What to check before putting a site live

The list we apply to our own deployments, including this site's.

> **Who it is for:** Your site is ready to ship and you want to know what can be checked before rather than after.

Backups, before anything else

A backup that has never been restored is not a backup: it is a file you are hoping something about. The only check that counts is restoring the copy into an empty database and comparing row counts, table by table.

Until that restore has been done once, assume you have no backup. This is not theoretical: the question arises the day someone deletes data by accident, and that day it is too late to discover the archive was empty.

- ✓ An automatic backup runs, at a frequency matching what you accept losing.
- ✓ A full restore has been performed at least once, and row counts compared.
- ✓ Archives live somewhere other than the server they back up.
- ✓ Someone is alerted when a backup fails — without an alert, failure goes unnoticed for months.

Secrets

A password or API key that enters a code repository once stays there, even if deleted in the next commit: the history keeps it. Removing it means rewriting that history, which breaks everyone else's work.

The check to run before going live is simple: search the full history, not just the current files. And rotate any key found there, rather than hoping nobody saw it.

- ✓ No secret in the repository, history included — editor backup files (“.env.bak”) are the classic trap.
- ✓ Repository access keys are read-only if they only serve to deploy.
- ✓ Passwords created during installation have been changed, and the files holding them deleted.
- ✓ Secrets are entered masked, never passed as command arguments: the process list is readable by other accounts.

What takes a minute to check and costs a lot to miss

None of these needs a special tool. They are nonetheless the most frequently missed, because they produce no visible error: the site works, it merely does not do what you think.

- ✓ The certificate is valid and renews itself — check the expiry date, not just the padlock.
- ✓ The site answers with and without “www”, and one redirects to the other.
- ✓ robots.txt does not forbid indexing. A forgotten staging directive makes the site invisible without breaking anything.
- ✓ Pages have titles and descriptions that differ from one another.
- ✓ A submitted form actually arrives: test it from an address outside the organisation.
- ✓ Error messages show visitors no file paths and no stack traces.
- ✓ A 404 page exists and offers a way onward.

Accessibility

Automated tools catch some problems — insufficient contrast, images without descriptions, fields without labels — and that is already a lot. They do not catch whether the reading order makes sense, or whether a form stays usable by keyboard alone.

The check with the best return, and it costs nothing: go through the whole site with the tab key, no mouse. If you cannot see where you are, or you get stuck in a menu, a keyboard user will too.

A conformance statement, on the other hand, requires a human audit. Producing one on the strength of an automated tool alone means declaring what you have not verified.

Knowing something is wrong

Most outages are not detected by whoever runs the site but reported by a customer — often hours later, sometimes never, the person having simply gone elsewhere.

A probe that calls one address every few minutes and alerts on failure solves most of the problem. It is not an expensive arrangement, and it is the difference between finding an outage yourself and being told about it.