

Microsoft ADC Cybersecurity Skilling Program

Week 8 Lab Assignment

Student Name: Vincent Onchieku Collins

Student ID: ADC-CSS02-25052

Introduction

The lab exercise titled "Configuring and Securing ACR and AKS" was designed to provide hands-on experience with setting up and securing containerized workloads in Microsoft Azure. This lab focused on the integration between Azure Container Registry (ACR) and Azure Kubernetes Service (AKS), demonstrating how to build container images, store them securely, and deploy them to a Kubernetes cluster. Through a series of eight sequential tasks, I learned how to provision Azure resources, build and push Docker images, configure Kubernetes clusters, and deploy both internal and external services. This lab is essential for understanding how to securely manage and scale containerized applications in a cloud-native environment. Below is a breakdown of each task and what I accomplished. It included the following tasks:

Task 1: Create an Azure Container Registry

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

Task 3: Create an Azure Kubernetes Service cluster

Task 4: Grant the AKS cluster permissions to access the ACR

Task 5: Deploy an external service to AKS

Task 6: Verify you can access an external AKS-hosted service

Task 7: Deploy an internal service to AKS

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

Tasks:

Task 1: Create an Azure Container Registry

The lab began by signing into the Azure Portal and launching the Cloud Shell with Bash selected. I then created a new resource group named AZ500LAB09 in the East US region using the Azure CLI. After confirming the group was successfully created, I provisioned a new Azure Container Registry (ACR) with a unique name using the `az acr create` command. The ACR was created with the Basic SKU, which is ideal for development or testing purposes. Finally, I confirmed the creation by listing all ACRs in the resource group and noting the exact name for use in subsequent tasks.

The screenshot displays the Azure Portal interface with the Cloud Shell open. The Cloud Shell terminal shows the following commands and output:

```
"lastModifiedAt": "2025-05-28T18:13:31.649009+00:00",
"lastModifiedBy": "LabUser-51743740@LODSPROD-MCA.ODSPRODMCA.onmicrosoft.com",
"lastModifiedByType": "User"
},
"tags": {},
"type": "Microsoft.ContainerRegistry/registries",
"zoneRedundancy": "Disabled"
}
labuser-51743740 [ ~ ]$ az acr list --resource-group AZ500LAB09
[
  {
    "adminUserEnabled": false,
    "anonymousPullEnabled": false,
```

The right-hand pane shows the lab instructions:

- 6. In the Bash session within the Cloud Shell pane, run the following to create a new Azure Container Registry (ACR) instance (The name of the ACR must be globally unique):
`az acr create --resource-group AZ500`
- 7. In the Bash session within the Cloud Shell pane, run the following to confirm that the new ACR was created:
`az acr list --resource-group AZ500LA`
Record the name of the ACR. You will need it in the next task.

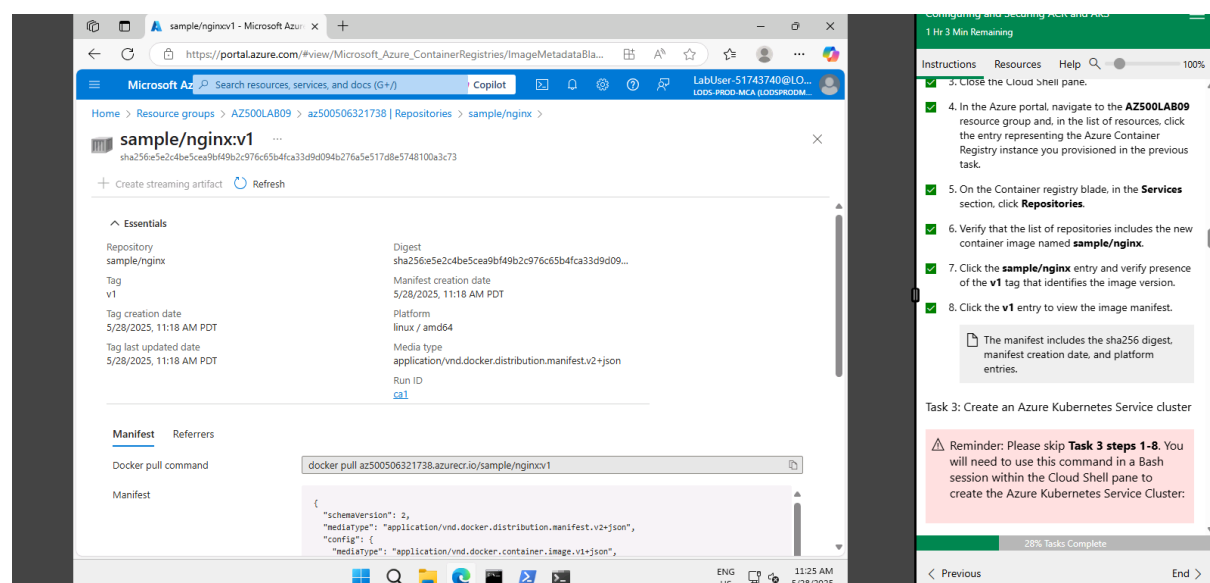
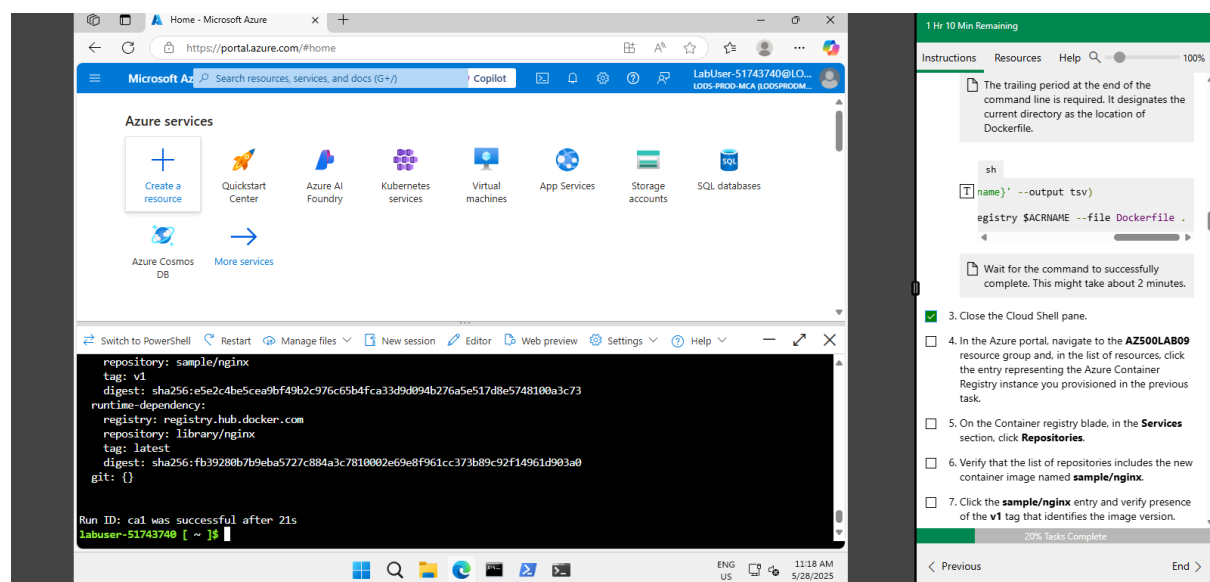
Task 2: Create a Dockerfile, build a container and push it to Azure Container Registry

In this task, you will create a Dockerfile, build an image from the Dockerfile, and deploy the image to the ACR.

- 1. In the Bash session within the Cloud Shell pane, run the following to create a Dockerfile to create an Nginx-based image:

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

Next, I created a Dockerfile directly within the Cloud Shell environment. The Dockerfile was a basic configuration using the official Nginx image as its base. I then built a Docker container image from this file and pushed it to the ACR created earlier using the `az acr build` command. This command automatically built the image and uploaded it to the registry. Once the process was complete, I navigated to the Azure Portal, found the ACR, and verified that the `sample/nginx:v1` image was listed in the repositories section. I also inspected the image manifest to review details such as digest and creation date.



Task 3: Create an Azure Kubernetes Service cluster

Following that, I created a new Azure Kubernetes Service (AKS) cluster via the Azure Portal. The cluster was named MyKubernetesCluster and was deployed in the East US region within the AZ500LAB09 resource group. I opted for a Dev/Test preset with one node and disabled features like virtual nodes and container monitoring to keep the configuration simple. After submitting the deployment, I waited for the AKS resources to be provisioned. Once deployed, I located a new resource group MC_AZ500LAB09_MyKubernetesCluster_eastus containing the infrastructure for the AKS cluster, including nodes and virtual networks.

The screenshot shows the Azure Portal interface for the 'MyKubernetesCluster' resource. The left sidebar contains navigation options like Overview, Activity log, Access control (IAM), Tags, Monitor, Diagnose and solve problems, Microsoft Defender for Cloud (preview), Cost analysis, Resource visualizer, Kubernetes resources, Settings, Monitoring, Automation, and Helm. The main content area displays the 'Overview' page for the cluster, showing details such as Resource group (AZ500LAB09), Kubernetes version (1.31.8), API server address, Power state (Running), Cluster operation status (Succeeded), Subscription (MOC Subscription-lod50651563), Location (East US), Subscription ID (9030ddef-445a-4a0a-9d99-b1ba7b5b0748), Fleet Manager, and Tags. A task list on the right side of the screen provides instructions for the next steps, including clicking 'Review + Create', waiting for deployment completion, and navigating to the 'Resource groups' blade to find the new resource group 'MC_AZ500LAB09_MyKubernetesCluster_eastus'.

The screenshot shows the Azure Portal interface for the 'Resource groups' page. The left sidebar contains navigation options like Home, Create, Manage view, Refresh, Export to CSV, Open query, Assign tags, and Group by none. The main content area displays a table of resource groups, including 'AZ500LAB09', 'cloud-shell-storage-west-europe', 'MC_AZ500LAB09_MyKubernetesCluster_eastus', 'NetworkWatcherRG', and 'ResourceGroup1'. A task list on the right side of the screen provides instructions for the next steps, including navigating to the 'Resource groups' blade to find the new resource group 'MC_AZ500LAB09_MyKubernetesCluster_eastus' and clicking the 'AZ500LAB09' entry.

Name	Subscription	Location
AZ500LAB09	MOC Subscription-lod50651563	East US
cloud-shell-storage-west-europe	MOC Subscription-lod50651563	West Europe
MC_AZ500LAB09_MyKubernetesCluster_eastus	MOC Subscription-lod50651563	East US
NetworkWatcherRG	MOC Subscription-lod50651563	East US
ResourceGroup1	MOC Subscription-lod50651563	East US

sample/nginxv1 - Microsoft Azure

AZ500LAB09 - Microsoft Azure

+

https://portal.azure.com/#@L0D5PR0DMCA.onmicrosoft.com/resource/subscriptions/90...

Microsoft Azure

Search resources, services, and docs (G+J)

Copilot

Home > Resource groups >

AZ500LAB09

Resource group

Search

+ Create

Manage view

Delete resource group

Refresh

Export to CSV

...

Overview

Activity log

Access control (IAM)

Tags

Resource visualizer

Add or remove favorites by pressing Ctrl+Shift+F

Showing 1 to 2 of 2 records.

Show hidden types

No grouping

List view

Name	Type	Location
az500506321738	Container registry	East US
MyKubernetesCluster	Kubernetes service	East US

Switch to PowerShell

Restart

Manage files

New session

Editor

Web preview

Settings

Help

Requesting a Cloud Shell.Succeeded.

Connecting terminal...

Storage fileshare subscription 9030ddef-445a-4a0a-9d99-b1ba7b5b0748 is not registered to Microsoft.CloudShell Namespace. Please follow these instructions "https://aka.ms/RegisterCloudShell" to register. In future, unregistered subscriptions will have restricted access to CloudShell service.

labuser-51743740 [~]\$ az aks get-credentials --resource-group AZ500LAB09 --name MyKubernetesCluster

Merged "MyKubernetesCluster" as current context in /home/labuser-51743740/.kube/config

labuser-51743740 [~]\$ kubectl get nodes

NAME	STATUS	ROLES	AGE	VERSION
aks-agentpool-30198516-0	Ready	<none>	32m	v1.31.8

labuser-51743740 [~]\$

ENG US

12:03 PM

5/28/2025

Configuring and Securing AKS and ACR

24 Minutes Remaining

Instructions

Resources

Help

100%

14. In the Bash session within the Cloud Shell pane, run the following to list nodes of the Kubernetes cluster:

sh

kubectl get nodes

Verify that the **Status** of the cluster node is listed as **Ready**.

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

Reminder: Before running **Task 4 step 2** you will need to manually create a virtual network. In the **Search Resources, Services, and Docs** text box at the top of the Azure portal page, type **Virtual networks** and press the **Enter** key. Click the **Create** button, and once that opens use the **AZ500LAB09 Resource Group** and name it **AZ500LAB09-vnet**. Once this is done you can use the script in **Task 4 step 2**.

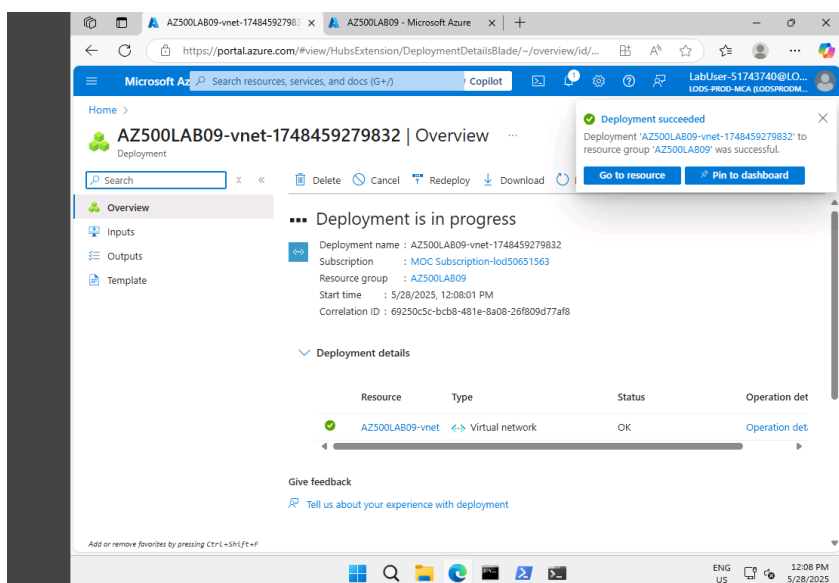
51% Tasks Complete

Previous

End

Task 4: Grant the AKS cluster permissions to access the ACR

To allow the AKS cluster to pull container images from the ACR, I attached the ACR to the AKS cluster using the `az aks update --attach-acr` command. This granted the necessary `acrpull` role. Additionally, I retrieved the virtual network ID associated with the AKS infrastructure and the cluster's managed identity. I then assigned the "Contributor" role to the managed identity over the virtual network using the `az role assignment create` command. This ensures the AKS cluster has permissions to interact with network resources, which is critical for real-world deployments.

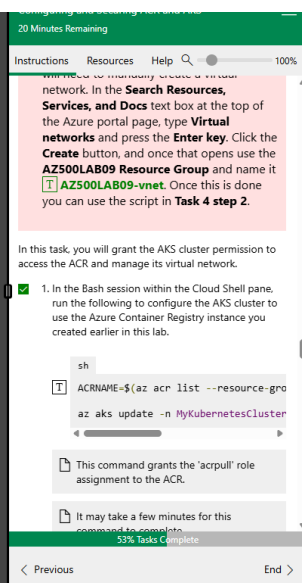


Deployment is in progress

Deployment name : AZ500LAB09-vnet-1748459279832
Subscription : MOC Subscription-lod50651563
Resource group : AZ500LAB09
Start time : 5/28/2025, 12:08:01 PM
Correlation ID : 69250c5c-bcb8-481e-8a08-26f809d77af8

Resource	Type	Status	Operation details
AZ500LAB09-vnet	Virtual network	OK	Operation details

Give feedback
Tell us about your experience with deployment



Instructions Resources Help 100%

network. In the **Search Resources, Services, and Docs** text box at the top of the Azure portal page, type **Virtual networks** and press the **Enter** key. Click the **Create** button, and once that opens use the **AZ500LAB09 Resource Group** and name it **AZ500LAB09-vnet**. Once this is done you can use the script in **Task 4 step 2**.

In this task, you will grant the AKS cluster permission to access the ACR and manage its virtual network.

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

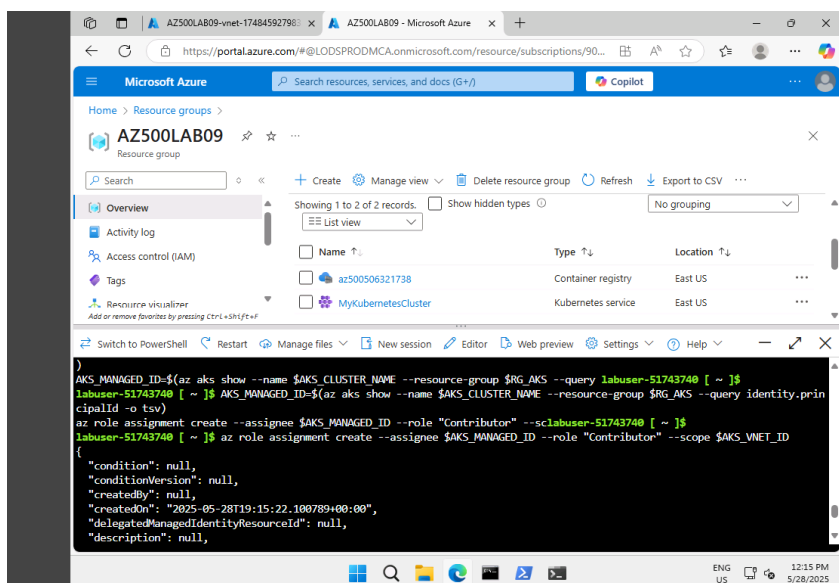
```
sh
ACRNAME=$(az acr list --resource-group $ACR_RESOURCE_GROUP --query [].name --output tsv)
az aks update -n MyKubernetesCluster --attach-acr $ACRNAME
```

This command grants the 'acrpull' role assignment to the ACR.

It may take a few minutes for this command to complete.

33% Tasks Complete

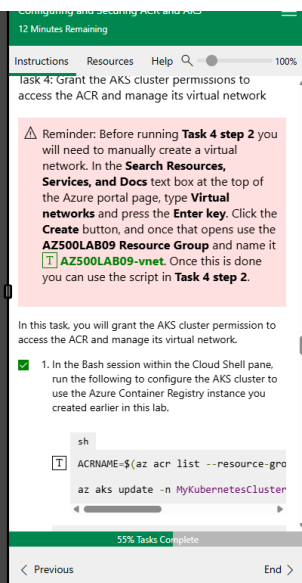
Previous End



Showing 1 to 2 of 2 records. Show hidden types No grouping

Name	Type	Location
az500506321738	Container registry	East US
MyKubernetesCluster	Kubernetes service	East US

```
AKS_MANAGED_ID=$(az aks show --name $AKS_CLUSTER_NAME --resource-group $RG_AKS --query labuser-51743740 [ ~ ]$ labuser-51743740 [ ~ ]$ 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 "Contributor" --scope $AKS_VNET_ID
az role assignment create --assignee $AKS_MANAGED_ID --role "Contributor" --scope $AKS_VNET_ID
```



Instructions Resources Help 100%

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

Reminder: Before running **Task 4 step 2** you will need to manually create a virtual network. In the **Search Resources, Services, and Docs** text box at the top of the Azure portal page, type **Virtual networks** and press the **Enter** key. Click the **Create** button, and once that opens use the **AZ500LAB09 Resource Group** and name it **AZ500LAB09-vnet**. Once this is done you can use the script in **Task 4 step 2**.

In this task, you will grant the AKS cluster permission to access the ACR and manage its virtual network.

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

```
sh
ACRNAME=$(az acr list --resource-group $ACR_RESOURCE_GROUP --query [].name --output tsv)
az aks update -n MyKubernetesCluster --attach-acr $ACRNAME
```

55% Tasks Complete

Previous End

Task 5: Deploy an external service to AKS

In this task, I uploaded two YAML files—`nginxexternal.yaml` and `nginxinternal.yaml`—to the Cloud Shell environment. These manifest files define Kubernetes deployments. I opened the `nginxexternal.yaml` file using the integrated code editor and replaced a placeholder with the actual name of the ACR. This step was important to ensure the deployment could reference the correct container image. After saving the file, I used `kubectl apply -f nginxexternal.yaml` to deploy the external Nginx service to the AKS cluster, enabling external access to the hosted web server.

The screenshot shows the Microsoft Azure portal with the resource group **AZ500LAB09** selected. The **Cloud Shell** terminal is open, displaying the contents of the `nginxexternal.yaml` file. The file defines a deployment for the `nginx` service, using the image `az500506321738.azurecr.io/sample/nginx:v1`. The terminal output shows the command `kubectl apply -f nginxexternal.yaml` being executed successfully, resulting in the deployment `deployment.apps/nginxexternal` being created.

On the right side, the **Instructions** pane for Task 5 is visible, showing steps 4 through 7. Step 4 instructs to replace the `<ACRUniqueName>` placeholder with the ACR name. Step 5 instructs to click the **Save** icon. Step 6 instructs to run `kubectl apply -f nginxexternal.yaml` in the Bash session. Step 7 instructs to review the output of the command.

The screenshot shows the Microsoft Azure portal with the resource group **AZ500LAB09** selected. The **Cloud Shell** terminal is open, displaying the output of the `kubectl apply -f nginxexternal.yaml` command. The output shows the deployment `deployment.apps/nginxexternal` being created successfully. The terminal also shows a message indicating that the storage fileshare subscription is not registered to the Microsoft CloudShell namespace.

On the right side, the **Instructions** pane for Task 6 is visible, showing steps 5 through 7. Step 5 instructs to click the **Save** icon. Step 6 instructs to run `kubectl apply -f nginxexternal.yaml` in the Bash session. Step 7 instructs to review the output of the command.

Task 6: Verify you can access an external AKS-hosted service

After deploying the external Nginx service, I verified its availability. I did this by running `kubectl get services` to obtain the external IP address assigned to the Nginx service. Once the external IP was available, I entered it into a web browser and confirmed that the Nginx welcome page loaded successfully. This validated that the service was not only deployed correctly but also accessible from the internet, proving the AKS cluster's networking and image pulling functionalities were correctly configured.

The screenshot shows the Microsoft Azure portal interface. The main content area displays the 'Overview' tab for the resource group 'AZ500LAB09'. It lists two resources: 'az500506321738' (Container registry) and 'MyKubernetesCluster' (Kubernetes service). Below this, a terminal window is open, showing the following commands and output:

```
labuser-51743740 [ ~ ]$ kubectl apply -f nginxexternal.yaml
deployment.apps/nginxexternal created
service/nginxexternal created
labuser-51743740 [ ~ ]$ kubectl get service nginxexternal
NAME          TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
nginxexternal LoadBalancer 10.0.12.39    52.226.177.26  80:32653/TCP    67s
labuser-51743740 [ ~ ]$
```

On the right side, a '41 Minutes Remaining' timer is visible, along with a list of instructions for the task. The instructions include:

1. In the Bash session within the Cloud Shell pane, run the following to retrieve information about the nginxexternal service including name, type, IP addresses, and ports.
2. In the Bash session within the Cloud Shell pane, review the output and record the value in the External-IP column. You will need it in the next step.
3. Open a new browser tab and browse to the IP address you identified in the previous step.
4. Ensure the **Welcome to nginx!** page displays.

Below the instructions, the task is labeled 'Task 7: Deploy an internal service to AKS'. The progress bar indicates '71% Tasks Complete'.

The screenshot shows a web browser window displaying the 'Welcome to nginx!' page. The page content includes:

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.

On the right side, a '40 Minutes Remaining' timer is visible, along with a list of instructions for the task. The instructions include:

1. In the Bash session within the Cloud Shell pane, run the following to open the nginxinternal.yaml file, so you can edit its content.
2. In the Bash session within the Cloud Shell pane, review the output and record the value in the External-IP column. You will need it in the next step.
3. Open a new browser tab and browse to the IP address you identified in the previous step.
4. Ensure the **Welcome to nginx!** page displays.

Below the instructions, the task is labeled 'Task 7: Deploy an internal service to AKS'. The progress bar indicates '73% Tasks Complete'.

Task 7: Deploy an Internal Service to AKS

I then proceeded to deploy the internal version of the Nginx service using the `nginxinternal.yaml` file. Just like the external deployment, I ensured that the ACR name was correctly referenced in the YAML configuration. I applied the configuration using the `kubectl apply -f nginxinternal.yaml` command. This internal deployment exposed the service only within the cluster, rather than making it accessible from the internet. This setup is commonly used for backend services or components that shouldn't be publicly reachable.

The screenshot shows the Microsoft Azure portal interface for resource group **AZ500LAB09**. The 'Overview' tab is selected, displaying a table with 2 records:

Name	Type	Location
az500506321738	Container registry	East US

Below the table, a terminal window shows the contents of the `nginxinternal.yaml` file:

```
17 labels:
18   app: nginxinternal
19 spec:
20   nodeSelector:
21     "kubernetes.io/os": linux
22   containers:
23   - name: nginx
24     image: az500506321738.azurecr.io/sample/nginx:v1
25     ports:
26     - containerPort: 80
```

The terminal also shows the command `code ./nginxinternal.yaml` being executed.

On the right, a task instruction panel for 'Task 7: Deploy an internal service to AKS' is visible. It includes steps 1 through 4, with step 1 completed. The panel shows a code editor with the command `kubectl apply -f nginxinternal.yaml` and a progress bar indicating 75% tasks complete.

The screenshot shows the Microsoft Azure portal interface for resource group **AZ500LAB09**. The 'Overview' tab is selected, displaying a table with 2 records:

Name	Type	Location
az500506321738	Container registry	East US
MyKubernetesCluster	Kubernetes service	East US

Below the table, a terminal window shows the output of the `kubectl apply -f nginxinternal.yaml` command:

```
labuser-51743740 [ ~ ]$ kubectl apply -f nginxinternal.yaml
deployment.apps/nginxinternal created
service/nginxinternal created
labuser-51743740 [ ~ ]$ kubectl get service nginxinternal
NAME      TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
nginxinternal LoadBalancer 10.0.184.110   10.224.0.33    80:30463/TCP 18s
```

On the right, a task instruction panel for 'Task 7: Deploy an internal service to AKS' is visible. It includes steps 5 through 7, with step 5 completed. The panel shows a code editor with the command `kubectl get service nginxinternal` and a progress bar indicating 85% tasks complete.

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

To validate the internal service, I first identified the pod name by running `kubectl get pods`. I then used `kubectl exec -it <pod-name> -- /bin/bash` to connect to the pod's shell. From inside the pod, I used `curl` to make a request to the internal service's ClusterIP and port. Receiving the expected HTML response confirmed that the internal service was functioning properly and was reachable from within the cluster. This completed the full cycle of deploying and verifying both public and private services in AKS using container images stored in ACR.

The screenshot shows the Microsoft Azure portal interface. The main pane displays the 'AZ500LAB09' resource group. Below the overview, a terminal window is open, showing the following commands and output:

```
labuser-S1743740 [ ~ ]$ kubectl apply -f nginxinternal.yaml
deployment.apps/nginxinternal created
service/nginxinternal created
labuser-S1743740 [ ~ ]$ kubectl get service nginxinternal
NAME      TYPE      CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
nginxinternal  LoadBalancer  10.0.184.110  10.224.0.33    80:30469/TCP  18s
labuser-S1743740 [ ~ ]$ kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
nginxexternal-6476d44f55-f9v7w  1/1     Running   0           19m
nginxinternal-6694d4dc85-xc5n7  1/1     Running   0           6m23s
labuser-S1743740 [ ~ ]$ kubectl exec -it nginxexternal-6476d44f55-f9v7w -- /bin/bash
root@nginxexternal-6476d44f55-f9v7w:/# curl http://10.224.0.33
<html>
```

The right-hand pane shows a 'Configuring and securing ACR and AKS' task list. The steps are:

3. In the Bash session within the Cloud Shell pane, run the following to connect interactively to the first pod (replace the `<pod_name>` placeholder with the name you copied in the previous step):
`sh`
`kubectl exec -it <pod_name> -- /bin/`
4. In the Bash session within the Cloud Shell pane, run the following to verify that the nginx web site is available via the private IP address of the service (replace the `<internal_IP>` placeholder with the IP address you recorded in the previous task):
`sh`
`curl http://<internal_IP>`
5. Close the Cloud Shell pane.

The result of the task is: 'You have configured and secured ACR and AKS.' The 'Clean up resources' section reminds the user to remove any newly created Azure resources that are no longer used. The task progress is 93% complete.

The screenshot shows the Microsoft Azure portal interface. The main pane displays the 'AZ500LAB09' resource group. Below the overview, a terminal window is open, showing the following commands and output:

```
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>.
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p>Thank you for using nginx.</p>
</body>
</html>
root@nginxexternal-6476d44f55-f9v7w:/#
```

The right-hand pane shows a 'Configuring and securing ACR and AKS' task list. The steps are:

3. In the Bash session within the Cloud Shell pane, run the following to connect interactively to the first pod (replace the `<pod_name>` placeholder with the name you copied in the previous step):
`sh`
`kubectl exec -it <pod_name> -- /bin/`
4. In the Bash session within the Cloud Shell pane, run the following to verify that the nginx web site is available via the private IP address of the service (replace the `<internal_IP>` placeholder with the IP address you recorded in the previous task):
`sh`
`curl http://<internal_IP>`
5. Close the Cloud Shell pane.

The result of the task is: 'You have configured and secured ACR and AKS.' The 'Clean up resources' section reminds the user to remove any newly created Azure resources that are no longer used. The task progress is 93% complete.

Conclusion

Completing this lab provided valuable practical experience with container orchestration and secure DevOps practices in Azure. By successfully creating and linking an Azure Container Registry to an AKS cluster, I learned how to control image access and enable secure, automated deployments. Additionally, deploying both internal and external services within the Kubernetes environment enhanced my understanding of cluster networking and service exposure in AKS. These tasks are foundational for implementing scalable, secure, and resilient container-based solutions in the cloud. The lab not only reinforced key Azure CLI skills but also emphasized best practices in resource management and security in a production-like environment.