RapidReach Application Deployment Guide

This guide will help you deploy the RapidReach application using Docker and Watchtower for automatic updates.

Prerequisites

1. Docker Desktop installed on your machine
2. Docker Hub account
3. Access to the application repository
4. Required credentials (MongoDB, etc.)

Initial Setup

1. Clone the Repository

```
git clone git@github.com:Tanmay-Patel2004/RapidReach.git
cd rapidreach
```

2. Environment Setup

1. Copy the example environment file:

```
cp backend/.env.example backend/.env
```

2. Edit the `.env` file with your credentials:

```
MONGODB_USER=your_mongodb_user
MONGODB_PASSWORD=your_mongodb_password
MONGODB_HOST=your_mongodb_host
NODE_ENV=production
```

3. Build and Push Docker Images

1. Login to Docker Hub:

```
docker login
```

2. Build and tag the images:

```
# Frontend
docker build -t your-dockerhub-username/rapidreach-frontend:latest ./Frontend
docker push your-dockerhub-username/rapidreach-frontend:latest

# Backend
docker build -t your-dockerhub-username/rapidreach-backend:latest ./backend
docker push your-dockerhub-username/rapidreach-backend:latest
```

Deployment

Local Development

1. Start the application:

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

2. Access the application:

- Frontend: http://localhost:5173
- Backend: http://localhost:5000

Production Deployment

1. Create a production docker-compose file:

```
cp docker-compose.yml docker-compose.prod.yml
```

2. Edit `docker-compose.prod.yml` to use your Docker Hub images:

```
version: '3.8'

services:
  frontend:
    image: your-dockerhub-username/rapidreach-frontend:latest
    ports:
      - "5173:80"
    environment:
      - VITE_API_URL=http://localhost:5000
    depends_on:
      - backend
    restart: unless-stopped

  backend:
    image: your-dockerhub-username/rapidreach-backend:latest
    ports:
      - "5000:5000"
    env_file:
      - ./backend/.env
    restart: unless-stopped

  watchtower:
    image: containrrr/watchtower
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
```

```
    command: --interval 30
    restart: unless-stopped
```

3. Start the production deployment:

```
docker-compose -f docker-compose.prod.yml up -d
```

Automatic Updates

The application uses Watchtower to automatically detect and apply updates:

1. Watchtower checks for new images every 30 seconds
2. When a new image is detected, it automatically:
    - Pulls the new image
    - Stops the existing container
    - Starts a new container with the updated image

Maintenance Commands

```
# View logs
docker-compose -f docker-compose.prod.yml logs -f

# Stop the application
docker-compose -f docker-compose.prod.yml down

# Restart the application
docker-compose -f docker-compose.prod.yml restart

# Update images manually
docker-compose -f docker-compose.prod.yml pull
docker-compose -f docker-compose.prod.yml up -d
```

Troubleshooting

1. If containers fail to start:
    - Check logs: `docker-compose -f docker-compose.prod.yml logs`

- Verify environment variables
- Ensure ports are not in use

2. If updates are not applying:
    - Check Watchtower logs: `docker logs watchtower`
    - Verify Docker Hub credentials
    - Ensure network connectivity
3. For database issues:
    - Verify MongoDB credentials in .env file
    - Check MongoDB connection
    - Verify network access to MongoDB host

Security Notes

1. Keep your .env file secure and never commit it to version control
2. Regularly update your Docker images for security patches
3. Use strong passwords for all credentials
4. Consider using Docker secrets for sensitive data in production

Support

For any deployment issues or questions, please contact the development team.