GitOps Workflow Using ArgoCD on Kubernetes

Introduction

This project demonstrates a GitOps-based deployment workflow using ArgoCD and K3s on an AWS EC2 instance. GitOps simplifies Kubernetes application management by continuously syncing cluster state with a version-controlled GitHub repository.

Abstract

GitOps introduces declarative infrastructure and applications managed via Git. In this implementation, K3s serves as the lightweight Kubernetes distribution hosted on a t2.medium EC2 instance. ArgoCD monitors a GitHub repository for changes to Kubernetes manifests and deploys them automatically, improving the deployment speed, traceability, and rollback capabilities.

Tools Used:

- **EC2 (Ubuntu)**: Cloud infrastructure host
- **K3s**: Lightweight Kubernetes distribution
- **ArgoCD**: GitOps deployment controller
- **GitHub**: Version control for manifests
- **Docker**: Building and pushing container image

Steps Involved in Building the Project

- Took ubuntu with t2.medium Instance
- Installed k3s on Ec2 → curl -sfL https://get.k3s.io | sh -
- checking the nodes → kubectl get nodes
- setting kubectl for current user

```
mkdir -p $HOME/.kube
sudo cp /etc/rancher/k3s/k3s.yaml $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

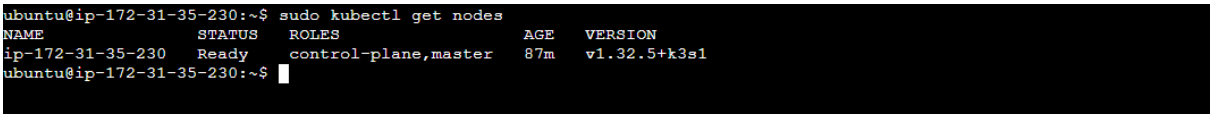
- Installing ArgoCD

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

- Accessing ArgoCD securely thru local from terminal

```
nohup kubectl port-forward svc/argocd-server -n argocd 8080:8083 > portforward.log
ssh -i "YourKey.pem" -L 8080:localhost:8080 ubuntu@your-remote-ip
```

```
sudo kubectl get nodes
```



- kubectl edit svc argocd-server -n argocd → Changing the clusterIP to NodePort
- Now we can access the ArgoCD UI thru local host → localhost:8080
- to get the ArgoCD password to login

```
kubectl get secret argocd-initial-admin-secret -n argocd -o jsonpath="{.data.password
```

- Creating deployment.yaml, service.yaml for my Nginx application, then
- configuring application.yaml files to sync my Git repository with my Kubernetes cluster
- pushing them to my Github Repo
- then refreshing the ArgoCD
- Updated replicas via git commits,
- kubectl get svc nginx-service → gives the port no. for nginx,
- enabling the port in security group of the instance and then accessing it.

Conclusion

This project successfully demonstrates the implementation of GitOps principles using ArgoCD, achieving streamlined and automated Kubernetes deployments. By syncing with a GitHub repo, it enables transparent updates, rollback, and improved operational efficiency.