## 📘 Project Title: Gym Pro Professional Static Web App Deployment Using DevOps Practices

### 📝 Overview

This project showcases a complete DevOps lifecycle by deploying a Flask-based Gym Web App on AWS using Docker containers, Amazon ECR, and Kubernetes on Amazon EKS. CI/CD is automated with GitHub Actions, and monitoring is planned using Prometheus and Grafana.

## 🔧 Tools & Technologies Used

| Tool/Service | Purpose |
| --- | --- |
|  |  |

| **Flask** | Web application framework |
| --- | --- |
| **Docker** | Containerization |
| **Amazon ECR** | Container registry |
| **Amazon EKS** | Kubernetes cluster hosting |
| **GitHub Actions** | CI/CD automation |
| **kubectl** | Kubernetes command-line tool |
| **AWS CLI** | AWS resource configuration |

## ⚙️ Project Workflow

### 1. **Flask App Setup**

* Basic Flask app defined in app.py
* Exposes port 5000

### 2. **Dockerize the App**

**Dockerfile**

FROM python:3.10  
  
WORKDIR /app  
  
COPY . /app  
  
RUN pip install flask  
  
EXPOSE 5000  
  
CMD ["python", "app.py"]

### 3. **ECR Setup and Push Docker Image**

aws ecr get-login-password --region ap-south-1 | docker login --username AWS --password-stdin 061039777231.dkr.ecr.ap-south-1.amazonaws.com  
  
docker build -t gym-app .  
docker tag gym-app:latest 061039777231.dkr.ecr.ap-south-1.amazonaws.com/gym-app:latest  
docker push 061039777231.dkr.ecr.ap-south-1.amazonaws.com/gym-app:latest

### 4. **Kubernetes Configuration**

#### deployment.yaml

apiVersion: apps/v1  
kind: Deployment  
metadata:  
 name: gym-app-deployment  
spec:  
 replicas: 2  
 selector:  
 matchLabels:  
 app: gym-app  
 template:  
 metadata:  
 labels:  
 app: gym-app  
 spec:  
 containers:  
 - name: gym-app-container  
 image: 061039777231.dkr.ecr.ap-south-1.amazonaws.com/gym-app:latest  
 ports:  
 - containerPort: 5000  
 livenessProbe:  
 httpGet:  
 path: /  
 port: 5000  
 initialDelaySeconds: 10  
 periodSeconds: 10  
 readinessProbe:  
 httpGet:  
 path: /  
 port: 5000  
 initialDelaySeconds: 5  
 periodSeconds: 5

#### service.yaml

apiVersion: v1  
kind: Service  
metadata:  
 name: gym-app-service  
spec:  
 type: LoadBalancer  
 selector:  
 app: gym-app  
 ports:  
 - protocol: TCP  
 port: 80  
 targetPort: 5000

### 5. **GitHub Actions Workflow**

``

name: Build and Deploy to EKS  
  
on:  
 push:  
 branches: [main]  
  
jobs:  
 deploy:  
 runs-on: ubuntu-latest  
  
 steps:  
 - name: Checkout Code  
 uses: actions/checkout@v3  
  
 - name: Configure AWS  
 uses: aws-actions/configure-aws-credentials@v2  
 with:  
 aws-access-key-id: ${{ secrets.AWS\_ACCESS\_KEY\_ID }}  
 aws-secret-access-key: ${{ secrets.AWS\_SECRET\_ACCESS\_KEY }}  
 aws-region: ap-south-1  
  
 - name: Login to Amazon ECR  
 run: |  
 aws ecr get-login-password --region ap-south-1 | docker login --username AWS --password-stdin 061039777231.dkr.ecr.ap-south-1.amazonaws.com  
  
 - name: Build Docker Image  
 run: |  
 docker build -t gym-app .  
 docker tag gym-app:latest 061039777231.dkr.ecr.ap-south-1.amazonaws.com/gym-app:latest  
  
 - name: Push to ECR  
 run: |  
 docker push 061039777231.dkr.ecr.ap-south-1.amazonaws.com/gym-app:latest  
  
 - name: Update Kubeconfig  
 run: |  
 aws eks update-kubeconfig --region ap-south-1 --name gym-app-cluster  
  
 - name: Deploy to EKS  
 run: |  
 kubectl apply -f k8s-manifests/

### 6. **Deploy and Verify**

kubectl get pods  
kubectl get svc

A screenshot of a computer

AI-generated content may be incorrect.

## 📌 Final Notes

* Pods were crashing initially because Flask was running only on 127.0.0.1. Fixed it by updating app.py:

app.run(host="0.0.0.0", port=5000, debug=True)

* Used port 5000 internally and exposed via Service on port 80.