**Containerize and Deploy a Next.js Application using Docker, GitHub Actions, and Minikube**

# **Quick overview (what you’ll create)**

1. Local Next.js app (devops-nextjs-app/)
2. Dockerfile and .dockerignore
3. k8s/ folder: deployment.yaml, service.yaml
4. GitHub repo (public) with .github/workflows/ci.yml that builds and pushes to GHCR
5. README.md and email submission message

**Techstack:**

Git installed (git --version)

Docker installed & running (docker --version)

Node.js (v18+) & npm (node --version && npm --version)

kubectl installed (kubectl version --client)

minikube installed (minikube version)

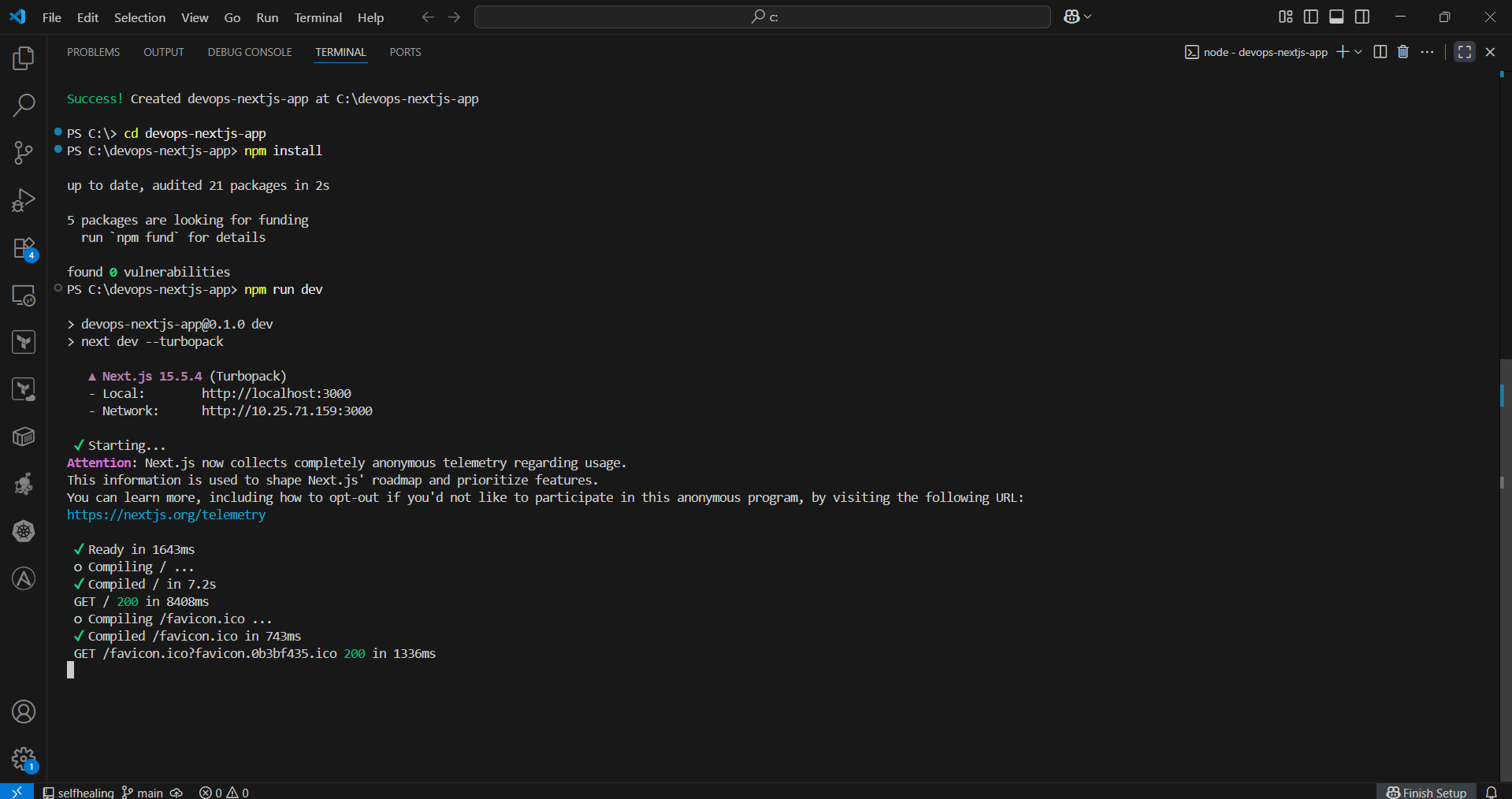
**STEP 1: Create Next.js app**

Open Terminal

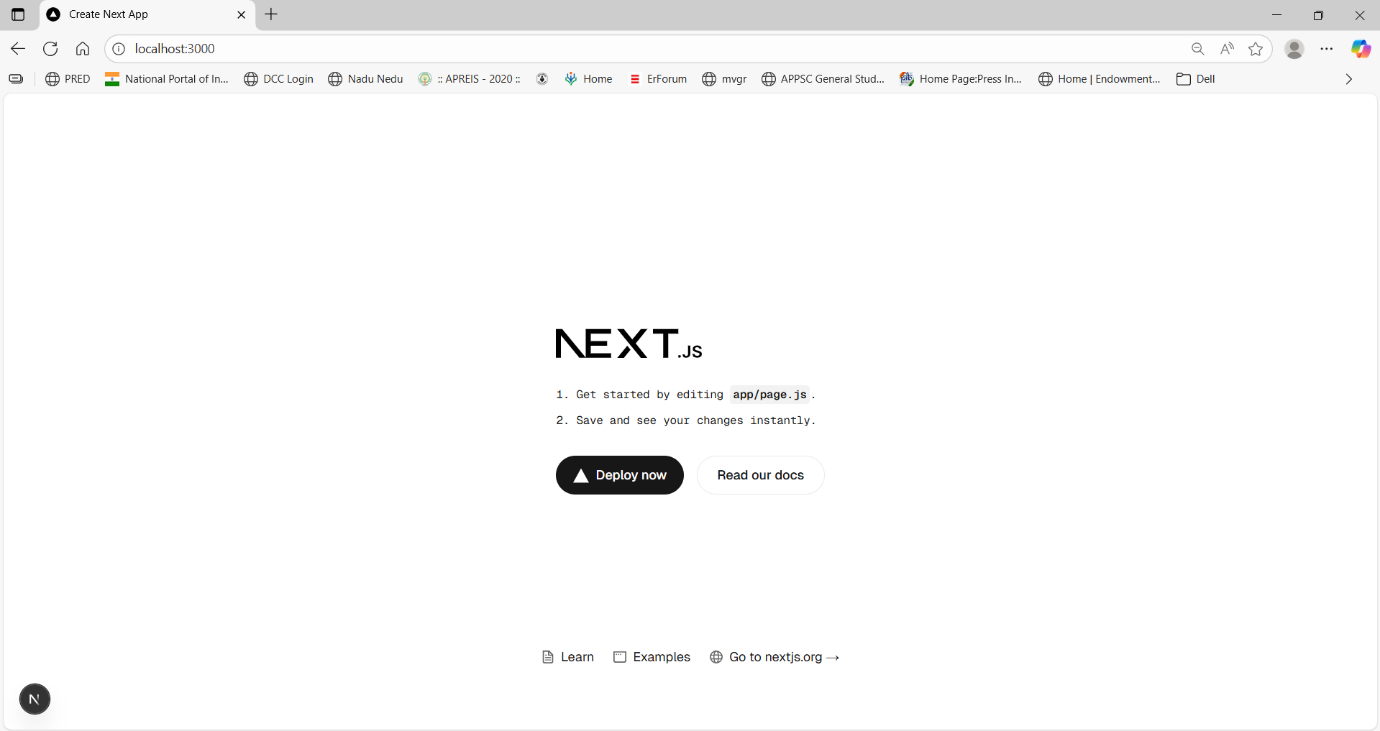
create project: npx create-next-app@latest devops-nextjs-app

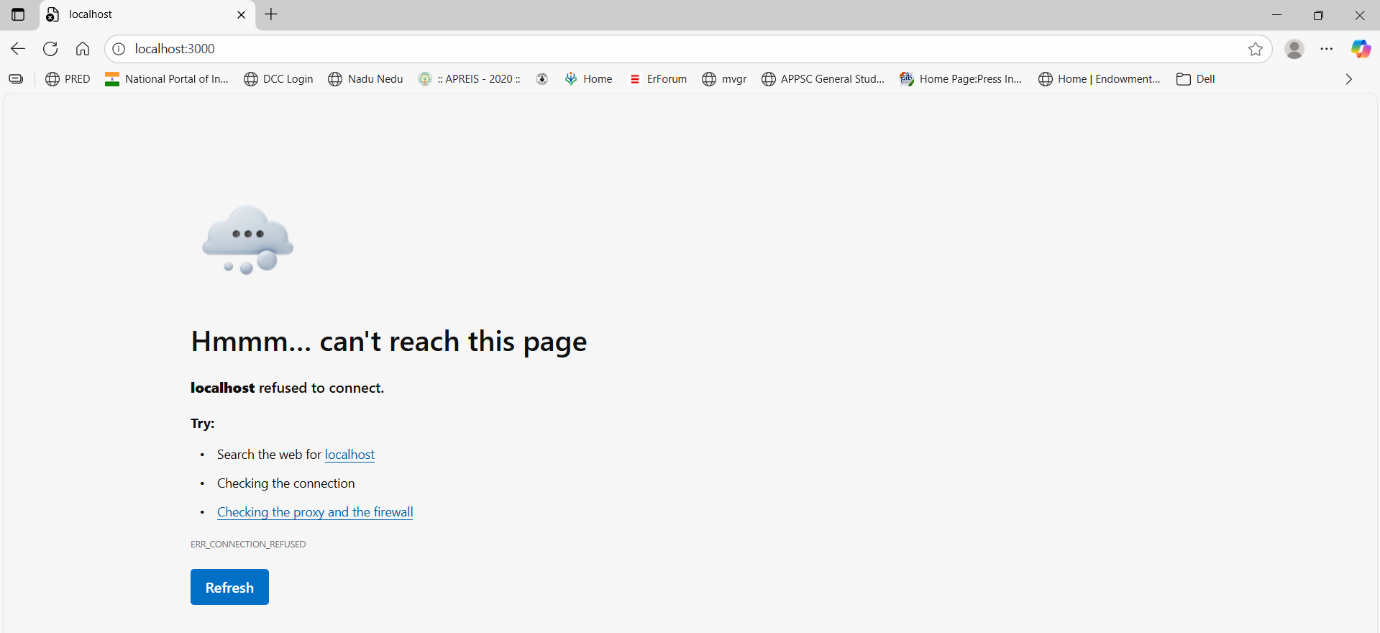
cd devops-nextjs-app

optionally test locally: npm install && npm run dev



open http://localhost:3000 in browser to verify





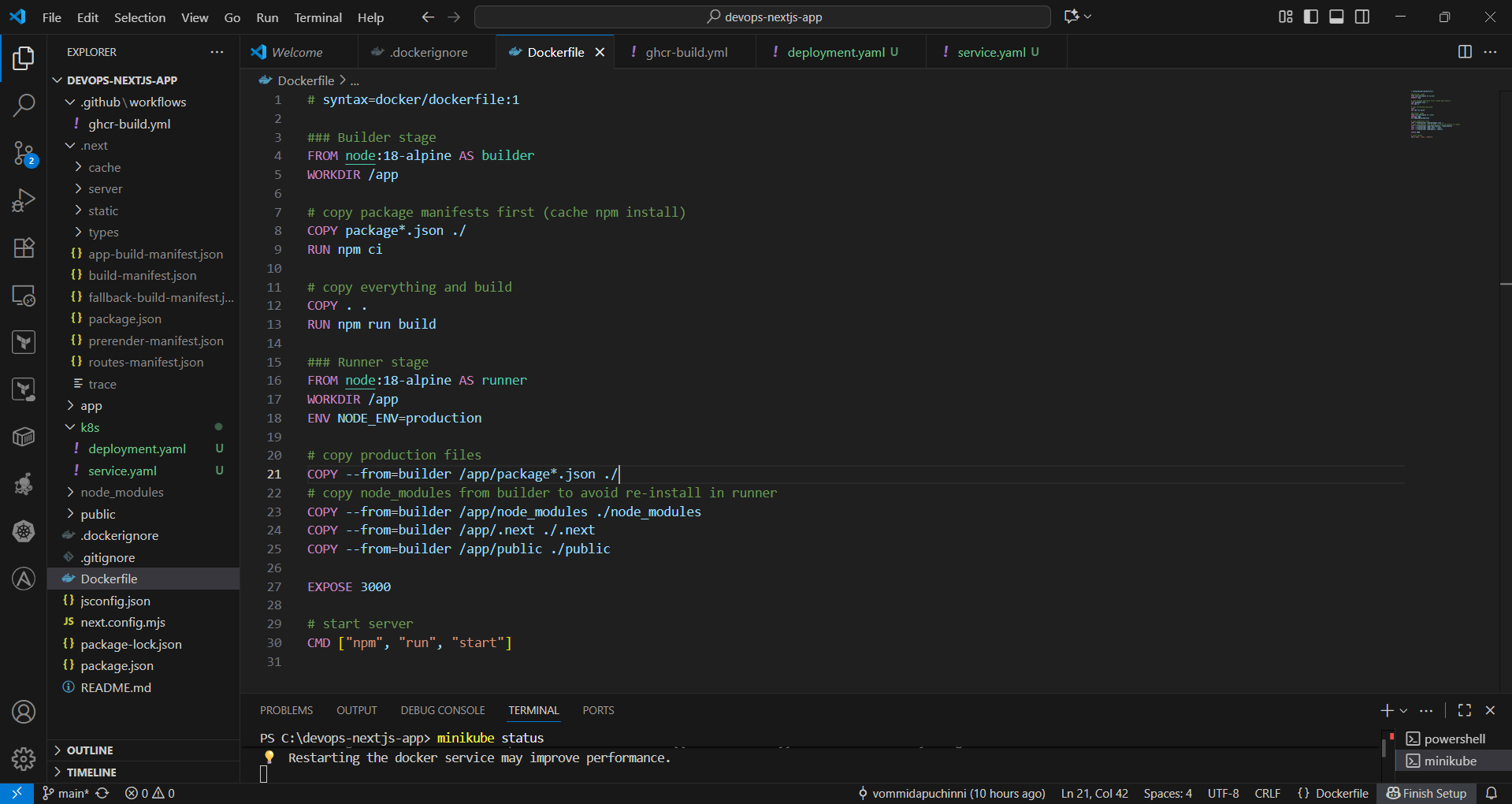
After stopping the npm run dev we cannot access the next app

STEP 2: Creating Docker file

### .dockerignore: to ignore unwanted files.

### Dockerfile (multi-stage — copy/paste)

Create Dockerfile

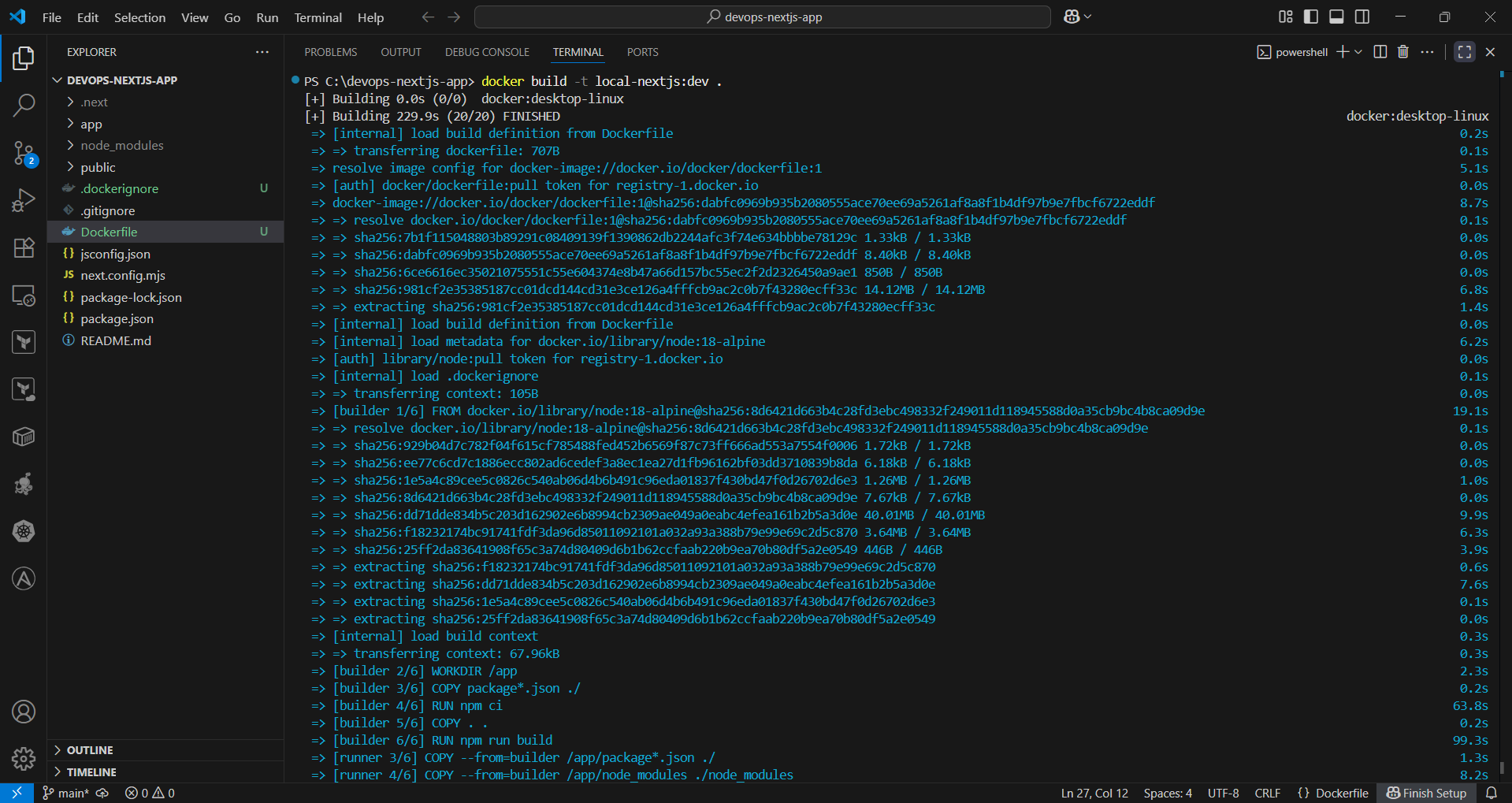


This is a simple, reliable Dockerfile. For smaller runtime images, you can use Next.js output: 'standalone' and adjust Dockerfile (advanced optimization).

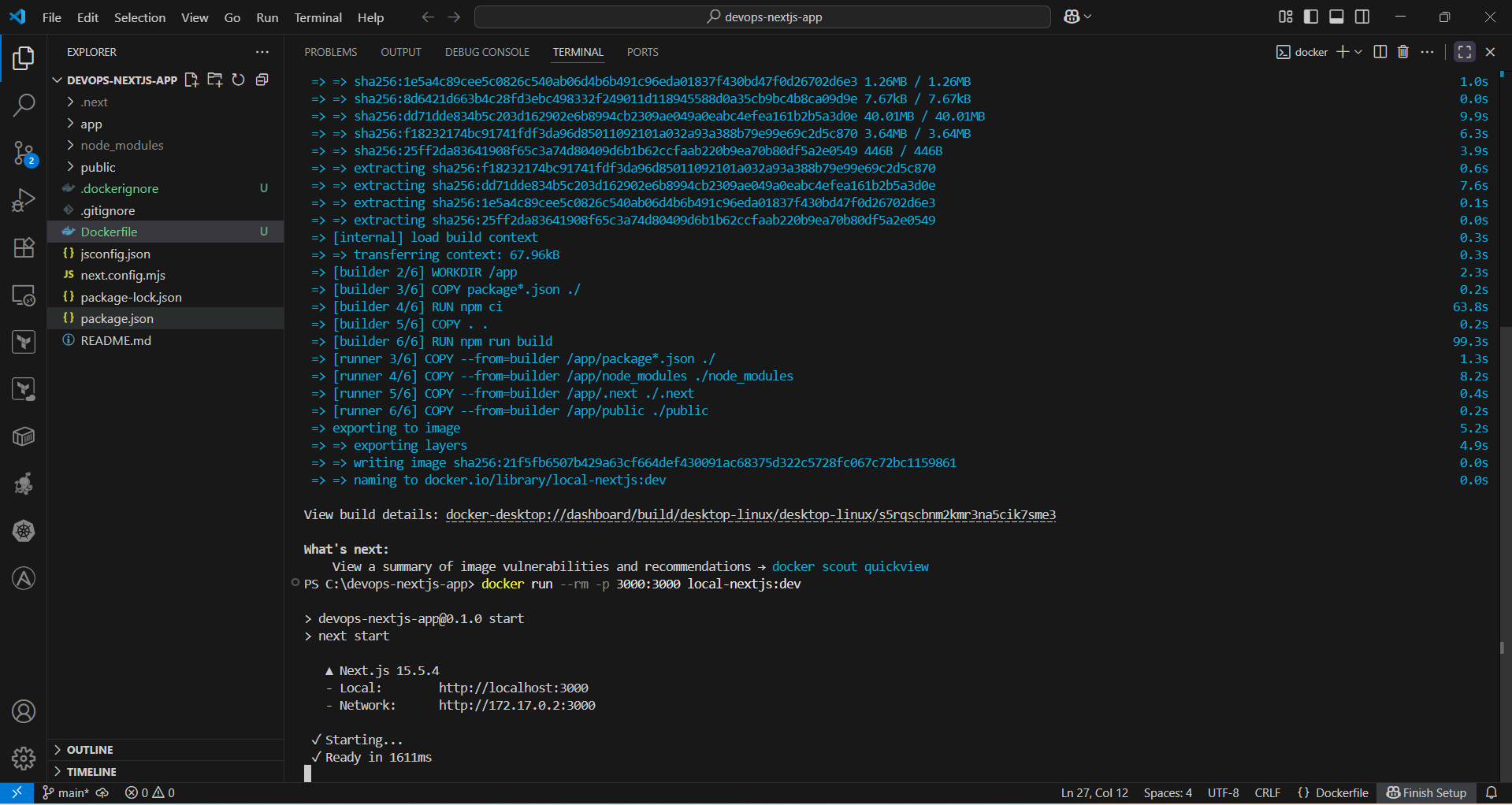
**STEP 3: Test Docker locally**

From project root:

build: docker build -t local-nextjs:dev .



run: docker run --rm -p 3000:3000 local-nextjs:dev



We can see the container in docker desktop.

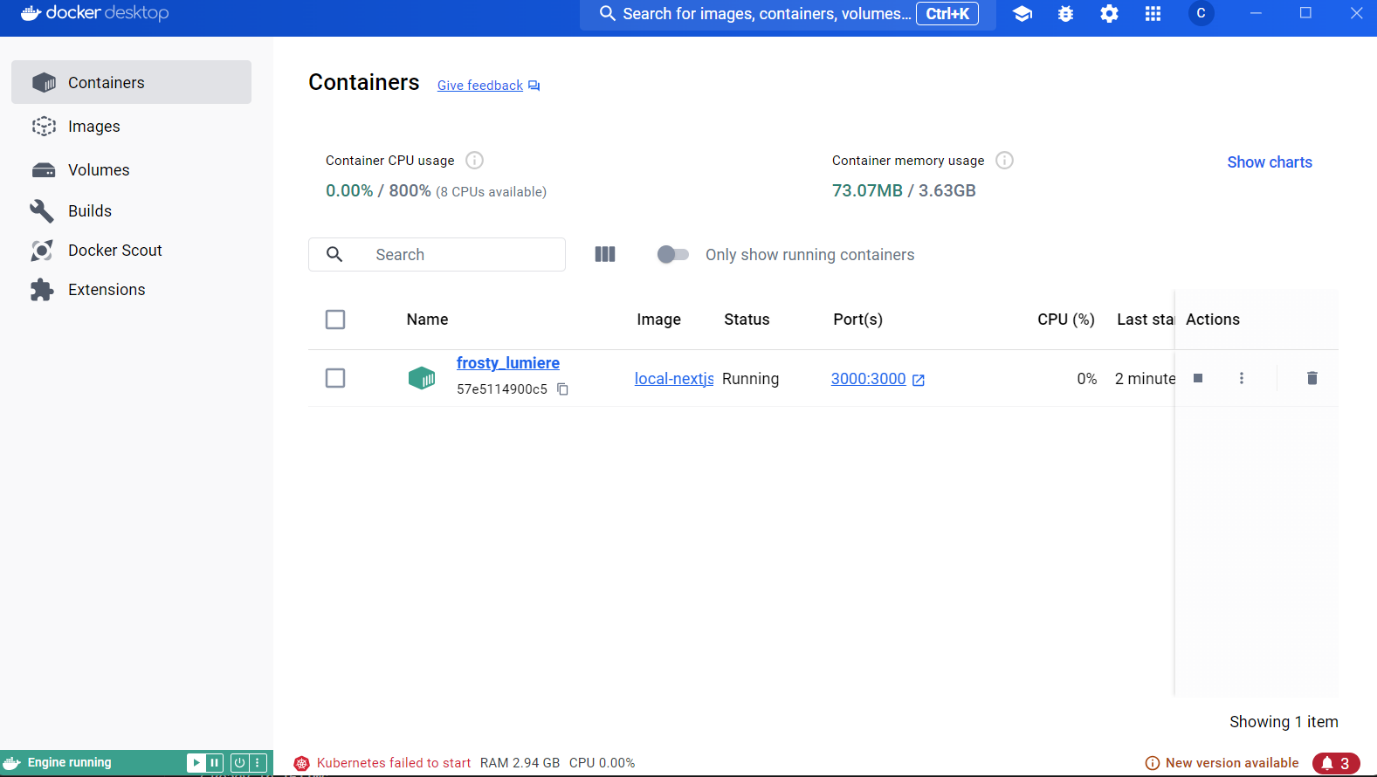
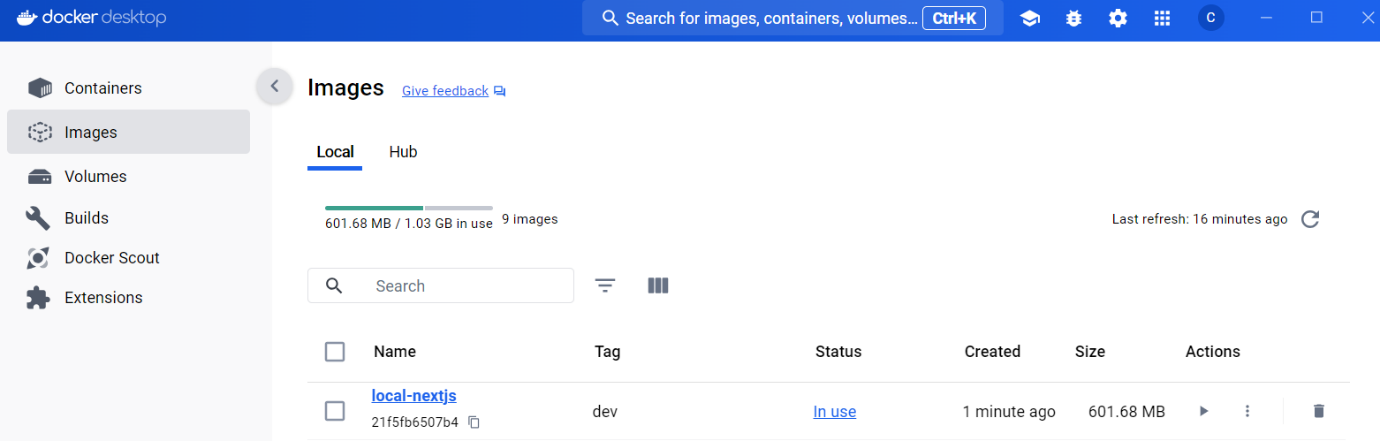
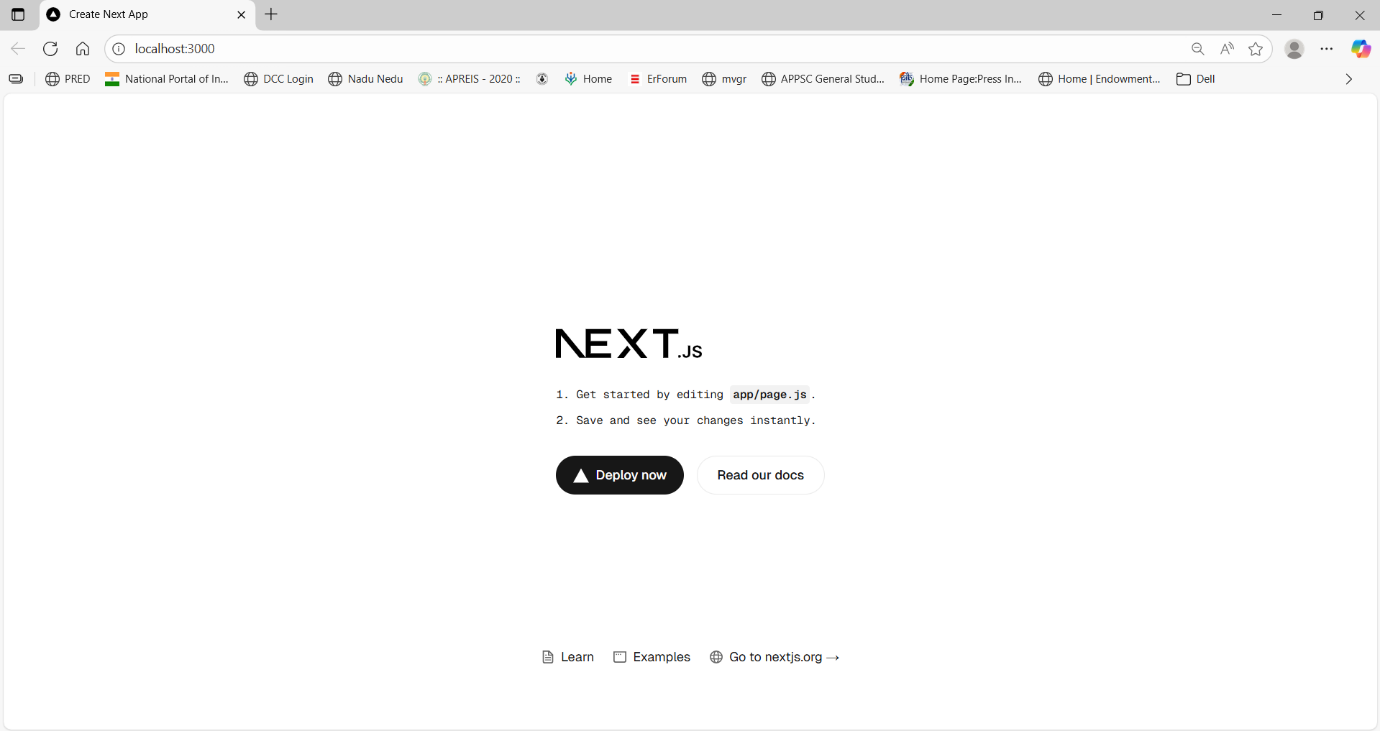


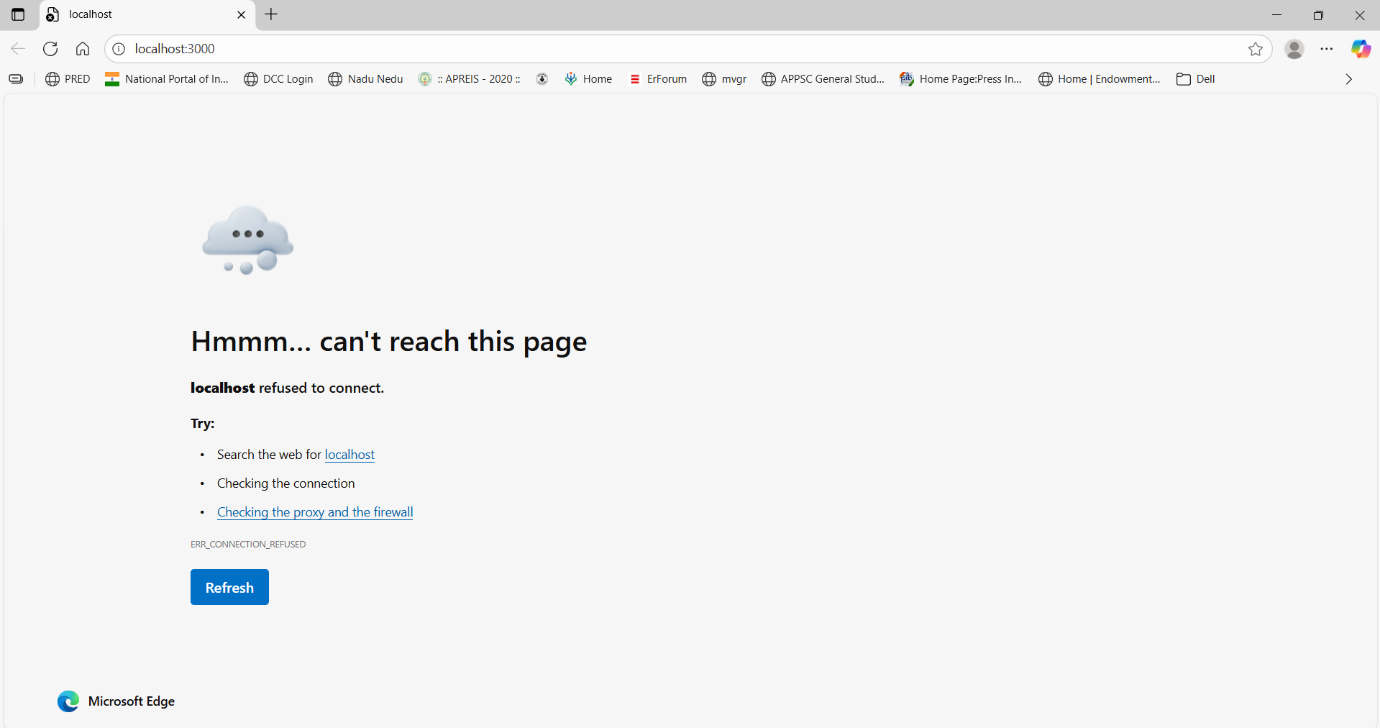
Image we see here



open http://localhost:3000



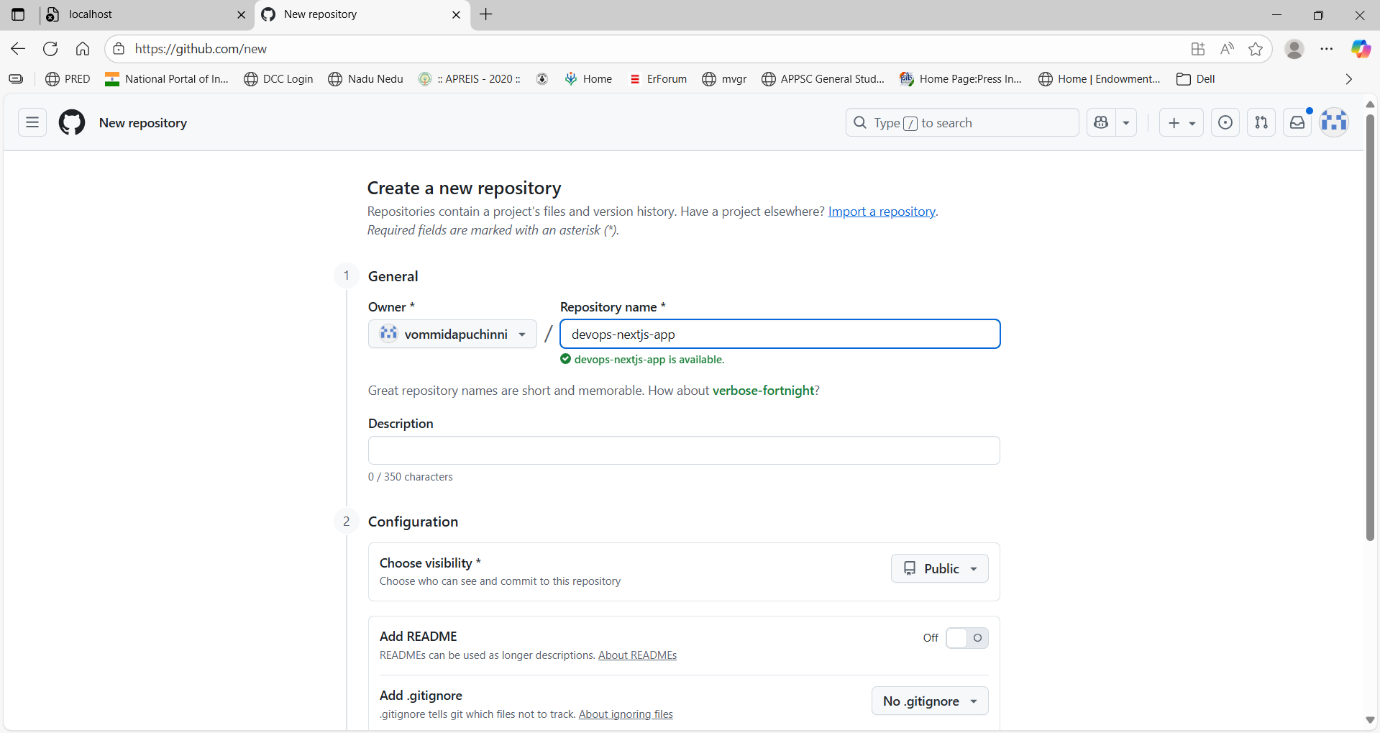
We cannot access the next app when the docker container stopped.



# **STEP 4: Create GitHub repository (clicks + commands)**

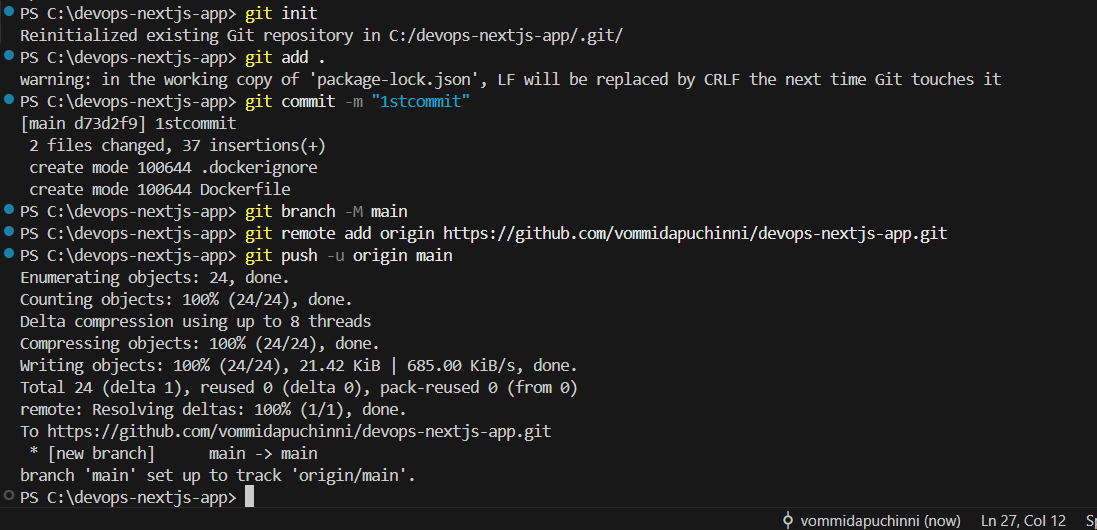
GitHub UI (clicks):

1. Sign in to GitHub.
2. Top-right + → New repository.
3. Name it devops-nextjs-app (or your choice).
4. Choose Public. (Important for easy GHCR pulls unless you want private images.)
5. Create a repository.



Git commands to push to origin:

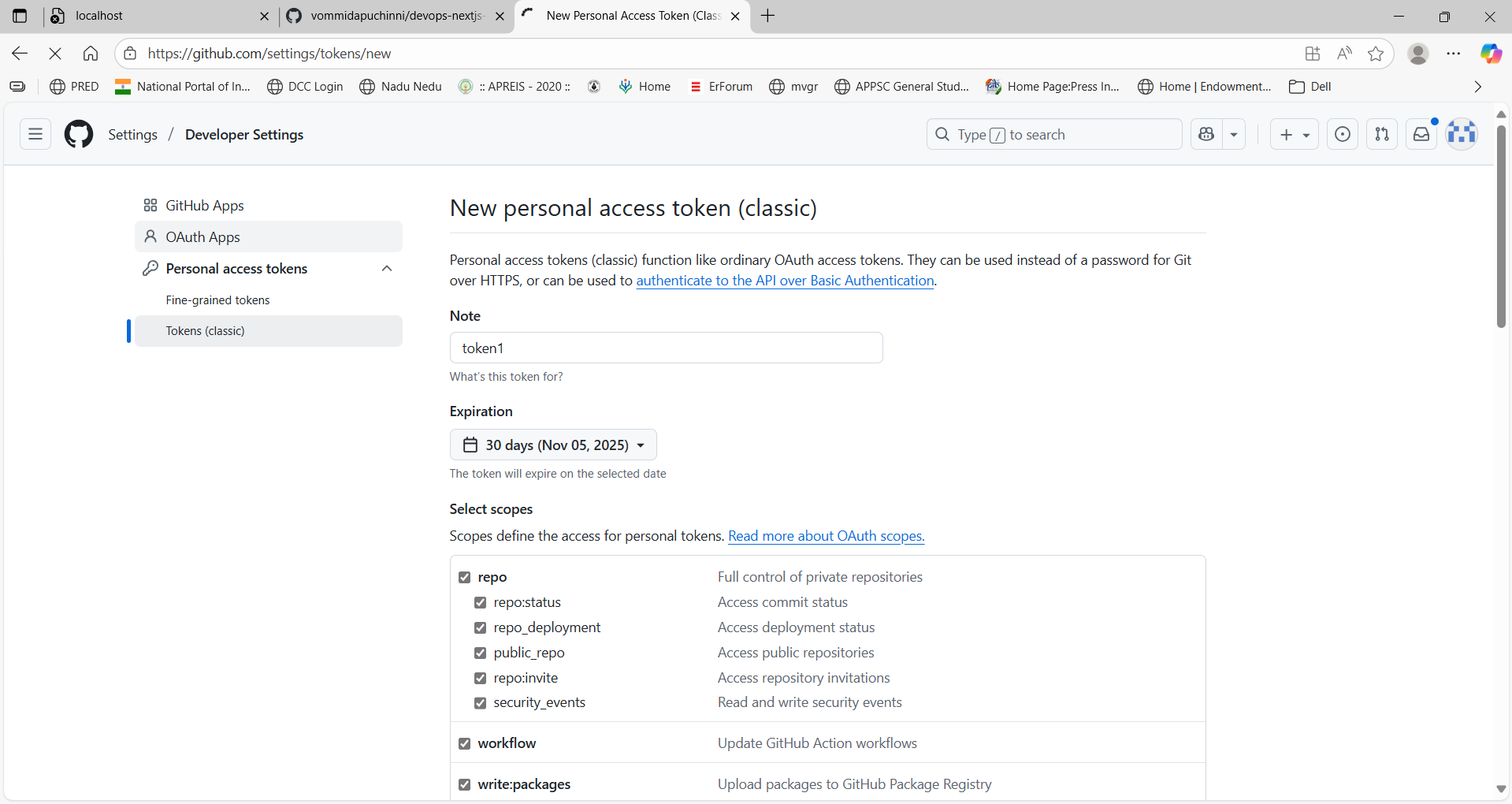
1. git init
2. git add .
3. git commit -m "Initial Next.js app + Dockerfile + k8s"
4. git branch -M main
5. git remote add origin https://github.com/<YOUR\_USERNAME>/devops-nextjs-app.git
6. git push -u origin main



# **STEP 5: Create GHCR Personal Access Token (PAT) only if needed**

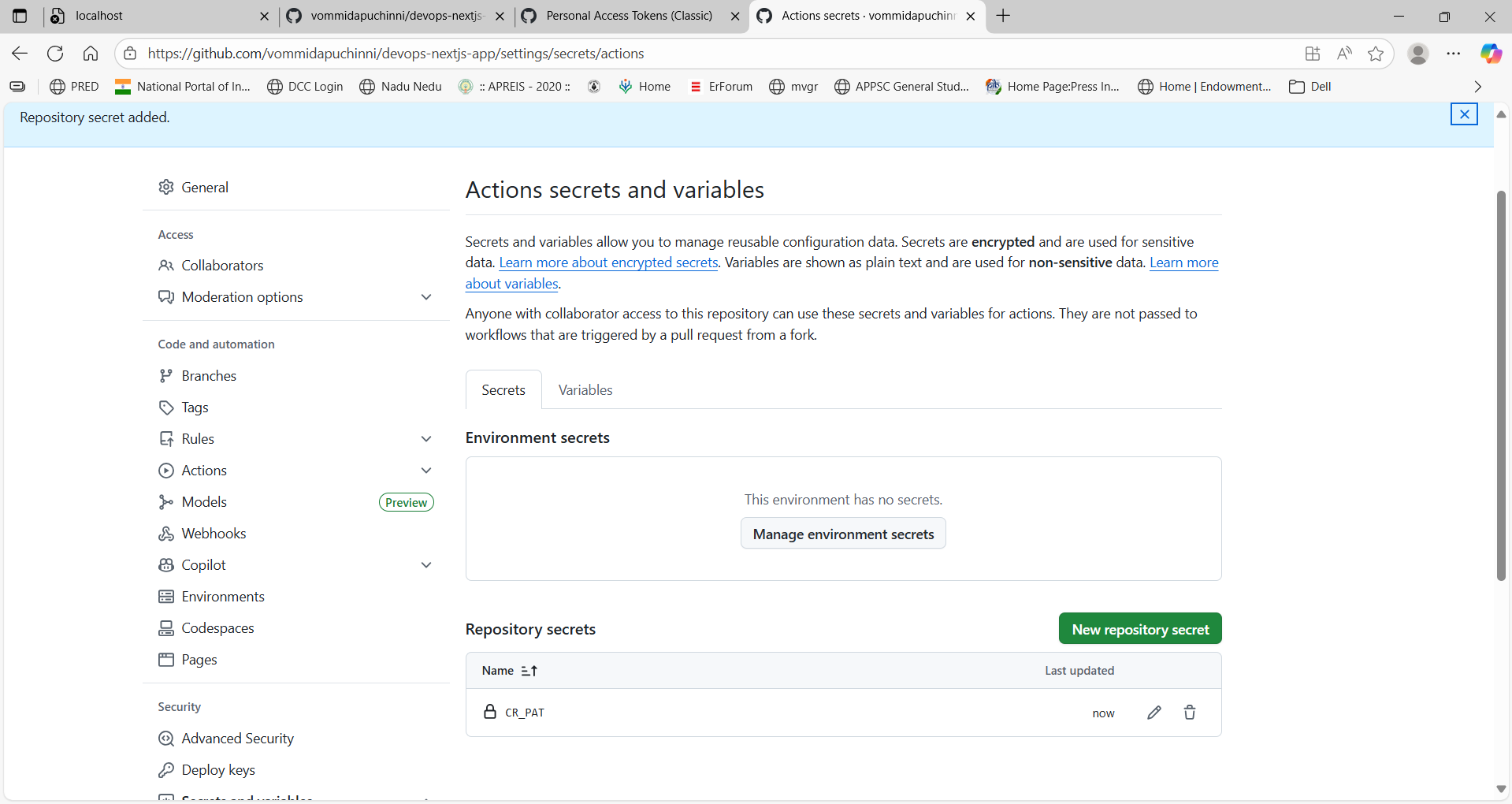
**Create PAT:**

1. GitHub top-right avatar → Settings.
2. Left menu → Developer settings → Personal access tokens → Tokens (classic).
3. Click Generate new token (classic).
4. Give name, expiry (e.g., 90d), then check scopes:  
   * write:packages (required to push images)
   * read:packages (to pull if needed)
   * optionally delete:packages
   * (If repo is private and you need repo access, add repo.)
5. Click Generate token → Copy token value (you’ll see it only once).



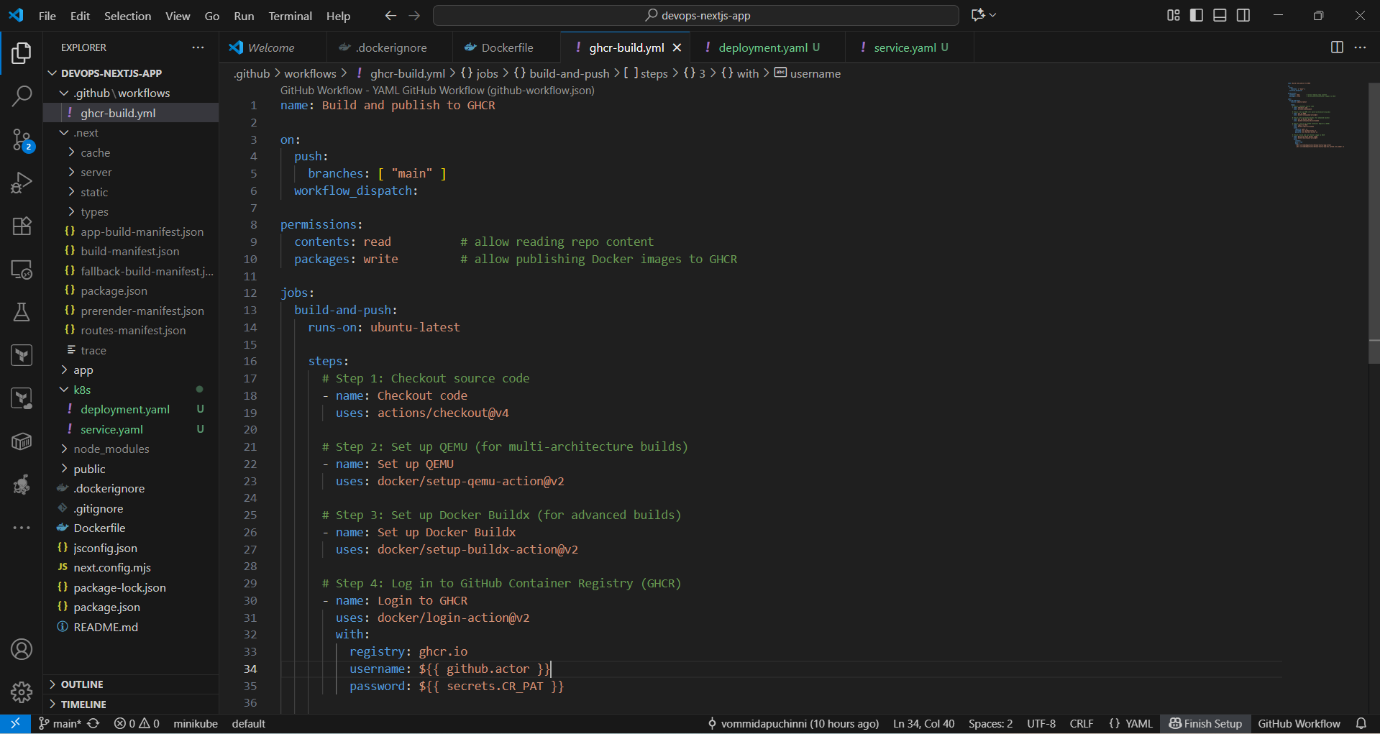
**Add PAT to repo secrets:**

* GitHub repo → **Settings** → **Secrets and variables** → **Actions** → **New repository secret**.
* Name: CR\_PAT
* Value: paste PAT → Save.

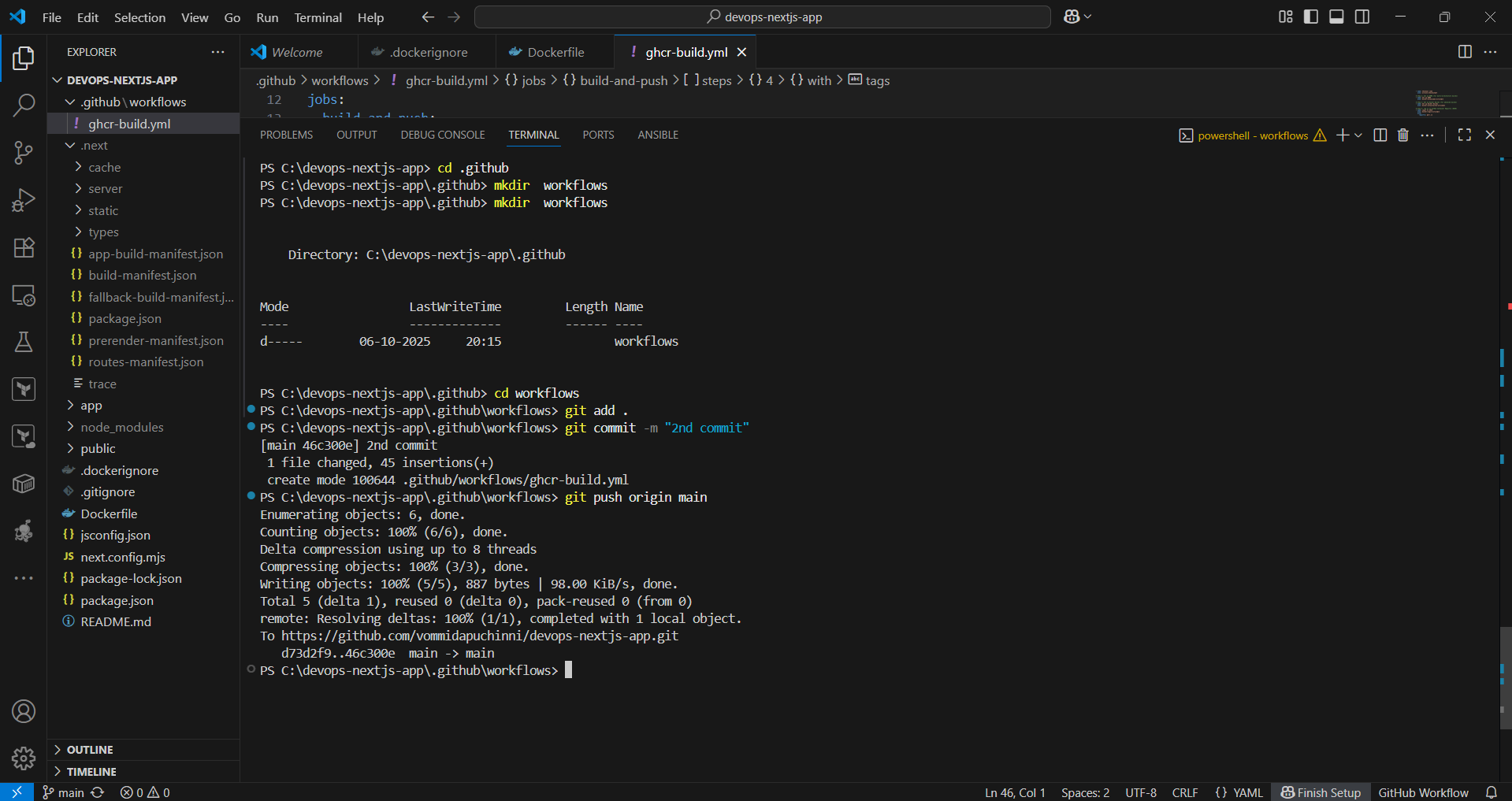


# **STEP 6: Add GitHub Actions workflow (exact file)**

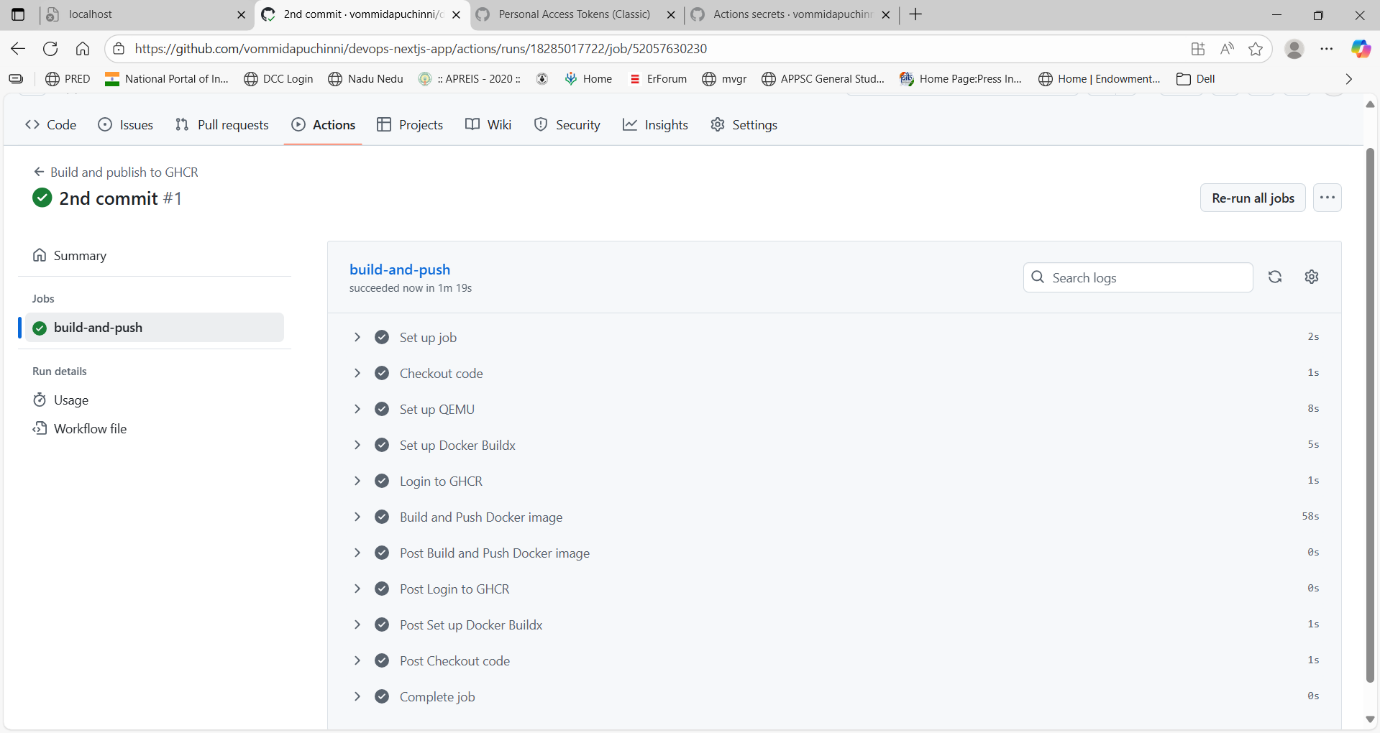
Create .github/workflow/ghcrbuild.yml



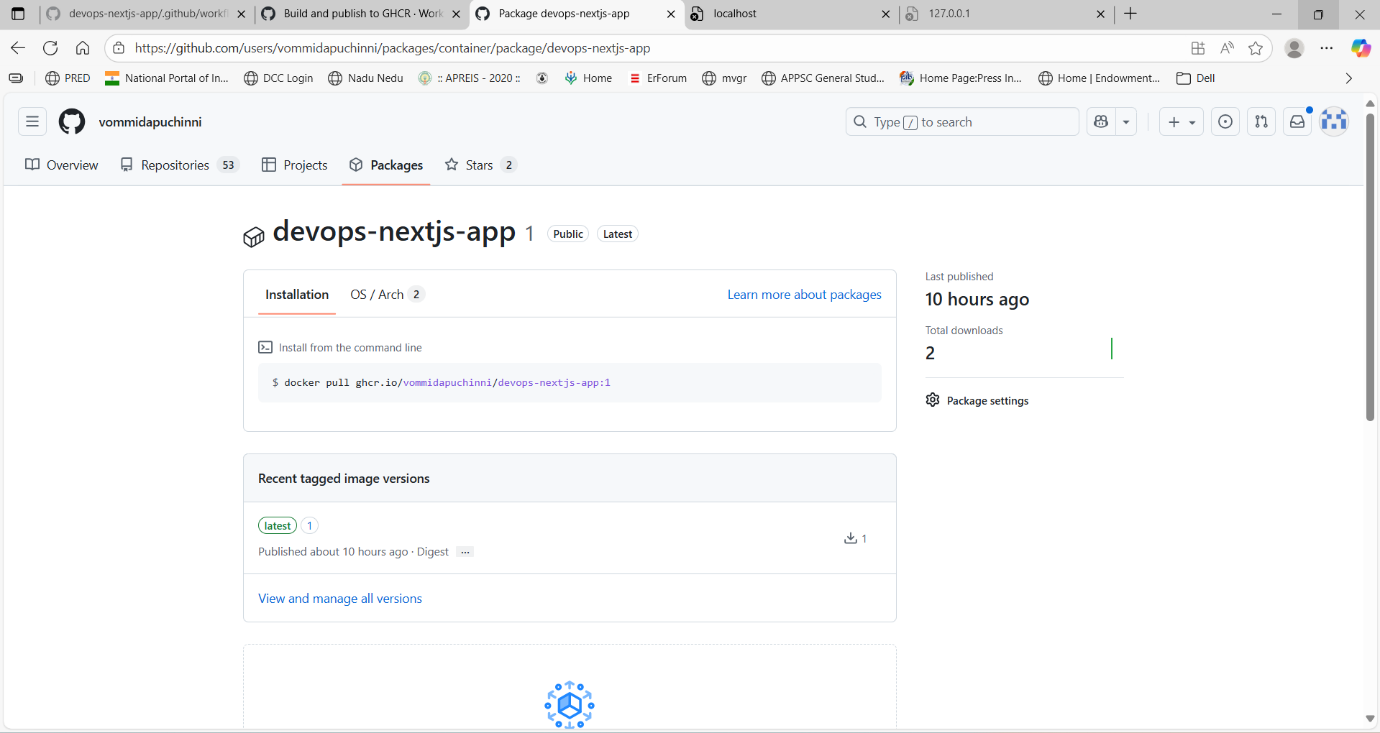
push files to the github repo



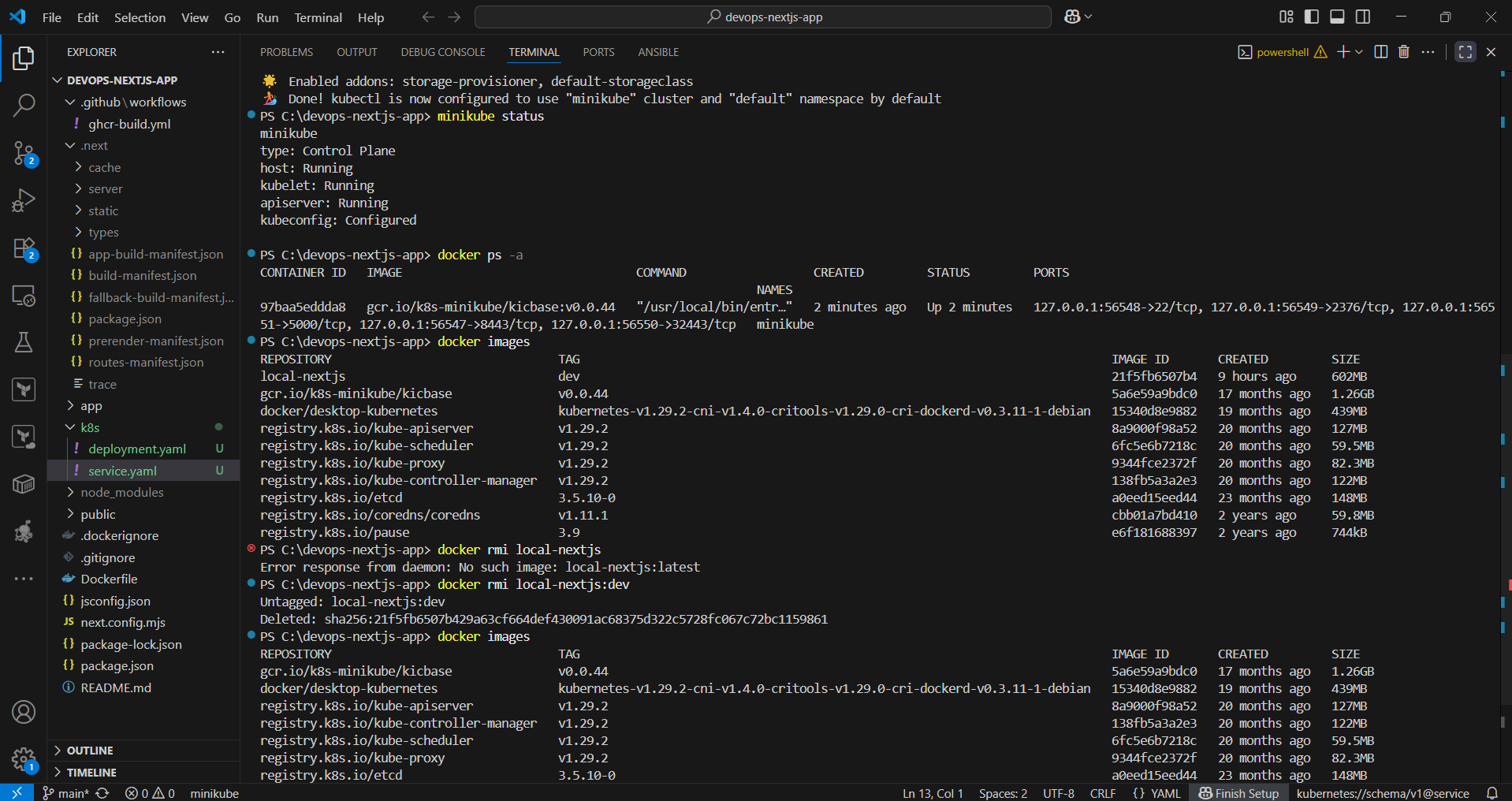
Run the workflow wait until job success



After job completion we see the ghcr image here.



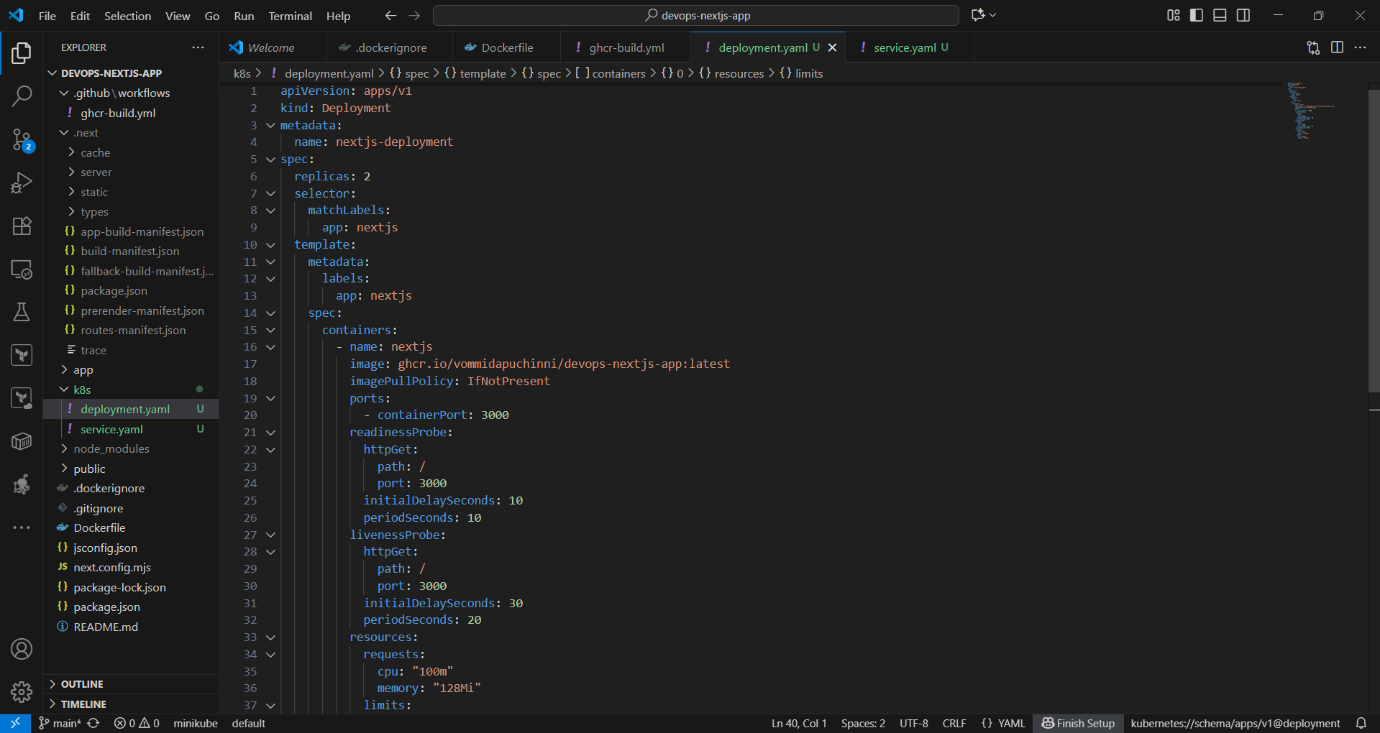
Start the minikube and before created docker images and containers delete those



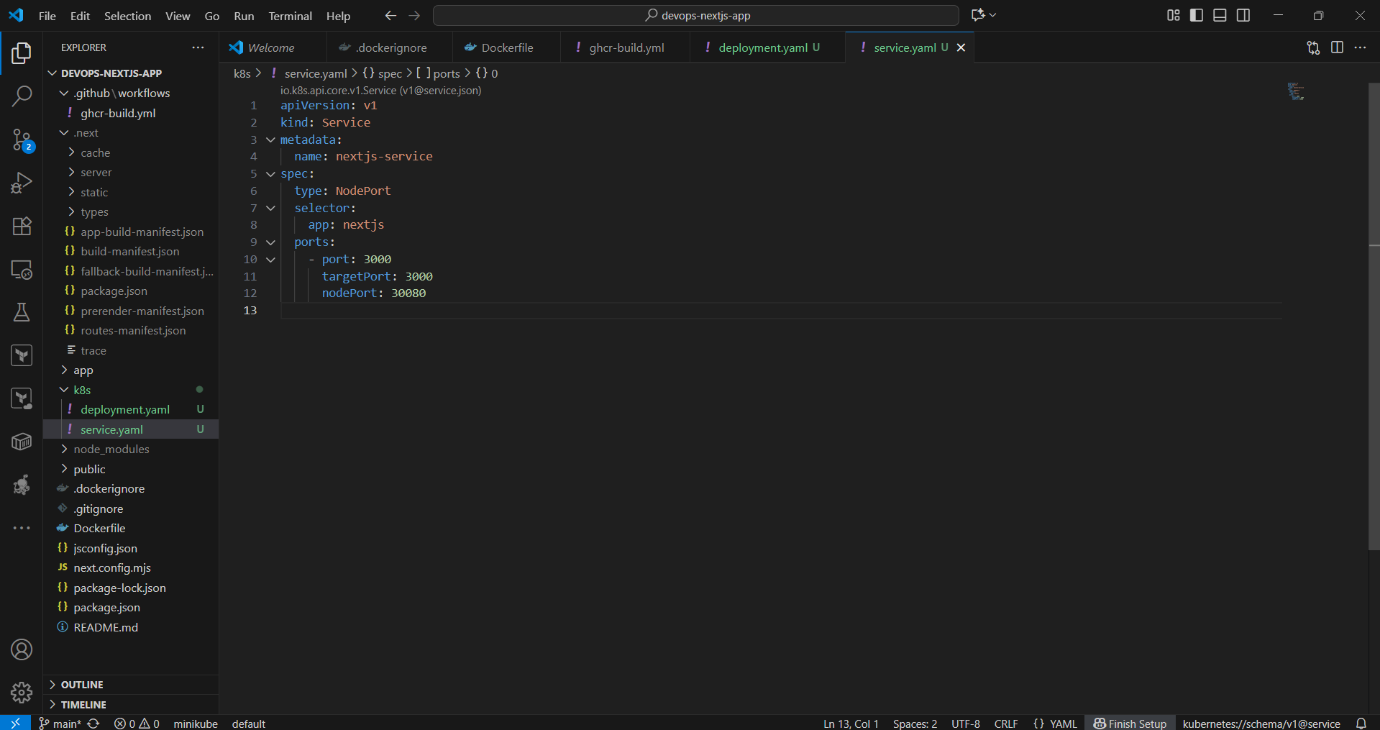
# **STEP 7: Create Kubernetes manifests (in k8s/)**

Make folder k8s/ and create these two files.

### k8s/deployment.yaml

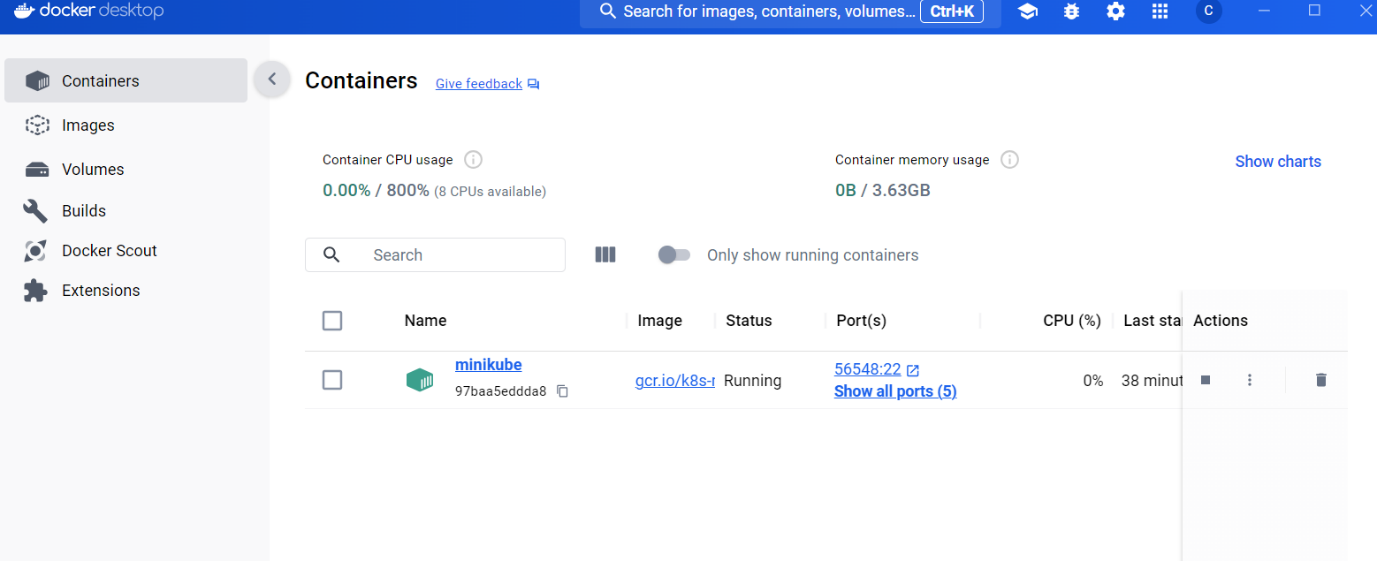


k8s/service.yaml



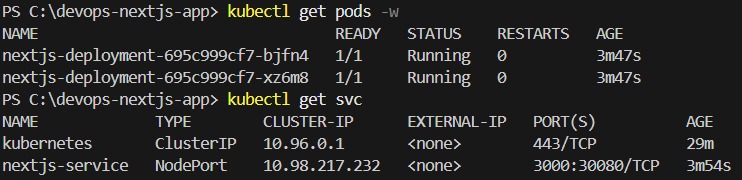
**STEP 9: Deploy to Minikube**

Make sure minikube running: minikube start --driver=docker



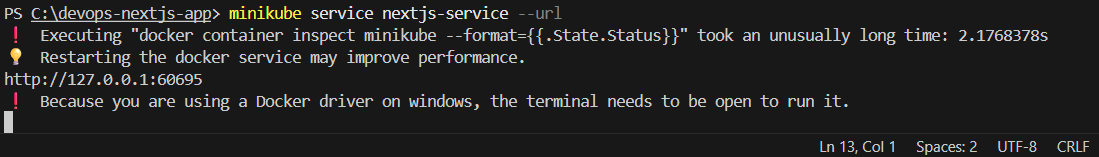
apply k8s manifests: kubectl apply -f k8s/

watch pods: kubectl get pods -w

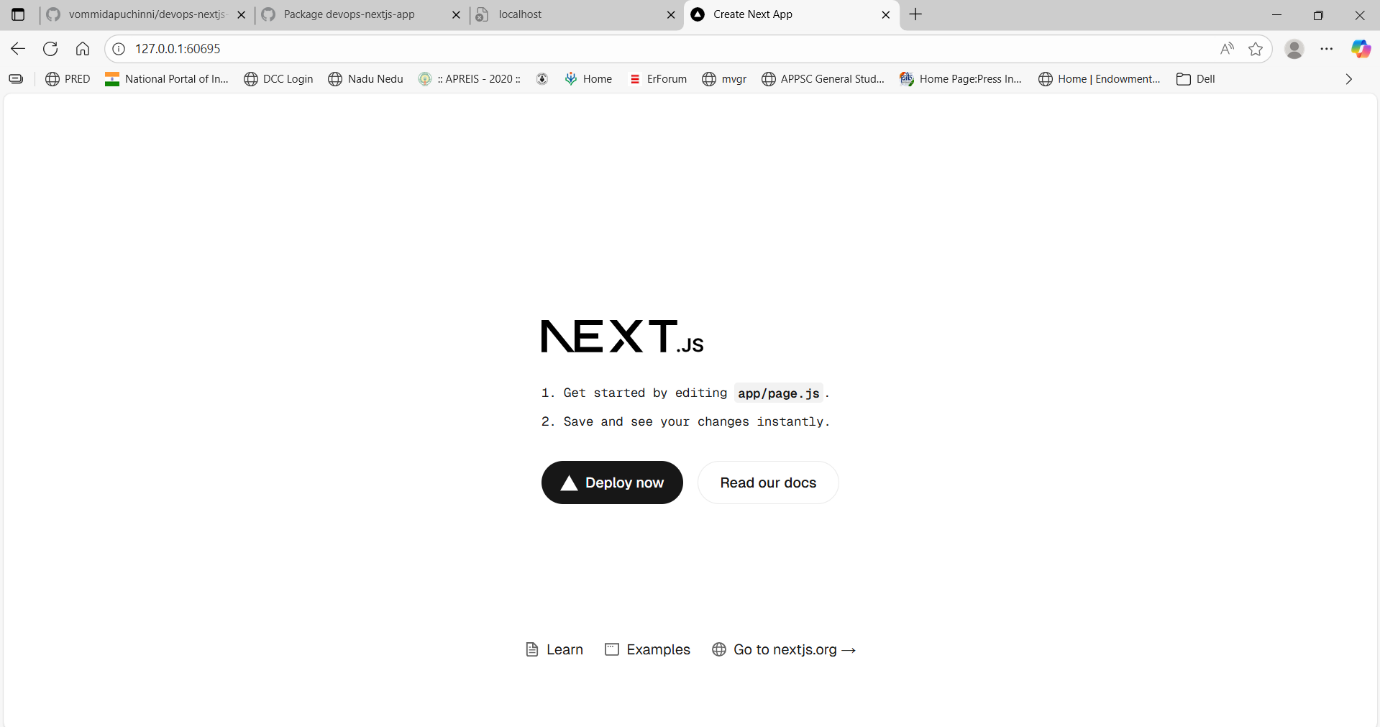


Then open service URL: minikube service nextjs-service --url

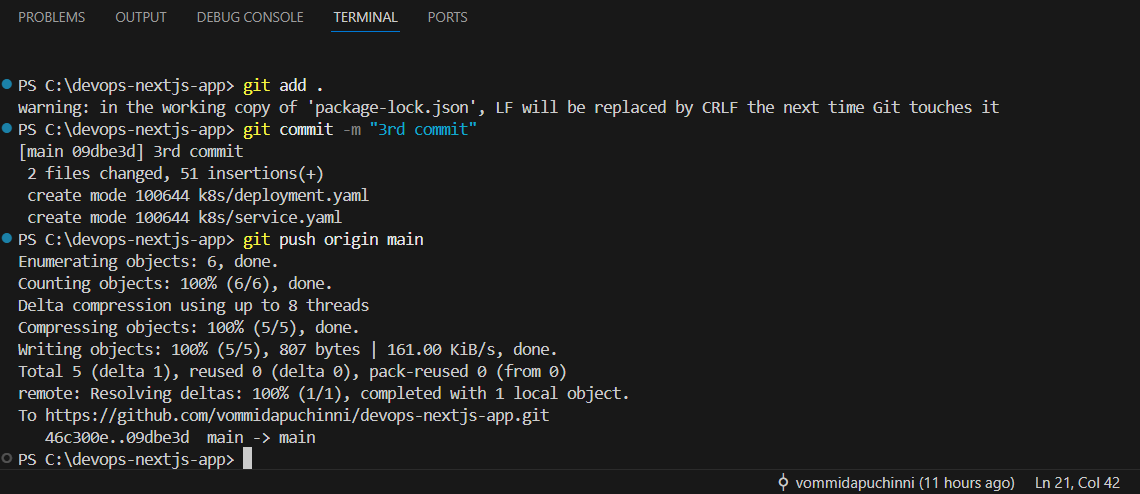
copy the URL and open it in browser

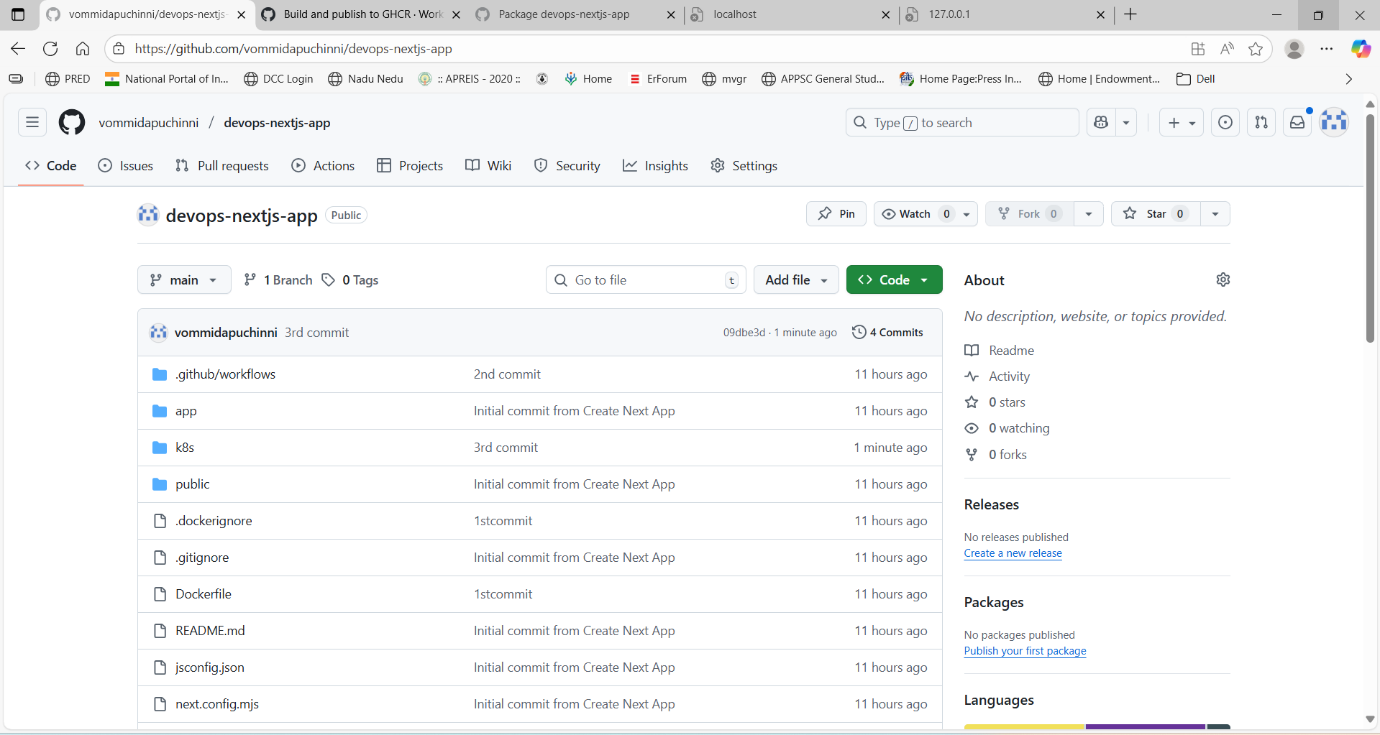


We can access the app by using this url provided here.

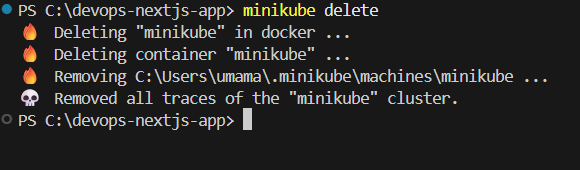


push codes to github.





terminate all after using



**Conclusion:** This project successfully containerized and deployed a Next.js app using Docker, automated image builds and pushes with GitHub Actions to GHCR, and deployed it on Minikube using Kubernetes manifests — demonstrating a complete CI/CD workflow.