ASSIGNMENT TWO: CONFIGURING AND SECURING ACR AND AKS WRITEUP

By: CS-CNS03-23082 – ABUOR ISABELLA MERCY

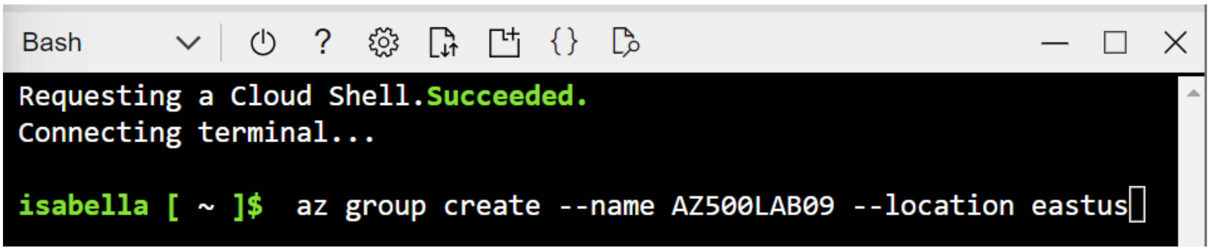
Introduction

In this assignment, we focused on the creation of an Azure Container Registry (ACR) and its integration with an Azure Kubernetes Service (AKS) cluster. The assignment was divided into a series of comprehensive tasks from creating an Azure Container Registry and Azure Kubernetes service cluster, building and pushing containers to ACR to deploying internal and external services to them. The following is a descriptive writeup of how the lab was completed.

Task 1: Create an Azure Container Registry

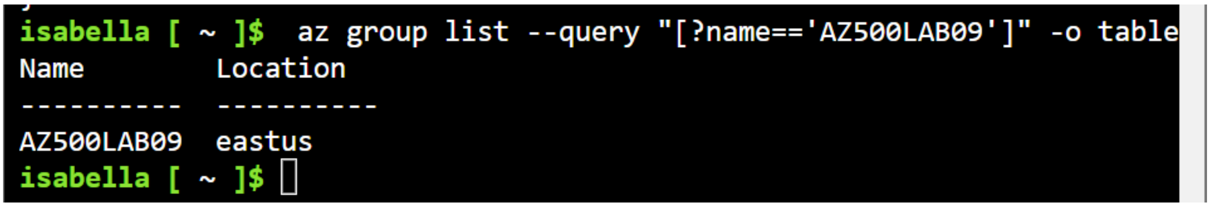
In this task, we created a resource group for the lab and an Azure Container Registry through the bash in the cloud shell.

To create the resource group through the bash the following command was used:



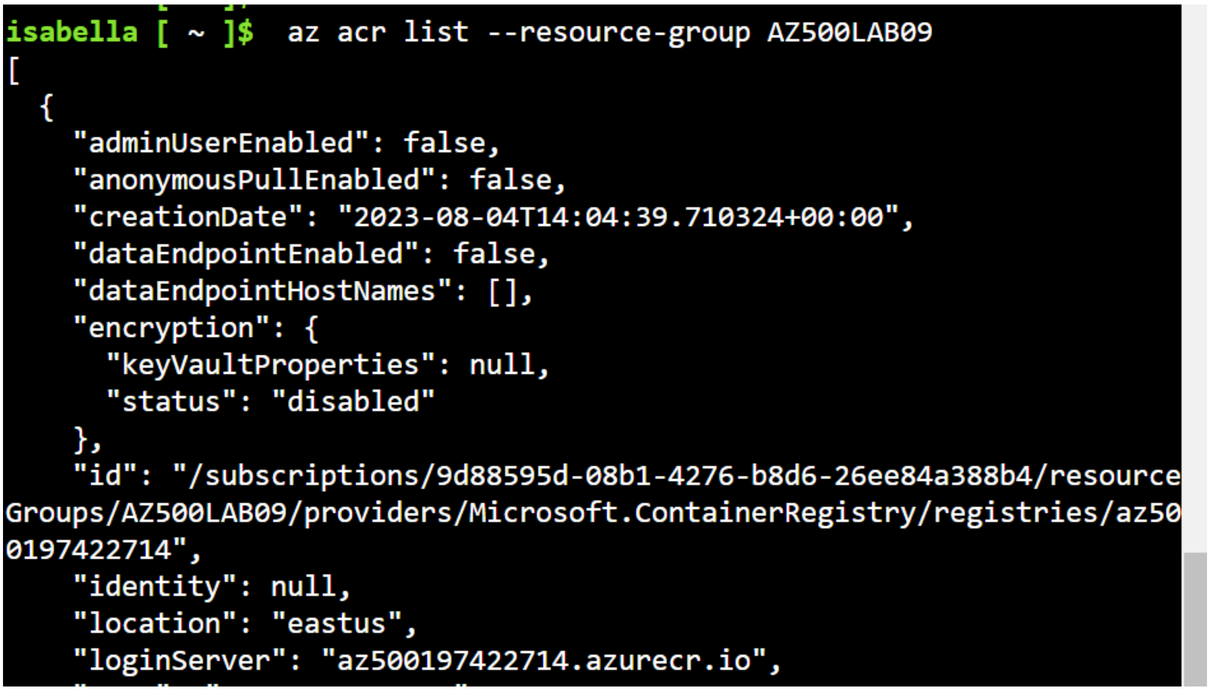
```
Requesting a Cloud Shell.Succeeded.
Connecting terminal...

isabella [ ~ ]$  az group create --name AZ500LAB09 --location eastus
```

To verify that the resource group was created the following command was prompted:



```
isabella [ ~ ]$  az group list --query "[?name=='AZ500LAB09']" -o table
Name        Location
----------  ----------
AZ500LAB09  eastus
isabella [ ~ ]$
```

In the Bash session within the Cloud Shell pane, the following code was used to create a new Azure Container Registry (ACR) instance.



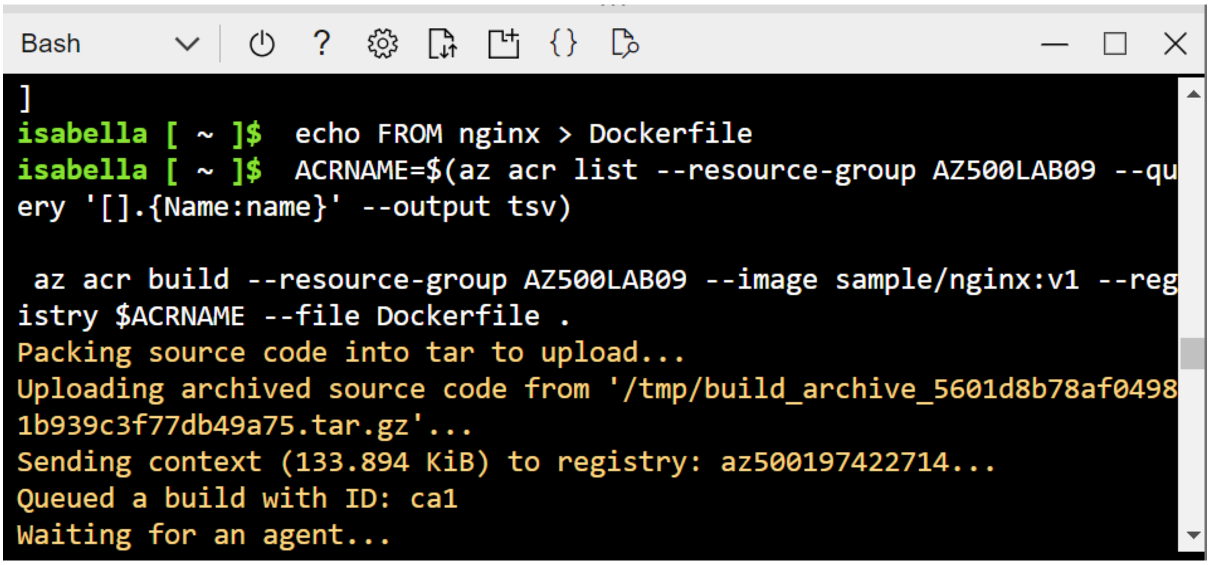
```
isabella [ ~ ]$  az acr list --resource-group AZ500LAB09
[
  {
    "adminUserEnabled": false,
    "anonymousPullEnabled": false,
    "creationDate": "2023-08-04T14:04:39.710324+00:00",
    "dataEndpointEnabled": false,
    "dataEndpointHostNames": [],
    "encryption": {
      "keyVaultProperties": null,
      "status": "disabled"
    },
    "id": "/subscriptions/9d88595d-08b1-4276-b8d6-26ee84a388b4/resource
Groups/AZ500LAB09/providers/Microsoft.ContainerRegistry/registries/az50
0197422714",
    "identity": null,
    "location": "eastus",
    "loginServer": "az500197422714.azurecr.io",
```

Task 2: Create a Dockerfile, build a container and push it to Azure Container Registry

In this task, we created a Dockerfile, builded an image from the Dockerfile, and deployed the image to the ACR.

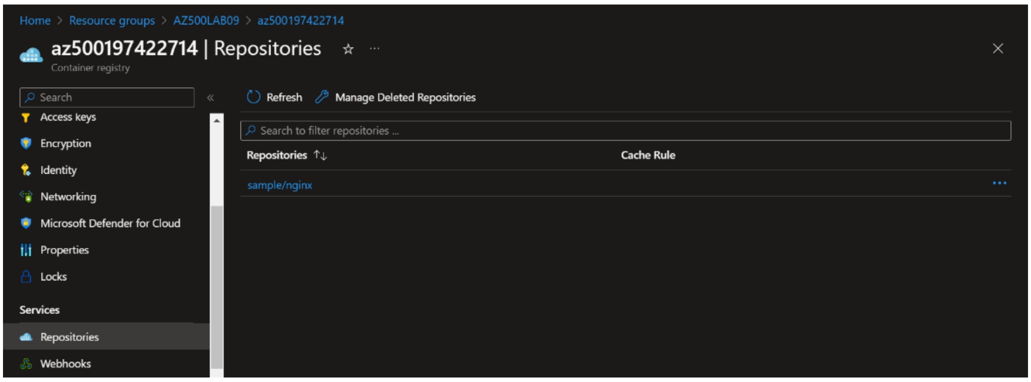
Firstly, in the Bash session within the Cloud Shell pane, the following code was used to create a Dockerfile to create an Nginx-based image:

Secondly, the second code was used to build an image from the Dockerfile and push the image to the new ACR.



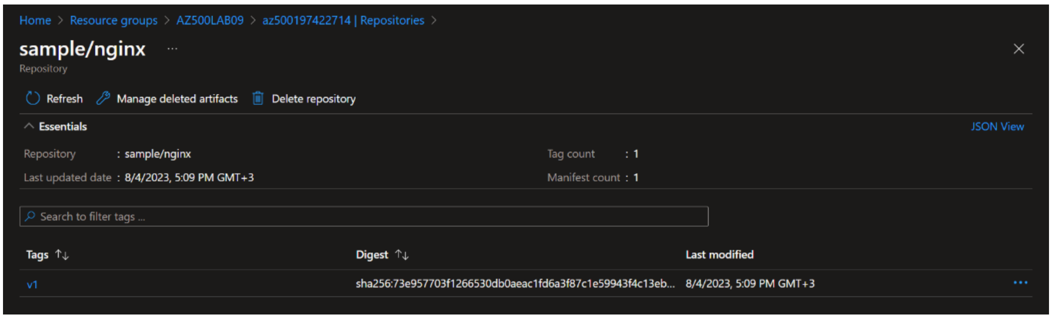
```
]
isabella [ ~ ]$  echo FROM nginx > Dockerfile
isabella [ ~ ]$  ACRNAME=$(az acr list --resource-group AZ500LAB09 --qu
ery '[].{Name:name}' --output tsv)

 az acr build --resource-group AZ500LAB09 --image sample/nginx:v1 --reg
istry $ACRNAME --file Dockerfile .
Packing source code into tar to upload...
Uploading archived source code from '/tmp/build_archive_5601d8b78af0498
1b939c3f77db49a75.tar.gz'...
Sending context (133.894 KiB) to registry: az500197422714...
Queued a build with ID: ca1
Waiting for an agent...
```

Then in the Azure portal, we confirmed that the list of repositories includes the new container image named sample/nginx on **AZ500Lab09** resource group.


Home > Resource groups > AZ500LAB09 > az500197422714
az500197422714 | Repositories
Container registry
Search
Access keys
Encryption
Identity
Networking
Microsoft Defender for Cloud
Properties
Locks
Services
Repositories
Webhooks
Refresh
Manage Deleted Repositories
Search to filter repositories ...

Repositories ↑↓	Cache Rule
sample/nginx	

And on the **sample/nginx** entry, we confirmed the presence of the **v1** tag which identifies the image version.


Home > Resource groups > AZ500LAB09 > az500197422714 | Repositories >
sample/nginx
Repository
Refresh
Manage deleted artifacts
Delete repository
Essentials
JSON View
Search to filter tags ...

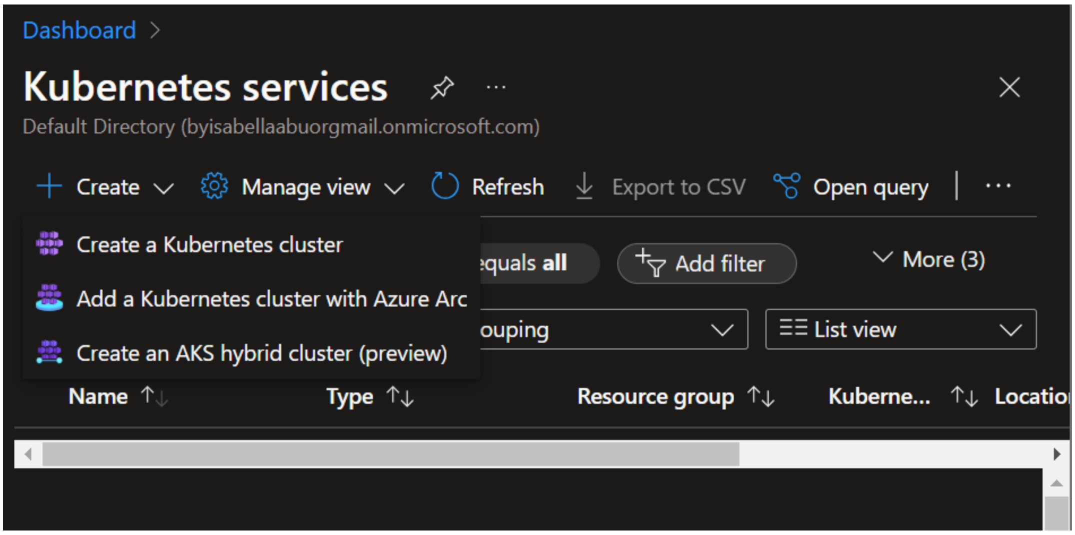
Repository : sample/nginx
Last updated date : 8/4/2023, 5:09 PM GMT+3
Tag count : 1
Manifest count : 1

Tags ↑↓	Digest ↑↓	Last modified
v1	sha256:73e957703f1266530db0aeac1fd6a3f87c1e59943f4c13eb...	8/4/2023, 5:09 PM GMT+3

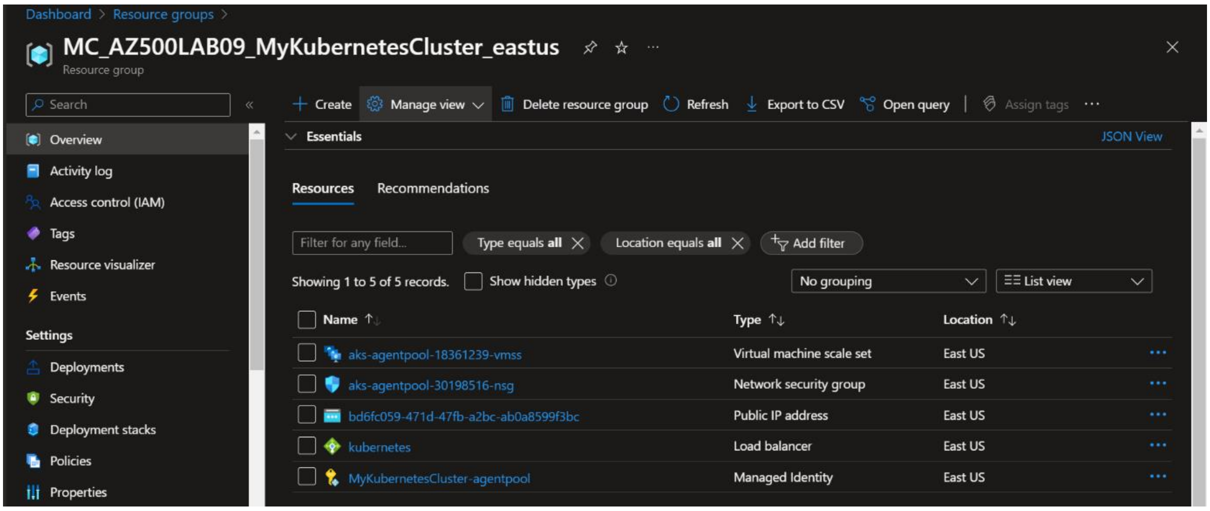
Task 3: Create an Azure Kubernetes Service cluster

In this task, we created an Azure Kubernetes service and review the deployed resources.

On **Kubernetes services** section click + **Create** and, **created a Kubernetes cluster and specified some settings on the kuberneters cluster for example the** Kubernetes cluster name, Node pools, Network configuration among others.



And to verify that the cluster was created and deployed, we went and confirmed in the resource group



MC_AZ500LAB09_MyKubernetesCluster_eastus

Type equals all

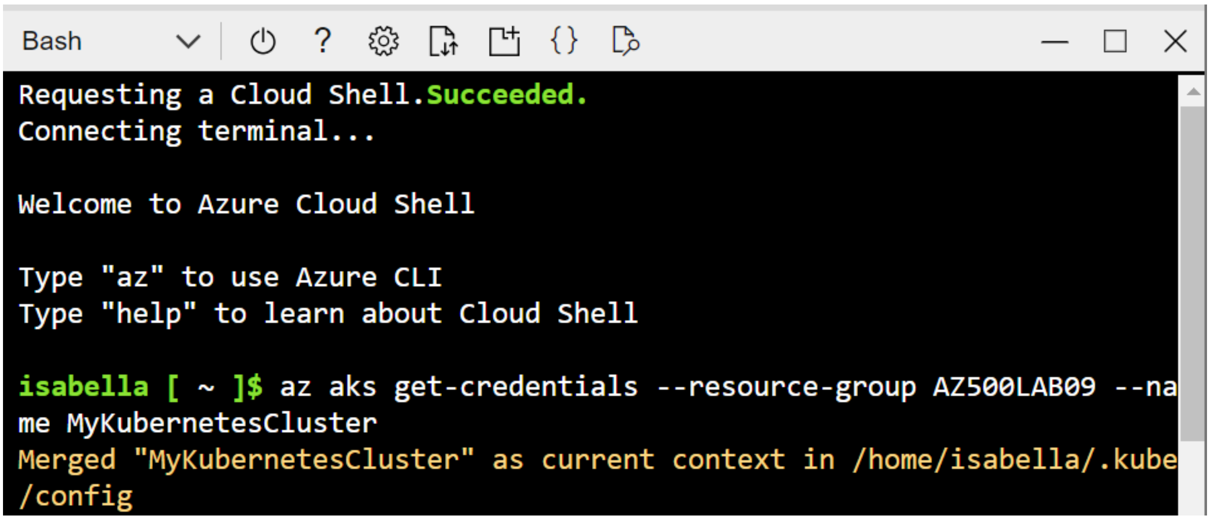
Location equals all

Showing 1 to 5 of 5 records.

No grouping

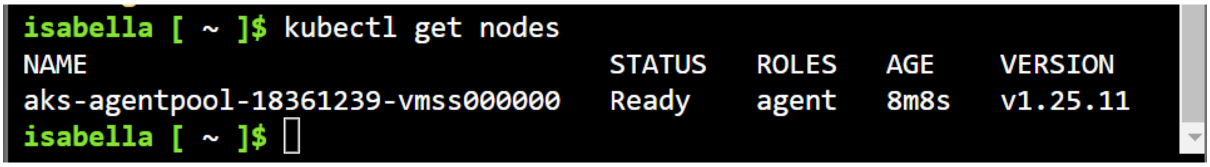
Name	Type	Location
aks-agentpool-18361239-vmss	Virtual machine scale set	East US
aks-agentpool-30198516-nsg	Network security group	East US
bd6fc059-471d-47fb-a2bc-ab0a8599f3bc	Public IP address	East US
kubernetes	Load balancer	East US
MyKubernetesCluster-agentpool	Managed Identity	East US

In a Bash session in the Cloud Shell, the following command was prompted to connect to the Kubernetes cluster:



```
Requesting a Cloud Shell.Succeeded.
Connecting terminal...

Welcome to Azure Cloud Shell

Type "az" to use Azure CLI
Type "help" to learn about Cloud Shell

isabella [ ~ ]$ az aks get-credentials --resource-group AZ500LAB09 --na
me MyKubernetesCluster
Merged "MyKubernetesCluster" as current context in /home/isabella/.kube
/config
```

To list nodes of the Kubenetes cluster the following code was prompted:

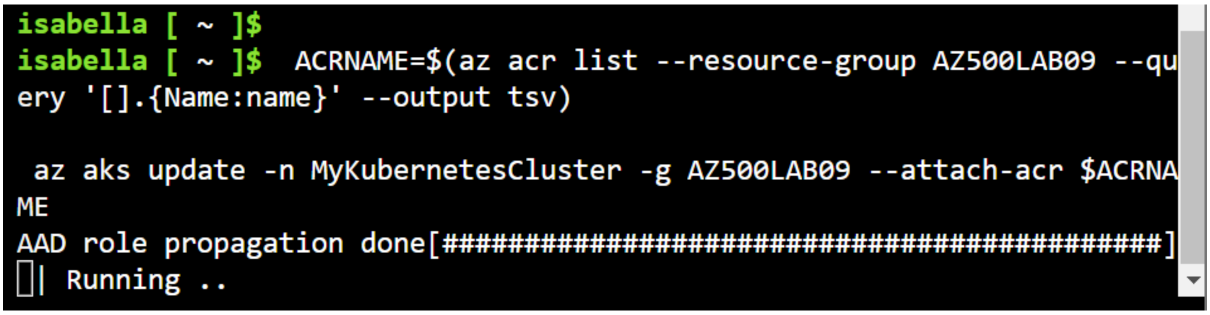


```
isabella [ ~ ]$ kubectl get nodes
NAME                                STATUS   ROLES   AGE    VERSION
aks-agentpool-18361239-vmss000000   Ready    agent   8m8s   v1.25.11
isabella [ ~ ]$ 
```

Task 4: Grant the AKS cluster permissions to access the ACR and manage its virtual network

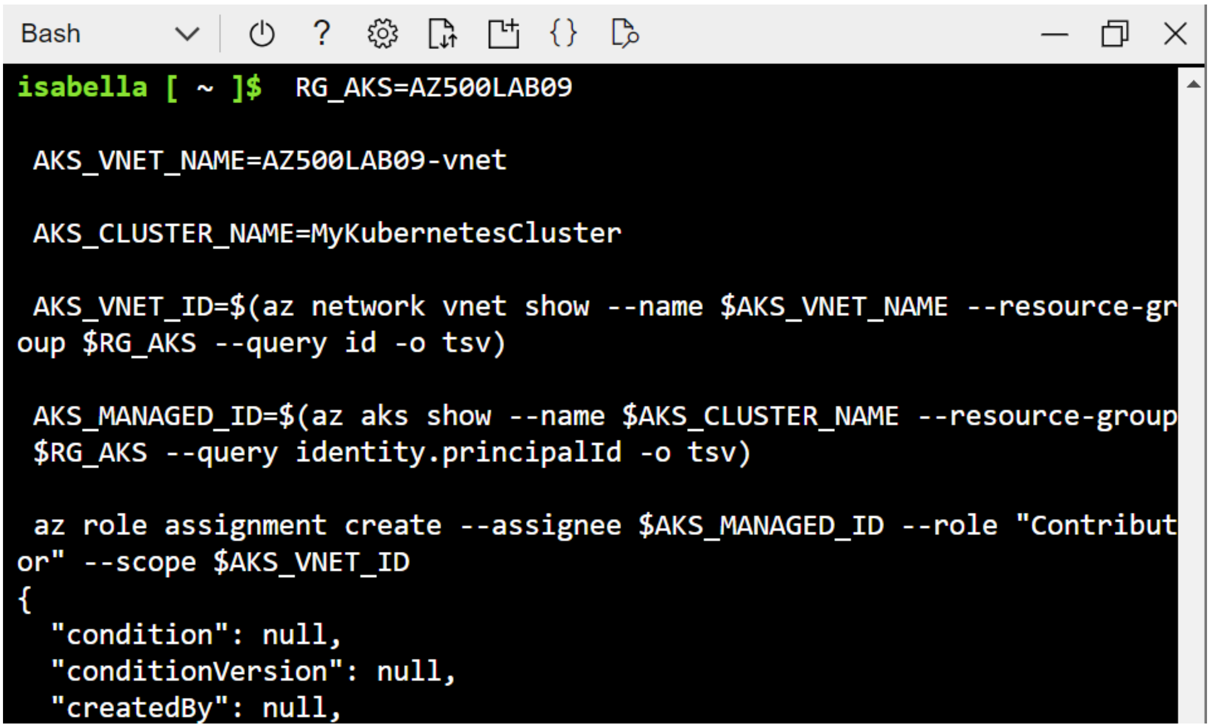
In this task, we granted the AKS cluster permission to access the ACR and manage its virtual network.

In the Bash session within the Cloud Shell pane, the following code was used to configure the AKS cluster to use the Azure Container Registry instance you created earlier in this lab.



```
isabella [ ~ ]$
isabella [ ~ ]$  ACRNAME=$(az acr list --resource-group AZ500LAB09 --qu
ery '[].{Name:name}' --output tsv)

 az aks update -n MyKubernetesCluster -g AZ500LAB09 --attach-acr $ACRNA
ME
AAD role propagation done[############################################]
▯| Running ..
```

The following code was used to grant the AKS cluster the Contributor role to its virtual network.

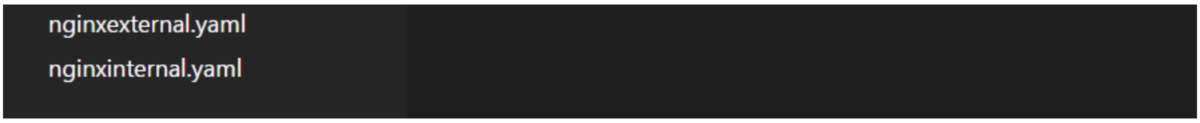


```
isabella [ ~ ]$  RG_AKS=AZ500LAB09

 AKS_VNET_NAME=AZ500LAB09-vnet

 AKS_CLUSTER_NAME=MyKubernetesCluster

 AKS_VNET_ID=$(az network vnet show --name $AKS_VNET_NAME --resource-gr
oup $RG_AKS --query id -o tsv)

 AKS_MANAGED_ID=$(az aks show --name $AKS_CLUSTER_NAME --resource-group
 $RG_AKS --query identity.principalId -o tsv)

 az role assignment create --assignee $AKS_MANAGED_ID --role "Contribut
or" --scope $AKS_VNET_ID
{
  "condition": null,
  "conditionVersion": null,
  "createdBy": null,
```

Task 5: Deploy an external service to AKS

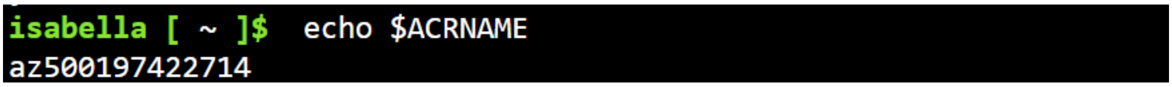
In this task, we downloaded the Manifest files, edit the YAML file, and apply your changes to the cluster.

Firstly, in the Bash session within the Cloud Shell pane, we downloaded the Manifest files:



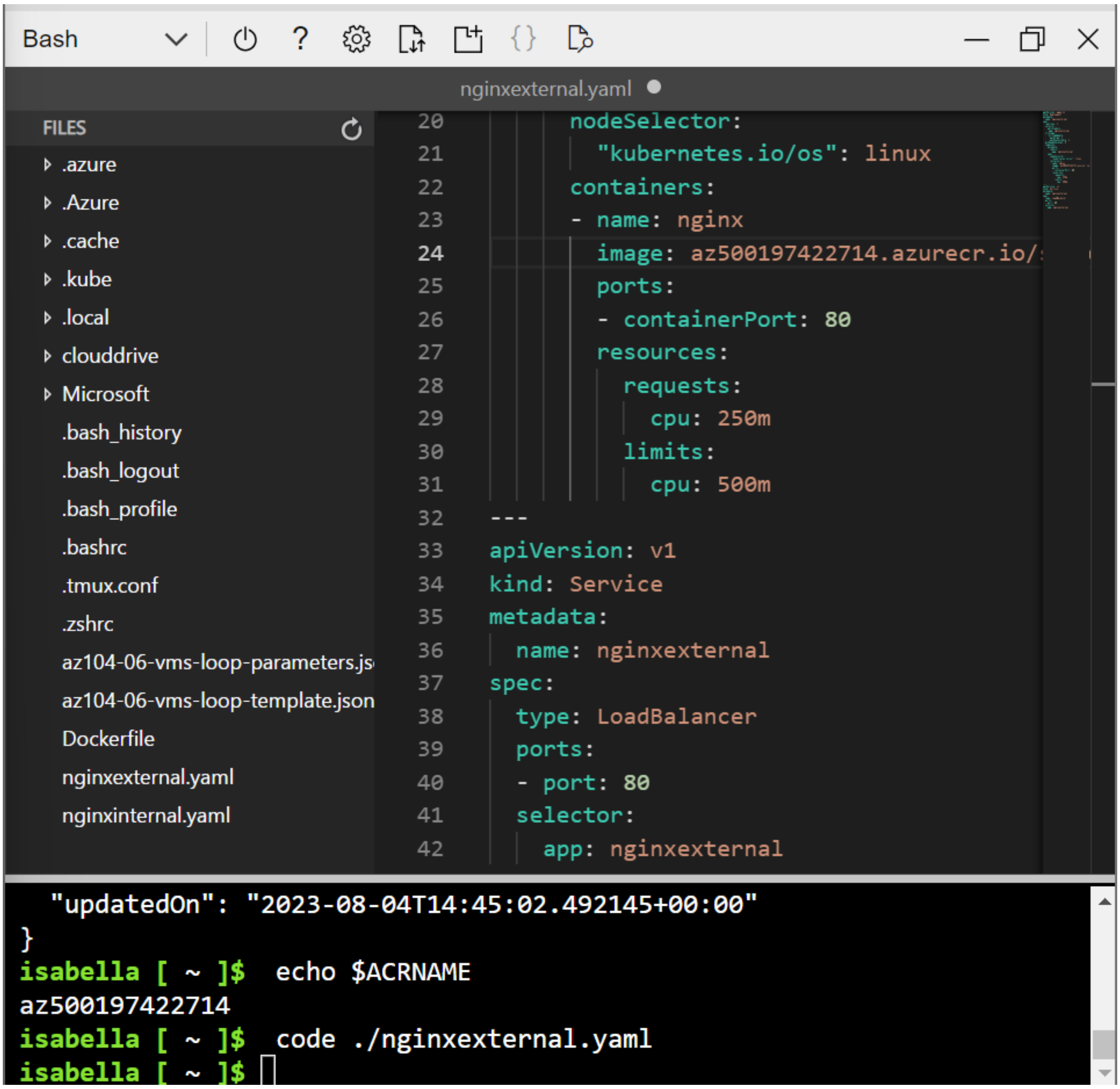
```
nginxexternal.yaml
nginxinternal.yaml
```

In the Bash session within the Cloud Shell pane, the following code was prompted to identify the name of the Azure Container Registry instance:



```
isabella [ ~ ]$  echo $ACRNAME
az500197422714
```

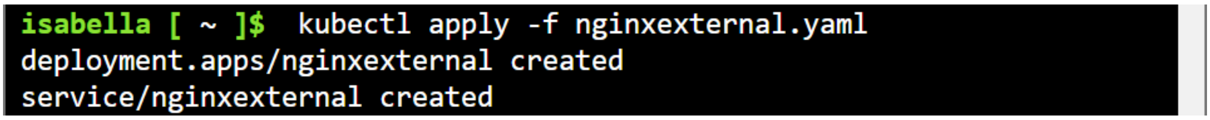
We then used the following code to open the nginxexternal.yaml file, so that we can edit **line 24** and replace the **<ACRUniquename>** placeholder with the ACR name.



```
20        nodeSelector:
21          "kubernetes.io/os": linux
22        containers:
23        - name: nginx
24          image: az500197422714.azurecr.io/
25          ports:
26          - containerPort: 80
27          resources:
28            requests:
29              cpu: 250m
30            limits:
31              cpu: 500m
32  ---
33  apiVersion: v1
34  kind: Service
35  metadata:
36    name: nginxexternal
37  spec:
38    type: LoadBalancer
39    ports:
40    - port: 80
41    selector:
42      app: nginxexternal
```

```
  "updatedOn": "2023-08-04T14:45:02.492145+00:00"
}
isabella [ ~ ]$  echo $ACRNAME
az500197422714
isabella [ ~ ]$  code ./nginxexternal.yaml
isabella [ ~ ]$ 
```

The following code was prompted to apply the change to the cluster:

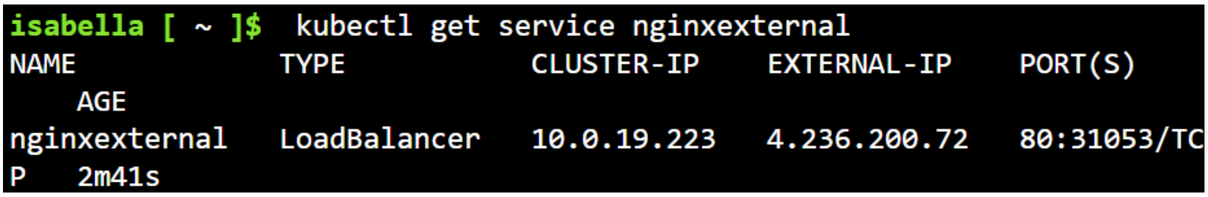


```
isabella [ ~ ]$  kubectl apply -f nginxexternal.yaml
deployment.apps/nginxexternal created
service/nginxexternal created
```

Task 6: Verification that one can access an external AKS-hosted service

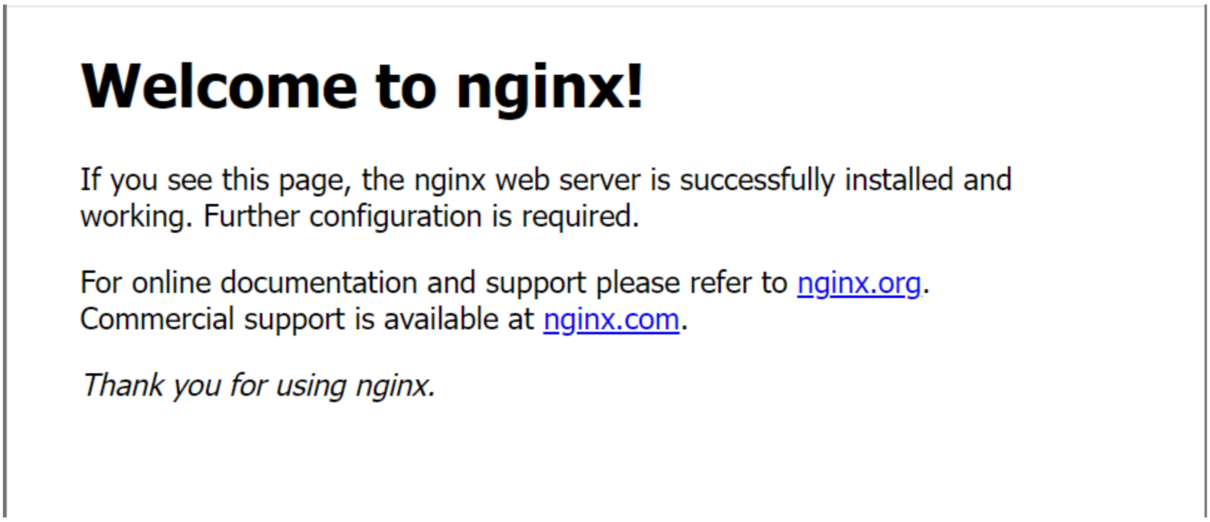
In this task, verification of the container can be accessed externally using the public IP address.

In the Bash session within the Cloud Shell pane, the following code was used to retrieve information about the nginxexternal service including name, type, IP addresses, and ports.



```
isabella [ ~ ]$  kubectl get service nginxexternal
NAME             TYPE           CLUSTER-IP     EXTERNAL-IP    PORT(S)
    AGE
nginxexternal   LoadBalancer   10.0.19.223   4.236.200.72   80:31053/TC
P    2m41s
```

On a new browser tab and we browsed to the IP address you identified in the previous step and the following were the results:



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

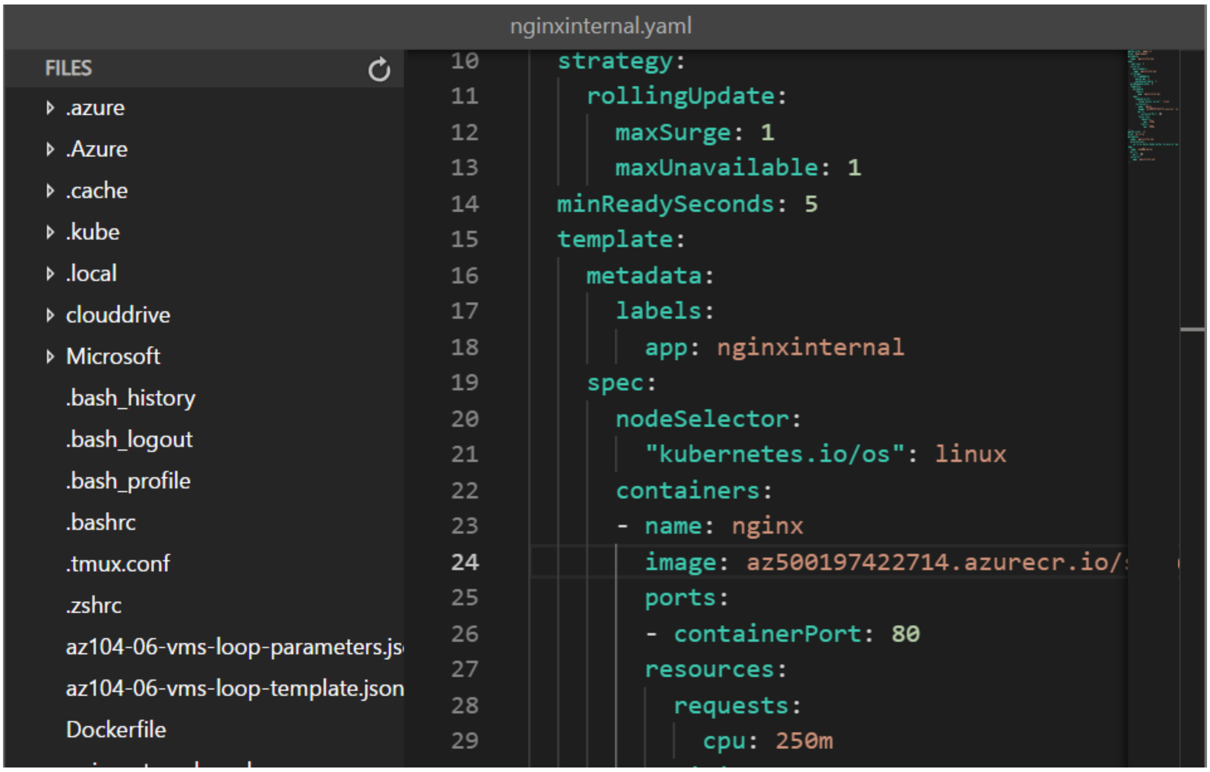
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Task 7: Deploy an internal service to AKS

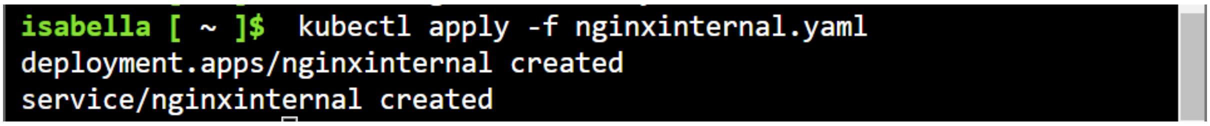
In this task, you will deploy the internal facing service on the AKS.

The following code was prompted to open the nginxintenal.yaml file, so you can edit the line containing the reference to the container image and replace the **<ACRUniquename>** placeholder with the ACR name.



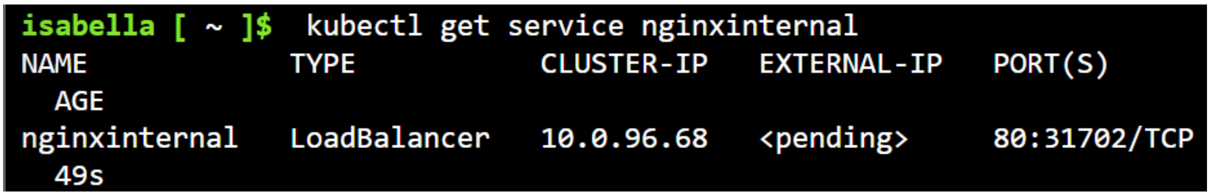
```
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1
  minReadySeconds: 5
  template:
    metadata:
      labels:
        app: nginxinternal
    spec:
      nodeSelector:
        "kubernetes.io/os": linux
      containers:
      - name: nginx
        image: az500197422714.azurecr.io/
        ports:
        - containerPort: 80
        resources:
          requests:
            cpu: 250m
```

In the Bash session within the Cloud Shell pane, the following was done to apply the change to the cluster:



```
isabella [ ~ ]$  kubectl apply -f nginxinternal.yaml
deployment.apps/nginxinternal created
service/nginxinternal created
```

To review the output to verify your deployment and the service have been created:

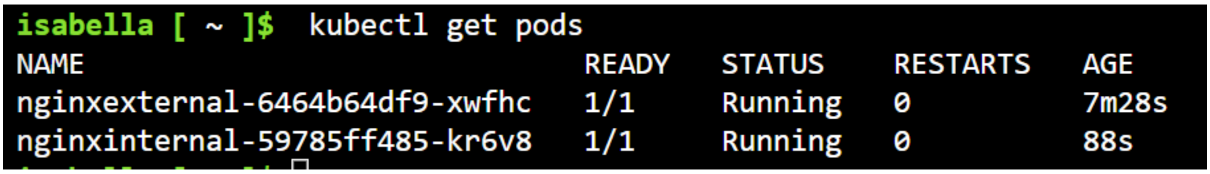


```
isabella [ ~ ]$  kubectl get service nginxinternal
NAME           TYPE           CLUSTER-IP    EXTERNAL-IP    PORT(S)
  AGE
nginxinternal  LoadBalancer   10.0.96.68    <pending>      80:31702/TCP
  49s
```

Task 8: Verify the you can access an internal AKS-hosted service

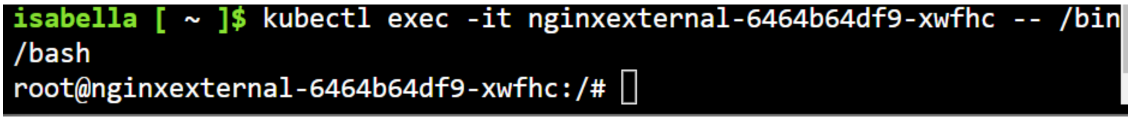
In this task, we use one of the pods running on the AKS cluster to access the internal service.

In the Bash session within the Cloud Shell pane, the following code was used to list the pods in the default namespace on the AKS cluster:



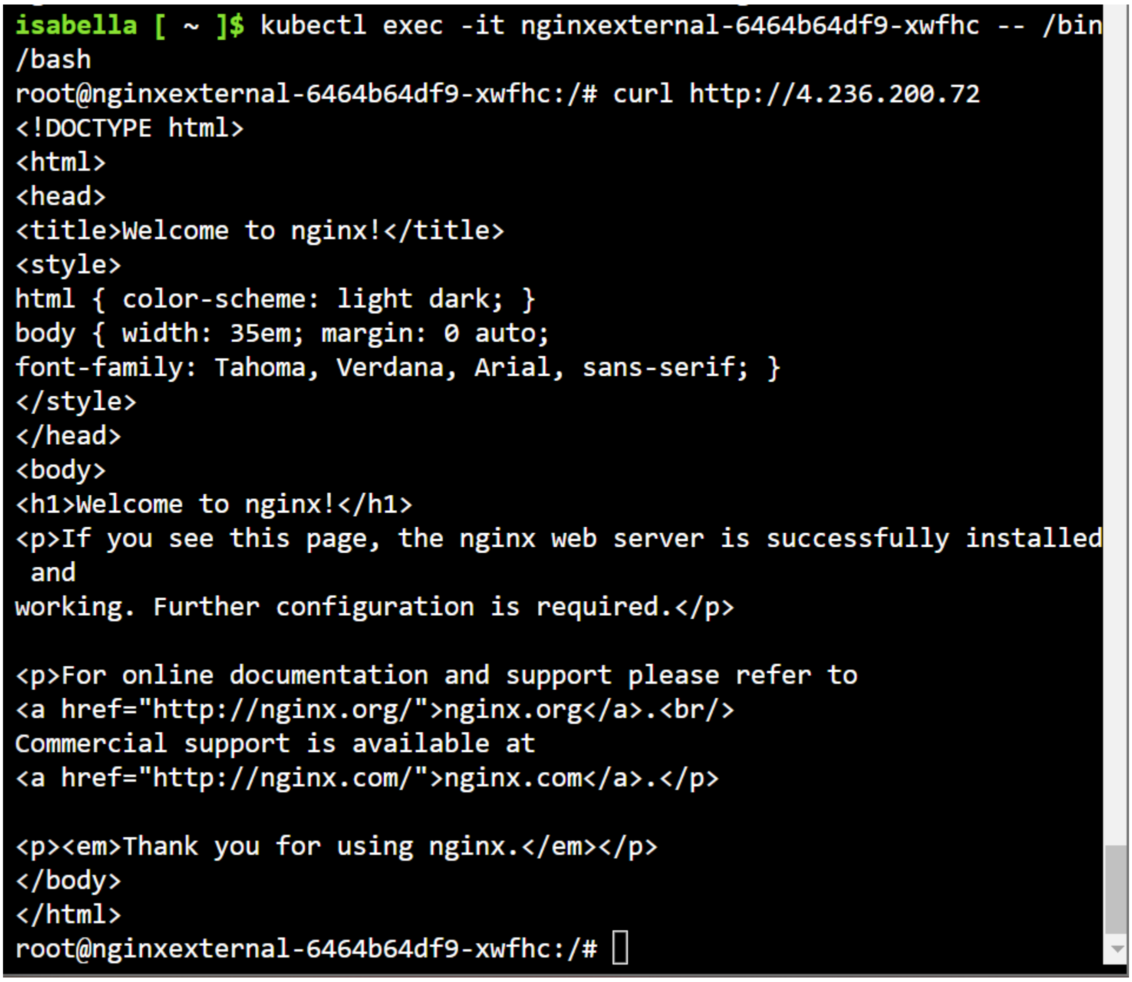
```
isabella [ ~ ]$  kubectl get pods
NAME                            READY   STATUS    RESTARTS   AGE
nginxexternal-6464b64df9-xwfhc  1/1     Running   0          7m28s
nginxinternal-59785ff485-kr6v8  1/1     Running   0          88s
```

In the Bash session within the Cloud Shell pane, the following code was prompted to connect interactively to the first pod (replace the <pod_name> placeholder with the name you copied in the previous step):



```
isabella [ ~ ]$ kubectl exec -it nginxexternal-6464b64df9-xwfhc -- /bin
/bash
root@nginxexternal-6464b64df9-xwfhc:/# 
```

To verify that the nginx web site is available via the private IP address of the service we replaced the <internal_IP> placeholder with the IP address you recorded in the previous task and the following was the result:



```
isabella [ ~ ]$ kubectl exec -it nginxexternal-6464b64df9-xwfhc -- /bin
/bash
root@nginxexternal-6464b64df9-xwfhc:/# curl http://4.236.200.72
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed
 and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
root@nginxexternal-6464b64df9-xwfhc:/# 
```

Conclusion

In conclusion, this assignment has provided a comprehensive and hands-on exploration of the intricate interplay between Azure Container Registry (ACR) and Azure Kubernetes Service (AKS). By creating an ACR, we established a secure repository for Docker images, enabling efficient version control, while the integration with AKS showcased Kubernetes' orchestration capabilities for scalable application management. The assignment was a good start to deepening my understanding of cloud-native deployment.