**Q2 - SCENARIO**

Macro Life, a healthcare company has recently setup the entire Network and Infrastructure on Azure.

The infrastructure has different components such as Virtual N/W, Subnets, NIC, IPs, NSG etc.

The IT team currently has developed PowerShell scripts to deploy each component where all the properties of each resource is set using PowerShell commands.

The business has realized that the PowerShell scripts are growing over period of time and difficult to handover when new admin onboards in the IT.

The IT team has now decided to move to ARM based deployment of all resources to Azure.

All the passwords are stored in a Azure Service known as key Vault. The deployments needs to be automated using Azure DevOps using IaC(Infrastructure as Code).

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

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

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

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

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

**Solution:**

To deploy the ARM template using Azure DevOps first we have to make sure the template files should be in the **Azure Repos GIT** and then we need to create the pipeline using **YAML or Classic edito**r.

We have to add or create the tasks like copying the files to the artifact staging directory and **publish the build artifact in CI pipeline** where it can drop the template files and we can make use of that artifact at the time of **release pipeline** where the actual deployment takes place.

In CD( release Pipeline ) have to add the artifact which we published at the time of build pipeline in release pipeline and in the stage add **task ARM deployment** and have to fill the required options like **Deployment scope, Azure resource manager connection type, Azure subscription, Action type create and update resource group, Resource group - Name, Location , Template and template parameters location path and Deployment mode to incremental**( because we are going to create new resources so have to use incremental).

We have to select appropriate options on all the fields.

As we know to automate all these deployments we have to **enable CI( continuous trigger ) pipeline** so that whenever any code committed to the files to that particular repo of that branch the pipeline will get triggered automatically and after the completion of CI Build pipeline the Deployment in release pipeline will triggered and deploy the resources automatically to the environment.( for that **Continuous deployment trigger should be enabled** along with **pre deployment approvals should be disabled** of that particular stage or environment ) So that everything goes in a automated way.

**Sample ARM template and added keyvault as a parameter.**

( where we have to create a key vault and save your admin password in your key vault as a secret and then we can pass the same key vault name in the template )

{

"$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",

"contentVersion": "1.0.0.0",

"parameters": {

"adminUsername": {

"type": "string",

"metadata": {

"description": "Username for the Virtual Machine."

}

},

"adminPassword": {

"reference": {

"keyVault": {

"id": "/subscriptions/<SubscriptionID>/resourceGroups/mykeyvaultdeploymentrg/providers/Microsoft.KeyVault/vaults/<KeyVaultName>"

},

"secretName": "vmAdminPassword"

}

},

"dnsLabelPrefix": {

"type": "string",

"defaultValue": "[toLower(concat(parameters('vmName'),'-', uniqueString(resourceGroup().id, parameters('vmName'))))]",

"metadata": {

"description": "Unique DNS Name for the Public IP used to access the Virtual Machine."

}

},

"publicIpName": {

"type": "string",

"defaultValue": "myPublicIP",

"metadata": {

"description": "Name for the Public IP used to access the Virtual Machine."

}

},

"publicIPAllocationMethod": {

"type": "string",

"defaultValue": "Dynamic",

"allowedValues": [

"Dynamic",

"Static"

],

"metadata": {

"description": "Allocation method for the Public IP used to access the Virtual Machine."

}

},

"publicIpSku": {

"type": "string",

"defaultValue": "Basic",

"allowedValues": [

"Basic",

"Standard"

],

"metadata": {

"description": "SKU for the Public IP used to access the Virtual Machine."

}

},

"OSVersion": {

"type": "string",

"defaultValue": "2019-Datacenter",

"allowedValues": [

"2016-Datacenter",

"2019-Datacenter",

],

"metadata": {

"description": "The Windows version for the VM. This will pick a fully patched image of this given Windows version."

}

},

"vmSize": {

"type": "string",

"defaultValue": "Standard\_D2s\_v3",

"metadata": {

"description": "Size of the virtual machine."

}

},

"location": {

"type": "string",

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

"metadata": {

"description": "Location for all resources."

}

},

"vmName": {

"type": "string",

"defaultValue": "simple-vm",

"metadata": {

"description": "Location for all resources."

}

}

},

"variables": {

"storageAccountName": "[concat('bootdiags', uniquestring(resourceGroup().id))]",

"nicName": "myVMNic",

"addressPrefix": "10.0.0.0/16",

"subnetName": "Subnet",

"subnetPrefix": "10.0.0.0/24",

"virtualNetworkName": "MyVNET",

"subnetRef": "[resourceId('Microsoft.Network/virtualNetworks/subnets', variables('virtualNetworkName'), variables('subnetName'))]",

"networkSecurityGroupName": "default-NSG"

},

"resources": [

{

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

"apiVersion": "2020-09-01",

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

"location": "[parameters('location')]",

"sku": {

"name": "Standard\_LRS"

},

"kind": "Storage",

"properties": {}

},

{

"type": "Microsoft.Network/publicIPAddresses",

"apiVersion": "2020-09-01",

"name": "[parameters('publicIPName')]",

"location": "[parameters('location')]",

"sku": {

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

},

"properties": {

"publicIPAllocationMethod": "[parameters('publicIPAllocationMethod')]",

"dnsSettings": {

"domainNameLabel": "[parameters('dnsLabelPrefix')]"

}

}

},

{

"type": "Microsoft.Network/networkSecurityGroups",

"apiVersion": "2020-09-01",

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

"location": "[parameters('location')]",

"properties": {

"securityRules": [

{

"name": "default-allow-3389",

"properties": {

"priority": 1000,

"access": "Allow",

"direction": "Inbound",

"destinationPortRange": "3389",

"protocol": "Tcp",

"sourcePortRange": "\*",

"sourceAddressPrefix": "\*",

"destinationAddressPrefix": "\*"

}

}

]

}

},

{

"type": "Microsoft.Network/virtualNetworks",

"apiVersion": "2020-09-01",

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

"location": "[parameters('location')]",

"dependsOn": [

"[resourceId('Microsoft.Network/networkSecurityGroups', variables('networkSecurityGroupName'))]"

],

"properties": {

"addressSpace": {

"addressPrefixes": [

"[variables('addressPrefix')]"

]

},

"subnets": [

{

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

"properties": {

"addressPrefix": "[variables('subnetPrefix')]",

"networkSecurityGroup": {

"id": "[resourceId('Microsoft.Network/networkSecurityGroups', variables('networkSecurityGroupName'))]"

}

}

}

]

}

},

{

"type": "Microsoft.Network/networkInterfaces",

"apiVersion": "2020-09-01",

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

"location": "[parameters('location')]",

"dependsOn": [

"[resourceId('Microsoft.Network/publicIPAddresses', parameters('publicIPName'))]",

"[resourceId('Microsoft.Network/virtualNetworks', variables('virtualNetworkName'))]"

],

"properties": {

"ipConfigurations": [

{

"name": "ipconfig1",

"properties": {

"privateIPAllocationMethod": "Dynamic",

"publicIPAddress": {

"id": "[resourceId('Microsoft.Network/publicIPAddresses', parameters('publicIPName'))]"

},

"subnet": {

"id": "[variables('subnetRef')]"

}

}

}

]

}

},

{

"type": "Microsoft.Compute/virtualMachines",

"apiVersion": "2020-09-01",

"name": "[parameters('vmName')]",

"location": "[parameters('location')]",

"dependsOn": [

"[resourceId('Microsoft.Storage/storageAccounts', variables('storageAccountName'))]",

"[resourceId('Microsoft.Network/networkInterfaces', variables('nicName'))]"

],

"properties": {

"hardwareProfile": {

"vmSize": "[parameters('vmSize')]"

},

"osProfile": {

"computerName": "[parameters('vmName')]",

"adminUsername": "[parameters('adminUsername')]",

"adminPassword": "[parameters('adminPassword')]"

},

"storageProfile": {

"imageReference": {

"publisher": "MicrosoftWindowsServer",

"offer": "WindowsServer",

"sku": "[parameters('OSVersion')]",

"version": "latest"

},

"osDisk": {

"createOption": "FromImage",

"managedDisk": {

"storageAccountType": "StandardSSD\_LRS"

}

},

"dataDisks": [

{

"diskSizeGB": 1023,

"lun": 0,

"createOption": "Empty"

}

]

},

"networkProfile": {

"networkInterfaces": [

{

"id": "[resourceId('Microsoft.Network/networkInterfaces', variables('nicName'))]"

}

]

},

"diagnosticsProfile": {

"bootDiagnostics": {

"enabled": true,

"storageUri": "[reference(resourceId('Microsoft.Storage/storageAccounts', variables('storageAccountName'))).primaryEndpoints.blob]"

}

}

}

}

],

"outputs": {

"hostname": {

"type": "string",

"value": "[reference(parameters('publicIPName')).dnsSettings.fqdn]"

}

}

}

Command to execute the template via powershell:  
  
New-AzResourceGroupDeployment -ResourceGroupName $resourceGroupName -TemplateFile "< path for actual template .json >" -TemplateParameterFile "< path of Parameter file .json >"