**1) Build the React app (on EC2)**

cd /opt/ProductionEC2/process-eng-app

# Install dependencies (only once or after package.json changes)  
npm ci

# Build static files  
npm run build

This creates:  
/opt/ProductionEC2/process-eng-app/build/

**2) Copy build output into Nginx’s web root**

sudo mkdir -p /var/www/process-eng-app

sudo rsync -a --delete /opt/ProductionEC2/process-eng-app/build/ /var/www/process-eng-app/

This command copies your React build output into the Nginx web folder, making sure old files are removed and new files are updated. After running it, your latest React version is what users see in their browsers.

**1. rsync**A Linux tool for **copying and synchronizing files/folders**.  
It’s smarter than cp because it only copies changes, not everything from scratch.

**2. -a  
Archive mode**.  
Preserves:  
- File permissions  
- Ownership  
- Timestamps  
- Symbolic links  
- Directory structure

* In short → it makes the copy look identical to the source.

**3. –delete**Deletes any files in the **destination** (/var/www/process-eng-app/) that no longer exist in the **source** (/opt/ProductionEC2/process-eng-app/build/).  
- Keeps the target in sync with the latest React build output.  
- Without this, old/removed files might stick around and confuse your web app.

**4. Source folder (/opt/ProductionEC2/process-eng-app/build/)**This is where React puts the freshly built static files (index.html, JS bundles, CSS, etc.).

**5. Destination folder (/var/www/process-eng-app/)**This is the folder served by Nginx to users’ browsers.  
After syncing, Nginx will show your new build.

**6. sudo  
-** Required because /var/www/... belongs to root (system folder).  
- Without sudo, your normal ubuntu user can’t write there.

**3) Configure Nginx (React + API proxy)**

Create/update a site config:

sudo tee /etc/nginx/sites-available/process-eng-app >/dev/null <<'NGINX'

server {

listen 80;

server\_name 3.149.53.156;

# Serve React static build

root /var/www/process-eng-app;

index index.html;

# SPA routing fallback

location / {

try\_files $uri /index.html;

}

# Proxy API calls to Node.js server (assumes Node on 5000)

location /api/ {

proxy\_pass http://127.0.0.1:5000/;

proxy\_http\_version 1.1;

proxy\_set\_header Connection "";

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header X-Forwarded-Proto $scheme;

}

}

NGINX

# Enable and reload

sudo ln -sf /etc/nginx/sites-available/process-eng-app /etc/nginx/sites-enabled/

sudo nginx -t && sudo systemctl reload nginx

**4) Security Group**

In AWS → EC2 → Security Groups → Inbound rules → open port 80 to 0.0.0.0/0.

But, after doing so, I’ve understood that AWS already had the HTTP security group (probably creates it by default).

**Steps to Edit Inbound Rules**

1. **Find the right Security Group**
   * In your EC2 **Instances** page, select your running instance.
   * Scroll down to the **Security** tab → you’ll see which security group(s) are attached (for example, launch-wizard-2).
2. **Click the Security Group ID**
   * From your screenshot, those are the blue links like sg-0eb4f18cc56435d04.
   * Clicking it will open the details page for that group.
3. **Go to the Inbound rules tab**
   * There you’ll see a list of allowed inbound ports (e.g., SSH on port 22).
4. **Edit inbound rules**
   * Click **Edit inbound rules**.
   * Add a new rule:
     + **Type:** HTTP (for port 80) or HTTPS (for port 443).
     + **Source:** 0.0.0.0/0 (meaning all IPs, if you want it public).
   * Save the rules.

Then browse:

http://<YOUR\_ELASTIC\_IP>/

**5) Deploy flow for updates**

When you change React code:

cd /opt/ProductionEC2/process-eng-app

npm ci # only if deps changed

npm run build

sudo rsync -a --delete build/ /var/www/process-eng-app/

sudo systemctl reload nginx # optional, usually not needed

**6) Create .env.production for process-eng-app**