GitOps with ArgoCD and Kubernetes

1. Project Overview:

This project demonstrates how to implement GitOps — a modern approach to automating Kubernetes deployments — using ArgoCD. It shows how application manifests stored in a GitHub repository can be deployed automatically to a Kubernetes cluster (K3s) running on an AWS EC2 Ubuntu server. Every change made to the Git repository is automatically reflected in the cluster through ArgoCD.

2. Objectives:

- Use Git as the single source of truth for Kubernetes deployments.

- Automatically synchronize application state from GitHub to Kubernetes using ArgoCD.

- Deploy and manage a simple HTTP-based application.

- Demonstrate full GitOps workflow — from committing code to auto-deployment.

3. Tools and Technologies Used:

- ArgoCD – a declarative GitOps continuous delivery tool for Kubernetes.

- K3s – a lightweight, easy-to-install Kubernetes distribution.

- GitHub – for storing Kubernetes manifest files.

- EC2 Ubuntu – the environment to run Kubernetes and ArgoCD.

- Docker – used for container operations.

4. System Architecture:

- EC2 instance hosts K3s.

- ArgoCD is installed within the K3s cluster.

- Kubernetes manifests (deployment, service, kustomization) are stored in a public/private GitHub repo.

- ArgoCD connects to the GitHub repo, watches for changes, and syncs them to the cluster.

- The app is exposed via a NodePort service, allowing external access.

5. Step-by-Step Implementation:

A. Launch EC2 Instance

- Start a t2.medium EC2 Ubuntu 20.04/22.04 instance.
- Open ports: 22 (SSH), 30080 (ArgoCD UI), 32000–32767 (NodePorts).

B. Set Up Environment

- SSH into EC2.
- Install essential packages: curl, git, docker, etc.
- Install Docker (if needed): curl -fsSL https://get.docker.com | bash

C. Install K3s

. Run: curl -sfL https://get.k3s.io | sh -

- Alias kubectl to use K3s's version:
  echo "alias kubectl='sudo k3s kubectl'" >> ~/.bashrc

D. Install ArgoCD

- Create namespace: kubectl create namespace argocd
- Deploy ArgoCD: kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml
- Expose ArgoCD via NodePort:
  kubectl patch svc argocd-server -n argocd -p '{"spec":{"type":"NodePort","ports":[{"port":80,"targetPort":8080,"nodePort":30080}]}}'

E. Access ArgoCD UI

- Visit: http://<EC2-PUBLIC-IP>:30080
- Default username: admin
- Get password:
  kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.password}" | base64 -d

F. Prepare GitHub Repository:

- Create a repo named Devopintership-project1
- Clone the repo: (git clone https://github.com/arjumandshafi/Devopintership-project1)
- Add the following files:
  1. deployment.yaml
  2. service.yaml
  3. kustomization.yaml

G. Deploy App with ArgoCD:

- Go to ArgoCD UI > NEW APP
- Fill in:
  - Name: hello-app
  - Repo URL: GitHub repo link
  - Path: .
  - Cluster: https://kubernetes.default.svc
  - Namespace: default
  - Sync Policy: automatic or manual
- Click Create. ArgoCD will sync the app and deploy it to the cluster.

H. Access the App:

- kubectl get svc hello-app
- Note the NodePort
- Open: http://<EC2-IP>:<NodePort>

I. Demonstrate GitOps:

- Change the app message in deployment.yaml (e.g., "-text=Hello from GitOps!")
- Commit and push to GitHub.
- ArgoCD detects the change and syncs it.
- Refresh the browser — the message is updated.