Write up topics like deploying function apps on my blog

Release YouTube videos of the function app specific

Course Title

Description

Learning objectives

Preview image

Preview video

**Video Introduction**

* Azure Bicep is a brand new powerful infrastructure-as-code tool for deploying and managing Azure resources. This new approach replaces difficult ARM templates with an easy to use and streamlined language.

Introduction – what is bicep?

List of things they will learn

Fast forward of me writing a complicated bicep file

Pipeline deploying (validating then deploying)

\*\* the most through Udemy course

TODO

Watch existing videos on YouTube for recording ideas

Mac

deployment steps

az login

az deployment group create --subscription udemy-courses --resource-group bicep-course --name deployment --template-file main.bicep

**--mode**

9f53062f-e565-443b-8eb8-63aeb88dc7fd

Software

Kap

Quicktime

**Screenium – paid**

handbrake app – converting videos

Equipment

Deploying resource groups, deployment scopes Where should I show them deploying resource groups?

Dependencies – implicit, explicit, Visualize dependencies

**Lesson 1: Introduction to Azure Bicep**

**Lesson 2: Creating your first Azure Bicep template**

**Lesson 3: Deploying resources with Azure Bicep**

**Lesson 4: Variables, parameters and outputs**

**Lesson 5: Bicep functions**

**Lesson 6: Working with modules in Azure Bicep**

**Lesson 7: Advanced Bicep concepts (loops, conditional deployments, existing resources)**

**Lesson 8: Advanced Bicep functions (files, dates, lambdas)**

**Lesson 9: Testing and validating Azure Bicep templates**

**Lesson 10: Troubleshooting Azure Bicep deployments**

**Lesson 11: Real-world Project – deploying a function app with logging**

**Lesson 12: Preview features**

**Lesson 1: Introduction to Azure Bicep**

* What is Infrastructure as Code (IaC)? Why use IaC?
* What is ARM?
* Introduction to Bicep
* Why use Bicep over other IaC tools?
* Terraform vs bicep
  + consistency in configurations across environments, no more ‘snowflake’ configurations
  + reduced human error in deployments
  + increased productivity of speed of changes
  + prevents environments contention with ability to dynamically create new infra
* Setting up the environment for Bicep
* VS code, bicep extension

Host (On-Screen): Welcome to our Azure Bicep course! In this video, we'll introduce you to Azure Bicep, a powerful infrastructure-as-code tool for deploying and managing Azure resources.

[Title Slide: Introducing Azure Bicep]

Azure Bicep is a brand new

Host (On-Screen): Azure Bicep is a domain-specific language (DSL) designed specifically for Azure deployments. It provides a simplified and streamlined approach to provisioning and managing Azure resources, making infrastructure-as-code more accessible and efficient.

[Slide 1: Benefits]

Host (On-Screen): So, why should you consider using Azure Bicep? Well, let's take a look at some of its key benefits.

[Slide 2: Simplicity]

Host (On-Screen): First and foremost, Azure Bicep offers a simplified syntax and structure. Its syntax resembles JSON, making it easy to read and write. With Bicep, you can express your infrastructure requirements concisely and clearly, reducing the chances of errors and improving the overall development experience.

[Slide 3: Familiarity]

Host (On-Screen): Another advantage of Azure Bicep is its familiarity. If you've worked with Azure Resource Manager (ARM) templates before, you'll find Bicep quite familiar. Bicep is built on top of ARM templates, providing a higher-level abstraction that simplifies the authoring process while still leveraging the power of ARM underneath.

[Slide 4: Integration]

Host (On-Screen): Azure Bicep integrates seamlessly with Azure. It leverages Azure Resource Manager (ARM) to deploy and manage resources, allowing you to take full advantage of Azure's extensive capabilities. Bicep makes it easy to define and deploy complex Azure infrastructure with just a few lines of code.

[Slide 5: Maintainability]

Host (On-Screen): With Azure Bicep, maintaining your infrastructure becomes more manageable. Bicep supports modularity and reusability through modules, allowing you to organize your codebase efficiently. Changes and updates can be made easily, and you can track the state of your infrastructure using version control systems.

[Slide 6: Collaboration]

Host (On-Screen): Collaboration is crucial in any development process. Azure Bicep promotes collaboration by enabling teams to work together on infrastructure-as-code projects. You can split your code into multiple Bicep files, assign specific sections to different team members, and merge their changes seamlessly.

[Slide 7: Ecosystem]

Host (On-Screen): Azure Bicep benefits from being part of the broader Azure ecosystem. It integrates with Azure DevOps, Azure CLI, and Azure PowerShell, providing a cohesive workflow for your Azure deployments. Additionally, Bicep leverages existing ARM templates, enabling you to reuse and leverage the vast collection of ARM templates available.

Host (On-Screen): In conclusion, Azure Bicep offers simplicity, familiarity, seamless integration with Azure, improved maintainability, enhanced collaboration, and an extensive ecosystem—all aimed at making infrastructure-as-code on Azure more efficient and accessible.

Host (On-Screen): Throughout this course, we'll dive deep into Azure Bicep, exploring its features, best practices, and practical examples to help you become proficient in deploying and managing Azure resources using Bicep.

Host (On-Screen): That wraps up our introduction to Azure Bicep. Get ready for an exciting journey into the world of Azure infrastructure-as-code. Happy learning!

* 



Windows

<https://learn.microsoft.com/en-us/cli/azure/install-azure-cli-windows?tabs=azure-cli>

Mac

brew install azure-cli

az bicep install

**Lesson 2: Creating your first Azure Bicep template**

* Create your first Bicep file
* Converting bicep file to ARM template

We are going dive straight in and create our first Bicep deployment template. So we need to create a new file with the bicep extension, which I’m I’m going to call main.bicep.

* Declare resource with resource
* Type name of the resource – this is a reference for the resource in the bicep file
* Autocomplete the resouece type, choose the API version
* Use autocomplete for the required properties
* Type the name, this is the name of the resource
* Location – hardcode to wesreurope

**Lesson 3: Deploying resources with Azure Bicep**

* Overview of deployment options Azure CLI, PowerShell, Azure DevOps, etc.)
* Deployment of Bicep files with Azure CLI
* Deployment of Bicep files with PowerShell
* Integrating Bicep deployments of Bicep files with Azure DevOps Pipelines
  + <https://dev.azure.com/dylanbudgen/udemy-courses/_build/results?buildId=2&view=logs&j=12f1170f-54f2-53f3-20dd-22fc7dff55f9>
* What are deployment modes? .
  + Validation, Incremental, Complete

**Lesson 4: Variables, parameters and outputs**

* Using variables in Bicep
* Using parameters in Bicep
  + Lower camel case
* Using outputs in Bicep
* Data types

**Lesson 5: Bicep functions**

* Introduction to Bicep functions
* Commonly used Bicep functions
  + Data type conversions?
  + Scope functions
  + Resource functions
  + Guid functions
  + Array functions
  + String functions
  + Numeric functions

**Lesson 6: Working with modules in Azure Bicep**

* What are Bicep modules?
* How to create and use Bicep modules.
  + Storage account, create one manually, then change to module
  + Explain that it creates new deployment
  + Explain the name different module and resource

**Lesson 7: Advanced Bicep concepts (loops, conditional deployments, existing resources)**

* Nested resources in Bicep
  + creation of single container in storage account module
* Loops in Bicep
  + Creating array of containers (follow up from parents)
* Explore different types of loops?
* Working with existing resources
  + role assignments (also show them with a different RG)
  + introduce with a single role assignment, then add loop for complex loops
* Managing dependencies in Bicep
  + RBAC module being run after the storage accounts deployed
  + Modules vs resources – show this by creating a storage account without the module
  + Dependency management with resources and modules
  + Implicit vs explicit (explicit is discouraged)
* Conditional deployments in Bicep
  + audit storage account with Boolean param
* Ternary operators in Bicep
  + The conditional deployment will need a local variable for storage account names in the RBAC module – follow up from last module
* **Logical operators**

**Lesson 8 – Advanced Bicep functions (files, dates, lambdas)**

* Date functions
* Lambda functions
* File functions

**LESSON 7 HAS CHANGED LOTS – MAYBE UPDATE THE REST**

**Lesson 9: Testing and validating Azure Bicep templates**

* Validating Bicep using Azure CLI
  + Using linter in vs code
  + Az bicep build
    - Reports warnings
* Testing Bicep and ARM templates
  + Arm-ttk (doesn’t look very simple)
  + Installation and running tests
  + Mention API versions and why you would wantto skip tests

**Lesson 10: Troubleshooting Azure Bicep deployments**

* Debugging Bicep deployments and troubleshooting common issues
* Understanding deployment logs and error messages
  + Template validation (predeployment) - Storage account wrong length
  + Runtime failure – invalid value for minimumTlsVersion
  + Mention that nested resources sometimes don’t produce good results, look at deployment logs
* Validating Bicep with Azure DevOps Pipelines
  + <https://dev.azure.com/dylanbudgen/udemy-courses/_build/results?buildId=2&view=logs&j=12f1170f-54f2-53f3-20dd-22fc7dff55f9>

**Lesson 10: Best Practices and Tips – SHOULD I INCLUDE THIS?**

* Bicep best practices
* Tips for writing efficient Bicep templates
* https://learn.microsoft.com/en-us/azure/azure-resource-manager/bicep/best-practices

**Lesson 11: Real-world project – deploying a function app**

* Building a real-world infrastructure setup using Bicep
  + Deploy using pipeline with variables?
* Think of how I can use advanced topics
  + Type the compute module with the wrong type of dependencies values, then correct it
  + Also show them that storageAccount.name is the deployment name

**Lesson 12: Preview features**

* Import for k8s
  + <https://learn.microsoft.com/en-us/azure/azure-resource-manager/bicep/bicep-import-providers>
* Object types

**Lesson 11: Review and Next Steps**

Review of the course

Next steps for learning more about Bicep and IaC