Task 1: Automate Code Deployment Using CI/CD Pipeline

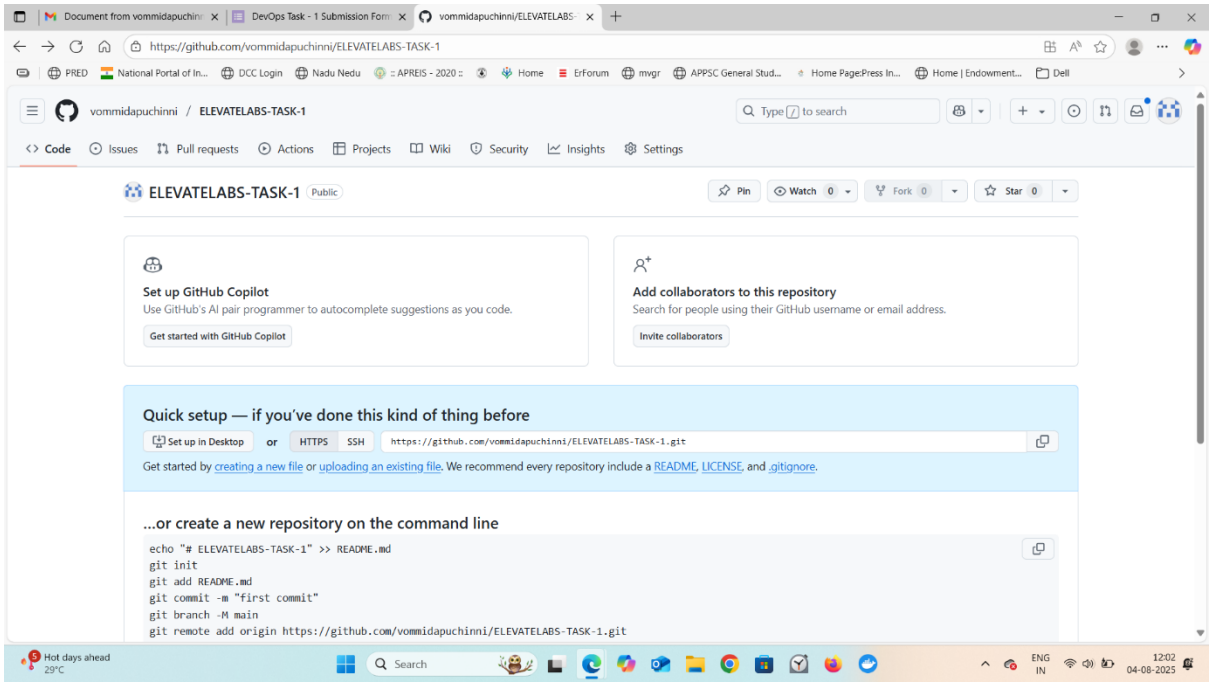
Objective: Set up a CI/CD pipeline using GitHub Actions to build and deploy a sample Node.js web app using DockerHub.

Tools Used

- GitHub → Version control and CI/CD triggering
- GitHub Actions → Workflow automation tool for CI/CD
- Node.js → Backend JavaScript runtime for the sample app
- Docker → Containerization of the application
- DockerHub → Hosting and storing Docker images

Project Setup:

Created GitHub Repository



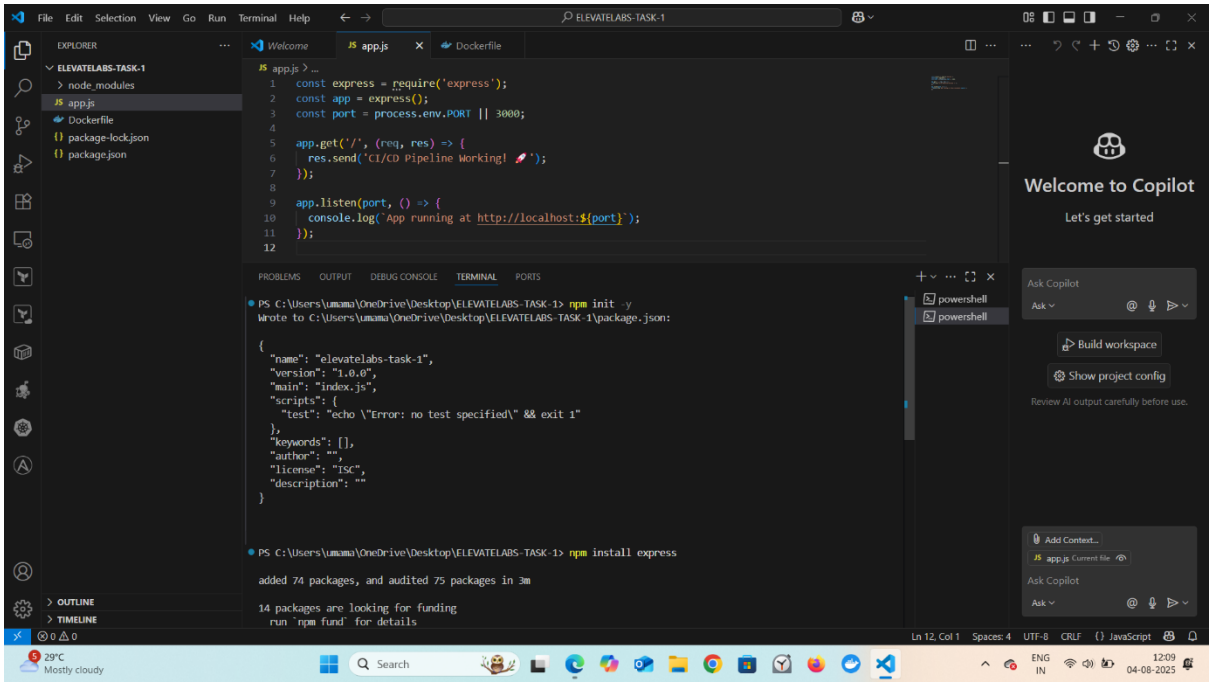
Install nodejs locally

Node.js App Initialization:

npm init -y

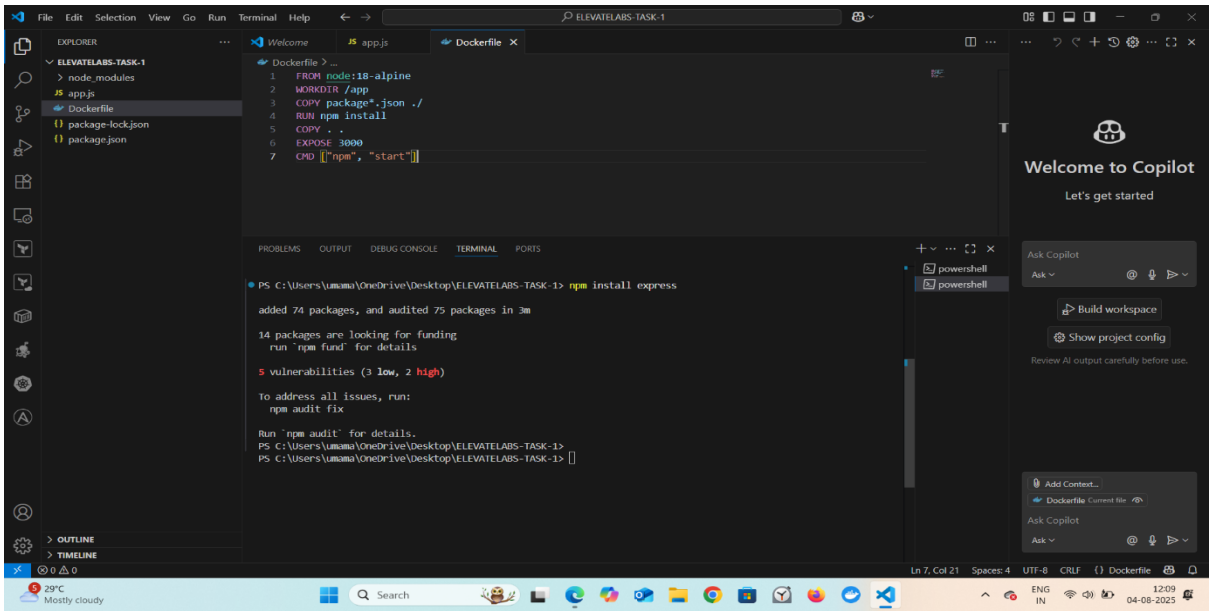
npm install express

app.js file is created



- It’s the main entry point for the Node.js application.
- Uses the **Express** module to create a web server.
- The server listens on **port 3000** (or an environment-defined port).
- Defines a route / that responds with: **"CI/CD Pipeline Working! 🚀 "**.
- Confirms the app is successfully running after deployment.
- Helpful to test if the CI/CD pipeline and Docker image deployment are working as expected.

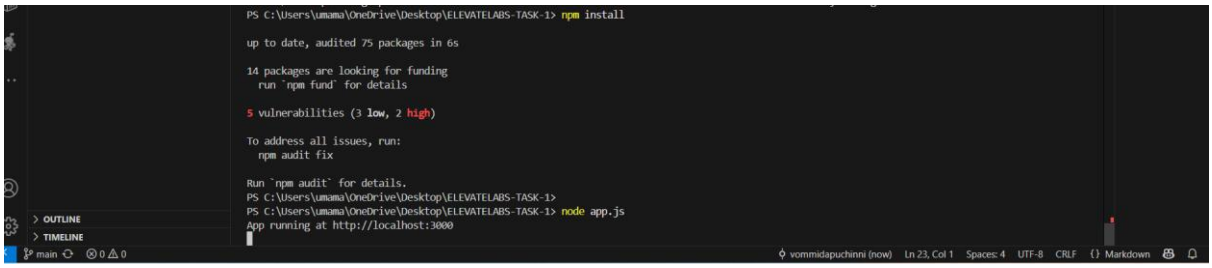
Docker file created.



- FROM node:14: Uses Node.js base image
- WORKDIR: Sets working directory
- COPY & RUN: Installs dependencies
- CMD: Starts the server

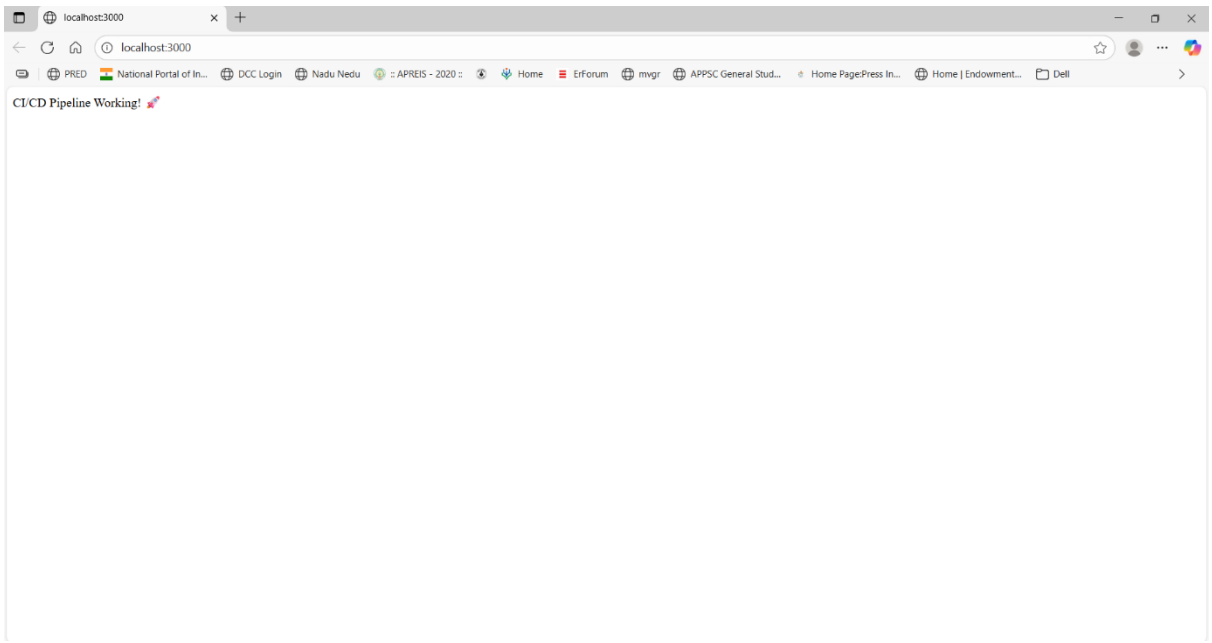
package.json (script added for testing)

```
"scripts": {
  "start": "node app.js",
  "test": "echo \"No tests yet\" && exit 0"
}
```

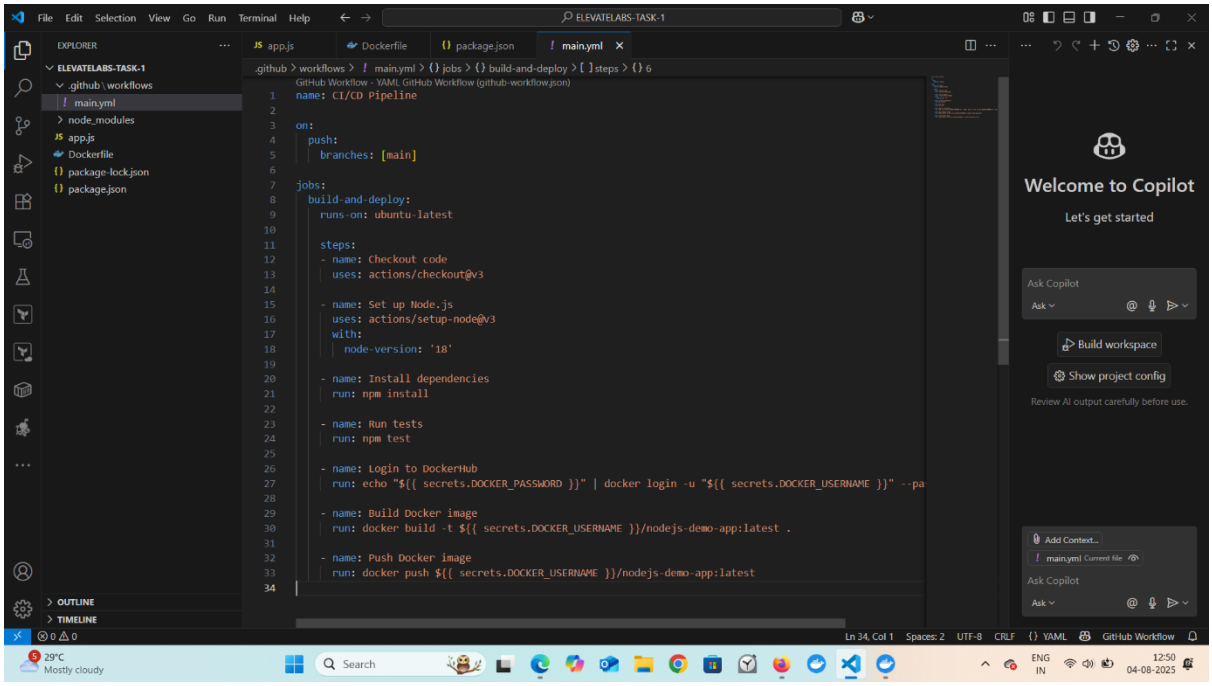


node app.js will start the server we can access the server by localhost:3000

to stop it we use ctrl+c



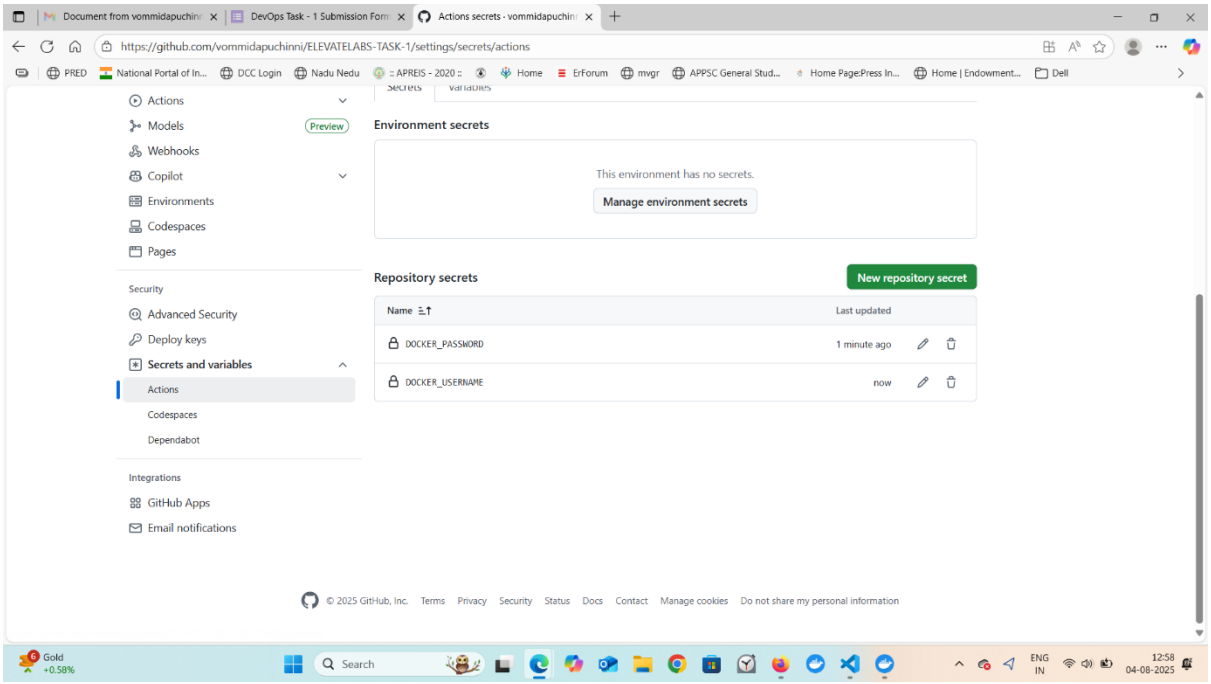
Create the CI/CD



This pipeline automatically builds, tests, and pushes Docker image to DockerHub on every push to main.

Secrets Used:

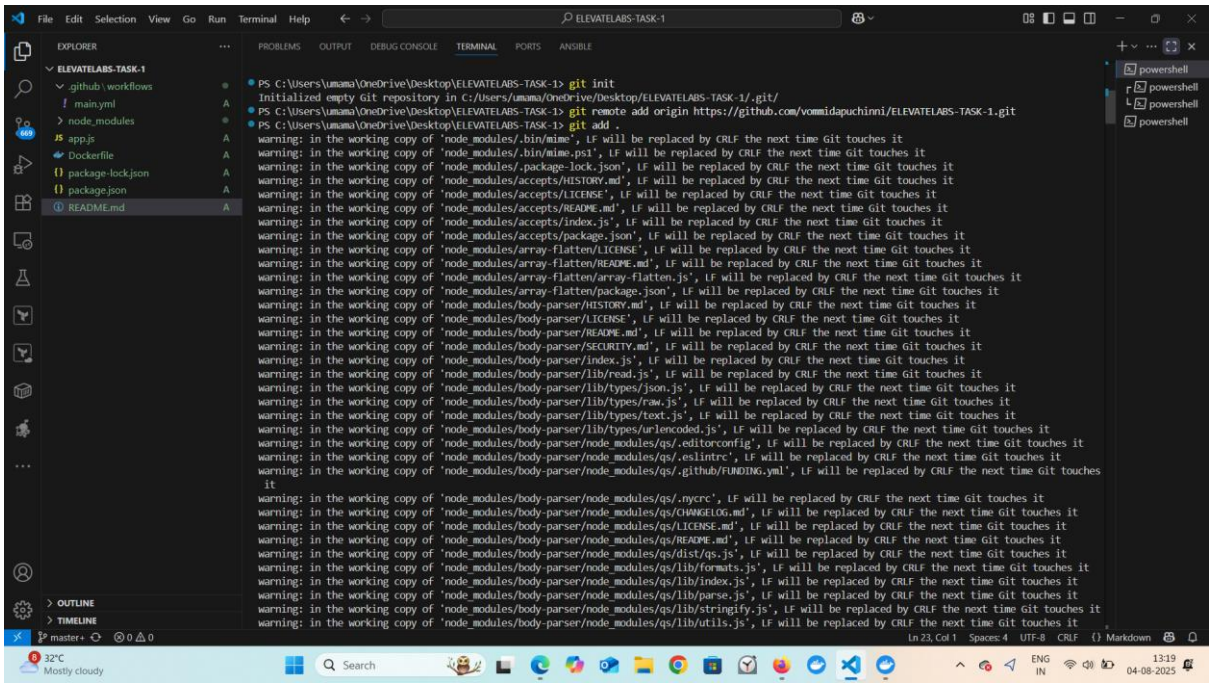
- DOCKER_USERNAME – My DockerHub username
- DOCKER_PASSWORD – My DockerHub password (stored securely in GitHub Secrets)



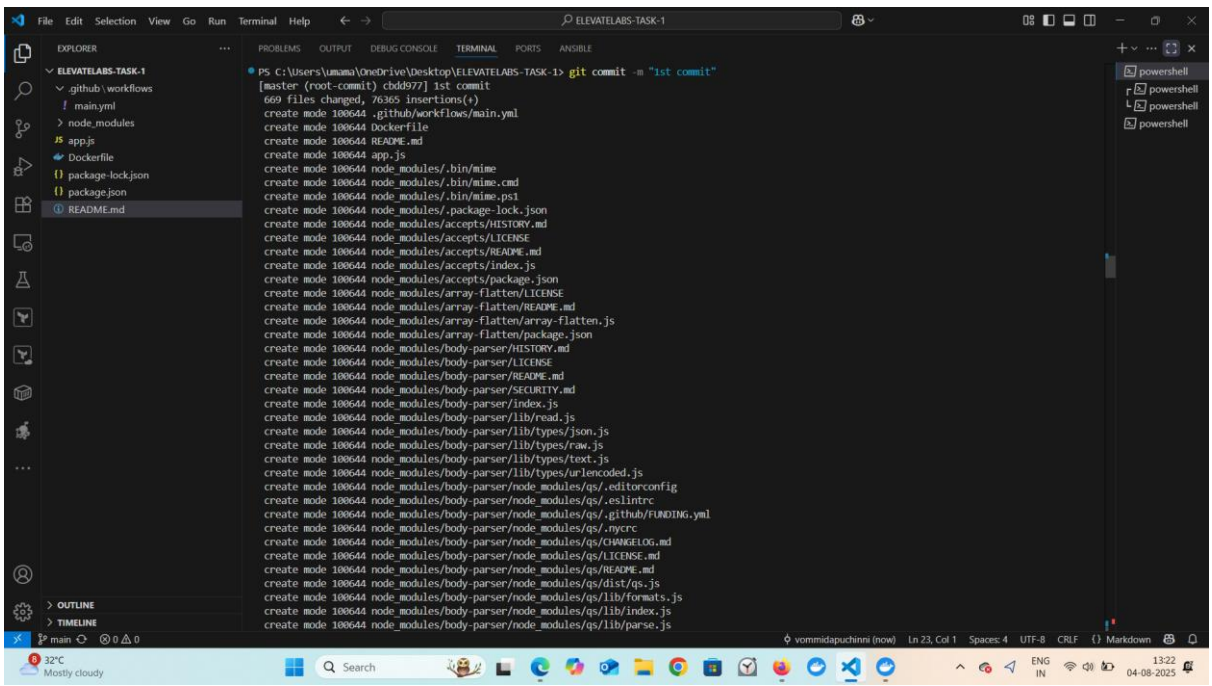
Push to GitHub

In terminal run these commands:

git init

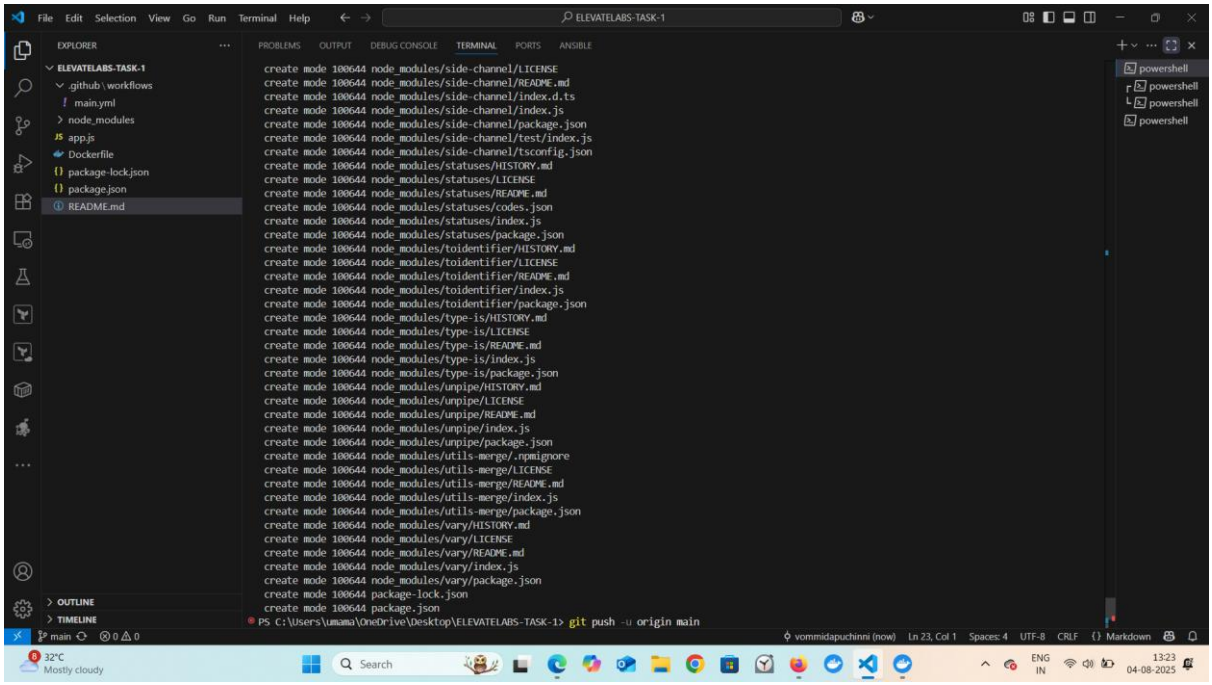


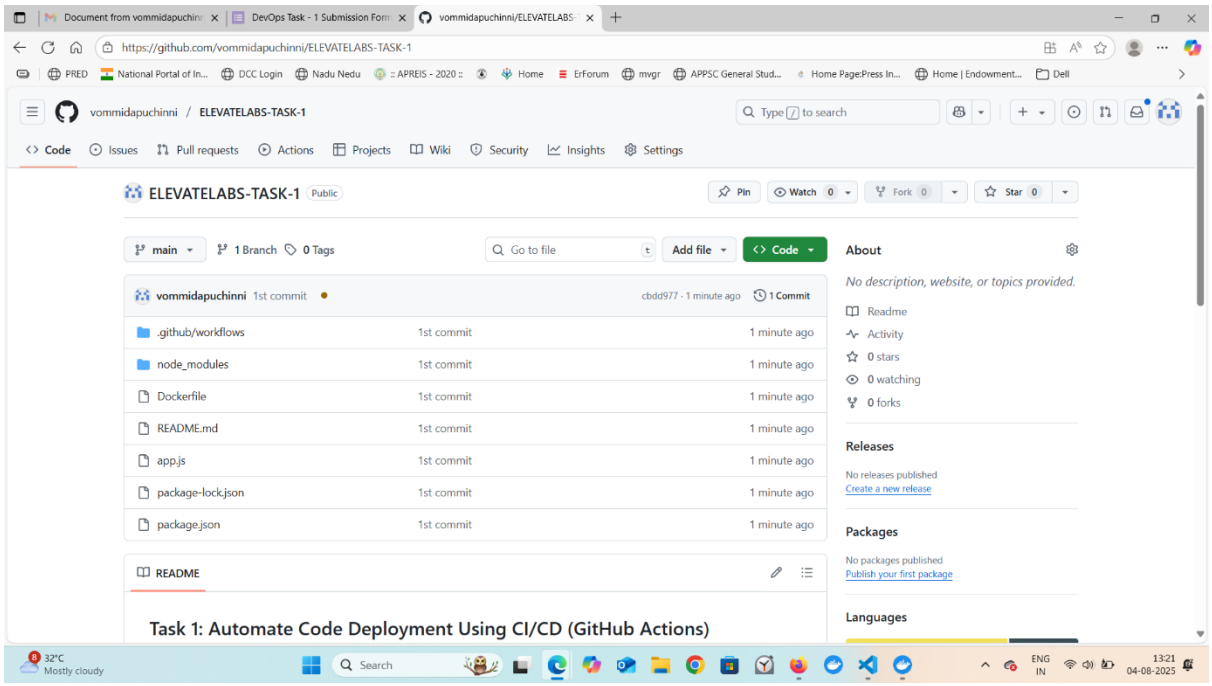
- git remote add origin https://github.com/vommidapuchinni/ELEVATELABS-TASK-1.git
- git add .
- git commit -m "Initial commit with CI/CD"



git branch -M main

git push -u origin main



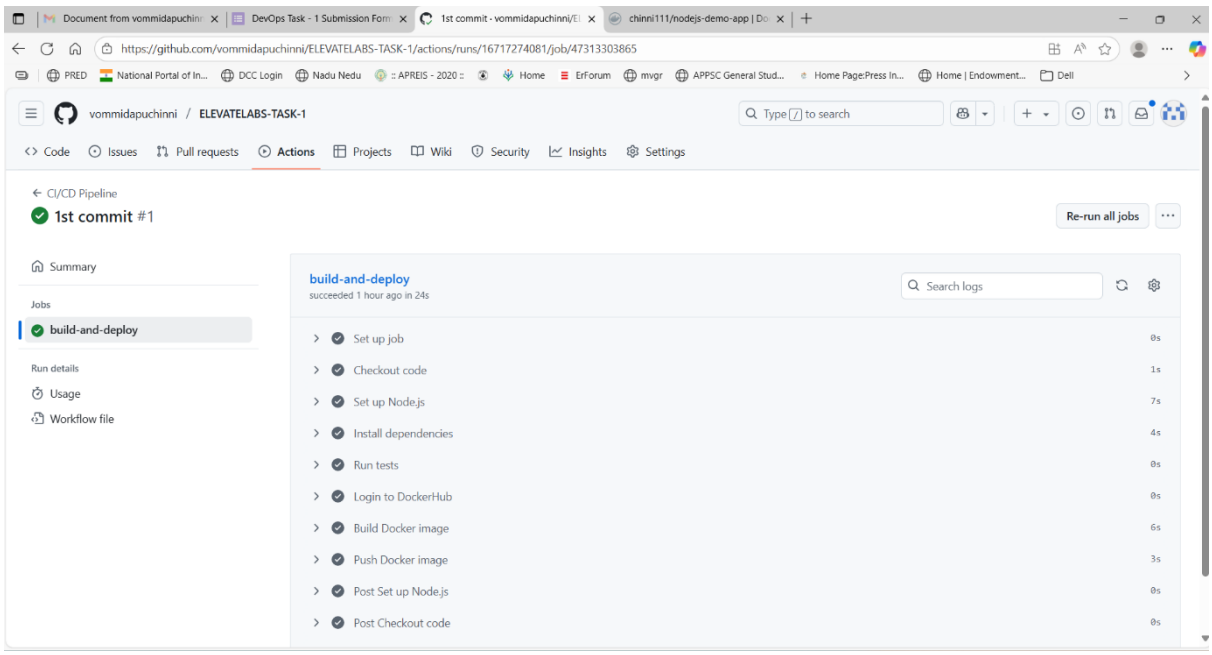


Project Setup Summary

- Created GitHub repository: nodejs-demo-app
- Wrote app.js to serve a simple route using Express
- Added Dockerfile to containerize the app

- Created GitHub Actions workflow .github/workflows/main.yml to:
    - Trigger on push to main
    - Build Docker image
    - Push to DockerHub

Successfully build and deployed



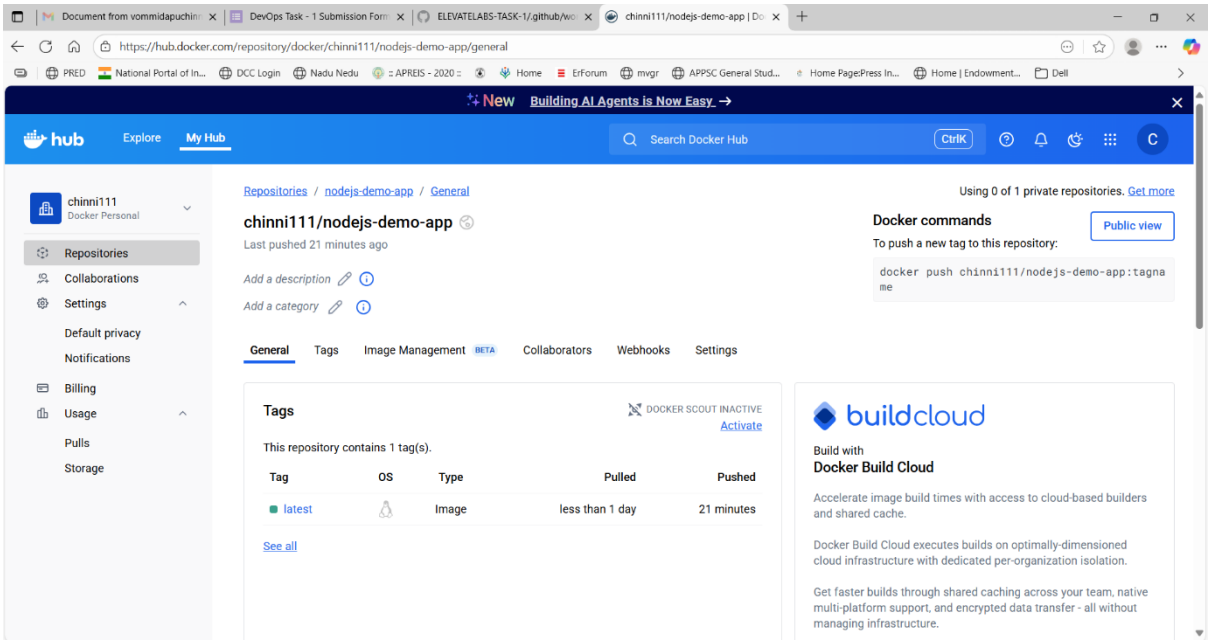
CI/CD Workflow Steps

- GitHub push triggers the pipeline
- Pipeline builds the Docker image
- Docker image is pushed to DockerHub
- Pulled the image manually using Docker CLI and verified app running on localhost:3000

DockerHub:

After successful GitHub Actions run:

- The image was pushed to my DockerHub: chinni111/nodejs-demo-app:latest



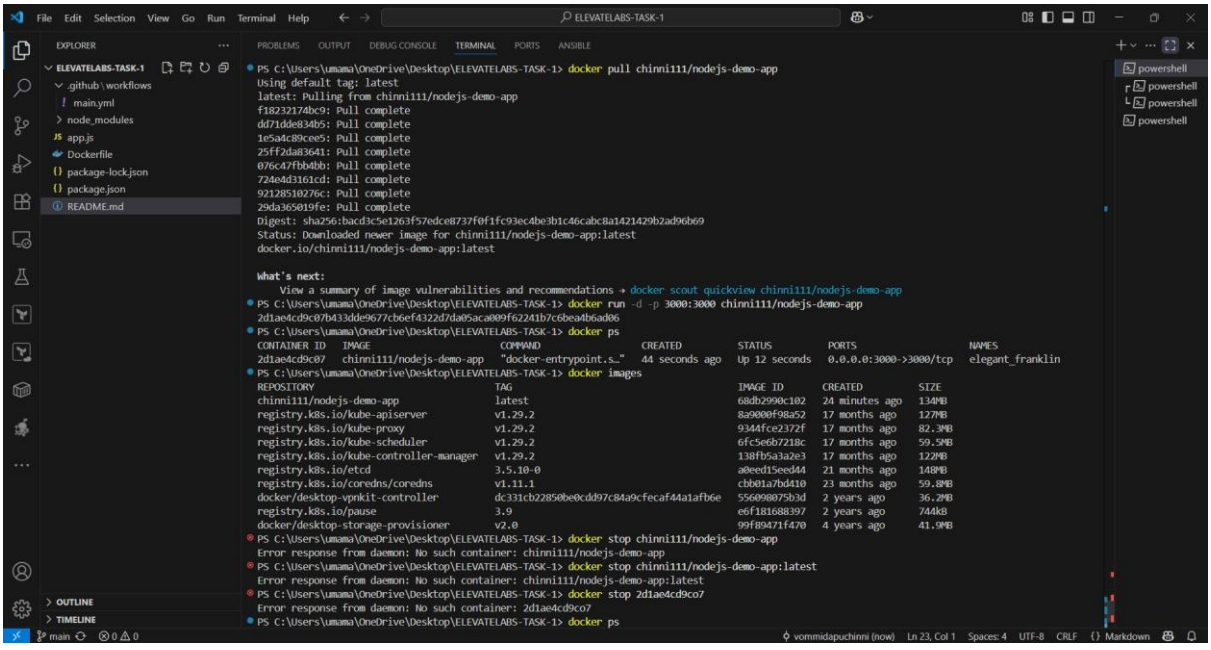
Running the App Locally:

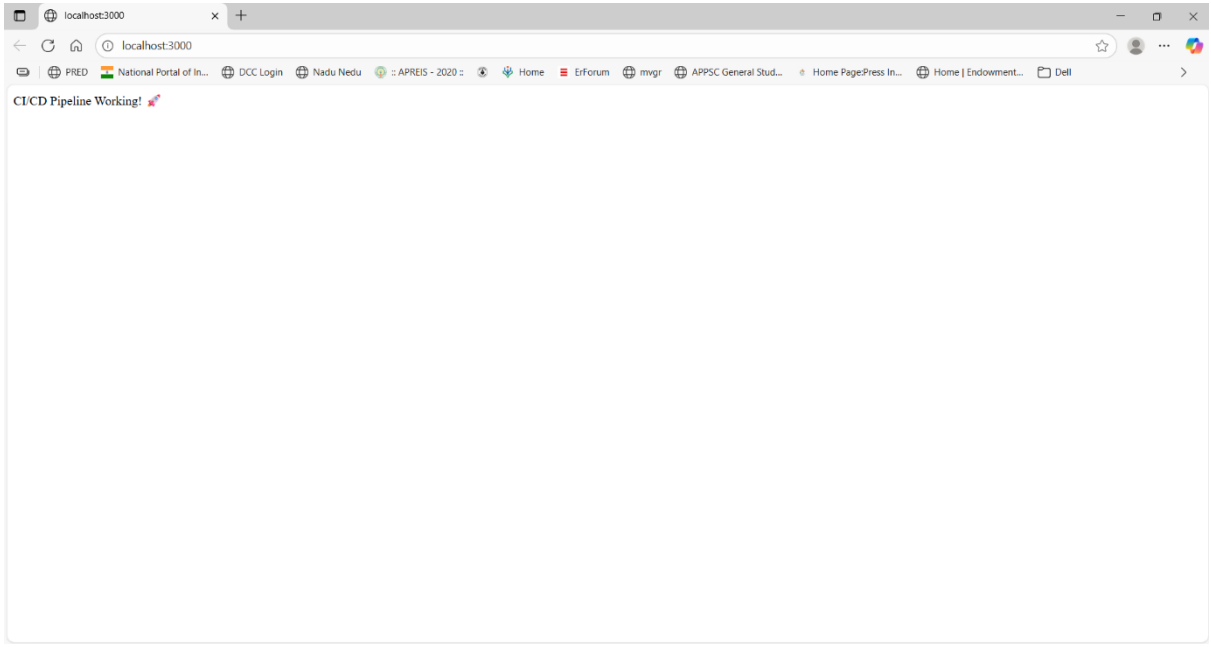
I did not build the image locally.
Instead, I pulled the image from DockerHub and ran it using:

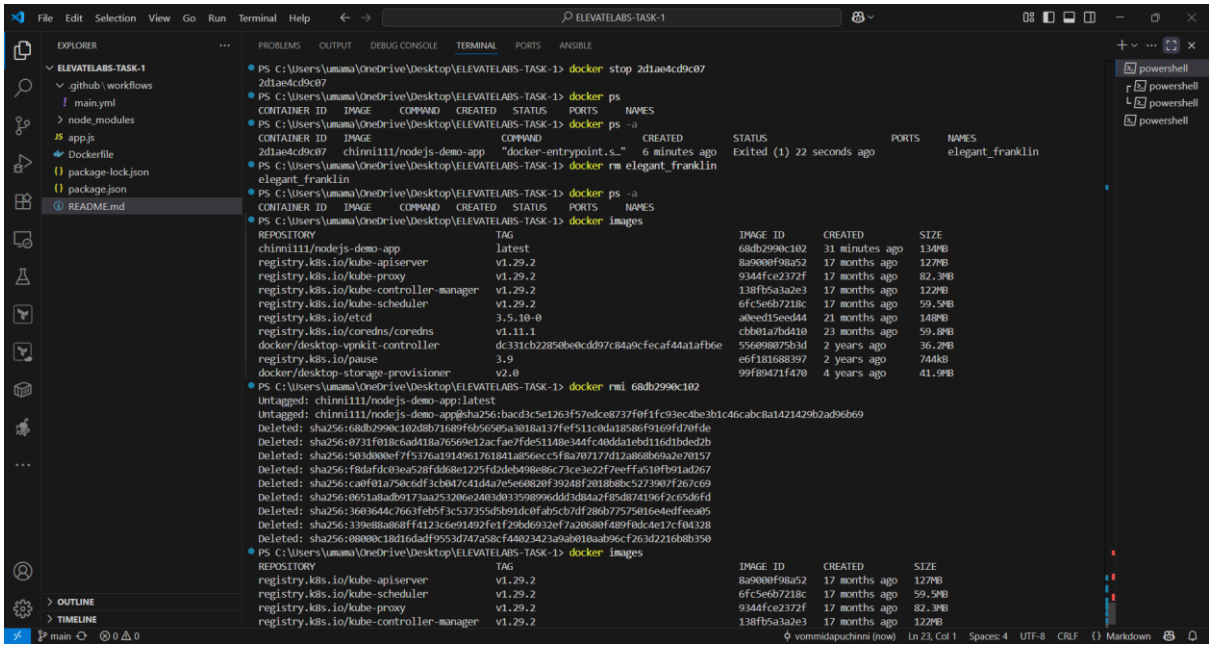
docker pull chinni111/nodejs-demo-app

docker run -d -p 3000:3000 chinni111/nodejs-demo-app

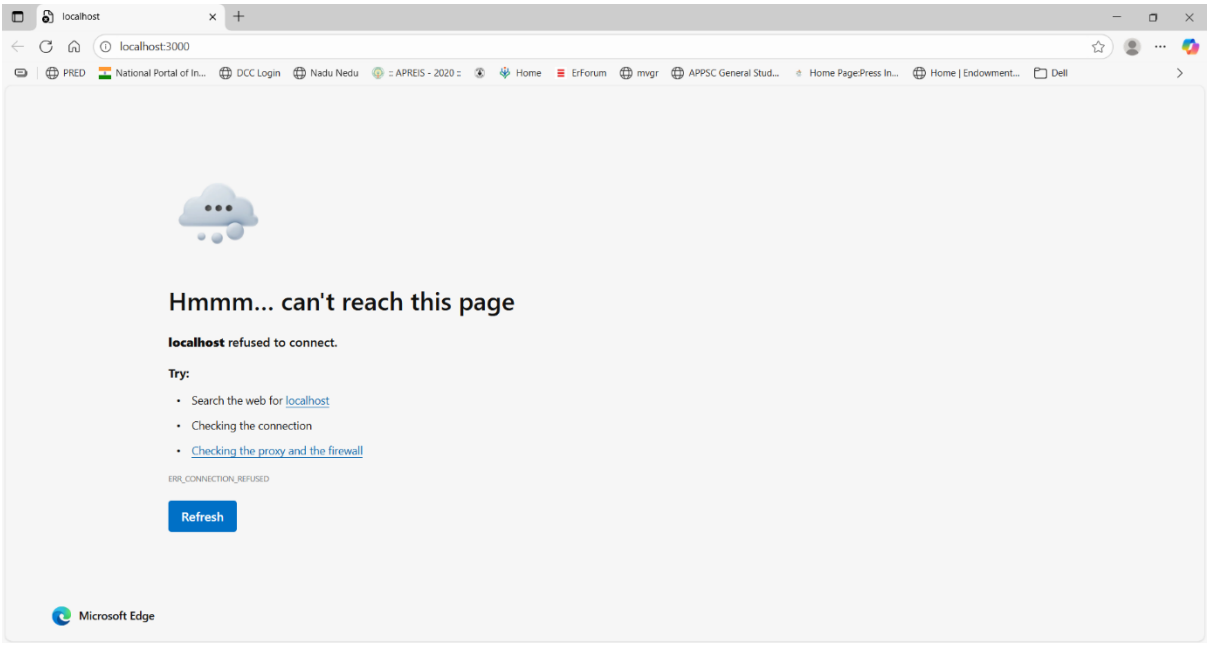
Then I accessed the app in my browser using:
http://localhost:3000







After container stopped, we cannot access the application.



Important Files

- app.js: Main Node.js server file
- Dockerfile: Instructions to build the Docker image
- main.yml: GitHub Actions workflow configuration

CONCLUSION: Through this task, I successfully implemented a complete CI/CD pipeline using GitHub Actions. I automated the process of testing, building, and deploying a Node.js application as a Docker image to DockerHub. Without building the image locally, I pulled it from DockerHub and ran it in a container, then accessed the running app through my local browser. This hands-on experience helped me understand practical DevOps workflows and the integration of version control, containerization, and automation.