Fuel Price Tracker - Phase 5: Deployment & Hosting

1. Introduction

In Phase 5, you'll make your Fuel Price Tracker available online by deploying both the frontend and backend. You?ll also set up your database and scheduler on hosted platforms.

2. What You?ll Deploy

- React frontend (map + UI)
- Node.js backend (API + scraper + scheduler)
- MongoDB database (user data, station data)
- Cron jobs to refresh prices and send alerts

3. Tools You'll Use

- Vercel (for frontend)
- Render or Railway (for backend)
- MongoDB Atlas (for cloud database)
- cron-job.org or built-in schedulers
- GitHub (for version control)

4. Frontend Deployment (Vercel)

1. Push your React project to GitHub
2. Go to https://vercel.com and connect your repo
3. Set environment variables if needed
4. Vercel will auto-build and host your app
5. Customize domain, deploy settings in dashboard

5. Backend Deployment (Render)

1. Push backend code to GitHub
2. Go to https://render.com and create a new web service
3. Set your MongoDB URI and other secrets as env variables
4. Add start command: `node server.js` or your entry point

Fuel Price Tracker - Phase 5: Deployment & Hosting

5. Expose port 3001 or similar in Express

6. Database Hosting (MongoDB Atlas)

1. Go to https://cloud.mongodb.com
2. Create a free cluster
3. Whitelist all IPs (0.0.0.0/0) for dev
4. Create DB + user
5. Get the connection string (MONGO_URI)
6. Plug it into backend and test connection

7. Scheduled Jobs

- If Render: Use Render?s built-in scheduler
- Else use cron-job.org (free):
  - Point to a webhook in your backend that triggers scraping
  - Example: `https://your-api/render.com/scrape-now`

8. Final Checklist

- Frontend live on Vercel ?
- Backend live and connected ?
- Database in the cloud ?
- Scraper works in cloud ?
- Scheduler tested ?

9. Bonus Tips

- Set up logging with `winston` or similar
- Add global error handling in Express
- Monitor app health with UptimeRobot or Cronitor