# Lab Answer Key: Module 7: Implementing containers in Azure

# Lab A: Implementing containers on Azure VMs

## Exercise 1: Implementing containers on Azure VMs

#### Task 1: Connect to an Azure VM running Linux

1. Ensure that you are signed in to MIA-CL1 as **Student** with the password **Pa55w.rd**, and that the **Add-20533EEnvironment** script successfully completed. The script provisions a Linux Azure VM Docker host.
2. Start Microsoft Edge and browse to [**https://portal.azure.com**](https://portal.azure.com). When prompted, sign in by using the Microsoft account that is the Service Administrator of your Azure subscription.
3. In the Azure portal, in the Microsoft Edge window, click the **Cloud shell** icon.

**Note:** You can open **Cloud shell** in a new window for ease of use via [**https://shell.azure.com**](https://shell.azure.com)

1. If you are presented with the **Welcome to Azure Cloud Shell** pane, click **Bash (Linux)**.
2. If you are presented with the **You have no storage mounted** message, click **Show advanced settings**.
3. In the resulting pane, specify the following settings and click **Create storage**:

* Subsciption: the same Azure subscription you chose when running the provisioning script at the beginning of this module
* Cloud Shell region: the same Azure region you chose when running the provisioning script at the beginning of this module
* Resource group: click **Use existing** and ensure that **20533E0701-LabRG** resource group name appers in the text box below
* Storage account: ensure that **Create new** is selected and type a unique name of betwen 3 and 24 characters consisting of lower case letters and digits
* File share: ensure that **Create new** is selected and type a unique name of betwen 3 and 63 characters consisting of lower case letters, digits and dashes.

1. To identify the fully qualified domain name of the target Azure VM Docker host, at the bash prompt, in the Cloud Shell pane, type the following, and then press Enter:

FQDN=$(az vm show --resource-group 20533E0701-LabRG --name 20533E0701-vm0 --show-details --query [fqdns] --output tsv)

1. To display the value of the FQDN variable, at the Cloud Shell prompt, type the following, and then press Enter (take a note of this value since you will need it in the next task):

echo $FQDN

1. To establish an SSH session to the Azure VM, at the Cloud Shell prompt, type the following, and then press Enter:

ssh student@$FQDN

1. The Cloud Shell will display a message informing you that the authenticity of the remote host cannot be established. This is expected. Type **yes** and press Enter to continue connecting.
2. When prompted for the password, type **Pa55w.rd1234** and then press Enter. You should be presented with the **student@20533E0701-vm0** prompt.

#### Task 2: Install Docker and deploy a container to an Azure VM

1. In the Cloud Shell pane, within the SSH session to the Azure VM running Docker, type the following, and then press Enter (when prompted whether you want to continue, type **y** and then press Enter):

sudo apt install docker.io

1. In the Cloud Shell pane, within the SSH session to the Azure VM running Docker, type the following, and then press Enter:

sudo docker run -d -p 80:80 --restart=always nginx

1. Monitor the progress of the container deployment. To verify the successful outcome, at the Cloud Shell prompt, type the following, and then press Enter:

sudo docker ps

1. On MIA-CL1, start Microsoft Edge and browse to the URL matching the DNS fully qualified domain name you obtained in the previous task. Verify that Microsoft Edge displays the **Welcome to nginx!** page

**Result**: After you complete this exercise, you should have successfully run a sample containerized web server nginx on the Docker host Azure VM.

## Exercise 2: Deploying multi-container applications with Docker Compose to Azure VMs

#### Task 1: Install docker-compose and create a compose file

1. In the Cloud Shell pane, within the SSH session to the Azure VM running Docker, to install Docker Compose on the target Docker Azure VM, type the following, and then press Enter (when prompted whether you want to continue, type **y** and then press Enter):

sudo apt install docker-compose

1. To create a **docker-compose.yml** file, type the following, and then press Enter:

nano docker-compose.yml

1. In the nano editor interface, type the following content (*alternatively, you can copy and paste the content of the* ***docker-compose.yml*** *file residing in the* ***F:\Labfiles\Lab07\Solution*** *folder*):

version: "2"  
 services:  
 wordpress:  
 image: wordpress  
 links:  
 - db:mysql  
 ports:  
 - 8080:80  
 db:  
 image: mariadb  
 environment:  
 MYSQL\_ROOT\_PASSWORD: Pa55w.rd

**Note**: Be careful when typing the text above. Make sure you include the spaces to the left of the text (each indent should be a multiple of 2 spaces).

1. Once you typed in the text, press the **Ctrl+O** key combination and then press Enter.
2. Next, press the **Ctrl+X** key combination to exit the nano editor.

#### Task 2: Deploy the containers with docker-compose to an Azure VM

1. In the Cloud Shell pane, within the SSH session to the Azure VM running Docker, to deploy the multi-container application defined in the docker-compose.yml file, type the following, and then press Enter:

sudo docker-compose up -d

1. Monitor the progress of the container deployment. To verify the successful outcome, type the following, and then press Enter:

sudo docker ps

1. On MIA-CL1, start Microsoft Edge and browse to the port 8080 on the target host using the same URL you used in the previous exercise. Verify that Microsoft Edge displays the initial Wordpress setup page.

**Note**: This is possible because the template used to provision the deployment of the Linux Azure VM Docker host included a network security group with a rule that allows inbound traffic on TCP port 8080. If that was not the case, you would need to add this rule.

1. Type exit to terminate the SSH session and then close the Cloud Shell pane.

**Result**: After you completed this exercise, you should have successfully implemented a multi-container application by using Docker Compose.

## Exercise 3: Implementing Azure Container Registry

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

1. On MIA-CL1, in the Azure portal in the Microsoft Edge window, click **+ Create resource**.
2. On the **New** blade, click **Containers**, and then click **Container Registry**.
3. On the **Create container registry** blade, specify the following settings and click **Create**:

* Registry name: a unique name consisting of between 5 and 50 alphanumeric characters
* Subscription: the name of the Azure subscription you are using in this lab
* Resource group: click **Create new** and, in the text box below, type **20533E0702-LabRG**
* Location : the same Azure region that you chose when running the provisioning script at the beginning of this module
* Admin user: **Enable** (this will allow you to use the registry name as username and admin user access key as its password to connect from the Docker host to the registry by using the **docker login** command)
* SKU : **Basic**

1. Wait for the operation to complete.

#### Task 2: Identify Azure Container Registry authentication settings.

1. In the Azure portal, in the Microsoft Edge window, click the **Cloud Shell** icon.
2. To identify the name of the newly created Container Registry instance, in the Cloud Shell pane, at the bash prompt, type the following, and then press Enter:

REGISTRY=$(az resource list --resource-group 20533E0702-LabRG --query '[].name' --output tsv)

1. To display the value of the REGISTRY variable, type the following, and then press Enter (take a note of this value since you will need it in the next task):

echo $REGISTRY

1. To identify the value of the **Login server** of the registry, type the following, and then press Enter (take a note of this value since you will need it in the next task):

az acr show -n $REGISTRY --query loginServer

1. To identify the value of one of the passwords that is necessary to access the registry, type the following, and then press Enter (take a note of this value since you will need it in the next task):

az acr credential show -n $REGISTRY --query passwords[0].value

#### Task 3: Push an image to Azure Container Registry.

1. In the Cloud Shell pane, at the bash prompt, set again the FQDN variable again the fully qualified domain name of the Azure VM Docker host by running the following:

FQDN=$(az vm show --resource-group 20533E0701-LabRG --name 20533E0701-vm0 --show-details --query [fqdns] --output tsv)

1. To establish an SSH session to the Azure VM Docker host, type the following, and then press Enter:

ssh student@$FQDN

1. When prompted for the password, type **Pa55w.rd1234** and then press Enter. You should be presented with the **student@20533E0701-vm0** prompt.
2. Within the SSH session to the Azure VM Docker host, to log in to the Azure Container registry you created in the first task, type the following, replacing the ***<user name>*** , ***<password>*** , and ***<login-server>*** entries with the values you identified in the previous task, and then press Enter:

sudo docker login --username <user name> --password <password> <login-server>

1. Ensure that you receive the **Login succeeded** message. Next to pull an existing image from Docker Hub, type the following, and then press Enter:

sudo docker pull microsoft/aci-helloworld

1. Wait for the image to be downloaded to the Docker Azure VM. Next, to tag the image with the Azure Container registry name, type the following, replacing the ***<login-server>*** entry with the value you identified in the previous task, and then press Enter:

sudo docker tag microsoft/aci-helloworld <login-server>/aci-helloworld:v1

1. To push the tagged image to the Azure Container registry, type the following, replacing the ***<login-server>*** entry with the value you identified in the previous task, and then press Enter:

sudo docker push <login-server>/aci-helloworld:v1

1. Wait for the image to be pushed to the registry. Next, to view the images stored in the Azure Container registry name, in the Azure portal, in the Microsoft Edge window, on the container registry blade, click **Repositories** and note that the list includes the **aci-helloworld** repository.

#### Task 4: Download and deploy an image from the Azure Container Registry

1. In the Cloud Shell pane, within the SSH session to the Docker host, to pull an image from the Azure Container registry, type the following, replacing the ***<login-server>*** entry with the values you identified earlier in this exercise, and then press Enter:

sudo docker pull <login-server>/aci-helloworld:v1

1. Note that, in this case, the image does not need to be downloaded, since it is already present on the target Azure VM Docker host.
2. Next, to deploy the image downloaded from the Azure Container registry, type the following, replacing the ***<login-server>*** entry with the values you identified earlier in this exercise, and then press Enter:

sudo docker run -d --restart=always -p 8081:80 <login-server>/aci-helloworld:v1

1. To verify that the image has been successfully deployed, type the following, and then press Enter:

sudo docker ps

1. Note that the output includes the ***<login-server>*/aci-helloworld:v1** image, where ***<login-server>*** represents the Container Registry instance you provisioned.
2. Type exit to terminate the SSH session and then close the Cloud Shell pane.

#### Task 5: Connect to a container running on a Docker host

1. In order to connect to the newly provisioned container, you will need to modify network security group associated with the network interface of the Linux Azure VM Docker host. On MIA-CL1, in the Azure portal window, in the hub menu, click **Resource groups**.
2. On the **Resource groups** blade, click **20533E0701-LabRG**.
3. On the **20533E0701-LabRG** blade, in the list of resources, click **20533E0701-web-nsg**.
4. On the **20533E0701-web-nsg** blade, click **Inbound security rules**.
5. Click **+ Add**.
6. On the **Add inbound security rule** blade, ensure that **Basic** appears in the toolbar, specify the following settings and click **OK**:

* Source: **Any**
* Source port ranges: **\***
* Destination: **Any**
* Destination port ranges: **8081**
* Protocol: **TCP**
* Action: **Allow**
* Priority: **1300**
* Name: **Port8081-TcpAllowAny**

1. Wait for the operation to complete.
2. On MIA-CL1, start Microsoft Edge and browse to the port 8081 on the target host using the same URL you used in the previous exercise. Verify that Microsoft Edge displays the **Welcome to Azure Container Instances!** page.

# Lab B: Implementing Azure Container Service (AKS)

## Exercise 1: Creating an AKS cluster

#### Task 1: Create an AKS cluster

1. Ensure that you are signed in to MIA-CL1 as **Student** with the password **Pa55w.rd**
2. In the Azure portal, in the Microsoft Edge window, click the **Cloud shell** icon.
3. In the Cloud Shell pane, at the bash prompt, type the following command (replacing ***<Azure region>*** with the name of the Azure region that you chose when running the provisioning script at the beginning of this module), and then press Enter:

az group create --name 20533E0703-LabRG --location <Azure region>

1. In the Cloud Shell pane, at the bash prompt, type the following command (replacing ***<VM size>*** with the VM size you chose when running the provisioning script at the beginning of this module) and then press Enter:

az aks create --resource-group 20533E0703-LabRG --name 20533E0703-k8scluster --node-count 1 --node-vm-size <VM size> --generate-ssh-keys

1. Wait for the deployment to complete.

#### Task 2: Connect to the AKS cluster.

1. To download and configure the credentials to access the AKS cluster, in the Cloud Shell pane, at the bash prompt, type the following command and then press Enter:

az aks get-credentials --resource-group 20533E0703-LabRG --name 20533E0703-k8scluster

1. To verify connectivity to the AKS cluster, type the following command and then press Enter:

kubectl get nodes

1. Review the output and verify that the agent node is reporting the **Ready** status. Rerun the command until the correct status is shown.

**Result**: After you complete this exercise, you should have successfully deployed a new AKS cluster.

## Exercise 2: Managing an AKS cluster

#### Task 1: Deploy a containerized application to the AKS cluster

1. In the Microsoft Edge window, in the Azure portal, in the Cloud Shell pane, at the bash prompt, in order to deploy the **nginx** container from the Docker Hub, type the following command and then press Enter:

kubectl run nginx-20533e0703 --image=nginx --replicas=1 --port=80

**Note**: Make sure to use lower case letters when typing the name of the deployment.

1. To verify that a Kubernetes pod has been created, in the Cloud shell pane, at the bash prompt, type the following command and then press Enter:

kubectl get pods

1. To identify the state of the deployment, type the following command and then press Enter:

kubectl get deployment

1. To make the pod available from Internet, type the following command and then press Enter:

kubectl expose deployment nginx-20533e0703 --port=80 --type=LoadBalancer

**Note**: Make sure to use lower case letters when typing the name of the deployment.

1. To identify whether the public IP address has been provisioned, type the following command and then press Enter:

kubectl get services

1. Repeat step 5 until the value in the **EXTERNAL-IP** column for nginx-20533E0703 changes from **<pending>** to a public IP address. Note the public IP address in the **EXTERNAL-IP** column for nginx-20533E0703.
2. Start Microsoft Edge and browse to the IP address you obtained in the previous step. Verify that Microsoft Edge displays the **Welcome to nginx!**

#### Task 2: Manage deployment of a containerized application on the AKS cluster

1. To scale the deployment, in the Cloud shell pane, at the bash prompt, type the following command and then press Enter:

kubectl scale --replicas=2 deployment/nginx-20533e0703

1. To verify the outcome of scaling the deployment, type the following command and then press Enter:

kubectl get pods

1. In the output of the command you ran in the previous step, verify that the number of pods increased to 2.
2. To delete the deployment, in the Cloud shell pane, at the bash prompt, type the following command and then press Enter:

kubectl delete deployment nginx-20533e0703

1. To verify that the command you ran in the previous step completed successfully, type the following command and then press Enter:

kubectl get deployment

#### Task 3: Remove the lab environment

1. On MIA-CL1, close all open windows without saving any files.
2. On the taskbar, right-click the **Windows PowerShell** icon, and then click **Run as Administrator**. In the User Account Control dialog box, click **Yes**.
3. Type the following command, and then press **Enter**:

Remove-20533EEnvironment

1. When prompted, sign in by using the Microsoft account that is the Service Administrator of your Azure subscription.
2. If you have multiple Azure subscriptions, select the one you want the script to target.
3. If prompted, specify the current lab number.
4. When prompted for confirmation, type **y**.
5. Start Microsoft Edge, browse to the Azure portal at [**https://portal.azure.com**](https://portal.azure.com), and sign in by using the Microsoft account that is the Service Administrator of your Azure subscription.
6. In the Azure portal, on Dashboard, click **Edit**.
7. Right-click unoccupied area of the dashboard and, in the right-click menu, click **Reset to default state**. When prompted to confirm, click **Yes**.
8. Click **Done customizing**.
9. Close all open windows.

**Result**: After you complete this exercise, you should have successfully deployed a containerized workload to the new AKS cluster and validated its availability.

©2016 Microsoft Corporation. All rights reserved.

The text in this document is available under the [Creative Commons Attribution 3.0 License](https://creativecommons.org/licenses/by/3.0/legalcode), additional terms may apply. All other content contained in this document (including, without limitation, trademarks, logos, images, etc.) are **not** included within the Creative Commons license grant. This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

This document is provided "as-is." Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it. Some examples are for illustration only and are fictitious. No real association is intended or inferred. Microsoft makes no warranties, express or implied, with respect to the information provided here.