MCP SERVER | NO CODE | CLOUD HOSTED

Ably MCP for AI Agents

Manage real-time messaging and presence tracking infrastructure

Ably MCP lets your AI agent manage real-time communication streams using natural language commands. You can publish messages to multiple channels at once, track who is currently online across various groups, and send push notifications directly to client devices. This gives you granular control over the entire messaging infrastructure.

A+ Quality Score 98.33/100

pub-sub | real-time-messaging | push-notifications | websocket | event-driven | api-integration

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

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.

Ably MCP

14 tools available
Cloud-hosted on Vinkius

Need to build an app that feels truly real-time? This MCP connects your agent directly to Ably's full API suite for robust communication orchestration. Instead of writing complex boilerplate code every time you want to check status or send a broadcast, you just ask your AI client to do it.

Think about the sheer complexity of managing presence data across dozens of channels, or updating historical messages after an event. With this MCP, your agent handles all that API interaction for you. You can publish new messages, retrieve message history instantly, and even monitor who is active right now—all by simply asking it to 'check presence' or 'get channel status.' It's like having a dedicated backend developer sitting next to your IDE, ready to run any comms query on demand. Because Vinkius hosts this MCP, you connect once from Claude, Cursor, or any compatible client and get immediate access to all these real-time communication tools.

Core Capabilities

01 — Sending Messages

Send messages to single channels or multiple channels simultaneously, and retrieve the message history.

02 — Tracking Presence Status

Get a real-time list of active members in any channel, and review historical presence records for auditing.

03 — Sending Notifications

Fire off direct push notifications to specific devices or client IDs for immediate mobile/web alerts.

04 — Managing Channels

List all active communication channels and fetch detailed metadata or usage statistics for any given channel.

05 — Updating Records

Modify, delete, or append data to existing messages within a channel.

One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to the Ably MCP and enter your API key credentials into Vinkius.

02 Tell your AI agent exactly what you need: 'Check presence in the #alerts channel' or 'Publish a message about X.'

03 Your agent runs the necessary commands through this MCP, and it returns structured data—like current member lists or successful messages—directly back to your chat window.

The bottom line is you get full access to Ably's real-time messaging functionality without ever writing an API call yourself.

Built For

Backend developers, DevOps engineers, and product managers need this. If your job involves monitoring system health or ensuring that users receive timely, reliable updates across multiple channels, you'll use this MCP.

Backend Developer

Needs to quickly debug message flow issues by retrieving old messages or checking if a new channel was correctly initialized.

DevOps Engineer

Must monitor application usage statistics and manage API key security (using revoke_tokens) without leaving the command line interface.

Product Manager

Tests push notification payloads for specific features or verifies presence logic when launching a new user group.

What Changes When You Connect

01 Audit message history instantly: Use `get_messages` or `get_message` to retrieve specific content, eliminating manual database lookups.

02 Monitor active users in real time: The `get_presence` tool shows you exactly who is online right now without needing a live dashboard view.

03 Scale communication efforts: With `batch_publish` and `batch_push_publish`, you can send messages or notifications to dozens of groups simultaneously.

04 Maintain security integrity: Use `revoke_tokens` whenever an API key needs to be shut down immediately, keeping your comms infrastructure locked down.

05 Understand usage trends: The `get_stats` tool gives you application-wide metrics, letting product engineers track message volumes and connection peaks.

Real-World Applications

Debugging a New Feature Release

A Product Engineer needs to verify that the 'support' channel is receiving messages correctly after a deployment. They ask their agent to use `get_messages` and confirm if the expected event was published, getting immediate verification of message flow.

System-Wide Alerting

A DevOps Engineer needs to alert all development teams that an outage occurred. Instead of logging into multiple systems, they ask their agent to use `batch_push_publish` across three different client IDs for simultaneous alerts.

Monitoring Critical Stakeholders

A Project Manager needs to know who is available in the 'executives' channel right now. They query the MCP using `get_presence`, getting a clean list of active users without clicking through status boards.

Updating Outdated Records

A developer discovers that an important announcement was posted incorrectly. They use the agent to execute `update_message` on the specific message serial, fixing the error and appending the correct information for everyone.

Patterns to Avoid

Treating all communication as simple chat logs

✗ AVOID

Just asking for 'the last few messages' without specifying the channel or timeframe, leading to irrelevant data dumps.

✓ INSTEAD

Always narrow your scope. Use `get_messages` and specify both the target channel name and a time range so you get exactly what you need.

Ignoring security requirements

✗ AVOID

Leaving API keys active indefinitely, which poses a major risk if compromised.

✓ INSTEAD

Implement token rotation by using `revoke_tokens` as soon as the key is no longer needed. This keeps your real-time access tightly controlled.

Overlooking concurrent status checks

✗ AVOID

Running multiple individual presence checks for different channels (`get_presence`) one by one, which wastes time and API calls.

✓ INSTEAD

Use `batch_presence` to check the current online status across five or more channels in a single call. It's much faster.

The Right Fit

You need this MCP if your primary concern is real-time, multi-channel communication flow and presence management. Use it when you need an agent to act as the system administrator for messaging—for instance, checking `get_stats` or sending a mass alert via `batch_push_publish`. Don't use it if you only manage simple CRUD operations on non-real-time data (e.g., updating user profiles in a database). If your workflow is purely about static records, an identity management MCP will work better. But if the action involves 'is this person online?' or 'send this message immediately,' then Ably is what you need.

Ably MCP for AI Agents: Managing Real-Time Messaging Infrastructure

Today, managing a communication platform means clicking through separate dashboards: one tab for presence status, another for message history, and yet another to send out alerts. If you need to debug why a notification failed or find out who was last active in a specific group, you're forced into disjointed workflows involving multiple system logins and copy-pasting data points.

With this MCP, the process changes entirely. You simply ask your agent to 'Check the activity on the support channel.' It instantly combines presence checks (`get_presence`), message retrieval (`get_messages`), and metadata fetching (`get_channel_metadata`) into one cohesive answer. The result is a single source of truth directly in your chat.

Ably MCP for AI Agents: Ensuring Presence Tracking Reliability

The pain point here isn't just knowing who is online; it’s tracking *when* they were online, and ensuring that every user sees the status update immediately. Manually auditing presence usually means querying logs across different endpoints to build a complete picture.

This MCP provides specialized tools like `get_presence_history` and `batch_presence` . You can now ask your agent for an audit trail of who was online over the last hour or check multiple groups at once, giving you reliable status data instantly.

Ably: 14 Tools for Real-Time Messaging Infrastructure Control

These tools allow you to manage every aspect of your communication infrastructure, from listing channels to sending bulk alerts.

#	TOOL	DESCRIPTION
01	batch_presence	Checks and returns the current online status of users across several different channels at once.
02	batch_publish	Sends a message to multiple designated communication channels with one single call.
03	batch_push_publish	Publishes several direct push notifications simultaneously to various clients.
04	get_channel_metadata	Fetches details about a specific channel, like its creation date or current status.
05	get_message	Retrieves the full content of one message using its unique serial identifier.
06	get_messages	Fetches a chronological list of messages that were posted to a particular channel.
07	get_presence_history	Provides an audit log detailing who was online and when within a specific chat room.
08	get_presence	Gets the immediate list of all active, currently connected members for a given channel.
09	get_stats	Retrieves usage statistics about your entire Ably application's performance over time.
10	list_channels	Lists every active channel configured within the connected Ably account.
11	publish_message	Sends a new message to one specified channel, making it visible instantly.
12	push_publish	Sends a direct push notification payload to a target device or client ID.
13	revoke_tokens	Immediately invalidates an Ably API token before its expiration date for security reasons.
14	update_message	Allows you to edit, delete, or append new content to a message that already exists in a channel.

See It in Action

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

U What's the current user list and activity in the 'marketing-team' channel?

 # Marketing Team Status

- 🟢 **Active Members (3):** jsmith, klee, opsadmin
- 📜 **Recent Activity:**

* `jsmith` : Sent message about Q3 goals (2 minutes ago)
* `opsadmin` : Updated channel metadata: 'New lead intake protocol active.' (1 hour ago)

- **Channel Metadata:** Created 2024-01-15. Usage Level: High.

U Publish an urgent announcement to the #all-staff channel and send a push notification to my mobile device.

✅ **Status Update:**

- Message successfully published to `#all-staff` .
- 🚀 **Push Notification Sent:** Alert sent to your primary device ID. Check for 'System Maintenance Tonight' in 3 seconds.

U I need the message history and who was online yesterday in #support.

 # Support Channel Activity

- **Presence History:** Users seen active: (alex, ryan, sarah). Peak activity at 2 PM EST.
- **Message Summary:** Retrieved 47 messages since yesterday. Key topics included 'billing issue' and 'API connection error.' The latest message was about a payment gateway update.

Frequently Asked Questions

01 **How do I check who is currently online using the Ably MCP for AI Agents?**

You ask your agent to get presence data for a specific channel. It instantly returns a list of every active member, letting you know exactly who's available without manually checking status dashboards.

02 **Can I send out announcements to multiple teams at the same time with Ably MCP for AI Agents?**

Yes. You can use batch publishing tools to send the exact same message or notification simultaneously to dozens of different channels, making large-scale communication simple.

03 **What is the best way to audit who was online in a channel over time with Ably MCP for AI Agents?**

The agent handles this by retrieving the presence history. This gives you a detailed record of user activity, showing when members were active and inactive, which is great for compliance checks.

04 **Does Ably MCP for AI Agents let me update old messages?**

Absolutely. If an announcement was posted with bad data, you can use the agent to retrieve the message by serial number and then update or append new information right into the thread.

05 **Is this tool better than just using a standard chat client for tracking status?**

Yes. This MCP connects directly to Ably's underlying API, giving you structured data (like usage stats and historical logs) that goes far beyond what a simple user interface can provide.

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 `"ably": { "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

Ably 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 Ably. 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	Ably MCP
Server ID	`019e385f-afa9-73d9-9c2a-e9cac4d01d56`
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/ably.