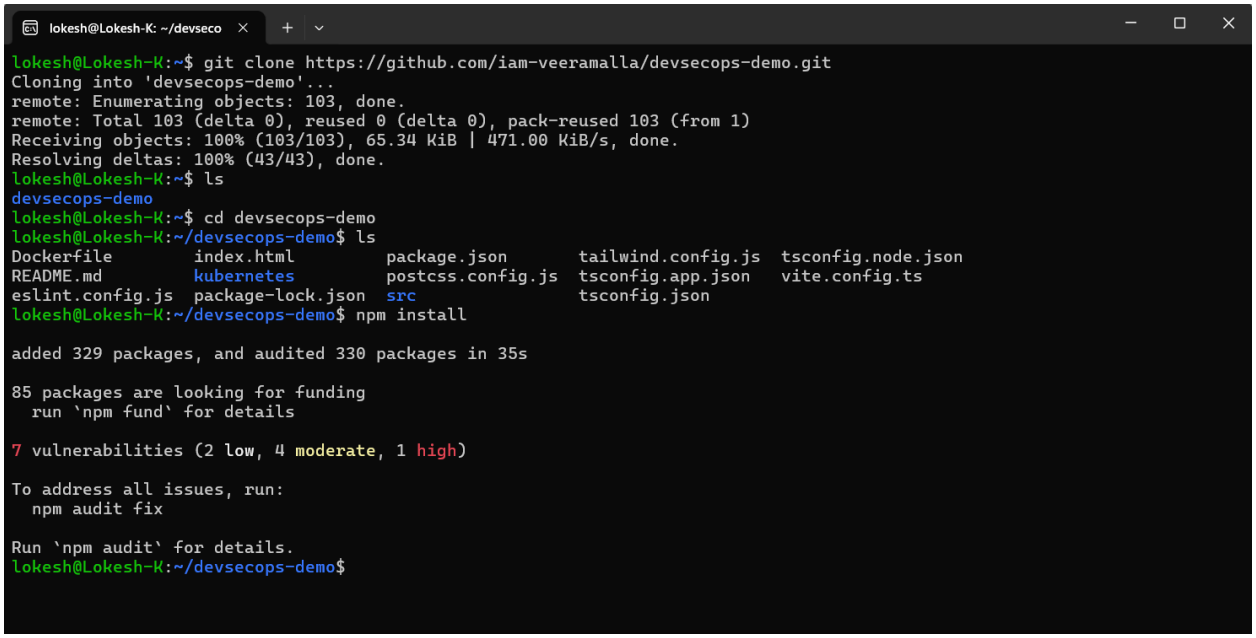
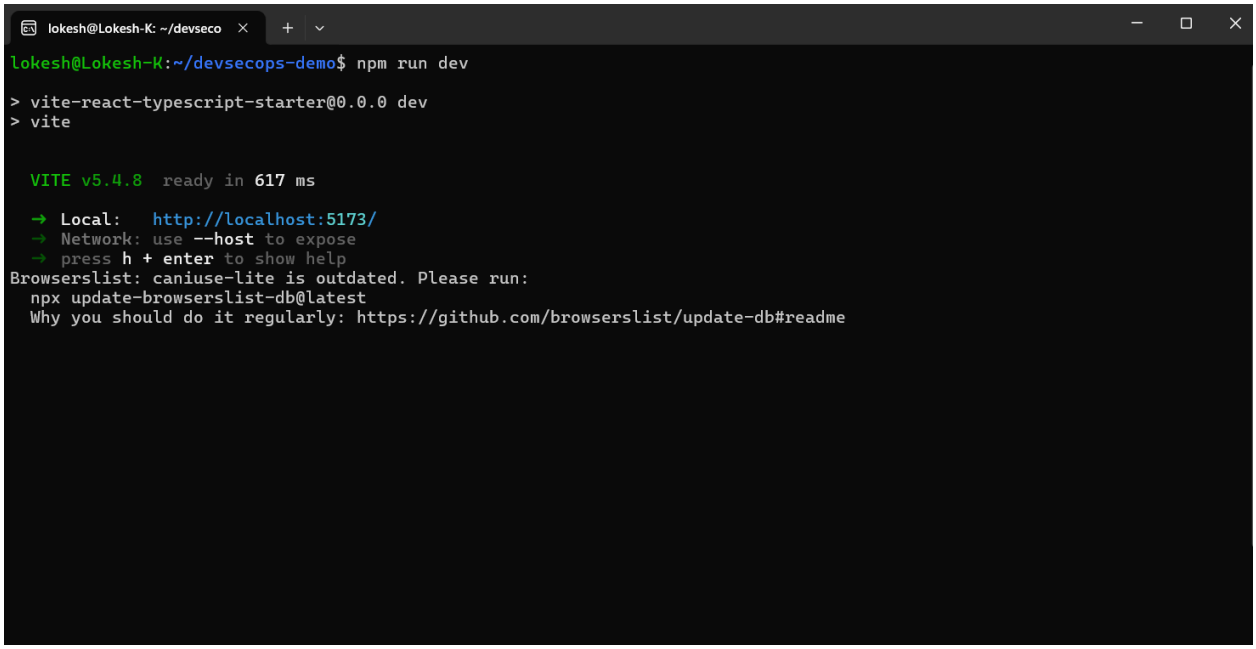


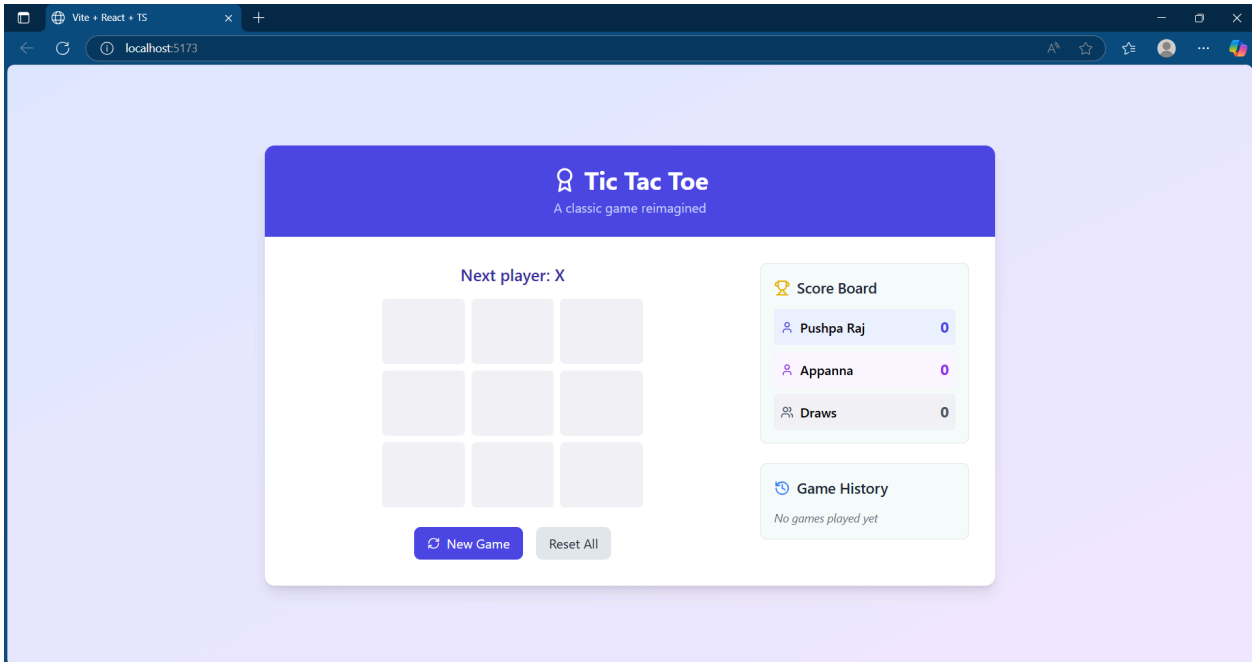
1.Cloned the project repository from GitHub

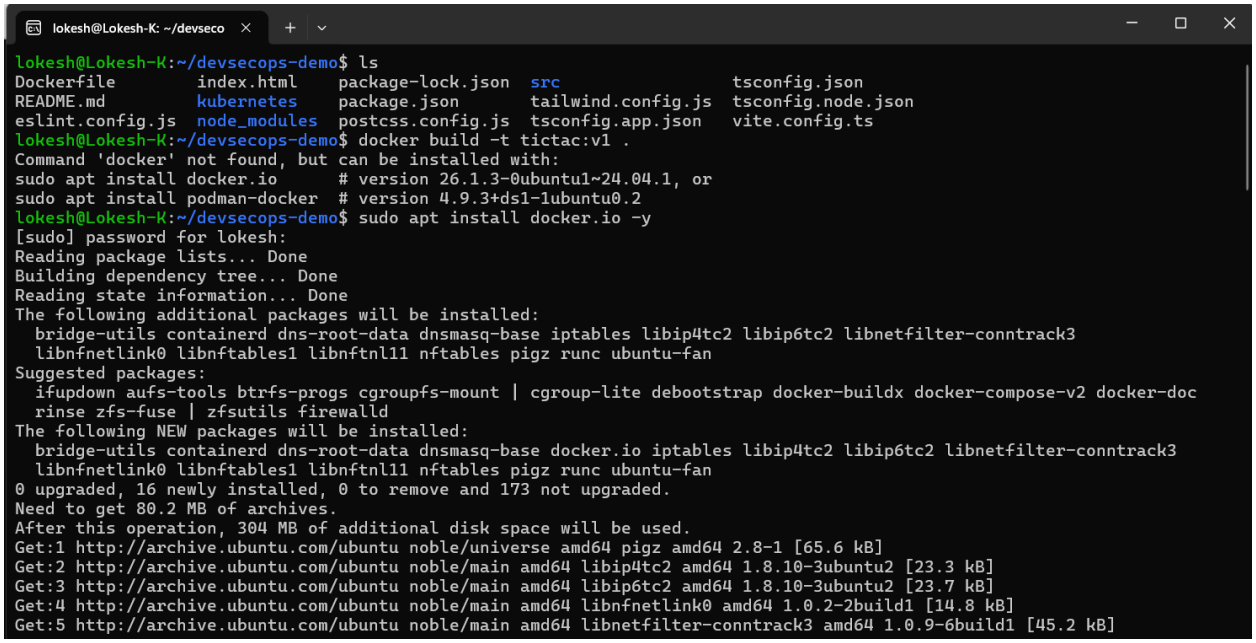
2.Navigated into the cloned project directory

3.Installed all dependencies using - npm install



4.Ran the development server using **npm run dev**





Running the App and Preparing Docker Build :

1.Verified App Running Locally:

- Accessed the running app on `http://localhost:5173/`
- Confirmed the **Tic Tac Toe** game UI loads properly with score board, game board, and control buttons.

2.Prepared for Docker Build:

- Listed project files using `ls` to ensure the presence of a `Dockerfile`.
- Attempted to build a Docker image named `tictac:v1` using:

❌ **Error Faced:**

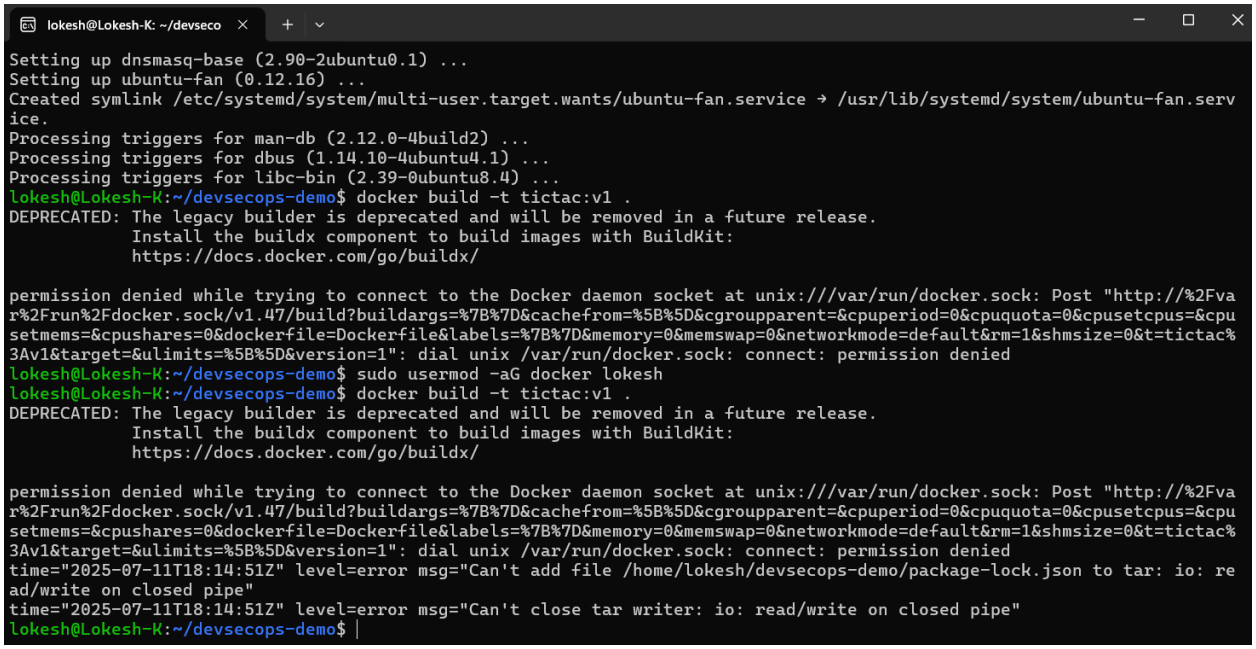
- **`docker` command not found** – The system did not have Docker installed.

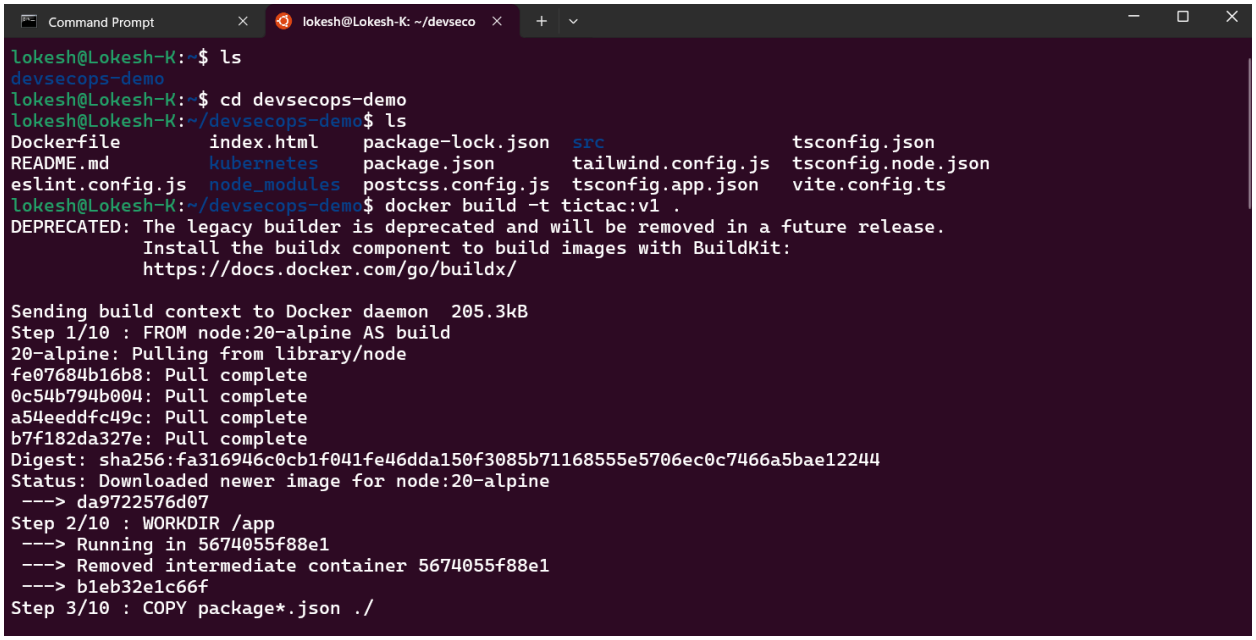
So i Installed Docker using **apt** :

```
sudo apt install docker.io -y
```

This installed the necessary Docker packages, including `docker-compose`, `buildx`, and other dependencies.

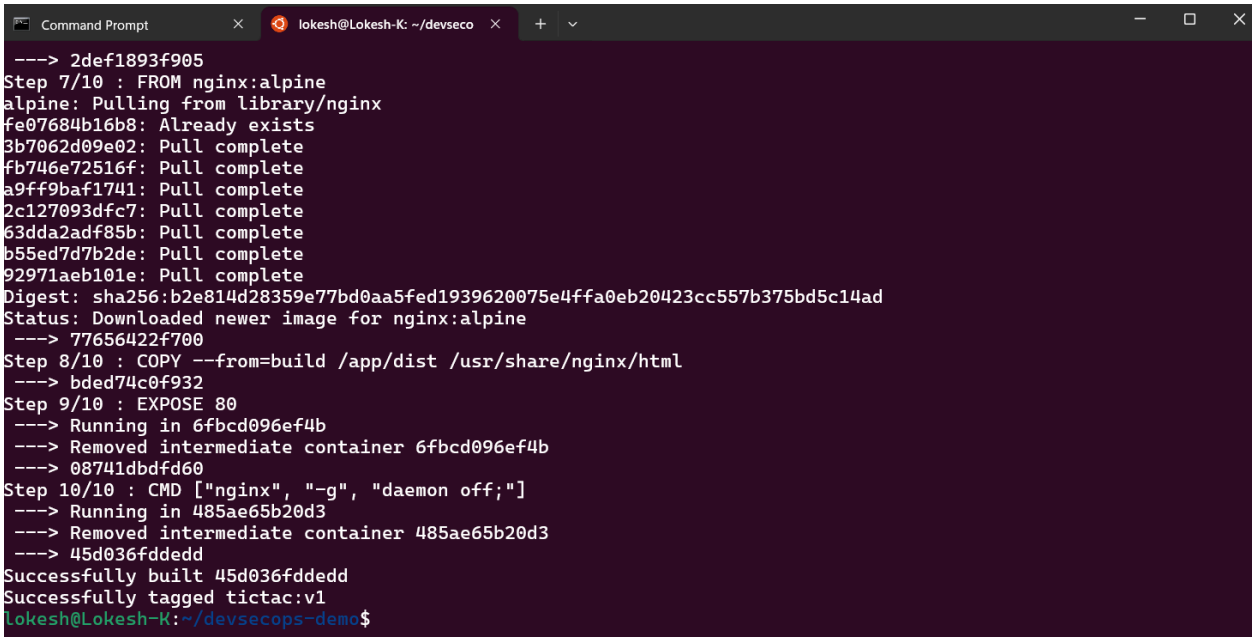
After installation, Docker was ready to build the image.

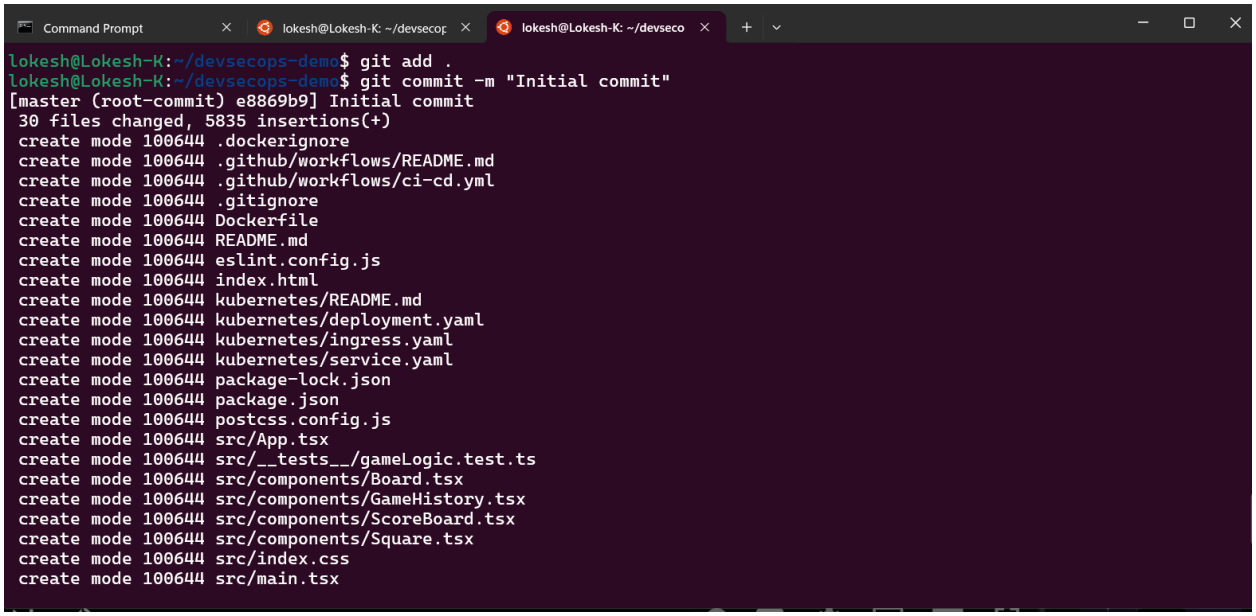


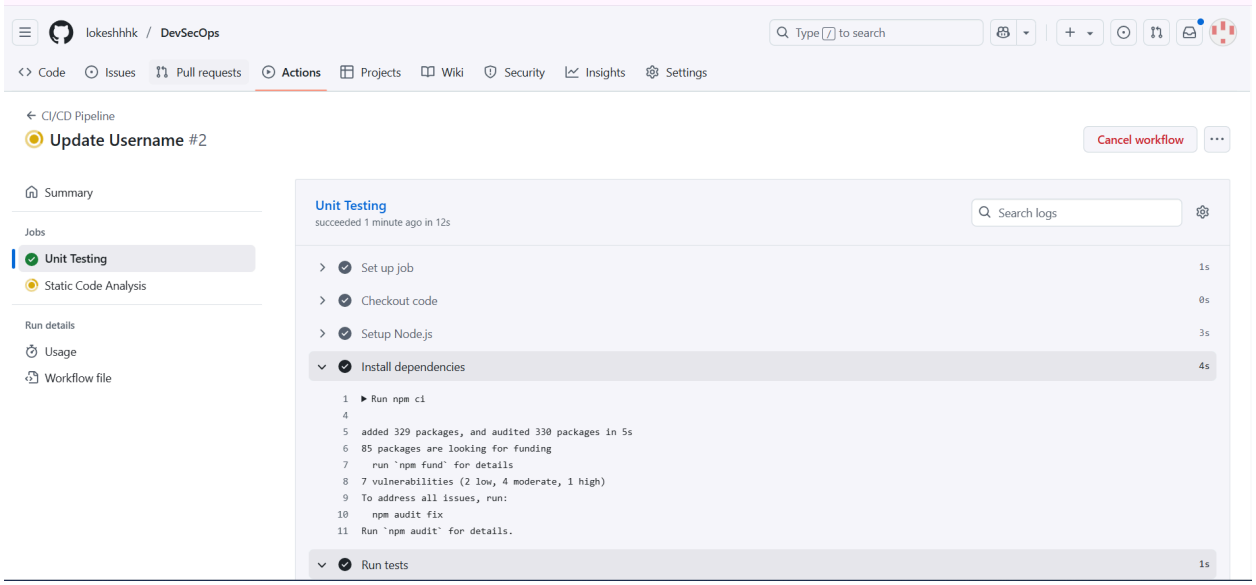


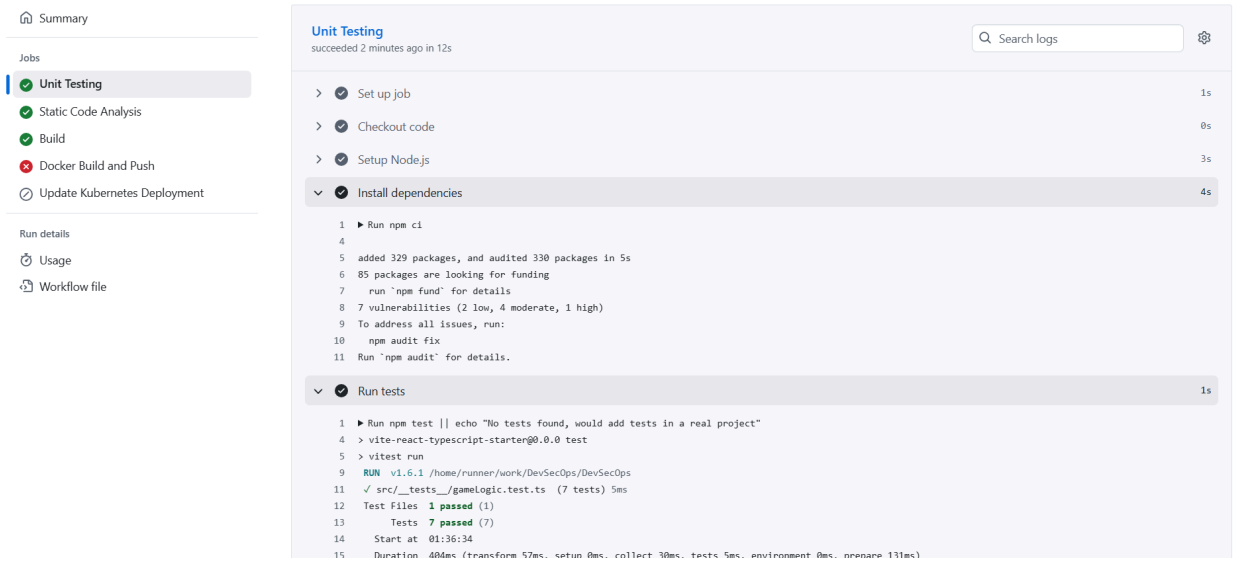
Executed Docker build command:

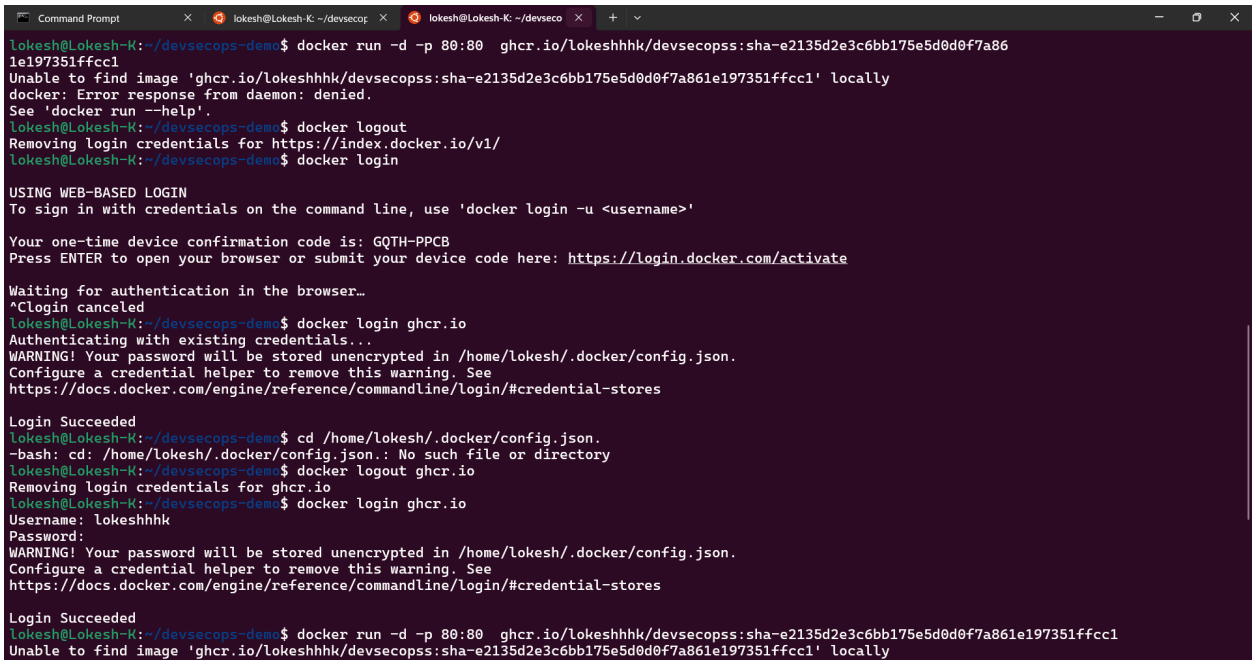
docker build -t tictac:v1 .




Command Prompt
lokesh@Lokesh-K: ~/devsecop
lokesh@Lokesh-K: ~/devseco
lokesh@Lokesh-K:~/devsecops-demo$ git add .
lokesh@Lokesh-K:~/devsecops-demo$ git commit -m "Initial commit"
[master (root-commit) e8869b9] Initial commit
30 files changed, 5835 insertions(+)
create mode 100644 .dockerignore
create mode 100644 .github/workflows/README.md
create mode 100644 .github/workflows/ci-cd.yml
create mode 100644 .gitignore
create mode 100644 Dockerfile
create mode 100644 README.md
create mode 100644 eslint.config.js
create mode 100644 index.html
create mode 100644 kubernetes/README.md
create mode 100644 kubernetes/deployment.yaml
create mode 100644 kubernetes/ingress.yaml
create mode 100644 kubernetes/service.yaml
create mode 100644 package-lock.json
create mode 100644 package.json
create mode 100644 postcss.config.js
create mode 100644 src/App.tsx
create mode 100644 src/__tests__/gameLogic.test.ts
create mode 100644 src/components/Board.tsx
create mode 100644 src/components/GameHistory.tsx
create mode 100644 src/components/ScoreBoard.tsx
create mode 100644 src/components/Square.tsx
create mode 100644 src/index.css
create mode 100644 src/main.tsx


lokeshhhk / DevSecOps
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
CI/CD Pipeline
Update Username #2
Cancel workflow
Summary
Jobs
Unit Testing
Static Code Analysis
Run details
Usage
Workflow file
Unit Testing
succeeded 1 minute ago in 12s
Search logs
Set up job 1s
Checkout code 0s
Setup Node.js 3s
Install dependencies 4s
1 ▶ Run npm ci
4
5 added 329 packages, and audited 330 packages in 5s
6 85 packages are looking for funding
7 run `npm fund` for details
8 7 vulnerabilities (2 low, 4 moderate, 1 high)
9 To address all issues, run:
10 npm audit fix
11 Run `npm audit` for details.
Run tests 1s



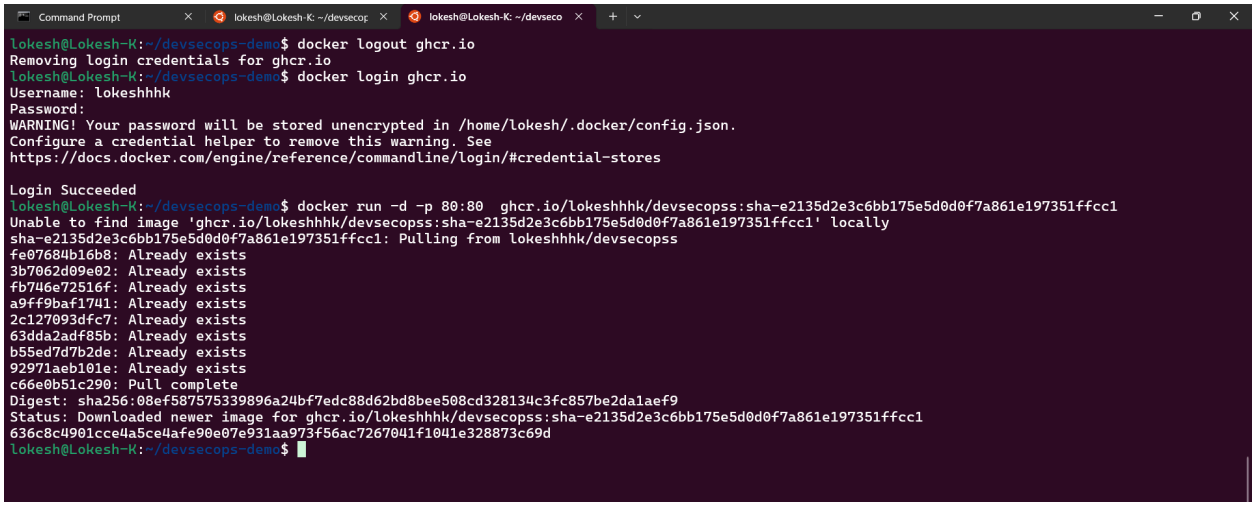


This error occured because if the image does not present locally it has to pulled from docker registry here in this case we used a Github container registry.

Then why the error occurred means when we try to login to docker registry it asks credentials Username and for password we need to give access token

I first logged in with docker with access token that does not have the ability of read packages So it wasn't able to fetch from that.

Debug : Login with the credentials with username and access token with read write packages enabled.

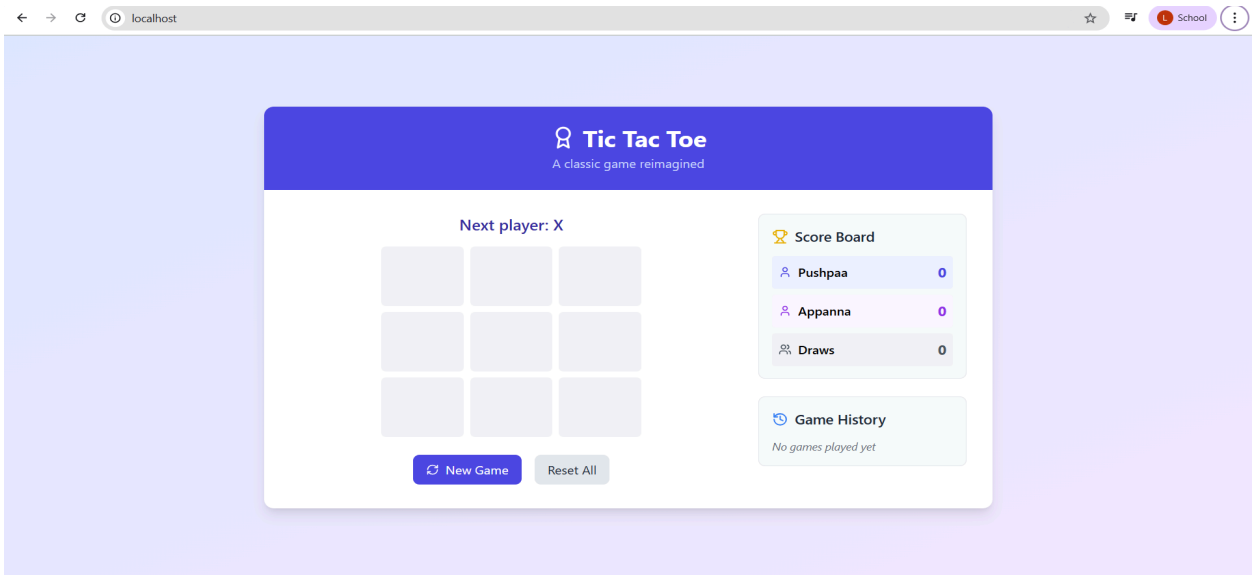


Logged In to Docker with GHCR Using Personal Access Token (PAT):

docker login ghcr.io

Username: lokeshhhh

Password: Used a GitHub Personal Access Token (PAT).



Pulled Image from GHCR:

`docker run -d -p 80:80 ghcr.io/lokeshhhh/devsecopss:<sha-tag>`

Pulled a specific image using its SHA digest tag from the GitHub Container Registry. Image layers already existed locally, so Docker reused them.

Confirmed That Application Was Running:

- Opened browser at `http://localhost:80`
- Verified that the Tic Tac Toe application loaded successfully.

❌ Error Faced:

Initially got:
`Unable to find image 'ghcr.io/lokeshhhh/devsecopss:sha=...' locally`

- This was expected, as Docker tried to pull it from GHCR.

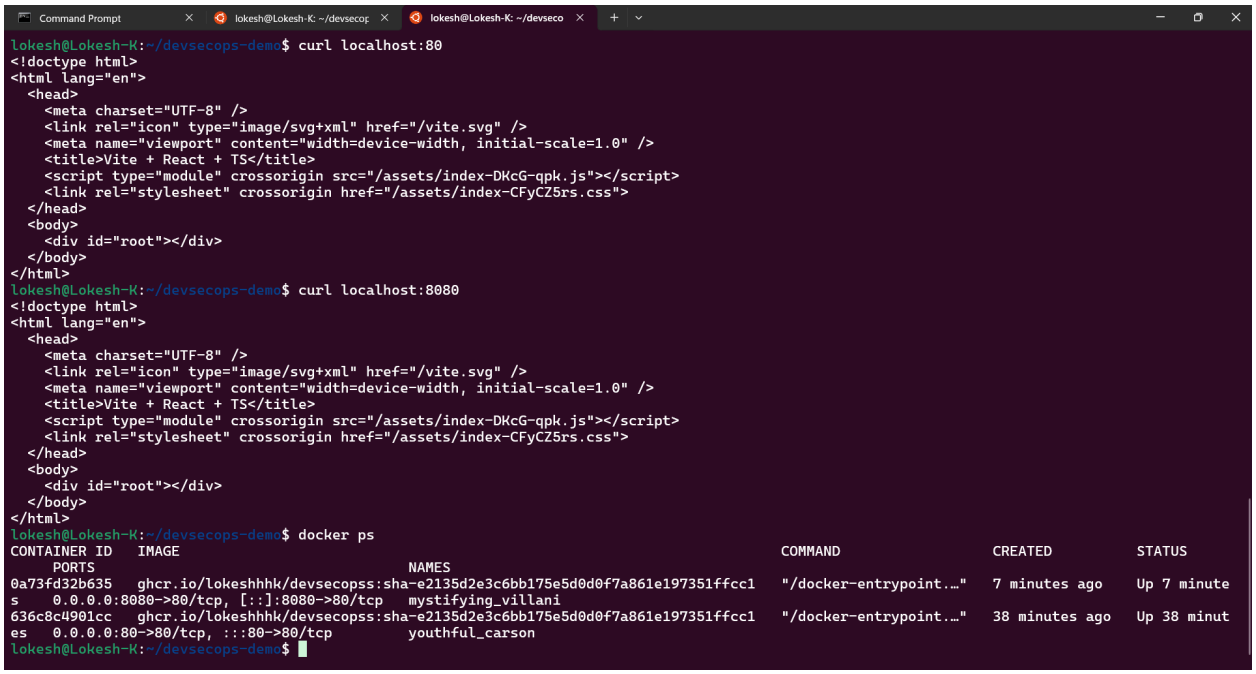
🛠️ Debugging :

- Ensured the correct image URL was used, with valid SHA digest.
- Verified Personal Axcess Token used during login had `read:packages` permission for GHCR.
- Once image was successfully pulled, the container started and the app was accessible via the browser.

Pulled and Ran Docker Image from GitHub Container Registry:

`docker run -d -p 8080:80 ghcr.io/lokeshhhh/devsecopss:<sha>`

- This launched the Docker container on port `8080` (host) mapped to port `80` (container).
- The image was successfully pulled and container started.

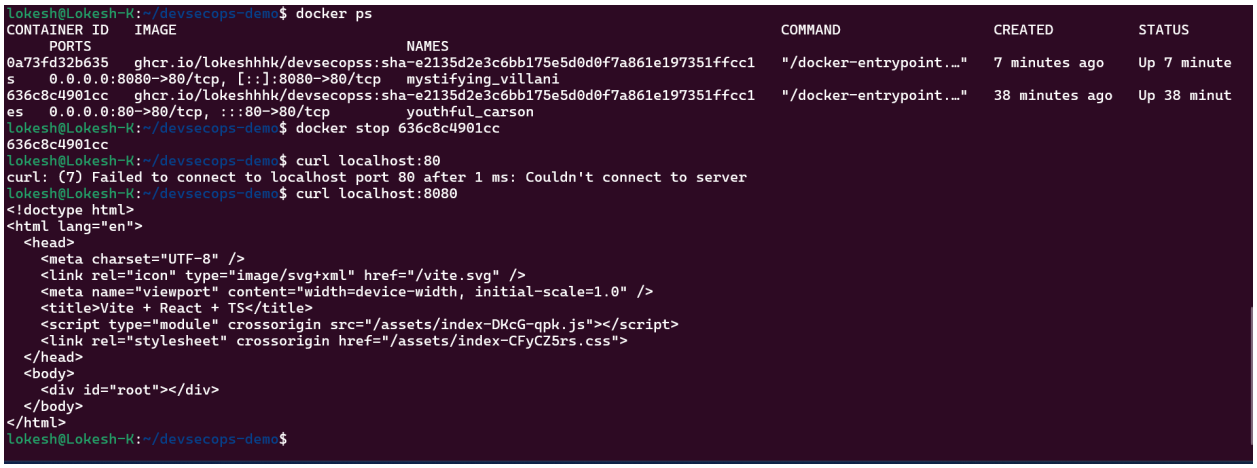


Verified Application Using `curl`:

```
curl localhost:80
```

```
curl localhost:8080
```

- Both commands returned raw HTML source of the React app, indicating that the container is serving content properly.



1. **Verified HTML Output Using `curl`:**

Ran both:
`curl localhost:80`
`curl localhost:8080`

   - Received correct HTML content from both ports, confirming both containers were serving the React app.

2. **Checked Running Docker Containers:**

Used:
`docker ps`
Confirmed two containers running the **same image**:

   - One exposed on port `8080:80`
   - One on port `80:80`

3. **Stopped One Container to Test Port Behavior:**

Stopped the container mapped to port 80:
`docker stop <container_id>`
Verified the result using:
`curl localhost:80`
`curl localhost :8080`
Got error:
`curl: (7) Failed to connect to localhost port 80: Couldn't connect to server`

   - But `curl localhost:8080` continued to work, confirming only the `80`-mapped container was stopped.

❌ **Error Faced:**

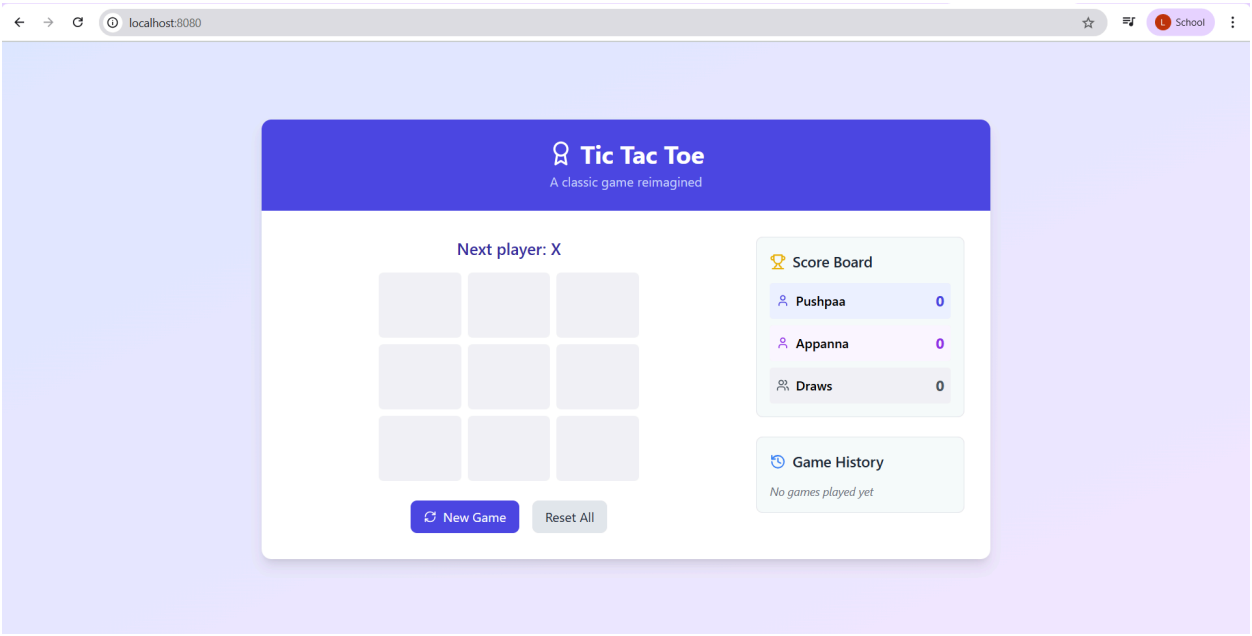
- After stopping the container listening on port 80, a curl request to `localhost:80` failed — **as expected**.

🛠️ **Debugging:**

- Used `docker ps` to identify port mappings and container statuses.
- Used `docker stop` to isolate and test container access behavior.
- Used `curl` on both ports to verify which port was still serving the app.

💡Output:

This shows how **Docker port mappings work**. If a container is mapped to `8080:80`, only `localhost:8080` will work, and not `localhost:80`, unless another container explicitly exposes that.



```
</html>
lokesh@Lokesh-K:~/devsecops-demo$ # For AMD64 / x86_64
[ $(uname -m) = x86_64 ] && curl -Lo ./kind https://kind.sigs.k8s.io/dl/v0.29.0/kind-linux-amd64
# For ARM64
[ $(uname -m) = aarch64 ] && curl -Lo ./kind https://kind.sigs.k8s.io/dl/v0.29.0/kind-linux-arm64
chmod +x ./kind
sudo mv ./kind /usr/local/bin/kind
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    97  100    97    0     0     78      0  0:00:01  0:00:01 --:--:--    78
  0     0    0     0    0     0      0      0 --:--:--  0:00:02 --:--:--     0
100 10.5M  100 10.5M    0     0   717k      0  0:00:15  0:00:15 --:--:--  779k
[sudo] password for lokesh:
lokesh@Lokesh-K:~/devsecops-demo$ kind
kind creates and manages local Kubernetes clusters using Docker container 'nodes'

Usage:
  kind [command]

Available Commands:
  build       Build one of [node-image]
  completion  Output shell completion code for the specified shell (bash, zsh or fish)
  create      Creates one of [cluster]
  delete      Deletes one of [cluster]
  export      Exports one of [kubeconfig, logs]
  get         Gets one of [clusters, nodes, kubeconfig]
  help        Help about any command
  load        Loads images into nodes
  version     Prints the kind CLI version

Flags:
  -h, --help              help for kind
  -q, --quiet             silence all stderr output
  -v, --verbosity int32   info log verbosity, higher value produces more output
      --version           version for kind

Use "kind [command] --help" for more information about a command.
lokesh@Lokesh-K:~/devsecops-demo$ clear
```

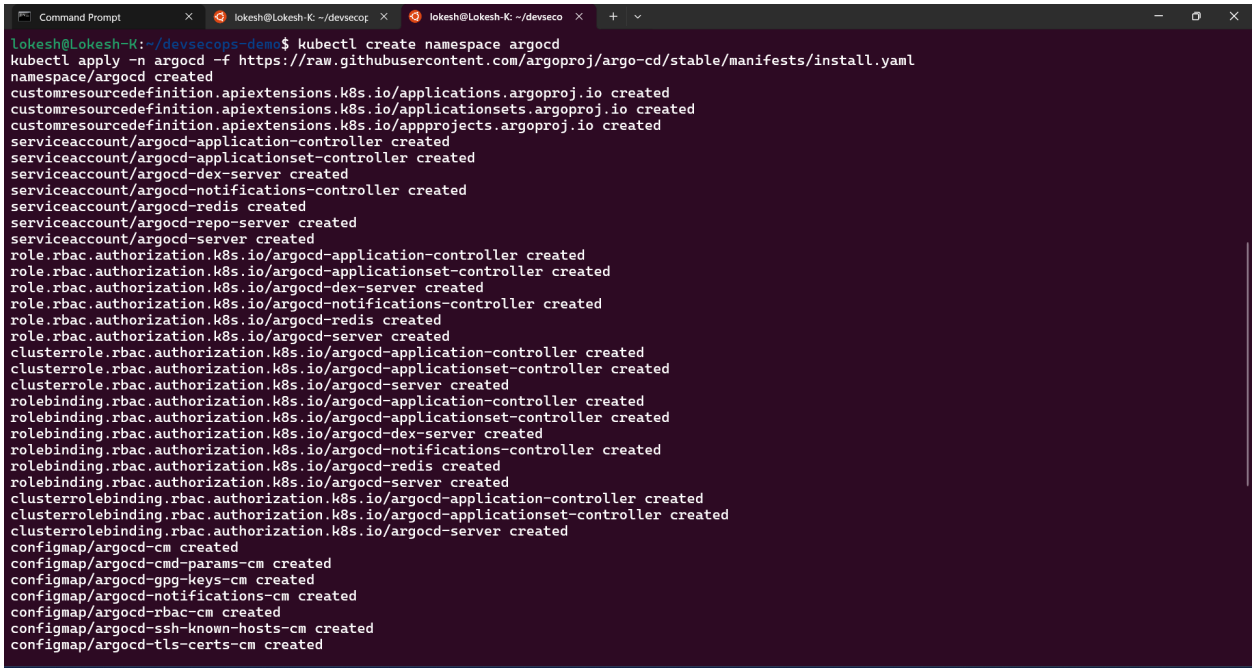
Created a Kubernetes Cluster Using KIND:

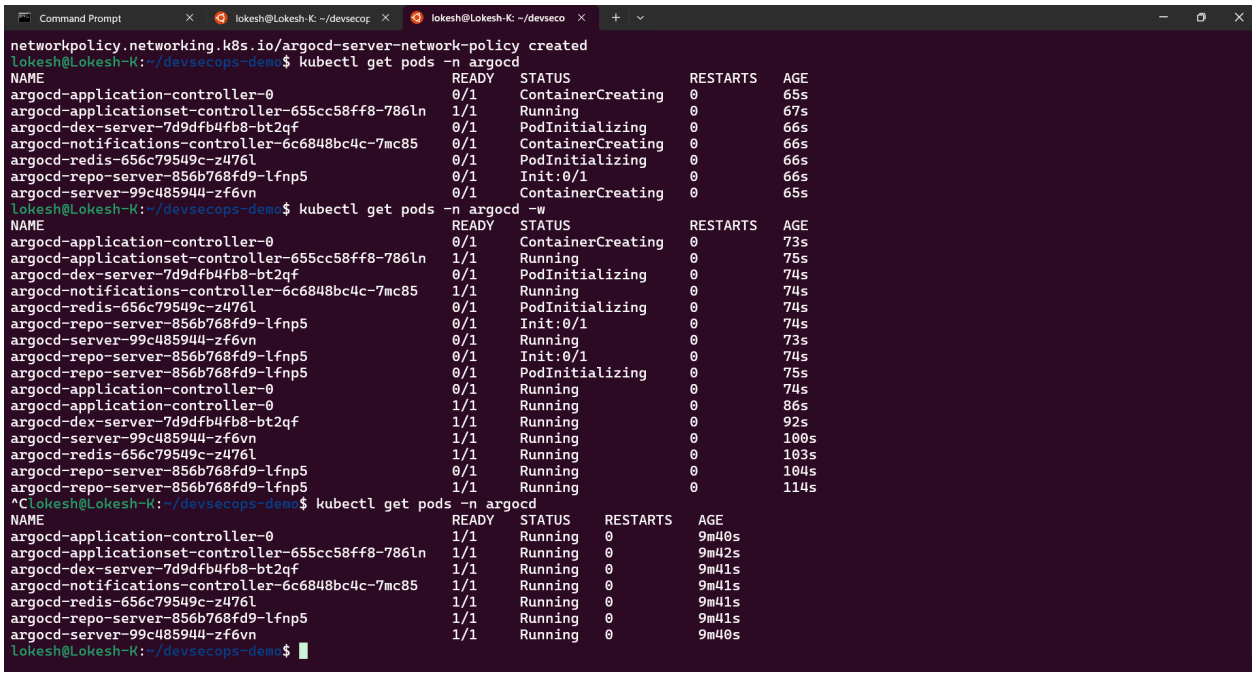
```
kind create cluster --name=devsecops-cluster
```

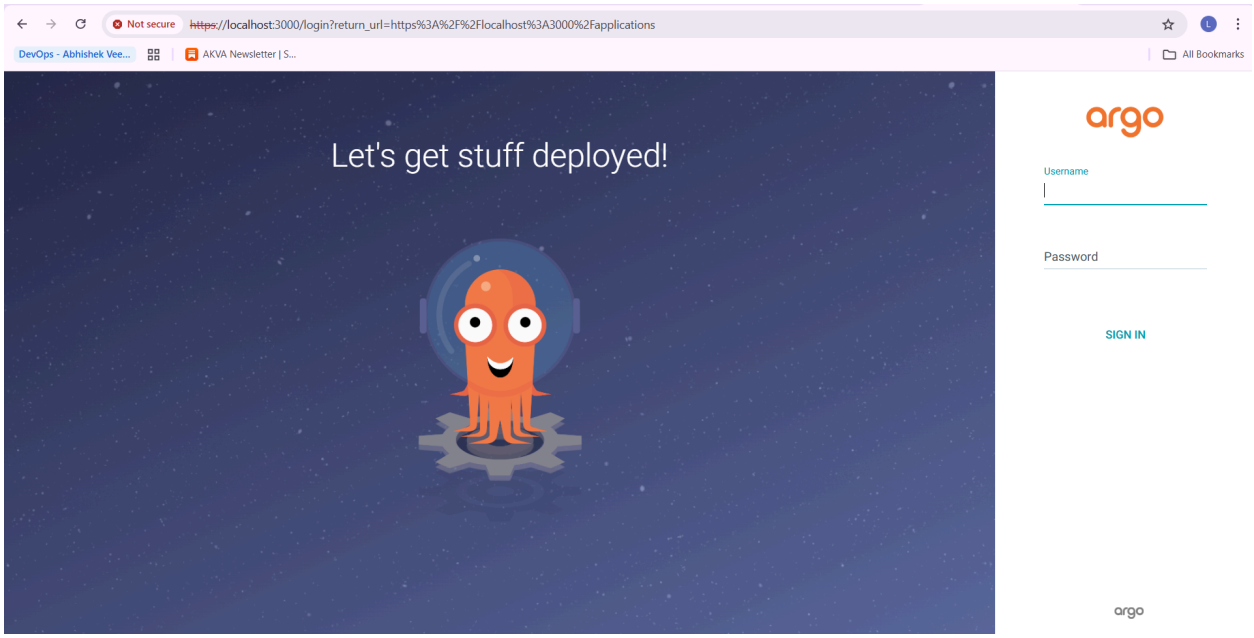
```
Command Prompt | lokesh@Lokesh-K: ~/devsecop | lokesh@Lokesh-K: ~/devseco
lokesh@Lokesh-K:~/devsecops-demo$ kind create cluster --name=devsecops-cluster
Creating cluster "devsecops-cluster" ...
 • Ensuring node image (kindest/node:v1.33.1) 🖼
 ⠙ Ensuring node image (kindest/node:v1.33.1) 🖼^C
lokesh@Lokesh-K:~/devsecops-demo$ kind create cluster --name=devsecops-cluster
Creating cluster "devsecops-cluster" ...
 ✓ Ensuring node image (kindest/node:v1.33.1) 🖼
 ✓ Preparing nodes 📦
 ✓ Writing configuration 📜
 ✓ Starting control-plane 🕹️
 ✓ Installing CNI 🔌
 ✓ Installing StorageClass 💾
Set kubectl context to "kind-devsecops-cluster"
You can now use your cluster with:

kubectl cluster-info --context kind-devsecops-cluster

Thanks for using kind! 😊
lokesh@Lokesh-K:~/devsecops-demo$    curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kube
ctl"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   138  100   138    0     0    303      0 --:--:-- --:--:-- --:--:--   303
100 57.3M  100 57.3M    0     0  3071k      0  0:00:19  0:00:19 --:--:-- 3775k
lokesh@Lokesh-K:~/devsecops-demo$ sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
lokesh@Lokesh-K:~/devsecops-demo$ kubectl  config current-context
kind-devsecops-cluster
lokesh@Lokesh-K:~/devsecops-demo$ kubectl get nodes
NAME                                STATUS   ROLES           AGE     VERSION
devsecops-cluster-control-plane   Ready    control-plane   3m21s   v1.33.1
lokesh@Lokesh-K:~/devsecops-demo$
```


Command Prompt
lokesh@Lokesh-K: ~/devsecop
lokesh@Lokesh-K: ~/devseco
lokesh@Lokesh-K:~/devsecops-demo$ kubectl create namespace argocd
kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml
namespace/argocd created
customresourcedefinition.apiextensions.k8s.io/applications.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/applicationsets.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/appprojects.argoproj.io created
serviceaccount/argocd-application-controller created
serviceaccount/argocd-applicationset-controller created
serviceaccount/argocd-dex-server created
serviceaccount/argocd-notifications-controller created
serviceaccount/argocd-redis created
serviceaccount/argocd-repo-server created
serviceaccount/argocd-server created
role.rbac.authorization.k8s.io/argocd-application-controller created
role.rbac.authorization.k8s.io/argocd-applicationset-controller created
role.rbac.authorization.k8s.io/argocd-dex-server created
role.rbac.authorization.k8s.io/argocd-notifications-controller created
role.rbac.authorization.k8s.io/argocd-redis created
role.rbac.authorization.k8s.io/argocd-server created
clusterrole.rbac.authorization.k8s.io/argocd-application-controller created
clusterrole.rbac.authorization.k8s.io/argocd-applicationset-controller created
clusterrole.rbac.authorization.k8s.io/argocd-server created
rolebinding.rbac.authorization.k8s.io/argocd-application-controller created
rolebinding.rbac.authorization.k8s.io/argocd-applicationset-controller created
rolebinding.rbac.authorization.k8s.io/argocd-dex-server created
rolebinding.rbac.authorization.k8s.io/argocd-notifications-controller created
rolebinding.rbac.authorization.k8s.io/argocd-redis created
rolebinding.rbac.authorization.k8s.io/argocd-server created
clusterrolebinding.rbac.authorization.k8s.io/argocd-application-controller created
clusterrolebinding.rbac.authorization.k8s.io/argocd-applicationset-controller created
clusterrolebinding.rbac.authorization.k8s.io/argocd-server created
configmap/argocd-cm created
configmap/argocd-cmd-params-cm created
configmap/argocd-gpg-keys-cm created
configmap/argocd-notifications-cm created
configmap/argocd-rbac-cm created
configmap/argocd-ssh-known-hosts-cm created
configmap/argocd-tls-certs-cm created


Command Prompt
lokesh@Lokesh-K: ~/devsecop
lokesh@Lokesh-K: ~/devseco
networkpolicy.networking.k8s.io/argocd-server-network-policy created
lokesh@Lokesh-K:~/devsecops-demo$ kubectl get pods -n argocd
NAME READY STATUS RESTARTS AGE
argocd-application-controller-0 0/1 ContainerCreating 0 65s
argocd-applicationset-controller-655cc58ff8-786ln 1/1 Running 0 67s
argocd-dex-server-7d9dfb4fb8-bt2qf 0/1 PodInitializing 0 66s
argocd-notifications-controller-6c6848bc4c-7mc85 0/1 ContainerCreating 0 66s
argocd-redis-656c79549c-z476l 0/1 PodInitializing 0 66s
argocd-repo-server-856b768fd9-lfnp5 0/1 Init:0/1 0 66s
argocd-server-99c485944-zf6vn 0/1 ContainerCreating 0 65s
lokesh@Lokesh-K:~/devsecops-demo$ kubectl get pods -n argocd -w
NAME READY STATUS RESTARTS AGE
argocd-application-controller-0 0/1 ContainerCreating 0 73s
argocd-applicationset-controller-655cc58ff8-786ln 1/1 Running 0 75s
argocd-dex-server-7d9dfb4fb8-bt2qf 0/1 PodInitializing 0 74s
argocd-notifications-controller-6c6848bc4c-7mc85 1/1 Running 0 74s
argocd-redis-656c79549c-z476l 0/1 PodInitializing 0 74s
argocd-repo-server-856b768fd9-lfnp5 0/1 Init:0/1 0 74s
argocd-server-99c485944-zf6vn 0/1 Running 0 73s
argocd-repo-server-856b768fd9-lfnp5 0/1 Init:0/1 0 74s
argocd-repo-server-856b768fd9-lfnp5 0/1 PodInitializing 0 75s
argocd-application-controller-0 0/1 Running 0 74s
argocd-application-controller-0 1/1 Running 0 86s
argocd-dex-server-7d9dfb4fb8-bt2qf 1/1 Running 0 92s
argocd-server-99c485944-zf6vn 1/1 Running 0 100s
argocd-redis-656c79549c-z476l 1/1 Running 0 103s
argocd-repo-server-856b768fd9-lfnp5 0/1 Running 0 104s
argocd-repo-server-856b768fd9-lfnp5 1/1 Running 0 114s
^Clokesh@Lokesh-K:~/devsecops-demo$ kubectl get pods -n argocd
NAME READY STATUS RESTARTS AGE
argocd-application-controller-0 1/1 Running 0 9m40s
argocd-applicationset-controller-655cc58ff8-786ln 1/1 Running 0 9m42s
argocd-dex-server-7d9dfb4fb8-bt2qf 1/1 Running 0 9m41s
argocd-notifications-controller-6c6848bc4c-7mc85 1/1 Running 0 9m41s
argocd-redis-656c79549c-z476l 1/1 Running 0 9m41s
argocd-repo-server-856b768fd9-lfnp5 1/1 Running 0 9m41s
argocd-server-99c485944-zf6vn 1/1 Running 0 9m40s
lokesh@Lokesh-K:~/devsecops-demo$

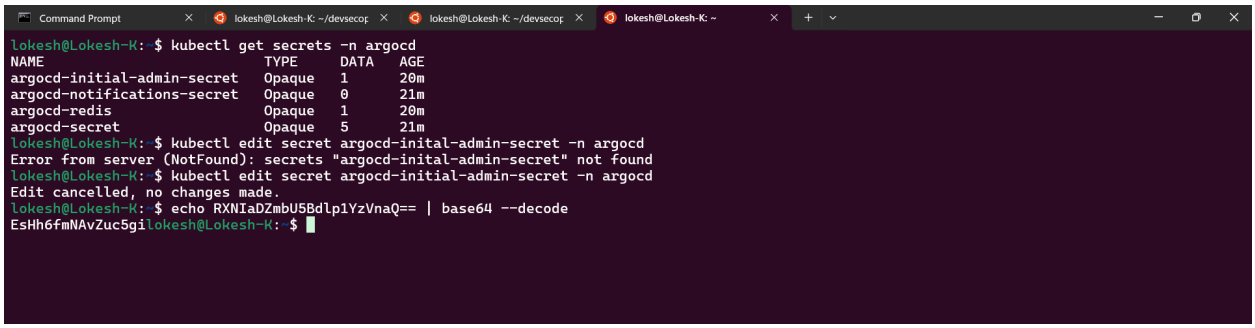


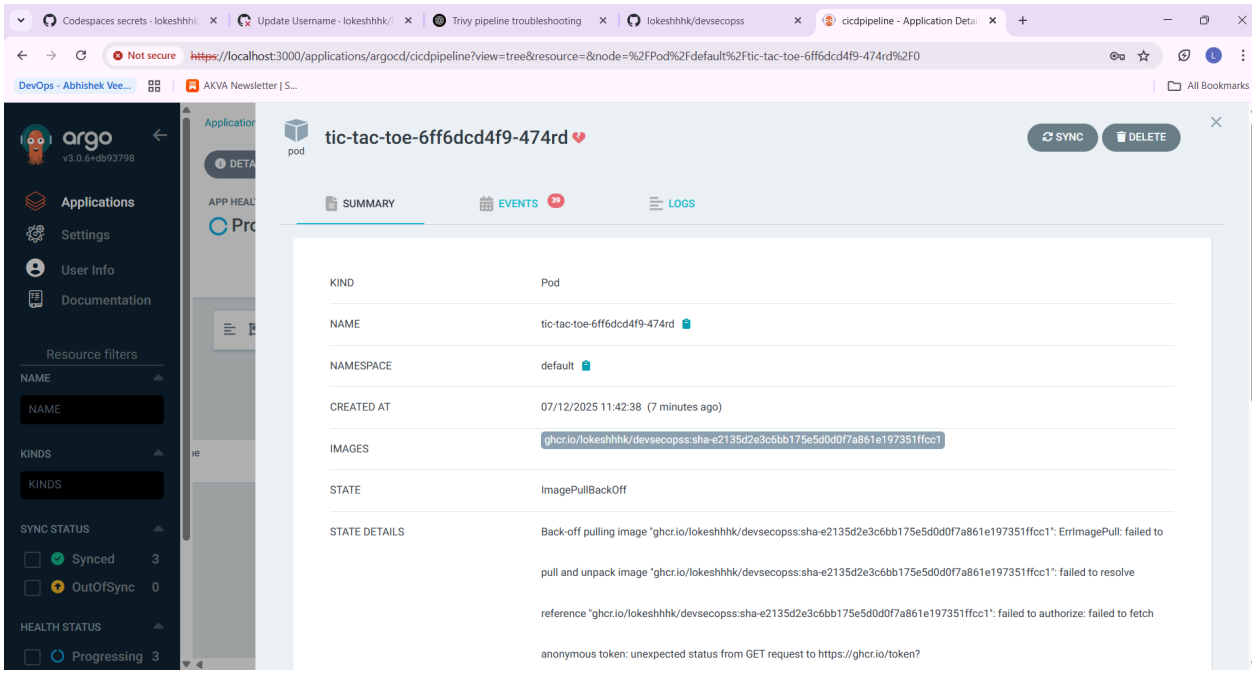
Here it will ask password for the argocd so we need to use like

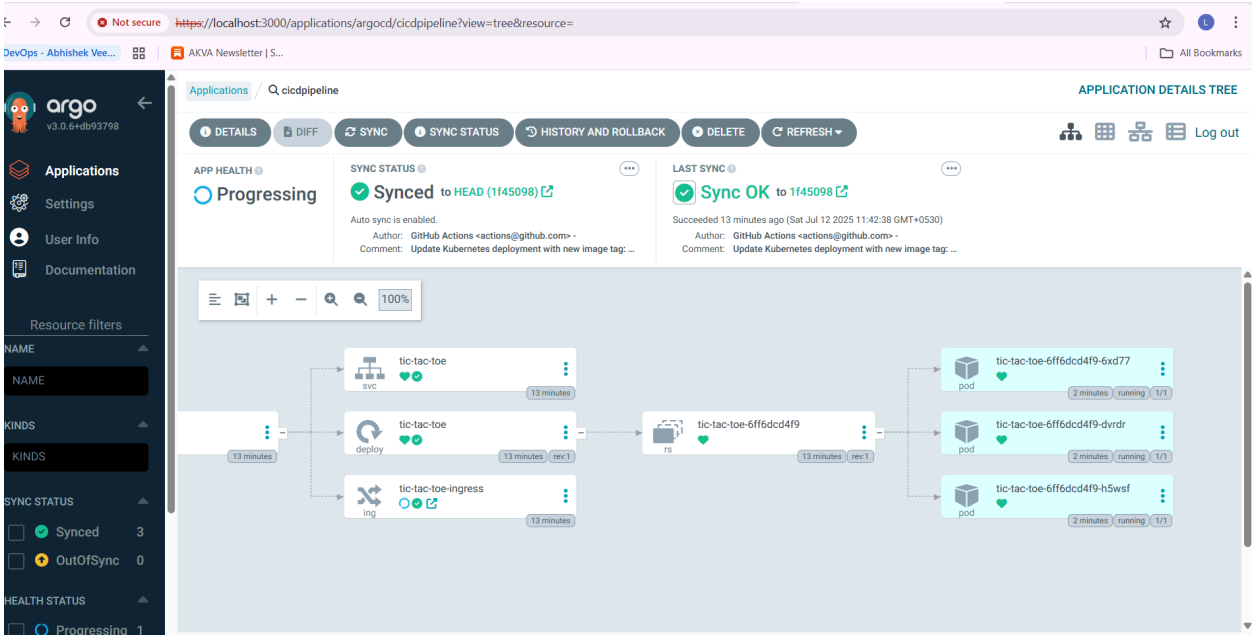
Kubectl get secrets -n argocd

In that we need to view initial admin secret

Kubectl edit secret argocd-initial -admin secret -n argocd

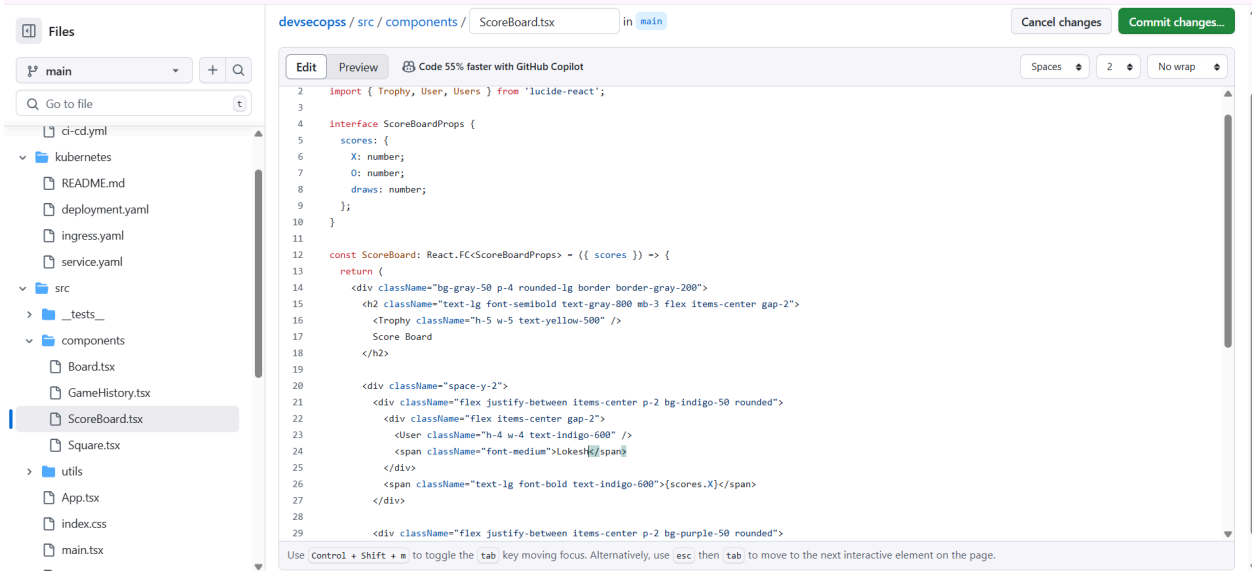



Not secure https://localhost:3000/applications/argocd/cicdpipeline?view=tree&resource=&node=%2FPod%2Fdefault%2Ftic-tac-toe-6ff6dcd4f9-474rd%2F0
argo
v3.0.6+db93798
Applications
Settings
User Info
Documentation
Resource filters
NAME
KINDS
SYNC STATUS
Synced 3
OutOfSync 0
HEALTH STATUS
Progressing 3
pod
tic-tac-toe-6ff6dcd4f9-474rd
SYNC
DELETE
SUMMARY
EVENTS 39
LOGS
KIND Pod
NAME tic-tac-toe-6ff6dcd4f9-474rd
NAMESPACE default
CREATED AT 07/12/2025 11:42:38 (7 minutes ago)
IMAGES ghcr.io/lokeshhhk/devsecopss:sha-e2135d2e3c6bb175e5d0d0f7a861e197351ffcc1
STATE ImagePullBackOff
STATE DETAILS Back-off pulling image "ghcr.io/lokeshhhk/devsecopss:sha-e2135d2e3c6bb175e5d0d0f7a861e197351ffcc1": ErrImagePull: failed to pull and unpack image "ghcr.io/lokeshhhk/devsecopss:sha-e2135d2e3c6bb175e5d0d0f7a861e197351ffcc1": failed to resolve reference "ghcr.io/lokeshhhk/devsecopss:sha-e2135d2e3c6bb175e5d0d0f7a861e197351ffcc1": failed to authorize: failed to fetch anonymous token: unexpected status from GET request to https://ghcr.io/token?

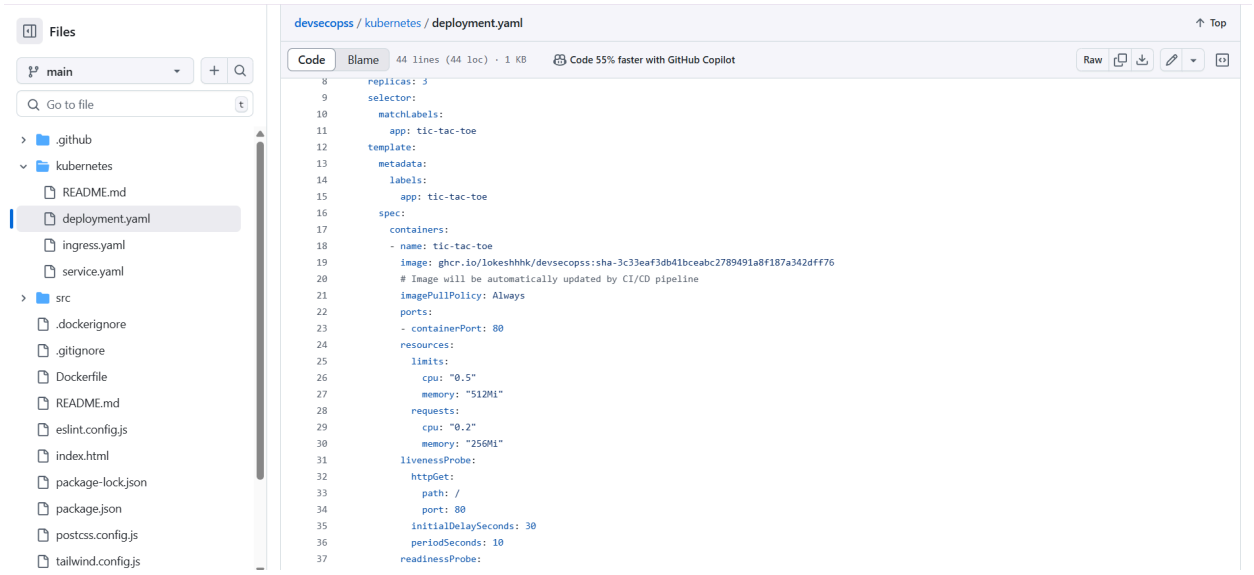

Not secure https://localhost:3000/applications/argocd/cicdpipeline?view=tree&resource=
Applications / cicdpipeline
APPLICATION DETAILS TREE
DETAILS DIFF SYNC SYNC STATUS HISTORY AND ROLLBACK DELETE REFRESH
Log out
APP HEALTH
Progressing
SYNC STATUS
Synced to HEAD (1f45098)
Auto sync is enabled.
Author: GitHub Actions <actions@github.com> -
Comment: Update Kubernetes deployment with new image tag: ...
LAST SYNC
Sync OK to 1f45098
Succeeded 13 minutes ago (Sat Jul 12 2025 11:42:38 GMT+0530)
Author: GitHub Actions <actions@github.com> -
Comment: Update Kubernetes deployment with new image tag: ...
100%
tic-tac-toe
svc
tic-tac-toe
deploy
tic-tac-toe-ingress
ing
tic-tac-toe-6ff6dcd4f9
rs
tic-tac-toe-6ff6dcd4f9-6xd77
tic-tac-toe-6ff6dcd4f9-dvrdr
tic-tac-toe-6ff6dcd4f9-h5wsf
pod

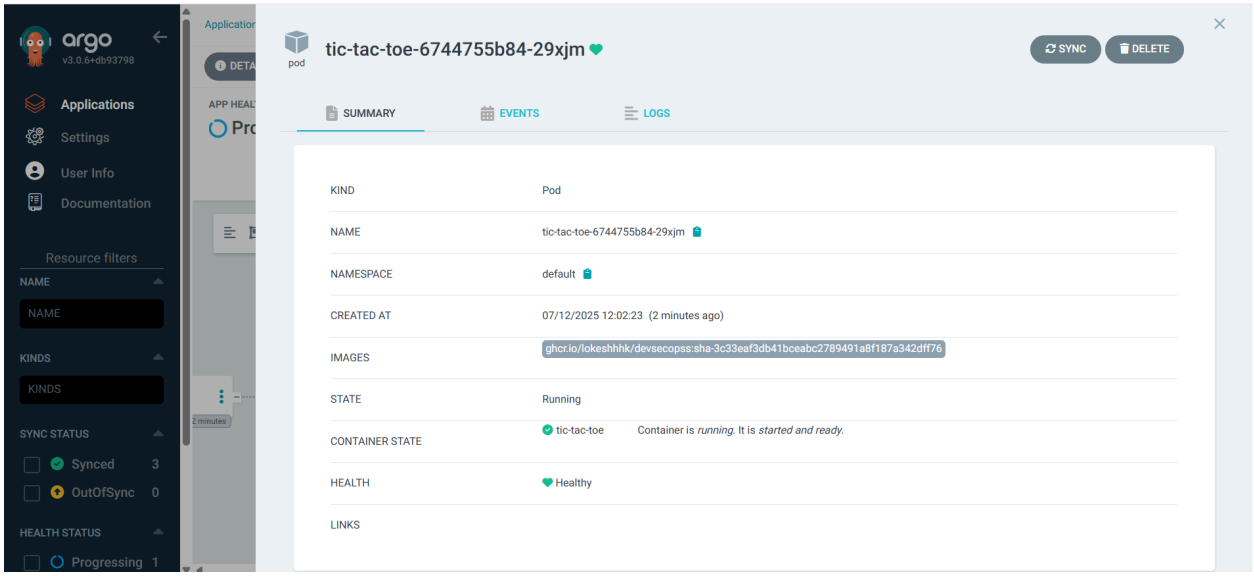
```yaml
template:
  metadata:
    labels:
      app: tic-tac-toe
  spec:
    containers:
    - name: tic-tac-toe
      image: ghcr.io/lokeshhhk/devsecopss:sha-e2135d2e3c6bb175e5d0d0f7a861e197351ffcc1
      # Image will be automatically updated by CI/CD pipeline
      imagePullPolicy: Always
      ports:
      - containerPort: 80
      resources:
        limits:
          cpu: "0.5"
          memory: "512Mi"
        requests:
          cpu: "0.2"
          memory: "256Mi"
      livenessProbe:
        httpGet:
          path: /
          port: 80
        initialDelaySeconds: 30
        periodSeconds: 10
      readinessProbe:
        httpGet:
          path: /
          port: 80
```

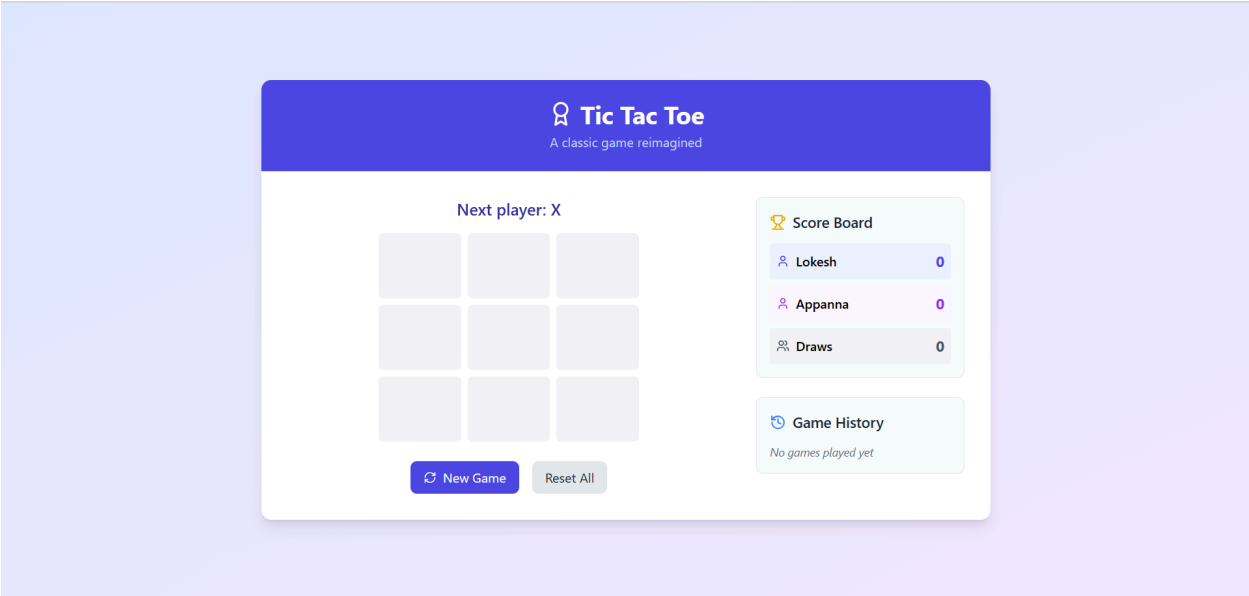
same image is update in the code also.



We make now change and lets see it is updated in application.


Files
main
Go to file
.github
kubernetes
README.md
deployment.yaml
ingress.yaml
service.yaml
src
.dockerignore
.gitignore
Dockerfile
README.md
eslint.config.js
index.html
package-lock.json
package.json
postcss.config.js
tailwind.config.js
devsecopss / kubernetes / deployment.yaml
Top
Code
Blame
44 lines (44 loc) · 1 KB
Code 55% faster with GitHub Copilot
Raw
replicas: 3
selector:
matchLabels:
app: tic-tac-toe
template:
metadata:
labels:
app: tic-tac-toe
spec:
containers:
- name: tic-tac-toe
image: ghcr.io/lokeshhhk/devsecopss:sha-3c33eaf3db41bceabc2789491a8f187a342dff76
Image will be automatically updated by CI/CD pipeline
imagePullPolicy: Always
ports:
- containerPort: 80
resources:
limits:
cpu: "0.5"
memory: "512Mi"
requests:
cpu: "0.2"
memory: "256Mi"
livenessProbe:
httpGet:
path: /
port: 80
initialDelaySeconds: 30
periodSeconds: 10
readinessProbe:


argo
v3.0.6+db93798
Applications
Settings
User Info
Documentation
Resource filters
NAME
KINDS
SYNC STATUS
Synced 3
OutOfSync 0
HEALTH STATUS
Progressing 1
pod
tic-tac-toe-6744755b84-29xjm
SYNC
DELETE
SUMMARY
EVENTS
LOGS
KIND Pod
NAME tic-tac-toe-6744755b84-29xjm
NAMESPACE default
CREATED AT 07/12/2025 12:02:23 (2 minutes ago)
IMAGES ghcr.io/lokeshhhk/devsecopss:sha-3c33eaf3db41bceabc2789491a8f187a342dff76
STATE Running
CONTAINER STATE tic-tac-toe Container is running. It is started and ready.
HEALTH Healthy
LINKS



Now the changes reflected in the application and the container image are also updated according to each commit.