**Lesson 8 Demo 3**

**Connecting AKS cluster from Windows PowerShell**

**Objectives:** To connect an AKS cluster from Windows PowerShell

**Tools required:** Azure Kubernetes Service, Storage Account, and File Share

**Prerequisites:** Follow Lesson 8 Demo 1 to set up an AKS cluster

**Steps to be followed:**

1. Setting up a storage account resource
2. Creating a new file share for the storage account
3. Setting up Azure Cloud Shell
4. Creating a deployment and rolling out an update for it

**Step 1: Setting up a storage account resource**

1. On create a resource page, search for the **storage account,** and select the **storage account** resource from the dropdown.  
     
   Graphical user interface, application

   Description automatically generated
2. On the storage account page, click on the **create** button to create this resource.  
     
   Graphical user interface, application

   Description automatically generated
3. On create a storage account page, enter the following details,and click on the **review + create** button.

**Storage account name:** **sl20storageaccount**

Graphical user interface, text, application

Description automatically generated**Region: West US**

|  |
| --- |
| **Note:** Keep the default value for all the other fields. |

Graphical user interface, text, application, email

Description automatically generated  
1.4 Once the validation is complete, click on the **create** button.

1. A screenshot of a computer

   Description automatically generatedCheck the newly created resource on the **resource group** page.

**Step 2: Creating a new file share for the storage account**

1. Graphical user interface, text, application, email

   Description automatically generatedIn the storage account resource, go to **file shares** and click on the **file share** button.
2. In the **new** **file share** pop-up window, enter the following details and click on the **create** button.

**Name: slfileshare  
  
Graphical user interface, text, application, email

Description automatically generated**

1. Check the newly created **slfileshare** under the **file shares** section of the storage account.  
     
   Graphical user interface, text, application, email

   Description automatically generated

**Step 3: Setting up Azure Cloud Shell**

1. Graphical user interface, text, application, email

   Description automatically generatedClick on the **cloud shell icon** from the top navigation bar.
2. Graphical user interface, text, application, email

   Description automatically generatedOn the Azure cloud shell window, click on the **bash** link.
3. Graphical user interface, text, application, email

   Description automatically generatedOn the next screen, click on the **show advanced settings** link.

|  |
| --- |
| **Note:** Make sure the **cloud shell region** is **West US** as all resources are created for the West US region. |

1. Graphical user interface, application

   Description automatically generatedOn the **advanced settings** window, select the **use existing** option for the **storage account** and enter **sl20storageaccount** as the storage account name.
2. For the **file share** field, select the **use existing** option, enter **slfileshare** as the file share name, and click on the **attach storage** button.

Graphical user interface, application

Description automatically generated

1. Go to the **overview** section in the **SL-Cluster** and click on the **connect** button.  
     
   Graphical user interface, text, application

   Description automatically generated

1. Graphical user interface, application

   Description automatically generatedCopy the commands from **connect to SL-Cluster** pop-up window.
2. Graphical user interface, text, application

   Description automatically generatedNavigate back to the **bash** window and paste the commands copied in **Step 3.7.**

**Step 4: Creating a deployment and rolling out an update for it**

1. A screenshot of a computer

   Description automatically generatedNavigate to the **workloads** section in **SL-Cluster**, select the **Deployments** tab, and click on the **create** button.
2. Add the following code to the **YAML** section and click on the **add** button:

***apiVersion: apps/v1***

***kind: Deployment***

***metadata:***

***name: second-deployment***

***namespace: first-namespace***

***labels:***

***app: second-deployment***

***spec:***

***replicas: 3***

***selector:***

***matchLabels:***

***app: second-deployment***

***template:***

***metadata:***

***labels:***

***app: second-deployment***

***spec:***

***containers:***

***- name: nginx-container***

***image: nginx:1.14.2***

***ports:***

***- containerPort: 80  
  
Graphical user interface, text, application, email

Description automatically generated***

1. Go to the **bash** screen and run the following commands to check the newly created second-deployment:

***kubectl get deployments -n first-namespace***

***kubectl get rs -n first-namespace***

***kubectl get pods --show-labels -n first-namespace  
  
Text

Description automatically generated***

1. Use the following command to update the **Nginx image version** to **1.20.2** in the second-deployment:

***kubectl set image deployment/second-deployment \***

***-n first-namespace nginx-container=nginx:1.20.2 –record  
  
Text

Description automatically generated with medium confidence***

1. Check the rollout status of the second-deployment:

***kubectl rollout status deployment/second-deployment -n first-namespace  
  
Text

Description automatically generated***

1. Execute the following command to describe the updated second-deployment:

***kubectl describe deployments second-deployment -n first-namespace  
  
Text

Description automatically generated***