
📘 Azure Bicep – The Modern IaC Language for Azure

🔹 1. What is Bicep?

- **Bicep = Domain-Specific Language (DSL)** for deploying Azure resources.
- It's a **replacement for writing raw ARM JSON templates**.
- Bicep files **compile into ARM templates** behind the scenes.
- File extension: .bicep.

👉 Think of Bicep as **"ARM templates, but human-friendly."**

🔹 2. Why Bicep?

Problems with ARM JSON:

- ❌ Too verbose and complex.
- ❌ Hard to read, modify, and debug.
- ❌ Nested objects and functions are messy.

Benefits of Bicep:

- ✅ **Simpler syntax** (like writing code, not JSON).

- ✅ **Modular** → break into smaller files.
- ✅ **Idempotent** → safe to redeploy.
- ✅ **Type-safe & IntelliSense** in VS Code.
- ✅ **Automatic dependency management**.
- ✅ **No need to learn JSON** → easier for DevOps engineers.

🔹 3. How Bicep Works

1. You write a **Bicep file (.bicep)**.
2. Azure CLI/PowerShell **compiles it into ARM JSON**.
3. Azure Resource Manager (ARM) **deploys resources** from it.

👉 Bicep is **just a friendlier layer over ARM**.

🔹 4. Installing Bicep

Via Azure CLI

```
az bicep install
az bicep upgrade
```

Verify installation

```
az bicep version
```

🔹 5. Deployment Methods

Deploy using **Azure CLI**:

```
az deployment group create \
  --resource-group MyRG \
  --template-file main.bicep \
  --parameters storageName=mystorage123
```

Or using **PowerShell**:

```
New-AzResourceGroupDeployment `
  -ResourceGroupName MyRG `
  -TemplateFile main.bicep `
  -storageName mystorage123
```

🔹 6. Bicep Syntax (Basic Example)

Storage Account in ARM JSON:

```
{
  "type": "Microsoft.Storage/storageAccounts",
  "apiVersion": "2023-01-01",
  "name": "mystorage123",
  "location": "eastus",
  "sku": { "name": "Standard_LRS" },
  "kind": "StorageV2",
  "properties": {}
}
```

Same thing in Bicep:

```
resource storage 'Microsoft.Storage/storageAccounts@2023-01-01' = {
  name: 'mystorage123'
  location: 'eastus'
  sku: {
    name: 'Standard_LRS'
  }
  kind: 'StorageV2'
}
```

👉 Much cleaner!

🔹 **7. Bicep File Structure**

- **Parameters** → external inputs.
- **Variables** → reusable values.
- **Resources** → actual Azure resources.
- **Outputs** → values returned post-deployment.

Example:

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

resource storage 'Microsoft.Storage/storageAccounts@2023-01-01' = {
  name: storageName
  location: location
  sku: {
    name: 'Standard_LRS'
  }
```

```
  kind: 'StorageV2'
}

output blobEndpoint string = storage.properties.primaryEndpoints.blob
```

🔹 8. Parameters in Bicep

Parameters allow dynamic values:

```
param adminUsername string
param adminPassword string {
  secure: true
}
param vmSize string = 'Standard_B1s'
```

🔹 9. Variables in Bicep

Variables help reuse values:

```
var storageSku = 'Standard_LRS'
var uniqueName = 'storage${uniqueString(resourceGroup().id)}'

resource storage 'Microsoft.Storage/storageAccounts@2023-01-01' = {
  name: uniqueName
  location: resourceGroup().location
  sku: {
    name: storageSku
  }
  kind: 'StorageV2'
}
```

🔹 10. Outputs

Return info after deployment:

```
output storageId string = storage.id
output blobUrl string = storage.properties.primaryEndpoints.blob
```

🔹 11. Modules in Bicep

You can break large templates into **modules** for reusability.

Example – vm.bicep (module):

```
param vmName string
param adminUsername string
param adminPassword string

resource vm 'Microsoft.Compute/virtualMachines@2023-03-01' = {
  name: vmName
  location: resourceGroup().location
  properties: {
    hardwareProfile: {
      vmSize: 'Standard_B1s'
    }
    osProfile: {
      computerName: vmName
      adminUsername: adminUsername
      adminPassword: adminPassword
    }
  }
}
```

Main file (main.bicep):

```
module myVM './vm.bicep' = {
  name: 'myVMmodule'
  params: {
    vmName: 'myVM'
    adminUsername: 'azureuser'
    adminPassword: 'P@ssw0rd123!'
  }
}
```

👉 Encourages **modularity & code reuse**.

🔹 12. Loops in Bicep

You can deploy multiple resources with loops:

```
param locations array = [ 'eastus', 'westus', 'centralus' ]

resource storages 'Microsoft.Storage/storageAccounts@2023-01-01' = [for location in locations: {
  name: 'storage${uniqueString(location)}'
  location: location
  sku: {
    name: 'Standard_LRS'
  }
  kind: 'StorageV2'
}]
```

🔹 13. Bicep vs ARM Templates

Feature	ARM Template (JSON)	Bicep (DSL)

Format	JSON	Simplified DSL
Readability	Hard	Easy
File Size	Large	Compact
Reusability	Limited	Modules
IntelliSense	Limited	Full support
Debugging	Manual	Easy

👉 **Bicep = Easier way to write ARM** (Azure itself still uses ARM behind the scenes).

🔹 **14. When to Use Bicep?**

✅ Best for:

- Deploying **repeatable environments**.
- **CI/CD pipelines** (Azure DevOps, GitHub Actions).
- **Multi-resource deployments** (VM + Network + Storage).
- Teams that want **IaC without Terraform**.

📝 **Summary Notes**

- **Bicep = DSL for Azure IaC** → Compiles to ARM JSON.
- **Advantages**: Cleaner, easier, modular, loops, IntelliSense.
- **Sections**: Parameters, Variables, Resources, Outputs, Modules.
- **Deployment**: Via CLI, PowerShell, or CI/CD pipelines.
- **Real-world use**: Automate infra provisioning (VMs, Storage, AKS, Networking).
