Part 3: CI/CD Testing and Validation (Full CI-CD automation)

Submitted by: Marion Queen Ramos 0401476

November 24, 2025

COMP_4001 Developing in a DevOps Environment

Part 3: CI/CD Testing and Validation (Full CI-CD automation)

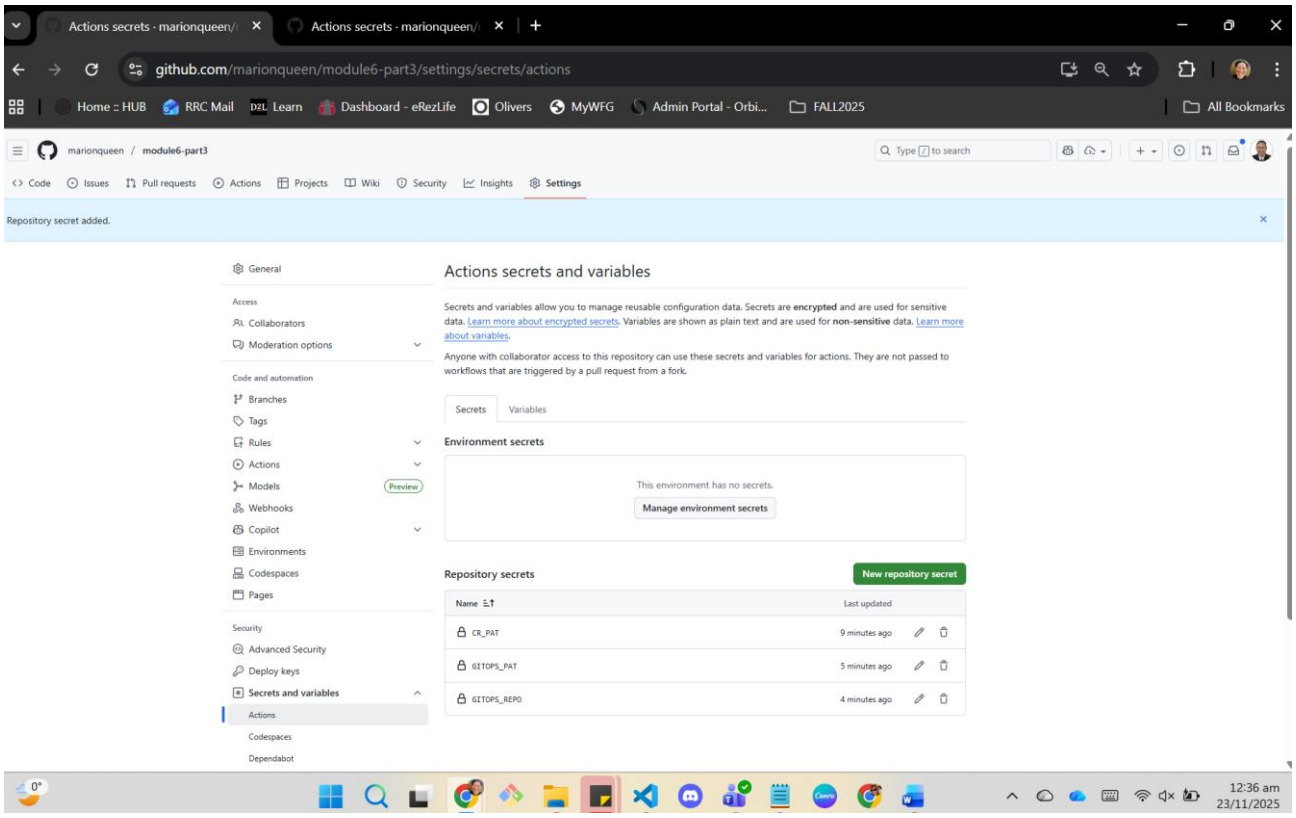
Deploy the application using Kubernetes and automate the entire CI-CD using github actions and argocd pipeline such that any code changes can be integrated, tested and deployment artifacts should be automatically generated and synchronized.

The goal is to create a system where a simple git push to your repository automatically triggers a build, a test, and a deployment to Kubernetes, all without manual intervention.

Screenshots

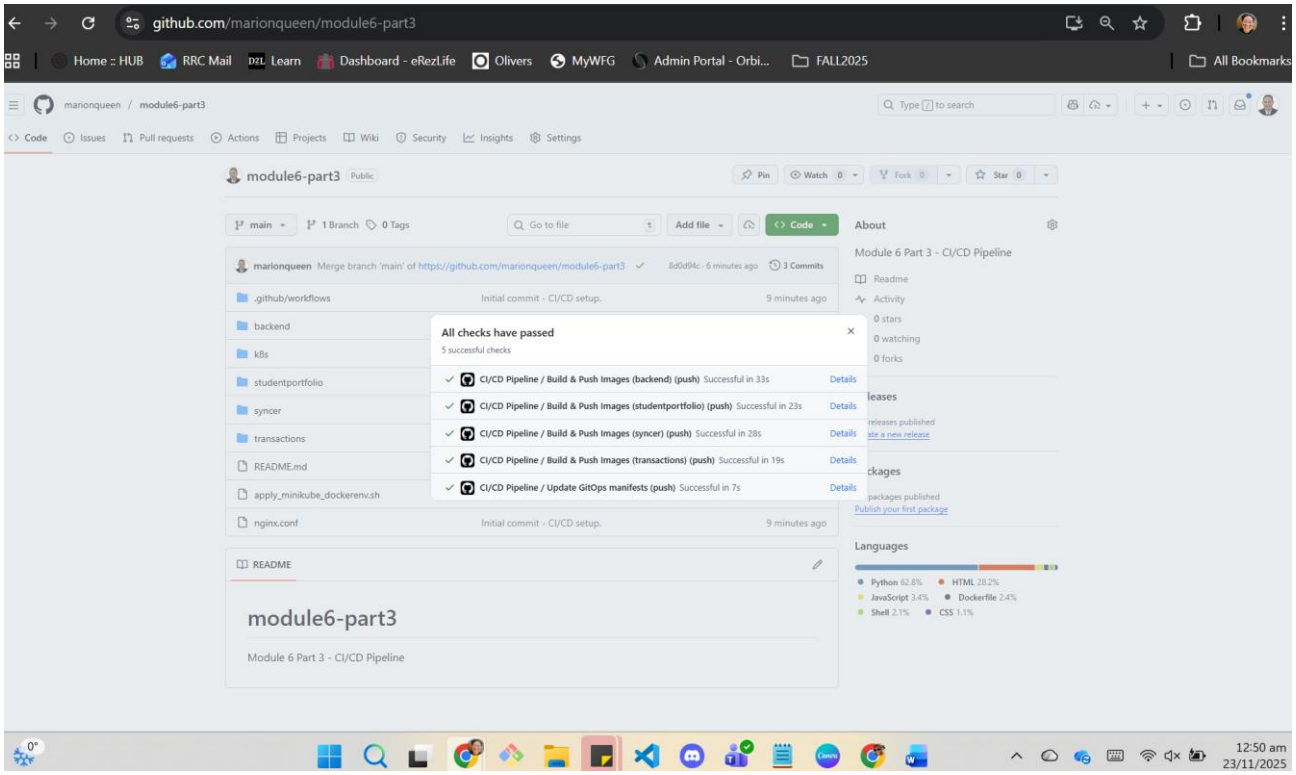
Screenshot 009

Secrets page should showing 3 secrets: CR_PAT, GITOPS_PAT, GITOPS_REPO



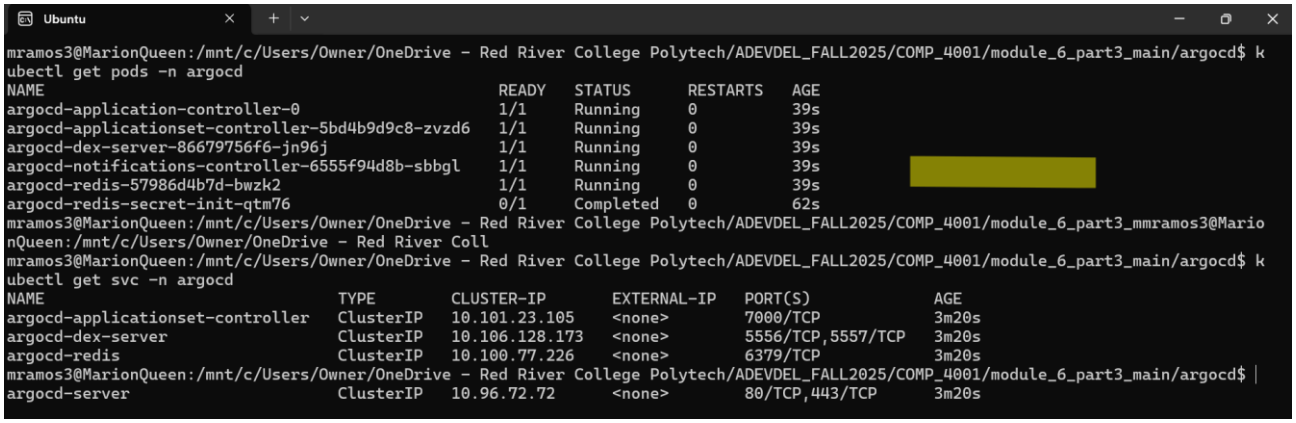
Screenshot 010

GitHub Actions Running Successfully (green checkmark)



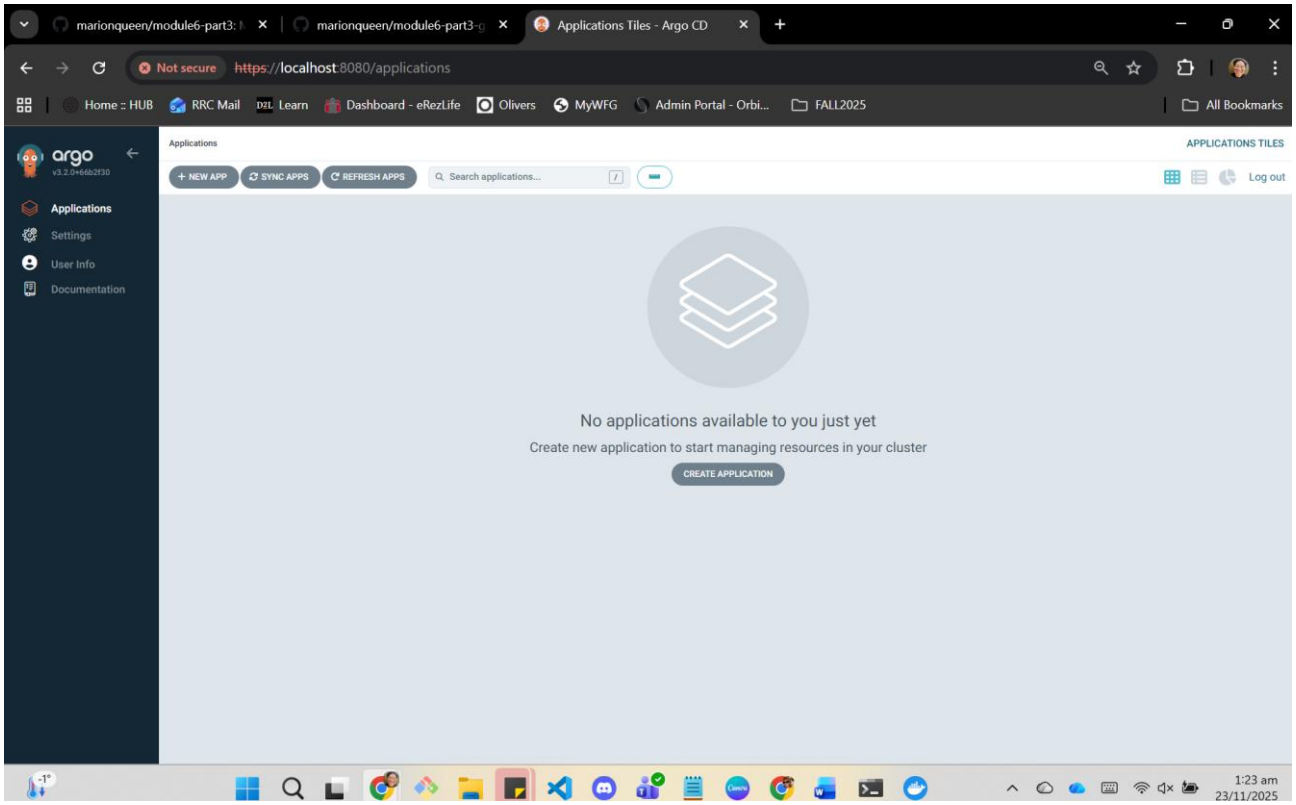
Screenshot Manual 011

ArgoCD Services list shown



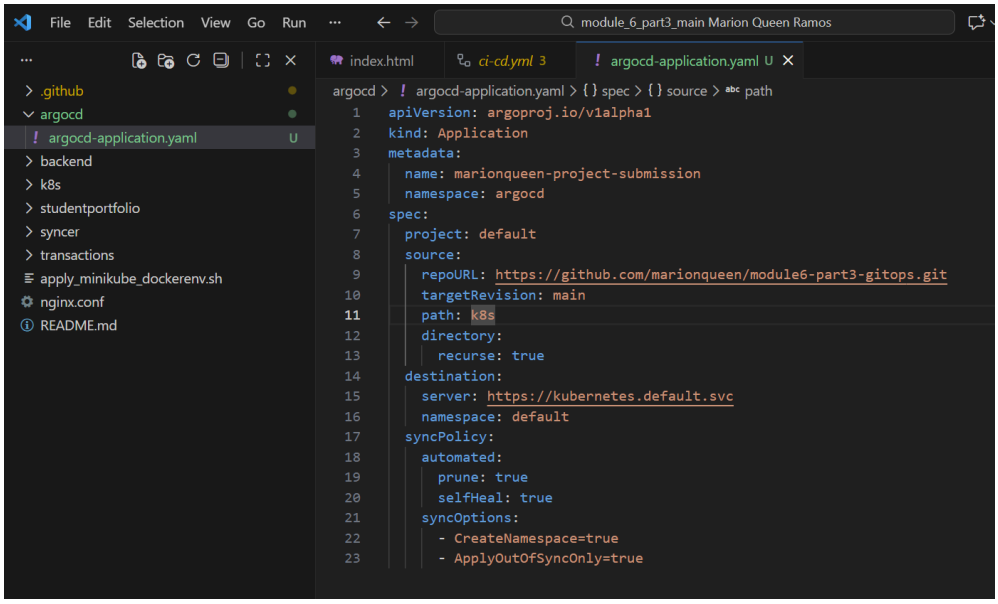
Screenshot Manual 012

ArgoCD Dashboard After Login



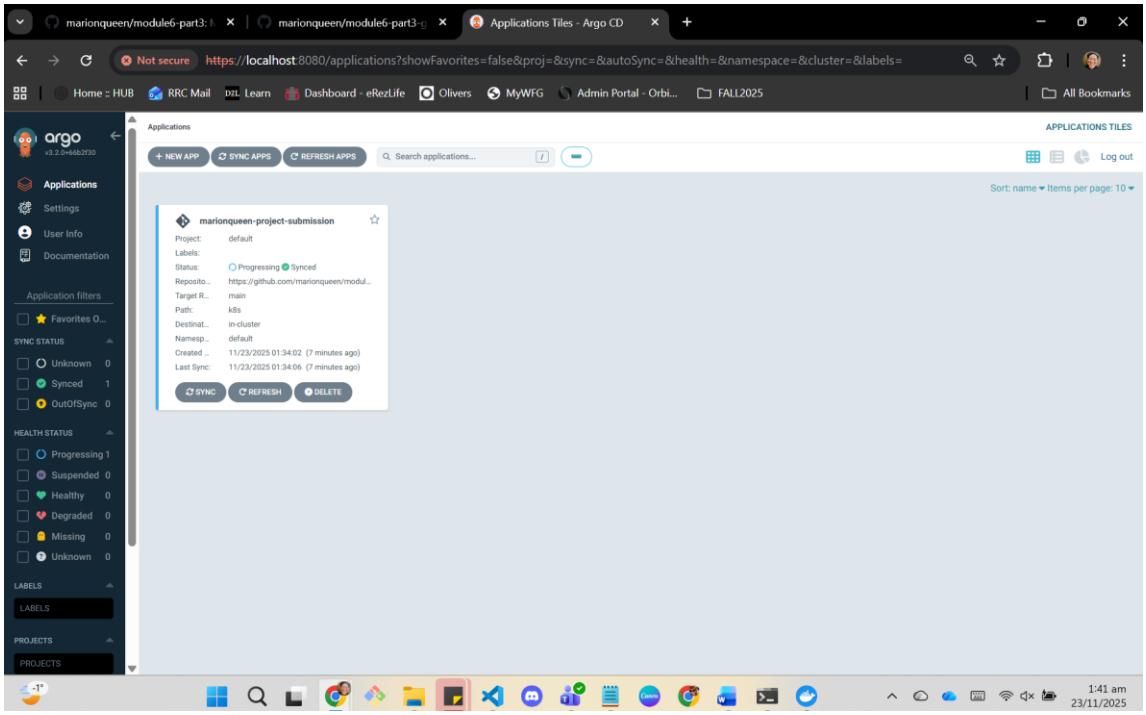
Screenshot Manual 013

argocd-application.yaml File



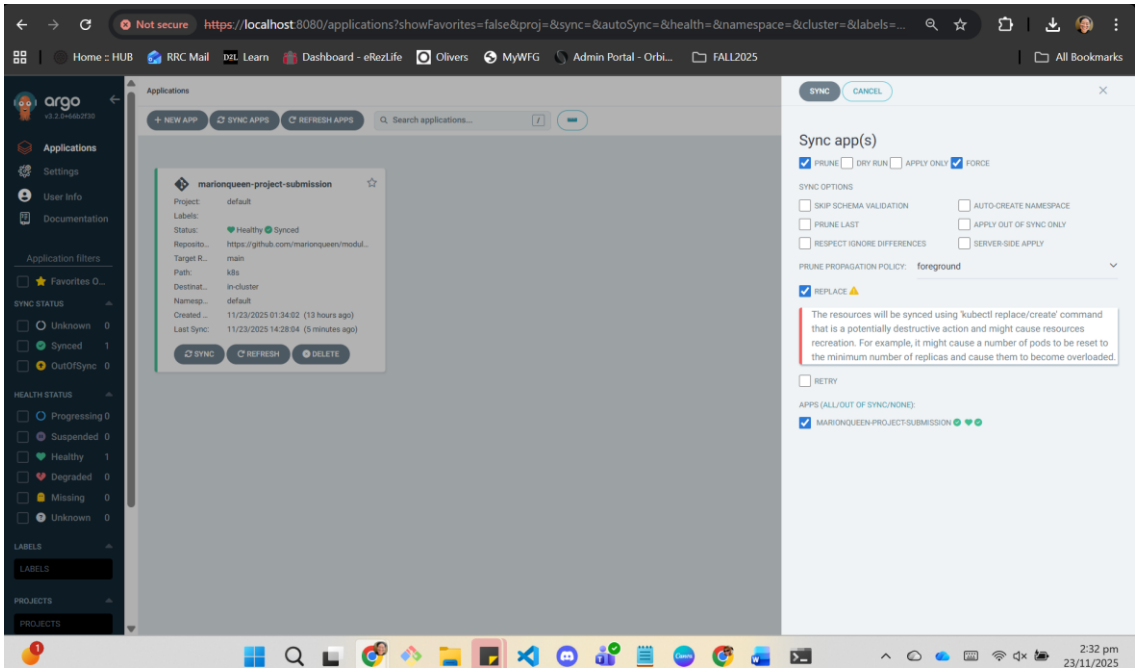
Screenshot Manual 014

ArgoCD Application Visible



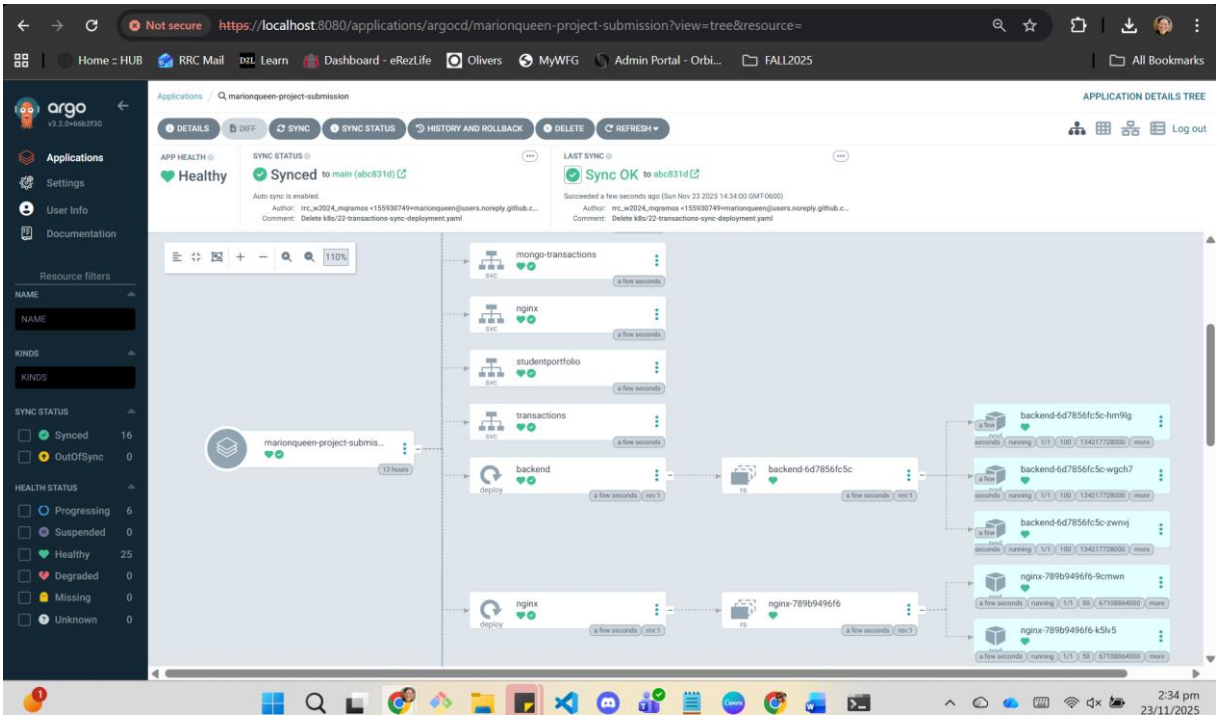
Screenshot Manual 015

Sync Options Selected



Screenshot Manual 016

All Pods Healthy (Green/Synced)



Question 1: Drift Detection

I manually changed the backend deployment replicas from 3 to 5 using kubectl edit deployment backend. Within 1-2 minutes, ArgoCD detected this difference and marked the application as "OutOfSync" (yellow status). The UI showed Git specified 3 replicas while the cluster had 5. Because auto-sync was enabled, ArgoCD automatically reverted my change back to 3 replicas, returning the status to "Synced" (green).

This demonstrates ArgoCD enforcing Git as the single source of truth. Even manual cluster changes get automatically corrected to match Git, preventing configuration drift. All changes must go through Git, which provides version control and audit trails rather than undocumented kubectl commands.

Question 2: History and Rollback

I created three versions by modifying studentportfolio/index.html three times, pushing each to GitHub and waiting for CI/CD deployment. In ArgoCD's "History and Rollback" section, all three commits appeared with timestamps and messages. I selected the oldest version and clicked "Rollback."

ArgoCD immediately redeployed the old version. Within 1-2 minutes, Kubernetes performed a rolling update and pods restarted with the original image. The browser showed Version 1 content was restored without rebuilding images—ArgoCD simply reapplied old manifests pointing to existing container registry tags.

ArgoCD rollback is safer and faster because: (1) Git history shows exactly what you're rolling back to, (2) it's instant using existing images, (3) it's auditable through Git, (4) no manual editing or remembering image tags, and (5) automatic zero-downtime rolling updates.

Question 3: Purpose of Auto-Sync

Auto-sync (syncPolicy.automated) automatically keeps Kubernetes synchronized with Git. Without it, I'd manually click "Sync" after every code push, defeating continuous deployment.

With auto-sync, my pipeline is fully automated: code push → GitHub Actions builds images → workflow updates GitOps repo → ArgoCD detects changes → automatic deployment. This takes 5-7 minutes with zero manual intervention.

Auto-sync includes self-healing, automatically reverting manual cluster changes to match Git. This eliminates the human bottleneck, enabling true continuous deployment rather than just continuous delivery.