

docker

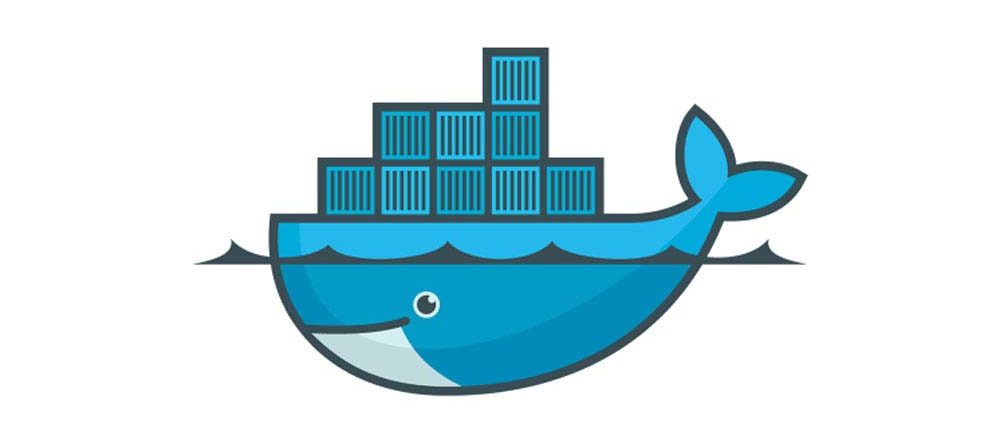
Sistemas Informáticos (1º DAM)



[Fecha]

[Nombre de la compañía]

[Dirección de la compañía]



Índice

Contenido

[1. Actividad 1: descargar imágenes y operaciones básicas 1](#_Toc187909460)

[1.1. Descargar imágenes de Docker Hub. 1](#_Toc187909461)

[1.2. Mostrar imágenes 3](#_Toc187909462)

[1.3. Contenedor con Ubuntu 18.04, nombre “ubuntu” 4](#_Toc187909463)

[1.3.1. Parar contenedor 4](#_Toc187909464)

[1.3.2. Rearrancar contenedor 4](#_Toc187909465)

[1.3.3. Mostrar fichero sin entrar al contenedor 5](#_Toc187909466)

[2. Actividad 2: trabajar con PHP y MariaDB 6](#_Toc187909467)

[2.1. Contenedor con PHP, nombre “web” y puerto 8181 6](#_Toc187909468)

[2.2. Colocar ficheros index.html e index.php en el directorio web del contenedor 6](#_Toc187909469)

[2.3. Contenedor con mariadb con diferentes parámetros 7](#_Toc187909470)

[2.4. Conectar con un cliente de base de datos 7](#_Toc187909471)

[3. Actividad 3: gestión de imágenes 9](#_Toc187909472)

[3.1. Descargar imagen Ubuntu:20.04 9](#_Toc187909473)

[3.2. Volcar la información de la nueva imagen en un archivo 9](#_Toc187909474)

[3.3. Crear contenedor con Ubuntu:20.04 llamada “modulo3” 10](#_Toc187909475)

[3.4. Borrar contenedor con Ubuntu:20.04 10](#_Toc187909476)

[3.5. Hacer acciones necesarias para borrar el contenedor Ubuntu:20.04 10](#_Toc187909477)

[4. Actividad 4: operaciones con volúmenes 11](#_Toc187909478)

[4.1. Crear volúmenes 11](#_Toc187909479)

[4.2. Instanciar contenedores en los volúmenes 11](#_Toc187909480)

[4.3. Parar y borrar contenedor “c2” y volumen “volumen\_datos” 12](#_Toc187909481)

[4.4. Comprobar que “c1” está montado sobre “volumen\_web” 12](#_Toc187909482)

[5. Actividad 5: editar fichero de configuración .yml 13](#_Toc187909483)

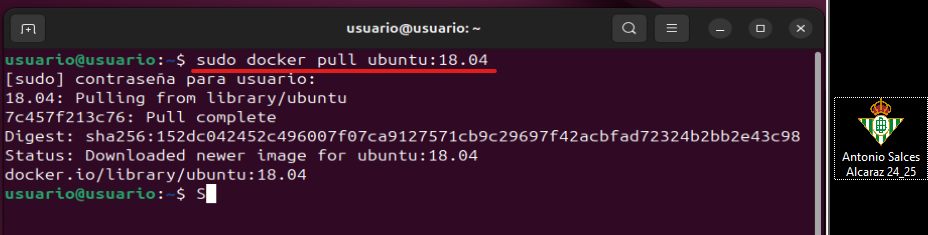
[5.1. Descargar archivo de configuración .yml para docker-compose 13](#_Toc187909484)

# Actividad 1: descargar imágenes y operaciones básicas

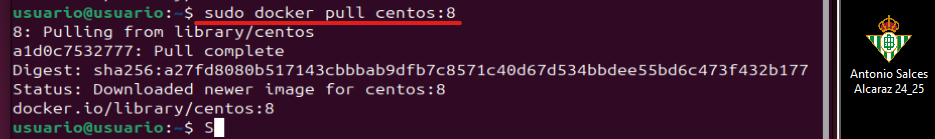
## Descargar imágenes de Docker Hub.

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

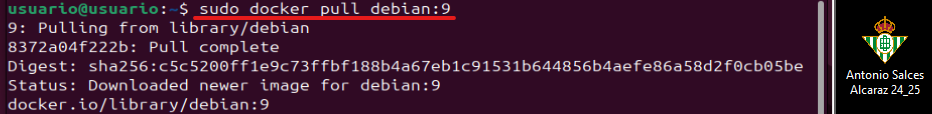
Ubuntu:18.04



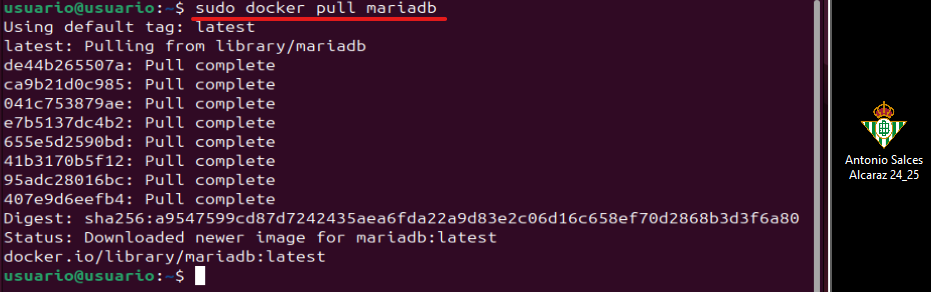
Centos:8



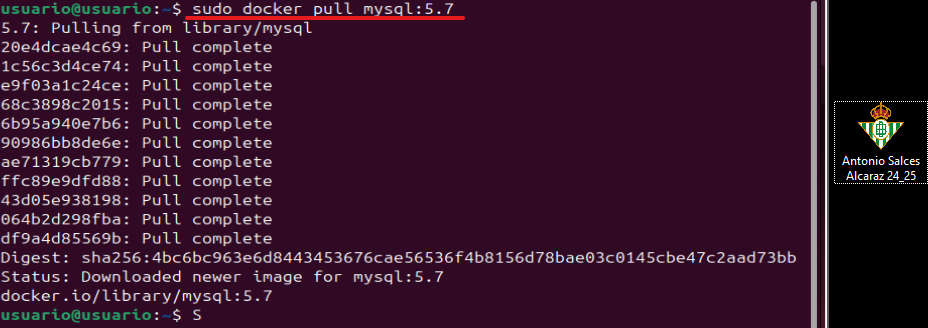
Debian: 9



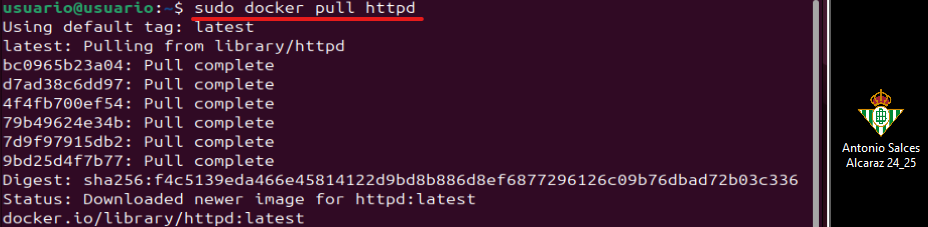
MariaDB:latest



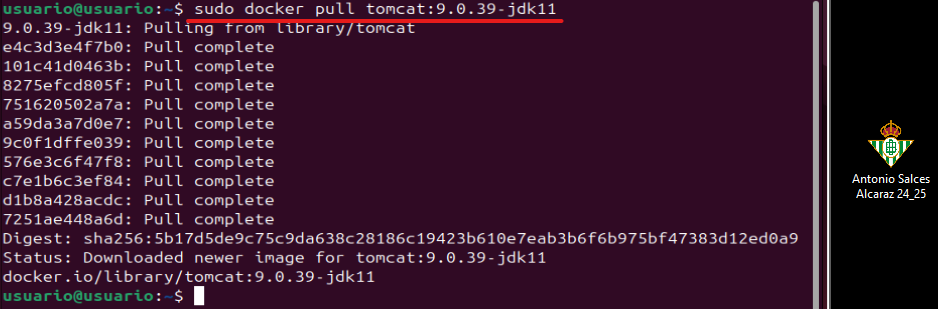
MySQL:5.7



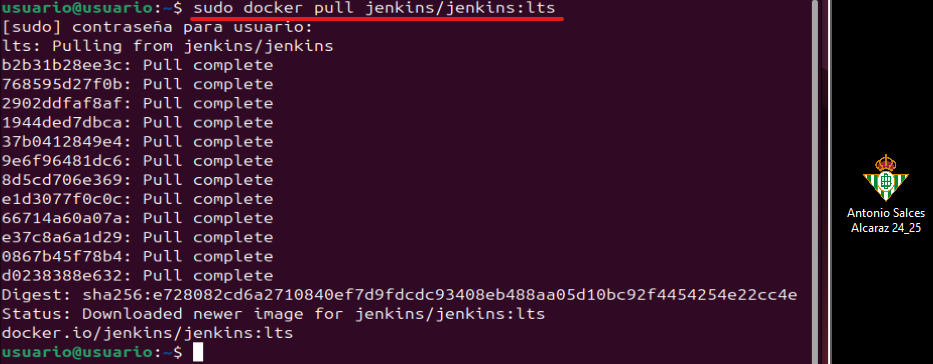
httpd



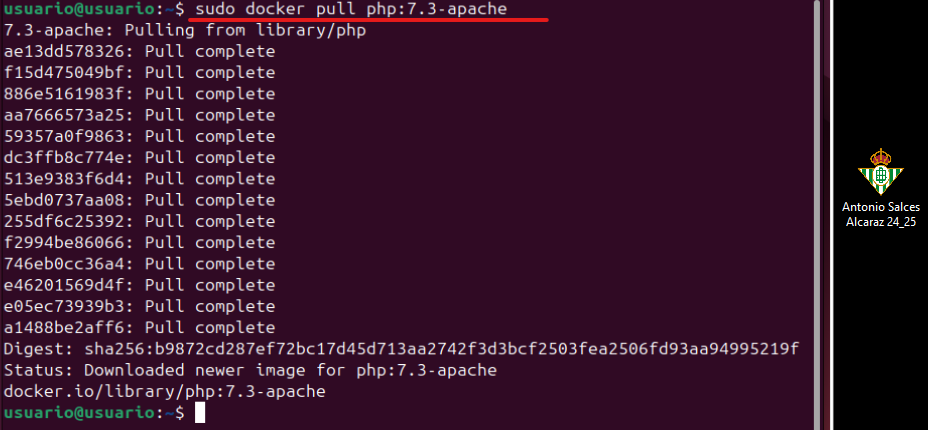
tomcat:9.0.39-jdk11



jenkins/jenkins:lts



php:7.3-apache



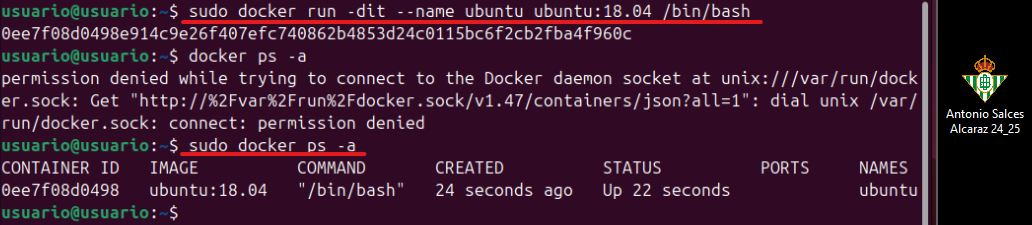
## Mostrar imágenes

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



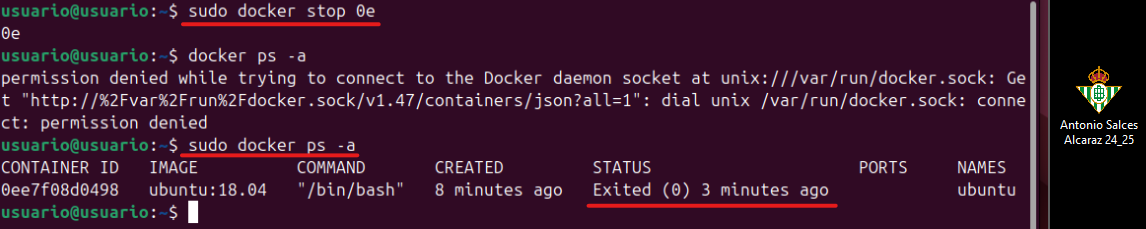
## Contenedor con Ubuntu 18.04, nombre “ubuntu”

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.



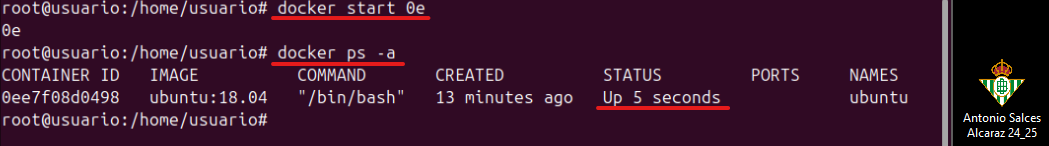
### Parar contenedor

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.



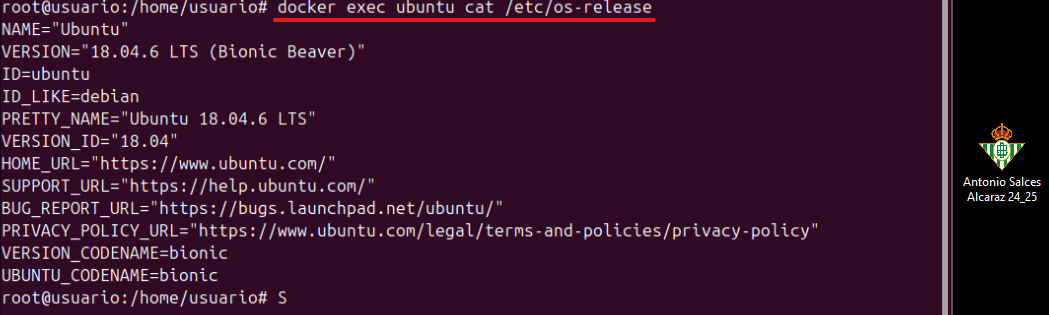
### Rearrancar contenedor

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



### Mostrar fichero sin entrar al contenedor

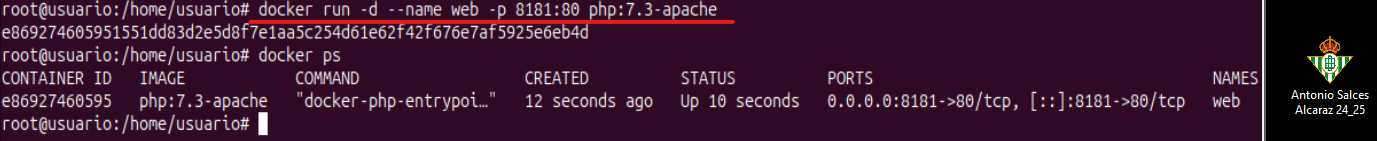
Para ello, debemos utilizar el comando “***docker exec <nombre> cat /etc/os-release***”. Docker exec nos permite ejecutar comandos del contenedor desde fuera del mismo, mientras que “cat” se utiliza para visualizar archivos. En lugar del nombre, también podemos poner el ID.



# Actividad 2: trabajar con PHP y MariaDB

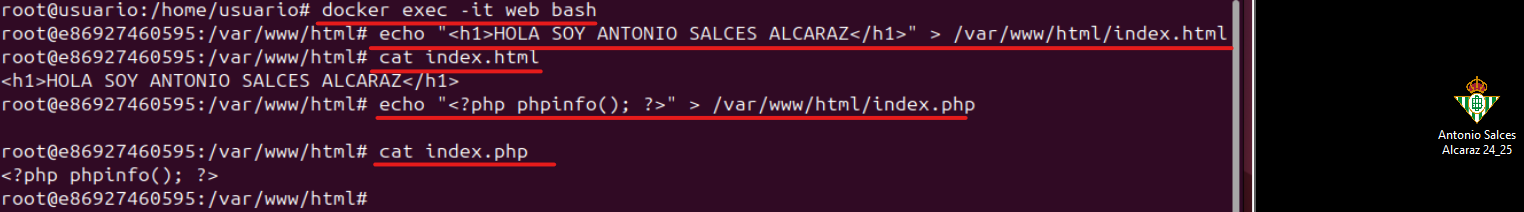
## Contenedor con PHP, nombre “web” y puerto 8181

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

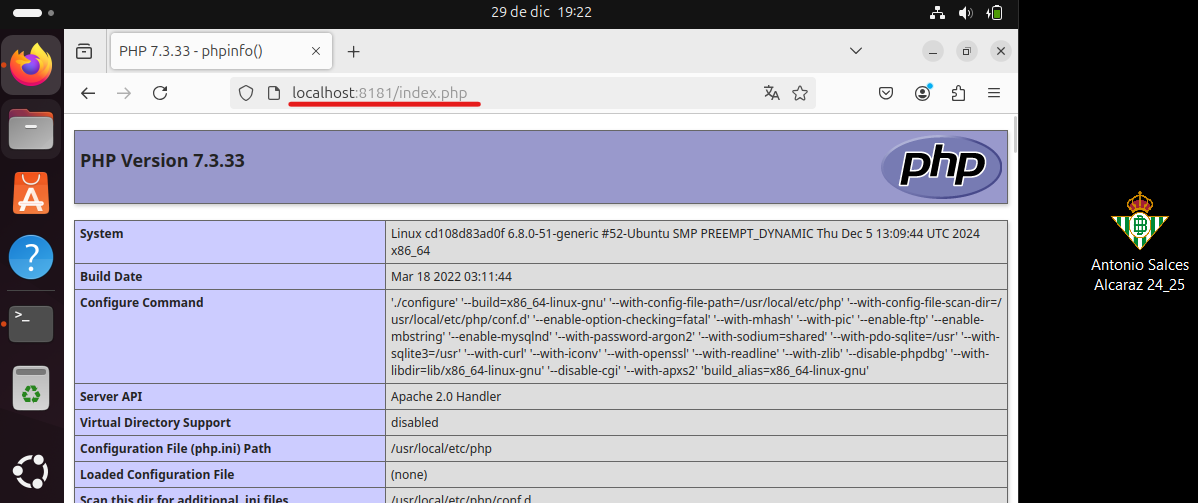
En primer lugar, accederemos al directorio web del contenedor con el primer comando que se ve en la imagen (“***docker exec -it [nombre o ID] bash***”). Una vez estemos en el contenedor, utilizaremos “***echo <contenido del archivo> > <ubicación del archivo>***”. Para comprobar el contenido del archivo,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



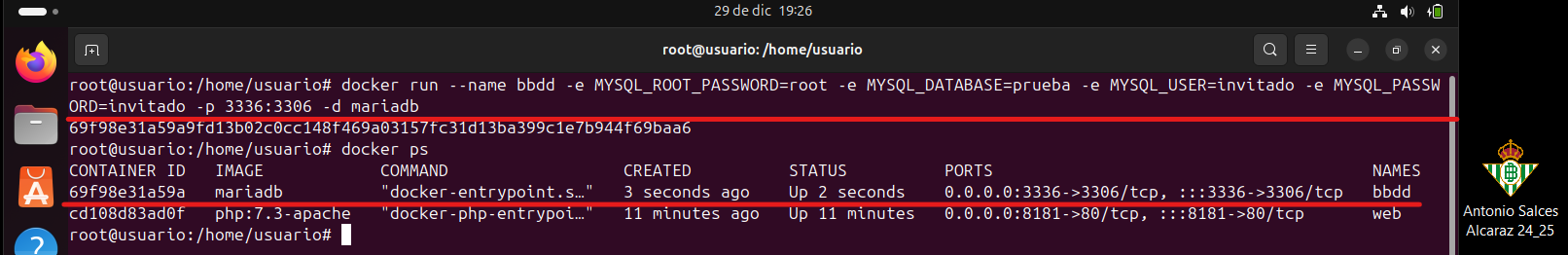
Y esto saldrá si buscamos el fichero index.php.



## 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.

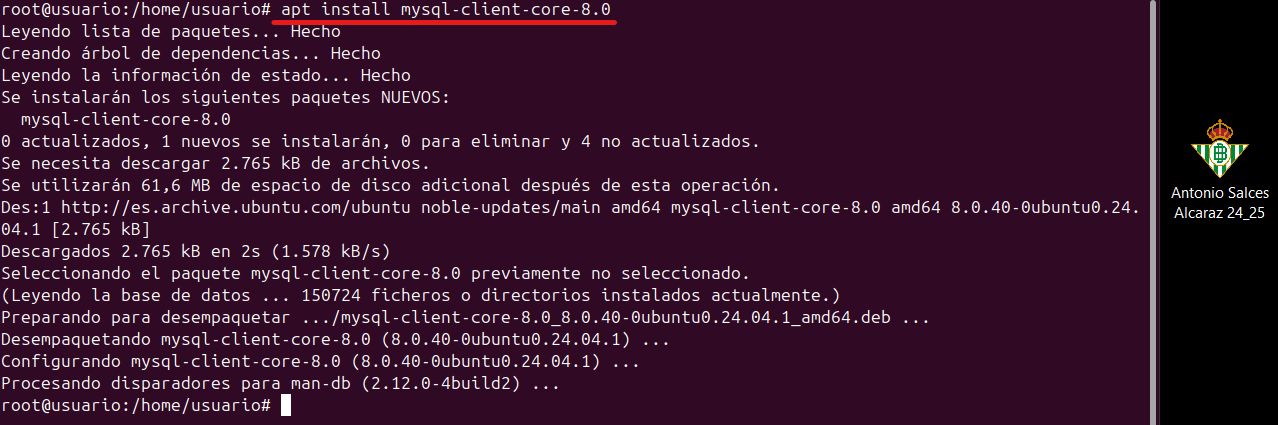


Los parámetros que se han utilizado son los siguientes:

* --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.

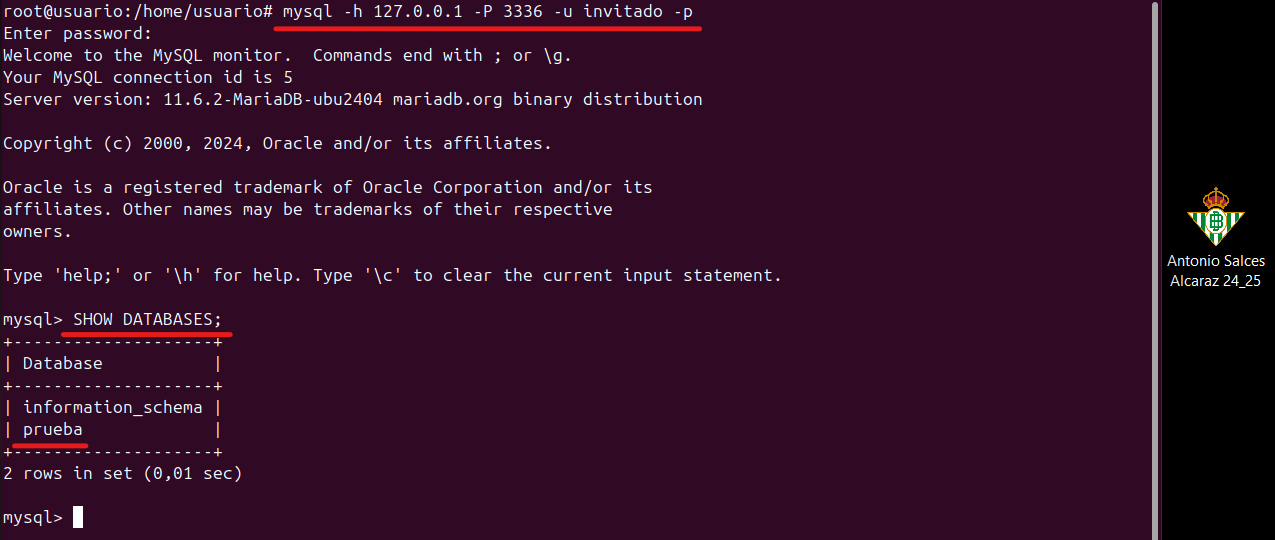
## Conectar con un cliente de base de datos

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.

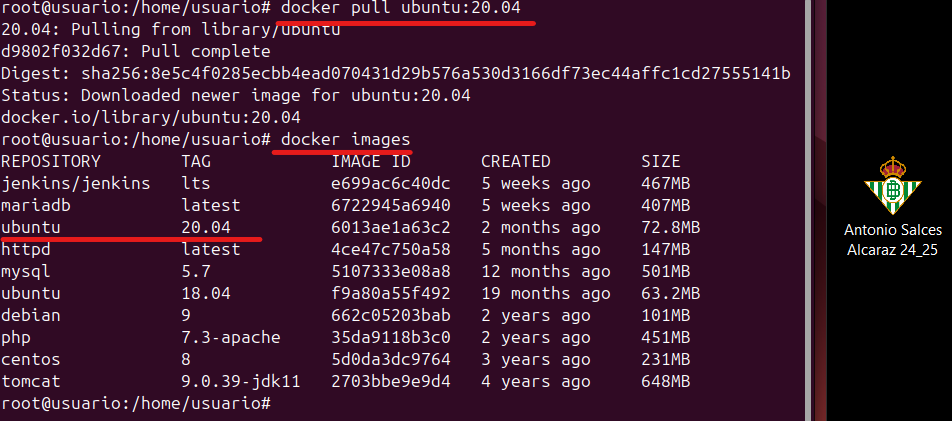
Una vez hayamos conectado con la base de datos, podemos utilizar el comando “***SHOW DATABASES;***” para ver las bases de datos creadas. Podemos ver en la imagen que la base de datos “prueba” se ha creado.



# Actividad 3: gestión de imágenes

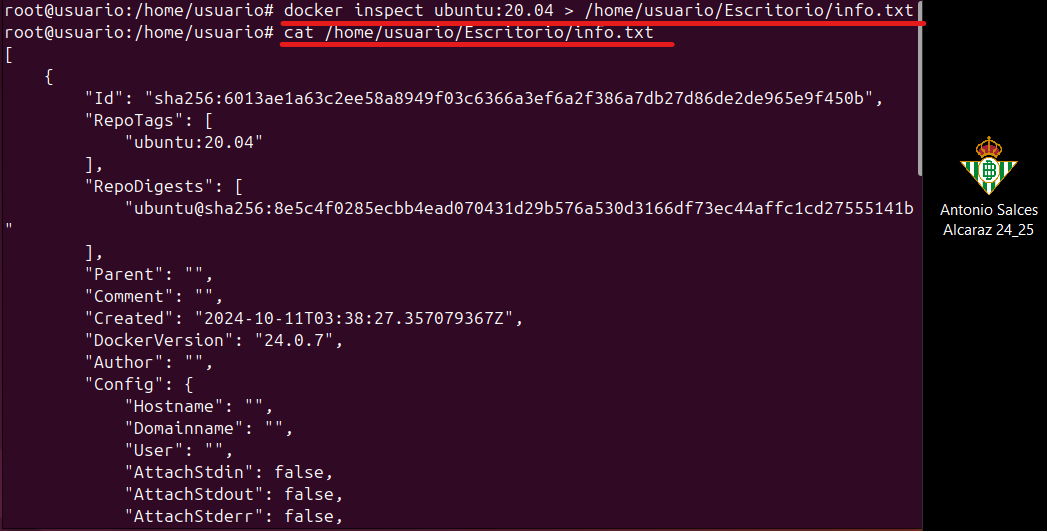
## Descargar imagen Ubuntu:20.04

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*”.**



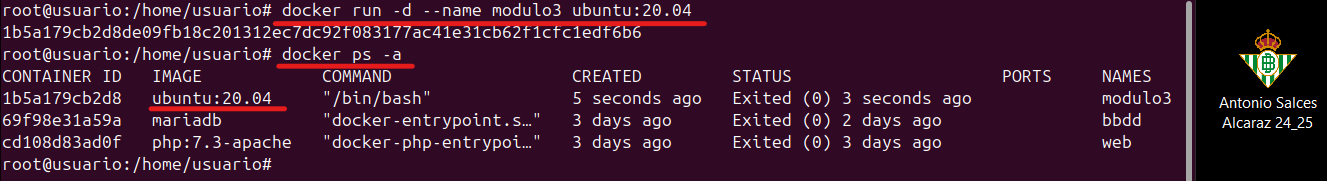
## Volcar la información de la nueva imagen en un archivo

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>***”.



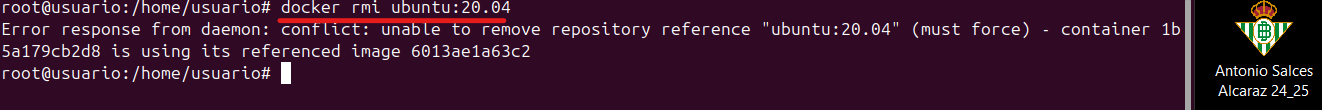
## Crear contenedor con Ubuntu:20.04 llamada “modulo3”

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.



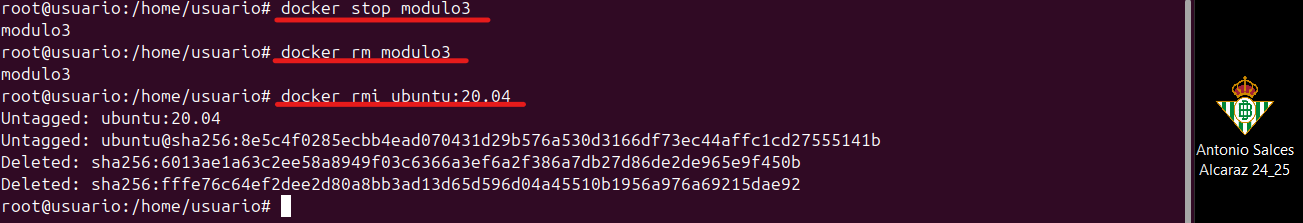
## Borrar contenedor con Ubuntu:20.04

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

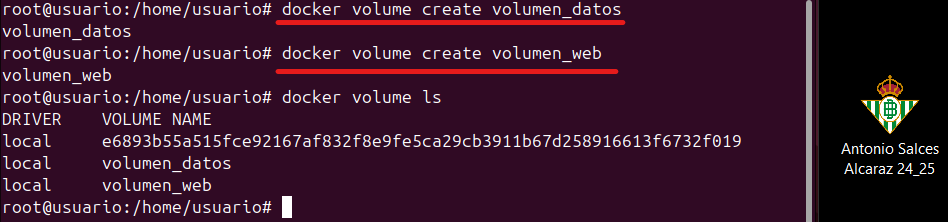
Para borrar la imagen, en primer lugar debemos parar el contenedor con “***docker stop <contenedor>***”, luego borrar con “***docker rm <contenedor>***”, y por ultimo borrar la imagen con “***docker rmi <imagen>***”.



# Actividad 4: operaciones con volúmenes

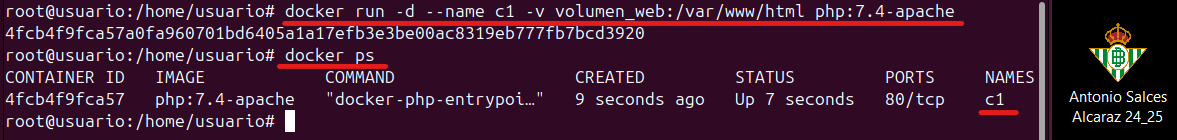
## Crear volúmenes

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.

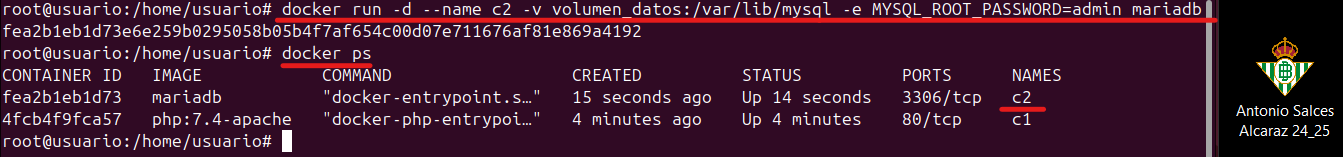


## Instanciar contenedores en los volúmenes

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

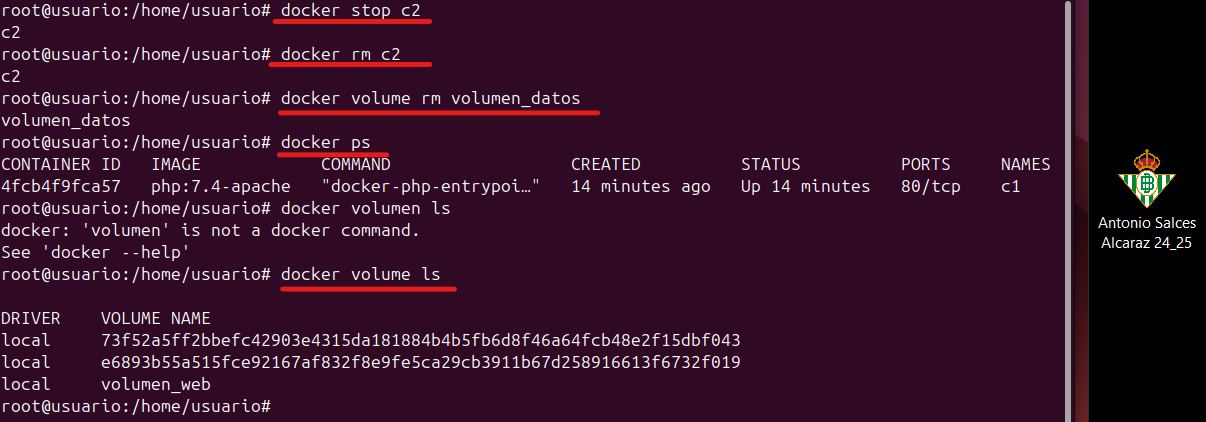


Con el comando “***docker run [-d] [--name <nombre>] >] [-v <volumen>:][ruta] [-e<MYSQL\_ROOT\_PASSWORD>=<contraseña>] <imagen>***” podremos crear un contenedor, elegir el volumen y la ruta donde se aloja y, en este caso (se trata de MariaDB) podemos asignar también una contraseña al usuario “root”.



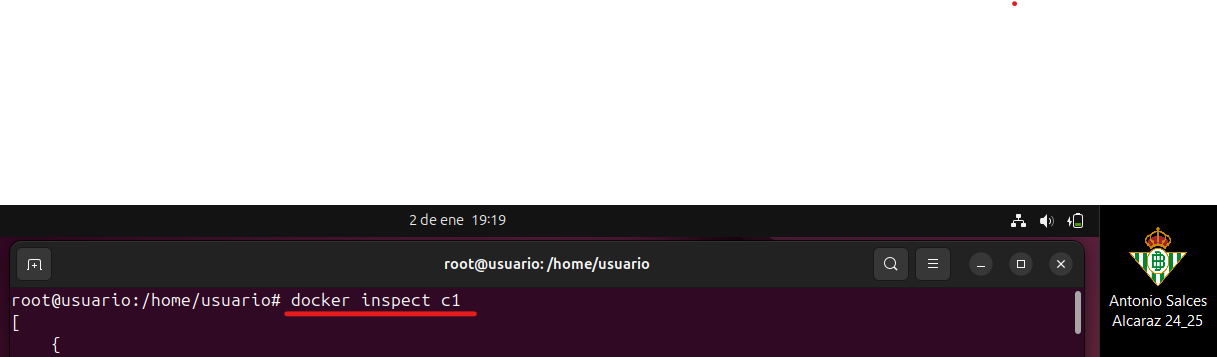
## Parar y borrar contenedor “c2” y volumen “volumen\_datos”

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”.

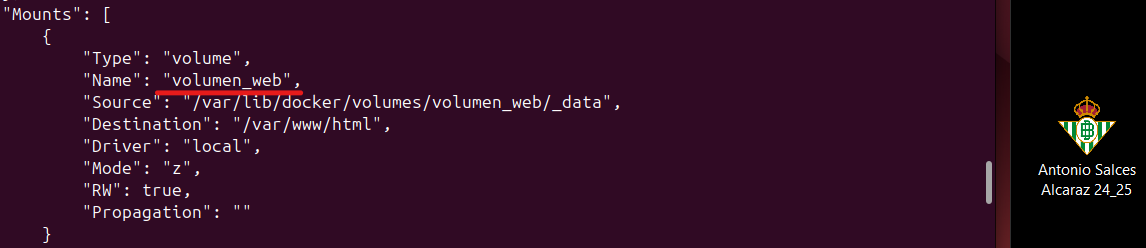


## Comprobar que “c1” está montado sobre “volumen\_web”

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”.



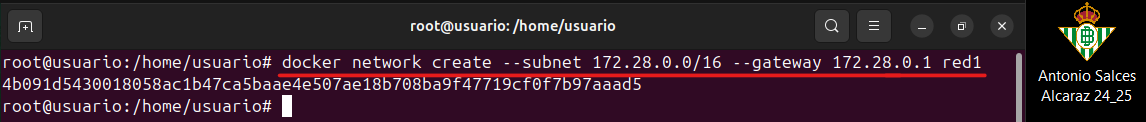
# Actividad 5: trabajo con redes

## Creación de redes

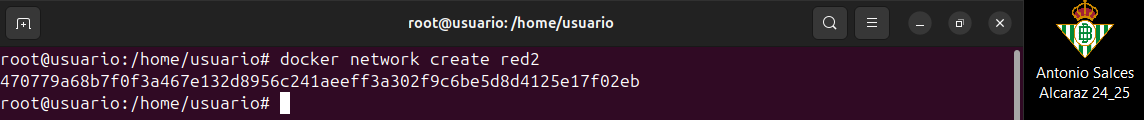
En primer lugar, debemos de crear una red con los siguientes requisitos:

* **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.

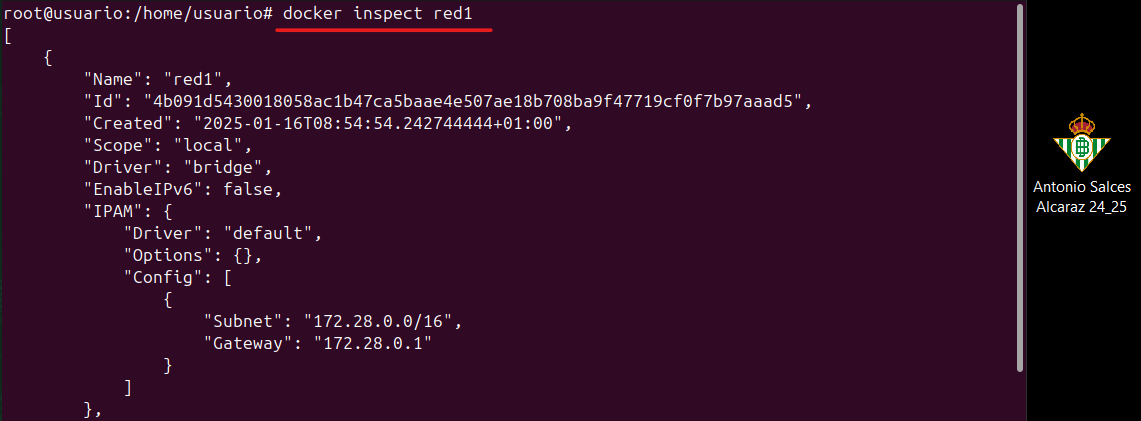


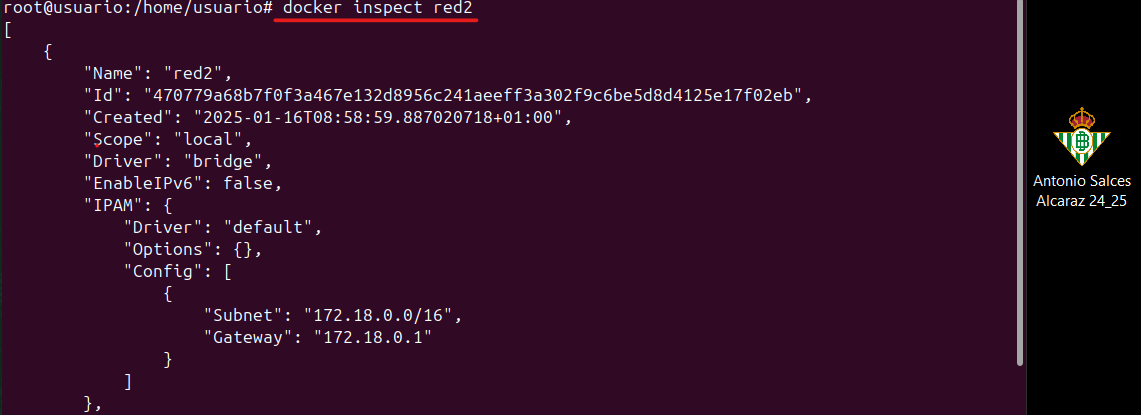
## Configuración de ambas redes

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

## 

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



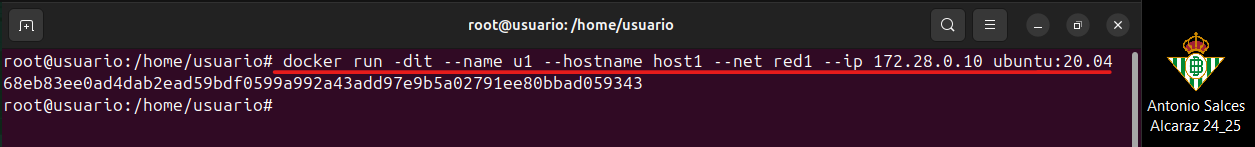


## Arrancar contenedor Ubuntu:20.04 con configuración específica

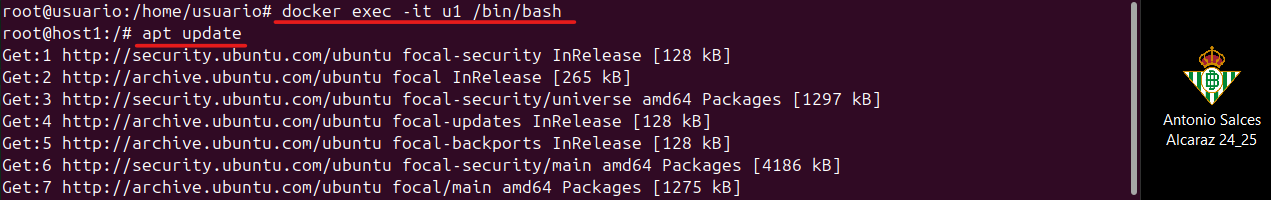
Debemos arrancar un contenedor con la siguiente configuración:

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

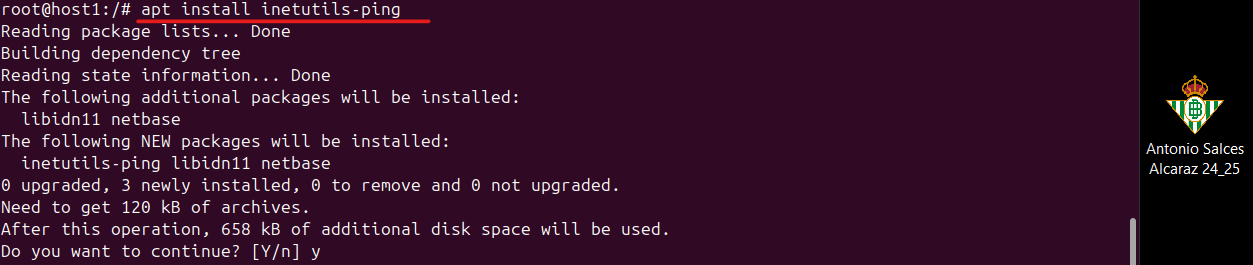
Para ello, utilizaremos el comando “***docker run [-dit] [--name <nombre>] [--hostname <nombre>] [--net <red>] [--ip <IP>] <imagen>***”.



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*”.

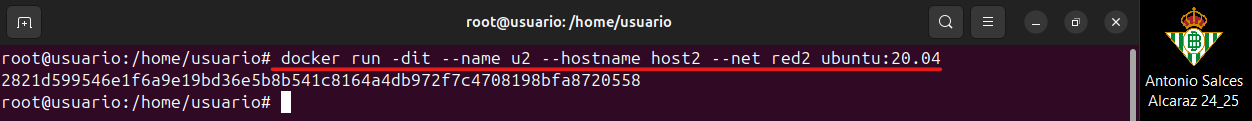


## Arrancar otro contenedor Ubuntu:20.04 con otra configuración

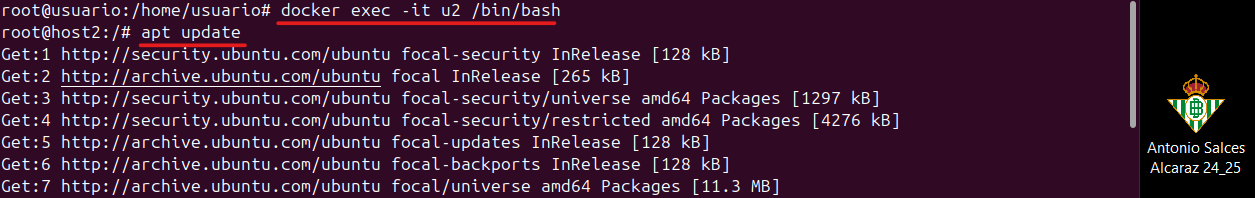
Debemos arrancar otro contenedor con Ubuntu:20.04 con la siguiente configuración:

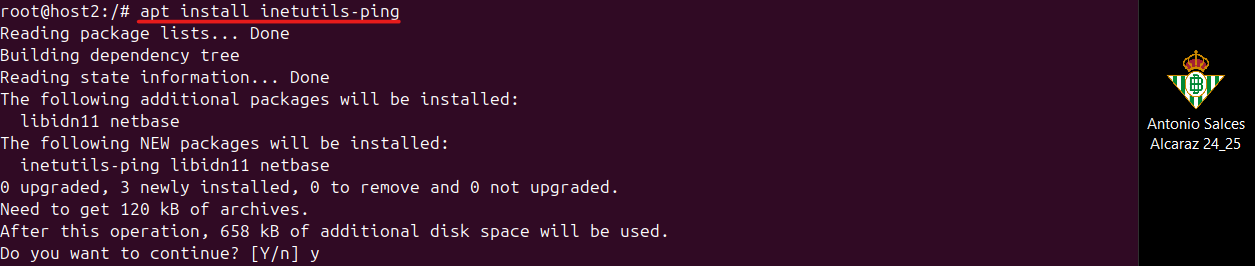
* **Nombre**: u2
* **Hostname**: host2
* **Red**: red2
* **IP**: automática
* **Imagen**: Ubuntu:20.04

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*”.



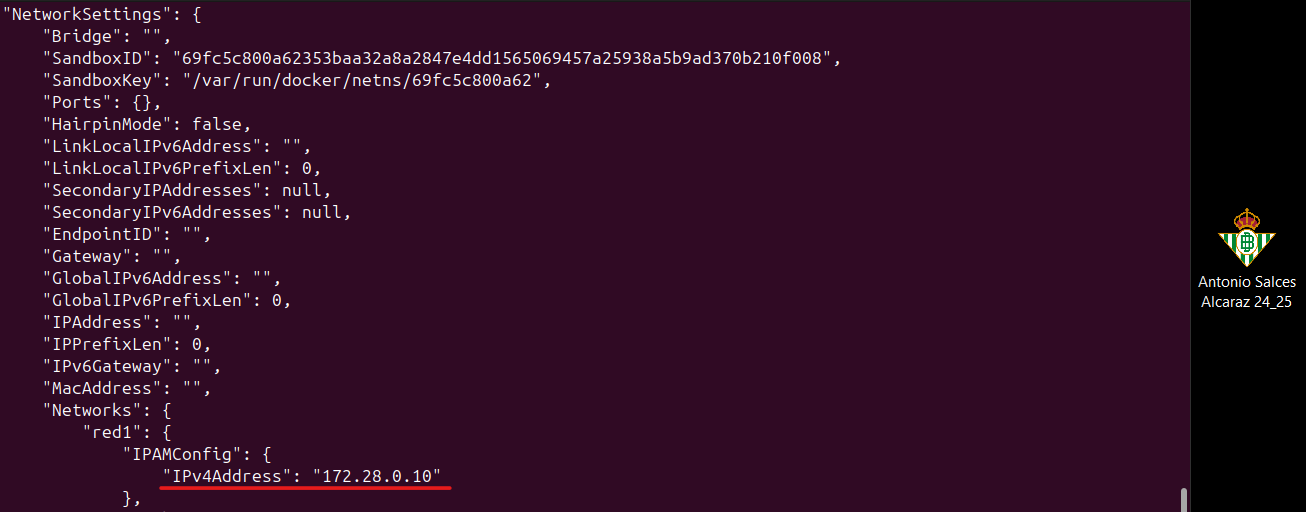


## Configuración de red de ambos contenedores

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”.

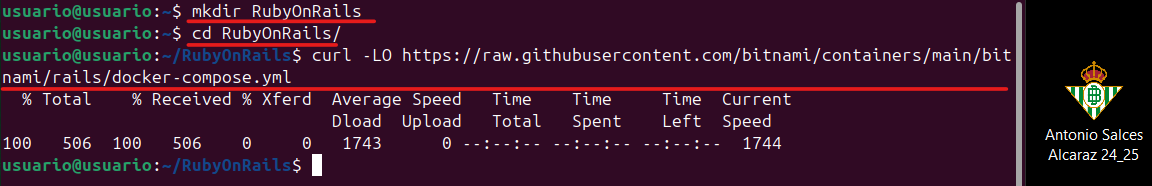




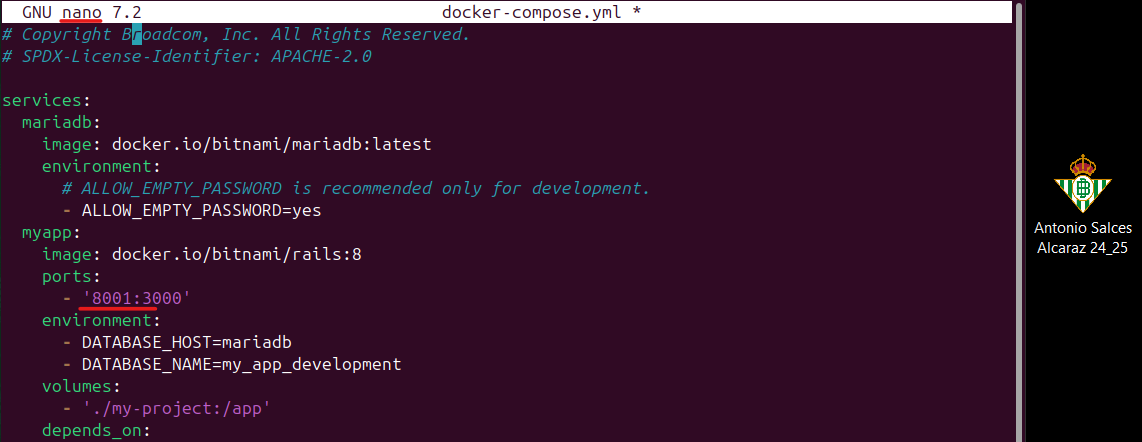
# Actividad 5: editar fichero de configuración .yml

## 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*.



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



Tras editar el fichero de configuración, utilizaremos “***docker-compose up***”