Installation

Meet Laravel

Laravel is a web application framework with expressive, elegant syntax. A web framework provides a structure and starting point for creating your application, allowing you to focus on creating something amazing while we sweat the details.

Laravel strives to provide an amazing developer experience while providing powerful features such as thorough dependency injection, an expressive database abstraction layer, queues and scheduled jobs, unit and integration testing, and more.

Whether you are new to PHP or web frameworks or have years of experience, Laravel is a framework that can grow with you. We'll help you take your first steps as a web developer or give you a boost as you take your expertise to the next level. We can't wait to see what you build.

Why Laravel?

There are a variety of tools and frameworks available to you when building a web application. However, we believe Laravel is the best choice for building modern, full-stack web applications.

A Progressive Framework

We like to call Laravel a "progressive" framework. By that, we mean that

Laravel grows with you. If you're just taking your first steps into web development, Laravel's vast library of documentation, guides, and video tutorials will help you learn the ropes without becoming overwhelmed.

If you're a senior developer, Laravel gives you robust tools for dependency injection, unit testing, queues, real-time events, and more. Laravel is fine-tuned for building professional web applications and ready to handle enterprise work loads.

A Scalable Framework

Laravel is incredibly scalable. Thanks to the scaling-friendly nature of PHP and Laravel's built-in support for fast, distributed cache systems like Redis, horizontal scaling with Laravel is a breeze. In fact, Laravel applications have been easily scaled to handle hundreds of millions of requests per month.

Need extreme scaling? Platforms like Laravel Vapor allow you to run your Laravel application at nearly limitless scale on AWS's latest serverless technology.

A Community Framework

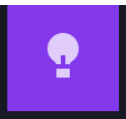
Laravel combines the best packages in the PHP ecosystem to offer the most robust and developer friendly framework available. In addition, thousands of talented developers from around the world have contributed to the framework. Who knows, maybe you'll even become a Laravel contributor.

Your First Laravel Project

We want it to be as easy as possible to get started with Laravel. There are a variety of options for developing and running a Laravel project on your own computer. While you may wish to explore these options at a later time, Laravel provides Sail, a built-in solution for running your Laravel project using Docker.

Docker is a tool for running applications and services in small, light-weight "containers" which do not interfere with your local computer's installed software or configuration. This means you don't have to worry about configuring or setting up complicated development tools such as web servers and databases on your personal computer. To get started, you only need to install Docker Desktop.

Laravel Sail is a light-weight command-line interface for interacting with Laravel's default Docker configuration. Sail provides a great starting point for building a Laravel application using PHP, MySQL, and Redis without requiring prior Docker experience.



Already a Docker expert? Don't worry! Everything about Sail can be customized using the `docker-compose.yml` file included with Laravel.

Getting Started On macOS

If you're developing on a Mac and Docker Desktop is already installed, you can use a simple terminal command to create a new Laravel project. For example, to create a new Laravel application in a directory named "example-app", you may run the following command in your terminal:

```
curl -s "https://laravel.build/example-app" | bash
```

Of course, you can change "example-app" in this URL to anything you like. The Laravel application's directory will be created within the directory you execute the command from.

After the project has been created, you can navigate to the application directory and start Laravel Sail. Laravel Sail provides a simple command-line interface for interacting with Laravel's default Docker configuration:

```
cd example-app

./vendor/bin/sail up
```

The first time you run the Sail `up` command, Sail's application containers will be built on your machine. This could take several minutes. **Don't worry, subsequent attempts to start Sail will be much faster.**

Once the application's Docker containers have been started, you can access the application in your web browser at: http://localhost.

To continue learning more about Laravel Sail, review its complete documentation.

Getting Started On Windows

Before we create a new Laravel application on your Windows machine, make sure to install Docker Desktop. Next, you should ensure that Windows Subsystem for Linux 2 (WSL2) is installed and enabled. WSL allows you to run Linux binary executables natively on Windows 10. Information on how to install and enable WSL2 can be found within Microsoft's developer environment documentation.

After installing and enabling WSL2, you should ensure that Docker Desktop is configured to use the WSL2 backend.

Next, you are ready to create your first Laravel project. Launch Windows Terminal and begin a new terminal session for your WSL2 Linux operating system. Next, you can use a simple terminal command to create a new Laravel project. For example, to create a new Laravel application in a directory named "example-app", you may run the following command in your terminal:

```
curl -s https://laravel.build/example-app | bash
```

Of course, you can change "example-app" in this URL to anything you like. The Laravel application's directory will be created within the directory you execute the command from.

After the project has been created, you can navigate to the application directory and start Laravel Sail. Laravel Sail provides a simple command-line interface for interacting with Laravel's default Docker configuration:

```
cd example-app

./vendor/bin/sail up
```

The first time you run the Sail `up` command, Sail's application containers will be built on your machine. This could take several minutes. **Don't worry, subsequent attempts to start Sail will be much faster.**

Once the application's Docker containers have been started, you can access the application in your web browser at: http://localhost.

To continue learning more about Laravel Sail, review its complete documentation.

Developing Within WSL2

Of course, you will need to be able to modify the Laravel application files that were created within your WSL2 installation. To accomplish this, we recommend using Microsoft's Visual Studio Code editor and their first-party extension for Remote Development.

Once these tools are installed, you may open any Laravel project by executing the `code .` command from your application's root directory using Windows Terminal.

Getting Started On Linux

If you're developing on Linux and Docker is already installed, you can use a simple terminal command to create a new Laravel project. For example, to create a new Laravel application in a directory named "example-app", you may run the following command in your terminal:

```
curl -s https://laravel.build/example-app | bash
```

Of course, you can change "example-app" in this URL to anything you like. The Laravel application's directory will be created within the directory you execute the command from.

After the project has been created, you can navigate to the application directory and start Laravel Sail. Laravel Sail provides a simple command-line interface for interacting with Laravel's default Docker configuration:

```
cd example-app

./vendor/bin/sail up
```

The first time you run the Sail `up` command, Sail's application containers will be built on your machine. This could take several minutes. **Don't worry, subsequent attempts to start Sail will be much faster.**

Once the application's Docker containers have been started, you can access the application in your web browser at: http://localhost.

To continue learning more about Laravel Sail, review its complete documentation.

Choosing Your Sail Services

When creating a new Laravel application via Sail, you may use the `with` query string variable to choose which services should be configured in your new application's `docker-compose.yml` file. Available services include `mysql`, `pgsql`, `mariadb`, `redis`, `memcached`, `meilisearch`, `minio`, `selenium`, and `mailhog`:

```
curl -s "https://laravel.build/example-app?with=mysql,redis" | bash
```

```
curl -s "https://laravel.build/example-app?with=mysql,redis" | bash
```

If you do not specify which services you would like configured, a default stack of `mysql`, `redis`, `meilisearch`, `mailhog`, and `selenium` will be configured.

If your computer already has PHP and Composer installed, you may create a new Laravel project by using Composer directly. After the application has been created, you may start Laravel's local development server using the Artisan CLI's `serve` command:

```
composer create-project laravel/laravel example-app

cd example-app

php artisan serve
```

The Laravel Installer

Or, you may install the Laravel Installer as a global Composer dependency:

```
composer global require laravel/installer

laravel new example-app

cd example-app

php artisan serve
```

Make sure to place Composer's system-wide vendor bin directory in your `$PATH` so the `laravel` executable can be located by your system. This directory exists in different locations based on your operating system; however, some common locations include:

- macOS: `$HOME/.composer/vendor/bin`
- Windows: `%USERPROFILE%\AppData\Roaming\Composer\vendor\bin`
- GNU / Linux Distributions: `$HOME/.config/composer/vendor/bin` or `$HOME/.composer/vendor/bin`

For convenience, the Laravel installer can also create a Git repository for your new project. To indicate that you want a Git repository to be created, pass the `--git` flag when creating a new project:

```
laravel new example-app --git
```

This command will initialize a new Git repository for your project and automatically commit the base Laravel skeleton. The `git` flag assumes

you have properly installed and configured Git. You can also use the `--branch` flag to set the initial branch name:

```
laravel new example-app --git --branch="main"
```

Instead of using the `--git` flag, you may also use the `--github` flag to create a Git repository and also create a corresponding private repository on GitHub:

```
laravel new example-app --github
```

The created repository will then be available at `https://github.com/<your-account>/example-app`. The `github` flag assumes you have properly installed the GitHub CLI and are authenticated with GitHub. Additionally, you should have `git` installed and properly configured. If needed, you can pass additional flags that are supported by the GitHub CLI:

```
laravel new example-app --github="--public"
```

You may use the `--organization` flag to create the repository under a specific GitHub organization:

```
laravel new example-app --github="--public" --organization="laravel"
```

Initial Configuration

All of the configuration files for the Laravel framework are stored in the `config` directory. Each option is documented, so feel free to look through the files and get familiar with the options available to you.

Laravel needs almost no additional configuration out of the box. You are free to get started developing! However, you may wish to review the `config/app.php` file and its documentation. It contains several options such as `timezone` and `locale` that you may wish to change according to your application.

Environment Based Configuration

Since many of Laravel's configuration option values may vary depending on whether your application is running on your local computer or on a production web server, many important configuration values are defined using the `.env` file that exists at the root of your application.

Your `.env` file should not be committed to your application's source control, since each developer / server using your application could

control, since each developer / server using your application could require a different environment configuration. Furthermore, this would be a security risk in the event an intruder gains access to your source control repository, since any sensitive credentials would get exposed.

For more information about the `.env` file and environment based configuration, check out the full configuration documentation.

Directory Configuration

Laravel should always be served out of the root of the "web directory" configured for your web server. You should not attempt to serve a Laravel application out of a subdirectory of the "web directory". Attempting to do so could expose sensitive files that exist within your application.

Next Steps

Now that you have created your Laravel project, you may be wondering what to learn next. First, we strongly recommend becoming familiar with how Laravel works by reading the following documentation:

- Request Lifecycle
- Configuration
- Directory Structure
- Service Container
- Facades

How you want to use Laravel will also dictate the next steps on your journey. There are a variety of ways to use Laravel, and we'll explore two primary use cases for the framework below.

Laravel The Full Stack Framework

Laravel may serve as a full stack framework. By "full stack" framework we mean that you are going to use Laravel to route requests to your application and render your frontend via Blade templates or using a single-page application hybrid technology like Inertia.js. This is the most common way to use the Laravel framework.

If this is how you plan to use Laravel, you may want to check out our documentation on routing, views, or the Eloquent ORM. In addition, you might be interested in learning about community packages like Livewire and Inertia.js. These packages allow you to use Laravel as a full-stack framework while enjoying many of the UI benefits provided by single-page JavaScript applications.

If you are using Laravel as a full stack framework, we also strongly encourage you to learn how to compile your application's CSS and JavaScript using Laravel Mix.

If you want to get a head start building your application, check out one of our official application starter kits.

Laravel The API Backend

Laravel may also serve as an API backend to a JavaScript single-page application or mobile application. For example, you might use Laravel as an API backend for your Next.js application. In this context, you may use Laravel to provide authentication and data storage / retrieval for your application, while also taking advantage of Laravel's powerful services such as queues, emails, notifications, and more.

If this is how you plan to use Laravel, you may want to check out our documentation on routing, Laravel Sanctum, and the Eloquent ORM.

Need a head start scaffolding your Laravel backend and Next.js frontend? Laravel Breeze offers an API stack as well as a Next.js frontend implementation so you can get started in minutes.

Highlights

- Our Team
- Release Notes
- Getting Started
- Routing
- Blade Templates
- Authentication
- Authorization
- Artisan Console
- Database
- Eloquent ORM
- Testing

Resources

- Laracasts
- Laravel News
- Laracon
- Laracon EU
- Jobs
- Certification
- Forums

Partners

- Vehikl
- Tighten
- 64 Robots
- Kirschbaum
- Curotec
- Jump24
- A2 Design
- ABOUT YOU
- Byte 5
- Cubet
- Cyber-Duck
- DevSquad
- Ideil
- Romega Software
- Worksome
- WebReinvent

Ecosystem

- Cashier
- Dusk
- Echo
- Envoyer
- Forge
- Homestead
- Horizon
- Lumen
- Mix
- Nova
- Passport
- Scout
- Socialite
- Spark
- Telescope
- Valet
- Vapor

Laravel is a web application framework with expressive, elegant syntax. We believe development must be an enjoyable and creative experience to be truly fulfilling. Laravel attempts to take the pain out of development by easing common tasks used in most web projects.