MCP SERVER | NO CODE | CLOUD HOSTED

# Qase MCP

## Check QA status without leaving your chat.

Qase MCP connects test management directly into your AI workflow. Ask your agent to pull project overviews, track failed test runs, or generate lists of open defects without ever opening the Qase dashboard. It brings critical QA data—from projects and milestones to individual test steps—right where you're coding.

A+ Quality Score 100/100

test-management | defect-tracking | qa-automation | test-reporting | milestone-tracking

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

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.

# Qase MCP

10 tools available
Cloud-hosted on Vinkius

Need to know if a feature is ready? This MCP connects your Qase workspace so you can treat your entire testing suite like another source of truth for your agent. You stop copying URLs, digging through dashboards, or manually compiling failure reports. Instead, you ask questions about the state of your code base. Want an overview? Your agent pulls up active projects and gives instant metrics on test cases and open defects across all of them. Need to check a specific build? You can get detailed lists of recent test runs and pinpoint exactly which steps failed, or even track project milestones against current execution status. All this happens through natural conversation, making your development process faster and less prone to context switching. By connecting Qase via Vinkius, you keep all your QA data connected to the tools your team already uses.

## Core Capabilities

**01 — Get Project Overviews**

Retrieve a list of active projects or gather specific details about one project's setup.

**02 — Inspect Test Cases and Suites**

Explore your test hierarchy, pull up the full steps for any case, or check automation status across entire suites.

**03 — Monitor Test Runs and Plans**

List all completed test runs, get deep analytics on execution outcomes (passed/failed), or view defined test plans.

**04 — Track Milestones and Defects**

Get a list of project milestones or pull every defect logged against failed tests, including severity levels.

**05 — List Specific Data Sets**

Fetch specific lists like test cases, runs, or defects that are linked to failures for review.

## One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to this MCP and provide your Qase API token.

02 Connect the service through your preferred AI client (Claude, Cursor, etc.).

03 Ask your agent a question like, 'List all projects with more than five open defects,' and it runs the query.

The bottom line is you manage QA operations by chatting with your agent instead of clicking through multiple dashboards.

## Built For

QA Engineers who are tired of switching between their IDE and a testing dashboard; Developers who need immediate failure reports linked to specific features, and Product Managers who need real-time proof of feature readiness.

**Quality Assurance Engineer**

Checks the test hierarchy for a project or pulls up all steps for a single test case while writing automation scripts.

**Software Developer**

Asks their agent to list recent defects or failed runs related to the feature branch they just committed.

**Product Manager**

Gets instant summaries of test run coverage and status against project milestones without needing a developer's report.

## What Changes When You Connect

01 You can immediately pull a list of active projects using `list_projects`, giving you an instant health check across the entire product line. No clicking through project dashboards needed.

02 When debugging, ask to retrieve details for a specific test case via `get_case`. You get all preconditions and step-by-step instructions in one go, perfect for writing unit tests or documentation.

03 Stop guessing about coverage. Use the combined tools to list runs and check milestones so you always know if the current build meets project requirements.

04 Don't waste time compiling bug reports. `list_defects` instantly surfaces every recorded defect linked to a failure, complete with severity levels and issue links.

05 Get full context on failures by using `get_run`. Instead of just seeing 'Failed,' you see the specific run details and can determine if it's an environment or code bug.

## Real-World Applications

**Need to know which features are blocked?**

A PM asks their agent, 'Show me all projects with open defects.' The agent runs `list_defects`, giving the PM a list of critical bugs and immediate visibility into release blockers.

**Debugging a failed build.**

A developer commits code. They ask their agent to 'Check recent test runs for project WEB.' The agent calls `list_runs` and then `get_run`, pinpointing the exact failing step so they can fix it fast.

**Preparing a status report.**

A QA engineer needs to summarize testing progress. They ask their agent to 'List current milestones.' The agent calls `list_milestones` and reports the completion percentage, giving an accurate view for the meeting.

**Onboarding a new team member.**

A new hire needs to see what's being tested. They ask their agent to 'List all projects.' The agent uses `list_projects` and provides a high-level overview of the entire testing portfolio.

# Patterns to Avoid

### Copying data manually

✗ AVOID

A user sees a failed test run in Qase, copies the defect ID, opens Jira, and pastes it into a spreadsheet to report the failure.

✓ INSTEAD

Instead, ask your agent to use `list_defects` or `get_run`. It pulls all necessary context—the bug ID, severity, and failing step—and presents it directly in your chat.

### Ignoring project scope

✗ AVOID

A developer only looks at the 'Mobile App' project status but forgets about the 'API V2' testing needed for deployment.

✓ INSTEAD

Start by asking to list all projects (`list_projects`). This forces a comprehensive view, ensuring you check every required service before declaring the build ready.

### Vague status checks

✗ AVOID

Asking 'Is testing done?' without context, leading to vague answers that don't help with actionable next steps.

✓ INSTEAD

Be specific. Ask for `list_milestones` and check the completion date against the current test run results using `get_run`.

## The Right Fit

Use this MCP if your main pain point is context switching. If you currently have to leave your chat window or IDE to open Qase, copy/paste data, and manually compile reports, this is for you. You gain deep visibility into the entire QA lifecycle—from listing projects to tracking defects and reviewing test steps. Don't use it if you just need a basic report of one thing; instead, build a custom script or workflow tool. This MCP excels at aggregation: taking data from many related areas (cases, runs, defects) and presenting it all in one conversational answer.

## The Problem with Dashboard Overload

Every time a feature is ready for QA review, your team has to open Qase. You click through the project overview, then navigate to test runs, check milestones, and finally filter down to defects. This process requires dozens of clicks, switching between tabs, and copying IDs just to create a simple status report for stakeholders.

With this MCP, you don't touch a dashboard. You tell your agent: 'Give me the health summary.' It gathers project overviews and defect lists in one shot, delivering an actionable text report right where you are working.

## Getting Real-Time QA Insights with Qase

Manual status checking requires running separate queries for projects, test runs, and defects. You then have to piece together whether the current development sprint has met its required milestones.

Now you can ask your agent to correlate these data points—checking `list_milestones` against recent `get_run` results—and get a definitive answer in seconds. It's instant, synthesized knowledge.

# Qase MCP: 10 Tools Available

These ten tools give your AI client access to every core function in Qase, allowing you to query projects, cases, runs, and defects using simple chat commands.

| # | TOOL | DESCRIPTION |
|---|---|---|
| 01 | `list_plans` | Lists all the formal test plans associated with the project. |
| 02 | `list_projects` | Retrieves a comprehensive list of all projects currently set up in Qase. |
| 03 | `list_cases` | Lists all individual test cases contained within a specified project. |
| 04 | `get_project` | Pulls specific details and metrics for one identified project. |
| 05 | `get_case` | Retrieves full details, including preconditions and steps, for one specific test case ID. |
| 06 | `list_suites` | Lists the groupings of test cases that make up a larger test suite. |
| 07 | `list_runs` | Gathers an overview of all test runs conducted within a project timeframe. |
| 08 | `get_run` | Provides detailed results and metrics for a single, specific test execution run. |
| 09 | `list_milestones` | Retrieves a chronological list of key project milestones and their status. |
| 10 | `list_defects` | Lists all recorded defects that were directly linked to failed test cases. |

# See It in Action

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

**U** List all Qase projects and show me their overall health.

I retrieved 3 active projects in your Qase instance.
1. Mobile App (Code: MOB) - 150 test cases, 3 active open defects.
2. Web Portal (Code: WEB) - 300 test cases, 12 test runs.
3. API V2 (Code: API) - 50 test cases, 0 defects.
Should I inspect the defects on the Mobile App project?

**U** Fetch the details of test case ID 45 in the WEB project.

Here are the details for WEB-45 ('User Login using SSO'):
- Pre-conditions: User must have an active Google SSO account.
- Step 1: Click 'Login with Google' - Expected: Google Auth modal opens.
- Step 2: Enter valid credentials - Expected: Redirect to dashboard.
Shall I write a Cypress test for this?

**U** Are there any recent defects added for the WEB project?

Yes, I queried the `list_defects` capability. There are 2 new defects recorded today:
1. Defect #11: 'CSS layout breaks on mobile' (Severity: High)
2. Defect #12: 'API timeout during checkout' (Severity: Critical)
Both are in 'open' status. Shall I pull the details for #12?

# Frequently Asked Questions

01 **How do I check all the test cases using Qase MCP?**

You can list available test cases by first asking to list projects, and then requesting specific test cases within a project via `list_cases`.

02 **Can I find defects from failed runs with Qase MCP?**

Yes. Use the `list_defects` tool. This function specifically pulls all logged defects that are linked to test case failures, so you never miss a critical bug.

03 **What if I need details on one specific project in Qase MCP?**

You use the `get_project` tool. This fetches detailed metrics for a single, identified project, giving you more than just the name and ID.

04 **Does Qase MCP help me track feature readiness?**

Absolutely. You can check progress by asking to list milestones (`list_milestones`) and cross-reference that with recent test runs using `get_run` for the latest status.

05 **How do I get steps for a single test case in Qase MCP?**

Use the `get_case` tool. This retrieves all the detailed information, including every step and its expected outcome, directly into your chat conversation.

06 **How do I securely obtain my Qase Token?**

Log in to Qase.io and click your profile icon to go to **Account settings**. Select **API Tokens** (or sometimes found under Apps for an integration token), and click **Create a new API token**. Add a name, click generate, and copy the string provided. It takes exactly 15 seconds. Paste it here to authenticate. Your token is encrypted at rest and injected securely at runtime.

07 **Can my AI write test scripts using the case details?**

Absolutely. Inside your IDE (like Cursor), you can ask the agent to 'Fetch case #12 from Qase project PROJ'. The tool retrieves the precise steps, preconditions, and expected results. The agent can then automatically generate Playwright, Cypress, or Selenium scripts based exactly on those Qase definitions.

08 **How can I check the results of a recent QA cycle?**

Ask your agent to `list_runs` for your project. This will surface your recent executions. If you notice a run with a high failure rate, ask the agent to pull `get_run` with that run's ID to dive into specifics and see which modules failed the automated checks.

### 09 Can it help me track Jira bugs linked to tests?

Yes. By using the `list_defects` capability, your AI can pull all registered defects in a Qase project. If your Qase is integrated with Jira or GitHub, the returned defect data includes external issue links, helping developers immediately map a failed test to the corresponding engineering ticket.

# 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 `"qase": { "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

# Qase 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 Qase. 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 | Qase MCP |
| Server ID | `019d75fb-1a3e-73d9-9f9a-edce7dd47c6f` |
| 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/qase.