DockerHub

просмотр запущенных контейнеров:

docker ps

просмотр имеющихся образов

docker images

создание образа

docker build -t <имя нового образа> .

docker build -t nodeapp:v2 .

запуск контейнера

docker run --name <имя контейнера> -p 8000:3000 <имя имеющегося образа>

где -p для переадресации портов: 8000 – это локальный порт (хост), 3000 – порт контейнера

Сбилдить образ из Docker-файла

Docker build -t <имя образа> <путь к Dockerfile>

Очистить весь кэш и удалить все неиспользуемые образы:

docker system prune -a --volumes

запуск нового контейнера из имеющегося образа с передачей файлов

docker run --name my-node-app-2 -p 8000:3000 -v "C:\Users\akozlov\Documents\Visual Studio 2022\Projects\nodejs\_docker\nodejs\_docker":/usr/src/app nodeapp:v2

изменения в сорсе немедленно передаются в работающий контейнер

доступ к файловому хранилищу из windows:

[\\wsl$\docker-desktop-data\data\docker\volumes](../../../../..//wsl$/docker-desktop-data/data/docker/volumes)

остановка запущенного контейнера

docker stop <имя контейнера>

удаление контейнера

docker rm <имя имеющегося контейнера>

переход в файловую систему контейнера

docker exec -it <имя работающего контейнера> /bin/sh

ls - просмотр каталога

cat <имя файла> - просмотр файла

для отслеживания изменений - пакет nodemon.

в виндошном докере работает так : "start:dev":"nodemon --legacy-watch index"

для отслеживания изменений в девелоперском сервере реакта - добавить при запуске контейнера параметр -e WATCHPACK\_POLLING=true

получиться:

docker run --name react-todo -it --rm -p 3000:3000 -v "C:\Users\akozlov\Documents\Visual Studio 2022\Projects\ToDoDocker\frontend\src":/app/src -e WATCHPACK\_POLLING=true frontend

просмотр сетей

docker network ls

создание сети

docker network create <имя сети>

билд и запуск пакета контейнеров

docker-compose up

остановка и удаление пакета контейнеров

docker-compose down

--------------------

для обновления какого-то компонента, например приложения в реакте, на DockerDesktop:

- правим реакт

- делаем обновленный образ -> docker build -t <наименование репозитория в docker hub>:<наименование версии>

например -> docker build -t kozlovas/store-app:v3

- обновляем образ в dockerhub -> docker push <имя образа>

например -> docker push kozlovas/store-app:v3

- обновляем файл разворачивания (deployment) -> указываем новый образ - image: kozlovas/store-app:v2

- обновляем кластер -> kubectl apply -f <имя папки с файлами yaml>

- все, должно все обновится

--------------------------

Для включения в существующий проект VS поддержки Docker

Правой кнопкой на проекте -> Add -> Docker Support.. -> Linux

Автоматически создастся DockerFile под нужную версию проекта, автоматичски подгрузятся пакеты и сбилдится.

Потом этот файл правим как нам нужно

Важно – если такого пункта в меню VS нет – добавляем файл ручками.

Это проекты на NodeJs. Просто копируем из какого-то другого проекта NodeJs файл Dockerfile в корень нужного проекта и правим.

Пример команды для создания образа для проекта на NodeJs:

docker build -f D:\Projects\Reactivities\client-app\Dockerfile -t kozlovas/reactivities-front "D:\Projects\Reactivities\client-app"

И пушим в DockerHub:

docker push kozlovas/reactivities-front