Best PHP Performance Tips From Community & Influencers

Updated on September 25, 2018

13 Min Read PHP (HTTPS://WWW.CLOUDWAYS.COM/BLOG/PHP/)

Getting your web application to the highest performance level is quite impossible. You need to apply different tactics at different points according to the nature of application.

PHP has faced a lot of criticism when it comes to performance benchmarking. But thanks to the contributors who gave the old language wings and introduced latest PHP 7.x versions, which is equipped with all the necessary tools and components required for PHP performance boost. Using PHP 7, you can build & deploy advanced apps on any hosting for PHP with such features and functionalities that could optimize its performance to higher levels.

In my opinion, there are few simple tips you can use in your code for making your application perform well. Like, you can take advantage of native functions, use JSON except XML, use caching systems, configure opcache, PHP-FPM, Memcached properly. You can also close db connection and limit the db hits.

For asset management you can add CDNs for fast content delivery. Integrate HTTP2/SSL certificates for better security optimization and can also read various other PHP security tips here. While if you are using PHP frameworks like Laravel and Symfony, you must enable profilers in them to track what's going wrong in code.

You Might Also Like: Tips for Laravel Performance Optimization

Enable OPcache on PHP server

OPcache is one of the building block element of PHP performance because it works directly with the code compiling process. Imagine if you are creating a request to the server and it is compiling the code every time and then sending you responses, the practice will eventually make your loading time slower. Thanks to opcache which will cache the pre-compile code and does not let it compile on later requests.

Conducting the PHP performance test after integrating Opcache, some experts claim that PHP performance will be 3x faster and the load time will reduce much impressively. So tracking these grounds, the importance of Opcache cannot be ignored.

OpCache is compiled by default on PHP5.5+. However it is disabled by default. In order to start using OpCache in PHP5.5+ you will first have to enable it. To do this you would have to do the following.

```
Add the following line to your php.ini:
zend_extension=/full/path/to/opcache.so (nix)
zend_extension=C:\path\to\php_opcache.dll (win)
```

Note that when the path contains spaces, you should wrap it in quotes:

```
zend_extension="C:\Program Files\PHP5.5\ext\php_opcache.dll"
```

Also note that you will have to use the zend_extension directive instead of the “normal” extension directive because it affects the actual Zend engine (i.e. the thing that runs PHP).

Cache and Minify Static Assets

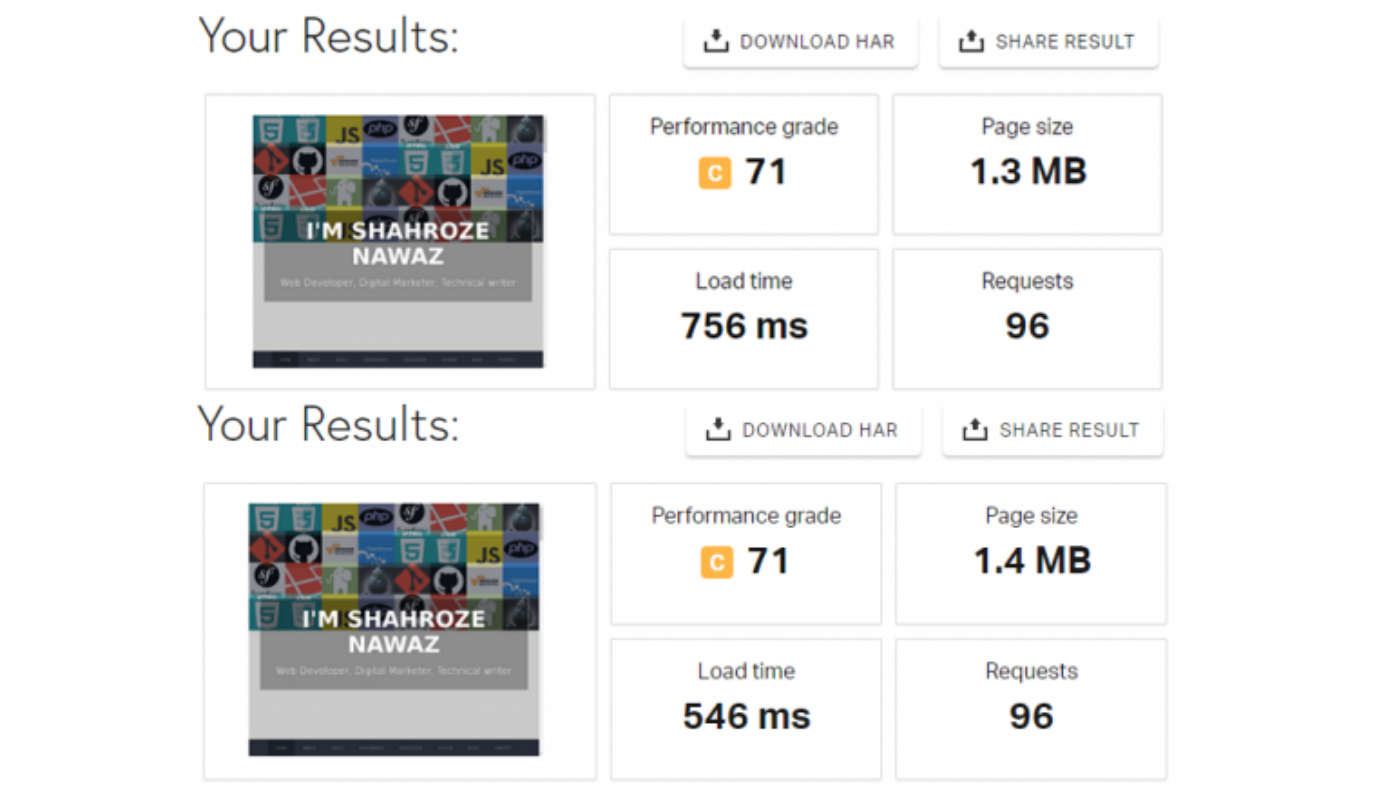
Working with PHP doesn’t means you are working with a sole tool. Instead, you will be working with lot of other tools by side including HTML, CSS, JS and other scripts, as they bring significant improvement in performance of the apps. Therefore, I always recommend to minify the static scripts so that the processing time can be decreased. You do not need to do this manually every time, as you can find lot of online tools to ease this job for you:

https://www.minifier.org/
https://javascript-minifier.com/
https://jscompress.com/
https://unminify.com/

What’s minify?

TWEET IT NOW

I just minimized the HTML and CSS of my PHP website and after doing the PHP performance test here is the result of load time lowering down from 756 ms to 546 ms. It can also low down more if i will minify all the scripts.



The next thing you must apply is caching systems like Varnish for static content. Varnish can also speedup the process of PHP websites and will give you optimum performance benefits because it caches HTML. It gives you the ability to cache components for certain time and then purge it all. Custom URLs which are helpful can also be cached when you create ecommerce shops in PHP.

Configure Memcache For Database

Memcached is a distributed memory caching system. It speeds up websites having large dynamic databases by storing database object in dynamic memory. The purpose of storing in a dynamic memory is that it reduces the pressure on a server whenever an external data source requests a read. A memcached layer reduces the number of times database requests are made. Memcached stores the values (v) with the key (k) and retrieves the values (v) with the key (k) without even parsing the database queries and stays away from all these hassles.

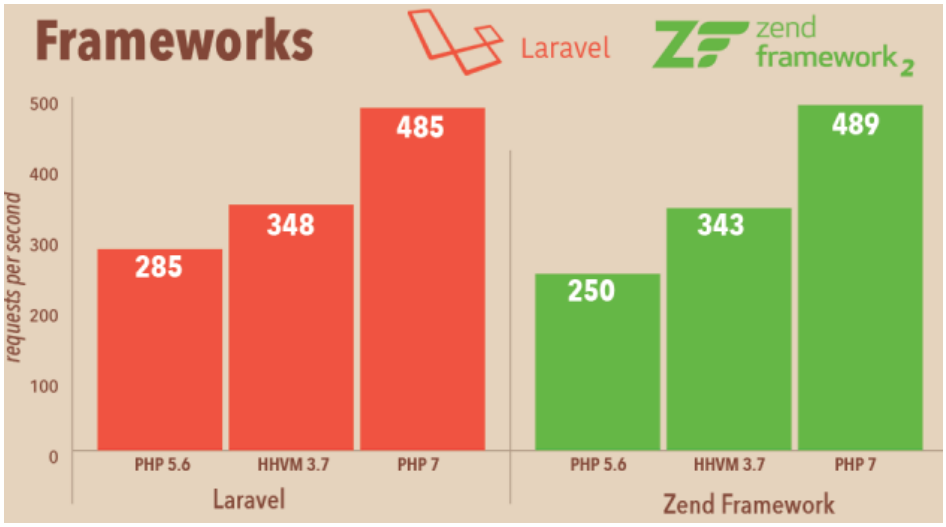
You can easily install memcache by running the following Sudo command:
sudo apt-get install memcached
After then just echo phpinfo() function and see if it is running.

But if you are a cloudways customer you don't need to do this. All you need to do is go to the manage services section and enable it by toggle button. Now you can start consuming memcache in your code for caching DB queries.

Update PHP Version

Since PHP 7.x versions are launched, it is recommended by many experts to upgrade your app to PHP7.x version. While now we already have PHP 7.3 RC1 available and you can see what's new in the version. Contributors are working hard to optimize PHP performance in each version and introducing updates for security and better performance.

If your application is still stuck at PHP 5.6 or much lower version, update that ASAP to PHP 7.x. According to Zend resource CMS like Magento runs 3x faster and Drupal 8 runs 75% faster. One WordPress request on PHP 5.6 executes under 100M cpu instructions, while PHP7 only executes it under 25M cpu instructions. When the PHP performance test conducted on the Laravel and Zend framework, we obtained following results:



So this is now a established fact that you must upgrade to newest version of PHP without breaking your application. You can get help from core contributors on GitHub also.

Improve Code for better Process

Code improvement is another important thing for performance optimization. You must avoid writing messy code and should try to optimize it by using more native functions. Utilize JSON data types instead of XML, never repeat variable declarations, always use isset() as it is much more faster.

Do not make your controllers fat and model thins, always write business logics in controllers and let models handle the DB things. Avoid redundant functions and classes and try to write standard class and inherit it in others.
Always disable debugging options on live servers, close the DB connections and avoid writing SQL queries in loops like foreach(). Also limit your DB hits and do not make so many requests at once.

You Might Also Like: Extensive Guide to MySQL Performance Tuning

Deploy PHP on Cloud Hosting

After completing development on dev servers, the next step is to deploy PHP apps on to the hosting servers. Hosting also plays a vital role in speed and performance of PHP applications. If your server is powerful and highly optimized then your requests will be processed fastly. Normally, I don't recommend Shared hosting servers for PHP applications when you have fully functional Cloud servers like Vultr, linode and Digitalocean servers available for just $. Honestly it can be a nightmare for you at any time. Also, if you don't know how to create LAMP/LEMP stacks on cloud servers then don't worry, because at Cloudways you can launch servers and applications on these server within few clicks. Just select custom PHP app from the dropdown and launch it.

The most important thing is that all the services which I defined above are available on single clicks and you don't require any manual configuration of it. Whether it is, GIT, Composer, Apache, Nginx, Memcache, Opcache, Elasticsearch, Varnish cache, PHP versions etc.

PHP Performance Tuning Tips From The Community

In order to clarify the issue to some extent, I myself, held an interactive session on the social media with the community to know what they think about PHP performance and benchmark level. I developed questions on Reddit and other hosting fora to get interesting replies from the large PHP community. Meanwhile, I received answers from known developers on Twitter too.

Here's what different developers on Twitter had to say on this topic:

Zeev Suraski, a known PHP developer and co-founder of Zend Technologies replied to the question emphasizing on few things, including updates for potential performance gains:

Zeev Suraski (https://twitter.com/zeevs)

> Take a look at Z-Ray. It's available for free within Homestead. Upgrade to 7.2 and to 7.3 when it comes out – substantial performance gains there too. And of course, cache whenever you possibly can so that the speed of your code becomes irrelevant.

Larry Garfield highlighted the use of Blackfireio for code profiling. He further emphasized on migrating to PHP 7 for optimized performance:

Larry Garfield (https://twitter.com/Crell)

> 1) Don't fear objects. They're surprisingly efficient:https://steemit.com/php/@crell/php-use-associative-arrays-basically-never
>
> 2) Use @blackfireio or similar to profile your code and work on the parts that are actually slow, not just theoretically slow.
>
> 3) Ignore the haters that say PHP is slow. PHP 7 is crazy fast.

Marco Pivetta replied with a very useful tweet. As he highlighted how to avoid different fancy stuff to enhance app performance:

Marco Pivetta (https://twitter.com/Ocramius)

> Correct architecture over fancy tech.
> Also: avoid I/O (and its side-effects) whenever it can be avoided.

Uncle Cal gave his opinion with a contrasting view. To use some other language for real-time applications. But classified PHP as the best choice for modern fast web apps.

Uncle Cal (https://twitter.com/CalEvans)

> Except that PHP has progressively gotten faster in the past 5 years. Yeah, if you are writing real-time sensor software, use something else. If you are looking for fast web, modern PHP is an excellent choice.

I too support Uncle Cal's view that PHP stands as the best choice for building faster web apps. But it should not be overestimated for developing real-time sensitive applications.

Another user Mike Pearson responded with some useful PHP performance tips to optimize overall app performance:

Mike Pearson (https://twitter.com/mike_pearson)

> Use native PHP functions where possible, instead of writing your own. Be aware of constraints (DB connections, API calls, IO, etc) and minimize the calls to those. Prefer array functions (map, filter, walk, etc) to foreach loops.

Byron Nagi replied with a question itself that indeed looked as a useful thought:

Byron Nagi (https://twitter.com/Nilithus)

> Just curious where the “prefer array functions” to “foreach” loops comes from? I thought the
> array functions were technically slower because you are invoking a function for every item in the array? I mean either of those is a micro optimization anyway but still just curious.

Mike Pearson again replied to this question with the following answer:

MP (https://twitter.com/mike_pearson)

> Good question. The array functions are a microptimization, but they typically run faster than foreach loops, because more of the process involves compiled code (native PHP) than interpreted code (PHP you write), but your mileage may vary.

Symfony enthusiast, Beno!t contributed his thoughts like a professional developer:

Beno!t (https://twitter.com/bpolaszek)

> Use PhpStorm with PHP EA plug-in. It will highlight anything that could be improved
> (use $data[] = ‘foo’ ; instead of array_push(), etc) + my own pro-tip: when instantiating objects
> within a big loop, use a factory which clones a prototype of the object instead of new MyObject(); (66% faster).

Peter Iglio recommended using Swoole extension. As it helps optimizing apps speed:

Pietro Iglio (https://twitter.com/dylandog68)

> Consider using the swoole extension. Some applications can run 30 times faster. Be sure you understand how it works.

Sylvester Ho shared his thoughts about using Cache for reducing load on DB queries. As these few little things works impressively for apps regardless of their infrastructure.

Sylvestre Ho (https://twitter.com/sylvestreho)

> Use cache wherever possible in order to reduce the amount of DB queries. Optimize the DB queries too, these are often the performance bottleneck of any web application regardless of the stacks / frameworks that are used.

Well, I too personally recommend this technique for reducing app’s dependence on DB queries. It is because of the fact that cache has the functionality to load app pages faster. It lessens the database communication and makes data retrieval faster.

Vladinator defined his solution in a quite smart way. He proposed avoiding DB queries dependence and looping of Entity manager:

Vladinator (https://twitter.com/kalessil)

> Avoid extensive memory usage: allocation is expensive, GC is very expensive. Avoid DB queries and Entity manager flushes in loop 😉

Aurora came up with a little different view. She particularly emphasized on installing and configuring PHP servers perfectly for an application:

Aurora Eos Rose (https://twitter.com/auroraeosrose)

> Install and configure it right. Turn on and tune the opcache. Keep it up to date. All the good code in the world isn't going to fix an improperly set up server.

In my opinion, her thought also holds the highest value. Because you just cannot optimize your app speed until you have a rightly configured web server.

A very well-known Magento developer, Raj replied with a remarkable tweet. He explained the usage of Varnish and Redis for app optimization:

Magento Guru (Raj) (https://twitter.com/magepsycho)

> No one mentioned #varnish (https://twitter.com/hashtag/varnish?src=hash) for full page caching & #redis (https://twitter.com/hashtag/redis?src=hash) for session caching but it's the best choice IMO.

Hassan Juniedi responded with his SQL queries and caching solutions:

Hassan Juniedi (https://twitter.com/hassanjuniedi)

> -Replace joins with multiple sql queries when tables are too big.
> -Use php-ppm for caching.

Joshua expressed his views in a quite definitive manner. He explained the question in detail. He highlighted firmly on the usage of PHP 7.x and Blackfireio for optimizing apps' speed.

Joshua Ray Copeland (https://twitter.com/OGProgrammer)

> We've separated local vs remote cache items to save on network round trip and for how much we use memcahce, results were good but you gotta be careful not to put values that need to be shared and atomic. Make sure opcode cache is on. Use php7+. Use black fire or cachegrids to opt.

Fraser Read gave a different view about keeping a check on ORMs. He emphasized on using them precisely to avoid extra and inefficient DB queries:

Fraser Reed (https://twitter.com/fraser_reed)

> Be careful with ORMs and eager loading objects. Both are very useful in the right scenarios but can also make a lot of unnecessary (and

> potentially very inefficient) database queries. Make sure you understand what the ORM is doing.

Apart from Twitter, I also decided to post the question on Reddit. So that I could get answers from the wider community of PHP developers. Here are some interesting replies I received from Redditors about PHP performance tuning.

mYkon123 replied the question highlighting the importance of apps and their main bottlenecks. They can range on many issues, but finding and fixing them is quite essential for app performance.

mYkon123 (https://www.reddit.com/user/mYkon123)

> Investing time for finding the actual bottleneck and not start caching/improve anything
> Anywhere. If the bottleneck is found, a good solution can be found. Usually something like: caching, background-tasks, queue, reorganize workflow, database indices, load-balancer, upscaling vertical/horizontal, outsourcing to micro-services with another language, etc.

A highly active user with major PHP threads on Reddit, AllenJB83 gave his view about Database queries. He emphasized on taking DB queries into consideration for fixing performance issues.

AllenJB83 (https://www.reddit.com/user/AllenJB83)

> Don't just look at PHP code. SQL (or whatever your database uses) is code too. Database/ query tuning is far too often ignored and is often a big source of performance issues.Recommended resources for MySQL:High Performance MySQL Percona Monitoring Management. For performance monitoring, tools like NewRelic APM really help – yes they cost money but it's well worth it to be able to find bottlenecks and quickly check on performance on a regular basis.

I too believe that handling DB queries smartly can result in better performance for apps. As it is one of the major issues different applications have in their run-time execution.

Another Reddit user, Firehed proposed a little but smart solution to improve PHP performance in his answer:

Firehed (https://www.reddit.com/user/Firehed)

> Additionally, log queries during development and fix N+1 query problems before they go live. It's usually a small tweak with huge rewards.

While BlueScreenJunky explained the problem in quite detail. As how the ping drop between the staging server and the production DB can cause incompressible delays for every query.

BlueScreenJunky (https://www.reddit.com/user/BlueScreenJunky)

> fix N+1 query problems before they go live. We currently have an (unintentional) safeguard that works pretty well : Our staging and

> production DBs are in the same data center along with the rest of our machines, but our staging server is on the other side of the country, which means there's a ~10ms ping between the staging PHP server and the DB, so an incompressible delay of 10ms for every query... The downside is that some.pages are really slow even if there's nothing wrong with our code, but the upside is that we instantly notice n+1 problems.

Beatniak explained his answer in detail and underlined several points in his reply. He particularly highlighted the importance of checking logs for PHP performance monitoring. As it enables you to know the actual problem causing the performance issue.

Beatniak (https://www.reddit.com/user/beatniak)

> Don't optimize before knowing what is slowing down your application. Log everything in development. When trying to improve performance in production, log stuff like mysql slow queries in production for a few hours.
> Log what costs the most time in your application with things like https://underground.works/clockwork/ (https://underground.works/clockwork/) You can log that in production just for your team:if ({requestIP} === {officeIP} && Request.has('debug')) Log.this.stuff()Improve the low hanging fruit. Most often those are database queries with non-indexed
> joins. Fix those problems with tips from https://use-the-index-luke.com/ (https://use-the-index-luke.com/)Because you can see the timing cascade (like the network tab in the developer tab of Chrome) of your application, you know what the bottlenecks are. And thus, what to spend time on to make your application faster.

twisted 1919 gave his answer in a little contrasting manner. He shared his experience of developing a simple project and analyzing its performance:

twisted1919 (https://www.reddit.com/user/twisted1919)

> Maybe this is a bit off-topic, but these days we had to create a small project which
> should display some items we have for sale which would be fetched from a remote api.Since the project was super simple, we decided to not use any framework nor a database. We generally use Yii Framework which is super fast anyway.We cache the api calls and load them at need and we do the display, search, sorting,
> filtering, etc just in PHP and we're using a bit JS for UI/UX.This quickly reminded us how blazing fast things can be when there is little to no overhead, as opposed to using a framework and connecting to a database.I know that most of the projects don't get away with this, but this was a super simple
> case where we wanted to experiment a bit.So i guess the performance tip is, one of the things you should always consider when
> developing is to make sure you avoid any overhead that can be avoided by

> Default.Also, talking with a database is expensive.I do also second his opinion about keeping apps simple in every possible way. You should always avoid to have any extra overhead of functional features. Which might not have any particular role in the application, but causes the app to slow down because of unwanted resource usage.

Another active Redditor, TorbenKoehn made his remarks about cache issues. He said that solving cache issues helps optimizing app load time and performance.

TorbenKoehn (https://www.reddit.com/user/TorbenKoehn)

> -Have one million data sets with like a total of 10 million related dataset
> -Your customer requires to make them searchable by many factors
> -Write SQL queries with 8 joins and GROUP BY/DISTINCT and wonder why the page takes 80 seconds to load (or your SQL server is suddenly gone)
> -Learn about ElasticSearch, build an index, use it
> -Have ~200ms response times for your searches
> -??????
> -PROFIT!!!There are three things that I constantly have to optimize regarding performance: 1. Database interaction. Use caching, memory tables/own search indexes and things like ES to solve this. 2. File system interaction. Especially related to file caching or logging. To solve this, use services like Redis for caching and log to ES. 3. Intensively mapping arrays to objects, back to arrays, back to objects etc.. To solve this, KISS. DTOs are nice for auto-completion and stability, but have a lot of memory overhead with large datasets and intensive mapping through multiple loops.

Jimlei replied with few points that hold importance for enhancing apps performance:

jimlei (https://www.reddit.com/user/jimlei)

> Lots of great tips already. Keep your code up to date, amazing work has been done with PHP itself to speed up apps so make sure you’re on a rather new version
> permalink (https://www.reddit.com/r/PHP/comments/8y9xqk/share_performance_tips_for_php_code_which_you_use/e29j1np/)
> -embed
> -save
> -report
> -give gold
> -reply

Yewbatrom shared his experience about how he optimized his apps performance by just fixing two issues:

Yewbatrom (https://www.reddit.com/user/Yewbatrom)

> Honestly the best performance improvements I've gotten out of PHP -> properly structuring my database and properly setting up Indexes.

While this is how another Redditor Garethp responded to the question:

Garethp (https://www.reddit.com/user/Garethp)

> If you've got large amounts of rows being returned from the db, fetch them as cursors instead of hydrating them on query. Then use nikic\iter to map and filter using generators so you can do that without unwinding the cursor until the end

Evilmaus also explained the question taking database issues into regard. He stated how he fixed small database issues to make his app performance faster.

evilmaus (https://www.reddit.com/user/evilmaus)

> I always put foreign key constraints on my database tables. I do it primarily for data integrity, but it also makes for fast joins. I find that foreign keys are also my second most used columns for selecting from a database. I also put in unique indexes on columns that should contain unique values. Again, this is for integrity purposes, but has the nice side benefit that these columns are also common for me to use in queries.
>
> When possible, I take collections of entities and key those collections by whatever ID the entities have. That way, I can quickly look up specific ones instead of having to search through the entire collection every time I want a particular item. If I'm wanting entities by something that isn't necessarily unique, I just group by that field instead.
>
> I haven't yet had call to use them, but binary trees are rather fun in concept.

Final Words

These were some of the valuable inputs I have gathered from Twitter and Reddit, that defines the valuable insights of developer community. As what they think about PHP performance tuning and what it takes to optimize it more to get the optimum benefit out of apps. So what is your opinion about this topic? Do let me know your thoughts in the comments section below.

Share your opinion in the comment section. COMMENT NOW

Shahroze Nawaz (https://www.cloudways.com/blog/author/shahroze-nawaz/)

Shahroze is a PHP Community Manager at Cloudways - A Managed PHP Hosting Platform. Besides his work life, he loves movies and travelling. You can email him at shahroze.nawaz@cloudways.com

Get Connected on: Twitter (https://twitter.com/_shahroznawaz) Community Forum (https://community.cloudways.com/)

THERE'S MORE TO READ.

PHP 8 Min Read

Perform PHP Debugging By Integrating Xdebug With VsCode (https://www.cloudways.com/blog/php-debug-with-xdebug/)

Shahroze Nawaz
Published on 1st June
(https://www.cloudways.com/blog/author/shahroze-nawaz/)

PHP 7 Min Read

How to Host PHP on Amazon AWS EC2 (https://www.cloudways.com/blog/host-php-on-aws-cloud/)

Ahmed Khan
Published on 31st May
(https://www.cloudways.com/blog/author/ahmed-khan/)

PHP 6 Min Read

Create a Real Time Chat App with Pusher in... (https://www.cloudways.com/blog/real-time-chat-app-php/)

Shahroze Nawaz
Published on 27th May
(https://www.cloudways.com/blog/author/shahroze-nawaz/)

PHP 5 Min Read

Best Open Source PHP Servers for Your Next Web... (https://www.cloudways.com/blog/best-php-servers/)

Shahroze Nawaz
Published on 23rd May
(https://www.cloudways.com/blog/author/shahroze-nawaz/)

PHP 5 Min Read

How To Install Mediawiki on Cloud (https://www.cloudways.com/blog/install-mediawiki/)

Shahroze Nawaz

Published on 22nd May
(https://www.cloudways.com/blog/author/shahroze-nawaz/)

PHP 8 Min Read

Configure Varnish Cache And Speed Up Your Application Load... (https://www.cloudways.com/blog/configure-varnish-cache/)

Olususi k Oluyemi
Published on 9th May
(https://www.cloudways.com/blog/author/olususi/)

PRODUCT & SOLUTION

WordPress Hosting (https://www.cloudways.com/en/wordpress-cloud-hosting.php)

Magento Hosting (https://www.cloudways.com/en/magento-managed-cloud-hosting.php)

PHP Cloud Hosting (https://www.cloudways.com/en/php-cloud-hosting.php)

Laravel Hosting (https://www.cloudways.com/en/laravel-hosting.php)

Drupal Hosting (https://www.cloudways.com/en/drupal-cloud-hosting.php)

Joomla Hosting (https://www.cloudways.com/en/joomla-cloud.php)

PrestaShop Hosting (https://www.cloudways.com/en/prestashop-hosting.php)

WooCommerce Hosting (https://www.cloudways.com/en/hosting-woocommerce.php)

Cloudways Platform (https://platform.cloudways.com/signup)

Cloudways API (https://developers.cloudways.com/)

Breeze – Free WordPress Cache (https://www.cloudways.com/en/free-wordpress-cache-plugin-breeze.php)

Add-ons (https://www.cloudways.com/en/addons.php)

CloudwaysCDN (https://www.cloudways.com/en/cdn-services.php)

CloudwaysBot (https://www.cloudways.com/en/cloudwaysbot.php)

COMPANY

SUPPORT

QUICK LINKS

Follow Us On

(https://www.facebook.com/cloudways) (https://twitter.com/cloudways) (https://www.linkedin.com/company/cloudways) (https://www.cloudways.com/blog/updates/)

(https://www.cloudways.com/en/)

52 Springvale, Pope Pius XII Street Mosta MST2653, Malta

80
SHARES