**SCENARIO – 2 ANSWERS**

1. *What are different artifacts you need to create - name of the artifacts and its purpose*

Ans) For creation of the pipelines, the below artifacts are used and their purposes are –

* Azuredeploy.json – Contains the code for the resources to be created and the parameters needed to create these resources. It contains the variables to create these resources. For example, the below code –

{

"name": "[variables('VmStorageName')]",

"type": "Microsoft.Storage/storageAccounts",

"location": "[resourceGroup().location]",

"apiVersion": "2016-01-01",

"sku": {

"name": "[parameters('VmStorageType')]"

},

It creates the resource – storage account. The various variables needed for this are –

Name – Defines the name of the storage account which would be created by the ARM template.

Type – Defines the type of resource which has to be created. In this case it is a storage account which is being created.

Location – Location decides where the particular resource would be created. There are various regions in Azure and the location parameter decides where we want to create the particular resource.

* Azuredeploy.parameters. json – It contains the parameters for the resources which are created by azuredeploy.json file. This parameter file provides the parameters for the variables which are used during the deployment of the template. For example, the below code –

"WindowsVMWindowsOSVersion": {

"type": "string",

"defaultValue": "2016-R2-Datacenter",

"allowedValues": [

"2008-R2-SP1",

"2012-Datacenter",

"2012-R2-Datacenter",

"Windows-Server-Technical-Preview",

"2016-Nano-Server",

"2016-Datacenter-with-Containers",

"2016-Datacenter",

"2019-Datacenter",

"2019-Datacenter-Core",

"2019-Datacenter-Core-smalldisk",

"2019-Datacenter-Core-with-Containers",

"2019-Datacenter-Core-with-Containers-smalldisk",

"2019-Datacenter-smalldisk",

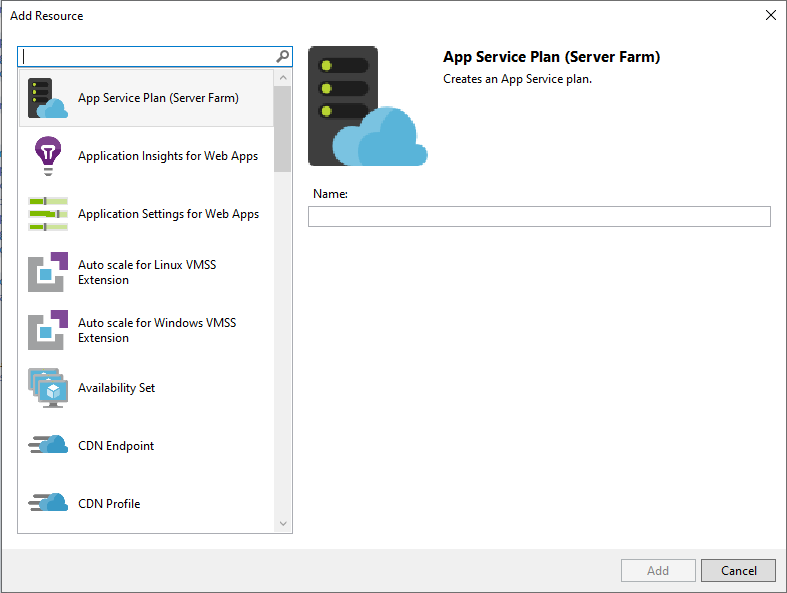
"2019-Datacenter-with-Containers",

"2019-Datacenter-with-Containers-smalldisk"

]

}

The above code defines what would be the version of the VM that would be created. There is a default value which is accepted by default however, there are other allowed values which can be used while defining the parameter file. Anything beyond this is not allowed. The resources tab defines the various types of resources which can be used in the parameter file. Multiple resources can be added from Visual Studio like VNet, Storage Account, Virtual Machine etc. in order to add a resource from Visual Studio we need to right click on the resources and add a resource type –



In our case it is a Windows Virtual Machine. We need to punch in the required details and the code would be populated in the visual studio which can be later edited as per the requirement.

1. *List the tools you will to create and store the ARM templates.*

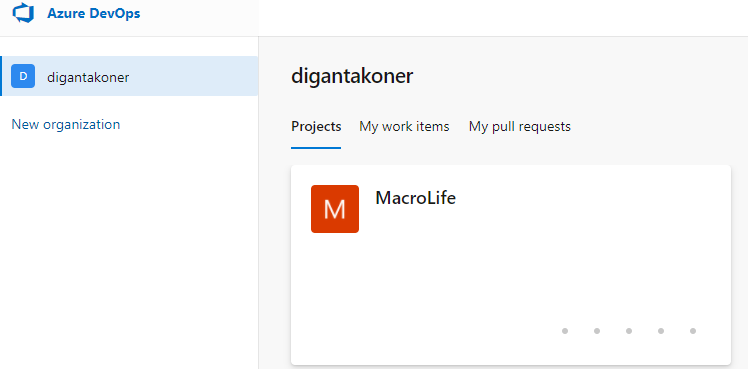
Ans) The tools used in this scenario are –

* Visual Studio – Used to create the ARM templates. Creating resources are easier in the pre-built templates in visual Studio.
* Git – ARM templates will be stored in Git so that it is easy to work on the code by creating a local copy of the code and make the changes before pushing it to the branch. Also, a trigger in the master branch would trigger a deployment of the pipeline thereby automating the process of creating new resources by just checking in the code to the branch.

1. *Explain the process and steps to create automated deployment pipeline.*

Ans) Steps to create the automated pipeline –

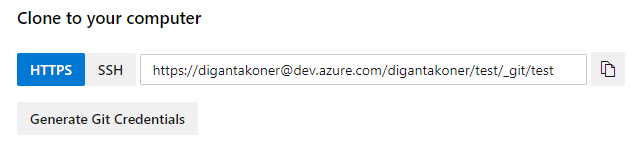
* ARM template to deploy the resource is written in Visual Studio post which it is checked into Git.
* This is followed by creating a project in the Azure Devops.



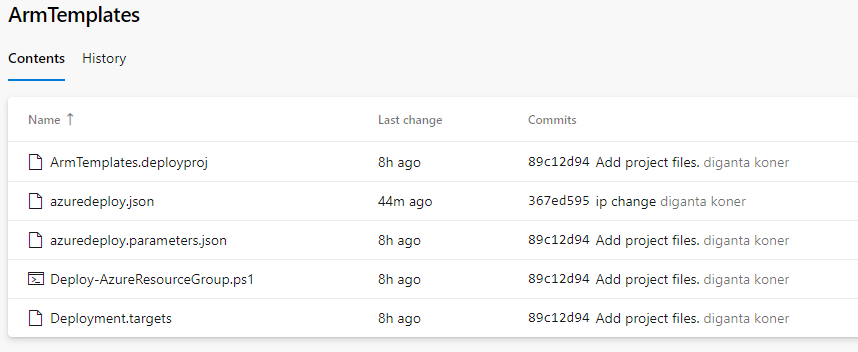
* The Version Control is set to Git so that any code that is written in Visual Studio is checked into this source control.



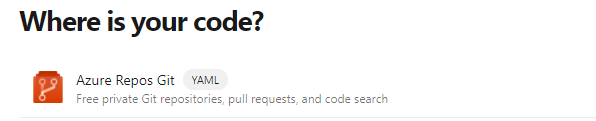
* The repo in Visual Studio is cloned using the URL of the Git.



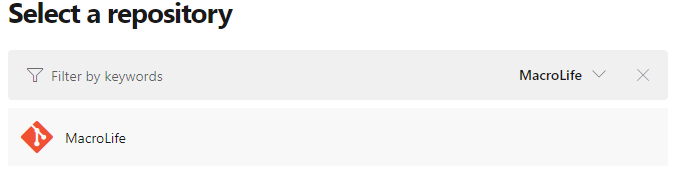
* Once the repo is cloned, the files would appear in the Azure Git Repo



* Once the files have been uploaded, the next stage is to create the pipeline which would actually deploy the code.



Since our code is in Azure Git, we would select this option and select the repository which was created.



Since this project has been built on ASP.NET Core, that option would be selected, once done the pipeline changes can be saved and run. This would create the AzurePipeline.yaml file.

The trigger is the master branch which means any changes made in the master branch would trigger a build and deploy.

Two below tasks have been added in the yaml file.

- task: CopyFiles@2

  inputs:

    SourceFolder: 'ArmTemplates'

    Contents: 'azuredeploy\*.json'

    TargetFolder: '$(build.artifactStagingDirectory)'

- task: PublishBuildArtifacts@1

  inputs:

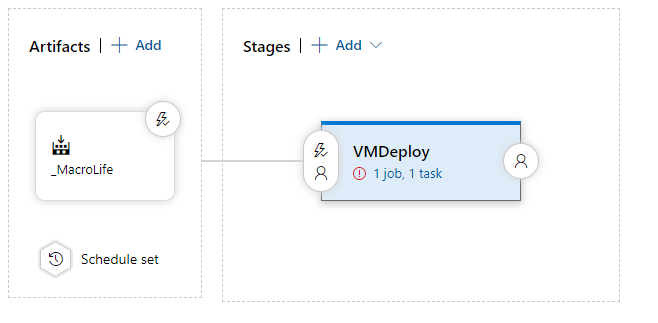
    PathtoPublish: '$(Build.ArtifactStagingDirectory)'

    ArtifactName: 'drop'

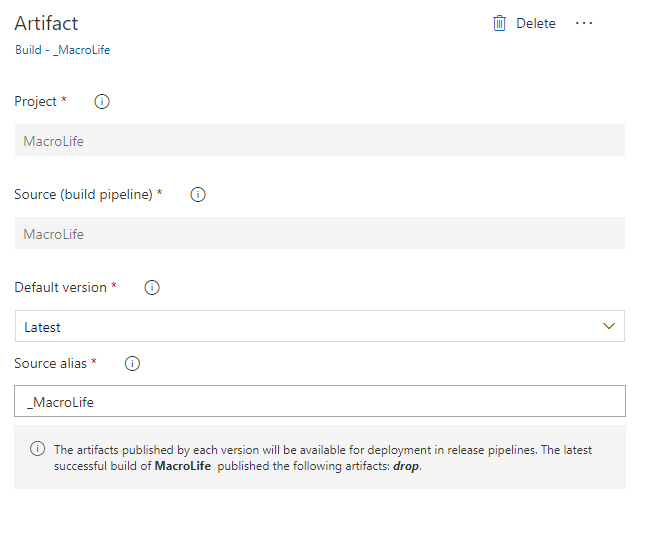
    publishLocation: 'Container'

The first task copies the files to the staging directory and the second task build the artifacts and deploys them.

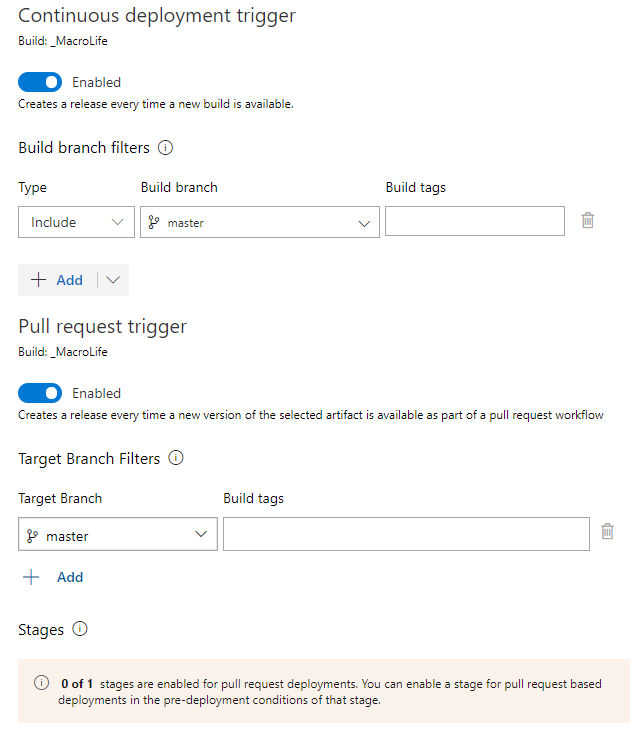
* Once the pipelines have been created, the releases need to be created which would deploy the code to the azure subscription.
* In the pipelines section, we would have to add the artifacts and the stages where the deployment would happen.



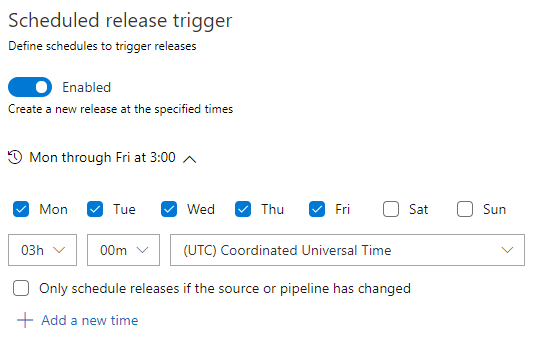
The artifact would be populated from the source code.



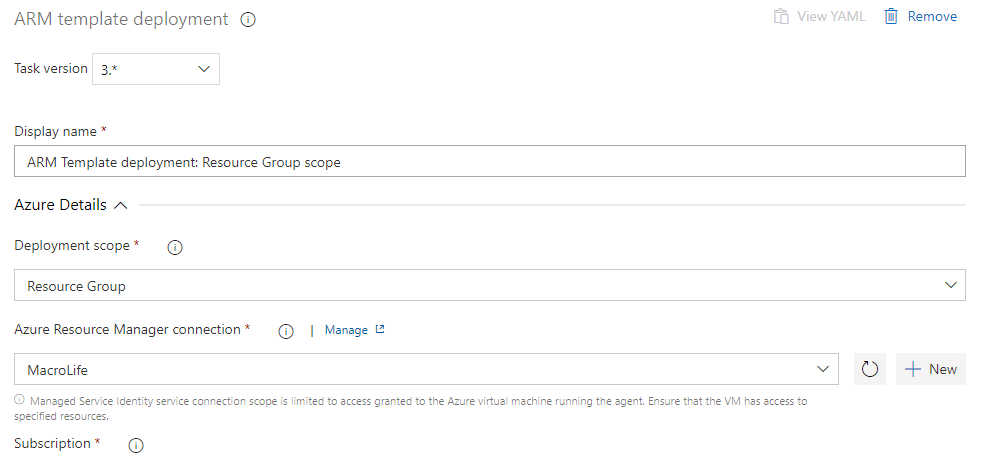
The continuous deployment trigger would make sure that any new builds would be triggering a new deployment.



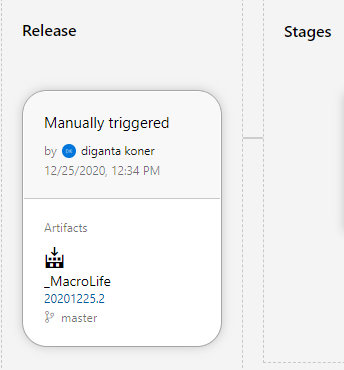
The schedule is set to a timeline so that period deployments keep happening.



* The next step is to create the tasks which would involve deploying the template.



Once the task is created, it is ready to be deployed.



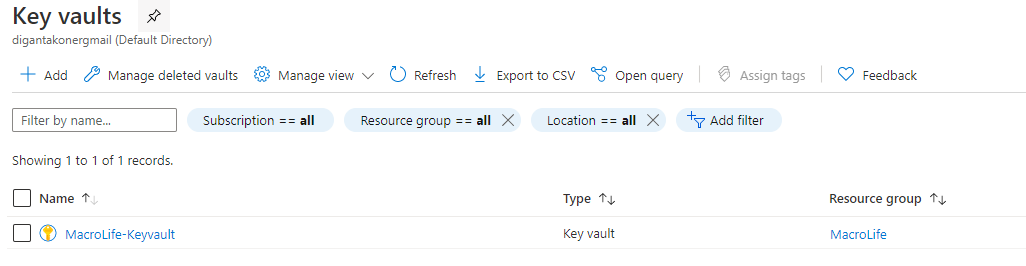
1. *Create a sample ARM template you will use to deploy a Windows VM of any size*

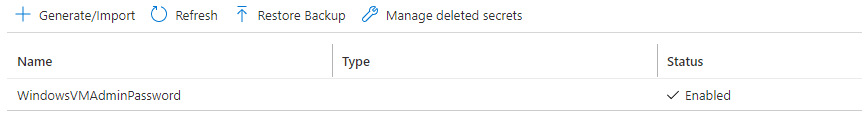
Ans) File has been uploaded to GitHub. Folder Name - ARM

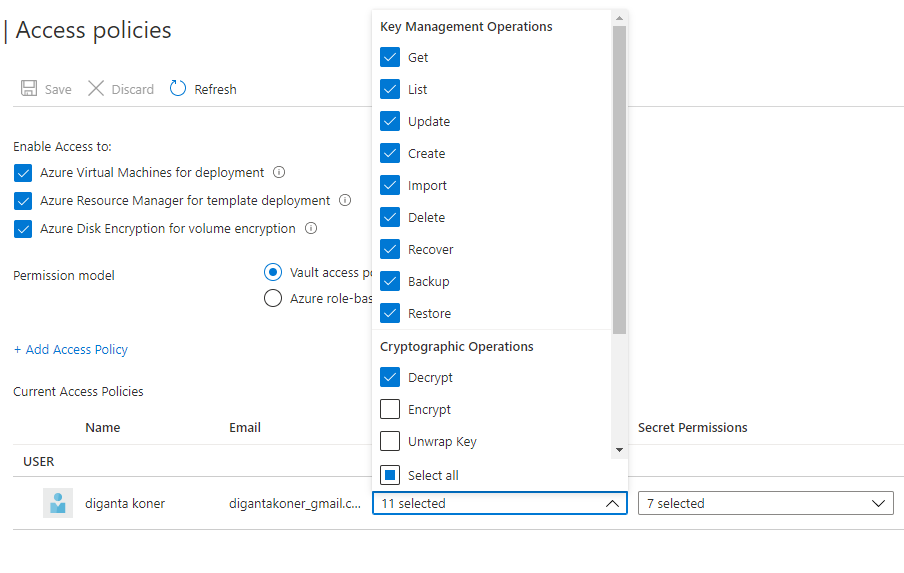
*5) Explain how will you access the password stored in Key Vault and use it as Admin Password in the VM ARM template.*

Ans) Steps to create the key and access it –

* Navigating to Azure Key vaults, a new key vault is created.



* A new secret is created, the same name is used in the ARM template to extract the key.
* Access control is set to read and decrypt the key



ARM template code used to extract the password from the keyvault.

"WindowsVMAdminPassword": {

"type": "securestring",

"reference": {

"keyVault": {

"id": "/subscriptions/9b7c8a84-7b15-4520-ab26-2ed509a4b469/resourceGroups/mykeyvaultdeploymentrg/providers/Microsoft.KeyVault/vaults/MacroLife-Keyvault"

},

"secretName": "WindowsVMAdminPassword"

}

},

The key vault and the secret name along with the subscription id is used to locate the key and deploy the VM.