Azure VM Deployment Workshop

Infrastructure as Code with ARM Templates and Azure Pipelines

Workshop Overview

Duration: 3-4 hours
Level: Intermediate
Prerequisites:

- Azure subscription with contributor access
- Azure DevOps organization (or GitHub account)
- Basic understanding of JSON and YAML
- Azure CLI installed locally

Module 1: ARM Template Fundamentals (45 minutes)

1.1 Understanding ARM Template Structure

Objective: Learn the core components of an ARM template

Key Concepts:

- Schema and API versions
- Parameters, variables, resources, and outputs
- Resource dependencies
- Template functions

Hands-on Exercise: Create a basic ARM template structure for a VM deployment.

Module 2: Building Your First VM Template (60 minutes)

2.1 Create the Base Template

File: `azuredeploy.json`

```
json
```

```
{
 "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
 "contentVersion": "1.0.0.0",
 "parameters": {
  "vmName": {
   "type": "string",
   "metadata": {
    "description": "Name of the virtual machine"
   }
  },
  "adminUsername": {
   "type": "string",
   "metadata": {
    "description": "Admin username for the VM"
   }
  },
  "adminPassword": {
   "type": "securestring",
   "metadata": {
    "description": "Admin password for the VM"
   }
  },
  "vmSize": {
   "type": "string",
   "defaultValue": "Standard_B2s",
   "allowedValues": [
    "Standard_B1s",
    "Standard_B2s",
    "Standard_D2s_v3"
   ],
   "metadata": {
    "description": "Size of the VM"
   }
  },
  "location": {
   "type": "string",
   "defaultValue": "[resourceGroup().location]",
   "metadata": {
    "description": "Location for all resources"
   }
  },
  "operatingSystem": {
   "type": "string",
   "defaultValue": "Windows",
   "allowedValues": [
    "Windows",
```

```
      "Ubuntu"
    ],
    "metadata": {
      "description": "Operating system type"
    }
  }
},
"variables": {
  "vnetName": "[concat(parameters('vmName'), '-vnet')]",
  "subnetName": "default",
  "nicName": "[concat(parameters('vmName'), '-nic')]",
  "publicIPName": "[concat(parameters('vmName'), '-pip')]",
  "nsgName": "[concat(parameters('vmName'), '-nsg')]",
  "addressPrefix": "10.0.0.0/16",
  "subnetPrefix": "10.0.0.0/24",
  "windowsImage": {
    "publisher": "MicrosoftWindowsServer",
    "offer": "WindowsServer",
    "sku": "2022-Datacenter",
    "version": "latest"
  },
  "ubuntuImage": {
    "publisher": "Canonical",
    "offer": "0001-com-ubuntu-server-jammy",
    "sku": "22_04-lts-gen2",
    "version": "latest"
  },
  "imageReference": "[if(equals(parameters('operatingSystem'), 'Windows'), variables('windowsImage'), variables('ubuntuIm
},
"resources": [
  {
    "type": "Microsoft.Network/networkSecurityGroups",
    "apiVersion": "2023-04-01",
    "name": "[variables('nsgName')]",
    "location": "[parameters('location')]",
    "properties": {
      "securityRules": [
        {
          "name": "default-allow-rdp",
          "properties": {
            "priority": 1000,
            "protocol": "Tcp",
            "access": "Allow",
            "direction": "Inbound",
            "sourceAddressPrefix": "*",
            "sourcePortRange": "*",
            "destinationAddressPrefix": "*",
```

```
        "destinationPortRange": "3389"
       }
      },
      {
       "name": "default-allow-ssh",
       "properties": {
        "priority": 1001,
        "protocol": "Tcp",
        "access": "Allow",
        "direction": "Inbound",
        "sourceAddressPrefix": "*",
        "sourcePortRange": "*",
        "destinationAddressPrefix": "*",
        "destinationPortRange": "22"
       }
      }
    ]
   }
  },
  {
   "type": "Microsoft.Network/publicIPAddresses",
   "apiVersion": "2023-04-01",
   "name": "[variables('publicIPName')]",
   "location": "[parameters('location')]",
   "sku": {
    "name": "Basic"
   },
   "properties": {
    "publicIPAllocationMethod": "Dynamic"
   }
  },
  {
   "type": "Microsoft.Network/virtualNetworks",
   "apiVersion": "2023-04-01",
   "name": "[variables('vnetName')]",
   "location": "[parameters('location')]",
   "dependsOn": [
    "[resourceId('Microsoft.Network/networkSecurityGroups', variables('nsgName'))]"
   ],
   "properties": {
    "addressSpace": {
     "addressPrefixes": [
      "[variables('addressPrefix')]"
     ]
    },
    "subnets": [
     {
```

```
        "name": "[variables('subnetName')]",
        "properties": {
          "addressPrefix": "[variables('subnetPrefix')]",
          "networkSecurityGroup": {
            "id": "[resourceId('Microsoft.Network/networkSecurityGroups', variables('nsgName'))]"
          }
        }
      }
    ]
  }
},
{
  "type": "Microsoft.Network/networkInterfaces",
  "apiVersion": "2023-04-01",
  "name": "[variables('nicName')]",
  "location": "[parameters('location')]",
  "dependsOn": [
    "[resourceId('Microsoft.Network/publicIPAddresses', variables('publicIPName'))]",
    "[resourceId('Microsoft.Network/virtualNetworks', variables('vnetName'))]"
  ],
  "properties": {
    "ipConfigurations": [
      {
        "name": "ipconfig1",
        "properties": {
          "privateIPAllocationMethod": "Dynamic",
          "publicIPAddress": {
            "id": "[resourceId('Microsoft.Network/publicIPAddresses', variables('publicIPName'))]"
          },
          "subnet": {
            "id": "[resourceId('Microsoft.Network/virtualNetworks/subnets', variables('vnetName'), variables('subnetName'))]"
          }
        }
      }
    ]
  }
},
{
  "type": "Microsoft.Compute/virtualMachines",
  "apiVersion": "2023-03-01",
  "name": "[parameters('vmName')]",
  "location": "[parameters('location')]",
  "dependsOn": [
    "[resourceId('Microsoft.Network/networkInterfaces', variables('nicName'))]"
  ],
  "properties": {
    "hardwareProfile": {
```

```
        "vmSize": "[parameters('vmSize')]"
      },
      "osProfile": {
        "computerName": "[parameters('vmName')]",
        "adminUsername": "[parameters('adminUsername')]",
        "adminPassword": "[parameters('adminPassword')]"
      },
      "storageProfile": {
        "imageReference": "[variables('imageReference')]",
        "osDisk": {
          "createOption": "FromImage",
          "managedDisk": {
            "storageAccountType": "Standard_LRS"
          }
        }
      },
      "networkProfile": {
        "networkInterfaces": [
          {
            "id": "[resourceId('Microsoft.Network/networkInterfaces', variables('nicName'))]"
          }
        ]
      }
    }
  ],
  "outputs": {
    "vmId": {
      "type": "string",
      "value": "[resourceId('Microsoft.Compute/virtualMachines', parameters('vmName'))]"
    },
    "publicIP": {
      "type": "string",
      "value": "[reference(resourceId('Microsoft.Network/publicIPAddresses', variables('publicIPName'))).ipAddress]"
    }
  }
}
```

2.2 Create Parameter Files

File: `azuredeploy.parameters.dev.json`

```
json
```

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "vmName": {
      "value": "vm-workshop-dev"
    },
    "adminUsername": {
      "value": "azureuser"
    },
    "adminPassword": {
      "reference": {
        "keyVault": {
          "id": "/subscriptions/{subscription-id}/resourceGroups/{rg-name}/providers/Microsoft.KeyVault/vaults/{vault-name}"
        },
        "secretName": "vmAdminPassword"
      }
    },
    "vmSize": {
      "value": "Standard_B2s"
    },
    "operatingSystem": {
      "value": "Ubuntu"
    }
  }
}
```

File: **azuredeploy.parameters.prod.json**

```
json
```

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "vmName": {
      "value": "vm-workshop-prod"
    },
    "adminUsername": {
      "value": "azureuser"
    },
    "adminPassword": {
      "reference": {
        "keyVault": {
          "id": "/subscriptions/{subscription-id}/resourceGroups/{rg-name}/providers/Microsoft.KeyVault/vaults/{vault-name}"
        },
        "secretName": "vmAdminPassword"
      }
    },
    "vmSize": {
      "value": "Standard_D2s_v3"
    },
    "operatingSystem": {
      "value": "Ubuntu"
    }
  }
}
```

2.3 Exercise: Manual Deployment

Task: Deploy the template using Azure CLI

```bash
```

```
# Login to Azure
az login

# Create resource group
az group create --name rg-workshop-dev --location eastus

# Validate template
az deployment group validate \
  --resource-group rg-workshop-dev \
  --template-file azuredeploy.json \
  --parameters azuredeploy.parameters.dev.json

# Deploy template
az deployment group create \
  --resource-group rg-workshop-dev \
  --template-file azuredeploy.json \
  --parameters azuredeploy.parameters.dev.json \
  --name vm-deployment-001
```

Discussion Points:

- What resources were created?
- How long did deployment take?
- What happens if you run deployment again?

Module 3: Azure Key Vault Integration (30 minutes)

3.1 Setup Key Vault

Objective: Securely store sensitive parameters

Steps:

```
bash
```

```
# Create Key Vault
az keyvault create \
  --name kv-workshop-unique \
  --resource-group rg-workshop-dev \
  --location eastus

# Store admin password
az keyvault secret set \
  --vault-name kv-workshop-unique \
  --name vmAdminPassword \
  --value 'YourSecurePassword123!'

# Grant access to service principal (for pipeline)
az keyvault set-policy \
  --name kv-workshop-unique \
  --spn {service-principal-id} \
  --secret-permissions get list
```

3.2 Exercise: Update Parameter Files

Update your parameter files to reference Key Vault secrets instead of hardcoded passwords.

Module 4: Azure DevOps Pipeline Setup (60 minutes)

4.1 Create Azure Pipeline

File: **azure-pipelines.yml**

```yaml
```

```yaml
trigger:
  branches:
    include:
    - main
  paths:
    include:
    - templates/*
    - pipelines/*

variables:
  azureSubscription: 'Azure-ServiceConnection'
  location: 'eastus'

stages:
- stage: Validate
  displayName: 'Validate Templates'
  jobs:
  - job: ValidateARM
    displayName: 'Validate ARM Template'
    pool:
      vmImage: 'ubuntu-latest'
    steps:
    - task: AzureCLI@2
      displayName: 'Validate ARM Template Syntax'
      inputs:
        azureSubscription: $(azureSubscription)
        scriptType: 'bash'
        scriptLocation: 'inlineScript'
        inlineScript: |
          az deployment group validate \
            --resource-group rg-workshop-dev \
            --template-file templates/azuredeploy.json \
            --parameters templates/azuredeploy.parameters.dev.json

    - task: RunARMTTKTests@1
      displayName: 'Run ARM-TTK Tests'
      inputs:
        templatelocation: '$(System.DefaultWorkingDirectory)/templates'
        resultLocation: '$(System.DefaultWorkingDirectory)/results'

- stage: DeployDev
  displayName: 'Deploy to Development'
  dependsOn: Validate
  condition: succeeded()
  jobs:
  - deployment: DeployVM
```

```yaml
    displayName: 'Deploy VM to Dev'
    pool:
      vmImage: 'ubuntu-latest'
    environment: 'Development'
    strategy:
      runOnce:
        deploy:
          steps:
          - checkout: self

          - task: AzureResourceManagerTemplateDeployment@3
            displayName: 'Deploy ARM Template'
            inputs:
              deploymentScope: 'Resource Group'
              azureResourceManagerConnection: $(azureSubscription)
              subscriptionId: '$(subscriptionId)'
              action: 'Create Or Update Resource Group'
              resourceGroupName: 'rg-workshop-dev'
              location: $(location)
              templateLocation: 'Linked artifact'
              csmFile: 'templates/azuredeploy.json'
              csmParametersFile: 'templates/azuredeploy.parameters.dev.json'
              deploymentMode: 'Incremental'
              deploymentName: 'vm-deployment-$(Build.BuildId)'
              deploymentOutputs: 'armOutputs'

          - task: PowerShell@2
            displayName: 'Parse ARM Outputs'
            inputs:
              targetType: 'inline'
              script: |
                $outputs = ConvertFrom-Json '$(armOutputs)'
                $vmId = $outputs.vmId.value
                $publicIP = $outputs.publicIP.value
                Write-Host "VM ID: $vmId"
                Write-Host "Public IP: $publicIP"
                Write-Host "##vso[task.setvariable variable=vmPublicIP;isOutput=true]$publicIP"

  - stage: DeployProd
    displayName: 'Deploy to Production'
    dependsOn: DeployDev
    condition: and(succeeded(), eq(variables['Build.SourceBranch'], 'refs/heads/main'))
    jobs:
    - deployment: DeployVM
      displayName: 'Deploy VM to Prod'
      pool:
        vmImage: 'ubuntu-latest'
```

```
    environment: 'Production'
    strategy:
      runOnce:
        deploy:
          steps:
          - checkout: self

          - task: AzureResourceManagerTemplateDeployment@3
            displayName: 'Deploy ARM Template'
            inputs:
              deploymentScope: 'Resource Group'
              azureResourceManagerConnection: $(azureSubscription)
              subscriptionId: '$(subscriptionId)'
              action: 'Create Or Update Resource Group'
              resourceGroupName: 'rg-workshop-prod'
              location: $(location)
              templateLocation: 'Linked artifact'
              csmFile: 'templates/azuredeploy.json'
              csmParametersFile: 'templates/azuredeploy.parameters.prod.json'
              deploymentMode: 'Incremental'
              deploymentName: 'vm-deployment-$(Build.BuildId)'
```

4.2 Setup Service Connection

Steps:

1. Go to Azure DevOps Project Settings
2. Navigate to Service Connections
3. Create new Azure Resource Manager connection
4. Choose Service Principal (automatic)
5. Select subscription and resource group scope
6. Name it 'Azure-ServiceConnection'

4.3 Create Environments

Steps:

1. Go to Pipelines > Environments
2. Create 'Development' environment
3. Create 'Production' environment
4. Add approval gates to Production

4.4 Exercise: Run Your Pipeline

Tasks:

1. Commit your code to Azure Repos
2. Create and run the pipeline
3. Monitor deployment stages
4. Verify resources in Azure Portal
5. Approve production deployment

Module 5: Advanced Scenarios (45 minutes)

5.1 Add VM Extensions

Extension for Windows - Install IIS:

```json
{
  "type": "Microsoft.Compute/virtualMachines/extensions",
  "apiVersion": "2023-03-01",
  "name": "[concat(parameters('vmName'), '/InstallIIS')]",
  "location": "[parameters('location')]",
  "dependsOn": [
    "[resourceId('Microsoft.Compute/virtualMachines', parameters('vmName'))]"
  ],
  "properties": {
    "publisher": "Microsoft.Compute",
    "type": "CustomScriptExtension",
    "typeHandlerVersion": "1.10",
    "autoUpgradeMinorVersion": true,
    "settings": {
      "commandToExecute": "powershell Add-WindowsFeature Web-Server"
    }
  }
}
```

Extension for Linux - Install Apache:

```json
```

```
{
  "type": "Microsoft.Compute/virtualMachines/extensions",
  "apiVersion": "2023-03-01",
  "name": "[concat(parameters('vmName'), '/InstallApache')]",
  "location": "[parameters('location')]",
  "dependsOn": [
    "[resourceId('Microsoft.Compute/virtualMachines', parameters('vmName'))]"
  ],
  "properties": {
    "publisher": "Microsoft.Azure.Extensions",
    "type": "CustomScript",
    "typeHandlerVersion": "2.1",
    "autoUpgradeMinorVersion": true,
    "settings": {
      "commandToExecute": "apt-get update && apt-get install -y apache2"
    }
  }
}
```

5.2 Add Managed Identity

```
json

{
  "type": "Microsoft.Compute/virtualMachines",
  "identity": {
    "type": "SystemAssigned"
  }
}
```

5.3 Exercise: Template Enhancement

Tasks:

1. Add a VM extension based on OS type
2. Enable managed identity
3. Add tags for cost tracking
4. Configure backup policy

Module 6: Testing and Validation (30 minutes)

6.1 ARM Template Test Toolkit (ARM-TTK)

Install and Run:

```powershell
# Install ARM-TTK
Import-Module .\arm-ttk\arm-ttk.psd1

# Run tests
Test-AzTemplate -TemplatePath .\templates\
```

6.2 What-If Deployment

```bash
# Preview changes before deployment
az deployment group what-if \
  --resource-group rg-workshop-dev \
  --template-file azuredeploy.json \
  --parameters azuredeploy.parameters.dev.json
```

6.3 Exercise: Add Validation Tests

Add pre-deployment validation steps to your pipeline.

Module 7: Monitoring and Rollback (20 minutes)

7.1 View Deployment History

```bash
# List deployments
az deployment group list \
  --resource-group rg-workshop-dev \
  --output table

# Show deployment details
az deployment group show \
  --resource-group rg-workshop-dev \
  --name vm-deployment-001
```

7.2 Rollback Strategy

```bash
# Redeploy previous successful deployment
az deployment group create \
  --resource-group rg-workshop-dev \
  --name vm-deployment-rollback \
  --mode Complete \
  --rollback-on-error
```

Workshop Challenges

Challenge 1: Multi-VM Deployment

Modify the template to deploy multiple VMs using copy loops.

Challenge 2: Add Load Balancer

Extend the template to include an Azure Load Balancer distributing traffic across 2 VMs.

Challenge 3: Implement Blue-Green Deployment

Create a pipeline strategy for zero-downtime deployments.

Challenge 4: Add Monitoring

Configure Azure Monitor and alerts for the deployed VM.

Best Practices Checklist

- ✅ Use parameter files for environment-specific values
- ✅ Store secrets in Azure Key Vault
- ✅ Implement validation stages in pipelines
- ✅ Use managed identities instead of passwords where possible
- ✅ Tag all resources for cost management
- ✅ Enable diagnostic logging
- ✅ Use incremental deployment mode for updates
- ✅ Implement approval gates for production
- ✅ Version control all templates and parameters
- ✅ Document template parameters and outputs

Resources

Documentation:

- ARM Template Reference
- Azure Pipelines Documentation
- ARM Template Best Practices

Tools:

- ARM Template Viewer (VS Code)
- Azure Resource Manager Tools (VS Code)
- ARM Template Test Toolkit

Workshop Completion Certificate

Upon completing all modules and challenges, you will have:

- Created production-ready ARM templates for VM deployment
- Built automated CI/CD pipelines with Azure DevOps
- Implemented security best practices with Key Vault
- Learned testing and validation techniques
- Understood deployment strategies and rollback procedures

Next Steps:

- Explore Bicep as an alternative to ARM templates
- Learn about Azure Policy for governance
- Investigate Terraform for multi-cloud scenarios
- Study Azure Landing Zones for enterprise deployments