MCP SERVER | NO CODE | CLOUD HOSTED

Codecov MCP for AI Agents

Monitor code coverage and track commit metrics in your repositories

Codecov brings all your test coverage data and engineering metrics into natural conversation. Your AI client can check build health by retrieving aggregate totals for specific commits, list repository details across an organization, or audit complex coverage reports using simple queries.

F Quality Score 3.6/100

test-coverage | code-quality | ci-cd-pipeline | software-testing | coverage-reports | automated-testing

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

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.

Codecov MCP

8 tools available
Cloud-hosted on Vinkius

Managing software quality often means staring at dashboards filled with graphs and percentages. This MCP changes that. You connect your Codecov account to any AI agent, and suddenly you can ask questions about your code quality the way you talk to a coworker. Instead of navigating multiple tabs—checking coverage for one repository, then switching to look up another commit's totals, and finally trying to map out a complex report structure—you just ask. Your agent handles all that complexity in plain language.

This setup lets developers monitor everything from branch-specific coverage metrics to the overall health of an entire codebase without ever leaving their chat window. It’s about getting immediate answers on build status and test completeness, letting you focus on writing code instead of clicking through reports across multiple repositories. You'll find that Vinkius makes connecting these deep technical workflows simple for any MCP-compatible client.

Core Capabilities

01 — List all repositories

Gets a list of every repository associated with an owner, along with its current coverage percentage.

02 — Check build health by commit SHA

Retrieves the total test coverage metrics for any specific code commit hash you provide.

03 — Analyze report structure

Generates a detailed, hierarchical view of how your project's coverage reports match its file system layout.

04 — Compare branches and flags

Allows you to monitor and compare test coverage across multiple development branches or custom-defined monitoring flags.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to this MCP and provide your unique Codecov Global API Token, which you get from your settings dashboard.

02 Connect the token to your preferred AI client (like Cursor or Claude).

03 Ask your agent a question like, 'What was the coverage percentage for the last commit in the core-api repo?' and get an instant answer.

The bottom line is that you treat complex code metrics like simple chat requests.

Built For

This MCP is built for technical roles who spend too much time switching between dashboards to check build health or test coverage. It saves the DevOps engineer from manual report aggregation and gives QA teams an instant way to audit code quality.

Software Engineer

Using this MCP, you can quickly compare current feature branch coverage against main, auditing specific file reports using natural language.

DevOps Engineer

When a build fails or passes, you don't have to open the dashboard. You ask your agent for commit coverage totals and verify the status straight from chat.

Engineering Manager

You can review team-wide trends—like which repositories are lagging on test coverage or what changes happened across different branches—without logging into Codecov's dashboard.

What Changes When You Connect

01 Instead of manually checking multiple tabs, you can ask your agent to list all associated repositories and their current coverage percentages instantly.

02 Need proof of build health? Use the `get_commit_coverage_totals` function to check aggregate test totals for any specific commit SHA without opening a dashboard.

03 Understand complex code structure by asking for the full report tree. The agent uses `get_coverage_report_tree` to map out coverage against your project's file system.

04 Comparing branches is easy. You can use `list_repository_branches` and then check metrics across them to see which development line needs more testing.

05 Get a clean view of all projects by using the function that lists Codecov repositories, giving you oversight for entire organizations.

Real-World Applications

The CI/CD pipeline passed, but was coverage adequate?

A DevOps engineer asks their agent: 'What's the overall coverage for this commit?' The agent uses `get_commit_coverage_totals` to confirm that SHA 'abc1234' achieved an 85.4% total coverage, confirming the build is ready to merge.

The team needs a quick audit of all projects.

An Engineering Manager asks: 'Show me every repository in the organization and their current coverage.' The agent runs `list_codecov_repositories`, providing an immediate, high-level health check across the entire portfolio.

Need to see how different features impact test quality.

A Software Engineer asks: 'Compare the coverage of the staging branch versus the main branch.' The agent uses `list_repository_branches` and then retrieves detailed comparisons, helping them prioritize testing efforts.

Diagnosing a low-coverage area in the codebase.

A QA specialist asks: 'Where is coverage lowest in our utility folder?' The agent uses `get_coverage_report_tree` to return a file-system map, immediately pointing the user toward the weak spot.

Patterns to Avoid

Treating Codecov like a simple list

✗ AVOID

A developer only asks for 'all repos' and ignores the context. They get a list but don't know if any are below threshold.

✓ INSTEAD

Always follow up by asking, 'For the core-api repo, show me the coverage details,' using `get_repository_coverage_details` to get actionable numbers.

Checking only the latest commit

✗ AVOID

A manager runs a single check and assumes the current build is perfect without checking historical context.

✓ INSTEAD

Use `list_repository_commits` first, then pick an older SHA to run `get_commit_coverage_totals`, giving you a true measure of coverage trend over time.

Overlooking branch differences

✗ AVOID

A team member merges code without realizing the feature branch had much lower test coverage than main.

✓ INSTEAD

Always run `list_repository_branches` first. Then, ask for comparisons between specific branches to understand the distribution and risk.

The Right Fit

Use this MCP if your primary need is to transform complex, graph-based code quality data into simple, conversational questions. You should use it when you need immediate insights like 'What was the coverage for my last commit?' or 'Which repo needs attention?'. Don't use it if you just want to run a visual report comparison that requires filtering by date ranges outside of available tools. If your goal is merely to manage API tokens or user accounts, those are separate identity management tools. This MCP is purely for deep code metrics and coverage reporting.

Codecov MCP: Auditing Software Quality Metrics via Conversational AI

Right now, auditing test coverage means clicking into the Codecov dashboard. You have to select a repository, then manually navigate through branches, find specific commit SHAs, and finally, click on the report tree view just to understand where your tests are failing or missing. It’s a multi-step process that kills momentum.

With this MCP, you simply ask: 'What was the coverage for the main branch after the last merge?' Your agent pulls all those metrics—the commit totals, the repository details, and the full report tree—and spits out one clean answer. You get immediate, conversational answers on your build status.

Codecov MCP: Tracking Code Quality Across Multiple Development Branches

Manually checking coverage across different development branches is a major pain point. You have to remember which branch you checked last, and then repeat the entire process for the next one —a tedious cycle of context switching.

Now, just ask your agent to compare coverage between two specific branches using `list_repository_branches` as a starting point. The comparison is instant, allowing you to spot coverage gaps or regressions without opening another browser tab.

Codecov: 8 Tools for Analyzing Repository Coverage Reports

Use these tools to list repositories, check branch status, retrieve report trees, and verify precise coverage metrics from any commit SHA.

#	TOOL	DESCRIPTION
01	`get_commit_coverage_totals`	Pulls the combined test coverage metrics for a specific commit hash.
02	`get_my_codecov_profile`	Retrieves metadata about your Codecov account and user profile.
03	`get_repository_coverage_details`	Gathers detailed coverage information for a single, specified repository.
04	`get_coverage_report_tree`	Builds and provides a hierarchical view that matches your project's folder structure.
05	`list_repository_branches`	Lists all development branches tracked by Codecov for an organization.
06	`list_repository_commits`	Shows a list of recent commits along with their associated coverage status.
07	`list_coverage_flags`	Retrieves all custom flags used to categorize and monitor different coverage metrics.
08	`list_codecov_repositories`	Lists every repository linked under a specified owner or organization.

See It in Action

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

U What's the overall test coverage for all my microservice repos?

Codebase Overview

Here are the five repositories found:

REPO NAME	COVERAGE %
`core-api`	**85.4%**
`web-frontend`	92.1%
`billing-service`	68.0%
`auth-module`	79.3%
`data-worker`	88.9%

The `billing-service` needs immediate attention. Would you like to see the report tree for that repo?

U Check the coverage totals for my latest commit in core-api.

✅ **Commit Coverage Report**

- **SHA:** `abc1234` (Latest)
- **Overall Coverage:** 85.4% (Target Met)
- **Metrics Breakdown:**

* Total Lines: 1,500
* Lines Covered: 1,200
* Missed Lines: 300
* Partial Coverage: 50

This build looks healthy and meets your team's minimum threshold.

U How is the coverage organized in the web-frontend project?

Report Structure for `web-frontend`

The report follows this hierarchy:

1. `/src/components` (95%):
* `Button.js` : 98%
* `CardLayout.jsx` : 92%
2. `/src/utils` (88%):
*`dateFormatter.ts` : 75% ⟵ Lower coverage area**
3. `/src/hooks` (91%):

Frequently Asked Questions

01 **How can I use Codecov MCP to check my overall test coverage?**

You simply ask your agent, 'What is the coverage for this project?' It will pull data from all linked repositories and give you a clean list of their current coverage percentages at a glance.

02 **Can Codecov MCP tell me if a specific commit passed testing?**

Yes. You can ask your agent to check the coverage totals for any recent commit SHA. It gives you precise numbers on hits, misses, and overall percentage, confirming build health instantly.

03 **Does Codecov MCP help me compare different code branches?**

Absolutely. You can ask your agent to list all development branches and then compare coverage between any two of them. This is crucial for seeing if a feature branch dropped below the main branch's quality standard.

04 **What kind of file structure information does Codecov MCP give me?**

The agent can retrieve a full, hierarchical report tree that mirrors your project's actual folder system. This allows you to pinpoint exactly which module or utility file needs more test coverage.

05 **Is Codecov MCP only useful for big organizations?**

No. While great for large codebases, it works just as well for small projects. You can list all your repositories and get a quick overview of where you stand on testing coverage.

06 What if I want to track metrics based on specific criteria?

You can use Codecov MCP to list defined coverage flags. This lets you monitor test completeness across custom categories, ensuring certain critical parts of the code are never overlooked.

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 `"codecov": { "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

Codecov 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 Codecov. 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	Codecov MCP
Server ID	`019d7576-4e75-733e-b1a9-3cd64c93a0f5`
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/codecov.