

docker

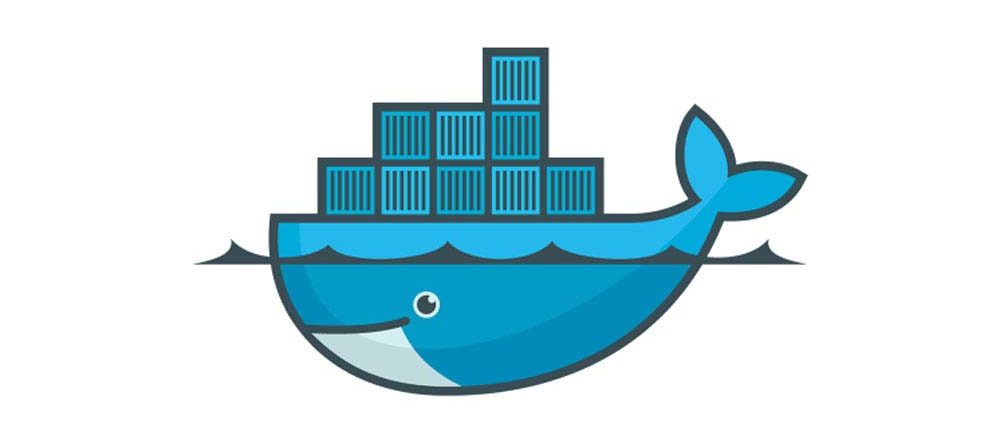
Sistemas Informáticos (1º DAM)



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](#_Toc187964435)

[1.1. Descargar imágenes de Docker Hub. 1](#_Toc187964436)

[1.2. Mostrar imágenes 3](#_Toc187964437)

[1.3. Contenedor con Ubuntu 18.04, nombre “ubuntu” 4](#_Toc187964438)

[1.3.1. Parar contenedor 4](#_Toc187964439)

[1.3.2. Rearrancar contenedor 4](#_Toc187964440)

[1.3.3. Mostrar fichero sin entrar al contenedor 5](#_Toc187964441)

[2. Actividad 2: trabajar con PHP y MariaDB 6](#_Toc187964442)

[2.1. Contenedor con PHP, nombre “web” y puerto 8181 6](#_Toc187964443)

[2.2. Colocar ficheros index.html e index.php en el directorio web del contenedor 6](#_Toc187964444)

[2.3. Contenedor con mariadb con diferentes parámetros 7](#_Toc187964445)

[2.4. Conectar con un cliente de base de datos 7](#_Toc187964446)

[3. Actividad 3: gestión de imágenes 9](#_Toc187964447)

[3.1. Descargar imagen Ubuntu:20.04 9](#_Toc187964448)

[3.2. Volcar la información de la nueva imagen en un archivo 9](#_Toc187964449)

[3.3. Crear contenedor con Ubuntu:20.04 llamada “modulo3” 10](#_Toc187964450)

[3.4. Borrar contenedor con Ubuntu:20.04 10](#_Toc187964451)

[3.5. Hacer acciones necesarias para borrar el contenedor Ubuntu:20.04 10](#_Toc187964452)

[4. Actividad 4: operaciones con volúmenes 11](#_Toc187964453)

[4.1. Crear volúmenes 11](#_Toc187964454)

[4.2. Instanciar contenedores en los volúmenes 11](#_Toc187964455)

[4.3. Parar y borrar contenedor “c2” y volumen “volumen\_datos” 12](#_Toc187964456)

[4.4. Comprobar que “c1” está montado sobre “volumen\_web” 12](#_Toc187964457)

[5. Actividad 5: trabajo con redes 13](#_Toc187964458)

[5.1. Creación de redes 13](#_Toc187964459)

[5.2. Configuración de ambas redes 13](#_Toc187964460)

[5.3. Arrancar contenedor Ubuntu:20.04 con configuración específica 14](#_Toc187964461)

[5.4. Arrancar otro contenedor Ubuntu:20.04 con otra configuración 15](#_Toc187964462)

[5.5. Configuración de red de ambos contenedores 16](#_Toc187964463)

[5.6. Comprobar conectividad entre contenedores 17](#_Toc187964464)

[5.7. Conectar el contenedor “u1” a “red2” 18](#_Toc187964465)

[6. Actividad 5: editar fichero de configuración .yml 19](#_Toc187964466)

[6.1. Descargar archivo de configuración .yml para docker-compose 19](#_Toc187964467)

[6.2. Editar el archivo .yml y cambiar puerto 19](#_Toc187964468)

[6.3. Arrancar con “docker-compose” 19](#_Toc187964469)

[6.4. Entrar en el contenedor mediante el navegador 20](#_Toc187964470)

[7. Operaciones con imágenes no firmadas 21](#_Toc187964471)

[7.1. Descargar imagen no firmada 21](#_Toc187964472)

[7.2. Cambiar la variable de entorno “DOCKER\_CONTENT\_TRUST” 22](#_Toc187964473)

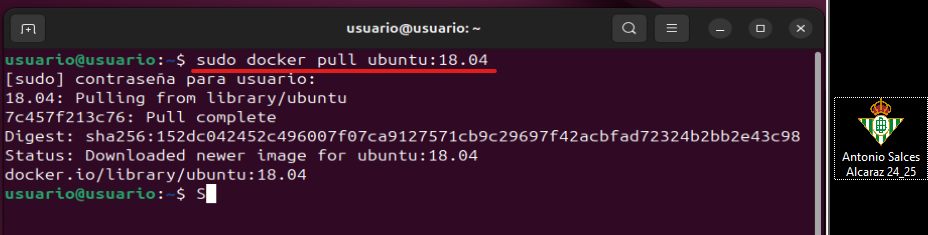
[7.3. Intentar descargar imagen no firmada 23](#_Toc187964474)

# Actividad 1: descargar imágenes y operaciones básicas

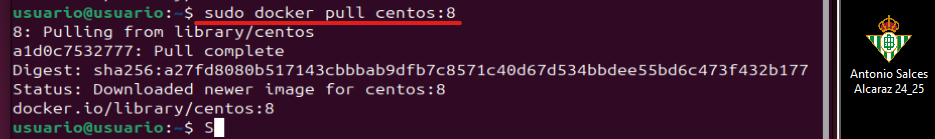
## Descargar imágenes de Docker Hub.

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

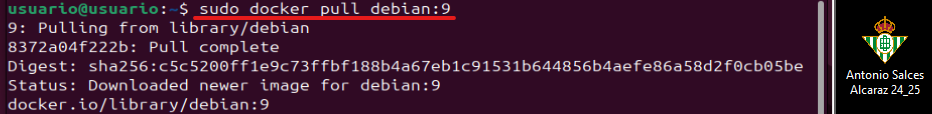
Ubuntu:18.04



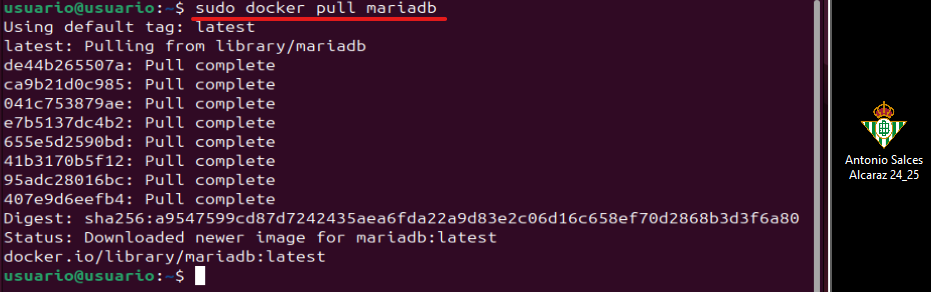
Centos:8



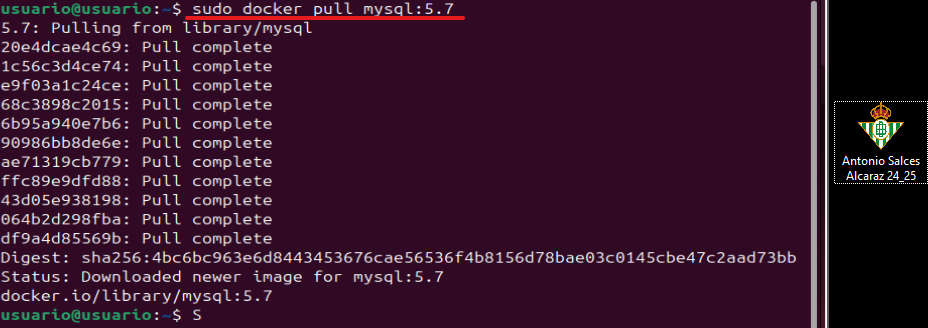
Debian: 9



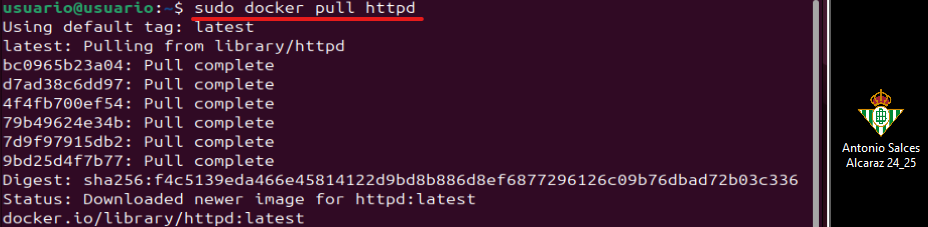
MariaDB:latest



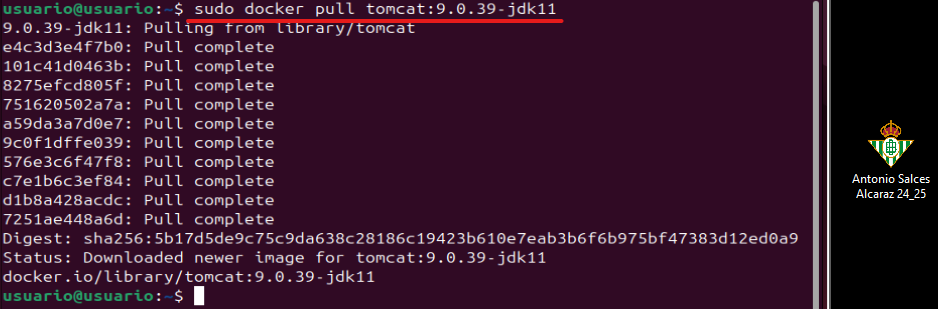
MySQL:5.7



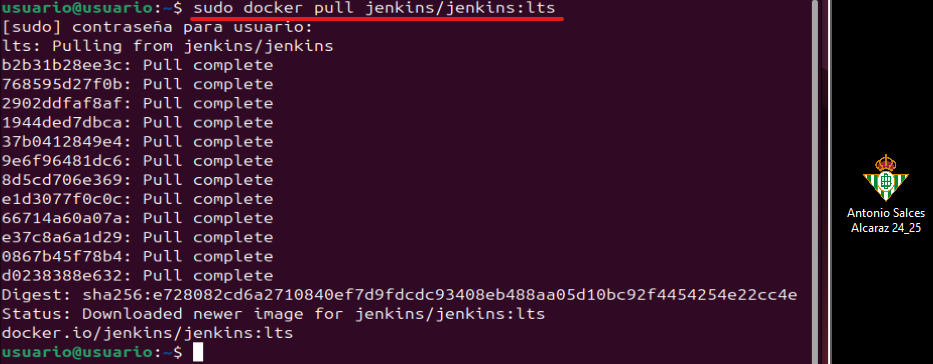
httpd



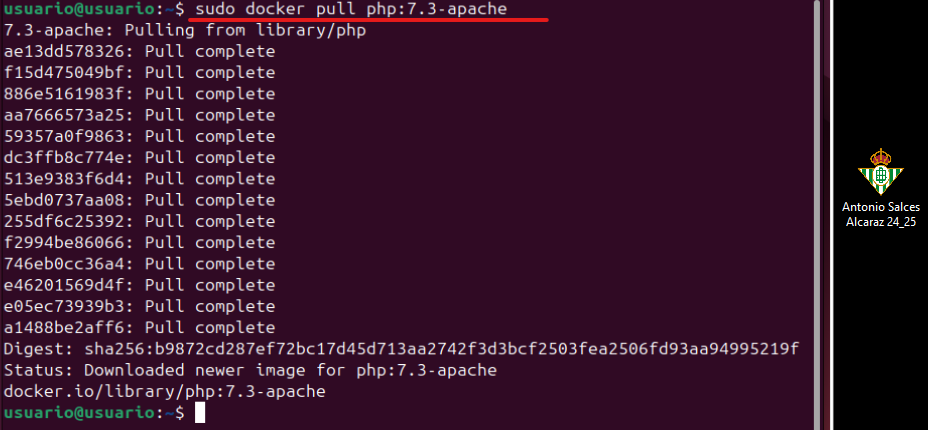
tomcat:9.0.39-jdk11



jenkins/jenkins:lts



php:7.3-apache



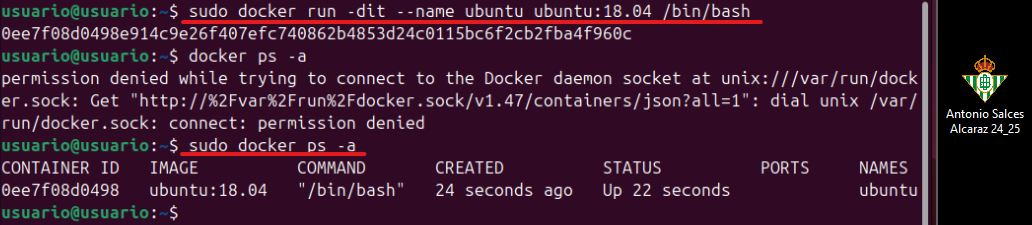
## Mostrar imágenes

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



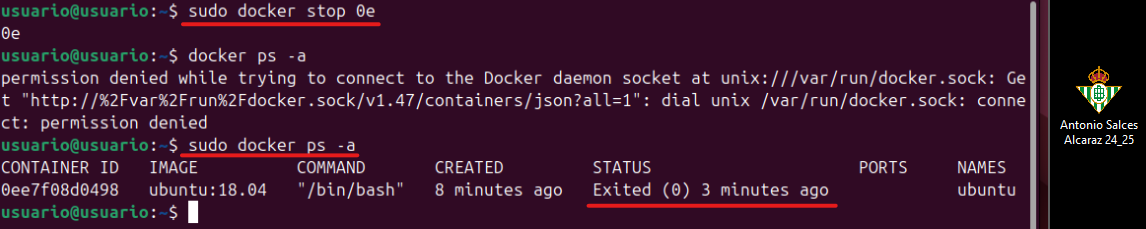
## Contenedor con Ubuntu 18.04, nombre “ubuntu”

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 asi poder saber el ID del docker de ubuntu.



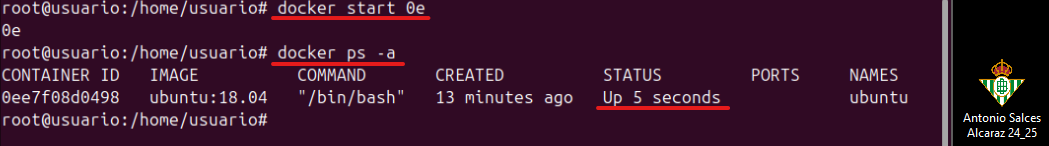
### Parar contenedor

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.



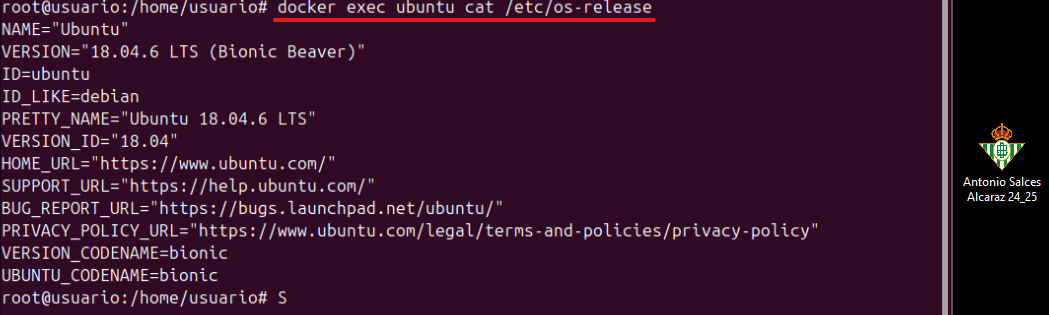
### Rearrancar contenedor

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



### Mostrar fichero sin entrar al contenedor

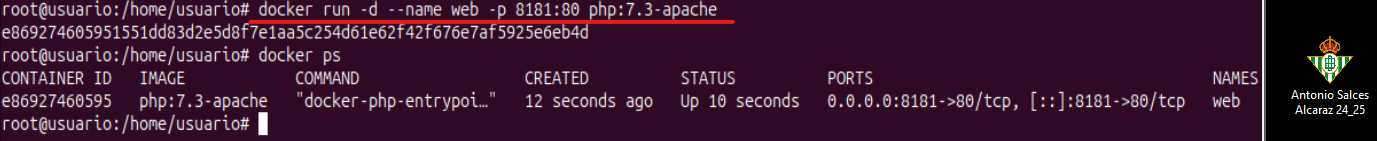
Para ello, debemos utilizar el comando “***docker exec <nombre> cat /etc/os-release***”. Docker exec nos permite ejecutar comandos del contenedor desde fuera del mismo, mientras que “cat” se utiliza para visualizar archivos. En lugar del nombre, también podemos poner el ID.



# Actividad 2: trabajar con PHP y MariaDB

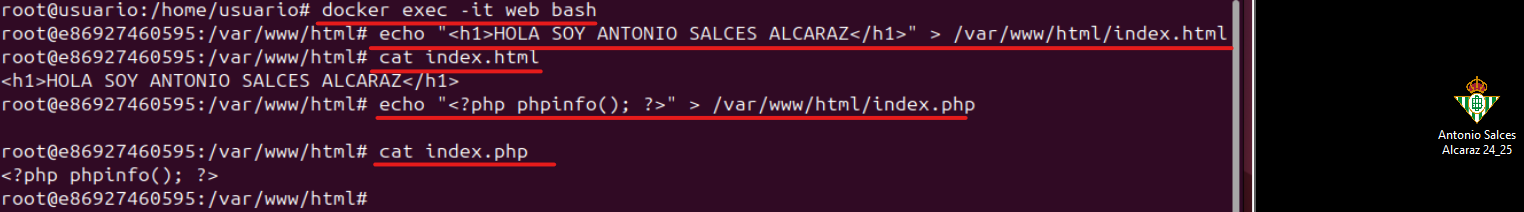
## Contenedor con PHP, nombre “web” y puerto 8181

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.



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

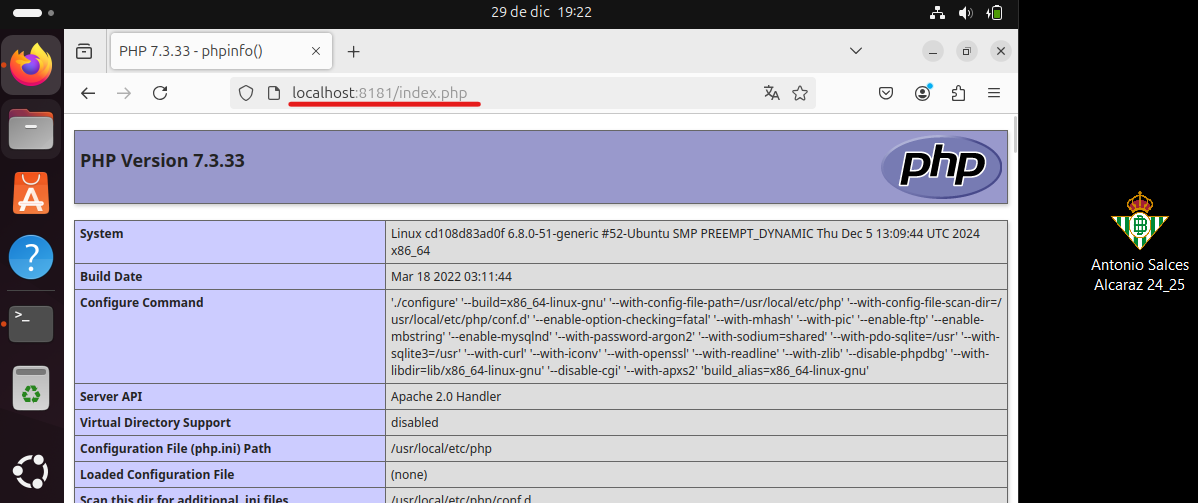
En primer lugar, accederemos al directorio web del contenedor con el primer comando que se ve en la imagen (“***docker exec -it [nombre o ID] bash***”). Una vez estemos en el contenedor, utilizaremos “***echo <contenido del archivo> > <ubicación del archivo>***”. Para comprobar el contenido del archivo,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



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



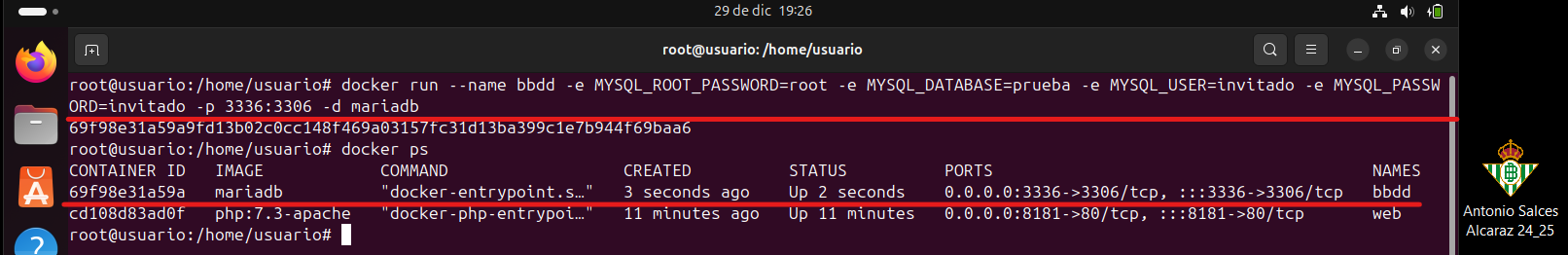
Y esto saldrá si buscamos el fichero index.php.



## Contenedor con mariadb con diferentes parámetros

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

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

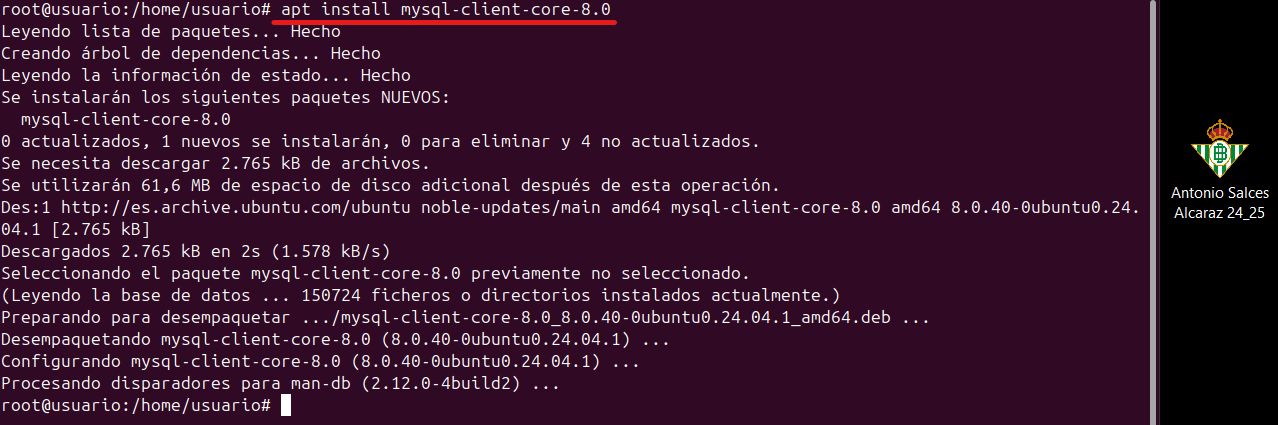


Los parámetros que se han utilizado son los siguientes:

* --name bbdd: el nombre del docker es “bbdd”.
* -e MYSQL\_ROOT\_PASSWORD=root: la constraseña 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.

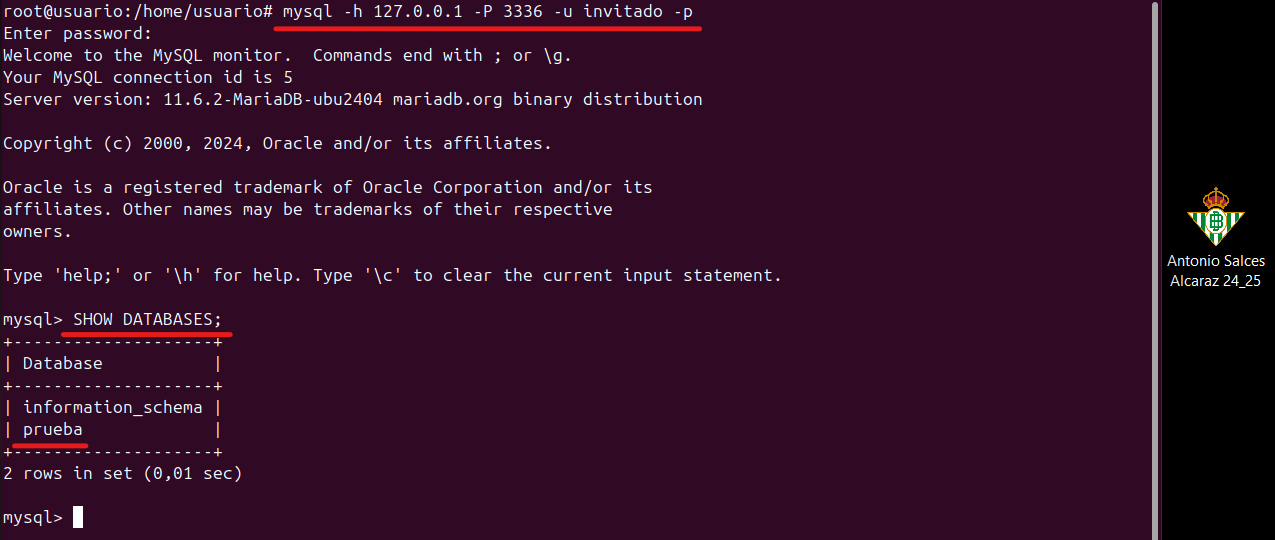
## Conectar con un cliente de base de datos

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>***”.



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.

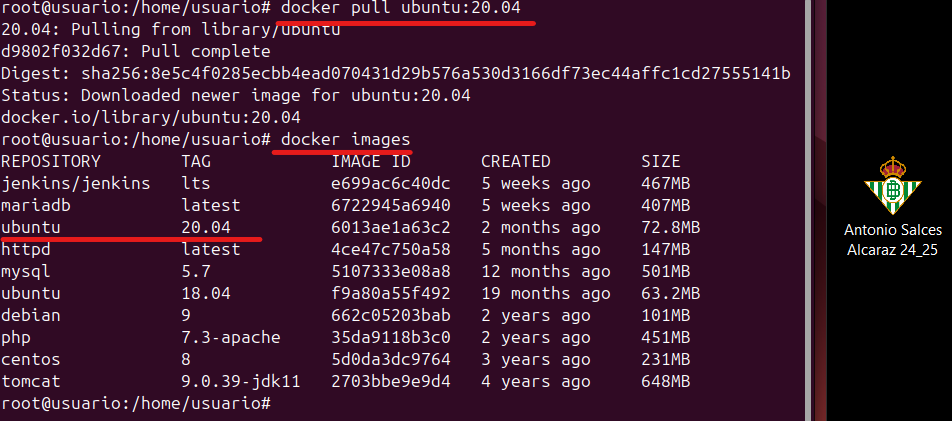
Una vez hayamos conectado con la base de datos, podemos utilizar el comando “***SHOW DATABASES;***” para ver las bases de datos creadas. Podemos ver en la imagen que la base de datos “prueba” se ha creado.



# Actividad 3: gestión de imágenes

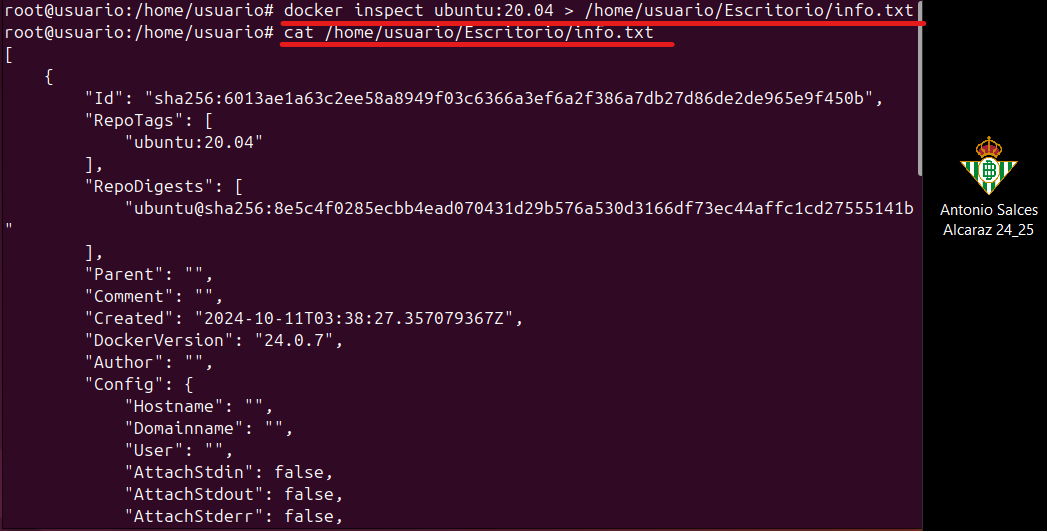
## Descargar imagen Ubuntu:20.04

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*”.**



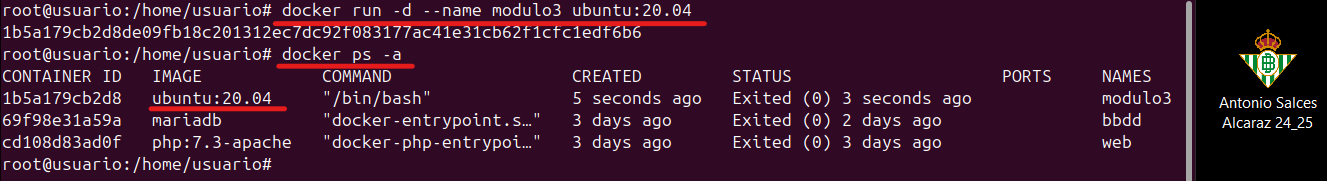
## Volcar la información de la nueva imagen en un archivo

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>***”.



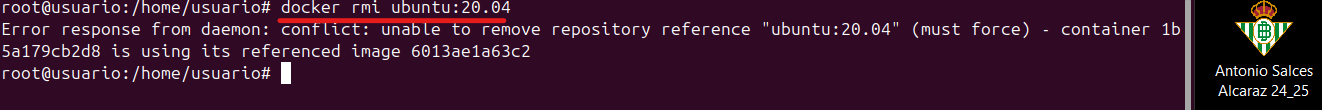
## Crear contenedor con Ubuntu:20.04 llamada “modulo3”

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.



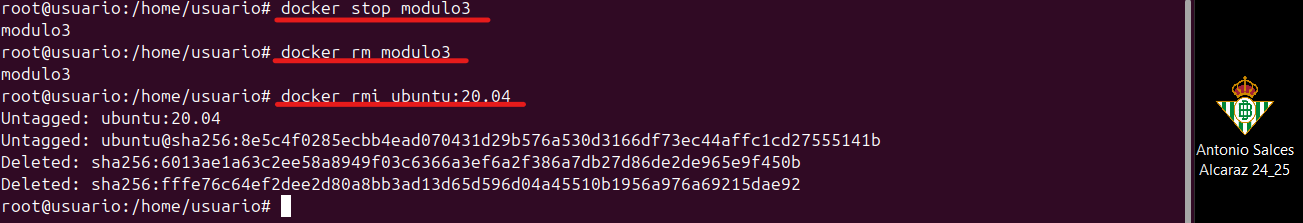
## Borrar contenedor con Ubuntu:20.04

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.



## Hacer acciones necesarias para borrar el contenedor Ubuntu:20.04

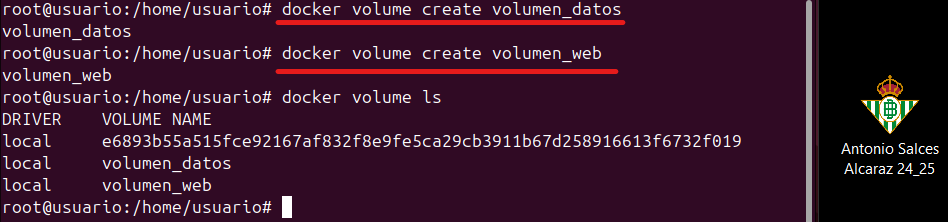
Para borrar la imagen, en primer lugar debemos parar el contenedor con “***docker stop <contenedor>***”, luego borrar con “***docker rm <contenedor>***”, y por ultimo borrar la imagen con “***docker rmi <imagen>***”.



# Actividad 4: operaciones con volúmenes

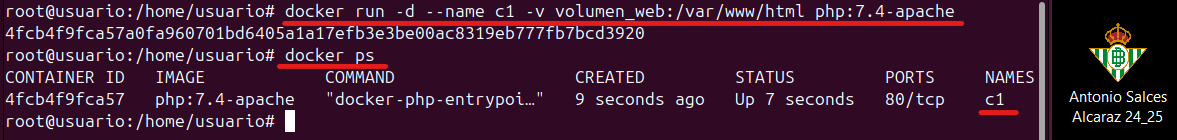
## Crear volúmenes

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.

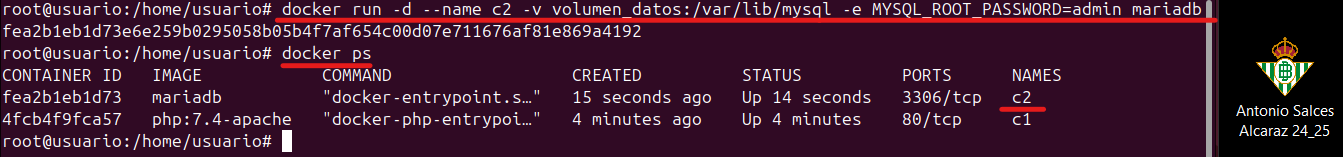


## Instanciar contenedores en los volúmenes

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

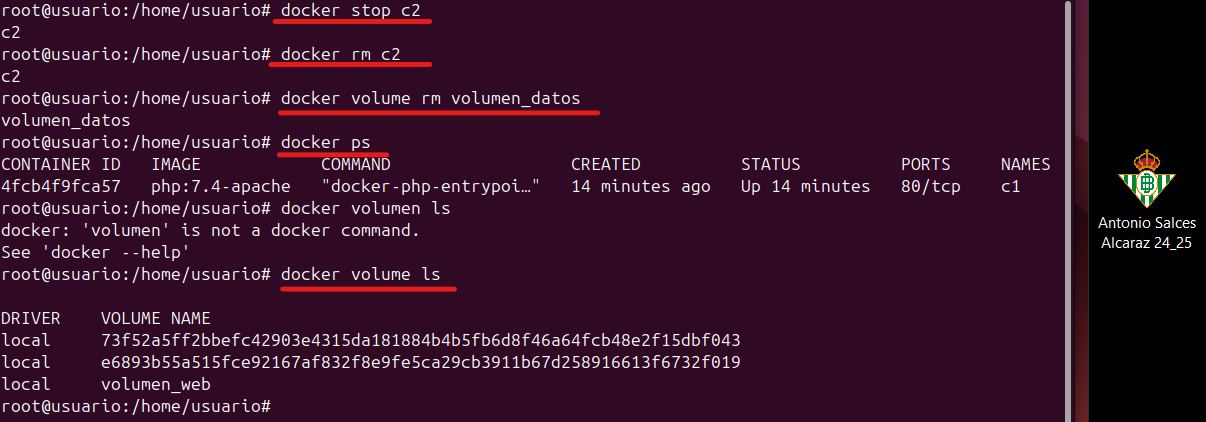


Con el comando “***docker run [-d] [--name <nombre>] >] [-v <volumen>:][ruta] [-e<MYSQL\_ROOT\_PASSWORD>=<contraseña>] <imagen>***” podremos crear un contenedor, elegir el volumen y la ruta donde se aloja y, en este caso (se trata de MariaDB) podemos asignar también una contraseña al usuario “root”.



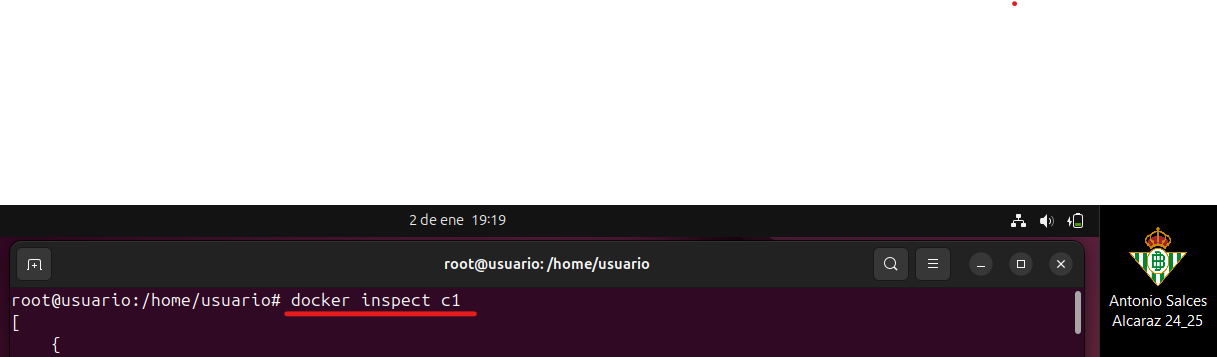
## Parar y borrar contenedor “c2” y volumen “volumen\_datos”

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”.

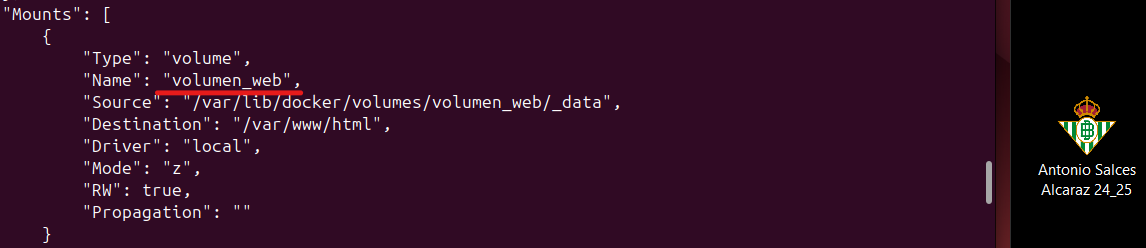


## Comprobar que “c1” está montado sobre “volumen\_web”

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



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



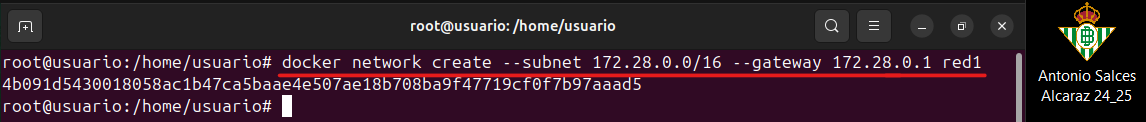
# Actividad 5: trabajo con redes

## Creación de redes

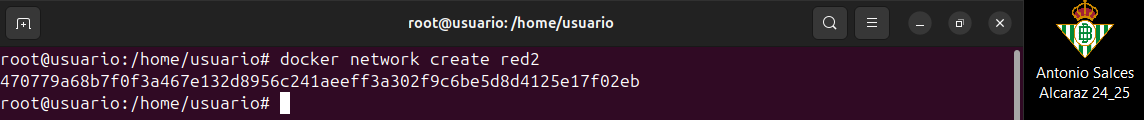
En primer lugar, debemos de crear una red con los siguientes requisitos:

* **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>***”.

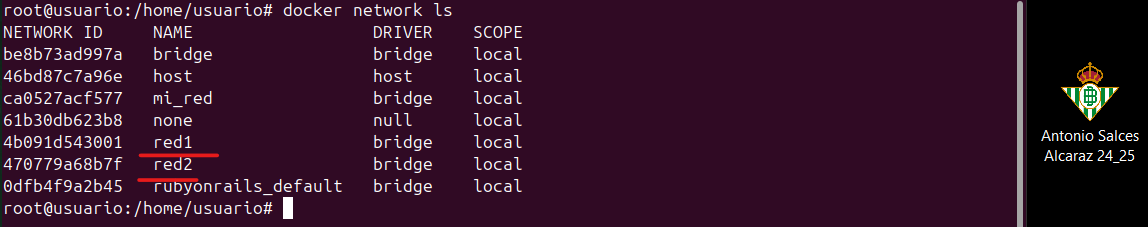


Ahora crearemos la segunda red, la cual solo tendrá configurada el nombre.

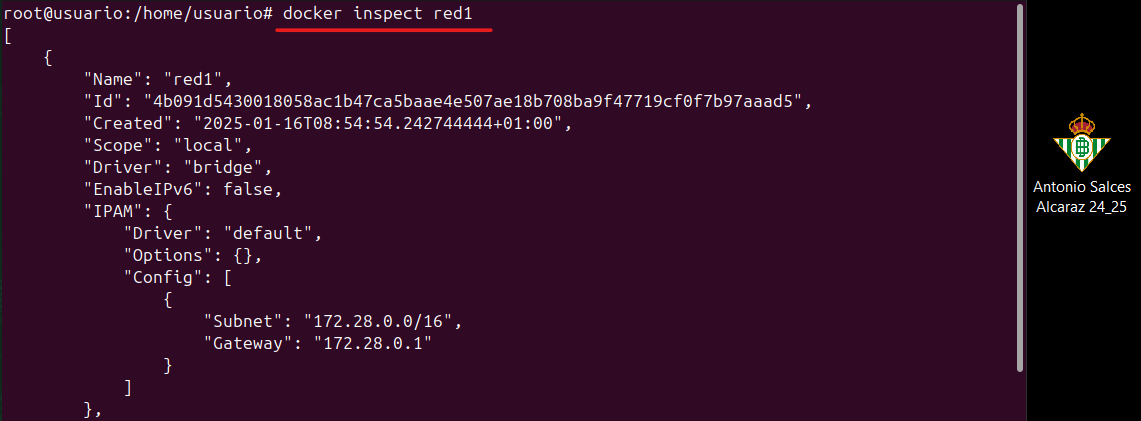


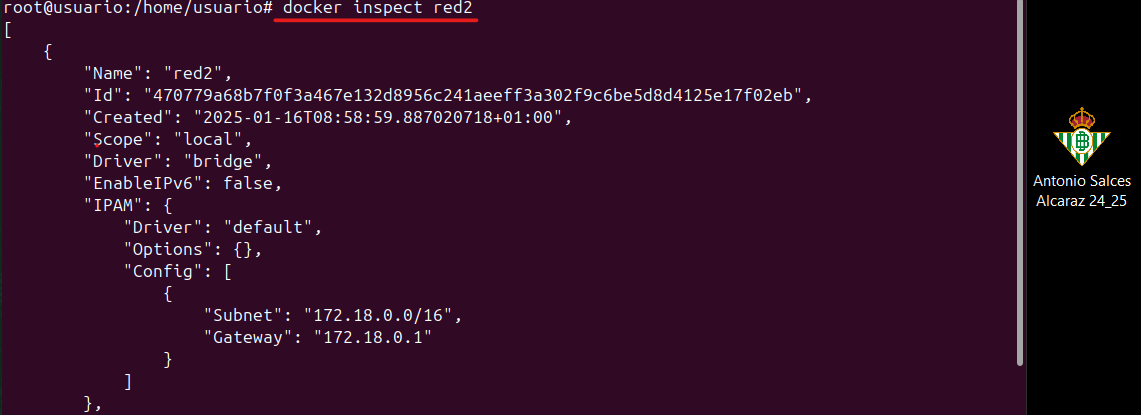
## Configuración de ambas redes

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



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



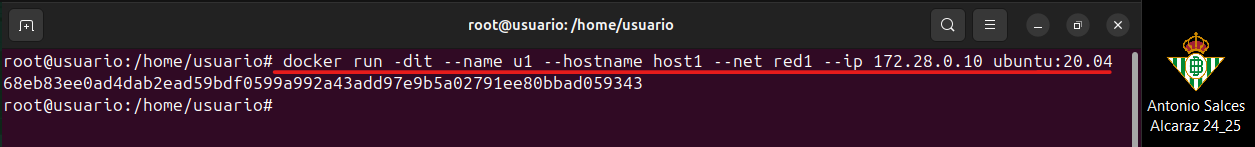


## Arrancar contenedor Ubuntu:20.04 con configuración específica

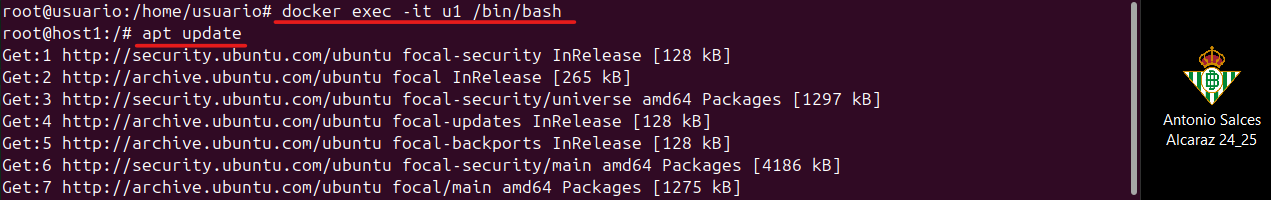
Debemos arrancar un contenedor con la siguiente configuración:

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

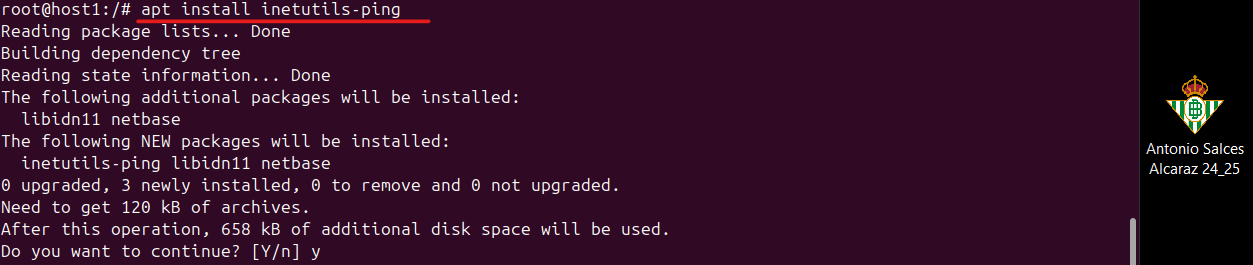
Para ello, utilizaremos el comando “***docker run [-dit] [--name <nombre>] [--hostname <nombre>] [--net <red>] [--ip <IP>] <imagen>***”.



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***”.



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

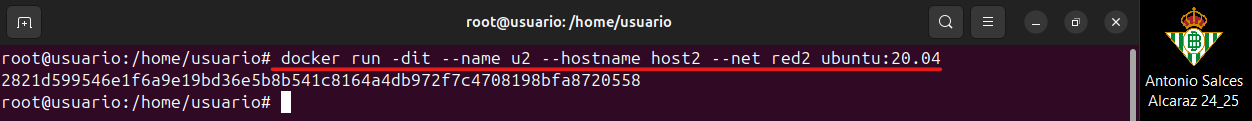


## Arrancar otro contenedor Ubuntu:20.04 con otra configuración

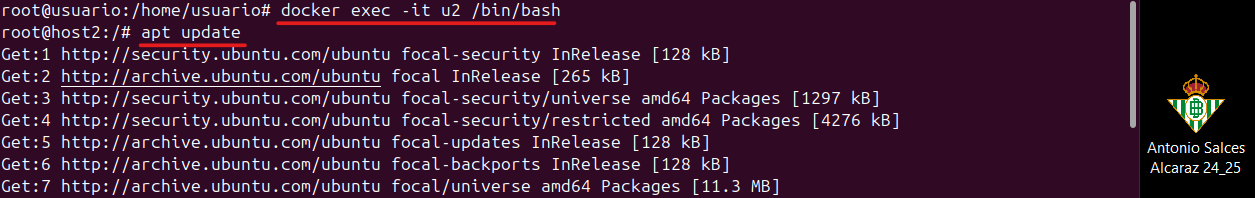
Debemos arrancar otro contenedor con Ubuntu:20.04 con la siguiente configuración:

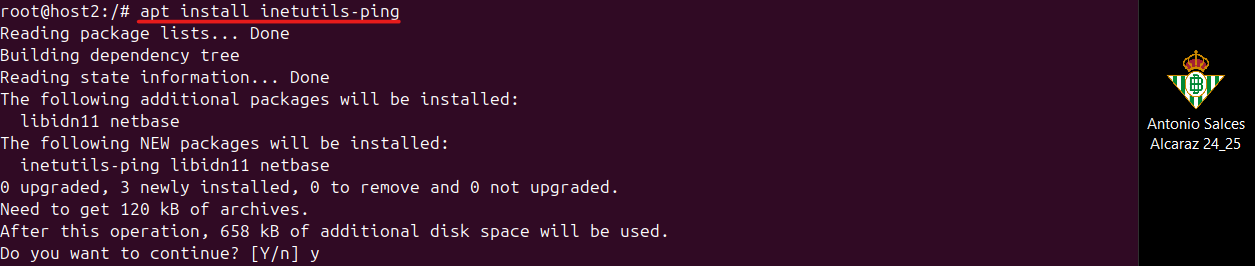
* **Nombre**: u2
* **Hostname**: host2
* **Red**: red2
* **IP**: automática
* **Imagen**: Ubuntu:20.04

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



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



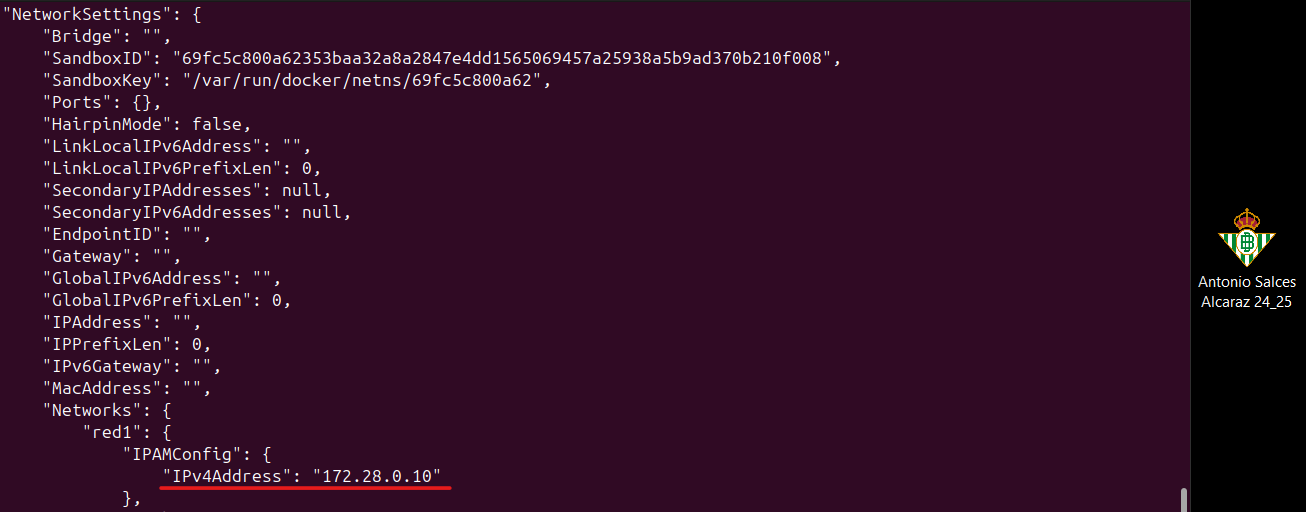


## Configuración de red de ambos contenedores

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

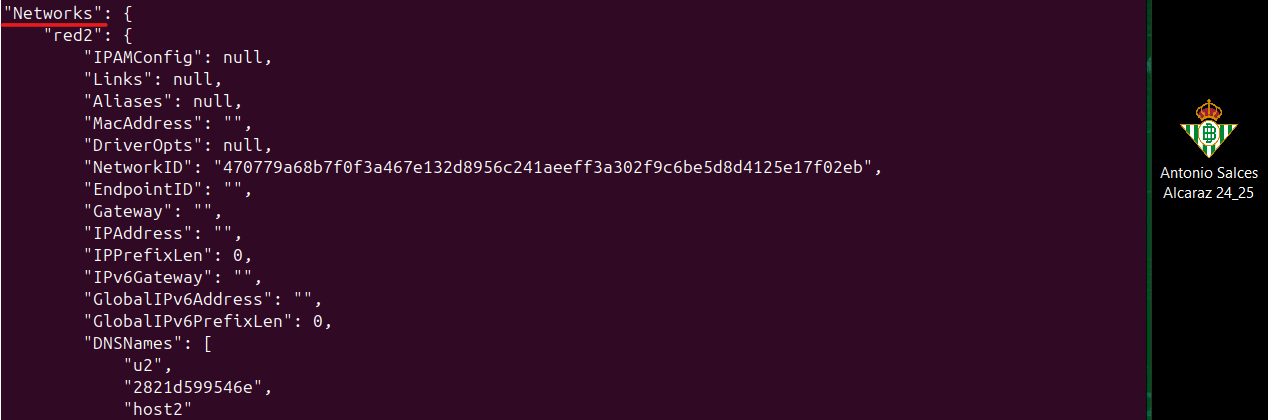
En las siguientes capturas podemos ver en que red y la IP del contenedor “u1”.





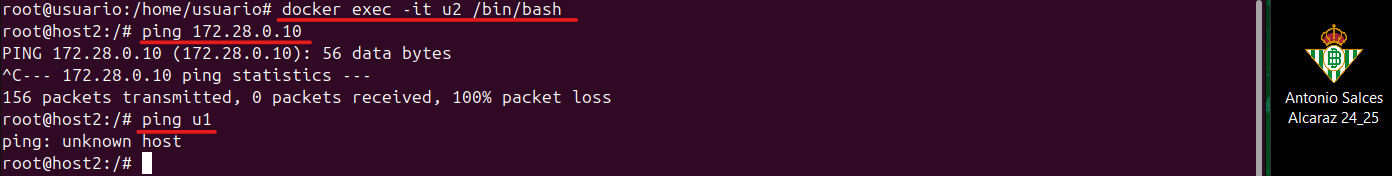
Ahora, en las siguientes dos capturas, se mostrará la configuración de red del contenedor “u2”.





## Comprobar conectividad entre contenedores

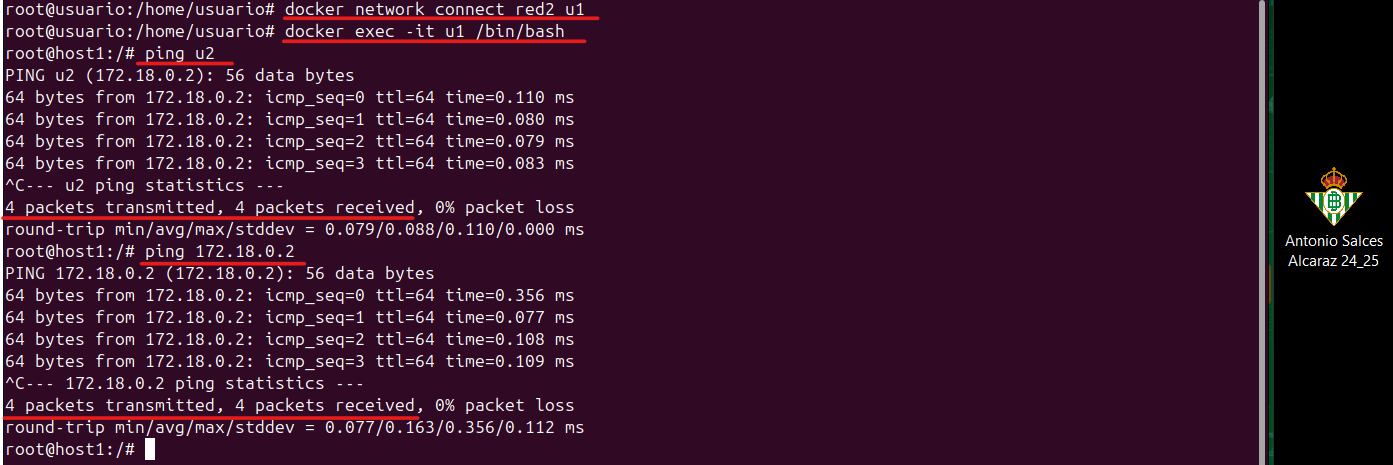
En primer lugar, nos debemos de meter en cualquier contenedor (en mi caso “u2”) con el comando “***docker exec -it <nombre o ID> /bin/bash***”. Una vez estemos en el contenedor, ejecutaremos el comando “***ping <nombre o ID***” (ejecutaremos ambos), para comprobar si ambos contenedores tienen conectividad.



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

## 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.

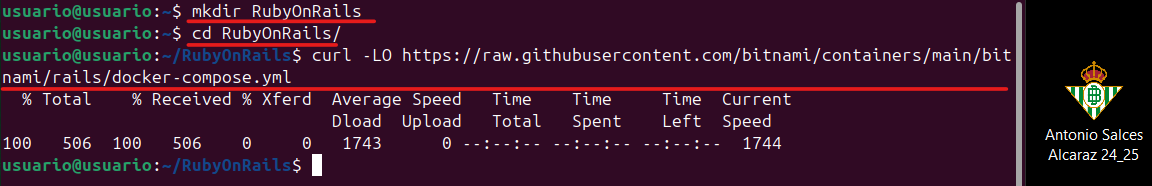


Como vemos en la imagen anterior, ambos contenedores tienen conexión entre ellos, como podemos ver en el mensaje traducido al español “4 paquetes trasmitidos, 4 paquetes recibidos”.

# Actividad 5: editar fichero de configuración .yml

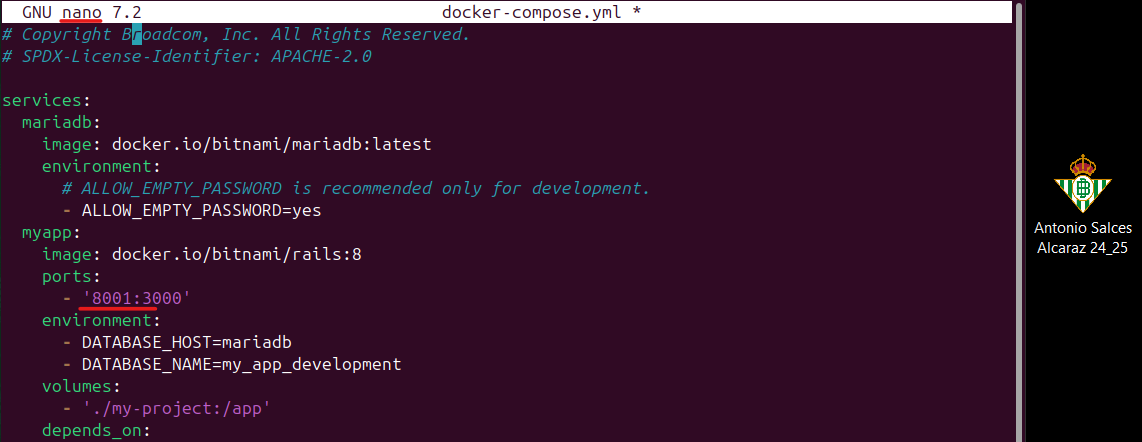
## Descargar archivo de configuración .yml para docker-compose

En primer lugar debemos de crear una carpeta donde guardaremos el archivo .yml 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*.



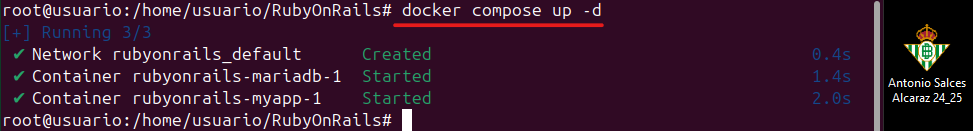
## Editar el archivo .yml y cambiar puerto

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”.



## Arrancar con “docker-compose”

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

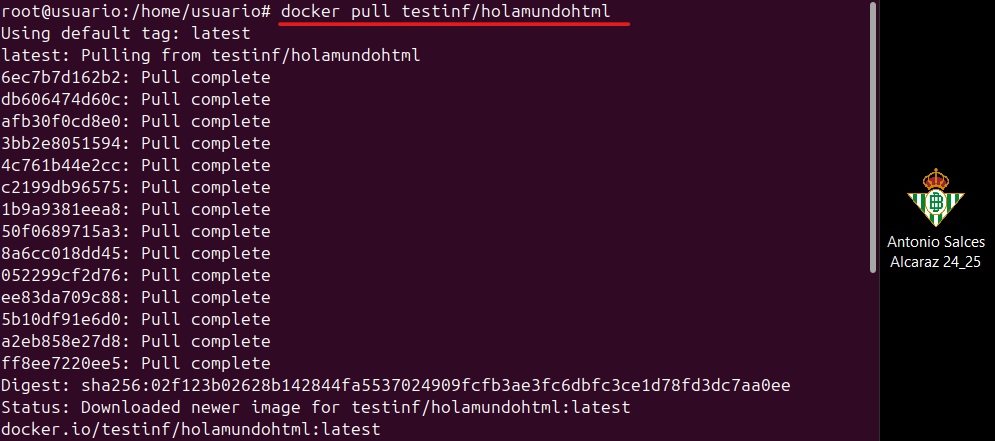


## Entrar en el contenedor mediante el navegador

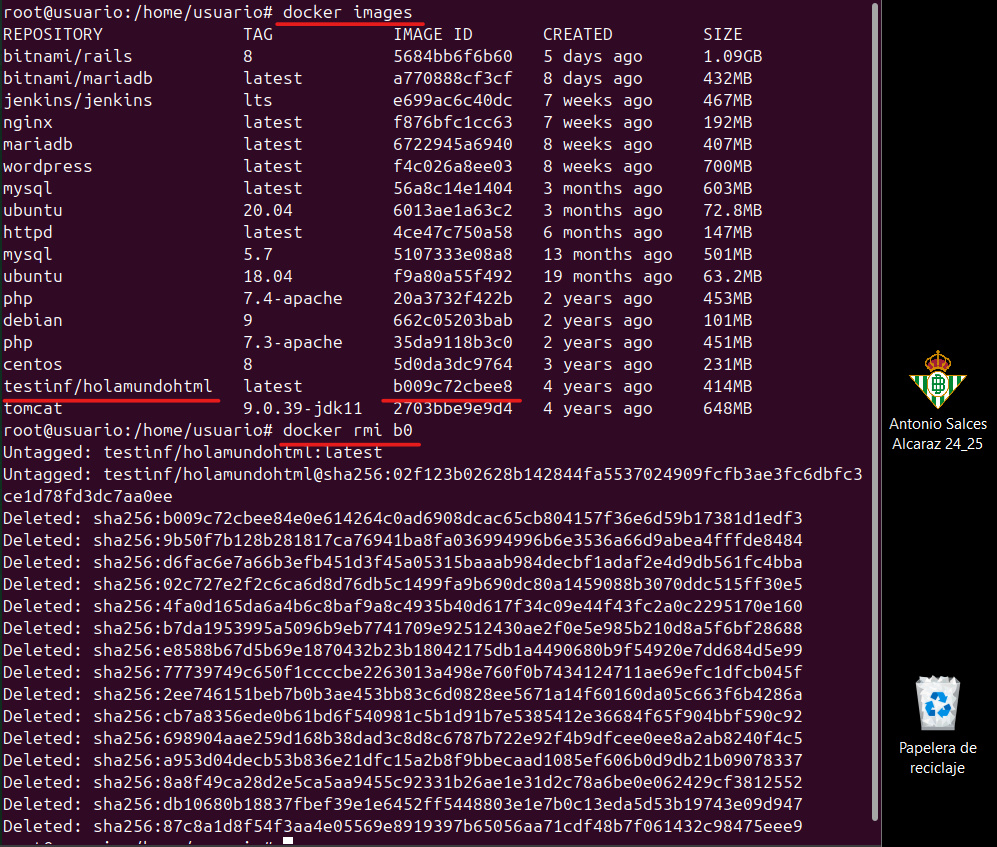
# Operaciones con imágenes no firmadas

## Descargar imagen no firmada

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



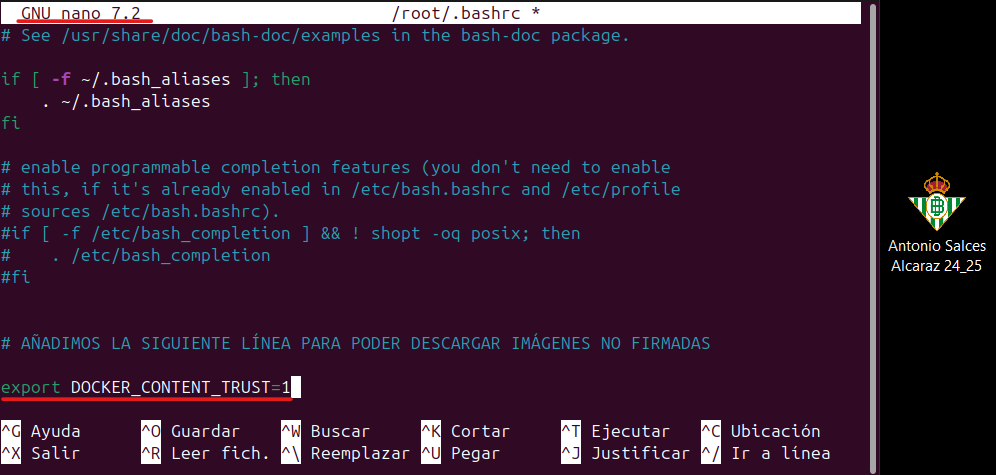
Ahora tenemos que eliminar la imagen. Para ello, primero buscaré el ID de la imagen con “***docker images***”, y luego utilizaré “***docker rmi <ID>***”.



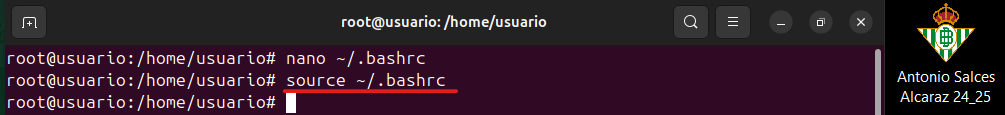
## Cambiar la variable de entorno “DOCKER\_CONTENT\_TRUST”

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”.

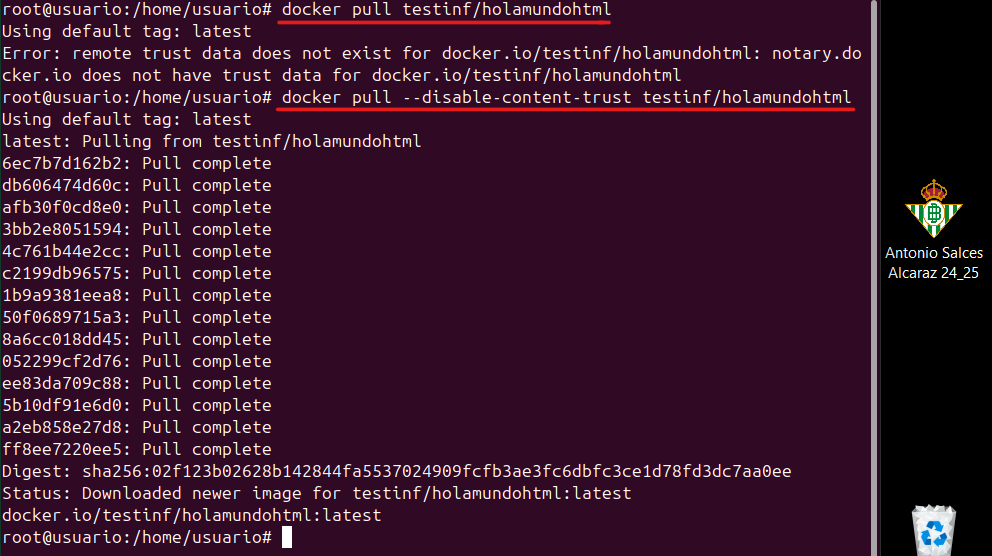


Ahora tenemos que recargar el archivo, que debemos de hacerlo con “***source ~/.bashrc***”.



## Intentar descargar imagen no firmada

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



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.