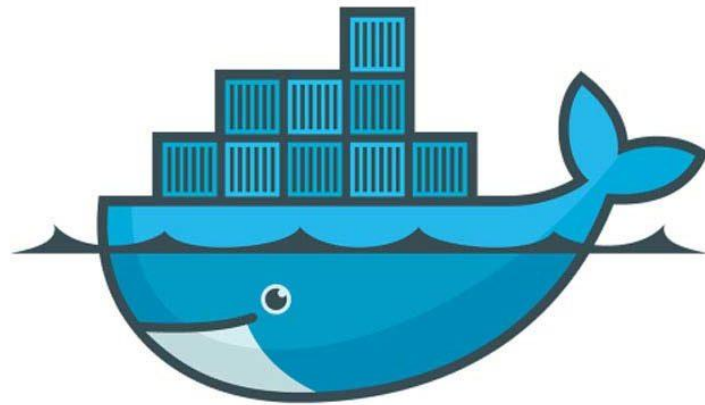




DOCKER

Sistemas Informáticos (1º DAM)



Antonio Salces Alcaraz (1º DAM)
C.P.I.F.P. Alan Turing
17/01/2025

Índice

1.	ACTIVIDAD 1: DESCARGAR IMÁGENES Y OPERACIONES BÁSICAS	1
1.1.	DESCARGAR IMÁGENES DE DOCKER HUB.	1
1.2.	MOSTRAR IMÁGENES	3
1.3.	CONTENEDOR CON UBUNTU 18.04, NOMBRE "UBUNTU"	4
1.3.1.	PARAR CONTENEDOR	4
1.3.2.	REARRANCAR CONTENEDOR	4
1.3.3.	MOSTRAR FICHERO SIN ENTRAR AL CONTENEDOR	5
2.	ACTIVIDAD 2: TRABAJAR CON PHP Y MARIADB	6
2.1.	CONTENEDOR CON PHP, NOMBRE "WEB" Y PUERTO 8181	6
2.2.	COLOCAR FICHEROS INDEX.HTML E INDEX.PHP EN EL DIRECTORIO WEB DEL CONTENEDOR	6
2.3.	CONTENEDOR CON MARIADB CON DIFERENTES PARÁMETROS	7
2.4.	CONECTAR CON UN CLIENTE DE BASE DE DATOS	7
3.	ACTIVIDAD 3: GESTIÓN DE IMÁGENES	9
3.1.	DESCARGAR IMAGEN UBUNTU:20.04	9
3.2.	VOLCAR LA INFORMACIÓN DE LA NUEVA IMAGEN EN UN ARCHIVO	9
3.3.	CREAR CONTENEDOR CON UBUNTU:20.04 LLAMADA "MODULO3"	10
3.4.	BORRAR CONTENEDOR CON UBUNTU:20.04	10
3.5.	HACER ACCIONES NECESARIAS PARA BORRAR EL CONTENEDOR UBUNTU:20.04	10
4.	ACTIVIDAD 4: OPERACIONES CON VOLÚMENES	11
4.1.	CREAR VOLÚMENES	11
4.2.	INSTANCIAR CONTENEDORES EN LOS VOLÚMENES	11
4.3.	PARAR Y BORRAR CONTENEDOR "C2" Y VOLUMEN "VOLUMEN_DATOS"	12
4.4.	COMPROBAR QUE "C1" ESTÁ MONTADO SOBRE "VOLUMEN_WEB"	12
5.	ACTIVIDAD 5: TRABAJO CON REDES	13
5.1.	CREACIÓN DE REDES	13
5.2.	CONFIGURACIÓN DE AMBAS REDES	13
5.3.	ARRANCAR CONTENEDOR UBUNTU:20.04 CON CONFIGURACIÓN ESPECÍFICA	14
5.4.	ARRANCAR OTRO CONTENEDOR UBUNTU:20.04 CON OTRA CONFIGURACIÓN	15
5.5.	CONFIGURACIÓN DE RED DE AMBOS CONTENEDORES	16
5.6.	COMPROBAR CONECTIVIDAD ENTRE CONTENEDORES	17
5.7.	CONECTAR EL CONTENEDOR "U1" A "RED2"	18
6.	ACTIVIDAD 5: EDITAR FICHERO DE CONFIGURACIÓN .YML	19
6.1.	DESCARGAR ARCHIVO DE CONFIGURACIÓN .YML PARA DOCKER-COMPOSE	19
6.2.	EDITAR EL ARCHIVO .YML Y CAMBIAR PUERTO	19
6.3.	ARRANCAR CON "DOCKER-COMPOSE"	19
6.4.	ENTRAR EN EL CONTENEDOR MEDIANTE EL NAVEGADOR	20
7.	OPERACIONES CON IMÁGENES NO FIRMADAS	21
7.1.	DESCARGAR IMAGEN NO FIRMADA	21
7.2.	CAMBIAR LA VARIABLE DE ENTORNO "DOCKER_CONTENT_TRUST"	22
7.3.	INTENTAR DESCARGAR IMAGEN NO FIRMADA	23
8.	BIBLIOGRAFIA	24

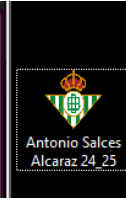
1. Actividad 1: descargar imágenes y operaciones básicas

1.1. Descargar imágenes de Docker Hub.

Para descargar los dockers, debemos de utilizar el comando ***“docker pull <imagen[:versión]>”***. Por defecto, cogerá la versión “latest” si no ponemos versión.

Ubuntu:18.04

```
usuario@usuario:~$ sudo docker pull ubuntu:18.04
[sudo] contraseña para usuario:
18.04: Pulling from library/ubuntu
7c457f213c76: Pull complete
Digest: sha256:152dc042452c496007f07ca9127571cb9c29697f42acbfad72324b2bb2e43c98
Status: Downloaded newer image for ubuntu:18.04
docker.io/library/ubuntu:18.04
usuario@usuario:~$ S
```



Centos:8

```
usuario@usuario:~$ sudo docker pull centos:8
8: Pulling from library/centos
a1d0c7532777: Pull complete
Digest: sha256:a27fd8080b517143cbbab9dfb7c8571c40d67d534bbdee55bd6c473f432b177
Status: Downloaded newer image for centos:8
docker.io/library/centos:8
usuario@usuario:~$ S
```



Debian: 9

```
usuario@usuario:~$ sudo docker pull debian:9
9: Pulling from library/debian
8372a04f222b: Pull complete
Digest: sha256:c5c5200ff1e9c73ffbf188b4a67eb1c91531b644856b4aefe86a58d2f0cb05be
Status: Downloaded newer image for debian:9
docker.io/library/debian:9
```



MariaDB:latest

```
usuario@usuario:~$ sudo docker pull mariadb
Using default tag: latest
latest: Pulling from library/mariadb
de44b265507a: Pull complete
ca9b21d0c985: Pull complete
041c753879ae: Pull complete
e7b5137dc4b2: Pull complete
655e5d2590bd: Pull complete
41b3170b5f12: Pull complete
95adc28016bc: Pull complete
407e9d6eefb4: Pull complete
Digest: sha256:a9547599cd87d7242435aea6fda22a9d83e2c06d16c658ef70d2868b3d3f6a80
Status: Downloaded newer image for mariadb:latest
docker.io/library/mariadb:latest
usuario@usuario:~$
```



MySQL:5.7

```
usuario@usuario:~$ sudo docker pull mysql:5.7
5.7: Pulling from library/mysql
20e4dcae4c69: Pull complete
1c56c3d4ce74: Pull complete
e9f03a1c24ce: Pull complete
68c3898c2015: Pull complete
6b95a940e7b6: Pull complete
90986bb8de6e: Pull complete
ae71319cb779: Pull complete
ffc89e9dfd88: Pull complete
43d05e938198: Pull complete
064b2d298fba: Pull complete
df9a4d85569b: Pull complete
Digest: sha256:4bc6bc963e6d8443453676cae56536f4b8156d78bae03c0145cbe47c2aad73bb
Status: Downloaded newer image for mysql:5.7
docker.io/library/mysql:5.7
usuario@usuario:~$
```

httpd

```
usuario@usuario:~$ sudo docker pull httpd
Using default tag: latest
latest: Pulling from library/httpd
bc0965b23a04: Pull complete
d7ad38c6dd97: Pull complete
4f4fb700ef54: Pull complete
79b49624e34b: Pull complete
7d9f97915db2: Pull complete
9bd25d4f7b77: Pull complete
Digest: sha256:f4c5139eda466e45814122d9bd8b886d8ef6877296126c09b76dbad72b03c336
Status: Downloaded newer image for httpd:latest
docker.io/library/httpd:latest
```

tomcat:9.0.39-jdk11

```
usuario@usuario:~$ sudo docker pull tomcat:9.0.39-jdk11
9.0.39-jdk11: Pulling from library/tomcat
e4c3d3e4f7b0: Pull complete
101c41d0463b: Pull complete
8275efcd805f: Pull complete
751620502a7a: Pull complete
a59da3a7d0e7: Pull complete
9c0f1dffe039: Pull complete
576e3c6f47f8: Pull complete
c7e1b6c3ef84: Pull complete
d1b8a428acdc: Pull complete
7251ae448a6d: Pull complete
Digest: sha256:5b17d5de9c75c9da638c28186c19423b610e7eab3b6f6b975bf47383d12ed0a9
Status: Downloaded newer image for tomcat:9.0.39-jdk11
docker.io/library/tomcat:9.0.39-jdk11
usuario@usuario:~$
```



jenkins/jenkins:lts

```

usuario@usuario:~$ sudo docker pull jenkins/jenkins:lts
[sudo] contraseña para usuario:
lts: Pulling from jenkins/jenkins
b2b31b28ee3c: Pull complete
768595d27f0b: Pull complete
2902ddfaf8af: Pull complete
1944ded7dbca: Pull complete
37b0412849e4: Pull complete
9e6f96481dc6: Pull complete
8d5cd706e369: Pull complete
e1d3077f0c0c: Pull complete
66714a60a07a: Pull complete
e37c8a6a1d29: Pull complete
0867b45f78b4: Pull complete
d0238388e632: Pull complete
Digest: sha256:e728082cd6a2710840ef7d9fcdcd93408eb488aa05d10bc92f4454254e22cc4e
Status: Downloaded newer image for jenkins/jenkins:lts
docker.io/jenkins/jenkins:lts
usuario@usuario:~$

```



Antonio Salces
Alcaraz 24_25

php:7.3-apache

```

usuario@usuario:~$ sudo docker pull php:7.3-apache
7.3-apache: Pulling from library/php
ae13dd578326: Pull complete
f15d475049bf: Pull complete
886e5161983f: Pull complete
aa7666573a25: Pull complete
59357a0f9863: Pull complete
dc3ffb8c774e: Pull complete
513e9383f6d4: Pull complete
5ebd0737aa08: Pull complete
255df6c25392: Pull complete
f2994be86066: Pull complete
746eb0cc36a4: Pull complete
e46201569d4f: Pull complete
e05ec73939b3: Pull complete
a1488be2aff6: Pull complete
Digest: sha256:b9872cd287ef72bc17d45d713aa2742f3d3bcf2503fea2506fd93aa94995219f
Status: Downloaded newer image for php:7.3-apache
docker.io/library/php:7.3-apache
usuario@usuario:~$

```



Antonio Salces
Alcaraz 24_25

1.2. Mostrar imágenes

Para hacer esto, debemos de hacer uso del comando ***"docker images"***

```

usuario@usuario:~$ sudo docker images

```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
jenkins/jenkins	lts	e699ac6c40dc	3 weeks ago	467MB
mariadb	latest	6722945a6940	4 weeks ago	407MB
httpd	latest	494b2b45fd74	5 months ago	147MB
mysql	5.7	5107333e08a8	12 months ago	501MB
ubuntu	18.04	f9a80a55f492	19 months ago	63.2MB
debian	9	662c05203bab	2 years ago	101MB
php	7.3-apache	35da9118b3c0	2 years ago	451MB
centos	8	5d0da3dc9764	3 years ago	231MB
tomcat	9.0.39-jdk11	2703bbe9e9d4	4 years ago	648MB

```

usuario@usuario:~$

```



Antonio Salces
Alcaraz 24_25

1.3. Contenedor con Ubuntu 18.04, nombre “ubuntu”

Para arrancar el contenedor y darle un nombre, utilizaremos el comando “**docker run [-dit] [--name <nombre>] <imagen:versión> [comando]**”. También utilizaremos el comando “**docker ps -a**” para ver todos los contenedores del sistema y así poder saber el ID del docker de ubuntu.

```

usuario@usuario:~$ sudo docker run -dit --name ubuntu ubuntu:18.04 /bin/bash
0ee7f08d0498e914c9e26f407efc740862b4853d24c0115bc6f2cb2fba4f960c
usuario@usuario:~$ docker ps -a
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get "http://%2Fvar%2Frun%2Fdocker.sock/v1.47/containers/json?all=1": dial unix /var/run/docker.sock: connect: permission denied
usuario@usuario:~$ sudo docker ps -a

```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
0ee7f08d0498	ubuntu:18.04	"/bin/bash"	24 seconds ago	Up 22 seconds		ubuntu

```

usuario@usuario:~$

```

1.3.1. Parar contenedor

Para parar el contenedor, ejecutaremos el comando “**docker stop <ID>**”. Basta con poner el inicio del ID, no es necesario ponerlo entero. Para comprobar que está parado, basta con poner “**docker ps -a**” y comprobar el estado, donde se ve que está parado.

```

usuario@usuario:~$ sudo docker stop 0e
0e
usuario@usuario:~$ docker ps -a
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get "http://%2Fvar%2Frun%2Fdocker.sock/v1.47/containers/json?all=1": dial unix /var/run/docker.sock: connect: permission denied
usuario@usuario:~$ sudo docker ps -a

```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
0ee7f08d0498	ubuntu:18.04	"/bin/bash"	8 minutes ago	Exited (0) 3 minutes ago		ubuntu

```

usuario@usuario:~$

```

1.3.2. Rearrancar contenedor

Para rearrancar el contenedor, utilizaremos “**docker start <ID>**”, y para comprobar si se ha iniciado, “**docker ps -a**”.

```

root@usuario:/home/usuario# docker start 0e
0e
root@usuario:/home/usuario# docker ps -a

```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
0ee7f08d0498	ubuntu:18.04	"/bin/bash"	13 minutes ago	Up 5 seconds		ubuntu

```


root@usuario:/home/usuario#

```

1.3.3. Mostrar fichero sin entrar al contenedor

Para ello, debemos utilizar el comando ***"docker exec <nombre> cat /etc/os-release"***. Docker exec nos permite ejecutar comandos del contenedor desde fuera del mismo, mientras que "cat" se utiliza para visualizar archivos. En lugar del nombre, también podemos poner el ID.

```
root@usuario:/home/usuario# docker exec ubuntu cat /etc/os-release
NAME="Ubuntu"
VERSION="18.04.6 LTS (Bionic Beaver)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 18.04.6 LTS"
VERSION_ID="18.04"
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
VERSION_CODENAME=bionic
UBUNTU_CODENAME=bionic
root@usuario:/home/usuario# S
```



Antonio Salces
Alcaraz 24_25

2. Actividad 2: trabajar con PHP y MariaDB

2.1. Contenedor con PHP, nombre “web” y puerto 8181

Para iniciar el contenedor, utilizaremos el comando “**docker run [-d] [--name <nombre>] [-p <puerto>] <imagen>**”. Con “-d” ejecutaremos el docker en segundo plano, y con “-p” podremos elegir el puerto para el docker, en este caso el 8181.

```
root@usuario:/home/usuario# docker run -d --name web -p 8181:80 php:7.3-apache
e869274605951551dd83d2e5d8f7e1aa5c254d61e62f42f676e7af5925e6eb4d
root@usuario:/home/usuario# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                               NAMES
e86927460595   php:7.3-apache  "docker-php-entrypoi..." 12 seconds ago Up 10 seconds 0.0.0.0:8181->80/tcp, [::]:8181->80/tcp   web
```

2.2. Colocar ficheros index.html e index.php en el directorio web del contenedor

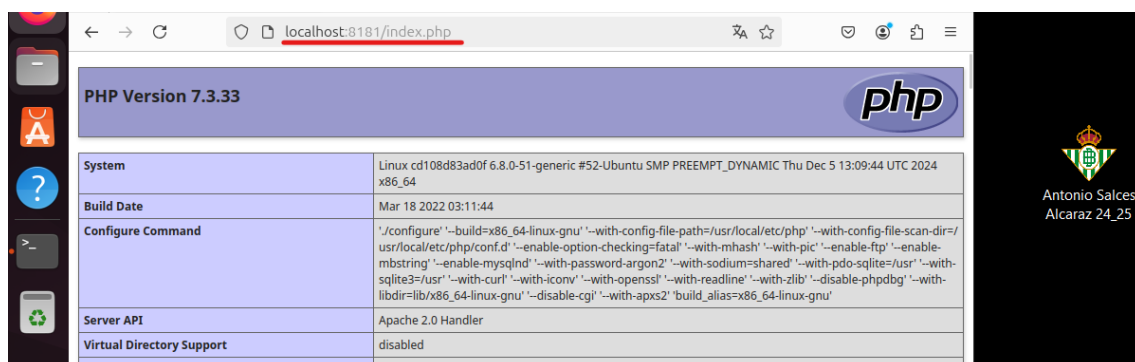
En primer lugar, accederemos al directorio web del contenedor con el primer comando que se ve en la imagen (“**docker exec -it [nombre o ID] bash**”). Una vez estemos en el contenedor, utilizaremos “**echo <contenido del archivo> > <ubicación del archivo>**”. Para comprobar el contenido del archivo, podemos utilizar “**cat <nombre>**”. Haremos lo mismo con el fichero “index.php”.

```
root@usuario:/home/usuario# docker exec -it web bash
root@e86927460595:/var/www/html# echo "<h1>HOLA SOY ANTONIO SALCES ALCARAZ</h1>" > /var/www/html/index.html
root@e86927460595:/var/www/html# cat index.html
<h1>HOLA SOY ANTONIO SALCES ALCARAZ</h1>
root@e86927460595:/var/www/html# echo "<?php phpinfo(); ?>" > /var/www/html/index.php
root@e86927460595:/var/www/html# cat index.php
<?php phpinfo(); ?>
root@e86927460595:/var/www/html#
```

Si buscamos la siguiente dirección en la maquina virtual, podremos ver el fichero index.html que hemos creado.



Y esto saldrá si buscamos el fichero index.php.



2.3. Contenedor con mariadb con diferentes parámetros

Debemos arrancar un contenedor que contenga “mariadb” con los siguientes parámetros:

- Nombre: bddd.
- Puerto: 3336.
- Contraseña de root: root.
- Crear base de datos con nombre: prueba.
- Contraseña del usuario: usuario.

```

root@usuario:/home/usuario# docker run --name bddd -e MYSQL_ROOT_PASSWORD=root -e MYSQL_DATABASE=prueba -e MYSQL_USER=invitado -e MYSQL_PASSWORD=invitado -p 3336:3306 -d mariadb
69f98e31a59a9d13b02c0cc148f469a83157fc31d13ba399c1e7b944f69baa6
root@usuario:/home/usuario# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED    STATUS    PORTS                               NAMES
69f98e31a59a   mariadb   "docker-entrypoint.s..." 3 seconds ago Up 2 seconds    0.0.0.0:3336->3306/tcp, :::3336->3306/tcp    bddd
cd108d83ad0f   php:7.3-apache   "docker-php-entrypoi..." 11 minutes ago Up 11 minutes    0.0.0.0:8181->80/tcp, :::8181->80/tcp        web
root@usuario:/home/usuario#

```

Los parámetros que se han utilizado son los siguientes:

- --name bddd: el nombre del docker es “bddd”.
- -e MYSQL_ROOT_PASSWORD=root: la contraseña del root será “root”.
- -e MYSQL_DATABASE=prueba: crea la base de datos llamada “prueba”.
- -e MYSQL_USER=invitado: crea el usuario “invitado”.
- -e MYSQL_PASSWORD=invitado: la contraseña del usuario “invitado” será “invitado”.
- -p 3336:3306: indica el puerto a utilizar.
- -d mariadb: lanzaremos “mariadb” en segundo plano.

2.4. Conectar con un cliente de base de datos

En primer lugar, debemos de instalar un cliente de base de datos. En mi caso, utilizaré un cliente de terminal (mysql-client). Utilizaremos el siguiente comando para instalarlo: “**apt install <paquete>**”.

```

root@usuario:/home/usuario# apt install mysql-client-core-8.0
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes NUEVOS:
  mysql-client-core-8.0
0 actualizados, 1 nuevos se instalarán, 0 para eliminar y 4 no actualizados.
Se necesita descargar 2.765 kB de archivos.
Se utilizarán 61,6 MB de espacio de disco adicional después de esta operación.
Des:1 http://es.archive.ubuntu.com/ubuntu noble-updates/main amd64 mysql-client-core-8.0 amd64 8.0.40-0ubuntu0.24.04.1 [2.765 kB]
Descargados 2.765 kB en 2s (1.578 kB/s)
Seleccionando el paquete mysql-client-core-8.0 previamente no seleccionado.
(Leyendo la base de datos ... 150724 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../mysql-client-core-8.0_8.0.40-0ubuntu0.24.04.1_amd64.deb ...
Desempaquetando mysql-client-core-8.0 (8.0.40-0ubuntu0.24.04.1) ...
Configurando mysql-client-core-8.0 (8.0.40-0ubuntu0.24.04.1) ...
Procesando disparadores para man-db (2.12.0-4build2) ...
root@usuario:/home/usuario#

```

Con el primer comando conseguiremos conectarnos con la base de datos del contenedor. El comando es el siguiente: “**mysql <-h IP> <-P puerto> <-u usuario> [-p]**”. “-p” indica que tenemos que introducir una contraseña para entrar.

Una vez hayamos conectado con la base de datos, podemos utilizar el comando “**SHOW DATABASES;**” para ver las bases de datos creadas. Podemos ver en la imagen que la base de datos “prueba” se ha creado.

```
root@usuario:/home/usuario# mysql -h 127.0.0.1 -P 3336 -u invitado -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5
Server version: 11.6.2-MariaDB-ubu2404 mariadb.org binary distribution

Copyright (c) 2000, 2024, 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> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| prueba |
+-----+
2 rows in set (0,01 sec)

mysql> 
```



3. Actividad 3: gestión de imágenes

3.1. Descargar imagen Ubuntu:20.04

Para descargar la imagen, utilizaremos el comando ***"docker pull ubuntu:20.04"***, y una vez se haya completado la descarga, para comprobar que está en nuestro sistema, utilizaremos el comando ***"docker images"***.

```
root@usuario:/home/usuario# docker pull ubuntu:20.04
20.04: Pulling from library/ubuntu
d9802f032d67: Pull complete
Digest: sha256:8e5c4f0285ecbb4ead070431d29b576a530d3166df73ec44affc1cd27555141b
Status: Downloaded newer image for ubuntu:20.04
docker.io/library/ubuntu:20.04
root@usuario:/home/usuario# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
jenkins/jenkins	lts	e699ac6c40dc	5 weeks ago	467MB
mariadb	latest	6722945a6940	5 weeks ago	407MB
ubuntu	20.04	6013ae1a63c2	2 months ago	72.8MB
httpd	latest	4ce47c750a58	5 months ago	147MB
mysql	5.7	5107333e08a8	12 months ago	501MB
ubuntu	18.04	f9a80a55f492	19 months ago	63.2MB
debian	9	662c05203bab	2 years ago	101MB
php	7.3-apache	35da9118b3c0	2 years ago	451MB
centos	8	5d0da3dc9764	3 years ago	231MB
tomcat	9.0.39-jdk11	2703bbe9e9d4	4 years ago	648MB

```
root@usuario:/home/usuario#
```



Antonio Salces
Alcaraz 24_25

3.2. Volcar la información de la nueva imagen en un archivo

Para ver la información de la nueva imagen y volcarla en un archivo, utilizaremos el comando ***"docker inspect <imagen> > <ruta>"***. Para ver la información guardada en el archivo, utilizamos el comando ***"cat <ruta>"***.

```
root@usuario:/home/usuario# docker inspect ubuntu:20.04 > /home/usuario/Escritorio/info.txt
root@usuario:/home/usuario# cat /home/usuario/Escritorio/info.txt
```

```
[
  {
    "Id": "sha256:6013ae1a63c2ee58a8949f03c6366a3ef6a2f386a7db27d86de2de965e9f450b",
    "RepoTags": [
      "ubuntu:20.04"
    ],
    "RepoDigests": [
      "ubuntu@sha256:8e5c4f0285ecbb4ead070431d29b576a530d3166df73ec44affc1cd27555141b"
    ],
    "Parent": "",
    "Comment": "",
    "Created": "2024-10-11T03:38:27.357079367Z",
    "DockerVersion": "24.0.7",
    "Author": "",
    "Config": {
      "Hostname": "",
      "Domainname": "",
      "User": "",
      "AttachStdin": false,
      "AttachStdout": false,
      "AttachStderr": false,
```



Antonio Salces
Alcaraz 24_25

3.3. Crear contenedor con Ubuntu:20.04 llamada "modulo3"

Utilizaremos el comando **"docker run -d --name modulo3 ubuntu:20.04"** para crear el contenedor con los parámetros especificados. Con el comando **"docker ps -a"** veremos el contenedor.

```
root@usuario:/home/usuario# docker run -d --name modulo3 ubuntu:20.04
1b5a179cb2d8de09fb18c201312ec7dc92f083177ac41e31cb62f1cfc1edf6b6
root@usuario:/home/usuario# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
1b5a179cb2d8	ubuntu:20.04	"/bin/bash"	5 seconds ago	Exited (0) 3 seconds ago		modulo3
69f98e31a59a	mariadb	"docker-entrypoint.s..."	3 days ago	Exited (0) 2 days ago		bbdd
cd108d83ad0f	php:7.3-apache	"docker-php-entrypoi..."	3 days ago	Exited (0) 3 days ago		web

```
root@usuario:/home/usuario#
```



3.4. Borrar contenedor con Ubuntu:20.04

Para borrar una imagen, utilizaremos el comando **"docker rmi <imagen>"**. Como podemos ver, no podemos borrar la imagen. Esto es debido a que la imagen se está utilizando en un contenedor, por lo que no podremos borrarla mientras el contenedor exista.

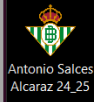
```
root@usuario:/home/usuario# docker rmi ubuntu:20.04
Error response from daemon: conflict: unable to remove repository reference "ubuntu:20.04" (must force) - container 1b5a179cb2d8 is using its referenced image 6013ae1a63c2
root@usuario:/home/usuario#
```



3.5. Hacer acciones necesarias para borrar el contenedor Ubuntu:20.04

Para borrar la imagen, en primer lugar debemos parar el contenedor con **"docker stop <contenedor>"**, luego borrar con **"docker rm <contenedor>"**, y por ultimo borrar la imagen con **"docker rmi <imagen>"**.

```
root@usuario:/home/usuario# docker stop modulo3
modulo3
root@usuario:/home/usuario# docker rm modulo3
modulo3
root@usuario:/home/usuario# docker rmi ubuntu:20.04
Untagged: ubuntu:20.04
Untagged: ubuntu@sha256:8e5c4f0285ecbb4ead070431d29b576a530d3166df73ec44affc1cd27555141b
Deleted: sha256:6013ae1a63c2ee58a8949f03c6366a3ef6a2f386a7db27d86de2de965e9f450b
Deleted: sha256:fffe76c64ef2dee2d80a8bb3ad13d65d596d04a45510b1956a976a69215dae92
root@usuario:/home/usuario#
```



4. Actividad 4: operaciones con volúmenes

4.1. Crear volúmenes

Con el comando ***"docker volume create <nombre>"*** podremos crear los volúmenes con el nombre que queramos. Luego, utilizaremos ***"docker volume ls"*** para ver los volúmenes creados.

```
root@usuario:/home/usuario# docker volume create volumen_datos
volumen_datos
root@usuario:/home/usuario# docker volume create volumen_web
volumen_web
root@usuario:/home/usuario# docker volume ls
DRIVER      VOLUME NAME
local       e6893b55a515fce92167af832f8e9fe5ca29cb3911b67d258916613f6732f019
local       volumen_datos
local       volumen_web
root@usuario:/home/usuario#
```



4.2. Instanciar contenedores en los volúmenes

Con el comando ***"docker run [-d] [--name <nombre>] [-v <volumen>:][ruta] <imagen>"*** podremos crear un contenedor y elegir en que volumen y que ruta queremos asignarle. Con ***"docker ps"*** veremos que el contenedor está activo

```
root@usuario:/home/usuario# docker run -d --name c1 -v volumen_web:/var/www/html php:7.4-apache
4fcb4f9fca57a0fa960701bd6405a1a17efb3e3be00ac8319eb777fb7bcd3920
root@usuario:/home/usuario# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS        NAMES
4fcb4f9fca57   php:7.4-apache   "docker-php-entrypoi..."   9 seconds ago   Up 7 seconds   80/tcp       c1
root@usuario:/home/usuario#
```



Con el comando ***"docker run [-d] [--name <nombre>] > [-v <volumen>:][ruta] [-e<MYSQL_ROOT_PASSWORD>=<contraseña>] <imagen>"*** podremos crear un contenedor, elegir el volumen y la ruta donde se aloja y, en este caso (se trata de MariaDB) podemos asignar también una contraseña al usuario "root".

```
root@usuario:/home/usuario# docker run -d --name c2 -v volumen_datos:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=admin mariadb
fea2b1eb1d73e6e259b0295058b05b4f7af654c00d07e711676af81e869a4192
root@usuario:/home/usuario# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS        NAMES
fea2b1eb1d73   mariadb   "docker-entrypoint.s..."   15 seconds ago   Up 14 seconds   3306/tcp     c2
4fcb4f9fca57   php:7.4-apache   "docker-php-entrypoi..."   4 minutes ago   Up 4 minutes   80/tcp       c1
root@usuario:/home/usuario#
```



4.3. Parar y borrar contenedor “c2” y volumen “volumen_datos”

En primer lugar, paramos el contenedor “c2” con “**docker stop <nombre o ID>**”, luego, lo borramos con “**docker rm <nombre o ID>**”, y a continuación borramos el volumen con “**docker volume rm <nombre>**”. Para acabar, comprobaremos que se han eliminado con “**docker ps**” y “**docker volume ls**” y que ya solo queda el contenedor “c1”.

```

root@usuario:/home/usuario# docker stop c2
c2
root@usuario:/home/usuario# docker rm c2
c2
root@usuario:/home/usuario# docker volume rm volumen_datos
volumen_datos
root@usuario:/home/usuario# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS        NAMES
4fcb4f9fca57   php:7.4-apache   "docker-php-entrypoi..."   14 minutes ago   Up 14 minutes   80/tcp        c1
root@usuario:/home/usuario# docker volumen ls
docker: 'volumen' is not a docker command.
See 'docker --help'
root@usuario:/home/usuario# docker volume ls
DRIVER      VOLUME NAME
local       73f52a5ff2bbefc42903e4315da181884b4b5fb6d8f46a64fcb48e2f15dbf043
local       e6893b55a515fce92167af832f8e9fe5ca29cb3911b67d258916613f6732f019
local       volumen_web
root@usuario:/home/usuario#

```

4.4. Comprobar que “c1” está montado sobre “volumen_web”

Debemos utilizar la orden “**docker inspect <nombre o ID>**”.

```

2 de ene 19:19
root@usuario:/home/usuario# docker inspect c1
[
  {

```

Ahora bajamos hasta la parte de “Mounts” y comprobamos en que volumen está. En este caso, está sobre “volumen_web”.

```

"Mounts": [
  {
    "Type": "volume",
    "Name": "volumen_web",
    "Source": "/var/lib/docker/volumes/volumen_web/_data",
    "Destination": "/var/www/html",
    "Driver": "local",
    "Mode": "z",
    "RW": true,
    "Propagation": ""
  }
]

```


5. Actividad 5: trabajo con redes

5.1. Creación de redes

En primer lugar, debemos de crear una red con los siguientes requisitos:

- **Nombre:** red1
- **Dirección de red:** 172.28.0.0
- **Mascara de red:** 255.255.0.0
- **Gateway:** 172.28.0.1
- **Tipo:** bridge

Para ello, ejecutaremos el siguiente comando: “**docker network create [--subnet <IP/mascara>] [--gateway <IP>] <nombre>**”.

```
root@usuario:/home/usuario# docker network create --subnet 172.28.0.0/16 --gateway 172.28.0.1 red1
4b091d5430018058ac1b47ca5baae4e507ae18b708ba9f47719cf0f7b97aaad5
root@usuario:/home/usuario#
```

Ahora crearemos la segunda red, la cual solo tendrá configurada el nombre.

```
root@usuario:/home/usuario# docker network create red2
470779a68b7f0f3a467e132d8956c241aeff3a302f9c6be5d8d4125e17f02eb
root@usuario:/home/usuario#
```

5.2. Configuración de ambas redes

Para ver ahora las redes creadas, utilizaremos “**docker network ls**”.

```
root@usuario:/home/usuario# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
be8b73ad997a        bridge              bridge               local
46bd87c7a96e        host                host                 local
ca0527acf577        mi_red              bridge               local
61b30db623b8        none                null                 local
4b091d543001        red1                 bridge               local
470779a68b7f        red2                 bridge               local
0dfb4f9a2b45        rubyonrails_default bridge               local
root@usuario:/home/usuario#
```

Con “**docker inspect <red>**” podremos ver la configuración de red.

```
root@usuario:/home/usuario# docker inspect red1
[
  {
    "Name": "red1",
    "Id": "4b091d5430018058ac1b47ca5baae4e507ae18b708ba9f47719cf0f7b97aad5",
    "Created": "2025-01-16T08:54:54.242744444+01:00",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": {},
      "Config": [
        {
          "Subnet": "172.28.0.0/16",
          "Gateway": "172.28.0.1"
        }
      ]
    }
  }
],
```

```
root@usuario:/home/usuario# docker inspect red2
[
  {
    "Name": "red2",
    "Id": "470779a68b7f0f3a467e132d8956c241aeff3a302f9c6be5d8d4125e17f02eb",
    "Created": "2025-01-16T08:58:59.887020718+01:00",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": {},
      "Config": [
        {
          "Subnet": "172.18.0.0/16",
          "Gateway": "172.18.0.1"
        }
      ]
    }
  }
],
```

5.3. Arrancar contenedor Ubuntu:20.04 con configuración específica

Debemos arrancar un contenedor con la siguiente configuración:

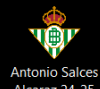
- **Nombre:** u1
- **Hostname:** host1
- **Red:** red1
- **Ip:** 172.28.0.10
- **Imagen:** Ubuntu:20.04

Para ello, utilizaremos el comando “**docker run [-dit] [--name <nombre>] [--hostname <nombre>] [--net <red>] [--ip <IP>] <imagen>**”.

```
root@usuario:/home/usuario# docker run -dit --name u1 --hostname host1 --net red1 --ip 172.28.0.10 ubuntu:20.04
68eb83ee0ad4dab2ead59bdf0599a992a43add97e9b5a02791ee80bbad059343
root@usuario:/home/usuario#
```

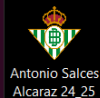
Ahora nos meteremos en el contenedor e instalaremos el comando “*ping*”. Para ello, primero tenemos que meternos en el contenedor con el comando “***docker exec -it <nombre o ID> /bin/bash***”. Después, actualizaremos el contenedor con “***apt update***”.

```
root@usuario:/home/usuario# docker exec -it u1 /bin/bash
root@host1:/# apt update
Get:1 http://security.ubuntu.com/ubuntu focal-security InRelease [128 kB]
Get:2 http://archive.ubuntu.com/ubuntu focal InRelease [265 kB]
Get:3 http://security.ubuntu.com/ubuntu focal-security/universe amd64 Packages [1297 kB]
Get:4 http://archive.ubuntu.com/ubuntu focal-updates InRelease [128 kB]
Get:5 http://archive.ubuntu.com/ubuntu focal-backports InRelease [128 kB]
Get:6 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [4186 kB]
Get:7 http://archive.ubuntu.com/ubuntu focal/main amd64 Packages [1275 kB]
```



Tras tener el contenedor actualizado, utilizaremos el comando “***apt install inetutils-ping***” para instalar el comando “*ping*”.

```
root@host1:/# apt install inetutils-ping
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libidn11 netbase
The following NEW packages will be installed:
  inetutils-ping libidn11 netbase
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
Need to get 120 kB of archives.
After this operation, 658 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
```



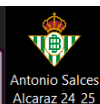
5.4. Arrancar otro contenedor Ubuntu:20.04 con otra configuración

Debemos arrancar otro contenedor con Ubuntu:20.04 con la siguiente configuración:

- **Nombre:** u2
- **Hostname:** host2
- **Red:** red2
- **IP:** automática
- **Imagen:** Ubuntu:20.04

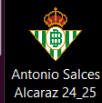
Como con el contenedor anterior, utilizaremos el siguiente comando: “***docker run [-dit] [--name <nombre>] [--hostname <nombre>] [--net <red>] <imagen>***”.

```
root@usuario:/home/usuario# docker run -dit --name u2 --hostname host2 --net red2 ubuntu:20.04
2821d599546e1f6a9e19bd36e5b8b541c8164a4db972f7c4708198bfa8720558
root@usuario:/home/usuario#
```

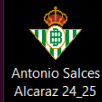


Como con el contenedor anterior, entraremos en él, lo actualizaremos e instalaremos el comando “ping”.

```
root@usuario:/home/usuario# docker exec -it u2 /bin/bash
root@host2:/# apt update
Get:1 http://security.ubuntu.com/ubuntu focal-security InRelease [128 kB]
Get:2 http://archive.ubuntu.com/ubuntu focal InRelease [265 kB]
Get:3 http://security.ubuntu.com/ubuntu focal-security/universe amd64 Packages [1297 kB]
Get:4 http://security.ubuntu.com/ubuntu focal-security/restricted amd64 Packages [4276 kB]
Get:5 http://archive.ubuntu.com/ubuntu focal-updates InRelease [128 kB]
Get:6 http://archive.ubuntu.com/ubuntu focal-backports InRelease [128 kB]
Get:7 http://archive.ubuntu.com/ubuntu focal/universe amd64 Packages [11.3 MB]
```



```
root@host2:/# apt install inetutils-ping
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libidn11 netbase
The following NEW packages will be installed:
  inetutils-ping libidn11 netbase
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
Need to get 120 kB of archives.
After this operation, 658 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
```



5.5. Configuración de red de ambos contenedores

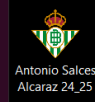
Para ello, debemos hacer “**docker inspect <nombre o ID>**” con ambos contenedores.

En las siguientes capturas podemos ver en que red y la IP del contenedor “u1”.

```
root@usuario:/home/usuario# docker inspect u1
[
  {
    "Id": "68eb83ee0ad4dab2ead59bdf0599a992a43add97e9b5a02791ee80bbad059343",
    "Created": "2025-01-16T08:40:56.66739166Z",
    "Path": "/bin/bash",
    "Args": [],
    "State": {
      "Status": "running",
      "Running": true,
      "Paused": false,
      "Restarting": false,
      "OOMKilled": false,
      "Dead": false,
      "Pid": 7703,
      "Exitcode": 0,
      "Error": "",
      "StartedAt": "2025-01-16T08:40:57.001260098Z",
      "FinishedAt": "0001-01-01T00:00:00Z"
    },
    "Image": "sha256:6813ae1a62c2ee58a8949f03c636a3ef6a2f386a7db27d86de2de965e9f450b",
    "ResolvConfPath": "/var/lib/docker/containers/68eb83ee0ad4dab2ead59bdf0599a992a43add97e9b5a02791ee80bbad059343/resolv.conf",
    "HostnamePath": "/var/lib/docker/containers/68eb83ee0ad4dab2ead59bdf0599a992a43add97e9b5a02791ee80bbad059343/hostname",
    "HostsPath": "/var/lib/docker/containers/68eb83ee0ad4dab2ead59bdf0599a992a43add97e9b5a02791ee80bbad059343/hosts",
    "LogPath": "/var/lib/docker/containers/68eb83ee0ad4dab2ead59bdf0599a992a43add97e9b5a02791ee80bbad059343/68eb83ee0ad4dab2ead59bdf0599a992a43add97e9b5a02791ee80bbad059343-json.log",
    "Name": "/u1",
    "RestartCount": 0,
    "Driver": "overlay2",
    "Platform": "linux",
    "MountLabel": "",
    "ProcessLabel": "",
    "AppArmorProfile": "docker-default",
    "ExecIDs": null,
    "HostConfig": {
      "Binds": null,
      "ContainerIDFile": "",
      "LogConfig": {
        "Type": "json-file",
        "Config": {}
      },
      "NetworkMode": "red1",
    }
  }
]
```



```
"NetworkSettings": {
  "Bridge": "",
  "SandboxID": "69fc5c800a62353baa32a8a2847e4dd1565069457a25938a5b9ad370b210f008",
  "SandboxKey": "/var/run/docker/netns/69fc5c800a62",
  "Ports": {},
  "HairpinMode": false,
  "LinkLocalIPv6Address": "",
  "LinkLocalIPv6PrefixLen": 0,
  "SecondaryIPAddresses": null,
  "SecondaryIPv6Addresses": null,
  "EndpointID": "",
  "Gateway": "",
  "GlobalIPv6Address": "",
  "GlobalIPv6PrefixLen": 0,
  "IPAddress": "",
  "IPPrefixLen": 0,
  "IPv6Gateway": "",
  "MacAddress": "",
  "Networks": {
    "red1": {
      "IPAMConfig": {
        "IPv4Address": "172.28.0.10"
      }
    }
  }
},
```



Ahora, en las siguientes dos capturas, se mostrará la configuración de red del contenedor “u2”.

```
root@usuario:/home/usuario# docker inspect u2
[
  {
    "Id": "2821d599546e1f6a9e19bd36e5bb541c8164a4db972f7c4708198bfa8720558",
    "Created": "2025-01-16T08:53:49.089952935Z",
    "Path": "/bin/bash",
    "Args": [],
    "State": {
      "Status": "exited",
      "Running": false,
      "Paused": false,
      "Restarting": false,
      "OOMKilled": false,
      "Dead": false,
      "Pid": 0,
      "ExitCode": 0,
      "Error": "",
      "StartedAt": "2025-01-16T08:53:49.480266773Z",
      "FinishedAt": "2025-01-16T09:16:30.239507718Z"
    },
    "Image": "sha256:6013ae1a032ee58a0949f03c036a3ef6a2f386a7db27d86de2de9e5e9f450b",
    "ResolveConfPath": "/var/lib/docker/containers/2821d599546e1f6a9e19bd36e5bb541c8164a4db972f7c4708198bfa8720558/resolve.conf",
    "HostnamePath": "/var/lib/docker/containers/2821d599546e1f6a9e19bd36e5bb541c8164a4db972f7c4708198bfa8720558/hostname",
    "HostsPath": "/var/lib/docker/containers/2821d599546e1f6a9e19bd36e5bb541c8164a4db972f7c4708198bfa8720558/hosts",
    "LogPath": "/var/lib/docker/containers/2821d599546e1f6a9e19bd36e5bb541c8164a4db972f7c4708198bfa8720558/2821d599546e1f6a9e19bd36e5bb541c8164a4db972f7c4708198bfa8720558-logs.log",
    "Name": "/u2",
    "RestartCount": 0,
    "Driver": "overlay2",
    "Platform": "linux",
    "MountLabel": "",
    "ProcessLabel": "",
    "AppArmorProfile": "docker-default",
    "ExecIDs": null,
    "HostConfig": {
      "Binds": null,
      "ContainerIDFile": "",
      "LogConfig": {
        "type": "json-file",
        "config": {}
      },
      "NetworkMode": "red2"
    }
  },
  "NetworkMode": "red2"
]
```

```
"Networks": {
  "red2": {
    "IPAMConfig": null,
    "Links": null,
    "Aliases": null,
    "MacAddress": "",
    "DriverOpts": null,
    "NetworkID": "470779a68b7f0f3a467e132d8956c241aefff3a302f9c6be5d8d4125e17f02eb",
    "EndpointID": "",
    "Gateway": "",
    "IPAddress": "",
    "IPPrefixLen": 0,
    "IPv6Gateway": "",
    "GlobalIPv6Address": "",
    "GlobalIPv6PrefixLen": 0,
    "DNSNames": [
      "u2",
      "2821d599546e",
      "host2"
    ]
  }
}
```

5.6. Comprobar conectividad entre contenedores

En primer lugar, nos debemos de meter en cualquier contenedor (en mi caso “u2”) con el comando “**docker exec -it <nombre o ID> /bin/bash**”. Una vez estemos en el contenedor, ejecutaremos el comando “**ping <nombre o ID>**” (ejecutaremos ambos), para comprobar si ambos contenedores tienen conectividad.

```
root@usuario:/home/usuario# docker exec -it u2 /bin/bash
root@host2:/# ping 172.28.0.10
PING 172.28.0.10 (172.28.0.10): 56 data bytes
^C--- 172.28.0.10 ping statistics ---
156 packets transmitted, 0 packets received, 100% packet loss
root@host2:/# ping u1
ping: unknown host
root@host2:/#
```

Como podemos ver en la imagen anterior, los contenedores no tienen conectividad entre ellos.

5.7. Conectar el contenedor “u1” a “red2”

Para comenzar, utilizaremos el comando **“docker network connect <nombre de red> <nombre de contenedor o ID>”** para conectar “u1” a “red2”. Una vez el contenedor se encuentre en su nueva red, nos meteremos en él con **“docker exec -it <nombre o ID> /bin/bash”**. Cuando estemos dentro del contenedor ejecutaremos **“ping <nombre o IP>”** (en este caso ambas), para comprobar si tienen conectividad.

```
root@usuario:/home/usuario# docker network connect red2 u1
root@usuario:/home/usuario# docker exec -it u1 /bin/bash
root@host1:/# ping u2
PING u2 (172.18.0.2): 56 data bytes
64 bytes from 172.18.0.2: icmp_seq=0 ttl=64 time=0.110 ms
64 bytes from 172.18.0.2: icmp_seq=1 ttl=64 time=0.080 ms
64 bytes from 172.18.0.2: icmp_seq=2 ttl=64 time=0.079 ms
64 bytes from 172.18.0.2: icmp_seq=3 ttl=64 time=0.083 ms
^C-- u2 ping statistics --
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.079/0.088/0.110/0.000 ms
root@host1:/# ping 172.18.0.2
PING 172.18.0.2 (172.18.0.2): 56 data bytes
64 bytes from 172.18.0.2: icmp_seq=0 ttl=64 time=0.356 ms
64 bytes from 172.18.0.2: icmp_seq=1 ttl=64 time=0.077 ms
64 bytes from 172.18.0.2: icmp_seq=2 ttl=64 time=0.108 ms
64 bytes from 172.18.0.2: icmp_seq=3 ttl=64 time=0.109 ms
^C-- 172.18.0.2 ping statistics --
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.077/0.163/0.356/0.112 ms
root@host1:/#
```



Antonio Salces
Alcaraz 24_25

Como vemos en la imagen anterior, ambos contenedores tienen conexión entre ellos, como podemos ver en el mensaje traducido al español **“4 paquetes trasmitidos, 4 paquetes recibidos”**.

6. Actividad 5: editar fichero de configuración .yaml

6.1. Descargar archivo de configuración .yaml para docker-compose

En primer lugar debemos de crear una carpeta donde guardaremos el archivo .yaml y ejecutaremos el contenedor. Tras crearla y meternos en ella, ejecutaremos el comando para descargarnos el archivo que viene en la página para descargar *Ruby on Rails*.

```

usuario@usuario:~$ mkdir RubyOnRails
usuario@usuario:~$ cd RubyOnRails/
usuario@usuario:~/RubyOnRails$ curl -LO https://raw.githubusercontent.com/bitnami/containers/main/bitnami/rails/docker-compose.yml
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
100  506  100  506    0     0  1743      0  --:--:-- --:--:-- --:--:--  1744
usuario@usuario:~/RubyOnRails$

```



6.2. Editar el archivo .yaml y cambiar puerto

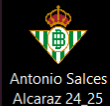
Ahora utilizaremos el editor “nano” para editar el archivo de configuración y poder cambiar el puerto al indicado en el enunciado de la práctica (8001). Para ello, buscaremos la sección de “ports”.

```

GNU nano 7.2 docker-compose.yml *
# Copyright Bitnami, Inc. All Rights Reserved.
# SPDX-License-Identifier: APACHE-2.0

services:
  mariadb:
    image: docker.io/bitnami/mariadb:latest
    environment:
      # ALLOW_EMPTY_PASSWORD is recommended only for development.
      - ALLOW_EMPTY_PASSWORD=yes
  myapp:
    image: docker.io/bitnami/rails:8
    ports:
      - '8001:3000'
    environment:
      - DATABASE_HOST=mariadb
      - DATABASE_NAME=my_app_development
    volumes:
      - './my-project:/app'
    depends_on:

```



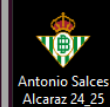
6.3. Arrancar con “docker-compose”

Tras editar el fichero de configuración, utilizaremos “**docker-compose up [-d]**” para iniciar los contenedores con la configuración del archivo .yaml.

```

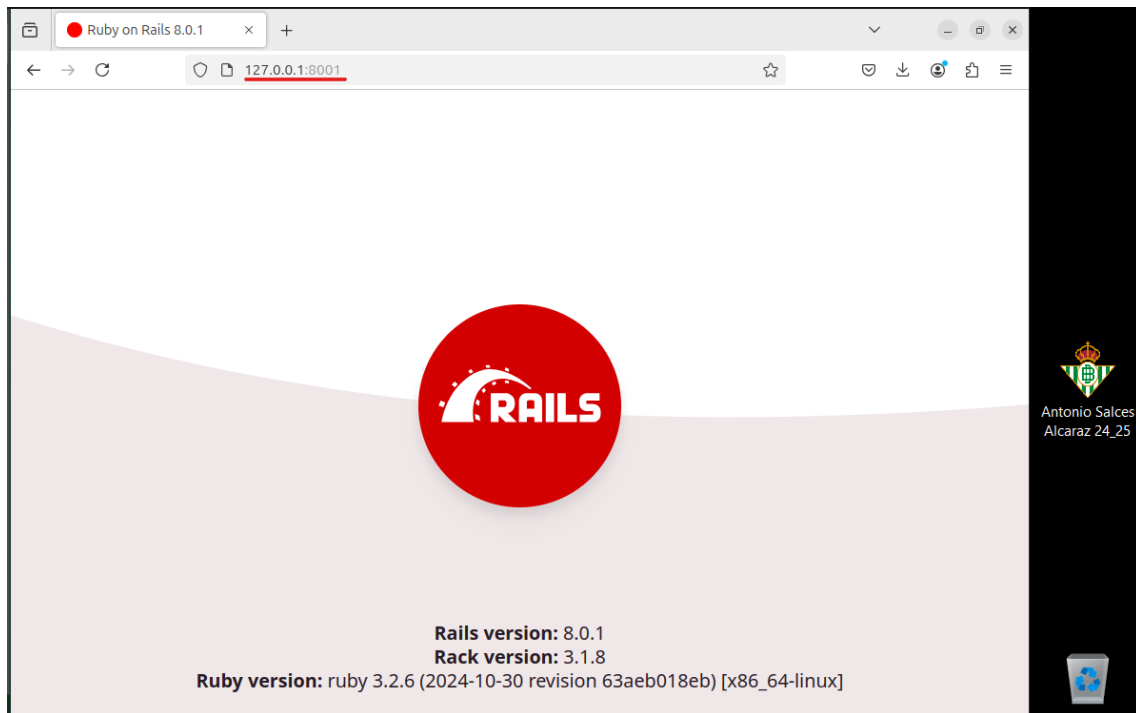
root@usuario:/home/usuario/RubyOnRails# docker compose up -d
[+] Running 3/3
 ✓ Network rubyonrails_default      Created           0.4s
 ✓ Container rubyonrails-mariadb-1   Started           1.4s
 ✓ Container rubyonrails-myapp-1     Started           2.0s
root@usuario:/home/usuario/RubyOnRails#

```



6.4. Entrar en el contenedor mediante el navegador

Ahora, si buscamos en el buscador “127.0.0.1:8001”, podremos ver el contenedor.

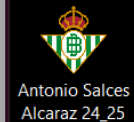


7. Operaciones con imágenes no firmadas

7.1. Descargar imagen no firmada

Para descargar la imagen indicada, debemos utilizar el comando ***"docker pull <imagen>"***.

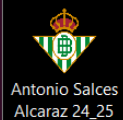
```
root@usuario:/home/usuario# docker pull testinf/holamundohtml
Using default tag: latest
latest: Pulling from testinf/holamundohtml
6ec7b7d162b2: Pull complete
db606474d60c: Pull complete
afb30f0cd8e0: Pull complete
3bb2e8051594: Pull complete
4c761b44e2cc: Pull complete
c2199db96575: Pull complete
1b9a9381eea8: Pull complete
50f0689715a3: Pull complete
8a6cc018dd45: Pull complete
052299cf2d76: Pull complete
ee83da709c88: Pull complete
5b10df91e6d0: Pull complete
a2eb858e27d8: Pull complete
ff8ee7220ee5: Pull complete
Digest: sha256:02f123b02628b142844fa5537024909fcfb3ae3fc6dbfc3ce1d78fd3dc7aa0ee
Status: Downloaded newer image for testinf/holamundohtml:latest
docker.io/testinf/holamundohtml:latest
```



Ahora tenemos que eliminar la imagen. Para ello, primero buscaré el ID de la imagen con ***"docker images"***, y luego utilizaré ***"docker rmi <ID>"***.

```
root@usuario:/home/usuario# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
bitnami/rails        8                   5684bb6f6b60       5 days ago         1.09GB
bitnami/mariadb      latest             a770888cf3cf       8 days ago         432MB
jenkins/jenkins      lts                e699ac6c40dc       7 weeks ago        467MB
nginx                latest             f876bfc1cc63       7 weeks ago        192MB
mariadb              latest             6722945a6940       8 weeks ago        407MB
wordpress            latest             f4c026a8ee03       8 weeks ago        700MB
mysql                latest             56a8c14e1404       3 months ago       603MB
ubuntu               20.04             6013ae1a63c2       3 months ago       72.8MB
httpd                latest             4ce47c750a58       6 months ago       147MB
mysql                5.7               5107333e08a8       13 months ago      501MB
ubuntu               18.04             f9a80a55f492       19 months ago      63.2MB
php                  7.4-apache        20a3732f422b       2 years ago        453MB
debian               9                  662c05203bab       2 years ago        101MB
php                  7.3-apache        35da9118b3c0       2 years ago        451MB
centos                8                  5d0da3dc9764       3 years ago        231MB
testinf/holamundohtml latest            b009c72cbee8       4 years ago        414MB
tomcat                9.0.39-jdk11      2703bbe9e9d4       4 years ago        648MB

root@usuario:/home/usuario# docker rmi b0
Untagged: testinf/holamundohtml:latest
Untagged: testinf/holamundohtml@sha256:02f123b02628b142844fa5537024909fcfb3ae3fc6dbfc3ce1d78fd3dc7aa0ee
Deleted: sha256:b009c72cbee84e0e614264c0ad6908dcac65cb804157f36e6d59b17381d1edf3
Deleted: sha256:9b50f7b128b281817ca76941ba8fa036994996b6e3536a66d9abea4fffd8484
Deleted: sha256:d6fac6e7a66b3efb451d3f45a05315baaab984decfb1adaf2e4d9db561fc4bba
Deleted: sha256:02c727e2f2c6ca6d8d76db5c1499fa9b690dc80a1459088b3070ddc515ff30e5
Deleted: sha256:4fa0d165da6a4b6c8baf9a8c4935b40d617f34c09e44f43fc2a0c2295170e160
Deleted: sha256:b7da1953995a5096b9eb7741709e92512430ae2f0e5e985b210d8a5f6bf28688
Deleted: sha256:e8588b67d5b69e1870432b23b18042175db1a4490680b9f54920e7dd684d5e99
Deleted: sha256:77739749c650f1ccccbe2263013a498e760f0b7434124711ae69efc1dfcb045f
Deleted: sha256:2ee746151beb7b0b3ae453bb83c6d0828ee5671a14f60160da05c663f6b4286a
Deleted: sha256:cb7a8356ede0b61bd6f540981c5b1d91b7e5385412e36684f65f904bbf590c92
Deleted: sha256:698904aae259d168b38dad3c8d8c6787b722e92f4b9dfcee0ee8a2ab8240f4c5
Deleted: sha256:a953d04dec53b836e21dfc15a2b8f9bbecaad1085ef606b0d9db21b09078337
Deleted: sha256:8a8f49ca28d2e5ca5aa9455c92331b26ae1e31d2c78a6be0e062429cf3812552
Deleted: sha256:db10680b18837fbef39e1e6452ff5448803e1e7b0c13eda5d53b19743e09d947
Deleted: sha256:87c8a1d8f54f3aa4e05569e8919397b65056aa71cdf48b7f061432c98475eee9
```

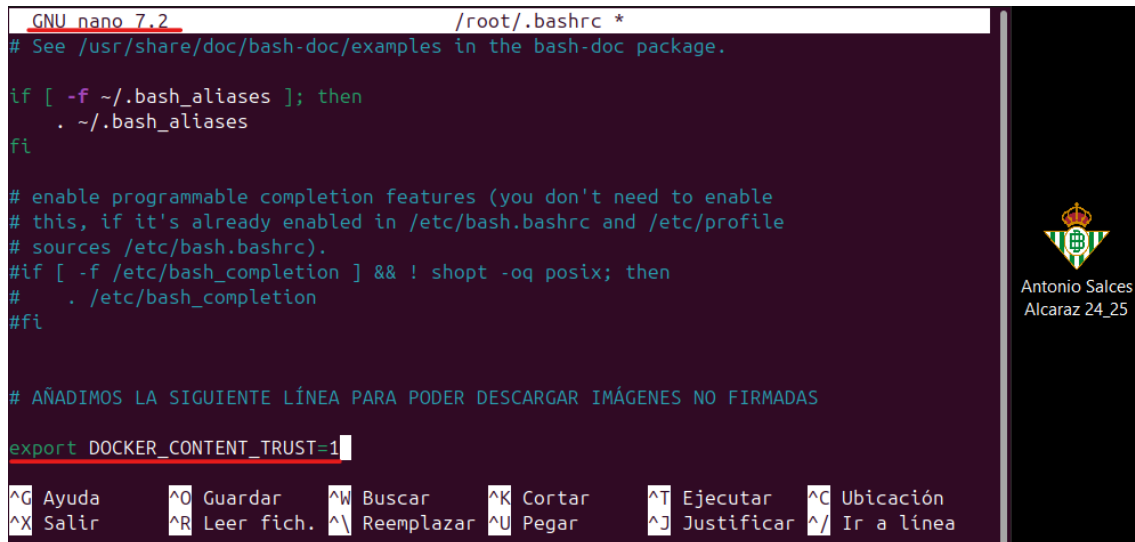


Papelera de
reciclaje

7.2. Cambiar la variable de entorno “DOCKER_CONTENT_TRUST”

Para descargar imágenes no firmadas, debemos de habilitar la variable de entorno “DOCKER_CONTENT_TRUST”.

Abriremos con el fichero /root/.bashrc con el comando “*nano ~/.bashrc*”, y añadimos al final la línea “export DOCKER_CONTENT_TRUST=1”.



```
GNU nano 7.2 /root/.bashrc *
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

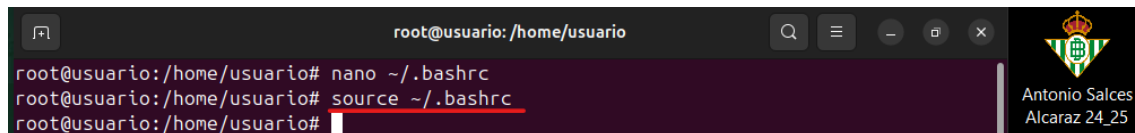
if [ -f ~/.bash_aliases ]; then
  . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
#if [ -f /etc/bash_completion ] && ! shopt -oq posix; then
#   . /etc/bash_completion
#fi

# AÑADIMOS LA SIGUIENTE LÍNEA PARA PODER DESCARGAR IMÁGENES NO FIRMADAS
export DOCKER_CONTENT_TRUST=1
```

^G Ayuda ^O Guardar ^W Buscar ^K Cortar ^T Ejecutar ^C Ubicación
^X Salir ^R Leer fich. ^\ Reemplazar ^U Pegar ^J Justificar ^_ Ir a línea

Ahora tenemos que recargar el archivo, que debemos de hacerlo con “*source ~/.bashrc*”.




```
root@usuario: /home/usuario# nano ~/.bashrc
root@usuario: /home/usuario# source ~/.bashrc
root@usuario: /home/usuario#
```


7.3. Intentar descargar imagen no firmada

Tras añadir la línea en el punto anterior, volvemos a intentar descargar la imagen no firmada.

```
root@usuario:/home/usuario# docker pull testinf/holamundohtml
Using default tag: latest
Error: remote trust data does not exist for docker.io/testinf/holamundohtml: notary.docker.io does not have trust data for docker.io/testinf/holamundohtml
root@usuario:/home/usuario# docker pull --disable-content-trust testinf/holamundohtml
Using default tag: latest
latest: Pulling from testinf/holamundohtml
6ec7b7d162b2: Pull complete
db606474d60c: Pull complete
afb30f0cd8e0: Pull complete
3bb2e8051594: Pull complete
4c761b44e2cc: Pull complete
c2199db96575: Pull complete
1b9a9381eea8: Pull complete
50f0689715a3: Pull complete
8a6cc018dd45: Pull complete
052299cf2d76: Pull complete
ee83da709c88: Pull complete
5b10df91e6d0: Pull complete
a2eb858e27d8: Pull complete
ff8ee7220ee5: Pull complete
Digest: sha256:02f123b02628b142844fa5537024909fcfb3ae3fc6dbfc3ce1d78fd3dc7aa0ee
Status: Downloaded newer image for testinf/holamundohtml:latest
docker.io/testinf/holamundohtml:latest
root@usuario:/home/usuario#
```



Antonio Salces
Alcaraz 24_25



Al intentar hacer el primer “pull” (sin ningún parámetro), podemos ver que no podemos descargar la imagen. Sin embargo, cuando añadimos el parámetro “--disable-content-trust” (“**docker pull --disable-content-trust <imagen>**”), vemos que podemos descargar la imagen sin problema.

8. BIBLIOGRAFIA

Coronado, G. (2025). Resumen de Comandos Básicos de Docker. Campanillas (Málaga).

Docker. (1 de febrero de 2020). Obtenido de <https://www.docker.com/101-tutorial/>

OpenAI. (diciembre de 2025). *ChatGPT*. Obtenido de ChatGPT:

<https://chatgpt.com/?model=auto>