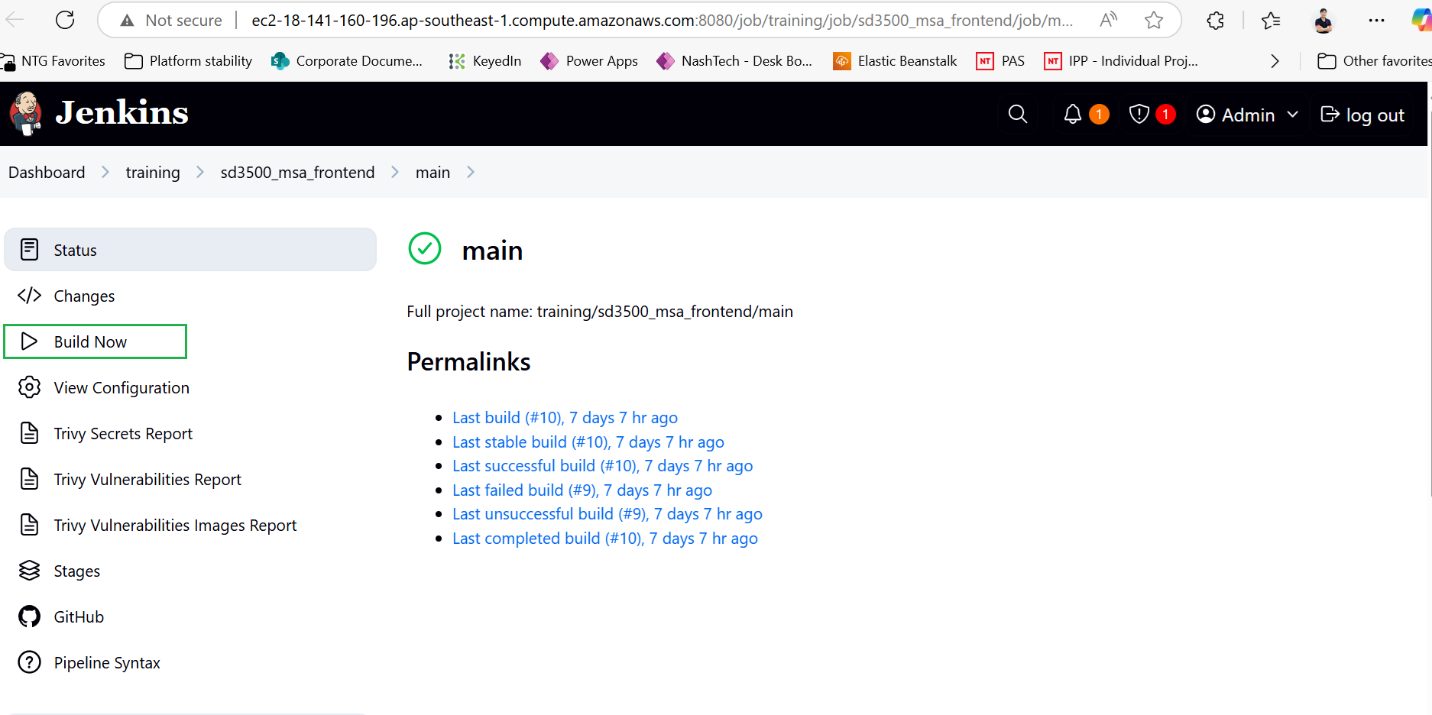
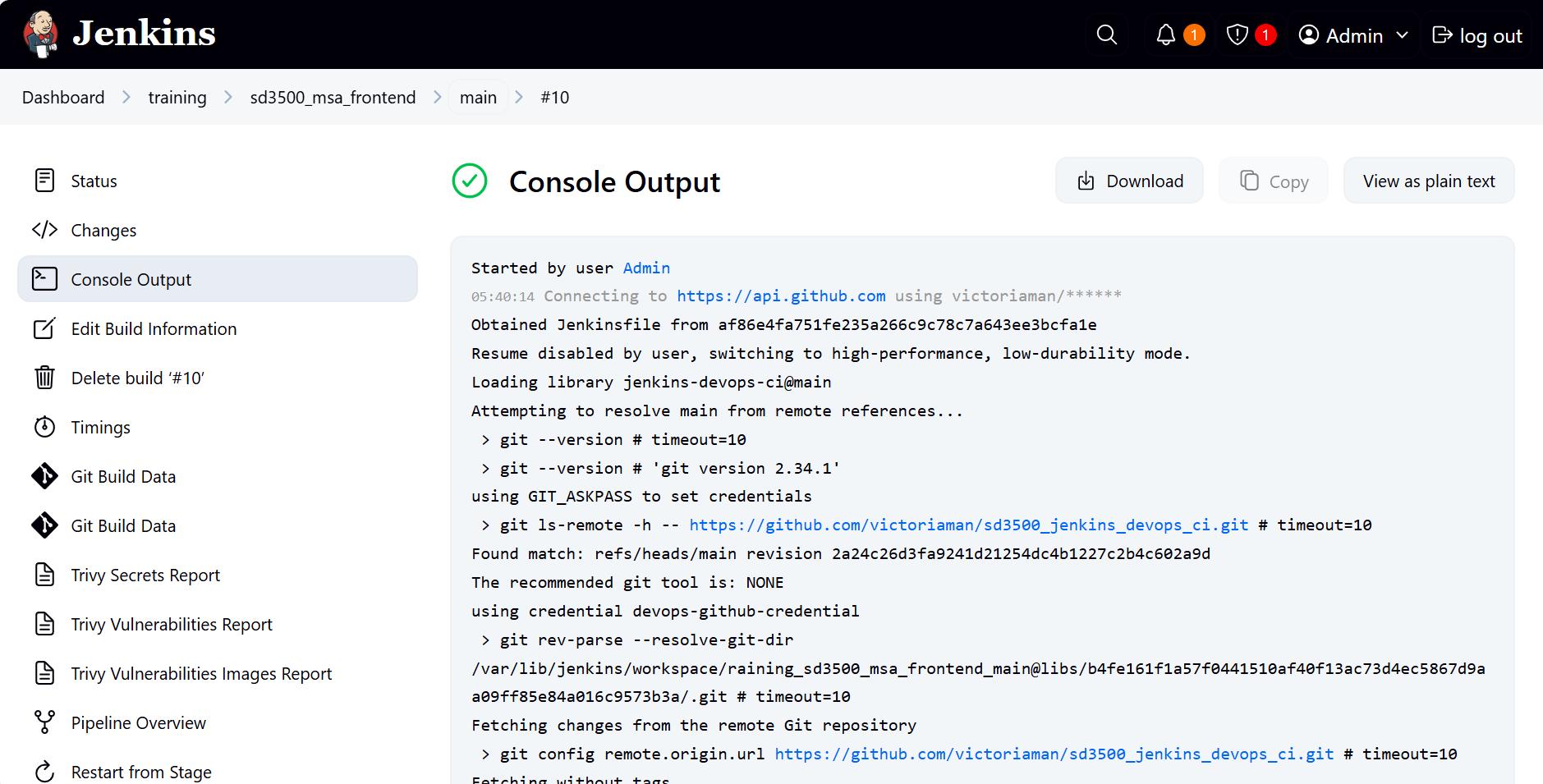
## Verify CI flow

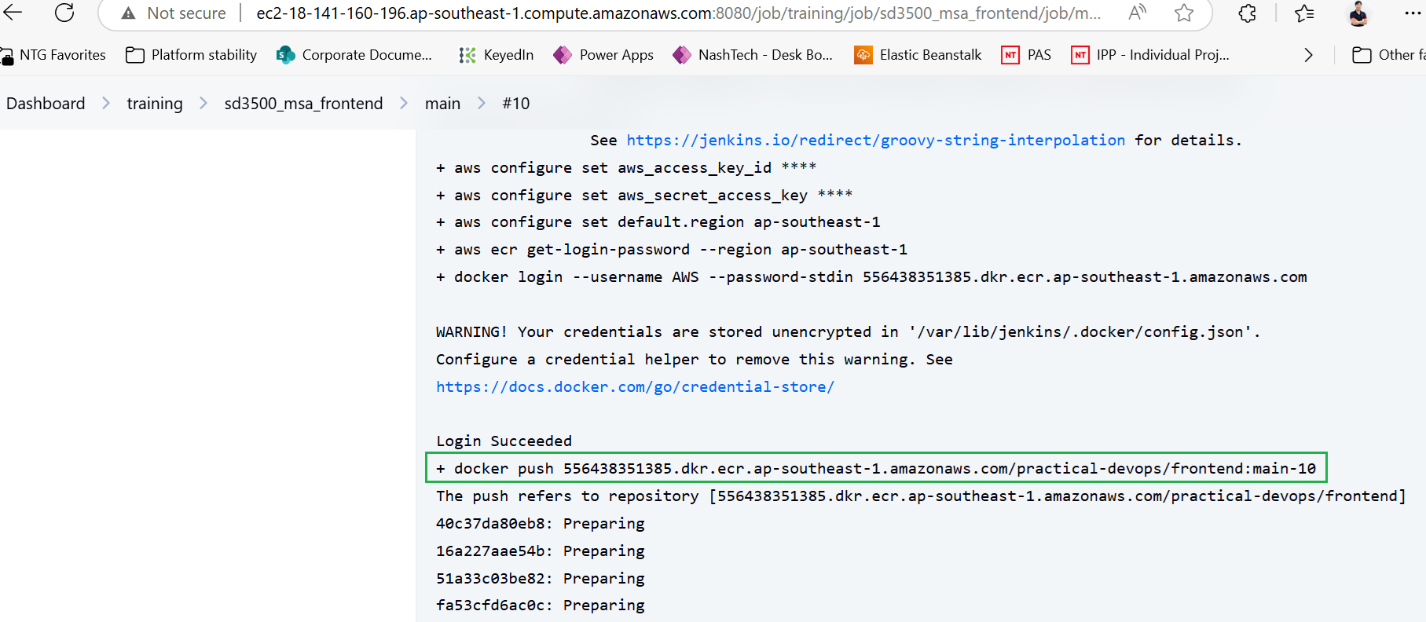
# Navigate to pipeline sd3500\_msa\_frontend/main => Build Now



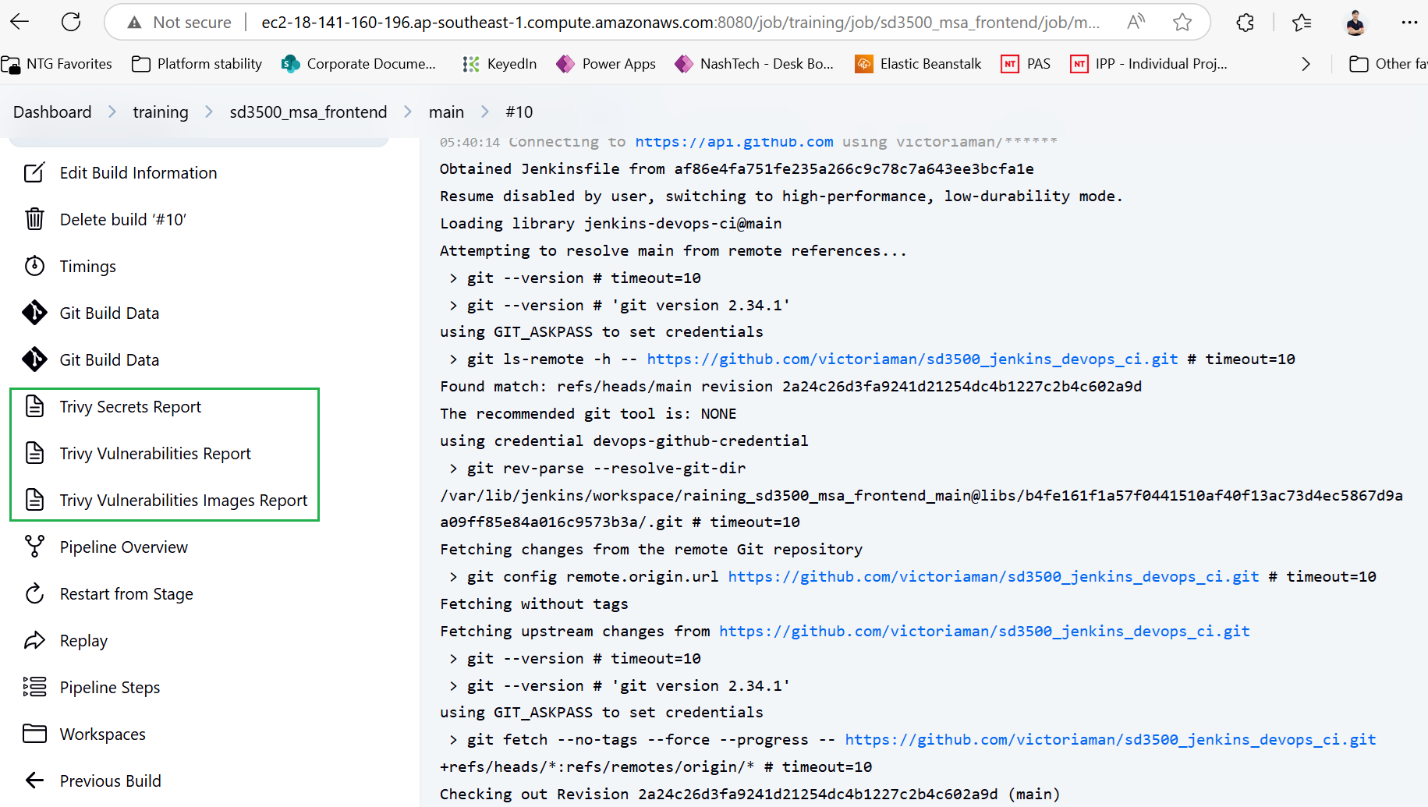
After pipeline run completed



# Tag of Docker image is print in console log

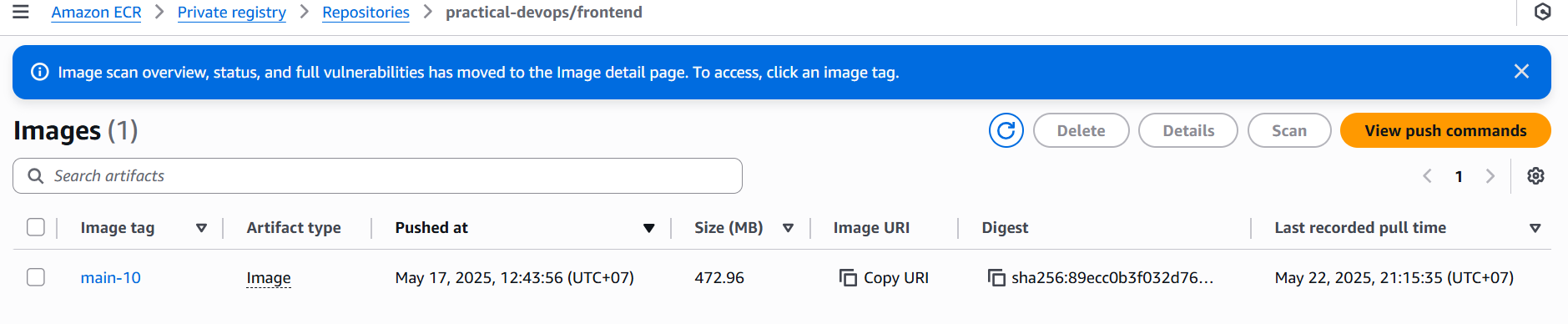


# Click Trivy Vulnerabilities Report to view detail report





# Check new image is published to AWS ECR

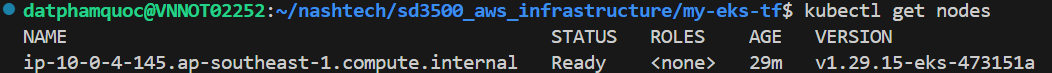


## Verify Argo CD flow

**Install Argo CD on EKS cluster**

# Check if kubectl is working as expected

kubectl get nodes



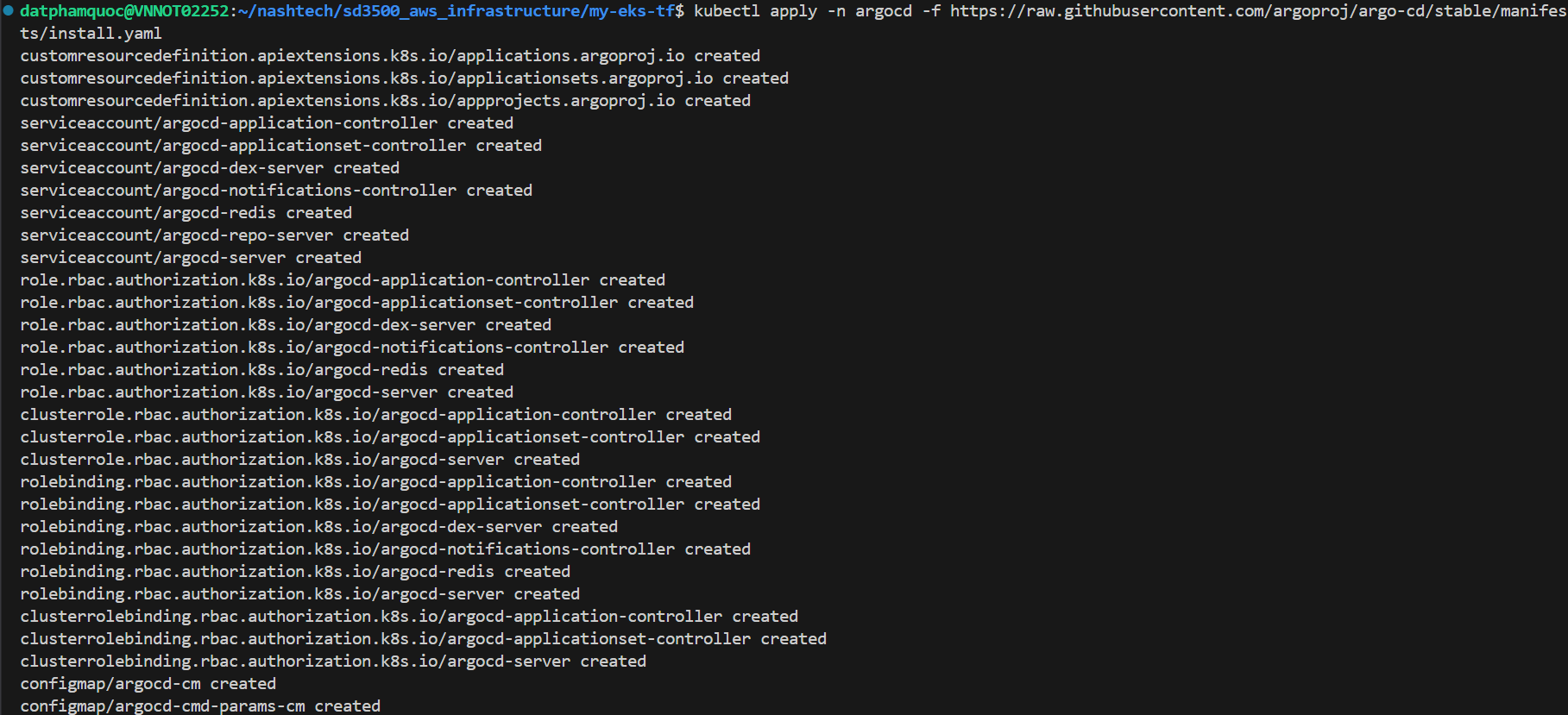
# Create namespace

kubectl create namespace argocd



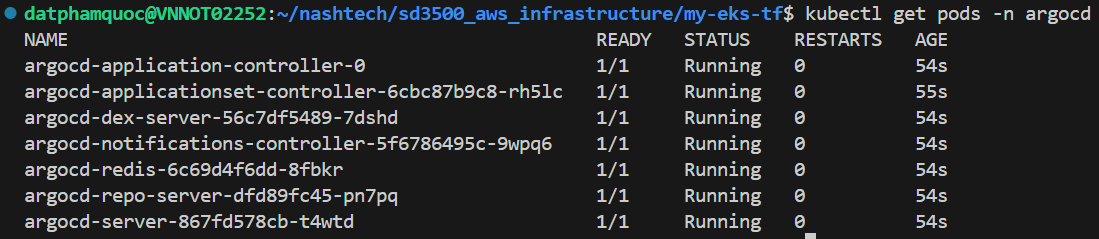
# Run the Argo CD install script provided by the project maintainers

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



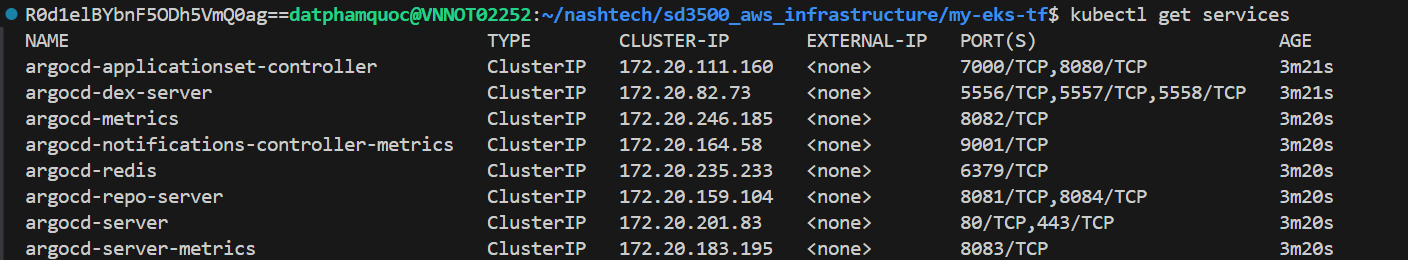
# Check the status of your Kubernetes pods

kubectl get pods -n argocd



# Check Argo service

kubectl get services –n argocd



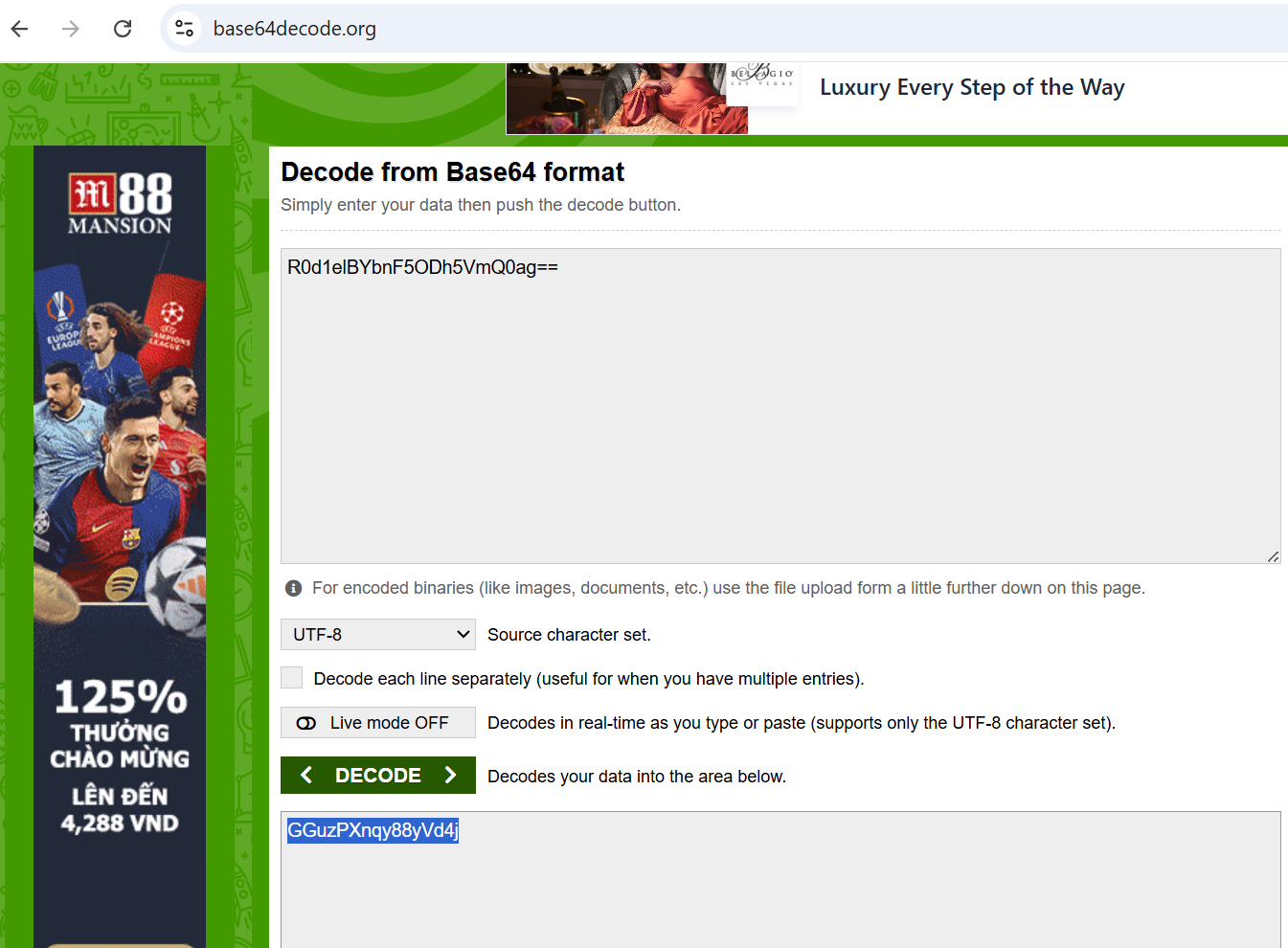
**Access to Argo CD on EKS cluster**

# Forwarding Ports to access Argo CD

Retrieve the admin password which was automatically generated during installation and decode from base64 from online.

kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath=”{.data.password}”



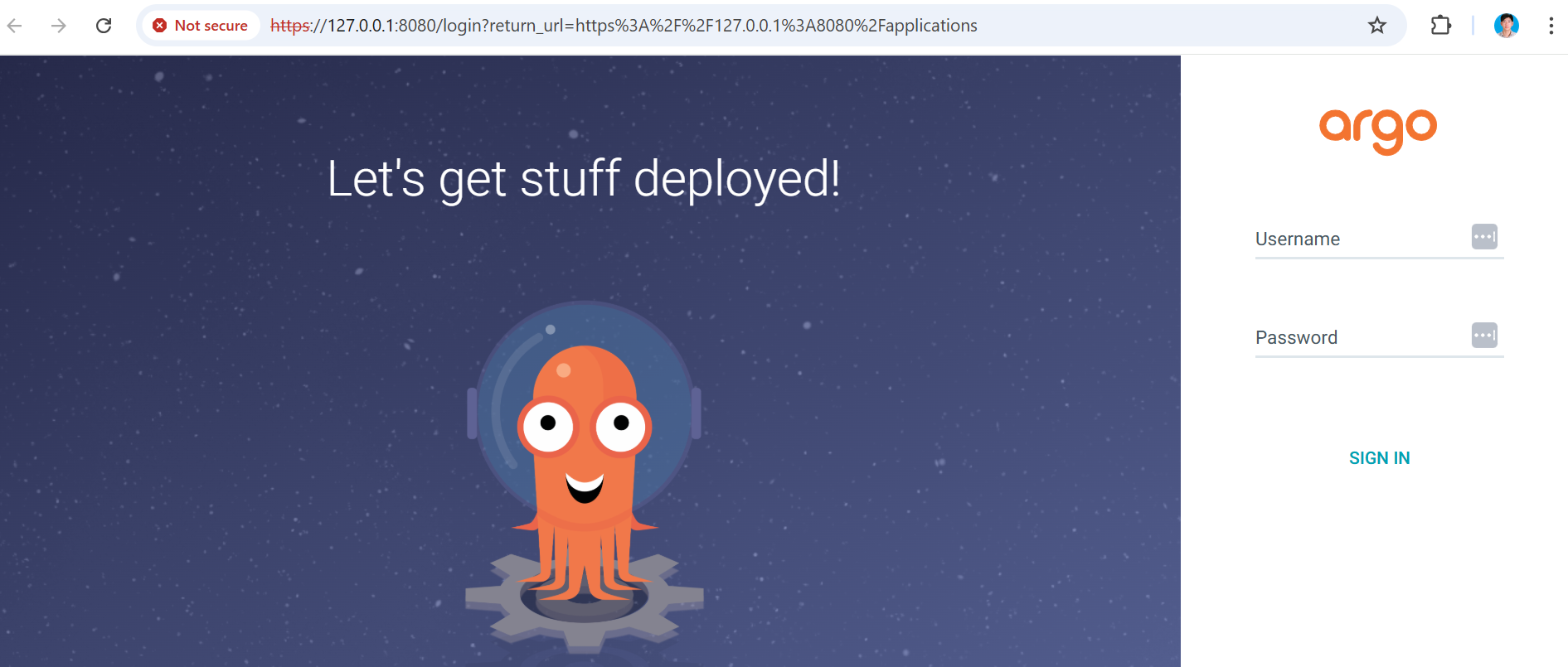
Admin password: R0d1elBYbnF5ODh5VmQ0ag==  
Decoded: GGuzPXnqy88yVd4j  


# Forward those to arbitrarily chosen other ports, like 8080

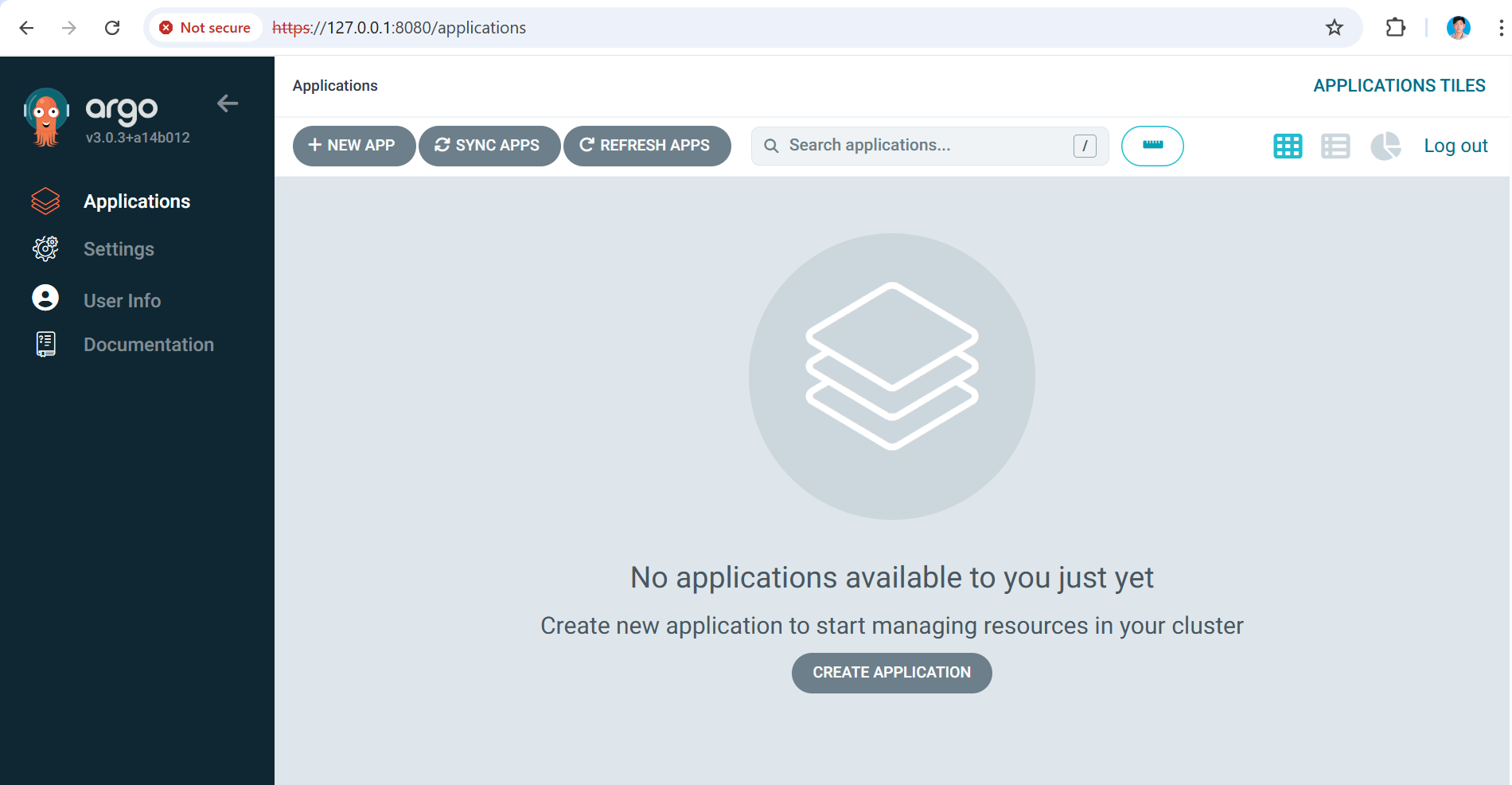
kubectl port-forward svc/argocd-server -n argocd 8080:443

# Access Argo from browser

<http://localhost:8080>



use id: admin  
password: <base64 decoded password>



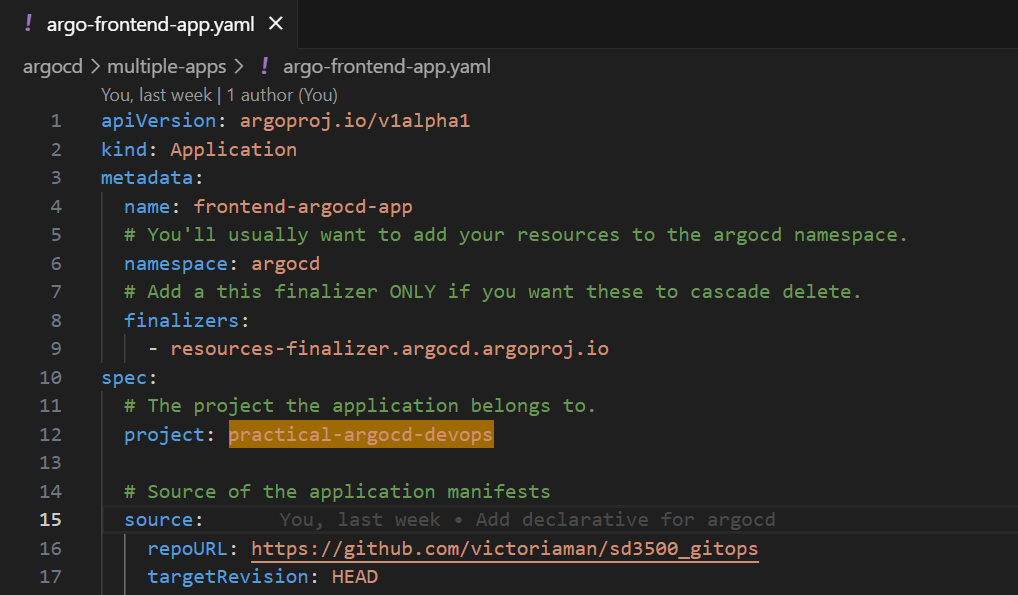
**Create frontend application in Argo CD using declarative**

An ArgoCD application is essentially a combination of a git repository, an Argo project, several sync options and other values.

This information doesn't need to be confined in Argo CD itself. It can be modeled in a Kubernetes resource so that it can be stored in Git and managed with GitOps as well.

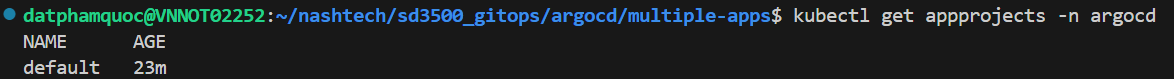
# Open terminal at ~/nashtech/sd3500\_gitops/argocd/multiple-apps

Because in argo-frontend-app.yaml, we have defined Argo project under name: practical-argocd-devops



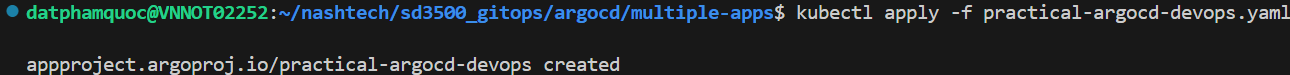
So, we need to check existence of practical-argocd-devops project

kubectl get appprojects -n argocd



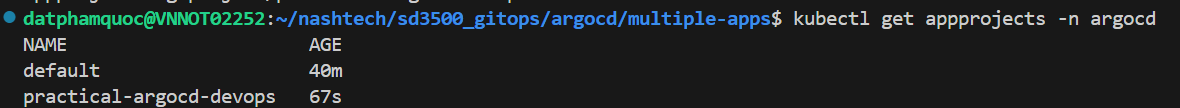
practical-argocd-devops project does display in result list, so we need run below command to create new one

kubectl apply -f practical-argocd-devops.yaml



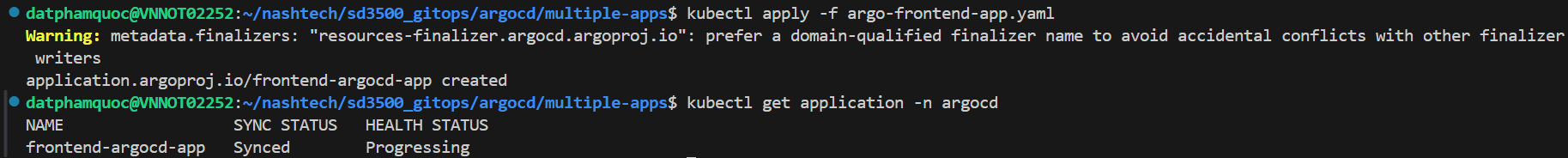
Check created practical-argocd-devops

kubectl get appprojects -n argocd

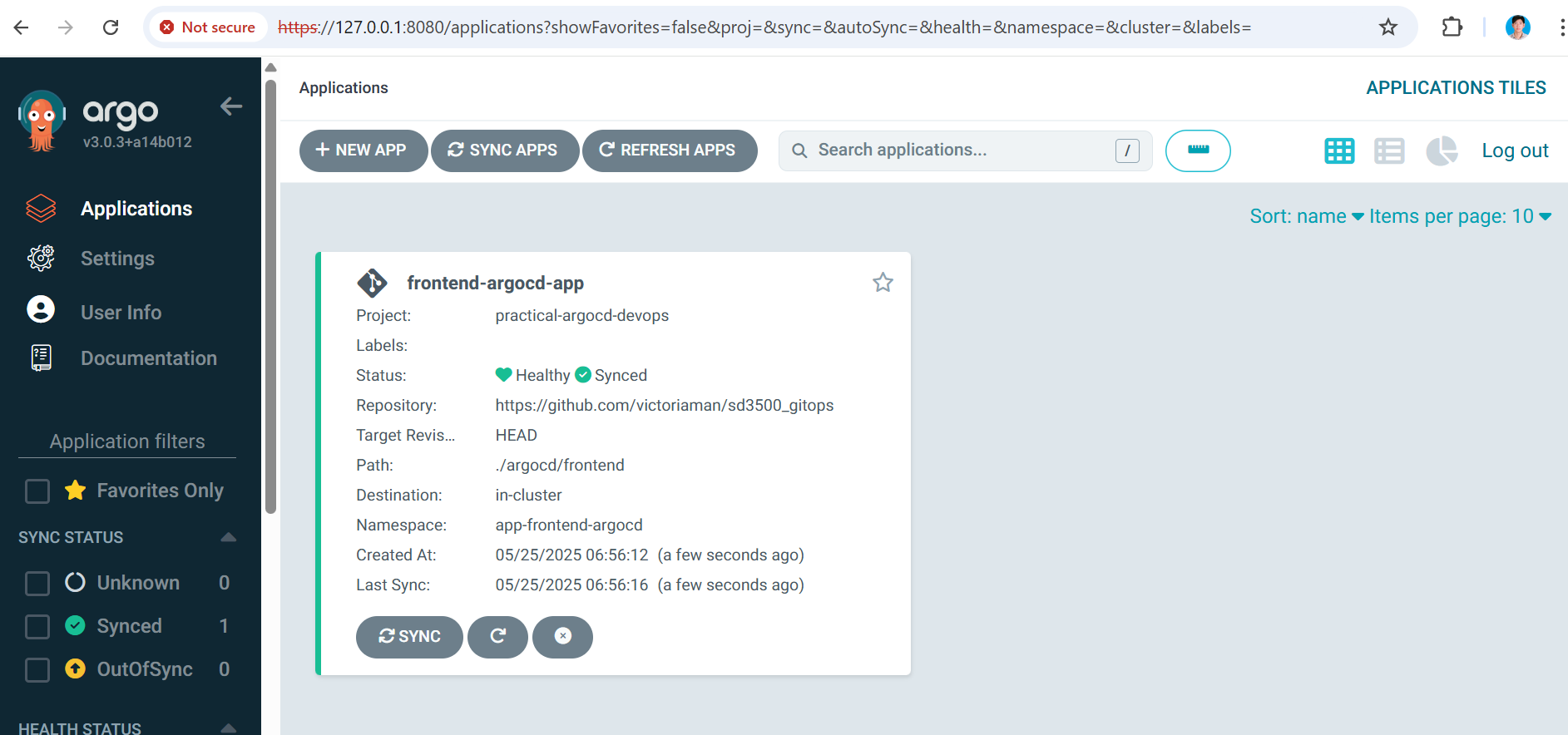


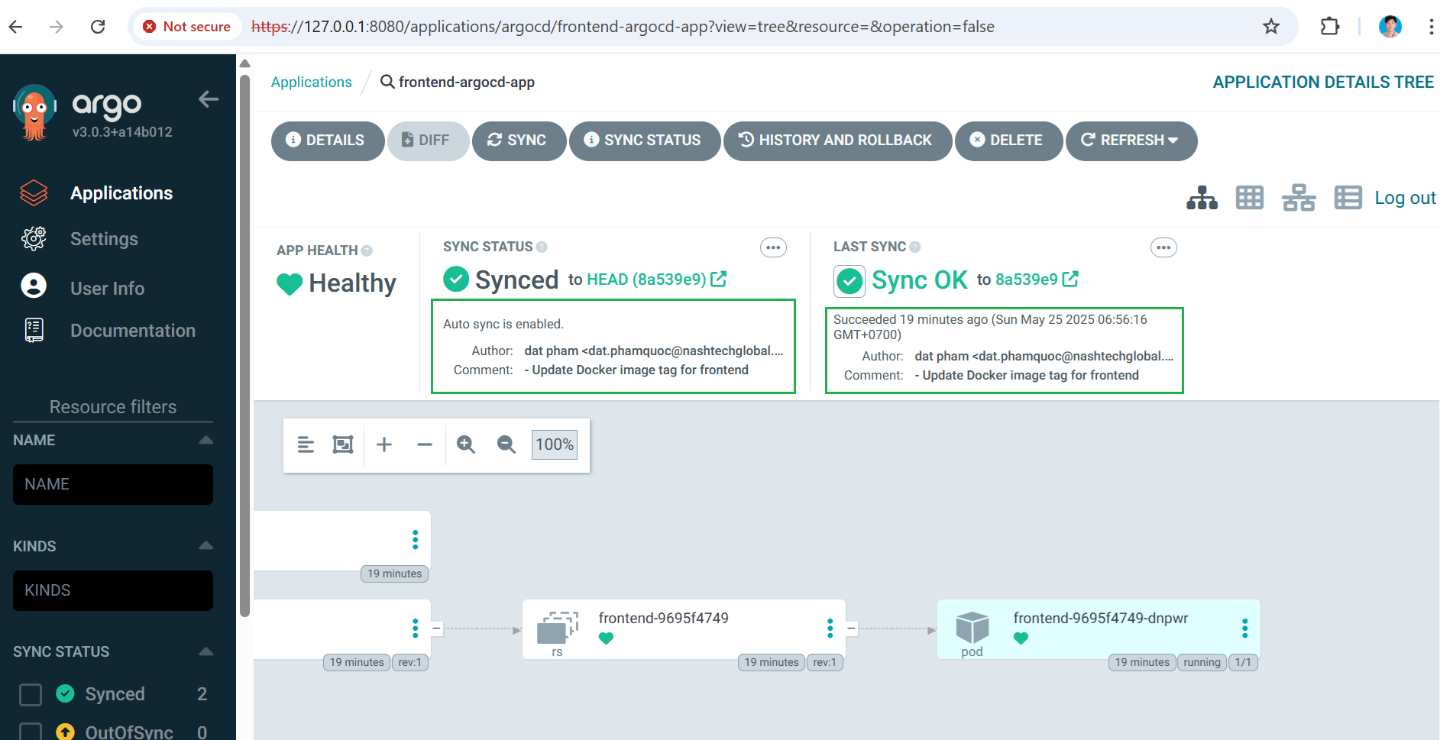
Create frontend app in Argo CD

kubectl apply -f argo-frontend-app.yaml



# Navigate to Argo CD GUI to verify

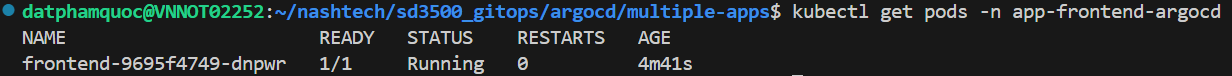




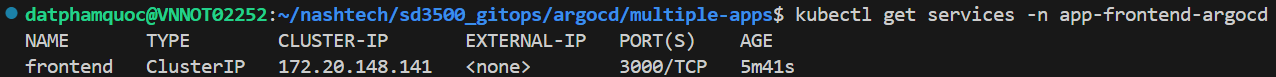
When the application created, the status should be Synced because we are using Auto sync in Sync Policy.



# Check pods  
kubectl get pods -n app-frontend-argocd

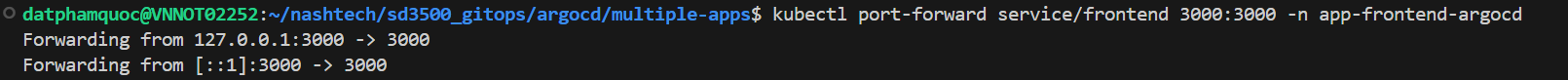


# Check services  
kubectl get services -n app-frontend-argocd

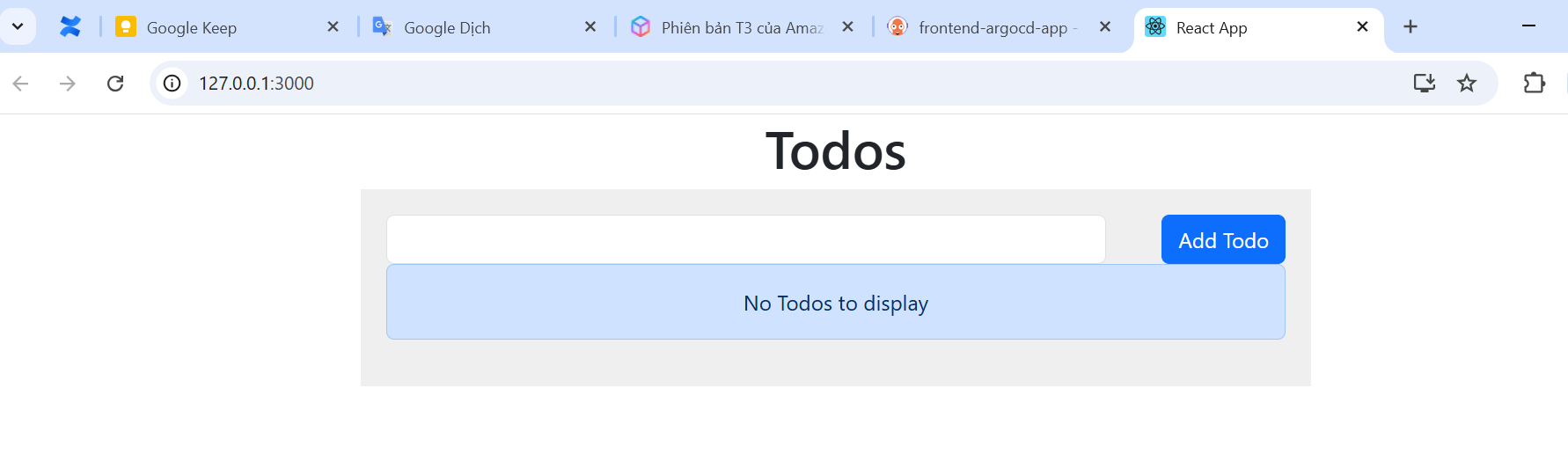


# Application verification by exposing frontend service, expose frontend service to access application

kubectl port-forward service/frontend 3000:3000 -n app-frontend-argocd



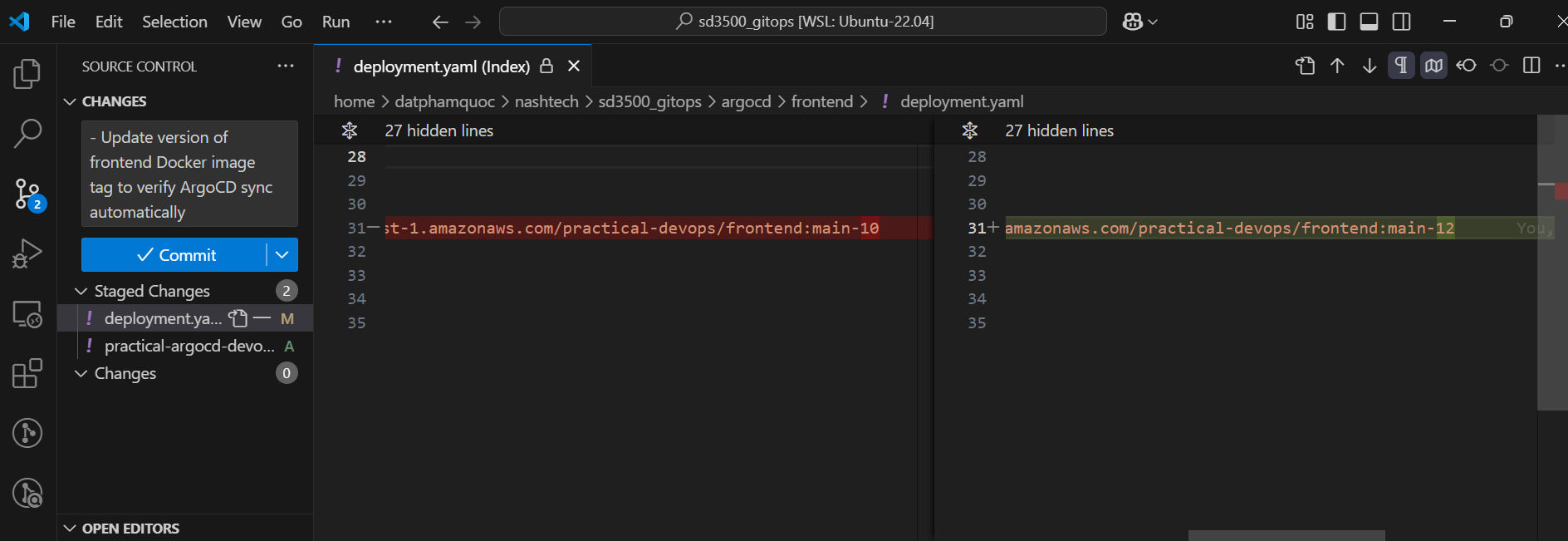
# Open browser to access application at URL: <http://locahost:3000>



## Deploy a new version – Make a change in Git

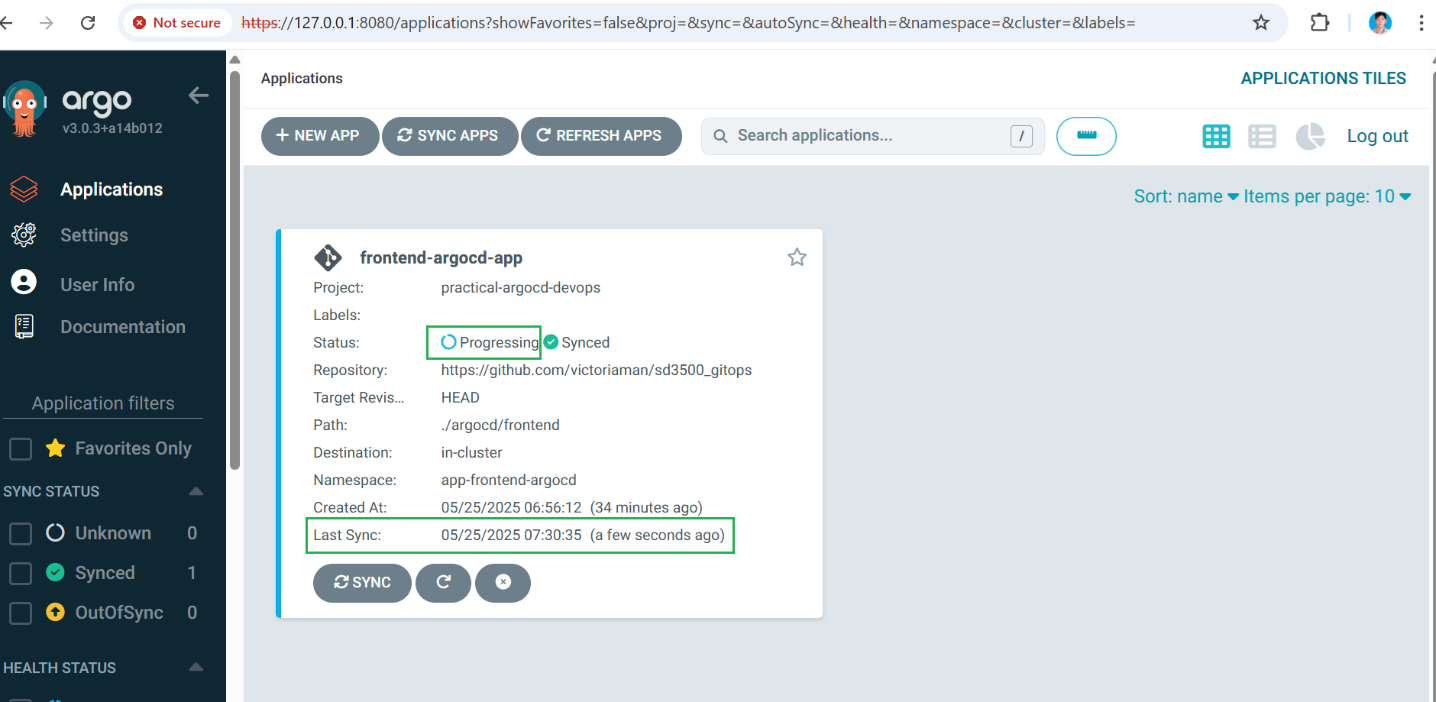
We want to deploy another version of our application. We will change the Git and see how Argo CD detects the change.

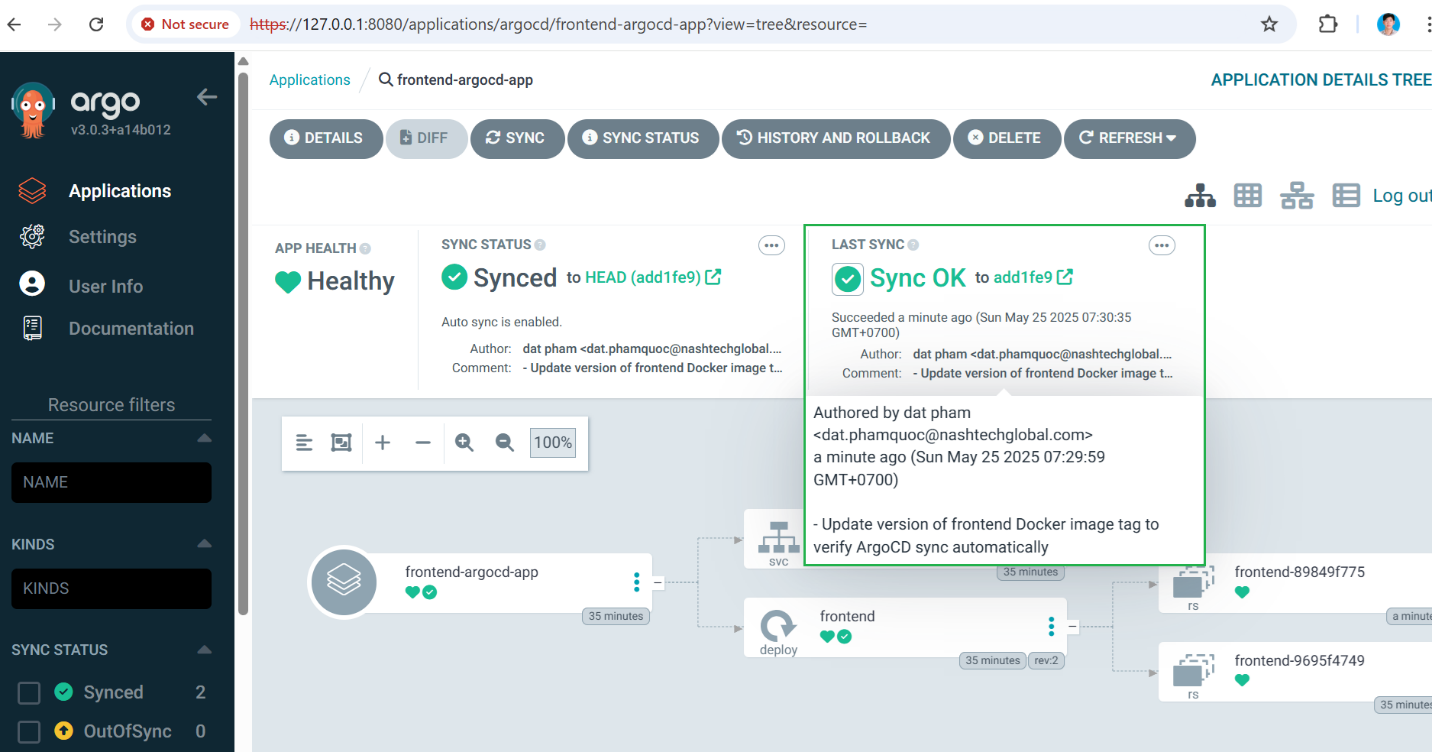
# Update version of frontend Docker image tag



Normally Argo CD checks the state between Git and the cluster every 3 minutes on its own. Just to speed things up you should click manually on the application in the Argo CD dashboard and press the "Refresh" button.

When we commit code, Argo CD will sync git with argo-frontend-app automatically





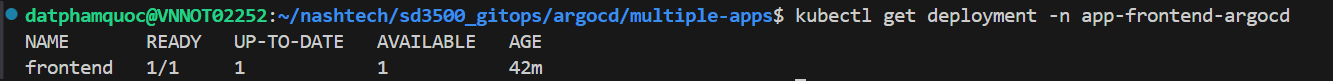
## Make a change in Cluster

Detecting changes in Git and applying is a well-known scenario. The big strength of GitOps, is that Argo CD works in the opposite direction as well. If you make any change in the cluster, then Argo CD will detect it and again tell you that something is different between Git and your cluster.

Let's say that somebody changes manually the replicas of the deployment without creating an official Pull Request (a bad practice in general).

Execute the following.  
# Get current replica of argo-frontend-app

kubectl get deployment -n app-frontend-argocd



# Manually change replica to 2

kubectl scale --replicas=2 deployment frontend -n app-frontend-argocd



Because of we have enabled self heal in argo-frontend-app, so Argo CD will prevent any change manually from CLI except from Git changing.

