MCP SERVER | NO CODE | CLOUD HOSTED

Laravel Forge MCP

Manage deployments and server states via chat.

Laravel Forge MCP lets you manage entire web application ecosystems directly through your AI agent's chat window. You can list all connected droplets, check internal structures, deploy site scripts safely, and query databases linked to specific domains—all without leaving the conversation.

A+ Quality Score 100/100

server-management | deployment | php | automation | database-management | web-hosting

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/laravel-forge

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.

Laravel Forge MCP

9 tools available
Cloud-hosted on Vinkius

This MCP connects your developer account to an AI agent, turning complex deployment tasks into natural conversations. Instead of jumping between a console, a dashboard, and a database client, you talk to your agent. You tell it what needs fixing or deploying, and it handles the backend work. It checks server statuses, lists connected databases for any given domain, and runs full site deployment scripts—all in one go. This is how modern development ops works: talking through complex processes instead of clicking through menus. With Vinkius, this MCP puts that entire suite of devops tools right at your fingertips, letting you act like a senior engineer without having to be one.

Core Capabilities

01 — Review server infrastructure status

List all connected droplets and inspect the internal structures of your deployed sites.

02 — Initiate site deployments

Send a deployment script to a specific repository site, watching the output as it executes.

03 — Check database configurations

Query active databases associated with any connected domain cluster.

04 — Audit worker processes

Examine the daemon configurations and queue workers running on a tracked site.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to this MCP endpoint in your AI client.

02 Provide your core Laravel Forge API token string for authentication.

03 Ask your agent to execute the required devops tasks through natural conversation.

The bottom line is, you treat complex infrastructure management like a simple chat command.

Built For

This MCP is for experienced technical roles—DevOps Engineers and Lead Developers who get bogged down in manual dashboards. You're the person tired of switching context between SSH, Git, and database clients just to roll out a simple patch.

DevOps Engineer

Runs site deployments or checks worker statuses natively during sprint reviews using chat commands.

Lead Developer

Instantly confirms SSH key metadata and active daemon configurations across multiple environments without logging into the console.

System Administrator

Audits entire physical networks or checks system-level structures by running commands through your AI agent instead of writing shell scripts.

What Changes When You Connect

01 Avoid jumping between multiple dashboards. You can check the status of all connected droplets, list websites, and view SSH keys—all from a single conversational interface using `list_servers`, `list_sites`, and `list_ssh_keys`.

02 Deployment is instant. Instead of manually executing scripts, you simply tell your agent to run a deployment script via `deploy_site`. The agent handles the queue execution and tracks output automatically.

03 Database management becomes conversational. You never have to guess which cluster is live; just ask your agent to list databases using `list_databases` and get the details immediately.

04 Audit worker processes effortlessly. Need to know if a background job is stuck? Use `list_workers` to check queue configurations running on any tracked site, ensuring nothing drops off.

05 System visibility improves dramatically. If you need deep context, your agent can pull detailed data on a specific droplet with `get_server`, or look up the specifics of an exact site layout using `get_site`.

Real-World Applications

Rolling out a major feature to staging

A developer needs to push code changes to the pre-production environment. They prompt their agent: 'Deploy the pending commits directly to site 5210 on server 1205.' The agent executes `deploy_site`, reports success, and confirms the new payload is live.

Investigating a slow background task

The application suddenly slows down because jobs aren't processing. A lead developer asks their agent to check the queue workers using `list_workers`. The agent immediately reports finding two active Queue workers, confirming if they are handling the necessary payloads.

Onboarding a new team member

A sysadmin needs a complete overview of the infrastructure. They ask their agent to list all connected servers using `list_servers`, which provides the master inventory, followed by listing every site on those machines via `list_sites`.

Verifying access for an external vendor

A developer needs to confirm what physical keys are available for a new service connection. They simply ask their agent and it uses `list_ssh_keys` to retrieve the active physical access key metadata.

Patterns to Avoid

Copying commands from documentation

✗ AVOID

Manually reading a guide, copying a complex series of shell commands, and pasting them into a terminal session. This is slow and error-prone.

✓ INSTEAD

Just tell your agent what you want done. For example, instead of typing out the deployment script for a site, just ask the agent to `deploy_site` and let it handle the rest.

Guessing which server has data

✗ AVOID

A developer is unsure if their application database lives on Server A or Server B and spends minutes querying network maps.

✓ INSTEAD

Start by asking the agent to `list_servers` to see your whole inventory. Then, use `list_databases` to query all active databases across the entire connected environment.

Forgetting necessary prerequisites

✗ AVOID

Trying to deploy a site without checking if the underlying infrastructure or worker processes are running correctly.

✓ INSTEAD

Before deploying, always check the health. Run `list_workers` first. If those look good, then proceed with your deployment via `deploy_site`.

The Right Fit

Use this MCP if you need to manage entire web stacks—from databases and worker queues up through live deployments—all without leaving the chat interface. You're looking for conversational control over complex infrastructure operations. Don't use it if your only goal is basic code review or writing simple Python scripts; that's better handled by a pure coding agent. If you just need to look at one specific file, any general-purpose reading tool will work. But if the task involves coordinating multiple layers—like confirming which server hosts a site, checking its associated database, and then deploying new code to it—this MCP is your single source of truth.

The Pain of Context Switching

Today, pushing out an update feels like juggling three different tools. You start in the Git client to pull commits. Then you jump to the dashboard to confirm which server droplet needs updating. After that, you open a separate database tool just to ensure the schema is ready for the new code. It's a constant stream of switching tabs and copying credentials.

With this MCP, the whole process collapses into one chat window. You tell your agent what to do, and it handles the sequence: checking server status, listing connected databases, and running `deploy_site`. You get back confirmation that the job is done, without ever leaving your conversation.

Get Server Data with `get_server`

Manually checking a server's status means logging into multiple portals and digging through different metrics pages to piece together what's wrong. You might find the uptime, but you'll miss critical details about worker configurations or active key lists.

Now, just ask your agent for detailed data on that specific droplet using `get_server`. It pulls the complete metadata in one go. You know exactly what you're working with.

Laravel Forge: 9 Tools for Ops

Use these tools to perform deep devops functions like running site deployments, listing server inventory, or checking database configurations through your agent.

#	TOOL	DESCRIPTION
01	`deploy_site`	Runs a deployment script queue to push updates to an entire repository site.
02	`get_server`	Pulls detailed information on one specific server droplet instance.
03	`get_site`	Retrieves specifics for a particular website layout running on a given server.
04	`list_databases`	Lists all active databases mounted to a Forge server instance.
05	`list_recipes`	Retrieves available custom shell recipes within your organizational team limits.
06	`list_servers`	Gathers the master list of every connected Forge server instance.
07	`list_sites`	Lists all websites that are mounted to a specific server droplet.
08	`list_ssh_keys`	Retrieves active physical access keys attached to the root server.
09	`list_workers`	Gets the queue worker configurations that are currently running on a tracked site.

See It in Action

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

U Display all the cloud server instances operating in the fleet.

Here's what I observed... You have 2 droplets attached: 'Production Web' and 'Staging Loadbalancer'. Should I probe their sites?

U Deploy the pending commits directly to staging site 5210 on server 1205.

Deploying package payload... Success! Forge emitted a validation flag kicking off the repository deployment routine on site 5210.

U Check the active workers running on the production server.

Scanning worker processes... Found 2 active Queue workers on the Production Server handling redis payloads securely.

Frequently Asked Questions

01 **Can I use Laravel Forge MCP to list all my servers?**

Yes, you can. Use the `list_servers` tool to retrieve a master inventory of every connected Forge server instance in one command.

02 **How do I deploy code using the Laravel Forge MCP?**

You execute deployments with the `deploy_site` tool. This runs the script queue and manages the full deployment process for your specific repository site.

03 **What if I need to check my database settings?**

Use `list_databases`. This tool lists all active databases that are mounted onto a specific Forge server, keeping your data sources organized and visible.

04 **Does Laravel Forge MCP help me find SSH keys?**

Yes. The `list_ssh_keys` tool lets you retrieve the list of all active physical access keys attached to your root server for auditing purposes.

05 **Can I check which workers are running on a site?**

You use the `list_workers` tool. It checks and reports on the queue worker configurations currently executing tasks on any tracked site, giving you full visibility into background jobs.

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 `"laravel-forge": { "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

Laravel Forge 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 Laravel Forge. 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	Laravel Forge MCP
Server ID	`019d75c4-c4d2-7077-ad13-08125e219f59`
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/laravel-forge.

© 2026 Vinkius · vinkius.com · support@vinkius.com