# Module 7: Implementing containers in Azure

# Lab A: Implementing containers on Azure VMs

### Scenario

Adatum Corporation plans to implement some of its applications as Docker containers on Azure VMs. To optimize this implementation, you intend to combine multiple containers by using Docker Compose. A. Datum would also like to deploy its own private Docker registry in Azure to store containerized images. Your task is to test the functionality of tools that facilitate deployment of Docker hosts and Docker containers. You also need to evaluate Azure Container Registry.

### Objectives

After completing this lab, you will be able to:

* Deploy containers to Azure VMs
* Deploy multicontainer applications with Docker Compose to Azure VMs
* Implement Azure Container Registry

### Lab Setup

Estimated Time: 30 minutes

Virtual Machine: **20533E-MIA-CL1**

User Name: **Student**

Password: **Pa55w.rd**

Before starting this lab, ensure that you have performed the "Preparing the Environment" demonstration tasks at the beginning of the first lesson in this module, and that the setup script has completed.

## Exercise 1: Implementing Docker hosts on Azure VMs

### Scenario

To test the planned deployment, you must identify the methods that would allow you to deploy Docker containers to Azure VMs. To accomplish this, you want to test deployment of a sample containerized nginx web server, available from Docker Hub.

Note: The Microsoft Azure portal is continually improved, and the user interface might have been updated since this lab was written. Your instructor will make you aware of any differences between the steps described in the lab and the current Azure portal user interface.

The main tasks for this exercise are as follows:

1. Connect to an Azure VM running Linux
2. Install Docker and deploy a container to an Azure VM

#### 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 running Docker.
2. Start Microsoft Edge and browse to the Azure portal. When prompted, sign in by using the Microsoft account that is the Service Administrator of your Azure subscription.
3. In the Azure portal, start **Bash (Linux)** session in **Cloud Shell**. If prompted, create a new storage account with the following settings:

* 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: **20533E0701-LabRG**
* Storage account: create a new storage account
* File share: create a new file share

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

1. From the Cloud Shell pane, use Azure CLI to identify the fully qualified domain name associated with the public IP address of the Linux Azure VM Docker host named **20533E0701-vm0**, which was created by the provisioning script.
2. From the Cloud Shell pane, establish an SSH session as **student** with the password **Pa55w.rd1234** to the Linux Azure VM Docker host **20533E0701-vm0** using the DNS name you identified in the previous step.

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

1. Within the SSH session to the Linux Azure VM Docker host, within the Cloud Shell pane, use the **apt install docker.io** command to install the Docker CE.
2. Within the SSH session to the Linux Azure VM Docker host, within the Cloud Shell pane, use the **docker run** command to start an **nginx** container from the Docker Hub, making it available via TCP port 80.
3. Monitor the progress of the container deployment. Verify the successful outcome, by running the **docker ps** command.
4. From MIA-CL1, start Microsoft Edge and browse to the fully qualified DNS 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 Linux Azure VM Docker host.

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

### Scenario

You intend to implement some A. Datum applications by using multiple containers. To accomplish this, you will test deployment of containers by using Docker Compose.

The main tasks for this exercise are as follows:

1. Install docker-compose and create a compose file
2. Deploy the containers with docker-compose to an Azure VM

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

1. In the Cloud Shell pane, within the SSH session to the Azure VM Docker host, install Docker Compose by running the **apt install docker-compose** command.
2. In the Cloud Shell pane, within the SSH session to the Azure VM Docker host, create a new file named **docker-compose.yml** with the following content (you can find the file 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

#### 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 Docker host, deploy containers defined in the **docker-compose.yml** file.
2. Monitor the progress of the container deployment. Use the **docker ps** command to verify the successful outcome.
3. On MIA-CL1, start Microsoft Edge and browse to the port 8080 on the target host via 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. Exit the SSH session and 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

### Scenario

Now that you have successfully implemented a Docker host in an Azure VM and deployed containerized images from Docker Hub, you want to test the setup and image management by using Container Registry. In your tests, you will use a sample image available from Docker Hub. You will start by creating a container registry. Next, you will download the sample image to your lab computer and upload it to the newly created private registry. Finally, you will deploy the image from the private registry to the Docker host in Azure VM.

The main tasks for this exercise are as follows:

1. Create an Azure Container Registry
2. Identify Azure Container Registry authentication settings.
3. Push an image to Azure Container Registry.
4. Download and deploy images from the Azure Container Registry
5. Remove the lab environment

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

1. In the Azure portal in the Microsoft Edge window, create a new **Azure Container Registry** instance with the following settings:

* 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: **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. If needed, on MIA-CL1, in the Azure portal, in the Microsoft Edge window, start Cloud Shell.
2. Use Azure CLI to identify the value of the **Login server** property of the newly created Azure Container Registry instance and one of the passwords necessary to access it.

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

1. From the Cloud Shell pane, connect via SSH to the Linux Azure VM Docker host.
2. Within the SSH session to the Linux Azure VM Docker host, within the Cloud Shell pane, use the **docker login** command to sign into the newly created Azure Container Registry instance. To sign in, use the name of the registry, the name of the **Login server**, and the password you identified in the previous task.
3. Ensure that you receive the **Login succeeded** message. Next, pull the **microsoft/aci-helloworld** image from Docker Hub by running the **docker pull** command.
4. Wait for the image to be downloaded to the Azure VM Docker host. Next, tag the image with the Azure Container registry name by running the **docker tag** command with the ***login-server*/aci-helloworld:v1** parameter, replacing the ***login-server*** entry with the value you identified in the previous task.
5. Push the tagged image to the Azure Container registry by running the **docker push** command with the ***login-server*/aci-helloworld:v1** parameter, replacing the ***login-server*** entry with the value you identified in the previous task.
6. Wait for the image to be pushed to the registry. Next, on MIA-CL1, in the Azure portal, from the container registry blade, verify that the Azure Container registry contains the **aci-helloworld** repository

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

1. Within the SSH session to the Linux Azure VM Docker host, within the Cloud Shell pane, pull the newly tagged image from the Azure Container registry, by running the **docker pull** command with the ***login-server*/aci-helloworld:v1** parameter, replacing the ***login-server*** entry with the value you identified earlier in this exercise.
2. Note that, in this case, the image does not need to be downloaded, since it is already present on the target Docker Azure VM.
3. Deploy the image downloaded from the Azure Container registry, by running the **docker run** command with the ***login-server*/aci-helloworld:v1** parameter, replacing the ***login-server*** entry with the value you identified earlier in this exercise. Deploy the web server running in the container such that it is accessible via port 8081.
4. Run the **docker ps** command to verify that the image has been successfully deployed.
5. Note that the output includes the tagged image.
6. Terminate the SSH session 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 the network security group associated with the network interface of the Linux Azure VM Docker host. On MIA-CL1, in the Azure portal window, navigate to the **20533E0701-web-nsg** blade.
2. From the **20533E0701-web-nsg** blade, add an inbound security rule with the following settings:

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

**Result**: After you complete this exercise, you should have successfully tested image deployment by using Azure Container Registry.

**Question**

Which method would you use when deploying Docker hosts on Azure VMs?

**Question**

What authentication and authorization method do you intend to use when implementing Azure Container Registry?

# Lab B: Implementing Azure Container Service (AKS)

### Scenario

A. Datum is considering implementing containers on a larger scale by leveraging the capabilities that AKS offers. You intend to test its functionality. You want to test load balancing and scaling of a sample containerized application.

### Objectives

After completing this lab, you will be able to:

* Create an AKS cluster.
* Manage the AKS cluster.

### Lab Setup

Estimated Time: 30 minutes

Virtual Machine: **20533E-MIA-CL1**

User Name: **Student**

Password: **Pa55w.rd**

## Exercise 1: Creating an AKS cluster

### Scenario

You must start by identifying the prerequisites for deploying an AKS cluster. You plan to install the cluster by using Azure CLI.

The main tasks for this exercise are as follows:

1. Create an AKS cluster
2. Connect to the 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**. In the Azure portal, in the Microsoft Edge window, start the Bash prompt in **Cloud Shell**.
2. From the bash prompt, in the Cloud Shell pane, create a new resource group named **20533E0703-LabRG** in the Azure region that you chose when running the provisioning script at the beginning of this module.
3. From the bash prompt, in the Cloud Shell pane, use the **az aks create** command with the following parameters to create a new Kubernetes cluster:

* --resource-group: **20533E0703-LabRG**
* --name: **20533E0703-k8scluster**
* --generate-ssh-keys
* --node-count: **1**
* --node-vm-size: the VM size you chose when running the provisioning script at the beginning of this module

1. Wait for the deployment to complete.

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

1. Download and configure the credentials to access the AKS cluster, by running the **az aks kubernetes get-credential** command with the following parameters from the bash prompt in the Cloud shell pane:

* --resource-group: **20533E0703-LabRG**
* --name **20533E0703-k8scluster**

1. Verify connectivity to the AKS cluster, by running the **kubectl get nodes** command from the bash prompt in the Cloud Shell pane
2. Review the output and verify that the agent nodes are reporting the **Ready** status.

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

## Exercise 2: Managing an AKS cluster

### Scenario

With the new AKS cluster running, you must connect to it, deploy a sample containerized application, and validate its availability and resiliency by testing clustering features such as scaling and load balancing.

The main tasks for this exercise are as follows:

1. Deploy a containerized application to the AKS cluster
2. Manage deployment of a containerized application on the AKS cluster
3. Remove the lab environment.

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

1. In the Azure portal, in the Microsoft Edge window, in the Cloud shell pane, create a deployment named **nginx-20533e0703** using the nginx image from the Docker Hub by running the **kubectl run** command with the following parameters:

* --image: **nginx**
* --replicas: **1**
* --port: **80**

1. Verify that a Kubernetes pod has been created by running **kubectl get pods** command from the bash prompt in the Cloud Shell pane.
2. Identify the state of the deployment by running **kubectl get deployment** command from the bash prompt in the Cloud Shell pane.
3. Make the deployment **nginx-20533e0703** available from Internet by running **kubectl expose** command from the bash prompt in the Cloud Shell pane with the following parameters:

* --port: **80**
* --type: **LoadBalancer**

1. Identify whether the public IP address has been provisioned by running **kubectl get services** command from the bash prompt in the Cloud Shell pane.
2. Repeat step 5 until the value in the **EXTERNAL-IP** column for **nginx-20533e0703** changes from **<pending>** to a public IP address. At that point, note the public IP address in the **EXTERNAL-IP** column for **nginx-20533e0703**.
3. 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. Scale the deployment **nginx-20533e0703** by running **kubectl scale** command from the bash prompt in the Cloud Shell pane with the --replicas parameter set to **2**.
2. Verify the outcome of scaling the deployment by running **kubectl get pods** command from the bash prompt in the Cloud Shell pane.
3. In the output of the command you ran in the previous step, verify that the number of pods increased to 2.
4. Delete the **nginx-20533e0703** deployment by running **kubectl delete** command from the bash prompt in the Cloud Shell pane.
5. Verify that the command you ran in the previous step completed successfully by running **kubectl get deployment** command from the bash prompt in the Cloud Shell pane.

#### Task 3: Remove the lab environment

1. On MIA-CL1, close all open windows without saving any files.
2. Start **Windows PowerShell** as Administrator and, from the **Administrator: Windows PowerShell** window, run **Remove-20533EEnvironment**.
3. When prompted, sign in by using the Microsoft account that is the Service Administrator of your Azure subscription.
4. If you have multiple Azure subscriptions, select the one you want the script to target.
5. If prompted, specify the current lab number.
6. When prompted for confirmation, type **y**.
7. Start Microsoft Edge, browse to the Azure portal, and sign in by using the Microsoft account that is the Service Administrator of your Azure subscription.
8. In the Azure portal, reset the dashboard to the default state.
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.

**Question**

What deployment methodology would you choose when deploying AKS clusters?

**Question**

What are the primary advantages of using AKS for container orchestration?

©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.