

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)

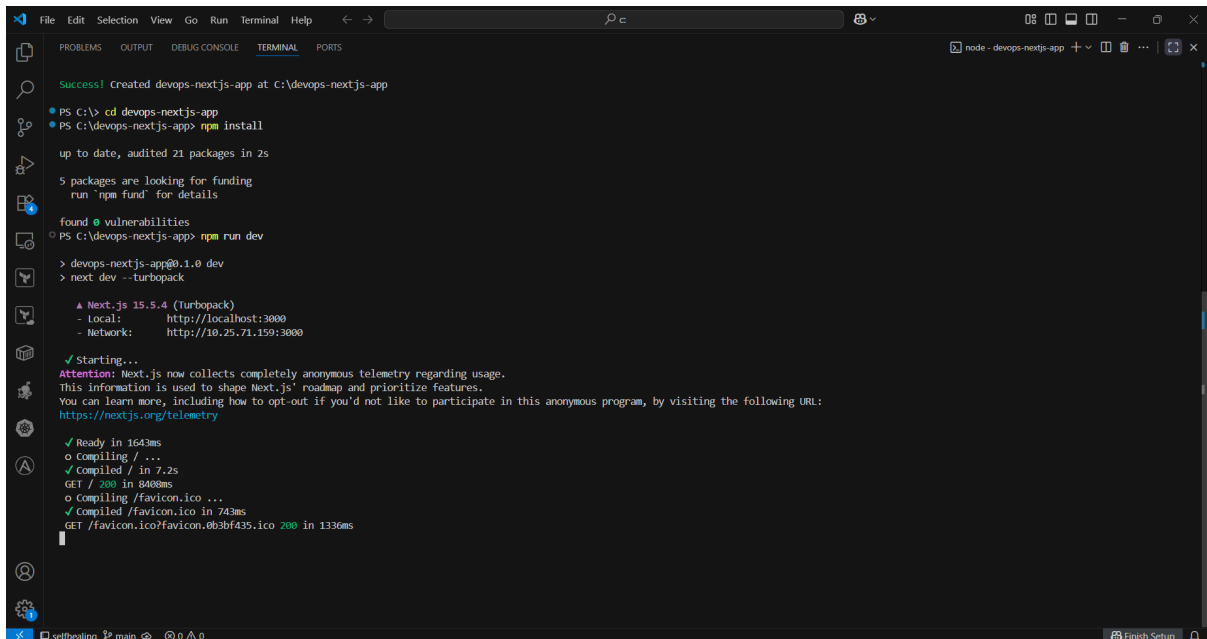
minikube installed (minikube version)

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



```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
node - devops-nextjs-app

Success! Created devops-nextjs-app at C:\devops-nextjs-app
PS C:\> cd devops-nextjs-app
PS C:\devops-nextjs-app> npm install

up to date, audited 21 packages in 2s

5 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\devops-nextjs-app> npm run dev

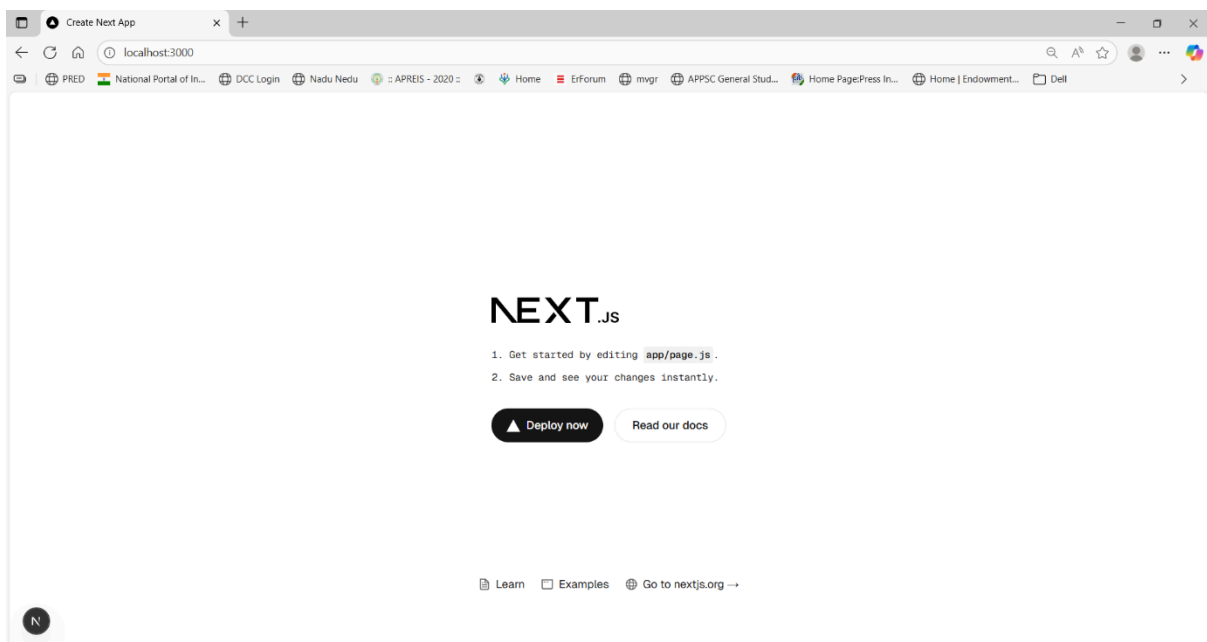
> devops-nextjs-app@0.1.0 dev
> next dev --turbo

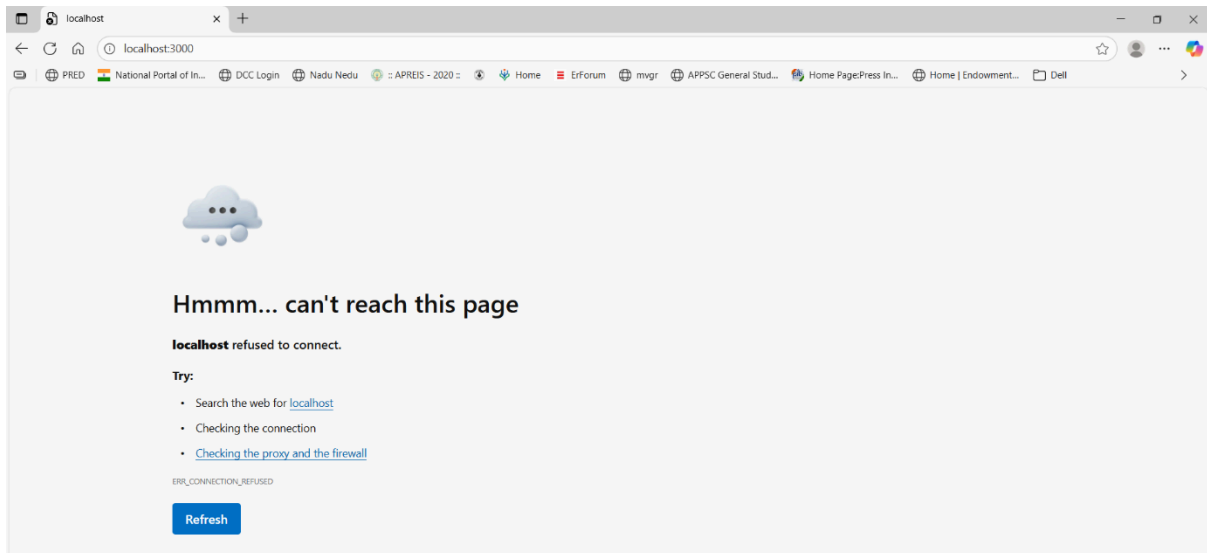
   ▲ Next.js 15.5.4 (turbo)
   - Local:    http://localhost:3000
   - Network:  http://10.25.71.159:3000

✓ Starting...
Attention: Next.js now collects completely anonymous telemetry regarding usage.
This information is used to shape Next.js' roadmap and prioritize features.
You can learn more, including how to opt-out if you'd not like to participate in this anonymous program, by visiting the following URL:
https://nextjs.org/telemetry

✓ Ready in 1643ms
  ○ Compiling / ...
  ✓ Compiled / in 7.2s
  GET / 200 in 840ms
  ○ Compiling /favicon.ico ...
  ✓ Compiled /favicon.ico in 743ms
  GET /favicon.ico/favicon.0b3bf435.ico 200 in 1336ms
```

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





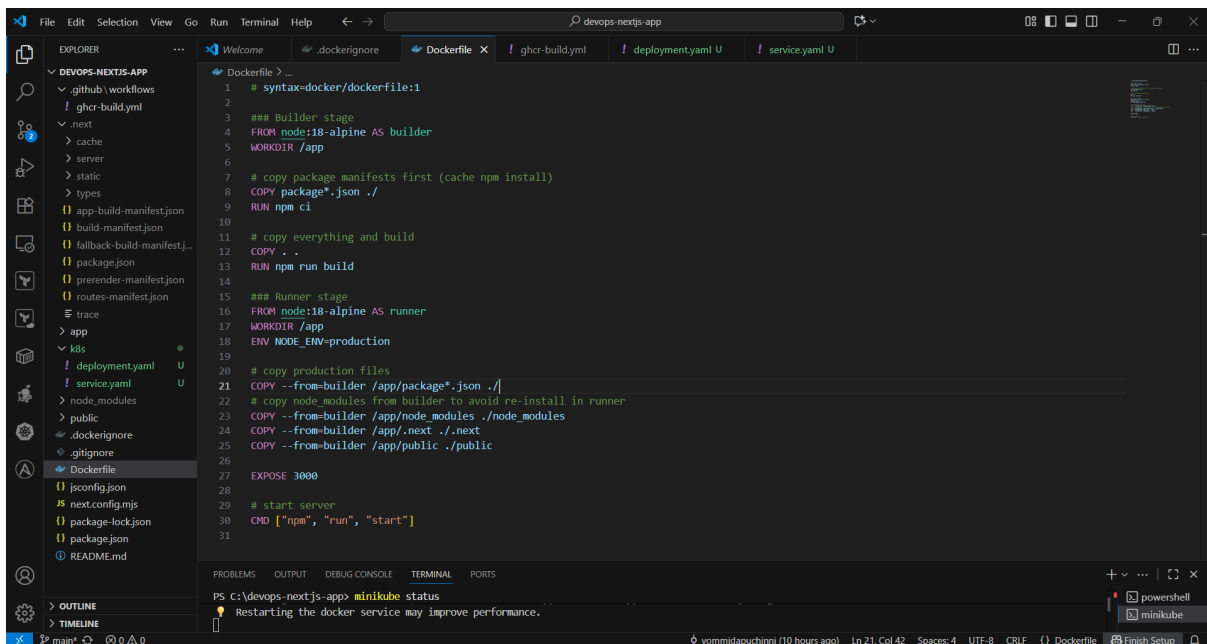
After stopping the npm run dev we cannot access the next app

STEP 2: Creating Docker file

.dockerignore: to ignore unwanted files.

Dockerfile (multi-stage — copy/paste)

Create Dockerfile

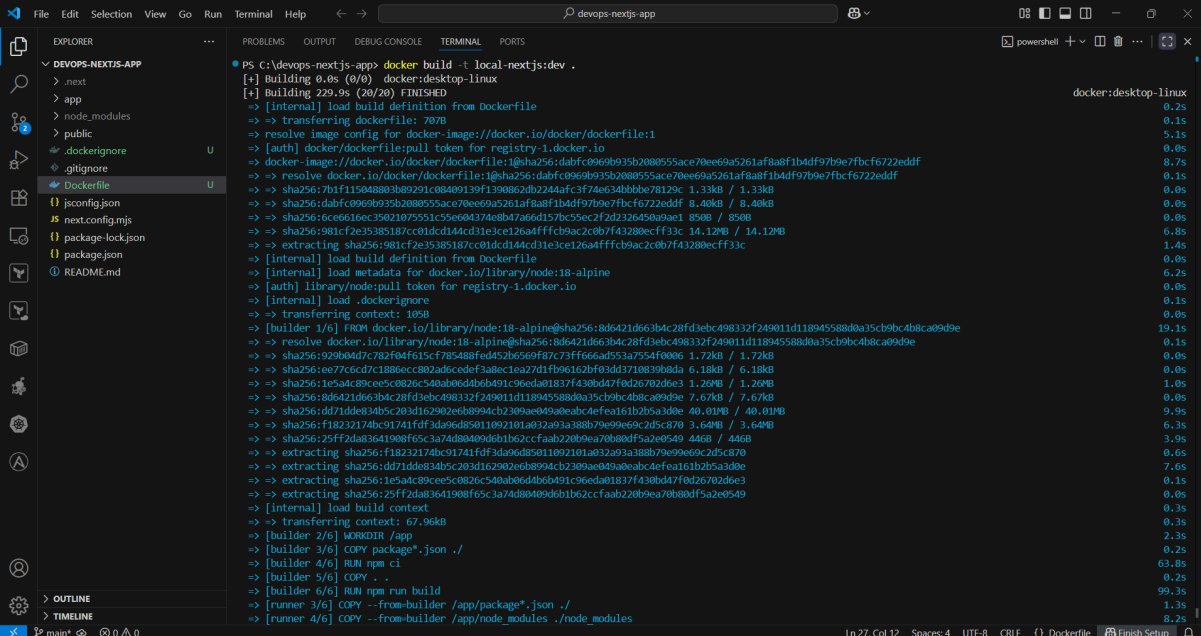


This is a simple, reliable Dockerfile. For smaller runtime images, you can use Next.js output: 'standalone' and adjust Dockerfile (advanced optimization).

STEP 3: Test Docker locally

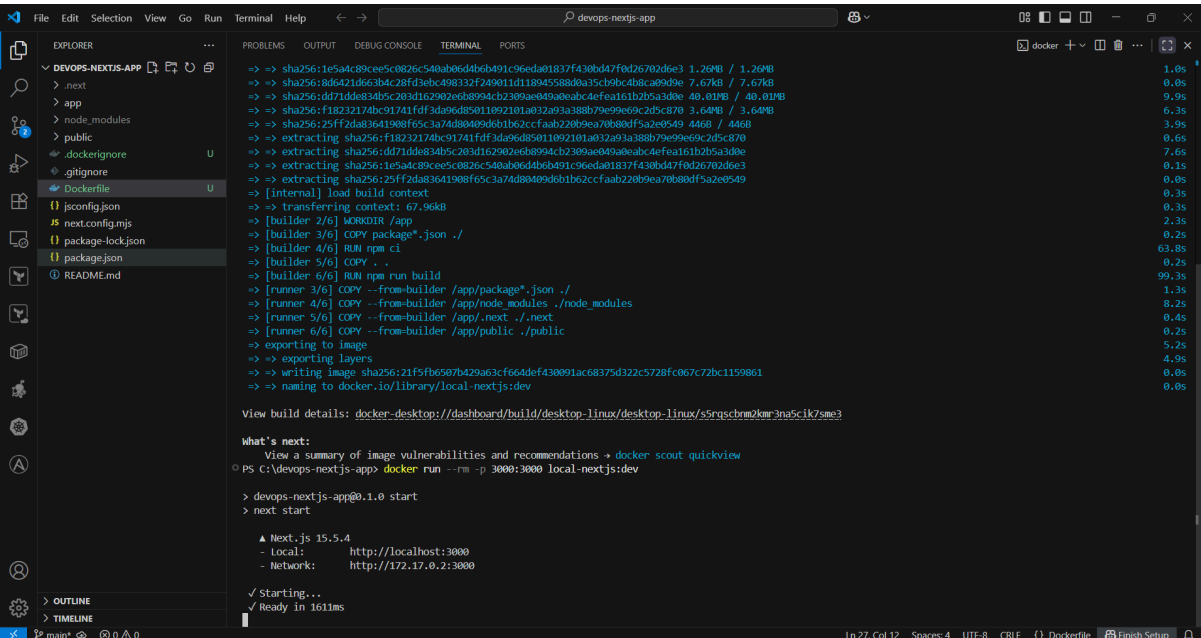
From project root:

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



```
PS C:\devops-nextjs-app> docker build -t local-nextjs:dev
[+] Building 0.0s (0/0) docker:desktop-linux
[+] Building 229.9s (20/20) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring Dockerfile: 707B
=> resolve image config for docker-image://docker.io/docker/dockerfile:1
=> [auth] docker/dockerfile:pull token for registry-1.docker.io
=> docker-image://docker.io/docker/dockerfile:1sha256:dabfc096b935b2080555ace70ee69a5261af8a8f1bd4f97b9e7fbc6722eddf
=> resolve docker.io/docker/dockerfile:1sha256:dabfc096b935b2080555ace70ee69a5261af8a8f1bd4f97b9e7fbc6722eddf
=> sha256:7b1f15048803b89291c08409139f1390862db2244f3f74e634bbbbe78129c 1.33kB / 1.33kB
=> sha256:dabfc096b935b2080555ace70ee69a5261af8a8f1bd4f97b9e7fbc6722eddf 8.40kB / 8.40kB
=> sha256:6ce6616ec35021075551c5e604374e8b47a66d157bc55ec2f2d2326450a0ae1 850B / 850B
=> sha256:981cf2e35385187cc01ddc144cd31e3c126a4ffcfb9ac2c0b7f43280ecff33c 14.12MB / 14.12MB
=> extracting sha256:981cf2e35385187cc01ddc144cd31e3c126a4ffcfb9ac2c0b7f43280ecff33c
=> [internal] load build definition from Dockerfile
=> [internal] load metadata for docker.io/library/node:18-alpine
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 105B
=> [builder 1/6] FROM docker.io/library/node:18-alpine@sha256:8d6421d663b4c28fd3ebc498332f249011d118945588da35cb9bc48ca09d9e
=> resolve docker.io/library/node:18-alpine@sha256:8d6421d663b4c28fd3ebc498332f249011d118945588da35cb9bc48ca09d9e
=> sha256:99b04047c7f04161c7f83488f4ed5b26969f6727ff666a55397554f0005 1.72kB / 1.72kB
=> sha256:ee77c6d7c1886cc802ad6cdecdf3a8ec1ea27d1f96162bf803d371083908da 6.18kB / 6.18kB
=> sha256:1e54dc89ceec5826c54ab06dd4bb491c96da01837f430bd47fd26702d6e3 1.26MB / 1.26MB
=> sha256:8d6421d663b4c28fd3ebc498332f249011d118945588da35cb9bc48ca09d9e 7.67kB / 7.67kB
=> sha256:dd71dd834b5c203d162902e6b8994cb2309a0a9a0eab4cfeaf161b2b5a3d0e 40.01MB / 40.01MB
=> sha256:f18232174bc91741fd3da96d85011092101a032a93a388b79e99e9c2d5c870 3.64MB / 3.64MB
=> sha256:25ff2da83641908f65c3a74d80409d0b1b62ccfaab220b9ea70b80df5a2e0549 446B / 446B
=> extracting sha256:f18232174bc91741fd3da96d85011092101a032a93a388b79e99e9c2d5c870
=> extracting sha256:dd71dd834b5c203d162902e6b8994cb2309a0a9a0eab4cfeaf161b2b5a3d0e
=> extracting sha256:1e54dc89ceec5826c54ab06dd4bb491c96da01837f430bd47fd26702d6e3
=> extracting sha256:25ff2da83641908f65c3a74d80409d0b1b62ccfaab220b9ea70b80df5a2e0549
=> [internal] load build context
=> => transferring context: 67.96kB
=> [builder 2/6] WORKDIR /app
=> [builder 3/6] COPY package*.json ./
=> [builder 4/6] RUN npm ci
=> [builder 5/6] COPY . .
=> [builder 6/6] RUN npm run build
=> [runner 3/6] COPY --from-builder /app/package*.json ./
=> [runner 4/6] COPY --from-builder /app/node_modules ./node_modules
```

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



```
=> sha256:1e54dc89ceec5826c54ab06dd4bb491c96da01837f430bd47fd26702d6e3 1.26MB / 1.26MB
=> sha256:8d6421d663b4c28fd3ebc498332f249011d118945588da35cb9bc48ca09d9e 7.67kB / 7.67kB
=> sha256:dd71dd834b5c203d162902e6b8994cb2309a0a9a0eab4cfeaf161b2b5a3d0e 40.01MB / 40.01MB
=> sha256:f18232174bc91741fd3da96d85011092101a032a93a388b79e99e9c2d5c870 3.64MB / 3.64MB
=> sha256:25ff2da83641908f65c3a74d80409d0b1b62ccfaab220b9ea70b80df5a2e0549 446B / 446B
=> extracting sha256:f18232174bc91741fd3da96d85011092101a032a93a388b79e99e9c2d5c870
=> extracting sha256:dd71dd834b5c203d162902e6b8994cb2309a0a9a0eab4cfeaf161b2b5a3d0e
=> extracting sha256:1e54dc89ceec5826c54ab06dd4bb491c96da01837f430bd47fd26702d6e3
=> extracting sha256:25ff2da83641908f65c3a74d80409d0b1b62ccfaab220b9ea70b80df5a2e0549
=> [internal] load build context
=> => transferring context: 67.96kB
=> [builder 2/6] WORKDIR /app
=> [builder 3/6] COPY package*.json ./
=> [builder 4/6] RUN npm ci
=> [builder 5/6] COPY . .
=> [builder 6/6] RUN npm run build
=> [runner 3/6] COPY --from-builder /app/package*.json ./
=> [runner 4/6] COPY --from-builder /app/node_modules ./node_modules
=> [runner 5/6] COPY --from-builder /app/next ./next
=> [runner 6/6] COPY --from-builder /app/public ./public
=> exporting image
=> exporting layers
=> writing image sha256:21f5fb6507b429a63cf64def430091ac08375d322c5728fc067c72bc1159861
=> naming to docker.io/library/local-nextjs:dev

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/S9rscbm0km3na5c1k7sme3

What's next:
View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\devops-nextjs-app> docker run --rm -p 3000:3000 local-nextjs:dev
> devops-nextjs-app@0.1.0 start
> next start

  ▲ Next.js 15.5.4
  - Local:    http://localhost:3000
  - Network:  http://172.17.0.2:3000

✓ Starting...
✓ Ready in 1611ms
```

We can see the container in docker desktop.

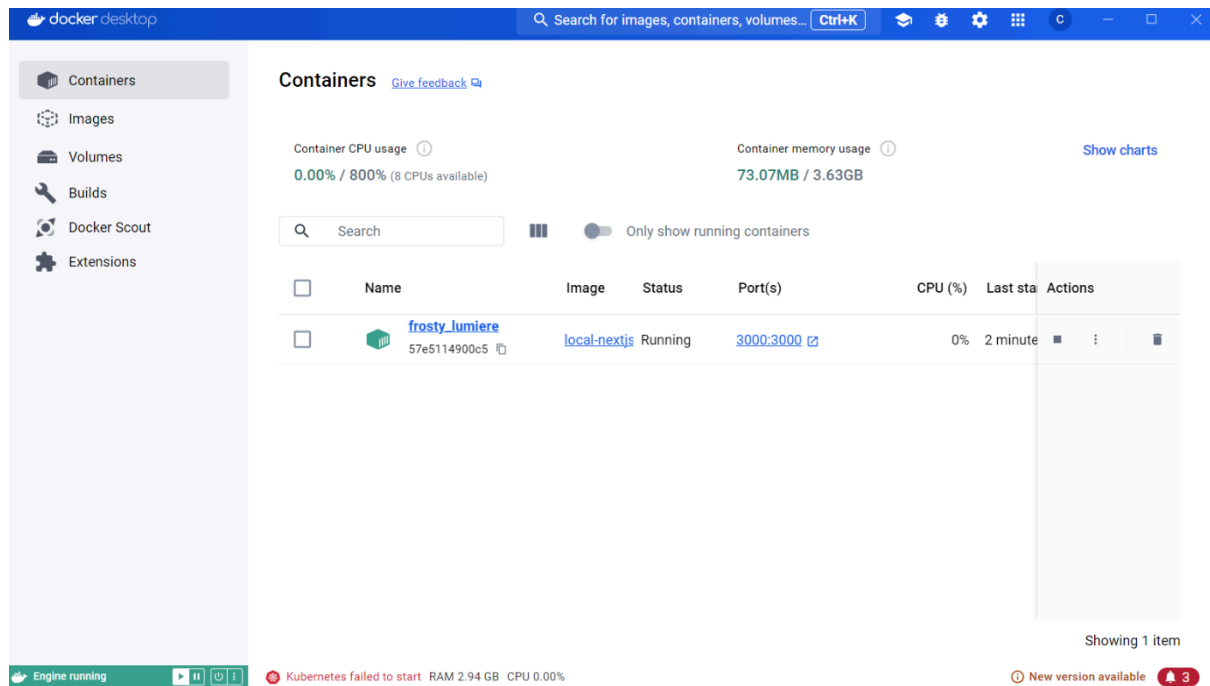
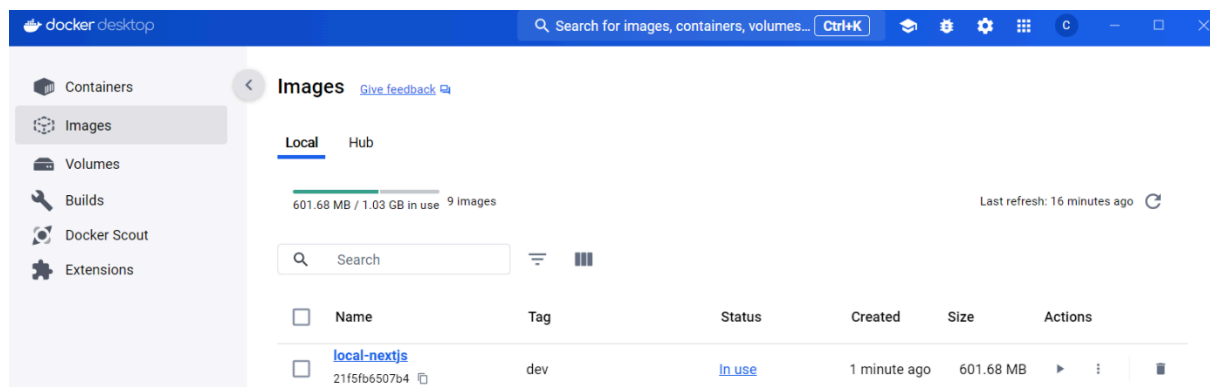
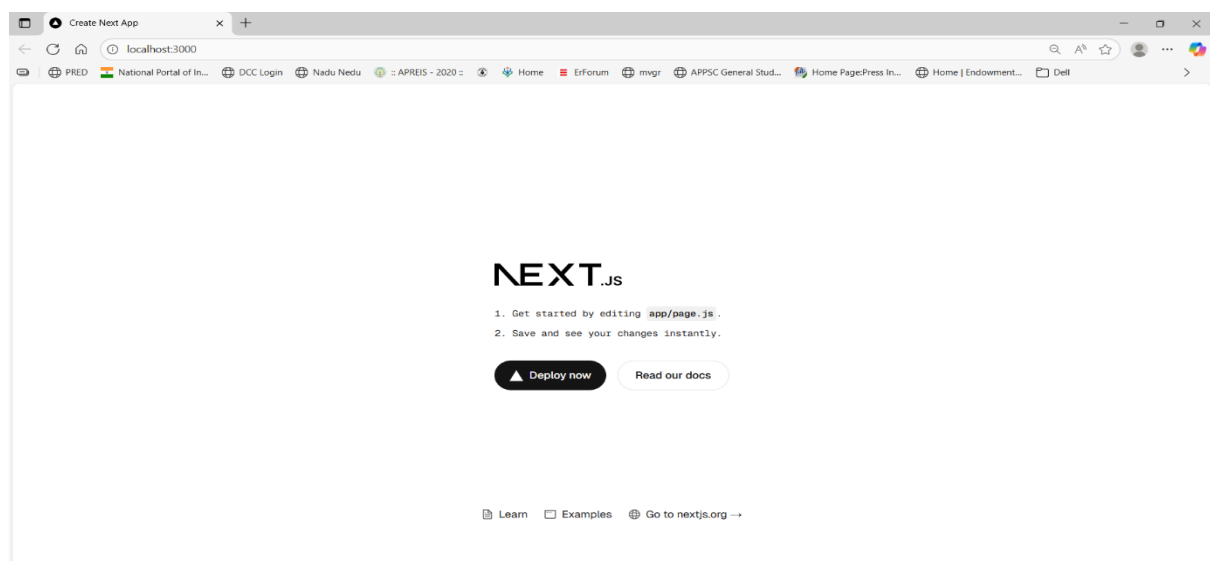


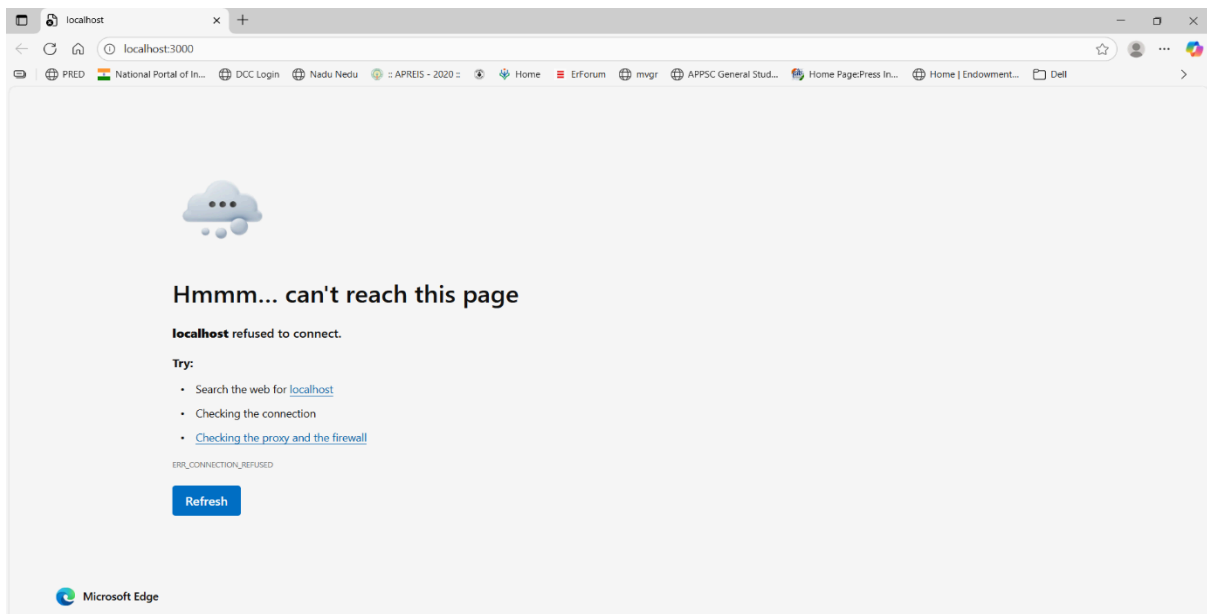
Image we see here



open <http://localhost:3000>



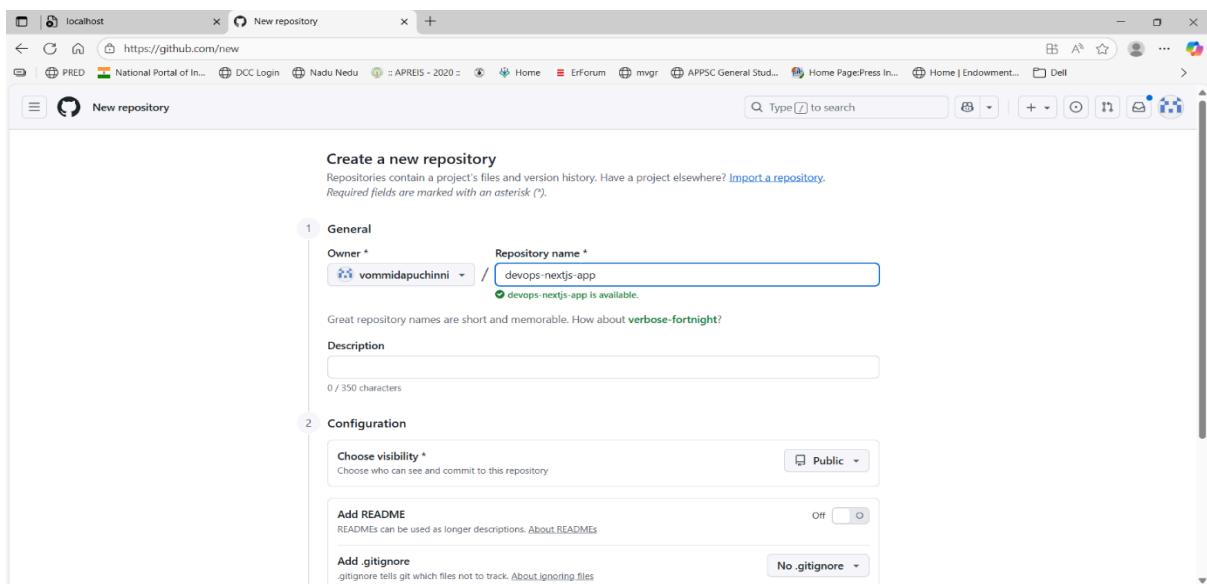
We cannot access the next app when the docker container stopped.



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`

```
PS C:\devops-nextjs-app> git init
Reinitialized existing Git repository in C:/devops-nextjs-app/.git/
PS C:\devops-nextjs-app> git add .
warning: in the working copy of 'package-lock.json', LF will be replaced by CRLF the next time Git touches it
PS C:\devops-nextjs-app> git commit -m "1stcommit"
[main d73d2f9] 1stcommit
 2 files changed, 37 insertions(+)
 create mode 100644 .dockerignore
 create mode 100644 Dockerfile
PS C:\devops-nextjs-app> git branch -M main
PS C:\devops-nextjs-app> git remote add origin https://github.com/vommidapuchinni/devops-nextjs-app.git
PS C:\devops-nextjs-app> git push -u origin main
Enumerating objects: 24, done.
Counting objects: 100% (24/24), done.
Delta compression using up to 8 threads
Compressing objects: 100% (24/24), done.
Writing objects: 100% (24/24), 21.42 KiB | 685.00 KiB/s, done.
Total 24 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/vommidapuchinni/devops-nextjs-app.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
PS C:\devops-nextjs-app> |
```

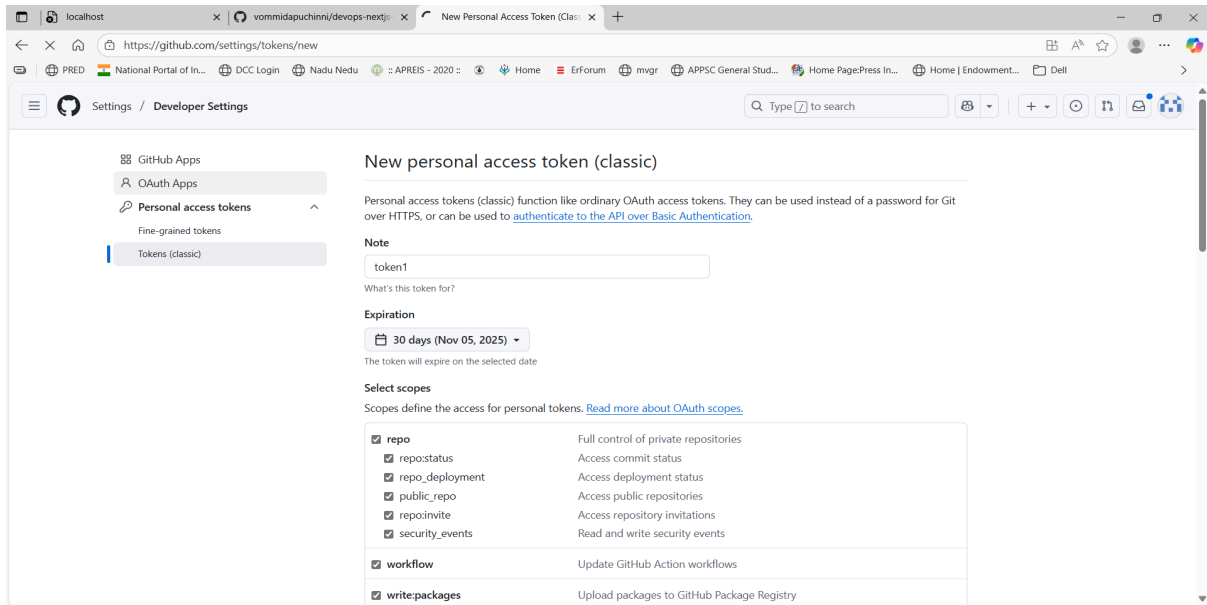
STEP 5: Create GHCR Personal Access Token (PAT) only if needed

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)

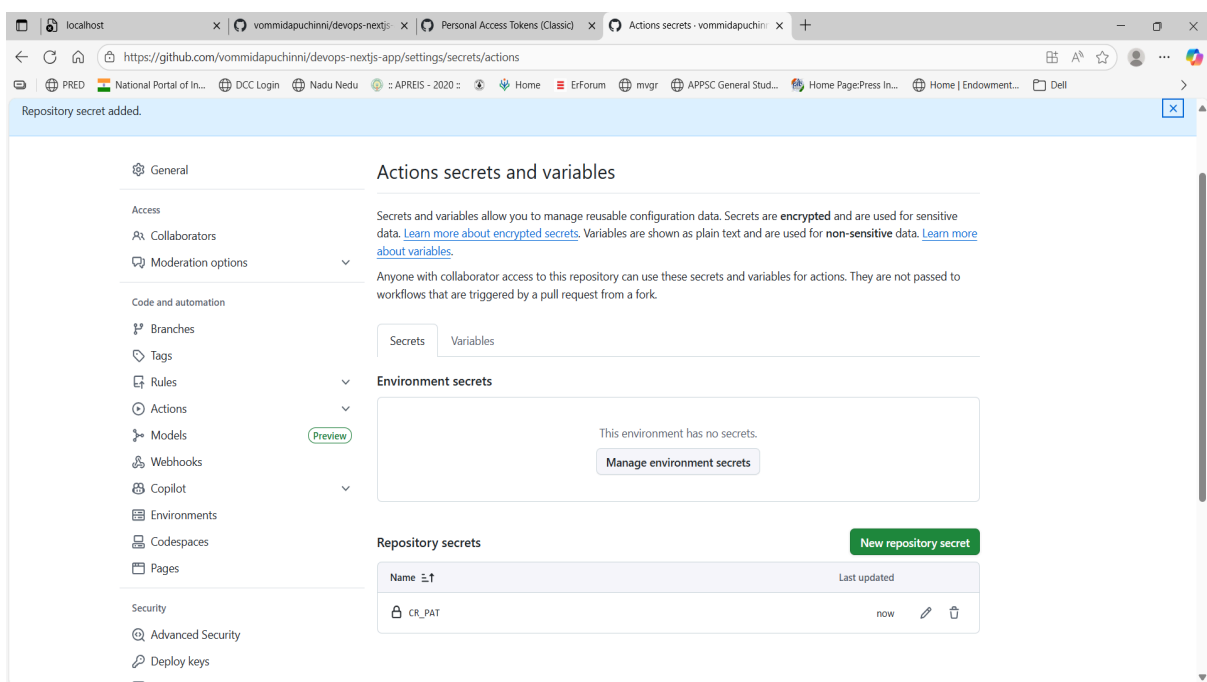
- optionally delete:packages
- (If repo is private and you need repo access, add repo.)

5. Click Generate token → Copy token value (you'll see it only once).



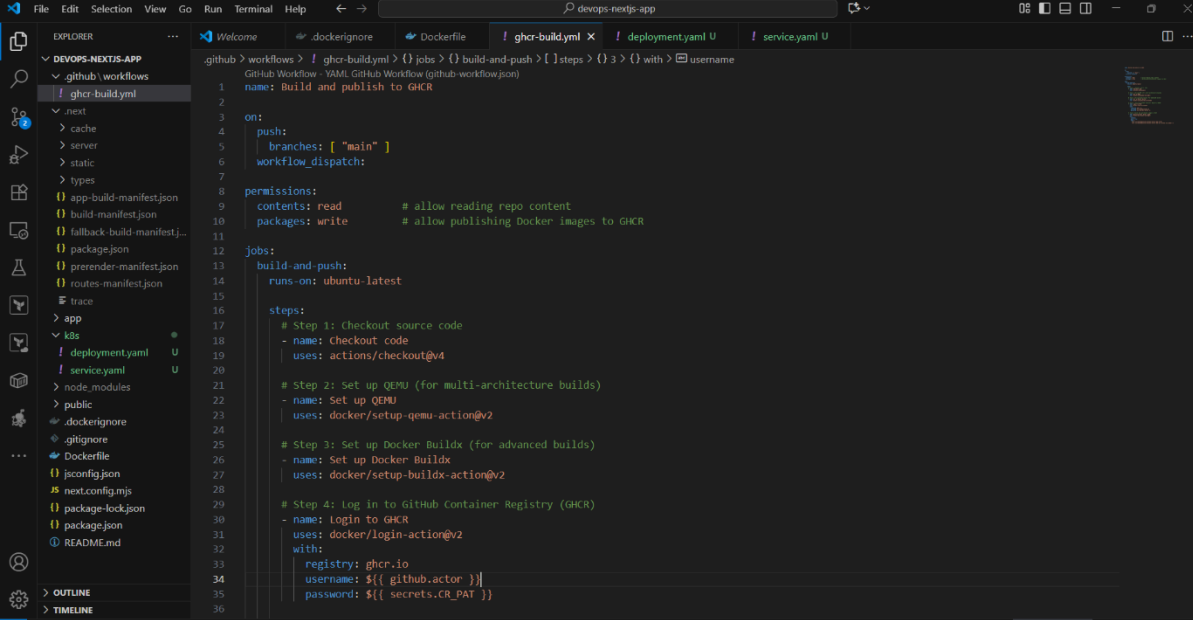
Add PAT to repo secrets:

- 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



The screenshot shows the Visual Studio Code editor with the file explorer on the left displaying the project structure. The main editor window shows the content of the `ghcr-build.yml` file, which is a GitHub Actions workflow. The workflow is named "Build and publish to GHCR" and runs on the `ubuntu-latest` runner. It includes four steps: 1. Checkout source code, 2. Set up QEMU for multi-architecture builds, 3. Set up Docker Buildx for advanced builds, and 4. Log in to GitHub Container Registry (GHCR). The workflow uses the `actions/checkout@v4`, `docker/setup-qemu-action@v2`, `docker/setup-buildx-action@v2`, and `docker/login-action@v2` actions.

```
name: Build and publish to GHCR

on:
  push:
    branches: [ "main" ]
  workflow_dispatch:

permissions:
  contents: read # allow reading repo content
  packages: write # allow publishing Docker images to GHCR

jobs:
  build-and-push:
    runs-on: ubuntu-latest

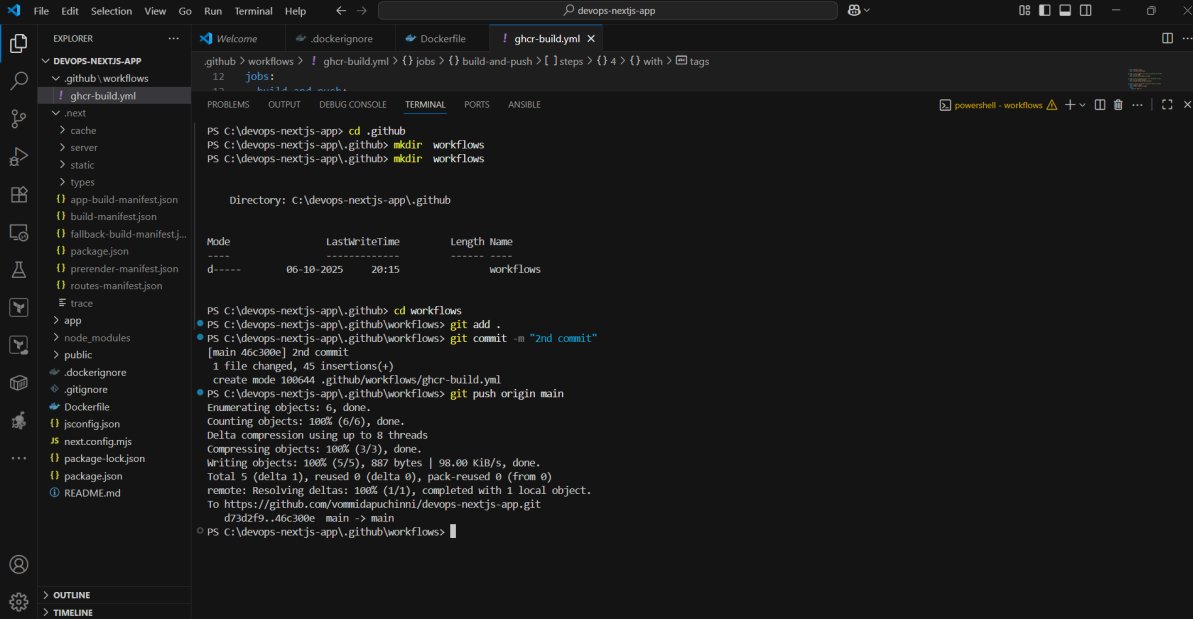
    steps:
      # Step 1: Checkout source code
      - name: Checkout code
        uses: actions/checkout@v4

      # Step 2: Set up QEMU (for multi-architecture builds)
      - name: Set up QEMU
        uses: docker/setup-qemu-action@v2

      # Step 3: Set up Docker Buildx (for advanced builds)
      - name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v2

      # Step 4: Log in to GitHub Container Registry (GHCR)
      - name: login to GHCR
        uses: docker/login-action@v2
        with:
          registry: ghcr.io
          username: ${github.actor}
          password: ${secrets.CR_PAT}
```

push files to the github repo



The screenshot shows the Visual Studio Code terminal with the following commands and output:

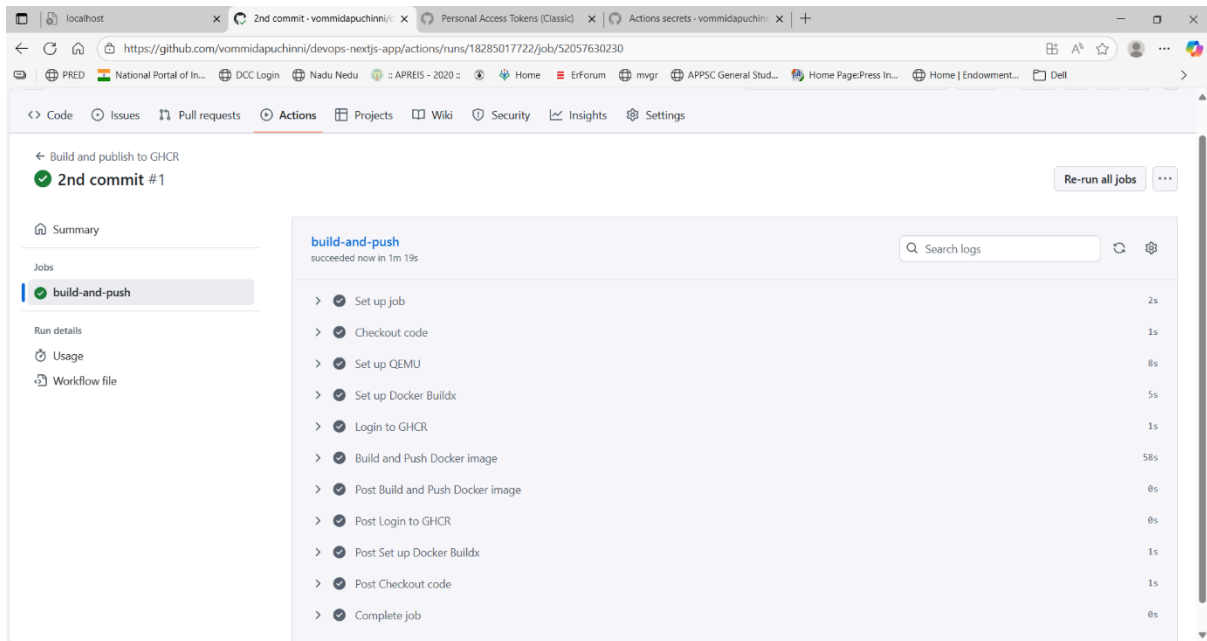
```
PS C:\devops-nextjs-app> cd .github
PS C:\devops-nextjs-app\github> mkdir workflows
PS C:\devops-nextjs-app\github> mkdir workflows

Directory: C:\devops-nextjs-app\github

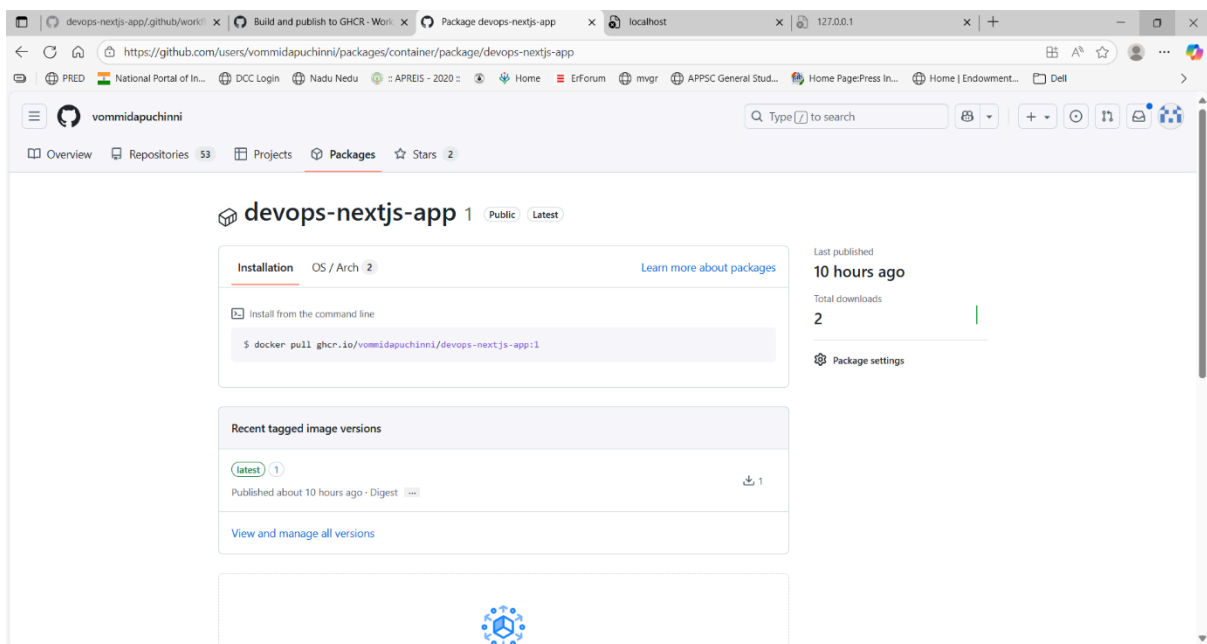
Mode                LastWriteTime         Length Name
----                -
d-----          06-10-2025   20:15         workflows

PS C:\devops-nextjs-app\github> cd workflows
PS C:\devops-nextjs-app\github\workflows> git add .
PS C:\devops-nextjs-app\github\workflows> git commit -m "2nd commit"
[main 46c300e] 2nd commit
1 file changed, 45 insertions(+)
create mode 100644 .github/workflows/ghcr-build.yml
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 887 bytes | 98.00 KiB/s, done.
Total 5 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/vommidapuchinni/devops-nextjs-app.git
d73d2f9..46c300e  main -> main
PS C:\devops-nextjs-app\github\workflows>
```

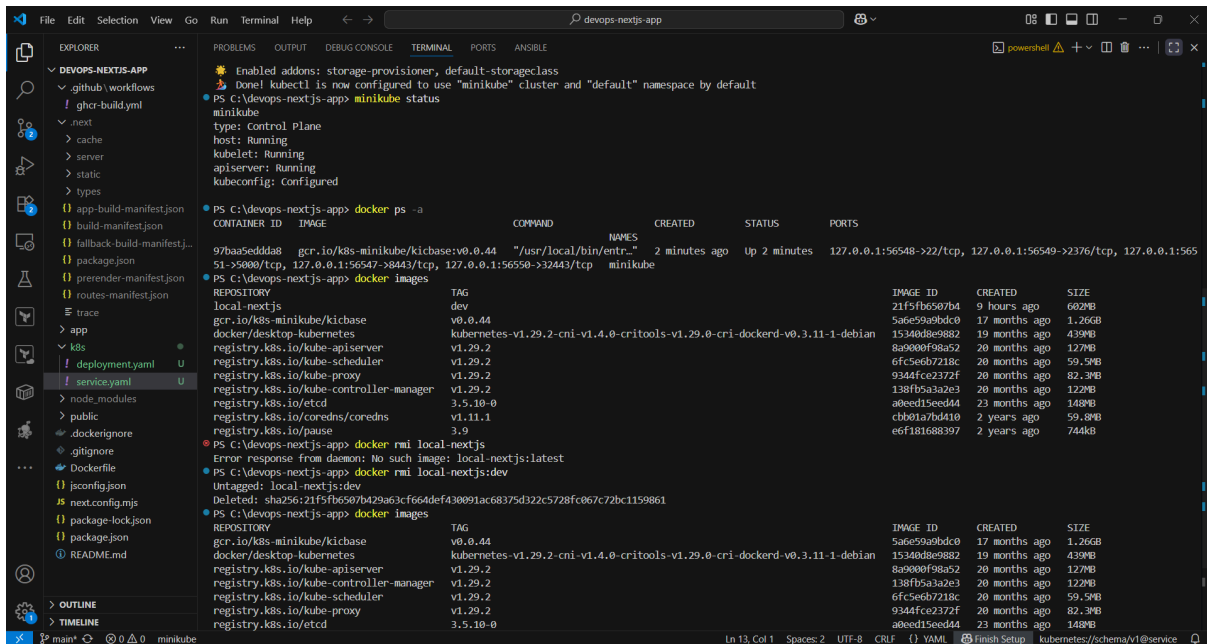
Run the workflow wait until job success



After job completion we see the ghcr image here.



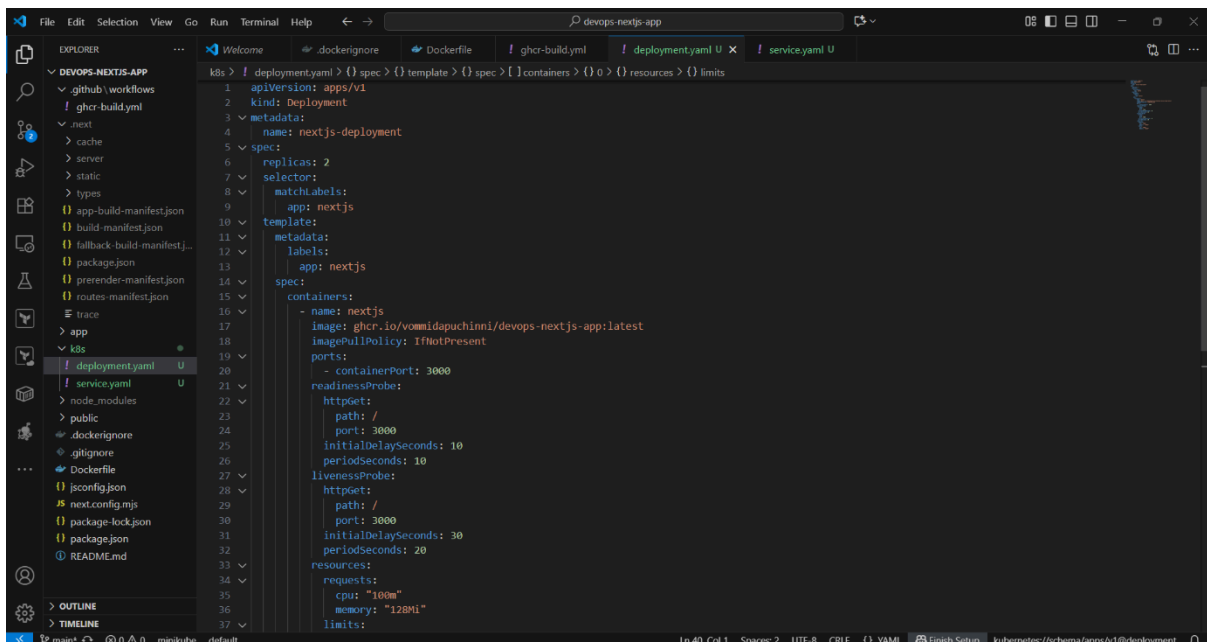
Start the minikube and before created docker images and containers delete those



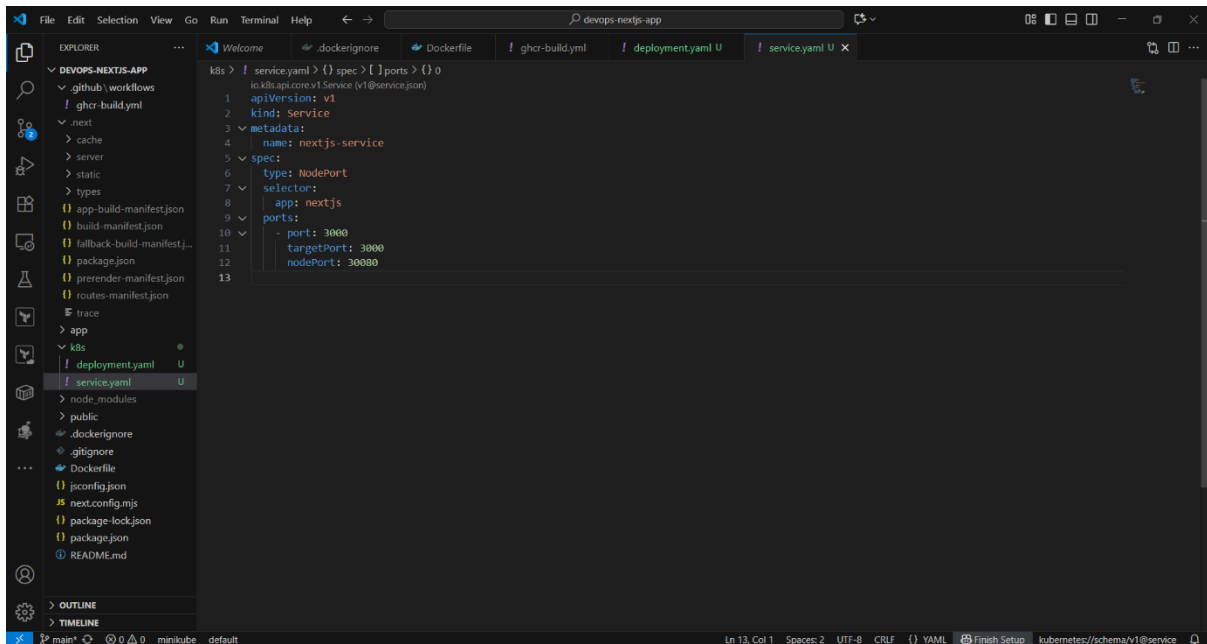
STEP 7: Create Kubernetes manifests (in k8s/)

Make folder k8s/ and create these two files.

k8s/deployment.yaml

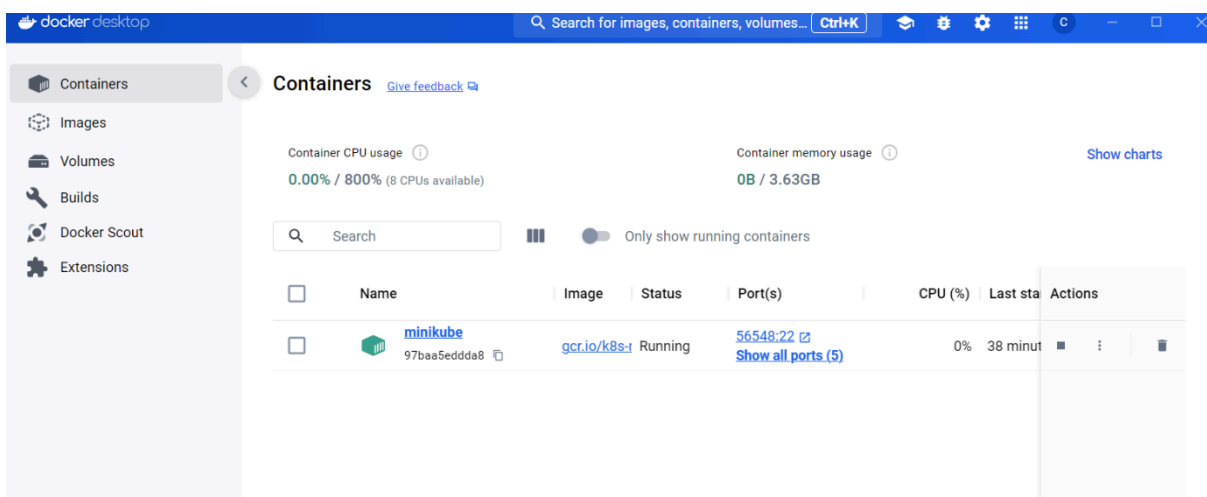


k8s/service.yaml



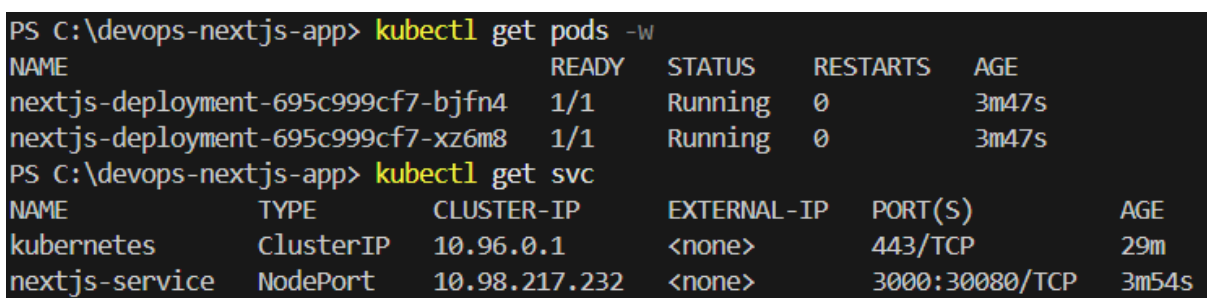
STEP 9: Deploy to Minikube

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



apply k8s manifests: kubectl apply -f k8s/

watch pods: kubectl get pods -w

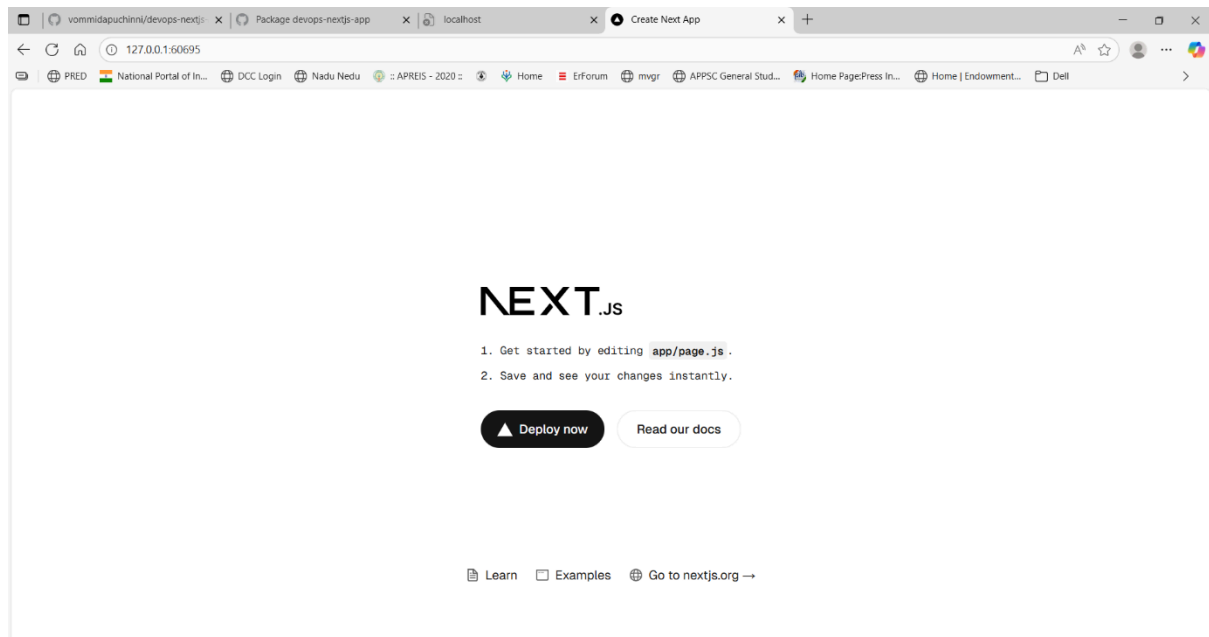


Then open service URL: minikube service nextjs-service --url

copy the URL and open it in browser

```
PS C:\devops-nextjs-app> minikube service nextjs-service --url
! Executing "docker container inspect minikube --format={{.State.Status}}" took an unusually long time: 2.1768378s
💡 Restarting the docker service may improve performance.
http://127.0.0.1:60695
! Because you are using a Docker driver on windows, the terminal needs to be open to run it.
```

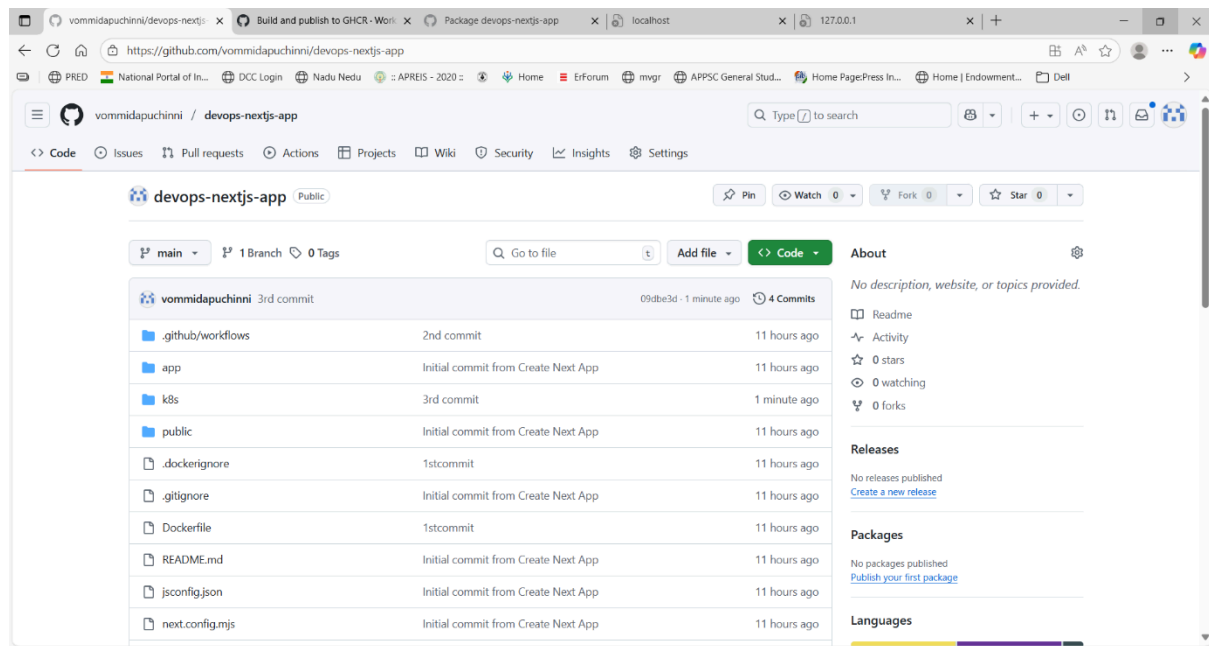
We can access the app by using this url provided here.



push codes to github.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\devops-nextjs-app> git add .
warning: in the working copy of 'package-lock.json', LF will be replaced by CRLF the next time Git touches it
PS C:\devops-nextjs-app> git commit -m "3rd commit"
[main 09dbe3d] 3rd commit
2 files changed, 51 insertions(+)
create mode 100644 k8s/deployment.yaml
create mode 100644 k8s/service.yaml
PS C:\devops-nextjs-app> git push origin main
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 807 bytes | 161.00 KiB/s, done.
Total 5 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/vommidapuchinni/devops-nextjs-app.git
46c300e..09dbe3d main -> main
PS C:\devops-nextjs-app>
```



terminate all after using

```
PS C:\devops-nextjs-app> minikube delete
🔥 Deleting "minikube" in docker ...
🔥 Deleting container "minikube" ...
🔥 Removing C:\Users\umama\.minikube\machines\minikube ...
💀 Removed all traces of the "minikube" cluster.
PS C:\devops-nextjs-app>
```

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.