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

+++++++++

Answer

+++++++

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

Azure Pipelines.

You can link a release pipeline to any of the build pipelines in Azure Pipelines or TFS project collection.

TFVC, Git, and GitHub.

There are scenarios in which you may want to consume artifacts stored in a version control system directly, without passing them through a build pipeline. For example:

You are developing a PHP or a JavaScript application that does not require an explicit build pipeline.

Jenkins.

To consume Jenkins artifacts, you must create a service connection with credentials to connect to your Jenkins server. For more information, see service connections and Jenkins service connection. You can then link a Jenkins project to a release pipeline

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

Create ssh-keys and store in KeyVault

Deploy Data Lake Store account with encryption(Key Vault)

Create a VM from Image Version

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

Sign into the Microsoft Azure portal.

Choose the + Create a resource icon in the left navigation bar, then search for DevOps project. Then choose DevOps Project in the list. Select Create.

Select the .NET sample application and click Next.

The .NET samples include a choice of either the open source ASP.NET framework or the cross-platform .NET Core framework. Select the .NET Core application framework. This sample is an ASP.NET Core MVC application. And also enable Add a database toggle to add the database to the application. When you’re done, choose Next.

Web App on Windows is the default deployment target. You can optionally choose Virtual Machine also. When you’re done, choose Next.

Select your Azure DevOps organization and choose a name for your project and Web app. When you’re done, choose Done.

4) ARM Sample 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": {

"type": "securestring",

"minLength": 12,

"metadata": {

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

}

},

"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": [

"2008-R2-SP1",

"2012-Datacenter",

"2012-R2-Datacenter",

"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"

],

"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\_D2\_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": "Name of the virtual machine."

}

}

},

"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": "2019-06-01",

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

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

"sku": {

"name": "Standard\_LRS"

},

"kind": "Storage",

"properties": {}

},

{

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

"apiVersion": "2020-06-01",

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

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

"sku": {

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

},

"properties": {

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

"dnsSettings": {

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

}

}

},

{

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

"apiVersion": "2020-06-01",

"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": "2020-06-01",

"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": "2020-06-01",

"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": "2020-06-01",

"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]"

}

}

}

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

This tutorial covers the following tasks:

Prepare a key vault

Open a quickstart template

Edit the parameters file

Deploy the template

Validate the deployment

Clean up resources