**My Web App - Kubernetes Deployment**

**Overview**

This project demonstrates how to containerize a simple web application using Docker and deploy it on Kubernetes with Amazon EKS. The application is served using an Nginx web server.

**Tech Stack**

* **Docker**: Containerizes the web application to ensure consistency across different environments.
* **AWS ECR**: Stores the Docker images for deployment.
* **Kubernetes (EKS)**: Manages and orchestrates the containerized application.
* **Nginx**: Serves the static HTML content.

**Project Structure**

Docker and Kube's/

│── my-web-app.html # HTML file for the web app

│── Dockerfile # Docker instructions to build the image

│── deployment.yaml # Kubernetes Deployment configuration

│── service.yaml # Kubernetes Service for exposing the app

│── README.md # Documentation (this file)

**Setup & Deployment**

**1. Build & Push Docker Image to AWS ECR**

aws ecr get-login-password --region eu-north-1 | docker login --username AWS --password-stdin <AWS\_ACCOUNT\_ID>.dkr.ecr.eu-north-1.amazonaws.com

docker build -t my-web-app .

docker tag my-web-app:latest <AWS\_ACCOUNT\_ID>.dkr.ecr.eu-north-1.amazonaws.com/my-web-app:latest

docker push <AWS\_ACCOUNT\_ID>.dkr.ecr.eu-north-1.amazonaws.com/my-web-app:latest

**2. Deploy to Kubernetes**

kubectl apply -f deployment.yaml

kubectl apply -f service.yaml

**3. Check Deployment & Access Application**

kubectl get pods

kubectl get services

Access the web app via the EXTERNAL-IP of the Kubernetes service.

**Troubleshooting**

* If you get a **403 Forbidden** error, ensure the file permissions are correct and Nginx is correctly configured.
* If the pods are not running, use kubectl logs <pod-name> to debug.

**Author**

Tafadzwa Shingirai Mwerenga