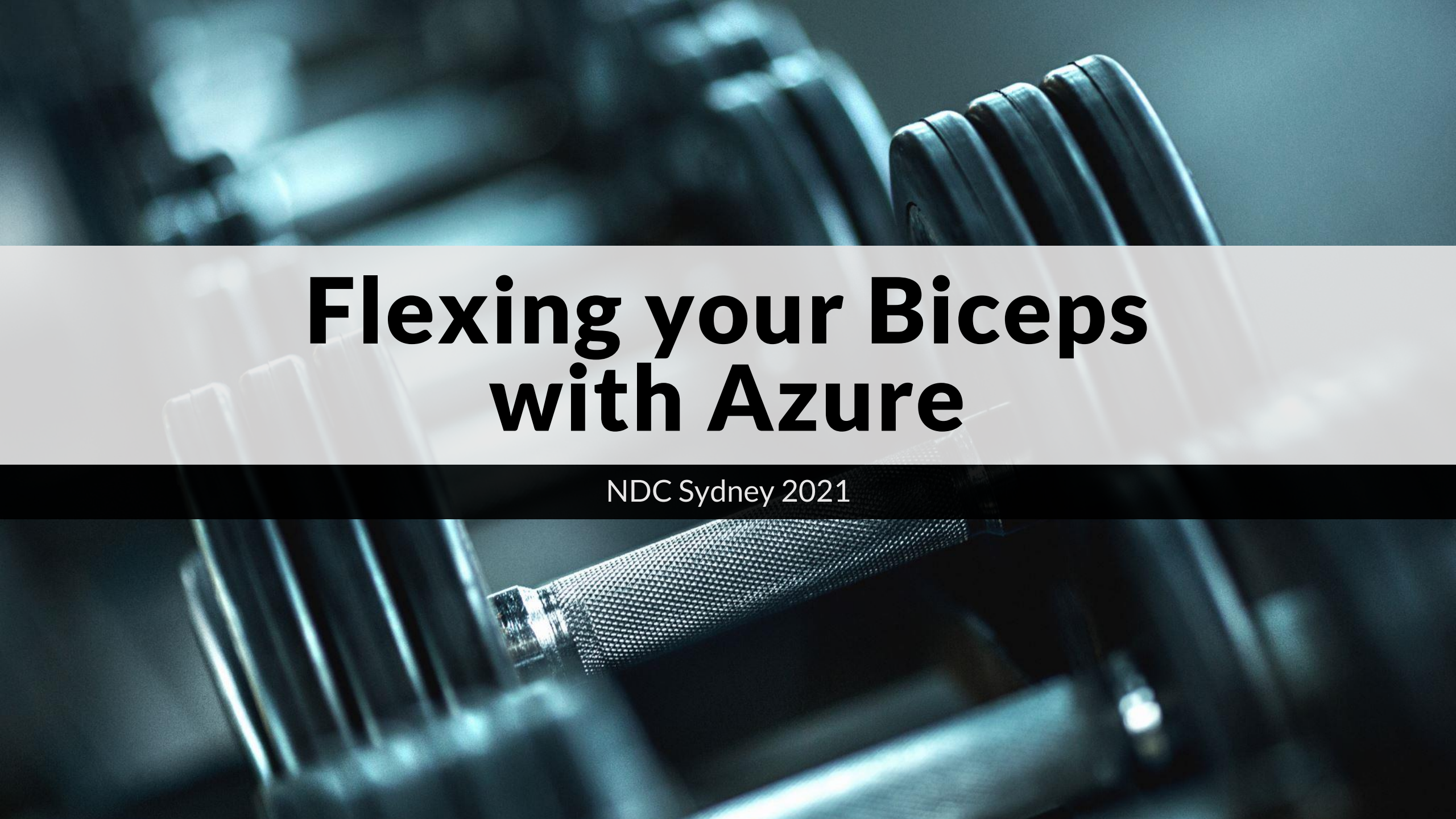


Flexing your Biceps with Azure

NDC Sydney 2021

Infrastructure as Code

- Automate resource provisioning
- Triggered by success of CI/CD processes
- Helps to manage cost, risk, and speed
- Cloud infrastructure is getting complicated…
- JSON, YAML, XML everywhere…
- We need to keep it sweet and short! (KISS principle)

William Liebenberg
Azenix Lead Software Engineer

@william_dotnet

/william-liebenberg

Melb.NET

/william-liebenberg

azuregems.io

William Liebenberg
Azenix Lead Software Engineer

@william_dotnet

/william-liebenberg

Melb.NET

/william-liebenberg

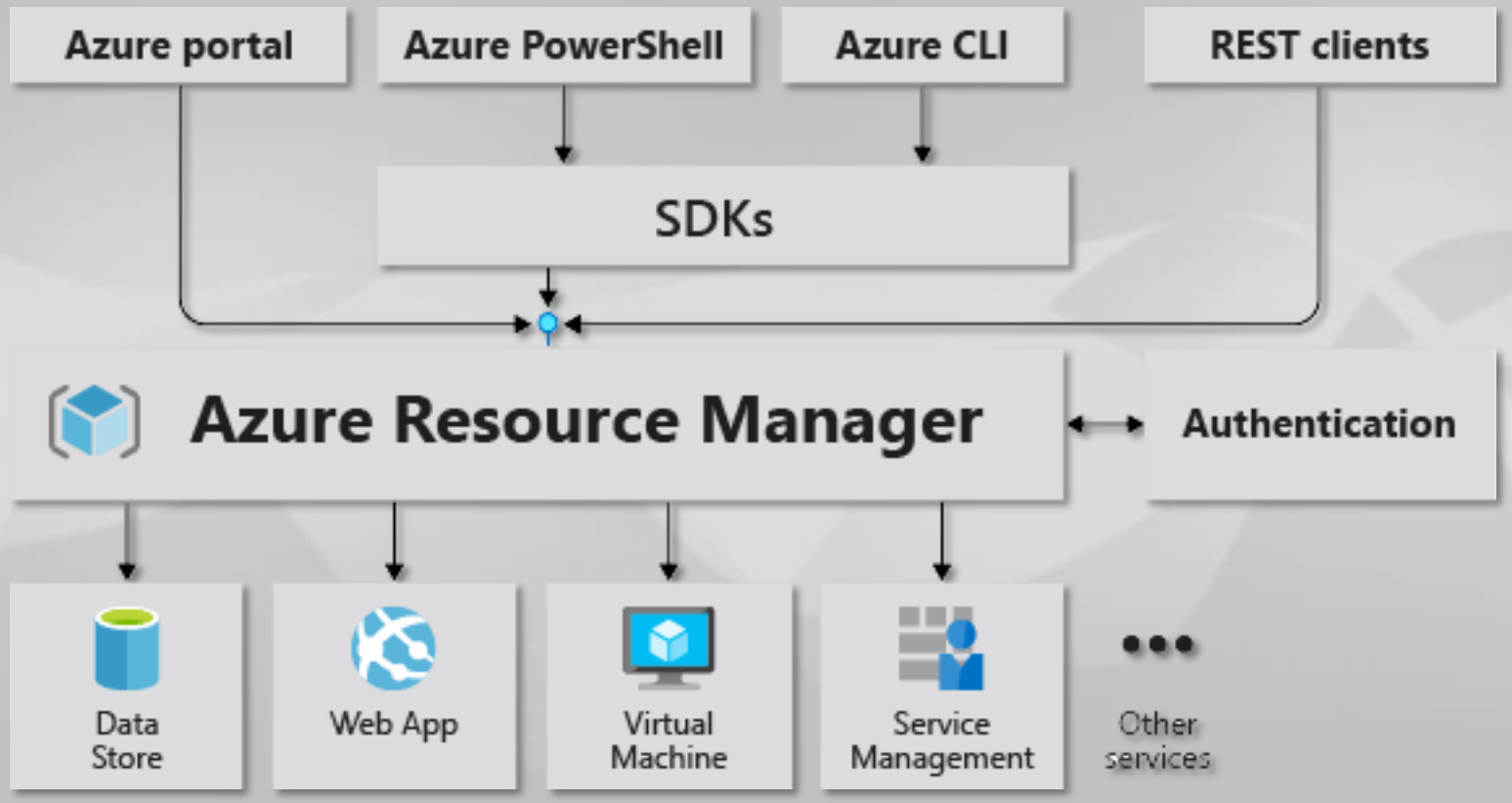
azuregems.io

Follow me: @william_dotnet | azuregems.io | @AzenixAU | #NDCSydney

Agenda

- What is Azure Bicep?
- Comparisons
- How to - Bicep
- Bicep Registries
- Demos

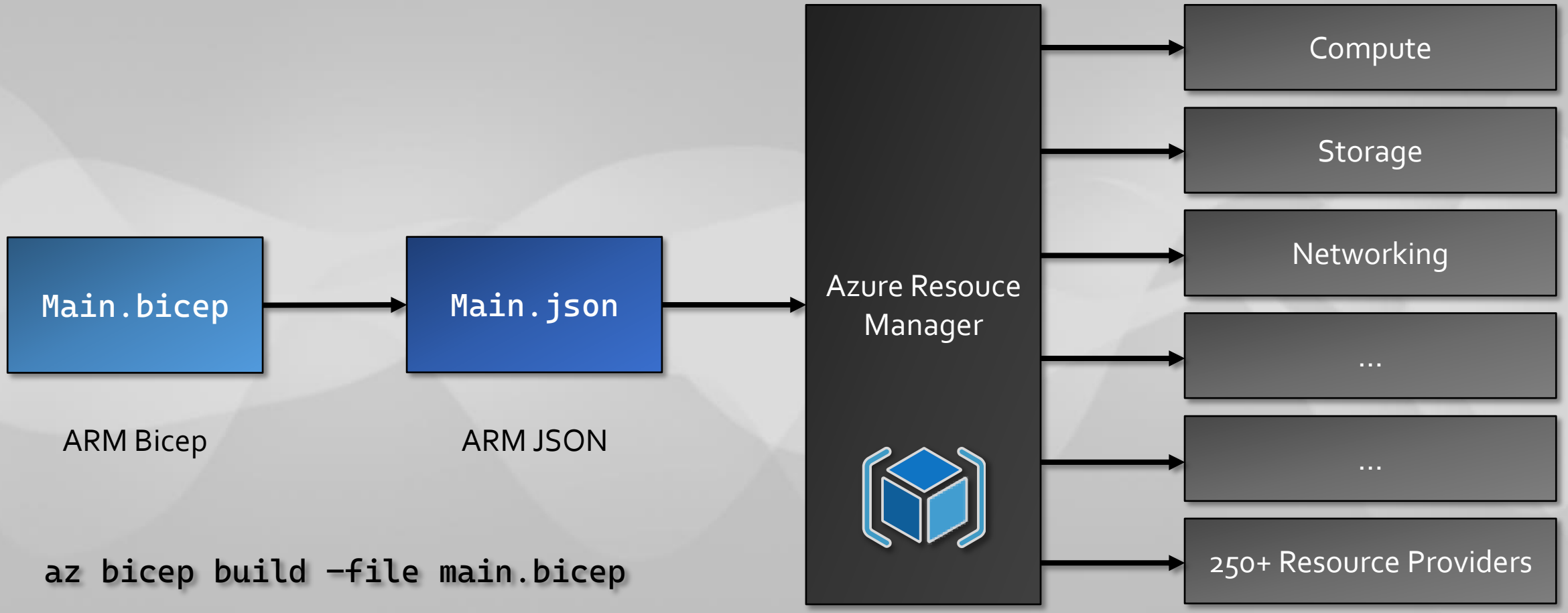
Azure Resource Manager



Source: Microsoft Docs

What is Azure Bicep?

What is Azure Bicep?




```
az bicep build –file main.bicep
```

What is Azure Bicep?

2 types of ARM templates:

- JSON
- Bicep

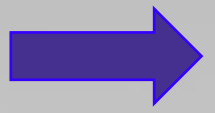
Both translate to SDK/REST calls to Azure Resource Manager (ARM) to provision resources

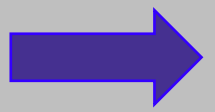
What is Azure Bicep?

Domain Specific Language (DSL) that aims to:

- simplify authoring of templates (compared to JSON)
- increase modularity
- increase code reuse
- transpile to ARM JSON equivalent

What is Azure Bicep?



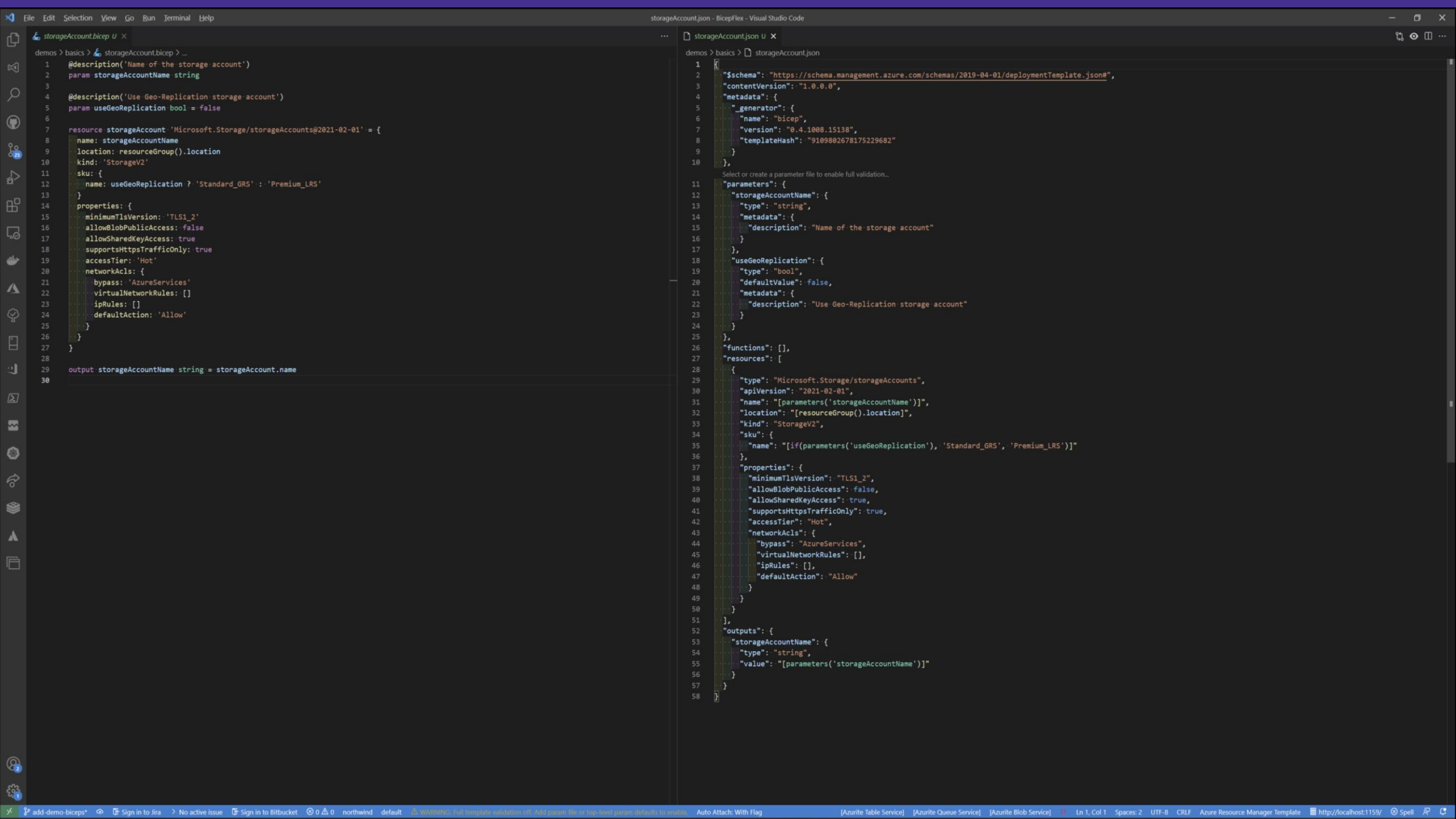


```
@description('Name of the storage account')
param storageAccountName string

@description('Use Geo-Replication storage account')
param useGeoReplication bool = false

resource storageAccount 'Microsoft.Storage/storageAccounts@2021-02-01' = {
  name: storageAccountName
  location: resourceGroup().location
  kind: 'StorageV2'
  sku: {
    name: useGeoReplication ? 'Standard_GRS' : 'Premium_LRS'
  }
  properties: {
    minimumTlsVersion: 'TLS1_2'
    allowBlobPublicAccess: false
    allowSharedKeyAccess: true
    supportsHttpsTrafficOnly: true
    accessTier: 'Hot'
    networkAcls: {
      bypass: 'AzureServices'
      virtualNetworkRules: []
      ipRules: []
      defaultAction: 'Allow'
    }
  }
}

output storageAccountName string = storageAccount.name
```


File Edit Selection View Go Run Terminal Help
storageAccount.json - BicepFlex - Visual Studio Code
storageAccount.bicep U
demos > basics > storageAccount.bicep > ...
@description('Name of the storage account')
param storageAccountName string
@description('Use Geo-Replication storage account')
param useGeoReplication bool = false
resource storageAccount 'Microsoft.Storage/storageAccounts@2021-02-01' = {
name: storageAccountName
location: resourceGroup().location
kind: 'StorageV2'
sku: {
name: useGeoReplication ? 'Standard_GRS' : 'Premium_LRS'
}
properties: {
minimumTlsVersion: 'TLS1_2'
allowBlobPublicAccess: false
allowSharedKeyAccess: true
supportsHttpsTrafficOnly: true
accessTier: 'Hot'
networkAcls: {
bypass: 'AzureServices'
virtualNetworkRules: []
ipRules: []
defaultAction: 'Allow'
}
}
}
output storageAccountName string = storageAccount.name
storageAccount.json U
demos > basics > storageAccount.json
{
"$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
"contentVersion": "1.0.0.0",
"metadata": {
"_generator": {
"name": "bicep",
"version": "0.4.1008.15138",
"templateHash": "9109802678175229682"
}
},
Select or create a parameter file to enable full validation...
"parameters": {
"storageAccountName": {
"type": "string",
"metadata": {
"description": "Name of the storage account"
}
},
"useGeoReplication": {
"type": "bool",
"defaultValue": false,
"metadata": {
"description": "Use Geo-Replication storage account"
}
}
},
"functions": [],
"resources": [
{
"type": "Microsoft.Storage/storageAccounts",
"apiVersion": "2021-02-01",
"name": "[parameters('storageAccountName')]",
"location": "[resourceGroup().location]",
"kind": "StorageV2",
"sku": {
"name": "[if(parameters('useGeoReplication'), 'Standard_GRS', 'Premium_LRS')]"
},
"properties": {
"minimumTlsVersion": "TLS1_2",
"allowBlobPublicAccess": false,
"allowSharedKeyAccess": true,
"supportsHttpsTrafficOnly": true,
"accessTier": "Hot",
"networkAcls": {
"bypass": "AzureServices",
"virtualNetworkRules": [],
"ipRules": [],
"defaultAction": "Allow"
}
}
}
],
"outputs": {
"storageAccountName": {
"type": "string",
"value": "[parameters('storageAccountName')]"
}
}
}
add-demo-biceps*
Sign in to Jira
No active issue
Sign in to Bitbucket
0 0
northwind
default
WARNING: Full template validation off. Add param file or top-level param defaults to enable.
Auto Attach: With Flag
[Azurite Table Service]
[Azurite Queue Service]
[Azurite Blob Service]
Ln 1, Col 1
Spaces: 2
UTF-8
CRLF
Azure Resource Manager Template
http://localhost:1159/
Spell

Demo 1 – The Basics

What Azure Bicep is NOT!

Bicep is NOT a Configuration Management Tool

Once resources are provisioned, that's it!

Does not deploy data/software onto resources

Chef, Puppet, SaltStack and Ansible are Configuration Management Tools*

Bicep vs Terraform vs Pulumi

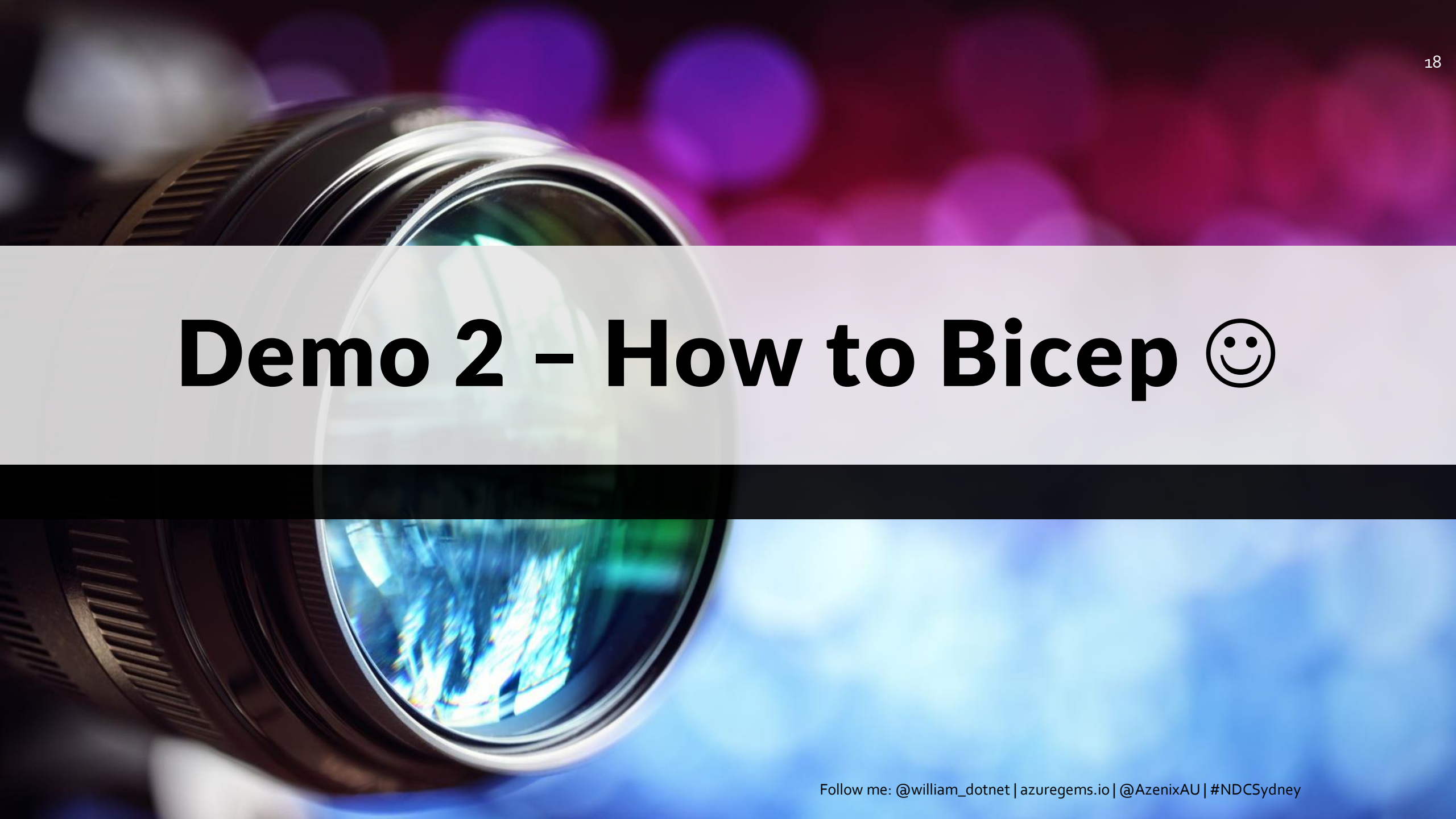
Quick Compare

	Azure Bicep	Terraform	Pulumi
State Management	Live-state	Self Managed + SaaS option	Managed by Pulumi + Self-serve
Style	Strict/Declarative	Strict/Declarative	Flexible/Imperative
Language	Bicep (DSL)	HCL (DSL)	C#, F#, JavaScript, TypeScript, Go, Python
Free	Yes	Yes + Paid offering	Yes + Paid offering
Project Management	Modules	Modules	Monolithic / Micro projects
Multi-cloud	No	Yes	Yes
Support	Microsoft	Hashicorp	Pulumi
Open Source	MIT License	Apache License 2.0	Mozilla PL 2.0
Learning Curve	Low	Med	Med/High

How to use Bicep

- Install the tools
  - Az CLI
  - Or Azure PowerShell
- Azure DevOps Pipelines
- GitHub Workflows

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



Demo 2 – How to Bicep ☺

GitHub Actions – Bicep Lint


```yaml
jobs:
  Lint:
    name: Check best practices (Bicep Linter)
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2
    - shell: pwsh
      run: |
        az bicep build --file .azure/azureDeploy.bicep
```

GitHub Actions – Bicep Validate


```yaml
jobs:
  Validate:
    name: Validate resources
    runs-on: ubuntu-latest
    steps:
    - uses: azure/login@v1
      with:
        creds: ${{ secrets.AZURE_CREDENTIALS }}
    - uses: actions/checkout@v2
    - name: Validate Bicep
      shell: pwsh
      run: |
        az deployment group validate `
          -g ${{ env.AZURE_RESOURCEGROUP_NAME }} `
          --template-file .azure/azureDeploy.bicep `
          --parameters .azure/parameters.json
```

GitHub Actions – Bicep What-If

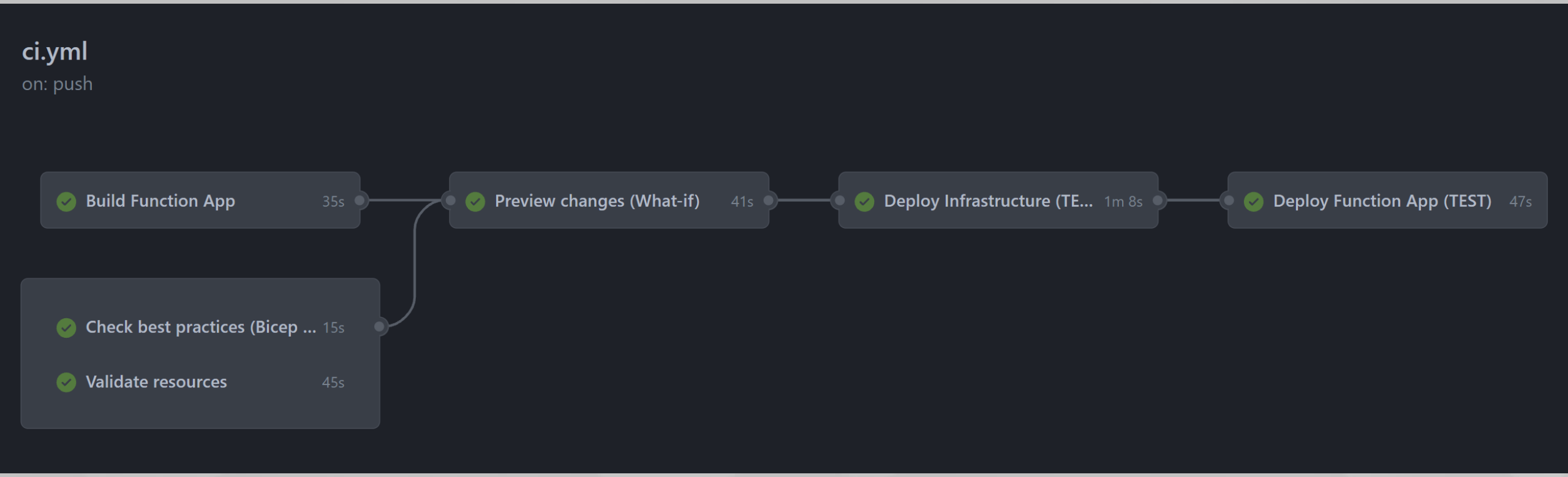

```yaml
jobs:
  PreviewInfra:
    name: Preview changes (What-if)
    runs-on: ubuntu-latest
    needs: [Lint, Validate]
    steps:
    - uses: azure/login@v1
      name: Sign in to Azure
      with:
        creds: ${{ secrets.AZURE_CREDENTIALS }}
    - uses: actions/checkout@v2
    - name: Run what-if
      shell: pwsh
      run: |
        az deployment group what-if `
          -g ${{ env.AZURE_RESOURCEGROUP_NAME }} `
          --template-file .azure/azureDeploy.bicep `
          --parameters .azure/parameters.json
```

GitHub Actions – Bicep Deploy


```yaml
jobs:
  Deploy:
    runs-on: ubuntu-latest
    needs: [Preview]
    outputs:
      funcAppName: ${{ steps.deploy.outputs.functionAppName }}
      storageAccountName: ${{ steps.deploy.outputs.storageAccountName }}
    steps:
     /* ... Checkout and Log into azure ... */
    - name: Deploy to Test resource group
      id: deploy
      shell: pwsh
      run: |
        $outputs = (az deployment group create -g ${{ env.AZURE_RESOURCEGROUP_NAME }} `
          --mode Incremental `
          --template-file .azure/azureDeploy.bicep `
          --parameters .azure/parameters.json --query properties.outputs) | ConvertFrom-Json

        Write-Output "::set-output name=functionAppName::$($outputs.functionAppName.value)"
        Write-Output "::set-output name=storageAccountName::$($outputs.storageAccountName.value)"
```

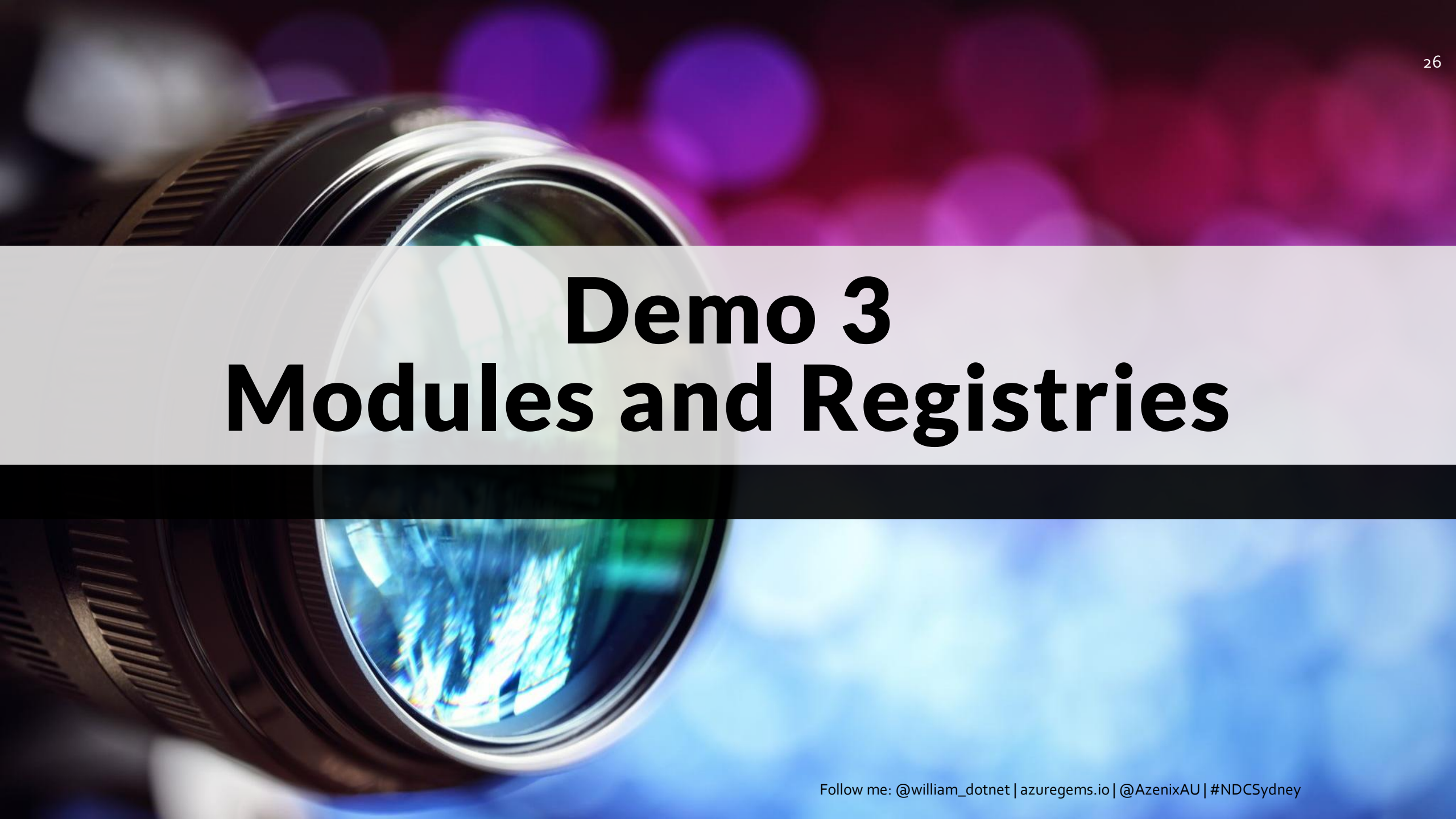
GitHub Actions – Bicep Deploy



Private Registies

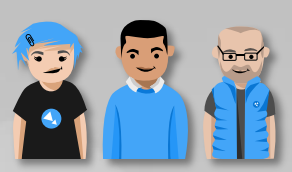
Private Registry

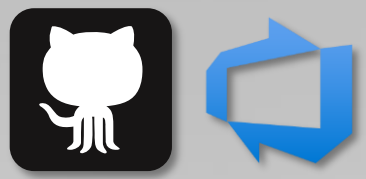
- Reference modules in **Azure Container Registry** (ACR)
- Encapsulate complex architectures
- Tagging (versioning)

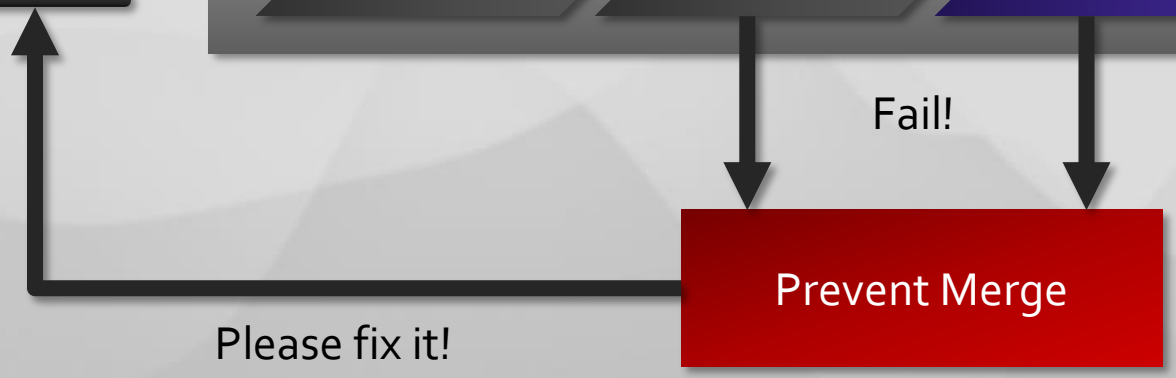


Demo 3
Modules and Registries

DevOps with Bicep Registry








```
bicep publish module.bicep
--target br:example.azurecr.io/hello/world:v1
```

Pros and Cons

Pros and Cons

Pros

- ✔ No managed state
- ✔ Simpler syntax
- ✔ Target Scopes (RG, Sub, Tenant)
- ✔ Modular
- ✔ Validation
- ✔ Registries
- ✔ Day-0 support from Microsoft
- ✔ Decompile ARM JSON into Bicep ☺

Cons

- ✘ Live-state (What-if) can be inaccurate
- ✘ Visualiser can't deal with modules
- ✘ No User Defined Functions (UDFs)
- ✘ No resource provider aliases
- ✘ Parameters file is still JSON!
- ✘ Listing Keys/ConnectionStrings are still ugly
- ✘ Lack of Discovery of modules

Links

Links

- Slides and Code: https://github.com/william-liebenberg/BicepFlex
- Azure Bicep on GitHub
- MS Learn Bicep Module
- Azure Architecture Center
- Azure Quickstart templates repo
- Bicep Playground

Bicep Summary

Summary

- KISS by design!
- Improved programming syntax
- Save time and keystrokes
- Registries rule!
- There is more coming (we're not at v1.0 yet!)
- Writing Infra-as-Code is fun again!

@william_dotnet

/william-liebenberg

/william-liebenberg

azuregems.io

Melb.NET