Absolutely — here's a **complete breakdown** of what I'm helping you build, **why we're doing it this way**, and **how GitHub and GitLab interact** in your DevOps pipeline:

**🚧 Your Goal**

You want to:

✅ Host your **source code** in **GitHub**  
✅ Use **GitHub Actions** to build Docker images (CI)  
✅ Store those images in **GitLab Container Registry**  
✅ Deploy them to your **EC2 instance** using docker-compose (CD)

**📦 What I'm Doing in the Workflow**

Let me walk through the **whole DevOps flow**, step by step — with no detail skipped:

**🔹 Step 1: GitHub Source Code (Source of Truth)**

Your code lives in a GitHub repo with this structure:

bash

CopyEdit

IncidentTracker/

│

├── incident-tracker-client/ ← React frontend

│ └── Dockerfile

│

├── incident-tracker-server/ ← Node.js + Express backend

│ └── Dockerfile

│

└── .github/workflows/deploy.yml ← GitHub Actions workflow

**🔹 Step 2: GitHub Actions Builds Docker Images (CI)**

In your GitHub Actions pipeline:

1. GitHub detects a push to the main branch.
2. It **checks out your code** (gets access to frontend + backend folders).
3. It **logs into the GitLab Container Registry** using your GITLAB\_TOKEN.
4. It **builds Docker images**:
   * Backend → using ./incident-tracker-server/Dockerfile
   * Frontend → using ./incident-tracker-client/Dockerfile
5. It **pushes these images to GitLab**:
   * registry.gitlab.com/<your-name>/incidenttracker/incident-tracker-server:main
   * registry.gitlab.com/<your-name>/incidenttracker/incident-tracker-client:main

✅ **Why?** You want to separate code (GitHub) and image storage (GitLab), which is a **real-world DevOps pattern**.

**🔹 Step 3: SSH into EC2 and Deploy (CD)**

After the images are pushed to GitLab:

1. GitHub Actions uses appleboy/ssh-action to **SSH into your EC2 instance**.
2. On EC2, it navigates to your project directory (e.g., ~/incident-tracker).
3. It logs in to GitLab Registry (so EC2 can pull Docker images).
4. It runs:

bash

CopyEdit

docker compose pull # Pull latest backend + frontend images

docker compose down # Stop old containers

docker compose up -d # Start the new containers with updated code

✅ **Why?** EC2 shouldn't waste time or resources building Docker images. It just **runs** what GitHub already built.

**❌ Why NOT Build Docker Images in EC2?**

* **Slower & less secure** (EC2 should only run prebuilt, tested containers)
* **Breaks the CI/CD pipeline model** — you're bypassing CI (GitHub) and turning EC2 into a CI server
* **Wastes EC2 compute time and risks breaking builds**
* **Harder to manage and scale**

**🔄 How Does GitLab Know When GitHub Code Changes?**

Actually — **it doesn't**.

GitLab is not hosting your code in this setup. It’s only used to **store Docker images** in its **Container Registry**.

So the flow is:

* You push code → GitHub detects it → GitHub builds images → GitHub pushes those images to **GitLab's Container Registry**
* GitLab doesn't "watch" your repo — it's **passive** in this setup

✅ GitLab's only job: **store the Docker images**, like a private DockerHub.

**✅ Summary**

| **Step** | **What Happens** | **Who Does It** |
| --- | --- | --- |
| 1 | Code is pushed to GitHub | You |
| 2 | GitHub Actions builds Docker images | GitHub |
| 3 | GitHub Actions pushes images to GitLab Registry | GitHub |
| 4 | GitHub Actions SSHes into EC2 | GitHub |
| 5 | EC2 pulls images from GitLab and runs them | EC2 (via GitHub trigger) |

Would you like me to draw a simple architecture diagram or help you test this end-to-end?

Let me know if you want to go one level deeper in any of these steps.

**You said:**

It \*\*builds Docker images\*\*: \* Backend → using `./incident-tracker-server/Dockerfile` \* Frontend → using `./incident-tracker-client/Dockerfile`

so after u login to gitlab, from the gitlab repo it creates the docker image right. like we have to say get this gitlap repo and then make it to a registry right?)

**ChatGPT said:**

Ah — great follow-up! Let me clarify that part carefully:

**🔍 Your Assumption:**

“After you log into GitLab, it takes the code from the GitLab repo and creates the Docker image... right?”

❌ **Not in your current setup.**  
✅ The Docker image is built **inside GitHub Actions**, **from your GitHub repo code**, not from GitLab.

**✅ What Actually Happens in Your Workflow**

Let’s go line-by-line:

yaml

CopyEdit

docker build -t registry.gitlab.com/... ./incident-tracker-server

This command is saying:

“Take the code inside ./incident-tracker-server (which came from GitHub), and build a Docker image from that locally.”

Then:

yaml

CopyEdit

docker push registry.gitlab.com/.../incident-tracker-server:main

This pushes that image to the **GitLab Container Registry**.

💡 GitLab is **just a place to store the built image**, not the source code.

**🔁 So GitLab Doesn’t Pull Your Code?**

Correct — **GitLab doesn't pull your code at all** in this workflow.

**GitHub** is:

* The source of your application code.
* The place where Docker images are built.
* The orchestrator of deployment.

**GitLab** is:

* Only used to **store Docker images** in its Container Registry (like a warehouse for prebuilt containers).
* It does not build the images or clone any code.

**🚫 What You’re Not Using in This Setup**

You're **not** using:

* GitLab CI (.gitlab-ci.yml)
* GitLab source code
* GitLab build pipelines

You're only using:

* **GitHub-hosted code**
* **GitHub Actions as your CI/CD tool**
* **GitLab's registry as a container image storage system**

**✅ If You Wanted GitLab to Build Images (Alternative Setup)**

Then yes — you’d have to:

1. Import your code into a GitLab repository
2. Add a .gitlab-ci.yml file to define how GitLab should build and push Docker images
3. Enable GitLab CI/CD

But in **your setup**, GitHub builds, GitHub deploys — GitLab only stores.

Would you like to see what it would look like *if* GitLab were building images itself?

Or are you happy to keep GitHub as your pipeline and just use GitLab for registry storage?

**✅ What Actually Happens in Your Workflow**

Let’s go line-by-line:

yaml

CopyEdit

docker build -t registry.gitlab.com/... ./incident-tracker-server

This command is saying:

“Take the code inside ./incident-tracker-server (which came from GitHub), and build a Docker image from that locally.”

Then:

yaml

CopyEdit

docker push registry.gitlab.com/.../incident-tracker-server:main

This pushes that image to the **GitLab Container Registry**.

💡 GitLab is **just a place to store the built image**, not the source code.