CFGS ADMINISTRACIÓN DE SISTEMAS INFORMÁTICOS EN RED

BIENVENIDOS!

Manuel Domínguez.

mftienda@gmail.com

@mafradoti

www.linkedin.com/in/mftienda

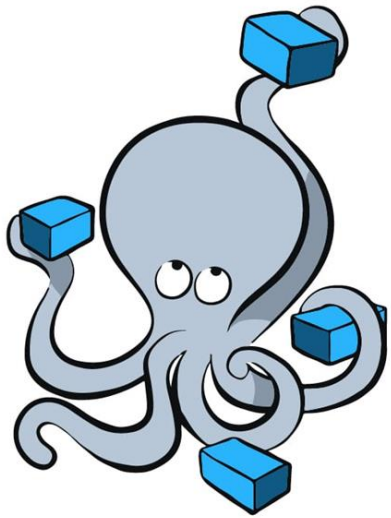
https://github.com/mftienda

Ud.- Docker compose

Índice

1.- Introducción

Escenarios multi-contenedor



docker
Compose

1.- Introducción

Hasta ahora nuestras aplicaciones eran muy simples, y sólo ejecutaban un proceso.

Pero en la mayoría de los casos las aplicaciones implica más de un proceso (microservicio).

Por ejemplo, imaginemos que queremos montar un Wordpress. Necesitaremos:

Servidor Web:Apache
Servidores de bases de datos: MYSQL
Servidor aplicaciones: PHP

¿Cómo podemos hacerlo?

2.- ¿Qué es Docker compose?

Docker Compose es una herramienta que nos facilita la creación de escenarios multicontenedores.

Actualmente se está trabajando con la **versión v3**, que nos permite trabajar tanto en local como en la nube.

Utilizaremos un archivo YAM (“yamel”) para definir los servicios (contenedores), así como sus dependencias.

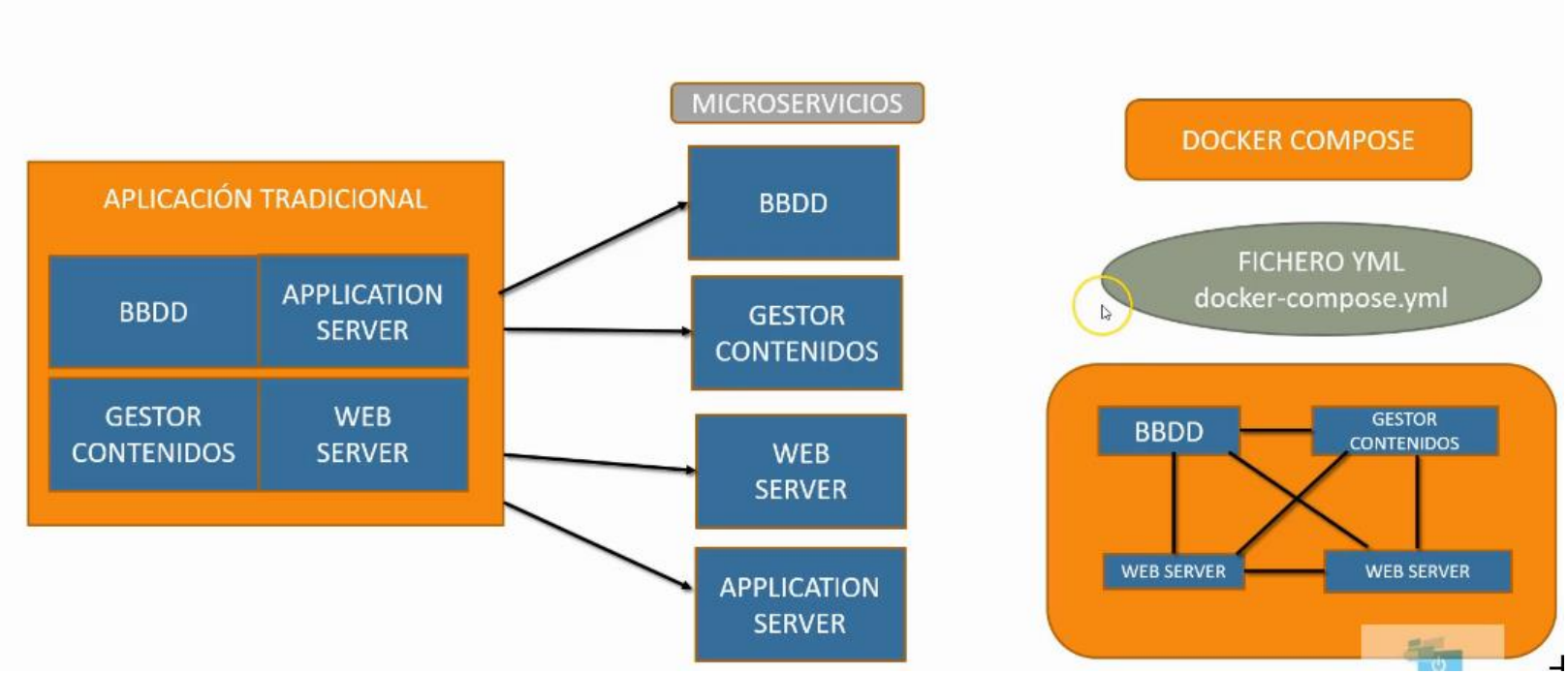
Con Docker Compose, podemos levantarlos o pararlos todos los servicios con un solo comando.

2.- ¿Qué es Docker compose?

Docker compose es un orquestador (desplegador de contenodores).

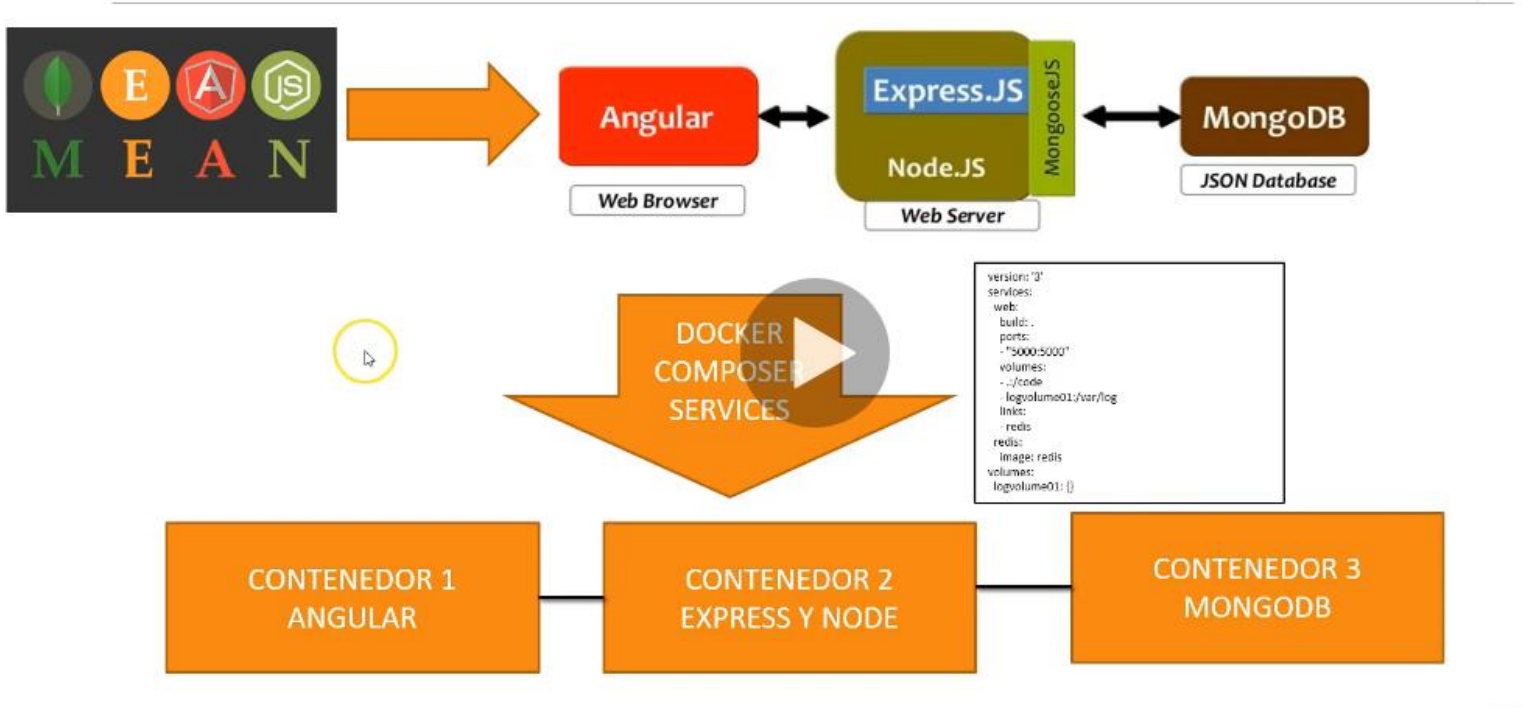
De forma sencilla despliega y relaciona los contenedores.

Es un frontal sobre Docker.



2.- ¿Qué es Docker compose?

Ejemplo:
MEAN → Entorno de desarrollo: Desarrollo de aplicaciones.



3.- Instalar Docker Compose

Referencias:

Consultamos el github la versión estable.
https://github.com/docker/compose/releases

En la página oficial de docker, podemos ver los pasos:
https://docs.docker.com/compose/install/

Instalación:

1.- Nos bajamos la versión de Docker:
sudo curl -L "https://github.com/docker/compose/releases/download/1.27.4/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

3.- Instalar Docker Compose

Instalación:

2.- Le damos permisos al archivo que nos acabamos de bajar:

chmod +x /usr/local/bin/docker-compose

3.- Comprobamos que está instalado:

docker-compose --version

```
root@debian:~# docker-compose --version
docker-compose version 1.26.2, build eefe0d31
root@debian:~#
```

4.- Ficheros YAML

YAML (“Yamel”) es un formato para guardar objetos de **datos con estructura de árbol**. Sus siglas significan YAML Ain’t Markup Language (YAML no es otro lenguaje de marcado).

La sintaxis es relativamente sencilla.

Se utiliza normalmente para:

Archivos de configuración

Traducciones

Representar información

4.- Ficheros YAML

¿Por qué utilizar un YAML en vez de un JSON/XML ?

Un formato mucho más amigable

Fácil de entender rápidamente

Facilita el mapeo de estructuras de datos complejas.

Es **case sensitive** y normalmente lo encontraremos con la extensión .yml o incluso .yaml.

Existen unas reglas generales que deben cumplirse en un documento YAML. Algunas de ellas son:

4.- Ficheros YAML

La estructura del documento se denota **indentando con espacios en blanco (utilizaremos dos espacios).** No se permite el uso de tabuladores para indentar.

Los miembros de las listas se denotan encabezados **por un guion (-)** con un miembro por cada línea, o bien entre corchetes ([ ]) y separados por coma espacio (,).

Los vectores asociativos se representan usando los dos puntos seguidos por un espacio. en la forma "**clave: valor"**, bien uno por línea o entre llaves ({ }) y separados por coma seguida de espacio (,).

Los **comentarios vienen encabezados por la almohadilla (#)** y continúan hasta el final de la línea.

4.- Ficheros YAML

```yaml
version: '2'

services:
    nginx:
        image: nginx:alpine
        ports:
            - 80:80
        restart: always
        volumes:
            - app-volume:/var/www/html
        depends_on:
            - php
    php:
        build: .
        ports:
            - 9000:9000
        restart: always
        volumes:
            - app-volume:/aplicacion

volumes:
    app-volume:
```

services: se define los contenedores. “nginx” y “php”.

Servicio **‘php’** : utiliza un archivo Dockerfile que se encuentra también en la carpeta raíz del proyecto (**build:** .).

Abrimos el puerto 9000.

Servicio ‘**nginx**’: utilizaremos la imagen **nginx:alpine**.

Abrimos el puerto 80 y especificamos **que depende del servicio ‘php’,** por lo que no se arrancará este contenedor hasta que no esté ejecutándose el otro.

4.- Ficheros YAML

```yaml
version: '2'

services:
    nginx:
        image: nginx:alpine
        ports:
            - 80:80
        restart: always
        volumes:
            - app-volume:/var/www/html
        depends_on:
            - php
    php:
        build: .
        ports:
            - 9000:9000
        restart: always
        volumes:
            - app-volume:/aplicacion

volumes:
    app-volume:
```

Para compartir información entre los contenedores se ha creado un volumen:

app-volume

4.- Ficheros YAML

Versión de yaml y Docker:
https://docs.docker.com/reference/

File Format → Compose file → #docker version

Compose file format	Docker Engine release
3.8	19.03.0+
3.7	18.06.0+
3.6	18.02.0+
3.5	17.12.0+
3.4	17.09.0+
3.3	17.06.0+
3.2	17.04.0+
3.1	1.13.1+
3.0	1.13.0+

5.- Procedimiento

Procedimiento:

1.- Nos creamos una carpeta para guardar el proyecto.
Es importante el nombre que le demos a la carpeta, porque después se construye el proyecto basándose en ese nombre.

2.- Creamos el fichero: **docker-compose.yml**

3.- Lo arrancamos:

#docker-compose up → Aparece los logs en pantalla

#docker-compose up -d → Segundo plano

6.- Prácticas

Práctica 1: nginx

1.- mkdir nginx

2.- Creamos el docker-compose.yml

```
version: '3'
services:
  nginx:
    image: nginx
    ports:
    - "80:80"
```

3.- Lo lanzamos: **#docker-compose up**

6.- Prácticas

Práctica 1: nginx

4.- Ha creado una red llamada: **nginx_default**

```
root@debian:~/nginx# docker-compose up
Creating network "nginx_default" with the default driver
```

5.- Ha creado un contenedor: **nginx_nginx_1**

```
Creating nginx_nginx_1 ... done
Attaching to nginx_nginx_1
```

6.- Con Attaching se asocia al primer contenedor.

6.- Prácticas

Práctica 1: nginx

7.- Comprobaciones docker:

#docker ps → Vemos el contenedor

docker network ls → Vemos la red

#docker network inspect ID → Podemos ver los contenedores asociados a la red

localhost:80

6.- Prácticas

Práctica 1: nginx

8.- Comprobaciones docker-compose: **DESDE DEL PROYECTO**

#docker compose ENTER → Vemos todos los comandos.

9.- #docker-compose ps → Vemos los procesos

10.- Paramos el proceso: CTRL+C

11.- Lo arrancamos de nuevo en 2º plano: **#docker-compose up -d**

12.-Podemos hacer las comprobaciones anteriores.

13.- Pararlo: **#docker-compose stop** → No hace falta ID. Estamos dentro del proy.

Resumen

1.- ¿Qué es Docker compose?

2.- ¿Qué es un fichero YAML?

Sugerencias/mejoras del tema

Sugerencias /mejoras del tema

Dudas