WUNDERLANDMEDIA

The WordPress to Astro Migration Checklist

The nine failure points the official guides skip, and the exact order to work through them.

By Kemal | wunderlandmedia.com

9	400+	1
Failure Points	Redirects	Checklist

1 Pre-Migration Inventory

Spend Day One Inventorying, Not Coding

The tooling for the conversion is fine. The tooling for finding out what actually exists is a crawl, some logs, and a slow afternoon. Every site I have migrated had something on it nobody mentioned.

CAPTURE THE SITE AS IT IS TODAY

- [ ] **Crawl the live site and save rendered HTML for every URL**
  Do this before you touch anything. It is your reference for builder pages, and your test list after launch.
- [ ] **Pull Search Console's Pages report**
  Every URL that ever received an impression, including ones the crawler cannot reach.
- [ ] **Pull 90 days of server access logs**
  Catches orphaned URLs with direct traffic and hotlinked files no page references.
- [ ] **Merge all three lists into one CSV**
  The crawl misses orphans. Search Console misses zero-impression URLs. Logs catch both.

THE PLUGIN AUDIT: WHAT EACH ONE SILENTLY DOES

- [ ] **List every active plugin and write down its front-end job**
  Not the plugin name. The behaviour that disappears when the plugin does.
- [ ] **SEO plugin: schema, canonicals, meta, XML sitemap**
  Four separate jobs you now own. Yoast or Rank Math was generating all of them invisibly.
- [ ] **SEO plugin redirect manager**
  Old redirects from previous URL changes live in here and still carry link equity. Export them.
- [ ] **Search, related posts, breadcrumbs**
  All three were request-time database queries. All three are now your code.
- [ ] **Cookie consent, analytics injection, social share buttons**
  Small individually, but each one is a ticket nobody scoped.
- [ ] **Anything on a cron schedule**
  WP-Cron jobs: digest emails, feed imports, cache warmers, backup runs. They stop silently.

STRUCTURE: COUNT BEFORE YOU SCOPE

- [ ] **Count custom post types**
  Each one is a decision, not a conversion. Six entries becomes a data collection; eighty with an archive becomes routes and pagination you own.
- [ ] **Count custom taxonomies and their terms**
  Every term generated an indexed archive URL that is about to 404.
- [ ] **List every ACF field group, including repeaters and flexible content**
  This is the item most likely to double your estimate.
- [ ] **Record form definitions, notification recipients and integrations**
  The visible form is five fields. The plumbing behind it is more.
- [ ] **Record user roles and who actually logs in**
  A static site has no wp-admin. Decide where editors go before launch day, not after.
- [ ] **Save menu structure and widget areas**
  Neither is in post_content, so neither is in the export.
- [ ] **Note the permalink setting and trailing slash behaviour**
  Both drive the redirect map in section 4.

The Discovery Gap

On one of three sites, the "simple brochure site" turned out to have four custom post types nobody had mentioned. Count them yourself. Do not take the brief's word for it.

2 Content Export: What the Tool Captures

1 field `wordpress-export-to-markdown` reads `post_content`. Anything not in that field does not exist as far as the export is concerned. That is a WordPress XML limitation the tool inherits, not a bug.

WHAT YOU ACTUALLY GET

- [ ] **One Markdown file per post, with frontmatter**
  Clean, predictable, and genuinely good at this job. Run it first.
- [ ] **Attached images downloaded, references rewritten**
  Saves a real chunk of manual work for a normal blog.
- [ ] **Drafts, pages and custom post types as separate files**
  The bodies come across. The structure around them does not.
- [ ] **Categories and tags**
  The two built-in taxonomies survive. Custom ones do not.

WHAT IT SILENTLY DROPS

- [ ] **Every custom field value and every ACF group**
  Stored in postmeta, not post_content. See section 3.
- [ ] **Taxonomy terms that are not categories or tags**
  Along with the archive URLs those terms generated.
- [ ] **Menu structure and widget areas**
  Rebuild from the rendered HTML you crawled.
- [ ] **Form definitions and form entries**
  Section 6. Export before teardown.
- [ ] **Comment threads**
  Section 6. Archive the JSON even if you never display them.
- [ ] **Permalink settings**
  The single fact that determines how large your redirect map is.
- [ ] **Any content a plugin keeps in its own database table**
  Invisible to the XML export by definition.

RIGHT MENTAL MODEL

The export is one input among several. Posts come from the tool, landing pages from rendered HTML, structured data from the REST API.

WRONG MENTAL MODEL

"The export ran clean, so the migration is done." The export ran clean because it only looked at one field.

Sanity Check After Export

Diff the file count against the post count in WordPress, then open the five longest and five shortest files. The short ones tell you where a builder or a custom field was doing the work.

> *"The posts and pages were never the hard part. The hard part was everything the plugins were doing quietly in the background, and the several hundred URLs that existed on the old site that nobody had a list of."*
>
> **Kemal, Wunderlandmedia**

3 Page Builders and Custom Fields

IDENTIFY WHAT EACH BUILDER DOES TO POST_CONTENT

Builder	Where the layout lives	What a clean XML export produces
Divi	Shortcodes in `post_content`	**Shortcode text, verbatim**
WPBakery	Shortcodes in `post_content`	**Shortcode text, verbatim**
Elementor	JSON in postmeta	**Nearly blank page**
Gutenberg / classic	`post_content`	**Usable Markdown**

```
post_content for a WPBakery page, as exported:

[vc_row][vc_column][vc_column_text]
Our Services
[/vc_column_text][/vc_column][/vc_row]

Rendered DOM for the same page:

<h2>Our Services</h2>
```

Do Not Buy an Automatic Builder Converter

There is no reliable automatic converter here, and I would be suspicious of anyone selling you one. A heading is an `h2` in the DOM even when it is a shortcode in the database. Work from the DOM.

RENDER-THEN-SCRAPE

- [ ] **Render every URL in a real browser and save the HTML**
  Same crawl artefact from section 1. Shortcodes and Elementor JSON are already resolved for you.
- [ ] **Blog posts: convert the rendered article body to Markdown**
  Scope the conversion to the article container, not the whole page.
- [ ] **Builder landing pages: rebuild as Astro components**
  Do not convert. Use the rendered HTML as the visual reference and write real components.
- [ ] **Budget honestly: rebuilding beats parsing**
  Five or ten marketing pages by hand is a day. Fighting an Elementor JSON parser is a week and the result is worse.

ACF IS SCHEMA, NOT CONTENT

- [ ] **Expose ACF over the REST API with `show_in_rest`**
  Pull structured data over the API instead of reconstructing it from the XML export.
- [ ] **Write it as JSON or YAML into a content collection**
  Then give the collection a Zod schema so the data is typed on the Astro side.
- [ ] **Let the build fail loudly on missing fields**
  A failed build beats an empty div shipped to production.
- [ ] **Treat repeaters and flexible content as the hard case**
  Flexible content is a per-page list of arbitrary blocks with arbitrary shapes. Reconstructing it from XML is possible but miserable.
- [ ] **Decide each CPT: data collection or content collection**
  Six team members is data. Eighty case studies with an archive is routes plus pagination you now own.
- [ ] **Map every custom taxonomy term to a redirect**
  Those archive URLs were indexed. They go in the map in section 4.

```
postmeta keys behind one ACF flexible content field:

sections_0_hero_title
sections_1_gallery_images_0_image
sections_1_gallery_images_1_image
```

4 The Redirect Map

If You Take One Thing From This

The redirect map is the migration. Everything else is rendering. WordPress generates far more indexed URLs than the page count suggests.

EVERY URL CLASS TO ACCOUNT FOR

- [ ] **Permalink structure change**
  If the old site used /2019/03/post-title/ and the new one uses /post-title/, every single post URL changes.
- [ ] **ID-based URLs: `/?p=123`**
  Every post is reachable at its ID URL forever, and those get shared, bookmarked and linked to.
- [ ] **Attachment pages**
  WordPress creates a page per uploaded file. They get indexed, they rank for image queries, and there can be thousands. WordPress 6.4 added an option to redirect them to the file itself.
- [ ] **Category, tag and custom taxonomy archives**
  Plus author archives and date archives.
- [ ] **Paginated archives**
  /blog/page/2/, /category/news/page/3/, on and on.
- [ ] **Feeds**
  /feed/, /comments/feed/, per-category feeds. Real subscribers live on these.
- [ ] **`sitemap_index.xml` and its child sitemaps**
  Whatever the SEO plugin generated is a URL Google still requests.
- [ ] **Trailing slashes**
  WordPress ends URLs with a slash by default. Astro's trailingSlash setting and your host's behaviour both get a vote, and they do not always agree. Decide once.
- [ ] **Comment pagination and `#comment-1234` anchors**
  Those were URLs too.
- [ ] **Redirects already in the SEO plugin's redirect manager**
  Merge them into the new map. Do not lose them.

Astro's Built-In Redirects Are Not 301s on a Static Build

On a fully static build with no adapter, Astro's `redirects` config writes an HTML page containing a `<meta http-equiv="refresh">` tag. The docs say plainly that it "does not support status codes." Google will usually follow a meta refresh, but it is not a permanent redirect and it is not what you want for a few hundred URLs carrying rankings. Do redirects at the edge.

CLOUDFLARE PAGES `_redirects` LIMITS

Limit	Value	What it means in practice
Static redirects	**2,000**	Sounds enormous until you have 1,400 attachment pages
Dynamic (wildcard) redirects	**100**	Where the leverage is. Spend them deliberately
Combined total	**2,100**	Beyond this you need Bulk Redirects
Per line	**1,000 characters**	Long query-string rules can hit this

```
_redirects — one dynamic rule beats 1,400 static lines

/wp-content/uploads/*   /wp-content/uploads/:splat   301
/?p=123                 /post-title/                 301
/2019/03/post-title/    /post-title/                 301
/category/news/page/*   /blog/                       301
/feed/                  /rss.xml                     301
```

Map Hygiene

Keep the whole map as a CSV in the repo, generate the host's redirect file from it at build time, and never hand-edit the generated file. Test it against the crawl list after every deploy.

5 Forms and Comments

WHERE NEW SUBMISSIONS LAND

- [ ] **Pick an endpoint before launch**
  A static build has no PHP to receive a post. That means a form endpoint service, a serverless function, or the host's native form handling.
- [ ] **Rebuild the plumbing, not just the fields**
  Notification recipients, Mailchimp or CRM integrations, conditional logic, hidden UTM fields, spam protection.
- [ ] **Send a real test submission through every path**
  Including the integration, not only the notification email.

WHERE THE OLD SUBMISSIONS ARE

Plugin	Stores entries?	Action before teardown
Gravity Forms	**Yes, in the WP database**	Export to CSV from Forms > Import/Export
WPForms	**Yes, in the WP database**	Export to CSV
Contact Form 7	**No, nothing by default**	Check for Flamingo or a CFDB add-on

The Contact Form 7 Trap

Unless someone installed Flamingo or a CFDB add-on, every submission that site ever received exists only in whatever inbox got the notification email. If the client asks for their old enquiries after you have decommissioned the server, there is nothing to give them. Check which case you are in while the site is still up.

Export-Before-Teardown Rule

There may be years of leads sitting in a form plugin that nobody has looked at since 2021. Export them all to CSV before the database goes away. This is a five-minute job that is impossible to redo.

COMMENTS: DECIDE BEFORE YOU EXPORT

- [ ] **Pull the comment count first**
  The number decides the answer. I have had all three answers on three sites.
- [ ] **Near zero: drop them**
  Nobody will notice, and you save yourself an integration.
- [ ] **Real archive with value: migrate to Giscus**
  Backed by GitHub Discussions. Community scripts create one discussion per post using the post URL as the title, then post each old comment into it with the original author and date in a header.
- [ ] **Accept the downgrade knowingly**
  Migrated comments are no longer editable by their authors and are all posted by your token's account. Still better than deleting them.
- [ ] **In doubt: export to JSON and archive it**
  Showing them is a decision you can make later. Deleting the database is not.

Comment URLs Were URLs Too

Comment pagination and `#comment-1234` anchors were indexed alongside the posts. Whatever you decide about displaying comments, those paths still need an entry in the redirect map.

EDITORS AND ACCESS AFTER CUTOVER

- [ ] **Tell every WordPress user where they go instead**
  There is no wp-admin any more. Decide this before launch day, not after the first "how do I edit this" email.
- [ ] **Map each role to a real permission on the new stack**
  Repo access, CMS seat, or nothing at all. Roles that had wp-admin logins but never used them can be dropped.
- [ ] **Write down the publish path in one paragraph**
  Who edits, where, and how long until it is live. Hand it over with the site.

6 Media, Search and the Silent Plugins

MEDIA URLS

- [ ] **Keep `/wp-content/uploads/` intact for existing media**
  It looks silly in a repo with no WordPress in it. It also means a directory of URLs people already link to keeps resolving, with zero redirect rules spent.
- [ ] **Preserve the YYYY/MM path structure**
  WordPress writes uploads to /wp-content/uploads/YYYY/MM/filename.jpg. Flatten it and every old URL breaks.
- [ ] **Copy the generated size variants too**
  thumbnail, medium, medium_large, large, plus whatever sizes the theme registered. Each has its own URL, appears in srcset, and gets indexed.
- [ ] **Put new images in the new structure**
  Only the legacy files stay in the legacy path. Draw the line at launch date.
- [ ] **Crawl for hotlinked assets in the other direction**
  Other sites hotlink your images, Google Images has them indexed, and old newsletters point at them.
- [ ] **Find files no page references but people still download**
  PDFs linked from client emails, images embedded in third-party articles. The access logs will tell you. Nothing else will.

0 rules Redirect budget spent when you keep the uploads path. Compare that with one static line per file for a media library of a few thousand images.

SEARCH

- [ ] **Replace the search box with Pagefind**
  It indexes your built HTML after the build, ships the index with the site, and runs the query in the browser.
- [ ] **Use the Astro integration and budget an hour, not a day**
  Pagefind downloads only the index chunks it needs rather than the whole index.

THE SILENT PLUGINS: REBUILD EACH ONE EXPLICITLY

What the plugin did	Who does it now
Site search	Pagefind, indexed post-build
Related posts	Your code: shared-tag matching at build time
Breadcrumbs	Your layout, plus BreadcrumbList markup if you had it
Schema markup	Your templates, per page type
XML sitemap	An Astro integration, wired into the build
Canonical tags and meta	Your head component
Stored redirects	**Merged into the CSV map, not lost**
Cookie consent, analytics	Explicit scripts you now own
Social share buttons	Static markup, no plugin needed
Cron jobs	Scheduled workers, or dropped on purpose

> **Related Posts Is the One That Catches People**
> On WordPress it was a plugin doing a taxonomy query at request time. On Astro you write the logic yourself. Not hard, but nobody puts it in the scope document because nobody remembers it was a plugin.

7 Launch Day: The Cutover Order

Do These In Order

Every step below depends on the one before it. The exports in particular are irreversible once the old server is gone, so they come first even though they feel like admin.

T-MINUS: WHILE WORDPRESS IS STILL UP

1 **Re-crawl the live site one final time**
Content changed while you were building. Refresh the URL list and the rendered HTML.

2 **Export form entries from every form plugin to CSV**
Gravity Forms, WPForms, Flamingo. Store them outside the server.

3 **Export comments to JSON**
Whatever you plan to do with them.

4 **Export the SEO plugin's stored redirects**
Fold them into the CSV map now, not after cutover.

5 **Take a full database and uploads backup**
Keep it somewhere you will still have access to in six months.

6 **Freeze content edits and tell the client the freeze started**
Anything published after this point has to be redone by hand.

CUTOVER

7 **Regenerate the redirect file from the final CSV and deploy**
Confirm the rule count is inside the host's limits before it ships.

8 **Deploy the new site to its own preview URL first**
Full smoke test on the real build, on the real host, before DNS moves.

9 **Verify uploads resolve at the legacy path on the preview**
Pick ten image URLs straight from the old srcset attributes.

10 **Confirm trailing slash behaviour matches the decision you made**
Astro config and host behaviour both get a vote. Check the actual response, not the config.

11 **Submit a real form and confirm the notification and integration fire**
On the preview, before cutover.

12 **Lower DNS TTL a day ahead, then switch DNS**
Short TTL means a fast rollback if something is badly wrong.

13 **Confirm HTTPS and the certificate on the apex and www**
First thing to break, first thing to check.

14 **Leave the old server running, reachable only by you**
Minimum 30 days. It is the only copy of anything you forgot to export.

Do Not Decommission on Launch Day

Keep the old server for at least 30 days. Every migration surfaces something in week two: an old campaign landing page, a PDF linked from a printed brochure, a custom post type nobody mentioned.

8 Post-Launch Verification

- [ ] **Test every URL from the merged list against the live site**
  Every one, not a sample. Assert the status code and the final destination.
- [ ] **Crawl the new site for 404s and broken internal links**
  Catches links inside converted Markdown that still point at old paths.
- [ ] **Spot-check one redirect per URL class by hand**
  One post, one /?p= URL, one attachment page, one tag archive, one paginated page, one feed.
- [ ] **Confirm redirects return 301, not a meta refresh**
  Check the response headers. A 200 with a refresh tag means the redirect is running in the framework, not at the edge.
- [ ] **Submit the new sitemap in Search Console**
  And keep the old sitemap URL redirecting to it.
- [ ] **Watch the coverage report for four weeks**
  Rising "Not found (404)" or "Page with redirect" counts point straight at gaps in the map.
- [ ] **Re-check server logs after two weeks for surprise URLs**
  The requests you never planned for show up here first.

QUICK REFERENCE: THE NINE FAILURE POINTS

#	Failure point	Phase	Cost if missed
1	Incomplete inventory	Before	**Doubles the estimate mid-project**
2	Export tool reads post_content only	Before	**Silent data loss**
3	Builder pages export blank or as shortcodes	During	**Pages rebuilt twice**
4	ACF and CPT structure not modelled	During	**Schema rework late**
5	Redirect map incomplete	During	**Lost rankings and traffic**
6	Framework redirects instead of edge 301s	During	**Meta refresh, no status code**
7	Form entries never exported	Before	**Unrecoverable lead history**
8	Media URLs moved	During	**Broken hotlinks and image rankings**
9	Silent plugins not replaced	During	**Missing search, related posts, schema**

> *"The bit I keep underestimating is not technical. It's the discovery. Every site I've migrated had something on it nobody mentioned: a custom post type, a landing page from an old campaign that still gets traffic, a PDF linked from a printed brochure."*
> **Kemal, Wunderlandmedia**

If You Only Have One Day

Spend it inventorying and write no code. Crawl the site, merge the URL lists, count the custom post types, export the form entries. It feels like stalling. It isn't.

Read the Full Post

wunderlandmedia.com/wordpress-to-astro-migration-problems

Want a sanity check on how deep a specific migration goes before you commit to it? That's exactly what I do. wunderlandmedia.com