

# **Actividad Evaluable 3**

---

## **Docker**

---

**Ejercicio 1 - Contenedores en red y Docker Desktop**

**Despliegue de Aplicaciones Web - DAW Distancia**

**CIFP Sect. Industrial y Servicios - La Laboral**

**Curso 2024-2025**

**01 de abril de 2025**

**Emilio Zaera Vidal - 46.911.234-C**

1. Introducción
2. Metodología
3. Preparativos
  - Creación de un nuevo repositorio `tareaDocker`
  - Clonado y vinculación en local
  - Creación de las ramas
4. Ejercicio 1 - Contenedores en red y Docker Desktop
  - Creación red bridge `redej1`
  - Creación contenedor de `mariadb`
    - Script creación de tabla `modulos`
  - Creación de contenedor con `Adminer`
  - Conexión de los contenedores a la red `redej1`
  - Conexión a la BD con Adminer
  - Ejecución del script SQL desde la GUI de Adminer
  - Instalación de Disk Usage
  - Borro los contenedores y la red

## 1. Introducción

---

En el módulo de **Despliegue de Aplicaciones Web**, uno de los objetivos fundamentales es aprender a gestionar entornos de despliegue modernos utilizando tecnologías basadas en contenedores. En este contexto, la herramienta **Docker** se ha convertido en un estándar para la creación, configuración y administración de entornos aislados, facilitando el despliegue y la distribución de aplicaciones web.

La presente tarea evaluable tiene como finalidad reforzar los conocimientos adquiridos sobre Docker mediante la realización de tres ejercicios prácticos. A través de estos ejercicios, se trabajará la creación de contenedores en red, la orquestación de servicios y la construcción de una imagen personalizada. Todo ello permitirá al alumno familiarizarse con el ciclo completo de creación, despliegue y gestión de contenedores, así como con las buenas prácticas de documentación y organización de proyectos en un repositorio.

La **orquestación de servicios** es el proceso de coordinar y gestionar varios contenedores para que funcionen juntos como una única aplicación. En Docker, se realiza con herramientas como **Docker Compose**, que permiten definir y automatizar la configuración y despliegue de todos los servicios desde un solo archivo.

---

## 2. Metodología

---

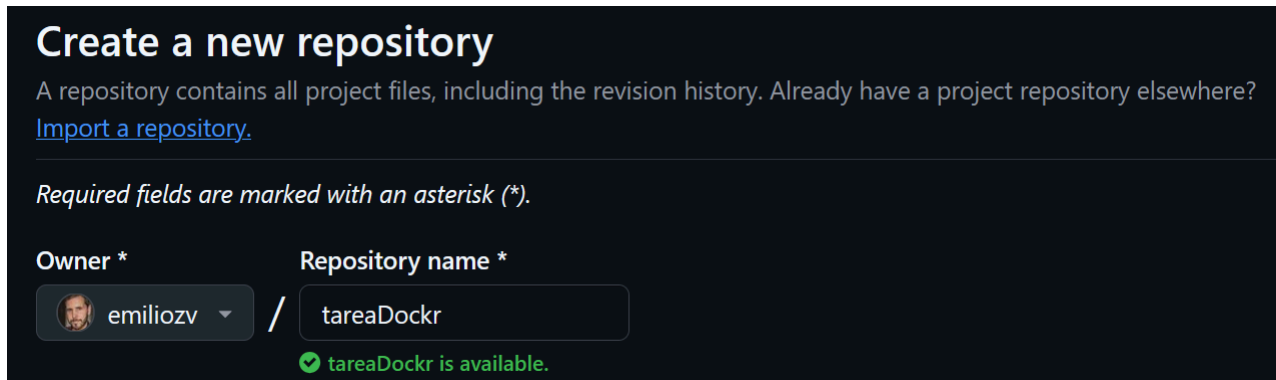
La metodología propuesta para esta tarea consiste en la resolución práctica de tres ejercicios diferenciados. En cada uno de ellos, se deberá crear un entorno utilizando distintas herramientas de Docker: redes y contenedores mediante Docker Desktop, despliegue con Docker Compose y creación de una imagen personalizada con Dockerfile. Todo el trabajo se documentará y organizará en un repositorio público de

GitHub, utilizando ramas para cada ejercicio. Además, como parte de la evaluación, se solicita un videoclip donde el estudiante muestre y explique parte del trabajo realizado.

### 3. Preparativos

#### Creación de un nuevo repositorio tareaDocker


Creo un nuevo repositorio público en [mi GitHub](#) para la tarea:



**Create a new repository**

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

*Required fields are marked with an asterisk (\*).*

**Owner \***  **Repository name \***

emiliozv / tareaDocker

✓ tareaDocker is available.

#### Clonado y vinculación en local

Trabajaré en local para, al finalizar, subir todo al repositorio remoto en GitHub. Para ello, creo un repositorio local mediante la línea de comandos de git, genero las carpetas y los ficheros `.md`, y lo vinculo con el remoto:

```
EyM@Sobremesa MINGW64 /d/TRABAJO/Formación/DAW/02 CURSO/Despliegue/Tareas/Evaluables/T3 Docker/repoTareaDocker
$ git init
Initialized empty git repository in D:/TRABAJO/Formación/DAW/02 CURSO/Despliegue/Tareas/Evaluables/T3 Docker/repoTareaDocker/.git/

EyM@Sobremesa MINGW64 /d/TRABAJO/Formación/DAW/02 CURSO/Despliegue/Tareas/Evaluables/T3 Docker/repoTareaDocker (master)
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)

EyM@Sobremesa MINGW64 /d/TRABAJO/Formación/DAW/02 CURSO/Despliegue/Tareas/Evaluables/T3 Docker/repoTareaDocker (main)
$ git commit -m "Creación de carpetas"
[main (root-commit) 4be2a88] Creación de carpetas
 4 files changed, 170 insertions(+)
 create mode 100644 Ejercicio1/tarea_evaluable_3_docker_ej1_Emilio_Zaera_Vidal.assets/image-20250401103607546.png
 create mode 100644 Ejercicio1/tarea_evaluable_3_docker_ej1_Emilio_Zaera_Vidal.md
 create mode 100644 Ejercicio2/tarea_evaluable_3_docker_ej2_Emilio_Zaera_Vidal.md
 create mode 100644 Ejercicio3/tarea_evaluable_3_docker_ej3_Emilio_Zaera_Vidal.md

EyM@Sobremesa MINGW64 /d/TRABAJO/Formación/DAW/02 CURSO/Despliegue/Tareas/Evaluables/T3 Docker/repoTareaDocker (main)
$ git branch -M main
```

```
EyM@Sobremesa MINGW64 /d/TRABAJO/Formación/DAW/02 CURSO/Despliegue/Tareas/Evaluables/T3 Docker/repoTareaDocker (main)
$ git remote add origin https://github.com/emiliozv/tareaDocker.git

EyM@Sobremesa MINGW64 /d/TRABAJO/Formación/DAW/02 CURSO/Despliegue/Tareas/Evaluables/T3 Docker/repoTareaDocker (main)
$ git remote -v
origin https://github.com/emiliozv/tareaDocker.git (fetch)
origin https://github.com/emiliozv/tareaDocker.git (push)

EyM@Sobremesa MINGW64 /d/TRABAJO/Formación/DAW/02 CURSO/Despliegue/Tareas/Evaluables/T3 Docker/repoTareaDocker (main)
$ git push -u origin main
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 32 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (10/10), 57.28 KiB | 19.09 MiB/s, done.
Total 10 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/emiliozv/tareaDocker.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

The screenshot shows the GitHub interface for the repository 'tareaDocker'. At the top, there's a header with the repository name, a 'Public' badge, and buttons for 'Pin' and 'Unwatch'. Below this, there's a navigation bar with 'main' selected, '1 Branch', and '0 Tags'. A search bar 'Go to file' and buttons 'Add file' and 'Code' are also present. The main content area shows a commit by 'emiliozv' titled 'Creación de carpetas' with a commit hash '4be2a88' and '5 minutes ago'. Below the commit, there's a table listing three files: 'Ejercicio1', 'Ejercicio2', and 'Ejercicio3', each with a description 'Creación de carpetas' and a timestamp '5 minutes ago'.

## Creación de las ramas

Creo las 3 ramas y cambio a ellas cuando lo necesite. Ejemplo con rama `ejercicio1`:

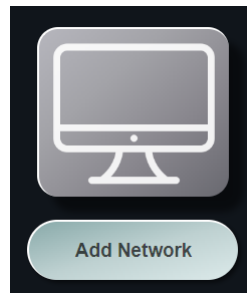
```
git branch ejercicio1
git switch ejercicio1
```

## 4. Ejercicio 1 - Contenedores en red y Docker Desktop

### Creación red bridge `redej1`

Este ejercicio se realiza desde Docker Desktop

En Docker Desktop instalamos previamente la extensión `PortNavigator`



X

Add Network

Name: redej1

☐ Advanced network settings

Subnetwork: Add a subnetwork (optional)

Gateway: Add a gateway (optional)

IP-Range: Add an IP-Range (optional)

Create Network

Network: redej1

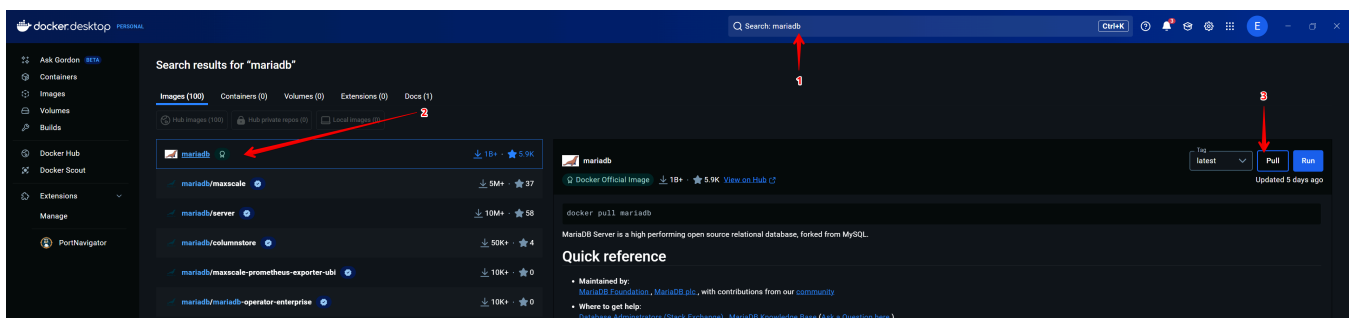
Driver: bridgeGateway: 172.18.0.1Subnet: 172.18.0.0/16

Connected Containers | 0

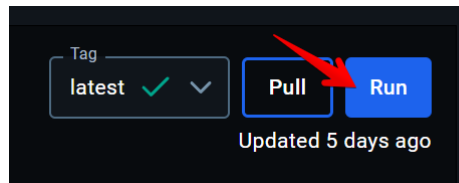
Add Container

## Creación contenedor de mariadb

Primero busco la imagen oficial de `mariadb` en el buscador de Docker Desktop y la descargo pulsando sobre el botón `Pull`:



A continuación creo el contenedor pulsando sobre `Run`:



Podemos ver esta ventana de ajustes opcionales, donde introduciremos la información solicitada en el enunciado:

**Run a new container**  
mariadb:latest

**Optional settings** ^

Container name

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

**Ports**  
Enter "0" to assign randomly generated host ports.

Host port :3306/tcp

**Volumes**

Host path ... Container path +

**Environment variables**

Variable Value +

Cancel Run

Para ello tendremos que encontrar en la documentación de la imagen la información necesaria:

- Definición de contraseña para el usuario root:

## Environment Variables

When you start the `mariadb` image, you can adjust the initialization of the MariaDB instance by passing one or more environment variables on the `docker run` command line. Do note that all of the variables, except `MARIADB_AUTO_UPGRADE`, will have no effect if you start the container with a data directory that already contains a database. I.e. any pre-existing database will always be left untouched on container startup.

One of `MARIADB_RANDOM_ROOT_PASSWORD`, `MARIADB_ROOT_PASSWORD_HASH`, `MARIADB_ROOT_PASSWORD` or `MARIADB_ALLOW_EMPTY_ROOT_PASSWORD` (or equivalents, including `*_FILE`), is required. The other environment variables are optional.

There is a large list of environment variables and the complete list is documented on [MariaDB's Knowledge Base : MariaDB Server Docker Official Image Environment Variables](#).

El nombre de la variable de entorno es `MARIADB_ROOT_PASSWORD`, para el valor (la contraseña) estableceré "root".

- Definir un usuario con mi nombre de pila y con contraseña:

## MARIADB\_USER / MYSQL\_USER, MARIADB\_PASSWORD\_HASH / MARIADB\_PASSWORD / MYSQL\_PASSWORD

Both user and password variables, along with a database, are required for a user to be created. This user will be granted all access (corresponding to GRANT ALL) to the MARIADB\_DATABASE database.

Do not use this mechanism to create the root superuser, that user gets created by default with the password specified by the MARIADB\_ROOT\_PASSWORD / MYSQL\_ROOT\_PASSWORD variable.


- Nombre de usuario: El nombre de la variable de entorno es `MARIADB_USER`. Su valor será "Emilio"
  - Contraseña: El nombre de la variable de entorno es `MARIADB_PASSWORD`. Su valor será "1234"
- Nombre de la BD por defecto será `DAW`:

## MARIADB\_DATABASE / MYSQL\_DATABASE

This variable allows you to specify the name of a database to be created on image startup.

- Nombre de la BD: El nombre de la variable de entorno es `MARIADB_DATABASE`. Su valor será "DAW"

Con la información anterior, queda de la siguiente manera:



Run a new container

mariadb:latest

Optional settings

Container name

mariadb\_ej1

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

Ports

Enter "0" to assign randomly generated host ports.

Host port

3306

:3306/tcp

Volumes

Host path

...

Container path

+

Environment variables

Variable

MARIADB\_ROOT\_PASSWORD

Value

root

—

Variable

MARIADB\_USER

Value

Emilio

—

Variable

MARIADB\_PASSWORD

Value

1234

—

Variable

MARIADB\_DATABASE

Value

DAW

+

Cancel

Run

Containers

[Give feedback](#)

View all your running containers and applications.

[Learn more](#)

Container CPU usage


0.01% / 3200% (32 CPUs available)

Container memory usage

302.7MB / 30.58GB

Show charts

Search



☒ Only show running containers

<input type="checkbox"/>	Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
<input type="checkbox"/>	<div><div></div>mariadb_ej1</div>	2d7d354ea20f	<a href="#">mariadb:latest</a>	<a href="#">3306:3306</a>	0.01%	38 seconds ago	<div><div></div><div></div><div></div></div>

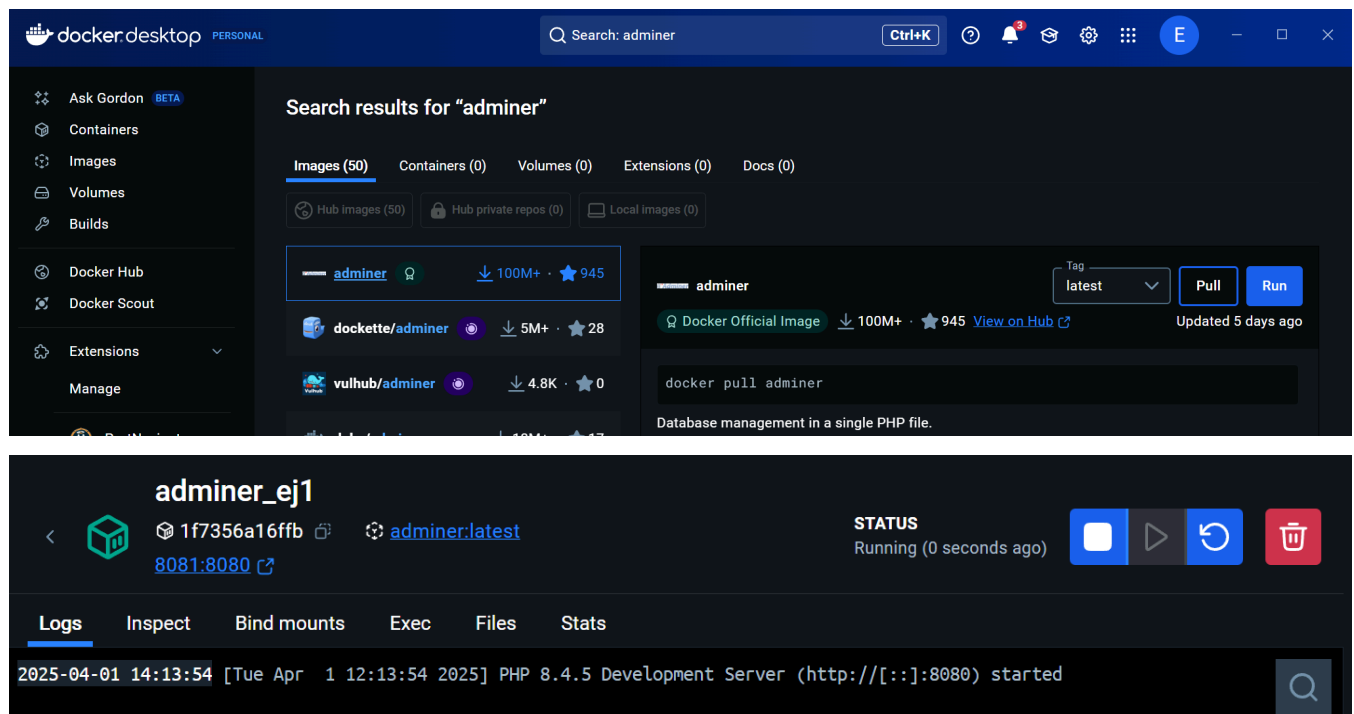


## Script creación de tabla `modulos`

```
CREATE TABLE modulos (  
  id INT AUTO_INCREMENT PRIMARY KEY,  
  nombre VARCHAR(50) NOT NULL  
);  
  
INSERT INTO modulos (nombre) VALUES  
('cliente'),  
('servidor'),  
('interfaces'),  
('despliegue'),  
('empresa'),  
('proyecto'),  
('prácticas');
```

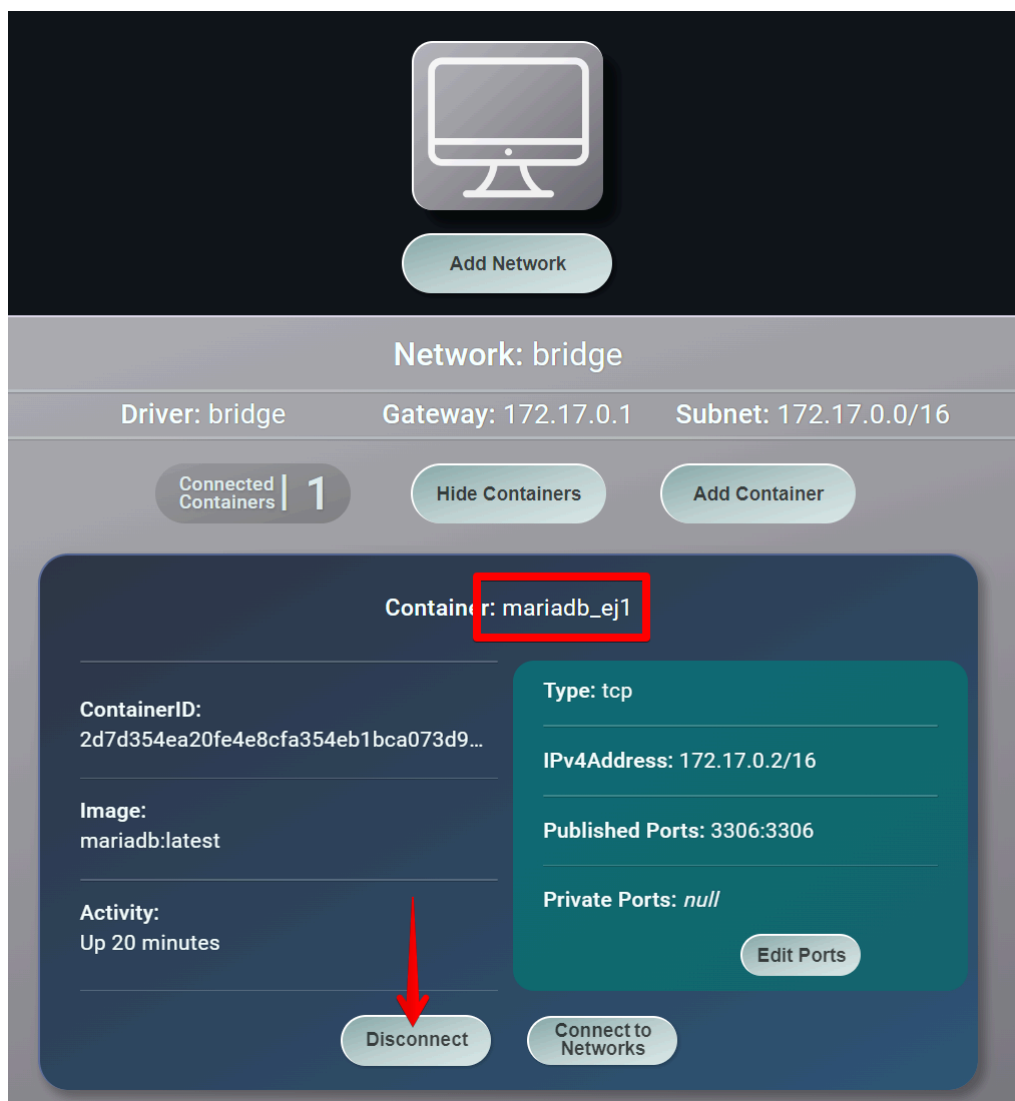
## Creación de contenedor con `Adminer`

Introduzco "adminer" en el buscador de Docker Desktop y hago un `Pull` de la imagen, para luego crear el contenedor mediante `Run`:

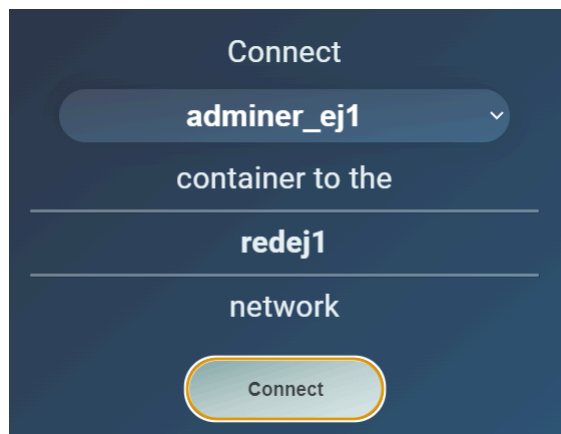


## Conexión de los contenedores a la red redej1

Primero desconecto de la red bridge por defecto (muestro mariadb\_ej1):



A continuación, conecto ambos contenedores a la red redej1 (muestro adminer\_ej1):



Resultado de la configuración de red:

Network: bridge

Driver: bridge

Gateway: 172.17.0.1

Subnet: 172.17.0.0/16

Connected Containers | 0

Add Container

Network: redej1

Driver: bridge

Gateway: 172.18.0.1

Subnet: 172.18.0.0/16

Connected Containers | 2


Show Containers

Add Container


Compruebo mediante ping la conexión de adminer con mariadb:

adminer\_ej1

<



1f7356a16ffb



adminer:latest

8081:8080

Logs

Inspect

Bind mounts

Exec

Files

Stats

```

/var/www/html $ ifconfig
eth0      Link encap:Ethernet  HWaddr BA:E6:5A:C0:43:82
          inet addr:172.18.0.3  Bcast:172.18.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:10  errors:0  dropped:0  overruns:0  frame:0
          TX packets:3  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0  txqueuelen:0
          RX bytes:872 (872.0 B)  TX bytes:126 (126.0 B)

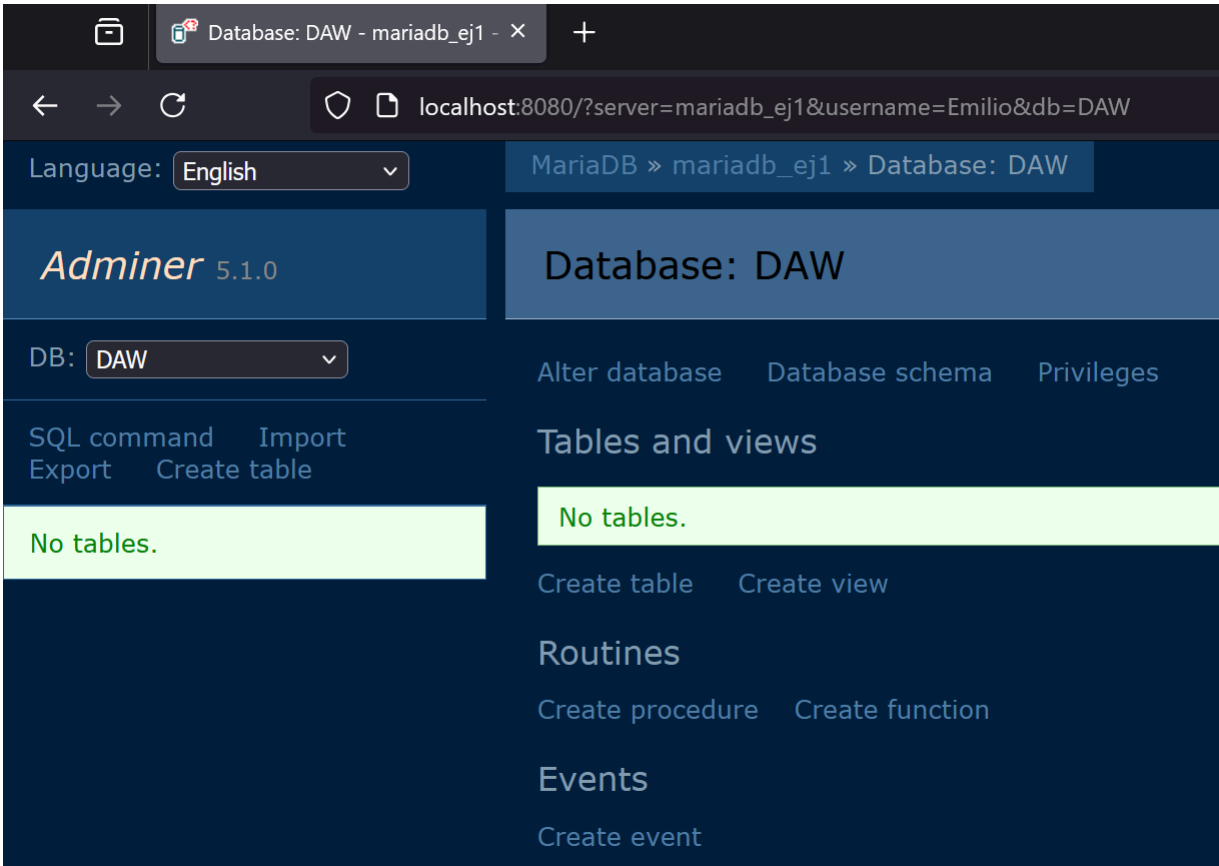
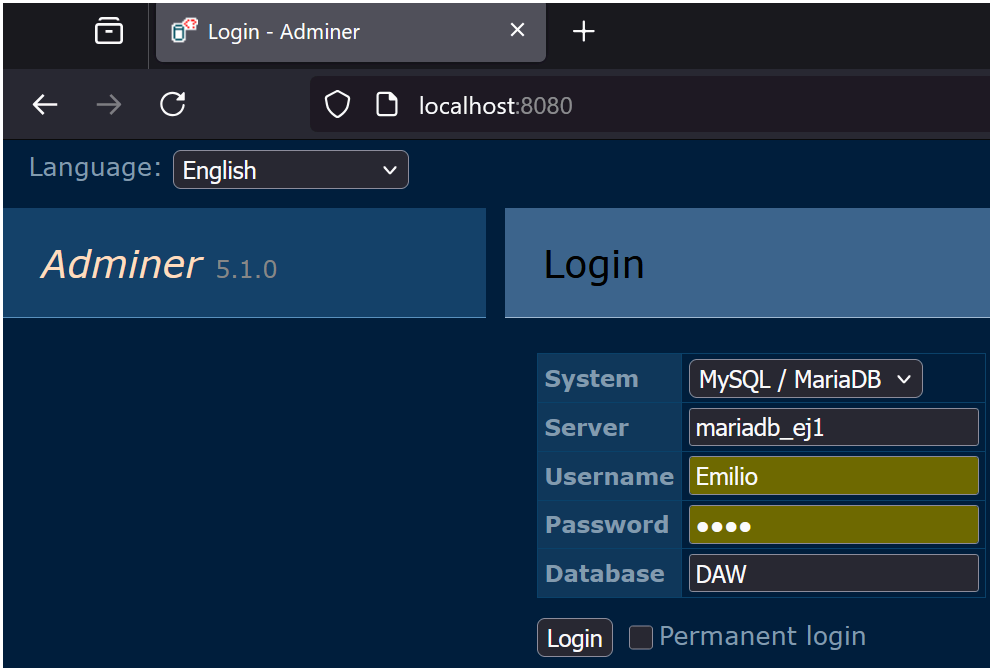
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0  errors:0  dropped:0  overruns:0  frame:0
          TX packets:0  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0  txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

/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=0.075 ms
64 bytes from 172.18.0.2: seq=1 ttl=42 time=0.058 ms
64 bytes from 172.18.0.2: seq=2 ttl=42 time=0.056 ms
64 bytes from 172.18.0.2: seq=3 ttl=42 time=0.059 ms
^C
--- 172.18.0.2 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.056/0.062/0.075 ms
/var/www/html $

```

# Conexión a la BD con Adminer

```
http://localhost:8088
```



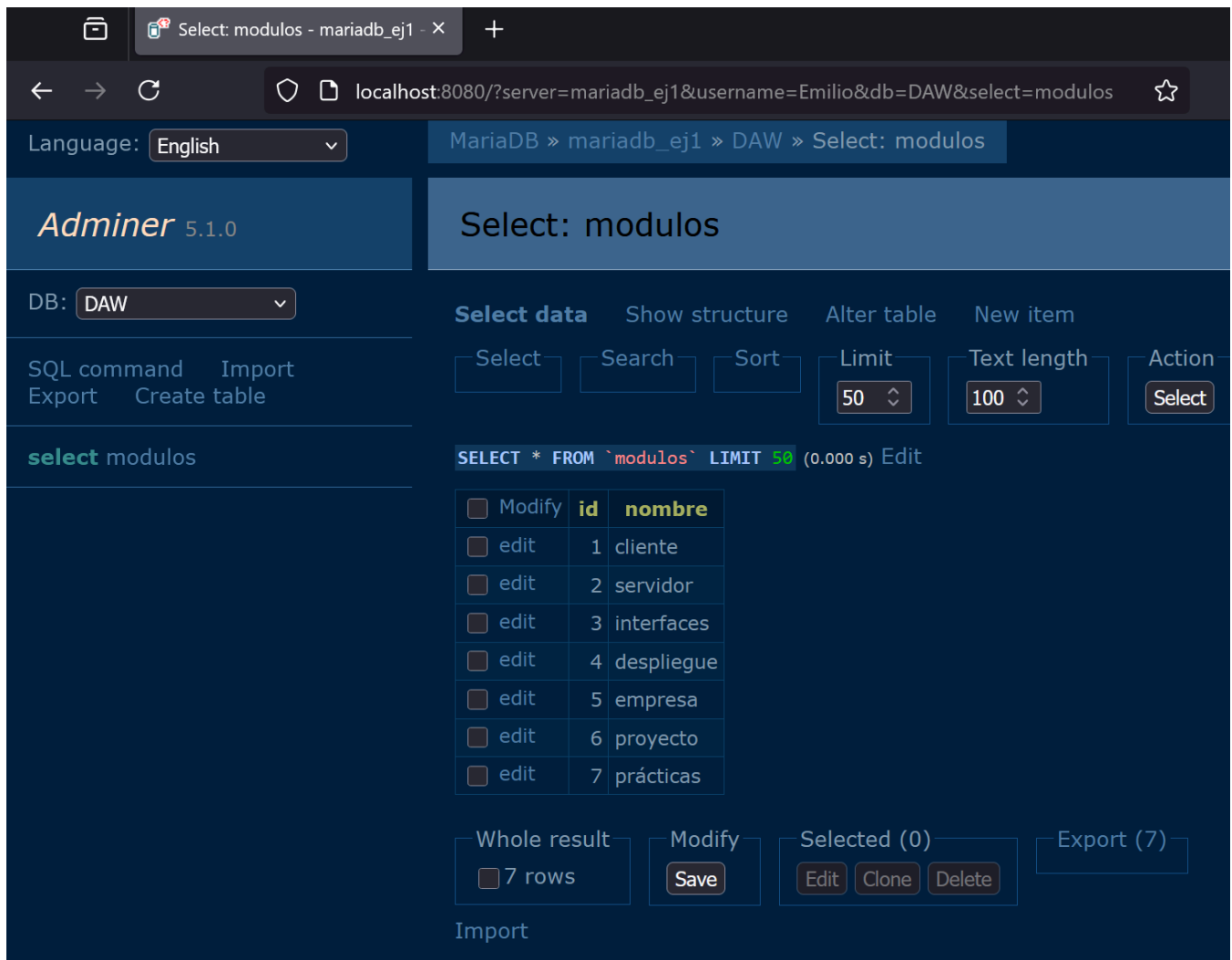
# Ejecución del script SQL desde la GUI de Adminer

Creación de la tabla `modulos` mediante el script (SQL command):

The screenshot shows the Adminer 5.1.0 web interface. The browser address bar shows `localhost:8080/?server=mariadb_ej1&username=Emilio&db=DAW&sql=CREATE TABLE`. The interface has a sidebar on the left with the Adminer logo, a language dropdown set to 'English', a database selector set to 'DAW', and buttons for 'SQL command', 'Import', 'Export', and 'Create table'. The 'SQL command' button is active. The main panel is titled 'SQL command' and contains two SQL queries. The first query is `CREATE TABLE modulos ( id INT AUTO_INCREMENT PRIMARY KEY, nombre VARCHAR(50) NOT NULL );`, followed by a green status bar indicating 'Query executed OK, 0 rows affected. (0.018 s) Edit'. The second query is `INSERT INTO modulos (nombre) VALUES ('cliente'), ('servidor'), ('interfaces'), ('despliegue'), ('empresa'), ('proyecto'), ('prácticas');`, followed by a green status bar indicating 'Query executed OK, 7 rows affected. (0.002 s) Edit'. At the bottom, there is an 'Execute' button, a 'Limit rows' dropdown, and checkboxes for 'Stop on error' and 'Show only errors'.

```
CREATE TABLE modulos (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    nombre VARCHAR(50) NOT NULL  
)  
  
Query executed OK, 0 rows affected. (0.018 s) Edit  
  
INSERT INTO modulos (nombre) VALUES  
(  
    ('cliente'),  
    ('servidor'),  
    ('interfaces'),  
    ('despliegue'),  
    ('empresa'),  
    ('proyecto'),  
    ('prácticas')  
)  
  
Query executed OK, 7 rows affected. (0.002 s) Edit  
  
CREATE TABLE modulos (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    nombre VARCHAR(50) NOT NULL  
);  
  
INSERT INTO modulos (nombre) VALUES  
(  
    ('cliente'),  
    ('servidor'),  
    ('interfaces'),  
    ('despliegue'),  
    ('empresa'),  
    ('proyecto'),  
    ('prácticas');  
)
```

Muestra de los datos contenidos en la tabla (SELECT modulos):



Adminer 5.1.0

Language: English

DB: DAW

SQL command: Import Export Create table

Select: modules

Select data Show structure Alter table New item

Select Search Sort Limit 50 Text length 100 Action Select

SELECT \* FROM `modulos` LIMIT 50 (0.000 s) Edit

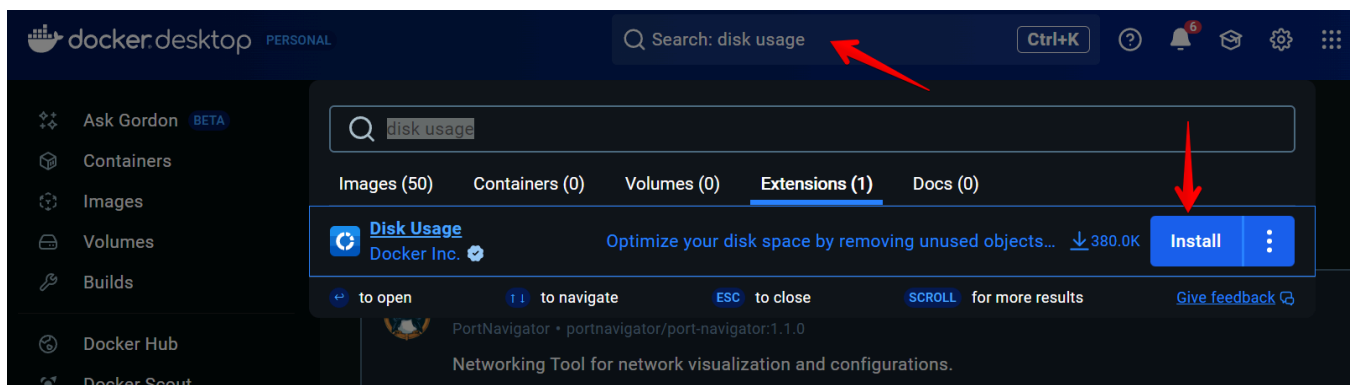
Modify	id	nombre
edit	1	cliente
edit	2	servidor
edit	3	interfaces
edit	4	despliegue
edit	5	empresa
edit	6	proyecto
edit	7	prácticas

Whole result 7 rows Modify Save Selected (0) Edit Clone Delete Export (7)

Import

## Instalación de Disk Usage

Lo busco en el buscador de Docker Desktop y filtro por extensiones:



docker:desktop PERSONAL

Search: disk usage

disk usage

Images (50) Containers (0) Volumes (0) Extensions (1) Docs (0)

Disk Usage Docker Inc. Optimize your disk space by removing unused objects... 380.0K Install

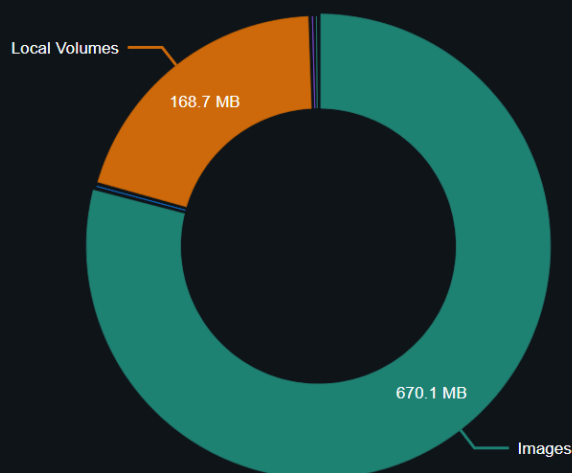
to open to navigate ESC to close SCROLL for more results Give feedback

Muestro el espacio ocupado:

## Disk Usage [Give Feedback](#)

Optimize your disk space by removing unused objects from Docker Desktop

Total size: 838.93 MB



32.61 MB  
~ Reclaimable space

Reclaim space

Objects	Total Count	Active Count	Size	Est. Reclaimable
Images	4	2	670.1 MB	32.61 MB (4%)
Containers	2	2	131.1 KB	0 B (0%)
Local Volumes	1	1	168.7 MB	0 B (0%)
Build Cache	0	0	0 B	0 B
Dangling	1	0	0 B	0 B (0%)
			Total	32.61 MB

Borro el contenedor `mariadb_ej1` y compruebo la diferencia:

### Delete container?

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

Cancel

Delete forever



## Borro los contenedores y la red

### Containers [Give feedback](#)

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

Container CPU usage ⓘ  
No containers are running.

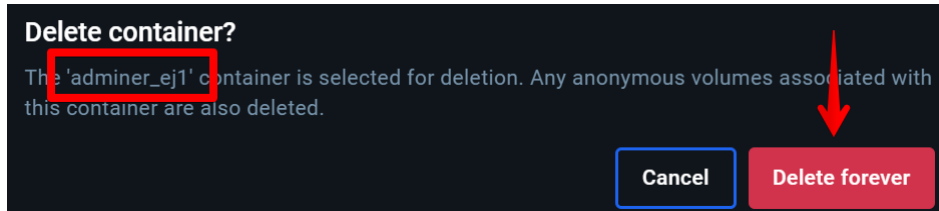
Container memory usage ⓘ  
No containers are running.

Show charts

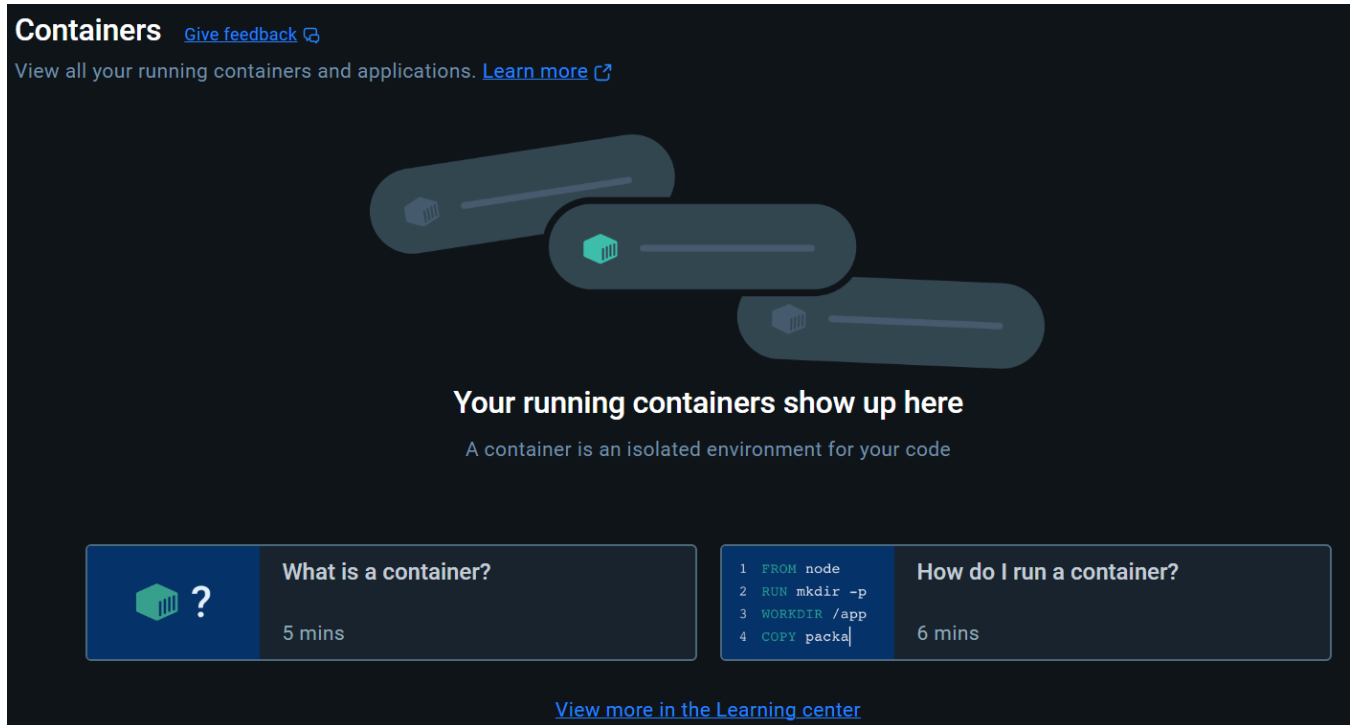
☐ Only show running containers

<input type="checkbox"/>	Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
<input type="checkbox"/>	adminer_ej1	61ac6caeb2e3	<a href="#">adminer:lat</a>	8080:8080	N/A	16 minutes a	





Resultado, ningún contenedor:



Borrado de la red redej1:

