Despliegue de una aplicación
Jose Alba Arrufat

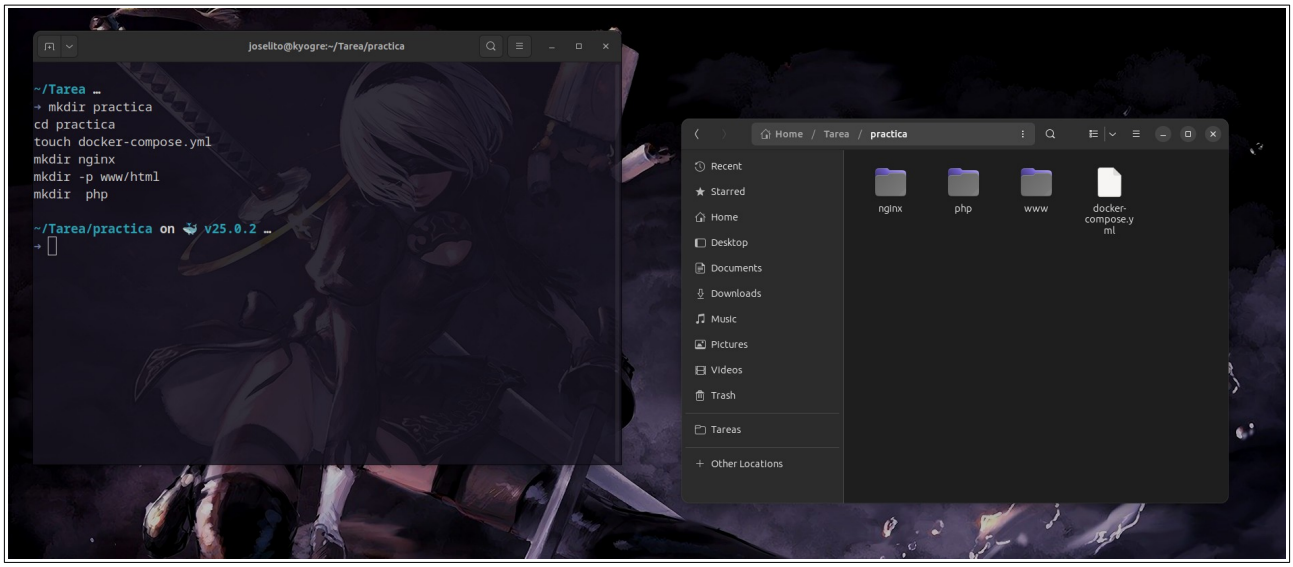
Index

Crear la raíz

Vamos a usar docker-compose, contenedores y una referencia “física” dentro del ordenador, un repositorio en el que tendremos todos los archivos que se reflejaran en los contenedores.

Empezemos creando una carpeta en la que crearemos ya una carpeta de nginx con su archivo de configuración y el docker-compose.yml en el que descargaremos todas las imagenes que queremos.

mkdir practica
cd practica
touch docker-compose.yml
mkdir nginx
mkdir -p www/html
mkdir php



Una vez con el entorno ya preparado, vamos a ir paso a paso modificando cada archivo.

Nginx

De nginx tendremos dos archivos, el default.conf y el Dockerfile

touch nginx/default.conf nginx/Dockerfile

En el default.conf meteremos la configuración del server de nginx que usaremos para el contenedor de nginx, este será el contenido del archivo:

```
server {

    listen 80 default_server;
    root /var/www/html;
    index index.html index.php;

    charset utf-8;

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

    location = /favicon.ico { access_log off; log_not_found off; }
    location = /robots.txt { access_log off; log_not_found off; }

    access_log off;
    error_log /var/log/nginx/error.log error;

    sendfile off;

    client_max_body_size 100m;

    location ~ .php$ {
     fastcgi_split_path_info ^(.+.php)(/.+)$;
     fastcgi_pass php:9000;
     fastcgi_index index.php;
     include fastcgi_params;
     fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
     fastcgi_intercept_errors off;
     fastcgi_buffer_size 16k;
     fastcgi_buffers 4 16k;
    }

    location ~ /.ht {
     deny all;
    }
   }
```

Y el Dockerfile lo usaremos pillar el contenido de esta carpeta, especificamente el default.conf:

```
FROM nginx:latest
COPY ./default.conf /etc/nginx/conf.d/default.conf
```

Php

En php solo meteremos un dockerfile para pillar las extensiones que necesitaremos para php a la hora d descargarlas en el contenedor de php:

```
FROM php:7.0-fpm
RUN docker-php-ext-install pdo_mysql
```

Contenido

Al final aquí estamos preparando el contenido web que se mostrará, en esta ocasión un index.php.

Con el comando **touch www/index.php**, crearemos el archivo y le meteremos el contenido:

```
    <!DOCTYPE html>
    <head>
     <title>¡Hola mundo!</title>
    </head>

    <body>
     <h1>¡Hola mundo!</h1>
     <p><?php echo 'Estamos corriendo PHP, version: ' . phpversion(); ?></p>
     <?
      $database ="mydb";
      $user = "root";
      $password = "secret";
      $host = "mysql";

      $connection = new PDO("mysql:host={$host};dbname={$database};charset=utf8", $user,
$password);
      $query = $connection->query("SELECT TABLE_NAME FROM
information_schema.TABLES WHERE TABLE_TYPE='BASE TABLE'");
      $tables = $query->fetchAll(PDO::FETCH_COLUMN);

       if (empty($tables)) {
         echo "<p>No hay tablas en la base de datos \"{$database}\".</p>";
       } else {
         echo "<p>La base de datos \"{$database}\" tiene las siguientes tablas:</p>";
         echo "<ul>";
           foreach ($tables as $table) {
             echo "<li>{$table}</li>";
           }
         echo "</ul>";
       }
       ?>
   </body>
```

</html>

Aquí estamos pillando las tablas que de default se generan, en esta ocasión, de un contenedor de mysql que pillaremos en el siguiente punto, es opcional, si quieres omitir lo de la base de datos perfectamente lo puedes quitar (si ni vamos a usar php, si creeis mejor, cread un archivo index.html):

```
    <!DOCTYPE html>
    <head>
     <title>¡Hola mundo!</title>
    </head>

    <body>
     <h1>Hola pibes</h1>
   </body>
</html>
```

Descargar los contenedores

Ahora, en la raíz vamos a modificar el docker-compose.yml con el comando **nano docker-compose.yml**, y le meteremos el siguiente contenido:

```
services:
  nginx:
    build: ./nginx/
    container_name: nginx-container
    ports:
      - 80:80
    links:
      - php
    volumes_from:
      - app-data
  php:
    build: ./php/
    container_name: php-container
    expose:
      - 9000
    links:
      - mysql
    volumes_from:
      - app-data

  app-data:
    image: php:7.0-fpm
    container_name: app-data-container
    volumes:
      - ./www/html/:/var/www/html/
    command: "true"

  mysql:
    image: mysql:5.7
```

```
    container_name: mysql-container
    volumes_from:
      - mysql-data
    environment:
      MYSQL_ROOT_PASSWORD: secret
      MYSQL_DATABASE: mydb
      MYSQL_USER: myuser
      MYSQL_PASSWORD: password

  mysql-data:
    image: mysql:5.7
    container_name: mysql-data-container
    volumes:
      - /var/lib/mysql
    command: "true"
```

Aquí al final estamos creando tres contenedores, uno para las funcionalidad de php y modificar ejecutar el contenido de www, el nginx para mostrarlo en el navegador como servidor, y una base de datos, al final la base de datos es algo opcional, pero se añade aquí para mostrar que puedes meterla por ahí.

A través de ello, ahorita hacemos el siguiente comando en la raíz para levantarlo todo:

docker-compose up -d

Y descargaremos todos los contenedores y levantaremos.

Al final estamos usando el port 80 sin especificar ninguno otro, así que en navegador pongamos la url de **http://localhost** y veremos que el proyecto funciona correctamente.