Deploying Nginx + Django + Gunicron + react production + postgresql

1. – first check all the necessary things from development team/part
   1. For Django requirement.txt
   2. Gunicron configations
   3. For react package.json
   4. For nginx .conf file
2. This project will deploy using docker compose for ease of use and following the production level standards
3. Here’s the example directory structure

my\_project/

│

├── django/

│ ├── app/

│ ├── Dockerfile

│ ├── requirements.txt

│ ├── manage.py

│ └── ...

│

├── react + nginx/

│ ├── src/

│ ├── public/

│ ├── Dockerfile

│ ├── package.json

| nginx.conf

│ └── ...

│

├── nginx/

│ ├── Dockerfile

│ ├── nginx.conf

│ └── certbot/

│ └── init-letsencrypt.sh

│

| -- db

* Dockerfile
* .env file
* Init-data.db

├── docker-compose.yml

└── .env

A ) Django

Add this docker file in your project file

Replace .env file and requirement.txt file

Docker file

# backend/Dockerfile

FROM python:3.12-alpine

WORKDIR /backend

COPY requirements.txt requirements.txt

RUN pip install -r requirements.txt

COPY . .

# Copy the entrypoint script

COPY entrypoint.sh /entrypoint.sh

RUN chmod +x /entrypoint.sh

# Set the entrypoint to the script

ENTRYPOINT ["/entrypoint.sh"]

#CMD ["gunicorn", "app.wsgi:application", "--bind", "0.0.0.0:8000"]

B ) react + nginx ( for production )

Add this docker file in your react project folder where package.json is located

# Stage 1: Build the React app

FROM node:latest AS build

# Set the working directory

WORKDIR /usr/src/app

# Copy package.json and package-lock.json to the container

COPY package\*.json ./

# Set up npm cache in a designated directory to improve caching

RUN --mount=type=cache,target=/usr/src/app/.npm \

npm set cache /usr/src/app/.npm && \

npm install

# Copy the rest of the application code to the container

COPY . .

# Build the React app for production

RUN npm run build

#CMD ["chmod+x", "777", "build"]

# Stage 2: Serve the React app with Nginx

FROM nginx:alpine

# Copy the build output from the builder stage to the Nginx html directory

COPY --from=build /usr/src/app/build /usr/share/nginx/html

# Copy custom Nginx configuration file (if any)

COPY nginx.conf /etc/nginx/nginx.conf

# Expose port 80

EXPOSE 80

# Start Nginx

CMD ["nginx", "-g", "daemon off;"]

Second step for react + nginx

Add this nginx.conf file in same directory as react

**Nginx.conf** file

events {

worker\_connections 1024;

}

http {

include /etc/nginx/mime.types;

server {

listen 80;

server\_name localhost;

location / {

root /usr/share/nginx/html;

index index.html;

try\_files $uri $uri/ /index.html;

}

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root /usr/share/nginx/html;

}

}

}

**Nginx as Web Server**

add this docker file in nginx named folder

# nginx/Dockerfile

FROM nginx:alpine

COPY nginx.conf /etc/nginx/nginx.conf

Add this nginx file in nginx named folder

**Nginx.conf file**

# nginx/nginx.conf

server {

listen 80;

server\_name your\_domain www.your\_domain;

location / {

proxy\_pass http://frontend:80;

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;

}

location /api/ {

proxy\_pass http://backend:8000;

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;

}

}

**Auto - SSL insert using shell**

# nginx/certbot/init-letsencrypt.sh

#!/bin/bash

domains=(your\_domain www.your\_domain)

rsa\_key\_size=4096

data\_path="./nginx/certbot"

email="your\_email@example.com"

staging=0 # Set to 1 for testing

# Add certbot commands and hooks here

**Docker compose file**

version: '3.9'

services:

  db:

    image: postgres:latest

    volumes:

      - postgres\_data:/var/lib/postgresql/data/

    environment:

      POSTGRES\_USER: ${POSTGRES\_USER}

      POSTGRES\_PASSWORD: ${POSTGRES\_PASSWORD}

      POSTGRES\_DB: ${POSTGRES\_DB}

    networks:

      - backend

    restart: always

  Django:

    build: ./django

    environment:

      DB\_HOST: db

      DB\_PORT: 5432

      POSTGRES\_USER: ${POSTGRES\_USER}

      POSTGRES\_PASSWORD: ${POSTGRES\_PASSWORD}

      POSTGRES\_DB: ${POSTGRES\_DB}

    volumes:

      - ./Django:/backend

    env\_file:

      - ./.env

    networks:

      - backend

    depends\_on:

      db

    restart: always

  React-Production:

    build: ./

    stdin\_open: true

    tty: true

    volumes:

      - ./frontend:/frontend

    networks:

      - frontend

  nginx:

    build: ./nginx

    ports:

      - "80:80"

      - "443:443"

    volumes:

      - ./nginx/certbot/conf:/etc/nginx/ssl

      - ./nginx/certbot/www:/var/www/certbot

      - ./nginx/nginx.conf:/etc/nginx/nginx.conf

    depends\_on:

      - backend

      - frontend

    networks:

      - backend

      - frontend

    restart: always

  certbot:

    image: certbot/certbot

    volumes:

      - ./nginx/certbot/conf:/etc/letsencrypt

      - ./nginx/certbot/www:/var/www/certbot

    entrypoint: "/bin/sh -c 'trap exit TERM; while :; do certbot renew; sleep 12h & wait $${!}; done;'"

    networks:

      - frontend

      - backend

volumes:

  postgres\_data:

networks:

  backend:

  frontend: