

## Application Containerization Steps:

For this assignment, I containerized a static Next.js application using Docker. First, I ensured the application was built successfully locally with `npm run build`, which generates static files in the `/out` directory. I created a multi-stage Dockerfile to optimize image size and separate the build process from the final runtime environment.

In the first stage, I used `node:22-alpine` as the base image, installed dependencies using `npm install --force`, and built the app. In the second stage, I used the `nginx:stable-alpine` image to serve the static output. The compiled contents from `/app/out/` were copied to the default nginx folder `/usr/share/nginx/html/`.

## Volumes and Bind Mounts:

Though persistent storage is not critical for a static site, I used a bind mount during development to reflect local file changes without rebuilding the image every time. This was done using the `docker run -v` option (e.g., `-v $(pwd)/out:/usr/share/nginx/html`) to map the local build output to the nginx content folder. This helped test frontend updates quickly during development.

## Challenges and Resolutions:

One challenge was to ensure the build output was compatible with nginx's static file serving structure. I initially used the Next.js server directly but switched to **`npm run build`** and **`next export`** to generate a static version. Another challenge was minimizing image size and avoiding unnecessary files, which I addressed by using a multi-stage build and `.dockerignore` to exclude `node_modules`.

## important command

Build the Docker image:

```
docker build -t next-static-site:v1.0 .
```

Run the container:

```
docker run -d -p 8080:80 next-static-site:v1.0
```

The screenshot shows a VS Code editor with a workspace named 'Untitled (Workspace)'. The file explorer on the left shows a project structure for 'memory-game' with files like .devcontainer, app, components, hooks, lib, public, styles, gitattributes, components.json, docker-compose.yml, Dockerfile, global.css, memory-game.tsx, next-env.d.ts, next.config.mjs, package-lock.json, package.json, postcss.config.mjs, readme.md, tailwind.config.ts, and tsconfig.json. The main editor area shows the 'readme.md' file with the following content:

```
## important command
'''bash
docker build -t next-static-site:v1.0 .
'''
Run the container:
'''bash
docker run -d -p 8880:80 next-static-site:v1.0
'''
The app will be accessible at (http://localhost:8880)(http://localhost:8880).
```

The terminal output shows the execution of the Docker build and run commands. The build process is successful, and the container is running. The terminal output is as follows:

```
rishhi@fenyxs-Laptop memory-game % docker build -t next-static-site:v1.0 .
[+] Building 0.5s (14/14) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfiles: 302B
=> [internal] load metadata for docker.io/library/nginx:stable-alpine
=> [internal] load metadata for docker.io/library/node:22-alpine
=> [internal] load .dockerignore
=> => transferring context: 20B
=> [stage-1 1/3] FROM docker.io/library/node:22-alpine@sha256:9f3ae84fa4d2188025083bf890792f33c39033c9241fc6bb281149470436ca
=> [internal] load build context
=> => transferring context: 21.88kB
=> CACHED [stage-1 2/3] RUN rm -rf /usr/share/nginx/html/*
=> CACHED [builder 2/5] WORKDIR /app
=> CACHED [builder 3/5] COPY - .
=> CACHED [builder 4/5] RUN npm install --force
=> CACHED [builder 5/5] RUN npm run build
=> CACHED [stage-1 3/3] COPY --from=builder /app/out/ /usr/share/nginx/html/
=> exporting to image
=> => exporting layers
=> writing image docker-desktop://dashboard/build/desktop-linux/desktop-
linux/kb43ru3lt13f9w56lx7qmvdf
=> naming to docker-desktop://dashboard/build/desktop-linux/desktop-Linux/kb43ru3lt13f9w56lx7qmvdf

View build details: docker-desktop://dashboard/build/desktop-Linux/desktop-Linux/kb43ru3lt13f9w56lx7qmvdf

What's next:
View a summary of image vulnerabilities and recommendations -> docker scout quickview
rishhi@fenyxs-Laptop memory-game %
```

Run command :

The screenshot shows the same VS Code editor workspace as the first image. The terminal output now shows the execution of the Docker run command. The terminal output is as follows:

```
rishhi@fenyxs-Laptop memory-game % docker run -d -p 8880:80 next-static-site:v1.0
2a8b24fdac1a7695b4ea5d7e1adb628399f13aac8304c390f490ee6fae91ac
rishhi@fenyxs-Laptop memory-game %
```

