FastAPI with NGINX Reverse Proxy in Docker

This project sets up a **FastAPI** application running behind an **NGINX reverse proxy** using Docker and Docker Compose.

 Banner

📖 Table of Contents

🚀 Features

- **FastAPI** backend running in a Docker container.
- **NGINX** as a reverse proxy to serve FastAPI.
- **Docker Compose** for easy deployment and management.
- **Swagger UI** for API documentation and testing.

📌 Prerequisites

- Docker & Docker Compose installed.
- Basic knowledge of FastAPI, NGINX, and Docker.

🌍 Why Use a Reverse Proxy?

A **reverse proxy** like NGINX is used in this project to:

- **Improve Security**: Protects the backend by hiding internal infrastructure.
- **Load Balancing**: Distributes traffic when multiple FastAPI instances are running.
- **SSL Termination**: Handles HTTPS at the proxy level.
- **Performance Optimization**: Caching and compression features can improve response times.
- **Custom Domain Handling**: Allows mapping FastAPI to a friendly domain instead of an IP and port.
  For Detials: https://github.com/tssundarraj/Reverse-Proxy-with-Nginx-and-Docker.git

🗁 Project Structure

```
Nginx-reverse-proxy-with-Docker/
├── app/
│   ├── main.py
│   ├── Dockerfile
├── nginx/
│   ├── nginx.conf
├── docker-compose.yml
├── README.md
```

🛠 Setup Instructions

Step 1: Clone the Repository

```
git clone https://github.com/tssundarraj/Nginx-reverse-proxy-with-Docker.git
cd Nginx-reverse-proxy-with-Docker
```

Step 6: Build and Run the Containers

Run the following command to build and start the containers:

```
docker-compose up --build -d
```

Step 7: Add Entry in `/etc/hosts`

Modify the `/etc/hosts` file to map `localhost` to the local environment:

```
echo "127.0.0.1 fastapi.local" | sudo tee -a /etc/hosts
```

This will allow you to access the FastAPI app via `http://fastapi.local` instead of using an IP address.

🌐 Accessing Swagger UI

FastAPI automatically provides interactive API documentation using **Swagger UI**.

- Open your browser and visit:

  ```
  http://fastapi.local/docs
  ```

  or

  ```
  http://127.0.0.1/docs
  ```

- For **Redoc UI**, visit:

  ```
  http://fastapi.local/redoc
  ```

🔎 Testing the Setup

Open your browser and visit:

```
http://fastapi.local
```

Or use `curl`:

```
curl http://fastapi.local
```

🗑️ Clean Up

To stop and remove the running containers, use:

```
docker-compose down
```

To remove all images, volumes, and networks created by Docker Compose:

```
docker system prune -a
```

📌 Summary

- FastAPI serves the API on port **8000**.
- NGINX acts as a reverse proxy on port **80**.
- Docker Compose manages both containers.
- Swagger UI is accessible at `/docs`.

Step 2: Create the FastAPI Application

Inside `app/`, create `main.py`:

```python
from fastapi import FastAPI

app = FastAPI()

@app.get("/")
def read_root():
    return {"message": "Hello from FastAPI!"}

@app.get("/items/{item_id}")
def read_item(item_id: int, q: str = None):
    return {"item_id": item_id, "query": q}
```

Step 3: Setup NGINX Reverse Proxy

Create an NGINX configuration file `nginx.conf` inside the `nginx/` directory.

```
worker_processes auto;

events {
    worker_connections 1024;
}

http {
    server {
        listen 80;
        server_name localhost;

        location / {
            proxy_pass http://fastapi_app: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;
        }
    }
}
```

🚀 Why This Fix Works?

The `server {}` block must be inside the `http {}` block.

`proxy_pass` should match the FastAPI service name (`fastapi_app in docker-compose.yml`).

Defines `worker_processes` and `events`, which NGINX requires.

Step 4: Create the Dockerfile

Inside `app/`, create `Dockerfile`:

```
# Use official Python image
FROM python:3.9

# Set working directory
WORKDIR /app

# Copy FastAPI app files
COPY . .

# Install dependencies
RUN pip install --no-cache-dir fastapi uvicorn

# Expose FastAPI port
EXPOSE 8000

# Run FastAPI server
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]
```

Step 5: Create `docker-compose.yml`

Create a `docker-compose.yml` file to define the services.

```
version: '3.8'

version: '3.8' # Use Docker Compose version 3.8

services:
  fastapi:
    build: ./app # Build FastAPI from the app directory
    container_name: fastapi_app # Name the container
    ports:
      - "8000:8000" # Expose FastAPI on port 8000

  nginx:
    image: nginx:latest # Use the latest NGINX image
    container_name: nginx_proxy # Name the NGINX container
    ports:
      - "80:80" # Expose NGINX on port 80
    volumes:
      - ./nginx/nginx.conf:/etc/nginx/nginx.conf:ro # Mount the NGINX
```

```
configuration file
    depends_on:
      - fastapi # Ensure NGINX starts after FastAPI
```

🚀 Next Steps

Here are some ideas for expanding this project:

- **Implement Authentication**: Add JWT or OAuth2-based authentication.
- **Database Integration**: Connect FastAPI to PostgreSQL, MySQL, or MongoDB.
- **Logging & Monitoring**: Use Prometheus, Grafana, or ELK Stack.
- **Deploy to the Cloud**: Deploy the setup to AWS, Azure, or DigitalOcean.
- **Load Balancing & Scaling**: Run multiple instances of FastAPI and use NGINX for load balancing.
- **Automate Deployment**: Use CI/CD tools like GitHub Actions or Jenkins.

👨‍💻 Author

T S Sundar Raj

🙏 Thank You!

Happy coding! 🎉