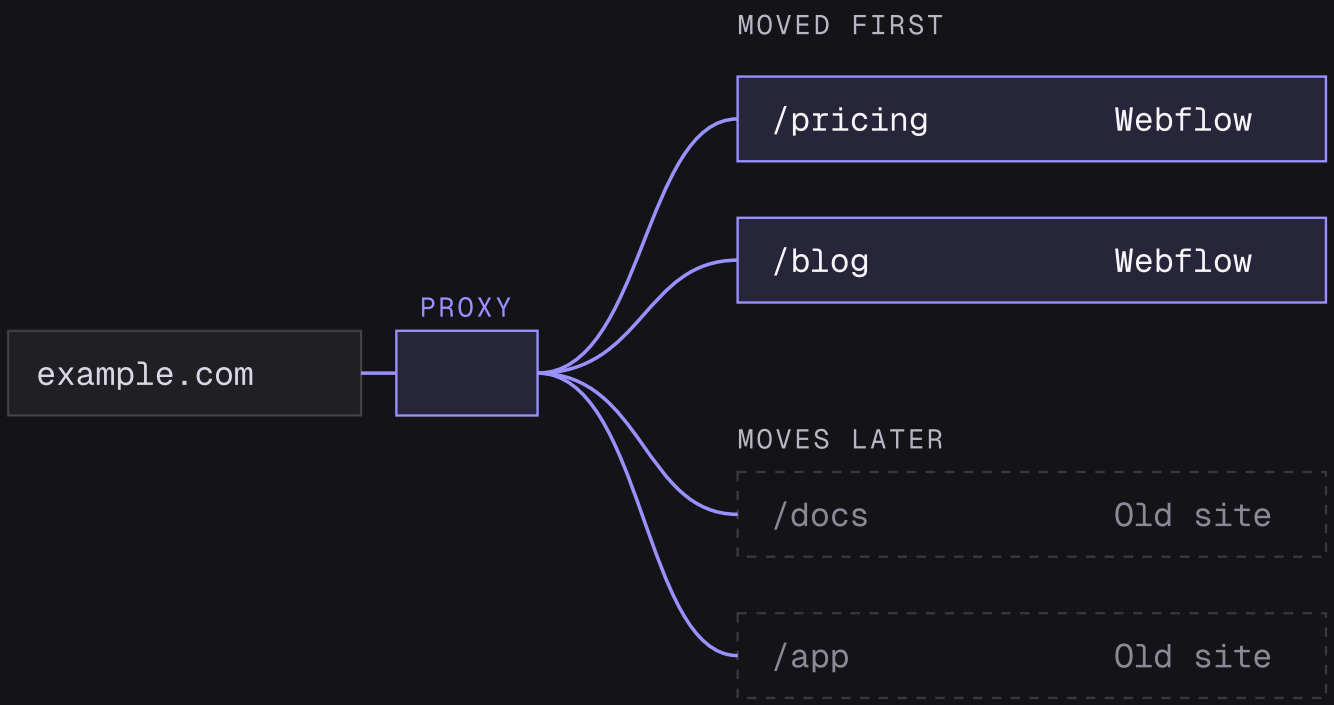
KOOKIES

GUIDE

Moving in Stages

6 CHAPTERS



WEBSITE MIGRATION

Move your site one piece at a time.

To move a website in stages, put a reverse proxy in front of your domain, send each path to the old or the new system from a written route table, and check every batch on the public domain before you start the next. It suits large sites and sites with parts that cannot move yet. Most small and medium sites should move all at once. It carries no ranking guarantee, and it adds a live system someone has to own until the old site is retired.

Who it is for. Marketing leads and developers on larger sites, or sites with parts that cannot move yet, who are planning a staged move or a platform change in phases.

01 When a full cutover is wrong

02 Reverse proxy setup in plain words

03 SEO parity for each page

04 Forms, tracking and sign-in across two systems

05 Moving platforms in phases

06 Retiring the old site

Read online at www.kookies.io/guides/moving-in-stages. Updated 2026-11-07.

CHAPTER 1 OF 6

When a full cutover is wrong

01

Most sites should switch in one go. Here are the few real reasons not to, the options that sit between the two, and what running two systems costs you each week.

The short answer

A full cutover is wrong when part of the site cannot move by the date the rest must, or when the site is so large that the old one keeps changing while you rebuild it. If neither is true, switch everything at once. It is simpler, and our Site Move Playbook covers that path.

Google's site move guide points the same way. It recommends that small and medium sites move all URLs at the same time, and says large sites can move a section at a time. That guide is about moves where URLs change, but the reasoning holds for a platform change too.

Two kinds of staged move

People say "in stages" and mean two different things. Work out which one you need, because the work is very different.

	ROUTING STAGES	PLATFORM PHASES
What is live	Two systems at once, on one domain	One system at a time
What moves	Batches of pages	The whole site, at planned points
Needs a reverse proxy	Yes	Not by default
Main risk	Two systems drifting apart	One phase never ending
Covered in	Chapters 2, 3, 4 and 6	Chapter 5

If you only want the idea of a reverse proxy and a quick view of the SEO points, read the short post on moving a site to Webflow in stages. This guide is for people who have decided to look closer and need the working detail.

Signs a full cutover is wrong

You need at least one of these, and you should be able to name the pages it applies to.

SIGN	HOW TO TEST IT
A section depends on a system you cannot replace yet	Name the system, the pages and who would have to rebuild it
The rebuild outlasts a realistic content freeze	Ask the content team how long they could stop publishing, then compare that with the build plan

SIGN	HOW TO TEST IT
The old contract or the new design has a date the full build cannot meet	Put both dates side by side, with the page count for each
The site is large enough that you want to see each section settle	Count the URLs in search, and ask if you could check them all in one launch day
A team owns one section and works to its own schedule	Ask that team when it could join a single launch

Google also says that, where it makes sense, you can start with a section that changes little, to see how traffic and indexing behave. That is a good first batch whichever method you choose.

Reasons that sound good but are not enough

- **"It feels less risky."** Stages split one big risk into many small ones, and add a new one: a live proxy. If your only worry is launch day, a better run sheet fixes that for less money. See the launch-day order.
- **"We can pay for it in parts."** You can usually phase the budget without phasing the routing. Launch the pages that matter on one date, then add the rest as new pages.
- **"We want to redesign page by page."** Visitors will cross from the new look to the old one on every visit. That is fine for a short time. It gets hard to defend if it lasts.
- **"We might keep the old system."** Undecided is not a plan. Decide, then read chapter 6 on staying with two systems on purpose.

The options in between

A proxy is the heaviest option here. Check the lighter ones first.

OPTION	HOW IT WORKS	GOOD FIT	CATCH
Smaller cutover	Remove or merge weak pages, then switch in one go	The site is long but only part of it earns anything	Every removed URL still needs a decision and a redirect
Short freeze	Stop publishing for an agreed window, then switch	The team can pause	Needs agreement across marketing
Separate subdomain	The part that cannot move goes to something like docs.example.com	A clear, self-contained section	Those URLs change, so they need redirects and their own move
Reverse proxy	Both systems sit behind one domain, and URLs stay the same	URLs must stay, and parts must wait	Someone has to run it, and everything depends on it

What two live systems cost

Add these to your plan before you commit. They run every week until the move ends.

- ✓ Two headers, footers and menus to keep matching.
- ✓ Two places to publish, and a list of which pages live where.
- ✓ Two sets of plans and contracts, including the paid Webflow Site plan the proxy setup requires.

- A proxy to host, watch and update, with someone on call for it.
- Caches at more than one level, and a way to clear them.
- Tracking, consent and forms that must behave the same on both sides.
- Support on your own. Webflow's help article on self-managed proxies says it does not support third-party proxies. If you are on Enterprise, it sends you to Sales to talk this through.

An outage matters more here too. If the proxy fails, every page behind it fails, on both systems.

Five questions before you commit

1. Which exact pages cannot move with the rest, and why?
2. Who owns the proxy, by name, and who covers when they are away?
3. What is the order of the batches, and which comes first?
4. What date do you expect the last batch, and what happens if it slips?
5. What will you do with the old system at the end?

DECISION RULE

Choose a proxy only if you can answer all five in writing. If you cannot name the pages, switch all at once. If you cannot name the owner, use a lighter option from the table above.

TAKEAWAYS

- Switch everything at once unless a named part of the site cannot move in time.
- Routing stages and platform phases are different projects, with different risks.
- Check a smaller cutover, a short freeze or a subdomain before you pick a proxy.
- Two live systems cost time every week, and Webflow will not support a third-party proxy for you.
- Write down the pages, the owner, the batches, the end date and the fate of the old site first.

CHAPTER 2 OF 6

Reverse proxy setup in plain words

02

Three hostnames, one route table and a few settings that decide if it works. How to plan the setup, what Webflow asks for, and how to test it before real traffic arrives.

The short answer

Point your public domain at the proxy. Give each system its own origin hostname, such as wf.example.com and old.example.com. Then write a route table that says which paths go where, with the old site as the default until most pages have moved. Everything else in this chapter is about making that hold up.

The routing idea itself is in the short post. Here we go through the setup.

Three hostnames, not two

Once the public domain points at the proxy, each system needs another address the proxy can reach.

HOSTNAME	POINTS TO	WHO SHOULD SEE IT
www.example.com	The proxy	Everyone
wf.example.com	Webflow	Only the proxy and your team
old.example.com	The old host	Only the proxy and your team

The old site is easy to forget here. It needs to answer on its new hostname, while still writing links with the public domain. Many CMSs have a site address setting that controls this, so check what yours does before you change it.

Do not leave these names to be sorted out on launch day. Set up and test the origins first, with the public domain still pointing where it does today.

What Webflow asks for

From Webflow's self-managed proxy article and its reverse proxy checklist:

- A paid Site plan and a connected domain.
- An origin subdomain, such as wf.example.com, set as the default domain.
- A global canonical set to your public URL, also used as the base for your sitemap.
- No proxying of *.webflow.io. Webflow says that is not supported and breaks its terms. It also stops that address working as your staging site.

- ✓ SSL on the origin. The checklist says wildcard certificates from Cloudflare are not compatible, so you need certificates for each explicit site name.
- ✓ The real visitor IP sent in `X-Forwarded-For`. Otherwise Webflow sees all traffic coming from one IP, which can look like an attack and get blocked.

These details change, so check the current docs before you build. If you are on Enterprise, Webflow directs you to Sales.

Write the route table first

Write this before you write any proxy code. It is your plan, your test list and your record. Here is an example for a site in its first batch.

PATH	GOES TO	BATCH	NOTE
`/` (exact)	Webflow	1	Exact match, or it catches everything
`/pricing`	Webflow	1	
`/about`	Webflow	1	
`/robots.txt`	One chosen system	Setup	Crawlers read just one file
`/sitemap.xml`	One chosen system	Setup	Can be an index of both
Everything else	Old site	Default	Includes the old site's files and scripts

Keeping the old site as the default means a mistake shows an old page, and that is rarely a disaster. It also means the old site's images, scripts and plugin files keep working without a rule each. When you later switch the default to the new system, you will need rules for every one of those paths, so log them now.

> DECISION RULE
>
> Switch the default only when the new system holds most of the pages and you have a list of every old path that must still go to the old site.

Rule patterns that catch people out

Pattern rules differ by tool, so read the docs for yours. Cloudflare's Workers routes are a clear example. A route maps a URL pattern to a Worker, and the Worker can act as a proxy in front of your server. Its docs say:

- `*` is the only operator, and it cannot sit in the middle of a path, so `example.com/*.jpg` is invalid.
- When several patterns match, the most specific one wins.
- Paths are case sensitive.
- Patterns cannot include query parameters, so URLs with query strings are matched only through a trailing `*`.
- A route with no Worker attached switches off a less specific route for matching requests.
- The domain needs a proxied DNS record.

Two habits follow from this, and they apply to most tools. A rule ending in `/blog/*` leaves `/blog` uncovered, so test both. A rule ending in `/blog*` also catches `/blogger`, so check nothing else starts with those letters.

Things the proxy must pass along

- **Query strings.** Campaign tags, site search and CMS pagination all depend on them. The checklist calls out all three, so test each.
- **Redirects.** If an origin answers with a redirect to wf.example.com, the visitor lands on the origin. Rewrite those addresses to the public domain, and fix the cause.
- **Trailing slashes.** Webflow removes them. The checklist says your proxy should match, to avoid redirects to the origin.
- **Status codes.** A 404 from an origin must stay a 404. If the proxy turns errors into a friendly 200 page, you create soft 404s.
- **The right cookies.** Chapter 4 covers this, since sending every cookie to both systems has a cost.

Caches and script deferral

You now have more than one cache. Webflow clears its own when you publish, but it cannot clear yours. Webflow's article gives two choices: turn off your CDN's caching, or plan to purge both. The checklist adds that the `site_publish` webhook can trigger your purge automatically, so no one has to remember.

If query strings change what a page shows, make sure the cache treats them as different pages. Never cache a response that sets a cookie or depends on a signed-in visitor.

Also turn off script deferral features on your CDN. The checklist names Cloudflare Rocket Loader, which can break interactions, forms, sliders and custom code.

Test before real traffic

You can test the whole setup on a spare hostname first, such as proxy.example.com, with a `noindex` header so it stays out of search. Then run this list against it, and again on the public domain.

1. Request every path in the route table and confirm which system answered. A temporary response header, such as `x-served-by: webflow`, makes this quick with `curl -I`.
2. Request a page that does not exist under each system, and confirm a real 404.
3. Search the page source for your origin hostnames.
4. Send a request with a query string and confirm it arrives intact.
5. Publish a small change and time how long it takes to show.
6. Turn the proxy's route for one page on and off, to rehearse a rollback.

Decide who gets the alert when it fails, too. Google's guidance on server errors says crawling slows down on 5xx responses, and URLs that keep returning them are eventually dropped from the index. An outage nobody notices is the real risk, so watch the public domain from outside, on a few URLs from each system.

TAKEAWAYS

- Each system gets its own origin hostname, and the public domain points at the proxy.
- Webflow asks for a paid Site plan, a connected domain, an origin subdomain and a global canonical, and rules out proxying webflow.io.
- Write the route table first, and keep the old site as the default until most pages have moved.
- Pass query strings, status codes and the real visitor IP, and rewrite redirects to the public domain.
- Plan your cache purge, turn off script deferral, and watch the public domain from outside.

CHAPTER 3 OF 6

SEO parity for each page

03

In a staged move, every batch is a small launch. What to capture before you switch a route, what to compare after, and the gate each batch must pass before the next one starts.

The short answer

Each page should say the same thing to a search engine after the switch as it did before, on the same URL. So capture each page before you change its route, switch the route, and compare the two on the public domain. Move the next batch only when the current one has passed.

This does not guarantee rankings. Google says to expect some temporary movement during a move, and nobody can promise where things land. Parity removes the problems you caused yourself.

The Site Move Playbook has a parity check for a one-off launch in its rebuild chapter. This chapter is what changes when you run that check again and again, with two systems live.

Why stages change the check

- **The URL stays, so nothing warns you.** In a normal move a redirect report shows what moved. Here the page simply starts coming from somewhere else. If a title goes missing, nothing flags it.
- **You check many times.** The routine has to be quick and repeatable, or it will be skipped by batch three.
- **Old pages can change too.** Links to moved pages and shared menus live on the old system, so old pages need checking as well.
- **A leak can come from either side.** Either system can expose an origin hostname.

Capture before you switch

Take this snapshot from the live public URL, just before the batch. Store it next to the route table, one row per URL.

CAPTURE	WHY IT MATTERS HERE
Status code and final URL	A new redirect is the first sign of a slash or path mismatch
Title, meta description, H1	The easiest to lose, and the easiest to compare by script
Canonical and robots tags	These tell you what the page is asking for
Schema as a block of text	Compare types and URLs, because they are often hard-coded
Internal links in the body	Shows if the new page links out to the same places
The heading list and word count	A fast way to spot content that was cut
Social tags	`og:url` and `og:image` often carry the origin hostname

CAPTURE	WHY IT MATTERS HERE
Clicks and impressions	From Search Console, for comparing later

Save the rendered HTML as well as the fields. When someone asks later what the old page said, it will be the only copy you have once the old site is gone.

On one revamp, the pages kept the same URLs, titles, meta, schema and links. That is the list that matters.

The batch check, step by step

Run this on the public domain right after you switch the rules for a batch. Repeat steps 2 to 6 the next day, in case a cache was hiding something.

#	CHECK	PASS MEANS
1	Confirm which system answered each URL	All on the system the route table says
2	Request each URL with redirects turned off	A 200, with no hop
3	Compare fields with the snapshot	No differences you did not plan
4	Search page source for origin hostnames	Zero matches
5	Read the canonical on every page	Absolute, public URL, and points to itself
6	Crawl the whole public domain	No new broken links or redirect chains
7	Check both sitemaps	Each URL in one sitemap, on the public domain
8	Inspect a few URLs in Search Console	Google can fetch the live page, and sees the right canonical

Canonicals: agree in every place

Google's canonical guide calls redirects and canonical tags strong signals, and sitemap listing a weak one. Signals can stack, but they should not disagree. Some details matter more once you have two systems:

- **Use absolute URLs.** Google recommends them, and warns that relative ones can cause trouble when a testing site gets crawled. An origin hostname is exactly that case.
- **Every page names itself.** Google recommends a canonical on the canonical page too.
- **Do not use robots.txt to hide an origin.** Google says not to use it for canonicalisation, because a blocked URL can still be indexed without its content. A canonical to the public URL is the fix Webflow's checklist points to.
- **Do not use noindex to choose between versions of a page.** Google advises against it, because it blocks the page from search entirely.

From time to time, run a `site:` query for your origin hostnames. It is a quick check that nothing has leaked.

Sitemaps during a staged move

Keep one sitemap per system, and list both in an index or in robots.txt. Google's sitemap guide allows several sitemaps and sitemap index files, and says submitting them separately lets you track each one in Search Console. That gives you a simple way to watch moved and unmoved pages separately.

- ✓ Full, absolute URLs on the public domain only. Google will try to crawl them exactly as listed.
- ✓ Serve them from the public domain. Unless you submit through Search Console, a sitemap affects only what sits below its own folder, so the site root is the safest place.
- ✓ In Webflow, the checklist says to use your global canonical URL as the base, or to enable a custom sitemap. It also says to replace the default sitemap address Webflow adds to robots.txt.
- ✓ Move each URL from one sitemap to the other in the same batch as its route.
- ✓ Keep `lastmod` honest. Google uses it only if it proves accurate, so a moved page with unchanged content should not look freshly rewritten.

Two things that often get missed

Links from unmoved pages. Old pages that link to moved pages should keep working, because the URL is the same. Check for old redirects, though. The old system may have a redirect rule that no longer fires, and that rule may need to exist on the new side. Our redirect map template works well for keeping track of where each rule lives.

Speed on the moved pages. A proxy adds a stop on every request. Compare Core Web Vitals and first response time for moved pages before and after, and look at your cache if they have slowed down.

When to start the next batch

Google suggests watching the Sitemaps and index reports, the search queries for your pages, and your server logs for crawler activity and errors during a move. For a staged move, compare the moved URLs against the group that has not moved. If both move together, the cause is probably not your batch.

> DECISION RULE
>
> Start the next batch when the check is clean, and Google has crawled the most important pages in the batch and accepted their canonicals. If the batch shows a real drop that the unmoved group does not, stop and find out why. Never move a new batch to cover a problem with the last one.

TAKEAWAYS

- With unchanged URLs, nothing warns you when a page loses its title, so compare each one yourself.
- Save a snapshot of every page, including rendered HTML, before you switch its route.
- Run the same short check on the public domain for every batch, and repeat most of it the next day.
- Absolute, self-naming canonicals on both systems, and one sitemap per system on the public domain.
- Compare moved pages with unmoved pages before you decide a batch is fine.

CHAPTER 4 OF 6

Forms, tracking and sign-in across two systems

04

Search engines are not the only ones crossing between systems. How to keep leads, data, consent and logins working when a visitor moves between two platforms on one domain.

The short answer

To the browser, two systems behind one proxy look like one website. That helps: same-domain cookies and browser storage carry across, and analytics usually needs no special setup. But it also means each system can read, overwrite or lose what the other set. So keep one tag setup and one consent tool on both sides, send forms to one destination, and test a real journey that crosses from one system to the other.

What the browser actually sees

Almost everything in this chapter follows from three points:

- **Same host, shared cookies.** A cookie set on www.example.com is sent for every matching path, whichever system answers. As MDN's Set-Cookie reference notes, Path is not a security boundary.
- **Same origin, shared storage.** Local storage and session storage belong to the scheme, host and port. So values saved on an old page are readable on a new one.
- **A cookie for the wrong domain is ignored.** If an origin sets `Domain=wf.example.com` while the browser is on www.example.com, the browser drops it. That is one reason a login or form "works on the origin, but not on the live site."

Forms

List every form on the site, with what it does after submit. Then decide this for each one.

QUESTION	WHAT TO CHECK
Where do the messages go?	The same inbox, CRM or sheet, whichever system holds the form
Does it post to its own path?	That path needs a route to the right system
Does the server check where the request came from?	It now arrives through the proxy, so origin and CSRF checks may fail
Is spam protection tied to a list of domains?	Make sure the public domain is on it
Does it redirect to a thank-you page?	Confirm the path, and any conversion tied to it
Does a security policy limit where forms can post?	Add the new destination before you switch the route

Move a page's form with the page, and test it in the same batch. Test by sending a real message and seeing it arrive, because a thank-you message only proves the page worked.

Run these tests with any CDN script deferral off. Webflow's checklist warns that features such as Rocket Loader can break forms.

Tracking

Google's cross-domain measurement help is for measuring across more than one domain, so pages on the same domain do not need it. On a proxy setup, both systems are on your public domain. So what matters is keeping the two sides identical.

- ✓ The same Tag Manager container or measurement ID on both systems, once each. A tag loaded twice, or a page with none, spoils the data.
- ✓ The same event names on both. Check the event a form sends on each side, and rename nothing during the move.
- ✓ Cookie domain the same on both, if you use subdomains as well. Google's help warns that different cookie domains can cause self-referrals.
- ✓ Conversions tied to a URL still fire. A thank-you path that moved systems should stay the same.
- ✓ Hits from origin hostnames are visible. Add a filter or check the hostname report, and look into any you find.
- ✓ Campaign tags survive. If hidden form fields read UTMs from the landing page, confirm the value carries over from a page on one system to a form on the other.

Add a `platform` value to page views, such as `webflow` or `old`. It lets you compare moved and unmoved pages in your reports without a separate list. Our tracking setup post has the base setup, and the analytics service page covers what we do.

Consent

Consent has to hold across the join. A visitor who rejects cookies on a new page should not be asked again, or tracked, on an old one.

- ✓ Use one consent tool, loaded the same way on both systems.
- ✓ Store the choice in a cookie with `Path=/`, so both sides can read it.
- ✓ Make sure tags wait for consent on both sides. Check the network panel before and after accepting, on a page from each system.
- ✓ Keep the same wording and categories, and one cookie list that covers both systems.

This is practical advice, not legal advice. Check what your rules require. There is more on the Webflow side in our post on cookie consent on Webflow.

Sign-in and private areas

If your site has any signed-in area, plan it as its own workstream. The safest setup is to keep it out of the move: leave the app where it is, on its own subdomain if it has one, and link to it.

If it has to stay on the shared domain, check these:

- ✓ **Session cookies.** The proxy would normally forward them to both systems. Only send cookies to the system that needs them, so an app session is never sent to your marketing platform.
- ✓ **Cookie scope.** MDN says a cookie with no Domain goes only to the exact host, while a cookie with `Domain=example.com` is sent to all subdomains, including your origins. Keep session cookies host-only where you can.
- ✓ **Name clashes.** Two systems may use the same cookie name. Look at what each one sets.
- ✓ **Login redirects.** After login, an app may send people to its own hostname. Rewrite or configure this to use the public domain.
- ✓ **Caching.** Signed-in pages should bypass the proxy cache. Test as two different users.
- ✓ **Webflow password pages.** The checklist says password protection does not work on pages served from a subdirectory through a proxy. It works only on root domains or the origin URL, so plan an alternative if you need it.

Site search is a related catch. Webflow's checklist says native site search does not work across multiple web properties, so a search box on one system will miss the other's pages. You may need custom code or a dedicated search tool for as long as the two systems run.

Test one crossing journey

Run this in a clean browser profile at each batch. Between them, these steps cover most of what goes wrong.

1. Land on an old page with a `utm_source` tag in the URL.
2. Accept or reject cookies, and note the choice.
3. Click through to a moved page. The banner should stay away, and tags should match your choice.
4. Send the form on the moved page.
5. Confirm the message arrived, with the campaign value on it.
6. Confirm the event shows in your analytics debug view, in one session.
7. Repeat the other way, from a moved page to an old one.
8. Sign in, if you have an area for that, and sign out.

`DONE WHEN`

All eight steps pass in both directions on the public domain, with a real message received. Keep the result beside the batch in your route table.

TAKEAWAYS

- Behind one proxy, the browser sees one site, so cookies and storage are shared between systems.
- A cookie set for an origin hostname is ignored on the public domain, which breaks logins and forms.
- Prove each form by receiving a real message, and check origin rules and spam protection lists.
- Same domain needs no cross-domain tracking, but tags, events and consent must match on both sides.
- Keep sign-in areas out of the move if you can, and test a journey that crosses both ways.

CHAPTER 5 OF 6

Moving platforms in phases

05

Sometimes a site should change platform more than once, or change one layer at a time. How to set up phases that each stand on their own, with Webflow to Astro and Sanity as the example.

The short answer

Change one thing per phase, and let each phase finish in a state you would be happy to stay in. A site that moves from Webflow to Astro and Sanity can do this: first a working site in Webflow, then the same site on the new stack, then new ways of making pages. If a later phase never happens, you still have a finished site.

This is different from routing stages. Here, usually only one system is live at a time, and each phase ends with a switch.

The rule: one change per phase

A site move can change four things: the platform, the design, the content and how the team edits pages. Each one that changes makes it harder to find the cause when something goes wrong. So split them.

PHASE	WHAT CHANGES	WHAT MUST NOT CHANGE	GATE TO THE NEXT PHASE
1. Build	Design and content, on a platform the team can use now	URLs you want to keep	Live and stable, with its URL and SEO list on file
2. Move	Platform and CMS	Design, copy, URLs, on-page SEO	Parity check passed, and the team can publish
3. New workflow	How pages get made	The look, and the URLs of live pages	A real page shipped through the new process

DECISION RULE

If someone asks to change the design during phase 2, write it down for after the move. Rewrites and redesigns are easier to judge against a stable baseline.

We have run all three phases on one site, in this order.

When to move off Webflow at all

Webflow is often the right place to stay, so a good build there is not a stepping stone by default. We cover the choice in when to move from Webflow to Astro. Have a clear reason in writing, such as wanting the site in a codebase your team controls, and name who will run that codebase after the move.

Phase 2: getting the content out

Plan this first, because it shapes the Sanity schema.

- ✓ **Code export will not bring your content.** Webflow's code export article says CMS content is not included. Collection lists export empty, and Collection pages do not show bound field content. User accounts, Ecommerce and localised content are left out as well.
- ✓ **Export each Collection.** The same article says Collections can be exported as CSV. Do it early to test your scripts, and again at the switch.
- ✓ **List every field and where it shows.** Plain text, rich text, images, references and option fields each need a matching Sanity field.
- ✓ **Keep slugs as they are.** They are what keep your URLs.
- ✓ **Copy your images.** Find every media URL in the content, including inside rich text, and bring those files across, so nothing depends on the old site.
- ✓ **Copy your SEO fields.** Titles, descriptions and social images per item often live in fields or templates that are easy to miss.

Phase 2: getting it into Sanity

Sanity's import docs set out a method that suits a repeatable move:

- Imports use newline-delimited JSON, one document per line, each with a `_type`.
- Give each document a stable `_id`. Sanity recommends this, and it means you can re-import later. Building the ID from the old item's slug or ID works well.
- References go in as weak and are strengthened after every document exists, so the order of documents does not matter.
- Assets can come from an HTTPS URL or a local path, using `_sanityAsset`.
- By default, the import fails if a document already exists. `--replace` overwrites, and `--missing` only adds what is absent.
- Imports can trigger webhooks, so consider turning those off first.
- Schema validation rules do not run on API or client changes, so write your own checks for what came in.

So write the move as a script you can run many times, not a one-off. Run it early and often, fix what breaks, and on the last day run it once more to pick up late edits. Our post on structuring Sanity content helps with the schema.

Phase 2: rebuilding the pages in Astro

Same design, so the test is simple: put the old and new page side by side at each breakpoint. Check these, as they do not come across on their own.

- ✓ **Trailing slashes.** Webflow removes them. Astro's configuration reference has a `trailingSlash` option that defaults to `ignore`, and a `build.format` option that defaults to folders. For prerendered pages, your host decides how slashes behave. So set these on purpose, and test on the real host.

- **Redirects.** Webflow's 301 list has to be recreated. Astro's docs warn that, in static mode with no adapter, its `redirects` option writes a meta refresh with no status code. For real 301s, use your host's config or a suitable adapter.
- **Interactions.** Plan to rebuild them in CSS and small scripts, with reduced motion respected.
- **Forms.** Decide where they post in the new setup, and test each one by receiving a real message.
- **Schema, sitemap and robots.txt.** Webflow generated its sitemap for you, so decide what now creates each of these.
- **Editors.** Build the Sanity Studio for the people using it, and have them publish a real page before the switch.

The switch: two ways

	WHOLE-SITE SWITCH	SECTION BY SECTION
How	Move DNS to the new host on one day	A proxy sends paths to Astro or Webflow
Best for	Most marketing sites, especially with the same design	Large sites, or a section with its own timeline
Follow	The Site Move Playbook	Chapters 2 to 4 and 6 of this guide

Because design and URLs stay the same in this phase, a whole-site switch is often the simpler route. Freeze publishing in Webflow for the last run of the script, and tell the content team when they can start publishing in Sanity. Keep the Webflow site published on a staging address until you are sure you do not need it, for quick comparisons.

Phase 3: a new way of making pages

Only start this once the new site is settled. A section library works because every part already matches the brand, so the job shrinks to choosing sections and filling them with content.

- Build sections from real pages, and name them by what they do.
- Write down what each section is for, and what content it needs. Both people and AI tools need this.
- Keep a person in the loop, with a staging link to review before anything goes live.
- Keep the URL, title and schema rules as part of the process, so new pages meet them from day one.

See Astro and Sanity and Webflow to Astro for how we run this work.

TAKEAWAYS

- Change one thing per phase, and finish each phase in a state you could stay in.
- Webflow code export leaves out CMS content, so export Collections separately.
- Script the Sanity import with stable IDs, and run it many times before the last day.
- Set slash behaviour on purpose, and do not rely on Astro static redirects for real 301s.
- Start new page-making workflows after the move settles, and keep a person reviewing.

CHAPTER 6 OF 6

Retiring the old site

06

A staged move ends only when the old system is gone, or when you decide on purpose to keep it. How to prove nothing depends on it, switch it off in order, and remove the proxy.

The short answer

Do not switch the old system off on the day the last page moves. First prove that nothing still depends on it, by reading its logs. Then take a backup you have tested, decide what happens to every leftover URL, restrict the old site for a while to see what breaks, and only then cancel it. Removing the proxy is a separate step, and you should treat it as a small launch.

First, decide if it should go at all

Sometimes the right ending is two systems, on purpose. A docs section or tool may belong on its own platform for good, behind the same domain. That is fine when all of this holds:

- ✓ The section has a named owner and a budget.
- ✓ The proxy has an owner, monitoring and a place in your yearly plans.
- ✓ The shared header, footer and tracking have a process for staying in step.

What you want to avoid is the accidental version, where a few forgotten pages stay on an old system nobody looks after, because nobody ever decided what to do with them.

> DECISION RULE
>
> Every path still on the old system needs one of three labels: moving by a set date, retiring by a set date, or staying on purpose with an owner. If a path has no label, give it one this week.

Prove nothing depends on it

Your route table shows what you meant to send to the old system. The logs show what actually goes there. Pull the old system's access logs and group them by path. If you have no logs there, use your proxy's.

WHAT YOU MAY FIND	WHAT TO DO
Pages with human visits	Move, redirect, or decide on purpose to remove them
PDFs, images and downloads	Move them with the same paths, or redirect them
Old scripts or styles loaded by new pages	Host them with the new site, then remove the references
Form or API endpoints	Find the sender, then move or replace the endpoint
Crawler visits to old URLs	Check them against your redirect map

WHAT YOU MAY FIND	WHAT TO DO
Bots probing for login pages and exploits	Ignore these

Then look at what the logs will not show. The old system may also send things out: scheduled jobs, emails from a plugin, feeds and webhooks to other tools. Check each of these by hand with whoever looks after the system.

Leftover URLs

Each URL still left over gets one treatment. Our redirect map tool helps with the list, and the Playbook's redirect chapter has the rules.

- **A close match exists.** Use a permanent redirect straight to it. Google's site move guide says to keep redirects for at least a year, and longer for users, if you can.
- **No match, and no value.** Let it return a real 404 or 410. Google's error guidance says URLs answering with these are removed from the index. Do not send many old URLs to the home page. Google advises against sending lots of old URLs to one irrelevant page.
- **Old redirects that lived on the old system.** These stop working when it does. Copy them to the new system, and point each one straight at its final URL, so you avoid chains.

The run sheet

Each step has a proof. Do not move on until you have it.

#	STEP	PROOF IT WORKED
1	Label every remaining path	Route table has no blanks
2	Stop publishing on the old system	Editors told, in writing
3	Back up the database and files, and try a restore	The restored copy opens
4	Save rendered HTML of any page being removed	Files stored with the project
5	Put redirects on the new system	Old URL list checked, each one correct
6	Change the proxy's default to the new system	Missing pages return a 404 from the new system
7	Remove the old sitemap from the index and robots.txt	Sitemap lists only live public URLs
8	Block public access to the old system, keeping it running	Old hostname asks for a login
9	Wait, and read the logs and error reports	No real traffic or breakage reported
10	Shut it down, and cancel the contract	Backup confirmed first, cost removed
11	Remove old DNS records	Mail and other services still work

Step 8 is the safety net. If something still needs the old system, you find out while switching it back on takes a minute. Agree how long to wait at step 9 based on what you found, and allow for anything that runs monthly or quarterly.

Take care at step 11. Your domain's DNS may also hold records for email and other tools. Remove only what you know belonged to the old site.

Removing the proxy

Once a single system holds everything, the proxy does nothing useful, and it is one more thing that can fail. Plan its removal like a small launch.

1. **List what it does beyond routing.** Redirects, security headers, rewrites, caching and logging may all live in it. Move what you still need to the host.
2. **Change the host's settings first where needed.** On Webflow, that means making the public domain the default again, in place of the origin subdomain. Review the global canonical, the sitemap base and robots.txt as part of this, and remove any base tag and href prefix you added for a subdirectory setup.
3. **Lower the DNS TTL ahead of time.** Do this at least a week before, then point the public domain directly at the host.
4. **Check SSL** on the public domain right after the change.
5. **Run your batch check** from chapter 3 across the whole site.
6. **Deal with the origin hostname.** Redirect it to the public domain so old links and caches resolve, and keep watching for it in search.

The URLs have not changed, so you should not need Search Console's Change of Address tool. Google's guide says it is only for moves between domains or subdomains.

What to keep afterwards

- ✓ The tested backup and your saved page snapshots.
- ✓ The final route table, and the full list of redirects.
- ✓ A short note on what moved, when, and which systems were involved, in your site docs.
- ✓ A date on the calendar to review the redirects, in about a year.
- ✓ An eye on search for a while. The Playbook's first 30 days has a schedule you can reuse.

Then look at what the move was for. Can the team now publish without help? Is there a single place to edit a page? If you want help looking after the site from here, see our care plan.

DONE WHEN

One system answers every URL, or every path on a second system has an owner, and nothing you pay for is serving a page nobody chose to keep.

TAKEAWAYS

- Keeping two systems can be right, as long as it is a decision with an owner.
- Read the logs to prove nothing still depends on the old system, then check what it sends out.
- Block public access and wait before you shut it down, so switching back stays easy.
- Copy old redirects to the new system, and let pages with no match return a real 404 or 410.
- Removing the proxy is a small launch of its own, with its own checks.

QUESTIONS

Questions people ask.

What is a staged website migration?

It means moving a site in parts. Usually a reverse proxy sends each URL to the old or the new system, so both stay live on one domain while pages move in batches. It can also mean changing platform in phases, with one system live at a time.

Should I move my website all at once or in stages?

Most small and medium sites should move all at once. Google recommends that in its site move guide, and says large sites can move a section at a time. Stages make sense when a named part of the site cannot move in time. Chapter 1 has the tests.

Does a reverse proxy hurt SEO?

Not by itself, if it is set up well. Search engines see one domain and the same URLs. Trouble comes from leaked origin hostnames, wrong canonicals, broken status codes and outages. No one can guarantee rankings during a move, so check each batch.

Does Webflow support reverse proxies?

Webflow says it does not support third-party proxies, and its help article sends Enterprise customers to Sales. The article lists a paid Site plan and a connected domain as needed. Do not proxy a webflow.io address, as Webflow says that is not supported and breaks its terms. Check the current docs.

How is this different from the Site Move Playbook?

The Site Move Playbook is for a move that happens all at once, from first decision to the first 30 days after launch. This guide is for what is different when two systems run together, or when a move happens in phases.

How long can two systems run side by side?

There is no fixed limit, but the cost keeps running: two sets of menus, tracking and plans, and a proxy to own. Set an end date before you start. If part of the site should stay on a second system for good, make that a decision with an owner.

Can I do this myself?

A developer who has worked with proxies and DNS can. Marketing teams can run the batch checks in this guide, but someone technical has to own the proxy. If you want a second opinion, send us your site and tell us which parts cannot move yet.

Keep going

- Site Move Playbook
- Move a site in stages (short post)
- What is a reverse proxy?
- Redirect map tool
- Redirect map template
- When to move from Webflow to Astro
- Webflow to Astro
- Astro and Sanity
- Tracking setup
- All our services

Planning a staged move? Book a free 20 minute call

Kookies. www.kookies.io. hello@kookies.io