THE COMPANION PAPER

The Affiliate Stack

Technical breakdown

The full build behind two affiliate publications, The Caddie's Den and Covetory. Two static sites, one shared redirect worker that owns every affiliate tag, a links manifest, newsletter capture, a deploy I trust, and the SEO that makes any of it matter. The wiring, written for builders. No secrets, just the architecture.

Michael Gaff · michaelgaff.com · Build Note N-003

Built by one human and one AI partner. No agency, no staff.

01 — OVERVIEW

System overview and scope

Two publications, one pattern. The Caddie's Den is the narrow, credible niche, golf gear for people who actually play. Covetory is the wide one, gift guides and best of lists across every category. They are different brands with different voices, but under the hood they are the same machine: a static site, a shared way to attach affiliate tags, an email list, and a deploy I can run from a terminal in under a minute.

The goal of this paper is not to sell a passive income dream. It is to hand another builder the exact wiring so the weekend is spent on judgment, niches, voice, and the lists themselves, instead of on plumbing. Everything here is the architecture only. No live tags, keys, or private endpoints.

02 — HOSTING

Site architecture and hosting

Each site is a clean static build with no heavy framework. Plain HTML, a shared stylesheet, and a little JavaScript only where it earns its place. The sites deploy to Cloudflare Pages, one project per brand, each with its own custom domain. Static means fast, cheap, and almost nothing to break at runtime. There is no server to patch and no database to babysit.

Pages also gives me preview deploys for free. A non production branch builds a throwaway URL I can check before anything reaches the real domain. The production branch is the one that publishes. Keeping those two clearly separate is the difference between shipping calmly and shipping scared.

03 — THE WORKER

The redirect worker, and how every affiliate tag stays in one place

The single most useful piece of the whole build is a tiny redirect worker. Every buy button on either site points at a local link like /go/sand-wedge rather than straight at the retailer. The worker looks that key up, attaches the right affiliate tag for that site, and forwards the visitor on with a 302. The retailer never sees a hand edited link, and I never paste a tag into a page.

Why this matters: the tag lives in exactly one place. If Amazon changes my tracking id, or I want to test a new one, I change a single value and every button on the site updates at once. Editing a hundred buttons by hand is how you end up with three of them wrong and no idea which three.

```
// pseudocode, not the live worker
export default {
  async fetch(request, env) {
    const id = new URL(request.url).pathname
                 .replace('/go/', '');
    const dest = LINKS[id];          // from the manifest
    if (!dest) return notFound();
    const tag = env.AFFILIATE_TAG;   // one place, per site
    const url = withTag(dest, tag);
    return Response.redirect(url, 302);
  }
}
```

04 — DATA

The links manifest and data model

Behind the worker is a links manifest, a small map from a short key to a destination URL and a few fields of metadata, such as the retailer and the product name. It is plain data, version controlled alongside the site, so every change to where a button points is a visible diff in history. Nothing about the money path is hidden in a dashboard somewhere.

This also keeps the writing clean. In a list post I reference a key, not a sprawling tracking URL. The page stays readable, the manifest stays the source of truth, and swapping a dead product for a live one is a one line edit.

05 — EMAIL

Newsletter capture

Each site has an email capture wired to its own list. The form posts to a small subscribe endpoint that validates the address and adds it to the right audience, with the source recorded so I know which page and which site earned the signup. The list is the asset that survives any single post. Traffic is rented; an email list is owned.

The same endpoint backs the companion paper you are reading. Dropping an email to get the PDF and joining the weekly letter are one action, with a one click unsubscribe and no dark patterns.

06 — DEPLOY

The deploy pipeline and source control

Both sites live in version control. Deploys go out from the command line with a single command to the production branch. I leaned on the dashboard early and it liked to wedge in the middle of a deploy, so the terminal became the standing path. It is faster, it is scriptable, and it never spins forever on a screen that will not tell you what went wrong.

```
npx wrangler pages deploy . \
  --project-name <site> \
  --branch production
```

One note that cost me an hour, so it does not cost you one: a domain can read Active in your registrar and still not resolve, because the hosting project never claimed the hostname. The fix is to add the custom domain inside the Pages project, not just point DNS at it. The site was live and down at the same time until I did.

07 — SEO

SEO and structured data

Static sites are a gift for search. Clean URLs, fast loads, a real sitemap, and proper titles and descriptions on every page. For the list posts I add structured data so a gear roundup can be read as an item list rather than a wall of text. None of this is a growth hack. It is just doing the boring things correctly so the content has a chance to be found.

Before launch I ran small review panels: I pointed the AI at the build through one lens at a time, design, then conversion, then credibility, then SEO, and fixed what each pass surfaced. A weekend of that is worth a month of squinting at the site myself.

08 — IMAGES

Images and product feeds

Early on it is tempting to hotlink product images straight from the retailer. Do not rely on that. The links rot, and it is not yours to lean on. The real fix is licensed product feeds, the official affiliate data that ships approved images and current prices. That is a fast follow, not a launch blocker. Ship the lists first, then upgrade the images to a feed you are allowed to use.

09 — RESILIENCE

Failure modes and recovery

The honest list of what breaks, and what to do. A custom domain that resolves nowhere: claim the hostname inside the Pages project. A dashboard deploy that hangs: switch to the command line. A

dead product link: fix the one key in the manifest, not the page. A tracking tag that needs to change: edit the single value in the worker. The pattern in every case is the same, keep one source of truth so recovery is one edit, not a scavenger hunt.

10 — MONEY

What it earns, honestly

It is early and it is small. Affiliate income is not a faucet you turn on. It is a garden you plant. Amazon does not even keep your account unless you log a few qualifying sales in the first window, and the real lever is not the build at all. It is traffic and trust, which take content and time. What a weekend buys you is the machine, two properties wired correctly and ready to compound. The earning is next year's work.

11 — PRINCIPLES

Design principles

One source of truth for anything that can change, especially money. Static where you can, dynamic only where it pays for itself. Own the list, because traffic is borrowed. Automate the loop, keep the judgment. And ship the boring version first, then improve it in passes, because a live site earning nothing yet still beats a perfect one that never went up.

Read the note this paper accompanies at michaelgaff.com/journal/two-affiliate-sites. New build notes land first in The Gaff Letter.

— MG