MCP SERVER | NO CODE | CLOUD HOSTED

Pub/Sub Subscription MCP

Build safe background workers for message queues.

Google Pub/Sub Subscription MCP lets your AI client act as a dedicated, secure background worker for one specific message queue. It pulls messages from a single Google Pub/Sub subscription and confirms completion. This setup is perfect for building automated systems that process incoming tasks reliably without needing global cloud permissions.

B Quality Score 87.3/100

event-driven | message-queue | asynchronous | pub-sub | data-streaming | webhook

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/google-pubsub-subscription

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.

Google Pub/Sub Subscription MCP

2 tools available
Cloud-hosted on Vinkius

This MCP gives your agent one surgical ability: safely pulling and confirming messages on a designated Google Pub/Sub Subscription. Because the access scope is locked down to just this single queue, you avoid dangerous global GCP permissions entirely. Your AI client can run as a highly reliable background worker, chewing through queued tasks without ever touching other workloads or subscriptions. It uses standard polling methods for maximum reliability. If your workflow needs an autonomous way to process messages coming into Google Cloud Pub/Sub, Vinkius hosts this MCP, letting you connect it easily from any compatible AI client like Cursor or Claude.

Core Capabilities

01 — Pulling queued tasks

Your agent retrieves batches of waiting messages from the configured subscription.

02 — Confirming message completion

The system removes processed messages from the queue, preventing them from being redelivered later.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/google-pubsub-subscription** — connect your AI agent in three steps.

01 Your AI client uses the `pull_messages` tool to check and retrieve waiting tasks from the designated Pub/Sub Subscription.

02 After your agent processes the data within the retrieved messages, it calls `acknowledge_messages` using the provided IDs.

03 The system confirms completion, permanently removing those messages so they don't reappear in the queue.

The bottom line is that you get a safe, focused mechanism to reliably process and clear out incoming message queues.

Built For

This MCP is for operations engineers and data architects who run mission-critical background processes. If your job involves reacting to events coming through a central messaging system—like order confirmations or sensor readings—you need this isolation. It's built for the person tired of manually checking logs across multiple cloud dashboards.

Data Engineer

They use this MCP to build reliable workers that consume data streams, ensuring every event is processed exactly once and acknowledged correctly.

DevOps Specialist

They connect it to automate batch processing jobs, allowing the AI agent to handle massive numbers of queued tasks without manual intervention or permission sprawl.

Software Architect

They rely on its strict scoping to isolate specific service functionalities, keeping worker processes safe and contained within a single queue.

What Changes When You Connect

01 You gain absolute containment. The agent only interacts with this one subscription, preventing it from accessing other critical workloads or queues.

02 Processing becomes reliable. By using `pull_messages` and then calling `acknowledge_messages`, you guarantee that processed tasks are removed, eliminating redelivery risks.

03 It handles scale naturally. This MCP allows your AI agent to chew through millions of queued tasks without performance degradation or manual scaling concerns.

04 You avoid permission overreach. Instead of granting broad global GCP permissions, this MCP limits access surgically to one subscription only.

05 The workflow is clean. Your agent pulls the messages with `pull_messages`, processes them completely, and then uses `acknowledge_messages` in a predictable two-step cycle.

Real-World Applications

Handling User Signups

A marketing team needs to process every new user signup event pushed into Pub/Sub. They ask their agent to first use `pull_messages` to gather the latest events, summarize them, and then use `acknowledge_messages` to confirm that all records were successfully updated in the CRM.

Real-time Sensor Data Ingestion

An industrial IoT system publishes sensor readings to a queue. The data team connects this MCP so their agent can autonomously run, pulling batches of data using `pull_messages`, running validation checks, and confirming receipt via `acknowledge_messages`.

Order Fulfillment Processing

A e-commerce backend pushes order fulfillment tasks to a queue. The operations team connects the MCP so their agent can pull the necessary messages with `pull_messages`, update inventory records, and then call `acknowledge_messages` once the entire transaction is complete.

Daily Batch Job Execution

Instead of running a cron job that needs high permissions, developers use this MCP. The agent pulls messages using `pull_messages`, runs the necessary financial calculations, and then uses `acknowledge_messages` to mark the batch as complete.

Patterns to Avoid

Using broad GCP credentials

✗ AVOID

Giving your AI agent full project read/write access because you think it's easier than setting up a single-purpose tool. This is dangerous and unnecessary.

✓ INSTEAD

Use the Google Pub/Sub Subscription MCP. It restricts the agent to one specific queue, ensuring that if the worker fails or acts maliciously, it only affects messages in that isolated subscription.

Manual cleanup after processing

✗ AVOID

After your client processes a batch of data, you manually run another command to delete the records from the queue. This is error-prone and violates standard Pub/Sub protocols.

✓ INSTEAD

Always use the `acknowledge_messages` tool immediately after successful processing. This confirms completion using native mechanisms, which is the correct way to handle message lifecycle.

Trying to read other queues

✗ AVOID

You need data from a second subscription but try to write generic code that reads both streams. This fails because global access is too risky.

✓ INSTEAD

If you have multiple queues, use this MCP for the primary queue and deploy a separate instance of the MCP for every other queue. Isolation keeps your system secure.

The Right Fit

Use this MCP if your process involves reacting to asynchronous events coming through a single, defined message queue. You need an autonomous worker that reliably pulls data (`pull_messages`), processes it, and confirms cleanup using `acknowledge_messages` . Don't use this if you need to *send* messages or write records; this

only handles consumption. Also, don't use it if your workflow requires reading from multiple different queues simultaneously—you must deploy the MCP once per subscription for maximum safety and scoping.

Message processing used to be a manual headache.

Today, when a critical event happens, you usually have to write complex scripts or build dashboards that constantly poll APIs. This means managing multiple endpoints, handling connection failures, and manually tracking which messages were processed versus which ones failed.

With this MCP, your agent handles the polling automatically. It pulls the data, gives your client the payload, and when the job is done, it confirms completion in one clean step. You just get the reliable result without managing the queue mechanics.

Achieve reliable message acknowledgment with `acknowledge_messages`.

Before this MCP, confirming a job's completion meant complex state management—you had to write custom logic just to tell the cloud service that your processing was finished. It felt brittle and required high-level permissions.

Now, after your agent processes data using `pull_messages`, calling `acknowledge_messages` is a single, clean instruction. You confirm success instantly, making the entire background workflow simple and robust.

Google Pub/Sub Subscription: 2 Tools Available

These tools allow you to pull waiting messages from the subscription and then confirm their processing status.

#	TOOL	DESCRIPTION
01	`acknowledge_messages`	Use this to confirm message completion, which removes them permanently from the Pub/Sub Subscription queue.
02	`pull_messages`	This tool pulls batches of messages from the configured Google Cloud Pub/Sub Subscription for processing.

See It in Action

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

U Pull the latest 5 messages from the subscription.

I retrieved 2 messages. The first message contains a JSON payload instructing a database backup. Here are their ackIds for when we finish processing.

U Process the first message payload and write a summary.

The message data contains an alert about high CPU usage. I have created the summary report as requested.

U Acknowledge the messages using these IDs: ["ack123", "ack456"].

Successfully acknowledged 2 messages. They will no longer appear in the queue.

Frequently Asked Questions

01 **How does Google Pub/Sub Subscription MCP work?**

This MCP lets your AI agent act as a specialized worker for one queue. It pulls messages from that single subscription and uses the `acknowledge_messages` tool to confirm processing when finished.

02 **Can I use this MCP if I need to read multiple queues?**

No, this MCP is strictly scoped to a single Pub/Sub Subscription. If you have more than one queue, you'll need to connect and deploy the MCP separately for each one.

03 **What happens if I forget to acknowledge messages using Google Pub/Sub Subscription MCP?**

The message will not be permanently removed. The system treats it as unprocessed and will redeliver it later, which can lead to duplicate processing errors.

04 **Which tool do I use first with Google Pub/Sub Subscription MCP?**

You must start by calling the `pull_messages` tool. This retrieves the current batch of tasks; you cannot acknowledge anything until you have data to process.

05 **Is this safe for production use?**

Yes, its primary feature is safety. Because it strips away global permissions and locks access to one subscription only, it's designed specifically for secure, contained background workers.

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 `"google-pubsub-subscription": { "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

Google Pub/Sub Subscription 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 Google Pub/Sub Subscription. 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	Google Pub/Sub Subscription MCP
Server ID	`019e38a3-407a-71cb-aeb0-43f3dd7f07ed`
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/google-pubsub-subscription.