

UNIT 5: INSTALLATION

1. PHP FRAMEWORKS

A **framework** is a tool that provides a number of modules that help organize and develop a software product.

In the specific case of PHP frameworks, most of them provide a series of commands or tools to create projects with a certain structure (usually, following the MVC pattern that we will see later), so that they already give a base of work done, and facilities to be able to create the data model, the connection to the database, the paths of the different sections of the application, etc.

1.1. Previous resources

When working with Laravel, we need to have previously installed on our system a series of software resources, such as:

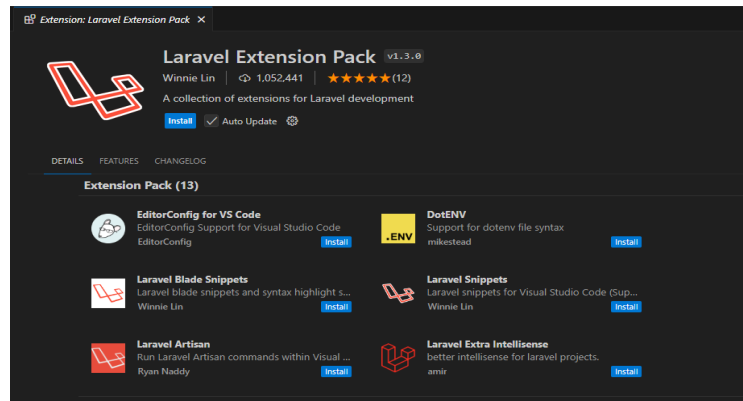
1. An IDE (development environment) with which to edit the code of our projects. We will use Visual Studio Code in these notes, although there are other similar alternatives, such as PHPStorm, Sublime Text, etc.
2. A web server that supports PHP.
3. A database server on which to store the information of our applications. We will use a MariaDB/MySQL server.
4. PHP
5. The Laravel framework itself.
6. In addition, we will need the 'npm' package manager to install client-side dependencies on Laravel projects.

In this document you have the steps to install each of these elements using a docker.

1.2. Visual Studio Code: Useful extensions

When we use Visual Studio Code to develop applications in a certain technology (Laravel, Node, etc.), it may be convenient to install a *plugin* or extension that facilitates the development of these applications. Through these extensions we can, for example, highlight the syntax of the files we edit, help us autocomplete code, etc.

In the case of Laravel, we can go to the *Extensions* section of the left menu of Visual Studio Code, and look for the extension *Laravel Extension Pack*. Then, click on the *Install* button to install this extension, which in turn contains a package of extensions useful for developing Laravel applications.



Specifically, we can highlight the syntax of the HTML views that we make with the Blade template engine, autocomplete certain parts of the code, auto-correct certain code errors, etc.

Also install the extensions: Dev containers and Docker if you want to manage docker from Visual Studio Code

2. ENVIRONMENT PREPARATION (optional)

1. Create an Ubuntu 22 virtual machine: ubuntu-22.04.2-desktop-amd64
2. Install Docker Engine

You can download an Ubuntu 22 virtual machine virtual from teams with Docker Engine installed. (user:marta and password: 123). So, you can skip this step and go directly to 3. INSTALLATION LARAVEL DOCKER AND CREATION OF CONTAINERS

2.1. Installation methods

[Install Docker Engine on Ubuntu | Docker Documentation](#)

You can install Docker Engine in different ways, depending on your needs:

1. Docker Engine comes bundled with [Docker Desktop for Linux](#). This is the easiest and quickest way to get started.
2. **Set up and install Docker Engine from [Docker's apt repository](#).**
3. [Install it manually](#) and manage upgrades manually.
4. Use a [convenience script](#). Only recommended for testing and development environments.

We will use the 2n method.

2.1.1. Install using the apt repository

[Install Docker Engine on Ubuntu | Docker Docs](#)

Before you install Docker Engine for the first time on a new host machine, you need to set up the Docker repository. Afterward, you can install and update Docker from the repository.

Set up the repository

- Update the `apt` package index and install packages to allow `apt` to use a repository over HTTPS:

```
$ sudo apt-get update
```

```
$ sudo apt-get install ca-certificates curl gnupg
```

- Add Docker's official GPG key:

```
$ sudo install -m 0755 -d /etc/apt/keyrings
```

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg  
--dearmor -o /etc/apt/keyrings/docker.gpg
```

```
$ sudo chmod a+r /etc/apt/keyrings/docker.gpg
```

- Use the following command to set up the repository:

```
$ echo \  
    "deb [arch="$(dpkg --print-architecture)" signed-  
by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \  
    "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \  
    sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Note

If you use an Ubuntu derivative distro, such as Linux Mint, you may need to use `UBUNTU_CODENAME` instead of `VERSION_CODENAME`.

Install Docker Engine

- Update the `apt` package index:

```
$ sudo apt-get update
```

- Install Docker Engine, containerd, and Docker Compose.

To install the latest version, run:

```
$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-  
plugin docker-compose-plugin
```

- Verify that the Docker Engine installation is successful by running the `hello-world` image.

```
$ sudo docker run hello-world
```

This command downloads a test image and runs it in a container. When the container runs, it prints a confirmation message and exits.

You have now successfully installed and started Docker Engine.

Post-installation

[Linux post-installation steps for Docker Engine | Docker Documentation](#)

Receiving errors when trying to run without root? If you execute "docker ps" ... You do not have permission

The `docker` user group exists but contains no users, which is why you're required to use `sudo` to run Docker commands. Continue to allow non-privileged users to run Docker commands and for other optional configuration steps.

To create the `docker` group and add your user:

- Create the `docker` group.

```
$ sudo groupadd docker (This step is not necessary because the group has already been created)
```

- Add your user to the `docker` group.

```
$ sudo usermod -aG docker $USER
```

- Log out and log back in so that your group membership is re-evaluated.

If you're running Linux in a virtual machine, it may be necessary to restart the virtual machine for changes to take effect.

You can also run the following command to activate the changes to groups:

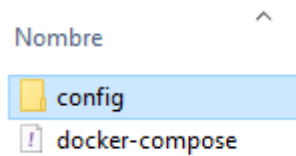
```
$ newgrp docker
```

- Verify that you can run `docker` commands without `sudo`.

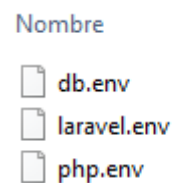
```
$ docker run hello-world
```

3. INSTALLATION LARAVEL DOCKER AND CREATION OF CONTAINERS:

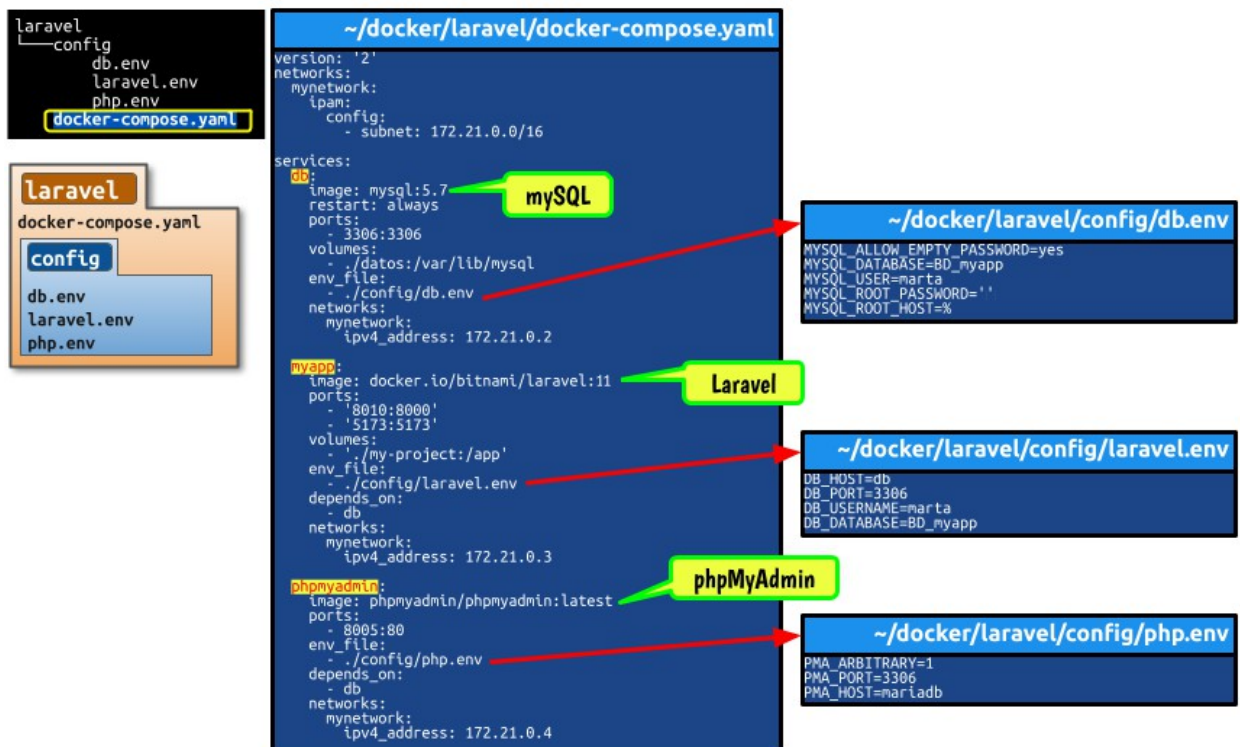
Create the folder laravel in /home/user and copy the docker files:



config folder with



Information docker-compose:



- db: network: 172.21.0.2

This is the container with the data in the DB.

- myapp: network: 172.21.0.3

This is the container with the Laravel project

It will be accessible in port <http://localhost:8010>

- phpmyadmin: network: 172.21.0.4

This is the container with phpMyadmin as database management system

It will be accessible in port <http://localhost:8005>

Open a terminal 1, Go to the laravel folder and execute:

```
$ docker compose up
```

```
marta@marta-VirtualBox: ~/laravel
marta@marta-VirtualBox:~/laravel$ docker compose up
[+] Running 17/21
  :: db 11 layers [ ] 47.81MB/50.5MB Pulling 43.5s
  :: phpmyadmin 18 layers [ ] 103.9MB/104.3MB Pulling 43.5s
  :: myapp 1 layers [ ] 28.37MB/267.3MB Pulling 43.5s
```

It stops "trying to connect the database"

Access denied for user 'marta'@'172.21.0.3' (using password: NO) means that the machine Laravel is going to use the user marta to connect to the machine mysql.

It gives an access error for user marta, so let's fix that

```
laravel-db-1 | 2024-11-18T16:36:07.553802Z 0 [Note] Skipping generation of RSA key pair a
s key files are present in data directory.
laravel-db-1 | 2024-11-18T16:36:07.554649Z 0 [Note] Server hostname (bind-address): '*';
port: 3306
laravel-db-1 | 2024-11-18T16:36:07.554690Z 0 [Note] IPv6 is available.
laravel-db-1 | 2024-11-18T16:36:07.554726Z 0 [Note] - '::' resolves to '::';
laravel-db-1 | 2024-11-18T16:36:07.554759Z 0 [Note] Server socket created on IP: '::'.
laravel-db-1 | 2024-11-18T16:36:07.561804Z 0 [Warning] Insecure configuration for --pid-f
ile: Location '/var/run/mysqld' in the path is accessible to all OS users. Consider choosing a dif
ferent directory.
laravel-db-1 | 2024-11-18T16:36:07.587065Z 0 [Note] Event Scheduler: Loaded 0 events
laravel-db-1 | 2024-11-18T16:36:07.587512Z 0 [Note] mysqld: ready for connections.
laravel-db-1 | Version: '5.7.44' socket: '/var/run/mysqld/mysqld.sock' port: 3306 MySQL
Community Server (GPL)
laravel-myapp-1 | laravel 16:36:33.12 INFO ==> Trying to connect to the database server
laravel-db-1 | 2024-11-18T16:36:33.189347Z 2 [Note] Got an error reading communication pa
ckets
laravel-myapp-1 | laravel 16:36:33.19 INFO ==> Executing database migrations
laravel-db-1 | 2024-11-18T16:36:33.619274Z 3 [Note] Access denied for user 'marta'@'172.2
1.0.3' (using password: NO)
laravel-myapp-1 exited with code 1
```

Don't close terminal 1.

Open a new terminal 2,

Go to the laravel folder and enter the laravel container in interactive mode executing:

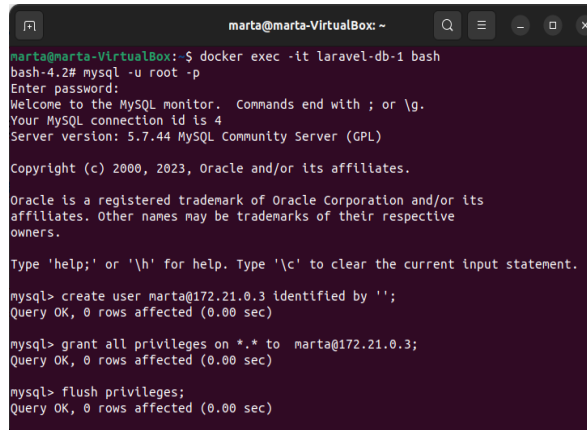
```
$ docker exec -it laravel-db-1 bash
```

and then....

```
bash-4.2#mysql -u root -p
```

password= " (like in db.env)

```
mysql>create user marta@172.21.0.3 identified by ''; //172.21.0.3 docker's ip
mysql>grant all privileges on *.* to marta@172.21.0.3;
mysql>flush privileges;
```



```
marta@marta-VirtualBox: ~
marta@marta-VirtualBox:~$ docker exec -it laravel-db-1 bash
bash-4.2# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.7.44 MySQL Community Server (GPL)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

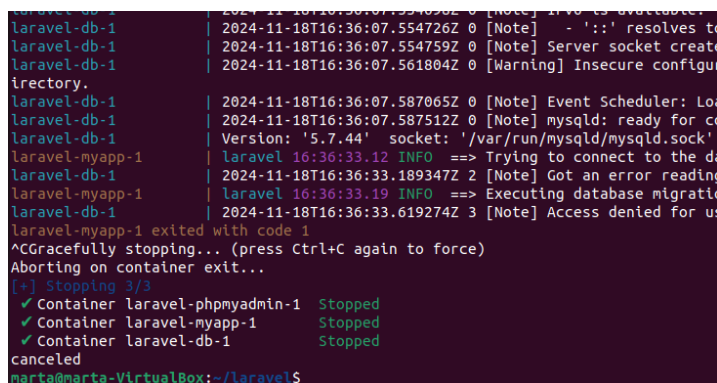
mysql> create user marta@172.21.0.3 identified by '';
Query OK, 0 rows affected (0.00 sec)

mysql> grant all privileges on *.* to marta@172.21.0.3;
Query OK, 0 rows affected (0.00 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)
```

Close terminal 2

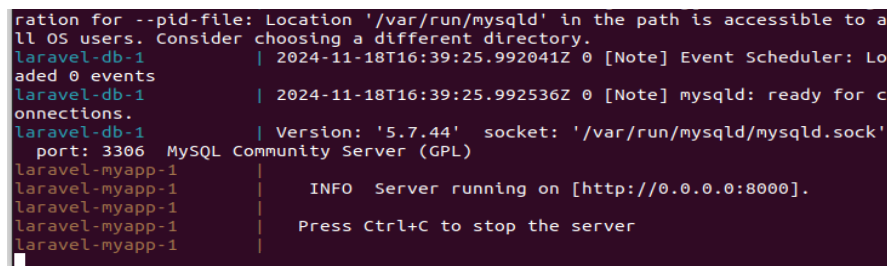
Return to terminal 1 press Ctrl + C in order to stop the containers.



```
laravel-db-1 | 2024-11-18T16:36:07.554726Z 0 [Note] Event Scheduler: Load
laravel-db-1 | 2024-11-18T16:36:07.554759Z 0 [Note] Server socket create
laravel-db-1 | 2024-11-18T16:36:07.561804Z 0 [Warning] Insecure configur
irectory.
laravel-db-1 | 2024-11-18T16:36:07.587065Z 0 [Note] Event Scheduler: Load
laravel-db-1 | 2024-11-18T16:36:07.587512Z 0 [Note] mysqld: ready for co
laravel-db-1 | Version: '5.7.44' socket: '/var/run/mysqld/mysqld.sock'
laravel-myapp-1 | laravel 16:36:33.12 INFO ==> Trying to connect to the da
laravel-db-1 | 2024-11-18T16:36:33.189347Z 2 [Note] Got an error reading
laravel-myapp-1 | laravel 16:36:33.19 INFO ==> Executing database migratio
laravel-db-1 | 2024-11-18T16:36:33.619274Z 3 [Note] Access denied for us
laravel-myapp-1 exited with code 1
^CGracefully stopping... (press Ctrl+C again to force)
Aborting on container exit...
[+] Stopping 3/3
✓ Container laravel-phpmyadmin-1 Stopped
✓ Container laravel-myapp-1 Stopped
✓ Container laravel-db-1 Stopped
canceled
marta@marta-VirtualBox:~/laravel$
```

Type again:

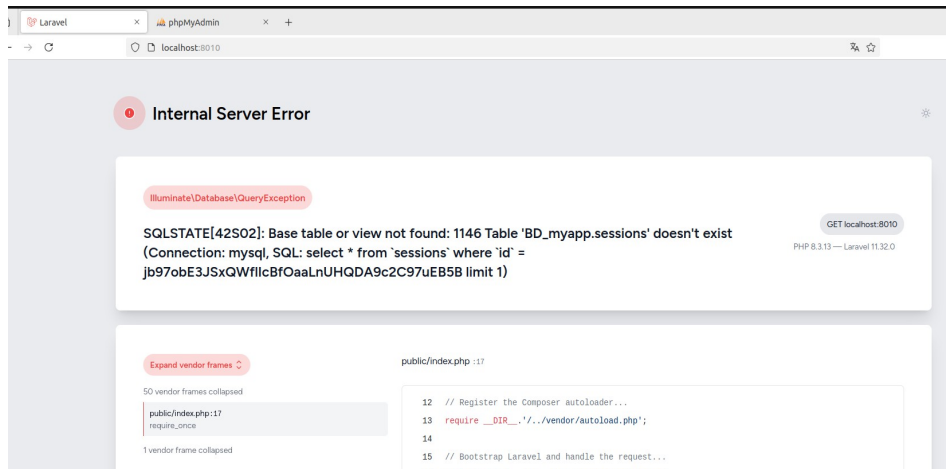
```
$ docker compose up
```



```
ration for --pid-file: Location '/var/run/mysqld' in the path is accessible to a
ll OS users. Consider choosing a different directory.
laravel-db-1 | 2024-11-18T16:39:25.992041Z 0 [Note] Event Scheduler: Lo
aded 0 events
laravel-db-1 | 2024-11-18T16:39:25.992536Z 0 [Note] mysqld: ready for c
onnections.
laravel-db-1 | Version: '5.7.44' socket: '/var/run/mysqld/mysqld.sock'
port: 3306 MySQL Community Server (GPL)
laravel-myapp-1 |
laravel-myapp-1 | INFO Server running on [http://0.0.0.0:8000].
laravel-myapp-1 |
laravel-myapp-1 | Press Ctrl+C to stop the server
laravel-myapp-1 |
```

Don't close this terminal, while you are working with your laravel project

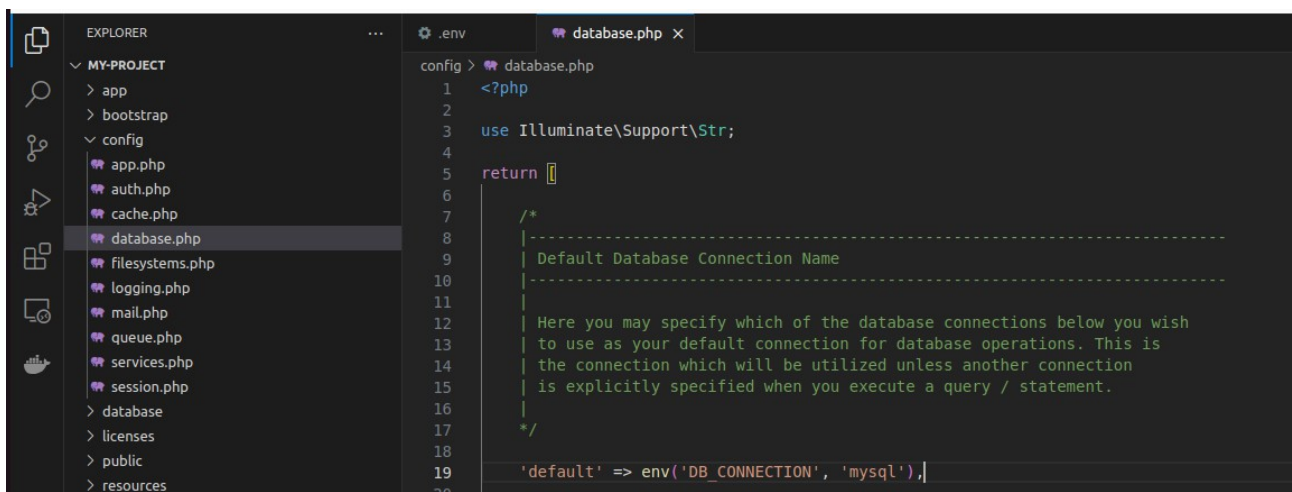
Our Laravel project is working at <http://localhost:8010/>
But with an error:



This is because we have to configure the DB

Open your project folder "my-project" (inside laravel) with the Visual Studio Code.

In config/database.php
'default' => env('DB_CONNECTION', 'mysql'),



In .env
DB_CONNECTION=mysql
DB_HOST=172.21.0.2
DB_PORT=3306
DB_DATABASE=BD_myapp
DB_USERNAME=marta
DB_PASSWORD=


```
session.php
> database
> licenses
> public
> resources
> routes
> storage
> tests
> vendor
≡ .buildcomplete
⚙ .editorconfig
⚙ .env
$ .env.example
🔗 .gitattributes
🔗 .gitignore
≡ .spx-laravel.spx
≡ artisan

11
12 APP_MAINTENANCE_DRIVER=file
13 # APP_MAINTENANCE_STORE=database
14
15 PHP_CLI_SERVER_WORKERS=4
16
17 BCRYPT_ROUNDS=12
18
19 LOG_CHANNEL=stack
20 LOG_STACK=single
21 LOG_DEPRECATIONS_CHANNEL=null
22 LOG_LEVEL=debug
23
24 DB_CONNECTION=mysql
25 DB_HOST=172.21.0.2
26 DB_PORT=3306
27 DB_DATABASE=bd_myapp
28 DB_USERNAME=marta
29 DB_PASSWORD=
30
```

From another terminal:

```
docker exec -it laravel-myapp-1 bash
php artisan migrate
```

And you can close the terminal

```
marta@marta-VirtualBox: ~/laravel
marta@marta-VirtualBox:~/laravel$ docker exec -it laravel-myapp-1 bash
root@aa29365d1ac1:/app# php artisan migrate

[INFO] Preparing database.

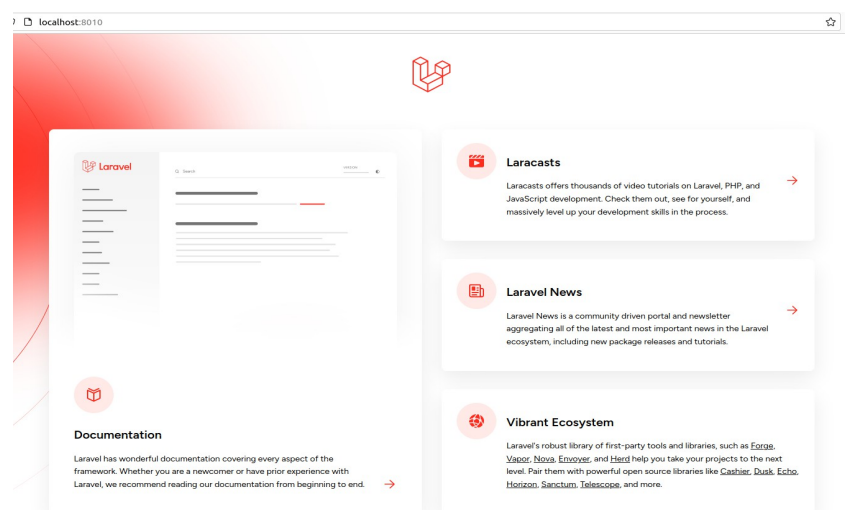
Creating migration table ..... 44.61ms DONE

[INFO] Running migrations.

0001_01_01_000000_create_users_table ..... 145.42ms DONE
0001_01_01_000001_create_cache_table ..... 61.72ms DONE
0001_01_01_000002_create_jobs_table ..... 102.78ms DONE

root@aa29365d1ac1:/app#
```

Then, your project is working:

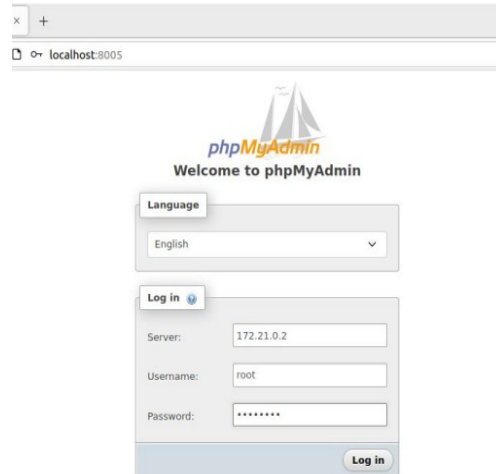


To access phpmyadmin: <http://localhost:8005/>

Server: 172.21.0.2

Username: root

Password: "

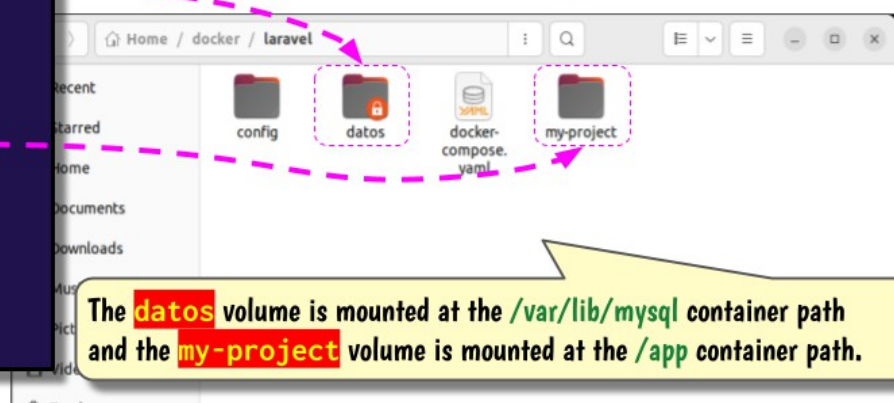


If we open the folder laravel, we can see there are 2 new folders: datos and my-project. They are the volumes for the database and the Laravel project respectively.

```
~/docker/laravel/docker-compose.yml
version: '2'
networks:
  mynetwork:
    ipam:
      config:
        - subnet: 172.21.0.0/16
services:
  db:
    image: mysql:5.7
    restart: always
    ports:
      - 3306:3306
    volumes:
      - ./datos:/var/lib/mysql
    env_file:
      - ./config/db.env
    networks:
      mynetwork:
        ipv4_address: 172.21.0.2
  myapp:
    image: docker.io/bitnami/laravel:11
    ports:
      - 8080:8080
      - 5173:5173
    volumes:
      - ./my-project:/app
    env_file:
      - ./config/laravel.env
    depends_on:
      - db
    networks:
      mynetwork:
        ipv4_address: 172.21.0.3
  phpmyadmin:
    image: phpmyadmin/phpmyadmin:latest
    ports:
      - 8899:80
    env_file:
      - ./config/php.env
    depends_on:
      - db
    networks:
      mynetwork:
        ipv4_address: 172.21.0.4
```

Two new folders have been created according to the specifications that I wrote in the "volumes" section of the docker-compose.yml file:

- **datos** (→ db service)
- **my-project** (→ myapp service)



To stop the project, from the terminal opened:

```
$ docker compose down
```

Rename my-project folder to library

Make a duplicate from folder library and name it blog

We are going to work with two projects during the course:

- library: Project about books and authors. We will do this project together. I will make a video for you with all the practices, and you will repeat it at home
- blog: This will be the project that you will deliver every fortnight.

IMPORTANT: And you will have to bring it for the exam, since it will be a mandatory validation practice in the exam

To work with both projects, you must change the name of the project you want to work with in docker-compose.yml. For example if you want to work with library:

In docker-compose.yml change:

volumes:

- './my-project:/app'

For:

volumes:

- './library:/app'