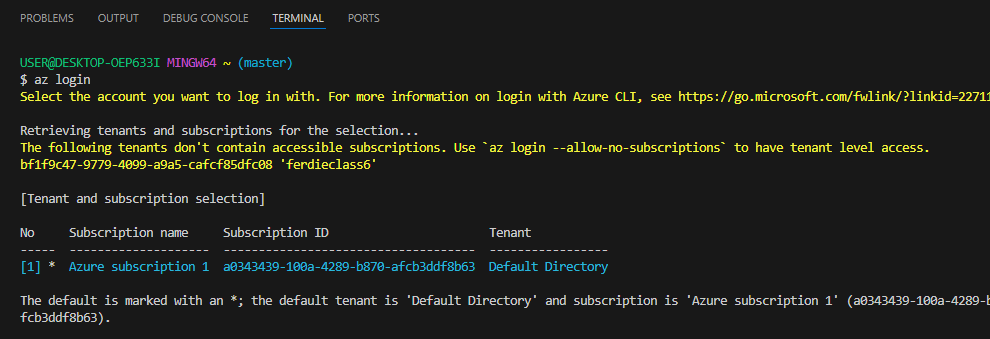
**AZURE ACR-AKS PROJECT**

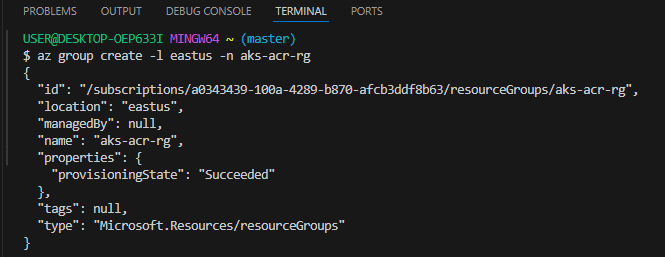
Login to bash terminal on visual studio code

|  |
| --- |
| az login |



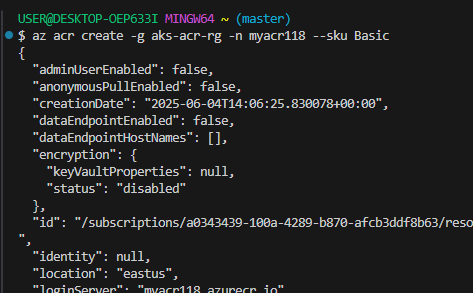
Create a resource group

|  |
| --- |
| az group create -l eastus -n aks-acr-rg |



Create ACR, specify the group, registry name, SKU

|  |
| --- |
| az acr create -g aks-acr-rg -n myacr118 --sku Basic |

****

Create the k8s cluster

az aks create -g aks-acr-rg -n myaks118 -c 2

NB: “—attach-acr myacr118” could’ve been used to integrate AKS and ACR during the cluster creation. While this installs, you can split the terminal and proceed with the others.

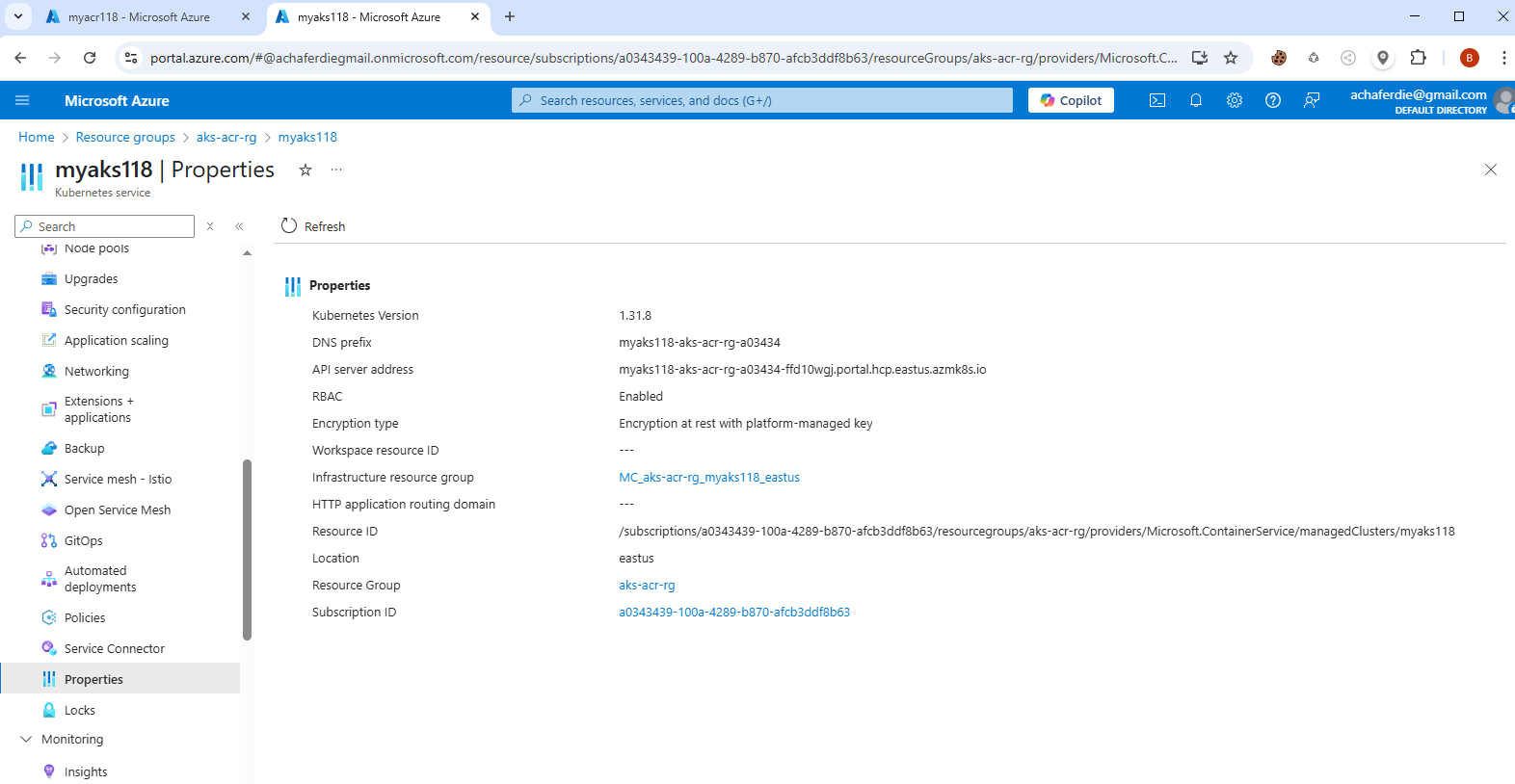
az acr import --name myacr118 --source docker.io/library/httpd:latest --image httpd:v1 --username markrobs \

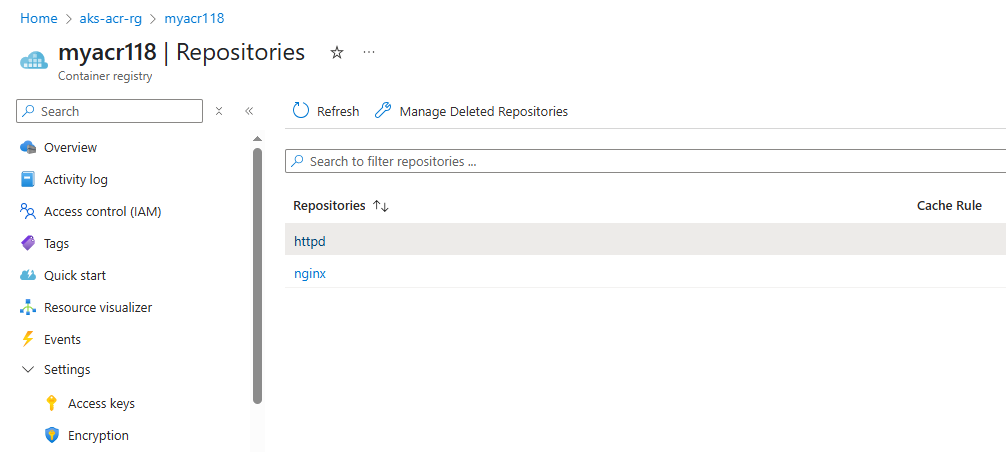
--password dckr\_pat\_\_ryeMobRbGQfkuDYrcE8-ty-pSA

|  |
| --- |
| az acr import \  --name myacr118 \  --source docker.io/library/httpd:latest \  --image httpd:v1 \  --username <DOCKER\_HUB\_USERNAME> \  --password <DOCKER\_HUB\_ACCESS\_TOKEN> |

PULL NGINX from docker hub to your ACR

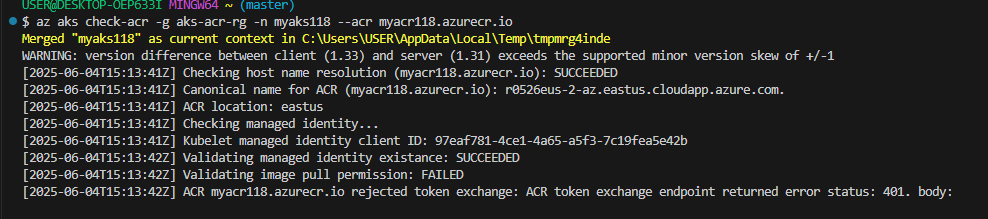
|  |
| --- |
| az acr import --name myacr118 --source docker.io/library/nginx:latest --image nginx:v1 --username markrobs \  --password dckr\_pat\_\_ryeMobRbGQfkuDYrcE8-ty-pSA |



ACR REGISTRY

Connect to aks cluster in terminal

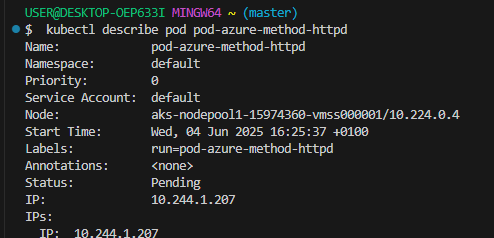
|  |
| --- |
| $ az aks get-credentials -g aks-acr-rg -n myaks118 |

Run the command to check if AKS has permission to pull from ACR

|  |
| --- |
| az aks check-acr -g aks-acr-rg -n myaks118 --acr myacr118.azurecr.io |

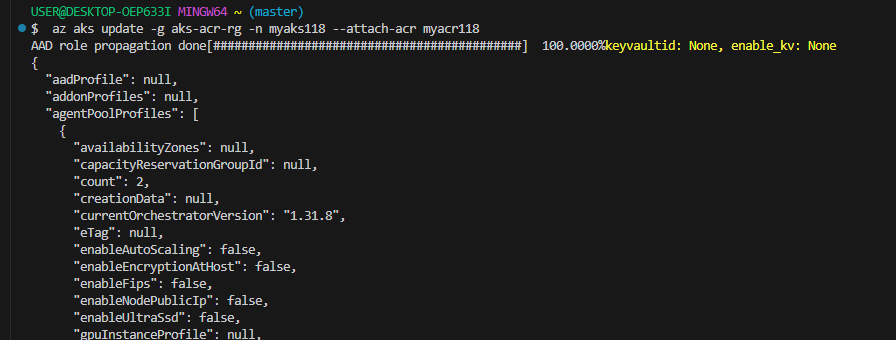
From the screenshot, it shows FAILED and also rejected token exchange which means it’s still unauthorized. Let’s create a pod and verify that it still doesn’t have permission to pull an image.

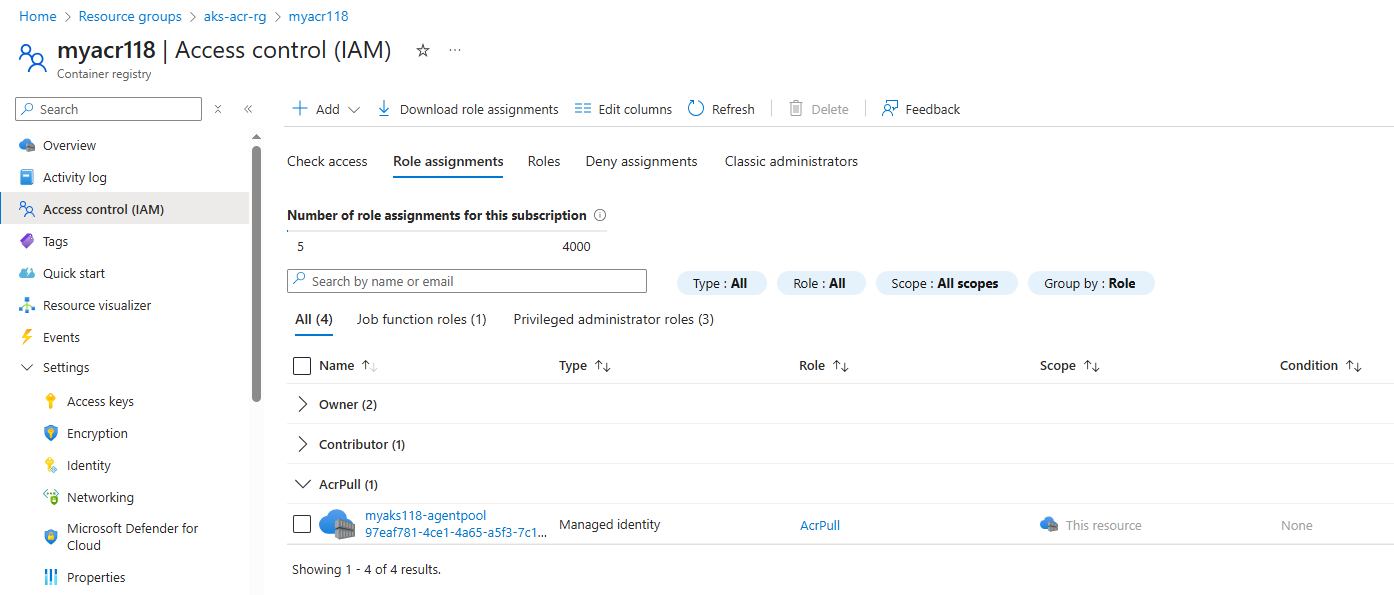
|  |
| --- |
| kubectl run pod-azure-method-httpd --image myacr118.azurecr.io/httpd:v1 |

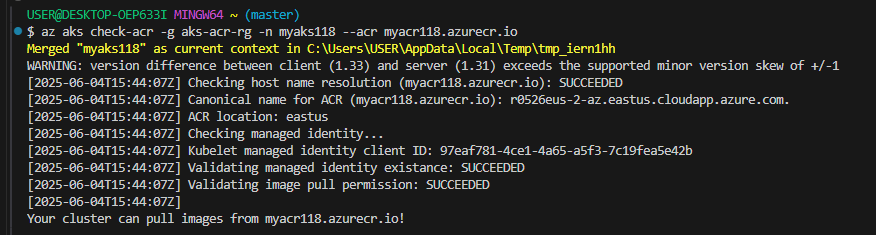
Describe the pod and view its log

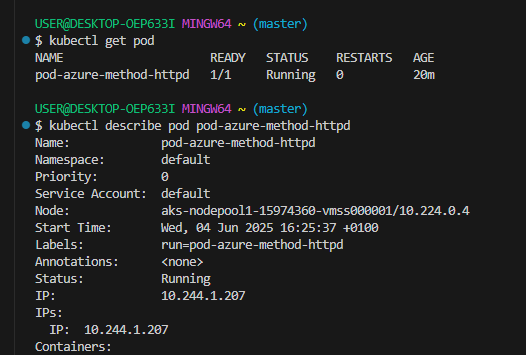
Attach your ACR to your AKS cluster

|  |
| --- |
| az aks update -g aks-acr-rg -n myaks118 --attach-acr myacr118 |

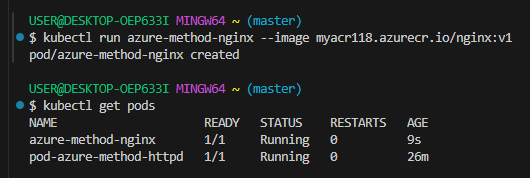
Once you run this, it’s going to create the role with permission

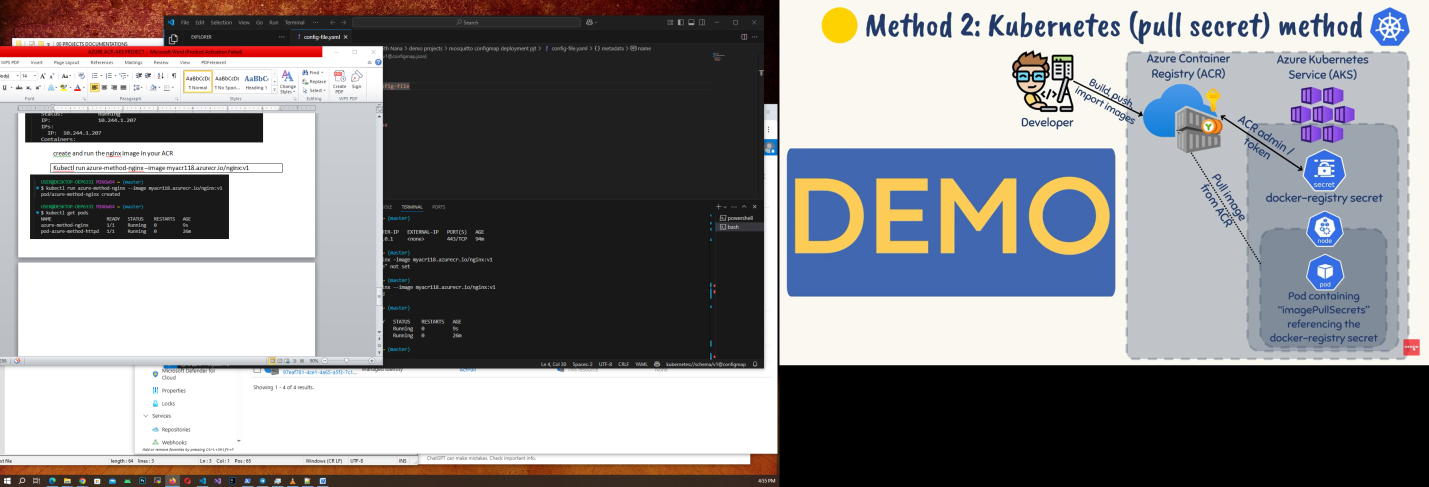
Lets go to the portal and verify the created role/assigned permission.

Check on the terminal again and see if AKS now has permission to ACR

create and run the nginx image in your ACR

|  |
| --- |
| Kubectl run azure-method-nginx –image myacr118.azurecr.io/nginx:v1 |





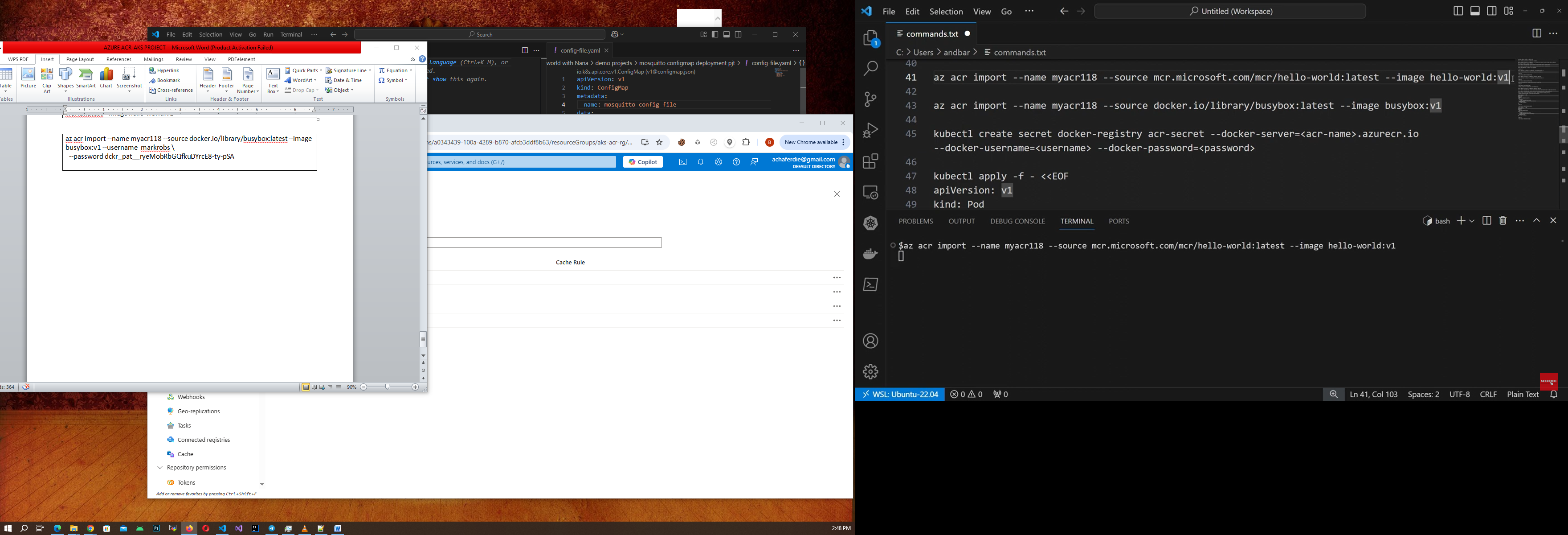
Delete pods and detach role before beginning this second method

|  |
| --- |
| $ az aks update -g aks-acr-rg -n myaks118 --detach-acr myacr118 |

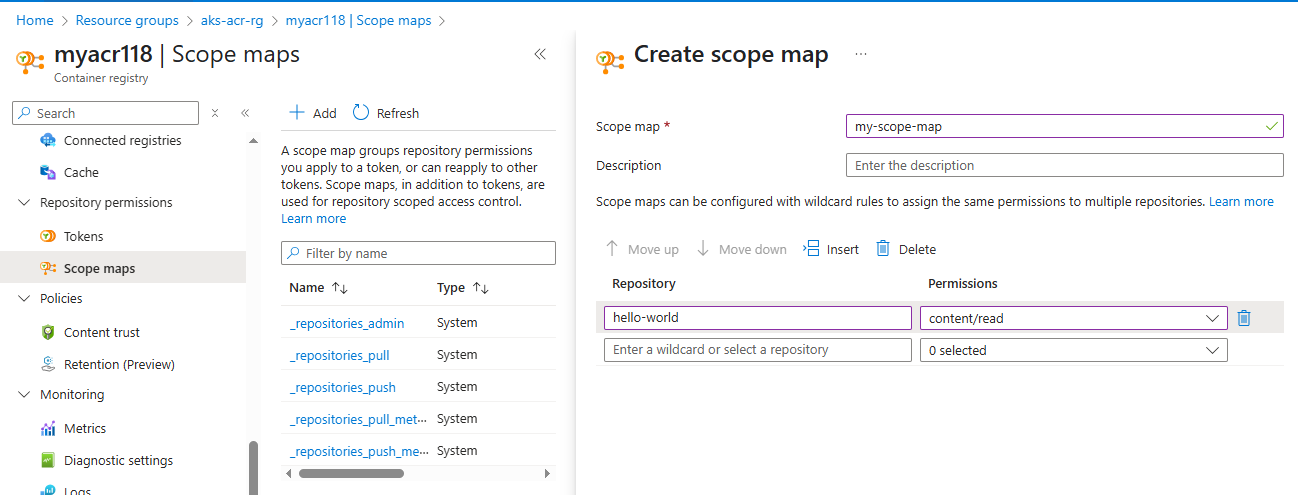
Import the following images to your ACR

|  |
| --- |
| az acr import --name myacr118 --source mcr.microsoft.com/mcr/hello-world:latest --image hello-world:v1 |

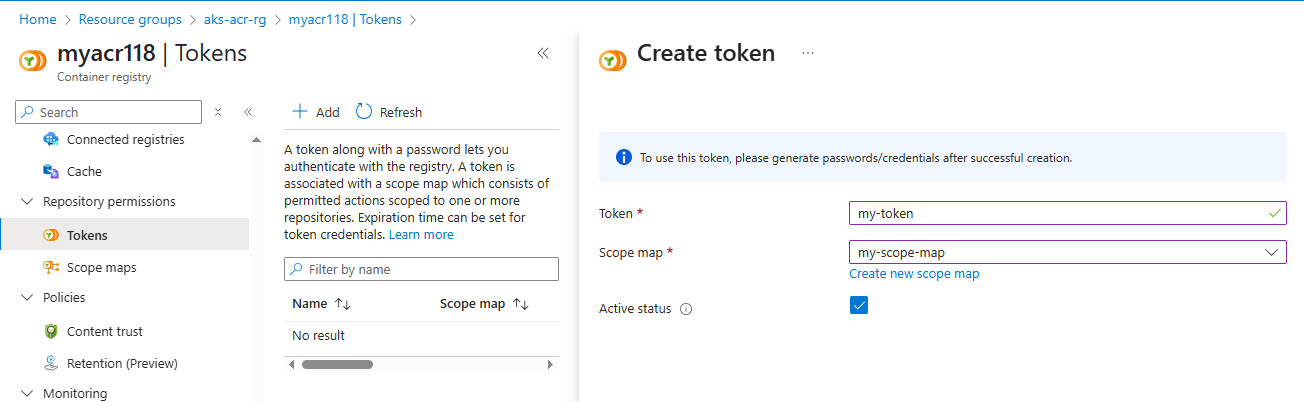
|  |
| --- |
| az acr import --name myacr118 --source docker.io/library/busybox:latest --image busybox:v1 --username markrobs \  --password dckr\_pat\_\_ryeMobRbGQfkuDYrcE8-ty-pSA |



Create a scope map to allow “hello-world” repo with content/read permission



Then create a token and associate it with the scope-map which you just created.



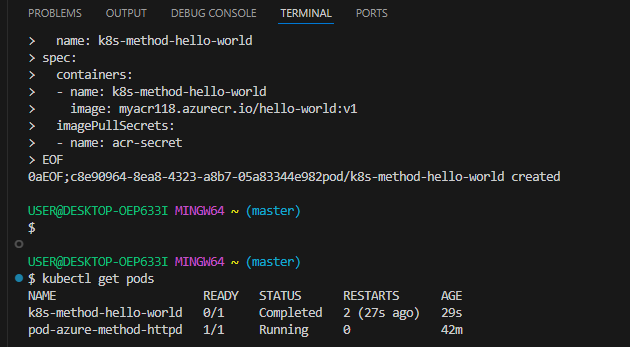
|  |
| --- |
| Kubectl create secret docker-registry acr-secret --docker-server=myacr118.azurecr.io --docker-username=my-token --docker-password= tDfKlEffAWqyYoHzBget7Bn/p85eDekr4c+tT0NMRd+ACRD0Athf |

#To view secret, run

$ kubectl get secret

Inline pod YAML with pull secret

|  |
| --- |
| kubectl apply -f - <<EOF  apiVersion: v1  kind: Pod  metadata:  name: k8s-method-hello-world  spec:  containers:  - name: k8s-method-hello-world  image: myacr118.azurecr.io/hello-world:v1  imagePullSecrets:  - name: acr-secret  EOF |



Create another pod for the busybox pod

|  |
| --- |
| kubectl apply -f - <<EOF  apiVersion: v1  kind: Pod  metadata:  name: k8s-method-busybox  spec:  containers:  - name: k8s-method-hello-world  image: myacr118.azurecr.io/busybox:v1  imagePullSecrets:  - name: acr-secret  EOF |