Project Bicep: ARM Templates RELOADED

Esther Barthel

@virtuEs_IT

github.com/cognitionit

Microsoft MVP

Freek Berson

@fberson

github.com/fberson

Microsoft MVP


ROADSHOW
April 29th Technine User Group
April 7th Dutch DevOps & Github comm.
February 17th Ask Wortell Live Stream
February 4th Azure Thursday
January 29th Azure Lowlands


BE A
NINJA
UNICORN

Agenda

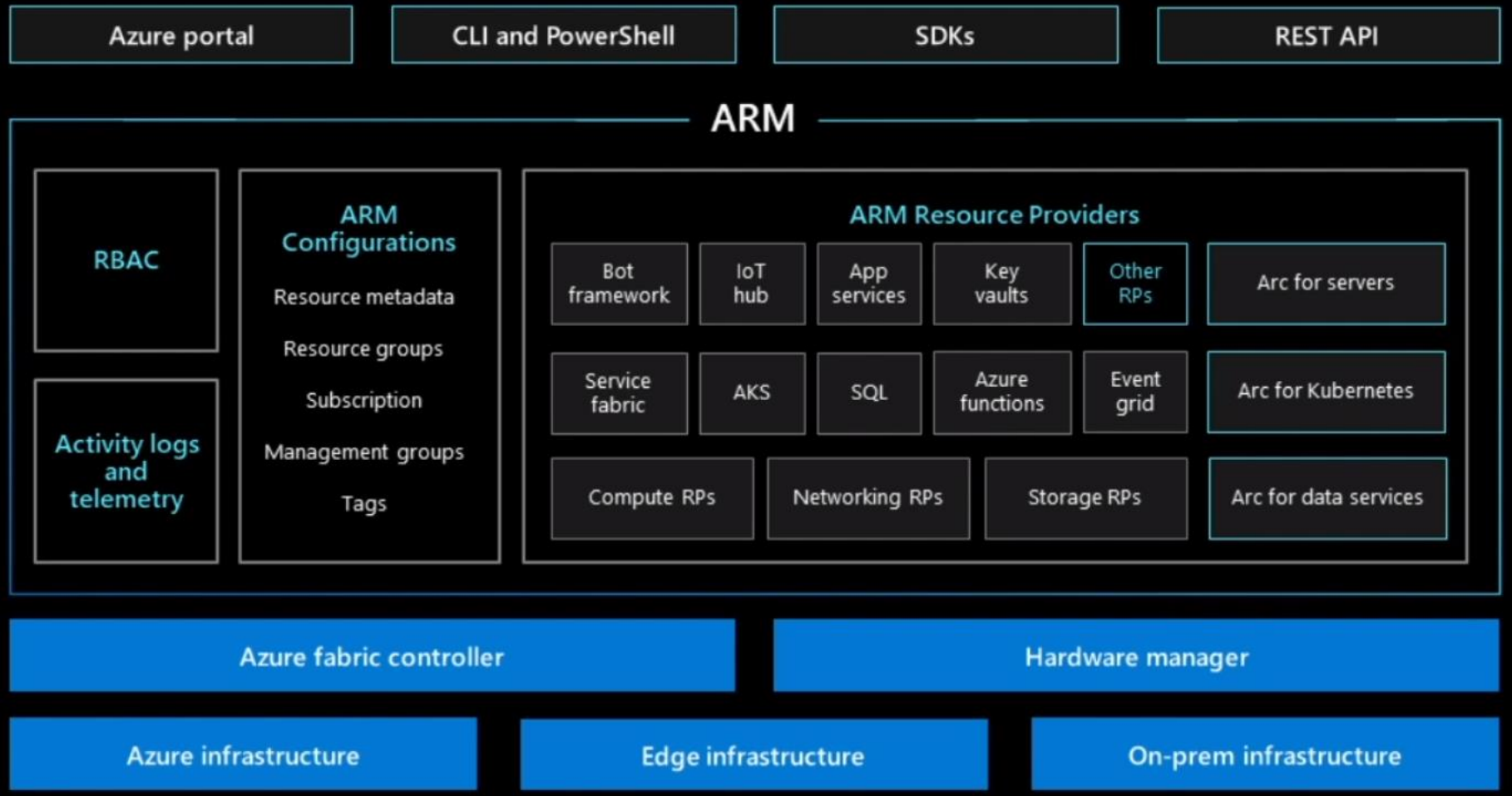
Azure Resource Manager & JSON

Project 'Bicep' architecture

Demos, demos, demos!

Bicep Roadmap & call to actions

Azure Resource Manager



How to get started with ARM & JSON

… structure of an **Azure Resource Manager template**.

The template consists of **JSON** and expressions that you can use to construct values for your deployment.

ARM Template

Template format

In its simplest structure, a template has the following elements:

JSON Copy

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "",
  "apiProfile": "",
  "parameters": {  },
  "variables": {  },
  "functions": [  ],
  "resources": [  ],
  "outputs": {  }
}
```

ARM Template

```
"parameters": {
  "<parameter-name>" : {
    "type" : "<type-of-parameter-value>",
    "defaultValue": "<default-value-of-parameter>",
    "allowedValues": [ "<array-of-allowed-values>" ],
    "minValue": <minimum-value-for-int>,
    "maxValue": <maximum-value-for-int>,
    "minLength": <minimum-length-for-string-or-array>,
    "maxLength": <maximum-length-for-string-or-array-parameters>,
    "metadata": {
      "description"
    }
  }
}
```

```
"variables": {
  "<variable-name>": "<variable-value>",
  "<variable-name>": {
    <variable-complex-type-value>
  },
  "<variable-object-name>": {
    "copy": [
      {
        "name": "<name-of-array-property>",
        "count": <number-of-iterations>,
        "input": <object-or-value-to-repeat>
```

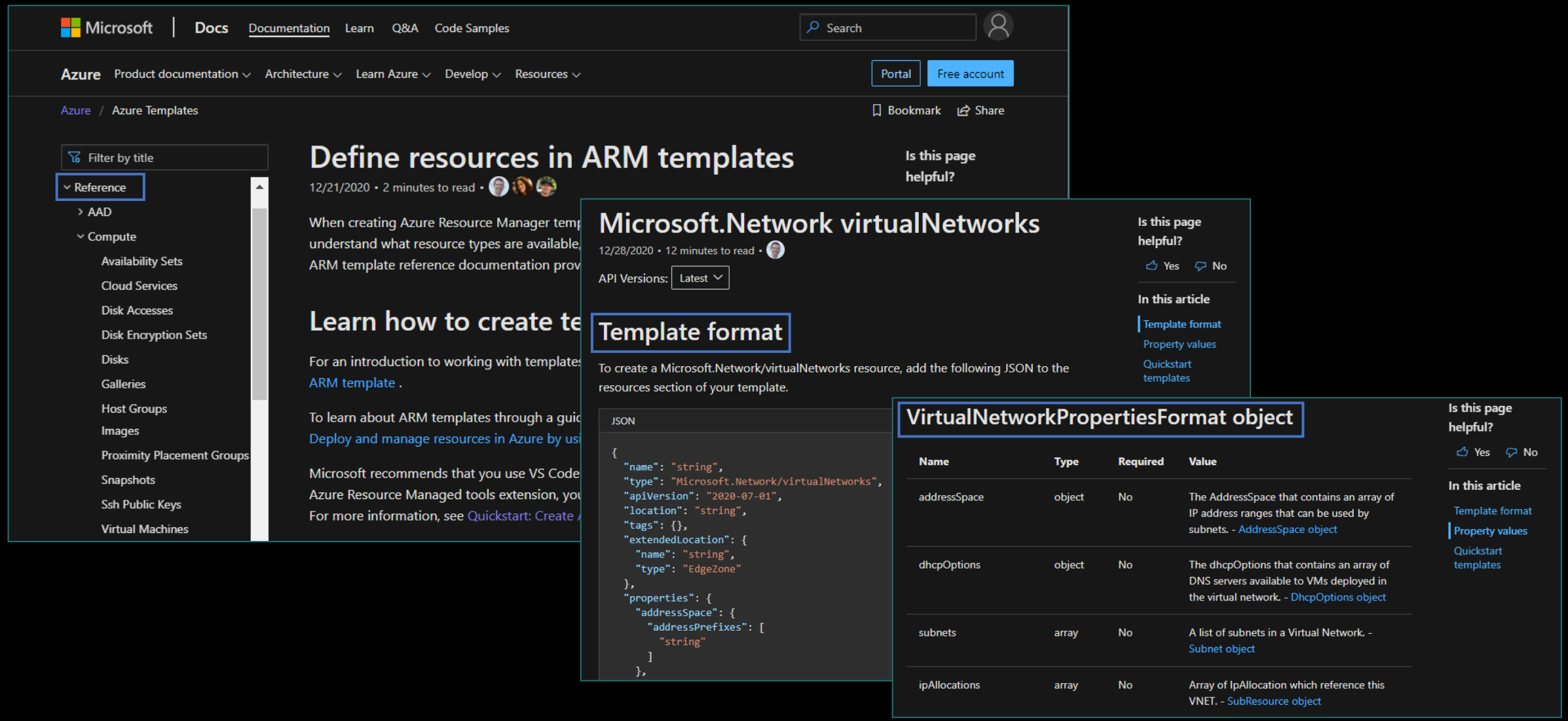
```
[

me": "<variable-array-name>",
unt": <number-of-iterations>,
put": <object-or-value-to-repeat>
```

```
"functions": [
  {
    "namespace": "<namespace-for-functions>",
    "members": {
      "<function-name>": {
        "parameters": [
          {
            "name": "<parameter-name>",
            "type": "<type-of-parameter-value>"
          }
        ],
        "output": {
          "type": "<type-of-output-value>",
          "value": "<function-return-value>"
        }
      }
    }
  }
],
```

```
"outputs": {
  "<output-name>": {
    "condition": "<boolean-value-whether-to-output-value>",
    "type": "<type-of-output-value>",
    "value": "<output-value-expression>",
    "copy": {
      "count": <number-of-iterations>,
      "input": <values-for-the-variable>
    }
  }
}
```

```
"resources": [
  {
    "condition": "<true-to-deploy-this-resource>",
    "type": "<resource-provider-namespace/resource-type-name>",
    "apiVersion": "<api-version-of-resource>",
    "name": "<name-of-the-resource>",
    "comments": "<your-reference-notes>",
    "location": "<location-of-resource>",
    "dependsOn": [
      "<array-of-related-resource-names>"
    ],
    "tags": {
      "<tag-name1>": "<tag-value1>",
      "<tag-name2>": "<tag-value2>"
    },
    "sku": {
      "name": "<sku-name>",
      "tier": "<sku-tier>",
      "size": "<sku-size>",
      "family": "<sku-family>",
      "capacity": <sku-capacity>
    },
    "kind": "<type-of-resource>",
    "copy": {
      "name": "<name-of-copy-loop>",
      "count": <number-of-iterations>,
      "mode": "<serial-or-parallel>",
      "batchSize": <number-to-deploy-serially>
    },
    "plan": {
      "name": "<plan-name>",
      "promotionCode": "<plan-promotion-code>",
      "publisher": "<plan-publisher>",
      "product": "<plan-product>",
```

ARM Template – Reference Guide



ARM Template learning path

https://bit.ly/3qZGNj1

MODULE 1: Deploy Azure infrastructure by using ARM templates

MODULE 2: Deploy to multiple Azure environments by using ARM template features

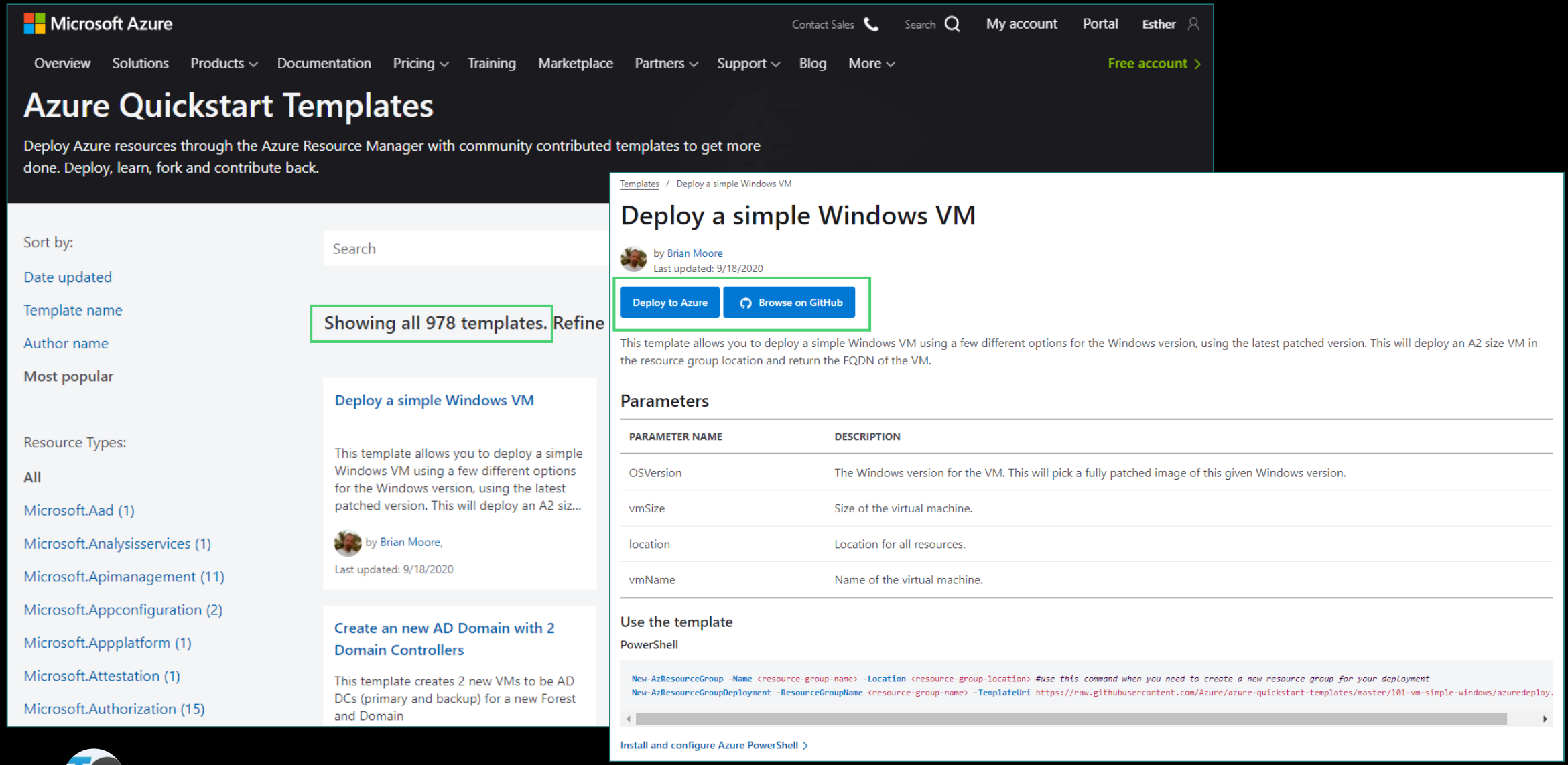
MODULE 3: Preview changes and validate Azure resources by using what-if and the ARM template test toolkit

MODULE 4: Automate the deployment of ARM templates by using GitHub Action

MODULE 5: Extend ARM templates by using deployment scripts

MODULE 6: Manage complex cloud deployments by using advanced ARM template features

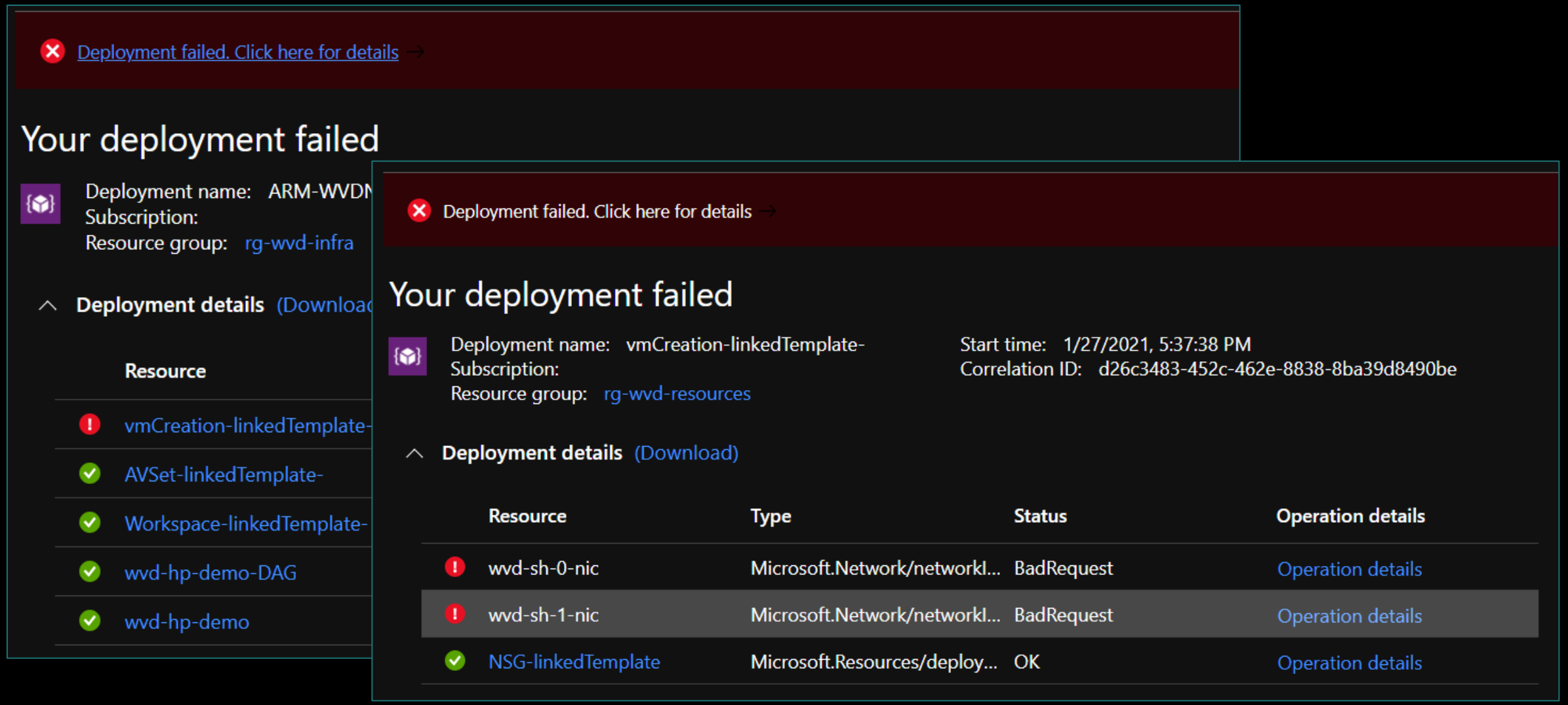
ARM Template – Azure Quickstart Templates



ARM Template – Automatic deployments

```powershell
# ARM Template file
$jsonARMTemplateFile = ".\ARM-WVDNewHostpool.template.json"
$jsonARMParameterFile = ".\ARM-WVDNewHostpool.parameter.json"

# Create WVD Hostpool, based on ARM Template
New-AzResourceGroupDeployment -ResourceGroupName "rg-wvd-infra" `
    -TemplateFile $jsonARMTemplateFile `
    -TemplateParameterFile $jsonARMParameterFile `
    -administratorAccountPassword $secureDomainAdminPassword `
    -vmAdministratorAccountPassword $secureLocalAdminPassword `
    -Verbose
```

ARM Template – Automatic deployments



ARM Template – Automatic deployments

```
{
  "code": "DeploymentFailed",
  "message": "At least one resource deployment operation failed. Please list deployment operations for details. Please see https://aka.ms/DeployOperations for usage details.",
  "details": [
    {
      "code": "InvalidResourceReference",
      "message": "Resource /subscriptions/
                                  /resourceGroups/
rg-wvd-resources/providers/Microsoft.Network/
virtualNetworks/vnet-wvd-resource/subnets/default
referenced by resource /subscriptions/
                                  /resourceGroups/
rg-wvd-resources/providers/Microsoft.Network/
networkInterfaces/wvd-sh-0-nic was not found. Please make
sure that the referenced resource exists, and that both
resources are in the same region."
    },
```

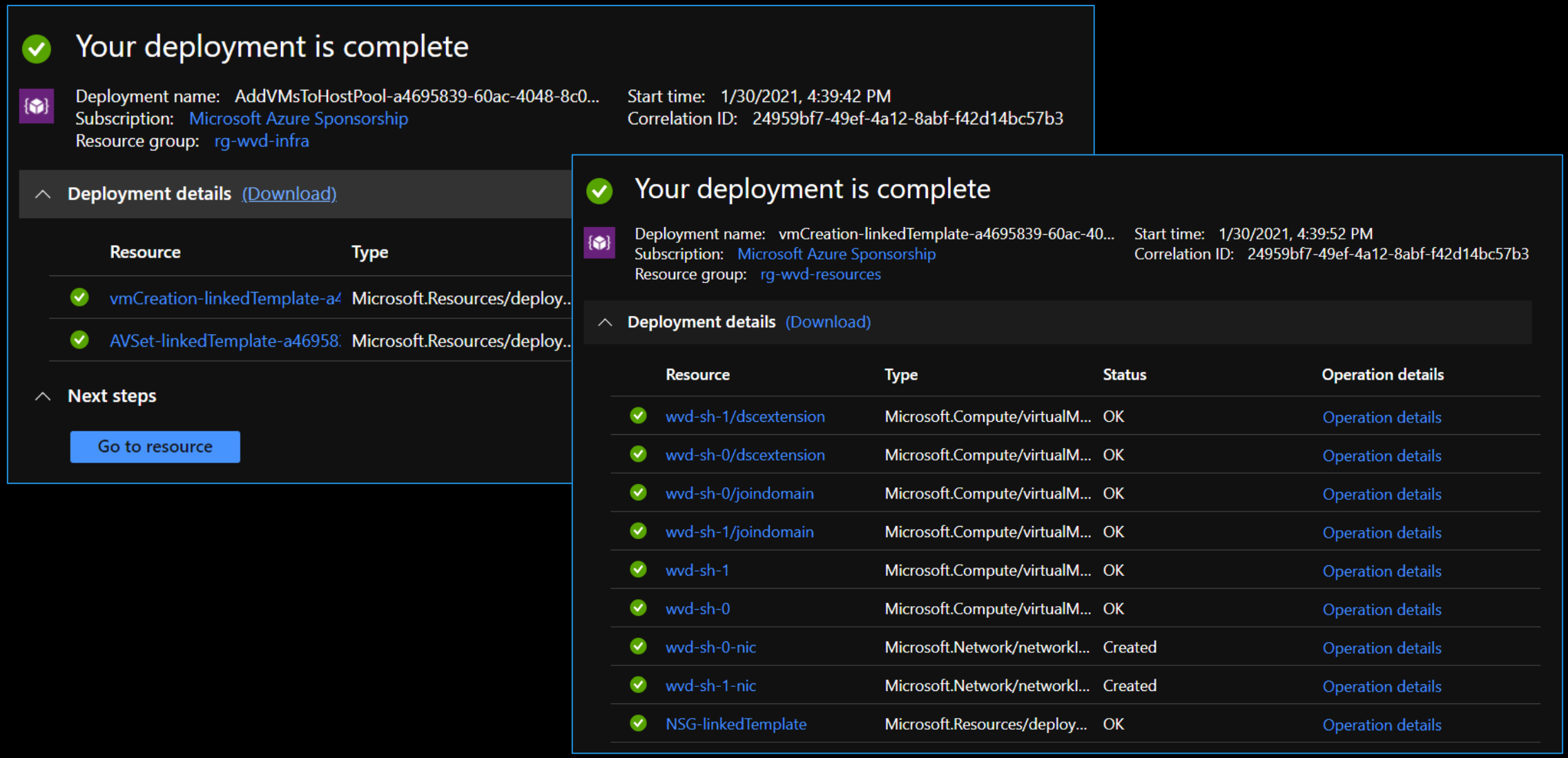
ARM Template – Automatic deployments

```
{
    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0"
    Select or create a p
    "parameters": {
    },
    "variables": {
    },
    "resources": [
        {
            "apiV
            "name
            "type
            "resou
            "condi
            "prope
            }
        },
        {...
        },
        {
            "apiV
            "name
            "resou
            "depe
                "
            ],
            "type
            "prope
            }
        }
    ],
    "outputs": {
        "rdshVmNa
            "valu
            "type
        }
    }
}
```

```
{
    "apiVersion": "2018-05-01",
    "name": "[concat('vmCreation-linkedTemplate-', parameters('deploymentId'))]",
    "resourceGroup": "[parameters('vmResourceGroup')]",
    "dependsOn": [
        "[concat('AVSet-linkedTemplate-', parameters('deploymentId'))]"
    ],
    "type": "Microsoft.Resources/deployments",
    "properties": {
        "mode": "Incremental",
        Linked template
        "templateLink": {
            "uri": "[variables('vmTemplateUri')]",
            "contentVersion": "1.0.0.0"
        },
        "parameters": {
            "artifactsLocation": {
                "value": "[parameters('artifactsLocation')]"
            },
            "vmImageVhdUri": {
                "value": "[parameters('vmImageVhdUri')]"
            },
            "storageAccountResourceGroupName": {
                "value": "[parameters('storageAccountResourceGroupName')]"
            },
            "vmGalleryImageOffer": {
                "value": "[parameters('vmGalleryImageOffer')]"
            },
            "vmGalleryImagePublisher": {
                "value": "[parameters('vmGalleryImagePublisher')]"
            },
            "vmGalleryImageSKU": {
                "value": "[parameters('vmGalleryImageSKU')]"
            }
        }
    }
}
```

- 531 lines of code
- complex JSON formatting
- advanced options:
  - nested templates
  - linked templates

ARM Template – Automatic deployments



What is Project ‘Bicep’?

"..Bicep is a Domain Specific Language (DSL) for deploying Azure resources declaratively. It aims to drastically simplify the authoring experience with a cleaner syntax and better support for modularity and code re-use. Bicep is a transparent abstraction over ARM and ARM templates.

Project ‘Bicep’

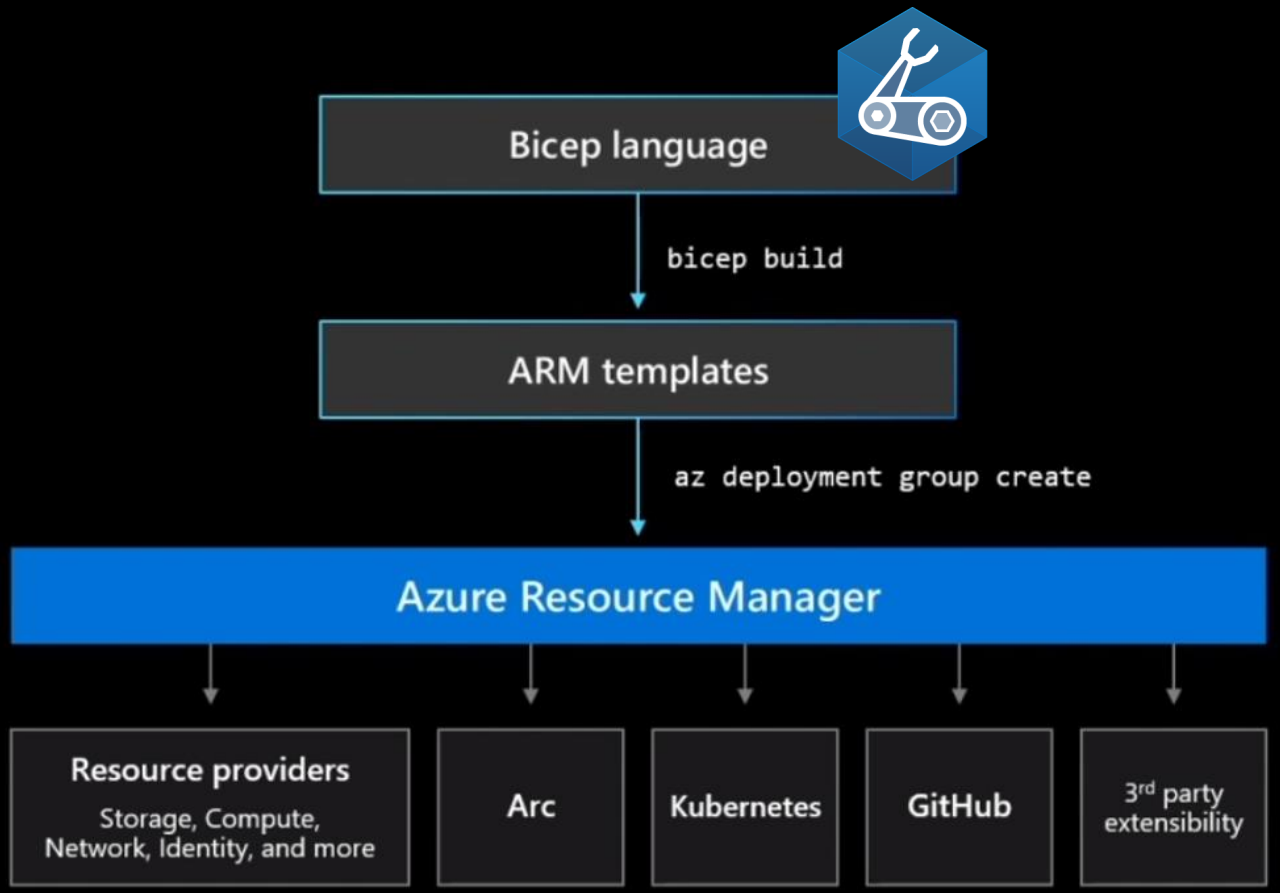
Simple declarative language to provision infrastructure to Azure.

Intuitive
Easy to read and to author

Transpiles to ARM Templates
Leverage ARM template knowledge and investments

Modular
Abstract common blocks of code into reusable parts

Open Source
Transparency and community



How to get started with ‘Bicep’?

1. Install the Bicep CLI (required)

```
# Create the install folder
$installPath = "$env:USERPROFILE\.bicep"
$installDir = New-Item -ItemType Directory -Path $installPath -Force
$installDir.Attributes += 'Hidden'
# Fetch the latest Bicep CLI binary
(New-Object Net.WebClient).DownloadFile("https://github.com/Azure/bicep/releases/latest/download/bicep-win-x64.exe", "$installPath\bicep.exe")
# Add bicep to your PATH
$currentPath = (Get-Item -path "HKCU:\Environment" ).GetValue('Path', '', 'DoNotExpandEnvironmentNames')
if (-not $currentPath.Contains("%USERPROFILE%\.bicep")) { setx PATH ($currentPath + ";%USERPROFILE%\.bicep") }
if (-not $env:path.Contains($installPath)) { $env:path += ";$installPath" }
# Verify you can now access the 'bicep' command.
bicep --help
# Done!
```

2. Install the Bicep VS Code extension(optional)

```
# Fetch the latest Bicep VSCode extension
$vsixPath = "$env:TEMP\vscode-bicep.vsix"
(New-Object Net.WebClient).DownloadFile("https://github.com/Azure/bicep/releases/latest/download/vscode-bicep.vsix", $vsixPath)
# Install the extension
code --install-extension $vsixPath
# Clean up the file
Remove-Item $vsixPath
# Done!
```

Demo

Deploy local 'Bicep' files

Note: Currently, both Azure CLI and Azure PowerShell can only deploy local Bicep files.
Bicep CLI is needed locally to compile Bicep files to JSON templates before deployment.

Azure CLI v2.20.0+

Azure CLI

```
az deployment group create \
  --name ExampleDeployment \
  --resource-group ExampleGroup \
  --template-file <path-to-template-or-bicep> \
  --parameters storageAccountType=Standard_GRS
```

Note: with Azure CLI v2.20.0+ installed, the Bicep CLI is automatically installed when a command that depends on it is executed.

PowerShell 5.6.0+

Azure PowerShell

```
New-AzResourceGroupDeployment `
  -Name ExampleDeployment `
  -ResourceGroupName ExampleGroup `
  -TemplateFile <path-to-template-or-bicep> `
  -storageAccountType Standard_GRS
```

Note: Azure PowerShell does not have the capability to install the Bicep CLI yet. Azure PowerShell (v5.6.0+) expects that the Bicep CLI is already installed and available on the PATH.

Demo – Putting Bicep into (GitHub) Action

cognitionIT/AzureWVD: Collectic | Resource groups - Microsoft Azu

https://github.com/cognitionIT/AzureWVD

Search or jump to... / Pull requests Issues Marketplace Explore

cognitionIT / AzureWVD

Unwatch 3 | Star 8 | Fork 0

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights | Settings

master | 2 branches | 0 tags | Go to file | Add file | Code

cognitionIT latest version 354b5b9 23 minutes ago 80 commits

.github/workflows	latest version	23 minutes ago
AIBCustomizations	added vscode	19 days ago
ARMTemplates	Adding ARMTemplates	4 months ago
AzModule	Demo Ready!	4 months ago
Bicep	testing bicep	7 days ago
InlineScripts	latest version	23 minutes ago
RESTAPI	RESTAPI Update	6 months ago
.gitignore	Demo ready!!!	4 months ago
LICENSE	Initial commit	7 months ago
README.md	Update README.md	4 months ago

About

Collection of scripts, templates and other resources to support automation of the Windows Virtual Desktop (WVD) Service provided by Microsoft Azure

Readme

MIT License

Releases

No releases published

Create a new release

Packages

Tip: run Bicep on windows-latest agent

```yaml
# Action = Azure PowerShell: Run inline script
# source: https://github.com/marketplace/actions/azure-powershell-action
- name: Install the min. version Az Module using Azure PowerShell
  uses: azure/powershell@v1
  with:
    inlineScript: |
      ## Add Az PowerShell Module version 5.6.0 to the runner (if not already on the runner)
      $minAzModuleVersion = '5.6.0'
      if(!(Test-Path "C:\Modules\az_$minAzModuleVersion")) {
        Install-Module -Name Az -AllowClobber -Scope CurrentUser -Force
        Save-Module -Path "C:\Modules\az_$minAzModuleVersion" -Name Az -RequiredVersion $minAzModuleVersion -Force
      }
      $env:PSModulePath = "C:\Modules\az_$($minAzModuleVersion);$($env:PSModulePath)"
      # Check installed versions of Az Module
      Get-InstalledModule -Name Az -AllVersions | sort Version -Descending
    azPSVersion: 'latest'
```

ARM deployment with bicep ARM deployment with bicep #15

Summary

Jobs

ARM Deployment with bicep file

ARM Deployment with bicep file

succeeded 21 days ago in 6m 54s

Search logs

- Set up job 24s
- Checkout 14s
- Login via Az module 1m 54s
- Prep runner with Bicep prereqs using Azure PowerShell 3m 56s

```
1   ▶ Run azure/powershell@v1
41  Validating inputs
42  Initializing Az Module
55  ***
56    "Success": "true",
57    "AzVersion": "5.5.0"
58  ***
97  Version              Name                                Repository           Description
98  -------              ----                                ----------           -----------
99  5.6.0                Az                                  PSGallery            Microsoft Azure PowerShell - Cmdlets to …
100
101 Script execution Complete
```

- Deploy ARM Resources with Bicep & Azure PowerShell 23s
- Post Checkout 3s
- Complete job 0s

The curious case of the templateHash

Bicep-generated files should include an autogenerated header #800

Closed **snarkywolverine** opened this issue on Nov 3, 2020 · 2 comments

Discussed it at the team meeting today. The consensus appears to have template code generators use the top-level `metadata` property to store this information. This is the proposed schema:

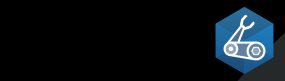
```
"metadata": {
  "_generator": {
    "name": "<name of the code generator>",
    "version": "<version of the code generator>",
    "templateHash": "<template hash>"
  }
}
```

Considerations:

- Discussed using a comment instead of a JSON property. We're not in favor of using meaningful comments due to their fragility and uneven support in JSON libraries across all the relevant platforms.
- Template hash logic should reuse the existing template hash calculation logic that we already have in ARM telemetry and exposed in the API at https://github.com/Azure/azure-rest-api-specs/blob/8cef8014762a839e98f0aeaa57a0bbdb8982d3d4/specification/resources/resource-manager/Microsoft.Resources/stable/2020-10-01/resources.json#L4236
- Template hash calculation should run on the entire content of the template except for the `metadata._generator.templateHash` property. This is technically a breaking change in ARM, but impact should be extremely low.
- Also discussed adding a top-level multi-line comment with text similar to "This file is generated. Do not modify." This should be deferred until we fix bugs in line number handling in the runtime.

Road map

Current release: CLI version 0.3.255



v0.1

(aug ‘20)

Alpha Release available on August 31st

v0.2

(Oct ‘20)

- VSCode Intellisense
- Support for modules

v0.3

(March ‘21)

- Loops
- Conditionals
- Decompiler
- Production usage

v0.4

- Quality release
- Learn module Available
- Linter (TTK successor)
- Snippets & resource scaffolding Available*
- Merging ARM Quickstarts & bicep
- IncludeFile() support

v0.5++

- Module Registry

And then…

Announcing the preview of PSArm

Steve

March 31st, 2021

Announcing PSArm preview

Today, we are pleased to announce the first preview of a new experimental module that make it easier than ever for PowerShell customers to create Azure Resource Manager (ARM) templates: PSArm.

This module enables users to author ARM templates using PowerShell. Similar to Azure Bicep, PSArm is an independent module that creates the necessary ARM JSON template to deploy and configure Azure infrastructure in a PowerShell context. PSArm allows PowerShell users who are familiar with ARM to write complex deployment templates by mixing the declarative syntax of ARM with the iterative syntax of PowerShell.

PSArm

wvd-backplane.psarm.ps1

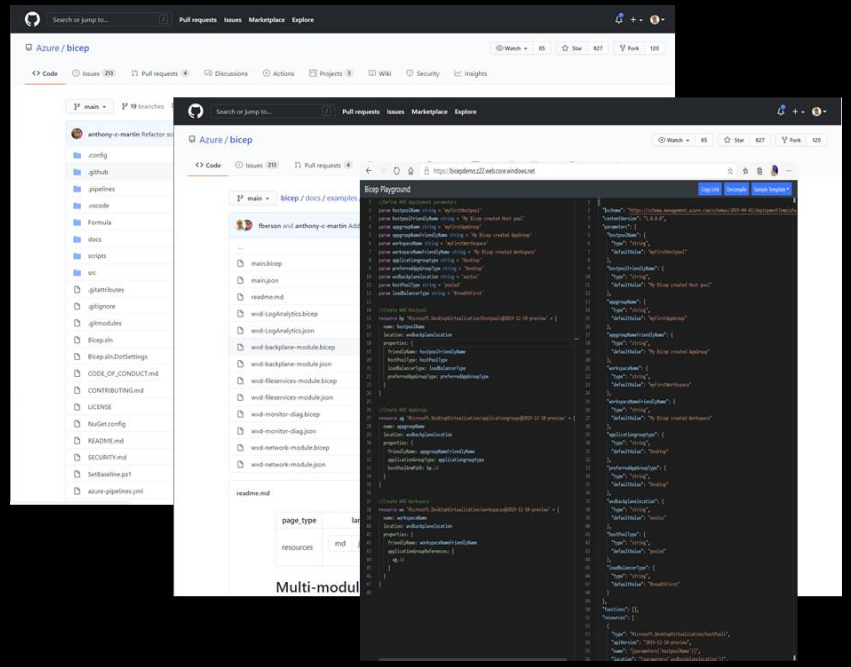
PSArm > wvd-backplane.psarm.ps1

```
Arm {
  param(

    [ArmParameter[string]]
    $hostpoolFriendlyName = 'My Bicep created Host pool',

    [ArmParameter[string]]
    $appgroupNameFriendlyName = 'My Bicep created AppGroup',

    [ArmParameter[string]]
    $workspaceNameFriendlyName = 'My Bicep created Workspace',

    [ValidateSet('Desktop', 'RemoteApp')]
    [ArmParameter[string]]
    $applicationgrouptype = 'Desktop',

    [ValidateSet('Desktop', 'RailApplications')]
    [ArmParameter[string]]
    $preferredAppGroupType = 'Desktop',

    [ArmParameter[string]]
    $wvdbackplanelocation = 'eastus',

    [ArmParameter[string]]
    $hostPoolType = 'pooled',

    [ArmParameter[string]]
    $loadBalancerType = 'BreadthFirst'
  )

  Resource $hostpoolName -Namespace 'Microsoft.DesktopVirtualization' -Type 'hostPools' -ApiVersion '2019-12-10-preview' -Location $
    properties {
      friendlyName $hostpoolFriendlyName
      hostPoolType $hostPoolType
      loadBalancerType $loadBalancerType
      preferredAppGroupType $preferredAppGroupType
    }
  }
```

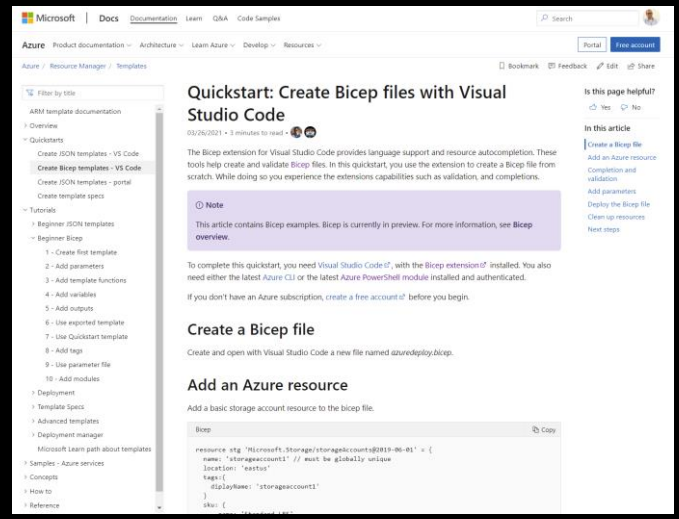
Call to actions:

Install guides, tutorials, example code & playgrounds!

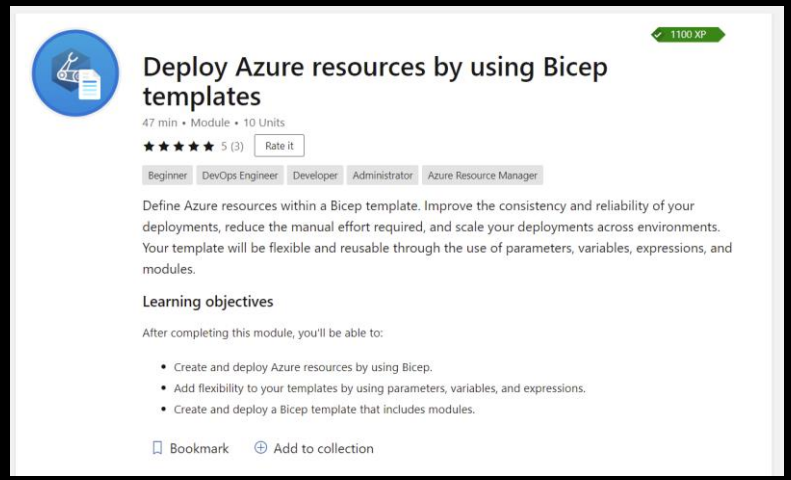
aka.ms/bicep



bit.ly/3ml2FnJ



bit.ly/3wIIuVA



Esther Barthel

@virtuEs_IT

github.com/cognitionit

Freek Berson

@fberson

github.com/fberson