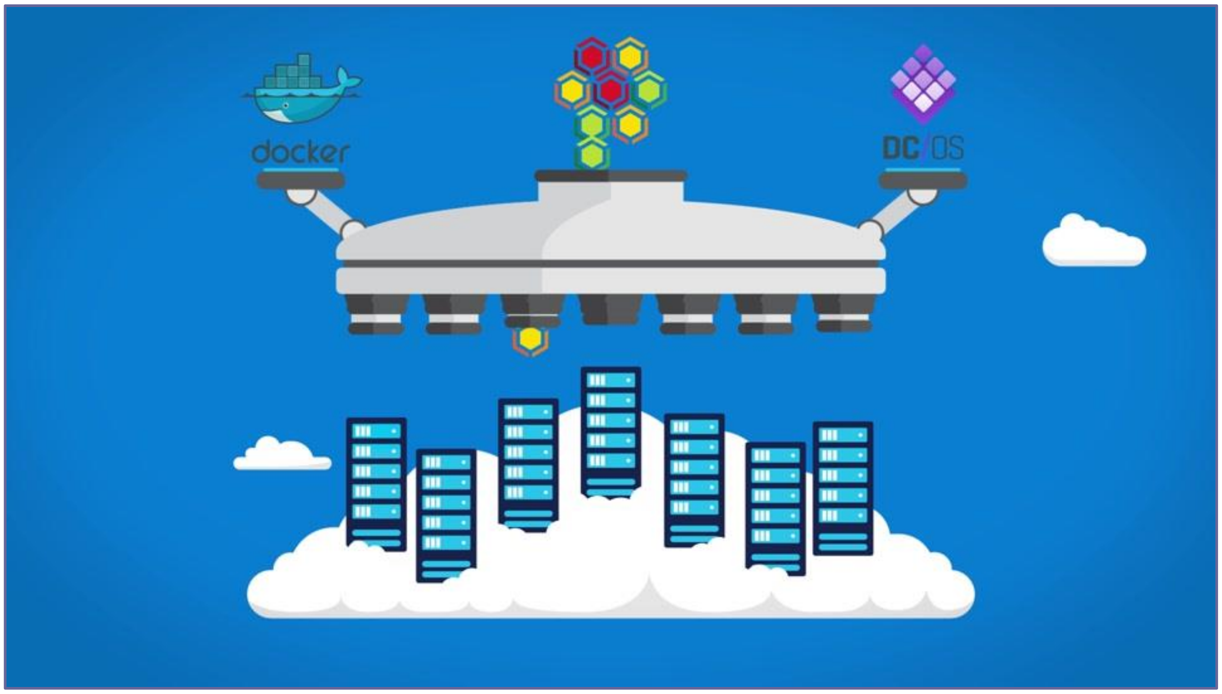


Azure Developer Series

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 – 2.0

Contents

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, wich 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.

Bonus Lab 9: Deploying a Windows Azure Kubernetes Services (AKS);

What you will learn

This lab is not part of the official content of the series, but since we managed to integrate the brand new announced (March 25th 2019) Windows Container Azure Kubernetes Service on Azure, and showcased a few demos in session 7 and 8, we decided to provide an additional lab guide on how to deploy it.

Time Estimate

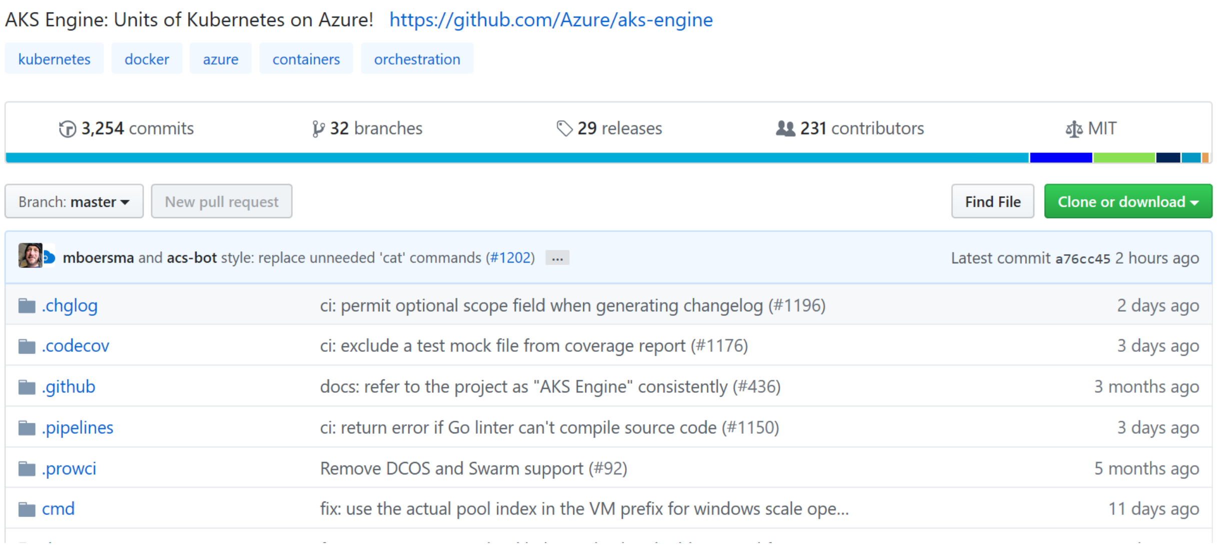
This lab shouldn't take longer than 60 minutes.

Task 1: Understanding Windows-based AKS Service Deployment with aks-engine

Azure provides a full and native integration of the Linux based AKS platform on Azure, allowing a deployment of the service using common Azure tools (Azure Portal, Azure PowerShell, Azure CLI, ARM templates, Terraform,…)

However, since the Windows-based AKS Cluster is still brand-new in Azure, the deployment and management works a little bit different. Instead of using any of the common Azure deployment tools, for now, we need to use aks-engine. Which is currently available as of GitHub Project.

1. Browse to https://github.com/Azure/aks-engine


AKS Engine: Units of Kubernetes on Azure! https://github.com/Azure/aks-engine

mboersma and acs-bot style: replace unneeded 'cat' commands (#1202) … Latest commit a76cc45 2 hours ago

.chglog	ci: permit optional scope field when generating changelog (#1196)	2 days ago
.codecov	ci: exclude a test mock file from coverage report (#1176)	3 days ago
.github	docs: refer to the project as "AKS Engine" consistently (#436)	3 months ago
.pipelines	ci: return error if Go linter can't compile source code (#1150)	3 days ago
.prowci	Remove DCOS and Swarm support (#92)	5 months ago
cmd	fix: use the actual pool index in the VM prefix for windows scale ope…	11 days ago

2. While we will not use a lot of the different tools and sample files offered in this Repo, it is an interesting resource if you want to continue learning how Windows Containers AKS will evolve and testing out different deployment.

3. From your management station (a Windows 10, Mac or Linux client having the Azure command line tools installed), you first need to install aks-engine on it. Binary downloads for the latest version of aks-engine are available on Github. Download AKS Engine for your operating system, extract the binary and copy it to your $PATH. You can also choose to install aks-engine using gofish. To do so, execute the command gofish install aks-engine. You can install gofish following the instructions for your OS.

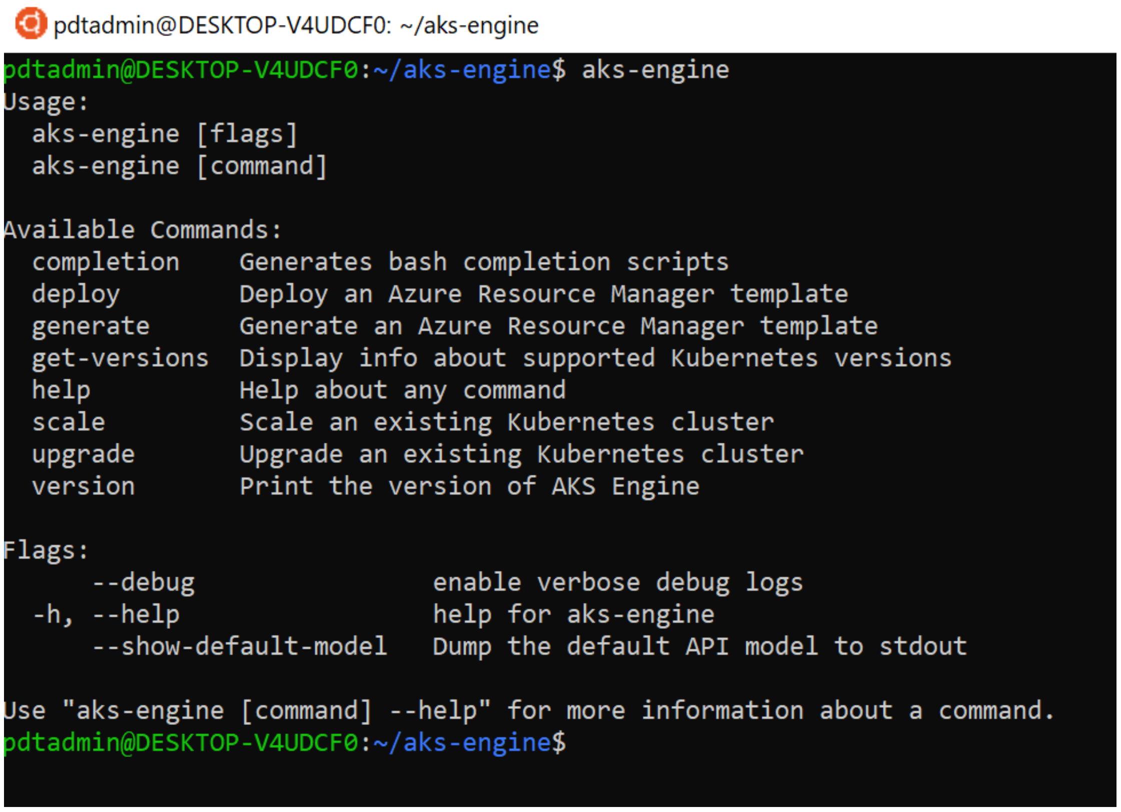
4. On macOS, you can install aks-engine with Homebrew. Run the command brew install Azure/aks-engine/aks-engine to do so.

5. On Windows, you can install aks-engine via Chocolatey by executing the command choco install aks-engine.

6. On Linux, the method I will be using, you can install aks-engine via install script doing:

```
curl -o get-akse.sh
https://raw.githubusercontent.com/Azure/aks-engine/master
/scripts/get-akse.sh

chmod 700 get-akse.sh

./get-akse.sh
```

7. Run aks-engine on the client to validate it is installed successfully and ready to run.


pdtadmin@DESKTOP-V4UDCF0: ~/aks-engine

```
pdtadmin@DESKTOP-V4UDCF0:~/aks-engine$ aks-engine
Usage:
  aks-engine [flags]
  aks-engine [command]

Available Commands:
  completion    Generates bash completion scripts
  deploy        Deploy an Azure Resource Manager template
  generate      Generate an Azure Resource Manager template
  get-versions  Display info about supported Kubernetes versions
  help          Help about any command
  scale         Scale an existing Kubernetes cluster
  upgrade       Upgrade an existing Kubernetes cluster
  version       Print the version of AKS Engine

Flags:
      --debug                enable verbose debug logs
  -h, --help                 help for aks-engine
      --show-default-model   Dump the default API model to stdout

Use "aks-engine [command] --help" for more information about a command.
pdtadmin@DESKTOP-V4UDCF0:~/aks-engine$
```

Task 2: Deploying the Windows-based AKS Cluster with aks-engine

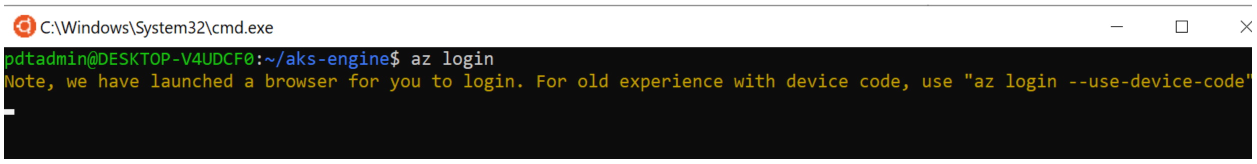
As our management client is up-and-running now, we can continue with preparing and running the actual AKS cluster deployment. This involves the following steps:

1. Log on to Azure
2. Create a new Azure Resource Group that will hold the AKS Cluster resources
3. Create a Service Principal, allowing all AKS Cluster resources to communicate with each other, within the boundaries of the Resource Group.
4. Create/Update an aks-engine deployment API-setup file (JSON), having the characteristics and parameters needed for the deployment
5. Run aks-deploy with deployment parameters

That will kick off the actual AKS Cluster deployment, based on an Azure Resource Manager template that gets generated.

1. Authenticate to Azure by running

   `az login`



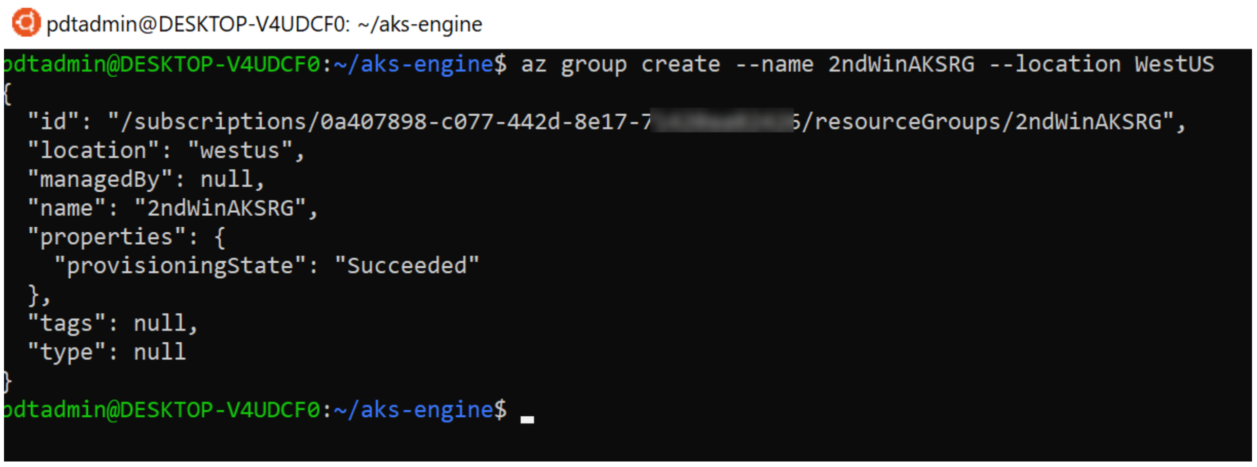
C:\Windows\System32\cmd.exe

```
pdtadmin@DESKTOP-V4UDCF0:~/aks-engine$ az login
Note, we have launched a browser for you to login. For old experience with device code, use "az login --use-device-code"
```

2. This will prompt you for your Azure administrative Credentials. Once logged on from the browser, you can close the browser session and return to the management client.

3. Create a new Azure Resource Group, by initiating the following command:

   `az group create –name 2ndWinAKSRG –location WestUS`

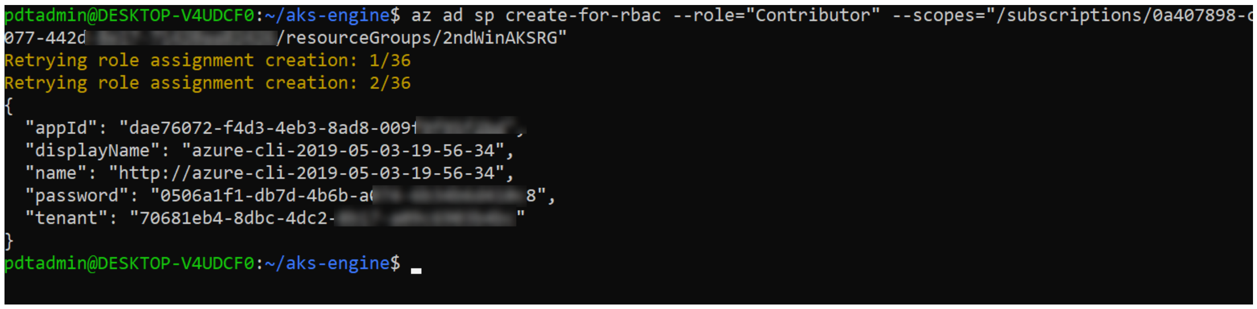


pdtadmin@DESKTOP-V4UDCF0: ~/aks-engine

```
pdtadmin@DESKTOP-V4UDCF0:~/aks-engine$ az group create --name 2ndWinAKSRG --location WestUS
{
  "id": "/subscriptions/0a407898-c077-442d-8e17-7[redacted]5/resourceGroups/2ndWinAKSRG",
  "location": "westus",
  "managedBy": null,
  "name": "2ndWinAKSRG",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null,
  "type": null
}
pdtadmin@DESKTOP-V4UDCF0:~/aks-engine$
```

4. Next, create a new Azure AD Service Principal, by initiating the following command, where the scope reflects the "id" of the Azure Resource Group created in the previous step:

```
az ad sp create-for-rbac --role="Contributor" --
scopes="/subscriptions/0a407898-c077-442d-0000-
0000000/resourceGroups/2ndWinAKSRG"
```



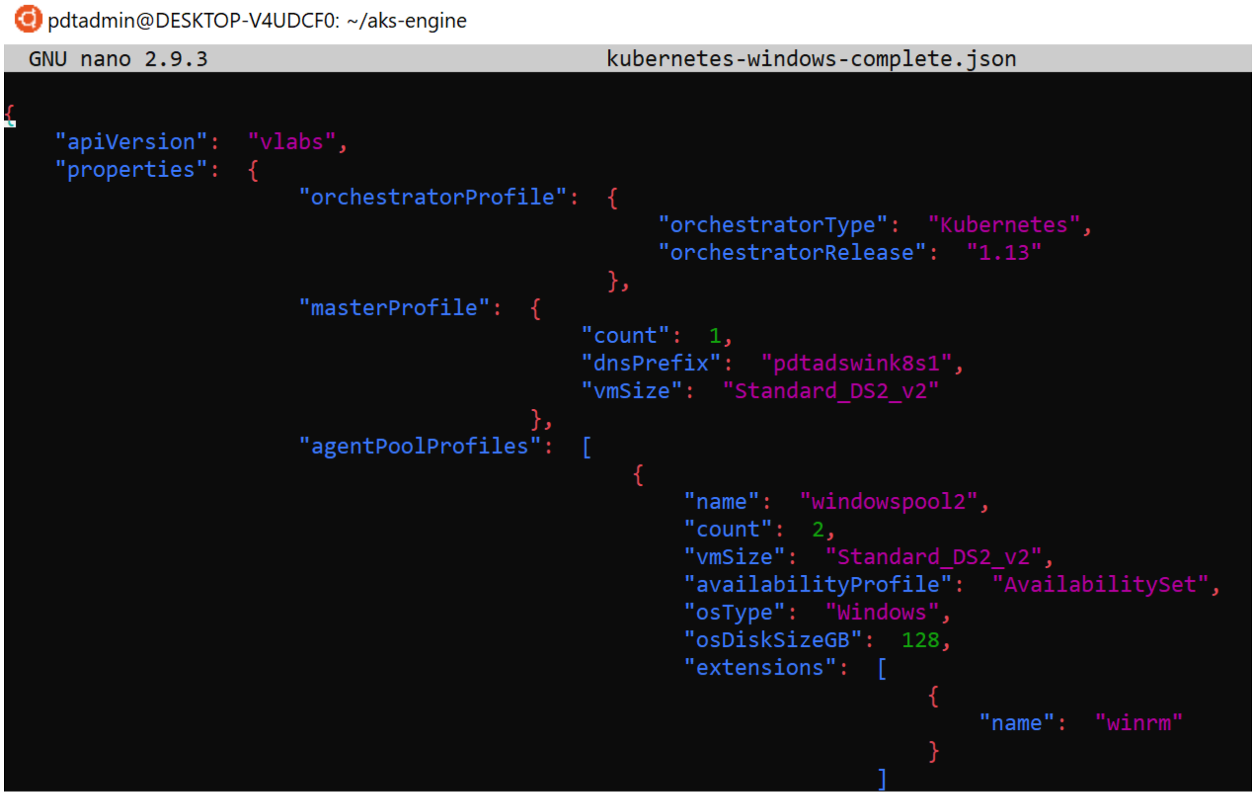
```
pdtadmin@DESKTOP-V4UDCF0:~/aks-engine$ az ad sp create-for-rbac --role="Contributor" --scopes="/subscriptions/0a407898-
077-442d[redacted]/resourceGroups/2ndWinAKSRG"
Retrying role assignment creation: 1/36
Retrying role assignment creation: 2/36
{
  "appId": "dae76072-f4d3-4eb3-8ad8-009[redacted]",
  "displayName": "azure-cli-2019-05-03-19-56-34",
  "name": "http://azure-cli-2019-05-03-19-56-34",
  "password": "0506a1f1-db7d-4b6b-a[redacted]8",
  "tenant": "70681eb4-8dbc-4dc2-[redacted]"
}
pdtadmin@DESKTOP-V4UDCF0:~/aks-engine$ _
```

5. **Take Note** of the **appId**, as well as the **password**, as we need that information later on.

6. The next steps involves creating our aks-engine deployment JSON file. You can grab several examples from the aks-engine Repo on GitHub, in the examples subfolder. I have used the **Kubernetes-windows-complete.json** as a starting point.

7. Run the following command to initiate this JSON file on your client:

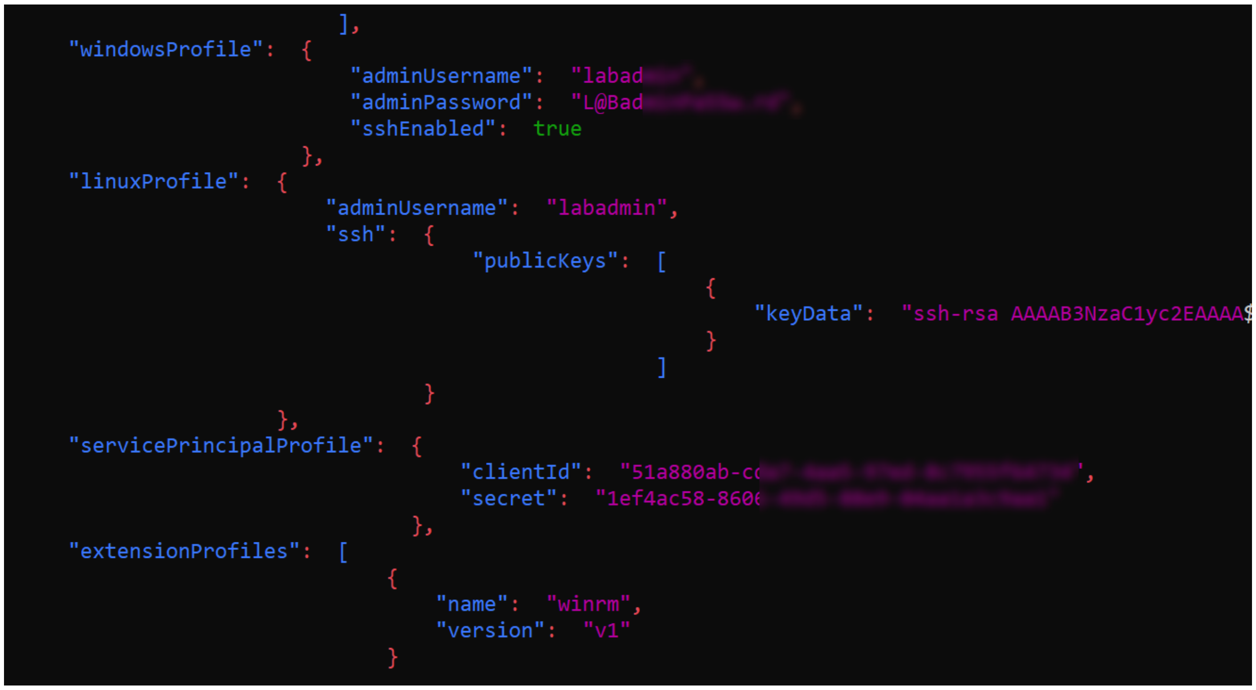
   `nano Kubernetes-windows-complete.json`



```
pdtadmin@DESKTOP-V4UDCF0: ~/aks-engine
  GNU nano 2.9.3                               kubernetes-windows-complete.json

{
    "apiVersion":  "vlabs",
    "properties":  {
                        "orchestratorProfile":  {
                                                     "orchestratorType":  "Kubernetes",
                                                     "orchestratorRelease":  "1.13"
                                                 },
                        "masterProfile":  {
                                              "count":  1,
                                              "dnsPrefix":  "pdtadswink8s1",
                                              "vmSize":  "Standard_DS2_v2"
                                          },
                        "agentPoolProfiles":  [
                                                  {
                                                      "name":  "windowspool2",
                                                      "count":  2,
                                                      "vmSize":  "Standard_DS2_v2",
                                                      "availabilityProfile":  "AvailabilitySet",
                                                      "osType":  "Windows",
                                                      "osDiskSizeGB":  128,
                                                      "extensions":  [
                                                                         {
                                                                             "name":  "winrm"
                                                                         }
                                                                     ]
```

8. Similar to an Azure Resource Manager (ARM) template file, or Terraform alike scenario, you define the technical building blocks and characteristics of the Windows AKS Cluster infrastructure resources. **Feel free to modify settings like vmSize, dnsPrefix and**

agentpoolProfiles name to your situation. Besides those, you also need to provide credentials for the Windows VMs, as well as providing the ServicePrincipalProfile information, clientid and secret, based on the results from the previous steps.

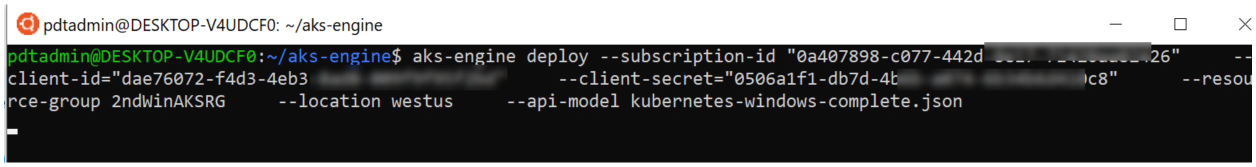


```
                        ],
    "windowsProfile":  {
                          "adminUsername":  "labad[redacted]
                          "adminPassword":  "L@Bad[redacted]
                          "sshEnabled":  true
                      },
    "linuxProfile":  {
                        "adminUsername":  "labadmin",
                        "ssh":  {
                                    "publicKeys":  [
                                                       {
                                                           "keyData":  "ssh-rsa AAAAB3NzaC1yc2EAAAA$
                                                       }
                                                   ]
                                }
                    },
    "servicePrincipalProfile":  {
                                    "clientId":  "51a880ab-cc[redacted]
                                    "secret":  "1ef4ac58-860[redacted]
                                },
    "extensionProfiles":  [
                              {
                                  "name":  "winrm",
                                  "version":  "v1"
                              }
```

9. When done, save the file and close nano editor.

10. Finally, we run the aks-engine deploy step, providing some additional parameters for the deployment.

```
aks-engine deploy --subscription-id "0a407898-c077-442d-8e17-
700000000000000" \
    --client-id="dae76072-f4d3-4eb3-8ad8-000000000" \
    --client-secret="0506a1f1-db7d-4b6b-a074-0000000000" \
    --resource-group 2ndWinAKSRG \
    --location westus \
    --api-model kubernetes-windows-complete.json
```

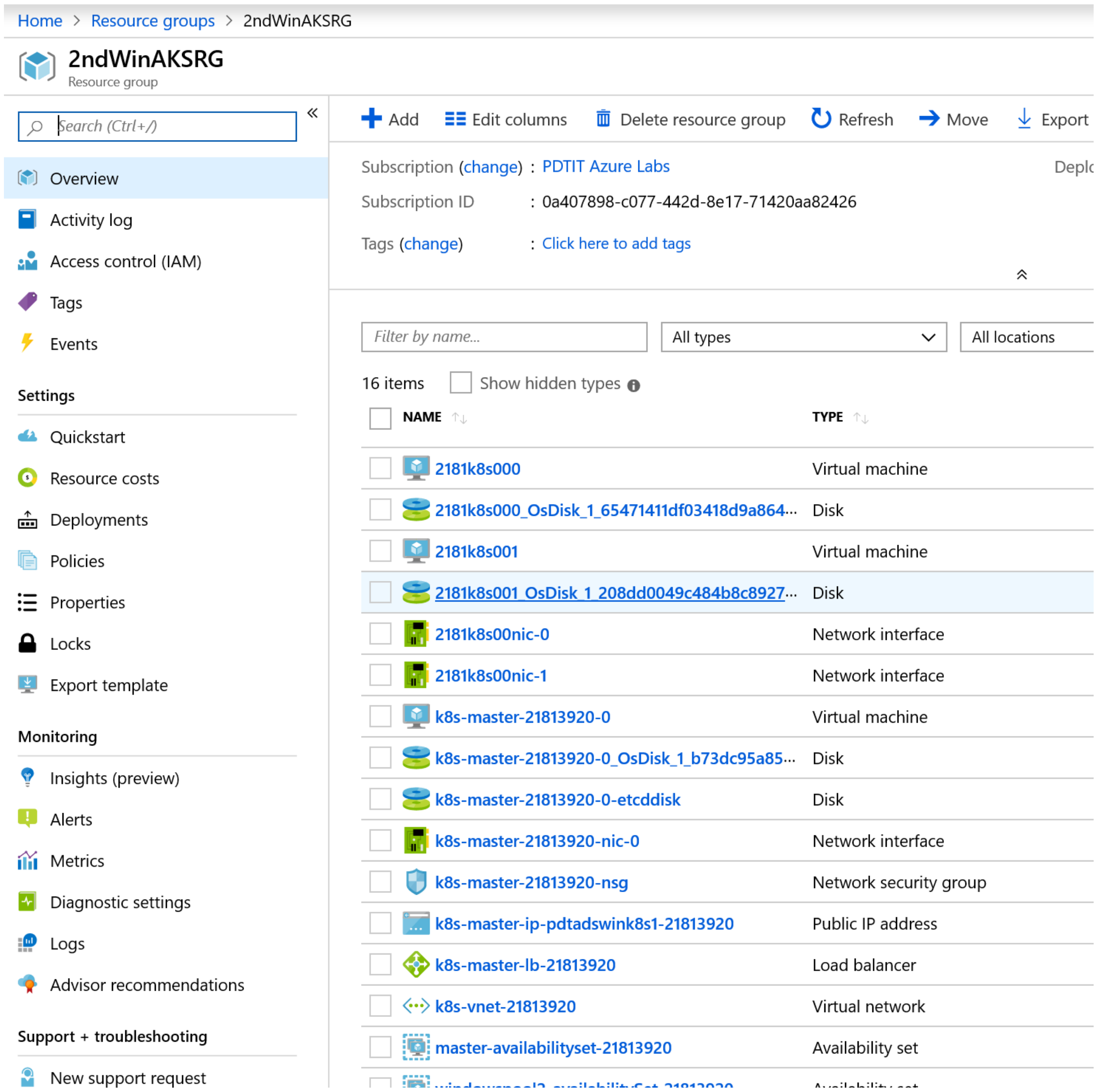


pdtadmin@DESKTOP-V4UDCF0: ~/aks-engine

```
pdtadmin@DESKTOP-V4UDCF0:~/aks-engine$ aks-engine deploy --subscription-id "0a407898-c077-442d[redacted]26"     --
client-id="dae76072-f4d3-4eb3[redacted]      --client-secret="0506a1f1-db7d-4b[redacted]c8"      --resou
rce-group 2ndWinAKSRG     --location westus     --api-model kubernetes-windows-complete.json
```

11. After a few seconds, you get a notification message, stating the deployment started, and will take some time.

12. From here, you can check back in the Azure Portal, under the 2ndWinAKSRG Resource Group, how the deployment is creating all the necessary Kubernetes IAAS resources.



Home > Resource groups > 2ndWinAKSRG

2ndWinAKSRG
Resource group

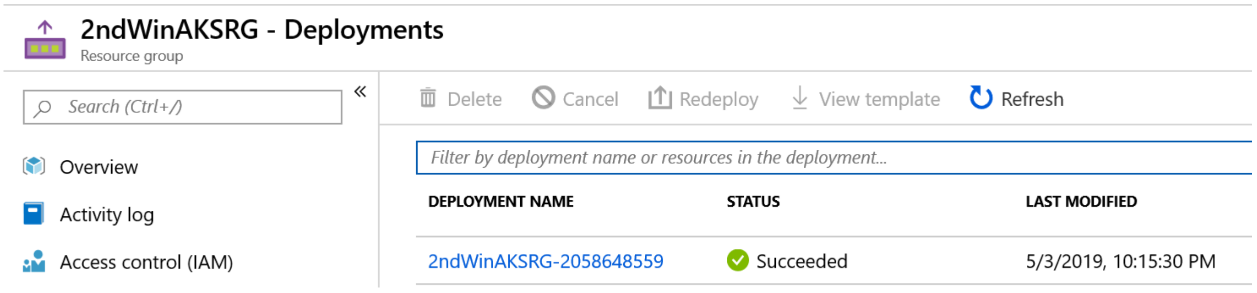
Subscription (change) : PDTIT Azure Labs
Subscription ID : 0a407898-c077-442d-8e17-71420aa82426
Tags (change) : Click here to add tags

Filter by name... | All types | All locations

16 items Show hidden types

NAME	TYPE
2181k8s000	Virtual machine
2181k8s000_OsDisk_1_65471411df03418d9a864...	Disk
2181k8s001	Virtual machine
2181k8s001_OsDisk_1_208dd0049c484b8c8927...	Disk
2181k8s00nic-0	Network interface
2181k8s00nic-1	Network interface
k8s-master-21813920-0	Virtual machine
k8s-master-21813920-0_OsDisk_1_b73dc95a85...	Disk
k8s-master-21813920-0-etcddisk	Disk
k8s-master-21813920-nic-0	Network interface
k8s-master-21813920-nsg	Network security group
k8s-master-ip-pdtadswink8s1-21813920	Public IP address
k8s-master-lb-21813920	Load balancer
k8s-vnet-21813920	Virtual network
master-availabilityset-21813920	Availability set

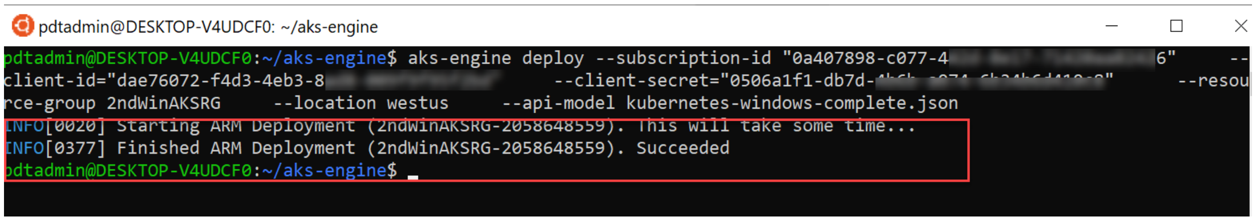
13. If you select **Deployments** under the **Settings** pane of the Resource Group, you can follow the actual deployment, like a typical ARM deployment.



DEPLOYMENT NAME	STATUS	LAST MODIFIED
2ndWinAKSRG-2058648559	✓ Succeeded	5/3/2019, 10:15:30 PM

14. Wait for the deployment to have the status succeeded.

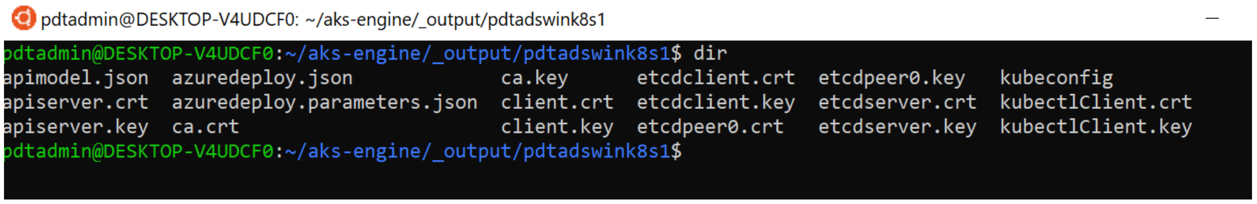
15. When connecting back to the management client, you will also see notification information there:



```
pdtadmin@DESKTOP-V4UDCF0:~/aks-engine$ aks-engine deploy --subscription-id "0a407898-c077-4[redacted]6"     --
client-id="dae76072-f4d3-4eb3-8[redacted]      --client-secret="0506a1f1-db7d-[redacted]"     --resou
rce-group 2ndWinAKSRG     --location westus     --api-model kubernetes-windows-complete.json
INFO[0020] Starting ARM Deployment (2ndWinAKSRG-2058648559). This will take some time...
INFO[0377] Finished ARM Deployment (2ndWinAKSRG-2058648559). Succeeded
pdtadmin@DESKTOP-V4UDCF0:~/aks-engine$ _
```

16. Compared to the Linux-based AKS Service, deploying both the AKS Cluster resource in a Resource Group, as well as the MC_Resource Group containing the actual cluster Infrastructure componentns, aks-engine deployment doesn't work in that way. It creates all the resources in the Azure Resource Group you define. But it does not create the actual Azure Kubernetes Service Resource itself. That's hopefully changing in an update soon enough.

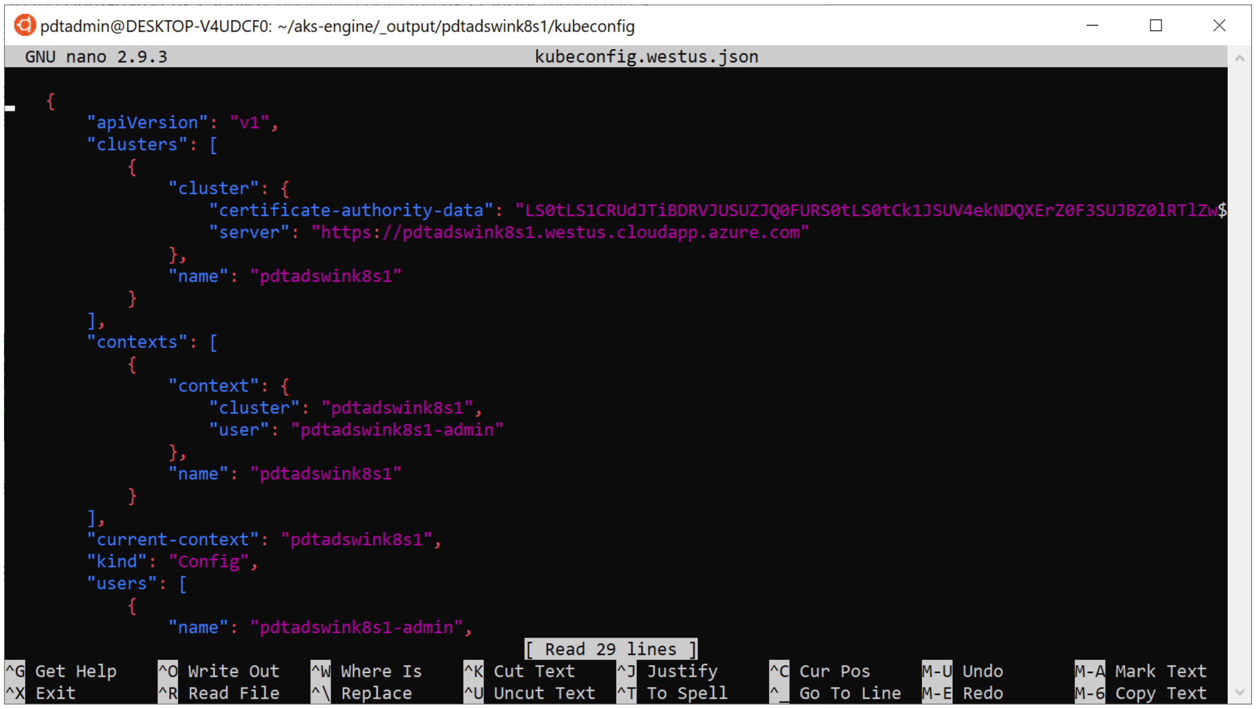
17. In order to manage the AKS cluster, we have to run all commands from our management client. Starting with reading out the details of our AKS Cluster object. During the aks-engine deployment cycle, several cluster-related files are getting created in an subfolder _output in the directory from where you ran the deployment itself.


pdtadmin@DESKTOP-V4UDCF0: ~/aks-engine/_output/pdtadswink8s1

```
pdtadmin@DESKTOP-V4UDCF0:~/aks-engine/_output/pdtadswink8s1$ dir
apimodel.json  azuredeploy.json             ca.key      etcdclient.crt  etcdpeer0.key   kubeconfig
apiserver.crt  azuredeploy.parameters.json  client.crt  etcdclient.key  etcdserver.crt  kubectlClient.crt
apiserver.key  ca.crt                       client.key  etcdpeer0.crt   etcdserver.key  kubectlClient.key
pdtadmin@DESKTOP-V4UDCF0:~/aks-engine/_output/pdtadswink8s1$
```

18. To see these details, browse to _output/<aksclustername>/kubeconfig folder. Here, run the following command to see the parameter information of the actual deployment:

```
nano kubeconfig.westus.json
```



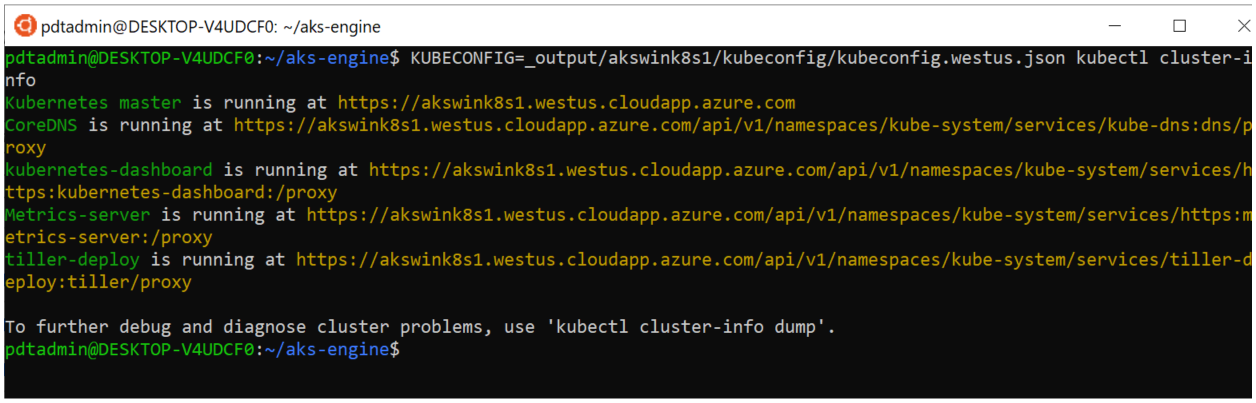
```
pdtadmin@DESKTOP-V4UDCF0: ~/aks-engine/_output/pdtadswink8s1/kubeconfig
 GNU nano 2.9.3                              kubeconfig.westus.json

    {
        "apiVersion": "v1",
        "clusters": [
            {
                "cluster": {
                    "certificate-authority-data": "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUV4ekNDQXErZ0F3SUJBZ0lRTlZw$
                    "server": "https://pdtadswink8s1.westus.cloudapp.azure.com"
                },
                "name": "pdtadswink8s1"
            }
        ],
        "contexts": [
            {
                "context": {
                    "cluster": "pdtadswink8s1",
                    "user": "pdtadswink8s1-admin"
                },
                "name": "pdtadswink8s1"
            }
        ],
        "current-context": "pdtadswink8s1",
        "kind": "Config",
        "users": [
            {
                "name": "pdtadswink8s1-admin",
                                           [ Read 29 lines ]
^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo       M-A Mark Text
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Spell    ^_ Go To Line  M-E Redo       M-6 Copy Text
```

It is the details of this kubeconfig file we need to authenticate to our AKS Cluster in Azure.

19. Close nano editor.

20. Run the following command to read the information of the deployed AKS Cluster:

KUBECONFIG=_output/pdtadswink8s1/kubeconfig/kubeconfig.westus.json kubectl cluster-info



```
pdtadmin@DESKTOP-V4UDCF0: ~/aks-engine
pdtadmin@DESKTOP-V4UDCF0:~/aks-engine$ KUBECONFIG=_output/akswink8s1/kubeconfig/kubeconfig.westus.json kubectl cluster-i
nfo
Kubernetes master is running at https://akswink8s1.westus.cloudapp.azure.com
CoreDNS is running at https://akswink8s1.westus.cloudapp.azure.com/api/v1/namespaces/kube-system/services/kube-dns:dns/p
roxy
kubernetes-dashboard is running at https://akswink8s1.westus.cloudapp.azure.com/api/v1/namespaces/kube-system/services/h
ttps:kubernetes-dashboard:/proxy
Metrics-server is running at https://akswink8s1.westus.cloudapp.azure.com/api/v1/namespaces/kube-system/services/https:m
etrics-server:/proxy
tiller-deploy is running at https://akswink8s1.westus.cloudapp.azure.com/api/v1/namespaces/kube-system/services/tiller-d
eploy:tiller/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
pdtadmin@DESKTOP-V4UDCF0:~/aks-engine$
```

21. This confirms connectivity with the Win AKS Cluster.

Task 3: Pushing a Windows container to the Windows AKS Cluster and run it

Now we have our AKS Cluster up-and-running, lets take it to the next step, and actually pushing an application container to it, and run our web application. We will use a sample Microsoft ASP.NET container, from the Microsoft public Container Registry.

1. Start nano Kubernetes.yml

Paste in the following sample code:

Note: this object looks like a screenshot, but you can actually select all lines and copy/paste 😉

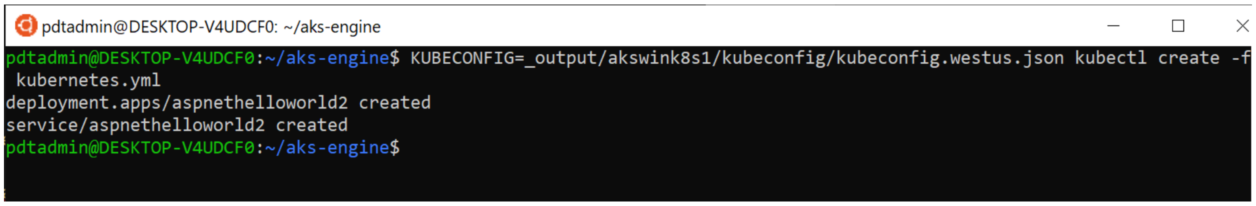
```
apiVersion: apps/v1beta1

kind: Deployment
metadata:
  name: aspnethelloworld2
spec:
  replicas: 1
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1
  minReadySeconds: 5
  template:
    metadata:
      labels:
        app: aspnethelloworld2
    spec:
      containers:
      - name: devopspdt813acr
        image: mcr.microsoft.com/azure-app-
service/samples/aspnethelloworld:latest
        ports:
        - containerPort: 80
      imagePullSecrets:
        - name: acr-auth


---
apiVersion: v1
kind: Service
metadata:
  name: aspnethelloworld2
spec:
  type: LoadBalancer
```

```
ports:
- port: 80
selector:
  app: aspnethelloworld2
```

2. Save the file as Kubernetes.yml and close nano editor

3. Run the following command, allowing the creation of a new Kubernetes Pod, based on the settings in the Kubernetes.yml file.

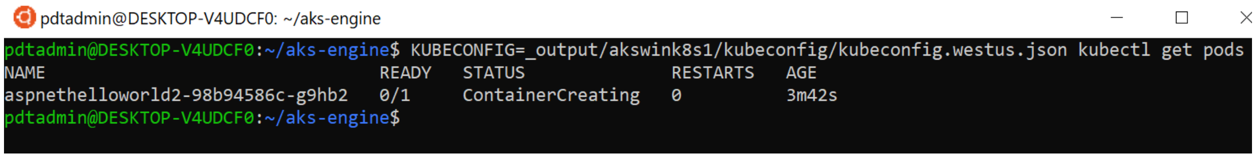
   KUBECONFIG=_output/akswink8s1/kubeconfig/kubeconfig.westus.json kubectl create -f kubernetes.yml



```
pdtadmin@DESKTOP-V4UDCF0: ~/aks-engine
pdtadmin@DESKTOP-V4UDCF0:~/aks-engine$ KUBECONFIG=_output/akswink8s1/kubeconfig/kubeconfig.westus.json kubectl create -f
 kubernetes.yml
deployment.apps/aspnethelloworld2 created
service/aspnethelloworld2 created
pdtadmin@DESKTOP-V4UDCF0:~/aks-engine$
```

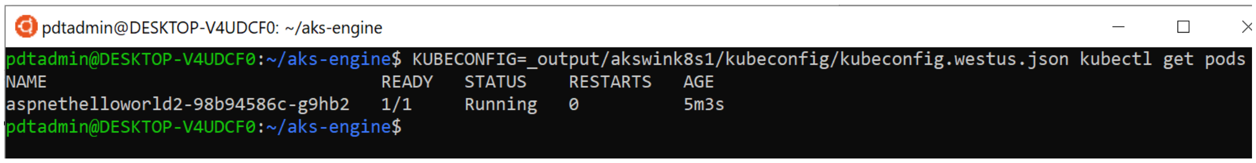
4. To validate the POD is actually getting created, run the following command:

   KUBECONFIG=_output/akswink8s1/kubeconfig/kubeconfig.westus.json kubectl get pods



```
pdtadmin@DESKTOP-V4UDCF0: ~/aks-engine
pdtadmin@DESKTOP-V4UDCF0:~/aks-engine$ KUBECONFIG=_output/akswink8s1/kubeconfig/kubeconfig.westus.json kubectl get pods
NAME                                 READY   STATUS              RESTARTS   AGE
aspnethelloworld2-98b94586c-g9hb2   0/1     ContainerCreating   0          3m42s
pdtadmin@DESKTOP-V4UDCF0:~/aks-engine$
```

The POD has a Status of COntainerCreating, which is positive. Give it another few minutes and check back by relaunching the command.

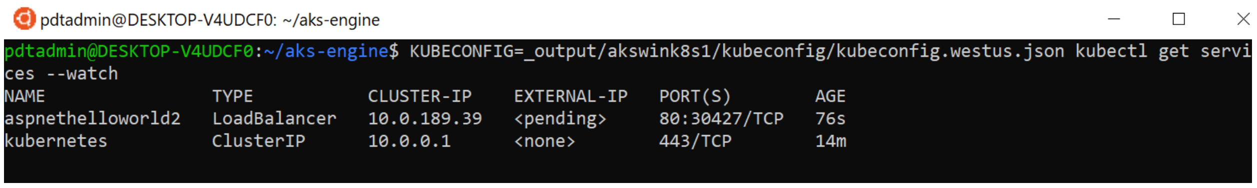


```
pdtadmin@DESKTOP-V4UDCF0: ~/aks-engine
pdtadmin@DESKTOP-V4UDCF0:~/aks-engine$ KUBECONFIG=_output/akswink8s1/kubeconfig/kubeconfig.westus.json kubectl get pods
NAME                                 READY   STATUS    RESTARTS   AGE
aspnethelloworld2-98b94586c-g9hb2   1/1     Running   0          5m3s
pdtadmin@DESKTOP-V4UDCF0:~/aks-engine$
```

Sweet! The POD is running.

5. In order to check if the application workload itself is running, we need to trace the external IP-address of our POD. Run the following command to see this information:

```
KUBECONFIG=_output/akswink8s1/kubeconfig/kubeconfig.westus.json
kubectl get services –watch
```

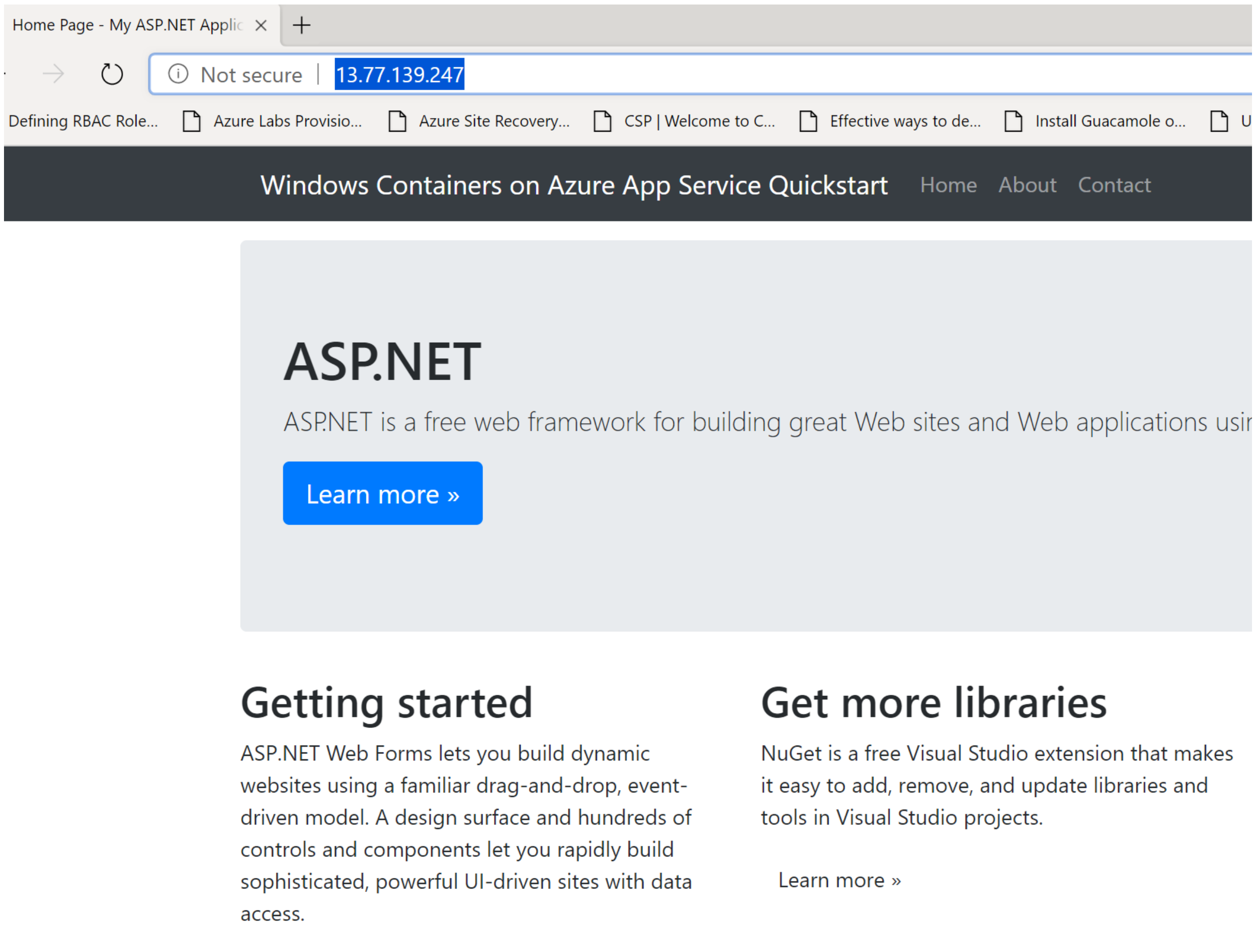


pdtadmin@DESKTOP-V4UDCF0: ~/aks-engine

```
pdtadmin@DESKTOP-V4UDCF0:~/aks-engine$ KUBECONFIG=_output/akswink8s1/kubeconfig/kubeconfig.westus.json kubectl get servi
ces --watch
NAME                TYPE           CLUSTER-IP     EXTERNAL-IP    PORT(S)        AGE
aspnethelloworld2   LoadBalancer   10.0.189.39    <pending>      80:30427/TCP   76s
kubernetes          ClusterIP      10.0.0.1       <none>         443/TCP        14m
```

Although the POD is already running, there is no external-IP attached to it. Wait a few more minutes, until the External-IP is known.

6. Once the external-IP shows up, try connecting to it from your browser: The sample asp.net Windows container is running as expected!



ASP.NET

ASP.NET is a free web framework for building great Web sites and Web applications usi

Getting started

ASP.NET Web Forms lets you build dynamic websites using a familiar drag-and-drop, event-driven model. A design surface and hundreds of controls and components let you rapidly build sophisticated, powerful UI-driven sites with data access.

Get more libraries

NuGet is a free Visual Studio extension that makes it easy to add, remove, and update libraries and tools in Visual Studio projects.

Summary

This completes the lab in which you deployed a Windows AKS Cluster, using aks-engine. You pushed a POD configuration using a Kubernetes.yml file, and validated the POD was running as expected.

Closing

This workshop enabled you 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, you got introduced to the basics of automating Azure resources deployments using Visual Studio and Azure Resource Manager (ARM) templates. Next, you learned about Microsoft SQL database migration to SQL Azure PaaS, as well as deploying and migrating Azure Web Apps.

After having covered these foundational platform components and app as well as database transformation to the Azure public cloud, the workshop continued with a focus on the core concepts and advantages of using different container services, available in Azure today:
- 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).

Throughout this workshop, the following labs were performed:

- Lab 1: Deploying a 2-tier Azure Virtual Machine (Webserver and SQL database Server) using ARM-template automation with Visual Studio 2017;
- Lab 2: Migrating a legacy SQL 2012 database to Azure SQL PaaS (Lift & Shift);
- Lab 3: Migrating a legacy ASP.NET web application to Azure Web Apps (Lift & Shift);
- Lab 4: Containerizing a legacy ASP.NET web application using Docker;
- Lab 5: Running Azure Container Instance (ACI) from an Azure Container Registry (ACR) image;
- Lab 6: Deploy and run Azure Container Services (ACS) with Kubernetes;
- Lab 7: Deploy and run Azure Kubernetes Services (AKS);
- Lab 8: Managing and Monitoring Azure Container Services (ACS) and Azure Kubernetes Services (AKS);
- Bonus Lab 9: Deploying a Windows Container AKS Cluster in Azure

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