OPEN-SOURCE BUILD GUIDE · AUTOJATE.COM/POST-ONCE

Build your own one-click publisher.

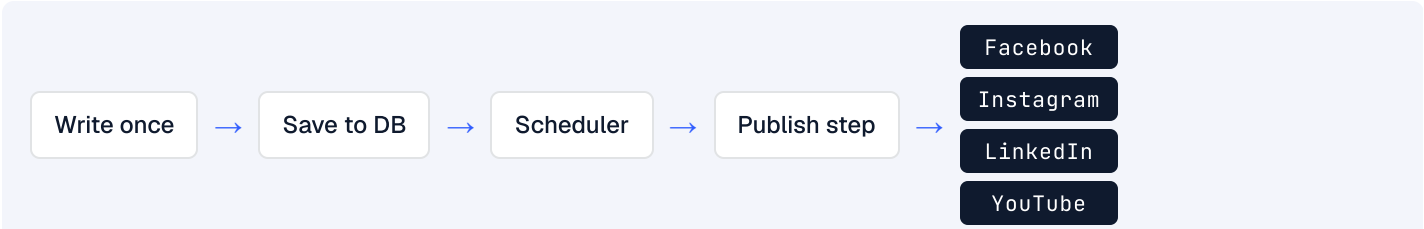
The exact stack I used, step by step, with the Claude Code prompts that do the heavy lifting.

> **~30 minutes** to build the app. The platform approvals take real-world days, so start those on day one (last page).

The stack I actually used

React + Vite · the front end
Supabase · database + storage
Edge Functions · the publish calls
pg_cron · the scheduler
ffmpeg · media prep
Claude Code · builds it

How it flows



Before you start (5 min, by hand)

- ☐ A free Supabase project
- ☐ A Meta developer account + a Facebook Page
- ☐ A LinkedIn developer app
- ☐ Node + Claude Code installed
- ☐ An Instagram Business account, linked to that Page
- ☐ A Google Cloud project (YouTube Data API on)

The build — 6 prompts

1 Scaffold the app

A React front end and a Supabase client, wired to your project.

CLAUDE CODE PROMPT — PASTE & RUN

```
Spin up a Vite + React + TypeScript app. Add Tailwind and a Supabase
client read from env vars (SUPABASE_URL, SUPABASE_ANON_KEY). One route,
one page, nothing fancy. Stop after it runs locally.
```

2 The database

Two tables: the post, and one row per platform so you can track status.

CLAUDE CODE PROMPT — PASTE & RUN

```
In Supabase, write a migration for two tables. posts: id, caption,
media_url, scheduled_at, created_at. post_targets: id, post_id (fk),
platform (facebook|instagram|linkedin|youtube), status (default
'pending'), external_id, error. Add RLS so only my authed user can
touch them. Give me the SQL.
```

3 The "write once" screen

Caption, media, four checkboxes. One submit fans out the rows.

CLAUDE CODE PROMPT — PASTE & RUN

```
Build the form: a caption textarea, a file input for image/video, and
four platform checkboxes. On submit: upload the media to Supabase
Storage, insert one posts row, and insert one post_targets row per
checked platform with status 'pending'. Toast on success.
```

4 The publish functions

One Edge Function per platform. This is the core of the whole thing.

CLAUDE CODE PROMPT — PASTE & RUN

```
Create three Supabase Edge Functions: publish-meta, publish-linkedin,
publish-youtube. Each takes a post_id, loads the post + its pending
target, and publishes using credentials from env vars only (never
hardcode). Meta: Graph API — post to my Facebook Page, and for
Instagram create a media container then publish it. LinkedIn: the UGC
posts endpoint (w_member_social) for my personal profile. YouTube:
Data API v3 videos.insert, resumable. On success set status 'posted'
and save external_id; on error set 'failed' + the message. Idempotent:
skip targets already 'posting' or 'posted'.
```

> **By hand:** you'll paste YOUR platform tokens in step 7. The code only reads them from env vars.

5 The scheduler

A cron job that picks up due posts and fires the right function.

CLAUDE CODE PROMPT — PASTE & RUN

```
Add a scheduled Edge Function 'publish-due' on pg_cron every 5 minutes.
Find posts whose scheduled_at has passed with pending targets, and
invoke the matching publish-* function for each. Fail closed: a missing
token marks that target 'failed' with a clear message, it never crashes
the run.
```

6 Media prep (the gotcha-killer)

Compress video and size images before upload. Saves you two real errors.

CLAUDE CODE PROMPT — PASTE & RUN

```
Before upload, transcode any video over 8 Mbps or 1080p to H.264 at a
sane bitrate (ffmpeg.wasm in the browser is fine), and resize carousel
images to 1080x1350. Add a note in the UI while it compresses. This
prevents Instagram error 2207053 and publish timeouts.
```

7 Your credentials (by hand, ~10 min)

Register each platform's app, get tokens, and store them as secrets.

SECURITY — READ THIS TWICE

Tokens go in **environment variables / Supabase secrets only.** Never in the code. Never in the repo. Never in the browser. The Edge Functions are the only thing that touch them.

If a token ever shows up in client-side code, treat it as burned and rotate it.

> **Meta:** create a Business-type app, generate a long-lived System User token in Business Manager → `META_TOKEN`.

> **LinkedIn:** create an app with w_member_social, run the OAuth flow → `LINKEDIN_TOKEN`. (Auto-refresh is partner-only; plan to re-auth by hand.)

> **YouTube:** OAuth client in Google Cloud, store the refresh token → `YT_REFRESH_TOKEN`.

The honest clock

~30 min

The build. Steps 1–6 with Claude Code. This part is fast.

Days

The approvals. Meta App Review, YouTube API audit, LinkedIn access. Start these on day one, in parallel.

The code is the fast part. The approvals are the wait. Don't let the wait stop you from building.

Want it built for you instead?

If you'd rather skip the developer portals and the waiting, that's the kind of thing I do. Book a call at **autojate.com/book** and I'll build the version that fits how you actually work.

AUTOJATE · AUTOJATE.COM/POST-ONCE · OPEN SOURCE, BUILD IT YOURSELF