

**PROGRAMACIÓN ORIENTADA A PRUEBAS**

**Revisado por: Ing. Miguel Cardona.**

**Aarón Hernández García.**

**Primeros pasos con Docker**

**¿Qué es Docker?**

**31/03/2025**

**PÉREZ JUÁR HERNANDEZ GARCIA**

**1.2 APLICACIONES DE LA DISTRIBUCIÓN NORMAL**

Docker es una plataforma de código abierto que permite empaquetar, distribuir y ejecutar aplicaciones en contenedores. Un contenedor es una unidad ligera y portable que incluye todo lo necesario para que una aplicación funcione: código, dependencias, configuraciones y sistema de archivos.

**¿Para qué sirve Docker?**

Docker facilita la creación, distribución y ejecución de aplicaciones en diferentes entornos sin preocuparse por problemas de compatibilidad. Sus usos más comunes son:

* **Desarrollo ágil:** Los desarrolladores pueden trabajar en entornos aislados sin conflictos de dependencias.
* **Despliegue eficiente:** Permite trasladar aplicaciones desde la máquina local a servidores o la nube sin cambios en la configuración.
* **Estandarización:** Garantiza que una aplicación funcione igual en cualquier sistema.
* **Escalabilidad:** Facilita la creación y administración de múltiples instancias de una aplicación.

**¿Por qué Docker es esencial en DevOps y desarrollo moderno?**

1. **Portabilidad:** Los contenedores funcionan en cualquier entorno con Docker instalado.
2. **Consistencia:** Evita problemas de "en mi máquina funciona, pero en producción no".
3. **Automatización:** Se integra con herramientas CI/CD para despliegues continuos.
4. **Eficiencia:** Consume menos recursos que una máquina virtual y permite una rápida escalabilidad.

**Conceptos clave en Docker**

**1. Imágenes**

Las imágenes de Docker son plantillas inmutables que contienen el sistema operativo, bibliotecas y el código necesario para ejecutar una aplicación. Se crean a partir de un **Dockerfile** y sirven como base para los contenedores.

**2. Contenedores**

Son instancias en ejecución de una imagen. Son ligeros, portables y se pueden crear, detener o eliminar rápidamente sin afectar el sistema anfitrión.

**3. Volúmenes**

Son mecanismos de almacenamiento que permiten a los contenedores guardar datos de forma persistente, incluso si el contenedor se detiene o elimina. Se usan para compartir datos entre contenedores o con el sistema anfitrión.

**4. Redes**

Docker proporciona redes para permitir la comunicación entre contenedores y con el mundo exterior. Existen diferentes tipos, como **bridge** (por defecto), **host** (usa la red del host) y **overlay** (para múltiples hosts en un clúster).

**Arquitectura de Docker**

Docker se basa en una arquitectura cliente-servidor, compuesta por los siguientes elementos:

1. **Docker Client:** La interfaz de usuario que permite interactuar con Docker mediante comandos.
2. **Docker Daemon (dockerd):** Se ejecuta en segundo plano y administra la construcción, ejecución y supervisión de los contenedores.
3. **Docker Images:** Plantillas inmutables que sirven como base para crear contenedores.
4. **Docker Containers:** Instancias en ejecución de imágenes de Docker.
5. **Docker Registry (Docker Hub):** Repositorio central donde se almacenan y comparten imágenes de Docker.

**Diferencia entre Imagen y Contenedor**

* **Imagen:** Es un archivo inmutable que contiene el sistema operativo, bibliotecas y aplicaciones necesarias para ejecutar un contenedor.
* **Contenedor:** Es una instancia de una imagen en ejecución. Se puede modificar, detener y eliminar sin afectar la imagen base.

**Docker Hub**

Docker Hub es un registro de imágenes en la nube donde los desarrolladores pueden encontrar y compartir imágenes preconfiguradas. Permite a los equipos de trabajo gestionar versiones de sus aplicaciones y acceder a imágenes oficiales de tecnologías populares.

**Dockerfile**

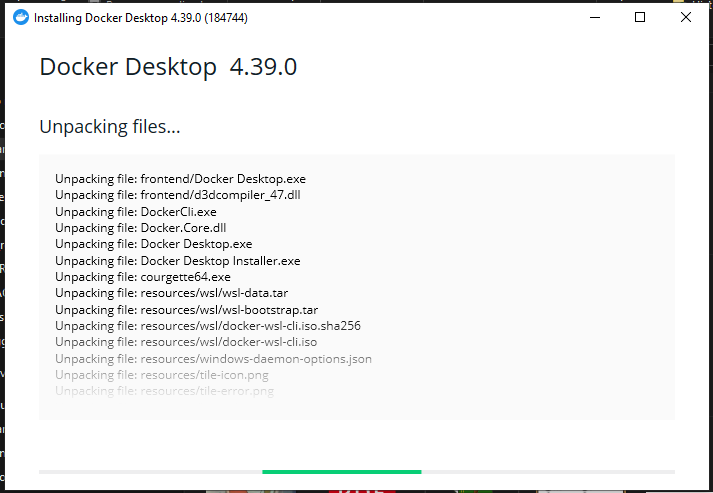
Un Dockerfile es un archivo de texto que contiene una serie de instrucciones para construir una imagen de Docker de manera automatizada. Algunas de las instrucciones comunes en un Dockerfile son:

* FROM: Define la imagen base.
* RUN: Ejecuta comandos dentro de la imagen durante su construcción.
* COPY o ADD: Copia archivos desde el host al contenedor.
* EXPOSE: Define puertos que estarán disponibles.
* CMD o ENTRYPOINT: Define el comando que se ejecutará cuando el contenedor inicie.

**Comandos Básicos de Docker**

A continuación, se presentan algunos de los comandos esenciales para trabajar con Docker:

* **Descargar una imagen desde Docker Hub:** docker pull <nombre\_de\_imagen>
  + Ejemplo: *docker pull nginx*
* **Ejecutar un contenedor:** docker run -d --name <nombre\_contenedor> <nombre\_de\_imagen>
  + Ejemplo: *docker run -d --name webserver nginx*
* **Listar contenedores en ejecución:** *docker ps*
* **Ejecutar un comando dentro de un contenedor:** docker exec -it <nombre\_contenedor>
  + *Ejemplo: docker exec -it webserver bash*
* **Detener un contenedor:** docker stop <nombre\_contenedor>
  + *Ejemplo: docker stop webserver*
* **Eliminar un contenedor:** docker rm <nombre\_contenedor>
  + *Ejemplo: docker rm webserver*

**Instalando Docker**





