Deploying this to AWS

Step 1: Prepare AWS Infrastructure

1.Create an AWS Account

2.Launch an EC2 Instance:

- AMI: Ubuntu 22.04 LTS
- Instance Type: t2.medium (or higher for production)
- Storage: 30GB (SSD)
- Security Group: Open ports 22 (SSH), 80 (HTTP), 443 (HTTPS), 3000 (Frontend), 3001 (Backend)
- Key Pair: Create/download a `.pem` file for SSH access

3.Connect to EC2 Instance:

```
chmod 400 mykey.pem
```

```
ssh -i my.pem ubuntu@my-ec2-public-ip
```

Step 2: Install Prerequisites on EC2

Run these commands on your EC2 instance

```
sudo apt update && sudo apt upgrade -y
sudo apt install docker.io -y
sudo systemctl enable docker
sudo systemctl start docker
sudo curl -L
"https://github.com/docker/compose/releases/latest/download/docker-
compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
docker --version
docker-compose --version
```

Step 3: Deploy Your Application

1.**Upload the Code** to EC2:

```
# On your local machine
scp -i your-key.pem -r ./folder ubuntu@your-ec2-public-ip:/home/ubuntu/
```

2.**Set Up Environment Variables**:

Create a `.env` file in `/backend` and `/frontend` with your configurations

```
# Backend .env example
```

```
DB_NAME=name
```

```
DB_USER=user
```

```
DB_PASSWORD=pwd
```

```
JWT_SECRET=secret
```

3.**Modify the provided `docker-compose.yml` for Production env**

```yaml
version: "3.8"
services:
  db:
    image: postgres:13
    environment:
      POSTGRES_DB: ${DB_NAME}
      POSTGRES_USER: ${DB_USER}
      POSTGRES_PASSWORD: ${DB_PASSWORD}
    volumes:
      - postgres_data:/var/lib/postgresql/data
    restart: always

  backend:
    build: ./backend
    ports:
      - "3001:3001"
    environment:
      - DB_HOST=db
    depends_on:
      - db
    restart: always

  frontend:
    build: ./frontend
    ports:
      - "3000:3000"
    depends_on:
      - backend
    restart: always

volumes:
```

```
  postgres_data:
```

Step 4: Build and Run Containers

```
# Navigate to the project folder
cd /home/ubuntu/folder

# Build and start containers
sudo docker-compose up -d --build

# Verify containers are running
sudo docker ps
```

Step 5: Set Up Nginx Reverse Proxy (Recommended)

1.Install Nginx

```
sudo apt install nginx -y
```

2.Configure Nginx for Frontend

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

```
server {
    listen 80;
    server_name my-domain.com;
    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;
    }
}
```

3.Configure Nginx for Backend

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

```
server {
    listen 3001;
    server_name api.my-domain.com;

    location / {
        proxy_pass http://localhost:3001;
        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;
    }
}
```

4.Enable the configurations:

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

Step 6: Set Up HTTPS with Certbot

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

Troubleshooting

•Check logs:

```
sudo docker-compose logs
```

•Restart services:

```
sudo docker-compose restart
```

The app should now be live at `http://my-ec2-public-ip:3000` (frontend) and `http://my-ec2-public-ip:3001` (backend)!