GitOps Workflow using ArgoCD on Kubernetes

Objective:

Implemented a GitOps workflow by syncing Kubernetes deployment states directly from a Git repository using ArgoCD. The goal is to automate application deployment and updates using Git as the single source of truth.

Tools Used

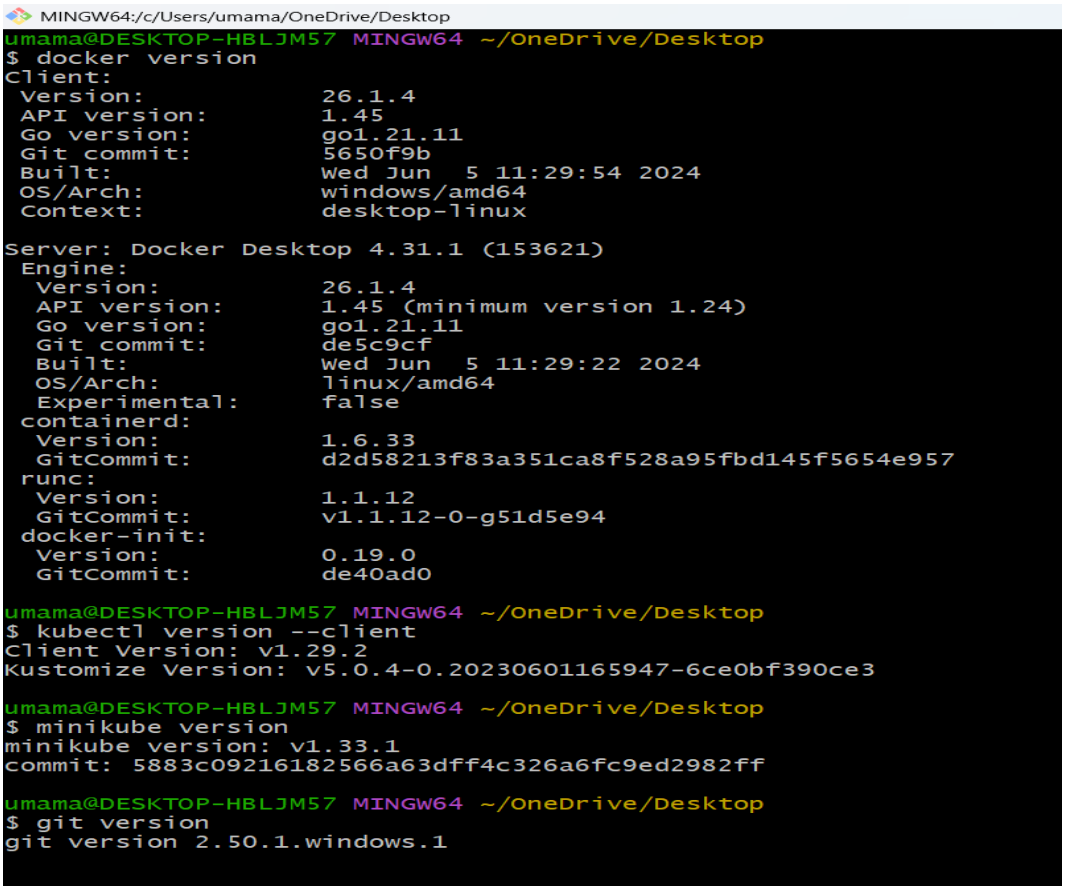
- Kubernetes: Minikube (local cluster)
- ArgoCD: GitOps continuous deployment tool
- GitHub: Source control for deployment manifests
- Docker: Containerization of applications

Prerequisites: Install and confirm these tools are available on our machine

1. Docker installed and running (or another Minikube driver).
2. kubectl installed and on your PATH.
3. minikube installed.
4. Git + GitHub account.

Verification

docker –version && kubectl version --client –short && minikube version && git –version



Starting local Kubernetes cluster (Minikube)

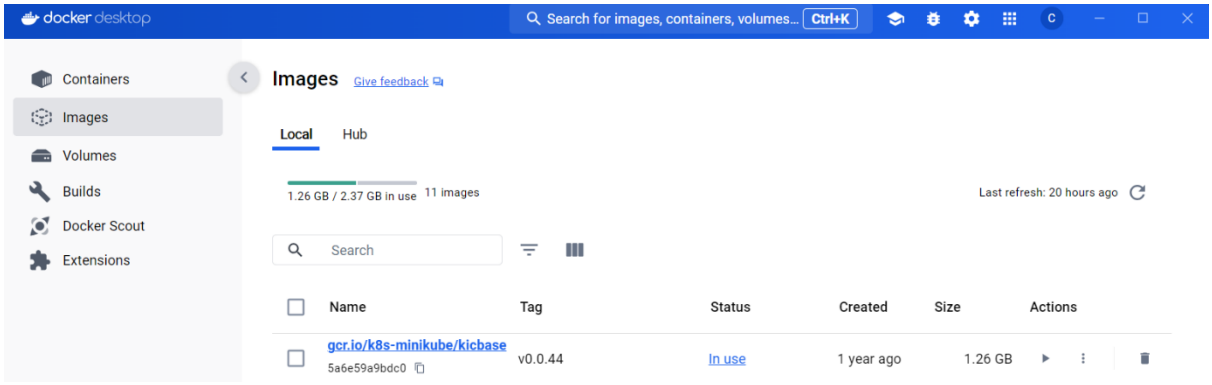
Commands: minikube start --driver=docker && kubectl get nodes

We can see status of minikube by using minikube status

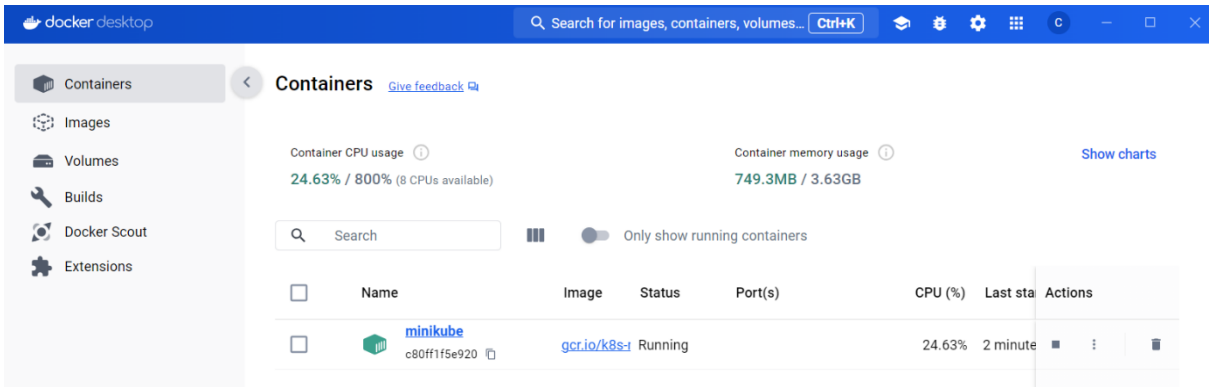
```
umama@DESKTOP-HBLJM57 MINGW64 ~/OneDrive/Desktop
$ minikube start --driver=docker
* minikube v1.33.1 on Microsoft Windows 11 Home Single Language 10.0.26100.4946 Build 26100.4946
* Using the docker driver based on user configuration
* Using Docker Desktop driver with root privileges
* Starting "minikube" primary control-plane node in "minikube" cluster
* Pulling base image v0.0.44 ...
* Creating docker container (CPUs=2, Memory=2200MB) ...
* Preparing Kubernetes v1.30.0 on Docker 26.1.1 ...
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Configuring bridge CNI (Container Networking Interface) ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: storage-provisioner, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default

umama@DESKTOP-HBLJM57 MINGW64 ~/OneDrive/Desktop
$ kubectl get nodes
NAME       STATUS   ROLES           AGE   VERSION
minikube   Ready    control-plane   28s   v1.30.0

umama@DESKTOP-HBLJM57 MINGW64 ~/OneDrive/Desktop
$ minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured


umama@DESKTOP-HBLJM57 MINGW64 ~/OneDrive/Desktop
$
```

We can see the minikube container is running in docker desktop

Image:



Container:

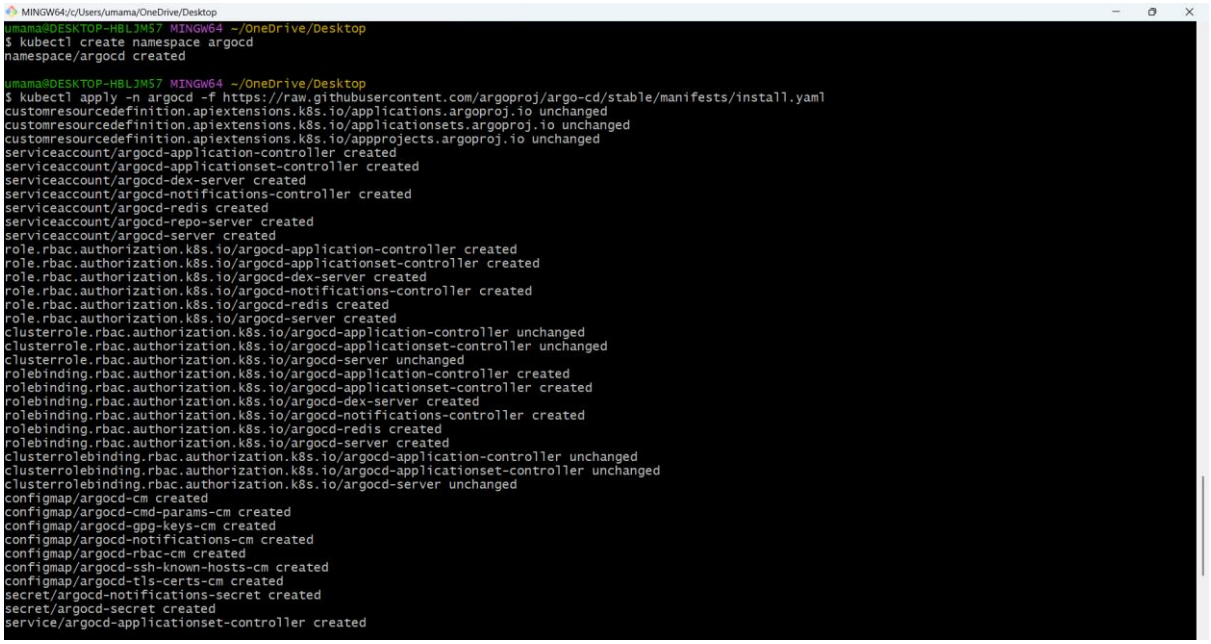


Install Argo CD into the cluster

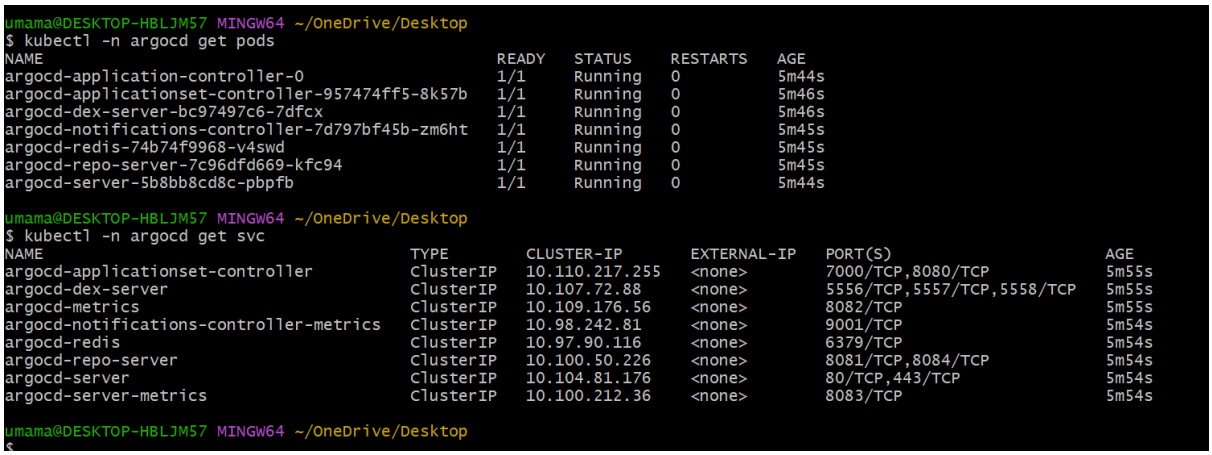
Create separate namespace: kubectl create namespace argocd

We create ArgoCD application by this command:

kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml



After applying check whether pods and services are running or not



Open the Argo CD UI (port-forward)

kubectl port-forward svc/argocd-server -n argocd 8080:443

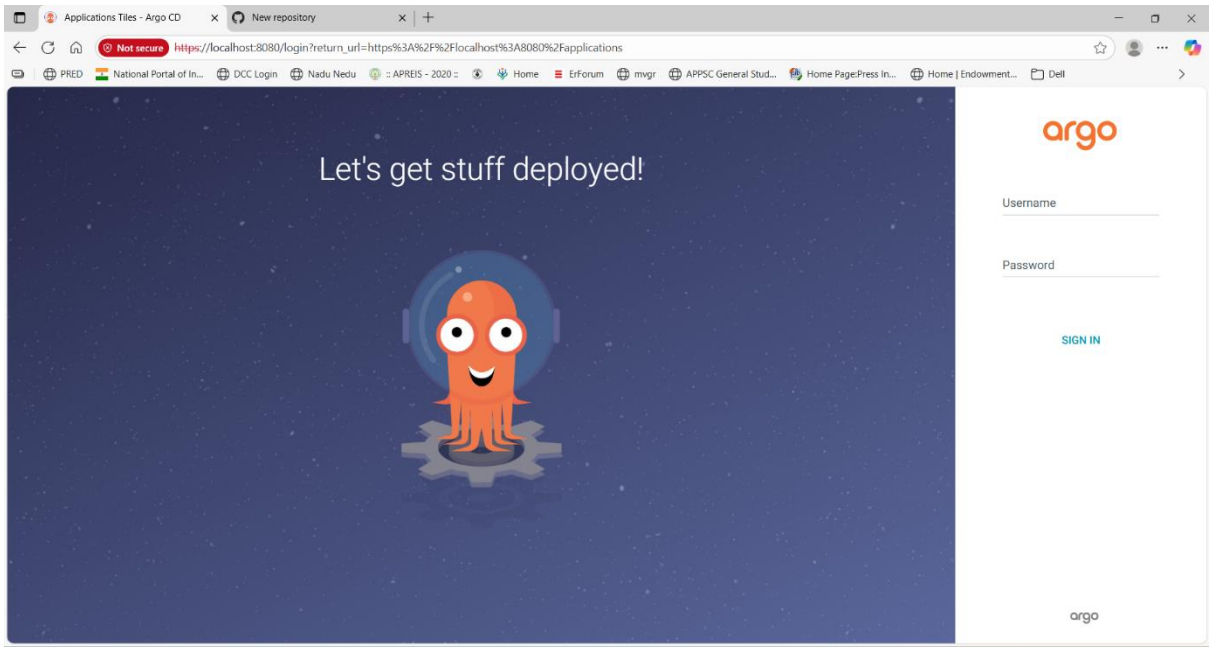
```
Windows PowerShell
PS C:\Users\umama\OneDrive\Desktop> minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured

PS C:\Users\umama\OneDrive\Desktop> kubectl port-forward svc/argocd-server -n argocd 8080:443
Forwarding from [::1]:8080 -> 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
```

Open browser: https://localhost:8080

we need to accept a TLS/self-signed warning — that's normal.

If port-forward runs successfully, we'll see it accept connections in the terminal.

In browser you should see the Argo CD login page.



Login to Argo CD (first time)

Default username: admin

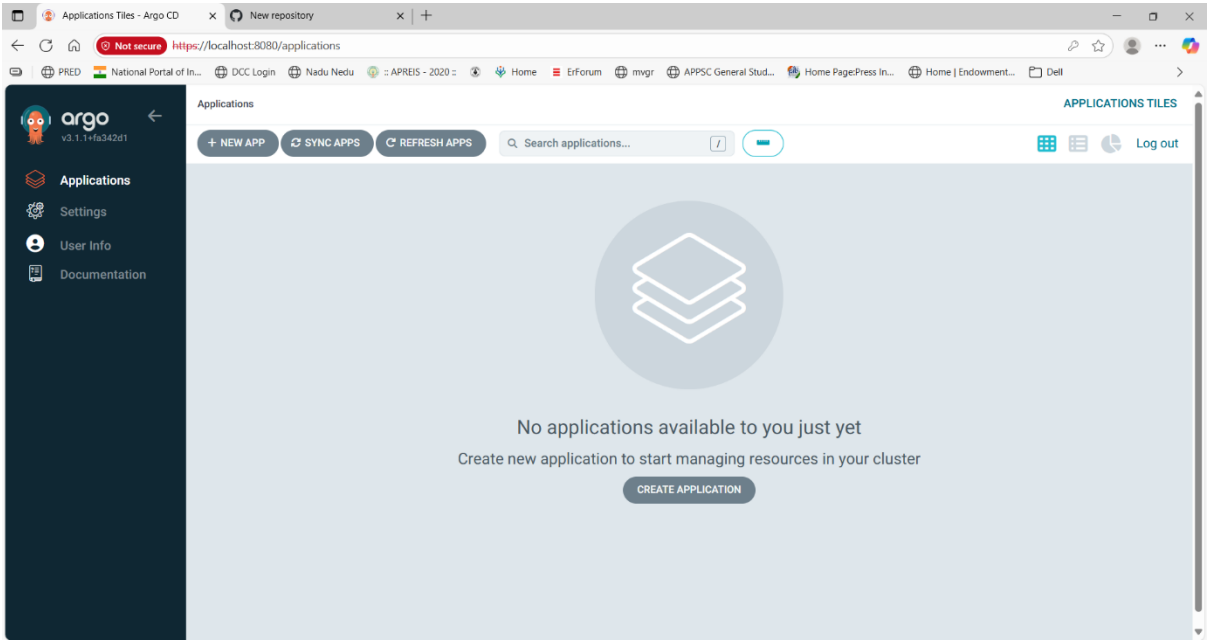
Get the initial password by running this command

kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.password}" | base64 -d && echo

```
umama@DESKTOP-HBLJM57 MINGW64 ~/OneDrive/Desktop
$ kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.password}" | base64 -d && echo
F6ZBAAUMV1jvdvlg

umama@DESKTOP-HBLJM57 MINGW64 ~/OneDrive/Desktop
$
```

After login we should see the Argo CD dashboard.

In UI top-left, click **Applications** — it should be empty for now.



Create the GitHub repository and manifest files

Locally create the folder and files (deployment.yaml and service.yaml)

Deployment.yaml:

```
umama@DESKTOP-HBLJM57 MINGW64 ~/OneDrive/Desktop/argocd-k8s (main)
$ cat deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-demo
  labels:
    app: nginx-demo
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-demo
  template:
    metadata:
      labels:
        app: nginx-demo
    spec:
      containers:
      - name: nginx
        image: nginx:stable
        ports:
        - containerPort: 80
```

The deployment.yaml defines a **Kubernetes Deployment** for running the Nginx application.

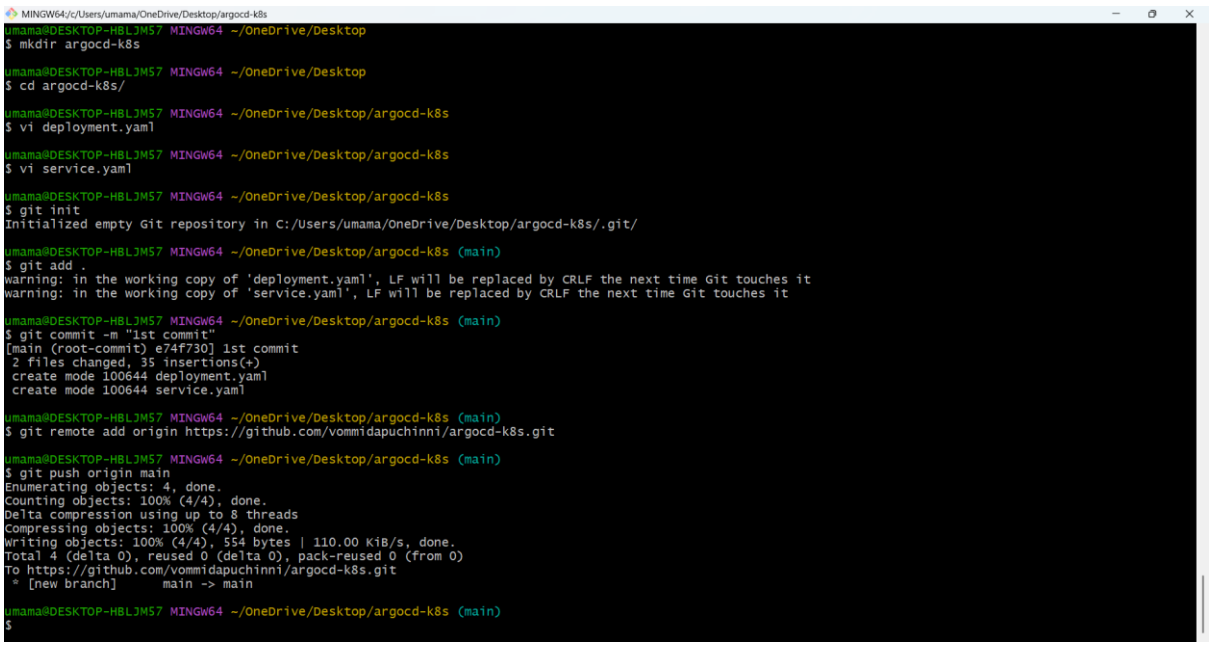
- **apiVersion, kind, metadata**: Specifies this is a Deployment named nginx-demo.
- **replicas: 1**: Runs only one Pod (single instance of Nginx).
- **selector & template labels**: Ensures the Deployment manages Pods with the label app: nginx-demo.
- **containers**: Defines one container named nginx using the **nginx:stable** image.
- **ports**: Exposes port **80** inside the container for HTTP traffic.

Service.yaml

```
umama@DESKTOP-HBLJM57 MINGW64 ~/OneDrive/Desktop/argocd-k8s (main)
$ cat service.yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-demo-svc
spec:
  selector:
    app: nginx-demo
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
  type: ClusterIP
```

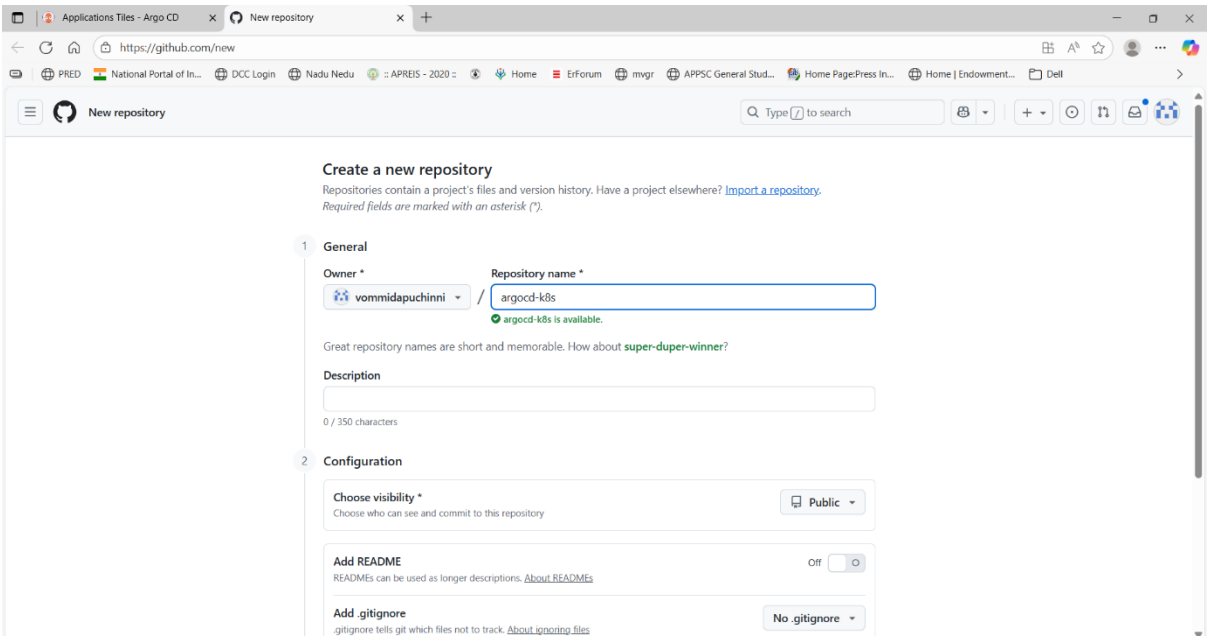
The service.yaml defines a **Kubernetes Service** to expose the Nginx Deployment inside the cluster.

- **apiVersion, kind, metadata**: Specifies this is a Service named nginx-demo-svc.
- **selector**: Targets Pods with the label app: nginx-demo (from the Deployment).
- **ports**: Maps **port 80** of the Service to **port 80** of the container for HTTP traffic.
- **type: ClusterIP**: Makes the Service accessible **only within the cluster** (not externally).

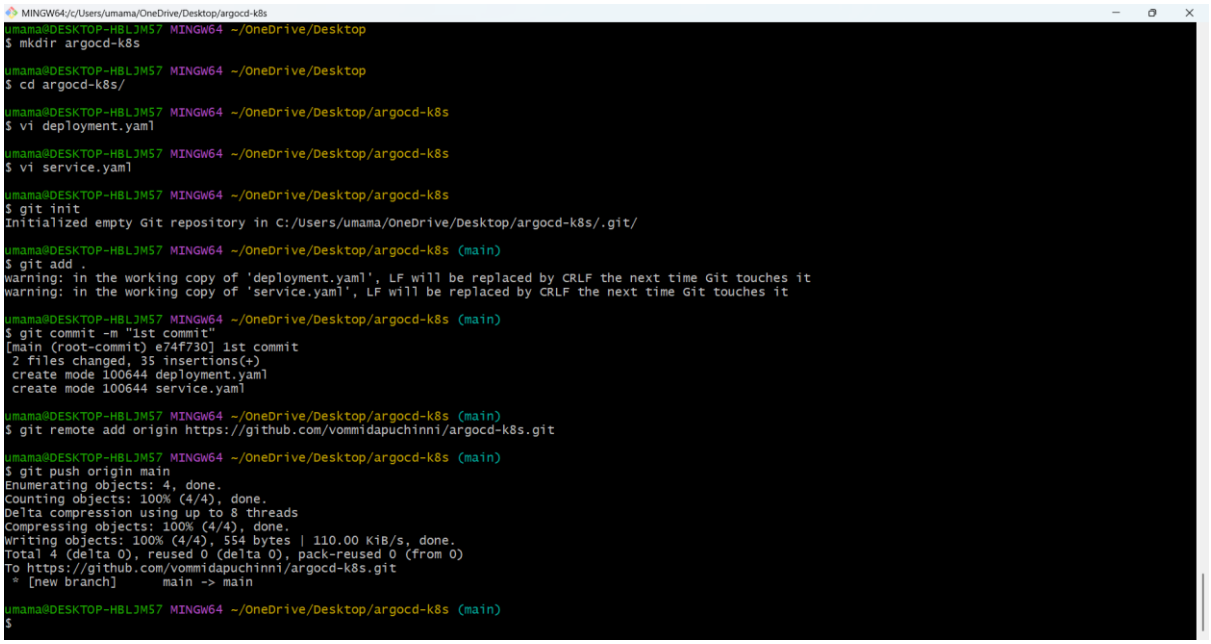


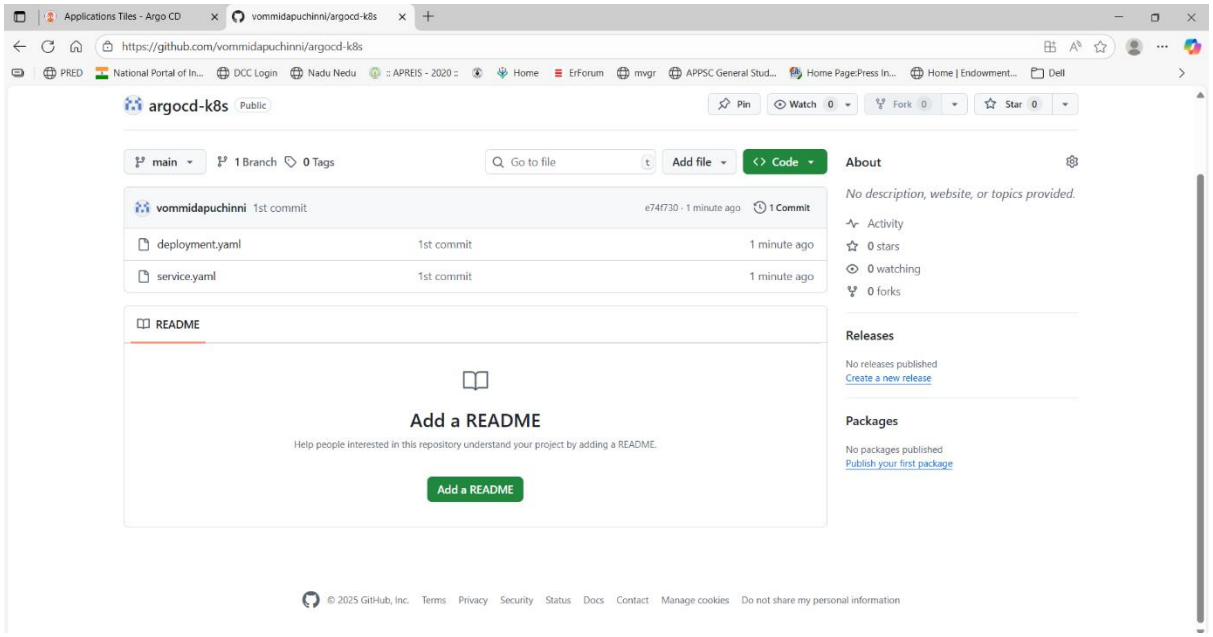
Open github in browser

Right click on + new repository gives name of the repo and give name and keep it public and with initiating readme file click created



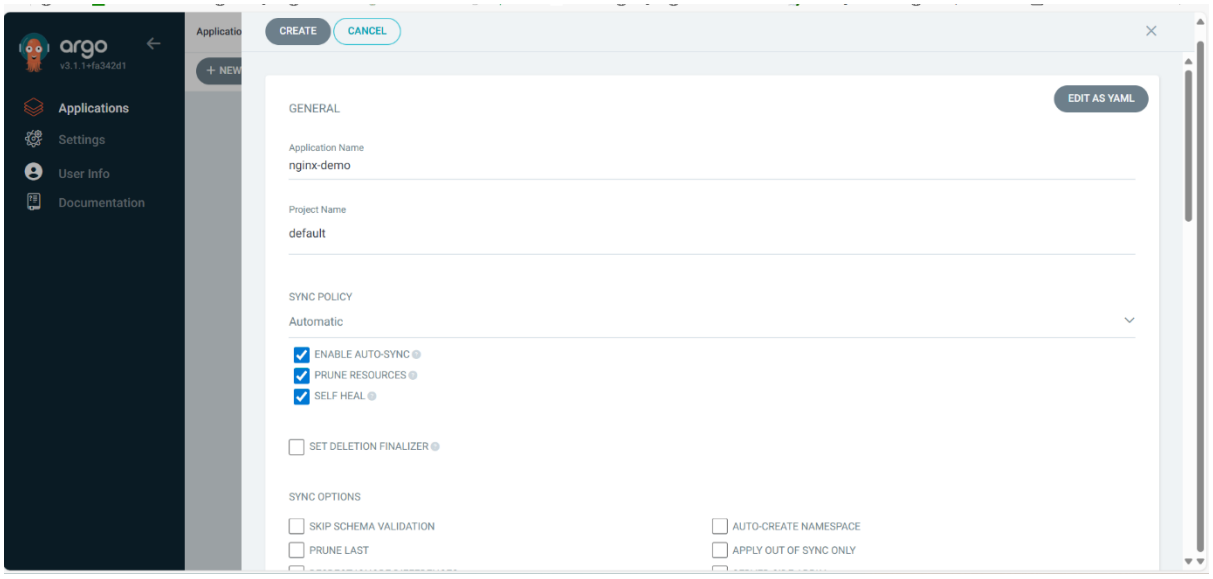
Push files to GitHub





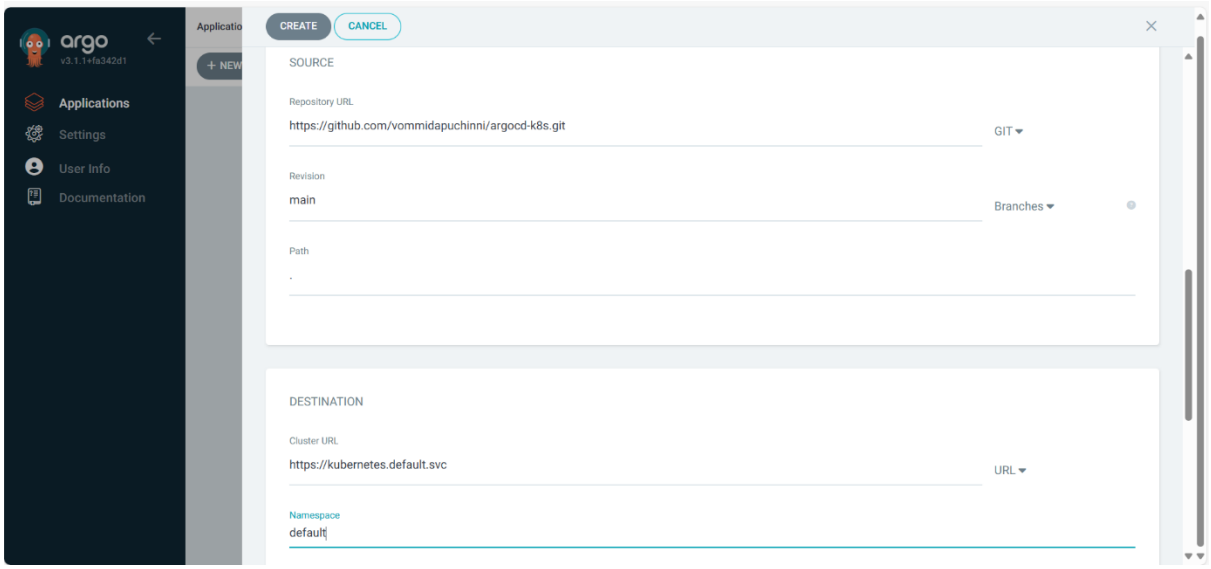
Create the Argo CD Application

- In Argo CD UI left sidebar click Applications.
- Click NEW APP (top-right).
- Fill the form (exact values):
- Application Name: nginx-demo
- Project: default
- Expand Sync Policy and choose Automatic (tick Prune and Self-Heal if shown).

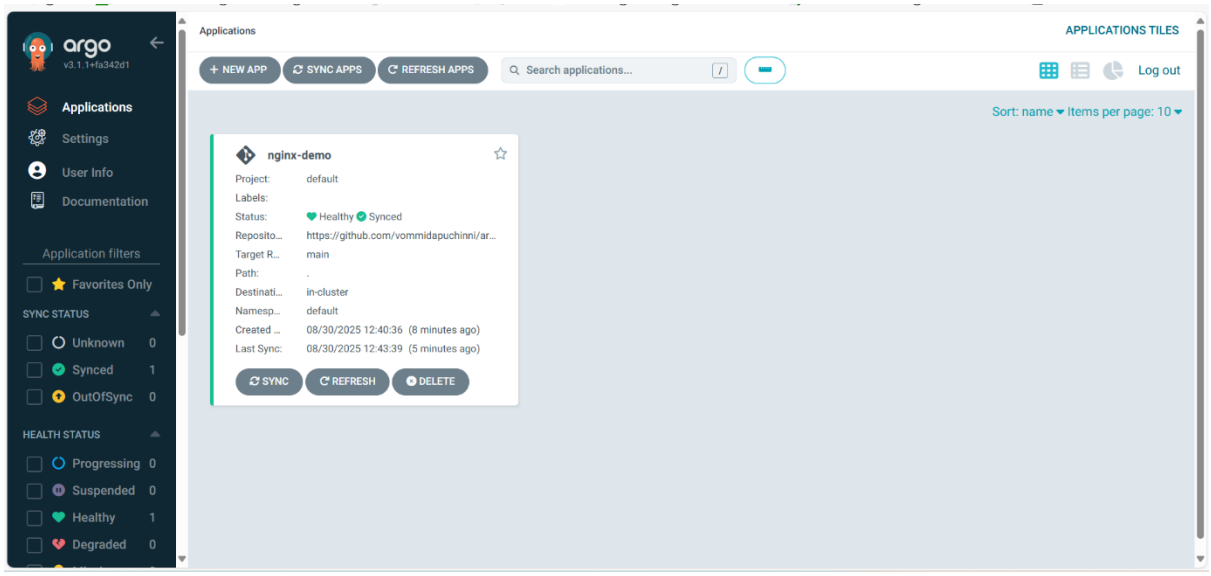


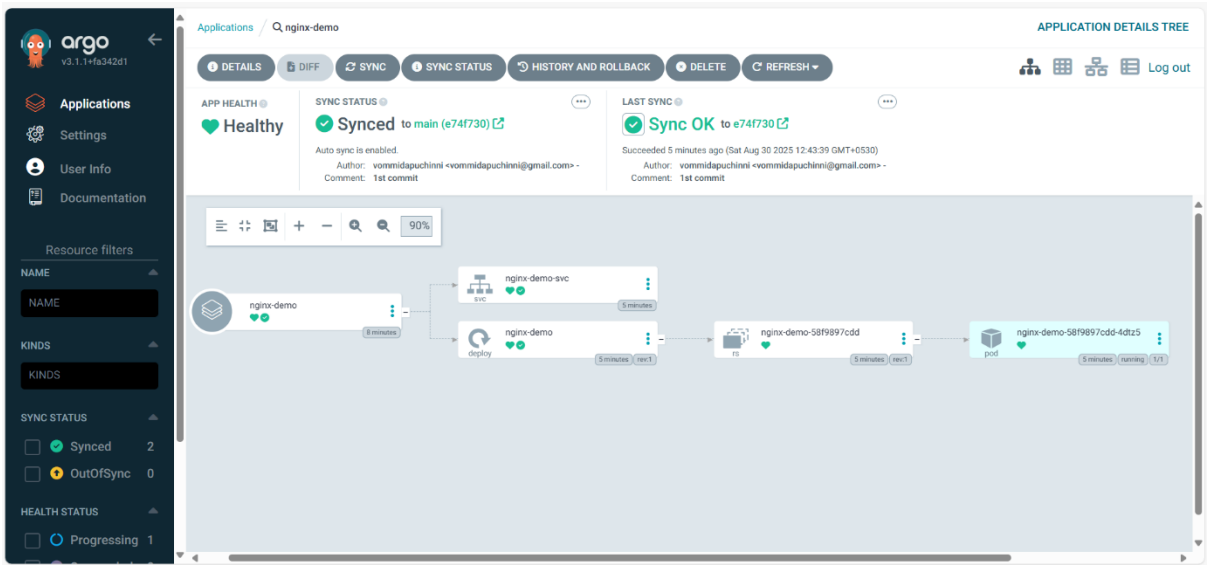
- Repository URL: https://github.com/vommidapuchinni/my-argocd-demo.git
- Revision: main
- Path: .

- Destination Cluster: https://kubernetes.default.svc
- Destination Namespace: default
- Click Create.



- In the Argo CD **Applications** list the nginx-demo app should appear.
- App status should move from OutOfSync → Synced (if automatic) and **Healthy**.
- In UI click the app → resource tree should show the Deployment and Service





Command-line verification:

```
umama@DESKTOP-HBLJM57 MINGW64 ~/OneDrive/Desktop/argocd-k8s (main)
$ kubectl get deploy nginx-demo -o wide
NAME         READY   UP-TO-DATE   AVAILABLE   AGE    CONTAINERS   IMAGES          SELECTOR
nginx-demo   1/1     1            1           6m9s   nginx        nginx:stable    app=nginx-demo

umama@DESKTOP-HBLJM57 MINGW64 ~/OneDrive/Desktop/argocd-k8s (main)
$ kubectl get svc nginx-demo-svc
NAME             TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE
nginx-demo-svc   ClusterIP   10.107.132.117   <none>        80/TCP    6m31s

umama@DESKTOP-HBLJM57 MINGW64 ~/OneDrive/Desktop/argocd-k8s (main)
$ kubectl get pod -l app=nginx-demo
NAME                          READY   STATUS    RESTARTS   AGE
nginx-demo-58f9897cdd-4dtz5   1/1     Running   0          6m49s

umama@DESKTOP-HBLJM57 MINGW64 ~/OneDrive/Desktop/argocd-k8s (main)
$
```

Test the GitOps flow (make a change in yaml files and push to github and watch Argo CD apply it)

Update the image version to show GitOps flow

- Changed nginx:stable → nginx:1.23

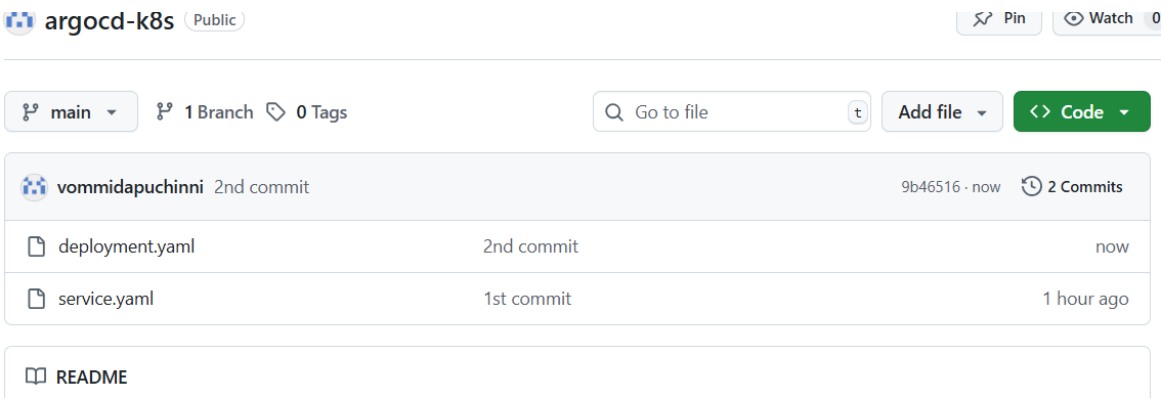
```
umama@DESKTOP-HBLJM57 MINGW64 ~/OneDrive/Desktop/argocd-k8s (main)
$ cat deployment.yaml
apiversion: apps/v1
kind: Deployment
metadata:
  name: nginx-demo
  labels:
    app: nginx-demo
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-demo
  template:
    metadata:
      labels:
        app: nginx-demo
    spec:
      containers:
      - name: nginx
        image: nginx:1.23
        ports:
        - containerPort: 80
```

Commit & push

git add deployment.yaml

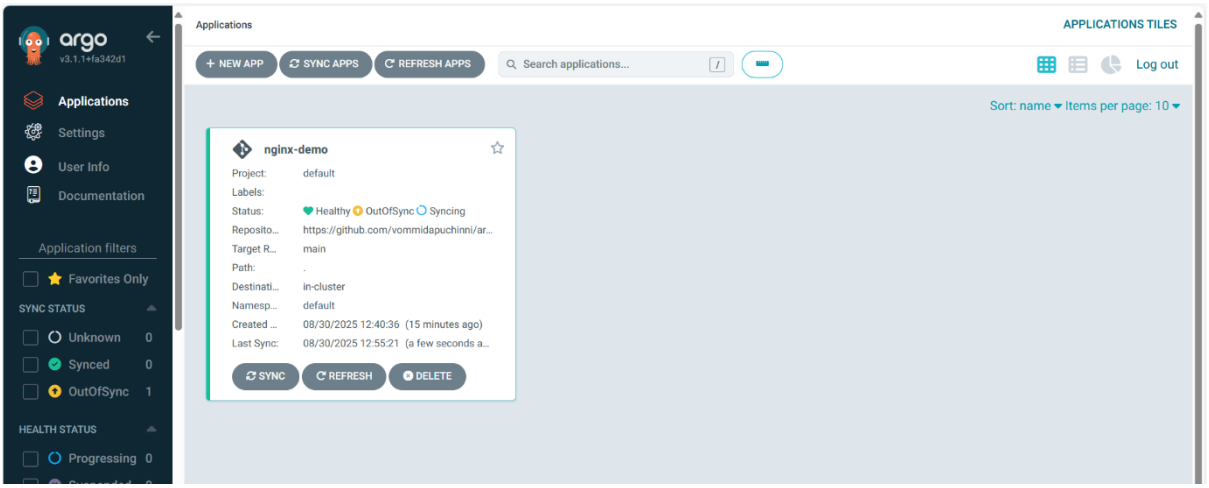
git commit -m "2nd commit"

git push origin main



ArgoCD automatically syncs (since Auto-Sync is enabled)

- UI: Shows OutOfSync → syncing → Synced & Healthy



Verify deployment

kubectl get pods -n default

kubectl describe pod <pod-name> -n default

```
umama@DESKTOP-HBLJM57 MINGW64 ~/OneDrive/Desktop/argocd-k8s (main)
$ kubectl get pods -n default
NAME                          READY   STATUS              RESTARTS   AGE
nginx-demo-58f9897cdd-4dtz5   1/1     Running             0          15m
nginx-demo-5ddc57794c-xrzpk   0/1     ContainerCreating   0          3m32s

umama@DESKTOP-HBLJM57 MINGW64 ~/OneDrive/Desktop/argocd-k8s (main)
$ kubectl get pods -n default
NAME                          READY   STATUS        RESTARTS   AGE
nginx-demo-58f9897cdd-4dtz5   1/1     Terminating   0          16m
nginx-demo-5ddc57794c-xrzpk   1/1     Running       0          4m54s

umama@DESKTOP-HBLJM57 MINGW64 ~/OneDrive/Desktop/argocd-k8s (main)
$ kubectl get pods -n default
NAME                          READY   STATUS    RESTARTS   AGE
nginx-demo-5ddc57794c-xrzpk   1/1     Running   0          5m4s
```

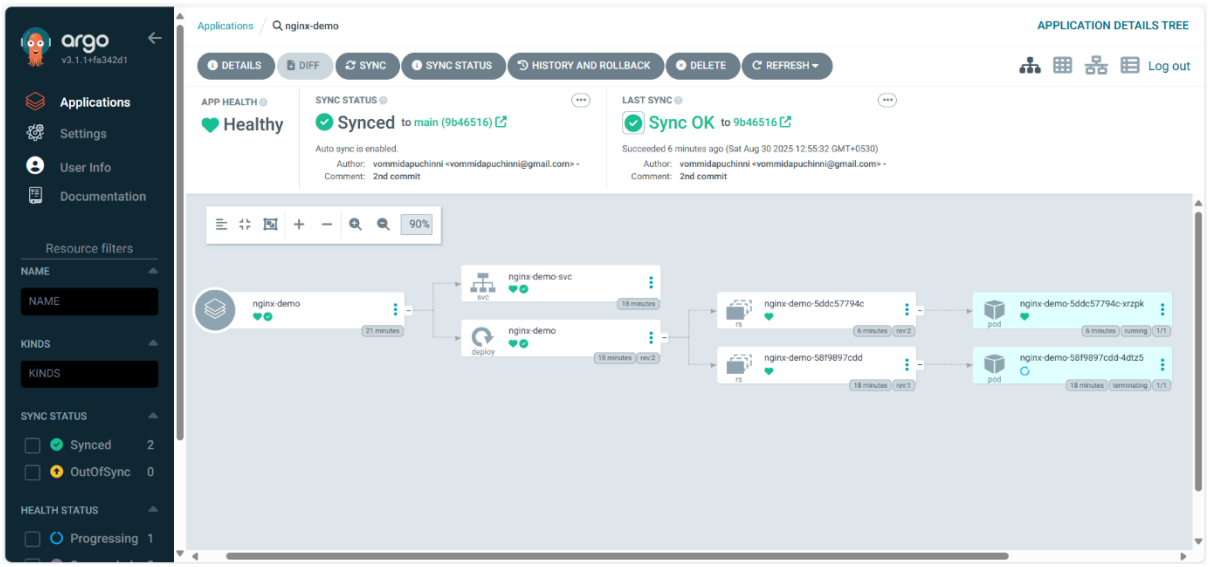
Pod should restart with new image.

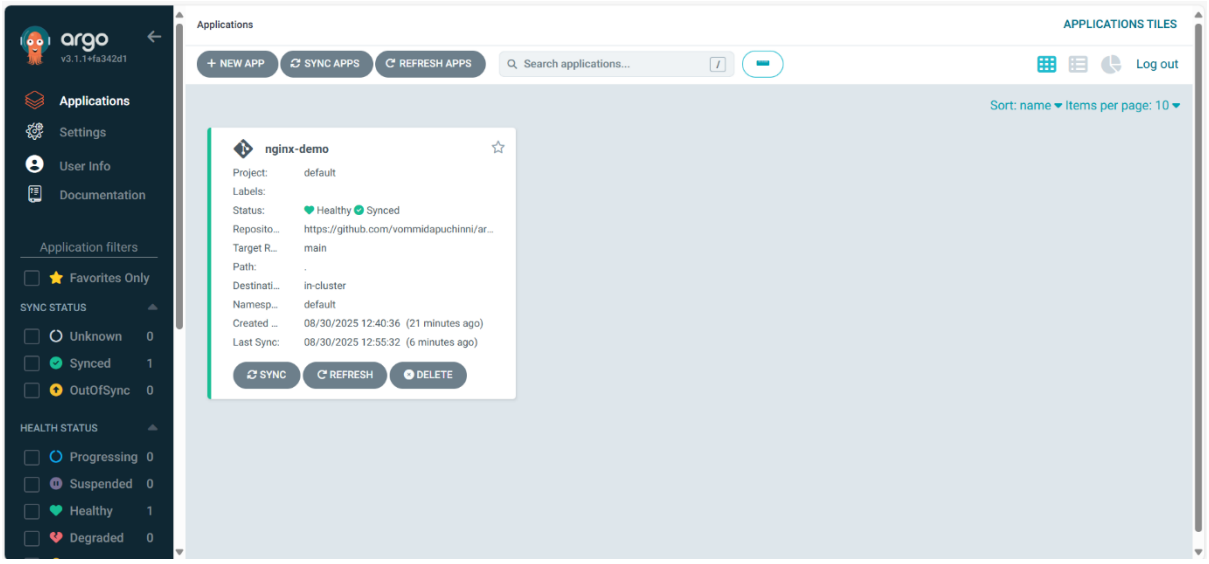
We can see that nginx image we changed and we can see that nginx images changes by kubectl pod describe

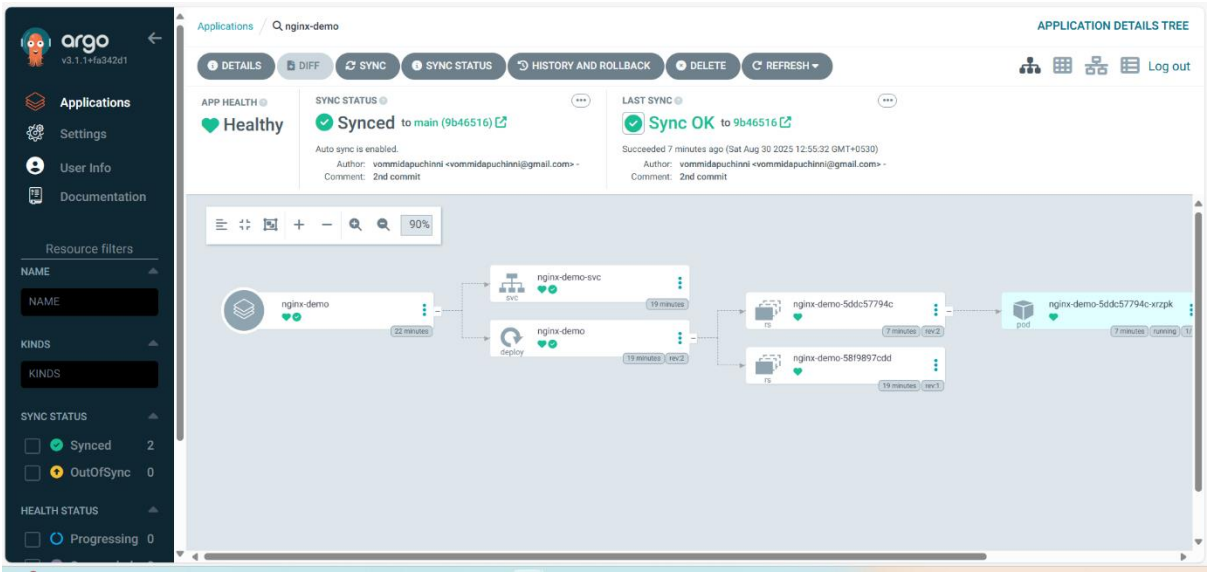
```
umama@DESKTOP-HBLJM57 MINGW64 ~/OneDrive/Desktop/argocd-k8s (main)
$ kubectl describe pod nginx-demo-58f9897cdd-4dtz5 -n default
Name:             nginx-demo-58f9897cdd-4dtz5
Namespace:        default
Priority:         0
Service Account:  default
Node:             minikube/192.168.49.2
Start Time:       Sat, 30 Aug 2025 12:43:45 +0530
Labels:           app=nginx-demo
                  pod-template-hash=58f9897cdd
Annotations:      <none>
Status:           Running
IP:               10.244.0.10
IPs:
  IP:           10.244.0.10
Controlled By:  ReplicaSet/nginx-demo-58f9897cdd
Containers:
  nginx:
    Container ID:   docker://2808b2244aaf6121f02089627c1527a830115
    Image:          nginx:stable
    Image ID:       docker-pullable://nginx@sha256:24ccf9a6192d2c6
```

```
umama@DESKTOP-HBLJM57 MINGW64 ~/OneDrive/Desktop/argocd-k8s (main)
$ kubectl describe pod nginx-demo-5ddc57794c-xrzpk -n default
Name:             nginx-demo-5ddc57794c-xrzpk
Namespace:        default
Priority:         0
Service Account:  default
Node:             minikube/192.168.49.2
Start Time:       Sat, 30 Aug 2025 12:55:33 +0530
Labels:           app=nginx-demo
                  pod-template-hash=5ddc57794c
Annotations:      <none>
Status:           Running
IP:               10.244.0.11
IPs:
  IP:           10.244.0.11
Controlled By:  ReplicaSet/nginx-demo-5ddc57794c
Containers:
  nginx:
    Container ID:   docker://1f4316927d2c109d5134a20ae2cdf3c9f5ac99b94
    Image:          nginx:1.23
    Image ID:       docker-pullable://nginx@sha256:f5747a42e3adcb31680
```

Watch Argo CD UI — with **Auto-Sync** enabled it will detect the change and apply automatically. The app sync history will show a new sync.





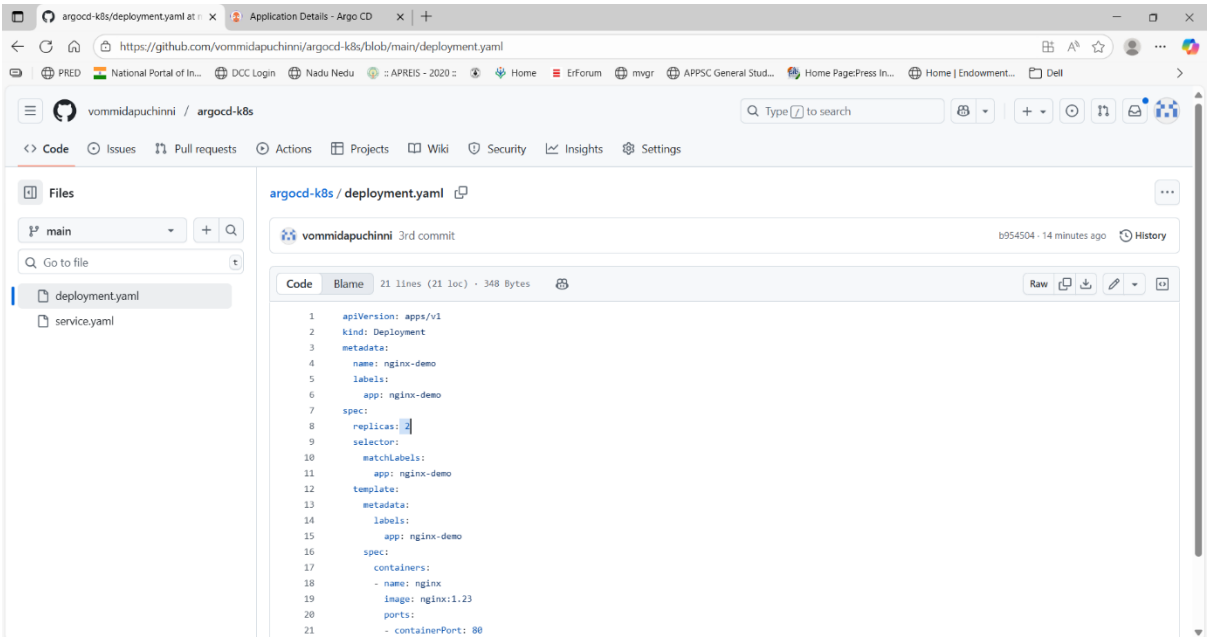


Now I changed the replicas from 1 to 2

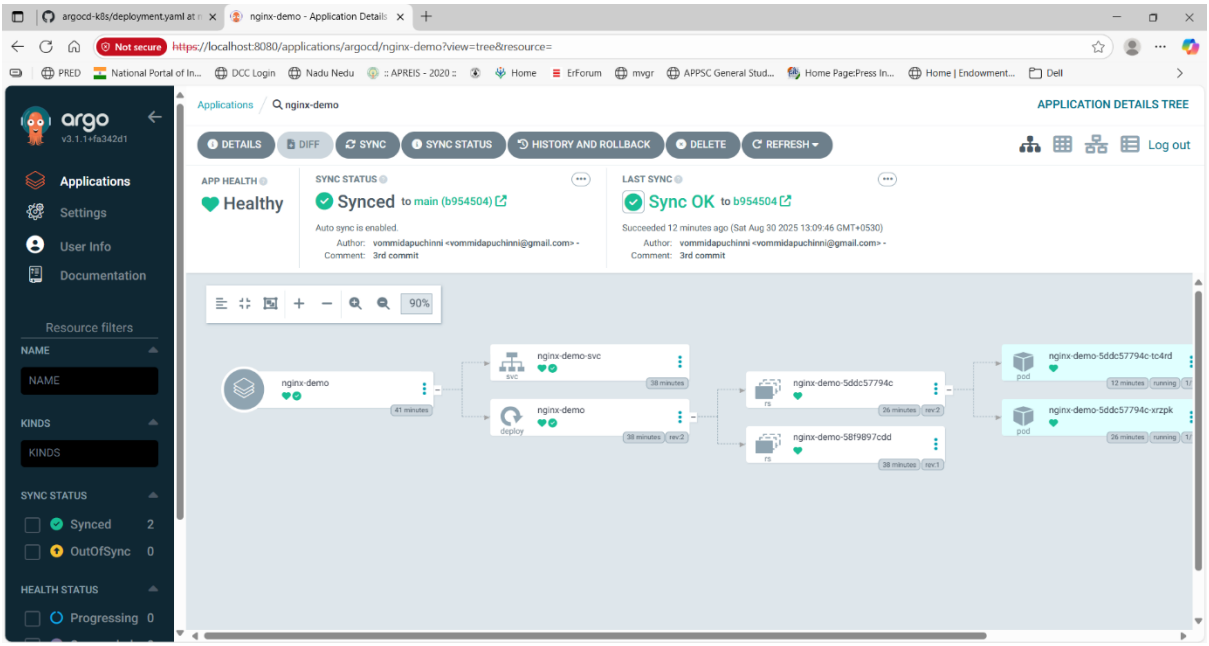
```
umama@DESKTOP-HBLJM57 MINGW64 ~/OneDrive/Desktop/argocd-k8s (main)
$ cat deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-demo
  labels:
    app: nginx-demo
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx-demo
  template:
    metadata:
      labels:
        app: nginx-demo
    spec:
      containers:
      - name: nginx
        image: nginx:1.23
        ports:
        - containerPort: 80
```

Pushed the edit file to git hub repo

```
umama@DESKTOP-HBLJM57 MINGW64 ~/OneDrive/Desktop/argocd-k8s (main)
$ vi deployment.yaml

umama@DESKTOP-HBLJM57 MINGW64 ~/OneDrive/Desktop/argocd-k8s (main)
$ git add .
warning: in the working copy of 'deployment.yaml', LF will be replaced by CRLF the next time Git touches it

umama@DESKTOP-HBLJM57 MINGW64 ~/OneDrive/Desktop/argocd-k8s (main)
$ git commit -m "3rd commit"
[main b954504] 3rd commit
 1 file changed, 1 insertion(+), 1 deletion(-)

umama@DESKTOP-HBLJM57 MINGW64 ~/OneDrive/Desktop/argocd-k8s (main)
$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 309 bytes | 154.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/vommidapuchinni/argocd-k8s.git
   9b46516..b954504  main -> main
```



We see auto sync is happened



Local verification of replicas

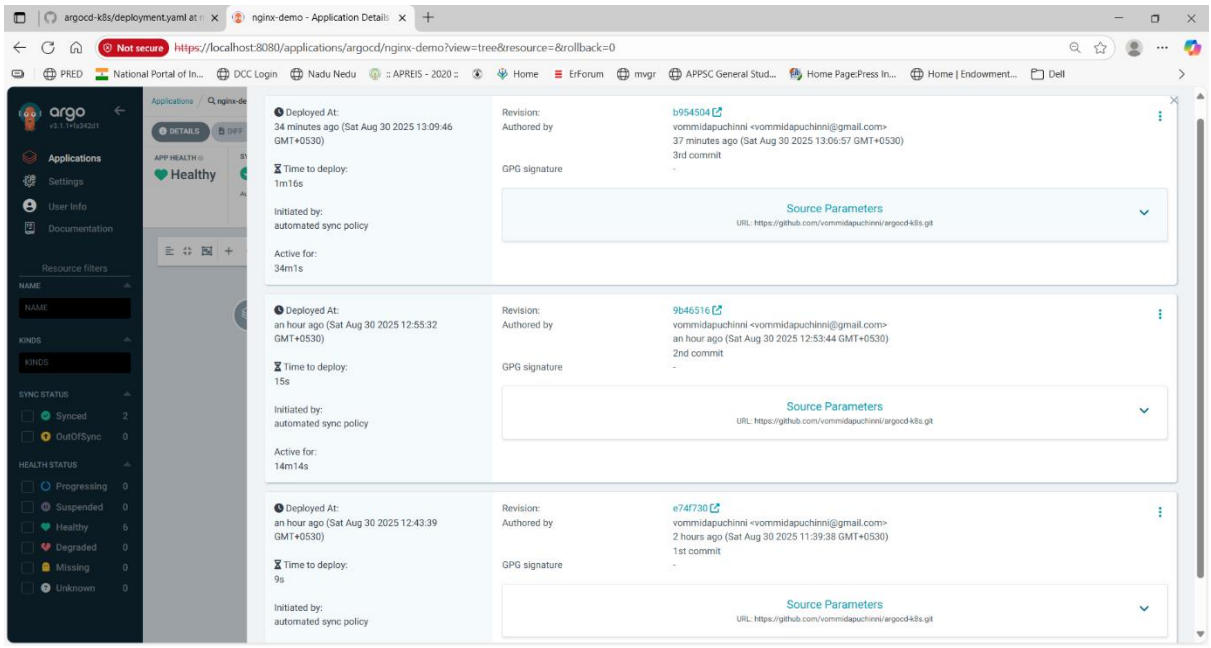
```
umama@DESKTOP-HBLJM57 MINGW64 ~/OneDrive/Desktop/argocd-k8s (main)
$ kubectl get deploy nginx-demo -o wide
NAME         READY   UP-TO-DATE   AVAILABLE   AGE   CONTAINERS   IMAGES         SELECTOR
nginx-demo   2/2     2            2           30m   nginx        nginx:1.23     app=nginx-demo

umama@DESKTOP-HBLJM57 MINGW64 ~/OneDrive/Desktop/argocd-k8s (main)
$ kubectl get pod -l app=nginx-demo
NAME                          READY   STATUS    RESTARTS   AGE
nginx-demo-5ddc57794c-tc4rd   1/1     Running   0          4m10s
nginx-demo-5ddc57794c-xrzpk   1/1     Running   0          18m

umama@DESKTOP-HBLJM57 MINGW64 ~/OneDrive/Desktop/argocd-k8s (main)
$ kubectl get svc nginx-demo-svc
NAME             TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE
nginx-demo-svc   ClusterIP   10.107.132.117   <none>        80/TCP    30m

umama@DESKTOP-HBLJM57 MINGW64 ~/OneDrive/Desktop/argocd-k8s (main)
$
```

We can see our path local by using this command

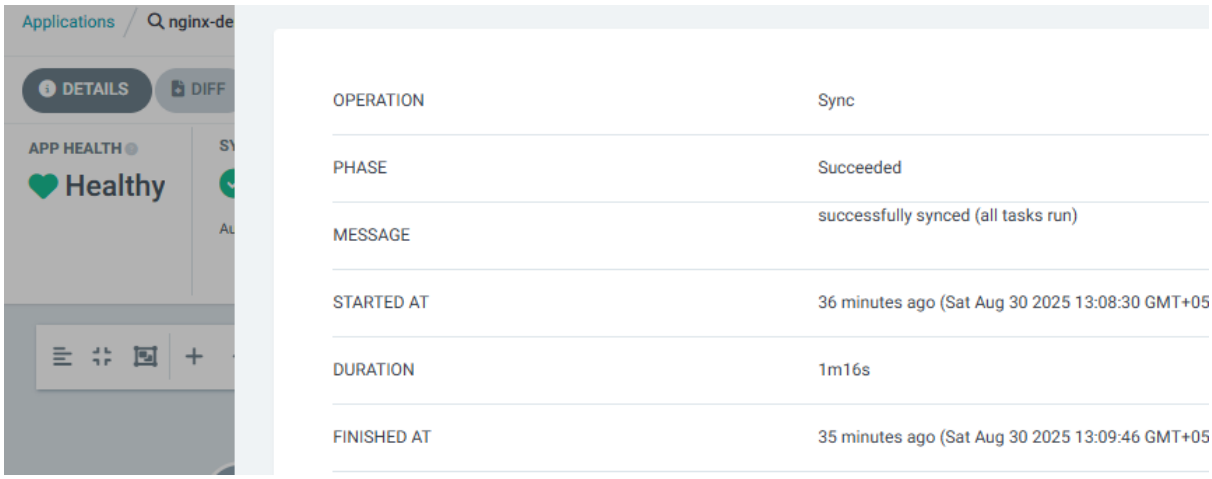
```
umama@DESKTOP-HBLJM57 MINGW64 ~/OneDrive/Desktop/argocd-k8s (main)
$ kubectl get applications -n argocd nginx-demo -o yaml | grep path:
    path: .
      path: .
      path: .
      path: .
        path: .
        path: .

umama@DESKTOP-HBLJM57 MINGW64 ~/OneDrive/Desktop/argocd-k8s (main)
$
```

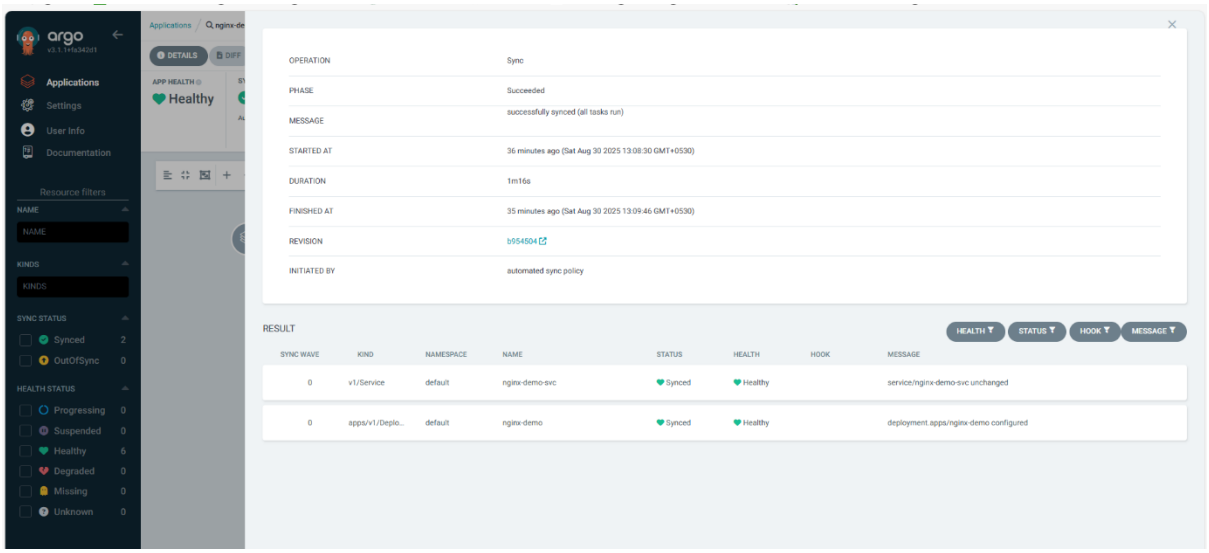
. is given because the files are in root of git repo not in inside other floders

We see history roll back



This is sync status





Our files repo

Optional: create the Argo CD Application declaratively (store Application YAML in Git)

If you want Argo CD itself to be configured from Git (pure GitOps), create an argocd-app.yaml:

apiVersion: argoproj.io/v1alpha1

kind: Application

metadata:

  name: nginx-demo

  namespace: argocd

```
spec:
  project: default
  source:
    repoURL: 'https://github.com/<you>/my-argocd-demo.git'
    targetRevision: main
    path: k8s
  destination:
    server: https://kubernetes.default.svc
    namespace: default
  syncPolicy:
    automated:
      prune: true
      selfHeal: true
```

apply by

kubectl apply -f argocd-app.yaml -n argocd

This is advanced but recommended for infra-as-code

Troubles Faced

- ArgoCD UI Access: TLS/self-signed warnings when using port-forward.
- Minikube Issues: Startup failures due to insufficient memory or Docker runtime problems.
- Pod Restarts: Some ArgoCD pods (e.g., repo-server) restarted; needed logs to debug.
- Git Sync Confusion: Changes didn't reflect until YAMLs were pushed to GitHub and auto-sync enabled.
- Service Access: NodePort optional; ClusterIP and port-forward worked after fixing Minikube and Docker.
- Resolution: Followed GitOps workflow properly—YAMLs in Git → ArgoCD Auto-Sync → changes applied automatically.

Key Concepts

- GitOps: Manage Kubernetes deployments declaratively via Git; Git is the single source of truth.
- ArgoCD: Continuous delivery tool for Kubernetes that automatically syncs cluster state with Git.
- Deployment: Defines the desired state of an application (replicas, container image, labels).
- Service: Exposes a deployment inside the cluster (ClusterIP, NodePort, LoadBalancer).
- Auto-Sync: ArgoCD feature to automatically apply changes from Git to the cluster.
- Port-Forwarding: Access cluster services locally for testing UI without exposing NodePorts.
- ClusterIP vs NodePort: ClusterIP is internal access; NodePort exposes service externally.
- Pods & ReplicaSets: Pods are running instances of containers; ReplicaSets ensure desired number of replicas.

Conclusion

In this project, we implemented a GitOps workflow using ArgoCD and Kubernetes. The deployment manifests for the Nginx application were stored in Git, and ArgoCD automatically synced the desired state to the cluster. This approach ensures version-controlled, automated, and reliable deployments, demonstrating the power of GitOps for managing Kubernetes applications efficiently.