MCP SERVER | NO CODE | CLOUD HOSTED

# Slack Bot MCP

## Audit history, manage users, and orchestrate your team's comms.

Slack Bot MCP gives your AI agent full control over your team's workspace communication. Instantly list channels, audit message history, check user activity, and manage messages across any Slack workspace without touching a dashboard or running complex queries.

A+ Quality Score 100/100

instant-messaging | workspace-management | channel-automation | notifications | team-collaboration

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

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.

# Slack Bot MCP

10 tools available
Cloud-hosted on Vinkius

Managing a large team's conversations in Slack usually means switching tabs, copying IDs, and manually checking multiple threads just to get context. This MCP lets your AI agent handle all that complexity through natural conversation. Instead of opening up the app, you simply ask your agent to perform actions across the entire workspace. It can list every public channel for quick auditing or pull message history from a specific chat thread instantly. Need to know who's online? Your agent checks user presence and delivers those status updates immediately. You can also send announcements or delete old messages right through your prompt, all without needing admin access or deep knowledge of Slack's internal workings. This whole capability is housed within the Vinkius catalog, connecting your preferred AI client to everything you need to manage communication.

## Core Capabilities

**01 — Audit channel details**

Get comprehensive metadata and information about specific or all public channels in the workspace.

**02 — Monitor user status**

Check if a specified team member is currently active, away, or offline.

**03 — Review message history**

Pull detailed archives of conversations from any channel for real-time monitoring or research.

**04 — Modify messages and channels**

Send new updates to specific channels, delete unwanted messages, join public chats, or leave them when done.

**05 — Manage user lists**

List all active users in the workspace and pull detailed profiles for any individual member.

## One Click on Vinkius — From Prompt to Execution

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

01 Subscribe to this MCP and provide your Slack Bot User OAuth Token.

02 Connect your agent via Claude, Cursor, or any compatible client.

03 Ask the agent to perform a specific action, like 'List all channels and check if Jane is online.'

The bottom line is that you tell your AI what needs doing in plain English, and it executes the necessary steps across Slack.

## Built For

This MCP is built for anyone whose job involves coordinating information across multiple team threads. If you spend too much time jumping between channels to get a single piece of context—whether status updates, old decisions, or member lists—you need this.

**Community Manager**

Auditing message history across public channels and verifying complete member lists without logging into the native dashboard.

**Project Lead**

Monitoring project-specific threads, sending status updates to relevant groups, or quickly checking user presence during a crunch time.

**Support Team Manager**

Performing rapid user status checks and sending bulk announcements directly from their chat interface for immediate team visibility.

## What Changes When You Connect

01 You stop manually checking dashboards. Your agent pulls detailed channel metadata using `get_channel_info`, telling you exactly what each public chat is for.

02 No more guessing who's available. Use the `get_presence` tool to instantly check if a user is active or away, letting you know where to direct urgent messages.

03 Need old context? Instead of scrolling through endless threads, use `get_history` to pull specific message archives from any channel directly into your workflow.

04 Managing announcements becomes instant. You can send updates using `send_message` and even delete mistakes with the `delete_message` tool—all via conversation.

05 You maintain control over data flow. By listing all users ( `list_users` ) and checking profiles ( `get_user_info` ), you keep a clear, auditable record of who's in the workspace.

## Real-World Applications

**Investigating a sudden drop in project momentum**

The Project Lead asks: 'What were the last 5 decisions made in #dev-updates?' The agent uses `get_history` to pull the critical messages, allowing them to quickly restart discussion without sifting through thousands of irrelevant posts.

**Onboarding a new employee into complex chats**

The Community Manager asks: 'List all public channels and summarize their purpose.' The agent runs `list_channels` and uses the metadata to give the new hire an instant, organized overview of every chat they need to know about.

**Handling a crisis announcement**

The Support Manager asks: 'Notify everyone that the server is down.' The agent uses `send_message` and targets the main `#general` channel, ensuring instant, reliable communication across all team members.

**Auditing compliance records post-incident**

The Operations Lead needs to verify who was involved in a chat. They use `list_users` and then `get_user_info` for key participants, building an auditable timeline of presence and accounts.

# Patterns to Avoid

### Trying to find missing context

✗ AVOID

The user manually scrolls through the `#general` channel using the web client, losing track of key dates or decisions buried deep in the chat.

✓ INSTEAD

Instead, ask your agent to use `get_history` on the specific channel. This retrieves a focused message archive, giving you exactly what you need without the scroll wheel.

### Over-relying on Slack's native search

✗ AVOID

The user searches by keyword but gets hundreds of irrelevant results from different threads and private messages.

✓ INSTEAD

Use `get_channel_info` to first narrow down the scope, then use `get_history` combined with targeted prompts. This limits the data retrieval to only the relevant public channels.

### Assuming everyone is reachable

✗ AVOID

The user sends an urgent message via chat but doesn't know if the person has seen it or even logged in.

✓ INSTEAD

First, check availability using `get_presence`. If they are offline, you can wait until they come back online instead of sending a non-read notification.

## The Right Fit

Use this MCP if your current workflow involves gathering data from multiple disparate Slack features—things like checking user status, auditing channel purpose, or pulling message archives. If the problem is 'I need to know what happened in Slack,' this tool solves it. Don't use this if you just need simple notifications; for that, a basic notification service works fine. However, if your requirement involves structuring and interpreting complex data—like running sentiment analysis on every single message retrieved—you might eventually need specialized tooling built *on top* of the `get_history` output.

## The Information Vacuum in Slack

Today, getting a clear picture of what happened in your team's chats is pure friction. You have to open up the main channel, scroll through dozens of conversations, jump to the user list to see who was available when the decision was made, and then copy-paste those details into another system just for documentation.

With this MCP, you talk directly to your agent. Instead of clicking through a dozen tabs and copying data piecemeal, you simply ask: 'What were the key decisions in #marketing last week?' Your agent runs `get_history` and presents a clean summary right back to you.

## Get Real-Time Status with Slack Bot MCP

Manual status checks involve either guessing or relying on the 'active' dot next to a name, which can be misleading. You waste time sending messages only to find out the person is actually away from their desk for the day.

Now, you check user presence using `get_presence`. Your agent gives you concrete, up-to-the-minute status reports directly in your chat window. It's immediate, reliable insight that saves minutes every single workday.

# Slack Bot With 10 Tools

These tools allow you to perform comprehensive actions across your Slack workspace, including managing messages, checking user availability, and auditing channel details.

| # | TOOL | DESCRIPTION |
|---|---|---|
| 01 | `send_message` | Sends a new text message to any specified Slack channel. |
| 02 | `delete_message` | Removes an existing message from within a Slack channel. |
| 03 | `get_channel_info` | Retrieves detailed metadata, including the purpose and topic, for a specific channel. |
| 04 | `get_user_info` | Fetches complete profile details and information about any user in the workspace. |
| 05 | `join_channel` | Causes your agent to join a designated public channel. |
| 06 | `leave_channel` | Removes your bot's presence from a specific Slack channel. |
| 07 | `list_channels` | Lists all available public channels within the entire workspace. |
| 08 | `list_users` | Retrieves an exhaustive list of every user currently registered in your Slack workspace. |
| 09 | `get_history` | Pulls the full message history for a given channel, allowing deep context retrieval. |
| 10 | `get_presence` | Checks and reports on whether a specific user is currently active or marked as away. |

# See It in Action

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

**U** List all public channels in my Slack workspace.

I've retrieved your public channels. You have 5 active channels, including #general, #marketing, and #dev-updates. Which one would you like to audit for history?

**U** Send 'Good morning team!' to #general.

Message delivered! I've sent your greeting to #general (ID: C12345). The message is now visible to all members.

**U** Check if user ID U12345 is currently active.

Checking presence... User U12345 is currently 'active'. I can retrieve their full profile details if you'd like.

# Frequently Asked Questions

01 **How can I use the Slack Bot MCP to see what channels exist?**

You run the `list_channels` tool. This immediately provides a complete list of all public channels in your workspace, giving you a full map of where team conversations happen.

02 **Does the Slack Bot MCP let me audit user data?**

Yes, it lets you retrieve detailed member lists using `list_users` and pull specific profiles for anyone with `get_user_info`. This helps keep your team's contact information up to date.

03 **What if I need message history from last month?**

Use the `get_history` tool. You simply prompt your agent to retrieve messages for a specific channel and time frame, bypassing the need for manual scrolling or search filters.

04 **Is it safe to use the Slack Bot MCP to delete old messages?**

Yes, you can use the `delete_message` tool. This is useful for removing sensitive information or correcting mistakes in public channels, but always ensure your agent's scope is correct.

05 **Does this MCP help me find out who was online?**

You can check real-time status using `get_presence`. This tool confirms if a user is currently marked as active or away, helping you know the best time to reach them.

# 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 `"slack-bot": { "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

# Slack Bot 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 Slack Bot. 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 | Slack Bot MCP |
| Server ID | `019d8481-e30e-727c-9f7e-efe4e245c341` |
| 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/slack-bot.