Which questions about free migrations and launch workflows will we answer — and why these matter to your agency's margins?

Short version at the bar: offering free migrations is a sales sweetener, but if you don't build a repeatable process it eats profit. Below I answer the practical questions I get asked most by other agency owners — the ones that save time on the shop floor, reduce scope fights, and stop developers from doing the same boring work every week.

- What exactly do we mean by free migration and why it should be strategic, not charitable?
- Won't free migrations create hidden cost and scope creep?
- How do you automate deployments and staging so migrations don't become a full-time job?
- When do you invest in custom scripts versus third-party tools?
- What should you plan for next in hosting, CI, and launch tooling?

What exactly is a "free migration" and how does it save billable hours if done right?

Free migration is an offering: you move a client's site from their current place to your managed hosting or new platform at no charge. The trap is treating that phrase literally without process. When you standardize the steps, you turn a costly, bespoke task into a predictable checklist that can be automated and delegated.

Concrete math from an agency I worked with: manual WordPress migrations averaged 6 hours each (diagnostics, DB export/import, media sync, find-replace for URLs, SSL, DNS, cache tuning, post-launch fixes). After standardizing and automating parts of the flow — one-click staging creation from host, WP-CLI imports, rsync with excludes, Git-based deployment hooks — the average became 45 minutes. That's a 5.25-hour saving per site.

If you migrate 10 clients a month, that's roughly 52 billable hours reclaimed. At $100/hr billing value that turns into $5,200 of time you can use for real client work, not tedious surgery.

Aren't free migrations just an invitation to scope creep and unpaid work?

Short answer: only if you let them be. You must define "free" precisely.

Set explicit boundaries in proposals and sales scripts. Common safe rules I use:

- Free migration covers "like-for-like" moves: same CMS, no architecture change, up to X GB of media, and up to Y database rows. Anything beyond is scoped as a migration project.
- Exclude custom integrations: third-party APIs, complex cron jobs, or proprietary plugins that require license transfers are not free.
- Cap free effort in hours (typical: 2 hours for basic, 6 hours for complex). Offer paid migration tiers beyond that with transparent pricing.
- Require client preparation: provide FTP/SSH access, admin creds, and confirm backups before you accept work.

Give the sales team a simple checklist to confirm whether a lead qualifies for free migration. If they skip it, you end up with hidden work and angry devs.

How do I actually automate deployments and staging so migrations are repeatable?

If you're not using Git-based deployment and a staging workflow, you're doing it wrong. Here is the pragmatic process I use and ask every agency to adopt.

Step 1 — Single source of truth

Keep code in Git. If the client's site wasn't in Git, migrate the codebase into a repo during the first migration and make that repo the deployment source going forward. For WordPress, split theme/plugin code into a repo and manage WP core and plugins with Composer or a plugin manager where possible.

Step 2 — Staging environments that match production

Use ephemeral or persistent staging on your host (Pantheon, WP Engine, Flywheel, or a Kubernetes/Container setup). The point is a staging environment that mirrors production — same PHP, same DB engine, same caching. That removes the "works on my machine" excuses.

Step 3 — CI builds and deployment hooks

Set up a CI pipeline (GitHub Actions, GitLab CI, or CircleCI) that runs on push to main:

- Install dependencies (Composer, npm)
- Run tests and code linters
- Build assets (CSS/JS)
- Push built artifacts to the target host or trigger the host's deploy hook

For WordPress, the pipeline should also generate a deployable tarball and upload it to staging via SSH or the host API.

Step 4 — Database and media sync automation

Use tools and scripts for DB export/import and media sync. Options:

- WP-CLI: wp db export/import and wp search-replace — it handles serialized data and is fast.
- rsync for wp-content/uploads with excludes for cache folders.
- WP Migrate Pro for push/pull with serialized support and media sync. Paid, but speeds things up.

Put these in the CI or in a deployment script you run once per migration. Make each step idempotent where possible so if something fails you can re-run safely.

Step 5 — DNS and SSL automation

Lower the DNS TTL before launch to 300 seconds, switch, and raise TTL afterwards. Use Let's Encrypt for SSL certs and automate issuance via ACME clients or your host's API. Document DNS rollback steps and keep access to the client's registrar credentials in a secure vault.

What are the staging and pre-launch checks you can't skip without paying later?

Here's the checklist I force every team member to follow before handing off a site from staging to production. It takes 15-30 minutes but prevents hours of firefighting after launch.

1. Functionality smoke test: pick key user flows (login, checkout, form submission) and test them end to end in staging.
2. Performance sanity: enable prod caching, run Lighthouse or WebPageTest on staging, verify no 500s or slow DB queries.
3. Search-replace validation: ensure URLs were replaced correctly and serialization is intact (use WP-CLI or WP Migrate Pro logs).
4. Assets and CDN: confirm images served from CDN, asset fingerprints updated, no console 404s.
5. SSL and mixed content: verify all resources load over HTTPS.
6. Backup snapshot: take a host snapshot or DB dump before changing DNS.
7. Monitoring hooks: add uptime and error tracking to the site before launch (UptimeRobot, Sentry, New Relic lightweight agent).

Tip: keep a template email for the client with required post-launch tasks and expected downtime notes. That reduces support calls in the first 24 hours.

Does offering free migrations mean I should stop charging for launch support?

No. Free migration is a lead tool. Launch support, SLA, and ongoing maintenance are separate sale items. Think of free migration as the front door — you still have to upsell optimization, security hardening, and integrations.

Offer a "launch https://rankvise.com/blog/best-hosting-companies-for-web-design-agencies/ comfort" add-on: 30 days of follow-up fixes, 24/7 emergency support for the first 72 hours, and a performance tuning session. Price it, and you'll find clients buy it because they want peace of mind.

When should you write custom migration scripts instead of using off-the-shelf tools?

Use off-the-shelf tools when the site is standard: WordPress, same plugins, under a few GB of media, and no external integrations. Build custom scripts when:

- Data transforms are required — e.g., moving from a custom database schema to a new CMS.
- Serialized or binary blobs need special handling beyond simple search-replace.
- You have frequent, high-volume migrations and need repeatable automation (e.g., migrating 50+ sites into a platform).

Custom scripts pay back when they eliminate repeated manual steps. Example: an agency that migrated 120 sites built a script that automated DB exports, search-replace, and media rsync. Build time: 2 developer-weeks. Savings: 120 * 5 hours = 600 hours. Payback happened in two months.

Should I hire a developer for this or can project managers run migrations?

Split the work by risk. Project managers can handle checklist-driven moves and passive tasks (credential gathering, scheduling, DNS coordination). Developers should own templating, automation, and any custom data work.

Train non-dev staff to run standardized scripts via a web UI or a small CLI wrapper that hides complexity. That lets your engineers focus on building and improving the pipeline rather than executing per-client steps.

What launch-day failures should we plan for and how do we rollback quickly?

Most launch failures fall into four buckets: DNS delays/misconfiguration, missing assets or media, DB mismatch, and performance/caching issues. Your rollback plan:

- Keep host-level snapshots or a database dump you can re-import in minutes.
- If DNS is the problem, switch to the pre-launch IP (you kept it) and lower TTL so you can flip quickly.
- Have rollback commands/scripts: a one-click script that reverts the live site to the pre-launch snapshot removes guesswork.
- Communicate with the client immediately with a clear ETA and the steps you're taking.

What changes in hosting and CI should agencies plan for in the next 12-24 months?

Short list of things to watch and prepare for so you don't get surprised.

- Host APIs and automation will grow. Choose hosts with a strong API so your scripts can create staging, issue certs, and trigger deploys programmatically.
- Edge compute and CDN-level logic will become more common. Expect builds that push static pages to the edge and dynamic parts to serverless functions.

- CI providers will expand free tiers and integrated artifact hosting — make builds deterministic so you can cache artifacts and avoid re-building every deploy.
- Security and data privacy rules will tighten. Keep migration logs, get client consent for handling credentials, and store secrets in a vault (HashiCorp Vault, GitHub Secrets).

Plan one small R&D sprint every quarter to test a new hosting feature or a deploy hook. Keep a short list of things that actually reduce time-to-deploy, not shiny features.

What tools and resources should every agency include in their migration toolbox?

Here's a compact list you can copy into your internal playbook.

Category Tools Why Version control Git, GitHub/GitLab Source of truth for code and deployments CI/CD GitHub Actions, GitLab CI, CircleCI Automated builds, tests, and deploys CMS migration WP-CLI, WP Migrate DB Pro, Drush (Drupal) Fast DB and serialized data handling File sync rsync over SSH, rclone Reliable uploads, resume, excludes Hosting Pantheon, WP Engine, Flywheel, DigitalOcean, AWS Choose hosts with strong staging and APIs Monitoring Sentry, UptimeRobot, New Relic Catch issues fast after launch DNS/SSL Cloudflare, Let's Encrypt Fast DNS management and automated certificates

More questions you should ask internally before promising "free migration"

- Who owns the migration playbook and who updates it?
- What is the maximum site size and complexity we'll accept for free?
- How do we log time and measure the real cost of migrations so we can adjust the policy?
- What training and tooling does a non-dev teammate need to run basic migrations safely?
- When will we escalate a migration to a paid project?

Final, blunt advice from the bar — what most agencies miss

Clients buy certainty, not free labor. When you offer free migrations, you're selling a promise: we'll move your site and it won't break. Make that promise credible by automating the repetitive bits, capping the free scope, and building a rollback plan. The magic isn't the "free" word — it's turning migrations into a repeatable, delegated task so senior people can work on value-added strategy.

If you're serious about reclaiming billable hours, pick one site type (WordPress e-commerce, brochure, or headless CMS), automate the full flow end-to-end, and document it until any competent PM can run it. Then scale the same approach to other site types.

Want the checklist and a sample deploy script outline I use? Ask and I'll drop a copy you can paste into your playbook.