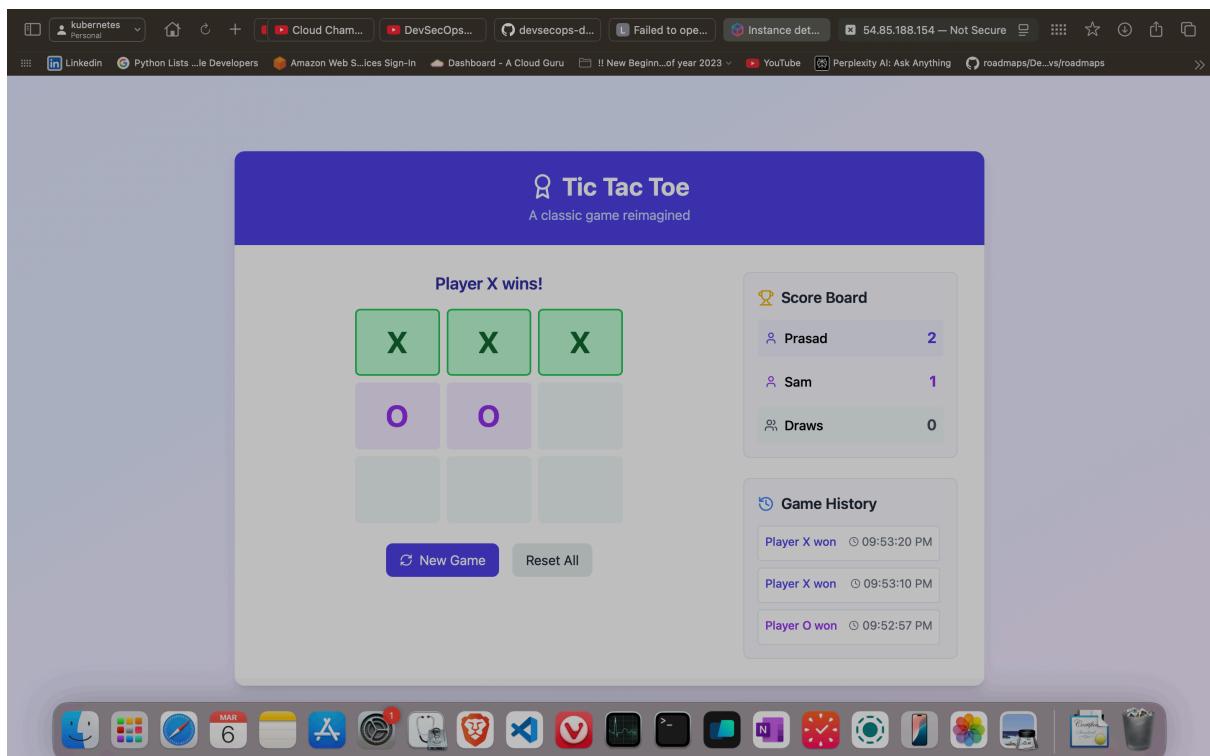
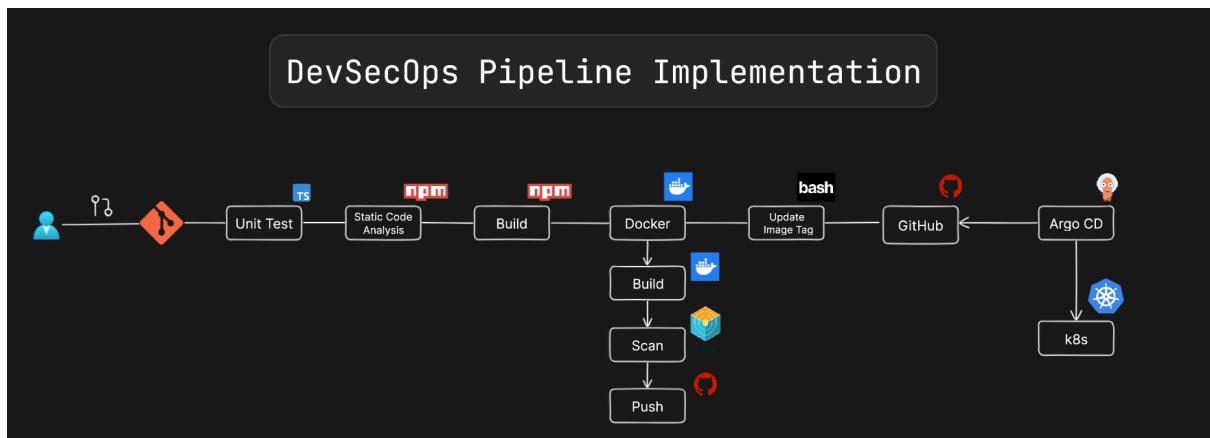




DevSecOps Demo Application Deployment Guide

Introduction



This guide will walk you through the process of setting up and deploying the DevSecOps demo application using GitHub, Docker, Kubernetes, and ArgoCD. Follow the steps below to create a seamless CI/CD pipeline that automatically deploys your application. Github link: <https://github.com/iam-veeramalla/devsecops-demo.git>



Prerequisites

- **Node.js:** Ensure you have Node.js (v14 or higher) installed.
- **npm or yarn:** Package manager for Node.js.
- **GitHub Account:** To clone the repository and manage secrets.
- **AWS Account:** To launch an EC2 instance.
- **Docker:** Containerization tool.
- **kubectl and kind:** For managing Kubernetes clusters.
- **ArgoCD:** For GitOps-based continuous delivery.

Getting Started

Step 1: Clone the Repository

Clone the DevSecOps demo application repository from GitHub:

```
git clone https://github.com/sumanprasad007/devsecops-demo.git
```

```
cd devsecops-demo
```

The screenshot shows the VS Code interface with the terminal tab active. The terminal output shows the command to clone the repository and the resulting directory structure:

```
git git clone https://github.com/sumanprasad007/devsecops-demo.git
Cloning into 'devsecops-demo'...
remote: Enumerating objects: 123, done.
remote: Counting objects: 100% (61/61), done.
remote: Compressing objects: 100% (20/20), done.
remote: Total 123 (delta 49), reused 41 (delta 41), pack-reused 62 (from 1)
Receiving objects: 100% (123/123), 68.11 KiB | 882.00 KiB/s, done.
Resolving deltas: 100% (56/56), done.

git
git ll
total 0
drwxr-xr-x@ 20 prasad staff 640B Mar 7 22:55 devsecops-demo
git cd devsecops-demo
devsecops-demo git:(main) ll
total 416
-rw-r--r--@ 1 prasad staff 333B Mar 7 22:55 Dockerfile
-rw-r--r--@ 1 prasad staff 2.0K Mar 7 22:55 README.md
-rw-r--r--@ 1 prasad staff 739B Mar 7 22:55 eslint.config.js
-rw-r--r--@ 1 prasad staff 366B Mar 7 22:55 index.html
drwxr-xr-x@ 6 prasad staff 192B Mar 7 22:55 kubernetes
-rw-r--r--@ 1 prasad staff 162K Mar 7 22:55 package-lock.json
-rw-r--r--@ 1 prasad staff 859B Mar 7 22:55 package.json
-rw-r--r--@ 1 prasad staff 81B Mar 7 22:55 postcss.config.js
drwxr-xr-x@ 9 prasad staff 288B Mar 7 22:55 src
-rw-r--r--@ 1 prasad staff 170B Mar 7 22:55 tailwind.config.js
-rw-r--r--@ 1 prasad staff 552B Mar 7 22:55 tsconfig.app.json
-rw-r--r--@ 1 prasad staff 119B Mar 7 22:55 tsconfig.json
-rw-r--r--@ 1 prasad staff 479B Mar 7 22:55 tsconfig.node.json
-rw-r--r--@ 1 prasad staff 220B Mar 7 22:55 vite.config.ts
```

Step 2: Install Dependencies

Install the necessary dependencies using npm or yarn:

```
npm install
```



Step 3: Start the Development Server

Start the development server to test the application locally:

```
npm run dev
```

Open your browser and navigate to <http://localhost:5173> to view the application.

Step 4: Build for Production

Create a production build of the application:

```
npm run build
```

The build artifacts will be stored in the dist/ directory.

GitHub Configuration

Step 5: Create a GitHub Access Token

1. Generate a GitHub personal access token with the necessary permissions.
2. Add the token to the repository settings under Secrets and variables with the name TOKEN.

```
git > devsecops-demo > src > components > ScoreBoard.tsx > ScoreBoard
3
4 interface ScoreBoardProps {
5   scores: {
6     X: number;
7     O: number;
8     draws: number;
9   };
10 }
11
12 const ScoreBoard: React.FC<ScoreBoardProps> = ({ scores }) => {
13   return (
14     <div className="bg-gray-50 p-4 rounded-lg border border-gray-200">
15       <h2 className="text-lg font-semibold text-gray-800 mb-3 flex items-center gap-2">
16         <Trophy className="h-5 w-5 text-yellow-500" />
17         Score Board
18       </h2>
19     </div>
20     <div className="space-y-2">
21       <div className="flex justify-between items-center p-2 bg-indigo-50 rounded">
22         <div className="flex items-center gap-2">
23           <User className="h-4 w-4 text-indigo-600" />
24           <span className="font-medium">Prasad Suman Mohan</span>
25         </div>
26         <span className="text-lg font-bold text-indigo-600">{scores.X}</span>
27       </div>
28     </div>
29   </div>
30 
```

<https://www.linkedin.com/in/prasad-suman-mohan>



Step 6: Trigger GitHub Actions Workflow

1. Create a new version of the application and push the code to the repository.
2. This will trigger the GitHub Actions workflow, creating a new Docker image stored in the GitHub Container Registry (GHCR).
3. The kubernetes/deployment.yaml file will be updated with the new image tag.

Kubernetes and ArgoCD Setup

Step 7: Launch an EC2 Instance

Launch an EC2 instance with the instance type t3a.medium or larger based on your requirements.

Step 8: Install Docker

Install Docker and log in to GHCR:

```
sudo apt install docker.io -y  
  
sudo usermod -aG docker ubuntu  
  
docker login ghcr.io
```

```
.025/aws creds | .025/aws creds | .025/aws creds | -zsh... +  
root@ip-172-31-43-14:/home/ubuntu# apt install docker.io -y  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following additional packages will be installed:  
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan  
Suggested packages:  
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-buildx docker-compose-v2 docker-doc rinse  
  zfs-fuse | zfsutils  
The following NEW packages will be installed:  
  bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc ubuntu-fan  
0 upgraded, 8 newly installed, 0 to remove and 134 not upgraded.  
Need to get 78.6 MB of archives.  
After this operation, 302 MB of additional disk space will be used.  
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 pigz amd64 2.8-1 [65.6 kB]  
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 bridge-utils amd64 1.7.1-1ubuntu2 [33.9 kB]  
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 runc amd64 1.1.12-0ubuntu3.1 [8599 kB]  
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 containerd amd64 1.7.24-0ubuntu1~24.04.1 [37.0 MB]  
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 dns-root-data all 2024071801~ubuntu0.24.04.1 [5918 B]  
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 dnsmasq-base amd64 2.90-2build2 [375 kB]  
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 docker.io amd64 26.1.3-0ubuntu1~24.04.1 [32.4 MB]  
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 ubuntu-fan all 0.12.16 [35.2 kB]  
Fetched 78.6 MB in 1s (62.7 MB/s)  
Preconfiguring packages ...  
Selecting previously unselected package pigz.  
(Reading database ... 70610 files and directories currently installed.)  
Preparing to unpack .../0-pigz_2.8-1_amd64.deb ...  
Unpacking pigz (2.8-1) ...  
Selecting previously unselected package bridge-utils.  
Preparing to unpack .../1-bridge-utils_1.7.1-1ubuntu2_amd64.deb ...  
Unpacking bridge-utils (1.7.1-1ubuntu2) ...  
Selecting previously unselected package runc.  
Preparing to unpack .../2-runc_1.1.12-0ubuntu3.1_amd64.deb ...
```



Step 9: Validate Docker Image

Run the latest Docker image to validate the changes:

```
docker run -d -p 80:80 ghcr.io/sumanprasad007/devsecops-demo:sha-e01062c9913b710fb823b7efe54ab5df110c4ae7
```

```
docker ps
```

```
.025/aws creds
root@ip-172-31-43-14:/home/ubuntu#
root@ip-172-31-43-14:/home/ubuntu# docker login ghcr.io
Username: sumanprasad007
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
root@ip-172-31-43-14:/home/ubuntu#
root@ip-172-31-43-14:/home/ubuntu#
root@ip-172-31-43-14:/home/ubuntu#
root@ip-172-31-43-14:/home/ubuntu#
root@ip-172-31-43-14:/home/ubuntu#
root@ip-172-31-43-14:/home/ubuntu# docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
root@ip-172-31-43-14:/home/ubuntu#
root@ip-172-31-43-14:/home/ubuntu#
root@ip-172-31-43-14:/home/ubuntu# docker run -d port 80:80 ghcr.io/sumanprasad007/devsecops-demo:sha-e01062c9913b710fb823b7efe54ab5df110c4ae7
Unable to find image 'port:latest' locally
docker: Error response from daemon: pull access denied for port, repository does not exist or may require 'docker login': denied: requested access to the resource is denied.
See 'docker run --help'.
root@ip-172-31-43-14:/home/ubuntu# docker run -d -p 80:80 ghcr.io/sumanprasad007/devsecops-demo:sha-e01062c9913b710fb823b7efe54ab5df110c4ae7
Unable to find image 'ghcr.io/sumanprasad007/devsecops-demo:sha-e01062c9913b710fb823b7efe54ab5df110c4ae7' locally
sha-e01062c9913b710fb823b7efe54ab5df110c4ae7: Pulling from sumanprasad007/devsecops-demo
f18232174bc9: Pull complete
ccc35e35d420: Pull complete
43f2ec460bdf: Pull complete
984583bcf083: Pull complete
8d27c072a58f: Pull complete
ab3286a73463: Pull complete
6d79cc6084d4: Pull complete
0c7e4c092ab7: Pull complete
```

Step 10: Install kind and kubectl

Install kind and kubectl for managing Kubernetes clusters:

```
# For AMD64 / x86_64
[ $(uname -m) = x86_64 ] && curl -Lo ./kind https://kind.sigs.k8s.io/dl/v0.27.0/kind-linux-amd64

# For ARM64
[ $(uname -m) = aarch64 ] && curl -Lo ./kind https://kind.sigs.k8s.io/dl/v0.27.0/kind-linux-arm64

chmod +x ./kind

sudo mv ./kind /usr/local/bin/kind
```



```
kind --version  
kind create cluster --name devsecops-k8s-cluster
```

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"  
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl  
kubectl config current-context  
kubectl get pods -A
```

```
..025/aws creds ..025/aws creds ..025/aws creds -zsh... +  
root@ip-172-31-43-14:/home/ubuntu# # For AMD64 / x86_64  
[ $uname -m ] = x86_64 ] && curl -Lo ./kind https://kind.sigs.k8s.io/dl/v0.27.0/kind-linux-amd64  
# For ARM64  
[ $uname -m ] = aarch64 ] && curl -Lo ./kind https://kind.sigs.k8s.io/dl/v0.27.0/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  848      0  --:--:--  --:--:--  --:--:--  843  
  0    0    0    0    0    0    0      0  --:--:--  --:--:--  --:--:--    0  
100  9.9M  100  9.9M  0    0  24.6M    0  --:--:--  --:--:--  --:--:--  24.6M  
root@ip-172-31-43-14:/home/ubuntu#  
root@ip-172-31-43-14:/home/ubuntu#  
root@ip-172-31-43-14:/home/ubuntu# kind --version  
Kind version 0.27.0  
root@ip-172-31-43-14:/home/ubuntu#  
root@ip-172-31-43-14:/home/ubuntu#  
root@ip-172-31-43-14:/home/ubuntu# kind create cluster --name devsecops-k8s-cluster  
Creating cluster "devsecops-k8s-cluster" ...  
✓ Ensuring node image (kindest/node:v1.32.2) [✓]  
✓ Preparing nodes [✓]  
✓ Writing configuration [✓]  
✓ Starting control-plane [✓]  
✓ Installing CNI [✓]  
✓ Installing StorageClass [✓]  
Set kubectl context to "kind-devsecops-k8s-cluster"  
You can now use your cluster with:  
  
kubectl cluster-info --context kind-devsecops-k8s-cluster  
  
Thanks for using kind! 😊  
root@ip-172-31-43-14:/home/ubuntu#  
root@ip-172-31-43-14:/home/ubuntu#  
root@ip-172-31-43-14:/home/ubuntu# curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
```

Step 11: Install ArgoCD

Install ArgoCD in your Kubernetes cluster:

```
kubectl create namespace argocd  
kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml
```



```
kubectl get pods -n argocd
```

```
.025/aws creds .025/aws creds .025/aws creds -zsh...
root@ip-172-31-43-14:/home/ubuntu# 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
```

Step 12: Access ArgoCD

```
.025/aws creds .025/aws creds .025/aws creds -zsh...
root@ip-172-31-43-14:/home/ubuntu# kubectl get svc -n argocd
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP  PORT(S)          AGE
argocd-applicationset-controller   ClusterIP  10.96.158.141 <none>        7000/TCP,8080/TCP  3m19
argocd-dex-server     ClusterIP  10.96.56.216  <none>        5556/TCP,5557/TCP,5558/TCP  3m19
argocd-metrics       ClusterIP  10.96.93.35   <none>        8082/TCP          3m19
argocd-notifications-controller-metrics ClusterIP  10.96.109.171 <none>        9001/TCP          3m19
argocd-redis         ClusterIP  10.96.255.22  <none>        6379/TCP          3m18
argocd-repo-server    ClusterIP  10.96.207.8   <none>        8081/TCP,8084/TCP  3m18
argocd-server         ClusterIP  10.96.192.6   <none>        80/TCP,443/TCP   3m18
argocd-server-metrics ClusterIP  10.96.105.87 <none>        8083/TCP          3m18
root@ip-172-31-43-14:/home/ubuntu#
root@ip-172-31-43-14:/home/ubuntu#
root@ip-172-31-43-14:/home/ubuntu# kubectl port-forward svc argocd-server 8000:80 -n argocd
Error from server (NotFound): pods "svc" not found
root@ip-172-31-43-14:/home/ubuntu#
root@ip-172-31-43-14:/home/ubuntu# kubectl port-forward svc/argocd-server 8000:80 -n argocd
Forwarding from 127.0.0.1:8000 -> 8080
Forwarding from [::1]:8000 -> 8080
^Croot@ip-172-31-43-14:/home/ubuntu#
root@ip-172-31-43-14:/home/ubuntu#
root@ip-172-31-43-14:/home/ubuntu# kubectl port-forward svc/argocd-server 8000:80 -n argocd --address 0.0.0.0
Forwarding from 0.0.0.0:8000 -> 8080

Handling connection for 8000
Handling connection for 8000
Handling connection for 8000
Handling connection for 8000
```

Make ArgoCD accessible via port-forwarding:

<https://www.linkedin.com/in/prasad-suman-mohan>



```
kubectl port-forward svc/argocd-server 8000:80 -n argocd --address 0.0.0.0
```

```
kubectl get secrets -n argocd
```

```
kubectl get secrets argocd-initial-admin-secret -o yaml -n argocd
```

```
echo d0M3Rzc3S1pHYjlvZlRsMw== | base64 --decode
```

Step 13: Create ArgoCD Application

Create an application inside ArgoCD for your DevSecOps demo application.

The screenshot shows the ArgoCD UI with a modal dialog open for creating a new application. The application name is set to 'devsecops-demo-app'. Under 'SYNC POLICY', 'Automatic' is selected. Under 'SYNC OPTIONS', 'SKIP SCHEMA VALIDATION' is checked. Other options like 'PRUNE LAST' and 'RESPECT IGNORE DIFFERENCES' are also present. On the left sidebar, there are filters for 'Favorites Only', 'SYNC STATUS' (Unknown: 0, Synced: 1, OutOfSync: 0), 'HEALTH STATUS' (Progressing: 1, Suspended: 0, Healthy: 0, Degraded: 0, Missing: 0, Unknown: 0), and 'LABELS'.

Step 14: Create Docker Registry Secret

Create a secret to allow ArgoCD to fetch the Docker image from GHCR:

```
kubectl create secret docker-registry github-container-registry \
--docker-server=ghcr.io \
--docker-username=sumanprasad007 \
--docker-password=gSG33hqgjia;dsjg;asd gj;a ldsg aosd;f90u4t0lwLBI \
--docker-email=test@gmail.com
```



```
kubectl get secrets -A
```

```
root@ip-172-31-43-14:/home/ubuntu# kubectl get svc
NAME      TYPE      CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
kubernetes  ClusterIP  10.96.0.1    <none>        443/TCP   22m
tic-tac-toe  ClusterIP  10.96.49.87  <none>        80/TCP    7m19s
root@ip-172-31-43-14:/home/ubuntu# kubectl edit svc tic-tac-toe
service/tic-tac-toe edited
root@ip-172-31-43-14:/home/ubuntu#
root@ip-172-31-43-14:/home/ubuntu# kubectl get svc
NAME      TYPE      CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
kubernetes  ClusterIP  10.96.0.1    <none>        443/TCP   28m
tic-tac-toe  NodePort   10.96.49.87  <none>        80:30735/TCP 12m
root@ip-172-31-43-14:/home/ubuntu# kubectl get svc
NAME      TYPE      CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
kubernetes  ClusterIP  10.96.0.1    <none>        443/TCP   29m
tic-tac-toe  NodePort   10.96.49.87  <none>        80:30735/TCP 14m
root@ip-172-31-43-14:/home/ubuntu#
root@ip-172-31-43-14:/home/ubuntu# kubectl get pods
NAME            READY   STATUS    RESTARTS   AGE
tic-tac-toe-79ccbb98d-s72xx  1/1     Running   0          10m
tic-tac-toe-79ccbb98d-wkf9b  1/1     Running   0          10m
tic-tac-toe-79ccbb98d-z652m  1/1     Running   0          10m
root@ip-172-31-43-14:/home/ubuntu#
```

Step 15: Verify Pods

Ensure the pods are up and running:

```
kubectl get pods
```

```
~/Desktop/2025/aws creds -- root@ip-172-31-43-14:/home/ubuntu -- ssh -i 2025.pem ubuntu@54.85.188.154 ... ...sktop/2025/aws creds -- root@ip-172-31-43-14:/home/ubuntu -- ssh -i 2025.pem ubuntu@54.85.188.154 +
tic-tac-toe-bf47ccf4c-qg8dg  1/1     Running   0          7m54s
root@ip-172-31-43-14:/home/ubuntu#
root@ip-172-31-43-14:/home/ubuntu# kubectl get svc
NAME      TYPE      CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
kubernetes  ClusterIP  10.96.0.1    <none>        443/TCP   45m
tic-tac-toe  ClusterIP  10.96.49.87  <none>        80/TCP    29m
root@ip-172-31-43-14:/home/ubuntu#
root@ip-172-31-43-14:/home/ubuntu# kubectl port-forward 80:80 svc/tic-tac-toe --address 0.0.0.0
Error from server (NotFound): pods "80:80" not found
root@ip-172-31-43-14:/home/ubuntu#
root@ip-172-31-43-14:/home/ubuntu# kubectl port-forward 80:80 tic-tac-toe-bf47ccf4c-9v268 --address 0.0.0.0
Error from server (NotFound): pods "80:80" not found
root@ip-172-31-43-14:/home/ubuntu# kubectl port-forward 8888:80 tic-tac-toe-bf47ccf4c-9v268 --address 0.0.0.0
Error from server (NotFound): pods "8888:80" not found
root@ip-172-31-43-14:/home/ubuntu#
root@ip-172-31-43-14:/home/ubuntu# kubectl port-forward 8888:80 pods/tic-tac-toe-bf47ccf4c-9v268 --address 0.0.0.0
Error from server (NotFound): pods "8888:80" not found
root@ip-172-31-43-14:/home/ubuntu#
root@ip-172-31-43-14:/home/ubuntu# kubectl port-forward pods/tic-tac-toe-bf47ccf4c-9v268 80:80 --address 0.0.0.0
Unable to listen on port 80: Listeners failed to create with the following errors: [unable to create listener: Error
listen tcp4 0.0.0.0:80: bind: address already in use]
error: unable to listen on any of the requested ports: [{80 80}]
root@ip-172-31-43-14:/home/ubuntu# kubectl port-forward pods/tic-tac-toe-bf47ccf4c-9v268 8888:80 --address 0.0.0.0
Forwarding from 0.0.0.0:8888 -> 80

Handling connection for 8888
Handling connection for 8888
```

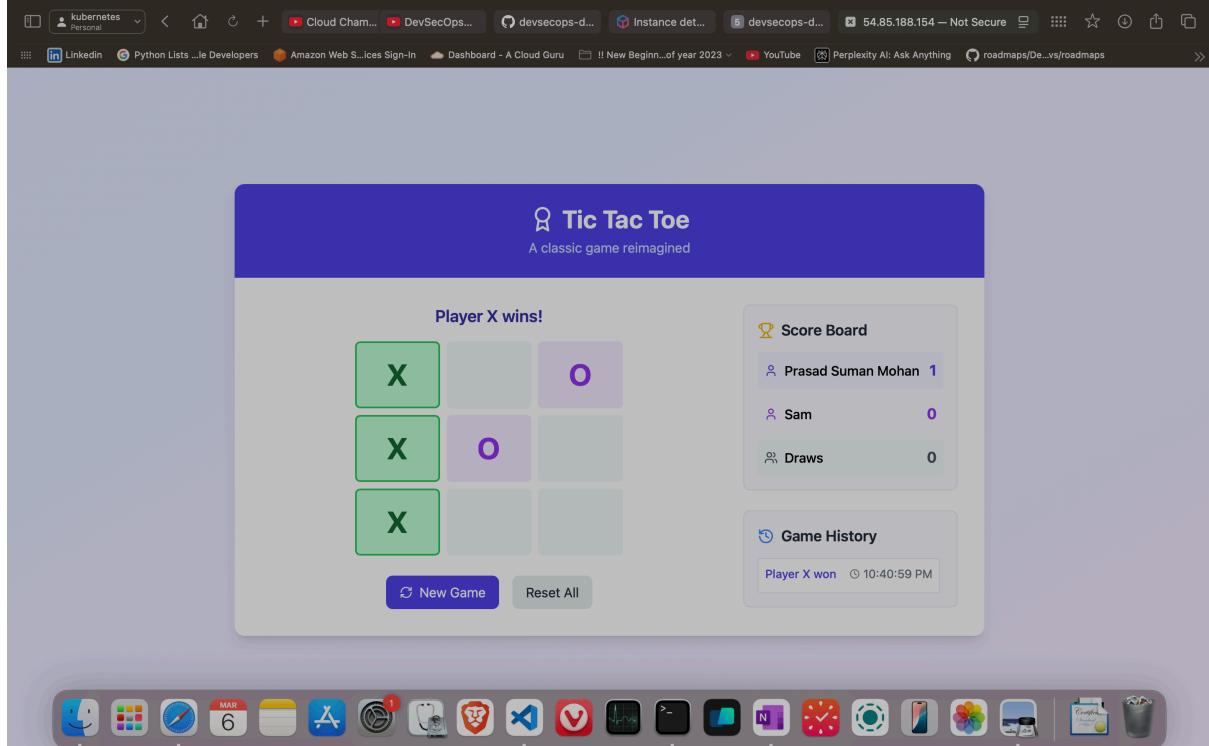


Step 16: Port Forward the Application

Port forward the application to access it in the browser:

```
kubectl port-forward pods/tic-tac-toe-bf47ccf4c-9v268 8888:80 --address 0.0.0.0
```

Access the application at <http://Public-ip:8888>.



Conclusion

By following these steps, you will have a fully automated CI/CD pipeline that deploys your DevSecOps demo application to a Kubernetes cluster using ArgoCD. The entire process, from code push to deployment, should take less than 180 seconds.



Understanding how to leverage AWS tools and features will enhance your capabilities, support certification preparation, and boost confidence in real-world problem-solving for DevOps, cloud engineering, and SRE roles. In the up-coming parts, we will discuss more such practical challenges along with steps for the different AWS based scenarios. So, stay tuned for the and follow @Prasad Suman Mohan for more such posts.

