

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

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

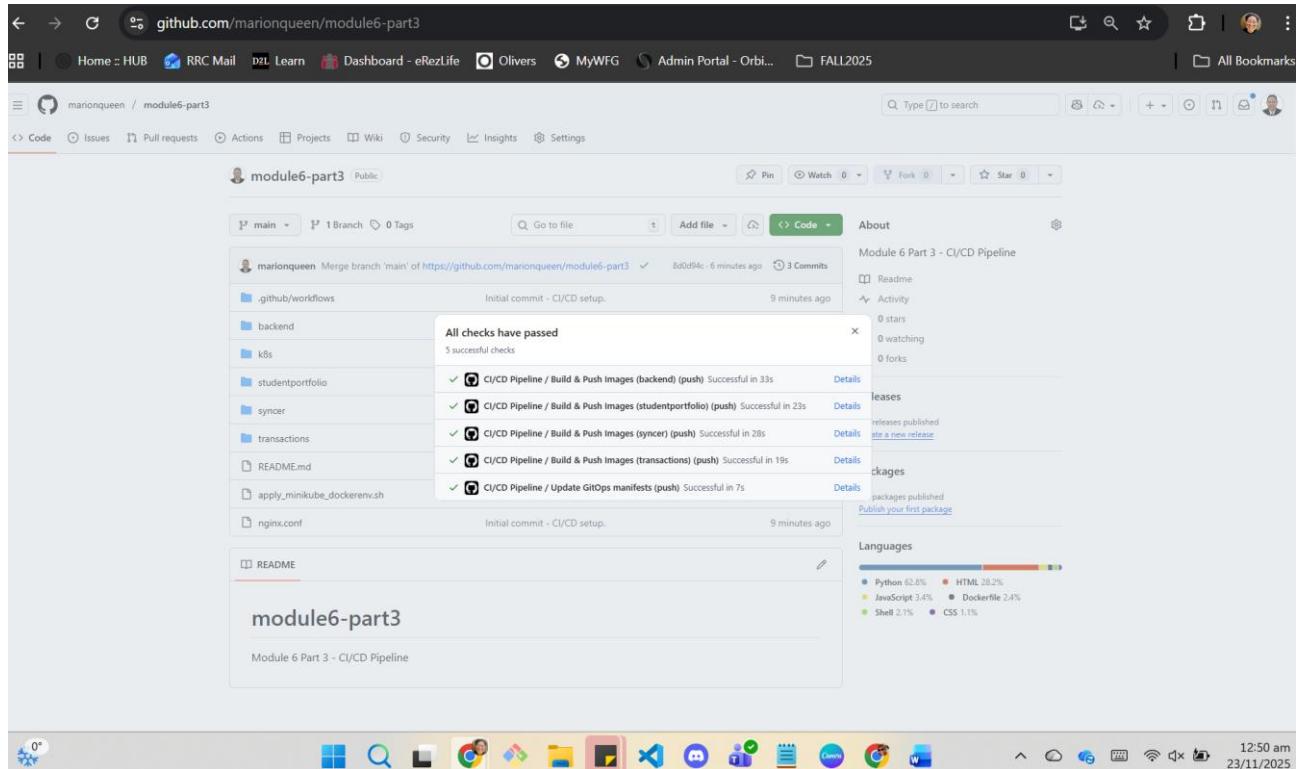
The screenshot shows a browser window with two tabs open, both titled "Actions secrets - marionqueen/...". The active tab is "github.com/marionqueen/module6-part3/settings/secrets/actions". The page displays the "Actions secrets and variables" section. On the left, there's a sidebar with various repository settings like General, Access, AI Collaborators, Moderation options, Code and automation, and Secrets and variables (which is currently selected). The main content area shows two sections: "Environment secrets" (empty) and "Repository secrets". Under "Repository secrets", there are three entries:

Name	Last updated
CR_PAT	9 minutes ago
GITOPS_PAT	5 minutes ago
GITOPS_REPO	4 minutes ago

A green "New repository secret" button is located at the top right of the "Repository secrets" table. The browser's address bar shows the URL "github.com/marionqueen/module6-part3/settings/secrets/actions". The taskbar at the bottom of the screen shows various pinned icons, and the system tray indicates the date and time as "23/11/2025 12:36 am".

Screenshot 010

GitHub Actions Running Successfully (green checkmark)



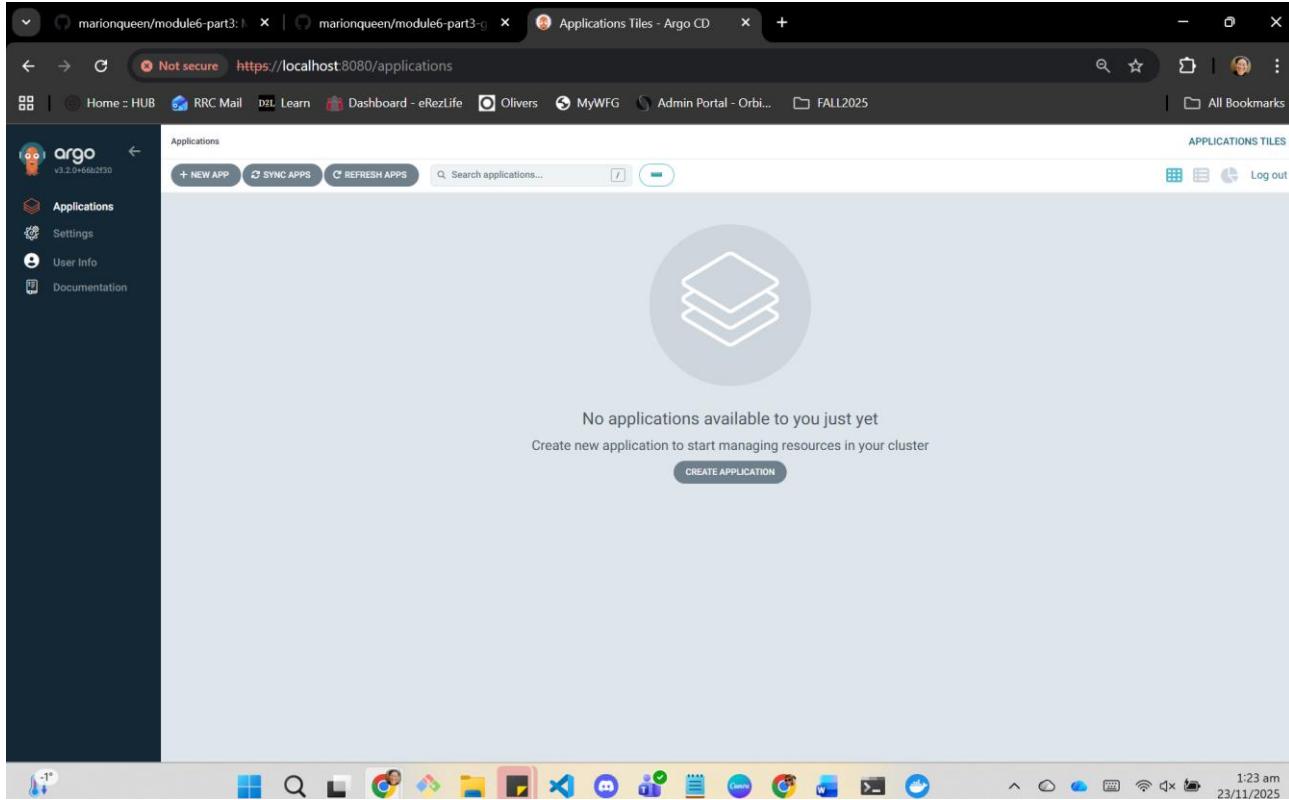
Screenshot Manual 011

ArgoCD Services list shown

```
mramos3@MarionQueen:/mnt/c/Users/Owner/OneDrive - Red River College Polytech/ADEVDEL_FALL2025/COMP_4001/module_6_part3_main/argocd$ kubectl get pods -n argocd
NAME                               READY   STATUS    RESTARTS   AGE
argocd-application-controller-0     1/1     Running   0          39s
argocd-applicationset-controller-5bd4b9d9c8-zvzd6   1/1     Running   0          39s
argocd-dex-server-86679756f6-jn96j   1/1     Running   0          39s
argocd-notifications-controller-6555f94d8b-sbbgl   1/1     Running   0          39s
argocd-redis-57986d4b7d-bwzk2      1/1     Running   0          39s
argocd-redis-secret-init-qtm76    0/1     Completed  0          62s
mramos3@MarionQueen:/mnt/c/Users/Owner/OneDrive - Red River College Polytech/ADEVDEL_FALL2025/COMP_4001/module_6_part3_mmrmos3@MarionQueen:/mnt/c/Users/Owner/OneDrive - Red River College Polytech/ADEVDEL_FALL2025/COMP_4001/module_6_part3_main/argocd$ kubectl get svc -n argocd
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)        AGE
argocd-applicationset-controller   ClusterIP  10.101.23.105  <none>        7000/TCP      3m20s
argocd-dex-server                 ClusterIP  10.106.128.173  <none>        5556/TCP,5557/TCP  3m20s
argocd-redis                      ClusterIP  10.100.77.226  <none>        6379/TCP      3m20s
mramos3@MarionQueen:/mnt/c/Users/Owner/OneDrive - Red River College Polytech/ADEVDEL_FALL2025/COMP_4001/module_6_part3_main/argocd$ | argocd-server                   ClusterIP  10.96.72.72    <none>        80/TCP,443/TCP   3m20s
```

Screenshot Manual 012

ArgoCD Dashboard After Login



Screenshot Manual 013

argocd-application.yaml File

```

index.html ci-cd.yaml argocd-application.yaml
... index.html ci-cd.yaml argocd-application.yaml
.argod .argocd .argocd-application.yaml
> .github > argocd ! argocd-application.yaml
> backend > studentportfolio > syncer > transactions
> k8s > apply_minikube_dockerenv.sh & README.md
> nginx.conf

```

```

apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: marionqueen-project-submission
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://github.com/marionqueen/module6-part3-gitops.git
    targetRevision: main
    path: k8s
    directory:
      recurse: true
  destination:
    server: https://kubernetes.default.svc
    namespace: default
  syncPolicy:
    automated:
      prune: true
      selfHeal: true
  syncOptions:
    - CreateNamespace=true
    - ApplyOutOfSyncOnly=true

```

Screenshot Manual 014

ArgoCD Application Visible

The screenshot shows the ArgoCD Applications Tiles interface. On the left, there is a sidebar with various filters and status counts. The main area displays a single application entry for "marionqueen-project-submission". The application details are as follows:

- Project:** default
- Labels:** Progressing Synced
- Status:** Progressing
- Repository:** https://github.com/marionqueen/module6-part3
- Target R.:** main
- Path:** k8s
- Destinat.:** in-cluster
- Namespace:** default
- Created:** 11/23/2025 01:34:02 (7 minutes ago)
- Last Sync:** 11/23/2025 01:34:02 (7 minutes ago)

At the bottom of the application card are three buttons: SYNC, REFRESH, and DELETE.

Screenshot Manual 015

Sync Options Selected

The screenshot shows the ArgoCD Applications Tiles interface with sync options selected for the "marionqueen-project-submission" application. A modal window titled "Sync app(s)" is open on the right side of the screen. In the "SYNC OPTIONS" section, the following checkboxes are checked:

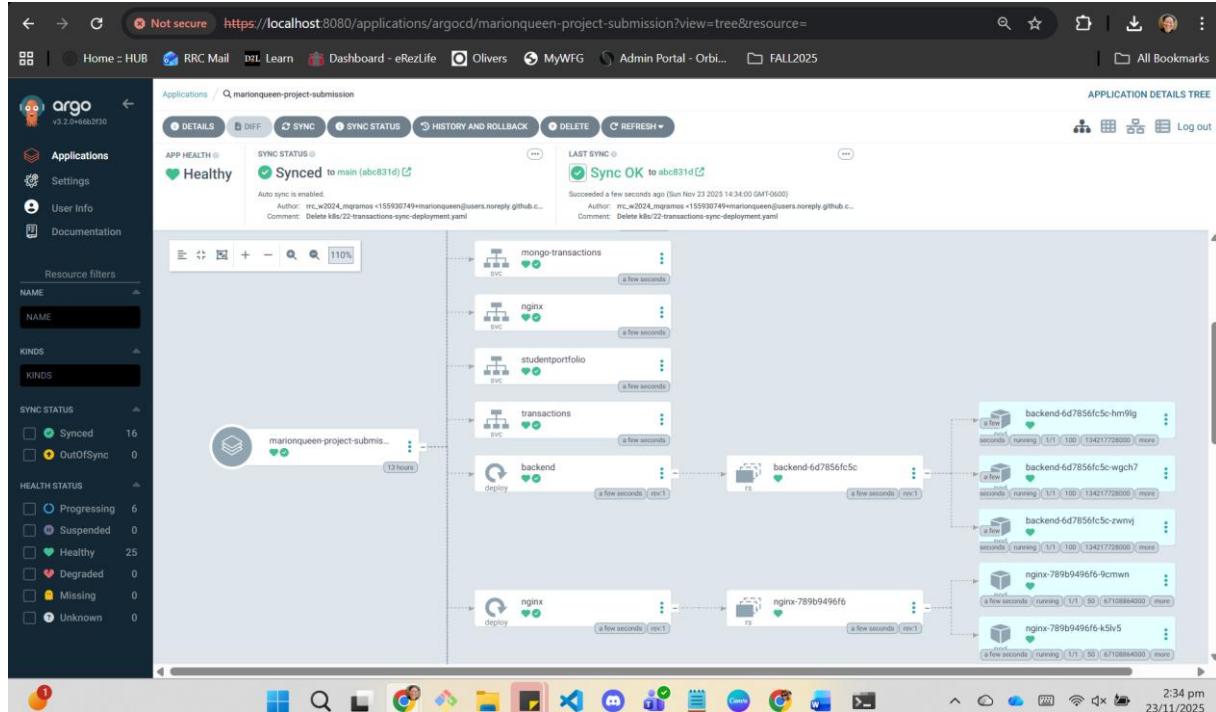
- PRUNE
- DRY RUN
- APPLY ONE
- FORCE

Below these options, there is a note: "The resources will be synced using 'kubectl replace/create' command that is a potentially destructive action and might cause resources recreation. For example, it might cause a number of pods to be reset to the minimum number of replicas and cause them to become overloaded."

At the bottom of the modal, there is a "RETRY" button and a "APPS (ALL/OUT OF SYNC/NONE)" section which contains a checked checkbox for "MARIONQUEEN-PROJECT-SUBMISSION".

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.