**DOCKER**

Text, letter

Description automatically generated

Diagram

Description automatically generated

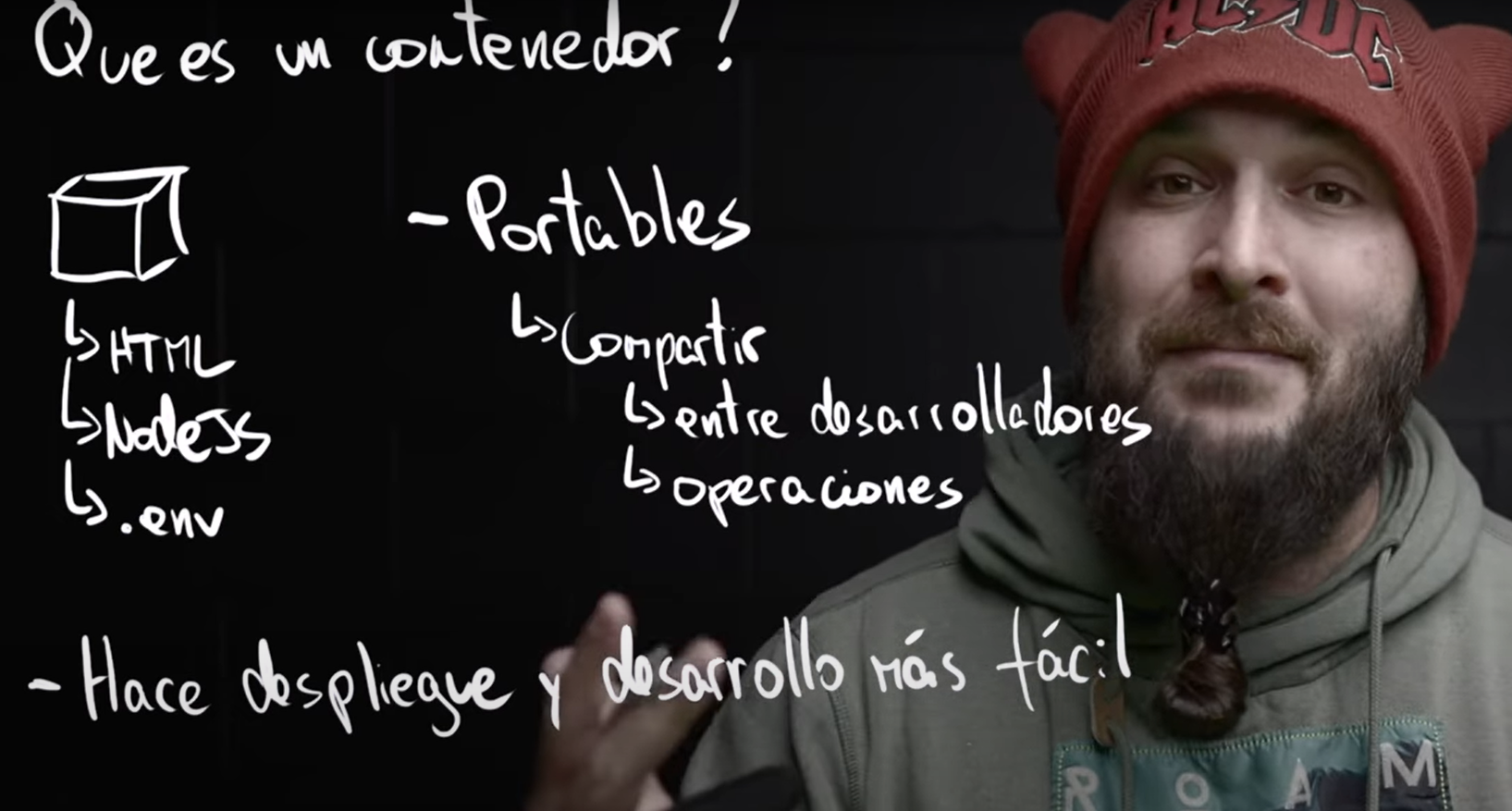
Al tener una capa menos, Docker es más rápido que una maquina virtual.

Text

Description automatically generated

Timeline

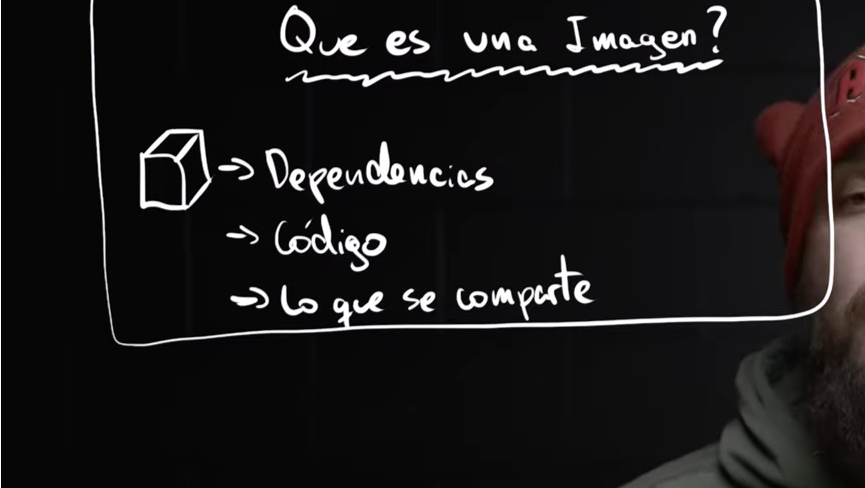
Description automatically generated



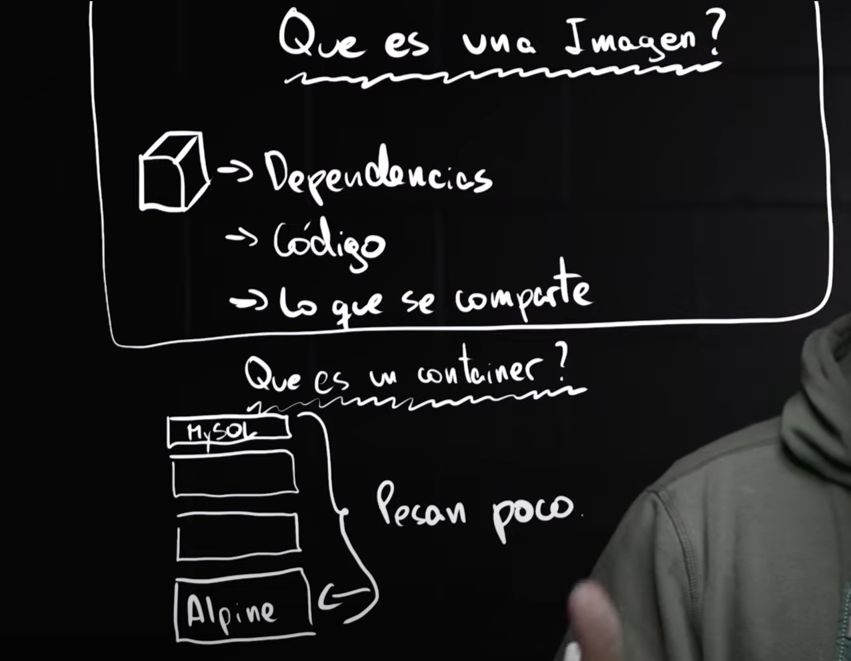
En un contenedor podemos tener el código, de cualquier lenguaje que conozcamos y le agregamos que podemos tener un archivo .env ( Que contiene las variable de entorno )

Text

Description automatically generated



Un container, son capas de imágenes



Text

Description automatically generated

* Para usar Docker lo primero es instalar Docker desktop.

<https://www.docker.com/products/docker-desktop/>

* Hay que tomar en cuenta la versión del sistema a la hora de instalar
* <https://hub.docker.com/> nos ayuda a tener y poder usar mas de 100 mil contenedores que hay disponibles y usar su respectiva imagen.

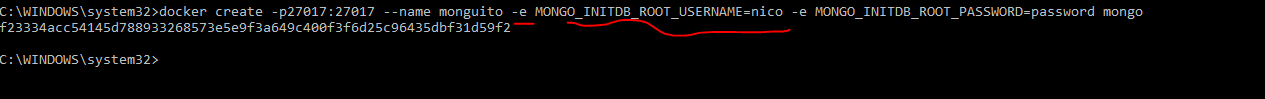
Comandos de Docker.

* **docker images** : Las imágenes que tenemos descargada en nuestra máquina.
* **Docker pull node:** para descargar una imagen**,** node es (la imagen que vamos a descargar) puede ser cualquier otra imagen, podemos indicar la versión pero si lo dejamos sin nada descargará la última versión. Para indicar la versión sería (**Docker pull node:18**).
* **Docker image rm node:**18: Aquí este comando funciona para eliminar imágenes.
* **Docker create mongo:** Esto es para crear un contenedor y usamos la imagen como base para crear el contenedor en este caso la imagen que usamos es **(mongo).**
* **Docker container créate mongo:** Es otra forma de crear, pero más larga.
* **Docker start (id del container):** De esta forma corremos el contenedor de Docker.
* **Docker ps:** para ver los contenedores que están corriendo y datos a lugar.
* **Docker stop (Id del contenedor):** Para detener el contenedor.
* **Docker ps -**a: Muestra todos los contenedores que tenemos aún no se estén corriendo
* **Docker créate –**name (name que le pondrems) mongo: Aquí le ponemos el nombre que queremos que tenga el contenedor y al final la imagen que usará como base está.
* **docker create -p27017:27017 --name monguito mongo**: creamos y mapeamos un contenedor con un puerto y para hacerlo usamos **-p** el primer puerto es el de la maquina donde tenemos instalado Docker que habilitamos para ese uso y el segundo es el puerto del contenedor que vamos a mapear con el de la máquina.
* **docker create -p27017 --name monguito mongo**: creamos y mapeamos un contenedor con un puerto y para hacerlo usamos **-p** si solo ponemos el puerto del contenedor el se encarga de mapearlo en un puerto que encuentre disponible
* **Docker logs (Id o nombre del contendor):** Se ve todos los logs de ese contenedor
* **Docker logs –follow (Id o nombre del contendor):** Mantiene los logs activos y escuchando los cambios del contenedor
* **Docker run (imagen):** Primero busca si encuentra la imagen de lo contrario la descarga, crea el contenedor e inicia el contenedor, es más rápido de hacerlo.
* **Docker run -d (imagen):** el -d hace que los logs no se queden corriendo.
* **Docker network ls:** Para ver todas las redes que tenemos (las redes) son para agrupar varios contenedores así se comunican entre ellos.
* **docker network create (nombre de la red):** Aquí creamos una nueva red.
* **docker network rm (nombre de la red):** Aquí eliminamos una red.
* **docker build -t miapp:1 .:** Nos permite crear imágenes en base a un archivo **Dockerfile q**ue nosotros hallamos creado antes (**miapp:1**) nombre de la imagen y su tag, y el (.) indicando el archivo donde está el **Dockerfile** .
* **docker create -p27017:27017 --name monguito --network mired:** Si queremos descargar una imagen y agregarla a una red solo ponemos **(--network)** y le ponemos la red que acabamos de crear o a donde queremos alojar esa imagen.
* **Docker compose up:** Para correr el compose , pero debemos tener listo el archivo **Docker-compose.yml y** estar en la ruta correspondiente**.**
* **Docker compose down:** Elimina todo lo relacionado (Imágenes y contenedores) que creó.
* **Docker compose -f Docker-compose.dev.yml up:** Indicamos el coompose en base al archivo customizado

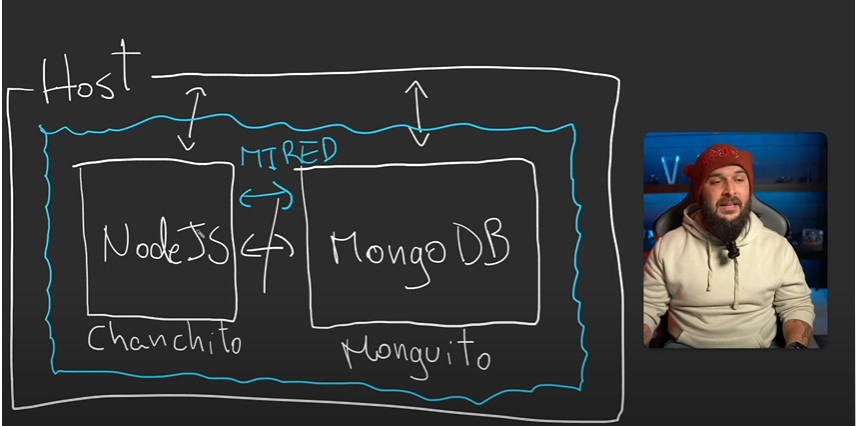
Ejemplo de descargar una imagen de Mongo

**docker create -p27017:27017 --name monguito --network mired -e MONGO\_INITDB\_ROOT\_USERNAME=nico -e MONGO\_INITDB\_ROOT\_PASSWORD=password mongo**

**NOTA**: Cada vez que buscamos una imagen en dockerhub es importante leer la documentación para así ver si hay alguna configuración que aplique para lo que queremos hacer. Y tomar en cuenta que no todos los contenedores se configuran igual. En este caso le aplicamos algunas variables de entorno con (-e) para agregar user and pass.



// Para entrar en este caso la aplicación de Node en Docker debemos agregarle un archivo llamado **dockerfile y** este debe ser su nombre y dentro agregamos las config del contenedor.



**Nota: No necesariamente porque creemos contenedores se van a comunicar entre ellos, debemos crear una red para que estos se puedan comunicar uno con otros. Y podemos perfectamente tener varias redes.**

1. Creamos una image en base a un docker file.
2. Creamos un contenedor en base a una imagen que Descargamos y la ponemos en la red que creamos.
3. Agregamos un contenedor en base a la imagen que creamos.

+Text

Description automatically generated

// Nota: Habrá momentos que por el chip M1 de mac no funcione algo como **Docker pull mysql y le indicamos la plataforma que queremos descargar.**

Text

Description automatically generated

Text

Description automatically generated

Una vez creamos el contenedor nos crea un Id para el mismo.

A picture containing text, blackboard

Description automatically generated

Hay que tomar en cuenta que hay que mapear para poder usarlo un puerto de la maquina que estoy usando con un puerto del contenedor contenedor.

Text

Description automatically generated

Podemos hacer el Docker run con el alias, el puerto etc.

**Docker Compose**

Nos ayuda a automatizar los procesos.

**Volumes**

Ayuda a la persistencia de los datos, existen tres tipos:

* **Anónimo**: Solo se indica la ruta, Docker decide donde guardarla, pero no se puede referenciar estos datos a otro contenedor nuevo
* **De anfitrión o host,** donde uno decide que carpeta montar y donde montarla.
* **Nombrado:** Es como el anónimo, pero si permite referenciar y usarlo en cualquier otra imagen en el futuro.

Aún si eliminamos los contenedores los datos se mantiene.

**Hotreload: Como nodemon, ng, etc que muestra los cambios al instante.**