

Test

Infrastructure As Code

<https://aka.ms/mark/tests-iac>

What is IaC?

Infrastructure As Code

<https://aka.ms/mark/ca101>

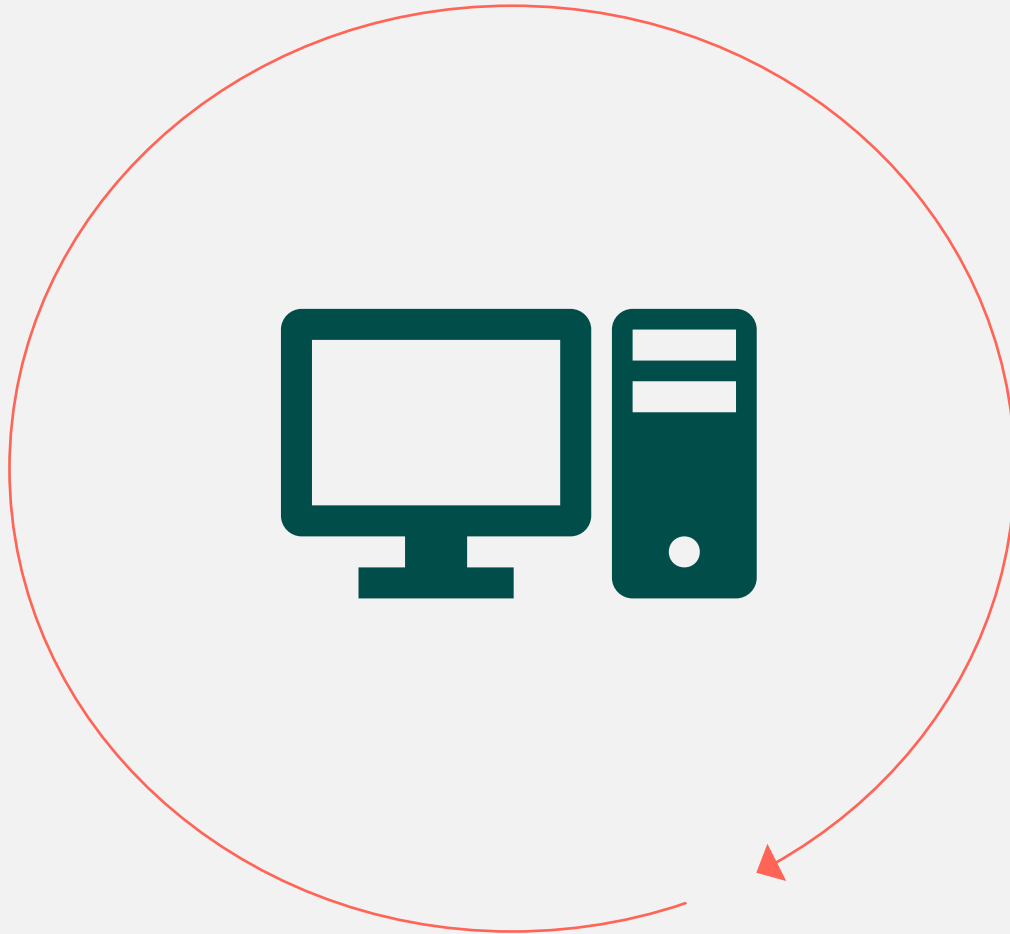
```
1 // azuredeploy.json
2 "comments": "Azure Data Lake Gen 2 Storage Account",
3 "type": "Microsoft.Storage/storageAccounts",
4 "apiVersion": "2019-04-01",
5 "name": "[parameters('resourceName')]",
6 "sku": {
7     "name": "[parameters('storageAccountSku')]"
8 },
9 "kind": "StorageV2",
10 "location": "[parameters('location')]",
11 "tags": {},
12 "identity": { "type": "SystemAssigned" },
13 "properties": {
14     "encryption": {
15         "services": {
16             "blob": { "enabled": true },
17             "file": { "enabled": true }
18         },
19     },
20     "keySource": "Microsoft.Storage"
21 },
22 "isHnsEnabled": true,
23 "networkAcls": "[json(parameters('networkAcls'))]",
24 "accessTier": "[parameters('storageAccountAccessTier')]",
25 "supportsHttpsTrafficOnly": true
26
```



Application Development



Infrastructure Development



Outside

Hardware Configuration

- VM Size, Disks, Network
- RBAC, secrets etc.
- Resource Settings

Inside

Software

- Application Code
- Desired state
- Configurations & scripts



declarative

describe final state



imperative

executing steps to get to final state

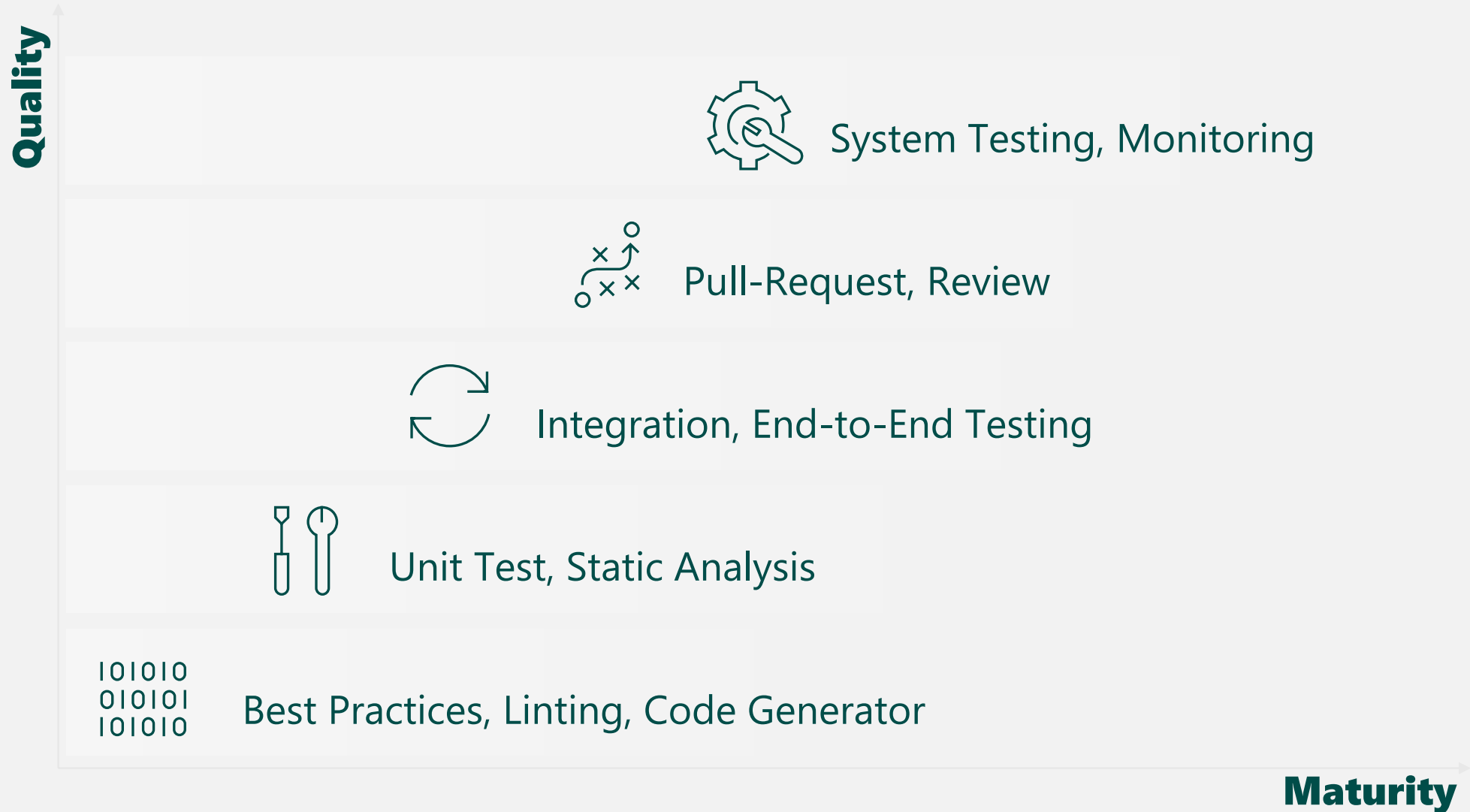
Idempotency

Idempotence is the property where a deployment command always sets the target environment into the same configuration, regardless of the environment's starting state.

configuration
(YAML, JSON)

no behavior
but state

validation of
requirements



```
1 // azuredeploy.json
2 "comments": "Azure Data Lake Gen 2 Storage Account",
3 "type": "Microsoft.Storage/storageAccounts",
4 "apiVersion": "2019-04-01",
5 "name": "[parameters('resourceName')]",
6 "sku": {
7     "name": "[parameters('storageAccountSku')]"
8 },
9 "kind": "StorageV2",
10 "location": "[parameters('location')]",
11 "tags": {},
12 "identity": { "type": "SystemAssigned" },
13 "properties": {
14     "encryption": {
15         "services": {
16             "blob": { "enabled": true },
17             "file": { "enabled": true }
18         },
19     },
20     "keySource": "Microsoft.Storage"
21 },
22 "isHnsEnabled": true,
23 "networkAcls": "[json(parameters('networkAcls'))]",
24 "accessTier": "[parameters('storageAccountAccessTier')]",
25 "supportsHttpsTrafficOnly": true
26
```

Unit Test

Azure Resource Manager Templates

```
1 # azuredeploy.adls.spec.ps1
2
3 param (
4     $Path = (Join-Path $PSScriptRoot "azuredeploy.json")
5 )
6
7 # Test for template
8 $null = Test-Path $Path -ErrorAction Stop
9
10 # Test if template content is readable
11 $text = Get-Content $Path -Raw -ErrorAction Stop
12
13 # Convert the template to object
14 $json = ConvertFrom-Json $text -ErrorAction Stop
15
16 # Query for type that match 'storageAccounts'
17 $resource = $json.resources
18     | Where-Object -Property "type" -eq "Microsoft.Storage/storageAccounts"
19
20 ...
21
22
23
24
25
26
```

```
1 # azuredeploy.adls.spec.ps1
2
3 ...
4 Describe "Azure Data Lake Generation 2 Resource Manager Template Unit" -Tag Unit {
5
6     # Mandatory requirement of ADLS Gen 2 are:
7     # - Resource Type is Microsoft.Storage/storageAccounts
8     # - Kind is StorageV2
9     # - Hierarchical namespace is enabled
10
11     it "should have resource properties present" {
12         $resource | Should -Not -BeNullOrEmpty
13     }
14
15     it "should be of type Microsoft.Storage/storageAccounts" {
16         $resource.type | Should -Be "Microsoft.Storage/storageAccounts"
17     }
18
19     it "should be of kind StorageV2" {
20         $resource.kind | Should -Be "StorageV2"
21     }
22
23     it "should have Hns enabled" {
24         $resource.properties.isHnsEnabled | Should -Be $true
25     }
26     ...
```

```
1 # azuredeploy.adls.spec.ps1
```

```
2  
3 ...
```

```
4  
5 # Optional validation tests:
```

```
6 # - Ensure encryption is as specified
```

```
7 # - Secure Transfer by enforcing HTTPS
```

```
8  
9 it "should have encryption key source set to Storage " {  
10     $resource.properties.encryption.keySource | Should -Be "Microsoft.Storage"  
11 }
```

```
12  
13 it "should have blob encryption enabled" {  
14     $resource.properties.encryption.services.blob.enabled | Should -Be $true  
15 }
```

```
16  
17 it "should have file encryption enabled" {  
18     $resource.properties.encryption.services.file.enabled | Should -Be $true  
19 }
```

```
20  
21 it "should enforce Https Traffic Only" {  
22     $resource.properties.supportsHttpsTrafficOnly | Should -Be $true  
23 }
```

```
24 }  
25  
26
```

Unit Test

PowerShell Deployment Scripts

```
1 # deploy.ps1 -WhatIf
2
3 [CmdletBinding(SupportsShouldProcess = $True)]
4
5 $Deployment = @{
6     ResourceGroupName      = $rg
7     TemplateFile           = $tf
8     TemplateParameterFile = $tpf
9 }
10
11 if ($PSCmdlet.ShouldProcess("ResourceGroupName $rg deployment of", "TemplateFile $tf")) {
12     # Code that runs the actual deployment
13     New-AzResourceGroupDeployment @Deployment
14 }
15 else {
16     # Code that dry runs the deployment
17     New-AzResourceGroupDeployment @Deployment -WhatIf
18     # Code that ,mocks' the deployment
19     Test-AzResourceGroupDeployment @Deployment
20 }
21
22
23
24
25
26
```


Acceptance Test

Azure Resources

```
1 # adls.acceptance.spec.ps1
2
3 param (
4     # Name of the resource
5     [Parameter(Mandatory)]
6     [string]
7     $Name,
8
9     # Name of the resource group
10    [Parameter()]
11    [string]
12    $ResourceGroupName
13 )
14 $adls = Get-AzStorageAccount -Name $resource.Name -ResourceGroupName $resource.ResourceGroupName
```

```
15
16 ...
17
18
19
20
21
22
23
24
25
26
```

```
1 # adls.acceptance.spec.ps1
2
3 ...
4
5 Describe "$Name Data Lake Storage Account Generation 2" {
6
7     # Mandatory requirement of ADLS Gen 2 are:
8     # - Resource Type is Microsoft.Storage/storageAccounts,
9     #   as we know we are looking for this it is obsolete to check
10    # - Kind is StorageV2
11    # - Hierarchical namespace is enabled
12
13    it "should be of kind StorageV2" {
14        $adls.Kind | Should -Be "StorageV2"
15    }
16
17    it "should have Hierarchical Namespace Enabled" {
18        $adls.EnableHierarchicalNamespace | Should -Be $true
19    }
20 ...
21
22
23
24
25
26
```

```
1 # adls.acceptance.spec.ps1
```

```
2 ...
```

```
3 <#
```

```
4 Optional validation tests:
```

- 5 - Ensure encryption is as specified
- 6 - Secure Transfer by enforcing HTTPS

```
7 #>
```

```
8  
9  
10 it "should enforce https traffic" {  
11     $adls.EnableHttpsTrafficOnly | Should -Be $true  
12 }  
13
```

```
14 it "should have encryption enabled" {  
15     $adls.Encryption.Services.Blob.Enabled | Should -Be $true  
16     $adls.Encryption.Services.File.Enabled | Should -Be $true  
17 }  
18
```

```
19 it "should have network rule set default action Deny" {  
20     $adls.NetworkRuleSet.DefaultAction | Should -Be "Deny"  
21 }  
22 }
```

```
23  
24  
25  
26
```

Integration Test

Azure Resource Manager deployment

```
1 # integration.Tests.ps1
2
3 Describe "Azure Data Lake Generation 2 Resource Manager Integration" -Tags Integration {
4
5     BeforeAll {
6         # Create test environment
7         Write-Host "Creating test environment $ResourceGroupName, cleanup..."
8
9         # Create a unique ResourceGroup
10        # 'unique' string base on the date
11        # e.g. 20190824T1830434620Z
12        # file date time universal format ~ 20 characters
13        $ResourceGroupName = 'TT-' + (Get-Date -Format FileDateTimeUniversal)
14
15        Get-AzResourceGroup -Name $ResourceGroupName -ErrorAction SilentlyContinue |
16        Remove-AzResourceGroup -Force
17
18        # Get a unique name for the resource too,
19        # Some Azure Resources have a limitation of 24 characters
20        # consider 20 for the unique ResourceGroup.
21        $ResourceName = 'pre-' + $ResourceGroupName.ToLower()
22
23        # Setup the environment
24        $null = New-AzResourceGroup -Name $ResourceGroupName -Location 'WestEurope'
25    }
26    ...
```

```
1 # integration.Tests.ps1
```

```
2 ...
```

```
3  
4  
5 AfterAll {
```

```
6     # Remove test environment after test
```

```
7     Write-Host "Removing test environment $ResourceGroupName..."
```

```
8  
9     Get-AzResourceGroup -Name $ResourceGroupName |
```

```
10         Remove-AzResourceGroup -Force -AsJob
```

```
11     }
```

```
12  
13 # Deploy Resource
```

```
14 New-AzResourceGroupDeployment -ResourceName $ResourceName `
```

```
15     -ResourceGroupName $ResourceGroupName
```

```
16  
17  
18 # Run Acceptance Test
```

```
19 . $PSScriptRoot/acceptance.spec.ps1 -ResourceName $ResourceName `
```

```
20     -ResourceGroupName $ResourceGroupName
```

```
21 }
```

```
22
```

```
23
```

```
24
```

```
25
```

```
26
```

Test Dashboard

Azure DevOps Test

xAz.New

Overview

Boards

Repos

Pipelines

Builds

Releases

Artifacts

#20190212.9: fix interactive prompt

All logs

Triggered 12 feb at 18:15 for Mark Warneke xAz.KV master c9d0d57 Retained by release

Logs Summary Tests

Summary

3 Run(s) Completed (3 Passed, 0 Failed)

1,405

Total tests
+1,405



1,405 Passed
0 Failed
0 Others

100%

Pass percentage
↑ 100%

45s 63ms

Run duration
↑ +45s 63ms

0

Tests not reported

Test run Column Options

Filter by test or run name

Tags Test file Owner Outcome

Test	Duration	Failing since	Failing build	Tags
> ✓ PS_Win2016_Module (1391/1391)	0:00:45.050			
> ✓ PS_Win2016_Unit (11/11)	0:00:04.067			
> ✓ PS_Win2016_Integration (3/3)	0:00:00.013			

```
1 # azure-pipelines.yml
```

```
2
```

```
3
```

```
4 steps:
```

```
5 - task: AzurePowerShell@4
```

```
6   inputs:
```

```
7     azureSubscription: $(azureSubscription)
```

```
8     scriptType: "FilePath"
```

```
9     # The name of the script where the pester test setup is located
```

```
10    scriptPath: $(Build.SourcesDirectory)\Invoke-Pester.ps1
```

```
11    scriptArguments: -OutputFormat 'NUnitXml' `
```

```
12                    -OutputFile 'TestResults.Pester.xml' -PassThru'
```

```
13    azurePowerShellVersion: "latestVersion"
```

```
14    errorActionPreference: "continue"
```

```
15 - task: PublishTestResults@2
```

```
16   inputs:
```

```
17     # Make sure to use the 'NUnit' test runner
```

```
18     testRunner: "NUnit" # !!!
```

```
19     testResultsFiles: "**/TestResults.Pester.xml"
```

```
20     testRunTitle: "PS_Win2016_Unit"
```

```
21     # Make the whole pipeline fail if a test is failed
```

```
22     failTaskOnFailedTests: true
```

```
23   displayName: "Publish Unit Test Results"
```

```
24   condition: in(variables['Agent.JobStatus'], 'Succeeded', 'SucceededWithIssues', 'Failed')
```

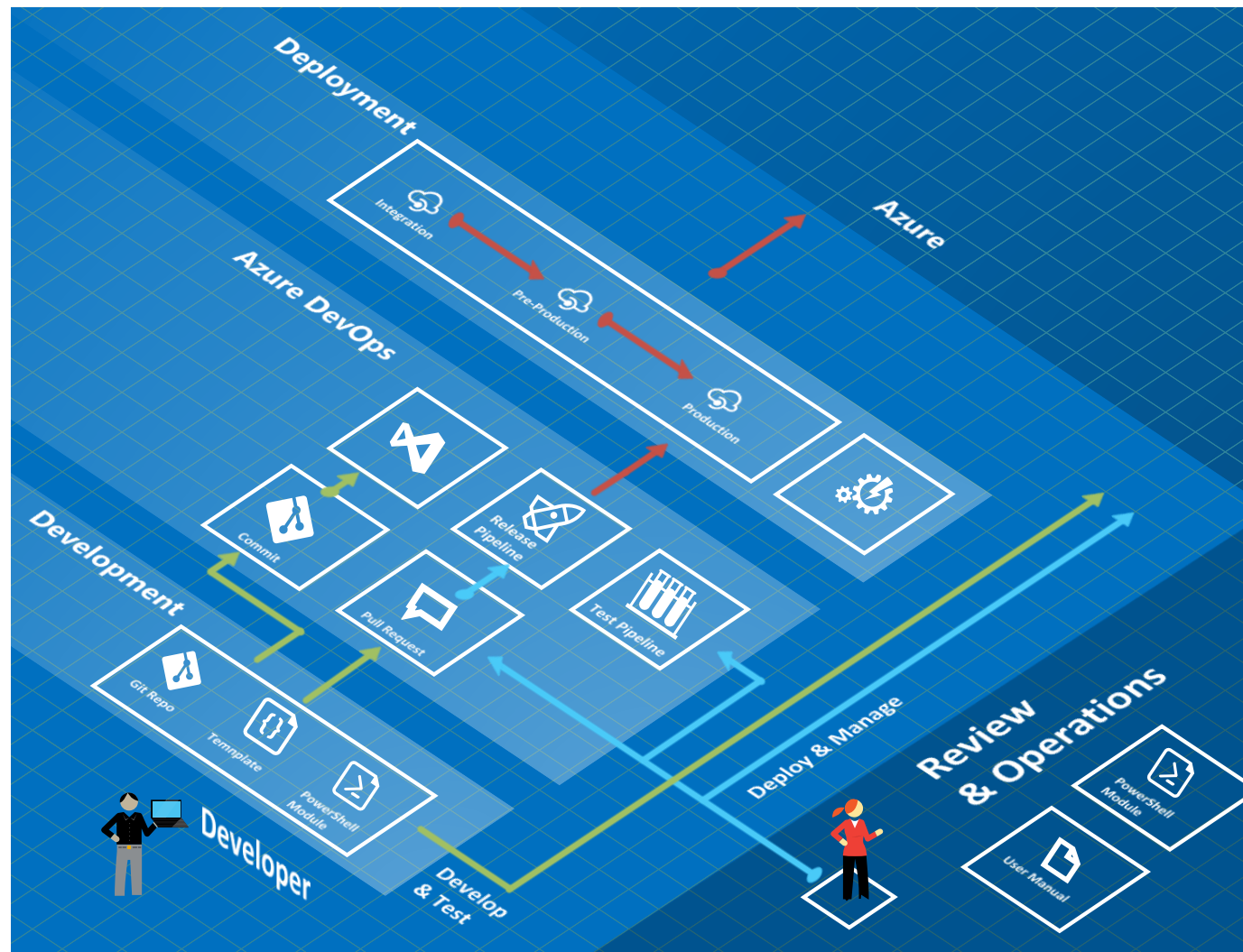
```
25
```

```
26
```

Peer Review

four eyes principle

Pull-Request & Validation



Add build policy

Build pipeline *

Path filter (optional) [i](#)

Trigger

Automatic (whenever the source branch is updated)

Manual


Policy requirement

Required
Build must succeed in order to complete pull requests.

Optional
Build failure will not block completion of pull requests.

Build expiration

Immediately when  bug/315423_adls_hdi_clusters_folder is updated

After hours if  bug/315423_adls_hdi_clusters_folder has been

Never

Display name

Save

Cancel

Build Policy Validation

- Add build policy for all tests

Build validation

Validate code by pre-merging and building pull request changes

+ Add build policy

Policies

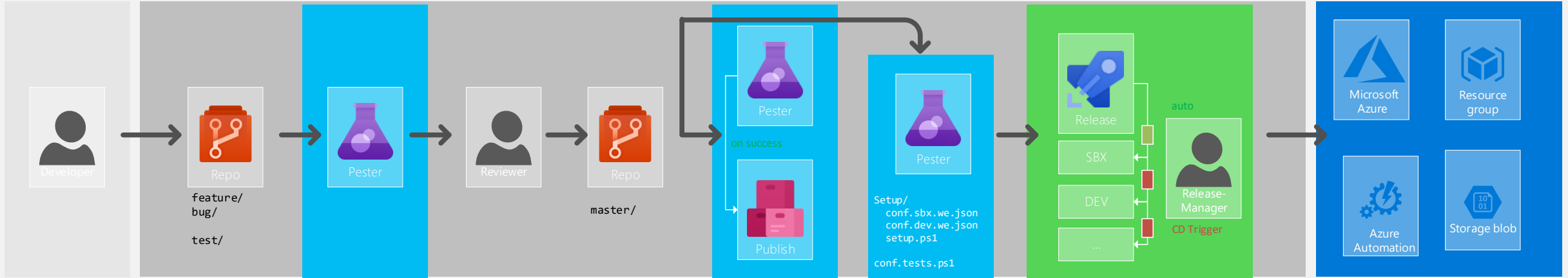
Required

✓ 1 reviewer approved

✗ Required reviewers have not approved

✓ [Build succeeded](#)

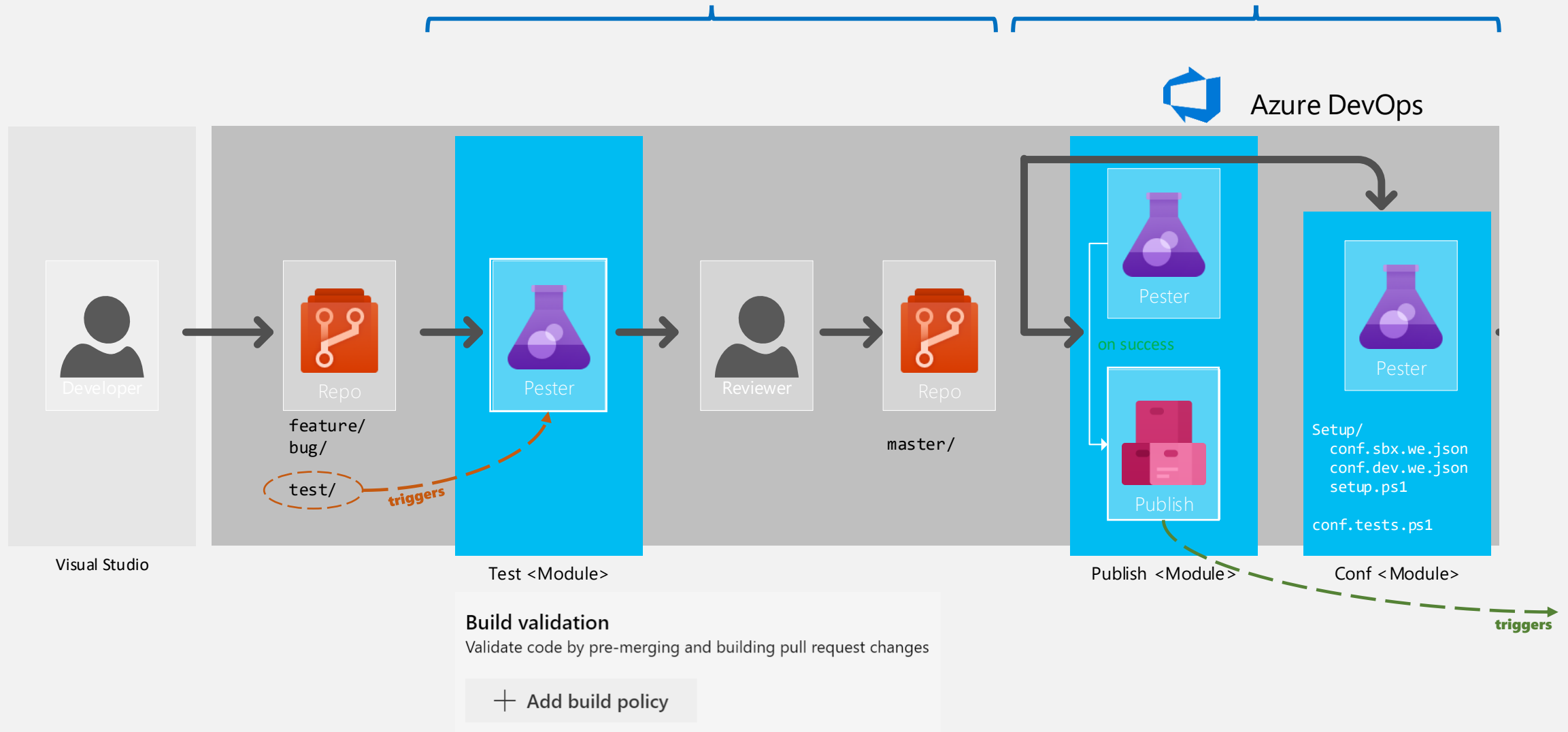




<https://aka.ms/az.new>

PULL REQUEST

PUBLISH PIPELINE



Wrap Up

<https://aka.ms/mark/test-iac>

