Docker Compose #11

So far we've seen how to make an Image, run the Image to spin-up a Container and also how to use Volumes to map our local project to a folder in app in the Container.

Now to use Volumes we are required to type out that pretty long command in the terminal and sometimes we might have multiple projects that we want Containers for and we want to be able to run all those Containers **at once**

For Example

We may have a database in addition to our api express app and also a React app. And we would want to run all the images of this 3 apps to spin-up a Container and have these applications working together

Brute Force Solution

We could carry on as we have been and use a docker build & a docker run to spin up Containers for each app but that would be very tedious and tasking and inefficient.

"Say Thanks to Docker Compose

What is Docker Compose?

It is a tool that comes with docker and - It is used for defining and running multi-container Docker applications

With Docker Compose, you can define the services that make up your application (Your Image) in a .yaml file, and then use docker compose to start and stop all running Containers

Create a Docker Compose file with a Service setup for our api

Docker-Crash-Course

> api

docker-compose.yaml # Add in main root folder

docker-compose.yaml

```
version: "3.8"                    # Docker Compose Version

services:                         # Services Usually have multiple Services / break to next line

  api:                            # api application added. It has multiple values to tag

    build: ./api                  # relative path to Dockerfile to be used to build image

    container_name: api-c         # Self-explanatory

    ports:
      - '4000:4000'               # Port mapping

    volumes:
      - ./api:/app                # Relative path to host container folder : folder in container
      - /app/node_modules         # Exempt node_modules from volume
```

Note: I speak in plural because docker-Compose usally runs several services (creates Images and several Containers each for those Images)

to add to our proj

Now lets say we created a new React app. Lets call add that also into our docker-Compose file as a new servic

Use

Docker Compose to Create And run our service

Terminal

```
> docker system prune          # Delete all Images & Contain

> cd ..\

... docker-Crash-Course > docker-compose up
...                          # Creates Images and runs Contain
```

Result

The above command finds the docker-compose file and gets all the services and then with our specified configurat in each it uses that to build Images and then immediat runs all the Images to get containers.

In our case, since we only have 1 service then we w have 1 Image and a Container of that Image running

```
> docker images          # To see the Images if created
> docker ps              # To see the running Contain

> docker-compose down          # Stops and deletes Containers
> docker-compose down --rmi all # Stops & deletes Containers, and Ima
> docker-Composed down --rmi all -v # deletes Containers Images and Vol
```

Dockerizing a React App #12

We want to now learn how to dockerize a frontend application, namely react

```
O Docker-Crash-Course
  > O api
+ v O myblog                    # Create react app => npm create-react-app myblog   (Terminal)
    ...
+     .dockerignore             # ignore node_modules => node_modules   (Inside file)
    ...
+     Dockerfile                # Add Instructions for Creating an Image
    ...
```

O Dockerfile

```
1  FROM node:17-alpine          # React Apps require a node Parent Image
2
3  WORKDIR /app                 # Select (Images) working Directory
4
5  COPY package.json .          # Copy package.json into (Images) WD
6
7  RUN npm install              # Install all dep using package.json copied
8
9  COPY . .                     # Copy all sourcecode and all into Images WD
10
11 EXPOSE 3000                  # Expose Isolated port for accessing app by the Container
12 CMD ["npm", "start"]         # Specify Command to be ran upon creation
```

Add React app as a new Service in our Docker Compose file

docker-compose.yaml

```
version: "3.8"

services:
  api:
    ...
  myblog:
    build: ./myblog                    # where to find Dockerfile to build
    container_name: myblog-c
    ports:
      - '3000:3000'
    volumes:
      - ./myblog:/app
      - ./myblog:/node_modules
    stdin_open: true
    tty: true
```

NOTE:

Volumes work on Linux & mac for React Applications.

But don't work on Windows because how docker works with WSL is different.

We could use a flag to bypass this

-e CHOKIDAR_USEPOLLING=true

But it only works occassionally

NOTE:

Both of this properties basically start the Container in Interactive mode. Which is what we should to with React Apps in docker

≡ Implement Simple React App that works with our Express API

○ myblog > src > App.js

```
import { useEffect, useState } from 'react';

function App() {
  const [blogs, setBlogs] = useState([]);

  useEffect(() => {
    fetch('http://localhost:4000/')
      .then(res => res.json())
      .then(data => setBlogs(data));
  }, []);

  return (
    <div className="App">
      <header className="App-header">
        <h1> all blogs </h1>
        { blogs && blogs.map(blog => (
            <div key={blog.id}>{ blog.title }</div>
        ))}
      </header>
    </div>
  );
}
```

≡ Use docker-compose to run both Apps

② Terminal

```
> docker-compose up
...
```

☐ After running, test apps to see that they both work and the React App actually fetches data from our API

Visit Port to See that our React Apps Container is Running

CHROME

localhost:3000

all blogs

Book Review: The Bear & Nightingale ABC

GAME REVIEW: Pokemon Brilliant Diamond

Show Review: Alice in Borderland

SHARING IMAGES ON DOCKER HUB #13

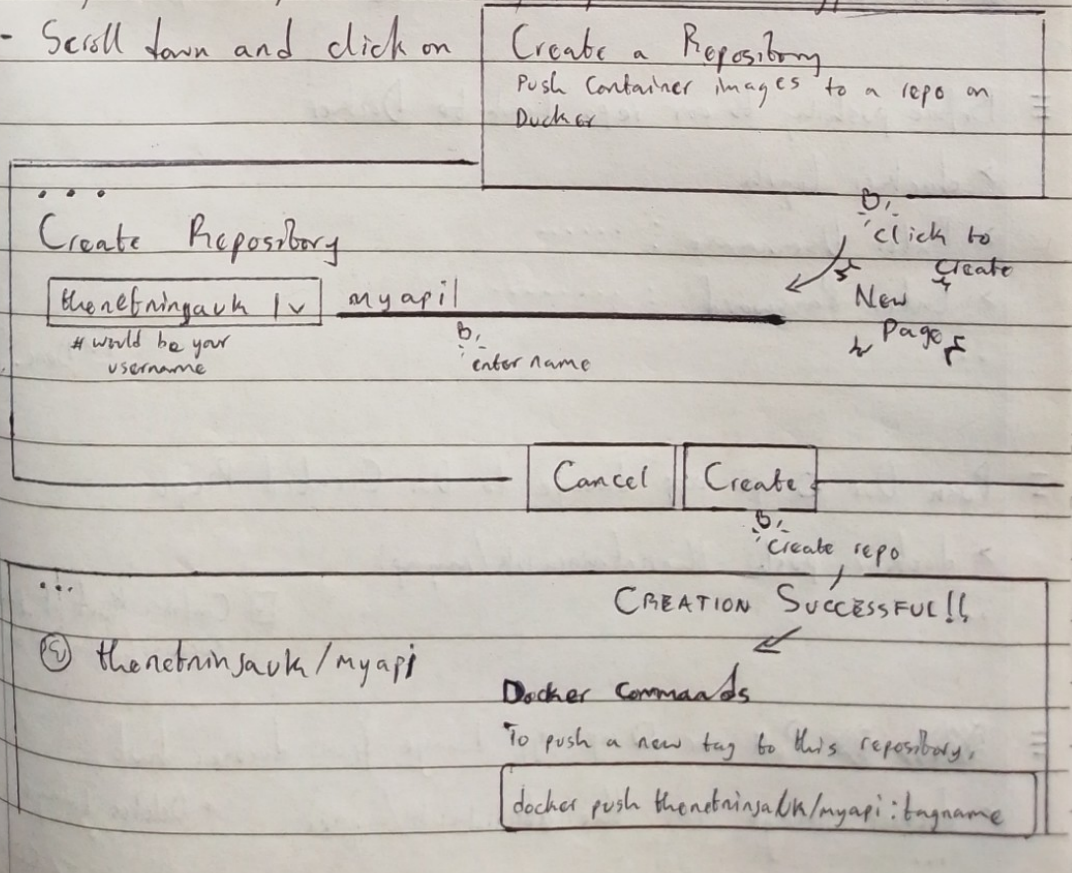
Just like GitHub where we share our code so other people can download and use them. On docker hub we can also share our images so other people can download and run them

"This is typically how we share Images with Others"

A WISE MAN ONCE SAID.

STEPS

- Visit hub.docker.com
- Register or Sign in (only if you are new or logged out)
- Scroll down and click on



(so we can push it to our repo)

- Now we want to build our ☐ api image only so therefore we won't be using docker compose for this

⊡ Terminal ≡ Build Only the ☐ api app Image

> cd api

api> docker build -t thenetninjauk/myapi .

...

☑ Image built

> docker images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
thenetninjauk/api	latest	de667de49760	3 hours ago	119MB

≡ Before pushing to our repo Login to Docker

> docker login

> Enter Username : .....

> Enter Password : .....

...

☑ Done. Now we can push

≡ Push Our ☐ api app Image to Our Created Repo

> docker push thenetninjauk/myapi

...

☑ Confirm that it pushed successfully on docker hub

≡ [redacted] Pull our ☐ api app Image from docker hub

> docker image rm thenetninjauk/myapi # Deletes Image

...

> docker pull thenetninjauk/myapi

☑ DONE

☑ COURSE COMPLETE