MCP SERVER | NO CODE | CLOUD HOSTED

LinearB MCP

Query metrics and track deployments via your AI client.

LinearB connects your AI agent directly to your software delivery pipeline, automating engineering intelligence and DORA reporting. You can query complex metrics like cycle time or coding duration across multiple teams. It also allows you to report new deployments using Git references and log incidents to accurately calculate MTTR and Change Failure Rate.

A+ Quality Score 100/100

dora-metrics | engineering-intelligence | deployment-tracking | incident-management | cycle-time | devops

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

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.

LinearB MCP

7 tools available
Cloud-hosted on Vinkius

Managing software health used to mean opening a dozen dashboards—one for deployments, one for team capacity, another for incident logs. Now, your agent handles the heavy lifting. This MCP lets your AI client access all those critical engineering metrics directly. You can ask natural language questions like, 'What was our average cycle time last month?' and get an immediate answer detailing coding time versus pickup time.

Need to log a new release? Your agent records that deployment using the Git reference, keeping your records current without you lifting a finger. If something breaks, reporting a new incident is just a command away. Because this capability lives in the Vinkius catalog, connecting it takes minutes. You get a single source of truth for performance data—the whole picture needed to audit organizational health and track deployments.

Core Capabilities

01 — Query Team Performance Metrics

Ask about complex metrics like average cycle time, coding duration, or pickup time across specific teams.

02 — Log Software Releases

Inform the system about a new software deployment by providing a Git reference (SHA or tag).

03 — Track Service Outages

Record and list engineering incidents, which is necessary for calculating Mean Time To Recover (MTTR) and Change Failure Rate.

04 — Map Technical Assets

View a comprehensive list of all connected repositories and defined engineering teams in the system.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to this MCP and provide your LinearB Public API Key.

02 Authorize your AI client to connect to the Vinkius catalog.

03 Use natural language commands within your agent to query metrics, list deployments, or report incidents.

The bottom line is that you talk to your agent like talking to a coworker; it handles all the API calls and data formatting for you.

Built For

This MCP is built for technical leaders and operational engineers. If you're an Ops Engineer tired of clicking through five different dashboards at 2 AM, or a CTO who needs to audit performance without opening the web app, this is for you.

DevOps Engineer

Automates logging deployments and incidents directly from CI/CD tools or IDEs so metrics stay accurate.

Engineering Manager

Gets a quick readout on team cycle times or overall delivery health using simple natural language commands.

CTO / VP of Engineering

Runs ad-hoc audits on organizational performance and DORA metrics without needing to navigate the full dashboard interface.

What Changes When You Connect

01 You can query complex engineering data, like average cycle time across teams, simply by asking a question. This is done using the `query_software_metrics` tool, eliminating manual dashboard report generation.

02 Need to keep deployment records current? You use `record_new_deployment` to tell LinearB about a new release using its Git reference, ensuring your DORA metrics never suffer from stale data.

03 When an outage happens, you immediately call the `record_new_incident` tool. This action logs the event and is critical for calculating accurate Mean Time To Recover (MTTR).

04 You don't need to manually map out who owns what. You can use `list_engineering_teams` and `list_connected_repos` to quickly understand your entire technical structure.

05 The ability to list both deployments (`list_software_deployments`) and incidents (`list_software_incidents`) in one flow gives you a full, chronological picture of system stability.

Real-World Applications

Auditing Team Performance

A manager needs to know if the 'Backend' team is falling behind. Instead of opening the dashboard and clicking filters for time ranges and metric types, they ask their agent: 'What was the average cycle_time for Backend over the last 30 days?' The agent uses `query_software_metrics` and immediately reports the data points.

Logging a Critical Outage

The primary service goes down. An engineer opens their chat client and tells their agent: 'Record an incident for OpsGenie starting now.' The agent uses `record_new_incident` so that the MTTR calculation starts instantly, without manual data entry.

Tracking a Major Release

The CI/CD pipeline finishes and pushes version v2.1.0 for repo 456. Instead of navigating to LinearB's UI, the DevOps engineer asks their agent: 'Report deployment v2.1.0 for repo 456.' The agent uses `record_new_deployment` instantly.

Understanding System Scope

A new team member joins and needs to understand the overall architecture. They ask their AI client: 'List all connected repositories and teams.' This triggers both `list_connected_repos` and `list_engineering_teams`, giving them a complete map.

Patterns to Avoid

Copy-pasting metrics

✗ AVOID

The user manually opens the LinearB dashboard, copies cycle time data for Q3 into a spreadsheet, and pastes it into an email summary.

✓ INSTEAD

Just ask your agent. Use `query_software_metrics` to request the specific range and metric you need directly in natural language. The metrics come straight through.

Forgetting deployment details

✗ AVOID

A developer finishes a feature branch, but forgets to manually update the system with the Git SHA, leading to gaps in DORA reporting.

✓ INSTEAD

Make it part of your process. Use `record_new_deployment` immediately after merging. This ensures every release is logged automatically.

Manual incident tracking

✗ AVOID

When an outage happens, the team starts a thread in Slack and manually updates spreadsheets with timestamps and affected providers.

✓ INSTEAD

Use `record_new_incident` immediately. This action logs the necessary start time against your provider ID, starting the clock on MTTR.

The Right Fit

Use this MCP if your core problem is synthesizing performance data from multiple sources into actionable insights, especially regarding software delivery metrics (DORA). It excels at automating logging—like recording deployments or incidents—and complex querying. Don't use this if you simply need to view a single dashboard chart; the agent will still pull that data for you via `query_software_metrics`. Conversely, don't expect it to handle ticket prioritization or communicate with external ticketing systems (like Jira). It lives purely in the realm of engineering performance tracking and logging. If your goal is 'What happened?' or 'How fast?', this MCP works. If your goal is 'Fix this problem,' you need a different tool.

Tracking software health usually means switching between five different tabs.

Right now, tracking delivery performance involves constant context switching. You check the deployment dashboard for recent releases; then you switch to the team management view to see who owns which repo. Next, you open a separate tab just to list out all current incidents and manually calculate the MTTR by looking at timestamps. It's a tedious process of clicking through dashboards and copy-pasting data into summary documents.

With this MCP, that whole routine vanishes. You tell your agent what you need—whether it’s querying cycle time or listing team structures. The agent handles the clicks across all those services behind the scenes. You get one clean answer: a comprehensive view of your engineering performance without touching a dashboard.

LinearB MCP gives you immediate control over incident and deployment tracking.

The biggest manual steps that go away are the need to manually log releases or track incidents. You no longer have to stop your workflow to navigate a UI just to record a new deployment; you simply tell your agent to use `record_new_deployment`. Similarly, logging an outage is as simple as using the `record_new_incident` tool.

It's not about viewing data anymore. It's about controlling the inputs that generate the metrics. You own the historical record of deployments and incidents instantly.

LinearB: 7 Tools for Delivery Intelligence

These tools allow your agent to perform specific actions within LinearB, such as listing teams or recording a new software incident. Use them to gather structured data instantly.

#	TOOL	DESCRIPTION
01	`record_new_deployment`	Reports a new software release into LinearB using a specific repository ID and Git reference.
02	`record_new_incident`	Creates an incident record for service outages, requiring the provider ID and time it started.
03	`list_software_deployments`	Retrieves a list of all recent software deployments recorded in LinearB.
04	`list_software_incidents`	Fetches a listing of engineering incidents to track service disruptions.
05	`query_software_metrics`	Queries detailed software engineering metrics, allowing you to specify what data points and time windows you need.
06	`list_connected_repos`	Retrieves a list of all repositories that have been connected and monitored by LinearB.
07	`list_engineering_teams`	Lists every team defined within the LinearB system for scope management.

See It in Action

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

U Query the average cycle_time for the last 30 days for team 'Backend'.

I've retrieved the metrics for team 'Backend'. The average cycle_time over the last 30 days is 3.5 days, with coding_time at 1.2 days and pickup_time at 0.8 days.

U Record a new deployment for repo ID '123' with Git ref 'v1.2.0'.

Successfully informed LinearB of the new deployment. The Git ref 'v1.2.0' for repository 123 has been recorded, and cycle times for the included PRs will be updated.

U Report a new incident starting now for provider 'OpsGenie'.

I've reported the incident to LinearB. It has been associated with provider 'OpsGenie' starting at [timestamp]. This will be used to calculate your MTTR.

Frequently Asked Questions

01 **How do I query cycle time using LinearB MCP?**

You ask your agent directly, specifying the metric and time frame. The `query_software_metrics` tool handles the complex data request, giving you immediate insight into coding time versus pickup time.

02 **Can I use LinearB MCP to track deployments from CI/CD?**

Yes. You can use your agent to trigger `record_new_deployment` by passing the Git SHA or tag, ensuring that every release is logged automatically for accurate reporting.

03 What happens when I list engineering teams with LinearB MCP?

The `list_engineering_teams` tool fetches a clean list of all defined teams in the system. This helps you map technical IDs to specific organizational units for better reporting.

04 Does LinearB MCP help calculate MTTR?

Yes, by using `record_new_incident`, your agent logs the start time of an incident against a provider. This critical data point allows you to accurately track and calculate Mean Time To Recover (MTTR).

05 Which repositories can I query with LinearB MCP?

First, use `list_connected_repos` to see all available sources. Then, your agent uses those IDs when calling tools like `query_software_metrics` or `record_new_deployment`.

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 `"linearb": { "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

LinearB 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 LinearB. 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	LinearB MCP
Server ID	`019d75c7-9e9e-7357-afef-75d61374aa2c`
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/linearb.