MCP SERVER | NO CODE | CLOUD HOSTED

DeepSource MCP for AI Agents

Monitor Code Quality and Security Vulnerabilities

DeepSource lets your AI client analyze code quality, find security flaws, and track complex metrics across repositories using natural language prompts. Instead of clicking through dashboards to check for bugs or high cyclomatic complexity, you just ask your agent. It pulls live data on everything from dependency vulnerabilities (CVEs) to overall repository health scores (A-F), giving instant reports without leaving your IDE.

A+ Quality Score 98.33/100

code-review | static-analysis | vulnerability-scanning | code-quality | automated-testing | security-linting

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

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.

DeepSource MCP

14 tools available
Cloud-hosted on Vinkius

Stop navigating complex web dashboards just to grade a codebase. DeepSource connects code quality analysis and security scanning directly to your AI client, letting you review massive amounts of technical debt using simple conversation.

Your agent acts as an expert developer or dedicated security reviewer for your entire repository history. Need to know if the latest pull request introduced high cyclomatic complexity? Just ask. Are there any critical CVEs in the dependencies that need immediate patching? Your AI client pulls those details instantly.

It gives you a comprehensive view of code smells, anti-patterns, and deep metrics like test coverage percentages—all while remaining inside your workflow. When you subscribe through Vinkius, you connect once and gain access to this powerful analysis engine from any compatible agent, making DeepSource an indispensable part of the modern development stack.

Core Capabilities

01 — Assess overall code health grade

Get a single, high-level report card for the repository that summarizes its overall quality status and identifies trends.

02 — Identify specific bugs and smells

List detailed code issues, such as anti-patterns or unused variables, complete with file paths and line numbers.

03 — Scan for security vulnerabilities

Find dependency flaws by listing known CVE IDs, CVSS scores, and determining if the flaw is reachable in your code.

04 — Query detailed code metrics

Retrieve specific quantitative data points like maintainability index, cyclomatic complexity, and test coverage percentages for comparison.

05 — Review recent analysis history

View a log of all past code analyses, including the branch name, analyzer used, and whether the run succeeded or failed.

06 — Manage repository status

Control which repositories are actively monitored by DeepSource, allowing you to pause analysis or update default branches as needed.

One Click on Vinkius — From Prompt to Execution

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

01 Connect your AI client to this MCP and enter your DeepSource Personal Access Token.

02 Ask your agent a specific question about the codebase, like 'What are the high-risk dependency vulnerabilities in the main branch?'

03 The MCP executes the necessary checks, pulls the data, and presents a clear summary of findings directly back through your conversation.

The bottom line is that you get deep code analysis reports without ever leaving your chat interface or opening the DeepSource web dashboard.

Built For

This MCP is for engineering teams who are tired of context switching. It helps Developers fix problems before merging, Security Teams prioritize CVEs immediately, and Engineering Managers get instant status checks across multiple repos.

Developer

You check code issues or metrics directly from your IDE to find bugs and quality problems right where you're coding, fixing them before they hit the main branch.

Security Engineer

You monitor dependency vulnerabilities, using CVE details and CVSS scores to prioritize remediation efforts based on whether the flaw is actually reachable in your application logic.

Engineering Manager

You instantly review code quality grades or analysis status across multiple large repositories without having to manually open and sort through dozens of dashboards.

What Changes When You Connect

01 Review complex metrics like cyclomatic complexity or maintainability index directly from your agent, without opening the DeepSource dashboard.

02 Immediately identify code issues, such as unused imports or anti-patterns, using list_issues to pinpoint exact lines of problematic code.

03 Prioritize security fixes by listing vulnerabilities with CVE IDs and CVSS scores, allowing you to focus remediation efforts on high-reachability flaws.

04 Get an instant overall health grade via get_report_card, giving stakeholders a single, actionable metric for repository quality at a glance.

05 Understand your dependencies' risk surface area by using list_sca_targets to see exactly which manifest files are being scanned for supply chain threats.

Real-World Applications

A security team needs an audit report on all critical flaws.

The agent runs list_vulnerabilities and filters the results, presenting a clear table of every CRITICAL CVE. The engineer then uses get_vulnerability to deep-dive into one specific issue, confirming the fix path before creating tickets.

A developer needs to know why their local code smells bad.

The developer asks the agent to check for issues and gets a list of problems. They then use get_repository_metrics to check the cyclomatic complexity score, confirming that a specific function is too complex and needs refactoring.

An engineering manager wants an instant health check across five repos.

The manager prompts for all report cards. The agent uses get_report_card multiple times in quick succession, providing a summary table of grades (A-F) and identifying the top three repositories needing immediate attention.

DevOps needs to adjust repository monitoring after a team migration.

The DevOps lead asks the agent to update the default branch using update_default_branch, ensuring that all future analyses run against the correct source code base (e.g., moving from 'master' to 'main').

Patterns to Avoid

Manually checking every single metric.

✗ AVOID

A team member opens the DeepSource dashboard, clicks on metrics, copies the cyclomatic complexity score, then has to open a separate tab to check test coverage. This is slow and error-prone.

✓ INSTEAD

Ask your agent to run get_repository_metrics; it pulls both the cyclomatic complexity and line coverage data into one conversational summary.

Forgetting to check dependency reachability.

✗ AVOID

A security team sees a HIGH CVSS score for an old library, flags it as critical, but doesn't know if the code actually uses that specific function. They waste time investigating a non-issue.

✓ INSTEAD

Use list_vulnerabilities and then get_vulnerability to check the 'reachability status,' confirming if the flaw is active in your current codebase.

Using outdated analysis tokens.

✗ AVOID

A new developer runs an analysis but it fails with an authentication error because the token was never rotated or regenerated, stopping all work until a manual fix.

✓ INSTEAD

Run regenerate_dsn first. This action invalidates the old token and provides a fresh one you can use to keep continuous monitoring running smoothly.

The Right Fit

Use this MCP if your primary need is converting deep, technical reports (metrics, CVEs, code smells) into conversational answers for your AI agent. This works best when you need quick comparisons or summaries of multiple data points; for example, comparing the cyclomatic complexity score against the overall report card grade.

Don't use this if you just need simple status checks on a single repository that don't involve deep code analysis. If you only need to know if a repo exists, using get_repository is fine, but it won't give you any quality insights. Also, if your security needs are limited to basic license checking and don't require CVE or CVSS scoring, then a simpler dependency checker tool might suffice instead.

DeepSource MCP for AI Agents: Addressing Code Smell and Technical Debt

Today, catching code smells means running a static analysis tool, getting an output file, then opening another system to manually cross-reference those issues against the repository's current state. You spend time translating technical warnings into actionable development tasks.

With this MCP, you just ask your agent: 'What are the top 5 code smells in the payments module?' The agent reads the data and gives you a prioritized list of anti-patterns right away. It makes finding technical debt instant.

DeepSource MCP for AI Agents: Managing Dependency Vulnerability Risk

Manually managing security risk involves maintaining spreadsheets that track every dependency version and cross-referencing those against public CVE databases. This process is slow, reactive, and often misses the 'reachability' factor.

This MCP allows you to ask for a vulnerability report by listing all risks. It doesn't just list flaws; it tells you if the flaw is reachable in your code, letting your team focus only on the high-impact, active threats.

14 Tools in the DeepSource MCP for Code Quality Metrics

Use these tools through your agent to manage repositories, list specific issues, or retrieve deep code quality metrics like coverage percentages and complexity scores.

#	TOOL	DESCRIPTION
01	activate_repository	Turns on deep source analysis for a repository that was previously paused or inactive, allowing code quality monitoring to start again.
02	deactivate_repository	Stops all new analyses for a given repository, useful when archiving or temporarily pausing billing for a specific project.
03	get_report_card	Provides an immediate, high-level grade (A-F) showing the overall code quality health of a specified repository.
04	get_repository_metrics	Retrieves specific quantitative data points for a repo, such as line coverage or cyclomatic complexity index.
05	get_repository	Inspects the basic configuration details of a repository to confirm its identity before running any analysis.
06	get_test_coverage	Shows the current test coverage percentage and checks it against configured quality thresholds for the codebase.
07	get_viewer	Verifies that your API token is working correctly and retrieves basic user profile information from DeepSource.
08	get_vulnerability	Deep dives into a single dependency flaw, providing detailed information about its impact and potential fix steps.
09	list_analysis_runs	Lists the most recent code analysis attempts for a repo, showing which analyzer ran (e.g., Python or JavaScript) and if it passed or failed.
10	list_issues	Identifies specific code smells, anti-patterns, or potential bugs across the codebase, noting exact file paths and line numbers for every issue.
11	list_sca_targets	Lists all dependency manifest files (like npm or pip) that DeepSource is currently scanning to check for supply chain security risks.
12	list_vulnerabilities	Generates a list of security flaws in your dependencies, detailing the severity, CVE ID, and fixability status.
13	regenerate_dsn	Creates a brand new Data Source Name token for the repository to ensure continuous analysis runs are authenticated securely.

#	TOOL	DESCRIPTION
14	`update_default_branch`	Changes the primary branch name that DeepSource uses as the default source of truth for code quality monitoring.

See It in Action

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

U Check test coverage for 'backend-api' and also tell me the latest analysis runs.

Test Coverage: 78.5% (Line Coverage - LCV). Threshold configured: 80%. You need to write more tests.

Recent Analysis Runs (last 5):
1. Run #abc123 — Status: PASSED — Branch: main — Analyzer: PYTHON — 2 hours ago
2. Run #ghi789 — Status: FAILED — Branch: feature/payments — Analyzer: PYTHON — Yesterday — Error: timeout

U What are the biggest security risks in our web frontend dependencies?

I found 5 dependency vulnerabilities in 'web-frontend'. The most critical is **CVE-2024-1234** (CRITICAL, CVSS: 9.8) affecting `lodash`. This flaw is highly reachable and needs upgrading to v4.17.21 immediately.

There are three other high/medium risks that should be reviewed this sprint.

U What is the cyclomatic complexity and maintainability index for the user service repo?

Code Metrics Report:
• Maintainability Index: 85/100. (Good)
• Cyclomatic Complexity: Average of 6.2. (Acceptable, but watch out).

This suggests the code is generally clean, though specific functions might need refactoring to keep complexity low.

Frequently Asked Questions

01 **How do I get a DeepSource Personal Access Token and where do I find it?**

Log in to your DeepSource account, go to **Account Settings** → **Personal Access Tokens**, and click **Create New Token**. Give it a descriptive name (e.g., 'Vinkius MCP') and copy the token immediately — it won't be shown again. Paste this token into the API key field below. The token is used as a Bearer token in the Authorization header for all GraphQL requests to `https://api.deepsource.com/graphql/`.

02 **What types of code issues can DeepSource detect and how are they categorized?**

DeepSource detects various code quality issues including code smells, anti-patterns, performance issues, security vulnerabilities, and bugs. Issues are categorized by severity (CRITICAL, HIGH, MEDIUM, LOW) and by analyzer type (e.g., PYTHON for Python issues, JS-A1 for JavaScript anti-patterns, GO for Go issues). Each issue includes a shortcode, title, category, and file locations with line numbers. You can filter issues by analyzer short code when querying repositories.

03 **How does DeepSource detect dependency vulnerabilities and what information is provided?**

DeepSource uses Supply Chain Analysis (SCA) to scan dependency manifest files (package.json, requirements.txt, Gemfile, etc.) for known vulnerabilities. Each vulnerability includes: CVE ID, CVSS score (0-10), severity level, description, affected package name and version, ecosystem (npm, pip, etc.), reachability status (whether the vulnerable code is actually called), and fixability (whether a fix version is available). This helps prioritize which vulnerabilities to address first based on real risk rather than just theoretical severity.

04 **What is the API rate limit and how many requests can I make per hour?**

DeepSource enforces a rate limit of 5,000 requests per hour per user account. This limit covers both read (queries) and write (mutations) operations. If you exceed this limit, the API will return HTTP 429 (Too Many Requests). For most code review and monitoring workflows, this limit is more than sufficient. If you need higher limits for large-scale analysis, contact DeepSource support.

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 `"deepsource": { "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

DeepSource 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 DeepSource. 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	DeepSource MCP
Server ID	`019d7583-6eb9-7012-842b-8929580a1728`
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/deepsource.