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

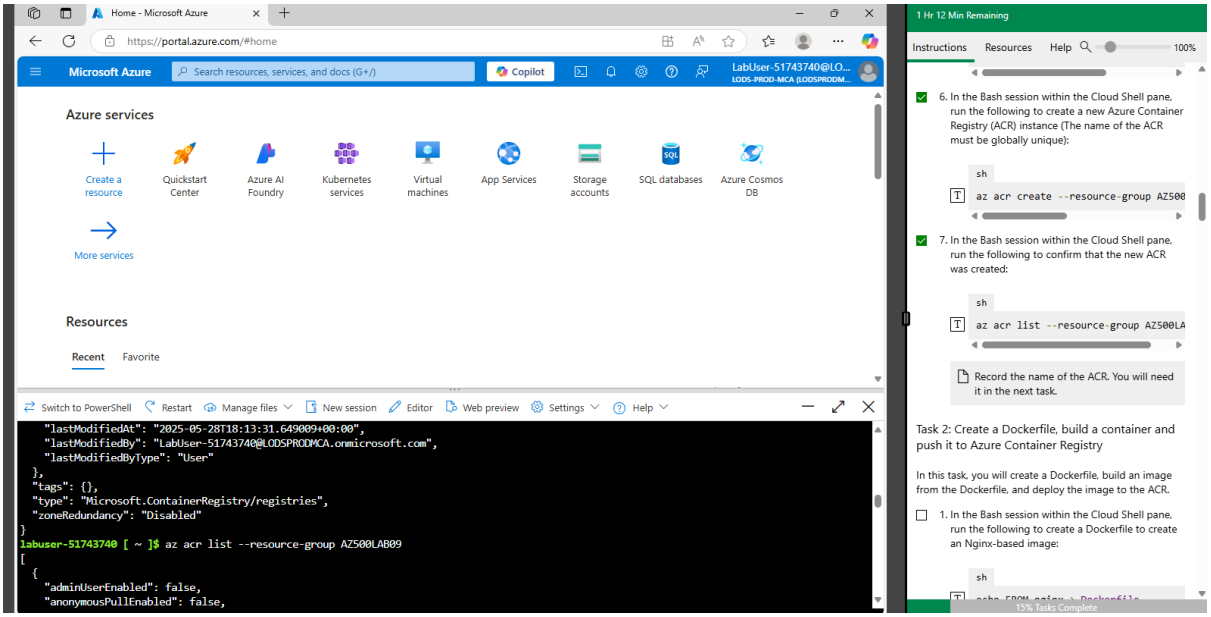
Task 7: Deploy an internal service to AKS

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

Tasks:

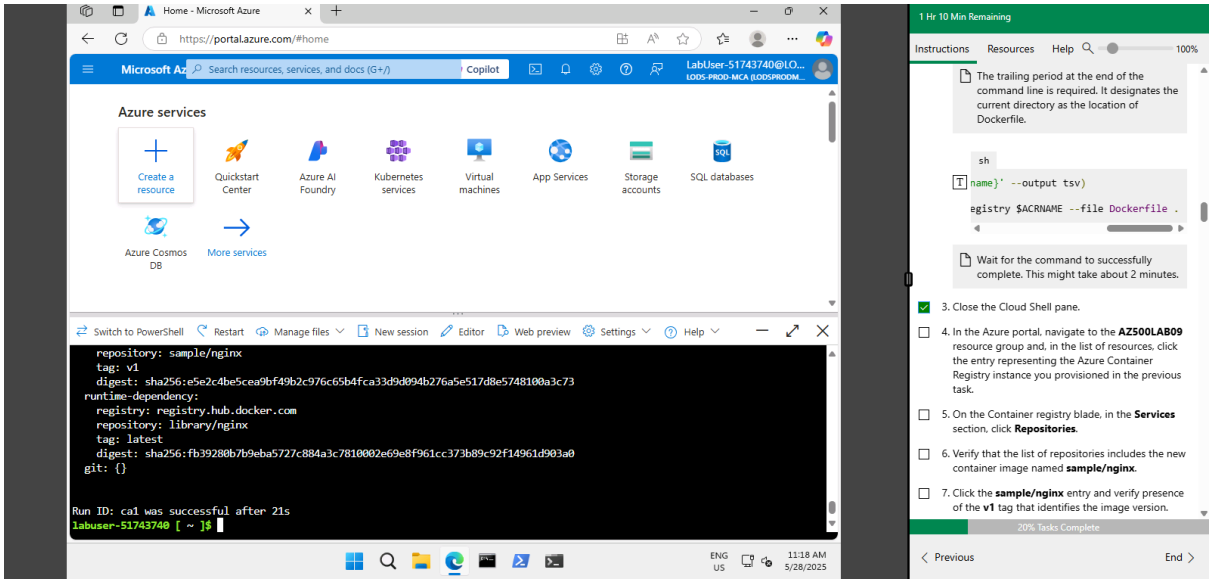
Task 1: Create an Azure Container Registry

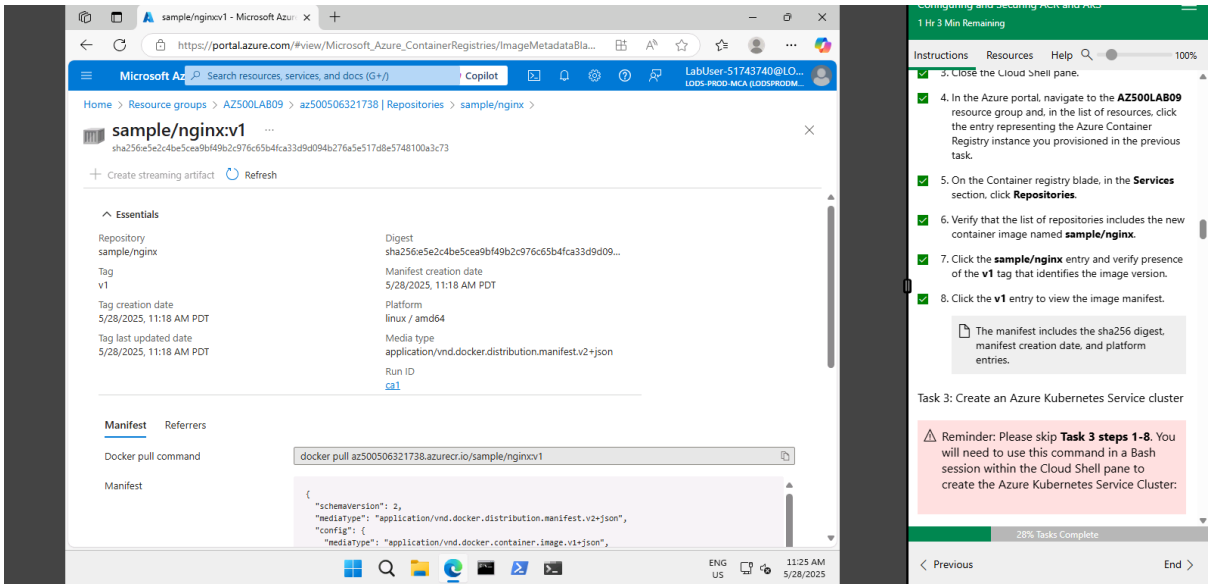
The lab began by signing into the Azure Portal and launching the Cloud Shell with Bash selected. I then created a new resource group named AZ500LAB09 in the East US region using the Azure CLI. After confirming the group was successfully created, I provisioned a new Azure Container Registry (ACR) with 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.



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

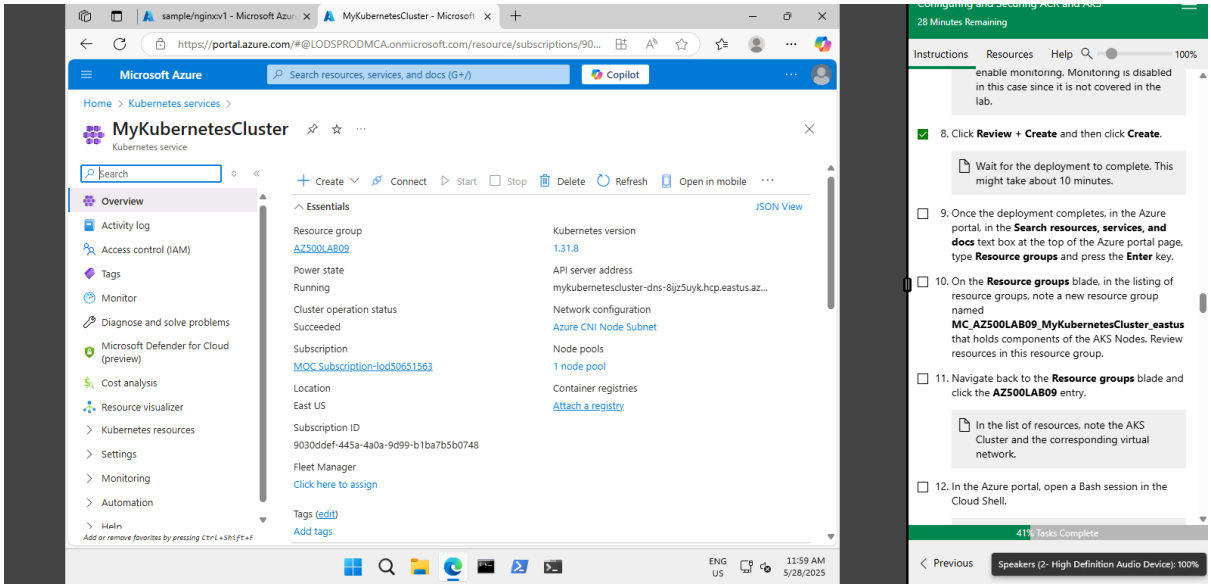
Next, I created a Dockerfile directly within the Cloud Shell environment. The Dockerfile was a basic configuration using the official Nginx image as its base. I then built a Docker container image from this file and pushed it to the ACR created earlier using the az acr build command. This command automatically 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.

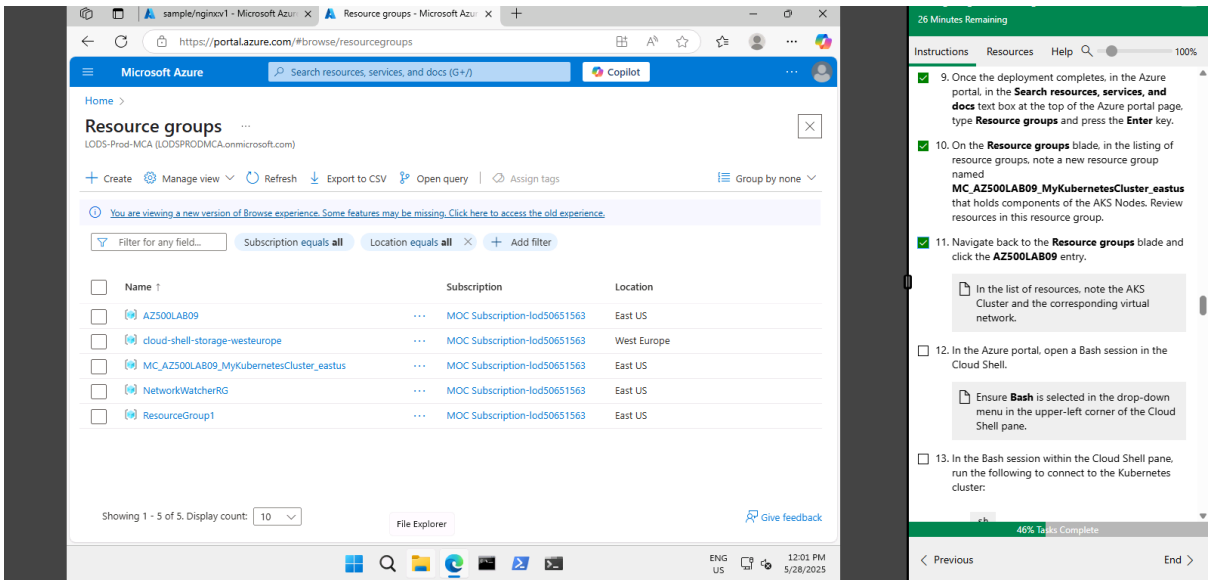


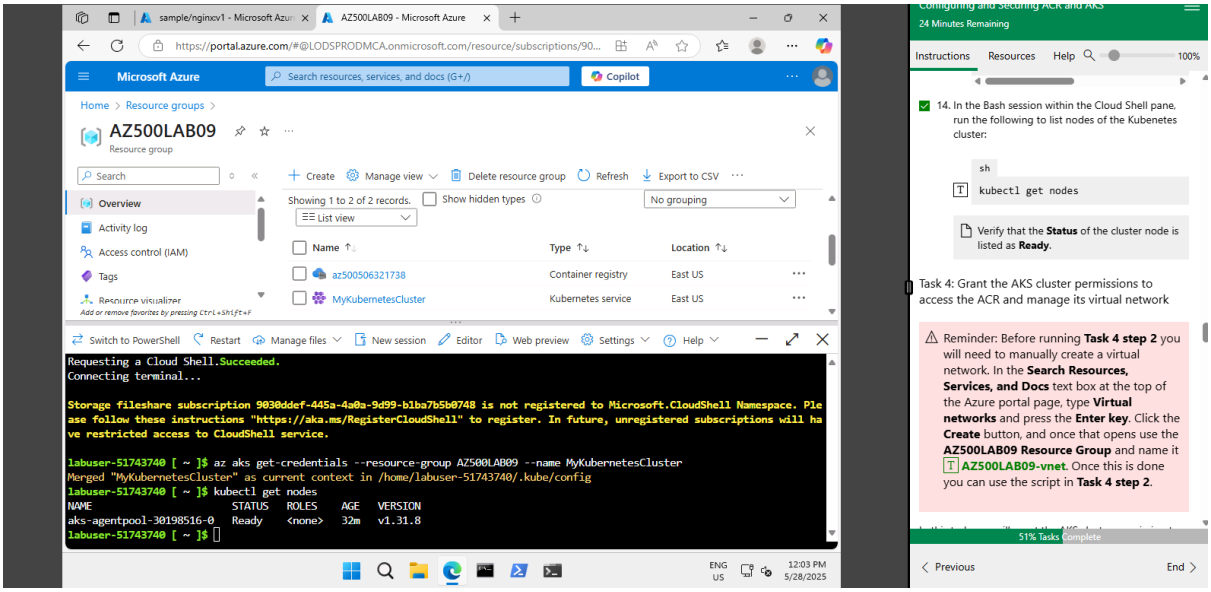


Task 3: Create an Azure Kubernetes Service cluster

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.

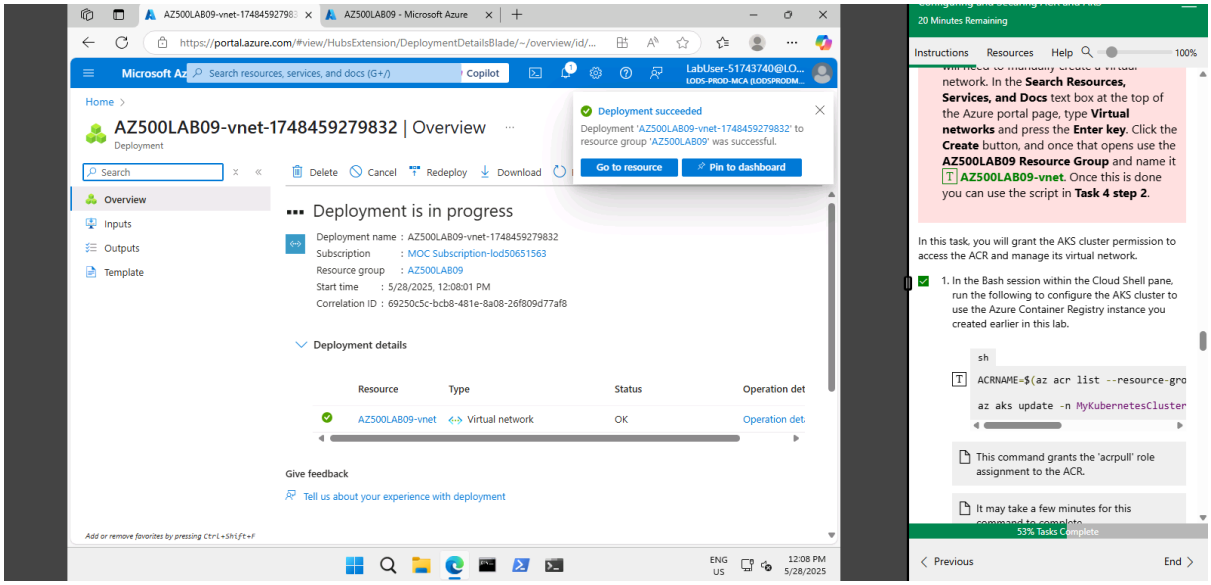


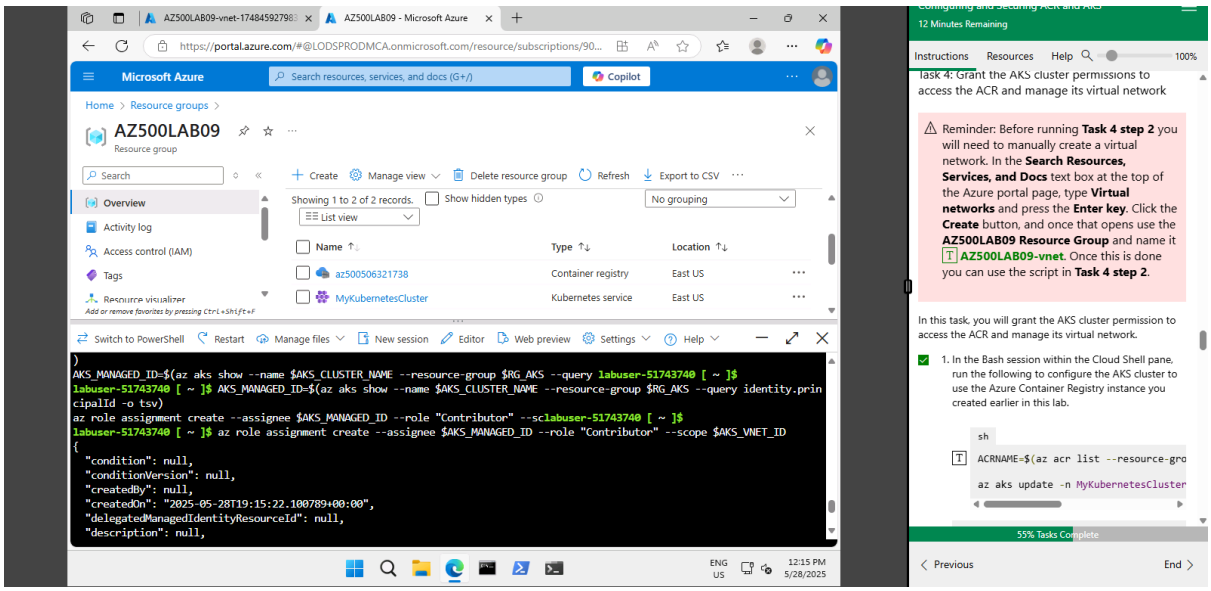



sample/nginxv1 - Microsoft Azur
AZ500LAB09 - Microsoft Azure
https://portal.azure.com/#@LODSPRODMCA.onmicrosoft.com/resource/subscriptions/90...
Microsoft Azure
Search resources, services, and docs (G+/)
Copilot
Home > Resource groups >
AZ500LAB09
Resource group
Search
Overview
Activity log
Access control (IAM)
Tags
Resource visualizer
Add or remove favorites by pressing Ctrl+Shift+F
Create
Manage view
Delete resource group
Refresh
Export to CSV
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
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 Kubenetes 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

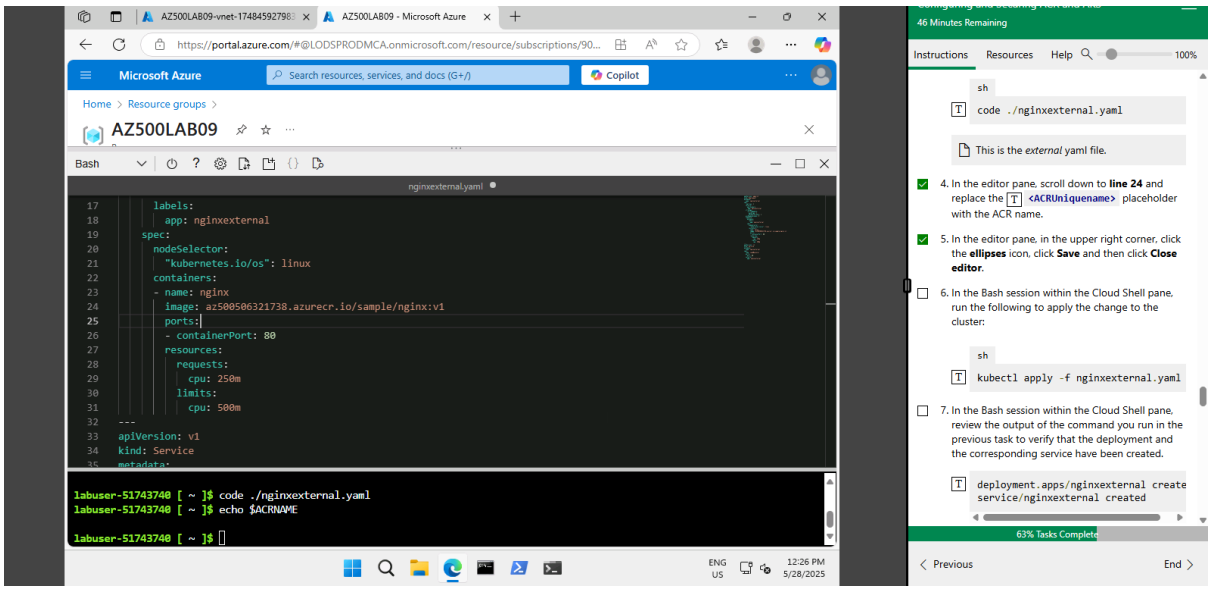
To allow the AKS cluster to pull container images from the ACR, I attached the ACR to the AKS cluster using the az aks update --attach-acr command. This granted the necessary acrpull role. Additionally, I retrieved the virtual network ID associated with the AKS infrastructure and the cluster's managed 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.

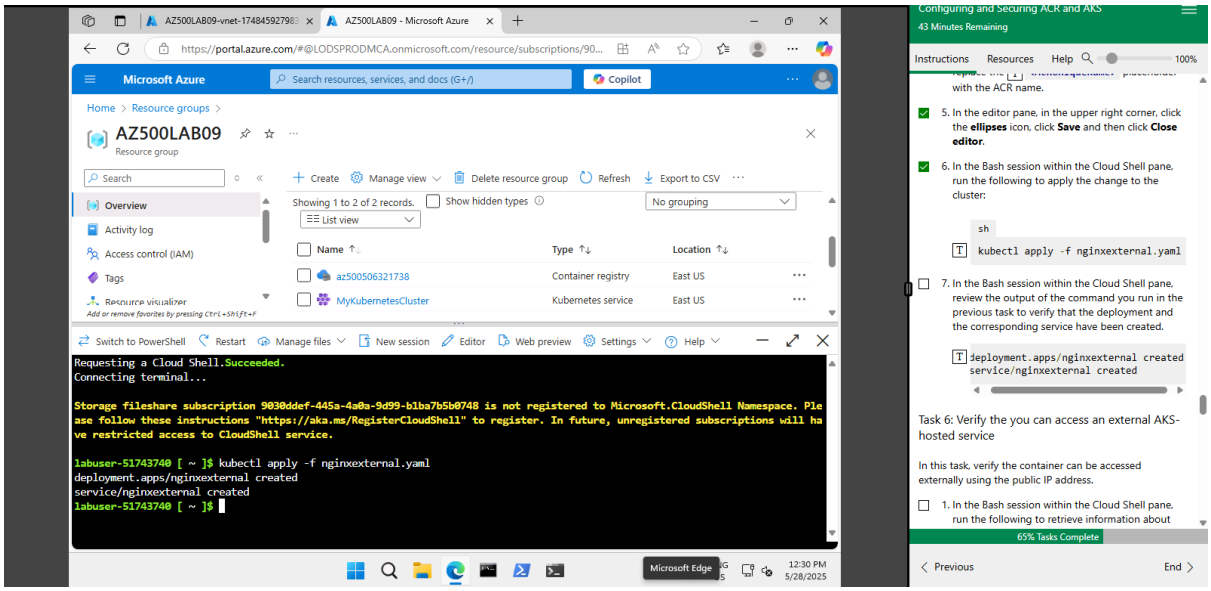




Task 5: Deploy an external service to AKS

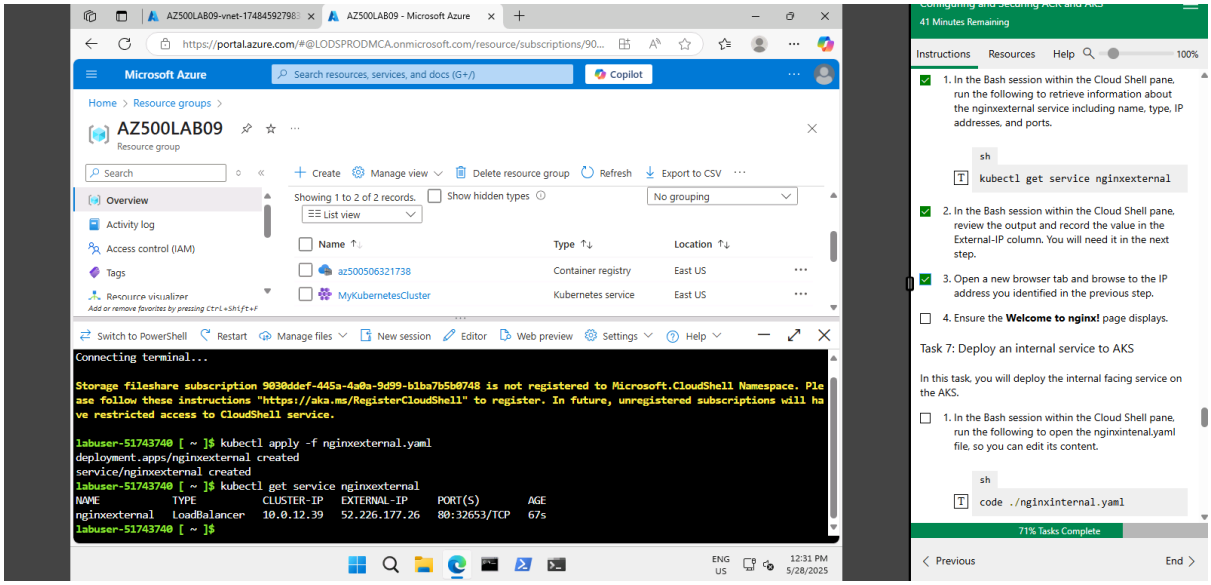
In this task, I uploaded two YAML files—nginxexternal.yaml and nginxinternal.yaml—to the Cloud Shell environment. These manifest files define Kubernetes deployments. I opened the nginxexternal.yaml file using the integrated code editor and replaced a placeholder with the actual name of the ACR. This 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.

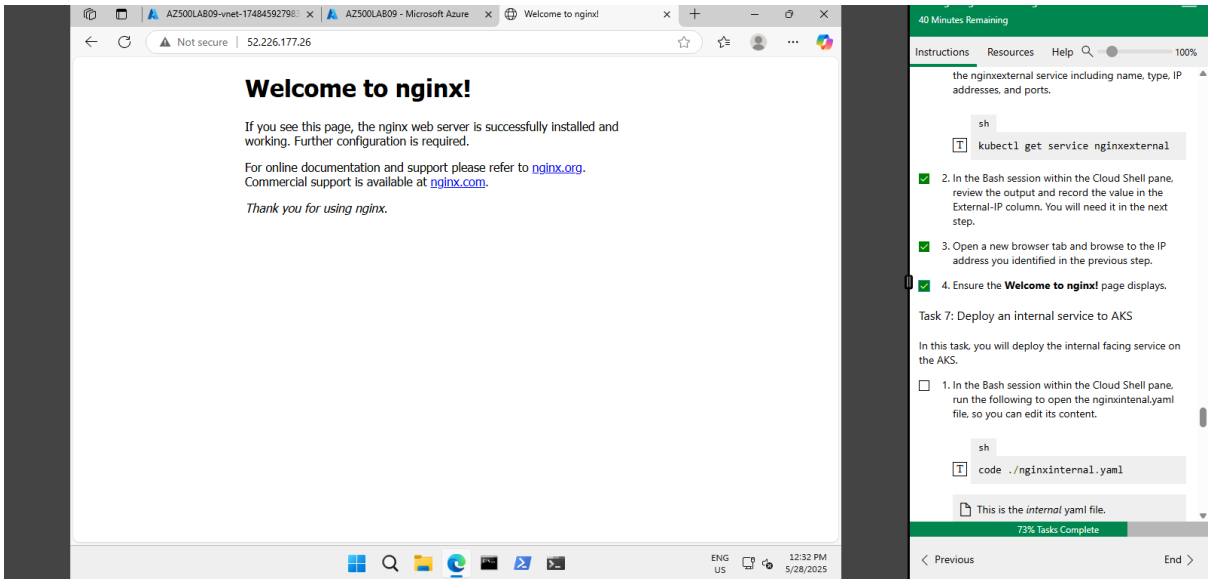




Task 6: Verify you can access an external AKS-hosted service

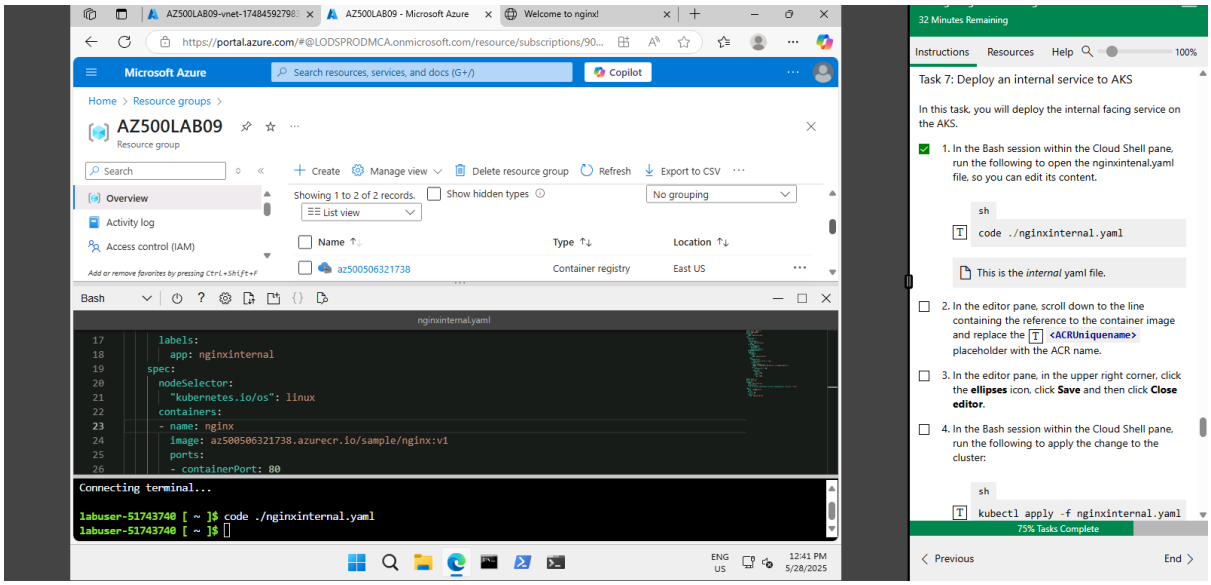
After deploying the external Nginx service, I verified its availability. I did this by running kubectl get services to obtain the external IP address assigned to the Nginx service. Once the external IP was available, I entered it into a web browser and confirmed that the Nginx welcome page loaded successfully. 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.

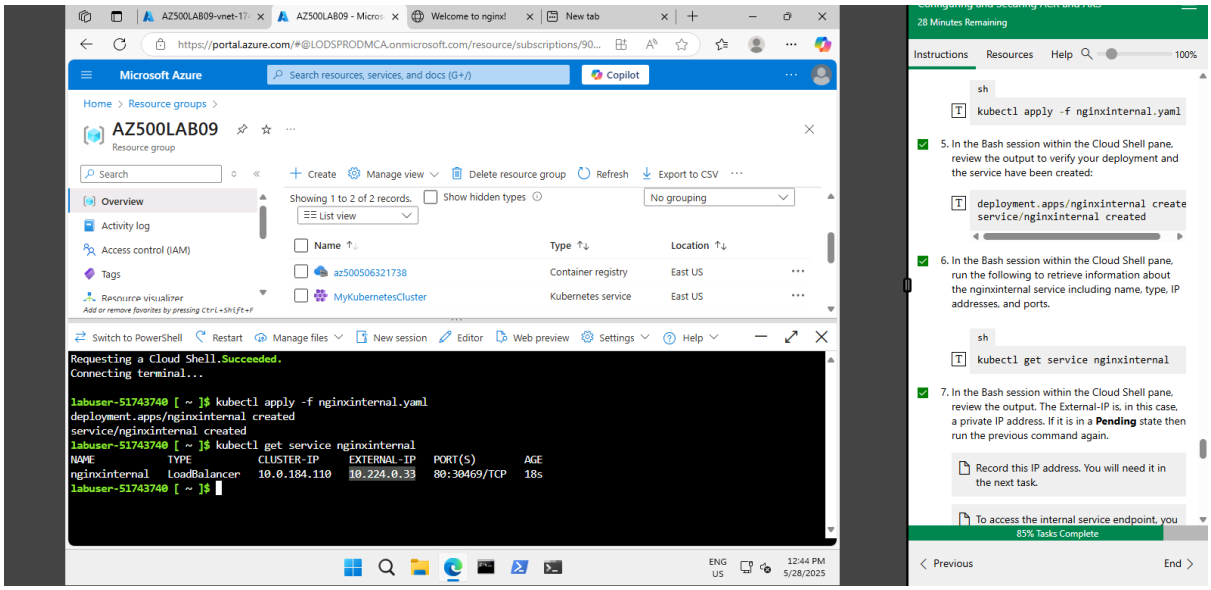




Task 7: Deploy an Internal Service to AKS

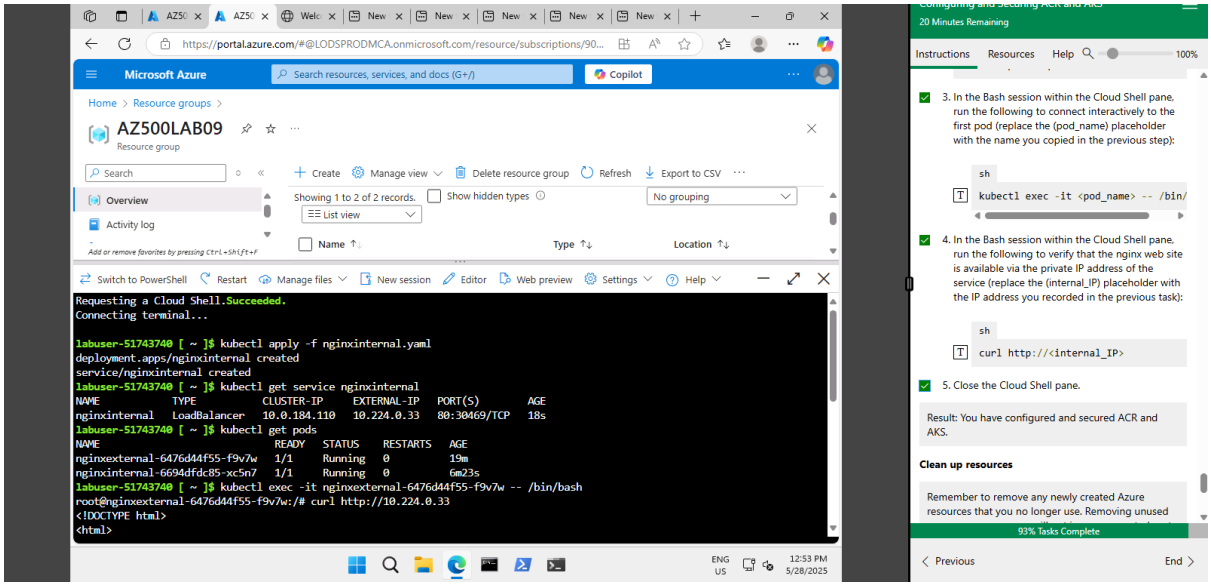
I then proceeded to deploy the internal version of the Nginx service using the nginxinternal.yaml file. Just like the external deployment, I ensured that the ACR name was correctly referenced in the YAML configuration. I applied the configuration using the kubectl apply -f nginxinternal.yaml command. 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.

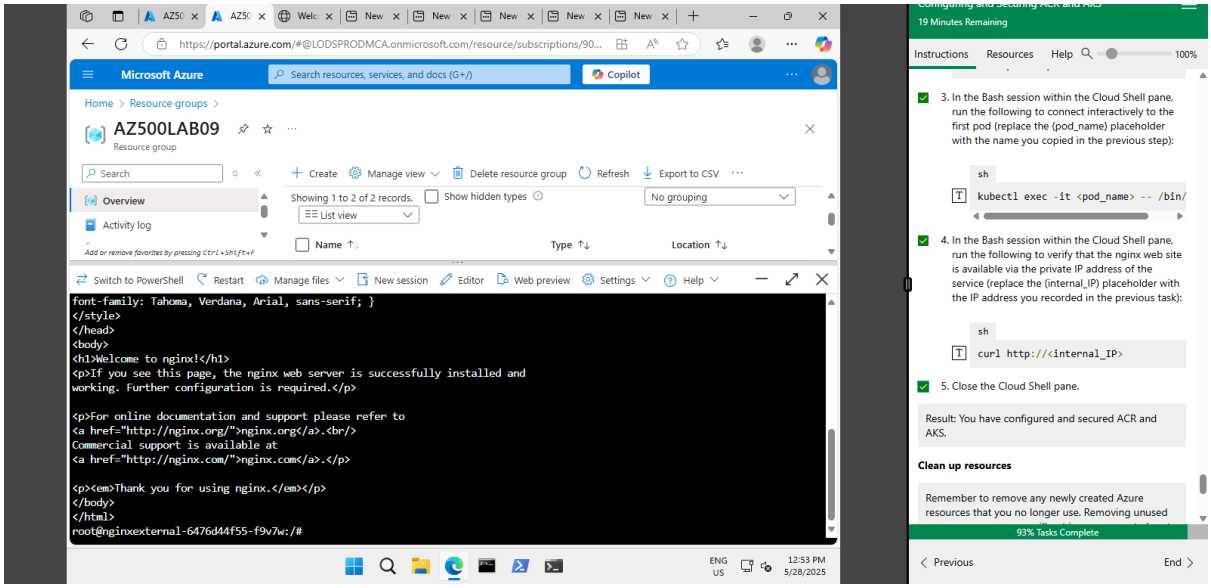




Task 8: Verify the you can access an internal AKS-hosted service

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.





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.