Creación de entornos locales Drupal con Docker en Ubuntu y Windows 10

Este documento pretende resumir los pasos básicos necesarios para montar un entorno local de un proyecto Drupal mediante el uso de Docker. Las instrucciones detalladas a continuación son un ejemplo genérico y tendrán que adaptarse a las necesidades particulares de cada proyecto.

1 Instalación de Docker y Docker Compose

Instalación en Ubuntu

Los siguientes comandos para la instalación de Docker y Docker Compose están definidos para Ubuntu 18.04. Deberá revisarse si la sintaxis es la correcta y válida para versiones anteriores y posteriores de la distribución, en particular el que añade el repositorio APT, ya que se especifica que es para bionic, el nombre de la versión 18.04.

En primer lugar, se actualiza la lista existente de paquetes que se tengan en Ubuntu:

```
sudo apt update
```

A continuación, en caso de que no estén ya instalados en la máquina, se instalan unos paquetes que permitirán al gestor de paquetes de Debian, apt, emplear paquetes mediante HTTPS:

```
sudo apt install apt-transport-https ca-certificates curl
software-properties-common
```

El siguiente paso es añadir la clave GPG para el repositorio de Docker oficial:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo
apt-key add -
```

Y, ahora, se añade el repositorio APT de Docker. Este comando es susceptible de cambio, ya que en él se especifica la arquitectura del sistema (amd64, suele ser la más común) y la versión de Ubuntu:

```
sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu bionic stable"
```

Con el repositorio ya añadido, es necesario actualizar de nuevo la lista de paquetes para que Ubuntu lo detecte:

```
sudo apt update
```

Ahora, antes de continuar con la instalación, conviene comprobar que Docker se va a instalar del repositorio que se le ha indicado, y no del que viene por defecto en Ubuntu. Para ello, ejecutamos el siguiente comando:

```
apt-cache policy docker-ce
```

Tras ejecutarlo, debería de obtenerse un resultado como el siguiente:

```
docker-ce:
  Installed: (none)
  Candidate: 18.03.1~ce~3-0~ubuntu
  Version table:
     18.03.1~ce~3-0~ubuntu 500
        500 https://download.docker.com/linux/ubuntu bionic/stable
amd64 Packages
```

Como se puede ver, se indica que Docker (docker-ce, la Community Edition), no está instalado, pero la versión candidata para descargar será la del repositorio oficial, que es la que se ha indicado en el comando anterior donde hemos añadido el repositorio APT.

Si todo es correcto, se procede a la instalación de Docker:

```
sudo apt install docker-ce
```

Una vez la instalación finalice, se puede comprobar que se ha instalado correctamente y el servicio está ejecutándose:

```
sudo systemctl status docker
```

La salida de este comando debería de indicar que el servicio de Docker está activo y en ejecución (running). Además, el servicio se habrá configurado para ejecutarse automáticamente al inicio de Ubuntu.

Ahora que Docker ya está instalado, sólo queda instalar Docker Compose, una herramienta que ayuda en el despliegue de aplicaciones mediante la instalación y ejecución de un conjunto de contenedores de Docker de una sola vez. Para ello, primero descargamos la versión más actual que haya disponible:

```
sudo curl -L
"https://github.com/docker/compose/releases/download/1.24.0/docker
-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

En este caso, se especifica la versión 1.24.0, que es la más nueva en el momento de redacción de este documento. Se puede especificar la que se quiera.

Finalmente, se le aplican los permisos de ejecución necesarios al binario que se ha descargado:

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

Tras la instalación, se puede comprobar si se ha instalado correctamente ejecutando:

```
docker-compose --version
```

Es recomendable, además, ejecutar el siguiente comando para crear un enlace simbólico al binario de Docker Compose que permita ejecutar comandos sin permisos de sudo:

```
sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

Para que surta efecto, será necesario reiniciar bash (cerrar y abrir la consola debería de bastar).

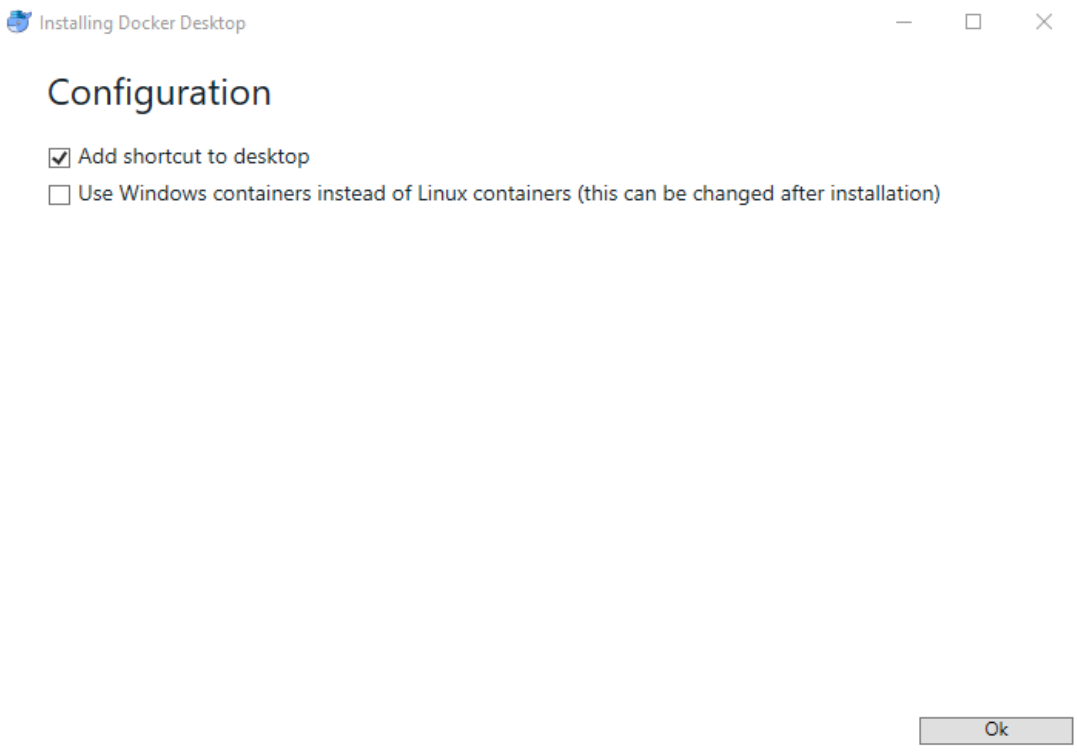
Instalación en Windows 10

Antes de proceder a la instalación en Windows 10 es necesario tener dos requisitos del sistema en cuenta: en primer lugar, que la virtualización tiene que estar activada en la BIOS del sistema. Habitualmente viene activada por defecto, o ya se tiene activada para el uso de VirtualBox, así que no debería de ser un problema. Si no, será necesario investigar la forma de acceder a la BIOS de cada sistema (varía en función del fabricante) y navegar por las opciones de ésta para encontrar las opciones de virtualización.

Y, en segundo lugar, también será necesario activar la opción de Hyper-V de Windows 10. No es necesario saber cómo hacerlo, ya que el propio Docker lo preguntará tras la instalación en una ventana emergente. No obstante, algo muy importante a tener en cuenta: si se activa Hyper-V (requisito indispensable para Docker), VirtualBox dejará de funcionar. Es decir, si se tienen otros entornos virtualizados, por ejemplo, con Vagrant, dejarán de funcionar mientras Hyper-V esté activo.

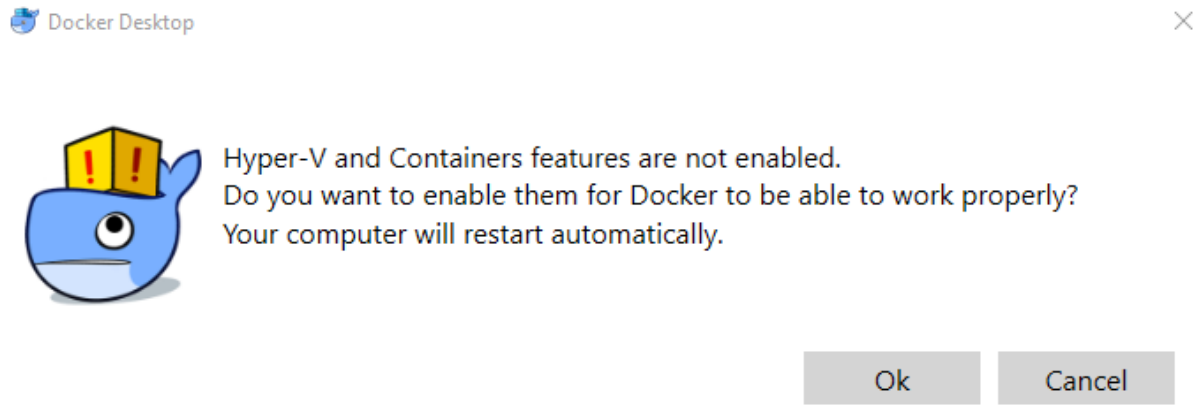
Una vez comprobados esos dos requisitos, se puede proceder con la instalación. Docker cuenta con un conjunto de herramientas para Windows 10 agrupadas en el paquete “Docker Desktop for Windows”, entre las que se encuentran Docker Community Edition y Docker Compose. Puede descargarse desde esta URL: https://hub.docker.com/editions/community/docker-ce-desktop-windows. Será necesario crearse antes una cuenta en Docker Hub para poder realizar la descarga.

Una vez descargado, tan sólo hay que lanzar el ejecutable y el wizard de instalación se ocupará de realizar todo lo necesario. En primer lugar, aparecerá la siguiente pantalla:

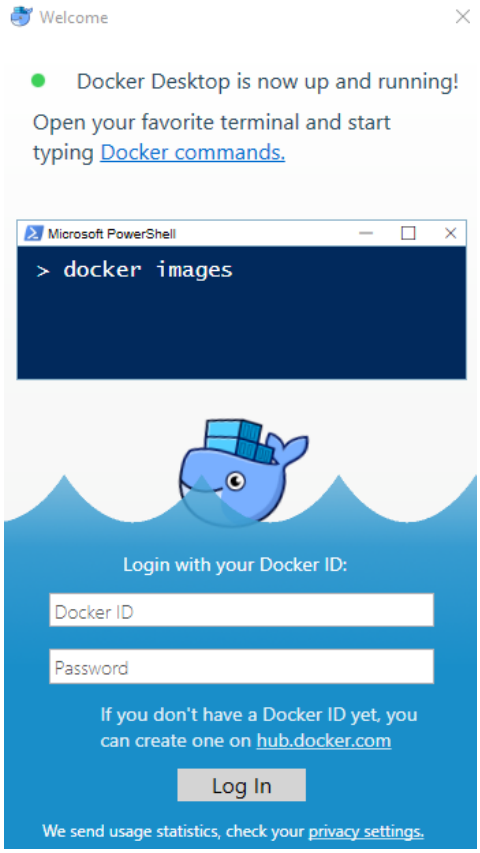


La primera opción puede marcarse o no si se quiere un acceso directo a Docker en el escritorio. La segunda, por otra parte, es recomendable dejarla sin marcar para que Docker utilice los contenedores de Linux, disponibles en Windows 10.

A continuación, Docker procederá con la instalación, y, una vez finalice, pedirá que se cierre la sesión y se vuelva a abrir para completar ciertas configuraciones. Tras hacerlo y volver a iniciar sesión, Docker empezará a configurarse, momento en el que aparecerá la ventana emergente para activar Hyper-V en el caso de que no se encuentre activado:



Como indica en la ventana, tras activarlo, Windows se reiniciará. Finalmente, tras volver a encenderse e iniciar sesión, Docker finalizará su configuración y aparecerá la siguiente ventana, que indicará que Docker se encuentra ya en funcionamiento:



La ventana puede cerrarse sin ningún problema, y Docker se iniciará automáticamente cada vez que se encienda el PC.

2 Configuración del entorno

Antes de crear el entorno, es necesario preparar y configurar unos pocos ficheros y directorios, donde se alojará. Es necesario destacar que, en este punto, hay dos formas de proceder en función de qué proyecto se vaya a crear: uno totalmente nuevo, o uno ya existente. A lo largo de esta sección, se destacarán aquellos pasos que sean diferentes para cada uno de los dos escenarios.

En primer lugar, se creará un nuevo directorio para el proyecto, con el nombre que se quiera. Dentro de este directorio, si se va a crear un Docker para **un proyecto ya existente,** será necesario crear otro directorio, por ejemplo, “www”, “web” o un nombre similar, donde se alojará el Drupal. También se creará este directorio si se va a crear un proyecto nuevo pero, por el motivo que sea, con una versión del Core que no sea la más nueva, donde habrá que descargar dicha versión y copiarla en el.

Una vez creado, se puede copiar el proyecto completo dentro del mismo. En caso de ser un Drupal 8, con su estructura de carpetas donde existe en un primer nivel los ficheros de composer, de Drush, etc., se copiarán igualmente todos los ficheros. Más adelante, se podrá definir el directorio docroot donde se encuentra realmente el Drupal.

A continuación, toca meterse en harina con la parte que puede resultar un poco más compleja en la creación de entornos con Docker, pero que es igualmente sencilla: los ficheros docker-compose.yml y .env. Ambos ficheros pueden descargarse tanto del repositorio de GitHub del stack Docker4Drupal (https://github.com/wodby/docker4drupal), o del repositorio de Hiberus (https://github.com/hinternet/drupalstack).

Fichero .env

Este fichero, como su nombre indica, contiene una serie de variables de entorno que se van a poder emplear en el docker-compose.yml para indicar ciertos parámetros que permitirán a Docker Compose crear y lanzar los contenedores que nosotros queramos.

Muy importante para Windows 10: debido al explorador de archivos de Windows, este fichero aparecerá como “env”, sin extensión. Se deberá de usar un editor de ficheros, como PHPStorm o Notepad++ para renombrar el fichero como “.env”, ya que el explorador de Windows no le permite. De no hacerlo, Docker Compose no encontrará las variables definidas en éste.

En primer lugar, se encuentran las propiedades generales del proyecto (project settings). En ellas, se puede definir el nombre del proyecto (que luego se empleará para nombrar los contenedores) y la URL de nuestro entorno local (habitualmente, “nombreDelProyecto.localhost”).

Además, también han de especificarse los datos relativos a la base de datos del proyecto; esto es, el nombre, el usuario y contraseña, el host y el driver. Aquí es **muy importante** que los datos que se especifiquen aquí coincidan totalmente con los que se encuentran en el fichero settings.php del proyecto.

Por lo general, sólo se trata de poner en este fichero los mismos datos que hay especificados en el settings.php; sin embargo, la variable del host será diferente, ya que habitualmente lo tenemos

configurado como “localhost” (ya que la base de datos suele encontrarse en el mismo servidor o máquina donde se encuentra el Drupal) y, ahora, tendremos que especificarlo como “mariadb”, de forma que Docker sepa que la base de datos está en el contenedor donde se está ejecutando el servicio de la base de datos MariaDB.

Una vez especificados estos datos generales del proyecto, todas las demás variables que siguen en el fichero especifican las versiones a descargar de los diferentes servicios que podrá necesitar el proyecto.

Aquí, cada proyecto tendrá sus necesidades, por lo que es tarea del desarrollador que esté montando el entorno (al menos, del primero que lo haga) el comprobar las versiones necesarias de cada servicio e indicarlas en este fichero. Sólo es necesario definir las versiones de aquellos servicios que se quieran para el proyecto, y no es necesario comentar (#) todos aquellos que no se vayan a usar, ya que sólo se usarán las variables que se quieran en el docker-compose.yml. Sí es necesario comentar todas las variables del mismo nombre, ya que lógicamente sólo puede existir una definida por servicio.

Como se puede observar, las versiones especificadas tienen un número más largo de lo habitual en la mayoría de los casos. Esto se debe a que los servicios van a descargarse desde los repositorios de GitHub creados específicamente para el stack de Docker4Drupal, cuyo listado puede encontrarse en el Readme.md del repositorio principal, y las versiones de dichos servicios se han definido como “versionDelServicio-tagDeGit”. Por ejemplo, para la versión 10.3 de MariaDB, la variable del fichero .env se especifica como “10.3-3.4.2”. Las versiones y tags disponibles pueden encontrarse en cada uno de los repositorios, o en las páginas del Docker Hub en el caso de servicios oficiales de Docker, como Portainer o Traefik.

Cabe destacar, sin embargo, que la versión indicada de Drupal no indica versión del Core y un tag, sino la versión de Drupal (7 u 8) y, seguido del guión, la versión de PHP. Como se puede ver, no permite especificar una versión concreta del Core de Drupal, por lo que se descargará la última versión disponible para Drupal 7 u 8. Es por ello que, como se mencionaba al principio de esta sección, este tag sólo se empleará si se quiere crear un entorno para un proyecto nuevo.

También como elemento a diferenciar del resto en este fichero, existe una sección para definir versiones para Solr. Como se puede observar, hay dos tags definidos, un “SOLR_TAG” y un “SOLR_CONIG_SET”. El primero es simplemente la versión de Solr, que habrá que comprobar igual que el resto de servicios para verificar cuál será la correcta para el proyecto. El segundo, sin embargo, hace referencia directa al módulo Search API Solr Search de Drupal (nombre máquina seach_api_solr”). Es decir, aquí se está indicando la versión del módulo que se instalará y con la que se configurará el servicio de búsqueda de Solr.

Fichero docker-compose.yml

En este segundo fichero se encuentra toda la configuración de los servicios que Docker Compose va a crear y ejecutar para el proyecto, creando así un entorno funcional local para el mismo. El fichero contiene bastantes servicios definidos, la mayoría de ellos comentados (#), ya que no se van a necesitar para el funcionamiento básico de un proyecto Drupal. No obstante, como ya se ha

comentado anteriormente, esto son unos ficheros básicos que pueden modificarse libremente según las necesidades del proyecto.

Como se puede observar, es aquí donde se hace uso de las variables definidas en el fichero .env, de forma que Docker Compose sepa qué versiones descargar de cada uno de los servicios, la configuración de la base de datos, y el nombre del proyecto.

Para montar un proyecto básico, sólo será necesario tocar apenas unas líneas del documento para tenerlo todo listo. En primer lugar, el servicio de MaríaDB. Éste, salvo que se quiera cambiar el nombre del contenedor, todas las variables que necesita ya están definidas en el fichero .env y, por tanto, no hay que tocar nada más.

El servicio de PHP es un caso especial en el que hay que diferenciar si se va a crear un entorno para un proyecto nuevo o para uno ya existente. Lo más probable es que el fichero que se obtenga, ya sea del repositorio de Docker4Drupal o del de Hiberus, esté apuntando a una imagen de Drupal, es decir, lo que está indicando a Docker Compose es que tiene que descargar el Core de Drupal y una versión de PHP. Esto sólo va a ser de utilidad si el entorno es para **un proyecto nuevo**, ya que descargará la última versión de Drupal, y además lo hará siempre que se recree el contenedor (por el motivo que sea, aunque generalmente sólo hay que crearlo una vez al principio).

Por tanto, si lo que se quiere es un entorno para un proyecto existente, con su versión del Core específica, es tan sencillo como cambiar la URL que apunta a la imagen de Drupal, por una URL que apunte simplemente a una versión de PHP, ya que será lo único que se va a necesitar. Por ejemplo, puede usarse la siguiente:

```
wodby/drupal-php:$PHP_TAG
```

De hecho, si se está montando un entorno para un proyecto nuevo, es recomendable cambiar la URL de la imagen de Drupal por la de PHP **después** de haber creado ya el contenedor con el Drupal, de forma que si hay que recrear el contenedor por el motivo que sea, Docker Compose no volverá a descargar el Core, tal vez con una versión más actualizada que, por un motivo u otro, podría romper el proyecto.

Finalmente, en relación a la configuración del servicio de PHP/Drupal, es necesario definir la ruta donde se alojará el código de nuestro proyecto en el contenedor de PHP y/o donde se descargará el Drupal. Será esta configuración la que mapee la ruta en la máquina local, donde se vaya a tener el Drupal, con la ruta del contenedor para el servicio de NFS. Un ejemplo es el siguiente:

```
volumes:
  -/la/ruta/de/mi/proyecto/www:/var/www/html
```

Como se mencionaba al principio de esta sección, el directorio www será donde se aloje el Drupal, que se encontrará al mismo nivel (el directorio) que los ficheros docker-compose.yml y .env.

Una vez configurado el servicio de PHP, el siguiente que hay que configurar es el de Apache. En primer lugar, será necesario personalizar la variable “APACHE_DOCUMENT_ROOT”. Si el proyecto que se está configurando es un Drupal 7, lo normal es que el Core de Drupal se encuentre

directamente en el directorio “www” que se ha creado al principio de esta sección, y que hemos mapeado en el servicio de PHP. Por tanto, la configuración quedaría así:

```
APACHE_DOCUMENT_ROOT: /var/www/html
```

Si, por el contrario, se tiene configurado un Drupal 8, es habitual que dentro del directorio “www” se encuentre el directorio “docroot”, que será el que contenga el Core de Drupal. Será ahí, por tanto, donde habrá que indicar a Apache que está el document root:

```
APACHE_DOCUMENT_ROOT: /var/www/html/docroot
```

A continuación, es recomendable añadir las dos siguientes líneas bajo la varaible “APACHE_DOCUMENT_ROOT” para evitar los errores de “Gateway timeout” que suelen aparecer durante la primera carga del entorno, tras levantar el Docker, tras la limpieza de caches o durante cualquier tarea que requiera un tiempo de procesamiento considerable (activar un módulo, revisar features, etc.):

```
APACHE_TIMEOUT: 6000
APACHE_FCGI_PROXY_TIMEOUT: 6000
```

Finalmente, al igual que con el servicio de PHP, deberemos configurar la ruta donde se encontrará el código en el contenedor. Dado que vamos a tener el código en la misma ruta, éste estará compartido entre ambos contenedores:

```
volumes:
  -/la/ruta/de/mi/proyecto/www:/var/www/html
```

Los demás servicios definidos no necesitarán, o no deberían de necesitar, ninguna configuración adicional ni cambios, a excepción, en todo caso, del puerto definido en el servicio de Traefik en caso de que el que se ha definido se encuentre ocupado. La recomendación suele ser configurar que el puerto 8000 se mapee al puerto 80:

```
ports:
      - '8000:80'
```

Si se quieren habilitar otros servicios que ya estén definidos en el documento, como Adminer, Mailhog, etc., tan sólo hay que descomentarlos y comprobar que la configuración es la correcta (principalmente, que apunta al tag con la versión correcta del servicio, definida en el .env).

3 Creación del entorno

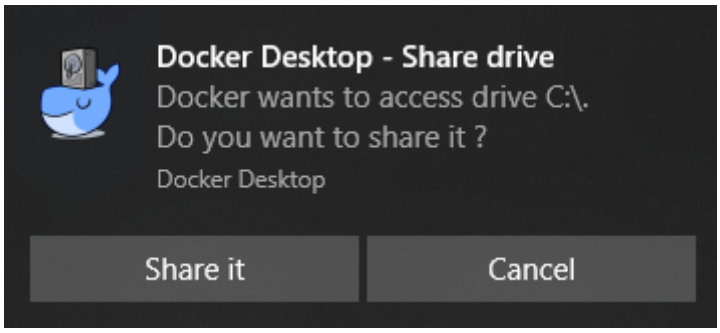
Ahora que ya está todo correctamente configurado, sólo queda crear los contenedores y dejar listo el entorno para usarlo. Docker Compose se encargará de la mayor parte, así que tan sólo es necesario conocer unos pocos comandos básicos para indicarle a Docker Compose lo que queremos hacer. **En el caso de Windows 10**, se recomienda altamente utilizar la consola PowerShell, ya que otras consolas tienen a dar problemas a la hora de ejecutar comandos de Docker o Docker Compose.

Tras navegar con la consola hasta el directorio del proyecto donde se encuentran los ficheros docker-compose.yml y .env, se ejecuta el siguiente comando para crear todos los contenedores definidos:

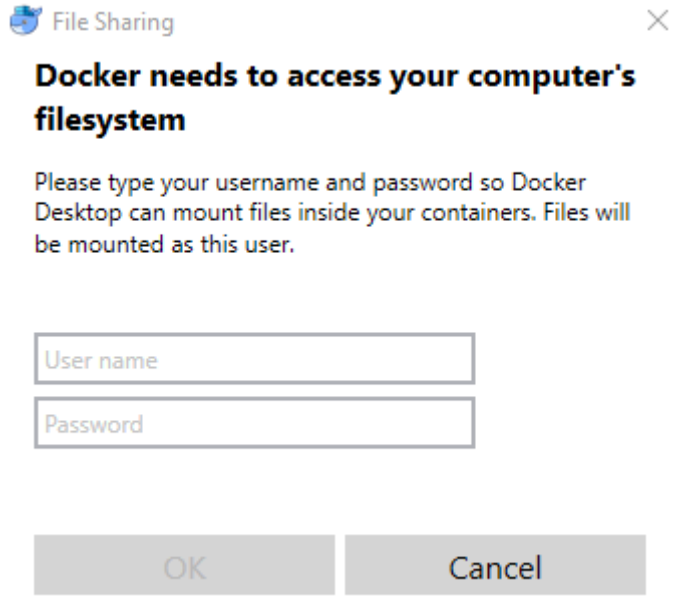
```
docker-compose up -d
```

El proceso podrá tardar unos pocos minutos en completarse, e indicará en la salida de la consola cuando todos los contenedores estén creados y lanzados con un "Done" junto a cada uno de ellos. Esta parte de la creación del entorno es una de las que más problemas puede dar, en el caso de que haya algo mal definido en el fichero docker-compose.yml o alguna versión no existente en el .env. **Es importante que en el fichero docker-compose.yml no haya ningún tabulador (espacios tabulados), sino únicamente espacios simples.**

En el caso de Windows 10, Docker necesitará que habilitéis la compartición de vuestro disco para tener acceso a las rutas definidas en los volúmenes de los servicios PHP y Apache. Para ello, durante la creación de los mismos, os aparecerán dos mensajes como el siguiente en la esquina inferior derecha de la pantalla, como una notificación de Windows:



Tras darle al botón "Share it", para permitir acceder, se solicitará introducir la contraseña del usuario del sistema para confirmar la acción:



Es probable que, debido a esto, los contenedores de PHP y Apache aparezcan como “Error”, en lugar del “Done” mencionado. Simplemente será necesario volver a ejecutar el comando anterior para que Docker intente de nuevo crear los contenedores, lo cual ahora podrá realizar con éxito al tener los permisos necesarios.

Si el proceso ha terminado con éxito, los contenedores estarán funcionando y el entorno estará listo para usarse. Se puede comprobar el estado de los contenedores, así como sus nombres, con el comando:

```
docker-compose ps
```

Para parar los contenedores en cualquier momento, se utilizará el siguiente comando:

```
docker-compose stop
```

De manera homónima, para levantarlos en las subsiguientes ocasiones (el comando up únicamente es para crear los entornos o recrearlos si se hacen cambios en el docker-compose.yml, aunque los levanta al finalizar la creación), se utilizará el comando:

```
docker-compose start
```

La única pieza restante para poder acceder al proyecto a través del navegador web y que todo funcione correctamente, es volcar la base de datos en el contenedor correspondiente, que Docker ya habrá creado con la configuración propocionada en el fichero .env. Si se cuenta con un fichero de backup (sql, mysql.gz, etc), será necesario copiarlo al contenedor de MariaDB:

```
docker cp /ruta/local/al/fichero nombreDelContenedor:/ruta/destino
```

Una vez copiado, será necesario acceder al contenedor para poder hacer el volcado:

```
docker exec -it nombreDelContenedor bash
```

A partir de aquí, los pasos son los mismos que con cualquier otro sistema con una base de datos MySQL o MariaDB:

```
mysql -uUsuario -p
use nombreDeLaBD;
source ficheroConElDump;
```

Esta forma de volcarlo, mediante el comando source, es especialmente recomendable si el nombre de la base de datos definida en MariaDB es diferente del nombre de la base de datos desde donde se ha hecho el dump, ya que a veces puede causar problemas (no por MariaDB, en MySQL también ocurre).

Una vez finalice el volcado, el entorno debería de estar listo para usar. Podrá accederse a este a través de la URL definida en el fichero .env, seguido de dos puntos y el puerto definido en la configuración de Traefik en el docker-compose.yml, por ejemplo:

```
urlProyecto.localhost:8000
```

Anexo I – Problemas habituales

A continuación se listan una serie de problemas que pueden surgir habitualmente durante la configuración y creación de un entorno con Docker, junto con sus posibles soluciones.

Error de sintaxis

Al ejecutar el “docker-compose up -d” para crear los contenedores de un nuevo entorno, es posible que aparezca el siguiente mensaje de error:

```
ERROR: yaml.scanner.ScannerError: while scanning for the next
token
found character '\t' that cannot start any token
  in "./docker-compose.yml", line 101, column 1
```

Esto significa que hay un error de sintaxis en el fichero docker-compose.yml. En este caso, el error, que es probablemente el más común, es que hay un tabulador que no debería. Como se puede observar, indica en qué línea del fichero, por lo que es fácil solucionarlo, eliminando el tabulador y añadiendo espacios si la línea necesita estar indentada.

Error de versión

Al ejecutar el “docker-compose up -d” para crear los contenedores de un nuevo entorno, es posible que aparezca el siguiente mensaje de error:

```
ERROR: manifest for wodby/mariadb:10.1-3.4. not found
```

En este caso, significa que la versión definida para MariaDB en el fichero .env no existe en el repositorio desde donde se está intentando descargar. Será necesario comprobar si es un simple error de sintaxis, como en este caso (falta un 2 al final), o si simplemente la versión no existe. Para ello, se tendrá que visitar el repositorio desde donde se esté intentando descargar el servicio y comprobar las versiones disponibles, habitualmente definidas en el Readme.md del mismo.

Variable no configurada

Al ejecutar el “docker-compose up -d” para crear los contenedores de un nuevo entorno, es posible que aparezca el siguiente mensaje de error:

```
The MARIADB_TAG variable is not set. Defaulting to a blank string.
```

Este error puede aparecer para cualquiera de los servicios definidos en el fichero docker-compose.yml. En este caso, el error es para el servicio de MariaDB. El error puede aparecer por varios motivos distintos:

- La variable no está fijada en el fichero .env, o existe pero está comentada (#).
- El nombre de la variable en el fichero .env y en el fichero docker-compose.yml es diferente.
- El fichero .env no está en el mismo directorio que el docker-compose.yml

- En **Windows 10**, el nombre del fichero no es ".env" sino "env", por lo que habrá que renombrarlo con un editor de texto para que Docker Compose lo reconozca.

Base de datos no encontrada

Si, tras la creación de todos los contenedores correctamente y el volcado de la base de datos aparece un error de MySQL al intentar acceder al portal mediante el navegador, significa que algo no está bien configurado en la base de datos. En este caso, hay que comprobar dos cosas:

- Las variables especificadas en el .env y el settings.php del proyecto son exactamente iguales.
- La configuración de la ruta para el servicio NFS en el fichero docker-compose.yml es la correcta. Podría darse el caso de que haya un error y Docker no esté encontrando la ruta en la máquina local, o podría ser que la ruta sea la correcta, pero esté apuntando a otro proyecto por descuido, o incluso al proyecto correcto, pero con una ruta de otro PC, lo cual puede ser habitual si se está reutilizando el fichero de otro proyecto o alguien lo ha proporcionado y las rutas son relativas a su PC.

Error 403 Forbidden

Si, a intentar entrar en el entorno, ya con todo configurado, a través del navegador y éste indica que no se puede acceder porque no se tienen permisos para ver la ruta "/", significa que el document root de Apache no está bien configurado. Esta es una de las variables que hay que configurar en el servicio del docker-compose.yml para Apache. La ruta ha de indicar dónde se encuentra el Core de Drupal. Es habitual tenerlo mal configurado en función de si es un Drupal 7 u 8, como se ha explicado en el punto 2 del documento.