# React App Deployment with Docker, CI/CD, and CloudWatch

## Repo link: vommidapuchinni/my-react-app

## Pages live link: https://vommidapuchinni.github.io/my-react-app/

### 1. Project Overview

This project demonstrates a full CI/CD workflow for deploying a React application to staging and production environments on AWS EC2 using Docker. It also includes log monitoring via Amazon CloudWatch.

**Key Features:**

- Separate Staging & Production environments on EC2.
- CI/CD Pipelines implemented using GitHub Actions.
- Docker used for containerized deployments.
- Nginx serves the React application.
- CloudWatch Agent collects logs from containers into AWS CloudWatch Logs.

### 2. Prerequisites

- AWS account with EC2 access.
- IAM role with CloudWatch permissions attached to EC2.
- GitHub repository with your React project.
- GitHub secrets:
  - STAGING_IP → Staging EC2 public IP
  - STAGING_KEY → Staging EC2 private key
  - EC2_IP → Production EC2 public IP
  - EC2_SSH_KEY → Production EC2 private key
  - Access and secrete access keys
- Docker and docker compose installed on EC2.

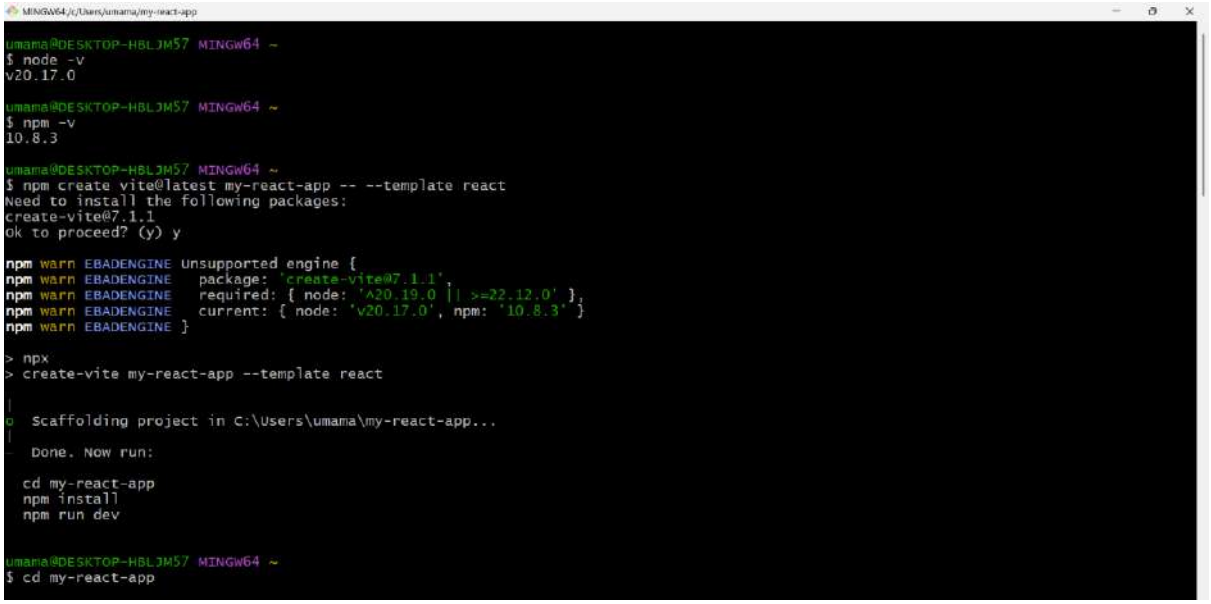
### 3. Create a ReactJS App

Open git bash and run commands

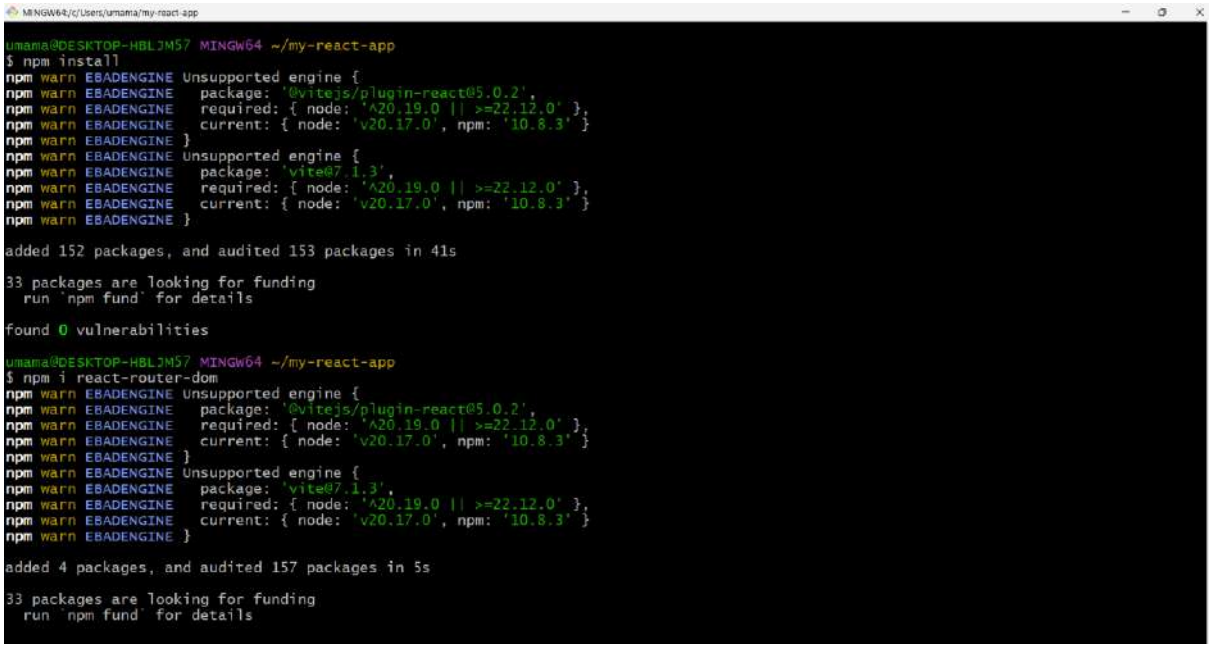
I recommend Vite (fast dev server)

npm create vite@latest my-react-app -- --template react

cd my-react-app




- npm install
- npm i react-router-dom




Add lint & test tools: npm install -D eslint vitest @testing-library/react @testing-library/jest-dom

Edit package.json scripts

```
umama@DESKTOP-HBLJM57 MINGW64 ~/my-react-app
$ npm install -D eslint vitest @testing-library/react @testing-library/jest-dom
npm warn EBADENGINE Unsupported engine {
npm warn EBADENGINE   package: '@vitejs/plugin-react@5.0.2',
npm warn EBADENGINE   required: { node: '^20.19.0 || >=22.12.0' },
npm warn EBADENGINE   current: { node: 'v20.17.0', npm: '10.8.3' }
npm warn EBADENGINE }
npm warn EBADENGINE Unsupported engine {
npm warn EBADENGINE   package: 'vite@7.1.3',
npm warn EBADENGINE   required: { node: '^20.19.0 || >=22.12.0' },
npm warn EBADENGINE   current: { node: 'v20.17.0', npm: '10.8.3' }
npm warn EBADENGINE }

added 54 packages, and audited 211 packages in 22s

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

found 0 vulnerabilities

umama@DESKTOP-HBLJM57 MINGW64 ~/my-react-app
$ ls
README.md  eslint.config.js  index.html  node_modules/  package-lock.json  package.json  public/  src/  vite.config.js

umama@DESKTOP-HBLJM57 MINGW64 ~/my-react-app
$ vi package.json

umama@DESKTOP-HBLJM57 MINGW64 ~/my-react-app
$ cat package.json
{
  "name": "my-react-app",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "lint": "eslint .",
```

Create these files: src/main.jsx, src/App.jsx, src/components/Header.jsx, src/components/Card.jsx, src/components/Footer.jsx, src/pages/Home.jsx, src/pages/About.jsx, src/pages/Blog.jsx,

We now have at least **three reusable components** (Header, Card, Footer) plus **routing** (react-router-dom).

Local test:

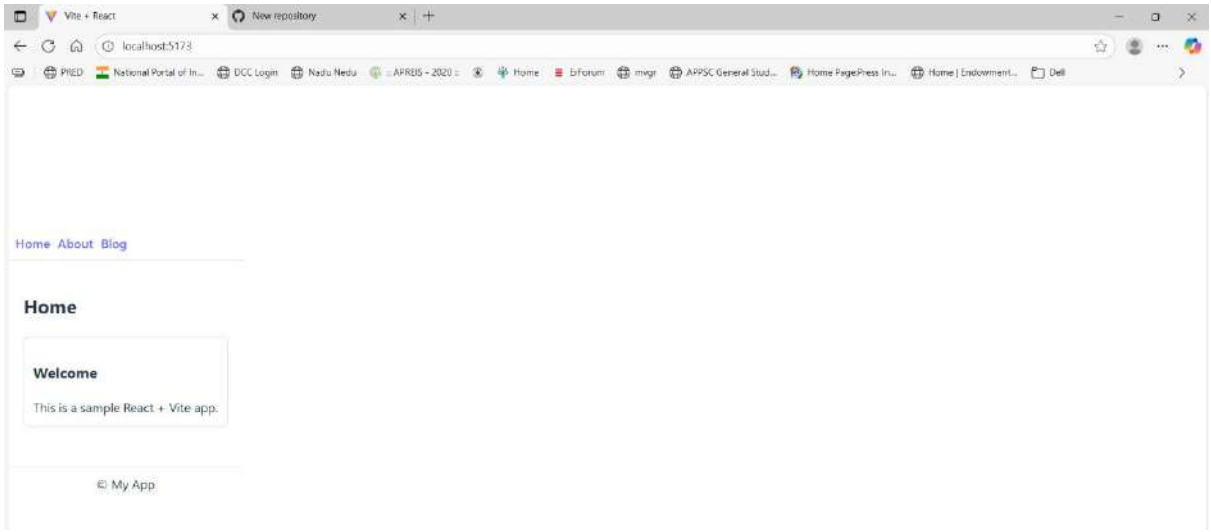
npm run dev  # open http://localhost:5173

npm run build  # creates dist/ (Vite's build folder)

```
umama@DESKTOP-HBLJM57 MINGW64 ~/my-react-app
$ npm run dev

> my-react-app@0.0.0 dev
> vite

You are using Node.js 20.17.0. Vite requires Node.js version 20.19+ or 22.12+. Please upgrade your Node.js version.

  VITE v7.1.3  ready in 763 ms

  ➜  Local:   http://localhost:5173/
  ➜  Network: use --host to expose
  ➜  press h + enter to show help
^Z^V

umama@DESKTOP-HBLJM57 MINGW64 ~/my-react-app
$ ls
README.md  eslint.config.js  index.html  node_modules/  package-lock.json  package.json  public/  src/  vite.config.js

umama@DESKTOP-HBLJM57 MINGW64 ~/my-react-app
$ cd src/

umama@DESKTOP-HBLJM57 MINGW64 ~/my-react-app/src
$ cat App.jsx
import React from "react";
import { Routes, Route, Link } from "react-router-dom";
import Home from "./pages/Home";
import About from "./pages/About";
import Blog from "./pages/Blog";
import Header from "./components/Header";
import Footer from "./components/Footer";

export default function App(){
  return (
    <>
      <Header />
      <main style={{padding:20}}>
        <Routes>
          <Route path="/" element={<Home/>} />
```

```
umama@DESKTOP-HBLJM57 MINGW64 ~/my-react-app
$ npm run build

> my-react-app@0.0.0 build
> vite build

You are using Node.js 20.17.0. Vite requires Node.js version 20.19+ or 22.12+. Please upgrade your Node.js version.
vite v7.1.3 building for production...
✓ 47 modules transformed.
dist/index.html                   0.46 kB │ gzip:  0.30 kB
dist/assets/index-Dtn62Xmo.css    0.91 kB │ gzip:  0.49 kB
dist/assets/index-BMOMM_uG.js   220.73 kB │ gzip: 70.96 kB
✓ built in 2.92s
```

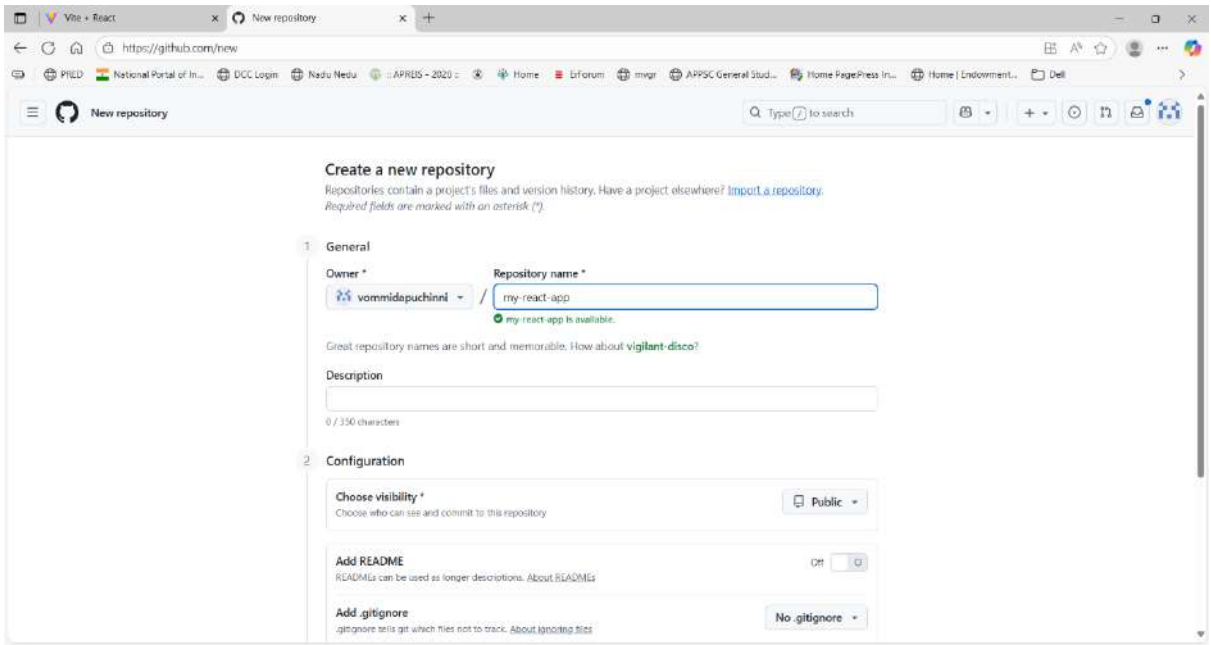
Accessing the file locally




On GitHub: click New repository (top-right, + → New repository).

Name it my-react-app, set Public, don't initialize with README (we already have local files). Click Create repository.

In our project folder (local)




git init

git add .

git commit -m "1st commit"

```
MINGW64:/c/Users/umama/my-react-app
umama@DESKTOP-HBLJM57 MINGW64 ~/my-react-app
$ git init
Initialized empty Git repository in C:/Users/umama/my-react-app/.git/

umama@DESKTOP-HBLJM57 MINGW64 ~/my-react-app (main)
$ git add .
warning: in the working copy of '.gitignore', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'README.md', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'eslint.config.js', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'index.html', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'package-lock.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'package.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/App.css', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/App.jsx', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/Card.jsx', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/Footer.jsx', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/components/Header.jsx', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/index.css', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/main.jsx', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/pages/About.jsx', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/pages/Blog.jsx', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/pages/Home.jsx', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'vite.config.js', LF will be replaced by CRLF the next time Git touches it

umama@DESKTOP-HBLJM57 MINGW64 ~/my-react-app (main)
$ git commit -m "1st commit"
[main (root-commit) 822a6c7] 1st commit
 19 files changed, 3861 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 README.md
 create mode 100644 eslint.config.js
 create mode 100644 index.html
 create mode 100644 package-lock.json
 create mode 100644 package.json
 create mode 100644 public/vite.svg
 create mode 100644 src/App.css
 create mode 100644 src/App.jsx
 create mode 100644 src/assets/react.svg
 create mode 100644 src/components/Card.jsx
```

git remote add origin https://github.com/vommidapuchinni/my-react-app.git

git push origin main

```
umama@DESKTOP-HBLJM57 MINGW64 ~/my-react-app (main)
$ git remote add origin https://github.com/vommidapuchinni/my-react-app.git

umama@DESKTOP-HBLJM57 MINGW64 ~/my-react-app (main)
$ git push origin main
Enumerating objects: 26, done.
Counting objects: 100% (26/26), done.
Delta compression using up to 8 threads
Compressing objects: 100% (24/24), done.
Writing objects: 100% (26/26), 35.93 KiB | 1.80 MiB/s, done.
Total 26 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/vommidapuchinni/my-react-app.git
 * [new branch]      main -> main

umama@DESKTOP-HBLJM57 MINGW64 ~/my-react-app (main)
$
```

**Create an AWS EC2 Linux server (exact clicks) and security group**

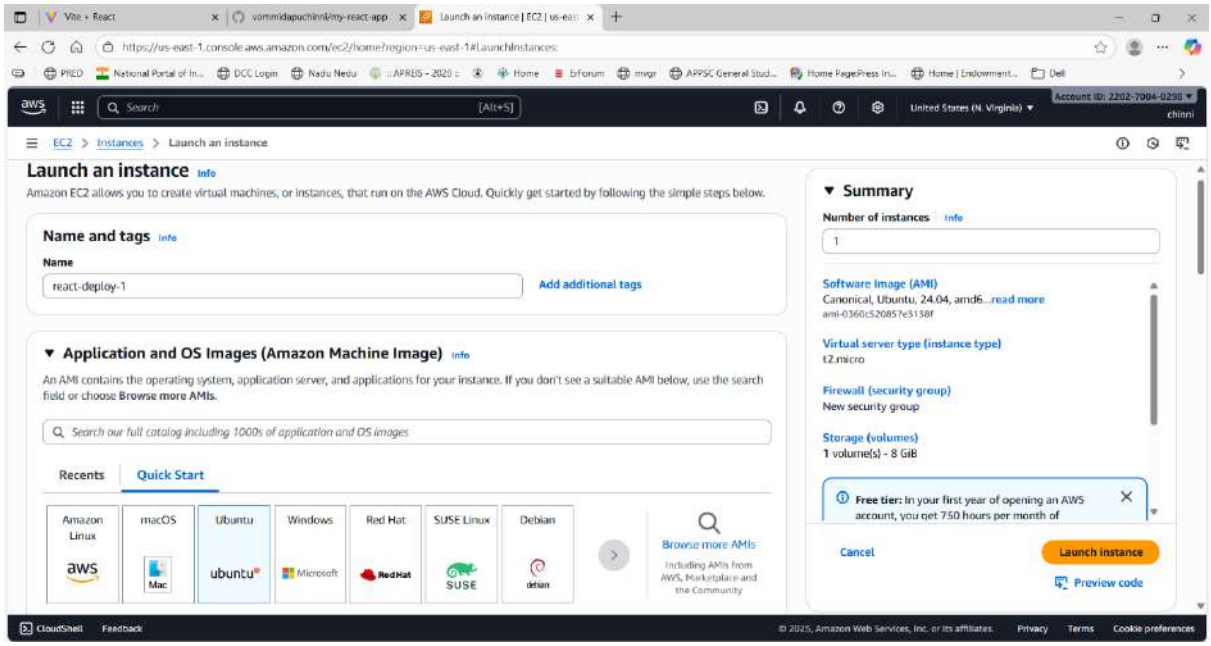
**Goal:** create an Ubuntu instance reachable by SSH (port 22) and HTTP (port 80).

Sign in to AWS Console → top search box type **EC2** → click **EC2** (or from Services → Compute → EC2).

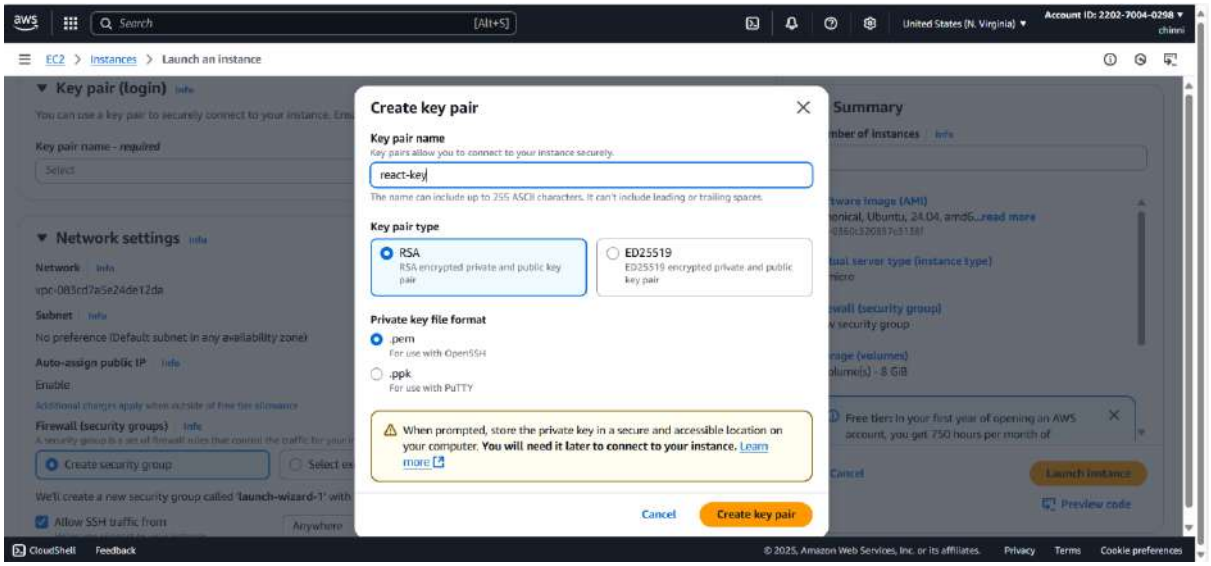
In EC2 Dashboard → left sidebar → **Instances** → **Launch instances** (button).

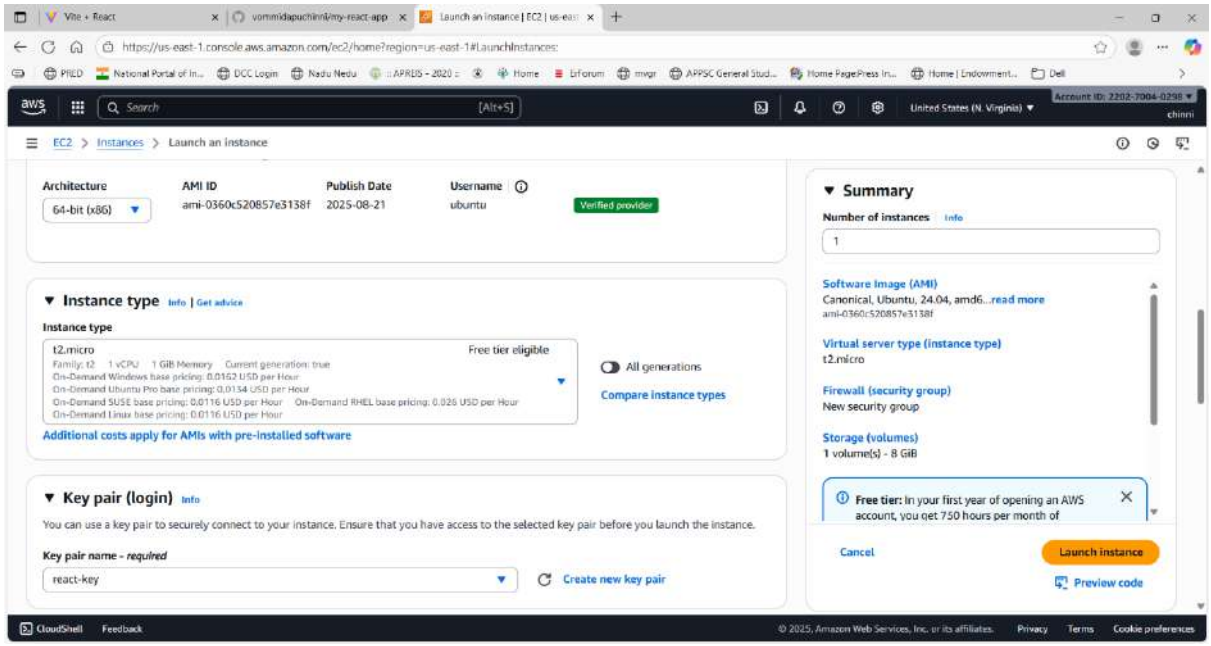
Launch instance wizard:

- Name: react-deploy-1
- AMI: choose Ubuntu Server 22.04 LTS (HVM), SSD (or 20.04 LTS)

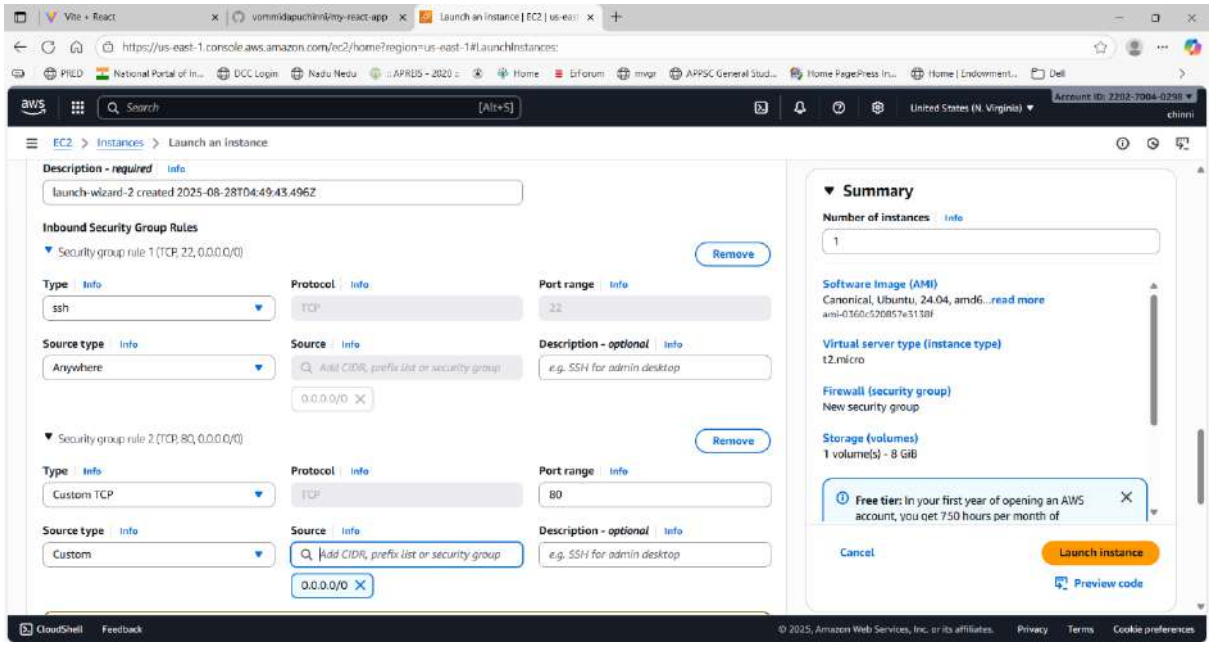


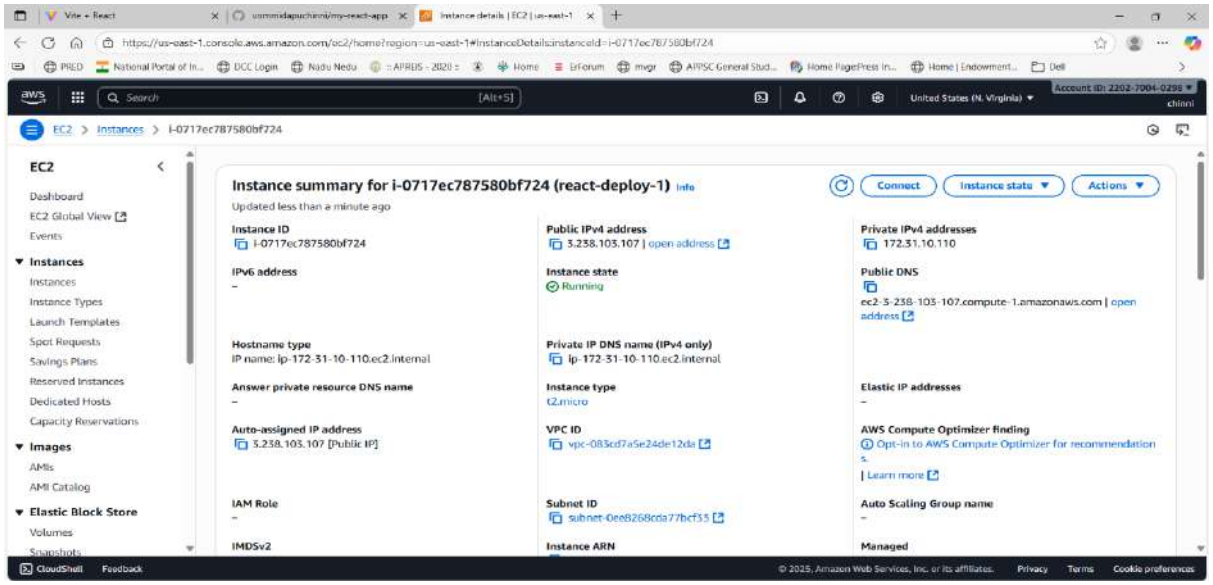

- Instance type: t2.micro (free tier eligible)
- Key pair (login): Create a new key pair → name it react-key → Download .pem file and keep it safe (you'll use it to SSH).



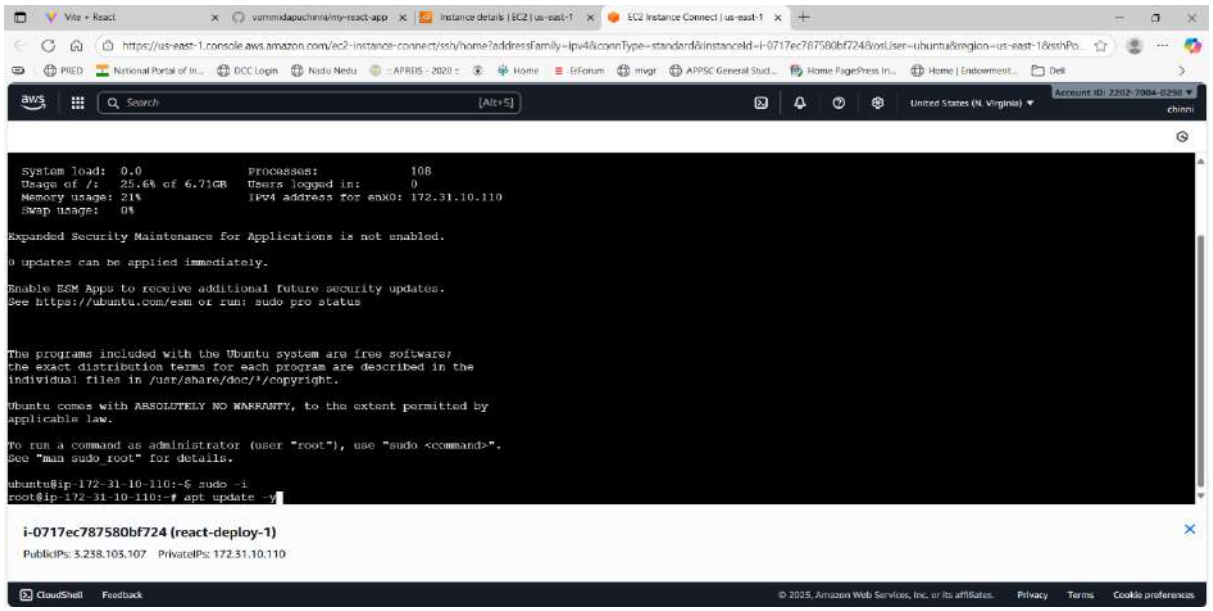


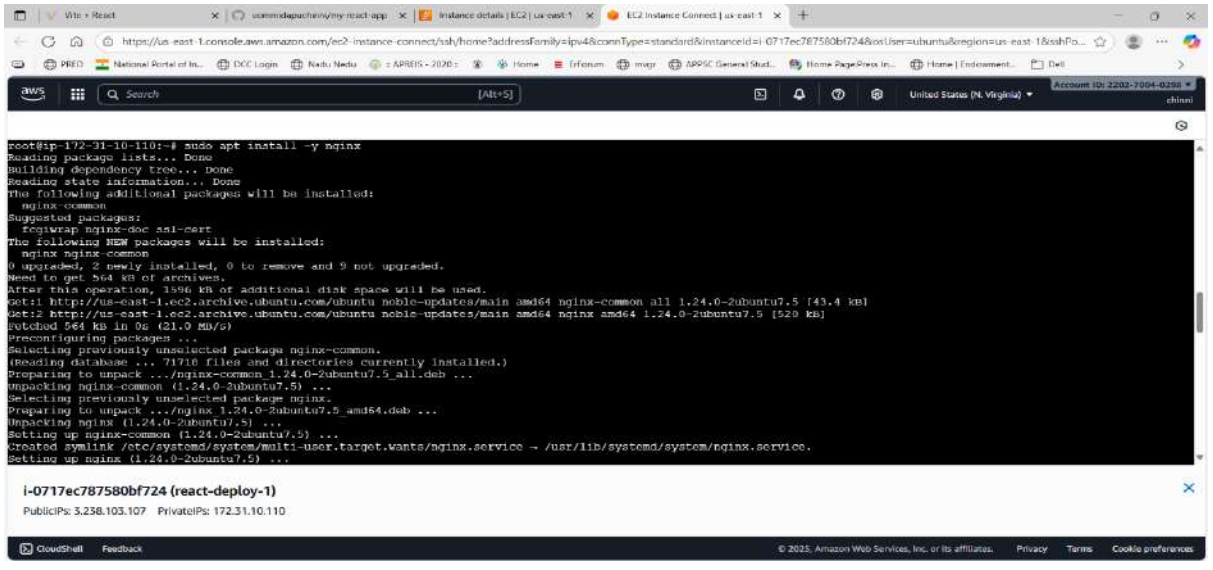

- Network settings (Security group): Create new security group with rules:
  - SSH (port 22) — Source: My IP (recommended) or your IP range
  - HTTP (port 80) — Source: Anywhere (0.0.0.0/0) (to allow web traffic)
  - Allow port 3000
- Click Launch instance.

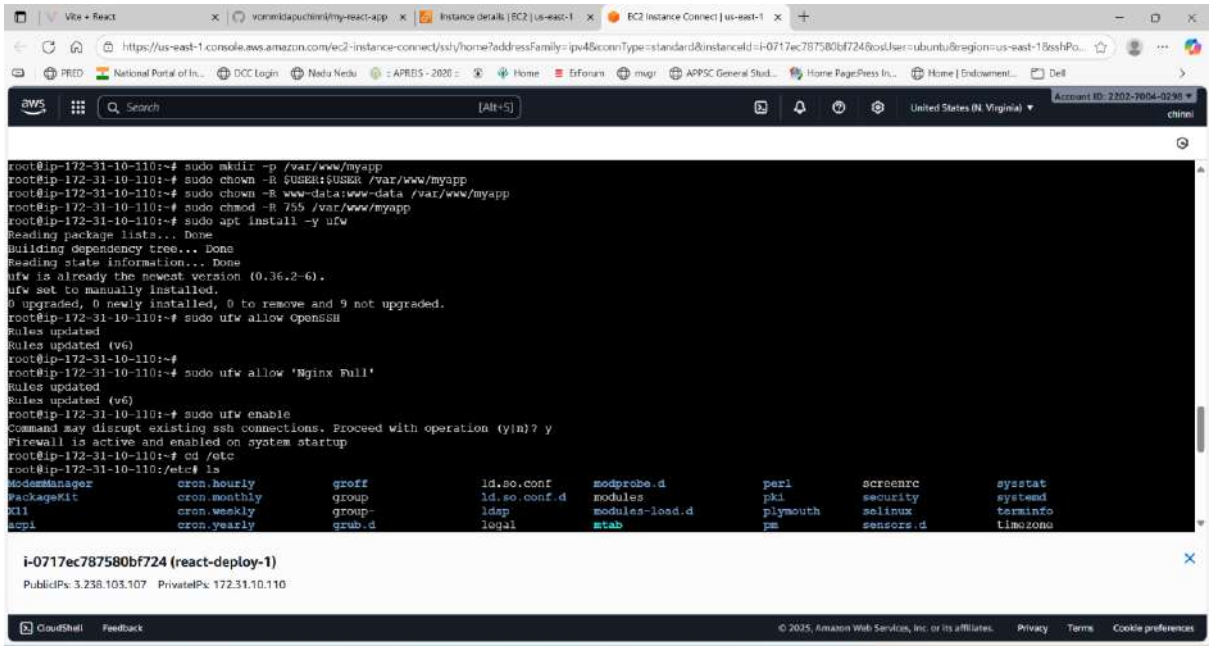


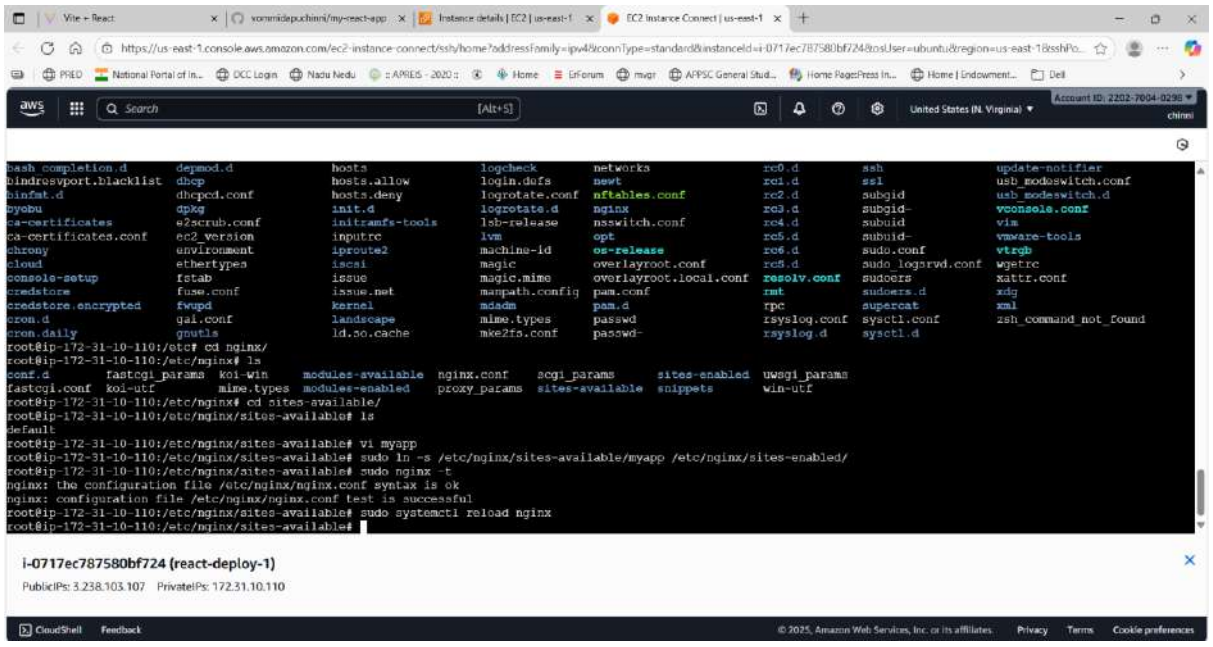



Prepare the server



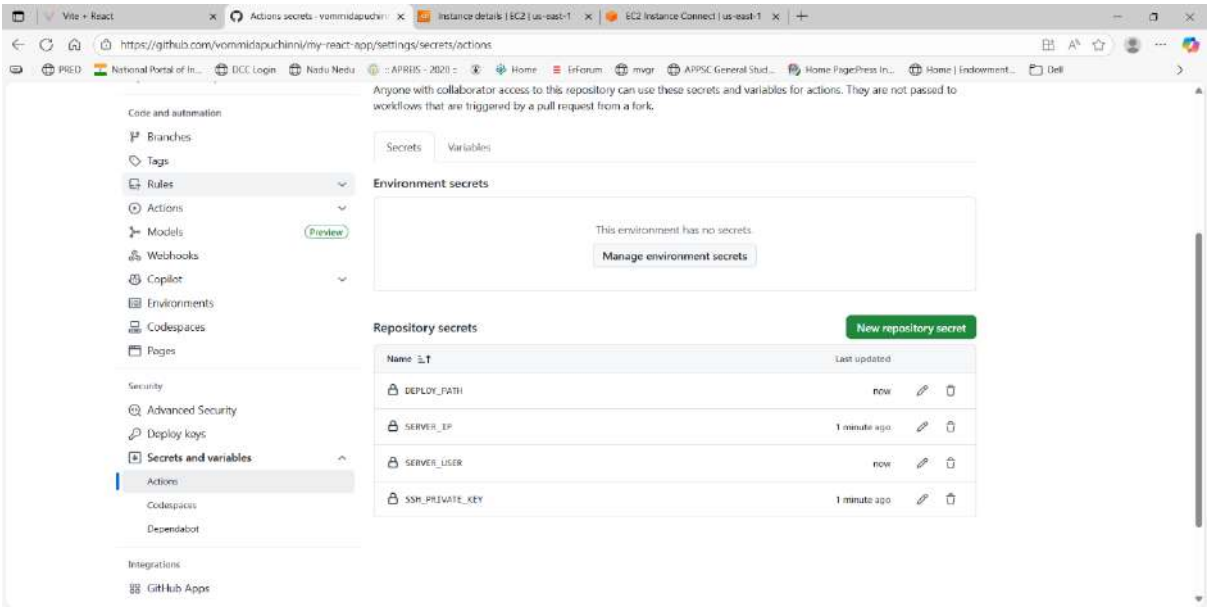






**On GitHub (UI steps)**:

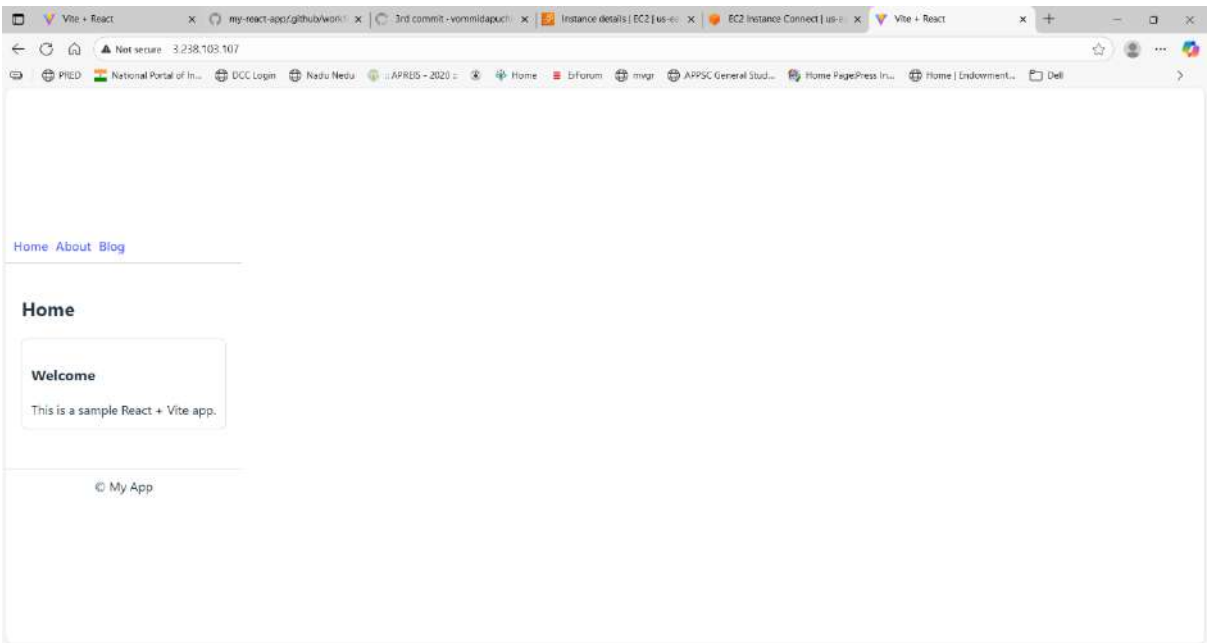
- Open your repo → Settings → Secrets and variables → Actions → New repository secret.
- Add secret SSH_PRIVATE_KEY — paste the content of downloaded key
- Add secret SERVER_IP — your EC2 IP.
- Add secret SERVER_USER — ubuntu (for Ubuntu AMI).
- Add secret DEPLOY_PATH — /var/www/myapp

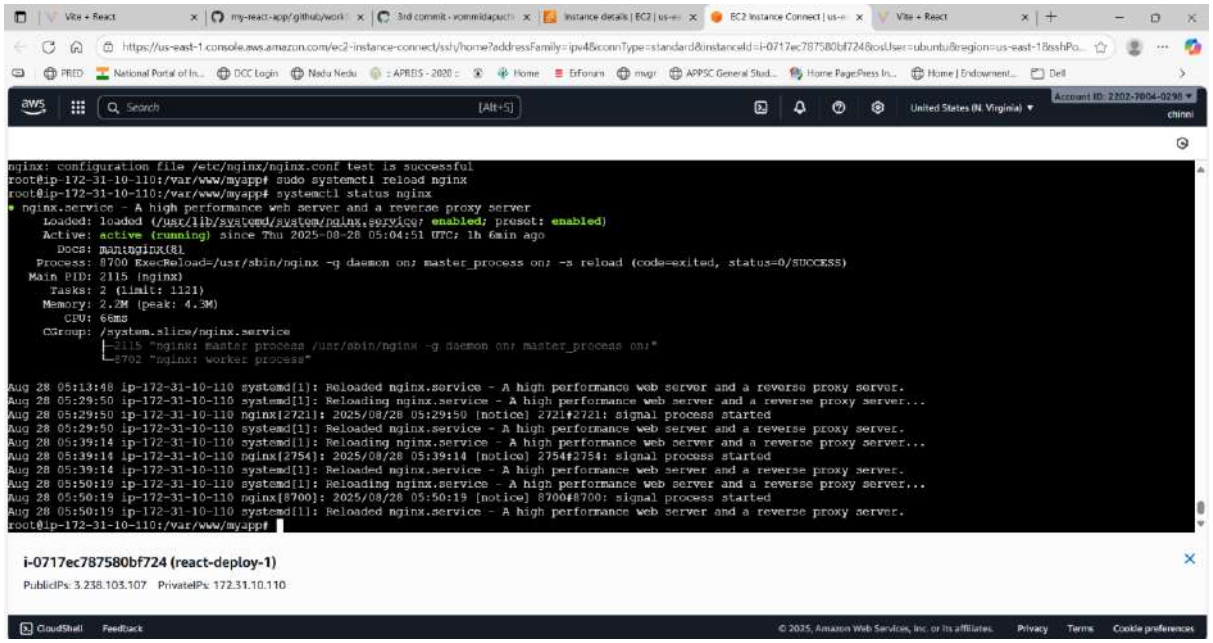


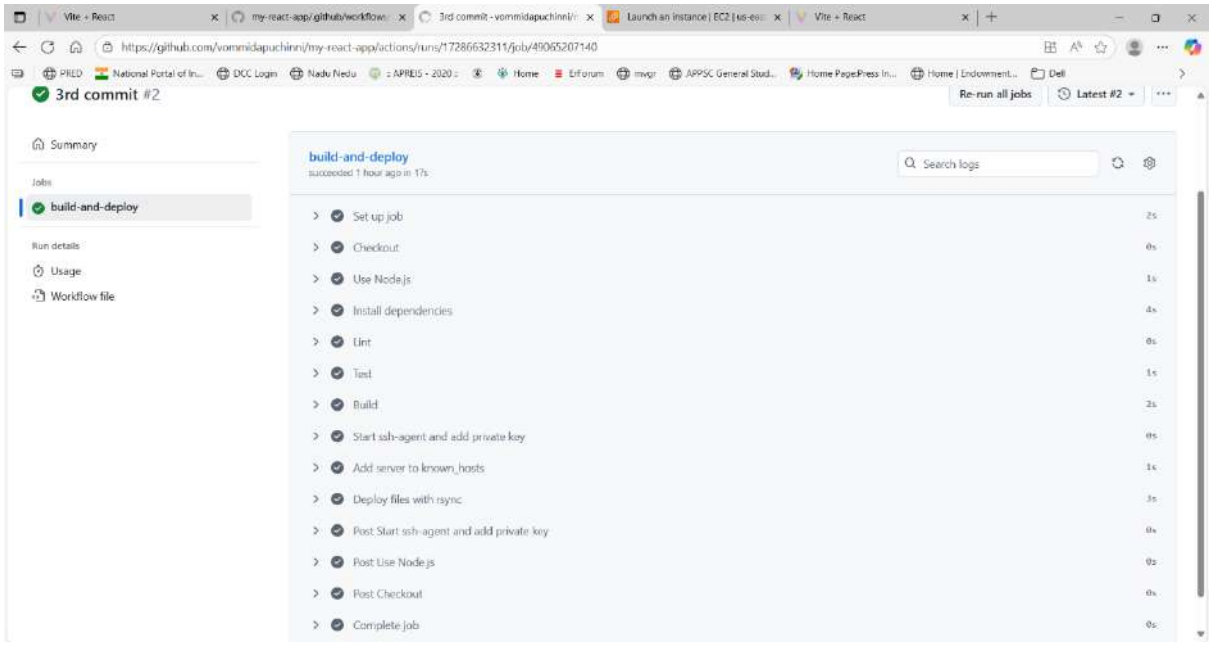

**Create deploy.yaml file and pull to github**

**Explanation of github CI/CD:**

- npm ci installs deterministically.
- npm run lint and npm run test are your CI quality gates.
- npm run build produces static assets (dist/ for Vite, build/ for CRA). The workflow checks for both.
- webfactory/ssh-agent loads the SSH_PRIVATE_KEY so rsync/ssh can connect to your EC2.
- rsync --delete synchronizes the build folder to the server folder and deletes removed files.
- After copying, we chown to www-data (nginx user) and reload nginx.






**Dockerize React App + Deploy via CI/CD**

**Create Dockerfile for React App**

In our React project root, create a file named **Dockerfile**:

- Node builds the React app → produces static files in build/.
- Nginx serves those static files efficiently.

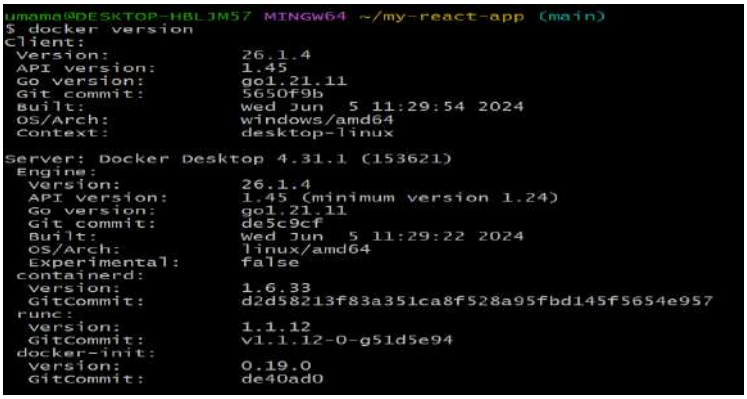
**Create docker-compose.yml**

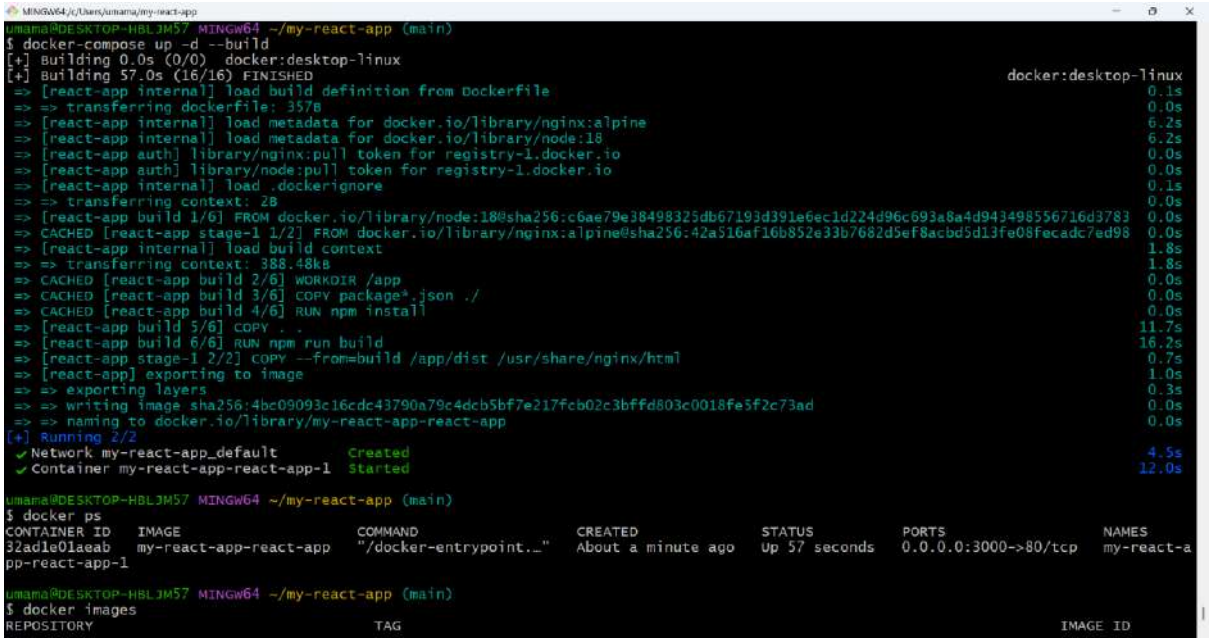
If you want easier management and future scaling

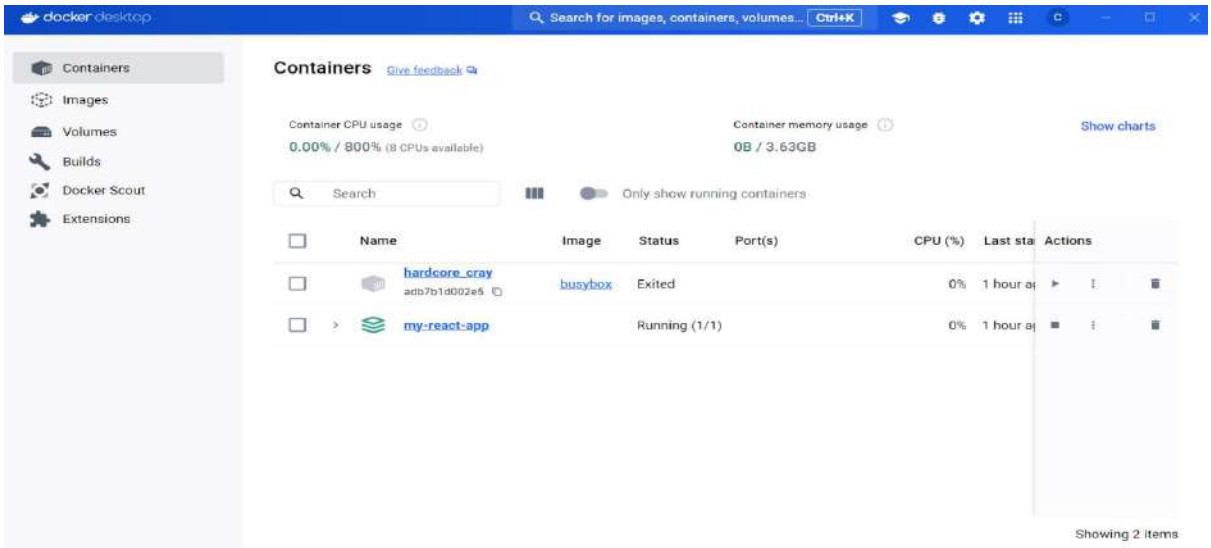
Build Docker Image locally: docker-compose up -d –build

- Docker will build and copy /app/dist to Nginx.
- After it finishes, check containers:

For locally createing we need to connect to docker desktop. Check docker version.



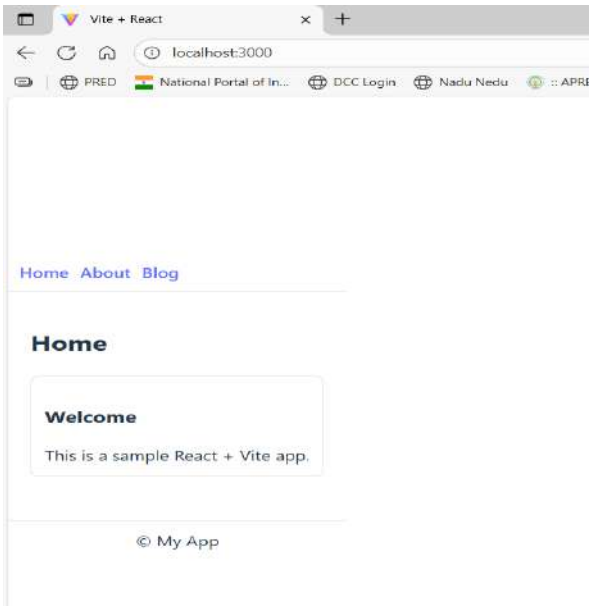



docker ps we see container is running

**Test React App**

Open browser → http://localhost:3000 → React app should load.

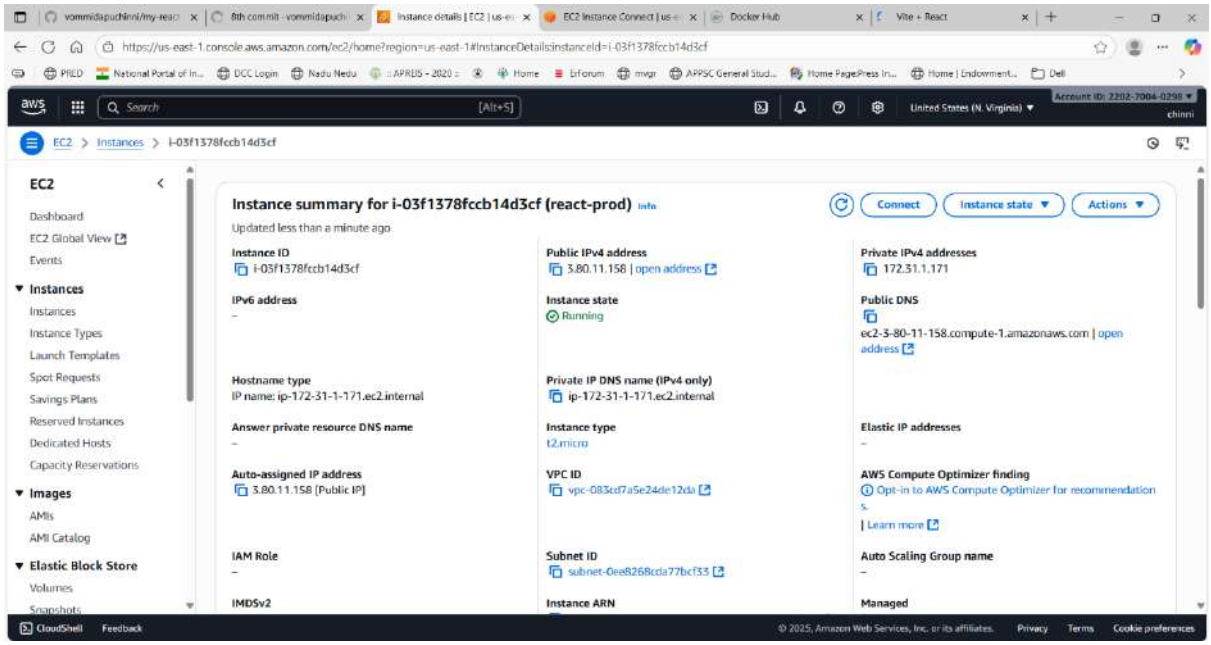



**Production environment:**

For pushing and pulling to ec2 in aws. Before we created ec2 that is used for production environment. Before we deployed that will be removed

In that instance install docker and docker-compose

- Update packages: sudo apt update && sudo apt upgrade -y
- Install Docker: sudo apt install -y docker.io
- Start Docker and enable at boot: sudo systemctl start docker& sudo systemctl enable docker
- Add your user to Docker group (optional, allows running docker without sudo):sudo usermod -aG docker $USER
- Install Docker Compose: sudo apt install -y docker-compose




Go to your GitHub repo → **Settings → Secrets → Actions → New repository secret**

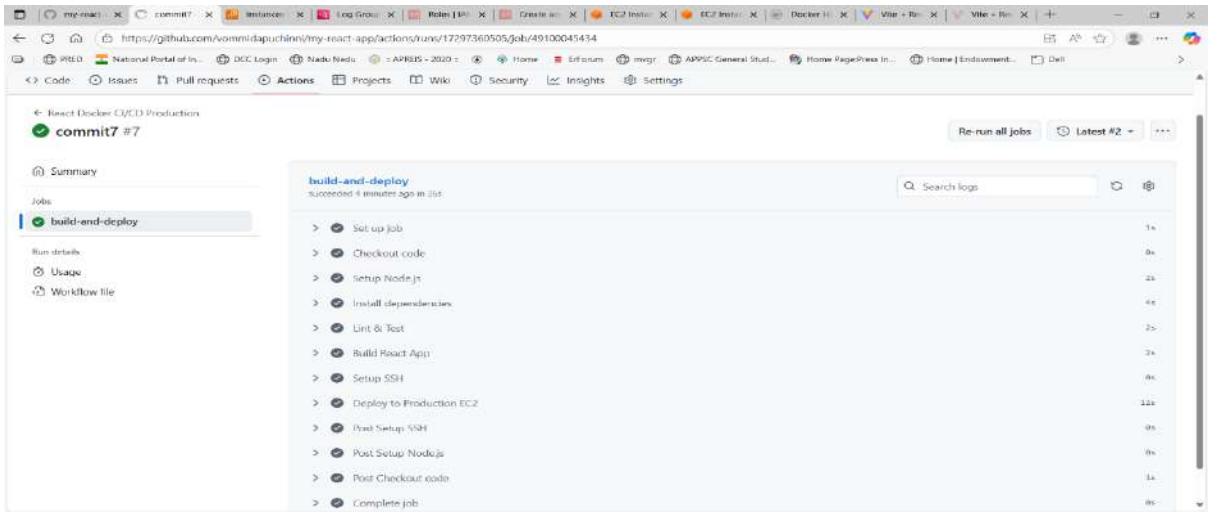
Name it, for example: EC2_IP

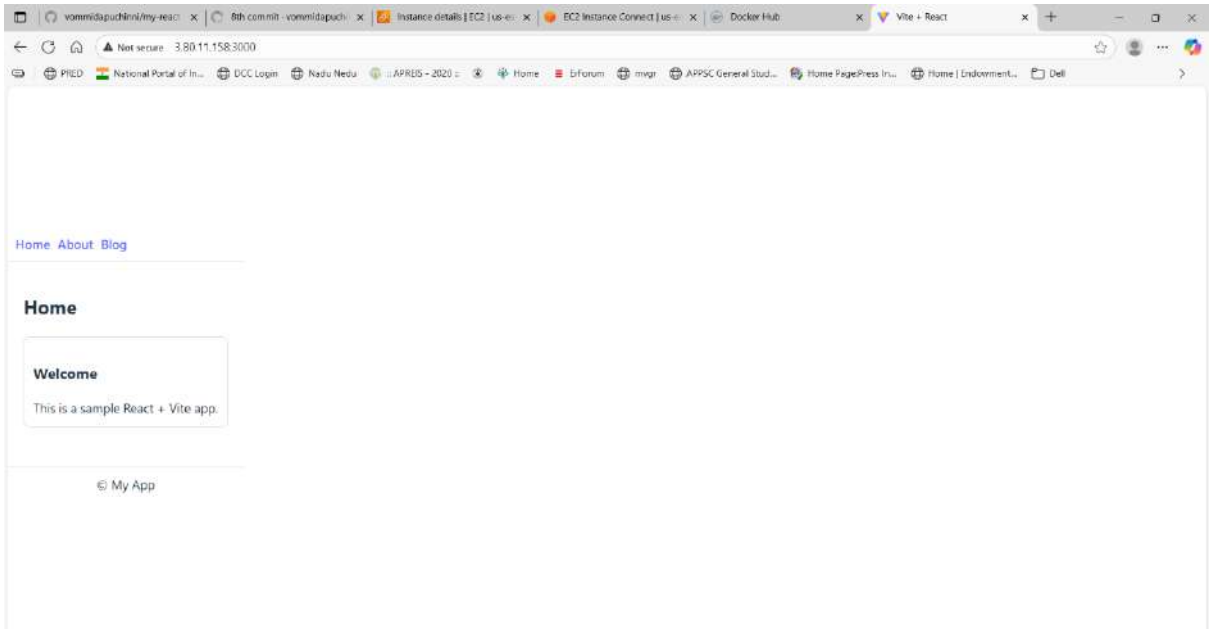
Value: your EC2 public IP (e.g., 3.80.11.158)

Save the secret

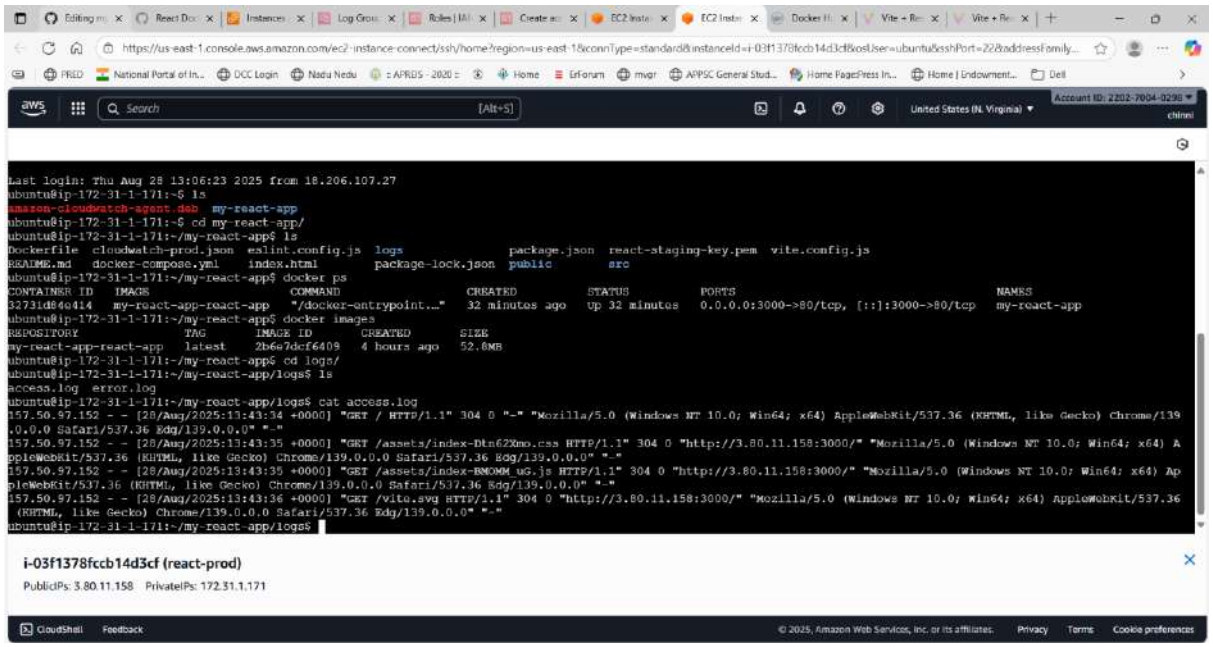
**Created deploy-prod.yaml file and push to github**

- Push this updated workflow to GitHub.
- Trigger a push to main.
- The workflow will:
- SSH to EC2
- Remove any old folder
- Clone repo fresh
- Run Docker → container starts
- Your app will be live at http://EC2_IP:3000






We can see the container running in ec2




**Staging environment:**

For this env we create separate branch

Push Staging Branch to GitHub: git checkout -b staging

We use separate docker-compose.yaml and deploy-staging.yaml files

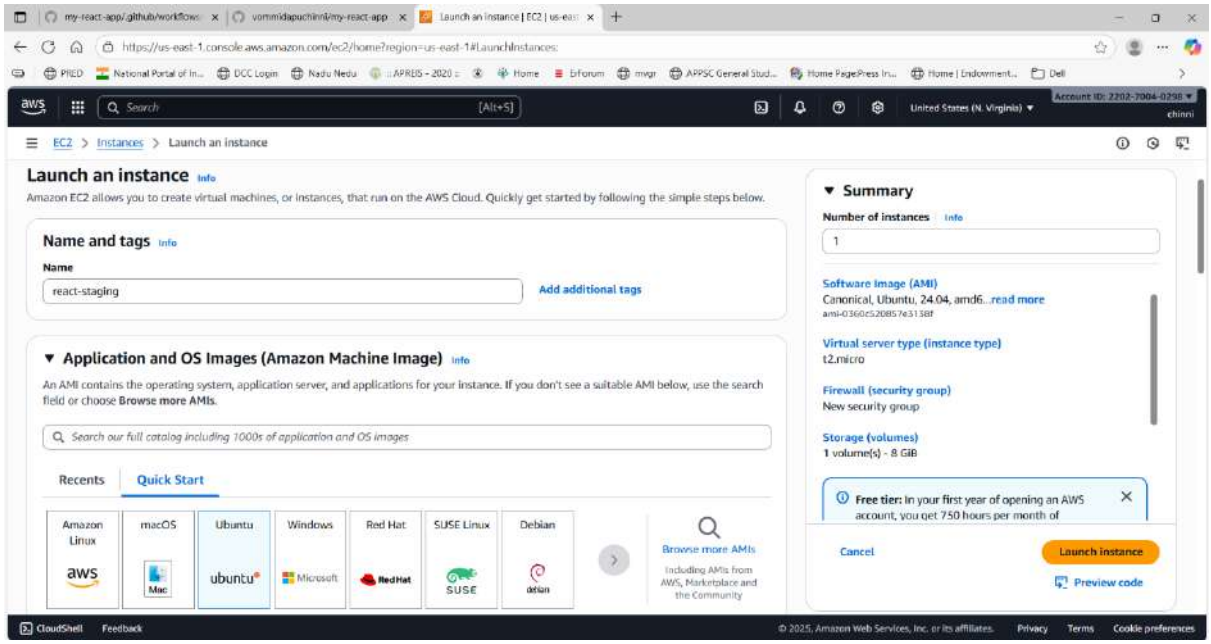
Staging docker-compose-staging.yml (port 4000 → 80 inside container)

For this created another EC2 instance

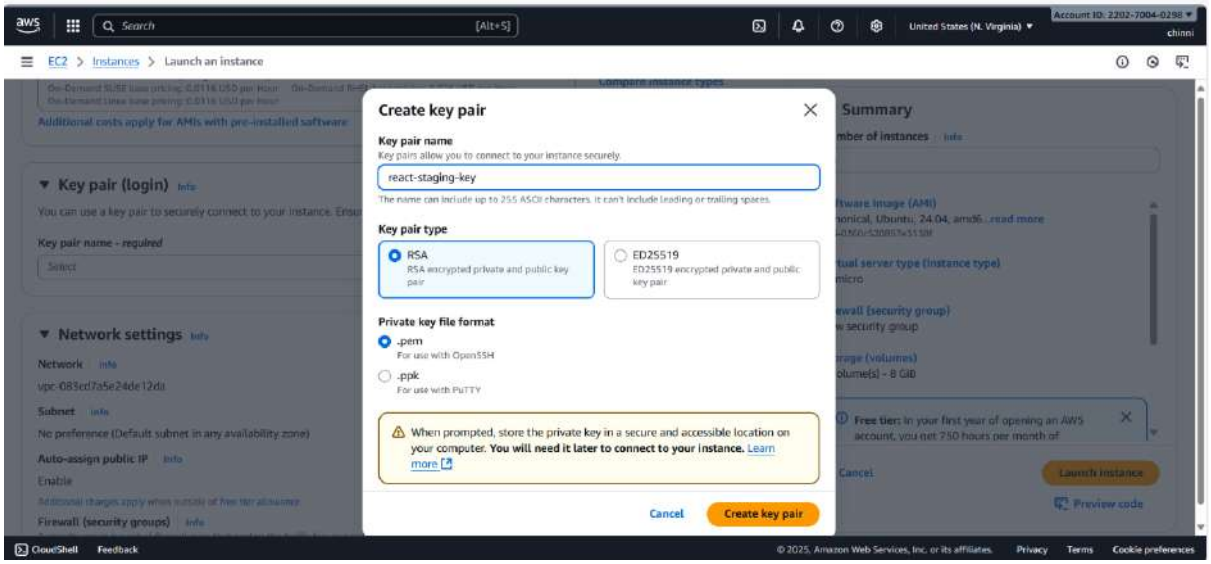
In EC2 Dashboard → left sidebar → **Instances** → **Launch instances** (button).

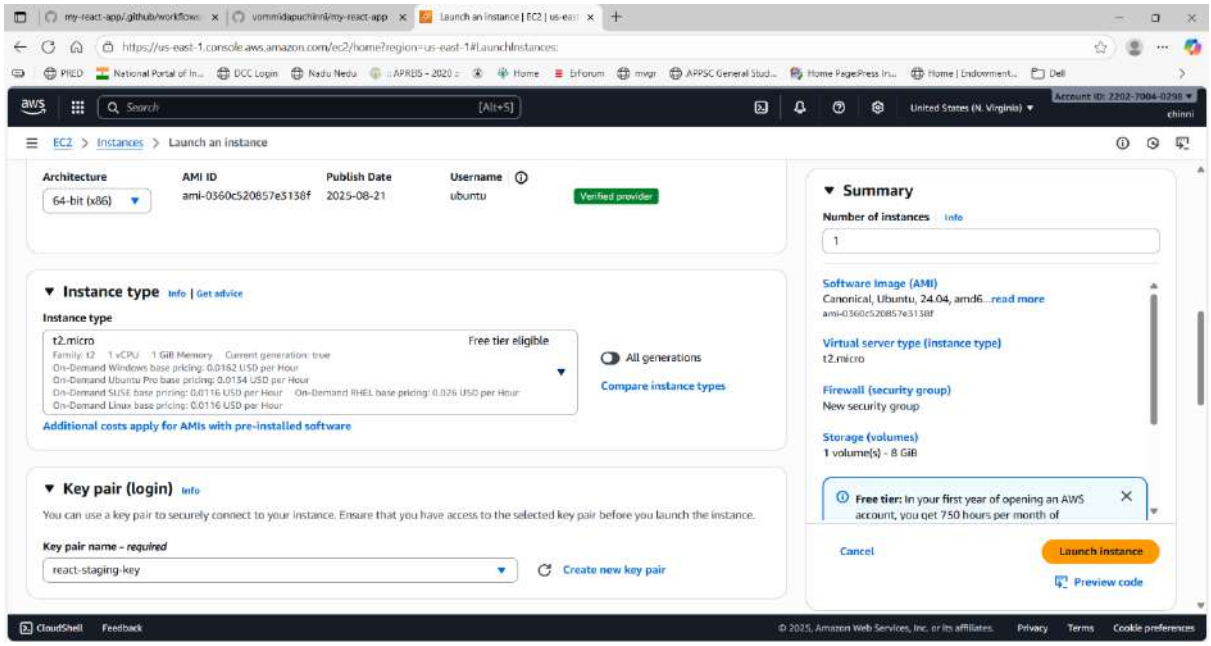
Launch instance wizard:

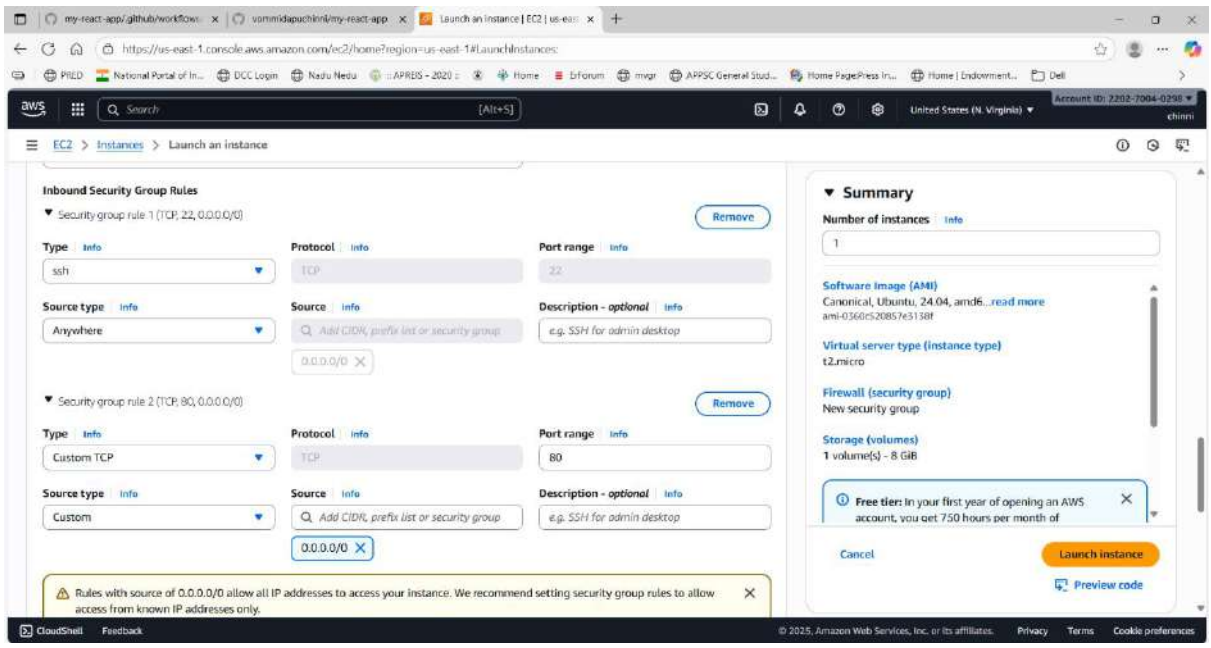
- Name: react-staging
- AMI: choose Ubuntu Server 22.04 LTS (HVM), SSD (or 20.04 LTS)



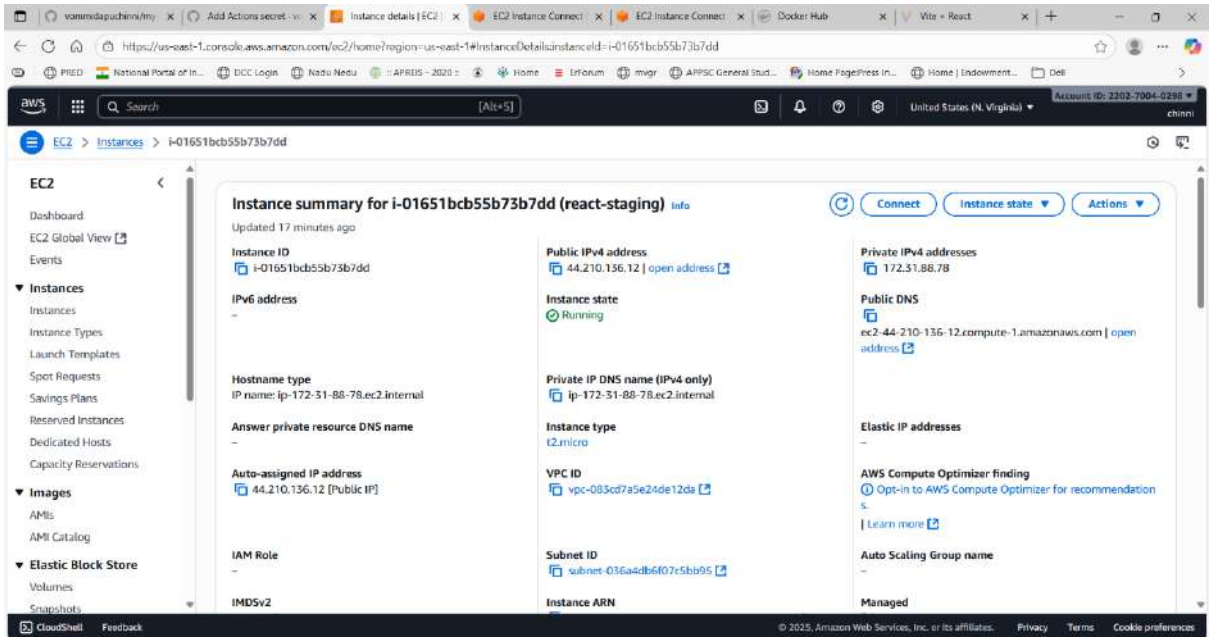

- Instance type: t2.micro (free tier eligible)
- Key pair (login): Create a new key pair → name it react-staging-key → Download .pem file and keep it safe (you'll use it to SSH).





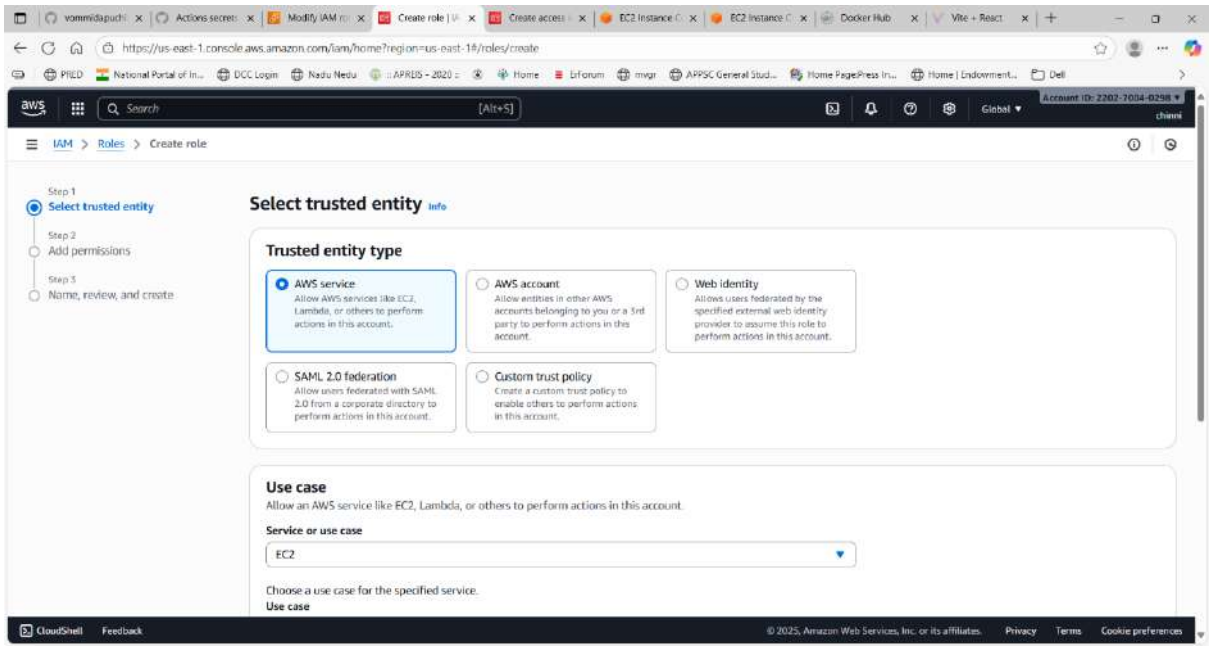



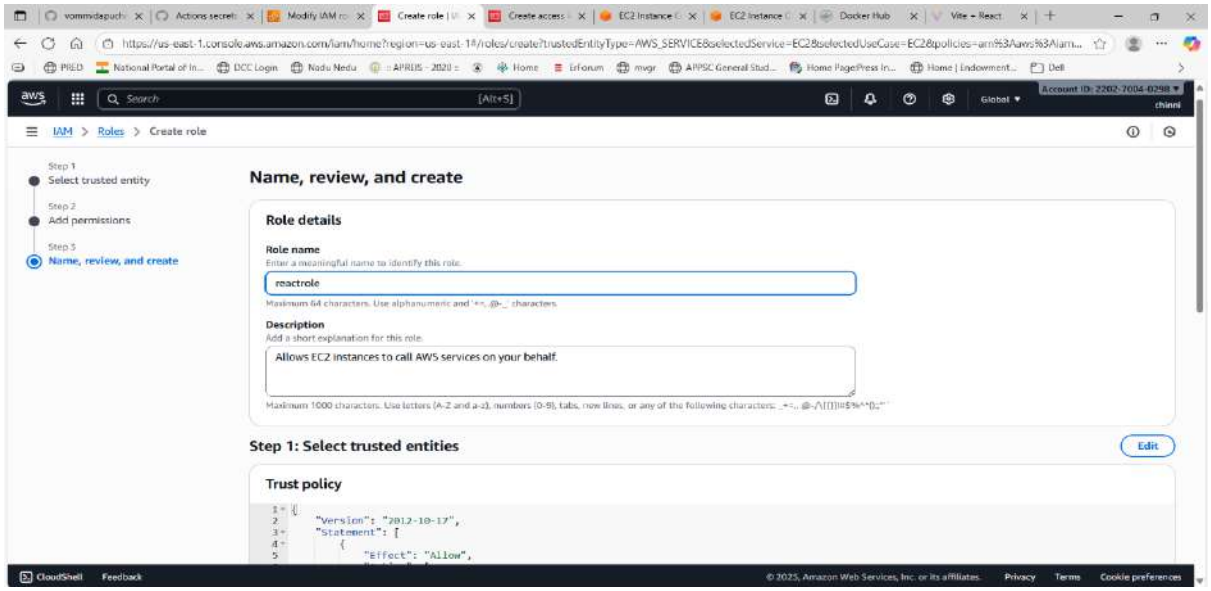

- Network settings (Security group): Create new security group with rules:
  - SSH (port 22) — Source: My IP (recommended) or your IP range
  - HTTP (port 80) — Source: Anywhere (0.0.0.0/0) (to allow web traffic)
  - Allow port 4000
- Click Launch instance.

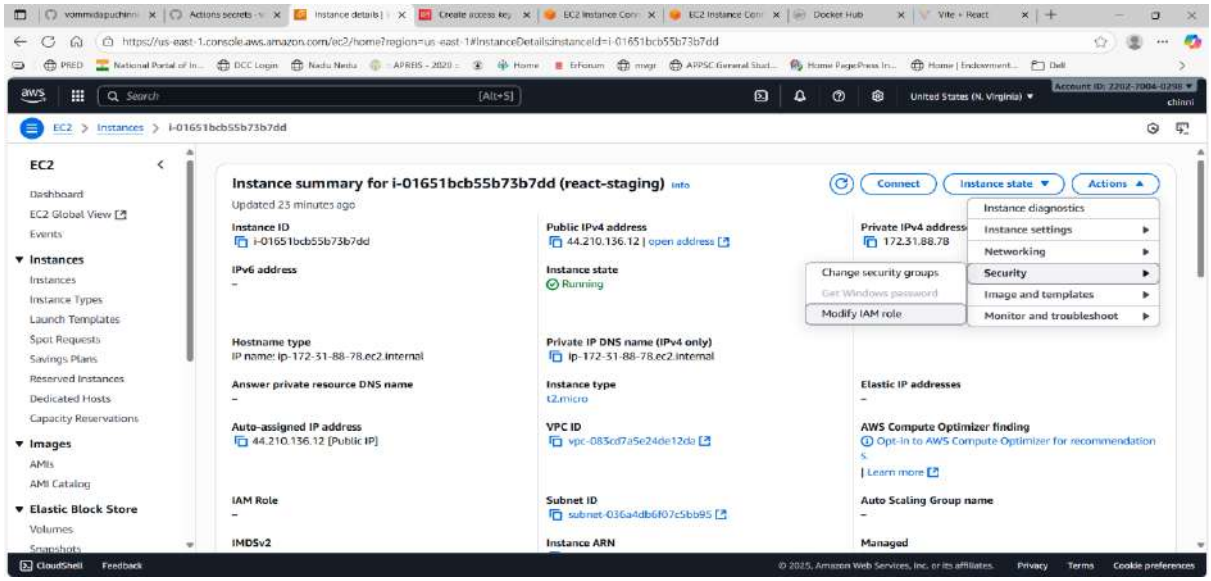


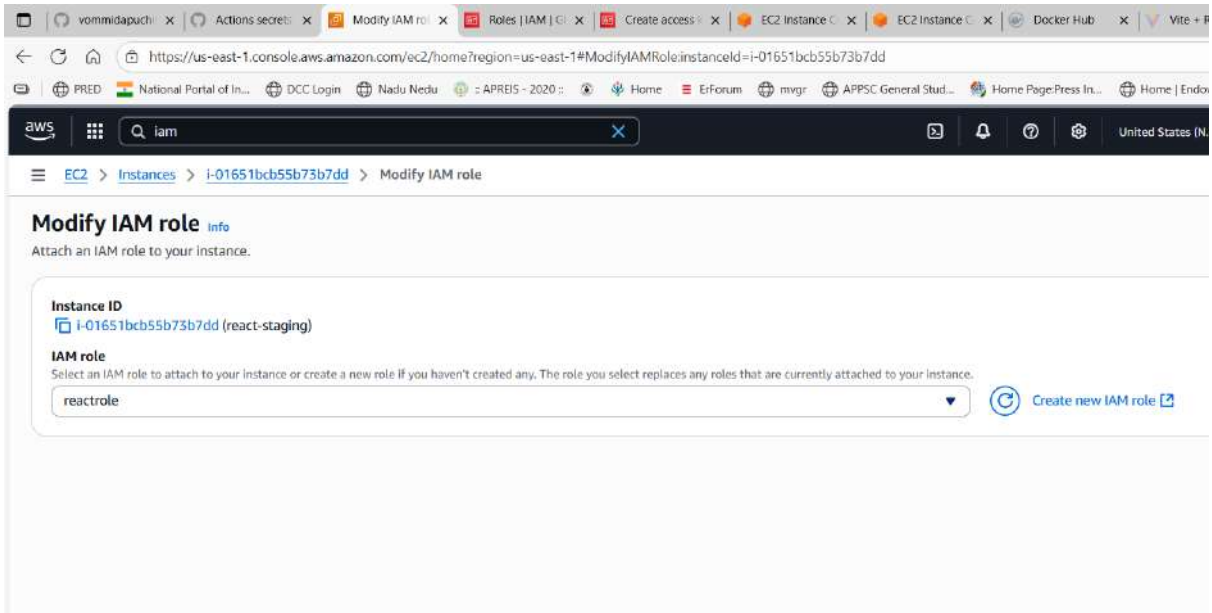

CloudWatch Works with Roles

- Create a role with **CloudWatchAgentServerPolicy**.
- Attach it to your EC2 instance.
- The agent automatically has permission — no credentials in JSON needed.




IAM > Roles > Create role
Step 1
Select trusted entity
Step 2
Add permissions
Step 3
Name, review, and create
Name, review, and create
Role details
Role name
Enter a meaningful name to identify this role.
reactrole
Maximum 64 characters. Use alphanumeric and '+=,.@-_' characters.
Description
Add a short explanation for this role.
Allows EC2 instances to call AWS services on your behalf.
Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: _+=,.@-/\[]{}!#$%^*();'"
Step 1: Select trusted entities
Edit
Trust policy
{
"Version": "2012-10-17",
"Statement": [
{
"Effect": "Allow",
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


EC2 > Instances > i-01651bcb55b73b7dd
EC2
Dashboard
EC2 Global View
Events
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Instance summary for i-01651bcb55b73b7dd (react-staging) Info
Updated 23 minutes ago
Connect
Instance state
Actions
Instance diagnostics
Instance settings
Networking
Security
Image and templates
Monitor and troubleshoot
Change security groups
Get Windows password
Modify IAM role
Instance ID
i-01651bcb55b73b7dd
Public IPv4 address
44.210.136.12 | open address
Private IPv4 address
172.31.88.78
IPv6 address
Instance state
Running
Hostname type
IP name: ip-172-31-88-78.ec2.internal
Private IP DNS name (IPv4 only)
ip-172-31-88-78.ec2.internal
Answer private resource DNS name
Instance type
t2.micro
Elastic IP addresses
Auto-assigned IP address
44.210.136.12 [Public IP]
VPC ID
vpc-083cd7a5e24de12da
AWS Compute Optimizer finding
Opt-in to AWS Compute Optimizer for recommendations
Learn more
IAM Role
Subnet ID
subnet-036a4db6f07c5bb95
Auto Scaling Group name
IMDSv2
Instance ARN
Managed
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#ModifyIAMRole:instanceId=i-01651bcb55b73b7dd
iam
EC2 > Instances > i-01651bcb55b73b7dd > Modify IAM role
Modify IAM role Info
Attach an IAM role to your instance.
Instance ID
i-01651bcb55b73b7dd (react-staging)
IAM role
Select an IAM role to attach to your instance or create a new role if you haven't created any. The role you select replaces any roles that are currently attached to your instance.
reactrole
Create new IAM role

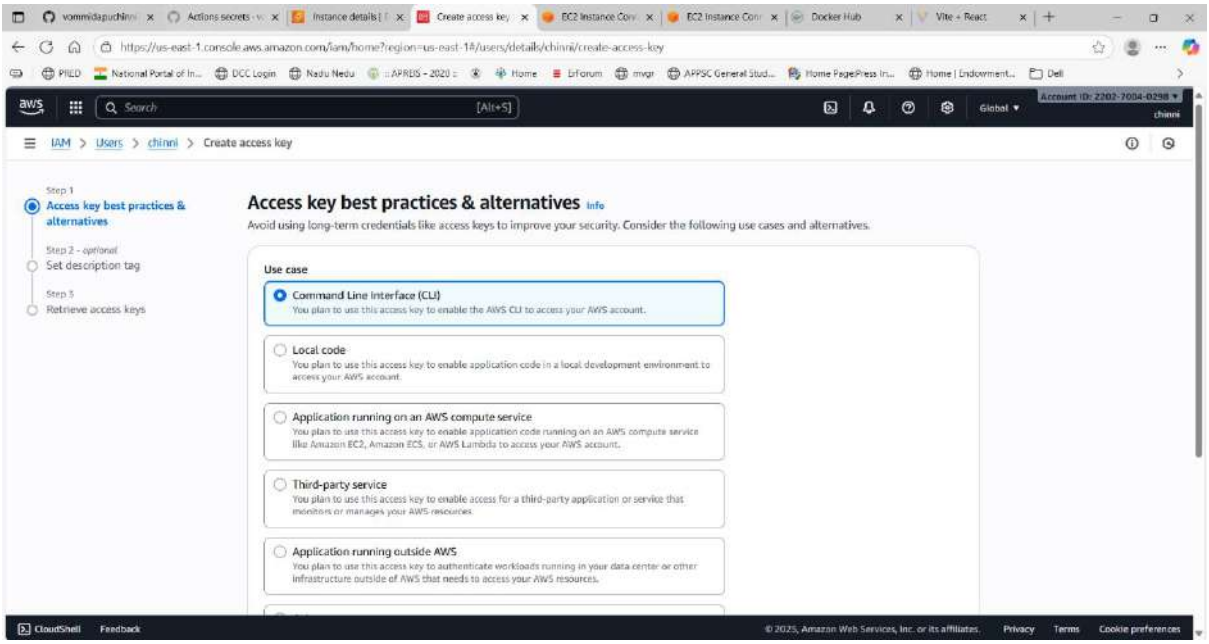
cloudwatch-staging.json created in root folder.

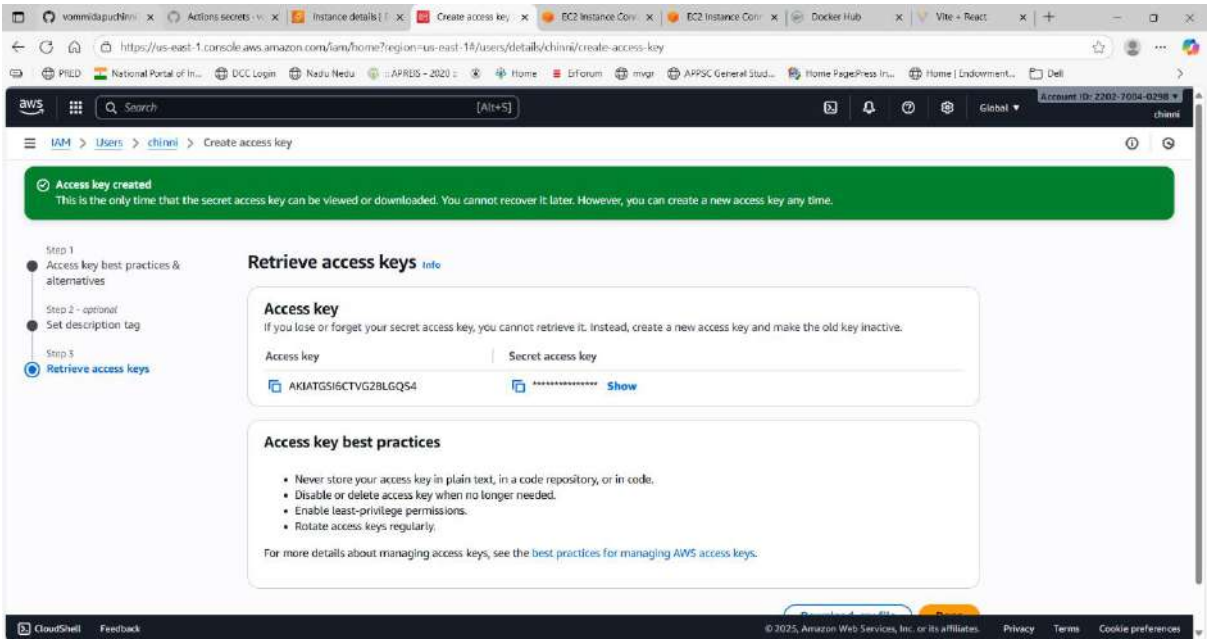
Start CloudWatch Agent : install cloud watch agent locally
sudo amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -c file:/home/ubuntu/my-react-app-staging/cloudwatch-staging.json -s

This will **start the agent** using the IAM role.

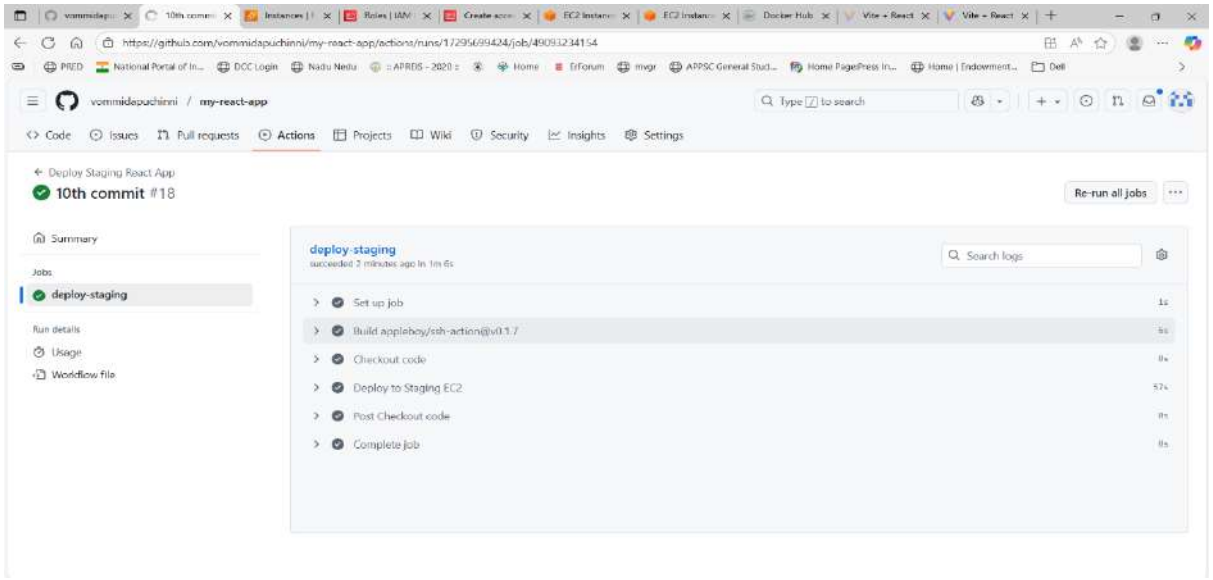
You can check status: amazon-cloudwatch-agent-ctl -a status

Created user and created keys for it and kept in github secrets

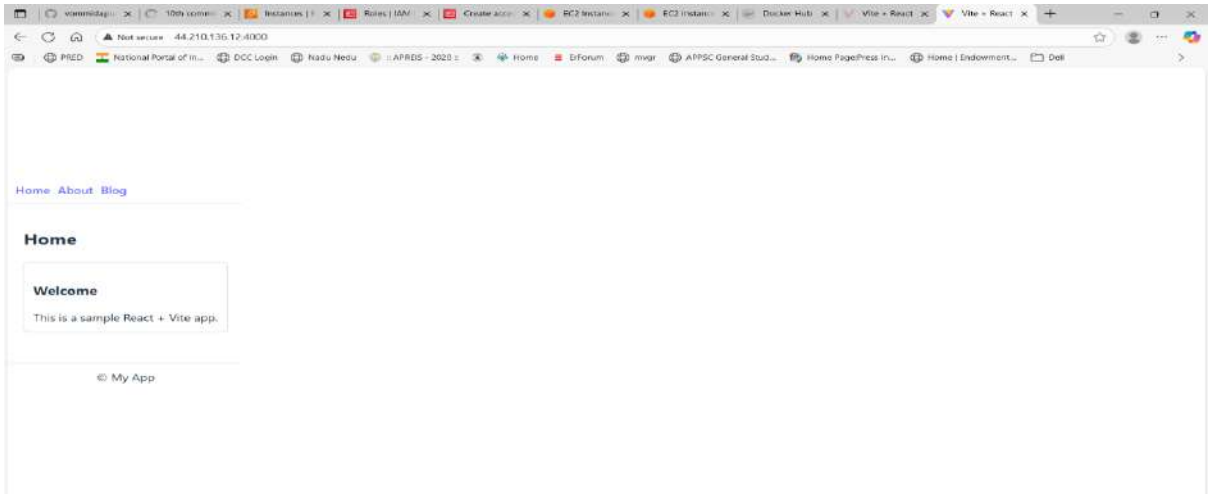


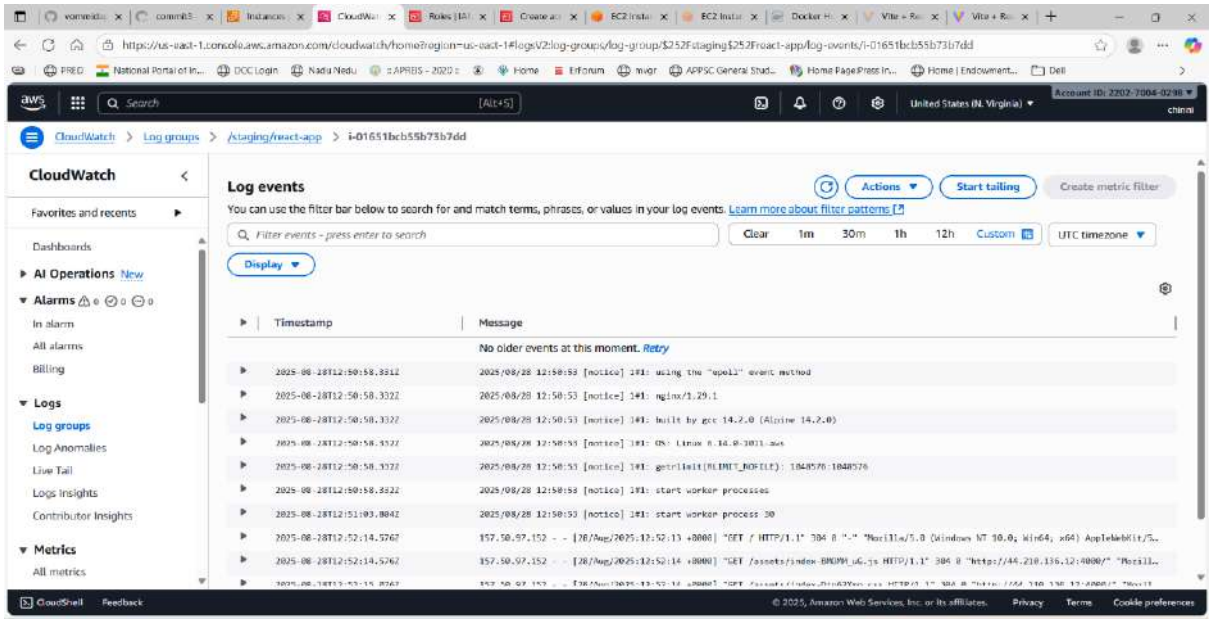

workflow




Accessed with ip




Cloud watch log groups



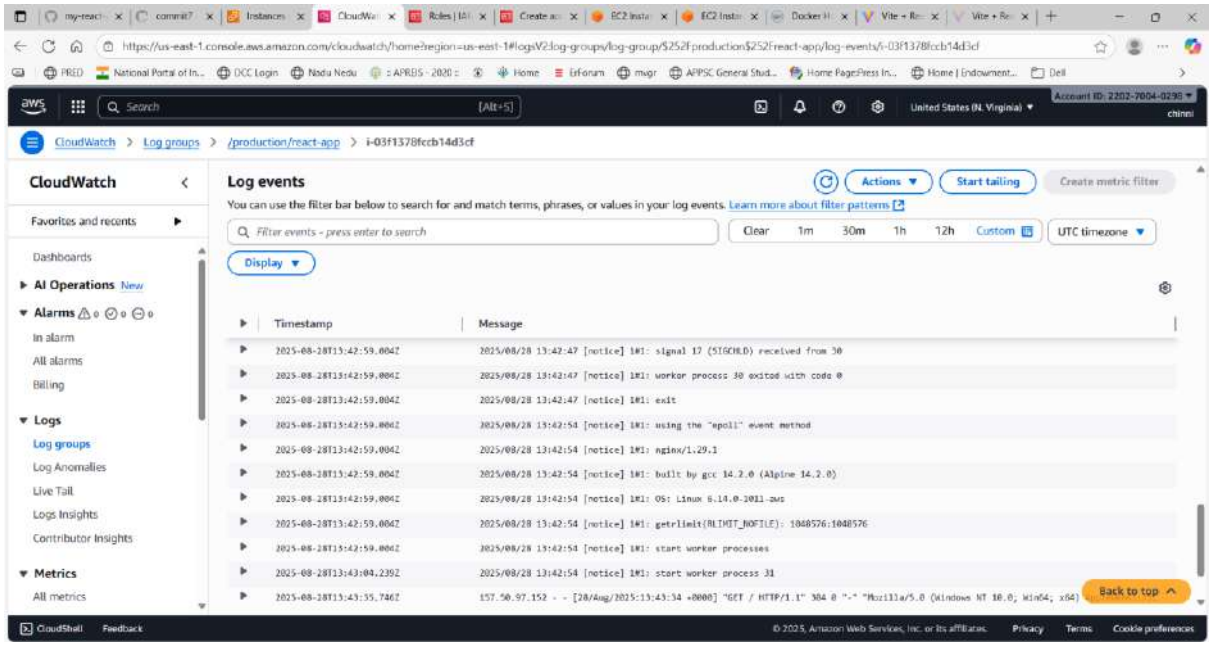
For production  I did not kept cloud watch log group for that I am doing I have attached role to ec2 instance and created cloud watch log group and given loggroup folder in docker compose files in productions and pushed to github

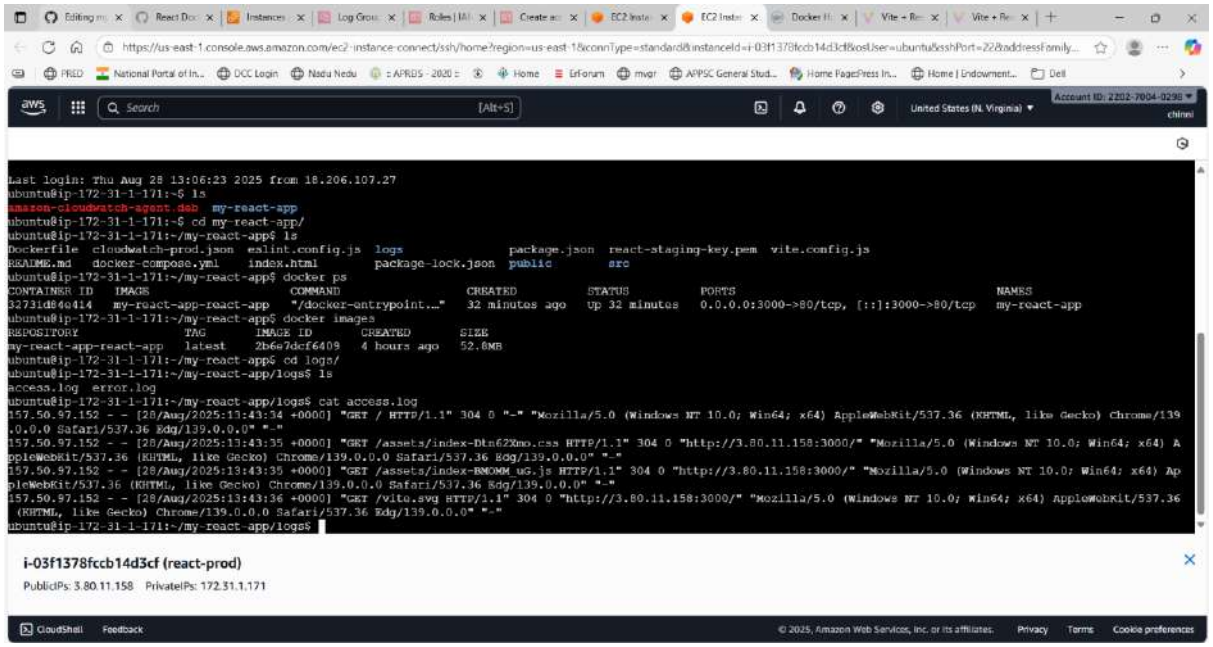
Create CloudWatch Configuration for Production: cloudwatch-prod.json

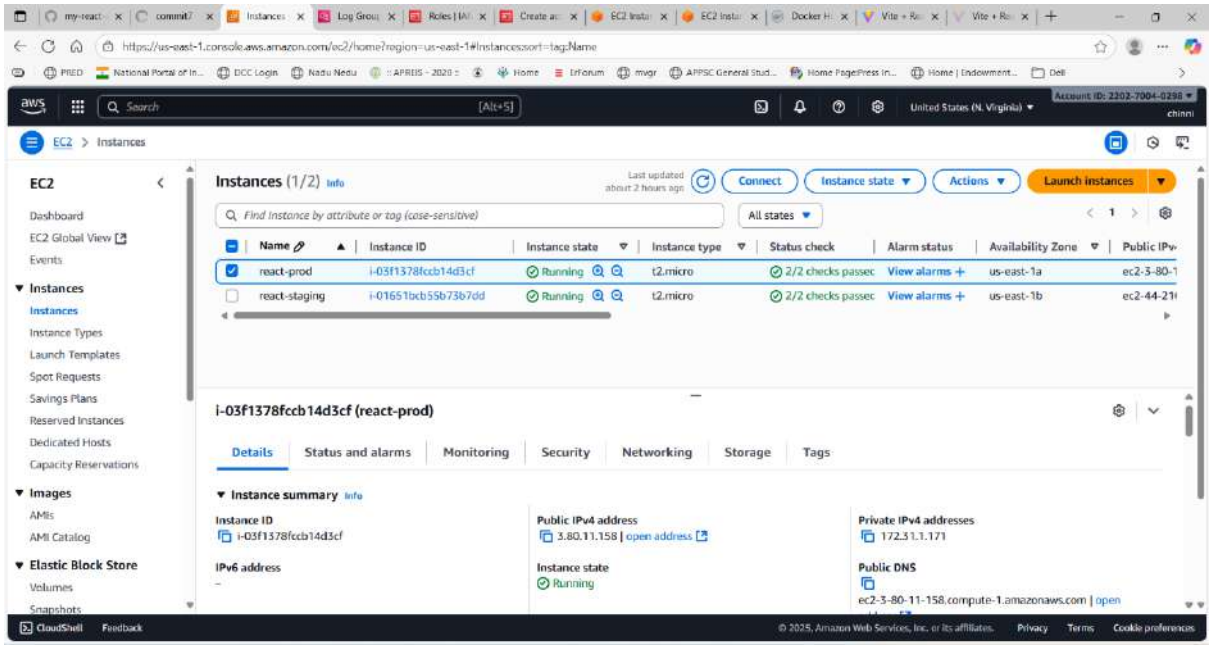
Update Production Docker Compose

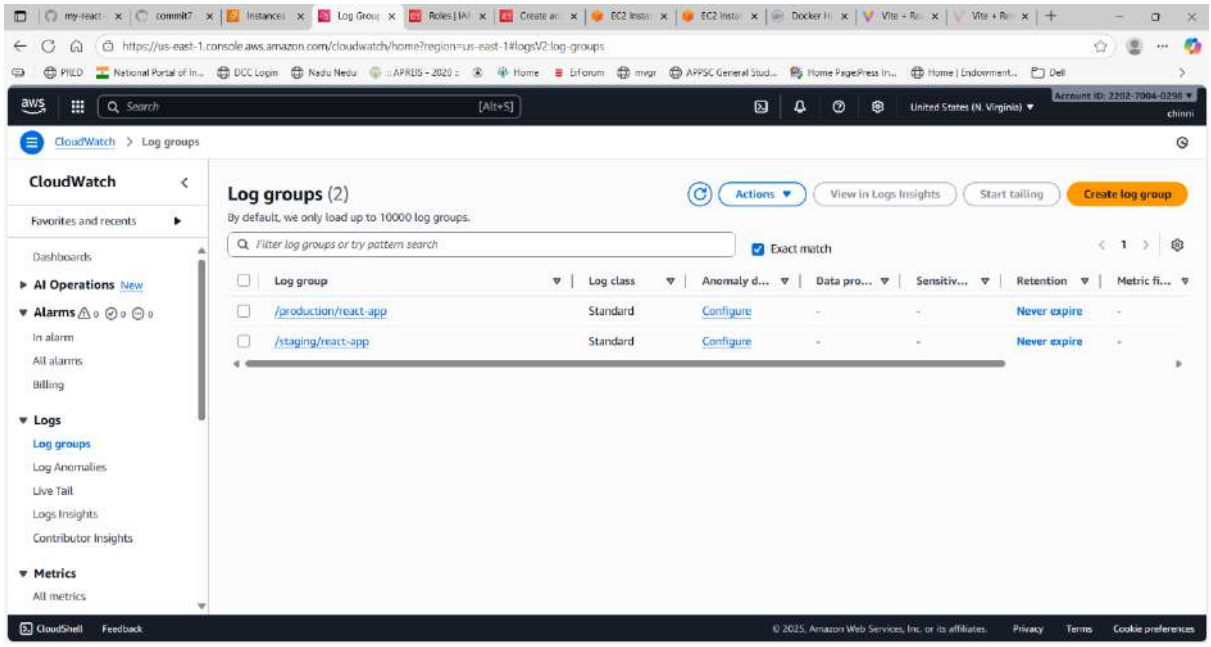
**Update Production CI/CD Pipeline**

Add **CloudWatch agent start command** to your production deploy YAML






want to **host our React app locally on our machine** and also make it **live on GitHub Pages**

**Prepare our React app for GitHub Pages:** Open our package.json and add the homepage field at the top level

**Next Steps to Deploy on GitHub Pages: Switch BrowserRouter to HashRouter in main.jsx**

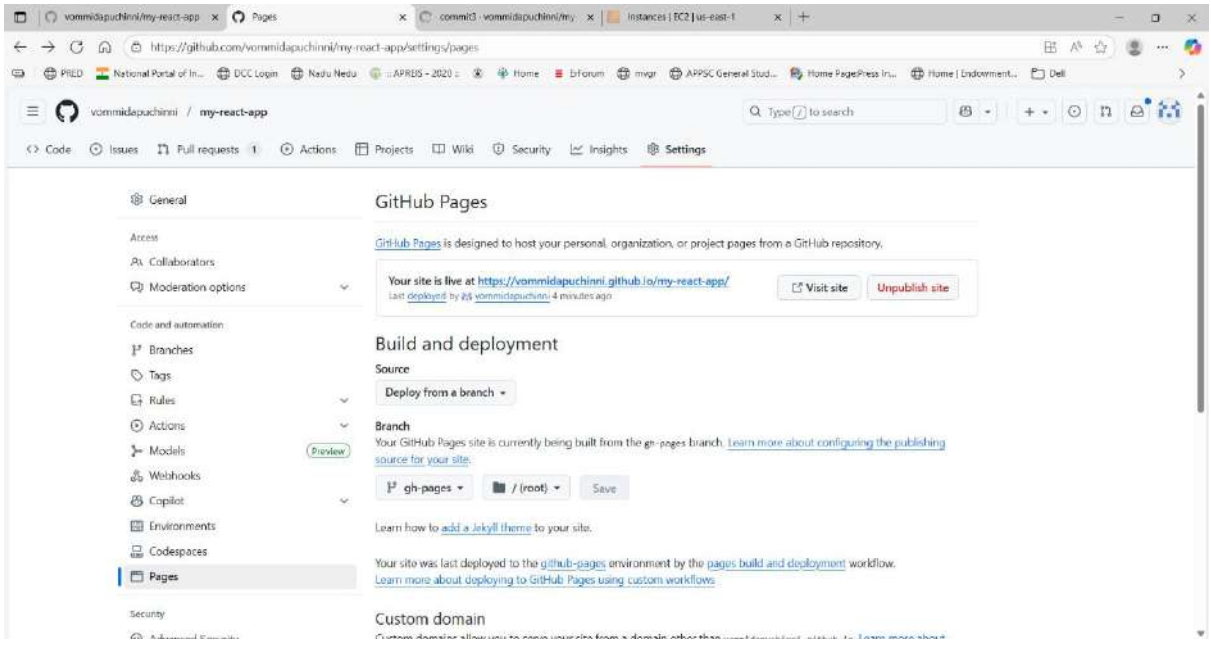
Install gh-pages: npm install --save-dev gh-pages

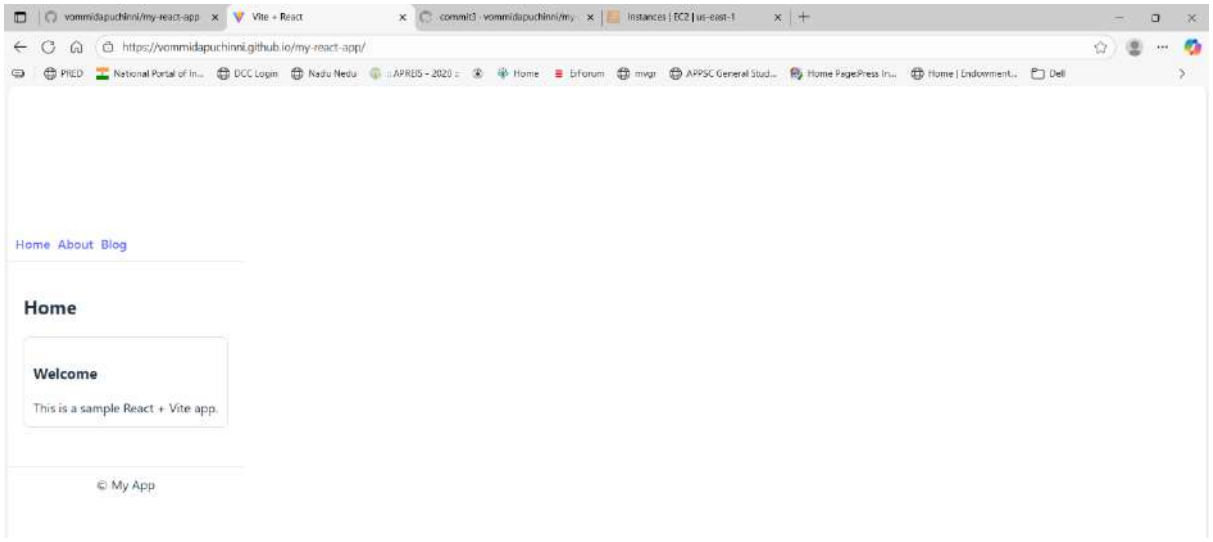
Build and Deploy: npm run predeploy  # Creates production build in dist/

npm run deploy  # Pushes dist/ to gh-pages branch

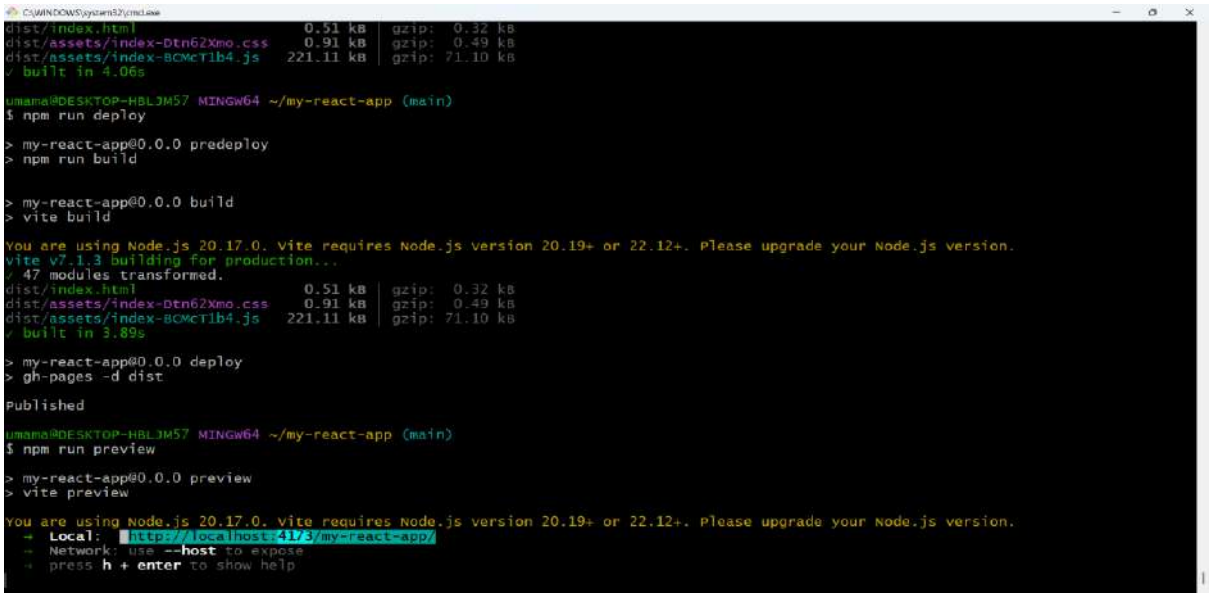
```
MINGW64:/c/Users/umama/my-react-app
umama@DESKTOP-HBLJM57 MINGW64 ~/my-react-app (main)
$ npm run build

> my-react-app@0.0.0 build
> vite build

You are using Node.js 20.17.0. Vite requires Node.js version 20.19+ or 22.12+. Please upgrade your Node.js version.
vite v7.1.3 building for production...
✓ 47 modules transformed.
dist/index.html                  0.47 kB │ gzip:  0.31 kB
dist/assets/index-Dtn62Xmo.css   0.91 kB │ gzip:  0.49 kB
dist/assets/index-BMOMM_uG.js  220.73 kB │ gzip: 70.96 kB
✓ built in 3.92s

umama@DESKTOP-HBLJM57 MINGW64 ~/my-react-app (main)
$ npm run deploy

> my-react-app@0.0.0 predeploy
> npm run build


> my-react-app@0.0.0 build
> vite build

You are using Node.js 20.17.0. Vite requires Node.js version 20.19+ or 22.12+. Please upgrade your Node.js version.
vite v7.1.3 building for production...
✓ 47 modules transformed.
dist/index.html                  0.47 kB │ gzip:  0.31 kB
dist/assets/index-Dtn62Xmo.css   0.91 kB │ gzip:  0.49 kB
dist/assets/index-BMOMM_uG.js  220.73 kB │ gzip: 70.96 kB
✓ built in 3.70s

> my-react-app@0.0.0 deploy
> gh-pages -d dist

Published

umama@DESKTOP-HBLJM57 MINGW64 ~/my-react-app (main)
$
```

After deployment, go to your repo **Settings → Pages** to confirm the live URL: https://vommidapuchinni.github.io/my-react-app

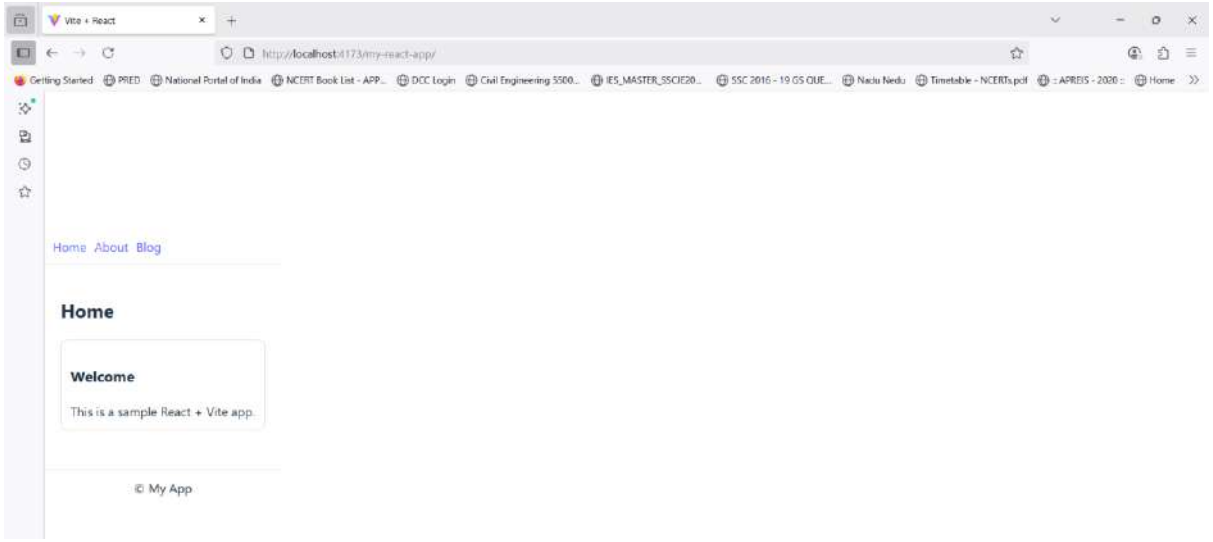





Preview Locally: npm run preview




- Opens your production build locally at http://localhost:4173.



**Troubleshooting**

- **Container restarts** → check Dockerfile build stage.
- **Logs not in CloudWatch** → verify agent config, JSON path, and volume mapping.
- **Workflow fails** → confirm SSH keys and IAM role permissions.
- **App not accessible** → check port mapping and EC2 security group.

**Key Concepts**

- **Docker multi-stage build**: Node.js build → Nginx serve.
- **Docker Compose**: defines service, ports, and log volume.
- **CloudWatch Agent**: collects logs from host directory to log groups.
- **CI/CD with GitHub Actions**: auto-deploys on branch push using SSH.
- **Monitoring**: logs viewable in CloudWatch for staging (/staging/react-app) and production (/production/react-app).

**Conclusion**

Successfully deployed a React app with **staging and production environments** on AWS EC2 using **Docker** and **GitHub Actions CI/CD**. Logs are collected in **CloudWatch** for monitoring.