Setting up Laravel Projects

The essentials

In this chapter you will find the things you must know about setting up Laravel projects in this course. The course mainly works with existing projects that are provided to you for further development. These projects are either provided to you as a ZIP file, a GitHub repository or via a GitHub Classroom.

Anatomy of a Laravel Application

A typical Laravel project consists of a fair amount of files and directories. These can be categorized into:

- Code files and folders: folders containing mostly PHP code, assets and application configuration that you are developing

- Packages: libraries and other dependencies that the framework and application depend on. In a Laravel app, they are all stored in the `vendor` directory. Optionally, a `node_modules` folder might exist as well. This depends on whether or not you need to use Node.js.
- Environment configuration: basically a file called `.env` which holds configuration settings for your environment only, like passwords required for the database connection.

When projects are shared, published or deployed, these different categories are treated differently. Basically, only the code files are shared. The other two categories are either created and configured manually or downloaded. This has some advantages like:

- Package size: both `vendor` and `node_modules` are quite large in size (typically +200MB)
- Safety: while `.env` typically holds user credentials, you don't want to share these

> 💡 It is considered good practice **not** to add your dependencies to a git repository directly. Imagine what would happen when you update some dependencies, that updates some 10,000 files in the `vendor` folder, and then a teammate tries to pull your changes after he added a new dependency as well... Good luck solving that merge conflict in less than 2 weeks! Always use the proper dependency tooling for your language: `npm` (for Javascript/Typescript) and `composer` (for PHP).

When looking at this, installing a project for development basically requires:

1. Download the project code into a project directory
2. Installing all the required packages

Installing existing projects

> 💡 We know that every working environment will differ in certain ways (no two computers are installed in exactly the same way). Therefore, this guide may lack details on certain steps that might be valid for your specific environment or may even be wrong for your environment. That is why we advise you to create and maintain your own installation procedure that you can use. It is even allowed to use it during the exam.

As mentioned before, only the project's code files and directories are shared (if it's done properly). To turn this into a working Laravel app, ready for development, you must:

1. Obtain the files, and place it somewhere on your working environment
2. Install the required dependencies
3. Configure your local development so you can test your application locally

Before you start, you need to select/create a proper directory that will be the *parent directory* of your project. It is common practice to select a directory that will be the parent of all your projects, eg. `/my-code` (alternatively, you might want to group per language)

```
\ my-code + projectA + projectB + ...
```

⚠️ We strongly advise you to select a directory that is **NOT** part of your OneDrive or Dropbox environment. E.g. the My Documents folder on a Windows machine is almost by default synchronized to OneDrive. This may cause performance issues when it tries to sync 200+MB of library files

There are basically two ways you can obtain existing code: either via a .ZIP (or other archive) or via Git.

Obtaining files via ZIP

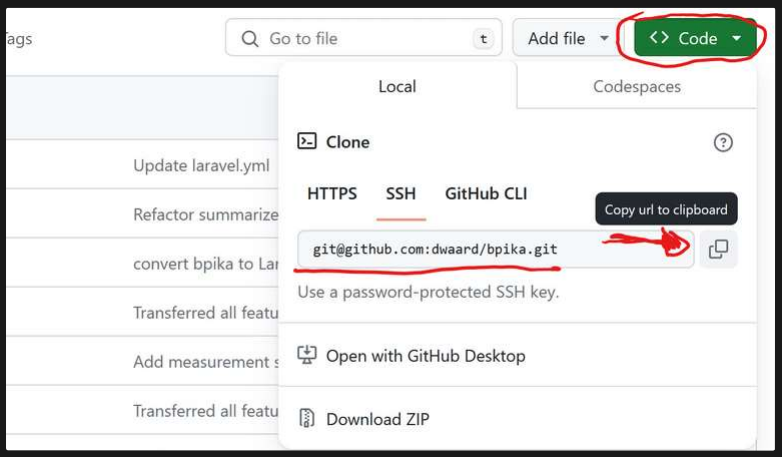
1. Download the provided ZIP file
2. Unzip the content in the selected folder. Keep in mind that sometimes the zip file already has all the code inside a folder and you might want to avoid directories that only contain 1 other directory. For example, when your project is `foobar`, you should have a directory structure like:

```
\ my-code + projectA + projectB + foobar + app + bootstrap + ... +
.editorconfig + ...
```

Obtaining files via Git

Nowadays, most code is shared via Git and some repository platform like GitHub. Using Git means that we do not need zip files to publish, download and unzip. Instead, you will create a working copy of the repository in your code directory. In order to do this, you need a URL of that repository.

In this course we will use GitHub as the main repository location. The URL you need can be found here:



Then, in the previous procedure replace step 1-3 with:

1. Select a directory which will be the parent directory of your project (again, keep away from OneDrive, Dropbox, etc.). Open a command terminal and cd to that folder
2. Clone the repository using the provided URL. Given the repository in the image above, the terminal command will be:

```
git clone git@github.com:dwaard/bpika.git
```

3. Typically, new folder is created with the repository name which holds the project. for the example, it will be `bpika`.

```
\ my-code + projectA + projectB + bpika + app + bootstrap + ... +
.editorconfig + ...
```

Installing dependencies

1. Cd into the project folder

2. Use Composer to install the projects dependencies and create the `vendor` directory. Typically, using a command like:

```
composer install
```

3. If you need to use Node and/or NPM, you should create the `node_modules` directory with the command

```
npm install
```

Setting up the local development environment

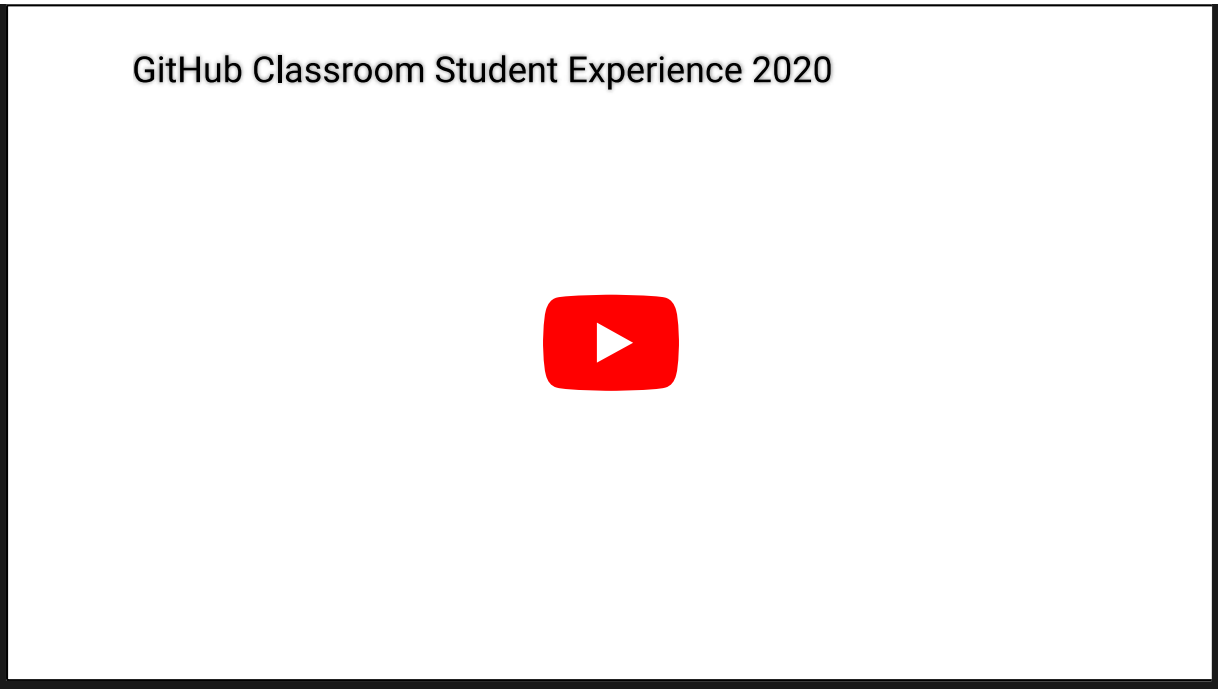
1. Create the local environment configuration file `.env`. In most cases you can do this by copying the provided `env.example` file
2. Then follow the steps you need to make the project testable on you localhost webserver, like XAMPP.

Setting up a GitHub Classroom Project

> 💡 It is good to know that we are using Github Classrooms for the weekly project assignments and the final exam. The steps are a little bit different as how real developers share their code, but they will also use Git and platforms like Github.

Github Classrooms is a platform that leverages *Github* as a tool that helps teachers and students collaborate. On this platform, teachers can create assignments based on existing code where students can work on individually or in groups.

The following video shows how students can use the platform:



a Basic workflow when using a Classroom is:

1. The teacher creates a classroom and publishes an invitation link
2. You accept the assignment by visiting this link
3. On acceptance, a GitHub repository is created for you that contains the starter code of the assignment, which is in our case a Laravel project
4. From here on, you can treat it like a normal GitHub repository and proceed as is described in the previous section

⚠️ Once the assignment is accepted, you do not need to do this step again.

Digging deeper

The SHOULDs about this topic, described as links with some extra information to help understanding the content of the link

Setting up using Docker and Sail

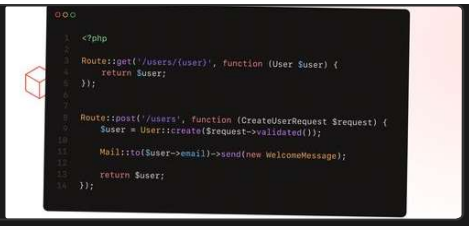
When using Docker and Sail, this are a bit different. The main reason is that some tools, like Composer and npm are (or should not be) installed. For this, Docker comes to the rescue.

Install all the dependencies using a small Docker container:

Laravel - The PHP Framework For Web Artisans

Laravel Sail is a light-weight command-line interface for interacting with Laravel's default Docker development

https://laravel.com/docs/sail#installing-composer-depend...



```
docker run --rm \ -u "$(id -u):$(id -g)" \ -v "$(pwd):/var/www/html" \ -w /var/www/html \ laravelsail/php82-composer:latest \ composer install --ignore-platform-reqs
```

And start the Docker container with **Sail**:

```
vendor/bin/sail up -d
```

💡 To avoid retyping `vendor/bin/sail` for each command you can add an alias, like:

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

Fortunately, you only need to do this once in your Linux distro.

Creating a new project

This page describes the steps and best practices to setup a new Laravel project (together with a GitHub repo)

🧙‍♂️ Initial Laravel Project Setup

Further reading

✨ Convenient Laravel Packages