Holen Ventures

AGENTS & WORKFLOWS · BUILD GUIDE

Build Your Own Daily AI Digest

Reads Hacker News, YouTube, newsletters, blogs, Reddit, and Twitter/X every morning, has Claude filter and theme it against what you care about, and posts a digest to Slack. Runs on Trigger.dev, a few cents a day, one config file to make it yours.

Henrik Holen

Updated June 2026 · ~30 minutes · Beginner-friendly

holenventures.com

Contents

WHAT YOU'LL BUILD, STEP BY STEP

● WHY TRIGGER.DEV, NOT A CRON ON A SERVER

Trigger.dev runs background jobs on a schedule, the kind of thing you'd otherwise stand up a server and a cron for. New to it? Their quickstart gets you running in a few minutes. The digest is six fetchers running at once, an analysis step, and a Slack post: Trigger.dev handles the scheduling, the parallel fan-out, and the retries when a source times out, with no server to keep alive. You write the tasks, it runs them on a schedule. This is the second build in this series that leans on managed infrastructure instead of plumbing, for the same reason: the boring parts are where these things break.

What it does (and doesn't do)

A scheduled run fans out to six fetchers in parallel, collects what they find, sends it to Claude, and posts the result to Slack.

1. **Collect.** Six sources, each a standalone task: Hacker News (top stories via the Algolia API, enriched with clean article text and the top comments), YouTube (recent videos from channels you list, with transcripts for real context), newsletters and blogs (RSS/Atom, last 24 hours), Reddit (Atom feeds plus stickied mod-bot TL;DRs on busy threads), and Twitter/X (via a public proxy, no API key needed).
2. **Analyse.** Claude Sonnet filters everything against your persona, groups it into your themes, and writes the digest: headline items, quick hits, and content ideas.
3. **Post.** A themed Slack message via an incoming webhook, formatted with Block Kit.

It reads the web and hands you a brief. That's all. It doesn't post anywhere but your Slack, take actions, or publish. Like a good morning paper, it tells you what's worth your attention and then gets out of the way.

Before you start

Services you'll need

You need three things. The rest are optional and only sharpen the output.

SERVICE	WHAT IT DOES	FREE TIER
Trigger.dev	Runs the pipeline on a schedule	Yes, covers a daily run easily
Anthropic	Claude Sonnet does the filtering and analysis	Pay-as-you-go, ~$0.02-0.05 per run
Slack incoming webhook	Delivers the digest	Free
Jina Reader *(optional)*	Clean article text for HN stories	Free (key raises limits)
YouTube Data API *(optional)*	Pulls videos from channels you list	Free, 10k units/day
Supadata *(optional)*	YouTube transcripts for deeper analysis	Free tier, 1 req/sec

You'll also need Node and a terminal. That's it.

> ● TIP
>
> Ship with just Trigger.dev, Anthropic, and Slack first. The optional keys (Jina, YouTube, Supadata) make individual sources richer, but the digest works without them. Add them once the basic loop is running.

One file does the configuring

`src/config/digest.config.ts` is the only file you need to touch. It controls four things: the **schedule** (a cron expression), the **persona** (a plain-English sentence telling Claude what you care about), the **themes** (how items get grouped, 2-4 works best), and the **sources** (which fetchers run and what they pull).

Clone, configure, deploy

Five steps from a fresh clone to a digest landing in Slack every morning. Closer to thirty minutes than an hour, and most of it is the config file.

Step 1: Clone and install

```
git clone https://github.com/hholen/daily-digest-trigger.git
cd daily-digest-trigger
npm install
```

Step 2: Configure your sources

This is where the digest becomes yours. Open `src/config/digest.config.ts`:

```
export const digestConfig = {
  cron: "0 6 * * *",
  persona: "a tech founder focused on developer tools, AI/LLMs, and SaaS growth",
  themes: ["AI & Agents", "Growth & PLG", "Builder & Startup"],
  sources: {
    hackerNews: {
      enabled: true,
      keywords: ["AI", "LLM", "agent", "startup", "SaaS", "developer tools"],
    },
    youtube: {
      enabled: true,
      channels: [
        { name: "Y Combinator", channelId: "UCcefcZRL2oaA_uBNeo5UOWg" },
      ],
    },
    // newsletters, blogs, reddit, twitter ...
  },
};
```

● THE PERSONA IS THE WHOLE GAME

It's one plain sentence, and Claude filters everything it collected against it. A vague persona ("someone interested in tech") gives you a generic digest. A specific one ("a B2B SaaS founder shipping an AI product, allergic to hype") gives you a sharp one. Spend your time here, not on the source list.

Finding the IDs you'll need:

- **YouTube channel ID:** open the channel page, view source, search for `channelId` (starts with `UC`).
- **Newsletter RSS:** most Substack newsletters are at `<name>.substack.com/feed`; Ghost blogs use `/rss/`.
- **Reddit:** just the subreddit name, no `r/` prefix (`"ClaudeAI"`, not `"r/ClaudeAI"`).

Step 3: Set your environment variables

```
cp .env.example .env
```

Fill in the three required keys (`TRIGGER_SECRET_KEY`, `ANTHROPIC_API_KEY`, `SLACK_WEBHOOK_URL`) and any optional ones you're using. For deployment, you'll set the same variables in the Trigger.dev dashboard under your project settings, not from the `.env` file.

Step 4: Run it locally

```
npx trigger.dev dev
```

This connects to Trigger.dev's dev server and registers your tasks. Open the Trigger.dev dashboard and trigger a test run. You'll watch the six fetchers fan out, the analysis step collect their results, and the Slack post fire. If a source returns nothing, that's usually fine (it just had nothing in the last 24 hours), but a thrown error in the run log points you straight at the misconfigured fetcher.

Step 5: Deploy

```
npx trigger.dev deploy
```

Set your environment variables in the Trigger.dev dashboard, then the cron schedule takes over. Tomorrow morning the digest lands in Slack on its own.

The fetchers, briefly

Each source is a standalone task. You don't need to read the code to use them, but it helps to know what each one actually pulls:

- **Hacker News** — top 30 front-page stories via the Algolia API, filtered by your keywords, then enriched with up to 1500 characters of clean article text (Jina Reader) and the top 10 comments.
- **YouTube** — recent videos from your channels via the Data API, each one's transcript fetched via Supadata so Claude sees the content, not just the title. Rate-limited to 1 request/second.
- **Newsletters** — RSS feeds (most Substack/Ghost newsletters expose one), filtered to the last 24 hours, HTML stripped.
- **Blogs** — RSS or Atom (set `format: "atom"` in the config), same 24-hour window.

- **Twitter/X** — tweets from accounts you list, via Feedly's public API as a proxy for xcancel.com feeds. No API key. Retweets filtered out.
- **Reddit** — subreddit content via Atom feeds, plus the stickied mod-bot TL;DR on threads with enough comments (high-signal community summaries).

Customising the analysis

The Claude prompt is built dynamically from your config, so changing `persona` and `themes` reshapes the whole digest without touching code. If you want to go further (change how many headline items it picks, or the shape of the content ideas), the prompt lives in `src/trigger/daily-digest/analyse-digest.ts`.

Adding a new source

The pipeline is built to extend. Every fetcher returns the same shape, `CollectedItem[]`, and as long as your new one does too, the analysis and posting steps work with it automatically. Three steps:

1. **Create the fetcher task** in `src/trigger/daily-digest/`, returning `CollectedItem[]` (title, url, snippet, source, sourceType, publishedAt, optional meta).
2. **Wire it into the orchestrator** in `run-daily-digest.ts`: import it and push it into the fan-out behind a config flag.
3. **Add config** in `digest.config.ts` if your source needs options.

TWO FIELDS DO QUIET WORK

`sourceType` controls how much of the snippet Claude sees (`"youtube"` and `"news"` get 1500 characters, everything else 500), so set it honestly. The optional `meta` field passes extra context straight to Claude (HN comments, Reddit TL;DRs), which is where a lot of the analysis quality comes from.

Known pitfalls

FOUR TRAPS WORTH KNOWING

Reddit RSS is actually Atom. The fetcher handles it, but it'll trip you up if you're debugging feed parsing and expecting RSS.

Supadata is rate-limited to 1 request/second on the free tier. The YouTube fetcher already adds a 1.1-second delay between calls; don't remove it or transcripts start failing.

Slack Block Kit sections cap at 3000 characters. The post task chunks long text automatically, so don't be surprised to see a digest split across blocks.

fast-xml-parser returns objects, not strings, sometimes. With `ignoreAttributes: false`, a text field can come back as `{ "#text": "...", "@_type": "html" }`. Every fetcher uses a `textOf()` helper to handle this. Reuse it in any source you add.

Going further

- **More sources.** The `CollectedItem[]` contract makes new fetchers cheap. A company changelog feed, a podcast transcript source, a search-alert API, all plug in the same way.
- **More than one digest.** Different persona, themes, and schedule per audience. A morning founder brief and a weekly competitor watch can run off the same pipeline with two configs.
- **Reshape the output.** The analysis prompt is just text. Point it at a different job (a tight three-bullet brief, a longer research roundup) by editing one file.

Build something with this?

This is one of a series of build guides. If you want help setting up something similar, or just want the weekly letter on building in the AI era, here's where to go.

MORE BUILDS	holenventures.com/agents-and-workflows
THE LETTER	holenventures.substack.com
WORK WITH ME	holenventures.com/work-with-me
EMAIL	henrik@holenventures.com