DevOps Pipeline

1. **GitHub-hosted code**
2. **GitHub Actions** to build Docker images from that code (frontend + backend)
3. **Push those images to GitLab Container Registry**
4. **SSH into EC2 (via GitHub Actions)** and:
   * Pull those GitLab-hosted Docker images
   * Use docker-compose to run the full stack (MySQL + backend + frontend)

**STEP 1: GitHub Code (Source of Truth)**

Your GitHub repo contains both:

/frontend ← React code

/backend ← Node.js + Express

/docker-compose.yml

**STEP 2: GitHub Actions Builds Docker Images (CI)**

In your GitHub Actions workflow:

1. GitHub pulls the source code (from itself)
2. It **builds Docker images**:
   * frontend image from ./frontend/Dockerfile
   * backend image from ./backend/Dockerfile
3. Then it **pushes those images to GitLab Container Registry**:
   * registry.gitlab.com/<you>/incident-tracker-frontend
   * registry.gitlab.com/<you>/incident-tracker-backend

You’ll use a GitLab personal access token to push to GitLab.

**STEP 3: GitHub Actions Deploys to EC2 (CD)**

In the same GitHub Actions pipeline:

1. GitHub SSHes into your **EC2 instance**
2. EC2 navigates to your project folder (with docker-compose.yml)
3. It **logs in to GitLab Container Registry**
4. It **pulls the images** for:
   * frontend (from GitLab)
   * backend (from GitLab)
5. It runs:

docker compose pull

docker compose up -d

* This spins up **frontend + backend + MySQL containers together**.

**EC2 Just Runs Containers**

Your EC2 is not responsible for building anything.

It only:

* Pulls ready-to-run Docker images from GitLab
* Runs them using docker-compose

**In Summary:\**

GitHub repo (source code)

↓

GitHub Actions CI:

- Build Docker image (frontend + backend)

- Push to GitLab Container Registry

↓

GitHub Actions CD:

- SSH into EC2

- Pull Docker images from GitLab

- Run `docker compose up` to deploy app