**PROJECT REPORT - 1**

**Objective: Implemented GitOps Workflow with ArgoCD on Kubernetes**

Implemented a production-ready GitOps Pipeline using ArgoCD and Kubernetes, demonstrating modern day DevOps practices for continuous deployment

Project Repository: <https://github.com/3lton007/GitOps-Workflow-using-ArgoCD-on-Kubernetes>

**Tools Used**

* **Container Orchestration:** Kubernetes (Minikube)
* **GitOps Controller:** ArgoCD
* **Version Control:** GitHub repository
* **Application Runtime:** Docker Engine
* **Configuration Management:** Kustomize

**Implementation:**

Source code is defined in git with complete tracking and rollback commit history. ArgoCD continuously monitors repository and infrastructure cluster state. Automatically self-heals by checking drift detection. Developers commit code to git repository, ArgoCD detects commit changes and automatically validated the deployment changes in the Kubernetes cluster. Realtime monitoring and health status reporting through Argocd dashboard. Rollback capabilities through git commit tracking sync. Created Demo App for ArgoCD Deployment on Kubernetes. Pushed The app code deployment changes to Git to Auto Sync with argoCD. Synced with the latest commit. Monitored multi-enviroment application setup with health checks using Prometheus.

**Process Workflow:**

1) Installed Minikube and ArgoCD, and implemented a Demo App that provides yaml script to initiate GitOps Pipeline using ArgoCD UI.

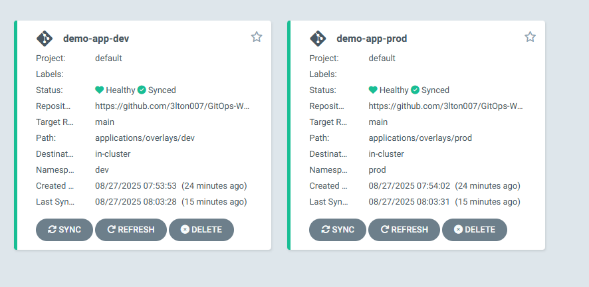
2) Pushed the code changes to Git to trigger ArgoCD. ArgoCD then creates a sync commit to track changes to the git repository. Created multi-enviroment setup using kustomize for dev and prod environment.

3) Kubernetes applies the configuration and updates the pods with new configuration.

4) Monitored using Prometheus to understand Observability.

**Analysis and Insights:**Understand the core declarative concepts of GitOps feature to handle infrastructure life management of Kubernetes deployments. How the synchronize keeps tracks of git commits to avoid state drifts and constant observability monitoring using health checks. Changes are deployed automatically to Kubernetes. Both Complete deployment history and process to revert back to previous history commits. Also, using kustomize to deploy multi-enviroment cluster to handle multiple infrastructure environments and handle cpu and memory usage of the pods.  **Screenshots:**

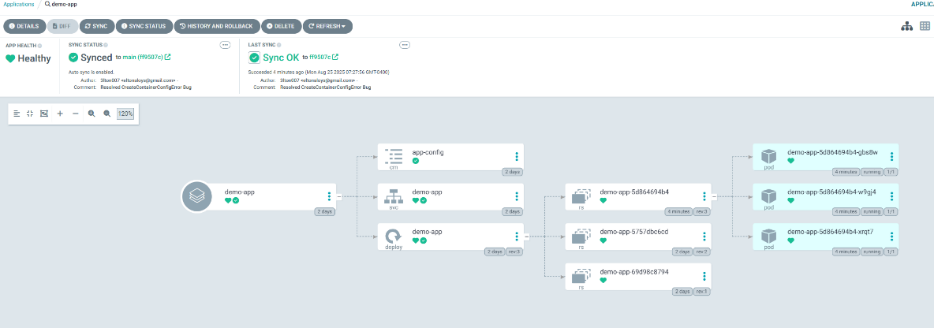
**Multi-Enviroment Setup**

****

**Using Prometheus for ArgoCD Observability**

****

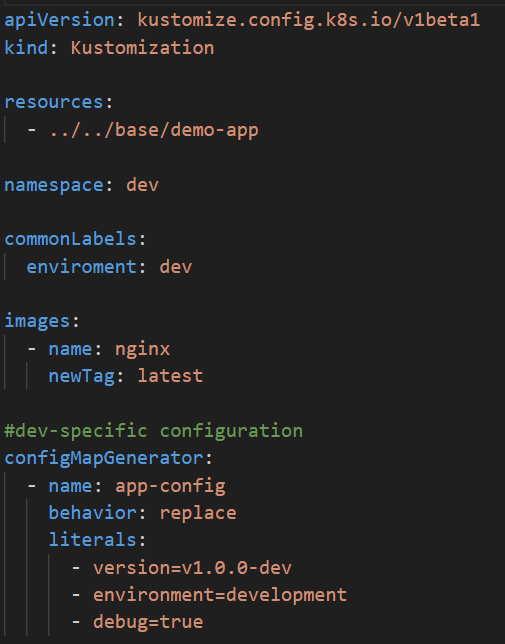
**ArgoCD UI**

****

**Deployment.yaml**

****

**Kustomize.yaml**

****

**Conclusion:** Successfully implemented GitOps by syncing Kubernetes deployment states directly from a git Repository.