Bicep – An Overview

What it is

- **Domain** specific Language
- Transparent Abstraction over ARM Templates
    - At runtime, **Transpiles** into *ARM Template JSON* files
- **Declarative**, **Clean** syntax to deploy resources onto Azure
    - **Easy** to understand and straightforward to **Learn**
- Support for handling **Repetitive** deployments - **for** loops
- **Conditional** deployments made easy with - **if** check
- **Modularisation** of the deployment - **module** blocks
- **Decompilation** feature - Convert existing *ARM* templates to *Bicep* templates

What it is NOT

- General purpose language to meet any need
- Pre or Post-Bicep execution tasks might be needed
- First-class provider model for non-Azure related tasks

Why Bicep

- Day-0 resource provider support. Any Azure resource — whether in private or public preview or GA — can be provisioned using Bicep
- Much simpler syntax compared to equivalent ARM Template JSON
- No state or state files to manage
    - All state is stored in Azure
    - Easy to collaborate and make changes to resources confidently

- VS Code extension for Bicep
  - Easy to write Bicep script
  - Support for Validation and Intellisense
- Easily break apart your code with native modules
- Supported by *Microsoft support* and 100% free to use

Known limitations

- No support for single-line object and arrays (i.e. `['a', 'b', 'c']`)
- Bicep is newline sensitive. We are exploring ways we can remove/relax this restriction
- No support for the concept of apiProfile which is used to map a single apiProfile to a set apiVersion for each resource type

Deep-dive

Structure

```
targetScope = '<scope>'

@<decorator>(<argument>)
param <parameter-name> <parameter-data-type> = <default-value>

var <variable-name> = <variable-value>

resource <resource-symbolic-name> '<resource-type>@<api-version>' = {
  <resource-properties>
}

module <module-symbolic-name> '<path-to-file>' = {
  name: '<linked-deployment-name>'
  params: {
    <parameter-names-and-values>
  }
}
```

```
// deploy to different scope
module <module-symbolic-name> '<path-to-file>' = {
  name: '<linked-deployment-name>'
  scope: <scope-object>
  params: {
    <parameter-names-and-values>
  }
}

output <output-name> <output-data-type> = <output-value>

// iterative output
output <output-name> array = [for <item> in <collection>: {
  <output-properties>
}]
```

Parameters

- Used to make templates dynamically configurable
- **Parameters** can be part of the *teamplate* file itself

```
param vnetName string
param vnetPrefix string
param subnetName string
param subnetPrefix string
param appgwSubnetName string
param appgwSubnetPrefix string
```

- **Simplicity** is the key - as **params** have been declared as normally been done in a PowerShell or bash script!
- **Parameters** can be in a separate parameter file - **<file_name>.parameters.com**

```
{
    "$schema": "https://schema.management.azure.com/schemas/2015-01-
  01/deploymentParameters.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
        "rootCertData": {
          "reference": {
              "keyVault": {
                  "id": "<keyVault_id>"
                },
            "secretName": "<secret_Name>"
        }
      },
        "certData": {
          "reference": {
              "keyVault": {
                  "id": "<keyVault_id>"
                },
            "secretName": "<secret_Name>"
        }
      },
      "certPassword": {
          "reference": {
              "keyVault": {
                  "id": "<keyVault_id>"
                },
            "secretName": "<secret_Name>"
        }
      }
    }
  }
```

- **Parameter File** approach is mostly used in passing secured parameters - which would be discussed later
- **Decorstors**

```
@description('Must be at least Standard_A3 to support 2 NICs.')
param virtualMachineSize string = 'Standard_DS1_v2'

@sys.description('The description of the instance to display.')
param description string
```

- **Secure** Pararmeters

```
@secure()
param rootCertData string

@secure()
param certData string

@secure()
param certPassword string
```

- **Allowed** values for a *parameter*

```
@allowed([
  'Standard'
  'Standard_v2'
  'WAF'
  'WAF_v2'
])
param tierSkuName string = 'WAF_v2'
```

- **Default** values for a *parameter*

```
param demoParam string = 'Contoso'
param location string = resourceGroup().location
```

- **Length Constraints**

```
@minLength(3)
@maxLength(24)
param storageAccountName string
```

- **Objects** as *parameter*

```
param vNetSettings object = {
  name: 'VNet1'
  location: 'eastus'
  addressPrefixes: [
    {
      name: 'firstPrefix'
      addressPrefix: '10.0.0.0/22'
    }
  ]
  subnets: [
    {
      name: 'firstSubnet'
      addressPrefix: '10.0.0.0/24'
    }
    {
      name: 'secondSubnet'
      addressPrefix: '10.0.1.0/24'
    }
  ]
}
```

  - Use this **Object** as *parameter*

```
resource vnet 'Microsoft.Network/virtualNetworks@2020-06-01' = {
  name: vNetSettings.name
  location: vNetSettings.location
  properties: {
    addressSpace: {
      addressPrefixes: [
        vNetSettings.addressPrefixes[0].addressPrefix
```

```
          ]
        }
        subnets: [
          {
            name: vNetSettings.subnets[0].name
            properties: {
              addressPrefix: vNetSettings.subnets[0].addressPrefix
            }
          }
          {
            name: vNetSettings.subnets[1].name
            properties: {
              addressPrefix: vNetSettings.subnets[1].addressPrefix
            }
          }
        ]
      }
    }
```

Variables

- **Derived** values from multiple *parameters* and other **Variables**

```
var appGwId = resourceId('Microsoft.Network/applicationGateways',
'${applicationGatewayName}')
var appGwIPConfigName = '${applicationGatewayName}-ipc'
```

- Use of the **Variables**

```
param rgLocation string
param storageNamePrefix string = 'STG'

var storageName =
'${toLower(storageNamePrefix)}${uniqueString(resourceGroup().id)}'
```

```
resource demoAccount 'Microsoft.Storage/storageAccounts@2021-02-01'
= {
  name: storageName
  location: rgLocation
  kind: 'Storage'
  sku: {
    name: 'Standard_LRS'
  }
}

output stgOutput string = storageName
```

- Variables can use Resource Manager functions - e.g. **resourceId()** function
- Important point to note is the interpolation of strings; no need to use **concat()** function as in ARM template

  ```
  ${applicationGatewayName}-ipc
  ```

  ```
  var appGwProbeRef = '${appGwId}/probes/${appGwProbeName}'
  ```

Resources

- Describes the components of the **Resources** to be deployed
- Completetly declarative syntax
- Modular approach in declaring resources; makes it more manageable, readable
- e.g. **VNET** deployment

```
resource vnetName_resource 'Microsoft.Network/virtualNetworks@2018-
10-01' = {
  name: vnetName
  location: location
  properties: {
    addressSpace: {
      addressPrefixes: [
        vnetPrefix
      ]
    }
  }
}
```

Corresponding **Subnet** deployments

```
resource vnetName_subnetName
'Microsoft.Network/virtualNetworks/subnets@2018-10-01' = {
  parent: vnetName_resource
  name: subnetName
  properties: {
    addressPrefix: subnetPrefix
  }
}

resource vnetName_appgwSubnetName
'Microsoft.Network/virtualNetworks/subnets@2018-10-01' = {
  parent: vnetName_resource
  name: appgwSubnetName
  properties: {
    addressPrefix: appgwSubnetPrefix
  }
  dependsOn: [
    vnetName_subnetName
  ]
}
```

- Compare this with an **ARM Template**; benefits are evident

```
{
    "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
      "vnetName": {
        "type": "string",
        "defaultValue": "aks-vnet"
      },
      "vnetPrefix": {
        "type": "string",
        "defaultValue": "173.0.0.0/16"
      },
      "subnetName": {
        "type": "string",
        "defaultValue": "aks-subnet"
      },
      "subnetPrefix": {
        "type": "string",
        "defaultValue": "173.0.0.0/22"
      },
      "appgwSubnetName": {
        "type": "string",
        "defaultValue": "appgw-subnet"
      },
      "appgwSubnetPrefix": {
        "type": "string",
        "defaultValue": "173.0.4.0/27"
      },
      "location": {
        "type": "string",
        "defaultValue": "[resourceGroup().location]"
      }
    },
    "variables": {},
    "resources": [
      {
        "apiVersion": "2018-10-01",
        "type": "Microsoft.Network/virtualNetworks",
        "name": "[parameters('vnetName')]",
        "location": "[parameters('location')]",
        "properties": {
          "addressSpace": {
            "addressPrefixes": [
              "[parameters('vnetPrefix')]"
            ]
          }
        },
        "resources": [
          {
            "apiVersion": "2018-10-01",
            "type": "subnets",
            "location": "[parameters('location')]",
            "name": "[parameters('subnetName')]",
            "dependsOn": [
              "[parameters('vnetName')]"
            ],
            "properties": {
              "addressPrefix": "[parameters('subnetPrefix')]"
            }
          },
          {
            "apiVersion": "2018-10-01",
            "type": "subnets",
            "location": "[parameters('location')]",
            "name": "[parameters('appgwSubnetName')]",
            "dependsOn": [
              "[parameters('vnetName')]",
              "[parameters('subnetName')]"
            ],
            "properties": {
              "addressPrefix": "[parameters('appgwSubnetPrefix')]"
            }
          }
        ]
      }
    ]
  }
```

- The steep hierarchial decalration is replaced by a more modular way of describing deloyment components

Repetitions

- This has always been the biggest pain area or ARM templates. ARM has **Copy** section to accommodate this requirement but it was tough to implement and manage
- Below 2 examples would show how Bicep handles loops and how easy it is to implement and manage
    - Multiple deployment of the same resource e.g. deploying *azure storage account* with multiple *blob containers*
        - **Multiple Blob Containers**

```
param blobContainers array
.....

resource storageAccountName_default_blobContainers
'Microsoft.Storage/storageAccounts/blobServices/containers@20
21-02-01' = [for blob in blobContainers: {
  name: '${storageAccountName}/default/${blob}'
  dependsOn: [
    storageAccountName_resource
]
}]
```

- **Multiple Queue Containers**

```
param queues array
......

resource storageAccountName_default_queueContainers
'Microsoft.Storage/storageAccounts/queueServices/queues@2021-02-
01' = [for queue in queues: {
  name: '${storageAccountName}/default/${queue}'
  dependsOn: [
  storageAccountName_resource
  ]
}]
```

- Multiple deployment of the child components under a parent resource e.g. deploying various sub-cmponents of *Application Gateway*
  - **Backend Http Settings**

```
resource applicationGateway
'Microsoft.Network/applicationGateways@2020-05-01' = {
  name: applicationGatewayName
  location: location
  properties: {
    backendHttpSettingsCollection: [for item in
httpsListenerNames: {
      name: '${item}-${appGwBackendHttpSettingsName}'
      properties: {
        port: backendPort
        protocol: backendProtocol
        cookieBasedAffinity: cookieBasedAffinity
        hostName:
'${item}${appGwBackendHttpSettingsHostName}'
```

```
          probeEnabled: true
          probe: {
            id: appGwProbeRef
          }
        trustedRootCertificates: [
            {
            id:
'${appGwId}/trustedRootCertificates/${appGwTrustedRootCertNam
e}'
            }
          ]
        }
      }]
........
```

- Just as a comparison, please refer the following links to see how **Bicep** facilitates the deployment of complex resources like *Application Gateway*
  - *Application Gateway* with **ARM** - https://github.com/monojit18/ARM-Projects/blob/main/AppGW/aksauto-appgw-deploy.json
  - *Application Gateway* with **Bicep** - https://github.com/monojit18/ARM-Projects/blob/main/AppGW/Bicep/aksauto-appgw-deploy.bicep

Conditions

- *ARM* used to handle this thru an in-built variable **condition**
- In *Bicep* this is now easier with an **if condition** check

```
resource vnetName_resource 'Microsoft.Network/virtualNetworks@2018-
10-01' = (if location == 'eastus') {
  name: vnetName
  location: location
  properties: {
    addressSpace: {
      addressPrefixes: [
        vnetPrefix
      ]
    }
  }
}
```

New Or Existing

- Although *ARM* dpeloyment is by default Incremental and hence it either creates a new resource or updates and existing one. There are scenarios where this might not work!
  - Virtual Network having Subnet already mapped to a resource like AKS or Application Gateway. Trying to add a new Subnet would fail as ARM has to refresh all its Subnets' list
  - Similarly for the Storage accounst with existing Containers - *Blob* or *Queue*
- **ARM** used to handle this using some variable, say, **newOrExisting** and check for its value - **new** may be; and decide whether to creqte a new Resource of just skipping that!

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "deployZone": {
      "type": "bool"
    }
  },
  "functions": [],
  "resources": [
    {
      "condition": "[parameters('deployZone')]",
      "type": "Microsoft.Network/dnsZones",
      "apiVersion": "2018-05-01",
      "name": "myZone",
      "location": "global"
    }
  ]
}
```

- **Bicep** has a nicer way of doing this; here also through **if Condition** check

```
@allowed([
  'new'
  'existing'
])
param newOrExisting string = 'new'

resource storageAccountName_resource
'Microsoft.Storage/storageAccounts@2019-06-01' = if (newOrExisting
== 'new') {
  name: storageAccountName
  location: location
  kind: 'StorageV2'
  sku: {
    name: 'Standard_LRS'
  }
  properties: {
    accessTier: 'Cool'
  }
}
```

Referring Existing Resources

- **Deployment** of a new resource might need to refer to an existing resource; e.g. *Storage account with Virtual Newtork integration needs to refer to an existing Network*

```
resource vnetName_resource 'Microsoft.Network/virtualNetworks@2018-
10-01' existing = {
  name: vnetName
}

output vnetId string = vnetName_resource.id
```

- **Virtual Network Identifier** (vnetId) is retrieved through **resourceId**

Modules

- How to write a reusable piece of deployment code and then use it as a pluggable component?
- This was missing in *ARM* and is a popular concept in tools like **Terraform**
- **Bicep** provides **module** keyword to refer to any other **.bicep* script

```
@minLength(3)
@maxLength(11)
param nameSuffix string
param location string = resourceGroup().location

module stgModule './storageAccount.bicep' = {
  name: 'storageDeploy'
  params: {
    storagePrefix: nameSuffix
    location: location
  }
}

output storageEndpoint object = stgModule.outputs.storageEndpoint
```

- Deployment of same resources can happen this way also; with the above script, once can deploy multiple storage accounts for different or same location - with only name changed with suffixes

Examples

- ACR – Azure Container Registry

```
@minLength(5)
@maxLength(50)
param acrName string
param acrAdminUserEnabled bool = true
param location string = resourceGroup().location

@allowed([
  'Basic'
  'Standard'
  'Premium'
])
param acrSku string = 'Standard'

var loginServer = acrName_resource.id

resource acrName_resource
'Microsoft.ContainerRegistry/registries@2020-11-01-preview' = {
  name: acrName
  location: location
  tags: {
    displayName: 'Container Registry'
    'container.registry': acrName
  }
  sku: {
    name: acrSku
  }
  properties: {
    adminUserEnabled: acrAdminUserEnabled
  }
}

output acrLoginServer string = reference(loginServer, '2020-11-01-
preview').loginServer
```

- Azure Storage

```
param storageAccountName string
param blobContainers array
param queues array
param location string = resourceGroup().location

resource storageAccountName_resource
'Microsoft.Storage/storageAccounts@2019-06-01' = {
  name: storageAccountName
  location: location
  kind: 'StorageV2'
  sku: {
    name: 'Standard_LRS'
  }
  properties: {
    accessTier: 'Cool'
  }
}

resource storageAccountName_default_blobContainers
'Microsoft.Storage/storageAccounts/blobServices/containers@2021-02-
01' = [for blob in blobContainers: {
  name: '${storageAccountName}/default/${blob}'
  dependsOn: [
    storageAccountName_resource
  ]
}]

resource storageAccountName_default_queueContainers
'Microsoft.Storage/storageAccounts/queueServices/queues@2021-02-01'
= [for queue in queues: {
  name: '${storageAccountName}/default/${queue}'
  dependsOn: [
    storageAccountName_resource
  ]
}]
```

- **Virtual Network**

```
param vnetName string
param vnetPrefix string
param subnetName string
param subnetPrefix string
param appgwSubnetName string
param appgwSubnetPrefix string
param location string = resourceGroup().location

resource vnetName_resource 'Microsoft.Network/virtualNetworks@2018-
10-01' = {
  name: vnetName
  location: location
  properties: {
    addressSpace: {
      addressPrefixes: [
        vnetPrefix
      ]
    }
  }
}

resource vnetName_subnetName
'Microsoft.Network/virtualNetworks/subnets@2018-10-01' = {
  parent: vnetName_resource
  name: subnetName
  properties: {
    addressPrefix: subnetPrefix
  }
}

resource vnetName_appgwSubnetName
'Microsoft.Network/virtualNetworks/subnets@2018-10-01' = {
  parent: vnetName_resource
  name: appgwSubnetName
  properties: {
    addressPrefix: appgwSubnetPrefix
  }
  dependsOn: [
    vnetName_subnetName
  ]
}

output vnetId string = vnetName_resource.id
output armSubnetId string = vnetName_subnetName.id
output apgwSubnetId string = vnetName_appgwSubnetName.id
```

- **KeyVault**

```
param keyVaultName string
param location string = resourceGroup().location

@allowed([
  true
  false
])
param enabledForDeployment bool = false

@allowed([
  true
  false
])
param enabledForDiskEncryption bool = false

@allowed([
  true
  false
])
param enabledForTemplateDeployment bool = true
param tenantId string = subscription().tenantId
param objectId string
param keysPermissions array = [
  'get'
  'list'
  'create'
  'delete'
  'update'
]
param secretsPermissions array = [
  'get'
  'list'
  'set'
  'delete'
]
param certificatesPermissions array = [
  'get'
  'list'
  'create'
  'delete'
  'update'
]

@allowed([
  'standard'
  'premium'
])
```

```bicep
param skuName string = 'standard'

resource keyVaultName_resource 'Microsoft.KeyVault/vaults@2021-04-01-preview' = {
  name: keyVaultName
  location: location
  properties: {
    enabledForDeployment: enabledForDeployment
    enabledForDiskEncryption: enabledForDiskEncryption
    enabledForTemplateDeployment: enabledForTemplateDeployment
    tenantId: tenantId
    accessPolicies: [
      {
        objectId: objectId
        tenantId: tenantId
        permissions: {
          keys: keysPermissions
          secrets: secretsPermissions
          certificates: certificatesPermissions
        }
      }
    ]
    sku: {
      name: skuName
      family: 'A'
    }
    networkAcls: {
      defaultAction: 'Allow'
      bypass: 'AzureServices'
    }
  }
}
```

- **Application Gateway**

```
param applicationGatewayName string = guid(resourceGroup().id)
param vnetName string = ''
param subnetName string = ''

@allowed([
  'Standard'
  'Standard_v2'
  'WAF'
  'WAF_v2'
])
param tierSkuName string = 'WAF_v2'

@allowed([
  'Standard_Small'
  'Standard_Medium'
  'Standard_Large'
  'Standard_v2'
  'WAF_Large'
  'WAF_Medium'
  'WAF_v2'
])
param sizeSkuName string = 'WAF_v2'

param minCapacity int = 2
param frontendPort int = 443

@allowed([
  'Https'
])
param frontendProtocol string = 'Https'
param backendPort int = 443

@allowed([
  'Http'
  'Https'
])
param backendProtocol string = 'Https'
param healthProbeHostName string = 'test.domain.com'
param healthProbePath string = '/'
param backendIpAddress string = ''

@allowed([
  'Enabled'
  'Disabled'
])
param cookieBasedAffinity string = 'Disabled'
param location string = resourceGroup().location
param httpsListenerNames array = []
param listenerHostName string = '.domain.com'
param backendPoolHostName string = '.internal.testdomain.com'

@secure()
param rootCertData string

@secure()
param certData string

@secure()
param certPassword string

var appGwId = resourceId('Microsoft.Network/applicationGateways',
'${applicationGatewayName}')
var appGwIPConfigName = '${applicationGatewayName}-ipc'
var appGwPublicIpName_var = '${applicationGatewayName}-pip'
var appGwFrontendIPConfigName = '${applicationGatewayName}-fre-ipc'
var appGwFrontendPortName = '${applicationGatewayName}-fre-port'
var appGwBackendPoolName = '${applicationGatewayName}-bkend-pool'
var appGwHttpsListenerName = '${applicationGatewayName}-https-
listener'
var appGwHttpsListenerHostName = listenerHostName
var appGwSSLCertName = '${applicationGatewayName}-ssl-cert'
var appGwSSLCertId = {
  id: '${appGwId}/sslCertificates/${appGwSSLCertName}'
}
var appGwBackendHttpSettingsName = '${applicationGatewayName}-bkend-
http-settings'
var appGwBackendHttpSettingsHostName = backendPoolHostName
var appGwHttpsRuleName = '${applicationGatewayName}-rule'
var appGwProbeName = '${applicationGatewayName}-health-probe'
var subnetRef =
resourceId('Microsoft.Network/virtualNetworks/subnets', vnetName,
subnetName)
var appGwPublicIPRef = appGwPublicIpName.id
var appGwProbeRef = '${appGwId}/probes/${appGwProbeName}'
var appGwSize = sizeSkuName
var appGwTier = tierSkuName
var appGwTrustedRootCertName = '${applicationGatewayName}-root-cert'

resource appGwPublicIpName
'Microsoft.Network/publicIPAddresses@2020-05-01' = {
  name: appGwPublicIpName_var
  location: location
  sku: {
    name: 'Standard'
  }
  properties: {
    publicIPAllocationMethod: 'Static'
  }
}

resource applicationGatewayName_resource
'Microsoft.Network/applicationGateways@2020-05-01' = {
  name: applicationGatewayName
  location: location
  properties: {
    backendHttpSettingsCollection: [for item in httpsListenerNames:
{
      name: '${item}-${appGwBackendHttpSettingsName}'
      properties: {
        port: backendPort
        protocol: backendProtocol
        cookieBasedAffinity: cookieBasedAffinity
        hostName: '${item}${appGwBackendHttpSettingsHostName}'
        probeEnabled: true
        probe: {
          id: appGwProbeRef
        }
        trustedRootCertificates: [
          {
            id:
'${appGwId}/trustedRootCertificates/${appGwTrustedRootCertName}'
          }
        ]
      }
    }]
    httpListeners: [for item in httpsListenerNames: {
      name: '${item}-${appGwHttpsListenerName}'
      properties: {
        frontendIPConfiguration: {
          id:
'${appGwId}/frontendIPConfigurations/${appGwFrontendIPConfigName}'
        }
        frontendPort: {
          id: '${appGwId}/frontendPorts/${appGwFrontendPortName}'
        }
        protocol: frontendProtocol
        sslCertificate: appGwSSLCertId
        hostName: '${item}${appGwHttpsListenerHostName}'
      }
    }]
    requestRoutingRules: [for item in httpsListenerNames: {
      name: '${item}-${appGwHttpsRuleName}'
      properties: {
        ruleType: 'Basic'
        httpListener: {
          id:
resourceId('Microsoft.Network/applicationGateways/httpListeners',
applicationGatewayName, '${item}-${appGwHttpsListenerName}')
        }
        backendAddressPool: {
          id:
resourceId('Microsoft.Network/applicationGateways/backendAddressPool
s', applicationGatewayName, appGwBackendPoolName)
        }
        backendHttpSettings: {
          id:
resourceId('Microsoft.Network/applicationGateways/backendHttpSetting
sCollection', applicationGatewayName, '${item}-
${appGwBackendHttpSettingsName}')
        }
      }
    }]
    sku: {
      name: appGwSize
      tier: appGwTier
    }
    autoscaleConfiguration: {
      minCapacity: minCapacity
    }
    trustedRootCertificates: [
      {
        name: appGwTrustedRootCertName
        properties: {
          data: rootCertData
        }
      }
    ]
    sslCertificates: [
      {
        name: appGwSSLCertName
        properties: {
          data: certData
          password: certPassword
        }
      }
    ]
    gatewayIPConfigurations: [
      {
        name: appGwIPConfigName
        properties: {
          subnet: {
            id: subnetRef
          }
        }
      }
    ]
    frontendIPConfigurations: [
      {
        name: appGwFrontendIPConfigName
        properties: {
          publicIPAddress: {
            id: appGwPublicIPRef
          }
        }
      }
    ]
    frontendPorts: [
      {
        name: appGwFrontendPortName
        properties: {
          port: frontendPort
        }
      }
```

```
      ]
      probes: [
        {
          name: appGwProbeName
          properties: {
            protocol: backendProtocol
            path: healthProbePath
            interval: 30
            timeout: 30
            unhealthyThreshold: 3
            pickHostNameFromBackendHttpSettings: false
            host: healthProbeHostName
            port: backendPort
          }
        }
      ]
      backendAddressPools: [
        {
          name: appGwBackendPoolName
          properties: {
            backendAddresses: [
              {
                ipAddress: backendIpAddress
              }
            ]
          }
        }
      ]
    }
    dependsOn: [
      appGwPublicIpName
    ]
  }
```

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-
01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
      "rootCertData": {
        "reference": {
            "keyVault": {
                "id": "<keyVault_id>"
              },
          "secretName": "<secret_Name>"
      }
    },
      "certData": {
        "reference": {
            "keyVault": {
                "id": "<keyVault_id>"
              },
          "secretName": "<secret_Name>"
      }
    },
    "certPassword": {
        "reference": {
            "keyVault": {
                "id": "<keyVault_id>"
              },
          "secretName": "<secret_Name>"
      }
    }
  }
}
```

Decompilation

- Users having familiarised with ARM templates can easily transition to Bicep templates using Decompilation feature of Azure CLI for Bicep
  - **Bicep** extension for Azure CLI need t be installed/upgarded

    ```
    az bicep install
    ```

    ```
    az bicep upgrade
    ```

  - **Decompile** from ARM template to *Bicep*

    ```
    az bicep decompile -f <file_name>
    ```

- **Deploy** Templates

  ```
  az deployment group create -f ./<file_name>.bicep -g
  <resource_group_name> \
  --parameters <key1>=<value1> <key2>=<value2> <key3>=<value3>
  ```

Which One to Use

- **Bicep** or **ARM**
  - **Bicep** transpiles to **ARM** only
  - Target Users are mostly Infra or Operations team; *so comfort factor should be the primary decisive factor*
  - For **Brownfield** scenarios, If comfortable with ARM then *no need for change*
    - If there is a need to move to 💪 then there is a **Decompilation** option to move to **ARM**
  - For **Greenfield** scenarios, preferred way is to *go for **Bicep***
  - For both **Brownfield** and **Greenfield**, if there is any one or more of the follwoing concerns with ARM Templates, then advise is to *go with **Bicep***
    - *ARM is NOT manageable; and not extensible*
    - *Hard to program Repetitive execution and Conditions*

        - *No or Very little prorgrammitic control*
- **Bicep** or **Terraform**
    - **Bicep** is NOT intended to be a replacement for **Terraform**
    - If already using **Terraform**. *then no need to change*
    - If Multi-Cloud is the choice and/or reason for opting **Terraform**; *then no need to change*
        - Bicep ios only for Azure with some extension points for non-Azure services to be deployed on Azure
    - Azure has excellent integration option for Terrraform as well

References

- Learn Bicep
- Source Repo