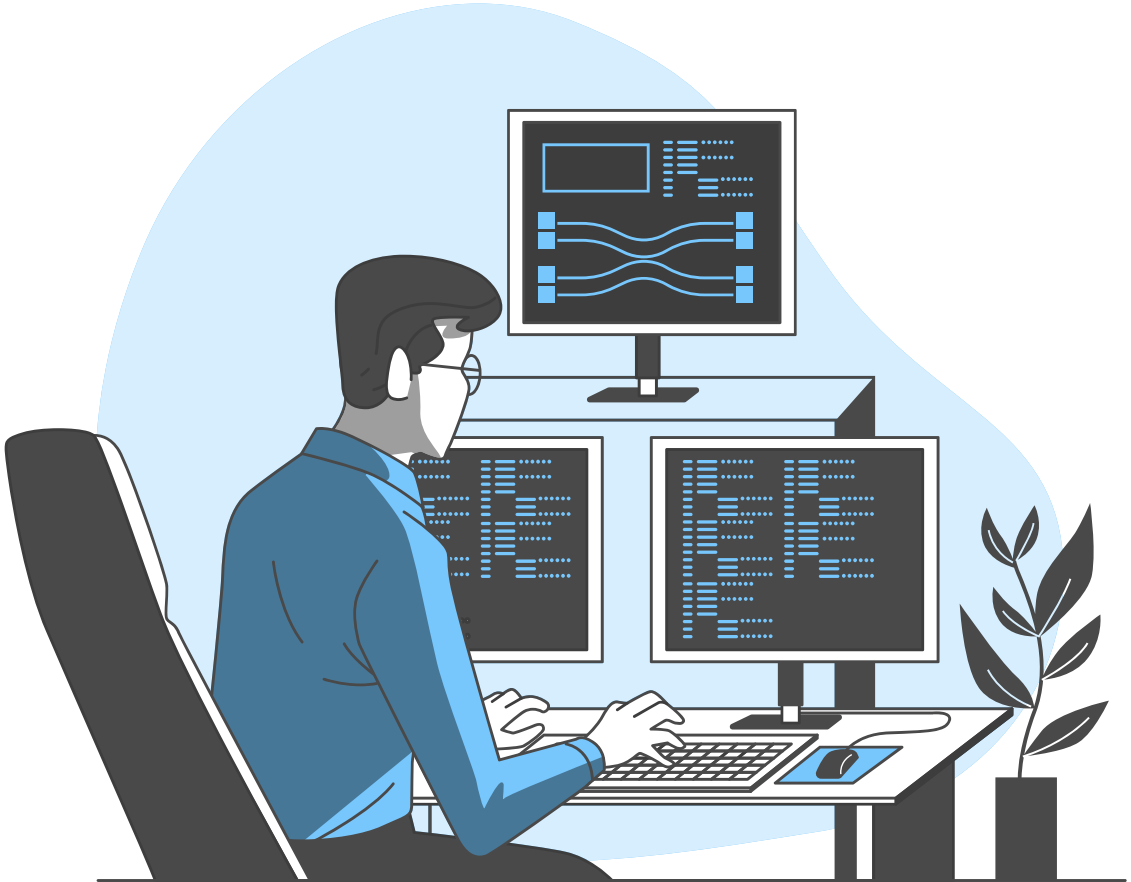


ABCDocker

El ABC de Docker y contenedores

01

Qué es Que?

Diferencia entre Docker y Contenedor

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

Qué es Docker y qué es un contenedor

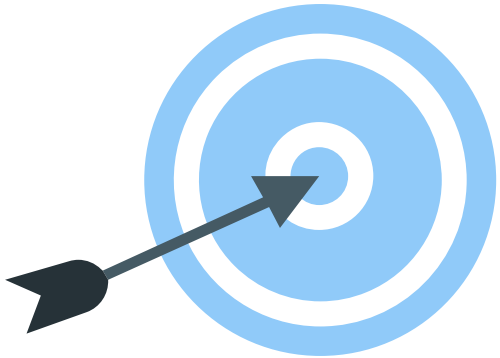
Docker

Es una empresa que desarrolló varios productos enfocados a contenedores

Contenedor

Es una tecnología basada en los cgroups de Linux

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

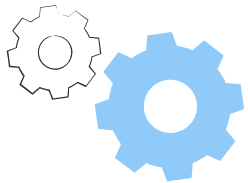


¿Para que?

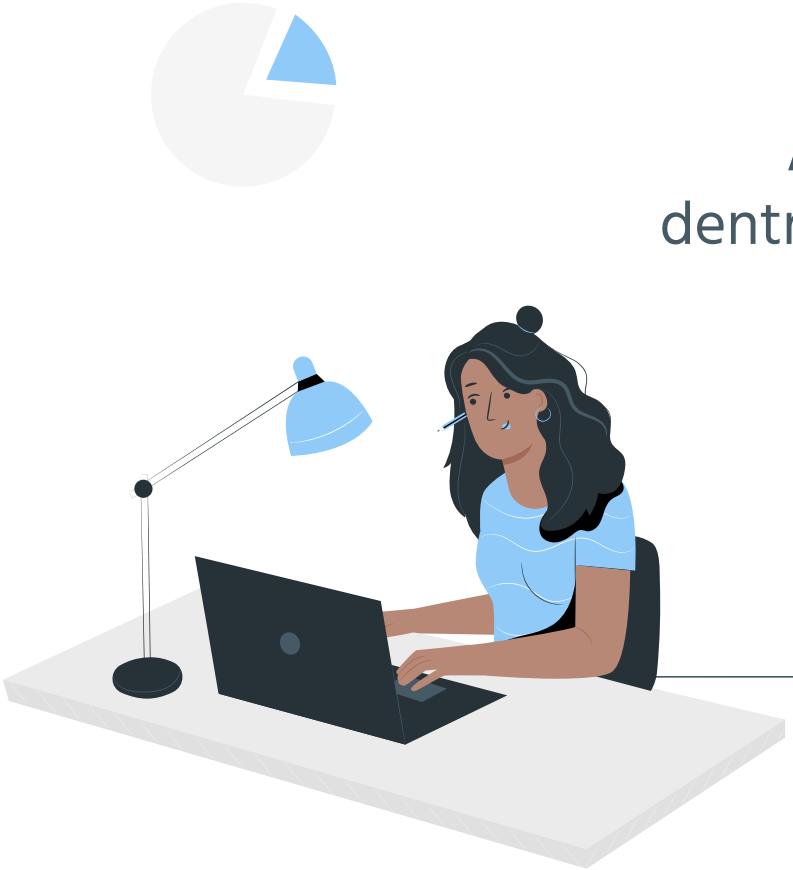
Contenedores

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..



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



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

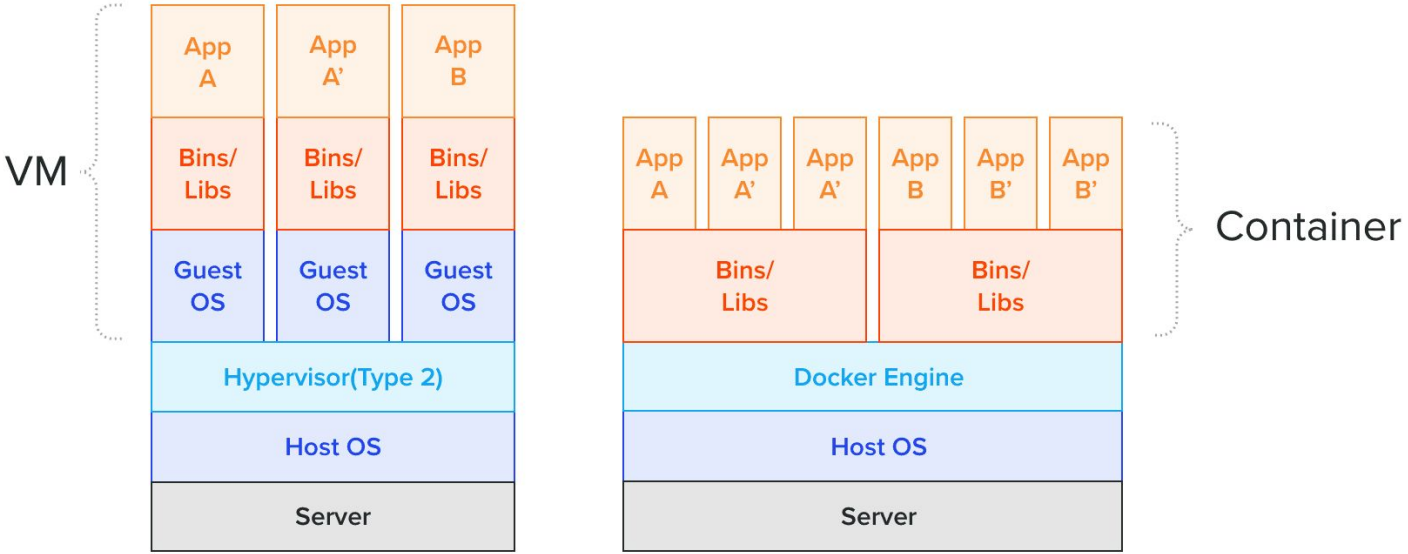
https://gdi3d.github.io/course/docs#/dia-1/intro

02

VM's vs Contenedores

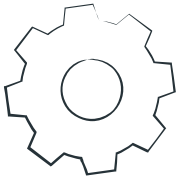
¿Cuál es la diferencia?

Máquinas Virtuales vs Contenedores



Los contenedores no utilizan un hipervisor para funcionar.
El acceso a los recursos del host es controlado por el engine de contenedores (Docker, Podman, Containerd, ...)

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



Ventajas

01

Recursos

Máximo aprovechamiento de los recursos del servidor

02

Operación

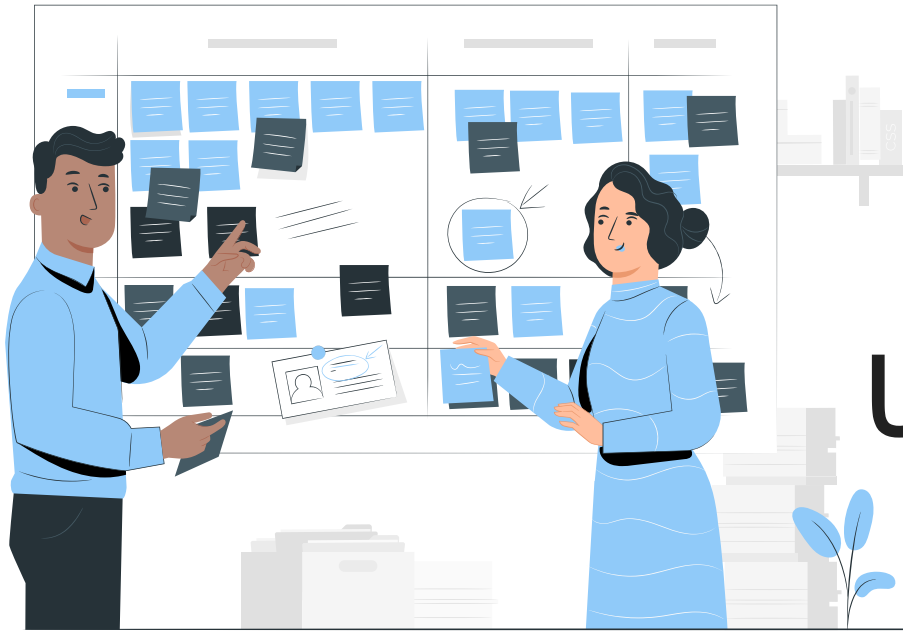
Reduce conflictos de dependencias.
Fácil de actualizar

03

Optimización

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

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



03

Como Se Crea Un Contenedor

Imágenes, la forma de *paquetizar*

Imágenes

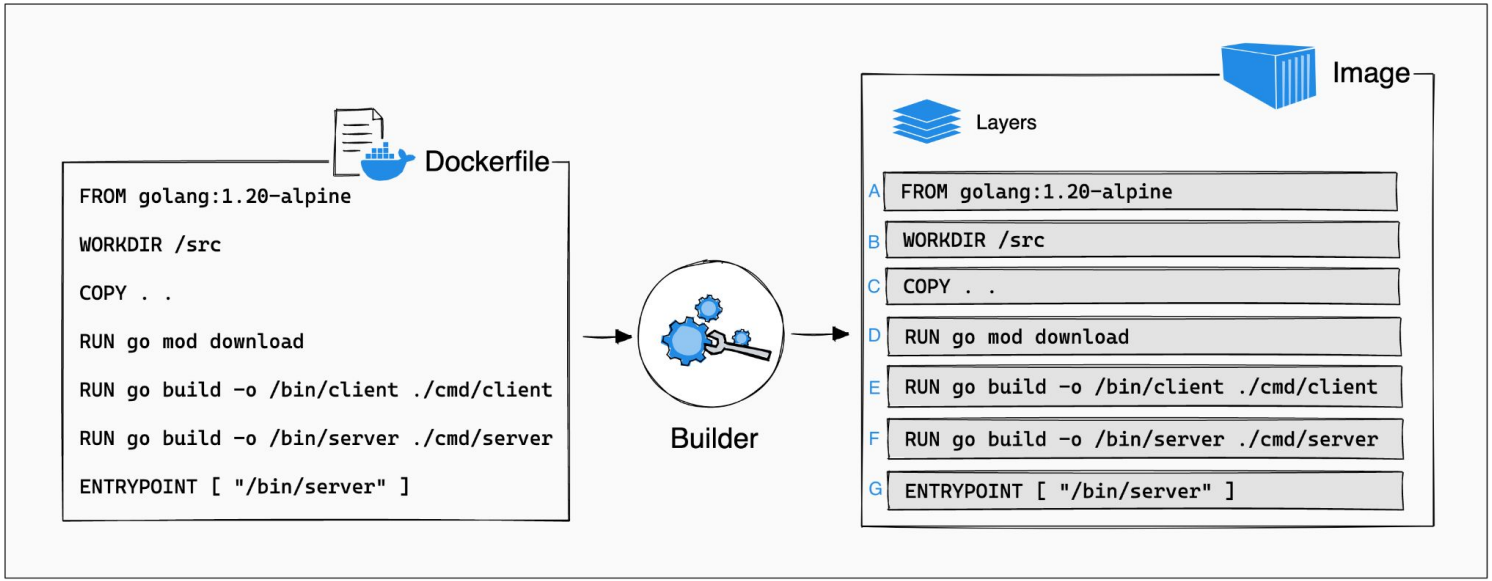
Las imágenes se construyen a través de un archivo nombrado Dockerfile

```
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

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

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

Comprobación de Instalación

Siguiendo los pasos de la guia

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`

```
  _     _       _ _                                  _      _ _
 | |   | |     | | |                                | |    | | |
 | |__ | | ___ | | | ___   __      _____  _ __ | | __| | |
 |  __  |/ _ \ | |/ _ \  \ \ /\ / / _ \| '__| |/ _` | |
 | |  | |  __/ | | (_) |  \ V  V / (_) | |  | | (_| |_|
 |_|  |_|\___|_|_|\___/    \_/\_/ \___/|_|  |_|\__,_(_)

 Ya estas listo para continuar aprendiendo sobre contenedores!
```

https://gdi3d.github.io/course/docs#/dia-1/instalacion

04

Comandos

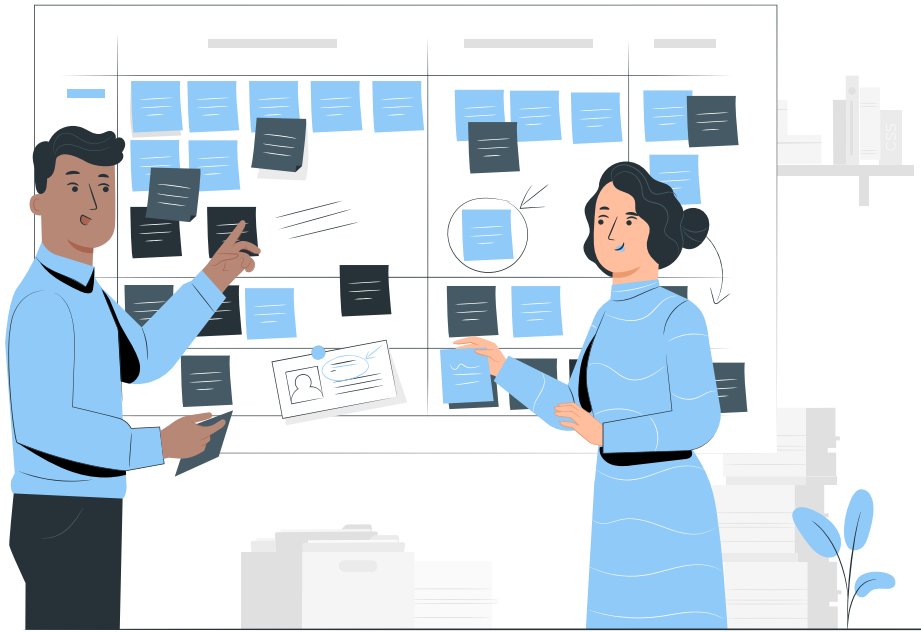
Aprendiendo comandos básicos

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

https://gdi3d.github.io/course/docs#/dia-1/comandos-basicos



05

Volumenes y Puertos

Desarrollando con Docker

Desarrollando con Docker

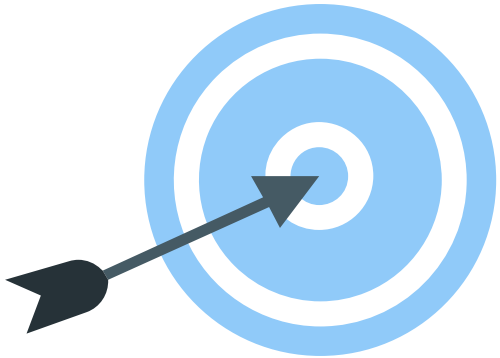
Si queremos desarrollar con contenedores y hacer nuestro día a día más cómodo, debemos aprender cómo exponer puertos y montar volúmenes.

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

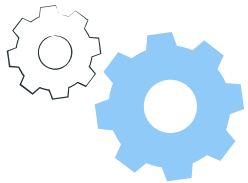
https://gdi3d.github.io/course/docs#/dia-1/desarrollando-docker



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/



https://gdi3d.github.io/course/docs#/dia-1/desarrollando-docker

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:

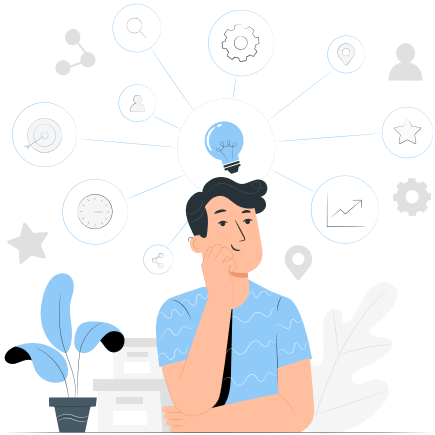
- Como se define y ejecuta un docker-compose.yml
- Establecer dependencia entre contenedores
- Entender cómo funciona el sistema de DNS desplegado por Docker
- Instalar y explorar el proyecto https://gdi3d.github.io/learning-to-build

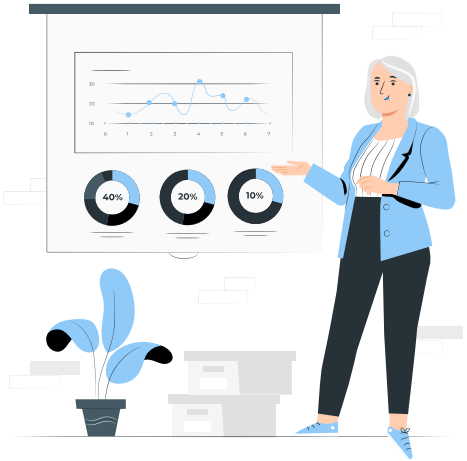
https://gdi3d.github.io/course/docs#/dia-1/docker-compose

Próximos pasos

Ahora que ya sabes algo más sobre Docker:

- Comienza a utilizarlo en tu dia a dia
- 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