

Week7: Assignment 2:- Configuring and Securing ACR and AKS

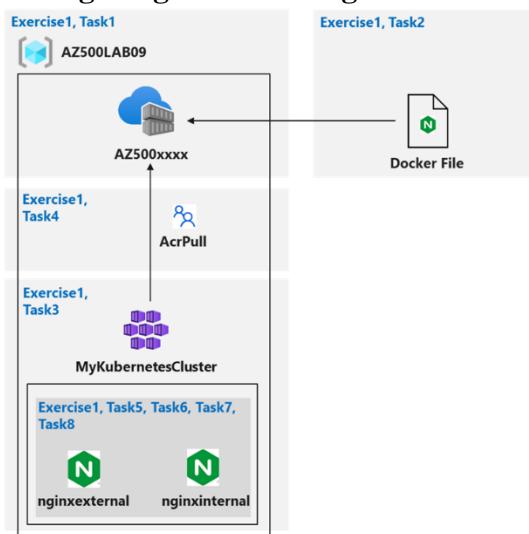
Report by: Aisha Khalifan, cs-cns04-23014

Introduction

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. Configuring and Securing ACR and AKS diagram

Configuring and Securing ACR and AKS diagram



Instructions

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.

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

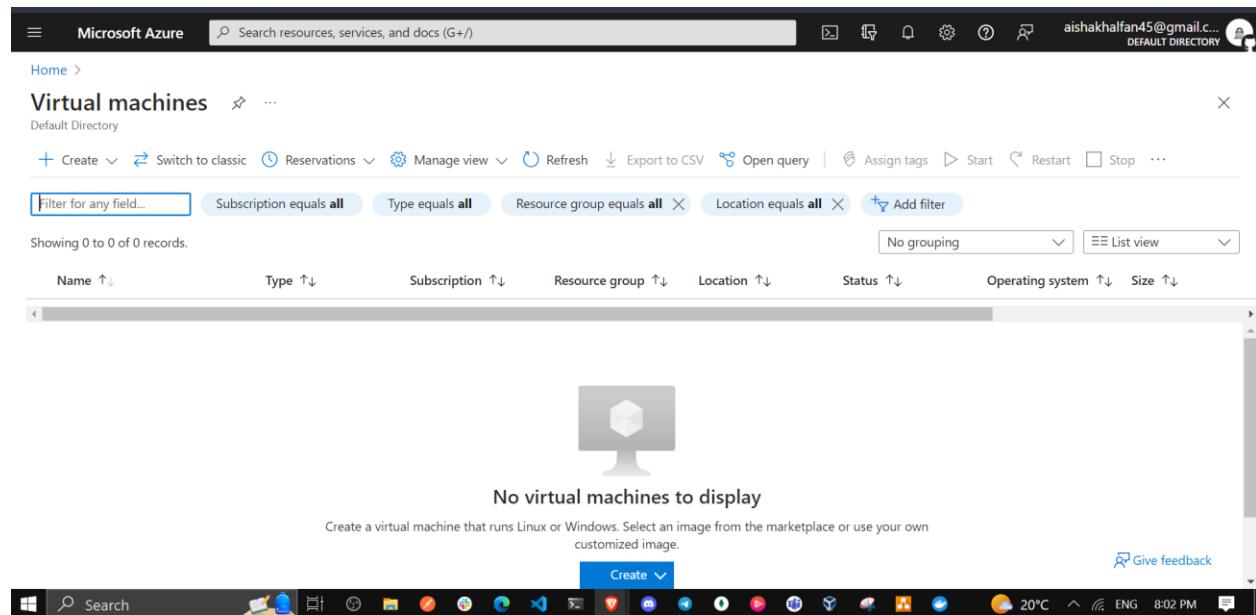
- Task 3: Create an Azure Kubernetes Service cluster
- Task 4: Grant the AKS cluster permissions to access the ACR
- Task 5: Deploy an external service to AKS
- Task 6: Verify the you can access an external AKS-hosted service
- Task 7: Deploy an internal service to AKS
- Task 8: Verify the you can access 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 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.



2. 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**.
3. Ensure **Bash** is selected in the drop-down menu in the upper-left corner of the Cloud Shell pane.

The screenshot shows the Microsoft Azure Cloud Shell interface. At the top, it says "Requesting a Cloud Shell. Succeeded. Connecting terminal...". Below that, it says "Welcome to Azure Cloud Shell" and provides instructions: "Type "az" to use Azure CLI" and "Type "help" to learn about Cloud Shell". The command prompt shows "aisha [~]\$". The bottom of the screen shows the Windows taskbar with various pinned icons.

4. 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
```

The screenshot shows the Microsoft Azure Cloud Shell interface. The command "az group create --name AZ500LAB09 --location eastus" is entered and executed. The output shows the creation of a new resource group with the following details:
{"id": "/subscriptions/3dbbb8d5-1829-48d8-a51d-c7f11b061059/resourceGroups/AZ500LAB09", "location": "eastus", "managedBy": null, "name": "AZ500LAB09", "properties": {"provisioningState": "Succeeded"}, "tags": null, "type": "Microsoft.Resources/resourceGroups"}
A survey message at the bottom encourages users to share their experience with Azure CLI.
[Survey] Help us improve Azure CLI by sharing your experience. This survey should take about 5 minutes. Run 'az survey' to open in browser. Learn more at <https://go.microsoft.com/fwlink/?linkid=2203309>

5. 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
```

The screenshot shows a Microsoft Azure Cloud Shell interface. The user is running a Bash session. They attempt to run the command `create storage`, which fails because the command does not exist. They then successfully run `az group create --name AZ500LAB09 --location eastus`. The output shows the creation of a new resource group with the specified properties. A survey message is displayed at the bottom. Finally, they run `az group list --query "[?name=='AZ500LAB09']" -o table` to verify the creation of the resource group, and the output shows a single row with Name AZ500LAB09 and Location eastus.

```
aisha [ ~ ]$ create storage
bash: create: command not found
aisha [ ~ ]$ az group create --name AZ500LAB09 --location eastus
{
  "id": "/subscriptions/3dbbb8d5-1829-48d8-a51d-c7f11b061059/resourceGroups/AZ500LAB09",
  "location": "eastus",
  "managedBy": null,
  "name": "AZ500LAB09",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null,
  "type": "Microsoft.Resources/resourceGroups"
}

[Survey] Help us improve Azure CLI by sharing your experience. This survey should take about 5 minutes. Run 'az survey' to open in browser. Learn more at https://go.microsoft.com/fwlink/?linkid=2203309

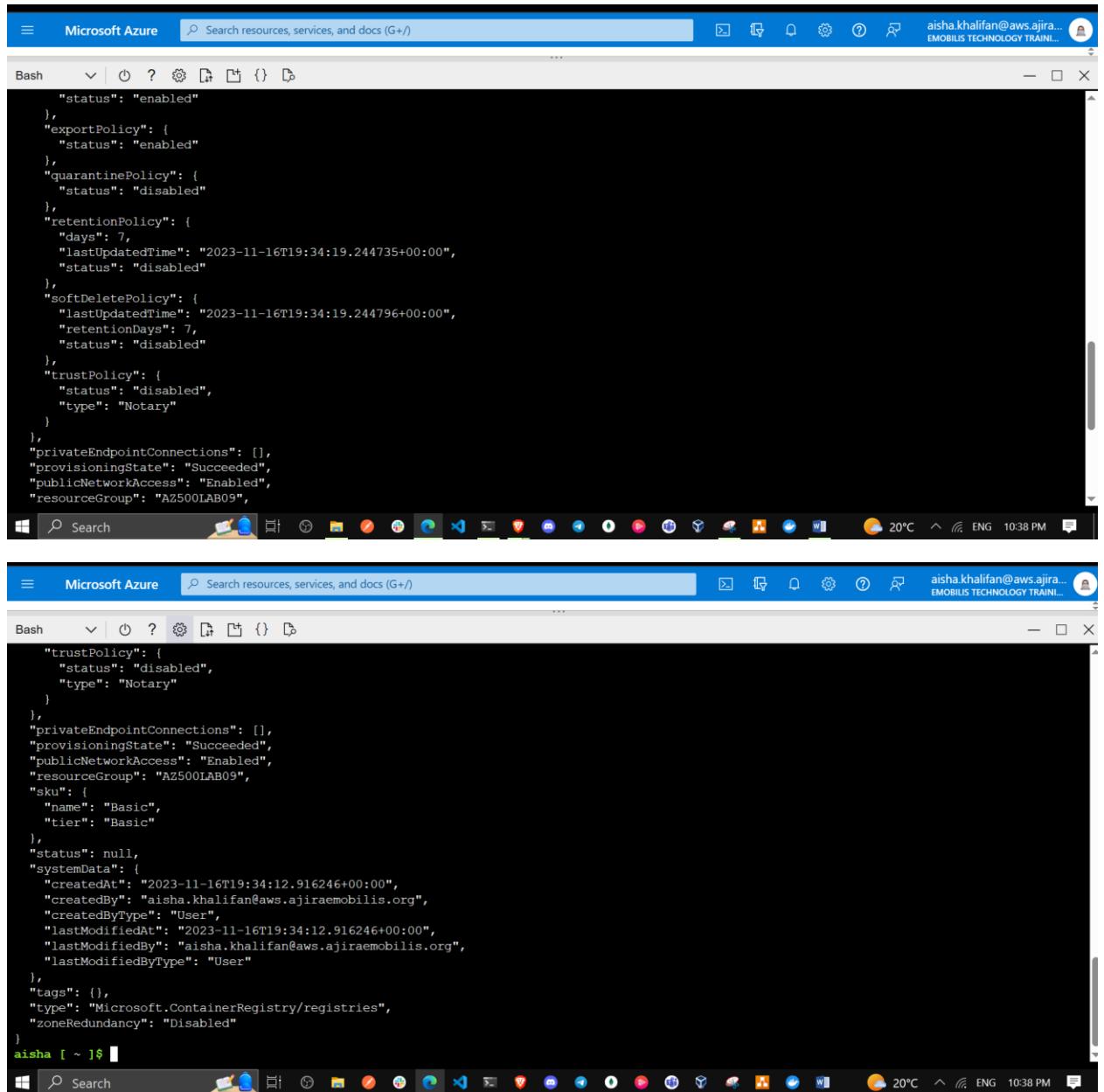
aisha [ ~ ]$ az group list --query "[?name=='AZ500LAB09']" -o table
Name          Location
-----
AZ500LAB09   eastus
aisha [ ~ ]$
```

- 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

The screenshot shows a Microsoft Azure Cloud Shell interface. The user runs the command `az acr create --resource-group AZ500LAB09 --name az500$RANDOM$RANDOM --sku Basic`. The output indicates that the Resource provider 'Microsoft.ContainerRegistry' is not registered, so it is being registered. The registration succeeds. The command then creates a new ACR instance with the specified properties, including its ID, location (eastus), and name (az500235959017.azurecr.io).

```
aisha [ ~ ]$ az acr create --resource-group AZ500LAB09 --name az500$RANDOM$RANDOM --sku Basic
Resource provider 'Microsoft.ContainerRegistry' used by this operation is not registered. We are registering for you.
Registration succeeded.
{
  "adminUserEnabled": false,
  "anonymousPullEnabled": false,
  "creationDate": "2023-11-16T19:34:12.916246+00:00",
  "dataEndpointEnabled": false,
  "dataEndpointHostNames": [],
  "encryption": {
    "keyVaultProperties": null,
    "status": "disabled"
  },
  "id": "/subscriptions/be98b827-9f6a-40b0-890a-52ceccfa74e6/resourceGroups/AZ500LAB09/providers/Microsoft.ContainerRegistry/registries/az500235959017",
  "identity": null,
  "location": "eastus",
  "loginServer": "az500235959017.azurecr.io",
  "name": "az500235959017",
  "networkRuleBypassOptions": "AzureServices",
  "networkRuleSet": null,
  "policies": {
    "azureAdAuthenticationAsArmPolicy": {
      "status": "enabled"
    },
    "exportPolicy": {
      "status": "enabled"
    }
  }
}
```



```

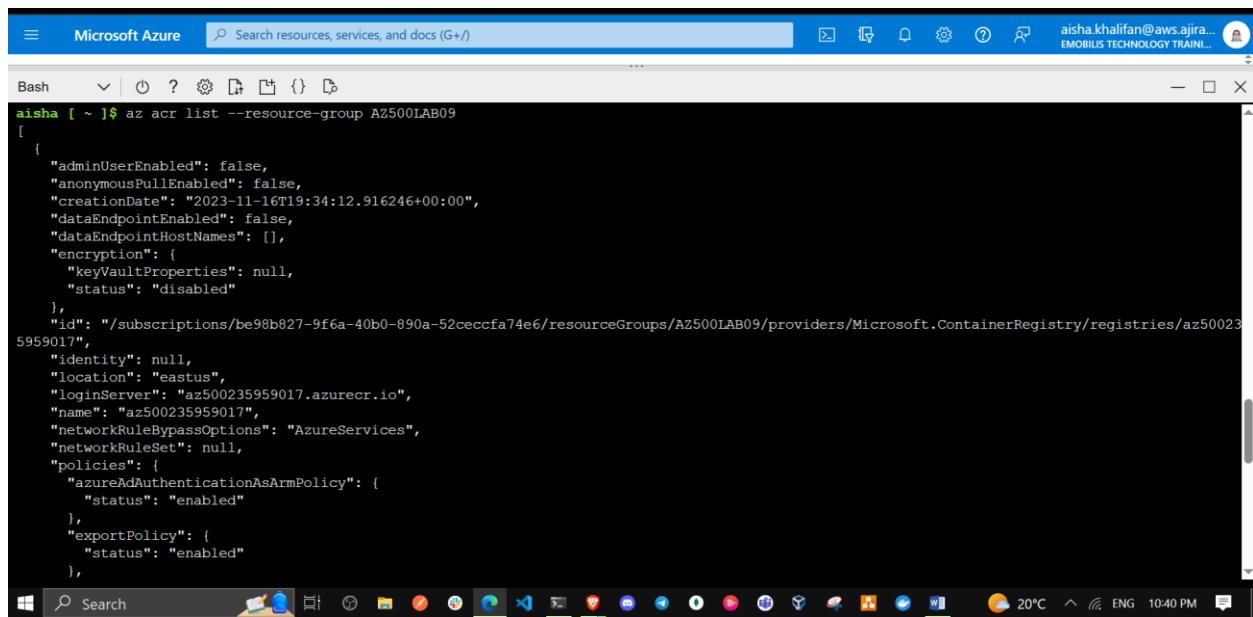
status: "enabled"
},
"exportPolicy": {
  "status": "enabled"
},
"quarantinePolicy": {
  "status": "disabled"
},
"retentionPolicy": {
  "days": 7,
  "lastUpdatedTime": "2023-11-16T19:34:19.244735+00:00",
  "status": "disabled"
},
"softDeletePolicy": {
  "lastUpdatedTime": "2023-11-16T19:34:19.244796+00:00",
  "retentionDays": 7,
  "status": "disabled"
},
"trustPolicy": {
  "status": "disabled",
  "type": "Notary"
}
},
"privateEndpointConnections": [],
"provisioningState": "Succeeded",
"publicNetworkAccess": "Enabled",
"resourceGroup": "AZ500LAB09",
sku": {
  "name": "Basic",
  "tier": "Basic"
},
"status": null,
"systemData": {
  "createdAt": "2023-11-16T19:34:12.916246+00:00",
  "createdBy": "aisha.khalifan@aws.ajiraemobilis.org",
  "createdByType": "User",
  "lastModifiedAt": "2023-11-16T19:34:12.916246+00:00",
  "lastModifiedBy": "aisha.khalifan@aws.ajiraemobilis.org",
  "lastModifiedByType": "User"
},
"tags": {},
"type": "Microsoft.ContainerRegistry/registries",
"zoneRedundancy": "Disabled"
}
aisha [ ~ ]$ 

```

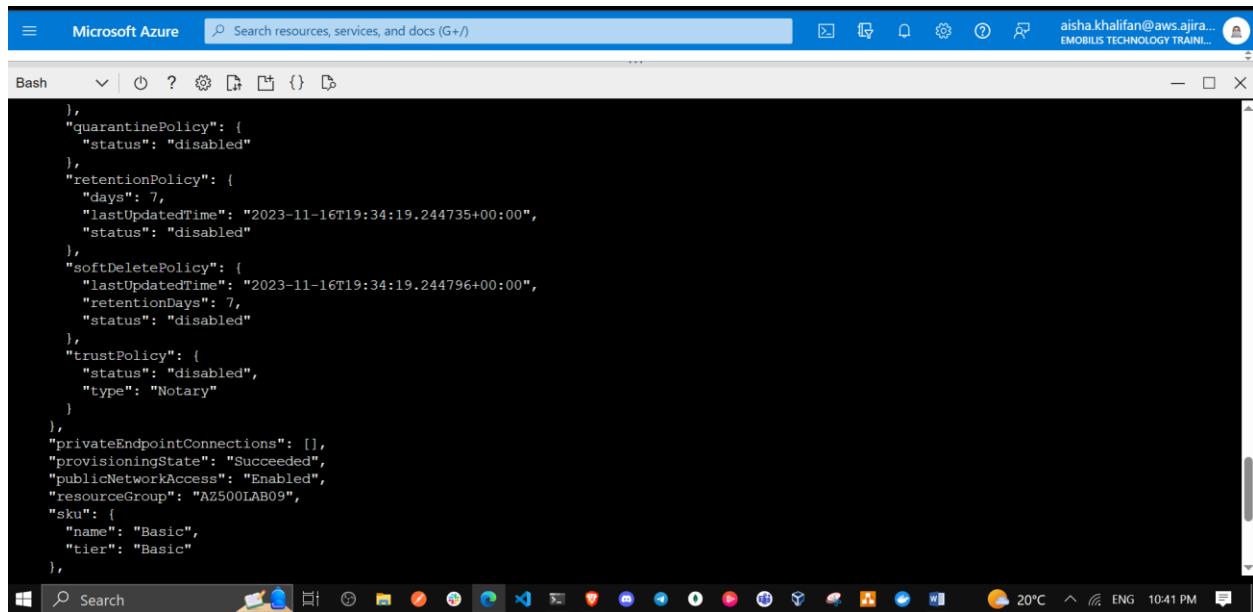
- In the Bash session within the Cloud Shell pane, run the following to confirm that the new ACR was created:

az acr list --resource-group AZ500LAB09

Note: Record the name of the ACR. You will need it in the next task.



```
aisha [ ~ ]$ az acr list --resource-group AZ500LAB09
[
  {
    "adminUserEnabled": false,
    "anonymousPullEnabled": false,
    "creationDate": "2023-11-16T19:34:12.916246+00:00",
    "dataEndpointEnabled": false,
    "dataEndpointHostNames": [],
    "encryption": {
      "keyVaultProperties": null,
      "status": "disabled"
    },
    "id": "/subscriptions/be98b827-9f6a-40b0-890a-52ceccfa74e6/resourceGroups/AZ500LAB09/providers/Microsoft.ContainerRegistry/registries/az500235959017",
    "identity": null,
    "location": "eastus",
    "loginServer": "az500235959017.azurecr.io",
    "name": "az500235959017",
    "networkRuleBypassOptions": "AzureServices",
    "networkRuleSet": null,
    "policies": {
      "azureAdAuthenticationAsArmPolicy": {
        "status": "enabled"
      },
      "exportPolicy": {
        "status": "enabled"
      }
    },
    "quarantinePolicy": {
      "status": "disabled"
    },
    "retentionPolicy": {
      "days": 7,
      "lastUpdatedTime": "2023-11-16T19:34:19.244735+00:00",
      "status": "disabled"
    },
    "softDeletePolicy": {
      "lastUpdatedTime": "2023-11-16T19:34:19.244796+00:00",
      "retentionDays": 7,
      "status": "disabled"
    },
    "trustPolicy": {
      "status": "disabled",
      "type": "Notary"
    }
  }
],
```



```
  ],
  "privateEndpointConnections": [],
  "provisioningState": "Succeeded",
  "publicNetworkAccess": "Enabled",
  "resourceGroup": "AZ500LAB09",
  "sku": {
    "name": "Basic",
    "tier": "Basic"
  },
  "softDeletePolicy": {
    "lastUpdatedTime": "2023-11-16T19:34:19.244796+00:00",
    "retentionDays": 7,
    "status": "disabled"
  },
  "trustPolicy": {
    "status": "disabled",
    "type": "Notary"
  }
},
```

The screenshot shows a Microsoft Azure Cloud Shell interface. The title bar says "Microsoft Azure" and "Search resources, services, and docs (G+/" followed by the user's email "aisha.khalifan@aws.ajira...". The main pane is a "Bash" session with a black background. It displays a JSON object representing a resource configuration. The JSON includes fields like "status": "disabled", "type": "Notary", "privateEndpointConnections": [], "provisioningState": "Succeeded", "publicNetworkAccess": "Enabled", "resourceGroup": "AZ500LAB09", "sku": { "name": "Basic", "tier": "Basic" }, "status": null, "systemData": { "createdAt": "2023-11-16T19:34:12.916246+00:00", "createdBy": "aisha.khalifan@aws.ajiraemobilis.org", "createdByType": "User", "lastModifiedAt": "2023-11-16T19:34:12.916246+00:00", "lastModifiedBy": "aisha.khalifan@aws.ajiraemobilis.org", "lastModifiedByType": "User" }, "tags": {}, "type": "Microsoft.ContainerRegistry/registries", "zoneRedundancy": "Disabled" }. The prompt at the bottom is "aisha [~]\$".

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
```

The screenshot shows a Microsoft Azure Cloud Shell interface. The title bar says "Microsoft Azure" and "Search resources, services, and docs (G+/" followed by the user's email "aisha.khalifan@aws.ajira...". The main pane is a "Bash" session with a black background. It shows the command "echo FROM nginx > Dockerfile" being typed. The prompt at the bottom is "aisha [~]\$".

2. 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.

```
aisha [ ~ ]$ echo FROM nginx > Dockerfile
aisha [ ~ ]$ 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 .
Packing source code into tar to upload...
Uploading archived source code from '/tmp/build_archive_bc3e2e37f2db42198d9aa68b38d306b7.tar.gz'...
Sending context (13.970 KiB) to registry: az500235959017...
Queued a build with ID: cal
Waiting for an agent...
2023/11/16 19:44:23 Downloading source code...
2023/11/16 19:44:24 Finished downloading source code
2023/11/16 19:44:24 Using acb_voi_flea38f7-e05d-41b7-87ed-a91afe6771de as the home volume
2023/11/16 19:44:24 Setting up Docker configuration...
2023/11/16 19:44:24 Successfully set up Docker configuration
2023/11/16 19:44:24 Logging in to registry: az500235959017.azurecr.io
2023/11/16 19:44:25 Successfully logged into az500235959017.azurecr.io
2023/11/16 19:44:25 Executing step ID: build. Timeout(sec): 28800, Working directory: '', Network: ''
2023/11/16 19:44:25 Scanning for dependencies...
2023/11/16 19:44:26 Successfully scanned dependencies
2023/11/16 19:44:26 Launching container with name: build
Sending build context to Docker daemon 78.34kB
Step 1/1 : FROM nginx
latest: Pulling from library/nginx
578acb154839: Pulling fs layer
e398db710407: Pulling fs layer
85c41ebbe6d66: Pulling fs layer
7170a263b582: Pulling fs layer
```

```
cldfc7e1671e: Pulling fs layer
8f28d06e2e2e: Waiting
6f837de2f887: Waiting
cldfc7e1671e: Waiting
7170a263b582: Waiting
85c41ebbe6d66: Download complete
7170a263b582: Verifying Checksum
7170a263b582: Download complete
8f28d06e2e2e: Verifying Checksum
8f28d06e2e2e: Download complete
6f837de2f887: Verifying Checksum
6f837de2f887: Download complete
e398db710407: Verifying Checksum
e398db710407: Download complete
578acb154839: Verifying Checksum
578acb154839: Download complete
cldfc7e1671e: Verifying Checksum
cldfc7e1671e: Download complete
578acb154839: Pull complete
e398db710407: Pull complete
85c41ebbe6d66: Pull complete
7170a263b582: Pull complete
8f28d06e2e2e: Pull complete
6f837de2f887: Pull complete
cldfc7e1671e: Pull complete
Digest: sha256:86e53c4c16a6a276b204b0fd3a8143d86547c967dc8258b3d47c3a21bb68d3c6
Status: Downloaded newer image for nginx:latest
```

```

Status: Downloaded newer image for nginx:latest
--> c20060033e06
Successfully built c20060033e06
Successfully tagged az500235959017.azurecr.io/sample/nginx:v1
2023/11/16 19:44:30 Successfully executed container: build
2023/11/16 19:44:30 Executing step ID: push. Timeout(sec): 3600, Working directory: '', Network: ''
2023/11/16 19:44:30 Pushing image: az500235959017.azurecr.io/sample/nginx:v1, attempt 1
The push refers to repository [az500235959017.azurecr.io/sample/nginx]
505f49f13fbe: Preparing
9920f1ebf52b: Preparing
768e28a222fd: Preparing
715b32fa0f12: Preparing
e503754c9a26: Preparing
609f2a18d224: Preparing
ec983b16360: Preparing
609f2a18d224: Waiting
ec983b16360: Waiting
505f49f13fbe: Pushed
e503754c9a26: Pushed
768e28a222fd: Pushed
715b32fa0f12: Pushed
9920f1ebf52b: Pushed
ec983b16360: Pushed
609f2a18d224: Pushed
v1: digest: sha256:d2e65182b5fd330470eca9b8e23e8ala0d87cc9b820eb1fb3f034bf8248d37ee size: 1778
2023/11/16 19:44:37 Successfully pushed image: az500235959017.azurecr.io/sample/nginx:v1
2023/11/16 19:44:37 Step ID: build marked as successful (elapsed time in seconds: 4.509885)

```



```

768e28a222fd: Pushed
715b32fa0f12: Pushed
9920f1ebf52b: Pushed
ec983b16360: Pushed
609f2a18d224: Pushed
v1: digest: sha256:d2e65182b5fd330470eca9b8e23e8ala0d87cc9b820eb1fb3f034bf8248d37ee size: 1778
2023/11/16 19:44:37 Successfully pushed image: az500235959017.azurecr.io/sample/nginx:v1
2023/11/16 19:44:37 Step ID: build marked as successful (elapsed time in seconds: 4.509885)
2023/11/16 19:44:37 Populating digests for step ID: build...
2023/11/16 19:44:38 Successfully populated digests for step ID: build
2023/11/16 19:44:38 Step ID: push marked as successful (elapsed time in seconds: 7.275858)
2023/11/16 19:44:38 The following dependencies were found:
2023/11/16 19:44:38
- image:
  registry: az500235959017.azurecr.io
  repository: sample/nginx
  tag: v1
  digest: sha256:d2e65182b5fd330470eca9b8e23e8ala0d87cc9b820eb1fb3f034bf8248d37ee
  runtime-dependency:
    registry: registry.hub.docker.com
    repository: library/nginx
    tag: latest
    digest: sha256:86e53c4c16a6a276b204bfd3a8143d86547c967dc8258b3d47c3a21bb68d3c6
  git: {}

Run ID: cal was successful after 15s
aisha [ ~ ]$ 

```

3. Close the Cloud Shell pane.
4. 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.

Azure Resource Groups blade for AZ500LAB09. The 'Overview' tab is selected, showing one resource: az500235959017 (Container registry) in East US.

5. On the Container registry blade, in the **Services** section, click **Repositories**.

Azure Container Registry blade for az500235959017. The 'Repositories' section is highlighted with a red box and a red '3' annotation. The table shows one repository: sample/nginx.

6. Verify that the list of repositories includes the new container image named **sample/nginx**.

The screenshot shows the Microsoft Azure portal interface. The top navigation bar includes the Microsoft Azure logo, a search bar, and user information (aisha.khalifan@aws.ajira...). Below the navigation is a breadcrumb trail: Home > Resource groups > AZ500LAB09 > az500235959017. The main title is 'az500235959017 | Repositories'. On the left, there's a sidebar with 'Container registry' settings like Access keys, Encryption, Identity, Networking, Microsoft Defender for Cloud, Properties, and Locks. Under 'Services', 'Repositories' is selected. The main content area shows a list of repositories with 'sample/nginx' highlighted by a red box. A search bar at the top also has 'sample/nginx' typed into it, with another red box highlighting it.

7. Click the **sample/nginx** entry and verify presence of the **v1** tag that identifies the image version.

This screenshot shows the details for the 'sample/nginx' repository within the Azure Container Registry. The top navigation bar and sidebar are identical to the previous screenshot. The main title is 'sample/nginx'. The 'Essentials' section shows the repository name, tag count (1), last updated date (11/16/2023, 10:44 PM GMT+3), and manifest count (1). Below this is a 'Tags' table with one entry: 'v1'. A red box highlights the 'v1' tag. The table also includes columns for Digest (sha256:d2e65182b5fd330470eca...) and Last modified (11/16/2023, 10:44 PM GMT+3).

Tags ↑↓	Digest ↑↓	Last modified
v1	sha256:d2e65182b5fd330470eca...	11/16/2023, 10:44 PM GMT+3

8. Click the **v1** entry to view the image manifest.

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

Microsoft Azure | Search resources, services, and docs (G+/)

Home > Resource groups > AZ500LAB09 > az500235959017 | Repositories > sample/nginx >

sample/nginx

Repository sample/nginx

Last updated date 11/16/2023, 10:44 PM GMT+3

Tag count 1

Manifest count 1

Search to filter tags ...

Tags ↑↓ Digest ↑↓

Tags	Digest
v1	sha256:d2e65182b5fd330470eca9b8e23e8a1a0d87cc9b820eb1fb3f034bf8248d37ee

Essentials JSON View

sample/nginx:v1

sha256:d2e65182b5fd330470eca9b8e23e8a1a0d87cc9b820eb1fb3f034bf8248d37ee

+ Create streaming artifact Refresh

Essentials

Repository sample/nginx

Tag v1

Tag creation date 11/16/2023, 10:44 PM GMT+3

Tag last updated date 11/16/2023, 10:44 PM GMT+3

Digest sha256:d2e65182b5fd330470eca9b8e23e8a1a0d87cc9b820eb1fb3f034bf8248d37ee

Manifest creation date 11/16/2023, 10:44 PM GMT+3

Platform linux / amd64

Media type application/vnd.docker.distribution.manifest.v2+json

Run ID ca1

Manifest Referrers

Docker pull command docker pull az500235959017.azurecr.io/sample/nginx:v1

Manifest

```
{ "schemaVersion": 2, "mediaType": "application/vnd.docker.distribution.manifest.v2+json", "config": { "mediaType": "application/vnd.docker.container.image.v1+json", "size": 8148, "digest": "sha256:c20060033e06f882b0fbe2db7d974d72e0887a3be554efdb0dcf8d53512647" }, "layers": [ { "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip", "size": 29149836, "digest": "sha256:578acb154839e9d0034432e8f53756d6f53ba62cf8c7ea5218a2476bf5b58fc9" }, { "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip", "size": 41378344, "digest": "sha256:e398db710407fbc310b4bc0b0db1c94161480ac9b44638c6655939f426529780" }, { "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip", "size": 627, "digest": "sha256:20060033e06f882b0fbe2db7d974d72e0887a3be554efdb0dcf8d53512647" } ] }
```

Windows taskbar: Search, Start button, Task View, File Explorer, Taskbar icons, 20°C, ENG, 10:51 PM

Microsoft Azure | Search resources, services, and docs (G+/)

Home > Resource groups > AZ500LAB09 > az500235959017 | Repositories > sample/nginx >

sample/nginx

Repository sample/nginx

Last updated date 11/16/2023, 10:44 PM GMT+3

Tag count 1

Manifest count 1

Search to filter tags ...

Tags ↑↓ Digest ↑↓

Tags	Digest
v1	sha256:d2e65182b5fd330470eca9b8e23e8a1a0d87cc9b820eb1fb3f034bf8248d37ee

Essentials JSON View

sample/nginx:v1

sha256:d2e65182b5fd330470eca9b8e23e8a1a0d87cc9b820eb1fb3f034bf8248d37ee

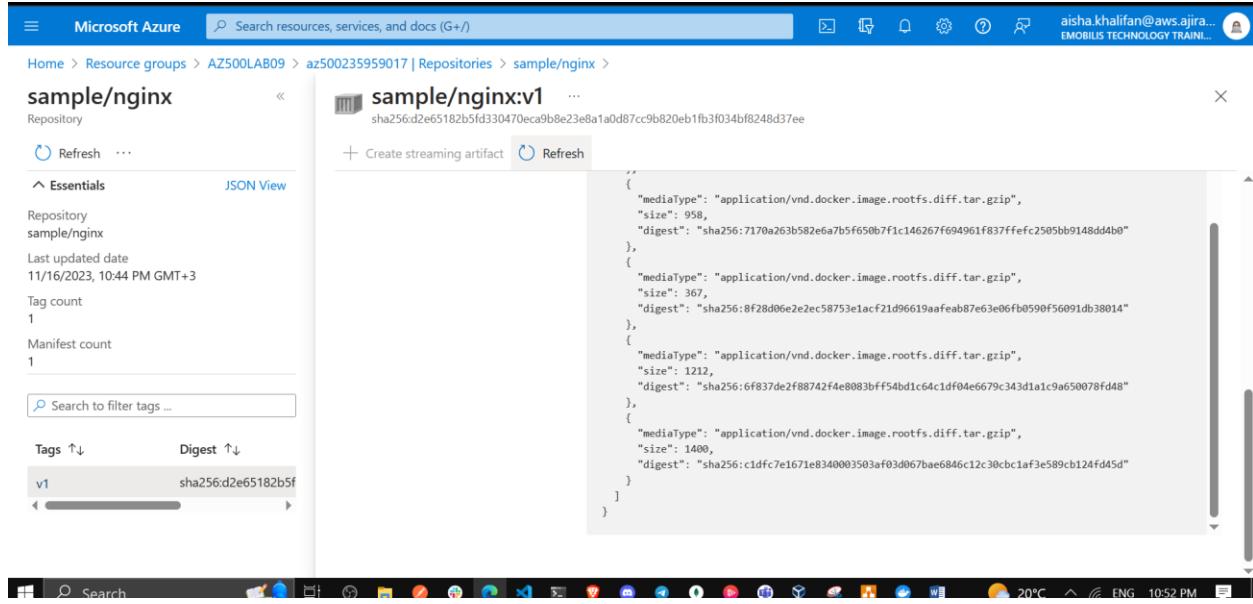
+ Create streaming artifact Refresh

Docker pull command docker pull az500235959017.azurecr.io/sample/nginx:v1

Manifest

```
{ "schemaVersion": 2, "mediaType": "application/vnd.docker.distribution.manifest.v2+json", "config": { "mediaType": "application/vnd.docker.container.image.v1+json", "size": 8148, "digest": "sha256:c20060033e06f882b0fbe2db7d974d72e0887a3be554efdb0dcf8d53512647" }, "layers": [ { "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip", "size": 29149836, "digest": "sha256:578acb154839e9d0034432e8f53756d6f53ba62cf8c7ea5218a2476bf5b58fc9" }, { "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip", "size": 41378344, "digest": "sha256:e398db710407fbc310b4bc0b0db1c94161480ac9b44638c6655939f426529780" }, { "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip", "size": 627, "digest": "sha256:20060033e06f882b0fbe2db7d974d72e0887a3be554efdb0dcf8d53512647" } ] }
```

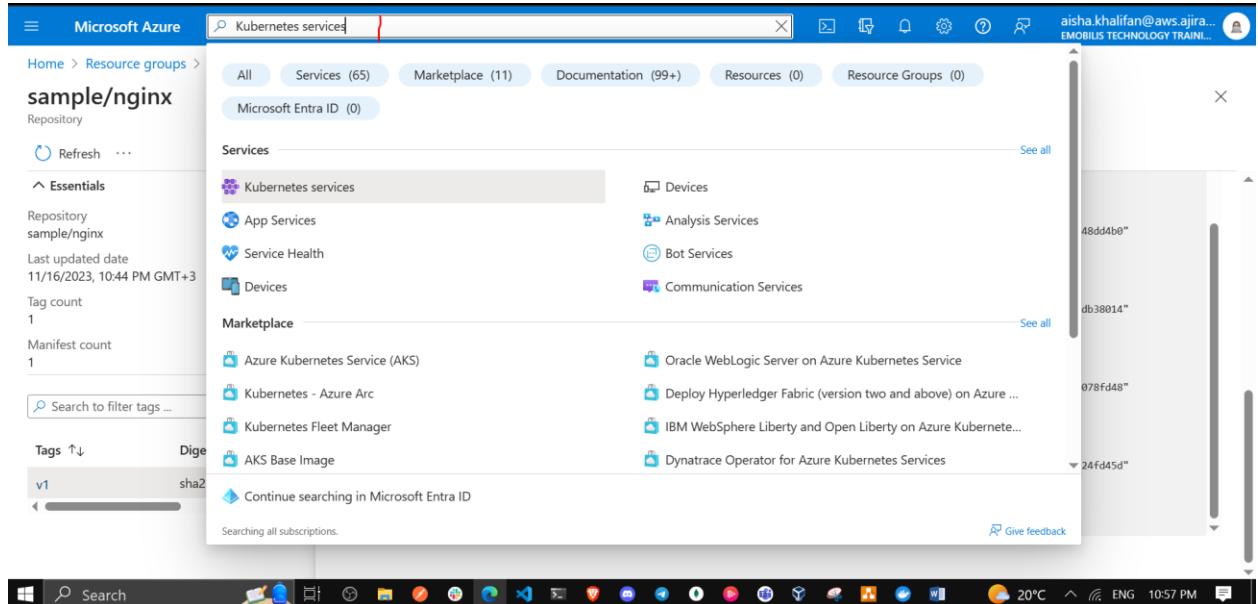
Windows taskbar: Search, Start button, Task View, File Explorer, Taskbar icons, 20°C, ENG, 10:52 PM



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**

The screenshot shows the Microsoft Azure Kubernetes services blade. At the top, there's a search bar and a navigation bar with options like 'Create', 'Manage view', 'Refresh', 'Export to CSV', 'Open query', and 'Assign tags'. Below the search bar, there are filter buttons for 'Type equals all', 'Resource group equals all', 'Location equals all', and a 'Add filter' button. The main area displays a message: 'No Kubernetes services to display'. It includes a small icon of a cluster of three cubes and a note: 'Use Azure Kubernetes Service to create and manage Kubernetes clusters. Azure will handle cluster operations, including creating, scaling, and upgrading, freeing up developers to focus on their application. To get started, create a cluster with Azure Kubernetes Service.' There's also a 'Give feedback' link. The bottom of the screen shows the Windows taskbar with various pinned icons.

- 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

Microsoft Azure

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

Home > Kubernetes services >

Create Kubernetes cluster

Project details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * (Azure subscription 1) | Resource group * (AZ500LAB09) | Create new

Cluster details

Cluster preset configuration (Production Standard) | To quickly customize your Kubernetes cluster, choose one of the preset configurations above. You can modify these configurations at any time. Learn more and compare presets

Kubernetes cluster name * (MyKubernetesCluster) | Region * (US) East US

The name can contain only letters, numbers, underscores, and hyphens. The name must start and end with a letter or number.
Kubernetes service name must be unique in the current resource group.

< Previous | Next : Node pools > | Review + create | Give feedback

Microsoft Azure

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

Home > Kubernetes services >

Create Kubernetes cluster

Kubernetes cluster name * (MyKubernetesCluster)

Region * (US) East US

Availability zones (None)

AKS pricing tier (Standard)

Kubernetes version * (1.27.7 (default))

Automatic upgrade (Disabled)

Choose between local accounts or Azure AD for authentication and Azure RBAC or Kubernetes RBAC for your authorization needs.

Authentication and Authorization (Local accounts with Kubernetes RBAC)

< Previous | Next : Node pools > | Review + create | Give feedback

- 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

Create Kubernetes cluster

Add node pool

Name	Mode	Node size	OS SKU	Node count
agentpool	System	Standard_D8ds_v5 (change)	Ubuntu	2 - 5
userpool	User	Standard_D8ds_v5 (change)	Ubuntu	2 - 100

Enable virtual nodes

Virtual nodes allow burstable scaling backed by serverless Azure Container Instances. [Learn more about virtual nodes](#)

Enable virtual nodes

Node pool OS disk encryption

By default, all disks in AKS are encrypted at rest with Microsoft-managed keys. For additional control over encryption, you can supply your own keys using a disk encryption set backed by an Azure Key Vault. The disk encryption set will be used to encrypt the OS disks for all node pools in the cluster. [Learn more](#)

< Previous Review + create

- Click **Next: Access >**, on the **Create Kubernetes cluster** blade, accept the defaults, and click **Next: Networking >**.

Create Kubernetes cluster

Basics Node pools **Networking** Integrations Advanced Tags Review + create

Azure provides various networking controls to help manage and secure access to your Kubernetes cluster.

Private access

Enable a private cluster to restrict worker node to API access, enhancing your Kubernetes workload's security and isolation.

Enable private cluster

Public access

Set authorized IP ranges

Container networking

Network configuration

kubenet Best for smaller node pools. Each pod is assigned a logically different IP address from the subnet for simpler setup

Azure CNI

< Previous Review + create

- 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](#) page.

The screenshot shows the 'Create Kubernetes cluster' blade in the Azure portal. Under 'Container networking', 'Network configuration' is set to 'Azure CNI'. A 'DNS name prefix' field contains 'MyKubernetesCluster-dns'. 'Network policy' is set to 'Calico'. Navigation buttons at the bottom include '< Previous', 'Next : Integrations >', and 'Review + create'.

7. 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.

The screenshot shows the 'Create Kubernetes cluster' blade with the 'Integrations' tab selected. It includes sections for 'Microsoft Defender for Cloud', 'Azure Container Registry', and 'Azure Monitor'. Under 'Container registry', the dropdown shows 'None'. Navigation buttons at the bottom include '< Previous', 'Next : Advanced >', and 'Review + create'.

The screenshot shows the 'Create Kubernetes cluster' wizard on the 'Alerting' step. At the top, there's a 'Create new' button. Below it, the 'Azure Monitor' section is visible, with a note about enabling Container Insights for more comprehensive data on performance and health. There are three options for monitoring: 'Default configuration' (selected), 'Custom configuration', and 'Off'. Under 'Alerting', the 'Enable recommended alert rules' checkbox is checked. The 'Alert me if' dropdown is open, showing several alert conditions. At the bottom of the page are navigation buttons: '< Previous', 'Next : Advanced >', and 'Review + create'.

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

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

The screenshot shows the 'Create Kubernetes cluster' wizard on the 'Review + create' step. A green header bar indicates 'Validation passed'. Below it, the 'Review + create' tab is selected. The 'Basics' section displays the following configuration:

Subscription	Azure subscription 1
Resource group	AZ500LAB09
Region	East US
Kubernetes cluster name	MyKubernetesCluster
Kubernetes version	1.27.7
Automatic upgrade	Patch

The 'Node pools' section is collapsed. At the bottom, there are navigation buttons: '< Previous', 'Next >', and a prominent blue 'Create' button. There's also a link to 'Download a template for automation'.

The screenshot shows the Microsoft Azure portal with the search bar at the top containing 'Search resources, services, and docs (G+)'. The main title is 'microsoft.aks-20231116233704 | Overview'. On the left, there's a sidebar with 'Overview', 'Inputs', 'Outputs', and 'Template' options. The main content area displays a message 'Deployment is in progress' with deployment details: Deployment name: microsoft.aks-20231116..., Start time: 11/16/2023, 11:40:25 PM, Subscription: Azure subscription 1, Correlation ID: ee940a5f-5e76-480a-8e4d-9aacd82b4c09, Resource group: AZ500LAB09. Below this, there's a table titled 'Deployment details' showing two entries: 'MyKubernetesCluster' (Microsoft.ContainerService) and 'InsightsActionGroupDep' (Microsoft.Resources). A sidebar on the right provides links to 'Container Insights', 'Free Microsoft tutorials', and 'Work with an expert'.

- 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.

The screenshot shows the Microsoft Azure portal with the search bar at the top containing 'Search resources, services, and docs (G+)'. The main title is 'microsoft.aks-20231116233704 | Overview'. The sidebar on the left shows 'Overview', 'Inputs', 'Outputs', and 'Template'. The main content area now displays a message 'Your deployment is complete' with deployment details: Deployment name: microsoft.aks-20231116..., Start time: 11/16/2023, 11:40:25 PM, Subscription: Azure subscription 1, Correlation ID: ee940a5f-5e76-480a-8e4d-9aacd82b4c09, Resource group: AZ500LAB09. Below this, there's a section titled 'Next steps' with four recommended actions: 'Create a quick start application' (Recommended), 'Create a Kubernetes deployment' (Recommended), 'Integrate automatic deployments within your cluster' (Recommended), and 'Connect to cluster' (Recommended). A sidebar on the right provides links to 'Cost Management', 'Container Insights', 'Free Microsoft tutorials', and 'Work with an expert'.

- 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.

Resource groups
eMobilis Technology Training Institute

MC_AZ500LAB09_MyKubernetesCluster_eastus

Resources

Name	Type	Location
846c4e02-f73c-48d9-afc5-6cec60b86b76	Public IP address	East US
aks-agentpool-30198516-nsg	Network security group	East US
aks-agentpool-30345461-vmss	Virtual machine scale s...	East US
aks-vnet-30198516	Virtual network	East US
kubernetes	Load balancer	East US
MyKubernetesCluster-agentpool	Managed Identity	East US

11. 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.

Resource groups
eMobilis Technology Training Institute

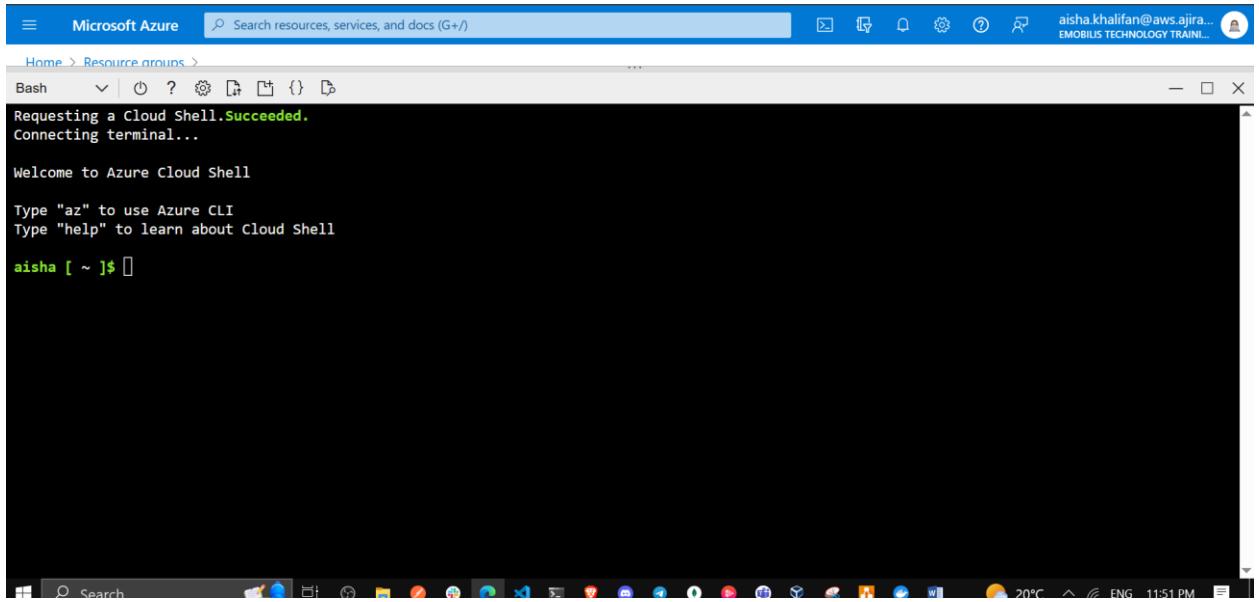
AZ500LAB09

Resources

Name	Type	Location
az500235959017	Container registry	East US
CPU Usage Percentage - MyKubernetesClu...	Metric alert rule	Global
Memory Working Set Percentage - MyKub...	Metric alert rule	Global
MyKubernetesCluster	Kubernetes service	East US
RecommendedAlertRules-AG-1	Action group	Global

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

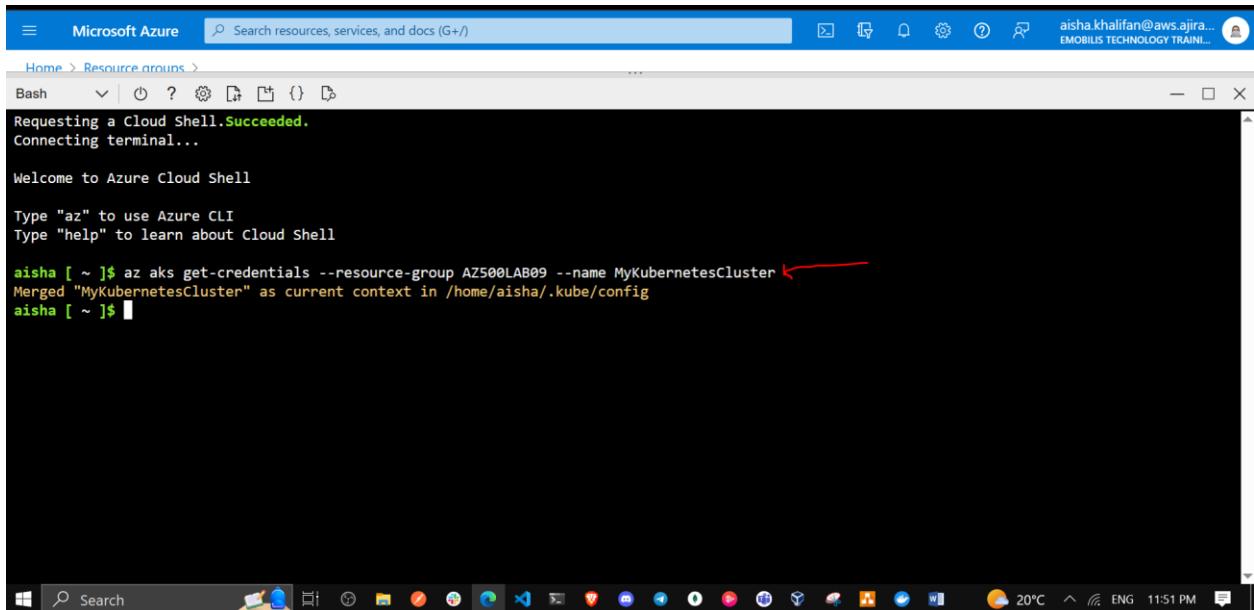
Note: Ensure **Bash** is selected in the drop-down menu in the upper-left corner of the Cloud Shell pane.



The screenshot shows the Microsoft Azure Cloud Shell interface. At the top, there's a blue header bar with the Microsoft Azure logo, a search bar, and user information. Below the header is a toolbar with icons for file operations like copy, paste, and refresh. The main area is a dark terminal window. It displays the following text:
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
aisha [~]\$

13. 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



The screenshot shows the Microsoft Azure Cloud Shell interface. The terminal window displays the following text:
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
aisha [~]\$ az aks get-credentials --resource-group AZ500LAB09 --name MyKubernetesCluster ↵
Merged "MyKubernetesCluster" as current context in /home/aisha/.kube/config
aisha [~]\$

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

kubectl get nodes

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

```
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

aisha [ ~ ]$ az aks get-credentials --resource-group AZ500LAB09 --name MyKubernetesCluster
Merged "MyKubernetesCluster" as current context in /home/aisha/.kube/config
aisha [ ~ ]$ kubectl get nodes
NAME           STATUS  ROLES   AGE    VERSION
aks-agentpool-30345461-vmss000000  Ready   agent   9m21s  v1.27.7
aks-agentpool-30345461-vmss000001  Ready   agent   9m27s  v1.27.7
aisha [ ~ ]$
```

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.

```
Microsoft Azure Search resources, services, and docs (G+/) aisha.khalifan@aws.ajira...  
EMOBILIS TECHNOLOGY TRAINING  
Home > Resource groups > ...  
Bash | ? | ⚡ | ⚙️ | 📁 | { } | 🔍  
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  
  
aisha [ ~ ]$ az aks get-credentials --resource-group AZ500LAB09 --name MyKubernetesCluster  
Merged "MyKubernetesCluster" as current context in /home/aisha/.kube/config  
aisha [ ~ ]$ kubectl get nodes  
NAME STATUS ROLES AGE VERSION  
aks-agentpool-30345461-vmss00000 Ready agent 9m21s v1.27.7  
aks-agentpool-30345461-vmss00001 Ready agent 9m27s v1.27.7  
aisha [ ~ ]$ ACRNAME=$(az acr list --resource-group AZ500LAB09 --query '[].{Name:name}' --output tsv)  
  
az aks update -n MyKubernetesCluster -g AZ500LAB09 --attach-acr $ACRNAME  
[ Running ...
```

Microsoft Azure Search resources, services, and docs (G+) ? ! ? ? ?

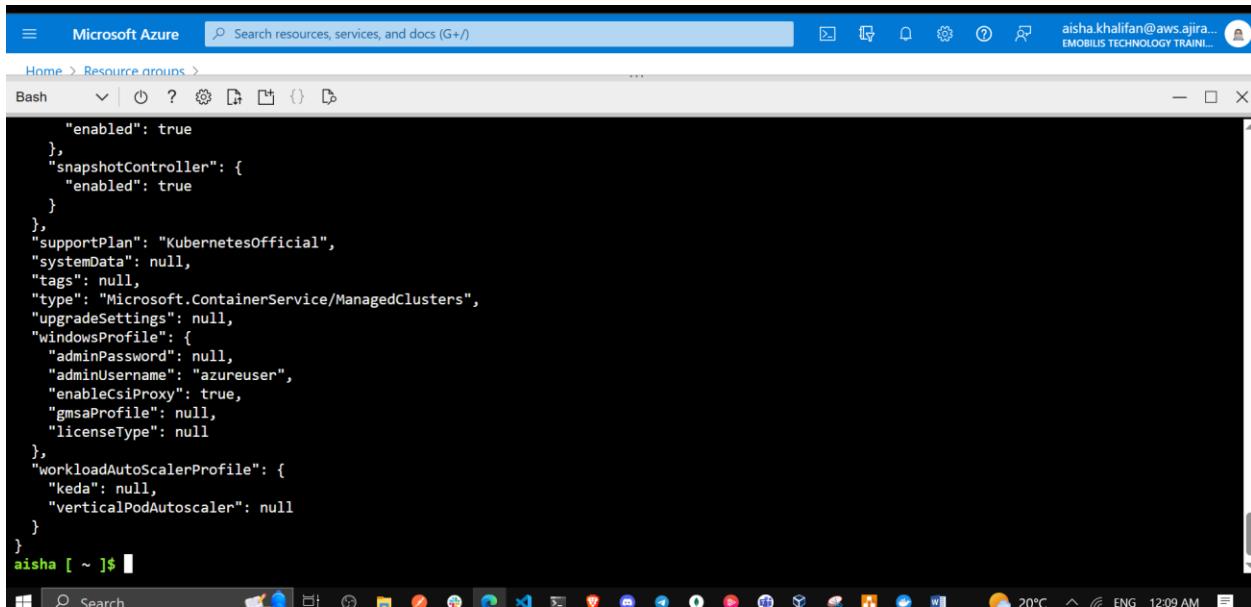
aisha [~]\$ ACRNAME=\$(az acr list --resource-group AZ500LAB09 --query '[].{Name:name}' --output tsv)

```
az aks update -n MyKubernetesCluster -g AZ500LAB09 --attach-acr $ACRNAME
AAD role propagation done[########################################] 100.0000%
  "aadProfile": null,
  "addonProfiles": {
    "azureKeyvaultSecretsProvider": {
      "config": null,
      "enabled": false,
      "identity": null
    },
    "azurepolicy": {
      "config": null,
      "enabled": false,
      "identity": null
    }
  },
  "agentPoolProfiles": [
    {
      "availabilityZones": null,
      "count": 2,
      "creationData": null,
      "currentOrchestratorVersion": "1.27.7",
      "enableAutoScaling": true,
      "enableEncryptionAtHost": null,
      "osDiskSizeGb": 30,
      "osType": "Linux",
      "podCidr": "10.240.0.0/16",
      "type": "VirtualMachine"
    }
  ],
  "apiServerIpRanges": [
    "10.240.0.0/16"
  ],
  "apiserverPort": 4466,
  "controlPlaneVmCount": 3,
  "controlPlaneVmSize": "Standard_D2s_v3",
  "dnsPrefix": "myakscluster",
  "enableHorizontalPodAutoscaling": true,
  "enableIngress": true,
  "enableVirtualMachineScaleSets": false,
  "fqdn": "myakscluster.azk8s.local",
  "httpPort": 4467,
  "imagePullSecrets": [
    "myacr"
  ],
  "nodeVmSize": "Standard_B2s_v3",
  "nodeVmSizeCount": 2,
  "nodeVmSizeCountTotal": 5,
  "nodeVmSizeTotal": 10,
  "nodeVmSizeType": "Standard_B2s_v3"
}
```

```
enableNodePublicIp": false,
"enableUltraSsd": null,
"gpuInstanceProfile": null,
"hostGroupId": null,
"kubeletConfig": null,
"kubeletDiskType": "OS",
"linuxOsConfig": null,
"maxCount": 5,
"maxPods": 110,
"minCount": 2,
"mode": "System",
"name": "agentpool",
"nodeImageVersion": "AKSUbuntu-2204gen2containerd-202310.31.0",
"nodeLabels": null,
"nodePublicPrefixId": null,
"nodeTaints": null,
"orchestratorVersion": "1.27.7",
"osDiskSizeGb": 128,
"osDiskType": "Managed",
"osSku": "Ubuntu",
"osType": "Linux",
"podSubnetId": null,
"powerState": {
  "code": "Running"
},
},
"provisioningState": "Succeeded",
"proximityPlacementGroupId": null,
"scaleDownMode": null,
"scaleSetEvictionPolicy": null,
"scaleSetPriority": null,
"spotMaxPrice": null,
"tags": null,
"type": "VirtualMachineScaleSets",
"upgradeSettings": null,
"vmSize": "Standard_DS2_v2",
"vnetSubnetId": null,
"workloadRuntime": null
}
],
"apiServerAccessProfile": null,
"autoScalerProfile": {
"balanceSimilarNodeGroups": "false",
"expander": "random",
"maxEmptyBulkDelete": "10",
"maxGracefulTerminationSec": "600",
"maxNodeProvisionTime": "15m",
"maxTotalUnreadyPercentage": "45",
"newPodScaleUpDelay": "0s",
"okTotalUnreadyCount": "3",

```

```
okTotalUnreadyCount": "3",
"scaleDownDelayAfterAdd": "10m",
"scaleDownDelayAfterDelete": "10s",
"scaleDownDelayAfterFailure": "3m",
"scaleDownUnneededTime": "10m",
"scaleDownUnreadyTime": "20m",
"scaleDownUtilizationThreshold": "0.5",
"scanInterval": "10s",
"skipNodesWithLocalStorage": "false",
"skipNodesWithSystemPods": "true"
},
"autoUpgradeProfile": {
"nodeOsUpgradeChannel": "NodeImage",
"upgradeChannel": "patch"
},
"azureMonitorProfile": null,
"azurePortalFqdn": "mykubernetescluster-dns-n5aeqypl.portal.hcp.eastus.azurek8s.io",
"currentKubernetesVersion": "1.27.7",
"disableLocalAccounts": false,
"diskEncryptionSetId": null,
"dnsPrefix": "MyKubernetesCluster-dns",
"enablePodSecurityPolicy": null,
"enableRbac": true,
"extendedLocation": null,
"fqdn": "mykubernetescluster-dns-n5aeqypl.hcp.eastus.azurek8s.io",
"fqdnSubdomain": null,
"httpProxyConfig": null,
"id": "/subscriptions/be98b827-9f6a-40b0-890a-52ceccfa74e6/resourcegroups/AZ500LAB09/providers/Microsoft.ContainerService/managedClusters/MyKubernetesCluster",
"identity": {
"delegatedResources": null,
"principalId": "a32675f4-576f-40f2-8433-d67dd0ef3d99",
"tenantId": "68e2a310-34b0-4970-871c-dbf9d83fef13",
"type": "SystemAssigned",
"userAssignedIdentities": null
},
"identityProfile": {
"kubeletIdentity": {
"clientId": "485bacd7-127f-4eca-b2d0-390ab23bcf6b",
"objectId": "bf2684ae-2f36-4341-aa0a-eaaf04e98347",
"resourceId": "/subscriptions/be98b827-9f6a-40b0-890a-52ceccfa74e6/resourcegroups/MC_AZ500LAB09_MyKubernetesCluster_eastus/providers/Microsoft.ManagedIdentity/userAssignedIdentities/MyKubernetesCluster-agentpool"
}
},
"kubernetesVersion": "1.27.7",
"linuxProfile": null,
"location": "eastus",
"maxAgentPools": 100,
"name": "MyKubernetesCluster",
"networkProfile": {
```



A screenshot of the Microsoft Azure Cloud Shell interface. The title bar says "Microsoft Azure" and "Search resources, services, and docs (G+/-)". The main pane is titled "Bash" and contains a JSON configuration file. The file includes fields like "enabled": true, "snapshotController": {"enabled": true}, "supportPlan": "KubernetesOfficial", "type": "Microsoft.ContainerService/ManagedClusters", and "windowsProfile": {"adminPassword": null, "adminUsername": "azureuser", "enableCsipProxy": true, "gmsaProfile": null, "licenseType": null}. At the bottom of the JSON, there's a line starting with "RG_AKS=AZ500LAB09". The status bar at the bottom shows "aisha [~]\$".

2. 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

```
Microsoft Azure Search resources, services, and docs (G+/-) aisha.khalifan@aws.ajira... EMOBILIS TECHNOLOGY TRAINI...
Bash ? { } x
{
  "type": "Microsoft.Authorization/roleAssignments",
  "updatedBy": "bac46206-6bd8-466d-bb05-278c9c753b94",
  "updatedOn": "2023-11-16T21:37:56.303665+00:00"
}
aisha [ ~ ]$ cat aks.sh
RG_AKS=AZ500LAB09
AKS_VNET_NAME=AZ500LAB09-vnet
AKS_CLUSTER_NAME=MyKubernetesCluster

# Create the resource group (if it doesn't exist)

# Create the virtual network (if it doesn't exist)
az network vnet create \
--resource-group $RG_AKS \
--name $AKS_VNET_NAME

# Get the ID of the virtual network
AKS_VNET_ID=$(az network vnet show --name $AKS_VNET_NAME --resource-group $RG_AKS --query id -o tsv)

# Get the managed identity ID of the AKS cluster
AKS_MANAGED_ID=$(az aks show --name $AKS_CLUSTER_NAME --resource-group $RG_AKS --query identity.principalId -o tsv)

# Assign the "Contributor" role to the AKS managed identity on the virtual network
az role assignment create --assignee $AKS_MANAGED_ID --role "Contributor" --scope $AKS_VNET_ID

aisha [ ~ ]$
```

```
Microsoft Azure Search resources, services, and docs (G+/-) aisha.khalifan@aws.ajira... EMOBILIS TECHNOLOGY TRAINI...
Bash ? { } x
aisha [ ~ ]$ ./aks.sh
{
  "newVNet": {
    "addressSpace": {
      "addressPrefixes": [
        "10.0.0.0/16"
      ]
    },
    "enableDdosProtection": false,
    "etag": "W/\"baf99658-d111-4a86-b771-5f209f0537fe\"",
    "id": "/subscriptions/be98b827-9f6a-40b0-890a-52ceccfa74e6/resourceGroups/AZ500LAB09/providers/Microsoft.Network/virtualNetworks/AZ500LAB09-vnet",
    "location": "eastus",
    "name": "AZ500LAB09-vnet",
    "provisioningState": "Succeeded",
    "resourceGroup": "AZ500LAB09",
    "resourceguid": "051f4f02-79e3-4726-bc3d-0eacb5c24438",
    "subnets": [],
    "type": "Microsoft.Network/virtualNetworks",
    "virtualNetworkPeerings": []
  }
}
{
  "condition": null,
  "conditionVersion": null,
  "createdBy": null,
  "createdOn": "2023-11-16T21:37:55.962660+00:00",
  "delegatedManagedIdentityResourceId": null,
  "description": null,
  "id": "/subscriptions/be98b827-9f6a-40b0-890a-52ceccfa74e6/resourceGroups/AZ500LAB09/providers/Microsoft.Network/virtualNetworks/AZ500LAB09-vnet/providers/Microsoft.Authorization/roleAssignments/73c52bb6-ed3a-468d-8ac2-2ba9eef0bd45",
  "name": "73c52bb6-ed3a-468d-8ac2-2ba9eef0bd45",
  "principalId": "a32675f4-576f-40f2-8433-d67dd0ef3d99",
  "principalType": "ServicePrincipal",
  "resourceGroup": "AZ500LAB09",
  "roleDefinitionId": "/subscriptions/be98b827-9f6a-40b0-890a-52ceccfa74e6/providers/Microsoft.Authorization/roleDefinitions/b24988ac-6180-42a0-ab88-20f7382dd24c",
  "scope": "/subscriptions/be98b827-9f6a-40b0-890a-52ceccfa74e6/resourceGroups/AZ500LAB09/providers/Microsoft.Network/virtualNetworks/AZ500LAB09-vnet",
  "type": "Microsoft.Authorization/roleAssignments",
  "updatedBy": "bac46206-6bd8-466d-bb05-278c9c753b94",
  "updatedOn": "2023-11-16T21:37:56.303665+00:00"
}
aisha [ ~ ]$
```

```
Microsoft Azure Search resources, services, and docs (G+/-) aisha.khalifan@aws.ajira... EMOBILIS TECHNOLOGY TRAINI...
Bash ? { } x
{
  "createdBy": null,
  "createdOn": "2023-11-16T21:37:55.962660+00:00",
  "delegatedManagedIdentityResourceId": null,
  "description": null,
  "id": "/subscriptions/be98b827-9f6a-40b0-890a-52ceccfa74e6/resourceGroups/AZ500LAB09/providers/Microsoft.Network/virtualNetworks/AZ500LAB09-vnet/providers/Microsoft.Authorization/roleAssignments/73c52bb6-ed3a-468d-8ac2-2ba9eef0bd45",
  "name": "73c52bb6-ed3a-468d-8ac2-2ba9eef0bd45",
  "principalId": "a32675f4-576f-40f2-8433-d67dd0ef3d99",
  "principalType": "ServicePrincipal",
  "resourceGroup": "AZ500LAB09",
  "roleDefinitionId": "/subscriptions/be98b827-9f6a-40b0-890a-52ceccfa74e6/providers/Microsoft.Authorization/roleDefinitions/b24988ac-6180-42a0-ab88-20f7382dd24c",
  "scope": "/subscriptions/be98b827-9f6a-40b0-890a-52ceccfa74e6/resourceGroups/AZ500LAB09/providers/Microsoft.Network/virtualNetworks/AZ500LAB09-vnet",
  "type": "Microsoft.Authorization/roleAssignments",
  "updatedBy": "bac46206-6bd8-466d-bb05-278c9c753b94",
  "updatedOn": "2023-11-16T21:37:56.303665+00:00"
}
aisha [ ~ ]$ cat aks.sh
```

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**.

```
aisha [ ~ ]$ cat aks.sh
RG_AKS=AZ500LAB09
AKS_VNET_NAME=AZ500LAB09-vnet
AKS_CLUSTER_NAME=MyKubernetesCluster

# Create the resource group (if it doesn't exist)
az resource group create \
--name RG_AKS \
--location "West US"

# Create the virtual network (if it doesn't exist)
az network vnet create \
--resource-group RG_AKS \
--name AKS_VNET_NAME

# Get the ID of the virtual network
AKS_VNET_ID=$(az network vnet show --name AKS_VNET_NAME --resource-group RG_AKS --query id -o tsv)

# Get the managed identity ID of the AKS cluster
AKS_MANAGED_ID=$(az aks show --name AKS_CLUSTER_NAME --resource-group RG_AKS --query identity.principalId -o tsv)

# Assign the "Contributor" role to the AKS managed identity on the virtual network
az role assignment create --assignee AKS_MANAGED_ID --role "Contributor" --scope AKS_VNET_ID

aisha [ ~ ]$
```

Upload destination: /home/aisha
nginxexternal.yaml COMPLETE

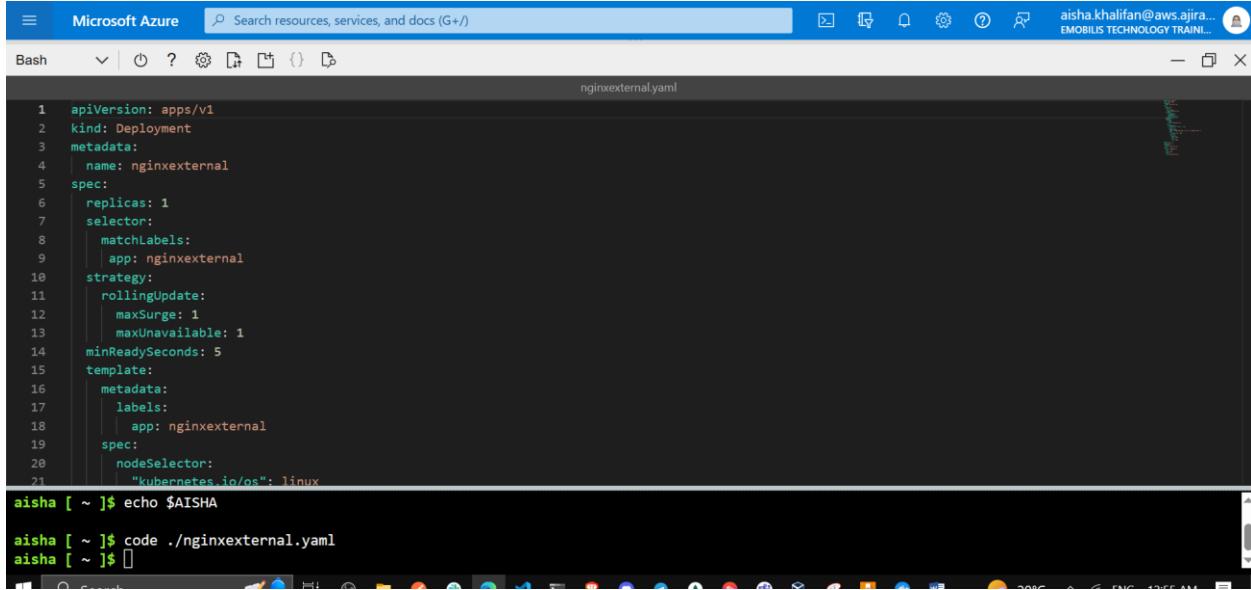
```
aisha [ ~ ]$
```

Upload destination: /home/aisha
nginxinternal.yaml COMPLETE

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.



The screenshot shows the Microsoft Azure Cloud Shell interface. The top bar displays "Microsoft Azure" and a search bar. Below the search bar is a toolbar with icons for copy, paste, refresh, and others. The main area is divided into two panes: a "Bash" pane on the left and an "nginxtutorial.yaml" editor pane on the right. The editor pane contains the following YAML code:

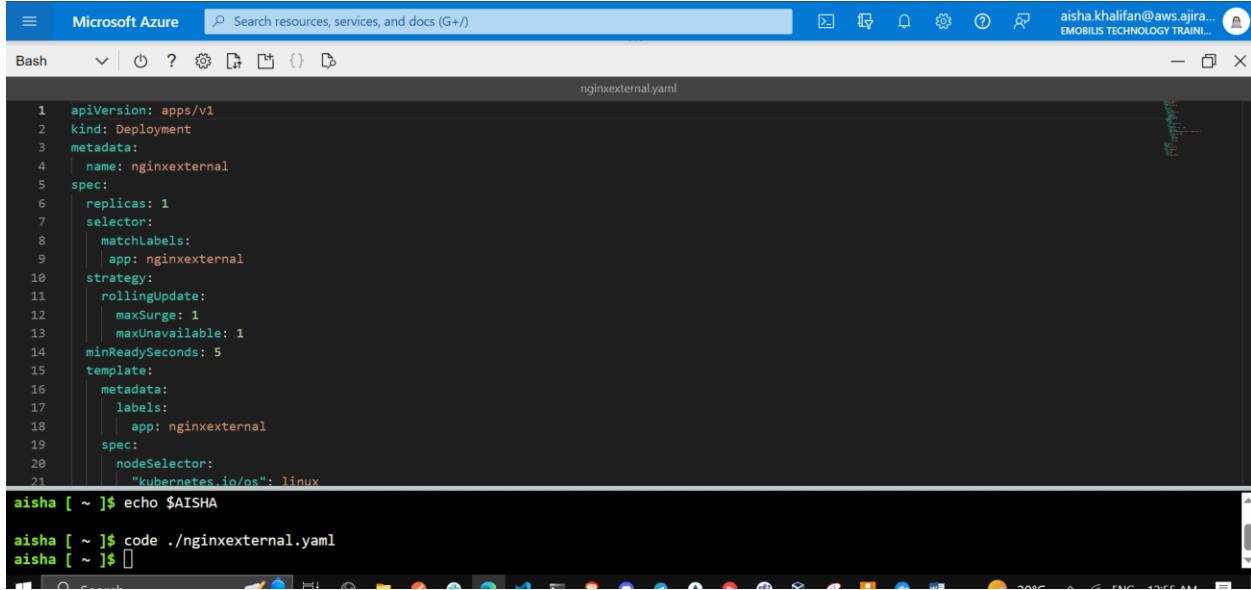
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginxexternal
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginxexternal
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1
    minReadySeconds: 5
  template:
    metadata:
      labels:
        app: nginxexternal
    spec:
      nodeSelector:
        "kubernetes.io/os": linux
```

At the bottom of the Bash pane, there are command history entries: "aisha [~]\$ echo \$AISHA", "aisha [~]\$ code ./nginxexternal.yaml", and "aisha [~]\$ []".

3. 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.



The screenshot shows the Microsoft Azure Cloud Shell interface, identical to the previous one but with a different file name in the editor pane: "nginxtutorial.yaml". The YAML code is the same as in the previous screenshot.

4. In the editor pane, scroll down to **line 24** and replace the <ACRUniqueName> placeholder with the ACR name.

The screenshot shows the Microsoft Azure Cloud Shell interface. In the top right corner of the editor pane, there is an ellipsis icon. Clicking this icon will bring up a context menu with options like 'Save' and 'Close editor'. The Bash session shows the command `kubectl apply -f nginxexternal.yaml` being run, which creates a deployment and a service named 'nginxexternal'.

```
12     maxSurge: 1
13     maxUnavailable: 1
14   minReadySeconds: 5
15   template:
16     metadata:
17       labels:
18         | app: nginxexternal
19     spec:
20       nodeSelector:
21         "kubernetes.io/os": linux
22       containers:
23         - name: nginx
24           image: az500235959017.azurecr.io/sample/nginx:v1
25         ports:
26           - containerPort: 80
27             resources:
28               requests:
29                 cpu: 250m
30               limits:
31                 cpu: 500m
32 ---
```

```
aisha [ ~ ]$ ACRNAME=az500235959017
aisha [ ~ ]$ echo $ACRNAME
az500235959017
aisha [ ~ ]$
```

5. In the editor pane, in the upper right corner, click the **ellipses** icon, click **Save** and then click **Close editor**.
6. In the Bash session within the Cloud Shell pane, run the following to apply the change to the cluster:

kubectl apply -f nginxexternal.yaml

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

7. 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

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

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.

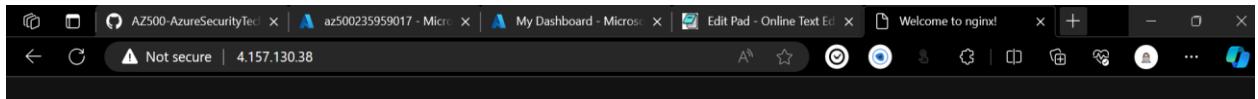
kubectl get service nginxexternal

```
aisha [ ~ ]$ kubectl get service nginxexternal
NAME        TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
nginxexternal  LoadBalancer  10.0.136.238  4.157.130.38  80:31767/TCP  3m52s
aisha [ ~ ]$
```

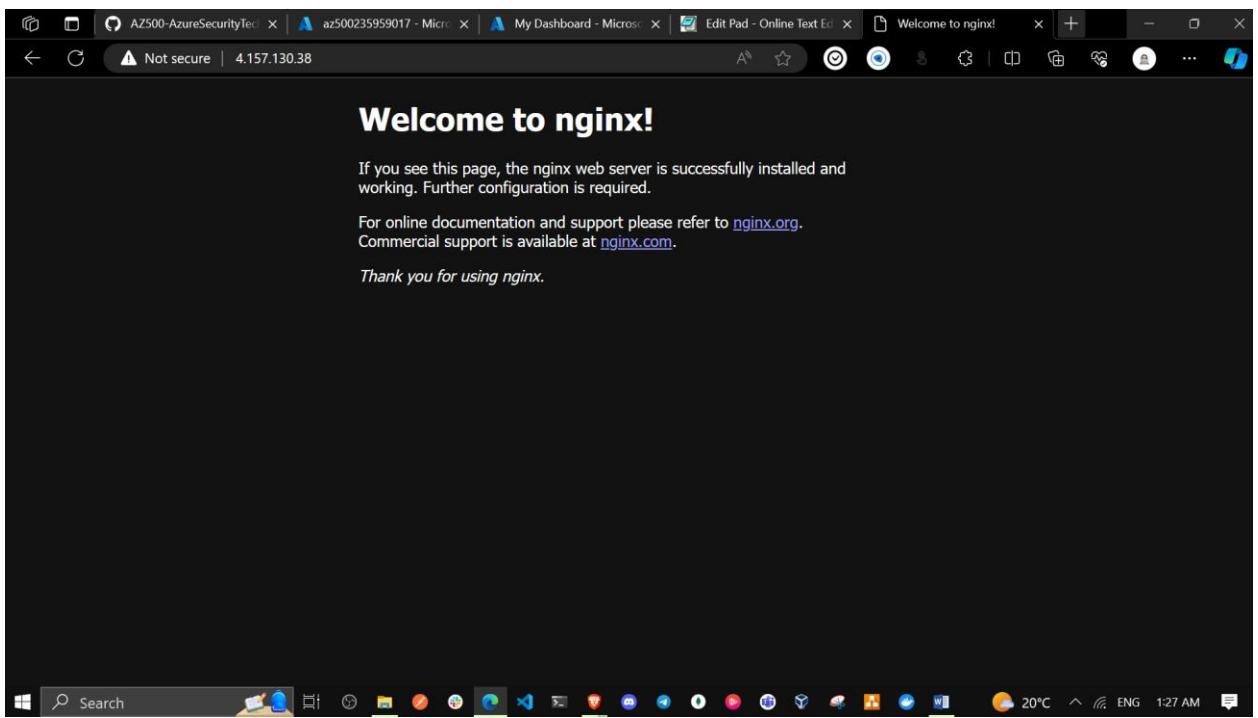
2. In the Bash session within the Cloud Shell pane, review the output and record the value in the External-IP column. You will need it in the next step.

EXTERNAL IP = 4.157.130.38

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.



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 `nginxinternal.yaml` file, so you can edit its content.

code ./nginxinternal.yaml

Note: This is the *internal* yaml file.

The screenshot shows a Microsoft Azure Bash terminal window. The title bar says "Bash". The terminal content includes the contents of the `nginxinternal.yaml` file and the output of the command `kubectl get service nginxexternal`. The output shows a LoadBalancer service named `nginxexternal` with an external IP of `4.157.130.38` and port `80:31767/TCP`.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginxinternal
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginxinternal
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1
    minReadySeconds: 5
  template:
    metadata:
      labels:
        app: nginxinternal
    spec:
      nodeSelector:
        "kubernetes.io/os": linux
aisha [ ~ ]$ kubectl get service nginxexternal
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)      AGE
nginxexternal   LoadBalancer   10.0.136.238  4.157.130.38  80:31767/TCP  3m52s
aisha [ ~ ]$ code ./nginxinternal.yaml
aisha [ ~ ]$ 
```

2. 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.

The screenshot shows a Microsoft Azure Bash terminal window. The title bar says "Bash". The terminal content includes the contents of the `nginxinternal.yaml` file, which has been edited to change the container image reference. A red arrow points to the line `image: <ACRUniqueName>.azurecr.io/sample/nginx:v1`. The terminal also shows the output of the command `kubectl get service nginxexternal`, which remains the same as in the previous screenshot.

```
labels:
  app: nginxinternal
spec:
  nodeSelector:
    "kubernetes.io/os": linux
  containers:
  - name: nginx
    image: <ACRUniqueName>.azurecr.io/sample/nginx:v1
    ports:
    - containerPort: 80
    resources:
      requests:
        cpu: 250m
      limits:
        cpu: 500m
    ...
  apiVersion: v1
  kind: Service
  metadata:
    name: nginxinternal
  annotations:
aisha [ ~ ]$ kubectl get service nginxexternal
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)      AGE
nginxexternal   LoadBalancer   10.0.136.238  4.157.130.38  80:31767/TCP  3m52s
aisha [ ~ ]$ code ./nginxinternal.yaml
aisha [ ~ ]$ 
```

The screenshot shows the Microsoft Azure Cloud Shell interface. In the top navigation bar, the user is identified as 'aisha.khalifan@aws.ajira...' and the environment is 'EMOBILIS TECHNOLOGY TRAINING'. The main pane is a Bash terminal. At the top of the terminal, there is a code editor window titled 'nginxinternal.yaml'. The code in the editor is as follows:

```
17     labels:
18       app: nginxinternal
19   spec:
20     nodeSelector:
21       "kubernetes.io/os": linux
22     containers:
23       - name: nginx
24         image: az500235959017.azurecr.io/sample/nginx:v1
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: nginxinternal
37   annotations:
```

Below the editor, the terminal shows the command 'kubectl get service nginxexternal' followed by its output:

```
aisha [ ~ ]$ kubectl get service nginxexternal
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
nginxexternal   LoadBalancer   10.0.136.238   4.157.130.38   80:31767/TCP   3m52s
```

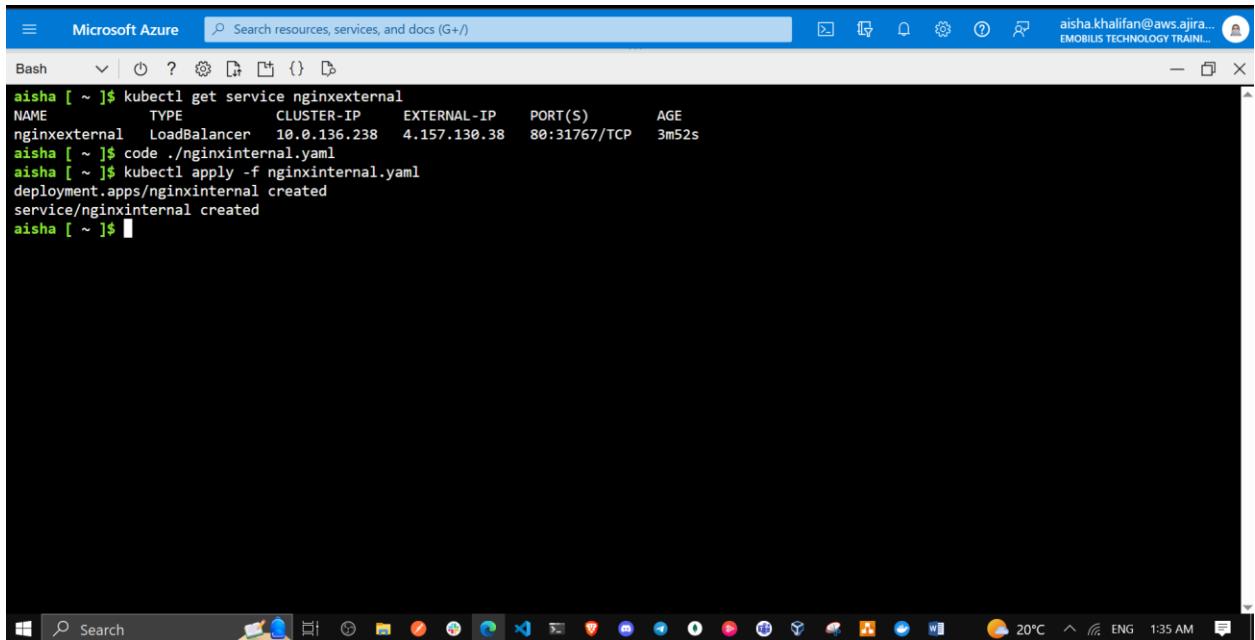
3. In the editor pane, in the upper right corner, click the ellipses icon, click **Save** and then click **Close editor**.

The screenshot shows the Microsoft Azure Cloud Shell interface. The terminal pane displays the same code as the previous screenshot. A red arrow points to the 'Save' option in the context menu that appears when clicking the ellipsis icon in the editor's top right corner. The menu also includes 'Close Editor', 'Open File...', and 'Command Palette...'. The terminal shows the command 'kubectl get service nginxexternal' followed by its output:

```
aisha [ ~ ]$ kubectl get service nginxexternal
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
nginxexternal   LoadBalancer   10.0.136.238   4.157.130.38   80:31767/TCP   3m52s
```

4. In the Bash session within the Cloud Shell pane, run the following to apply the change to the cluster:

kubectl apply -f nginxinternal.yaml



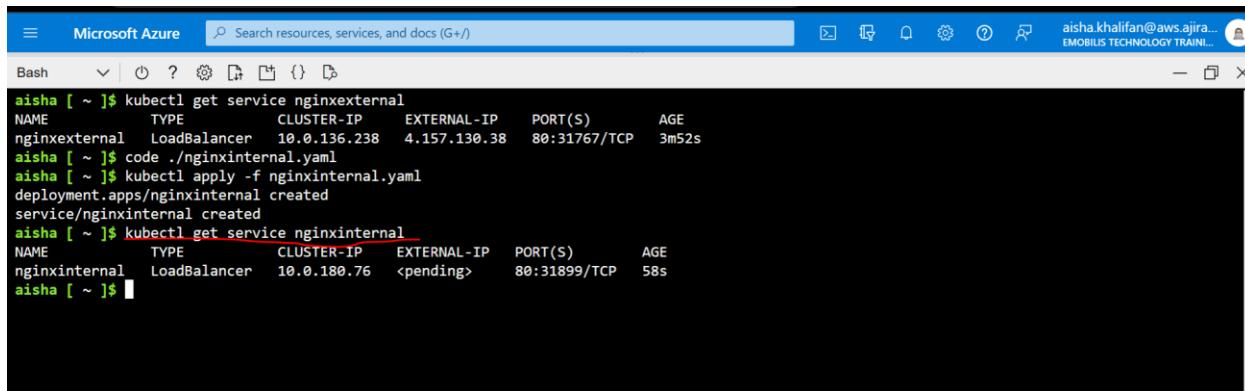
```
aisha [ ~ ]$ kubectl get service nginxexternal
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
nginxexternal   LoadBalancer  10.0.136.238  4.157.130.38  80:31767/TCP  3m52s
aisha [ ~ ]$ code ./nginxinternal.yaml
aisha [ ~ ]$ kubectl apply -f nginxinternal.yaml
deployment.apps/nginxinternal created
service/nginxinternal created
aisha [ ~ ]$
```

5. 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*

6. 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



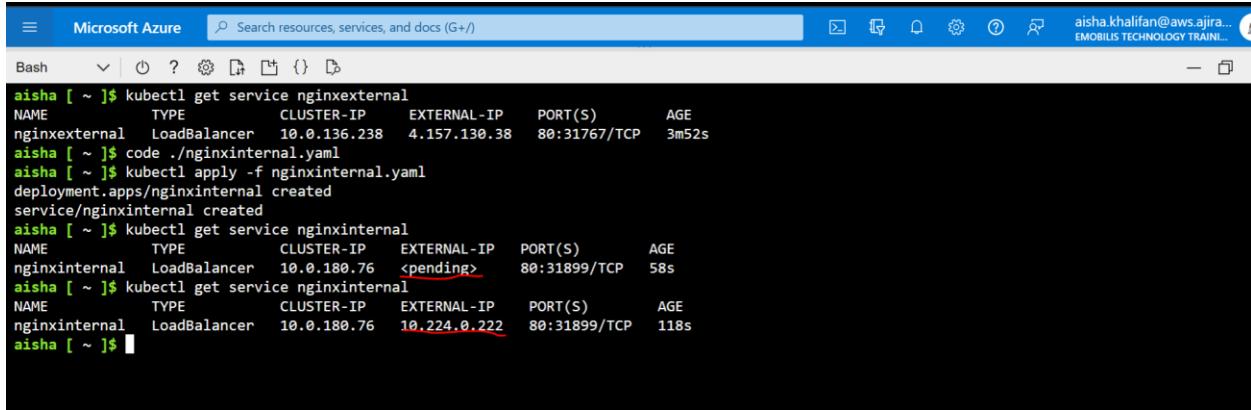
```
aisha [ ~ ]$ kubectl get service nginxexternal
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
nginxexternal   LoadBalancer  10.0.136.238  4.157.130.38  80:31767/TCP  3m52s
aisha [ ~ ]$ code ./nginxinternal.yaml
aisha [ ~ ]$ kubectl apply -f nginxinternal.yaml
deployment.apps/nginxinternal created
service/nginxinternal created
aisha [ ~ ]$ kubectl get service nginxinternal
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
nginxinternal   LoadBalancer  10.0.180.76  <pending>      80:31899/TCP  58s
aisha [ ~ ]$
```

7. 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.



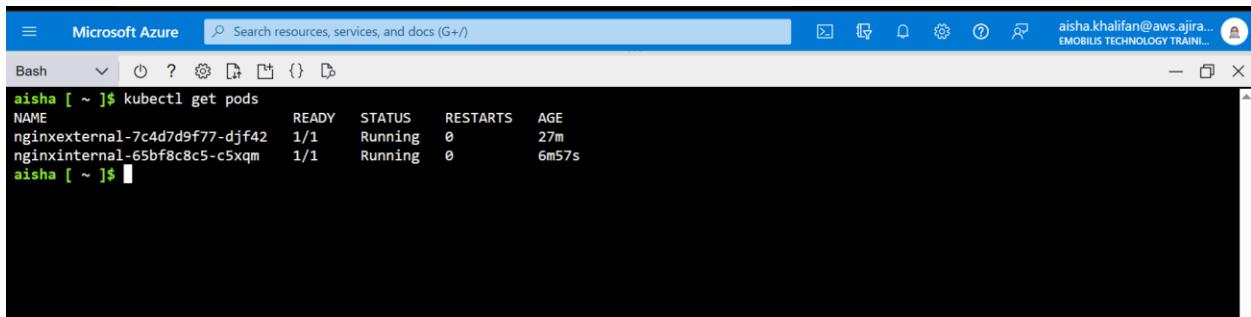
```
aisha [ ~ ]$ kubectl get service nginxexternal
NAME      TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)      AGE
nginxexternal   LoadBalancer  10.0.136.238  4.157.130.38  80:31767/TCP  3m52s
aisha [ ~ ]$ code ./nginxinternal.yaml
aisha [ ~ ]$ kubectl apply -f nginxinternal.yaml
deployment.apps/nginxinternal created
service/nginxinternal created
aisha [ ~ ]$ kubectl get service nginxinternal
NAME      TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)      AGE
nginxinternal   LoadBalancer  10.0.180.76   <pending>    80:31899/TCP  58s
aisha [ ~ ]$ kubectl get service nginxinternal
NAME      TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)      AGE
nginxinternal   LoadBalancer  10.0.180.76   10.224.0.222  80:31899/TCP  118s
aisha [ ~ ]$
```

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



```
aisha [ ~ ]$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
nginxexternal-7c4d7d9f77-djf42  1/1     Running   0          27m
nginxinternal-65bf8c8c5-c5xqm  1/1     Running   0          6m57s
aisha [ ~ ]$
```

2. 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.

nginxexternal-7c4d7d9f77-djf42

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):

kubectl exec -it <pod_name> -- /bin/bash

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):

curl http://<internal_IP>

This was my Internal IP = 4.157.130.38/

The screenshot shows a Microsoft Azure Cloud Shell window. The terminal pane displays a command being run: `aisha [~]$ kubectl exec -it nginxexternal-7c4d7d9f77-djf42 -- /bin/bash`. Below this, the command `root@nginxexternal-7c4d7d9f77-djf42:/# curl http://4.157.130.38/` is executed. The response is a standard Nginx welcome page with the text "Welcome to nginx!" and configuration details. A red bracket highlights the URL in the curl command, and the word "Successful" is handwritten in red above the page content.

5. Close the Cloud Shell pane.

Result: You have configured and secured ACR and AKS.

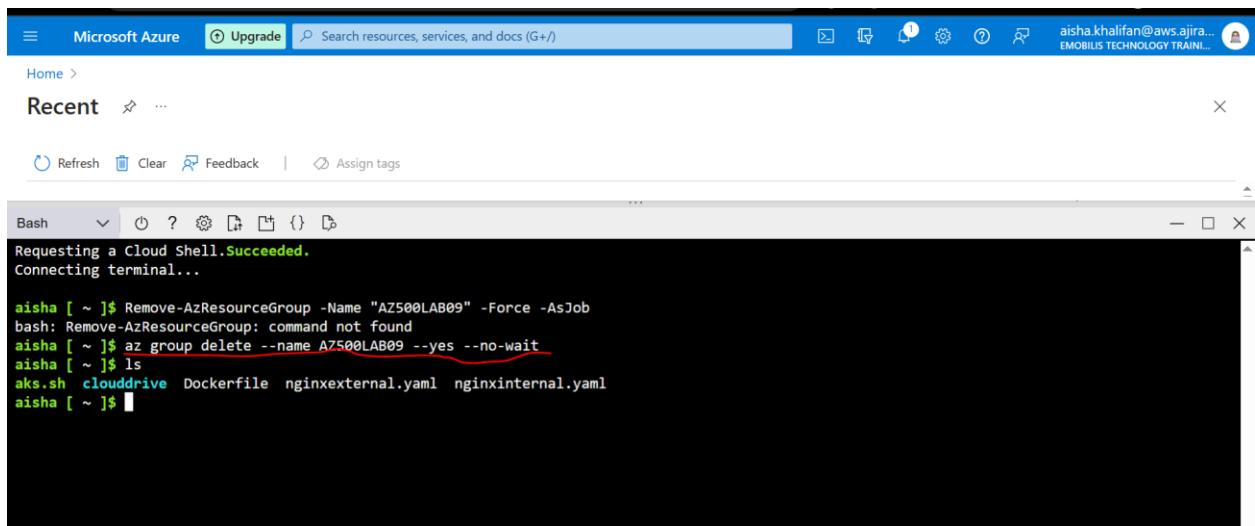
Clean up resources

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



The screenshot shows the Microsoft Azure portal with the Cloud Shell pane open. The terminal window is titled 'Bash' and displays the following command history:

```
Requesting a Cloud Shell.Succeeded.
Connecting terminal...
aisha [ ~ ]$ Remove-AzResourceGroup -Name "AZ500LAB09" -Force -AsJob
bash: Remove-AzResourceGroup: command not found
aisha [ ~ ]$ az group delete --name AZ500LAB09 --yes --no-wait
aisha [ ~ ]$ ls
aks.sh  clouddrive  Dockerfile  nginxexternal.yaml  nginxinternal.yaml
aisha [ ~ ]$
```

4. Close the **Cloud Shell** pane.

Conclusion

In the lab, I had a hands-on journey with Azure tech. Learned to make apps work in containers using Docker. Azure Container Registry helped keep the app stuff safe and organized. Then, we jumped into Azure Kubernetes Service, setting up a smart system for our apps that can grow easily. Made sure apps were safe to use, whether we were working together or letting others see them. This all happened in the East US part of Azure, giving me a cool experience with putting apps on the cloud.