MCP SERVER | NO CODE | CLOUD HOSTED

PubNub MCP

Track live presence and chat history in real-time.

PubNub enables your AI client to manage real-time communication, track user presence across channels, and handle message history. Use this MCP to publish messages instantly, monitor who is online right now, and retrieve past conversations for support or auditing purposes.

F Quality Score 40.67/100

real-time-sync | pub-sub | data-streams | presence-tracking | messaging | low-latency

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/pubnub-real-time-messaging

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.

PubNub (Real-time Messaging) MCP

31 tools available
Cloud-hosted on Vinkius

This MCP connects your agent directly into a global data stream network. It lets you orchestrate real-time communication—not just sending simple text messages, but managing complex interactions like tracking user activity or archiving entire chat logs through natural language prompts. You can monitor exactly who is in which conversation and even manage group memberships instantly. For instance, if your application needs to handle high volume data streams, the Vinkius catalog makes connecting this functionality straightforward, letting you focus on the logic rather than the connection details. With this MCP, you control everything from generating secure file upload links to purging old message logs for compliance.

Core Capabilities

01 — Publishing real-time content

Send a message payload instantly to any specified channel.

02 — Monitoring live presence

Determine which users are currently online, or what channels they have recently visited.

03 — Retrieving message history

Fetch logs of past messages from a channel and count how many total messages were sent.

04 — Managing user and group data

Get or update specific user profiles, list all available users, or change who belongs to which group.

05 — Handling shared files

Generate secure URLs for uploading assets or retrieving file content from the network.

06 — Maintaining data integrity

Delete old message histories in channels or remove specific user accounts when necessary.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/pubnub-real-time-messaging** — connect your AI agent in three steps.

01 Subscribe to this MCP and provide your PubNub Subscribe Key, Publish Key, and UUID credentials.

02 Your AI client uses these keys to establish a connection to the real-time data stream network.

03 You interact with the system using natural language commands, allowing your agent to manage live communications and retrieve history.

The bottom line is you get a persistent, low-latency channel for all things real-time communication.

Built For

This MCP is essential for platform architects building live chat features, support teams needing immediate visibility into customer status, or IoT developers monitoring connected devices. If your application relies on instant updates or tracking user state, this is the connector you need.

Support Engineer

Needs to check message history and see if a user has left a channel before escalating a live support ticket.

Platform Developer

Builds chat applications, managing both the sending of new messages and tracking which users are currently active online.

IoT Solutions Architect

Monitors device heartbeats through real-time channels and sends control commands that need immediate confirmation or logging.

What Changes When You Connect

01 You can track who is currently online using the `presence_here_now` tool. This means your support agent knows instantly if a user has logged off, which drastically improves response quality.

02 Building out an audit log? Use `get_message_history` to pull message logs for any channel and count them with `get_message_count`. This gives you the full context needed for compliance reports.

03 Instead of relying on a separate directory service, use `get_all_users` and `get_user` to manage user metadata right from your agent. You can ensure every interaction is tied back to accurate profile data.

04 File sharing becomes simple. The `generate_file_upload_url` tool gives you secure upload links, so users never have to send sensitive files through the chat itself.

05 You gain granular control over group access using tools like `get_memberships`. You can let your agent check who is allowed in a private channel before attempting to publish anything.

Real-World Applications

A customer support queue needs real-time status updates.

The support team asked their agent, 'Who's online right now in the billing chat?' The agent used `presence_here_now` and reported a list of active users. This allowed the technician to immediately prioritize help for the most engaged customers.

An internal tool needs to archive communications for legal reasons.

A compliance officer asked their agent, 'Get all messages from the 'project-x' channel last month.' The agent used `get_message_history` and also ran `delete_message_history` on older, irrelevant channels, keeping only what was required.

An IoT dashboard needs to know if a device is communicating.

The system architect told their agent, 'Check the heartbeat status for Device-7.' The agent used `presence_heartbeat` and confirmed that the connection remained active, triggering the next scheduled check.

A collaborative workspace needs to manage roles dynamically.

The team lead asked their agent, 'Add User A to the 'admin' channel.' The agent used `set_memberships` and then immediately updated the user profile using `set_user`, ensuring the permissions were correct.

Patterns to Avoid

Assuming data consistency

✗ AVOID

The developer tried to list users and assume they had all the necessary group details, leading to incomplete access checks.

✓ INSTEAD

Always check permissions first. Use `get_memberships` before allowing a user to participate in any channel, ensuring your agent validates both their profile (`get_user`) and their group status.

Relying on manual data cleanup

✗ AVOID

The team manually copied old message logs into a database, wasting hours of labor and risking format errors.

✓ INSTEAD

Use `get_message_history` to pull structured log data directly from the channel. If you need to prune old records for storage reasons, use `delete_message_history`.

Ignoring file security

✗ AVOID

The agent tried to send a large attachment by pasting it into the chat message body, which failed or was flagged.

✓ INSTEAD

First, generate secure access using `generate_file_upload_url`. Then, instruct your agent to upload the file via that URL and reference the content in the chat.

The Right Fit

Use this MCP if your core business logic revolves around communication flow: live chats, notifications, presence tracking, or collaborative workspaces. If you need to know 'who,' 'when,' or 'where' a piece of data was shared, PubNub is the right tool. Don't use it if your main job is simply reading and writing single records in a database; for that, a standard CRUD-type connector is better. Also, don't use this MCP to replace dedicated identity management services; rather, treat it as the communication layer that *uses* those identities. It excels at low-latency data streams, not long-term persistent storage.

Handling real-time user status used to be a nightmare of manual dashboard checks.

Before this MCP, knowing who was actively available involved juggling multiple dashboards. You'd have to check the main chat board for current activity, then switch to the internal directory to see if the person had changed roles, and finally consult a separate status page just to confirm they were logged in. It was slow, fragmented, and required constant manual cross-referencing.

Now, your agent handles it all automatically. By using tools like `presence_here_now`, you get a single source of truth on user availability. Your application instantly knows who's online and what channels they are active in. The result is immediate context for every interaction.

Controlling Message History with PubNub MCP

Manually auditing communications meant writing complex database queries across multiple message tables, trying to filter by specific date ranges and user UUIDs. If the chat system changed its logging format even slightly, your entire audit process broke down and required a full rewrite.

With this MCP, you simply ask for it using `get_message_history`. The agent handles the complex filtering and retrieval of chronological logs across channels, giving you accurate data whether you're looking at last week's chat or the single message from three months ago. It just works.

PubNub (Real-time Messaging) With 31 Tools

These tools allow your AI client to handle every aspect of real-time communication, from publishing a single chat message to managing large-scale user memberships.

#	TOOL	DESCRIPTION
01	admin_create_app	Creates a new PubNub application using the Admin API.
02	admin_get_metrics	Retrieves current usage metrics for administrative oversight.
03	admin_list_apps	Lists all existing PubNub applications on the account.
04	admin_list_keysets	Retrieves a list of all keyset identifiers.
05	delete_file	Removes a specific file from within a channel.
06	delete_message_history	Purges message history records for selected channels.
07	generate_file_upload_url	Creates a secure URL that allows another service to upload files directly.
08	get_all_channels	Lists every channel the application has access to.
09	get_all_users	Retrieves a list of all user accounts associated with the system.
10	get_channel	Fetches details and metadata for one specific channel.
11	get_file_url	Retrieves a direct, accessible URL for a file within the network.
12	get_memberships	Checks which groups a specific user belongs to or is restricted by.
13	get_user	Retrieves detailed information for a single user account.
14	get_message_history	Fetches the chronological log of messages from a specified channel.
15	list_files	Retrieves a list of all files stored within a given channel.
16	get_message_count	Counts the total number of messages ever sent to a specific channel.
17	pam_grant_token	Issues a temporary, secure token for internal system authentication (PAM v3).
18	pam_revoke_token	Immediately invalidates and revokes an existing PAM v3 token.
19	presence_get_state	Checks the current online status of a user within a specific channel.

#	TOOL	DESCRIPTION
20	presence_heartbeat	Sends an update confirming that a user is still active in their session.
21	presence_here_now	Gets a list of all users currently online and active in the network (Here Now).
22	presence_leave	Signals that a user is manually exiting or leaving a channel.
23	presence_set_state	Allows the system to set a custom online status for a user in a channel.
24	presence_where_now	Determines all channels that a specific user is currently participating in (Where Now).
25	publish_message	Sends an immediate, new message to a specified channel.
26	push_manage	Manages or lists the push notification channels for connected devices.
27	push_remove_all	Removes all registered push notification channels from a device's profile.
28	remove_user	Permanently removes a user account from the application context.
29	set_channel	Creates or updates metadata for an existing channel.
30	set_memberships	Updates which groups a user belongs to, defining their access rights.
31	set_user	Creates or updates the profile information for an existing user account.

See It in Action

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

U Publish a message to the 'alerts' channel saying 'System update scheduled for midnight'.

I've published the message to the 'alerts' channel. The timetoken for this message is 17156348210000000.

U Get the last 5 messages from the 'support-chat' channel.

I retrieved the last 5 messages from 'support-chat'. They include inquiries about billing and a technical setup question from user UUID-987.

U Who is currently active in the 'lobby' channel?

There are currently 3 users in the 'lobby' channel: 'admin-01', 'user-55', and 'guest-12'.

Frequently Asked Questions

01 **How do I check who is online using PubNub MCP?**

You use `presence_here_now` to get a list of all users currently active in the network. This tells you their real-time status without needing to query specific channels first.

02 **Can I delete old messages using PubNub MCP?**

Yes, use `delete_message_history` to purge message logs from entire channels. You can also target individual files with the `delete_file` tool for specific cleanup tasks.

03 **What is the difference between getting all users and getting a single user?**

Use `get_all_users` when you need to list every account for administrative purposes. Use `get_user` when you already know the ID and only need to retrieve specific profile details.

04 **How do I manage file uploads with PubNub MCP?**

First, call `generate_file_upload_url` to get a secure link. Then, instruct your agent on how to upload the file using that URL so it lands correctly in the channel.

05 **Can I update user permissions with PubNub MCP?**

Yes. You can manage this by first checking roles with `get_memberships`, and then updating them directly using `set_memberships` to control access rights.

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 `"pubnub-real-time-messaging": { "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

PubNub (Real-time Messaging) 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 PubNub (Real-time Messaging). 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	PubNub (Real-time Messaging) MCP
Server ID	`019e38dd-3d63-72ea-9d9e-b15066065d12`
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/pubnub-real-time-messaging.

© 2026 Vinkius · vinkius.com · support@vinkius.com