**Project Documentation: Automated GitOps Deployment with Jenkins, Docker, Kubernetes & Argo CD**

**1. Project Overview**

This project implements a **fully automated CI/CD pipeline** using **Jenkins, Docker, Kubernetes, and Argo CD** to streamline the deployment process. Changes pushed to GitHub trigger an end-to-end deployment, ensuring that both application code and infrastructure updates are applied seamlessly.

**2. Tech Stack**

* **Version Control**: GitHub
* **CI/CD Automation**: Jenkins
* **Containerization**: Docker
* **Orchestration**: Kubernetes
* **GitOps Tool**: Argo CD
* **Registry**: Docker Hub

**3. CI/CD Pipeline Flow**

1. **Code Commit**: A developer pushes changes (application code or manifest files) to GitHub.
2. **Jenkins Job Triggered**: Jenkins detects the commit and executes the pipeline.
3. **CI Process**:
   * Clones the repository.
   * Builds the Docker image.
   * Pushes the image to Docker Hub.
4. **CD Process**:
   * Kubernetes manifest files are configured to always pull the latest Docker image.
   * Argo CD automatically syncs with the repository and deploys the latest changes.

**4. Deployment Workflow**

* **Jenkins** builds and pushes the updated Docker image.
* **Kubernetes Deployment Manifest** ensures pods always pull the latest image.
* **Argo CD** continuously monitors the GitHub repository and applies any changes in the manifest files.
* Updates to either the **application code** or **Kubernetes manifests** trigger automatic deployment.

**5. Infrastructure Details**

* **Kubernetes Cluster**: Manages application pods and services.
* **ClusterIP Service**: Internal service communication.
* **LoadBalancer (if applicable)**: External access to the application.
* **Argo CD Configuration**: Ensures GitHub is the **single source of truth**.

**6. Setting Up & Usage**

**6.1. Prerequisites**

* Kubernetes Cluster set up (Minikube, EKS, AKS, or GKE)
* Docker installed
* Jenkins installed and configured
* Argo CD installed on the cluster

**6.2. Steps to Deploy**

1. Clone the repository.
2. Configure Jenkins with the GitHub webhook.
3. Ensure Kubernetes manifest files are set to **pull the latest Docker image**.
4. Deploy Argo CD and sync it with the repository.
5. Commit changes to GitHub to trigger the pipeline.

**7. Troubleshooting & Best Practices**

**Common Issues & Fixes**

* **Jenkins build fails**: Check Docker build logs and ensure credentials for Docker Hub are correct.
* **Pods not updating**: Verify image pull policy is set to Always in the Kubernetes deployment.
* **Argo CD sync issues**: Check if Argo CD is pointing to the correct branch and repository.

**Best Practices**

* Keep all configurations **declarative** in GitHub.
* Use **branch protection rules** to prevent breaking changes.
* Regularly **monitor Argo CD sync status** to detect drift.
* Implement **logging and monitoring** for Jenkins and Kubernetes.

**8. Conclusion**

This setup enables a fully automated deployment workflow, reducing manual intervention and ensuring a **GitOps-driven approach**. Any updates in the GitHub repository seamlessly propagate to the live environment, making deployments **fast, consistent, and reliable**.

**🚀 Future Enhancements**

* Implement **Helm for better Kubernetes management**.
* Introduce **canary deployments with Argo Rollouts**.
* Add **security scanning for Docker images** in the pipeline.

This document provides a structured and professional overview of the project, ensuring clarity for team members, new joiners, and stakeholders.