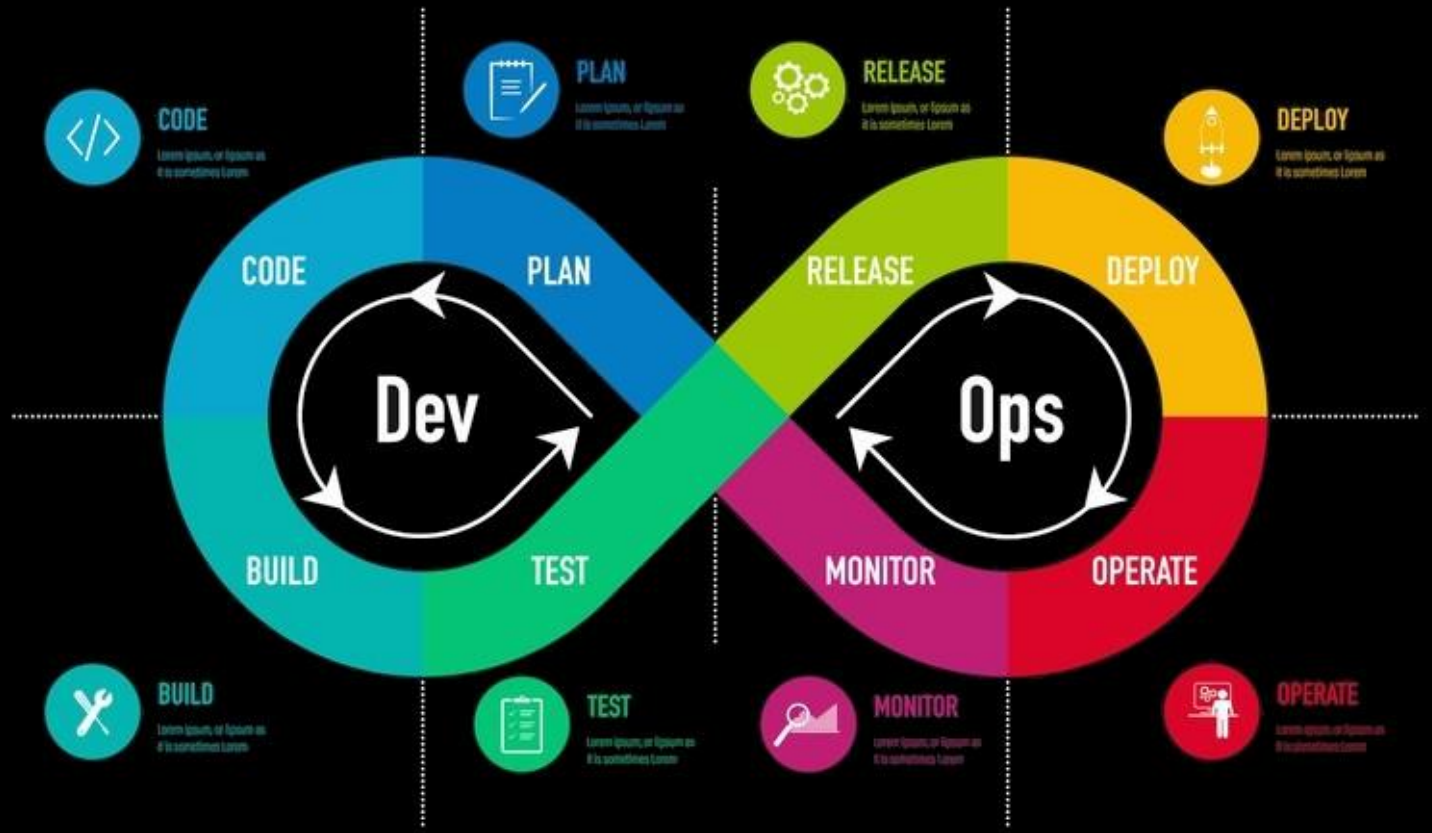
CFGS DAM

IES MONTE NARANCO



UT- 3 DevOps y Cloud Computing

Fundamentos de contenedores

Práctica 2 – Volúmenes y bid mounts

Volúmenes y bid mounts

Por defecto, todo lo que ocurre dentro de un contenedor desaparece cuando el contenedor se elimina.

Para solucionar esto, **Docker** ofrece dos mecanismos de persistencia:

Volúmenes

Sistema de almacenamiento que se crea y gestiona directamente por **Docker**, existiendo de manera **independiente** de los contenedores activos.

Bind Mounts (montajes de enlace)

Permiten montar un directorio específico del sistema de archivos del host directamente en el contenedor. Son útiles para el desarrollo, ya que permiten editar código en el host y verlo reflejado instantáneamente en el contenedor.

Crear un volumen

```
docker volume create <nombre_volumen>
```

docker volumen create mi_volumen

Listar los volúmenes existentes

```
docker volume ls
```

Mostrar información de un volumen

```
docker volume inspect <nombre_volumen>
```

docker volumen inspect mi_volumen

Eliminar un volumen

(Sólo se pueden borrar volúmenes que no estén siendo utilizados por ningún contenedor)

```
docker volume rm <nombre_volumen>
```

docker volumen rm mi_volumen

Borrar todos los volúmenes no usados

El siguiente comando borra todos los volúmenes que no estén siendo utilizados.

```
docker volume prune
```

Ejercicio de volúmenes

- Crea un volumen llamando **volumen001**
- Crea un segundo volumen llamando **volumen002**
- Lista los volúmenes existentes
- Muestra información del volumen **volumen001**
- Borra el volumen **volumen002**
- Borra todos los volúmenes que no estén en uso

Ejercicio de volúmenes (solución)

- docker volume create **volumen001**
- docker volume create **volumen002**
- docker volume ls
- docker volume inspect **volumen001**
- docker volumen rm **volumen002**
- docker volumen prune

Parámetros --mount y -v

- Los parámetros **--mount** y **-v** de **docker run** nos permite gestionar volúmenes y bind mounts.

- **--mount** es la forma más moderna y explícita de montar volúmenes en Docker.

Formato de --mount

```
--mount type=<tipo>,source=<fuente>,target=<destino>,<opciones>
```

1. **type**=<tipo> (**Obligatorio**). Sus posibles valores son los siguientes:

   - **bind:** monta un **directorio o archivo específico** del *host* (tu máquina) dentro del contenedor. Es ideal para desarrollo o configuración.
   - **volume:** monta un **volumen gestionado por Docker** (almacenado en la ubicación de datos de Docker). Es el tipo recomendado para **persistencia de datos** (ej: bases de datos).
   - **tmpfs:** monta un sistema de archivos temporal en la memoria RAM del *host*. Ideal para archivos temporales que no deben persistir.

Formato de --mount

2. **source**=<fuente> (**Obligatorio**). Define la ubicación del recurso en el host (tu máquina) o el nombre del volumen. La sintaxis de este valor depende del valor del parámetro **type**:

   - type=**bind:** la ruta absoluta del archivo o directorio en el host. Si se quiere usar en la ruta una letra de unidad se escribirá entre barras ya que esta es la sintaxis de Docker.

     Ejemplo: source=/c/datos/mysql

   - type=**volume**: el nombre del volumen gestionado por Docker.

     Ejemplo: source=volumen_base_datos

   - type=**tmpfs**: Se omite (no tiene una fuente en el disco).

Formato de --mount

3. **target**=<destino> (**Obligatorio**). Define la ubicación de destino dentro del contenedor donde se montará el recurso.
    - Debe ser una ruta absoluta dentro del contenedor.
    - **No es recomendable que sea una carpeta del sistema** ya que puede fallar el arranque del contenedor.
    - **Ejemplos**:
        - **target**=/var/www/html
        - **target**=/etc/nginx/conf.d

Formato de --mount

4. <**opciones**> (**Opcional**). Permite controlar el comportamiento y los permisos del montaje.

    - **readonly** (o **ro**). Hace que el montaje sea solo de lectura.

Montaje de un volumen

```
docker run --name conte1 --mount type=volume,source=volumen1,target=/var/lib/alpine alpine echo "hola y adios"
```

Si no hemos creado previamente el volumen Docker lo creará al crear el contenedor

(Comprobar en Docker Desktop que se ha creado el volumen y el contenedor)

Montaje de un volumen

Listamos los volúmenes

docker volume ls

- Importante: al finalizar la ejecución del contenedor los volúmenes montados se desmontan automáticamente.
- Docker mantiene anotado que el contenedor usa el volumen.

Montaje de un volumen

Intentamos borrar el volumen

docker volume rm volumen1

No nos deja borrarlo porque el volumen tiene asociado un contenedor que es el que acabamos de crear (comprobarlo visualmente en Docker Desktop)

Montaje de un volumen

Borramos el contenedor

docker rm conte1

Borramos el volumen

docker volumen rm volumen1

Montaje de un volumen

```
docker run --rm --name conte1 --mount type=volume,source=volumen1,target=/var/lib/alpine alpine echo "hola y adios"
```

Añadimos el parámetro --rm para que se borre el contenedor en cuanto termine la ejecución del mismo para ahorrarnos tener que borrarlo cada vez

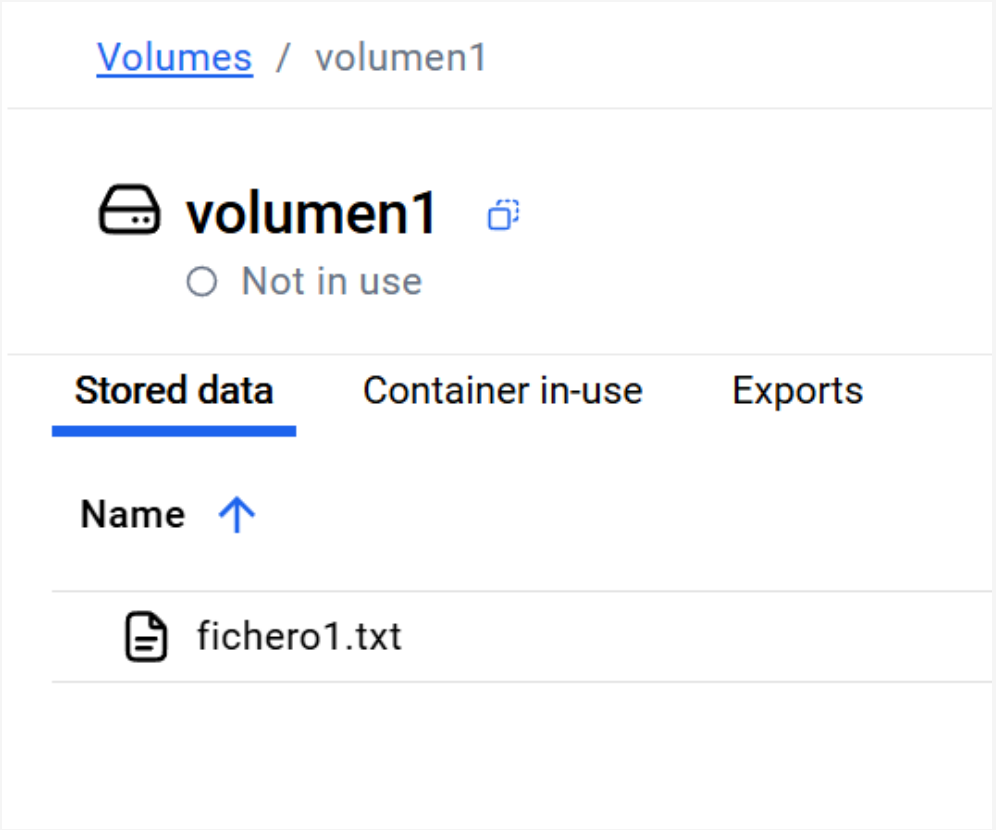
Escribir datos en el volumen

```
docker run --rm --name conte1 --mount type=volume,source=volumen1,target=/var/lib/alpine alpine sh -c "echo 'hola y adios' > /var/lib/alpine/fichero1.txt"
```

- **sh -c** indica que la siguiente cadena es la secuencia de comandos a ejecutar
- Ejecutamos un comando que redirige la cadena ‘hola y adios’ al fichero /var/lib/alpine/fichero1.txt
- La ruta del fichero tiene que ser la que indicamos al crear el volumen para que el fichero quede guardado.

Escribir datos en el volumen

- Podemos consultar el estado del volumen en Docker Desktop y comprobar que se ha creado el fichero.
- Aunque hemos puesto que se borre el contenedor al finalizar la ejecución, la existencia del volumen es independiente del contenedor.



Leer datos del volumen

```
docker run --rm --name conte1 --mount type=volume,source=volumen1,target=/var/lib/alpine alpine sh -c "cat /var/lib/alpine/fichero1.txt"
```

- Ejecutamos el comando cat para mostrar el contenido del fichero

Crear un bid mount con --mount

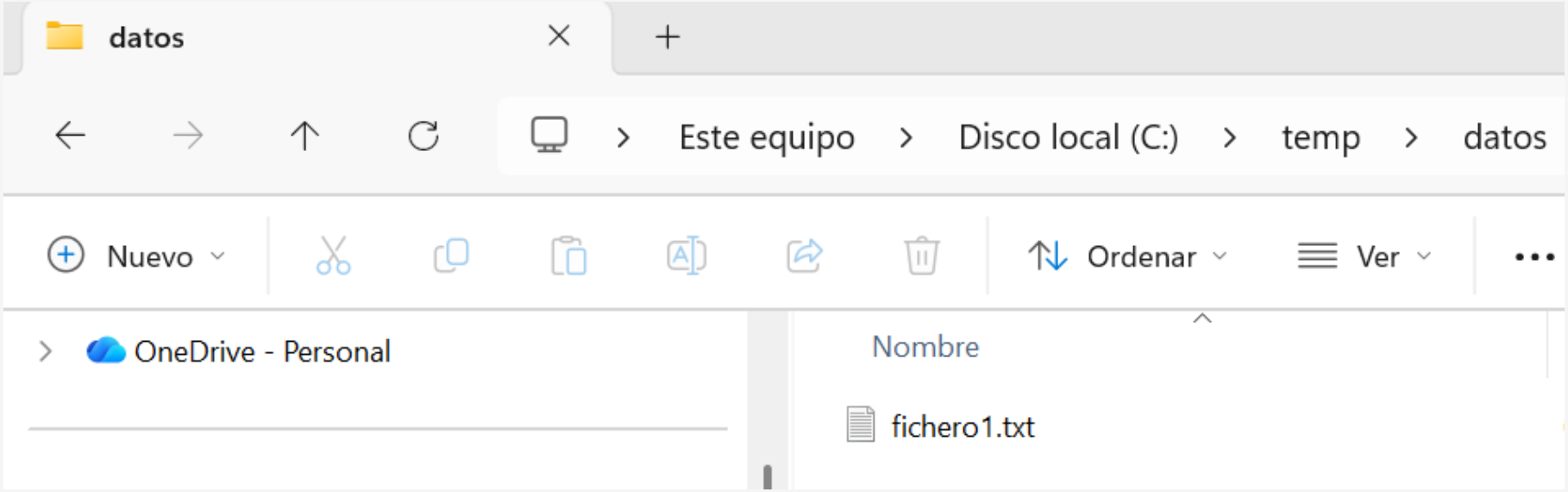
```
docker run --rm --name conte1 --mount type=bind,source=/c/temp/datos,target=/datos alpine sh -c "cat /datos/fichero1.txt"
```

- **type**=bind al estar haciendo un bid mount
- **source**=/c/temp/datos la unidad va entre barras
- **target**=/datos es la carpeta dentro del contenedor

Se produce un error ya que **no existe el fichero fichero1.txt** en la ruta local c:\temp\datos

Crear un bid mount con --mount

Creamos el fichero en la carpeta local de nuestro equipo y le añadimos un texto de prueba:



Crear un bid mount con --mount

Con el fichero creado ejecutamos de nuevo el comando ahora ya se debería mostrar el contenido del fichero.

```
C:\Users\Usuario>docker run --rm --name conte1 --mount type=bind,source=/c/temp/datos,ta
rget=/datos alpine sh -c "cat /datos/fichero1.txt"
esto es una prueba
C:\Users\Usuario>
```