



Got to Deploy Them All

Learning how to stop clicking and move to DevOps for deploying Azure Resources





Speakers



Scott Corio

MMS Staff, Microsoft MVP



DumpsterDave



scottcorio



David Brook

CloudXP



@DBBrook24

Can't see our slides? Can't hear? Need to repeat the question? Call us out!



What is in this Session?

- ◆ Learn about Azure Resource Manager (ARM) Deployments
- ◆ Learn about why and how to use Bicep for deploying Azure Resources
- ◆ Learn about using this to start your journey to DevOps
- ◆ How to write and deploy Bicep for resources like storage accounts, networks, VMs etc
- ◆ Already used to ARM and looking to find ways to easily migrate your code and knowledge? We'll show you how.
- ◆ Learn how to take all these concepts and start creating pipelines for a true DevOps approach

ARM? I have two of those right?

Azure Resource Manager templates, what are those?

What is ARM?

Definition:

- ARM is a domain-specific language (DSL) for deploying Azure resources declaratively in a JSON format.

Why Use ARM?

- **Repeatable Results:** Deploy your resources over and over with the same results.
- **Orchestration:** ARM orchestrates your deployments of dependent resources.
- **Integrated with Azure:** Deep integration with Azure tools like VSCode, Azure CLI, and Azure DevOps.

Drawbacks?

- **Steep Learning Curve:** Compared to other deployment resources, ARM templates can be a steep learning curve.
- **Complex Structures:** ARM Templates can become very complex, very quickly.

ARM Basics

- ◆ Resource (Red)
 - ◆ Defining that you want to create a resource
- ◆ Symbolic Name (Blue)
 - ◆ A unique name for this resource which can be used for reference in other resources and outputs.
- ◆ Type & API Version (Green)
 - ◆ The Type of resource you want to deploy, and the API version you wish to use.
- ◆ ARM File Structure Information
 - ◆ <https://learn.microsoft.com/en-us/azure/azure-resource-manager/templates/syntax>

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]"
    }
  },
  "resources": {
    "mystore": {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2023-04-01",
      "name": "mystorageaccount",
      "location": "[parameters('location')]",
      "sku": {
        "name": "Standard_LRS"
      },
      "kind": "StorageV2"
    }
  }
}
```

A tropical beach scene with palm trees, mountains, and a flock of flamingos. The background features a dark, dramatic sky with large, white, billowing clouds. In the distance, a range of jagged, dark mountains rises above a line of green vegetation. To the right, several tall palm trees lean towards the left. In the foreground, a flock of pink flamingos stands on a sandy beach, with some wading in the shallow, turquoise water. The overall lighting is soft and warm, suggesting a sunset or sunrise.

Bicep? Are we going t' Gym?

Why Bicep, and why should we use it?

What is Bicep?

Definition:

- Bicep is a domain-specific language (DSL) for deploying Azure resources declaratively. It's an abstraction layer over ARM (Azure Resource Manager) templates that simplifies infrastructure as code.

Why Use Bicep?

- **Simplified Syntax:** Easier to read and write compared to JSON-based ARM templates.
- **Modular and Reusable:** Supports the creation of modules to promote code reuse.
- **Integrated with Azure:** Deep integration with Azure tools like VSCode, Azure CLI, and Azure DevOps.
- **Easier Maintenance:** Cleaner code structure reduces errors and makes template maintenance simpler.
- **Automatic Translation:** Bicep is transpiled into ARM templates, making it backward-compatible with existing ARM infrastructure.

Bicep Basics

- ◆ Resource (Red)
 - ◆ Defining that you want to create a resource
- ◆ Symbolic Name (Blue)
 - ◆ A unique name for this resource which can be used for reference in other resources and outputs.
- ◆ Type & API Version (Green)
 - ◆ The Type of resource you want to deploy, and the API version you wish to use.
- ◆ Bicep File Structure Information
 - ◆ <https://learn.microsoft.com/en-us/azure/azure-resource-manager/bicep/file>

```
resource my_resource 'Microsoft.Storage/storageAccounts@2023-05-01' = {  
  name: 'mystorageaccount'  
  location: 'eastus'  
  kind: 'StorageV2'  
  sku: {  
    name: 'Standard_LRS'  
  }  
}
```

Usage in a Bicep File

Example Structure:

- ◆ **Main Module File (e.g., vnet.bicep):** Contains the core logic for deploying a virtual network, accepting parameters like the address space, subnets, and resource group.
- ◆ **Parameters:** Input values required by the module (e.g., addressPrefix, subnetNames).
- ◆ **Resources:** The Azure resources that the module will deploy (e.g., a Microsoft.Network/virtualNetworks resource).
- ◆ **Outputs:** Values returned by the module after deployment (e.g., the virtual network's ID).

```
module vnetModule './vnet.bicep' = {  
  name: 'vnetDeployment'  
  params: {  
    addressPrefix: '10.0.0.0/16'  
    subnetNames: ['subnet1', 'subnet2']  
  }  
}  
  
output vnetId string = vnetModule.outputs.vnetId
```

A tropical beach scene with palm trees, mountains, and a flock of flamingos. The background features a dark, moody sky with large, white clouds. In the distance, there are jagged, dark mountains. The foreground shows a sandy beach with a flock of pink flamingos standing near the water's edge. The water is a deep blue-green color. The overall atmosphere is serene and exotic.

DevOps? What's the big deal?

A journey that can change your views!

Journey to DevOps

Why Infrastructure as Code?

- Infrastructure as Code (IaC) is a fundamental practice in modern cloud environments, enabling the automation and management of infrastructure through machine-readable definition files, rather than through physical hardware configuration or interactive configuration tools.
- IaC helps ensure consistency, repeatability, and scalability in deploying and managing infrastructure, making it easier to adopt DevOps practices.

Version Control with Bicep

- **Git Integration:** Store your Bicep files in Git repositories to leverage version control. This allows you to track changes, collaborate with team members, and roll back to previous versions if necessary.

Consistency Across Environments

- **Environment Parity:** Use Bicep to ensure that your development, testing, and production environments are consistently configured. By using the same Bicep files across environments, you reduce the risk of configuration drift and errors.
- **Reusability:** Define reusable modules in Bicep to standardize resource configurations across different projects and teams. This promotes best practices and reduces duplication of effort.

Laying the Foundation for DevOps

- **Automated Deployments:** As the first step in your DevOps journey, start by automating your infrastructure deployments using Bicep scripts. This sets the stage for more advanced automation and CI/CD practices down the road.
- **Scalability:** By adopting IaC with Bicep, you position your organization to easily scale infrastructure as demand grows, ensuring that resources are provisioned efficiently and consistently.



Journey to DevOps

Step 1: Automate Deployments

- **Scripted Deployments:** Use Bicep in combination with PowerShell, Azure CLI, or Cloud Shell to automate resource deployments.
- **Repeatable and Consistent:** Ensures that every environment (dev, test, prod) is provisioned consistently.

Step 2: Introduce CI/CD Pipelines

- **Build Pipelines:** Integrate Bicep into Azure DevOps pipelines to automate the validation and compilation of your infrastructure code.
- **Release Pipelines:** Automate the deployment process using Azure DevOps or GitHub Actions, enabling continuous delivery of infrastructure changes.

Step 3: Enhance with DevOps Practices

- **Testing:** Incorporate testing frameworks to validate your infrastructure before deployment.
- **Monitoring & Feedback:** Set up monitoring and alerts for deployed resources, feeding back into the DevOps cycle for continuous improvement.
- ◆ **End Goal: Continuous Delivery of Infrastructure**
- **Full Automation:** Achieve fully automated infrastructure deployment, from code commit to production, using Bicep within your DevOps pipelines.
- **Scalability & Flexibility:** Scale your DevOps practices across teams and projects, making your infrastructure agile and responsive to change.

A tropical beach scene with palm trees, mountains, and a flock of flamingos. The background features a dark, dramatic sky with large, white, billowing clouds. In the distance, a range of jagged, dark mountains rises from the water. The foreground shows a sandy beach with a flock of pink flamingos standing near the water's edge. The water is a deep blue-green color. The overall lighting is dim, suggesting dusk or dawn.

Let's Get Flexing!!

Popeye would be jealous!!

Let's get building!

- ◆ Creating a basic ARM and Bicep file from scratch
 - ◆ How to interpret the documentation (psst!! It not always clear)
- ◆ Can we use cheatcodes??
 - ◆ Oh yeah!!
 - ◆ <https://resources.azure.com/>
 - ◆ Learn how to easily get started by creating resources in the GUI and then convert them
 - ◆ Using CMDline
 - ◆ Using VSCode
- ◆ Azure Verified Modules
 - ◆ <https://azure.github.io/Azure-Verified-Modules/>
 - ◆ <https://aka.ms/AVM>
 - ◆ Supported Modules (<https://aka.ms/AVM/support>)

DEMO

Give them a Curl!!

Creating a Bicep file!!



A tropical beach scene with palm trees, mountains, and a flock of flamingos. The background features a dark, dramatic sky with large, white, billowing clouds. In the distance, a range of jagged, dark mountains rises above a line of green trees. To the right, several tall palm trees lean towards the left. In the foreground, a flock of pink flamingos stands on a sandy beach, with some wading in the shallow, turquoise water. The overall lighting is soft and warm, suggesting a sunset or sunrise.

Let's get some reps in!

How to deploy your templates!

Deployment Methods

- ◆ PowerShell
 - ◆ Great for Testing
- ◆ DevOps
 - ◆ IaC FTW!
 - ◆ Service Connections
 - ◆ Pipelines
- ◆ Azure Cloud Shell



DEMO

Pump that Iron!!

Let's break a sweat!!





Let's add more weights!!

Let's talk about more advanced techniques!

More Advanced Bicep Techniques

◆ Modules

- ◆ A Bicep module is designed to be reusable across multiple deployments. Instead of duplicating code, you can define a module once and use it wherever you need to deploy similar resources.
- ◆ Modules encapsulate the logic for creating resources, hiding the implementation details. This allows you to use a module without needing to understand the intricacies of its internal workings.

◆ Parameters and Outputs:

- ◆ Modules typically accept input parameters and produce outputs, allowing them to be flexible and adaptable to different scenarios. For example, a storage account module might take parameters for the storage account name, SKU, and access tier, and output the account's connection string.



This Photo by Unknown Author is licensed under [CC BY-NC](#)

DEMO

99, 100!!

Deploying with Advanced Techniques



Summary

- ◆ What have we covered?
 - ◆ ARM
 - ◆ BICEP
 - ◆ Azure DevOps
- ◆ Use you cheat codes!
 - ◆ Azure Verified Modules
 - ◆ Create you resource in Azure, and Export your template.
- ◆ Start Small, grow with your infrastructure.
- ◆ Non-Domain Specific Languages
 - ◆ Ansible
 - ◆ <https://learn.microsoft.com/en-us/azure/developer/ansible/overview>
 - ◆ Terraform
 - ◆ <https://learn.microsoft.com/en-us/azure/developer/terraform/overview>

Save the Dates



May 4-8, 2025



Oct 12-15, 2025



May 3-7, 2026



PATCH MY PC

ninjaOne®

2Pint



RECAST SOFTWARE

1E

eido



VIAMONSTRA
ONLINE ACADEMY

systemcenterdudes

Extended Q&A

MMS
10 YEARS

