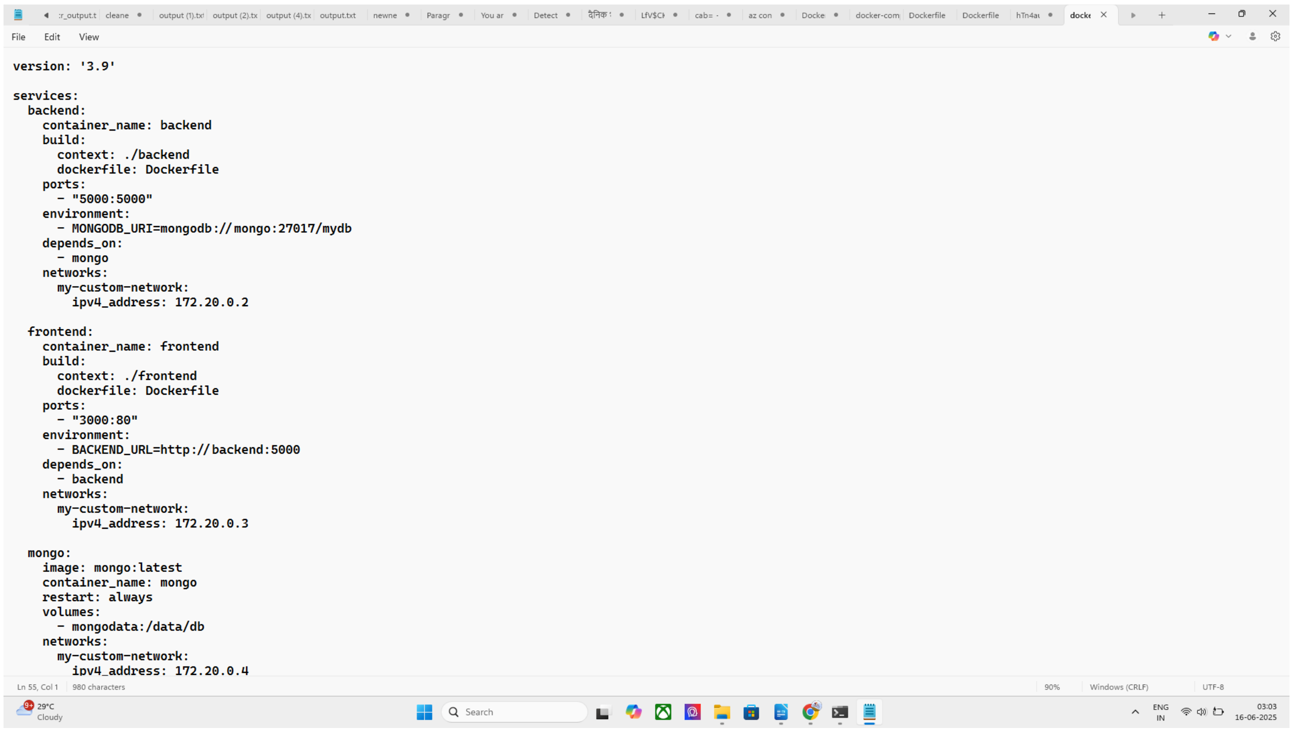
Ques 8 : **Docker Compose for multi-container applications, Docker security best practices**

What it is:
Docker Compose is a tool that lets you define and run multiple containers (services) as one unified application using a YAML file (`docker-compose.yml`).
Why it's useful:

- Instead of starting each container manually, you define everything (networks, volumes, dependencies) in one file.
- You can start all containers with **`docker compose up`**.
- Ideal for full-stack apps (e.g., **React + Node.js + MongoDB** all running together).

Lets create a docker compose file for creating multiple containers at single time containing

:frontend
:backend
:mongodb database

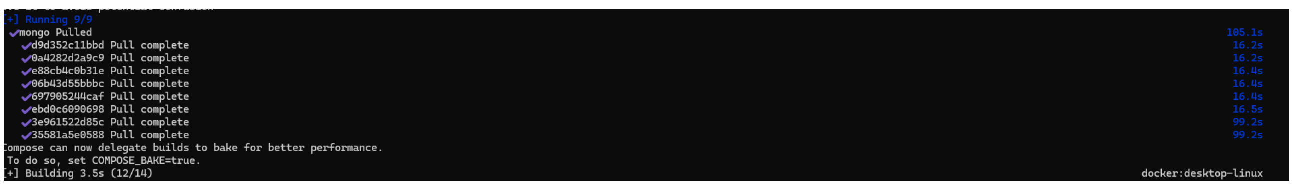


```
version: '3.9'

services:
  backend:
    container_name: backend
    build:
      context: ./backend
      dockerfile: Dockerfile
    ports:
      - "5000:5000"
    environment:
      - MONGODB_URI=mongodb://mongo:27017/mydb
    depends_on:
      - mongo
    networks:
      my-custom-network:
        ipv4_address: 172.20.0.2

  frontend:
    container_name: frontend
    build:
      context: ./frontend
      dockerfile: Dockerfile
    ports:
      - "3000:80"
    environment:
      - BACKEND_URL=http://backend:5000
    depends_on:
      - backend
    networks:
      my-custom-network:
        ipv4_address: 172.20.0.3

  mongo:
    image: mongo:latest
    container_name: mongo
    restart: always
    volumes:
      - mongodata:/data/db
    networks:
      my-custom-network:
        ipv4_address: 172.20.0.4
```

Ln 55, Col 1 | 980 characters | 90% | Windows (CRLF) | UTF-8

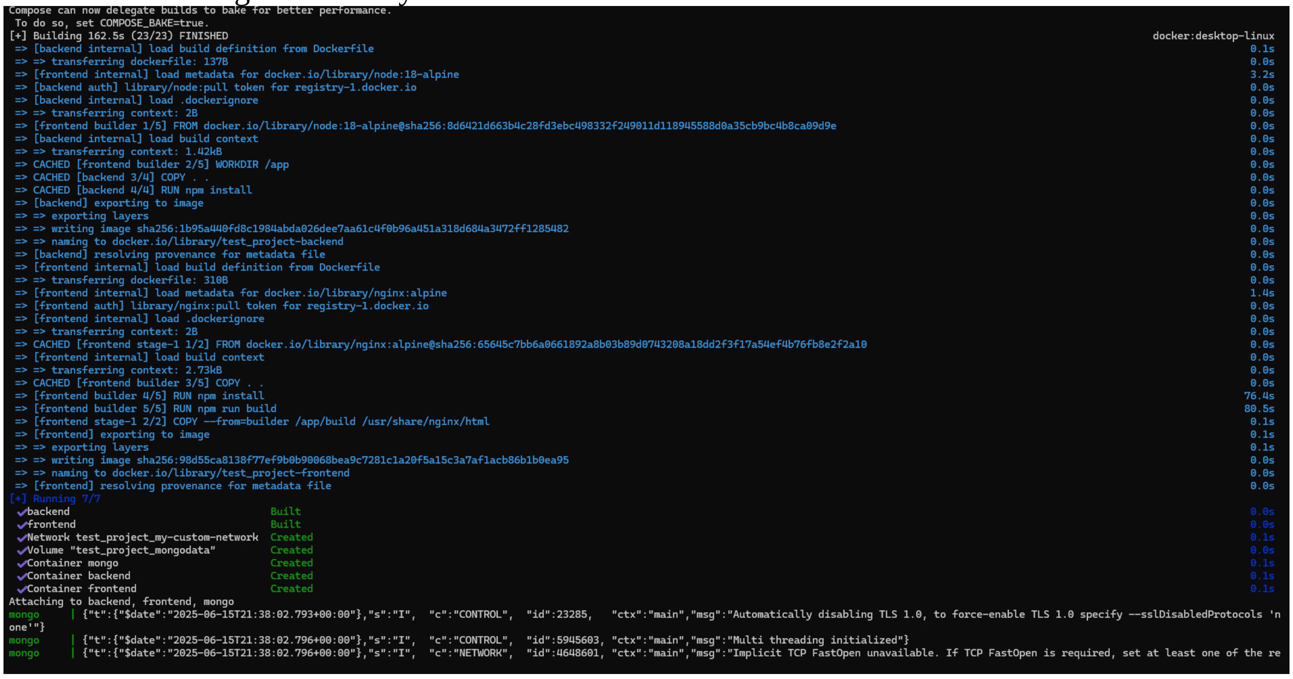
Now use command docker compose up –build
to build the images using docker compose file

Compose file pulled mongo 1st

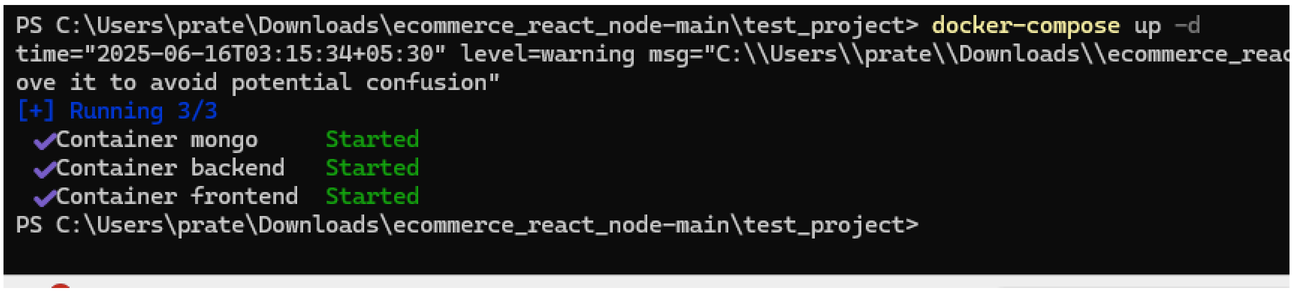


```
[+] Running 9/9
 ✓mongo Pulled                                                                                   105.1s
   ✓d9d352c11bbd Pull complete                                                                    16.2s
   ✓0a4282d2a9c9 Pull complete                                                                    16.2s
   ✓e88cb4c0b31e Pull complete                                                                    16.4s
   ✓06b43d55bbbc Pull complete                                                                    16.4s
   ✓697905244caf Pull complete                                                                    16.4s
   ✓ebd0c6090698 Pull complete                                                                    16.5s
   ✓3e961522d85c Pull complete                                                                    99.2s
   ✓35581a5e0588 Pull complete                                                                    99.2s
ompose can now delegate builds to bake for better performance.
 To do so, set COMPOSE_BAKE=true.
[+] Building 3.5s (12/14)                                                          docker:desktop-linux
```

Containers running successfully :

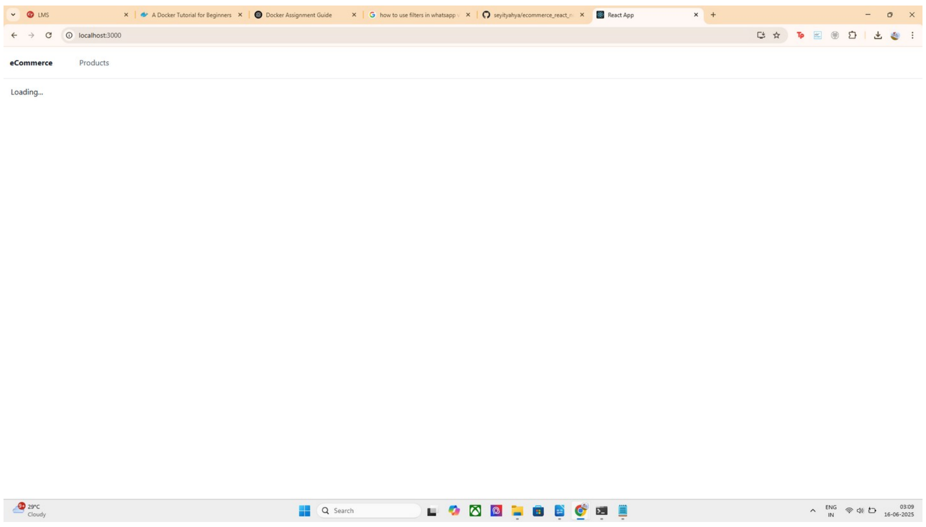


```
Compose can now delegate builds to bake for better performance.
 To do so, set COMPOSE_BAKE=true.
[+] Building 162.5s (23/23) FINISHED                                                docker:desktop-linux
 => [backend internal] load build definition from Dockerfile                                      0.1s
 => => transferring dockerfile: 137B                                                              0.0s
 => [frontend internal] load metadata for docker.io/library/node:18-alpine                        3.2s
 => [backend auth] library/node:pull token for registry-1.docker.io                               0.0s
 => [backend internal] load .dockerignore                                                         0.0s
 => => transferring context: 2B                                                                   0.0s
 => [frontend builder 1/5] FROM docker.io/library/node:18-alpine@sha256:8d6421d663b4c28fd3ebc498332f249011d118945588d0a35cb9bc4b8ca09d9e  0.0s
 => [backend internal] load build context                                                         0.0s
 => => transferring context: 1.42kB                                                               0.0s
 => CACHED [frontend builder 2/5] WORKDIR /app                                                    0.0s
 => CACHED [backend 3/4] COPY . .                                                                 0.0s
 => CACHED [backend 4/4] RUN npm install                                                          0.0s
 => [backend] exporting to image                                                                  0.0s
 => => exporting layers                                                                           0.0s
 => => writing image sha256:1b95a440fd8c1984abda026dee7aa61c4f0b96a451a318d684a3472ff1285482      0.0s
 => => naming to docker.io/library/test_project-backend                                           0.0s
 => [backend] resolving provenance for metadata file                                              0.0s
 => [frontend internal] load build definition from Dockerfile                                     0.0s
 => => transferring dockerfile: 310B                                                              0.0s
 => [frontend internal] load metadata for docker.io/library/nginx:alpine                          1.4s
 => [frontend auth] library/nginx:pull token for registry-1.docker.io                             0.0s
 => [frontend internal] load .dockerignore                                                        0.0s
 => => transferring context: 2B                                                                   0.0s
 => CACHED [frontend stage-1 1/2] FROM docker.io/library/nginx:alpine@sha256:65645c7bb6a0661892a8b03b89d0743208a18dd2f3f17a54ef4b76fb8e2f2a10  0.0s
 => [frontend internal] load build context                                                        0.0s
 => => transferring context: 2.73kB                                                               0.0s
 => CACHED [frontend builder 3/5] COPY . .                                                        0.0s
 => [frontend builder 4/5] RUN npm install                                                       76.4s
 => [frontend builder 5/5] RUN npm run build                                                     80.5s
 => [frontend stage-1 2/2] COPY --from=builder /app/build /usr/share/nginx/html                   0.1s
 => [frontend] exporting to image                                                                 0.1s
 => => exporting layers                                                                           0.1s
 => => writing image sha256:98d55ca8138f77ef9b0b90068bea9c7281c1a20f5a15c3a7af1acb86b1b0ea95      0.0s
 => => naming to docker.io/library/test_project-frontend                                          0.0s
 => [frontend] resolving provenance for metadata file                                             0.0s
[+] Running 7/7
 ✓backend                              Built                                                      0.0s
 ✓frontend                             Built                                                      0.0s
 ✓Network test_project_my-custom-network  Created                                                 0.1s
 ✓Volume "test_project_mongodata"      Created                                                    0.0s
 ✓Container mongo                      Created                                                    0.1s
 ✓Container backend                    Created                                                    0.1s
 ✓Container frontend                   Created                                                    0.1s
Attaching to backend, frontend, mongo
mongo     | {"t":{"$date":"2025-06-15T21:38:02.793+00:00"},"s":"I",  "c":"CONTROL",  "id":23285,   "ctx":"main","msg":"Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'n
one'"}
mongo     | {"t":{"$date":"2025-06-15T21:38:02.796+00:00"},"s":"I",  "c":"CONTROL",  "id":5945603, "ctx":"main","msg":"Multi threading initialized"}
mongo     | {"t":{"$date":"2025-06-15T21:38:02.796+00:00"},"s":"I",  "c":"NETWORK",  "id":4648601, "ctx":"main","msg":"Implicit TCP FastOpen unavailable. If TCP FastOpen is required, set at least one of the re
```



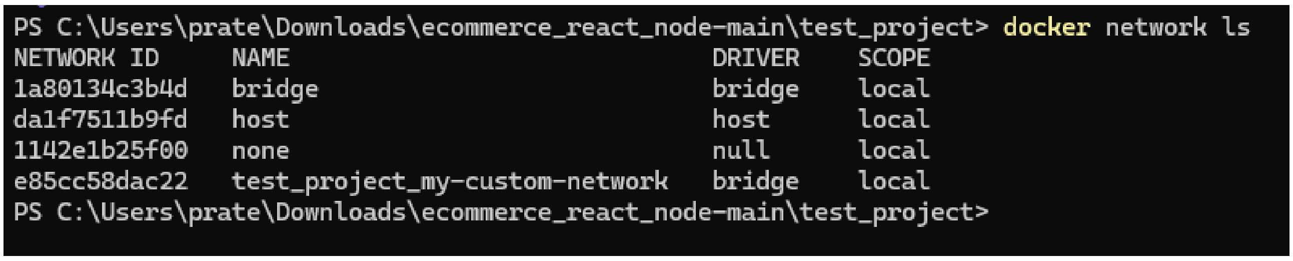
```
PS C:\Users\prate\Downloads\ecommerce_react_node-main\test_project> docker-compose up -d
time="2025-06-16T03:15:34+05:30" level=warning msg="C:\\Users\\prate\\Downloads\\ecommerce_reac
ove it to avoid potential confusion"
[+] Running 3/3
 ✓Container mongo      Started
 ✓Container backend    Started
 ✓Container frontend   Started
PS C:\Users\prate\Downloads\ecommerce_react_node-main\test_project>
```

TEST E COMMERE WEBSITE ON PORT 3000



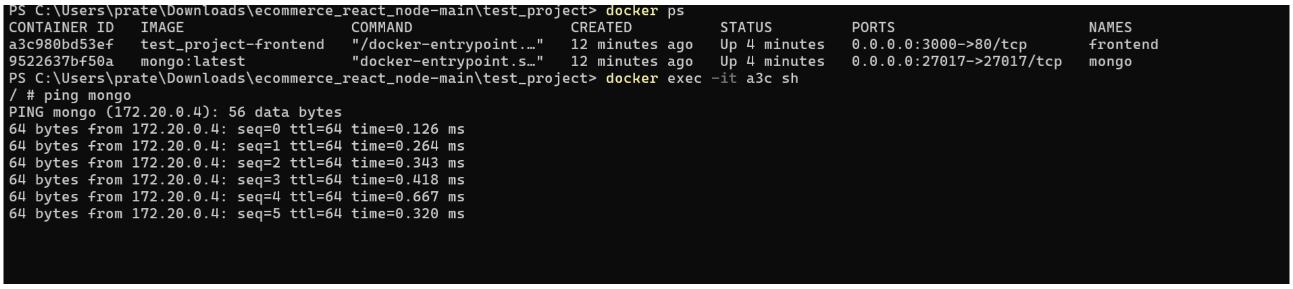
LMS
A Docker Tutorial for Beginners
Docker Assignment Guide
how to use filters in whatsapp
seyityahya/ecommerce_react_n
React App
localhost:3000
eCommerce
Products
Loading...
Search

Lets test Networking by command : docker network ls



```
PS C:\Users\prate\Downloads\ecommerce_react_node-main\test_project> docker network ls
NETWORK ID     NAME                             DRIVER    SCOPE
1a80134c3b4d   bridge                           bridge    local
da1f7511b9fd   host                             host      local
1142e1b25f00   none                             null      local
e85cc58dac22   test_project_my-custom-network   bridge    local
PS C:\Users\prate\Downloads\ecommerce_react_node-main\test_project>
```

connected to network : verified



```
PS C:\Users\prate\Downloads\ecommerce_react_node-main\test_project> docker ps
CONTAINER ID   IMAGE                    COMMAND                  CREATED          STATUS         PORTS                        NAMES
a3c980bd53ef   test_project-frontend    "/docker-entrypoint.…"   12 minutes ago   Up 4 minutes   0.0.0.0:3000->80/tcp         frontend
9522637bf50a   mongo:latest             "docker-entrypoint.s…"   12 minutes ago   Up 4 minutes   0.0.0.0:27017->27017/tcp     mongo
PS C:\Users\prate\Downloads\ecommerce_react_node-main\test_project> docker exec -it a3c sh
/ # ping mongo
PING mongo (172.20.0.4): 56 data bytes
64 bytes from 172.20.0.4: seq=0 ttl=64 time=0.126 ms
64 bytes from 172.20.0.4: seq=1 ttl=64 time=0.264 ms
64 bytes from 172.20.0.4: seq=2 ttl=64 time=0.343 ms
64 bytes from 172.20.0.4: seq=3 ttl=64 time=0.418 ms
64 bytes from 172.20.0.4: seq=4 ttl=64 time=0.667 ms
64 bytes from 172.20.0.4: seq=5 ttl=64 time=0.320 ms
```

At the end, I successfully containerized a multi-service application with Docker Compose, including frontend (React), backend (Node.js), and a MongoDB database. I ensured smooth inter-container communication via a custom bridge network, verified using ping tests between services. This validates the use of Docker Compose as an efficient tool to orchestrate, isolate, and run multi-container apps in a clean and reproducible manner.

Docker Security Best Practices

1. **Use trusted images** (official or verified) — don't blindly pull unknown ones from Docker Hub.
2. **Keep containers minimal** — use Alpine or slim images to reduce attack surface.
3. **Don't run as root** inside containers — create a non-root user in your Dockerfile.
4. **Use `.dockerignore`** — to avoid uploading secrets (e.g., `.env`, `node_modules`) into images.
5. **Scan images** for vulnerabilities (e.g., with Docker Scout, Trivy).
6. **Keep Docker and your base images updated** regularly.
7. **Limit container privileges** using flags like `--read-only`, `--cap-drop`, and avoid `--privileged`.

Docker Security (Practical Application)

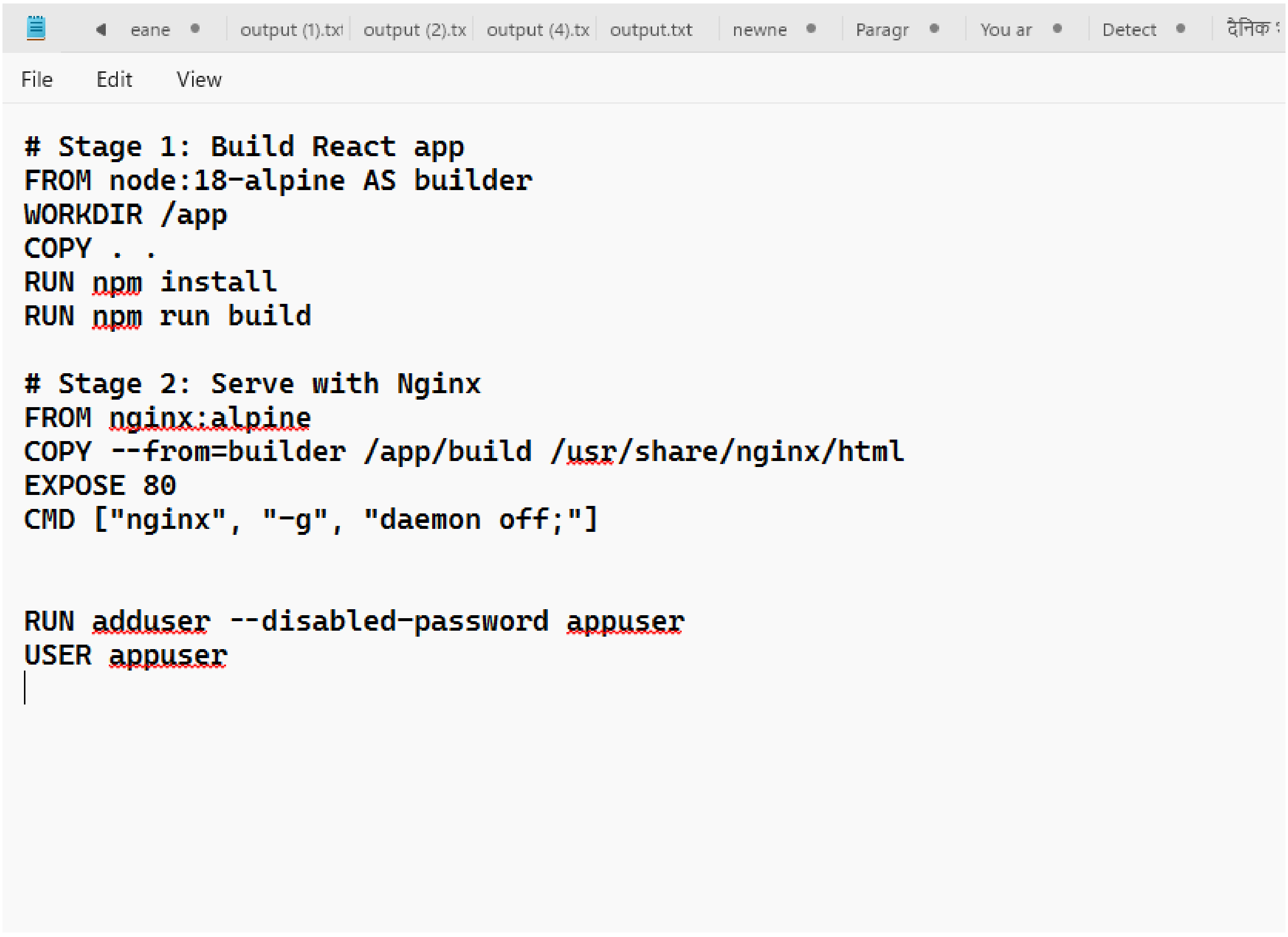
Let's say you're deploying a **public-facing web service**.

You follow these **security best practices**:

Practical Steps:

1. **Use Alpine-based images** like `node:18-alpine` to reduce size and vulnerabilities.
2. **Add a non-root user** in your Dockerfile:

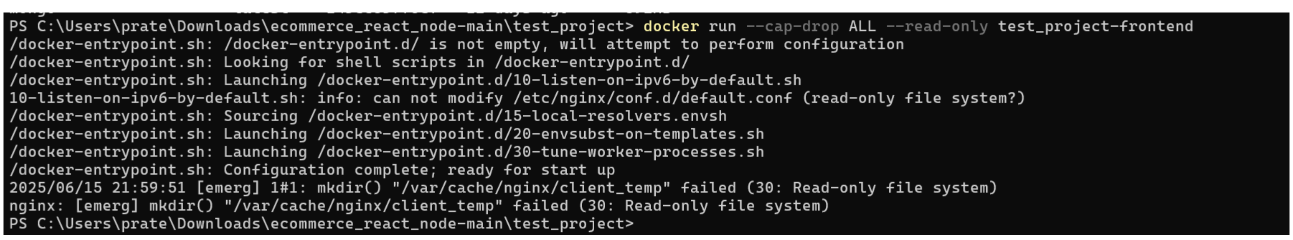
   ```
   Dockerfile

   RUN adduser --disabled-password appuser
   USER appuser
   ```



```
# Stage 1: Build React app
FROM node:18-alpine AS builder
WORKDIR /app
COPY . .
RUN npm install
RUN npm run build

# Stage 2: Serve with Nginx
FROM nginx:alpine
COPY --from=builder /app/build /usr/share/nginx/html
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]


RUN adduser --disabled-password appuser
USER appuser
```

3. **Limit container capabilities:**

   ```
   docker run --cap-drop ALL --read-only my-image
   ```



```
PS C:\Users\prate\Downloads\ecommerce_react_node-main\test_project> docker run --cap-drop ALL --read-only test_project-frontend
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: can not modify /etc/nginx/conf.d/default.conf (read-only file system?)
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2025/06/15 21:59:51 [emerg] 1#1: mkdir() "/var/cache/nginx/client_temp" failed (30: Read-only file system)
nginx: [emerg] mkdir() "/var/cache/nginx/client_temp" failed (30: Read-only file system)
PS C:\Users\prate\Downloads\ecommerce_react_node-main\test_project>
```

4. **Avoid putting `.env`, API keys, etc. in the image** — use environment variables or Docker secrets.
5. **Scan your image before pushing** (e.g., using Trivy).