✅ Steps to Connect Backend to a Domain on EC2

🔧 Step 1: Get a Domain & Point DNS to EC2

If you haven't already:

1. **Buy a domain** (e.g., from Namecheap, GoDaddy, etc.)
2. Go to your **domain's DNS settings**
3. **Create an A record** like this:

Type	Name	Value
A	`@`	`YOUR_EC2_IP`
A	`api`	`YOUR_EC2_IP`

So `api.yourdomain.com` → your EC2 public IP.

🐳 Step 2: Expose the Docker Container Port

Your container is exposing `5000`, so you need to map that to your host.

In your `docker-compose.yml` or when running manually:

yaml
CopyEdit

```yaml
ports:
  - "5000:5000"
```

Or:

bash
CopyEdit

```
docker run -p 5000:5000 your-backend-image
```

⚙️ Step 3: Install & Configure Nginx on EC2

Log into your EC2 instance:

bash
CopyEdit
```
sudo apt update && sudo apt install nginx -y
```

Create Nginx config:

bash
CopyEdit
```
sudo nano /etc/nginx/sites-available/backend
```

Paste this (replace domain):

nginx
CopyEdit
```
server {
  listen 80;
  server_name api.yourdomain.com;

  location / {
    proxy_pass http://localhost:5000;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection 'upgrade';
    proxy_set_header Host $host;
    proxy_cache_bypass $http_upgrade;
  }
}
```

Enable config:

bash
CopyEdit

```
sudo ln -s /etc/nginx/sites-available/backend /etc/nginx/sites-enabled/
sudo nginx -t
sudo systemctl restart nginx
```

🔒 Step 4: Secure It with HTTPS (Let’s Encrypt)

bash
CopyEdit

```
sudo apt install certbot python3-certbot-nginx -y
sudo certbot --nginx -d api.yourdomain.com
```

It will automatically edit your Nginx config and set up HTTPS.

✅ Step 5: Check

Visit:

arduino
CopyEdit

```
https://api.yourdomain.com
```

Your backend should be live and secure 🎉

✅ Notes on `entrypoint.sh`

Make sure your `entrypoint.sh` is running your backend app, e.g.,

bash
CopyEdit

```
#!/bin/sh

# Optional: wait-for database or other services
# ./wait-for-it.sh db:5432 -- echo "Database is up"

npm run start  # or whatever your backend entry point is
```

Make it executable:

bash
CopyEdit
```
chmod +x entrypoint.sh
```

If you'd like help writing your actual `entrypoint.sh` or validating DNS setup, feel free to share the tech stack or backend command (e.g., `node index.js`, `nest start`, etc.)

Thanks for the detailed explanation — I've read and understood everything.

✅ Current Setup

- Your **backend** is deployed on EC2 and connected to the subdomain: `https://api.pristinecouture.us`.
- You used **Docker + Nginx + Let's Encrypt** for this setup.
- Now you want to deploy your **Admin React app**, built with Docker and served via Nginx, and expose it at a new subdomain — e.g., `https://admin.pristinecouture.us`.

✅ Goal

Deploy the admin panel container on the **same EC2 instance**, and connect it to the subdomain `admin.pristinecouture.us`.

🧩 Step-by-Step Guide to Add Admin Panel

🔧 Step 1: Add DNS Record for Admin Subdomain

On your domain registrar's DNS settings (e.g., Namecheap/GoDaddy), add:

Type	Name	Value
A	admin	YOUR_EC2_IP

This points `admin.pristinecouture.us` to your EC2 instance.

🐳 Step 2: Run the Admin Docker Container

Make sure port **3000** (or any available one) is free on the host. You can run the container like this:

bash
CopyEdit

```
docker build -t admin-panel .
docker run -d --name admin-panel -p 3000:80 admin-panel
```

This maps the container's internal Nginx port 80 → host's port 3000.

If you're using Docker Compose, your `docker-compose.yml` might include:

yaml
CopyEdit

```
admin:
  build: .
  ports:
    - "3000:80"
```

⚙️ Step 3: Configure Nginx to Proxy the Admin Subdomain

Create a new Nginx config file:

bash
CopyEdit

```
sudo nano /etc/nginx/sites-available/admin
```

Paste this (update the domain):

nginx
CopyEdit

```
server {
  listen 80;
  server_name admin.pristinecouture.us;

  location / {
    proxy_pass http://localhost:3000;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection 'upgrade';
    proxy_set_header Host $host;
    proxy_cache_bypass $http_upgrade;
  }
}
```

Then:

bash
CopyEdit

```
sudo ln -s /etc/nginx/sites-available/admin /etc/nginx/sites-enabled/
sudo nginx -t
sudo systemctl restart nginx
```

🔒 Step 4: Set Up HTTPS with Certbot

Run Certbot again to issue a cert for the new subdomain:

bash
CopyEdit

```
sudo certbot --nginx -d admin.pristinecouture.us
```

It'll update your Nginx config automatically for HTTPS.

✅ Step 5: Visit the Admin Panel

Now check:

bash
CopyEdit

```
https://admin.pristinecouture.us
```

You should see your React Admin Panel!

❗ Notes & Tips

- Your Docker image is great for static React hosting via Nginx.

- Make sure the **Admin container is auto-started** on reboot (`--restart unless-stopped` in Docker run).

- You can also use `docker-compose` for both backend and admin setups to keep things organized.

- Ensure **port 3000** is allowed in EC2's **Security Group**, or better, only allow localhost access if traffic comes only via Nginx.