MCP SERVER | NO CODE | CLOUD HOSTED

Balena MCP for AI Agents

Manage IoT Edge Device Fleets and Deployments

Balena MCP gives your AI client direct access to manage entire IoT infrastructures and edge device fleets. Check device statuses, deploy environment variable updates, track software releases across multiple applications, and query specific hardware details—all without opening a dashboard.

A+ Quality Score 100/100

fleet-management | edge-computing | device-monitoring | linux-devices | deployment-automation | remote-management

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

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.

Balena MCP

10 tools available
Cloud-hosted on Vinkius

Managing physical devices spread out in the real world is usually a nightmare of dashboards, filters, and tedious clicking. This MCP lets your AI client bypass that pain entirely. You talk to it naturally, and it runs complex commands against your entire BalenaCloud setup. Need to know if all 50 cameras are running v2.115? Ask the agent. Want to tag a specific sensor node with its physical location? Have it do it. This connectivity allows you to monitor device health, manage fleet configurations, and handle deployments directly from your chat interface. Because this MCP lives on Vinkius, you connect once—whether that's through Cursor or Claude—and instantly gain access to the entire catalog of enterprise tools.

Core Capabilities

01 — Querying Device Status and Location

Retrieve comprehensive lists of all devices or specific fleets by applying detailed filters, letting you quickly check the status of hardware units.

02 — Managing Device Metadata

Add or update custom tags and environment variables on individual edge devices to keep your physical inventory organized.

03 — Tracking Software Deployments

Inspect deployment history and check release versions across different applications to confirm that all hardware is running the required software build.

04 — Retrieving OS Images

Query available operating system versions for specific device types and get direct download URLs for rapid prototyping or updates.

05 — Inspecting Account Details

Verify your current user profile, associated organizations, and active API keys to manage your account permissions.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to this MCP on Vinkius and provide your Balena API Key.

02 Your AI client authenticates the connection, giving it read/write access to your cloud infrastructure data.

03 You instruct your agent using natural language (e.g., 'List all devices in the industrial fleet that are currently offline').

The bottom line is: you get instant, programmatic control over complex edge device environments without ever touching a web console.

Built For

This MCP is built for engineers who spend too much time clicking through dashboards. It's essential for DevOps teams automating deployments and IoT Engineers needing real-time, programmatic visibility into hardware fleets.

IoT Engineer

Checks the status or logs of devices in remote locations by having the agent run a targeted query across all active fleets.

DevOps Team Lead

Automates environment variable updates and checks deployment release history for dozens of nodes during CI/CD cycles, eliminating manual dashboard work.

Product Owner

Gets high-level overviews of fleet health across multiple organizations to report status without coordinating with the engineering team.

What Changes When You Connect

01 Eliminate dashboard hopping. Instead of navigating multiple tabs to check device status, you just ask the agent, which runs advanced queries via `list_devices`.

02 Maintain strict hardware organization. Use the MCP to dynamically add metadata tags or environment variables to devices using `create_device_tag` and `create_device_env_var`, keeping your inventory clean.

03 Accelerate development cycles. Quickly find necessary images for prototyping by calling `list_os_versions` and getting direct URLs via `get_os_download_url`.

04 Simplify compliance checks. Periodically verify that all deployed hardware is running the required software version by inspecting releases using `list_releases`.

05 Understand your scope immediately. Use `list_fleets` to see every application group you manage, and `whoami` to confirm which organization context the agent is working in.

Real-World Applications

Emergency Device Audit

A sensor fails in a remote location. Instead of logging into the dashboard, the engineer asks their agent to 'List all devices in the factory floor fleet that are offline.' The agent uses `list_devices` and immediately provides a filtered list with specific UUIDs.

Mass Configuration Update

The team needs to update every device's reporting key. They ask the agent to 'Set the new API key variable for all nodes in the warehouse fleet.' The MCP uses `create_device_env_var` across the entire scope, saving hours of manual updates.

Pre-Deployment Setup

A project requires a specific OS version. The product owner asks their agent, 'What are the available balenaOS versions for my Raspberry Pi 4?' The MCP uses `list_os_versions` and provides the necessary download URLs.

Compliance Check

The team needs proof that all edge nodes run the stable production build. They ask the agent to 'Show me the latest release status for the critical sensor fleet.' The MCP calls `list_releases` and confirms adherence.

Patterns to Avoid

Searching by vague descriptions

✗ AVOID

Asking the AI, 'How are my devices doing?' This is too broad and will result in general status reports that don't pinpoint the root cause or necessary action.

✓ INSTEAD

Be specific. Instead of asking generally, use `list_devices` with targeted filters, such as: 'List all devices where the status equals offline AND belongs to the industrial fleet.' This forces precise data retrieval.

Missing device context

✗ AVOID

Trying to change a setting without knowing which group or application owns the hardware. The agent might fail or update the wrong things.

✓ INSTEAD

Always scope your request first by calling `list_fleets` to identify the target fleet's slug, and then use that identifier when asking the agent to perform actions like creating tags.

Attempting manual data entry

✗ AVOID

Manually copying API keys or configuration variables into a chat window. This is error-prone and bypasses version control.

✓ INSTEAD

Use `list_api_keys` to audit your existing credentials, and then use the agent to programmatically apply new settings using `create_device_env_var` across all necessary nodes.

The Right Fit

Use this MCP if your team needs to manage hundreds or thousands of physical edge devices that live outside a central office. Specifically, you need programmatic access to check device statuses (`list_devices`), update configurations on groups of hardware (`create_device_env_var`), or verify software compliance across fleets (`list_releases`). Don't use it if your primary need is managing abstract cloud resources (like databases or user roles) that aren't tied to physical hardware. For simple, single-user profile checks, `whoami` works, but for complex fleet operations, this MCP is necessary.

Balena MCP for AI Agents: Managing Edge Device Fleet Status

Right now, checking the health of a multi-site IoT deployment means jumping between five different dashboards. You pull up the fleet overview, then drill down into specific device groups to check statuses, and if you need configuration details, you have to copy UUIDs and paste them into another tool. It’s click fatigue writ large.

With this MCP, all that complexity vanishes. Your agent handles the filtering and querying for you. You just ask: 'What's wrong with the cameras in building C?' The MCP uses `list_devices` to run a filtered query, returning only the necessary data points—a list of failures or warnings—directly into your conversation.

Balena MCP for AI Agents: Automating Edge Device Provisioning

Setting up new hardware is a multi-step nightmare. First, you need to know what operating systems are compatible; then you have to find the right download URL; finally, you update every device's environment variables with specific network settings.

This MCP sequences those steps automatically. You can ask it to 'Get the OS images and tag them for the new site.' It orchestrates calling `list_os_versions` followed by tagging actions like `create_device_tag`, providing a complete, automated provisioning sequence.

Balena: 10 Tools for Edge Device Fleet Management

Use these tools to query device data, manage tags and environment variables on hardware, or track software releases across all connected fleets.

#	TOOL	DESCRIPTION
01	list_api_keys	Lists all API keys associated with your Balena account for management and auditing.
02	create_device_env_var	Sets or modifies a specific environment variable on an individual device's operating system.
03	create_device_tag	Applies a custom metadata tag to a physical device for better organization and grouping.
04	list_devices	Queries devices within specific fleets using advanced filtering methods like UUID or status.
05	list_fleets	Lists all application groups (fleets) hosted on your account, allowing you to scope device queries.
06	get_os_download_url	Retrieves the direct download URL for a specific balenaOS image optimized for a given hardware type.
07	list_os_versions	Presents an inventory of all available operating system versions compatible with a chosen device model.
08	list_organizations	Provides a list of distinct organizational units within your Balena account structure.
09	list_releases	Checks the deployment history and available releases for specific application fleets.
10	whoami	Retrieves basic information about your current user profile within BalenaCloud.

See It in Action

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

U Show me which devices in my sensor fleet are running the old OS version.

Device Status Report: Sensor Fleet (20 nodes)

DEVICE ID	CURRENT VERSION	LAST SEEN	STATUS
1a3b-c4d5	v2.108.27	2024-06-01	Online
9e8f-g7h6	v2.108.27	2024-06-01	Online
3c2b-a1d2	**v2.115.18**	2024-06-12	Offline (Needs Update)

Action Required: Device 3c2b-a1d2 needs an OS update to match the current stable release.

U Can you find me a download link for the latest OS image for my RPi4?

I found several versions available for the Raspberry Pi 4 (64-bit). The most recent stable releases are:

- **v2.115.18**: Download URL: `https://balena-download/rpi4-2.115`
- **v2.113.18**: Download URL: `https://balena-download/rpi4-2.113`

You can use the link for v2.115.18 to start your build immediately.

U I need to update the location tag and add a new API key variable to all devices in the warehouse.

Update successful!

- ✅ Tag 'Location: Warehouse-B' applied to 15/20 nodes.
- 🔑 Environment Variable `REPORTING_KEY` successfully set for all scoped devices. The previous key has been overwritten with the new value.

Frequently Asked Questions

01 **How do I check if my remote IoT hardware is running the right software version using Balena MCP?**

You can confirm compliance by asking your agent to list releases for a specific fleet. This tells you exactly what versions are available and which ones should be deployed, helping you verify that all nodes match the required build.

02 **Can Balena MCP help me organize my physical device inventory?**

Yes. You can use the agent to add custom tags or environment variables to specific devices, acting like a programmatic way to label and categorize your hardware without logging into the dashboard.

03 **What if I need an OS image for a new project? Can Balena MCP help me get it?**

The agent can list all operating system versions compatible with your specific device type, giving you direct download URLs right in the chat. This speeds up prototyping dramatically.

04 **Does Balena MCP let me update settings on many devices at once?**

Absolutely. You tell the agent which group of devices needs updating—like setting a new reporting key or adding a location tag—and it applies that change across all selected nodes in one go.

05 **Is Balena MCP only for checking status, or can I actually make changes?**

It does both. You can read the status of devices using advanced querying, but you also have write permissions to modify tags and environment variables on live hardware.

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 `"balena": { "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

Balena 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 Balena. 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	Balena MCP
Server ID	`019e386a-e88b-7363-9813-1a904d2c0977`
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/balena.