

Introducción a Docker

UD 07. Caso práctico 01

- Desarrollando con Visual Studio Code en un contenedor



Fons Social Europeu

L'FSE inverteix en el teu futur

Autor: Sergi García Barea

Actualizado Marzo 2023

Licencia



Reconocimiento – NoComercial - CompartirIgual (BY-NC-SA): No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

Importante

Atención

Interesante

1. Introducción	3
2. Visual Studio Code y plugins para Docker	3
3. Creando y editando aplicación en PHP dentro de un contenedor	3
4. Bibliografía	6

UD07. CASO PRÁCTICO 01

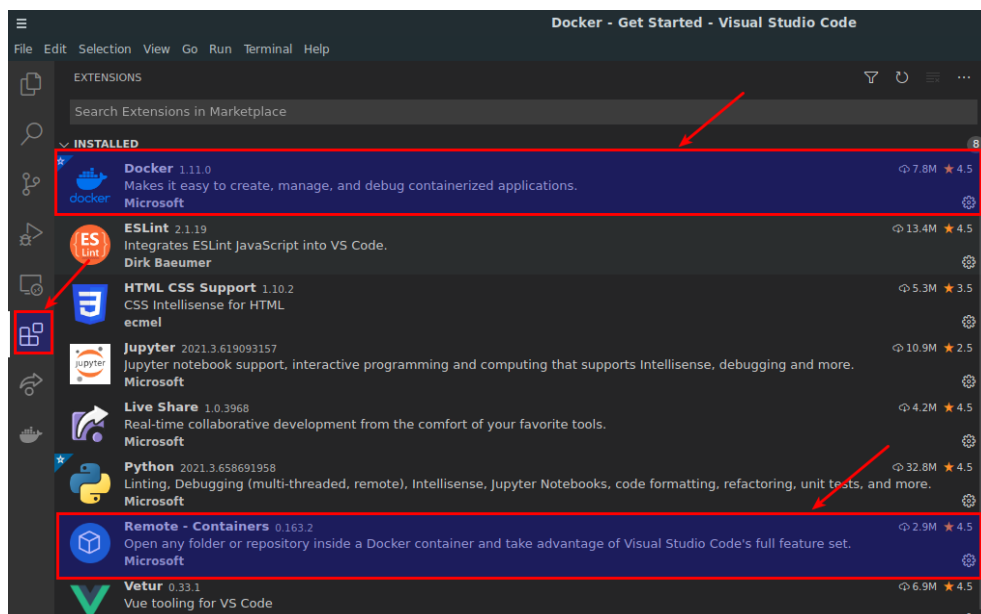
1. INTRODUCCIÓN

En este caso práctico vamos a utilizar el editor multiplataforma “**Visual Studio Code**” junto con plugins relacionados con Docker para poder desarrollar una sencilla aplicación web en un contenedor.

2. VISUAL STUDIO CODE Y PLUGINS PARA DOCKER

Podemos descargar el editor “Visual Studio Code” en <https://code.visualstudio.com/> y en sí no lo conocemos y queremos saber más, podemos acceder a <https://code.visualstudio.com/learn>

Al instalarlo, si detecta Docker instalado en el sistema, el propio editor nos sugerirá una serie de plugins. Estos plugins son los que se pueden ver en la imagen:



Los plugins recomendados son:

<https://marketplace.visualstudio.com/items?itemName=ms-azuretools.vscode-docker>

<https://marketplace.visualstudio.com/items?itemName=ms-vscode.remote.remote-containers>

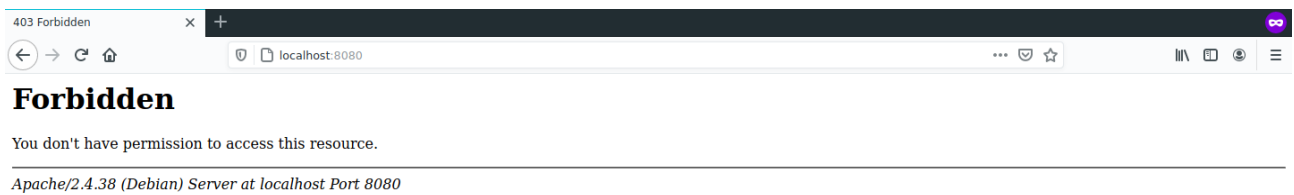
Podemos obtener mucha información de como trabajar con contenedores con Visual Studio Code en <https://code.visualstudio.com/docs/containers/overview>

3. CREANDO Y EDITANDO APLICACIÓN EN PHP DENTRO DE UN CONTENEDOR

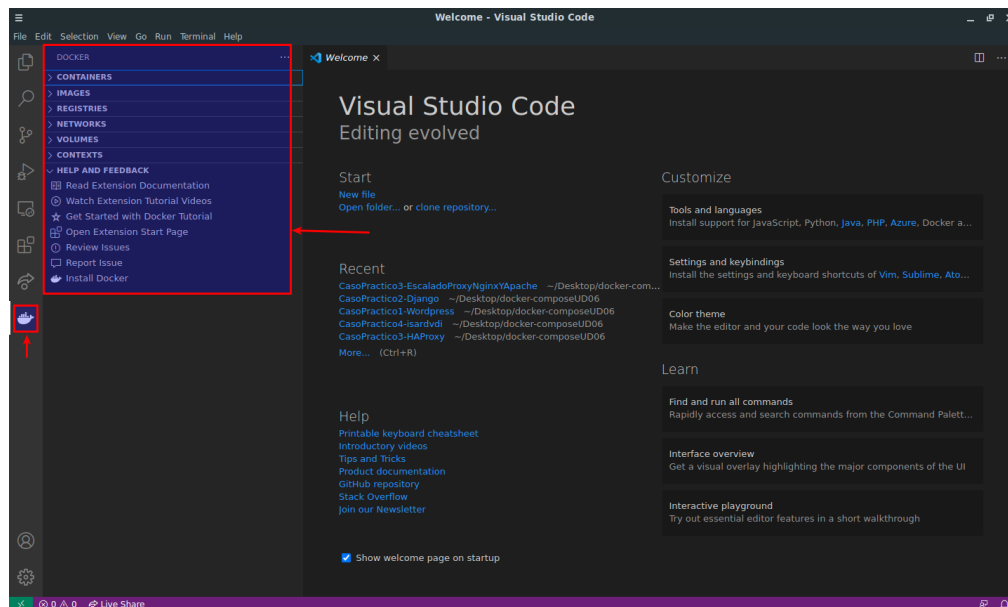
Con el siguiente comando, podemos crear un sencillo contenedor que contendrá “**Apache+PHP**” y en cuyo directorio “**/var/www/html**” se almacena su sitio web, que se servirá por el puerto **8080**.

```
docker run -d --name servidordesarrollo -p 8080:80 php:7.2-apache
```

Una vez creado nuestro contenedor probaremos a acceder a <http://localhost:8080> y observaremos algo similar a:

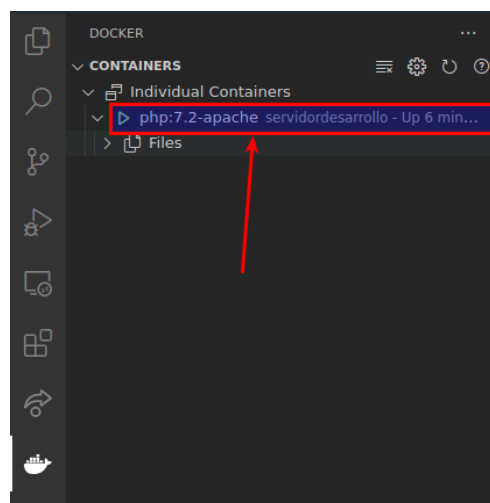


Tras ello, abriremos el editor Visual Studio Code y accederemos a las utilidades Docker (icono de la ballena) y accederemos a un menú con distintas (opciones) tal como se ve en la imagen:



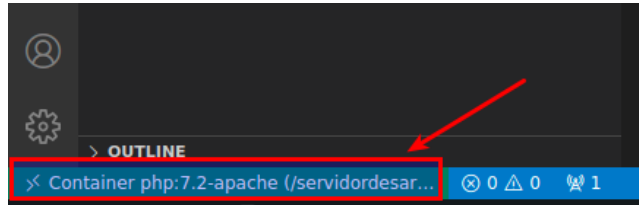
Ahí hay diversas pestañas. Algunas de las más destacadas son **“Containers”** (información de los contenedores), **“Images”** (información de las imágenes disponibles), **“Networks”** (con información de las redes entre contenedores) o **“Volumes”** (con información de los volúmenes creados).

Siguiendo el caso práctico, si desplegamos la pestaña del menú **“Containers”**, veremos todos los contenedores y podremos operar sobre ellos. En la siguiente imagen, vemos el contenedor que hemos creado anteriormente en funcionamiento.



Utilizando el botón derecho, podremos parar/iniciar el contenedor, obtener más información, etc. También podemos observar información, simplemente dejando el cursor encima del contenedor.

Para proseguir con nuestro caso práctico, vamos a pulsar el botón derecho sobre el contenedor y seleccionaremos la opción **“Attach Visual Studio Code”**. Esto abrirá una nueva instancia de Visual Studio Code conectada a dicho contenedor. Podemos comprobar que es correcto observando en la parte inferior de la pantalla de la nueva instancia algo similar a:

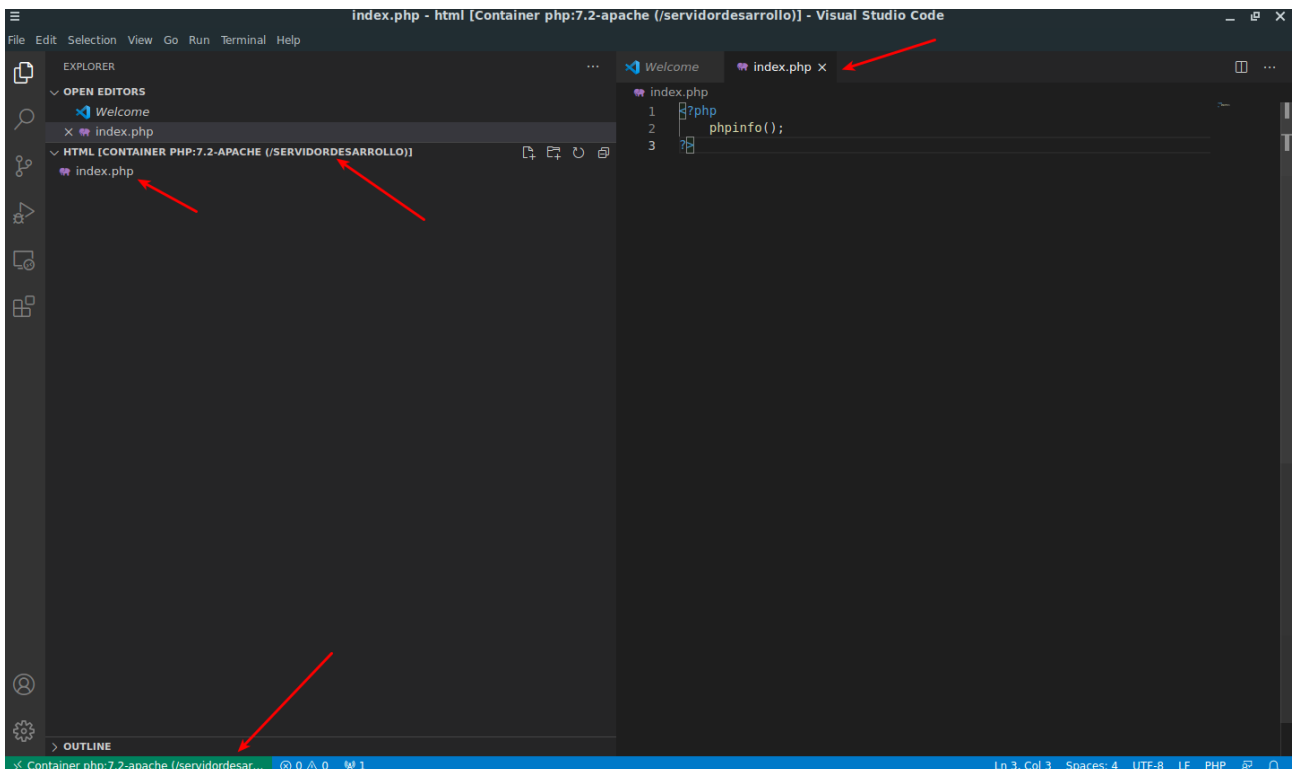


! Atención: este paso utiliza internamente “remote container”. Podríamos haberlo hecho manualmente habiendo pulsado el icono de abajo a la izquierda y habiendo seleccionado a mano el contenedor.

Una vez en esa instancia, podemos abrir el directorio **“/var/www/html”** y lo editaremos como si estuviéramos en local. Ahí crearemos un sencillo fichero **“index.php”** con el siguiente contenido

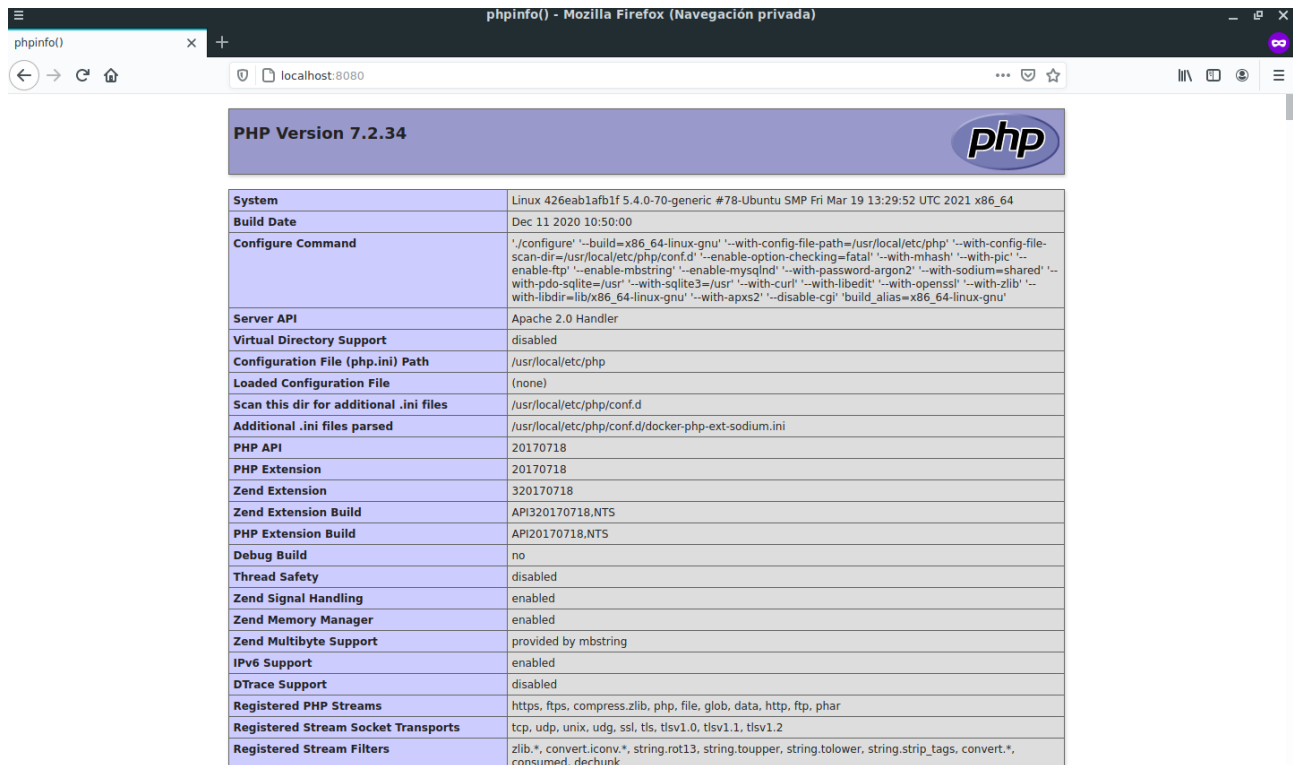
```
<?php
    phpinfo();
?>
```

En el editor nos quedará algo similar a:



Con esto, ya hemos modificado el contenido del contenedor, creando un fichero dentro de la carpeta **“/var/www/html”**.

Si ahora accedemos a <http://localhost:8080> observamos que todo funciona correctamente y que hemos podido desarrollar una aplicación dentro de un contenedor de forma gráfica y sin necesidad de mapear ficheros a nuestra máquina anfitriona.



The screenshot shows a web browser window titled 'phpinfo() - Mozilla Firefox (Navegación privada)' with the address bar showing 'localhost:8080'. The page displays the PHPinfo() output for PHP Version 7.2.34. The output is organized into a table with two columns: the property name and its value.

System	Linux 426eab1afb1f 5.4.0-70-generic #78-Ubuntu SMP Fri Mar 19 13:29:52 UTC 2021 x86_64
Build Date	Dec 11 2020 10:50:00
Configure Command	'./configure' '--build=x86_64-linux-gnu' '--with-config-file-path=/usr/local/etc/php' '--with-config-file-scan-dir=/usr/local/etc/php/conf.d' '--enable-option-checking=fatal' '--with-mhash' '--with-pic' '--enable-ftp' '--enable-mbstring' '--enable-mysqlnd' '--with-password-argon2' '--with-sodium=shared' '--with-pdo-sqlite=/usr' '--with-sqlite3=/usr' '--with-curl' '--with-libedit' '--with-openssl' '--with-zlib' '--with-libdir=lib/x86_64-linux-gnu' '--with-apxs2' '--disable-cgi' 'build_alias=x86_64-linux-gnu'
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/etc/php
Loaded Configuration File	(none)
Scan this dir for additional .ini files	/usr/local/etc/php/conf.d
Additional .ini files parsed	/usr/local/etc/php/conf.d/docker-php-ext-sodium.ini
PHP API	20170718
PHP Extension	20170718
Zend Extension	320170718
Zend Extension Build	API320170718.NTS
PHP Extension Build	API20170718.NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled
DTrace Support	disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2
Registered Stream Filters	zlib.*, convert.iconv.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk

4. BIBLIOGRAFÍA

- [1] Docker Docs <https://docs.docker.com/>
- [2] Visual Studio Code “Working with containers”
<https://code.visualstudio.com/docs/containers/overview>