

# ABCDocker

El ABC de Docker y  
contenedores



# 01

## Qué es Que?

Diferencia entre Docker y Contenedor



# Qué es Docker y qué es un contenedor



## Docker

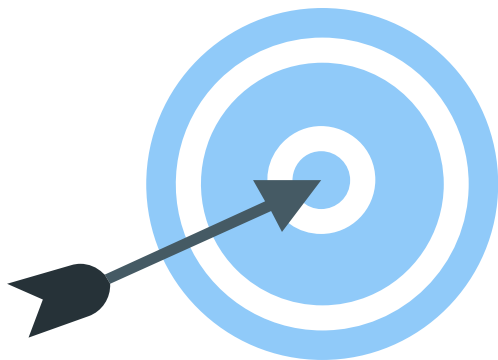
Es una empresa que desarrolló  
varios productos enfocados a  
contenedores



## Contenedor

Es una tecnología basada en los  
cgroups de Linux



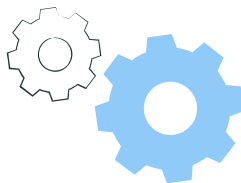


## ¿Para que?

## Contenedores

Los contenedores son una forma de paquetizar software para poder correrlo en diferentes plataformas (Windows, Linux, OSX, etc) sin necesidad de preocuparse por compatibilidades ni dependencias.

Hay muchas opciones: Docker, Podman, Containerd y más..





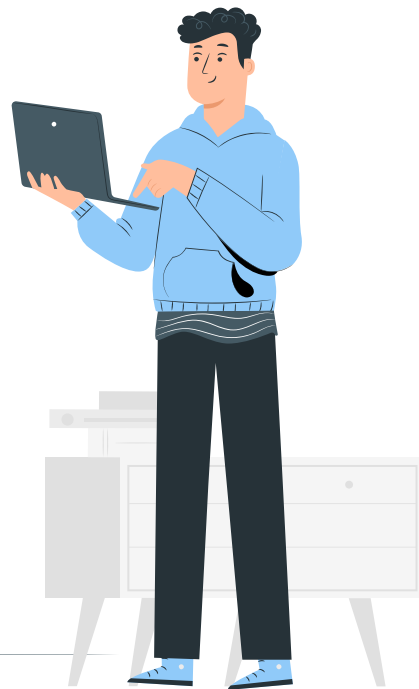
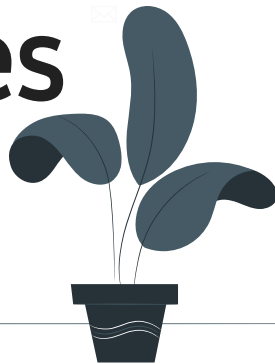
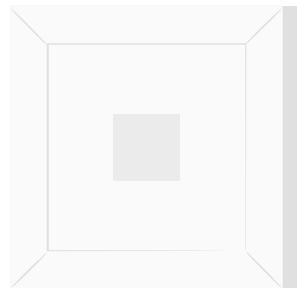
Algo importante a saber es que dentro de tu contenedor solo debes ejecutar un (1) proceso.  
Ej: Webserver, DB, etc.



# 02

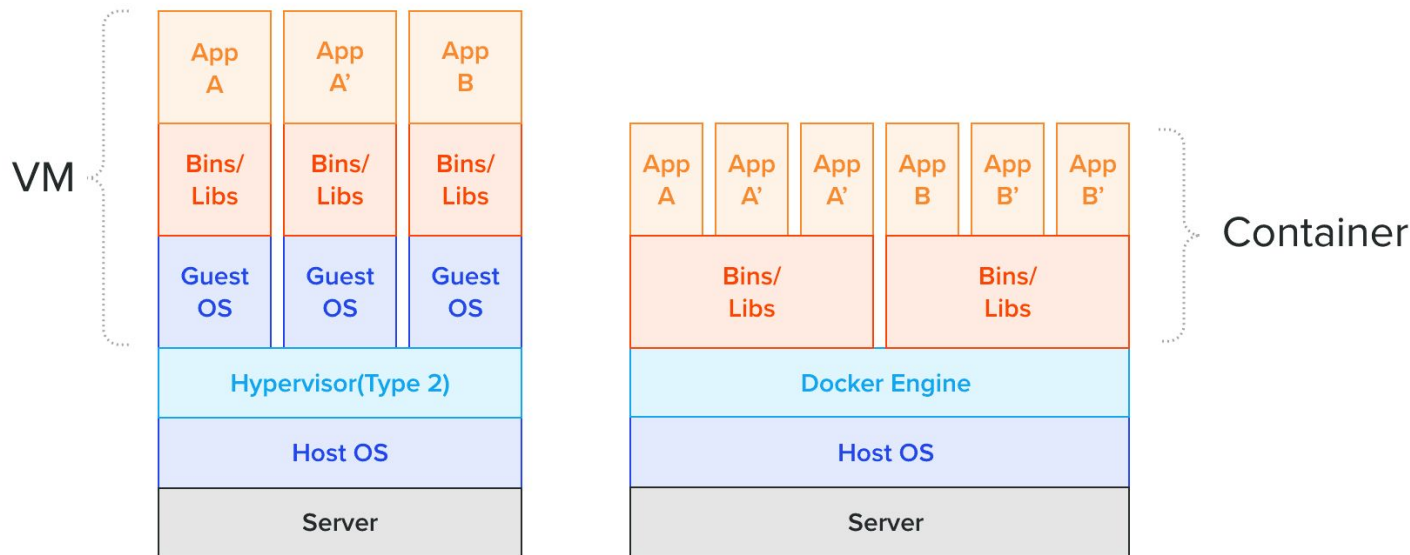
## VM's vs Contenedores

¿Cuál es la diferencia?

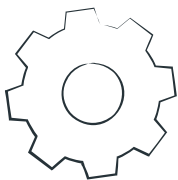




# Máquinas Virtuales vs Contenedores



Los contenedores proporcionan aislamiento de procesos en un sistema operativo compartido, en lugar del aislamiento del sistema operativo en hardware compartido que se obtiene con las máquinas virtuales.



# Ventajas



## 01

### Recursos

Máximo  
aprovechamiento de los  
recursos del servidor

## 02

### Operación

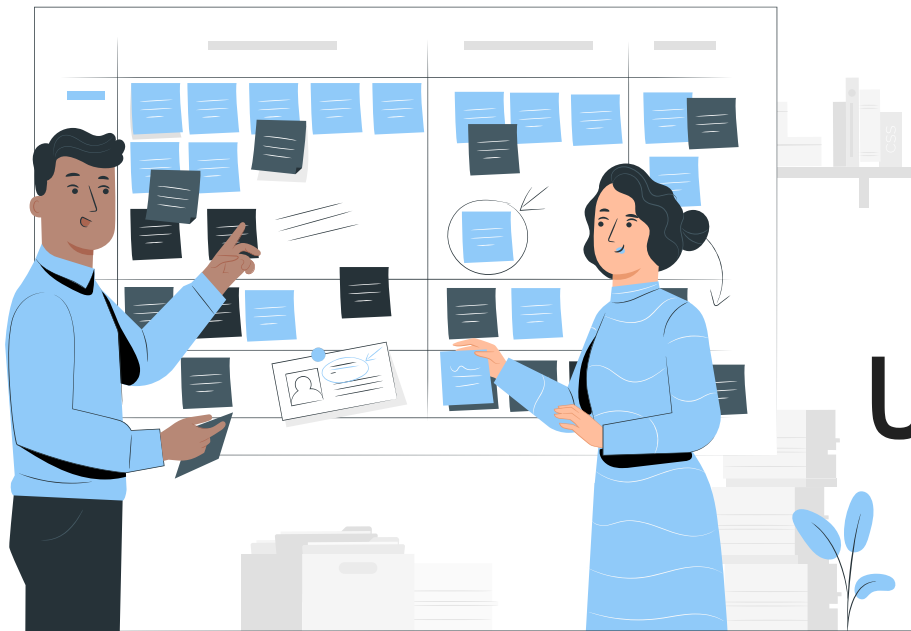
Reduce conflictos de  
dependencias.  
Fácil de actualizar

## 03

### Optimización

Ejecuta PHP, Python,  
C#, C, Golang,  
ASP.NET, NodeJS...





03

# Como Se Crea Un Contenedor

Imágenes, la forma de *paquetizar*



# Imágenes



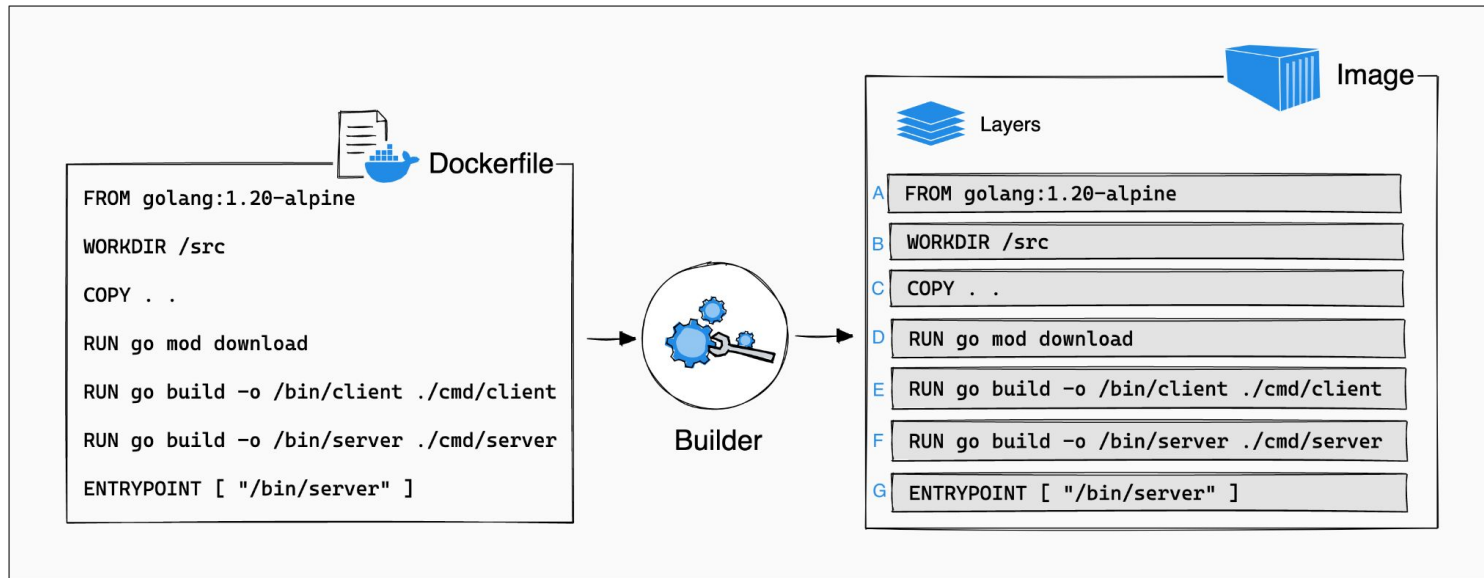
Las imágenes se construyen a través de un archivo nombrado Dockerfile

```
FROM python:3.9-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY ./src ./
EXPOSE 80
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "80"]
```

Al ejecutar el contenido del archivo, se generan *layers* que formarán la imagen final. Esto trae como ventaja que esos *layers* pueden ser reutilizados para otras imágenes



# Imágenes



Cada comando crea una *layer* que está vinculado al anterior. (A → B → C → D → ....)

💡 Si modificas la última línea del Dockerfile y vuelvas a crear la imagen, solo se modifica ese layer (G). Pero si modificas la primera, todos los otros layers tendrán que volverse a crear.

<https://gdi3d.github.io/course/docs#/dia-1/intro>

# Instalar Docker





Ve al directorio `course/resources/dia-1/check` y ejecuta:

```
docker build -t abcdocker_check .
```

Y para comprobar que el contenedor puede ser lanzado correctamente:

```
docker run --rm abcdocker_check
```

**Figure 6**

The figure illustrates the relationship between the number of nodes ( $N$ ) and the number of links ( $L$ ) for different network topologies. It is divided into three main sections:

- Left Section (Grid Topology):** Shows a square grid where each node is connected to four neighbors. The number of links is calculated as  $L = \frac{4}{2}N - \frac{1}{2}$ . Examples shown include a single node ( $N=1, L=0$ ), two nodes ( $N=2, L=1$ ), three nodes ( $N=3, L=2$ ), four nodes ( $N=4, L=3$ ), five nodes ( $N=5, L=4$ ), six nodes ( $N=6, L=5$ ), seven nodes ( $N=7, L=6$ ), eight nodes ( $N=8, L=7$ ), nine nodes ( $N=9, L=8$ ), ten nodes ( $N=10, L=9$ ), eleven nodes ( $N=11, L=10$ ), twelve nodes ( $N=12, L=11$ ), thirteen nodes ( $N=13, L=12$ ), fourteen nodes ( $N=14, L=13$ ), fifteen nodes ( $N=15, L=14$ ), sixteen nodes ( $N=16, L=15$ ), seventeen nodes ( $N=17, L=16$ ), eighteen nodes ( $N=18, L=17$ ), nineteen nodes ( $N=19, L=18$ ), twenty nodes ( $N=20, L=19$ ).
- Middle Section (Hexagonal Lattice Topology):** Shows a hexagonal lattice where each node is connected to three neighbors. The number of links is calculated as  $L = \frac{3}{2}N - \frac{1}{2}$ . Examples shown include a single node ( $N=1, L=0$ ), two nodes ( $N=2, L=1$ ), three nodes ( $N=3, L=2$ ), four nodes ( $N=4, L=3$ ), five nodes ( $N=5, L=4$ ), six nodes ( $N=6, L=5$ ), seven nodes ( $N=7, L=6$ ), eight nodes ( $N=8, L=7$ ), nine nodes ( $N=9, L=8$ ), ten nodes ( $N=10, L=9$ ), eleven nodes ( $N=11, L=10$ ), twelve nodes ( $N=12, L=11$ ), thirteen nodes ( $N=13, L=12$ ), fourteen nodes ( $N=14, L=13$ ), fifteen nodes ( $N=15, L=14$ ), sixteen nodes ( $N=16, L=15$ ), seventeen nodes ( $N=17, L=16$ ), eighteen nodes ( $N=18, L=17$ ), nineteen nodes ( $N=19, L=18$ ), twenty nodes ( $N=20, L=19$ ).
- Right Section (Triangular Lattice Topology):** Shows a triangular lattice where each node is connected to six neighbors. The number of links is calculated as  $L = \frac{6}{2}N - \frac{1}{2}$ . Examples shown include a single node ( $N=1, L=0$ ), two nodes ( $N=2, L=1$ ), three nodes ( $N=3, L=2$ ), four nodes ( $N=4, L=3$ ), five nodes ( $N=5, L=4$ ), six nodes ( $N=6, L=5$ ), seven nodes ( $N=7, L=6$ ), eight nodes ( $N=8, L=7$ ), nine nodes ( $N=9, L=8$ ), ten nodes ( $N=10, L=9$ ), eleven nodes ( $N=11, L=10$ ), twelve nodes ( $N=12, L=11$ ), thirteen nodes ( $N=13, L=12$ ), fourteen nodes ( $N=14, L=13$ ), fifteen nodes ( $N=15, L=14$ ), sixteen nodes ( $N=16, L=15$ ), seventeen nodes ( $N=17, L=16$ ), eighteen nodes ( $N=18, L=17$ ), nineteen nodes ( $N=19, L=18$ ), twenty nodes ( $N=20, L=19$ ).

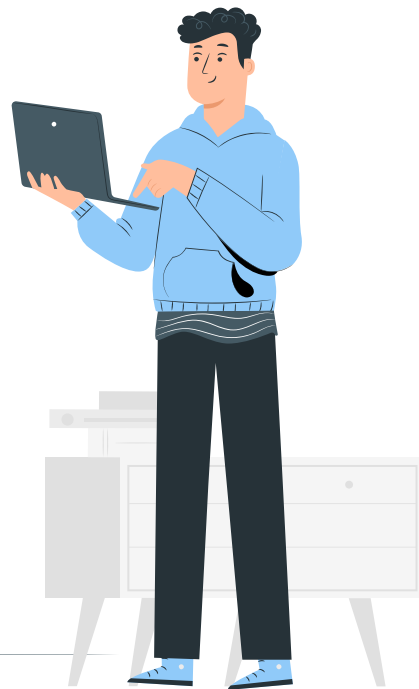
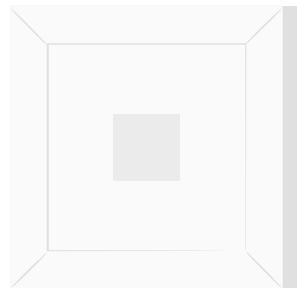
Ya estas listo para continuar aprendiendo sobre contenedores!

# 04



## Comandos

Aprendiendo comandos básicos





# Comandos básicos



En la siguiente parte de la guía vamos a empezar a interactuar con los comandos básicos de docker haciendo lo siguiente:

- Examinar un Dockerfile
- Construir una imagen para el Dockerfile
- Verificar la creación de la imagen
- Lanzar el contenedor de diferentes formas
- Examinar los logs
- Listar los contenedores existentes
- Acceder al contenedor lanzado
- Detenerlo
- Eliminar



# 05 Volumenes y Puertos

Desarrollando con Docker





# Desarrollando con Docker

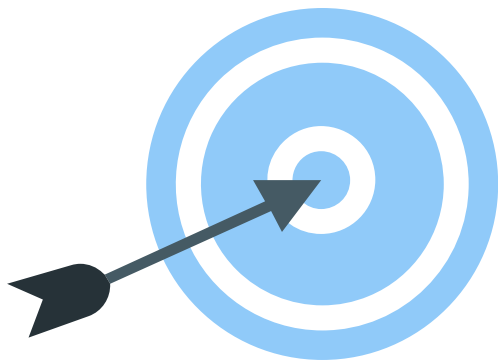


Si queremos desarrollar con contenedores y hacer nuestro día a día más cómodo, debemos aprender cómo exponer puertos y montar volúmenes.

El próximo ejercicio aprenderás:

- Exponer puertos
- Montar un volumen externo a un contenedor
- Modificar el código de nuestra app en nuestro disco y ver el cambio dentro del contenedor

Volvamos a los docs para comenzar



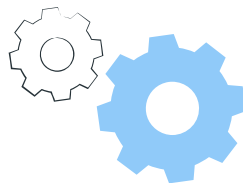
## Recomendación

Explora los docs! (RTFM 😊)

Te recomiendo examinar por tu cuenta:

**run, exec, pull, logs, inspect, start/stop/restart,  
cp, images, ps, rm, stats, build...**

<https://docs.docker.com/engine/reference/commandline/cli/>





# 06 Docker Compose

Define tu aplicación usando YAML



# Docker Compose



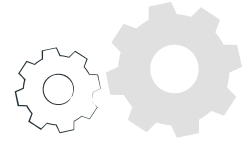
Docker Compose es una herramienta que permite definir y ejecutar aplicaciones multicontenedores. Con Docker Compose defines en un *YAML* los **servicios** de tu aplicación y sus configuraciones. Luego solo ejecutas **docker-compose up** y todos los contenedores serán lanzados.

Vamos a aprender:

- Como se define y ejecuta un `docker-compose.yml`
- Establecer dependencia entre contenedores
- Entender cómo funciona el sistema de DNS desplegado por Docker
- Instalar y explorar el proyecto <https://gdi3d.github.io/learning-to-build>

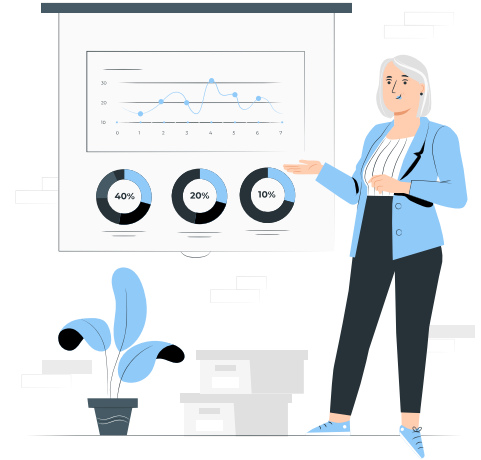


# Próximos pasos



Ahora que ya sabes algo más sobre Docker:

- Comienza a utilizarlo en tu día a día
- Lee y aprende sobre Docker Swarm y Kubernetes



# Gracias!

Preguntas?

Abre un ticket en el repo con la etiqueta **Question**

<https://github.com/gdi3d/abcdocker/issues/new>

-----

LinkedIn <https://www.linkedin.com/in/adrianogalello/>

Calendly <https://calendly.com/adrianogalello/techadvise>

**CREDITS:** This presentation template was created by **Slidesgo**, including icons by **Flaticon**, and infographics & images by **Freepik** and illustrations by **Storyset**.

