Lab 1 - ARM Templates - Common Deployment Methods & Deployment Order

Create ARM template

Create ARM template skeleton

1. Open `C:\Lab_Files\M04` in Visual Studio Code and create a subfolder named `S03-Lab1`
2. Create a new file in `C:\Lab_Files\M04\S03-Lab1` named `DeploymentMethods.template.json` and open the file.
3. Type `arm!` and press `Enter` to insert the ARM template skeleton code snippet

```json
{
    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploy
    "contentVersion": "1.0.0.0",
    "parameters": {},
    "variables": {},
    "resources": [],
    "outputs": {}
}
```

Add parameters and variables

1. Using the `arm-param` snippet, add the following parameters to the `"parameters": {},` section with descriptions of your choice
   a. `"environment"`
   b. `"projectName"`

```json
"parameters": {
    "environment": {
        "type": "string",
        "metadata": {
            "description": "Environment (Dev/QA/Prod)"
        }
    },
    "projectName": {
        "type": "string",
        "metadata": {
            "description": "Project Name"
```

```
        }
    }
},
```

2. Add the following variables and values to the `"variables": {},` section

   a. `"storageName": "[uniqueString(subscription().subscriptionId)]"`
   b. `"vnetName": "[concat('VNet-', parameters('projectName'), '-', parameters('environment'))]"`

```json
"variables": {
    "storageName": "[uniqueString(subscription().subscriptionId)]",
    "vnetName": "[concat('VNet-', parameters('projectName'), '-', parameters(
},
```

Add storage account resource

1. Using the `arm-storage` snippet, add a storage account resource to the `"resources": [],` section. (**NOTE:** Depending on the version of the snippet extension, the snippet may be referenced by another name such as `arm-stg`)
2. Change the values of `"name"` and `"displayName"` from `StorageAccount1` to `"[variables('storageName')]"`

```json
  "resources": [
   {
       "type": "Microsoft.Storage/storageAccounts",
       "apiVersion": "2018-07-01",
       "name": "[variables('storageName')]",
       "location": "[resourceGroup().location]",
       "tags": {
           "displayName": "[variables('storageName')]"
       },
       "sku": {
           "name": "Standard_LRS"
       },
       "kind": "StorageV2"
   }
  ],
```

1. Add a blob container child resource to the storage account resource after the `"kind": "StorageV2"`. There is not an ARM snippet for this child resource so use the following code block:

```json
```

```
    {
        "name": "default/templates",
        "type": "blobServices/containers",
        "apiVersion": "2018-07-01",
        "dependsOn": [
            "[variables('storageName')]"
        ],
        "properties": {
            "publicAccess": "Blob"
        }
    }
```

1. The full storage account resource should look as follows:

```json
    {
        "type": "Microsoft.Storage/storageAccounts",
        "apiVersion": "2018-07-01",
        "name": "[variables('storageName')]",
        "location": "[resourceGroup().location]",
        "tags": {
            "displayName": "[variables('storageName')]"
        },
        "sku": {
            "name": "Standard_LRS"
        },
        "kind": "StorageV2",
        "resources": [
            {
                "name": "default/templates",
                "type": "blobServices/containers",
                "apiVersion": "2018-07-01",
                "dependsOn": [
                    "[variables('storageName')]"
                ],
                "properties": {
                    "publicAccess": "Blob"
                }
            }
        ]
    }
```

Add virtual network resource

1. Using the `arm-vnet` snippet, add a virtual network resource to the `"resources": [],` section. (**NOTE:** Depending on the version of the snippet extension, the snippet may be referenced by another name such as `arm-vn`)
2. Change the values of `"name"` and `"displayName"` from `"VirtualNetwork1"` to `"[variables('vNetName')]"`

```json
    {
        "type": "Microsoft.Network/virtualNetworks",
        "apiVersion": "2018-08-01",
        "name": "[variables('vnetName')]",
        "location": "[resourceGroup().location]",
        "tags": {
            "displayName": "[variables('vnetName')]"
        },
        "properties": {
            "addressSpace": {
                "addressPrefixes": [
                    "10.0.0.0/16"
                ]
            },
            "subnets": [
                {
                    "name": "Subnet-1",
                    "properties": {
                        "addressPrefix": "10.0.0.0/24"
                    }
                },
                {
                    "name": "Subnet-2",
                    "properties": {
                        "addressPrefix": "10.0.1.0/24"
                    }
                }
            ]
        }
    }
```

Review completed ARM template

The completed ARM template should look as follows:

```json
{
"$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTempl
"contentVersion": "1.0.0.0",
"parameters": {
    "environment": {
        "type": "string",
        "metadata": {
            "description": "Environment (Dev/QA/Prod)"
        }
    },
    "projectName": {
        "type": "string",
        "metadata": {
            "description": "Project Name"
```

```
        }
    }
},
"variables": {
    "storageName": "[uniqueString(subscription().subscriptionId)]",
    "vnetName": "[concat('VNet-', parameters('projectName'), '-', parameters('envi
},
"resources": [
    {
        "type": "Microsoft.Storage/storageAccounts",
        "apiVersion": "2018-07-01",
        "name": "[variables('storageName')]",
        "location": "[resourceGroup().location]",
        "tags": {
            "displayName": "[variables('storageName')]"
        },
        "sku": {
            "name": "Standard_LRS"
        },
        "kind": "StorageV2",
        "resources": [
            {
                "name": "default/templates",
                "type": "blobServices/containers",
                "apiVersion": "2018-07-01",
                "dependsOn": [
                    "[variables('storageName')]"
                ],
                "properties": {
                    "publicAccess": "Blob"
                }
            }
        ]
    },
    {
        "type": "Microsoft.Network/virtualNetworks",
        "apiVersion": "2018-08-01",
        "name": "[variables('vnetName')]",
        "location": "[resourceGroup().location]",
        "tags": {
            "displayName": "[variables('vnetName')]"
        },
        "properties": {
            "addressSpace": {
                "addressPrefixes": [
                    "10.0.0.0/16"
                ]
            },
            "subnets": [
                {
                    "name": "Subnet-1",
                    "properties": {
                        "addressPrefix": "10.0.0.0/24"

                    }
                },
```

```
                {
                    "name": "Subnet-2",
                    "properties": {
                        "addressPrefix": "10.0.1.0/24"
                    }
                }
            ]
        }
    }
],
"outputs": {}
}
```

Deploy ARM template via Portal

1. Open the Azure Portal as T {USERNAME} using T {PASSWORD} as the password.
2. (+) Create a resource -> search for "Template Deployment" - Create
3. Click "Build your own template in the editor"
4. Copy and paste the contents of the newly created `DeploymentMethods.template.json` file
5. Select your existing resource group `{RESOURCE_GROUP_NAME}`
6. Enter a value for `environment`
7. Set the value of `projectName` to `DeploymentMethod1`
8. Agree to the terms and conditions and click Purchase
9. Navigate to the resource group to see the newly created resources

Deploy template file & parameter file via PowerShell

Create parameter file

1. Create a new file in `C:\Lab_Files\M04\S03-Lab1` named `DeploymentMethod2.parameters.json` and open the file in Visual Studio Code
2. Type `armp!` and press `Enter` to insert the ARM parameters skeleton code snippet
3. Using the `new-parameter-value` snippet, add the following parameters to the `"parameters": {},` section
    a. `"environment"`
        i. Use a value of your choice
    b. `"projectName"`
        i. Use a value of `"DeploymentMethod2"`
4. The completed ARM parameter file should look as follows:

`json`

```
    {
```

```
    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentP
    "contentVersion": "1.0.0.0",
    "parameters": {
        "environment": {
            "value": "dev"
        },
        "projectName": {
            "value": "DeploymentMethod2"
        }
    }
}
```

Deploy with PowerShell

1. Open PowerShell in `C:\Lab_Files\M04\S03-Lab1`
2. Authenticate PowerShell to Azure by running `Connect-AzAccount` as T {USERNAME} using T {PASSWORD} as the password.

PowerShell

```
Set-AzContext -Subscription '{SUBSCRIPTION_ID}'
New-AzResourceGroupDeployment -Name 'DeploymentMethod2' -ResourceGroupName '{RE
```

1. Open to the Azure Portal https://portal.azure.com
2. Navigate to the resource group `{RESOURCE_GROUP_NAME}` to see the newly created resources

Deploy template file via PowerShell with parameters in-line

1. Run the following PowerShell commands to deploy the template with parameters in-line

PowerShell

```
New-AzResourceGroupDeployment -Name 'DeploymentMethod3' -ResourceGroupName '{RE
```

1. Open to the Azure Portal https://portal.azure.com
2. Navigate to the resource group `{RESOURCE_GROUP_NAME}` to see the newly created resources

Deploy template & parameters from URI via PowerShell

Create DeploymentMethod4.parameters.json

1. In Visual Studio Code, create a copy of `DeploymentMethod2.parameters.json` named as `DeploymentMethod4.parameters.json`
2. Update `"projectName"` with a value of `"DeploymentMethod4"`

Upload files to blob storage

In order to deploy the template & parameters from URI, they first must be made available at a public accessible URL. We will use the blob container child resource created with Azure Storage account from the last deployment to host these files.

1. Open to the Azure Portal https://portal.azure.com
2. Navigate to the resource group `{RESOURCE_GROUP_NAME}
3. Open the storage account
4. Click Storage Explorer
5. Expand Blob Containers and select templates
6. Upload the following files:
    a. DeploymentMethods.template.json
    b. DeploymentMethod4.parameters.json
7. Select each file and use the Copy URL. Use these values in the PowerShell step below

Deploy with PowerShell

```PowerShell
New-AzResourceGroupDeployment -Name 'DeploymentMethod4' -ResourceGroupName '{RESOU
```

Lab 2 - ARM Templates - Common Deployment Methods & Deployment Order - Linked / Nested Templates

---

Create ARM template

Create ARM template skeleton

1. Open `C:\Lab_Files\M04` in Visual Studio Code and create a subfolder named `S03-Lab1`
2. Create a new file in `C:\Lab_Files\M04\S03-Lab1` named `DeploymentMethods.LinkedNested.template.json` and open the file.
3. Type `arm!` and press `Enter` to insert the ARM template skeleton code snippet

```json
{
    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploy
    "contentVersion": "1.0.0.0",

    "parameters": {},
    "variables": {},
```

```
    "resources": [],
    "outputs": {}
}
```

Add parameters and variables

1. Add the following parameters to the `"parameters": {},` section with descriptions of your choice. (**NOTE:** Use snippets to make this easier)

    a. `"environment"`
    b. `"projectName"`

```json
"parameters": {
    "environment": {
        "type": "string",
        "metadata": {
            "description": "Environment (Dev/QA/Prod)"
        }
    },
    "projectName": {
        "type": "string",
        "metadata": {
            "description": "Project Name"
        }
    }
},
```

2. Add the following variables and values to the `"variables": {},` section

    a. `"availSetName": "[concat('AvailSet-', parameters('projectName'), '-', parameters('environment'))]"`
    b. `"linkedTemplateURI": "DeploymentMethods.template.json_URL_HERE"`
        i. Use the blob storage URL of DeploymentMethods.template.json from the previous lab

```json
"variables": {
    "availSetName": "[concat('AvailSet-', parameters('projectName'), '-', par
    "linkedTemplateURI": "DeploymentMethods.template.json_URL_HERE"
},
```

Add linked template resource

1. Using the `arm-nested` snippet, add a linked template resource to the `"resources": [],` section. (**NOTE:** Depending on the version of the snippet extension, the snippet may be referenced

by another name such as `arm-nest`)

2. Change the value of `"name"` and from `NestedDeployment1` to `"LinkedTemplate"`
3. If there is a `"tags": {}` section, remove it from the resource
4. Update the value of `"uri"` from `"[concat(parameters('_artifactsLocation'), '/NestedTemplates/${NestedTemplate.json}', parameters('_artifactsLocationSasToken'))]"` to `"[variables('linkedTemplateURI')]"`
5. Add the following parameters to the `"parameters": {},` section of the **linked template resource**,
    a. `"environment"`
        i. `"value": "[parameters('environment')]"`
    b. `"projectName"`
        i. `"value": "[parameters('projectName')]"`

```json
  "resources": [
      {
          "type": "Microsoft.Resources/deployments",
          "apiVersion": "2018-05-01",
          "name": "LinkedTemplate",
          "properties": {
              "mode": "Incremental",
              "templateLink": {
                  "uri": "[variables('linkedTemplateURI')]",
                  "contentVersion": "1.0.0.0"
              },
              "parameters": {
                  "environment": {
                      "value": "[parameters('environment')]"
                  },
                  "projectName": {
                      "value": "[parameters('projectName')]"
                  }
              }
          }
      }
  ],
```

Add nested resource

1. Using the `arm-nested` snippet, add a nested resource to the `"resources": [],` section below the linked template resource
2. Change the value of `"name"` and from `NestedDeployment1` to `"NestedDeployment"`
3. If there is a `"tags": {}` section, remove it from the resource
4. Remove the `"templateLink": {}` and `"parameters": {}` sections from the nested resource's `"properties": {}` section

5. Add a new `"template": {}` section to the nested resource's `"properties": {}` section
6. Inside the `"template": {}` section, add the following properties
    a. `"$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#"`
    b. `"contentVersion": "1.0.0.0"`
7. Inside the `"template": {}` section, add a `"resources": []` array below the two properties above

```json
{
    "type": "Microsoft.Resources/deployments",
    "apiVersion": "2018-05-01",
    "name": "NestedDeployment",
    "properties": {
        "mode": "Incremental",
        "template": {
            "$schema": "https://schema.management.azure.com/schemas/2015-01-01/
            "contentVersion": "1.0.0.0",
            "resources": []
        }
    }
}
```

8. Using the `arm-availability-set` snippet, add an Availability Set resource to the `"resources": [],` section of the **nested resource**. (**NOTE:** Depending on the version of the snippet extension, the snippet may be referenced by another name such as `arm-avail`)
9. Change the values of `"name"` and `"displayName"` from `AvailabilitySet1` to `"[variables('availSetName')]"`

Review completed ARM template

The completed ARM template should look as follows:

```json
{
    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentT
    "contentVersion": "1.0.0.0",
    "parameters": {
        "environment": {
            "type": "string",
            "metadata": {
                "description": "Environment (Dev/QA/Prod)"
            }
        },
        "projectName": {

            "type": "string",
            "metadata": {
```

```
                "description": "Project Name"
            }
        }
    },
    "variables": {
        "availSetName": "[concat('AvailSet-', parameters('projectName'), '-', para
        "linkedTemplateURI": "DeploymentMethods.template.json_URL_HERE"
    },
    "resources": [
        {
            "type": "Microsoft.Resources/deployments",
            "apiVersion": "2018-05-01",
            "name": "LinkedTemplate",
            "properties": {
                "mode": "Incremental",
                "templateLink": {
                    "uri": "[variables('linkedTemplateURI')]",
                    "contentVersion": "1.0.0.0"
                },
                "parameters": {
                    "environment": {
                        "value": "[parameters('environment')]"
                    },
                    "projectName": {
                        "value": "[parameters('projectName')]"
                    }
                }
            }
        },
        {
            "type": "Microsoft.Resources/deployments",
            "apiVersion": "2018-05-01",
            "name": "NestedDeployment",
            "properties": {
                "mode": "Incremental",
                "template": {
                    "$schema": "https://schema.management.azure.com/schemas/2015-0
                    "contentVersion": "1.0.0.0",
                    "resources": [
                        {
                            "type": "Microsoft.Compute/availabilitySets",
                            "apiVersion": "2015-06-15",
                            "name": "[variables('availSetName')]",
                            "location": "[resourceGroup().location]",
                            "tags": {
                                "displayName": "[variables('availSetName')]"
                            },
                            "properties": {}
                        }
                    ]
                }
            }
        }
    ],
    "outputs": {}
```

```
}
```

Deploy template and review deployments

1. Deploy DeploymentMethods.LinkedNested.template.json with a method of your choice.
    a. `"projectName"` should have a value of `"DeploymentMethod5"`
2. Open to the Azure Portal https://portal.azure.com
3. Navigate to the resource group `{RESOURCE_GROUP_NAME}` to see the newly created resources
4. Click "Deployments" under the "Settings" section
5. Review the three deployments created by this deployment operation:
    a. Master deployment used for deploying DeploymentMethods.LinkedNested.template.json
    b. LinkedTemplate deployment
    c. NestedDeployment deployment