web development & Framework

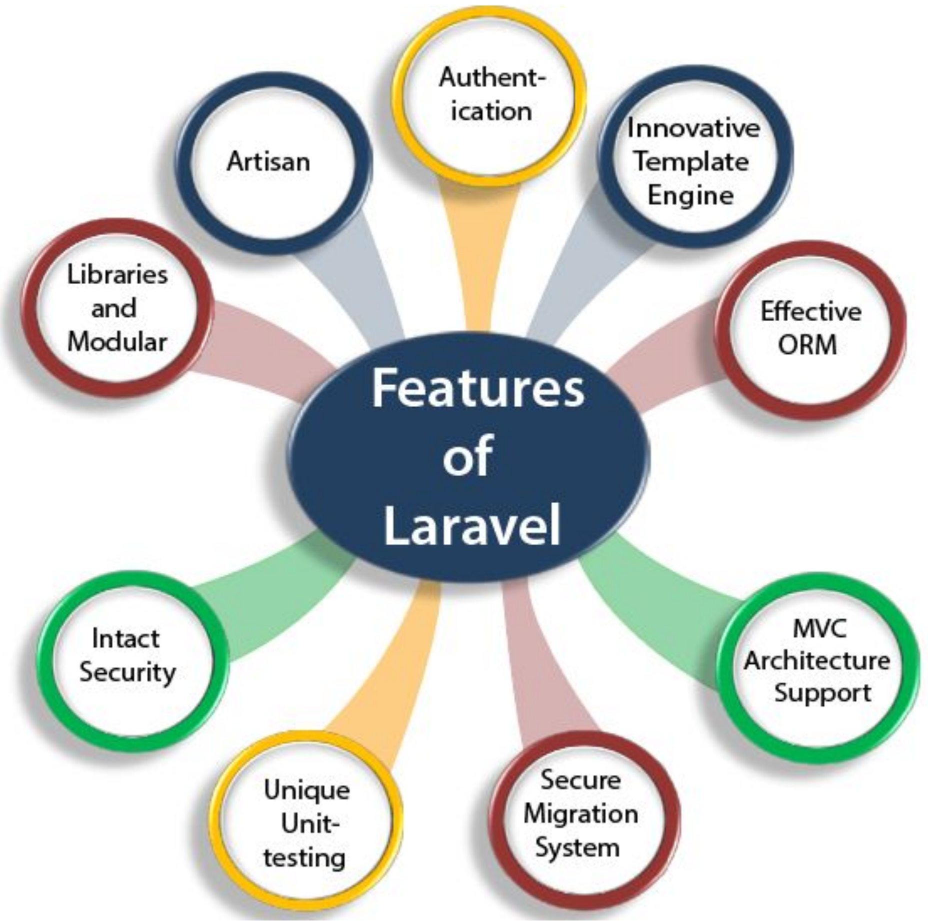
About Laravel

- Created by Taylor Otwell as a free open-source PHP web framework, Laravel is meant to ease and accelerate the development process of web applications with a great taste for simplicity.
- It follows the model–view–controller (MVC) architectural pattern as well as the PSR-2 coding standard, and the PSR-4 autoloading standard.
- Running a Test Driven Development (TDD) in Laravel is fun and easy to implement
- Laravel boasts of a microservices architecture, making it tremendously extendable and this, with ease, with the use of custom-made and or existing third-party packages.

Introduction of Laravel PHP Framework:

- Laravel is a MVC framework with bundles, migrations, and Artisan CLI. Laravel offers a robust set of tools and an application architecture that incorporates many of the best features of frameworks like CodeIgniter, Yii, ASP.NET MVC, Ruby on Rails, Sinatra, and others.
- Laravel is an Open Source framework.
- It has a very rich set of features which will boost the speed of Web Development.
- If you familiar with Core PHP and Advanced PHP, Laravel will make your task easier. It will save a lot time.
- Laravel attempts to take the pain out of development by easing common tasks used in the majority of web projects, such as authentication, routing, sessions, and caching.

Features of Laravel



Authentication

- Authentication is the most important factor in a web application, and developers need to spend a lot of time writing the authentication code.
- Laravel makes a simpler authentication when Laravel is updated to Laravel 5.
- Laravel contains an inbuilt authentication system, you only need to configure models, views, and controllers to make the application work.

Innovative Template Engine

- Laravel provides an innovative template engine which allows the developers to create the dynamic website.
- The available widgets in Laravel can be used to create solid structures for an application.

Effective ORM

- Laravel contains an inbuilt ORM with easy PHP Active Record implementation.
- An effective ORM allows the developers to query the database tables by using the simple PHP syntax without writing any SQL code.
- It provides easy integration between the developers and database tables by giving each of the tables with their corresponding models.

MVC Architecture Support

- Laravel supports MVC architecture.
- It provides faster development process as in MVC, one programmer can work on the view while other is working on the controller to create the business logic for the web application.
- It provides multiple views for a model, and code duplication is also avoided as it separates the business logic from the presentation logic.

Secure Migration System

- Laravel framework can expand the database without allowing the developers to put much effort every time to make changes, and the migration process of Laravel is very secure and full-proof.
- In the whole process, php code is used rather than SQL code.

Unique Unit-testing

- Laravel provides a unique unit-testing.
- Laravel framework can run several test cases to check whether the changes harm the web app or not.
- In Laravel, developers can also write the test cases in their own code.

Intact Security

- Application security is one of the most important factors in web application development.
- While developing an application, a programmer needs to take effective ways to secure the application.
- Laravel has an inbuilt web application security i.e, it itself takes care of the security of an application.
- It uses "Bcrypt Hashing Algorithm" to generate the salted password means that the password is saved as an encrypted password in a database, not in the form of a plain text.

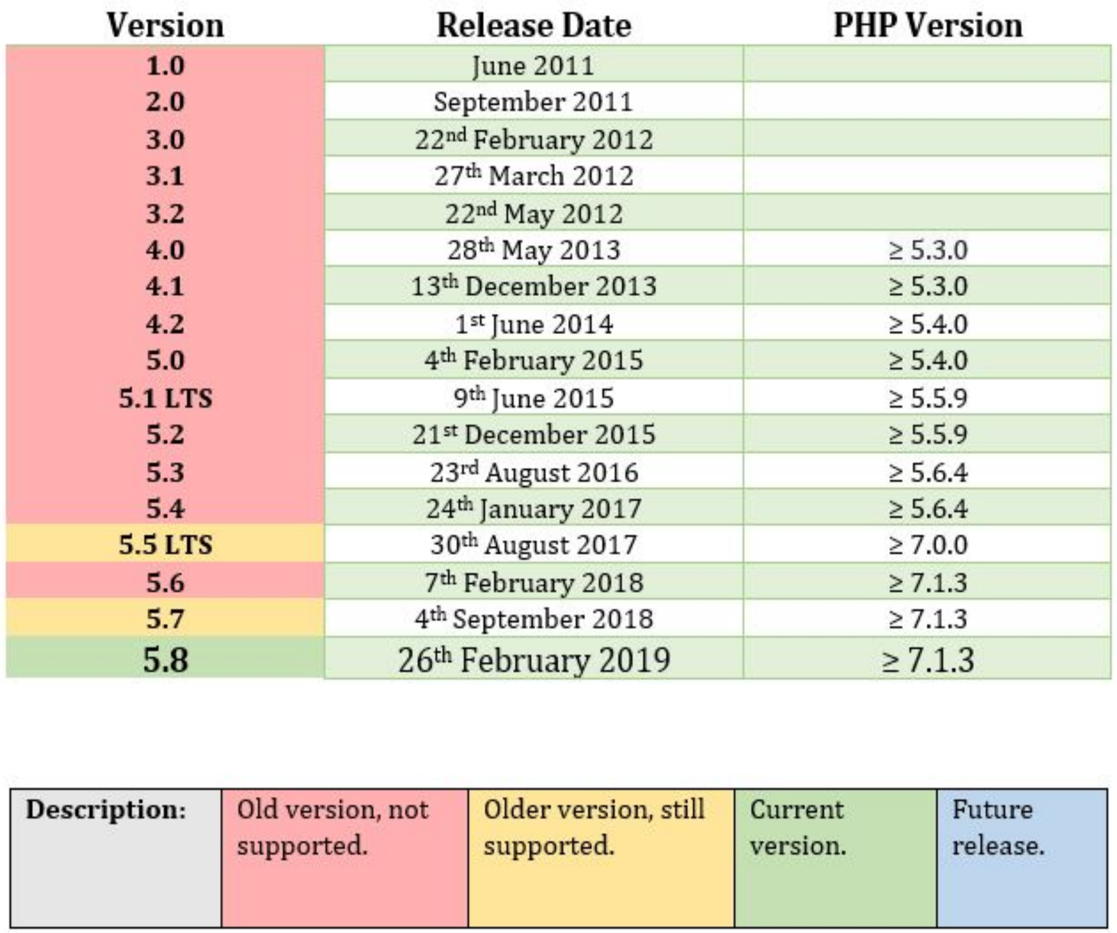
Libraries and Modular

- Laravel is very popular as some Object-oriented libraries, and pre-installed libraries are added in this framework, these pre-installed libraries are not added in other php frameworks.
- One of the most popular libraries is an authentication library that contains some useful features such as password reset, monitoring active users, Bcrypt hashing, and CSRF protection.
- This framework is divided into several modules that follow the php principles allowing the developers to build responsive and modular apps.

Artisan

- Laravel framework provides a built-in tool for a command-line known as Artisan that performs the repetitive programming tasks that do not allow the php developers to perform manually.
- These artisans can also be used to create the skeleton code, database structure, and their migration, so it makes it easy to manage the database of the system.
- It also generates the MVC files through the command line. Artisan also allows the developers to create their own commands.

Versions



Version	Release Date	PHP Version
🟥 1.0	June 2011	
🟥 2.0	September 2011	
🟥 3.0	22nd February 2012	
🟥 3.1	27th March 2012	
🟥 3.2	22nd May 2012	
🟥 4.0	28th May 2013	≥ 5.3.0
🟥 4.1	13th December 2013	≥ 5.3.0
🟥 4.2	1st June 2014	≥ 5.4.0
🟥 5.0	4th February 2015	≥ 5.4.0
🟥 5.1 LTS	9th June 2015	≥ 5.5.9
🟥 5.2	21st December 2015	≥ 5.5.9
🟥 5.3	23rd August 2016	≥ 5.6.4
🟥 5.4	24th January 2017	≥ 5.6.4
🟨 5.5 LTS	30th August 2017	≥ 7.0.0
🟥 5.6	7th February 2018	≥ 7.1.3
🟨 5.7	4th September 2018	≥ 7.1.3
🟩 5.8	26th February 2019	≥ 7.1.3

Description:	🟥 Old version, not supported.	🟨 Older version, still supported.	🟩 Current version.	🟦 Future release.

Advantages of Laravel

- Laravel offers you the following advantages, when you are designing a web application based on it
- The web application becomes more scalable, owing to the Laravel framework.
- Considerable time is saved in designing the web application, since Laravel reuses the components from other framework in developing web application.
- It includes namespaces and interfaces, thus helps to organize and manage resources.

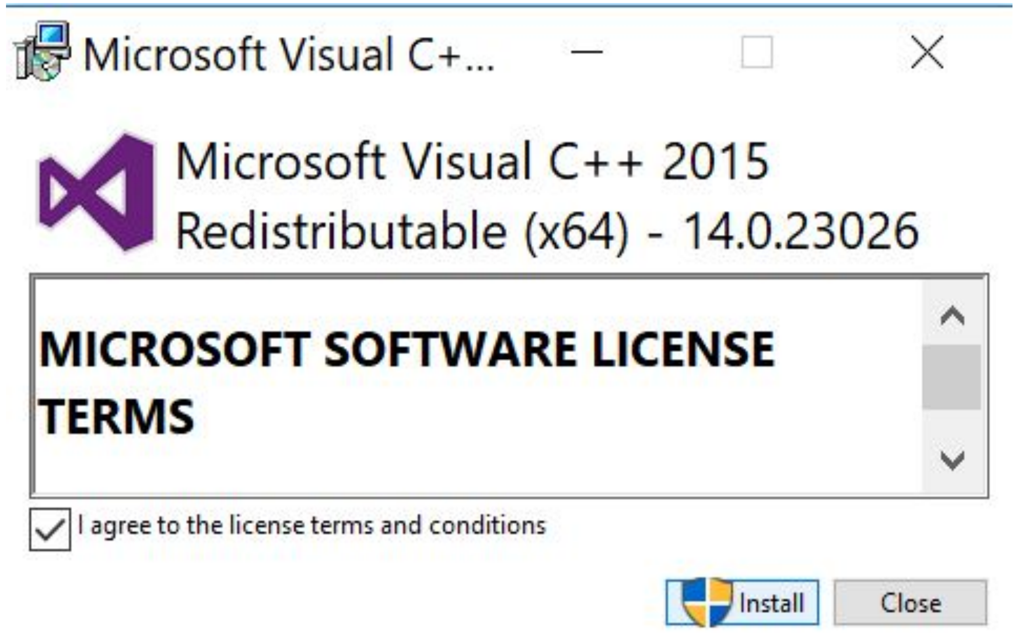
Installing Laravel

Prerequisites:

- Basic knowledge of PHP
- PHP Installation
- Basic knowledge of Terminal/Command Prompt

Steps to install Composer and PHP

- you will need to install *Visual C++ Redistributable for Visual Studio*
- Download: **VC_redist.x64.exe (64 bit)** or VC_redist.x86.exe (32 bit)
- Once downloaded, run the installer and click **Install**.



Microsoft Visual C+...

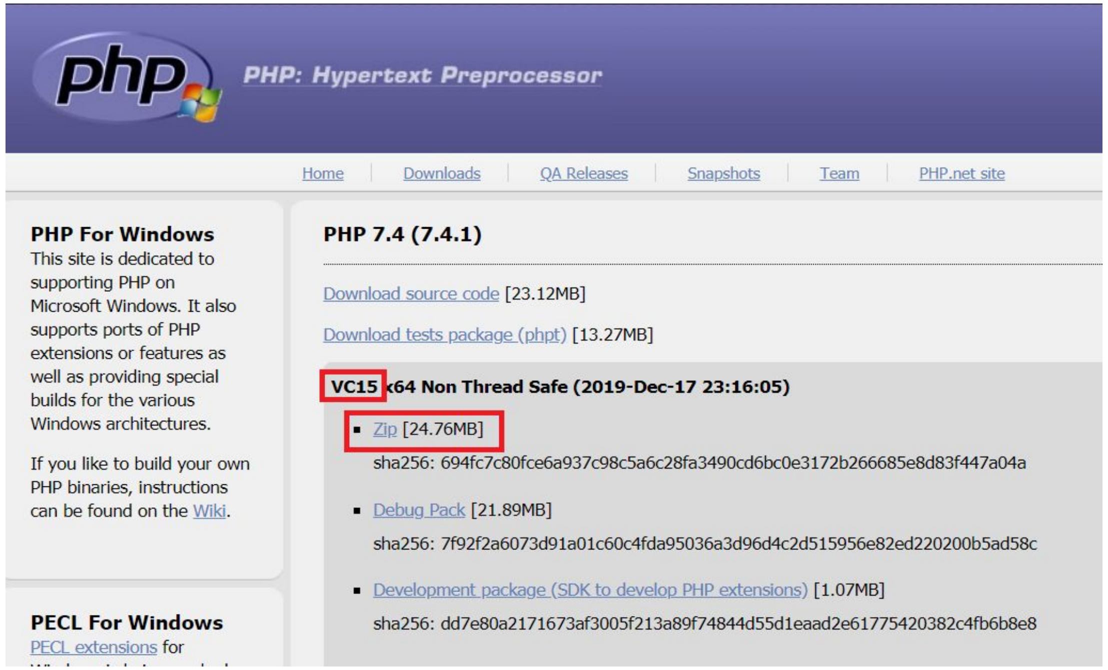
Microsoft Visual C++ 2015
Redistributable (x64) - 14.0.23026

MICROSOFT SOFTWARE LICENSE TERMS

☑ I agree to the license terms and conditions

Download and Extract PHP

- Go to the Windows PHP download page and download the .zip of the version of PHP you require.
- In this guide, we are installing **PHP 8.4 VC15 x64 Non Thread Safe**, which is the latest version as of writing



PHP For Windows
This site is dedicated to supporting PHP on Microsoft Windows. It also supports ports of PHP extensions or features as well as providing special builds for the various Windows architectures.

If you like to build your own PHP binaries, instructions can be found on the Wiki.

PECL For Windows
PECL extensions for

PHP 7.4 (7.4.1)

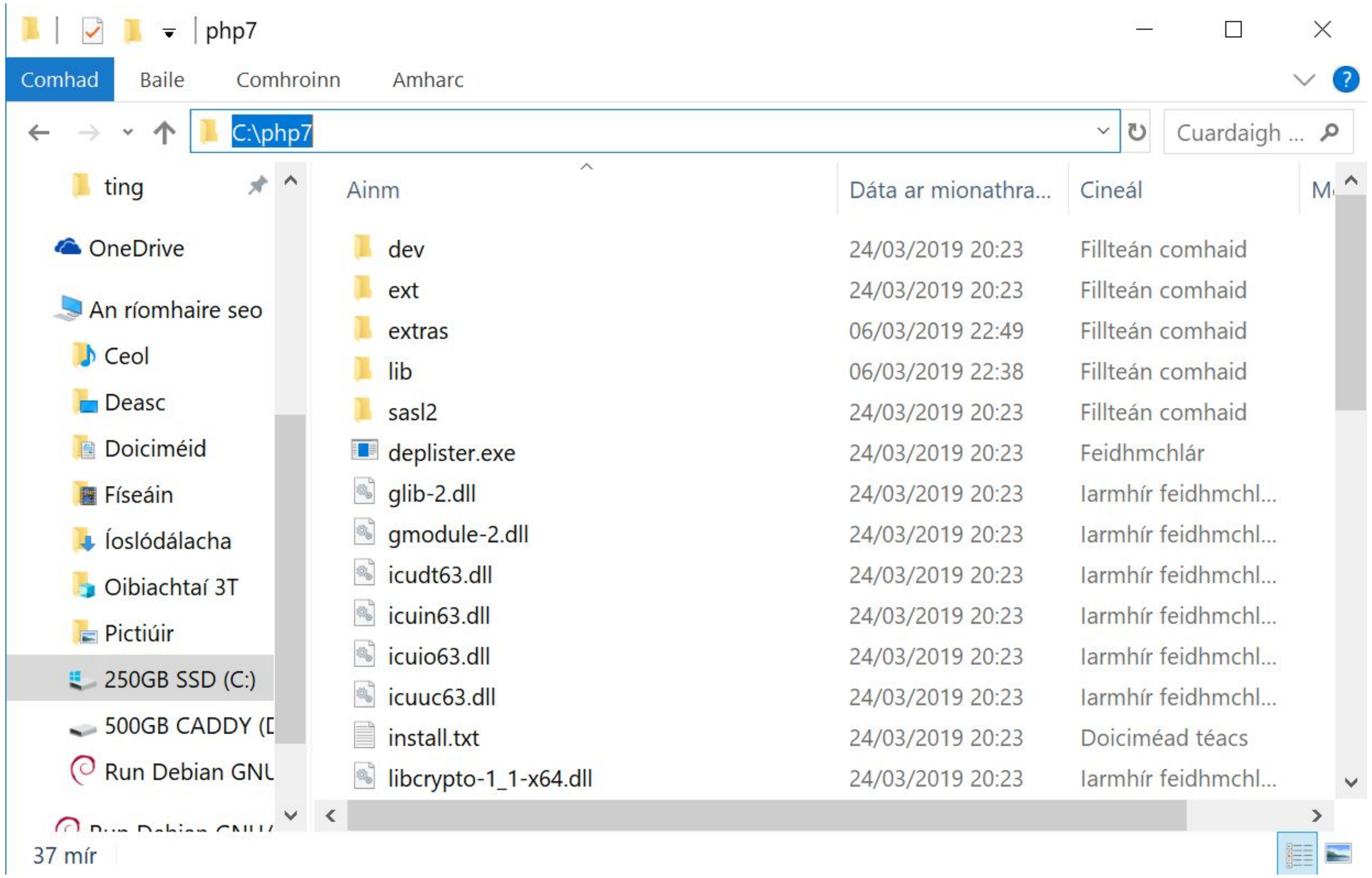
Download source code [23.12MB]

Download tests package (phpt) [13.27MB]

VC15 x64 Non Thread Safe (2019-Dec-17 23:16:05)

- Zip [24.76MB]
  sha256: 694fc7c80fce6a937c98c5a6c28fa3490cd6bc0e3172b266685e8d83f447a04a
- Debug Pack [21.89MB]
  sha256: 7f92f2a6073d91a01c60c4fda95036a3d96d4c2d515956e82ed220200b5ad58c
- Development package (SDK to develop PHP extensions) [1.07MB]
  sha256: dd7e80a2171673af3005f213a89f74844d55d1eaad2e61775420382c4fb6b8e8

- Once downloaded, create a new folder in c:\php7 (or wherever you prefer) and extract your PHP zip to it.



php7

Comhad Baile Comhroinn Amharc

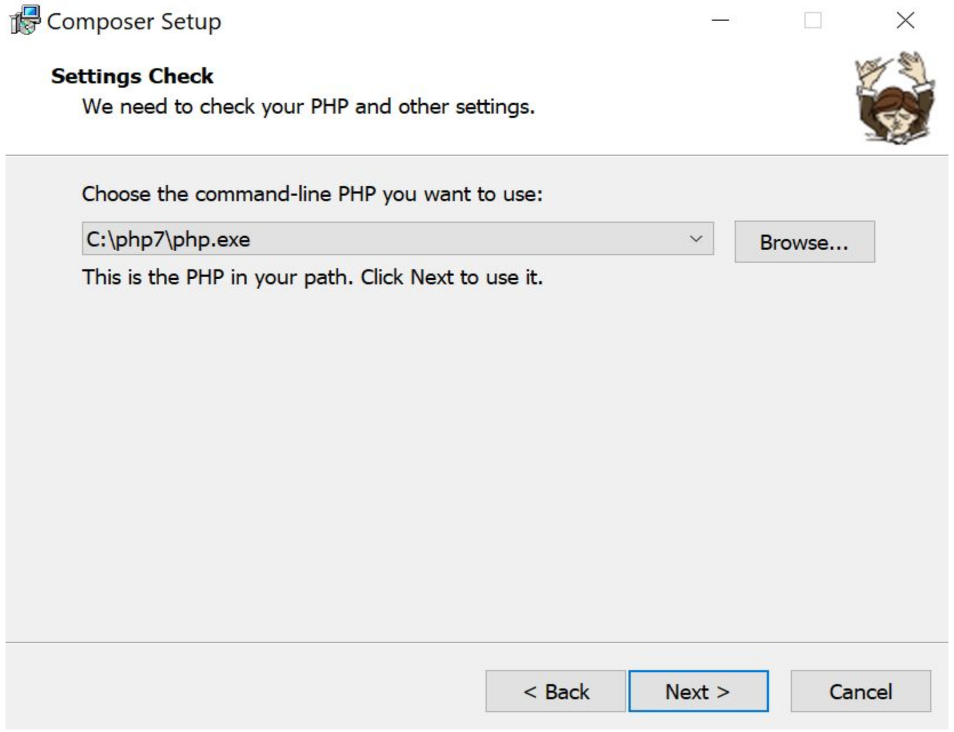
C:\php7

Cuardaigh ...

Ainm	Dáta ar mionathra...	Cineál
dev	24/03/2019 20:23	Fillteán comhaid
ext	24/03/2019 20:23	Fillteán comhaid
extras	06/03/2019 22:49	Fillteán comhaid
lib	06/03/2019 22:38	Fillteán comhaid
sasl2	24/03/2019 20:23	Fillteán comhaid
deplister.exe	24/03/2019 20:23	Feidhmchlár
glib-2.dll	24/03/2019 20:23	Iarmhír feidhmchl...
gmodule-2.dll	24/03/2019 20:23	Iarmhír feidhmchl...
icudt63.dll	24/03/2019 20:23	Iarmhír feidhmchl...
icuin63.dll	24/03/2019 20:23	Iarmhír feidhmchl...
icuio63.dll	24/03/2019 20:23	Iarmhír feidhmchl...
icuuc63.dll	24/03/2019 20:23	Iarmhír feidhmchl...
install.txt	24/03/2019 20:23	Doiciméad téacs
libcrypto-1_1-x64.dll	24/03/2019 20:23	Iarmhír feidhmchl...

Download and Install Composer

- Download Composer-Setup.exe from the Composer Download page.
- In the second step of Composer Setup, make sure the correct path to php.exe is set and click Next.


Composer Setup

Settings Check

We need to check your PHP and other settings.

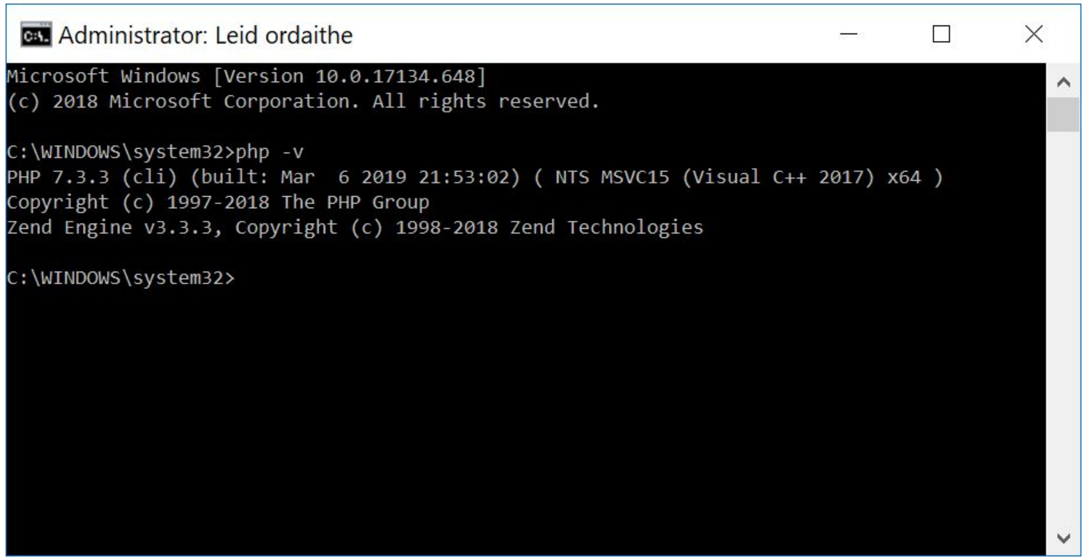
Choose the command-line PHP you want to use:

C:\php7\php.exe

This is the PHP in your path. Click Next to use it.

Test PHP

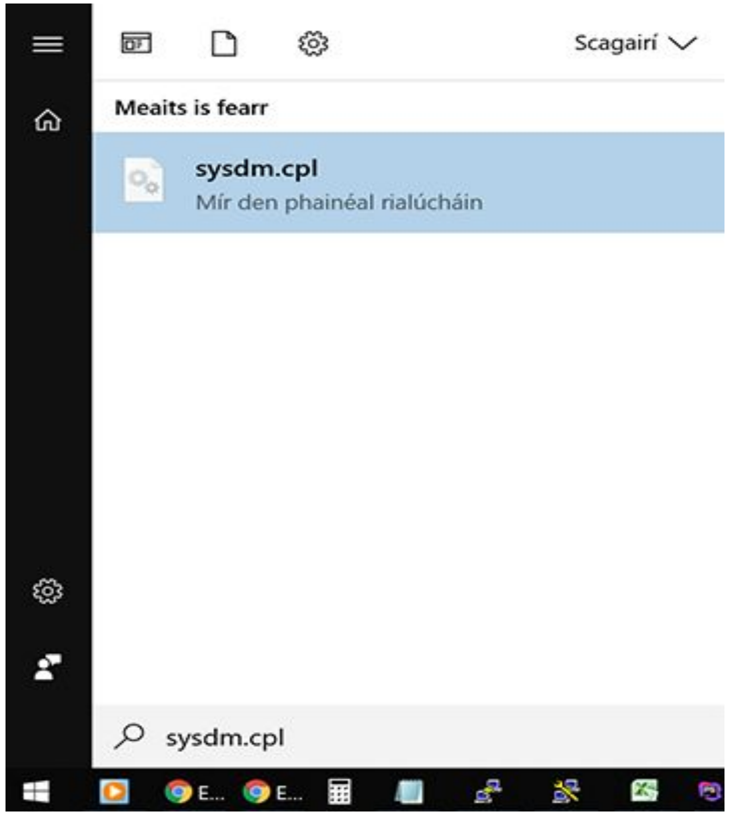
- If you have any Command Prompt windows currently open, close them now.
- Open Command Prompt, type php -v and press Enter. You should now see the PHP version.
- If it returns a PHP version, you can skip to Step 6 to test Composer.
- If you get an error *"'php' is not recognized as an internal or external command, operable program or batch file."*, you may need to Add a Path Environment Variable.



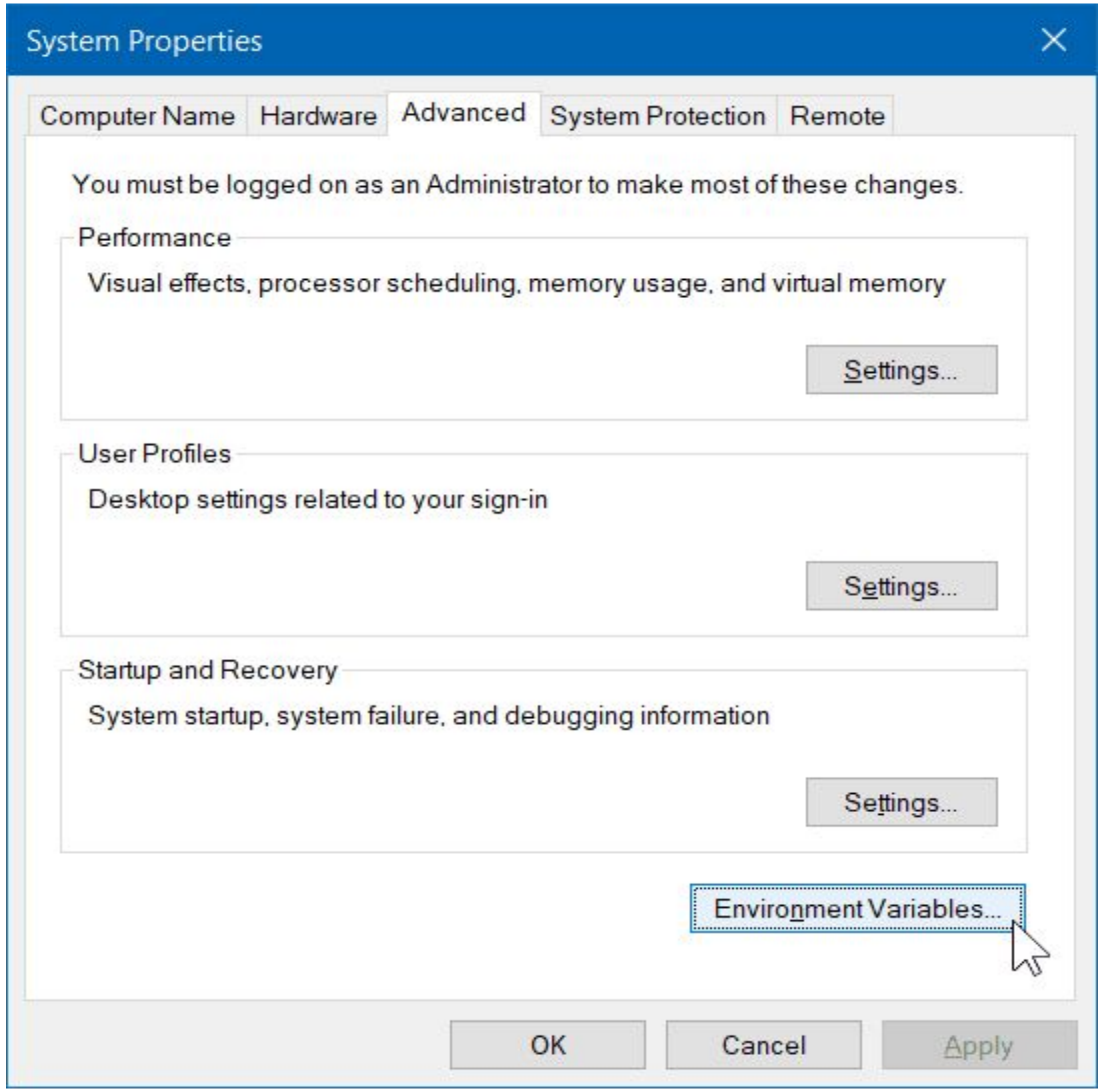
```
Microsoft Windows [Version 10.0.17134.648]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>php -v
PHP 7.3.3 (cli) (built: Mar  6 2019 21:53:02) ( NTS MSVC15 (Visual C++ 2017) x64 )
Copyright (c) 1997-2018 The PHP Group
Zend Engine v3.3.3, Copyright (c) 1998-2018 Zend Technologies

C:\WINDOWS\system32>
```

Add Path Environment Variable

- So that you don't have to type the whole path to php.exe every time you run a PHP command, you should add c:\php7 as a path environment variable.
- Open System Properties by clicking the start menu and typing sysdm.cpl and press Enter.



- In **System Properties**, click **Environment Variables**.


System Properties

Computer Name | Hardware | Advanced | System Protection | Remote

You must be logged on as an Administrator to make most of these changes.

Performance

Visual effects, processor scheduling, memory usage, and virtual memory

Settings...

User Profiles

Desktop settings related to your sign-in

Settings...

Startup and Recovery

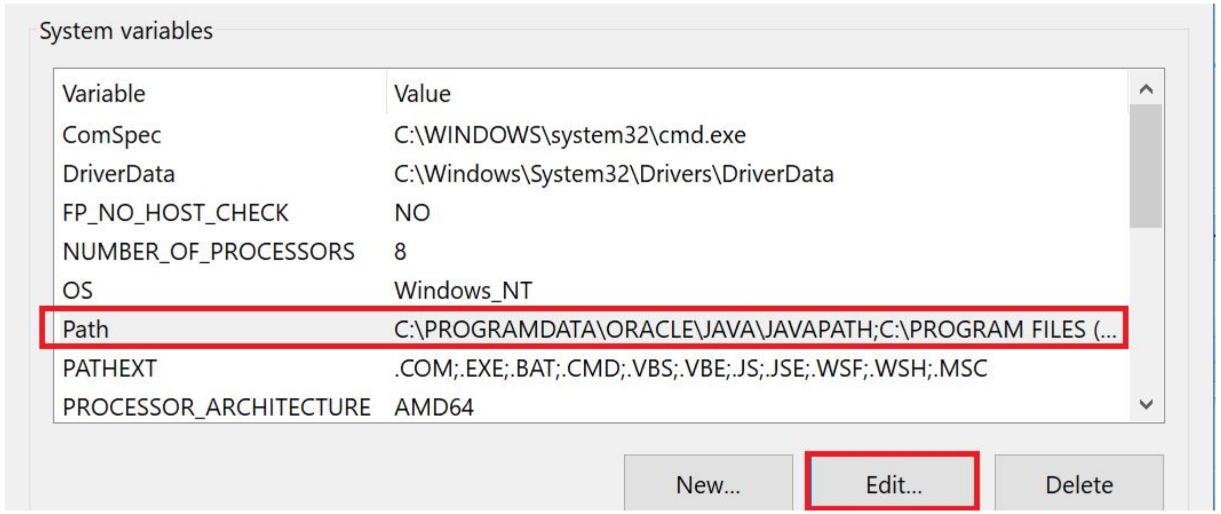
System startup, system failure, and debugging information

Settings...

Environment Variables...

OK | Cancel | Apply

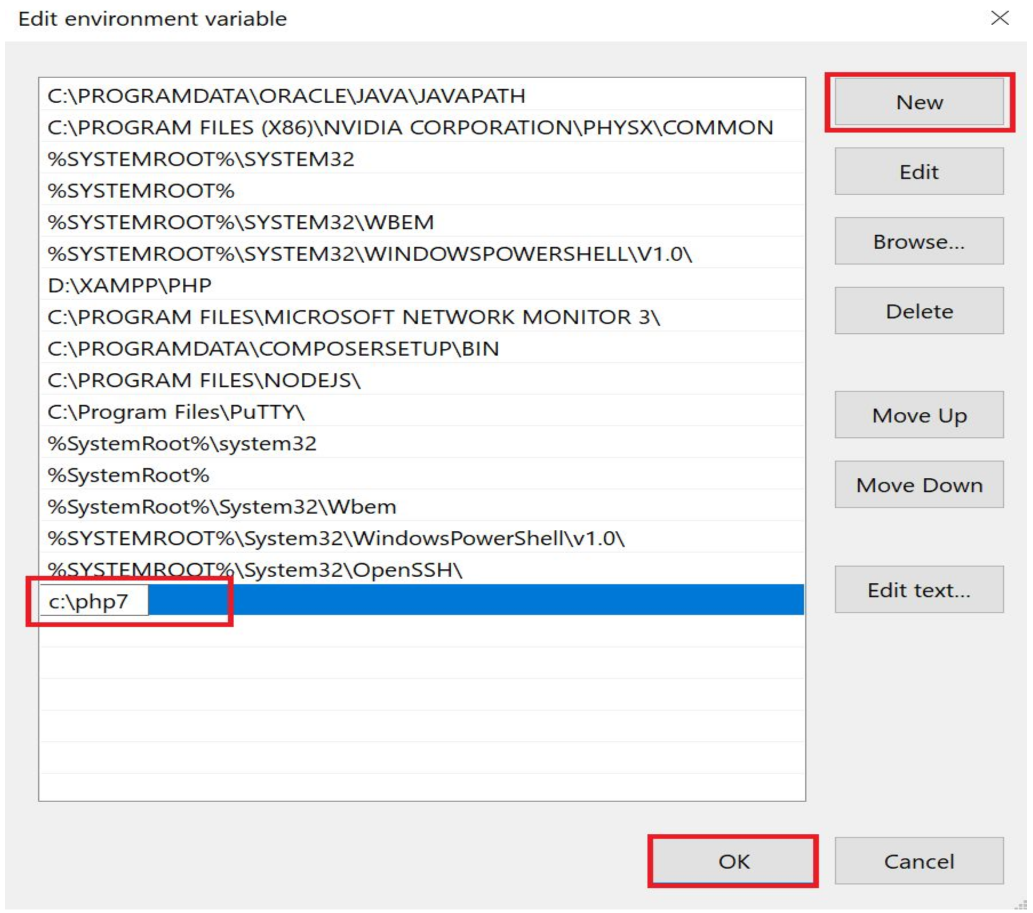
- In **System Variables**, click **Path** and then click **Edit**.



System variables

Variable	Value
ComSpec	C:\WINDOWS\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData
FP_NO_HOST_CHECK	NO
NUMBER_OF_PROCESSORS	8
OS	Windows_NT
Path	C:\PROGRAMDATA\ORACLE\JAVA\JAVAPATH;C:\PROGRAM FILES (...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE	AMD64

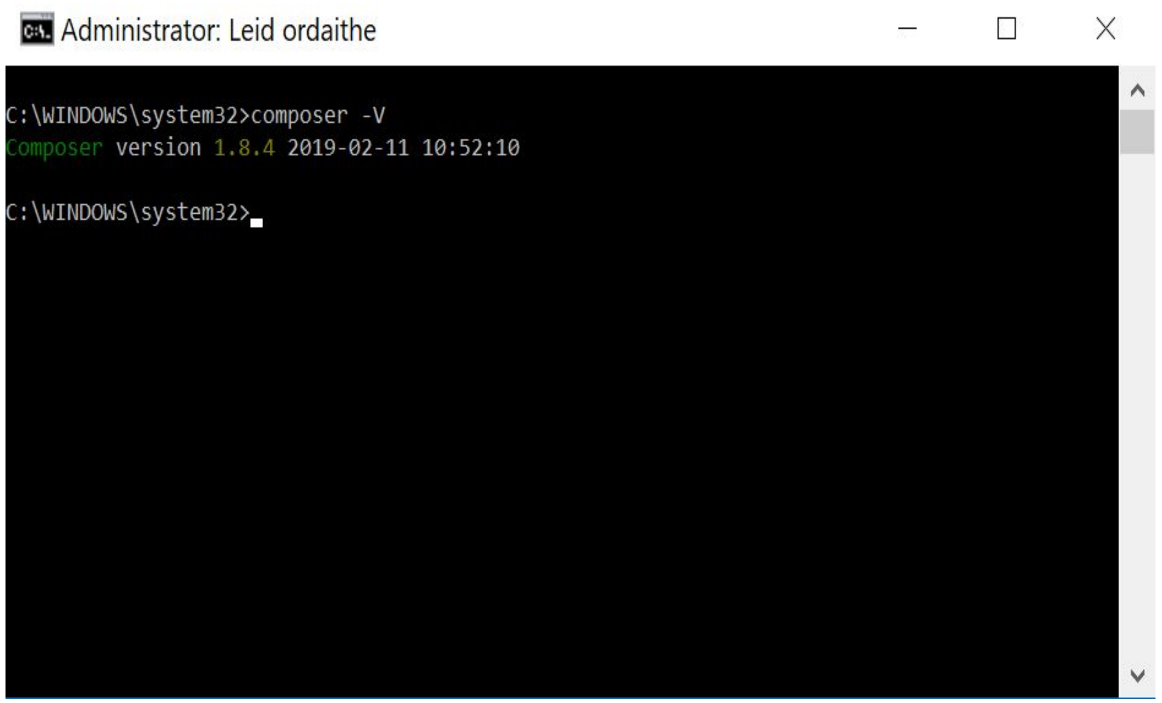
- Click New, type the path to your PHP folder (c:\php7) and click OK.


Edit environment variable

```
C:\PROGRAMDATA\ORACLE\JAVA\JAVAPATH
C:\PROGRAM FILES (X86)\NVIDIA CORPORATION\PHYSX\COMMON
%SYSTEMROOT%\SYSTEM32
%SYSTEMROOT%
%SYSTEMROOT%\SYSTEM32\WBEM
%SYSTEMROOT%\SYSTEM32\WINDOWSPOWERSHELL\V1.0\
D:\XAMPP\PHP
C:\PROGRAM FILES\MICROSOFT NETWORK MONITOR 3\
C:\PROGRAMDATA\COMPOSERSETUP\BIN
C:\PROGRAM FILES\NODEJS\
C:\Program Files\PuTTY\
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
c:\php7
```

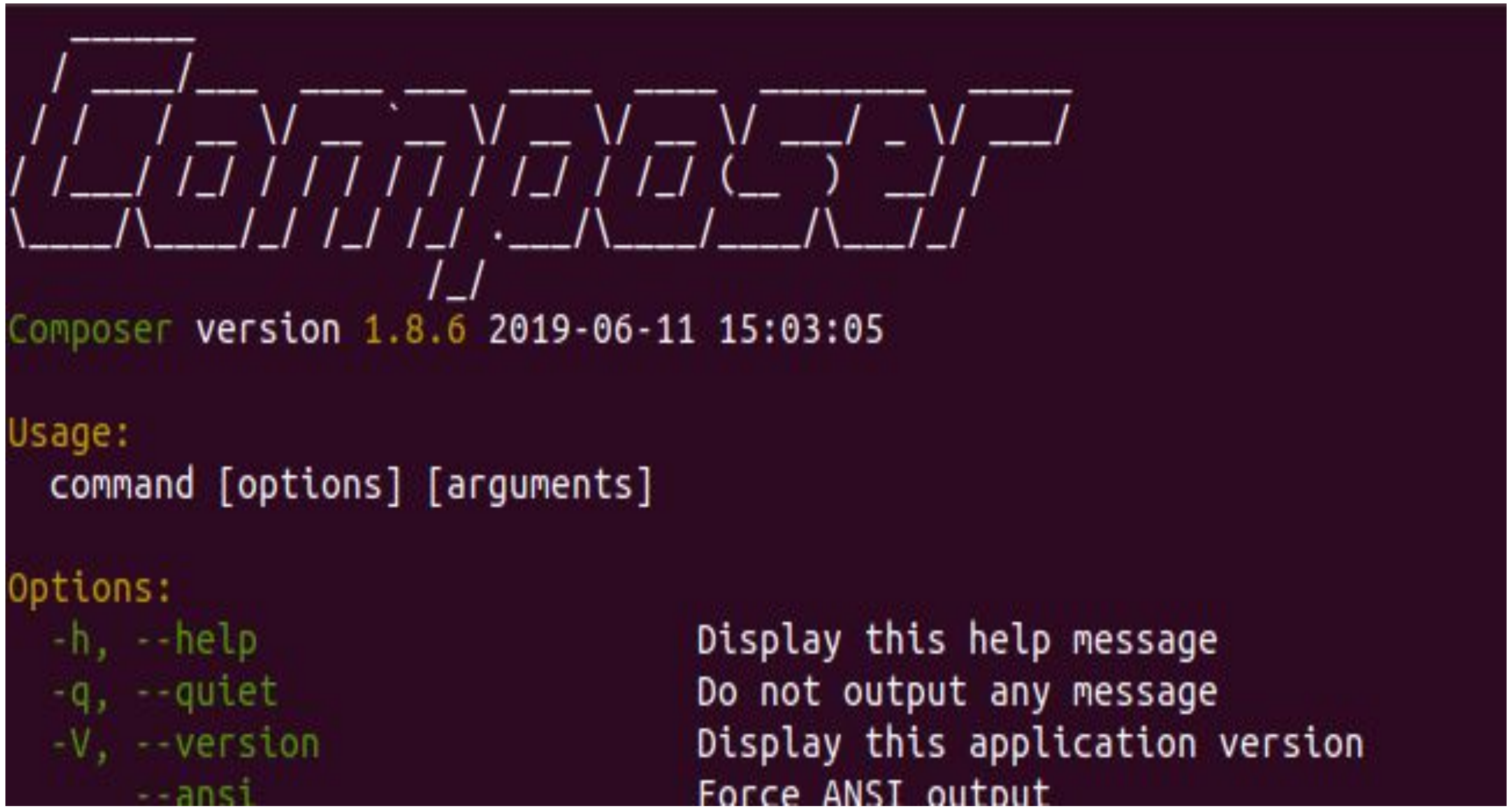
Test Composer

- Open up Command Prompt and type composer –V.
- If all was installed correctly, you should see a version number.



```
C:\WINDOWS\system32>composer -V
Composer version 1.8.4 2019-02-11 10:52:10

C:\WINDOWS\system32>
```

- If it gives an output like given below then it means the installation was successful.



```
   ______
  / ____/___  ____ ___  ____  ____  ________  _____
 / /   / __ \/ __ `__ \/ __ \/ __ \/ ___/ _ \/ ___/
/ /___/ /_/ / / / / / / /_/ / /_/ (__  )  __/ /
\____/\____/_/ /_/ /_/ .___/\____/____/\___/_/
                    /_/
Composer version 1.8.6 2019-06-11 15:03:05

Usage:
  command [options] [arguments]

Options:
  -h, --help                     Display this help message
  -q, --quiet                    Do not output any message
  -V, --version                  Display this application version
      --ansi                     Force ANSI output
```

Tools Include

- PHP >= 7.3.
- Database (MySql).
- A localhost Web Server – In our case we'll use WAMP (for Windows), LAMP (for Linux), or MAMP (for MacOs). This localhost webserver comes installed with latest PHP and MySQL database so you will not need to install them manually. To install either MAMP, LAMP, or WAMP go to http://ampps.com/downloads and choose the software your platform.
- Composer – This is a dependency management software for PHP. To install the composer visit https://getcomposer.org/ and download it there for your platform.
- Node.js – This is a free and open source JavaScript runtime environment that executes JavaScript outside of the browser. We will not write any Node.js code but it will be used in the background by Laravel to streamline our development.
- Code editor – A code editor will be required. We recommend to use Visual Studio Code: It is free.
- A browser – Google Chrome, Edge, Safari, or Mozilla Firefox will do just fine.
- Background knowledge of the PHP Programming Language.
- With our machine setup complete, it's time to start developing.

ARTISAN(Command-line interface)

- ARTISAN is a command-line interface that Laravel provides which helps in making the production process fast and easy.
- Laravel has its own Command Line interface called Artisan.
- Its like a Linux command line but the commands are helpful for building a Laravel application.
- With this command-line tool, we can make models, controllers, and can do data migrations and many more.
- First, we will have to change the directory in your command line console (i.e. cmd on windows or terminal on Linux/Mac) or any other CLI software, to the directory of your Laravel app.

Types of Commandd Lines

For Controller:

- The following command will create a controller:

```
php artisan make:controller ArticleController
```

For Eloquent Model:

- The following command will create a eloquent model:

```
php artisan make:model Article
```

For Front-end Scaffolding:

- The following command will create a front-end scaffolding for the for Bootstrap:

```
php artisan ui bootstrap
```

For Authentication Configuration:

- The following command will create a full authentication system:

```
php artisan ui vue –auth
```

For Migration:

- The following command will create a migration:

```
php artisan make:migration create_articles_table
```

For Route:

- The following command will display list of all the routes:

```
php artisan route:list
```

For Tinker:

- The following command will start tinker:

```
php artisan tinker
```

For Starting Development Server:

- The following command will start the Laravel development server and provide a URL to visit the running Laravel application:

```
php artisan serve
```

For Maintenance Mode:

- The following command can be used to take the Laravel application in or out of the Maintenance Mode:

```
php artisan down
```

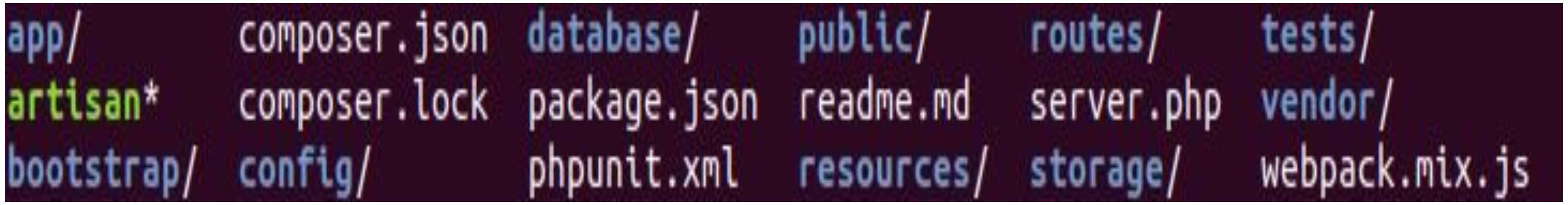
For Listing Commands:

- The following command will display a list of all the command that are available:

```
php artisan list
```

Laravel Directory Structure

- When you will create your fresh Laravel application, it will contain a large number of folders as shown in image below:



```
app/        composer.json  database/     public/     routes/     tests/
artisan*    composer.lock  package.json  readme.md   server.php  vendor/
bootstrap/  config/        phpunit.xml   resources/  storage/    webpack.mix.js
```

- Each of these folders fulfills a specific task for the overall functioning of the framework.
- The purpose of each of these folders is explained below but before that let’s look at each of them once:

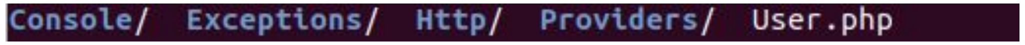
Types Of Directory Structure

1. app directory
2. bootstrap directory
3. config directory
4. database directory
5. public directory
6. resources directory
7. routes directory
8. storage directory
9. tests directory
10. vendor directory

Purpose of each of these directories:

1. app directory:

This directory is the heart of the framework and backend developers mostly work on this directory. It contains all the backend code of our web application like Controllers, Broadcasts, Providers, Custom Artisan Commands, Middlewares, etc. This directory further contains many sub-directories like shown in the image below:



```
Console/  Exceptions/  Http/  Providers/  User.php
```

2. bootstrap directory:

This directory contains app.php from where the whole framework bootstraps. This directory also contains the cache directory which is used to store framework generated files for performance optimization.

3. config directory:

This directory contains all the configuration files related to database, mail, session, services, etc.

4. database directory:

This directory contains database migrations, model factories, and seeds.

5. public directory:

This directory contains the index.php file which is the entry point and handles all requests received by the application and configures autoloading too. Apart from this, this directory also contains assets used in the application like images, javaScript, and CSS.

6. resources directory:

This directory contains the frontend of the application. All the HTML code which makes the frontend of application is present here in the form of Blade templates which is a templating engine Laravel comes with.

7. routes directory:

This directory contains all the route definitions of the application.

8. storage directory:

This directory contains the compiled Blade templates, file based sessions, file caches, and other files generated by framework.

9. tests directory:

This directory contains all of our automated tests which are required to ensure that the application is working as per expectations or not.

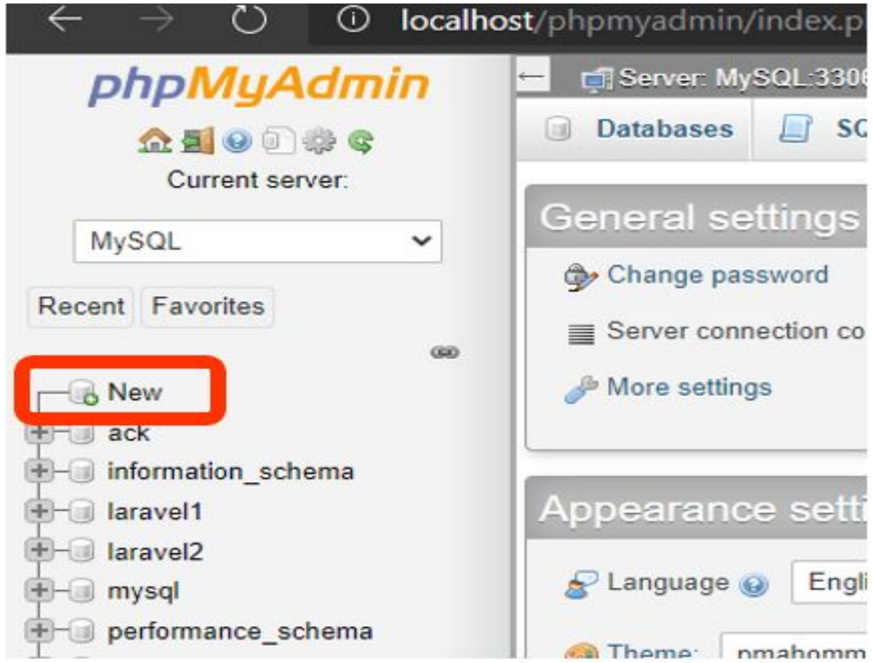
10. vendor directory:

This directory contains all the dependencies downloaded through Composer needed by our framework.

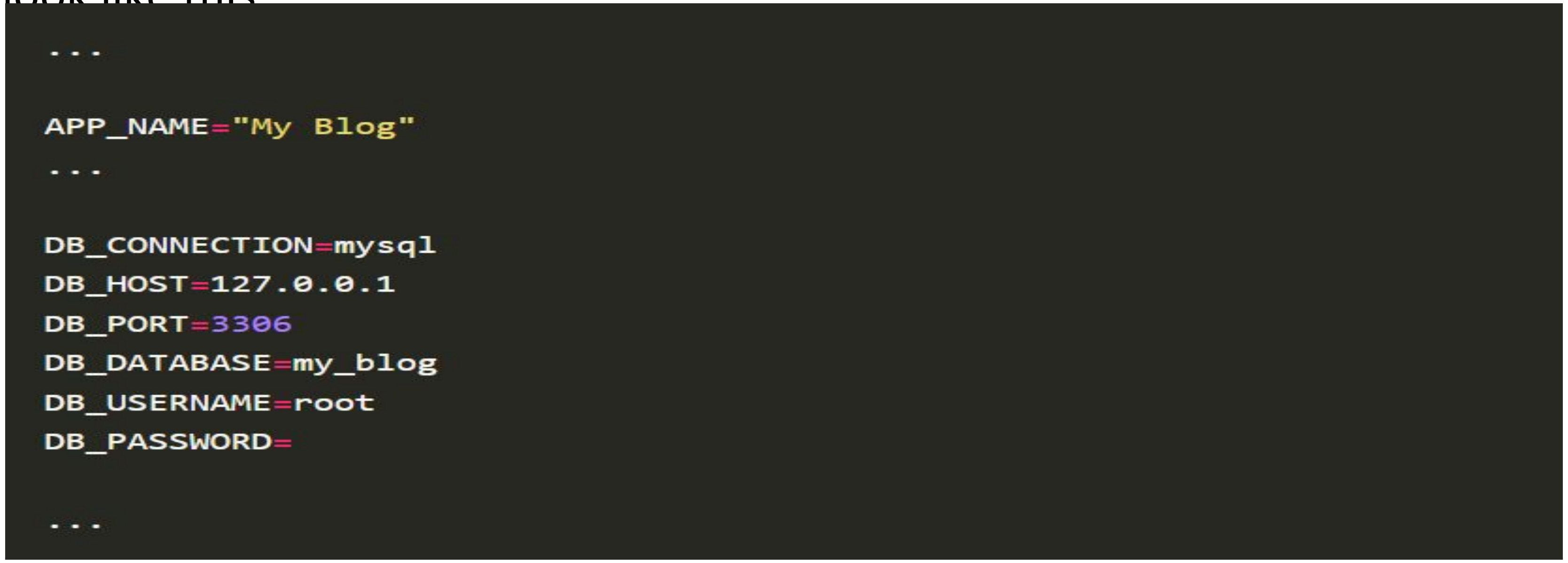
Directory	Purpose
Console	This directory contains all Artisan commands which are created by us. These commands can be generated using the php artisan make:command command.
Exceptions	This directory contains the application’s exception handling files. Here you can create your own specific exceptions to be thrown by our application.
Http	This directory contains our controllers, middleware, and form requests. Almost all of the backend to handle requests entering our application will be placed here.
Providers	This directory contains all of the service providers for the application. Service providers bootstrap our application by making services available to us by registering them.
Broadcasting	This directory is not there by default but can be created by using the php artisan make:channel command. It contains all of the broadcast channel classes for our application to broadcast your events.
Events	This directory is not there by default but can be created by using the php artisan make:event command. This directory contains event classes which can be used to give signals to other parts of the application or vice-versa.
Jobs	This directory is not there by default but can be created by using the php artisan make:job command. This directory contains lineup jobs for our application.
Listeners	This directory is not there by default but can be created by using the php artisan make:listener command. This directory contains the classes that handle our events.
Mail	This directory is not there by default but can be created by using the php artisan make:mail command. This directory contains all of our classes that represent emails sent by application.
Notifications	This directory is not there by default but can be created by using the php artisan make:notification command. This directory contains all of the “transactional” notifications that are sent by our application.
Policies	This directory is not there by default but can be created by using the php artisan make:policy command. This directory contains the authorization policy classes which are used to determine if a user can access or change a specific data or not.
Rules	This directory is not there by default but can be created by using the php artisan make:rule command. This directory contains the self-created validation rule objects which are used to encapsulate complicated validation logic in a simple object.

Configuring a new Laravel project

- After installing our Laravel application we will need to configure our database for it to work.
- Go to http://localhost/phpmyadmin/
- Create a new Database by clicking on new (shown below in red)



- Name it my_blog and click Create
- Now that we have a database we can proceed to set up the application to work with the database.
- Open your file explorer and navigate to my-blog folder
- Open the .env file in your code editor
- Change the following in the file:-
- APP_NAME key to name of your blog i.e. “My Blog”
- DB_DATABASE key to database name i.e. my_blog
- The final file should look like this:

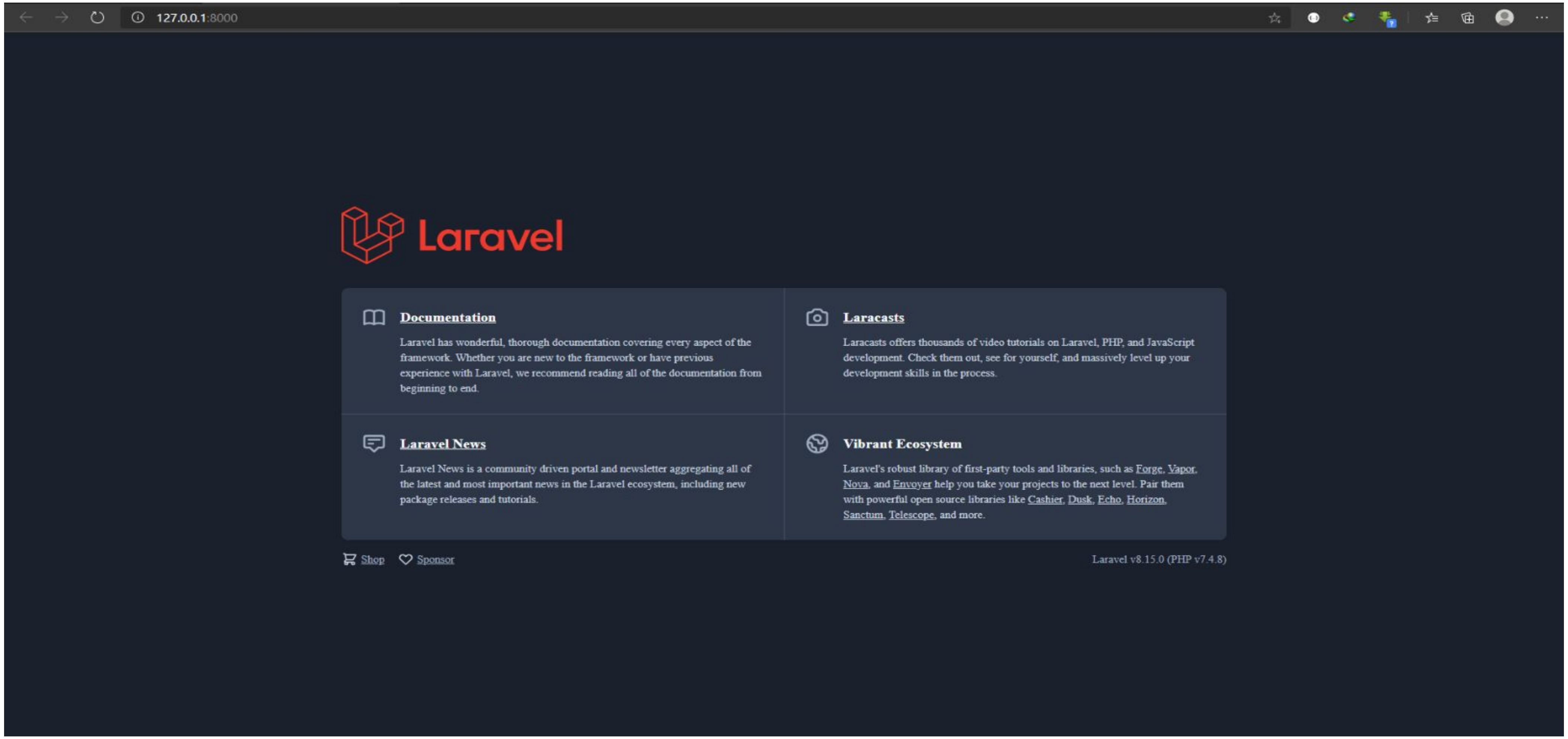


```
...

APP_NAME="My Blog"
...

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=my_blog
DB_USERNAME=root
DB_PASSWORD=

...
```

- With everything configured it’s time to run our app and see what it looks like.
- To run the application, type the following command:

```
php artisan serve
```

- It will start the application and give you a URL, http://127.0.0.1:8000, open the URL in your browser and see the application (shown below).



127.0.0.1:8000

Laravel

Documentation

Laravel has wonderful, thorough documentation covering every aspect of the framework. Whether you are new to the framework or have previous experience with Laravel, we recommend reading all of the documentation from beginning to end.

Laracasts

Laracasts offers thousands of video tutorials on Laravel, PHP, and JavaScript development. Check them out, see for yourself, and massively level up your development skills in the process.

Laravel News

Laravel News is a community driven portal and newsletter aggregating all of the latest and most important news in the Laravel ecosystem, including new package releases and tutorials.

Vibrant Ecosystem

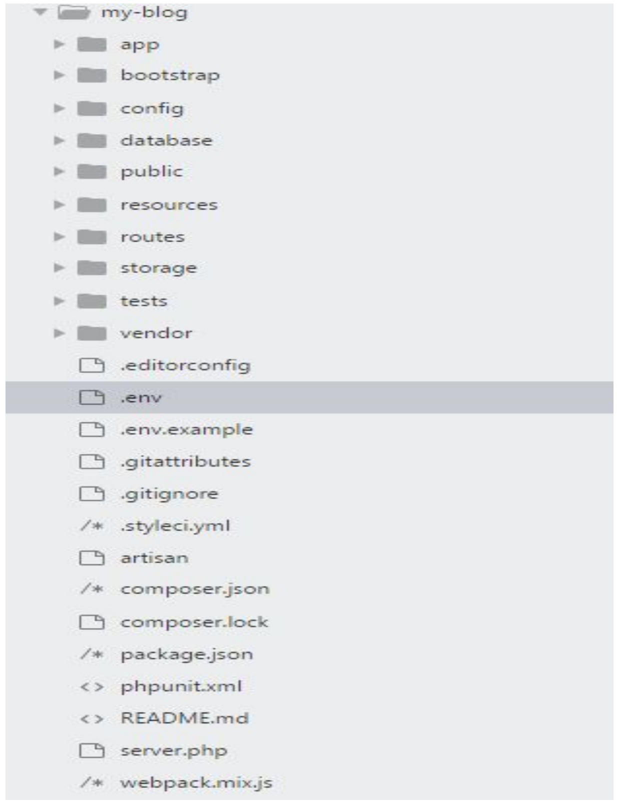
Laravel's robust library of first-party tools and libraries, such as Forge, Vapor, Nova, and Envoyer help you take your projects to the next level. Pair them with powerful open source libraries like Cashier, Dusk, Echo, Horizon, Sanctum, Telescope, and more.

Shop Sponsor

Laravel v8.15.0 (PHP v7.4.8)

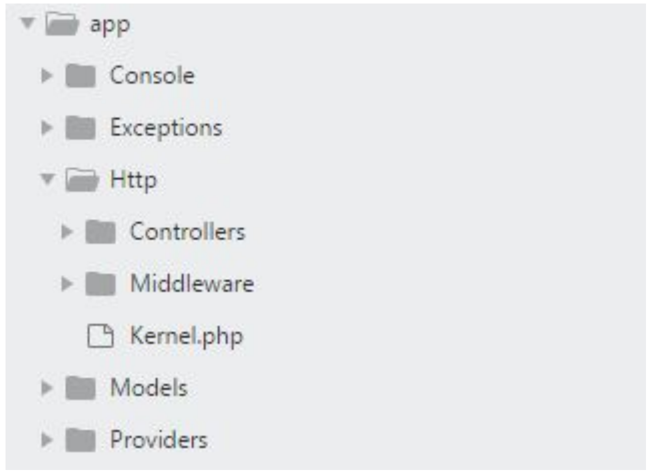
Understanding the Laravel application file structure

- Let’s understand the file structure of a Laravel application.
- For example, most of you may not understand why we changed the .env file.
- Below is the Laravel App file structure:



```
my-blog
  app
  bootstrap
  config
  database
  public
  resources
  routes
  storage
  tests
  vendor
  .editorconfig
  .env
  .env.example
  .gitattributes
  .gitignore
  .styleci.yml
  artisan
  composer.json
  composer.lock
  package.json
  phpunit.xml
  README.md
  server.php
  webpack.mix.js
```

- app folder – This contains all the logic in our application, this includes the models, controllers, service providers etc.



```
▼ app
  ▶ Console
  ▶ Exceptions
  ▼ Http
    ▶ Controllers
    ▶ Middleware
      Kernel.php
  ▶ Models
  ▶ Providers
```

- Models folder- This is where the business logic of your App is stored in, a model is a representation of a real life object. For example, a blog post.
- Models will be generated using php artisan command make:model and follow a convention of singular title case wording. For example, for a blog post model we could call it BlogPost.php.
- Http/Controllers folder – This will contain all the controller files of your application.
- A controller creates a link between your Models and your Views. When a new blog post form is submitted by the user, the data comes into the controller where it’s sanitized and then passed to the model to be stored in the database, then the controller sends feedback back to the view saying the blog post has been created.
- Controllers will be generated using php artisan command make:controller and follow a convention of singular title case wording with the word Controller trailing. For our blog post controller we will call it BlogPostController.php

Types of Controller

- A controller has 7 signature methods that enables crud operations:
- index() – to fetch all the resources e.g. all blog posts available.
- show() – to fetch a single resource e.g. a single blog post, say, post 5.
- create() – shows the form to use to create a resource (not available for API controllers).
- store() – to commit the resource to database e.g. save blog post.
- index() – to show the form to edit the resource (not available for API controllers).
- update() – to commit the edited resource to database.
- destroy() – to delete a resource from database.

Laravel app file structure

- **Http/Middleware folder** – This contains all the middleware, middleware is code that is to be executed before the request gets to the controller e.g. Authenticating a user before allowing access.
- **Exceptions folder** – This contains all the Exception handling in your App, you can add custom exceptions here too.
- **Console folder** – This contains all the PHP artisan commands (PHP Artisan is the command line tool that ships with Laravel to help us design our application faster). These commands are used to create application files and also do some actions like start a development server. An example of artisan command is the one we ran at the beginning after installing Laravel (php artisan serve).
- **Providers folder** – This contains all the service providers in your App, a service provider in Laravel is a group of code that does specific task across the app whenever needed. For example a billing service provider will be designed to allow multiple payment platforms but all you have to do is call the service provider and it will dynamically provide a payment platform instead of specifying a platform in the controller.

- **bootstrap folder** – This contains the app file that bootstraps the framework by initializing it (setting up path & environment), it also contains the cache folder that contains the framework generated files for the optimization of the app.
- **config folder** – This contains all the configuration files of the App. To get a certain configuration, Laravel provides a helper method to do it.
- **database folder** – This folder contains database migrations, factories and seeds. Migrations are database tables definitions such as columns and their datatypes, necessary keys definitions etc.

- **public folder** – This folder contains the index file that is the entry point of the app, once the request is made, it hits that file and then is directed to the necessary route. We will learn about routes later. You can also store public assets here like public images, css, and js.
- **resources folder** – This folder contains our app’s compliable source files, these include views, sass, and raw JavaScript (mostly Node.js, or from JS Frameworks) files. Views are made using HTML with a mixture of a Laravel templating engine called blade, we will learn more about this later.
- **routes folder** – This folder contains all the route files to our app, these route files include web.php, api.php, channels.php, console.php. Each files contains multiple routes as defined by the user. A route is simply a web address that points to a certain function either in the routes file or in the controller.
- **storage folder** – This contains all the private files, such as customer profile pictures. A dynamic link can be created from here to public. All the app logs are stored here also.
- **tests folder** – This is where your app tests are stored.
- **vendor folder** – This is where all third party packages brought by composer are stored.
- .env file – This file contains the environment variables these variables are brought in through config files using the env helper method.

Creating a Controller in Laravel

- Open the command prompt or terminal based on the operating system you are using and type the following command to create controller using the Artisan CLI (Command Line Interface).

```
php artisan make:controller <controller-name> --plain
```

- Replace the <controller-name> with the name of your controller.
- This will create a plain constructor as we are passing the argument — **plain**.
- If you don’t want to create a plain constructor, you can simply ignore the argument.
- The created constructor can be seen at **app/Http/Controllers**.
- You will see that some basic coding has already been done for you and you can add your custom coding. The created controller can be called from routes.php by the following syntax.

Controller Middleware

- We have seen middleware before and it can be used with controller also.
- Middleware can also be assigned to controller's route or within your controller's constructor.
- You can use the middleware method to assign middleware to the controller.
- The registered middleware can also be restricted to certain method of the controller.

Assigning Middleware to Route

```
Route::get('profile', [ 'middleware' => 'auth', 'uses' => 'UserController@showProfile'
]);
```

Restful Resource Controllers

- Often while making an application we need to perform CRUD (Create, Read, Update, Delete) operations.
- Laravel makes this job easy for us.
- Just create a controller and Laravel will automatically provide all the methods for the CRUD operations.
- You can also register a single route for all the methods in routes,php file.

Implicit Controllers

- Implicit Controllers allow you to define a single route to handle every action in the controller.
- You can define it in route.
- php file with **Route:controller** method as shown below.

```
Route::controller('base
URI','<class-name-of-the-controller>');
```

- Replace the <class-name-of-the-controller> with the class name that you have given to your controller.
- The method name of the controller should start with HTTP verb like get or post. If you start it with get, it will handle only get request and if it starts with post then it will handle the post request. After the HTTP verb you can, you can give any name to the method but it should follow the title case version of the URI.

Laravel Basic routing

- Routing is one of the essential concepts in Laravel.
- The main functionality of the routes is to route all your application requests to the appropriate controller.

Default Route files

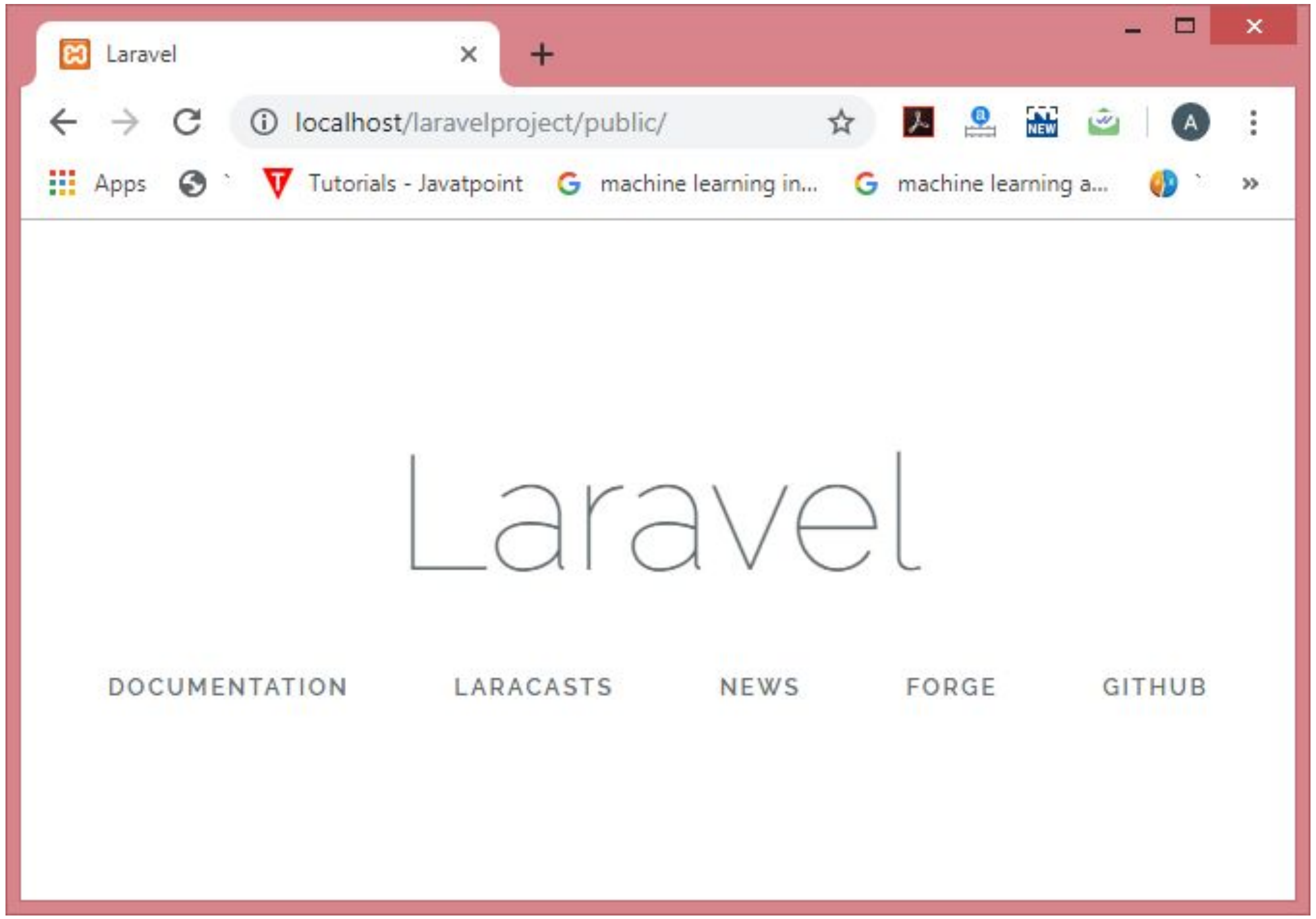
- All Laravel routes are defined inside the route files located in the **routes** directory.
- When we create a project, then a route directory is created inside the project.
- The **route/web.php** directory contains the definition of route files for your web interface.
- The routes in web.php are assigned with the web middleware group that provides the features like session state and CSRF protection.
- The routes defined in **routes/api.php** are assigned with the **API** middleware group, and they are stateless.
- We will start by defining the routes in **routes/web.api** file.
- The routes defined in the routes/web.php can be accessed by entering the defined URL to the browser. Let's understand this through an example.

The definition of default route files.

```
<?php
Route::get('/', function ()
 {
return view ('welcome');
});
```

- In the above case, Route is the class which defines the static method get(). The get() method contains the parameters '/' and function() closure.
- The '/' defines the root directory and function() defines the functionality of the get() method.
- In the above route, the url is '/'; therefore, we entered the **localhost/laravelproject/public URL** in the web browser.

Output



As the method returns the **view('welcome')**, so the above output shows the welcome view of the Laravel.

CSRF Protection

- The HTML forms that are pointing to Post, Put or Delete routes defined in the web route files should include CSRF token field.
- If the CSRF token field is not included, then the request will be rejected.

```
 <form method="POST" action="/profile">
    @csrf
    ...
</form>
```

The router defines the routes that can respond to the following http verbs:

- Route::get($uri, $callback);
- Route::post($uri, $callback);
- Route::put($uri, $callback);
- Route::patch($uri, $callback);
- Route::delete($uri, $callback);
- Route::options($uri, $callback);

Laravel Controller Basics

- Laravel is an MVC based PHP framework.
- In MVC architecture, ‘C‘ stands for ‘Controller‘.
- A Controller is that which controls the behavior of a request.
- It handles the requests coming from the Routes.
- In Laravel, a controller is in the ‘app/Http/Controllers’ directory. All the controllers, that are to be created, should be in this directory.
- We can create a controller using ‘make:controller’ Artisan command.

Syntax:

```
php artisan make:controller
UserController
```

ways for passing data to view in Laravel

- Laravel provides different ways to pass data to a view. We can pass data directly from routes or through the controller.
- Here are some of the ways we can pass data to the view:
  - **Using view()**
  - **Using with()**
  - **Using compact()**
  - **Using Controller Class**

Using with():

- We can directly pass the data in the 'view()' helper function by using the second parameter in the function which takes an array as key and value pair.
- In view(), the first parameter is the name of the view and the second parameter is where we have specified one key and multiple values of the data.
- We can also store the array elements in a variable and then pass it in the view function.

Using with():

- The 'with()' is a method to pass individual form of data and is used with the 'view' helper function.

Using compact():

- The 'compact()' is a PHP function that can be used for creating an array with variable and there value.

Using Controller Class:

- Passing data using controller class is easy and is the right way. Note: Comment or delete any previous route in the '*web.php*' file in '*routes*' directory.