

Lectura I - Principios de Construcción

martes, 14 de noviembre de 2023
7:04

4.3.4 Principios de Construcción

4.3.5 Principios de Despliegue

Lectura II -Bad Smell

viernes, 17 de noviembre de 2023
22:42

Primero comprender

Para entender la lectura, considere los siguientes términos y sus definiciones:

- Greenability: Degree to which a product lasts over time, optimising the parameters, the amounts of energy and the resources used
- Greenability (in use): Degree to which a software product can be used by optimising its efficiency, by minimising environmental effects and by improving the environmental user perception
- Energy efficiency: Degree of efficiency with which a software product consumes energy when performing its functions
- Efficiency optimisation: Optimisation of resources expended in relation to the accuracy and completeness with which users achieve goals. Relevant resources can include time consumption, software resources, etc.
- User's environmental perception: Degree to which users are satisfied with their perception of the consequences that the use of software will have on the environment
- Minimisation of environmental effects: Degree to which a product or system reduces the effects on the environment in the intended contexts of use.

Imágenes

martes, 21 de noviembre de 2023
17:01

Comandos para imágenes

Listar imágenes

Docker images

Aquí podemos ver los elementos de docker

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
hello-world	latest	9c7a54a9a43c	6 months ago	13.3kB

Descargar imágenes

Primero buscar la imagen que queramos en: [Docker Hub](#)

Docker pull nombreImagen

- Así se descarga la última version
- Pero si queremos una version especifica tomamos
Docker pull nombreImagen : tag
- Siempre tener documentado que versiones son compatibles con las aplicaciones que vamos a usar

Filtrar imágenes

Este comando es válido solo en una terminal gitbash o ubuntu o cualquiera que sea compatible con Linux

Docker images | grep terminoABuscar

Inspeccionar

Este comando nos permite ver cualquiera de los 4 [Elementos](#) de Docker

Docker inspect [elementosDocker](#)

Docker inspect [elementosDocker](#) : tag

Eliminar imágenes

Siempre que debamos eliminar algo primero debemos detenerlo y luego eliminarlo

Docker rmi nombreImagen : tag

0 podemos forzar esa eliminacion

Docker rmi -f nombreImagen : tag

Ejecutar

Run por debajo hace la creación y ejecución de un contenedor en base a la nameImagen, así que por ende si inicializamos una imagen pero sin mencionar el nombre del contenedor se creará un contenedor para que se pueda ejecutar esta imagen

Docker run nameImagen

Container

martes, 21 de noviembre de 2023

21:16

Comandos para contenedores

Creando el Entorno

Crear

Docker create --name nameContenedor nameImagen : tag

- El nameContenedor es opcional, se pone un nombre al azar
- Si creamos un contenedor que requiera una imagen y esta no la tenemos de forma local, pues se descargara

Pero tener en cuenta que no podemos crear a todos los contenedores de esta misma manera ya que a veces podemos tener: [Problemas LOGS](#)

Para de ahí poder [Crear un contenedor con variables de entorno](#)

Listar contenedores ejecutándose

Docker ps

- Solo se verán contenedores ejecutándose

Si queremos ver contenedores que no se ejecutan:

Docker ps -a

- Cada contenedor tiene un status, si esta en 0
- Si eliminamos una imagen sus contenedores no se borran

Podemos buscar de forma más específica

Docker ps | grep nameContenedor

Docker ps | grep nameImagen

Iniciar

Docker start nameContenedor

- El contenedor debe ser previamente [Creado](#)

- Si queremos acceder a los puertos que se inicializaron no podremos ya que esta en la [Red](#) de docker no en la de la computadora main.

Detener

Docker stop nameContenedor

Crear y ejecutarlo inmediatamente

Docker run --name nameContenedor nameImagen : tag

Este comando hace estas 3 cosas:

- [Descarga la imagen](#)
- [La creación del contenedor](#)
- [Inicio del contenedor](#)

Cuando se ejecuta se ejecuta en el Docker no en nuestra maquina anfitrión.

Lo malo de esto es que esto esta vinculado lo cual es un problema.

Crear y ejecutarlo inmediatamente

Docker run --name nameContenedor nameImagen : tag

Cuando se ejecuta se ejecuta en el Docker no en nuestra maquina anfitrión.

Lo malo de esto es que esto esta [vinculado](#) lo cual es un problema ya que el contenedor se está ejecutando en el entorno de Docker, y cualquier recurso que esté vinculado a ese contenedor está "conectado" a él de alguna manera.

Ahora, hablemos de por qué la vinculación podría considerarse un problema:

- **Acoplamiento Rígido:**
La vinculación puede resultar en un acoplamiento rígido entre el contenedor y el recurso externo. Por ejemplo, si un contenedor está vinculado a un puerto específico en la máquina anfitriona, ese puerto estará ocupado y no se puede utilizar para otros fines sin detener y volver a ejecutar el contenedor.
- **Portabilidad Limitada:**
Si un contenedor está fuertemente vinculado a ciertos recursos, como puertos específicos o rutas en la máquina anfitriona, puede volverse menos portátil. Dificulta la migración del contenedor a otro entorno sin ajustar la vinculación.
- **Escalabilidad Limitada:**
La vinculación puede limitar la capacidad de escalar horizontalmente, ya que puede haber conflictos de puertos o dependencias en los

recursos vinculados. Escalar podría requerir una cuidadosa gestión de las vinculaciones para evitar conflictos.

- **Dificultad en Entornos Distribuidos:**
En entornos distribuidos o en la nube, donde los contenedores pueden ejecutarse en diferentes máquinas, las vinculaciones pueden ser difíciles de gestionar y pueden requerir configuraciones específicas.
- **Problemas de Mantenimiento:**
Si hay cambios en la configuración del entorno de ejecución, como puertos ocupados o conflictos de nombres, puede ser necesario ajustar manualmente las vinculaciones. Esto puede aumentar la complejidad y el riesgo de errores durante el mantenimiento.
- Visualizamos lo que pasa a manera de ejecución, pero si no queremos ver el log tenemos

Crear y ejecutarlo inmediatamente sin la Vinculación

```
Docker run -d --name nameContenedor nameImagen : tag
```

Cuando se ejecute solo obtendremos de respuesta el identificador.

Eliminar

```
Docker rm nameContenedor
```

No se puede eliminar contenedores que se estén ejecutando, podemos eliminar de forma forzada

```
Docker rm -f nameContenedor
```

Interactuando dentro del Entorno

Iniciar una sesión shell dentro del contenedor de docker

```
docker exec nameContainer namePrograma0comando
```

Una vez ejecutado el shell no tendremos acceso como tal al terminal, debemos mandar comandos de forma iterativa o sea mandarlos al contenedor, como hacemos eso?

Ejecutar un shell interactivo en un contenedor docker especificado

```
docker exec -i nameContainer sh
```

Una vez ejecutado este comando nuestra terminal quedará esperando a que ingresemos los comandos del shell ya estamos dentro del container

Lo malo de esto es que tenemos una comunicación unidireccional, así que lo que escribamos se ejecutará pero no tendremos respuesta. Pero para ello tenemos:

Finalizar una shell dentro del docker

```
CTRL + Z
```

Básico

Ejecutar un shell interactivo en un contenedor docker especificado - BIDIRECCIONAL

```
docker exec -it nameContainer sh
```

Agregamos -it

Veríamos algo así:

```
PS C:\Users\david\Documents\CyE> docker exec -it container1 sh
/ $
```

Problemas LOGS

Revisar los logs

Primero revisar [Listar contenedores ejecutándose](#) -a y si no vemos a nuestro contenedor ejecutándose y vemos que dice Exited(1) Entonces tenemos problemas ya que no vemos 0, por ende nos hace falta **variables de entornos** que nos sirve para gestionar el entorno

Lo recomendable es siempre leer la documentación del contenedor que vayamos a implementar

```
docker logs nameContainer
```

Se utiliza para ver los registros (logs) generados por un contenedor de Docker específico.

La salida mostrará la información de registro del contenedor, lo que incluye:

- mensajes de la aplicación que se está ejecutando dentro del contenedor
- Errores
- eventos del sistema
- otra información relevante para el funcionamiento del contenedor
- No sirve para [Verificar los logs](#)

Esto puede ser útil para diagnosticar problemas, monitorear la actividad de la aplicación y comprender lo que está sucediendo dentro del contenedor en términos de registro.

Crear un contenedor con variables de entorno

```
Docker run -d --name nameContenedor -e nameVariable=valor nameImagen : tag
```

Esta parte se repite cuantas variables requiramos

```
-e nameVariable=valor -e nameVariable=valor -e nameVariable=valor ...
```

Pero en el momento de crear una variable de entorno que sea una contraseña, se ve la contraseña creada lo cual no es seguro por ende

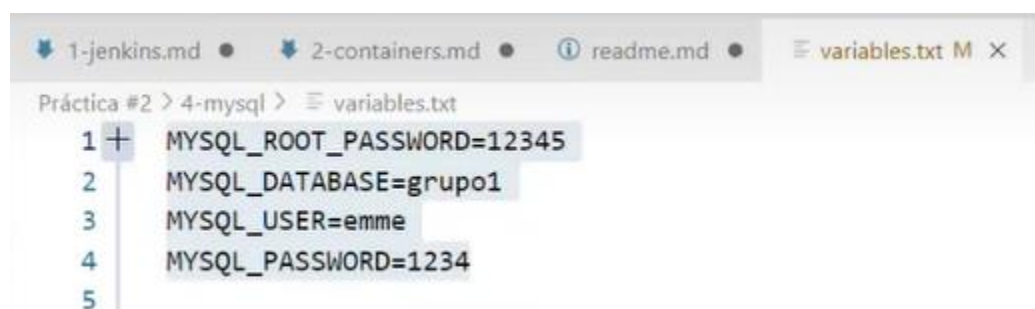
Podemos pasar un archivo que contenga las contraseñas:

Docker Secrets

```
docker run --name nameContainer --env-file=/run/secrets/mysql-root -d nameImagen:tag
```

```
docker run --name nameContainer -e MYSQL_ROOT_PASSWORD_FILE=/run/secrets/mysql-root -d nameImagen:tag
```

- `MYSQL_ROOT_PASSWORD_FILE` es específico para el container de MySQL
- Esta `run/secrets/mysql-root` es un ejemplo de ruta donde tiene un archivo de texto plano que contiene las contraseñas requeridas para un container aquí un ejemplo de mysql



```
1 + MYSQL_ROOT_PASSWORD=12345
2   MYSQL_DATABASE=grupo1
3   MYSQL_USER=emme
4   MYSQL_PASSWORD=1234
5
```

- Tener cuidado con la ruta relativa

Verificar los logs

Primero necesitamos [Ejecutar un shell interactivo en un contenedor docker especificado](#) Luego en el caso de MySQL ingresar:

```
echo nameVariableEntorno
```

Pero esta vaina no tiene el problema de que se visualizan los datos

Contenedor & Redes

Previamente conocer: [Creando Redes](#)

Crear un contenedor vinculado a una red

```
Docker run --name nameContenedor network nameRed -d bridge  
nameImagen : tag
```

Est

Redes

sábado, 25 de noviembre de 2023
15:23

Conociendo el Entorno

Consulta de Ip

Le contenedor debe estar obviamente iniciado

```
Docker inspect nameContenedor
```

El navegador esta externo a la red así que necesitamos hacer lo que se conoce como un mapeo de puertos

Mapeo de puertos

Desde nuestro browser acceder al puerto de nuestro contenedor.

Para evitar puertos que suelen ya usarse, estar pendiente de que puertos se usan.

Si nosotros ya tenemos un contenedor con una imagen ese otro contenedor va a tener un IP con el mismo puerto 80 y yo no puedo acceder con el mismo puerto host a varios

Se debe hacer esto en la parte de Crear el contenedor, para tener los puertos ya definidos

```
Docker run -d --name nameContenedor -p
#PuertoHost:#PuertoContenedor nameImagen : tag
```

- Para saber el `#PuertoContenedor`, es el puerto que maneja la aplicación por default, en <https://hub.docker.com/> podemos buscar dicha aplicación y averiguarlo
- Como el puerto contenedor no varía y puede usarse para varios contenedores entonces la llave para acceder a un contenedor específico viene siendo el número del puerto que usamos en nuestro host

Por seguridad tampoco tener tanto documentado los puertos.

No hay problema si más de un contenedor expone un mismo puerto

Forma más semántica cuando se especifican puertos

```
Docker run --name nameContenedor --publish
published=(host), target=(puerto a mapear con el
contenedor) nameImagen : tag
```

`-d` : se crea y continua ejecutándose
`-p` : publish

Aquí se vuelve más semántica la forma de mapear puertos, pero ahora si queremos mapear más de un puerto debemos copiar y pegar la línea:

```
--publish published=(host), target=(puerto
a mapear con el contenedor)
```

No es lo ideal hacer esto

Organizando el Entorno

Crear una red de tipo bridge

```
Docker network create nameRed -d bridge
```

Est

Inspeccionando Redes

Podemos usar Inspeccionar o

```
Docker network inspect nameRed
```

Vinculando redes

docker network connect nameRed nameContainer

desvinculando redes

docker network disconnect nameRed nameContainer

Listar las redes existentes

Docker network ls

Es

Eliminar I

Docker system prune

Para eliminar contenedores detenidos, redes no usadas, imágenes huérfanas, cache huérfana

Eliminar II

Docker system prune -a

Contenedores detenidos, redes no usadas, imágenes sin vincular a ningún contenedor

f

Docker run --name nameContenedor nameImagen : tag

Est

Proyecto I

sábado, 25 de noviembre de 2023

21:35

EL PROYECTO ES INCREMENTAL hasta el final del semestre!!!!

- La aplicación es web
- Tenemos que aplicar todas las fases de desarrollo del ciclo de software

Aplicaremos la Lógica, esta es solo la parte inicial no el proyecto final.

Transcripción de la parte base del proyecto:

Tengo un dinero invertido en la bolsa de Milán hoy yo llevo es

Pero invertido en la bolsa de New York estoy ahí comprando algunas acción actualmente yo llevo un registro de mis acciones de esta manera en una hoja de cálculo coloco el nombre de la acción la fecha en la que compré la acción el precio por cada acción y la cantidad de acción en mi hoja de cálculo automáticamente con esta información pues yo sé cuál es el costo total de mis acción yo registro todas tus compras de acciones en la bolsa de New York de esta manera como ejemplo les presento 3 registros de diferentes acciones sí de Tesla de Microsoft APL no estoy segura que sí la era pero todas son acciones de la bolsa de New York sí entonces como ya no quiero hacer esto en Excel porque si no tengo mi computadora a la mano no lo puedo colocar quiero hacerlo a través de una aplicación web el proyecto inicialmente consiste en un registro para esa parte sí recuerden que este material se encuentra disponible en el aula virtual para que lo puedan seguir pero como mencioné o era más la narrativa yo registro mis acciones actualmente en una hoja de cal el nombre la fecha de compra el precio y la cantidad son datos que yo voy ingresando manualmente en base a los 2 últimos valores dentro de la hoja de cálculo automáticamente se calcula el costo total de la compra compré 5 acciones de Microsoft por 250 USD cada una total 1250 sí 2 acciones de Tesla cada una me costó 700 USD total 1004 y así yo voy registrando mis acciones en mi hoja de cálculo pero ya no quiero que sea una hoja de cal quiero que cada grupo desarrolle una pequeña aplicación web que me permita realizar la misma mecánica que en este momento acabo de mencionar

Mas requisitos

- Usar MVC, requiere una interfaz y controladores y el modelo.
- El código se analizara con sonarqube

Requerimos hacer documentacion del codigo fuente

Jenkins

domingo, 3 de diciembre de 2023

9:36

Para vero su contraseña via SHELL

- Ir la directorio:

```
/# cat directorio
```

```
cat /var/jenkins_home/secrets/initialAdminPassword
```

Lo que nos arroje es la contraseña

Cuenta

Create First Admin User

Usuario

Contraseña

Confirma la contraseña

Contraseña : 4862

MYSQL - SEVER - DOCKER

domingo, 3 de diciembre de 2023

10:45

```
/ $
```

```
PS C:\Users\david\Documents\CyE> docker run -d --name MIsqL -e MYSQ
```

Paso a paso para visualizar las bases de datos

1. Primero conectarse por la terminal
1. Creamos nuestro usuario ya como nos cante, esta es una forma rápida y de ejemplo nomas:

```
Docker run -d --name mysql-containerI --env-  
file=archivo.txt mysql:8
```

```
PS C:\Users\Marcela\Documents\2023B\ISWD633-GR1\Práctica #2\4-mysql> docker exec
sh-4.4# echo $MYSQL_USER
emme
sh-4.4# mysql -u emme -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.2.0 MySQL Community Server - GPL
```

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

```
mysql> █
```

```
Docker exec -it mysql-containerI sh
```

```
echo $MYSQL_USER
```

```
mrdudu
```

```
mysql -u mrdudu -p
```

```
4862
```

2. Asi ya estamos dentro de mysql terminal, aquí depende con qué tipo de usuario estemos si es root pues tendremos todos los privilegios

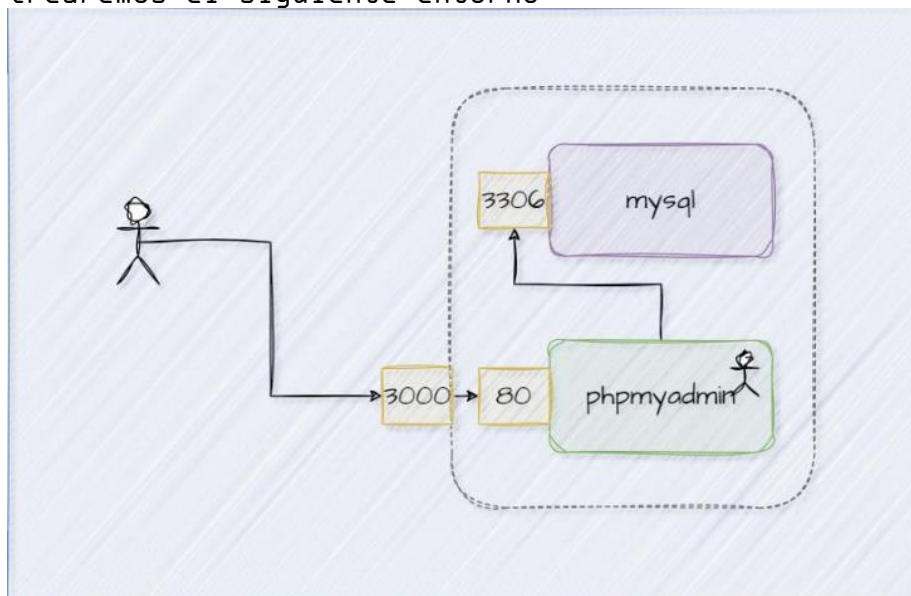
2. Ver la base de datos

Entorno I

domingo, 3 de diciembre de 2023

12:02

Crearemos el siguiente entorno



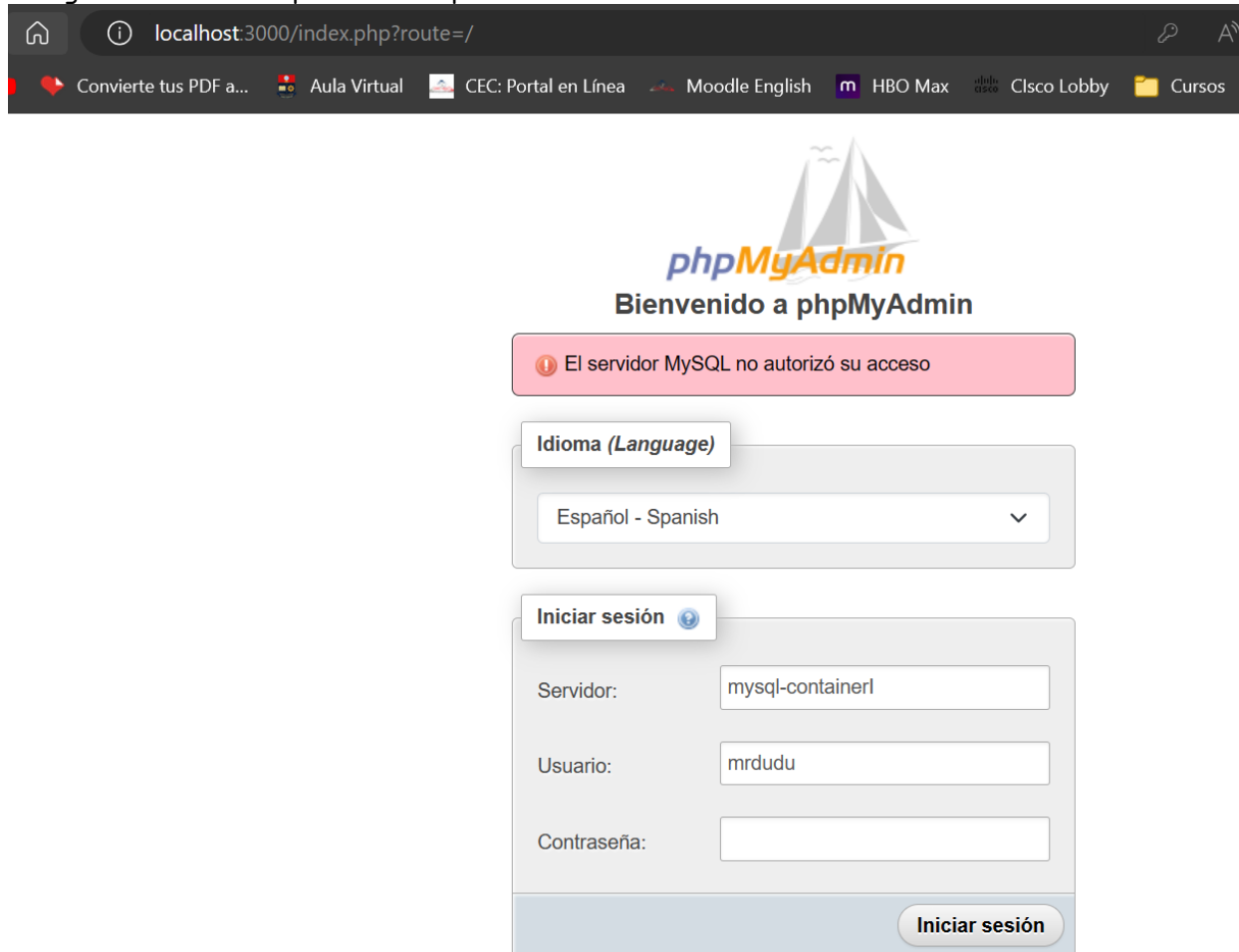
Primero vamos a crear nuestro entorno y tenemos puertos por ende tenemos que hacer un [Mapeo de puertos](#) sin olvidar [Crear un contenedor con variables de entorno](#)

Para crear el phpmyadmin

```
docker run -d --name phpmyadmin-containerI --publish published=3000,target=80 -e PMA_ARBITRARY=1 phpmyadmin
```

Nuestro MYSQL usaremos el visto en [MYSQL - SEVER - DOCKER](#) :
mysql-containerI

Llegados a este punto comprobamos



Pero como vemos si están haciendo conexión entre estos 2 servicios? Pues con ping, pero primero

Consejo

- Dentro del shell del contenedor debemos actualizar siempre, aunque a veces tenemos que instalar ping

En este caso se instalo dentro de la terminal del
phpadmin

```
apt-get update
```

```
apt-get install iputils-ping
```

Comunicación entre dockers

Para hacer ping podemos hacerlo por el nombre del host , puerto, ip

Todo esto usando [Inspecciona](#), pero es mejor usar el ip y con dicho ip, lo mismo para ingresar al phpmyadmin en nuestro browser

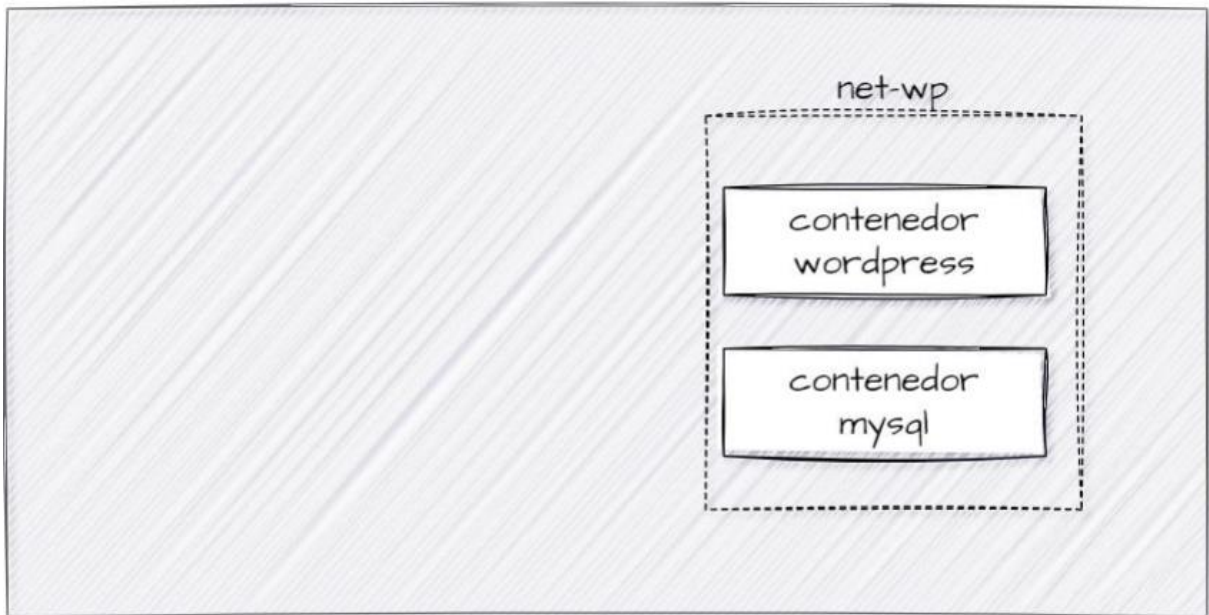
Si queremos hacer que se comuniquen



Ejercicio 2

domingo, 3 de diciembre de 2023

21:38



Creamos red

```
Docker run -d --name ejercicio2 -e WORDPRESS_DB_HOST=mysql-container1 -e WORDPRESS_DB_USER=mrdudu -e WORDPRESS_DB_PASSWORD=12345 -e WORDPRESS_DB_NAME=courses --network net-wp --publish published=9300, target=80 wordpress
```

Est

Una vez creado correctamente veremos wordpress
Podemos hacer modificaciones respecto a las publicaciones pero
sino le atamos a un volumen cuando le eliminemos al wordprees
todos los articulos se perderan, para su persistencia debe
atarse a un volumen