

# Docker

Christian Mendoza

16 de noviembre de 2023

## Resumen

Docker Desktop, una extensión de la plataforma de contenerización Docker, revoluciona el desarrollo y despliegue de aplicaciones al proporcionar un entorno homogéneo y aislado para los desarrolladores en sistemas Windows y macOS. Este artículo explora los objetivos generales y específicos de Docker Desktop, destacando su capacidad para simplificar la configuración del entorno de desarrollo, garantizar la reproducibilidad, optimizar el ciclo de desarrollo y mejorar la colaboración entre equipos. La herramienta facilita la creación y ejecución de contenedores, permitiendo una rápida iteración y despliegue de aplicaciones. El resumen aborda también los pasos básicos para utilizar Docker Desktop, desde la instalación hasta la creación, ejecución y despliegue de contenedores. En conclusión, Docker Desktop emerge como una pieza fundamental para desarrolladores que buscan mejorar la consistencia y eficiencia en el desarrollo de aplicaciones en el dinámico panorama tecnológico actual.

**Palabras clave:** Docker Desktop, Contenerización, Kubernetes, Eficiencia operativa, Tecnología de contenedores

## 1. Objetivos

### 1. Objetivos Generales

- Establecer y mantener la consistencia entre los entornos de desarrollo y producción en sistemas Windows mediante el uso de Docker.
- Buscar una mejora significativa en la eficiencia del ciclo de desarrollo al adoptar Docker en entornos Windows.

### 2. Objetivos Especificos

- Asegurar que la configuración de los contenedores definida en el Dockerfile sea reproducible en diferentes entornos Windows.
- Establecer y utilizar herramientas de monitoreo integradas en Docker Desktop para gestionar y diagnosticar eficazmente el rendimiento de los contenedores en sistemas Windows.

## 2. Introducción

Docker ha ganado popularidad como una herramienta esencial para el desarrollo y despliegue de aplicaciones en entornos modernos. Este informe se centra en la implementación y utilización de Docker en sistemas operativos Windows, destacando sus beneficios y proporcionando pautas para su integración efectiva.

### 3. Instalacion

Para la instalacion vamos a tener que ir a la pagina principal de docker, con esto haremos lo siguiente(Nickoloff, 2016):

1. Vamos a descargar el instalador y cuando se descargue instalaremos el programa, al termianr reiniciaremos nuestra maquina.

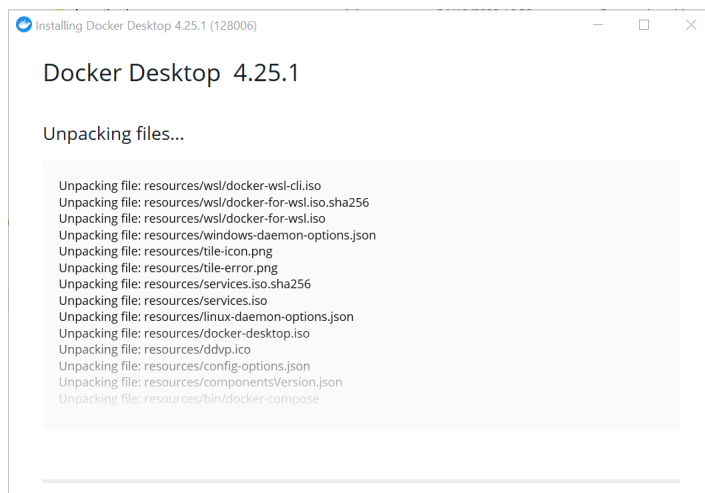


Figura 1:

2. Nos dirigimos a la parte de Windows PwerShell e instalamos el aplicativo para instalar maquinas virtuales en Windows.

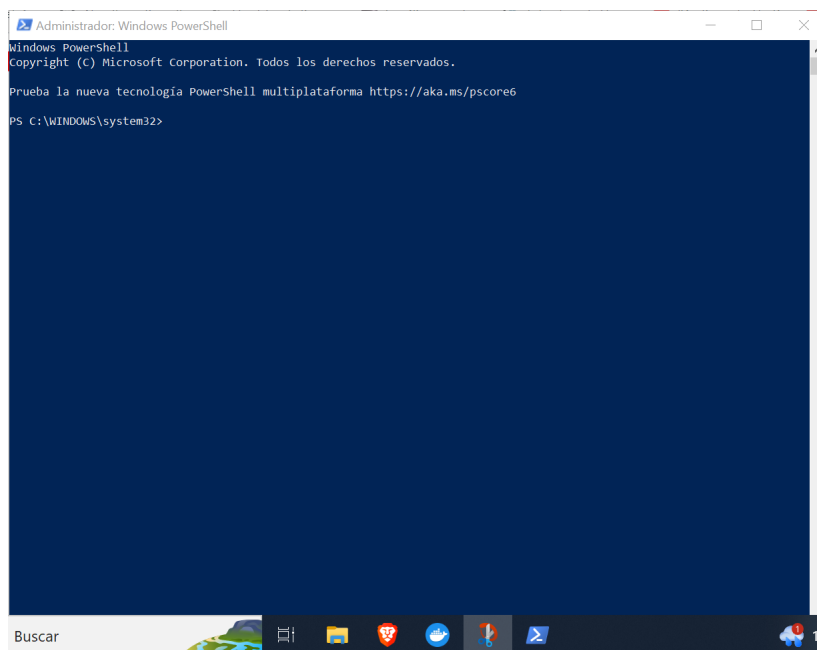
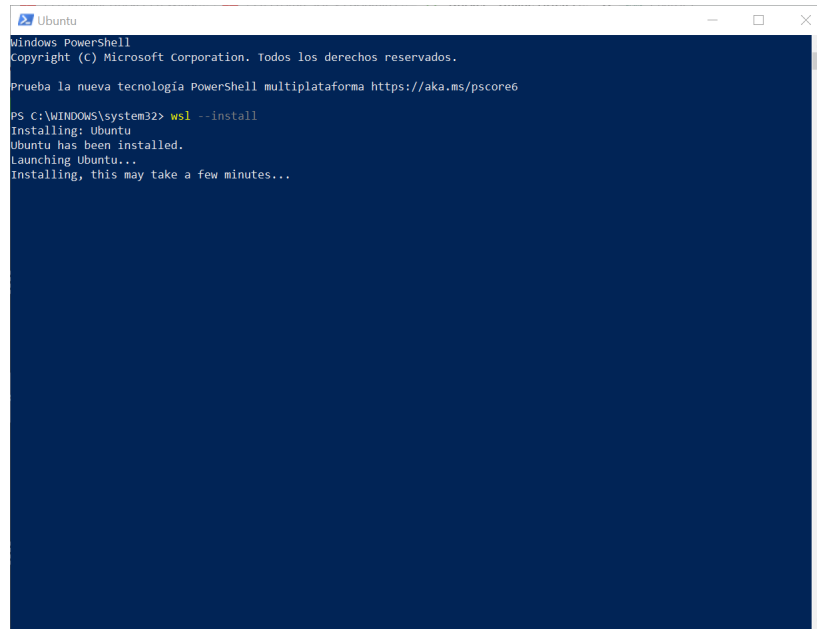


Figura 2:

3. Podemos ver que se instalo los adicionales para manejar maquinas virtuales.



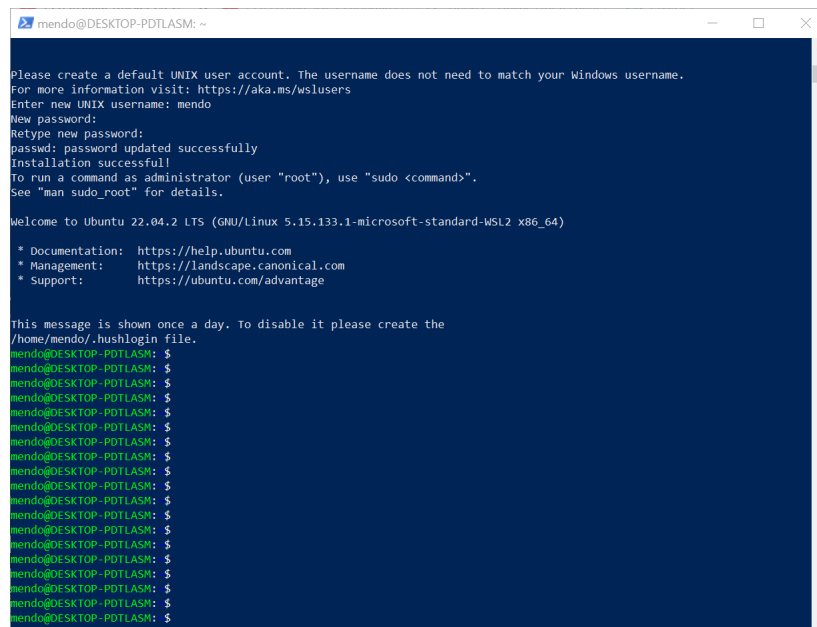
```
Windows PowerShell
Copyright (c) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\WINDOWS\system32> wsl --install
Installing: Ubuntu
Ubuntu has been installed.
Launching Ubuntu...
Installing, this may take a few minutes...
```

Figura 3:

4. Abrimos nuestro entorno virtual, donde debemos ingresar un usuario y una contraseña.



```
mendo@DESKTOP-PDTLASH: ~
Please create a default UNIX user account. The username does not need to match your Windows username.
For more information visit: https://aka.ms/wslusers
Enter new UNIX username: mendo
New password:
Retype new password:
passwd: password updated successfully
Installation successful!
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.133.1-microsoft-standard-WSL2 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

This message is shown once a day. To disable it please create the
/home/mendo/.hushlogin file.

mendo@DESKTOP-PDTLASH: $
mendo@DESKTOP-PDTLASH: $
mendo@DESKTOP-PDTLASH: $
mendo@DESKTOP-PDTLASH: $
mendo@DESKTOP-PDTLASH: $
mendo@DESKTOP-PDTLASH: $
mendo@DESKTOP-PDTLASH: $
mendo@DESKTOP-PDTLASH: $
mendo@DESKTOP-PDTLASH: $
mendo@DESKTOP-PDTLASH: $
mendo@DESKTOP-PDTLASH: $
mendo@DESKTOP-PDTLASH: $
mendo@DESKTOP-PDTLASH: $
mendo@DESKTOP-PDTLASH: $
mendo@DESKTOP-PDTLASH: $
mendo@DESKTOP-PDTLASH: $
```

Figura 4:

5. Nos dirigimos características de windows y debemos ver si tenemos activado plataforma de maquina virtual.

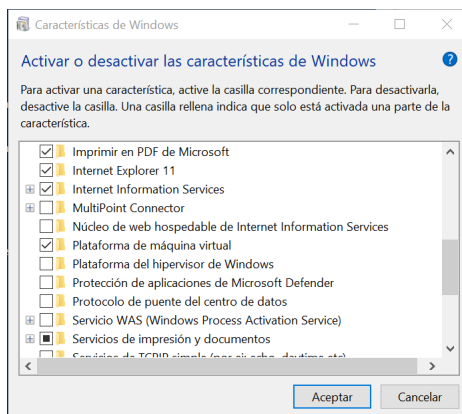


Figura 5:

Asi mismo debemos de ver si tenemos activado subsistemas de windows para linux

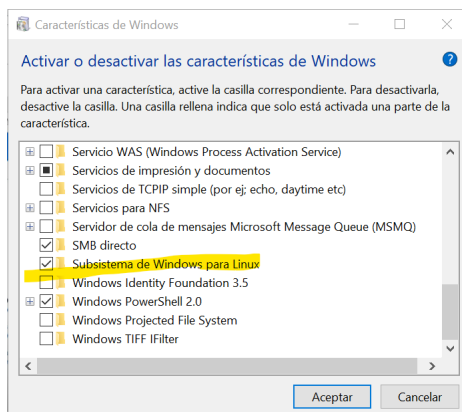


Figura 6:

6. Vemos que ya esta instalado.

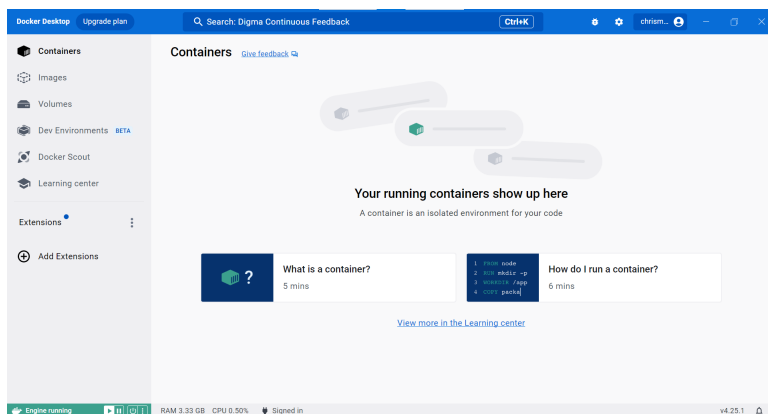


Figura 7:

## 4. Ejemplo de uso con PostgreSQL y Docker

Docker es una plataforma que facilita la implementación y ejecución de aplicaciones en contenedores. PostgreSQL, por otro lado, es un sistema de gestión de bases de datos relacional de código abierto. Al combinar Docker y PostgreSQL, se logra una forma eficiente y reproducible de gestionar bases de datos en entornos de desarrollo y producción. (Riggs y Krosing, 2010) Docker permite la contenerización de aplicaciones, lo que significa empaquetar una aplicación y todas sus dependencias en un contenedor. Un contenedor es una unidad ligera y portátil que se puede ejecutar de manera consistente en diferentes entornos. En el contexto de PostgreSQL, esto implica empaquetar el sistema de gestión de bases de datos y sus configuraciones en un contenedor, independientemente del sistema operativo subyacente. (Poulton, 2018) Una imagen de Docker es una plantilla que incluye un sistema de archivos con todo lo necesario para ejecutar una aplicación, incluidas bibliotecas, dependencias y configuraciones. Para PostgreSQL, existen imágenes de Docker oficiales que simplifican su implementación. Estas imágenes proporcionan una base lista para usar, y los usuarios pueden personalizarlas según sus necesidades.

```
version: "3.8"
```

```
services:
```

```
  postgres:
```

```
    image: postgres
```

```
    restart: always
```

```
    ports:
```

```
      - "5432:5432"
```

```
    environment:
```

```
      - DATABASE_HOST=127.0.0.1
```

```
      - POSTGRES_USER=root
```

```
      - POSTGRES_PASSWORD=root
```

```
      - POSTGRES_DB=root
```

```
  pgadmin:
```

```
    image: dpage/pgadmin4
```

```
    environment:
```

```
      PGADMIN_DEFAULT_EMAIL: "admin@admin.com"
```

```
      PGADMIN_DEFAULT_PASSWORD: "admin"
```

```
    ports:
```

```
      - "8080:80"
```

```
    depends_on:
```

```
      - postgres
```

Para el código buscamos las versiones tanto como de PostgreSQL y de pgAdmin4, dentro de esto utilizamos estos contenedores para generar esta práctica. Utilizamos imágenes de estos contenedores, esto quiere decir en que contenedor estamos trabajando. Podemos dar contraseñas y puertos para trabajar. Para mandar a correr este código, abrimos una terminal en nuestro visual studio code, y enviamos el comando de docker-compose up:

```
PS C:\Users\wendo\OneDrive\Documents\Tecnologías\2024\Deberes_1_Parcial\Docker>
PS C:\Users\wendo\OneDrive\Documents\Tecnologías\2024\Deberes_1_Parcial\Docker> docker compose up
[*] Building 0.0s (0/0)
[*] Running 2/2
Container docker-postgres-1 Running 0.0s
Container docker-pgadmin-1 Recreated 0.5s
Attaching to docker-pgadmin-1, docker-postgres-1
docker-pgadmin-1 NOTE: Configuring authentication for SERVER mode.
docker-pgadmin-1 pgAdmin 4 - Application Initialization
=====
docker-pgadmin-1 postfix/postlog: starting the Postfix mail system
docker-pgadmin-1 [2023-11-16 00:31:56 +0000] [1] [INFO] Starting unicorn 20.1.0
docker-pgadmin-1 [2023-11-16 00:31:56 +0000] [1] [INFO] Listening at: http://*:1.80 (1)
docker-pgadmin-1 [2023-11-16 00:31:56 +0000] [1] [INFO] Using worker: gthread
docker-pgadmin-1 [2023-11-16 00:31:56 +0000] [91] [INFO] Binding worker with pid: 91
docker-postgres-1 [2023-11-16 00:34:50.253 UTC [62] LOG: checkpoint starting: time
docker-postgres-1 2023-11-16 00:34:52.393 UTC [62] LOG: checkpoint complete: wrote 15 buffers (0.1%); 0 WAL file(s) added, 0 removed, 0 recycled; write=1.292
s, sync=0.595 s, total=1.541 s; sync files=0, longest=0.372 s, average=0.008 s; distance=26 kB, estimate=26 kB; lsn=0/33300C4, redo lsn=0/3330088
docker-pgadmin-1 [::ffff:172.18.0.1 - - [16/Nov/2023:00:35:23 +0000] "GET / HTTP/1.1" 302 217 "-" Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/119.0.0.0 Safari/537.36"
docker-pgadmin-1 [::ffff:172.18.0.1 - - [16/Nov/2023:00:35:23 +0000] "GET /loginnext=2f HTTP/1.1" 200 2217 "-" Mozilla/5.0 (Windows NT 10.0; Win64; x64) App
leWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.0.0 Safari/537.36"
docker-pgadmin-1 [::ffff:172.18.0.1 - - [16/Nov/2023:00:35:23 +0000] "GET /static/js/generated/style.css?ver=70800 HTTP/1.1" 200 37665 "http://localhost:8080/
/loginnext=2f" Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.0.0 Safari/537.36"
docker-pgadmin-1 [::ffff:172.18.0.1 - - [16/Nov/2023:00:35:23 +0000] "GET /static/js/generated/pgadmin.style.css?ver=70800 HTTP/1.1" 200 47878 "http://localho
st:8080/loginnext=2f" Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.0.0 Safari/537.36"
docker-pgadmin-1 [::ffff:172.18.0.1 - - [16/Nov/2023:00:35:23 +0000] "GET /static/vendor/require/require.min.js?ver=70800 HTTP/1.1" 200 6947 "http://localhost:
8080/loginnext=2f" Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.0.0 Safari/537.36"
docker-pgadmin-1 [::ffff:172.18.0.1 - - [16/Nov/2023:00:35:23 +0000] "GET /static/js/generated/pgadmin.css?ver=70800 HTTP/1.1" 200 44216 "http://localhost:808
0/loginnext=2f" Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.0.0 Safari/537.36"
docker-pgadmin-1 [::ffff:172.18.0.1 - - [16/Nov/2023:00:35:23 +0000] "GET /static/js/generated/vendor/main.js?ver=70800 HTTP/1.1" 200 8859 "http://localhost:
8080/loginnext=2f" Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.0.0 Safari/537.36"
docker-pgadmin-1 [::ffff:172.18.0.1 - - [16/Nov/2023:00:35:23 +0000] "GET /static/js/generated/vendor/main.js?ver=70800 HTTP/1.1" 200 126534 "http://localhost:
8080/loginnext=2f" Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.0.0 Safari/537.36"
```

Figura 8:

Vemos como esto en nuestro puerto se despliega y ver nuestra pagina:

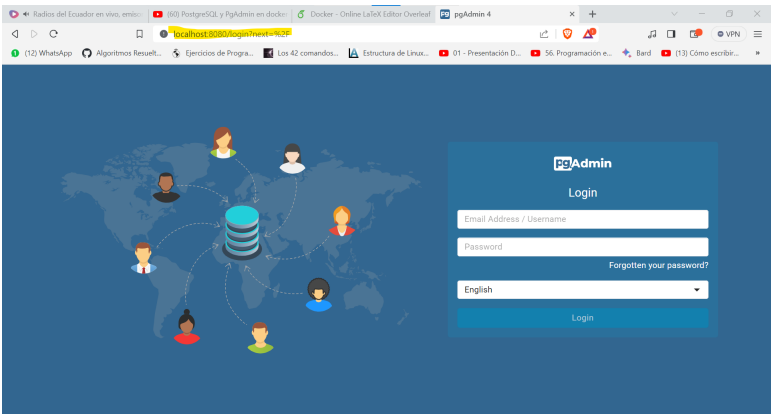


Figura 9:

Ingresamos a nuestra base de datos de Postgres y podemos crar nuestras bases de datos y tablas por demas:



Figura 10:

Como final, podemos ver que se creo nuestro contenedor en docker:

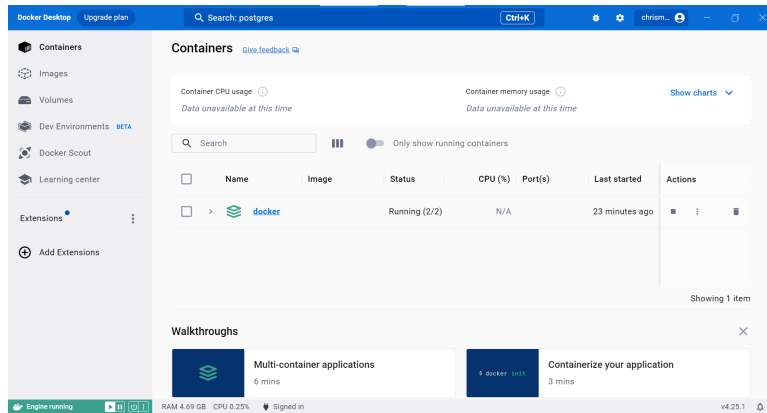


Figura 11:

## 5. Conclusiones:

En general, Docker ofrece una poderosa solución para la gestión de entornos de desarrollo y despliegue. Al entender las consideraciones específicas al trabajar con Docker en entornos Windows y con bases de datos como PostgreSQL, puedes aprovechar al máximo estas tecnologías para mejorar la eficiencia y la consistencia en tus proyectos.

## Referencias

- Nickoloff, J. (2016). *Docker in action*. Manning Publications.
- Poulton, N. (2018). *Docker deep dive*. Autor.
- Riggs, S., y Krosing, H. (2010). *Postgresql 9 administration cookbook*. Packt Publishing.