**Curso de Docker**

Página para crear laboratorios: <https://labs.play-with-docker.com>

**Instalación en Windows**

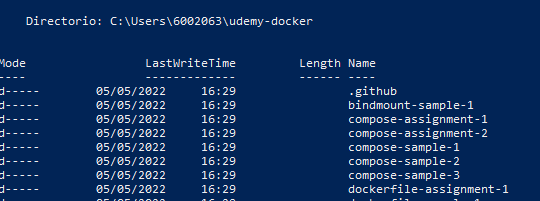
Lo primero que haremos será la instalación en Windows, link de descarga

<https://docs.docker.com/desktop/windows/install/>

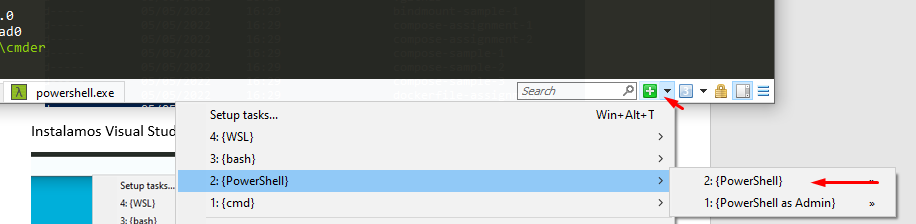
Clonaremos un repositorio de GIT que es donde están los recursos del curso

<https://github.com/BretFisher/udemy-docker-mastery>

en mi caso será está ruta



Instalamos Visual Studio Code y también Cmder, el cual configuramos lo siguiente



**TAB-COMPLETION (Power-Shell)**

Install-Module DockerCompletion -Scope CurrentUser

Este hay que ponerlo siempre que se inicie el terminal

Import-Module DockerCompletion

**Comandos generales**

Ayuda

docker container --help

Descargar imágenes

docker pull alpine

Muestra los contenedores

docker container ls

docker ps

Ver las imagenes descargadas

docker image ls

Lista todos los contenedores (Arrancados y parados)  
docker container ls -a

Muestra información sobre la configuración el contenedor (Configuración, red, volúmenes, arranque)

docker container inspect mysql

Muestra las estadísticas de los contenedores (Uso de CPU, memoria ...)

docker container stats

**Arrancar y levantar un contendor**

El puerto de la izquierda es por el cual nosotros accedemos al contenedor, debe ser distinto en los contenedores o creará conflicto

Levanta un contenedor con nginx y usa el puerto 80

docker container run --publish 80:80 nginx

Lo mismo, pero en segundo plano

docker container run --publish 80:80 --detach nginx

Lo mismo, pero agregando un nombre (Webhost)

docker container run --publish 80:80 --detach --name webhost nginx

Levantar un contenedor en segundo plano (-d), con nombre y usando la imangen mongo

docker run --name mongo -d mongo

Arranca el contenedor indicado

docker start mongo

Levanta un contenedor, pero con -it abre una terminal que indiquemos (bash) y con exit salimos

docker container run -it --name proxy nginx bash

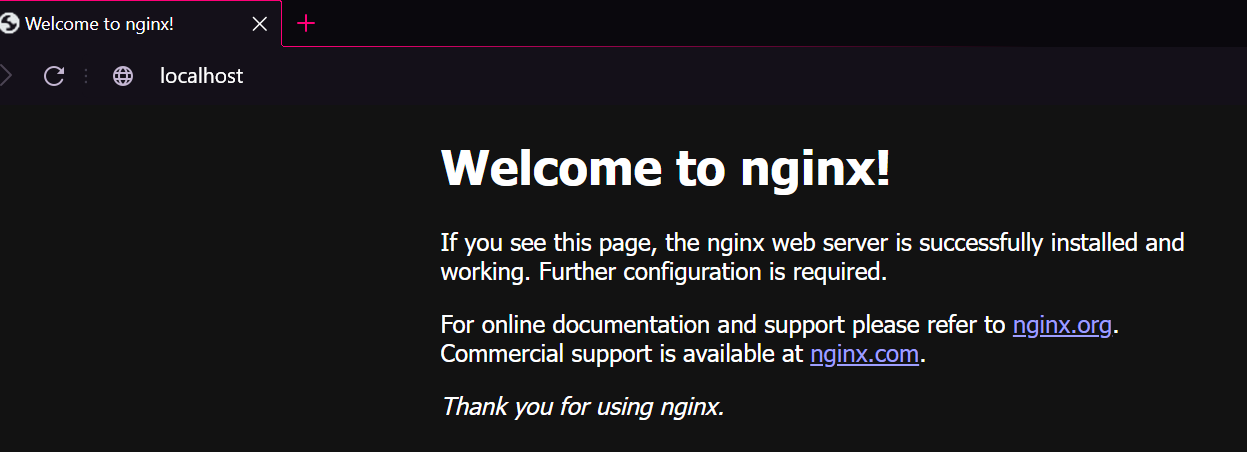
docker container run -it --name ubuntu ubuntu

Arranca el contenedor indicado y además entra en la terminal

docker container start -ai ubuntu

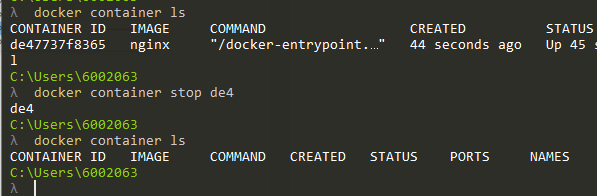
Acceder al contenedor indicado (mysql) usando la terminal bash

docker container exec -it mysql bash



**Parar o detener un contenedor**

Con poner los primeros dígitos del ID sirve



Para o detiene el contenedor indicado

docker container stop de4

Para eliminar un contenedor primero debe de estar parado, le pasamos el PID

docker container stop de4

docker container rm de4

Forzar la eliminación del contenedor (Elimina, aunque esté arrancado), le pasamos el PID del contenedor

docker container rm -f 5b8

**Ver logs**

Ver los logs del contenedor webhost

docker container logs webhost

**Ver procesos**

Ver los procesos de un contenedor

docker container top webhost

**BBDD**

Creamos un contenedor con una base de datos mysql, con nombre db, (-d segundo plano, -p puerto, -e las variables de entorno que queremos).

docker container run -d -p 3306:3306 --name db -e MYSQL\_RANDOM\_ROOT\_PASSWORD=yes mysql

La contraseña que se ha generado debemos mirarla en el log

docker container logs db



**Apache**

Levantar un servidor Web Apache

docker container run -d --name webserver -p 8080:80 httpd

**RED**

Muestra las redes que se han creado

docker network ls

Ver los contenedores conectados al bridge/host

docker network inspect bridge

docker network inspect host

Crear una red virtual

docker network create my\_app\_net

Arranca una máquina la opción --network es para unirlo a la red que elijamos

docker container run -d --name new\_ngix --network my\_app\_net nginx

Conectarte de una red y desconectarte (Se introduce el id de la red)

docker network connect 9d8ebc7d8350

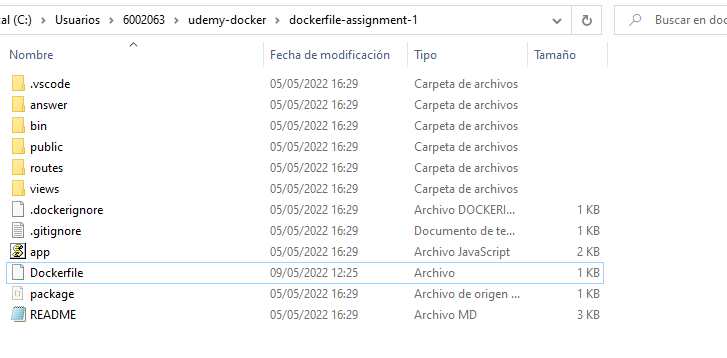
docker network disconnect 9d8ebc7d8350

Hacer ping (Deben estar en la misma red)

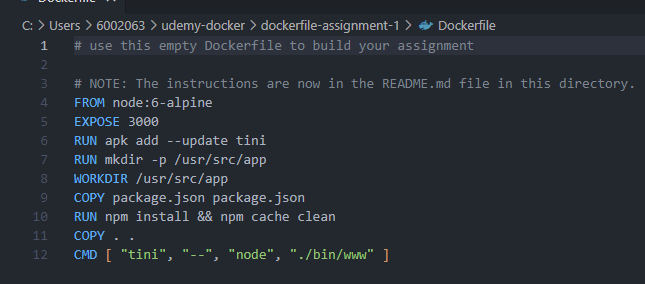
docker container exec -it ubuntu1 ping Ubuntu

**Crear un Docker file**

Los archivos nos lo proporcionan el curso



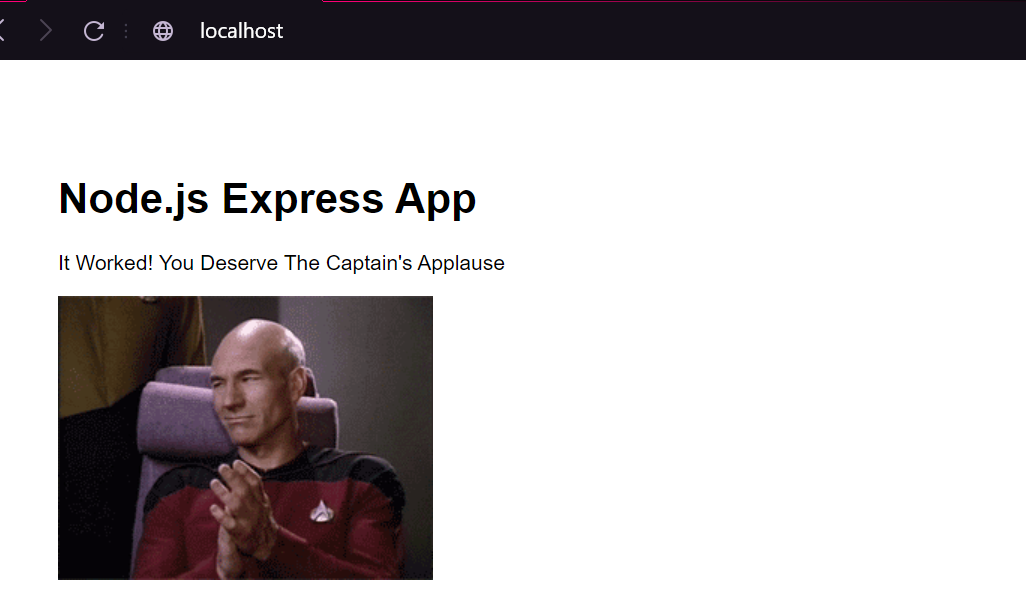
Editamos el DockerFile con lo siguiente



docker build -t testnode .

docker container run --rm -p 80:3000 testnode

Y listo



**Volúmenes**

Ver los volúmenes que tenemos de todos los contenedores

docker volume ls

Arrancar un contenedor con una BBDD, le indicamos el nombre del volumen (mysql-db) y la ruta donde se va a montar (/var/lib/mysql)

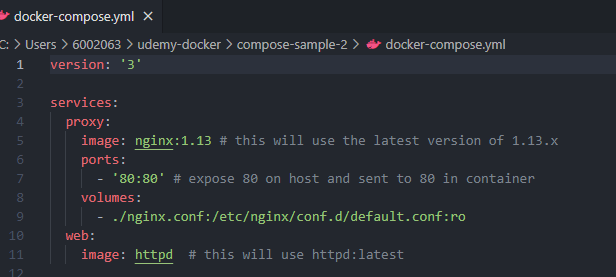
docker container run -d --name mysql3 -e MYSQL\_ALLOW\_EMPTY\_PASSWORD=True -v mysql-db:/var/lib/mysql mysql

Arrancamos un nginx pero le indicamos que coja los archivos de nuestra ruta (nos lo proporciona el curso) y después los va a sustituir de este modo no saldrá la página por defecto de nginx si no muestro archivo index.html

docker container run -d --name nginx2 -p 80:80 -v ${pwd}:/usr/share/nginx/html nginx

**Docker Compose**

Lo primero es tener un Docker Compose



Levantar un contenedor con Docker Compose (desde la misma carpeta que tenemos el Docker Compose)

docker-compose up

Crear un Docker Compose, dependerá de la cantidad de aplicaciones que queramos incluir, este es un ejemplo básico (Importante la extensión YAML)



Todo es correcto



**Swarm**

El Swarm es crear un nodo donde dentro hay contenedores, para que actúen como clúster

Iniciar Swarm y crear el primer nodo

docker swarm init

Ver los nodos

docker node ls

**Servicios**

Se tiene que haber creado antes el Swarm

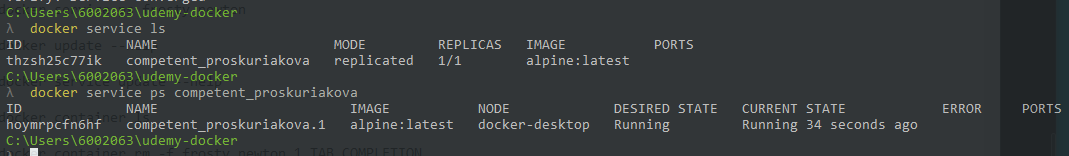
Crear un servicio

docker service create alpine ping 8.8.8.8

Listar los servicios

docker service ls

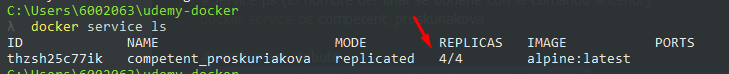
docker service ps competent\_proskuriakova



Cambiar un atributo

docker service update competent\_proskuriakova --replicas 4

Podemos ver que se ha cambiado



Eliminar el contenedor (Vemos que borra una réplica y la vuelve a crear inmediatamente)

docker container rm -f competent\_proskuriakova.4.q1u89cg53m42c4l6pah7b8q21

Eliminar el servicio

docker service rm competent\_proskuriakova

Crear un driver de red

docker network create --driver overlay mydrupal

Añadir dos servicios al driver creado

docker service create --name psql --network mydrupal -e POSTGRES\_PASSWORD=mypass postgres

docker service create --name drupal --network mydrupal -p 80:80 drupal

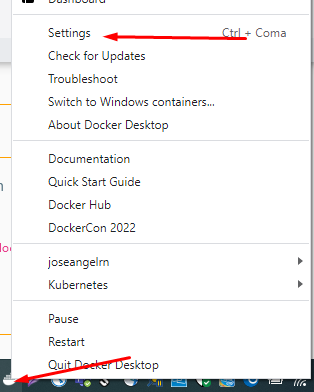
**Deploy**

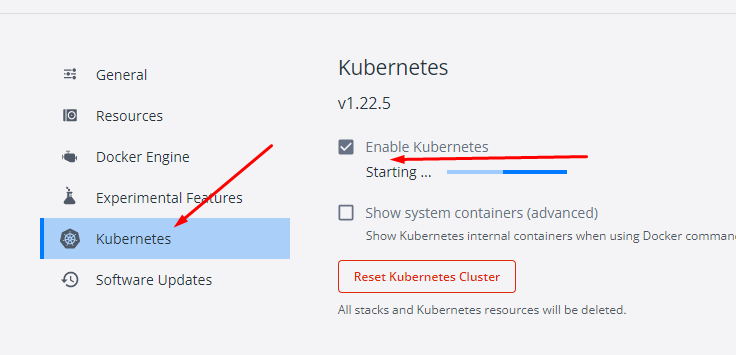
Hacer un deploy de una app

docker stack deploy -c example-voting-app-stack.yml voteapp

**Instalar kubernetes en DockerHub**

Seguir los siguientes pasos





kubectl get nodes

kubectl config get-contexts

kubectl config use-context docker-desktop

**Comandos básicos de kubernetes**

Arrancar un pod

kubectl run my-nginx --image nginx

Ver los pods

kubectl get pods

Ver todos los pods, servicios, namespace, etc.

kubectl get all

Eliminar un pod

kubectl delete pod my-nginx

Crear un deploy

kubectl create deployment httpenv --image=bretfisher/httpenv

Escalar un deploy

kubectl scale deployment/httpenv --replicas=5

Crear un deploy con -o yaml vemos el ficho que crea

kubectl create deployment test --image nginx --dry-run -o yaml

Exponer un deploy a un puerto

kubectl expose deployment/httpenv --port 8888

Ver los servicios (Muestra puertos, ip, etc)

kubectl get service

Acceder al bash de un pod (Se crea temporalmente)

kubectl run --rm -it --image bretfisher/netshoot -- bash

Acceder al bash de un pod, pero el pod le ponemos un nombre (Se crea temporalmente)

kubectl run tmp-shell --rm -it --image bretfisher/netshoot -- bash

Crear un NodePort Service (Acceder al host por la ip)

kubectl expose deployment/httpenv --port 8888 --name httpenv-np --type NodePort

Crear un LoadBalancer Service (Acceder al host por el puerto)

kubectl expose deployment/httpenv --port 8888 --name httpenv-lb --type LoadBalancer

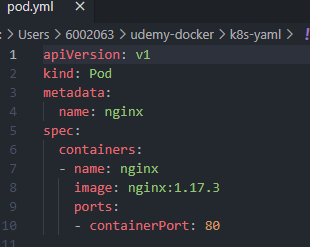
Borrar un NodePort

kubectl delete service/httpenv service/httpenv-np

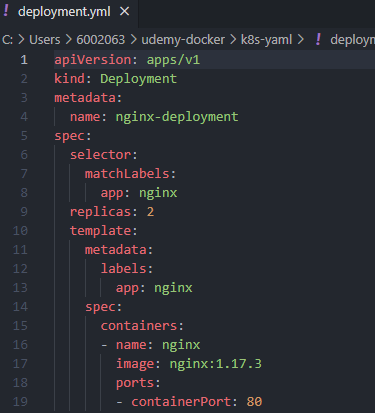
Borrar un LoadBalancer

kubectl delete deployment/httpenv service/httpenv-lb

Ejemplo de crear un pod



Ejemplo de crear un Deployment



Tener varios



Crear un pod, servicio, app, etc. A través de un fichero (-f)

kubectl apply -f app.yml --dry-run

Ver namespaces

kubectl get namespaces

kubectl get all --all-namespaces

Configuración namespaces

~/.kube/config file