

Nabila Maamou, 2ºASIR

Docker. Práctica 4

Modulo 1.....	2
Configuración de contenedores con variables de entorno	2
Configuración de un contenedor con la imagen mariadb	2
Accediendo a servidor de base de datos desde el exterior.....	3
Contenedor	4
Ejecución simple de contenedores	4
Domonio	4
HolaMundo	5
El "Hola Mundo" de docke.....	5
interactivo.....	6
Web.....	7
Modificación del contenido servidor por el servidor web	7
Modulo 2.....	8
Creacion	8
Dockerhub.....	8
GestionGestión de imágenes.....	9
organizacion.....	11
Cómo se organizan las imágenes.....	11
Modulo 3.....	12
asociacion_bind_mount	12
Asociando almacenamiento a los contenedores: bind mount.....	12
asociacion_volumen	13
Asociando almacenamiento a los contenedores: volúmenes Docker.....	13
guestbook	14
Ejemplo 1: Despliegue de la aplicación Guestbook.....	14
Redes en Docker	14
Tipos de redes en Docker	14
redes_usuario	16
Redes definidas por el usuario	16
temperaturas.	17
Ejemplo 2: Despliegue de la aplicación Temperaturas.....	17
tomcat.....	18

Ejemplo 4: Despliegue de tomcat + nginx	18
Desplegando tomcat.....	18
wordpress	18
Ejemplo 3: Despliegue de Wordpress + mariadb	18

Modulo 1

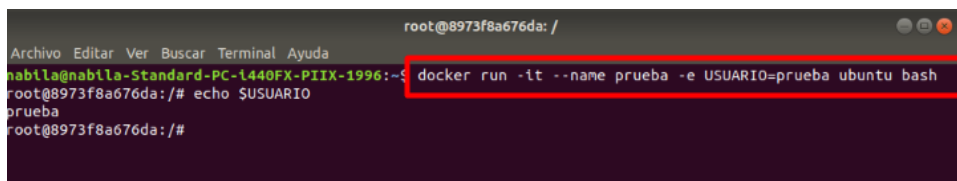
Configuración

Lleva a cabo al menos tres de los ejemplos mostrados en el módulo y documentalo en tu repositorio incluyendo capturas de pantalla.

Configuración de contenedores con variables de entorno

crear un contenedor que necesita alguna configuración específica, lo que vamos a hacer es crear variables de entorno en el contenedor, para que el proceso que inicializa el contenedor pueda realizar dicha configuración.

Para crear una variable de entorno al crear un contenedor usamos el flag `-e` o `--env`:



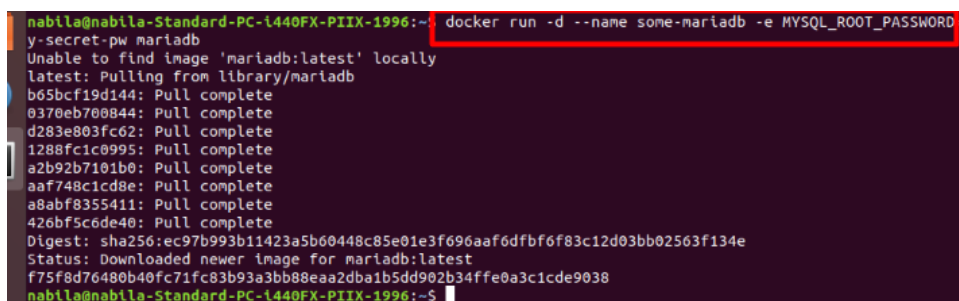
```

root@8973f8a676da: /
Archivo Editar Ver Buscar Terminal Ayuda
nabila@nabila-Standard-PC-l440FX-PIIX-1996:~$ docker run -it --name prueba -e USUARIO=prueba ubuntu bash
root@8973f8a676da: /# echo $USUARIO
prueba
root@8973f8a676da: /#

```

Configuración de un contenedor con la imagen mariadb

En ocasiones es obligatorio el inicializar alguna variable de entorno para que el contenedor pueda ser ejecutado. Si miramos la documentación en Docker Hub de la imagen mariadb, observamos que podemos definir algunas variables de entorno para la creación y configuración del contenedor (por ejemplo: `MYSQL_DATABASE`, `MYSQL_USER`, `MYSQL_PASSWORD`,...). Pero hay una que la tenemos que indicar de forma obligatoria, la contraseña del usuario root (`MYSQL_ROOT_PASSWORD`), por lo tanto:



```

nabila@nabila-Standard-PC-l440FX-PIIX-1996:~$ docker run -d --name some-mariadb -e MYSQL_ROOT_PASSWORD=...
y-secret-pw mariadb
Unable to find image 'mariadb:latest' locally
latest: Pulling from library/mariadb
b65bcf19d144: Pull complete
0370eb700844: Pull complete
d283e803fc62: Pull complete
1288fc1c0995: Pull complete
a2b92b7101b0: Pull complete
aaf748c1cd8e: Pull complete
a8abf8355411: Pull complete
426bf5c6de40: Pull complete
Digest: sha256:ec97b993b11423a5b60448c85e01e3f696aaf6dfbf6f83c12d03bb02563f134e
Status: Downloaded newer image for mariadb:latest
f75f8d76480b40fc71fc83b93a3bb88eaa2dba1b5dd902b34ffe0a3c1cde9038
nabila@nabila-Standard-PC-l440FX-PIIX-1996:~$

```

Podemos ver que se ha creado una variable de entorno:

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS        NAMES
f75f8d76480b   mariadb   "docker-entrypoint.s..." 39 seconds ago Up 38 seconds 3306/tcp     some-mariadb
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$
```

ejecutar:

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker exec -it some-mariadb env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=f75f8d76480b
TERM=xterm
MYSQL_ROOT_PASSWORD=my-secret-pw
GOSU_VERSION=1.14
LANG=C.UTF-8
MARIADB_VERSION=1:10.11.2+maria-ubu2204
HOME=/root
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$
```

Para acceder

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker exec -it some-mariadb bash
root@f75f8d76480b:/#
```

Accediendo a servidor de base de datos desde el exterior

vamos a mapear los puertos para acceder desde el exterior a la base de datos:

Lo primero que vamos a hacer es eliminar el contenedor anterior:

```
Archivo Editar Ver Buscar Terminal Ayuda
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker rm -f some-mariadb
some-mariadb
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$
```

vamos a crear otro contenedor, pero en esta ocasión vamos a mapear el puerto 3306 del anfitrión con el puerto 3306 del contenedor:

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker run -d -p 3306:3306 --name some-mariadb -e MYSQL_ROOT_PASSWORD=my-secret-pw mariadb
d13cfa52c8c326cb343928be8de23a77393c09a3377a7a2051450d7403ce5cd6
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$
```

Comprobamos

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS        NAMES
d13cfa52c8c3   mariadb   "docker-entrypoint.s..." 36 seconds ago Up 34 seconds 0.0.0.0:3306->3306/tcp   some-mariadb
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$
```

desde nuestro equipo (donde hemos instalado un cliente de mysql) nos conectamos que tiene la ip 127.0.0.1 vamos a conectarnos a la base de datos (hay que tener instalado el cliente de mariadb):

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ mysql -u root -p -h 127.0.0.1
Enter password:
```

Contenedor

Ejecución simple de contenedores

Con el comando run vamos a crear un contenedor donde vamos a ejecutar un comando:

```
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker run ubuntu echo 'Hello word'
Hello word
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$
```

Comprobamos con este comando:

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS
247a6c5efcbd   ubuntu    "echo 'Hello word'"     40 seconds ago Exited (0) 38 seconds ago
d13cfa52c8c3   mariadb   "docker-entrypoint.s..." 12 minutes ago Up 12 minutes   0.0.0.0:3306->3306/tcp
8973f8a676da   ubuntu    "bash"                  24 minutes ago Exited (0) 22 minutes ago
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$
```

Con el comando `docker images` podemos visualizar las imágenes que ya tenemos descargadas en nuestro registro local:

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
dockerfile    latest   8fec96b2307f   6 days ago    615MB
wordpress     latest   8fec96b2307f   6 days ago    615MB
mariadb        latest   6e11fcfc66ad   7 days ago    401MB
nginx         latest   904b8cb13b93   7 days ago    142MB
ubuntu        latest   74f2314a03de   8 days ago    77.8MB
hello-world    latest   feb5d9fea6a5   17 months ago 13.3kB
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$
```

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker run -it ubuntu bash
root@db00a47b9856:/#
```

Para ver los contenedores que no se están ejecutando:

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS
d13cfa52c8c3   mariadb   "docker-entrypoint.s..." 24 minutes ago Up 24 minutes   0.0.0.0:3306->3306/tcp
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$
```

Para eliminar el contenedor podemos identificarlo con su id:

Ejecutamos el comando :

`docker rm 372ca4634d53`

o con su nombre:

`docker rm nombre`

Domonio

Creando un contenedor demonio

En esta ocasión hemos utilizado la opción `-d` del comando `run`, para que la ejecución del comando en el contenedor se haga en segundo plano.

```
Archivo Editor Ver Buscar Terminal Ayuda
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker run -d --name contenedor2 ubuntu bash -c "while true
do echo hello world; sleep 1; done"
c739df94c0bd84ca78f46fc539e8d301ae733f08d750aa4cbf57d4b364348880
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$
```

interactivo

Ejecutando un contenedor interactivo

```
Archivo Editar Ver Buscar Terminal Ayuda
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker run -it --name contenedor1 ubuntu bash
root@6098786c8364:/#
```

El contenedor se para cuando salimos de él. Para volver a conectarnos a él:

```
contenedor1
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker start contenedor1
contenedor1
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$
```

Si el contenedor se está ejecutando podemos ejecutar comandos en él con el subcomando exec:

```
contenedor1
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker start contenedor1
contenedor1
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker exec contenedor1 ls -al
total 56
drwxr-xr-x 1 root root 4096 Mar  9 17:50 .
drwxr-xr-x 1 root root 4096 Mar  9 17:50 ..
-rwxr-xr-x 1 root root  0 Mar  9 17:50 .dockerenv
lrwxrwxrwx 1 root root  7 Mar  1 02:03 bin -> usr/bin
drwxr-xr-x 2 root root 4096 Apr 18  2022 boot
drwxr-xr-x 5 root root 360 Mar  9 17:51 dev
drwxr-xr-x 1 root root 4096 Mar  9 17:50 etc
drwxr-xr-x 2 root root 4096 Apr 18  2022 home
lrwxrwxrwx 1 root root  7 Mar  1 02:03 lib -> usr/lib
lrwxrwxrwx 1 root root  9 Mar  1 02:03 lib32 -> usr/lib32
lrwxrwxrwx 1 root root  9 Mar  1 02:03 lib64 -> usr/lib64
lrwxrwxrwx 1 root root 10 Mar  1 02:03 libx32 -> usr/libx32
drwxr-xr-x 2 root root 4096 Mar  1 02:03 media
drwxr-xr-x 2 root root 4096 Mar  1 02:03 mnt
drwxr-xr-x 2 root root 4096 Mar  1 02:03 opt
dr-xr-xr-x 309 root root  0 Mar  9 17:51 proc
drwx----- 1 root root 4096 Mar  9 17:51 root
drwxr-xr-x 5 root root 4096 Mar  1 02:06 run
lrwxrwxrwx 1 root root  8 Mar  1 02:03 sbin -> usr/sbin
drwxr-xr-x 2 root root 4096 Mar  1 02:03 srv
dr-xr-xr-x 13 root root  0 Mar  9 17:51 sys
drwxrwxrwt 2 root root 4096 Mar  1 02:06 tmp
drwxr-xr-x 14 root root 4096 Mar  1 02:03 usr
drwxr-xr-x 11 root root 4096 Mar  1 02:06 var
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$
```

Con la orden docker restart reiniciamos el contenedor, lo paramos y lo iniciamos.

Para mostrar información de un contenedor ejecutamos docker inspect:

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker inspect contenedor1
[
  {
    "Id": "6098786c83640a34f10beb536384607ffa6d5279e904d737b6168c943d537bca",
    "Created": "2023-03-09T17:50:50.266934461Z",
    "Path": "bash",
    "Args": [],
    "State": {
      "Status": "running",
      "Running": true,
      "Paused": false,
      "Restarting": false,
      "OOMKilled": false,
      "Dead": false,
      "Pid": 9829,
      "ExitCode": 0,
      "Error": "",
      "StartedAt": "2023-03-09T17:51:50.86388607Z",
      "FinishedAt": "2023-03-09T17:51:38.811513904Z"
    },
    "Image": "sha256:74f2314a03de34a0a2d552b05411fc9553a02ea71c1291b815b2f645f565683",
    "ResolvConfPath": "/var/lib/docker/containers/6098786c83640a34f10beb536384607ffa6d5279e904d737b6168c943d537bca/resolv.conf",
    "HostnamePath": "/var/lib/docker/containers/6098786c83640a34f10beb536384607ffa6d5279e904d737b6168c943d537bca/hostname",
    "HostsPath": "/var/lib/docker/containers/6098786c83640a34f10beb536384607ffa6d5279e904d737b6168c943d537bca/hosts",
    "LogPath": "/var/lib/docker/containers/6098786c83640a34f10beb536384607ffa6d5279e904d737b6168c943d537bca/6098786c83640a34f10beb536384607ffa6d5279e904d737b6168c943d537bca.json.log",
    "Name": "/contenedor1",
    "RestartCount": 0
  }
]
```

En realidad, todas las imágenes tienen definidos un proceso que se ejecuta, en concreto la imagen ubuntu tiene definida por defecto el proceso bash, por lo que podríamos haber ejecutado:

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker run -it --name contenedor3 ubuntu
root@73ff49e7b2ee:/#
```

Web

Creando un contenedor con un servidor web

Tenemos muchas imágenes en el registro público **docker hub**, por ejemplo podemos crear un servidor web con apache 2.4:

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker run -d --name my-apache-app -p 8080:80 httpd:2.4
Unable to find image 'httpd:2.4' locally
2.4: Pulling from library/httpd
3f9582a2cbe7: Already exists
9423d69c3be7: Pull complete
d1f584c02b5d: Pull complete
758a20a64707: Pull complete
08507f02f391: Pull complete
Digest: sha256:76618ddd53f315a1436a56dc84ad57032e1b2123f2f6489ce9c575c4b280c4f4
Status: Downloaded newer image for httpd:2.4
1ecd9150e58d2338743d8d5ea538a15148289d1b73fabab18ed5d2d4cddb8b89
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$
```

Para probarlo accede desde un navegador a <http://localhost:8080>



Para acceder al log del contenedor podemos ejecutar:

```
1ecd9150e58d2338743d8d5ea538a15148289d1b73fabab18ed5d2d4cddb8b89
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker logs my-apache-app
AH00558: httpd: Could not reliably determine the server's fully qualified domain
```

Modificación del contenido servidor por el servidor web

Accediendo de forma interactiva al contenedor y haciendo la modificación:

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker exec -it my-apache-app bash
root@1ecd9150e58d:/usr/local/apache2#
```

Ejecutando directamente el comando de creación del fichero index.html en el contenedor:

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker exec my-apache-app bash -c 'echo "<h1>Curso Docker</h1>" > /usr/local/apache2/htdocs/index.html'
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$
```

Accedemos al navegador:



Modulo 2

Creacion

Todas las imágenes tiene definidas un proceso que se ejecuta por defecto, pero en la mayoría de los casos podemos indicar un proceso al crear un contenedor.

Por ejemplo en la imagen ubuntu el proceso por defecto es bash, por lo tanto podemos ejecutar:

```
Archivo Editar Ver Buscar Terminal Ayuda
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker run -it --name contenedor2 ubuntu
root@f58410c88b60:/#
```

Pero podemos indicar el comando a ejecutar en la creación del contenedor:

```
exit
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker run ubuntu /bin/echo 'Hello world'
Hello world
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$
```

Otro ejemplo: la imagen httpd:2.4 ejecuta un servidor web por defecto, por lo tanto al crear el contenedor:

```
see docker run -it httpd
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker run -d --name my-apache2-app -p 8080:80 httpd:2.4
6c760b5d665d8a78898c85c85bff01fac98cd518aebd41a7fa2e00c073cafb69
```

Dockerhub

Las **imágenes** de Docker son plantillas de solo lectura, es decir, una imagen puede contener el sistema de archivos de un sistema operativo como Debian, pero esto solo nos permitirá crear los contenedores basados en esta configuración. Si hacemos cambios en el contenedor ya lanzado, al detenerlo esto no se verá reflejado en la imagen.

El **Registro docker** es un componente donde se almacena las imágenes generadas por el Docker Engine. Puede estar instalada en un servidor independiente y es un componente fundamental, ya que nos permite distribuir nuestras aplicaciones. Es un proyecto open source que puede ser instalado gratuitamente en cualquier servidor, pero, como hemos comentado, el proyecto nos ofrece Docker Hub.

El nombre de una imagen suele estar formado por tres partes:

usuario/nombre:etiqueta

- usuario: El nombre del usuario que la ha generado. Si la subimos a Docker Hub debe ser el mismo usuario que tenemos dado de alta en nuestra cuenta. Las **imágenes oficiales** en Docker Hub no tienen nombre de usuario.

- nombre: Nombre significativo de la imagen.
- etiqueta: Nos permite versionar las imágenes. De esta manera controlamos los cambios que se van produciendo en ella. Si no indicamos etiqueta, por defecto se usa la etiqueta latest, por lo que la mayoría de las imágenes tienen una versión con este nombre.

Gestion

Gestión de imágenes

Para crear un contenedor es necesario usar una imagen que tengamos descargado en nuestro registro local. Por lo tanto al ejecutar `docker run` se comprueba si tenemos la versión indicada de la imagen y si no es así, se precede a su descarga.

Las principales instrucciones para trabajar con imágenes son:

- `docker images`: Muestra las imágenes que tenemos en el registro local.
- `docker pull`: Nos permite descargar la última versión de la imagen indicada.
- `docker rmi`: Nos permite eliminar imágenes. No podemos eliminar una imagen si tenemos algún contenedor creada a partir de ella.
- `docker search`: Busca imágenes en Docker Hub.
- `docker inspect`: nos da información sobre la imagen indicada:
 - El id y el checksum de la imagen.
 - Los puertos abiertos.
 - La arquitectura y el sistema operativo de la imagen.
 - El tamaño de la imagen.
 - Los volúmenes.
 - El ENTRYPOINT que es lo que se ejecuta al hacer `docker run`.
 - Las capas.
 - Y muchas más cosas....

Mediawiki

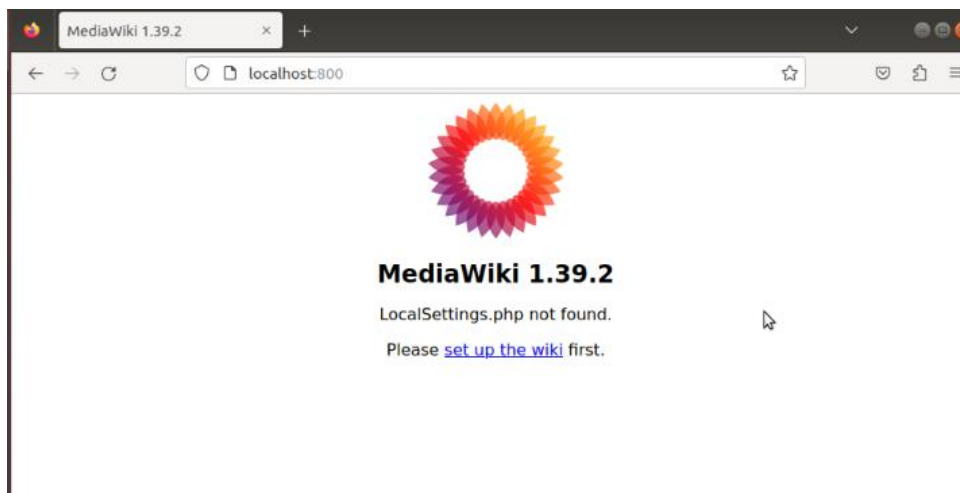
Ejemplo: Desplegando la aplicación mediawiki

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

En primer lugar vamos a instalar la última versión:

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker run -d -p 800:80 --name mediawiki2 mediawiki
76975b8493399715d9b51242fe0c2a120568d9d372897e8ad2001f3797e991ca
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$
```

Si accedemos a la ip de nuestro ordenador, al puerto 800, podemos observar que hemos instalado la versión 1.39.1:



Para instalar otra version

Elegemos otro puerto y otro nombre del contenedor

```
nabila@nabila-Standard-PC-l440FX-PIIX-1996:~$ docker run -d -p 8009:80 --name mediawiki4 mediawiki:1.38.5
Unable to find image 'mediawiki:1.38.5' locally
1.38.5: Pulling from library/mediawiki
3f9582a2cbe7: Already exists
0b95dc92ce55: Already exists
3630ff9f0131: Already exists
49efbc577363: Already exists
df983cae2963: Already exists
52fed2cf4dcf: Already exists
badf42672f1b: Already exists
baaffcad7804: Already exists
e65c63290641: Already exists
56fbc7142e6c: Already exists
0b0f64aca9a8: Already exists
1a0de30c1e15: Already exists
c28dd6954d0b: Already exists
00f756ea224a: Pull complete
b9ce5ce5d290: Pull complete
129b5e5405f8: Pull complete
55908e2b75: Pull complete
7adbd720ac37: Pull complete
3484f096ea40: Pull complete
e2d8e3d062e5: Pull complete
Digest: sha256:59052e38c77b5282771be30073f27243e54f0a9ee1e7cdc3f303b4ad10dc8e7a
Status: Image is up to date for mediawiki:1.38.5
9993175653ed0da23fdf47ec9b7e6a940299b445604e520ba9907ec187a7355a
nabila@nabila-Standard-PC-l440FX-PIIX-1996:~$
```

Cambiamos el puerto en navegador.



organizacion

Cómo se organizan las imágenes

cuando creamos un contenedor ocupa muy poco de disco duro, porque las capas de la imagen desde la que se ha creado se comparten con el contenedor:

Veamos el tamaño de nuestra imagen ubuntu:

```
Archivo Editar Ver Buscar Terminal Ayuda
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker images
REPOSITORY          TAG             IMAGE ID         CREATED          SIZE
httpd                2.4             daab1fa13f86    2 days ago      145MB
dockerfile           latest          8fec96b2307f    6 days ago      615MB
wordpress            latest          8fec96b2307f    6 days ago      615MB
mariadb              latest          6e11fcfc66ad    7 days ago      401MB
mediawiki            1.38.5         16e397360334    7 days ago      813MB
mediawiki            latest          fb3561c43a67    7 days ago      817MB
nginx                latest          904b8cb13b93    8 days ago      142MB
ubuntu               latest          74f2314a03de    8 days ago      77.8MB
hello-world          latest          febsd9fea6a5    17 months ago   13.3kB
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$
```

Si creamos un contenedor interactivo:

```
hello-world latest febsd9fea6a5 17 months ago 13.3kB
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker run -it --name contenedor5 ubuntu /bin/bash
root@d38a948fe4e2:/#
```

Nos salimos, y a continuación visualizamos los contenedores con la opción -s (size):

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker ps -a -s
CONTAINER ID   IMAGE      COMMAND                  NAMES                CREATED          STATUS
PORTS
d38a948fe4e2   ubuntu    "/bin/bash"             contenedor5          About a minute ago Exited (0) 20 seconds ago
o
9993175653ed   mediawiki:1.38.5 "docker-php-entrypoi..." mediawiki4           4 minutes ago     Up 4 minutes
0.0.0.0:8009->80/tcp, :::8009->80/tcp
c45a40797667   mediawiki "docker-php-entrypoi..." mediawiki3           6 minutes ago     Up 6 minutes
0.0.0.0:8008->80/tcp, :::8008->80/tcp
76975b849339   mediawiki "docker-php-entrypoi..." mediawiki2           9 minutes ago     Up 9 minutes
0.0.0.0:800->80/tcp, :::800->80/tcp
b026891bdda2   mediawiki "docker-php-entrypoi..." mediawiki1           16 minutes ago    Created
0B (virtual 817MB)
6c760b5d665d   httpd:2.4 "httpd-foreground"     my-apache2-app       21 minutes ago    Created
0B (virtual 145MB)
1ecd9150e58d   httpd:2.4 "httpd-foreground"     my-apache-app        3 hours ago       Up 3 hours
0.0.0.0:8080->80/tcp, :::8080->80/tcp
6098786c8364   ubuntu    "bash"                  contenedor1          4 hours ago       Up 3 hours
5B (virtual 77.8MB)
d13cfa52c8c3   mariadb "docker-entrypoint.s..." some-mariadb          4 hours ago       Up 4 hours
0.0.0.0:3306->3306/tcp, :::3306->3306/tcp
2B (virtual 401MB)
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$
```

Nos damos cuenta de que el tamaño real del contenedor es 0B y el virtual, el que comparte con la imagen son los 77,9MB que es el tamaño de la imagen ubuntu.

Si a continuación volvemos a acceder al contenedor y creamos un fichero:

```

nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker ps -a -s
CONTAINER ID   IMAGE          COMMAND                  NAMES      SIZE      CREATED          STATUS          PORTS
d38a948fe4e2   ubuntu        "/bin/bash"             contenedor5 5B (virtual 77.8MB) 2 minutes ago Up 9 seconds
9993175653ed   mediawiki:1.38.5 "docker-php-entrypoi..." mediawiki4 2B (virtual 813MB) 5 minutes ago Up 5 minutes    0.0.0.0:8009->80/tcp, :::8009->80/tcp
c45a40797667   mediawiki     "docker-php-entrypoi..." mediawiki3 2B (virtual 817MB) 7 minutes ago Up 7 minutes    0.0.0.0:8008->80/tcp, :::8008->80/tcp
76975b849339   mediawiki     "docker-php-entrypoi..." mediawiki2 2B (virtual 817MB) 10 minutes ago Up 10 minutes   0.0.0.0:800->80/tcp, :::800->80/tcp
b026891bdda2   mediawiki     "docker-php-entrypoi..." mediawiki1 0B (virtual 817MB) 17 minutes ago Created
6c760b5d665d   httpd:2.4     "httpd-foreground"     my-apache2-app 0B (virtual 145MB) 22 minutes ago Created
1ecd9150e58d   httpd:2.4     "httpd-foreground"     my-apache2-app 29B (virtual 145MB) 3 hours ago Up 3 hours      0.0.0.0:8080->80/tcp, :::8080->80/tcp
6098786c8364   ubuntu        "bash"                  contenedor1 5B (virtual 77.8MB) 4 hours ago Up 4 hours
d13cfa52c8c3   mariadb       "docker-entrypoint.s..." some-mariadb 2B (virtual 401MB) 4 hours ago Up 4 hours      0.0.0.0:3306->3306/tcp, :::3306->3306/tcp
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$

```

Por último, al solicitar información de la imagen, podemos ver información sobre las capas:

```

nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker inspect ubuntu:latest
[
  {
    "Id": "sha256:74f2314a03de34a0a2d552b805411fc9553a02ea71c1291b815b2f645f565683",
    "RepoTags": [
      "ubuntu:latest"
    ],
    "RepoDigests": [
      "ubuntu@sha256:2adf22367284330af9f832ffefb717c78239f6251d9d0f58de50b86229ed1427"
    ],
    "Parent": "",
    "Comment": "",
    "Created": "2023-03-01T04:38:49.239257335Z",
    "Container": "298f60554671ae2f5bf43b9892526aaa221e8093c9cee1ca68ef65fc3ac67600",
    "ContainerConfig": {
      "Hostname": "298f60554671",
      "Domainname": "",
      "User": "",
      "AttachStdin": false,
      "AttachStdout": false,
      "AttachStderr": false,
      "Tty": false,
      "OpenStdin": false,
      "StdinOnce": false,
      "Env": [
        "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
      ],
      "Cmd": null
    },
    "Image": "sha256:74f2314a03de34a0a2d552b805411fc9553a02ea71c1291b815b2f645f565683"
  }
]

```

Modulo 3

asociacion_bind_mount

Asociando almacenamiento a los contenedores: bind mount

En este caso vamos a crear un directorio en el sistema de archivo del host, donde vamos a crear un fichero index.html:

```

nabila@nabila-Standard-PC-i440FX-PIIX-1996:~/web$ docker run -d --name my-apache2-app -v /home/usuario/web:/usr/local/apache/htdocs -p 8082:80 httpd:2.4
48800aa5ddf252e3b435aeba3ddb007a22acbc9788f9dff89884952753b70aca
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~/web$

```

Y comprobamos que realmente estamos sirviendo el fichero que tenemos en el directorio que hemos creado.

```

48800aa5ddf252e3b435aeba3ddb007a22acbc9788f9dff89884952753b70aca
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~/web$ curl http://localhost:8082
<html><body><h1>It works!</h1></body></html>
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~/web$

```

Eliminamos el contenedor y volvemos a crear otro con el directorio montado:

```

nabila@nabila-Standard-PC-i440FX-PIIX-1996:~/web$ docker stop my-apache2-app
my-apache2-app
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~/web$ docker rm my-apache2-app
my-apache2-app
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~/web$

```

asociacion_volumen

Asociando almacenamiento a los contenedores: volúmenes Docker

Veamos como puedo usar los volúmenes y los bind mounts en los contenedores. Aunque dos formas de asociar el almacenamiento al contenedor nosotros vamos a usar el flag `--volume` o `-v`.

Si usamos imágenes de DockerHub, debemos leer la información que cada imagen nos proporciona en su página ya que esa información suele indicar cómo persistir los datos de esa imagen, ya sea con volúmenes o bind mounts, y cuáles son las carpetas importantes en caso de ser imágenes que contengan ciertos servicios (web, base de datos etc...)

Ejemplo usando volúmenes docker

Lo primero que vamos a hacer es crear un volumen docker:

```
see docker --help
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~/web$ docker volume create miweb
miweb
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~/web$
```

A continuación creamos un contenedor con el volumen asociado, usando `--mount`, y creamos un fichero `index.html`:

```
miweb
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~/web$ docker run -d --name my-apache-app -v miweb:/usr/local
/apache2/htdocs -p 8080:80 httpd:2.4
e0992de4102034ad06080805c0d38c5929c6e2e34629603b5225473351c8b9e52
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~/web$
```

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker exec my-apache-app bash -c "echo <n1>Hola</h1>" > /
usr/local/apache2/htdocs/index.html'
> ^C
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ curl http://localhost:8080
<html><body><h1>It works!</h1></body></html>
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$
```

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker stop my-apache-app
my-apache-app
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker rm my-apache-app
my-apache-app
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$
```

Después de borrar el contenedor, volvemos a crear otro contenedor con el mismo volumen asociado:

```
my-apache-app
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker run -d --name my-apache-app -v miweb:/usr/local/ap
ache2/htdocs -p 8083:80 httpd:2.4
6a07fb2705e9c3fe4c43e46d78ce6d74c2bd636e34317c0c8892f3ffc6e40c10
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$
```

Y podemos comprobar que no se ha perdido la información (el fichero `index.html`):

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ curl http://localhost:8083
<html><body><h1>It works!</h1></body></html>
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$
```

guestbook

Ejemplo 1: Despliegue de la aplicación Guestbook

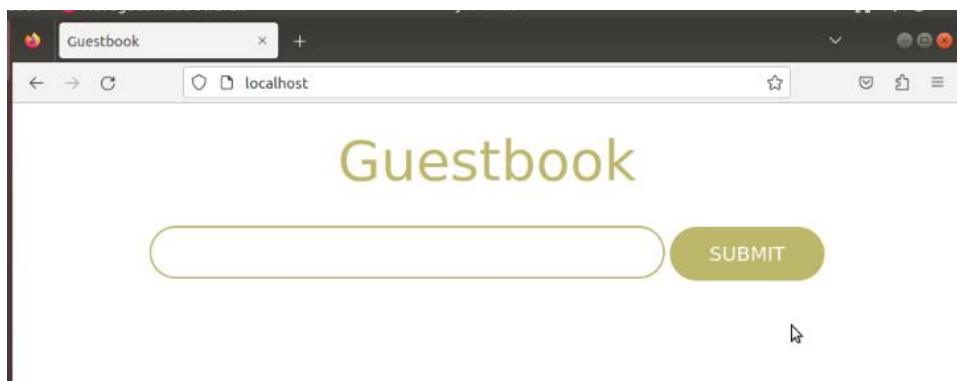
Los dos contenedores tienen que estar en la misma red y deben tener acceso por nombres (resolución DNS) ya que de principio no sabemos que ip va a coger cada contenedor. Por lo tanto vamos a crear los contenedores en la misma red:

```
nabila@nabila-Standard-PC-l440FX-PIIX-1996:~$ docker network create red_guestbook
a79f8b94de38b76fd0ee4ddd96f6c4e6ae7956a057896f7bf9c600e487bf3e3e
nabila@nabila-Standard-PC-l440FX-PIIX-1996:~$
```

Para ejecutar los contenedores:

```
See 'docker run --help'.
nabila@nabila-Standard-PC-l440FX-PIIX-1996:~$ docker run -d --name redis --network red_guestbook -v /opt
/redis:/data redis redis-server --appendonly yes
Unable to find image 'redis:latest' locally
latest: Pulling from library/redis
3f9582a2cbe7: Already exists
241c2d338588: Pull complete
89515d93a23e: Pull complete
65e8ba9473fe: Pull complete
585124038cab: Pull complete
b483de716a47: Pull complete
Digest: sha256:e50c7e23f79ae81351beacb20e004720d4bed657415e68c2b1a2b5557c075ce0
Status: Downloaded newer image for redis:latest
7e5dbd3f0fb7df71ea0ea21e196a5d804140776eb5273ecd44037083b9c17bad
nabila@nabila-Standard-PC-l440FX-PIIX-1996:~$
```

```
nabila@nabila-Standard-PC-l440FX-PIIX-1996:~$ docker run -d -p 80:5000 --name guestbook --network red_gu
estbook iesgn/guestbook
Unable to find image 'iesgn/guestbook:latest' locally
latest: Pulling from iesgn/guestbook
0ecb575e629c: Extracting 42.47MB/50.4MB
7467d1831b69: Download complete
feab2c490a3c: Download complete
f15a0f46f8c3: Download complete
937782447ff6: Download complete
e78b7aaaab2c: Download complete
dfce8611166c: Download complete
3a6aeb6d9625: Download complete
2b7e1323c92f: Download complete
4bf029403d35: Download complete
5e777db1f033: Download complete
```



Redes en Docker

Tipos de redes en Docker

Cuando instalamos docker tenemos las siguientes redes predefinidas:


```
Archivo Editar Ver Buscar Terminal Ayuda
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker network ls
NETWORK ID        NAME                DRIVER              SCOPE
707cb20f0c11      bridge              bridge              local
0298c1782b15      host                host                local
ead1350ed86a      none                null                local
a79f8b94de38      red_guestbook       bridge              local
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$
```

Vamos a crear un contenedor interactivos con la imagen debian:

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker run -it --name contenedor1 --rm debian bash
Unable to find image 'debian:latest' locally
latest: Pulling from library/debian
32fb02163b6b: Pull complete
Digest: sha256:f81bf5a8b57d6aa1824e4edb9aea6bd5ef6240bcc7d86f303f197a2eb77c430f
Status: Downloaded newer image for debian:latest
docker: Error response from daemon: Conflict. The container name "/contenedor1" is already in use by con
tainer "6098786c83640a34f10beb536384607ffa6d5279e904d737b6168c943d537bca". You have to remove (or rename
) that container to be able to reuse that name.
```

En otra pestaña, podemos ejecutar esta instrucción para obtener la ip que se le ha asignado:

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker inspect -f '{{range.NetworkSettings.Networks}}{{.IP
Address}}{{end}}' contenedor1
172.17.0.3
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$
```

Observamos que el contenedor tiene una ip en la red 172.17.0.3/16. Además podemos comprobar que se ha creado un bridge en el host, al que se conectan los contenedores:

```
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ sudo apt install iproute2
[sudo] contraseña para nabila:
Lo sentimos, vuelva a intentarlo.
[sudo] contraseña para nabila:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
 fonts-liberation2 fonts-opensymbol gir1.2-gst-plugins-base-1.0 gir1.2-gstreamer-1.0 gir1.2-gudev-1.0
 gir1.2-udisks-2.0 grilo-plugins-0.3-base gstreamer1.0-gtk3 libboost-date-time1.65.1
 libboost-filesystem1.65.1 libboost-iostreams1.65.1 libboost-locale1.65.1 libbdr-0.1-1
 libclucene-contribs1v5 libclucene-core1v5 libcmis-0.5-5v5 libcolamd2 libdazzle-1.0-0 libe-book-0.1-1
 libdataserverui-1.2-2 libeat0 libepubgen-0.1-1 libetonyek-0.1-1 libevent-2.1-6 libexiv2-14
 libfreerdp-client2-2 libfreerdp2-2 libgic2 libgee-0.8-2 libgexiv2-2 libgom-1.0-0 libgpgmepp6
 libgpgod-common libgpgod4 liblangtag-common liblangtag1 liblirc-client0 liblua5.3-0 libmediaart-2.0-0
 libmispub-0.1-1 libodfgen-0.1-1 libqwing2v5 libraw16 libvenge-0.0-0 libsgutils2-2 libssh-4
 libsuitesparseconfig5 libvncclient1 libwinpr2-2 libxapian30 libxmlsec1 libxmlsec1-nss
 linux-hwe-5.4-headers-5.4.0-42 lp-solve media-player-info python3-mako python3-markupsafe syslinux
 syslinux-common syslinux-legacy usb-creator-common
Utilice «sudo apt autoremove» para eliminarlos.
Paquetes sugeridos:
 iproute2-doc
Se actualizarán los siguientes paquetes:
 iproute2
1 actualizados, 0 nuevos se instalarán, 0 para eliminar y 81 no actualizados.
Se necesita descargar 0 B/721 kB de archivos.
Se utilizarán 0 B de espacio de disco adicional después de esta operación.
(Leyendo la base de datos ... 172769 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../iproute2_4.15.0-2ubuntu1.3_amd64.deb ...
Desempaquetando iproute2 (4.15.0-2ubuntu1.3) sobre (4.15.0-2ubuntu1.2) ...
Configurando iproute2 (4.15.0-2ubuntu1.3) ...
Procesando disparadores para man-db (2.8.3-2ubuntu0.1) ...
Progreso: [ 83% ] [#####.....[....]
```

```

Procesando utmp para mail-00 (2:8:3-2000000.1) ...
nabila@nabila-Standard-PC-l440FX-PIIX-1996:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: ens18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 12:4d:8d:f5:1a:01 brd ff:ff:ff:ff:ff:ff
    inet 10.4.3.206/16 brd 10.4.255.255 scope global dynamic noprefixroute ens18
        valid_lft 6681sec preferred_lft 6681sec
    inet6 fe80::461a:e88e:97e0:1046/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
3: ens19: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether de:6d:2c:0c:f4:43 brd ff:ff:ff:ff:ff:ff
4: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:80:e1:a8:85 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
        valid_lft forever preferred_lft forever
    inet6 fe80::42:80ff:fe01:a885/64 scope link
        valid_lft forever preferred_lft forever
20: veth8934a86@l1f19: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker0 state UP group default
    link/ether 92:9d:48:09:57:e4 brd ff:ff:ff:ff:ff:ff link-netnsid 0

```

Si conecto un contenedor a la red **host**, el contenedor ofrece el servicio que tiene configurado en el puerto de la red del anfitrión. No tiene ip propia, sino es cómo si tuviera la ip del anfitrión. Por lo tanto, los puertos son accesibles directamente desde el host. Por ejemplo:

```

nabila@nabila-Standard-PC-l440FX-PIIX-1996:~$ docker run -d --name mi_servidor --network host josedom24/aplicacionweb:v1
Unable to find image 'josedom24/aplicacionweb:v1' locally
v1: Pulling from josedom24/aplicacionweb
c5e155d5a1d1: Extracting 43.58MB/45.34MB
f2b76f8f3462: Download complete
2e8f82335e4d: Download complete

```

```

nabila@nabila-Standard-PC-l440FX-PIIX-1996:~$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
a7c3bb2e7437   iesgn/guestbook   "python3 app.py"        11 minutes ago   Up 11 minutes   0.0.0.
0:80->5000/tcp, :::80->5000/tcp
7e5dbd3f8fb7   redis            "docker-entrypoint.s..." 13 minutes ago   Up 13 minutes   6379/t
cp
6a07fb2705e9   httpd:2.4        "httpd-foreground"      18 minutes ago   Up 17 minutes   0.0.0.
0:8083->80/tcp, :::8083->80/tcp
9993175653ed   mediawiki:1.38.5  "docker-php-entrypoi..." About an hour ago   Up About an hour   0.0.0.
0:8009->80/tcp, :::8009->80/tcp
c45a40797667   mediawiki        "docker-php-entrypoi..." About an hour ago   Up About an hour   0.0.0.
0:8008->80/tcp, :::8008->80/tcp
76975b849339   mediawiki        "docker-php-entrypoi..." About an hour ago   Up About an hour   0.0.0.
0:800->80/tcp, :::800->80/tcp
6098786c8364   ubuntu           "bash"                   5 hours ago      Up 5 hours      0.0.0.
d13cfa52c8c3   mariadb         "docker-entrypoint.s..." 5 hours ago      Up 5 hours      0.0.0.
0:3306->3306/tcp, :::3306->3306/tcp
nabila@nabila-Standard-PC-l440FX-PIIX-1996:~$

```

redes_usuario

Redes definidas por el usuario

```

nabila@nabila-Standard-PC-l440FX-PIIX-1996:~$ docker network create red1
e10656b08fa402bae1725c8d761b0aaea0cbb93609c0277685a8ea944fb841fd
nabila@nabila-Standard-PC-l440FX-PIIX-1996:~$

```

Como no hemos indicado ninguna configuración en la red que hemos creado, docker asigna un direccionamiento a la red:


```

Error: NO SUCH NETWORK: red1
nabila@nabila-Standard-PC-l440FX-PIIX-1996:~$ docker network inspect red1
[
  {
    "Name": "red1",
    "Id": "e10656b08fa402bae1725c8d761b0aaea0cbb93609c0277685a8ea944fb841fd",
    "Created": "2023-03-09T23:40:23.573268923+01:00",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": {},
      "Config": [
        {
          "Subnet": "172.19.0.0/16",
          "Gateway": "172.19.0.1"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {},
    "Options": {},
    "Labels": {}
  }
]
nabila@nabila-Standard-PC-l440FX-PIIX-1996:~$

```

Vamos a crear dos contenedores conectados a dicha red:

```

my-apache-app
nabila@nabila-Standard-PC-l440FX-PIIX-1996:~$ docker run -d --name my-apache-app --network red1 -p 8080:
80 httpd:2.4
5eb76e2c64dbafc7c9cc372528466d07bb85209b6b8f740d7279dc33f3b7cb18
nabila@nabila-Standard-PC-l440FX-PIIX-1996:~$

```

Lo primero que vamos a comprobar es la resolución DNS:

```

root@68d262b30c07:/# apt-get install dnsutils -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package dnsutils
root@68d262b30c07:/#

```

temperaturas.

Ejemplo 2: Despliegue de la aplicación Temperaturas

Vamos a crear una red para conectar los dos contenedores:

```

Archivo Editar Ver Buscar Terminal Ayuda
nabila@nabila-Standard-PC-l440FX-PIIX-1996:~$ docker network create red_temperaturas
8b00d63a41614902f01edbbaa20ffc28ec119876f31ef261a55162a2af2758b1
nabila@nabila-Standard-PC-l440FX-PIIX-1996:~$

```

Para ejecutar los contenedores:

```

8b00d63a41614902f01edbbaa20ffc28ec119876f31ef261a55162a2af2758b1
nabila@nabila-Standard-PC-l440FX-PIIX-1996:~$ docker run -d --name temperaturas-backend --network red_te
mperaturas iesgn/temperaturas_backend
Unable to find image 'iesgn/temperaturas_backend:latest' locally
latest: Pulling from iesgn/temperaturas_backend
32de3c850997: Downloading 30.63MB/55.03MB
4a604c2354e7: Downloading 20.93MB/164MB
f70ec8148670: Download complete
7238ddc8ca51: Download complete

```

```
1e76f73ff3061c84d181b8ee355a582b6f11edb60c0 5a64a1/a1e3bee1b7c1
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker run -d -p 80:3000 --name temperaturas-frontend --ne
twork red_temperaturas iesgn/temperaturas_frontend
Unable to find image 'iesgn/temperaturas_frontend:latest' locally
latest: Pulling from iesgn/temperaturas_frontend
0ecb575e629c: Already exists
1b0ea791d129: Downloading 78.35MB/192.9MB
18ffb3e704a1: Download complete
d724eea63276: Download complete
```



tomcat

Ejemplo 4: Despliegue de tomcat + nginx

Desplegando tomcat

Antes de hacer el despliegue del primer contenedor, vamos a crear una red bridge para conectar los contenedores:

```
Archivo Editar Ver Buscar Terminal Ayuda
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$ docker network create red_tomcat
caf9c9725d44d50faa2bda0753a1e3eef1a6c47f945812a69454fba45a18cd24
nabila@nabila-Standard-PC-i440FX-PIIX-1996:~$
```

A continuación vamos a crear un contenedor a partir de la imagen [tomcat](#). En la documentación podemos ver que el directorio `/usr/local/tomcat/webapps/` es donde tenemos que poner el fichero de despliegue war (vamos a usar **bind mount** para montar el fichero war en el directorio). No vamos a mapear puerto porque no vamos a acceder a este contenedor desde el exterior.

Tenemos un directorio donde tenemos el fichero war (puedes encontrar estos ficheros en el [repositorio github](#)):

wordpress

Ejemplo 3: Despliegue de Wordpress + mariadb

Vamos a hacer un contenedor de WordPress

Para ello primero creamos un directorio: **sudo mkdir WordPress**

Dentro de este directorio creamos un fichero se llama Dockerfile: **sudo gedit Dockerfile**

Dentro de este fichero ponemos este contenido:

```
Abrir  Dockerfile [Solo lectura]  Guardar
/wordpress

Version: '3.1'

services:
  wordpress:
    image: wordpress
    restart: always
    ports:
      - 8080:80
    environment:
      WORDPRESS_DB_HOST: db
      WORDPRESS_DB_USER: exampleuser
      WORDPRESS_DB_PASSWORD: examplepass
      WORDPRESS_DB_NAME: exampledb
    volumes:
      - wordpress:/var/www/html

  db:
    image: mysql:5.7
    restart: always
    environment:
      MYSQL_DATABASE: exampledb
      MYSQL_USER: exampleuser
      MYSQL_PASSWORD: examplepass
      MYSQL_RANDOM_ROOT_PASSWORD: '1'
    volumes:
      - db:/var/lib/mysql

volumes:
  wordpress:
  db:
```

Guardamos y cerrarlo

Para ejecutarlo usamos este comando.

```
nabila@nabila-Standard-PC-l440FX-PIIX-1996:/wordpress$ sudo chmod 777 /var/run/*nabila@nabila-Standard-P
C-l440FX-PIIX-1996:/wordpress$ docker run --name some-wordpress -p 8080:80 -d wordpress
Unable to find image 'wordpress:latest' locally
latest: Pulling from library/wordpress
3f9582a2cbe7: Pull complete
0b95dc92ce55: Pull complete
3630ff9f8131: Pull complete
49efbc577363: Pull complete
df983cae2963: Pull complete
52fed2cf4dcf: Pull complete
badf42672f1b: Pull complete
baaffcad7804: Pull complete
e65c63290641: Pull complete
56fbc7142e6c: Pull complete
0b0f64aca9a8: Pull complete
1a0de30c1e15: Pull complete
c28dd6954d0b: Pull complete
12b130c64439: Pull complete
b7956b05af90: Pull complete
bce2c538d546: Pull complete
275d52fd4f6c: Pull complete
841c64b802b3: Pull complete
af298c263b20: Pull complete
6d98f1e7e80e: Pull complete
e77d1e0ab8c5: Pull complete
Digest: sha256:52496c5b4dbfe89fc4a646bad5b247d10df32fe1c9a49e7cde90cf3d320230a1
Status: Downloaded newer image for wordpress:latest
179d8f242383bdbbb4bd724e0649095467f9b6ea32f9553aad248923cd5acaa
nabila@nabila-Standard-PC-l440FX-PIIX-1996:/wordpress$ sudo docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
wordpress     latest    8fec96b2307f   3 days ago    615MB
hello-world    latest    feb5d9fea6a5   17 months ago 13.3kB
nabila@nabila-Standard-PC-l440FX-PIIX-1996:/wordpress$
```

En el navegador ponemos <http://localhost:8080>

Y como veremos ya funciona correctamente

