AF6.2.2. Desplegament d'una pàgina web dinàmica. Gestió de Volums

Index

1. Introducció

Una vegada coneguts els elements principals de Docker, com són la gestió d'imatges, contenidors i els components essencials de la seua arquitectura, és moment d'aprofundir en altres aspectes crucials per al desplegament d'aplicacions. En aquesta pràctica **ens centrarem** en **els volums, una ferramenta fonamental per a garantir la persistència de dades** entre diferents desplegaments**.**

Com ja sabem, la informació emmagatzemada dins d'un contenidor és **efímera**. Això significa que la informació existirà únicament mentre el contenidor estiga en execució. Si s'elimina el contenidor, qualsevol dada emmagatzemada en el seu sistema de fitxers intern es perdrà. Aquest comportament planteja diversos problemes, com ara:

1. **Pèrdua de dades crítiques:** Si la informació gestionada per una aplicació (per exemple, els logs, les imatges de perfil pujades o la informació d'una base de dades) s'emmagatzema dins del contenidor, aquesta desapareixerà al eliminar el contenidor.
2. **Dificultat en l'escalabilitat i el manteniment:** En entorns de producció, és habitual necessitar reemplaçar o actualitzar contenidors. Sense un mecanisme per a la persistència, totes les dades es perdrien durant aquests processos.

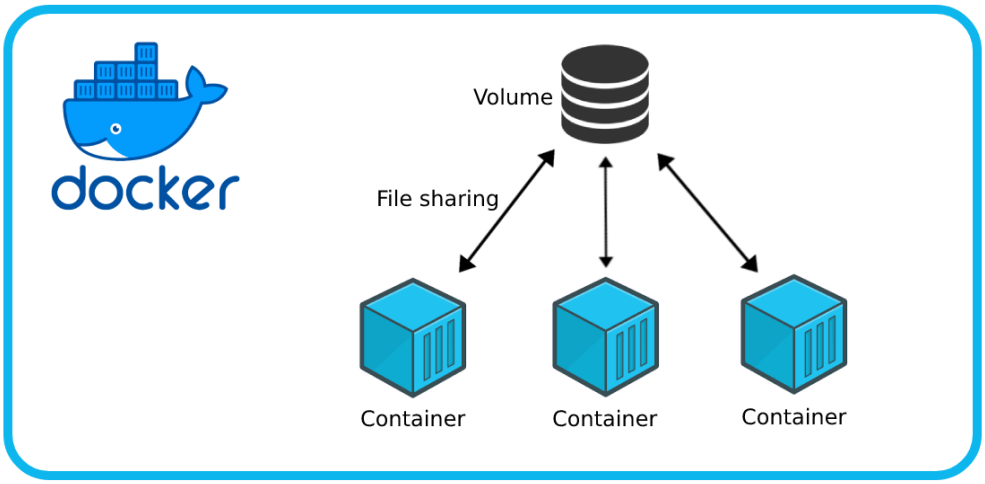
Per un altre costat, no menys important tenim:

3. **Impossibilitat de compartir dades entre contenidors:** Quan diversos contenidors necessiten accedir als mateixos fitxers o dades, es complica garantir la coherència i l'accessibilitat de la informació sense una solució comuna.

2. Què son els volums de docker?

Els **volums en Docker** són un mecanisme d'emmagatzematge que permet separar les dades generades o utilitzades per un contenidor del sistema de fitxers efímer d'aquest. Això fa

que les dades es mantinguen persistents encara que el contenidor siga eliminat o reemplaçat.

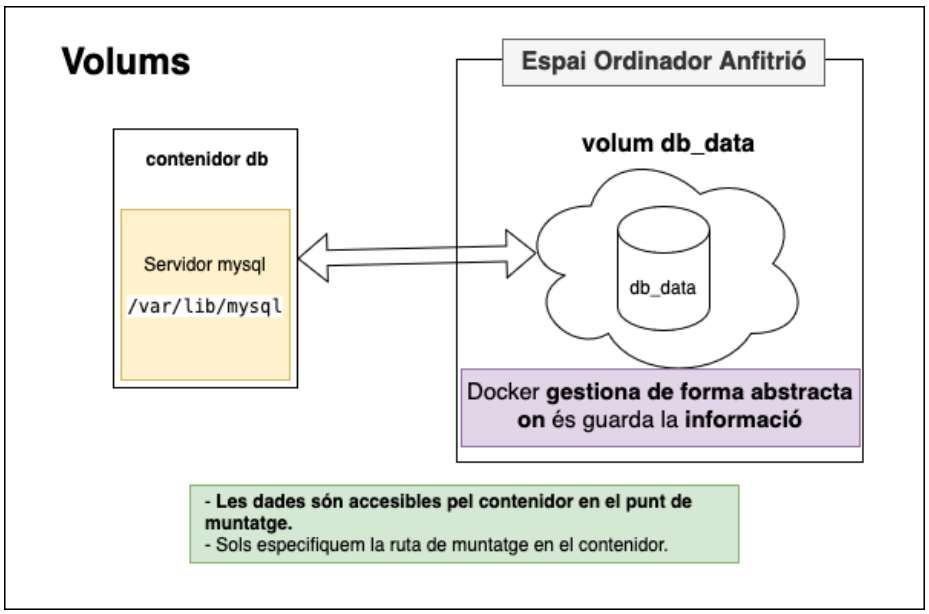


2.1.Tipus de Volums

Docker ofereix, principalment, dos tipus principals d'opcions d'emmagatzematge:

Volums gestionats per Docker:

- Són volums creats i gestionats automàticament per Docker.
- La informació s'emmagatzema en una ubicació específica del sistema host que Docker controla completament.
- Es recomanen per a la majoria de casos d'ús on no és necessari accedir directament a les dades des del sistema host.

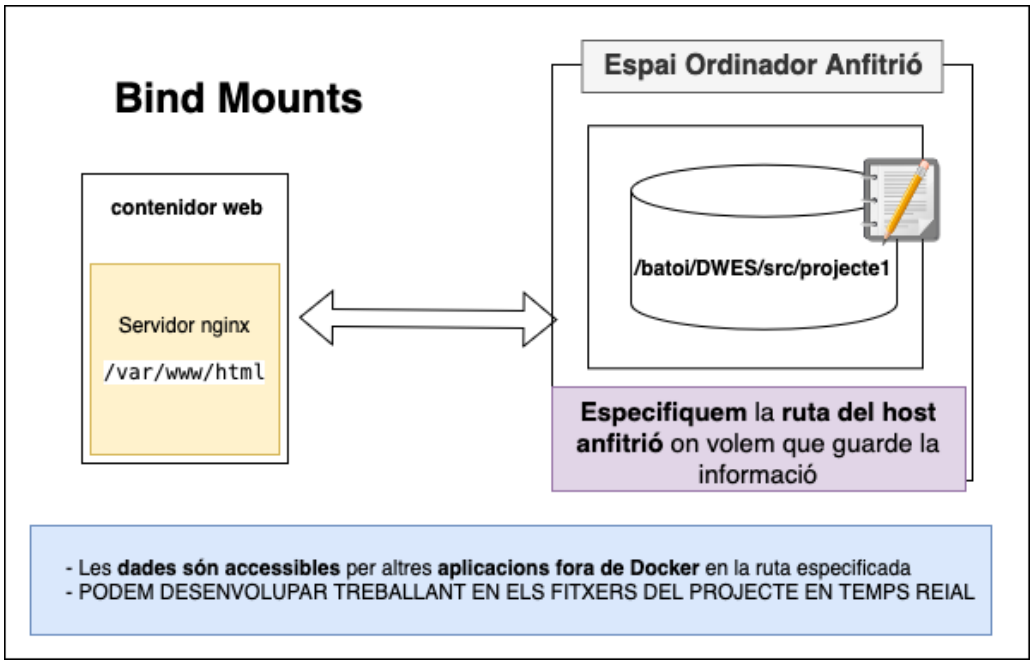


Bind Mounts (Enllaços al sistema host):

- Permeten muntar un directori o fitxer del sistema host dins del contenidor.
- Ideals per a desenvolupament o quan necessites accedir directament a les dades des

del sistema host.

- Requereixen especificar una ruta absoluta del host.



2.2 Client de Docker

2.1.1 Gestió de Volums

Per a **exemplificar l'ús de volums**, crearem un volum que continga les imatges pujades **amb un formulari** d'una **aplicació php**, de forma que aquestes imatges no es perden entre diferents desplegaments a la vegada que diferents contenidors puguen accedir a elles.

Per crear un volum utilitzem l'ordre `docker volume create nom_volum`

```
$ docker volume create upload-data
```

```
➜  ~ docker volume create upload-data
upload-data
```

Una vegada creat el volum podem consultar aquells disponibles a l'entorn de docker amb l'ordre `docker volume ls`

```
➜  ~ docker volume ls
DRIVER    VOLUME NAME
local     upload-data
```

Podem **eliminar el volum del nostre sistema** fent servir el comandament `docker volume rm {VOLUME_NAME}`

```
➜  ~ docker volume rm upload-data
upload-data

➜  ~ docker volume ls
DRIVER    VOLUME NAME
```

Podem eliminar tots els volums que tenim parats amb una única ordre

```
$ docker volume prune
```

```
➜  ~ docker volume prune
WARNING! This will remove anonymous local volumes not used by at least one container.
Are you sure you want to continue? [y/N] y
Total reclaimed space: 2B
```

Has de tindre en compte que per a esborrar un volum aquest no ha d'estar sent utilitzat per cap contenidor, fins i tot si el contenidor està aturat. Docker protegeix els volums actius **per evitar pèrdues accidentals** de dades. Si hem associat el volum a un contenidor, primer haurem de esborrar el contenidor tal i com vam veure a l'activitat anterior.

També podem esborrar el contenidor i el volum associat

```
docker rm -v contenedor_id
```

Activitat 1

1.1 Llista tots els volums que tens creats a la teua màquina. ¿Quan tens?

1.2 Crea **3 volums** amb noms **web-logs, upload-data i cache-data**. Consulta els volums que tens disponibles. Esborra els volums creats per a la caché i per als log i torna a consultar els volums disponibles.

1.3.- Fes una captura de pantalla dels volums que tens disponibles a la teua màquina.

2.1.2 Associació del volum al contenidor

Un volum és una unitat independent en docker que podem associar a qualssevol contenidor que llancem per fer-ho, hem d'utilitzar la opció -v.

```
docker run -dp 8080:80 --name prova-nginx -v upload-data:/app/uploads
nginx:latest
```

Explicació

- -v estem especificant el volum que volem que utilitze (i que hem creat anteriorment) anomenat `upload-data`
- `/app/uploads` especifiquem la ruta de la màquina on volem que es munte el volum Una vegada en marxa el contenidor, **qualssevol informació** que **es guarde** al directori **/app/uploads, estarà guardant-se al volum.**

Seguint amb l'exemple anterior, configurarem un entorn Docker per a desplegar una aplicació PHP senzilla que permeta pujar imatges. Utilitzarem **Apache amb mod_php** per gestionar les peticions i Docker per gestionar els volums i els contenidors. Les passes a seguir són les següents:

1. Crear el Directori de Treball

- Comencem per crear una estructura de fitxers on gestionar ell projecte:

```
app-php/
    ├──html/
```

2. Crear l'Aplicació PHP

Al directori `html`, crea un fitxer `index.php` amb el següent contingut. Aquest formulari permet pujar imatges i les guarda en un directori designat.

html/index.php

```
<!DOCTYPE html>
<html lang="ca">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Pujar Imatges</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 2em;
        }
        .success {
            color: green;
        }
        .error {
            color: red;
        }
    </style>
</head>
<body>
    <h1>Pujar Imatges</h1>

    <?php
    $targetDir = "uploads/";
    if (!is_dir($targetDir)) {
        mkdir($targetDir, 0755, true);
    }

    if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_FILES['image'])) {
        $targetFile = $targetDir . basename($_FILES["image"]["name"]);
        $fileType = strtolower(pathinfo($targetFile, PATHINFO_EXTENSION));

        // Comprovar que és una imatge
        $check = getimagesize($_FILES["image"]["tmp_name"]);
        if ($check !== false) {
            if (move_uploaded_file($_FILES["image"]["tmp_name"], $targetFile)) {
                echo "<p class='success'>Imatge pujada correctament: <a href='$targetFile'>" .
htmlspecialchars($_FILES["image"]["name"]) . "</a></p>";
            } else {
                echo "<p class='error'>Error en pujar la imatge.</p>";
```

```
        }
    } else {
        echo "<p class='error'>El fitxer seleccionat no és una imatge.</p>";
    }
}
?>

<form action="index.php" method="post" enctype="multipart/form-data">
    <label for="image">Selecciona una imatge:</label>
    <input type="file" name="image" id="image" accept="image/*" required>
    <br><br>
    <button type="submit">Pujar</button>
</form>

<h2>Imatges pujades</h2>
<ul>
    <?php
    // Llistar les imatges al directori d'uploads
    $files = array_diff(scandir($targetDir), array('.', '..'));
    foreach ($files as $file) {
        echo "<li><a href='$targetDir$file' target='_blank'>$file</a></li>";
    }
    ?>
</ul>
</body>
</html>
```

3. Crear el Dockerfile

Crea un fitxer **Dockerfile** al directori arrel que llance l'aplicació a partir de la imatge personalitzada basada en **Apache** amb suport per a PHP.

Dockerfile

```
# Utilitzar una imatge base amb Apache i PHP preinstal·lats
FROM php:8.1-apache
# Copiar el contingut de l'aplicació a la carpeta d'Apache
COPY html/ /var/www/html/
# Establir permisos per al directori d'uploads
RUN mkdir -p /var/www/html/uploads && \
chown -R www-data:www-data /var/www/html/uploads && \
    chmod -R 755 /var/www/html/uploads
# Exposar el port 80
```

EXPOSE 80

4. Llançar el Contenedor

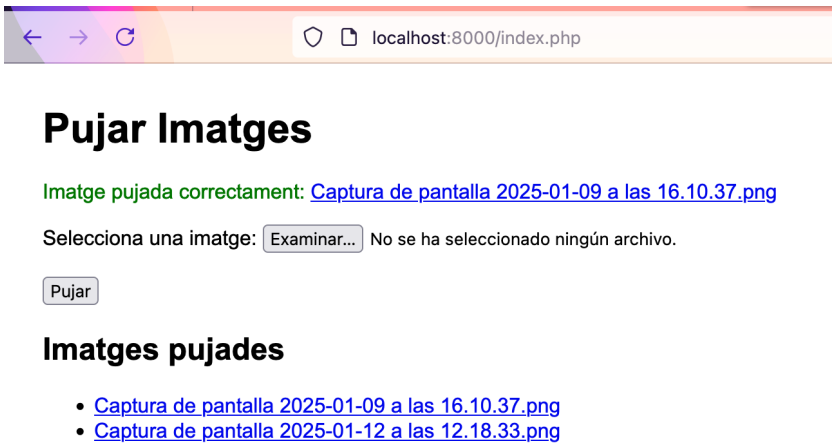
Construeix la imatge i llança el contenedor:

```
$ docker build -t php-apache-upload .

$ docker run -d -p 8000:80 -v upload-data:/var/www/html/uploads --name
app1 php-apache-upload
```

5. Accedir a l'Aplicació

- Obri el navegador i ves a http://localhost:8000.
- Utilitza el formulari per pujar una imatge.
- Les imatges pujades es guardaran al directori al volum upload-data



6. Verificar que les dades persisteixen entre l'execució i llançaments de diferents contenidors

Para i esborra el contenidor que acabes de crear

```
➜  app-php docker container stop app1
app1
➜  app-php docker container rm app1
app1
➜  app-php
```

Llença un nou contenidor anomenat app2 que utilize el port 8002 de la màquina local i comprova que les imatges pujades anteriorment encara existeixen:

```
$ docker run -d -p 8002:80 -v upload-data:/var/www/html/uploads --name
app2 php-apache-upload
```

Activitat 2.- Desplegament de l'aplicació

- Executa les passes anteriors i llença 3 contenidors app1, app2 i app3 que estiguen executant l'aplicació i utilitzen el mateix volum per a gestionar les imatges pujades.

Pega captures de pantalla on es vega:

- El volum creat
- Els contenidors que estan en marxa
- El accés a les aplicacions de cada contenidor on es vegen totes les imatges.
- Els logs que demostren que cada contenidor a pujat una imatge

2.1.3 Volums de tipus bind-mount

Els volums de tipus **bind mounts** permeten vincular un directori local del teu sistema operatiu amb un contenidor Docker. Això és útil quan vols accedir, compartir o modificar dades d'un directori de la maquina anfitriona amb el contenidor durant l'execució d'aquest. Per exemple, quan estem desenvolupant una aplicació.

Per exemplificar el seu ús, en compte de copiar el codi de l'aplicació en el moment en el que construïm la imatge, crearem un **bind-mount** entre el directori **/html** del host amfitrió i el document root del contenidor que trobem al directory **`/var/www/html`**, que qualssevol modificació dels fitxers de la web que fem en el host anfitrió es vegen reflectits sense haver de

reconstruir la imatge. Per fer-ho:

1. Llancem un contenidor

```
docker run -d -p 8003:80 -v upload-data:/var/www/html/uploads -v $(pwd)/html:/var/www/html --name app5 php-apache-upload
```

```
CONTAINER ID   IMAGE              COMMAND                  CREATED          STATUS          PORTS                  NAMES
e004636115b2   php-apache-upload  "docker-php-entrypoi…"   17 minutes ago   Up 17 minutes   0.0.0.0:8003->80/tcp   app5
```

2. Accedim al contenidor

3. Modifiquem el fitxer index.php i tornem a accedir a l'aplicació. Veurem com els canvis s'han aplicat sense tenir que reconstruir la imatge

4. Si llistem el contingut de la carpeta html del host amfitrió, podrem veure els fitxer pujats a la web

```
→  html ls -la
total 8
drwxr-xr-x  4 batoi  staff   128 12 ene 21:13 .
drwxr-xr-x  4 batoi  staff   128 12 ene 20:01 ..
```

```
-rw-r--r--  1 batoi  staff  2000 12 ene 21:13 index.php
drwxr-xr-x@ 2 batoi  staff    64 12 ene 20:54 uploads
```

Activitat 3.- Desplegament de l'aplicació

- Executa les passes anteriors fes una modificació en la web en la que fiques un títol amb el teu nom i accedeix a l'aplicació per comprovar que, sense tenir que contruir la imatge una altra vegada pots modificar l'aplicació. Pega una captura de pantalla.

4. Bibliografia / Webgrafía

- Documentació de la imatge docker Nginx. https://hub.docker.com/_/nginx
- Docker cli reference. https://docs.docker.com/engine/reference/run/