# NYC FHV Driver Dashboard (NYCAD-sndx)

## Project Description

**Short description (GitHub About section):** > NYC FHV Driver Dashboard — A full-stack web app for visualizing New York City’s For-Hire Vehicle driver data with interactive charts, search, and daily data sync from NYC Open Data APIs.

**Long description:** The NYC FHV Driver Dashboard is a full-stack web application that allows users to explore and analyze NYC For-Hire Vehicle (FHV) driver data. It fetches driver data from NYC Open Data APIs, stores it in a PostgreSQL database, and provides an interactive dashboard with search and statistics.

**Key Features:** - Interactive dashboard showing driver distribution by borough  
- Advanced search for drivers by name, license, or borough  
- Automated daily data synchronization  
- REST API exposing driver data and statistics  
- Fully containerized deployment with Docker

**Tech Stack:** - Frontend: React (Vite) + Tailwind CSS  
- Backend: Node.js + Express  
- Database: PostgreSQL  
- Data Source: NYC Open Data API  
- Deployment: Docker & Docker Compose

## Folder Structure

NYCAD-sndx/  
│  
├─ backend/ # Node.js backend with Express  
│ ├─ src/  
│ ├─ package.json  
│ └─ .env.example  
│  
├─ frontend/ # React frontend with Vite  
│ ├─ src/  
│ ├─ package.json  
│ └─ .env.example  
│  
├─ docker-compose.yml  
└─ README.md

## Setup Instructions

### 1. Prerequisites

* Windows system
* Docker Desktop
* Node.js v20+ (if running locally without Docker)

### 2. Clone the repository

git clone https://github.com/Lokeshanthagiri/NYCAD-sndx.git  
cd NYCAD-sndx

### 3. Docker Setup (Recommended)

#### Step 3a — Check docker-compose.yml

Ensure services: postgres, backend, frontend. Ports: 5432, 4000, 5173.

#### Step 3b — Remove old containers

docker compose down -v

#### Step 3c — Build and start services

docker compose up --build

### 4. Running Backend

**Docker:**

docker compose up -d backend  
docker compose logs -f backend

**Local Node.js:**

cd backend  
npm install  
npm run dev

* Backend API URL: http://localhost:4000

### 5. Running Frontend

**Docker:**

docker compose up -d frontend  
docker compose logs -f frontend

**Local Node.js:**

cd frontend  
npm install  
npm run dev

* Dashboard URL: http://localhost:5173

### 6. Viewing the Graph

* Dashboard fetches data from backend endpoint: GET http://localhost:4000/stats
* Open browser at http://localhost:5173 to see the interactive graph.

### 7. Docker Commands Reference

* Start all containers: docker compose up --build
* Start single container: docker compose up -d backend
* Check logs: docker compose logs -f backend
* Stop containers: docker compose down
* Remove volumes: docker compose down -v

**Exec into backend container (interactive mode):**

docker exec -it <backend\_container\_name> sh

### 8. CI Pipeline (GitHub Actions)

Create .github/workflows/ci.yml:

name: NYCAD CI Pipeline  
  
on:  
 push:  
 branches: [ main ]  
 pull\_request:  
 branches: [ main ]  
  
jobs:  
 build:  
 runs-on: ubuntu-latest  
  
 services:  
 postgres:  
 image: postgres:15-alpine  
 env:  
 POSTGRES\_USER: postgres  
 POSTGRES\_PASSWORD: postgres  
 POSTGRES\_DB: fhv  
 ports:  
 - 5432:5432  
 options: >-  
 --health-cmd "pg\_isready -U postgres"  
 --health-interval 10s  
 --health-timeout 5s  
 --health-retries 5  
  
 steps:  
 - name: Checkout repository  
 uses: actions/checkout@v3  
  
 - name: Set up Node.js  
 uses: actions/setup-node@v3  
 with:  
 node-version: 20  
  
 - name: Install backend dependencies  
 working-directory: ./backend  
 run: npm install  
  
 - name: Run backend tests  
 working-directory: ./backend  
 run: |  
 echo "Add backend tests here"  
 # npm test  
  
 - name: Install frontend dependencies  
 working-directory: ./frontend  
 run: npm install  
  
 - name: Run frontend build  
 working-directory: ./frontend  
 run: npm run build  
  
 - name: Build Docker images  
 run: |  
 docker build -t nycad-backend ./backend  
 docker build -t nycad-frontend ./frontend

**How it works:**  
- Triggers on push or PR to main  
- Starts Postgres service for backend testing  
- Installs dependencies, runs backend tests, builds frontend  
- Builds Docker images

Optional: push images to Docker Hub with secrets.

*This documentation covers the project setup, running, and CI pipeline.*