# DOCUMENTACION

# TEORIZACION SOBRE QUE ES DOCKER

TDG-SI-21349-2025-02-13

Marco teórico inicial

Comenzamos instruyéndonos acerca de que es Docker y sus diferencias con VMs (Virtual Machines), donde se nos explica que Docker es una tecnología donde por medio del kernel de Linux y los grupos de control se crean contenedores, donde cada uno contiene únicamente los valores necesarios para ejecutar un programa, servicio o aplicación especifico llamados “contenedores”.

En términos generales podemos observar en el siguiente grafico cuales son las diferencias entre las VMs y los Contenedores (ver ilustración 1):

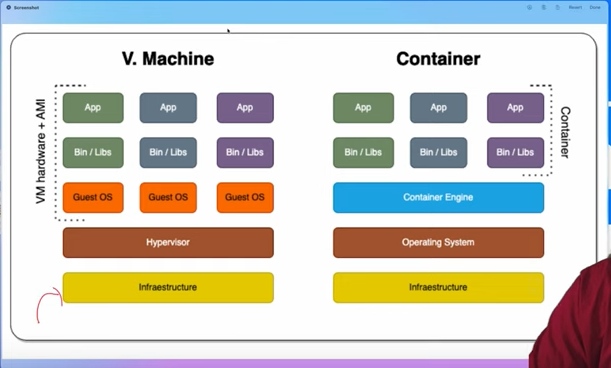


Ilustración . Diferencias entre las VMs y los contenedores. Tomado de: <https://www.youtube.com/watch?v=AquOM-ISsnA&list=PLQhxXeq1oc2n7YnjRhq7qVMzZWtDY7Zz0>.

En nuestra ilustración 1 podemos observar como en una maquina virtual se tiene una infraestructura común pero se puede modificar y dividir el sistema operativo a utilizar (por medio de software creador de máquinas virtuales), lo cual hace que se tengan que destinar una cantidad importante de recursos en cada maquina para conseguir utilizar en las diferentes maquinas virtuales un sistema operativo, mientras que en el caso de los contenedores se tiene una infraestructura y un sistema operativo igual pero que por medio de un software de control de contenedores hace que las librerías, apps y procesos se puedan dividir, esto hace que un controlador de contenedores sea mas beneficioso, ya que utilizando los mismos recursos no recarga el procesador para utilizar diferentes sistemas operativos y a cambio nos da una mejor velocidad y control de las aplicaciones y servicios que se ejecutan en el espacio designado.

Docker a diferencia de otras tecnologías tambien de contenedores nos da un beneficio extra que es el fraccionamiento de los procesos teniendo cada contenedor sus propios procesos, librerías y mas a diferencia de tecnologías como LXC donde solo serían las librerías como vemos en el siguiente grafico (ver ilustración 2):

Escala de tiempo

El contenido generado por IA puede ser incorrecto.

Ilustración . Diferencias entre LXC y Docker, tomado de: <https://www.redhat.com/es/topics/containers/what-is-docker>.

# Adaptar el espacio de Linux Ubuntu

Instalación de librerías

# Instalación de Docker

Ahora continuamos con la instalación de Docker a nuestro Linux, en nuestro caso mas especifico es un Linux Ubuntu Focal 20.04, por lo cual, procedemos a buscar como instalarlo correctamente en la documentación de Docker (Documentacion de Docker en Ubuntu: <https://docs.docker.com/engine/install/ubuntu/>).

Ahora, en la documentación se nos hace mención de que antes de la instalación de Docker se debe instalar el repositorio de Docker para nuestra computadora, por lo que procedemos a instalarlo siguiendo los comandos en la terminal (ejecutando como root o con “sudo” antes de los comandos a ejecutar)

Texto

El contenido generado por IA puede ser incorrecto.

Usamos los diferentes comandos paso por paso para descargar apt

Texto

El contenido generado por IA puede ser incorrecto.

Después de instalar apt podemos proceder a instalar Docker Engine, en nuestro caso hay un pequeño atajo que nos ahorra este proceso y es utilizar un código que crea un script de instalación de Docker y las herramientas necesarias para su funcionamiento, el código del script es el siguiente:

*curl -fsSL https://get.docker.com -o get-docker.sh && sh get-docker.sh*

Tomado de: <https://pabpereza.dev/docs/cursos/docker/Instalacion>

Ejecución y descarga de Docker

Texto

El contenido generado por IA puede ser incorrecto.

Prueba para observar que Docker esta correctamente instalado

Texto

El contenido generado por IA puede ser incorrecto.

(Hasta aquí tendríamos nuestro primer día de desarrollo)

TDG-SI-21349-2025-02-14

Prueba contenedor servidor de Nginx ejecución y instalación en el servidor

Texto

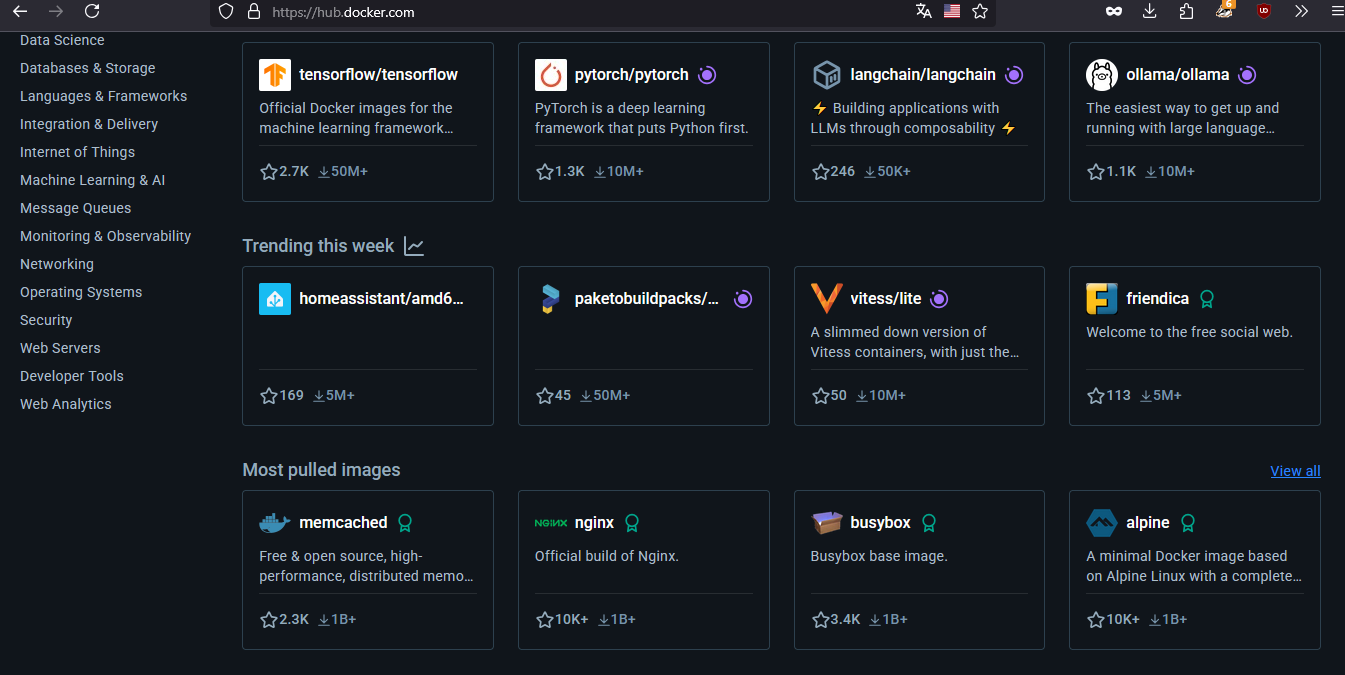
El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

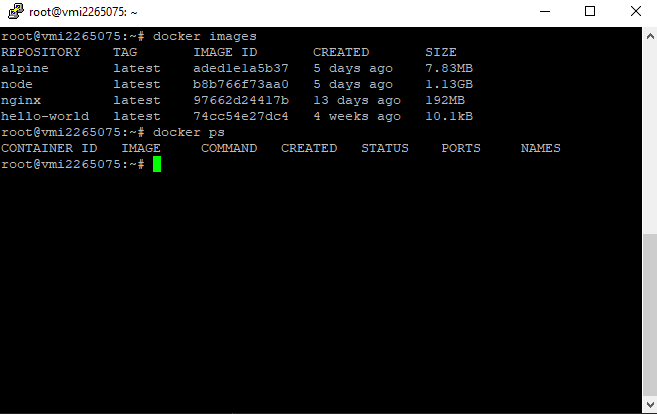
Procedemos aplanear un poco como vamos a estructurar nuestros contenedores, llevamos a usar herramientas como flask, Python o node js.

Y procedemos a descargarlos, haciendo uso de <https://hub.docker.com/> donde podemos encontrar las diferentes herramientas y como descargarlas en nuestro Docker como imágenes.



TDG-SI-21349-2025-02-17

Después de descargar todas las librerías



Procedemos a hacer algunas pruebas.

TDG-SI-21349-2025-02-25

Ahora tenemos que agregar apache y hacer la creación de algunos usuarios para que nuestros estudiantes puedan hacer uso de un espacio propio en el servidor, con sus límites, para empezar por la parte de la creación de usuarios podemos crearlos con el siguiente comando

sudo useradd -m -s /bin/bash -g grupo\_nombre -p $(openssl passwd -1 'contraseña') nombre\_usuario

haciendo uso de este podemos crear un usuario añadirlo a un grupo, una contraseña cifrada y un nombre al usuario todo en un solo comando, hay que tener creado previamente los grupos con el siguiente comando

sudo groupadd developers

como lo podemos ver en esta captura ambos comandos funcionan perfectamente

//aca iría el screenshot del comando trabajando

Ahora después de agregar un par de usuarios en los diferentes grupos creados procedemos instalar apache2 apoyandonos en Docker hub

//instalación de apache

Ahora para poder usar correctamente apache en nuestro servidor y que este contacte con nuestros usuarios previamente creados debemos modificar un poco su instalación en los contenedores, pero antes de esto crearemos con un script a todos nuestros usuarios creados (sin ir uno por uno) hacemos lo siguiente

|  |
| --- |
| for user in $(ls /home); do  echo "Creando public\_html para: $user"  mkdir -p /home/$user/public\_html  chmod o+rx /home/$user /home/$user/public\_html  done |

Este script te crea el mismo archivo a todos tus usuarios, ademas de que por la terminal te muestra cuando lo hace, a quien se le ha creado y asi tambien poder verificar de forma mas rápida.

Faltaría poner un poco el set up de Linux, proxys y demás, eso lo puedo hacer en un contenedor esto para cumplir con “ la administración básica del server”

# Instalación de servidores de juego

Empezamos planteando dos servidores de juegos que sean de juegos muy conocidos como lo pueden ser minecraft y terraria. Por lo que, primero tenemos que documentarnos acerca de como montarlo, por lo que indagamos en dos webs que nos apoyan y nos explican como podemos hacerlo, iniciando con minecraft:

<https://docker-minecraft-server.readthedocs.io/en/latest/>

<https://hub.docker.com/r/itzg/minecraft-server> <- web donde sacamos nuestra versión de minecraft para montar nuestro servidor

ahora vamonos a nuestro servidor, empezamos creando una carpeta para el proyecto, vamos a llamarlo “minecraft-server”

Forma

El contenido generado por IA puede ser incorrecto.

Ahora, lo siguiente y el paso final en la construcción del servidor es hacer la construcción de nuestro Docker-compose.yml que es un archivo donde vamos a poner que imágenes y de donde las vamos a extraer, versiones, versión de Docker, y algunos datos requeridos para el correcto funcionamiento de nuestro servidor, nos queda algo como el siguiente archivo

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Deshabilitamos el Online mode para darle acceso a cualquier persona que tenga acceso a la Ip, otra cosa es que se limita la memoria a 2gigas de RAM y el puerto habilitado (en este caso el 25565. Y eso seria, hacemos un Docker compose up y tenemos nuestro servidor en la ip

212.56.46.43:25565

El proceso queda en nuestro Docker algo asi



Al introducir esto en nuestra sesión de minecraft que tiene que estar en la versión 1.20.6 (una versión ya bastante moderna), podemos ingresar normalmente y jugar con un numero máximo de 20 personas (limite que nos da la imagen por defecto pero se puede ampliar dependiente de las necesidades)

\*\*Imagen del juego captura a añadir servidor y dentro del servidor mismo

Tambien a través de la consola podemos monitorear y ver los jugadores activos, baneos y demás cosas, dar rol de administrador y demás cosas

Comando para entrar a la consola del contenedor:

*docker exec -it mc1122 rcon-cli*

\*\*caps con dos jugadores adentro, dar admin a uno y asi  
  
  
  
  
  
Ahora pasando al servidor de terraria para poder hacer esto seguimos el mismo proceso, primero buscamos en Docker hub una imagen que nos genere un servidor,

En nuestro caso hay varias inactivas por lo que hacemos uso de la imagen de hexlo

<https://github.com/hexlo/terraria-server-docker>

Hacemos nuestro Docker compose.yml



Hacemos Docker compose up

Y ya nuestro servidor esta montado asi lo podemos ver en los procesos de docker



La idea con este juego es ponerlo en la ultima versión ya que normalmente se actualiza solo y no te deja volver a versiones anteriores.

Con el juego abierto es tan sencillo como introducir nuestra ip

\*\*screnshoot ip

Después puerto

\*\*screnshoot apartado puerto

Y ya estamos adentro, una capa extra que no fue agregada pero se puede es ponerle alguna contraseña para que solo aquellos con la clave entren nada mas, pero en nuestro caso no nos interesa

\*\*screnshoot adentro del servidor

En términos generales ya en nuestro Docker ps podemos ver como podemos fragmentar nuestra computadora por medio de Docker en diferentes contenedores, cada uno trabajando de forma individual o conjunta (en el caso de una network)

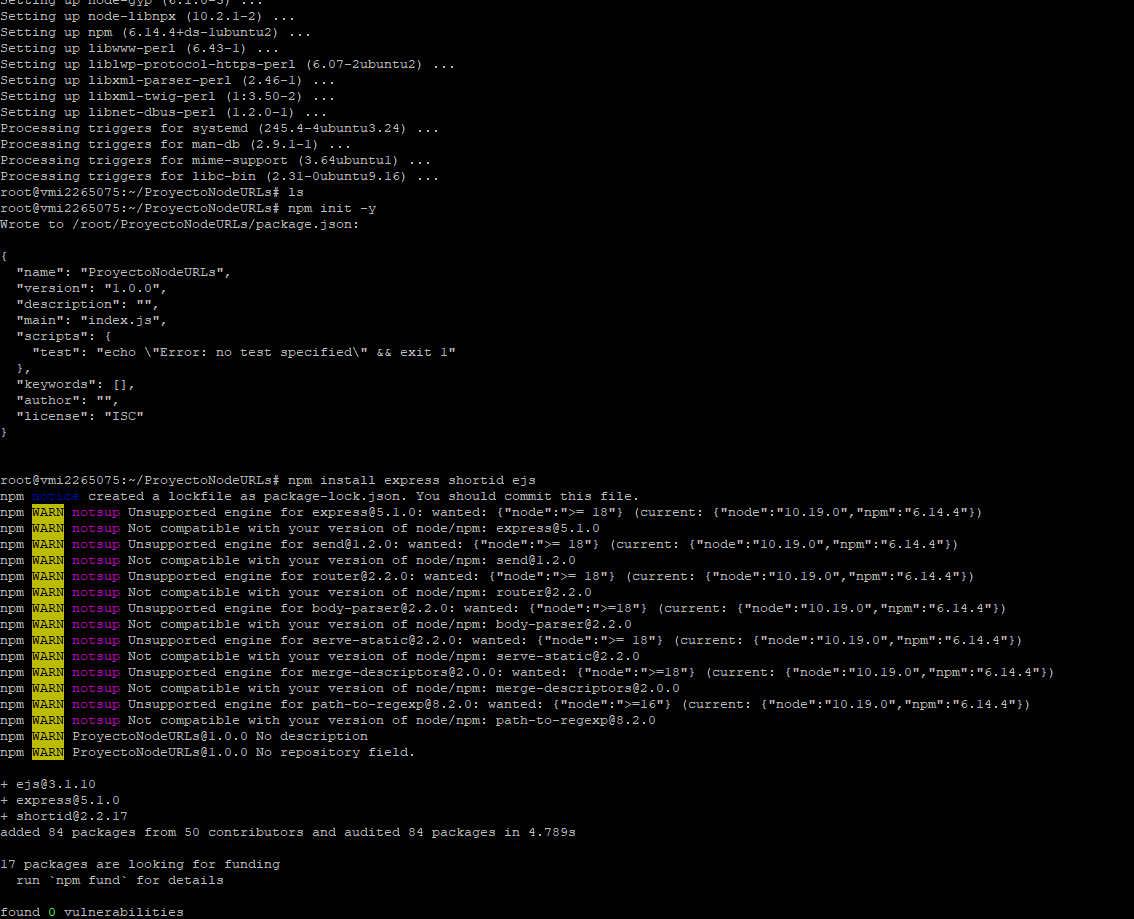
Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

Los diferentes programas que aquí se ven estan en la misma computadora en espacios aislados y algunos trabajando de forma conjunta (ambos “proyecto\_docker-api” y “proyecto\_docker-apache” estan en constante contacto siendo ambos dos contenedores diferentes)

# Aplicación sencilla de Node Js

Npm install



Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.