MCP SERVER | NO CODE | CLOUD HOSTED

GetStream MCP

Manage Feeds, Followers, and Activities Natively

GetStream manages complex social feeds and activity networks for your applications. Use this MCP to programmatically add activities, track who follows what, update user timelines, and manage file collections without touching a database schema.

F Quality Score 3.6/100

activity-feeds | chat-api | social-infrastructure | real-time-sync | user-engagement | api-integration

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

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.

GetStream MCP

23 tools available
Cloud-hosted on Vinkius

This connector lets you orchestrate sophisticated social architectures using natural language instructions given to your AI agent. Instead of building complex microservices just to handle feeds or follow counts, you simply tell the MCP what needs doing—like adding a new activity post to a specific timeline or listing everyone who follows a particular feed. It handles the heavy lifting of managing activities and maintaining relationships between users and content streams automatically. For instance, your agent can quickly scrape Open Graph data from a URL so an activity post is immediately rich with context. When you connect this MCP through Vinkius, you gain access to industry-standard tools for building robust social features right inside your workflow, treating feed management like any other function call.

Core Capabilities

01 — Manage User Timelines

Add or remove activities directly from a specific user's activity feed.

02 — Control Follower Relationships

Programmatically make an account follow another feed, and list all current followers for any given feed.

03 — Update Content Metadata

Modify specific fields on existing activities—like marking a post as featured—without rewriting the entire activity record.

04 — Handle Rich Media and Collections

Upload, process, organize, and delete files or groups of content (collections) within your application's social graph.

05 — Scrape Web Content Details

Grab structured Open Graph data from a URL to enrich activity posts automatically.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to the MCP and provide your Stream API Key and JWT Token credentials.

02 Tell your AI agent exactly what social action you need—for example, 'Make this feed follow that user' or 'Add a new activity about X'.

03 The MCP executes the necessary tool calls against the live social infrastructure, updating feeds and relationships immediately.

The bottom line is, your agent handles all the complex API orchestration so you just talk about what you want done to your social network.

Built For

This MCP is critical for application developers and product managers who own features that require a dynamic activity feed or social graph. If your app needs anything from simple user following to complex content curation, this is the tool you need.

Product Manager

Monitoring overall feed health and spotting patterns in how users interact with content without having to dive into technical dashboards.

Frontend Developer

Testing out new social feature logic, like adding a post or managing follower lists, directly from the IDE instead of waiting for backend deployment cycles.

Community Manager

Running moderation tasks by checking who follows key feeds and updating content status through natural conversation prompts.

What Changes When You Connect

01 Stop writing boilerplate code for basic social features. You can manage follower relationships and activity feeds using simple natural language commands, like making an account follow a feed or listing all followers.

02 Keep your content rich and up-to-date. Use the get_open_graph tool to automatically pull structured data from any URL so that every posted activity is contextually relevant right out of the gate.

03 Handle large amounts of media efficiently. You can upload files using upload_file or process images with process_image, ensuring your content feeds always have high-quality, usable assets.

04 Maintain clean data structures. Instead of rewriting whole records, use partial_update_activity to modify single fields on an activity—for example, marking a post as featured—without risking unrelated metadata changes.

05 Organize complex media groups easily. Use tools like add_to_collection or batch_post_collections to group related items together, treating collections like first-class objects in your workflow.

Real-World Applications

Launching a New User Profile Feature

A Product Manager needs to test the new 'Follow' feature. Instead of writing backend integration tests, they ask their agent to run 'Make the feed X follow user:Y.' The MCP executes the tool call instantly, confirming the relationship works, and they move on to testing content creation.

Content Moderation Audit

A Community Manager needs to audit a specific topic. They instruct their agent to 'List all feeds following this feed' and then use get_activities to pull the last 50 posts, quickly identifying potential spam or policy violations.

Building an Asset Gallery

A developer needs a feature that groups images into themed galleries. They upload multiple pictures using upload_image and then use add_to_collection to group them all under 'Autumn Vibes,' which is immediately available for display.

Integrating External Links

A content creator posts a link from an article they just wrote. They instruct their agent to 'Get the open graph data for this URL.' The MCP runs get_open_graph, and the activity post automatically populates with the title, description, and image, making it look professional.

Patterns to Avoid

Trying to manage feeds via general database queries

✗ AVOID

Writing custom SQL or ORM logic to figure out if User A is following Feed B, leading to complex joins and stale data.

✓ INSTEAD

Use the specific tools. To check relationships, use list_feed_followers. To establish one, use follow_feed. This keeps your social graph logic contained and reliable.

Sending activities through generic message APIs

✗ AVOID

Treating a feed like a simple chat log, resulting in posts losing their structured metadata or inability to be paginated correctly.

✓ INSTEAD

Use the dedicated get_feed tool. It handles pagination and activity structure specifically for feeds, ensuring your client always gets clean, chronological data.

Using manual API calls for every update

✗ AVOID

Having to manually write code to change a single field on an old post (e.g., changing the 'is_featured' status) when everything else is fine.

✓ INSTEAD

Use partial_update_activity. This tool lets your agent target only the specific metadata you want to change, minimizing risk and keeping the operation clean.

The Right Fit

Use this MCP if your application's core functionality relies on maintaining a dynamic social graph, managing activity streams (feeds), or organizing content into collections. It excels at handling relationships like 'following' and structured data types like 'activities.' However, don't use this if you simply need to store unstructured user comments; for that, a basic database text field will suffice. If your primary goal is complex backend business logic—like

calculating inventory stock or processing payments—this MCP isn't the right fit. You should look at specialized financial or inventory tools instead.

The pain of building social features manually

Today, adding a simple 'Follow' button requires multiple steps: checking if the relationship exists in your database, writing triggers to update follower counts, and then updating various feed tables. This process isn't just repetitive; it introduces failure points that only surface when high traffic hits.

With this MCP, you simply instruct your agent to manage the social graph. Whether listing who follows a specific feed using list_feed_followers or enabling a new follower relationship with follow_feed, the complexity disappears. You get reliable, industry-standard social infrastructure logic right in your workflow.

GetStream MCP: Structured Activity and Content Management

Manual systems make it hard to group related content or process media consistently. You often end up with separate endpoints for image uploads, file storage, and then another set of API calls just to link them all together into a coherent 'collection.'

Now, you can use the dedicated tools like upload_image and add_to_collection in one go. The MCP handles the entire lifecycle—from uploading the asset to organizing it into a usable collection object—so your agent returns clean, structured data every time.

GetStream: 23 Tools for Feeds and Collections

These tools allow you to manipulate every aspect of a social feed structure, from managing individual activities to handling bulk collection updates.

#	TOOL	DESCRIPTION
01	`add_activity_to_feed`	Adds a new activity item to any specified feed timeline.
02	`add_to_collection`	Places one or more objects into an existing content collection.
03	`batch_delete_collections`	Deletes multiple specified collections in a single batch operation.
04	`batch_follow`	Makes one account follow several different target feeds simultaneously.
05	`batch_get_collections`	Retrieves data for multiple collections at once.
06	`batch_post_collections`	Creates or updates several collection objects in a single batch request.
07	`delete_collection_object`	Removes a single, specific content collection object.
08	`delete_file`	Deletes a file resource using its URL.
09	`follow_feed`	Establishes a following relationship by making one account follow a target feed.
10	`get_activities`	Retrieves activity records based on an ID or a foreign identifier.
11	`get_collection_object`	Fetches the details for one individual collection object.
12	`get_feed`	Retrieves a paginated list of activities from any specified feed.
13	`get_open_graph`	Scrapes and returns structured data (Open Graph) from a given URL.
14	`list_feed_followers`	Lists all the feeds that are currently following this feed.
15	`list_feed_follows`	Shows which other feeds this specific feed is following.
16	`partial_update_activity`	Updates only selected fields on an existing activity record, leaving all others untouched.
17	`process_image`	Performs operations like resizing or optimizing a given image file.
18	`remove_activity_from_feed`	Removes an existing activity item from its feed timeline.

#	TOOL	DESCRIPTION
19	`unfollow_feed`	Breaks a following relationship by making one account unfollow a target feed.
20	`update_activities`	Updates general metadata across multiple activity records.
21	`update_collection_object`	Modifies the details of an individual collection object.
22	`upload_file`	Uploads a generic file to be managed by the system.
23	`upload_image`	Uploads and processes an image file for use in content feeds.

See It in Action

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

U Show me the latest activities in the 'user' feed for user ID 'user_123'.

I've retrieved the feed for 'user:user_123'. There are 10 recent activities, including a post about 'New Product Launch' and a photo upload. Would you like to see the details of the most recent one?

U Make the feed 'timeline:alice' follow 'user:bob'.

Successfully executed the follow command. 'timeline:alice' is now following 'user:bob'. Future activities from Bob will now appear in Alice's timeline.

U Update activity ID 'act_999' to set the 'is_featured' field to true.

I have partially updated activity 'act_999'. The 'is_featured' field is now set to true. All other metadata remains unchanged.

Frequently Asked Questions

01 **Can GetStream MCP handle user following relationships?**

Yes. You can use follow_feed to create a new following relationship and list_feed_followers to see who is already following a feed, making graph management easy.

02 **How do I add a post to an existing user timeline using GetStream MCP?**

Use the add_activity_to_feed tool. You just need to specify the target feed slug and provide the activity content, letting the MCP handle the posting logic.

03 **Is get_open_graph used for anything other than links?**

No, it specifically scrapes Open Graph metadata from a URL. This is useful because it allows you to enrich activities with rich data (like titles and descriptions) even if the source link doesn't provide that info.

04 **What if I only want to update one small part of an activity?**

Use partial_update_activity. This tool is designed so you can change a single field, like setting 'is_featured' to true, without overwriting the rest of the post's data.

05 **Can I manage media files with GetStream MCP?**

Absolutely. You have dedicated tools like upload_image and process_image that handle file uploads, resizing, and general image processing for your content feeds.

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 `"getstream": { "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

GetStream 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 GetStream. 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	GetStream MCP
Server ID	`019e389e-736f-718f-b8df-ca50d1fc999a`
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/getstream.