MCP SERVER | NO CODE | CLOUD HOSTED

Liveblocks (Collaborative) MCP

Manage live collaboration and shared data states.

Liveblocks (Collaborative) lets your AI agent manage real-time, shared infrastructure directly from conversation. Use this MCP to control collaborative rooms, monitor who is in a session, and synchronize complex documents across multiple users without touching an API endpoint. It's the complete toolset for orchestrating live state management.

F Quality Score 3.6/100

real-time-sync | multiplayer-experience | presence-tracking | collaborative-editing | web-sockets | state-management

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/liveblocks-collaborative

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.

Liveblocks (Collaborative) MCP

19 tools available
Cloud-hosted on Vinkius

This connector lets you treat your entire real-time backend—room creation, user presence, shared document storage, and discussion threads—as just another function call in a conversation. You can instruct your agent to create an entirely new project space, automatically assign permissions, and then instantly get a full list of who's currently working in it. Need to update the core data or patch a specific section of shared notes? Your agent handles that, treating complex document states like simple variables.

It's built for environments where state changes constantly; think live code editors, collaborative whiteboards, or multiplayer game lobbies. Instead of juggling multiple dashboards and manual API calls, you talk to your AI client, and it interacts with the infrastructure via Vinkius. This lets developers debug room storage, PMs review user feedback threads, and support staff verify permissions—all in one place.

Core Capabilities

01 — Manage Room Lifecycle

Create, read, update, or delete collaborative rooms with specific access controls.

02 — Track User Presence

Monitor which users are active in a room and set temporary presence indicators for agents.

03 — Synchronize Shared Data

Retrieve or update the content stored within a room's document state (Yjs).

04 — Control Collaborative Threads

Create new discussion threads and manage existing feedback loops.

One Click on Vinkius — From Prompt to Execution

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

01 First, subscribe to this MCP in your Vinkius catalog and provide your Liveblocks Secret Key.

02 Next, prompt your AI client with a goal, such as 'Create a room for Project X,' or 'List all users currently active in the design sprint.'

03 Your agent executes the necessary functions, retrieves the live data (like user lists or storage patches), and presents the result back to you in natural language.

The bottom line is your AI client performs complex infrastructure operations that used to require direct code integration.

Built For

Product Managers who get frustrated by needing engineering support just to check user flow; Full-stack Developers drowning in debugging multiple dashboards; and Support Teams that need immediate, real-time visibility into a customer's shared session state.

Full-stack Developer

Debug room storage or inspect Yjs document states directly from the IDE without writing boilerplate API calls.

Product Manager

Review active collaboration sessions and check user feedback threads to gauge project momentum without leaving their chat interface.

Technical Support Engineer

Quickly identify who is currently logged into a customer's room and verify the exact permissions or configuration needed for troubleshooting.

What Changes When You Connect

01 You can stop manually checking multiple dashboards for who's online. Use the `list_active_users` tool to instantly verify all active participants in any room, keeping your support process fast and accurate.

02 Debugging document state used to mean pulling raw JSON files. Now you use `get_ydoc` or `get_storage` to get a clean, actionable snapshot of shared data directly into your chat context.

03 When project scope changes, you don't need an admin ticket. Use `create_room` and `update_room` together to spin up new collaboration spaces and set precise permissions instantly.

04 Managing team feedback is easier than ever. Instead of losing threads in email chains, the agent can use `create_thread`, `list_threads`, and eventually `resolve_thread` to keep discussions organized right where they happen.

05 Need to replicate a development environment? The ability to call `initialize_storage` resets your workspace instantly, giving you a clean slate for testing without manual cleanup.

Real-World Applications

Onboarding a new client team

A PM needs to set up a private working space. They ask their agent: 'Create a room called Q3 Strategy with write access and list the available rooms.' The agent uses `create_room` and then calls `list_rooms`, giving the PM immediate confirmation they're in the right place.

Live debugging shared documents

A developer finds a bug in shared notes. They prompt: 'What is the current state of the core document?' The agent uses `get_ydoc` to retrieve the structured data, allowing the dev to see exactly what's wrong without manually inspecting logs.

Support ticket escalation

A support engineer needs to verify if a user can access a feature. They prompt: 'Check all permissions for Project X.' The agent runs `get_room` and then uses `list_active_users` to confirm who was last in the room, providing comprehensive context.

Starting a new design cycle

A team leader needs to gather feedback. They ask: 'Start a thread about the landing page copy and notify everyone.' The agent executes `create_thread` and then uses `broadcast_event` to alert all current users in the room.

Patterns to Avoid

Treating it like simple CRUD

✗ AVOID

Trying to use this MCP just to read a single user's profile or update a contact record. This tool is built for live, shared sessions.

✓ INSTEAD

If you only need basic data records (like names and emails), use a standard database MCP instead. Only use Liveblocks when the *state* of collaboration is the core requirement.

Forgetting permissions

✗ AVOID

Attempting to delete a room or patch storage without first confirming who can do it. The operation will fail with vague permission errors.

✓ INSTEAD

Always start by calling `list_active_users` and then use `authorize_user` if you need to grant specific permissions before attempting a destructive action like `delete_room`.

Misunderstanding data flow

✗ AVOID

Treating the shared storage as simple text. You can't just write 'Hi' and expect it to stick—it needs proper structure.

✓ INSTEAD

To modify content, you must use `patch_storage` with a defined JSON Patch operation, or send a binary update using `update_ydoc`. Don't try to manually format the data.

The Right Fit

Use this MCP when your business logic revolves around shared state and real-time presence. If you need an AI agent to know who is currently viewing, editing, or discussing something in a live environment, this is required. Specifically, if you need the capability to check active users with `list_active_users` or synchronize documents using `get_ydoc`, use it. Don't use it if your primary goal

is simple data storage (use a dedicated database MCP) or if you only need to manage user accounts without tracking their activity in specific rooms. This tool handles the 'session layer,' not just the 'data layer.'

The headache of stitching together real-time project status.

Right now, managing a collaborative project means jumping through hoops. You check Slack for who is available. Then you click into Figma to see what design assets are shared. Next, you open the wiki to review documentation and finally, you run an internal script just to confirm who was logged in when the last decision was made. It's tedious clicking across five different tabs.

With this MCP, your agent handles it all. You tell it: 'Check the status of the Q3 marketing room.' The agent immediately calls `get_room` and `list_active_users`, giving you a single, consolidated report that tells you who's present, what the current storage state is, and if any critical threads need attention. It just works.

Getting instant room status with Liveblocks (Collaborative)

Manual checking of user presence means opening a dashboard, filtering by date, and copying names—a process that takes minutes and is always out of sync. You also have to manually check if the project storage was properly initialized.

Now, you simply ask your agent to verify the room's status. It calls `get_room` and confirms the metadata and permissions instantly. The visibility into who's present and what the state is changes everything; it gives you actionable intelligence immediately.

Liveblocks (Collaborative): 19 Tools

Use these tools to manage every aspect of a live collaboration workspace, from creating rooms and tracking users to synchronizing complex document states.

#	TOOL	DESCRIPTION
01	`list_active_users`	Gets a list of all users currently connected to a specified room.
02	`authorize_user`	Generates an access token for a user, applying specific permissions and scopes.
03	`broadcast_event`	Sends a structured JSON event to all clients connected within a room.
04	`create_room`	Initializes an entirely new collaborative workspace with defined metadata and access rules.
05	`create_thread`	Starts a new discussion thread within a room, automatically posting the initial comment.
06	`delete_room`	Permanently removes an existing collaborative workspace and all its contents.
07	`get_room`	Retrieves comprehensive details about a specific room, including its current status and metadata.
08	`get_storage`	Fetches the entire contents of the room's shared storage in JSON or LSON format.
09	`get_ydoc`	Retrieves a snapshot of the document state using Yjs for structured data inspection.
10	`identify_user`	Obtains an ID token needed to verify and identify a user's presence in the system.
11	`initialize_storage`	Sets up or resets the shared storage area for a room, giving it a clean slate.
12	`list_rooms`	Lists all existing rooms, allowing filtering and pagination based on metadata.
13	`list_threads`	Retrieves a list of all discussion threads associated with a particular room.
14	`list_versions`	Shows historical snapshots and version control logs for the shared document state.
15	`patch_storage`	Applies specific JSON Patch operations to update only a small part of the room's storage content.
16	`resolve_thread`	Marks a discussion thread as resolved, closing out the conversation topic.

#	TOOL	DESCRIPTION
17	set_presence	Sets or updates the ephemeral presence status for an agent or user within a room.
18	update_room	Changes properties of a room, including its metadata and access permissions.
19	update_ydoc	Sends binary updates to modify the core Yjs document state.

See It in Action

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

U List all Liveblocks rooms created after January 1st with the metadata 'status:active'.

I've found 3 rooms matching your criteria: 'Project-Alpha' (ID: room_01), 'Design-Sprint' (ID: room_05), and 'Team-Sync' (ID: room_09). Would you like to see the active users for any of these?

U Check who is currently collaborating in room 'editor-prod-42'.

There are currently 2 active users in 'editor-prod-42': 'Alice (ID: user_A)' and 'Bob (ID: user_B)'. Alice has been connected for 15 minutes.

U Create a new collaborative room for 'Sprint 24 Planning' with default access set to 'room:write'.

Room 'Sprint 24 Planning' has been successfully created with ID 'sprint-24-plan'. Default access is set to write. You can now start adding threads or initializing storage.

Frequently Asked Questions

01 **How do I use Liveblocks (Collaborative) MCP to check users?**

You use the `list_active_users` tool. Just prompt your agent and tell it which room you want to check, and the system returns a real-time list of all connected participants.

02 **Can I reset shared document state with Liveblocks (Collaborative) MCP?**

Yes, you use `initialize_storage`. This function sets up or resets the room's storage area, giving it a clean starting point for new collaboration sessions.

03 **What is the difference between get_ydoc and get_storage?**

The `get_ydoc` tool retrieves the shared state using Yjs format, which is best for structured data. Use `get_storage` when you need the full, general contents of the room's storage area.

04 **How do I make a new collaboration space?**

You use the `create_room` tool. You provide details like the name and initial access controls, and the MCP handles generating the necessary infrastructure for you.

05 **Can Liveblocks (Collaborative) MCP handle user permissions?**

Yes. You can use `authorize_user` to generate tokens with specific read/write scopes, and `update_room` to adjust overall room permissions.

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 `"liveblocks-collaborative": { "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

Liveblocks (Collaborative) 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 Liveblocks (Collaborative). 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	Liveblocks (Collaborative) MCP
Server ID	`019e38b8-c89d-701e-9e1c-bfac05464d96`
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/liveblocks-collaborative.