1. **Source code:**

Create Application repo:

<https://github.com/ngaito/sd1462_msa.git>

Create Infrastructure repo:

<https://github.com/ngaito/sd1462_azure_infrastructure.git>

1. **Provision Azure resources with Terraform**

Clone Infastructure repo and deploy Azure resources using Terraform:

Install Azure CLI:

*curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash*

Login to Azure using credentials:

*az login*

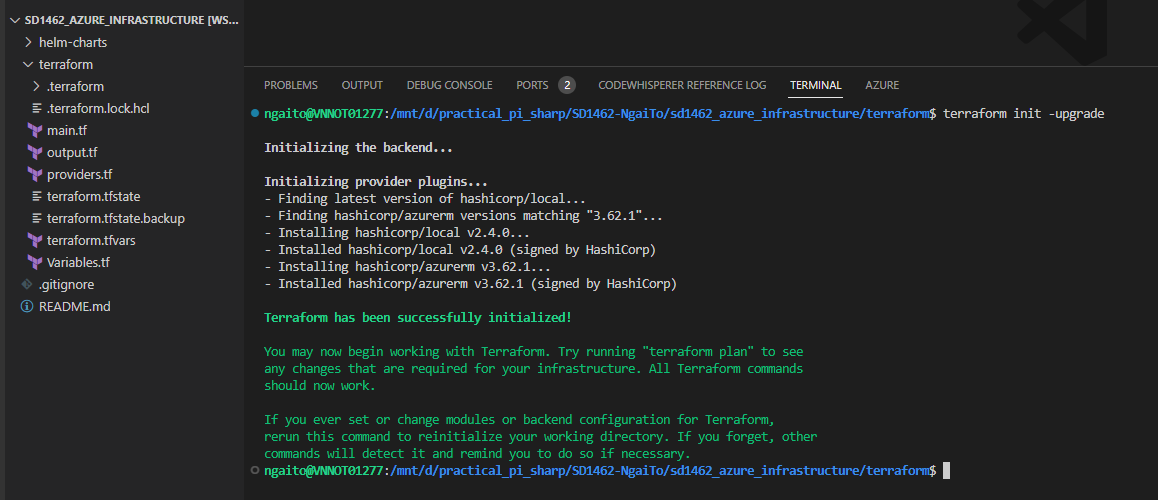


Install Terraform:

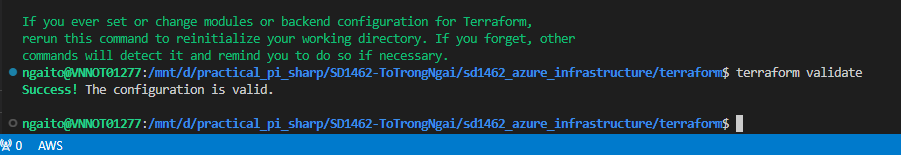
*sudo snap install terraform --classic*

*terraform version*

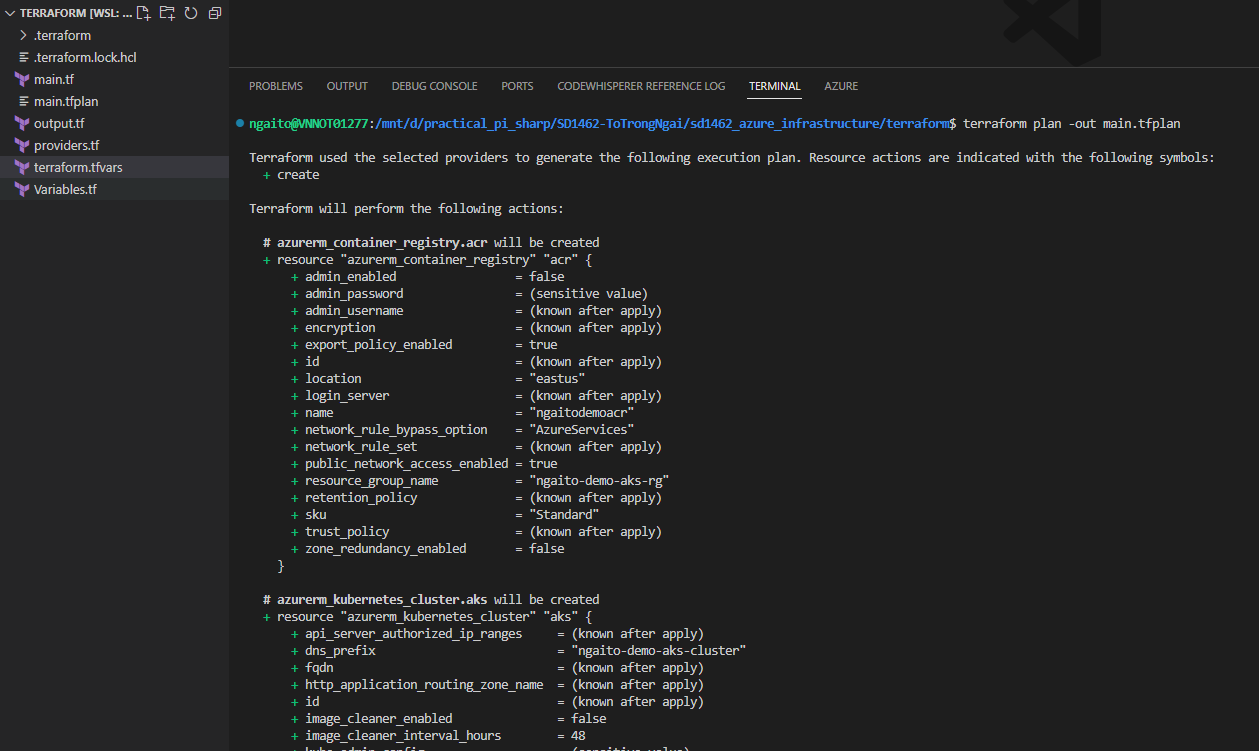
*terraform init -upgrade*



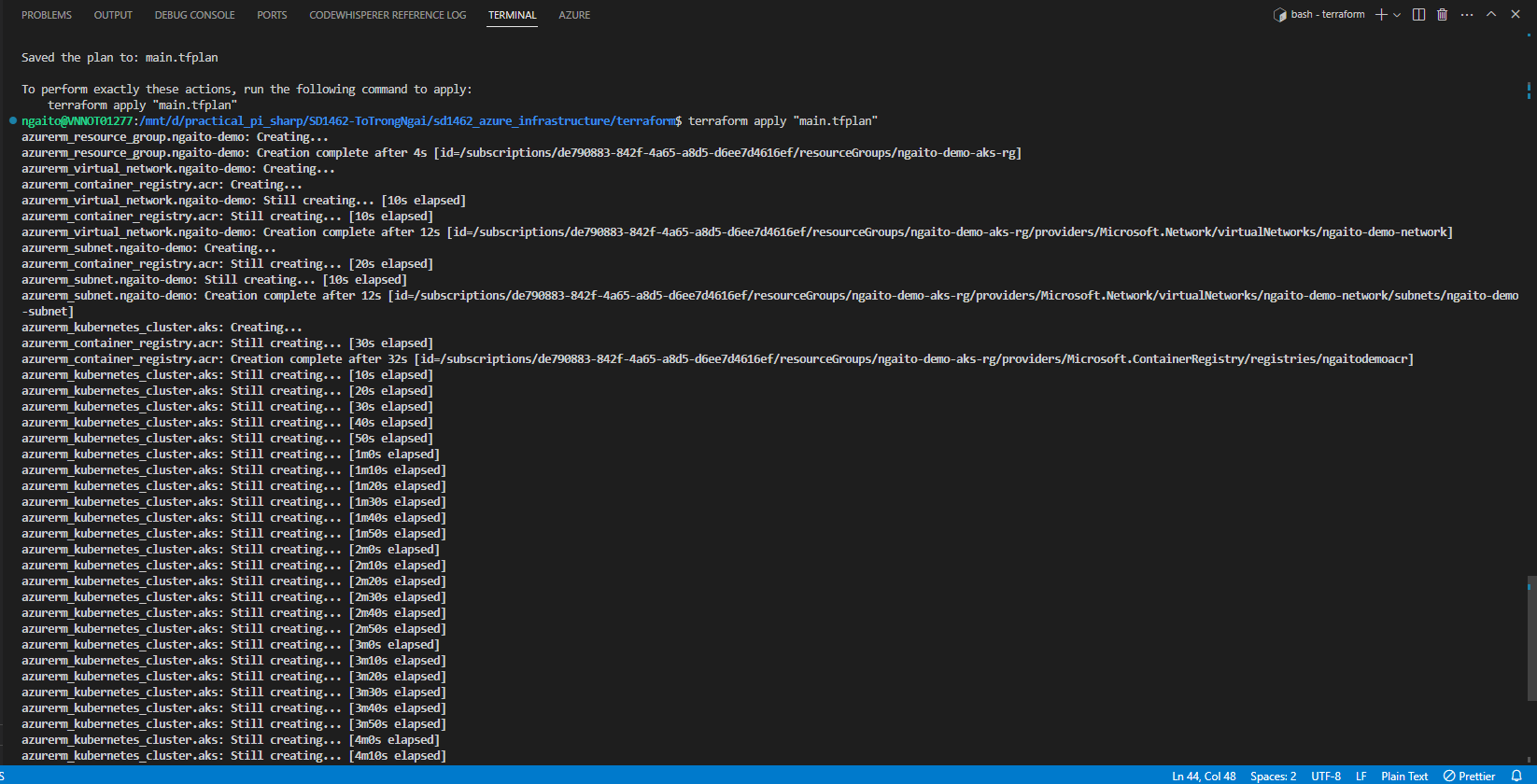
*terraform validate*



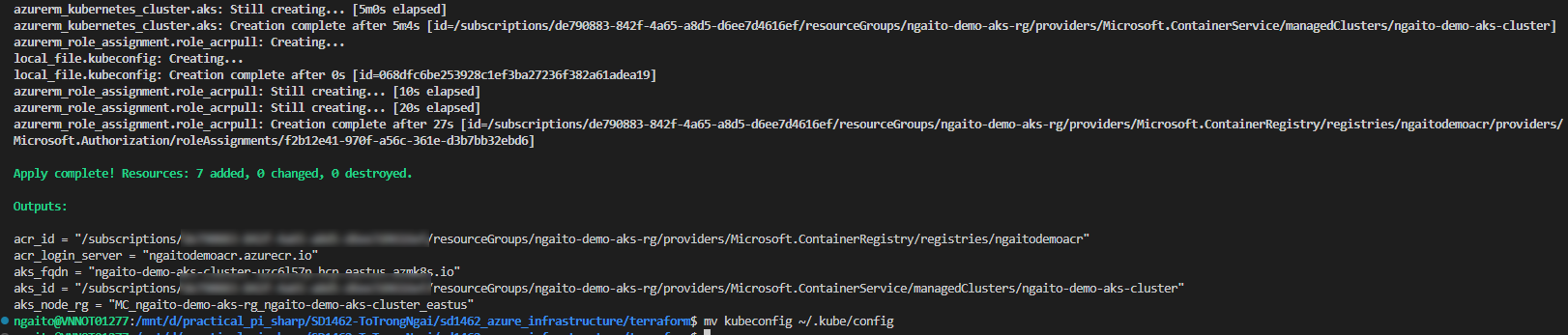
*terraform plan -out main.tfplan*



*terraform apply "main.tfplan"*



Write down the cluster name and resource group

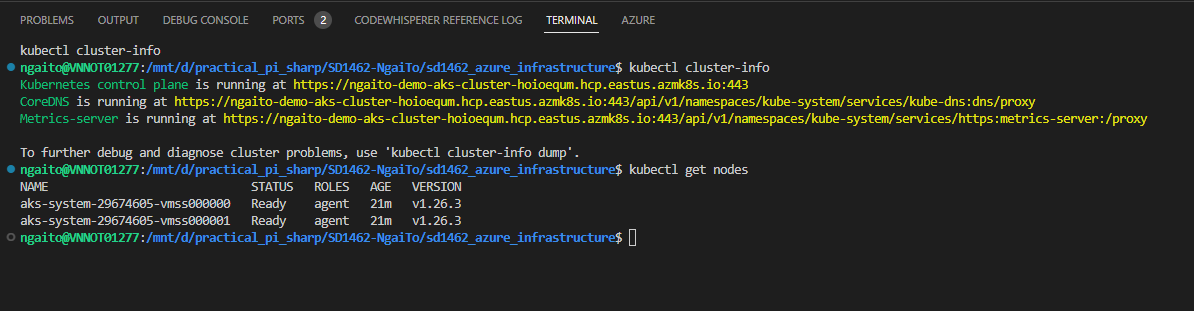


Move the generated Kubeconfig file to ~/.kube/config

*mv kubeconfig ~/.kube/config*

To verify if worker nodes are created, use the kubectl get nodes command to return a list of the cluster nodes:

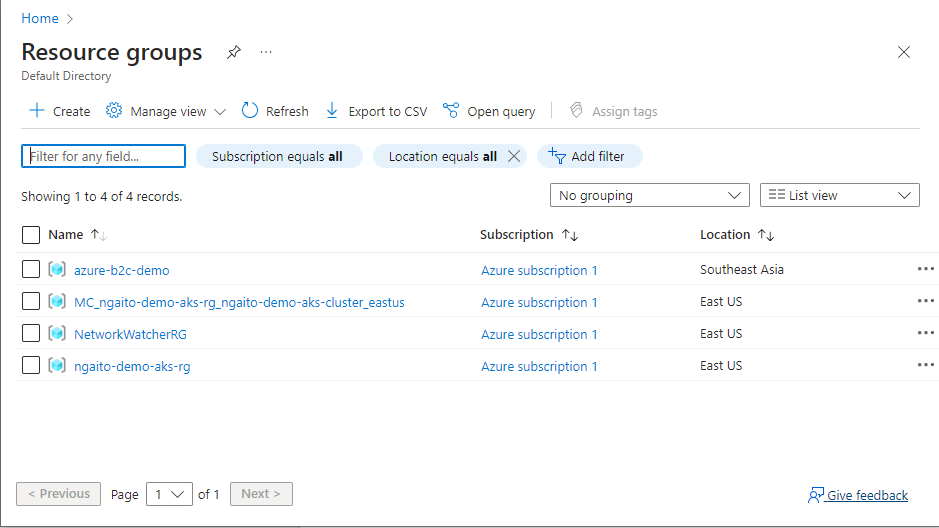
*kubectl get nodes*

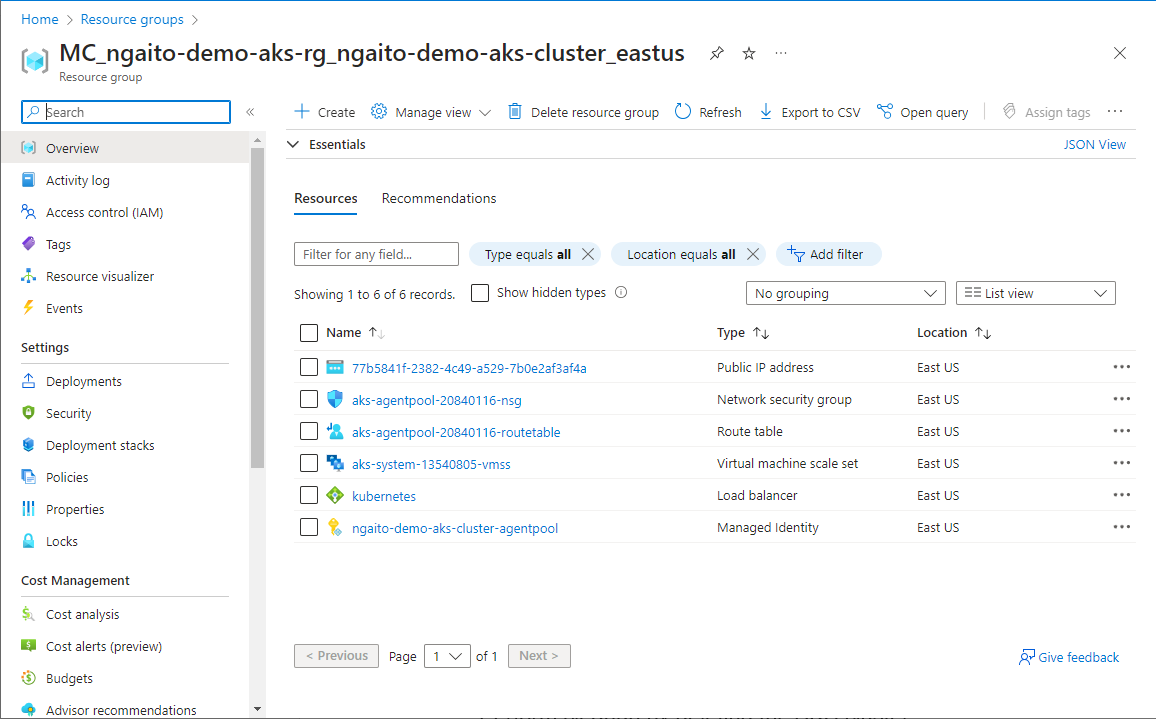


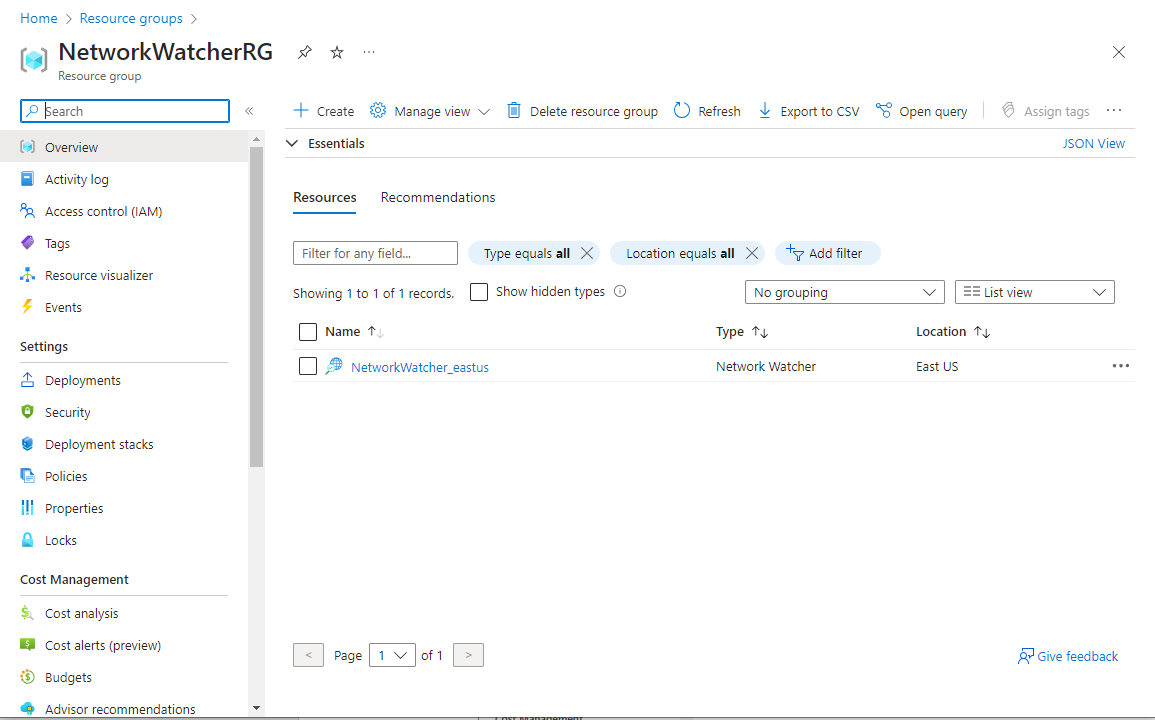
*kubectl get all --all-namespaces*

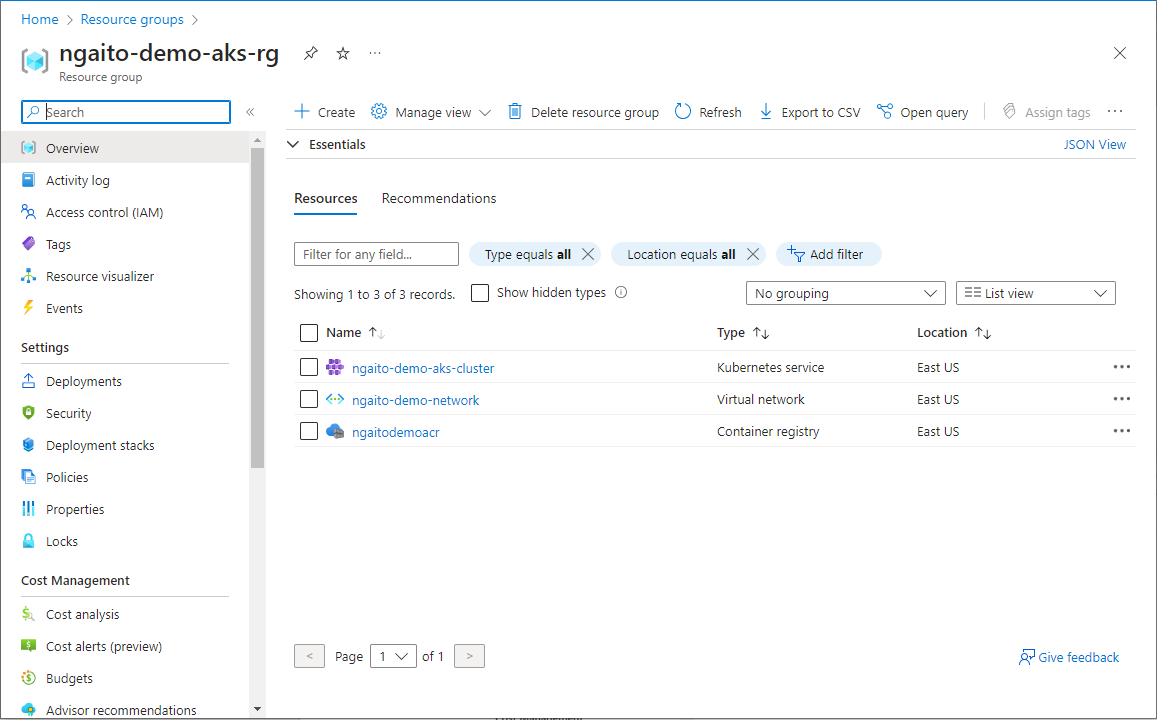


Verify the results on Azure Portal:



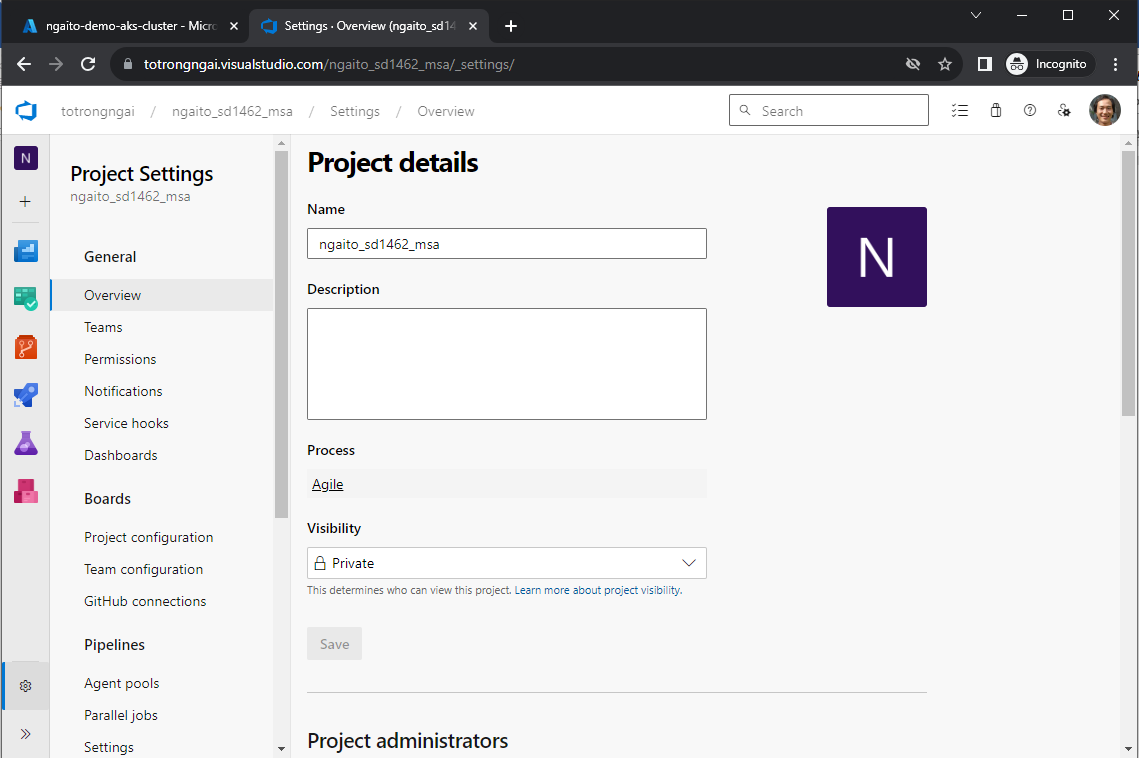


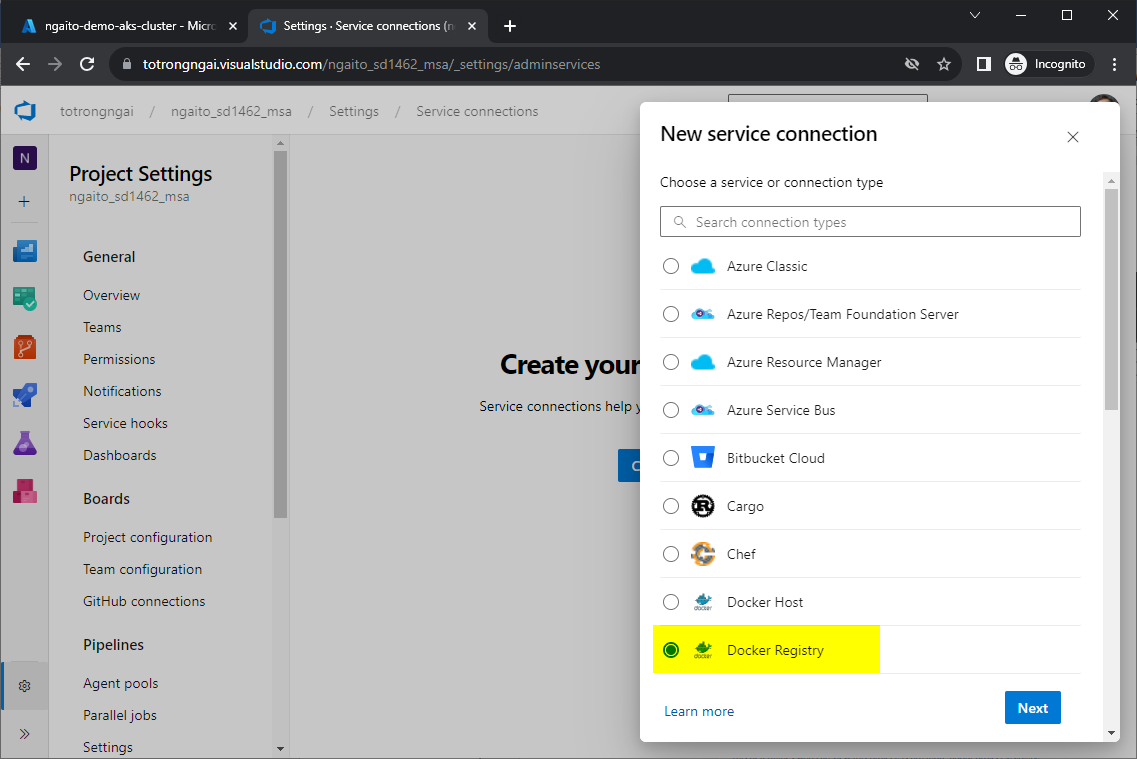


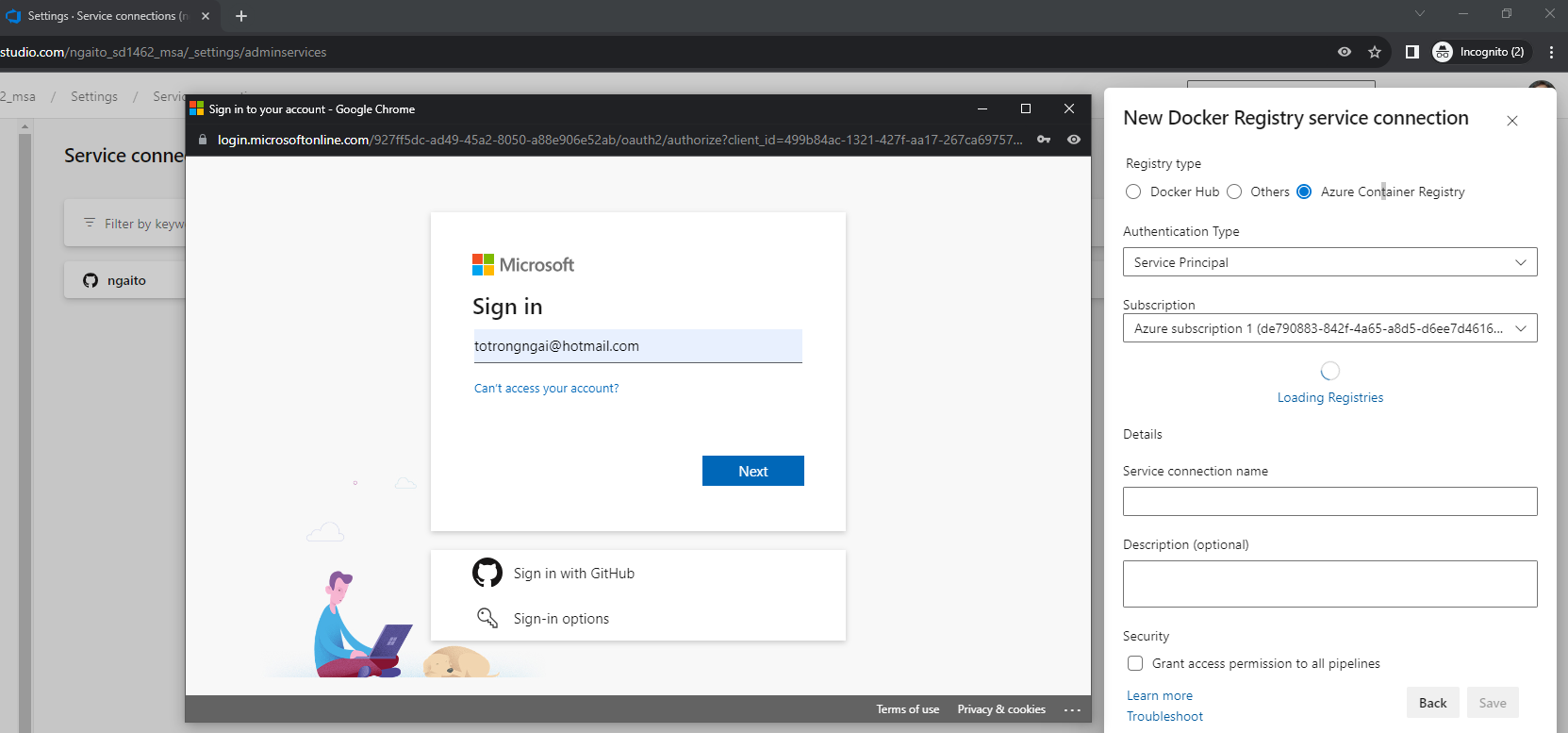


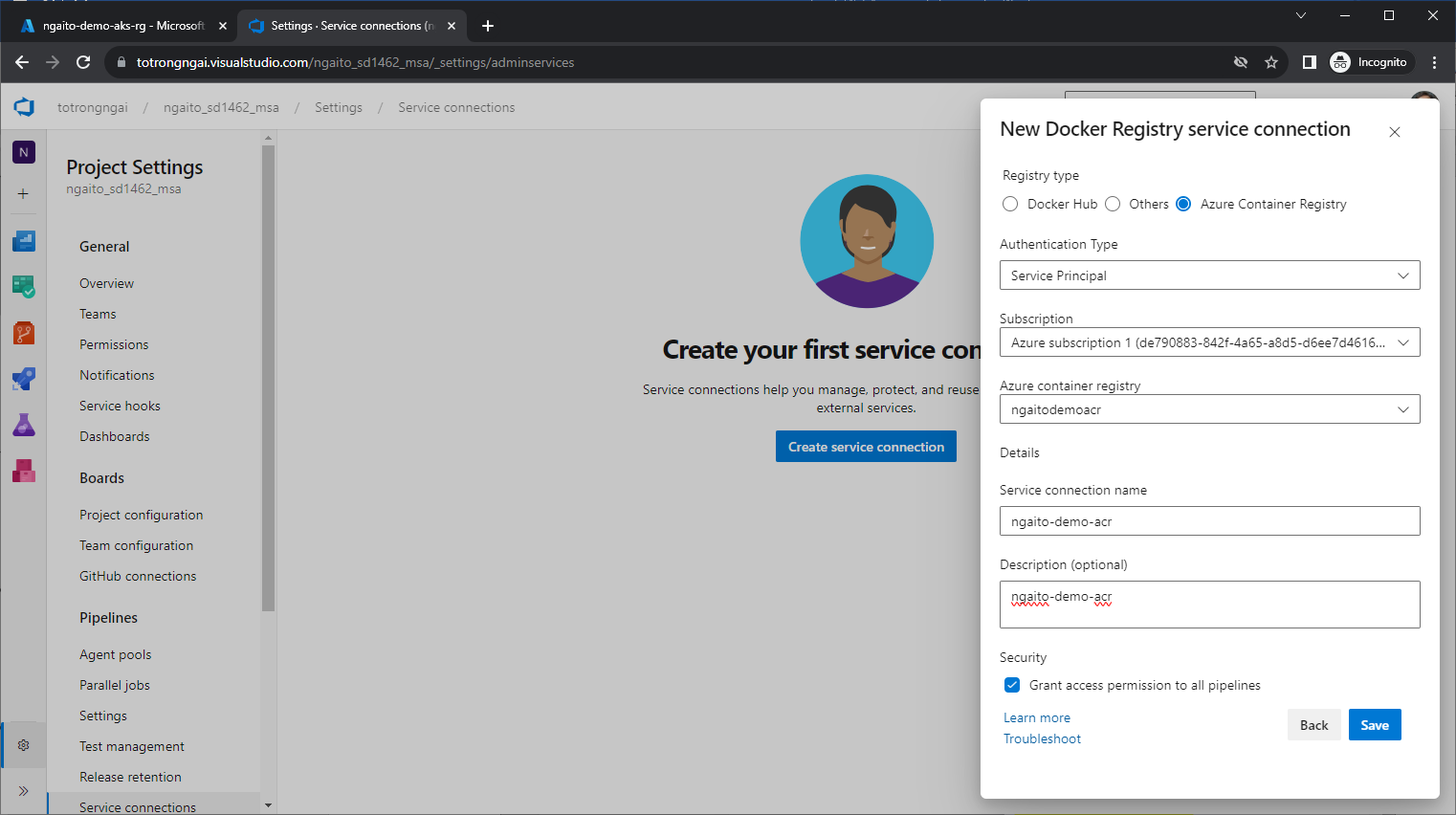
1. **Setup Azure Pipeline for CI**

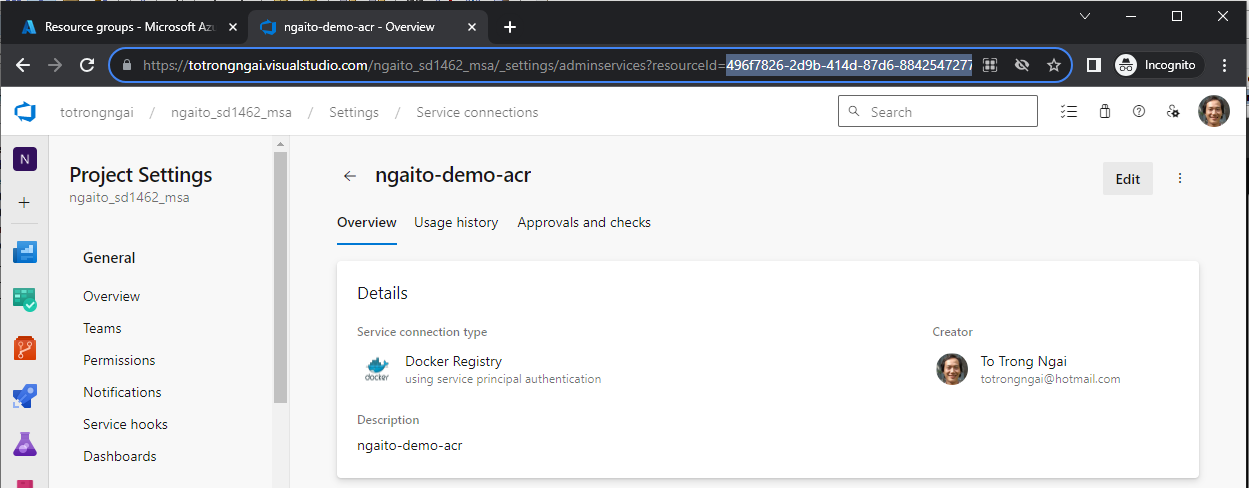
Project settings

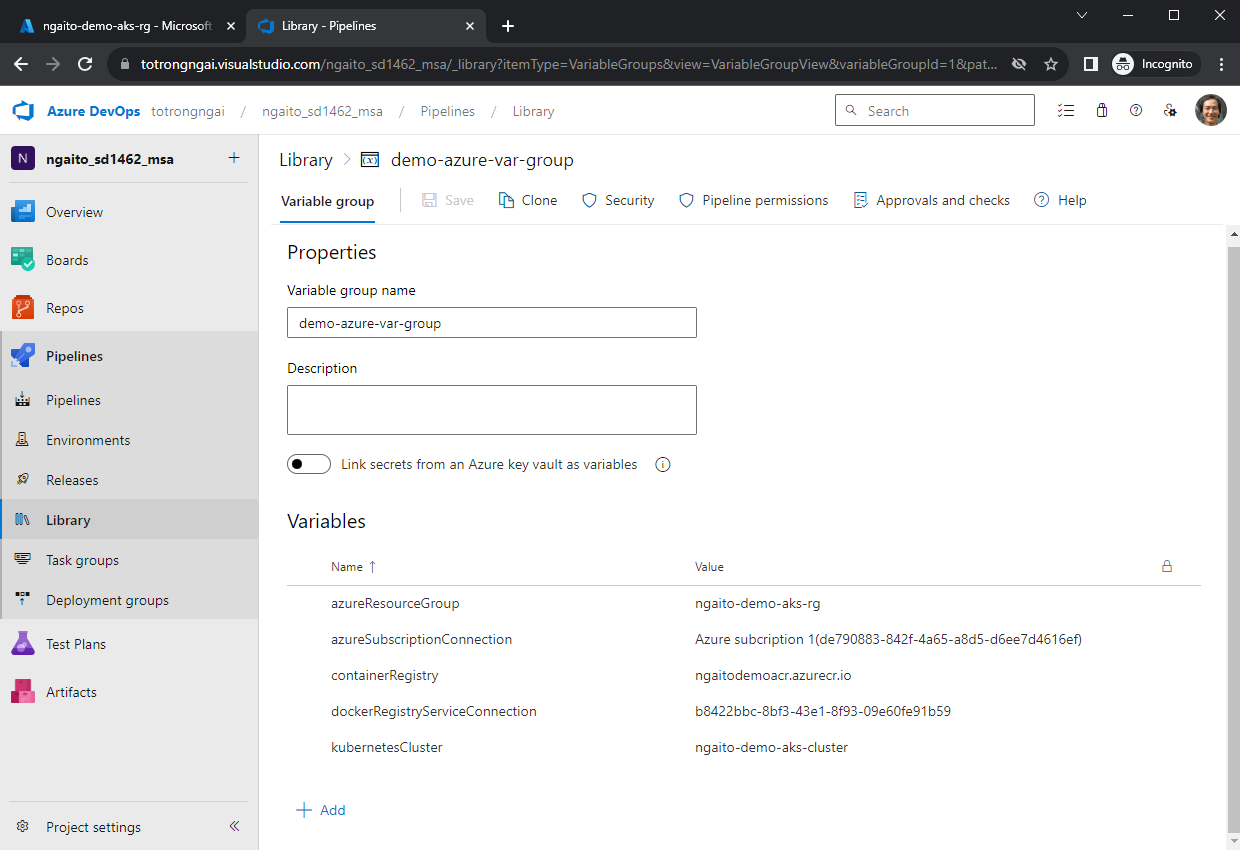




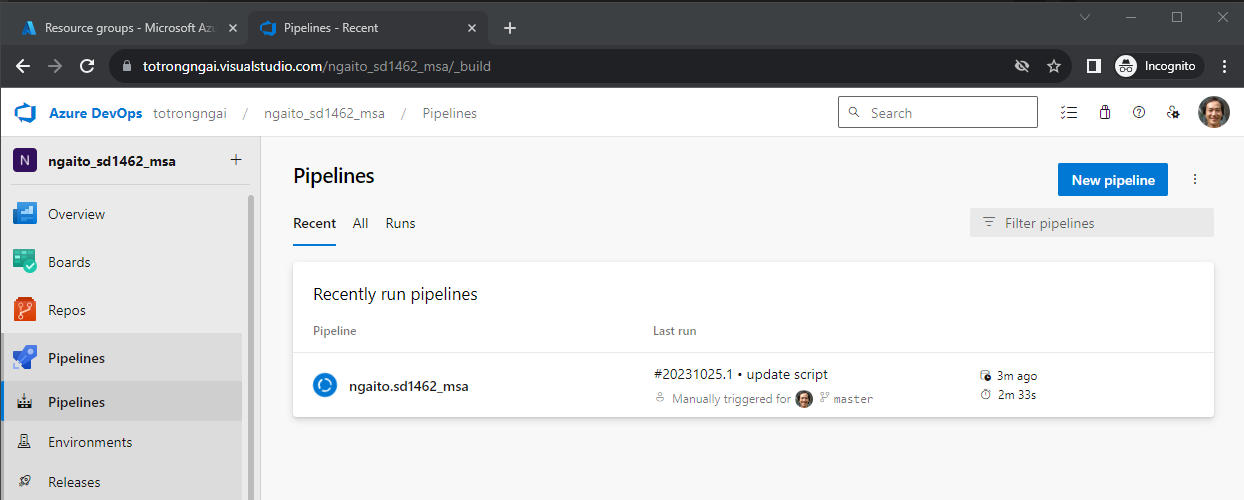




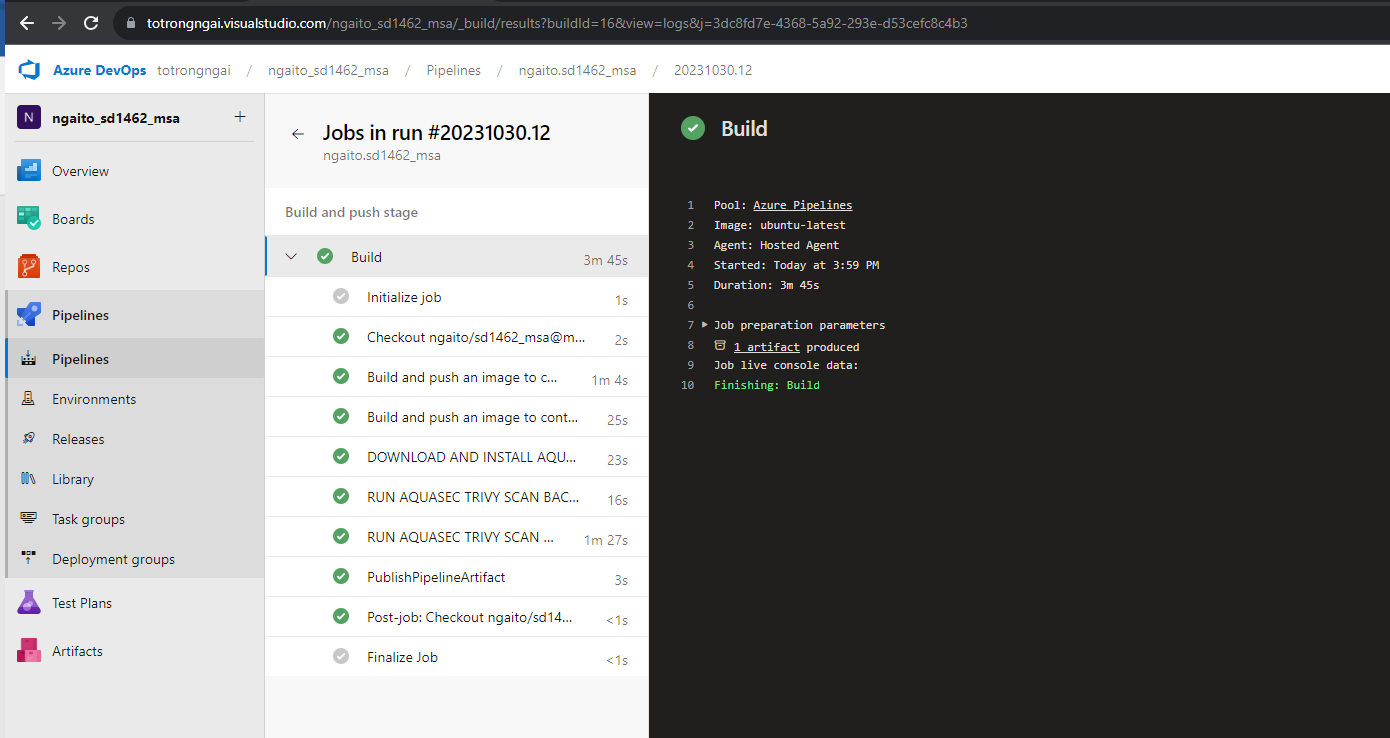




Run the pipeline:

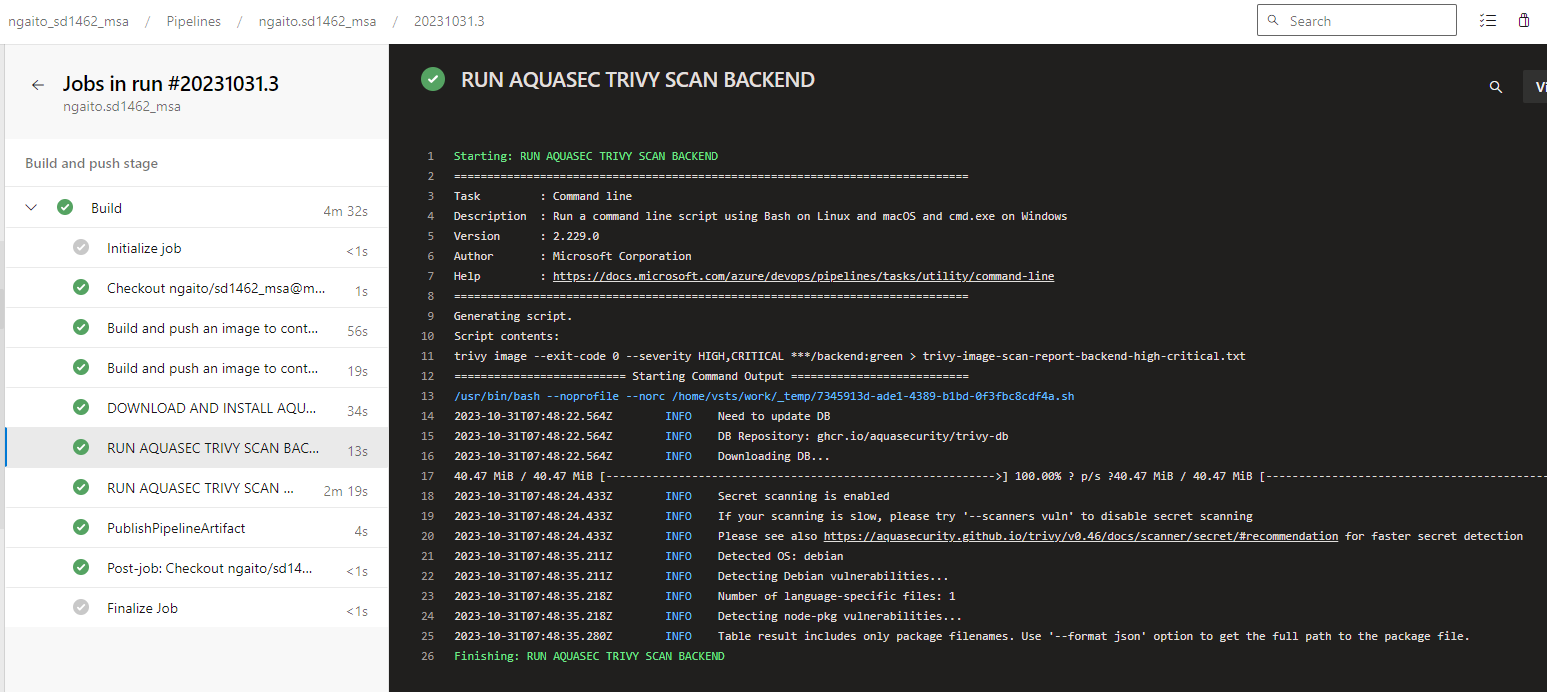


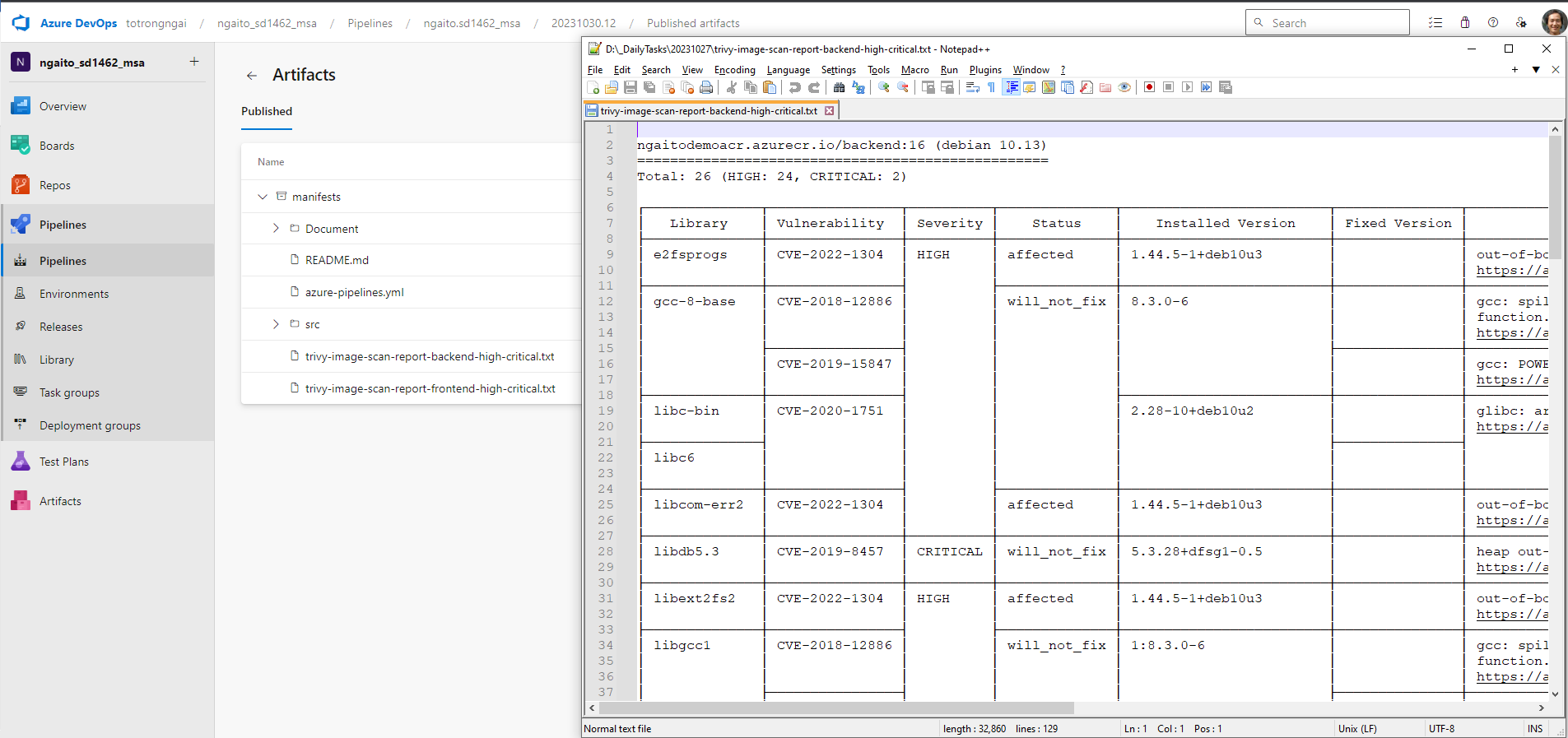
Verify result at pipeline:



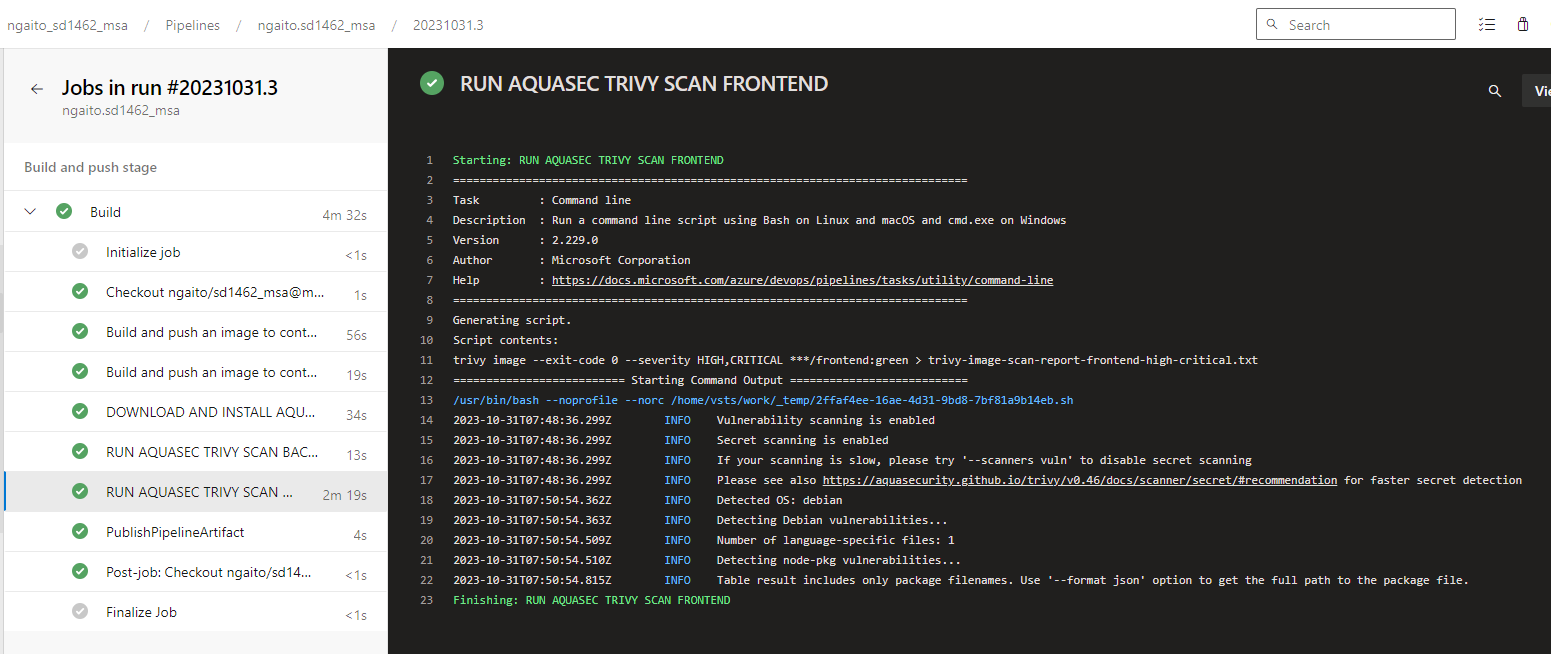
**Use Trivy in Azure Pipeline CI:**

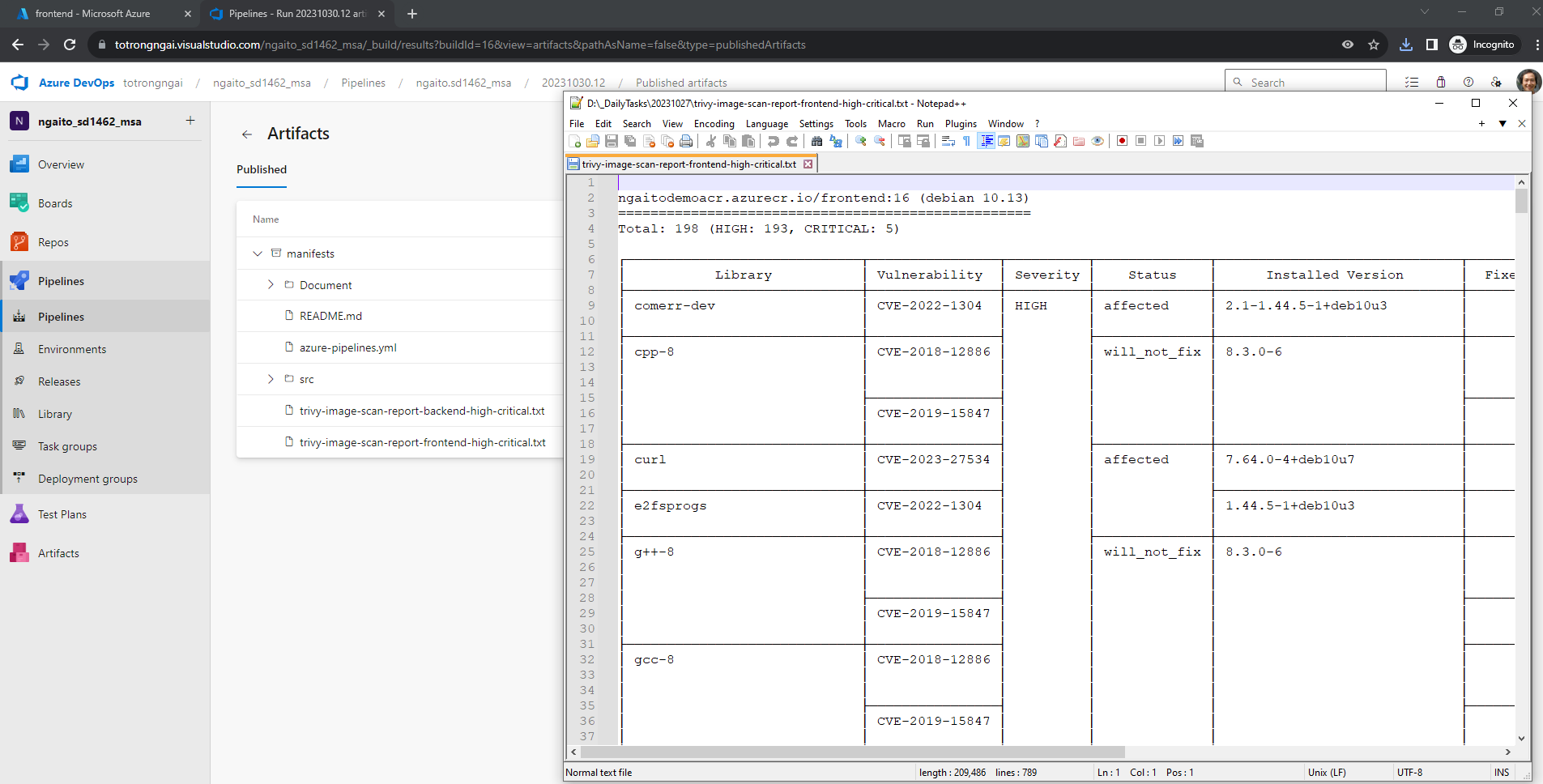
#Report backend:



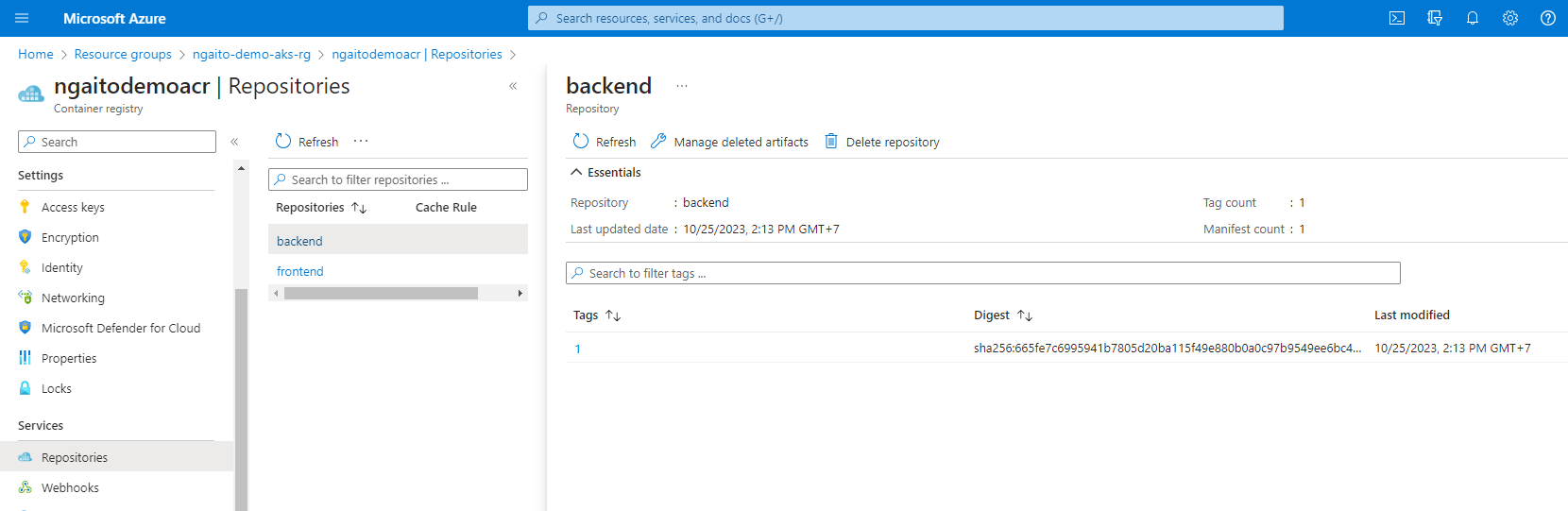


#Report Frontend:





Verify result at ACR:



Connect to AKS:

az aks list -o table

az aks get-credentials --resource-group <your resource group> --name <your cluster name>

az aks get-credentials -n ngaito-demo-aks-cluster -g ngaito-demo-aks-rg

Install NGINX ingress controller:

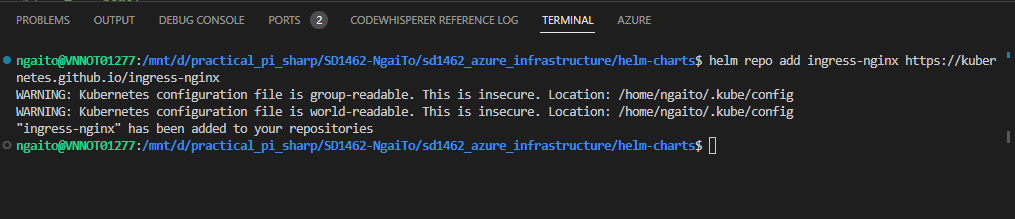
#search ingressClasses

kubectl get ingressClasses

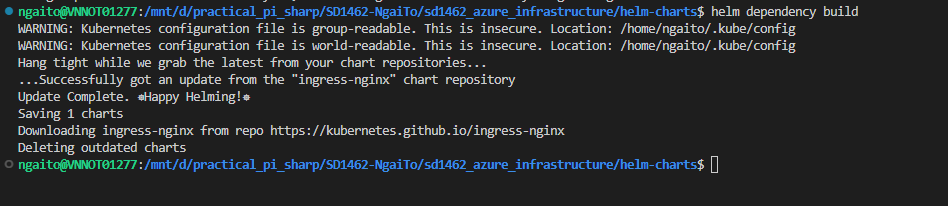
#Delete the current nginx ingressClass

kubectl delete ingressClasses nginx

*helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx*



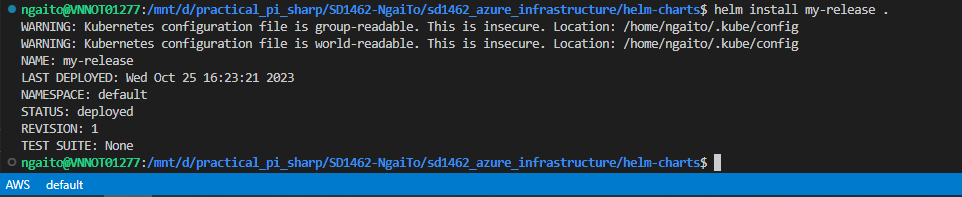
*helm dependency build*

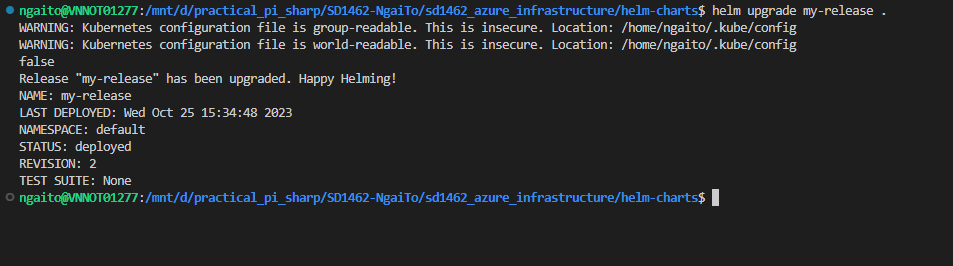


**Deploy application using helm-charts:**

*helm install my-release .*

*helm upgrade my-release .*

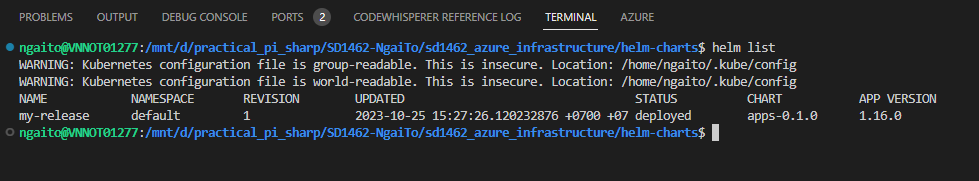




Verify the chart installation:

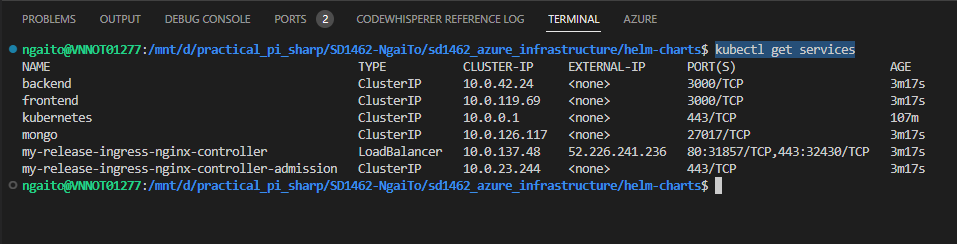
# verify the release

*helm list*



# verify the application resources

*kubectl get services*



HELM login to ACR:

*export HELM\_EXPERIMENTAL\_OCI=1*

*USER\_NAME="00000000-0000-0000-0000-000000000000"*

*PASSWORD=$(az acr login --name ngaitodemoacr --expose-token --output tsv --query accessToken)*

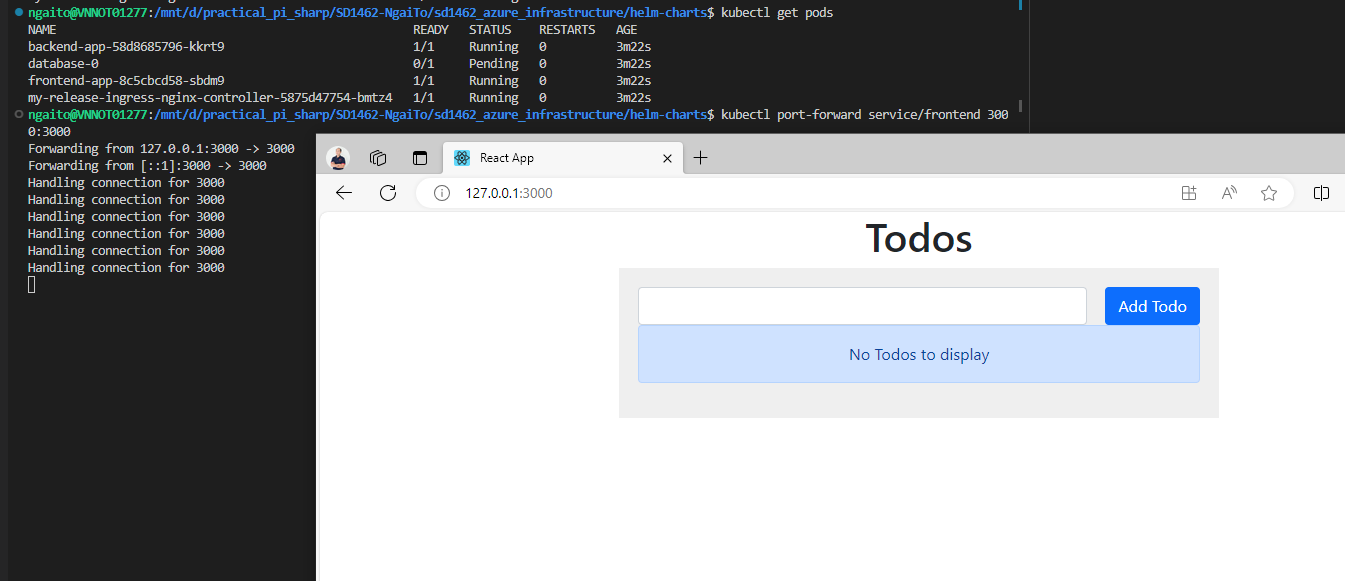
*helm registry login ngaitodemoacr.azurecr.io \*

*--username $USER\_NAME \*

*--password $PASSWORD*

Expose frontend service to access application:

*kubectl port-forward service/frontend 3000:3000*



1. **Monitoring:**

Install istio with Helm:

*helm repo add istio https://istio-release.storage.googleapis.com/charts*

*helm repo update*

*kubectl create namespace istio-system*

*helm install istio-base istio/base -n istio-system --set defaultRevision=default*

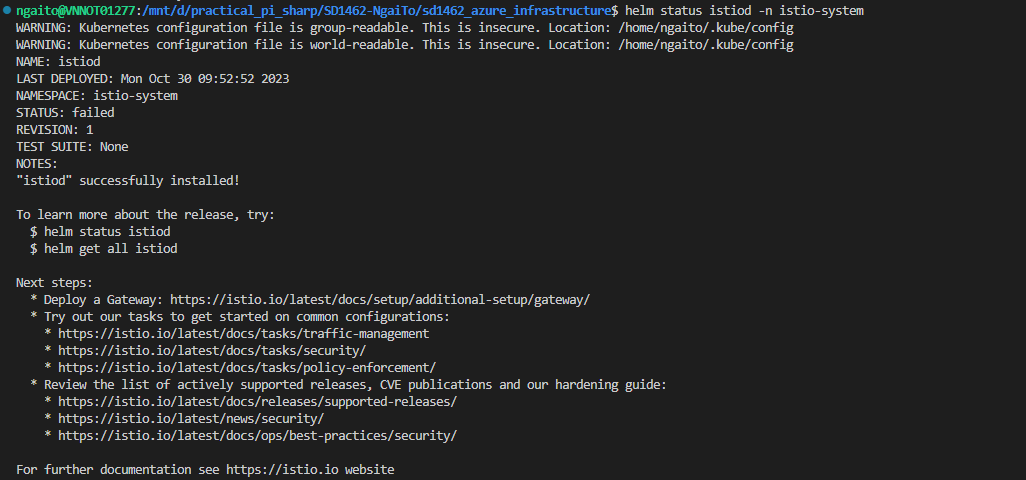
*helm ls -n istio-system*

*helm install istiod istio/istiod -n istio-system*

*helm ls -n istio-system*

*helm status istiod -n istio-system*

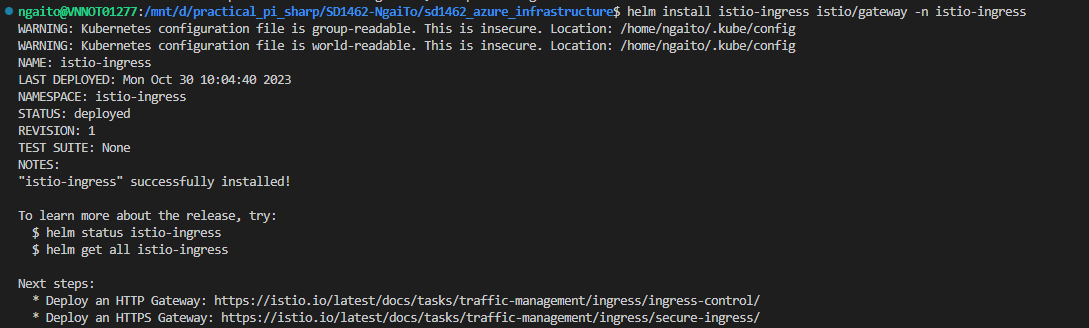
*kubectl get deployments -n istio-system --output wide*



Install an ingress gateway:

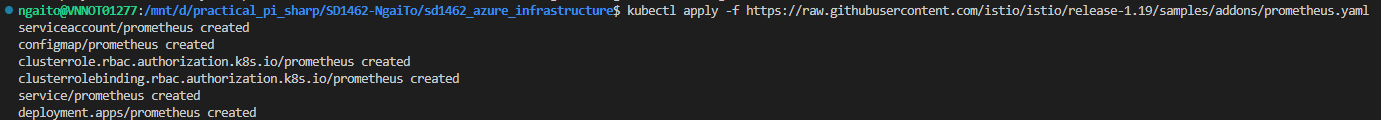
*kubectl create namespace istio-ingress*

*helm install istio-ingress istio/gateway -n istio-ingress*



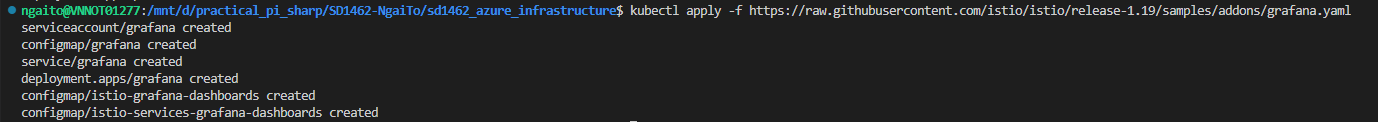
Prometheus:

*kubectl apply -f* [*https://raw.githubusercontent.com/istio/istio/release-1.19/samples/addons/prometheus.yaml*](https://raw.githubusercontent.com/istio/istio/release-1.19/samples/addons/prometheus.yaml)



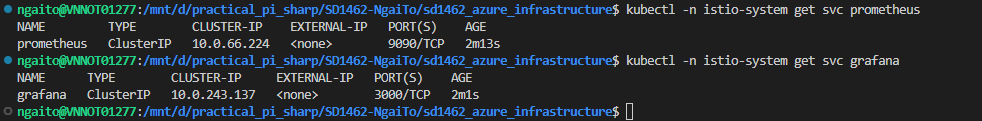
Grafana:

*kubectl apply -f* [*https://raw.githubusercontent.com/istio/istio/release-1.19/samples/addons/grafana.yaml*](https://raw.githubusercontent.com/istio/istio/release-1.19/samples/addons/grafana.yaml)



*kubectl -n istio-system get svc prometheus*

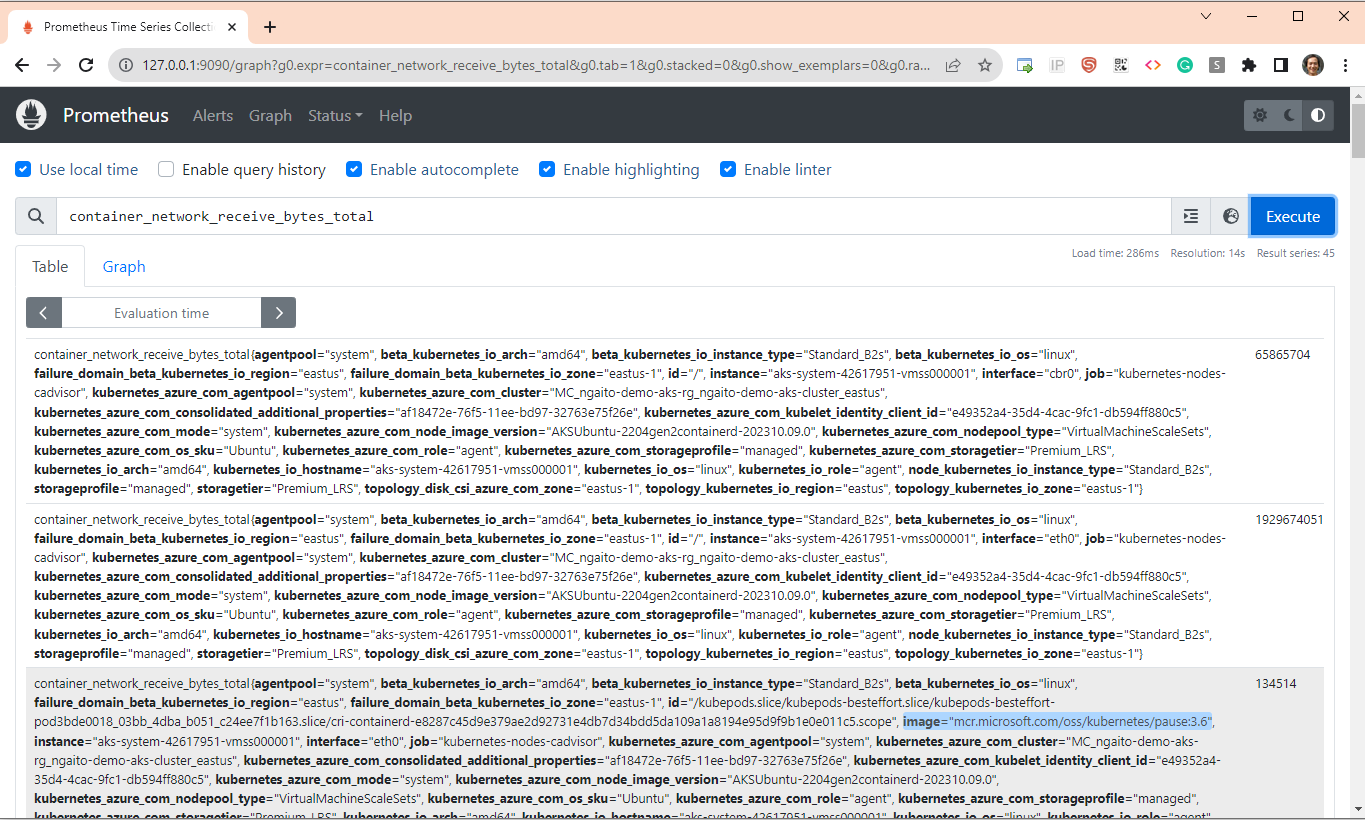
*kubectl -n istio-system get svc grafana*



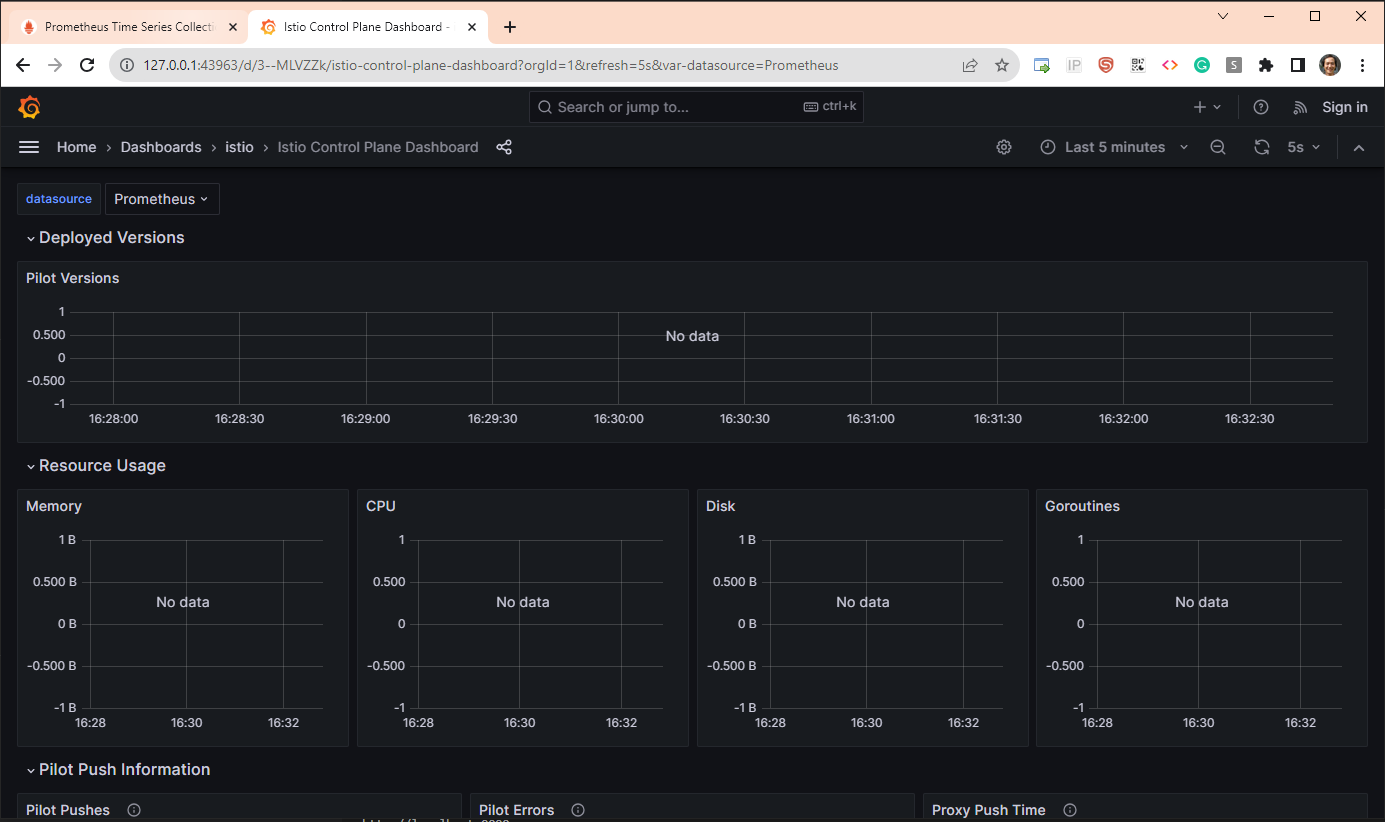
*curl -sL https://istio.io/downloadIstioctl | sh -*

*export PATH=$HOME/.istioctl/bin:$PATH*

*istioctl dashboard prometheus*



*istioctl dashboard grafana*

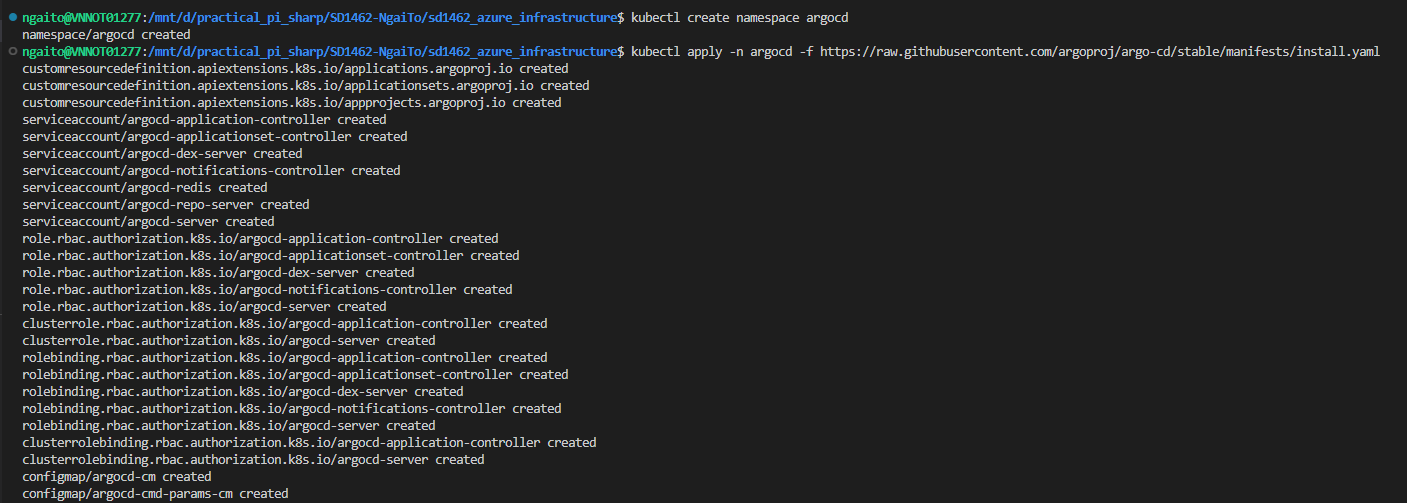


1. **Use GitOps for the CD pipeline:**

Install Argo CD on AKS cluster:

*kubectl create namespace argocd*

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



Access to Argo CD UI:

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

*export argocd\_password=$(kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.password}" | base64 -d)*

https://localhost:8080/

usename: admin

password: *echo ${argocd\_password}*

Install the ArgoCD CLI:

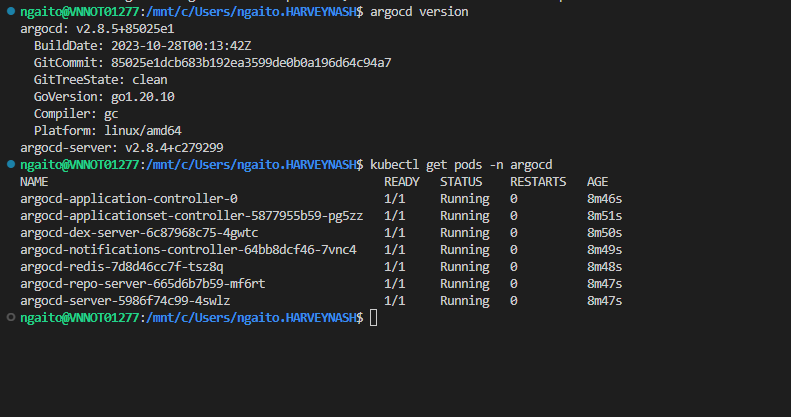
*sudo curl -sSL -o /usr/local/bin/argocd https://github.com/argoproj/argo-cd/releases/latest/download/argocd-linux-amd64*

*sudo chmod +x /usr/local/bin/argocd*

To verify if the ArgoCD CLI:

*argocd version*

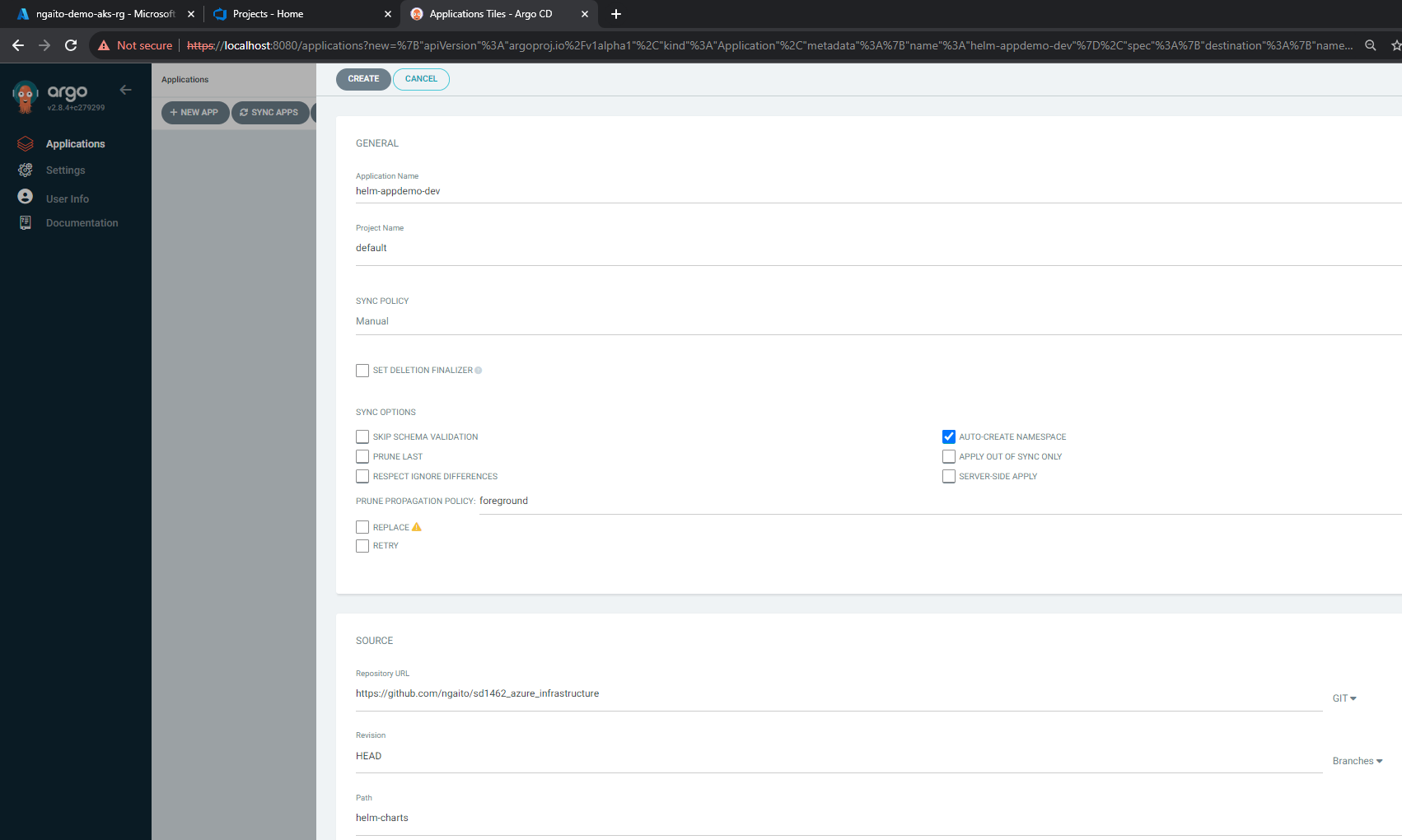
*kubectl get pods -n argocd*

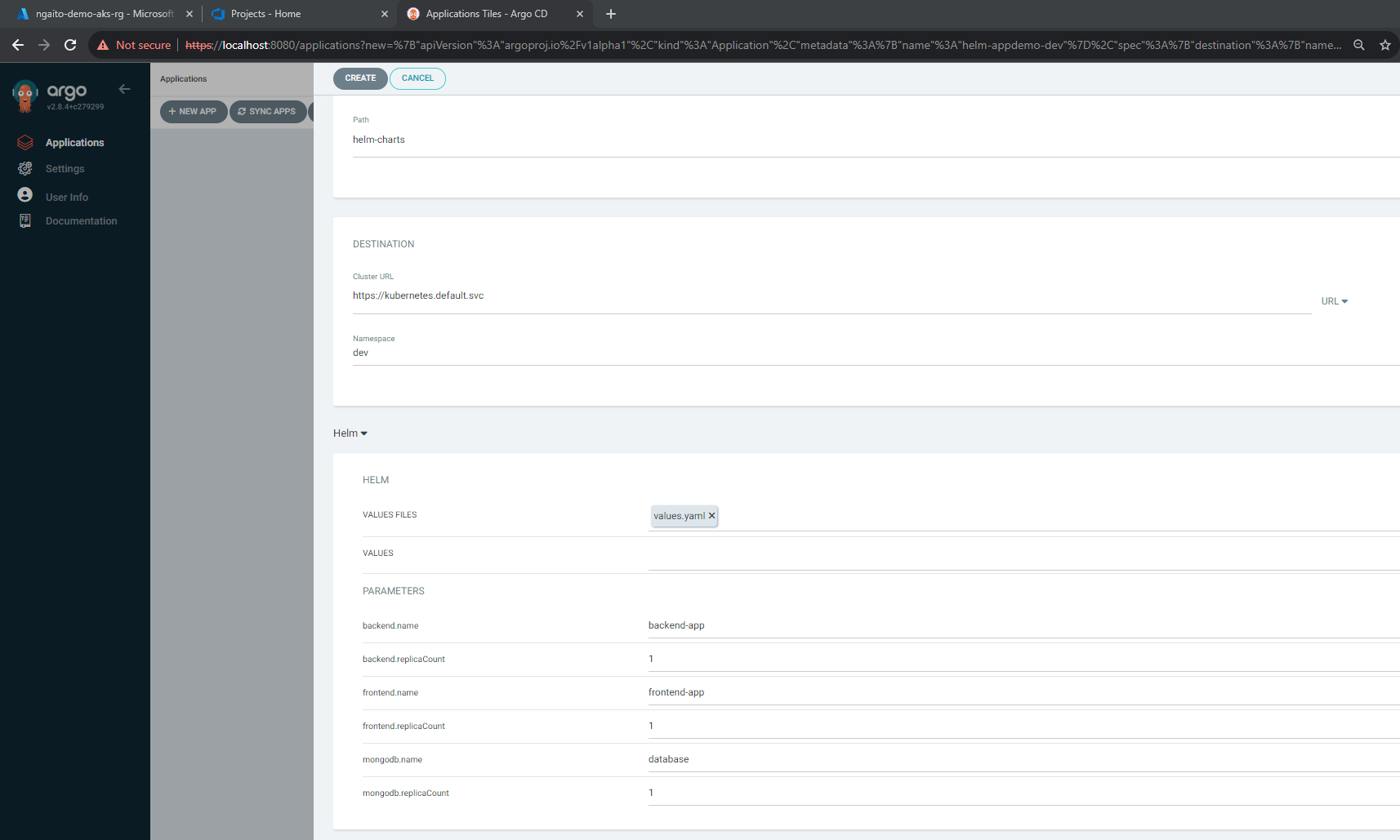


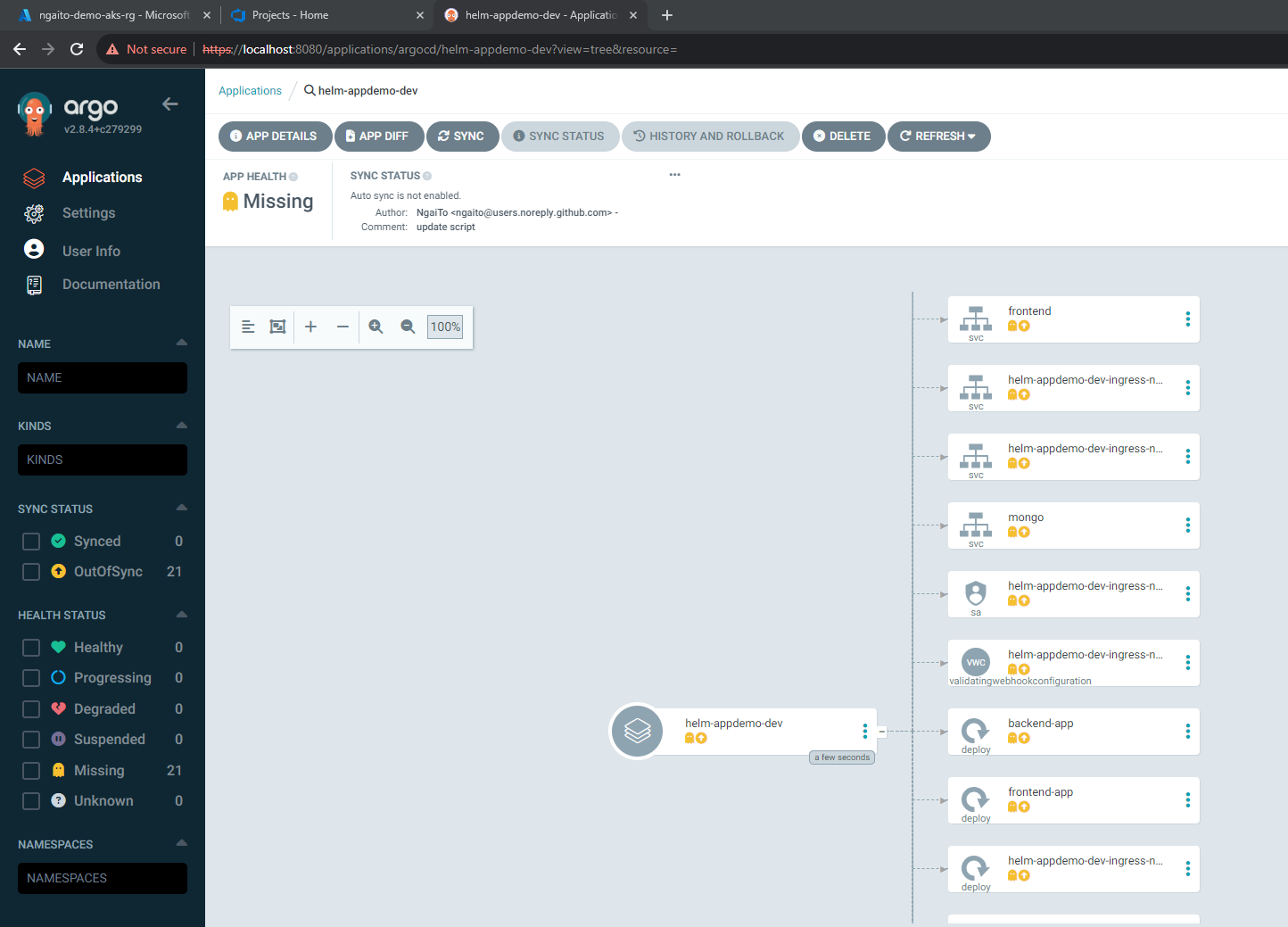
Login via the CLI:

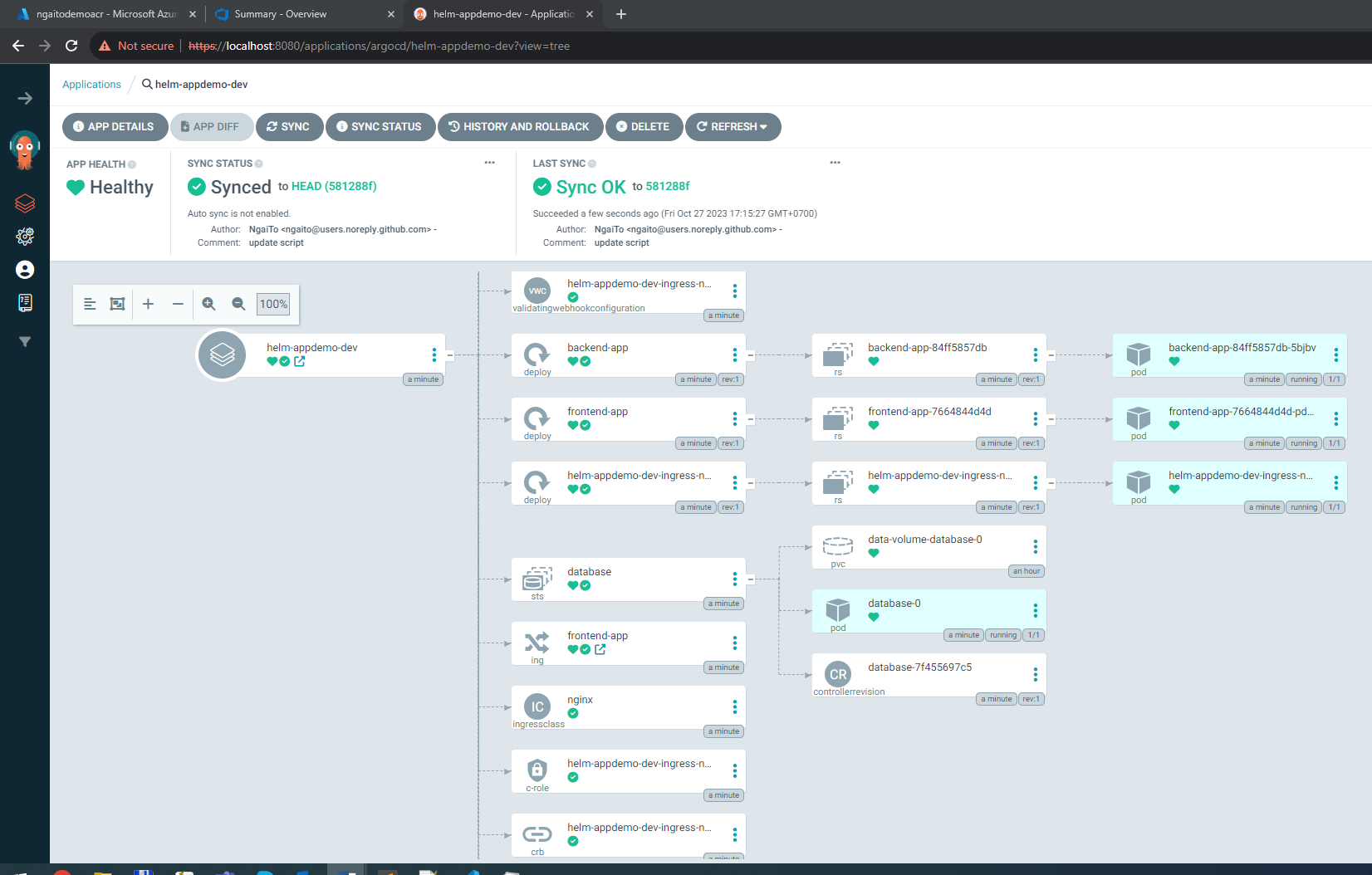
*argocd login --insecure --username=admin --password=${argocd\_password} localhost:8080*

Create an application in Argo CD using GUI:



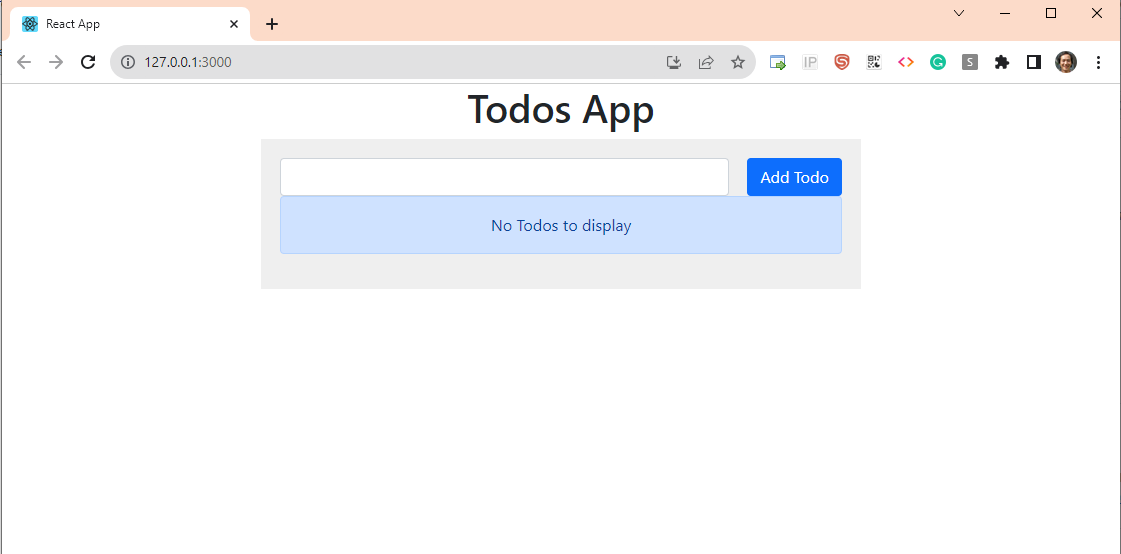






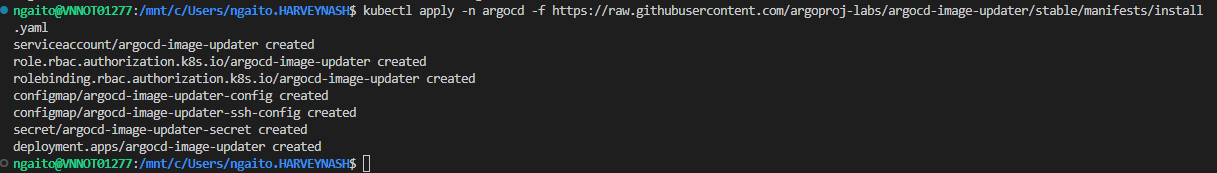
Expose frontend service to access application:

*kubectl port-forward service/frontend 3000:3000 –n dev*



**Install Argo CD image Update:**

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

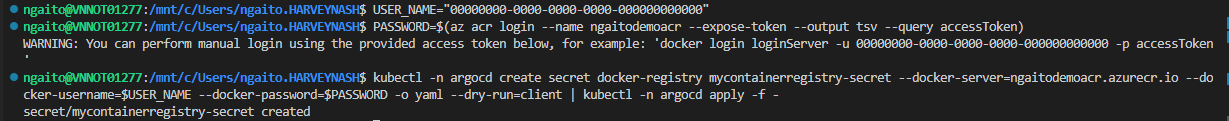


Create pull secret:

*USER\_NAME="00000000-0000-0000-0000-000000000000"*

*PASSWORD=$(az acr login --name ngaitodemoacr --expose-token --output tsv --query accessToken)*

*kubectl -n argocd create secret docker-registry mycontainerregistry-secret --docker-server=ngaitodemoacr.azurecr.io --docker-username=$USER\_NAME --docker-password=$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*

*…*

*data:*

*registries.conf: |*

*registries:*

*- name: Azure Container Registry*

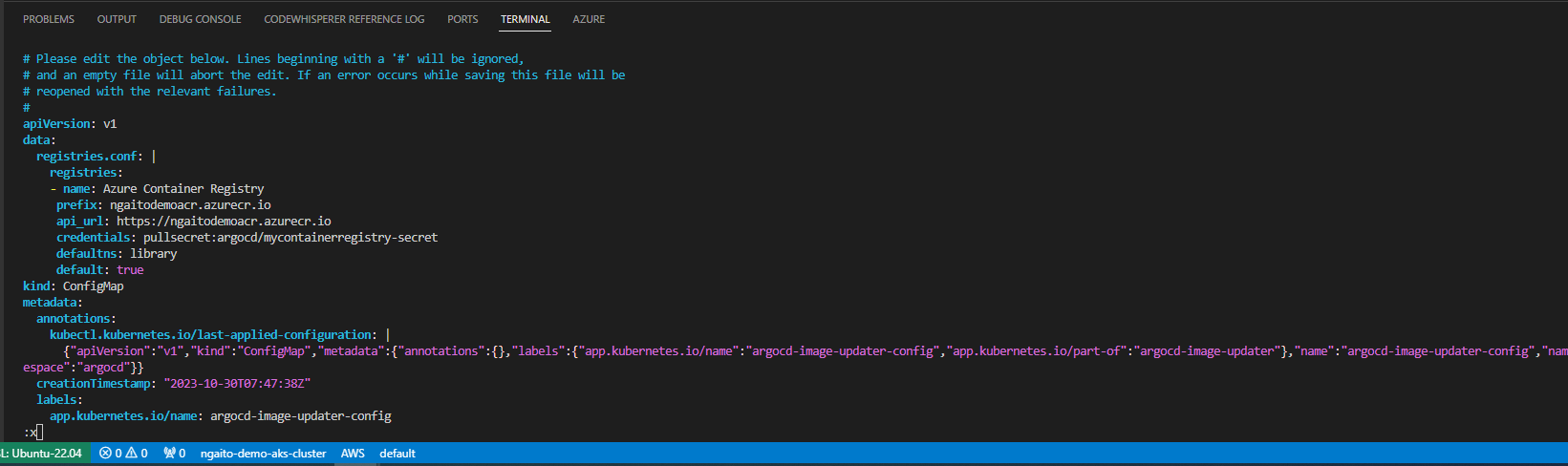
*prefix: ngaitodemoacr.azurecr.io*

*api\_url: https://ngaitodemoacr.azurecr.io*

*credentials: pullsecret:argocd/mycontainerregistry-secret*

*defaultns: library*

*default: true*



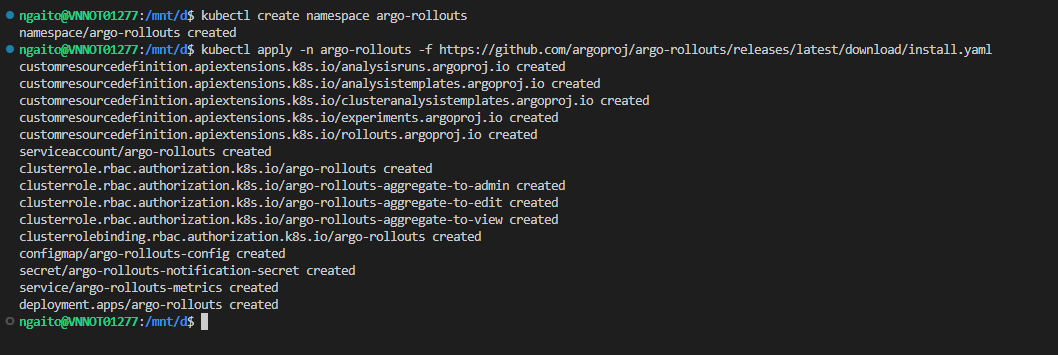
Modify app and trigger CI build for new version

**Use Blue/Green strategy for deployment:**

Controller Installation:

*kubectl create namespace argo-rollouts*

*kubectl apply -n argo-rollouts -f* [*https://github.com/argoproj/argo-rollouts/releases/latest/download/install.yaml*](https://github.com/argoproj/argo-rollouts/releases/latest/download/install.yaml)

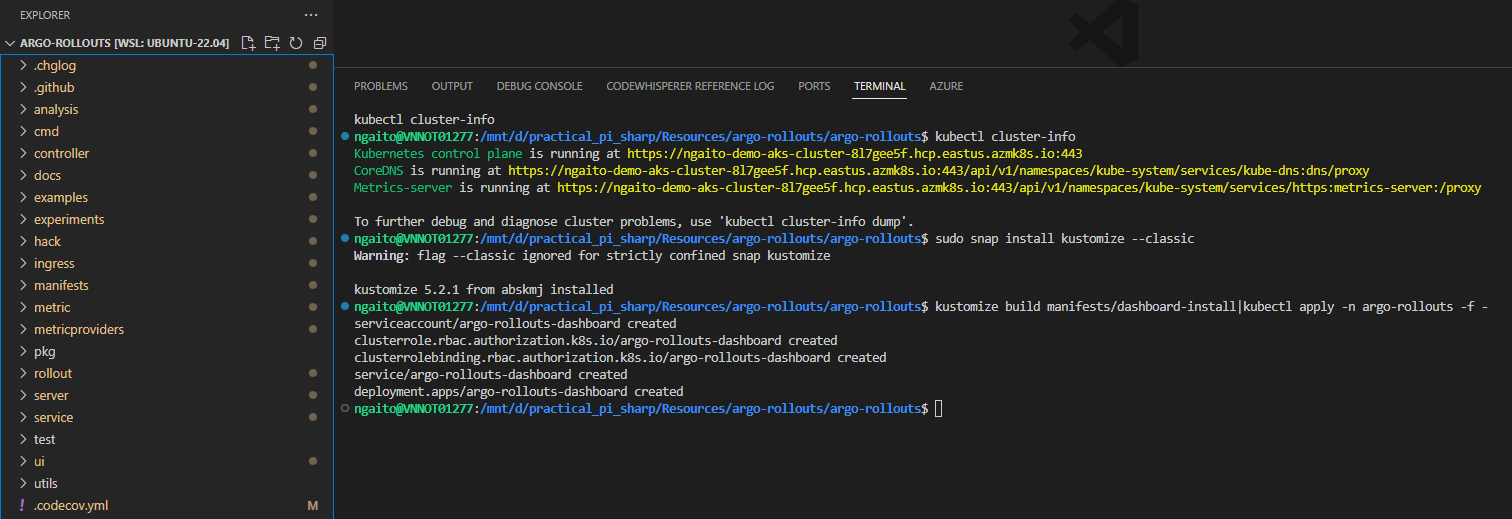


Deploy the Rollouts Dashboard:

*git clone* [*https://github.com/argoproj/argo-rollouts.git*](https://github.com/argoproj/argo-rollouts.git)

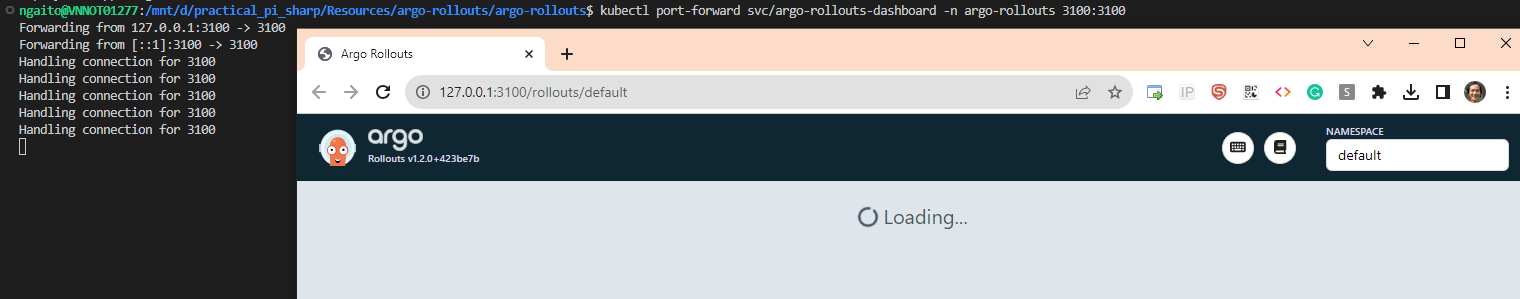
*sudo snap install kustomize --classic*

*kustomize build manifests/dashboard-install|kubectl apply -n argo-rollouts -f –*

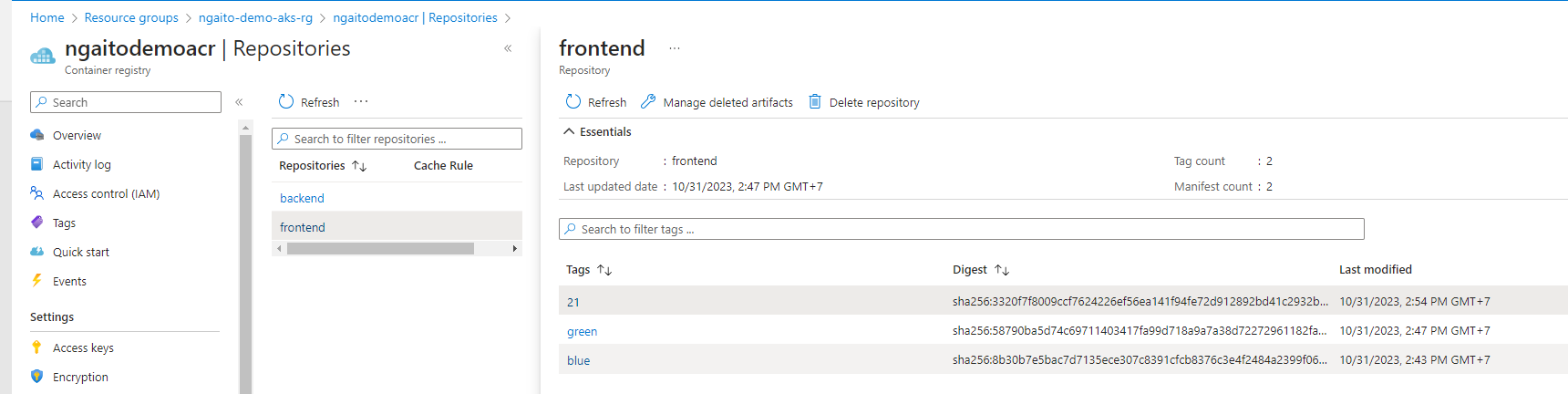


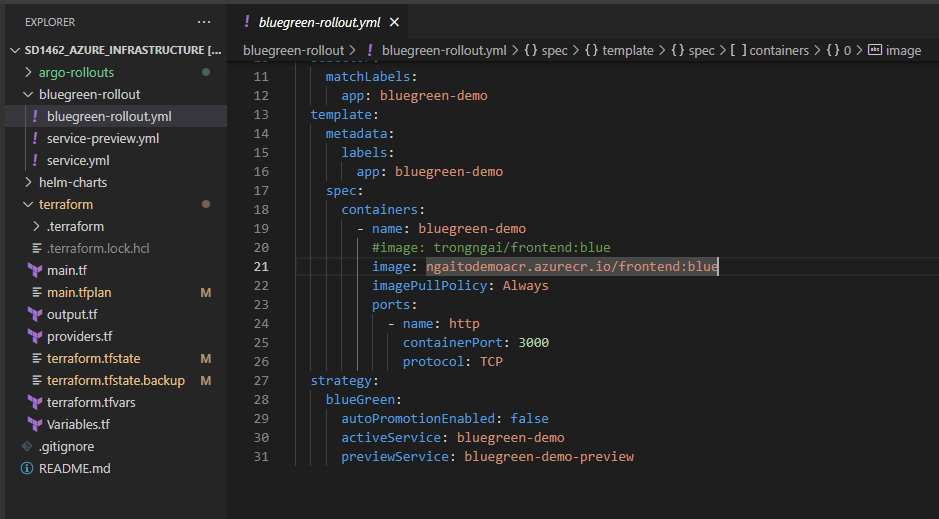
*kubectl port-forward svc/argo-rollouts-dashboard -n argo-rollouts 3100:3100*

[*http://127.0.0.1:3100*](http://127.0.0.1:3100)



Deploy the Blue version:

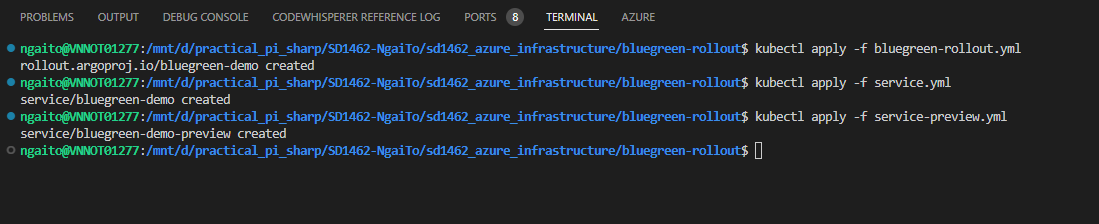


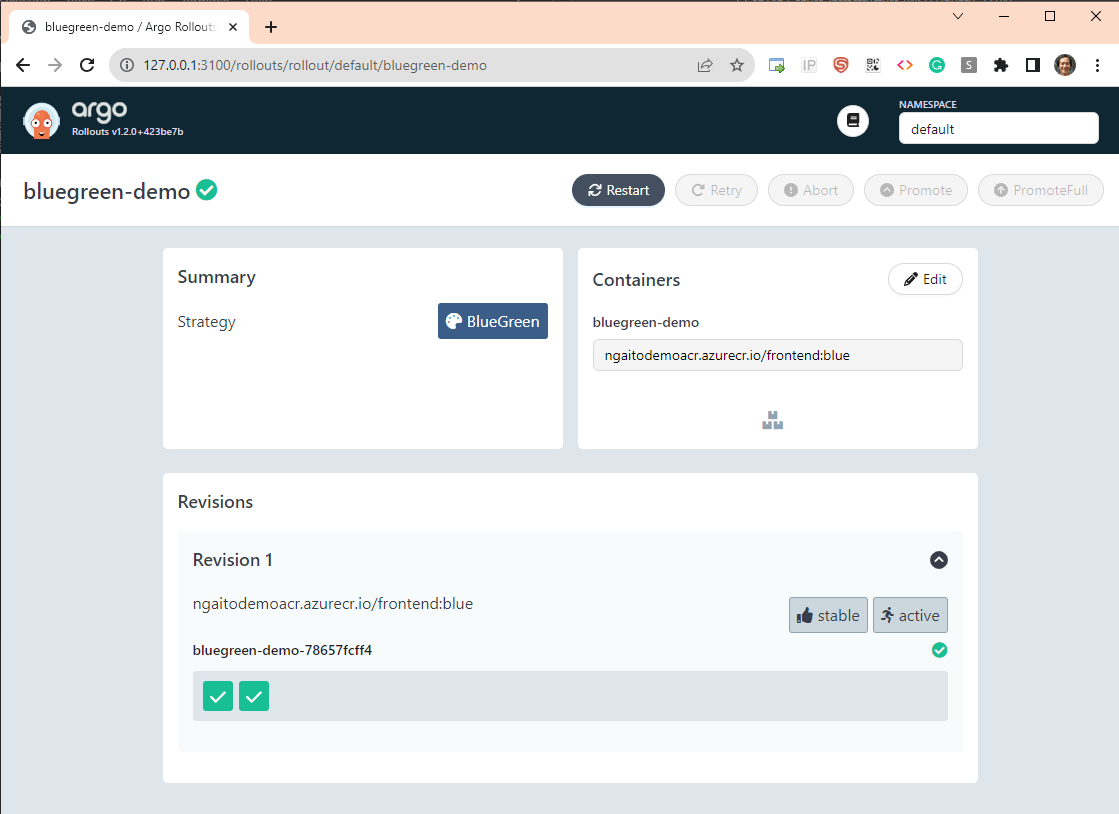


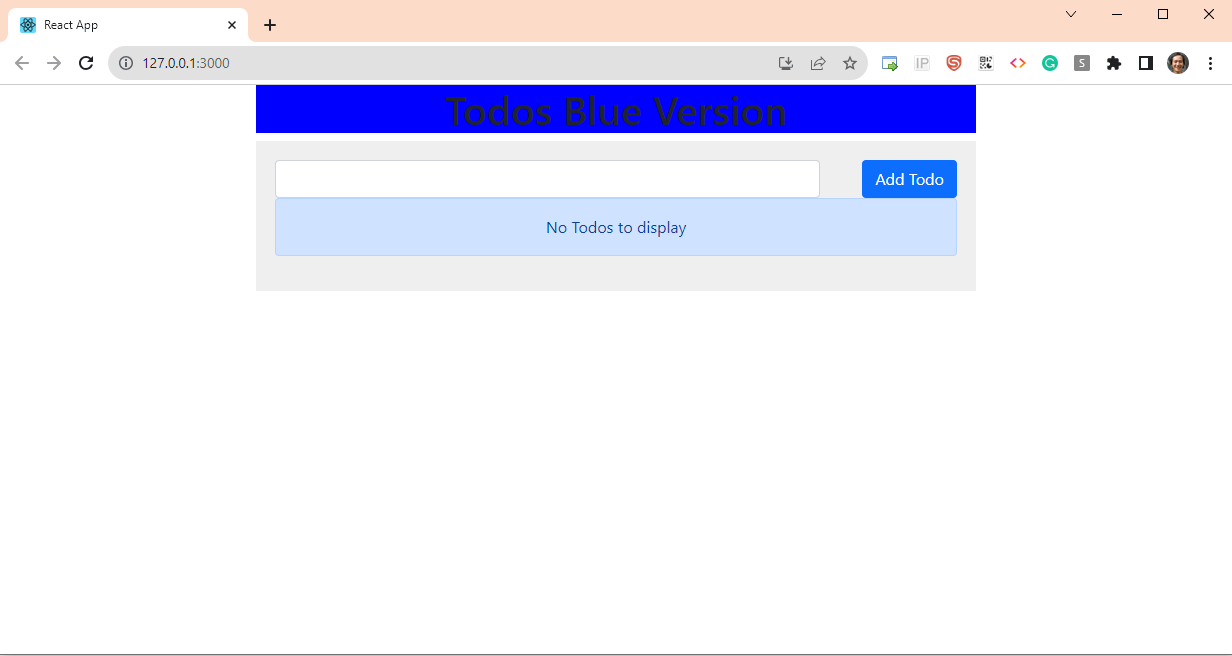
*kubectl apply -f bluegreen-rollout.yml*

*kubectl apply -f service.yml*

*kubectl apply -f service-preview.yml*

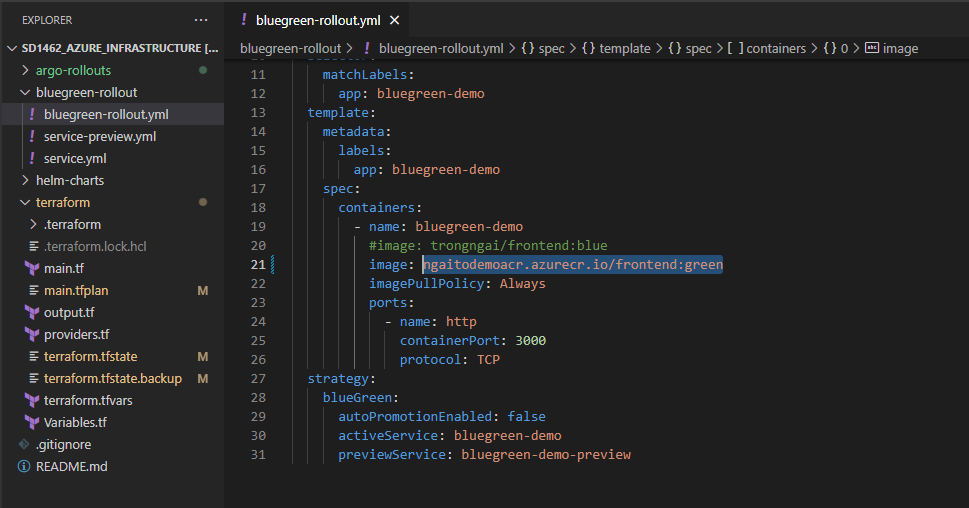




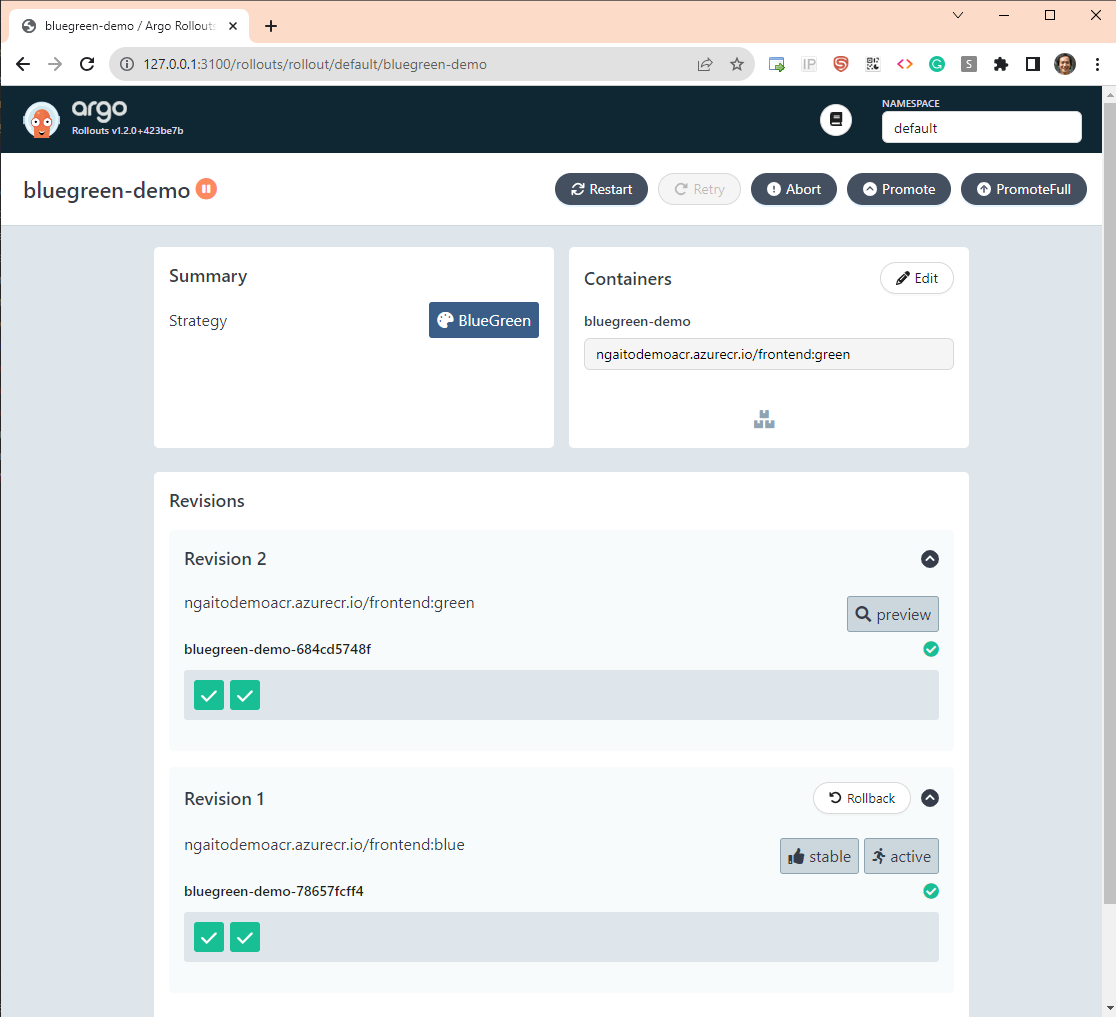


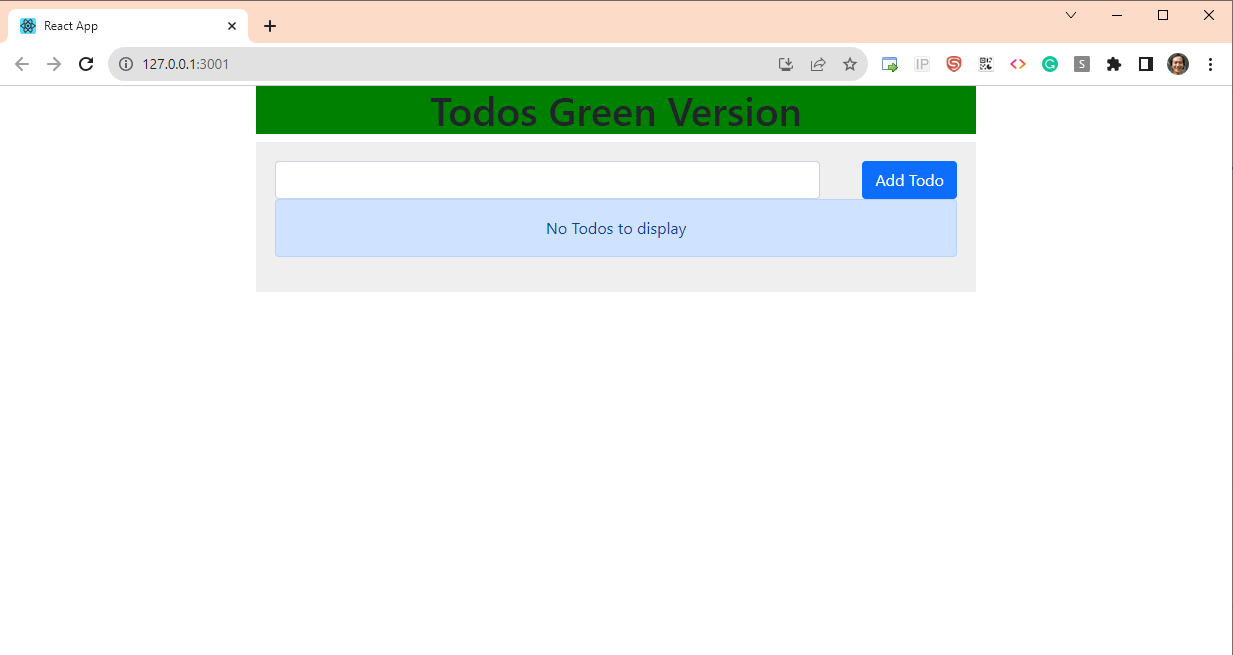
Deploy the Green version:

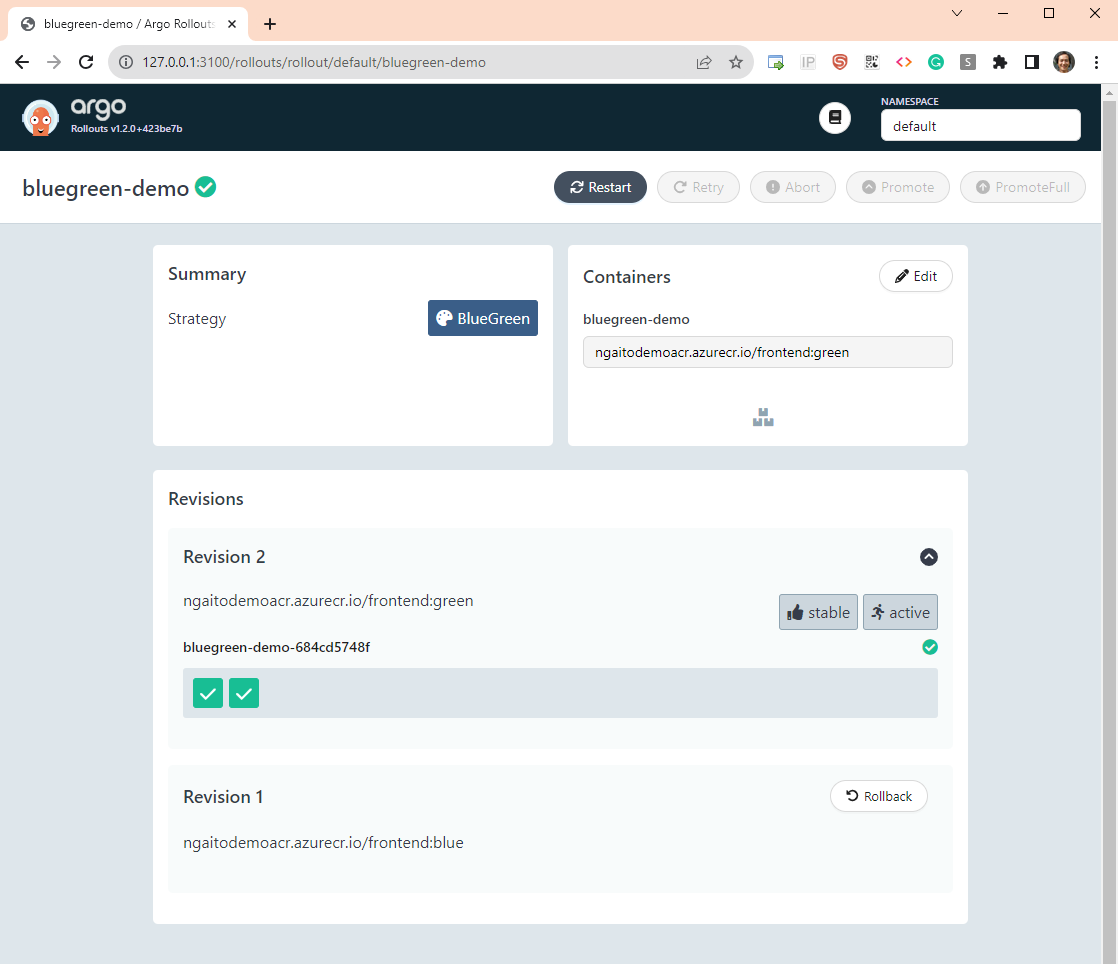
Change the image property of Rollout Object:

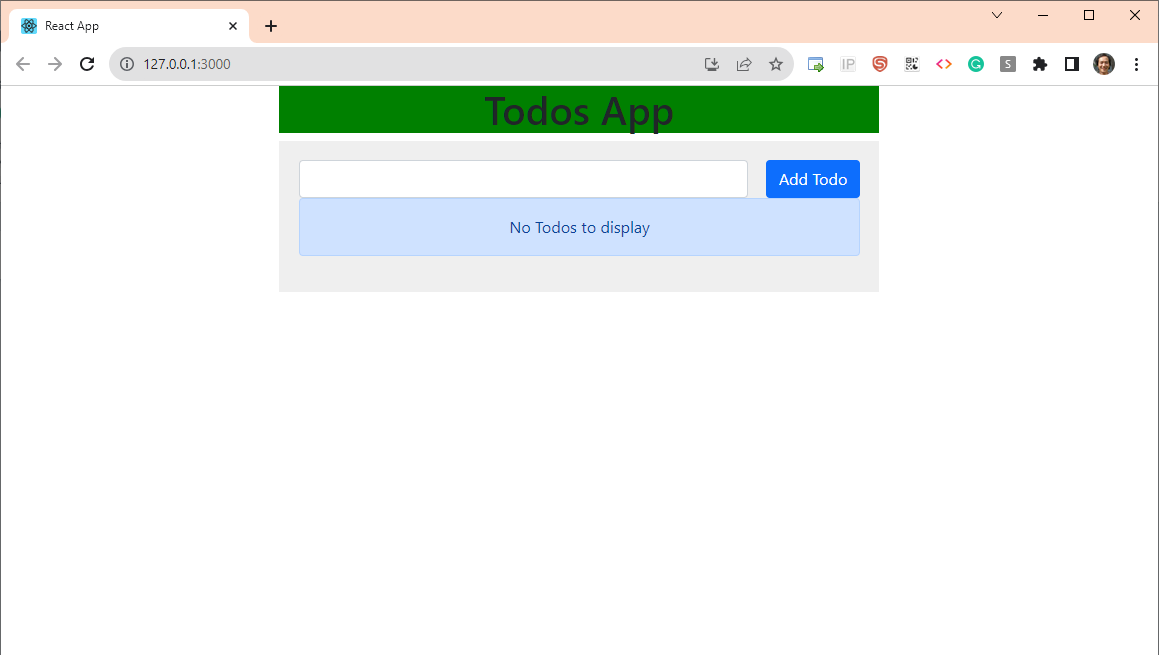


*kubectl apply -f bluegreen-rollout.yml*









1. **Perform clean up by deleting the AKS cluster:**

*terraform destroy --auto-approve*

