

# Mi primer interacción con Docker

Fase	Semana	Día	Lección
6 - Bases de Datos	3	2	3

# Explorando el GUI de Docker Desktop - Containers

Lista de Contenedores presentes en el sistema

Containers

Images

Volumes

Dev Environments BETA

Docker Scout EARLY ACCESS

Learning center

Extensions

Containers [Give feedback](#)


Container CPU usage ⓘ  
*No containers are running.*

Container memory usage ⓘ  
*No containers are running.*

Show charts ▾

☐

Only show running containers

☐	Name	Image	Status	Port(s)	Last started	CPU (%)	Actions
☐	 <a href="#">competent</a> 1f52c238544	<a href="#">hello-world</a>	Exited		17 minutes ago	N/A	<div>▶ ⋮ 🗑</div>

# Explorando el GUI de Docker Desktop - Containers

Al dar click sobre el contenedor (link azul **competent**) podremos ver más información relacionado al contenedor como los logs, inspeccionar información del contenedor corriendo, archivos del contenedor, ...

The screenshot displays the Docker Desktop interface. On the left is a sidebar with navigation options: Containers, Images, Volumes, Dev Environments (marked BETA), Docker Scout (marked EARLY ACCESS), Learning center, Extensions, and Add Extensions. The main panel shows details for a container named 'competent\_varahamihira' (ID: 1f52c2385449). The container's status is 'Exited (0) (2 minutes ago)'. Below the status bar are tabs for Logs, Inspect, Bind mounts, Exec, Files, and Stats. The 'Logs' tab is active, showing a series of log entries with timestamps and messages. On the right side of the logs, there are icons for search, copy, refresh, and delete.

**competent\_varahamihira**

hello-world  
1f52c2385449

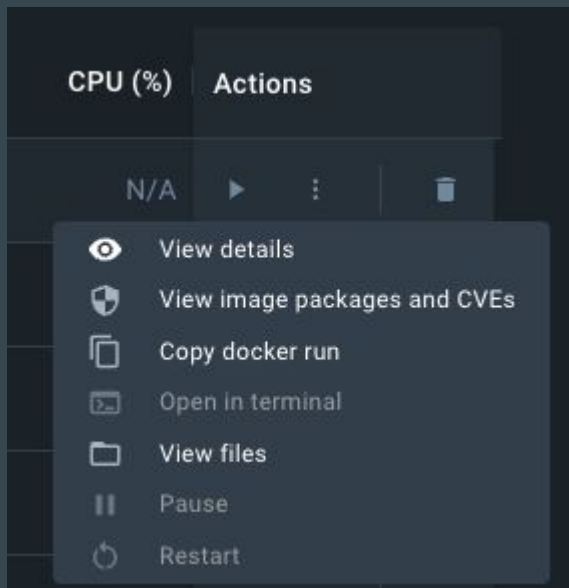
STATUS  
Exited (0) (2 minutes ago)

Logs Inspect Bind mounts Exec Files Stats

2023-09-17 14:36:44 3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading.  
2023-09-17 14:36:44 4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal.  
2023-09-17 14:36:44 To try something more ambitious, you can run an Ubuntu container with:  
2023-09-17 14:36:44 \$ docker run -it ubuntu bash  
2023-09-17 14:36:44 Share images, automate workflows, and more with a free Docker ID:  
2023-09-17 14:36:44 <https://hub.docker.com/>  
2023-09-17 14:36:44 For more examples and ideas, visit:  
2023-09-17 14:36:44 <https://docs.docker.com/get-started/>  
2023-09-17 14:36:48 Hello from Docker!  
2023-09-17 14:36:48 This message shows that your installation appears to be working correctly.

# Explorando el GUI de Docker Desktop - Containers


## Acciones en los contenedores





De es posible realizar las siguientes acciones sobre cada contenedor


- Ejecutar (botón de play)
- Ver detalles
- Copiar comando para ejecutar contenedor en la terminal
- Pausar
- Reiniciar


# Explorando el GUI de Docker Desktop - Learning Center


 Containers


 Images


 Volumes

 Dev Environments BETA

 Docker Scout EARLY ACCESS

 Learning center


Extensions 

 Add Extensions

## Learning center [Give feedback](#)

### Walkthroughs

Quick hands-on guides to show you around

 **What is a container?**  
5 mins


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


**How do I run a container?**  
6 mins


[View all](#)


### Docker for beginners by language


45 min guides written for different programming languages


 NodeJS

 Python

 Go

 Java

 C# (.NET)

 Rust

[Request a guide](#)

### Samples

Popular starter projects for you to test out

**React, ExpressJS, and MongoDB**  
Simple todo list application

**Spring & Postgres**  
Java application with Spring framework

# Comandos básicos de Docker

`docker pull nombre_de_la_imagen`

Descarga una imagen de Docker desde un repositorio.

`docker images`

Lista las imágenes de Docker que se han descargado en nuestro sistema.

`docker rmi nombre_de_la_imagen`

Elimina una imagen de Docker del sistema.

`docker build -t nombre_de_la_imagen ruta_del_contexto`

Crea una imagen de Docker a partir de un archivo `Dockerfile` y un contexto.

`docker create nombre_de_la_imagen`

Crea un nuevo contenedor en base a una imagen (ya sea que la imagen exista o no en nuestro sistema)

`docker create --name nombre_nuevo_del_contenedor nombre_de_la_imagen`

Crea un nuevo contenedor asignándole un nombre específico basado en la imagen indicada

`docker run --name nombre_custom_para_contenedor nombre_de_la_imagen`

Ejecuta un contenedor a partir de una imagen.

`docker ps`

Lista los contenedores en ejecución.

`docker ps -a`

Lista todos los contenedores (ya sea en ejecución o no).

`docker stop ID_del_contenedor`

Detiene un contenedor en ejecución (por ID).

`docker rm ID_del_contenedor`

Elimina un contenedor usando el ID.

`docker exec -it ID_del_contenedor | nombre_del_contenedor comando`

Ejecuta un comando dentro de un contenedor en ejecución. Se utiliza para interactuar con contenedor



`docker login`

Se utiliza para iniciar sesión en docker hub

`docker push nombre_del_repositorio/nombre_del_contenedor`

Subir imagen docker al repositorio indicado con nombre indicado al docker hub

`docker volume create`

Crear un volumen para almacenar datos

`docker volume ls`

Lista los volúmenes existentes

`docker cp file nombre_o_ID_del_contenedor:directorio_dentro contenedor directorio_en_computadora_local`

Copia archivo desde el contenedor a nuestra computadora

`docker logs ID_del_contenedor | nombre_del_contenedor`

Muestra los logs del contenedor indicado

Para acceder a la lista completa de comandos con cada una de sus opciones / flags visitar la documentación oficial: <https://docs.docker.com/engine/reference/commandline/cli/>

# Ejercicio comandos básicos

El ejercicio consiste en aplicar algunos de los comandos que se vieron anteriormente.

## Pasos del ejercicio

- Hacer pull de la imagen **alpine** del Docker Hub ([https://hub.docker.com/\\_/alpine](https://hub.docker.com/_/alpine))
- Listar las imágenes para saber que si se encuentra en nuestra computadora
- Ejecute un contenedor de manera interactiva (-it) usando la imagen alpine pero con el nombre de alpine-test. Esto hará que ingresemos dentro del contenedor

```
docker run --name alpine-test -it alpine
```

- Abra otra terminal y corra el comando para listar todos los contenedores corriendo
- Tome el ID del contenedor que está corriendo
- Detenga el contenedor
- Elimine el contenedor del sistema
- Elimine la imagen del sistema