

# Azure Developer Series

## Migrating a legacy ASP.NET 2-tier application to Azure using Container Services

Hands-On-Labs step-by-step guides

Prepared by:

Peter De Tender

CEO and Lead Technical Trainer  
PDTIT and 007FFFlearning.com

@pdtit

@007FFFlearning

Version: April 2019 – 2.0

Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

The names of manufacturers, products, or URLs are provided for informational purposes only and Microsoft makes no representations and warranties, either expressed, implied, or statutory, regarding these manufacturers or the use of the products with any Microsoft technologies. The inclusion of a manufacturer or product does not imply endorsement of Microsoft of the manufacturer or product. Links may be provided to third party sites. Such sites are not under the control of Microsoft and Microsoft is not responsible for the contents of any linked site or any link contained in a linked site, or any changes or updates to such sites. Microsoft is not responsible for webcasting or any other form of transmission received from any linked site. Microsoft is providing these links to you only as a convenience, and the inclusion of any link does not imply endorsement of Microsoft of the site or the products contained therein.

© 2018 Microsoft Corporation. All rights reserved.

Microsoft and the trademarks listed at <https://www.microsoft.com/en-us/legal/intellectualproperty/Trademarks/Usage/General.aspx> are trademarks of the Microsoft group of companies. All other trademarks are property of their respective owners.



# Contents

Abstract and Learning Objectives .....	4
Hands-On-Lab Scenario .....	5
Requirements .....	5
Naming Conventions:.....	5
Azure Subscription:.....	5
Other requirements: .....	5
Alternative Approach: .....	6
Final Remarks:.....	6
Lab 6: Deploy and run Azure Container Services (ACS) with Kubernetes; .....	7
What you will learn .....	7
Time Estimate .....	7
Task 1: Deploying Azure Container Services (ACS) using Azure CloudShell .....	7
Task 2: Integrating Azure Container Registry docker image with ACS for Kubernetes .....	10
Summary .....	17

# Migrating a legacy ASP.NET 2-tiered application to Azure using Container Services - Hands-On-Labs step-by-step

## Abstract and Learning Objectives

This workshop enables anyone to learn, understand and build a Proof of Concept, in performing a multi-tiered legacy ASP.NET web application using Microsoft SQL Server database, platform migration to Azure public cloud, leveraging on different Azure Platform Azure A Service (PaaS) and Azure Container Services.

After an introductory module on cloud app migration strategies and patterns, students get introduced to the basics of automating Azure resources deployments using Visual Studio and Azure Resource Manager (ARM) templates. Next, attendees will learn about Microsoft SQL database migration to SQL Azure PaaS, as well as deploying and migrating Azure Web Apps.

After these foundational platform components, the workshop will totally focus on the core concepts and advantages of using containers for running web apps, based on Docker, Azure Container Registry (ACR), Azure Container Instance (ACI), as well as how to enable container cloud-scale using Azure Container Services (ACS) with Kubernetes and Azure Kubernetes Service (AKS).

The focus of the workshop is having a Hands-On-Labs experience, by going through the following exercises and tasks:

- Deploying a 2-tier Azure Virtual Machine (Webserver and SQL database Server) using ARM-template automation with Visual Studio 2017;
- Migrating a legacy SQL 2012 database to Azure SQL PaaS (Lift & Shift);
- Migrating a legacy ASP.NET web application to Azure Web Apps (Lift & Shift);
- Containerizing a legacy ASP.NET web application using Docker;
- Running Azure Container Instance (ACI) from an Azure Container Registry (ACR) image;
- Deploy and run Azure Container Services (ACS) with Kubernetes;
- Deploy and run Azure Kubernetes Services (AKS);
- Managing and Monitoring Azure Container Services (ACS) and Azure Kubernetes Services (AKS);

## Hands-On-Lab Scenario

The baseline of the hands-on-lab scenario is starting from an 8-year-old legacy ASP.NET application, developed around 2010, currently offering a product catalog browsing web page, pulling the product catalog list from a legacy Microsoft SQL Server 2012 database, running on dedicated Windows Server 2012 R2 Virtual Machines. (This could be seen as a subset of a larger application landscape within your organization, think of manufacturing database information, HR solutions, e-commerce platforms,... and alike). Imagine this application is running in our on-premises datacenter, where you want to perform an “application digital migration” to Azure Public cloud. You will use several Azure cloud-native services and solutions, ranging from Virtual Machines, Platform Services and different Container Solutions on Azure.

## Requirements

### Naming Conventions:

IMPORTANT: Most Azure resources require unique names. Throughout these steps you will see the word “[SUFFIX]” as part of resource names. You should replace this with your initials, guaranteeing those resources get uniquely named.

### Azure Subscription:

Participants need a “pay-as-you-go”, MSDN or other paid Azure subscription

- a) **Azure Trial subscriptions won't work**
- b) In one of the Azure Container Services tasks, you are required to create an Azure AD Service Principal, which typically requires an Azure subscription owner to log in to create this object. If you don't have the owner right in your Azure subscription, you could ask another person to execute this step for you.
- c) The Azure subscription must allow you to run enough cores, used by the baseline Virtual Machines, but also later on in the tasks when deploying the Azure Container Services, where ACS agent and master machines are getting set up. If you follow the instructions as written out in the lab guide, you need 12 cores.
- d) If you run this lab setup in your personal or corporate Azure payable subscription, using the configuration as described in the lab guide, the estimated Azure consumption costs for running the setups during the 2 days of the workshop is \$20.

### Other requirements:

Participants need a local client machine, running a recent Operating System, allowing them to:

- browse to <https://portal.azure.com> from a most-recent browser;
- establish a secured Remote Desktop (RDP) session to a lab-jumpVM running Windows Server 2016;

## Alternative Approach:

Where the lab scenario assumes all exercises will be performed from within the lab-jumpVM, (since several tools will be installed on the lab-jumpVM or are already installed by default), participants could also execute (most, if not all...) steps from their local client machine.

The following tools are being used throughout the lab exercises:

- Visual Studio 2017 community edition (updated to latest version)
- Docker for Windows (updated to latest version)
- Azure CLI 2.0 (updated to latest version)
- Kubernetes CLI (updated to latest version)

Make sure you have these tools installed prior to the workshop, if you are not using the lab-jumpVM. You should also have full administrator rights on your machine to execute certain steps within using these tools.

## Final Remarks:

**VERY IMPORTANT:** You should be typing all of the commands as they appear in the guide, except where explicitly stated in this document. Do not try to copy and paste from Word to your command windows or other documents where you are instructed to enter information shown in this document. There can be issues with Copy and Paste from Word or PDF that result in errors, execution of instructions, or creation of file content.

**IMPORTANT:** Most Azure resources require unique names. Throughout these steps you will see the word "[SUFFIX]" as part of resource names. You should replace this with your initials, guaranteeing those resources get uniquely named.

## Lab 6: Deploy and run Azure Container Services (ACS) with Kubernetes;

### What you will learn

After having deployed Docker images as a stand-alone Azure Container Instance and Azure Web App for Containers, you move on with the more serious container solution in Azure, Azure Container Services (ACS) with Kubernetes. Starting from the deployment of the ACS resources in Azure using Azure CloudShell, you will again integrate the `webvmsamplesitedocker` Docker image, and run it in a Kubernetes cluster.

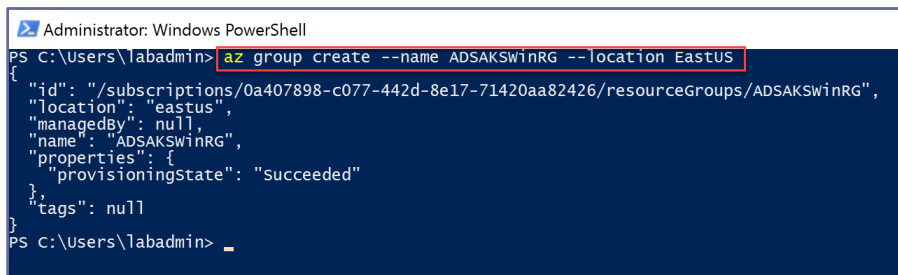
### Time Estimate

This lab is estimated to take 60min

### Task 1: Deploying Azure Container Services (ACS) using Azure CloudShell

1. From the lab-jumpVM, **Open PowerShell**. (depending on your machine state, it might be required to authenticate to Azure again before Azure CLI can be used, using **az login**).
2. Run the following command to create a new Azure Resource Group for our ACS resources:

```
az group create --name [SUFFIX]AKSWinRG --location EastUS
```



```
Administrator: Windows PowerShell
PS C:\Users\labadmin> az group create --name ADSAKSWinRG --location EastUS
{
  "id": "/subscriptions/0a407898-c077-442d-8e17-71420aa82426/resourceGroups/ADSAKSWinRG",
  "location": "eastus",
  "managedBy": null,
  "name": "ADSAKSWinRG",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null
}
PS C:\Users\labadmin>
```

3. With the Resource Group in place, **execute the following command to deploy the Azure Container Service** itself, using **Kubernetes** as the orchestrator type:

```
az acs create --name ADSKubernetesCluster --orchestrator-type
Kubernetes --resource-group ADSAKSWinRG --agent-count 2 --
generate-ssh-keys --windows --admin-username azureuser --admin-
password L@BadminPa55w.rd
```

```

Administrator: Windows PowerShell
PS C:\Users\labadmin> az acs create --name ADSKubernetesCluster --orchestrator-type Kubernetes --resource-group ADSAKSWinRG --agent-count 2 --generate-ssh-keys --windows --admin-username labadmin --admin-password L@BadminPa55w.rd
SSH key files C:\Users\labadmin\.ssh\id_rsa and C:\Users\labadmin\.ssh\id_rsa.pub have been generated under ~/.ssh to allow SSH access to the VM. If using machines without permanent storage like Azure Cloud Shell without an attached file share, back up your keys to a safe location
Finished service principal creation[#####] 100.0000%

```

4. **Note** the informational message about the SSH keys. These are stored in the C:\Users\labadmin\.ssh folder
5. **Wait for** the deployment to succeed. A JSON output should be visible in the CloudShell with the Azure Container Services details. This could take up to 10 minutes on average.

Intermittently checking back in the Azure Resource Group, will show the different Azure Resources being created.

Home > Resource groups > ADSAKSWinRG - Deployments > 82669fca-bb73-4c86-808d-a5019d41c2d6 - Overview

### 82669fca-bb73-4c86-808d-a5019d41c2d6 - Overview

Deployment

«

Delete Cancel Redeploy Refresh

Overview

Outputs

Inputs

Template

...

Your deployment is underway

Check the status of your deployment, manage resources, or troubleshoot deployment issues. Pin this page to your dashboard to easily find it next time.

Deployment name: 82669fca-bb73-4c86-808d-a5019d41c2d6  
Subscription: [Microsoft Azure Sponsorship](#)  
Resource group: [ADSAKSWinRG](#)

DEPLOYMENT DETAILS [\(Download\)](#)  
Start time: 10/1/2018 4:39:39 AM  
Duration: 4 minutes 17 seconds  
Correlation ID: 6f6b99c9-053f-425e-8944-14e4392311bd

RESOURCE	TYPE	STATUS	OPERATION DETAILS
k8s-master-6D5E96F6	Microsoft.Compute/...	Created	<a href="#">Operation details</a>
6D5E9acs9001	Microsoft.Compute/...	Created	<a href="#">Operation details</a>
6D5E9acs9000	Microsoft.Compute/...	Created	<a href="#">Operation details</a>
6D5E9acs900nic-0	Microsoft.Network/...	Created	<a href="#">Operation details</a>
k8s-master-6D5E96F6	Microsoft.Compute/...	OK	<a href="#">Operation details</a>
k8s-master-6D5E96F6	Microsoft.Network/...	Created	<a href="#">Operation details</a>

6. A successful deployment shows you the JSON output of the ACS resources

THE BEST WAY TO KNOW THE FUTURE  
IS TO LEARN IT

<http://www.007FFFLearning.com>



```
Administrator: Windows PowerShell
PS C:\Users\labadmin> az acs create --name ADSkubernetesCluster --orchestrator-type Kubernetes --resource-group ADSAKSwi
nRG --agent-count 2 --generate-ssh-keys --windows --admin-username azureuser --admin-password L@BadminPa55w.rd
{
  "id": "/subscriptions/0a407898-c077-442d-8e17-71420aa82426/resourceGroups/ADSAKSwinRG/providers/Microsoft.Resources/de
ployments/azurecli1538368717.649531194121",
  "location": null,
  "name": "azurecli1538368717.649531194121",
  "properties": {
    "correlationId": "edc41767-fdab-445b-8191-bddd87547ca1",
    "debugSetting": null,
    "dependencies": [],
    "duration": "PT12M57.3655654s",
    "mode": "Incremental",
    "onErrorDeployment": null,
    "outputResources": [
      {
        "id": "/subscriptions/0a407898-c077-442d-8e17-71420aa82426/resourceGroups/ADSAKSwinRG/providers/Microsoft.Contai
nerService/containerServices/ADSkubernetesCluster",
        "resourceGroup": "ADSAKSwinRG"
      }
    ],
    "outputs": {
      "masterFQDN": {
        "type": "String",
        "value": "adskuberne-adsakswinrg-0a4078mgmt.eastus.cloudapp.azure.com"
      },
      "sshMaster0": {
        "type": "String",
        "value": "ssh azureuser@adskuberne-adsakswinrg-0a4078mgmt.eastus.cloudapp.azure.com -A -p 22"
      }
    },
    "parameters": {
      "clientSecret": {
        "type": "SecureString"
      }
    }
  }
}
```

7. Next, get the credentials to authenticate to the Azure Container Services cluster, using the following command:

```
az acs kubernetes get-credentials --resource-group
[SUFFIX]akswinrg --name [SUFFIX]kubernetescluster
```

```
Administrator: Windows PowerShell
PS C:\Users\labadmin> az acs kubernetes get-credentials --resource-group adsakswinrg --name adskubernetescluster
PS C:\Users\labadmin>
```

and if you run it again:

```
Administrator: Windows PowerShell
PS C:\Users\labadmin> az acs kubernetes get-credentials --resource-group adsakswinrg --name adskubernetescluster
Merged "adskuberne-adsakswinrg-0a4078mgmt" as current context in C:\Users\labadmin\.kube\config
PS C:\Users\labadmin>
```

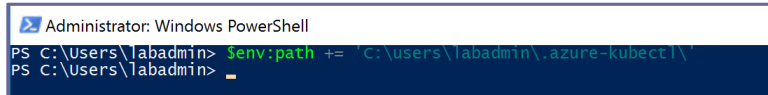
8. In order to manage the Kubernetes cluster we deployed, one needs to install the Kubectl command line tool. This is done using the following command:

```
az acs kubernetes install-cli
```

```
Administrator: Windows PowerShell
PS C:\Users\labadmin> az acs kubernetes install-cli
Downloading client to "C:\Users\labadmin\.azure-kubectl\kubectl.exe" from "https://storage.googleapis.com/kubernetes-rele
ase/release/v1.12.0/bin/windows/amd64/kubectl.exe"
Please add "C:\Users\labadmin\.azure-kubectl" to your search PATH so the 'kubectl.exe' can be found. 2 options:
1. Run "set PATH=%PATH%;C:\Users\labadmin\.azure-kubectl" or "$env:path += 'C:\Users\labadmin\.azure-kubectl'" for P
owerShell. This is good for the current command session.
2. Update system PATH environment variable by following "Control Panel->System->Advanced->Environment Variables", an
d re-open the command window. you only need to do it once
```

9. As well as **updating the Path variable**, by running the following PowerShell command:

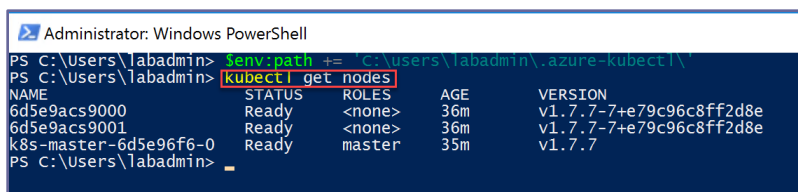
```
$env:path += 'C:\users\labadmin\.azure-kubectl'
```



```
Administrator: Windows PowerShell
PS C:\Users\labadmin> $env:path += 'C:\users\labadmin\.azure-kubectl\'
PS C:\Users\labadmin>
```

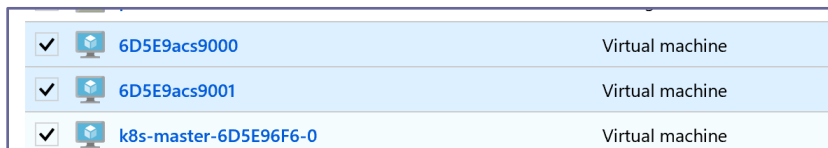
10. To validate our Kubernetes cluster, initiate the following kubectl command:




```
kubectl get nodes
```



```
Administrator: Windows PowerShell
PS C:\Users\labadmin> $env:path += 'C:\users\labadmin\.azure-kubectl\'
PS C:\Users\labadmin> kubectl get nodes
NAME                                STATUS    ROLES    AGE      VERSION
6d5e9acs9000                        Ready    <none>    36m      v1.7.7-e79c96c8ff2d8e
6d5e9acs9001                        Ready    <none>    36m      v1.7.7-e79c96c8ff2d8e
k8s-master-6d5e96f6-0              Ready    master    35m      v1.7.7
```

11. The different nodes refer to the **Kubernetes resources** running in Azure, which can be seen from within the **Azure Resource Group** that holds the Kubernetes cluster. **Note each node refers to an Azure Virtual Machine.**

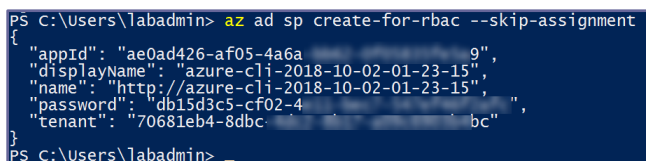


<input checked="" type="checkbox"/>	 6D5E9acs9000	Virtual machine
<input checked="" type="checkbox"/>	 6D5E9acs9001	Virtual machine
<input checked="" type="checkbox"/>	 k8s-master-6D5E96F6-0	Virtual machine

## Task 2: Integrating Azure Container Registry docker image with ACS for Kubernetes

1. To allow Kubernetes to pull our Docker image from the Azure Container Registry, **we need to define an Azure Service Principal object**, which integrates with RBAC. **Create the Service Principal** as follows, from within your **PowerShell** window:

```
az ad sp create-for-rbac --skip-assignment
```



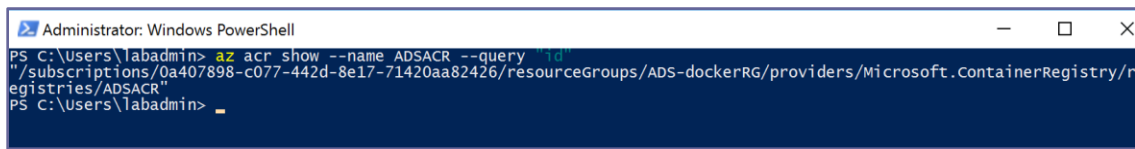
```
PS C:\Users\labadmin> az ad sp create-for-rbac --skip-assignment
{
  "appId": "ae0ad426-af05-4a6a-9",
  "displayName": "azure-cli-2018-10-02-01-23-15",
  "name": "http://azure-cli-2018-10-02-01-23-15",
  "password": "db15d3c5-cf02-4",
  "tenant": "70681eb4-8dbc-bc"
}
PS C:\Users\labadmin>
```

Since we need parts of this information later on, might be good to **copy this to a Notepad**

doc for easy retrieval.

2. This command creates an applicationID, provides displayname and tenant information that you need later on in the Kubernetes YAML-file (similar to the Dockerfile we used earlier, but for Kubernetes deployments).
3. Next item information we need is the **full Azure Resource ID** for our Azure Container Registry. This information **can be retrieved** using the following command:

```
az acr show --name [SUFFIX]ACR --query "id" --output table
```



```
Administrator: Windows PowerShell
PS C:\Users\labadmin> az acr show --name ADSACR --query "id"
"/subscriptions/0a407898-c077-442d-8e17-71420aa82426/resourceGroups/ADS-dockerRG/providers/Microsoft.ContainerRegistry/registries/ADSACR"
PS C:\Users\labadmin>
```

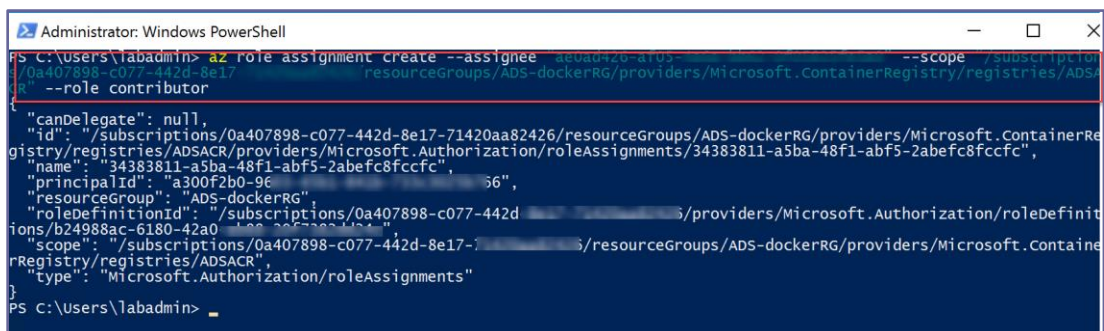
Since we need parts of this information later on, might be good to **copy this to a Notepad doc** for easy retrieval.

4. Next, **assign the contributor role** for the previously created "appid", to this ACR object, by executing the following command:

```
az role assignment create --assignee "appid" --scope "ACRid" --role contributor
```

which maps like this in my environment (replaced some characters for security reasons):

```
az role assignment create --assignee "ae0ad426-af05-4a6a-0000-00000000" --scope "/subscriptions/0a407898-c077-0000-0000-714200000000/resourceGroups/ADS-dockerRG/providers/Microsoft.ContainerRegistry/registries/ADSACR" --role contributor
```

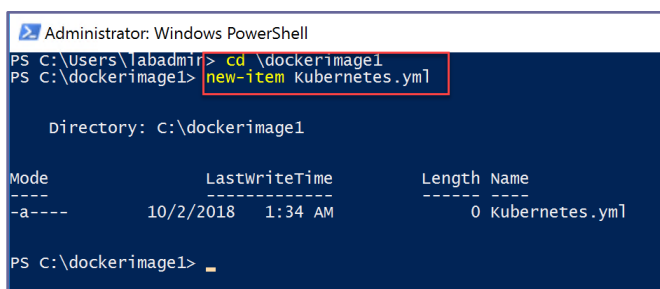


```
Administrator: Windows PowerShell
PS C:\Users\labadmin> az role assignment create --assignee ae0ad426-af05-4a6a-0000-00000000 --scope "/subscriptions/0a407898-c077-442d-8e17-71420aa82426/resourceGroups/ADS-dockerRG/providers/Microsoft.ContainerRegistry/registries/ADSACR" --role contributor
{
  "canDelegate": null,
  "id": "/subscriptions/0a407898-c077-442d-8e17-71420aa82426/resourceGroups/ADS-dockerRG/providers/Microsoft.ContainerRegistry/registries/ADSACR/providers/Microsoft.Authorization/roleAssignments/34383811-a5ba-48f1-abf5-2abefc8fccfc",
  "name": "34383811-a5ba-48f1-abf5-2abefc8fccfc",
  "principalId": "a300f2b0-96[REDACTED]56",
  "resourceGroup": "ADS-dockerRG",
  "roleDefinitionId": "/subscriptions/0a407898-c077-442d-8e17-71420aa82426/providers/Microsoft.Authorization/roleDefinitions/b24988ac-6180-42a0-b266-206f30211144",
  "scope": "/subscriptions/0a407898-c077-442d-8e17-71420aa82426/resourceGroups/ADS-dockerRG/providers/Microsoft.ContainerRegistry/registries/ADSACR",
  "type": "Microsoft.Authorization/roleAssignments"
}
PS C:\Users\labadmin>
```

With all the back-end information and RBAC Service Principals in place, we can build our **YAML-deployment file for Kubernetes**. Key information in here is the name of your Azure Container Registry, the Docker container file that you want to push to the Kubernetes cluster, and what port the container should run on.

5. Start with creating a new empty "Kubernetes.yml" file on your lab-jumpVM machine, using the following PowerShell commands:

```
cd \Dockerimage1          (we use the same location where we created the Dockerfile)
New-item kubernetes.yml
```

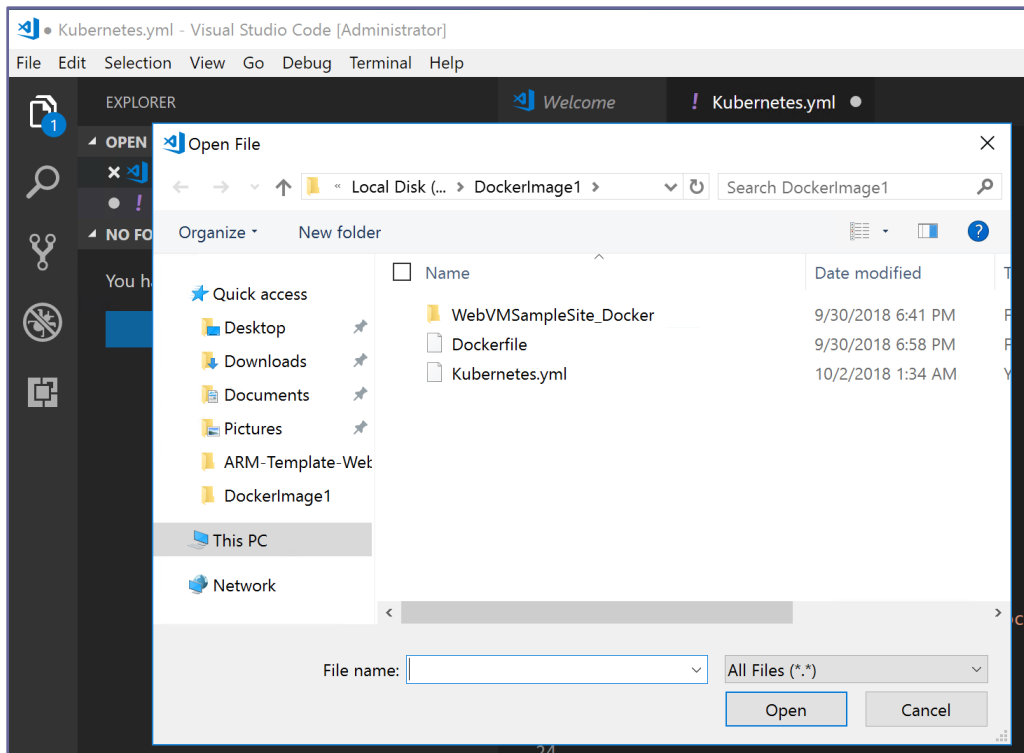


The screenshot shows a Windows PowerShell window titled "Administrator: Windows PowerShell". The prompt is at "PS C:\Users\labadmin>". The user enters "cd \dockerimage1", and the prompt changes to "PS C:\dockerimage1>". Then, the user enters "new-item kubernetes.yml", which is highlighted with a red box. Below the command, the directory is listed as "Directory: c:\dockerimage1". A table shows the file creation details:

Mode	LastWriteTime	Length	Name
-a----	10/2/2018 1:34 AM	0	kubernetes.yml

The prompt returns to "PS C:\dockerimage1>".

6. Next, open the kubernetes.yml file in Visual Studio Code (a much better editor than Notepad ☺), by Starting Visual Studio Code from the Start Screen | File | Open | browse to the kubernetes.yml file



Paste in the following lines of code:

```
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: kubernetestsample
spec:
  replicas: 1
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1
      minReadySeconds: 5
  template:
    metadata:
      labels:
```

```
app: kubernetes-sample
spec:
  containers:
    name: adsacr
    image: adsacr.azurecr.io/webvmsamplesitedocker3:latest
  ports:
    containerPort: 80
  imagePullSecrets:
    name: acr-auth
---
apiVersion: v1
kind: Service
metadata:
  name: kubernetes-sample
spec:
  type: LoadBalancer
  ports:
    port: 80
  selector:
    app: kubernetes-sample
```

```

! kubernetes.yml x
1  apiVersion: apps/v1beta1
2  kind: Deployment
3  metadata:
4    name: dockerwebvmsample
5  spec:
6    replicas: 1
7    strategy:
8      rollingUpdate:
9        maxSurge: 1
10       maxUnavailable: 1
11    minReadySeconds: 5
12    template:
13      metadata:
14        labels:
15          app: dockerwebvmsample
16      spec:
17        containers:
18          - name: webapplication3
19            image: webapplication3pdt.azurecr.io/dockerwebvmsample3:latest
20            ports:
21              - containerPort: 80
22            imagePullSecrets:
23              - name: acr-auth
24
25  ---
26  apiVersion: v1
27  kind: Service
28  metadata:
29    name: dockerwebvmsample
30  spec:
31    type: LoadBalancer
32    ports:
33      - port: 80
34    selector:
35      app: dockerwebvmsample
36
37

```

7. Replacing the image URL to your specific Azure Container Registry name:

- name: [SUFFIX]acr.azurecr.io/webvmsamplesitedocker3:latest

8. Save the kubernetes.yml file

9. Now we are ready to run the actual kubernetes deployment, running the following command:

kubectl create -f "path to your yaml file"

```
Administrator: Windows PowerShell
PS C:\dockerimage1> kubectl create -f .\kubernetes.yml
deployment.apps "dockerwebvmsample" created
service "dockerwebvmsample" created
PS C:\dockerimage1> _
```

10. While this command **should complete immediately**, it is **no guarantee** our container is already deployed into the Kubernetes pod. But you already know **you can validate this** by running the following command:

```
kubectl get pods
```

```
Administrator: Windows PowerShell
PS C:\dockerimage1> kubectl get pods
NAME                                READY    STATUS             RESTARTS   AGE
dockerwebvmsample-2370420970-86929 0/1      ContainerCreating   0           2m
PS C:\dockerimage1> _
```

11. This **shows different status** output, from **ContainerCreating** to **Running**; repeat the same **command** a few times to see the full process happening in the back-end. (in my lab setup here, it took about 8 minutes before the status changed to Running, fyi)

```
Administrator: Windows PowerShell
PS C:\dockerimage1> kubectl get pods
NAME                                READY    STATUS             RESTARTS   AGE
dockerwebvmsample-2370420970-86929 1/1      Running            0           8m
PS C:\dockerimage1> _
```

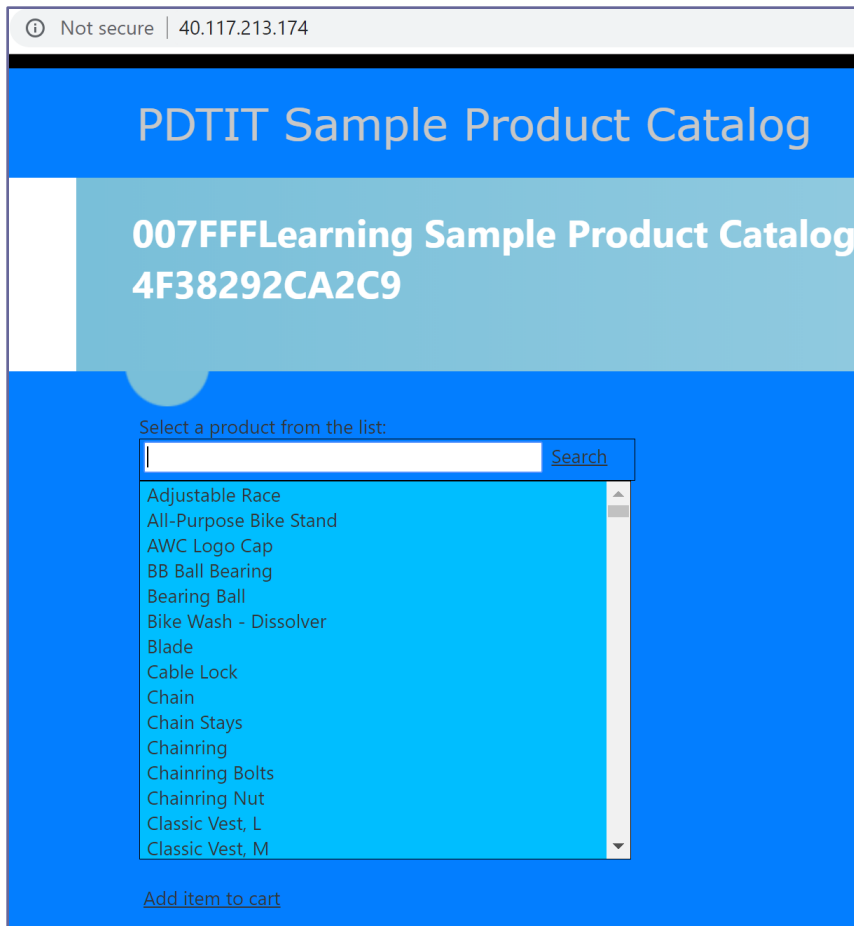
12. Similar to the Docker container running on our local lab-VM several labs back, we need to check on the actual Public IP-address of this running container. This can be retrieved using the following command:

```
Kubectl get service dockerwebvmsample --watch
```

```
Administrator: Windows PowerShell
PS C:\dockerimage1> kubectl get service dockerwebvmsample --watch
NAME      TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
dockerwebvmsample LoadBalancer 10.0.96.172    40.117.213.174 80:32262/TCP 9m
_
```

13. **Testing the container** is actually running and exposing the web application, requires browsing to the EXTERNAL-IP from an internet browser





14. In a later lab, you will learn how to perform more detailed Azure Container Services monitoring for Kubernetes.
15. This completes this lab.

## Summary

In this lab, you deployed an Azure Container Service (ACS) for Kubernetes, as well as learned how to push a Docker image from your Azure Container Registry repository into this ACS Kubernetes cluster.

For any further information or references, please have a look at the following Microsoft Docs around containers:

<https://azure.microsoft.com/en-us/overview/containers/>

<https://azure.microsoft.com/en-us/services/kubernetes-service/>

<https://azure.microsoft.com/en-us/services/app-service/containers/>

<https://azure.microsoft.com/en-us/services/container-registry/>

<https://azure.microsoft.com/en-us/services/container-instances/>

<https://docs.microsoft.com/en-us/azure/aks/>

## Migrating a legacy ASP.NET 2-tier application to Azure using Container Services

Hands-On-Labs step-by-step guides

Prepared by:

**Peter De Tender**

CEO and Lead Technical Trainer  
PDTIT and 007FFFLearning.com

@pdtit                      @007FFFLearning

Version: April 2019 – 2.0

