Configuración de Laravel 6 en Docker

Objetivo: Configurar docker para nginx (última versión), php (última versión), mysql8 y laravel 6 (útima versión)

Autor: Ambrosio Cardoso Jiménez

Fecha última revisión: 26-Feb-2020

Requisitos:

- Tener instalado docker (para esta práctica se uso versión 19.03.5, build 633a0ea838)
- Tener instalado docker-compose
- Tener instalado composer (gestor de paquetes para programar en PHP el cual provee los formatos estándar necesarios para manejar dependencias y librerías)

Paso 1: Crear una carpeta **dockers/ugm**

Paso 2: Dentro de ella crear carpeta **nginx** y dentro de ella un archivo de texto **nginx.conf** cuyo contenido será la siguiente:

```
server {
    listen 80;
    root /var/www/html/proy_ugm/public;
    index index.php index.html;
    server_name localhost;

    error_log  /var/log/nginx/error.log;
    access_log /var/log/nginx/access.log;
```

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

    location ~ \.php$ {
        try_files $uri =404;
        fastcgi_split_path_info ^(.+\.php)(/.+)$;
        fastcgi_pass ugm_php:9000;
        fastcgi_index index.php;
        include fastcgi_params;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        fastcgi_param PATH_INFO $fastcgi_path_info;
    }
}
```

Paso 3: Crear la carpeta vacía **dockers/ugm/www**

Paso 4: Crear el archivo **docker-compose.yml** en el directorio **ugm** con el siguiente contenido:

```
version: '3.7'

networks:
    ugm:
services:
    ugm_nginx:
```

```
        image: nginx:stable-alpine
        container_name: ugm_nginx
        ports:
            - "8000:80"
        volumes:
            - "./www:/var/www/html"
            - "./nginx/nginx.conf:/etc/nginx/conf.d/default.conf"
        depends_on:
            - ugm_php
            - ugm_mysql
        networks:
            - ugm

    ugm_mysql:
        image: mysql:latest
        container_name: ugm_mysql8
        volumes:
            - ugm_mysql8_data:/var/lib/mysql
        command: ['--character-set-server=utf8mb4', '--collation-
server=utf8mb4_unicode_ci','--default-authentication-
plugin=mysql_native_password']
        tty: true
        ports:
            - "3333:3306"
        environment:
            MYSQL_DATABASE: bdugm
            MYSQL_USER: ugmDev
```

```
            MYSQL_PASSWORD: pwdUgmDev
            MYSQL_ROOT_PASSWORD: t0pS3cr3t
            SERVICE_NAME: ugm_mysql
        networks:
            - ugm

    ugm_php:
        build:
            context: .
            dockerfile: Dockerfile
        container_name: ugm_php
        volumes:
            - "./www:/var/www/html"
        ports:
            - "9999:9000"
        networks:
            - ugm
volumes:
    ugm_mysql8_data:
```

Paso 5: Crear el archivo **Dockerfile** en el directorio **ugm** con el siguiente contenido:

```
#--- El repositorio docker de PHP
FROM php:7.2-fpm-alpine
RUN docker-php-ext-install pdo pdo_mysql mysqli
```

```
#--- Usar apk en lugar de apt-get para cuando se usa alpine
#--- nota de Cardoso 26-Feb-2020
RUN apk update  && apk add \
    curl \
    wget \
    zip \
    unzip \
    composer
```

Paso 6: Ejecutar la siguiente instrucción desde la carpeta **ugm**

docker-compose build

NOTA: En ocasiones aparece el siguiente mensaje de error:

ERROR: Couldn't connect to Docker daemon at http+docker://localunixsocket - is it running?

SOLUCION: iniciar docker (systemctl start docker para linux y simplemente ejecutarlo para Windows)

Volver a ejecutar la sentencia **docker-compose build**

Paso 7: Iniciar los contenedores ejecutando

docker-compose up -d (-d=>para ejecutar en segundo plano)

Paso 8: Comprobar que los tres contenedores (nginx, mysql8 y php) estén corriendo

docker ps

```
[cardoso@asuboshy ugm]$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                               NAMES
9f24d371f4dc        nginx:stable-alpine   "nginx -g 'daemon of…"   4 days ago          Up 14 minutes       0.0.0.0:8000->80/tcp                ugm_nginx
112cef5a1fc3        ugm_ugm_php           "docker-php-entrypoi…"   4 days ago          Up 14 minutes       0.0.0.0:9999->9000/tcp              ugm_php
0a9ac1b36dae        mysql:latest          "docker-entrypoint.s…"   4 days ago          Up 14 minutes       33060/tcp, 0.0.0.0:3333->3306/tcp   ugm_mysql8
[cardoso@asuboshy ugm]$
```

Figura 1. Contenedores en ejecución

Paso 9: Entrar al contenedor

docker exec -it php /bin/sh

Paso 10: **Escriba composer.** Debe mostrar la documentación de composer, si dice que no existe o no reconoce la sentencia entonces **ejecutar las siguientes sentencias**

 # php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"

 # HASH=="$(wget -q -O - https://composer.github.io/installer.sig)"

 # php composer-setup.php --install-dir=/usr/local/bin –filename=composer

 # mv /usr/local/bin/composer.phar /usr/bin/composer

Paso 11: Estando en el contenedor cambiar al directorio ***/var/www/html***

cd /*var/www/html*

Paso 12: Crear el proyecto (proy_ugm) laravel estando en la ruta /var/www/html

 # composer create-project --prefer-dist laravel/laravel proy_ugm

esperar a que termine la descarga...

Paso 13: Editar el archivo /var/www/html/proy_ugm/.env cambiando los datos resaltados

APP_NAME=**ugm**

APP_ENV=local

APP_KEY=base64:82Z0EFT/ysqWDaaThBcTajemwOsVT1Pkov4pLkTJePw=

APP_DEBUG=true

APP_URL=http://localhost:8000

LOG_CHANNEL = stack

DB_CONNECTION = mysql

DB_HOST = **192.168.80.2**

DB_PORT = 3306

DB_DATABASE = ugm

DB_USERNAME = root

DB_PASSWORD = t0pS3cr3t

...

La **IP** se obtiene con la siguiente instrucción

docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' **ugm_mysql8**

(En una sola línea, **ugm_mysql8** es el nombre del contenedor como se aprecia en la figura 1)

Paso 14: Desde el host probar desde el navegador http://localhost:8000

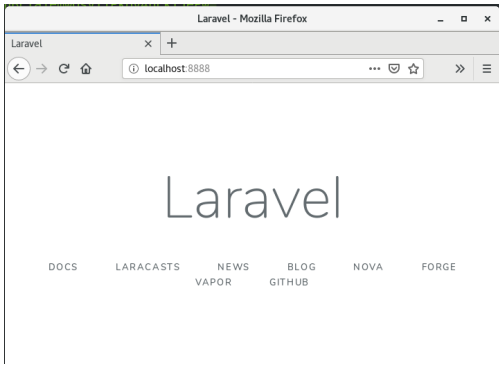

Figura 2. Laravel en ejecución

CRUD

a) Entrar al contenedor de **ugm_mysql8**

docker exec -it ugm_mysql8 /bin/sh

b)

Si se cuenta con una BD existente, es conveniente generar los modelos que representen las tablas y sus relaciones