**Lesson 12 Demo 1**

**Azure Kubernetes Service**

**Objective:** Contoso has several multitier applications that are not suitable to run by using Azure Container instances. To determine whether they can be run as containerized workloads, you want to evaluate using Kubernetes as the container orchestrator. To further minimize management overhead, you want to test Azure Kubernetes Service, including its simplified deployment experience and scaling capabilities.

**Tools required:** Azure account with administrator access

**Prerequisites:** None

**Steps to be followed:**

1. Registering the Microsoft Kubernetes resource providers
2. Deploying an Azure Kubernetes Service cluster
3. Deploying pods into the Azure Kubernetes Service cluster
4. Scaling containerized workloads in the Azure Kubernetes Service cluster

**Step 1: Registering the Microsoft Kubernetes resource providers**

1. Sign in to the Azure portal <https://portal.azure.com/>
2. Open the Cloud Shell by clicking the first icon in the top right of the Azure Portal. If prompted, select **PowerShell,** and then select **Create storage**Graphical user interface, application

   Description automatically generatedGraphical user interface, text

   Description automatically generated

Graphical user interface, text, application

Description automatically generated

1. In the PowerShell session within the Cloud Shell pane, run the following to register the Microsoft.Kubernetes and Microsoft.KubernetesConfiguration resource providers:

**Register-AzResourceProvider -ProviderNamespace Microsoft.Kubernetes**

**Register-AzResourceProvider –ProviderNamespace Microsoft.KubernetesConfiguration**

Text, timeline

Description automatically generated

**Step 2: Deploying an Azure Kubernetes Service cluster**

1. Type **Kubernetes services** in the search bar and select it

Graphical user interface, text, application, email

Description automatically generated

1. On the **Kubernetes services** blade, click **+ Create,** and then click on **+ Create a Kubernetes cluster**

Graphical user interface, text, application, email

Description automatically generated

1. On the **Basics** tab, specify the following settings (leave others with their default values):

**Subscription:** the name of the Azure subscription you are using in this lab

**Resource group:** (New resource group) az305-rg1

**Name:** az305-aks1

**Region:** (US) East US

**Availability zones:** Zone 1, 2, 3

**Scaling method:** Manual

**Node count:** 1

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

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

**Authentication method:** System-assigned managed identity (default)

**Role-based access control (RBAC):** Enabled

1. Go to the Integrations tab, select **create new** for **Container Registry**, provide registry name, enable **admin user** and then click Ok

Graphical user interface, text, application, email

Description automatically generated

1. Set Container monitoring to **Disabled**, click **Review + create**, ensure that the validation passed, and click **Create**

Graphical user interface, text, application, email

Description automatically generated

**Step 3: Deploying pods into the Azure Kubernetes Service cluster**

1. Click on **Go to resource**

Graphical user interface, text, application, email

Description automatically generated

1. In the Settings section, click **Node pools**
2. Verify that the cluster consists of a single pool with node

Graphical user interface, text, application

Description automatically generated

1. Now, navigate back to **Azure Cloud Shell** by clicking on the icon in the top right of the Azure Portal

Graphical user interface, application

Description automatically generated

1. Switch the Azure Cloud Shell to Bash (black background)

Graphical user interface, text

Description automatically generated

1. Run the following command to retrieve the credentials to access the AKS cluster:

**RESOURCE\_GROUP='az305-rg1'**

**AKS\_CLUSTER='az305-aks1'**

**az aks get-credentials --resource-group $RESOURCE\_GROUP --name $AKS\_CLUSTER**

Text

Description automatically generated

1. Run the following to verify connectivity to the AKS cluster:

**kubectl get nodes**

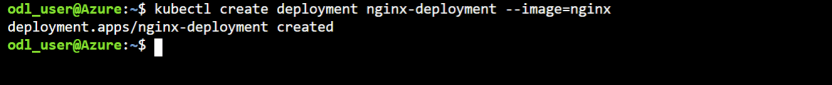
1. Review the output and verify that the one node that the cluster consists of at this point is reporting the **Ready** status

A picture containing text

Description automatically generated

1. Run the following command to deploy the nginx image from the Docker Hub:

**kubectl create deployment nginx-deployment --image=nginx**



1. Run the following to verify that a Kubernetes pod has been created:

**kubectl get pods**

Text

Description automatically generated

1. Run the following to identify the state of the deployment:

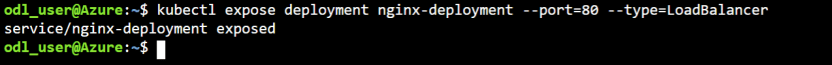
**kubectl get deployment**

Text

Description automatically generated

1. Run the following to make the pod available from the internet:

**kubectl expose deployment nginx-deployment --port=80 --type=LoadBalancer**



1. Run the following to identify whether a public IP address has been provisioned:

**kubectl get service**

Graphical user interface, application

Description automatically generated

Note: Rerun the command until the value in the **EXTERNAL-IP column** for the **nginx-deployment** entry changes from **<pending>** to a **public IP address**. Note the public IP address in the EXTERNAL-IP column for nginx-deployment.

1. Open a browser window and navigate to the IP address you obtained in the previous step. Verify that the browser page displays the “**Welcome to nginx!”** message

Graphical user interface, text, application

Description automatically generated

**Step 4: Scaling containerized workloads in the Azure Kubernetes service cluster**

1. Navigate back to the Cloud Shell pane and run the following to scale the deployment by increasing the number of pods to 2:

**RESOURCE\_GROUP='az305-rg1'**

**AKS\_CLUSTER='az305-aks1'**

**kubectl scale --replicas=2 deployment/nginx-deployment**

Text

Description automatically generated

1. Run the following to verify the outcome of scaling the deployment:

**kubectl get pods**

Text

Description automatically generated

1. Run the following to scale out the cluster by increasing the number of nodes to 2:

**az aks scale --resource-group $RESOURCE\_GROUP --name $AKS\_CLUSTER --node-count 2**

Text

Description automatically generated

Text

Description automatically generated

1. Run the following to scale the deployment:

**kubectl scale --replicas=10 deployment/nginx-deployment**



1. Run the following to verify the outcome of scaling the deployment:

**kubectl get pods**

Graphical user interface, text

Description automatically generated

Note: Review the output of the command and verify that the number of pods has increased to 10.

1. Run the following to review the pods’ distribution across cluster nodes:

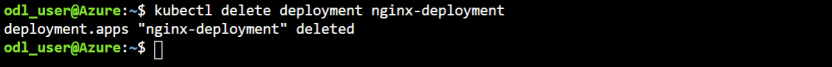
**kubectl get pod -o=custom-columns=NODE:.spec.nodeName,POD:.metadata.name**

Graphical user interface, text

Description automatically generated

1. Run the following to delete the deployment:

**kubectl delete deployment nginx-deployment**



You have successfully deployed an Azure Kubernetes Service cluster, deployed pods into the Azure Kubernetes Service cluster, and scaled containerized workloads in it.