



Bloque 1 –Tema 4. Docker.

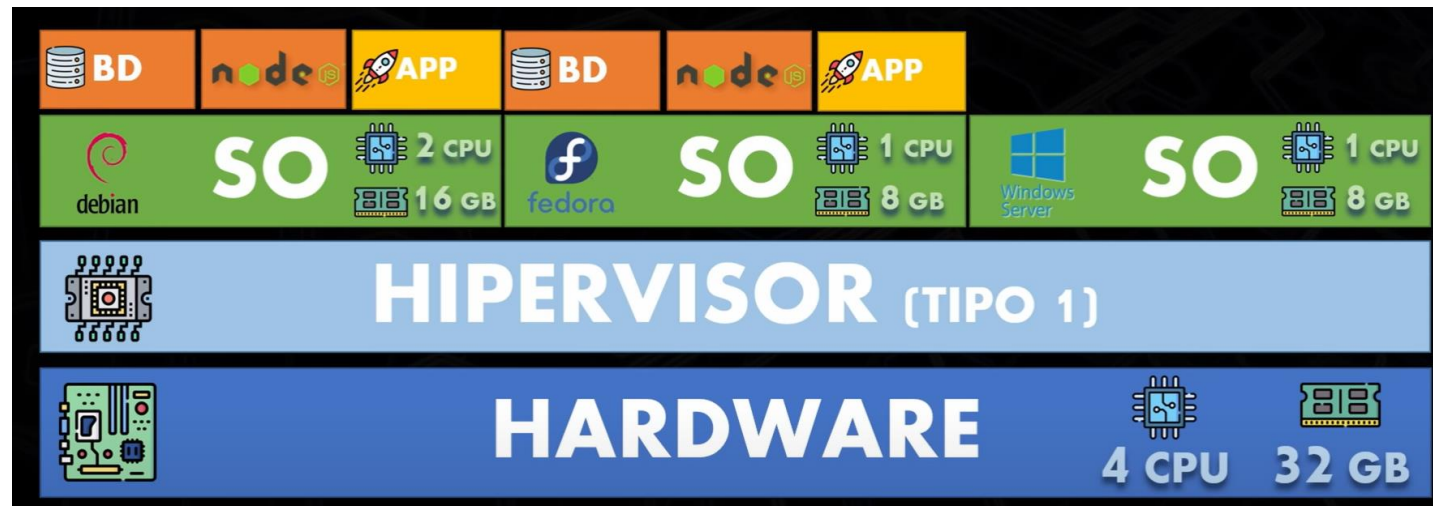
Contenedores para la Gestión, Orquestación y
Procesamiento Escalable de Datos en entornos Big
Data y Data Engineering

Servidor físico vs máquina virtual



Servidor
Físico

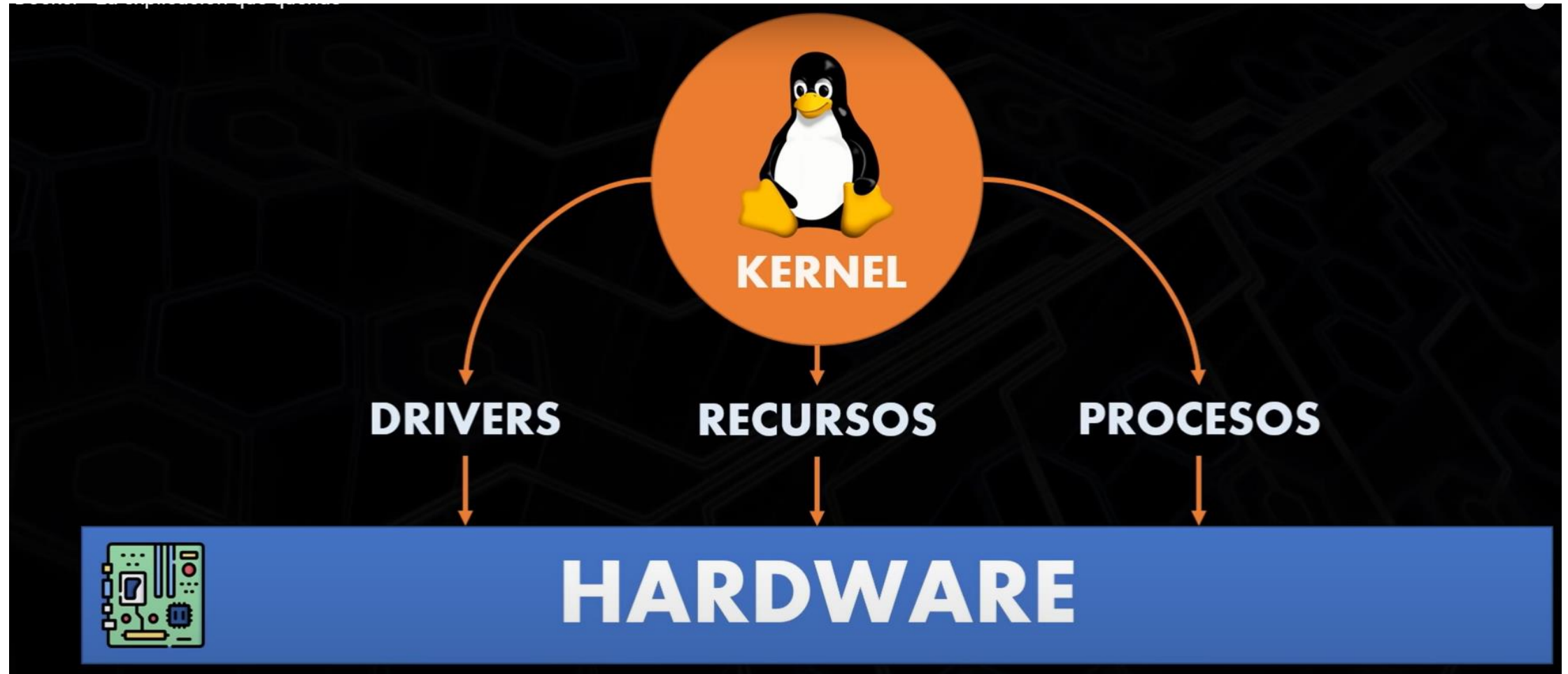
Máquina
Virtual



Al aumentar el nro de máquinas virtuales se incrementa el costo en hardware.



En vez de tener distintos kernels ¿Por qué no utilizar un único kernel a partir del cual se gestionen todos los servicios?



En vez de tener distintos kernels
¿Por qué no utilizar un único kernel a partir del cual se
gestionen todos los servicios?





Docker es un conjunto de herramientas que se utiliza para empaquetar aplicaciones con todas sus herramientas y bibliotecas necesarias en "contenedores" ordenados y portátiles. Estos contenedores se ejecutan en cualquier lugar, aislados unos de otros, lo que garantiza un rendimiento uniforme y eficiente.

Contenedores y contenedorización (Containers and Containerization)

Un contenedor es como un conjunto de bibliotecas, dependencias y archivos de configuración. En comparación con las máquinas virtuales que emulan sistemas operativos completos, los contenedores comparten el núcleo (kernel) del host, lo que los hace más ligeros y rápidos.

Cada contenedor está aislado del otro, por lo que es independiente y sus procesos y recursos no interfieren entre sí.

¿Qué son contenedores?

Los contenedores son un paquete de elementos que permiten ejecutar una aplicación determinada en cualquier sistema operativo.



Docker

Linux-VServer

LXC

LXD

OpenVZ

Systemd-nspawn

Contenedores y contenedorización (Containers and Containerization)

La contenedorización es el proceso de empaquetar una aplicación en un contenedor estandarizado. Docker es una popular plataforma de contenedorización que proporciona herramientas para crear, compartir y ejecutar contenedores.

La contenedorización beneficia a la ciencia de datos de muchas maneras:

Containerization

Reproducibilidad: Ejecute su análisis con el mismo entorno (versiones de software, dependencias) en cualquier lugar, garantizando la coherencia de los resultados.

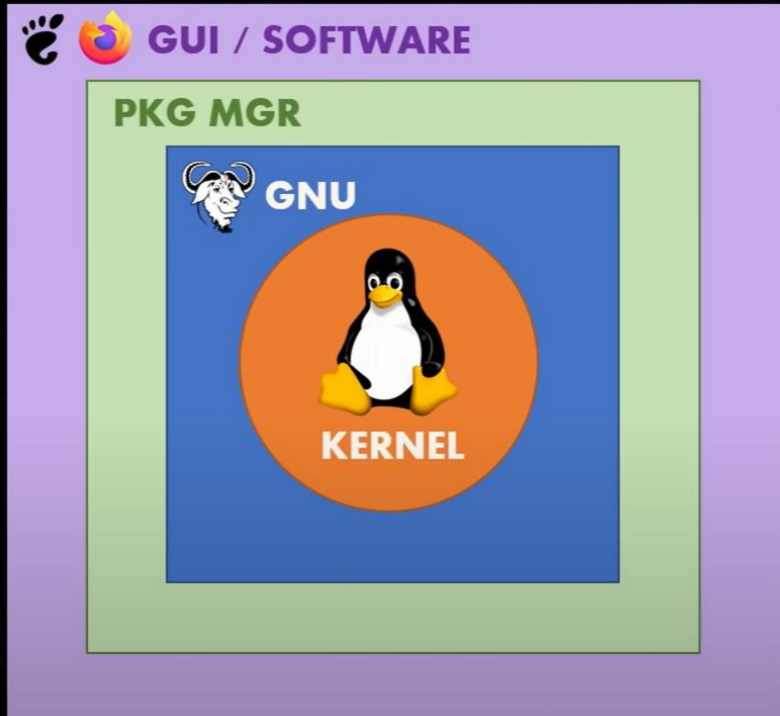
Portabilidad: Comparta y traslade fácilmente sus proyectos de ciencia de datos entre máquinas sin preocuparse por la configuración del entorno.

Eficiencia de recursos: Múltiples contenedores pueden ejecutarse en una sola máquina, maximizando los recursos informáticos para el procesamiento y análisis de datos.

Aislamiento: Aísle tareas específicas como la ingeniería de características o la formación de modelos para una mejor colaboración y experimentación.

Máquinas virtuales frente a Docker

Máquinas virtuales



Contenedores



Máquinas virtuales frente a Docker

Las máquinas virtuales (VM) y los contenedores Docker son tecnologías utilizadas para aislar entornos de software, pero difieren significativamente en su enfoque y casos de uso. Comprender estas diferencias es crucial para elegir la herramienta adecuada para sus necesidades específicas.

Máquinas virtuales (VMs):

Emular un sistema operativo completo: Cada VM actúa como un ordenador virtual con su CPU, memoria, almacenamiento y sistema operativo.

Pros:

Aislamiento total: Las máquinas virtuales ofrecen un fuerte aislamiento entre entornos, lo que las hace ideales para ejecutar aplicaciones con dependencias conflictivas o requisitos de seguridad.

Máquinas virtuales (VMs):

Pros:

Flexibilidad: Las máquinas virtuales pueden ejecutar cualquier sistema operativo, lo que permite crear entornos adaptados a necesidades específicas.

Máquinas virtuales (VMs):

Contras:

Consumo intensivo de recursos: Las máquinas virtuales requieren muchos recursos, lo que limita el número de ellas que se pueden ejecutar en una sola máquina.

Arranque lento: Arrancar un sistema operativo completo lleva tiempo, por lo que las máquinas virtuales tardan más en iniciarse que los contenedores.

Contenedores Docker:

Empaqueta una aplicación con sus dependencias: Los contenedores comparten el núcleo del sistema operativo anfitrión y sólo necesitan bibliotecas y archivos específicos para ejecutarse.

Contenedores Docker:

Ligeros y portátiles: Los contenedores son mucho más pequeños y rápidos de arrancar que las máquinas virtuales, lo que los hace ideales para microservicios y aplicaciones escalables.

Entornos coherentes: Los contenedores estandarizados garantizan un comportamiento coherente de las aplicaciones en diferentes máquinas.

Uso eficiente de los recursos: Compartir el kernel permite ejecutar muchos más contenedores en una sola máquina en comparación con las VM.

Contenedores Docker:

Contras:

Aislamiento limitado: Aunque están aislados, los contenedores comparten el kernel, lo que introduce algunos riesgos potenciales de seguridad en comparación con las máquinas virtuales.

Elección de SO limitada: Los contenedores suelen utilizar el sistema operativo del host, lo que limita su flexibilidad en comparación con las máquinas virtuales.

Utilizar máquinas virtuales para:

- Aplicaciones con dependencias conflictivas o estrictos requisitos de seguridad.
- Ejecución de varios sistemas operativos en una sola máquina.
- Cuando el aislamiento completo es crítico.

Utiliza contenedores Docker para:

- Microservicios y aplicaciones escalables.
- Despliegue de aplicaciones en entornos consistentes.
- Utilización eficiente de los recursos y despliegues rápidos.



¿Por qué es importante Docker en la ciencia de datos?

Deploy your Machine Learning
Model on Docker



¿Por qué es importante Docker en la ciencia de datos?

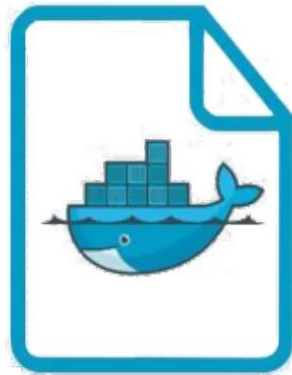
Suele pasar con mucha frecuencia que has desarrollado un modelo de aprendizaje automático y luego, cuando has cambiado de portátil, tu código se está rompiendo y te encuentras con una declaración infinita de "Error de importación" o "Módulo no encontrado". A veces, podría ser un error de versión debido a que se está tratando de ejecutar el código en una versión diferente de Python. La solución para esto es Docker.

¿Por qué es importante Docker en la ciencia de datos?

Docker es una herramienta para crear y desplegar entornos aislados para ejecutar aplicaciones con sus dependencias. Básicamente, Docker facilita la escritura y ejecución de códigos sin problemas en otras máquinas con sistemas operativos diferentes, reuniendo el código y todas sus dependencias en un contenedor.

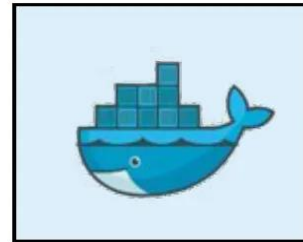
Este contenedor hace que el código sea autónomo e independiente del sistema operativo.

Terminología de Docker



Dockerfile

Build

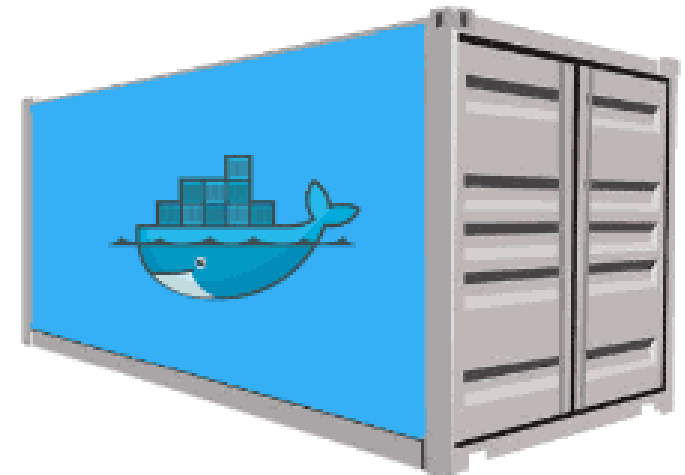


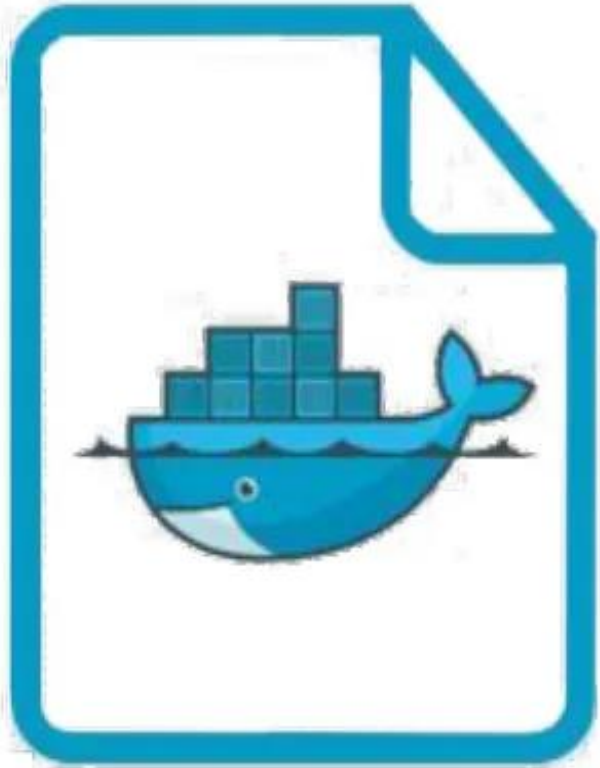
Docker
Image

Run



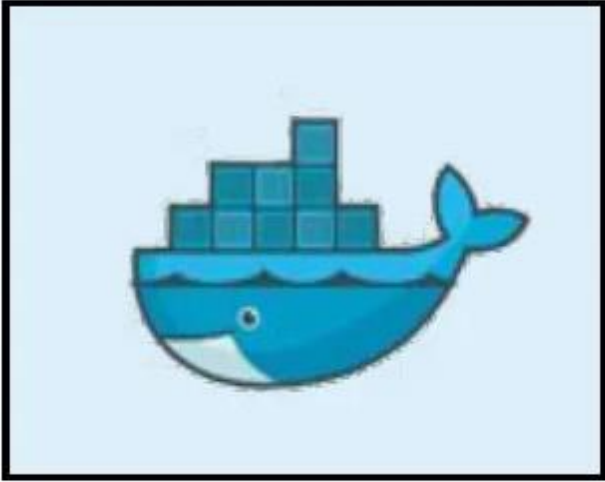
Docker
Container





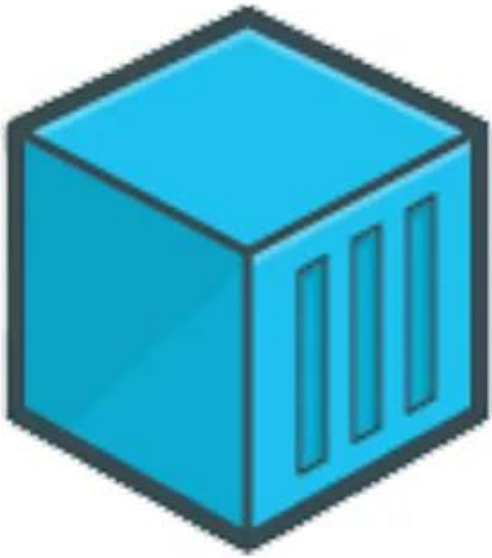
Dockerfile

Dockerfile - Un Dockerfile contiene todo el código para configurar un contenedor docker desde la descarga de la imagen docker hasta la configuración del entorno. Puedes pensar en él como si describiera la instalación completa del sistema operativo del sistema que quieres ejecutar.



Docker
Image

La imagen Docker es una plantilla de sólo lectura que contiene un conjunto de instrucciones para crear un contenedor que pueda ejecutarse en la plataforma Docker.



Docker
Container

Contenedor Docker - Un contenedor es una instancia ejecutable de una imagen. Puede crear, iniciar, detener, mover o eliminar un contenedor mediante la API o la CLI de Docker.

¿Por qué es importante Docker en la ciencia de datos?

Ejecutar modelos ML en contenedores Docker aporta una serie de ventajas:

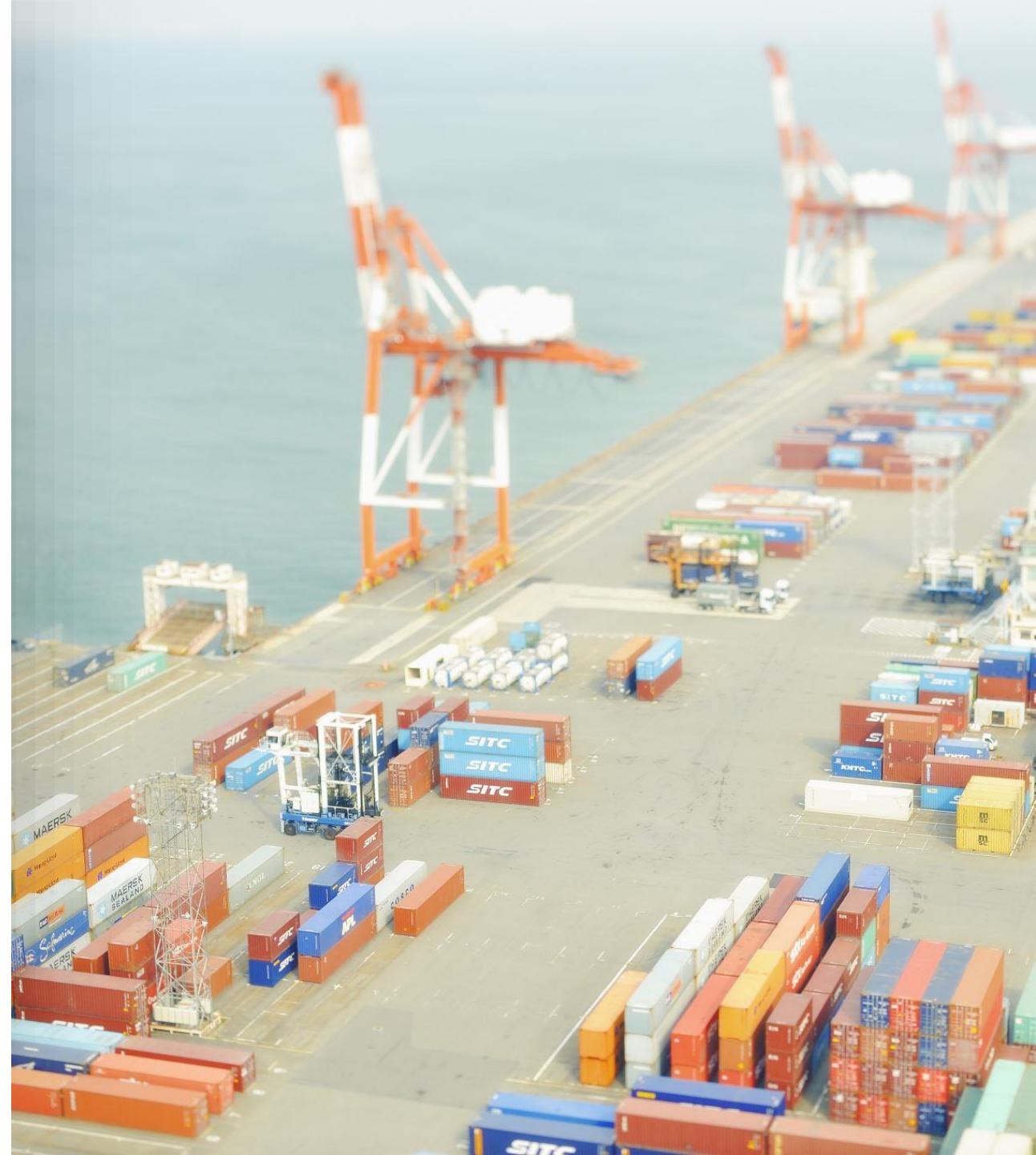
Reproducibilidad: Una vez que hayas probado tu aplicación en contenedores, puedes desplegarla en cualquier otro sistema en el que se ejecute Docker y puedes estar seguro de que tu aplicación funcionará exactamente igual que cuando la probaste.



¿Por qué es importante Docker en la ciencia de datos?

Agilidad: La portabilidad y las ventajas de rendimiento que ofrecen los contenedores pueden ayudarle a que su proceso de desarrollo sea más ágil y receptivo. Mejora sus procesos de integración continua y entrega continua.

Portabilidad: Esto significa que pasar del desarrollo local a un clúster de supercomputación es fácil. Puede evitar problemas de configuración.



Importancia de Docker en Big Data

Consistencia en los entornos: Permite crear entornos de ejecución consistentes que se comportan de la misma manera, independientemente del sistema operativo o la infraestructura subyacente.

Los contenedores Docker encapsulan las dependencias, librerías y configuraciones necesarias para que las aplicaciones de Big Data funcionen correctamente. Esto asegura que el código funcione igual en desarrollo, prueba y producción

Importancia de Docker en Big Data

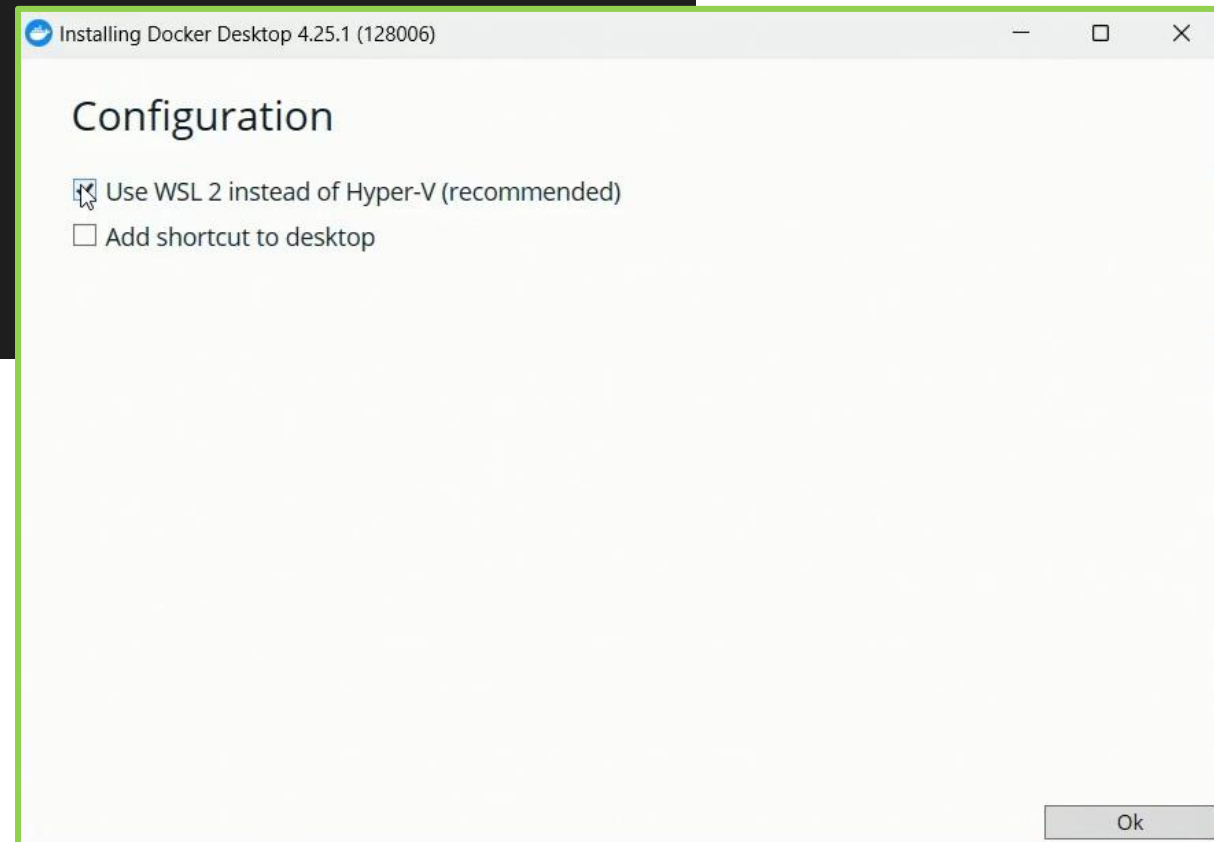
Docker facilita la escalabilidad al permitir que los nodos de procesamiento se creen y gestionen rápidamente en contenedores que pueden desplegarse en cualquier servidor o clúster.

Con Docker Swarm o Kubernetes, la orquestación de múltiples contenedores se simplifica, permitiendo escalar horizontalmente aplicaciones como Hadoop o Spark para adaptarse a cargas de trabajo cambiantes.

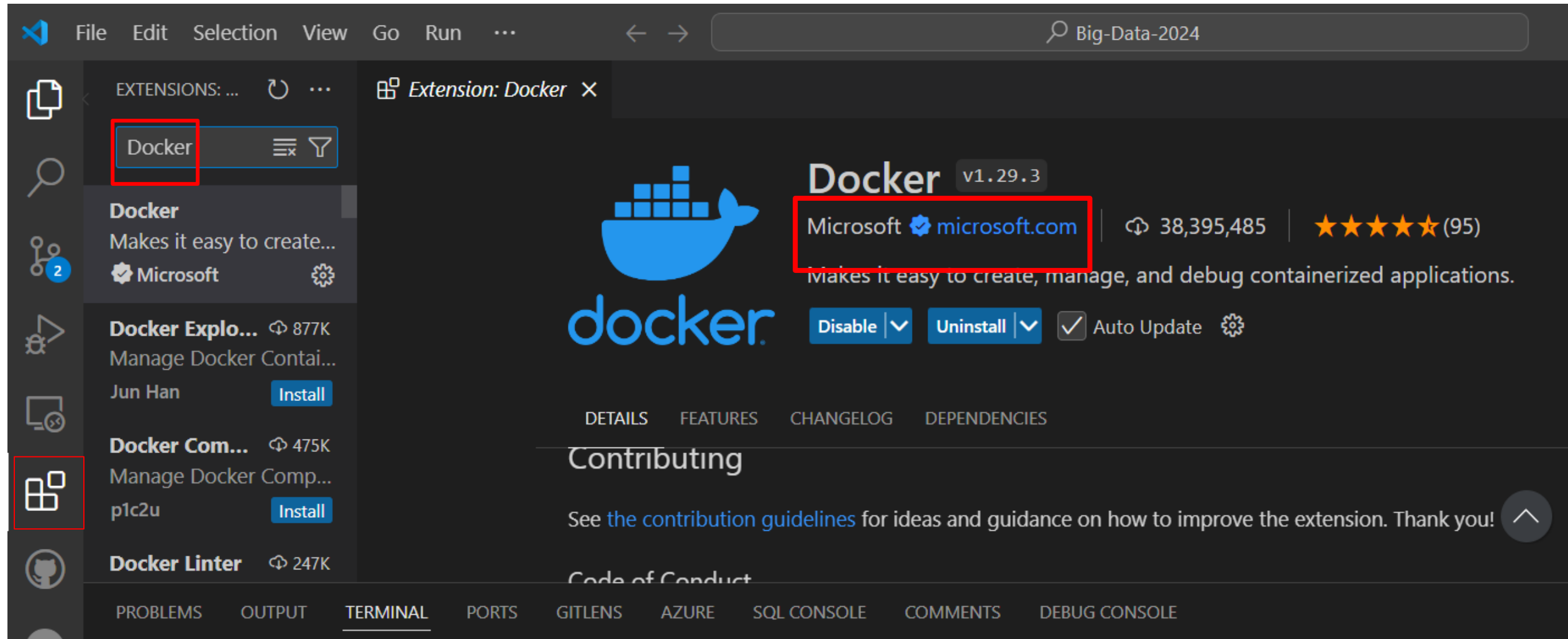
Verificar versión WSL en VS Code

PROBLEMS OUTPUT TERMINAL PORTS GITLENS AZURE SQL CONSOLE COMMENTS DEBUG CONSOLE

```
PS C:\Users\juanj\OneDrive\Escritorio\Big-Data-2024> wsl --version
Versión de WSL: 2.3.24.0
Versión de kernel: 5.15.153.1-2
Versión de WSLg: 1.0.65
Versión de MSRDC: 1.2.5620
Versión de Direct3D: 1.611.1-81528511
Versión DXCore: 10.0.26100.1-240331-1435.ge-release
Versión de Windows: 10.0.22631.4317
PS C:\Users\juanj\OneDrive\Escritorio\Big-Data-2024>
```



Instalar extensión en VS Code



Instalación de Docker en Windows

[Home](#) / [Manuals](#) / [Docker Desktop](#) / [Install](#) / Windows

Install Docker Desktop on Windows

Docker Desktop terms

Commercial use of Docker Desktop in larger enterprises (more than 250 employees OR more than \$10 million USD in annual revenue) requires a [paid subscription](#) [↗](#).

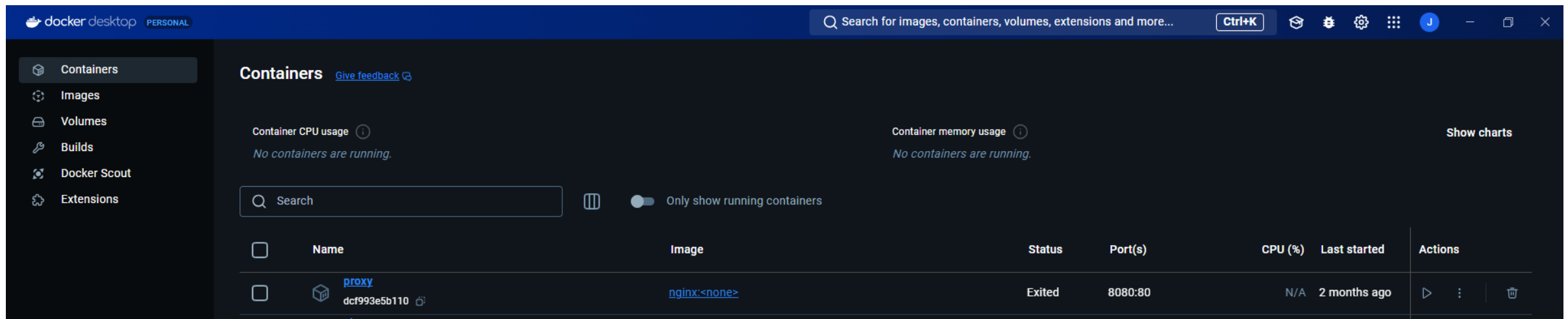
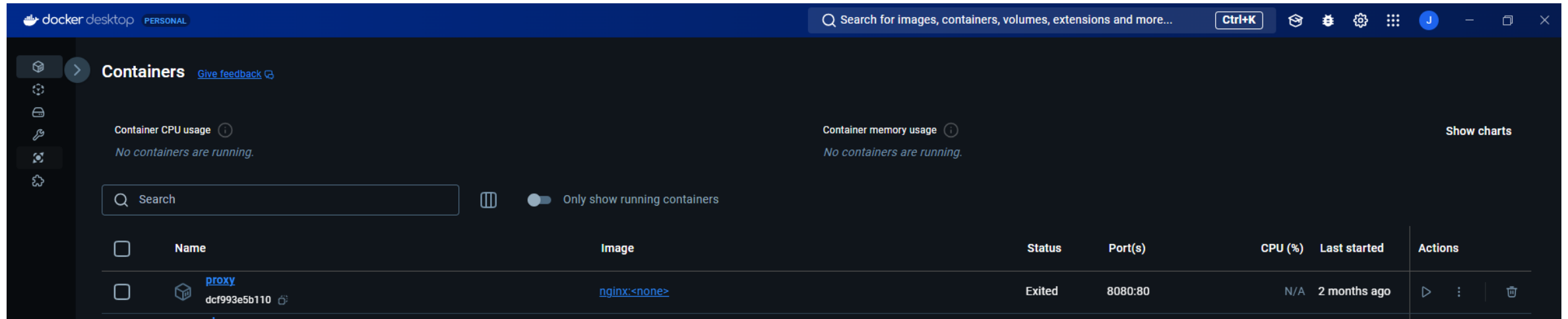
This page contains the download URL, information about system requirements, and instructions on how to install Docker Desktop for Windows.

Docker Desktop for Windows - x86_64








Docker Desktop for Windows - Arm (Beta)

For checksums, see [Release notes](#)

UI de Docker cuando se ejecuta desde Windows



Learning Center


ainers, volumes, extensions and more... Ctrl+K     J   

Learning center

×

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 packa
```

How do I run a container?
6 mins

[View all](#)


AI/ML guides


Get started with AI/ML using Docker


 **GenAI Stack**
Start your GenAI application using Neo4j, Langchain, Ollama, Python, and Docker Compose


Docker for beginners by language


45 min guides written for different programming languages



NodeJS



Python


Go


Java


C# (.NET)


Rust

[Request a guide](#) 

Docker Images

Images

[Give feedback](#)

Local

Hub

265.74 MB / 265.74 MB in use

2 images

Q Search

<input type="checkbox"/>	Name	Tag
<input type="checkbox"/>	nginx 5ef79149e0ec	latest
<input type="checkbox"/>	ubuntu edbf74c41f8	latest

Images

[Give feedback](#)

Local

Hub

juanjorb23



Q Search

Tags







	juanjorb23/sitioweb	latest
--	---------------------	--------

Volumes en Docker:

Volumes [Give feedback](#)


Search  

[Create](#)








<input type="checkbox"/> Name ↑	Status	Created	Size	Scheduled exports <small>BETA</small>	Actions
<input type="checkbox"/> 03fee6fec6a1fe1efefb3136c54d8a6dde3d4a618cf4458329ffc77b971b1115	-	2 months ago	0 Bytes	Inactive	 
<input type="checkbox"/> 20ef28bd50e42339859896a248ee28e04b4b7b1318101a4164f08767a1e6effd	-	2 months ago	300.4 MB	Inactive	 
<input type="checkbox"/> 4806a44ba1e1c549d556d99acd9bfe80ea9b82804a997834ced0c8c3335fa84b8	-	2 months ago	300.2 MB	Inactive	 

Es un mecanismo para persistir datos generados o utilizados por contenedores. Los volúmenes se utilizan cuando los datos necesitan sobrevivir al ciclo de vida de un contenedor (es decir, cuando un contenedor se detiene o es eliminado, los datos persisten).


Settings - Configuración


 **docker desktop** PERSONAL


Search for images, containers, volumes, extensions and more... **Ctrl+K**


      


Settings [Give feedback](#)


 **General**


 Resources


 Docker Engine


 Builders

 Kubernetes

 Software updates

 Extensions

 Features in development

 Notifications

General

☐ Start Docker Desktop when you sign in to your computer

☒ Open Docker Dashboard when Docker Desktop starts

Choose theme for Docker Desktop

☐ Light ☐ Dark ☒ Use system settings

Choose container terminal

☒ Integrated ☐ System default

Determines which terminal is launched when opening the terminal from a container.

☒ Enable Docker terminal

☐ Enable Docker Debug by default [Learn more](#)

Active subscription required. [Upgrade](#)

☐ Expose daemon on tcp://localhost:2375 without TLS

Exposing daemon on TCP without TLS helps legacy clients connect to the daemon. It also makes yourself vulnerable to remote code execution attacks. Use with caution.

☒ Use the WSL 2 based engine (Windows Home can only run the WSL 2 backend)

WSL 2 provides better performance than the Hyper-V backend. [Learn more](#)

☐ Add the *.docker.internal names to the host's /etc/hosts file (Requires password)

Lets you resolve *.docker.internal DNS names from both the host and your containers. [Learn more](#)

☐ Use containerd for pulling and storing images [Give feedback](#)

The containerd image store enables native support for multi-platform images, attestations, Wasm, and more.

☒ Send usage statistics

Docker CLI

Desde vs code abrimos una terminal de wsl y ejecutamos: **docker --version**

```
PROBLEMS  OUTPUT  TERMINAL  PORTS  GITLENS  AZURE  SQL CONSOLE  COMMENTS  DEBUG CONSOLE

balrodjuan@balrodjj:/mnt/c/Users/juanj/OneDrive/Escritorio/Big-Data-2024$ docker --version
Docker version 27.2.0, build 3ab4256
balrodjuan@balrodjj:/mnt/c/Users/juanj/OneDrive/Escritorio/Big-Data-2024$
```

Para acceder a toda la documentación (todos los comandos) se escribe: **docker**

```
balrodjuan@balrodjj:/mnt/c/Users/juanj/OneDrive/Escritorio/Big-Data-2024$ docker

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Common Commands:
```

El comando `docker info` proporciona un resumen detallado de la configuración actual de Docker, incluyendo información sobre el estado del servidor, los recursos disponibles y la configuración del entorno de Docker.

```
balrodjuan@balrodjj:/mnt/c/Users/juanj/OneDrive/Escritorio/Big-Data-2024$ docker info
Client:
Version:      27.2.0
Context:      default
Debug Mode:   false
Plugins:
buildx: Docker Buildx (Docker Inc.)
  Version:    v0.16.2-desktop.1
  Path:       /usr/local/lib/docker/cli-plugins/docker-buildx
compose: Docker Compose (Docker Inc.)
  Version:    v2.29.2-desktop.2
  Path:       /usr/local/lib/docker/cli-plugins/docker-compose
debug: Get a shell into any image or container (Docker Inc.)
  Version:    0.0.34
  Path:       /usr/local/lib/docker/cli-plugins/docker-debug
desktop: Docker Desktop commands (Alpha) (Docker Inc.)
  Version:    v0.0.15
  Path:       /usr/local/lib/docker/cli-plugins/docker-desktop
dev: Docker Dev Environments (Docker Inc.)
  Version:    v0.1.2
  Path:       /usr/local/lib/docker/cli-plugins/docker-dev
extension: Manages Docker extensions (Docker Inc.)
  Version:    v0.2.25
  Path:       /usr/local/lib/docker/cli-plugins/docker-extension
feedback: Provide feedback, right in your terminal! (Docker Inc.)
```


Actividad:

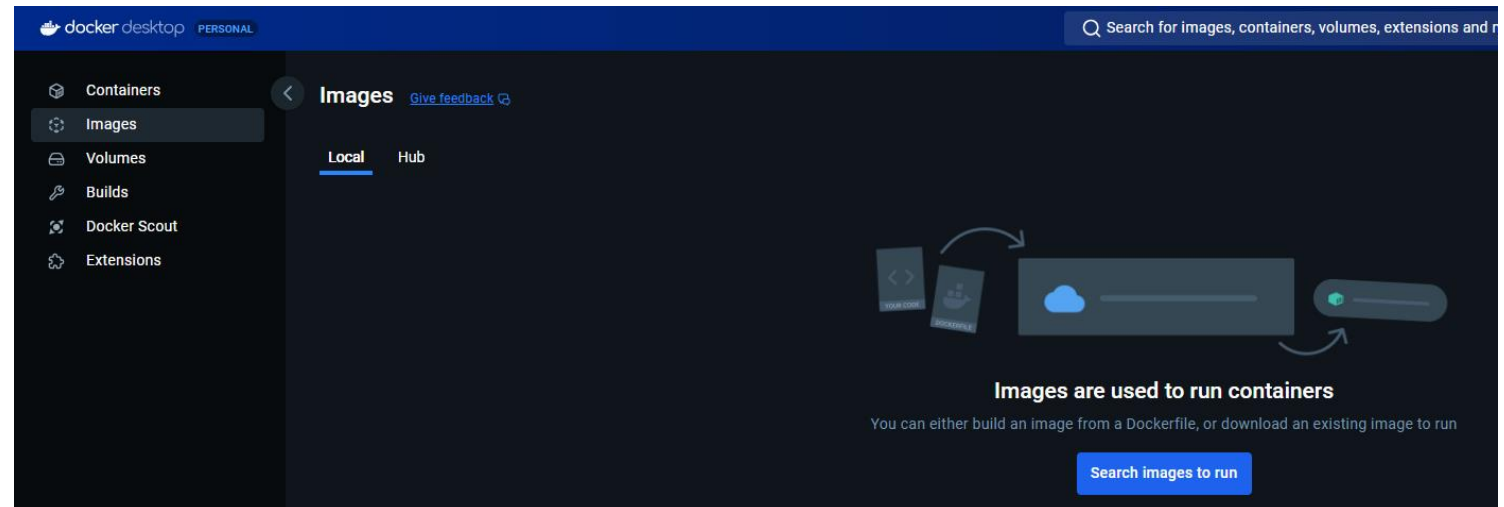
Escribe docker info y luego explica lo que contiene la salida. Investigar para que se usa. Guarda tus resultados en tus apuntes

docker images

Se usa para listar las imágenes disponibles localmente en tu sistema Docker. Proporciona un resumen de todas las imágenes que están almacenadas en tu máquina, mostrando detalles clave como el nombre de la imagen, la etiqueta, el ID, la fecha de creación y el tamaño.

```
PROBLEMS  OUTPUT  TERMINAL  PORTS  GITLENS  AZURE  SQL CONSOLE  COMMENTS  DEBUG CONSOLE
Remote Explorer
balrodjuan@balrodjj:/mnt/c/Users/juanj/OneDrive/Escritorio/Big-Data-2024$ docker images
REPOSITORY    TAG       IMAGE ID   CREATED   SIZE
balrodjuan@balrodjj:/mnt/c/Users/juanj/OneDrive/Escritorio/Big-Data-2024$
```

Desde Docker Desktop



Documentación de docker images:

docker images --help

```
balrodjuan@balrodjj:/mnt/c/Users/juanj/OneDrive/Escritorio/Big-Data-2024$ docker images --help
```

```
Usage:  docker images [OPTIONS] [REPOSITORY[:TAG]]
```

```
List images
```

```
Aliases:
```

```
    docker image ls, docker image list, docker images
```

```
Options:
```

-a, --all	Show all images (default hides intermediate images)
--digests	Show digests
-f, --filter filter	Filter output based on conditions provided
--format string	Format output using a custom template:
	'table': Print output in table format with column headers (default)
	'table TEMPLATE': Print output in table format using the given Go template
	'json': Print in JSON format
	'TEMPLATE': Print output using the given Go template.
	Refer to https://docs.docker.com/go/formatting/ for more information about formatting output with templates
--no-trunc	Don't truncate output
-q, --quiet	Only show image IDs
--tree	List multi-platform images as a tree (EXPERIMENTAL)





















