





DZone (/) > Web Dev Zone (/web-development-programming-tutorials-tools-news) > How to Dockerize a ReactJS App?

# How to Dockerize a ReactJS App?



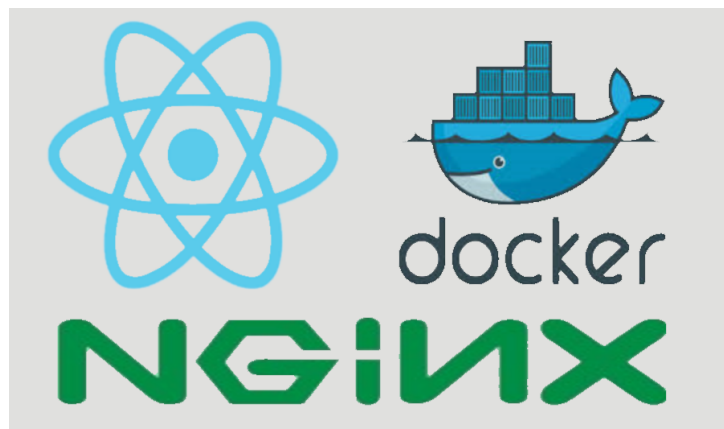
(/users/4009420/sanjaysaini2000.html) by [Sanjay Saini \(/users/4009420/sanjaysaini2000.html\)](#) · Sep. 08, 20 · Web Dev

Zone (/web-development-programming-tutorials-tools-news) · Tutorial

 Like (4)  Comment (3)  Save  Tweet

## Set Up a GraphQL Endpoint on a Database

StepZen lets you unlock the data in GraphQL, and database backends enabling any REST API or database gap between the database view of



In this article I will explain to you how to dockerize a ReactJS App from scratch. Some time back, I had created a Todo app using ReactJS and wanted to dockerize it.

So, one fine day, while doing so, I thought to note down the steps as well and share it with fellow developers. The result is this article.

If you have some basic idea about Docker or have just heard about it and want to learn how to dockerize ReactJS app, then this article is for you.

## What Is Docker?

In brief, Docker is an open source DevOps tool designed to help developers and operations to streamline application development and deployment.

By dockerizing an application, you can deploy and run an application using containers.

Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and deploy it as one package.

By doing so application can be deployed on any target environment/machine regardless of any customized settings that machine might have that could differ from the machine used for writing and testing the code.

To learn more about docker checkout this link (<https://www.docker.com/>).

REPCARDZ (Research) RESEARCH (research) WEBINARS (Webinars) ZONES

## Prerequisites

- First, we need to have docker installed on our machine so that we can build docker image and run docker containers. There are different installation for Mac and Windows. For Windows 10 Professional and Enterprise install docker for desktop from this link (<https://hub.docker.com/editions/community/docker-ce-desktop-windows>) but if you have Windows 10 Home edition like I have then you will have to installed Docker Toolbox from this link (<https://docs.docker.com/toolbox/overview/>).
- We also need to have account at Docker Hub registry so that we can pull and push Docker images. It's free so if you already haven't one, checkout this link (<https://hub.docker.com/>) to create one for yourself.
- Last, we need ReactJS application that we want to dockerize. So if you already have one that's great but if you don't then you can get the ReactJS app code that I am using in this article from my GitHub repository from this link (<https://github.com/sanjaysaini2000/react-todo-app>).

## Get Started...

In order to dockerize our ReactJS App we need to perform following steps.

- Launch the Docker machine.
- Create Dockerfile in our app folder.
- Create Docker image from the Dockerfile.
- And last, run the ReactJS Todo App in the container using Docker image.

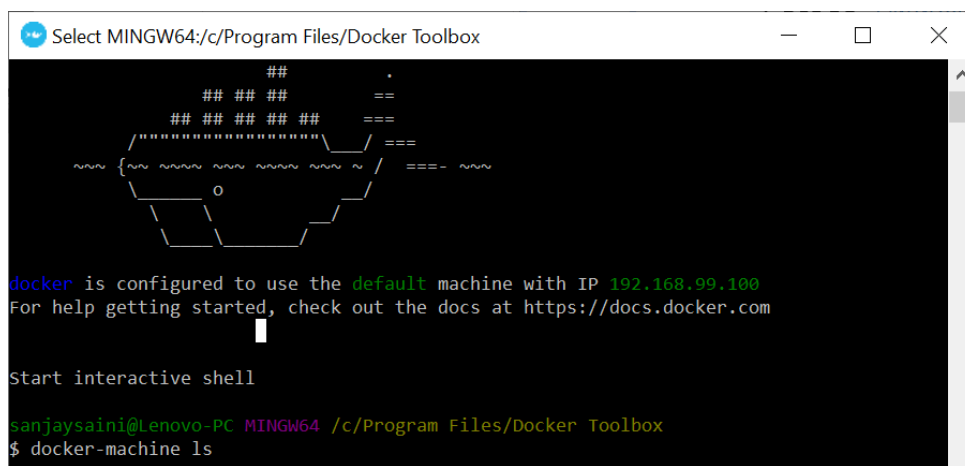
So let's get started...

## Launch Docker Machine

Docker machine is a small Linux VM that hosts the Docker Engine, which is a client-server application con of Docker Daemon and Docker CLI that interacts with Docker Daemon to create Docker images or running containers etc.

- In case, you have installed Docker Desktop for Window or Mac when the installation finishes, Docker Machine is launched automatically. The whale image in the notification area indicates that Docker is running, and accessible from a terminal.
- If you have installed Docker Toolbox then there are 2 ways to create docker machine locally.
  - By double clicking on the docker Quick Start Terminal icon on your desktop.
  - Using docker-machine CLI "create" command to create new Docker machine.

Since I have Docker Toolbox installed so I will choose the easy way and click on the Quick Start Terminal icon that will open the terminal and launch the Docker machine.



```

Select MINGW64:/c:/Program Files/Docker Toolbox

      ##
    ## ## ##
  ## ## ## ## ##
{ ~~~~~ }
  ~~~~~
    ~~~~~
  ~~~~~
    ~~~~~
  ~~~~~

docker is configured to use the default machine with IP 192.168.99.100
For help getting started, check out the docs at https://docs.docker.com

Start interactive shell

sanjaysaini@Lenovo-PC MINGW64 /c:/Program Files/Docker Toolbox
$ docker-machine ls
NAME          ACTIVE    DRIVER        STATE     URL                         SWARM   DOCKER
-----

```

You can run following docker-machine CLI command to check the Docker machine details and note the URL that we will use to open our ReactJS app in the browser.

```
$ docker-machine ls
```

You can do much more with docker-machine CLI commands like create, kill, start, stop Docker machine and much more but that is not in scope for this article however you can check-out **docker.com** for complete documentation on docker-machine CLI and also docker CLI as well.

Since now our Docker setup is up and running now we will focus on creating Dockerfile for our ReactJS app.

## Create a Dockerfile

Now, in the terminal, change directory to your ReactJS app folder and create a file name “Dockerfile” without any file extension using any dev editor like VS Code or just use Notepad.

Write following instructions in the Dockerfile and save it.

```
YAML
1 # Step 1
2
3 FROM node:10-alpine as build-step
4
5 RUN mkdir /app
6
7 WORKDIR /app
8
9 COPY package.json /app
10
11 RUN npm install
12
13 COPY . /app
14
15 RUN npm run build
16
17
18
19 # Stage 2
20
21 FROM nginx:1.17.1-alpine
22
23 COPY --from=build-step /app/build /usr/share/nginx/html
```

### Explanation

- In Stage 1, we are copying our app code in the “app” folder and installing app dependencies from package.json file and creating production build using Node image.
- In the Stage 2, we are using nginx server image to create nginx server and deploy our app on it by copying build items from \*/app/build\* folder to nginx server at \*/usr/share/nginx/html\* location.

## Create a .dockerignore file

Although it’s not necessary to have this file, it’s a good practice to have it since it can speed up image build process and also keep the image lean by excluding the unnecessary code from the Docker build context so that it doesn’t get into the image.

So just the way we created Dockerfile at the same location, we create a .dockerignore file and add following items that we don’t want to be copied into our docker image.

```
6
7 *.md
8
9 .gitignore
```

## Create a Docker Image

Now, run the Docker build command to build Docker image of our app using Dockerfile that we have just created.

Note that I have given *sanjaysaini2000/react-app* as name to my Docker image but you must replace it with the name you want to give to your app's Docker image.

Also note that image name must be followed by the dot which means that the path of the Docker build context and Dockerfile is the current folder.

```
$ docker build -t sanjaysaini2000/react-app .
```

```

MINGW64:/c/Sandbox/reactJS/todo-app
sanjaysaini@Lenovo-PC MINGW64 /c/Sandbox/reactJS/todo-app (master)
$ docker build -t sanjaysaini2000/react-app .
Sending build context to Docker daemon 563.2kB
Step 1/9 : FROM node:10-alpine as build-step
--> 6bf4087f2679
Step 2/9 : RUN mkdir /app
--> Running in 311bc901e304
Removing intermediate container 311bc901e304
--> 519a41d509c4
Step 3/9 : WORKDIR /app
--> Running in 13367f5792fb
Removing intermediate container 13367f5792fb
--> 8d99997a52f6
Step 4/9 : COPY package.json /app
--> 6d1aee05ca2a
Step 5/9 : RUN npm install
--> Running in eecde26ab721

```

This process will take 1-2 minutes to complete, and at the end, you will get successful message with image tag name.

```

MINGW64:/c/Sandbox/reactJS/todo-app
Find out more about deployment here:
https://bit.ly/CRA-deploy
Removing intermediate container 6b9a4a48e435
--> 6b32f2a12b3f
Step 8/9 : FROM nginx:1.17.1-alpine
--> ea1193fd3dde
Step 9/9 : COPY --from=build-step /app/build /usr/share/nginx/html
--> 0d67272867d5
Successfully built 0d67272867d5
Successfully tagged sanjaysaini2000/react-app:latest
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows Docker host. All files and directories added to build context will have '-rwxr-xr-x' permissions. It is recommended to double check and reset permissions for sensitive files and directories.
sanjaysaini@Lenovo-PC MINGW64 /c/Sandbox/reactJS/todo-app (master)
$

```

You can run following Docker command to list the images created along with your ReactJS app image. You will also find node and nginx images that we used to create our app image and an intermediate image <none>. However these images are not required and can be deleted.

```
$ docker images
```

```

sanjaysaini@lenovo-PC MINGW64 /c/Sandbox/reactJS/todo-app (master)
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
sanjaysaini2000/react-app   latest             6b32f2a1253f       12 minutes ago     274MB
node                  10-alpine          6bf4087f2679       3 weeks ago        83.8MB
nginx                 1.17.1-alpine      ea1193fd3dde       12 months ago      20.6MB

```

## Run the Docker Container

Finally run the following command in the terminal to run your ReactJS Todo App in the Docker container and make sure to replace *sanjaysaini2000/react-app* with your image name in this command.

```
$ docker run -d -it -p 80:80/tcp --name react-app sanjaysaini2000/react-app:latest
```

```

MINGW64/c/Sandbox/reactJS/todo-app
sanjaysaini@lenovo-PC MINGW64 /c/Sandbox/reactJS/todo-app (master)
$ docker run -d -it -p 80:80/tcp --name react-app sanjaysaini2000/react-app:latest
050c3c65225b885bc6f0a5b1b82238a4c9567566fa51544b5b54a95197ed8382
sanjaysaini@lenovo-PC MINGW64 /c/Sandbox/reactJS/todo-app (master)
$

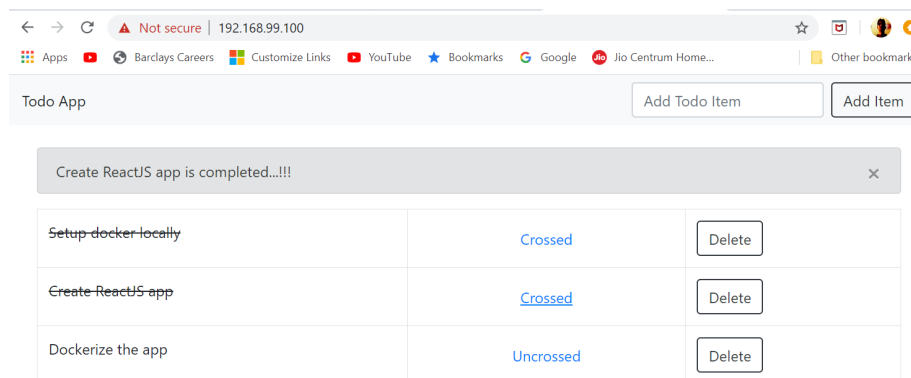
```

Basically, we want to create and run an interactive container in the background so we have used options `-d` and `-it` with the Docker run command. Since app in the container is available at port 80 so we used `-p` option and map the container port to the external host port using `80:80/tcp` and name our container using `--name` option to `react-app` followed by the image name.

Now open the browser and type URL **`http://<docker machine url>:80`** in the address bar.

In my case it's `http://192.168.99.100:80` (`http://192.168.99.100:80`)

Congratulations...you have successfully dockerize and hosted ReactJS app in a Docker container.



This Docker image is also available at my Docker Hub registry as well. So if you don't want to go through above process and only interested in test running this ReactJS app in the Docker container then You can get it from my `react-app` (<https://hub.docker.com/repository/docker/sanjaysaini2000/react-app>) repository at Docker Hub registry.

Keep reading and learning...Cheers!!!

Topics: DOCKER, REACTJS, DEVOPS

Opinions expressed by DZone contributors are their own.

## Popular on DZone

- **DZone Live Episode 5** (/articles/dzone-live-episode-5?fromrel=true)
- **Rest API Documentation and Client Generation With OpenAPI** (/articles/rest-api-documentation-and-client-generation-with?fromrel=true)
- **How to Create a Kubernetes Cluster and Load Balancer for Local Development** (/articles/how-to-create-a-kubernetes-cluster-and-load-balanc?fromrel=true)
- **JavaFX — Overview with Hands-on** (/articles/javafx-overview-with-hands-on-3?fromrel=true)

## Web Dev Partner Resources



### One GraphQL API. Any Data Source. Zero Infrastructure.

Easily connect REST, GraphQL, databases, or any backend in a unified managed GraphQL API. StepZen hosts and manages the service, encrypts your keys to set fine-grained access control, eliminates cloud functions.. [Try for free](#) ►

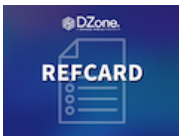
Presented by **StepZen**



### A GraphQL API for Any Data Source

StepZen helps developers connect data from REST, GraphQL, databases, or any backend in a unified managed GraphQL API. Frontend developers can query the backend data sources in a single query without the backend code. [Try for free](#) ►

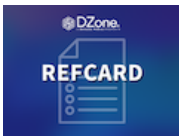
Presented by **StepZen**



### Getting Started With Data Lakes

Learn how a data lake helps tackle data complexity with its enhanced architecture designed to ingest and manage data. [Read now](#) ►

Presented by **Dremio**



### Getting Started With Distributed Systems

Learn the key characteristics of distributed systems and how functionality compares across vendors. [Read now](#) ►

Presented by **MariaDB**



### Get Started With Static Code Analysis

Explore the necessary steps for getting started with static code analysis, including CI/CD integration, Benchmark, and more. [Read now](#) ►

Presented by **ShiftLeft**

## ABOUT US

About DZone (/pages/about)

Send feedback (mailto:support@dzone.com)

Careers (https://devada.com/careers/)

Sitemap (/sitemap)

---

**CONTRIBUTE ON DZONE**

[Article Submission Guidelines \(/articles/dzones-article-submission-guidelines\)](/articles/dzones-article-submission-guidelines)

[MVB Program \(/pages/mvb\)](/pages/mvb)

[Become a Contributor \(/pages/contribute\)](/pages/contribute)

[Visit the Writers' Zone \(/writers-zone\)](/writers-zone)

**LEGAL**

[Terms of Service \(/pages/tos\)](/pages/tos)

[Privacy Policy \(/pages/privacy\)](/pages/privacy)

**CONTACT US**





600 Park Offices Drive

Suite 150

Research Triangle Park, NC 27709

[support@dzone.com \(mailto:support@dzone.com\)](mailto:support@dzone.com)

+1 (919) 678-0300 (tel:+19196780300)

Let's be friends:    

(<https://www.linkedin.com/company/dzone/>)

DZone.com is powered by  (<https://devada.com/answerhub/>)