Lab. 01 Contenedores Docker

Desplegando la aplicación mediawiki

La mediawiki en una aplicación web escrita en PHP que nos permite gestionar una wiki. En este ejemplo vamos a hacer un ejemplo simple de despliegue en contenedor usando la imagen [mediawiki](https://hub.docker.com/_/mediawiki) que encontramos en DockerHub.

En este ejemplo nos vamos a fijar cómo por medio de la etiqueta del nombre de la imágen podemos tener distintas versiones de la aplicación.

En concretos, si estudiamos la [documentación](https://hub.docker.com/_/mediawiki) de la imagen mediawiki, podemos ver las etiquetas disponibles para la imagen que corresponden a versiones distintas de la aplicación.

### La etiqueta **latest**

Si utilizamos el nombre de una imagen sin indicar la etiqueta, se toma por defecto la etiqueta latest que suele corresponder a la última versión de la aplicación. en el caso concreto de mediawiki observamos que la etiqueta latest corresponde a la última versión la 1.37.1. Es más, podemos usar las siguientes etiquetas para indicar la misma versión: 1.37.1, 1.37, stable, latest.

### Las imágenes bases y la arquitectura también son indicadas con las etiquetas

Podemos seguir observando que algunas etiquetas, nos indican además de la versión, los servicios que tienen instalada la imagen, por ejemplo si usamos la etiqueta 1.37.1-fpm estaremos creando un contenedor con la ultima versión de la aplicación pero que además tendrá un servidor de aplicaciones php-fpm para servir la aplicación.

Otro ejemplo: si usamos la etiqueta 1.37.1-fpm-alpine, además de la última versión y que tiene instalado php-fpm, nos indica que la imagen base que se ha usado para crear la imagen es una distribución alpine que se caracteriza por ser una distribución muy liviana.

### Instalación de distintas versiones de la mediawiki

Vamos a crear distintos contenedores usando etiquetas distintas al indicar el nombre de la imagen, posteriormente accedermos a la aplicación y podremos ver la versión instalada:

En primer lugar vamos a instalar la última versión:

docker run -d -p 8080:80 --name mediawiki1 mediawiki

Si accedemos a la ip de nuestro ordenador, al puerto 8080, podemos observar que versión hemos instalado.

##### **Inserta un pantallazo**

**Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente**

A continuación vamos a instalar otra versión de la mediawiki, la 1.36.3, creamos otro contenedor con otro nombre y mapeamos otro puerto:

docker run -d -p 8081:80 --name mediawiki2 mediawiki:1.36.3

Si accedemos a la ip de nuestro ordenador, al puerto 8081, podemos observar que hemos instalado la versión 1.36.3:

##### **Inserta un pantallazo**

**Interfaz de usuario gráfica, Texto, Aplicación

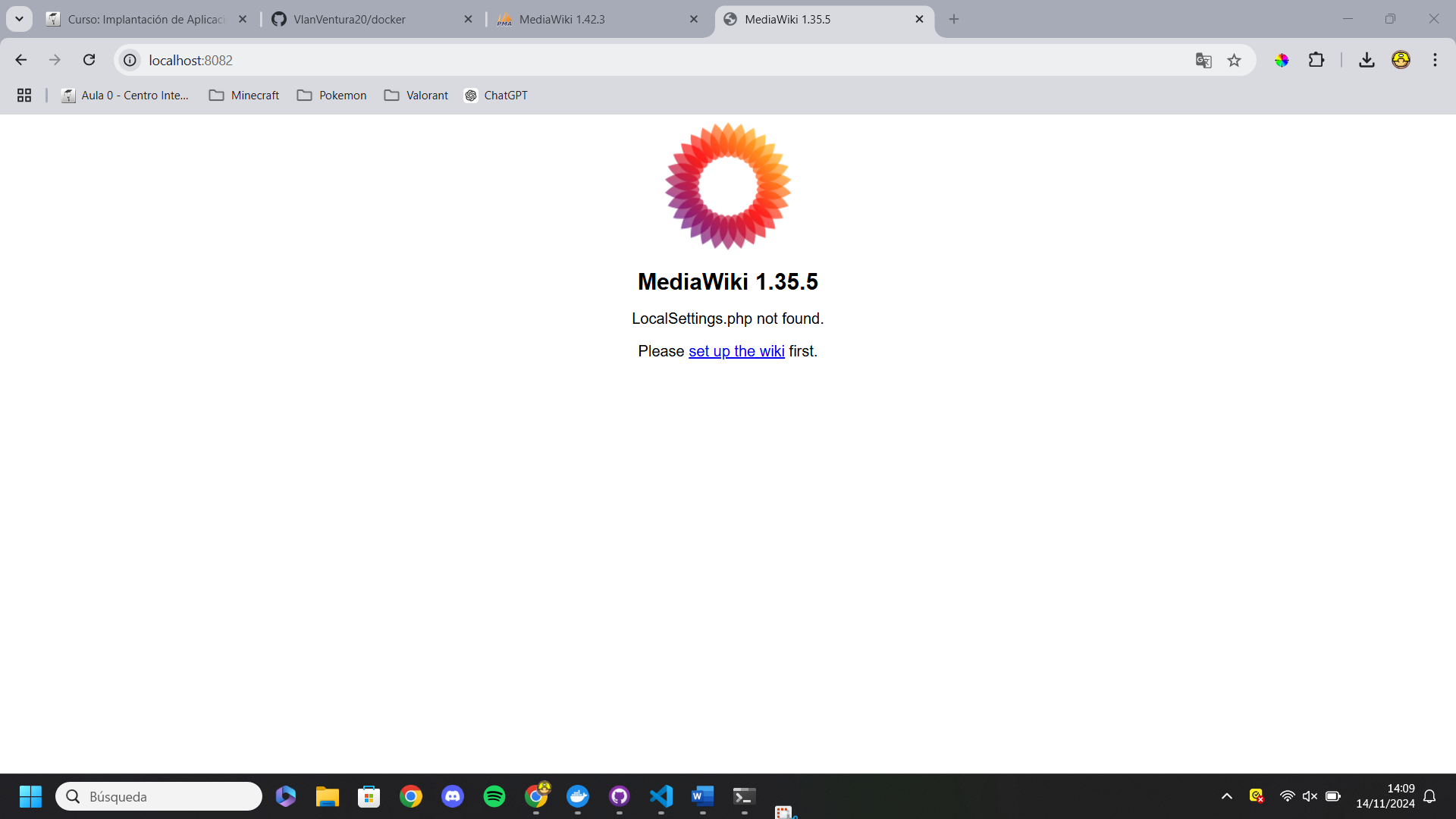
Descripción generada automáticamente**

Y finalmente vamos a instalar otra versión en otro contenedor:

docker run -d -p 8082:80 --name mediawiki3 mediawiki:1.35.5

Si accedemos a la ip de nuestro ordenador, al puerto 8082, podemos observar que hemos instalado la versión 1.31.12:

##### **Inserta un pantallazo**

****

**Nota: Puedes observar que la primera imagen que se baja, descargas todas las capas, sin embargo al descargar las otras versiones de la imagen, sólo se bajan las capas que difieren de la primera.**

Lab. 01 Redes en Docker

# Despliegue el CMS Wordpress

En este ejemplo vamos a desplegar el CMS Wordpress, el creador de páginas web más popular del mundo.

|  |  |
| --- | --- |
|  | Un sistema de gestión de contenidos o CMS es un programa informático que permite crear un entorno de trabajo para la creación y administración de contenidos, principalmente en páginas web. |
|  | **Ejecutar desde la consola de WLS** |

Aquí los comandos necesarios para montar el entorno de Wordpress sobre dos contenedores conectados a la red, red\_wp.

$ docker network create red\_wp

$ docker run -d --name servidor\_mysql \

--network red\_wp \

-v /opt/mysql\_wp:/var/lib/mysql \

-e MYSQL\_DATABASE=bd\_wp \

-e MYSQL\_USER=user\_wp \

-e MYSQL\_PASSWORD=asdasd \

-e MYSQL\_ROOT\_PASSWORD=asdasd \

mariadb

$ docker run -d --name servidor\_wp \

--network red\_wp \

-v /opt/wordpress:/var/www/html/wp-content \

-e WORDPRESS\_DB\_HOST=servidor\_mysql \

-e WORDPRESS\_DB\_USER=user\_wp \

-e WORDPRESS\_DB\_PASSWORD=asdasd \

-e WORDPRESS\_DB\_NAME=bd\_wp \

-p 80:80

wordpress

El contenedor **servidor\_mysql** ejecuta un script **docker-entrypoint.sh** que es el encargado, a partir de las variables de entorno, configurar la base de datos y termina ejecutando el servidor mariadb.

Al crear la imagen **mariadb** han tenido en cuenta de que tiene que permitir la conexión desde otra máquina, por lo que en la configuración tenemos comentado el parámetro **bind-address.**

Del mismo modo el contenedor **servidor\_wp** ejecuta un script **docker-entrypoint.sh**, que entre otras cosas, a partir de las variables de entorno, ha creado el fichero **wp-config.php** de wordpress, por lo que durante la instalación no te ha pedido las credenciales de la base de datos.

Si te das cuenta la variable de entorno **WORDPRESS\_DB\_HOST** la hemos inicializado al nombre del servidor de base de datos. Como están conectada a la misma red definida por el usuario, el contenedor wordpress al intentar acceder al nombre **servidor\_mysql** estará accediendo al contenedor de la base de datos.

Al servicio al que vamos a acceder desde el exterior es al servidor web, es por lo que hemos mapeado los puertos con la opción **-p**. Sin embargo, en el contenedor de la base de datos no es necesario mapear los puertos porque no vamos a acceder a ella desde el exterior. Sin embargo, el contenedor **servidor\_wp** puede acceder al puerto 3306 del **servidor\_mysql** sin problemas ya que están conectados a la misma red.

1. Ejecuta una instrucción docker para visualizar el contenido del fichero `wp-config.php` y verifica que los parámetros de conexión a la base de datos son los mismo que los indicados en las variables de entorno.

##### Agrega un pantallazo

Texto

Descripción generada automáticamente

1. Ejecuta una instrucción docker para comprobar que desde wl `servidor\_wp` podemos hacer ping usando el nombre `servidor\_mysql`. (Tendrás que instalar el paquete `iputils-ping` en el contenedor).

##### Agrega un pantallazo

Texto

Descripción generada automáticamente

1. Visualiza el fichero `/etc/mysql/mariadb.conf.d/50-server.cnf` del contenedor con la base de datos y comprueba cómo está configurado el parámetro `bind-address`.

##### Agrega un pantallazo

Texto

Descripción generada automáticamente

1. Instala Drupal (CMS PHP) siguiendo la documentación de Docker Hub de la aplicación seleccionada.

##### Agrega un pantallazo de los comandos ejecutados

Texto

Descripción generada automáticamente

Lab. 03 Docker Compose

# Moodle – Plataforma educativa de código abierto

Realiza la instalación de Moodle en un contenedor Docker Compose partiendo de la configuración utilizada en el proyecto anterior. *No son necesarias muchas modificaciones.*

La descarga de la aplicación de Moodle la tienes accesible desde aquí. <https://download.moodle.org/releases/latest/>. Si descomprimes el proyecto puedes darte cuenta de que es una aplicación php, *bastante más grande que la anterior*. Por lo que los pasos a seguir van a ser muy similares.

Tienes un documento de los pasos a seguir aquí (<https://docs.moodle.org/401/en/Installation_quick_guide>), pero debes tener en cuenta que la instalación la haces sobre Docker.

##### Genera un manual con los pasos utilizados utilizando <https://www.gitbook.com/>