Bicep

Infrastructure as Code in Azure

Daniel Scott-Raynsford

Partner Technology Strategist | Microsoft

@dscottraynsford

Learn more: aka.ms/Bicep

Bicep is…

…a **transparent abstraction** over ARM and ARM templates and aims to drastically **simplify the authoring experience** with a **cleaner syntax** and better support for **modularity** and **code re-use.**

Learn more: aka.ms/Bicep

Goals of Bicep

Simple declarative language to provision infrastructure on Azure

Transpiles to ARM Templates: Leverage ARM template knowledge and investments.

Best Possible Language: It is not JSON. Not YAML. Or HCL (it does look similar).

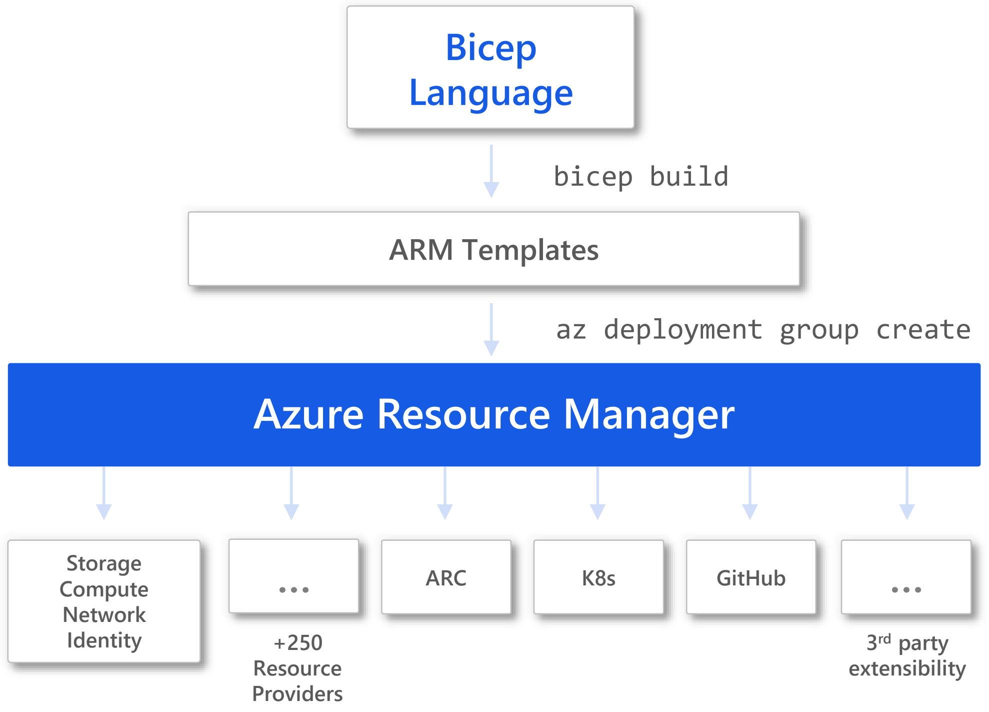
Transparent Abstraction: It should just work with all Azure resources.

Easy to Understand: Simple to learn, simple to read.

Platform Integrations: Integrated with Azure Policy, Blueprints, TemplateSpecs, Service Catalog etc.

Modular: easy to build modular and reusable code.

Validation & Type checking: Enable high degree of confidence in the syntactic correctness of the language before deploying.



Learn more: aka.ms/Bicep

Demo

A look at the Bicep syntax

Flexible Ordering of Type Declarations

No more quotes on property names

Simple string interpolation

https://github.com/Azure/bicep/blob/master/docs/spec/bicep.md

```
resource dataAccount 'Microsoft.Storage/storageAccounts@2017-06-01' = {
  name: '${storageAccountName}-data'
  location: resourceGroup().location
  sku: {
    name: storageAccountType
  }
  kind: 'Storage'
}

var imagesAccountSuffix = 'img' // Variable declared after resource

resource imagesAccount 'Microsoft.Storage/storageAccounts@2017-06-01' = {
  name: '${storageAccountName}-${imagesAccountSuffix}'
  location: resourceGroup().location
  sku: {
    name: storageAccountType
  }
  kind: 'Storage'
}
```

No more […] expressions syntax

No more parameters()

No more variables()

https://github.com/Azure/bicep/blob/master/docs/spec/bicep.md

```
param storageAccountName string
param storageAccountType string

var location = 'westus'

resource diagsAccount 'Microsoft.Storage/storageAccounts@2017-06-01' = {
  name: storageAccountName
  location: location
  sku: {
    name: storageAccountType
  }
  kind: 'Storage'
}
```

Simpler resource declaration

Direct property access of a resource

Automatic dependency management in certain scenarios

https://github.com/Azure/bicep/blob/master/docs/spec/bicep.md

```
param clusterName string
param dnsPrefix string

resource aks 'Microsoft.ContainerService/managedClusters@2020-03-01' = {
  name: clusterName
  location: resourceGroup().location
  properties: {
    dnsPrefix: dnsPrefix
    // More properties here
  }
}

output controlPlaneFQDN string = aks.properties.fqdn
```

Ternary Expressions

Better Comments

```
param replicateGlobally bool // Single-line Comment

/*
Block comment
*/
resource myStorageAccount 'Microsoft.Storage/storageAccounts@2017-10-01' = {
  // Single-line Comment
  name: storageAccountName
  location: resourceGroup().location
  properties: {
    supportsHttpsTrafficOnly: true
    accessTier: 'Hot'
  }
  kind: StorageV2
  sku: {
    name: replicateGlobally ? 'Standard_GRS' : 'Standard_LRS'
  }
}
```

https://github.com/Azure/bicep/blob/master/docs/spec/expressions.md

For loops

No more nasty Copy syntax

```
// array of storage account names
param storageAccounts array

resource storageAccountResources 'Microsoft.Storage/storageAccounts@2019-06-
01' = [for storageName in storageAccounts: {
  name: storageName
  location: resourceGroup().location
  properties: {
    supportsHttpsTrafficOnly: true
    accessTier: 'Hot'
    encryption: {
      keySource: 'Microsoft.Storage'
      services: {
        blob: {
          enabled: true
        }
        file: {
          enabled: true
        }
      }
    }
  }
  kind: 'StorageV2'
  sku: {
    name: 'Standard_LRS'
  }
}]
```

https://github.com/Azure/bicep/blob/master/docs/spec/loops.md

Modules

```
// Input parameters must be specified by the module consumer
param publicIpResourceName string
param publicIpDnsLabel string = '${publicIpResourceName}-${newGuid()}'
param location string = resourceGroup().location
param dynamicAllocation bool

resource publicIp 'Microsoft.Network/publicIPAddresses@2020-06-01' = {
  name: publicIpResourceName
  location: location
  properties: {
    publicIPAllocationMethod: dynamicAllocation ? 'Dynamic' : 'Static'
    dnsSettings: {
      domainNameLabel: publicIpDnsLabel
    }
  }
}

// Set an output which can be accessed by the module consumer
output ipFqdn string = publicIp.properties.dnsSettings.fqdn
```

https://github.com/Azure/bicep/blob/master/docs/spec/loops.md

What you will need - tooling

Genaral

1. Bicep CLI – the *transpiler*
2. Bicep VSCode Extension – *a must have for authoring!*
3. Bicep Playground
4. Bicep in GitHub Codespaces
5. Az CLI - *Bicep fully integrated as of v2.2.0+*
   - It will download the Bicep CLI automatically.
   - Or manually with: `az bicep install`
6. Azure PowerShell – Bicep fully integrated as of v5.6.0+
   - *Does not download Bicep CLI automatically.*
   - 3rd Party OSS PowerShell module: `Install-Module -Name Bicep`

CI/CD toolchain

1. GitHub Actions:
   - Bicep GitHub Action: Bicep Build · Actions · GitHub Marketplace
2. Azure DevOps Pipelines:
   - Third party Bicep extension: Bicep Tasks - Visual Studio Marketplace
   - *Can use AzPowerShell or AzCLI task; install Bicep first.*
3. Others… use scripts with AZ CLI or Azure PowerShell.

FAQ

Some common questions:

Q: Is Bicep ready for production use?

A: Yes, as of 0.3.

Q: Can I export templates from the Azure Portal in Bicep syntax?

A: Not yet, but it is planned.

Q: Should I convert all my ARM templates to Bicep?

A: Probably, but It depends.

Q: Something in Bicep doesn't work – what should I do?

A: Report it in GitHub repo as an issue.

Q: Should I use this instead of Terraform?

A: Both are great tools. There are pros and cons with each.

Q: Is Bicep a declarative or imperative syntax?

A: Yes. Bicep is a declarative syntax.

Q: What is in Bicep v0.4 and when is that?

A: The Bicep Linter! And soon… ☺

Q: Are more MS Learn modules coming soon?

A: Yes, parameters, conditions & loops, Scopes, What-if & Linter, team collaboration + more…

Resources

The following are some resources to help you get started with Bicep:

- Where to start: https://aka.ms/Bicep
- Installing Bicep: https://github.com/Azure/bicep/blob/main/docs/installing.md
- Bicep Playground (interactive Bicep compiler): https://aka.ms/bicepdemo
- Bicep Tutorial: https://github.com/Azure/bicep/blob/master/docs/tutorial/01-simple-template.md
- GitHub Action for Bicep Build: https://github.com/marketplace/actions/bicep-build
- Bicep Learn Modules:
  - Introduction to infrastructure as code using Bicep - Learn | Microsoft Docs
  - Deploy Azure resources by using Bicep templates - Learn | Microsoft Docs
  - Deploy child and extension resources by using Bicep - Learn | Microsoft Docs
  - Extend ARM templates by using deployment scripts - Learn | Microsoft Docs
  - More coming soon…

Takeaways

Bicep Syntax is Clearer

Bicep Syntax is far easier to learn, understand and maintain than ARM.

Transpiles to ARM – does not replace it

Bicep is an **abstraction layer** on top of ARM.

Users are not forced to adopt Bicep syntax but it will become the de-facto standard over time.

Can be used in production?

Yes.