Day 17 — Nginx Introduction (Serve React via Nginx)

Nginx is a high-performance web server commonly used to serve static files and act as a reverse proxy. In production, React applications are built into static files (HTML, CSS, JS). Nginx is used to serve these files efficiently. Why Nginx for React? • Faster than Node for static files • Lower memory usage • Production-standard • Easy Docker integration Workflow: 1. React app is built using `npm run build` 2. Build folder (`dist` or `build`) contains static files 3. Nginx serves these files on port 80 In Docker: • Multi-stage build is used • Node builds the app • Nginx serves the output Important Rules: • Nginx is for frontend static content • Do NOT use Nginx to run Node backend code • Backend remains a separate container

Day 18 — Multi-Container MERN Architecture

A real MERN application runs multiple services together: • Frontend (React + Nginx) • Backend (Node.js + Express) • Database (MongoDB) Each service runs in its own container. Docker Compose is used to manage them together. Why Multi-Container? • Separation of concerns • Easy scaling • Real production structure • Independent updates Container Roles: Frontend Container: • Built React app • Served by Nginx • Exposes port 80 Backend Container: • Express API • Runs on internal port (e.g., 4000) • Communicates with MongoDB MongoDB Container: • Official Mongo image • Uses volumes for persistent data • No direct public exposure Communication: • Containers talk using service names • Example: backend connects to `mongodb://mongo:27017/dbname`

Nginx vs Backend (Important Difference)

Nginx: • Static file server • Reverse proxy • Handles frontend assets • Very fast and lightweight Backend (Node/Express): • Handles logic and APIs • Authentication • Database communication • Runs application code Rule: Nginx NEVER replaces backend. Nginx and backend always run separately.

Production Best Practices

• Use `.env` for secrets • Never hardcode credentials • Use volumes for MongoDB • Use Docker Compose for orchestration • Frontend and backend must be isolated • Test full app using Docker only Final Outcome: A fully Dockerized MERN app that can run on any server consistently.