Introduction

The purpose of this project is to implement a **GitOps pipeline** using **ArgoCD** on Kubernetes. GitOps ensures that the **Git repository is the single source of truth**, enabling automatic synchronization of application states between Git and the Kubernetes cluster.

Abstract

This project demonstrates deploying a sample application (NGINX) on a Kubernetes cluster and managing it through ArgoCD.

- Application manifests (Deployment, Service) are stored in GitHub.
- ArgoCD continuously monitors the repository and **automatically applies updates** to the cluster.
- The workflow enables **version-controlled deployments**, **auditable changes**, and **reliable automation** of Kubernetes applications.

Tools Used

- **Kubernetes** (K3s / Minikube) – Lightweight cluster for local testing
- **ArgoCD** – GitOps continuous deployment controller
- **Docker Hub** – Application images
- **GitHub** – Source repository for manifests
- **kubectl** – Kubernetes CLI
- **Optional**: VS Code or any text editor for YAML configuration

---

Steps Involved in Building the Project

1. **Setup Kubernetes Cluster**
    - Installed K3s / Minikube on local machine or EC2 instance.
2. **Install ArgoCD**
3. kubectl create namespace argocd
4. kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml
5. **Access ArgoCD UI**

- Port-forward ArgoCD service or expose via NodePort.
- Retrieve admin password:
- kubectl get secret argocd-initial-admin-secret -n argocd \
- -o jsonpath="{.data.password}" | base64 -d

6. **Prepare Application Manifests**
    - deployment.yaml → NGINX Deployment
    - service.yaml → NodePort service for external access
7. **Push Manifests to GitHub**
    - Initialize repo, commit files, and push.
8. **Configure ArgoCD Application**
    - Set **Repo URL**, **Path**, **Cluster**, **Namespace**.
    - Enable **Auto-Sync** for automatic deployment.
9. **Test Auto-Sync Workflow**
    - Change image tag in deployment.yaml → commit & push → ArgoCD updates cluster automatically.
10. **Verify Deployment**
    - Check pods and service with kubectl get pods, kubectl get svc.
    - Access app via NodePort or LoadBalancer.

Conclusion

This project successfully implements a **GitOps pipeline**:

- All cluster changes are managed via **Git commits**.
- **ArgoCD auto-syncs** the cluster, ensuring the desired state is always applied.
- Provides **auditability, automation, and reproducibility** for Kubernetes application deployments.