# Lab 07: Configuring and Securing ACR and AKS

# Student lab manual

## Lab scenario

You have been asked to deploy a proof of concept with Azure Container Registry and Azure Kubernetes Service. Specifically, the proof of concept should demonstrate:

* Using Dockerfile to build an image.
* Using Azure Container Registry to store images.
* Configuring an Azure Kubernetes Service.
* Securing and accessing container applications both internally and externally.

For all the resources in this lab, we are using the **East US** region. Verify with your instructor this is the region to use for class.

## Lab objectives

In this lab, you will complete the following exercise:

* Exercise 1: Configuring and Securing ACR and AKS

## Configuring and Securing ACR and AKS diagram

image

image

## Intructions

## Lab files:

* **\Allfiles\Labs\09\nginxexternal.yaml**
* **\Allfiles\Labs\09\nginxinternal.yaml**

### Exercise 1: Configuring and Securing ACR and AKS

### Estimated timing: 45 minutes

For all the resources in this lab, we are using the **East (US)** region. Verify with your instructor this is region to use for you class.

In this exercise, you will complete 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 the 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

#### Task 1: Create an Azure Container Registry

In this task, you will create a resource group for the lab an an Azure Container Registry.

1. Sign-in to the Azure portal **https://portal.azure.com/**.

* **Note**: Sign in to the Azure portal using an account that has the Owner or Contributor role in the Azure subscription you are using for this lab and the Global Administrator role in the Microsoft Entra tenant associated with that subscription.

1. In the Azure portal, open the Cloud Shell by clicking the first icon in the top right of the Azure Portal. If prompted, click **Bash** and **Create storage**.
2. Ensure **Bash** is selected in the drop-down menu in the upper-left corner of the Cloud Shell pane.
3. In the Bash session within the Cloud Shell pane, run the following to create a new resource group for this lab:

* az group create --name AZ500LAB09 --location eastus

1. In the Bash session within the Cloud Shell pane, run the following to verify the resource group was created:

* az group list --query "[?name=='AZ500LAB09']" -o table

1. 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 AZ500LAB09 --name az500$RANDOM$RANDOM --sku Basic

1. 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 AZ500LAB09
* **Note**: 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:

* echo FROM nginx > Dockerfile

1. In the Bash session within the Cloud Shell pane, run the following to build an image from the Dockerfile and push the image to the new ACR.

* **Note**: The trailing period at the end of the command line is required. It designates the current directory as the location of Dockerfile.
* ACRNAME=$(az acr list --resource-group AZ500LAB09 --query '[].{Name:name}' --output tsv)  
    
  az acr build --resource-group AZ500LAB09 --image sample/nginx:v1 --registry $ACRNAME --file Dockerfile .
* **Note**: Wait for the command to successfully complete. This might take about 2 minutes.

1. Close the Cloud Shell pane.
2. In the Azure portal, navigate to the **AZ500Lab09** resource group and, in the list of resources, click the entry representing the Azure Container Registry instance you provisioned in the previous task.
3. On the Container registry blade, in the **Services** section, click **Repositories**.
4. Verify that the list of repositories includes the new container image named **sample/nginx**.
5. Click the **sample/nginx** entry and verify presence of the **v1** tag that identifies the image version.
6. Click the **v1** entry to view the image manifest.

* **Note**: The manifest includes the sha256 digest, manifest creation date, and platform entries.

#### Task 3: Create an Azure Kubernetes Service cluster

In this task, you will create an Azure Kubernetes service and review the deployed resources.

1. In the Azure portal, in the **Search resources, services, and docs** text box at the top of the Azure portal page, type **Kubernetes services** and press the **Enter** key.
2. On the **Kubernetes services** blade, click **+ Create** and, in the drop-down menu, click **+ Create a Kubernetes cluster**
3. On the **Basics** tab of the **Create Kubernetes cluster** blade, select **Cluster preset configuration**, select **Dev/Test ($)**. Now specify the following settings (leave others with their default values):

| * Setting | * Value |
| --- | --- |
| * Subscription | * the name of the Azure subscription you are using in this lab |
| * Resource group | * **AZ500LAB09** |
| * Kubernetes cluster name | * **MyKubernetesCluster** |
| * Region | * **(US) East US** |
| * Availability zones | * **None** |
| * Scale method | * **Manual** |
| * Node count | * **1** |

1. Click **Next: Node Pools >** and, on the **Node Pools** tab of the **Create Kubernetes cluster** blade, specify the following settings (leave others with their default values):

| * Setting | * Value |
| --- | --- |
| * Enable virtual nodes | * cleared checkbox |

1. Click **Next: Access >**, on the **Access** tab of the **Create Kubernetes cluster** blade, accept the defaults, and click **Next: Networking >**.
2. On the **Networking** tab of the **Create Kubernetes cluster** blade, specify the following settings (leave others with their default values):

| * Setting | * Value |
| --- | --- |
| * Network configuration | * **Azure CNI** |
| * DNS name prefix | * **Leave the default value** |

* **Note**: AKS can be configured as a private cluster. This assigns a private IP to the API server to ensure network traffic between your API server and your node pools remains on the private network only. For more information, visit [Create a private Azure Kubernetes Service cluster](https://docs.microsoft.com/en-us/azure/aks/private-clusters) page.

1. Click **Next: Integrations >** and, on the **Integrations** tab of the **Create Kubernetes cluster** blade, set **Container monitoring** to **Disabled**.

* **Note**: In production scenarios, you would want to enable monitoring. Monitoring is disabled in this case since it is not covered in the lab.

1. Click **Review + Create** and then click **Create**.

* **Note**: Wait for the deployment to complete. This might take about 10 minutes.

1. Once the deployment completes, in the Azure portal, in the **Search resources, services, and docs** text box at the top of the Azure portal page, type **Resource groups** and press the **Enter** key.
2. On the **Resource groups** blade, in the listing of resource groups, note a new resource group named **MC\_AZ500LAB09\_MyKubernetesCluster\_eastus** that holds components of the AKS Nodes. Review resources in this resource group.
3. Navigate back to the **Resource groups** blade and click the **AZ500LAB09** entry.

* **Note**: In the list of resources, note the AKS Cluster and the corresponding virtual network.

1. In the Azure portal, open a Bash session in the Cloud Shell.

* **Note**: Ensure **Bash** is selected in the drop-down menu in the upper-left corner of the Cloud Shell pane.

1. In the Bash session within the Cloud Shell pane, run the following to connect to the Kubernetes cluster:

* az aks get-credentials --resource-group AZ500LAB09 --name MyKubernetesCluster

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

* kubectl get nodes
* **Note**: 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

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.

* ACRNAME=$(az acr list --resource-group AZ500LAB09 --query '[].{Name:name}' --output tsv)  
    
  az aks update -n MyKubernetesCluster -g AZ500LAB09 --attach-acr $ACRNAME
* **Note**: This command grants the ‘acrpull’ role assignment to the ACR.
* **Note**: It may take a few minutes for this command to complete.

1. In the Bash session within the Cloud Shell pane, run the following to grant the AKS cluster the Contributor role to its virtual network.

* RG\_AKS=AZ500LAB09  
    
  AKS\_VNET\_NAME=AZ500LAB09-vnet  
    
  AKS\_CLUSTER\_NAME=MyKubernetesCluster  
    
  AKS\_VNET\_ID=$(az network vnet show --name $AKS\_VNET\_NAME --resource-group $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 "Contributor" --scope $AKS\_VNET\_ID

#### Task 5: Deploy an external service to AKS

In this task, you will download the Manifest files, edit the YAML file, and apply your changes to the cluster.

1. In the Bash session within the Cloud Shell pane, click the **Upload/Download files** icon, in the drop-down menu, click **Upload**, in the **Open** dialog box, naviate to the location where you downloaded the lab files, select **\Allfiles\Labs\09\nginxexternal.yaml** click **Open**. Next, select **\Allfiles\Labs\09\nginxinternal.yaml**, and click **Open**.
2. In the Bash session within the Cloud Shell pane, run the following to identify the name of the Azure Container Registry instance:

* echo $ACRNAME
* **Note**: Record the Azure Container Registry instance name. You will need it later in this task.

1. In the Bash session within the Cloud Shell pane, run the following to open the nginxexternal.yaml file, so you can edit its content.

* code ./nginxexternal.yaml
* **Note**: This is the *external* yaml file.

1. In the editor pane, scroll down to **line 24** and replace the **<ACRUniquename>** placeholder with the ACR name.
2. In the editor pane, in the upper right corner, click the **ellipses** icon, click **Save** and then click **Close editor**.
3. In the Bash session within the Cloud Shell pane, run the following to apply the change to the cluster:

* kubectl apply -f nginxexternal.yaml

1. In the Bash session within the Cloud Shell pane, review the output of the command you run in the previous task to verify that the deployment and the corresponding service have been created.

* deployment.apps/nginxexternal created  
  service/nginxexternal created

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

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

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.

* kubectl get service nginxexternal

1. 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.
2. Open a new browser tab and browse to the IP address you identified in the previous step.
3. Ensure the **Welcome to nginx!** page displays.

#### Task 7: Deploy an internal service to AKS

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

1. In the Bash session within the Cloud Shell pane, run the following to open the nginxintenal.yaml file, so you can edit its content.

* code ./nginxinternal.yaml
* **Note**: This is the *internal* yaml file.

1. In the editor pane, scroll down to the line containing the reference to the container image and replace the **<ACRUniquename>** placeholder with the ACR name.
2. In the editor pane, in the upper right corner, click the **ellipses** icon, click **Save** and then click **Close editor**.
3. In the Bash session within the Cloud Shell pane, run the following to apply the change to the cluster:

* kubectl apply -f nginxinternal.yaml

1. In the Bash session within the Cloud Shell pane, review the output to verify your deployment and the service have been created:

* deployment.apps/nginxinternal created  
  service/nginxinternal created

1. In the Bash session within the Cloud Shell pane, run the following to retrieve information about the nginxinternal service including name, type, IP addresses, and ports.

* kubectl get service nginxinternal

1. In the Bash session within the Cloud Shell pane, review the output. The External-IP is, in this case, a private IP address. If it is in a **Pending** state then run the previous command again.

* **Note**: Record this IP address. You will need it in the next task.
* **Note**: To access the internal service endpoint, you will connect interactively to one of the pods running in the cluster.
* **Note**: Alternatively you could use the CLUSTER-IP address.

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

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

1. In the Bash session within the Cloud Shell pane, run the following to list the pods in the default namespace on the AKS cluster:

* kubectl get pods

1. In the listing of the pods, copy the first entry in the **NAME** column.

* **Note**: This is the pod you will use in the subsequent steps.

1. 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):

* kubectl exec -it <pod\_name> -- /bin/bash

1. 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):

* curl http://<internal\_IP>

1. Close the Cloud Shell pane.

Result: You have configured and secured ACR and AKS.

**Clean up resources**

Remember to remove any newly created Azure resources that you no longer use. Removing unused resources ensures you will not incur unexpected costs.

1. In the Azure portal, open the Cloud Shell by clicking the first icon in the top right of the Azure Portal.
2. In the upper-left drop-down menu of the Cloud Shell pane, select **PowerShell** and, when prompted, click **Confirm**.
3. In the PowerShell session within the Cloud Shell pane, run the following to remove the resource groups you created in this lab:

* Remove-AzResourceGroup -Name "AZ500LAB09" -Force -AsJob

1. Close the **Cloud Shell** pane.