MCP SERVER | NO CODE | CLOUD HOSTED

Gitpod MCP

Automate Dev Environments Via Chat.

Gitpod connects your AI agent directly to cloud development environments. Use this MCP to automatically create, start, stop, and manage entire workspaces for any repository. You can also handle organization settings, update environment variables, and audit team activity without touching a dashboard.

A+ Quality Score 98.33/100

cloud-ide | development-environments | automation | workspace-management | remote-development | ci-cd

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

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.

Gitpod MCP

26 tools available
Cloud-hosted on Vinkius

Your agent uses Gitpod to orchestrate complex development workflows through natural language commands. Instead of manually spinning up a dev machine or updating shared configuration files in a web UI, you tell your AI client exactly what needs to happen—like starting a workspace for a specific pull request URL. This MCP lets you manage the entire lifecycle: from creating a new organization and adding members to getting detailed audit logs on who used which environment.

It's about controlling the infrastructure of development itself. You can list all existing organizations, get details on their settings, or even delete an old workspace when the project wraps up. When you connect this MCP via Vinkius, your agent gains a full view of your cloud IDE assets, making devOps tasks feel like simple chat commands.

Core Capabilities

01 — Provisioning and managing workspaces

Start new development environments from repository URLs or delete existing ones when they are no longer needed.

02 — Organization governance

Manage team access by creating organizations, listing members, adding users, or removing them completely.

03 — Configuration management

Define and update repository-specific settings and environment variables to ensure consistent dev setups across all team members.

04 — Monitoring and auditing

Track usage by listing workspace sessions, checking audit logs, or reviewing current organization details.

One Click on Vinkius — From Prompt to Execution

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

01 First, subscribe to this MCP in Vinkius and provide your Gitpod Personal Access Token.

02 Next, issue a natural language command through any compatible AI client—for instance, 'Start a dev environment for the main branch of my repo.'

03 Your agent executes the necessary tools, creating the workspace or updating the configuration, and you get immediate confirmation that the task is done.

The bottom line is your agent manages cloud development resources using simple instructions instead of requiring complex API calls.

Built For

This MCP targets technical roles who spend too much time in multiple dashboards just to get a single dev environment running. It's for the DevOps engineer who needs programmatic control over team access and the developer who hates leaving their chat window while setting up code.

DevOps Engineer

Uses this MCP to automate organization settings, manage environment variables across multiple repositories, and track audit logs for compliance.

Engineering Lead

Monitors workspace usage by listing sessions or reviewing organizations to maintain cost efficiency and security standards.

Software Developer

Spins up fresh, isolated development environments instantly for feature branches or PRs without having to manually copy URLs into a dedicated IDE.

What Changes When You Connect

01 Stop clicking through multiple dashboards. You can now initiate a workspace directly via your agent, using the `create_and_start_workspace` tool to spin up an environment for any PR link instantly.

02 Never forget who has access. Use the listing tools—like `list_organization_members` and `get_organization_settings`—to manage team roles and ensure compliance without logging into a separate admin panel.

03 Maintain perfect consistency across projects by managing variables programmatically. You can use `create_environment_variable` to inject required keys, ensuring every developer starts with the exact same setup.

04 Save time and money on orphaned resources. When a feature is done, trigger the agent to run `delete_workspace` or `stop_workspace` instead of manually tracking down running machines.

05 Audit your team's activity easily. The `list_audit_logs` tool gives you an immediate view of resource usage across all teams, which is critical for cost management and security reviews.

Real-World Applications

Onboarding a new developer to a complex service

A new hire asks the agent to get started. The agent uses `create_organization` if needed, then runs `list_configurations` to find the correct project settings, and finally uses `create_and_start_workspace` to hand them a ready-to-go dev environment in seconds.

Reviewing access rights before an audit

The security lead asks for confirmation of who can deploy. The agent runs `list_organization_members`, checks the results with `get_organization_settings`, and then sends a summary report, eliminating manual database lookups.

Reproducing bugs from production code

A tester reports an issue linked to specific environment variables. Instead of asking the team to manually set up the build, the agent uses `get_configuration` and then `create_environment_variable` with the necessary values to recreate the bug locally.

Decommissioning a retired project

The manager knows Project Phoenix is finished. The agent first runs `list_workspaces`, confirms all environments are stopped, and then uses `delete_organization` to wipe clean the entire project footprint.

Patterns to Avoid

Trying to update settings manually

✗ AVOID

A developer sees a required API key is missing. They copy the key into a text file and tell their teammate to paste it, which introduces human error and delay.

✓ INSTEAD

Tell your agent directly: 'Update the repo config with a new environment variable.' The agent uses `create_environment_variable` to inject the key securely and programmatically.

Forgetting to shut down testing environments

✗ AVOID

A team finishes a massive feature branch, but the workspace is left running because no one remembered to kill it. This results in unexpected cloud billing charges.

✓ INSTEAD

When done, command your agent: 'Stop the dev environment for PR #45.' The agent executes `stop_workspace`, ensuring you don't incur unnecessary costs.

Modifying team roles via UI

✗ AVOID

An Engineering Lead needs to demote a former contractor. They log into the web interface, navigate to the member list, and try to change permissions—a process that takes multiple clicks.

✓ INSTEAD

Simply tell your agent: 'Remove Jane Doe from the organization.' The agent runs `remove_organization_member`, handling the entire role update instantly.

The Right Fit

Use this MCP if your primary pain point involves managing the lifecycle of development environments, team access controls, or repository configurations. If you need to programmatically start a dev workspace based on a PR link, manage organizational membership, or inject environment variables into a project setup, this is the tool for you.

Don't use it if your goal is simple chat-based knowledge retrieval (use an RAG/knowledge base MCP instead). Also, don't use it if you only need to send messages—that requires a messaging MCP. This MCP is strictly about controlling cloud compute resources and configuration data; it doesn't write code or generate documentation itself.

The headache of managing dev environments today

Setting up a fresh development environment for a pull request used to be a messy chore. You'd click into the Gitpod dashboard, manually paste in repository URLs, select the correct branch, and ensure all necessary secrets were entered under the 'Environment Variables' tab. If anything was wrong—a missing key or an outdated configuration—you spent 15 minutes emailing teammates to fix it.

With this MCP, you simply tell your agent: 'Start a workspace for PR #123.' The system handles everything in the background using `create_and_start_workspace`. You get instant access to a running dev machine without leaving your chat window. It turns a multi-step administrative process into a single command.

Gitpod MCP gives you control over environment setup

You don't have to manually track down who needs access or if the old project workspace was shut down. You can ask the agent, 'List all members of the core team and check their roles.' It pulls that data via `list_organization_members` instantly.

Your development workflow is now fully controlled by language. Instead of navigating between five different tabs—the settings page, the member list, the variable panel, and the workspace status—you talk to your agent, and it makes the changes.

Gitpod: 25 Tools for Cloud Development

Use these tools to control every aspect of your cloud development lifecycle. Create, start, stop, and manage workspaces, configurations, and user memberships with simple commands.

#	TOOL	DESCRIPTION
01	`create_and_start_workspace`	Starts an active cloud development environment from scratch.
02	`create_configuration`	Sets up a new set of rules and settings for a specific repository in Gitpod.
03	`create_environment_variable`	Adds a necessary secret or setting, like an API key, to a repository configuration.
04	`create_organization`	Establishes a new container for multiple teams and projects within Gitpod.
05	`delete_configuration`	Removes an old, unused set of rules from a repository configuration.
06	`delete_environment_variable`	Clears out an environment variable that is no longer needed for development.
07	`remove_organization_member`	Removes a user's access and membership from a Gitpod organization.
08	`delete_organization`	Completely deletes an entire Gitpod organization structure.
09	`delete_workspace`	Shuts down and removes a specific, active development workspace.
10	`get_configuration`	Retrieves all the details about an existing repository configuration to review its settings.
11	`get_organization_settings`	Gets a full report of the operational settings for a specific Gitpod organization.
12	`get_organization`	Fetches general details about an entire Gitpod organization, like its name and owner.
13	`get_workspace`	Retrieves the status and details of a specific workspace by its ID.
14	`join_organization`	Adds the current user to an organization using an invitation or access ID.

#	TOOL	DESCRIPTION
15	`list_audit_logs`	Provides a history of all actions taken within an enterprise Gitpod organization (requires Enterprise plan).
16	`list_configurations`	Shows a list of every repository configuration that exists inside an organization.
17	`list_environment_variables`	Displays all environment variables currently set for a specific repository configuration.
18	`list_organization_members`	Lists every user and their current role within the organization.
19	`list_organizations`	Returns a list of all Gitpod organizations that the connected account has access to.
20	`list_workspace_sessions`	Provides a log showing when specific workspaces were started or stopped in an organization.
21	`list_workspaces`	Shows a list of all currently defined workspaces within the connected organization.
22	`start_workspace`	Activates and starts up an existing, pre-defined workspace for immediate use.
23	`stop_workspace`	Pauses a running workspace, stopping its compute resources to save money.
24	`update_configuration`	Modifies the rules or settings for an existing repository configuration.
25	`update_organization_member`	Changes a user's role or permissions within the organization structure.
26	`update_organization_settings`	Modifies core operational settings for the entire Gitpod organization.

See It in Action

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

U List all my Gitpod organizations.

I've found 2 organizations: 'Engineering-Team' (ID: org_123) and 'Open-Source-Projects' (ID: org_456). Which one would you like to explore?

U Create and start a workspace for https://github.com/gitpod-io/gitpod.

Starting the `create_and_start_workspace` action... I've successfully triggered a new workspace for that repository. You can now access your cloud IDE.

U Stop the workspace with ID 'ws-abc-123'.

I am stopping the workspace 'ws-abc-123' now. The environment will be hibernated to save resources.

Frequently Asked Questions

01 **How do I use Gitpod MCP to start a development environment?**

You initiate this by telling your agent what repository URL you need. The agent then calls `create_and_start_workspace` and provides you with immediate access to the live cloud IDE.

02 **Can Gitpod MCP help me manage team roles?**

Yes, it manages membership using tools like `list_organization_members` and `update_organization_member`. This lets you change a user's access level instantly via command.

03 **What is the best way to audit workspace usage with Gitpod MCP?**

You use the `list_audit_logs` tool (if available for your account). It gives you a comprehensive history of actions, which is essential for tracking resource consumption.

04 **Does Gitpod MCP allow me to delete an entire organization?**

Yes. If a project is completely finished and decommissioned, the agent can run `delete_organization`, ensuring all associated resources are cleaned up.

05 **How do I update environment variables using Gitpod MCP?**

You simply ask your agent to add or change an environment variable. It calls `create_environment_variable` and ensures the setting is applied correctly to the target configuration.

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 `"gitpod": { "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

Gitpod 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 Gitpod. 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	Gitpod MCP
Server ID	`019e389f-955e-7225-b7ee-5f6eab188e83`
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/gitpod.