



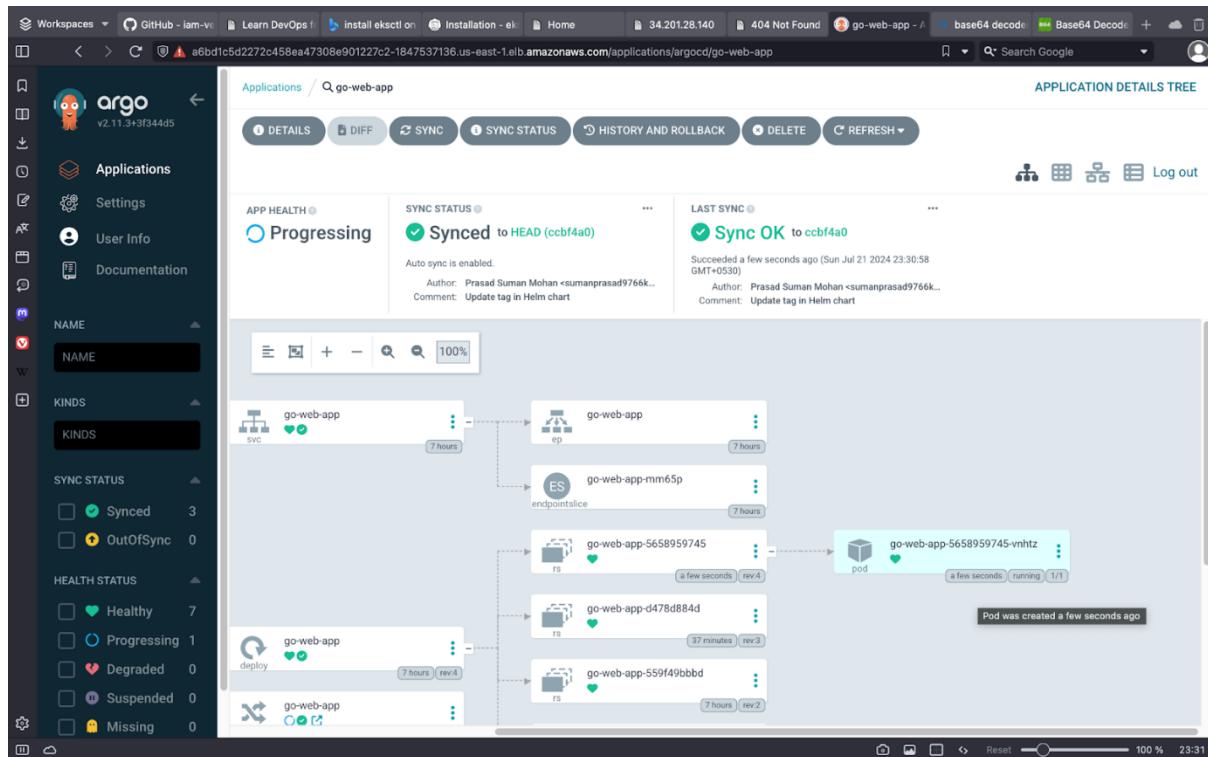
End to end GitOps Project using AWS EKS & ArgoCD

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.

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



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
```

The screenshot shows the VS Code interface with the terminal tab active. The terminal window displays the following command and its output:

```
go web-app-devops
```

```
tool      run specified go tool
version   print Go version
vet       report likely mistakes in packages

Use "go help <command>" for more information about a command.

Additional help topics:

buildconstraint build constraints
buildmode       build modes
c               calling between Go and C
cache          build and test caching
environment    environment variables
filetype        file types
go.mod         the go.mod file
gopath         GOPATH environment variable
importpath     module proxy protocol
modules        import path syntax
module-auth    modules, module versions, and more
packages       module authentication using go.sum
private        package lists and patterns
testflag       configuration for downloading non-public code
testfunc       testing flags
vcs           testing functions
              controlling version control with GOVCS

Use "go help <topic>" for more information about that topic.

● → go web-app-devops git:(main) go version
go version go1.22.5 darwin/amd64
○ → go web-app-devops git:(main)
○ → go web-app-devops git:(main)
● → go web-app-devops git:(main) go build -o main .
○ → go web-app-devops git:(main) x
○ → go web-app-devops git:(main) x
○ → go web-app-devops git:(main) x ./main
```

2. Access the Application

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



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

The screenshot shows a web browser window with the URL localhost:8080/courses. The page content includes:

- Learn DevOps from Basics**: A brief introduction to DevOps.
- Main Tutorials**: Links to "DevOps Zero to Hero", "AWS Zero to Hero", "Azure Zero to Hero", "Terraform Zero to Hero", and "Python Zero to Hero".
- Additional Tutorials**: Links to "Networking Fundamentals", "Ansible Zero to Hero", and "GitOps Zero to Hero".
- Troubleshooting Tutorials**: A link to "Troubleshooting Kubernetes".

Writing a Multi-Stage Dockerfile

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

vi Dockerfile

The screenshot shows the VS Code interface with the Dockerfile open. The code is as follows:

```
about.html Dockerfile M go.mod
Dockerfile > ...
2
3  # Start with a base image
4 FROM golang:1.21 as base
5
6 # Set the working directory inside the container
7 WORKDIR /app
8
9 # Copy the go.mod and go.sum files to the working directory
10 COPY go.mod .
11
12 # Download all the dependencies
13 RUN go mod download
14
15 # Copy the source code to the working directory
16 COPY .
17
18 # Build the application
19 RUN go build -o main .
20
21 #####
22 # Reduce the image size using multi-stage builds
23 # We will use a distroless image to run the application
24 FROM gcr.io/distroless/base
25
26 # Copy the binary from the previous stage
27 COPY --from=base /app/main .
28
29 # Copy the static files from the previous stage
30 COPY --from=base /app/static ./static
31
32 # Expose the port on which the application will run
33 EXPOSE 8080
34
35 # Command to run the application
36 CMD ["./main"]
```



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

```
k8s > manifests > ! deployment.yaml
1 # This is a sample deployment manifest file for a simple web application.
2 apiVersion: apps/v1
3 kind: Deployment
4 metadata:
5   name: go-web-app
6   labels:
7     app: go-web-app
8 spec:
9   replicas: 1
10  selector:
11    matchLabels:
12      app: go-web-app
13  template:
14    metadata:
15      labels:
16        app: go-web-app
17    spec:
18      containers:
19        - name: go-web-app
20          image: sumanprasad007/go-web-app:v1
21          ports:
22            - containerPort: 8080
```

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



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

```
PROBLEMS OUTPUT DEBUG CONSOLE PORTS TERMINAL
2024-07-21 13:58:39 [i] waiting for CloudFormation stack "eksctl-k8s-cluster-cluster"
2024-07-21 13:59:40 [i] waiting for CloudFormation stack "eksctl-k8s-cluster-cluster"
2024-07-21 14:00:41 [i] waiting for CloudFormation stack "eksctl-k8s-cluster-cluster"
2024-07-21 14:01:42 [i] waiting for CloudFormation stack "eksctl-k8s-cluster-cluster"
2024-07-21 14:01:43 [i] creating addon
2024-07-21 14:01:49 [i] successfully created addon
2024-07-21 14:01:50 [i] recommended policies were found for "vpc-cni" addon, but since OIDC is disabled on the cluster, eksctl cannot configure the requested permissions; the recommended way to provide IAM permissions for "vpc-cni" addon is via pod identity annotations; after addon creation is completed, add all recommended policies to the config file, under 'addon.PodIdentityAssociations', and run eksctl update addon
2024-07-21 14:01:50 [i] creating addon
2024-07-21 14:01:51 [i] successfully created addon
2024-07-21 14:01:52 [i] successfully created addon
2024-07-21 14:03:57 [i] building managed nodegroup stack "eksctl-k8s-cluster-nodegroup-ng-2d0607d5"
2024-07-21 14:03:59 [i] deploying stack "eksctl-k8s-cluster-nodegroup-ng-2d0607d5"
2024-07-21 14:03:59 [i] waiting for CloudFormation stack "eksctl-k8s-cluster-nodegroup-ng-2d0607d5"
2024-07-21 14:04:30 [i] waiting for CloudFormation stack "eksctl-k8s-cluster-nodegroup-ng-2d0607d5"
2024-07-21 14:05:20 [i] waiting for CloudFormation stack "eksctl-k8s-cluster-nodegroup-ng-2d0607d5"
2024-07-21 14:06:58 [i] waiting for CloudFormation stack "eksctl-k8s-cluster-nodegroup-ng-2d0607d5"
2024-07-21 14:06:59 [i] waiting for the control plane to become ready
2024-07-21 14:07:00 [✓] saved kubeconfig as "/Users/prasad/.kube/config"
2024-07-21 14:07:00 [✓] no tasks
2024-07-21 14:07:00 [✓] all EKS cluster resources for "k8s-cluster" have been created
2024-07-21 14:07:00 [✓] created 0 nodegroup(s) in cluster "k8s-cluster"
2024-07-21 14:07:01 [i] nodegroup "ng-2d0607d5" has 2 node(s)
2024-07-21 14:07:01 [i] node "ip-192-168-25-33.ec2.internal" is ready
2024-07-21 14:07:01 [i] node "ip-192-168-63-199.ec2.internal" is ready
2024-07-21 14:07:01 [i] waiting for at least 2 node(s) to become ready in "ng-2d0607d5"
2024-07-21 14:07:02 [i] nodegroup "ng-2d0607d5" has 2 node(s)
2024-07-21 14:07:02 [i] node "ip-192-168-25-33.ec2.internal" is ready
2024-07-21 14:07:02 [i] node "ip-192-168-63-199.ec2.internal" is ready
2024-07-21 14:07:02 [✓] created 1 managed nodegroup(s) in cluster "k8s-cluster"
2024-07-21 14:07:03 [i] kubectl command should work with "/Users/prasad/.kube/config", try 'kubectl get nodes'
2024-07-21 14:07:03 [✓] EKS cluster "k8s-cluster" in "us-east-1" region is ready
○ + go-web-app-devops git:(main) ✘
○ + go-web-app-devops git:(main) ✘
```

Apply Kubernetes Manifests

kubectl apply -f k8s/manifests/deployment.yaml

kubectl apply -f k8s/manifests/service.yaml

kubectl apply -f k8s/manifests/ingress.yaml



```
PROBLEMS OUTPUT DEBUG CONSOLE PORTS TERMINAL
+ go-web-app-devops git:(main) ✘ kubectl get nodes -o wide
NAME STATUS ROLES AGE VERSION INTERNAL-IP EXTERNAL-IP
ip-192-168-25-33.ec2.internal Ready <none> 8m38s v1.30.0-eks-036c24b 192.168.25.33 44.220.183.33
Amazon Linux 2 5.10.219-208.866.amzn2.x86_64 containerd://1.7.11
ip-192-168-63-199.ec2.internal Ready <none> 8m32s v1.30.0-eks-036c24b 192.168.63.199 34.201.28.140
Amazon Linux 2 5.10.219-208.866.amzn2.x86_64 containerd://1.7.11
+ go-web-app-devops git:(main) ✘
+ go-web-app-devops git:(main) ✘ kubectl apply -f k8s/manifests/deployment.yaml
deployment.apps/go-web-app created
+ go-web-app-devops git:(main) ✘ kubectl apply -f k8s/manifests/service.yaml
service/go-web-app created
+ go-web-app-devops git:(main) ✘
+ go-web-app-devops git:(main) ✘ kubectl apply -f k8s/manifests/ingress.yaml
ingress.networking.k8s.io/go-web-app created
+ go-web-app-devops git:(main) ✘
+ go-web-app-devops git:(main) ✘ kubectl get pods
NAME READY STATUS RESTARTS AGE
go-web-app-775969bf7b-47jj7 1/1 Running 0 25s
+ go-web-app-devops git:(main) ✘
+ go-web-app-devops git:(main) ✘ kubectl get svc
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
go-web-app ClusterIP 10.100.60.36 <none> 80/TCP 28s
kubernetes ClusterIP 10.100.0.1 <none> 443/TCP 18m
+ go-web-app-devops git:(main) ✘ kubectl edit svc go-web-app
error: services "go-web-app" is invalid
A copy of your changes has been stored to "/var/folders/k9/0f38lcqx4ll0y7mzyplx6yx40000gn/T/kubectl-edit-237152892
0.yaml"
error: Edit cancelled, no valid changes were saved.
● + go-web-app-devops git:(main) ✘ kubectl edit svc go-web-app
service/go-web-app edited
○ + go-web-app-devops git:(main) ✘
● + go-web-app-devops git:(main) ✘ kubectl get svc
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
go-web-app NodePort 10.100.60.36 <none> 80:30429/TCP 2m11s
kubernetes ClusterIP 10.100.0.1 <none> 443/TCP 20m
○ + go-web-app-devops git:(main) ✘
```

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.

The screenshot shows the AWS EC2 Dashboard with the 'Security Groups' tab selected. A specific security group named 'eksctl-k8s-cluster-cluster-ClusterSharedNodeSecurityGroup-4vbQUAc61Jld' is being viewed. The 'Details' section shows the security group name, ID, and description. The 'Inbound rules' tab is selected, displaying three rules:

Source	Protocol	Port range
0.0.0.0/0	TCP	30000 - 33000
sg-02bd9244	All	All
sg-07b502dc	All	All

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



Nginx Ingress Controller Setup

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

```
PROBLEMS OUTPUT DEBUG CONSOLE PORTS TERMINAL

● + go-web-app-devops git:(main) ✘ kubectl get svc
NAME          TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
go-web-app   NodePort   10.100.60.36   <none>       80:30429/TCP  2m11s
kubernetes   ClusterIP  10.100.0.1    <none>       443/TCP       20m

○ + go-web-app-devops git:(main) ✘
○ + go-web-app-devops git:(main) ✘
○ + go-web-app-devops git:(main) ✘
● + go-web-app-devops git:(main) ✘ kubectl get svc
NAME          TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
go-web-app   NodePort   10.100.60.36   <none>       80:30429/TCP  10m
kubernetes   ClusterIP  10.100.0.1    <none>       443/TCP       28m

○ + go-web-app-devops git:(main) ✘
○ + go-web-app-devops git:(main) ✘
● + go-web-app-devops git:(main) ✘ kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.11.1/deploy/static/provider/aws/deploy.yaml

namespace/ingress-nginx created
serviceaccount/ingress-nginx created
serviceaccount/ingress-nginx-admission created
role.rbac.authorization.k8s.io/ingress-nginx created
role.rbac.authorization.k8s.io/ingress-nginx-admission created
clusterrole.rbac.authorization.k8s.io/ingress-nginx-created
clusterrole.rbac.authorization.k8s.io/ingress-nginx-admission created
rolebinding.rbac.authorization.k8s.io/ingress-nginx created
rolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
configmap/ingress-nginx-controller created
service/ingress-nginx-controller created
service/ingress-nginx-controller-admission created
deployment.apps/ingress-nginx-controller created
job.batch/ingress-nginx-admission-create created
job.batch/ingress-nginx-admission-patch created
ingressclass.networking.k8s.io/nginx created
validating webhook configuration.admissionregistration.k8s.io/ingress-nginx-admission created
○ + go-web-app-devops git:(main) ✘
○ + go-web-app-devops git:(main) ✘

Ln 6, Col 1 (133 selected) Spaces: 4 UTF-8 LF Markdown Go Live
```

Domain Mapping

```
sudo vi /etc/hosts
```

```
# Add entry
```

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

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

```
READY STATUS RESTARTS AGE
ingress-nginx-admission-create-f2q75 0/1 Completed 0 5m46s
ingress-nginx-admission-patch-k98dn 0/1 Completed 0 5m46s
ingress-nginx-controller-666487-h2988 1/1 Running 0 5m47s

NAME CLASS HOSTS ADDRESS
PORTS AGE
go-web-app nginx go-web-app.local a610b27c2118c4e628fc68c1f115c656-ccb812a64cabf395.elb.us-east-1.amazonaws.com 80 24m

Non-authoritative answer:
Name: a610b27c2118c4e628fc68c1f115c656-ccb812a64cabf395.elb.us-east-1.amazonaws.com
Address: 52.73.214.227
Name: a610b27c2118c4e628fc68c1f115c656-ccb812a64cabf395.elb.us-east-1.amazonaws.com
Address: 52.201.172.183
```



The screenshot shows a web browser window with the following details:

- Address Bar:** go-web-app.local/courses
- Page Title:** Learn DevOps from Basics
- Page Content:**
 - Section 1:** **Learn DevOps from Basics**
 - Text: DevOps is a set of practices that combines software development (Dev) and IT operations (Ops)
 - Text: It aims to shorten the systems development life cycle and provide continuous delivery with high software quality. DevOps is complementary with Agile software development; several DevOps aspects came from Agile methodology
 - Section 2:** **Main Tutorials**
 - [DevOps Zero to Hero](#)
 - [AWS Zero to Hero](#)
 - [Azure Zero to Hero](#)
 - [Terraform Zero to Hero](#)
 - [Python Zero to Hero](#)
 - Section 3:** **Additional Tutorials**
 - [Networking Fundamentals](#)
 - [Ansible Zero to Hero](#)
 - [GitOps Zero to Hero](#)
 - Section 4:** **Troubleshooting Tutorials**
 - [Troubleshooting Kubernetes](#)
- Bottom Bar:** macOS Dock with various application icons.

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

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



The screenshot shows a GitHub repository page for 'end-to-end-cicd-pipeline-for-go-web-app'. The repository has 1 branch and 0 tags. The commit history shows 10 commits from 'Prasad Suman Mohan' over the last 21 minutes. The commits include updates to Helm charts, GitHub Actions pipelines, Dockerfiles, K8S manifests, and static files. The repository has 0 stars, 2 watchers, and 0 forks. It also includes sections for Releases, Packages, and Languages.

GitHub Actions Workflow

The screenshot shows a GitHub Actions workflow configuration in a VS Code editor. The workflow file is named 'cicd.yaml' and is located in the '.github\workflows' directory. The configuration includes a CI/CD job that runs on pushes to the 'main' branch, ignoring changes to the 'helm', 'k8s', and 'README.md' files. The job involves checking out the repository, setting up Go 1.22, and running tests.

```
# CICD using GitHub actions
name: CI/CD

# Exclude the workflow to run on changes to the helm chart
on:
  push:
    branches:
      - main
  paths-ignore:
    - 'helm/**'
    - 'k8s/**'
    - 'README.md'

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout repository
        uses: actions/checkout@v4
      - name: Set up Go 1.22
        uses: actions/setup-go@v2
        with:
```

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>

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



The screenshot shows a GitHub Actions pipeline summary. It indicates a successful run triggered via push 33 minutes ago by user sumanprasad007. The total duration was 1m 20s, and there was 1 artifact. The workflow file is named cicd.yaml and contains three steps: build, push, and update-newtag-in-helm-chart. The code-quality step took 16s. The push step took 38s. The update-newtag-in-helm-chart step took 5s. The Docker Build summary table shows one build record with ID 9X3RX, status completed, and duration 24s.

The screenshot shows a terminal window with the title bar "go-web-app-devops". The command `kubectl create namespace argocd` is run, followed by `kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml`. The output shows various ArgoCD resources being created, including namespaces, custom resource definitions, service accounts, application controllers, dex servers, notifications controllers, redis instances, and RBAC roles and bindings. The terminal also shows configuration map creation for cmd-params, gpg-keys, notifications, and rbac.

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.

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



```
go-web-app-devops git:(main) kubectl patch svc argocd-server -n argocd -p '{"spec": {"type": "LoadBalancer"}}'
service/argocd-server patched
go-web-app-devops git:(main) kubectl get svc -n argocd
NAME           TYPE        CLUSTER-IP      EXTERNAL-IP
argocd-applicationset-controller   ClusterIP   10.100.31.254   <none>
argocd-dex-server          ClusterIP   10.100.135.12    2m5s
argocd-metrics            ClusterIP   10.100.106.219   <none>
argocd-notifications-controller-metrics ClusterIP   10.100.1.224    2m4s
argocd-redis              ClusterIP   10.100.205.148   <none>
argocd-repo-server         ClusterIP   10.100.57.88     2m2s
argocd-server             LoadBalancer 10.100.3.206    a6bd1c5d2272c458ea47308e901227c2-1847537
136.us-east-1.elb.amazonaws.com  80:31386/TCP,443:31232/TCP 119s
argocd-server-metrics       ClusterIP   10.100.39.55     118s
go-web-app-devops git:(main) kubectl get secrets -n argocd
NAME          TYPE  DATA  AGE
argocd-initial-admin-secret  Opaque  1    2m26s
argocd-notifications-secret Opaque  0    2m40s
argocd-redis           Opaque  1    2m28s
argocd-secret          Opaque  5    2m39s
go-web-app-devops git:(main) kubectl edit svc argocd-initial-admin-secret -n argocd
Error from server (NotFound): services "argocd-initial-admin-secret" not found
● go-web-app-devops git:(main) kubectl edit secrets argocd-initial-admin-secret -n argocd
Edit cancelled, no changes made.
go-web-app-devops git:(main)
```

Ln 81, Col 12 Spaces: 2 UTF-8 LF YAML Go Live

GitOps with ArgoCD

The screenshot shows the ArgoCD UI with the following details:

- Left Sidebar:** Includes sections for Workspaces, GitHub integration, Learn DevOps, and various status counts (Unknown: 0, Synced: 1, OutOfSync: 0).
- Central Area:** Shows the "go-web-app" application configuration. Key details:
 - Project: default
 - Status: Progressing Synced
 - Repository: https://github.com/sumanprasad007/end-to-end-cicd-pipel...
 - Target Revision: HEAD
 - Path: helm/go-web-app-chart
 - Destination: in-cluster
 - Namespace: default
 - Created At: 07/21/2024 22:53:46 (24 minutes ago)
 - Last Sync: 07/21/2024 22:53:47 (24 minutes ago)
- Bottom Navigation:** SYNC, CANCEL, and REVERT buttons.

- 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:

The screenshot shows the ArgoCD interface for the 'go-web-app' application. The left sidebar lists 'Applications' (go-web-app), 'Settings', 'User Info', and 'Documentation'. The main area displays the 'APPLICATION DETAILS TREE' with tabs for 'DETAILS', 'DIFF', 'SYNC', 'SYNC STATUS', 'HISTORY AND ROLLBACK', 'DELETE', and 'REFRESH'. The 'SYNC STATUS' tab shows 'Synced to HEAD (a288b42)' with an 'Auto sync is enabled.' message. The 'LAST SYNC' section shows a 'Sync OK' event to 'a288b42' at 'Succeeded a minute ago (Sun Jul 21 2024 22:53:47 GMT+0530)'. The 'HISTORY AND ROLLBACK' tab is selected, showing a timeline of events: 'Sync OK to a288b42' (Synced to HEAD), 'Deploy go-web-app' (a minute ago), and 'Sync OK to a288b42' (Synced to HEAD again). Below this, a detailed deployment history shows various pods and services: 'go-web-app' (SVC), 'go-web-app' (ep), 'ES endpointslice', 'go-web-app-mm65p', 'go-web-app-d478d884d', 'go-web-app-559f49bbd', and 'go-web-app-7d48fc6f79'. Each pod entry includes its status (green), revision (rev1, rev2, rev3), and last updated time.

- 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.

The screenshot shows a browser window displaying a GitHub page for a project named 'end-to-end-cicd-pipeline-for-go-web-app'. The page contains a large block of HTML code, likely a course or tutorial content. The code includes various sections such as 'Main Tutorials', 'Additional Tutorials', and 'Troubleshooting Tutorials', each with links to YouTube playlists. The code also includes comments and blame information for the file.

```
<!-- Main Header -->
58     <li><a href="#about">About</a></li>
59     <li><a href="#contact">Contact</a></li>
60     <li><a href="#courses">Courses</a></li>
61   </ul>
62 </header>
63
64 <main>
65   <section>
66     <h2>HAPPY GURU PUNJIMA Abhishek Sir, thank you so much for this end-to-end GitOps Project </h2>
67     <p>DevOps is a set of practices that combines software development (Dev) and IT operations (Ops)</p>
68     <p>It aims to shorten the systems development life cycle and provide continuous delivery with high software quality. DevOps is complementary with Agile software development; several DevOps aspects came from Agile met</p>
69
70     <h3>Main Tutorials</h3>
71     <a href="https://www.youtube.com/playlist?list=PLdpzxQ0A1wvD0H8whpJpYo31VUCs=">DevOps Zero to Hero</a>
72     <a href="https://www.youtube.com/playlist?list=PLdpzxQ0A1wLNxDX0fdiyStLeGazz=">AWS Zero to Hero</a>
73     <a href="https://www.youtube.com/playlist?list=PLdpzxQ0A1w1cx0Bh0Ck6kRkjxob=">Azure Zero to Hero</a>
74     <a href="https://www.youtube.com/playlist?list=PLdpzxQ0A1w1804pKXV1-yJoxZAGIMU=">Terraform Zero to Hero</a>
75     <a href="https://www.youtube.com/playlist?list=PLdpzxQ0A1wvTyNCJluGpq18TrPgv">Python Zero to Hero</a>
76     <a href="https://www.youtube.com/playlist?list=PLdpzxQ0A1wvTqZB8I1farpl5_0SUBXJMLK">Troubleshooting Kubernetes</a>
77
78     <h3>Additional Tutorials</h3>
79     <a href="https://www.youtube.com/playlist?list=PLdpzxQ0A1wvK4MFw35XskZBqort8Ep->Networking Fundamentals</a>
80     <a href="https://www.youtube.com/playlist?list=PLdpzxQ0A1wLxd5nmt0RCh05jkrNbeE">Ansible Zero to Hero</a>
81     <a href="https://www.youtube.com/playlist?list=PLdpzxQ0A1wvKu70Zpg1-MzJFqZ8R8p6f">GitOps Zero to Hero</a>
82
83     <h3>Troubleshooting Tutorials</h3>
84     <a href="https://www.youtube.com/playlist?list=PLdpzxQ0A1wv1rFBt1farpl5_0SUBXJMLK">Troubleshooting Kubernetes</a>
85
86   </section>
87
88   <footer>
89     <p>&copy; Abhishek.Veeramalla</p>
90   </footer>
91 </main>
```

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



- Triggers the GitHub Action CI Pipelines:

The screenshot shows a GitHub Actions pipeline run for a workflow named "Update courses.html #4". The pipeline has four jobs: build, code-quality, push, and update-newtag-in-helm-chart. The push job is currently queued. The update-newtag-in-helm-chart job has a duration of 21s. A "push summary" section shows a Docker Build summary with a download link.

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

The screenshot shows the Argo CD web interface for the "go-web-app" application. It displays sync status, deployment history, and a detailed view of the Kubernetes resources (EVC, go-web-app, go-web-app-mm65p, go-web-app-5658959745, go-web-app-d478d84d, go-web-app-559f49bbcd, pod) with their current status and revision numbers.

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



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

HAPPY GURU PURNIMA Abhishek Sir, thank you so much for this end-to-end GitOps Project

DevOps is a set of practices that combines software development (Dev) and IT operations (Ops)

It aims to shorten the systems development life cycle and provide continuous delivery with high software quality. DevOps is complementary with Agile software development; several DevOps aspects came from Agile methodology

Main Tutorials

- [DevOps Zero to Hero](#)
- [AWS Zero to Hero](#)
- [Azure Zero to Hero](#)
- [Terraform Zero to Hero](#)
- [Python Zero to Hero](#)

Additional Tutorials

- [Networking Fundamentals](#)
- [Ansible Zero to Hero](#)
- [GitOps Zero to Hero](#)

Troubleshooting Tutorials

- [Troubleshooting Kubernetes](#)

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! 🎉



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