

AN OPENSOURCE EBOOK

Laravel



Tips and Tricks

Bobby Iliev

Table of Contents

About the book

- **This version was published on July 25 2021**

This is an open-source **Laravel Tips and Tricks eBook** that is a collection of my own notes that I've put together for myself throughout the years. You would more likely than not need many of those tips at some point in your career as a Laravel Developer.

The guide is suitable for anyone working as a Laravel developer and would love to learn some random Laravel tips and tricks. You can read the chapters in this book in a random order as they are completely separate tips or tricks.

About the author

My name is Bobby Iliev, and I have been working as a Linux DevOps Engineer since 2014. I am an avid Linux lover and supporter of the open-source movement philosophy. I am always doing that which I cannot do in order that I may learn how to do it, and I believe in sharing knowledge.

I think it's essential always to keep professional and surround yourself with good people, work hard, and be nice to everyone. You have to perform at a consistently higher level than others. That's the mark of a true professional.

For more information, please visit my blog at <https://bobbyiliev.com>, follow me on Twitter [@bobbyiliev_](#) and [YouTube](#).

Sponsors

This book is made possible thanks to these fantastic companies!

DigitalOcean

DigitalOcean is a cloud services platform delivering the simplicity developers love and businesses trust to run production applications at scale.

It provides highly available, secure, and scalable compute, storage, and networking solutions that help developers build great software faster.

Founded in 2012 with offices in New York and Cambridge, MA, DigitalOcean offers transparent and affordable pricing, an elegant user interface, and one of the largest libraries of open source resources available.

For more information, please visit <https://www.digitalocean.com> or follow [@digitalocean](#) on Twitter.

If you are new to DigitalOcean, you can get a free \$100 credit and spin up your own servers via this referral link here:

[Free \\$100 Credit For DigitalOcean](#)

DevDojo

The DevDojo is a resource to learn all things web development and web design. Learn on your lunch break or wake up and enjoy a cup of coffee with us to learn something new.

Join this developer community, and we can all learn together, build

together, and grow together.

[Join DevDojo](#)

For more information, please visit <https://www.devdojo.com> or follow [@thedeveloper](#) on Twitter.

Ebook PDF Generation Tool

This ebook was generated by [Ibis](#) developed by [Mohamed Said](#).

Ibis is a PHP tool that helps you write eBooks in markdown.

Book Cover

The cover for this ebook was created with [Canva.com](https://www.canva.com).

If you ever need to create a graphic, poster, invitation, logo, presentation – or anything that looks good — give Canva a go.

License

MIT License

Copyright (c) 2020 Bobby Iliev

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

How to Install Laravel on DigitalOcean with 1-Click?

[Laravel](#) is an amazing PHP framework that makes working with PHP great.

As of the time of writing this tutorial, Laravel has more than 105 million installs according to [Packagist](#), so as you can imagine there are multiple ways of **installing Laravel on DigitalOcean** or any other cloud provider.

In this tutorial, we will go through a few ways of installing Laravel and also show you how to do this with just 1-Click on DigitalOcean!

All you need in order to follow along is a DigitalOcean account. To make things even better you can use the following referral link to get **free \$100 credit** that you could use to deploy your servers and test the guide yourself:

[DigitalOcean \\$100 Free Credit](#)

Installing Laravel on DigitalOcean

If you wanted to configure your LEMP server from scratch you could, for example, do it manually by following this amazing step by step guide provided by DigitalOcean:

- [How to Install and Configure Laravel with LEMP on Ubuntu 18.04](#)

Another option of automating the above steps is to use the [LaraSail](#) open-source script to set up your server and install Laravel:

- <https://github.com/thedevdojo/larasail>

Using the Laravel DigitalOcean 1-Click

DigitalOcean has a Marketplace, where you could get a lot of various 1-Click Applications which you can deploy on your servers and Kubernetes clusters.

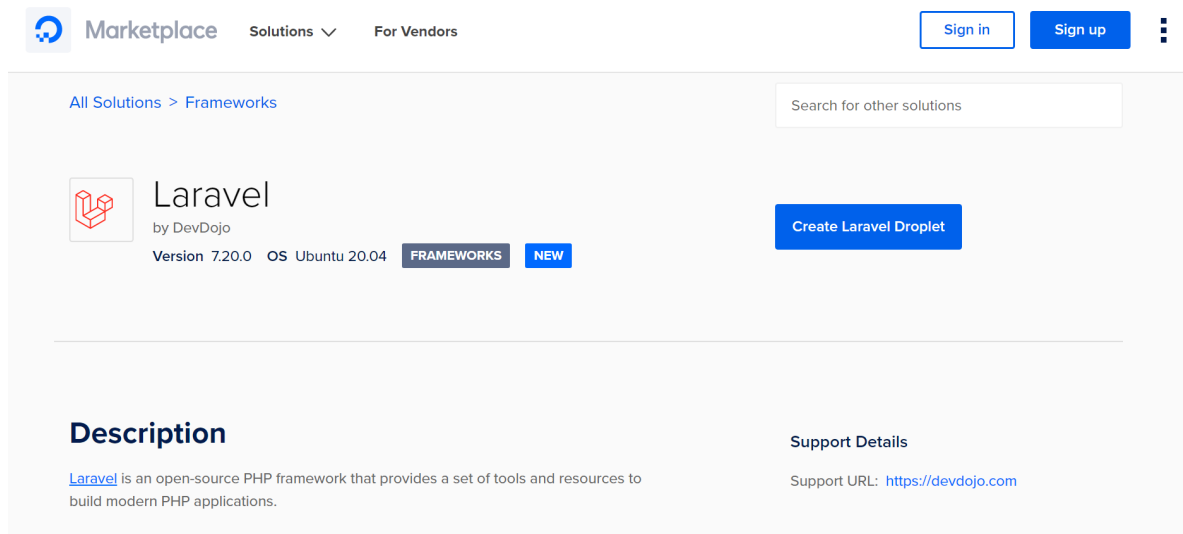
Recently we built a Laravel image and submitted a listing to DigitalOcean. The image is now available on the DigitalOcean Marketplace:

<https://marketplace.digitalocean.com/apps/laravel>

The image comes with a preconfigured LEMP stack and certbot installed. The software stack that comes with the image is:

- [Laravel](#) 7.20.0
- [Nginx](#) 1.18.0
- [MySQL server](#) 8.0.20
- [PHP](#) 7.4.3
- [Certbot](#) 0.40.0
- [Composer](#) 1.10.1

In order to use the image just go to the [1-Click Laravel application page](#) and hit the **Create Laravel Droplet** button:



The screenshot shows the Laravel Marketplace interface. At the top, there's a navigation bar with 'Marketplace', 'Solutions', and 'For Vendors'. On the right, there are 'Sign in' and 'Sign up' buttons. Below the navigation bar, there's a search bar with the placeholder text 'Search for other solutions'. The main content area features the 'Laravel' solution by DevDojo. It includes a Laravel logo, the text 'by DevDojo', and details like 'Version 7.20.0' and 'OS Ubuntu 20.04'. There are also labels for 'FRAMEWORKS' and 'NEW'. A prominent blue button labeled 'Create Laravel Droplet' is visible. Below this, there's a 'Description' section stating that Laravel is an open-source PHP framework for building modern PHP applications. To the right of the description is a 'Support Details' section with the URL 'https://devdojo.com'.

After that just follow the standard flow and choose the details that match your needs like:

- The size of the Droplet
- Choose a datacenter region
- Choose your SSH keys
- Choose hostname
- I would recommend enabling backups as well

Finally hit the **Create Droplet** button. Then in around 30 seconds or so your Laravel server will be up and running!

Completing the Laravel configuration

To complete the installation, copy your IP address and [SSH to the Droplet](#).

You would get to an interactive menu that would ask you for the following details:

```
Enter the domain name for your new Laravel site.
(ex. example.org or test.example.org) do not include www or
http/s
-----
Domain/Subdomain name:
```

There just type your domain name or subdomain name, I would go for `laravel.bobbyiliev.com` for my example.

After that you would see the following output:

```
Configuring Laravel database details
Generating new Laravel App Key
Application key set successfully.
```

Then you will be asked if you want to secure your Laravel installation with an SSL certificate:

Next, you have the option of configuring LetsEncrypt to secure your **new** site.

Before doing this, be sure that you have pointed your domain **or** subdomain to this server's **IP address**.

You can also run LetsEncrypt certbot later with the command **'certbot --nginx'**

**Would you like to use LetsEncrypt (certbot)
to configure SSL(https) for your new site? (y/n):**

Note that before hitting **y** make sure to have your domain name or subdomain DNS configured so that your A record points to the Droplet's IP address, otherwise Let's Encrypt will not be able to validate your domain and would not issue a certificate for you.

If you don't want a certificate, just type **n** and hit enter.

That is pretty much it! Now if you visit **yourdomain.com** in your browser you would see a fresh new installation!

Video Demo

Here's also a quick video demo on how to do the above:

[How to Install Laravel on DigitalOcean with 1-Click](#)

Conclusion

Thanks to the Laravel 1-Click installation from the DigitalOcean's Marketplace, we can have a fully running LEMP server with Laravel installed in less than 30 seconds!

[Originally posted here.](#)

How to get a free domain name for your Laravel project

There could be various reasons why you would need a free domain for your project.

For example, having multiple side projects could be quite costly in case that your projects are not generating any income. So saving costs could be crucial.

Another reason why you might need a free domain name is that you might want to do some just testing and not really need an actual domain which could cost you anywhere from \$5 to +\$50 dollars depending on the provider and the domain extension.

In this tutorial, I will show you **how to get a free domain name** from [Freenom](#) and use it with your Laravel Project.

Choosing a free domain name

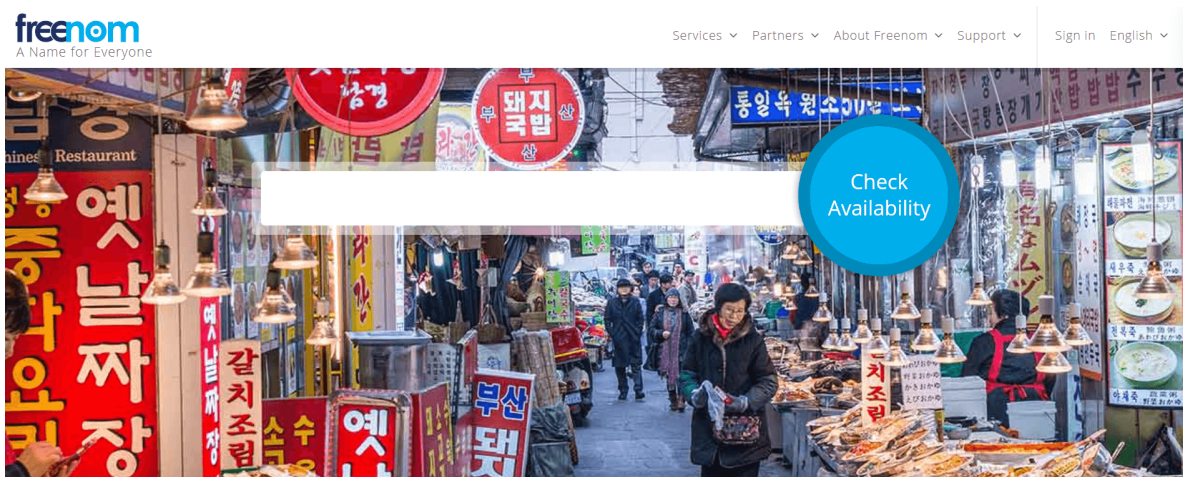
In this tutorial, we will use [Freenom](#) to register a free domain name! As far as I am aware, Freenom is the world's only free domain name provider.

They provide free domain names with the following domain extensions:

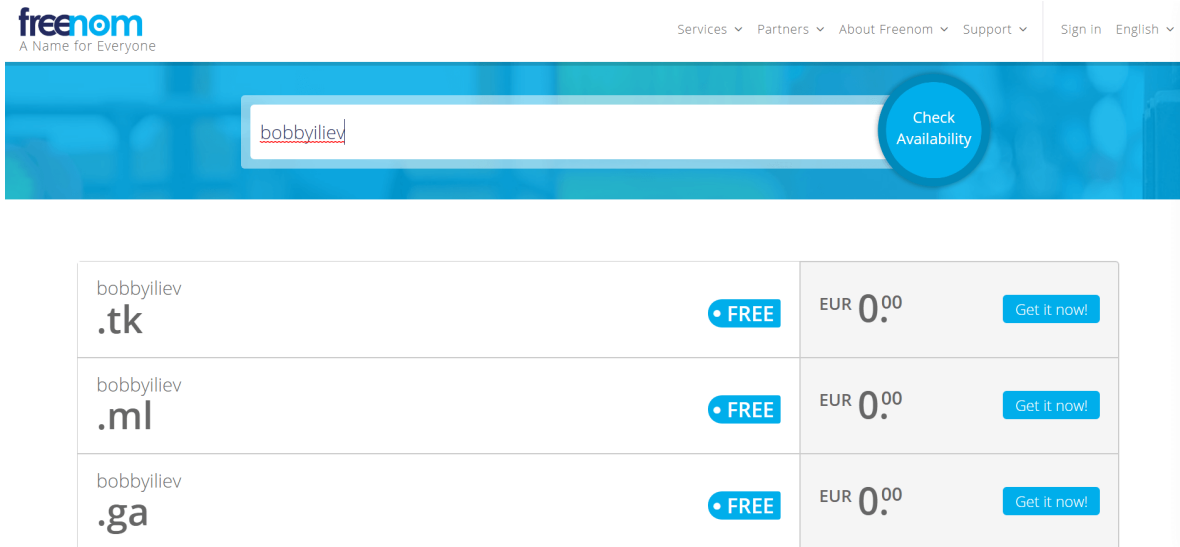
- .TK
- .ML
- .GA
- .CF
- .GQ

Unfortunately, Freenom does not offer free **.com** domains but for testing purposes, only the above domain extensions are a great alternative!

To check if a specific domain is available, just visit the [Freenom](#) website, you would get to a page where you can search and check if a specific domain is available:



In my case I will look for **bobbyiliev** and then choose the available extension:

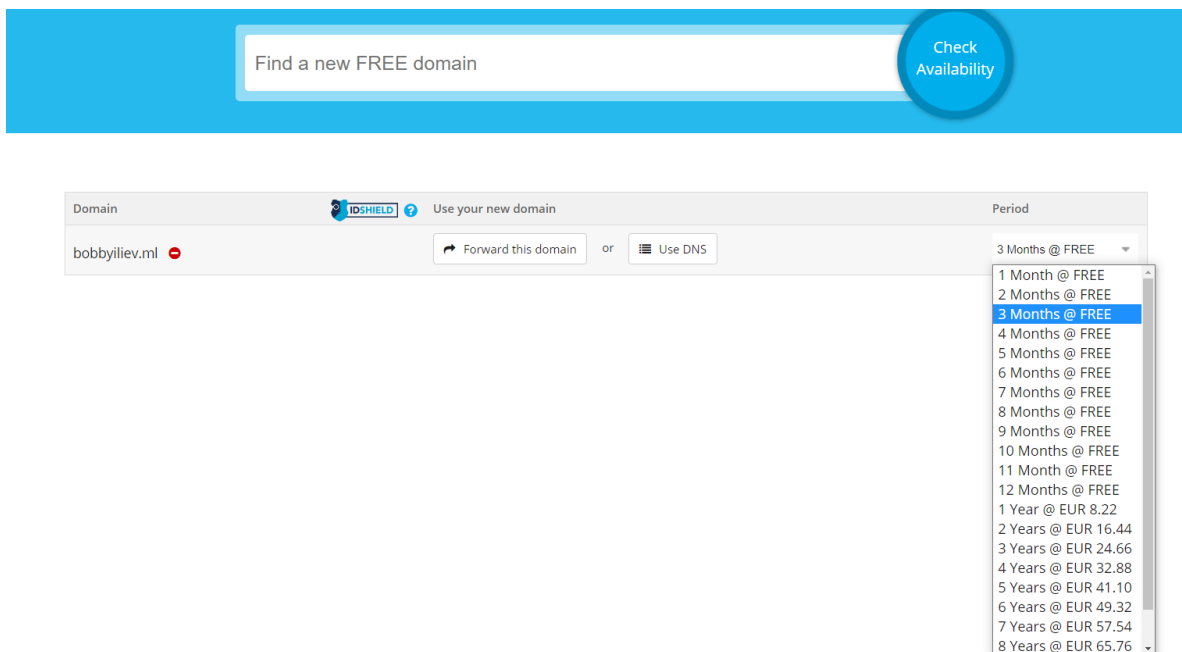


The screenshot shows the Freenom website interface. At the top, there's a navigation bar with links for Services, Partners, About Freenom, Support, Sign in, and English. Below this is a search bar with the text "bobbyiliev" entered. To the right of the search bar is a "Check Availability" button. Below the search bar, there's a table showing the availability of domains for "bobbyiliev".

Domain	Status	Price	Action
bobbyiliev.tk	FREE	EUR 0.00	Get it now!
bobbyiliev.ml	FREE	EUR 0.00	Get it now!
bobbyiliev.ga	FREE	EUR 0.00	Get it now!

In my case, **bobbyiliev.ml** is available so I will go for that one by clicking on the **Get it now!** button, and then click on **Checkout**.

You would get to a page where you could choose for how long would you like to keep the domain name for:



The screenshot shows the Freenom website interface. At the top, there's a search bar with the text "Find a new FREE domain". To the right of the search bar is a "Check Availability" button. Below the search bar, there's a table showing the availability of domains for "bobbyiliev".

Domain	Status	Price	Action
bobbyiliev.ml	FREE	EUR 0.00	Get it now!

Below the table, there's a section for "Use your new domain". It has a "Forward this domain" button and a "Use DNS" button. To the right of these buttons is a "Period" dropdown menu. The dropdown menu is open, showing a list of periods and their corresponding prices:

- 1 Month @ FREE
- 2 Months @ FREE
- 3 Months @ FREE
- 4 Months @ FREE
- 5 Months @ FREE
- 6 Months @ FREE
- 7 Months @ FREE
- 8 Months @ FREE
- 9 Months @ FREE
- 10 Months @ FREE
- 11 Month @ FREE
- 12 Months @ FREE
- 1 Year @ EUR 8.22
- 2 Years @ EUR 16.44
- 3 Years @ EUR 24.66
- 4 Years @ EUR 32.88
- 5 Years @ EUR 41.10
- 6 Years @ EUR 49.32
- 7 Years @ EUR 57.54
- 8 Years @ EUR 65.76

Choose the period from the dropdown menu and then hit next. After that follow the steps and sign up for a free account.

Once you are ready with the registration process, then go ahead and proceed to the next step where you will need to add your new free domain name to your DigitalOcean account.

Adding your Domain to DigitalOcean

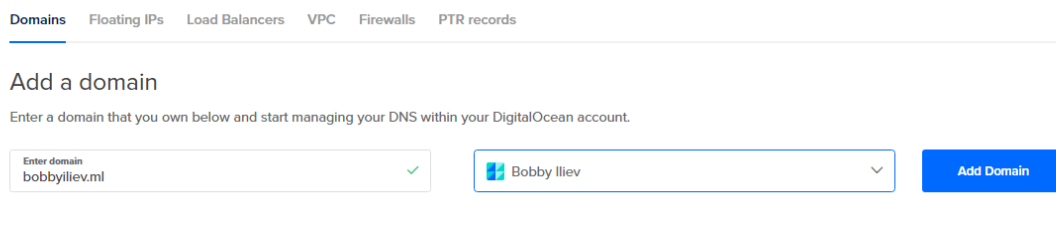
If you don't have a DigitalOcean account already make sure to create one via this link here:

[Sign up for DigitalOcean](#)

Once you've created a DigitalOcean account follow these steps here:

- First go to your [DigitalOcean Control Panel](#)
- Then click on Networking on the left
- After that click on Domains
- And there add your new domain name button and then add the domain name that you just registered:

Networking



The screenshot shows the 'Networking' section of the DigitalOcean control panel. Under the 'Domains' tab, there is a section titled 'Add a domain' with the instruction: 'Enter a domain that you own below and start managing your DNS within your DigitalOcean account.' Below this instruction is a form with two input fields and an 'Add Domain' button. The first input field is labeled 'Enter domain' and contains the text 'bobbyiliev.ml' with a green checkmark to its right. The second input field contains a user profile icon and the name 'Bobby Iliev' with a dropdown arrow to its right. The 'Add Domain' button is blue and located to the right of the second input field.

Then you would see your new DNS zone where you would need to add an A record for your domain name and point it your Laravel server! For more information on [how to manage your DNS make sure to read this guide!](#)

Updating your Nameservers

Once you have your Domain name all configured and ready to go in DigitalOcean, then you need to update your Nameservers, so that your domain would start using your new DigitalOcean DNS zone.

To do so just copy the following 3 nameservers:

- ns1.digitalocean.com
- ns2.digitalocean.com
- ns3.digitalocean.com

And then go back to your Freenom account, there follow these steps:

- From the menu click on Services
- Then click on My Domains
- After that click on Manage Domain for your new domain
- Then from the "Management Tools" dropdown click on "Nameservers"
- There choose "Use custom nameservers (enter below)" and enter the 3 DigitalOcean nameservers from above and click "Save"

Managing bobbyiliev.ml

Information Upgrade Management Tools Manage Freenom DNS

Changes Saved Successfully!

Nameservers

You can change where your domain points to here. Please be aware changes can take up to 24 hours to propagate.

☐ Use default nameservers (Freenom Nameservers)

☒ Use custom nameservers (enter below)

Nameserver 1

ns1.digitalocean.com

Nameserver 2

ns2.digitalocean.com

Nameserver 3

ns3.digitalocean.com

Finally, it could take up to 24 hours for the DNS to propagate over the Globe and after that, you will be able to see your Laravel application when visiting your free domain name in your browser!

Conclusion

Now you know how to get a free domain name for your Laravel Project and add it to DigitalOcean where you could manage your DNS zone and point it to your Laravel server.

[Originally posted here](#)

8 Awesome VS Code Extensions for Laravel Developers

While I'm still a sublime fan for quite some time, I've been mainly using VS Code.

For anyone who is just getting started with Laravel, I would recommend going through this [Laravel basics course here!](#)

Here is a list of my top 8 VS Code extensions for Laravel developers, which would help you be more productive!

1. Laravel Blade Snippets

The [Laravel blade snippets](#) extension adds syntax highlight support for Laravel Blade to your VS Code editor.



Some of the main features of this extension are:

- Blade syntax highlight
- Blade snippets
- Emmet works in blade template
- Blade formatting

In order to make sure that the extension works as expected, there is some additional configuration that needs to be done. Go to **File -> Preferences -> Settings** and add the following to your **settings.json**:

```
"emmet.triggerExpansionOnTab": true,  
"blade.format.enable": true,  
"[blade]": {  
  "editor.autoClosingBrackets": "always"  
},
```

This will enable tab completion for emmet tags and if enable blade formatting.

For more information on the available snippets, make sure to check the documentation here:

[VSCode extensions for laravel](#)

2. Laravel Snippets

This one is probably my personal favorite! The [Laravel Snippets extension](#) adds snippets for the Facades like `Request::`, `Route::` etc.



Some of the supported snippet prefixes include:

- Auth
- Broadcast
- Cache
- Config
- Console
- Cookie
- Crypt
- DB
- Event
- View

For more information on the available snippets, make sure to check the documentation here:

[VSCode extensions for laravel](#)

3. Laravel Blade Spacer

Isn't it annoying when you try to echo out something in your Blade views with `{{ }}` and your whole line going back 4 spaces? Well, luckily, the [Laravel Blade Spacer](#) fixes that!

The Laravel blade spacer extension automatically adds spacing to your blade templating markers:



For more information, make sure to check the documentation here:

[VSCode extensions for laravel](#)

4. Laravel Artisan

I personally like to use the command line all the time, but I have to admit that the [Laravel Artisan](#) extension is awesome! It lets you run Laravel Artisan commands from within Visual Studio Code directly!



Some of the main features are:

- Make files like Controllers, Migrations, etc.
- Run Your own Custom Commands
- Manage your database
- Clear the Caches
- Generate Keys
- View all application routes
- Manage your local php server for test purposes

For more information, make sure to check the documentation here:

[VSCode extensions for laravel](#)

5. Laravel Extra Intellisense

The [Laravel Extra Intellisense](#) extension provides autocompletion for Laravel in VSCode.



The extension comes with auto-completion for:

- Route names and route parameters
- Views and variables
- Configs
- Translations and translation parameters
- Laravel mix function
- Validation rules
- View sections and stacks
- Env
- Route Middlewares

For more information, make sure to check the documentation here:

[VSCode extensions for laravel](#)

6. Laravel goto Controller

As your application grows, the number of your Controllers grows as well, so at some point, you might end up with hundreds of controllers. Hence finding your way around might get tedious.

This is the exact problem that the [Laravel-goto-controller](#) VScode extension solves.

The extension allows you to press **Alt** + click on the name of the controller in your routes file, and it will navigate you from the route to the respective controller file:



For more information, make sure to check the documentation here:

[VSCode extensions for laravel](#)

7. Laravel goto View

Similar to the Laravel goto Controller extension, the [Laravel goto View VSCode extension](#) allows you to go from your Controller or Route to your view. This can save you quite a bit of time!

You can use **Ctrl** or **Alt** + click to jump to the first matched Blade View file:



For more information, make sure to check the documentation here:

[VSCode extensions for laravel](#)

8. DotENV syntax highlighting

This one is pretty simple but handy. The [DotENV](#) VS Code extension is used to highlight the syntax of your `.env` file, which could be quite handy for spotting some problems:



For more information, make sure to check the documentation here:

[VSCode extensions for laravel](#)

Book recommendation

If you are a Laravel fan, make sure to check out the [The Laravel Survival Guide](#) ebook!

Conclusion

If you like all those extensions, you can take a look at the [Laravel Extension Pack for Visual Studio Code](#), where you could get all of the mentioned extensions as 1 bundle!

The only extension not included in the pack is the Laravel Blade Spacer, so make sure to install it separately!

[Originally posted here.](#)

What is Laravel Jetstream and how to get started

[Laravel 8](#) was released on September 8th 2020 along with Laravel Jetstream.

[Laravel Jetstream](#) is a new application scaffolding for Laravel. Laravel Jetstream replaces the legacy Laravel authentication UI available for previous Laravel versions.

In this tutorial, I will give you a quick introduction to what exactly Laravel Jetstream is and how to get started with it.

If you want to follow along, you would need a LEMP server together with [composer](#) or the latest Laravel installer.

I will use DigitalOcean for the demo. If you do not have a DigitalOcean account yet, you can use the following referral link to get free \$100 credit that you could use to deploy your servers and test the guide yourself:

[DigitalOcean \\$100 Free Credit](#)

What is Laravel Jetstream

Jetstream gives you a better starting point for your new projects. It includes the following components:

- Login and registration functionality
- Email verification
- Two-factor authentication
- Session management
- API support via Laravel Sanctum

Laravel Jetstream replaces the legacy Laravel authentication UI available for previous Laravel versions.

Jetstream uses Tailwind CSS, and you can choose between Livewire or Inertia.

Laravel Jetstream is free and opensource.

Installing Laravel Jetstream

You can choose between a couple of ways of installing Laravel Jetstream. You could either use **composer** or the Laravel installer.

Installing Jetstream with Laravel installer

If you already have the latest version of the [Laravel installer](#), you just need to use the **--jet** flag in order to install a new Laravel Jetstream project:

```
laravel new project-name --jet
```

After that, as usual, make sure to run your migrations:

```
php artisan migrate
```

Installing Jetstream with Composer

If you prefer using composer, you need to run the following command inside your Laravel directory just like you would with any other package:

```
composer require laravel/jetstream
```

Note: you need to have Laravel 8 installed. Otherwise, the above command will fail.

After that, you would need to run **artisan jetstream:install** and

specify the stack that you want to use:

- If want to use Livewire with Blade run:

```
php artisan jetstream:install livewire
```

- And if you want to use Inertia with Vue run:

```
php artisan jetstream:install inertia
```

You may also add the `--teams` flag to enable [Laravel Jetstream team support](#).

After that, execute:

```
npm install && npm run dev
```

The command above will build your assets.

Finally, make sure to run your migrations:

```
php artisan migrate
```


Authentication

Your new Jetstream application comes out of the box with:

- Login form
- Two-factor authentication
- Registration form
- Password reset
- Email verification

You can find those views at:

```
resources/views/auth
```

The backend logic is powered by [Laravel Fortify](#).

You can find the Fortify actions at the following directory:

```
app/Actions/Fortify/
```

And you can find the Fortify configuration at:

```
config/fortify.php
```

In the `fortify.php` config file, you can make some changes like enable and disable different features like:

```
'features' => [  
    Features::registration(),  
    Features::resetPasswords(),  
    // Features::emailVerification(),  
    Features::updateProfileInformation(),  
    Features::updatePasswords(),  
    Features::twoFactorAuthentication(),  
],
```

Profile management

Right out of the box, Jetstream provides you and your users with user profile management functionality which allows users to update their name, email address, and their profile photo.

The user profile view is stored at:

```
resources/views/profile/update-profile-information-  
form.blade.php
```

And in case you are using Inertia, the view can be found at:

```
resources/js/Pages/Profile/UpdateProfileInformationForm.vue
```

The following file handles the user update logic:

```
app/Actions/Fortify/UpdateUserProfileInformation.php
```

In case that you want to, you could also disable the user profile picture via your Jetstream config file at:

```
config/jetstream.php
```

Just comment out the `Features::profilePhotos()` line:

```
'features' => [  
    // Features::profilePhotos(),  
    Features::api(),  
    // Features::teams(),  
],
```

Jetstream Security

Laravel Jetstream comes with the standard functionality that allows users to update their password and log out:



However, what's more, impressive is that Jetstream also offers two-factor authentication with QR code, which the users could enable and disable directly:



Another brilliant security feature is that users can logout other browser sessions as well.



The profile Blade views can be found at:

```
resources/views/profile/
```

And the if you are using Inertia, you can find them at:

```
resources/js/Pages/Profile/
```

Jetstream API

Laravel Jetstream uses [Laravel Sanctum](#) to provide simple token-based API.

With Sanctum, each user can generate API tokens with specific permissions like Create, Read, Update, and Delete.

Then to check the incoming requests, you can use the `tokenCan` method like this:

```
$request->user()->tokenCan('read');
```

Again you can disable API support in your `config/jetstream.php` config file.

Jetstream Teams

If you used the `--team` flag during your Jetstream installation, your website would support team creation and management.

With the Jetstream teams feature, each user can create and belong to multiple different teams.

For more information about Jetstream teams, you can take a look at the official documentation [here](#).

Conclusion

Laravel Jetstream gives you a great head start when starting a new project!

I also suggest going through this post here on [what's new in Laravel 8!](#)

[Originally posted here](#)

How to check Laravel Blade View Syntax using artisan

The Laravel community has been growing exponentially, and there are a lot of fantastic packages out there. I recently came across a Laravel package that provides an artisan command to check the syntax of your blade templates.

In this tutorial, I will show you how to use the [Laravel Blade Linter package](#) to check your blade views syntax!

Before getting started, you need to have a Laravel project up and running.

You can use the following referral link to get free \$100 credit that you could use to deploy your servers and test the guide yourself:

[DigitalOcean \\$100 Free Credit](#)

After that, you can follow the steps on [How to Install Laravel on DigitalOcean with 1-Click here!](#)

Installation

```
composer require magentron/laravel-blade-lint
```

Testing

Once you have the package installed, in order to test your Blade syntax, you need to run the following command:

```
php artisan blade:lint
```

If you don't have any errors, you would see the following output:

```
All Blade templates OK!
```

On another note, if there are any issues anywhere in your blade files, you would get an error like this one:

```
PHP Parse error:  syntax error, unexpected ':', expecting '('  
in /var/www/html/resources/views/welcome.blade.php on line 103  
Found 1 errors in Blade templates!
```

In my case, it tells me that I have a syntax error on line 103 in my `welcome.blade.php` file.

If you don't want to check all of your views, you can specify the path to a specific directory:

```
php artisan blade:lint resources/views/blog
```

Conclusion

The Laravel Blade Linter is a great package that can help you avoid errors before pushing your code to production!

If you like the package, make sure to contribute at on [Github](#)!

[Originally posted here](#)

How to speed up your Laravel application with PHP OPcache

Using PHP [OPcache](#) is a great way to improve your overall performance. OPcache stores pre-compiled script bytecode in memory, which eliminates the need for PHP to load and parse scripts on every request.

In this tutorial, you will learn how to use the Laravel along with OPcache to speed up your website!

In order for you to be able to follow along, you need to have a Laravel project up and running.

You can use the following referral link to get free \$100 credit that you could use to deploy your servers and test the guide yourself:

[DigitalOcean \\$100 Free Credit](#)

After that, you can follow the steps on **[How to Install Laravel on DigitalOcean with 1-Click here!](#)**

Enable PHP OPcache

There are a couple of ways of checking if you already have PHP OPcache enabled.

You could either add a PHP info file in your Laravel **public** folder with the following content:

```
<?php
phpinfo();
```

And then visit the file via your browser. After that, use **CTRL+F** and search for OPcache. You will see the following information:



Another way of checking if OPcache is enabled is to run the following command via your terminal:

```
php -m | grep -i opcache
```

The output that you would see the following result:

```
Zend OPcache
```

If you don't have OPcache enabled, you can install it with the following command on Ubuntu:

```
sudo apt install php-opcache
```

If you are not using Ubuntu, you can install PHP OPcache using **pecl**:

<https://pecl.php.net/package/ZendOPcache>

If you used the 1-Click image from the prerequisites, OPcache would already be installed for you.

Configure PHP OPcache

Once you have OPcache installed, you can adjust some of the default configuration settings to optimize the performance.

To make those changes, you need to edit your `php.ini` file. If you are not sure where exactly the `php.ini` file is located at, you can again use a PHP info file to check the location.

If you have used the 1-Click image from the prerequisites, you will find the `php.ini` file at:

```
/etc/php/7.4/fpm/conf.d/10-opcache.ini
```

Using your favorite text editor, open the file:

```
nano /etc/php/7.4/fpm/conf.d/10-opcache.ini
```

Then at the bottom of the file add the following configuration:

```
opcache.memory_consumption=256
opcache.interned_strings_buffer=64
opcache.max_accelerated_files=32531
opcache.validate_timestamps=0
opcache.enable_cli=1
```

A quick rundown of the values:

- `opcache.memory_consumption=256`: This is the size of the memory storage used by OPcache. You can increase the `opcache.memory_consumption=256` value in case that you have

enough RAM on your server.

- `opcache.interned_strings_buffer=64`: The amount of memory allocated to storing interned strings. The value is in megabytes.
- `opcache.max_accelerated_files=32531`: The number of scripts in the OPcache hash table or, in other words, how many scripts can be cached in OPcache.
- `opcache.validate_timestamps=0`: With this directive, we specify that OPcache should not be cleared automatically, which means that we would need to do this manually.
- `opcache.enable_cli=1`: This enables the OPcache for the CLI version of PHP, which could be beneficial if you are using any `artisan` commands.

Once you make the change, you need to restart PHP FPM:

```
systemctl restart php7.4-fpm.service
```

For more information on the OPcache configuration, make sure to go through the official documentation here:

<https://www.php.net/manual/en/opcache.configuration.php>

Configure Laravel OPCache

In order to have some better management over the caching, we will use the [Laravel OPCache](#) package.

To use the package, you would need to have Laravel 7 or newer.

To install the package, just run the following command:

```
composer require appstract/laravel-opcache
```

One important thing that you need to keep in mind is that your `APP_URL` needs to be set correctly in your `.env` file so that it matches your domain.

Once you have the package installed, it provides you with some excellent PHP `artisan` commands which you could use to help you manage your OPCache:

- Artisan command to clear OPCache:

```
php artisan opcache:clear
```

- Artisan command to show OPCache config:

```
php artisan opcache:config
```

- Artisan command to show OPCache status:

```
php artisan opcache:status
```

- Artisan command to pre-compile your application code:

```
php artisan opcache:compile {--force}
```

Conclusion

Enabling PHP OPcache is an excellent and quick way to boost the performance of your Laravel application without having to make any significant changes.

If you like the Laravel OPcache package, make sure to give it a star on [Github](#)!

I hope that this helps!

[Originally posted here](#)

What is Laravel Sail and how to get started

Laravel is a fantastic open-source PHP framework that is designed to develop web applications following the model-view-controller scheme. And now here comes Laravel Sail.

Laravel Sail was just recently released, and it is a light-weight command-line interface for interacting with Laravel's default Docker development environment. What this means is that you won't be needing to use Docker to create different containers manually, but you can just use use Laravel Sail to do that for you.

By using Laravel Sail and you will get a fully working local development environment.

Laravel Sail uses the `docker-compose.yml` file and the sail script that is stored at the vendor folder of your project at `vendor/bin/sail`. The sail script provides a CLI with convenient methods for interacting with the Docker containers defined by the `docker-compose.yml` file.

In this post, you are going to learn how to install and get started with Laravel Sail.

In order to get started with Laravel Sail, all that you need is to have Docker and Docker Compose installed on your Laptop or your server.

If you don't know what Docker is or how it works, you can check out these post series to help you understand what Docker is and how it works.

Introduction to Docker

Installing And Setting Up Laravel Sail

One of the many great things about Laravel Sail is that it is automatically installed with all new Laravel applications so this way you can start using it straight away.

To get a new blank project you can run the following command:

```
curl -s https://laravel.build/devdojo-sail | bash
```

This will install a new Laravel application in a `devdojo-sail` folder. Note that you can change the `devdojo-sail` with the name of the folder that you would like to use.

You will see the following output:



Another great thing about it is how easy it is to set up. By default, Sail commands are invoked using the `vendor/bin/sail` script that is included with all new Laravel applications, so to make life easier let's configure the Bash alias that allows us to execute Sail's commands faster.

```
alias sail='bash vendor/bin/sail'
```

What this does is it sets `sail` to point to the `vendor/bin/sail` script, so that you can execute Sail commands without having to type `vendor/bin` each time.

Laravel Sail Commands

Now by simply typing `sail`, you may execute Sail commands. So let's start up our Sail!

One important thing to keep in mind is that before starting it up make sure that **no** other web servers or databases are up and running on your local computer.

To start all of the Docker containers defined in your application's `docker-compose.yml` file, just type in the `up` command:

```
sail up
```

This will start pulling all of the necessary Docker images and it will setup a full blown development environment on your machine:



Another thing that you could do is to start the Docker containers in the background by just adding `-d` after the `up` command. The `-d` stands for **detached**.

```
sail up -d
```

Note that if you run this initially it will take some time to get everything prepared, but after that it should be much quicker



Once you've ran the `sail up` command you can visit your server IP address or `localhost` if you are running it on your Laptop, and you will

see a brand new Laravel installation.

Also if you run `docker ps -a` you will be able to see all of the Docker containers that have been started automatically thanks to Laraval Sail in order to get your Laravel Development environment working:



In case that you want to stop the running containers, you can press `Control + C`, or if you are running the containers in the background, type in the `down` command.

```
sail down
```

You can also execute PHP, Composer, Artisan, and Node/NPM commands. For example, if you want to run a PHP command it should look something like this:

```
sail php --version
```

Or if you wanted to run some Laravel Artisan commands you would usually run `php artisan migrate` but with Laravel sail it would be be a matter of adding `sail` before the artisan command:

```
sail artisan migrate
```

Video Introduction

Make sure to check out the official introduction video here:

{% youtube mgyo0kfV7Vg %}

Conclusion

Laravel Sail will definitely make your work on your new Laravel application be much more enjoyable and more comfortable.

In case that you get a brand new server or a Laptop, all that you would need to get started with Laravel is Docker and Docker compose installed. Then thanks to Laravel Sail you will not have to do any fancy server configuration or install any additional packages on your Laptop like PHP, MySQL, Nginx, Composer and etc. It all comes out of the box with Sail!

Be sure to check out the documentation on Sail to understand it even more:

<https://laravel.com/docs/8.x/sail#executing-artisan-commands>

[Originally posted here](#)

How to add simple search to your Laravel blog/website

There are many ways of adding search functionality to your Laravel website.

For example, you could use [Laravel Scout](#) which is an official Laravel package, you can take a look at this tutorial here on [how to install, setup and use Laravel scout with Algolia](#).

However, in this tutorial here, we will focus on building a very simple search method without the need of installing additional packages.

Before you begin you need to have Laravel installed.

If you do not have that yet, you can follow the steps on how to do that [here](#) or watch this video tutorial on [how to deploy your server and install Laravel from scratch](#).

You would also need to have a model ready that you would like to use. For example, I have a small blog with a `posts` table and `Post` model that I would be using.

Controller changes

First, create a controller if you do not have one already. In my case, I will name the controller **PostsController** as it would be responsible for handling my blog posts.

To create that controller just run the following command:

```
php artisan make:controller PostsController
```

Then open your controller with your text editor, and add the following **search** method:

```
public function search(Request $request){
    // Get the search value from the request
    $search = $request->input('search');

    // Search in the title and body columns from the posts
    table
    $posts = Post::query()
        ->where('title', 'LIKE', "%{$search}%")
        ->orWhere('body', 'LIKE', "%{$search}%")
        ->get();

    // Return the search view with the results compacted
    return view('search', compact('posts'));
}
```

Note that you would need to change the table names which you would like to search in.

In the example above we are searching in the title and body columns from the posts table.

Route changes

Once our controller is ready, we need to add a new route in the `web.php` file:

```
Route::get('/search/',  
    'PostsController@search')->name('search');
```

We will just use a standard get request and map it to our `search` method in the `PostsController`.

This is a sample from "Laravel Tips and Tricks" by Bobby Iliev.

For more information, [Click here](#).