**🔹 Módulo 1: Comprender qué es Docker**

**Objetivo:** Entender su arquitectura, propósito y ventajas sobre las máquinas virtuales.

**Conceptos Clave:**

* Docker es una plataforma que permite empaquetar aplicaciones con todas sus dependencias en *contenedores* ligeros y portátiles.
* Componentes principales: **Daemon (dockerd)**, **Cliente (CLI)**, **Imágenes** y **Contenedores**.
* Diferencia principal vs máquinas virtuales: los contenedores comparten el mismo kernel del sistema operativo, lo que los hace más ligeros y rápidos.

**Recursos oficiales:**

* [Docker: ¿Qué es Docker? (oficial)](https://docs.docker.com/get-started/overview/)
* [Guía paso a paso de Docker (español)](https://www.docker.com/resources/what-container/)

**Actividad:**

* Explica en tus propias palabras cómo Docker mejora el proceso de despliegue frente a métodos tradicionales.

**🔹 Módulo 2: Instalación de Docker**

**Objetivo:** Tener Docker operativo en tu entorno.

**Pasos:**

1. **Instalar Docker Desktop o Docker Engine:**
   * Windows/Mac: [Docker Desktop](https://www.docker.com/products/docker-desktop/" \t "_blank)
   * Linux: [Guía de instalación para Linux](https://docs.docker.com/engine/install/)
2. Verifica la instalación:

bash

docker --version

docker run hello-world

1. Consulta información básica:

bash

docker info

**Comando destacado:** docker --version y docker info (muestran la versión instalada y la configuración del sistema).​

**🔹 Módulo 3: Comandos básicos de Docker**

**Objetivo:** Dominar los comandos esenciales para crear, gestionar e inspeccionar contenedores.

**Concepto general:**  
Los comandos Docker siguen esta estructura:

text

docker <comando-de-gestión> <subcomando> [--opciones] <objeto>

Ejemplo:

text

docker container run -it httpd

Significa: ejecutar el contenedor de la imagen httpd en modo interactivo.​

**Comandos clave:**

* docker pull <imagen> → descarga una imagen de Docker Hub.​
* docker run <opciones> <imagen> → crea y ejecuta un contenedor.​
* docker ps → lista contenedores activos.​
* docker images → lista imágenes disponibles localmente.​
* docker stop <id> → detiene un contenedor.​
* docker rm <id> → elimina un contenedor.​
* docker build -t <nombre> . → construye una imagen desde un Dockerfile.​

**Recursos recomendados:**

* [Lista completa de comandos Docker (Kinsta)](https://kinsta.com/es/blog/comandos-docker/)​
* [Comandos Docker con ejemplos prácticos (Hostinger)](https://www.hostinger.com/es/tutoriales/comandos-docker)​
* [Docker Command Cheat Sheet (GitHub)](https://github.com/adrianlois/Docker-CheatSheet)​

**🔹 Módulo 4: Dockerfiles e Imágenes Personalizadas**

**Objetivo:** Crear imágenes personalizadas para desplegar webs u otros servicios.

**Instrucciones del Dockerfile:**​

| **Instrucción** | **Función** |
| --- | --- |
| FROM <image> | Define la imagen base. |
| COPY <origen> <destino> | Copia archivos del host al contenedor. |
| RUN <comando> | Ejecuta comandos durante la construcción. |
| EXPOSE <puerto> | Abre puertos en el contenedor. |
| CMD ["comando"] | Comando que se ejecuta al iniciar el contenedor. |

**Ejemplo básico:**

text

FROM nginx:latest

COPY ./html /usr/share/nginx/html

EXPOSE 80

Construye la imagen personalizada:

bash

docker build -t miweb-nginx .

Ejecuta el contenedor:

bash

docker run -d -p 8080:80 miweb-nginx

**🔹 Módulo 5: Comprendiendo Nginx**

**Objetivo:** Entender qué es Nginx y por qué es ideal para despliegues con Docker.

**¿Qué es Nginx?**

Nginx es un **servidor web y proxy inverso** de alto rendimiento. Se caracteriza por su eficiencia en manejo de múltiples conexiones simultáneas y su bajo uso de memoria.

**Diferencias clave con otros servidores:**

| **Característica** | **Nginx** | **Apache** | **Tomcat** |
| --- | --- | --- | --- |
| Arquitectura | Asíncrona, basada en eventos | Multiproceso, por conexión | Plataforma Java (servlet container) |
| Rendimiento | Muy alto en servir contenido estático | Menor eficiencia con muchas peticiones simultáneas | Ideal para aplicaciones Java EE |
| Uso típico | Servidor web, proxy, balanceo | Sitios dinámicos PHP/WordPress | Aplicaciones Java (Spring, JSP) |

**Recursos recomendados:**

* [Guía oficial de Nginx (English)](https://nginx.org/en/docs/)
* [Documentación de Docker Nginx (Docker Hub)](https://hub.docker.com/_/nginx)
* [Tutorial básico Nginx + Docker en Docker Docs](https://docs.docker.com/samples/nginx/)

**🔹 Módulo 6: Despliegue de un sitio con Nginx + Docker**

**Objetivo:** Poner en marcha una web estática dentro de un contenedor Nginx.

**Pasos:**

1. Crea una carpeta para el proyecto:

bash

mkdir web-nginx && cd web-nginx

mkdir html

1. Dentro de html/index.html escribe:

xml

<h1>¡Hola desde Nginx en Docker!</h1>

1. Crea un Dockerfile como antes:

text

FROM nginx:latest

COPY ./html /usr/share/nginx/html

1. Construye y ejecuta:

bash

docker build -t miweb .

docker run -d -p 8080:80 miweb

1. Accede en tu navegador: [http://localhost:8080](http://localhost:8080/)

**🔹 Módulo 7: Mantenimiento y limpieza**

**Objetivo:** Aprender a mantener limpio tu entorno Docker.

**Comandos útiles:**

* docker system df → muestra el uso de disco por Docker.​
* docker system prune → elimina contenedores e imágenes no usadas.​

**Actividad final:** Escribe un resumen de tu flujo de trabajo Docker + Nginx, incluyendo los comandos usados.

**🔹 Herramientas recomendadas para seguimiento**

* **Notion** o **Obsidian:** para registrar tu progreso y notas.
* **Trello:** crea tarjetas por módulo y marca completados.
* **Google Docs / LibreOffice Writer:** para guardar esta guía y tus proyectos.

**🔹 Evaluación Final (Opcional)**

* Crear un servidor Nginx personalizado con una página HTML y publicarlo en Docker Hub.
* Describir los comandos principales usados y su función