**Docker**



Docker es una plataforma que facilita la creación, prueba e implementación de aplicaciones. Esta herramienta permite generar contenedores: unidades estandarizadas de software empaquetado que contienen todo lo necesario para su ejecución, desde el código hasta las bibliotecas.

Gracias a Docker, las aplicaciones pueden implementarse en diversos entornos con la garantía de que su código se ejecutará correctamente. Cabe destacar que Docker es una plataforma de código abierto



La arquitectura de contenedores Docker es de tipo cliente-servidor.

**Historia De Docker**

Docker tiene su origen en la compañía **dotCloud**, que se dedicaba al desarrollo de **PaaS** (**plataforma como servicio**). El proyecto fue impulsado por **Solomon Hykes**, con contribuciones de **Francois-Xavier Bourlet**, **Andrea Luzzardi** y otros ingenieros de la empresa. Podríamos decir que **Docker** representa la evolución de una tecnología previa de **dotCloud**, basada en **Cloudlets** y otras iniciativas de código abierto.

En 2013, Docker se liberó como una plataforma de código abierto. Al año siguiente, comenzó a utilizar su propia biblioteca llamada **libcontainer**. Con el tiempo, **dotCloud** **Inc**. se transformó en **Docker** **Inc.**, la empresa que actualmente promueve el proyecto y ofrece una versión comercial de la plataforma.



Las imágenes se almacenan en un registro Docker (Docker Registry).

**Cómo Funcionan Los Contenedores**

Docker se basa en la **producción de** **contenedores**. Estas unidades ejecutables funcionan como **entornos virtuales**, empaquetando tanto el código de la aplicación como sus dependencias y bibliotecas. De esta manera, el código puede ejecutarse en un escritorio, en la nube u otros entornos.

Es importante destacar que los contenedores utilizan un **modo de virtualización** **del sistema operativo** que aprovecha las propiedades del kernel para aislar procesos y controlar el acceso a la memoria y la CPU. A menudo se comparan con las **máquinas virtuales** (**VM**), pero los contenedores ofrecen funcionalidades similares (disponibilidad, escalabilidad, aislamiento de aplicaciones) con ventajas como mayor eficiencia, productividad mejorada y menor peso.



Cada image layer (capa de imagen) constituye una versión diferente.

**Características De Docker**

**Docker** se basa en el **kernel de Linux** y utiliza funciones como los espacios de nombres (**Namespaces**) y los grupos de control para lograr la separación de procesos y permitir su ejecución independiente.

En términos de implementación, **Docker** opera con un modelo basado en imágenes. Cada I**magen Docker** contiene el código de la aplicación, sus bibliotecas y otros componentes necesarios. Esto facilita la compartición de servicios y aplicaciones en múltiples entornos.

Aunque es posible crear imágenes desde cero, lo más común es obtenerlas de repositorios comunes como **Docker Hub**. A partir de una imagen base, se pueden generar diferentes versiones estructuradas en capas.

La tecnología cliente-servidor utilizada para construir y contenerizar aplicaciones se llama **Docker Engine**. Este motor permite llevar a cabo todas las acciones relacionadas con la ejecución de aplicaciones basadas en contenedores. Sus componentes incluyen **Docker Daemon** (encargado de gestionar imágenes Docker, contenedores y otros elementos), **Docker CLI** (interfaz de línea de comandos para comunicarse con Docker Daemon) y **API REST** (para interactuar con Docker Daemon).

Por otro lado, **Docker Compose** permite ejecutar varios contenedores como un solo servicio. Primero, se crea un **Dockerfile** para definir el entorno, luego se especifican los distintos servicios y, finalmente, se utiliza el comando **Docker Compose** para iniciar la aplicación.

Además, al instalar **Docker Desktop**, es posible utilizar Docker en **Windows**, **Linux** o **Mac**. Esta interfaz facilita la gestión de contenedores, imágenes Docker y aplicaciones en la computadora.

**Ventajas Y Desventajas**

**Docker** ofrece numerosas ventajas. No solo permite trabajar con diferentes **sistemas operativos**, sino también con diversos **lenguajes de programación**.

Gracias a **Docker**, los programadores pueden estar seguros de que su código funcionará en distintos entornos debido a los contenedores (que, como mencionamos anteriormente, ocupan menos espacio que las **máquinas virtuales**). Al almacenar cada aplicación en un contenedor aislado, la gestión de la seguridad es más sencilla y se evita la propagación de virus.

Además de la rápida implementación, **Docker** ofrece opciones de restauración mediante las **capas de imágenes**, **modularidad** a través de **microservicios** y una **portabilidad** simplificada.

Entre las desventajas, se encuentra la necesidad de **aprender la línea de comandos**, ya que **Docker** no ejecuta aplicaciones con **GUI** (**Interfaz Gráfica de Usuario**). También se debe considerar que la velocidad es menor en comparación con un servidor físico, y la ejecución de los contenedores se produce en el sistema operativo del anfitrión, lo que conlleva ciertos riesgos de seguridad.