Kinsta

WHITE PAPER

Advanced WordPress Site Speed Optimizations

White Paper: Advanced WordPress Site Speed Optimizations

Having your WordPress site run as fast as possible is important for growing your business.

When each page loads fast, site visitors are more likely to stay and look around instead of leaving. They are more likely to convert into customers.

This detailed White Paper has advanced techniques for speeding up your WordPress websites. Many of these action items may require a web developer. Most of them require you to have full access to your web server and database (not just your WordPress admin).

Table Of Contents

Introduction

Types of Sites: Static or Dynamic

Before we dive into the optimizations, it's important first to understand that **not all WordPress sites are the same**. This is why a lot of users have problems, as you can't go about tackling every issue the same way. It helps to give your WordPress site/s a classification: static or dynamic.

Static Sites

Static would typically include sites such as blogs, small business sites, lower volume news sites, personal, photography, etc. By static, we mean that the data on these WordPress sites is **not changing very often** (perhaps a couple of times a day).

This becomes incredibly important as many of the requests can be served directly from cache on the server at lightning-fast speeds! Don't worry; we'll dive into the topic of caching in length further below. This means they will have fewer database calls and not as many resources will be needed to achieve good performance.

Dynamic Sites

On the other side, we have highly dynamic sites. These include sites such as eCommerce (WooCommerce or Easy Digital Downloads), community, membership, forums (bbPress or BuddyPress) and learning management systems (LMS). By dynamic, we mean that the data on these WordPress sites is f**requently changing** (server transactions are taking place every few minutes or even every few seconds). This means that not all requests to the server can be served directly from cache and require additional server resources and database queries.

These sites also typically have a **large number of concurrent visitors and sessions**. On an informational or corporate WordPress site which is mostly static, a visitor might stay for five or 10 minutes until they find what they need. On dynamic sites, you have the opposite happening. Visitors typically come to the site to engage with something or someone. If they're going through an online course, it's not unusual for them to stay for hours.

Static sites may need less resources, optimizations, and customization than dynamic sites.

Types of Hosting

A WordPress host is a company that stores all of your website's data. You sign up for a plan and all your images, content, etc., reside on a server sitting in the host's data center. The WordPress host gives you an easy way to access the data, manage it, and route it to your visitors. Pretty simple right? Well, not quite.

There are very different types of WordPress hosts you'll encounter around the web. Let's dive into the pros and cons of each. It's important you choose the right one from the beginning, otherwise, you'll simply cause yourself headaches and wasted time down the road.

1. Shared WordPress Hosting

The first and very popular type of WordPress hosting is what most people call "shared hosting." These include companies like Bluehost and HostGator as well as providers like Siteground, GoDaddy, and InMotion Hosting. They typically utilize cPanel, and the average customer usually pays between $3 to $25 a month. Pay close attention to the pricing, shared hosting often offers 'get you in the door' cheaper pricing during the first year (or few months) and doubles after that.

Anyone using this type of hosting will almost certainly, at some point, experience slow performance. It's probably just a matter of time. Why? Because **shared hosts share your server with other sites.** This in turn can (or eventually will) impact the performance of your site. Site suspensions or seeing frequent 500 errors are common things you'll experience as they have to place limits on everything and consolidate resources to survive. Or even worse, you'll experience website downtime. Even though you don't know it, your WordPress site is most likely sitting on the same server as 200+ other people. Any issues that pop up with other sites can trickle over into your site.

No matter how you do the math, after expenses, $3-25 a month isn't generating any revenue for the hosting company. Especially when you attribute support into that. One support ticket and they're already in the red. The way they make a lot of their money is on upselling and hidden fees. These upsells include things like migrations, domain registrations, SSL certificates, etc. Another common tactic, as we said, is to provide huge signup discounts. But once the renewal comes around, you get the bigger bill.

Most of these hosts offer what they call their "unlimited resources" plan. You have probably seen this. Well, there is no such thing in the real world as unlimited resources. What hosts do behind the scenes is throttle the clients using up a lot of the resources. This, in turn, ends up with those angry clients leaving, making room for more clients that don't use a lot of resources. In the end, you have a vicious cycle of the hosting company pushing cheap plans and signing up customers who they hope won't use a lot of resources and will purchase upsells.

Customer service and support with shared hosting is almost always subpar due to the sheer volume of sites vs. support representatives. Shared hosts have to spread themselves very thin to even make a profit and this usually leads to an unpleasant experience for the client.

> ***"When it comes to shared hosting, you usually get what you pay for."***

Make sure to check out an in-depth article from our CFO on the shocking truths behind how cheap WordPress hosting really works.

2. Dedicated or VPS WordPress Hosting

The second type of WordPress hosting is Dedicated or VPS (virtual private server). You aren't sharing server resources with this type of hosting. But it can be a bit DIY (do it yourself). This type of hosting is typically used by bootstrap startups and users with more development, server management, and WordPress skills.

With Dedicated or VPS you get to configure your own server, but you also need the technical know-how to choose the optimal resources, settings, and configurations. If something goes wrong, the tech support doesn't help you with your app (which is WordPress in this case).

The customizable, yet DIY approach, can cut costs, but it also means that you are responsible if something breaks and for optimizing your server for performance. Dedicated and VPS can also backfire on you if you aren't careful. Don't go this route if you aren't tech-savvy or just because you want to tinker! Your time is worth money and we recommend spending it on growing your business.

3. Managed WordPress Hosting

The third type of hosting is managed hosting. These types of hosts handle all the back-end server related tasks for you, along with providing expert support when you need it. They typically fine-tune your server environment to work with WordPress and usually include features such as one-click staging environments and automatic backups. Their support teams will be more knowledgeable when it comes to your app (WordPress) since that is their daily focus.

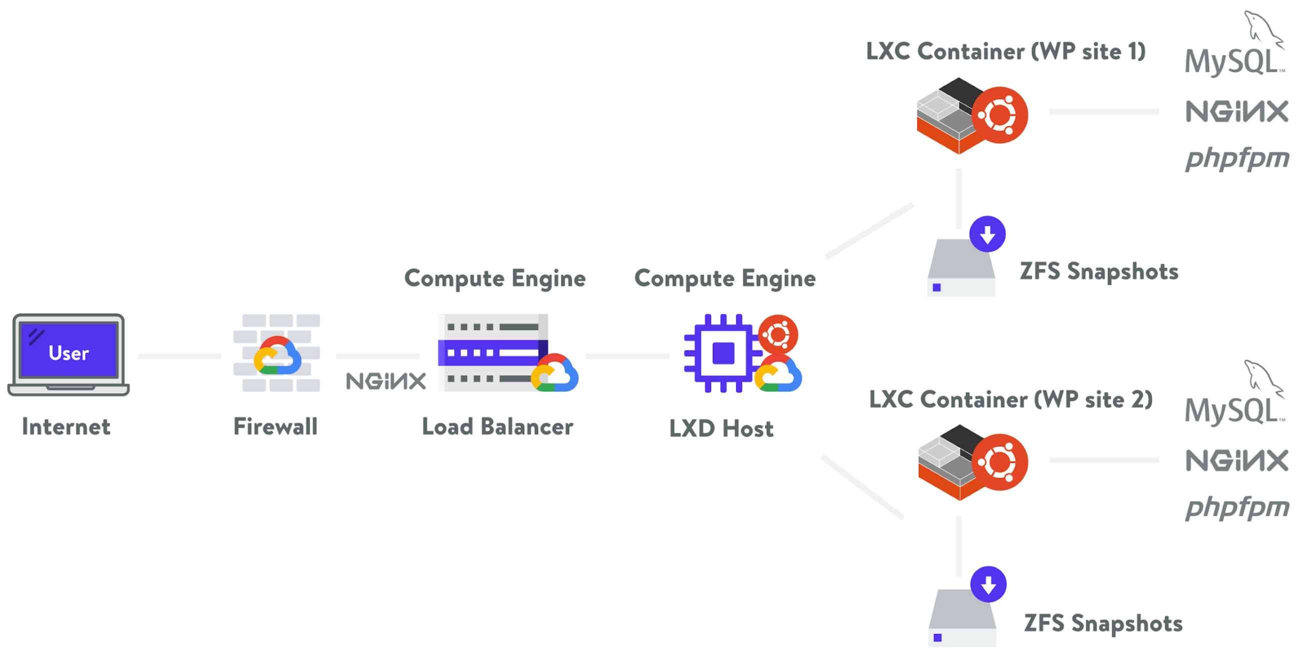
> ***If you want to save time and focus on growing your business, managed WordPress hosting is the way to go!***

Plans for managed WordPress hosting typically range anywhere from $30 to $150 a month or more depending on the size of your site and needs.

Kinsta Takes a Different Approach

Kinsta takes managed WordPress hosting to the next level. Our hosting platform doesn’t quite fit into any of the traditional hosting categories. Our entire infrastructure is built on Google Cloud Platform (GCP) and is different from traditional shared, VPS, or dedicated infrastructure.

Every WordPress site on our platform runs in an isolated software container that contains all of the software resources required to run the site (Linux, NGINX, PHP, MySQL/MariaDB). This means that the software that runs each site is completely private and is not shared, even between your own sites.



Each site container runs on virtual machines in one of our multiple GCP data centers. Each machine has up to 96 CPUs and hundreds of GB of RAM. Hardware resources (RAM/CPU) are allocated to each site container automatically by our virtual machines on an as-needed basis.

Read this article to learn what makes Kinsta different if you want to learn more. No matter who your WordPress host is, the tips in this whitepaper can help you get your website to run as fast as possible.

1: Server Optimizations

The optimizations described below require a lot of technical skill. In many cases your web host controls this, and they may not grant you access to the controls.

The sections below will either show you how to make optimizations to your site, teach you how to ask your technical support team the right questions, or can help you decide if you want to migrate your site to another host that manages these things for you.

1:1 Caching

Caching is by far one of the **most important and easiest ways to speed up WordPress!** But before we show you how to use caching, it's essential first to understand how it works and the different kinds of caching available.

What is Caching?

In short, every web page visited on your WordPress site requires a request to the server, processing by that server (including database queries), and then a final result sent from the server to the user's web browser.

For instance, you might have a header, images, a menu, and a blog post. Since the server has to process all of those requests, it takes some time for the complete web page to be delivered to the user, especially with clunky or larger websites.

That's where a WordPress caching plugin comes into play! Caching instructs the server to store some files to disk or RAM, depending on the configuration. Therefore, it can remember and duplicate the same content it's been serving in the past. Basically, it reduces the amount of work required to generate a page view. As a result, **your web pages load much faster, directly from cache**.

Some other benefits of caching include:

- **Your server uses fewer resources** – This ties into speed, since fewer resources used make for a faster site. However, it also puts less of a strain on your server. This is very important when it comes to highly dynamic sites, such as membership sites, and determining what you can and cannot serve from cache.
- **You'll see lower Time to First Byte (TTFB)** – Caching is one of the easiest ways to lower your TTFB. In fact, in our tests caching typically reduces TTFB by up to 90%!

Types of Caching

When it comes to types of caching, there are several different approaches commonly used:

1. Caching at the Server-Level
2. Caching with a Plugin
3. Edge Caching

1. Caching at the Server-Level

This type of caching is by far one of the easiest approaches for the end-user. What this means is that the WordPress hosting provider handles it for you. At Kinsta, we utilize the following **four types of server-level cache**, which are all automatically done for you.

- Bytecode cache
- Object cache
- Page cache
- CDN cache

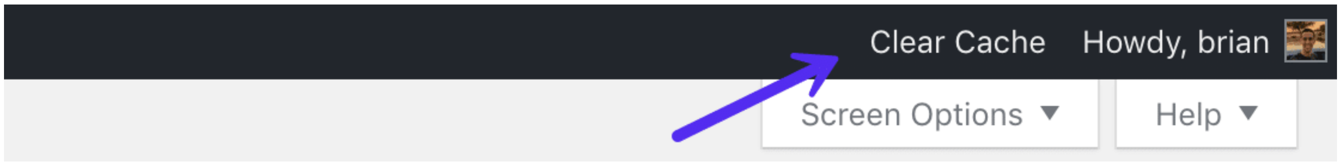
This means you don't need to worry about messing with any complicated and confusing caching plugins. You can stop Googling around for the "best caching plugins" and focus on more productive tasks.

The page cache is configured to work right out of the box with standard WordPress. You don't have to do a thing! Simply launch your WordPress site and page caching will start happening.

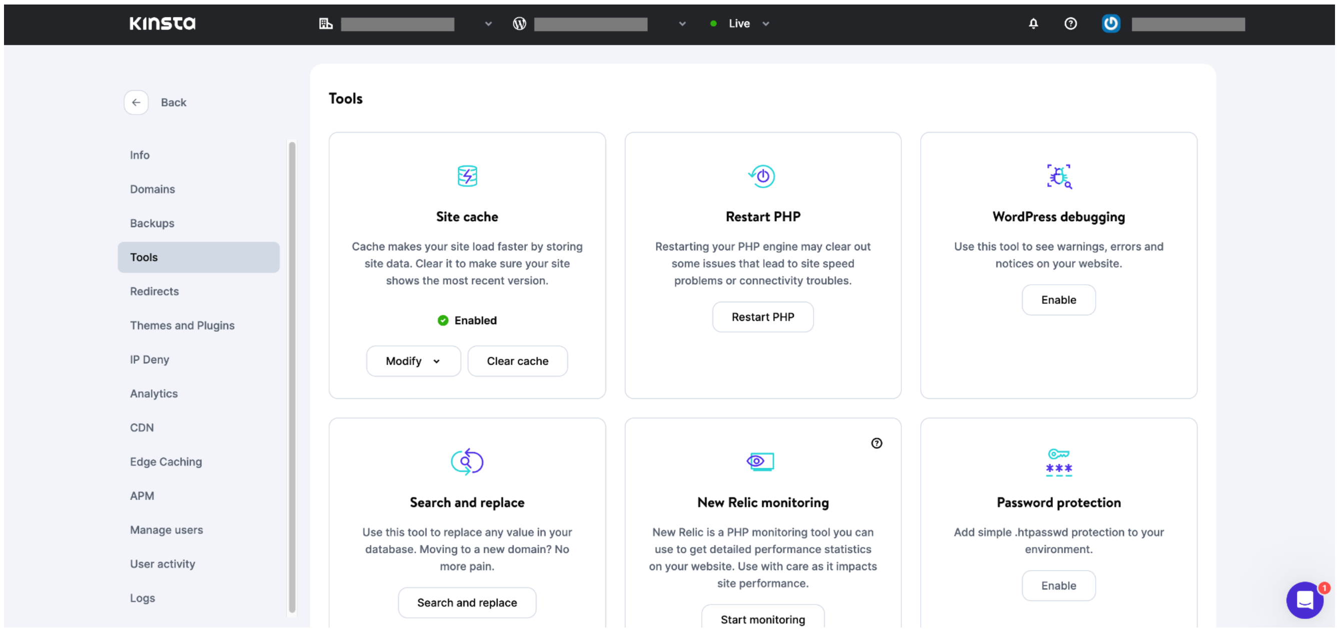
Some web hosts, Kinsta included, also have caching rules in place for ecommerce sites such as WooCommerce and Easy Digital Downloads. By default, certain pages that should never be cached, such as cart, my-account, and checkout, are excluded from caching.

Users automatically bypass the cache when the *woocommerce_items_in_cart* cookie or *edd_items_in_cart* are detected to ensure a smooth and in-sync checkout process.

You can easily clear your WordPress site's cache at any time from the admin toolbar.



It’s also integrated into our MyKinsta dashboard. Just click into **Tools** > **Site Cache** and click on **Clear cache**.


KINSTA
Live
Back
Info
Domains
Backups
Tools
Redirects
Themes and Plugins
IP Deny
Analytics
CDN
Edge Caching
APM
Manage users
User activity
Logs
Tools
Site cache
Cache makes your site load faster by storing site data. Clear it to make sure your site shows the most recent version.
Enabled
Modify
Clear cache
Restart PHP
Restarting your PHP engine may clear out some issues that lead to site speed problems or connectivity troubles.
Restart PHP
WordPress debugging
Use this tool to see warnings, errors and notices on your website.
Enable
Search and replace
Use this tool to replace any value in your database. Moving to a new domain? No more pain.
Search and replace
New Relic monitoring
New Relic is a PHP monitoring tool you can use to get detailed performance statistics on your website. Use with care as it impacts site performance.
Start monitoring
Password protection
Add simple .htpasswd protection to your environment.
Enable

2. Caching with a Plugin

If your hosting provider doesn’t provide caching, you can use a third-party WordPress caching plugin. Based on our experience, we recommend one of the following:

- WP Rocket (premium)
- Cache Enabler (free)
- W3 Total Cache (free)

You can also check out some additional options in our in-depth post on WordPress caching plugins.

At Kinsta, in addition to our server-level caching, we also fully support WP Rocket. We usually don't allow caching plugins in our environment because they conflict with our built-in caching solution. However, as of WP Rocket 3.0, their page caching functionality will automatically be disabled when running on Kinsta servers. This allows Kinsta clients to use our fast server-level caching but still take advantage of the fantastic optimization features WP Rocket has to offer.

3. Edge Caching

Edge Caching is a newer type of caching technology. It stores your often-updated HTML text content as close as possible to your visitors "on the edge", or on a global network of servers. Edge Caching requires advanced functionality that dynamically updates the HTML content and replicates it all over the world anytime it's changed.

It would be very difficult to build this yourself. Instead, you should use a service like Cloudflare with their extensive network of Edge Servers. Ideally, your web host will provide this service integrated into their servers due to the difficulty in setting it up.

- Read this article to learn more about Edge Caching

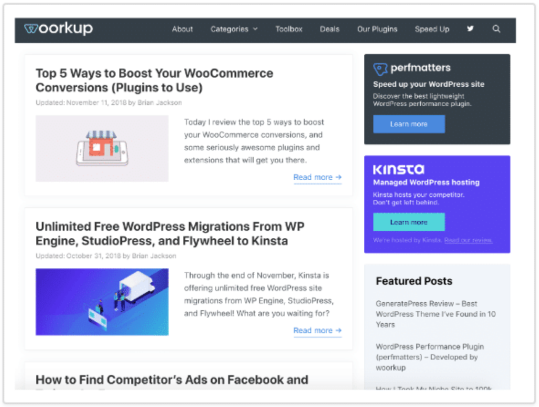
No Caching vs. Caching

How much does caching help? The proof is in the pudding.

We ran a few speed tests with Kinsta's server-level caching so you can see the difference it makes, both in terms of overall speed and TTFB.

No Caching

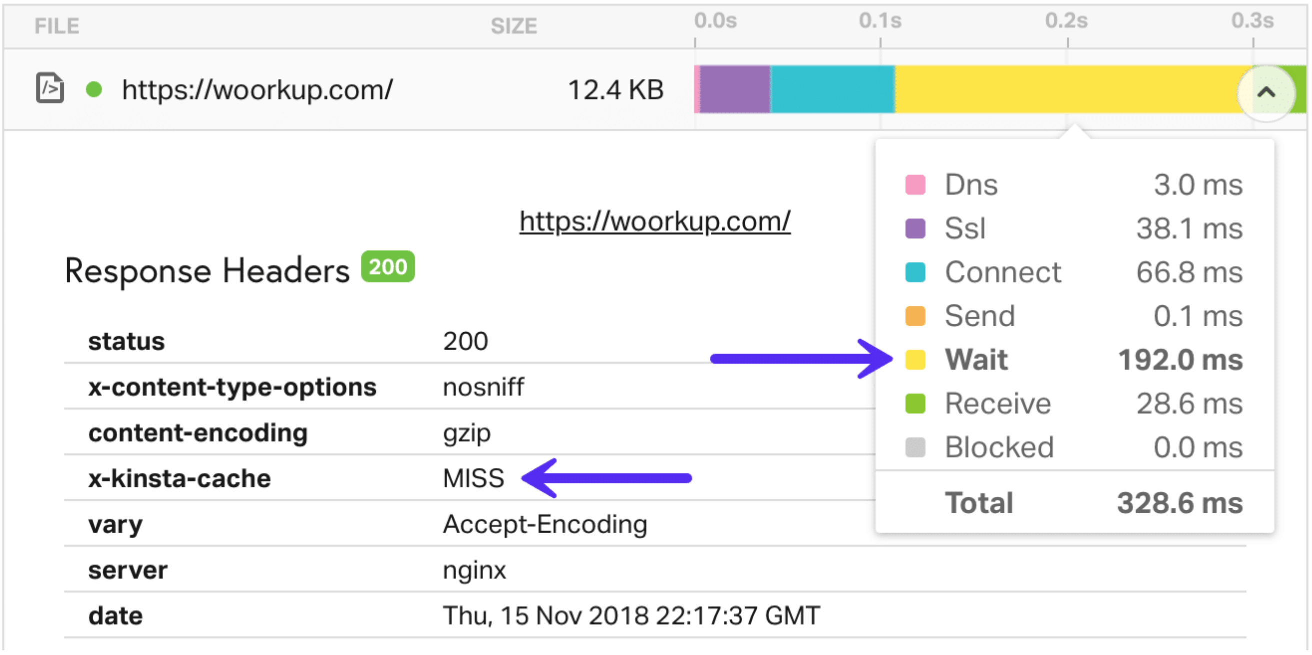
We first ran five tests on Pingdom without caching enabled and took the average.



Performance grade	Page size	Load time	Requests
B 89	230.6 KB	507 ms	40

No Caching TTFB

It’s also important to note the difference in TTFB without and with caching. TTFB in Pingdom is represented by the yellow “waiting” bar. As you can see the TTFB with no caching is 192 ms. You can see that it’s not serving from cache as the *x-kinsta-cache* header is showing a *MISS*.



FILE	SIZE
https://woorkup.com/	12.4 KB

🟪 Dns	3.0 ms
🟪 Ssl	38.1 ms
🟦 Connect	66.8 ms
🟧 Send	0.1 ms
🟨 **Wait**	**192.0 ms**
🟩 Receive	28.6 ms
⬜ Blocked	0.0 ms
Total	**328.6 ms**

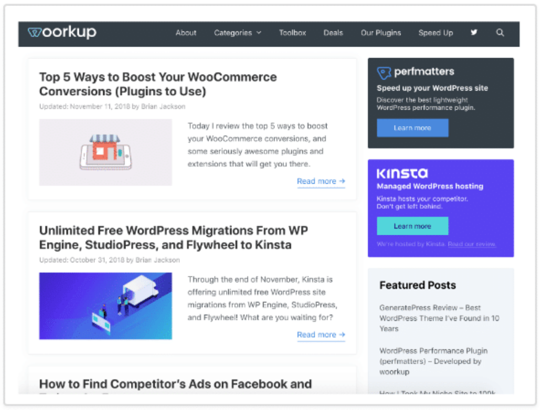
https://woorkup.com/

Response Headers 200

status	200
x-content-type-options	nosniff
content-encoding	gzip
x-kinsta-cache	MISS
vary	Accept-Encoding
server	nginx
date	Thu, 15 Nov 2018 22:17:37 GMT

With Caching Enabled

We then enabled server-level caching and ran five tests on Pingdom and took the average.

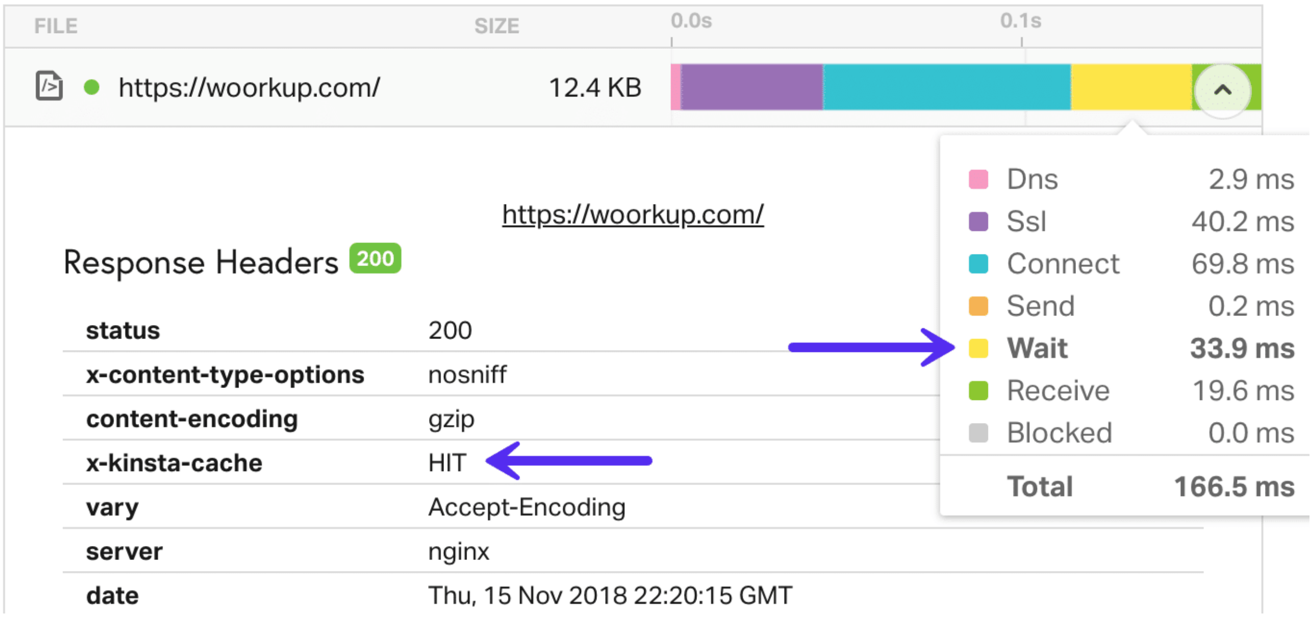


Performance grade	Page size
B 89	230.6 KB
Load time	**Requests**
351 ms	40

As you can see server-level **caching decreased our page load time by 33.77%!** And that's without any extra work involved. This site we tested is also fairly optimized, so larger unoptimized sites are bound to see even greater differences.

TTFB with Caching Enabled

Now if we take a look at the TTFB with caching enabled, we can see that it's under 35 ms. You can see that it's serving from cache as the *x-kinsta-cache* header is showing a *HIT*.



FILE	SIZE
https://woorkup.com/	12.4 KB

🟪	Dns	2.9 ms
🟪	Ssl	40.2 ms
🟦	Connect	69.8 ms
🟧	Send	0.2 ms
🟨	**Wait**	**33.9 ms**
🟩	Receive	19.6 ms
⬜	Blocked	0.0 ms
	Total	**166.5 ms**

https://woorkup.com/

Response Headers 200

status	200
x-content-type-options	nosniff
content-encoding	gzip
x-kinsta-cache	HIT
vary	Accept-Encoding
server	nginx
date	Thu, 15 Nov 2018 22:20:15 GMT

CDN cache is also equally as important as cache from your WordPress host. We'll dive more into CDNs further below.

> ***WordPress caching can easily decrease your page load times by over 33%!***

Issues with Caching and Membership Sites

Membership sites contain a lot of **uncacheable content** and pages that are continuously changing. Things such as the login page for community members (which could be getting hit constantly depending on the size of the site), checkout pages for digital goods or courses, and discussion boards are common culprits and pain points, as these cannot typically be cached.

However, it doesn't end there. On standard WordPress sites, the WordPress dashboard is also not cached for **"logged-in" users**. This is fine when you have just a few authors and admins, but when you suddenly have thousands of members using the dashboard, this immediately causes performance issues as none of it can serve from the cache on the server. This means you need the power and architecture behind the scenes to back it up. Shared hosting providers will usually cripple under these circumstances.

Object Caching for Highly Dynamic Sites

When it comes to WordPress membership sites, your common caching setups are usually not enough as they don't always take full advantage of it. This is where **object caching comes into play**.

Object cache stores the results of database queries so that the next time that particular bit of data is needed it can be delivered from cache without querying the database. This speeds up PHP execution times and reduces the load on your database. This becomes extremely important with membership sites! With WordPress, you can implement object caching in a couple of different ways:

1. A third-party caching solution such as W3 Total Cache
2. **Redis (recommended)**
3. Memcached

We offer Redis as an add-on at Kinsta so you can take full advantage of persistent object caching for your membership sites.

1:2 CDN

CDN is short for content delivery network. This is a network of servers located around the earth, each one called a point-of-presence, or PoP. They are designed to host and deliver copies of your WordPress site's static (and sometimes dynamic) content such as images, CSS, JavaScript, and video streams.

First off, you don't want to get a CDN confused with your WordPress host. These are entirely separate services. A CDN isn't a replacement for your hosting provider, but rather an additional way to increase the speed of your site.

How a CDN Works

How does a CDN work exactly? Well, for example, when you host your website with Kinsta you have to choose a physical data center location, such as the USA, Europe, Asia-Pacific, or South America.

Let's say you choose US Central. This means your website is physically located on a "host server" in Council Bluffs, Iowa. When people over in Europe visit your website it is going to take longer for it to load versus someone visiting it from say Dallas, TX.

Why? Because the data has to travel a further distance. This is what is known as latency. Latency refers to the time and or delay that is involved in the transmission of data over a network. The further the distance the greater the latency.

A CDN lets you store certain assets, like CSS, Javascript files, and images closer to the user on a global network of servers. This makes sites load faster for users all over the world.

Types of CDNs

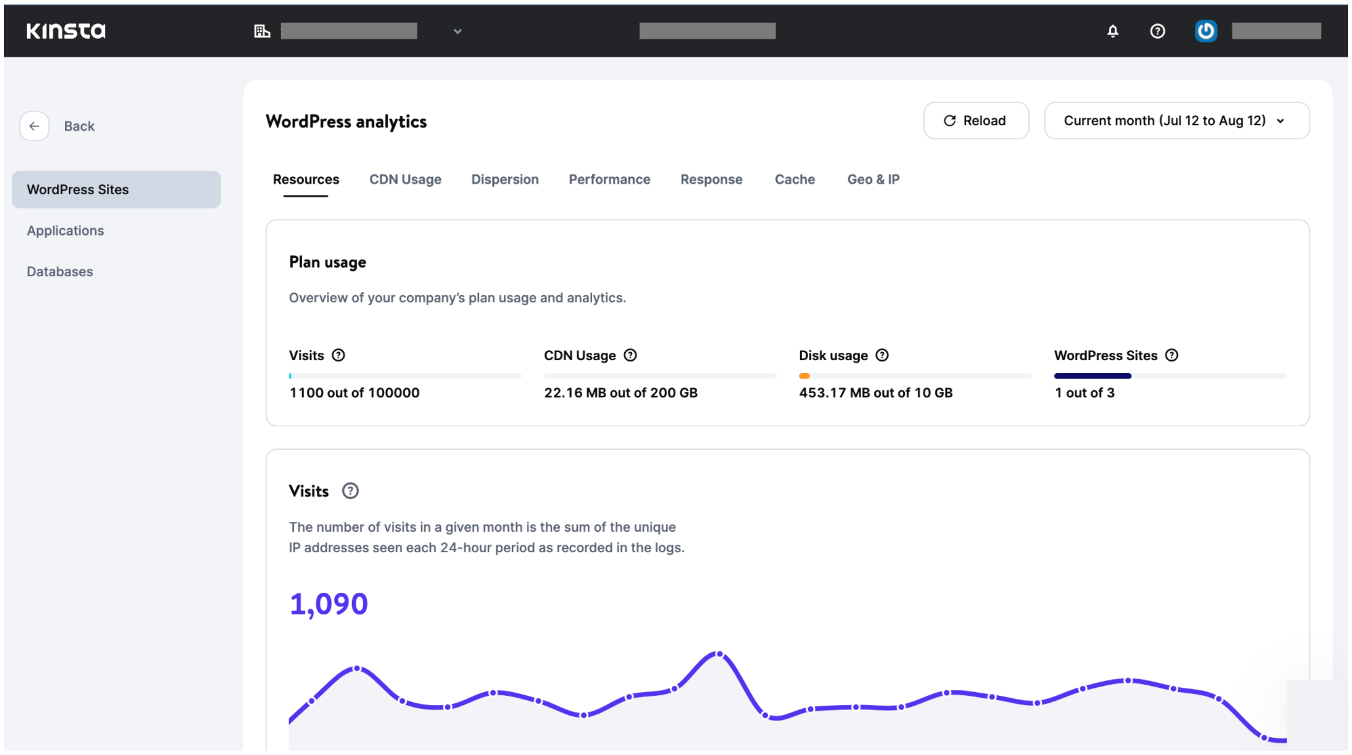
There are two different main types of content delivery networks:

1. Traditional Pull CDN
2. Reverse Proxy CDN

Traditional pull CDNs cache a copy of all of your content and media, but a request from the client is still made directly to your hosting provider. KeyCDN and CDN77 are examples of traditional CDNs.

A reverse proxy CDN is slightly different. While it still acts like a CDN, it intercepts all incoming requests and acts as an intermediary server between the client and your host. Cloudflare and Sucuri are examples of reverse proxy CDNs. This is one reason why you have to point your DNS directly to these providers instead of your host.

The benefit of these is because they act as an intermediary server, they can provide strong web application firewalls which can help block the bad traffic from ever hitting your WordPress site and or hosting provider. One downfall to this is that they do come with a little additional overhead in terms of performance compared to a traditional pull CDN. But with additional performance and security features, this could be argued as negligible.



WordPress analytics

Plan usage

Overview of your company's plan usage and analytics.

Visits	CDN Usage	Disk usage	WordPress Sites
1100 out of 100000	22.16 MB out of 200 GB	453.17 MB out of 10 GB	1 out of 3

Visits

The number of visits in a given month is the sum of the unique IP addresses seen each 24-hour period as recorded in the logs.

1,090

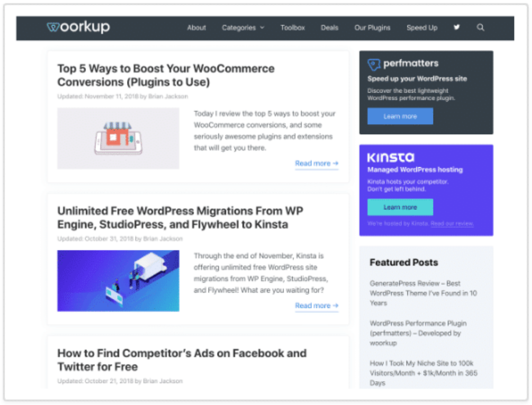
CDN Speed Tests

Earlier we talked about the huge benefits of WordPress caching. Well, CDN caching is also super powerful. This is because CDNs typically have a lot more server locations than hosting providers. This means they can cache all of your assets (images, JS, CSS) closer to your visitors and serve them up at lightning-fast speeds.

Let's do a few quick tests to see just how much faster your site could be with a CDN.

Without CDN

Our test website is hosted at Kinsta and is physically located at the Iowa, USA data center. We first ran five speed tests in Pingdom (without the CDN enabled), and took the average. Important: We are using the Europe – United Kingdom – London location at Pingdom to demonstrate the real power of a CDN. The total load time was **1.03 sms**.



Performance grade

B 89

Page size

231.0 KB

Load time

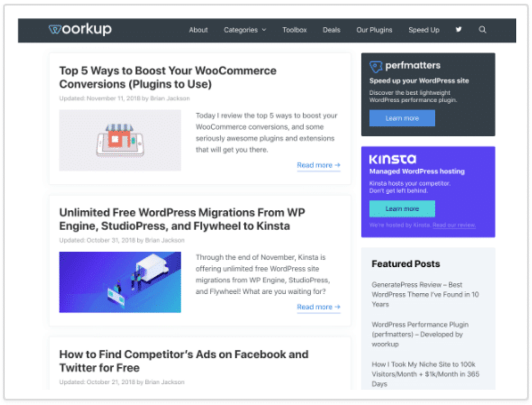
1.03 s

Requests

40

With CDN

We then enabled our CDN and ran five additional speed tests in Pingdom. Our total load time is now **585 ms** from the Europe – United Kingdom – London Pingdom test location. So by using the CDN, we were able to **decrease our page load times by 43.2%!** That is huge.



Performance grade

B 89

Page size

230.6 KB

Load time

585 ms

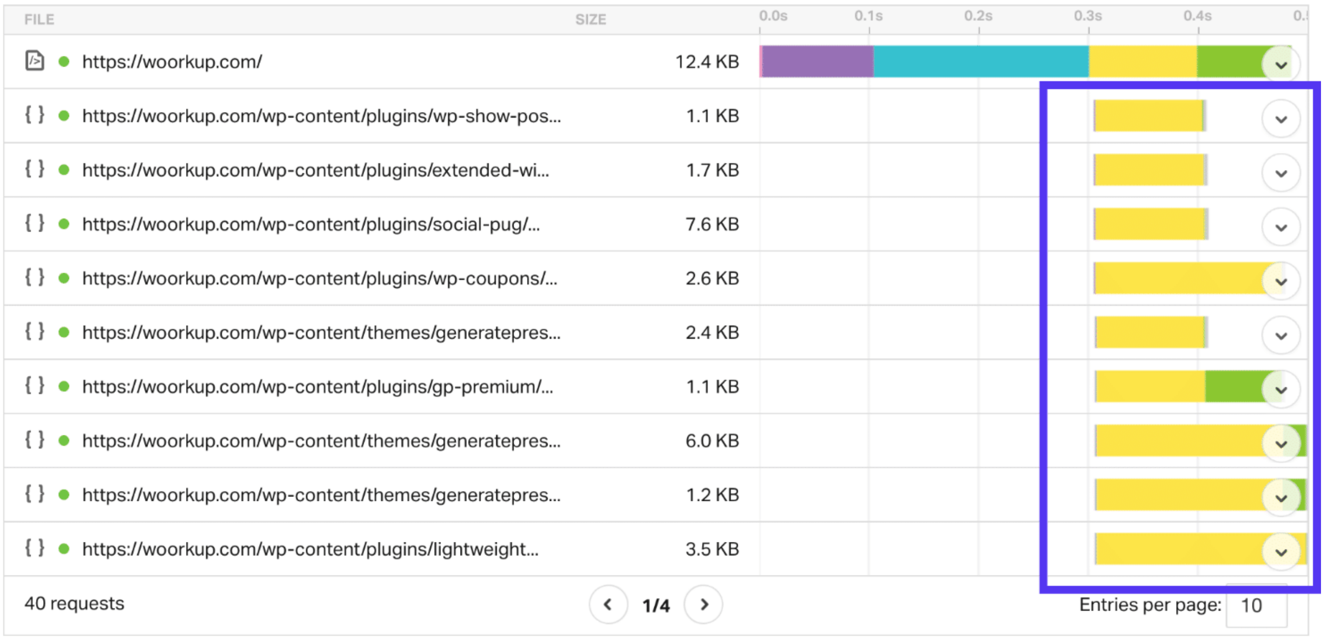
Requests

40

The reason for such a drastic difference is because the CDN has a data center in London. This means all the assets are cached in that location and ready to be served with minimal latency.

TTFB without CDN

Remember that the yellow bar in Pingdom stands for wait time, which is time to first byte (TTFB). On our speed tests without the CDN running the average TTFB on assets was around 98 ms.

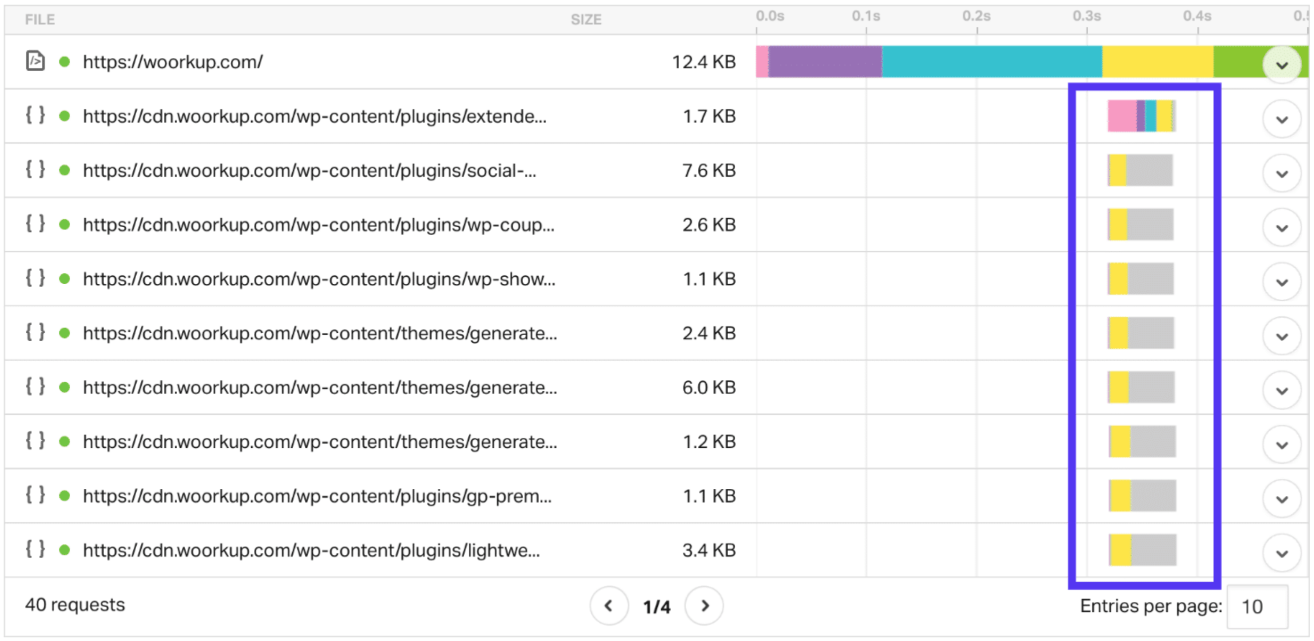


FILE	SIZE
https://woorkup.com/	12.4 KB
https://woorkup.com/wp-content/plugins/wp-show-pos...	1.1 KB
https://woorkup.com/wp-content/plugins/extended-wi...	1.7 KB
https://woorkup.com/wp-content/plugins/social-pug/...	7.6 KB
https://woorkup.com/wp-content/plugins/wp-coupons/...	2.6 KB
https://woorkup.com/wp-content/themes/generatepres...	2.4 KB
https://woorkup.com/wp-content/plugins/gp-premium/...	1.1 KB
https://woorkup.com/wp-content/themes/generatepres...	6.0 KB
https://woorkup.com/wp-content/themes/generatepres...	1.2 KB
https://woorkup.com/wp-content/plugins/lightweight...	3.5 KB

40 requests 1/4 Entries per page: 10

TTFB with CDN

Once we enabled the CDN, the average TTFB on assets dropped to an average of **15 ms**. So by using a CDN **our average TTFB dropped by 84.69%**. This is primarily because the assets were being served directly from the CDN's cache.



FILE	SIZE
https://woorkup.com/	12.4 KB
https://cdn.woorkup.com/wp-content/plugins/extende...	1.7 KB
https://cdn.woorkup.com/wp-content/plugins/social-...	7.6 KB
https://cdn.woorkup.com/wp-content/plugins/wp-coup...	2.6 KB
https://cdn.woorkup.com/wp-content/plugins/wp-show...	1.1 KB
https://cdn.woorkup.com/wp-content/themes/generate...	2.4 KB
https://cdn.woorkup.com/wp-content/themes/generate...	6.0 KB
https://cdn.woorkup.com/wp-content/themes/generate...	1.2 KB
https://cdn.woorkup.com/wp-content/plugins/gp-prem...	1.1 KB
https://cdn.woorkup.com/wp-content/plugins/lightwe...	3.4 KB

40 requests 1/4 Entries per page: 10

A CDN decreased our page load times by 43.2%! Check out why you should be using one.

How to Enable a CDN

Enabling a CDN on your WordPress site doesn't have to be hard, it's quite easy! Just follow these steps.

Step 1

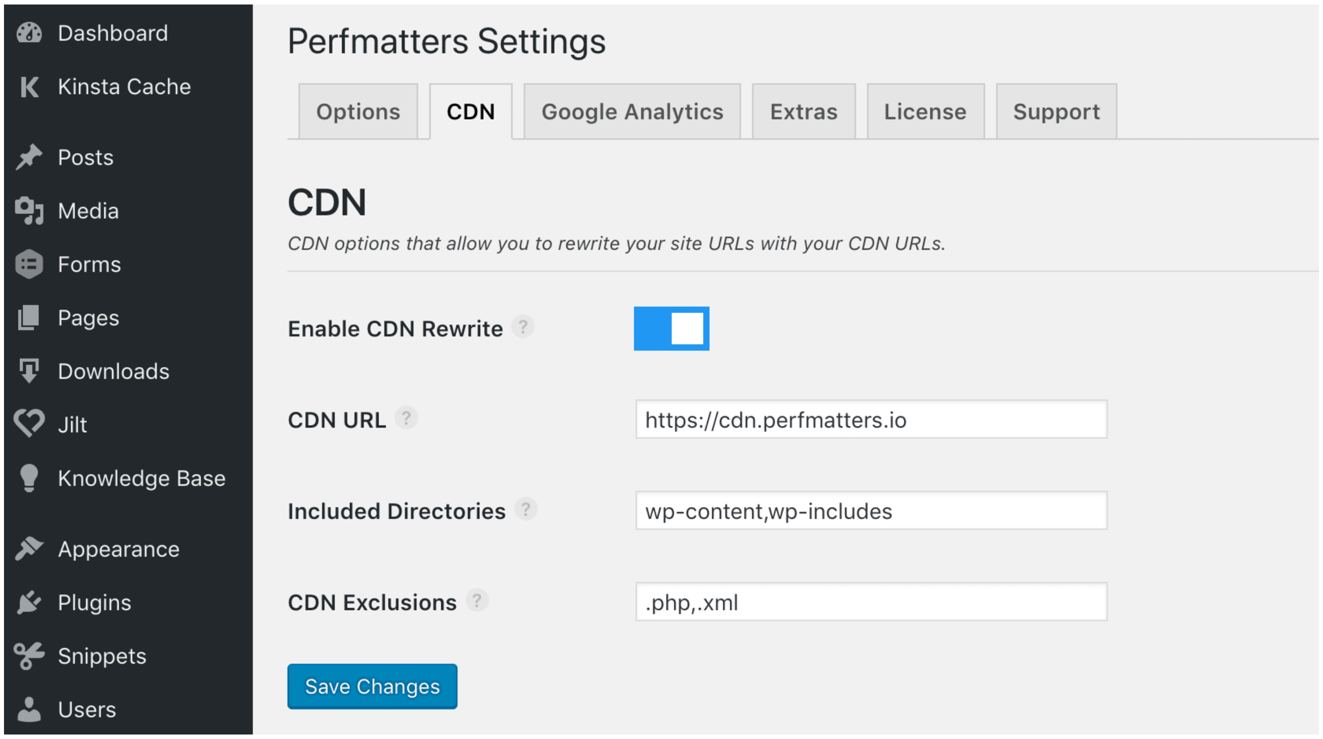
Select a CDN provider and subscribe to their service. These are typically billed on a monthly basis or by data usage. Most providers will have a calculator to estimate your costs.

- If you are looking into deploying KeyCDN yourself, we recommend reading this article on CDN for dummies. Each CDN provider should also have documentation to help you get started.
- We have in-depth tutorials on how to install Cloudflare and how to install Sucuri.

Step 2

If you're using a traditional pull CDN, you can utilize a free plugin like CDN Enabler, WP Rocket, or Perfmatters to integrate it with your WordPress site. These plugins automatically link up your assets to the CDN. There is no work needed on your part to get your content on the CDN; this is all hands-off!

Reverse Proxy CDNs typically don't require any plugins, although sometimes they have them to enable additional features.



Perfmatters Settings

CDN

CDN options that allow you to rewrite your site URLs with your CDN URLs.

Enable CDN Rewrite

CDN URL: https://cdn.perfmatters.io

Included Directories: wp-content,wp-includes

CDN Exclusions: .php,.xml

How to Use CDN

Did you like the numbers in those CDN speed tests above? Use a CDN yourself!

- Learn more about CDNs
- Learn how to set up a Cloudflare CDN
- Learn how to set up a Sucuri CDN

At Kinsta we include a powerful CDN which is enabled by default, learn more about it here.

Additional CDN Optimizations

Here are a few additional CDN optimizations you might want to check out or think about.

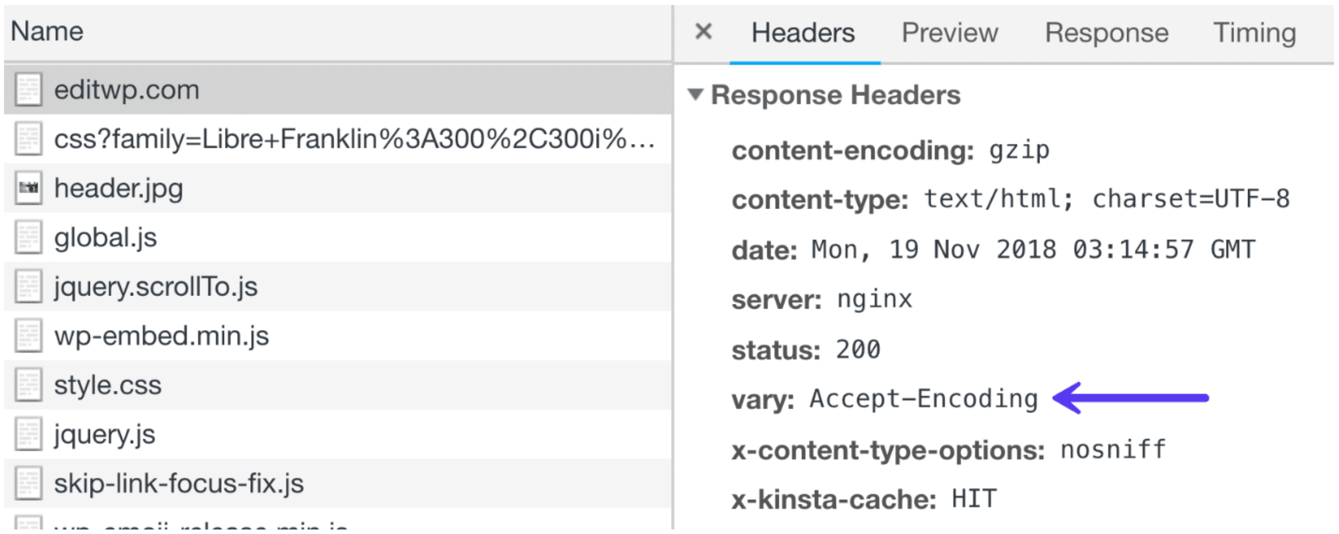
- If you have a lot of comments, gravatars can generate a lot of requests. They load from *secure.gravatar.com*. Check out this tutorial on how to load gravatars from your CDN instead. We do this on the Kinsta website.

- You can host your custom web fonts from your CDN or even Google fonts on your CDN. Check out our in-depth tutorial on local fonts.
- Make sure to load your favicon from your CDN. Even though it's small, every request counts!

1:3 GZIP Compression | Add a "Vary: Accept-Encoding" Header

The **vary: Accept-Encoding** header should be included on every origin server response, as it tells the browser whether or not the client can handle compressed versions of the content. If this isn't properly set, you might see a warning that you need to "Specify a Vary: Accept-Encoding Header."

For example, let's say you have an old browser without gzip compression and a modern browser with it. If you don't utilize the *vary: Accept-Encoding* header your web server or CDN could cache the uncompressed version and deliver that to the modern browser by mistake, which in turn hurts the performance of your WordPress site. By using the header you can ensure that your web server and or CDN delivers the appropriate version.



```
content-encoding: gzip
content-type: text/html; charset=UTF-8
date: Mon, 19 Nov 2018 03:14:57 GMT
server: nginx
status: 200
vary: Accept-Encoding
x-content-type-options: nosniff
x-kinsta-cache: HIT
```

Kinsta automatically adds the above headers on all server requests, and if you're using a CDN, they will most likely add these headers for you as well. Just like with the other cache headers we've discussed above, you can't manually set this header on external resources.

Add Vary: Accept-Encoding Header in Apache

You can add the *vary: Accept-Encoding* header in Apache by adding the following to your *.htaccess* file.

```
<IfModule mod_headers.c>
  <FilesMatch ".(js|css|xml|gz|html)$">
    Header append Vary: Accept-Encoding
  </FilesMatch>
</IfModule>
```

Add Vary: Accept-Encoding Header in Nginx

You can add the vary: Accept-Encoding header in Nginx by adding the following code to your config file. All Nginx configuration files are located in the */etc/nginx/* directory. The primary configuration file is */etc/nginx/nginx.conf*.

```
gzip_vary on
```

1:4 Clean Up and Optimize Your Database

Next up are some tips on how to fine-tune your WordPress database. Just like a car your database needs upkeep as over time it can become bloated.

Membership sites especially make it tricky, as they usually **generate more complex queries**, which in turn adds additional latency in retrieving the information from the MySQL database. A lot of this is due to all the additional moving parts and large amounts of data sites like these have. This might also be caused by sites that heavily rely on search queries for navigation or use *WP_Query*.

Not to mention, you also have large amounts of concurrent users continuously querying the database.

1:4.1 Use the InnoDB MySQL Storage Engine

A lot of older sites are still using the MyISAM storage engine in their database. Over recent years, InnoDB has shown to perform better and be more reliable.

Here are a couple of advantages of InnoDB over MyISAM:

- InnoDB has **row-level locking**. MyISAM only has full table-level locking. This allows your queries to process faster.
- InnoDB has what is called referential integrity which involves supporting **foreign keys** (RDBMS) and relationship constraints, MyISAM does not (DMBS).
- InnoDB supports **transactions**, which means you can commit and roll back. MyISAM does not.
- InnoDB is more reliable as it uses transactional logs for auto recovery. MyISAM does not.

So now you might be wondering, are you running InnoDB or MyISAM? If you are running on a fairly new WordPress site chances are you are already using the InnoDB MySQL storage engine. But with older WordPress sites you might want to do a quick check. Some sites might even have mixed and matched MyISAM and InnoDB tables, in which you could see improvements by converting them all over.

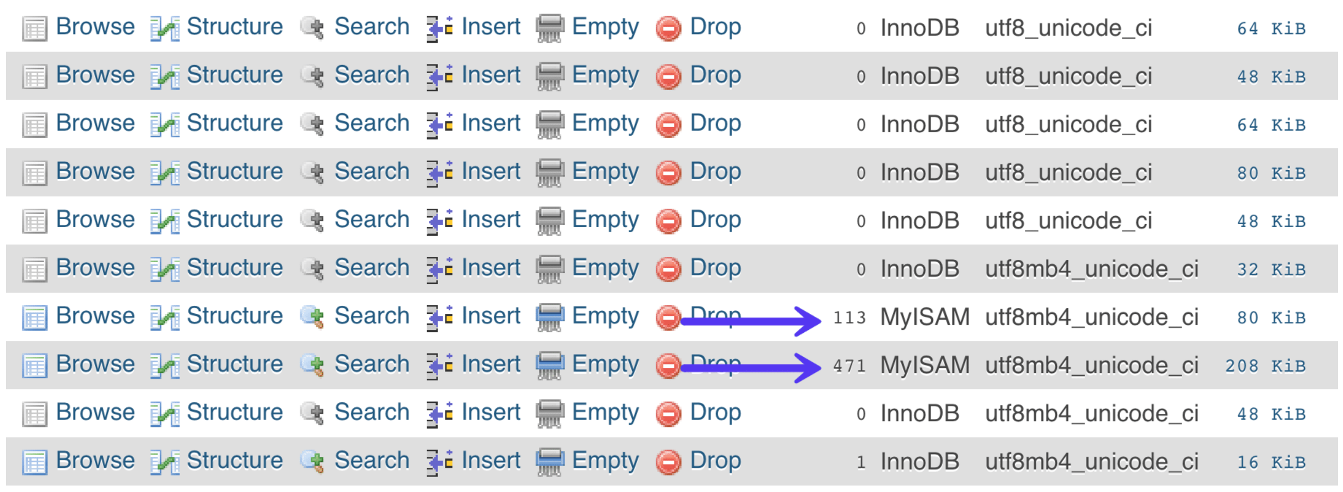
Follow these simple steps below to check.

Step 1

Login to phpMyAdmin and click on your MySQL database.

Step 2

Do a quick scan or sort of the "Type" column, and you can see which Storage Engine types your tables are using. In this example below, you can see that two of the tables are still using MyISAM.



Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	utf8_unicode_ci	64 KiB
Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	utf8_unicode_ci	48 KiB
Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	utf8_unicode_ci	64 KiB
Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	utf8_unicode_ci	80 KiB
Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	utf8_unicode_ci	48 KiB
Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	utf8mb4_unicode_ci	32 KiB
Browse	Structure	Search	Insert	Empty	Drop	113	MyISAM	utf8mb4_unicode_ci	80 KiB
Browse	Structure	Search	Insert	Empty	Drop	471	MyISAM	utf8mb4_unicode_ci	208 KiB
Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	utf8mb4_unicode_ci	48 KiB
Browse	Structure	Search	Insert	Empty	Drop	1	InnoDB	utf8mb4_unicode_ci	16 KiB

If you found some, then it's probably time to move them to InnoDB. We always recommend reaching out to your host and asking if they can do this for you. At Kinsta, every client's database tables automatically get converted to InnoDB by our migration team.

But you can always follow these tutorials below to convert your MyISAM tables to InnoDB manually:

- Convert MyISAM to InnoDB with phpMyAdmin
- Convert MyISAM to InnoDB with WP-CLI

1:4.2 Clean up Your wp_options Table and Autoloaded Data

The *wp_options* table often gets overlooked when it comes to overall WordPress and database performance. Especially on older and large sites, this can easily be the culprit for slow query times on your site due to autoloaded data that is left behind from third-party plugins and themes. Trust us; we see this every single day!

The wp_options table contains all sorts of data for your WordPress site such as:

- Site URL, home URL, admin email, default category, posts per page, time format, etc
- Settings for plugins, themes, widgets

- Temporarily cached data

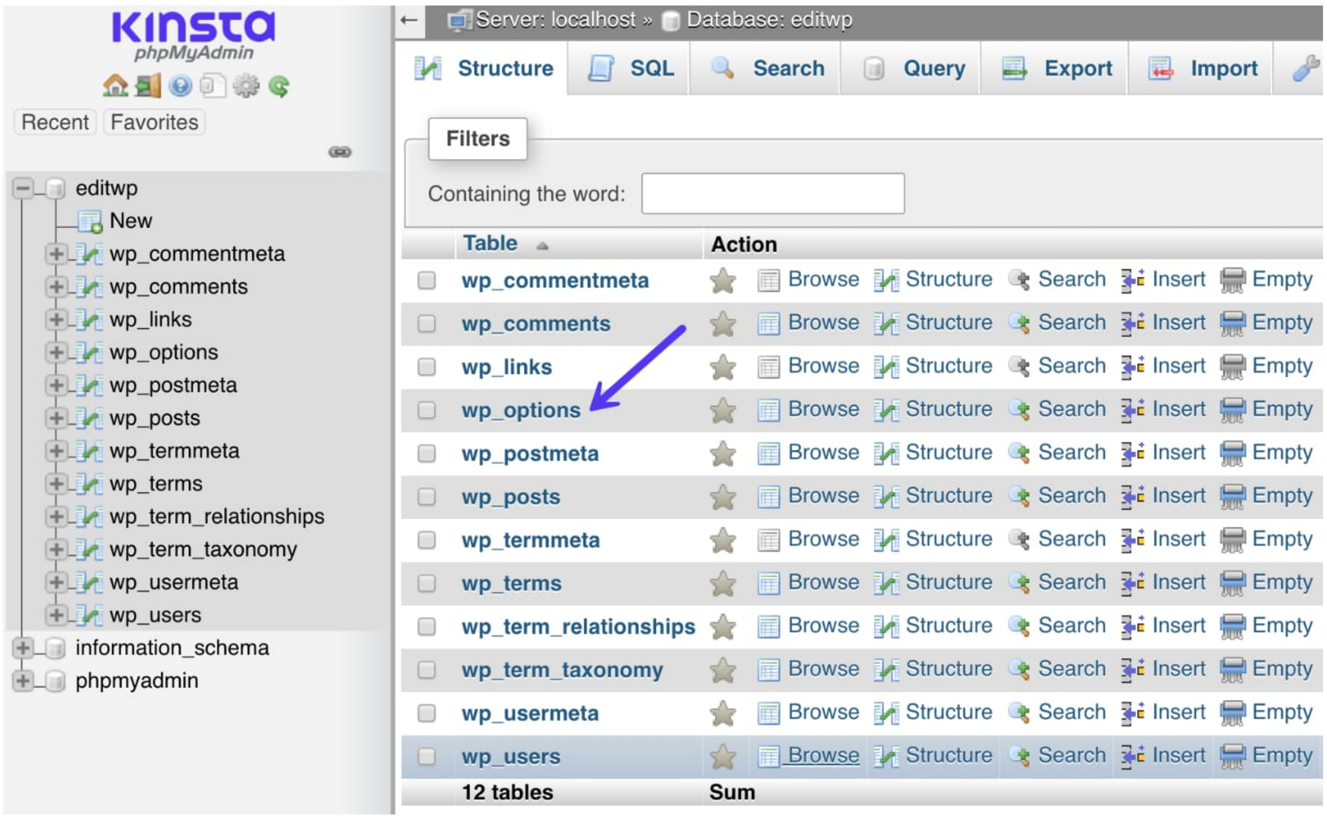
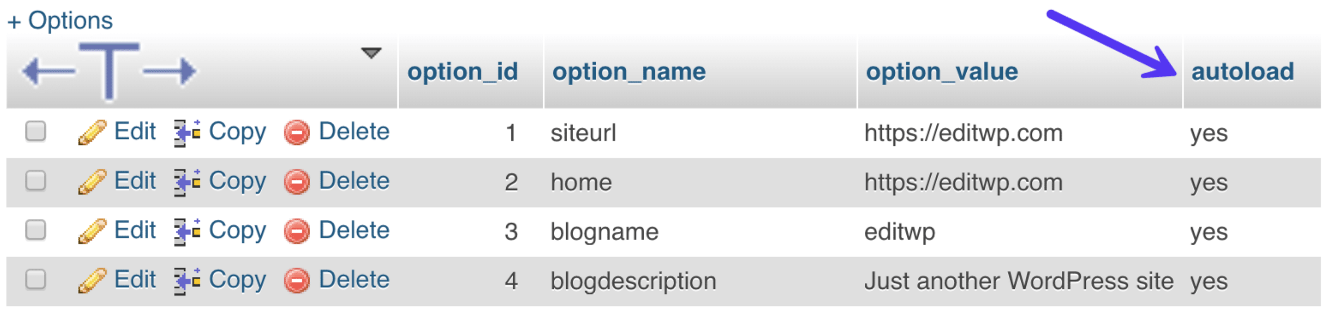


	Table	Action					
☐	wp_commentmeta	☆	Browse	Structure	Search	Insert	Empty
☐	wp_comments	☆	Browse	Structure	Search	Insert	Empty
☐	wp_links	☆	Browse	Structure	Search	Insert	Empty
☐	wp_options	☆	Browse	Structure	Search	Insert	Empty
☐	wp_postmeta	☆	Browse	Structure	Search	Insert	Empty
☐	wp_posts	☆	Browse	Structure	Search	Insert	Empty
☐	wp_termmeta	☆	Browse	Structure	Search	Insert	Empty
☐	wp_terms	☆	Browse	Structure	Search	Insert	Empty
☐	wp_term_relationships	☆	Browse	Structure	Search	Insert	Empty
☐	wp_term_taxonomy	☆	Browse	Structure	Search	Insert	Empty
☐	wp_usermeta	☆	Browse	Structure	Search	Insert	Empty
☐	wp_users	☆	Browse	Structure	Search	Insert	Empty
	12 tables	Sum					

- editwp
  - New
  - wp_commentmeta
  - wp_comments
  - wp_links
  - wp_options
  - wp_postmeta
  - wp_posts
  - wp_termmeta
  - wp_terms
  - wp_term_relationships
  - wp_term_taxonomy
  - wp_usermeta
  - wp_users
- information_schema
- phpmyadmin

This table contains the following fields (columns):

- option_id
- option_name
- option_value
- **autoload** (this is the one we care about when it comes to performance)



+ Options

				option_id	option_name	option_value	autoload
☐	Edit	Copy	Delete	1	siteurl	https://editwp.com	yes
☐	Edit	Copy	Delete	2	home	https://editwp.com	yes
☐	Edit	Copy	Delete	3	blogname	editwp	yes
☐	Edit	Copy	Delete	4	blogdescription	Just another WordPress site	yes

One of the important things to understand about the *wp_options* table is the **autoload** field. This contains a yes or a no value (flag). This essentially controls whether or not it is loaded by the wp_load_alloptions() function. Autoloaded data is **data that is loaded on every page** of your WordPress site. Just like we showed you how to disable certain scripts from loading sitewide, the same idea applies here. The autoload attribute is set to “yes” by default for developers, but not every plugin should theoretically load their data on every page.

The problem WordPress sites can run into is when there is a large amount of autoloaded data in the *wp_options* table. This is typically a result of the following:

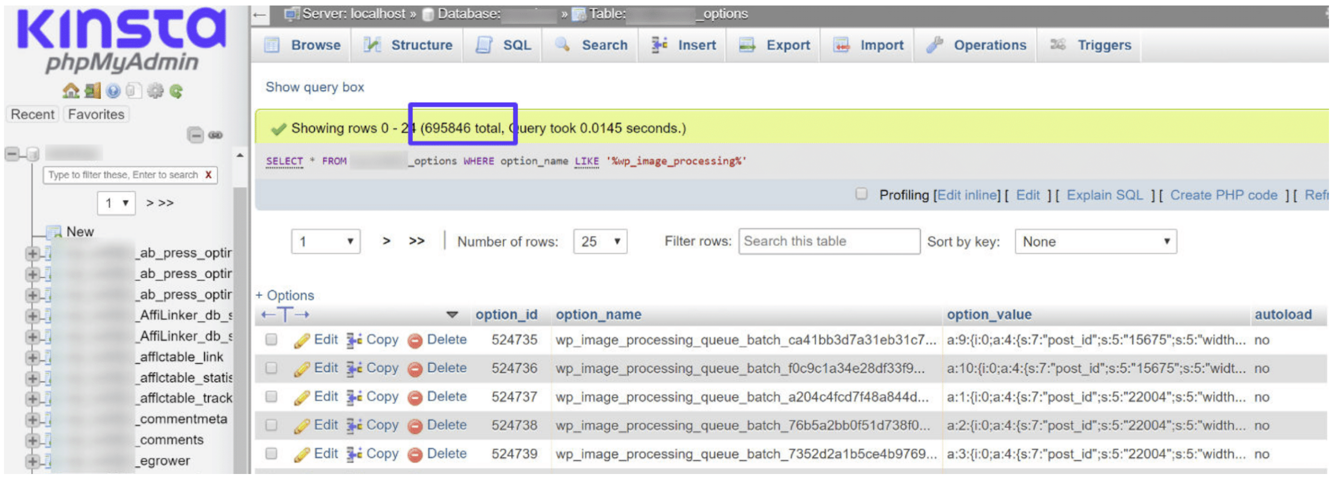
- Data is being autoloaded by a plugin when it should be set to “no.” A good example of this would be a contact form plugin. Does it need to load data on every page or just the contact page?
- Plugins or themes have been removed from the WordPress site, but their options are still left behind in the *wp_options* table. This could mean unnecessary autoloaded data is getting queried on each request.
- Plugin and theme developers are loading data into the *wp_options* table instead of utilizing their own tables. There are arguments to both sides of this, as some developers prefer plugins that don’t create additional tables. However, the *wp_options* table also wasn’t designed to hold thousands of rows.

How much is too much-autoloaded data? This can vary of course, but ideally, you want this to be between 300 KB to 1MB. Once you start approaching the 3-5 MB range or more, there are most likely things that can be optimized or removed from being autoloaded. And anything above 10 MB should be addressed right away. This doesn’t always mean it’s going to cause an issue, but it’s a good place to start.

Because this is such a problem we have a whole separate tutorial you’ll want to read on how to best troubleshoot autoloaded data as well as how to clean it up.

1:4.3 Clean up Transients

Unless you’re using an object cache, WordPress stores transient records in the wp_options table. Typically these are given an expiration time and should disappear over time. However, that is not always the case. We have seen some databases where there are thousands of old transient records. In fact, on one site, we dealt with some corrupt transient records in which over **695,000 rows were generated** in the *wp_options* table. Yikes!



Showing rows 0 - 24 (695846 total, Query took 0.0145 seconds.)

```
SELECT * FROM [redacted]_options WHERE option_name LIKE '%wp_image_processing%'
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refr

1 > >> | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

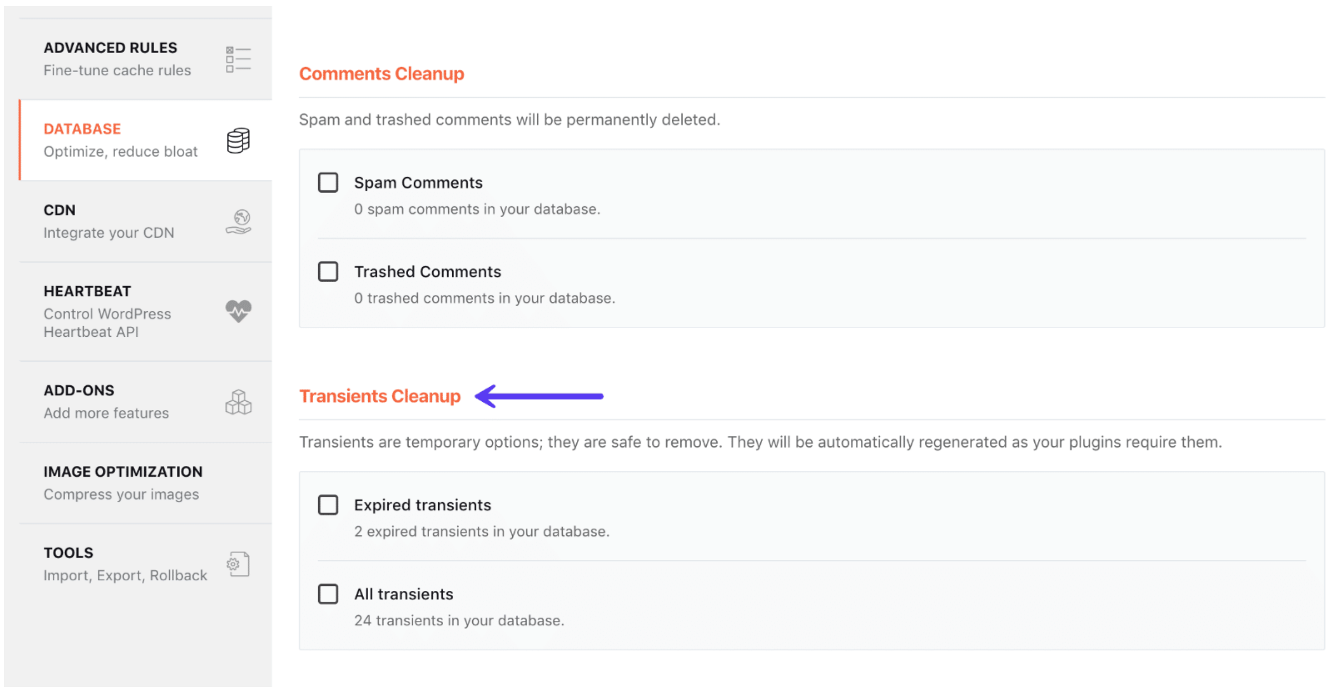
	option_id	option_name	option_value	autoload
Edit Copy Delete	524735	wp_image_processing_queue_batch_ca41bb3d7a31eb31c7...	a:9:{i:0;a:4:{s:7:"post_id";s:5:"15675";s:5:"width...	no
Edit Copy Delete	524736	wp_image_processing_queue_batch_f0c9c1a34e28df33f9...	a:10:{i:0;a:4:{s:7:"post_id";s:5:"15675";s:5:"widt...	no
Edit Copy Delete	524737	wp_image_processing_queue_batch_a204c4fcd7f48a844d...	a:1:{i:0;a:4:{s:7:"post_id";s:5:"22004";s:5:"width...	no
Edit Copy Delete	524738	wp_image_processing_queue_batch_76b5a2bb0f51d738f0...	a:2:{i:0;a:4:{s:7:"post_id";s:5:"22004";s:5:"width...	no
Edit Copy Delete	524739	wp_image_processing_queue_batch_7352d2a1b5ce4b9769...	a:3:{i:0;a:4:{s:7:"post_id";s:5:"22004";s:5:"width...	no

It's also important to note that transients are not auto-loaded by default. You could use a query like the below to see if there are any autoloaded transient data.

```
SELECT *
FROM `wp_options`
WHERE `autoload` = 'yes'
AND `option_name` LIKE '%transient%'
```

A better and safer option would be to utilize a free plugin like Transient Cleaner or Delete Expired Transients which can clean up only the expired transients from your *wp_options* table. However, it appears there is now a function in WordPress, added in 4.9, that cleans up expired transients. So hopefully that is happening automatically on your site now.

WP Rocket also has the ability to cleanup transients in their database optimization options.


ADVANCED RULES
Fine-tune cache rules

DATABASE
Optimize, reduce bloat

CDN
Integrate your CDN

HEARTBEAT
Control WordPress Heartbeat API

ADD-ONS
Add more features

IMAGE OPTIMIZATION
Compress your images

TOOLS
Import, Export, Rollback

Comments Cleanup

Spam and trashed comments will be permanently deleted.

☐ Spam Comments
0 spam comments in your database.

☐ Trashed Comments
0 trashed comments in your database.

Transients Cleanup

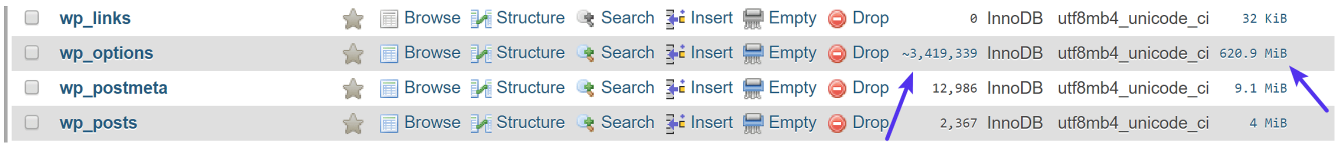
Transients are temporary options; they are safe to remove. They will be automatically regenerated as your plugins require them.

☐ Expired transients
2 expired transients in your database.

☐ All transients
24 transients in your database.

1:4.4 Clean up WordPress Sessions

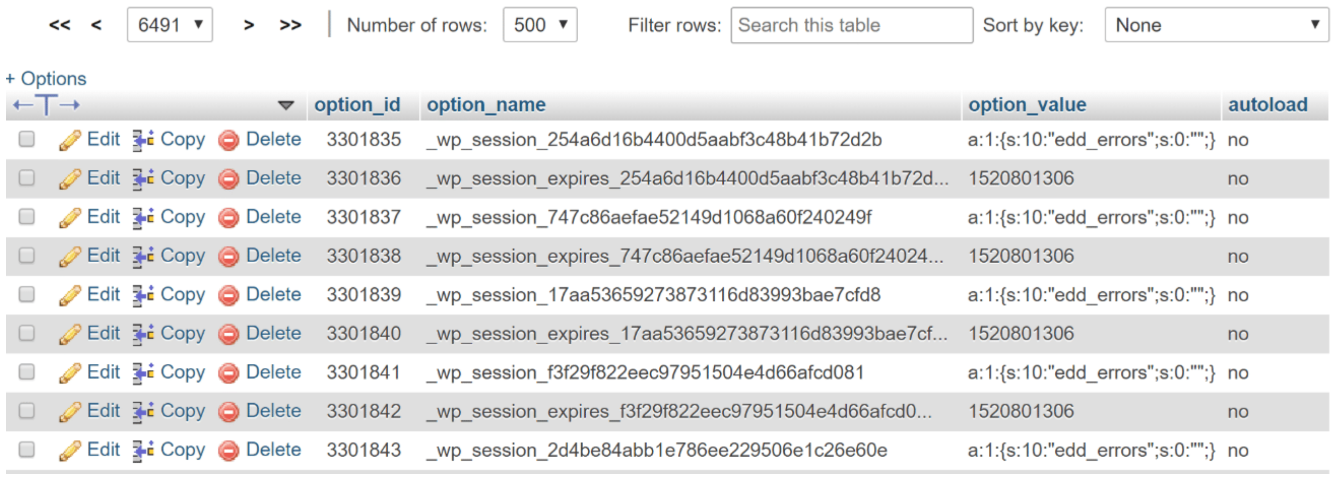
Another common issue we've seen is sometimes cron jobs get out of sync or don't fire properly, and therefore sessions don't get cleaned up. You can wind up getting tons of _*wp_session*_ rows in your database. In this example below the site in question wound up with over 3 million rows in their *wp_options* table. And the table had grown to over 600 MB in size.



☐	wp_links	Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	utf8mb4_unicode_ci	32 KiB
☐	wp_options	Browse	Structure	Search	Insert	Empty	Drop	~3,419,339	InnoDB	utf8mb4_unicode_ci	620.9 MiB
☐	wp_postmeta	Browse	Structure	Search	Insert	Empty	Drop	12,986	InnoDB	utf8mb4_unicode_ci	9.1 MiB
☐	wp_posts	Browse	Structure	Search	Insert	Empty	Drop	2,367	InnoDB	utf8mb4_unicode_ci	4 MiB

You could use a query like the one below to see if you're running into this issue:

```
SELECT *
FROM `wp_options`
WHERE `option_name` LIKE '_wp_session_%'
```



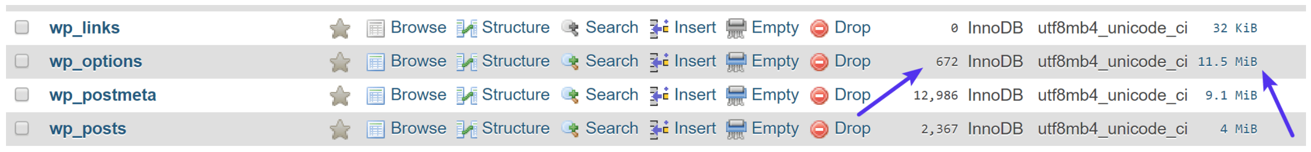
<< < 6491 > >> | Number of rows: 500 Filter rows: Search this table Sort by key: None
+ Options

	option_id	option_name	option_value	autoload
Edit Copy Delete	3301835	_wp_session_254a6d16b4400d5aabf3c48b41b72d2b	a:1:{s:10:"edd_errors";s:0:"";}	no
Edit Copy Delete	3301836	_wp_session_expires_254a6d16b4400d5aabf3c48b41b72d...	1520801306	no
Edit Copy Delete	3301837	_wp_session_747c86aefae52149d1068a60f240249f	a:1:{s:10:"edd_errors";s:0:"";}	no
Edit Copy Delete	3301838	_wp_session_expires_747c86aefae52149d1068a60f24024...	1520801306	no
Edit Copy Delete	3301839	_wp_session_17aa53659273873116d83993bae7cfd8	a:1:{s:10:"edd_errors";s:0:"";}	no
Edit Copy Delete	3301840	_wp_session_expires_17aa53659273873116d83993bae7cf...	1520801306	no
Edit Copy Delete	3301841	_wp_session_f3f29f822eec97951504e4d66afcd081	a:1:{s:10:"edd_errors";s:0:"";}	no
Edit Copy Delete	3301842	_wp_session_expires_f3f29f822eec97951504e4d66afcd0...	1520801306	no
Edit Copy Delete	3301843	_wp_session_2d4be84abb1e786ee229506e1c26e60e	a:1:{s:10:"edd_errors";s:0:"";}	no

In most cases you can then safely delete these (as a cron job should have) with the following command:

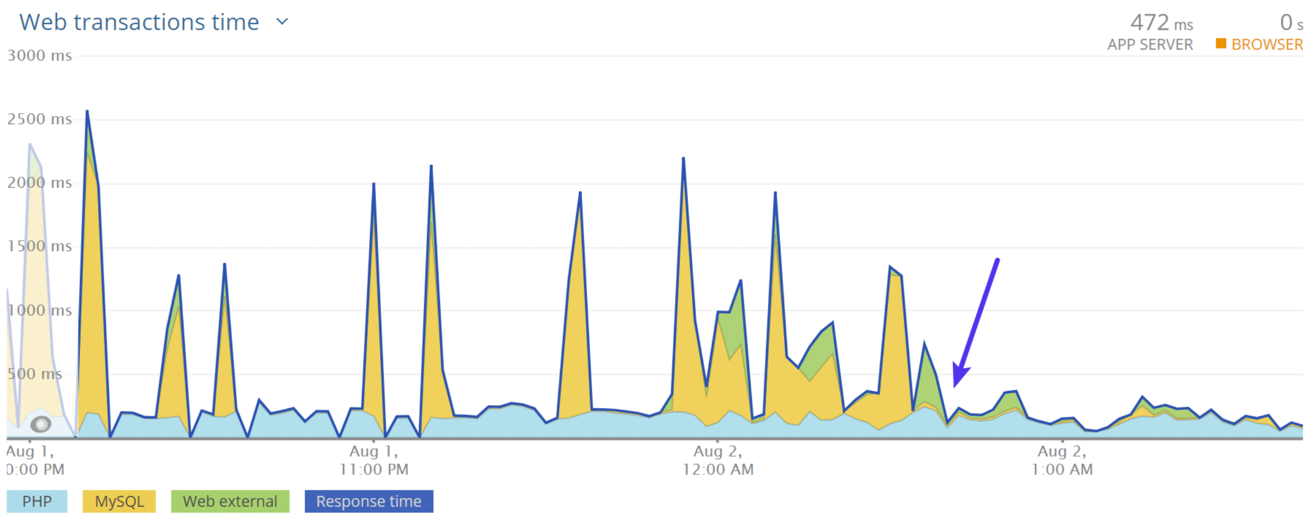
```
DELETE FROM `wp_options`
WHERE `option_name` LIKE '_wp_session_%'
```

After cleaning up all the leftover _*wp_session*_ rows the table had less than 1,000 rows and was reduced to 11 MB in size.



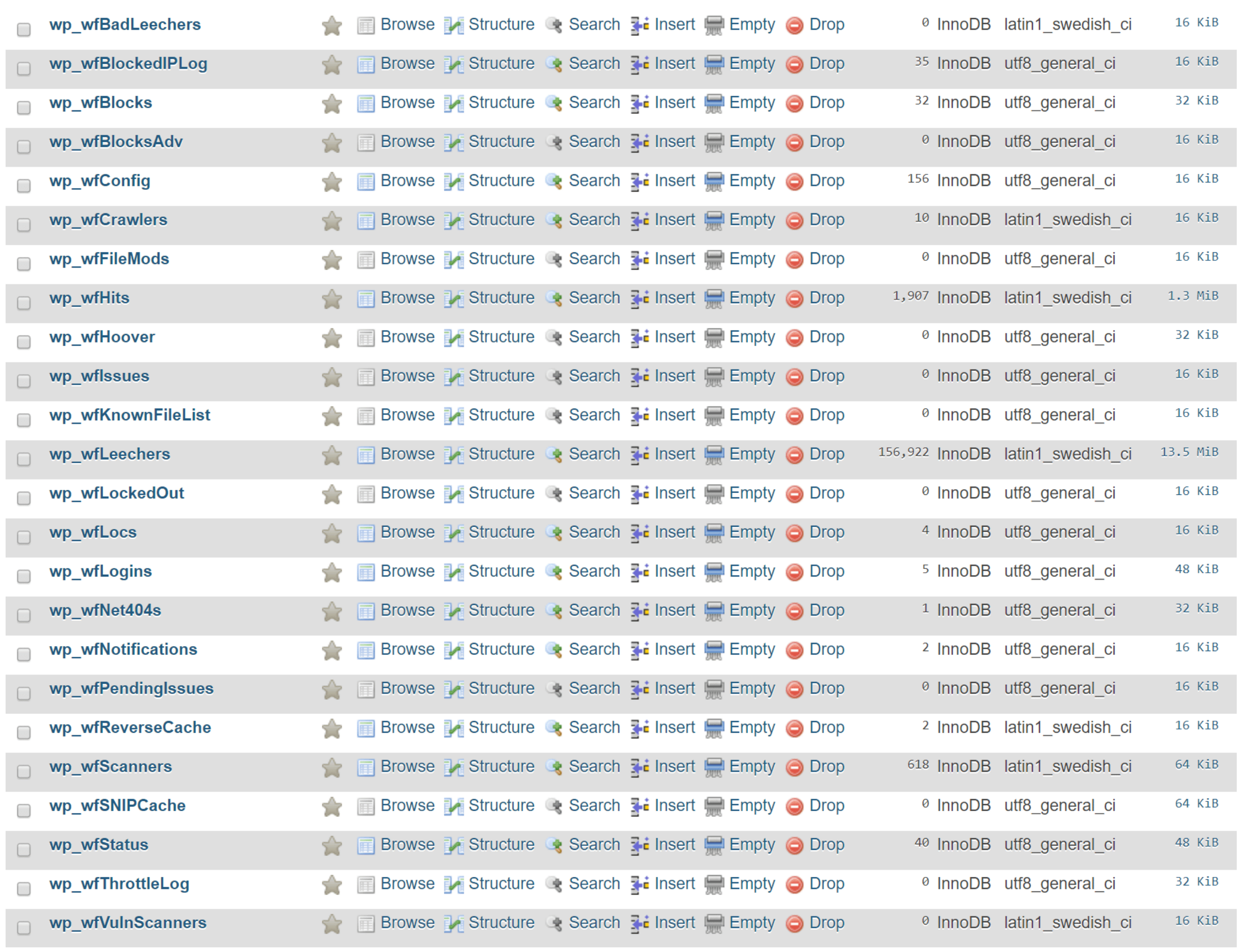
wp_links	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32 KiB
wp_options	Browse Structure Search Insert Empty Drop	672	InnoDB	utf8mb4_unicode_ci	11.5 MiB
wp_postmeta	Browse Structure Search Insert Empty Drop	12,986	InnoDB	utf8mb4_unicode_ci	9.1 MiB
wp_posts	Browse Structure Search Insert Empty Drop	2,367	InnoDB	utf8mb4_unicode_ci	4 MiB

It also fixed the spikes the site was getting in MySQL.



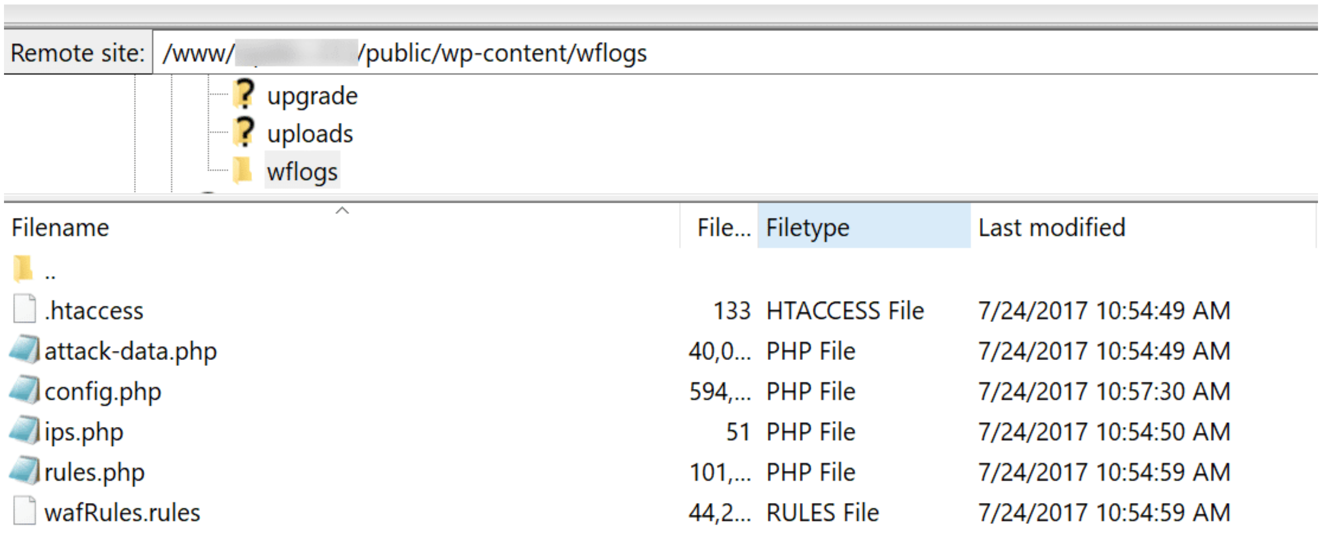
1:4.5 Clean Up Old WordPress Plugins Data

One big issue with WordPress plugins is the uninstall process. Whenever you install a WordPress plugin or theme, it stores the data in the database. The problem is that when you delete a plugin using one of the standard methods, it typically leaves behind tables and rows in your database. Over time this can add up to a lot of data and even begin to slow your site down. In our example, we uninstalled the Wordfence security plugin, and it left behind 24 tables in our database (as seen below). It's even worse if they're behind data in your *wp_options* table.



wp_wfBadLeechers	Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	latin1_swedish_ci	16 KiB
wp_wfBlockedIPLog	Browse	Structure	Search	Insert	Empty	Drop	35	InnoDB	utf8_general_ci	16 KiB
wp_wfBlocks	Browse	Structure	Search	Insert	Empty	Drop	32	InnoDB	utf8_general_ci	32 KiB
wp_wfBlocksAdv	Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	utf8_general_ci	16 KiB
wp_wfConfig	Browse	Structure	Search	Insert	Empty	Drop	156	InnoDB	utf8_general_ci	16 KiB
wp_wfCrawlers	Browse	Structure	Search	Insert	Empty	Drop	10	InnoDB	latin1_swedish_ci	16 KiB
wp_wfFileMods	Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	utf8_general_ci	16 KiB
wp_wfHits	Browse	Structure	Search	Insert	Empty	Drop	1,907	InnoDB	latin1_swedish_ci	1.3 MiB
wp_wfHoover	Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	utf8_general_ci	32 KiB
wp_wfIssues	Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	utf8_general_ci	16 KiB
wp_wfKnownFileList	Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	utf8_general_ci	16 KiB
wp_wfLeechers	Browse	Structure	Search	Insert	Empty	Drop	156,922	InnoDB	latin1_swedish_ci	13.5 MiB
wp_wfLockedOut	Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	utf8_general_ci	16 KiB
wp_wfLocs	Browse	Structure	Search	Insert	Empty	Drop	4	InnoDB	utf8_general_ci	16 KiB
wp_wfLogins	Browse	Structure	Search	Insert	Empty	Drop	5	InnoDB	utf8_general_ci	48 KiB
wp_wfNet404s	Browse	Structure	Search	Insert	Empty	Drop	1	InnoDB	utf8_general_ci	32 KiB
wp_wfNotifications	Browse	Structure	Search	Insert	Empty	Drop	2	InnoDB	utf8_general_ci	16 KiB
wp_wfPendingIssues	Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	utf8_general_ci	16 KiB
wp_wfReverseCache	Browse	Structure	Search	Insert	Empty	Drop	2	InnoDB	latin1_swedish_ci	16 KiB
wp_wfScanners	Browse	Structure	Search	Insert	Empty	Drop	618	InnoDB	latin1_swedish_ci	64 KiB
wp_wfSNIPCache	Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	utf8_general_ci	64 KiB
wp_wfStatus	Browse	Structure	Search	Insert	Empty	Drop	40	InnoDB	utf8_general_ci	48 KiB
wp_wfThrottleLog	Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	utf8_general_ci	32 KiB
wp_wfVulnScanners	Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	latin1_swedish_ci	16 KiB

And besides the database, a lot of plugins also leave behind additional folders and files. In our experience, this is commonly seen with security and caching plugins which create additional directories for logging. For example, after the Wordfence plugin was deleted, we were left with a “wflogs” folder in our wp-content directory. And we aren’t trying to pick on Wordfence, the majority of plugins and themes on the market work this way.


Remote site: /www/[redacted]/public/wp-content/wflogs

upgrade
uploads
wflogs

Filename	File...	Filetype	Last modified
..			
.htaccess	133	HTACCESS File	7/24/2017 10:54:49 AM
attack-data.php	40,0...	PHP File	7/24/2017 10:54:49 AM
config.php	594,...	PHP File	7/24/2017 10:57:30 AM
ips.php	51	PHP File	7/24/2017 10:54:50 AM
rules.php	101,...	PHP File	7/24/2017 10:54:59 AM
wafRules.rules	44,2...	RULES File	7/24/2017 10:54:59 AM

1:4.6 Add an Index to Autoload

If cleaning up your *wp_options* table wasn't enough, you could try adding an "index" to the autoload field. This essentially can help it to be searched more efficiently. The awesome team over at 10up performed some test scenarios on a *wp_options* table with a typical number of autoloaded records to show how adding an autoload index to *wp_options* queries can boost performance.

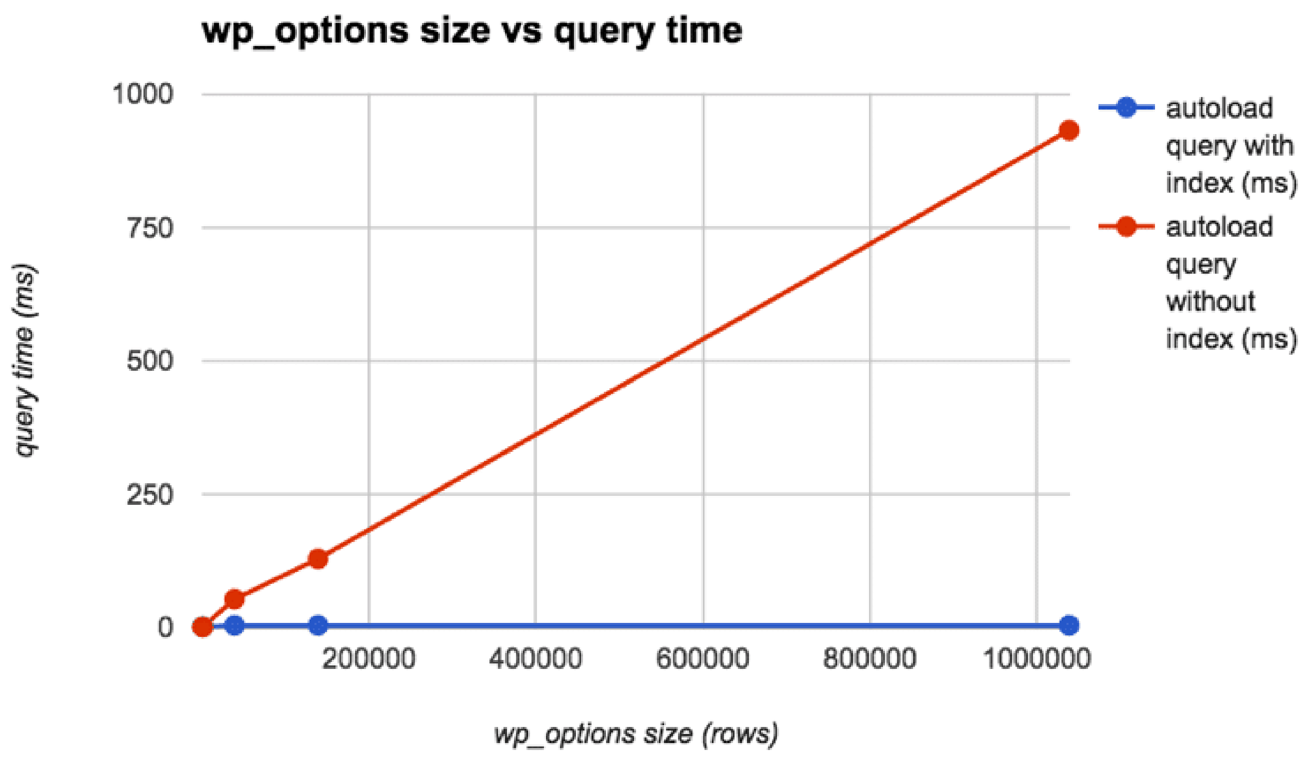


Image source: 10up

We also recommend checking out these two additional resources from WP Bullet:

- How to Add MySQL Index to wp_options table
- Cleaning up the wp_options table using WP-CLI

1:5 Increase PHP Workers

PHP workers might be a term you've never heard of, but they are how many hosts, including Kinsta, handle limiting requests (rather than limiting you by CPU or RAM, which is typically what shared hosting providers do).

PHP workers determine **how many simultaneous requests your site can handle at a given time**. To put it simply, each uncached request for your website is handled by a PHP Worker. For example, if you have 4 requests that come to your site at the exact same time and your site has 2 PHP workers, two of those requests will get processed while the other two will have to wait in the queue until the first two have finished processing.

Remember we discussed earlier that one of the biggest problems with WordPress membership sites is all of those uncached requests. This is why PHP workers become very important as they have to do work for each request. Therefore, these sites will typically require additional PHP workers to ensure every request is processed without delays and completed successfully.

What happens if you continuously max out your PHP workers? Basically, the queue starts to push out older requests which could result in 500 errors on your site.

- Learn more about PHP Workers

1:6 Enable Hotlink Protection

The concept of hotlinking is pretty straightforward. You find an image on the internet somewhere and use the URL of the image directly on your site. This image will be displayed on your website but it will be served from the original location. This is very convenient for the hotlinker, but it’s actually theft as it is using the hotlinked site’s resources. It’s like if we were to get in our car and drive away with gas we siphoned off from our neighbor’s car.

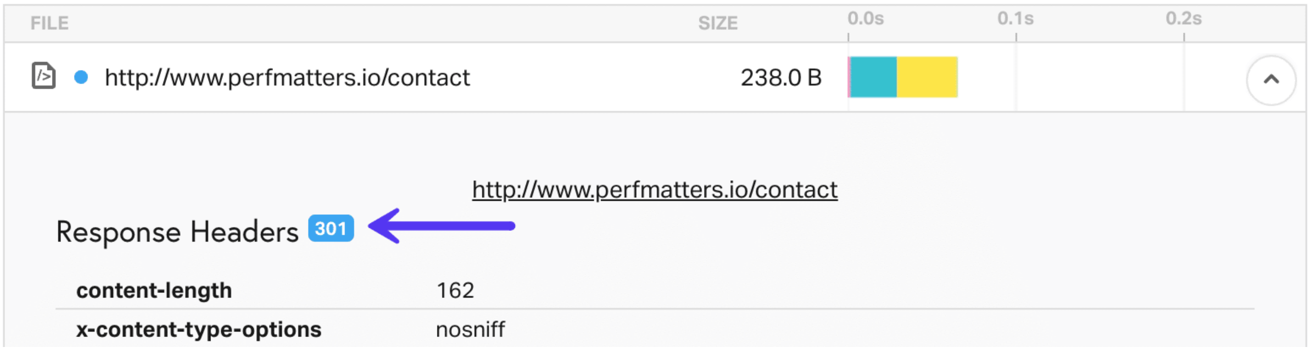
Hotlinking can be a **huge drain on resources for the target server**. Imagine if you are on a shared WordPress host and Huffington Post suddenly links to your images. You could go from a couple hundred queries an hour on your site to a couple hundred thousand. This could even result in a suspension of your hosting account. This is a reason to not only use a high-performance host (which can handle hiccups like this), but also to enable hotlink protection, so this doesn’t happen.

- Check out our tutorial on how to prevent hotlinking.

1:7 Minimize Redirects and Add Them at the Server-Level

Too many redirects are always something you need to watch out for. Simple redirects like a single 301 redirect, HTTP to HTTPS, or www to non-www (vice versa) are fine. And a lot of times these are needed in certain areas of your website. However, each has a cost on your site's performance. And if you start stacking redirects on top of each other, it's important to realize how they impact your site. This applies to page and post redirects, image redirects, everything.

A redirect will generate a 301 or 302 on the response header status.



FILE	SIZE	0.0s 0.1s 0.2s
http://www.perfmatters.io/contact	238.0 B	

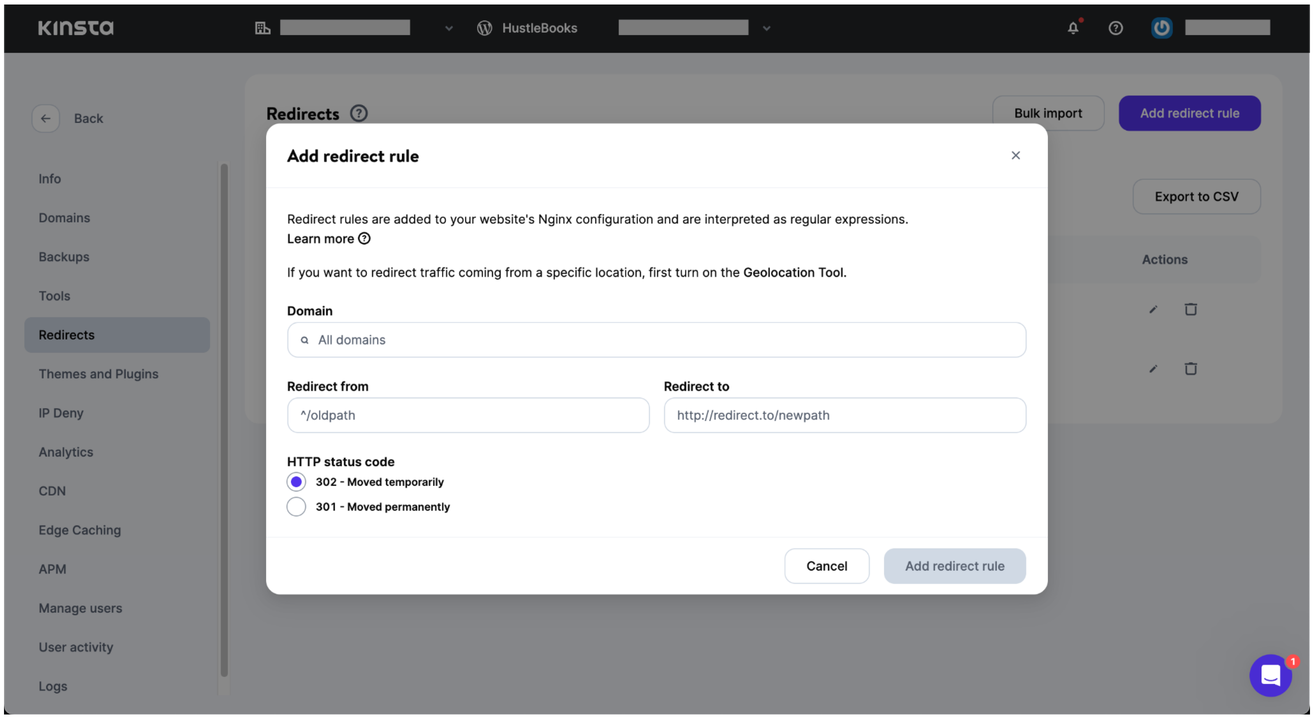
http://www.perfmatters.io/contact

Response Headers 301

content-length	162
x-content-type-options	nosniff

Using free WordPress plugins to implement redirects can sometimes cause performance issues as most of them utilize the wp_redirect function, which requires additional code execution and resources. Some of them also add autoloaded data to your *wp_options* table, which increases database bloat. Adding them at the server-level is where they should be done.

We allow you to do that at MyKinsta with our redirect rules tool.



Add redirect rule

Redirect rules are added to your website's Nginx configuration and are interpreted as regular expressions.
Learn more

If you want to redirect traffic coming from a specific location, first turn on the **Geolocation Tool.**

Domain: All domains

Redirect from: ^/oldpath

Redirect to: http://redirect.to/newpath

HTTP status code
[✓ 302 - Moved temporarily]
301 - Moved permanently

Cancel | Add redirect rule

You can also see a complete breakdown of how many redirects are happening on your sites in our MyKinsta analytics tool. See the total number of 301's, 302's, and 304's.

Check out our in-depth post on WordPress redirects, and the best practices for faster performance.

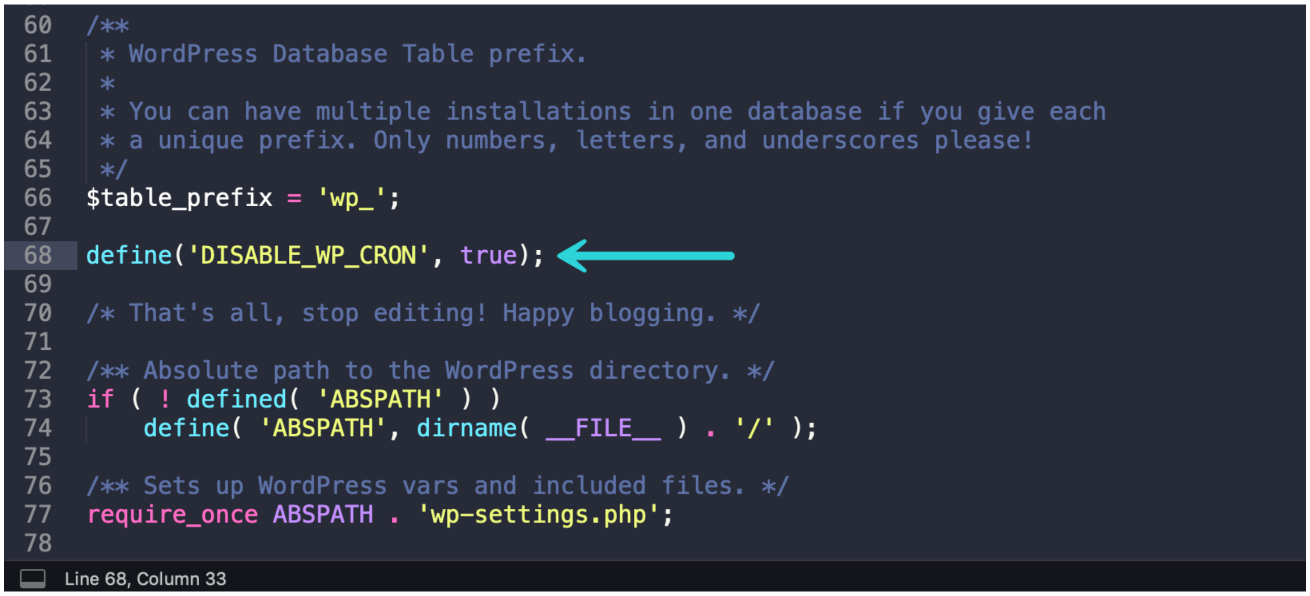
1:8 Manage and Minimize Cron Jobs

CRON jobs (WP-Cron) are used to schedule repetitive tasks for your WordPress site. However, over time, these can get out of control and cause performance issues. You can use the free WP Control plugin to check a handle on all the Cron jobs happening on your site.

We have also seen performance issues with the WordPress built-in Cron handler: WP-Cron. If a site doesn't have enough PHP workers, sometimes a request will come in, WordPress will spawn the cron, but the cron has to wait for the worker, and therefore just sits there. A better approach is to disable WP-Cron and use the system cron instead. This is even recommended in the official Plugin handbook.

To disable WP-Cron, add the following to your *wp-config.php* file, just before the line that says “That’s all, step editing! Happy blogging.” Note: This disables it from running on page load, not when you call it directly via *wp-cron.php*.

define('DISABLE_WP_CRON', true);



```
/**
 * WordPress Database Table prefix.
 *
 * You can have multiple installations in one database if you give each
 * a unique prefix. Only numbers, letters, and underscores please!
 */
$table_prefix = 'wp_';

define('DISABLE_WP_CRON', true);

/* That's all, stop editing! Happy blogging. */

/** Absolute path to the WordPress directory. */
if ( ! defined( 'ABSPATH' ) )
    define( 'ABSPATH', dirname( __FILE__ ) . '/' );

/** Sets up WordPress vars and included files. */
require_once ABSPATH . 'wp-settings.php';

```

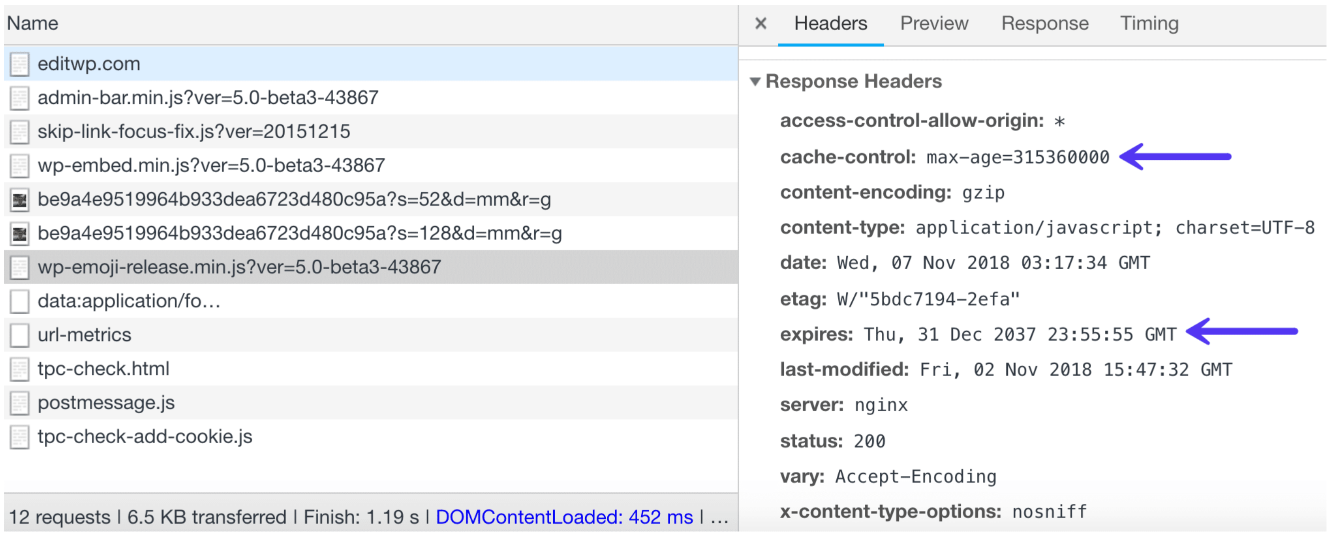
Line 68, Column 33

1:9 Add Cache-Control and Expires Headers (Determine Cache Length)

Every script on your WordPress site needs to have an HTTP cache header attached to it (or it should). This **determines when the cache on the file expires**. To fix this, ensure your WordPress host has the proper *cache-control* headers and *expires* headers setup. If you don’t, you will most likely see warnings about needing to add Expires Headers or leverage browser caching in speed testing tools.

While the *cache-control* header turns on client-side caching and sets the max-age of a resource, the *expires* header is used to specify a specific point in time the resource is no longer valid. While both the headers can be used together, you don't necessarily need to add both of the headers. cache-control is newer and usually the recommended method.

Kinsta automatically adds HTTP cache headers on all server requests, and if you're using a CDN, they will most likely add these headers for you as well.



Name
editwp.com
admin-bar.min.js?ver=5.0-beta3-43867
skip-link-focus-fix.js?ver=20151215
wp-embed.min.js?ver=5.0-beta3-43867
be9a4e9519964b933dea6723d480c95a?s=52&d=mm&r=g
be9a4e9519964b933dea6723d480c95a?s=128&d=mm&r=g
wp-emoji-release.min.js?ver=5.0-beta3-43867
data:application/fo…
url-metrics
tpc-check.html
postmessage.js
tpc-check-add-cookie.js

12 requests | 6.5 KB transferred | Finish: 1.19 s | DOMContentLoaded: 452 ms | …

▾ Response Headers

```
access-control-allow-origin: *
cache-control: max-age=315360000
content-encoding: gzip
content-type: application/javascript; charset=UTF-8
date: Wed, 07 Nov 2018 03:17:34 GMT
etag: W/"5bdc7194-2efa"
expires: Thu, 31 Dec 2037 23:55:55 GMT
last-modified: Fri, 02 Nov 2018 15:47:32 GMT
server: nginx
status: 200
vary: Accept-Encoding
x-content-type-options: nosniff
```

If your server is missing these headers, you can manually add them.

Adding Cache-Control Header in Nginx

You can add *cache-control* headers in Nginx by adding the following to your server config's server location or block.

```
location ~* \.(js|css|png|jpg|jpeg|gif|svg|ico)$ {
 expires 30d;
 add_header Cache-Control "public, no-transform";
}
```

Adding Expires Header in Nginx

You can add *expires* headers in Nginx by adding the following to your server block. In this example, you can see how to specify different expire times based on file types.

```
   location ~*  \.(jpg|jpeg|gif|png|svg)$ {
       expires 365d;
   }

   location ~*  \.(pdf|css|html|js|swf)$ {
       expires 2d;
   }
```

Adding Cache-Control Header in Apache

You can add *cache-control* headers in Apache by adding the following to your *.htaccess* file. Snippets of code can be added at the top or bottom of the file (before # BEGIN WordPress or after # END WordPress).

```
<filesMatch ".(ico|pdf|flv|jpg|jpeg|png|gif|svg|js|css|swf)$">
Header set Cache-Control "max-age=84600, public"
</filesMatch>
```

Adding Expires Header in Apache

You can add expires headers in Apache by adding the following to your .htaccess file.

```
## EXPIRES HEADER CACHING ##
<IfModule mod_expires.c>
ExpiresActive On
ExpiresByType image/jpg "access 1 year"
ExpiresByType image/jpeg "access 1 year"
ExpiresByType image/gif "access 1 year"
ExpiresByType image/png "access 1 year"
ExpiresByType image/svg "access 1 year"
ExpiresByType text/css "access 1 month"
ExpiresByType application/pdf "access 1 month"
ExpiresByType application/javascript "access 1 month"
ExpiresByType application/x-javascript "access 1 month"
ExpiresByType application/x-shockwave-flash "access 1 month"
ExpiresByType image/x-icon "access 1 year"
ExpiresDefault "access 2 days"
```

</IfModule>
EXPIRES HEADER CACHING

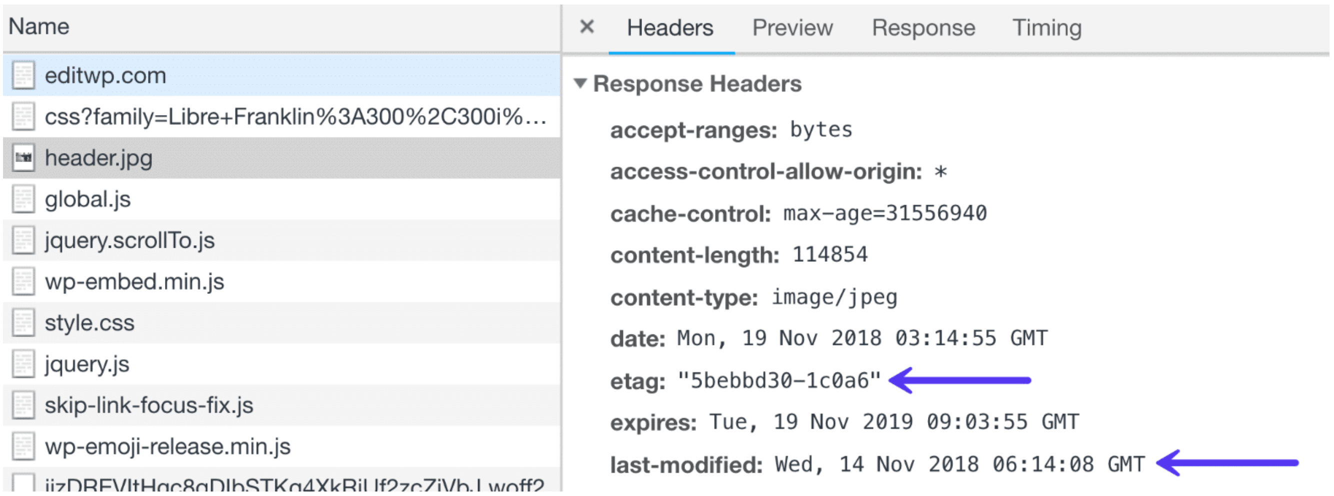
It's also important to note that **you can only add HTTP cache headers on resources on *your* server**. If you're getting a warning about that perhaps you need to leverage browser caching on a third-party request, there is nothing you can do, as you don't have control over their server. Common culprits include the Google Analytics script and marketing pixels, like Facebook and Twitter.

If you're trying to fix this with the Google Analytics script, you can host it locally or on your CDN (although this isn't officially supported) with a plugin like Perfmatters or WP Rocket.

1:10 Add Last-Modified and ETag Headers (Validate Cache)

Next, we have another two sets of headers, *last-modified* and *etag*.

While the *cache-control* and *expires* headers help the browser determine **if the file has changed** since the last time it was requested (or rather they validate the cache). The *last-modified* and *etag* headers both **validate and set the length of the cache** and should be included on every origin server response. If these aren't properly set you might see a warning that you need to "Specify a cache validator."



```
accept-ranges: bytes
access-control-allow-origin: *
cache-control: max-age=31556940
content-length: 114854
content-type: image/jpeg
date: Mon, 19 Nov 2018 03:14:55 GMT
etag: "5bebbd30-1c0a6"
expires: Tue, 19 Nov 2019 09:03:55 GMT
last-modified: Wed, 14 Nov 2018 06:14:08 GMT
```

If the headers aren't found, it will generate a new request for the resource every time, which increases the load on your server. Utilizing caching headers ensures that subsequent requests don't have to be loaded from the server, thus saving bandwidth and improving performance for the user.

Kinsta automatically adds the above headers on all server requests, and if you're using a CDN, they will most likely add these headers for you as well. Just like with *cache-control* and *expires*, you can't manually set these HTTP headers on external resources.

Last-Modified Header

The **last-modified** header is generally sent automatically from the server. This is one header you **generally won't need to add manually**. It is sent to see if the file in the browser's cache has been modified since the last time it was requested. You can look at the header request in Pingdom or use Chrome DevTools to see the value of the last-modified header.

ETag Header

The **ETag** header is also very similar to the last-modified header. It is also used to validate the cache of a file. If you're running Apache 2.4 or higher, the ETag header is already automatically added using the FileETag directive. And as far as NGINX goes, the ETag header has been enabled by default since 2016.

You can enable the ETag header manually in Nginx using the following code.

```
etag on
```

1:11 Changing the WordPress Memory Limit in wp-config.php

As stated in the WordPress DevHub, with WordPress Version 2.5, the *WP_MEMORY_LIMIT* option allows you to specify the maximum amount of memory that can be consumed by PHP. This setting may be necessary in the event you receive a message such as “Allowed memory size of xxxxxx bytes exhausted”.

By default, WordPress will attempt to increase the memory allocated to PHP to 40MB for a single site and 64MB for multisite. They define the memory limits in the file *./wp-includes/default-constants.php*, on lines 32 – 44 (source).

You then also have PHP *memory_limit* on the server by your hosting provider. These are two different things. At Kinsta we set the default *memory_limit* to 256M. If you’re running into the memory size exhausted error you can try increasing the PHP memory limit in WordPress.

Add the following to your wp-config.php file, just before the line that says “That’s all, stop editing! Happy blogging.”

```
define( 'WP_MEMORY_LIMIT', '256M' );
```

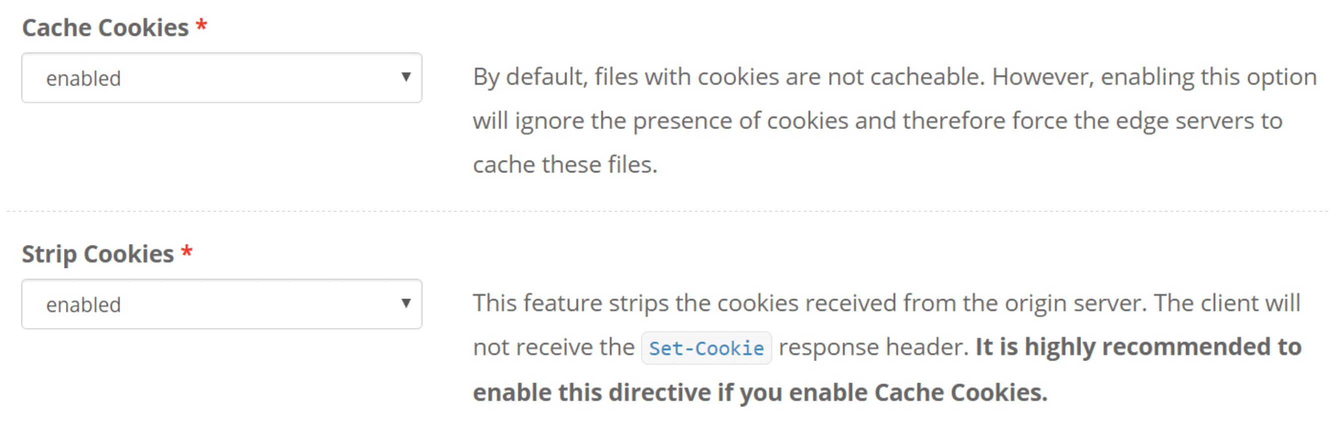
Jan Reilink also has a great blog post which describes the WordPress memory limit issue in more detail. He also gives a variation on the code you could use. Instead of setting the amount manually, you can set it to the PHP *memory_limit* value.

```
define( 'WP_MEMORY_LIMIT', ini_get( 'memory_limit' ) );
```

1:12 Use Cookie-Free Domains

Generally, when you are serving content such as images, JavaScript, CSS, there is no reason for an HTTP cookie to accompany it, as it creates additional overhead. Once the server sets a cookie for a particular domain, all subsequent HTTP requests for that domain must include the cookie. This warning is typically seen on sites with a large number of requests.

We have an in-depth post on how to deal with the ”Serve static content from a cookieless domain” warning.


Cache Cookies *

enabled

By default, files with cookies are not cacheable. However, enabling this option will ignore the presence of cookies and therefore force the edge servers to cache these files.

Strip Cookies *

enabled

This feature strips the cookies received from the origin server. The client will not receive the `Set-Cookie` response header. **It is highly recommended to enable this directive if you enable Cache Cookies.**

If you're running Cloudflare, you can't disable cookies on resources served through their network. CloudFlare includes their own security cookie in your header. Again these cookies are very small and the performance implications are extremely minimal. But if you use CloudFlare, there is no way to get around this warning.

A second way to get around this is to re-configure your WordPress site to deliver the static assets from a new domain or subdomain.

1:13 Use Prefetch and Preconnect

Resource hints and directives such as *prefetch* and *preconnect* can be a great way to speed up WordPress behind the scenes. KeyCDN has an excellent article and overview of resource hints.

Prefetch

DNS prefetch allows you to resolve domain names (perform a DNS lookup in the background) before a user clicks on a link, which in turn can help improve performance. It's done by adding a *rel="dns-prefetch"* tag in the header of your WordPress site.

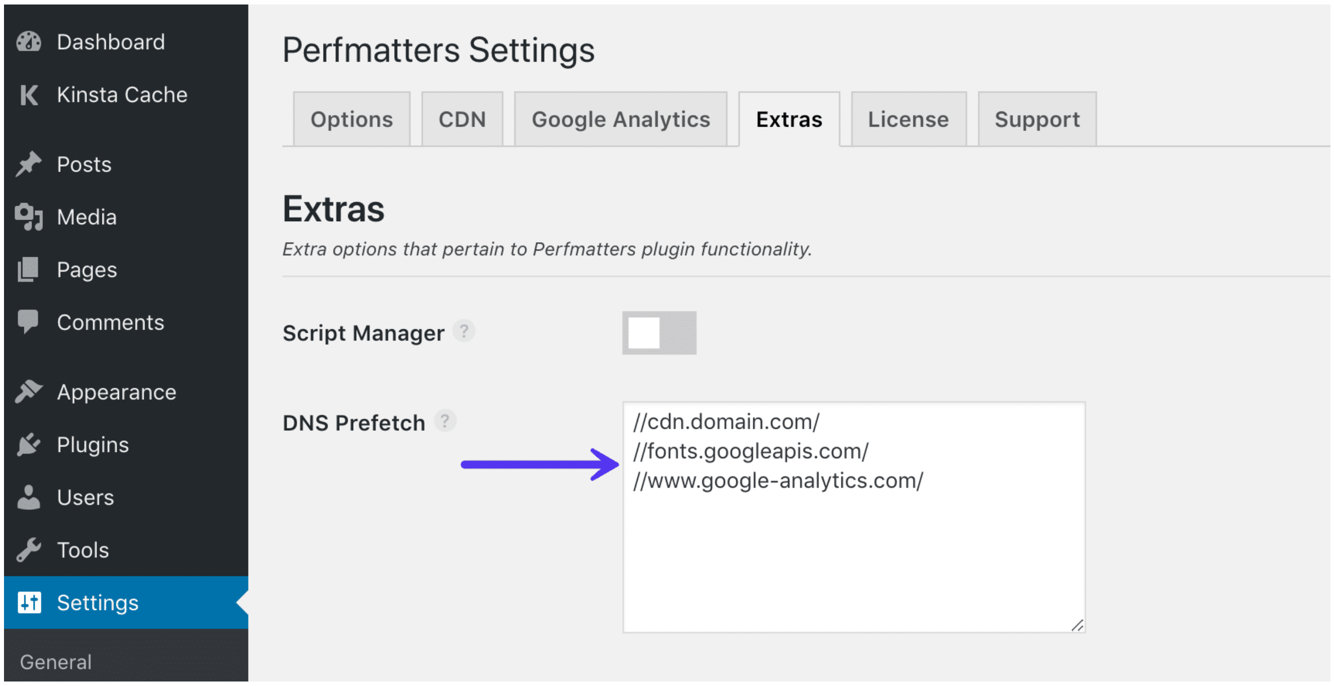
```
<link rel="dns-prefetch" href="//domain.com">
```

Some common things to use DNS prefetching for is your CDN URL, Google fonts, Google Analytics, etc.

```
<link rel="dns-prefetch" href="//cdn.domain.com/">
<link rel="dns-prefetch" href="//fonts.googleapis.com/">
<link rel="dns-prefetch" href="//www.google-analytics.com">
```

Prefetch is also supported by most modern browsers. Check out our tutorial on how to add code to your WordPress header.

Or you can easily implement DNS prefetch using a plugin like Perfmatters. Simply click on the "Extras" tab in the Perfmatters plugin and add domains. Format: *//domain.tld* (one per line)



Perfmatters Settings

Extras

Extra options that pertain to Perfmatters plugin functionality.

Script Manager

DNS Prefetch

```
//cdn.domain.com/
//fonts.googleapis.com/
//www.google-analytics.com/
```

Preconnect

Preconnect allows the browser to set up early connections before an HTTP request, eliminating round-trip latency and saving time for users.

> ***Preconnect is an important tool in your optimization toolbox… it can eliminate many costly roundtrips from your request path – in some cases reducing the request latency by hundreds and even thousands of milliseconds. – Iya Grigorik (source)***

It’s done by adding a *rel=”preconnect”* tag in the header of your WordPress site.

```
<link rel="preconnect" href="//domain.com">
```

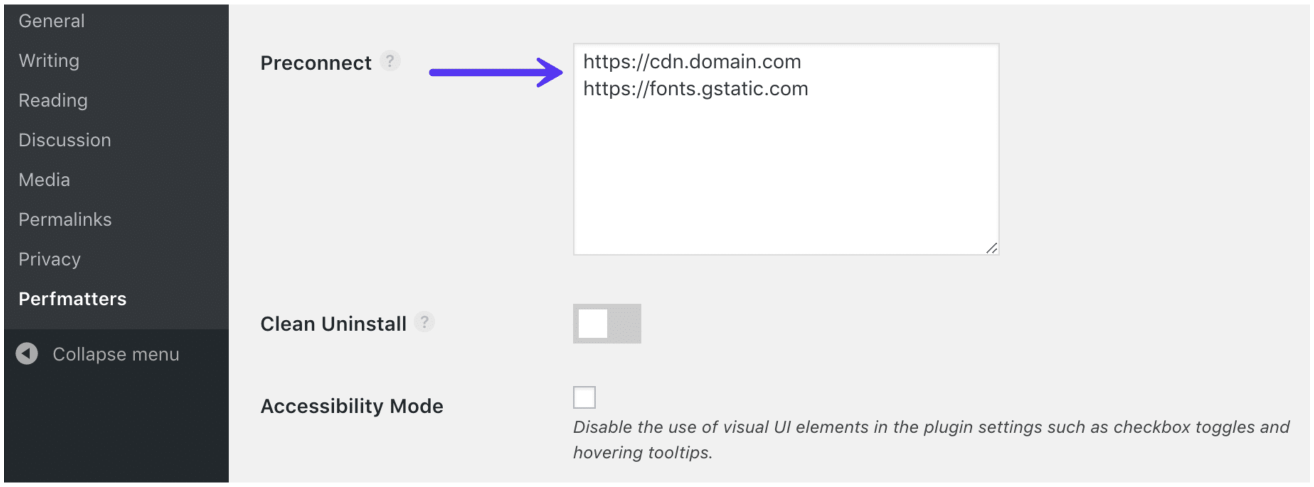
A few examples of things you might want to utilize this for include your CDN URL or Google Fonts.

```
<link rel="preconnect" href="https://cdn.domain.com">
<link rel="preconnect" href="https://fonts.gstatic.com">
```

Preconnect is supported by most modern browsers, with the exception of Internet Explorer, Safari, IOS Safari, and Opera Mini. There are a couple ways to implement this.

Preconnect is supported by most modern browsers, with the exception of Internet Explorer, Safari, IOS Safari, and Opera Mini. Check out our tutorial on how to add code to your WordPress header.

Or you can easily implement preconnect using a plugin like Perfmatters. Simply click on the “Extras” tab in the Perfmatters plugin and add domains. Format: *scheme://domain.tld* (one per line).



Preconnect

```
https://cdn.domain.com
https://fonts.gstatic.com
```

Clean Uninstall

Accessibility Mode

Disable the use of visual UI elements in the plugin settings such as checkbox toggles and hovering tooltips.

2: WordPress Settings and Optimizations

Now to move on to optimal WordPress settings. Here are actions you can take to help speed up your WordPress site. Many of these are very subtle changes, but everything really adds (or subtracts) up!

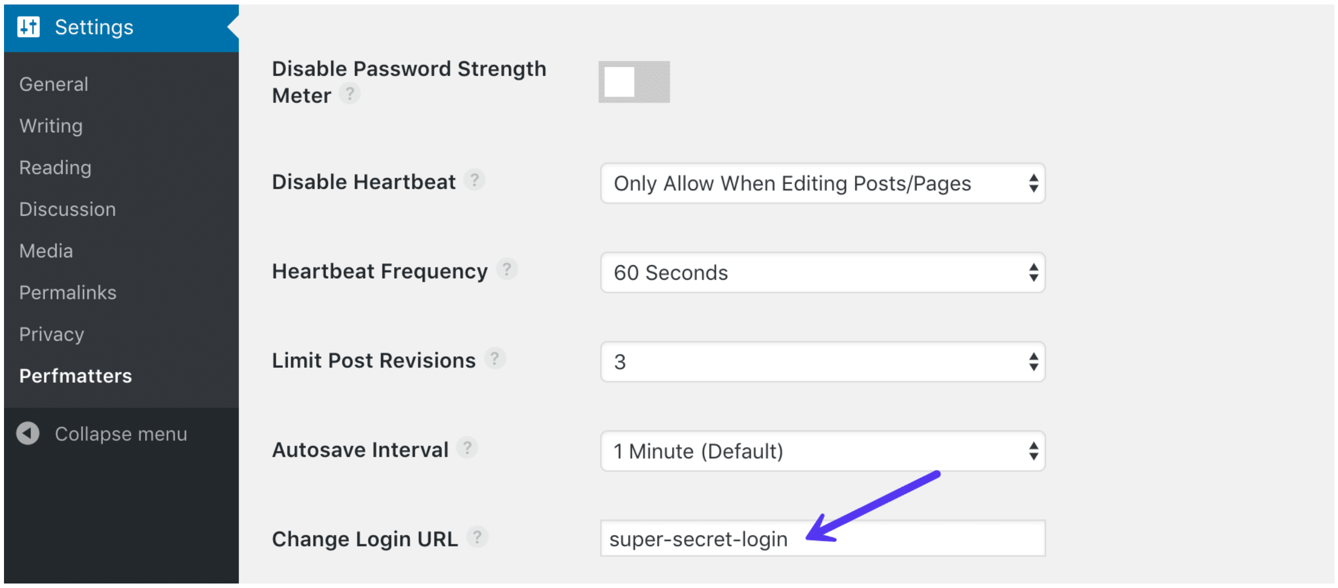
2:1 Change Your WordPress Login URL

By default your WordPress site's login URL is domain.com/wp-admin/. One of the problems with this is that all of the bots, hackers, and scripts out there also know this. By changing the URL, you can make yourself less of a target, better protect yourself against brute force attacks, and decrease the bandwidth used by the bots that hit this URL repeatedly.

Changing your WordPress login URL can also help prevent common errors like "429 Too Many Requests." This is not a fix all solution, it's merely one little trick that can help protect you and decrease the load on that page.

To change your WordPress login URL we recommend using one of the following plugins:

- WPS Hide Login (free)
- Perfmatters (premium, but includes other performance optimization settings)



Disable Password Strength Meter

Disable Heartbeat: Only Allow When Editing Posts/Pages

Heartbeat Frequency: 60 Seconds

Limit Post Revisions: 3

Autosave Interval: 1 Minute (Default)

Change Login URL: super-secret-login

2:2 Create a Light 404 Page

We've seen first-hand that highly dynamic sites typically generate a lot of 404 errors. Your website might be generating more than you think!

The reason these errors are bad is that many 404 pages are **very resource intensive**. For a highly dynamic WordPress site, you'll want to avoid a heavy 404 page. Create a simple 404 template that avoids querying the database any further if possible. And of course, spend some time and fix the 404 errors as this is not only resource intensive, it's simply bad for the user experience.

2:3 Understanding Your Theme's Impact and Sample Speed Tests

Every theme mentioned below is fully compatible with WooCommerce and Easy Digital Downloads, WPML, BuddyPress, and bbPress. We run a few speed tests with each theme using the following configuration:

- Hosted on Kinsta, running WordPress 5.0
- SSL (HTTPS)
- Kinsta CDN
- Imagify was used to automatically compress images.

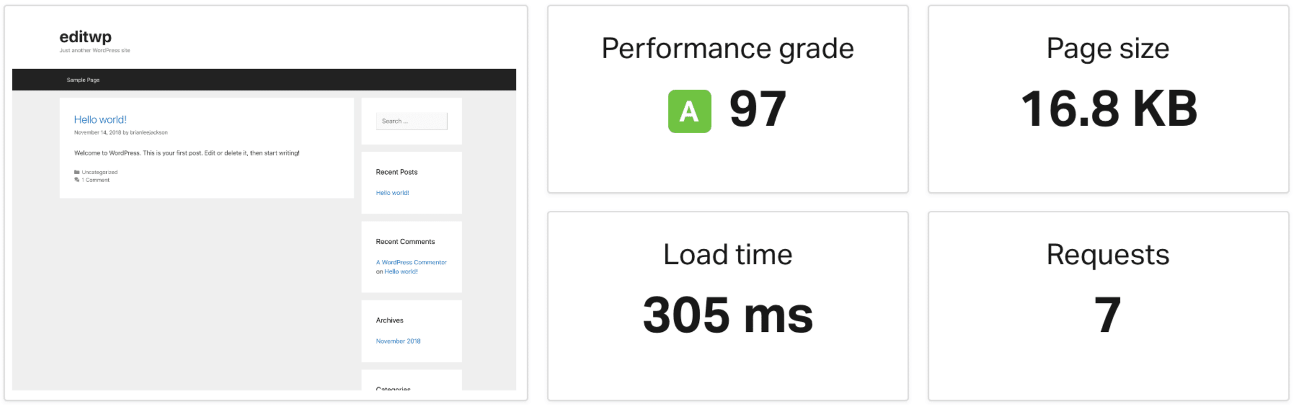
GeneratePress

GeneratePress is a fast, lightweight, mobile responsive WordPress theme built with speed, SEO and usability in mind. Built by Tom Usborne, a developer from Canada. It is actively updated and well supported. Even a few Kinsta team members use GeneratePress for their projects.

There is both a free and premium version available. If you take a look at the WordPress repository, the free version currently has over 600,000 active installs and an impressive 5 out of 5-star rating (over 850 people have given it 5 stars).

One of the great things about GeneratePress is that all the options use the native WordPress Customizer, meaning you can see every change you make instantly before pressing the publish button. This also means you don't have to learn a new theme control panel.

Just how fast is it? We did a fresh install of GeneratePress, ran five speed tests in Pingdom, and took the average. The total load time was **305 ms** with a total **page size of only 16.8 KB**. It's always good to have a baseline test to see what the theme is capable of in terms of raw performance.


editwp
Just another WordPress site
Sample Page
Hello world!
November 14, 2018 by brianleejackson
Welcome to WordPress. This is your first post. Edit or delete it, then start writing!
Uncategorized
1 Comment
Search ...
Recent Posts
Hello world!
Recent Comments
A WordPress Commenter on Hello world!
Archives
November 2018
Categories

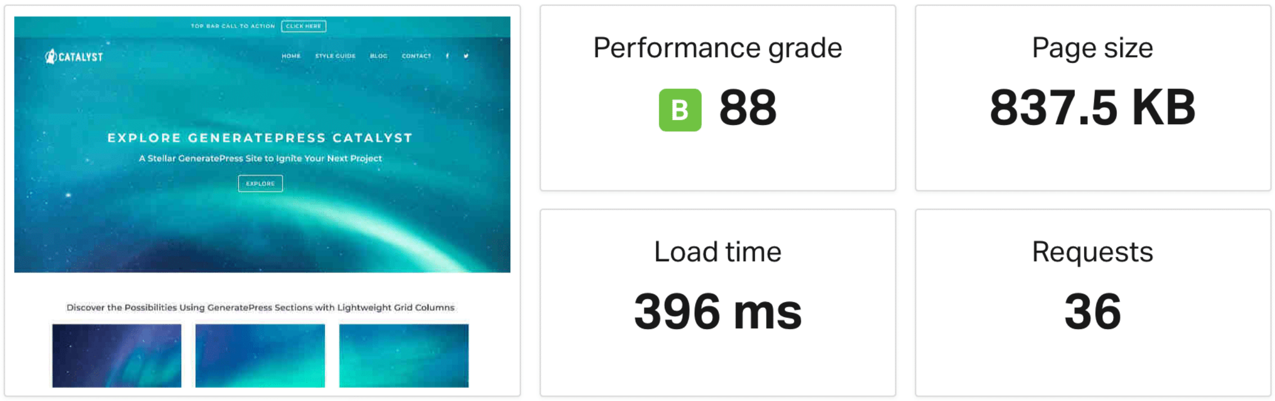
Performance grade
A 97

Page size
16.8 KB

Load time
305 ms

Requests
7

We then ran another set of tests with one of the pre-built themes from the GeneratePress site library. This contains images, backgrounds, new sections, etc. One advantage GeneratePress has is that it has a lot of pre-built themes that don't require a page builder plugin. You can see that it's still clocked under 400 ms.


TOP BAR CALL TO ACTION
CLICK HERE
CATALYST
HOME
STYLE GUIDE
BLOG
CONTACT
EXPLORE GENERATEPRESS CATALYST
A Stellar GeneratePress Site to Ignite Your Next Project
EXPLORE
Discover the Possibilities Using GeneratePress Sections with Lightweight Grid Columns

Performance grade
B 88

Page size
837.5 KB

Load time
396 ms

Requests
36

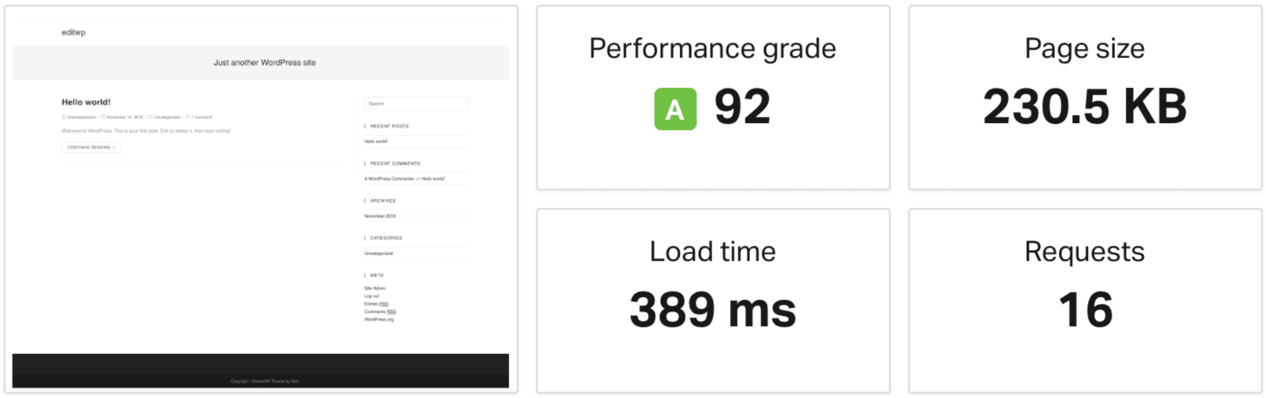
Now of course, in a real-world environment you might have other things running such as Google Analytics, Facebook remarketing pixel, Hotjar, etc. But you should be able to aim for under the 1-second mark easily.

OceanWP

The OceanWP theme is lightweight and highly extendable. It enables you to create almost any type of website, such as a blog, portfolio, business website or WooCommerce storefront with a beautiful & professional design. Built by Nicolas Lecocq, it is also actively updated and well supported.

Just like with GeneratePress, there is both a free and premium version available. If you take a look at the WordPress repository, the free version currently has over 700,000 active installs, and another impressive 5 out of 5-star rating.

Just how fast is it? We did a fresh install of OceanWP, ran five speed tests in Pingdom, and took the average. The total load time was **389 ms** with a total **page size of only 230.8 KB**. The scripts in OceanWP are slightly larger, but nothing to write home about.



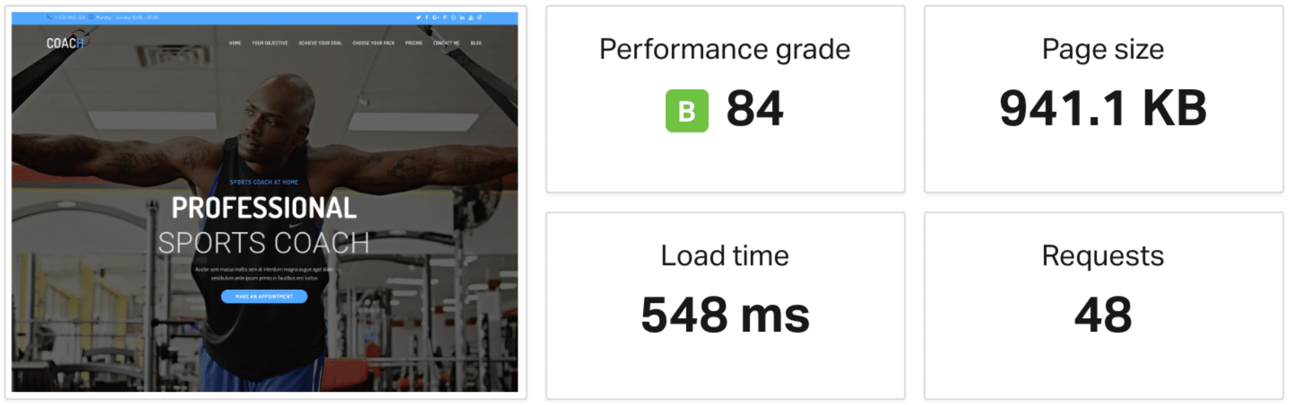
Performance grade A 92

Page size 230.5 KB

Load time 389 ms

Requests 16

We then ran another set of tests with one of the demo themes from the OceanWP site library. This contains images, backgrounds, new sections, and requires the Elementor page builder plugin. You can see that it's still clocked under 600 ms.



Performance grade
B 84

Page size
941.1 KB

Load time
548 ms

Requests
48

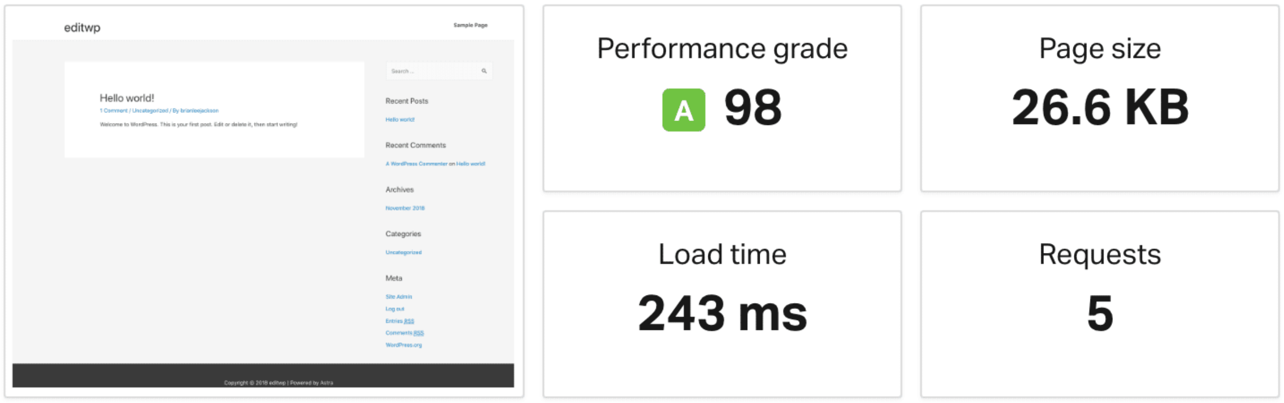
You can check out a more in-depth review of OceanWP on our blog.

Astra

Astra is a fast, fully customizable & beautiful theme suitable for blogs, personal portfolios, business websites, and WooCommerce storefronts. It is very lightweight (less than 50 KB on frontend) and offers unparalleled speed. Built by the team at Brainstorm Force, it is actively updated and well supported. You might recognize them as the creators of the popular All In One Schema Rich Snippets plugin which has been around for many years.

Just like with GeneratePress and OceanWP, there is both a free and premium version available. If you take a look at the WordPress repository, the free version currently has over 1 million active installs and 5-star rating.

Just how fast is it? We did a fresh install of Astra, ran five speed tests in Pingdom, and took the average. The total load time was **243 ms** with a total **page size of only 26.6 KB**.



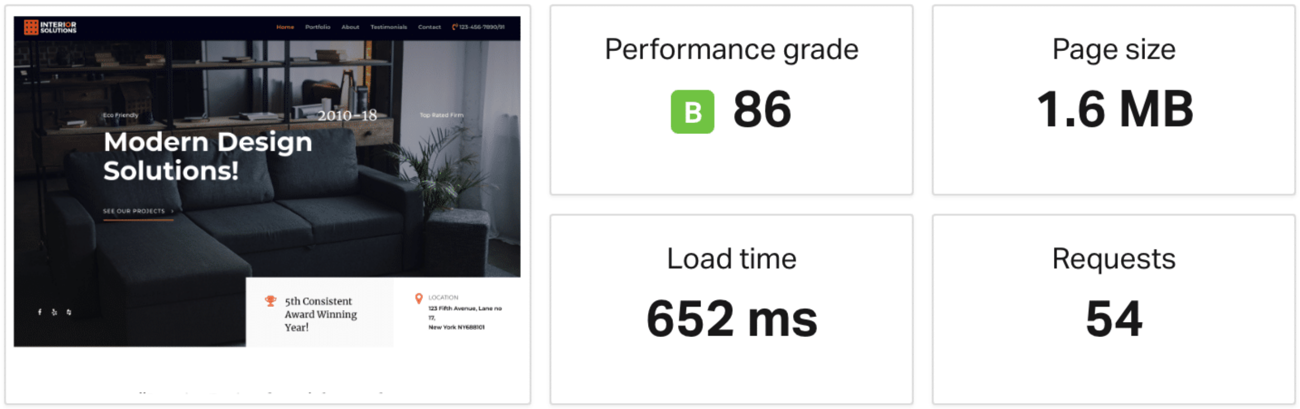
Performance grade A 98

Page size 26.6 KB

Load time 243 ms

Requests 5

We then ran another set of tests with one of the demo themes from the Astra Starter kit site library. This contains images, backgrounds, new sections, and requires the Elementor page builder plugin. You can see that it's still clocked under 700 ms. Note: the images in this demo were fully compressed, but they chose very high-resolution ones from the start.



Performance grade B 86

Page size 1.6 MB

Load time 652 ms

Requests 54

It's important to take the differences between the speed tests with these three themes with a grain of salt. The problem is that it's almost impossible to run a completely accurate side by side comparison. The important thing we wanted to show you is that all of these WordPress themes are blazing fast, both out of the box and full demos!

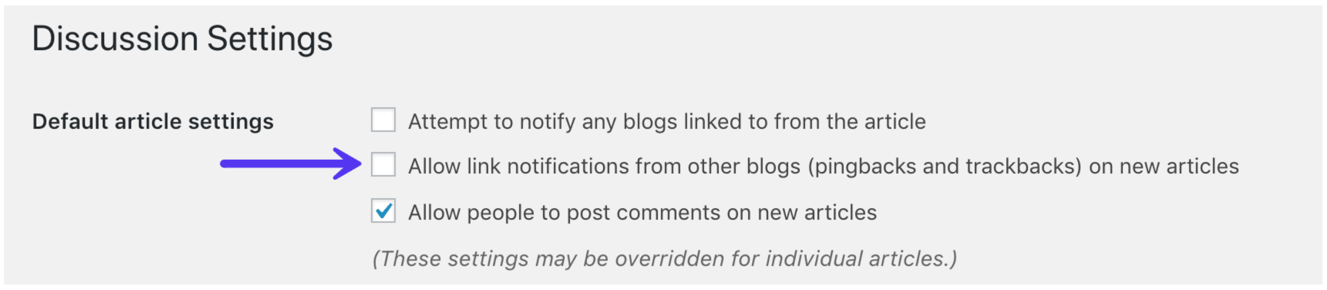
2:5 Disable Pingbacks

A pingback is an automated comment that gets created when another blog links to you. There can also be self-pingbacks which are created when you link to an article within your own blog.

We recommend simply disabling these as they generate worthless queries and additional spam on your site. Remember, the less calls your WordPress site has to make the better, especially on high-traffic sites. Not to mention the fact that a pingback on your own website is just downright annoying. Follow the steps below to disable pingbacks.

Step 1 – Disable Pingbacks From Other Blogs

In your WordPress dashboard, click into “Settings → Discussion.” Under the Discussion Settings section uncheck the option “Allow link notifications from other blogs (pingbacks and trackbacks) on new articles.”



Discussion Settings

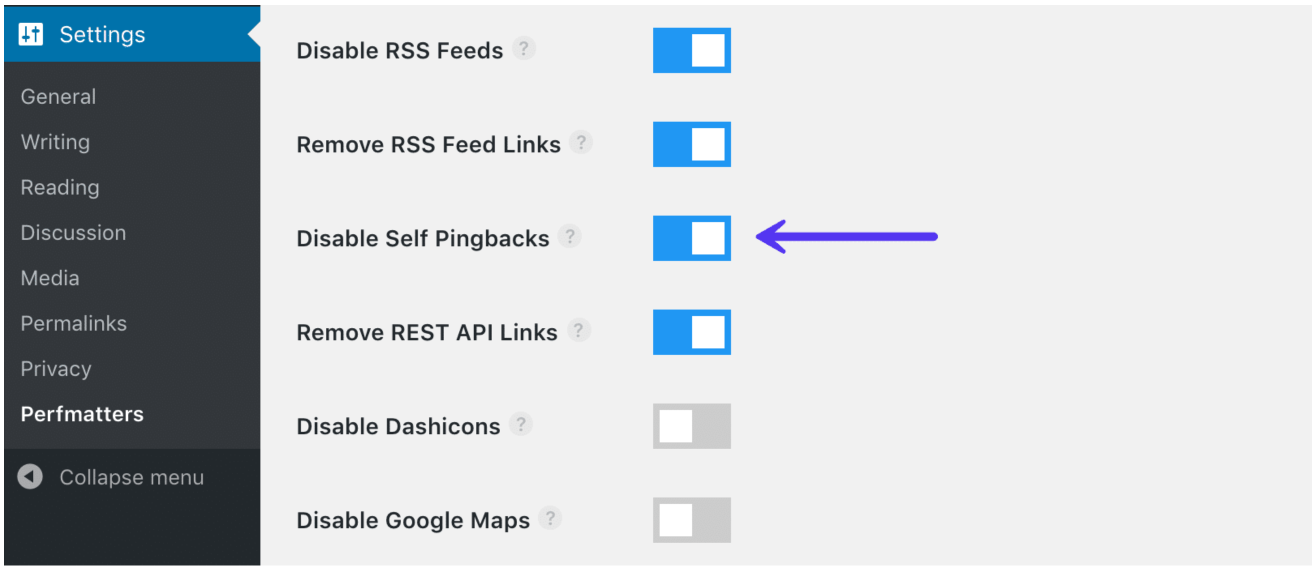
Default article settings

- ☐ Attempt to notify any blogs linked to from the article
- ☐ Allow link notifications from other blogs (pingbacks and trackbacks) on new articles
- ✓ Allow people to post comments on new articles

(These settings may be overridden for individual articles.)

Step 2 – Disable Self-Pingbacks

When it comes to disabling self-pingbacks you have a couple of options. You can use the free No Self Pings plugin. Or you can use a premium plugin like Perfmatters.



Disable RSS Feeds ●
Remove RSS Feed Links ●
Disable Self Pingbacks ●
Remove REST API Links ●
Disable Dashicons ○
Disable Google Maps ○

Alternatively, you could also disable self-pingbacks by adding the following code to your WordPress theme's functions.php file. Warning, editing the source of a WordPress theme could break your site if not done correctly. Tip, you can easily add PHP snippets like this with the free Code Snippets plugin. This means you never have to touch your theme.

```
function wpsites_disable_self_pingbacks( &$links ) {
  foreach ( $links as $l => $link )
        if ( 0 === strpos( $link, get_option( 'home' ) ) )
            unset($links[$l]);
}

add_action( 'pre_ping', 'wpsites_disable_self_pingbacks' );
```

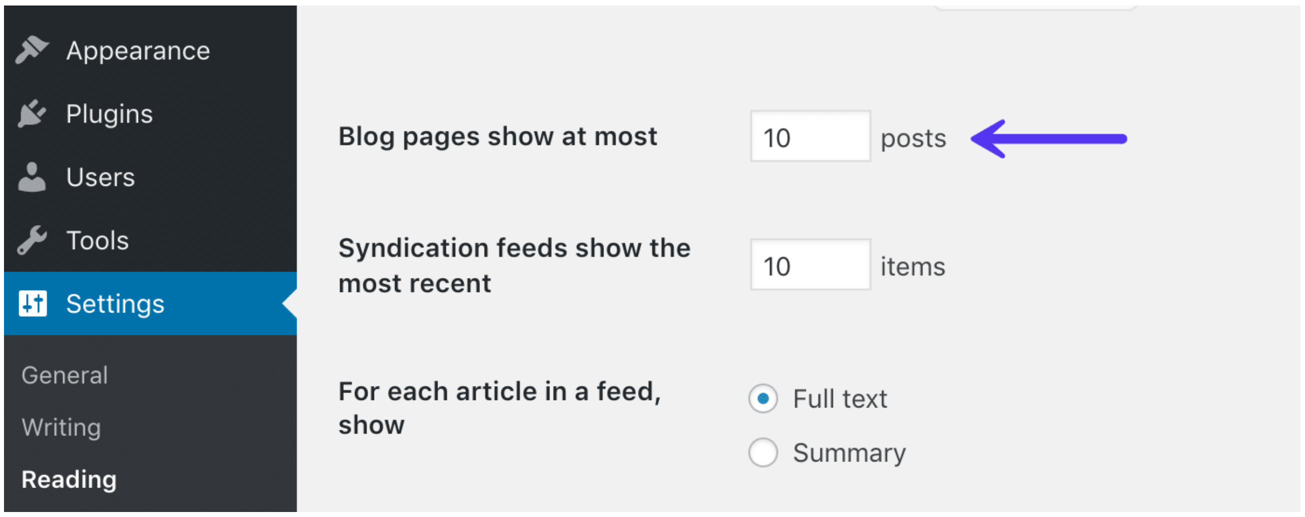
2:6 Limit Posts on Your Blog Feed

Whether your blog feed is set as your homepage or is another page of your site, you don't need 50 thumbnails all loading at the same time. For those that run high-traffic blogs, your homepage is the most important page of your site, and you want this to load fast. The fewer requests and media the better in terms of performance.

Also, this is precisely why pagination was invented (as seen below). Pagination is what you see at the end of blog feeds that allow you to browse to the next page. Typically these are numbers,or they might use “next/previous” posts. Your WordPress theme will most likely already have customized pagination built-in.

WordPress by default sets the limit on fresh WordPress installations to 10. Make sure to double check what value you’re using. We recommend somewhere between 8 and 12.

You can find this option in your WordPress admin dashboard under “Settings → Reading.” You can then change the value for “Blog pages show at most.”



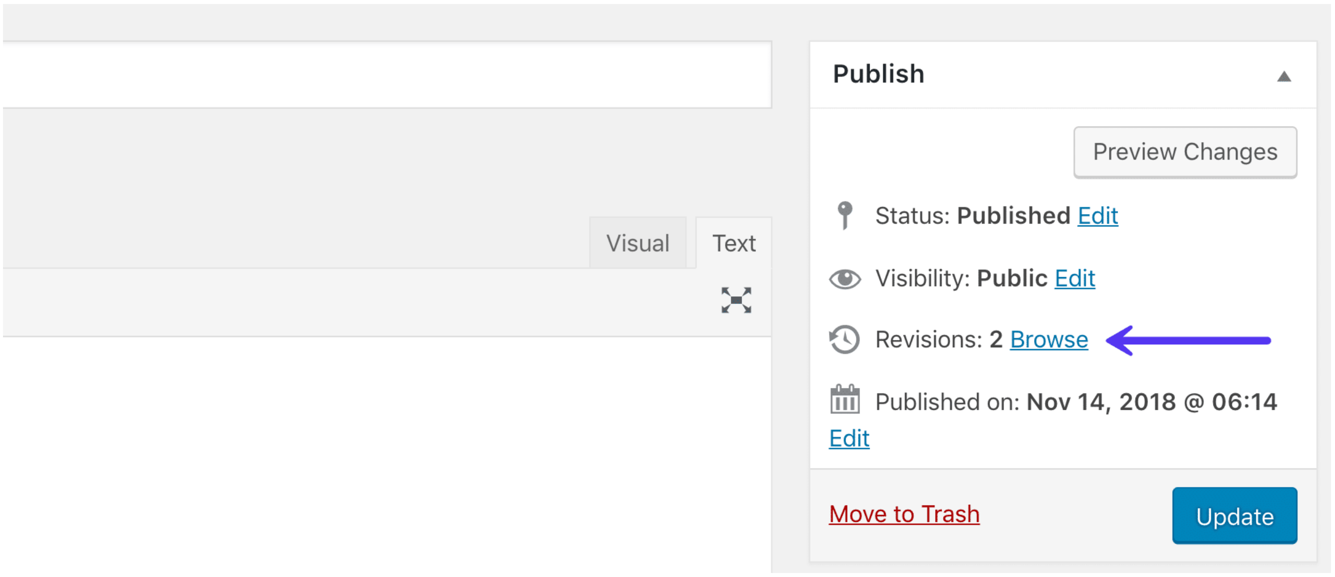
Blog pages show at most 10 posts

Syndication feeds show the most recent 10 items

For each article in a feed, show [✓ Full text] Summary

2:7 Delete and Limit Page and Post Revisions

Whenever you save a page or post in WordPress, it creates what is called a revision. This occurs in both drafts and already published posts that are updated. Revisions can be helpful in case you need to revert to a previous version of your content.



Status: **Published** Edit

Visibility: **Public** Edit

Revisions: **2** Browse

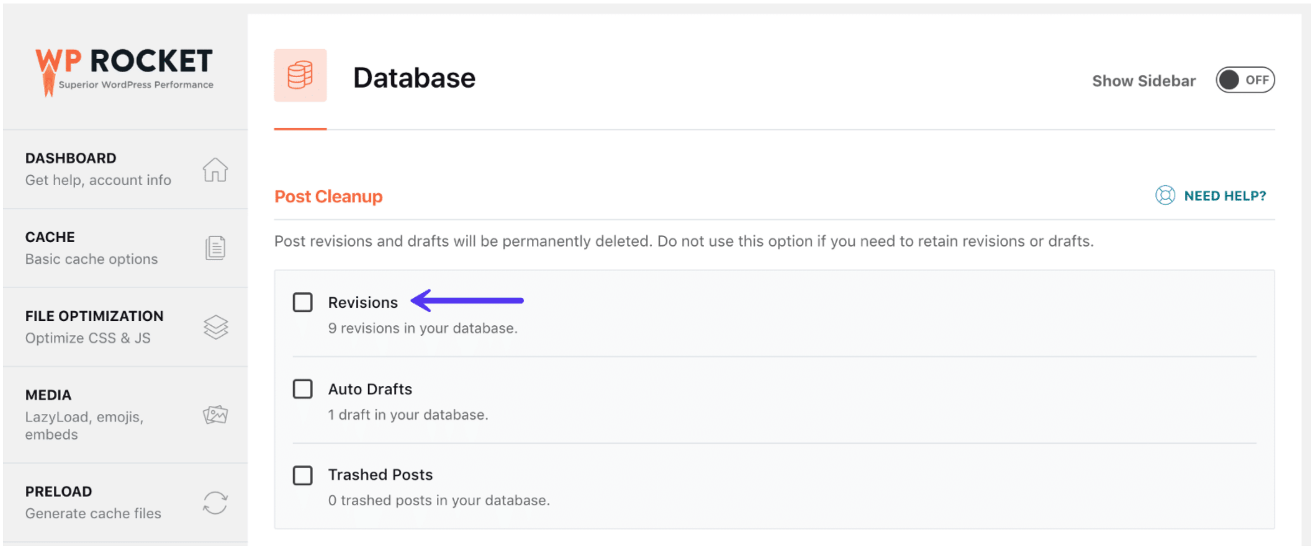
Published on: **Nov 14, 2018 @ 06:14** Edit

However, revisions can also hurt the performance of your WordPress site. On large sites, this can add up very quickly to thousands of rows in your database which are not necessarily needed. And the more rows you have, the larger your database in size, which takes up storage space. While indexes were created for this very purpose, we've still seen this issue cripple WordPress sites. There are a couple of things you can do.

1. Delete Old Revisions

If you have an older WordPress site with a lot of pages and posts, it might be time to do a quick cleanup and delete those old revisions. You can do this with MySQL, but with all the bad snippets of code floating around the web, we recommend doing a backup of your site and using a free plugin like WP-Sweep.

Another one of our favorite plugins, WP Rocket, also has a database optimization feature to clear out revisions.



WP ROCKET
Superior WordPress Performance

DASHBOARD
Get help, account info

CACHE
Basic cache options

FILE OPTIMIZATION
Optimize CSS & JS

MEDIA
LazyLoad, emojis, embeds

PRELOAD
Generate cache files

Database

Show Sidebar OFF

Post Cleanup

NEED HELP?

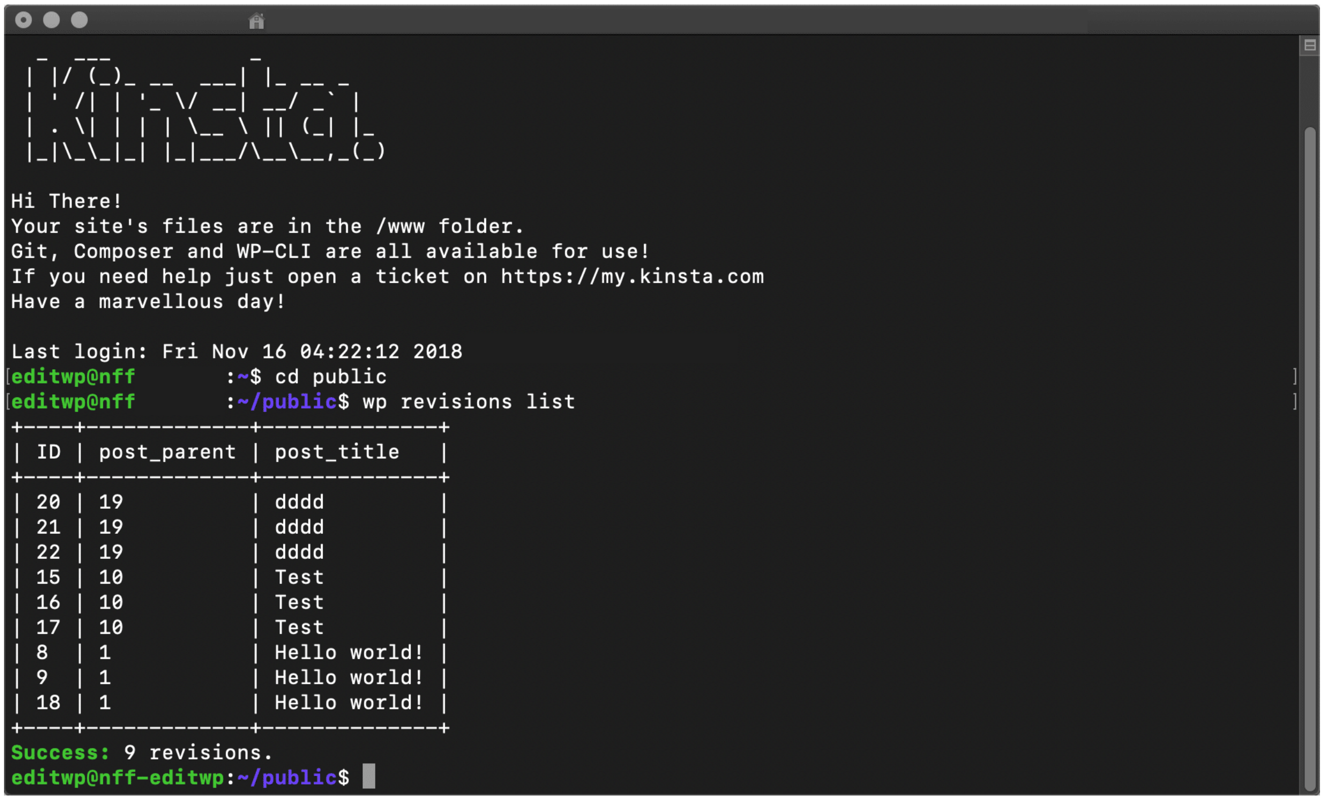
Post revisions and drafts will be permanently deleted. Do not use this option if you need to retain revisions or drafts.

- [ ] Revisions
  9 revisions in your database.
- [ ] Auto Drafts
  1 draft in your database.
- [ ] Trashed Posts
  0 trashed posts in your database.

If you’re handy with WP-CLI, there’s a couple of commands you can use for this.

Login to your server via SSH and run the following command to get and see the number of revisions currently in the database.

wp revisions list



```
  _  ___           _
 | |/ (_)_ __  ___| |_ __ _
 | ' /| | '_ \/ __| __/ _` |
 | . \| | | | \__ \ || (_| |_
 |_|\_\_|_| |_|___/\__\__,_(_)

Hi There!
Your site's files are in the /www folder.
Git, Composer and WP-CLI are all available for use!
If you need help just open a ticket on https://my.kinsta.com
Have a marvellous day!

Last login: Fri Nov 16 04:22:12 2018
editwp@nff       :~$ cd public
editwp@nff       :~/public$ wp revisions list
+----+-------------+--------------+
| ID | post_parent | post_title   |
+----+-------------+--------------+
| 20 | 19          | dddd         |
| 21 | 19          | dddd         |
| 22 | 19          | dddd         |
| 15 | 10          | Test         |
| 16 | 10          | Test         |
| 17 | 10          | Test         |
| 8  | 1           | Hello world! |
| 9  | 1           | Hello world! |
| 18 | 1           | Hello world! |
+----+-------------+--------------+
Success: 9 revisions.
editwp@nff-editwp:~/public$
```

If you get an error, you might need to first install the wp-revisions-cli package with the following command:

wp package install trepmal/wp-revisions-cli

You can then run the following command to clean up the revisions:

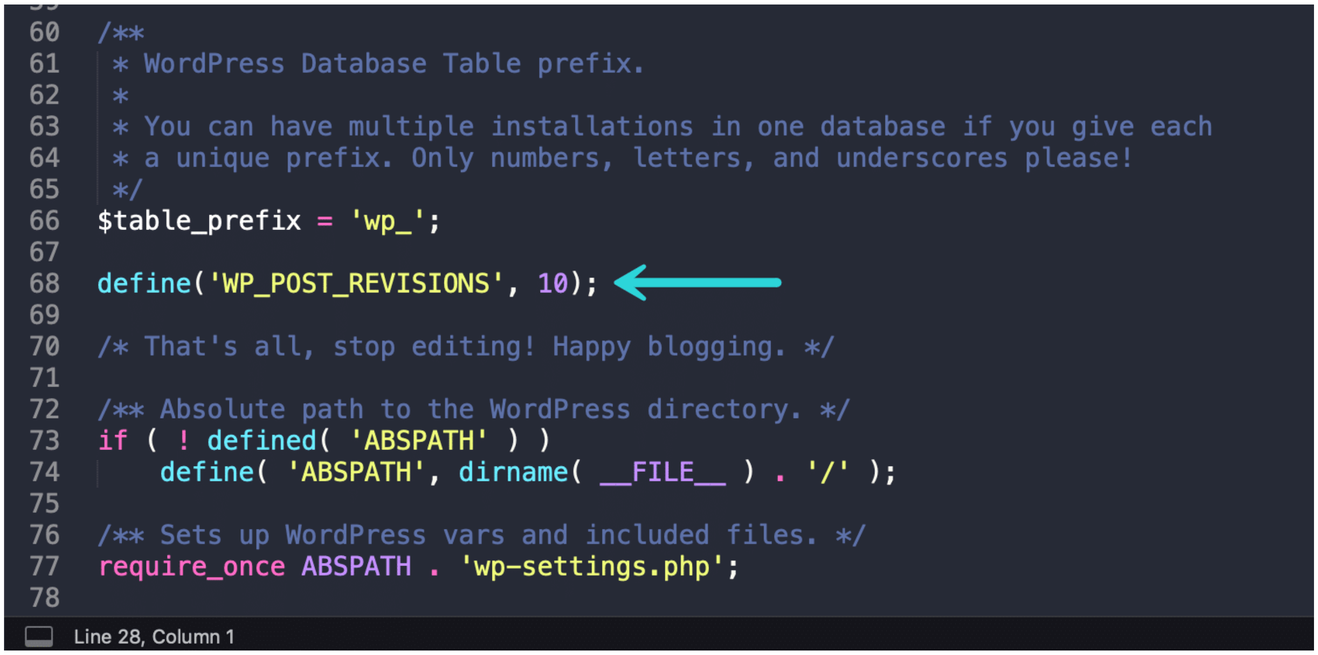
wp revisions clean

2. Limit Revisions

Another good strategy and one that we use at Kinsta is to limit the number of revisions that can be stored per post or page. Even setting it to something like ten will keep revisions from getting out of hand, especially if you do a lot of updating.

To limit revisions, you can add the following code to your *wp-config.php* file. The code below needs to be inserted above the 'ABSPATH' otherwise it won't work. You can change the number to however many revisions you want to keep stored in your database.

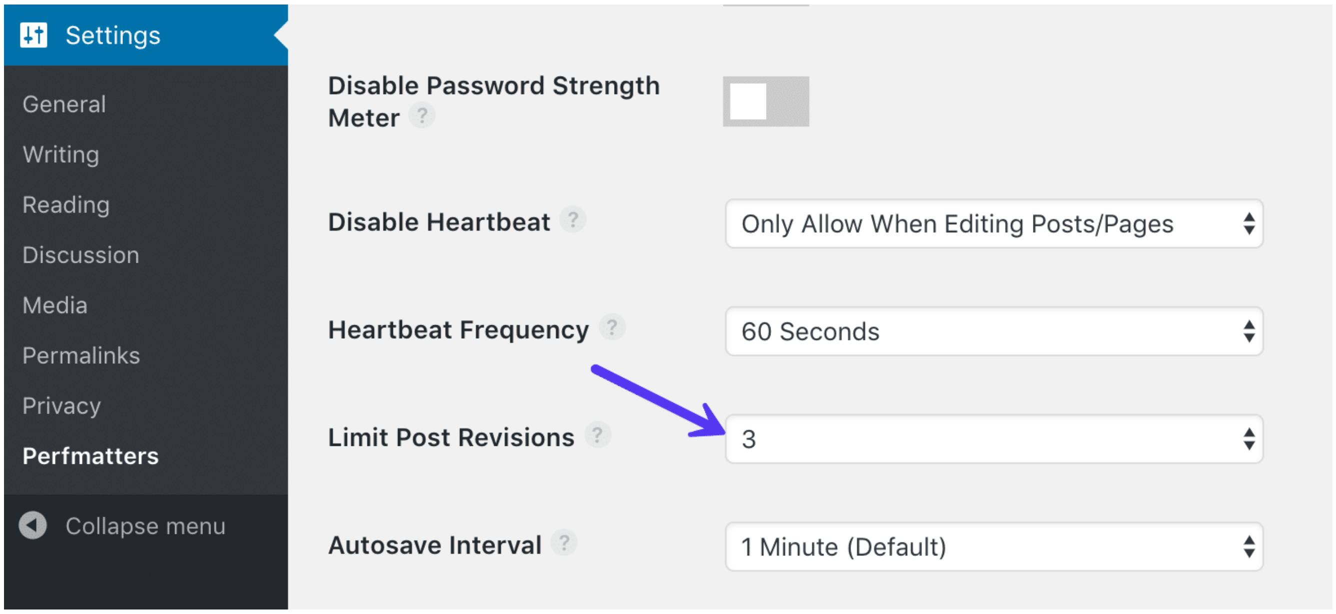
define('WP_POST_REVISIONS', 10);



```
/**
 * WordPress Database Table prefix.
 *
 * You can have multiple installations in one database if you give each
 * a unique prefix. Only numbers, letters, and underscores please!
 */
$table_prefix = 'wp_';

define('WP_POST_REVISIONS', 10);

/* That's all, stop editing! Happy blogging. */

/** Absolute path to the WordPress directory. */
if ( ! defined( 'ABSPATH' ) )
    define( 'ABSPATH', dirname( __FILE__ ) . '/' );

/** Sets up WordPress vars and included files. */
require_once ABSPATH . 'wp-settings.php';
```

Line 28, Column 1

Or you can utilize a premium plugin like Perfmatters to limit revisions.



Disable Password Strength Meter	
Disable Heartbeat	Only Allow When Editing Posts/Pages
Heartbeat Frequency	60 Seconds
Limit Post Revisions	3
Autosave Interval	1 Minute (Default)

2:8 Offload Media From Your WordPress Site

Everything that generates an HTTP Request has an impact on your site’s performance. For sites hosting hundreds of thousands of images, videos, or other large media, it may be wise to offload this completely.

Offloading is different from serving it up via a CDN. With a CDN the original data still resides at your host, the CDN makes copies of it and stores it closer to your visitors.

When caching expires on your CDN assets it queries your host again for the latest copies of the files. CDNs are meant to cache files for long periods of time. But due to the fact that they have so many POPs, there could be a lot of re-querying going on as cache expires in different regions.

When you offload media or files it means actually moving the original physical location of them off of your hosting provider. So while it might appear that the files are served from your site, they are really located somewhere else entirely. Besides reducing additional queries back to the host, the number one reason obviously is to also save on disk space.

Offload Media to Amazon S3

One of the most popular offloading solutions is Amazon S3. Amazon S3 is a storage solution, and part of Amazon Web Services many products. Typically this is used for large sites that either need additional backups or are serving up large files (downloads, software, videos, games, audio files, PDFs, etc.). Amazon has a proven track record of being very reliable, and because of their massive infrastructure, they can offer very low storage costs. Some of S3’s customers include Netflix, Airbnb, SmugMug, Nasdaq, etc.

Because they deal entirely with bulk storage, you can almost guarantee that pricing will be cheaper than your WordPress host. Offloading media to AWS can be a great way to save money and is free for your first year (up to 5 GB storage). Also, because the requests for your media are served directly from Amazon, this puts less load on your WordPress site, meaning faster load times.

Check out our in-depth tutorial on how to offload WordPress media to Amazon S3. You can also use a CDN with the offloaded media for the best of both worlds.

Offload Media to Google Cloud Storage

Another popular offloading solution is Google Cloud Storage. Since Kinsta is powered by Google Cloud Platform, we are big fans of their technology and infrastructure. Due to Google's massive infrastructure and the fact that they deal with storage in bulk, they can offer very low storage costs. Some of their customers include Spotify, Vimeo, Coca-Cola, Philips, Evernote, and Motorola.

Check out our in-depth tutorial on how to offload WordPress media to Google Cloud Storage.

2:9 Offload Email From Your WordPress Site

Similar to offloading media (above) if you are sending a lot of transactional or marketing emails, it may be best for your use case to have all that work happen off your main WordPress site.

Offload Transactional and Marketing Emails

Whether you think so or not, emails do have an impact on your server and server resources. With some hosts, especially shared hosts, abusing this could even get you suspended. This especially becomes a problem with those trying to send bulk emails. This is the reason why third-party transactional email providers exist and why a lot of hosting providers block email delivery on standard ports altogether. We never recommend using your hosting provider for email.

If you are sending newsletters or bulk emails, we always recommend the following alternatives to get the best results:

- Use a **third-party professional email marketing software** that isn't part of WordPress
- Use a **transactional email service provider** (HTTP API or SMTP) along with WordPress

Other advantages of using a third-party service include:

- Better email deliverability. Let the email providers do what they do best!
- Less chance to get blacklisted.
- It might not always be possible to set up DMARC records with your hosting provider.

Email Marketing Tools

Some examples of marketing emails include newsletters, product and feature announcements, sales, event invitations, onboarding reminders, etc. Here are a few email marketing tools we recommend:

- MailChimp
- MailerLite
- Drip

Transactional Email Services

Some examples of transactional emails include purchase receipts from WooCommerce or EDD, account creation notifications, shipping notifications, app error messages, password resets, etc. If you're a Kinsta client, we rely on a third-party SMTP provider to ensure high deliverability. But depending on your volume, we always recommend moving this offsite. Here are a few transaction email services we recommend:

- SendGrid
  - Learn how to configure SendGrid in WordPress.
- Mailgun
  - Learn how to configure Mailgun in WordPress.
- SparkPost

2:10 Use In-Memory Data Storage for WordPress

Redis and Memcahced are open-source, in-memory data structure stores. In the context of WordPress, Redis can be used to store the values generated by WordPress' native object cache persistently so that cached objects can be reused between page loads.

Using a persistent object cache such as Redis allows for the **reuse of cached objects** rather than requiring the MySQL database to be queried a second time for the same object. The result is that Redis can reduce the load on a website's MySQL database, simultaneously decreasing the response time of the site and increasing the site's ability to scale and handle additional traffic.

Highly dynamic websites (WooCommerce, Easy Digital Downloads, membership sites, forums, discussion boards, and blogs with extremely active comment systems) often can't make good use of page caching are potential candidates for a persistent object caching option such as Redis.

- Read our article comparing Redis and Memcached
- If you're a Kinsta client, we offer a Redis add-on, check out how to add Redis to your hosting plan.

2:11 Use Elasticsearch to Speed Up WordPress Search

Elasticsearch is an open-source full-text search engine. It is used to index data and search that data incredibly quickly.

In the context of WordPress, Elasticsearch can be used to **speed up querying of the WordPress database**. This is done by building an index of the content of your site's database and then using Elasticsearch to search this index much more quickly than a MySQL query is capable of performing the same search.

If you have the time and ability, Elasticsearch can be integrated with a WordPress site by a highly knowledgeable WordPress and Elasticsearch developer. If your site makes relatively standard use of WP_Query, Elasticsearch can also be integrated by installing ElasticPress, a free WordPress plugin from 10up, available on WordPress.org, which automatically integrates with the WP_Query object to generate query results with Elasticsearch rather than MySQL.

Any site that makes heavy use of WP_Query can benefit from Elasticsearch. Examples of sites that can benefit from Elasticsearch:

- Sites where search is the primary means of navigation.
- WooCommerce sites with a huge number of orders where site admins need to be able to search the list of orders regularly.
- Any site with a large number of posts where MySQL queries are producing unacceptably slow results.

Just like with Redis, we also have an Elasticsearch add-on. Check out how to add Elasticsearch to your hosting plan.

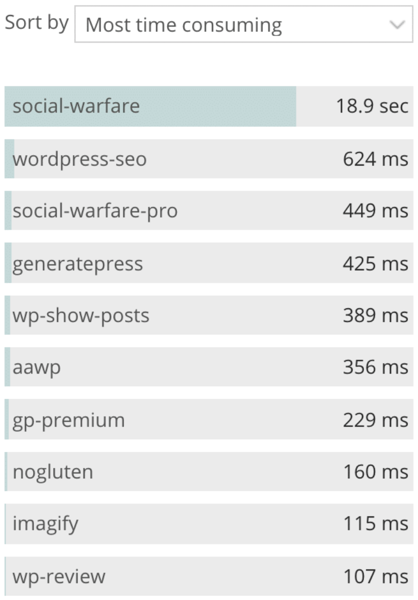
2:12 Disable Non-Critical Features That Are Database-Intensive

This might seem a little obvious, but it can make a world of difference if you disable non-critical plugins and theme features that are database-intensive.

- Popular and or related post widgets and plugins are horrible. They typically have heavy sitewide queries.
- Image optimization plugins that compress images using your server. You should always use an image optimization plugin that optimizes images externally.

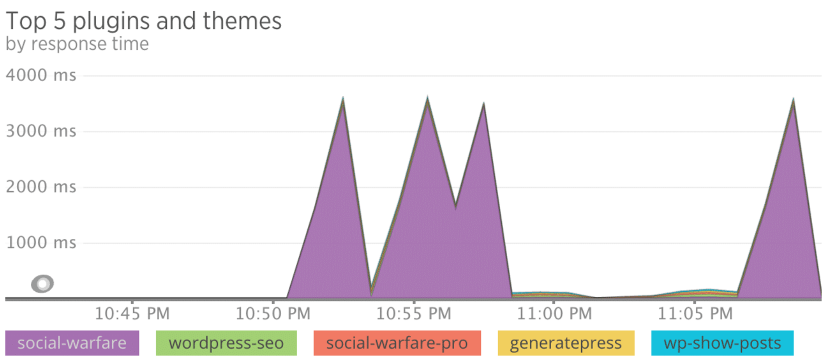
We also recommend staying away from plugins that add a view/post counter to your site, unless you absolutely need it. For example, avoid things like “792 posts” next to a user’s avatar in forum posts or “5,243 views” when listing forum posts. When you have a long discussion, these counters will take a huge toll on your database. In general, minimize the use of counters and only use them if necessary.

This also goes for a lot of social counters. For example, on this site below you can see the response time from the popular Social Warfare plugin is 30x more than the next plugin below it. Caching is enabled, but obviously, this plugin has a considerable performance toll. After disabling the plugin on the site, load times instantly improved and the responsiveness of the WordPress admin dashboard improved.



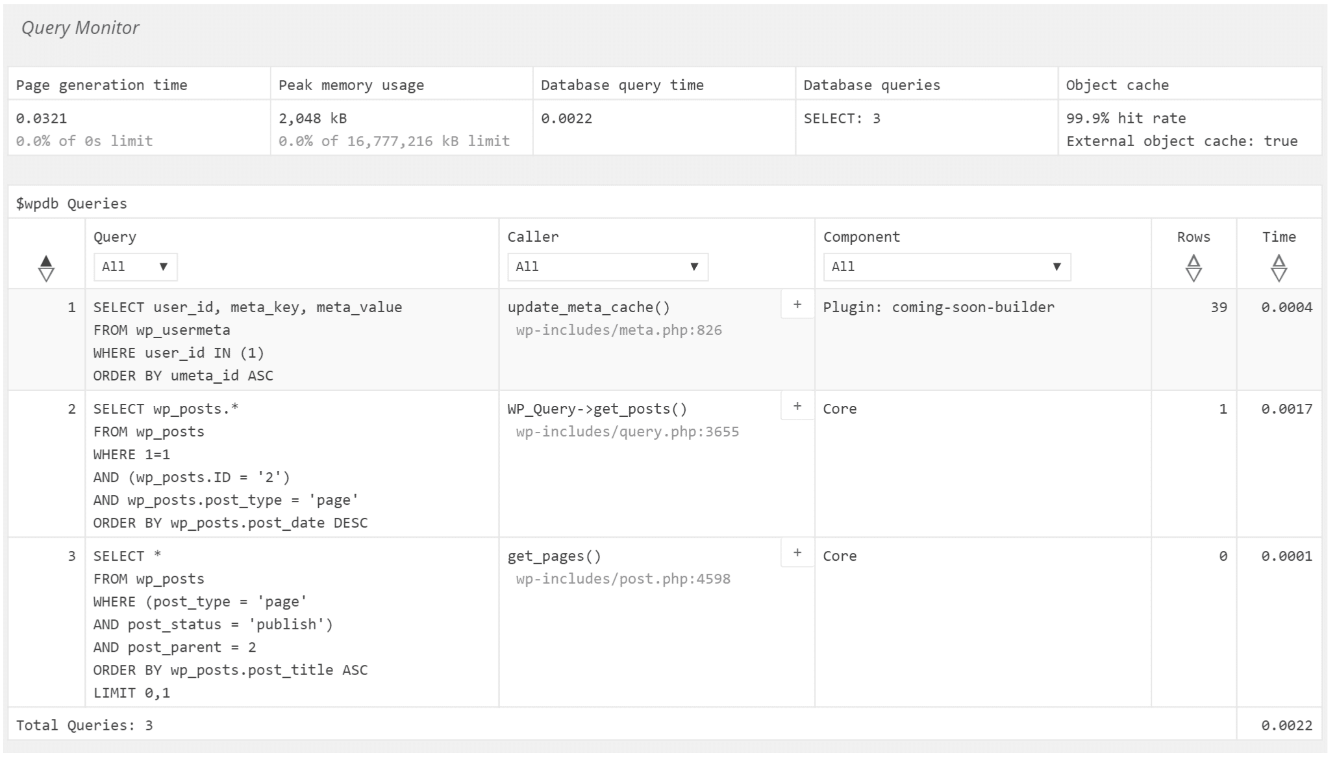
Sort by Most time consuming

social-warfare	18.9 sec
wordpress-seo	624 ms
social-warfare-pro	449 ms
generatepress	425 ms
wp-show-posts	389 ms
aawp	356 ms
gp-premium	229 ms
nogluten	160 ms
imagify	115 ms
wp-review	107 ms



2:13 Use the Free Query Monitor Plugin

You can also use the free Query Monitor WordPress plugin. Use it to identify and debug slow database queries, AJAX calls, REST API requests, and much more. In addition, the plugin reports back website details such as script dependencies and dependents, WordPress hooks that fired during page generation, hosting environment details, conditional query tags met by the current page, and a lot more.



Query Monitor

Page generation time	Peak memory usage	Database query time	Database queries	Object cache
0.0321 0.0% of 0s limit	2,048 kB 0.0% of 16,777,216 kB limit	0.0022	SELECT: 3	99.9% hit rate External object cache: true

$wpdb Queries

	Query	Caller	Component	Rows	Time
	All	All	All		

1	SELECT user_id, meta_key, meta_value
	FROM wp_usermeta
	WHERE user_id IN (1)
	ORDER BY umeta_id ASC
		update_meta_cache()
		wp-includes/meta.php:826
			Plugin: coming-soon-builder	39	0.0004

2	SELECT wp_posts.*
	FROM wp_posts
	WHERE 1=1
	AND (wp_posts.ID = '2')
	AND wp_posts.post_type = 'page'
	ORDER BY wp_posts.post_date DESC
		WP_Query->get_posts()
		wp-includes/query.php:3655
			Core	1	0.0017

3	SELECT *
	FROM wp_posts
	WHERE (post_type = 'page'
	AND post_status = 'publish')
	AND post_parent = 2
	ORDER BY wp_posts.post_title ASC
	LIMIT 0,1
		get_pages()
		wp-includes/post.php:4598
			Core	0	0.0001

Total Queries: 3					0.0022

The plugin was developed by John Blackbourn, a core WordPress committer and well-known developer, someone who knows WordPress extensively. Query Monitor was added to the WordPress plugin directory in 2013 and currently boasts more than 100,000 active installs – an impressive sum for a development plugin. The plugin's user rating of five out of five stars helps explain its popularity among developers.

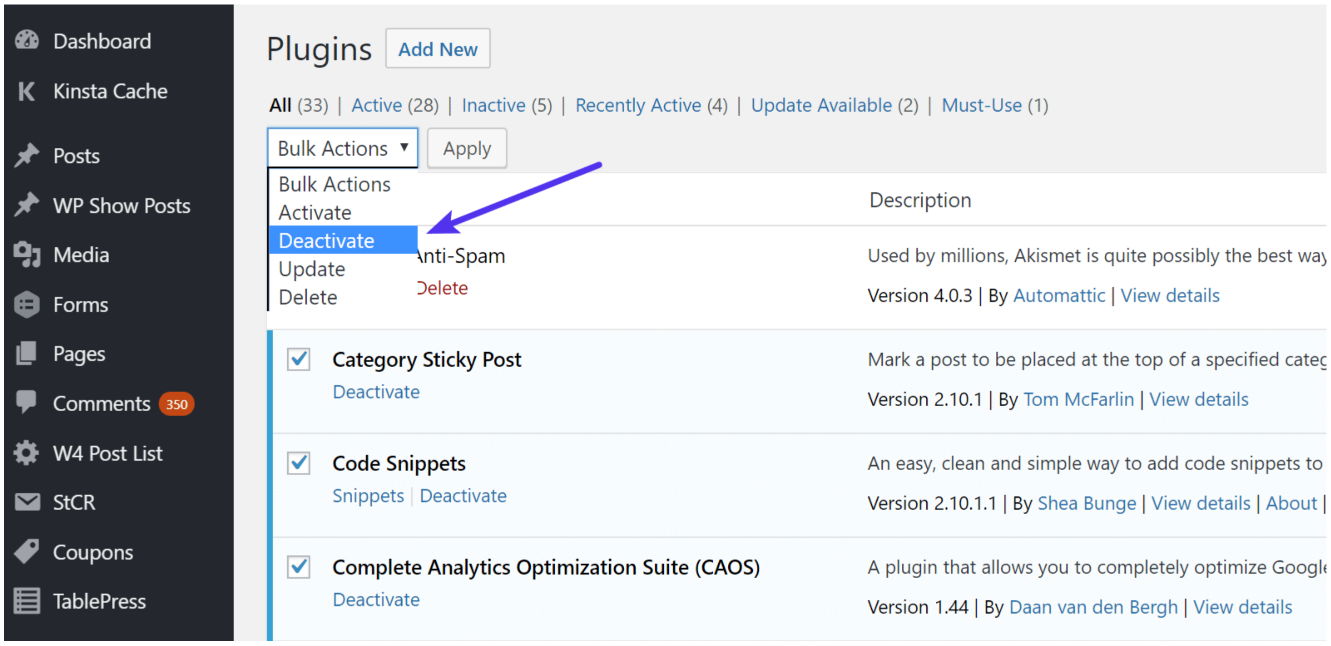
Check out our complete tutorial on how to use Query Monitor.

2:14 Utilize Staging Sites Without Touching Production

We don't know what we would do without staging environments. These can be invaluable when it comes to troubleshooting performance issues. Thankfully, Kinsta has one-click staging environments. If your WordPress host doesn't offer staging environments, you could also use a plugin like WP Staging, although it's not as easy.

- Learn more about setting up a staging environment

After you have a staging site up and running, the first thing you can do is disable all of your plugins. Since this is a copy of your live site, you don't have to worry about breaking anything. It's by far one of the easiest ways to narrow down issues. Simply go to Plugins, select all of them and choose "Deactivate" from the bulk options.



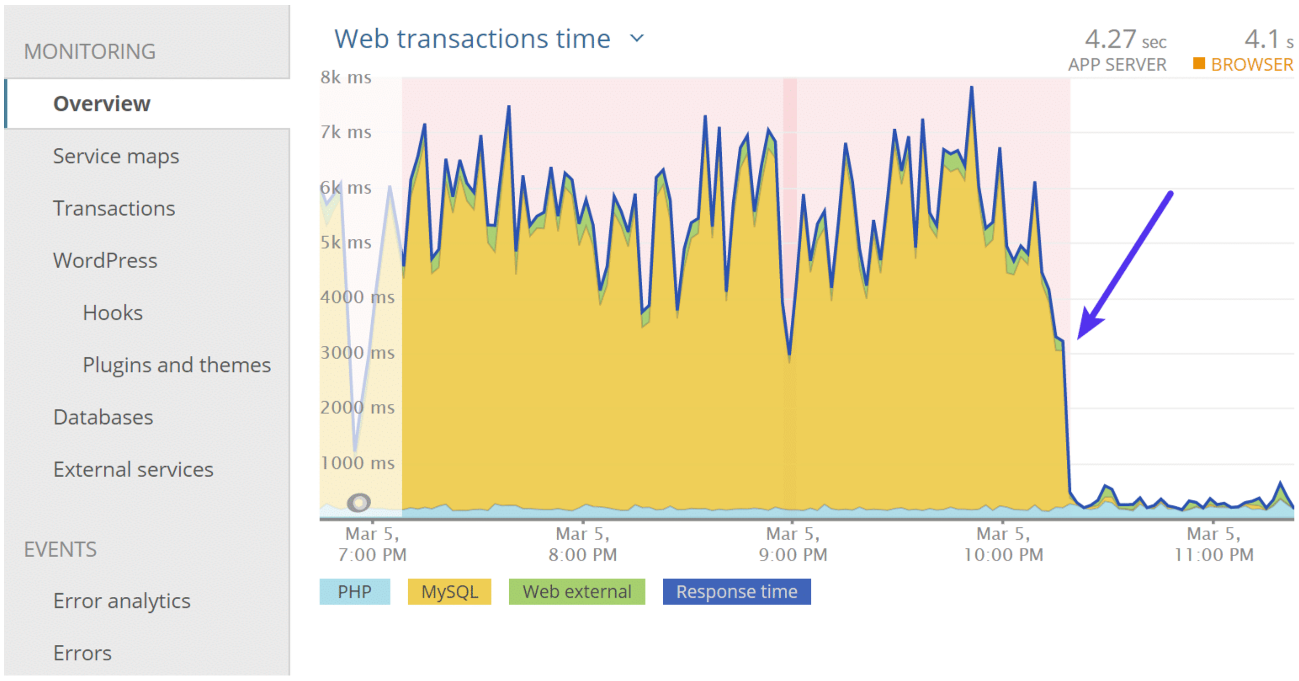
Anti-Spam
Delete
Used by millions, Akismet is quite possibly the best way
Version 4.0.3 | By Automattic | View details

Category Sticky Post
Deactivate
Mark a post to be placed at the top of a specified categ
Version 2.10.1 | By Tom McFarlin | View details

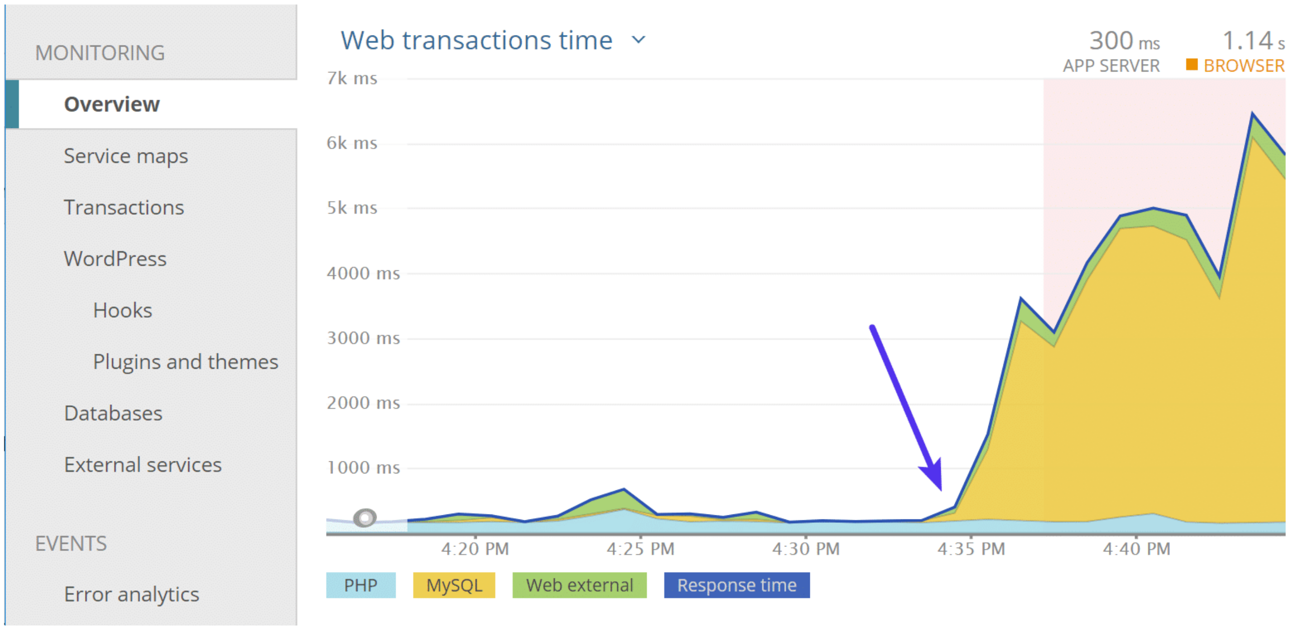
Code Snippets
Snippets | Deactivate
An easy, clean and simple way to add code snippets to
Version 2.10.1.1 | By Shea Bunge | View details | About |

Complete Analytics Optimization Suite (CAOS)
Deactivate
A plugin that allows you to completely optimize Googl
Version 1.44 | By Daan van den Bergh | View details

After doing this, you can monitor response times in New Relic, our Kinsta APM tool, or Query Monitor (see above). See what happens after making the change. In this example below the response times immediately dropped back down to normal on the site, so we knew it was one of the plugins causing an issue. You can then re-enable them one by one, repeating the same process until you find the culprit.



Here is an example of what happened when we enabled the plugin that was causing the problem. Load times (web transaction times) immediately went back up.



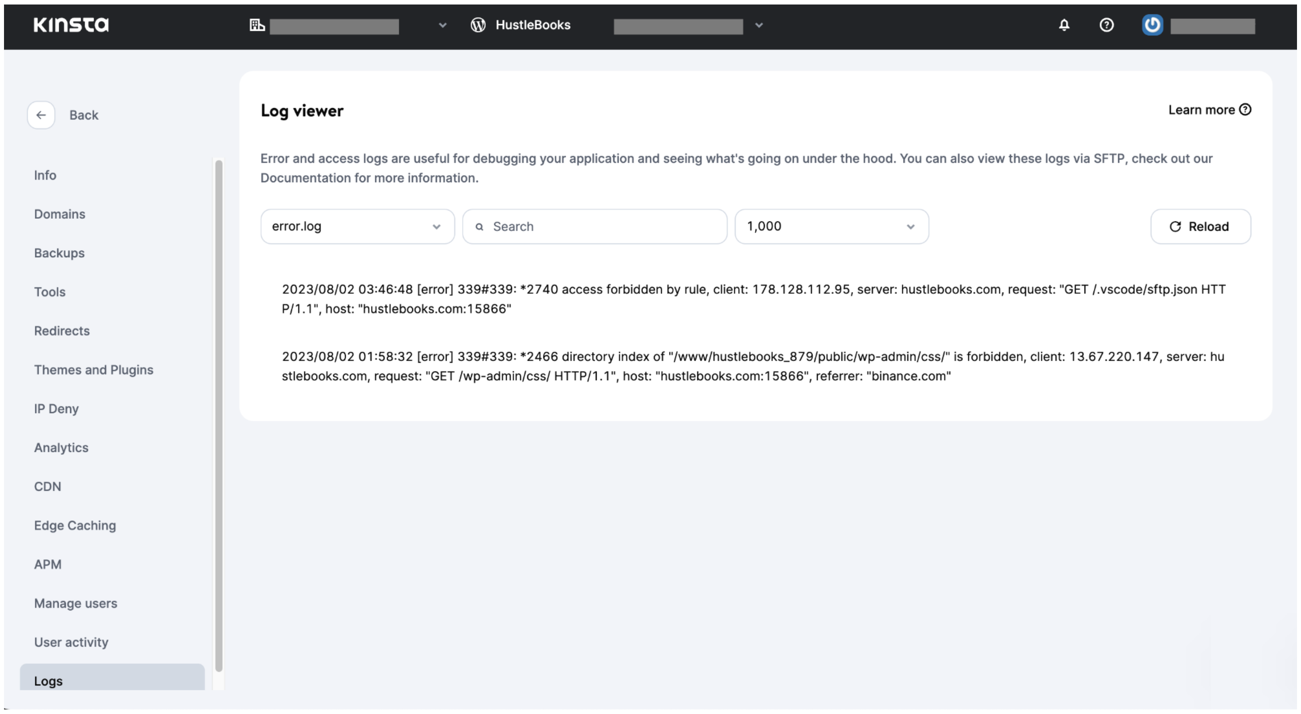
What should you do after you find the plugin causing the slowness? Here is what we advise:

1. Update your plugins and themes to the latest version if you haven't already.
2. Reach out to the developer of the plugin or theme and ask them for assistance.
3. Find an alternative plugin that can deliver the same functionality.
4. Perhaps your PHP version is causing an issue. Change your PHP engine to a lower version and see if the plugin or theme then works.
5. You may want help and can hire a WordPress developer to fix the issue.

2:15 Check Your Error Logs

Checking error logs is never fun, but can reveal a lot about performance issues with WordPress plugins.

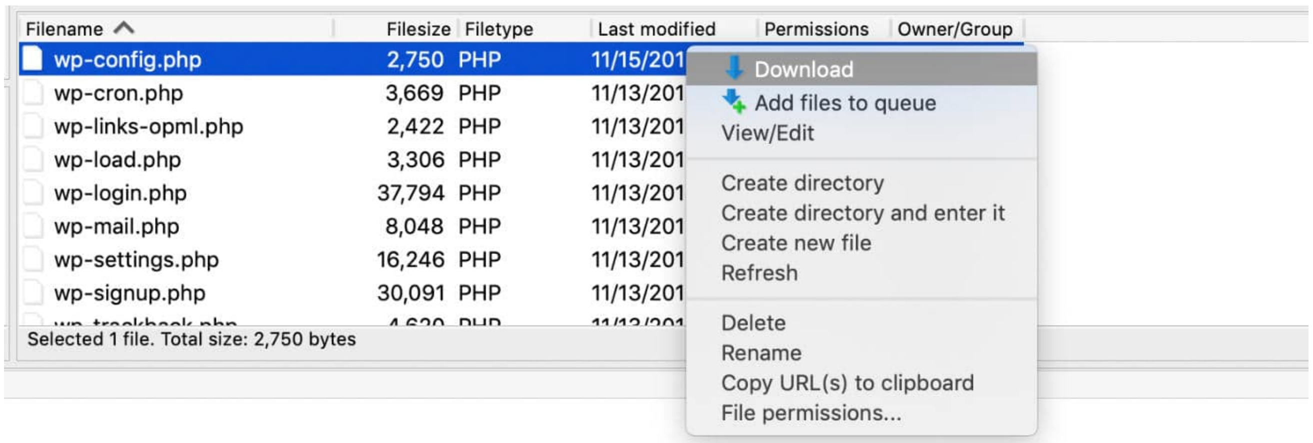
If you're a Kinsta client, you can easily view your error logs, cache logs, and access logs right from the MyKinsta dashboard.



Error and access logs are useful for debugging your application and seeing what's going on under the hood. You can also view these logs via SFTP, check out our Documentation for more information.

```
2023/08/02 03:46:48 [error] 339#339: *2740 access forbidden by rule, client: 178.128.112.95, server: hustlebooks.com, request: "GET /.vscode/sftp.json HTT
P/1.1", host: "hustlebooks.com:15866"

2023/08/02 01:58:32 [error] 339#339: *2466 directory index of "/www/hustlebooks_879/public/wp-admin/css/" is forbidden, client: 13.67.220.147, server: hu
stlebooks.com, request: "GET /wp-admin/css/ HTTP/1.1", host: "hustlebooks.com:15866", referrer: "binance.com"
```

You can also enable error logs by adding some code to your *wp-config.php* file. First, you will want to connect to your site via SFTP. Then download your *wp-config.php* so you can edit it. Note: Always make a backup of this file first!

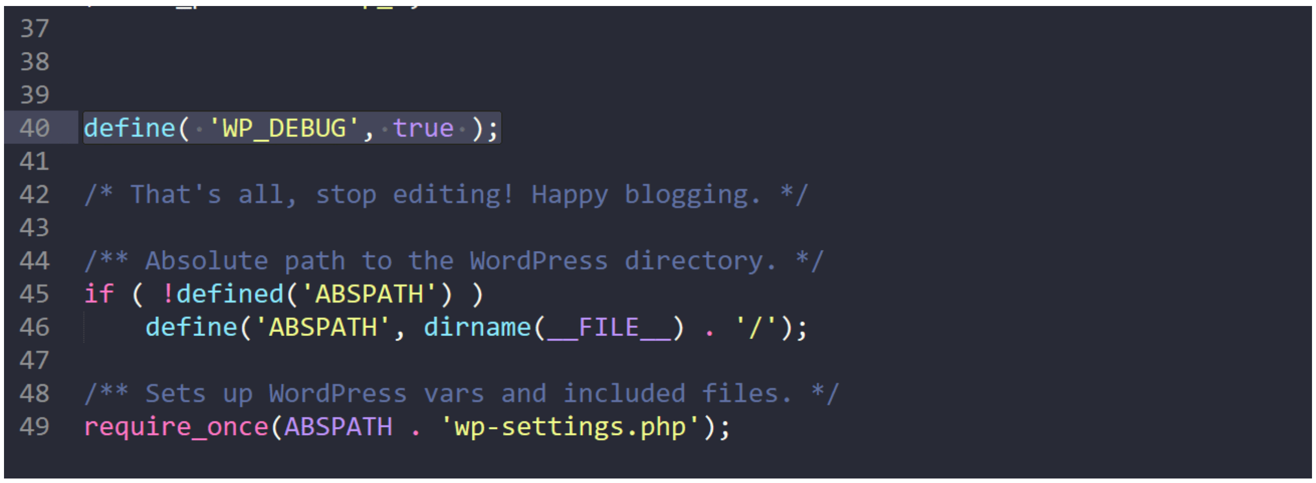


Filename	Filesize	Filetype	Last modified	Permissions	Owner/Group
wp-config.php	2,750	PHP	11/15/201		
wp-cron.php	3,669	PHP	11/13/201		
wp-links-opml.php	2,422	PHP	11/13/201		
wp-load.php	3,306	PHP	11/13/201		
wp-login.php	37,794	PHP	11/13/201		
wp-mail.php	8,048	PHP	11/13/201		
wp-settings.php	16,246	PHP	11/13/201		
wp-signup.php	30,091	PHP	11/13/201		

Selected 1 file. Total size: 2,750 bytes

Find the line that says /* That's all, stop editing! Happy blogging. */ and just before it, add the following (as seen below):

define('WP_DEBUG', true);

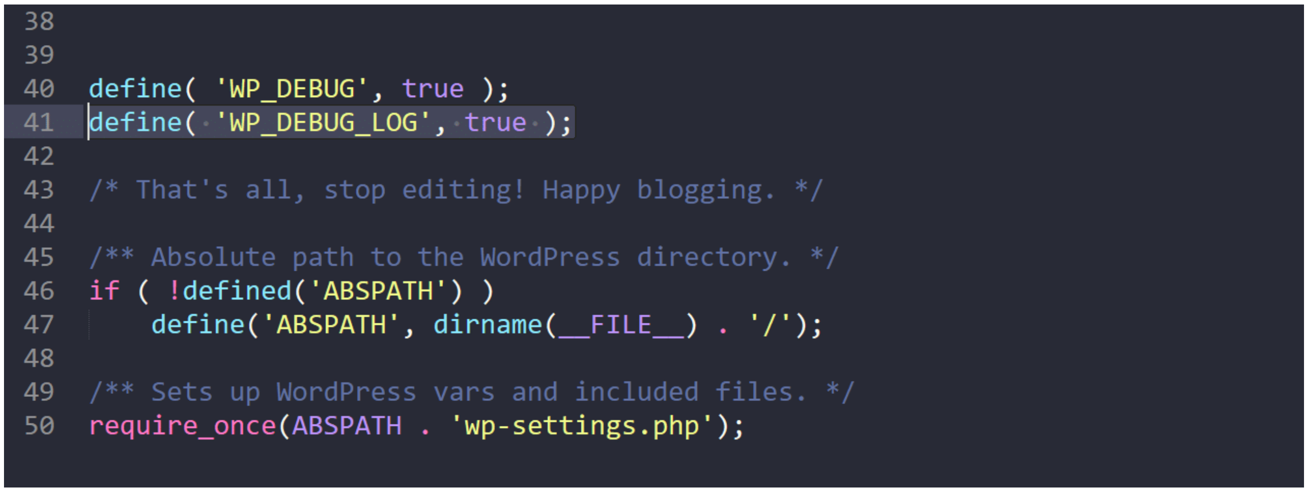


```php



define( 'WP_DEBUG', true );

/* That's all, stop editing! Happy blogging. */

/** Absolute path to the WordPress directory. */
if ( !defined('ABSPATH') )
    define('ABSPATH', dirname(__FILE__) . '/');

/** Sets up WordPress vars and included files. */
require_once(ABSPATH . 'wp-settings.php');
```

If the above code already exists in your *wp-config.php* file but is set to “false,” simply change it to “true.” This will enable debug mode. Note: You will also see warnings or errors in your WordPress admin if they exist.

You can then enable the debug log to send all errors to a file by adding the following code just after the *WP_DEBUG* line (as seen below):

define('WP_DEBUG_LOG', true);



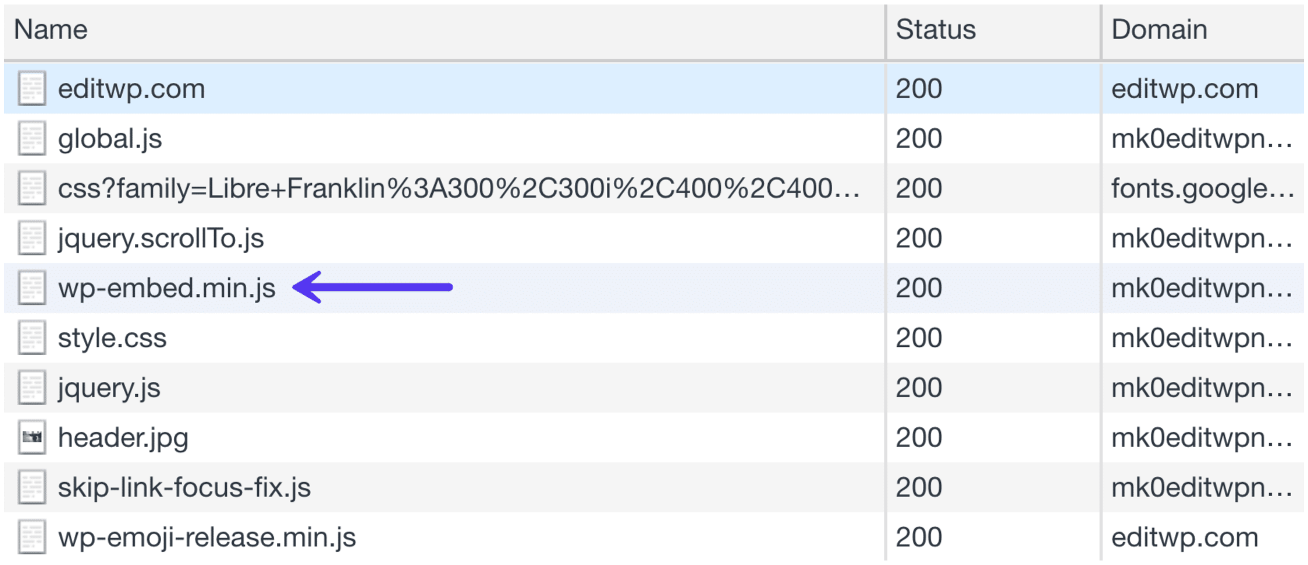
```
define( 'WP_DEBUG', true );
define( 'WP_DEBUG_LOG', true );

/* That's all, stop editing! Happy blogging. */

/** Absolute path to the WordPress directory. */
if ( !defined('ABSPATH') )
    define('ABSPATH', dirname(__FILE__) . '/');

/** Sets up WordPress vars and included files. */
require_once(ABSPATH . 'wp-settings.php');
```

Save your changes and re-upload this to your server. The errors will then get logged to the *debug.log* file within your */wp-content/* folder. If for some reason you don't see this file, you can always create one.

2:16 Disable Embeds in WordPress

When they released WordPress 4.4, they merged the oEmbed feature into core. This allows users to embed YouTube videos, tweets and many other resources on their sites simply by pasting a URL, which WordPress automatically converts into an embed and provides a live preview in the visual editor. With the update, WordPress itself became an oEmbed provider.

This feature is useful for a lot of people, and you may want to keep it enabled. However, what this means is that it also generates an additional HTTP request on your WordPress site to load the *wp-embed.min.js* file. And this loads site-wide. While this file is only 1.7 KB, things like these add up over time. The request itself is sometimes a bigger deal than the content download size.



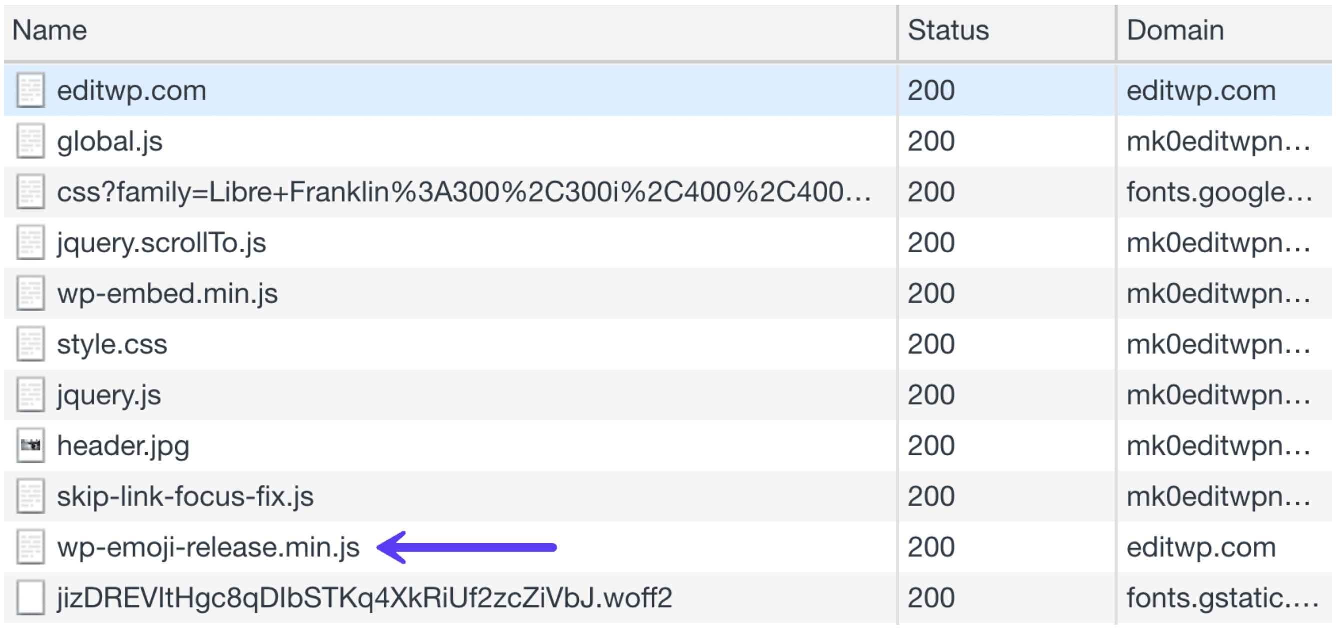
Name	Status	Domain
editwp.com	200	editwp.com
global.js	200	mk0editwpn…
css?family=Libre+Franklin%3A300%2C300i%2C400%2C400…	200	fonts.google…
jquery.scrollTo.js	200	mk0editwpn…
wp-embed.min.js	200	mk0editwpn…
style.css	200	mk0editwpn…
jquery.js	200	mk0editwpn…
header.jpg	200	mk0editwpn…
skip-link-focus-fix.js	200	mk0editwpn…
wp-emoji-release.min.js	200	editwp.com

You can easily disable this file from loading. Here are three different options:

- Option 1 – Disable Embeds with Plugin
- Option 2 – Disable Embeds with Code
- Option 3 – Move the JavaScript Inline

2:17 Disable Emojis in WordPress

Similar to embeds, in WordPress 4.2, they added support for emojis into core for older browsers. The big issue with this is that it generates an additional HTTP request on your WordPress site to load the *wp-emoji-release.min.js* file. And this loads site-wide. While this file is only 10.5 KB, it's useless if you're not using emojis on your site.



Name	Status	Domain
editwp.com	200	editwp.com
global.js	200	mk0editwpn…
css?family=Libre+Franklin%3A300%2C300i%2C400%2C400…	200	fonts.google…
jquery.scrollTo.js	200	mk0editwpn…
wp-embed.min.js	200	mk0editwpn…
style.css	200	mk0editwpn…
jquery.js	200	mk0editwpn…
header.jpg	200	mk0editwpn…
skip-link-focus-fix.js	200	mk0editwpn…
wp-emoji-release.min.js	200	editwp.com
jizDREVItHgc8qDIbSTKq4XkRiUf2zcZiVbJ.woff2	200	fonts.gstatic.…

There are a couple of different ways to disable Emojis in WordPress. You can do it with a free plugin or with code.

- Disable Emojis with a Plugin
- Disable Emojis with Code

2:18 How to Speed Up WordPress Comments or Disable Them

A busy comment section on a site can cause a lot of performance issues. Just think about the resources that go into making comments work:

- A database is queried to pull up existing comments.
- Database entries are created for each new comment.
- Comments and comment metadata are received and processed by a visitor's browser.
- External resources, such as Gravatars, are requested, downloaded, and loaded (requiring a separate DNS lookup).
- In many cases, large JavaScript and jQuery resources have to be downloaded and processed to make the commenting system work the way it's supposed to.

Here are four different options you can do to speed up WordPress comments:

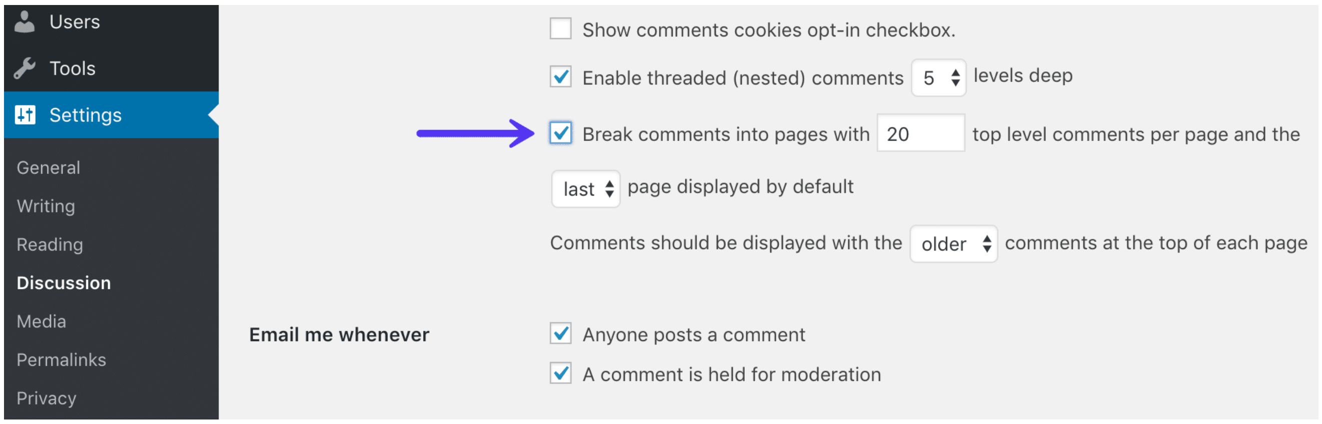
Option 1 – Disable Comments

If your site isn't getting very many comments and you don't think they are adding any value, it might be better to disable comments altogether. Remember, comments can impact your SEO as Google will typically crawl these as additional content on the page, so you should only approve high-quality comments.

Option 2 – Optimize Native WordPress Comments

Your second option would be to optimize the native WordPress comment system. One way would be to reduce the number of comments loaded with the initial page load.

1. Go to Settings → Discussion in the WordPress admin area.
2. Look for the Other comment settings section.
3. Select the checkbox next to Break comments into pages with and add a value for the number of comments you want to display with the initial page load.



☐ Show comments cookies opt-in checkbox.

☑ Enable threaded (nested) comments 5 levels deep

☑ Break comments into pages with 20 top level comments per page and the last page displayed by default

Comments should be displayed with the older comments at the top of each page

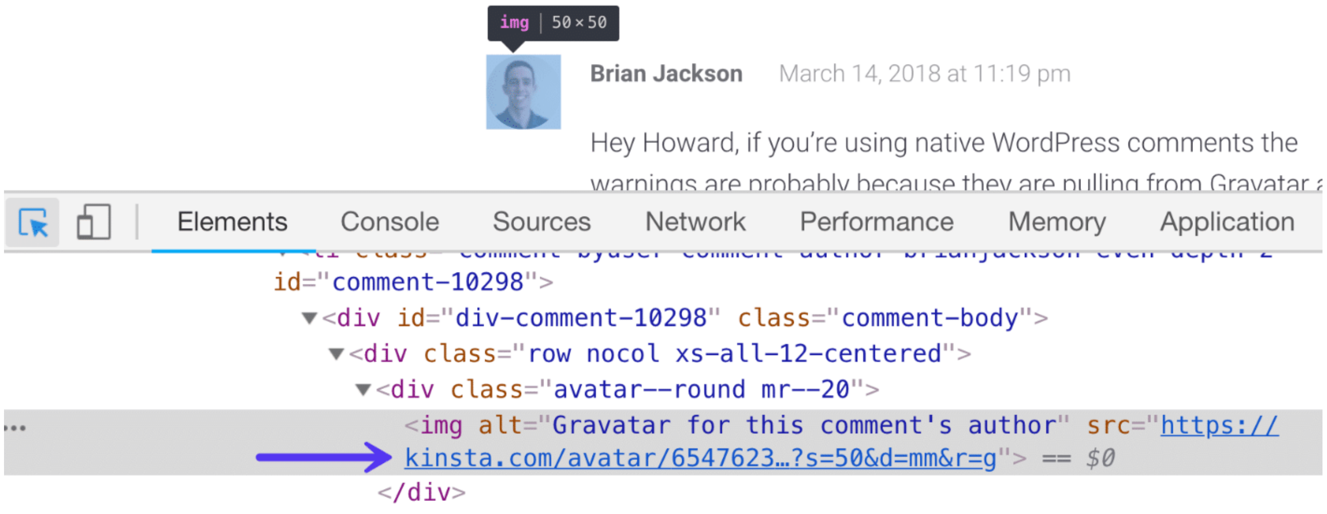
Email me whenever ☑ Anyone posts a comment
☑ A comment is held for moderation

Another option you have is to use host Gravatars on your CDN. This is the approach we take at Kinsta.

By default, when WordPress comments are loaded, every single unique Gravatar requires an HTTP request. So if a page is loaded up with comments from 50 different commenters, 50 HTTP requests will be required to download all of those

Gravatars. As you can imagine, this can impact your page speed. Not to mention the fact that we've seen the external DNS lookup to gravatar.com be slow sometimes and in some cases even timeout.

If you look at Gravatars on the Kinsta blog, you can see they are loading from Kinsta.com (including our CDN). Check out how to load gravatars from your CDN.

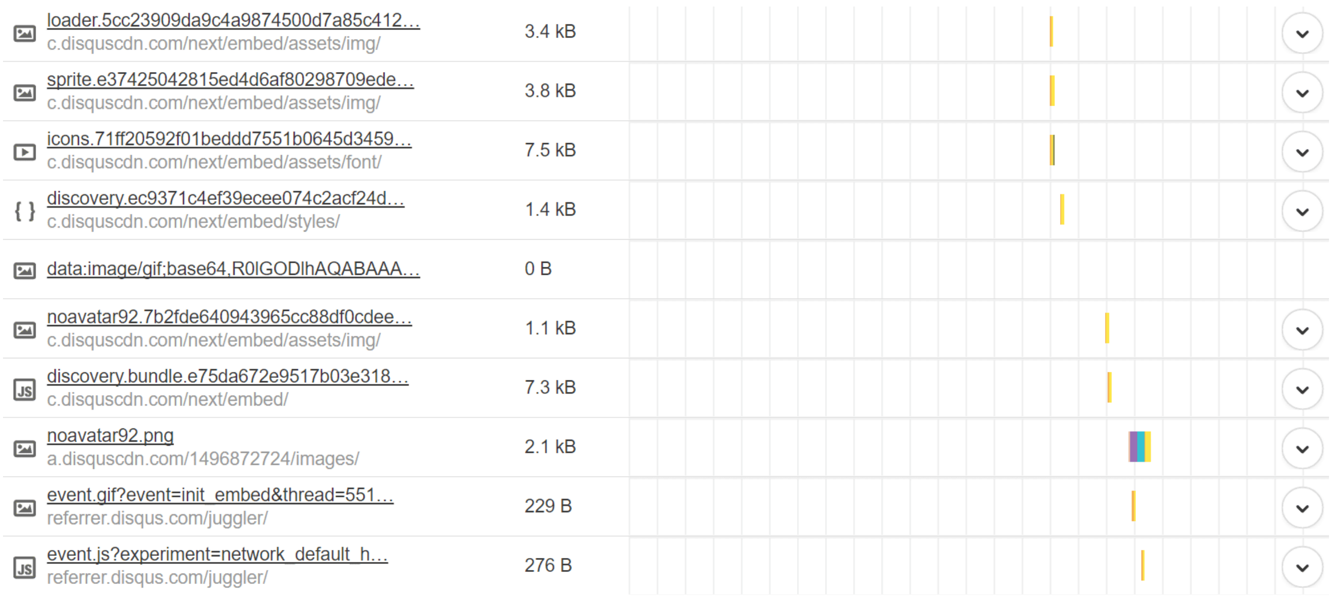


Hey Howard, if you're using native WordPress comments the warnings are probably because they are pulling from Gravatar

```
id="comment-10298">
  ▼<div id="div-comment-10298" class="comment-body">
    ▼<div class="row nocol xs-all-12-centered">
      ▼<div class="avatar--round mr--20">
          <img alt="Gravatar for this comment's author" src="https://
          kinsta.com/avatar/6547623…?s=50&d=mm&r=g"> == $0
        </div>
```

Option 3 – Use a Third-Party Comment System

Your third option is to use a third-party comment system. If your site is hosted on a cheap, resource-starved shared server, then using a third-party commenting system may speed up pages with lots of comments. It's the same idea as image optimization, offloading the work. However, if you're hosted with Kinsta or another quality web host, switching to a third-party won't do much to help your website's load speed and may slow it down.



loader.5cc23909da9c4a9874500d7a85c412… c.disquscdn.com/next/embed/assets/img/	3.4 kB
sprite.e37425042815ed4d6af80298709ede… c.disquscdn.com/next/embed/assets/img/	3.8 kB
icons.71ff20592f01beddd7551b0645d3459… c.disquscdn.com/next/embed/assets/font/	7.5 kB
discovery.ec9371c4ef39ecee074c2acf24d… c.disquscdn.com/next/embed/styles/	1.4 kB
data:image/gif;base64,R0lGODlhAQABAAA…	0 B
noavatar92.7b2fde640943965cc88df0cdee… c.disquscdn.com/next/embed/assets/img/	1.1 kB
discovery.bundle.e75da672e9517b03e318… c.disquscdn.com/next/embed/	7.3 kB
noavatar92.png a.disquscdn.com/1496872724/images/	2.1 kB
event.gif?event=init_embed&thread=551… referrer.disqus.com/juggler/	229 B
event.js?experiment=network_default_h… referrer.disqus.com/juggler/	276 B

Always make sure to speed test the third-party comment system you're trying. Take a look at all the separate requests Disqus generates (as shown below). While most of these requests are loading asynchronously, you'll still notice some additional load time if you're using Disqus.

Option 4 – Lazy Load Comments

Your fourth option is to lazy load comments so that they don't slow down the initial page rendering. Here are a couple of plugins you might want to check out:

- Lazy Load for Comments: This plugin allows you to lazy load native WordPress comments.
- Disqus Conditional Load: If you want to use the Disqus comment system, this is a must-have plugin to lazy load comments.

2:19 Disable WordPress RSS Feeds

If you're not using the blogging portion of WordPress on your site, you can disable the WordPress RSS feeds. While this won't have a huge impact on performance, everything helps. It's also one less thing you have to worry about.

Check out these two different ways to disable RSS feeds in WordPress:

- Disable RSS Feed with Plugin
- Disable RSS Feed with Code

2:20 Use MyKinsta Analytics

If you’re a Kinsta client, you can take advantage of the free performance insights we have built into our MyKinsta Analytics tool. Reports available include:

- Average PHP + MySQL response time
- PHP Throughput
- AJAX Usage
- Top Average PHP + MySQL Response Time
- Top Maximum Upstream Time
- And more…

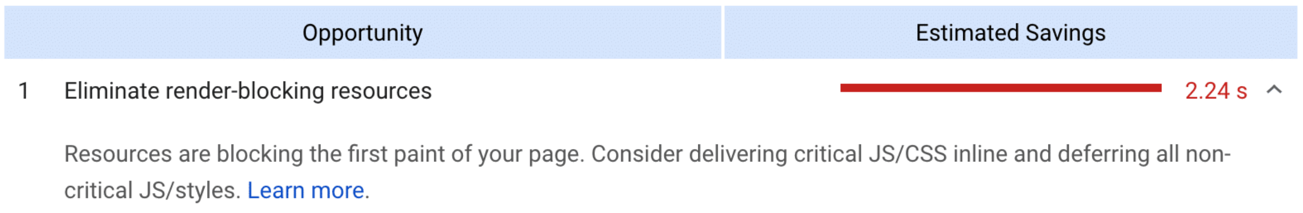
3: Frontend Optimizations

What the user sees on your website is called the frontend or presentation layer. It is controlled by HTML, CSS, and JavaScript. Fonts and graphics also come into play here in relation to site speed.

In section 8 of our Site Speed Ebook we explained minification. The steps below are ways to further optimize how CSS and JS load on your website frontend to increase speed.

3:1 Eliminate Render-Blocking JavaScript and CSS

A warning about render-blocking JavaScript and CSS might appear when you have files preventing the page from loading as fast as possible. Specific JS and CSS are sometimes conditional, meaning they aren't required to display above-the-fold content. You can prevent them from becoming render-blocking by using *async* and *defer* attributes.



Opportunity	Estimated Savings
1 Eliminate render-blocking resources	2.24 s

Resources are blocking the first paint of your page. Consider delivering critical JS/CSS inline and deferring all non-critical JS/styles. Learn more.

To eliminate render-blocking JavaScript and CSS you need to do the following:

Clear JS from the Critical Rendering Path

Moving JavaScript out of the critical rendering path is typically done by adding either the *defer* or the *async* attribute to the script HTML elements that call JavaScript resources.

- The **async attribute** tells the browser to start downloading the resource right away without slowing down HTML parsing. Once the resource is available, HTML parsing is paused so the resource can be loaded.
- The **defer attribute** tells the browser to hold off on downloading the resource until HTML parsing is complete. Once the browser has finished

with the HTML it will then download and render all deferred scripts in the order in which they appear in the document.

Optimize Delivery of CSS Resources

Optimizing the delivery of CSS essentially means you need to figure out how to make it non-render blocking.

- Identify the styles that are required to render above-the-fold content and deliver those styles inline with the HTML
- Use CSS conditionally on devices only when needed
- Load remaining CSS asynchronously

Doing all of the above can sometimes be a tricky process and definitely takes some tweaking based on the scripts you have loading on your site. Here are a couple of WordPress plugins that can help:

- Autoptimize (free)
- Async JavaScript (free)
- Hummingbird (free)

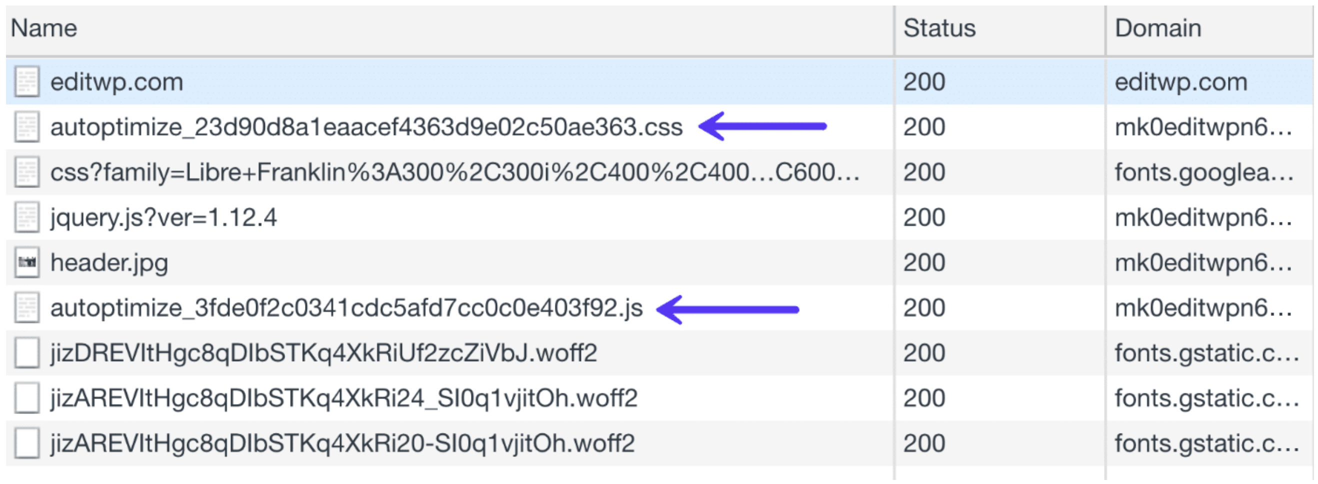
For a more detailed explanation and walk through, we recommend checking out our post on eliminating render-blocking JavaScript and CSS.

3:2 Combine External CSS and JavaScript in WordPress

The ‘combine external CSS’ warning is typically seen when using a CDN because you are hosting your CSS files on an external domain, such as cdn.domain.com. In the past, a quick way to fix this was to concatenate your CSS files, or combine them so that they are loading in a single request.

But that doesn’t necessarily mean this optimization is completely dead. In some instances, we have seen this still speed up WordPress sites. It depends on the size of the files and how many of them there are. Therefore, this is one optimization we recommend you still test on your site.

One of the easiest ways to combine your external CSS and JavaScript files is with the free Autoptimize plugin. After combining them, you will see a "autoptimize_xxxxx.css" or "autoptimize_xxxxx.js" file. It also supports loading them from your CDN. You can also do this with the WP Rocket plugin.



Name	Status	Domain
editwp.com	200	editwp.com
autoptimize_23d90d8a1eaacef4363d9e02c50ae363.css	200	mk0editwpn6...
css?family=Libre+Franklin%3A300%2C300i%2C400%2C400...C600...	200	fonts.googlea...
jquery.js?ver=1.12.4	200	mk0editwpn6...
header.jpg	200	mk0editwpn6...
autoptimize_3fde0f2c0341cdc5afd7cc0c0e403f92.js	200	mk0editwpn6...
jizDREVItHgc8qDIbSTKq4XkRiUf2zcZiVbJ.woff2	200	fonts.gstatic.c...
jizAREVItHgc8qDIbSTKq4XkRi24_SI0q1vjitOh.woff2	200	fonts.gstatic.c...
jizAREVItHgc8qDIbSTKq4XkRi20-SI0q1vjitOh.woff2	200	fonts.gstatic.c...

Check out our in-depth post on how to combine external CSS and JavaScript in WordPress.

3:3 Lazy Loading

If you have a lot of images, you might consider lazy loading them. This is an optimization technique that loads visible content but delays the downloading and rendering of content that appears below the fold.

Check out our guide on how to implement lazy loading in WordPress. This can be especially important on blog posts with lots of gravatar icons from comments. Google also just released their recommendations for lazy loading.

3:4 Additional Image Optimization Tips

Here are a few final image optimization tips to walk away with.

The days of uploading images only sized to the width of the column or DIV are over. Responsive images work out of the box in WordPress (since version 4.4) and will automatically display smaller image sizes to mobile users.

Format: Simple Vector Graphic (SVG)

SVGs can be another awesome alternative to using images. Many of the hand-drawn illustrations you see around the Kinsta website are SVGs (vectors). SVGs are typically a lot smaller in file size, although not always.

- Check out our tutorial on how to use SVGs on your WordPress site.

Format: Web Picture Format (WebP)

WebP is a newer image format created by Google. WebP files are often 25-35% smaller than JPEGs, but look just as good. One issue with this format, that is getting better over time, is that not all browsers fully support the format yet.

- Learn how to use WebP and more about browser support

Use Icon Fonts

Consider using icon fonts instead of placing text within images – they look better when scaled and take less space. And if you use a font generator, you can optimize them even more. Check out how we decreased the size of our icon fonts file by a whopping 97.59% using a font generator.

3:5 Mobile Optimizations

Google began rolling out its mobile-first index on March 26th, 2018. Previously Google's crawling, indexing, and ranking systems have used the desktop version of websites. Mobile-first indexing means that Googlebot will now use the mobile version of your WordPress site for indexing and ranking. This helps improve the search experience for mobile users.

When it comes to optimizing your site for mobile-first, **speed is one of the most important factors to focus on**. Speed plays a major role in everything from usability to bounce rates and determining whether or not potential buyers will return to your site. In fact, speed is now a landing page factor for Google Search and Ads for mobile searches.

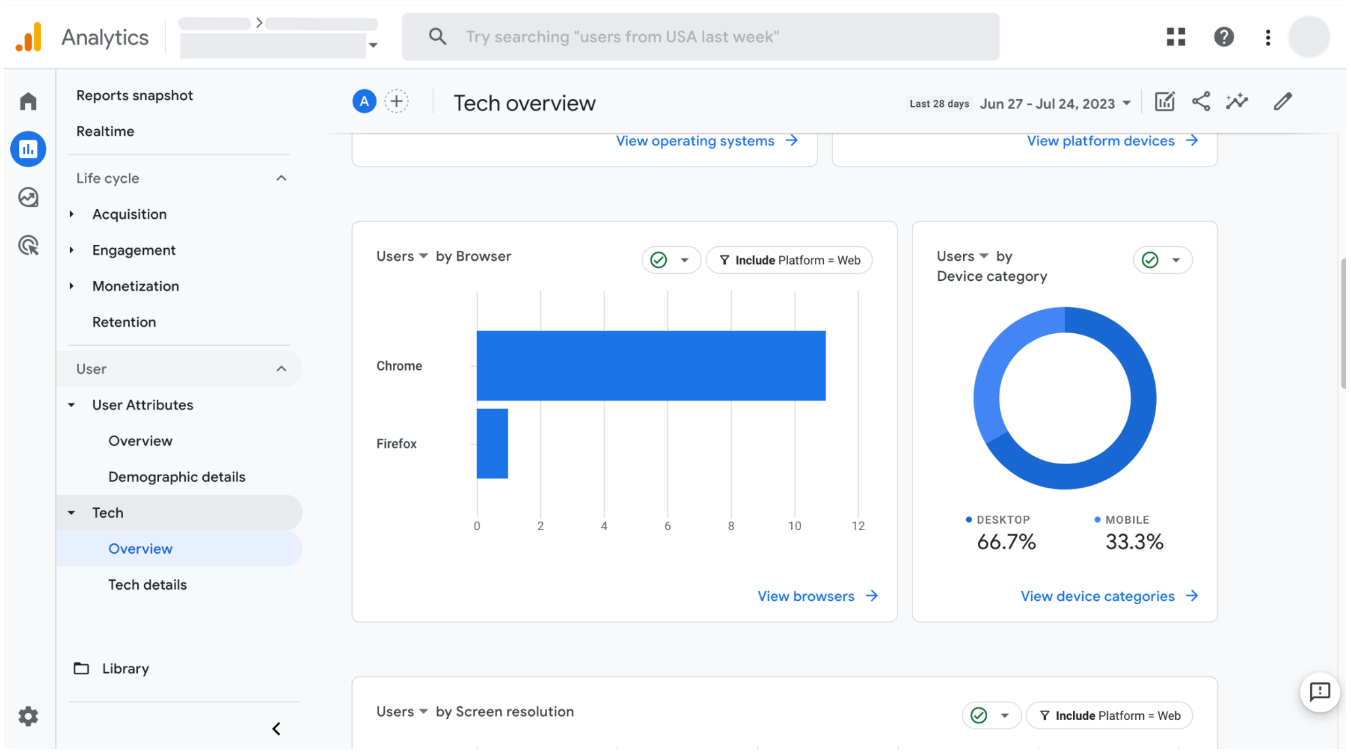
Bad mobile experiences will lead the majority of users to never return. According to the latest Google page speed report, the average time a mobile site took to load in 2018 was 15 seconds. Can you imagine waiting that long to load a single page? Astounding.

Users demand (and deserve) better. According to the same page speed report, **53% of mobile site visitors leave pages that take longer than a measly three seconds to load**.

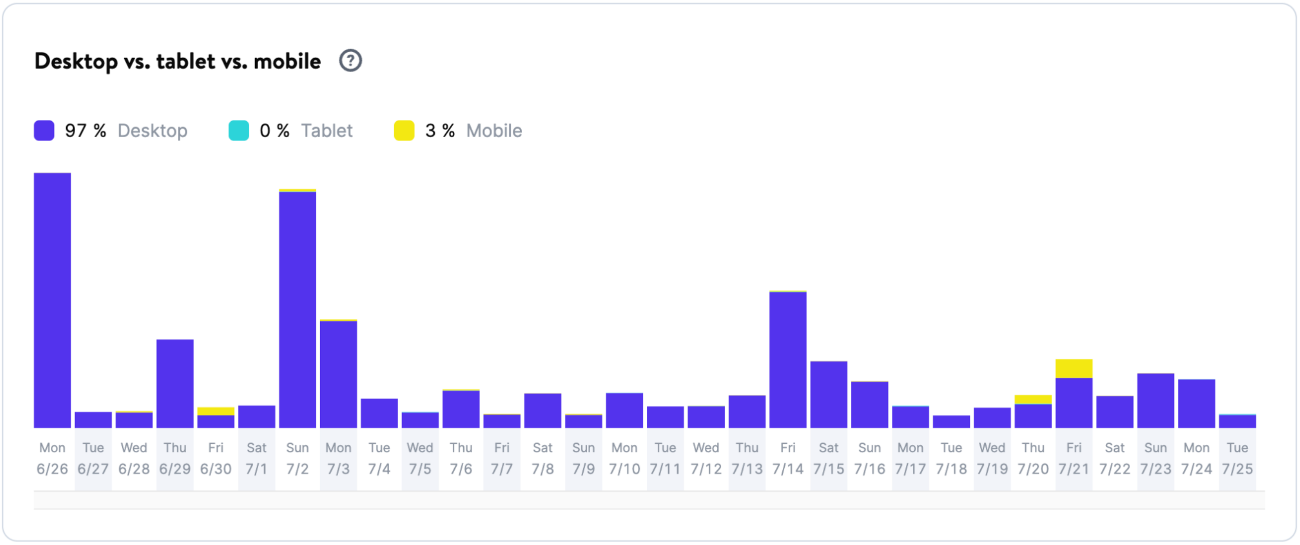
Slow mobile experiences aren't killing conversions. They're preventing you from even getting a chance to convert prospects. As page load times increase by just a few seconds, the likelihood of someone bouncing climbs exponentially. Here are a few things to consider when optimizing for mobile.

Check Out Your Mobile Traffic

It's always important to take a look at how much mobile traffic you're getting, as this might shift your priorities a bit. You can see how many mobile devices are visiting your site in Google Analytics under "User → Tech → Overview." Look for the "Users by Platform / device category" section.



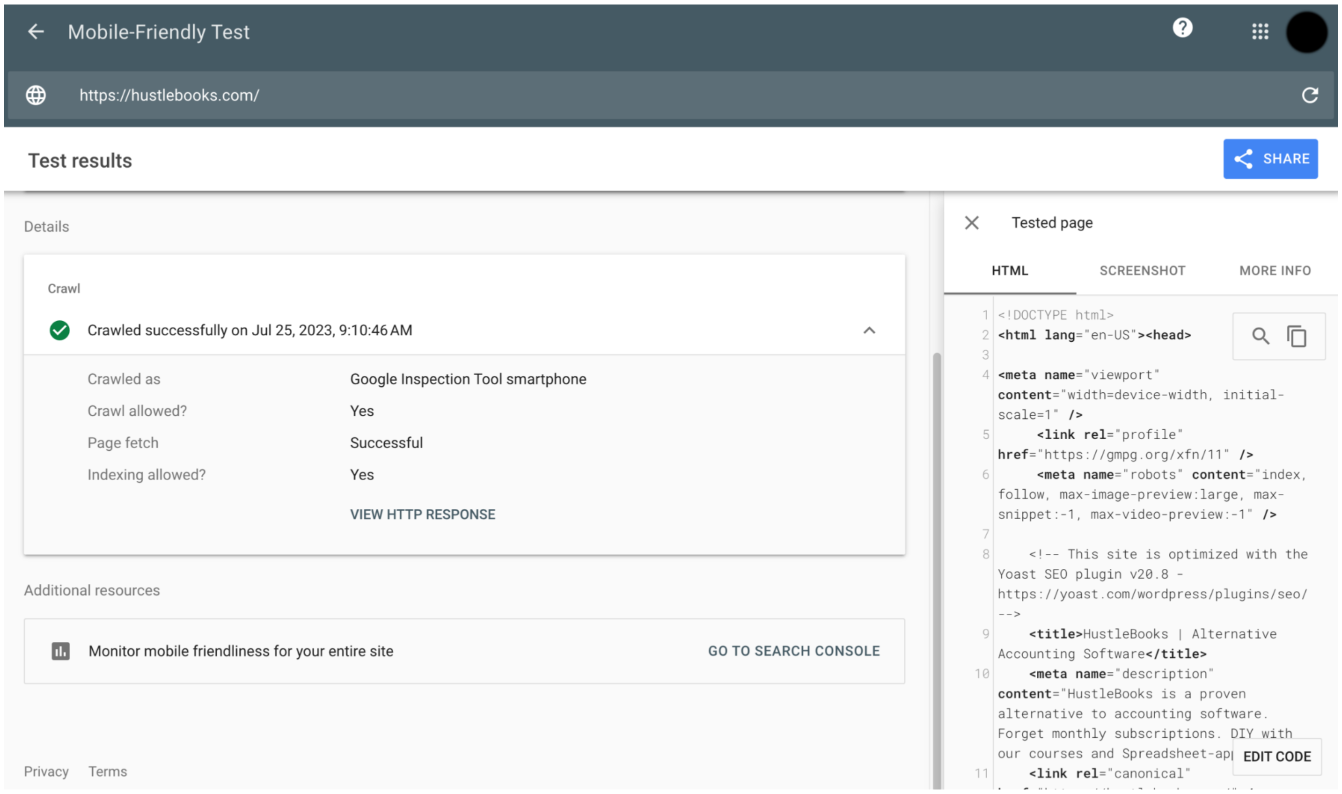
If you're a Kinsta client, you can also check out your mobile vs. desktop traffic in MyKinsta Analytics. As you can see on this site, over 88% of the traffic is from the desktop. It's always important to check and not just assume. Just because everyone says things are going mobile, doesn't always mean it is for your site. Look at the data.



Make Sure Your Site is Responsive

For many websites and businesses, more people visit your site with mobile devices (smartphones) than desktop devices (laptops and computers). So make sure you work on mobile! This means it utilizes media queries to scale things down automatically on mobile devices. If you still haven't done this, you're far behind your competition. All of the WordPress themes we mentioned earlier in this post are fully responsive and look awesome on all devices.

Use Google's Mobile-Friendly tool to test and ensure that your website passes all the requirements.



Details

Crawl

Crawled successfully on Jul 25, 2023, 9:10:46 AM

Crawled as	Google Inspection Tool smartphone
Crawl allowed?	Yes
Page fetch	Successful
Indexing allowed?	Yes

VIEW HTTP RESPONSE

Additional resources

Monitor mobile friendliness for your entire site — GO TO SEARCH CONSOLE

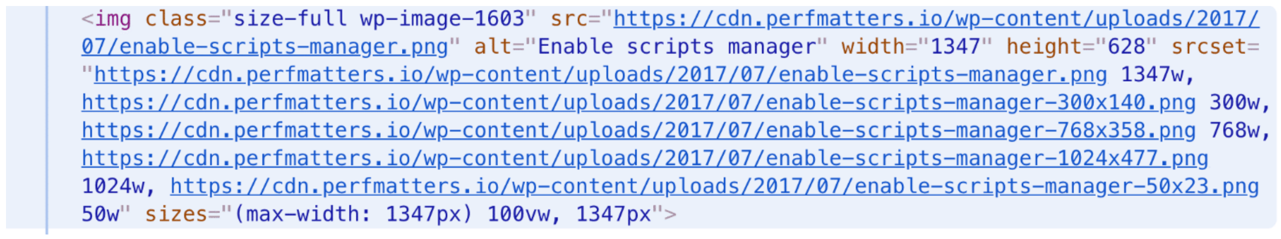
Privacy Terms

```
<!DOCTYPE html>
<html lang="en-US"><head>

<meta name="viewport"
content="width=device-width, initial-
scale=1" />
     <link rel="profile"
href="https://gmpg.org/xfn/11" />
     <meta name="robots" content="index,
follow, max-image-preview:large, max-
snippet:-1, max-video-preview:-1" />

    <!-- This site is optimized with the
Yoast SEO plugin v20.8 -
https://yoast.com/wordpress/plugins/seo/
-->
    <title>HustleBooks | Alternative
Accounting Software</title>
    <meta name="description"
content="HustleBooks is a proven
alternative to accounting software.
Forget monthly subscriptions. DIY with
our courses and Spreadsheet-ap
    <link rel="canonical"
```

Double Check to Make Sure srcset is Working

In the past, it was very important that you upload images to scale and not let CSS resize them. However, this is no longer as important since WordPress 4.4 now supports responsive images (not scaled down by CSS). WordPress automatically creates several sizes of each image uploaded to the media library.

By including the available sizes of an image into a *srcset* attribute, browsers can now choose to download the most appropriate size and ignore the others. See an example of what your code looks like below.



```
<img class="size-full wp-image-1603" src="https://cdn.perfmatters.io/wp-content/uploads/2017/
07/enable-scripts-manager.png" alt="Enable scripts manager" width="1347" height="628" srcset=
"https://cdn.perfmatters.io/wp-content/uploads/2017/07/enable-scripts-manager.png 1347w,
https://cdn.perfmatters.io/wp-content/uploads/2017/07/enable-scripts-manager-300x140.png 300w,
https://cdn.perfmatters.io/wp-content/uploads/2017/07/enable-scripts-manager-768x358.png 768w,
https://cdn.perfmatters.io/wp-content/uploads/2017/07/enable-scripts-manager-1024x477.png
1024w, https://cdn.perfmatters.io/wp-content/uploads/2017/07/enable-scripts-manager-50x23.png
50w" sizes="(max-width: 1347px) 100vw, 1347px">
```

Due to all the third-party image plugins and customizations out there, there have been a lot of times where we’ve seen this not working correctly. Therefore, it’s important to double check that your images are properly getting the *srcset* attribute added with different versions for different screen sizes. Image optimization is now important forever.

3:6 Disable Scripts on a Per Page/Post Basis

Another very powerful way to speed up WordPress is to dig through each request that is loading on your pages and posts. You’ll most likely end up finding scripts that are loading site-wide that shouldn’t be.

You can use a premium plugin like Perfmatters which has a “Script Manager” feature built-in. This allows you to disable scripts (CSS and JavaScript) on a per page/post basis, or even site-wide with a single click. Again, this plugin is developed by a team member at Kinsta.

A few examples of what this can be used for:

- The popular Contact Form 7 plugin loads itself on every page and post. You can easily disable it everywhere with one click and enable only on your contact page.
- Social media sharing plugins should only be loaded on your posts. You can easily disable it everywhere and load only on post types, or even custom post types.

- The Table of contents plugin (TOC) loads on every page and post. With the scripts manager, you can easily control where you want it loading.
- If you've upgraded to WordPress 5.0 and aren't using the Gutenberg block editor, perhaps you're still using the classic editor or another third-party editor, there are two additional front-end scripts that are added site-wide which you can disable: */wp-includes/css/dist/block-library/style.min.css* and */wp-includes/css/dist/block-library/theme.min.css*.

Why Are Some Plugins Coded This Way?

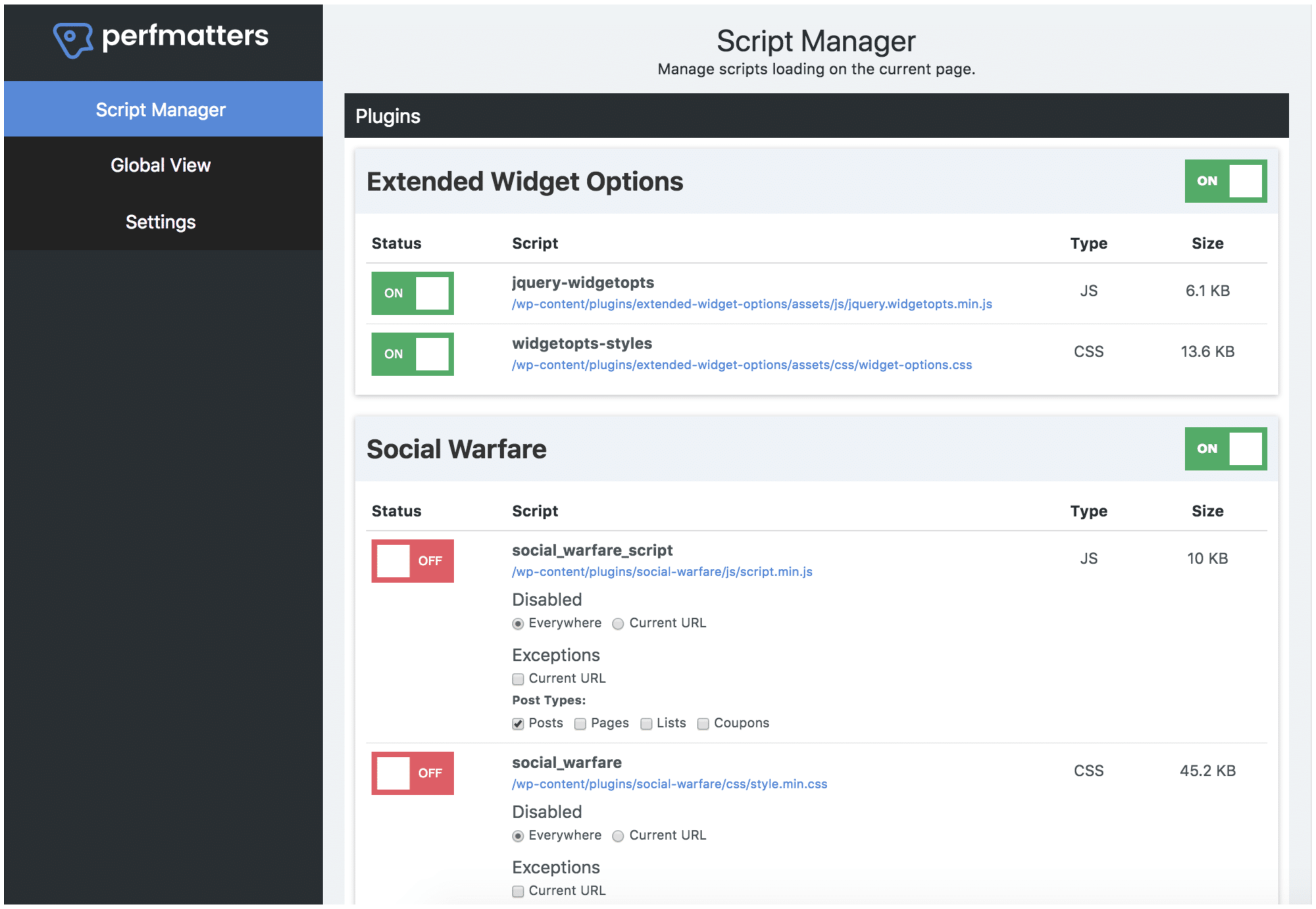
You might be wondering why all plugin developers don't just load their scripts only when the plugin is detected on the page? Well, it is a little more complicated than that. For example, if you have a plugin like Contact Form 7, it also has shortcodes which allow you to place it anywhere. This includes dropping it in a widget. With WordPress, it is much harder to query data from them when you dequeue scripts as opposed to querying data from the post or page metadata.

Therefore, a lot of times this is due to usability issues. The less chance they have for a plugin to break, the fewer tickets and support they will have. However, with a lot of plugins on the marketplace, there are ways to get around this and code for performance if they wanted to. Unfortunately, sometimes the sheer number of downloads and users makes coding for usability a priority.

Touring the Script Manager

We'll give you a little tour of the Script Manager. After clicking it in your toolbar you will be presented with all the scripts loading on that current URL, both JavaScript and CSS files. You then have the following options:

1. **Status On** (default setting)
2. **Status Off:** Disable Everywhere (you can then choose which posts types you want it enabled on, along with the current URL)
3. **Status Off:** Disable only on current URL (this is very useful for using on your homepage)
4. **Status Off:** Exceptions (current URL, post type, or archive)



Script Manager

Manage scripts loading on the current page.

Plugins

Extended Widget Options ON

Status	Script	Type	Size
ON	jquery-widgetopts /wp-content/plugins/extended-widget-options/assets/js/jquery.widgetopts.min.js	JS	6.1 KB
ON	widgetopts-styles /wp-content/plugins/extended-widget-options/assets/css/widget-options.css	CSS	13.6 KB

Social Warfare ON

Status	Script	Type	Size
OFF	social_warfare_script /wp-content/plugins/social-warfare/js/script.min.js Disabled [● Everywhere] ○ Current URL Exceptions ☐ Current URL Post Types: [✓ Posts] ☐ Pages ☐ Lists ☐ Coupons	JS	10 KB
OFF	social_warfare /wp-content/plugins/social-warfare/css/style.min.css Disabled [● Everywhere] ○ Current URL Exceptions ☐ Current URL	CSS	45.2 KB

Everything is **grouped together by the plugin or theme name**. This makes it super easy to disable an entire plugin at once. Typically a WordPress plugin will have both a JavaScript and CSS file. A WordPress theme might have 10+ files. The Script Manager even supports regex for when you have a more complicated URL structure or installation.

After you select and or modify the settings, make sure to hit "Save" at the bottom. You can then test in a website speed tool to ensure the scripts are no longer loading on the page or post. Make sure to clear your cache first! And if anything goes wrong on your site visually, you can always re-enable it in the settings to return to normal.

In a speed test we were able to **decrease the total load times by 20.2%**. On their homepage alone they were able to reduce the number of HTTP requests from 46 down to 30. Their page size also shrunk from 506.3 KB to 451.6 KB.

For other ways to disable scripts, check out our blog post on how to disable WordPress plugins from loading.

3:7 Analyzing Third-Party Performance

Basically, anything you call externally from your site has load time consequences. What makes this problem even worse is that some of them are only slow intermittently, making identification of the issue even more difficult.

A third-party external service could be considered anything that communicates with your WordPress site from outside your own server. Here are a few common examples we encounter on a regular basis:

- Social media platforms like Twitter, Facebook, and Instagram (widgets or conversion pixels)
- 3rd-party advertising networks like Google Adsense, Media.net, BuySellAds, Amazon Associates
- Website analytics and tracking scripts like Google Analytics, Crazy Egg, Hotjar, AdRoll
- A/B testing tools such as Optimizely, VWO, Unbounce
- WordPress comment systems such as Disqus, Jetpack, Facebook comments
- Backup and security tools such as VaultPress, Sucuri, CodeGuard
- Social sharing tools such as SumoMe, HelloBar
- CDN networks like KeyCDN, Amazon CloudFront, CDN77, and StackPath
- Externally hosted Javascript

How much do some of these third-party trackers impact performance? In our own case study, we saw that third-party scripts **increased the page load times by 86.08%**.

You have to be very careful on your WordPress site. Just one bad third-party API call could timeout your entire site! Yes, it shouldn’t work that way, but in a lot of cases, it does. We’ve seen it more times than we can count.

It’s important that whenever you add a new feature or plugin to your site that you investigate the external resources loading from it. The less the better!

- Read this article about using an Application Performance Monitoring (APM) tool to analyze site performance

Appendix: Your Site Could Be Slow Because It’s Been Quietly Hacked

If you’re having trouble tracking down a performance issue, it could very well be that your site is hacked, infected with malware, or undergoing a DDoS attack. This can impact your site’s speed and even the responsiveness of your WordPress admin dashboard. In these cases we recommend the following:

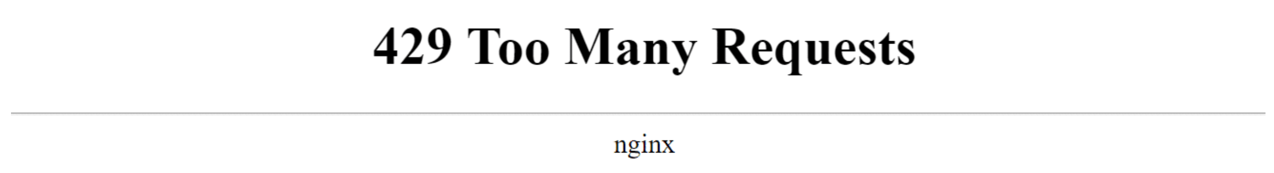
1. Implement a proxy server and WAF such as Cloudflare or Sucuri.
2. Block bad IP addresses using the services above (or if you’re a Kinsta client you can easily block IP addresses from your MyKinsta dashboard).
3. You can also implement geo-blocking. Some countries are really bad when it comes to the quality of the traffic they generate. If you’re under attack, you might need to block the entire country, either temporarily or permanently. Ask your web host if and how you can implement this on your site (here’s how it works on Kinsta).

Troubleshooting with Error Codes (HTTP Status Codes)

HTTP status codes are like a short note from the web server that gets tacked onto the top of a web page. It’s not part of the web page. Instead, it’s a message from the server letting you know how things went when the request to view the page was received by the server. These can be invaluable when it comes to troubleshooting!

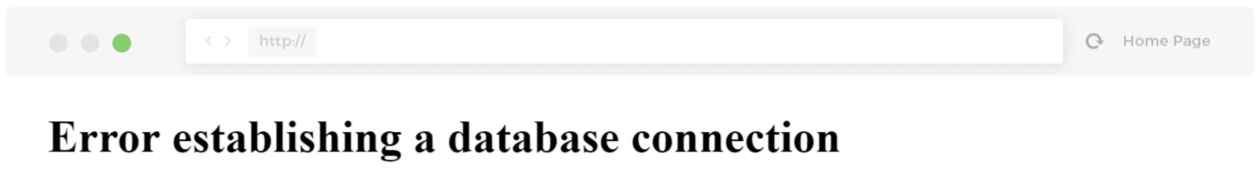
While there are over 40 different status codes, below are the common ones we see WordPress users struggling with.

429: “Too many requests.” Generated by the server when the user has sent too many requests in a given amount of time (rate limiting). This can sometimes occur from bots or scripts attempting to access your site. In this case, you might want to try changing your WordPress login URL.


429 Too Many Requests

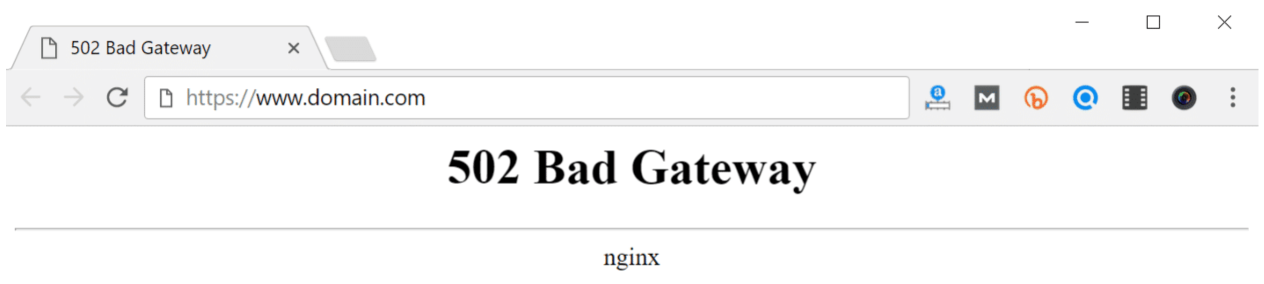
nginx

500: “There was an error on the server and the request could not be completed.” A generic code that simply means “internal server error”. Something went wrong on the server, and the requested resource was not delivered. This code is typically generated by third-party plugins, faulty PHP, or even the connection to the database breaking. Check out our tutorials on how to [fix the error establishing a database connection]() and other ways to resolve a [500 internal server error]().



Error establishing a database connection

502: “Bad Gateway.” This error code typically means that one server has received an invalid response from another. Sometimes a query or request will take too long, and so it is canceled or killed by the server and the connection to the database breaks. Check out our in-depth tutorial on how to fix the [502 Bad Gateway error]().

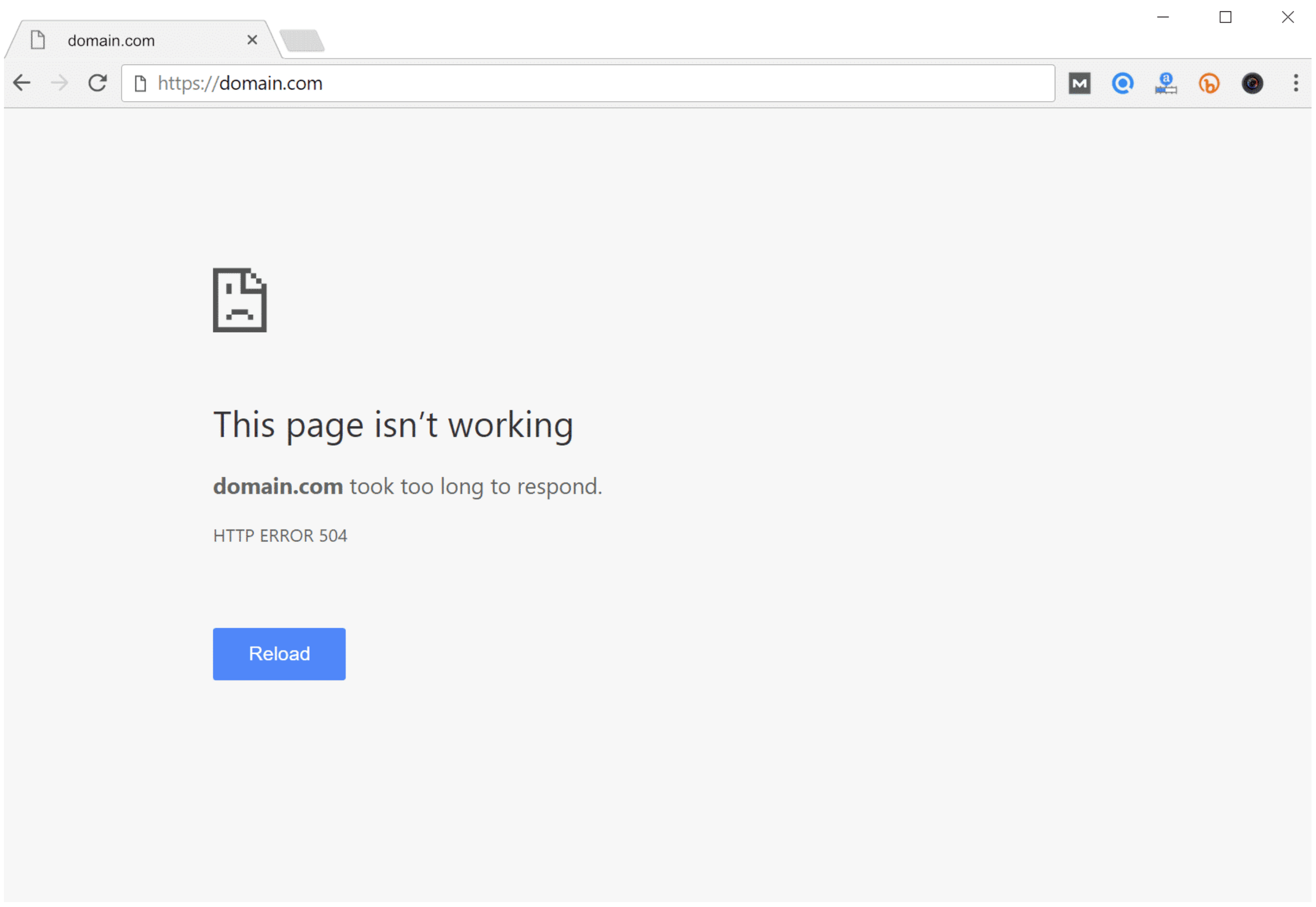


502 Bad Gateway

nginx

503: “The server is unavailable to handle this request right now.” The request cannot be completed right now. This code may be returned by an overloaded server that is unable to handle additional requests.

504: “The server, acting as a gateway, timed out waiting for another server to respond.” The code returns when there are two servers involved in processing a request, and the first server times out waiting for the second server to respond. Read more about how to fix 504 errors.



domain.com

https://domain.com

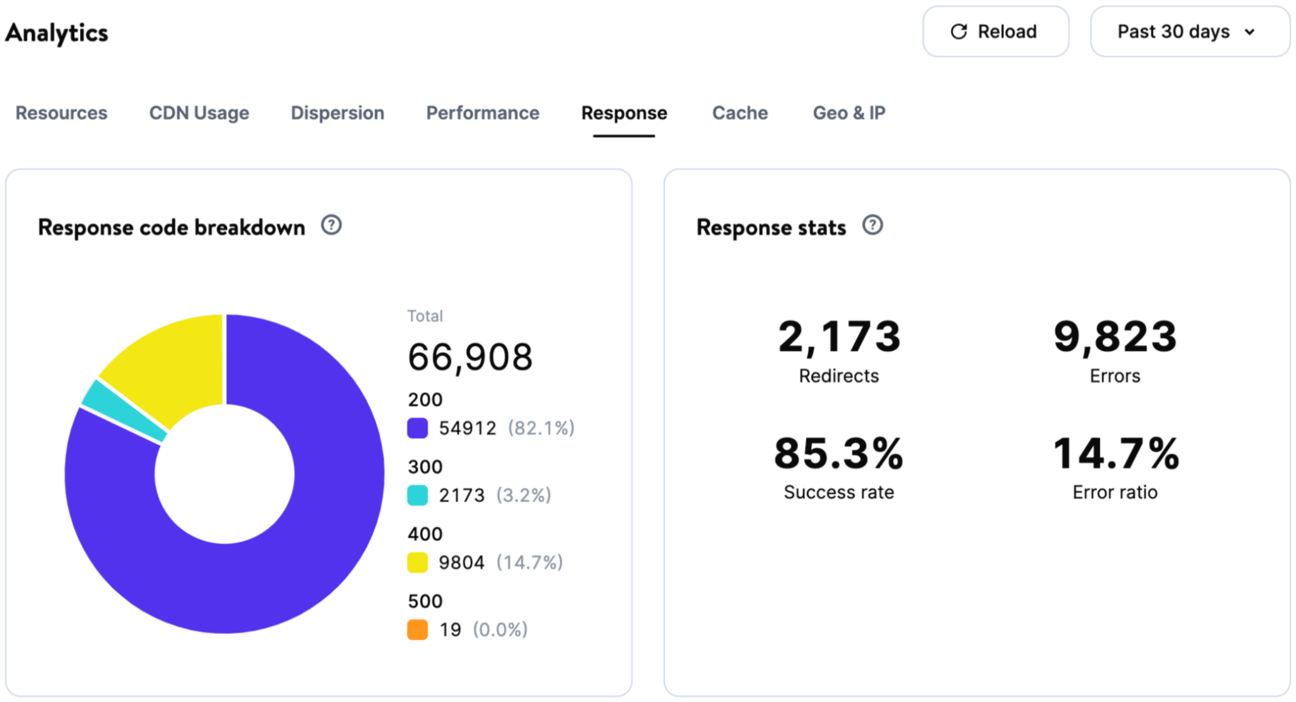
This page isn’t working

domain.com took too long to respond.

HTTP ERROR 504

You can also dig into these HTTP response codes in our MyKinsta Analytics tool. Our response code breakdown report lets you see an overview of the distribution of HTTP status codes served for the requested resources.

The response stats report lets you see the total number of redirects happening, errors, success rate, and error ratio. Every WordPress site will typically have a small error rate ratio; this is completely normal. If it’s above one percent, look into why.



Response code breakdown

Total
66,908

200
🟦 54912 (82.1%)

300
🟦 2173 (3.2%)

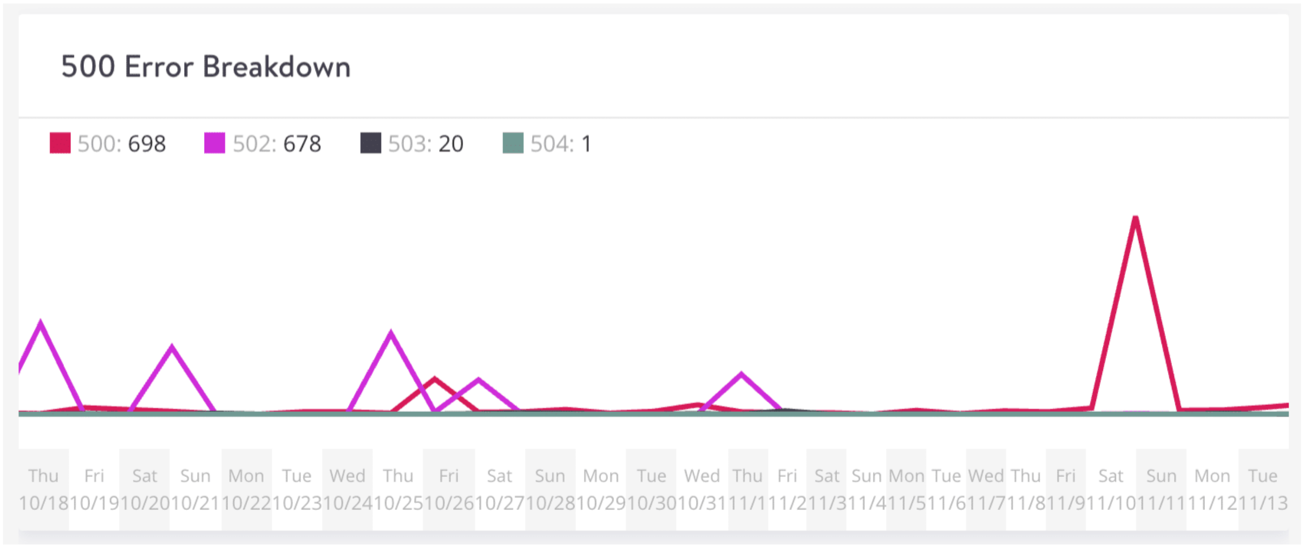
400
🟨 9804 (14.7%)

500
🟧 19 (0.0%)

Response stats

2,173 Redirects	9,823 Errors
85.3% Success rate	14.7% Error ratio

There are then breakdown reports for each type of error code, such as 500 errors, 400 errors, redirects, etc.


500 Error Breakdown

🟥 500: 698 🟪 502: 678 ⬛ 503: 20 🟩 504: 1

Conclusion

Thanks for checking out our Site Speed White Paper. As you can probably tell, we are obsessed with all the different ways you can speed up WordPress sites.

We hope these tips, how-tos, and explanations help you get your site running at peak speed.

- Be sure to **bookmark our Website Speed resource page** where we list new resources related to speed and performance each month.
- Get help from our experts! **Get a FREE performance audit** and get a second opinion on your website's speed and performance.
- Make your life easier with our Managed WordPress Hosting. You can focus on growing your business and we'll handle making your site run fast. We include and manage most of the optimizations described in this White Paper with each of our plans. **Migrate for free** and see how fast your site runs. It's risk free with a 30-day money back guarantee.
