Deployment

Arash Mehrabi

December 25, 2020

1 Server Requirements

We have used Laravel 7 for developing this project so you need to make sure that your server meets the following requirements:
For Laravel:

- PHP >= 7.2.5
- BCMath PHP Extension
- Ctype PHP Extension
- Fileinfo PHP extension
- JSON PHP Extension
- Mbstring PHP Extension
- OpenSSL PHP Extension
- PDO PHP Extension
- Tokenizer PHP Extension
- XML PHP Extension
- Composer PHP Package Manager

For database you should use MySql:

- MySql >= 5.7.31

Operating System:

- For operating system you can use anything (MAC, Linux, Windows, etc.) but you should be able to setup server on that machine.
- You should probably have superuser privileges.
- We recommend using Linux.

For web server:

- We recommend using Nginx if you are going to deploy it on a real server but you can use apache or any other web server too.

2 Install Prerequistes

General:

```
sudo apt update
sudo apt install -y git curl wget zip unzip
```

Nginx:

```
sudo apt install nginx
sudo ufw allow 'Nginx HTTP'
```

MySql:

```
sudo apt install mysql-server
```

PHP:

```
sudo apt install php7.4 php7.4-fpm php7.4-mysql php-common php7.4-cli php7.4-common
```

Composer:

```
curl -sS https://getcomposer.org/installer | php
sudo mv composer.phar /usr/local/bin/composer
# this make the composer executable ->
sudo chmod +x /usr/local/bin/composer
```

3 How to deploy?

3.1 Clone

After installing the requirements, first clone the code in an arbitrary directory by directing to your directory and entering the command:

```
git clone [address of the repository]
```

Then move the cloned repository to `/var/www/html`. Now we are done with cloning.

3.2 Ownership & Permissions

We need to change ownership & permissions of the files of the project so the server can read & write necessary files. Simultaneously we should avoid giving too much freedom to our web server so hackers could harm our server.

You should own all of the directories and files by your user. Enter the command:

```
sudo chown -R my-user:www-data /path/to/your/laravel/root/directory
```

Then give both your user and web server required permissions:

```
sudo find /path/to/your/laravel/root/directory -type f -exec chmod 664 {} \;
sudo find /path/to/your/laravel/root/directory -type d -exec chmod 775 {} \;
```

Then you need to give read and write permissions to the web server for storage, cache and any other directories the web server needs to upload or write too (depending on your situation), so run the commands:

```
sudo chgrp -R www-data storage bootstrap/cache
sudo chmod -R ug+rwx storage bootstrap/cache
```

Now, you're secure and your website works and you can work with the files fairly easily.[1]

[1] For more information please visit: bgies' answer on stackoverflow

3.3 Setup Nginx

First you need to change the group ownership of the `storage` and `bootstrap/cache` directories to **www-data**.

```
sudo chgrp -R www-data storage bootstrap/cache
```

Then recursively grant all permissions, including write and execute, to the group.

```
sudo chmod -R ug+rwx storage bootstrap/cache
```

Now we have granted all necessary privileges. Next we should create a new server blog config file by copying over the default file.

```
sudo cp /etc/nginx/sites-available/default /etc/nginx/sites-available/example.com
```

Open the newly created configuration file.

```
sudo nano /etc/nginx/sites-available/example.com
```

In this file you need change a few things:

- Removing the `default_server` designation from listen directives,
- Updating the web root by changing the `root` directive,
- Updating the the `server_name` directive to correctly point to a domain name for the server,
- Updating the request URI handling by changing the `try_files` directive.

The modified Nginx configuration file will look like this:

```
    server {
    listen 80;
    listen [::]:80;

    . . .

    root /var/www/html/quickstart/public;
    index index.php index.html index.htm index.nginx-debian.html;

    server_name example.com www.example.com;

    location / {
        try_files $uri $uri/ /index.php?$query_string;
    }

    . . .
}
```

Then save & close the file. Now we should enable this new configuration by creating a symbolic link to the `sites_enabled` directory.

```
sudo ln -s /etc/nginx/sites-available/example.com /etc/nginx/sites-enabled/
```

And finally reload the Nginx to apply the changes.[2]

```
sudo systemctl reload nginx
```

Now website has been deployed and if you access it from your server's IP you will see the first page of the website! Or if you had deployed it in your local machine you can access it from `localhost`.

3.4 Installing Application

Now we need to install the application. Run MySql by typing:

```
mysql -u [your user] -p
```

Then type your MySql's username's password and then create a database for this application by entering:

```
CREATE DATABASE [an arbitrary name here];
```

After this quit mysql and go to the project's directory.

Copy the `.env.example` file by typing:

```
cp .env.example .env
```

then edit the `.env` file and enter database's information there.

It should be like this:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=[your database name]
DB_USERNAME=[your mysql user]
DB_PASSWORD=[your mysql password]
```

Then again route to Laravel's directory and type:

```
composer install
```

to install the dependencies of Laravel application.

Then run:

```
php artisan key:generate
```

to generate a key for your Laravel application.

Now everything is configured. You should let Laravel to create necessary tables in your database by typing:

```
php artisan migrate
```

Congratulations! Now the web application has been installed on your machine.

[2]For more information you can check: Digital Ocean