

# Contenedores en red y Docker Desktop

Documento realizado por Roberto Delgado Sánchez - Alumno de Despliegue de Aplicaciones Web - DAW

## Contenedores en red y Docker Desktop

1. Enunciado
2. Creación de la red bridge
3. Creación de un contenedor con imagen de mariaDB
4. Creación de un contenedor con Adminer o phpMyAdmin
5. Comprobación de la conexión entre ambos contenedores
6. Instalación de Disk Usage
7. Borrado de los contenedores y red creados

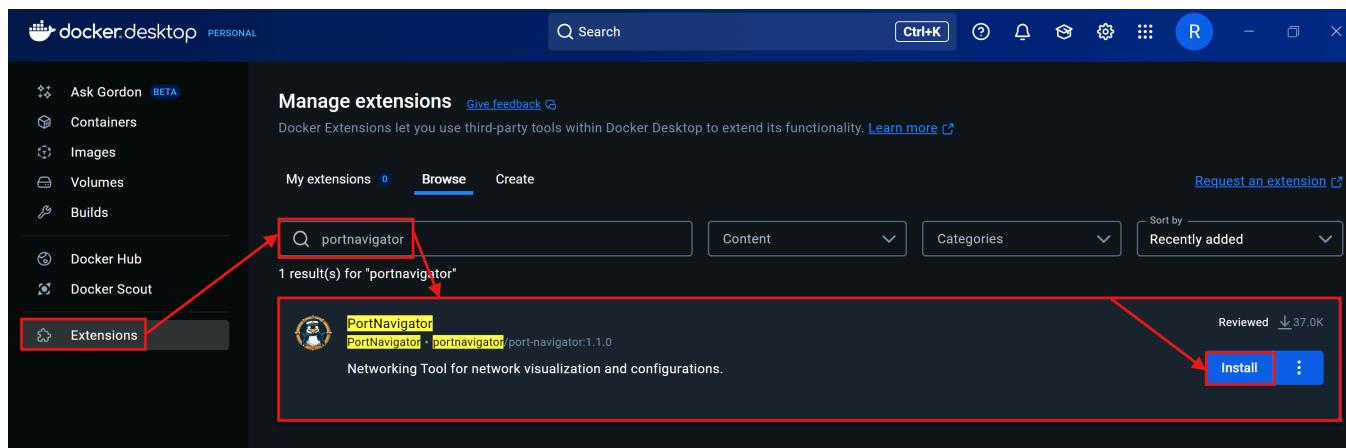
## 1. Enunciado

En este ejercicio vamos a crear dos contenedores, uno a partir de una imagen de **mariaDB** y otro a partir de una imagen de **Adminer** o **phpMyAdmin** (hay que elegir una). El objetivo final es conectar ambos contenedores en red y acceder desde el segundo al primero.

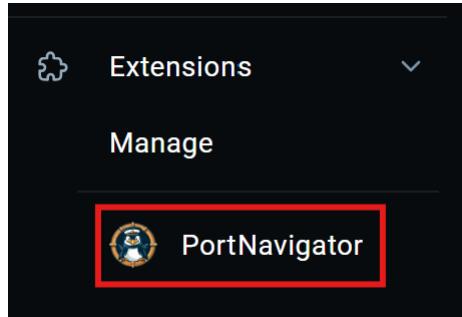
Es obligatorio el empleo de **Docker Desktop** para la resolución del ejercicio, así como la instalación de un par de extensiones (**PortNavigator** y **Disk Usage**) en dicha aplicación. Se incluirán en esta documentación capturas de pantalla y los datos que se consideren oportunos respecto al proceso seguido.

## 2. Creación de la red bridge

El primer paso es entrar en **Docker Desktop** e instalar la extensión **PortNavigator**, que usaremos para crear y gestionar redes:



Una vez instalada nos aparecerá su ícono:



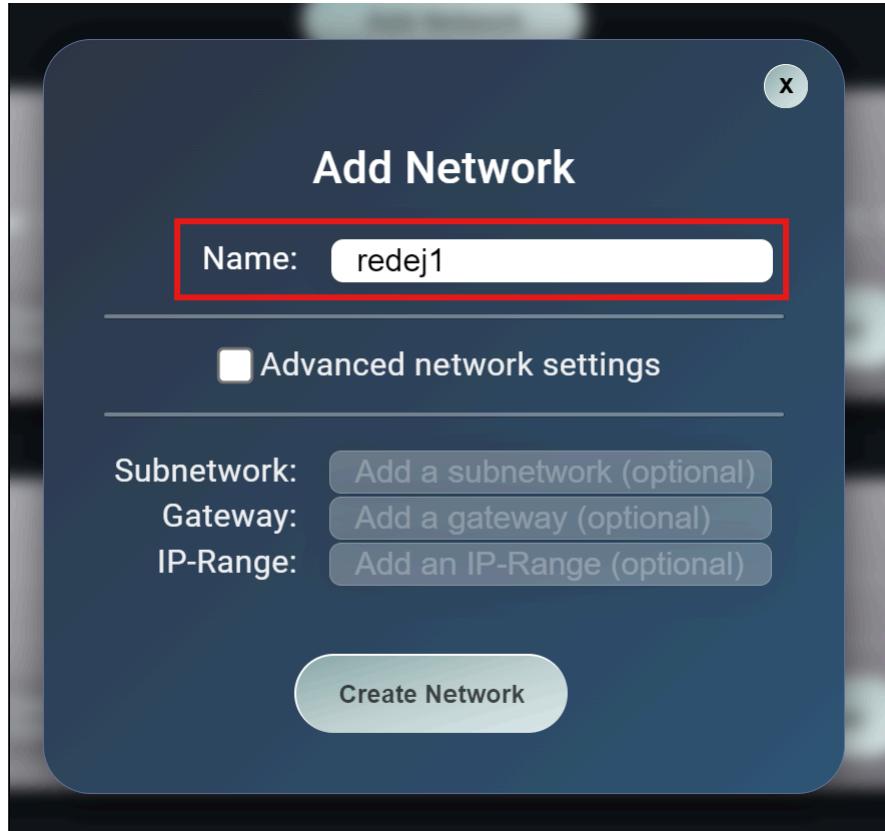
Y si hacemos clic en el mismo accederemos al menú de configuración de redes, donde podemos ver las redes que **Docker** tiene creadas por defecto:

A screenshot of the Docker Network configuration interface. It shows three network profiles: "Network: bridge", "Network: host", and "Network: none".

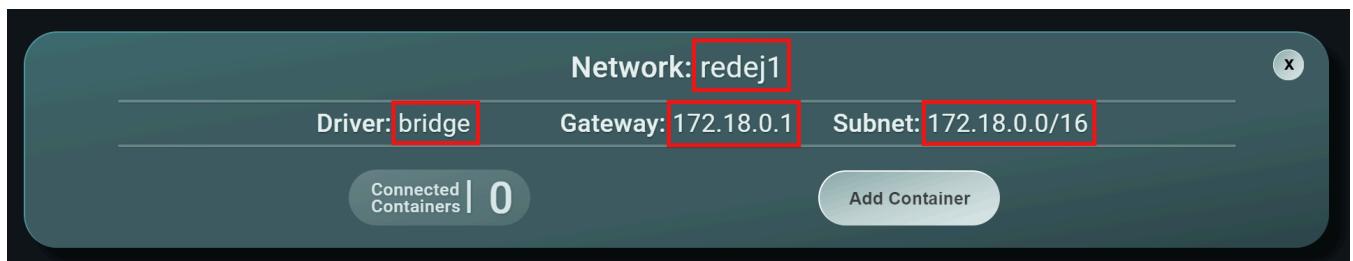
- Network: bridge**: Driver: bridge, Gateway: 172.17.0.1, Subnet: 172.17.0.0/16. It has 0 connected containers and a "Add Container" button.
- Network: host**: Driver: host, Gateway: null, Subnet: null. It has 0 connected containers and a "Add Container" button.
- Network: none**: Driver: null, Gateway: null, Subnet: null. It has 0 connected containers and a "Add Container" button.

The "Visualizer" tab is selected at the top left.

Para crear una red hacemos clic en el botón **Add Network** y rellenamos los parámetros necesarios (sólo se nos indica el nombre de la red, que debe ser **redej1**):



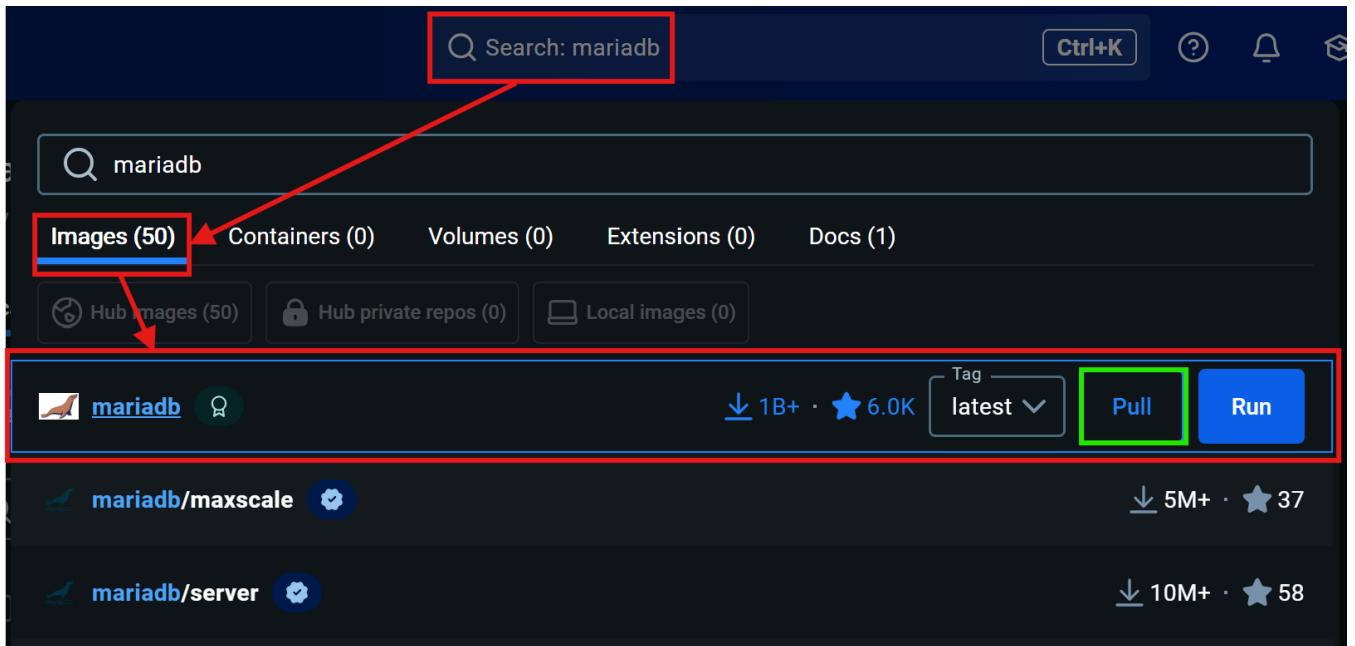
Los parámetros de la red que hemos creado (nombre, tipo de red, puerta de enlace y dirección de red) son los siguientes:



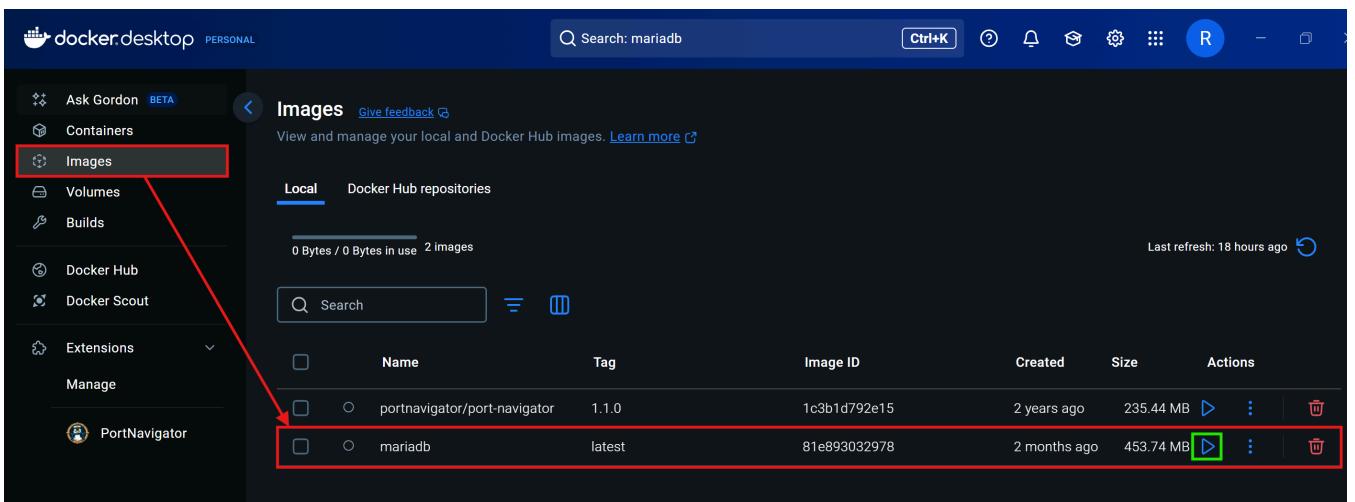
Ya tenemos creada la red.

### 3. Creación de un contenedor con imagen de mariadb

Para crear un contenedor a partir de una imagen basada en **mariadb**, lo primero que tenemos que hacer es buscar una imagen oficial de **mariadb** en **Docker Desktop**, usando para ello el buscador y descargándola con el botón **Pull**:



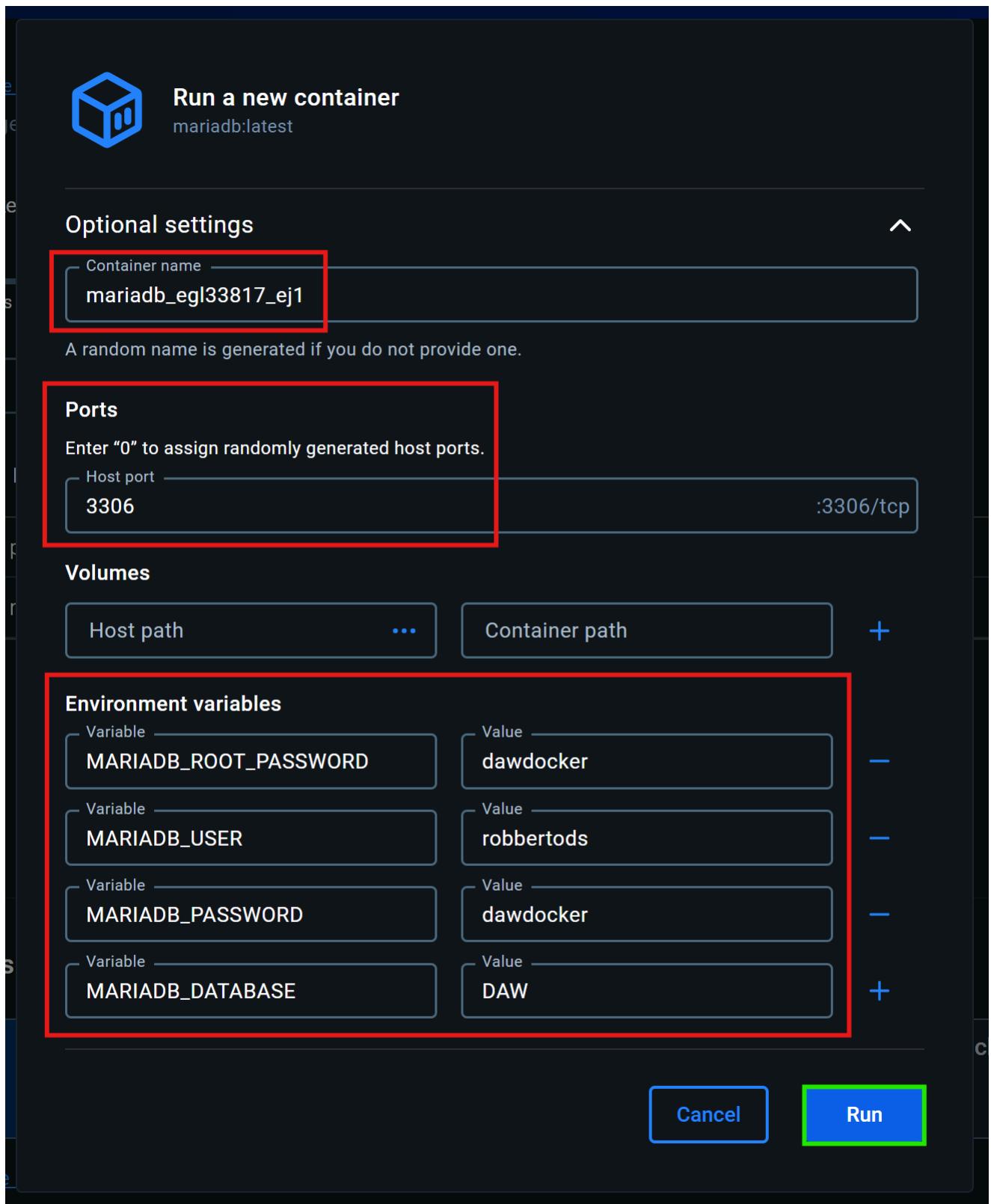
Una vez descargada la imagen, la tendremos disponible en el listado de imágenes. Para crear el contenedor sólo tenemos que hacer clic en el icono con forma de triángulo marcado en verde en la imagen siguiente, que es el equivalente al comando `docker run` de la línea de comandos:



Cuando pulsamos dicho botón, nos aparece una ventana en la que podemos configurar diversos parámetros de nuestro contenedor. Aquí tenemos que incluir los datos aportados por el enunciado del ejercicio:

- **nombre del contenedor:** no se especifica ninguno, así que uso una combinación del nombre de la imagen, mi nombre de usuario en `educastur` y el número de ejercicio.
- **puerto:** debe ser accesible desde el puerto 3306.
- **variables de entorno:** según la documentación de la imagen de `mariadb`, cuando creamos un contenedor a partir de la misma podemos definir diversos parámetros mediante este tipo de variables, como por ejemplo la contraseña del usuario root (`MARIADB_ROOT_PASSWORD`), crear un usuario (`MARIADB_USER`), su contraseña (`MARIADB_PASSWORD`) y una base de datos por defecto (`MARIADB_DATABASE`), que en nuestro caso debe llamarse `DAW`. El resto de datos se escogen al azar, siendo el usuario una combinación de mi nombre y apellidos.

Estas opciones quedan tal y como se puede ver en la siguiente imagen:



Al hacer clic en el botón **Run** se crea el contenedor (en la siguiente imagen se puede ver parte del log donde se recoge el proceso de instalación y se pueden ver cosas como el nombre del contenedor o la creación del usuario y de la base de datos por defecto):

```

mariadb_egl33817_ej1
877144cebd34 mariadb:latest
3306:3306

Logs Inspect Bind mounts Exec Files Stats
2025-04-17 08:48:09+00:00 [Note] [Entrypoint]: Temporary server started.
2025-04-17 08:48:10+00:00 [Note] [Entrypoint]: Creating database DAW
2025-04-17 08:48:10+00:00 [Note] [Entrypoint]: Creating user robbertods
2025-04-17 08:48:10+00:00 [Note] [Entrypoint]: Giving user robbertods access to schema DAW
2025-04-17 08:48:10+00:00 [Note] [Entrypoint]: Securing system users (equivalent to running mysql_secure_installation)

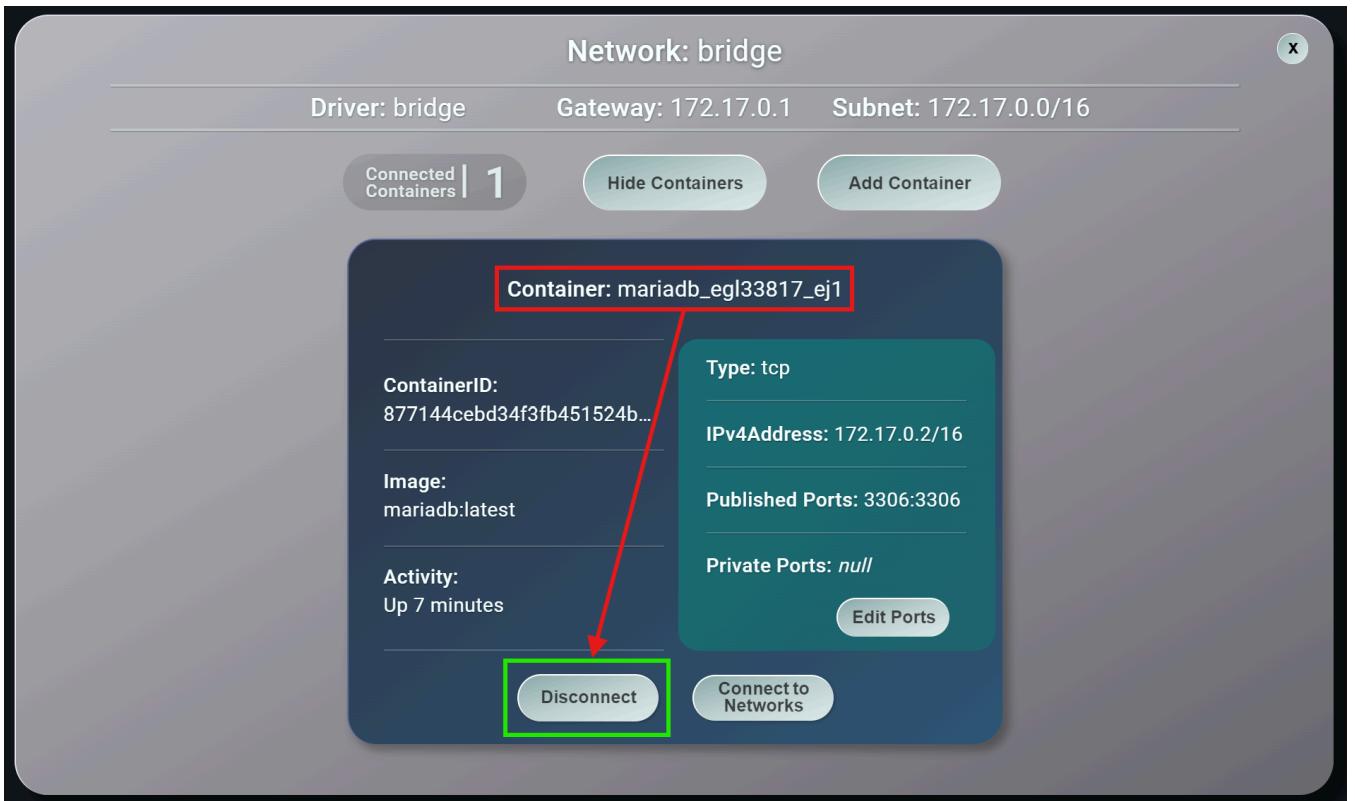
2025-04-17 08:48:10+00:00 [Note] [Entrypoint]: Stopping temporary server
2025-04-17 08:48:10+00:00 [Note] mariadb (initiated by: unknown): Normal shutdown
2025-04-17 08:48:10+00:00 [Note] InnoDB: FTS optimize thread exiting.
2025-04-17 08:48:10+00:00 [Note] InnoDB: Starting shutdown...

```

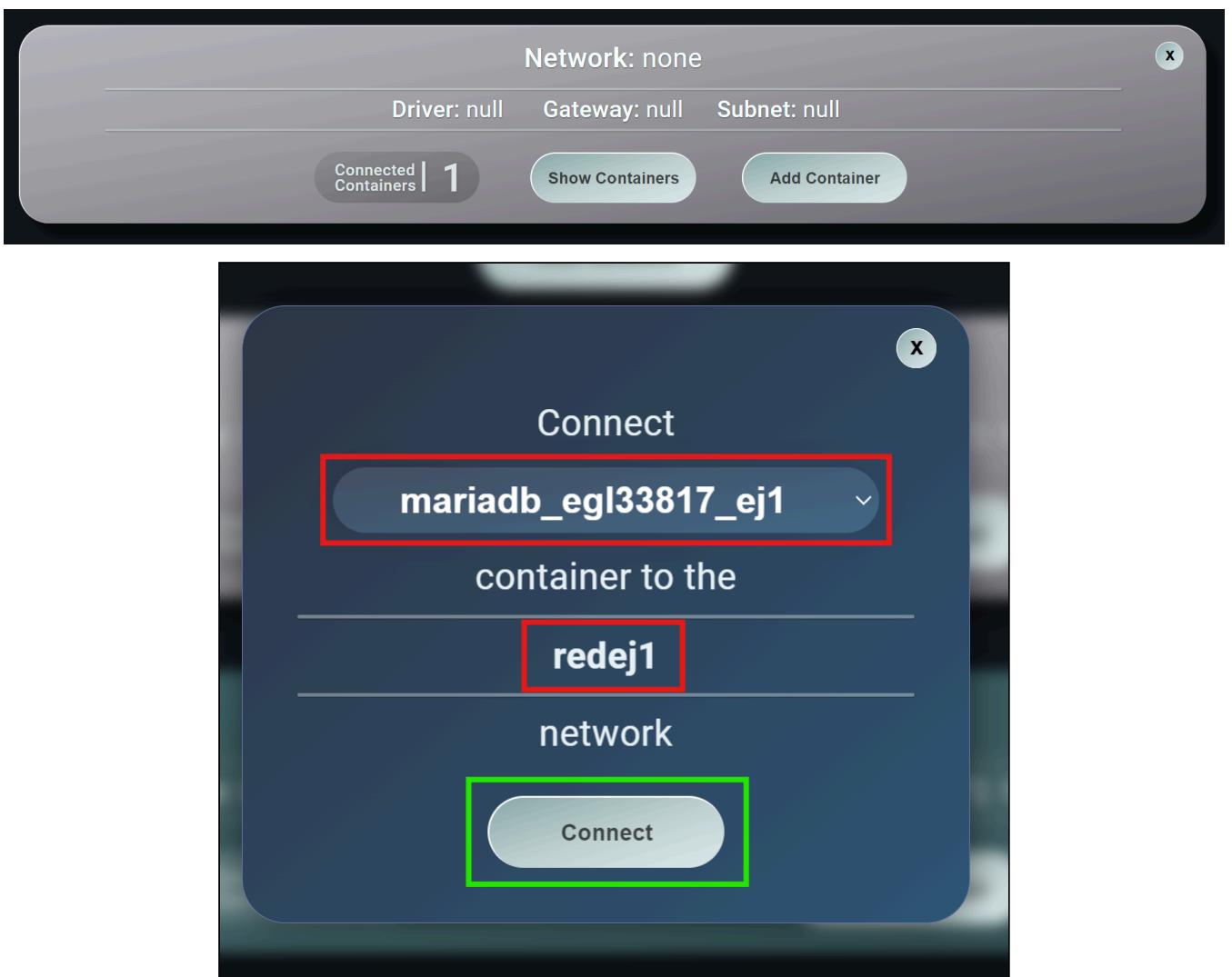
Ahí tenemos nuestro contenedor creado y en funcionamiento:

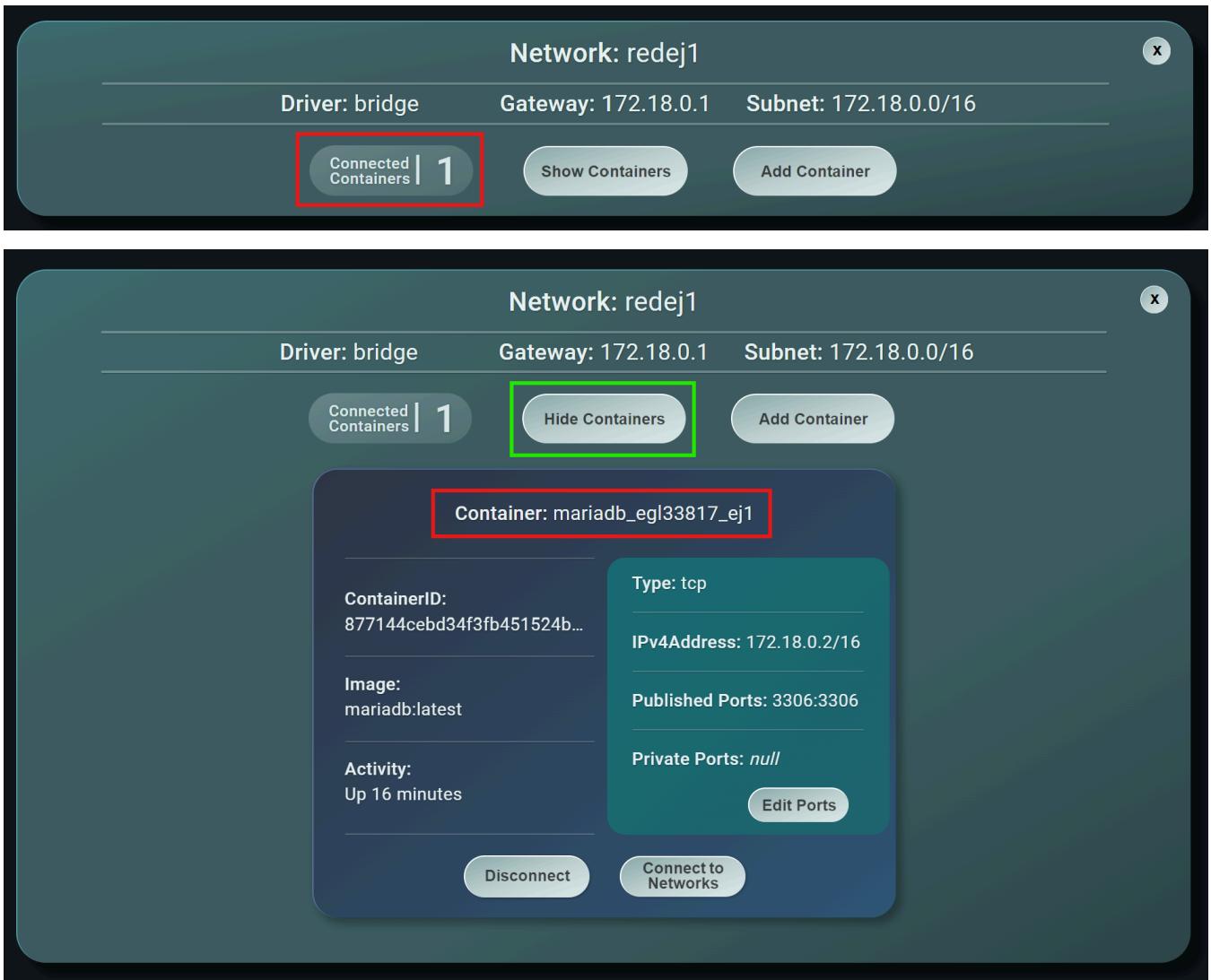
|                          | Name                 | Container ID | Image          | Port(s)   | CPU (%) | Last started  | Actions |
|--------------------------|----------------------|--------------|----------------|-----------|---------|---------------|---------|
| <input type="checkbox"/> | mariadb_egl33817_ej1 | 877144cebd34 | mariadb:latest | 3306:3306 | 0.02%   | 5 minutes ago |         |

Sólo quedaría pendiente conectar el contenedor que acabamos de crear a la red `redej1`. Para ello, tenemos en primer lugar que desconectar el contenedor de la red bridge que Docker crea por defecto, y después conectarlo a nuestra red `redej1`. Los pasos se pueden ver a continuación:



Al desconectarlo, queda asignado automáticamente a una red sin nombre, desde la cual lo podemos conectar a la red que hemos creado **redej1**:





Ya tenemos creado el contenedor basado en la imagen de **mariDB**. Ahora hay que generar un script **SQL** que cree una tabla de nombre módulos con algunos registros, como los nombres de las asignaturas en las que estoy matriculado:

```
CREATE TABLE modulos
(
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombreAsignatura VARCHAR(60) NOT NULL
);

INSERT INTO modulos (nombreAsignatura) VALUES
('bases de datos'),
('programación'),
('interfaces'),
('despliegue'),
('cliente'),
('servidor');
```

De momento lo dejamos así, ya que este script se usará más adelante.

## 4. Creación de un contenedor con Adminer o phpMyAdmin

De las dos opciones que se nos dan escogemos **Adminer** para crear otro contenedor. Los pasos son similares a los que seguimos para crear el contenedor basado en **mariadb**, por lo que mostraremos las capturas de pantalla del proceso haciendo sólo hincapié en aquello que sea diferente:

The screenshot shows the Docker desktop application interface. At the top, there is a search bar with the placeholder "Search: adminer". Below the search bar, there is a navigation bar with tabs: "Images (50)", "Containers (0)", "Volumes (0)", "Extensions (0)", and "Docs (0)". The "Images (50)" tab is selected. Underneath the tabs, there are three buttons: "Hub images (50)", "Hub private repos (0)", and "Local images (0)". The main area displays a list of Docker images. One image, "adminer", is highlighted with a red box. This image has a download count of "100M+", a star rating of "943", and a "latest" tag. To the right of the image details are two buttons: "Pull" and "Run", with "Pull" being highlighted by a green box. Below this, another image, "dockette/adminer", is listed with a download count of "5M+" and a star rating of "29". At the bottom of the list, there is a table with columns: "Name", "Tag", "Image ID", "Created", "Size", and "Actions". The "adminer" image from the list is also present in this table, highlighted with a red box. The "Actions" column for this row contains icons for "Edit", "Delete", and "Copy".

| Name                         | Tag    | Image ID     | Created      | Size      | Actions |
|------------------------------|--------|--------------|--------------|-----------|---------|
| portnavigator/port-navigator | 1.1.0  | 1c3b1d792e15 | 2 years ago  | 235.44 MB |         |
| mariadb                      | latest | 81e893032978 | 2 months ago | 453.74 MB |         |
| adminer                      | latest | b1c1395ff805 | 2 days ago   | 173.84 MB |         |

Find your local and Docker Hub images. [Learn more](#)



## Run a new container

adminer:latest

### Optional settings

Container name

adminer\_egl33817\_ej1

A random name is generated if you do not provide one.

### Ports

Enter "0" to assign randomly generated host ports.

Host port

8080

:8080/tcp

### Volumes

Host path

...

Container path

+

### Environment variables

Variable

Value

+

Cancel

Run

[Containers](#) / adminer\_egl33817\_ej1

adminer\_egl33817\_ej1



109337b40696



adminer:latest

8080:8080



Logs

Inspect

Bind mounts

Exec

Files

Stats

[Thu Apr 17 13:41:56 2025] PHP 8.4.6 Development Server (http://[::]:8080) started

Nuestros dos contenedores ya están en marcha:

## Containers [Give feedback](#)

View all your running containers and applications. [Learn more](#)

Container CPU usage [\(i\)](#)

0.03% / 1200% (12 CPUs available)

Container memory usage [\(i\)](#)

282.23MB / 7.49GB

Search



Only show running containers

| <input type="checkbox"/> | Name                 | Container ID | Image                          | Port(s)                   | CPU (%) | Last started | Actions |
|--------------------------|----------------------|--------------|--------------------------------|---------------------------|---------|--------------|---------|
| <input type="checkbox"/> | mariadb_egl33817_ej1 | 877144cebd34 | <a href="#">mariadb:latest</a> | <a href="#">3306:3306</a> | 0.03%   | 5 hours ago  |         |
| <input type="checkbox"/> | adminer_egl33817_ej1 | 109337b40696 | <a href="#">adminer:latest</a> | <a href="#">8080:8080</a> | 0%      | 1 minute ago |         |

Metemos al contenedor creado con la imagen de Adminer en la red `redej1`:

Network: bridge

Driver: bridge      Gateway: 172.17.0.1      Subnet: 172.17.0.0/16

Connected Containers | 1      Hide Containers      Add Container

Container: adminer\_egl33817\_ej1

ContainerID: dbbce031717e4d1a3bbc95...

Image: adminer:latest

Activity: Up 2 minutes

Type: tcp

IPv4Address: 172.17.0.2/16

Published Ports: null

Private Ports: 8080

[Edit Ports](#)

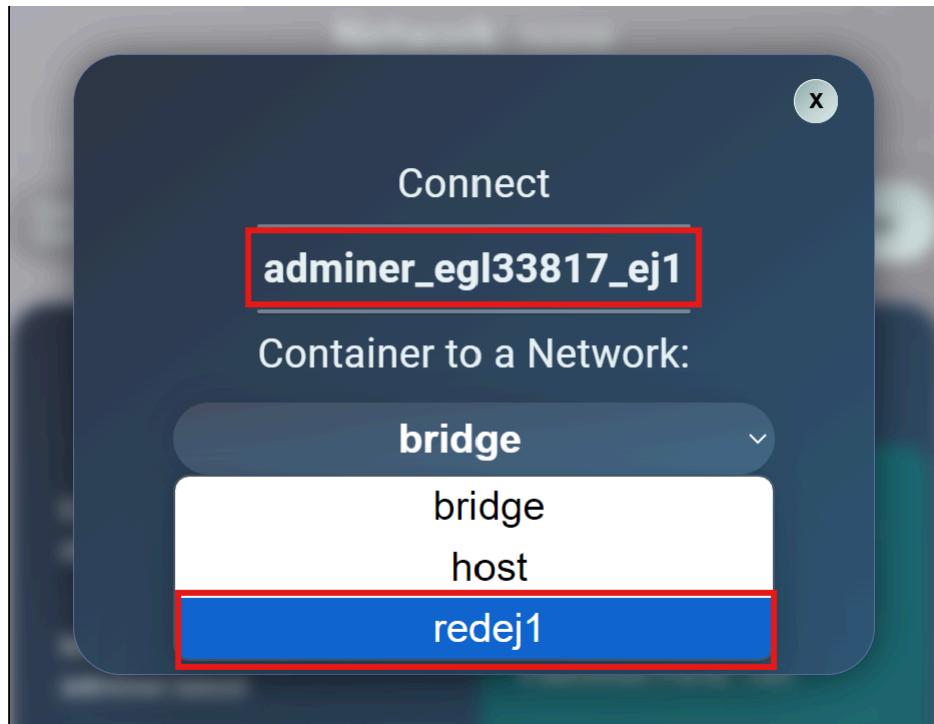
[Disconnect](#)      [Connect to Networks](#)

Al desconectarlo, queda asignado automáticamente a una red sin nombre, desde la cual lo podemos conectar a la red que hemos creado `redej1`:

Network: none

Driver: null      Gateway: null      Subnet: null

Connected Containers | 1      Show Containers      Add Container



Ya tenemos a ambos contenedores conectados a nuestra red **redej1**:

| Container                       | Type | IP Address    | Published Ports | Private Ports |
|---------------------------------|------|---------------|-----------------|---------------|
| Container: adminer_egl33817_ej1 | tcp  | 172.18.0.3/16 | 8080:8080       | null          |
| Container: mariadb_egl33817_ej1 | tcp  | 172.18.0.2/16 | 3306:3306       | null          |

## 5. Comprobación de la conexión entre ambos contenedores

Una vez que sabemos la dirección IP de cada contenedor, podemos comprobar si se puede hacer un **ping** de **adminer** a **mariadb**:

[Containers](#) / adminer\_eg133817\_ej1

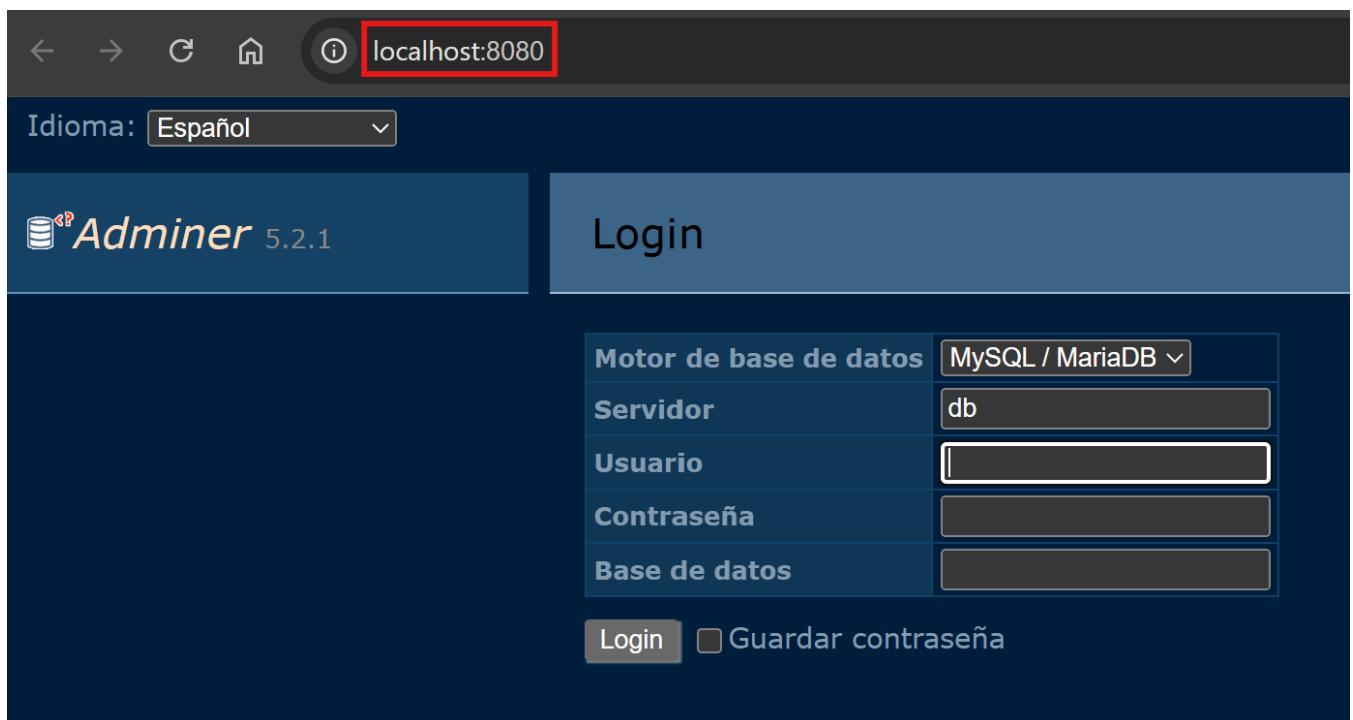
### adminer\_eg133817\_ej1

<  109337b40696 ⚡  [adminer:latest](#)  
[8080:8080](#)

Logs Inspect Bind mounts **Exec** Files Stats

```
/var/www/html $ ping 172.18.0.2
PING 172.18.0.2 (172.18.0.2): 56 data bytes
64 bytes from 172.18.0.2: seq=0 ttl=42 time=0.804 ms
64 bytes from 172.18.0.2: seq=1 ttl=42 time=0.118 ms
64 bytes from 172.18.0.2: seq=2 ttl=42 time=0.108 ms
^C
--- 172.18.0.2 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.108/0.343/0.804 ms
/var/www/html $ []
```

El resultado es correcto, así que vamos a conectarnos ahora sí a la interfaz gráfica de `adminer_eg133817_ej1` a través del navegador:



Ponemos los datos de conexión que definimos al crear el contenedor `mariadb_eg133817_ej1`:

Idioma: Español

**Adminer** 5.2.1

Login

|                               |                      |
|-------------------------------|----------------------|
| <b>Motor de base de datos</b> | MySQL / MariaDB      |
| <b>Servidor</b>               | mariadb_egl33817_ej1 |
| <b>Usuario</b>                | robbertods           |
| <b>Contraseña</b>             | .....                |
| <b>Base de datos</b>          | DAW                  |

Login  Guardar contraseña

Una vez dentro se nos muestra la pantalla principal:

Idioma: Español

MariaDB » mariadb\_egl33817\_ej1 » Base de datos: DAW

robbertods Cerrar sesión

**Adminer** 5.2.1

Base de datos: DAW

BD: DAW

Modificar Base de datos Esquema de base de datos Privilegios

Comando SQL Importar Exportar Crear tabla

No existen tablas.

Crear tabla Crear vista

Procedimientos

Crear procedimiento Crear función

Eventos

Crear Evento

Vamos a ejecutar ahora el script **SQL** que generamos anteriormente. Para ello vamos a la opción **Comando SQL** del menú:

localhost:8080/?server=mariadb\_egl33817\_ej1&username=robbertods&db=DAW

Idioma: Español

MariaDB » mariadb\_egl33817\_ej1 » DAW » Comando SQL

Adminer 5.2.1

BD: DAW

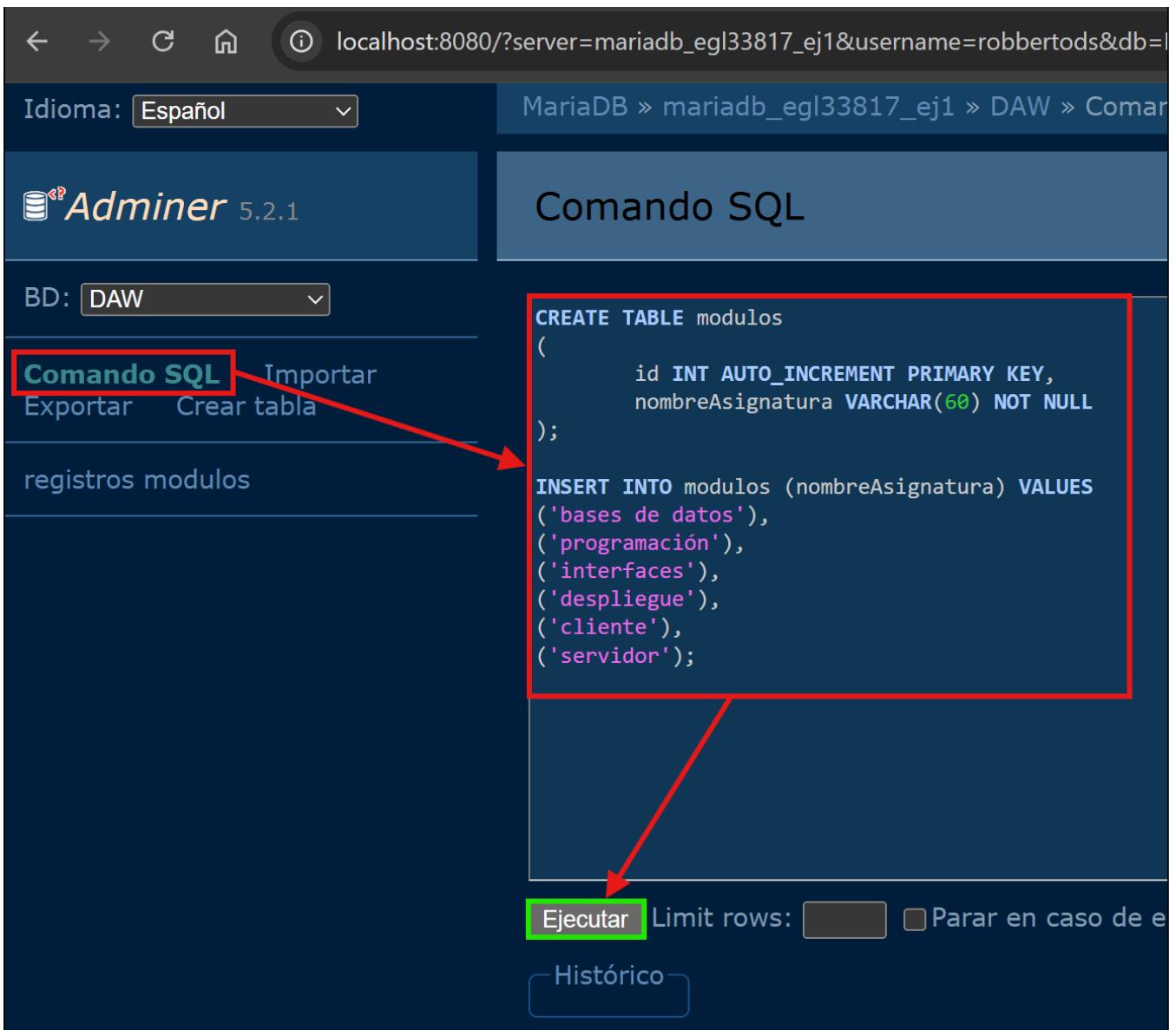
Comando SQL Importar  
Exportar Crear tabla

registros modulos

CREATE TABLE modulos  
(  
 id INT AUTO\_INCREMENT PRIMARY KEY,  
 nombreAsignatura VARCHAR(60) NOT NULL  
);  
  
INSERT INTO modulos (nombreAsignatura) VALUES  
('bases de datos'),  
('programación'),  
('interfaces'),  
('despliegue'),  
('cliente'),  
('servidor');

Ejecutar Limit rows:  Parar en caso de error

Histórico



Las sentencias SQL se han ejecutado correctamente:

localhost:8080/?server=mariadb\_egl33817\_ej1&username=robbertods&db=DAW&s...

Idioma: Español

MariaDB » mariadb\_egl33817\_ej1 » DAW » Comando SQL

## Adminer 5.2.1

BD: DAW

Comando SQL Importar  
Exportar Crear tabla

registros modulos

### Comando SQL

```
CREATE TABLE modulos
(
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombreAsignatura VARCHAR(60) NOT NULL
)
```

Consulta ejecutada, 0 registros afectados. (0.027 s) Modificar

```
INSERT INTO modulos (nombreAsignatura) VALUES
('bases de datos'),
('programación'),
('interfaces'),
('despliegue'),
('cliente'),
('servidor')
```

Consulta ejecutada, 6 registros afectados. (0.008 s) Modificar

```
CREATE TABLE modulos
(
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombreAsignatura VARCHAR(60) NOT NULL
);

INSERT INTO modulos (nombreAsignatura) VALUES
('bases de datos'),
('programación'),
('interfaces'),
('despliegue'),
('cliente'),
('servidor');
```

Comprobamos que, efectivamente, ahora tenemos una tabla `modulos` en la base de datos `DAW` que contiene los datos aportados anteriormente:

The screenshot shows the Adminer web interface at `localhost:8080/?server=mariadb_egl33817_ej1&username=robbertods&db=DAW`. The interface has a dark blue header with the Adminer logo and version 5.2.1. On the left, there's a sidebar with language selection (Español), database selection (BD: DAW), and links for Comando SQL, Importar, Exportar, Crear tabla, and registros modulos. The main area is titled "Comando SQL" and contains a red-bordered SQL query: `SELECT * FROM modulos`. Below the query is a table with a red border showing the results:

| <b>id</b> | <b>nombreAsignatura</b> |
|-----------|-------------------------|
| 1         | bases de datos          |
| 2         | programación            |
| 3         | interfaces              |
| 4         | despliegue              |
| 5         | cliente                 |
| 6         | servidor                |

Below the table, it says "6 registros (0.001 s) Modificar, Explain, Exportar". The bottom part of the interface shows the same SQL command again, with execution buttons: "Ejecutar", "Limit rows: 10", and "Parar en caso de error". There's also a "Histórico" button.

Y así completamos la realización de este apartado del ejercicio.

## 6. Instalación de Disk Usage

Vamos a instalar la extensión **Disk Usage**, que es equivalente al comando `docker system df`, es decir, da información sobre el espacio ocupado, etc. La buscamos desde el gestor de extensiones y la instalamos:

Manage extensions [Give feedback](#)

Docker Extensions let you use third-party tools within Docker Desktop to extend its functionality. [Learn more](#)

My extensions 0    Browse    Create    Request an extension

Sort by Recently added

Content Categories

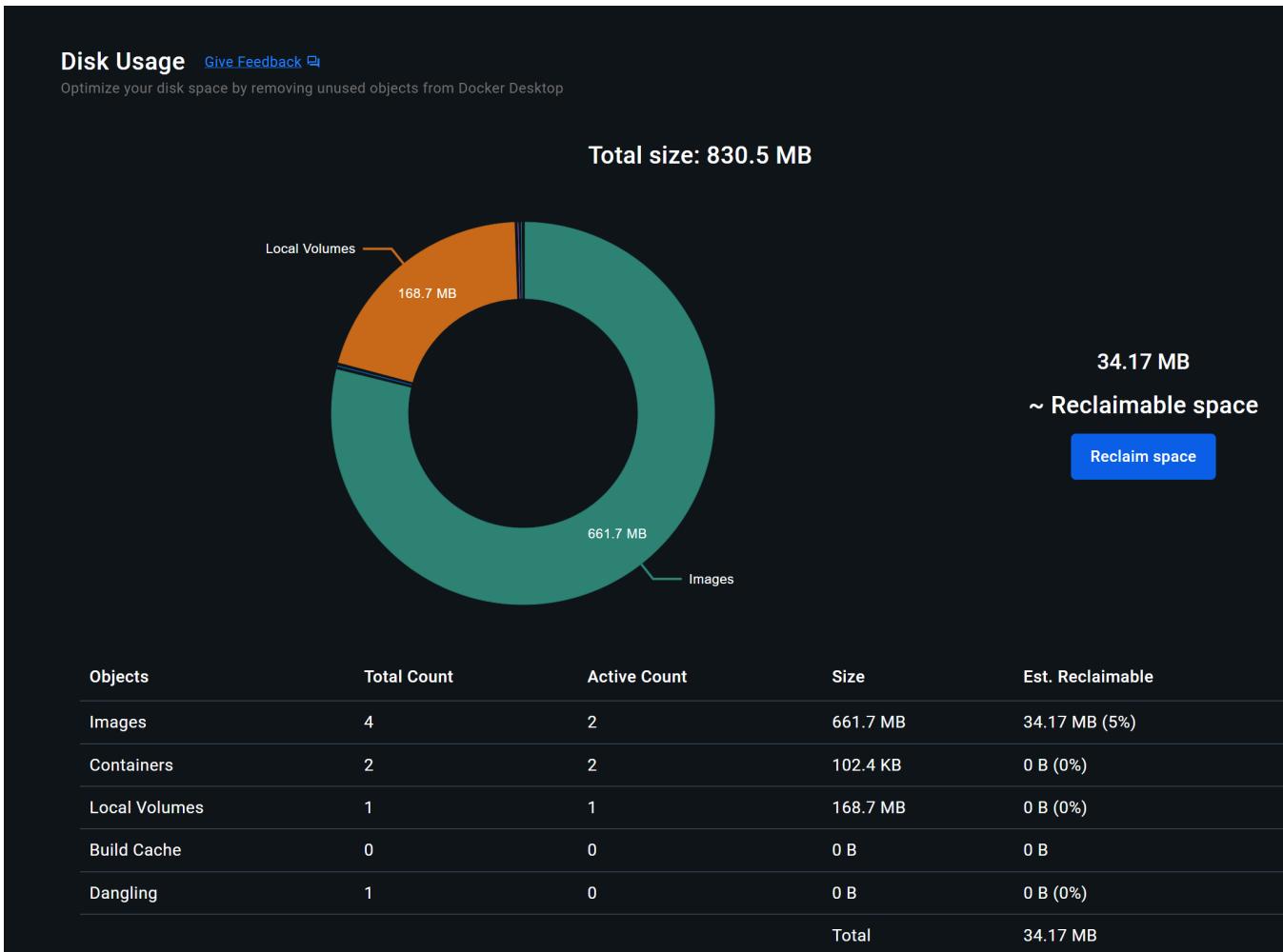
1 result(s) for "disk usage"

**Disk Usage** Docker Inc. · docker/disk-usage-extension:0.2.9

Optimize your disk space by removing unused objects from Docker Desktop.

Reviewed 390.0K    Install

Una vez instalada, podemos ver cuánto espacio ocupan las imágenes, contenedores, etc.:



## 7. Borrado de los contenedores y red creados

Como nos piden borrar los contenedores y la red creados (no hemos creado ningún volumen), los eliminaremos de uno en uno para ir viendo el efecto de ese borrado en el espacio utilizado (el borrado se gestiona con los botones con forma de papelera que aparecen en el apartado **acciones** de cada contenedor):

**Containers** [Give feedback](#)

View all your running containers and applications. [Learn more](#)

Container CPU usage ⓘ  
0.04% / 1200% (12 CPUs available)

Container memory usage ⓘ  
292.49MB / 7.49GB

Show charts

Search  Filter  Only show running containers

| <input type="checkbox"/> | Name                 | Container ID | Image          | Port(s)     | CPU (%) | Last started   | Actions  |
|--------------------------|----------------------|--------------|----------------|-------------|---------|----------------|--|
| <input type="checkbox"/> | mariadb_egl33817_ej1 | 877144cebd34 | mariadb:latest | 3306:3306 ↗ | 0.03%   | 6 hours ago    | <input type="checkbox"/> ⋮ <span style="border: 2px solid red; padding: 2px;">trash</span>   |
| <input type="checkbox"/> | adminer_egl33817_ej1 | 109337b40696 | adminer:latest | 8080:8080 ↗ | 0.01%   | 45 minutes ago | <input type="checkbox"/> ⋮ <span style="border: 2px solid green; padding: 2px;">trash</span> |

Delete container?

The 'mariadb\_egl33817\_ej1' container is selected for deletion. Any anonymous volumes associated with this container are also deleted.

Cancel Delete forever

Tras confirmar el borrado, podemos ver que el espacio usado es menor:

**Disk Usage** [Give Feedback](#)

Optimize your disk space by removing unused objects from Docker Desktop

Total size: 661.78 MB

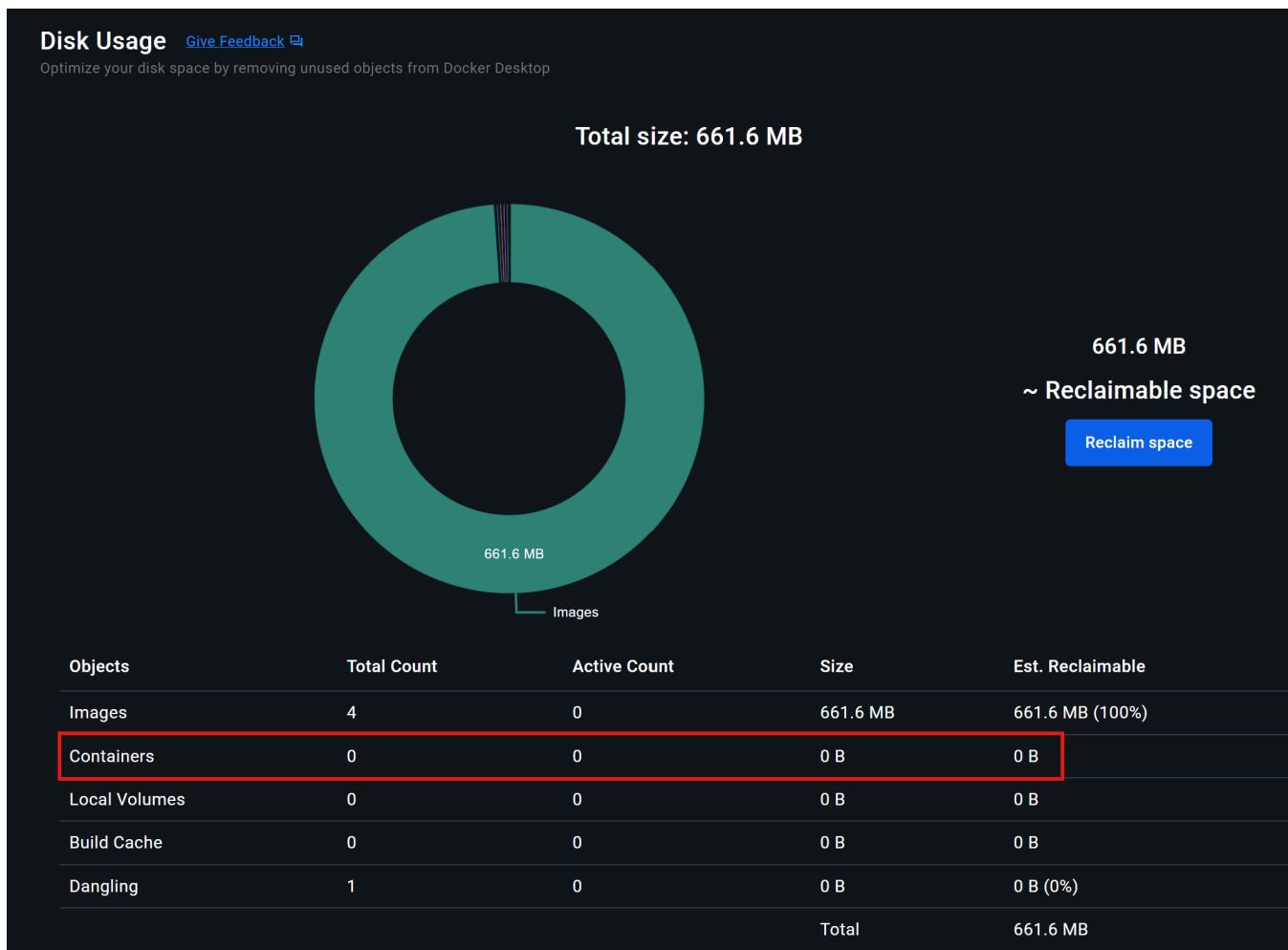
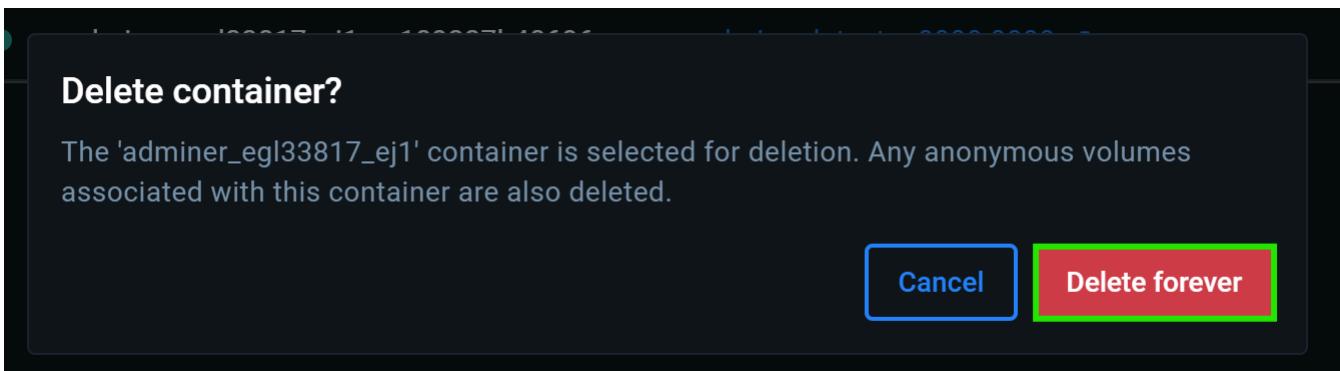
661.7 MB

487.9 MB

~ Reclaimable space

Reclaim space

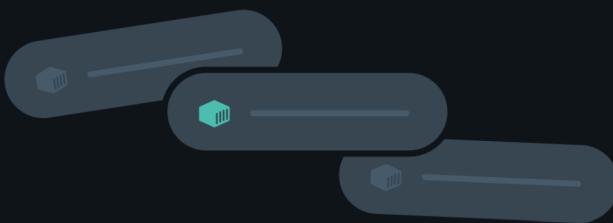
| Objects       | Total Count | Active Count | Size     | Est. Reclaimable |
|---------------|-------------|--------------|----------|------------------|
| Images        | 4           | 1            | 661.7 MB | 487.9 MB (73%)   |
| Containers    | 1           | 1            | 77.82 KB | 0 B (0%)         |
| Local Volumes | 0           | 0            | 0 B      | 0 B              |
| Build Cache   | 0           | 0            | 0 B      | 0 B              |
| Dangling      | 1           | 0            | 0 B      | 0 B (0%)         |
|               |             |              | Total    | 487.9 MB         |



Ya no hay ningún contenedor:

## Containers [Give feedback](#)

View all your running containers and applications. [Learn more](#)



### Your running containers show up here

A container is an isolated environment for your code



What is a container?

5 mins

```
1 FROM node
2 RUN mkdir -p /app
3 WORKDIR /app
4 COPY packa|
```

How do I run a container?

6 mins

[View more in the Learning center](#)

La red se borra haciendo clic en el botón resaltado en la siguiente imagen:

**Network: redej1**

Driver: bridge    Gateway: 172.18.0.1    Subnet: 172.18.0.0/16

Connected Containers | 0    Add Container

**Network: bridge**

Driver: bridge    Gateway: 172.17.0.1    Subnet: 172.17.0.0/16

Connected Containers | 0    Add Container

**Network: host**

Driver: host    Gateway: null    Subnet: null

Connected Containers | 0    Add Container

**Network: none**

Driver: null    Gateway: null    Subnet: null

Connected Containers | 0    Add Container

✓ Successfully deleted Network! ✎

Y con esto finaliza la realización de este primer ejercicio con **Docker Desktop**.