

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

Repo link: [vommidapuchinni/my-react-app](https://github.com/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 secret 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
```

```
MINGW64 ~/my-react-app
umama@DESKTOP-HBL3M57 MINGW64 ~
$ node -v
v20.17.0

umama@DESKTOP-HBL3M57 MINGW64 ~
$ npm -v
10.8.3

umama@DESKTOP-HBL3M57 MINGW64 ~
$ npm create vite@latest my-react-app -- --template react
Need to install the following packages:
create-vite@7.1.1
ok to proceed? (y) y

npm warn EBADENGINE Unsupported engine {
npm warn EBADENGINE   package: 'create-vite@7.1.1',
npm warn EBADENGINE   required: { node: 'v20.19.0 || >=22.12.0' },
npm warn EBADENGINE   current: { node: 'v20.17.0', npm: '10.8.3' }
}

> npx
> create-vite my-react-app --template react

Scaffolding project in C:\Users\umama\my-react-app...

Done. Now run:

cd my-react-app
npm install
npm run dev

umama@DESKTOP-HBL3M57 MINGW64 ~
$ cd my-react-app
```

- npm install
- npm i react-router-dom

```
MINGW64 ~/my-react-app
umama@DESKTOP-HBL3M57 MINGW64 ~/my-react-app
$ npm install
npm warn EBADENGINE Unsupported engine {
npm warn EBADENGINE   package: '@vitejs/plugin-react@5.0.2',
npm warn EBADENGINE   required: { node: 'v20.19.0 || >=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: 'v20.19.0 || >=22.12.0' },
npm warn EBADENGINE   current: { node: 'v20.17.0', npm: '10.8.3' }
}
npm warn EBADENGINE }

added 152 packages, and audited 153 packages in 41s

33 packages are looking for funding
  run 'npm fund' for details

found 0 vulnerabilities

umama@DESKTOP-HBL3M57 MINGW64 ~/my-react-app
$ npm i react-router-dom
npm warn EBADENGINE Unsupported engine {
npm warn EBADENGINE   package: '@vitejs/plugin-react@5.0.2',
npm warn EBADENGINE   required: { node: 'v20.19.0 || >=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: 'v20.19.0 || >=22.12.0' },
npm warn EBADENGINE   current: { node: 'v20.17.0', npm: '10.8.3' }
}
npm warn EBADENGINE }

added 4 packages, and audited 157 packages in 5s

33 packages are looking for funding
  run 'npm fund' for details
```

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 vite@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";

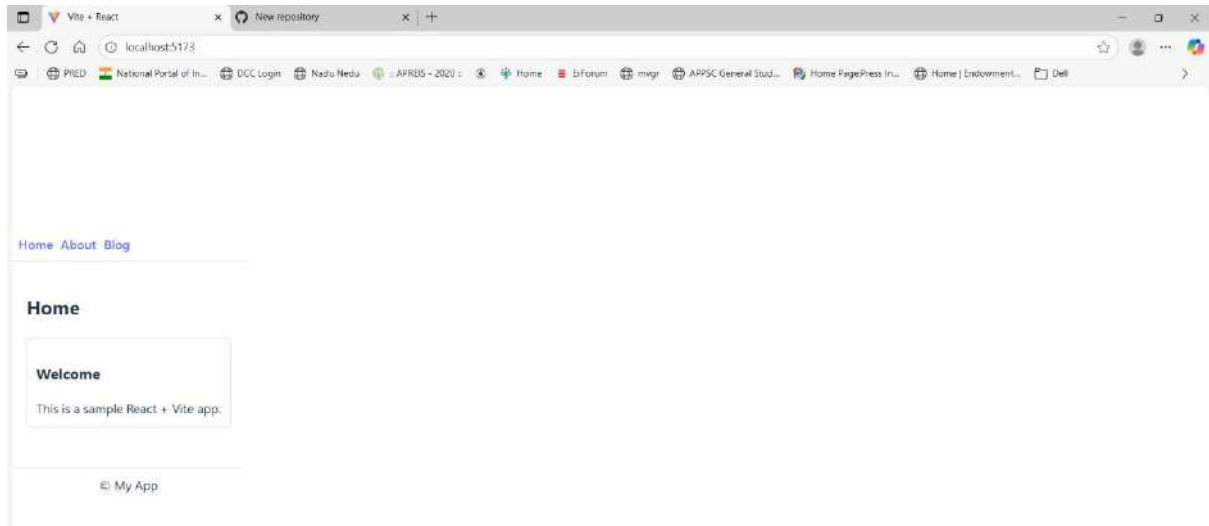
export default function App() {
  return (
    <>
      <Header />
      <main style={{padding:20}}>
        <Routes>
          <Route path="/" element={<Home />} />
        </Routes>
      </main>
      <Footer />
    </>
  );
}
```

```
umama@DESKTOP-HBL3M57 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-DtnG2Xmo.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
```

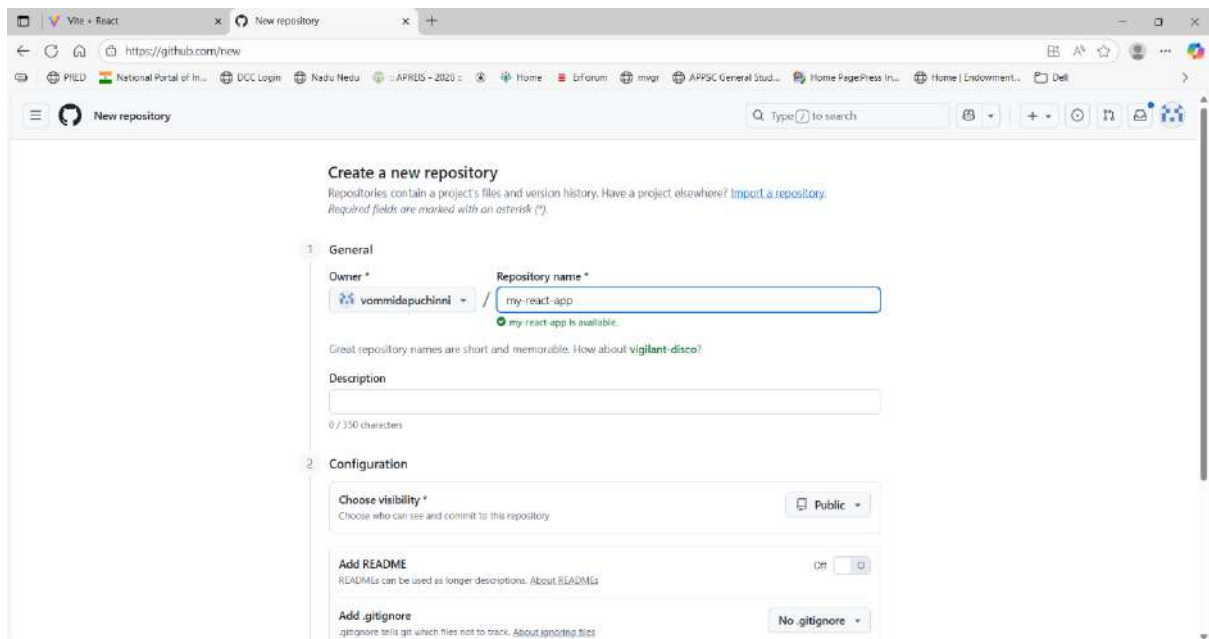
Accessing the file locally



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"

```
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

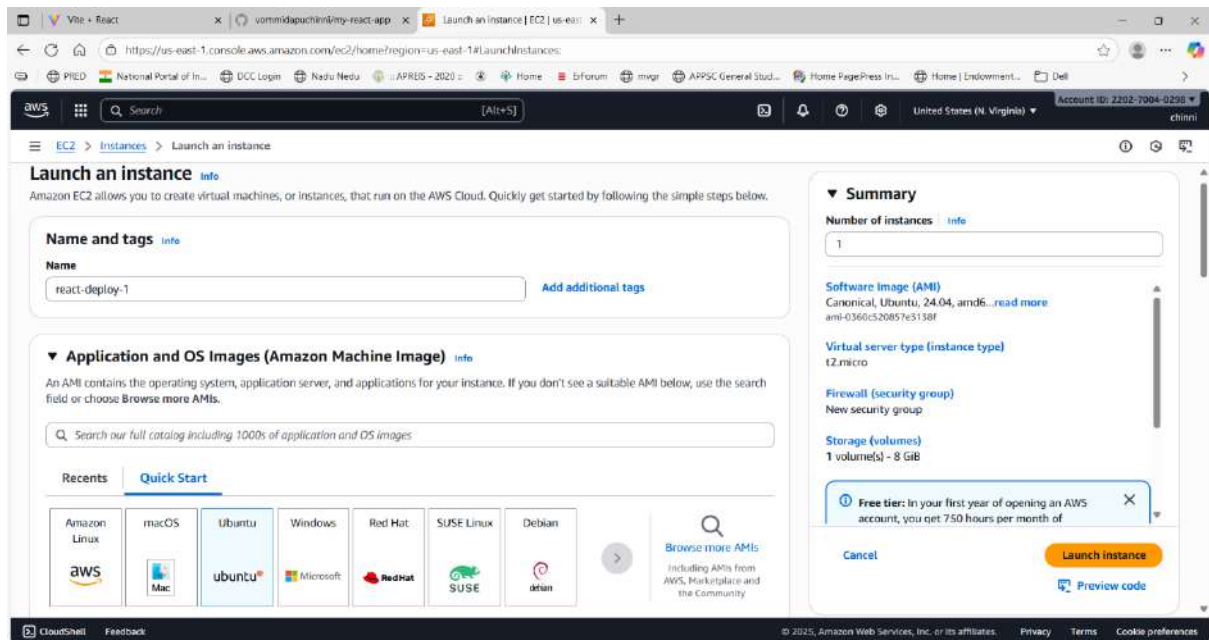
Goal: create an Ubuntu instance reachable by SSH (port 22) and HTTP (port 80).

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

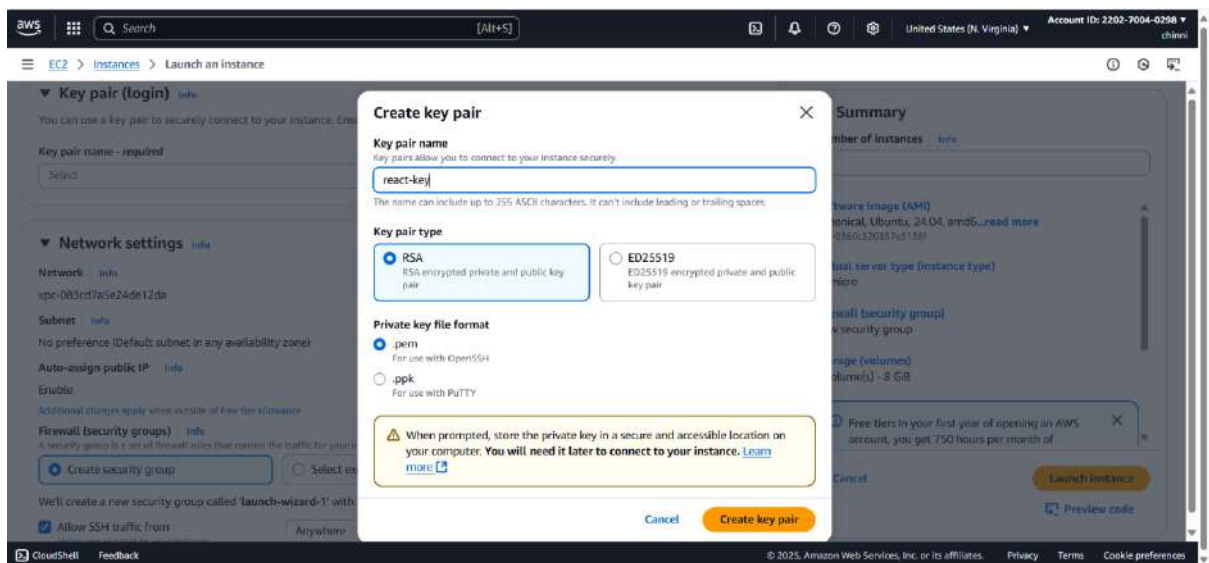
In EC2 Dashboard → left sidebar → **Instances** → **Launch instances** (button).

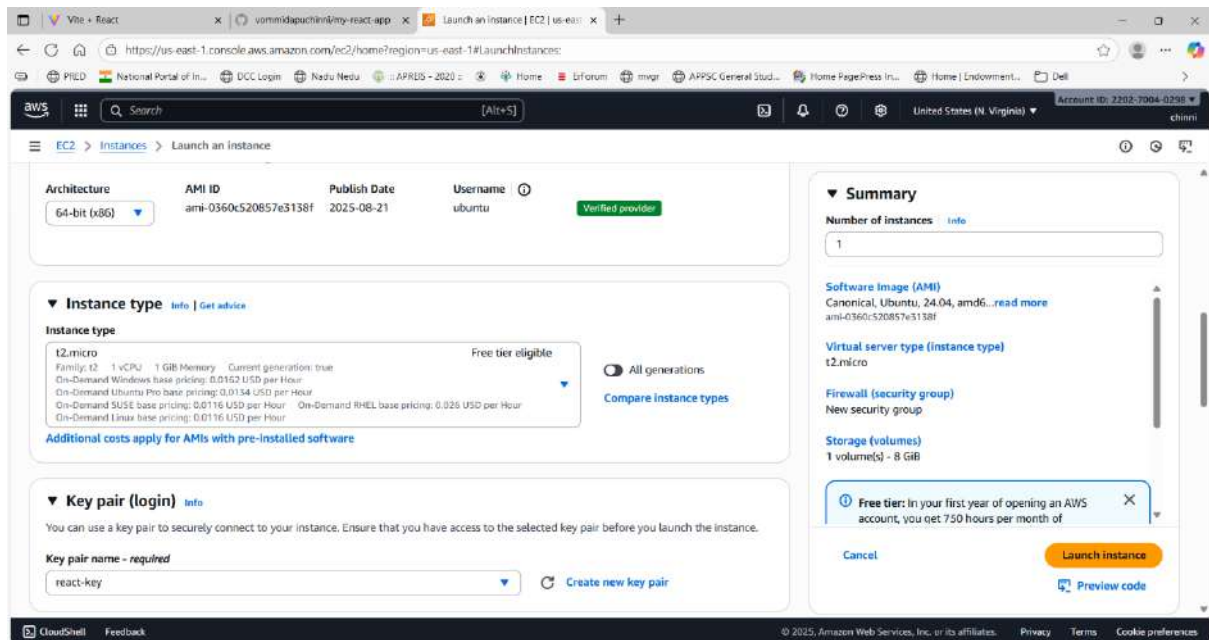
Launch instance wizard:

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

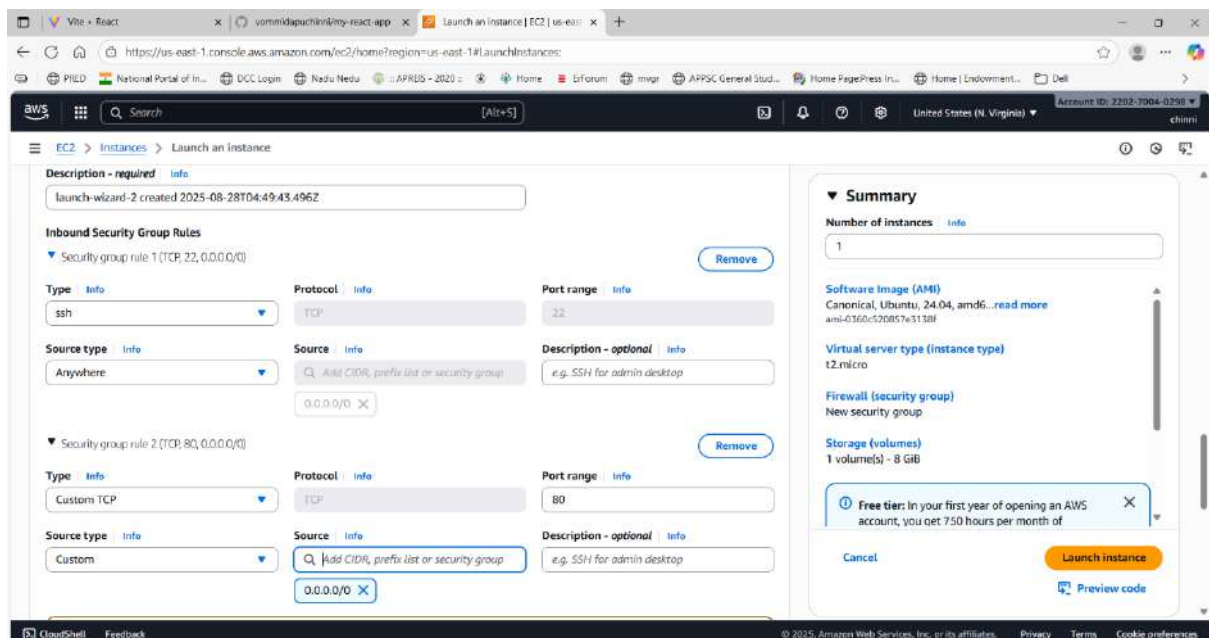


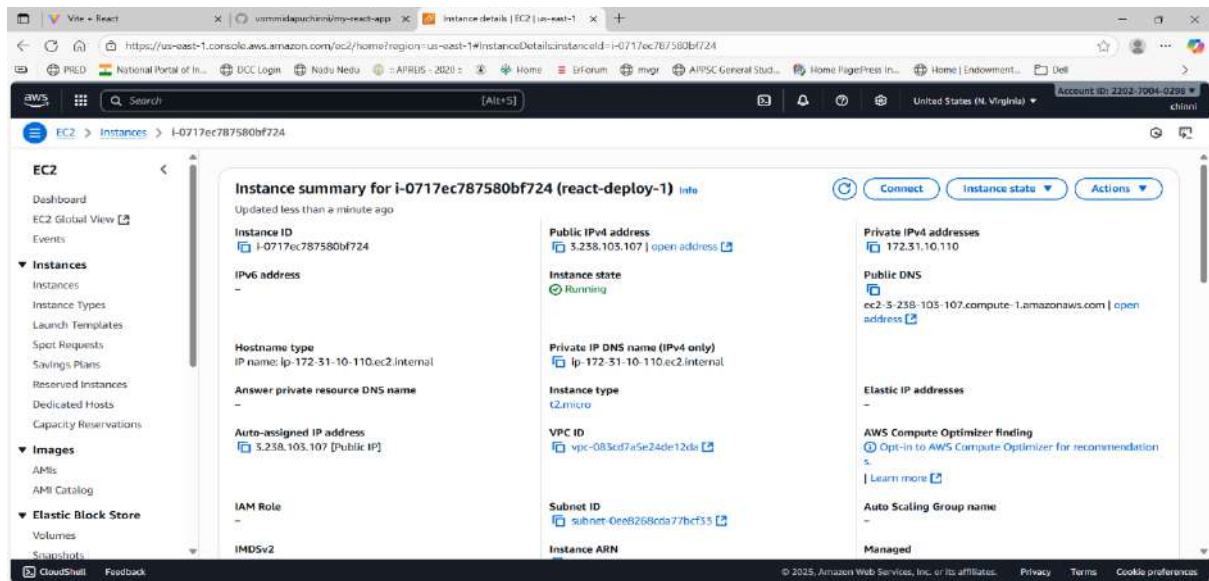
- Instance type: t2.micro (free tier eligible)
- Key pair (login): Create a new key pair → name it react-key → Download .pem file and keep it safe (you'll use it to SSH).



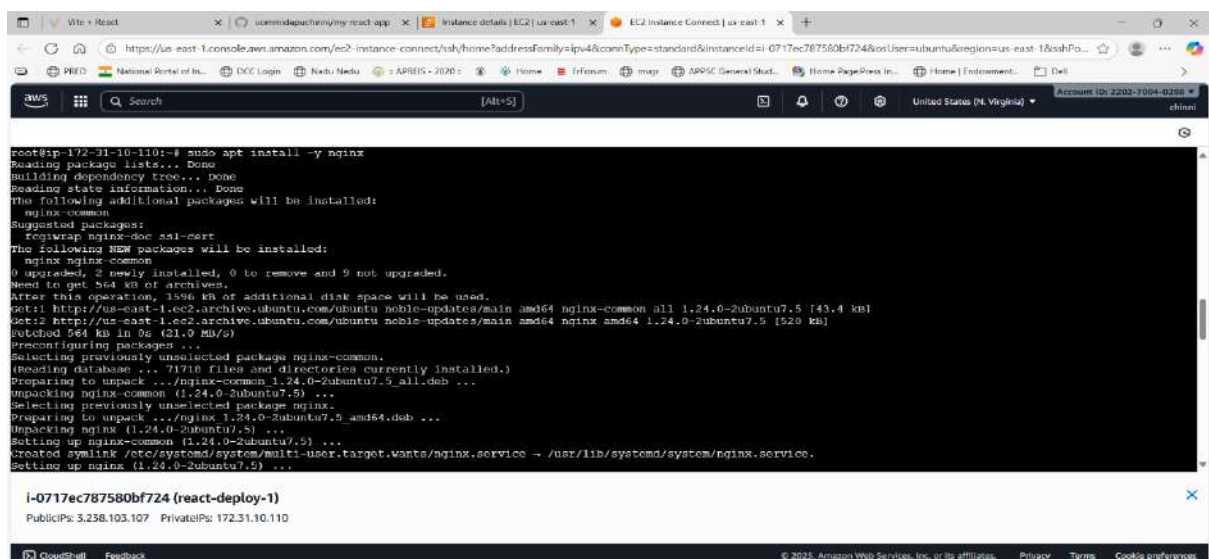
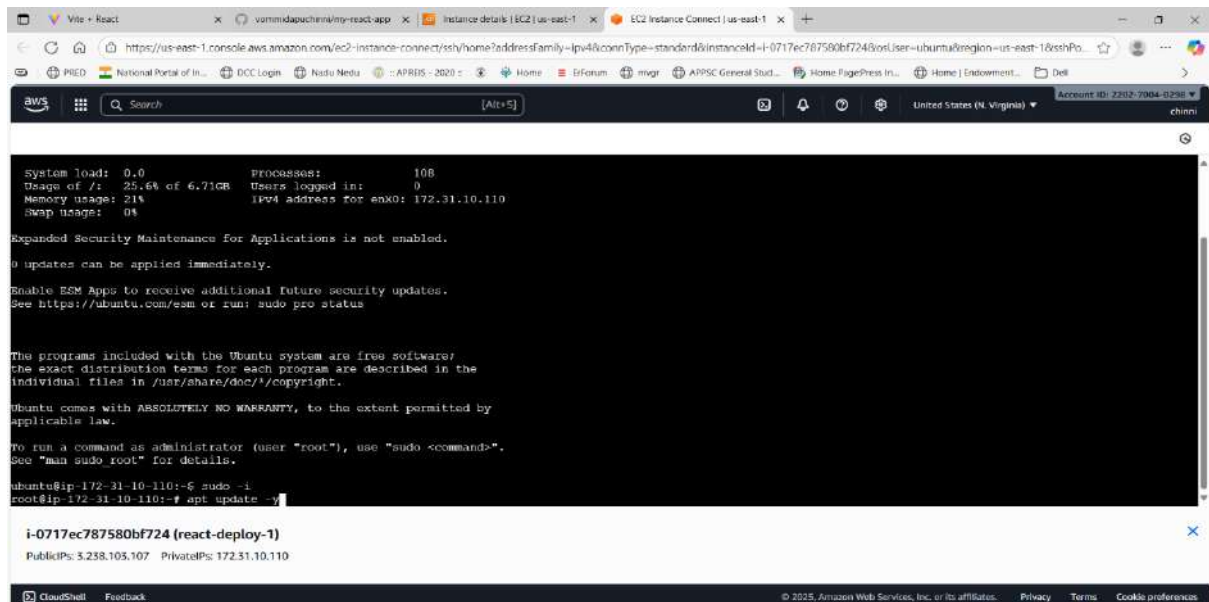


- Network settings (Security group): Create new security group with rules:
 - SSH (port 22) — Source: My IP (recommended) or your IP range
 - HTTP (port 80) — Source: Anywhere (0.0.0.0/0) (to allow web traffic)
 - Allow port 3000
- Click Launch instance.





Prepare the server




```
root@ip-172-31-10-110:~# sudo mkdir -p /var/www/myapp
root@ip-172-31-10-110:~# sudo chown -R $USER:$USER /var/www/myapp
root@ip-172-31-10-110:~# sudo chown -R www-data:www-data /var/www/myapp
root@ip-172-31-10-110:~# sudo chmod -R 755 /var/www/myapp
root@ip-172-31-10-110:~# sudo apt install -y ufw
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ufw is already the newest version (0.36.2-6).
ufw set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 9 not upgraded.
root@ip-172-31-10-110:~# sudo ufw allow OpenSSH
Rules updated
Rules updated (v6)
root@ip-172-31-10-110:~#
root@ip-172-31-10-110:~# sudo ufw allow 'Nginx Full'
Rules updated
Rules updated (v6)
root@ip-172-31-10-110:~# sudo ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y/n)? y
Firewall is active and enabled on system startup
root@ip-172-31-10-110:~# cd /etc
root@ip-172-31-10-110:~# cd /etc
ls
cron.hourly      groff            ld.so.conf       modprobe.d       perl             screenrc         sysstat
cron.monthly     group           ld.so.conf.d     modules           pki              security         systemd
cron.weekly      group-          ldconfig         modules-load.d    plymouth         selinux          terminfo
cron.yearly      grub.d          logcheck         mtabs             pm               sensors.d        timezone
PackageKit
X11
acpi
```

i-0717ec787580bf724 (react-deploy-1)
PublicIPs: 3.238.103.107 PrivateIPs: 172.31.10.110

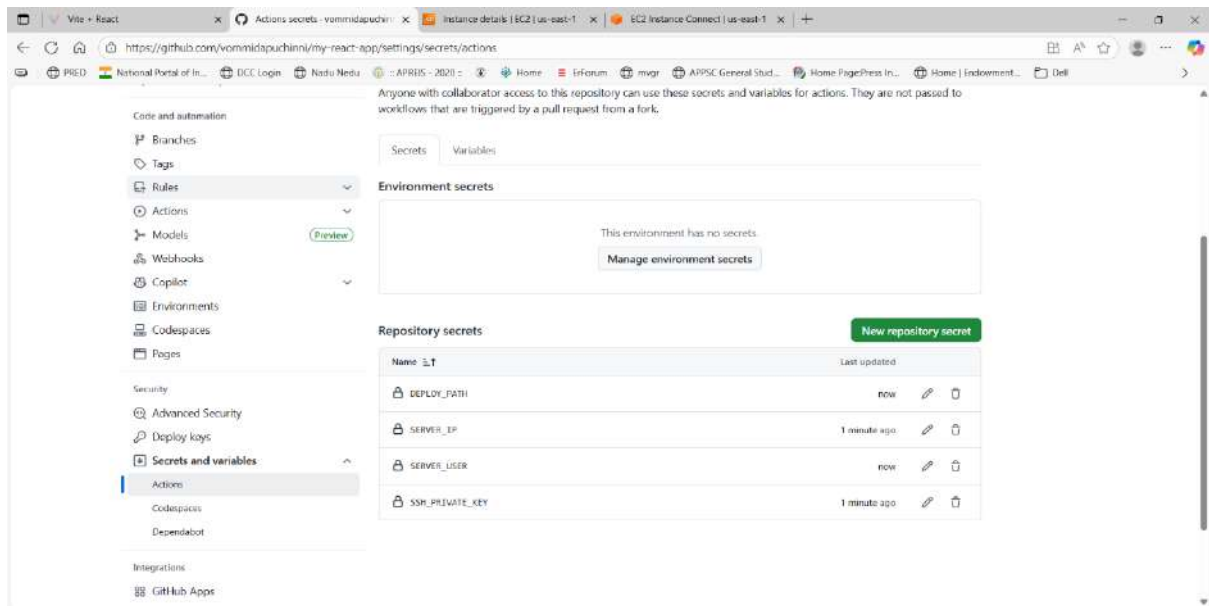
```
bash_completion.d  depmod.d          hosts              logcheck           networks           rc0.d              ssh                update-notifier
bindresvport.blacklist  dhcp              hosts.allow        login.defs          newt               rc1.d              ssl                usb_modeswitch.conf
binfmt.d           dhcpcd.conf       hosts.deny          logrotate.conf      nftables.conf     rc2.d              subgid             usb_modeswitch.d
byobu              dpkg              init.d             logrotate.d         nginx              rc3.d              subgid-            vconsole.conf
ca-certificates    e2scrub.conf     initramfs-tools    lsb-release         nsdswitch.conf    rc4.d              subuid             vis
ca-certificates.conf  ec2_version       inputrc            machine-id          on-release         rc5.d              sudo.conf          vmware-tools
cloud              environment       iproute2           mawk               overlayroot.conf  rc6.d              sudo.logrd.conf    vtrgrb
console-setup       ethertypes        issue              magic.mime          overlayroot.local.conf  rc7.d              sudoers            wgetrc
credstore           fstab             issue.net          manpath.conf        pam.conf           rc8.d              sudoers.d          xattr.conf
credstore.encrypted  fwupd            kernel             mdadm               pam.d              resolv.conf        sysctl.conf        xml
cron.d             gcal.conf         ldconfig           mime.types          parted            rsyslog.conf       sysctl.d           zsh_command_not_found
cron.daily          gnss              ld.so.cache        mk2fs.conf          password           rsyslog.d          sysctl.d           zsh

root@ip-172-31-10-110:~# cd /etc/nginx
root@ip-172-31-10-110:~# cd /etc/nginx# ls
conf.d  fastcgi_params  koi-win  modules-available  nginx.conf  scgi_params  sites-enabled  uwsgi_params
fastcgi.conf  koi-utf         mime.types  modules-enabled    proxy_params  sites-available  snippets      win-utf
root@ip-172-31-10-110:~# cd /etc/nginx# cd sites-available/
root@ip-172-31-10-110:~# cd /etc/nginx# cd sites-available# ls
default
root@ip-172-31-10-110:~# cd /etc/nginx/sites-available# vi myapp
root@ip-172-31-10-110:~# cd /etc/nginx/sites-available# sudo ln -s /etc/nginx/sites-available/myapp /etc/nginx/sites-enabled/
root@ip-172-31-10-110:~# cd /etc/nginx/sites-enabled# sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
root@ip-172-31-10-110:~# cd /etc/nginx/sites-enabled# sudo systemctl reload nginx
root@ip-172-31-10-110:~# cd /etc/nginx/sites-enabled#
```

i-0717ec787580bf724 (react-deploy-1)
PublicIPs: 3.238.103.107 PrivateIPs: 172.31.10.110

On GitHub (UI steps):

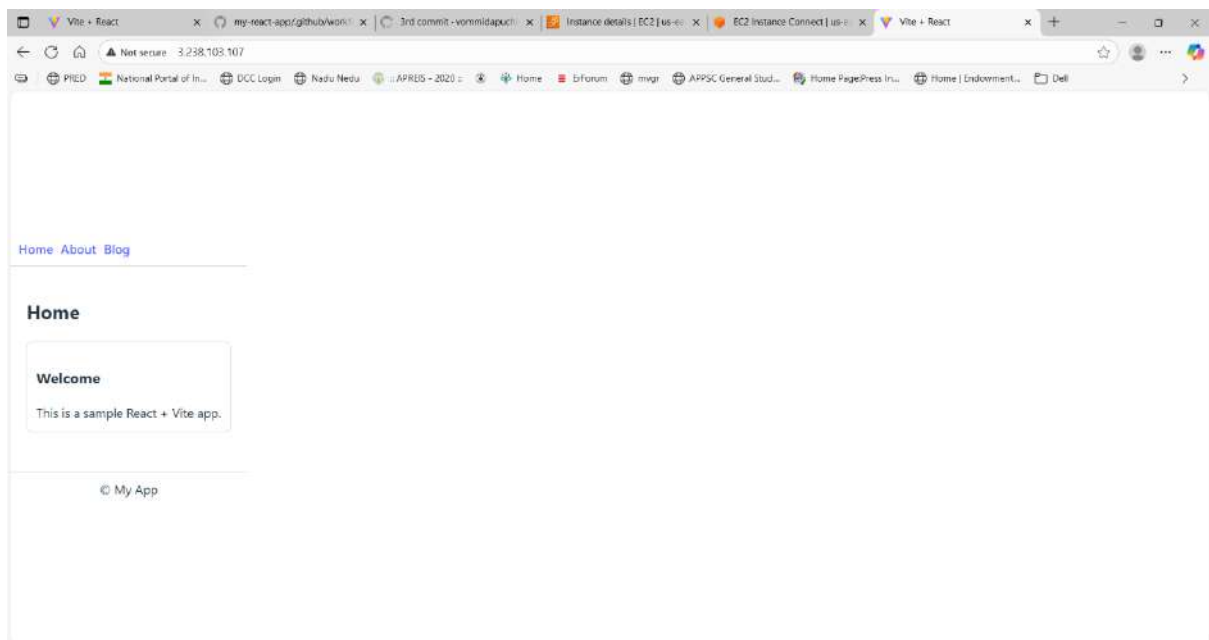
- Open your repo → Settings → Secrets and variables → Actions → New repository secret.
- Add secret SSH_PRIVATE_KEY — paste the content of downloaded key
- Add secret SERVER_IP — your EC2 IP.
- Add secret SERVER_USER — ubuntu (for Ubuntu AMI).
- Add secret DEPLOY_PATH — /var/www/myapp



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.



The screenshot shows the AWS Management Console terminal for an EC2 instance. The terminal output displays the status of the nginx service, which is active and running. It also shows the logs for the nginx service, indicating that it has been successfully reloaded and is now serving traffic. The instance ID is i-0717ec787580bf724 (react-deploy-1).

```
nginx: configuration file /etc/nginx/nginx.conf test is successful
root@ip-172-31-10-110:/var/www/myapp# sudo systemctl reload nginx
root@ip-172-31-10-110:/var/www/myapp# systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: enabled)
   Active: active (running) since Thu 2025-08-28 05:04:51 UTC; 1h 6min ago
     Docs: man:nginx(8)
    Process: 8700 ExecReload=/usr/sbin/nginx -g daemon on; master_process on; -- reload (code=exited, status=0/SUCCESS)
   Main PID: 2115 (nginx)
      Tasks: 2 (limit: 1121)
     Memory: 2.2M (peak: 4.3M)
        CPU: 66ms
    CGroup: /system.slice/nginx.service
            └─2115 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
               └─8702 "nginx: worker process"

Aug 28 05:13:48 ip-172-31-10-110 systemd[1]: Reloaded nginx.service - A high performance web server and a reverse proxy server.
Aug 28 05:29:50 ip-172-31-10-110 systemd[1]: Reloading nginx.service - A high performance web server and a reverse proxy server...
Aug 28 05:29:50 ip-172-31-10-110 nginx[2721]: 2025/08/28 05:29:50 [notice] 2721#2721: signal process started
Aug 28 05:29:50 ip-172-31-10-110 systemd[1]: Reloaded nginx.service - A high performance web server and a reverse proxy server.
Aug 28 05:39:14 ip-172-31-10-110 systemd[1]: Reloading nginx.service - A high performance web server and a reverse proxy server...
Aug 28 05:39:14 ip-172-31-10-110 nginx[2754]: 2025/08/28 05:39:14 [notice] 2754#2754: signal process started
Aug 28 05:39:14 ip-172-31-10-110 systemd[1]: Reloaded nginx.service - A high performance web server and a reverse proxy server.
Aug 28 05:50:19 ip-172-31-10-110 systemd[1]: Reloading nginx.service - A high performance web server and a reverse proxy server...
Aug 28 05:50:19 ip-172-31-10-110 nginx[8700]: 2025/08/28 05:50:19 [notice] 8700#8700: signal process started
Aug 28 05:50:19 ip-172-31-10-110 systemd[1]: Reloaded nginx.service - A high performance web server and a reverse proxy server.
root@ip-172-31-10-110:/var/www/myapp#
```

The screenshot shows a GitHub Actions workflow run for the 'build-and-deploy' job. The workflow is successful and took 1 hour and 17 minutes to complete. The job steps include: Set up job, Checkout, Use Node.js, Install dependencies, Lint, Test, Build, Start ssh-agent and add private key, Add server to known_hosts, Deploy files with rsync, Post Start ssh-agent and add private key, Post Use Node.js, Post Checkout, and Complete job.

Summary

Jobs

- build-and-deploy

Run details

Usage

Workflow file

build-and-deploy

succeeded 1 hour ago in 17s

Search logs

- Set up job
- Checkout
- Use Node.js
- Install dependencies
- Lint
- Test
- Build
- Start ssh-agent and add private key
- Add server to known_hosts
- Deploy files with rsync
- Post Start ssh-agent and add private key
- Post Use Node.js
- Post Checkout
- Complete job

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 creating we need to connect to docker desktop. Check docker version.

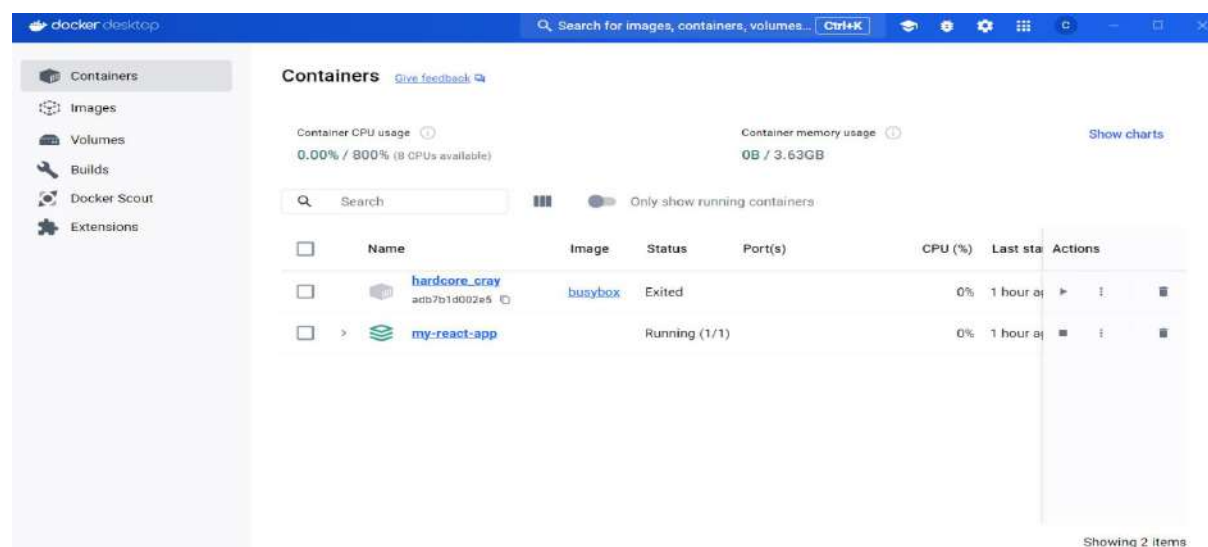
```
umama@DESKTOP-HBL3M57 MINGW64 ~/my-react-app (main)
$ docker version
Client:
Version:           26.1.4
API version:       1.45
Go version:        go1.21.11
Git commit:        5650f9b
Built:             wed Jun  5 11:29:54 2024
OS/Arch:           windows/amd64
Context:           desktop-linux

Server: Docker Desktop 4.31.1 (153621)
Engine:
Version:           26.1.4
API version:       1.45 (minimum version 1.24)
Go version:        go1.21.11
Git commit:        de5c9cf
Built:             wed Jun  5 11:29:22 2024
OS/Arch:           linux/amd64
Experimental:      false
Containerd:
Version:           1.6.33
GitCommit:         d2d58213f83a351ca8f528a95fbd145f5654e957
runc:
Version:           1.1.12
GitCommit:         v1.1.12-0-g51d5e94
docker-init:
Version:           0.19.0
GitCommit:         de40ad0
```

```
umama@DESKTOP-HBL3M57 MINGW64 ~/my-react-app (main)
$ docker-compose up -d --build
[+] Building 0.0s (0/0)  docker:desktop-linux
[+] Building 57.0s (16/16) FINISHED
=> [react-app internal] load build definition from dockerfile
=> => transferring dockerfile: 357B
=> [react-app internal] load metadata for docker.io/library/nginx:alpine
=> [react-app internal] load metadata for docker.io/library/node:18
=> [react-app auth] library/nginx:pull token for registry-1.docker.io
=> [react-app auth] library/node:pull token for registry-1.docker.io
=> [react-app internal] load .dockerignore
=> => transferring context: 2B
=> [react-app build 1/6] FROM docker.io/library/node:18@sha256:c6ae79e38498325db67193d391e6ec1d224d96c693a8a4d943498556716d3783
=> CACHED [react-app stage-1 1/2] FROM docker.io/library/nginx:alpine@sha256:42a516af16b852e33b7682d5ef8acbd5d13fe08fecadc7ed98
=> [react-app internal] load build context
=> => transferring context: 388.48kB
=> CACHED [react-app build 2/6] WORKDIR /app
=> CACHED [react-app build 3/6] COPY package*.json ./
=> CACHED [react-app build 4/6] RUN npm install
=> [react-app build 5/6] COPY . .
=> [react-app build 6/6] RUN npm run build
=> [react-app stage-1 2/2] COPY --from=build /app/dist /usr/share/nginx/html
=> [react-app] exporting to image
=> => exporting layers
=> => writing image sha256:4bc09093c16cdc43790a79c4dcb5bf7e217fcb02c3bffd803c0018fe5f2c73ad
=> => naming to docker.io/library/my-react-app-react-app
[+] Running 2/2
✔ Network my-react-app-default Created
✔ Container my-react-app-react-app-1 Started

umama@DESKTOP-HBL3M57 MINGW64 ~/my-react-app (main)
$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED          STATUS          PORTS                               NAMES
32ad1e01aeab  my-react-app-react-app             "/docker-entrypoint..." About a minute ago Up 57 seconds    0.0.0.0:3000->80/tcp  my-react-a
pp-react-app-1

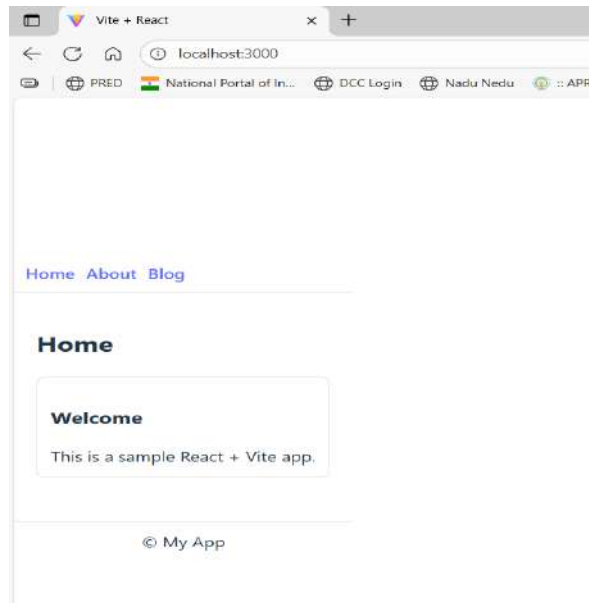
umama@DESKTOP-HBL3M57 MINGW64 ~/my-react-app (main)
$ docker images
REPOSITORY      TAG                IMAGE ID
```



docker ps we see container is running

Test React App

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

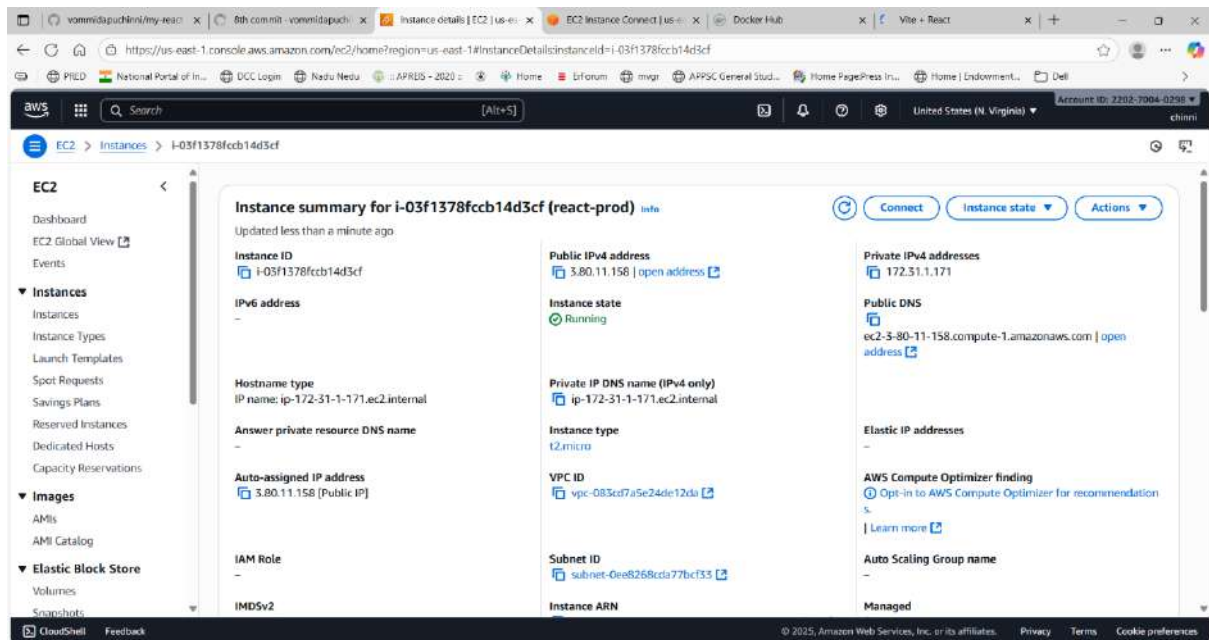


Production environment:

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**

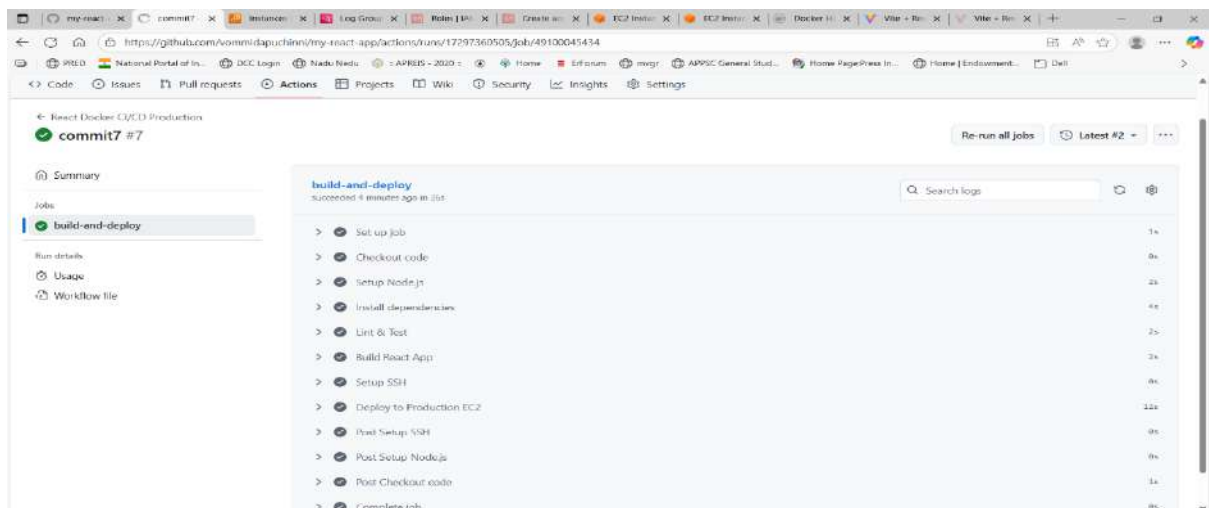
Name it, for example: **EC2_IP**

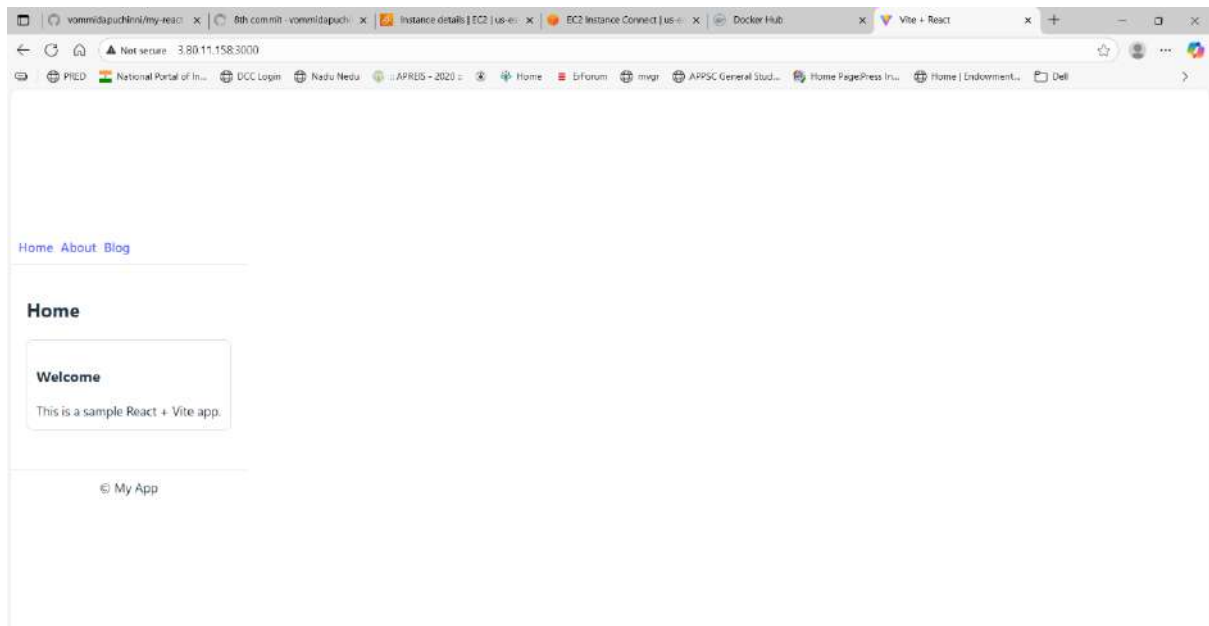
Value: your EC2 public IP (e.g., 3.80.11.158)

Save the secret

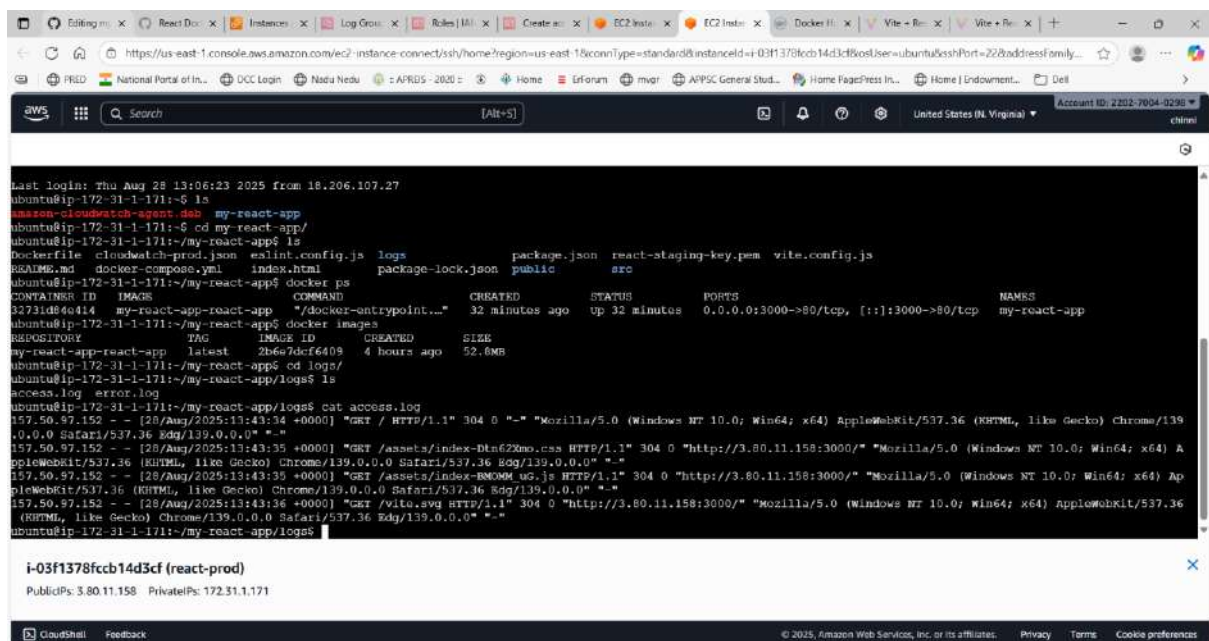
Created deploy-prod.yaml file and push to github

- 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.yml` and `deploy-staging.yml` files

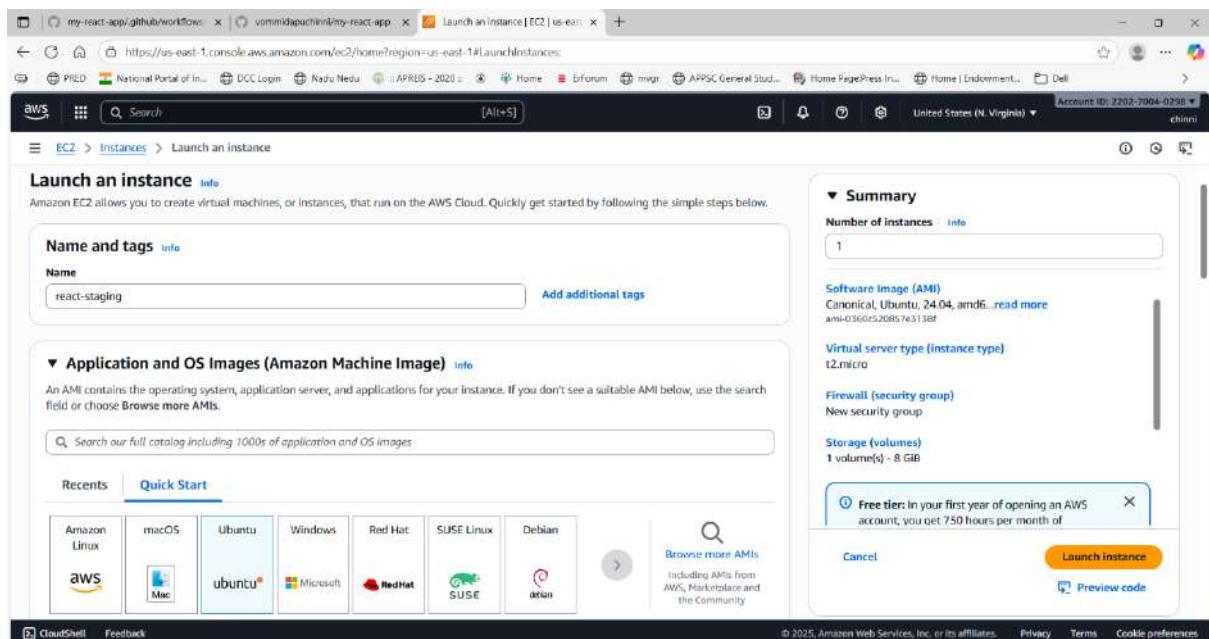
Staging `docker-compose-staging.yml` (port 4000 → 80 inside container)

For this created another EC2 instance

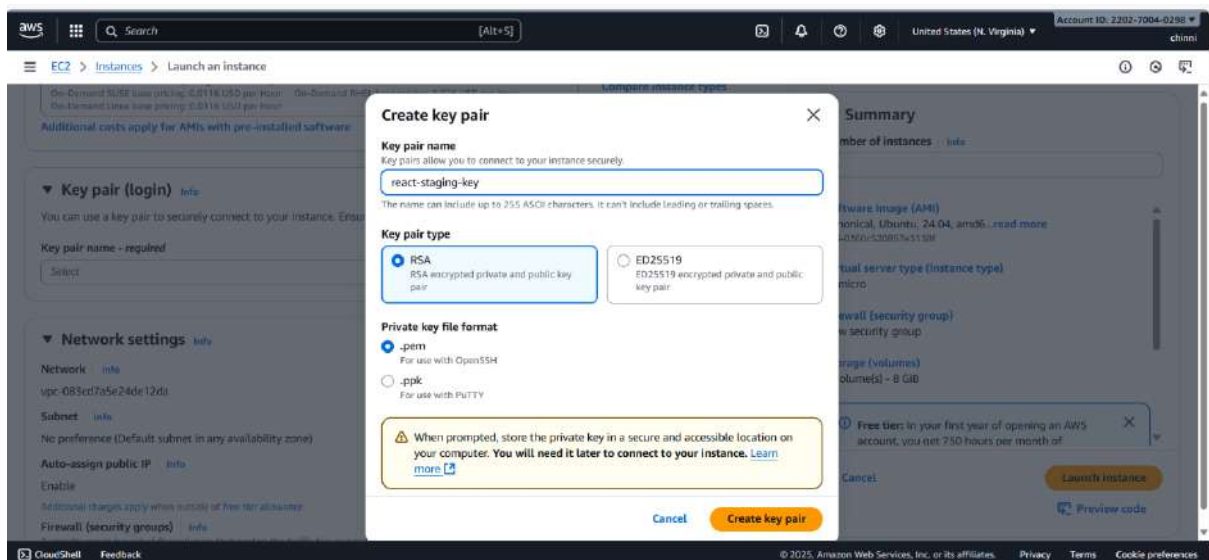
In EC2 Dashboard → left sidebar → **Instances** → **Launch instances** (button).

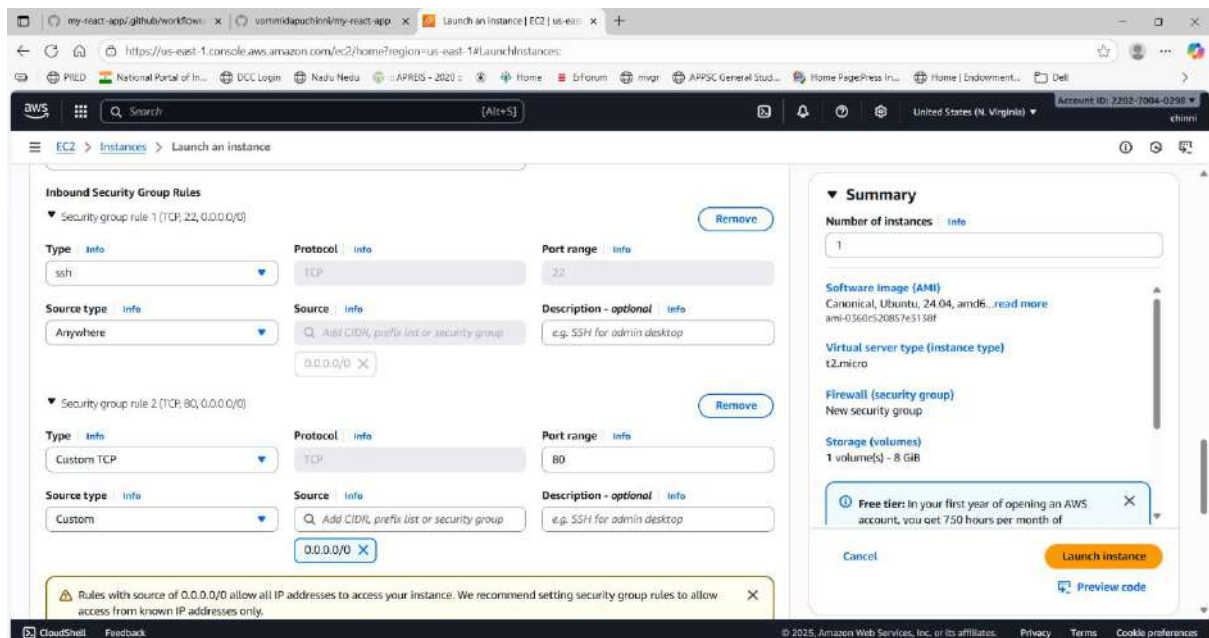
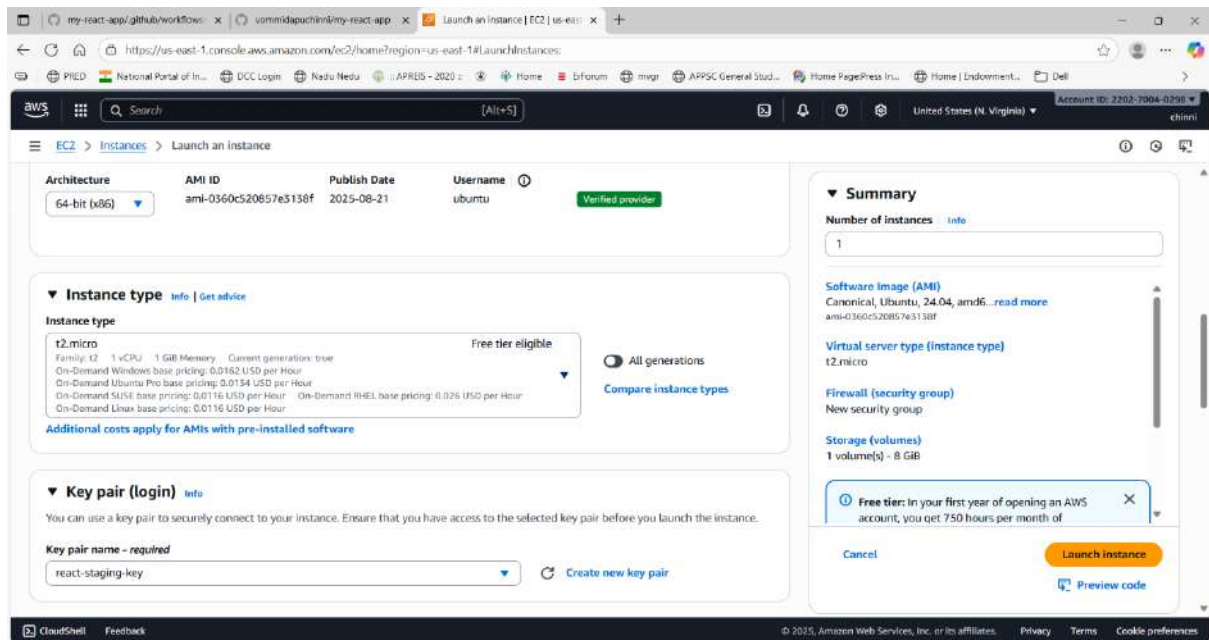
Launch instance wizard:

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

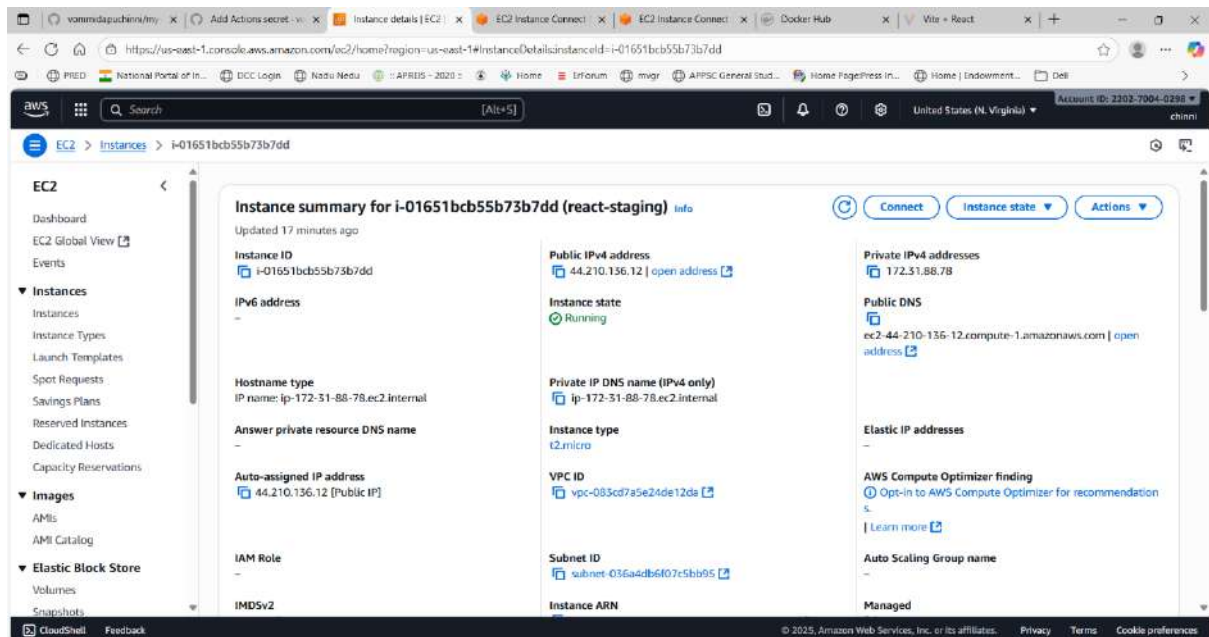


- Instance type: t2.micro (free tier eligible)
- Key pair (login): Create a new key pair → name it react-staging-key → Download .pem file and keep it safe (you'll use it to SSH).



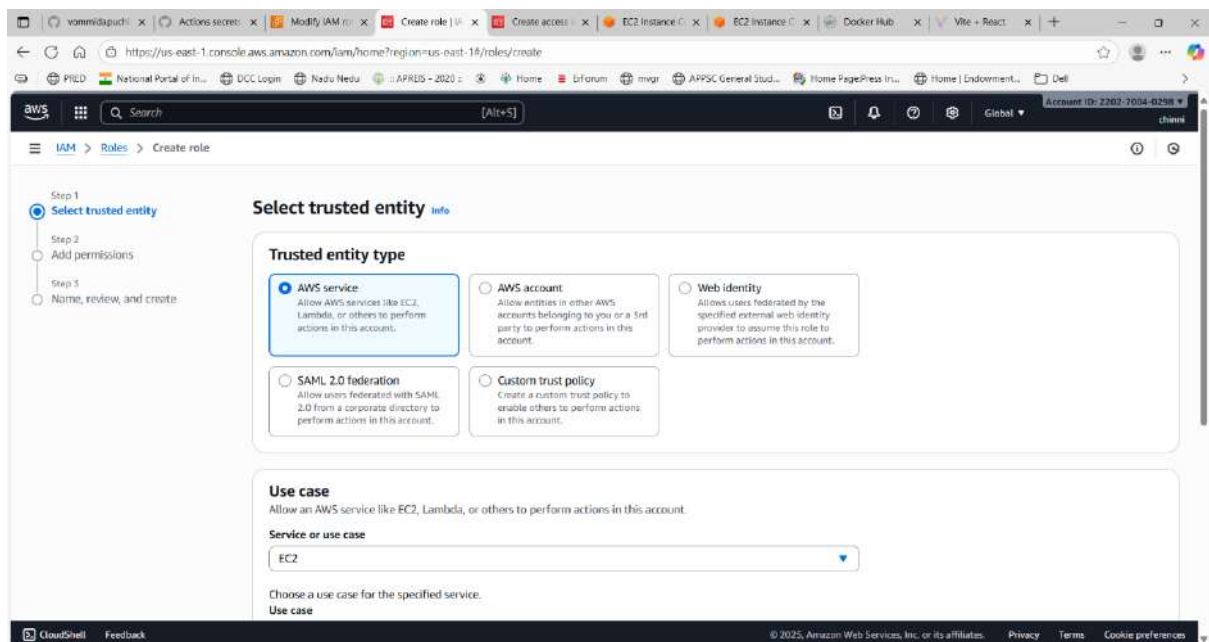


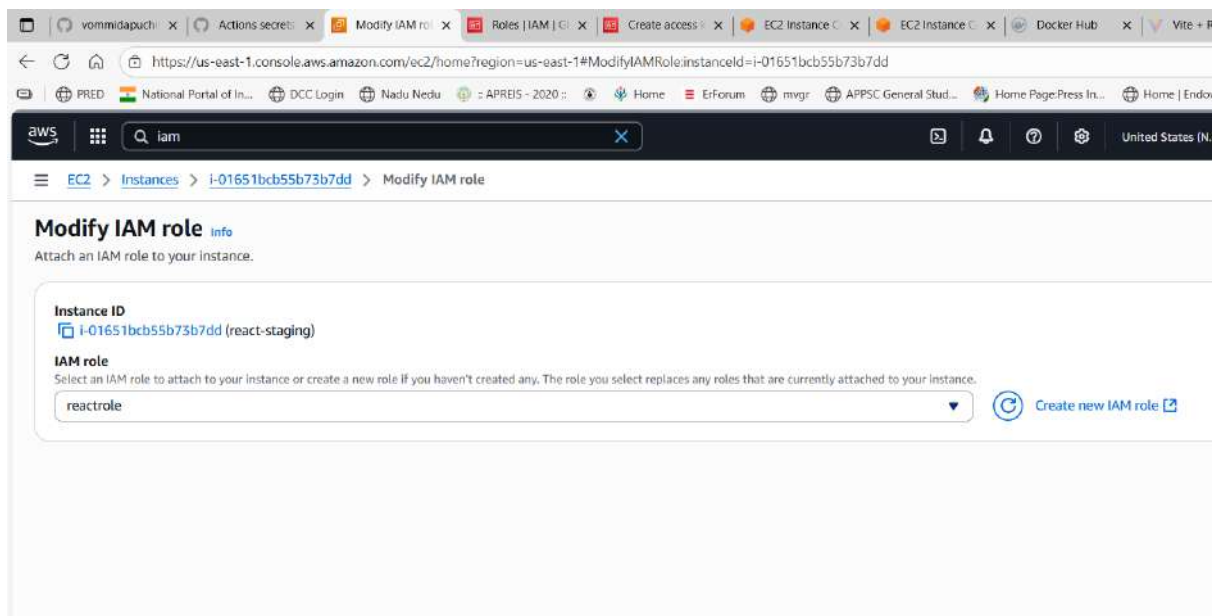
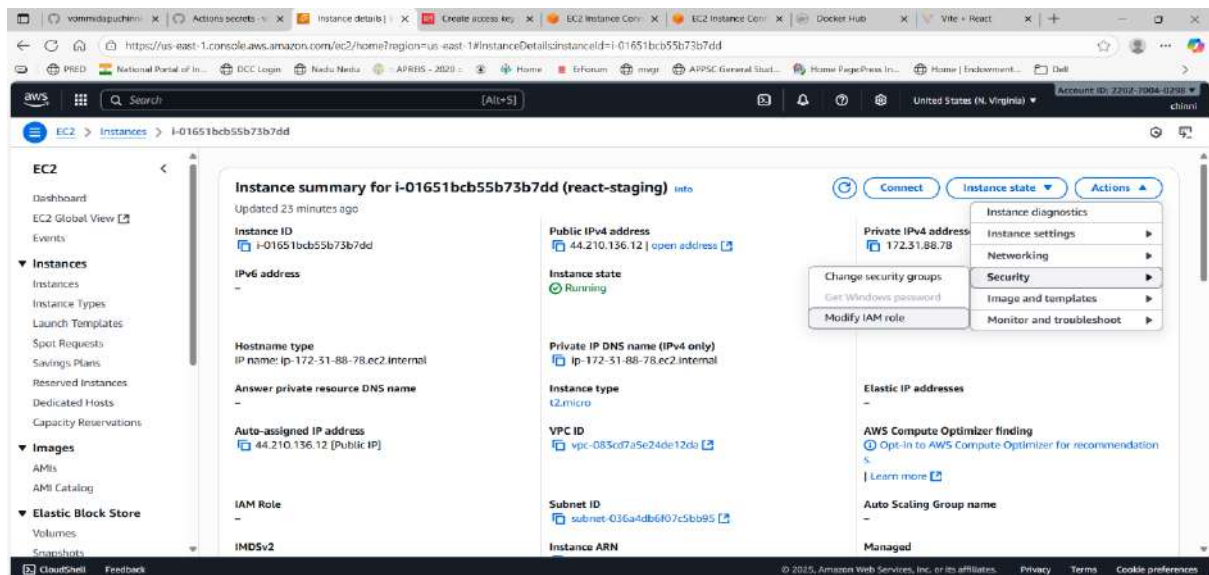
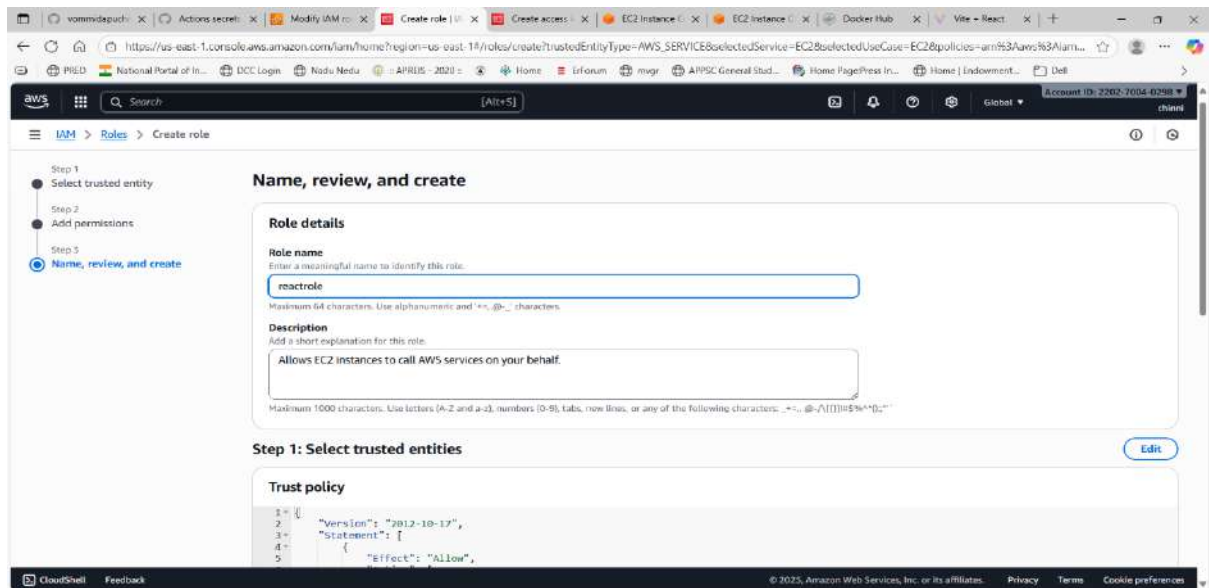
- Network settings (Security group): Create new security group with rules:
 - SSH (port 22) — Source: My IP (recommended) or your IP range
 - HTTP (port 80) — Source: Anywhere (0.0.0.0/0) (to allow web traffic)
 - Allow port 4000
- Click Launch instance.



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.





cloudwatch-staging.json created in root folder.

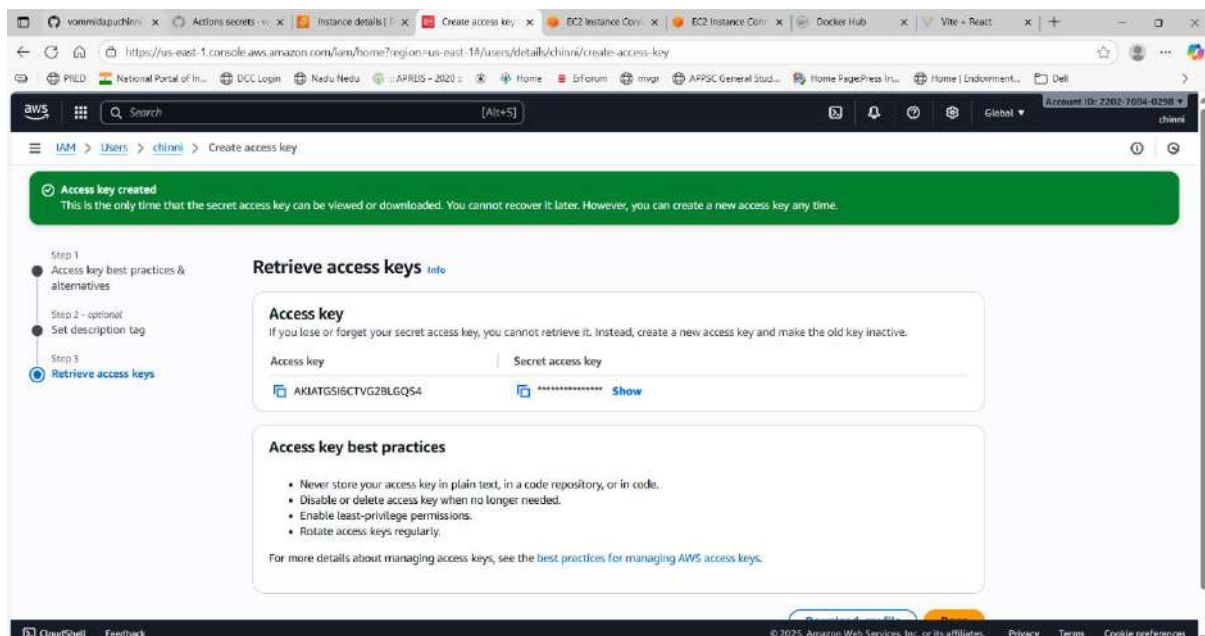
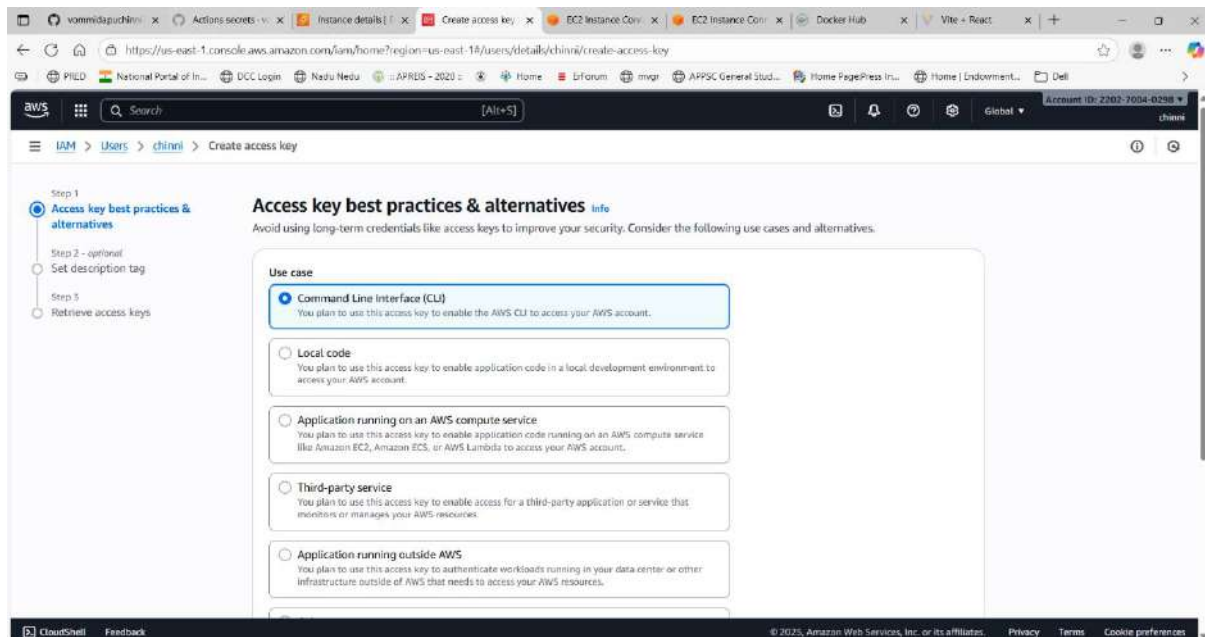
Start CloudWatch Agent : install cloud watch agent locally

```
sudo amazon-cloudwatch-agent-ctl -a fetch-config -m ec2 -c file:/home/ubuntu/my-react-app-staging/cloudwatch-staging.json -s
```

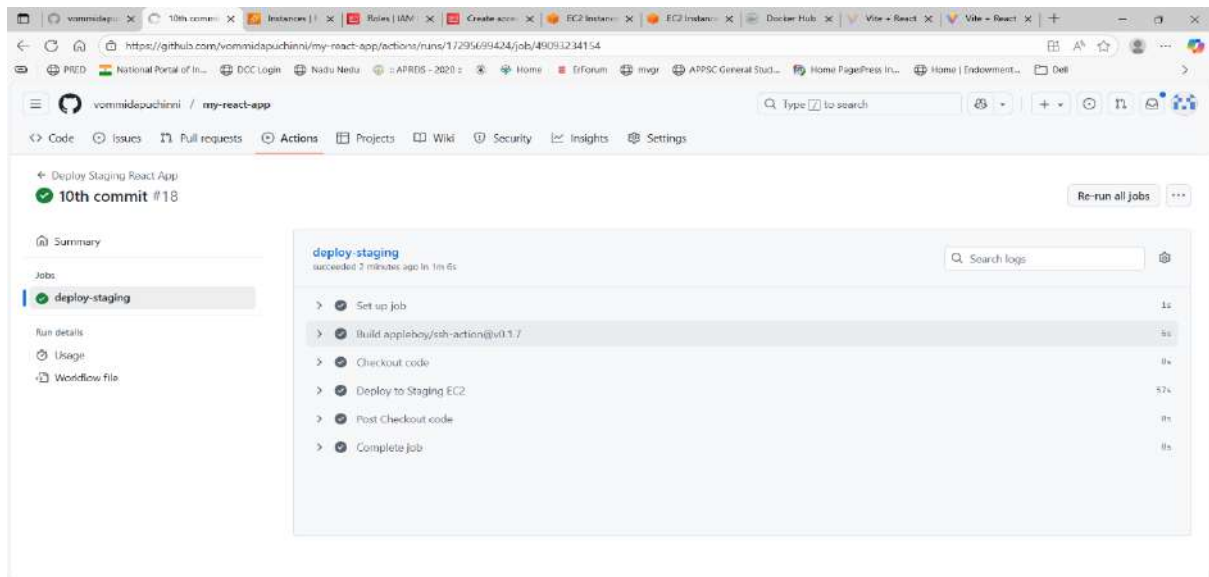
This will **start the agent** using the IAM role.

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

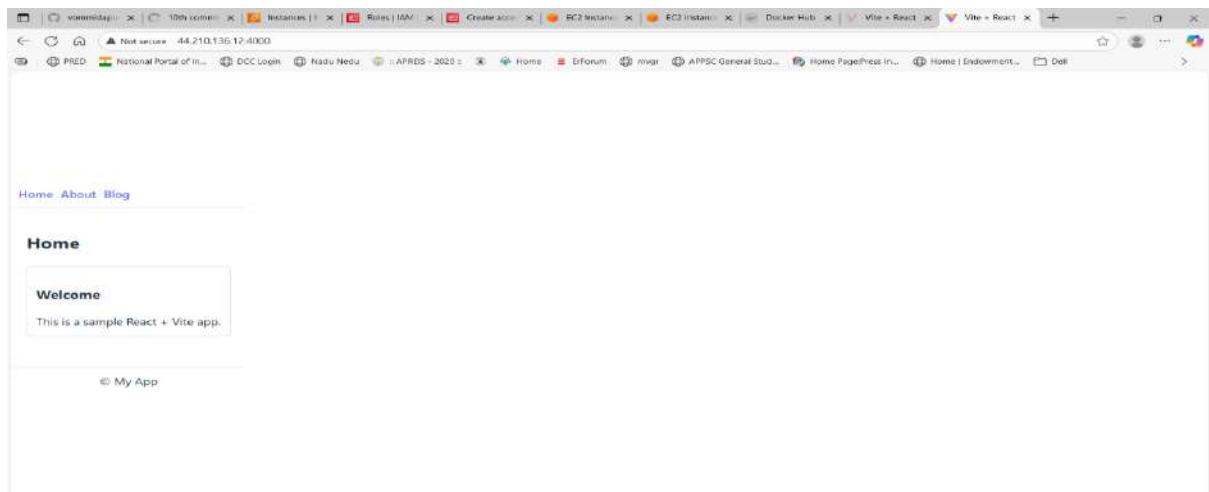
Created user and created keys for it and kept in github secrets



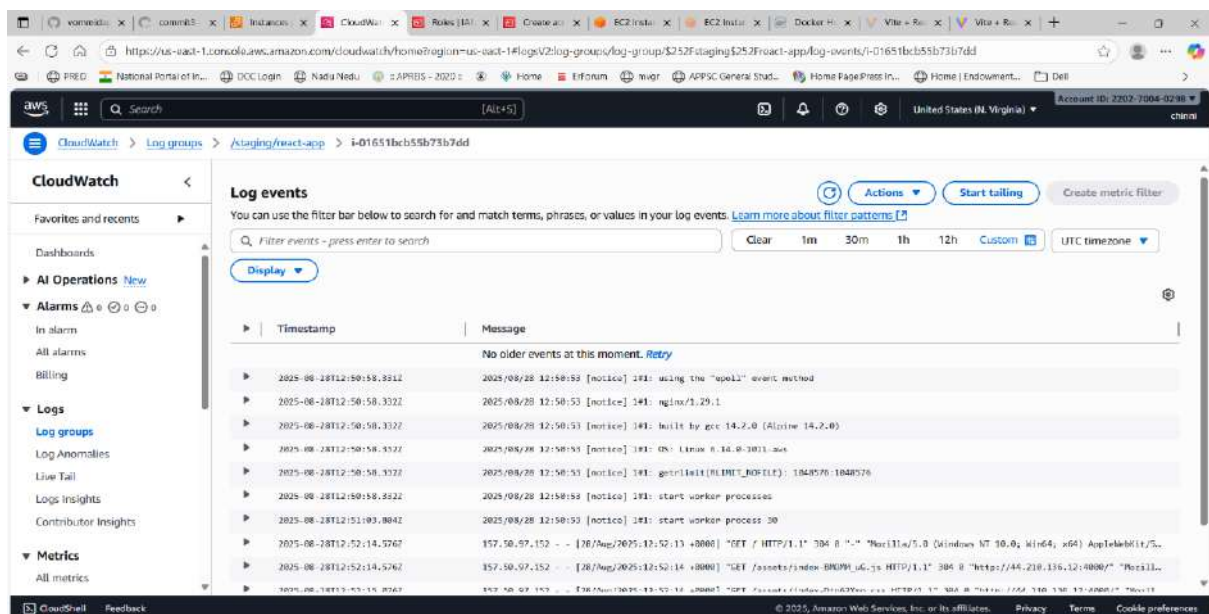
workflow



Accessed with ip



Cloud watch log groups



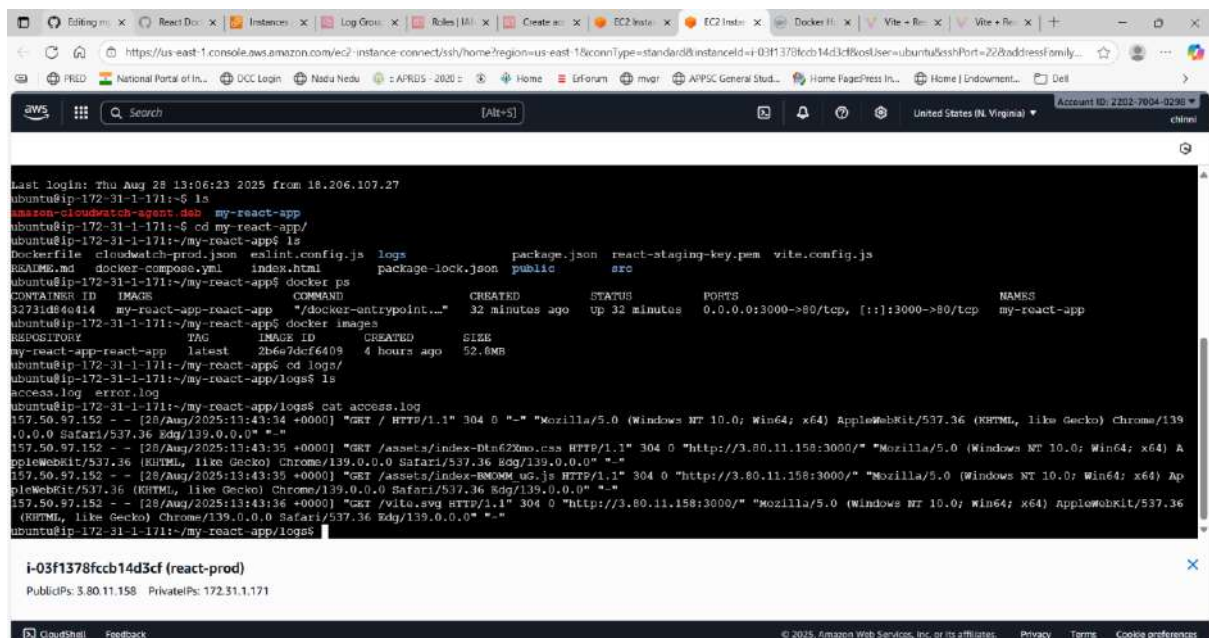
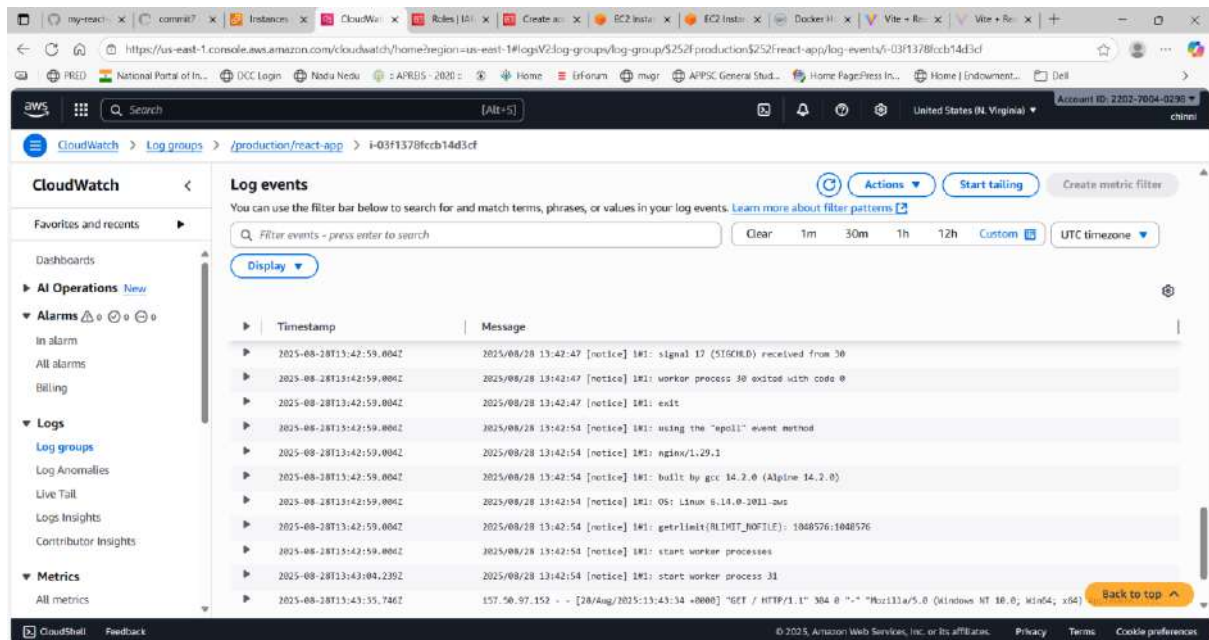
For production I did not keep cloud watch log group for that I am doing I have attached role to ec2 instance and created cloud watch log group and given loggroup folder in docker compose files in productions and pushed to github

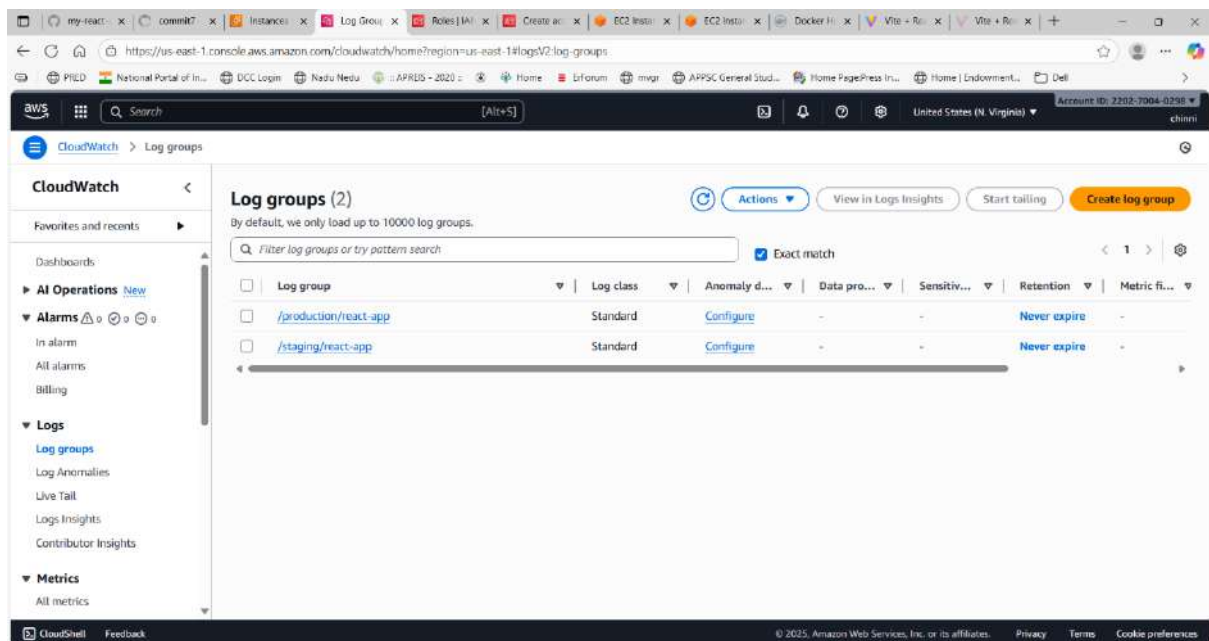
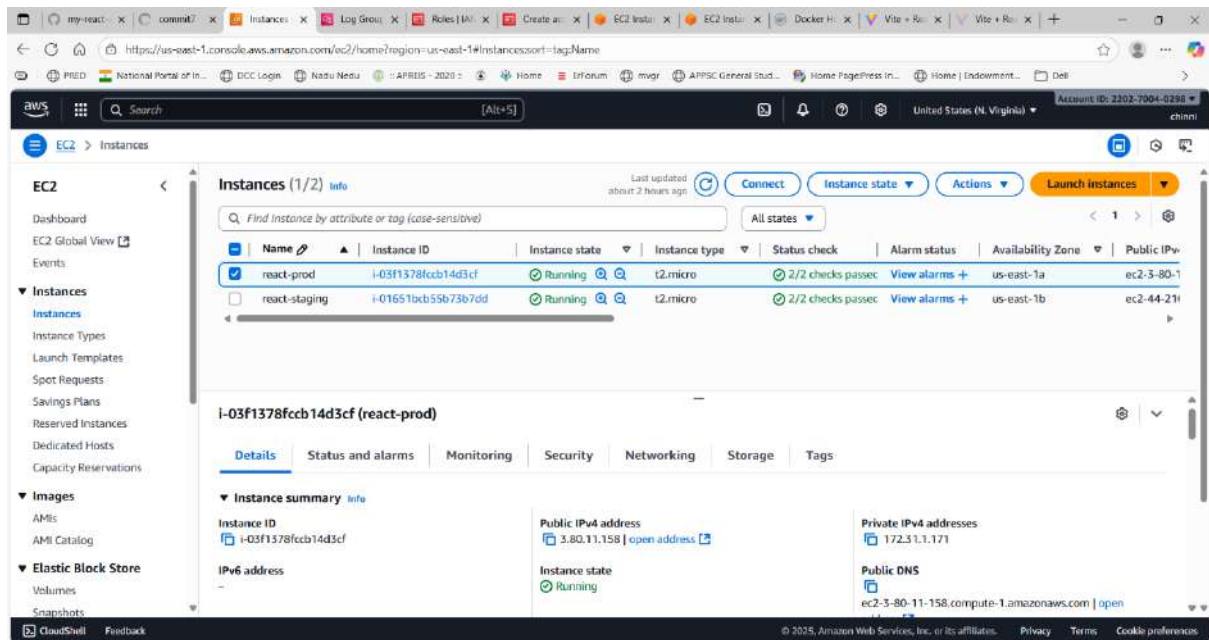
Create CloudWatch Configuration for Production: cloudwatch-prod.json

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


```
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-8MOMM_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

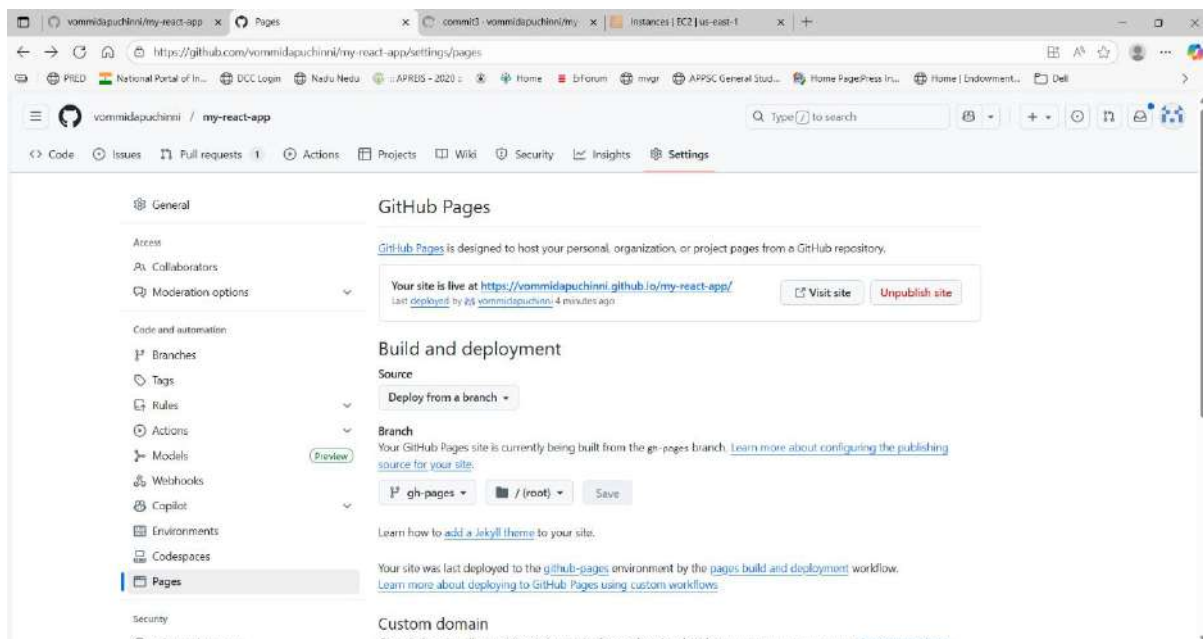
You are using Node.js 20.17.0. Vite requires Node.js version 20.19+ or 22.12+. Please upgrade your Node.js version.
vite v7.1.3 building for production...
✓ 47 modules transformed.
dist/index.html 0.47 kB | gzip: 0.31 kB
dist/assets/index-Dtn62Xmo.css 0.91 kB | gzip: 0.49 kB
dist/assets/index-8MOMM_uG.js 220.73 kB | gzip: 70.96 kB
✓ built in 3.70s

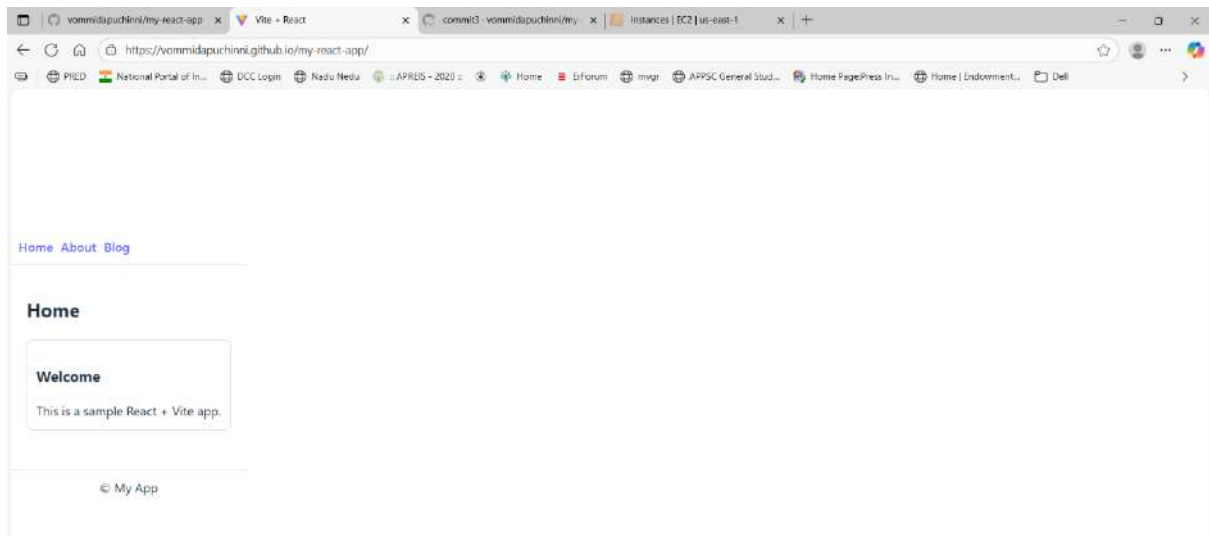
> my-react-app@0.0.0 deploy
> gh-pages -d dist

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`

```
CS\WINDOWS\system32\cmd.exe
dist/index.html 0.51 kB | gzip: 0.32 kB
dist/assets/index-Dtn62Xmo.css 0.91 kB | gzip: 0.49 kB
dist/assets/index-8CMcTlb4.js 221.11 kB | gzip: 71.10 kB
v built in 4.06s

umama@DESKTOP-HBL3M57 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 0.51 kB | gzip: 0.32 kB
dist/assets/index-Dtn62Xmo.css 0.91 kB | gzip: 0.49 kB
dist/assets/index-8CMcTlb4.js 221.11 kB | gzip: 71.10 kB
v built in 3.89s

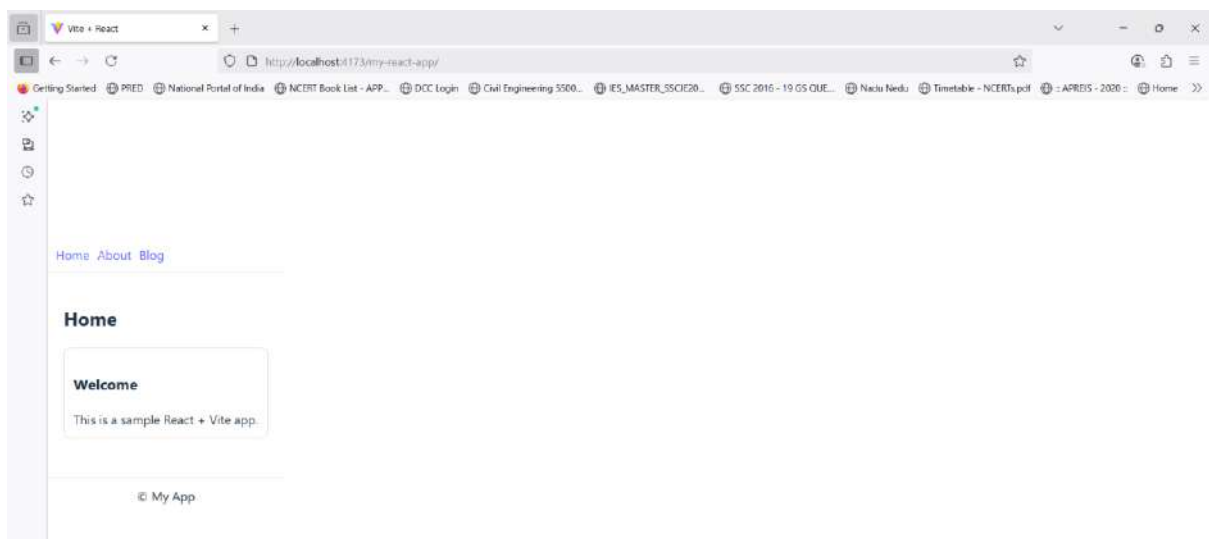
> my-react-app@0.0.0 deploy
> gh-pages -d dist

Published

umama@DESKTOP-HBL3M57 MINGW64 ~/my-react-app (main)
$ npm run preview
> my-react-app@0.0.0 preview
> vite preview

You are using Node.js 20.17.0. Vite requires Node.js version 20.19+ or 22.12+. Please upgrade your Node.js version.
→ Local: http://localhost:4173/my-react-app/
→ Network: use --host to expose
→ press h + enter to show help
```

- 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.