**ARM Templates**

**ARM template strcture:**

**-----------------**

Here is an example of the basic structure of an ARM template:

{

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

  "contentVersion": "1.0.0.0",

  "parameters": {

    // Define template parameters here

  },

  "variables": {

    // Define template variables here

  },

  "resources": [

    // Define resources here

  ],

  "outputs": {

    // Define template outputs here

  }

}

The $schema property specifies the location of the JSON schema file that defines the syntax and structure of the template.

The contentVersion property specifies the version of the template.

The parameters section allows you to define input parameters that can be passed to the template during deployment.

The variables section allows you to define reusable values that can be used throughout the template.

The resources section is where you define the resources to create as part of the deployment. Each resource is defined as a JSON object with properties that specify the type and other details about the resource.

The outputs section allows you to define values that can be returned by the template after deployment.

Here is an example of a simple ARM template that creates an Azure storage account:

{

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

  "contentVersion": "1.0.0.0",

  "parameters": {

    "storageAccountName": {

      "type": "string",

      "minLength": 3,

      "maxLength": 24,

      "metadata": {

        "description": "The name of the storage account."

      }

    }

  },

  "variables": {

    "storageAccountType": "Standard\_LRS"

  },

  "resources": [

    {

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

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

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

      "sku": {

        "name": "[variables('storageAccountType')]"

      },

      "kind": "Storage",

      "properties": {}

    }

  ]

}

This template defines a single parameter, storageAccountName, and a single variable, storageAccountType. It also defines a single resource, an Azure storage account, with a name that is based on the storageAccountName parameter and a type that is based on the storageAccountType variable.

**Variables:**

Here is an example of using variables in an ARM template:

{

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

  "contentVersion": "1.0.0.0",

  "parameters": {

    "vmSize": {

      "type": "string",

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

      "allowedValues": [

        "Standard\_D1\_v2",

        "Standard\_D2s\_v3"

      ],

      "metadata": {

        "description": "The size of the virtual machine."

      }

    }

  },

  "variables": {

    "vmImagePublisher": "MicrosoftWindowsServer",

    "vmImageOffer": "WindowsServer",

    "vmImageSKU": "2019-Datacenter"

  },

  "resources": [

    {

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

      "name": "myVM",

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

      "properties": {

        "hardwareProfile": {

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

        },

        "storageProfile": {

          "imageReference": {

            "publisher": "[variables('vmImagePublisher')]",

            "offer": "[variables('vmImageOffer')]",

            "sku": "[variables('vmImageSKU')]",

            "version": "latest"

          }

        }

      }

    }

  ]

}

In this example, the variables section defines three variables: vmImagePublisher, vmImageOffer, and vmImageSKU. These variables are used to specify the publisher, offer, and SKU of the image to use for the virtual machine. This allows you to reuse these values throughout the template, rather than hardcoding them in multiple places.

To use a variable in your template, you can reference it using the variables() function, followed by the name of the variable in square brackets. For example, to use the vmImagePublisher variable, you would use [variables('vmImagePublisher')].

You can also define variables that depend on other variables or parameters. For example:

"variables": {

  "vmSize": "[parameters('vmSize')]",

  "vmMemory": "[if(equals(variables('vmSize'), 'Standard\_D2s\_v3'), 4, 2)]"

}

In this example, the vmMemory variable is set to 4 if the vmSize variable is equal to Standard\_D2s\_v3, and 2 otherwise. This allows you to use the vmMemory variable in other parts of the template, rather than repeating the conditional logic.

parameters:

------------

Here is an example of using parameters in an ARM template:

{

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

  "contentVersion": "1.0.0.0",

  "parameters": {

    "vmName": {

      "type": "string",

      "minLength": 1,

      "maxLength": 15,

      "metadata": {

        "description": "The name of the virtual machine."

      }

    },

    "vmSize": {

      "type": "string",

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

      "allowedValues": [

        "Standard\_D1\_v2",

        "Standard\_D2s\_v3"

      ],

      "metadata": {

        "description": "The size of the virtual machine."

      }

    }

  },

  "variables": {

    // Define template variables here

  },

  "resources": [

    {

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

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

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

      "properties": {

        "hardwareProfile": {

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

        }

      }

    }

  ]

}

In this example, the parameters section defines two parameters: vmName and vmSize. The vmName parameter is a string with a minimum length of 1 and a maximum length of 15. The vmSize parameter is a string with a default value of Standard\_D2s\_v3 and a list of allowed values.

To use a parameter in your template, you can reference it using the parameters() function, followed by the name of the parameter in square brackets. For example, to use the vmName parameter, you would use [parameters('vmName')].

You can pass the values for the parameters when you deploy the template using the Azure portal, Azure PowerShell, or the Azure CLI. For example, to deploy the template using the Azure CLI, you can use the following command:

az deployment group create --name myDeployment --resource-group myResourceGroup --template-file template.json --parameters vmName=myVM vmSize=Standard\_D2s\_v3

This command deploys the template in the myResourceGroup resource group and passes the vmName parameter a value of myVM and the vmSize parameter a value of Standard\_D2s\_v3.

You can also specify default values for your parameters in the template, as shown in the example above. This allows you to deploy the template without specifying a value for every parameter.

Regenerate response

**Functions:**

**----------**

Here are some examples of functions that can be used in Azure Resource Manager (ARM) templates:

concat: This function concatenates two or more strings. For example:

"outputs": {

  "concatOutput": {

    "type": "string",

    "value": "[concat('oursbank', 'storacgeaccount')]"

  }

}

resourceId: This function returns the resource ID of a resource. For example:

"variables": {

  "resourceId": "[resourceId('Microsoft.Storage/storageAccounts', 'myStorageAccount')]"

}

variables: This function returns the value of a variable. For example:

"variables": {

  "var1": "value1",

  "var2": "value2"

},

"outputs": {

  "output1": {

    "type": "string",

    "value": "[variables('var1')]"

  },

  "output2": {

    "type": "string",

    "value": "[variables('var2')]"

  }

}

uniqueString: This function generates a unique string based on a specified prefix. For example:

"variables": {

  "uniqueString": "[uniqueString('myPrefix')]"

}

reference: This function returns the value of a property for a resource. For example:

"outputs": {

  "vmName": {

    "type": "string",

    "value": "[reference(variables('vmId')).name]"

  }

}

These are just a few examples of the functions that are available in ARM templates. There are many more functions that you can use, depending on your specific needs.

If conditions in ARM templates:

----------------------------------------------------

Checking the value of a parameter:

{

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

  "name": "myVM",

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

  "properties": {

    "hardwareProfile": {

      "vmSize": "[if(equals(parameters('vmSize'), 'Standard\_D2s\_v3'), 'Standard\_D2s\_v3', 'Standard\_D1\_v2')]"

    }

  }

}

In this example, the vmSize property of the virtual machine is set to Standard\_D2s\_v3 if the vmSize parameter is equal to Standard\_D2s\_v3, and Standard\_D1\_v2 otherwise.

Checking the value of a resource:

{

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

  "name": "myVM",

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

  "properties": {

    "hardwareProfile": {

      "vmSize": "[if(equals(reference(variables('vmId')).sku.name, 'Standard\_D2s\_v3'), 'Standard\_D2s\_v3', 'Standard\_D1\_v2')]"

    }

  }

}

In this example, the vmSize property of the virtual machine is set to Standard\_D2s\_v3 if the sku.name property of the virtual machine referenced by the vmId variable is equal to Standard\_D2s\_v3, and Standard\_D1\_v2 otherwise.

Checking the value of an environment variable:

{

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

  "name": "myVM",

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

  "properties": {

    "hardwareProfile": {

      "vmSize": "[if(equals(environment('AzureWebJobsEnv'), 'Development'), 'Standard\_D2s\_v3', 'Standard\_D1\_v2')]"

    }

  }

}

In this example, the vmSize property of the virtual machine is set to Standard\_D2s\_v3 if the AzureWebJobsEnv environment variable is equal to Development, and Standard\_D1\_v2 otherwise.

These are just a few examples of using the if function in ARM templates. You can use the if function to include any type of conditional logic in your templates.

Loops:

----------

In Azure Resource Manager (ARM) templates, you can use the copy function to iterate over an array and create resources based on the elements in the array. This can be useful when you need to create multiple resources of the same type, such as virtual machines or storage accounts.

Here is an example of using a loop in an ARM template:

{

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

"contentVersion": "1.0.0.0",

"parameters": {

"storageCount": {

"type": "int",

"defaultValue": 3

}

},

"resources": [

{

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

"apiVersion": "2019-04-01",

"name": "[concat(copyIndex(),'storage', uniqueString(resourceGroup().id))]",

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

"sku": {

"name": "Standard\_LRS"

},

"kind": "Storage",

"properties": {},

"copy": {

"name": "storagecopy",

"count": "[parameters('storageCount')]"

}

}

]

}

VM’s Creation

----------------

{

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

  "contentVersion": "1.0.0.0",

  "parameters": {

    "vmNames": {

      "type": "array",

“defaultValue”: “vm1”

“allowedValues”: [

“vm1”,

“vm2”,

“vm3” ]

      "metadata": {

        "description": "The names of the virtual machines."

      }

    },

    "vmSize": {

      "type": "string",

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

      "allowedValues": [

        "Standard\_D1\_v2",

        "Standard\_D2s\_v3"

      ],

      "metadata": {

        "description": "The size of the virtual machines."

      }

    }

  },

  "variables": {

    "vmImagePublisher": "MicrosoftWindowsServer",

    "vmImageOffer": "WindowsServer",

    "vmImageSKU": "2019-Datacenter"

  },

  "resources": [

    {

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

      "name": "[parameters('vmNames')[copyIndex()]]",

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

      "copy": {

        "name": "vmLoop",

        "count": "[length(parameters('vmNames'))]"

      },

      "properties": {

        "hardwareProfile": {

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

        },

        "storageProfile": {

          "imageReference": {

            "publisher": "[variables('vmImagePublisher')]",

            "offer": "[variables('vmImageOffer')]",

            "sku": "[variables('vmImageSKU')]",

            "version": "latest"

          }

        }

      }

    }

  ]

}

In this example, the vmNames parameter is an array of strings that specifies the names of the virtual machines to create. The copy function is used to iterate over this array and create a virtual machine for each element. The `copyIndex

keyvault-to-store-secrets

---------------------------------

{

    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",

    "contentVersion": "1.0.0.0",

    "resources": [

        {

            "type": "Microsoft.KeyVault/vaults",

            "apiVersion": "2015-06-01",

            "name": "OursBankVault",

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

            "properties": {

                "sku": {

                    "family": "A",

                    "name": "Standard"

                },

                "tenantId": "[subscription().tenantId]",

                "accessPolicies": []

            }

        },

        {

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

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

            "name": "oursbankstore",

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

            "sku": {

                "name": "Standard\_LRS"

            },

            "kind": "Storage"

        },

        {

            "type": "Microsoft.KeyVault/vaults/secrets",

            "apiVersion": "2015-06-01",

            "name": "OursBankVault/SomeSecret",

            "dependsOn": [

                "[resourceId('Microsoft.KeyVault/vaults', 'OursBankVault')]"

            ],

            "properties": {

                "contentType": "text/plain",

                "value": "ThisIpsemIsSecret"

            }

        },

        {

            "type": "Microsoft.KeyVault/vaults/secrets",

            "apiVersion": "2015-06-01",

            "name": "OursBankVault/SomeCertificate",

            "dependsOn": [

                "[resourceId('Microsoft.KeyVault/vaults', 'OursBankVault')]"

            ],

            "properties": {

                "contentType": "application/x-pkcs12",

                "value": "MIIV0QIBAzCC...LoremIpsum...RIJcq3QACAggA"

            }

        },

        {

            "type": "Microsoft.KeyVault/vaults/secrets",

            "apiVersion": "2015-06-01",

            "name": "OursBankVault/ConnectionString",

            "dependsOn": [

                "[resourceId('Microsoft.KeyVault/vaults', 'OursBankVault')]",

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

            ],

            "properties": {

                "contentType": "text/plain",

                "value": "[concat('DefaultEndpointsProtocol=https;AccountName=oursbankstore;AccountKey=', listKeys(resourceId('Microsoft.Storage/storageAccounts', 'oursbankstore'), providers('Microsoft.Storage', 'storageAccounts').apiVersions[0]).keys[0].value, ';')]"

            }

        }

    ]

}