

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

trgger package without installing

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

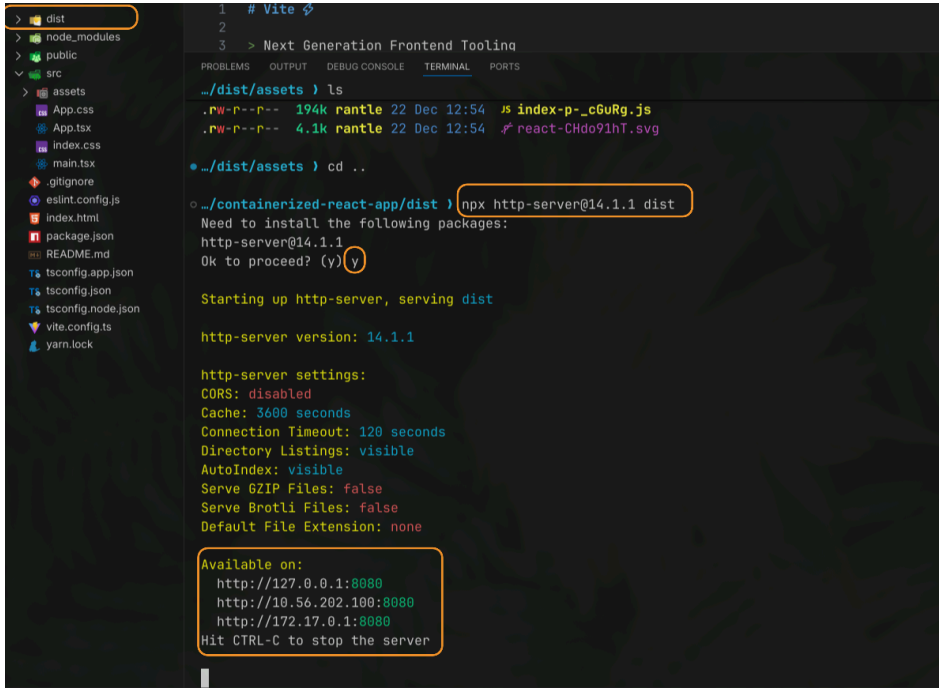
note: no hot reload here need to run build command again

didnt work coz i was already in the dist folder i needed to

cd ..

then

npx http..



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

```
~/projects/containerized-react-app × docker build -t react-app:alpine .
=> [internal] load build definition from dockerfile 0.6s
=> => transferring dockerfile: 167B 0.0s
=> [internal] load metadata for docker.io/library/node:22-alpine 3.8s
=> [internal] load .dockerignore 0.5s
=> => transferring context: 34B 0.0s
=> [internal] load build context 1.2s
=> => transferring context: 464.74kB 0.4s
=> [1/6] FROM docker.io/library/node:22-alpine@sha256:0340fa682d72068edf603c305bf 0.0s
=> CACHED [2/6] WORKDIR /app 0.0s
=> CACHED [3/6] COPY package.json yarn.lock ./ 0.0s
=> [4/6] RUN yarn install --frozen-lockfile 239.0s
=> [5/6] COPY . . 12.2s
=> [6/6] RUN yarn build 12.6s
=> exporting to image 65.3s
=> => exporting layers 64.6s
=> => writing image sha256:d60e67ddeb9842695da16e2a81198f73a28b9f535bbd828bf5990e 0.1s
=> => naming to docker.io/library/react-app:alpine 0.1s

~/projects/containerized-react-app )
```

docker images

```
~/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	2bdf7178cad	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

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

```
~/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 #
```

tree dist

```
/app # tree dist
dist
├── assets
│   ├── index-C0cDBgFa.css
│   ├── index-p-_cGuRg.js
│   └── react-CHDo9lhT.svg
├── index.html
└── vite.svg

1 directories, 5 files
/app #
```

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=build /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
```

outcomes

```
./projects/containerized-react-app $ docker build -t react-app:nginx .
=> Image 1.27f app --from=build /app/dist /usr/share/nginx/html 5.0s
=> exporting to image
=> exporting layers 2.0s
=> writing image sha256:51140b22f6a31726f3b3c3e0a708015271e1fa705b4d070e 0.1s
=> naming to docker.io/library/react-app:nginx 0.1s

./projects/containerized-react-app $ docker images
CONTAINER ID        IMAGE               CREATED             SIZE
react-app          nginx               6 hours ago        100MB
react-app          nginx               6 hours ago        100MB
express_app        v0.8.1             16 hours ago       1.34GB
nginx              550a0a0d37e        2 days ago         1.25GB
nginx              57d30a625d79       12 days ago        102MB
python             91008ba0c710       11 days ago        1.12GB
node              653a0a0135d        11 days ago        1.33GB
Postgres           20d7f178eac0       2 weeks ago        18.5GB
dynamodb           1e3119d4017        4 weeks ago        307MB
quants             c3d13472eac0       2 months ago       78.1MB
k8s-woof1d         12a0c03d0e00       4 months ago       28.1MB
postgres           5d408852f20        11 months ago      439MB
python             68d70e49c28c       5 years ago        902MB
```

```
./projects/containerized-react-app $ docker run --rm -d -p 9000:80 react-app:nginx
b80da301754f47f1a0d4d4a1e9b05c0a8a6c370e0d0b3c421

./projects/containerized-react-app $ docker container ls
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
b80da301754f      react-app:nginx    "/docker-entrypoint..." 25 seconds ago     Up 38 seconds      0.0.0.0:9000->80/tcp, ::::9000->80/tcp   anova-bart1a

./projects/containerized-react-app $
```

localhost:9000

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>
  </>
)
```

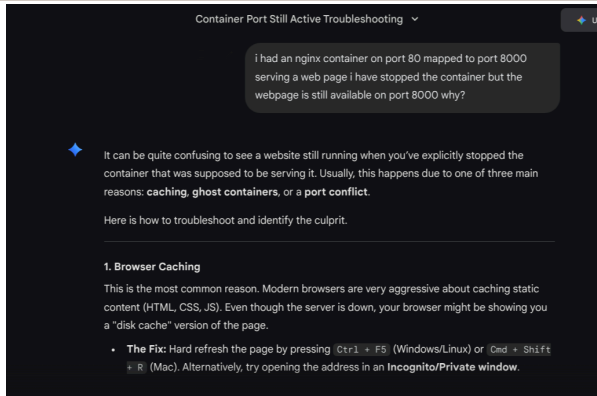
Changes

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

```
export default App
```

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

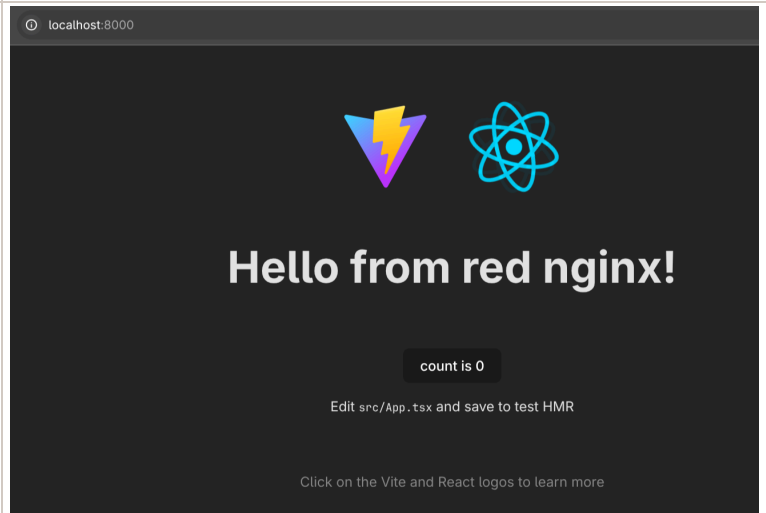
container port still active



caching was the problem

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

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



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)