**Avoid enormous network payloads**

May 2, 2019 • Updated May 29, 2020

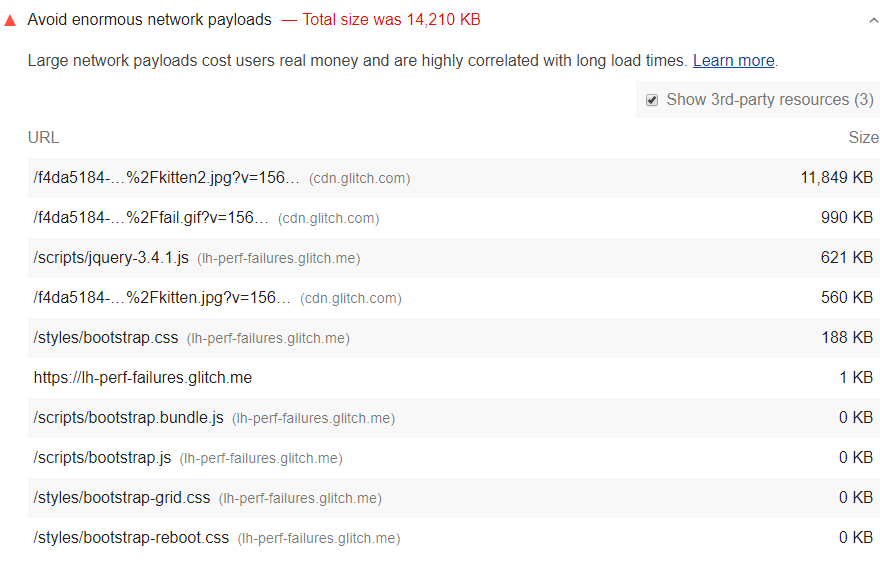
Appears in: [Performance audits](https://web.dev/lighthouse-performance)

Large network payloads are highly correlated with long load times. They also cost users money; for example, users may have to pay for more cellular data. So, reducing the total size of your page's network requests is good for your users' experience on your site *and* their wallets.

To see what accessing your site costs around the world, check out WebPageTest's [What Does My Site Cost?](https://whatdoesmysitecost.com/) You can adjust the results to factor in purchasing power.

**How the Lighthouse network payload audit fails** [**#**](https://web.dev/total-byte-weight/?utm_source=lighthouse&utm_medium=devtools#how-the-lighthouse-network-payload-audit-fails)

[Lighthouse](https://developers.google.com/web/tools/lighthouse/) shows the total size in [kibibytes (KiB)](https://en.wikipedia.org/wiki/Kibibyte) of all resources requested by your page. The largest requests are presented first:



Based on [HTTP Archive data](https://httparchive.org/reports/state-of-the-web?start=latest#bytesTotal), the median network payload is between 1,700 and 1,900 KiB. To help surface the highest payloads, Lighthouse flags pages whose total network requests exceed 5,000 KiB.

See the [Lighthouse performance scoring](https://web.dev/performance-scoring) post to learn how your page's overall performance score is calculated.

**How to reduce payload size** [**#**](https://web.dev/total-byte-weight/?utm_source=lighthouse&utm_medium=devtools#how-to-reduce-payload-size)

Aim to keep your total byte size below 1,600 KiB. This target is based on the amount of data that can be theoretically downloaded on a 3G connection while still achieving a [Time to Interactive](https://web.dev/interactive) of 10 seconds or less.

Here are some ways to keep payload size down:

* Defer requests until they're needed. See the [PRPL Pattern](https://web.dev/apply-instant-loading-with-prpl) for one possible approach.
* Optimize requests to be as small as possible. Possible techniques include:
  + [Minify and compress network payloads](https://web.dev/reduce-network-payloads-using-text-compression).
  + [Use WebP instead of JPEG or PNG for your images](https://web.dev/serve-images-webp).
  + [Set the compression level of JPEG images to 85](https://web.dev/use-imagemin-to-compress-images).
* Cache requests so that the page doesn't re-download the resources on repeat visits. (See the [Network reliability landing page](https://web.dev/reliable) to learn how caching works and how to implement it.)

**Stack-specific guidance** [**#**](https://web.dev/total-byte-weight/?utm_source=lighthouse&utm_medium=devtools#stack-specific-guidance)

**Angular** [**#**](https://web.dev/total-byte-weight/?utm_source=lighthouse&utm_medium=devtools#angular)

Apply [route-level code splitting](https://web.dev/route-level-code-splitting-in-angular/) to minimize the size of your JavaScript bundles. Also, consider precaching assets with the [Angular service worker](https://web.dev/precaching-with-the-angular-service-worker/).

**Drupal** [**#**](https://web.dev/total-byte-weight/?utm_source=lighthouse&utm_medium=devtools#drupal)

Consider using [Responsive Image Styles](https://www.drupal.org/docs/8/mobile-guide/responsive-images-in-drupal-8) to reduce the size of images loaded on your page. If you are using Views to show multiple content items on a page, consider implementing pagination to limit the number of content items shown on a given page.

**Joomla** [**#**](https://web.dev/total-byte-weight/?utm_source=lighthouse&utm_medium=devtools#joomla)

Consider showing excerpts in your article categories (e.g. via a "read more" link), reducing the number of articles shown on a given page, breaking your long posts into multiple pages, or using a plugin to lazy-load comments.

**WordPress** [**#**](https://web.dev/total-byte-weight/?utm_source=lighthouse&utm_medium=devtools#wordpress)

Consider showing excerpts in your post lists (e.g. via the "more" tag), reducing the number of posts shown on a given page, breaking your long posts into multiple pages, or using a plugin to lazy-load comments

**How To Avoid Enormous Network Payloads In WordPress (15 Ways)**

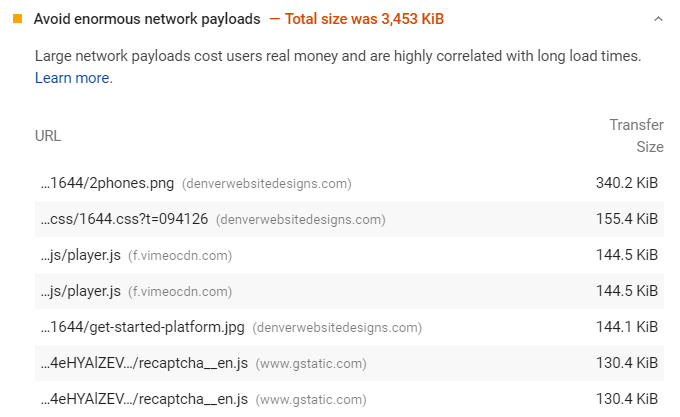
* [0 Comments](https://onlinemediamasters.com/avoid-enormous-network-payloads-wordpress/#comments)
* [Tom Dupuis](https://onlinemediamasters.com/about-us/)
* Last Updated: November 22, 2021

Let’s talk about [enormous network payloads](https://web.dev/total-byte-weight/) in PageSpeed Insights.

This is triggered when the total page size is above [1,600 KiB](https://web.dev/total-byte-weight/#how-to-reduce-payload-size). To pass this test, you need to **reduce page size**.

A large page size is often caused by unoptimized images, videos, third-party code, or heavy CSS and JavaScript files. It can also be from slow page builders (specifically Elementor and Divi) which add extra CSS, JavaScript, and div wrappers to every single page on your WordPress site.

PageSpeed Insights shows your largest files which is where you should be focusing your attention. Once you know these, reference the solutions in this guide (which are specific to WordPress sites) and you should be able to pass the avoid enormous network payloads test.



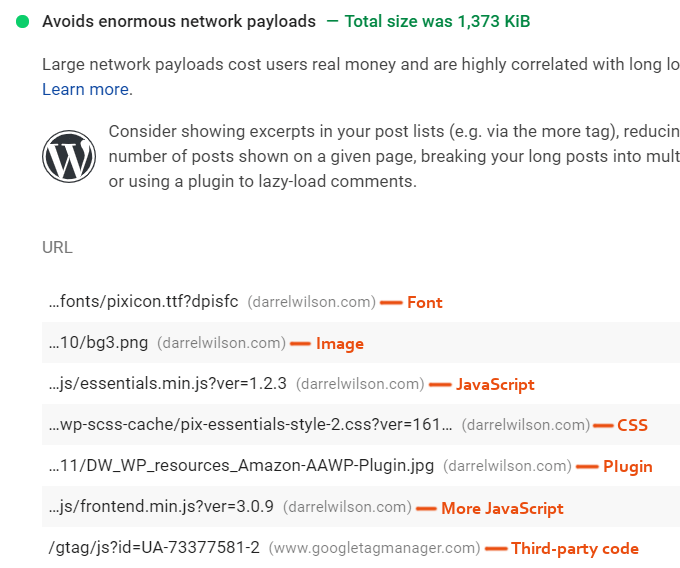
1. [Identify The Cause Of Enormous Network Payloads](https://onlinemediamasters.com/avoid-enormous-network-payloads-wordpress/#cause)
2. [Avoid Enormous Images](https://onlinemediamasters.com/avoid-enormous-network-payloads-wordpress/#large-images)
3. [Compress Images](https://onlinemediamasters.com/avoid-enormous-network-payloads-wordpress/#compress-images)
4. [Consider WebP](https://onlinemediamasters.com/avoid-enormous-network-payloads-wordpress/#webp)
5. [Minify CSS + JavaScript](https://onlinemediamasters.com/avoid-enormous-network-payloads-wordpress/#minify)
6. [Remove Slow Page Builders](https://onlinemediamasters.com/avoid-enormous-network-payloads-wordpress/#page-builders)
7. [Remove Unused CSS + JavaScript](https://onlinemediamasters.com/avoid-enormous-network-payloads-wordpress/#remove-unused-css-js)
8. [Optimize Google Fonts](https://onlinemediamasters.com/avoid-enormous-network-payloads-wordpress/#google-fonts)
9. [Optimize Third-Party Code](https://onlinemediamasters.com/avoid-enormous-network-payloads-wordpress/#third-party-code)
10. [Delay JavaScript](https://onlinemediamasters.com/avoid-enormous-network-payloads-wordpress/#delay-javascript)
11. [Identify Your Slowest Plugins](https://onlinemediamasters.com/avoid-enormous-network-payloads-wordpress/#slow-plugins)
12. [Use An Efficient Caching Plugin](https://onlinemediamasters.com/avoid-enormous-network-payloads-wordpress/#cache)
13. [Avoid Enormous Payloads With WP Rocket](https://onlinemediamasters.com/avoid-enormous-network-payloads-wordpress/#wp-rocket)
14. [Take Advantage Of Server-Side Caching](https://onlinemediamasters.com/avoid-enormous-network-payloads-wordpress/#server-caching)
15. [Reduce Number Of Elements On The Page](https://onlinemediamasters.com/avoid-enormous-network-payloads-wordpress/#reduce-elements)

**1. Identify The Cause Of Enormous Network Payloads**

Use PageSpeed Insights to identify files causing enormous network payloads.

Pay attention to the **file type** (image, CSS, JS) and **where the file is being served from** (your domain, CDN, a third-party service).

Just by glimpsing at your report, you will know which files need to be optimized so you can narrow down the solution.



**2. Avoid Enormous Images**

Huge images can cause enormous network payloads.

These appear in the [properly size images recommendation](https://onlinemediamasters.com/properly-size-images-wordpress/).

This simply means you need to resize images to smaller dimensions. I created a “cheat sheet” with image dimensions for my logo, blog, sidebar, and inner pages. That way, I know the exact dimensions images should be resized to before uploading them. Some image optimization plugins have resizing options, but it’s best to upload them with correct dimensions beforehand.

[Adaptive image plugins](https://wordpress.org/plugins/search/adaptive+images/) can serve smaller images to mobile devices and [improve mobile speed](https://onlinemediamasters.com/mobile-wordpress-speed/).

**3. Compress Images**

This is also known as [efficiently encoding images](https://web.dev/uses-optimized-images/).

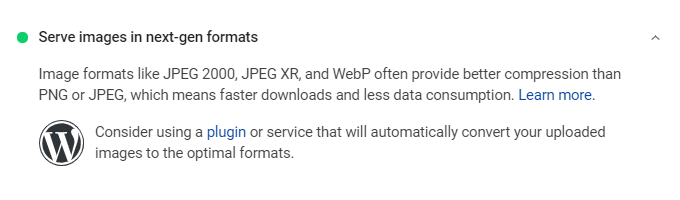
Lighthouse sets the compression level to 85% and flags images that would have a savings of 4KiB or higher.

ShortPixel and TinyPNG are popular options, or you can do this in Photoshop or GIMP. Try setting the compression level to 85% and you shouldn’t have errors for this item anymore.

**4. Consider WebP**

[WebP images](https://onlinemediamasters.com/serve-images-in-next-gen-formats-wordpress/) typically have a [25-34% smaller file size](https://developers.google.com/speed/webp) than JPEGs and PNGs (and are also higher quality).

You can convert images to WebP using ShortPixel, Imagify, and many other image optimization plugins for WordPress. If your plugin doesn’t support it, you can also use [WebP Converter For Media](https://wordpress.org/plugins/webp-converter-for-media/). When selecting the conversion method, <picture> tag is usually the preferred method.



However, many speed-related sites like Kinsta and WP Rocket use SVGs for their high quality images (logo, background image, etc). SVGs don’t get flagged in Lighthouse and still produce high quality images similar to WebP. You may need to use an [SVG support plugin](https://wordpress.org/plugins/svg-support/) to avoid errors.

**5. Minify CSS + JavaScript**

Most cache plugins have settings to minify CSS and JavaScript which removes unnecessary characters from your code (reducing the file size).

If enabling this gives you errors, you’ll need to find the problematic file(s) in your source code and exclude them from minification. Otherwise, make sure you’re minifying CSS and JavaScript.

**6. Remove Slow Page Builders**

[Elementor](https://onlinemediamasters.com/slow-elementor-website/) and [Divi](https://onlinemediamasters.com/divi-slow-loading-website/) add unnecessary CSS and JavaScript.

I was previously using Elementor and [hired WP Johnny](https://wpjohnny.com/wordpress-speed-optimization-service/?ref=16) for his page builder removal services and it made a huge difference. Just by hard coding my header, footer, menu, and sidebar, my entire web vitals score shot up. Then he migrated my Elementor pages to Gutenberg which made an even bigger difference. I was averaging C scores and after doing this, I’m averaging A’s.

In Facebook Groups, there is a large trend of people migrating away from Elementor/Divi. Popular alternatives are Gutenberg, Oxygen Builder, Kadence Theme, and GeneratePress.

*Source:* [*WordPress Speed Up Facebook Group*](https://www.facebook.com/groups/wordpressspeedup/posts/1242603109262667)

*Source:* [*WordPress Speed Up Facebook Group*](https://www.facebook.com/groups/wordpressspeedup/posts/1535484913307817)

*Source:* [*Bloggers Passion Facebook Group*](https://www.facebook.com/groups/BloggersPassionVIP/posts/1562548614112710)

*Source:* [*Oxygen User Facebook Group*](https://www.facebook.com/groups/1626639680763454/posts/3587742217986514)

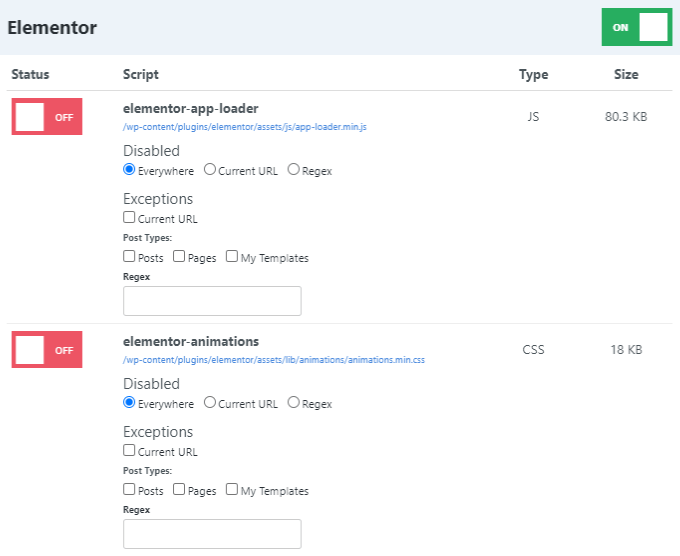
*Source:* [*Oxygen4Fun*](https://oxygen4fun.supadezign.com/wp-charts-comparisons/#pagebuilders)

**7. Remove Unused CSS + JavaScript**

Asset CleanUp and [Perfmatters](https://perfmatters.io?ref=146) can trim the size of CSS and JavaScript.

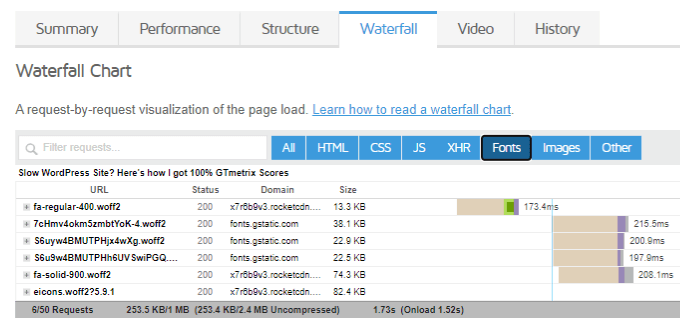
They essentially provide you with a [script manager](https://perfmatters.io/docs/disable-scripts-per-post-page/?ref=146) to disable CSS/JS on pages where they don’t need to load. So if your contact form loads across your entire site, only load it on the contact page. Plugins for social sharing, sliders, rich snippets, live chat, and others can often be disabled on certain content. I wrote a comparison about [Perfmatters vs. Asset CleanUp](https://onlinemediamasters.com/perfmatters-vs-asset-cleanup/), but I use Perfmatters since the UI/UX is better. However, [Asset CleanUp Pro](https://onlinemediamasters.com/go/asset-cleanup-pro/) can disable custom CSS.

As mentioned previously, Elementor and Divi can add lots of CSS and JavaScript to your site. Some files might be able to be disabled such as elementor-sticky, dialog, share-link, swiper, animations, icons, and wp-block-library if you don’t use these features. Elementor also released Optimized Dom Output and Improved Asset Loading which can help reduce network payloads.



**8. Optimize Google Fonts**

Use the GTmetrix Waterfall chart to see how long your fonts take to load. They can slow especially if you’re loading multiple font families, weights, and they’re not hosted locally.



Make your fonts load faster:

* Host fonts locally using [OMGF](https://wordpress.org/plugins/host-webfonts-local/) or [Transfonter](https://transfonter.org/).
* Be very minimal with font families, weights, icons.
* Use browser resource hints (preload, preconnect, prefetch).
* Avoid font plugins and serving fonts from any type of plugin.
* Add [font-display: swap](https://onlinemediamasters.com/ensure-text-remains-visible-during-webfont-load-wordpress/) to ensure text remains visible during webfont load.

**9. Optimize Third-Party Code**

When trying to avoid enormous network payloads, you may notice files are being loaded from a third-party site.

These can flag additional PageSpeed Insights items such as [reduce third-party code](https://onlinemediamasters.com/reduce-impact-of-third-party-code/). Below are common examples of third-party code and common solutions to optimize them. Some items are easy to optimize (hosting fonts and analytics locally) while others (AdSense and GTM) are not.

* **Google Fonts** – host locally instead of serving them from //fonts.gstatic.com. Also take advantage of browser resource hints (preload, preconnect, prefetch).
* **Google Maps** – take a screenshot of the map and link it to driving directions.
* **Google Analytics** – host locally using WP Rocket or Flying Analytics. If using Perfmatters, use a smaller tracking code (minimal, minimal inline, or analytics.js) and [disable display features](https://perfmatters.io/docs/local-analytics/#disable-display-features) to prevent a second HTTP request to Doubleclick.
* **Google AdSense** – lazy load and delay JavaScript in WP Rocket or Flying Scripts.
* **Google Tag Manager** – delay in WP Rocket or Flying Scripts and clean up tags.
* **Facebook Pixel** – hosting it locally using WP Rocket is the only way I know of.
* **YouTube** – lazy load, replace iframes with preview images, and delay JavaScript.
* **Social Media** – use [Grow by Mediavine](https://wordpress.org/plugins/social-pug/) which was the fastest social sharing plugin in WP Rocket’s test, avoid social media widgets (e.g. Facebook Like boxes).
* **Gravatars** – delay Gravatars and use a local Gravatar image with [WP User Avatar](https://wordpress.org/plugins/wp-user-avatar/) (my blog comments show an example of a custom Gravatar image I use).
* **WPdiscuz** – tweak the settings to [initiate AJAX loading after page](https://onlinemediamasters.com/wp-content/webpc-passthru.php?src=https://onlinemediamasters.com/wp-content/uploads/2020/06/Disqus-Comment-Thread-Displaying.png&nocache=1), disable [WordPress native AJAX functions](https://onlinemediamasters.com/wp-content/webpc-passthru.php?src=https://onlinemediamasters.com/wp-content/uploads/2020/06/Disqus-General-Settings.png&nocache=1), and disable [load font awesome CSS lib](https://onlinemediamasters.com/wp-content/webpc-passthru.php?src=https://onlinemediamasters.com/wp-content/uploads/2020/06/Disqus-Load-Font-Awesome-CSS-Lib.png&nocache=1). After delaying comments and using WP User Avatar, your comments should load very quickly. Native WordPress comments fast too (I just like the look of WPdiscuz).

**10. Delay JavaScript**

Delaying JavaScript can be done in WP Rocket ([delay JavaScript execution](https://docs.wp-rocket.me/article/1349-delay-javascript-execution)) or [Flying Scripts](https://wordpress.org/plugins/flying-scripts/).

Each plugin works a little differently. WP Rocket delays JavaScript until user interaction (e.g. mouse scroll, click). Flying Script sets a timeout period in seconds until the JavaScript is loaded.

This **won’t** help avoid enormous network payloads since you’re just delaying JavaScript (not optimizing it) but it can improve your site’s initial load time and potentially, your web vital scores. This is especially true if you delay AdSense, comments, and heavy third-party code.

Below is the default list of JavaScript WP Rocket delays. If you have other non-critical JavaScript that can be delayed (check your GTmetrix Waterfall report), try delaying that too.

addtoany

cookie-law-info

apps.elfsight.com/p/platform.js

xfbml.customerchat.js

fbevents.js

fbq(

/busting/facebook-tracking/

/sdk.js#xfbml

feedbackcompany.com/includes/widgets/feedback-company-widget.min.js

ft.sdk.min.js

getbutton.io

adsbygoogle.js

google-analytics.com/analytics.js

ga( '

ga('

/gtag/js

gtag(

/gtm.js

/gtm-

translate.google.com/translate\_a/element.js

static.hotjar.com/c/hotjar-

static.leadpages.net/leadbars/current/embed.js

snap.licdn.com/li.lms-analytics/insight.min.js

livechatinc.com/tracking.js

LiveChatWidget

widget.manychat.com

olark

//a.omappapi.com/app/js/api.min.js

pinit.js

pixel-caffeine/build/frontend.js

shareaholic

ShopifyBuy

sharethis

simple-share-buttons-adder

smartsuppchat.com/loader.js

Tawk\_API

widget.trustpilot.com/bootstrap

static.ads-twitter.com/uwt.js

twq(

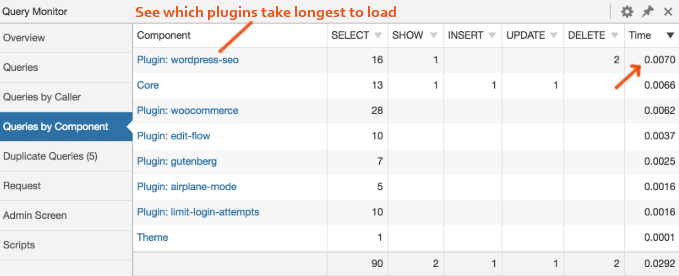
platform.twitter.com/widgets.js

wpdiscuz

**11. Identify Your Slowest Plugins**

[Query Monitor](https://wordpress.org/plugins/query-monitor/) and [WP Hive](https://chrome.google.com/webstore/detail/wp-hive/hbbcblcogociooaeodndcimehhchadka) are great tools for finding slow plugins.

Query Monitor has a “queries by components” tab which shows your slowest loading plugins.

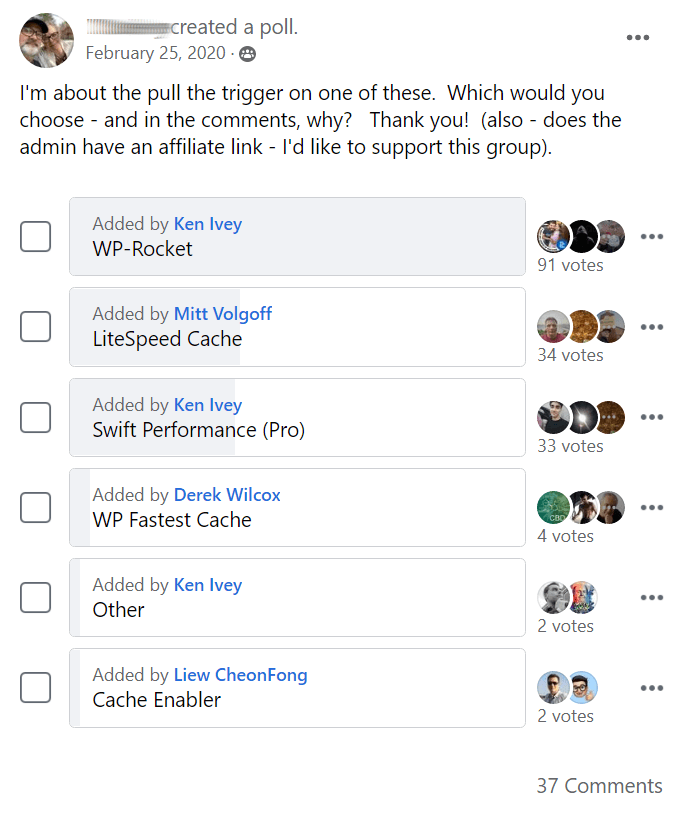


WP Hive is a Chrome Extension that add a little section when searching the WordPress repository that shows whether a plugin has an impact on memory usage or PageSpeed Insights.

Otherwise, eliminating unnecessary plugins and replacing [high CPU plugins](https://onlinemediamasters.com/slow-wordpress-plugins/) with lightweight plugins is an ongoing process that can take time, but can prevent enormous network payloads.

**12. Use An Efficient Caching Plugin**

[WP Rocket](https://onlinemediamasters.com/go/wp-rocket/) and [LiteSpeed Cache](https://wordpress.org/plugins/litespeed-cache/) are the gold standards for cache plugins.



*Source:* [*WordPress Speed Up Facebook Group*](https://www.facebook.com/groups/wordpressspeedup/permalink/1269066179949693/)

*Source:* [*Oxygen User Group*](https://www.facebook.com/groups/1626639680763454/permalink/3441023242658413/)

If you’re using a LiteSpeed Server, use LiteSpeed Cache. Otherwise I would personally use WP Rocket for nearly all other cases (with the exception of SG Optimizer on SiteGround, but [I don’t recommend their hosting](https://onlinemediamasters.com/siteground-wordpress-hosting-review/) as they have a slow TTFB). Here’s my [WP Rocket configuration guide](https://onlinemediamasters.com/wp-rocket-settings/).

**Which** cache plugin you’re using and **how** you configure the settings has a large impact on your web vital scores. Make sure you test each setting (specifically the file optimization tab in WP Rocket) to see how each setting impacts scores and load times.

**13. Avoid Enormous Payloads With WP Rocket**

WP Rocket [says](https://docs.wp-rocket.me/article/1415-avoid-enormous-network-payloads) they can help avoid enormous payloads with:

* Browser caching
* Minify CSS
* Minify JavaScript
* Delay JavaScript Execution
* Lazy Load for images and iframes

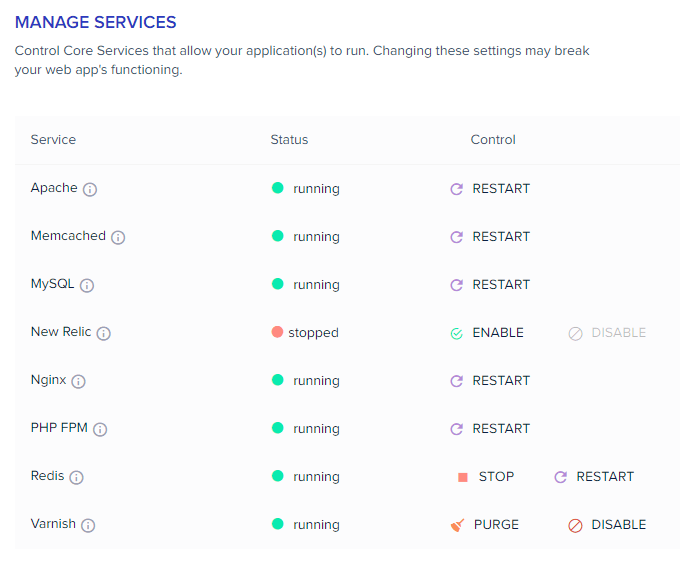
You’ll want to enable these in your WP Rocket settings.

**14. Take Advantage Of Server-Side Caching**

Most cloud hosts have server-side caching built-in to their dashboard.

Redis, memcached, and Varnish are all types of caching that can reduce network payloads. [I use Cloudways](https://onlinemediamasters.com/cloudways-review/) and have Redis enabled which is a popular, efficient caching option. SiteGround has NGINX-based delivery which is available in Site Tools, and Kinsta has their own caching as well.

Server-side caching is faster than file-based caching done by cache plugin. Take advantage of it.

****

**15. Reduce Number Of Elements On The Page**

Reducing the number of elements on your pages will reduce page size and network payloads.

* Reduce images
* Reduce videos
* Reduce sliders
* Reduce featured posts
* Reduce social media feeds
* Break blog comments
* Show smaller excerpts

**Frequently Asked Questions**

**How do I avoid enormous network payloads in WordPress?**

Reducing page size is the best way to avoid enormous network payloads. This can be done by reducing the size of images, videos, scripts, styles, and fonts.

**How do I avoid enormous network payloads using WP Rocket?**

Test each setting in the file optimization tab to see it's impact on your scores. Minifying CSS and JavaScript is one of the only ways WP Rocket can avoid enormous network payloads.

**How do I reduce CSS and JavaScript?**

Minify and combine CSS and JavaScript. However, the underlying problem is often due to extra CSS and JavaScript added by slow page builders and third-party code.

# [Natively lazy-loading Facebook social plugins](https://www.phpied.com/natively-lazy-loading-facebook-social-plugins/)

July 24th, 2020. Tagged: [facebook](https://www.phpied.com/category/facebook/), [JavaScript](https://www.phpied.com/category/javascript/)

tl;dr: Add data-lazy="true" to your Facebook social plugins that are below the fold and reap the benefits.

In code:

// before

<div

class="fb-like"

data-href="https://phpied.com"></div>

// after

<div

class="fb-like"

data-href="https://phpied.com"

data-lazy="true"></div>

The following 18 seconds video demonstrates the difference. Where currently your visitors load Facebook iframe content even if it's way down the page, after you implement lazy-loading, they'll only load the iframes if there's a chance of ever seeing them. Look for the like.php request and its dependencies.

## How these plugins work?

[Facebook's social plugins](https://developers.facebook.com/docs/plugins) are pieces of Facebook content you can embed into your pages. Like buttons, share buttons, comments, embedded videos, posts and photos and so on.

The plugins are iframes that point to e.g. facebook.com/plugins/like.php, facebook.com/plugins/comments.php and so on. Some plugins allow you to create the iframes youself, where others do not. And in general it's better not to. Why? Because content can vary based on language and other factors and by creating the iframe yourself you risk the content being cut off.

So who writes the iframes then? The [FB JSSDK](https://developers.facebook.com/docs/javascript/). You load the SDK (asynchronously, of course) and sprinkle divs where you want the plugins to appear. The SDK writes and resizes the iframes as required.

## What's new?

Browsers now support [native lazy-loading of iframes](https://web.dev/iframe-lazy-loading/).

Facebook now supports native lazy-loading of all social plugins.

All you have to do is add the data-lazy="true" attribute to your plugin div elements.