Test

Infrastructure As Code

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

What is IaC?

Infrastructure As Code

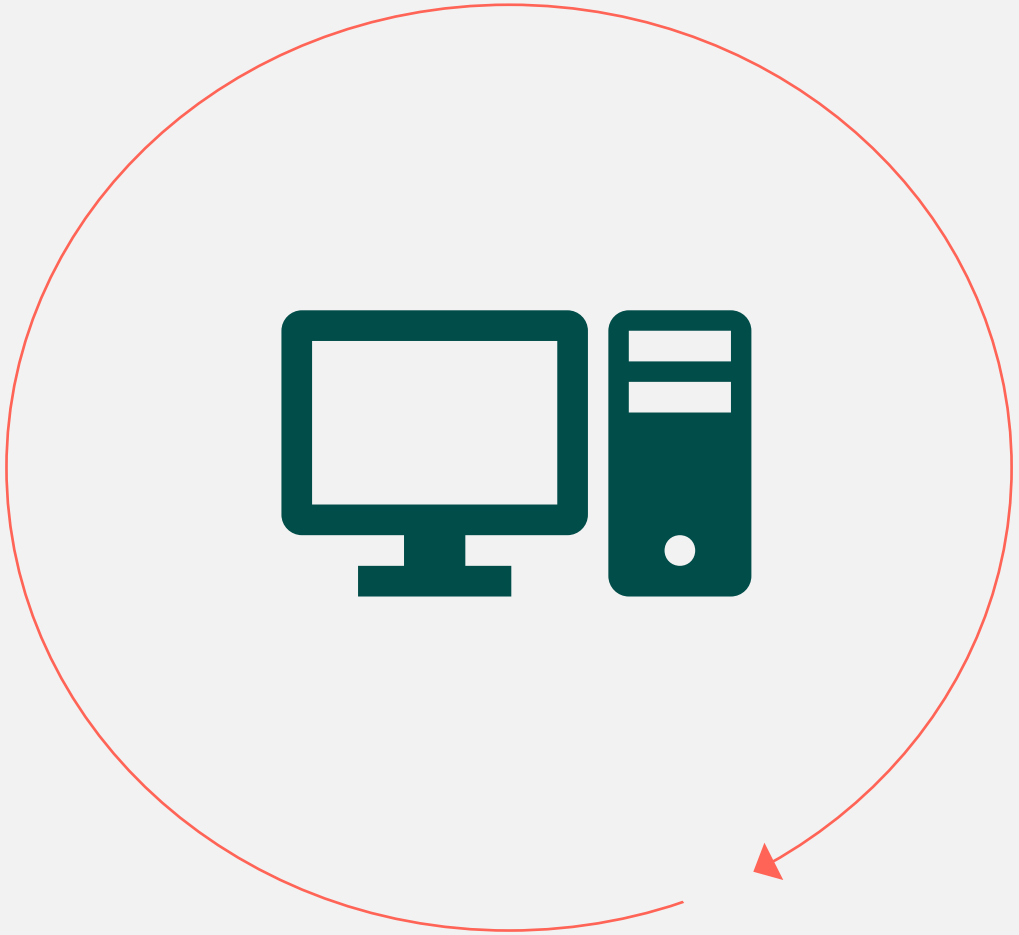
https://aka.ms/mark/ca101

@MarkWarneke

```
// azuredeploy.json
"comments": "Azure Data Lake Gen 2 Storage Account",
"type": "Microsoft.Storage/storageAccounts",
"apiVersion": "2019-04-01",
"name": "[parameters('resourceName')]",
"sku": {
      "name": "[parameters('storageAccountSku')]"
},
"kind": "StorageV2",
"location": "[parameters('location')]",
"tags": {},
"identity": { "type": "SystemAssigned" },
"properties": {
      "encryption": {
          "services": {
              "blob": { "enabled": true },
              "file": { "enabled": true }
          },
          "keySource": "Microsoft.Storage"
      },
      "isHnsEnabled": true,
      "networkAcls": "[json(parameters('networkAcls'))]",
      "accessTier": "[parameters('storageAccountAccessTier')]",
      "supportsHttpsTrafficOnly": true
```

Application Development

Infrastructure Development



Outside

Hardware Configuration

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

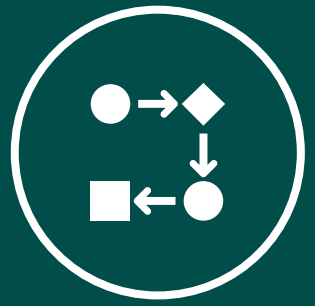
Inside

Software

- Application Code
- Desired state
- Configurations & scripts

declarative

describe final state



imperative

executing steps to get to final state

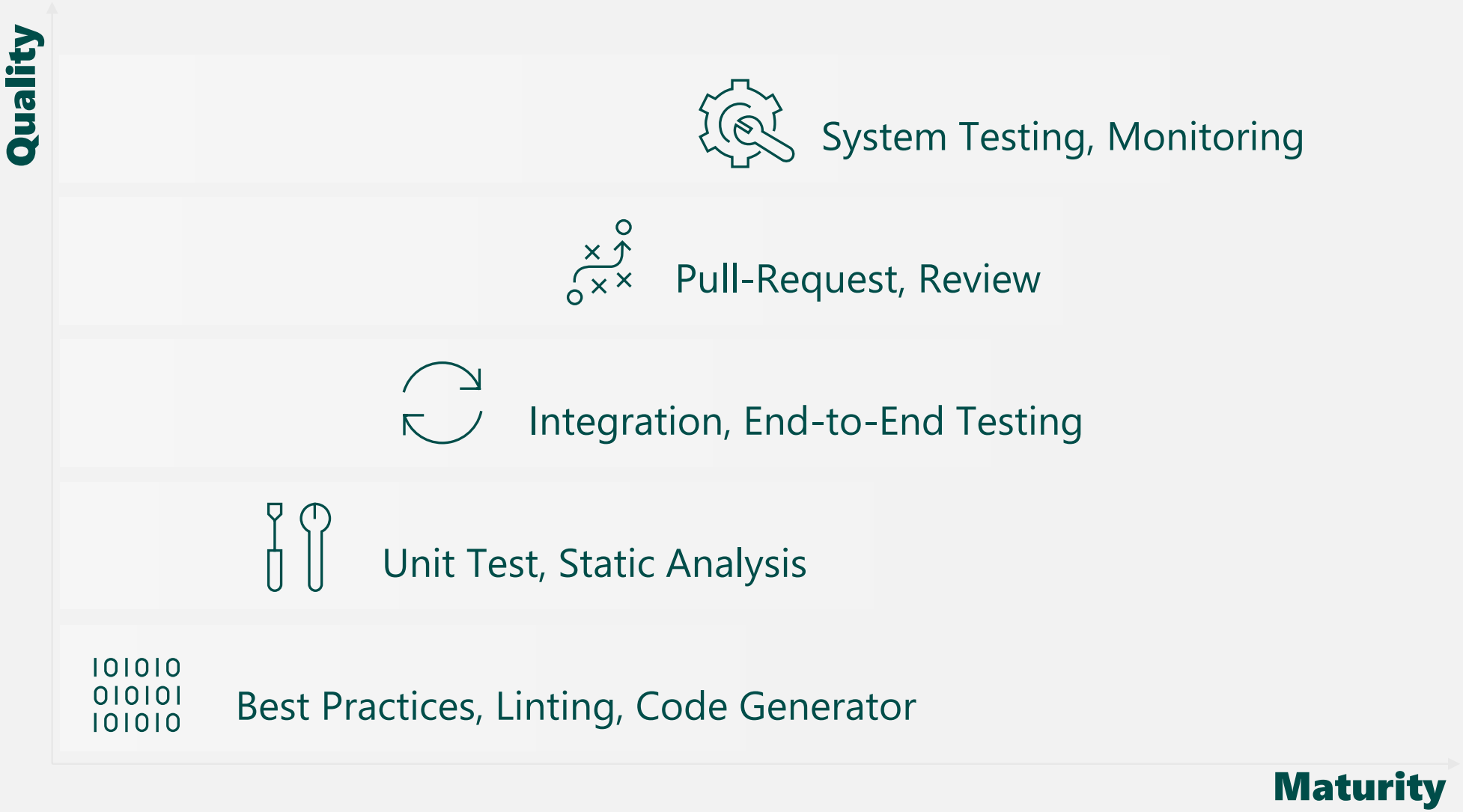
Idempotency

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


Quality
System Testing, Monitoring
Pull-Request, Review
Integration, End-to-End Testing
Unit Test, Static Analysis
Best Practices, Linting, Code Generator
Maturity

@MarkWarneke

```
// azuredeploy.json
"comments": "Azure Data Lake Gen 2 Storage Account",
"type": "Microsoft.Storage/storageAccounts",
"apiVersion": "2019-04-01",
"name": "[parameters('resourceName')]",
"sku": {
      "name": "[parameters('storageAccountSku')]"
},
"kind": "StorageV2",
"location": "[parameters('location')]",
"tags": {},
"identity": { "type": "SystemAssigned" },
"properties": {
      "encryption": {
          "services": {
              "blob": { "enabled": true },
              "file": { "enabled": true }
          },
          "keySource": "Microsoft.Storage"
      },
      "isHnsEnabled": true,
      "networkAcls": "[json(parameters('networkAcls'))]",
      "accessTier": "[parameters('storageAccountAccessTier')]",
      "supportsHttpsTrafficOnly": true
```

Unit Test

Azure Resource Manager Templates

@MarkWarneke

```
# azuredeploy.adls.spec.ps1

param (
    $Path = (Join-Path $PSScriptRoot "azuredeploy.json")
)

# Test for template
$null = Test-Path $Path -ErrorAction Stop

# Test if template content is readable
$text = Get-Content $Path -Raw -ErrorAction Stop

# Convert the template to object
$json = ConvertFrom-Json $text -ErrorAction Stop

# Query for type that match ’storageAccounts’
$resource = $json.resources
        | Where-Object -Property "type" -eq "Microsoft.Storage/storageAccounts"

...
```

@MarkWarneke

```
# azuredeploy.adls.spec.ps1

...
Describe "Azure Data Lake Generation 2 Resource Manager Template Unit" -Tag Unit {

    # Mandatory requirement of ADLS Gen 2 are:
    # - Resource Type is Microsoft.Storage/storageAccounts
    # - Kind is StorageV2
    # - Hierarchical namespace is enabled

    it "should have resource properties  present" {
        $resource | Should -Not -BeNullOrEmpty
    }

    it "should be of type Microsoft.Storage/storageAccounts" {
        $resource.type | Should -Be "Microsoft.Storage/storageAccounts"
    }

    it "should be of kind StorageV2" {
        $resource.kind | Should -Be "StorageV2"
    }

    it "should have Hns enabled" {
        $resource.properties.isHnsEnabled | Should -Be $true
    }
...
```

@MarkWarneke

```
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

@MarkWarneke

```
# deploy.ps1 -WhatIf

[CmdletBinding(SupportsShouldProcess = $True)]

$Deployment = @{
    ResourceGroupName     = $rg
    TemplateFile          = $tf
    TemplateParameterFile = $tpf
}

if ($PSCmdlet.ShouldProcess("ResourceGroupName $rg deployment of", "TemplateFile $tf")) {
    # Code that runs the actual deployment
    New-AzResourceGroupDeployment @Deployment
}
else {
    # Code that dry runs the deployment
    New-AzResourceGroupDeployment @Deployment -WhatIf
    # Code that ‚mocks‘ the deployment
    Test-AzResourceGroupDeployment @Deployment
}
```

Acceptance Test

Azure Resources

@MarkWarneke

```
# adls.acceptance.spec.ps1

param (
    # Name of the resource
    [Parameter(Mandatory)]
    [string]
    $Name,

    # Name of the resource group
    [Parameter()]
    [string]
    $ResourceGroupName
)

$adls = Get-AzStorageAccount -Name $resource.Name -ResourceGroupName $resource.ResourceGroupName

…
```

@MarkWarneke

```
# adls.acceptance.spec.ps1

...

Describe "$Name Data Lake Storage Account Generation 2" {

    # Mandatory requirement of ADLS Gen 2 are:
    # - Resource Type is Microsoft.Storage/storageAccounts,
    #   as we know we are looking for this it is obsolete to check
    # - Kind is StorageV2
    # - Hierarchical namespace is enabled

    it "should be of kind StorageV2" {
        $adls.Kind | Should -Be "StorageV2"
    }

    it "should have Hierarchical Namespace Enabled" {
        $adls.EnableHierarchicalNamespace | Should -Be $true
    }

...
```

@MarkWarneke

```
# adls.acceptance.spec.ps1

...

    <#
      Optional validation tests:
       - Ensure encryption is as specified
       - Secure Transfer by enforcing HTTPS
    #>

    it "should enforce https traffic" {
        $adls.EnableHttpsTrafficOnly | Should -Be $true
    }

    it "should have encryption enabled" {
        $adls.Encryption.Services.Blob.Enabled | Should -Be $true
        $adls.Encryption.Services.File.Enabled | Should -Be $true
    }

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

Integration Test

Azure Resource Manager deployment

@MarkWarneke

```
# integration.Tests.ps1

Describe "Azure Data Lake Generation 2 Resource Manager Integration" -Tags Integration {

    BeforeAll {
        # Create test environment
        Write-Host "Creating test environment $ResourceGroupName, cleanup..."

        # Create a unique ResourceGroup
        # ‘unique‘ string base on the date
        # e.g. 20190824T1830434620Z
        # file date time universal format ~ 20 characters
        $ResourceGroupName = 'TT-' + (Get-Date -Format FileDateTimeUniversal)

        Get-AzResourceGroup -Name $ResourceGroupName -ErrorAction SilentlyContinue |
                Remove-AzResourceGroup -Force

        # Get a unique name for the resource too,
        # Some Azure Resources have a limitation of 24 characters
        # consider 20 for the unique ResouceGroup.
        $ResourceName = 'pre-' + $ResourceGroupName.ToLower()

        # Setup the environment
        $null = New-AzResourceGroup -Name $ResourceGroupName -Location 'WestEurope'
    }

...
```

@MarkWarneke

```
# integration.Tests.ps1

...

    AfterAll {
        # Remove test environment after test
        Write-Host "Removing test environment $ResourceGroupName..."

        Get-AzResourceGroup -Name $ResourceGroupName |
                Remove-AzResourceGroup -Force -AsJob
    }

    # Deploy Resource
    New-AzResourceGroupDeployment -ResourceName $ResourceName `
        -ResourceGroupName $ResourceGroupName


    # Run Acceptance Test
    . $PSScriptRoot/acceptance.spec.ps1 -ResourceName $ResourceName `
        -ResourceGroupName $ResourceGroupName
}
```

Test Dashboard

Azure DevOps Test

https://aka.ms/az.new

Azure DevOps | az-new / xAz.New / Pipelines / Builds / xAz.KV / #20190212.9 | Search | Sign in

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	1,405 Passed	100%	45s 63ms	0
Total tests	0 Failed	Pass percentage	Run duration	Tests not reported
+1,405	0 Others	↑ 100%	↑ +45s 63ms	

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			

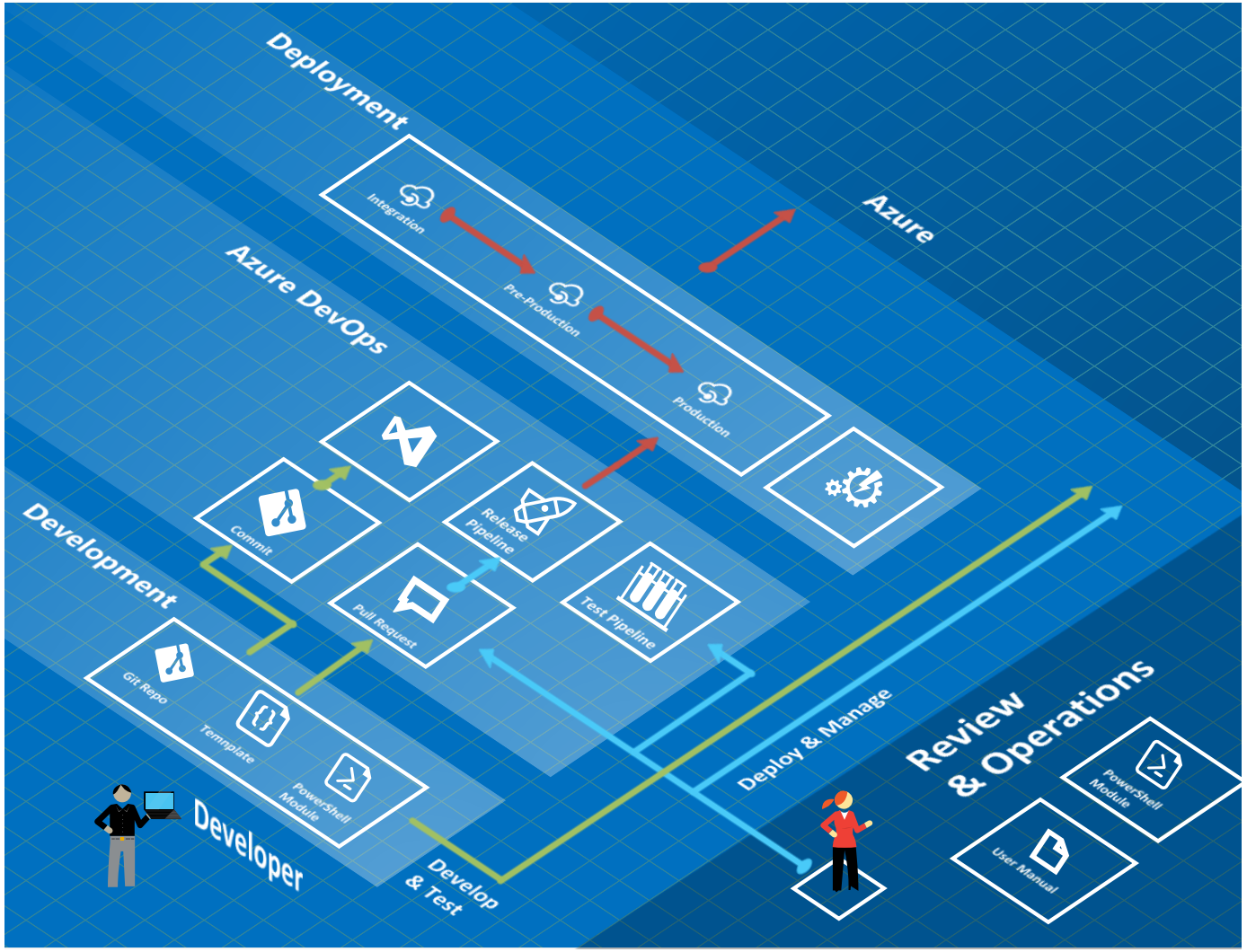
@MarkWarneke

```yaml
# azure-pipelines.yml

steps:
  - task: AzurePowerShell@4
    inputs:
      azureSubscription: $(azureSubscription)
      scriptType: "FilePath"
      # The name of the script where the pester test setup is located
      scriptPath: $(Build.SourcesDirectory)\Invoke-Pester.ps1
      scriptArguments: -OutputFormat 'NUnitXml‘ `
                       -OutputFile 'TestResults.Pester.xml' -PassThru'
      azurePowerShellVersion: "latestVersion"
      errorActionPreference: "continue"

  - task: PublishTestResults@2
    inputs:
      # Make sure to use the 'NUnit' test runner
      testRunner: "NUnit" # !!!
      testResultsFiles: "**/TestResults.Pester.xml"
      testRunTitle: "PS_Win2016_Unit"
      # Make the whole pipeline fail if a test is failed
      failTaskOnFailedTests: true
    displayName: "Publish Unit Test Results"
    condition: in(variables['Agent.JobStatus'], 'Succeeded', 'SucceededWithIssues', 'Failed')
```

Peer Review

four eyes principle

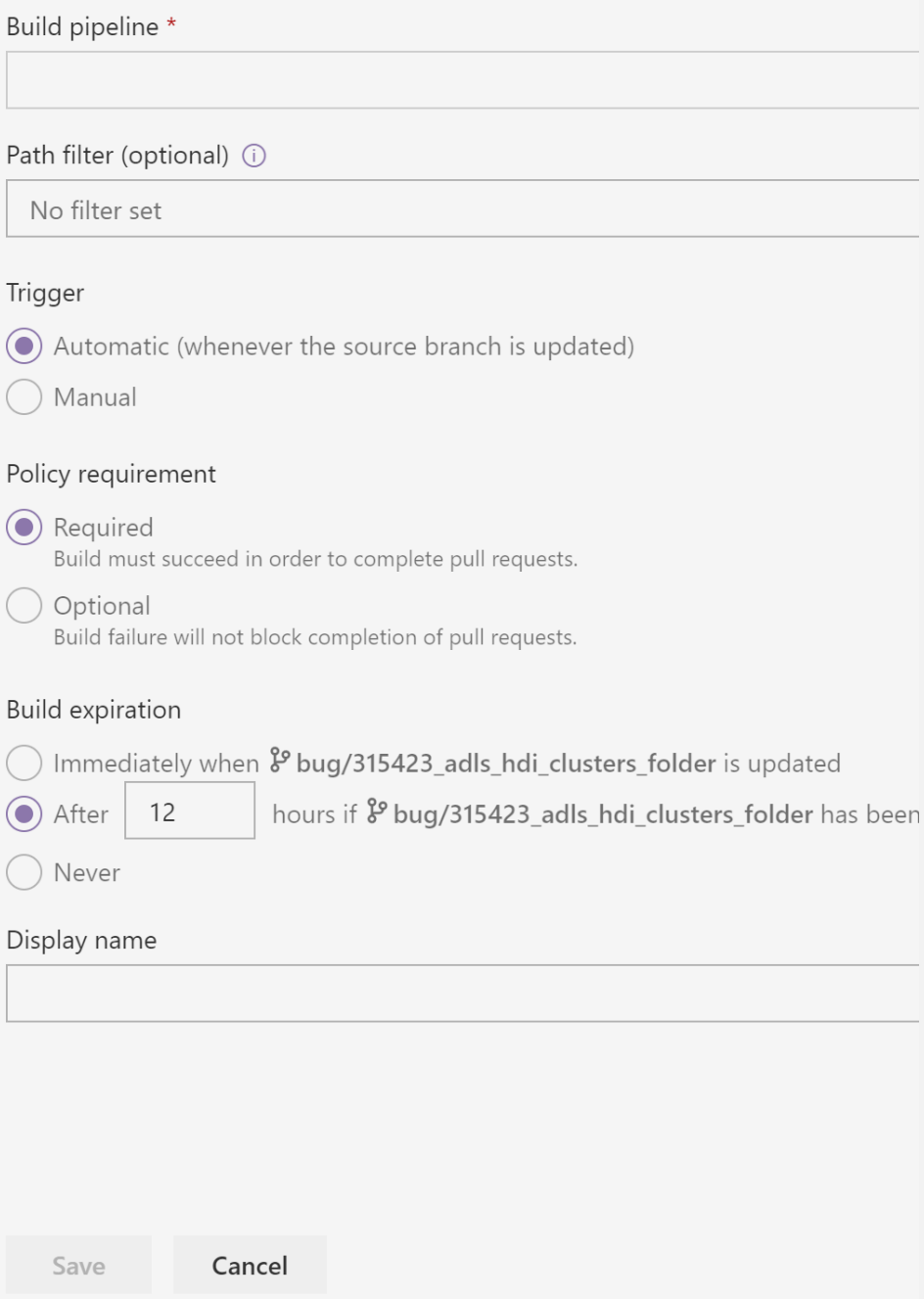
Pull-Request & Validation

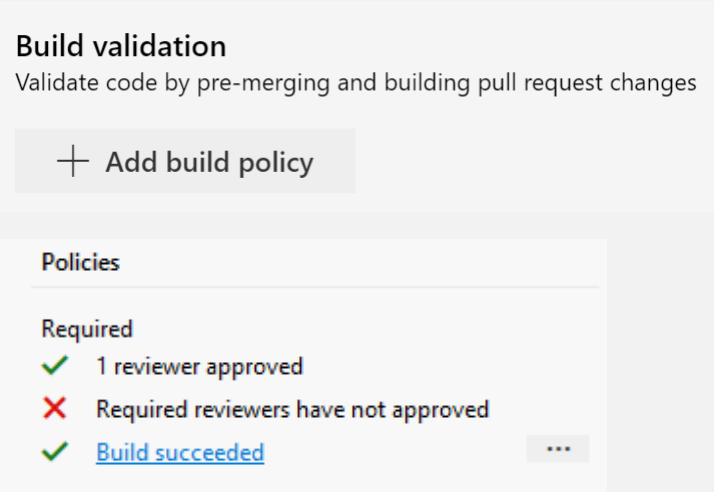


@MarkWarneke

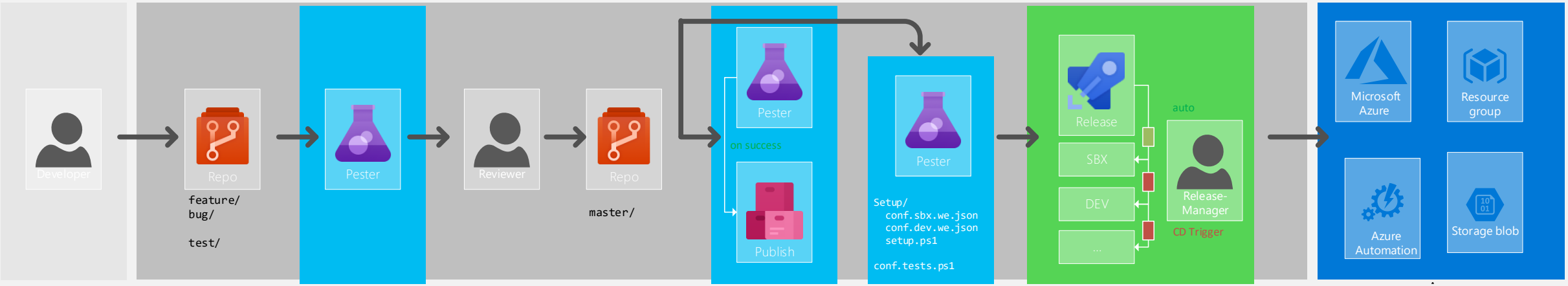
Build Policy Validation

- Add build policy for all tests

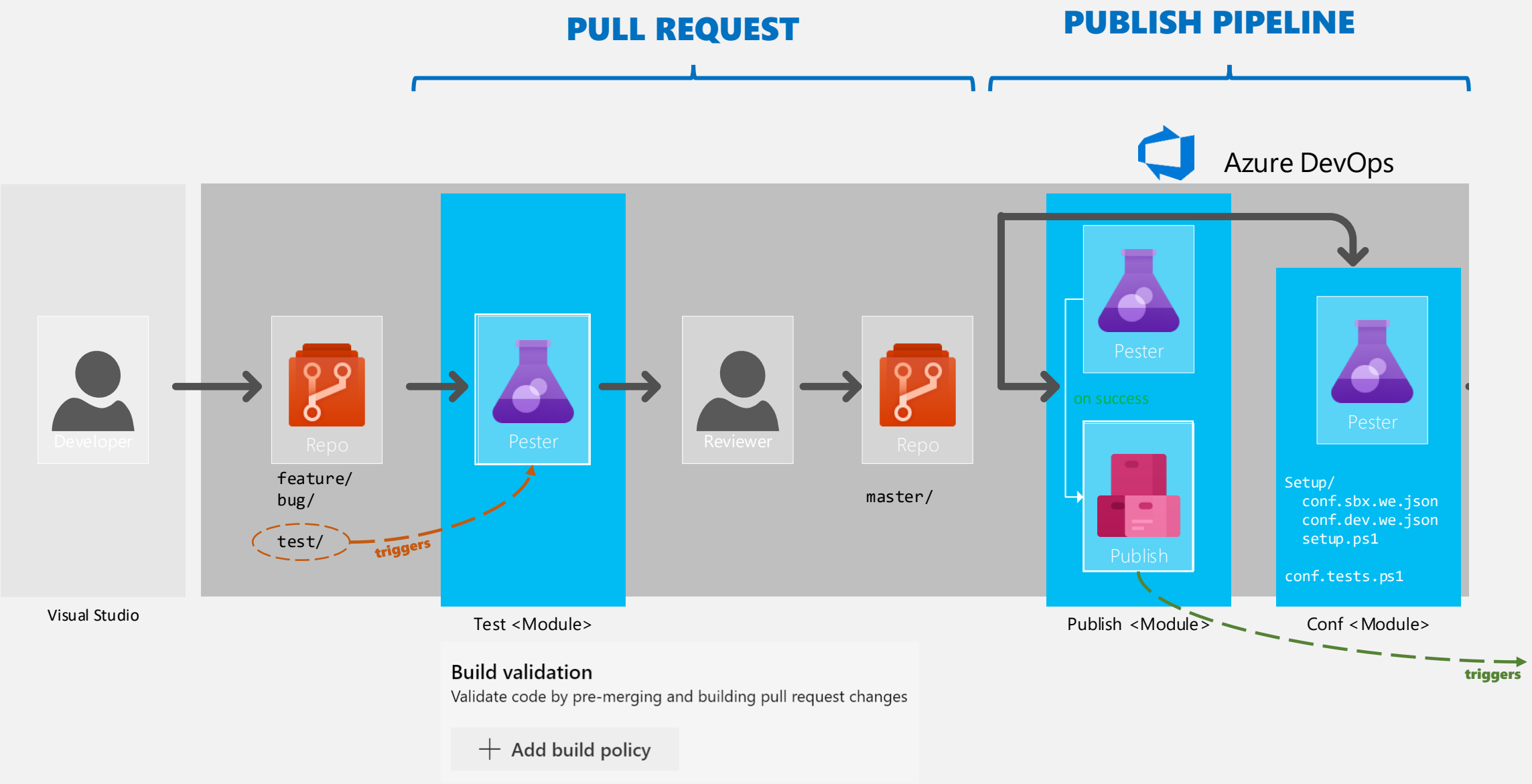




@MarkWarneke



https://aka.ms/az.new

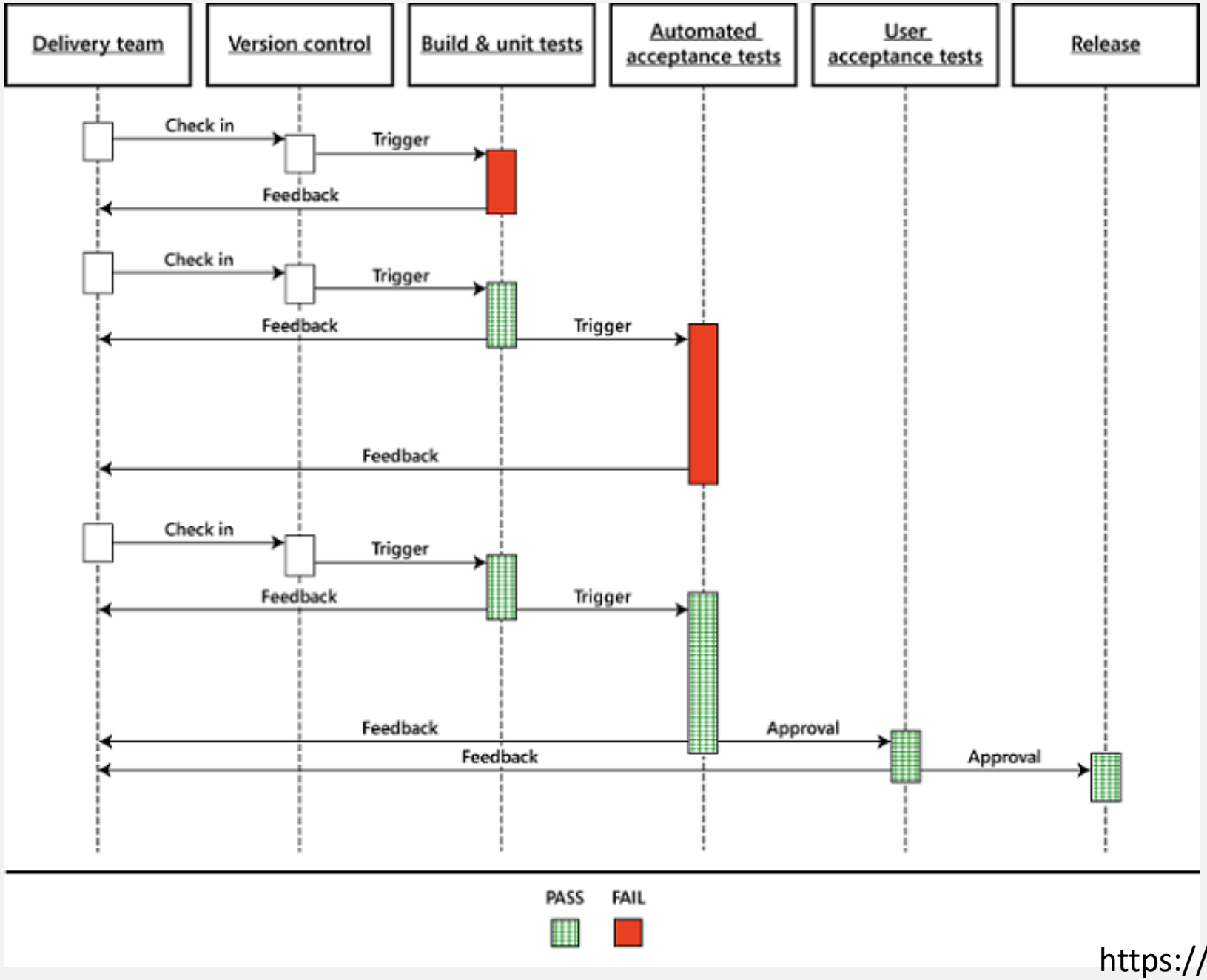


Build validation

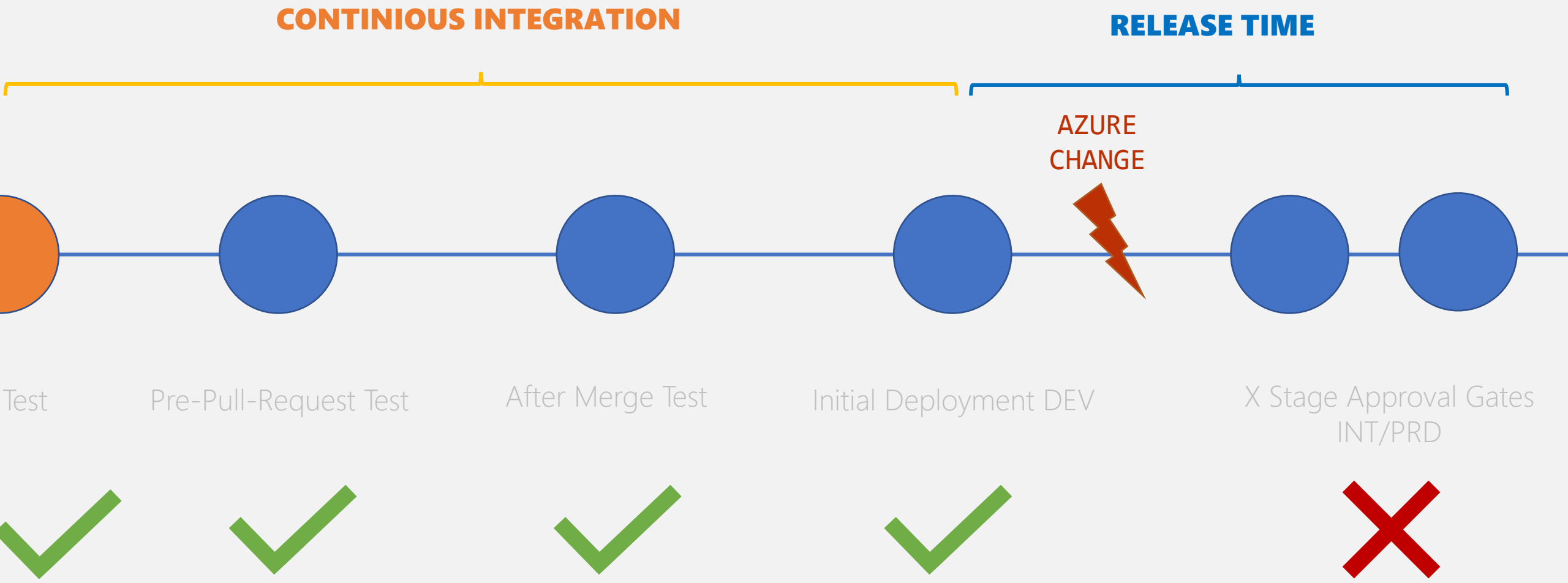
Validate code by pre-merging and building pull request changes

+ Add build policy

@MarkWarneke



https://continuousdelivery.com/


CONTINIOUS INTEGRATION
RELEASE TIME
AZURE CHANGE
Test
Pre-Pull-Request Test
After Merge Test
Initial Deployment DEV
X Stage Approval Gates INT/PRD

Wrap Up

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

@MarkWarneke

Microsoft