Rehobath Marine Service — AWS Docker Deployment Guide

This document explains the COMPLETE deployment process of the RMS (Rehobath Marine Service) MERN application into AWS EC2 using Docker and Docker Compose.

It includes: - Server preparation - Docker installation - Application architecture - File explanations - Build & run workflow - Troubleshooting

---

1. Project Architecture

The project contains two main parts:

Frontend (Vite + React) → Served via Nginx Backend (Node.js + Express API) Database (MongoDB container)

Container Architecture

Browser → Nginx (Frontend Container) → Backend API → MongoDB

---

2. Launch EC2 Server

Create Ubuntu EC2 instance and open ports in Security Group:

Port	Purpose
22	SSH
80	Website
5000	API (testing)

Connect:

ssh -i key.pem ubuntu@EC2_PUBLIC_IP

---

3. Install Docker (Official Repository)

sudo apt update sudo apt install ca-certificates curl gnupg lsb-release -y

sudo mkdir -p /etc/apt/keyrings curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt update sudo apt install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin -y

sudo usermod -aG docker ubuntu newgrp docker

Verify:

docker --version docker compose version

---

4. Clone Project

git clone https://github.com/sachinrokith/Rehobath-Marine-Deploy.git cd Rehobath-Marine-Deploy/RMS-main

---

5. Backend Environment Variables

Create file:

nano backend/.env

PORT=5000 MONGODB_URI=mongodb://mongo:27017/rms_db JWT_SECRET=production_secret_key

Explanation: - mongo = docker service name (internal DNS inside docker network)

---

6. Backend Dockerfile

Location: backend/Dockerfile

Full file content:

```
FROM node:18

WORKDIR /app

COPY package*.json ./
RUN npm install --legacy-peer-deps

COPY . .
```

```
EXPOSE 5000
CMD ["npm","start"]
```

Explanation: - FROM node:18 → official Node runtime - WORKDIR /app → container working directory - COPY package*.json → copy dependency list first (docker cache optimization) - RUN npm install → install backend dependencies - COPY . . → copy remaining source code - EXPOSE 5000 → API runs on port 5000 - CMD → starts express server

---

7. Frontend Dockerfile

Location: RMS-main/Dockerfile

Full file content:

```
FROM node:18 AS build

WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build

FROM nginx:alpine
COPY --from=build /app/dist /usr/share/nginx/html
COPY nginx.conf /etc/nginx/conf.d/default.conf

EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

Explanation: Stage 1 (build stage): - builds React Vite production files into /dist

Stage 2 (production stage): - nginx serves static website

---

8. Nginx Configuration

File: nginx.conf

Full content:

```
server {
    listen 80;

    location / {
        root /usr/share/nginx/html;
        index index.html;
        try_files $uri /index.html;
    }
}
```

Explanation: - listen 80 → website accessible on browser - root → where built React files are stored - try_files → very important for React routing

Without this → refresh on /about gives 404

---

9. Backend Environment Variables (.env)

File: backend/.env

Full content:

```
PORT=5000
MONGODB_URI=mongodb://mongo:27017/rms_db
JWT_SECRET=production_secret_key
```

Explanation: - PORT → Express server port - mongo → docker service name used as hostname - rms_db → database name created automatically - JWT_SECRET → used for login authentication tokens

---

10. Docker Compose File Build and Run Application

From RMS-main directory:

docker compose up --build -d

What happens internally: 1. Pull mongo image 2. Build backend image 3. Build frontend image 4. Create network 5. Start containers in dependency order

---

11. Verify Deployment

Check running containers:

docker ps

Expected: frontend → port 80 backend → port 5000 mongo → internal only

Open browser:

http://EC2_PUBLIC_IP

---

12. Common Errors & Fixes

Port 80 already in use

```
sudo systemctl stop nginx sudo systemctl disable nginx
```

Docker permission denied

```
sudo usermod -aG docker ubuntu newgrp docker
```

Containers crash

```
docker compose logs
```

Cannot connect frontend to backend

Use service name instead of localhost inside frontend API URL

---

13. Deployment Workflow Summary

Developer pushes code → Server pulls code → Docker rebuilds images → Containers restart → Website updated

---

Deployment completed successfully.