prometheus, allowing you to query metrics, analyze trends, and manage data integrity from one place.

#	TOOL	DESCRIPTION
01	`clean_tombstones`	Removes deleted data entries from disk, requiring admin permissions to run.
02	`delete_series`	Deletes specific time series data within a defined range, also requires admin permissions.
03	`get_label_values`	Retrieves every unique value associated with a specified label name.
04	`get_labels`	Lists all available labels attached to the metrics in your environment.
05	`get_metadata`	Pulls detailed metadata about a metric, including its unit and type, scraped from monitored targets.
06	`query_range`	Evaluates a PromQL expression to show how metrics have changed over an extended period of time.
07	`query`	Runs a PromQL query to get the metric value at one specific moment in time.
08	`find_series`	Locates all time series data that match your specified label selectors.
09	`create_snapshot`	Creates a complete snapshot of all current metric data, requiring admin permissions.
10	`get_status_buildinfo`	Retrieves general build information about the Prometheus instance itself.
11	`get_status_config`	Displays the currently loaded YAML configuration settings for the monitoring stack.
12	`get_status_flags`	Shows all configured flag values set within Prometheus.
13	`get_status_runtimeinfo`	Provides general runtime details and operational information about the monitoring service.
14	`get_status_tsdb`	Retrieves cardinality statistics for the Time Series Database (TSDB).

See It in Action

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

U Run an instant query for 'up' to see which targets are currently reachable.

I've executed the `query`. All 5 targets are currently returning a value of 1, meaning they are all 'up' and healthy.

U Show me the average CPU usage for the last 30 minutes using query_range.

I've fetched the range data. The average CPU usage across your nodes peaked at 45% about 10 minutes ago and has since stabilized at 20%.

U What is the metadata for the metric 'http_requests_total'?

Using `get_metadata`, I found that 'http_requests_total' is a Counter metric. It tracks the total number of HTTP requests received, partitioned by status code and method.

Frequently Asked Questions

01 **How do I query Prometheus metrics using the Prometheus MCP?**

You ask your agent a question in plain English. The agent interprets your request, builds the necessary PromQL expression, and runs the query for you.

02 **Can I use the Prometheus MCP to check service uptime?**

Yes, you can. You simply ask the agent about the 'up' metric for a specific target or service to see if it is currently reporting a value of 1 (healthy).

03 **What does `get_metadata` do in the Prometheus MCP?**

`get_metadata` lets you check what a metric actually measures. It tells you things like whether the data is counted, measured, or if it's partitioned by status code.

04 **Is `query_range` different from `query` in this MCP?**

`query` provides a single point-in-time value. You use `query_range` when you need to see how the metric changed over an entire period of time, giving you a trend.

05 **Do I need admin rights for all tools in the Prometheus MCP?**

No. Basic querying and metadata retrieval are standard. However, administrative actions like `create_snapshot` or `delete_series` require elevated permissions.

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 `"prometheus": { "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

Prometheus 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 Prometheus. 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	Prometheus MCP
Server ID	`019e38db-cdb5-720b-9e9b-448800ac7d43`
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/prometheus.