############################# Pre-requisite ####################################

# # AWS CLI – On Windows

# # kubectl – On Windows using chocolatey

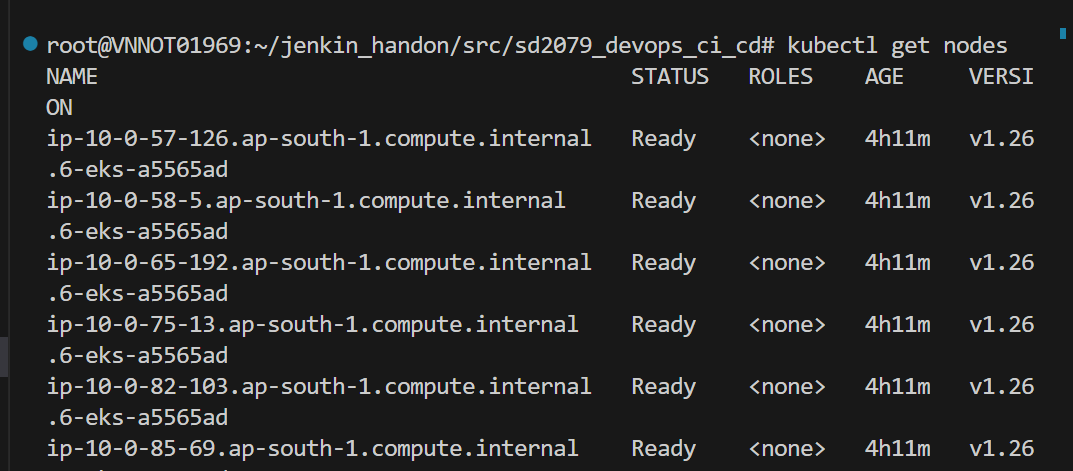
# # eksctl - On Windows using chocolatey

# # argocd-cli

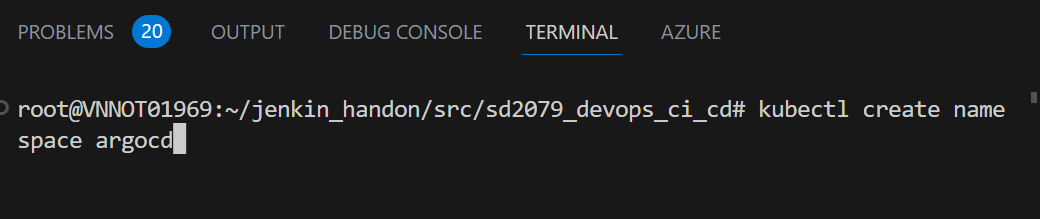
# # Reuse CI pipeline from previous documents SD2079\_CICD\_AKS.docx

############################# Install Argo CD on EKS cluster #############################

1. Check if kubectl is working as expected   
   kubectl get nodes

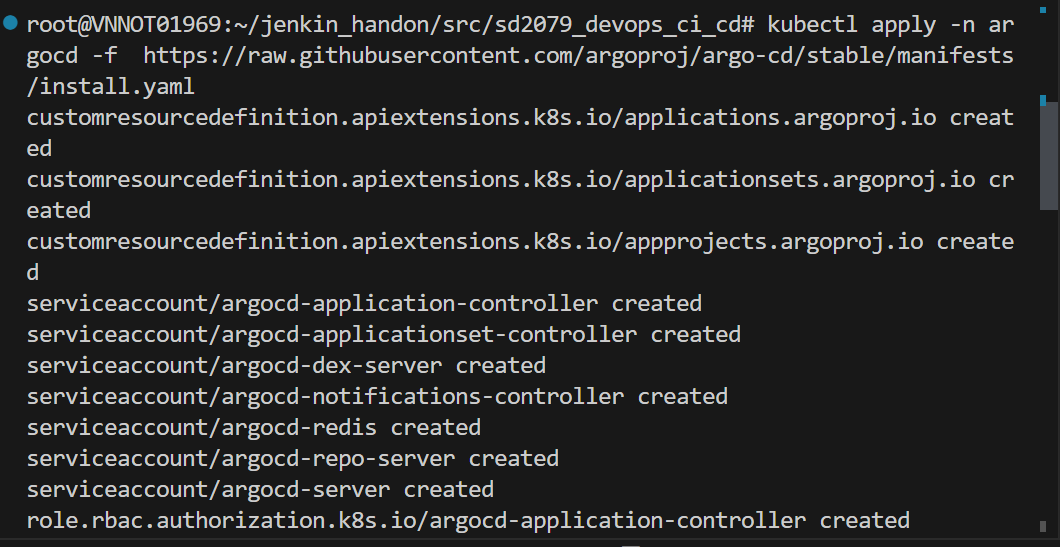


1. Create namespace  
   kubectl create namespace argocd



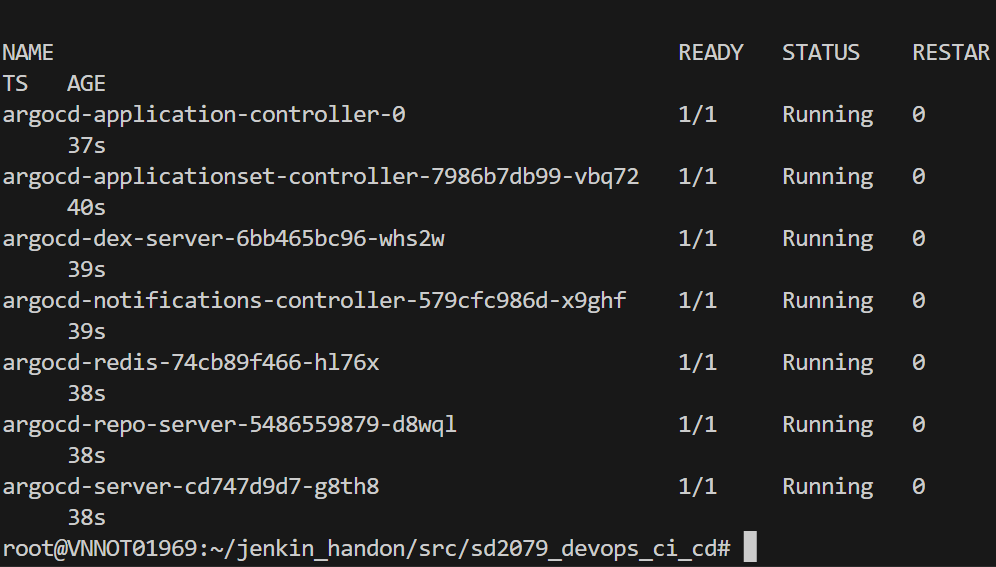
1. Run the Argo CD install script provided by the project maintainers

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



1. Check the status of your Kubernetes pods.

kubectl get pods -n argocd

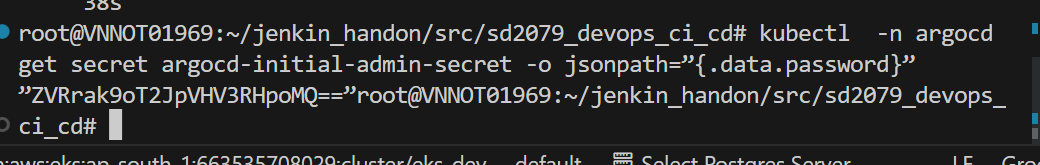


############################# Access to Argo CD ###########################

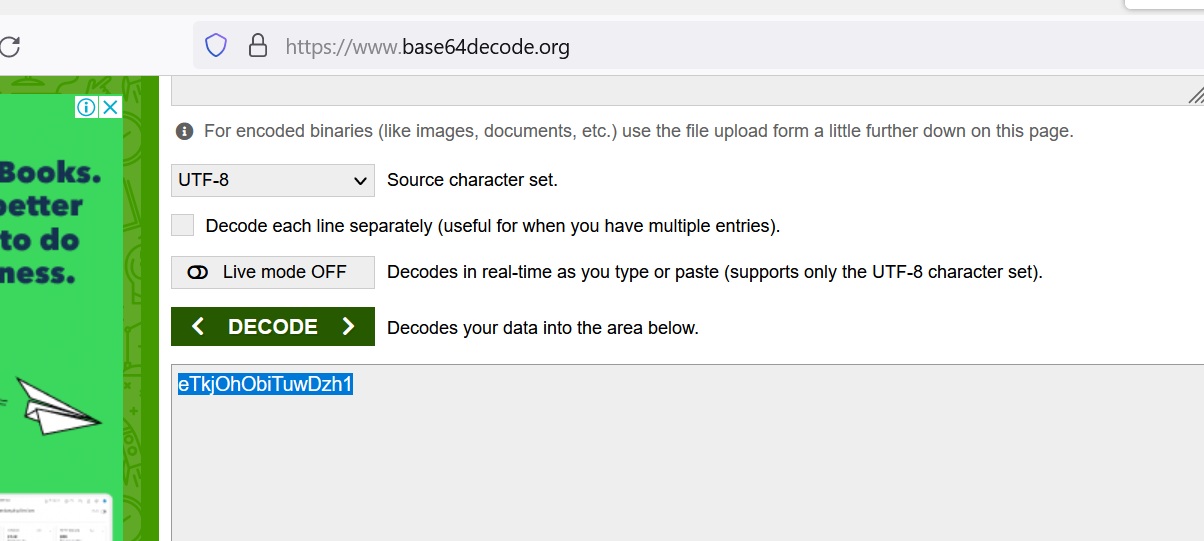
# # Forwarding Ports to Access Argo CD

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}”



eTkjOhObiTuwDzh1



Forward those to arbitrarily chosen other ports, like 8080

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



Access from internet from browser

<http://localhost:8080>

use id: admin

password: <base64 decoded password>

A picture containing text, screenshot, software, multimedia software

Description automatically generated

A screenshot of a web browser

Description automatically generated with low confidence

########################### Create an application in Argo CD using GUI #####################

1. Create a namespace for application ex: app-argocd.( this is optional, you can use default or any namespace you want)

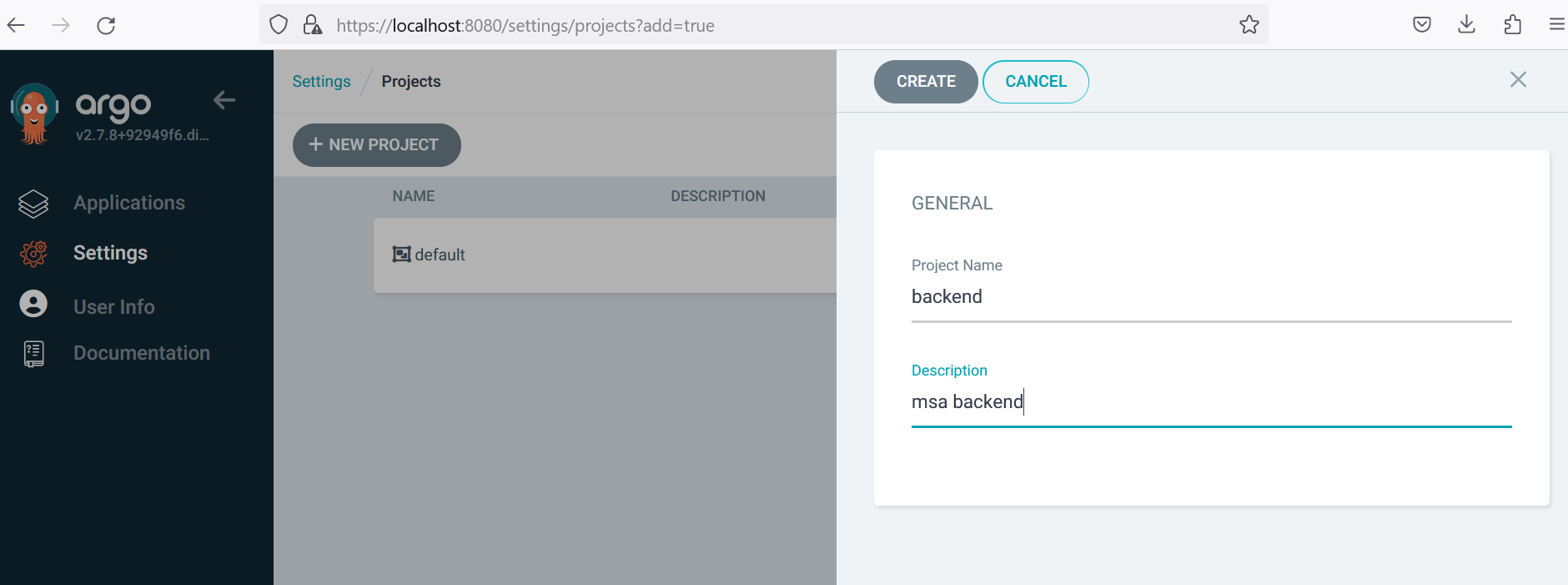
kubectl create namespace app-argocd



1. Create a new project.

A screenshot of a computer

Description automatically generated



Add source repositories.

A white rectangular object with a black border

Description automatically generated

Add destination.

A screenshot of a computer

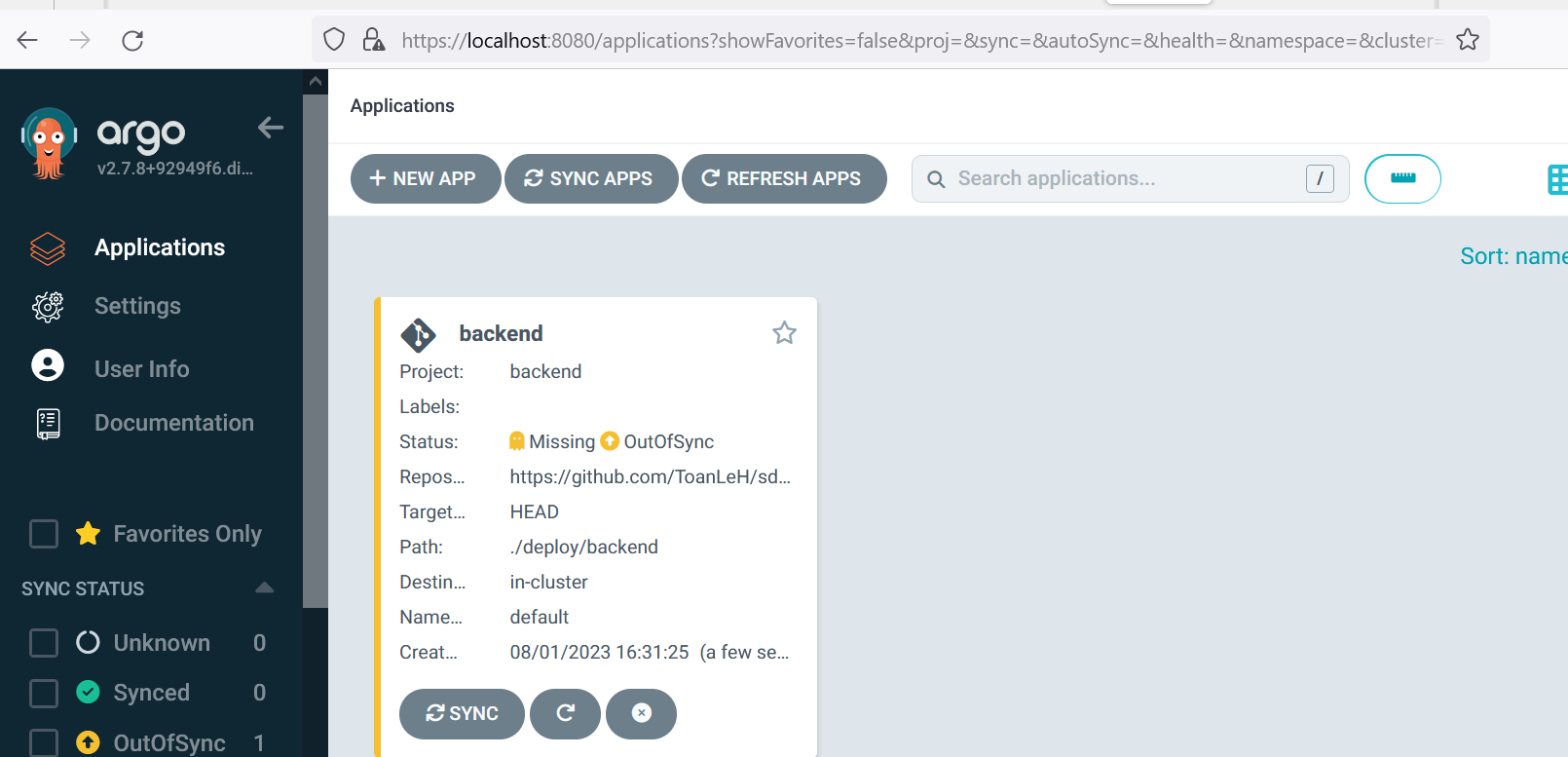
Description automatically generated

Add cluster resource allow list

A screenshot of a computer

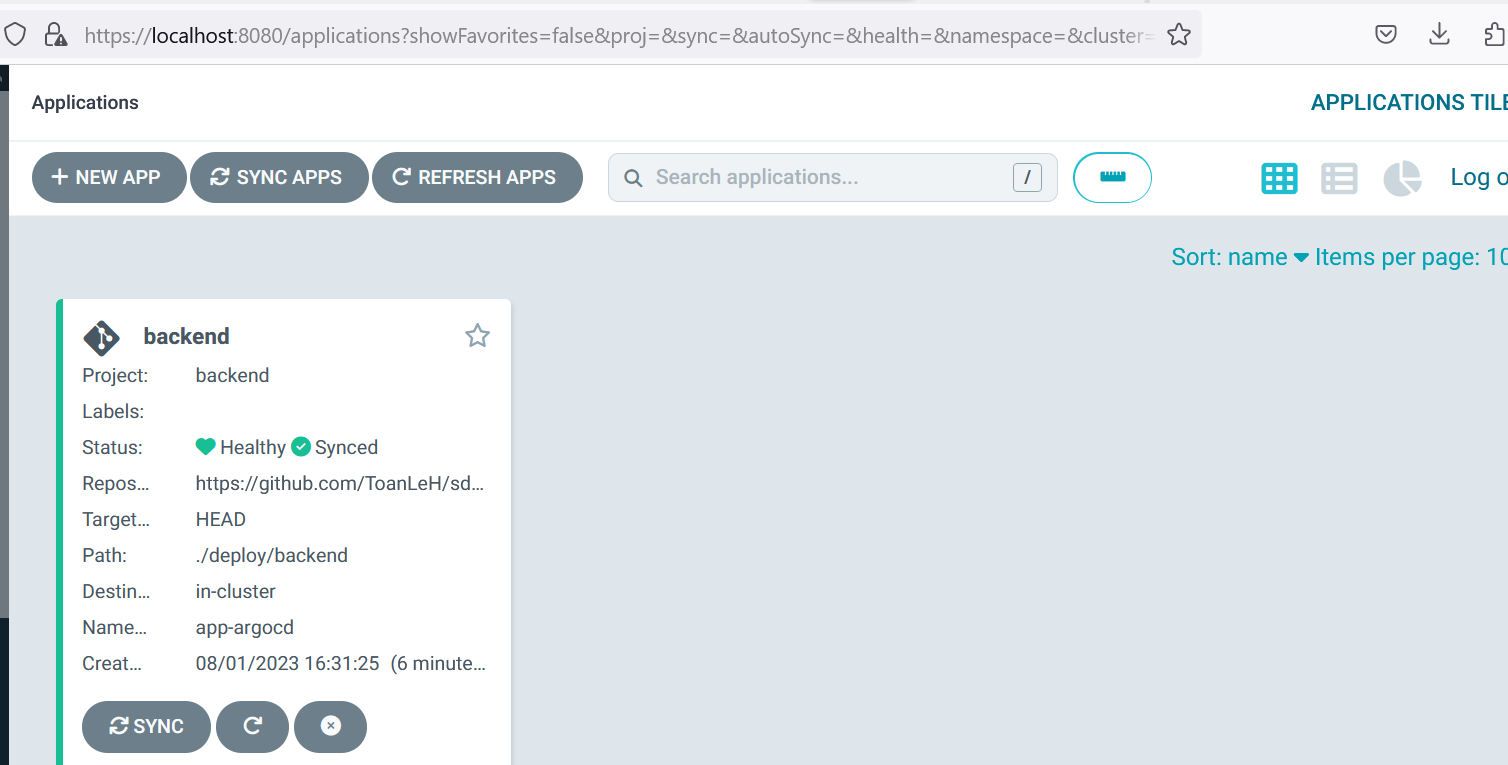
Description automatically generated

1. Click the "New app" button on Argo CD UI and fill the following details:
   * application name: backend
   * project: backend
   * repository URL: https://github.com/ToanLeH/sd2079\_msa ( Your Github)
   * path: ./deploy/backend
   * Cluster: https://kubernetes.default.svc (this is the same cluster where ArgoCD is installed)
   * Namespace: app-argocd
   * Leave all the other values empty or with default selections. Finally click the Create button. The application entry will appear in the main dashboard. Click on it.

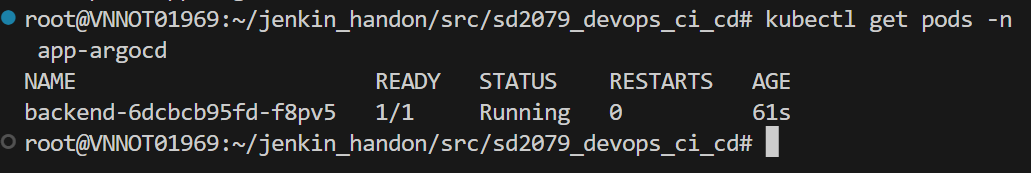


When the application created, the status should be OutOfSync because we are using Manual in Sync Policy.

1. Syncing the application

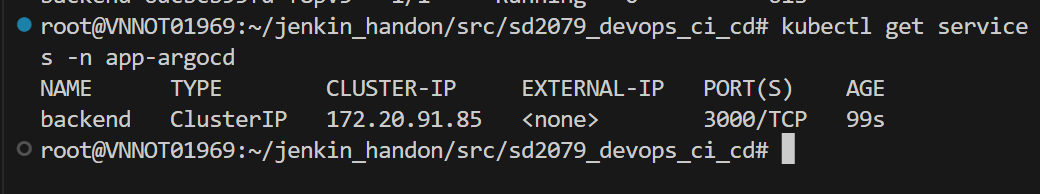


* Check pods  
  kubectl get pods -n app-argocd

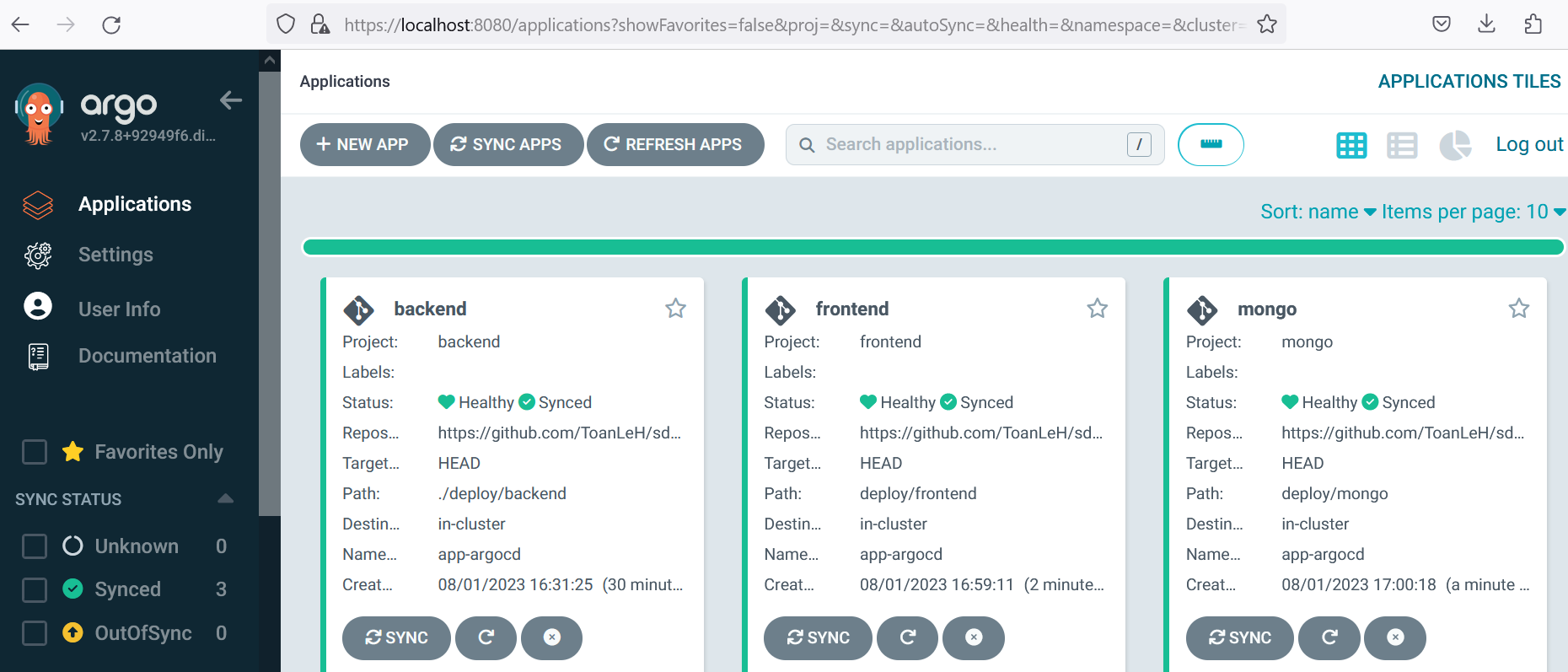


* Check services.

kubectl get services -n app-argocd



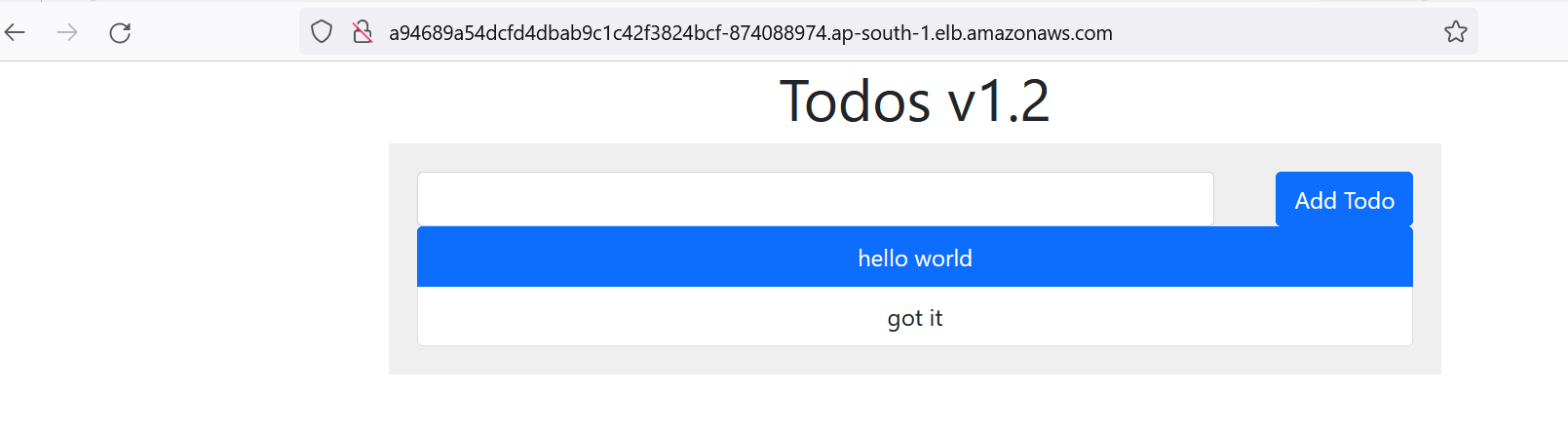
1. Do the same for frontend and mongo



1. Run the application.

Point the nginx to app-argocd frontend service

http://a94689a54dcfd4dbab9c1c42f3824bcf-874088974.ap-south-1.elb.amazonaws.com/



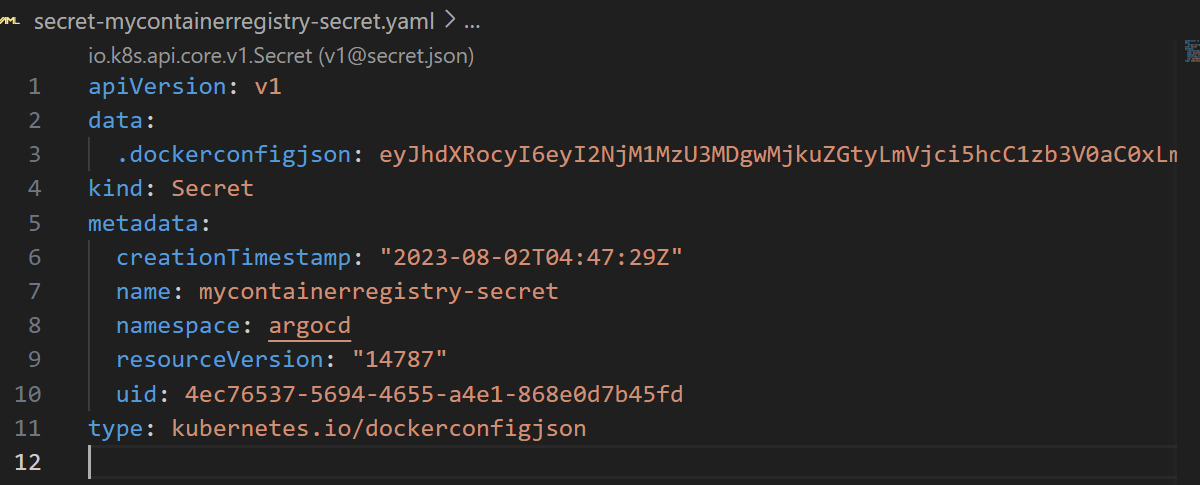
############################Install Argo CD Image Update ############################

Install Argo CD image Update

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

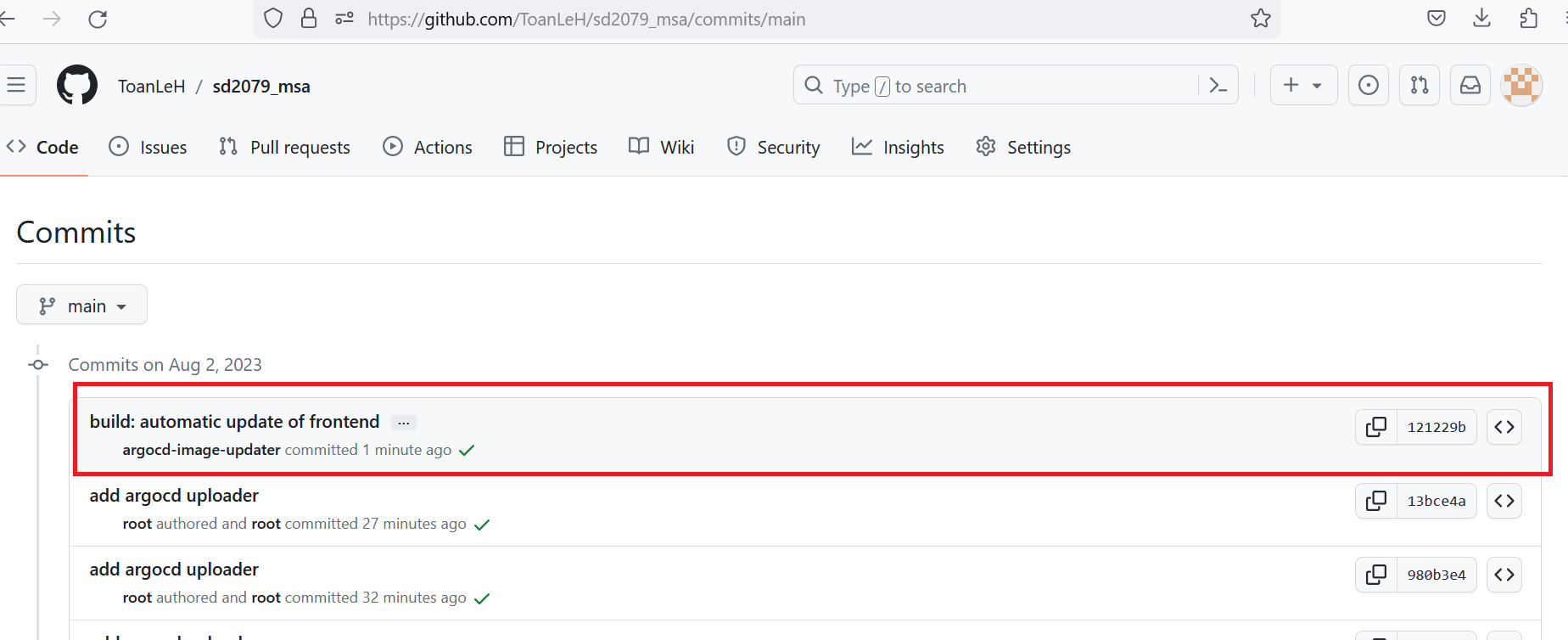
#Create pull secret

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 –

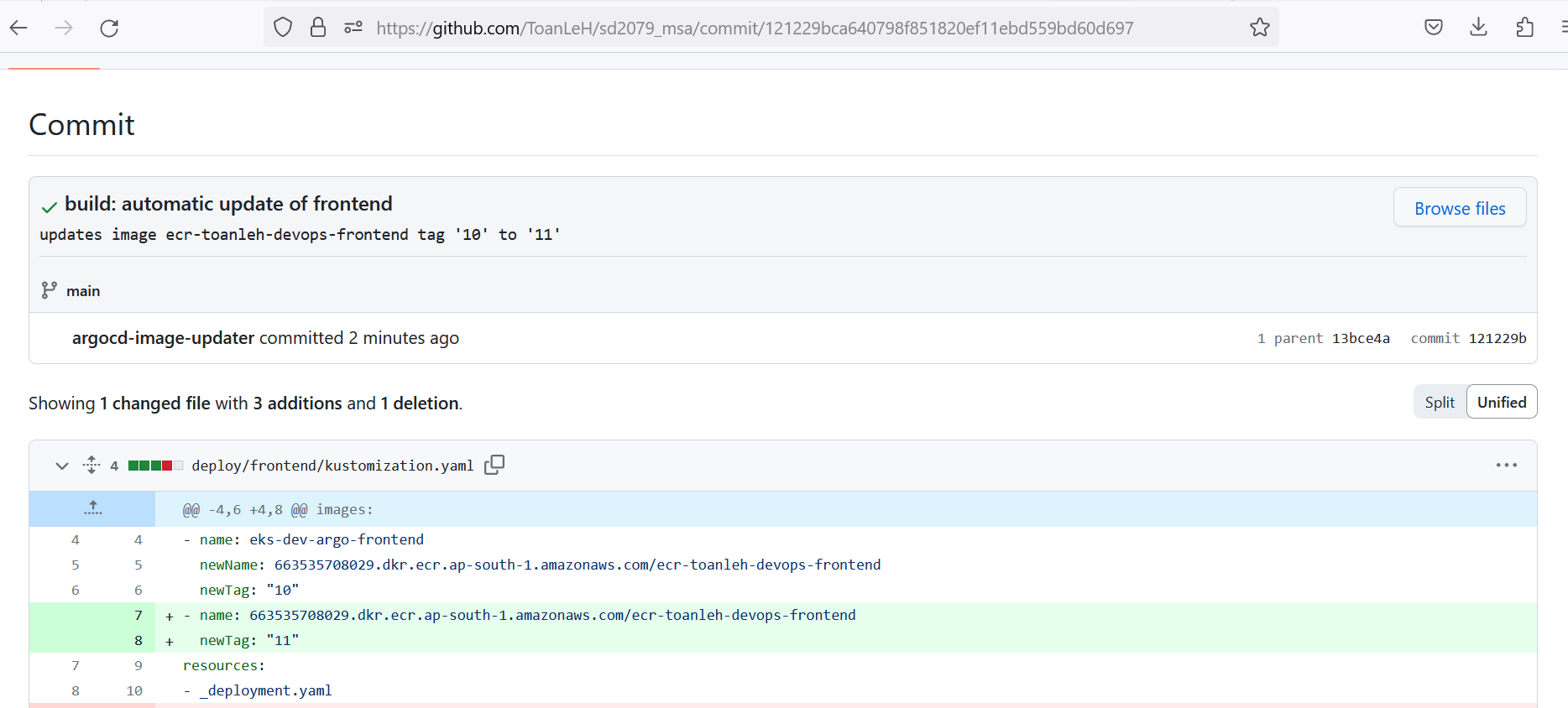


#Modify Config map to connect to container registry

|  |  |
| --- | --- |
| 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

