# **Dockerize ReactJs App**

To create dockerize app for reactjs you first need to have desired node version on your local or prod machine.

[Install NodeJs using NPM](https://learn2torials.com/a/how-to-install-nvm" \o "how to install nodejs using nvm?" \t "https://learn2torials.com/a/_blank)

I assume that you already have docker installed on your local machine. Moving on let's now install react-create-app cli tool.

Open your terminal window and type following commands:

# install create-react-app

npm install -g create-react-app

# let's create a sampler react app

create-react-app reactjs-example

# let's head into our project dir

cd reactjs-example

## **Build local or dev dockerfile**

Let's first get react up and running on our local environment using docker. Open your terminal window and let's create a Dockerfile with following contents:

# get the base node image

FROM node:alpine as builder

# set the working dir for container

WORKDIR /frontend

# copy the json file first

COPY ./package.json /frontend

# install npm dependencies

RUN npm install

# copy other project files

COPY . .

# build the folder

CMD [ "npm", "run", "start" ]

Basically we are doing following with our dockerfile:

1. Downloading a base image from dockerhub
2. Defining a working directory for our container
3. Copying package.json file and place it in container working dir
4. Installing our npm dependencies
5. Copying rest of the project files
6. Finally, running npm start command to run dev env

Now, we have a dockerfile with all instructions that we need in order to create our docker image.

Let's now define and create our container using docker-compose-local.yml file.

Open your terminal window and create docker-compose-local.yml file with following contents:

version: '3'

services:

frontend:

build:

context: .

dockerfile: Dockerfile.dev

command: npm run start

container\_name: frontend

ports:

- "3000:3000"

volumes:

- ./:/frontend

- /frontend/node\_modules

Alright, our docker compose file defines how our container will look like and port we expose to our host machine.

Let's now create a container using this definition.

Open your terminal window and run following command to create our dev env:

# create and run our container

docker-compose -f docker-compose-local.yml up -d

Finally, above command will get our react app up and running. Now, you can access your running app on http://localhost:3000.

## **Build production dockerfile**

Now, that we have our local environment all set we need to get our production environement ready. Let's now create following two files:

* Dockerfile
* docker-compose.yml

Open Dockerfile and add following contents:

# get the base node image

FROM node:alpine as builder

# set the working dir for container

WORKDIR /frontend

# copy the json file first

COPY ./package.json /frontend

# install npm dependencies

RUN npm install

# copy other project files

COPY . .

# build the folder

RUN npm run build

# Handle Nginx

FROM nginx

COPY --from=builder /frontend/build /usr/share/nginx/html

COPY ./docker/nginx/default.conf /etc/nginx/conf.d/default.conf

Let's now create a new nginx config file in docker/nginx/default.conf with following contents:

server {

listen 80;

server\_name \_;

index index.html;

root /usr/share/nginx/html;

error\_log /var/log/nginx/error.log;

access\_log /var/log/nginx/access.log;

location / {

try\_files $uri /index.html =404;

}}

We defined out nginx production config file finally

we need to create docker-compose.yml  file with following contents:

version: '3'

services:

frontend:

build: .

container\_name: frontend

ports:

- "80:80"

volumes:

- ./:/frontend

- /frontend/node\_modules

we are now done here in order to check our production environment locally let's run following command:

# create and run a container

docker-compose up -d

Open your browser and visit http://localhost  to see the react project is up and running.