

github project link: <https://github.com/katleo-rantle/containerized-react-app>

working with docker multi stage builds

nothing fancy goal is to learn how to use docker to build and serve a react app

todo

- spin up react app with `create-react-app` ✓ 2025 12 22
- write multi stage docker file to build react app ✓ 2025 12 27
- extend dockerfile to serve static files thru nginx server ✓ 2025 12 27

create build and serve a react app

a word on CRA:: in 2025 **Create React App (CRA)** has been officially deprecated and is no longer recommended.

```
npx create-react-app --template typescript containerized-react-app
```

npm start uses webpackdevServer

```
yarn create vite containerized-react-app --template react-ts
```

```
yarn run dev
```

```
src/app.tsx edit it if u want
```

 more info >

gemini: When you run the "dev" command, your tools (Vite, Next.js, etc.) prioritize **developer speed over user performance**.

If you want to see how your app actually performs or if you're ready to deploy, you must run the build command:

```
yarn build
```

This creates a `/dist` (Vite) or `/.next` (Next.js) folder containing the highly optimized version of your app.

```
yarn preview
```

This starts a local server that serves the *production* files, allowing you to check performance before you go live.

 Warning: Never deploy the output of a dev server to a real website. It will be 5–10x slower and may expose your source code or sensitive debugging information to the public.

build app

```
yarn build
```

serve the output from build step

`dist` - folder containing build files
trigger package without installing

```
npx http-server@14.1.1 dist
```

<p>note: no hot reload here need to run build command again</p> <p>didn't work coz i was already in the dist folder i needed to</p> <pre>cd ..</pre> <p>then</p> <pre>npx http..</pre>	<pre> 1 # Vite 🌐 2 3 > Next Generation Frontend Tooling PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS ..>/dist/assets > ls .rw-r--r-- 194k rante 22 Dec 12:54 JS index-p_cGuRg.js .rw-r--r-- 4.1k rante 22 Dec 12:54 r react-CHdo9ihT.svg ...>/dist/assets > cd>/containerized-react-app/dist > npx http-server@14.1.1 dist Need to install the following packages: http-server@14.1.1 Ok to proceed? (y) y Starting up http-server, serving dist http-server version: 14.1.1 http-server settings: CORS: disabled Cache: 3400 seconds Connection Timeout: 120 seconds Directory Listings: visible AutoIndex: visible Serve GZIP Files: false Serve Brotli Files: false Default File Extension: none Available on: http://127.0.0.1:8080 http://10.56.202.100:8080 http://172.17.0.1:8080 Hit CTRL-C to stop the server </pre>
---	---

multi stage docker file (Build and serve usecase)

Todo

- 1: build production bundle (files) ✓ 2025-12-22
- 2: serve the bundle with http server

1: build production bundle (files)

in root

```
touch dockerfile .dockerignore
```

```
node_modules
dist
```

alpine = lightweight downloads faster

4: run / install dep

5: copy everything else (safe to do coz we are ignoring some)

```
FROM node:22-alpine

WORKDIR /app

COPY package*.json .

RUN npm ci

COPY . .

RUN npm run build
```

💡 yarn equivalent >

```
FROM node:22-alpine

WORKDIR /app

# 1. Copy both the package.json AND the yarn.lock file
# This is crucial so Yarn can verify the exact versions

COPY package.json yarn.lock .

# 2. Run the Yarn equivalent of 'npm ci' # For Yarn v1 (Classic): --frozen-lockfile # For Yarn v2+ (Modern): --immutable

RUN yarn install --frozen-lockfile

# 3. Copy the rest of your source code
COPY . .
```

```
# 4. Build the application
RUN yarn build
```

create container from this image

First we create the image

.. = curr dir as context

```
docker build -t react-app:alpine .
```

out	docker images
<pre>./projects/containerized-react-app x docker build -t react-app:alpine . => [internal] load build definition from dockerfile => [internal] load metadata for docker.io/library/node:22-alpine => [internal] load .dockerignore => => transferring context: 34B => [internal] load build context => => transferring context: 464.74kB => [1/6] FROM docker.io/library/node:22-alpine@sha256:0340fa682d72068edf603c305bf => CACHED [2/6] WORKDIR /app => CACHED [3/6] COPY package.json yarn.lock . => [4/6] RUN yarn install --frozen-lockfile => [5/6] COPY . => [6/6] RUN yarn build => exporting to image => => exporting layers => => writing image sha256:d60e67ddeb9842695da16e2a81198f73a28b9f535bbdb828bf5990e => => naming to docker.io/library/react-app:alpine ./projects/containerized-react-app }</pre>	<pre>./projects/containerized-react-app) docker images REPOSITORY TAG IMAGE ID CREATED SIZE react-app alpine d60e67ddeb98 4 minutes ago 920MB express_app v0.0.1 c61c034189ec 6 hours ago 1.14GB backend latest 55cadadac3fa 7 days ago 1.25GB nginx latest 576306625d79 12 days ago 152MB python latest 91b058ae471b 13 days ago 1.12GB node latest b514aab1b25f 13 days ago 1.13GB fedora latest 2bdf7178cadc 2 weeks ago 181MB dockurr/windows latest 1d1c11048017 4 weeks ago 382MB ubuntu latest c3a134f2ace4 2 months ago 78.1MB hello-world latest 1b44b5a3e06a 4 months ago 10.1kB postgres 12 56fe80523f20 13 months ago 419MB python 2 68e7be49c28c 5 years ago 902MB</pre>

run container from image we just created

sh = shell command

```
docker run --rm -it react-app:alpine sh
```

we fall inside container in /app dir

```
ls -la

tree build # for npm

# for yarn / vite
tree dist

exit #out of container since we ran --rm container should be removed

# confirm no containers running
docker ps

# or
docker container ls
```

out > same structure as local machine	tree dist
<pre>./projects/containerized-react-app) docker run --rm -it react-app:alpine sh /app # ls -la total 116 drwxr-xr-x 1 root root 32 Dec 22 12:25 . drwxr-xr-x 1 root root 8 Dec 22 13:02 .. -rw-r--r-- 1 root root 0 Dec 22 12:10 .dockerignore -rw-r--r-- 1 root root 253 Dec 22 10:26 .gitignore -rw-r--r-- 1 root root 2555 Dec 22 10:26 README.md drwxr-xr-x 1 root root 48 Dec 22 12:57 dist -rw-r--r-- 1 root root 128 Dec 22 12:25 dockerfile -rw-r--r-- 1 root root 616 Dec 22 10:26 eslint.config.js -rw-r--r-- 1 root root 372 Dec 22 10:26 index.html drwxr-xr-x 1 root root 28 Dec 22 10:54 node_modules -rw-r--r-- 1 root root 724 Dec 22 10:26 package.json drwxr-xr-x 1 root root 16 Dec 22 10:26 public drwxr-xr-x 1 root root 74 Dec 22 10:26 src -rw-r--r-- 1 root root 732 Dec 22 10:26 tsconfig.app.json -rw-r--r-- 1 root root 119 Dec 22 10:26 tsconfig.json -rw-r--r-- 1 root root 653 Dec 22 10:26 tsconfig.node.json -rw-r--r-- 1 root root 161 Dec 22 10:26 vite.config.ts -rw-r--r-- 1 root root 74363 Dec 22 10:31 yarn.lock /app #</pre>	<pre>/app # tree dist dist ├── assets │ ├── index-C0cDBgFa.css │ ├── index-p-_cGuRg.js │ └── react-CHdo91hT.svg └── index.html └── vite.svg 1 directories, 5 files /app #</pre>

2: serve the bundle with http server

name the 1st stage

```
# stage1: build production bundle
FROM node:22-alpine AS build

WORKDIR /app

COPY package.json yarn.lock ./

RUN yarn install --frozen-lockfile

COPY . .

RUN yarn build

# stage2: serve the bundle with http server nginx
FROM nginx:1.27.0

COPY --from=dist /app/dist /usr/share/nginx/html
```

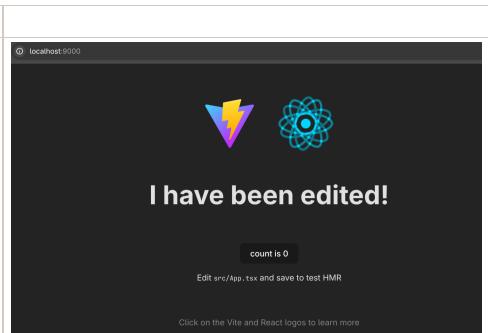
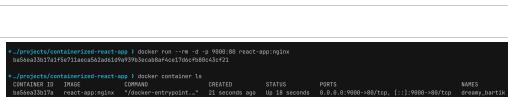
create image > name it

```
docker build -t react-app:nginx .
```

nginx will expose http server on port 80 inside container

```
run container based on image  
map local machine port 9000 to (direct traffic to) container port 80  
use react-app:nginx image
```

```
docker run --rm -d -p 9000:80 react-app:nginx
```



changes to source code > new image

OG code

```
import { useState } from 'react'
import reactLogo from './assets/react.svg'
import viteLogo from '/vite.svg'
import './App.css'

function App() {
  const [count, setCount] = useState(0)

  return (
    <>
      <div>
        <a href="https://vite.dev" target="_blank">
          <img src={viteLogo} className="logo" alt="Vite logo" />
        </a>
        <a href="https://react.dev" target="_blank">
          <img src={reactLogo} className="logo react" alt="React
logo" />
        </a>
      </div>
      <h1>I have been edited!</h1>
      <div className="card">
        <button onClick={() => setCount((count) => count + 1)}>
          count is {count}
        </button>
        <p>
          Edit <code>src/App.tsx</code> and save to test HMR
        </p>
      </div>
      <p className="read-the-docs">
        Click on the Vite and React logos to learn more
      </p>
    </>
  )
}
```

Changes

```
<h1>Hello from blue nginx!</h1>
```

```
export default App
```

```
docker build -t react-app:blue .
```

container port still active

Container Port Still Active Troubleshooting

I had an nginx container on port 80 mapped to port 8000 serving a web page I have stopped the container but the webpage is still available on port 8000 why?

It can be quite confusing to see a website still running when you've explicitly stopped the container that was supposed to be serving it. Usually, this happens due to one of three main reasons: **caching ghost containers**, or a **port conflict**.

Here is how to troubleshoot and identify the culprit.

1. **Browser Caching**

This is the most common reason. Modern browsers are very aggressive about caching static content (HTML, CSS, JS). Even though the server is down, your browser might be showing you a "disk cache" version of the page.

- The Fix:** Hard refresh the page by pressing **Ctrl + F5** (Windows/Linux) or **Cmd + Shift + R** (Mac). Alternatively, try opening the address in an **Incognito/Private window**.

caching was the problem

```
- docker build -t react-app:red .
```

```
- docker run --rm -d -p 8000:80 react-app:red
```

localhost:8000



Hello from red nginx!

count is 0

Edit src/App.tsx and save to test HMR

Click on the Vite and React logos to learn more

stop and remove containers

```
//stop containers
docker stop $(docker ps -q)

// or
docker stop $(docker container ls -q)

// remove containers
docker rm $(docker ps -aq)

// remove images
docker rmi -f $(docker images -q)
```

DONE!!!

what i learnt

To build and ship a react app from a single docker file

How different base images can be used for different stages of our build & ship process (a use case for Multi stage build)