☁️Cloud Deployment — Dockerized React App

A production-ready deployment setup for a React application using **Docker multi-stage builds** and **Nginx** for optimized static file serving.

🌳Overview

This project demonstrates how to: - Build a React app using Node.js - Create a **multi-stage Docker build** to reduce final image size - Serve the optimized build using **Nginx** - Run the final application in a lightweight container

---

🦏Tech Stack

Layer	Technology	Purpose
Frontend	React	UI Application
Build Environment	Node 18 (Alpine)	Dependency installation & build
Runtime Environment	Nginx (Alpine)	Serves optimized static files
Containerization	Docker	Portable deployment

🦺Project Structure

```
my-react-app/
│
├── src/
├── public/
├── package.json
├── Dockerfile
└── .dockerignore
```

⛰️.dockerignore

```
node_modules
build
.git
.gitignore
Dockerfile
.dockerignore
```

🌘Dockerfile (Multi-Stage Build)

```
# ---- Build Stage ----
FROM node:18-alpine AS build

WORKDIR /app

COPY package*.json ./
RUN npm install

COPY . .
RUN npm run build

# ---- Production Stage ----
FROM nginx:stable-alpine

COPY --from=build /app/build /usr/share/nginx/html

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]
```

🍁Build & Run

```
docker build -t react-app-prod .
docker run -p 80:80 react-app-prod
```

Then open:

```
http://localhost
```

🏞️Image Size Verification

```
docker images
```

Expected result (optimized):

```
react-app-prod   latest   ~25MB
```

🌀Key Advantages

- Smaller production image
- Clear separation of build & runtime stages
- Faster deployment and startup

Author

Cloud Deployment Project by Deepti ☁️