Iniciar docker

sudo systemctl start docker

docker --version

Descargar imagen nginx

docker pull nginx

Levantar contenedor

docker run nginx

Cada vez q se ejecuta se levanta una imagen diferente

Levantar imagen sin bloquear el terminal

docker run -d nginx

Listar imagnes activas y detenidas

docker ps -a

Listar imágenes corriendo

Docker start + ID

Eliminar contenedor

docker rm + id

Eliminar imagen

docker rmi +id

En este caso se bajara y se ejecutara el container a la ve

docker run -d --name ws-nginx -p 8080:80 nginx

Con -d no va a depender de la consola, es decir la libera

Se va a dar como nombre al container con el –-name a ws-nginx

Con -p se va a port guardiar el puerto 8080 del servidor al puerto 80 del container

Si se levanta el localhost:8080 se iniciara el nginx

Ver los logs

docker logs ws-nginx

Para poder instalar docker en RH 8.5 se tuvo que consultar los siguientes links

<https://docs.docker.com/engine/install/centos/>

<https://platzi.com/discusiones/1432-docker/46840-hola-guido-al-correr-el-comando-me-sale-esto/>

Para logearse a docker hub:

docker login -u erdl1970

Pediara el password

Luego saldra este aviso:

WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.

Eso quiere decir que en el archivo config.json se guardaran las credenciales de acceso a docker hub

Craendo nuestro primera imagen

Crear el archivo Dockerfile y dentro editarlo con:

FROM hello-world:linux

Guardar

Luego ejecutar para construer la imagen: docker build -t local-hello-world:linux .

Se creo:

[ec2-user@ip-172-31-9-84 ~]$ docker build -t local-hello-world:linux .

[+] Building 1.3s (6/6) FINISHED

=> [internal] load .dockerignore 0.0s

=> => transferring context: 2B 0.0s

=> [internal] load build definition from Dockerfile 0.0s

=> => transferring dockerfile: 119B 0.0s

=> [internal] load metadata for docker.io/library/hello-world:linux 1.0s

=> [auth] library/hello-world:pull token for registry-1.docker.io 0.0s

=> [1/1] FROM docker.io/library/hello-world:linux@sha256:726023f73a8fc51 0.2s

=> => resolve docker.io/library/hello-world:linux@sha256:726023f73a8fc51 0.0s

=> => sha256:726023f73a8fc5103fa6776d48090539042cb822531 2.07kB / 2.07kB 0.0s

=> => sha256:7e9b6e7ba2842c91cf49f3e214d04a7a496f8214356f41d 525B / 525B 0.0s

=> => sha256:9c7a54a9a43cca047013b82af109fe963fde787f63f 1.47kB / 1.47kB 0.0s

=> => sha256:719385e32844401d57ecfd3eacab360bf551a1491c0 2.46kB / 2.46kB 0.1s

=> => extracting sha256:719385e32844401d57ecfd3eacab360bf551a1491c05b858 0.0s

=> exporting to image 0.0s

=> => exporting layers 0.0s

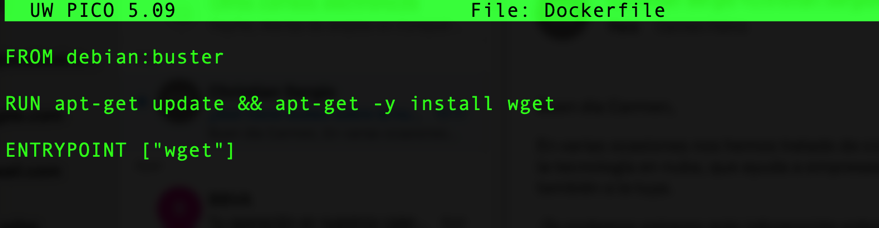
=> => writing image sha256:1ae33c64cbfffc2a22f28b78ae9659c4ac26df8c4686d 0.0s

=> => naming to docker.io/library/local-hello-world:linux 0.0s

Si vemos con docker images



Creando otra imagen, con su respetivo Dockerfile



docker build -t wget-downloader:local .

Ya temenos nuestras dos imagenes creadas

eduardoramirez@MacBook-Air-de-Eduardo 02 % docker images

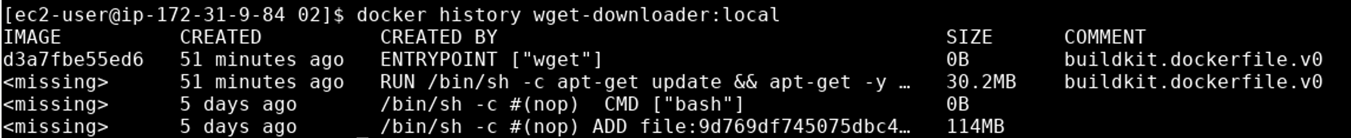
REPOSITORY TAG IMAGE ID CREATED SIZE

wget-downloader local b51b9af944c0 19 seconds ago 145MB

local-hello-world linux d2c94e258dcb 9 months ago 13.3kB

Para ver las capas que se crearon sobre la imagen

docker history wget-downloader:local



Para probar el funcionamiento

docker run wget-downloader:local

<https://www.google.com/images/branding/googlelogo/1x/googlelogo_color_272x92dp.png>

Desacargara una imagen, pero si en contenedor no es levantado se perdera

docker run wget-downloader:local <https://www.google.com/images/branding/googlelogo/1x/googlelogo_color_272x92dp.png>

Enlazar un volume si es en linux:

docker run -v ${PWD}/image\_download: /image\_download wget-downloader:local -0 /image\_download/download.png <https://www.google.com/images/branding/googlelogo/1x/googlelogo_color_272x92dp.png>

Ese comado lo modifique a:

docker run -v /Users/eduardoramirez/tmpd:/image\_download wget-downloader:local -O /image\_download/download.png <https://www.google.com/images/branding/googlelogo/1x/googlelogo_color_272x92dp.png>

Creo se descargo en la carpeta tmpd que está dentro de la carpeta eduardoramirez

**Crear imagen con app node y correr el contendor**

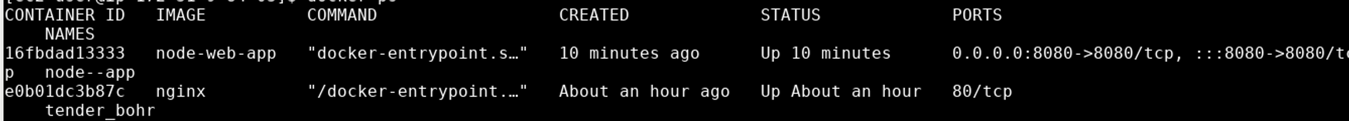
Utilizando la Carpeta “Dockerfile\_ejemplos” de la clase en la carpeta 03

Ejecutar docker build -t node-web-app .

Luego: docker run --name node--app -p 8080:8080 -d node-web-app

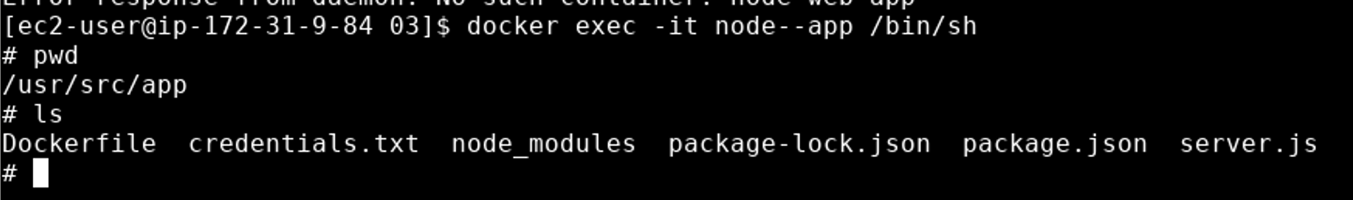
Luego probar en el navegador: localhost:8080

Ingresar al contenedor node--app



Con el siguiente comando: docker exec -it node--app /bin/sh

Podemos ubicarnos en la carpeta construida



Igual podemos listar directo los archivos de la carpeta mediante

docker exec -it node--app ls

Para detener un container

docker stop + nombre del container

Para eliminar en container

docker rm + nombre de container

Para evitar compiar archivos utilizar el .dockerignore

Creando una nueva imagen del ejemplo 08

docker build -t python-course .

eduardoramirez@MacBook-Air-de-Eduardo 08 % docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

python-course latest ed480c05bb1c About a minute ago 160MB

node-web2-app latest c2952085e911 2 hours ago 915MB

wget-downloader local b51b9af944c0 10 hours ago 145MB

local-hello-world linux d2c94e258dcb 9 months ago 13.3kB

Luego ejecutar:

docker run -d --rm --name python \--hostname python \ -v ${PWD}:/usr/src/app python-course

Corregido para mac:

docker run -d --rm --name python --hostname python -v "$(pwd)":/usr/src/app python-course

OJO el rm es porque el profe no quiere que el container se quede en su equipo

Este seria el ideal: docker run -d --name python --hostname python -v "$(pwd)":/usr/src/app python-course

Esto crea un container

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

c38ebe18029b python-course "sleep 86400" About a minute ago Up About a minute python

18fac829fbdd node-web2-app "docker-entrypoint.s…" 2 hours ago Up 2 hours 0.0.0.0:8080->8080/tcp node-app

Luego para inspeccionar el container ejecultamos:

docker exec -it python /bin/bash

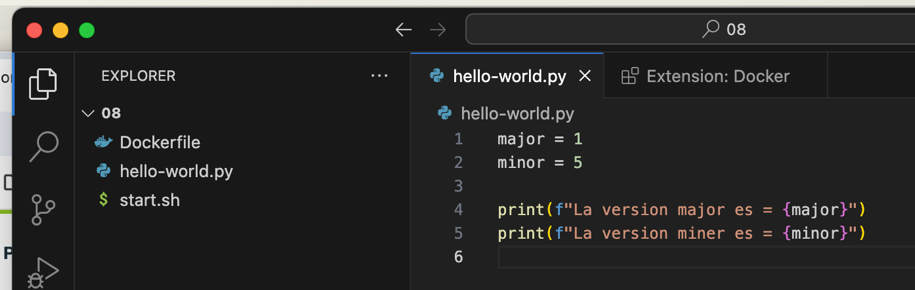
Mostrara:

root@python:/usr/src/app# ls

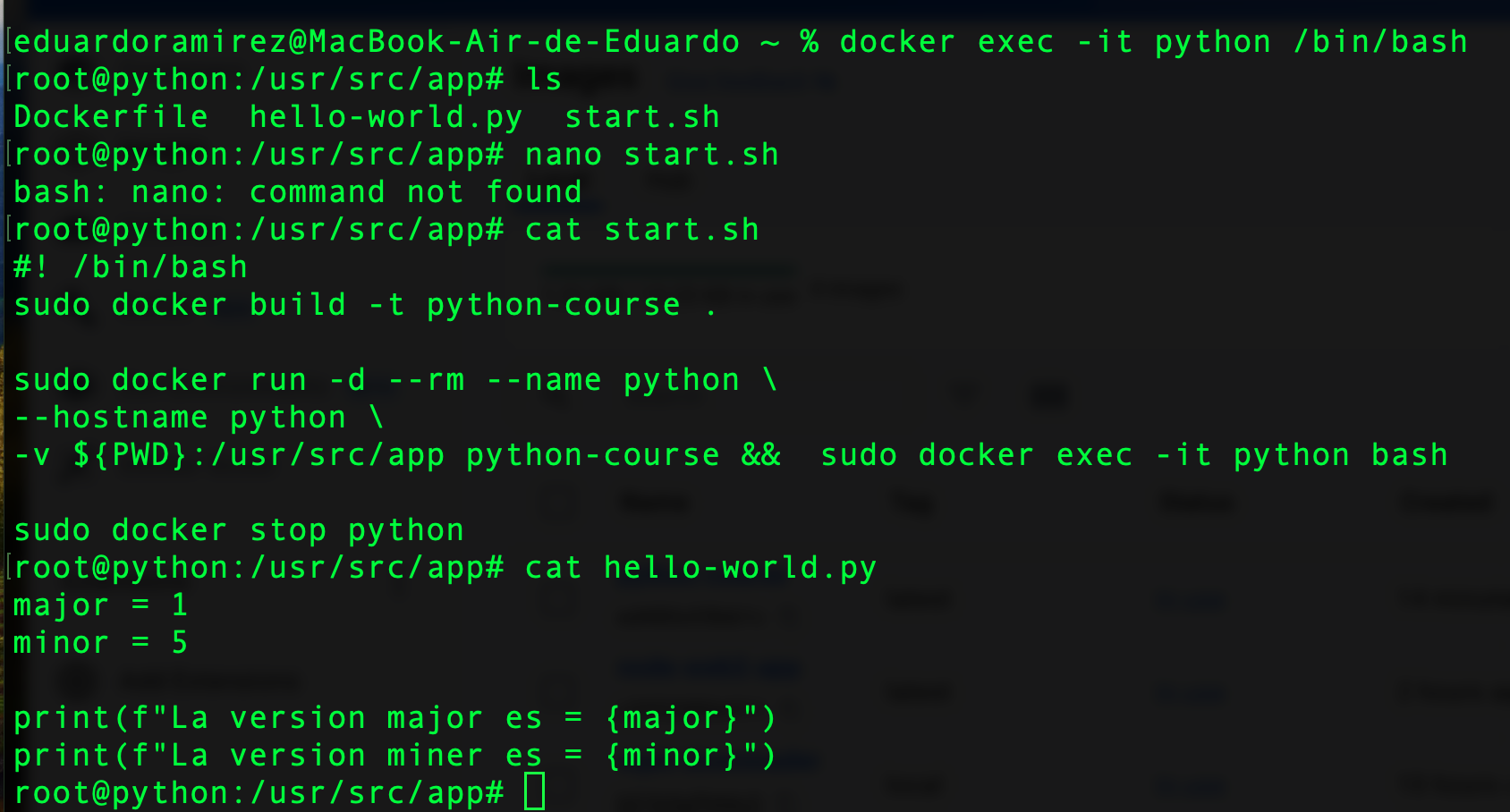
Dockerfile hello-world.py start.sh

Ahora todo lo que se modifique en el ejemplo 08 se modificara en el contenedor, como por ejemplo:

En visual code cambie el valor de minor a 5



Si entro al contenedor igual se modificara



Esto quiere decir que se puede cambiar todo lo de un contenedor incluso las versiones de los lenguajes u otros

Para buscar imágenes: docker images | grep python(este ultimo es el texto a buscar)

Tambien se puede llegar al contenedor mediante:

./start.sh

Para publicar imagen en el dockerhub

Crearla

docker build -t node-web-app:1.0.0 .

La nombras con el usuario de docker hub

docker tag node-web-app:1.0.0 erdl1970/node-web-app:1.0.0

Y se sube: docker push erdl1970/node-web-app:1.0.0