



VERSION

9.x



Search

Installation

Meet Laravel

Why Laravel?

Your First Laravel Project

Laravel & Docker

Getting Started On macOS

Getting Started On Windows

Getting Started On Linux

Choosing Your Sail Services

Initial Configuration

Environment Based Configuration

Databases & Migrations

Next Steps

Laravel The Full Stack Framework

Laravel The API Backend

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 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.

New to Laravel? Check out the [Laravel Bootcamp](#) for a hands-on tour of the framework while we walk you through building your first Laravel application.

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

Before creating your first Laravel project, you should ensure that your local machine has PHP and [Composer](#) installed. If you are developing on macOS, PHP and Composer can be installed via [Homebrew](#). In addition, we recommend [installing Node and NPM](#).

After you have installed PHP and Composer, you may create a new Laravel project via the Composer `create-project` command:

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

Or, you may create new Laravel projects by globally installing the Laravel installer via Composer:

```
composer global require laravel/installer
```

```
laravel new example-app
```

After the project has been created, start Laravel's local development server using the Laravel's Artisan CLI `serve` command:

```
cd example-app
```

```
php artisan serve
```

Once you have started the Artisan development server, your application will be accessible in your web browser at `http://localhost:8000`. Next, you're ready to [start taking your next steps into the Laravel ecosystem](#). Of course, you may also want to [configure a database](#).

If you would like a head start when developing your Laravel application, consider using one of our [starter kits](#). Laravel's starter kits provide backend and frontend authentication scaffolding for your new Laravel application.

Laravel & Docker

We want it to be as easy as possible to get started with Laravel regardless of your preferred operating system. So, there are a variety of options for developing and running a Laravel project on your local machine. 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 machine'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 local machine. 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 - just make sure the application name only contains alpha-numeric characters, dashes, and underscores. The Laravel application's directory will be created within the directory you execute the command from.

Sail installation may take several minutes while Sail's application containers are built on your local machine.

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
```

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 - just make sure the application name only contains alpha-numeric characters, dashes, and underscores. The Laravel application's directory will be created within the directory you execute the command from.

Sail installation may take several minutes while Sail's application containers

are built on your local machine.

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
```

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 Compose](#) 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 - just make sure the application name only contains alpha-numeric characters, dashes, and underscores. The Laravel application's directory will be created within the directory you execute the command from.

Sail installation may take several minutes while Sail's application containers are built on your local machine.

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
```

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
```

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

You may instruct Sail to install a default [Devcontainer](#) by adding the `devcontainer` parameter to the URL:

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

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 machine 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 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](#).

Databases & Migrations

Now that you have created your Laravel application, you probably want to store some data in a database. By default, your application's `.env` configuration file specifies that Laravel will be interacting with a MySQL database and will access the database at `127.0.0.1`. If you are developing on macOS and need to install MySQL, Postgres, or Redis locally, you may find it convenient to utilize [DBngin](#).

If you do not want to install MySQL or Postgres on your local machine, you

can always use a [SQLite](#) database. SQLite is a small, fast, self-contained database engine. To get started, create a SQLite database by creating an empty SQLite file. Typically, this file will exist within the `database` directory of your Laravel application:

```
touch database/database.sqlite
```

Next, update your `.env` configuration file to use Laravel's `sqlite` database driver. You may remove the other database configuration options:

```
DB_CONNECTION=sqlite
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=laravel
DB_USERNAME=root
DB_PASSWORD=
```

Once you have configured your SQLite database, you may run your application's [database migrations](#), which will create your application's database tables:

```
php artisan migrate
```

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](#)[Frontend](#)[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.

New to Laravel? Check out the [Laravel Bootcamp](#) for a hands-on tour of the framework while we walk you through building your first Laravel application.

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 a single-page application hybrid technology like [Inertia](#). This is the most common way to use the Laravel framework, and, in our opinion, the most productive way to use Laravel.

If this is how you plan to use Laravel, you may want to check out our documentation on [frontend development](#), [routing](#), [views](#), or the [Eloquent ORM](#). In addition, you might be interested in learning about community

packages like [Livewire](#) and [Inertia](#). 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 [Vite](#).

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.



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.



HIGHLIGHTS

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

RESOURCES

- Laravel Bootcamp
- Laracasts
- Laravel News
- Laracon
- Laracon EU
- Jobs
- 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

Nova

Passport

Scout

Socialite

Spark

Telescope

Valet

Vapor

Laravel is a Trademark of Taylor Otwell. Copyright © 2011-2022 Laravel LLC.

Code highlighting provided by Torchlight