Introduction to Azure Bicep

CSCI E-94

Fundamentals of Cloud Computing - Azure

Joseph Ficara

Agenda

- Overview
- Getting Started - Tooling
- Creating your first Bicep Template
  - Deploying to Azure
- Bicep Language Overview
- Creating multiple resources
- Best Practices

Overview

- What is Azure Bicep?
  - Declarative language for
    - Describing and deploying Azure resources
  - **D**omain **S**pecific **L**anguage **DSL**
    - Simplifies syntax & structure of
      - **A**zure **R**esource **M**anager **ARM** templates
  - Can be used with existing Azure tools
    - Azure CLI, Azure PowerShell
    - Azure Portal, Azure DevOps

Overview

- Why do you care?
  - Concise readable code compared to ARM
    - Much easier to write & maintain
    - Enables infrastructure as code
      - Repeatability in deployments
      - Verifiable through PRs
      - Reducing human error
  - Supports code reuse & modularization
  - Facilitates
    - Collaboration & Best practices

Overview

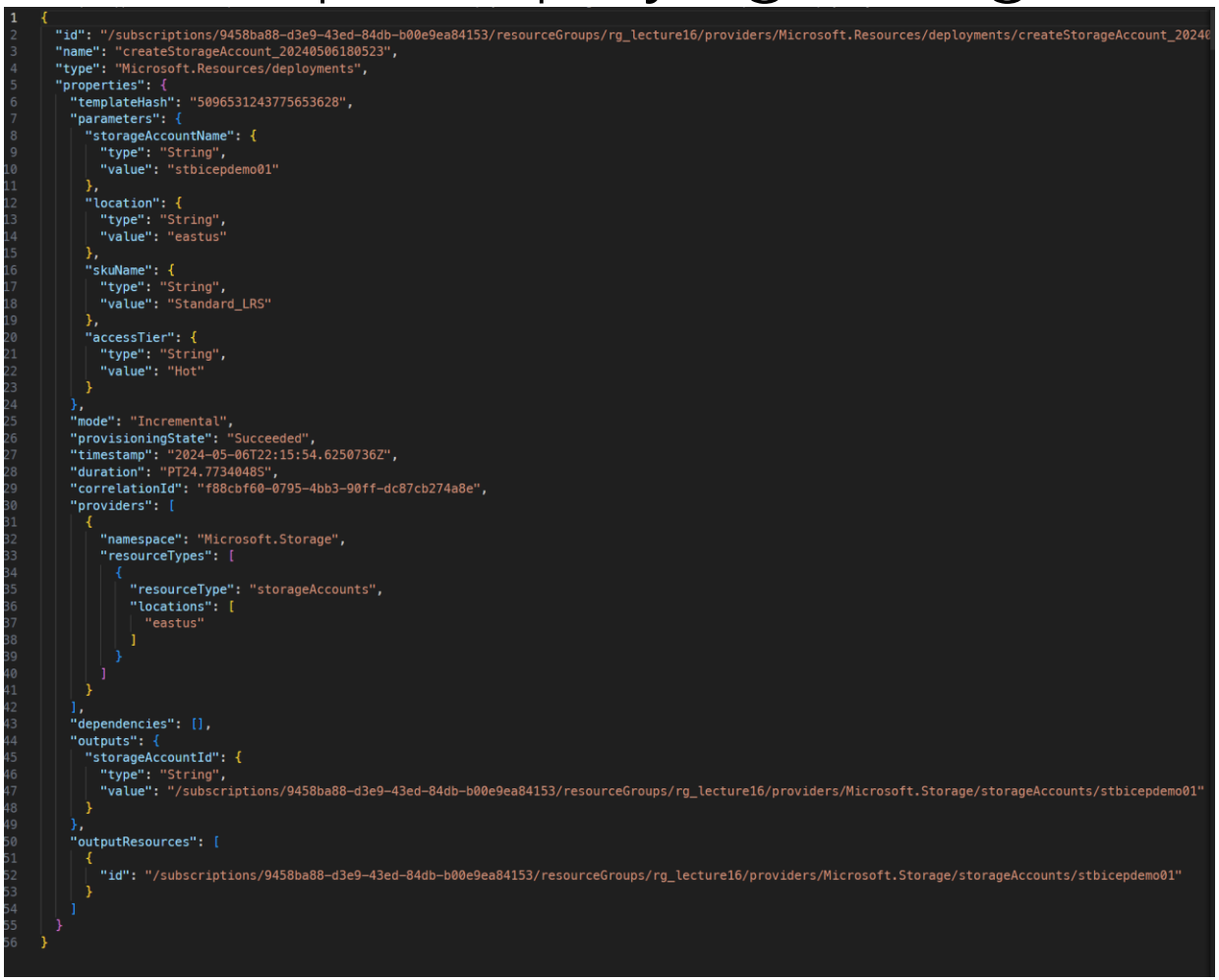
- Bicep example deploying storage account

StorageAccountExample > createStorageAccount.bicep > ...

```bicep
// Example deployment cmdline: (must be all on the same line broken out here for clarity)
// az deployment group create --name $("createStorageAccount_" + (Get-Date -Format "yyyyMMddHHMMss"))
//                            --resource-group rg_lecture16  --template-file createStorageAccount.bicep
//                            --parameters storageAccountName='<storage account name>'
// Parameter to specify the name of the storage account
@description('The name of the storage account to create')
param storageAccountName string

// Parameter to specify the location of the storage account,
// defaulted from resource group's location
param location string = resourceGroup().location

// Parameter to specify the SKU name of the storage account
param skuName string = 'Standard_LRS' // Standard Locally-Redundant Storage

// Parameter to specify the access tier of the storage account
param accessTier string = 'Hot'

// Resource block to create a storage account
resource storageAccount 'Microsoft.Storage/storageAccounts@2021-04-01' = {
  name: storageAccountName
  location: location
  sku: {
    name: skuName
  }
  kind: 'StorageV2'
  properties: {
    accessTier: accessTier
  }
}

// Output to retrieve the ID of the storage account
output storageAccountId string = storageAccount.id

```

Overview

- ARM example deploying storage account



Questions ?

Getting Started - Tooling

- Command line tools
  - **Azure CLI or PowerShell**
    - Azure CLI
      - Windows
        - winget install -e --id Microsoft.AzureCLI
      - Mac
        - Homebrew package manager
        - brew update && brew install azure-cli
      - Linux
        - curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash
    - PowerShell
      - Windows, Mac, Linux
        - Download and install corresponding file from the Get PowerShell section

Getting Started - Tooling
Command line tools

- **Bicep CLI**
  - Is included with the Azure CLI
    - Can be updated via
      - **az bicep install**
  - PowerShell
    - Azure PowerShell
      - **Install-Module -Name Az -AllowClobber -Scope CurrentUser**
    - Bicep Install or update
      - **Install-AzBicep**
      - **Install-AzBicep -Force**
        - Forces update to the latest version without confirmations

Getting Started - Tooling
Command line tools

- Visual Studio Code (VS Code)
  - Windows, Linux, Mac
    - Download and install the corresponding file
- Bicep Extension for VS Code
  - Is available here
    - Validation, Intellisense, Refactoring
    - Commands
      - Insert Resource from Azure

Getting Started - Tooling
Command line tools

- **Azure Resource Manager (ARM) Tools**
  - Is available here
  - Provides additional
    - Linting
      - Misspelled function or resource type errors
    - Schema Validation
      - Structure and type validation
      - Errors
        - When specifying an invalid VM size identifier
        - Settings that are not compatible with Azure Region

Questions ?

Creating your first Bicep Template

- Our Bicep template will
  - Support user specified storage account name
  - Create the storage account:
    - In the same location as the resource group
      - The resource group will be provided when run
    - Use "**Standard_LRS**" through **sku name**
      - Standard locally redundant storage
    - Use **StorageV2** through **kind**
      - General Purpose V2
    - **Deny** public blob access through **allowBlobPublicAccess**
    - Require TLS1_2 through **minimumTlsVersion**
      - Transport layer security version 1.2

Creating your first Bicep Template

- Example template

```
// Example deployment cmdline: (must be all on the same line broken out here for clarity)
// az deployment group create --name $("createStorageAccount_" + (Get-Date -Format "yyyyMMddHHMMss"))
//                            --resource-group rg_lecture16  --template-file createStorageAccount.bicep
//                            --parameters storageAccountName='<storage account name>'
// Parameter to specify the name of the storage account
@description('The name of the storage account to create')
param storageAccountName string

// Parameter to specify the location of the storage account,
// defaulted from resource group's location
@description('The location of the storage group, defaults to the resource group location.')
param location string = resourceGroup().location

// Parameter to specify the SKU name of the storage account
@description('The the SKU defaults to Standard_LRS')
param skuName string = 'Standard_LRS' // Standard Locally-Redundant Storage

// Parameter to specify the access tier of the storage account
@description('The access tier of the storage account, defaults to Hot')
param accessTier string = 'Hot'

// Parameter to specify if public access is allowed
@description('Specifying if public access is allowed, default is false')
param allowPublicAccess bool = false

@description('Indicates the default network action. Default is Allow ')
@allowed(['Allow','Deny'])
param networkDefaultAction string ='Allow'
```

Creating your first Bicep Template

- Example template

```
// Resource block to create a storage account
resource storageAccount 'Microsoft.Storage/storageAccounts@2021-04-01' = {
  name: storageAccountName
  location: location
  sku: {
    name: skuName
  }
  kind: 'StorageV2'
  properties: {
    accessTier: accessTier
    minimumTlsVersion: 'TLS1_2'
    allowBlobPublicAccess: allowPublicAccess
    networkAcls: {
      defaultAction: networkDefaultAction
    }
  }
}
```

Creating your first Bicep Template

Deployment using Azure CLI

```
// Azure CLI command to create a new deployment at the resource group level.
az deployment group create
// These are parameters to the AZ Command
// Name of the deployment with the current date appended
// Visible in the Resource Group Deployments area
--name $("createStorageAccount_" + (Get-Date -Format "yyyyMMddHHMMss"))
// The resource group where the storage account will be deployed to
// also defines the default location for the storage account
--resource-group rg_lecture16
// The bicep file to be deployed
--template-file createStorageAccount.bicep

// These are the parameters to the Bicep script
--parameters storageAccountName='stbicepdemo01' networkDefaultAction='Deny'
```

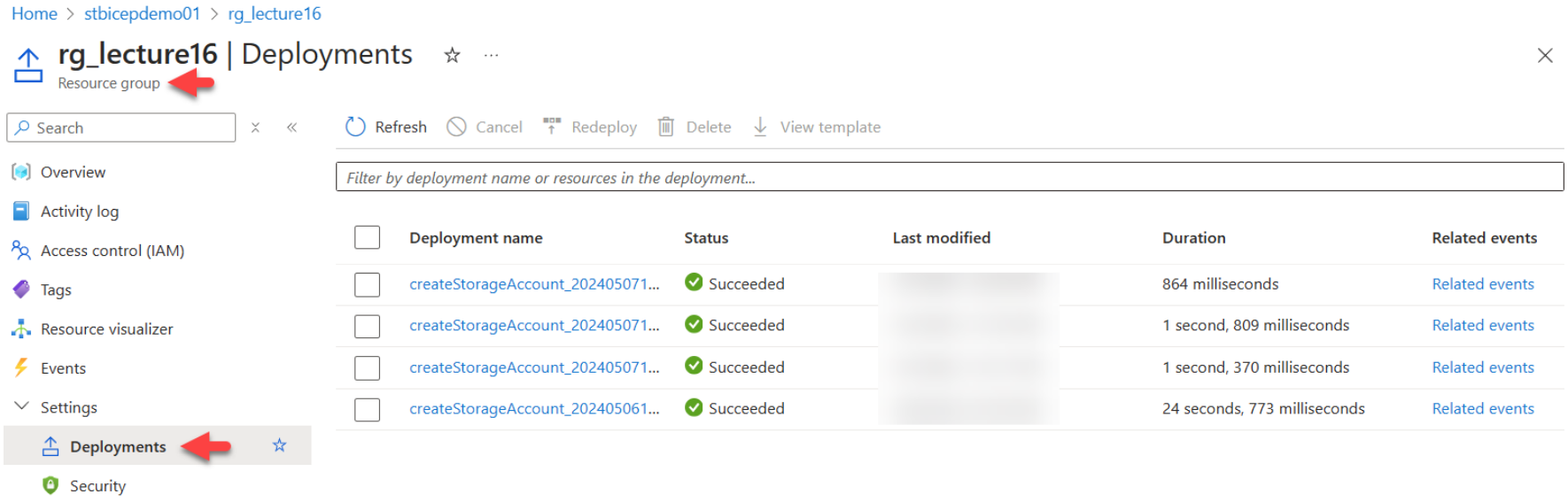
Creating your first Bicep Template

Deployment using Azure CLI - Full Command Line

```
az deployment group create --name $("createStorageAccount_" + (Get-Date -Format
"yyyyMMddHHMMss"))  --resource-group rg_lecture16  --template-file
createStorageAccount.bicep --parameters storageAccountName='stbicepdemo01'
networkDefaultAction='Deny'
```

Getting Started - Tooling
Command line tools

- **Resource Group - Deployments**
  - Supports
    - View, Redeploy
    - And … Download associated ARM Templates!



Demo

Creating & Deploying a Bicep Template

```
StorageAccountExamples
    createStorageAccount.bicep
    createStorageAccountEx.bicep
Azure Deployment
```

Questions ?

Bicep Language Overview

- Data types
  - Primitive Types
    - **string**, **int**, **bool**, **array**
  - Object
    - Composite type
    - Can include multiple properties of different types
  - **SecureString** & **SecureObject**
    - Used for sensitive data that should not be exposed in logs or the Azure portal

Bicep Language Overview

- Syntax Elements
  - Parameters
    - Used to pass external values into your Bicep file
    - Syntax:
      - **param parameterName dataType = defaultValue**
  - Variables
    - Used to simplify complex expressions & readability
    - Syntax:
      - **var variableName = expression**

Bicep Language Overview

- Syntax Elements …
  - Resources
    - Core components for defining Azure Resources
    - Syntax:
      - **resource reasourceName 'ResourceType@Version' = { }**
        - **{ }** Contains the configuration of the resource
  - Outputs
    - Used to export values from a deployment
    - Syntax:
      - **output outputName dataType = expression**

Bicep Language Overview

- Modularity
  - Modules
    - Support for creating modular templates by using other Bicep files.
    - Syntax:
      - module moduleName 'path/module.bicep' = { name: value}
        - { name: value} allows passing parameters to the module
  - Built-in Functions
    - Large number of built-in functions for
      - strings, arrays and other types

Bicep Language Overview

- Operators
  - Supports many of the "Standard" operators

Symbol	Type of Operation	Associativity
`( ) [ ] . ::`	Parentheses, array indexers, property accessors, and nested resource accessor	Left to right
`! -`	Unary	Right to left
`% * /`	Multiplicative	Left to right
`+ -`	Additive	Left to right
`<= < > >=`	Relational	Left to right
`== != =~ !~`	Equality	Left to right
`&&`	Logical AND	Left to right
`||`	Logical OR	Left to right
`??`	Coalesce	Left to right
`? :`	Conditional expression (ternary)	Right to left

Bicep Language Overview

- Scope set via **targetScope**
  - Defines scope at which the template is deployed
    - Resource Group
      - Bicep default scoped to the resource group
    - Subscription
      - Simplifies deployment in many cases
      - See: Supported resources
    - Management Group
      - Helpful for larger organizations
        - Administrative group to manage access, policy and compliance across multiple subscriptions
      - See: Supported Resources
    - Tenant
      - Define policies across Microsoft Entra tenant.
        - Apply policies and roles at a global level
      - See: Supported Resources

Bicep Language Overview

- Conditionals
  - Resources can be conditionally deployed
    - Using the 'if' statement
    - Syntax:
      - **resource resourceName 'ResourceType@Version' = if (condition) { }**
        - { } The properties and settings of the resource to be deployed
    - Ternary Operator
      - Used for conditional expressions
      - Syntax:
        - **condition ? trueValue : falseValue**

Bicep Language Overview

- Loops
  - For Loops
    - Iterate over items in a collection, define multiple copies of a resource, module, variable, property or output
    - Syntax
      - [for <index> in range(<startIndex>, <numberOfElements>): { ...}]
      - [for <item> in <collection>: { ... }]
      - [for <item> in items(<object>): { ...}]

Bicep Language Overview

- Loops
  - Limits
    - Only work with values that can be determined at the start of deployment.
    - Loop iterations
      - Can't be a negative number
      - Can't exceed 800 iterations
    - Can't loop a resource with nested child resources.

Questions ?

Creating multiple resources

- Typical to create multiple resources
  - Dependencies are supported
  - As an example:
    - An app service has a dependency
      - On an App Service Plan
  - Let's take this example
    - Resource Group
      - App Service Plan
        - App Service

Creating multiple resources

- When creating multiple resources
  - Modules are typically used
  - **main.bicep**
    - Creates the resource group
    - Invokes the other modules
      - **appService.bicep**
        - App Service
        - App Service Plan

Creating multiple resources

- main.bicep

```
25  param location string = 'eastus'
26  param resourceGroupName string = 'rg_lecture02_bicep'
27  param appServicePlanName string = 'asp-linuxpaidbicep'
28  param webAppName string = 'app-linuxdemobicep-cscie94'
29  param planSkuName string = 'B1'
30  param planSkuTier string = 'Basic'
31
32  targetScope = 'subscription'
33
34  resource resourceGroup 'Microsoft.Resources/resourceGroups@2021-04-01' = {
35    name: resourceGroupName
36    location: location
37  }
38
39  module appService 'appService.bicep' = {
40    name: 'appService'
41    scope: resourceGroup
42    params: {
43      appServicePlanName: appServicePlanName
44      webAppName: webAppName
45      location: location
46      planSkuName: planSkuName
47      planSkuTier: planSkuTier
48    }
49  }
50
```

The name of the deployment for the app service and app service plan

Creating multiple resources

- appService.bicep

```
22  param appServicePlanName string
23  param webAppName string
24  param location string
25  param planSkuName string = 'B1'
26  param planSkuTier string = 'Basic'
27
28  resource appServicePlan 'Microsoft.Web/serverfarms@2021-01-01' = {
29    name: appServicePlanName
30    location: location
31    properties: {
32      reserved: true // This is for Linux
33    }
34    sku: {
35      name: planSkuName
36      tier: planSkuTier
37    }
38  }
39
40  resource webApp 'Microsoft.Web/sites@2021-01-01' = {
41    name: webAppName
42    location: location
43    properties: {
44      serverFarmId: appServicePlan.id
45      httpsOnly: true
46    }
47  }
48
```

Demo

Creating multiple resources

AppServiceExample

main.bicep, appService.bicep

StorageAccountExample

createStorageAccountWithRg.bicep

createStorageAccountModule.bicep

Azure Deployment

Questions ?

Best Practices

- Modular Design
  - Break down into reusable modules
- Parameterization
  - Don't hard-code values
    - Parameterize with defaults
- Use Descriptions
  - Include descriptions to improve clarity
- Setup Resource Dependencies
  - Define dependencies to reduce deployment failures

Best Practices

- Secure Secret Management
  - Use **Azure KeyVault** to store secrets
    - Don't hard code secrets in bicep files
- Error Handling
  - Use error handing and conditionals to manage
    - Deployment based on criteria
- Version Pinning
  - Pin version of modules & resources
    - Avoids unexpected changes due to updates in dependent resources

Best Practices

- Version Pinning
  - Here the API version specified is 2021-04-01

```
resource resourceGroup 'Microsoft.Resources/resourceGroups@2021-04-01' = {
  name: resourceGroupName
  location: location
}
```

Best Practices

- Idempotency
  - Bicep is idempotent
    - You can create non-idempotent scripts though
      - Conditional deployments based on external factors not in template
      - External resource modifications
        - Prevent them as Bicep may not revert the changes

Questions ?

Further Reading

- Bicep Documentation
- What is Bicep?
- Bicep syntax and structure
- Target Scope - Resource Group
- Target Scope - Subscription
- Deploy using VS Code
- Deploy using CLI
- Create Bicep files with VS Code
- Frequently Asked Questions