



Automation > Gmail PubSub

Automation

Gmail PubSub

Goal: Gmail watch → Pub/Sub push → `gog gmail watch serve` → OpenClaw webhook.

Prereqs

`gcloud` installed and logged in ([install guide](#)).

`gog` (`gogcli`) installed and authorized for the Gmail account ([gogcli.sh](#)).

OpenClaw hooks enabled (see [Webhooks](#)).

`tailscale` logged in ([tailscale.com](#)). Supported setup uses Tailscale Funnel for the public HTTPS endpoint. Other tunnel services can work, but are DIY/unsupported and require manual wiring. Right now, Tailscale is what we support.

Example hook config (enable Gmail preset mapping):

```
{
  hooks: {
    enabled: true,
    token: "OPENCLAW_HOOK_TOKEN",
    path: "/hooks",
    presets: ["gmail"],
  },
}
```

To deliver the Gmail summary to a chat surface, override the preset with a mapping that sets `deliver` + optional `channel` / `to` :

```
{
  hooks: {
    enabled: true,
    token: "OPENCLAW_HOOK_TOKEN",
    presets: ["gmail"],
    mappings: [
      {
        match: { path: "gmail" },
        action: "agent",
        wakeMode: "now",
        name: "Gmail",
        sessionKey: "hook:gmail:{{messages[0].id}}",
        messageTemplate: "New email from {{messages[0].from}}\nSubject: {{message:",
        model: "openai/gpt-5.2-mini",
        deliver: true,
        channel: "last",
        // to: "+15551234567"
      },
    ],
  },
}
```

If you want a fixed channel, set `channel` + `to` . Otherwise `channel: "last"` uses the last delivery route (falls back to WhatsApp).

To force a cheaper model for Gmail runs, set `model` in the mapping (`provider/model` or `alias`). If you enforce `agents.defaults.models` , include it there.

To set a default model and thinking level specifically for Gmail hooks, add `hooks.gmail.model` / `hooks.gmail.thinking` in your config:



```
hooks: {
  gmail: {
    model: "openrouter/meta-llama/llama-3.3-70b-instruct:free",
    thinking: "off",
  },
},
}
```

Notes:

Per-hook `model` / `thinking` in the mapping still overrides these defaults.

Fallback order: `hooks.gmail.model` → `agents.defaults.model.fallbacks` → primary (auth/rate-limit/timeouts).

If `agents.defaults.models` is set, the Gmail model must be in the allowlist.

Gmail hook content is wrapped with external-content safety boundaries by default. To disable (dangerous), set `hooks.gmail.allowUnsafeExternalContent: true`.

To customize payload handling further, add `hooks.mappings` or a JS/TS transform module under `~/.openclaw/hooks/transforms` (see [here](#)).

Wizard (recommended)

Use the OpenClaw helper to wire everything together (installs deps on macOS via brew):

```
openclaw webhooks gmail setup \
  --account openclaw@gmail.com
```

Defaults:



Uses Tailscale Funnel for the public push endpoint.

Writes `hooks.gmail` config for `openclaw webhooks gmail run` .

Enables the Gmail hook preset (`hooks.presets: ["gmail"]`).

Path note: when `tailscale.mode` is enabled, OpenClaw automatically sets `hooks.gmail.serve.path` to `/` and keeps the public path at `hooks.gmail.tailscale.path` (default `/gmail-pubsub`) because Tailscale strips the set-path prefix before proxying. If you need the backend to receive the prefixed path, set `hooks.gmail.tailscale.target` (or `--tailscale-target`) to a full URL like `http://127.0.0.1:8788/gmail-pubsub` and match `hooks.gmail.serve.path` .

Want a custom endpoint? Use `--push-endpoint <url>` or `--tailscale off` .

Platform note: on macOS the wizard installs `gcloud` , `gogcli` , and `tailscale` via Homebrew; on Linux install them manually first.

Gateway auto-start (recommended):

When `hooks.enabled=true` and `hooks.gmail.account` is set, the Gateway starts `gog gmail watch serve` on boot and auto-renews the watch.

Set `OPENCLAW_SKIP_GMAIL_WATCHER=1` to opt out (useful if you run the daemon yourself).

Do not run the manual daemon at the same time, or you will hit `listen tcp 127.0.0.1:8788: bind: address already in use` .

Manual daemon (starts `gog gmail watch serve` + auto-renew):

```
openclaw webhooks gmail run
```

One-time setup

1. Select the GCP project **that owns the OAuth client** used by `gog` .

```
gcloud auth login
gcloud config set project <project-id>
```

>

Note: Gmail watch requires the Pub/Sub topic to live in the same project as the OAuth client.

2. Enable APIs:

```
gcloud services enable gmail.googleapis.com pubsub.googleapis.com
```

3. Create a topic:

```
gcloud pubsub topics create gog-gmail-watch
```

4. Allow Gmail push to publish:

```
gcloud pubsub topics add-iam-policy-binding gog-gmail-watch \
  --member=serviceAccount:gmail-api-push@system.gserviceaccount.com \
  --role=roles/pubsub.publisher
```

Start the watch

```
gog gmail watch start \
  --account openclaw@gmail.com \
  --label INBOX \
  --topic projects/<project-id>/topics/gog-gmail-watch
```

Save the `history_id` from the output (for debugging).

Run the push handler

Local example (shared token auth):



```
gog gmail watch serve \  
  --account openclaw@gmail.com \  
  --bind 127.0.0.1 \  
  --port 8788 \  
  --path /gmail-pubsub \  
  --token <shared> \  
  --hook-url http://127.0.0.1:18789/hooks/gmail \  
  --hook-token OPENCLAW_HOOK_TOKEN \  
  --include-body \  
  --max-bytes 20000
```

Notes:

- token protects the push endpoint (x-gog-token or ?token=).
- hook-url points to OpenClaw /hooks/gmail (mapped; isolated run + summary to main).
- include-body and --max-bytes control the body snippet sent to OpenClaw.

Recommended: `openclaw webhooks gmail run` wraps the same flow and auto-renews the watch.

Expose the handler (advanced, unsupported)

If you need a non-Tailscale tunnel, wire it manually and use the public URL in the push subscription (unsupported, no guardrails):

```
cloudflared tunnel --url http://127.0.0.1:8788 --no-autoupdate
```

Use the generated URL as the push endpoint:

```
gcloud pubsub subscriptions create gog-gmail-watch-push \  
  --topic gog-gmail-watch \  
  --push-endpoint "https://<public-url>/gmail-pubsub?token=<shared>"  
>
```

Production: use a stable HTTPS endpoint and configure Pub/Sub OIDC JWT, then run:

```
gog gmail watch serve --verify-oidc --oidc-email <svc@...>
```

Test

Send a message to the watched inbox:

```
gog gmail send \  
  --account openclaw@gmail.com \  
  --to openclaw@gmail.com \  
  --subject "watch test" \  
  --body "ping"
```

Check watch state and history:

```
gog gmail watch status --account openclaw@gmail.com  
gog gmail history --account openclaw@gmail.com --since <historyId>
```

Troubleshooting

Invalid topicName : project mismatch (topic not in the OAuth client project).

User not authorized : missing roles/pubsub.publisher on the topic.

Empty messages: Gmail push only provides historyId ; fetch via gog gmail history .

Cleanup



```
gog gmail watch stop --account openclaw@gmail.com  
gcloud pubsub subscriptions delete gog-gmail-watch-push  
gcloud pubsub topics delete gog-gmail-watch
```

[< Webhooks](#)[Polls >](#)

Powered by [mintlify](#)