ninegravity

NINEGRAVITY.COM

WORDPRESS PERFORMANCE · 2026 EDITION

WordPress *Performance* Checklist

A practical, data-backed checklist for every WordPress site owner, developer and agency. Beginner-friendly, expert-level results.

53%
of users abandon sites that take over 3s to load

7×
faster TTFB once page caching is enabled

44%
of WP sites pass mobile Core Web Vitals

PROFESSIONAL WORDPRESS
PERFORMANCE & MAINTENANCE

Why WordPress performance *matters*.

Performance is now a business metric, not just a technical one.

Google uses Core Web Vitals as a direct ranking factor. Users abandon slow sites and rarely return. The good news: because WordPress gives you full control over hosting, caching, code, and content delivery, it is the platform where smart optimization makes the biggest measurable difference.

CORE WEB VITALS: YOUR PERFORMANCE TARGETS

METRIC	GOOD	NEEDS WORK	POOR
LCP: Largest Contentful Paint	Under 2.5s	2.5s – 4.0s	Over 4.0s
INP: Interaction to Next Paint	Under 200ms	200 – 500ms	Over 500ms
CLS: Cumulative Layout Shift	Under 0.1	0.1 – 0.25	Over 0.25
TTFB: Time to First Byte	Under 200ms	200 – 800ms	Over 800ms

Google officially calls TTFB “good” at under 800ms, but high-performing WordPress sites target LCP under 1.5s, INP under 100ms, and TTFB under 200ms. These are the thresholds that separate fast sites from great ones.

HOW TO USE THIS CHECKLIST

Work through each step in order: they are ordered by impact and dependency. Hosting and caching (Steps 2 and 3) have the largest impact and must be addressed before any frontend optimization, because no amount of image compression can overcome a one-second server response time. Each item has a checkbox so you can print and work through it.

START HERE	Do this first: highest impact, biggest score change.
HIGH IMPACT	Significant, measurable improvement expected.
LCP / CLS FIX	Directly improves a specific Core Web Vitals score.
CRITICAL	Must not be skipped: skipping causes problems.
VERIFY	A check to confirm the work above is actually live.

STEP 01 · BASELINE

1 Measure *first*: know your baseline.

Run these tools before making any changes.

The single most common mistake in performance optimization is making changes without first measuring. Without a baseline, you are guessing which issues matter most for your specific site. Every site has different bottlenecks: one site's biggest issue is images, another's is TTFB, another's is JavaScript. Measurement tells you exactly where to start and how to prove improvement.

YOUR FOUR ESSENTIAL MEASUREMENT TOOLS

TOOL	WHAT IT SHOWS	URL	COST
Google PageSpeed Insights	LCP, INP, CLS, TTFB with specific elements highlighted. Lab data + real-user field data.	`pagespeed.web.dev`	**Free**
Google Search Console	Site-wide CWV data from real Chrome users. URLs grouped as Good / Needs Improvement / Poor.	`search.google.com/sc`	**Free**
GTmetrix	Full waterfall of every resource loaded. Identifies large files, slow requests, render-blocking.	`gtmetrix.com`	Free / Paid
WebPageTest	Tests from multiple global locations. Reveals TTFB gaps and CDN effectiveness.	`webpagetest.org`	**Free**

HOW TO RUN PAGESPEED INSIGHTS CORRECTLY

01 Go to `pagespeed.web.dev`, enter your URL and click **Analyse**.

02 Click the **"Mobile"** tab first: Google uses mobile-first indexing, so mobile scores affect your rankings.

03 Note your LCP, INP, and CLS scores and whether each is Good, Needs Improvement, or Poor.

04 Scroll to **"Diagnostics"**: PageSpeed names the specific element causing each issue (e.g. which image is causing your slow LCP).

05 Run it on three different pages: homepage, most popular blog post, and main service or product page.

06 Screenshot or copy your scores: you need a *before* number to measure any improvement against.

WHAT YOUR RESULTS ARE TELLING YOU

TTFB over 400ms: Hosting or caching issue. Fix Steps 2 and 3 first. **LCP over 4s:** usually an unoptimised hero image, render-blocking CSS, or slow TTFB. **CLS over 0.1:** images without declared dimensions or dynamic content pushing elements down.
INP over 500ms: heavy JavaScript blocking the browser from responding to clicks.

STEP 01 · ACTION CHECKLIST

Measure first: your checklist

- [ ] START HERE **Run PageSpeed Insights (Mobile tab) on your homepage**
  pagespeed.web.dev: record LCP, INP, CLS and TTFB values before anything else.
- [ ] **Run PSI on your most popular blog post and key service/product page**
  Performance varies significantly by page type: you need all three.
- [ ] **Set up Google Search Console and check the Core Web Vitals report**
  Experience > Core Web Vitals: real user data across your whole site, grouped by URL type.
- [ ] **Run a GTmetrix waterfall report on your homepage**
  Identifies the specific files causing slowness, large payloads and render-blocking resources.
- [ ] **Run WebPageTest from a server location close to your audience**
  Reveals CDN gaps and shows how users in different countries experience your site.
- [ ] CRITICAL **Screenshot all scores as your official baseline**
  Essential: without a before number you cannot measure whether any optimization worked.
- [ ] **Install the Web Vitals Chrome extension for ongoing spot-checks**
  Shows live LCP, INP and CLS values as you browse your own site.

> TIP
>
> Always run PageSpeed Insights in an incognito window to prevent browser extensions from affecting scores. Also test from a fast connection: you want to isolate server-side performance, not your own network conditions.

STEP 02 · INFRASTRUCTURE

2 Hosting & *server* configuration.

The #1 performance factor: no plugin can overcome bad hosting.

Your hosting environment sets a hard ceiling on everything else. A slow TTFB means your LCP cannot be fast, regardless of how well-optimised your images are. This is why WordPress appears to underperform Shopify in aggregate Core Web Vitals statistics: Shopify includes enterprise-grade edge caching on every plan by default. You can replicate the same advantage on WordPress with the right hosting choices, and Cloudflare's free CDN tier is a big part of the solution.

TTFB PERFORMANCE RANGES: WHAT TO TARGET

TTFB RANGE	WHAT IT MEANS	ACTION REQUIRED
< 200ms	Excellent. Managed hosting with CDN edge caching.	**Maintain as-is.**
200–400ms	Good. Quality managed or VPS hosting.	Fine. Add Redis if not present.
400–600ms	Acceptable. Quality shared hosting.	Add Redis object cache and CDN.
600–800ms	Concerning. Missing server-level caching.	**Upgrade plan or switch hosts.**
> 800ms	Infrastructure problem. Plugins cannot fix this.	**New host or server upgrade required.**

Real benchmark data: even the best shared hosts average 395–400ms TTFB. Managed WordPress hosting without edge caching averages 444ms. Budget shared hosting can sit near 790ms. The difference between tiers is larger than any single frontend optimization.

THE BIGGEST FACTORS AFFECTING YOUR TTFB

PHP VERSION

PHP 8.2 / 8.1 are ~18% faster than PHP 7.4 on real WordPress workloads. Many shared accounts still default to 7.4: check yours immediately.

STORAGE TYPE

NVMe SSD reads ~3,500 MB/s, SATA SSD ~550 MB/s, HDD ~100 MB/s. Critical for database-heavy sites with many posts and products.

OBJECT CACHING (REDIS)

Without Redis, WordPress re-queries the database for the same options, menu and widget data on every uncached request. Redis returns them from RAM in microseconds.

SERVER LOCATION

A US server serving European visitors adds 80–150ms of network latency that no amount of optimization can eliminate.

HOW TO CHECK AND UPGRADE YOUR PHP VERSION

01 In WordPress admin, go to **Tools > Site Health > Info**, then expand the “Server” section to see the current PHP version.

02 If it shows 7.x, you need to upgrade immediately.

03 Log in to your hosting control panel (cPanel, Plesk, or your host’s dashboard).

04 Find “PHP Version Manager”, “MultiPHP Manager”, or a similar setting.

05 Select **PHP 8.1 or 8.2** from the dropdown and save.

06 Test your site immediately. Most plugins support PHP 8.x, but verify key pages load correctly.

07 If something breaks, downgrade to 8.0 temporarily while you identify the incompatible plugin.

STEP 02 · ACTION CHECKLIST

Hosting & server: your checklist

- [ ] HIGH IMPACT **Check PHP version in Tools > Site Health: upgrade to 8.1 or 8.2**
  ~18% faster than PHP 7.4, the easiest free server-side win available.
- [ ] **Confirm server uses NVMe SSD storage**
  Affects database query speed: ask your host if it is not clearly stated in your plan details.
- [ ] **Enable HTTP/2 or HTTP/3 on your server**
  HTTP/2 allows parallel file loading; HTTP/3 significantly reduces latency on mobile connections.
- [ ] KEY **Add a CDN with edge caching: Cloudflare free plan or your host’s CDN**
  This is the infrastructure reason Shopify outperforms unoptimised WordPress on CWV (see Step 8).
- [ ] **Consider managed WordPress hosting if currently on shared hosting**
  WP Engine, Kinsta and Cloudways include server-level caching, Redis and CDN in every plan.
- [ ] **Verify your server is in the same region as your primary audience**
  A US server serving a European audience adds 80–150ms network latency on every request.
- [ ] **Enable Brotli or GZIP compression at server level**
  Brotli compresses text assets 15–26% better than GZIP. Most modern hosts support both.
- [ ] **Check server response consistency, not just average TTFB**
  Occasional 1200ms spikes are as damaging as a consistently slow average.

> **TIP: QUICK TTFB TEST, NO TOOLS**
>
> Open DevTools (F12) > Network tab, reload your page in an incognito window, click the first HTML request, and read the “Waiting (TTFB)” value in the Timing panel. This is your raw server response time.

STEP 03 · HIGHEST IMPACT

3 *Caching*: the single biggest win.

Page cache + Redis object cache = 7× faster TTFB.

Caching is the most impactful optimization available to most WordPress sites. Field data from 5.7 million WordPress pageviews (Q4 2025) shows a clear picture: cached pages deliver a median TTFB of 106ms; uncached pages, 723ms. That is a 7× improvement from a single change. Nothing else in this checklist comes close to that ratio of effort to result.

THE HEADLINE NUMBER

106ms cached vs *723ms* uncached median TTFB, a 7× improvement from one change.

THE FOUR TYPES OF CACHING, AND WHAT EACH DOES

CACHE TYPE	WHAT IT STORES	PERFORMANCE GAIN	BEST TOOL
Page caching	Full HTML output of each page: PHP and MySQL skipped entirely on cached requests.	**7× faster TTFB** `106ms vs 723ms`	WP Rocket, LiteSpeed
Object caching	Database query results stored in RAM: no repeated DB lookups for the same data.	**67% less PHP time** `508ms vs 1,542ms`	Redis (via plugin)
Browser caching	Static files stored on the visitor's own device for repeat visits.	Near-instant repeat loads	Plugin / .htaccess
CDN caching	Static assets served from edge nodes physically close to each visitor.	50–200ms TTFB reduction globally	Cloudflare, BunnyCDN

HOW TO SET UP WP ROCKET (RECOMMENDED FOR NON-DEVELOPERS)

01 Purchase WP Rocket at `wp-rocket.me` ($59/year for 1 site): install and activate in WordPress admin.

02 Go to **Settings > WP Rocket**. Under the "Cache" tab, enable **Enable Page Caching**.

03 Under "File Optimization" enable Minify CSS, Minify JavaScript, and Load JavaScript Deferred.

04 Under "Media" enable LazyLoad for images and iframes, and "Add Missing Image Dimensions".

05 Under "Advanced Rules" exclude `/cart/`, `/checkout/`, `/my-account/`, and any logged-in pages.

06 Click "Clear Cache" from the top bar. Cached pages are built automatically as visitors arrive.

07 Re-test TTFB in PageSpeed Insights: you should see an immediate improvement.

STEP 03 · OBJECT CACHING

Setting up Redis object caching

Without object caching, WordPress queries the database for the same options, menu configurations, widget settings and user data on every single request, even if nothing has changed. Redis stores these query results in RAM so they are returned in microseconds. The result is a 67% reduction in PHP execution time (508ms median with Redis vs 1,542ms without).

HOW TO ENABLE REDIS OBJECT CACHING ON WORDPRESS

01 First, enable Redis on your server: contact your host or check your control panel. Redis is included by default on Kinsta, WP Engine, Cloudways and SiteGround.

02 In WordPress admin, go to **Plugins > Add New**. Search for “Redis Object Cache” by Till Kruss.

03 Install and activate the plugin.

04 Go to **Settings > Redis**. You will see a connection status panel.

05 Click “Enable Object Cache”: the status should change to **Connected** (shown in green).

06 Run PageSpeed Insights again: TTFB on uncached pages (admin, search, logged-in views) should drop noticeably.

STEP 03: YOUR CHECKLIST

- [ ] DO THIS FIRST **Install and configure a page caching plugin**
  WP Rocket (easiest, paid), LiteSpeed Cache (best for LiteSpeed servers, free), or W3 Total Cache (free).
- [ ] HIGH IMPACT **Enable Redis object caching via the Redis Object Cache plugin**
  Reduces PHP execution time by 67%, one of the highest-impact free optimizations available.
- [ ] CRITICAL **Exclude WooCommerce cart, checkout and account pages from page cache**
  These pages are user-specific and must never be served from a shared cache.
- [ ] **Set browser cache headers: minimum 1 year for CSS, JS and images**
  Tells visitors’ browsers to store these files locally for fast repeat visits.
- [ ] **Enable GZIP or Brotli compression for all text responses**
  Reduces HTML, CSS and JS file sizes by 70–90% during transfer with no quality loss.
- [ ] VERIFY **Confirm your cache is working using response headers**
  cf-cache-status: HIT or X-Cache: HIT confirms pages are being served from cache.
- [ ] **Configure your CDN to cache static assets at edge nodes globally**
  See Step 8: Cloudflare’s free plan handles this automatically.
- [ ] **Audit autoloaded wp_options data: target under 800KB**
  Autoloaded data loads on every request including cached page builds (see Step 6 for cleanup).

> TIP: VERIFY YOUR CACHE
>
> Open a terminal and run `curl -I https://yoursite.com | grep -i cache`. A working cache returns `cf-cache-status: HIT` (Cloudflare) or `X-Cache: HIT` (WP Rocket / LiteSpeed). “MISS” means the page just loaded uncached, refresh once and it should show HIT on the second request.

STEP 04 · MEDIA

4 Image *optimization*.

Images are 45–65% of total page weight on most WordPress sites.

Images are the largest performance bottleneck on most WordPress sites. They are also the primary driver of LCP scores, since the largest contentful element on most pages is a hero image or banner. Getting images right (format, compression, loading priority and dimensions) has the most direct and visible impact on your Core Web Vitals, particularly LCP and CLS.

IMAGE FORMAT GUIDE: WHAT TO USE IN 2026

FORMAT	SIZE VS JPEG	BROWSER SUPPORT	BEST USE CASE
WebP	**25–35% smaller**	97%+ (all modern browsers)	All photos and general images. Use as your default format.
AVIF	**40–50% smaller**	~85% (Chrome, Firefox, Safari 16+)	Maximum compression. Ideal for new projects with modern audiences.
JPEG	Baseline	100%	Legacy fallback. Use as a WebP fallback where needed.
PNG	Lossless	100%	Logos, screenshots, images requiring transparency.
SVG	Tiny	100%	All icons, logos and illustrations. Always prefer SVG for vector graphics.

HOW TO SET UP AUTOMATIC WEBP CONVERSION AND COMPRESSION

01 Install Smush (free) or ShortPixel (paid, higher quality output) from WordPress.org.

02 Set compression to “Lossy” for photos: 25–40% smaller files with no visible quality difference.

03 Set compression to “Lossless” for logos, diagrams and brand graphics where quality must be exact.

04 Enable WebP Conversion: it automatically serves WebP to all supporting browsers.

05 Run “Bulk Compress” on your entire existing media library, a one-time task for all legacy images.

06 Set maximum upload dimensions in **Settings > Media** to your largest display size (usually 1200–1920px).

THE LCP IMAGE: YOUR MOST CRITICAL TARGET

WordPress does not automatically add `fetchpriority="high"` to your hero image. Without it, the browser treats your most important image like any thumbnail. Adding this single attribute can reduce LCP by 200–500ms. Meanwhile, incorrectly applying lazy loading to this image actively makes LCP worse.

STEP 04 · LCP IMAGE

How to optimise your LCP image specifically

01 Run PageSpeed Insights and click the LCP timestamp in the filmstrip view at the top: the LCP element is highlighted.

02 Find that image in your theme template files (usually header.php, front-page.php, or your page builder's hero block).

03 Add `fetchpriority="high"` directly to the img tag.

04 Add a preload link in your theme's head: `<link rel="preload" as="image" href="hero.webp">`

05 Ensure this image is in your initial HTML source, not loaded by JavaScript, which delays the download.

06 If using a page builder: set the hero block's loading priority to eager, not lazy.

```
<!-- Hero image, optimised for LCP -->
<img src="hero.webp" fetchpriority="high" width="1600" height="900" alt="...">
<link rel="preload" as="image" href="hero.webp">
```

STEP 04: YOUR CHECKLIST

- [ ] LCP FIX **Convert all images to WebP format**
  25–35% smaller than JPEG at equal quality: use Smush, ShortPixel, Imagify or EWWW.
- [ ] **Set up automatic compression on all new image uploads**
  The plugin handles this automatically: no manual compression needed for future uploads.
- [ ] **Bulk compress all existing images in your media library**
  One-time task: run in your chosen image optimisation plugin's bulk processing tool.
- [ ] LCP FIX **Add fetchpriority="high" to your LCP hero image**
  The most direct LCP fix available: tells the browser to prioritise this download above all others.
- [ ] CRITICAL **Remove loading="lazy" from your hero image if present**
  Lazy loading the LCP image is one of the most common LCP mistakes: it directly delays your score.
- [ ] CLS FIX **Add explicit width and height attributes to every img tag**
  Without dimensions the browser cannot reserve space: images loading in push content down, causing CLS.
- [ ] **Enable lazy loading for all images below the fold**
  WordPress adds loading="lazy" since 5.5 automatically: verify your theme has not disabled it.
- [ ] **Serve images at the correct display dimensions using srcset**
  Don't upload a 3000px image for a 400px thumbnail: WordPress srcset handles responsive sizing.
- [ ] **Deliver images through a CDN**
  Cloudflare, BunnyCDN or Cloudflare Images: serves from the nearest edge node to each visitor.

5

STEP 05 · FRONT-END

CSS, JavaScript & *fonts*.

Render-blocking resources delay everything the user sees.

When a browser loads a page, it reads HTML from top to bottom. When it encounters a script tag or a stylesheet link in the head, it stops rendering and waits to fully download and process that file. This is render-blocking. The consequence is that your page appears completely blank during this time, even if your server responded in 150ms. Every render-blocking resource adds its full download time to your LCP.

UNDERSTANDING DEFER VS ASYNC FOR JAVASCRIPT

ATTRIBUTE	WHEN IT DOWNLOADS	WHEN IT EXECUTES	BEST FOR
none	Immediately: pauses HTML parsing	Immediately on download	**Nothing. Always add defer or async.**
defer	In parallel with HTML parsing	After HTML fully parsed	**Most WordPress plugins. Safest default.**
async	In parallel with HTML parsing	Immediately when downloaded	Analytics, chat widgets, independent scripts.

HOW TO IDENTIFY AND FIX RENDER-BLOCKING RESOURCES

01 Run PageSpeed Insights. Look for “Eliminate render-blocking resources” in the Opportunities section.

02 It lists every blocking CSS and JS file with an estimated time saving for removing each one.

03 For JavaScript: in WP Rocket, enable “Load JS Deferred” under File Optimization (LiteSpeed has an equivalent).

04 For third-party scripts: use WP Rocket’s “Delay JS Execution” so they load only on first user interaction.

05 For CSS: enable “Minify CSS” and “Remove Unused CSS” to reduce the size of the blocking stylesheet.

06 To find unused code: in Chrome DevTools, press Ctrl+Shift+P, type “Coverage”, reload: red bars are unused code.

WEB FONT OPTIMISATION

Google Fonts loaded from `fonts.googleapis.com` adds a cross-origin DNS lookup and a render-blocking request to Google’s servers: this alone can delay rendering by 100–300ms. Self-hosting eliminates this: fonts load from your own domain, under your own caching rules, with no external dependency.

01 Install the “OMGF | Host Google Fonts Locally” plugin (free) from WordPress.org.

02 Go to **Settings > OMGF**: it automatically downloads your fonts and serves them locally.

03 Enable “font-display: swap” to show a system font immediately while your custom font loads (prevents FOIT).

04 Preload your primary font file: in WP Rocket’s Preload tab, add the font URL to the preload list.

05 Reduce your font to only the weights you actually use: every extra weight is a separate download.

06 After setup, run GTmetrix and confirm fonts.googleapis.com no longer appears in your waterfall chart.

STEP 05 · ACTION CHECKLIST

CSS, JavaScript & fonts: your checklist

- [ ] **Defer all non-critical JavaScript**
  WP Rocket “Load JS Deferred” or LiteSpeed “JS Defer”: applies defer automatically to all eligible scripts.
- [ ] HIGH IMPACT **Use “Delay JavaScript Execution” for third-party scripts**
  Analytics, chat widgets and embeds load only on first user interaction, not on page load.
- [ ] **Minify CSS and JavaScript files**
  Remove whitespace and comments: 20–40% file size reduction with zero visual change.
- [ ] LCP FIX **Remove unused CSS per page using WP Rocket or Asset CleanUp**
  Page builders load CSS for all modules on every page even if those modules are not used.
- [ ] **Self-host Google Fonts using the OMGF plugin**
  Eliminates the cross-origin DNS lookup and external render-blocking request to Google servers.
- [ ] CLS FIX **Add font-display: swap to all @font-face declarations**
  Shows fallback text immediately while the custom font loads: prevents invisible text.
- [ ] **Preload your primary font file**
  Add via WP Rocket’s Preload section or manually with <link rel="preload"> in the theme head.
- [ ] **Restrict WooCommerce scripts to shop and product pages only**
  WooCommerce loads cart scripts on every page by default: unnecessary overhead on blogs and contact pages.
- [ ] **Use Query Monitor to find plugins loading assets where not needed**
  Query Monitor > Scripts and Styles: note assets on irrelevant pages, then disable with Asset CleanUp.

> TIP: FIND WASTED JAVASCRIPT
>
> In DevTools press F12, then Ctrl+Shift+P, type “Coverage” and Show Coverage. Reload. Each file shows a usage percentage: red portions are downloaded but never executed. A plugin loading 200KB of JS where 80% is unused is costing you 160KB for nothing.

6

STEP 06 · DATABASE

Database *optimisation*.

Database bloat adds latency to every uncached and dynamic request.

Database overhead manifests as increased PHP execution time and appears directly in your TTFB. A bloated WordPress database slows every page that cannot be fully served from cache: admin pages, logged-in user views, WooCommerce dynamic content and search results. On high-traffic or WooCommerce sites, database optimisation can reduce server load by 30–50%.

THE MAIN SOURCES OF WORDPRESS DATABASE BLOAT

POST REVISIONS

WordPress saves a revision on every manual save and autosaves every 60s. A post edited 50 times creates 50 revisions: over years this becomes tens of thousands of rows in wp_posts.

SPAM COMMENTS

Akismet catches spam but stores it in wp_comments. Without regular cleanup, hundreds of thousands of spam rows accumulate.

EXPIRED TRANSIENTS

Plugins store temporary cached data in wp_options as transients. Many never clean up expired entries, and some autoload.

ORPHANED PLUGIN DATA

Deleting a plugin often leaves its tables and wp_options entries behind: dead weight that wastes space.

HOW TO DIAGNOSE AUTOLOADED OPTION BLOAT

Autoloaded options are the most impactful database performance issue: they load on every single request, cached or not, because WordPress needs them to bootstrap. Run this query in phpMyAdmin to check your current size:

```
SELECT SUM(LENGTH(option_value)) AS total_bytes,
       COUNT(*) AS total_options
FROM wp_options
WHERE autoload = 'yes';
-- Under 800,000 bytes (800KB) = acceptable
-- Over 800,000 bytes = performance issue, cleanup needed
-- Over 2,000,000 bytes (2MB) = serious TTFB overhead
```

A real-world case study: reducing a WooCommerce store's autoload table from 2.8MB to 700KB improved TTFB by 28% on uncached requests.

LIMIT FUTURE REVISION ACCUMULATION

Add these lines to `wp-config.php`, just above the line that reads "That's all, stop editing!":

```
// Limit revisions to 5 per post (recommended for active sites)
define( 'WP_POST_REVISIONS', 5 );
// Increase the autosave interval from 60 to 120 seconds
define( 'AUTOSAVE_INTERVAL', 120 );
```

STEP 06 · ACTION CHECKLIST

Database optimisation: your checklist

- [ ] TTFB FIX **Check autoloaded options size: target under 800KB**
  Run the SQL query in phpMyAdmin (Databases > SQL) or use the WP Options Optimizer plugin.
- [ ] **Add a WP_POST_REVISIONS limit to wp-config.php**
  Prevents unlimited future accumulation: do this before continuing content creation.
- [ ] **Clean up all existing post revisions with WP-Optimize**
  WP-Optimize > Database > Post Revisions: bulk delete, keeping only the last 5 per post.
- [ ] **Delete all spam and trashed comments**
  Comments > Spam and Comments > Trash: bulk select all, delete permanently.
- [ ] **Remove all expired transients**
  WP-Optimize handles this automatically on a schedule, or use Advanced DB Cleaner for manual control.
- [ ] **Delete auto-draft and trashed posts and pages**
  Posts and Pages > Trash: accumulated unused drafts bloat wp_posts unnecessarily.
- [ ] **Run OPTIMIZE TABLE on wp_posts and wp_options**
  WP-Optimize > Database > Optimize Tables: reclaims disk space freed by bulk deletions.
- [ ] **Schedule automated weekly database cleanup**
  WP-Optimize can run all cleanup tasks automatically on a schedule: configure once and forget.

> ⓘ TIP: FIND SLOW QUERIES
>
> Install Query Monitor (free, by John Blackbourn) and load any page while logged in as admin. Click “Queries” in the toolbar: it shows every database query, how long it took, and which plugin triggered it. Queries over 0.05s are highlighted red: these are your bottlenecks.

7

STEP 07 · AUDIT

Plugin & theme *audit*.

Every active plugin is a potential performance cost.

The real performance challenge with WordPress is not the core software: it is what gets added on top. A typical site runs 20–40 plugins, each potentially adding JavaScript, CSS, database queries and PHP processing time. A disciplined plugin audit: identifying which plugins add the most overhead and removing or replacing them: often produces larger gains than any technical optimization in isolation, with zero additional cost.

HOW TO AUDIT PLUGIN PERFORMANCE WITH QUERY MONITOR

01 Install “Query Monitor” (free, by John Blackbourn) from WordPress.org and activate it.

02 Visit your homepage while logged in as admin, a dark toolbar appears at the top of the screen.

03 Click “Scripts” to see every JavaScript file loaded, grouped by the plugin that registered it.

04 Click “Styles” to see every CSS file loaded: note plugins loading large stylesheets where they have no function.

05 Click “Queries” to view all database queries sorted by execution time: identify plugins generating slow ones.

06 For each plugin loading assets on pages that don’t need them, use “Asset CleanUp” to restrict it by page type.

BINARY ELIMINATION: FIND THE SLOW PLUGIN IN 5 STEPS

1) Record your current TTFB. 2) Deactivate exactly half your plugins (keep caching active). 3) Re-check TTFB: if it improved, the culprit is in the deactivated group. 4) Re-activate half of that group; if TTFB rises again, the culprit is in that sub-group. 5) Repeat halving until you identify the exact plugin, then replace it or restrict its scope.

LIGHTWEIGHT THEME PERFORMANCE COMPARISON

THEME	TOTAL SIZE	KEY PERFORMANCE ADVANTAGE	PAGE BUILDER
Astra	~50KB	No jQuery dependency. Fastest load of any multipurpose theme.	Elementor, Beaver, all builders
GeneratePress	~30KB	Cleanest code output. Consistently top Lighthouse scores.	Any page builder
Kadence	~65KB	Modern Full Site Editing support with excellent performance defaults.	Kadence Blocks + others
Block themes	~20KB	Zero PHP template overhead. Native Gutenberg performance.	Gutenberg blocks only

STEP 07 · ACTION CHECKLIST

Plugin & theme audit: your checklist

- [ ] AUDIT **Audit all active plugins using Query Monitor: Scripts & Styles**
  Identify JS and CSS files loading on pages where they serve no purpose.
- [ ] **Deactivate and delete all unused or redundant plugins**
  Even deactivated plugins add admin overhead and represent security vulnerabilities.
- [ ] **Replace heavy plugins with lighter alternatives where possible**
  e.g. replace a bulky contact form stack with a single lightweight plugin.
- [ ] **Use Asset CleanUp to restrict plugin CSS/JS to relevant page types**
  Disable a slider plugin's assets everywhere except the homepage where the slider actually appears.
- [ ] **Avoid page builders on performance-critical pages**
  Elementor and Divi add significant DOM complexity: native Gutenberg blocks are substantially lighter.
- [ ] **Switch to a lightweight theme (Astra, GeneratePress, Kadence, block)**
  Heavy themes can add 400KB+ of CSS and JS: lightweight themes add under 80KB.
- [ ] **Use a child theme for all theme file customisations**
  Prevents losing customisations on updates: never edit parent theme files directly.
- [ ] **Perform a binary plugin elimination test for highest-overhead plugins**
  Methodically find and replace the plugin adding the most PHP overhead.

> TIP: MEASURE YOUR PLUGIN TAX
>
> Deactivate ALL plugins temporarily (keeping only your caching plugin), run PageSpeed Insights, then compare to your baseline. The gap is the total performance tax of your plugin stack. Reactivate one by one to identify which add the most overhead.

8

STEP 08 · DELIVERY

CDN & content *delivery*.

Replicate the edge-caching advantage Shopify builds in by default.

A Content Delivery Network stores copies of your content on servers around the world. When a visitor in Berlin opens your site hosted in New York, instead of their request travelling 6,000km and back, Cloudflare serves it from a data centre 30km away. This is the primary reason Shopify outperforms unoptimised WordPress on Core Web Vitals: every Shopify site includes enterprise-grade CDN caching by default. Cloudflare's free plan provides the same capability in minutes.

HOW TO SET UP CLOUDFLARE CDN (FREE PLAN: FULL WALKTHROUGH)

01 Sign up at `cloudflare.com` and click "Add a site". Enter your domain and select the Free plan.

02 Cloudflare scans your existing DNS records and imports them: verify all records imported correctly.

03 Update your nameservers at your registrar (GoDaddy, Namecheap, etc.) to the two Cloudflare nameservers shown.

04 DNS propagation usually takes 1–4 hours. Cloudflare emails you when your site is active on their network.

05 In Cloudflare > Speed > Optimisation: enable Auto Minify for JavaScript, CSS and HTML.

06 Under SSL/TLS > Overview: set encryption mode to "Full (Strict)" if you have an SSL certificate on origin.

07 Under Caching > Configuration: set Browser Cache TTL to "1 year" for static resources.

08 To cache HTML at the edge: go to Rules > Cache Rules and create a rule caching your main page templates.

VERIFYING YOUR CDN IS WORKING CORRECTLY

```
# Check Cloudflare cache status from your terminal:
curl -I https://yoursite.com | grep -i cf-cache-status

cf-cache-status: HIT     -> Served from Cloudflare edge cache. Fast.
cf-cache-status: MISS    -> Fetched from origin. Normal on first visit.
cf-cache-status: BYPASS  -> Cache bypassed. Check your Cache Rules.
cf-cache-status: DYNAMIC -> Dynamic content (logged-in, cart). Not cacheable.
```

STEP 08 · ACTION CHECKLIST

CDN & content delivery: your checklist

- [ ] START HERE **Set up Cloudflare and point your domain nameservers to it**
  Free plan includes global CDN, DDoS protection, SSL certificate and Auto Minify.
- [ ] VERIFY **Verify static assets are served with CDN cache headers**
  Check Response Headers in DevTools > Network for cf-cache-status: HIT on static files.
- [ ] **Configure Cache Rules to cache HTML pages at the Cloudflare edge**
  Rules > Cache Rules: cache by URL pattern for the homepage and key page templates.
- [ ] **Enable Cloudflare Auto Minify for JavaScript, CSS and HTML**
  Speed > Optimisation, an additional minification layer at CDN level, no plugin required.
- [ ] **Set browser Cache TTL to 1 year for static assets**
  Caching > Configuration > Browser Cache TTL: visitors cache your assets locally for fast repeat visits.
- [ ] **Add preconnect hints for critical third-party origins**
  Add <link rel="preconnect"> in your theme head for analytics, fonts and payment providers.
- [ ] CRITICAL **Exclude /cart/, /checkout/, /my-account/ from CDN page caching**
  These pages are user-specific and must always be fetched from origin: add exceptions in Cache Rules.
- [ ] **Run WebPageTest from multiple global locations after CDN setup**
  Confirm TTFB improved for users in Europe, Asia and Australia, not just your local region.

> **TIP: KEEP THE EDGE FRESH**
>
> WP Rocket users: install the free Cloudflare add-on in WP Rocket > Add-ons. It automatically purges the Cloudflare edge cache whenever you clear the WP Rocket cache, ensuring visitors always see fresh content after you publish or update pages.

9

STEP 09 · CORE WEB VITALS

Core Web *Vitals*: metric-specific fixes.

LCP, INP and CLS each need targeted, different approaches.

Google's Core Web Vitals are three distinct measurements of user experience. Based on 2025 field data, TTFB is the biggest failure point for WordPress (only 65% pass), followed by LCP (86% pass). INP (91% pass) and CLS (94% pass) are already good for most sites. The lesson: check your own site's specific failures in Search Console before deciding where to invest effort.

LCP: LARGEST CONTENTFUL PAINT · TARGET UNDER 2.5S · BEST UNDER 1.5S

LCP measures how long the largest visible element takes to fully render: almost always your hero image or main heading text. LCP is directly bounded by TTFB: it cannot be faster than your server response. This is why fixing hosting and caching (Steps 2–3) must come before image optimisation. The full timeline is: TTFB + resource load delay + resource load time + render time.

01 Fix TTFB first (Steps 2–3). A 200ms TTFB improvement produces at least a 200ms LCP improvement automatically.

02 Identify your LCP element: run PSI and click the LCP timestamp in the filmstrip at the top of the report.

03 If LCP is an image: ensure it is in your HTML source, add fetchpriority="high", convert to WebP.

04 If LCP is text (H1): fix render-blocking CSS so the text renders without waiting for stylesheets (Step 5).

05 Add a preload hint for the LCP image in your theme head.

06 Verify the LCP image does not have loading="lazy": remove it if present.

07 Test on the Mobile tab in PSI specifically: Google uses mobile field data for rankings.

LCP CHECKLIST

- [ ] **Identify your LCP element using the PSI filmstrip view**
  Click the LCP timestamp: PSI highlights the exact element and names it.
- [ ] LCP FIX **Add fetchpriority="high" to the LCP image in your theme HTML**
  Direct browser prioritisation: can reduce LCP by 200–500ms. The most impactful single LCP fix.
- [ ] LCP FIX **Preload the LCP image using <link rel="preload">**
  In your theme head or via WP Rocket's Preload tab: starts the download before HTML fully parses.
- [ ] CRITICAL **Confirm the LCP image does NOT have loading="lazy"**
  Lazy loading the hero image actively delays LCP, one of the most common LCP mistakes.
- [ ] **Confirm the LCP image is in the initial HTML, not loaded via JavaScript**
  JS-loaded images cannot begin downloading until the JS executes, a major LCP delay.
- [ ] **Check LCP on Mobile specifically in PageSpeed Insights**
  Mobile LCP is almost always worse than desktop: Google's ranking signal uses mobile data.

INP: Interaction to Next Paint `Target <200ms · Best <100ms`

INP replaced FID as a Core Web Vital in March 2024. It measures the time from any user interaction (click, tap, keyboard input) to the next visual update. The good news: 91% of WordPress sites already pass INP, because core uses minimal client-side JavaScript. INP failures are almost always caused by heavy third-party scripts or complex plugin interactions. Check yours in Search Console before spending time here.

- [ ] **Check INP status in Search Console > Experience > Core Web Vitals**
  91% of WP sites pass: confirm yours is in that group. If passing, focus effort on LCP instead.
- [ ] **If INP fails: install the Web Vitals extension and interact with your page**
  The extension shows live INP values as you click: identify which interactions are slow.
- [ ] **Use DevTools > Performance to find long tasks on interaction**
  Record, interact, then look for red “Long Task” markers (over 50ms) in the main thread.
- [ ] **Remove or defer heavy click handlers and animation scripts**
  Heavy synchronous JavaScript on interaction events is the primary INP cause on WordPress sites.

CLS: Cumulative Layout Shift `Target <0.1 · Best <0.05`

CLS measures visual stability: how much elements move during page load. A high CLS means users try to click a link and accidentally tap a button because the page jumped as an image loaded above it. 94% of WordPress sites already pass CLS. The most common cause of failures is images without explicit width and height attributes.

- [ ] `CLS FIX` **Add explicit width and height attributes to every img tag**
  Without dimensions the browser does not reserve space: image loading pushes content down.
- [ ] `CLS FIX` **Reserve space for advertisement containers with min-height CSS**
  Ad slots that expand from zero height when loaded are a major, common source of CLS.
- [ ] **Use font-display: swap and preload your main font**
  Text reflow when fonts swap causes CLS: swap + preload minimises the window during which this happens.
- [ ] **Use aspect-ratio CSS for all video embeds and iframes**
  aspect-ratio: 16/9 reserves the correct space before the embed loads, preventing height-shift.
- [ ] **If CLS is failing: use PSI to identify the specific shifting element**
  PSI highlights CLS elements in the filmstrip: click the CLS score to see exactly what is shifting.

> **TIP: FIELD DATA VS LAB DATA**
>
> PSI shows both. Lab data (Lighthouse) is measured in a controlled test. Field data comes from real Chrome users: your Search Console CWV report uses field data, and this is what Google uses for rankings. Optimise for real user data, not just your lab score.

10

STEP 10 · MAINTENANCE

Monitoring & ongoing maintenance.

Performance degrades without active monitoring: catch regressions early.

Performance optimisation is not a one-time project. A single new plugin, a large image upload, a theme update, or a new analytics script can undo weeks of careful work without warning. Sites that stay fast have monitoring in place. The goal is to catch regressions before Google's crawlers do, because once Search Console shows "Poor" URLs, the ranking impact has already begun.

HOW TO SET UP AUTOMATIC CWV ALERTS FROM GOOGLE (FREE, 2 MINUTES)

01 Go to `search.google.com/search-console` and select your property.

02 Click "Core Web Vitals" under the "Experience" section in the left navigation.

03 Google automatically emails your verified GSC address when any URLs drop from "Good" to "Poor".

04 No additional setup is needed: just confirm your email is verified and receiving GSC notifications.

05 Check this report weekly: it shows URL-level issues with real-user data, grouped by page type.

06 When URLs show "Needs Improvement" or "Poor": click through to see which metric is failing and where.

YOUR MONTHLY PERFORMANCE ROUTINE: 15 MINUTES

01 Run PSI (Mobile) on your homepage and top 3 pages. Record LCP and TTFB scores. (5 min)

02 Open GSC > Core Web Vitals. Look for any new Needs Improvement or Poor URLs vs last month. (3 min)

03 Check TTFB: DevTools > Network > reload homepage in incognito > first HTML request > Waiting (TTFB). (2 min)

04 Review uptime monitoring alerts from the past 30 days in UptimeRobot or Better Uptime. (2 min)

05 If any score dropped, check what changed: new plugin, theme update, new image? (3 min)

RECOMMENDED FREE MONITORING TOOLS

TOOL	WHAT IT MONITORS	COST	SETUP
UptimeRobot	Uptime and response-time alerts, checks every 5 minutes.	**Free**	5 min
Google Search Console	Core Web Vitals email alerts when pages drop to Poor.	**Free**	Already done
Web Vitals Extension	Live CWV scores as you browse your site: instant spot-checks.	**Free**	2 min
Better Uptime	Uptime monitoring with real browser checks and a status page.	Free / Paid	10 min
GTmetrix Monitoring	Scheduled weekly performance reports sent to your email.	Free / Paid	5 min

STEP 10 · ACTION CHECKLIST

Monitoring & maintenance: your checklist

- [ ] SET UP NOW **Set up UptimeRobot or Better Uptime for downtime alerts**
  Free plans check every 5 minutes and send email/SMS on downtime: essential for any live site.
- [ ] **Confirm Google Search Console CWV email notifications are active**
  GSC emails you automatically when URLs drop to Poor: no configuration needed beyond GSC setup.
- [ ] **Install the Web Vitals Chrome Extension for quick ongoing checks**
  Shows live LCP, INP and CLS values as you browse: spot regressions immediately.
- [ ] **Create a monthly PSI score log in a spreadsheet**
  Record date, PSI mobile score, LCP and TTFB monthly: reveals trends and catches slow degradations.
- [ ] **Run PSI within 24 hours of any major plugin update**
  Plugin updates can silently add scripts: check scores immediately after significant updates.
- [ ] **Use a staging environment for testing before deploying updates**
  Test all plugin and theme updates on staging first: deploy to live only after verifying no regression.
- [ ] **Document your current performance baseline and save it**
  Date, PSI score, LCP, TTFB, hosting setup, active plugins, a historical record saves debugging time.
- [ ] **Run a full GTmetrix + WebPageTest audit every quarter**
  More thorough than monthly checks: revisit hosting, CDN effectiveness and plugin audit quarterly.

> TIP: GIVE FIELD DATA TIME
>
> After every optimisation, wait 24–48 hours before measuring in Search Console. Field data uses a 28-day rolling window: changes take time to appear. Use PSI lab data for immediate feedback, then confirm the real-user improvement in Search Console after 2–4 weeks.

Performance maintenance *schedule*.

Consistency matters more than intensity. Use this schedule to keep your site fast over time: 15 minutes weekly is more effective than one large annual sprint.

DAILY

- Check site uptime status
- Review security scan results
- Monitor backup completion
- Check for critical WP or plugin updates

WEEKLY

- Run PSI on homepage (Mobile)
- Check GSC CWV report
- Update plugins and themes
- Clear cache after updates
- Review new 404 errors

MONTHLY

- Full PSI audit on all key pages
- Database cleanup with WP-Optimize
- Audit new image uploads for size
- Check autoloaded options size
- User access review

QUARTERLY

- Complete plugin audit
- Hosting performance review
- Full GTmetrix & WebPageTest audit
- Content strategy & SEO review
- Update documentation

QUICK DIAGNOSTIC GUIDE: WHAT YOUR SCORES ARE TELLING YOU

SYMPTOM	MOST LIKELY CAUSE	FIX IN
TTFB over 600ms	No page cache, no object cache, or underpowered hosting.	Steps 2, 3
LCP over 4s on mobile	Slow TTFB, unoptimised hero image, no fetchpriority.	Steps 3, 4, 9
CLS over 0.1	Images without dimensions, font swap, ad containers.	Steps 4, 5, 9
INP over 500ms	Heavy JavaScript on click events or third-party scripts.	Steps 5, 9
Good lab but bad Search Console	Lab vs real-user data gap: optimise for field data.	Steps 9, 10
Scores dropped after a plugin update	Plugin added JS, CSS, or slow DB queries.	Steps 6, 7
Mobile much worse than desktop	Images not responsive, no mobile cache, render-blocking CSS.	Steps 4, 5
Was fine, now suddenly worse	New plugin, large image upload, or theme update.	Step 10

Recommended tools & *plugins*.

These are the tools used by professional WordPress developers and agencies worldwide. Each has been selected for reliability, active maintenance and proven real-world performance impact.

MEASUREMENT & DIAGNOSTICS

TOOL / PLUGIN	WHERE TO FIND	WHAT IT DOES	COST
Google PageSpeed Insights	pagespeed.web.dev	LCP/INP/CLS/TTFB with highlighted elements + real-user field data.	**Free**
Google Search Console	search.google.com/sc	Site-wide CWV report from real Chrome users: what Google sees.	**Free**
GTmetrix	gtmetrix.com	Full waterfall chart, video recording of page load process.	Free / $14mo
WebPageTest	webpagetest.org	Multi-location test, advanced diagnostics, TTFB by region.	**Free**
Query Monitor	WordPress.org	DB queries, scripts, styles, hooks: per-page breakdown.	**Free**
Web Vitals Extension	Chrome Web Store	Live CWV scores displayed as you browse your own site.	**Free**

CACHING

TOOL / PLUGIN	WHERE TO FIND	WHAT IT DOES	COST
WP Rocket	wp-rocket.me	All-in-one page caching, JS defer, minification: easiest setup.	$59/yr
LiteSpeed Cache	WordPress.org	Best for LiteSpeed servers: page cache, object cache, CDN.	**Free**
Redis Object Cache	WordPress.org	Connects WordPress to Redis: by Till Kruss. Essential.	**Free**
W3 Total Cache	WordPress.org	Highly configurable, works with any host and server type.	Free / $99yr

IMAGE OPTIMISATION

TOOL / PLUGIN	WHERE TO FIND	WHAT IT DOES	COST
Smush	WordPress.org	WebP conversion, auto-compress on upload, lazy loading.	Free / $7.50mo
ShortPixel	shortpixel.com	AVIF support, best compression quality, bulk processing.	From $4.99
Imagify	imagify.io	Simple setup, 25MB free compression monthly.	Free / $9.99mo
EWWW Image Optimizer	WordPress.org	Unlimited local optimisation, no API key or cloud needed.	Free / $7mo

CSS, JAVASCRIPT & FONTS

TOOL / PLUGIN	WHERE TO FIND	WHAT IT DOES	COST
Autoptimize	WordPress.org	Minify and aggregate CSS and JS, critical CSS extraction.	**Free**
Asset CleanUp	WordPress.org	Disable specific CSS and JS files per page type or URL.	Free / $49
OMGF	WordPress.org	Download and self-host Google Fonts locally, automatically.	**Free**

DATABASE OPTIMISATION

TOOL / PLUGIN	WHERE TO FIND	WHAT IT DOES	COST
WP-Optimize	WordPress.org	DB cleanup, revision removal, transient clearing, scheduling.	Free / Paid
Advanced DB Cleaner	WordPress.org	Granular control over database tables and orphaned data.	Free / $29

CDN & DELIVERY

TOOL / PLUGIN	WHERE TO FIND	WHAT IT DOES	COST
Cloudflare	cloudflare.com	Global CDN, DDoS protection, edge caching, free SSL, Auto Minify.	Free / $20mo
BunnyCDN	bunny.net	High-performance CDN at $0.01/GB: extremely cost-effective.	From $1/mo
Cloudflare Images	Cloudflare dashboard	Image CDN with auto WebP and AVIF conversion at the edge.	$5/mo (100K)

MONITORING

TOOL / PLUGIN	WHERE TO FIND	WHAT IT DOES	COST
UptimeRobot	`uptimerobot.com`	Uptime monitoring every 5 minutes, email and SMS alerts.	Free / $7mo
Better Uptime	`betteruptime.com`	Uptime + real browser checks + public status page.	Free / $20mo
GTmetrix	`gtmetrix.com`	Scheduled weekly performance reports sent by email.	Free / $14mo

ninegravity

NEED A HAND?

Need expert help implementing *this checklist?*

NineGravity provides professional WordPress performance optimisation, maintenance and managed services. Our team handles everything in this checklist: from hosting configuration and Redis setup to Core Web Vitals optimisation, image delivery, CDN configuration and ongoing monitoring.

www.ninegravity.com/contact ↗