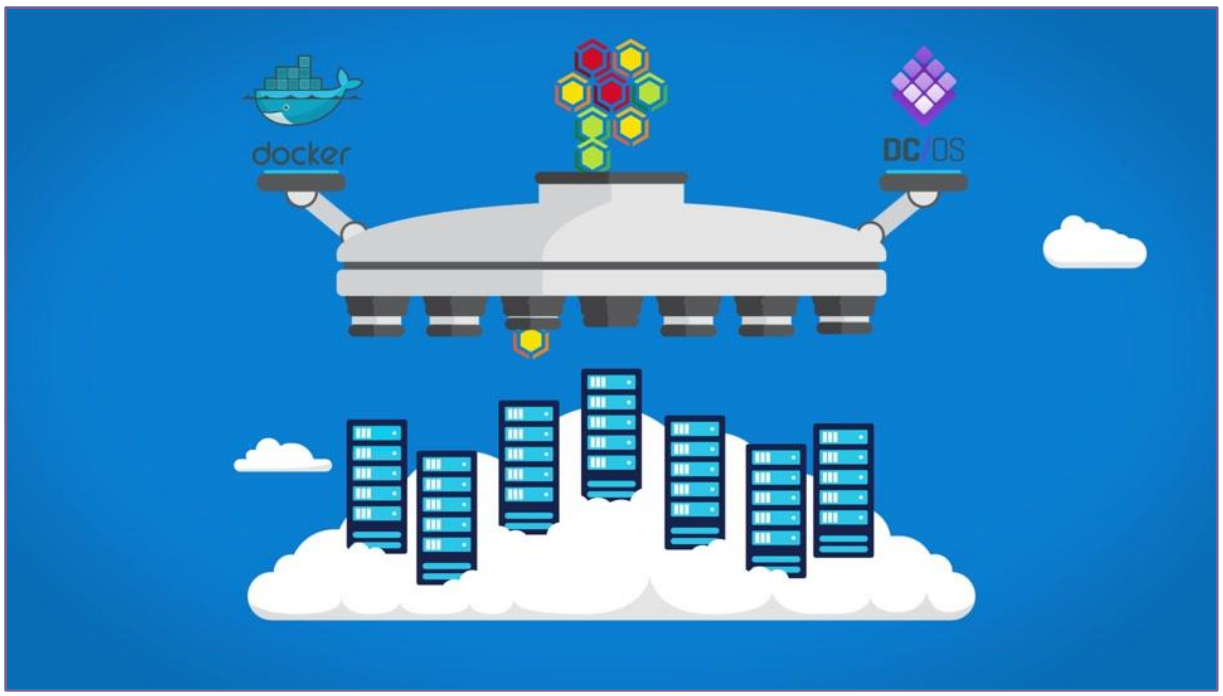


Azure Developer Series

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.

Lab 8: Managing and Monitoring Azure Container Services (ACS) and Azure Kubernetes Services (AKS);

What you will learn

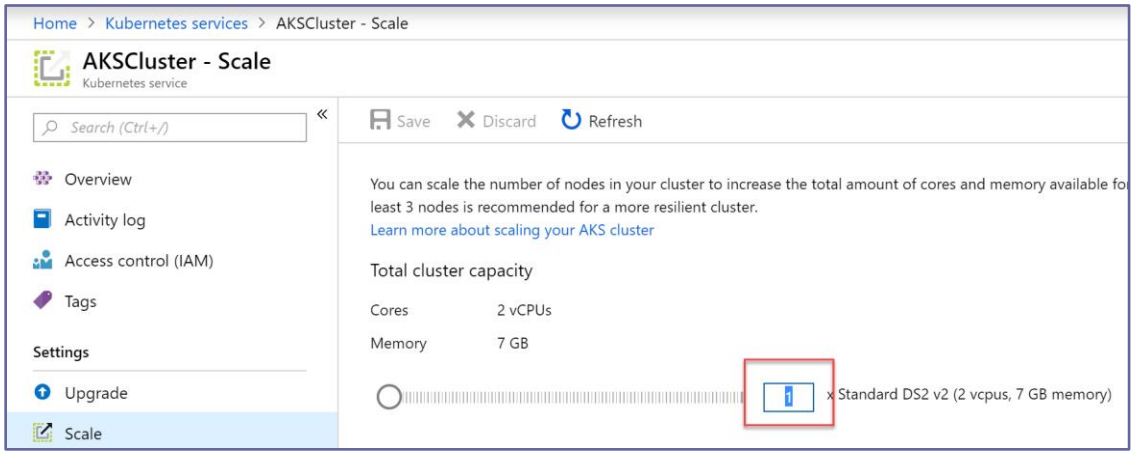
In this last lab of this workshop, we will focus on common operations, related to ACS and AKS. This includes enabling the basics of container scalability within the platform , as well as configuring the Azure built-in monitoring capabilities for these services, using Azure Monitor for ACS and Azure Application Insights for Kubernetes.

Time Estimate

This lab shouldn't take longer than 60 minutes.

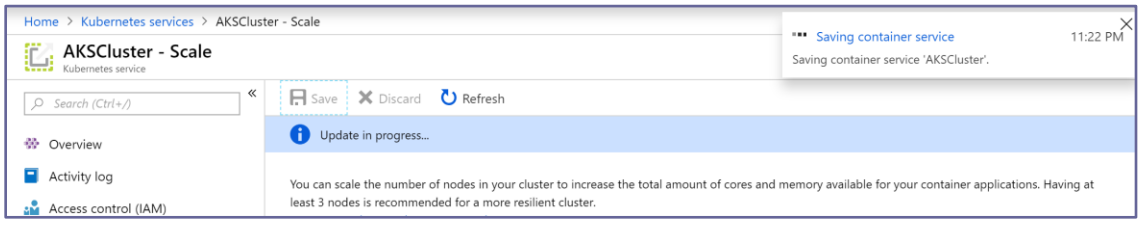
Task 1: Enabling container scalability in Azure Kubernetes Services (AKS)

1. AKS provides some nice integration in the Azure Portal, for example on how to **scale out** your Kubernetes Service. **From the Azure Portal,** browse to your **Azure Kubernetes Service**. In the detailed blade, **go to settings | scale**



2. **Change the single node configuration** to **2**, by updating the number of moving the bar.

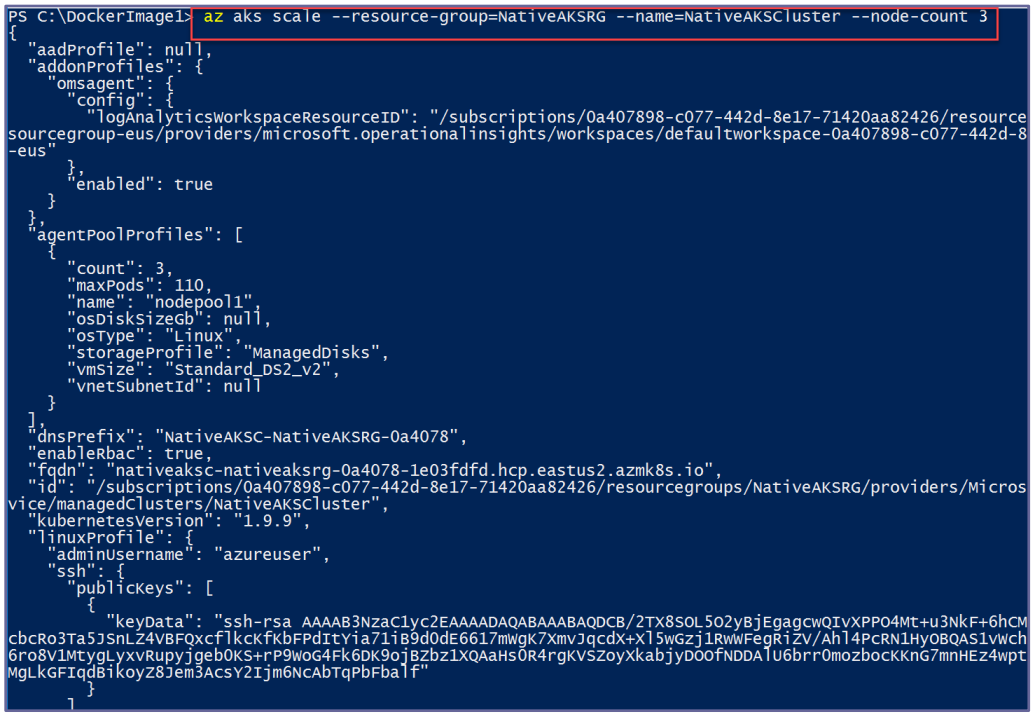
3. **Save** the changes.



4. You can achieve the same by using the kubectl commandline syntax:

```
az aks scale --resource-group=[SUFFIX]AKSRG --name
=[SUFFIX]AKSCluster --node-count 3
```

Note the command takes a couple of minutes to complete, without having impact on the already running pods. The result is published in the JSON output.



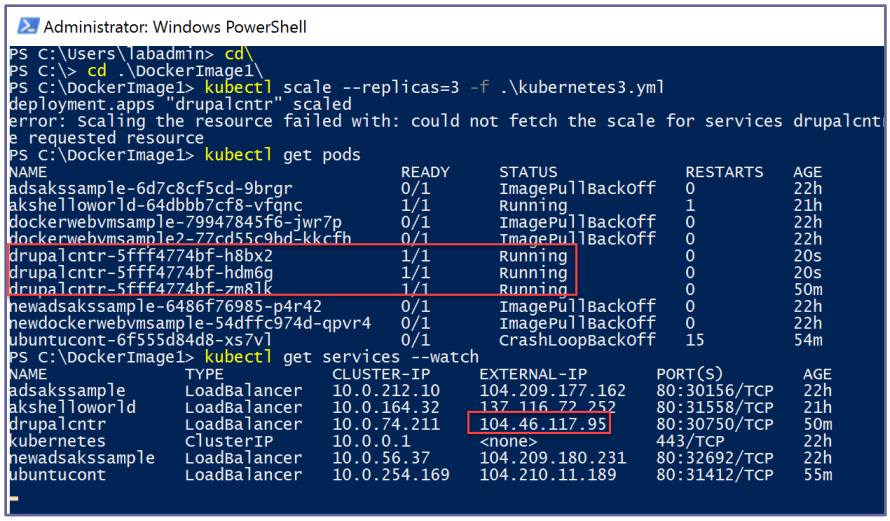
5. Another "scale" option, is not scaling the number of Kubernetes Nodes, but scaling the actual pods. This is done by running the following command:

Note, since this pushes a new deployment, the previous configuration will be overwritten; browsing to the EXTERNAL-IP will show you a new Drupal welcome screen. If you want to keep your existing Drupal configured site, create a copy of the Kubernetes3.yml, and define a new name within the yaml-file itself for the application.

```
kubectl scale --replicas=3 -f .\kubernetes3.yml
```

Which in this scenario spins up 3 instances of the Drupal container.

6. You can validate the creation using kubectl get pods and kubectl get services --watch

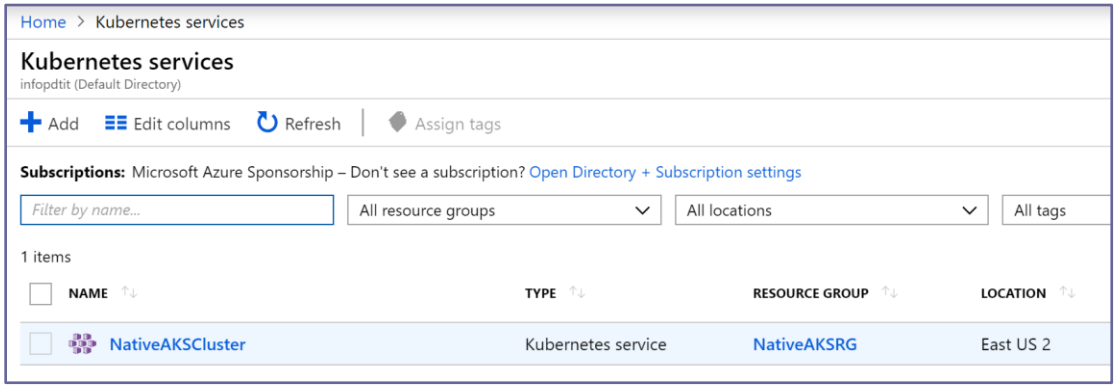


7. This completes the task on learning different scaling methods in AKS.

Task 2: Monitoring Azure Kubernetes Services in Azure

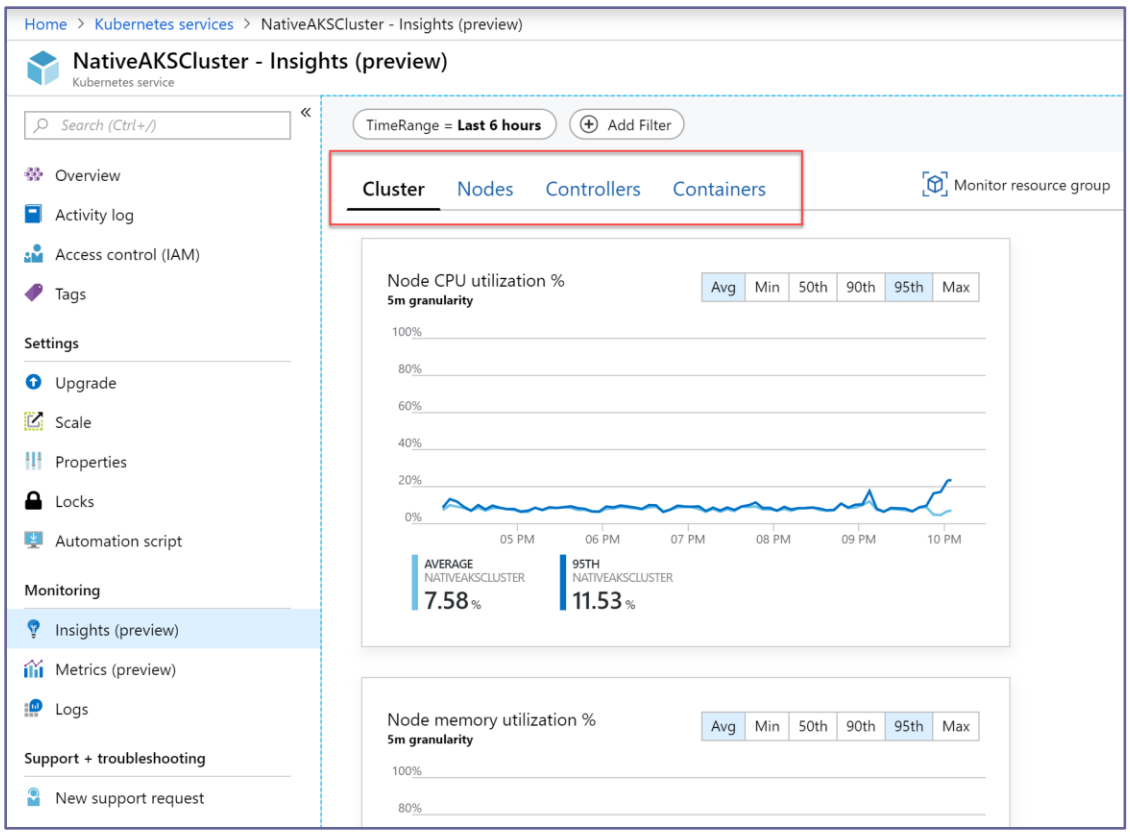
Azure provides a nice integration (Insights) between standard Azure monitoring capabilities and the AKS services.

1. From the Azure Portal, browse to Azure Kubernetes Services, and select your AKS Service.

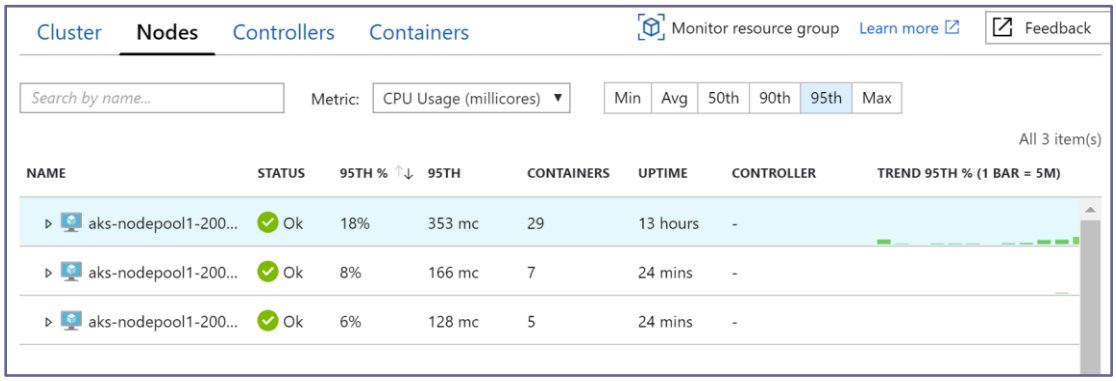


2. Selecting the AKS Cluster Service object, will open the detailed blade for this service. Here, select Monitoring | Insights (Preview)

3. This opens the Container monitoring blade details.

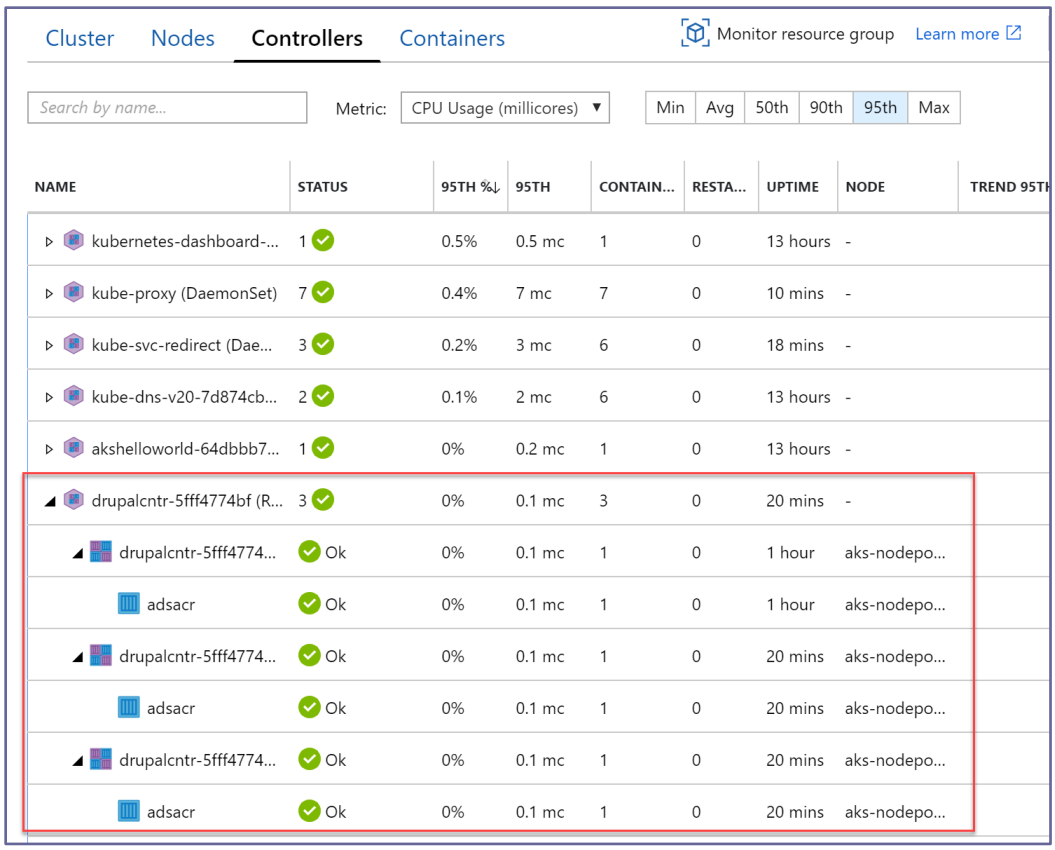


4. On top of the detailed blade, **Select Nodes**. This shows a more detailed view of the different AKS Nodes within that running cluster.



Note: since we are not having a lot of load on your lab-setup, only Node 1 is showing usage

5. Next, **select Controllers** in the top menu. This opens a more detailed view of the running AKS controllers. **Highlight the drupal controller, and open its details.** Here, you can nicely see the 3 scaled pods, with some details on performance for each, as well as uptime/running time.

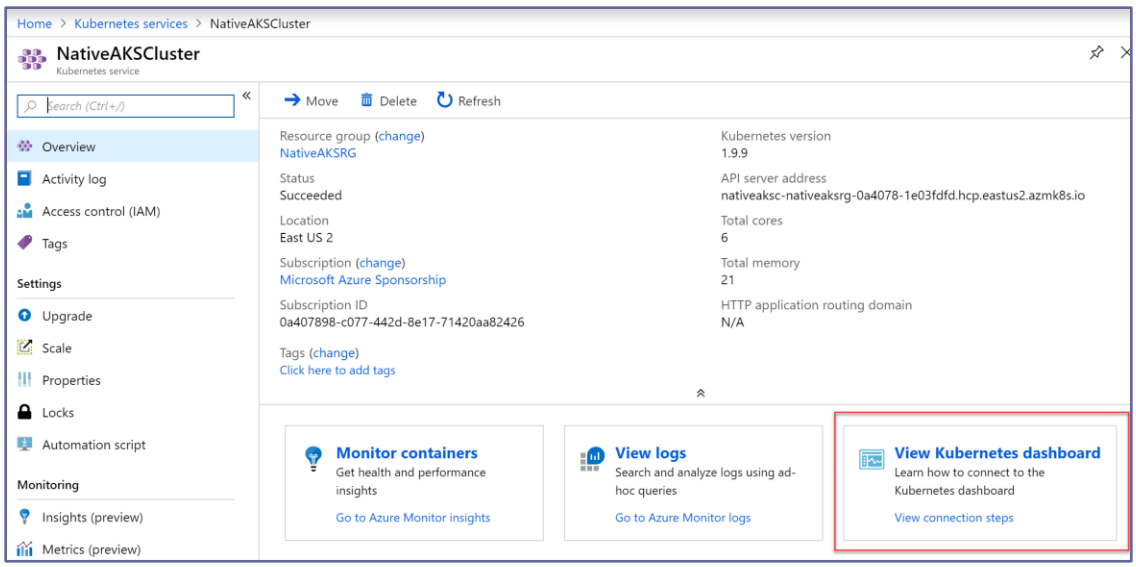


6. This concludes this part of the task.

Task 3: Using the Kubernetes Dashboard in Azure Kubernetes Services

Besides the Azure built-in monitoring tools in the previous task, AKS also provides a "Kubernetes-specific" dashboard.

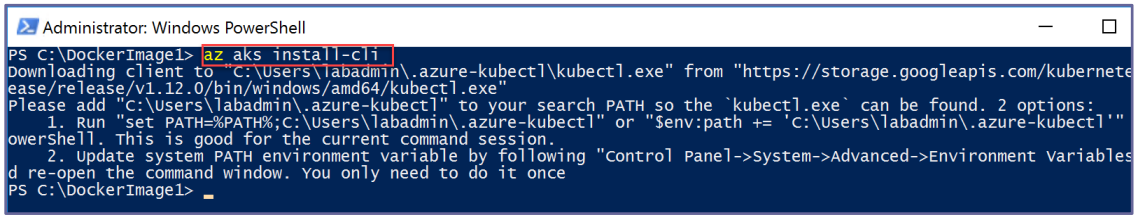
1. From your **Azure Kubernetes Services object | Overview**, notice "**View Kubernetes Dashboard**".



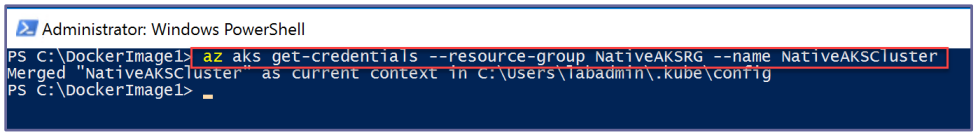
2. This shows you the az aks commands required to run this dashboard. Note that it is starting a built-in Kubernetes dashboard in its own portal, proxied through your localhost IP-address on the client. There is no separate Kubernetes dashboard integration in the Azure Portal.

3. Let's run these commands one by one, in PowerShell on our lab-jumpVM:

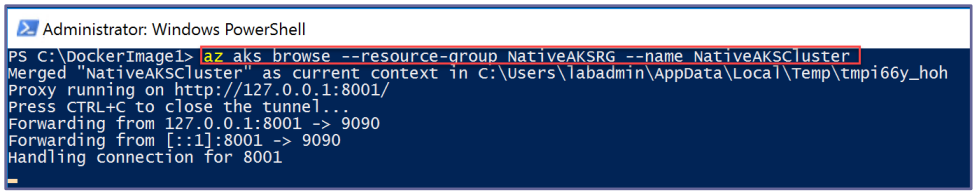
```
az aks install-cli
```



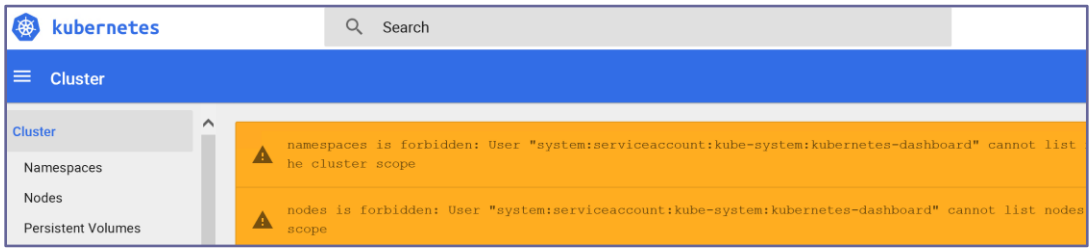

```
az aks get-credentials --resource-group [SUFFIX]AKSRG --name
[SUFFIX]AKSCluster
```




```
az aks browse --resource-group [SUFFIX]AKSRG --name
[SUFFIX]AKSCluster
```



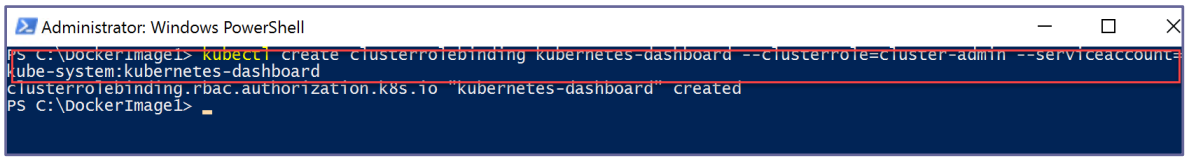
4. This opens your internet browser, and shows the Kubernetes Dashboard.



5. Notice the error messages, saying your serviceaccount cannot list the cluster scope or nodes.

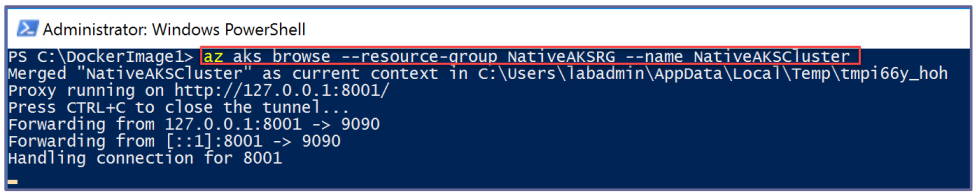
6. This is related to a known "issue" / feature 😊, related to the fact our **Azure Kubernetes Service** is managed by **RBAC**. We need to tell the Kubernetes dashboard built-in service account to "trust/allow" RBAC, by **setting a clusterrolebinding**, using the following command:

```
kubectl create clusterrolebinding kubernetes-dashboard --
clusterrole=cluster-admin --serviceaccount=kube-
system:kubernetes-dashboard
```

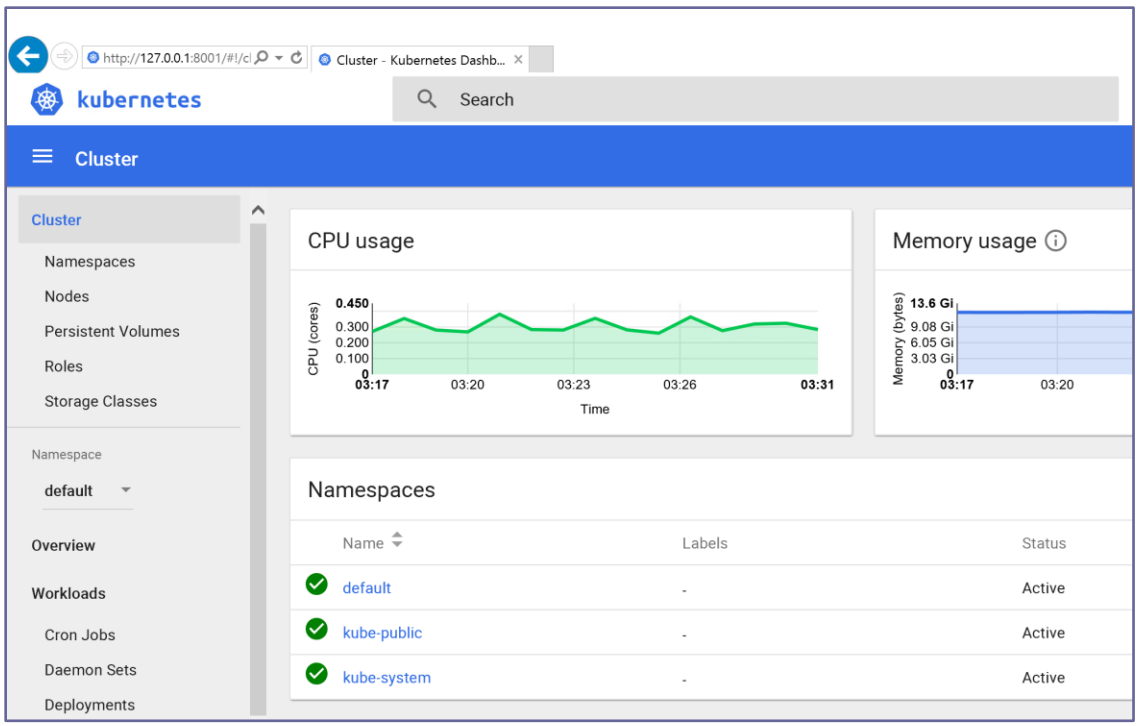


7. If we then run our "az aks browse…" command again:

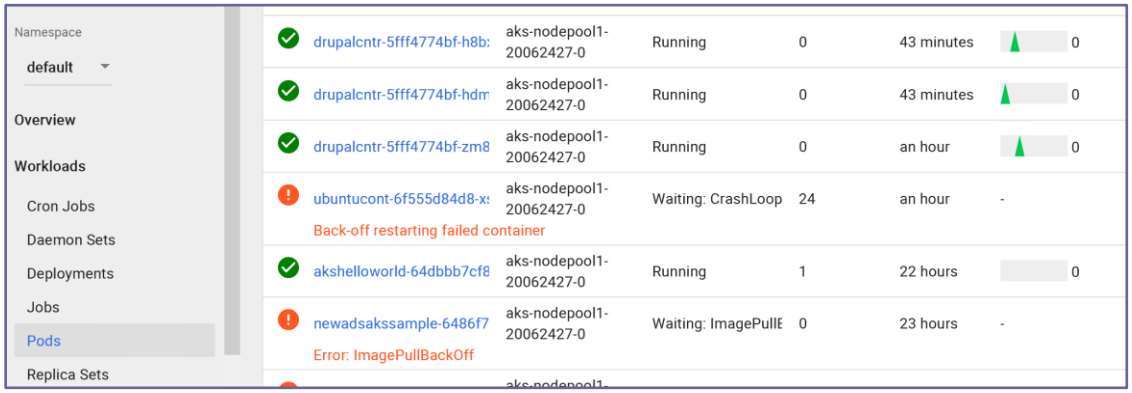
```
az aks browse --resource-group [SUFFIX]AKSRG --name
[SUFFIX]AKSCluster
```



The Kubernetes dashboard will load successfully now:



8. From here, you can click around and get equally detailed views on the AKS Cluster, click-through on Nodes, Pods,... and more detailed status information on each of these resources.



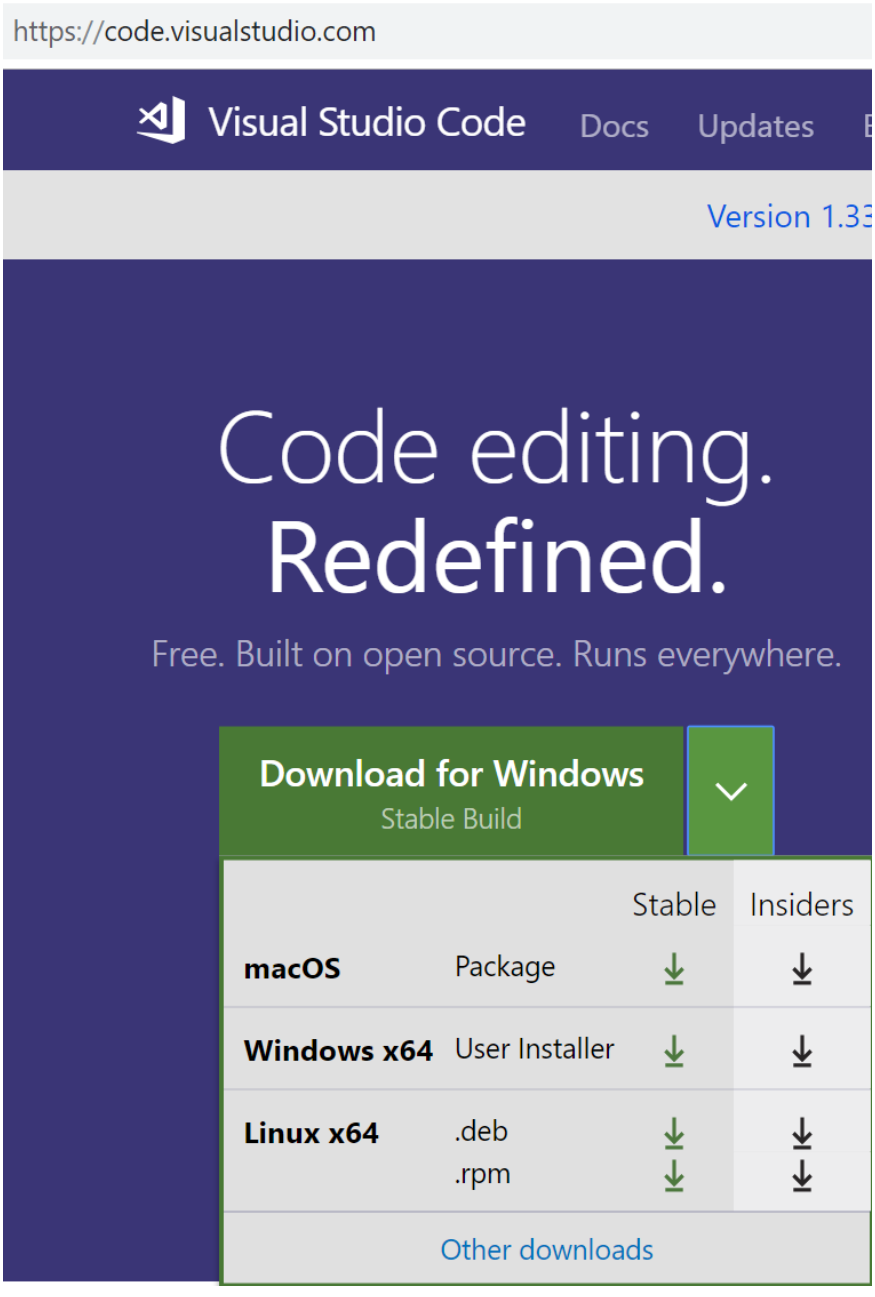
Note: the failed deployments from my testing when building this lab guide are also included, containing a nice descriptive reason for the failed status of each pod.

9. This completes this part of the task.

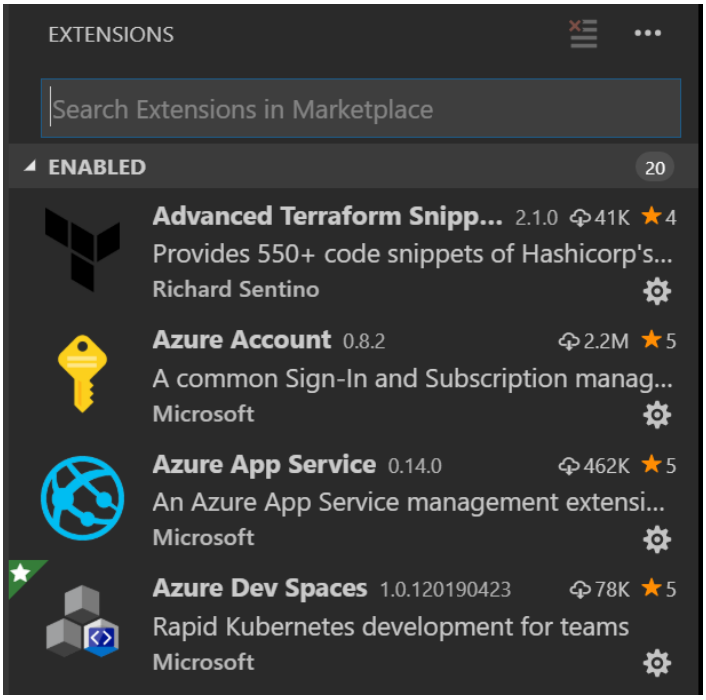
Task 4: Managing Kubernetes from Visual Studio Code

Besides the Azure built-in monitoring tools in the previous task, or the provided "Kubernetes-specific" dashboard, one can also manage the AKS cluster using Visual Studio Code.

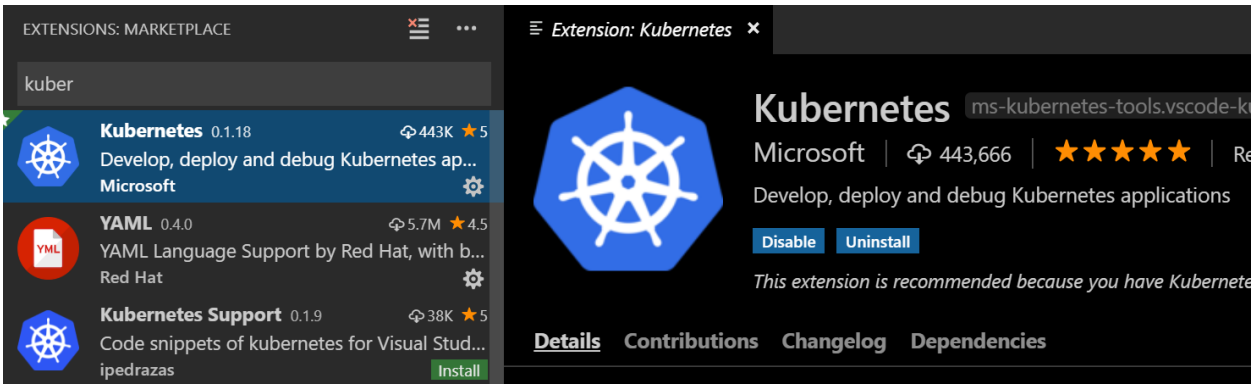
1. If Visual Studio Code is not installed on your machine yet, run the install from code.visualstudio.com



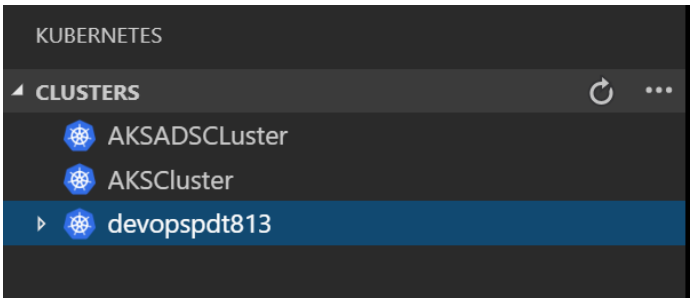
2. Once Visual Studio is installed, from the menu, go to File / Preferences / Extensions. This shows a list of community and 3rd party vendor provided extensions.



3. In the Search Extensions in MarketPlace, type "Kubernetes".



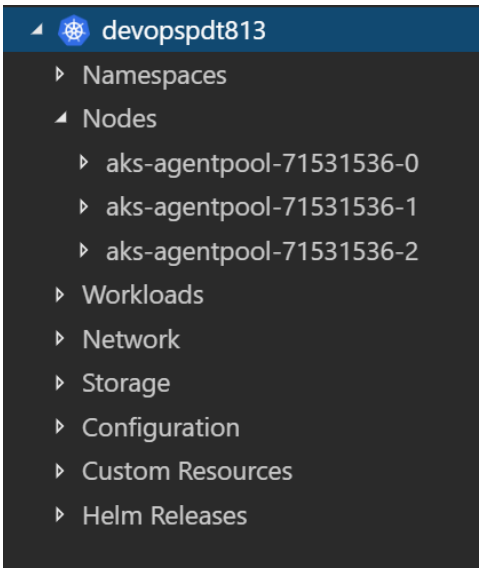
4. Click Install and wait for the extension to get installed successfully. You will see a shortcut to it in the left menu sidebar. Click on it. Out of your Azure subscription ID and Azure admin account credentials, it will list all Kubernetes clusters it recognizes.



5. Note: if no AKS cluster is showing up here, Open PowerShell or Azure CLI, and run the following commands:
   - az account login (this authenticates your session with Azure)
   - az aks get-credentials --resource-group <your AKS RG> --name <your AKS Cluster Name>

```
PS C:\DockerImage1> az aks get-credentials --resource-group devopspdt813-rg --name devopspdt813
Merged "devopspdt813" as current context in C:\Users\PeterDeTender\.kube\config
```

6. Refresh the Visual Studio Code window, by select Kubernetes again. This time, your AKS Cluster should show up fine.



7. You can browse through the different core Kubernetes cluster components, like Nodes, Storage, Configuration and more.

Summary

In this lab, you learned the basic admin tasks about scaling Azure Kubernetes Services, using the Azure Portal and Kubectl command line. Next, you became familiar with the built-in AKS monitoring solutions out of Kubernetes Insights, as well as how to deploy and use the standard Kubernetes Dashboard.

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);

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