Dockerizing React Applications for Continuous Integration .

Day 79/100 .

Why You Should Consider Using Docker

- Docker helps to make shipping environments more predictable.
- Solves the “works on my machine” problem.
- You can share and reuse Docker containers between projects and teams.

Docker Installation Confirmation

To confirm if you have installed Docker on your machine, open the terminal and type the following. You should see a docker version in the console. However, if you don’t see a version, please reinstall or restart your machine.

```
→  ~ docker -v
Docker version 19.03.8, build afacb8b
→  ~
```

Creating A React Project :

Remember: Create React App doesn't handle backend logic or databases; it just creates a frontend build pipeline, so you can use it with any backend you want. Under the hood, it uses Babel and webpack.

```
➜  Work npx create-react-app my-react-docker-app


Creating a new React app in /Users/indreklasn/Work/
my-react-docker-app.

Installing packages. This might take a couple of mi
nutes.
Installing react, react-dom, and react-scripts...

yarn add v1.22.4
[1/4] 🔍  Resolving packages...
warning react-scripts > webpack-dev-server > chokid
ar@2.1.8: Chokidar 2 will break on node v14+. Upgra
de to chokidar 3 with 15x less dependencies.
[2/4] 🚚  Fetching packages...
[3/4] 🔗  Linking dependencies...
warning "react-scripts > @typescript-eslint/eslint-
plugin > tsutils@3.17.1" has unmet peer dependency
"typescript@>=2.8.0 || >= 3.2.0-dev || >= 3.3.0-dev
 || >= 3.4.0-dev || >= 3.5.0-dev || >= 3.6.0-dev ||
 >= 3.6.0-beta || >= 3.7.0-dev || >= 3.7.0-beta".
[----------------------------------------] 0/1837
```

Dockerizing a React App

Next, open the React project with your favorite text editor or IDE. We're going to need to create the following three files for Docker, Dockerfile, Docker.dev and docker-compose.yml

Use the following command:

```
touch Dockerfile Dockerfile.dev docker-compose.yml
```

- Dockerfile: A Dockerfile is a set of instructions used to construct an image. Inside this file, we declare the software we want to use for our project. For instance, for a React project, we're going to need Node.js. Dockerfile is usually used for production purposes.

- Dockerfile.dev: The same concept as the above Dockerfile, the main difference between a Dockerfile and Dockerfile.dev is the former is used for the production environment, the latter is used for the local development environment.

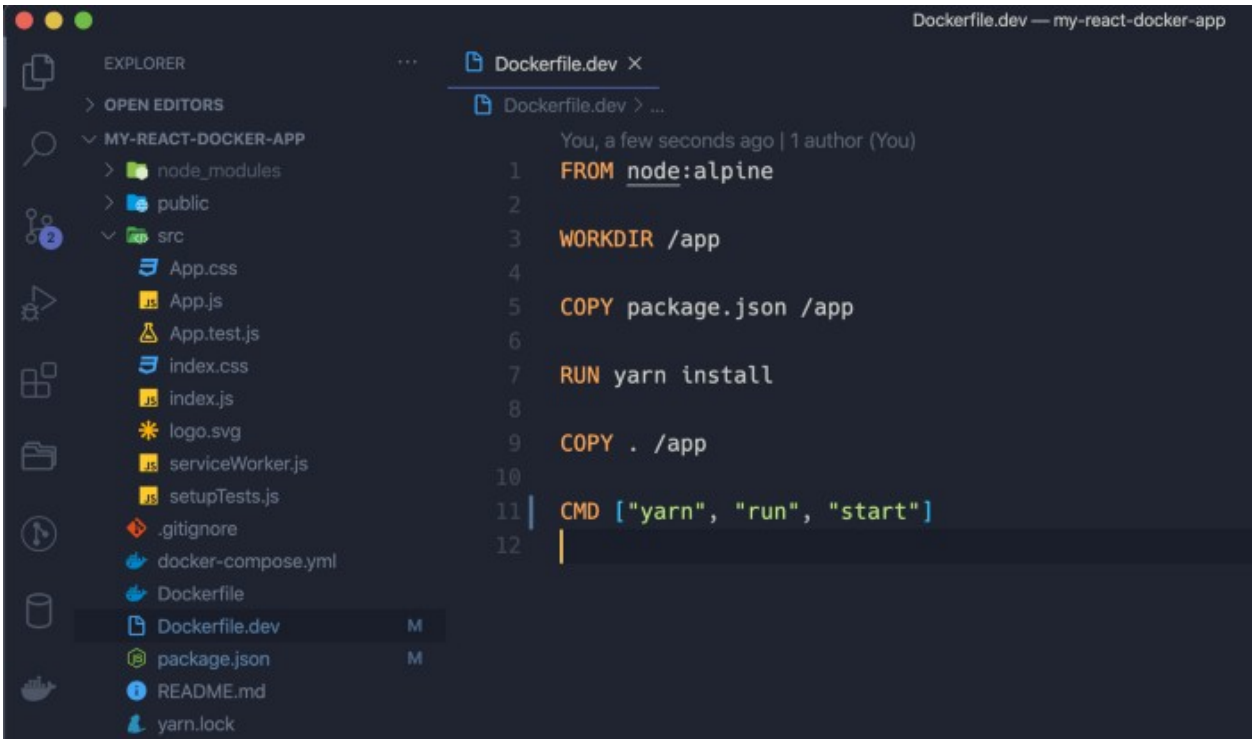
- docker-compose.yml: Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a YAML file to configure your application's services. Then, with a single command, you create and start all the services from your configuration. To learn more about all the features of Compose, see the list of features.

Note:

When we start a new Docker project, Docker doesn't know anything about React, Node, or the project at all. We need to explicitly tell Docker what software we need to run our project. For instance, we know that for a React project we definitely are going to make use of Node.js and NPM or Yarn.

Dockerfile.dev

Open the Dockerfile.dev and write the following:



Here's what's happening:

- We're telling Docker to grab a copy of Node and specify its Linux distribution as Alpine. Docker uses the Alpine Linux distribution by default. Why Alpine? Alpine Linux is much smaller than most distribution base images (~5MB), and thus leads to much slimmer images in general.

- Next, we set our working directory for the Docker container with the WORKDIR command. Once we have declared a working directory, any CMD, RUN, ADD, COPYcommand will be executed in the specified working directory.

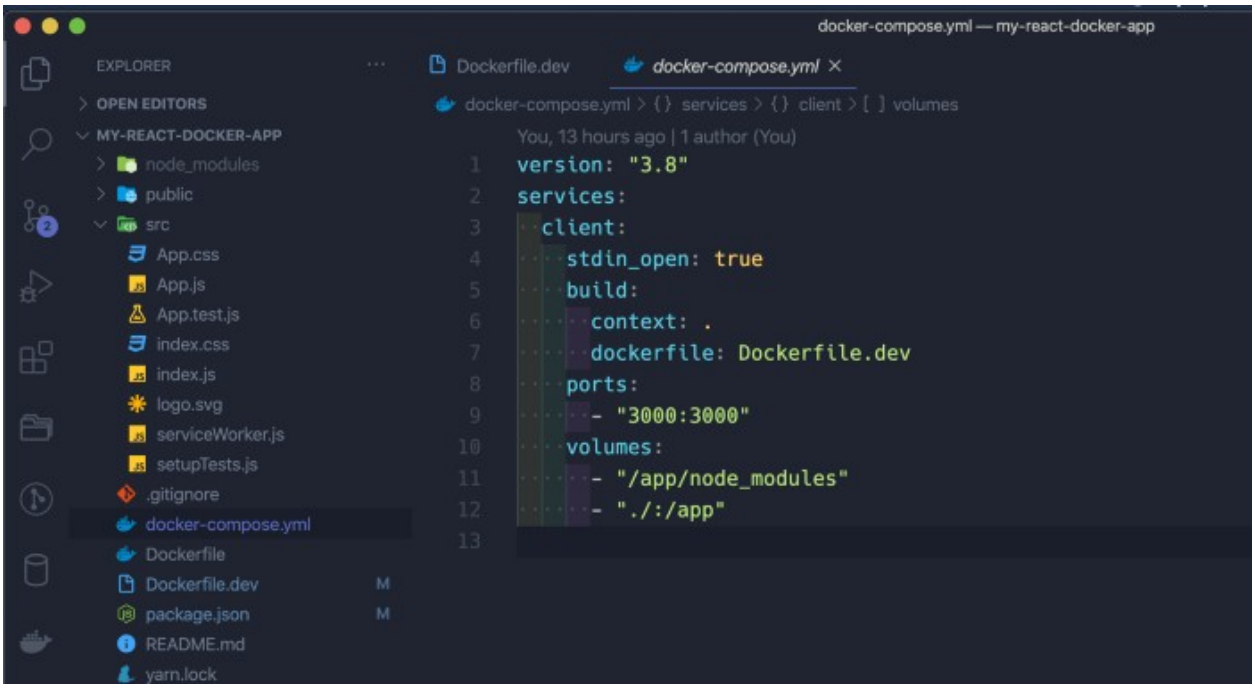
- Copy the package.json from our React project to the Docker container.

- Install the dependencies and copy the rest of our application to the Docker container.

- Lastly, run the development script. For CRA it's yarn run start to start the development server.

Setting Up the docker-compose.yml

Remember, Compose is a useful tool to chain all the docker commands and services together. Let's create a service called client and specify what it should do.



Here's what's happening inside the docker-compose.yml file

•We specify a version for Compose. Make sure Compose is compatible with Docker, here's a full list of all versions. we're going using version 3.8.

•Define the client service so we can run it in an isolated environment.

•The client service requires a docker file to be specified. For development, we're going to use the Dockerfile.dev file.

- Next, we map the port 3000 to Docker. The React application runs on port 3000, so we need to tell Docker which port to expose for our application.

Using Compose

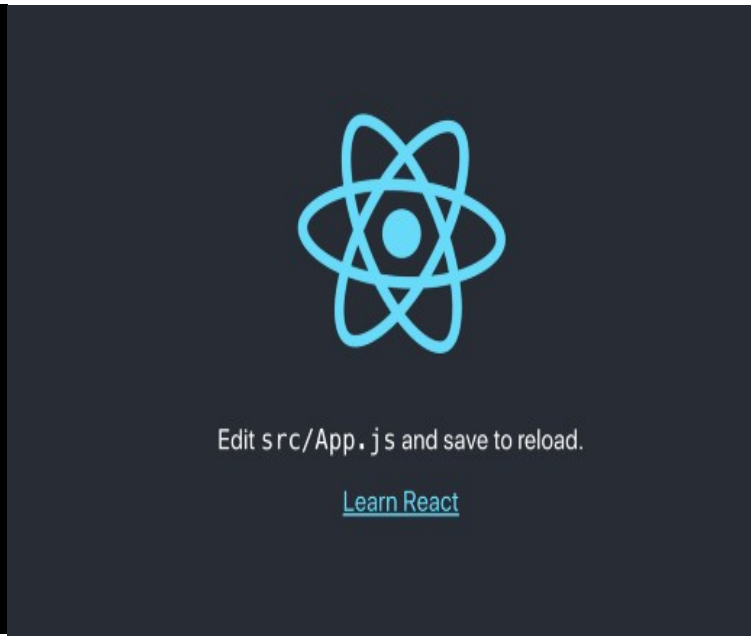
Once we're done writing the configuration, we can simply run the docker-compose up command and Docker builds up everything for us.

```
➜  my-react-docker-app git:(master) docker-compose up
Creating network "my-react-docker-app_default" with the default driver
Building client
Step 1/6 : FROM node:alpine
alpine: Pulling from library/node
cbdbe7a5bc2a: Extracting [==================================================>]  2.813MB/2
bdfc8e0b7a31: Downloading [==========================>                        ]  19.94MB/
35.91MBc8893: Download complete
057ed1351239: Download complete
280B
```

You should see all the commands which we declared in the Dockerfile.dev being executed. Once Compose is done doing its stuff, we can open the React application at http://localhost:3000/.

Output:

```
→  my-react-docker-app git:(master) ✗ docker-compose up
Starting my-react-docker-app_client_1 ... done
Attaching to my-react-docker-app_client_1
client_1  |
client_1  | > my-react-docker-app@0.1.0 start /app
client_1  | > react-scripts start
client_1  |
client_1  | ℹ ｢wds｣: Project is running at http://172.18.0.2/
client_1  | ℹ ｢wds｣: webpack output is served from
client_1  | ℹ ｢wds｣: Content not from webpack is served from /app/public
client_1  | ℹ ｢wds｣: 404s will fallback to /
client_1  | Starting the development server...
client_1  |
client_1  | Compiled successfully!
client_1  |
client_1  | You can now view my-react-docker-app in the browser.
client_1  |
client_1  |   Local:            http://localhost:3000
client_1  |   On Your Network:  http://172.18.0.2:3000
client_1  |
client_1  | Note that the development build is not optimized.
client_1  | To create a production build, use yarn build.
client_1  |
```



All the best,

Regards Lux Tech Academy.