############################# Source Code Management ################################

# Create Application repo in github

https://github.com/ToanLeH/sd2079\_msa

# Create Infrastructure repo in github

<https://github.com/ToanLeH/sd2079_devops_ci_cd/tree/main>

############################# Provision AWS Resource ################################

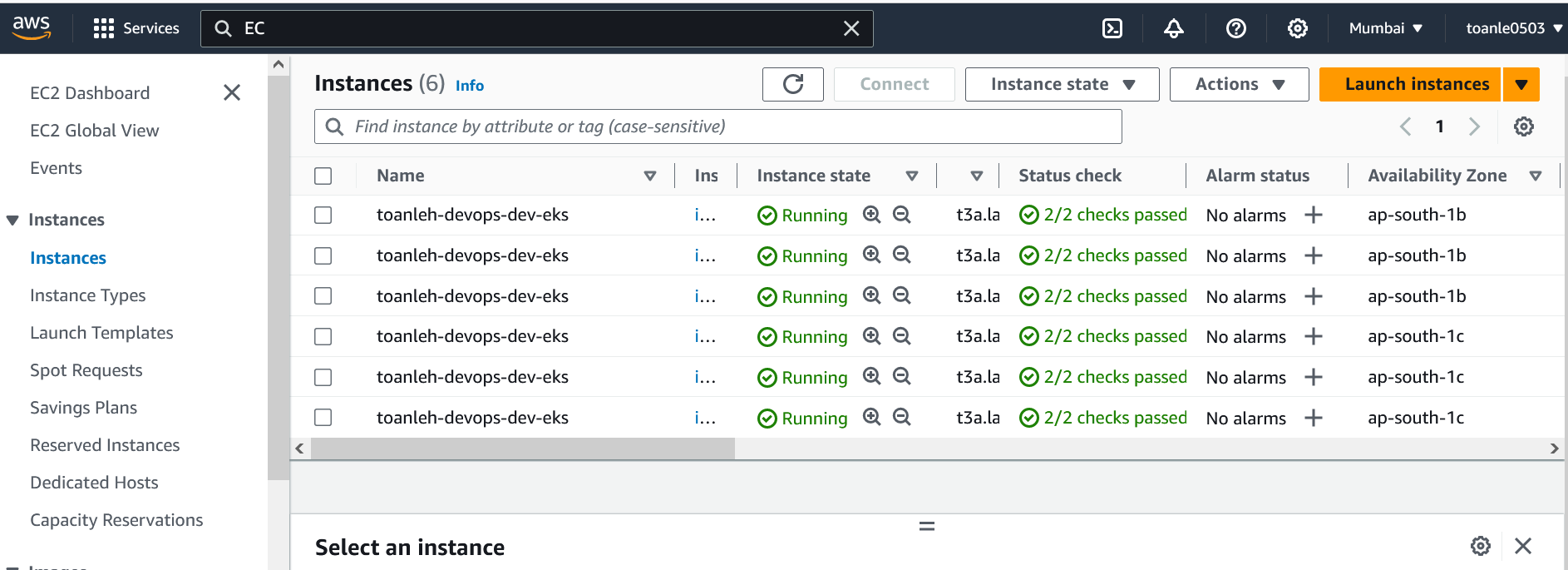
# Clone Infrastructure code and deploy aws resouce using terraform (PVC, EC2, EKS)

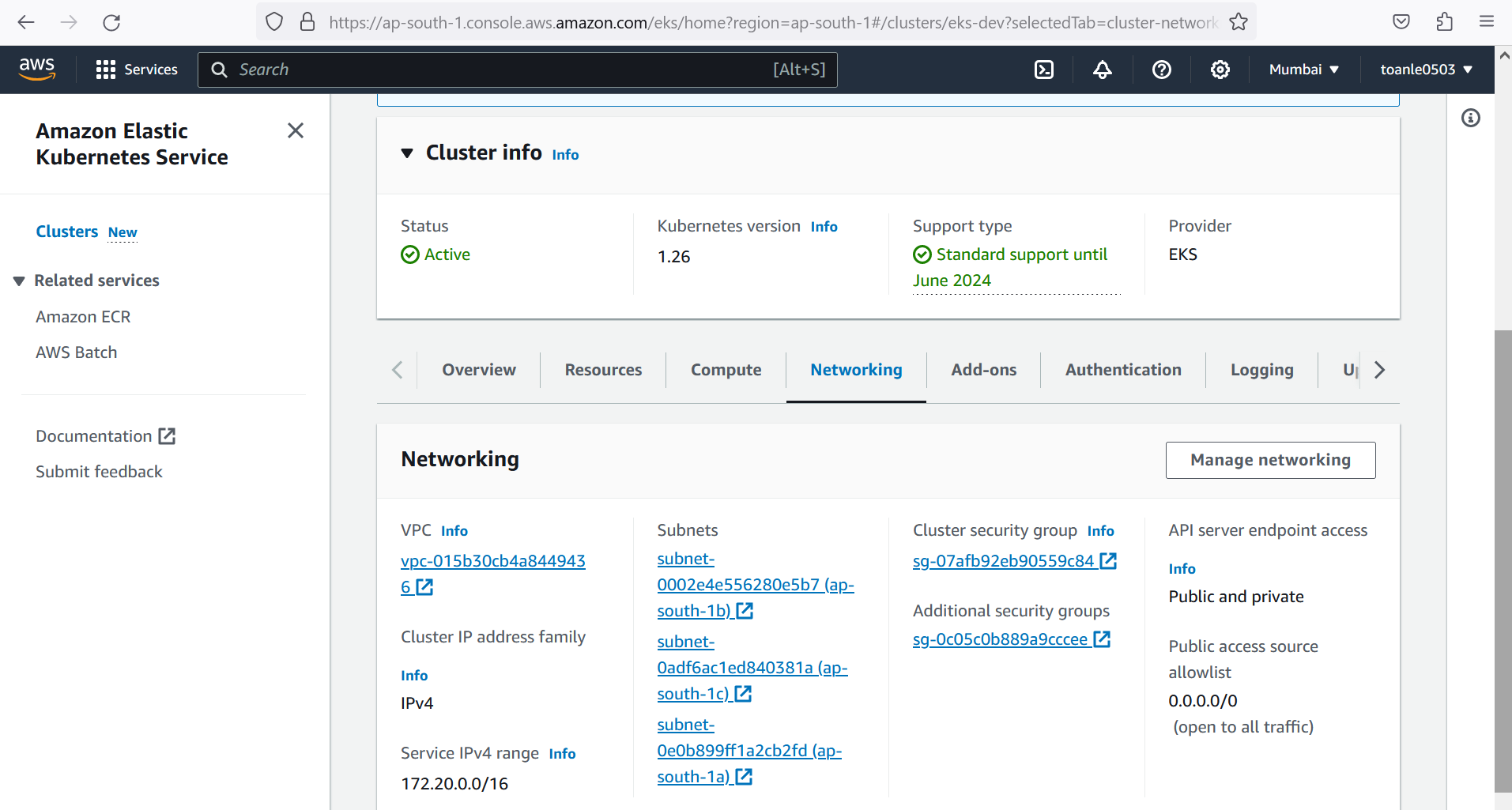
Terraform init

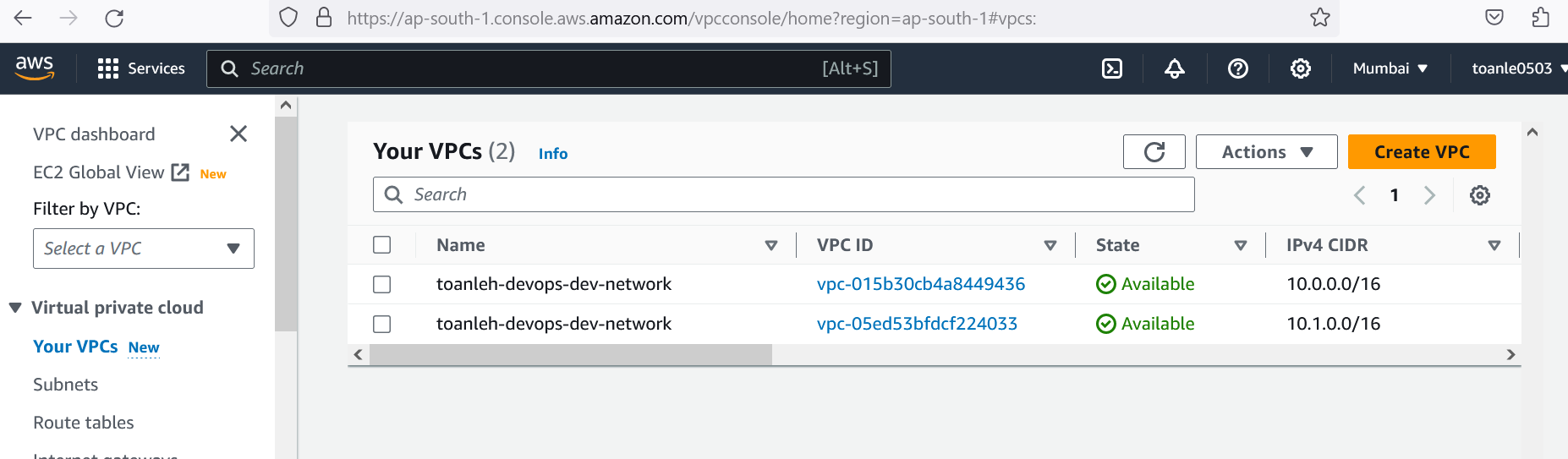
Terraform Plan

Terraform Apply

EkS workers nodes via multiple availability zones, cluster use custom vpc instead of default vpc

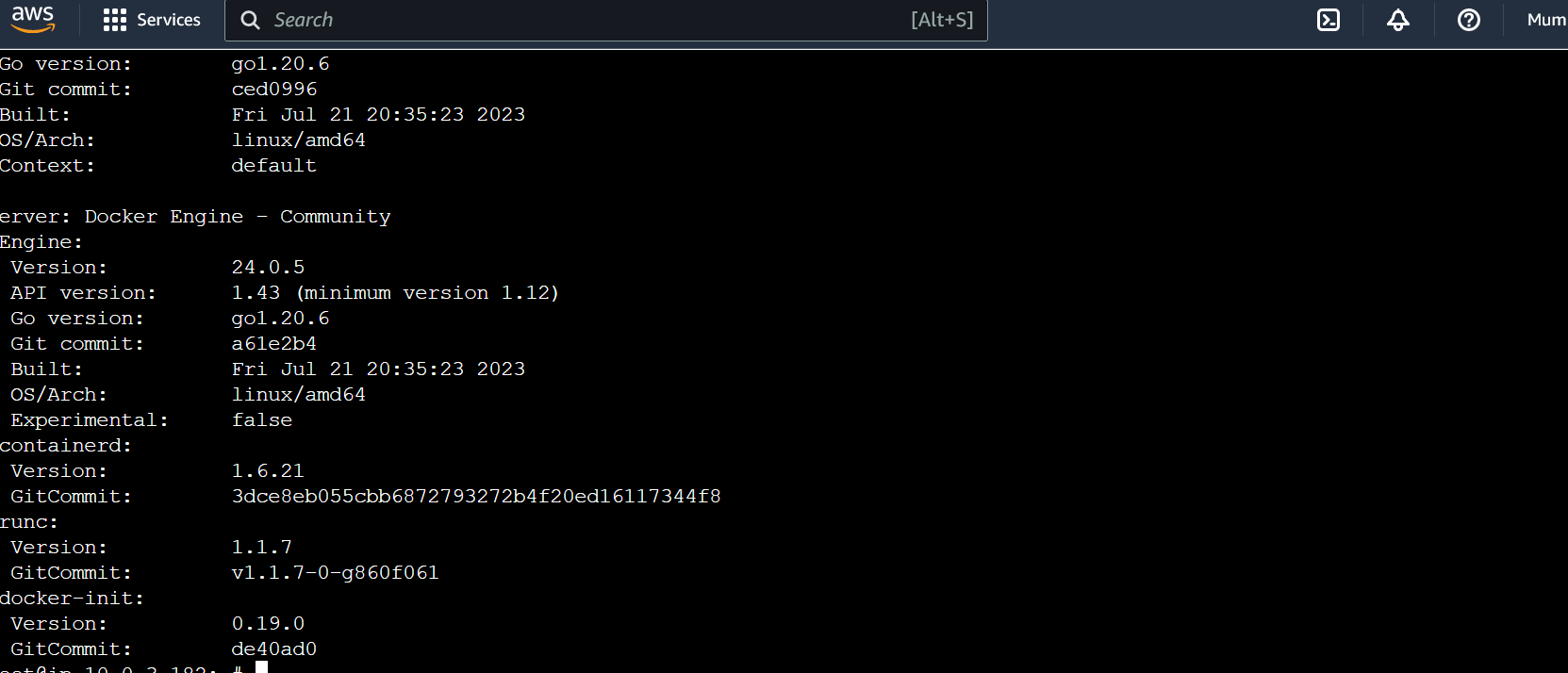


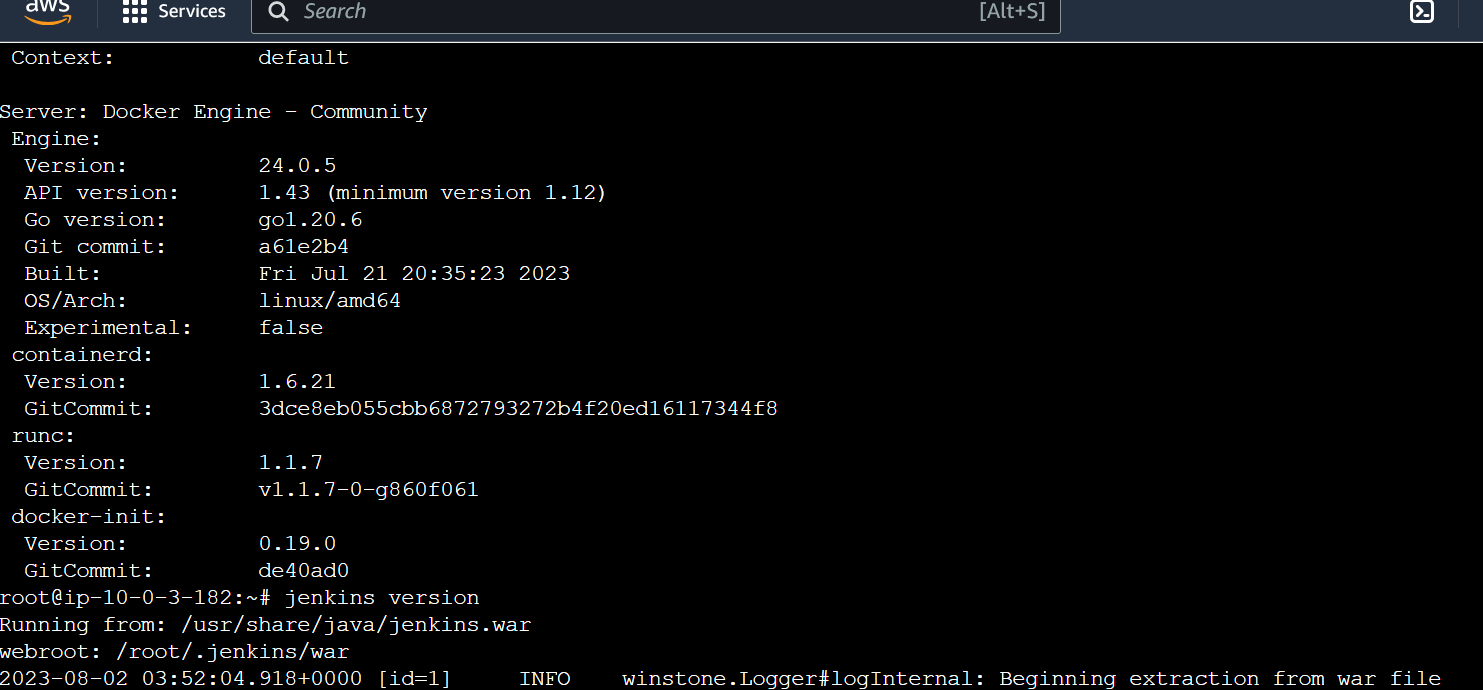


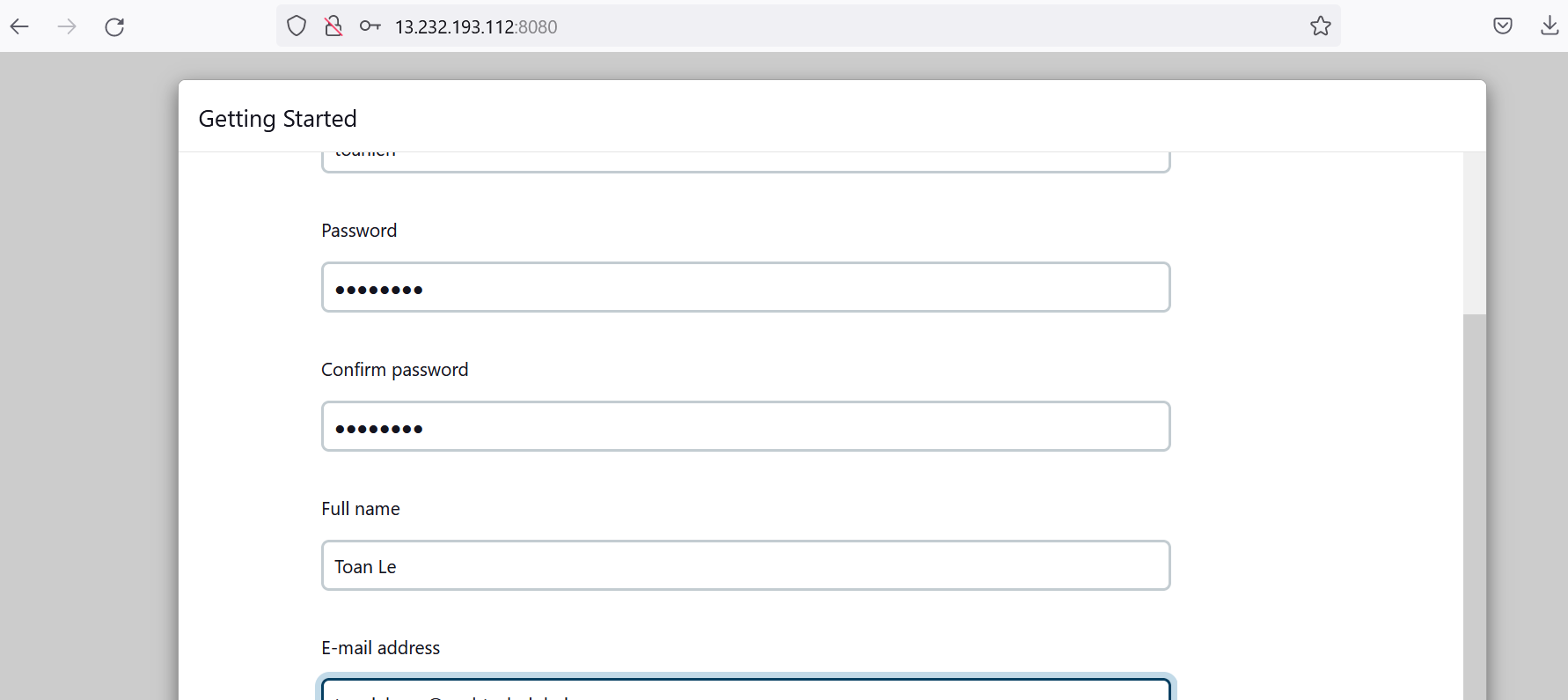


############################# Install Docker and Jenkins ################################

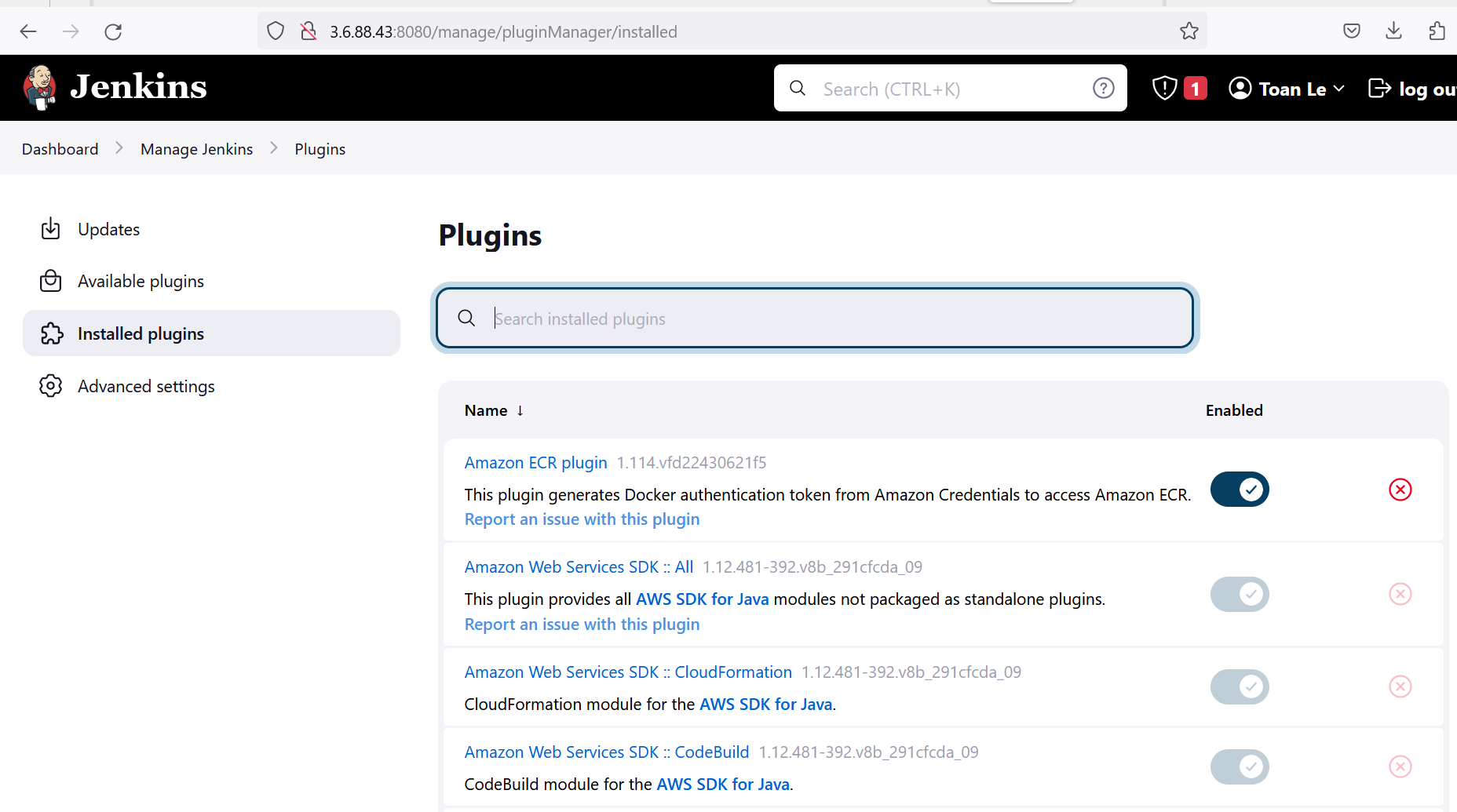
Install Jenkins via EC2 and login to set up Ci/CD pipeline





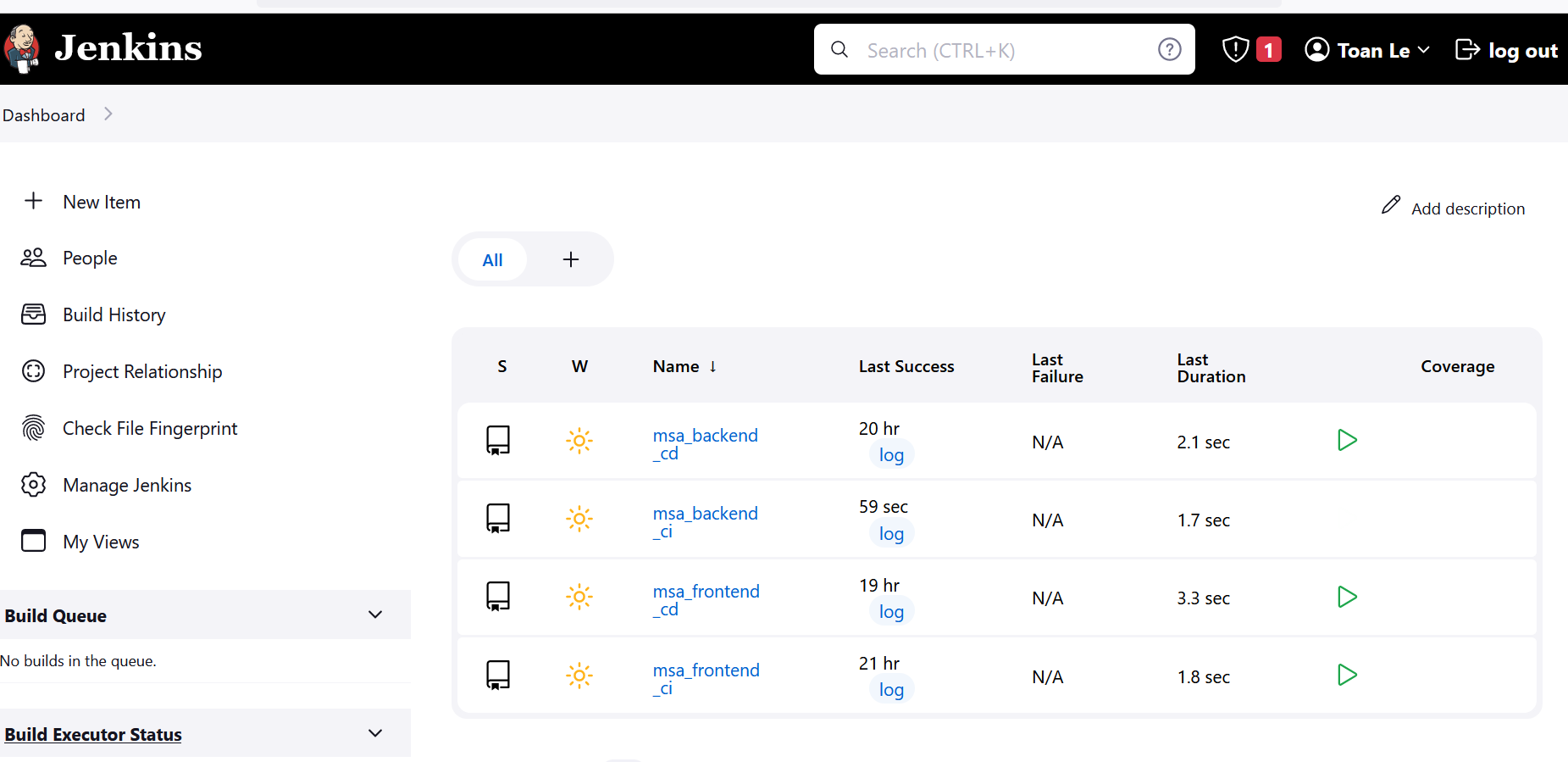


Follow prerequisite from <https://github.com/nashtech-garage/devops-ci-cd> to add require plugins and settings

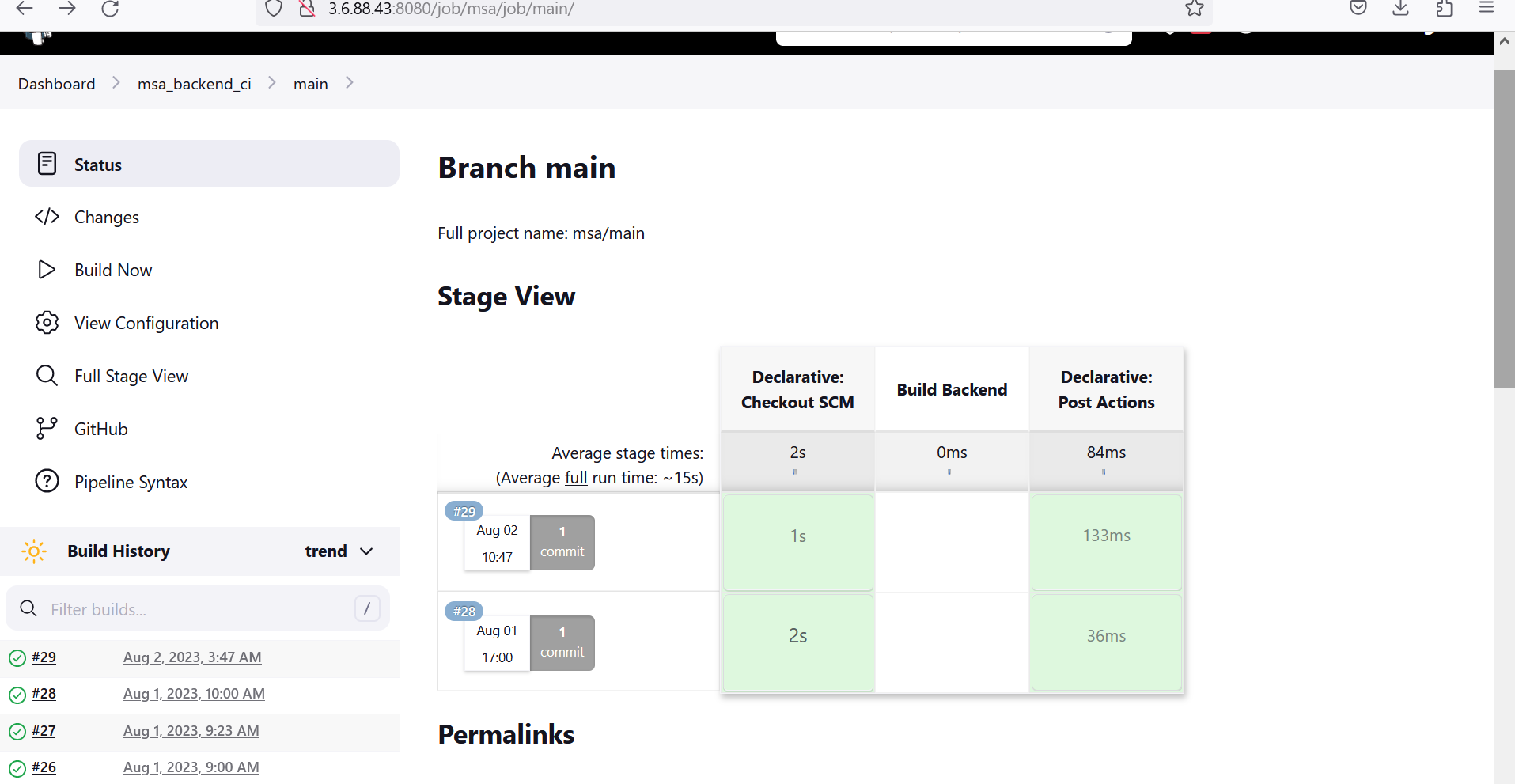


############################# Set up Jenkins CI pipeline ############################

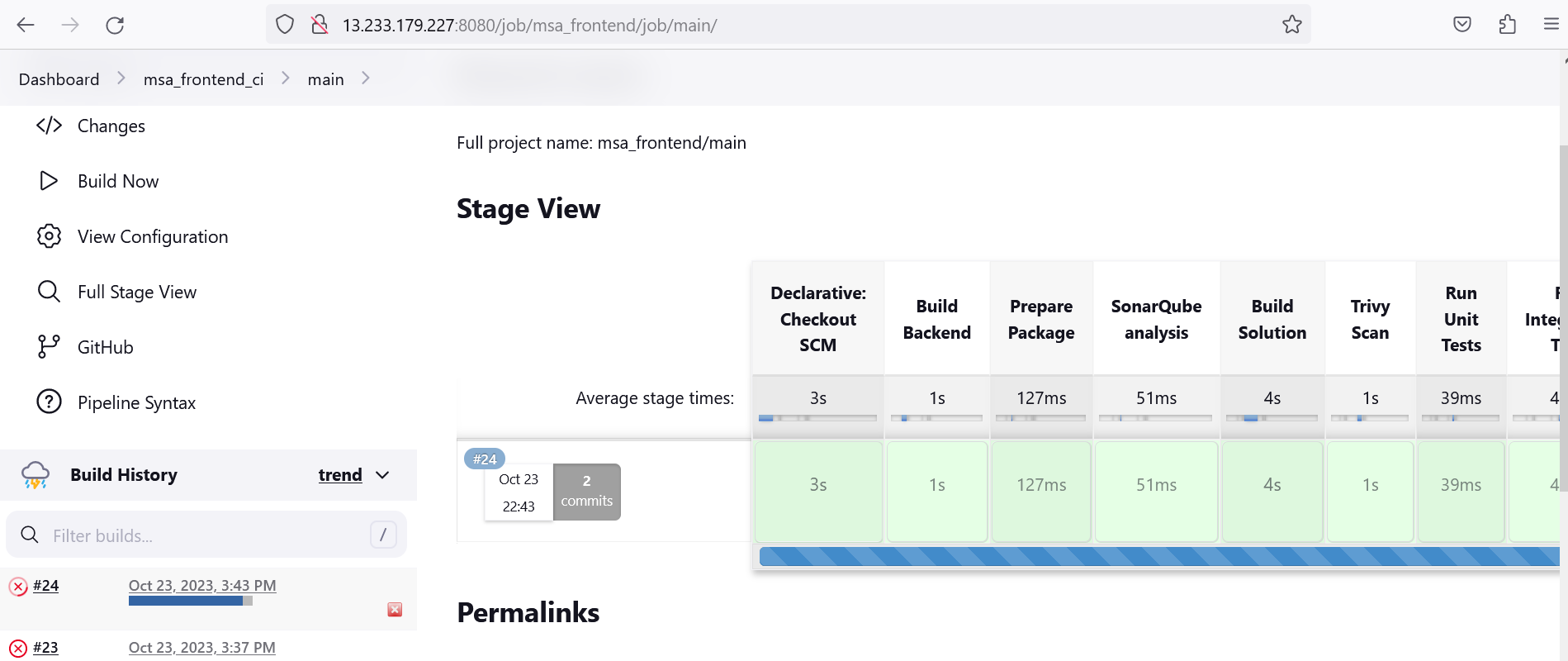
#set up CI for backend and frontend

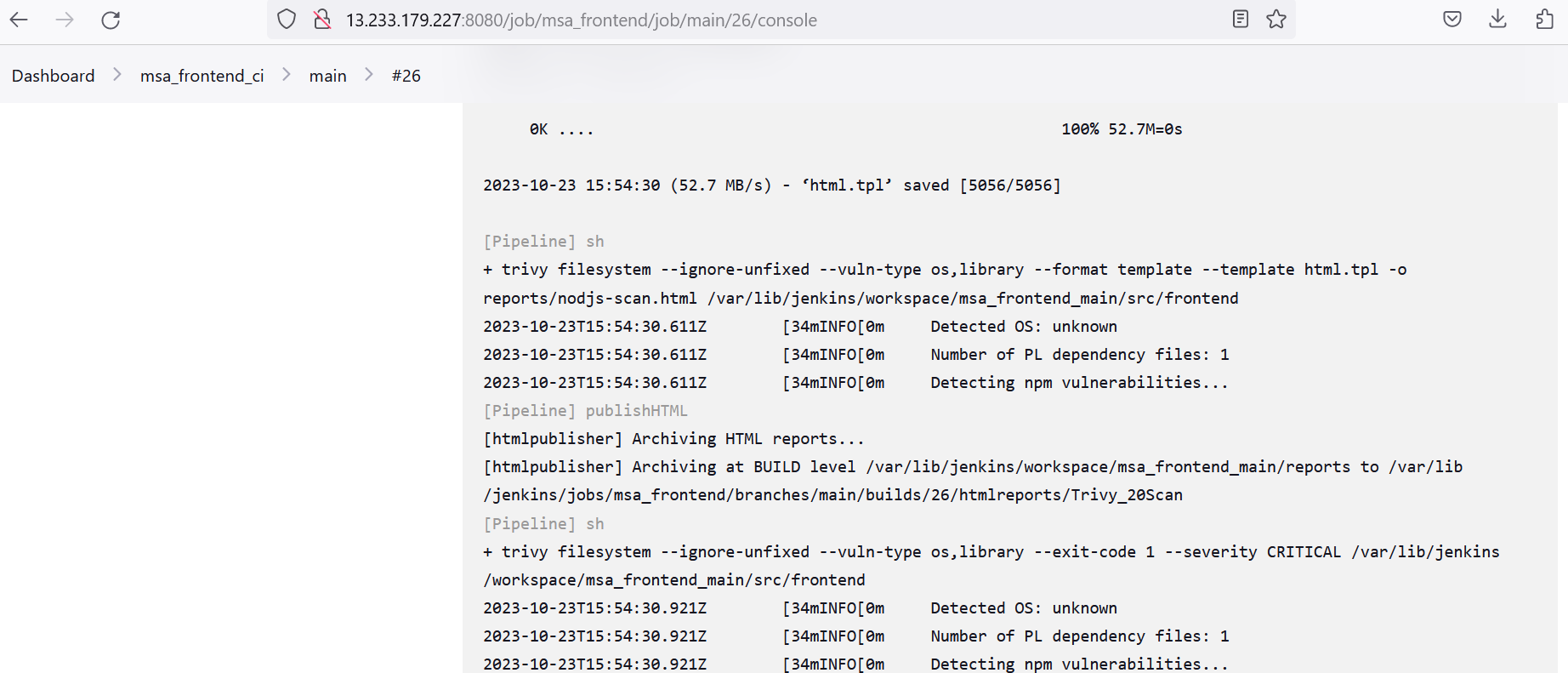


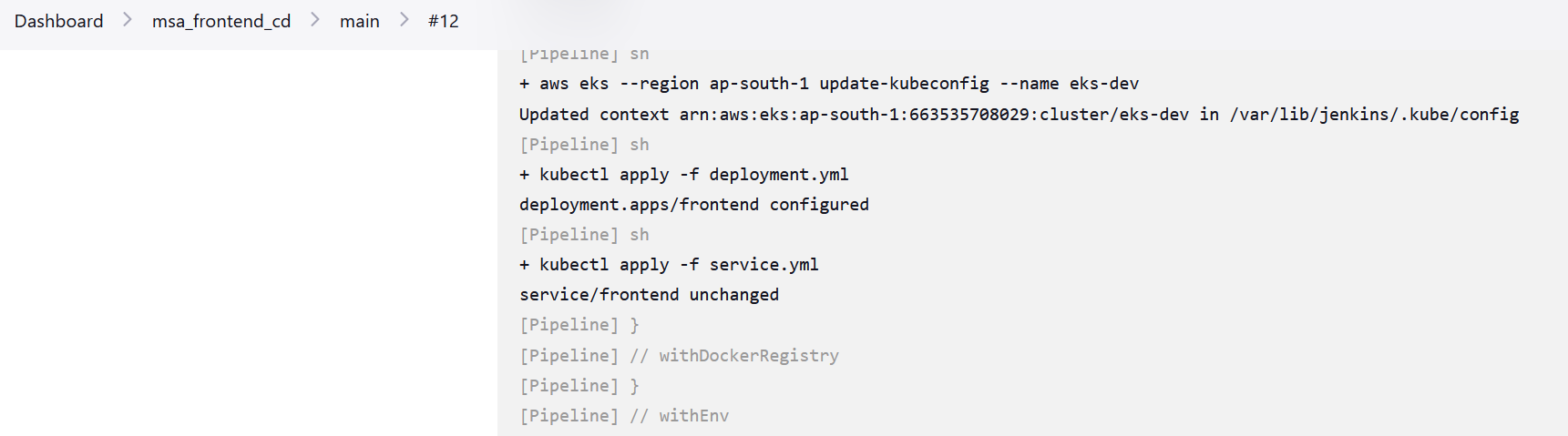
#backend ci



#FrontEnd CI (with Trivy Support)





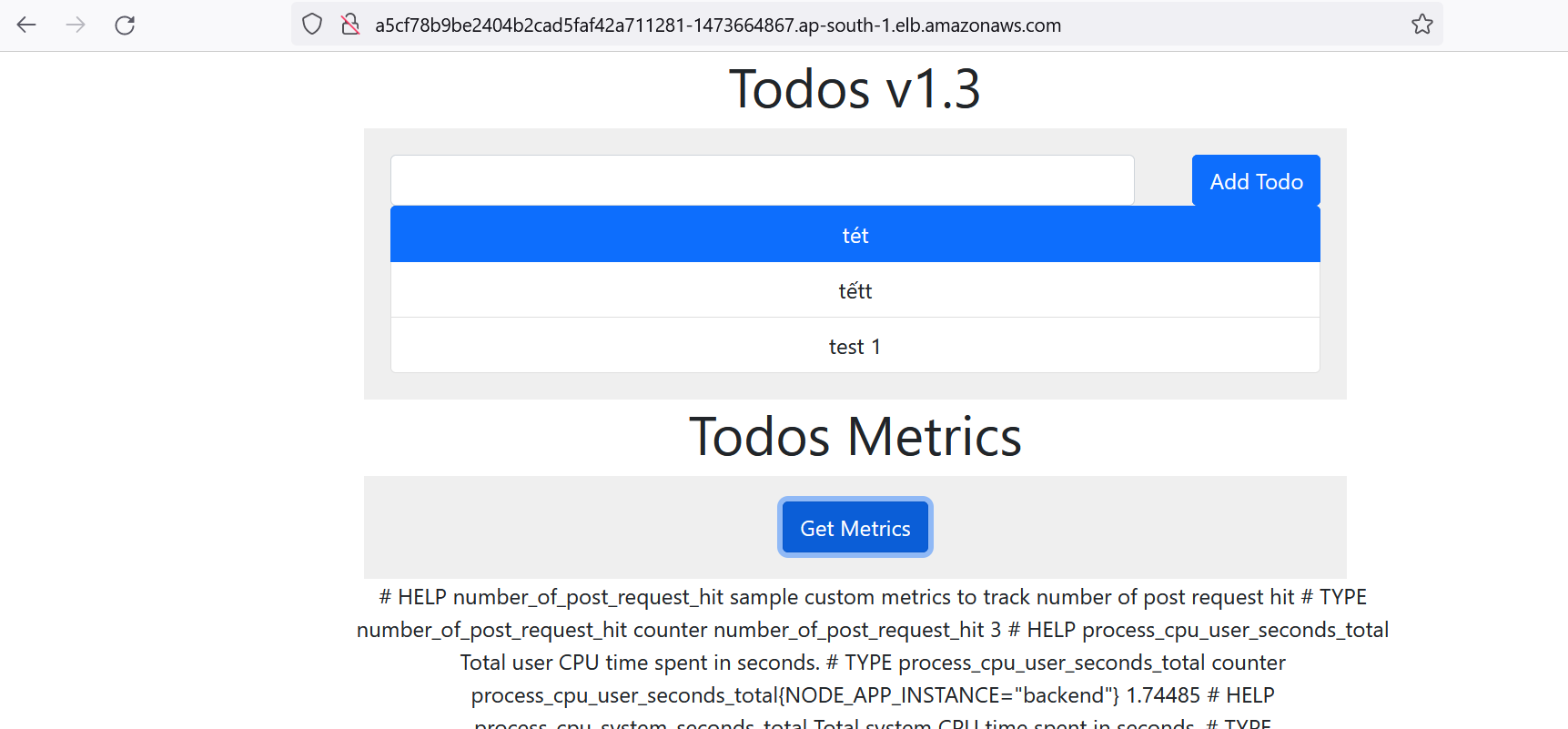


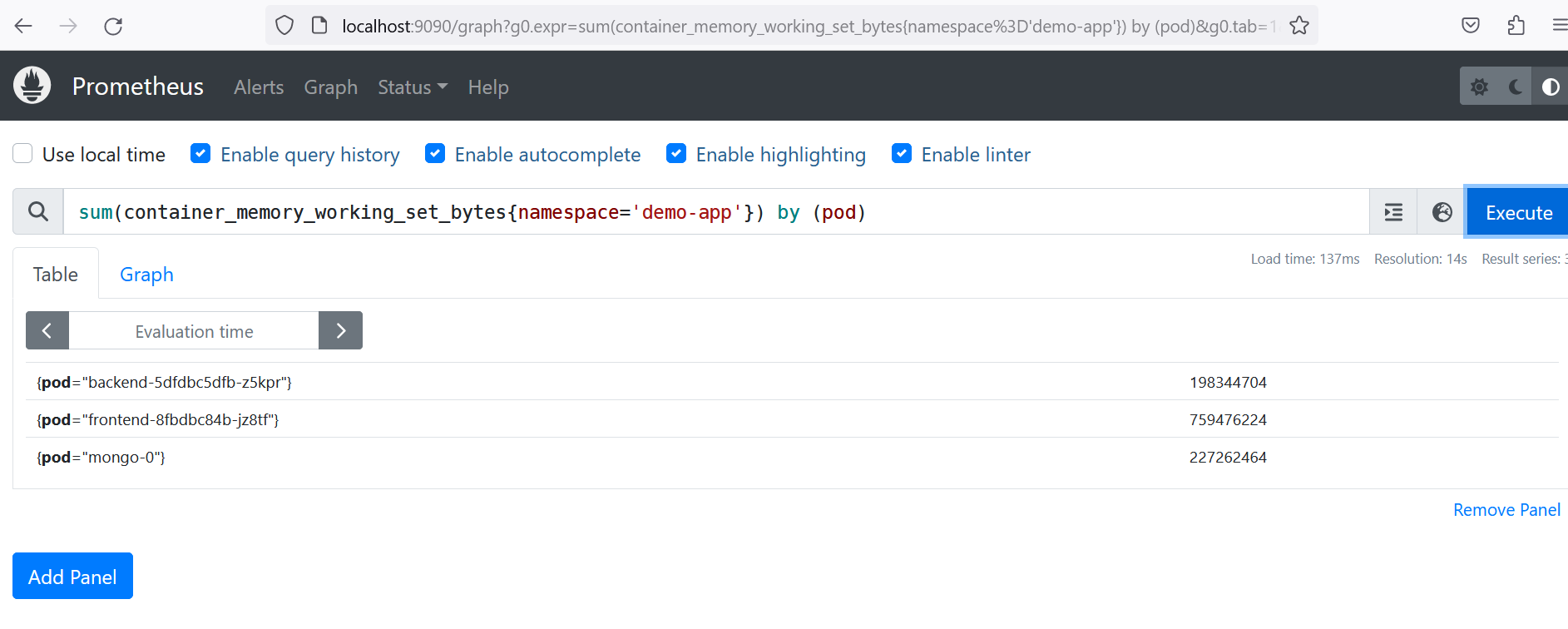
############################# Monitoring with Istio############################

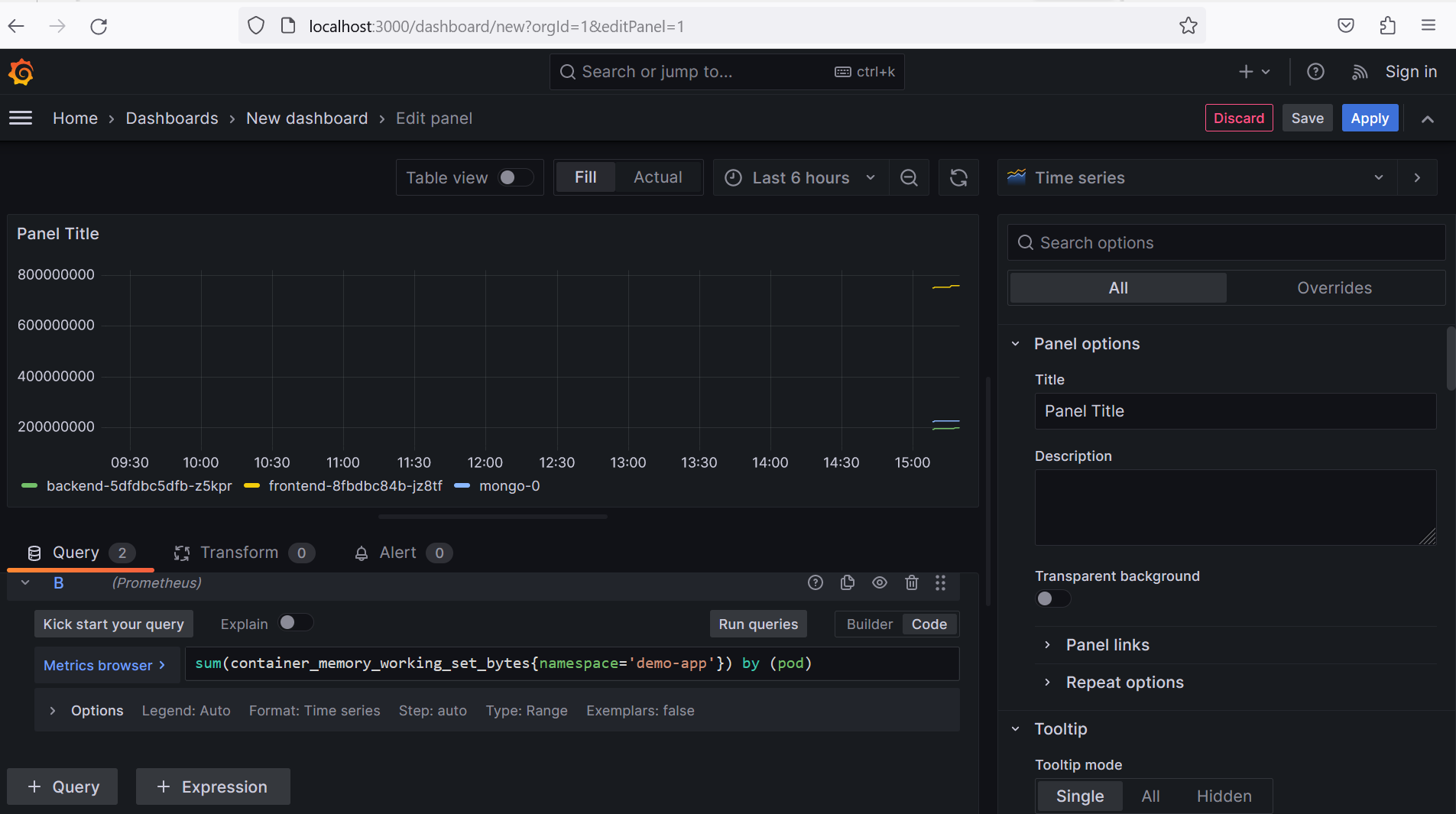
Use Istio to provide observability metrics, which can be visualized and collected using Grafana and Prometheus.

Install Istio then use istio-ingress-gateway to load balance traffic to our frontend todo app

Install Prometheus and Grafana, ensure those dashboard can visualize our pod (frontend, backend, mongo) traffic







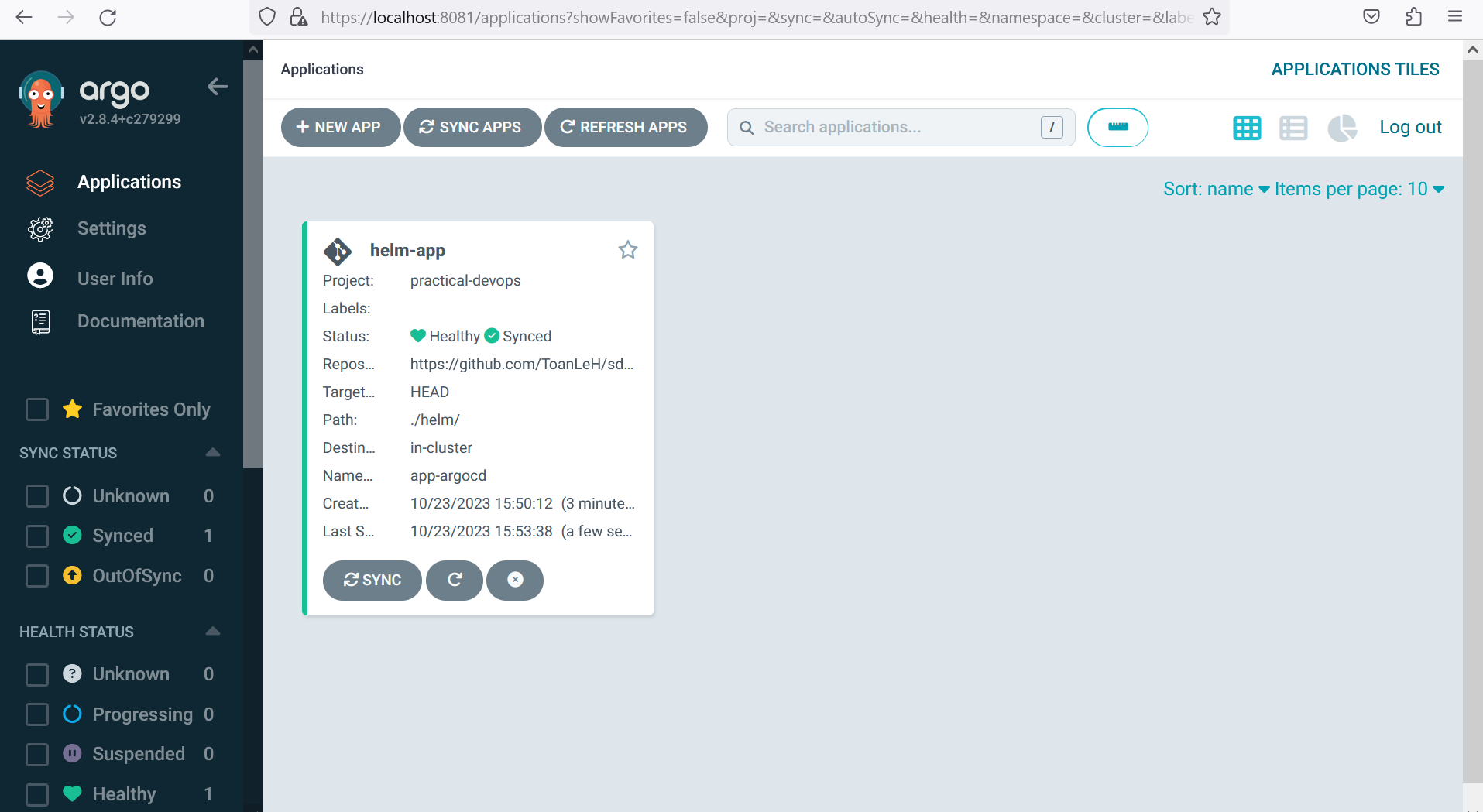
############################# ArgoCD with Helm Chart ############################

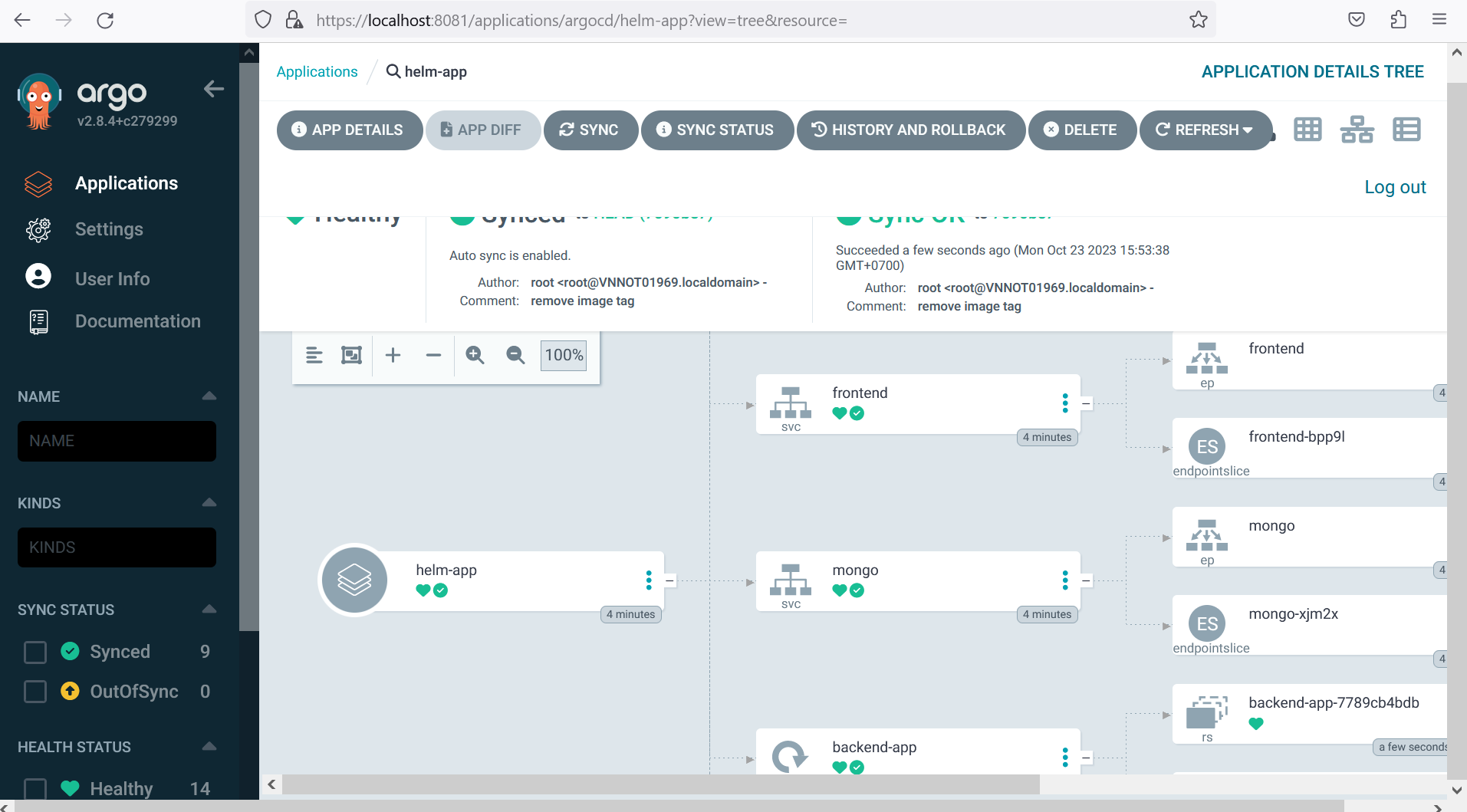
Install ArgoCD and portforward to access UI

A picture containing text, screenshot, software, multimedia software

Description automatically generated

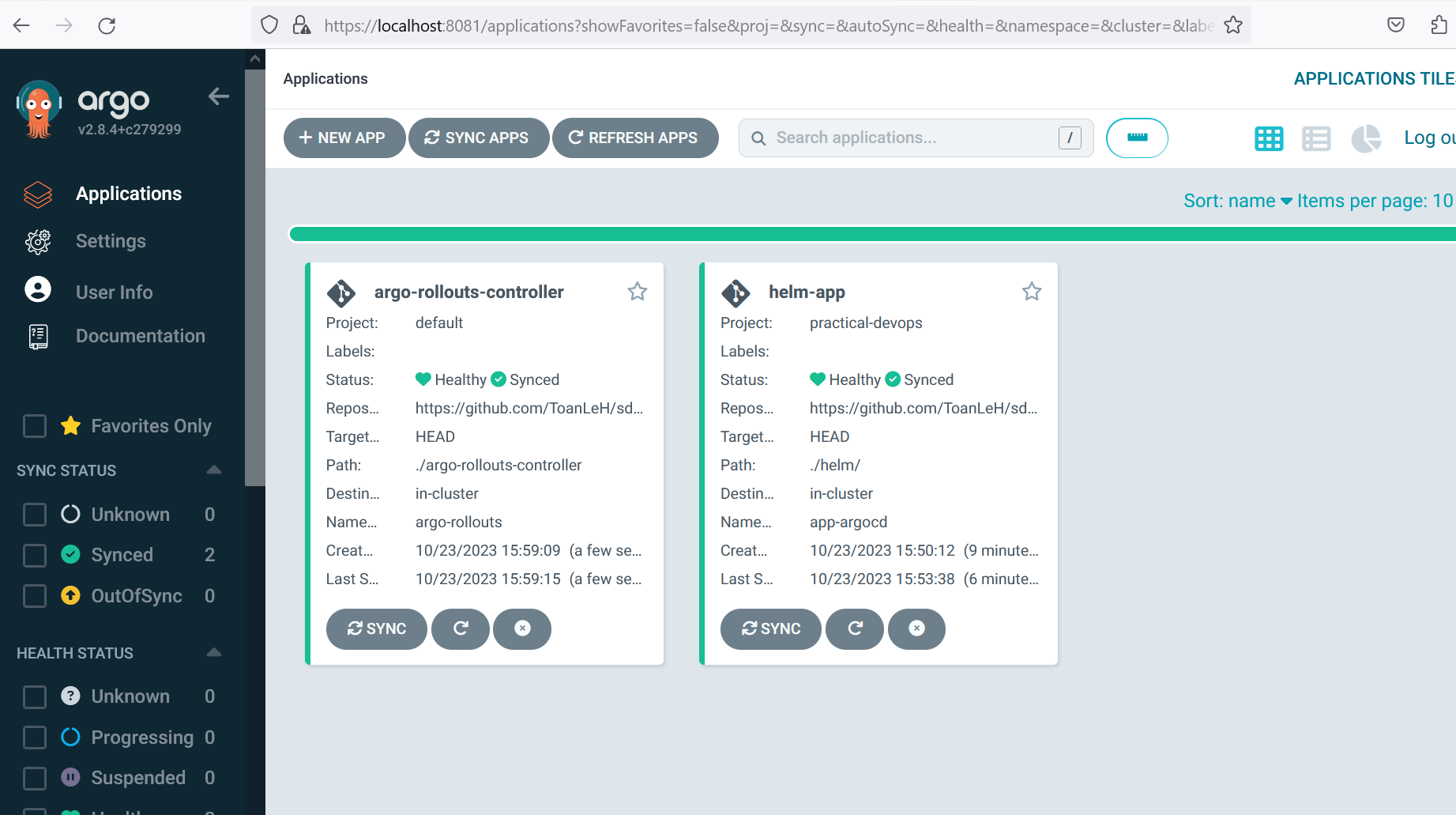
Login to ArgoCD and run command to create the helm application



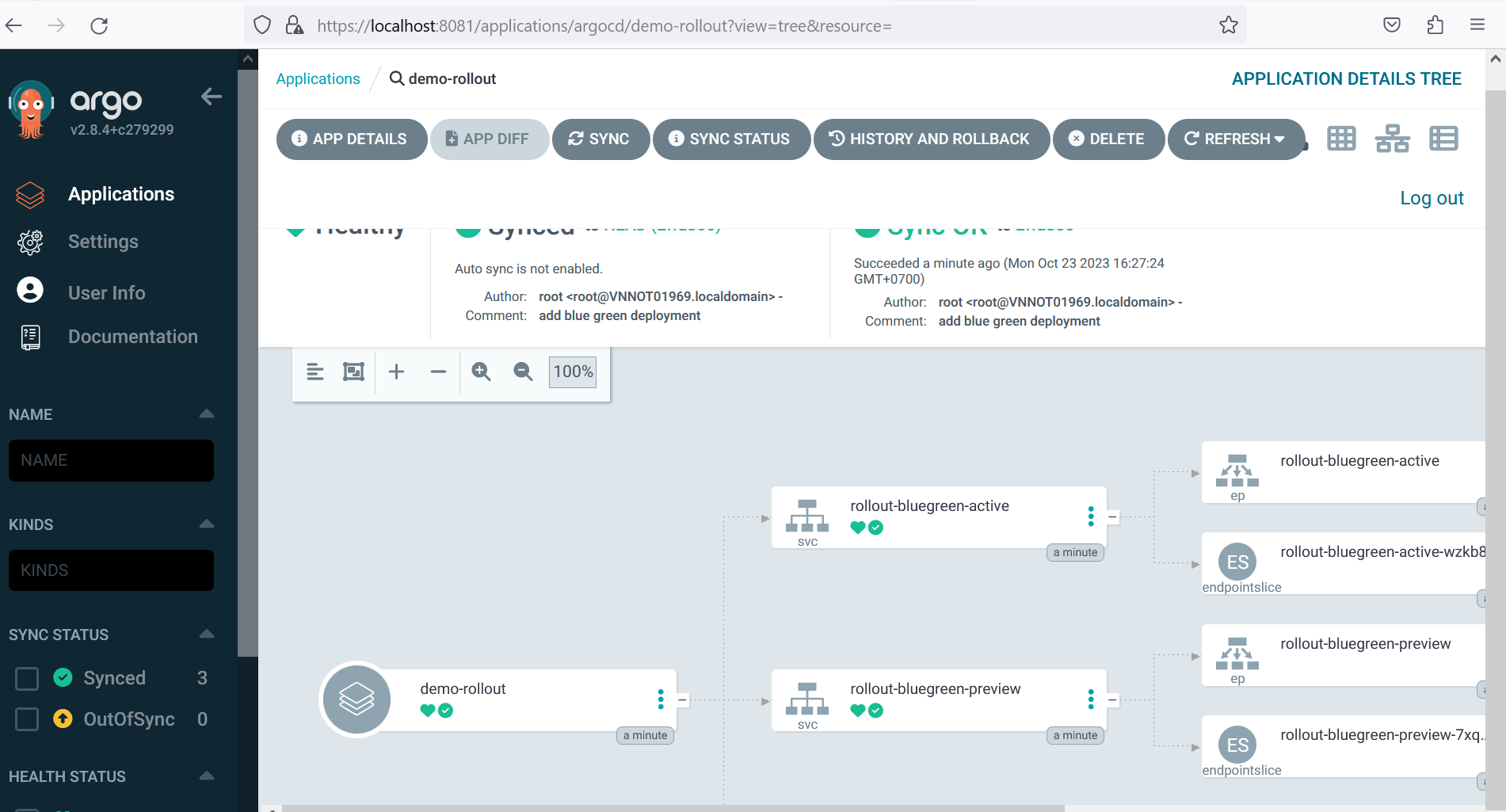


######################### Argo Rollout for Progressive Delivery ############################

Install the Argo Rollouts Controller



Create first deployment and port forward to see the blue version of our app

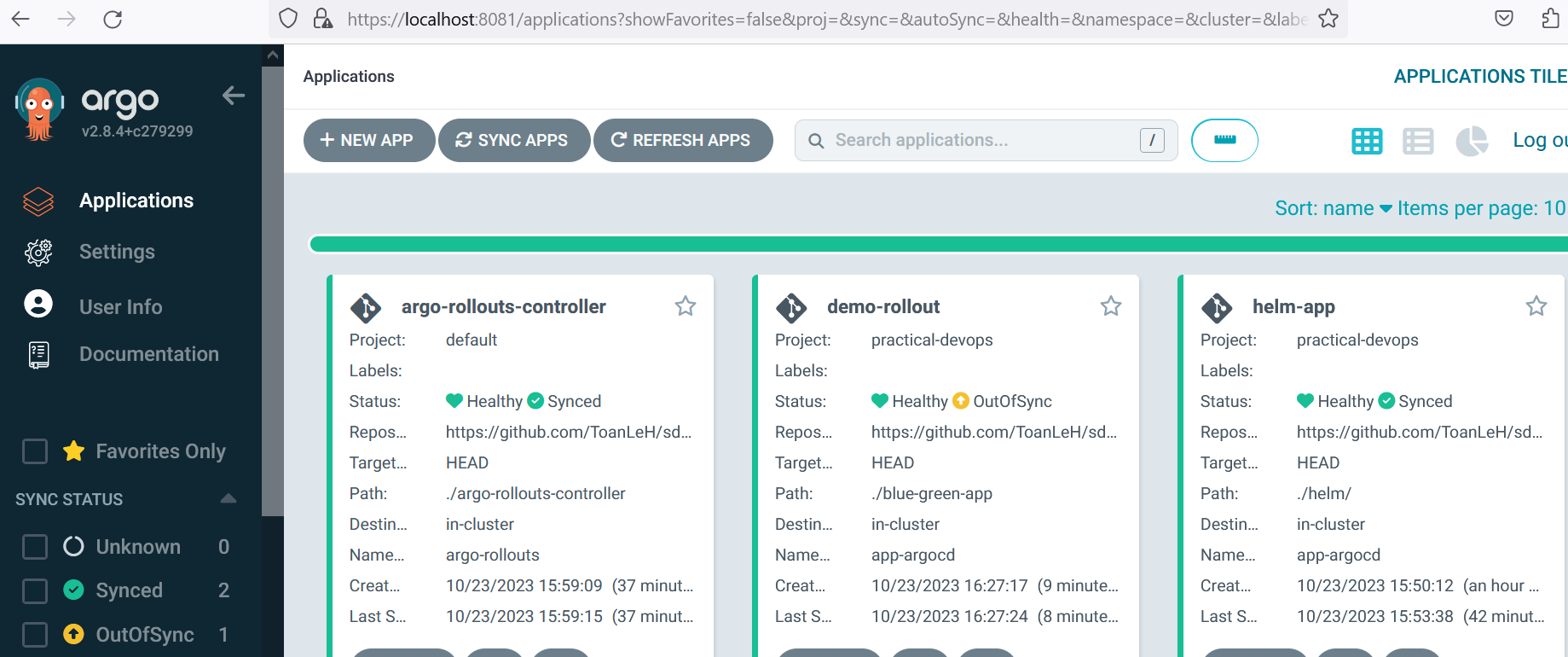




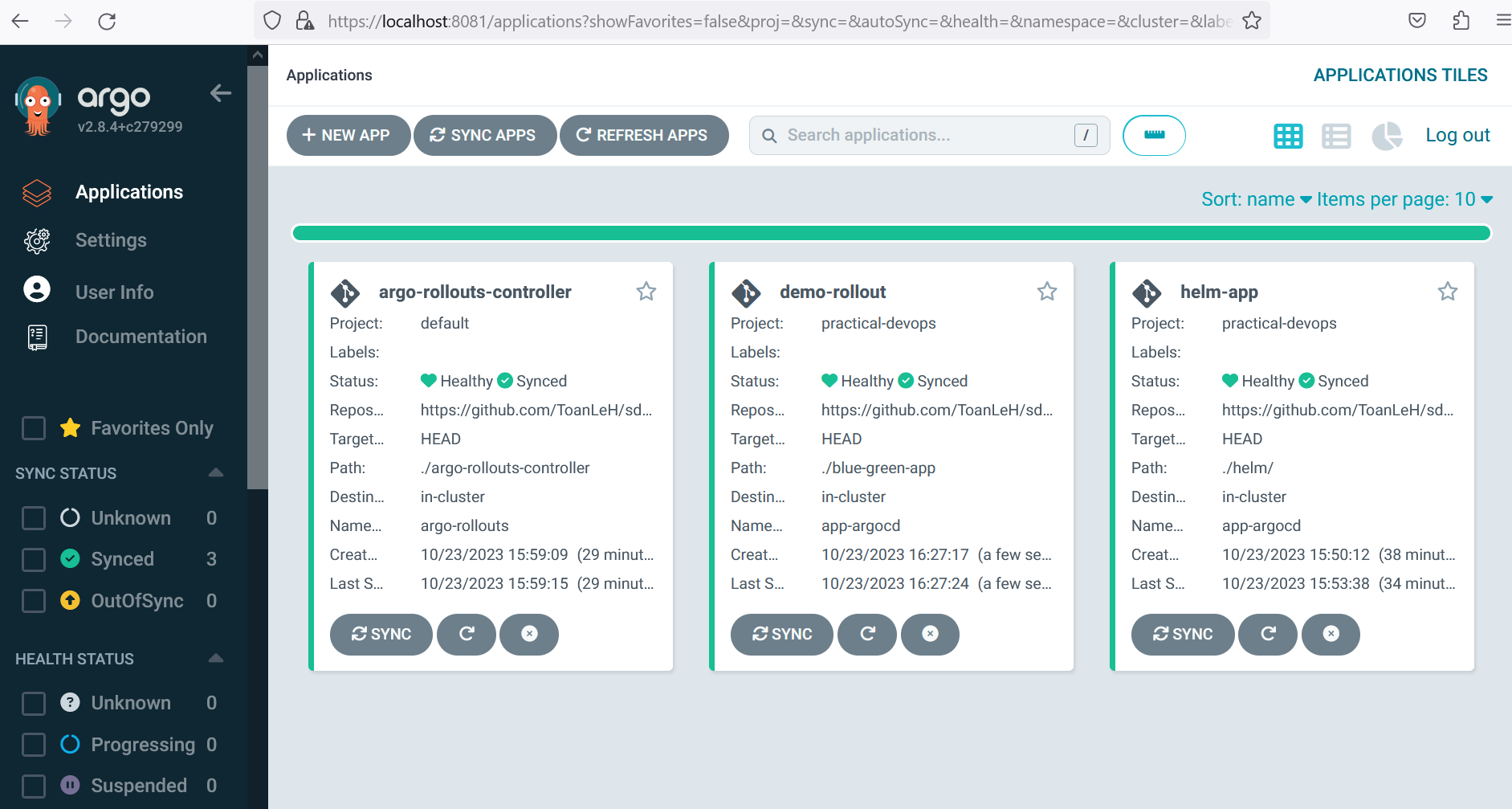
**Blue/Green deployments**

We are now ready to have a blue/Green deployment with the next version. Change the container image of the rollout to the next version with:

image: 663535708029.dkr.ecr.ap-south-1.amazonaws.com/ecr-toanleh-devops-frontend:green

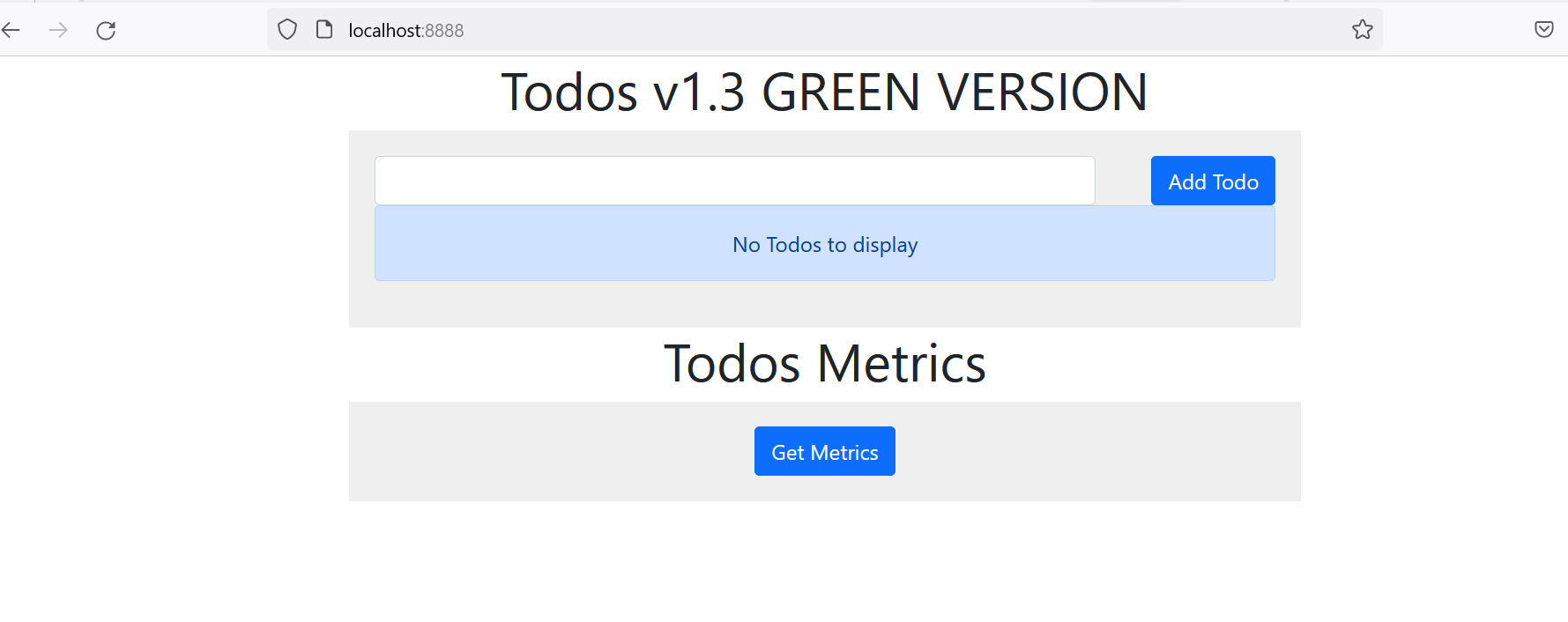


Synced



After you change the image, the following things happen.

* Argo Rollouts creates another replicaset with the new version
* The old version is still there and gets live/active traffic
* ArgoCD will mark the application as out-of-sync
* ArgoCD will also mark the health of the application as "suspended" because we have setup the new color to wait.



Notice that we used the autoPromotionEnabled: true property in the definition of the rollout.

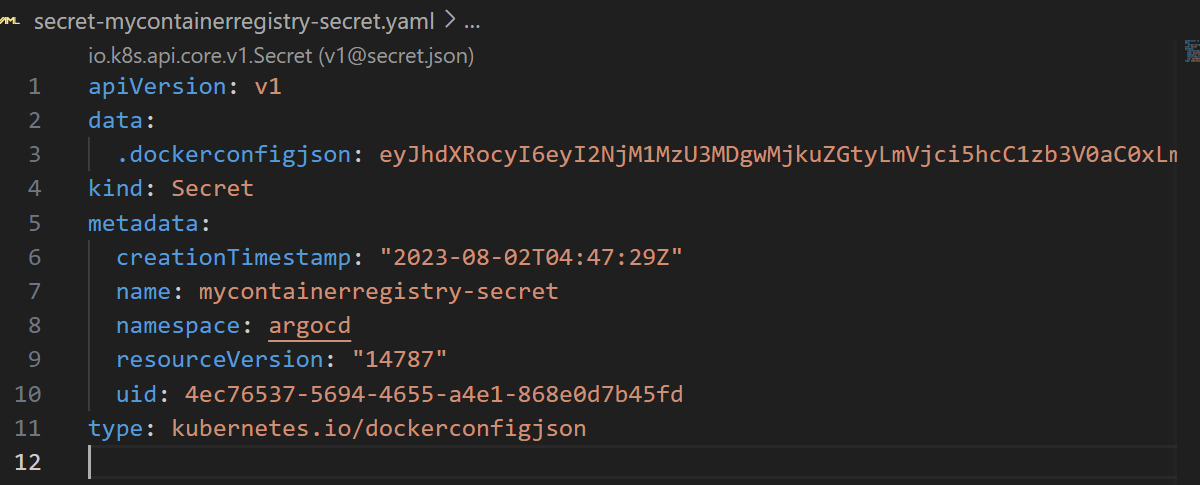
############################Install Argo CD Image Update ############################

Install Argo CD image Update

kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj-labs/argocd-image-updater/stable/manifests/install.yaml

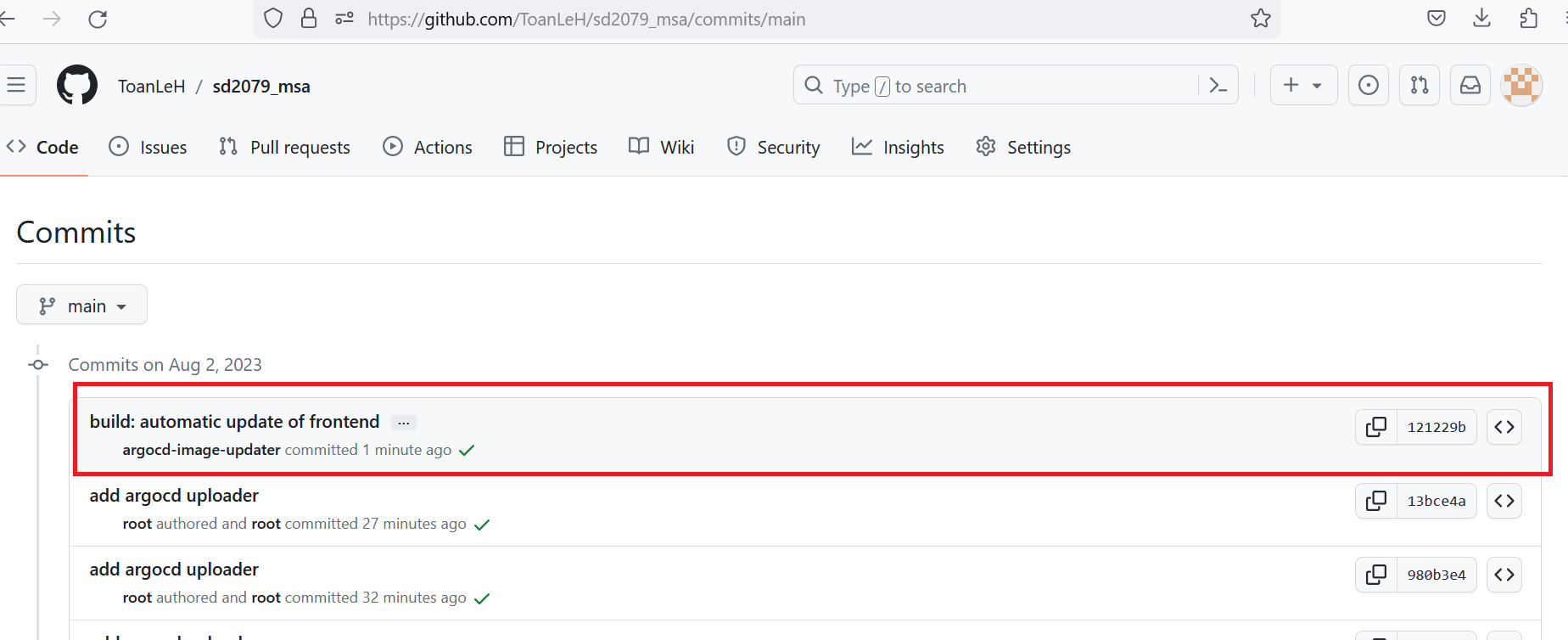
#Create pull secret

kubectl -n argocd create secret docker-registry mycontainerregistry-secret --docker-server=<SERVER\_HERE> --docker-username=<USERNAME> --docker-password=<PASSWORD> -o yaml --dry-run=client | kubectl -n argocd apply -f –

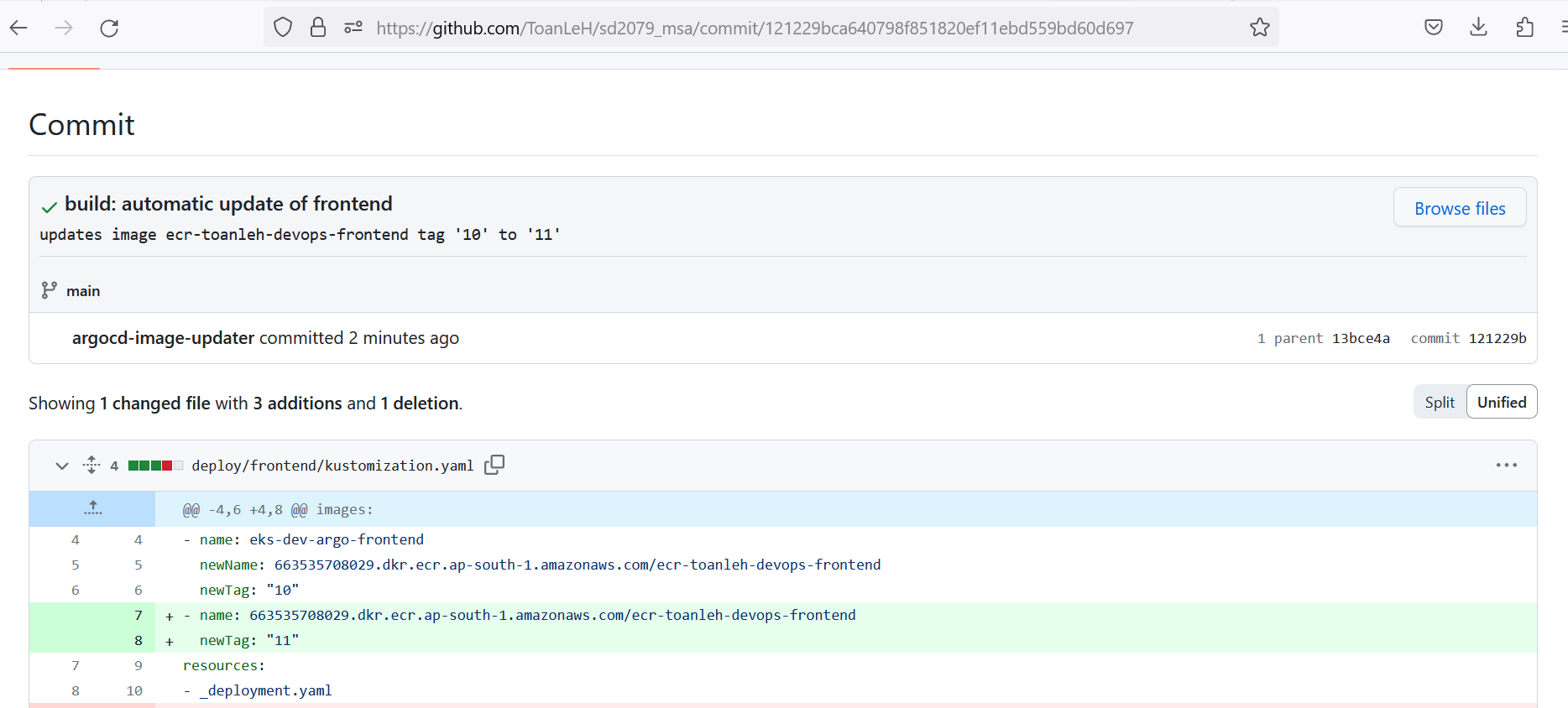


#Modify Config map to connect to container registry

|  |  |
| --- | --- |
| kubectl edit configmaps --namespace argocd argocd-image-updater-config |  |
| registries.conf: |  registries:  - name: Elastic Container Registry  prefix: <YOUR\_ECR\_HERE>  api\_url: https:// <YOUR\_ECR\_HERE>  credentials: pullsecret:argocd/mycontainerregistry-secret  #Modify app and trigger CI build for new version |  |
|  |  |



#wait a minutes you will see argocd-image-updater modify the file and commit to git repo



#And ArgoCD will auto sync the changes to our cluster, point to the app endpoint you can see the changes are synced

