

A site migration can change its domain, URL structure, content management system, rendering framework, or several of these at once. Each change creates a different kind of SEO risk. A domain move can break redirects and canonical signals; a replatform can alter templates, internal links, metadata, or how search engines access content. When both happen together, small implementation gaps can become widespread losses in crawlability, indexation, and organic traffic.

A reliable migration audit is more than a list of best practices. It combines forensic analysis of the current site with developer-ready specifications, testable acceptance criteria, and oversight before and after launch. This guide explains how to audit a domain migration SEO project or replatform SEO project in a way that supports implementation—not just a presentation.

Start with the right kind of migration audit

A checklist audit asks whether a site has redirects, canonicals, XML sitemaps, and analytics. Those questions matter, but a “yes” answer does not confirm that the redirects map to the right destinations, that canonicals are consistent, or that analytics measures the new property correctly.



A forensic technical SEO audit investigates how the site actually behaves. It compares crawl data, server responses, rendered pages, internal links, traffic, and logs to identify risks specific to the existing platform and the proposed build. It asks what changed, which URLs matter, how search engines encounter the new site, and how the team will verify fixes.

Checklist audit Forensic migration audit Confirms that a redirect plan exists Tests URL-level mappings, redirect chains, destination relevance, and response codes Checks that the site has an XML sitemap Validates sitemap URLs against canonicals, indexability, status codes, and market or language rules Checks that analytics is installed Verifies event continuity, property configuration, landing-page reporting, and launch annotations in GA4 Confirms that pages can be crawled Compares HTML, rendered output, crawl paths, and server logs to identify practical access problems

Use a checklist to ensure coverage. Use forensic analysis to decide what to do, how to prioritize it, and what evidence will count as done.

Build the audit around migration scope

Before reviewing URLs, document the changes. Record whether the project includes a domain change, URL redesign, CMS replacement, JavaScript framework change, international site consolidation, or hosting and infrastructure work. Also identify what is staying the same. A domain-only move has different risks from a new CMS with client-side rendering and changed URL patterns.

Agree on owners, milestones, and rollback constraints with product, engineering, analytics, and SEO. Keep one source of truth for the URL mapping and issues. Slack can support fast coordination, but decisions, specifications, and acceptance evidence should also live in tickets or project documentation. A Slack message is not a durable redirect specification.

Define acceptance criteria before implementation

For every significant SEO requirement, state how it will be tested. Examples include:

- Every priority old URL returns a single 301 redirect to its closest relevant live destination, with no redirect loop or multi-hop chain.
- Every indexable destination returns a 200 response, has a self-referencing canonical, and is not blocked by robots directives.
- All internal links on the new site point directly to the intended HTTPS destination rather than relying on redirects.
- Important page content and links are present in rendered output and accessible to the agreed crawler test.
- GA4 records page views on the new site, and the team can compare post-launch organic landing pages with the pre-migration baseline.

Criteria should be specific to the site. Define thresholds for priority URL coverage, acceptable redirect behavior, and critical template checks rather than relying on vague statements such as “SEO is ready.”

Migration audit checklist: pre-launch

1. Establish a dependable baseline

Build a source URL inventory from more than one place. Combine a crawl of the live site with XML sitemaps, analytics landing pages, search performance data, CMS exports, backlink data, and server logs where available. No single source contains every valuable URL: crawlers can miss orphaned pages, analytics can miss low-traffic assets, and sitemaps may be incomplete.

- Identify high-value URLs by organic visits, conversions, backlinks, impressions, and strategic importance.
- Capture current status codes, indexability, canonical tags, titles, headings, structured data, and internal links.
- Record language and country targeting, including hreflang relationships where applicable.
- Save a crawl and analytics baseline with a clear date range, so post-launch comparisons are meaningful.

Reporting products such as Reportz.io can help teams keep important performance measures visible, while audit providers such as SEO-Audits.com or agencies including Four Dots may support assessment and delivery. Regardless of who conducts the work, the output should identify affected URLs, evidence, owners, and testable actions—not just general recommendations.

2. Validate URL mapping and redirect behavior

For a domain migration SEO plan, map each old URL to its new equivalent. Do not redirect every retired page to the homepage: that can confuse users and search engines and discard the relevance of the original page. If a direct equivalent does not exist, select the closest useful destination or return an appropriate not-found response.

- Include URLs with and without trailing slashes, uppercase variants, parameters, legacy subdomains, and common file extensions where relevant.
- Test redirects in batches and inspect representative high-value URLs manually.
- Confirm that redirects are server-side where practical and return the intended permanent redirect status.
- Check for chains, loops, incorrect protocol or hostname versions, and redirects to non-indexable destinations.
- Update internal links, canonicals, hreflang references, structured data URLs, and sitemap entries to the final URL.

3. Audit the new platform and templates

For replatform SEO, compare equivalent page types—not only a few sample URLs. Review product, category, article, location, search, and other templates that matter to the business. Check whether the CMS preserves editable metadata, heading structure, indexation controls, pagination behavior, and structured data. Confirm that filters and faceted navigation do not create uncontrolled URL combinations.

Here's what kills me: test staging with the same crawl settings planned for production. Check authentication or IP restrictions that may prevent a crawler from reaching it, while ensuring staging remains excluded from search engines. Do not treat a staging noindex as a substitute for launch controls; confirm precisely how it will be removed or replaced at release.

4. Analyze JavaScript rendering and crawl behavior

When a new framework renders content with JavaScript, compare the initial HTML response with the rendered page. Check whether key text, links, canonical tags, metadata, and structured data appear at the right stage. A browser showing a complete page does not prove that every search crawler will receive the same content or discover its links efficiently.

- Test priority templates with JavaScript disabled and enabled, and inspect both raw HTML and rendered DOM.
- Verify that navigation and pagination expose crawlable links, not only click handlers.
- Check for hydration errors, blocked scripts or APIs, delayed content, and consent overlays that obscure content.
- Confirm that server-side rendering or prerendering delivers critical content reliably if the architecture depends on it.
- Test canonical and robots directives for consistency between the server response and rendered output.

Record the tool, settings, URL, and expected result for each rendering test. That makes the finding reproducible for developers.

5. Use server logs to understand crawl allocation

Server log forensics shows which URLs verified search engine bots actually request, how often they return, and what responses they receive. This is especially useful on large sites where crawl budget allocation matters: search engines have limited time and resources to discover and refresh URLs, and wasteful URL spaces can distract from important pages.

Analyze a representative log period and, where possible, distinguish verified bots from user agents that merely claim to be Googlebot or another crawler. Group requests by template, status code, URL pattern, and response time. Look for repeated crawling of parameter combinations, redirecting legacy URLs, soft errors, slow endpoints, and important pages that receive little or no crawl activity.



Use the findings to prioritize fixes. For example, reduce crawlable paths into low-value filter combinations, correct internal links that send crawlers through redirects, and ensure important pages are linked from relevant, accessible sections. Crawl budget is not a universal explanation for weak performance; log evidence should show whether crawl allocation is a material issue for this site.

Turn findings into developer-ready specifications

Each ticket should explain the problem, affected patterns or URLs, expected behavior, implementation notes, test method, and acceptance criteria. Include examples of current and desired output. For a redirect defect, provide sample source and destination URLs, the required status code, and whether parameters should be preserved. For a rendering issue, specify which content or links are missing from the response or rendered page.

Separate blockers from enhancements. Launch blockers include widespread incorrect redirects, accidental noindex directives, broken canonical rules, unavailable priority content, and analytics failures that prevent meaningful measurement. Lower-priority refinements can be scheduled after launch if they do not undermine crawlability, indexation, or essential reporting.

Implementation oversight and launch-day checks

Do not assume that an approved specification was seo-audits.com implemented as written. Re-crawl staging after fixes, retest redirect rules against the full mapping, and compare representative rendered templates. For high-risk projects, test a production-like environment and rehearse the release sequence: DNS, certificates, redirects, robots rules, sitemaps, analytics, and monitoring.

1. Confirm the live hostname, HTTPS behavior, and preferred domain are consistent.
2. Verify that staging restrictions are removed from production and that production is not accidentally blocked.
3. Test priority old URLs and new destinations immediately after release.
4. Check the live robots.txt, XML sitemaps, canonicals, hreflang, and internal navigation.
5. Verify GA4 page-view collection and key events, and annotate the migration date for reporting.

6. Run a focused crawl of priority templates and review server responses for unexpected errors.

Assign a named owner to each check. Set up a Slack launch channel for rapid incident coordination, but log defects in the ticketing system with evidence and severity so they are not lost in the conversation.

Post-launch monitoring: things that break after launch

Monitor daily at first, then adjust frequency as the site stabilizes. I remember a project where I learned this lesson the hard way.. Compare like-for-like periods and account for seasonality, campaign changes, and tracking differences. Do not treat a temporary fluctuation as proof of failure—or assume recovery without checking technical evidence.

- Track organic landing-page sessions, conversions, and key events in GA4 alongside search performance and indexation data.
- Review server logs for Googlebot response codes, crawl changes, and unexpected demand on low-value URLs.
- Check for 404 spikes, redirect failures, canonical mismatches, missing content, and new internal links to old URLs.
- Compare sitemap URLs against live indexable pages and resubmit updated sitemaps when appropriate.
- Inspect major template changes and newly reported issues, rather than relying only on aggregate site metrics.

Common post-launch failures include a redirect rule deployed to the wrong host, a staging noindex carried into production, JavaScript assets blocked by a new content security policy, analytics tags firing twice, or a CMS default that changes canonical tags across an entire template. These are reasons to keep a running “things that break after launch” list and assign owners before the release begins.

Make the audit actionable

A strong migration audit connects evidence to implementation and verification. It covers the site’s actual architecture, URL inventory, rendering behavior, and crawler activity; it produces specifications developers can build from; and it defines how the team will know each fix is complete. Whether the project is a domain move, a replatform, or both, the goal is not a perfect-looking checklist. The goal is a controlled change with clear acceptance criteria, accountable owners, and enough monitoring to find and fix problems quickly.