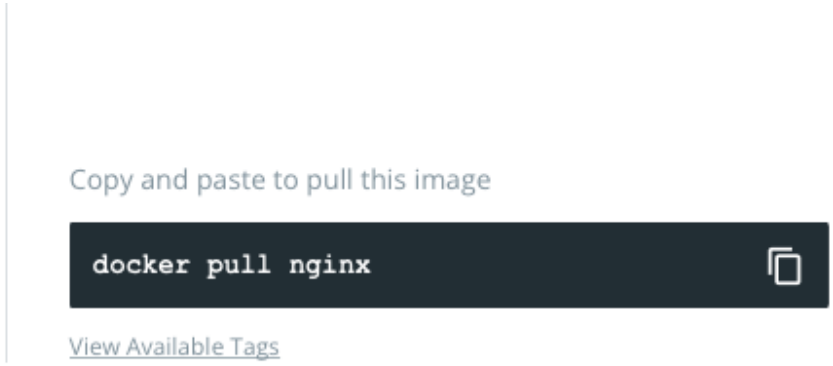
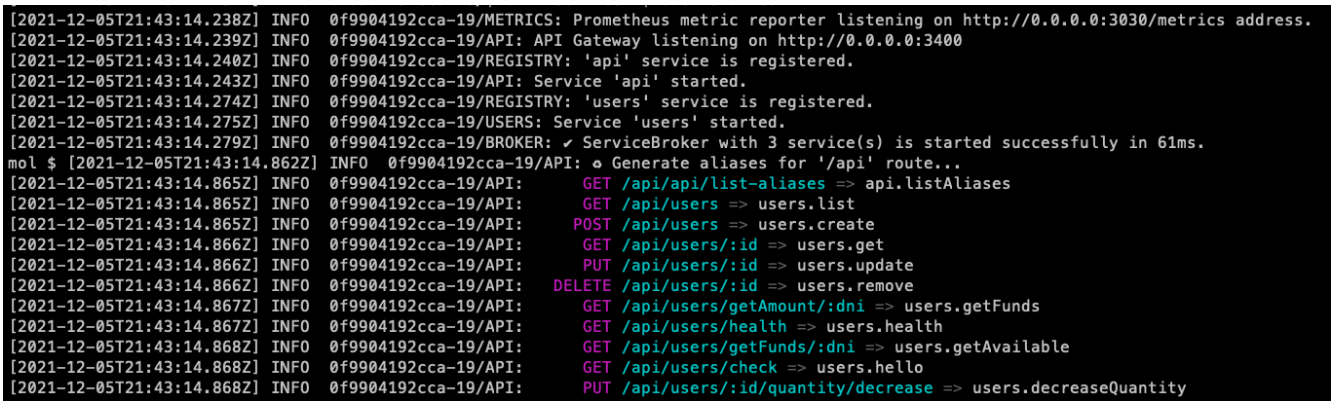
Practica Parte I.

1. Entre a Dockerhub ,busque la imagen de nginx y ejecútela sobre su máquina.
    a. Entrar a dockerhub https://hub.docker.com/ , buscar nginx. Le recomendamos que siempre trate de buscar las imágenes oficiales.
    b. Copiar el docker pull command (esto lo encontrara en el costado derecho) y pegarlo en la consola de la máquina. Darle enter, esto iniciara una tarea de pulling: **docker pull nginx**



    c. Chequear que la imagen se encuentre en nuestro entorno local: **docker image ls**
    d. Lanzar un contenedor a partir de la imagen descargada en forma desatachada: **docker run -d "image-id"**
2. Entramos sobre un contenedor en ejecución.
    e. Listamos todos los contenedores que están ejecución: **docker ps**
    f. Recuperamos el image id (solo los 4 primeros caracteres) y entramos al contenedor: **docker exec -it "container-id" bash**
    g. Listamos las carpetas dentro del contenedor (la que usted desee): **ls /opt/**
    h. Stopeamos el contenedor: **exit**
3. Mapeamos a un puerto**:**
    i. A continuación vamos a descargar una imagen personalizada que realiza llamadas a apis (la construcción de este programa está fuera del alcance de este módulo, debido a que es código backend): **docker pull matiasnmont/curso-docker:v1**
    j. Chequeamos si la imagen se descargó correctamente: **docker image ls**
    k. Lanzamos el contenedor basado en la imagen descargada anteriormente de manera desatachada: **docker run -d "image-id"**
    l. Chequeamos si el contenedor está corriendo correctamente: **docker ps**

m. Una vez que la imagen este corriendo, mostramos lo que loguea el contenedor por pantalla, debería ver una salida similar a esta : **docker logs "container-id"**



2. Este contenedor mapea algunas apis, para esta práctica solo se habilito la api para el chequeo de salud de la aplicación : **GET /api/users/health**
3. Stopeamos el contenedor momentáneamente: **docker stop "id-container"**
4. La forma en la que se levantó el contenedor anteriormente no permite que nosotros le hagamos hit a las apis debido a que no publicamos los puertos al exterior, para ello vamos a correr un nuevo contenedor de manera desatachada y publicando el puerto 3400: **docker run -d -p 3400:3400 "image-id"**
5. Chequeamos que el contenedor este corriendo correctamente , junto con el puerto publicado: **docker ps**
6. Chequeamos que la aplicación funciones correctamente: **curl localhost:3400/api/users/health**
7. Stopeamos el contenedor momentáneamente: **docker stop "id-container"**
4. Manejamos Networks
    a. Creamos una nueva network (red virtual dentro de docker): **docker network create MUNDOSE**
    b. Levantamos el contenedor a partir de la imagen custom del punto anterior, publicando el puerto y adjuntado a la red creada en el punto anterior: **docker run -d -p 3400:3400 --network MUNDOSE "image-id"**
    c. Levantamos una segunda imagen instalando curl: **docker run -it ubuntu bash**
    d. Una vez dentro del contenedor corremos el comando: **apt-get update && apt-get install -y curl**
    e. Hacemos un curl al contenedor que contiene la api con el siguiente comando: **curl MUNDOSE:3400/api/users/health** (Nos debería dar error)
    f. Salimos del contenedor: **exit**

g. Lanzamos el contenedor ubuntu cliente en la red MUNDOS-E y luego instalamos nuevamente curl: **docker run -it --network MUNDOSE ubuntu bash** y adentro corremos **apt-get update && apt-get install -y curl**
h. Abrimos otra terminal e inspeccionamos las ip que tomo el contenedor de la imagen custom que contiene la api con el siguiente comando: **docker inspect -f '{{range.NetworkSettings.Networks}}{{.IPAddress}}{{end}}' "container-id"**
i. Volvemos a la ventana donde tenemos el otro container corriendo (ubuntu cliente) y hacemos un curl para q nos responda la imagen custom que contiene la api: **curl <ip-punto-anterior>:3400/api/users/health**

Practica Parte II.

5. Vamos al siguiente link y descargamos el repositorio que contiene la imagen custom del punto anterior: **git clone** https://github.com/MatiasNMont/getOf.git
6. Abrimos el proyecto, en la raíz se encuentra un archivo llamado Dockerfile , necesitamos abrirlo para modificarlo. Discuta con sus compañeros las malas prácticas que tiene este archivo: **cd getOf**
7. Si hizo una cuenta en dockerhub es hora de logearse para subir esta imagen a su cuenta, deberá introducir usuario y contraseña: **docker login**
8. Antes de subir la imagen a la docker registry de su cuenta necesitamos buildear la imagen y probarla localmente. Eso lo realizara en los próximos puntos.
9. Para correr el siguiente comando debe posicionarse en el mismo directorio donde se encuentra el archivo dockerfile, una vez que se asegure de esto , realice el siguiente comando: **docker build -t <tag> .**
10. Para verificar que la imagen esta correctamente construida correa el siguiente comando, deberá ver la imagen con el tag definido anteriormente: **docker image ls**
11. Inicie el container a partir de la imagen buildeada anteriormente en modo desatachado, y publicando el puerto 3000: **docker run -d -p 3000:3000 "id-image"**
12. Chequee que la aplicación se publico correctamente en el puerto: **curl localhost:3000/api/users/health**
13. Etiquetamos la imagen para subirla a dockerhub: **docker tag <tag-image> <username>/<tag-image>:<version>**
14. Publicamos la versión a dockerhub: **docker push <username>/<tag-image>:<version>**