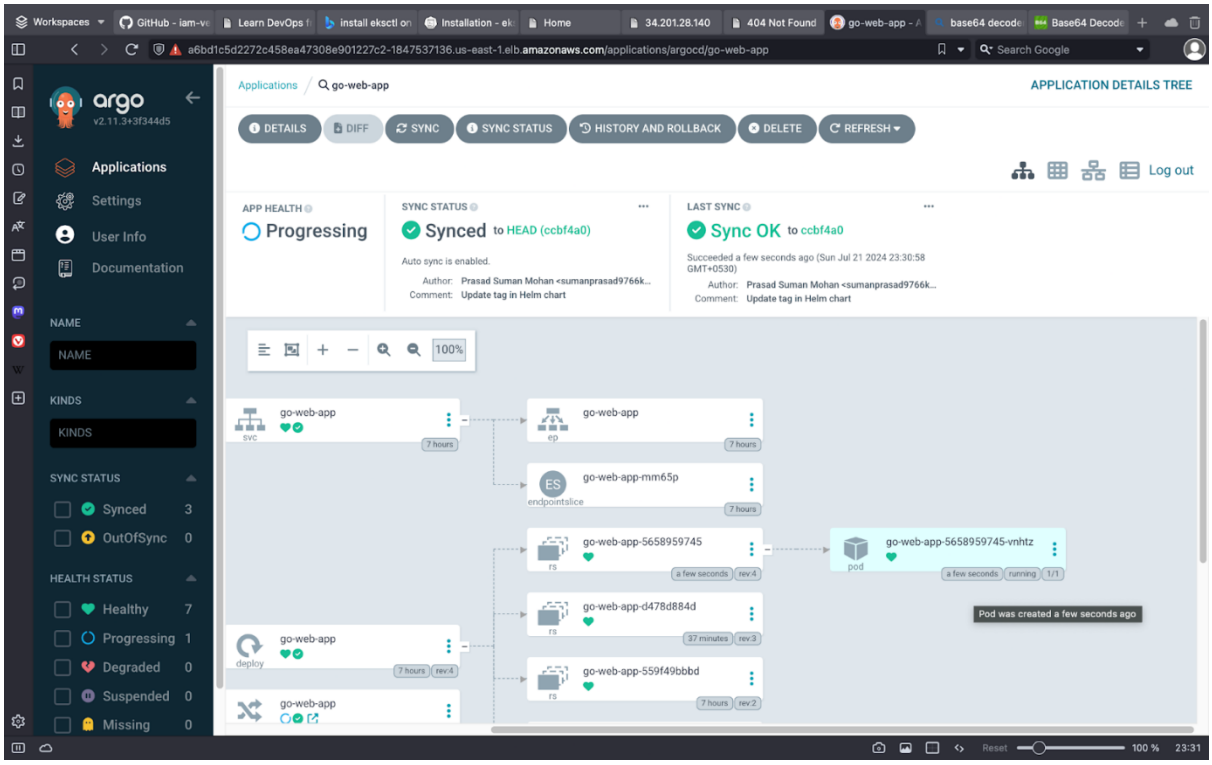
End to end GitOps Project using AWS EKS & ArgoCD

Introduction 🌐

End-to-End GitOps Project Using AWS EKS & ArgoCD

Introduction 🌐

This guide provides an end-to-end DevOps implementation for a Golang web application, covering containerization, Kubernetes deployment, CI/CD pipelines, and GitOps with ArgoCD on AWS EKS.



Project Flow Architecture

1. **Installation and Setup**
    - Install Golang, other dependencies, AWS CLI, EKS CTL, Kubectl, and Helm.
2. **Containerization**
    - Create a multi-stage Dockerfile for the Golang web application.
3. **Kubernetes Manifests**
    - Create deployment.yaml, service.yaml, and ingress.yaml files.

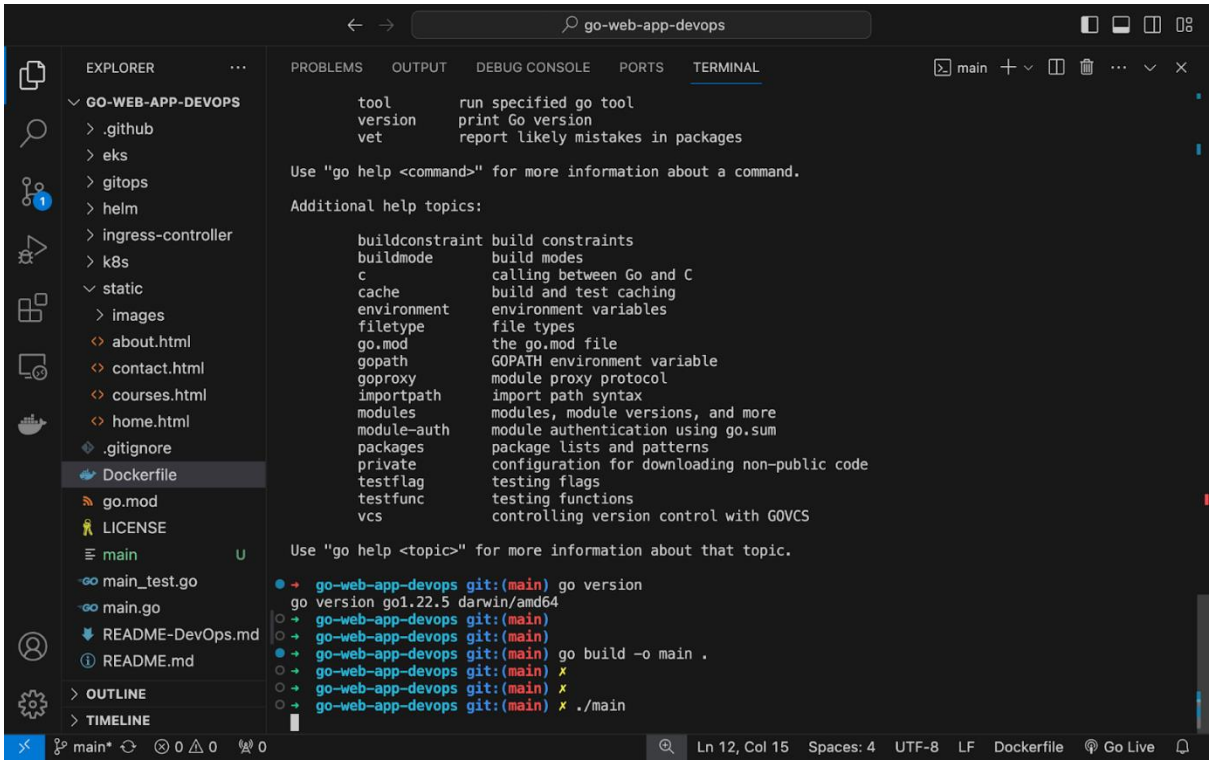
4. **EKS Cluster Creation**
    - Use EKS CTL to create an EKS cluster on AWS.
5. **Ingress Controller**
    - Set up Nginx Ingress Controller for domain mapping and load balancing.
6. **Helm Charts**
    - Create Helm charts for Kubernetes manifests to enable multi-environment support and facilitate GitOps.
7. **Continuous Integration**
    - Configure GitHub Actions for build, test, docker image creation, pushing to DockerHub, and updating values.yaml.
8. **Continuous Deployment**
    - Set up ArgoCD for automated deployment of the latest Docker images from DockerHub.

Running the Go Application Locally

1. **Build and Run**

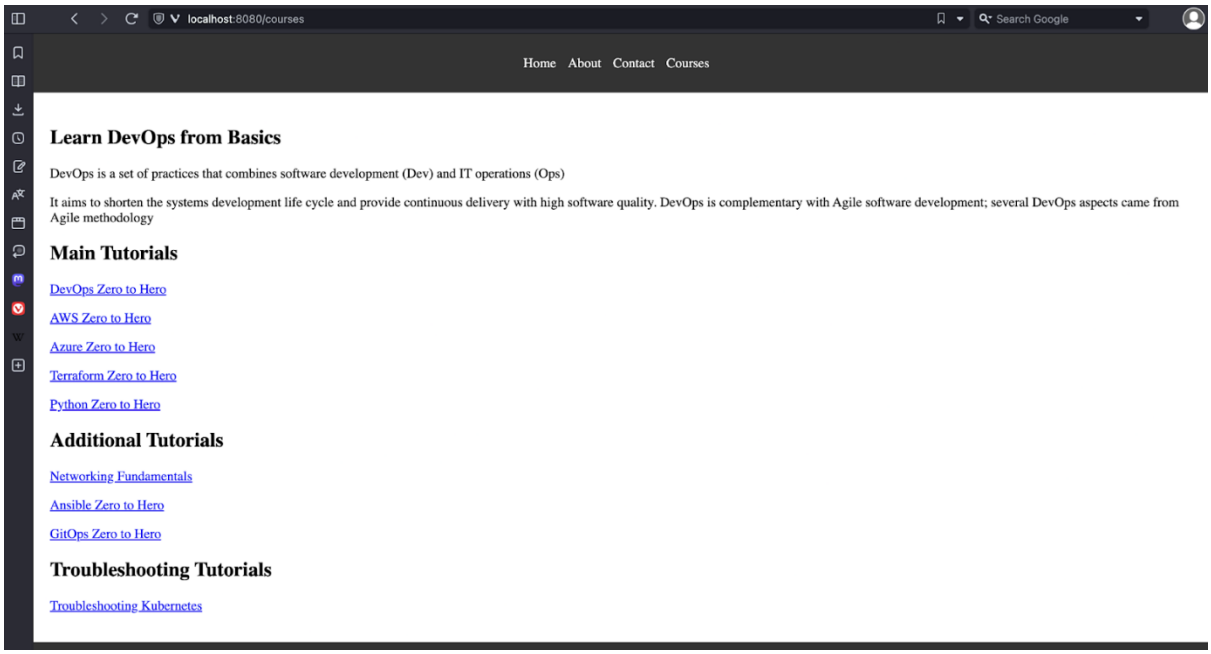
go build -o main .

./main



2. **Access the Application**

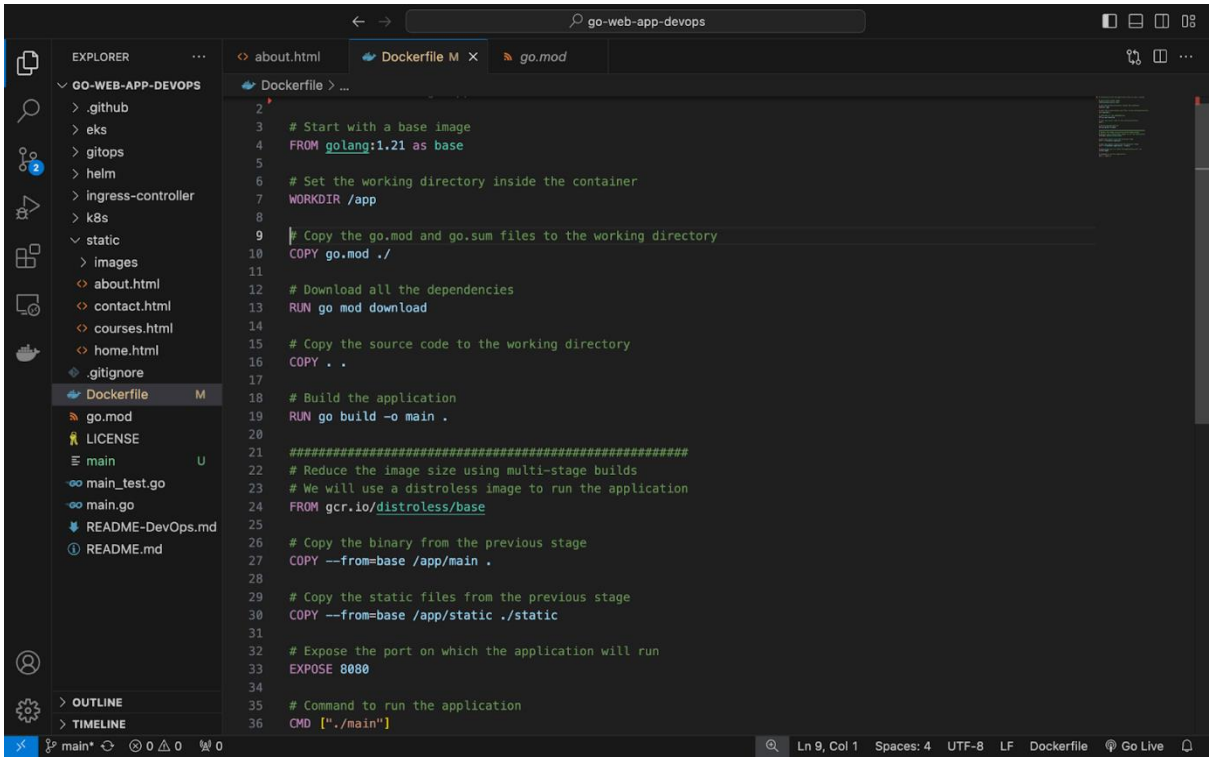
- Visit http://localhost:8080/courses.



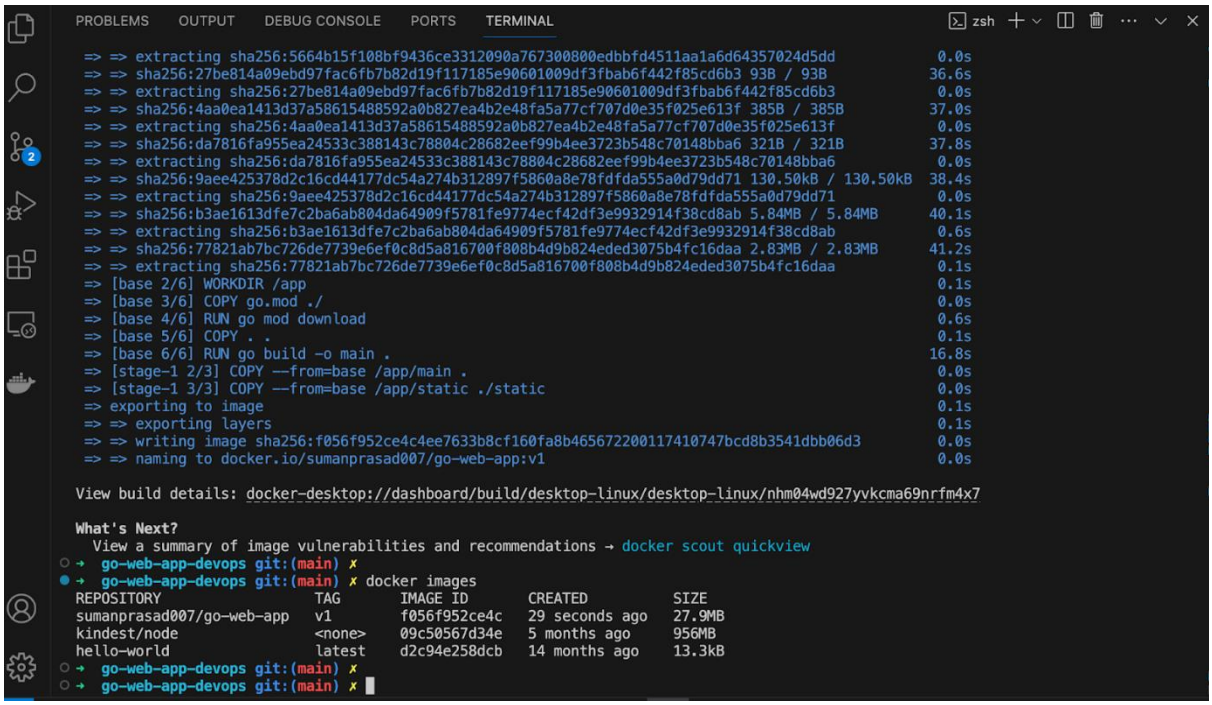
Writing a Multi-Stage Dockerfile

cd end-to-end-cicd-pipeline-for-go-web-app

vi Dockerfile



Build the Multi-Stage Dockerfile



docker build -t sumanprasad007/go-web-app:v1 .

docker images

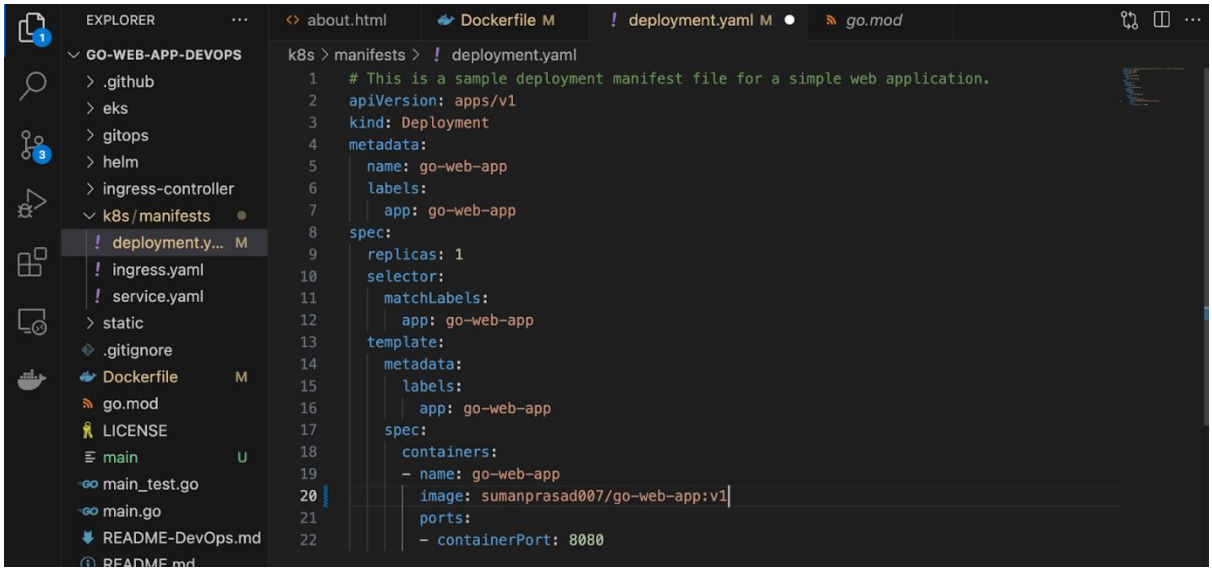
Test the Docker Image

docker run -it -p 8080:8080 sumanprasad007/go-web-app:v1

Push the Docker Image to DockerHub

docker push sumanprasad007/go-web-app:v1

Creating Kubernetes Manifests



EKS Cluster Creation

Prerequisites

- **Kubectl**: Install or update kubectl
- **EKS CTL**: Install or update eksctl

brew tap weaveworks/tap

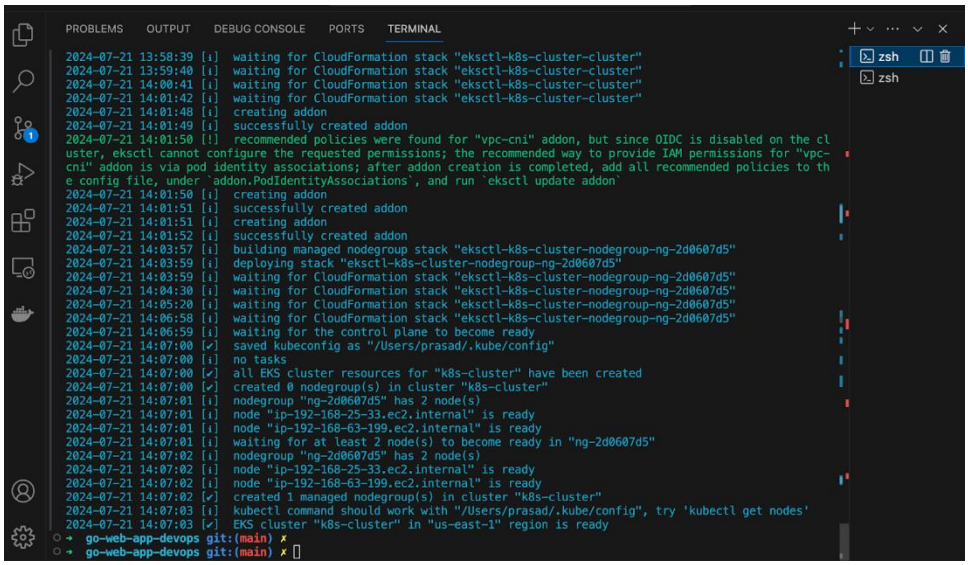
brew install weaveworks/tap/eksctl

- **AWS CLI**: Install, update, and configure AWS CLI

aws configure

Create EKS Cluster

eksctl create cluster --name demo-cluster --region us-east-1

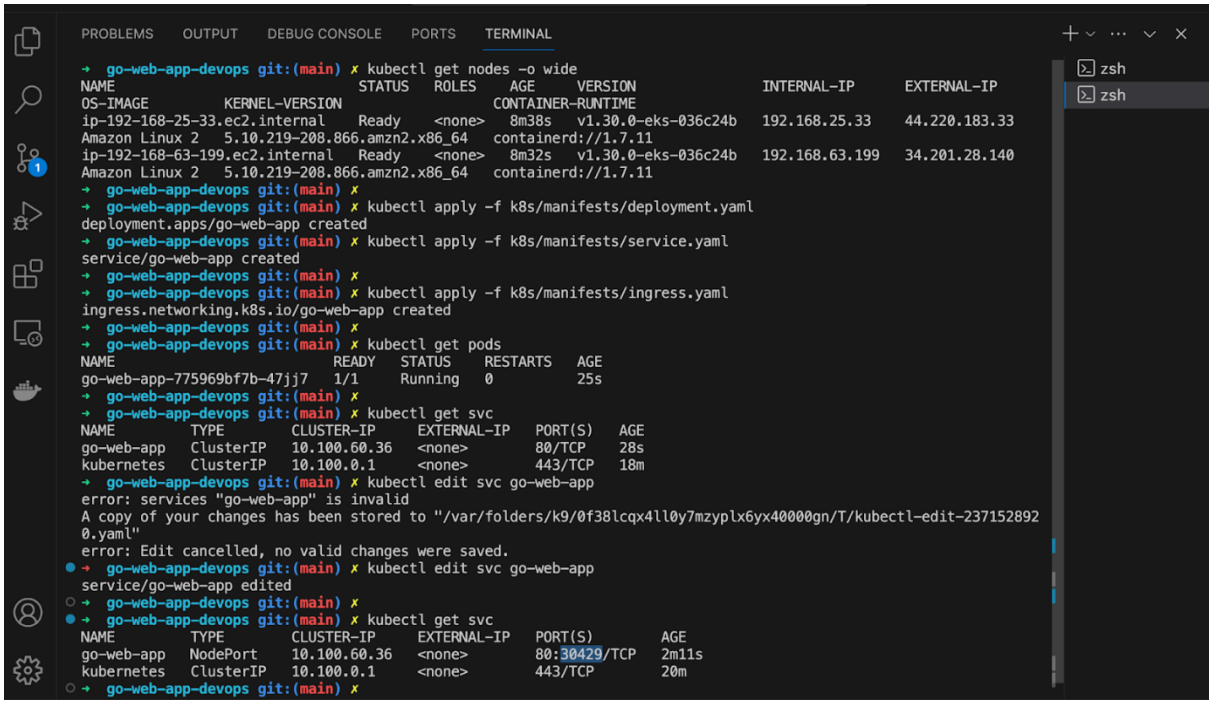


Apply Kubernetes Manifests

kubectl apply -f k8s/manifests/deployment.yaml

kubectl apply -f k8s/manifests/service.yaml

kubectl apply -f k8s/manifests/ingress.yaml



Check Service Reachability

kubectl get svc

kubectl edit svc go-web-app

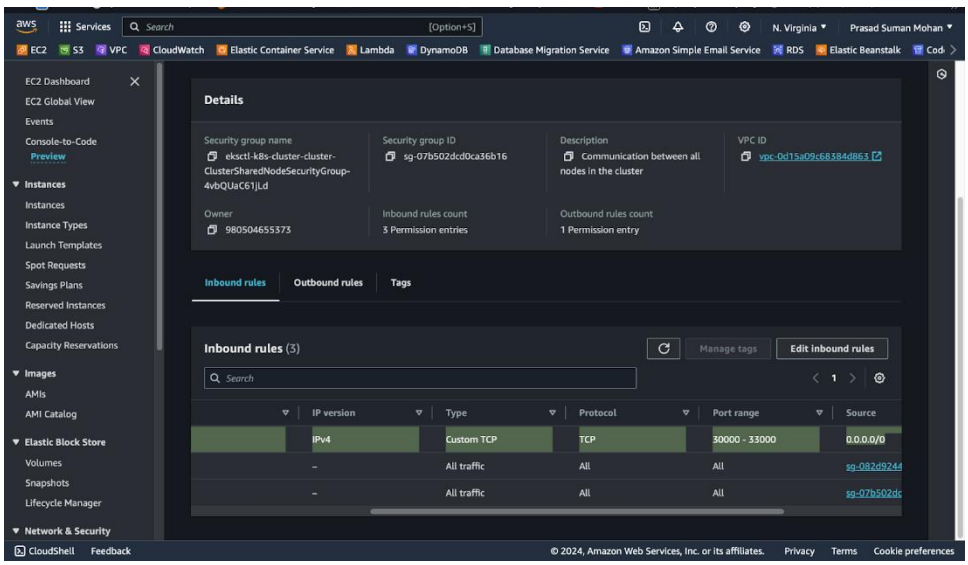
Change type from ClusterIP to NodePort

kubectl get nodes -o wide

Copy external IP and access http://<external-ip>:<nodePort>/courses

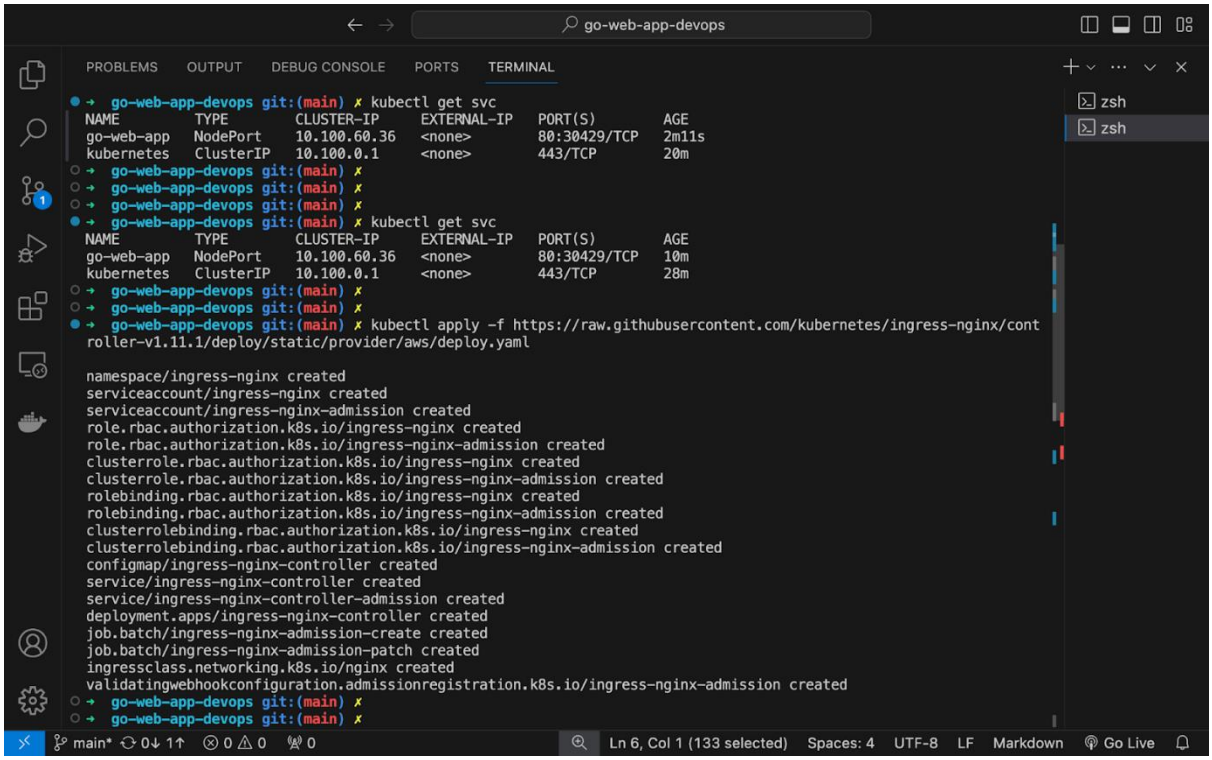
Enable Ports in EKS Cluster Security Group

- Ensure necessary ports are open in the security group.



Nginx Ingress Controller Setup

kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.11.1/deploy/static/provider/aws/deploy.yaml



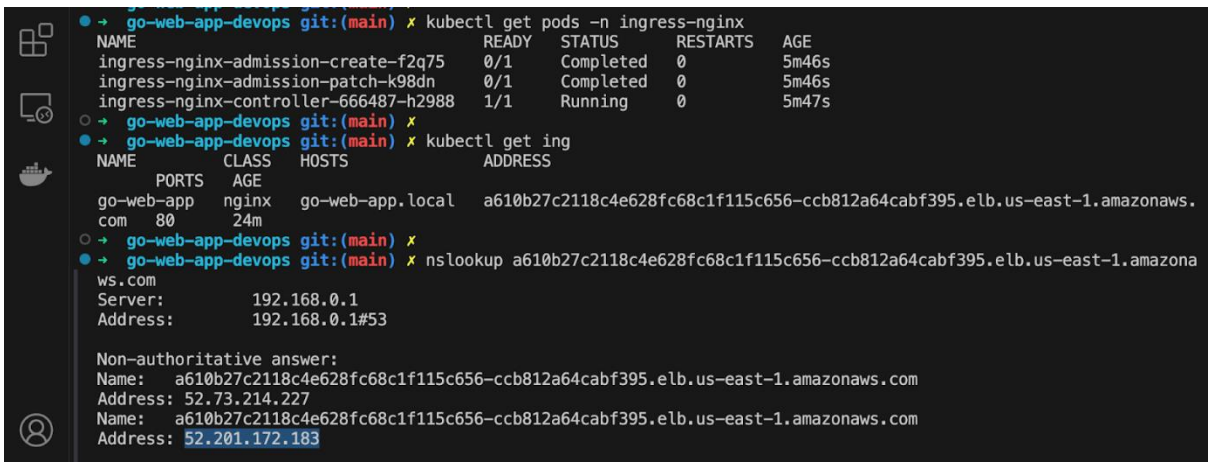
Domain Mapping

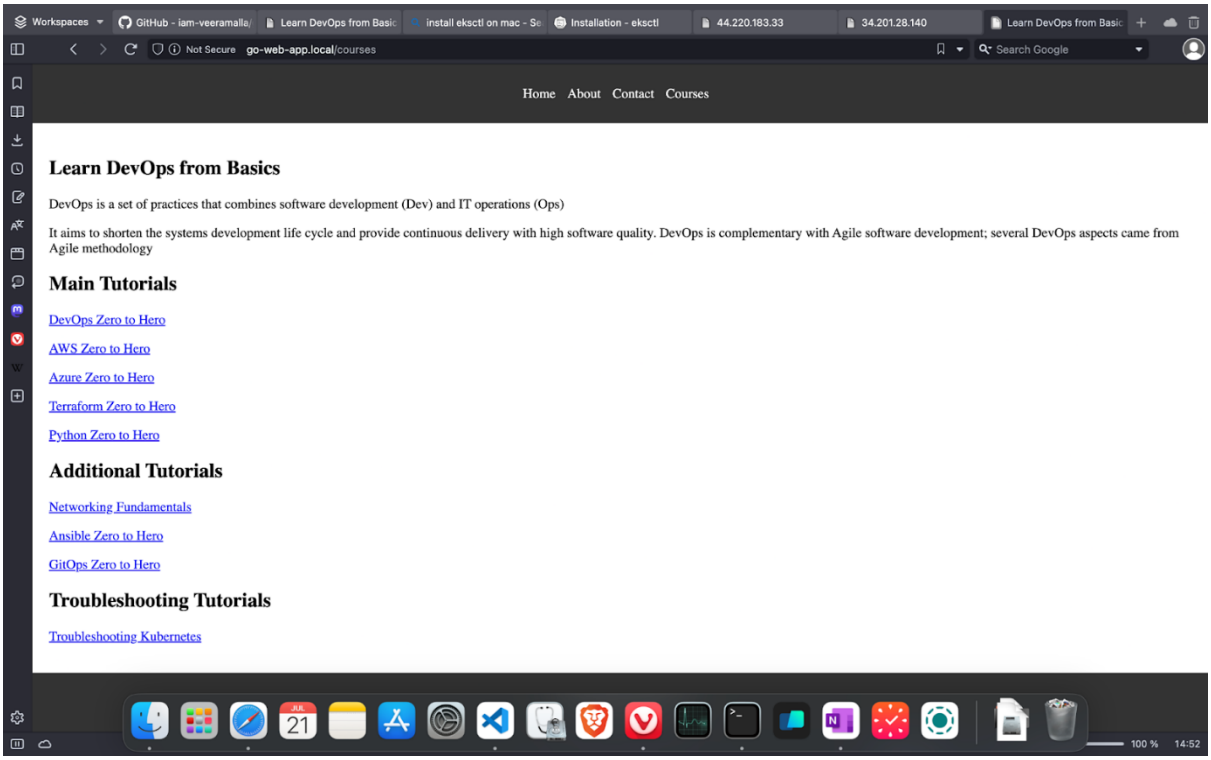
sudo vi /etc/hosts

Add entry

<external-ip> go-web-app.local

- Access the application at http://go-web-app.local/courses.





Helm Charts for Multi-Environment Deployment

Install Helm

brew install helm

helm version

helm create go-web-app-chart

Helm Chart Structure

- **Charts**: Metadata of the chart.
- **Templates**: Kubernetes manifests.
- **Values.yaml**: Environment-specific customization.

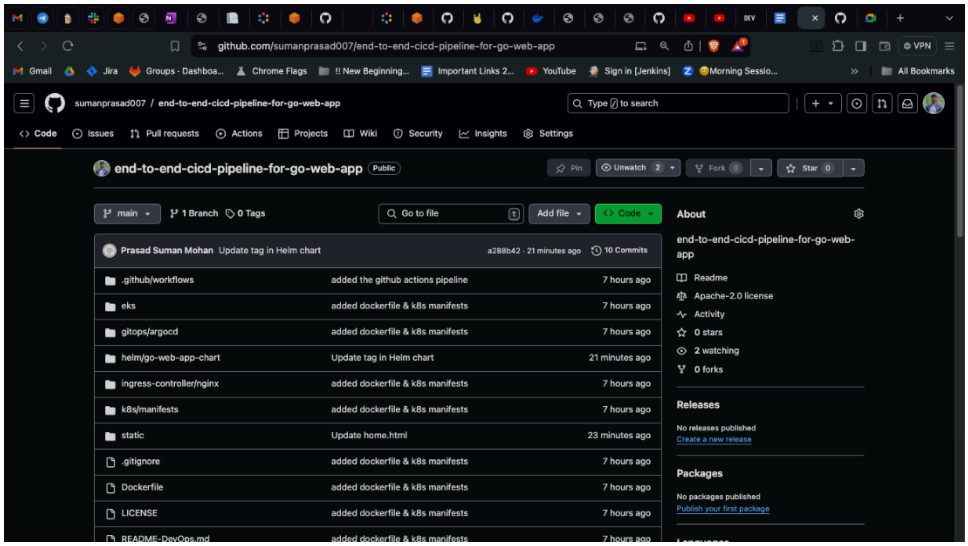
Push Code to GitHub

- Ensure code is pushed to a GitHub repository.

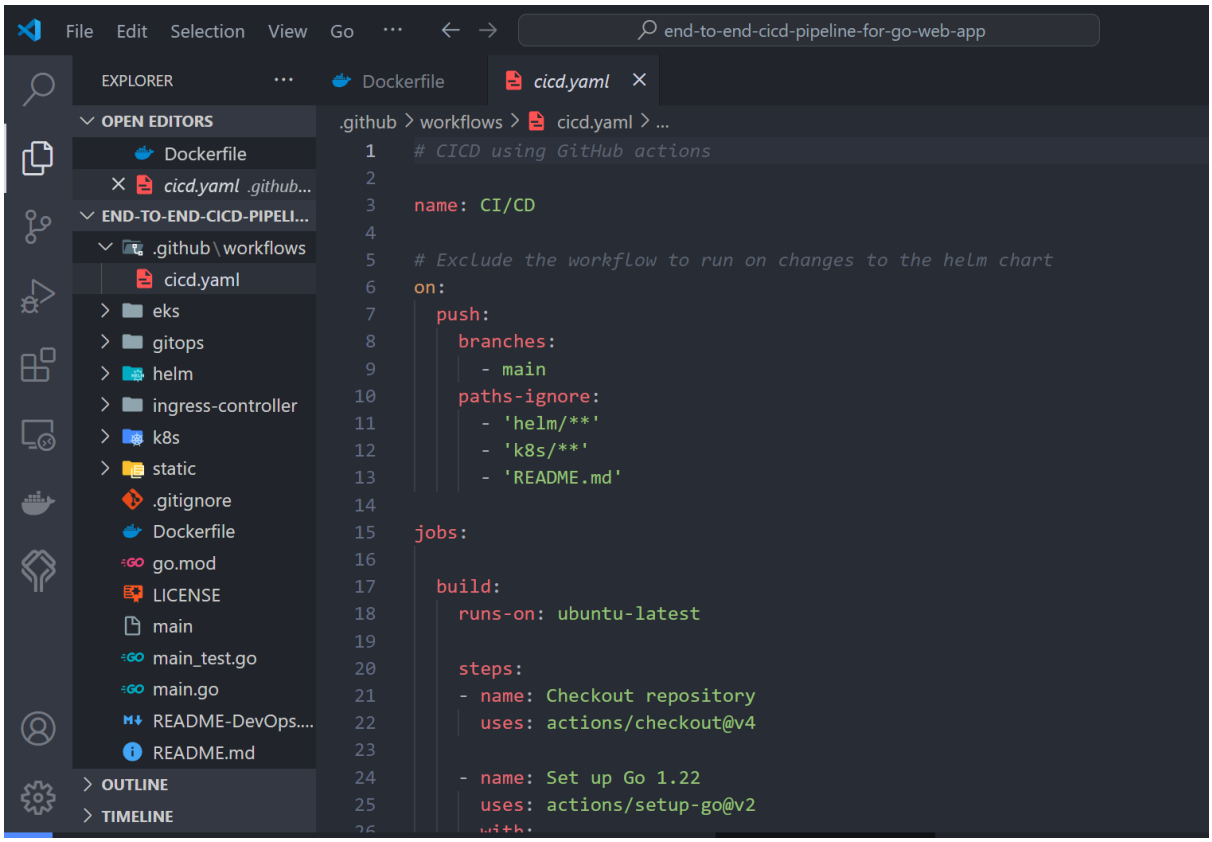
Creating the CI/CD Pipeline with GitHub Actions

Set up Secrets in GitHub Repository

- DOCKERHUB_USERNAME
- DOCKERHUB_TOKEN
- TOKEN



GitHub Actions Workflow

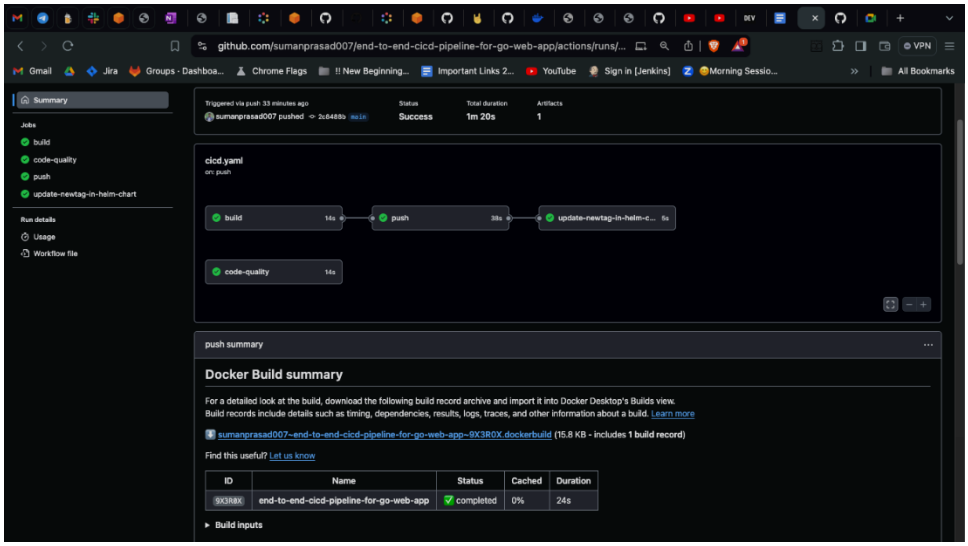


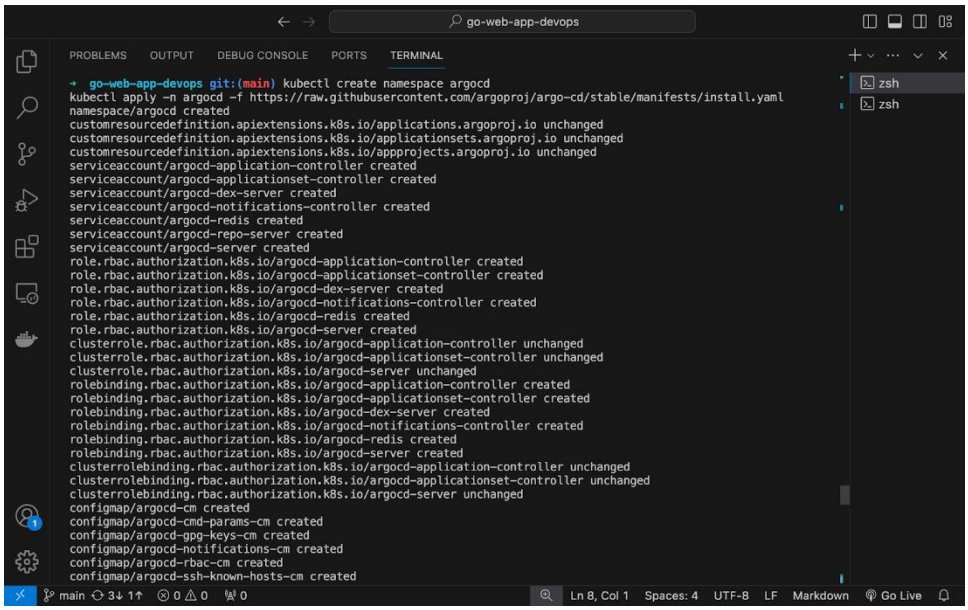
Continuous Deployment with ArgoCD

Install ArgoCD

kubectl create namespace argocd

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





Access ArgoCD UI

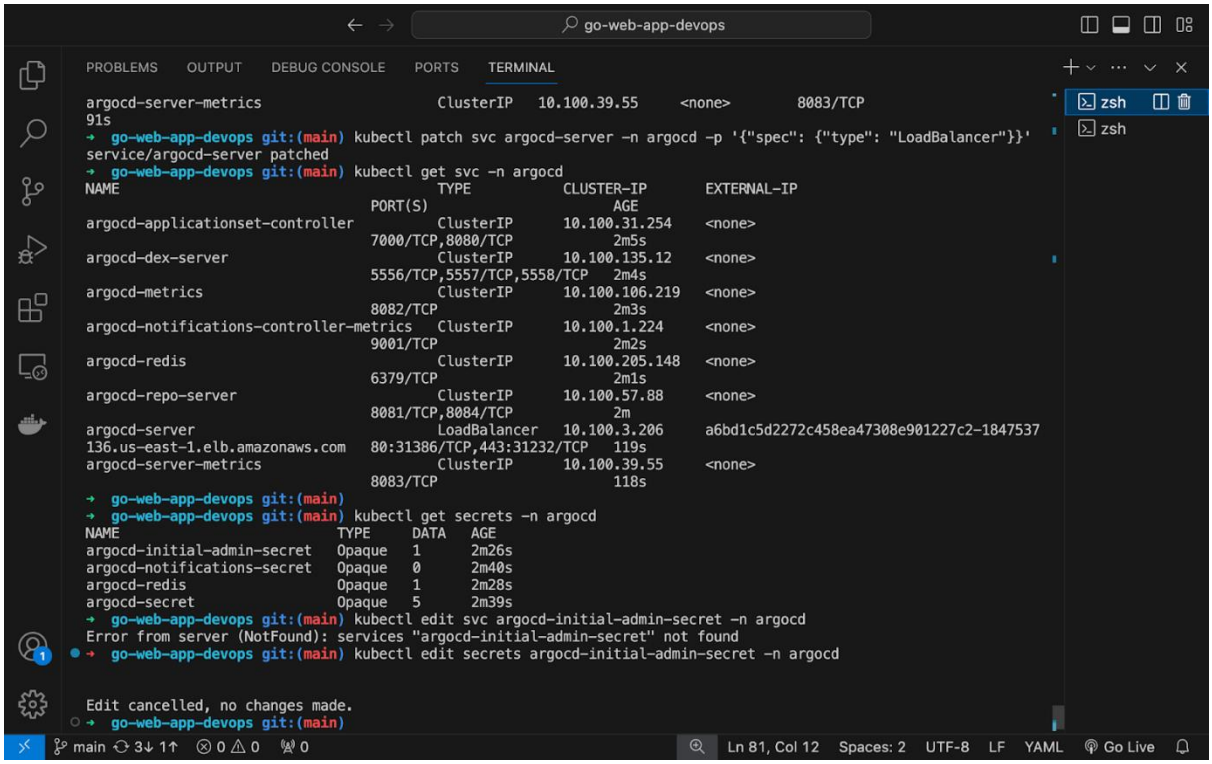
kubectl patch svc argocd-server -n argocd -p '{"spec": {"type": "LoadBalancer"}}'

kubectl get svc argocd-server -n argocd

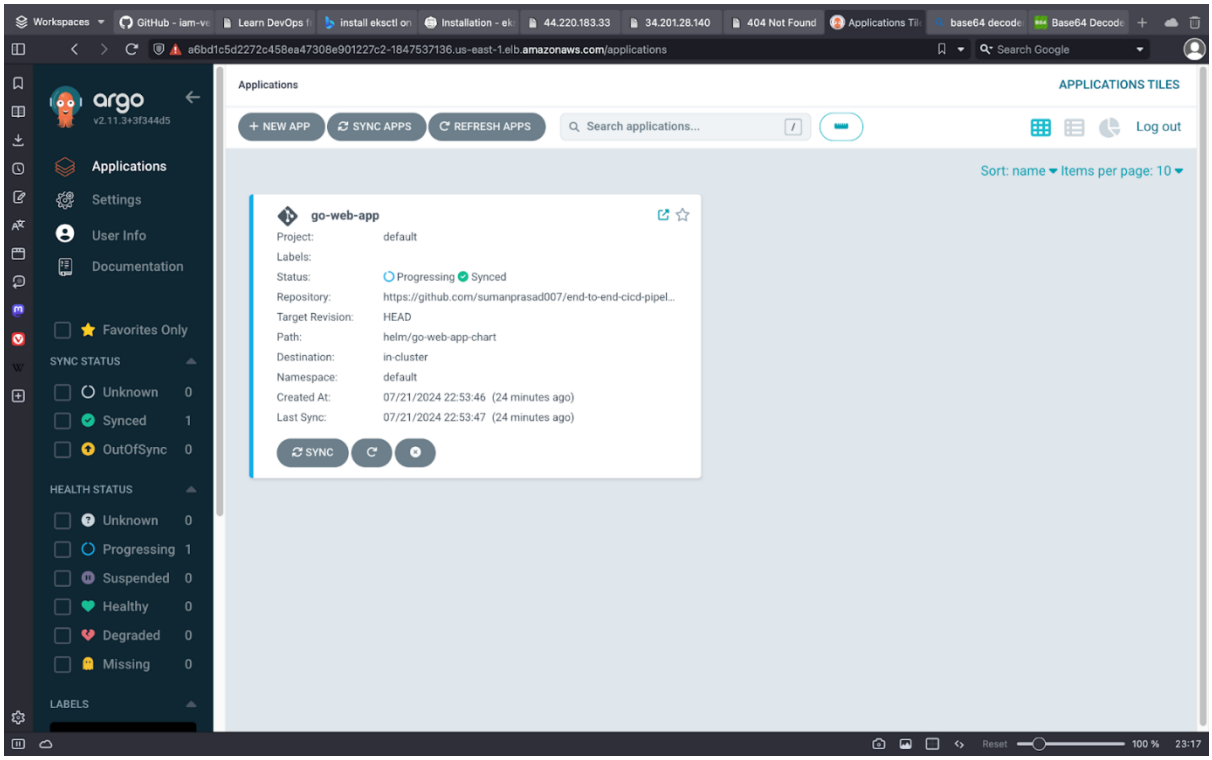
Retrieve Initial Password

kubectl get secret argocd-initial-admin-secret -n argocd -o jsonpath="{.data.password}" | base64 --decode

- Log in to the ArgoCD dashboard with the retrieved password.

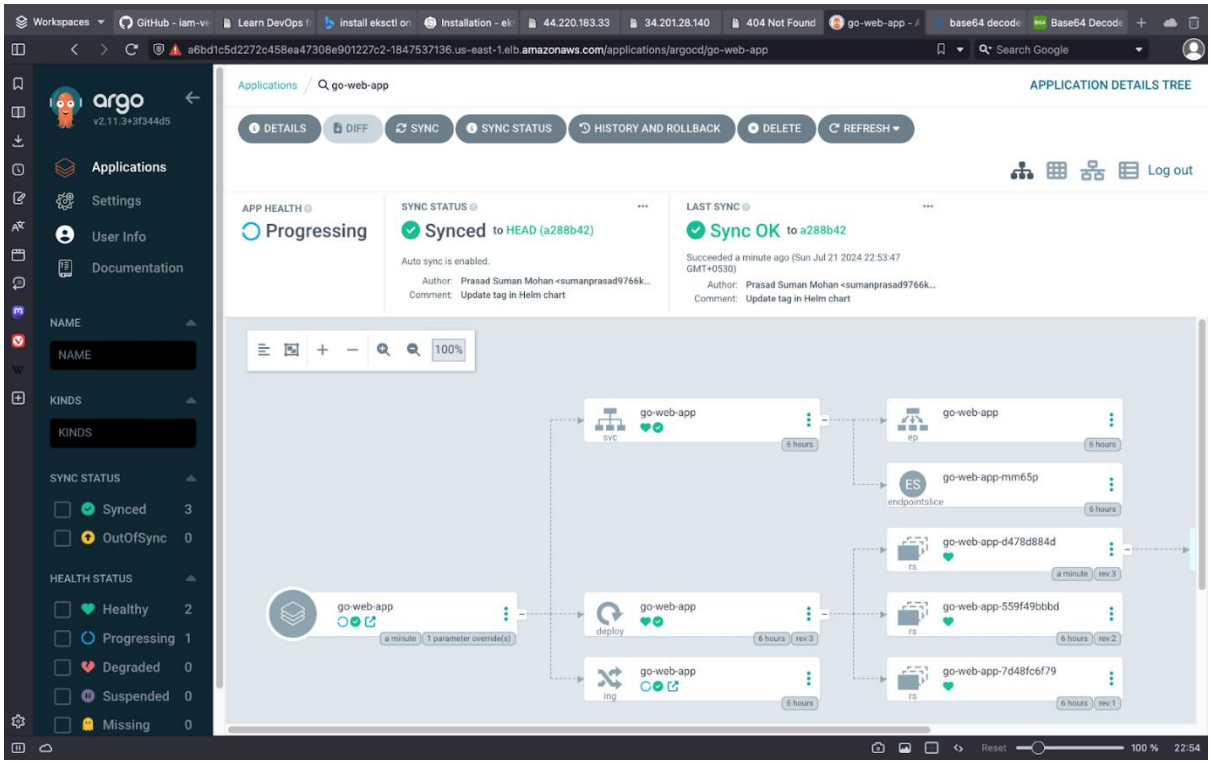


GitOps with ArgoCD

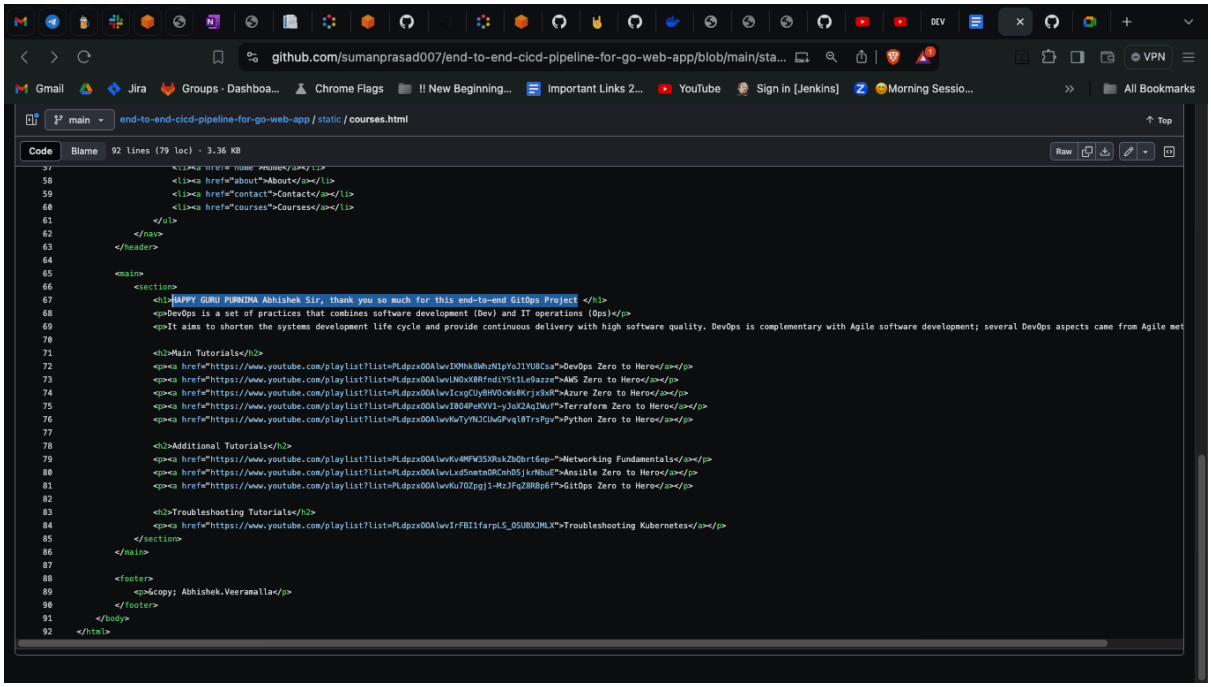


- ArgoCD will monitor the Git repository and deploy the latest Docker image to the Kubernetes cluster whenever changes are detected.

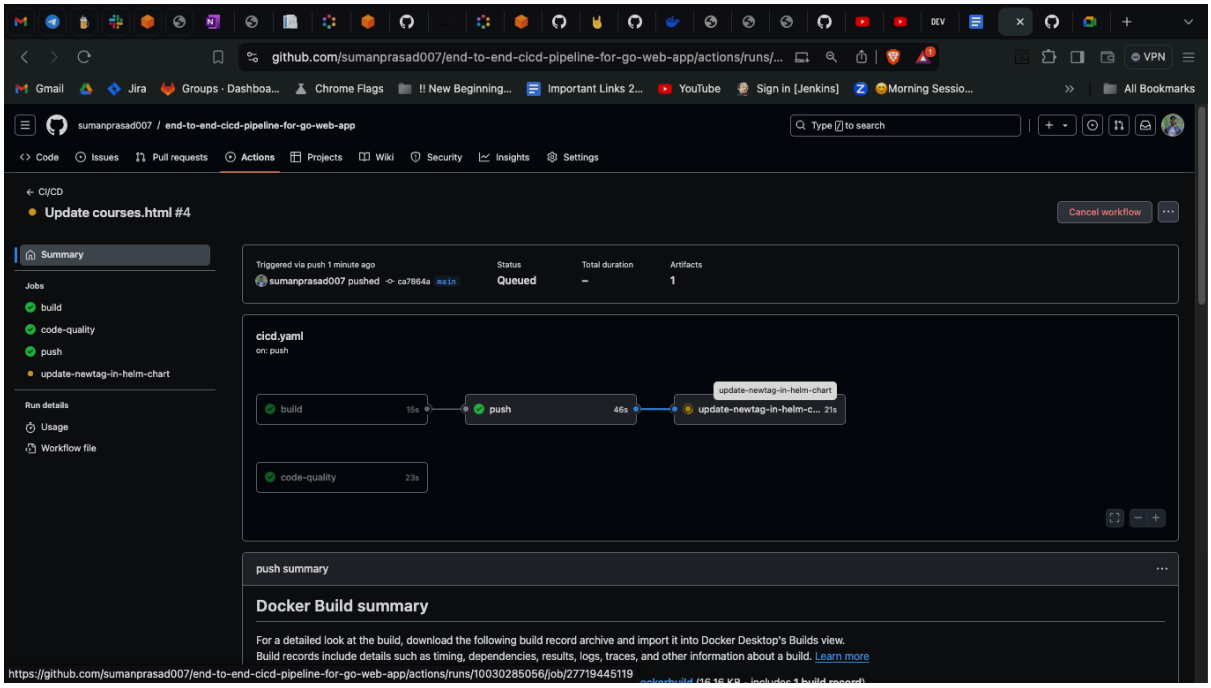
Changes has been deployed to the Kubernetes cluster using the ArgoCD:



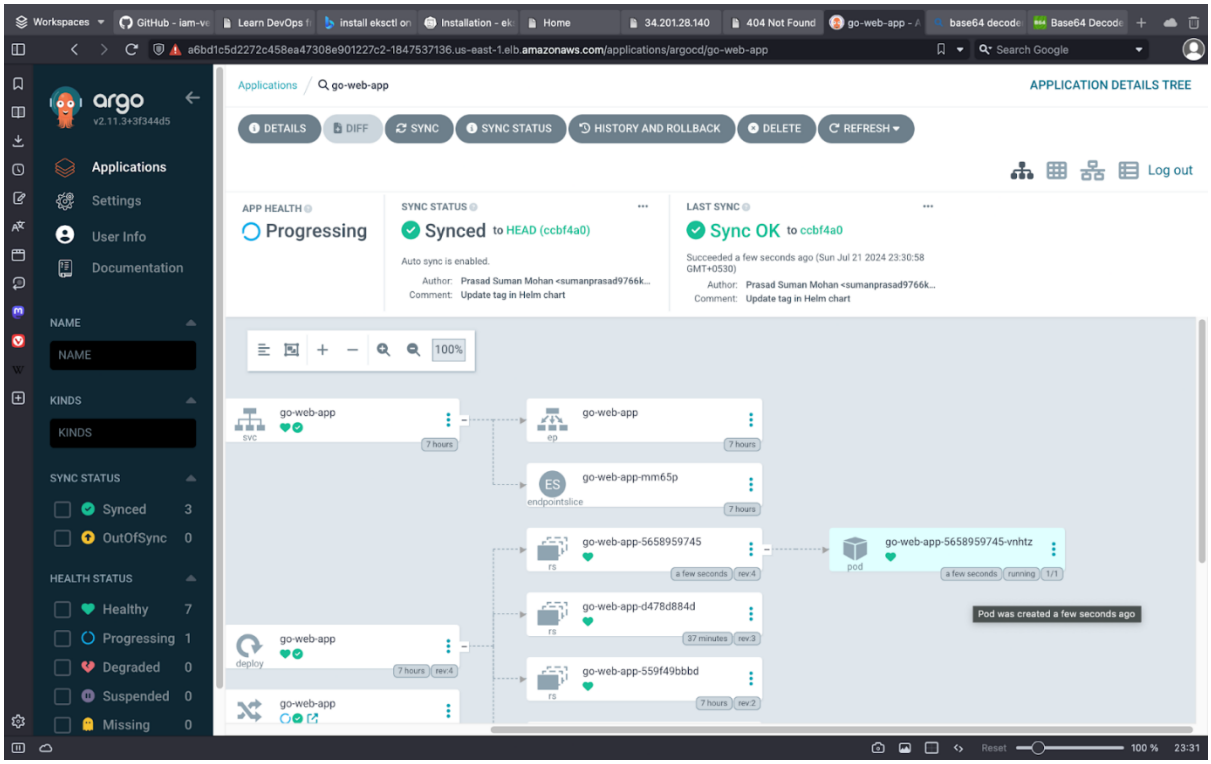
- Every time you make changes to the website, an automated process is triggered. This process builds the code, runs tests, creates a Docker image, uploads it to Docker Hub, and updates the deployment configuration with the new image tag. Argo CD continuously monitors for image tag changes and automatically deploys the updated image to the Kubernetes cluster, reflecting the website changes.



- Triggers the GitHub Action CI Pipelines:

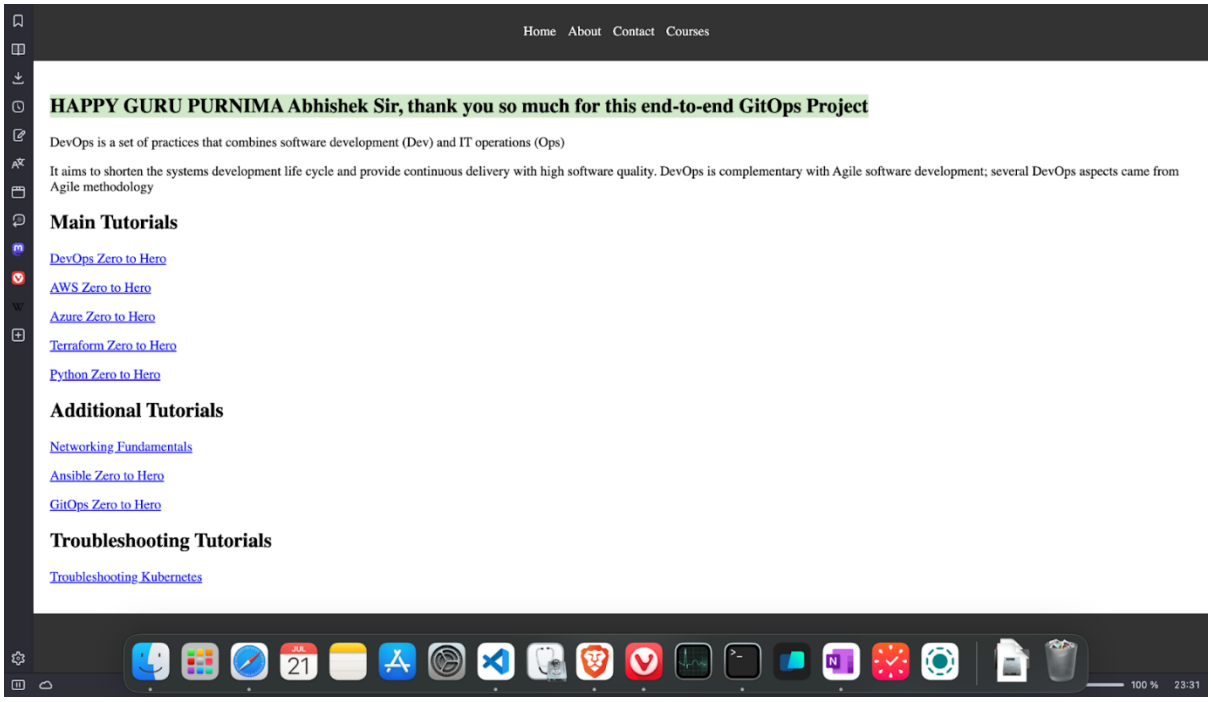


- Argo CD continuously monitors for image tag changes and automatically deploys the updated image to the Kubernetes cluster



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

- Now, we can see it has updated our website:



Resources

- **DevOpsified**: Complete DevOps Implementation
- **Golang Web Application**: GitHub Repository

Celebrate your successful deployment of the application using EKS, Nginx Ingress Controller, Helm, GitHub Actions, and ArgoCD! 🎉