

Please use Azure Portal with your personal email and not organization specific email

Whenever any string is in <> it means you need to provide the proper value at that location.
Exp <VM Name> means enter actual value without <>

Exercise: Azure Virtual Machine Creation - Using Azure Portal

1. Login to Azure Portal, click on Virtual Machines Blade and click on Add
2. Create a new Resource Group, select the subscription and location for VM

Create a virtual machine

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ

Visual Studio Enterprise

Resource group * ⓘ

(New) RG-VM

Create new

Instance details

Virtual machine name * ⓘ

MyWin10VM

Region * ⓘ

(Asia Pacific) Southeast Asia

Availability options ⓘ

No infrastructure redundancy required

Image * ⓘ

Windows 10 Pro, Version 1809

Browse all public and private images

Size * ⓘ

Standard B2ms

2 vcpus, 8 GiB memory

Change size

Click here for available locations

3. Create a simple VM with Windows 10 OS with the size B2ms. Provide the user name and password.

Administrator account

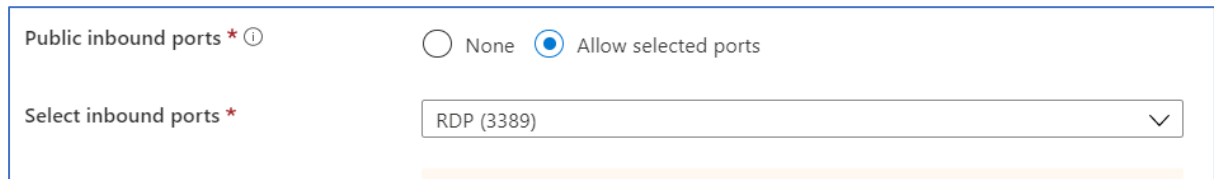
Username * ⓘ

Gouri

Password * ⓘ

Confirm password * ⓘ

4. Make sure that RDP is selected so that we can connect to VM later



Public inbound ports * ⓘ ☐ None ☒ Allow selected ports

Select inbound ports * RDP (3389) ▼

5. In Advanced tab find out how extensions can be installed for VM
6. After successful creation of VM, connect using remote desktop

Note: you can create a virtual network, assign nsg to it and use the same when creating a VM. Advantage will be that the VM will automatically have the set rules from nsg.

Exercise: Azure Virtual Machine Creation - Using Azure PowerShell

Use New-AzureRmVm

You can provide parameters when prompted or can use them while using the command (in this case many default settings will be considered)

New-AzureRmVm

- ResourceGroupName <RGName>
- Name <VM Name>
- Location "East US"
- VirtualNetworkName <vNet Name>
- SubnetName "default"
- OpenPorts 80,3389

You can also declare the variables first and use them while giving the command

Exercise: Azure Virtual Machine Creation - Using Visual Studio

1. Start Visual Studio and create a console application
2. Add NuGet package to Microsoft.Azure.Management.Fluent
3. Add azureauth.properties with following
 - subscription=<subscription-id>
 - client=<application-id>
 - key=<authentication-key>
 - tenant=<tenant-id>
 - managementURI=https://management.core.windows.net/
 - baseURL=https://management.azure.com/
 - authURL=https://login.windows.net/
 - graphURL=https://graph.windows.net/

In order to get values for client create an application with Azure active directory blade and selecting App Registration. You will get client id and tenant id from here.

Home > SSGS IT Educon Services - App registrations > MyAppForVMCreation

MyAppForVMCreation

Search (Ctrl+/) << Delete Endpoints

Get a second? We would love your feedback on Microsoft identity platform (previous)

Display name : MyAppForVMCreation

Application (client) ID :

Directory (tenant) ID :

Object ID :

Overview

Quickstart

Manage

Branding

Authentication

Create a secret for the application and copy the value and that becomes your key

4. Add environment variable by using following PowerShell
[Environment]::SetEnvironmentVariable("AZURE_AUTH_LOCATION", "C:\Visual Studio 2019\Projects\myDotnetProject\myDotnetProject\azureauth.properties", "User")

5. You can use following code

```
var loc = Environment.GetEnvironmentVariable("AZURE_AUTH_LOCATION");
var credentials =
SdkContext.AzureCredentialsFactory.FromFile(Environment.GetEnvironmentVariable("AZURE_AUTH_LOCATION"));
```

```
var azure =
azure.Configure().WithLogLevel(HttpLoggingDelegatingHandler.Level.Basic).Authenticate(credentials).WithDefaultSubscription();
```

```
var groupName = "RG-VMFromCode";
var vmName = "mySimpleVM";
var location = Region.USCentral;

Console.WriteLine("Creating resource group...");
var resourceGroup = azure.ResourceGroups.Define(groupName)
.WithRegion(location).Create();

Console.WriteLine("Creating availability set...");
var availabilitySet = azure.AvailabilitySets.Define("myAVSet")
.WithRegion(location).WithExistingResourceGroup(groupName)
.WithSku(AvailabilitySetSkuTypes.Classic).Create();
Console.WriteLine("Creating public IP address...");
var publicIPAddress =
azure.PublicIPAddresses.Define("myPublicIP").WithRegion(location)
.WithExistingResourceGroup(groupName).WithDynamicIP().Create();
Console.WriteLine("Creating virtual network...");
var network = azure.Networks.Define("myVNet")
.WithRegion(location).WithExistingResourceGroup(groupName)
.WithAddressSpace("10.0.0.0/16")
.WithSubnet("mySubnet", "10.0.0.0/24").Create();

Console.WriteLine("Creating network interface...");
var networkInterface = azure.NetworkInterfaces.Define("myNIC")
.WithRegion(location)
```

```
.WithExistingResourceGroup(groupName)
.WithExistingPrimaryNetwork(network)
.WithSubnet("mySubnet")
.WithPrimaryPrivateIPAddressDynamic()
.WithExistingPrimaryPublicIPAddress(publicIPAddress)
.Create();
```

```
Console.WriteLine("Creating virtual machine...");
azure.VirtualMachines.Define(vmName)
    .WithRegion(location)
    .WithExistingResourceGroup(groupName)
    .WithExistingPrimaryNetworkInterface(networkInterface)
    .WithLatestWindowsImage("MicrosoftWindowsServer", "WindowsServer", "2012-R2-Datacenter")
    .WithAdminUsername("<username>").WithAdminPassword("<password>")
    .WithSize(VirtualMachineSizeTypes.StandardDS1).Create();
```

6. If you find that the file cannot be read with the help of environmental variable use actual path to read the file
7. If you get authority related error use another method for authentication as follows
`var credentials = SdkContext.AzureCredentialsFactory.FromServicePrincipal(clientId, clientSecret, tenantId, AzureEnvironment.AzureGlobalCloud);`
Provide appropriate values for parameters
8. It will take quite some time to create the Virtual Machine. Connect to VM and see that its working properly. In order to avoid using resources ensure that you delete the whole Resource Groups at the end

Exercise: ARM Template Creation - Using Azure Portal

1. Sign in to Azure Portal

2. Click on Create a Resource – Storage – Storage Account

Create storage account

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *

Visual Studio Enterprise

Resource group *

RG-StorageAc-ARM

[Create new](#)

Instance details

The default deployment model is Resource Manager, which supports the latest Azure features. You may choose to deploy using the classic deployment model instead. [Choose classic deployment model](#)

Storage account name * ⓘ

mystoragegss

✓

Location *

(US) Central US

▼

Performance ⓘ

☒ Standard ☐ Premium

Account kind ⓘ

StorageV2 (general purpose v2)

▼

Replication ⓘ

Locally-redundant storage (LRS)

▼

Access tier (default) ⓘ

☐ Cool ☒ Hot

Review + create

< Previous

Next : Networking >

Provide resource group name, storage account name and other details.

- Click on Review + Create, click on Create after successful validation
- Click on Download template from Export Template Blade
- Look at the parameter created
- Extract json files at some location
- Create another resource for template deployment and select build your own template in editor
- Let us change the parameter and create another resource
- Click on Load file and load the template downloaded earlier
- We will pick up the value for the storage account using a variable. In order to do that remove the parameter declared and add a variable called storageAccountName
- Change the name for resource of account name as
"storageAccountName": "[concat(uniquestring(resourceGroup().id), 'standardsa')]"
change the value for parameter wherever it appears
- Save the template, create resource after successful validation
- Click on Purchase after entering values for resource group and location
- Ensure that the resource gets created
- Remove unwanted resource groups and clean up resources

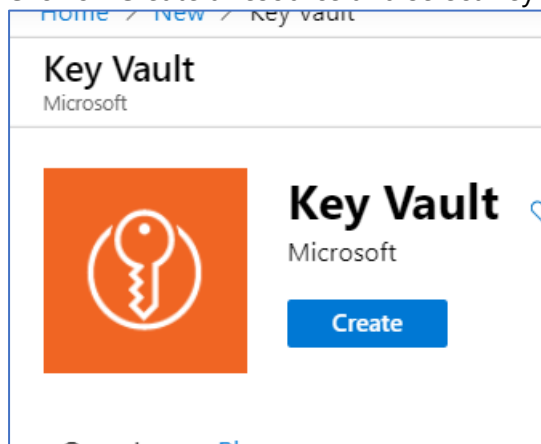
Exercise: ARM Template Creation - Using Visual Studio Code

1. Open Visual Studio Code
2. Open Extensions Pane and search for azure manager resource tools and Install
3. File - Open File - azureDeploy.json (this file is available in lab documents folder)
4. Deploy using Azure Cloud Shell
5. <https://shell.azure.com>
6. Open Bash - Upload
7. Run following commands
8.

```
echo "Enter the Resource Group name:" &&  
read resourceGroupName &&  
echo "Enter the name for this deployment:" &&  
read deploymentName &&  
echo "Enter the location (i.e. centralus):" &&  
read location &&  
az group create --name $resourceGroupName --location $location && az group  
deployment create --name $deploymentName --resource-group $resourceGroupName -  
-template-file "azureDeploy.json"
```
9. Delete resource group after ensuring that the resource got created

Exercise: Create Azure Key Vault

1. Login to Azure Portal
2. Click on Create a resource and select Key Vault



3. Provide details for resource group name, name for key vault, location and click on Review + Create. After successful validation click on Create
4. Once the resource is created add a key to it Select Keys – Generate/Import, provide name and click on enter.
5. We will be using this key vault with key in next exercise

Exercise: Azure Virtual Machine Decryption

Pre-requisite: existing Virtual Machine, key vault with 1 key in it(depending on if your VM and Key Vault are in same resource group or not you need to provide names in the commands)

1. Login to Azure Portal

2. Start Cloud Shell
3.

```
$keyVaultName = "<key vault name>"
$rgName="<Resource Group Name>"
$keyVault = Get-AzKeyVault -VaultName $keyVaultName -ResourceGroupName
$rgName;
$diskEncryptionKeyVaultUrl = $keyVault.VaultUri;
$keyVaultResourceId = $keyVault.ResourceId;
$keyEncryptionKeyUrl = (Get-AzKeyVaultKey -VaultName $keyVaultName -Name <key
name>).Key.kid;
```

4. Write-Host \$keyVault


```
Set-AzVMDiskEncryptionExtension -ResourceGroupName $rgName `
-VMName "<VM Name>" `
-DiskEncryptionKeyVaultUrl $diskEncryptionKeyVaultUrl `
-DiskEncryptionKeyId $keyVaultResourceId `
-KeyEncryptionKeyUrl $keyEncryptionKeyUrl `
-KeyEncryptionKeyId $keyVaultResourceId
```

5. This is a time-consuming process and will take at least 10-15 minutes to complete

RequestId	IsSuccess	Status	Code	Status	Code	Reason	Phrase
-----	-----	-----	-----	-----	-----	-----	-----
		True		OK	OK		

6. You can use


```
az vm encryption show --resource-group "<RG name>" --name "<VM name>"
```

 OR


```
Get-AzureRmVMDiskEncryptionStatus -ResourceGroupName <RG Name> -VMName
<VM Name>
```

 to find
7. The encryption can be reversed by following command


```
Disable-AzureRmVMDiskEncryption -ResourceGroupName <resource-group> -VMName
<VM name>
```

Exercise: Azure Batch Job Creation - Using Azure CLI

1. Create resource group


```
az group create
--name myResourceGroup
--location eastus2
```
2. Create storage account


```
az storage account create \
--resource-group myResourceGroup \
```

- ```
--name mystorageaccount \
--location eastus2 \
--sku Standard_LRS
```
3. Create batch account

```
az batch account create \
--name mybatchaccount \
--storage-account mystorageaccount \
--resource-group myResourceGroup \
--location eastus2
```
  4. Login to batch account

```
az batch account login \
--name mybatchaccount \
--resource-group myResourceGroup \
--shared-key-auth
```
  5. Create pool of compute nodes

```
az batch pool create \
--id mypool --vm-size Standard_A1_v2 \
--target-dedicated-nodes 2 \
--image canonical:ubuntu:16.04-LTS \
--node-agent-sku-id "batch.node.ubuntu 16.04"
```
  6. Find allocation state of pool

```
az batch pool show --pool-id mypool \
--query "allocationState"
```
  7. Create job

```
az batch job create \
--id myjob \
--pool-id mypool
```
  8. Create tasks

```
for ($i=1;$i -le 5;$i++) {az batch task create --task-id mytask$i --job-id myjob --
command-line "/bin/bash -c 'printenv | grep AZ_BATCH; sleep 90s'" }
```
  9. View task status

```
az batch task show \
--job-id myjob \
--task-id mytask1
```
  10. View task output

```
az batch task file list \
--job-id myjob \
--task-id mytask1 \
--output table
```
  11. Download one of the output files

```
az batch task file download \
--job-id myjob \
--task-id mytask1 \

```



```
--file-path stdout.txt \
--destination ./stdout.txt
```

## Exercise: Azure Batch Job Creation - Using Azure Portal

1. Login to Azure Portal
2. Select Create a resource – Compute – Batch Service. Provide subscription, resource group, name and location

### New Batch account

Provide basic Batch account info

running on workstations and clusters today can be readily enabled to run in Azure at scale, and with no on-premises infrastructure required. Common application workloads include image and video rendering, media transcoding, engineering simulations, Monte Carlo simulations, and software test execution, among others; all highly parallel, computationally intensive workloads that can be broken into individual tasks for execution. With Azure Batch, you can scale from a few VMs, up to tens of thousands of VMs, and run the largest, most resource-intensive workloads. [Learn more](#)

#### PROJECT DETAILS

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription \*

Visual Studio Enterprise

Resource group \*

(New) RG-BatchService

[Create new](#)

#### INSTANCE DETAILS

Account name \* ⓘ

mybatchservicegs


.southindia.batch.azure.com

Location \* ⓘ

South India

#### STORAGE ACCOUNT

Specify an optional storage account. For best performance we recommend a storage account (general purpose v2) located in the same region as the associated Batch account. [Select a storage account](#)



Review + create

Previous

Next: Advanced >

3. Click on Review + Create, after successful validation click on Create

4. Go to the resource and select Pools – Add

|                                            |                           |   |
|--------------------------------------------|---------------------------|---|
| Image Type ⓘ                               | Marketplace               | ▼ |
| Enable unverified image                    | <input type="checkbox"/>  |   |
| *Publisher                                 | microsoftwindowsserver    | ▼ |
| *Offer                                     | windowsserver             | ▼ |
| *Sku                                       | 2016-datacenter-smalldisk | ▼ |
| Batch Node Agent SKU ID                    | batch.node.windows amd64  |   |
| Enable automatic updates<br>(Windows only) | <input type="checkbox"/>  |   |

|                                                           |                                        |
|-----------------------------------------------------------|----------------------------------------|
| <b>NODE SIZE</b>                                          |                                        |
| VM size ( <a href="#">View full pricing details</a> ) * ⓘ | Standard A1 (1 vCPUs, 1.8 GB Memory) ▼ |
| <b>SCALE</b>                                              |                                        |
| Mode                                                      | <b>Fixed</b> Auto scale                |
| Target dedicated nodes ⓘ                                  | 2 ✓                                    |
| Target low-priority nodes ⓘ                               | 0                                      |
| Total target vCPUs: 2                                     |                                        |
| Resize timeout ⓘ                                          | 10 ✓                                   |
| minutes                                                   |                                        |

5. After a few minutes state of pool will become steady

|        |        |   |   |             |          |     |
|--------|--------|---|---|-------------|----------|-----|
| myPool | 0 -> 2 | 0 | 0 | standard_a1 | Resizing | ... |
| myPool | 2      | 0 | 2 | standard_a1 | Steady   |     |

- 6.

7. Create a job and assign it to existing pool

**BASIC INFORMATION**

Job ID \* ⓘ

myJob ✓

Pool

\*Pool ⓘ  
myPool >

**JOB MANAGER, PREPARATION AND RELEASE TASKS**

Mode


None Custom

**ADVANCED SETTINGS**

Mode

Default Custom

8. Create Task – Add – provide id and enter cmd /c "set AZ\_BATCH & timeout /t 90 > NUL" in command line, click on Submit by keeping all default settings
9. Click Files on node and select the file stdout.txt

 **stdout.txt**  
myTask

[Download](#) [Delete](#) [Refresh](#)

|                                                                                                                                           |                                       |
|-------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------|
| File name                                                                                                                                 | Creation time                         |
| stdout.txt                                                                                                                                | Thursday, 28 November, 2019, 17:09:31 |
| URL                                                                                                                                       | Last modified                         |
| <a href="https://myservicebatch.southindia.batch.azure.com/jobs/myJ...">https://myservicebatch.southindia.batch.azure.com/jobs/myJ...</a> | Thursday, 28 November, 2019, 17:09:42 |
| Content type                                                                                                                              | Size                                  |
| text/plain                                                                                                                                | 715 Bytes                             |

```
AZ_BATCH_ACCOUNT_URL=https://myservicebatch.southindia.batch.azure.com/
AZ_BATCH_POOL_ID=myPool
AZ_BATCH_NODE_ID=tvmps_79ea0b81e98f481a5511eb2e5713d724b1718524a86f6b704abcf2763e851aef_d
AZ_BATCH_ACCOUNT_NAME=myservicebatch
AZ_BATCH_NODE_ROOT_DIR=D:\batch\tasks
AZ_BATCH_NODE_SHARED_DIR=D:\batch\tasks\shared
AZ_BATCH_NODE_STARTUP_DIR=D:\batch\tasks\startup
AZ_BATCH_NODE_IS_DEDICATED=true
AZ_BATCH_NODE_MOUNTS_DIR=D:\batch\tasks\fsmounts
AZ_BATCH_JOB_ID=myJob
AZ_BATCH_TASK_ID=myTask
AZ_BATCH_TASK_DIR=D:\batch\tasks\workitems\myJob\job-1\myTask
AZ_BATCH_TASK_WORKING_DIR=D:\batch\tasks\workitems\myJob\job-1\myTask\wd
AZ_BATCH_TASK_USER_IDENTITY=PoolNonAdmin
AZ_BATCH_TASK_USER=PoolNonAdmin23187785
```

## Exercise: Azure Kubernetes Service (AKS) – Creation -Using Azure Portal

1. Login to Azure Portal
2. Create Azure Kubernetes Cluster (AKS Cluster) – this has VMs in the form of nodes  
Click on Create a resource – Kubernetes Service

### Create Kubernetes cluster

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription \* ⓘ

Visual Studio Enterprise

Resource group \* ⓘ

(New) RG-DemoAKS

Create new

#### Cluster details

Kubernetes cluster name \* ⓘ

AKSDemo

Region \* ⓘ

(US) Central US

Kubernetes version \* ⓘ

1.13.12 (default)

DNS name prefix \* ⓘ

AKSDemo-dns

#### Primary node pool

The number and size of nodes in the primary node pool in your cluster. For production workloads, at least 3 nodes are recommended for resiliency. For development or test workloads, only one node is required. You will not be able to change the node size after cluster creation, but you will be able to change the number of nodes in your cluster after creation. If you would like additional node pools, you will need to enable the "X" feature on the "Scale" tab which will allow you to add more node pools after creating the cluster. [Learn more about node pools in Azure Kubernetes Service](#)

Node size \* ⓘ

Standard DS2 v2

Change size

Node count \* ⓘ

1

Review + create

< Previous

Next : Scale >

This in turn will create service principal which gets created in Azure AD (used to grant permission to access resources)

Home > Research > Get Started > Kubernetes Service > Create Kubernetes cluster

Create Kubernetes cluster

BasicsScaleAuthenticationNetworkingMonitoringTagsReview + create

The **cluster infrastructure** service principal is used by the Kubernetes cluster to manage cloud resources attached to the cluster. [Learn more about service principals in AKS](#)

**Kubernetes authentication and authorization** is used by the Kubernetes cluster to control user access to the cluster as well as what the user may do once authenticated. [Learn more about Kubernetes authentication](#)

CLUSTER INFRASTRUCTURE

\* Service principal ⓘ

(new) default service principal

[Configure service principal](#)

KUBERNETES AUTHENTICATION AND AUTHORIZATION

Enable RBAC ⓘ

NoYes

You can check it by going to Azure AD and selecting App registrations blade. I can see mine as follows

|    |                                          |
|----|------------------------------------------|
| AK | <a href="#">AKSDemoSP-20191128170246</a> |
|----|------------------------------------------|

3. Create Azure Container Registry (Container Registry) in the same resource group. Provide unique name and click on Create

Registry name \*

myaksdemogs ✓

.azurecr.io

Subscription \*

Visual Studio Enterprise ▼

Resource group \*

RG-DemoAKS ▼

[Create new](#)

Location \*

Central US ▼

Admin user \* ⓘ

Enable

Disable

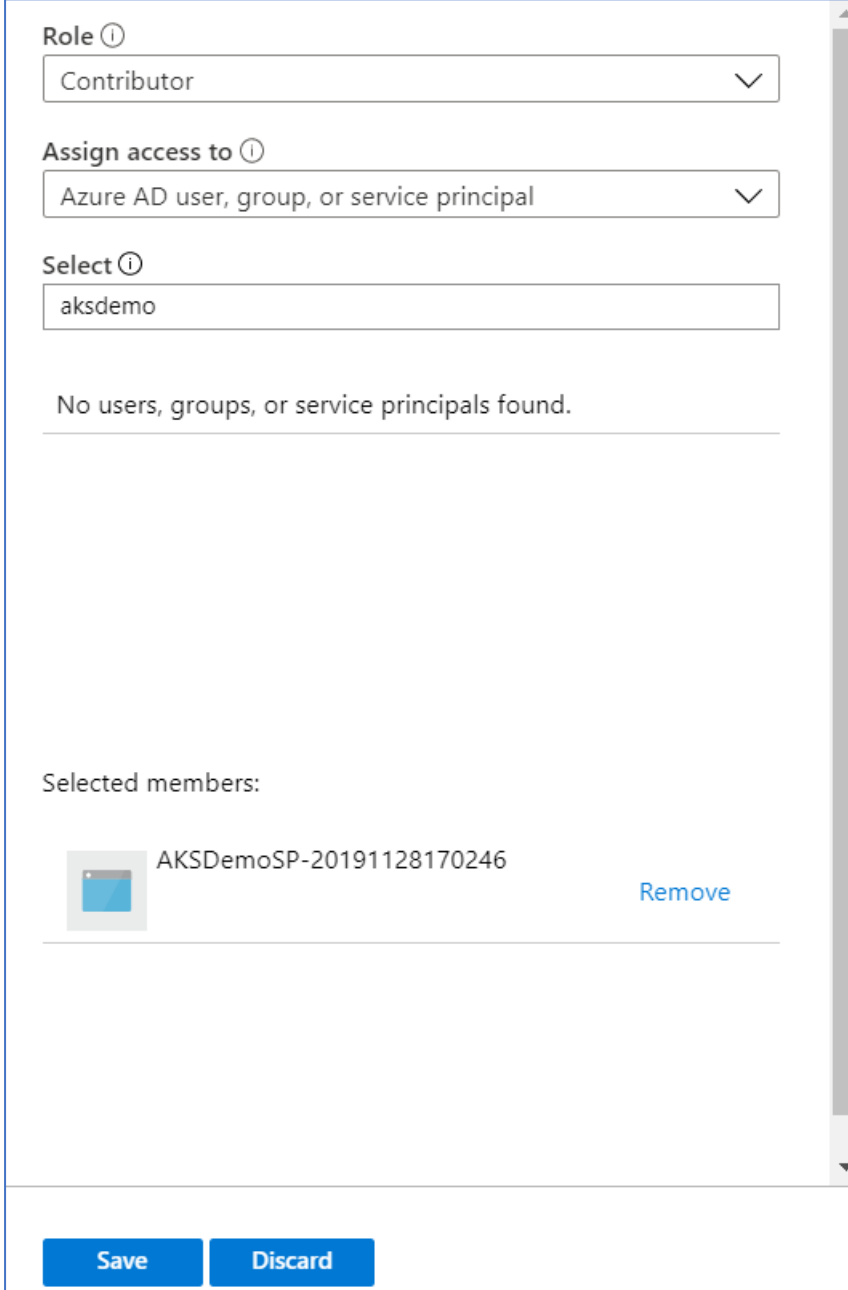
SKU \* ⓘ

Standard ▼

Create

Automation options

4. We need to edit Access Control of ACR to add Service Principal of AKS cluster with the role of contributor (to push and pull images)



The screenshot shows the 'Assign access to' section of the Azure portal. It includes three dropdown menus: 'Role' set to 'Contributor', 'Assign access to' set to 'Azure AD user, group, or service principal', and 'Select' set to 'aksdemo'. Below these is a message 'No users, groups, or service principals found.' and a 'Selected members:' section containing one entry: 'AKSDemoSP-20191128170246' with a 'Remove' link. At the bottom are 'Save' and 'Discard' buttons.


Role ⓘ  
Contributor

Assign access to ⓘ  
Azure AD user, group, or service principal

Select ⓘ  
aksdemo

No users, groups, or service principals found.

Selected members:

 AKSDemoSP-20191128170246 [Remove](#)

[Save](#) [Discard](#)

5. Now that Azure resources are ready, let us create a Team Project in Azure DevOps as follows (create a new account using <https://dev.azure.com> – use same email used to Azure to create this account)

## Create new project



Project name \*

aksfordemo



Description

### Visibility



Public ⓘ

Anyone on the internet can view the project. Certain features like TFVC are not supported.



Private

Only people you give access to will be able to view this project.



Public projects are disabled for your organization. You can turn on public visibility with [organization policies](#).

### ^ Advanced

Version control ⓘ

Git



Work item process ⓘ

Agile

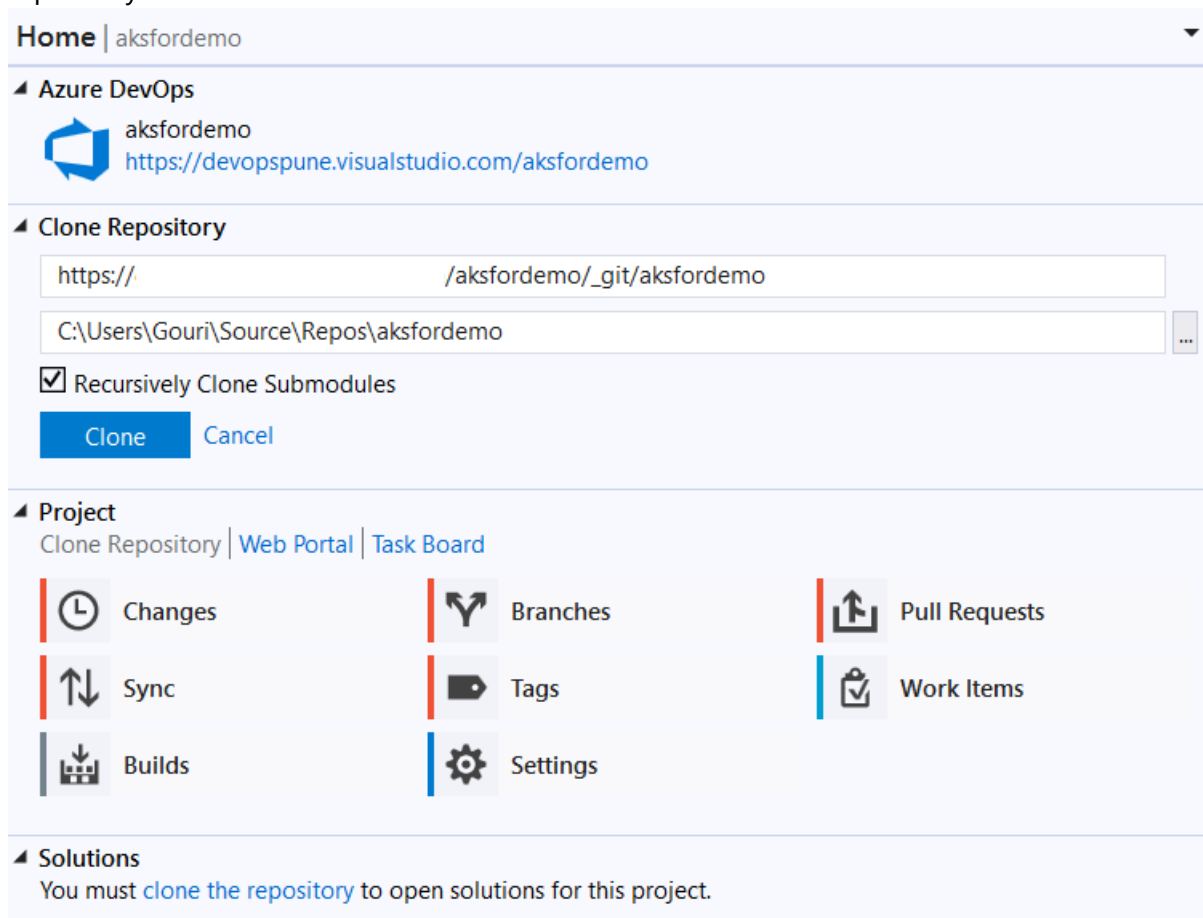


Cancel

Create

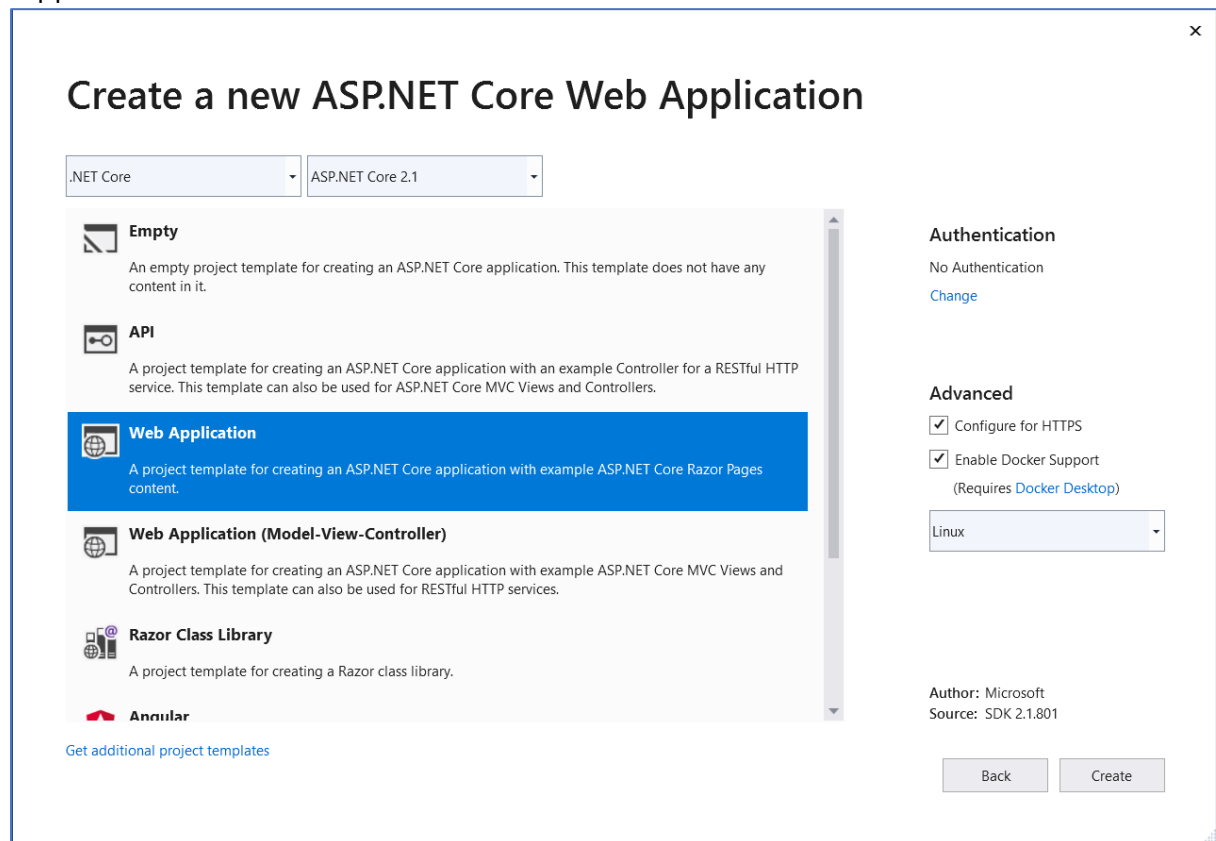


6. Start Visual Studio and connect to the Team Project from Team Explorer. Clone the repository.



7. Click on New for Solutions from Team Explorer (Please do not use File – New Project to create a new solution- if done the mapping may not be what is required after cloning). Select ASP.NET Core Web Application, select Web Application and ensure that Docker

support is enabled



8. Change existing dockerfile with following (you can download this from <https://github.com/GouriSohoni/Training> with file name as dockerrelated.zip)  
FROM microsoft/dotnet:2.2-aspnetcore-runtime AS base

```
WORKDIR /app
EXPOSE 80
```

```
FROM microsoft/dotnet:2.2-sdk AS build
WORKDIR /src
COPY AKSDemo.csproj .
RUN dotnet restore AKSDemo.csproj
COPY . .
RUN dotnet build AKSDemo.csproj -c Release -o /app
```

```
FROM build AS publish
RUN dotnet publish AKSDemo.csproj -c Release -o /app
```

```
FROM base AS final
WORKDIR /app
COPY --from=publish /app .
ENTRYPOINT ["dotnet", "AKSDemo.dll"]
```

9. Add yaml filename deployment.yml. Create a folder named manifest and add the file to it (this is also available in zip file. Remember to do the necessary changes to file – in this the image needs to be copied from container registry created earlier)

```
apiVersion: apps/v1beta1
kind: Deployment
metadata:
 name: aksdemo
```

```

spec:
 replicas: 1
 template:
 metadata:
 labels:
 app: aksdemo-app
spec:
 containers:
 - name: aksdemo-services-app
 image: myaksdemogs.azurecr.io/aksdemo:latest
 ports:
 - containerPort: 80

 apiVersion: v1
 kind: Service
 metadata:
 name: aksdemo-app
spec:
 ports:
 - name: http-port
 port: 80
 targetPort: 80
 selector:
 app: aksdemo-app
 type: LoadBalancer

```

10. Do some small change to any of the pages. Commit and push all code. If you are using Visual Studio 2019 you need to take care of adding ignore file
11. Let us create a build definition. Select Pipelines – create pipelines- use the classic editor and select the template of Docker Container. Provide azure subscription and authorize. Make sure that your browser supports popups. Now the name of container registry can be selected. Provide the Image name as <app name>:\$(Build.BuildId) and select the dockerfile from repo. Click on include latest tag

Image Name \* ⓘ

AKSApp:\$(Build.BuildId)

☒ Qualify Image Name ⓘ

Additional Image Tags ⓘ

☐ Include Source Tags ⓘ

☒ Include Latest Tag ⓘ

Complete configuration looks as follows


The screenshot displays the Azure DevOps pipeline configuration interface. The left sidebar shows the pipeline structure with tasks: 'Get sources', 'Agent job 1', 'Build an image', 'Push an image' (highlighted), 'Copy Files to: \$(Build.ArtifactStagingDirectory)', and 'Publish Pipeline Artifact'. The main area shows the configuration for the 'Push an image' task, which is a Docker action. The configuration includes: 'Azure subscription' set to 'Visual Studio Enterprise', 'Azure Container Registry' set to 'myaksdemogs', 'Image Name' set to 'AKSDemo:\$(Build.BuildId)', and 'Qualify Image Name' checked. There are also checkboxes for 'Include Source Tags' (unchecked) and 'Include Latest Tag' (checked).


12. Provide similar configuration for push image
13. Make sure that the build name is small and without any spaces
14. Add 2 tasks for copy and publish. For copy file select source as manifest folder and destination as `$(Build.ArtifactStagingDirectory)`. Publish artifact picks up from `$(Build.ArtifactStagingDirectory)`, so you do not have to do anything (we need this file to deploy which will be taken care of in next stage)  
Note: Make sure you are not selecting Publish Pipeline artifact
15. Save and queue the build, you will observe that the agent has already picked up as ubuntu
16. After successful build you can find that the repository is shown in the Container Registry
17. Now we need to deploy the application we created using Visual Studio, create a release definition for it. Select Release blade – New Pipeline – Select template for Deploy to Kubernetes Cluster.


18. Add the artifact to build we created earlier, observe the source alias created


### Add an artifact

Source type

  
✓ Build

  
Azure Repos ...

  
GitHub

  
TFVC

[5 more artifact types](#) ▾

Project \* ⓘ

aksfordemo ▾

Source (build pipeline) \* ⓘ

aksbulddemo ▾

Default version \* ⓘ

Latest ▾

Source alias \* ⓘ

aksbulddemo

ⓘ The artifacts published by each version will be available for deployment in release pipelines. The latest successful build of **aksbulddemo** published the following artifacts: **Job1**.

Add

19. Configure kubectl task which got added, create connection to Kubernetes service by using New- select Azure Subscription and provide the required details for authentication. Select the cluster name and namespace as default

**New service connection**

Azure Subscription  
Visual Studio Enterprise (

Cluster  
AKSDemo (RG-DemoAKS)

Namespace  
default

☐ Use cluster admin credentials

**Details**

Service connection name  
MyAKsConn

Description (optional)

Security  
☒ Grant access permission to all pipelines

[Learn more](#) **Save**

You can decide if you want to grant access to all pipelines or not.

20. Do not select any namespace for kubectl task, select command as apply

21. Add an argument by clicking on ellipse, enter f for file and provide value as <source alias for build>/drop/deployment.yml

Arguments

Name

Value

f

\_aksbuilddemo/drop/deployment.yml

+ Add

It will look as follows

Arguments ⓘ

```
-f _aksbuilddemo/drop/deployment.yml
```

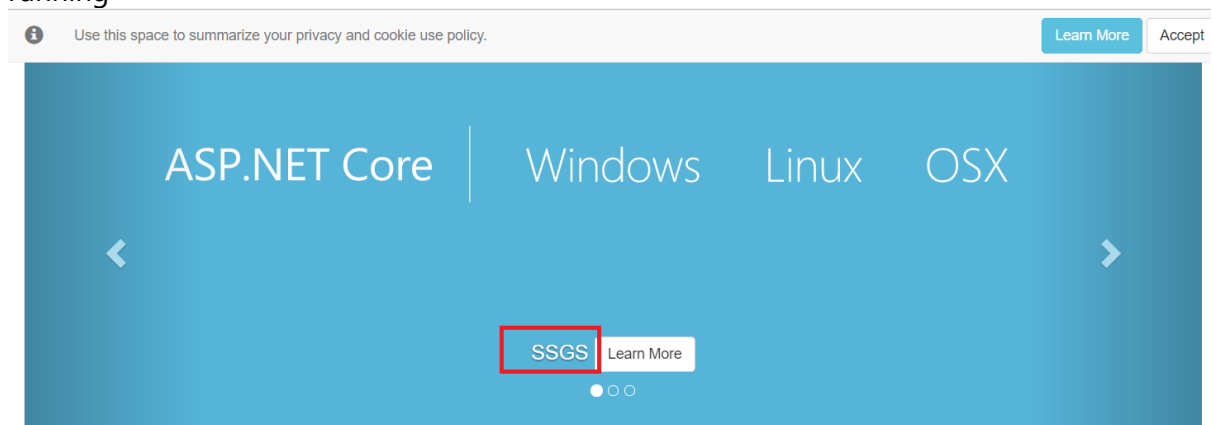
22. Save the release definition with proper name (ensure that the agent selected is ubuntu) and create release
23. After release succeeds, start cloud shell and provide credentials by using `az aks get-credentials --resource-group <RG name> --name <cluster name>` You should get

```
Merged "AKSDemo" as current context in /home/gouri/.kube/config
```

24. Enter commands `kubectl get pods` and `kubectl get services`

```
PS Azure:\> kubectl get pods
NAME READY STATUS RESTARTS AGE
aksdemo-b758d4d97-kk4wn 1/1 Running 0 4m3s
Azure:/
PS Azure:\> kubectl get services
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
aksdemo-app LoadBalancer 10.0.176.182 52.191.222.5 80:32401/TCP 4m14s
kubernetes ClusterIP 10.0.0.1 <none> 443/TCP 56m
```

25. Select the external IP address and enter in browser, you should see the application running



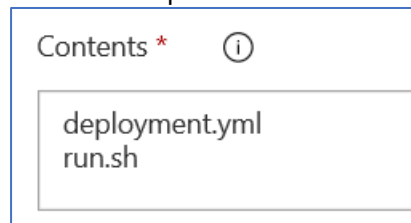
26. I had made one change in index.cshtml which is being shown in the above diagram

Remaining steps can be done later if time permits – to create complete CI CD for AKS

27. Add a file named run.sh to manifest folder using Visual Studio and enter

```
sed -i "s/latest/$1/1" _aksbulddemo/drop/deployment.yml
```

28. We need to publish this file to drop folder so change the copy file as



Change the build trigger to CI and just save the build definition

29. Commit and push the changes so that build will be automatically rejiggered.

30. Let us take care of having latest tag in release definition, add bash script task to release definition before kubectl apply. (this can be configured only after build is successful so as to have run.sh in drop folder).