PowerShell Conference Europe 2019

Hannover, Germany

June 4-7, 2019

Test Infrastructure as Code?

Mark Warneke

After this Session...

I am able to create a test-suite for an Infrastructure as Code project from scratch.

I can articulate why Infrastructure as Code testing is necessary and increases the quality and reliability of provisioned services.

I have the ability to create an Infrastructure as Code project from scratch, quicker and much more mature.

Agenda

- Introduction to Infrastructre as Code
- DevOps foundations
- Quality & Maturity Framework
- Running Test Code (Hopefully)

Infrastructure as code is an approach to infrastructure automation **based on practices from software development**.

It emphasizes consistent, repeatable routines for provisioning and changing systems and their configuration.

– Kief Morris

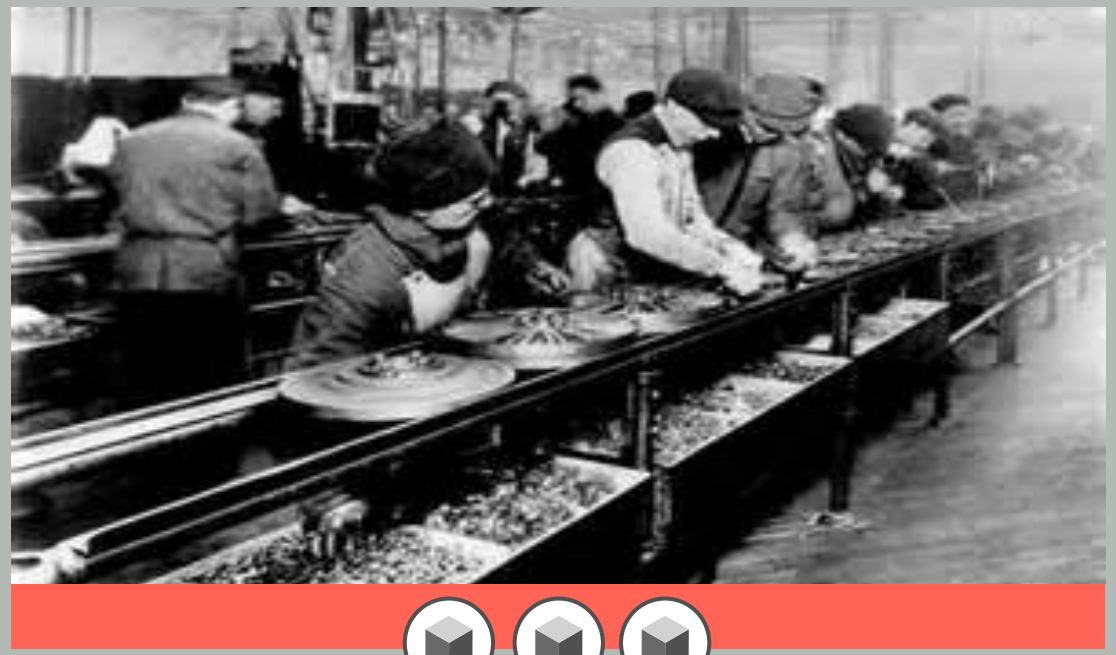
AppDev – InfraDev

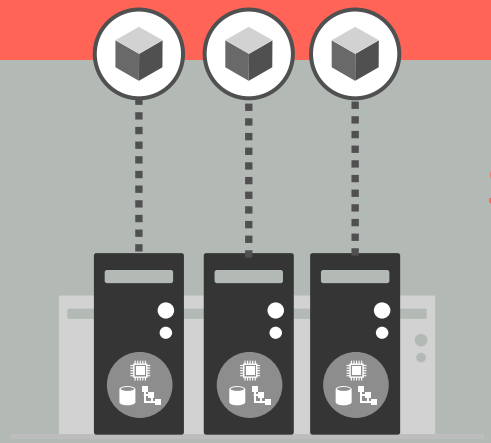
Application Development

Infrastructure Development

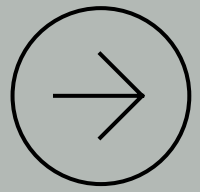
@MarkWarneke

IaaS - SaaS





Servers



Services

@MarkWarneke

Outside – Inside View

Outside

Hardware Configuration
- VM Size, Disks, Network
- RBAC, secrets etc.

Inside

Software
- Desired state
- Extensions & scripts

→ @IrwinStrachan
→@devblackops (Brandon Olin)

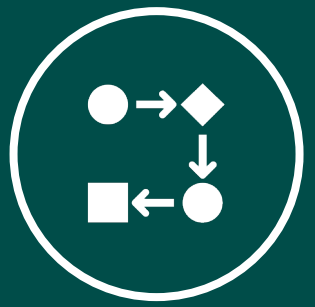
@MarkWarneke

Approaches

Declarative vs Imperative

functional

describe final state



procedural

executing steps to get to final state

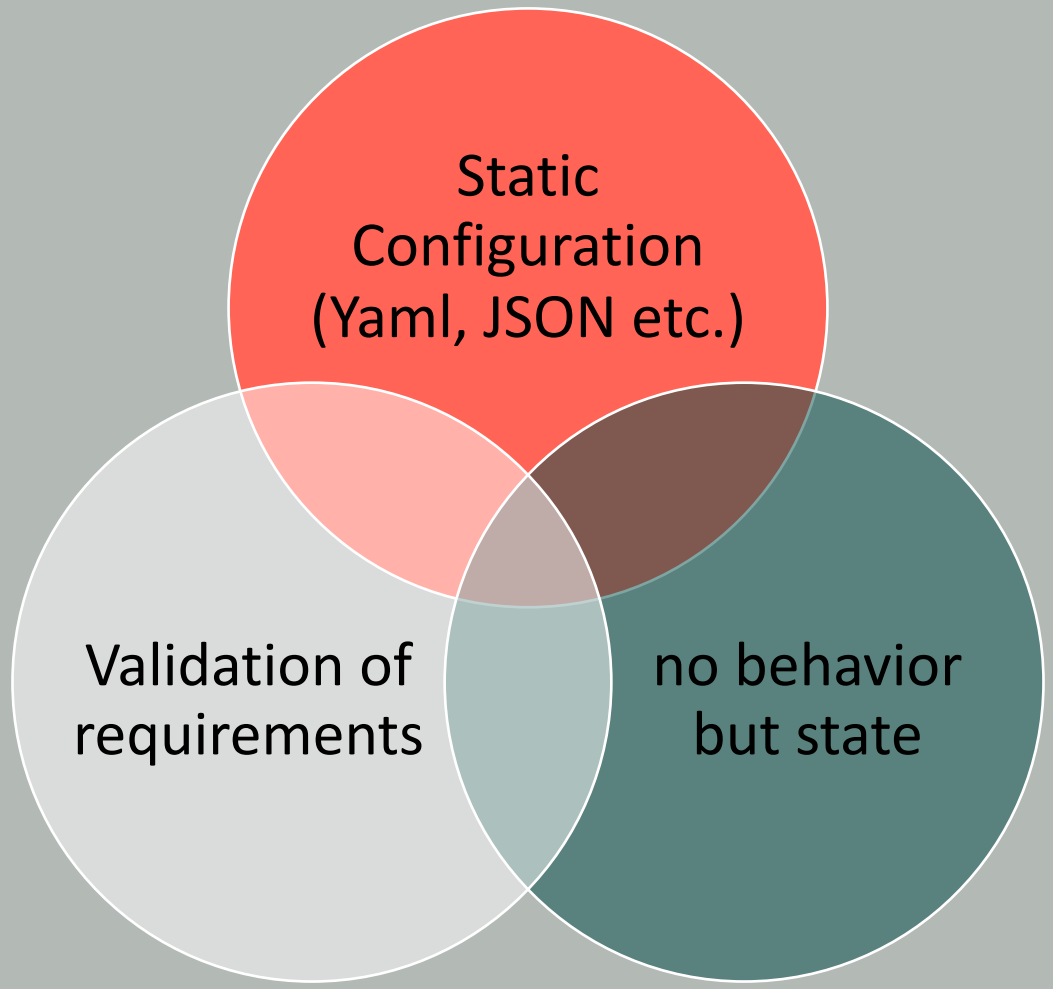
treat your servers
like cattle, not pets

– Bill Baker

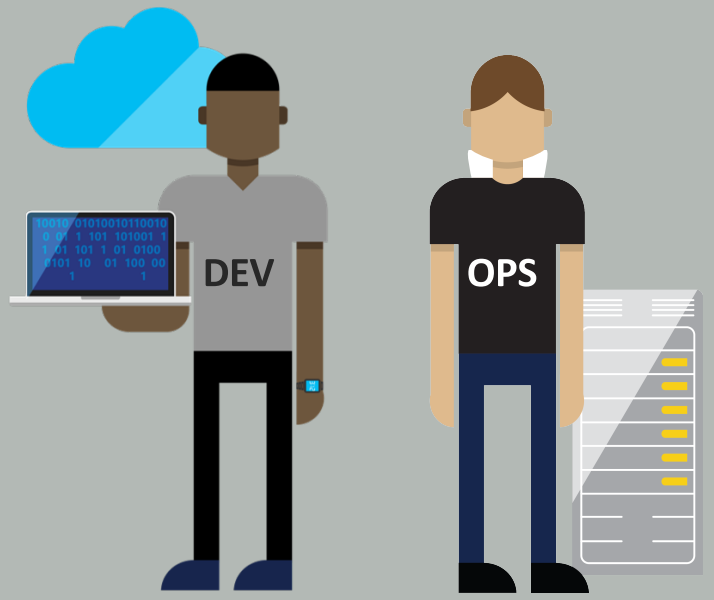
PSCONF.EU

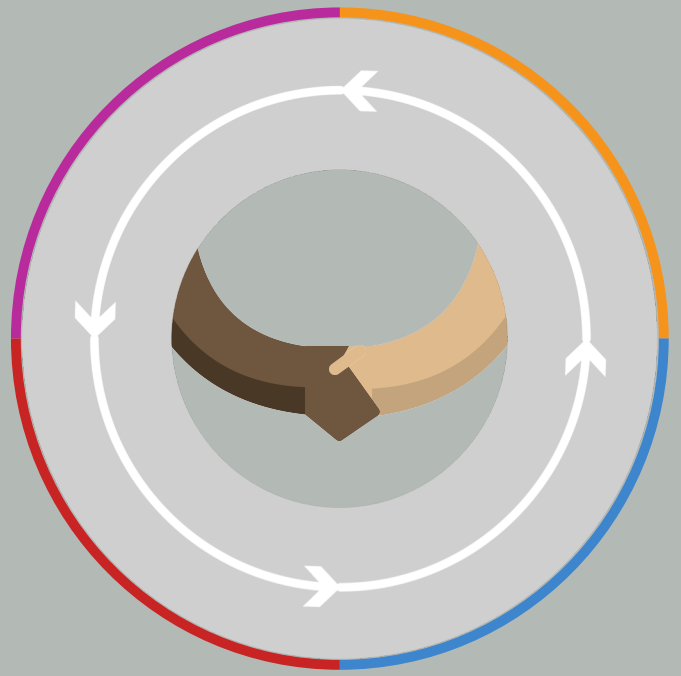
@MarkWarneke

Problem Statement



DevOps = People + Process + Tools

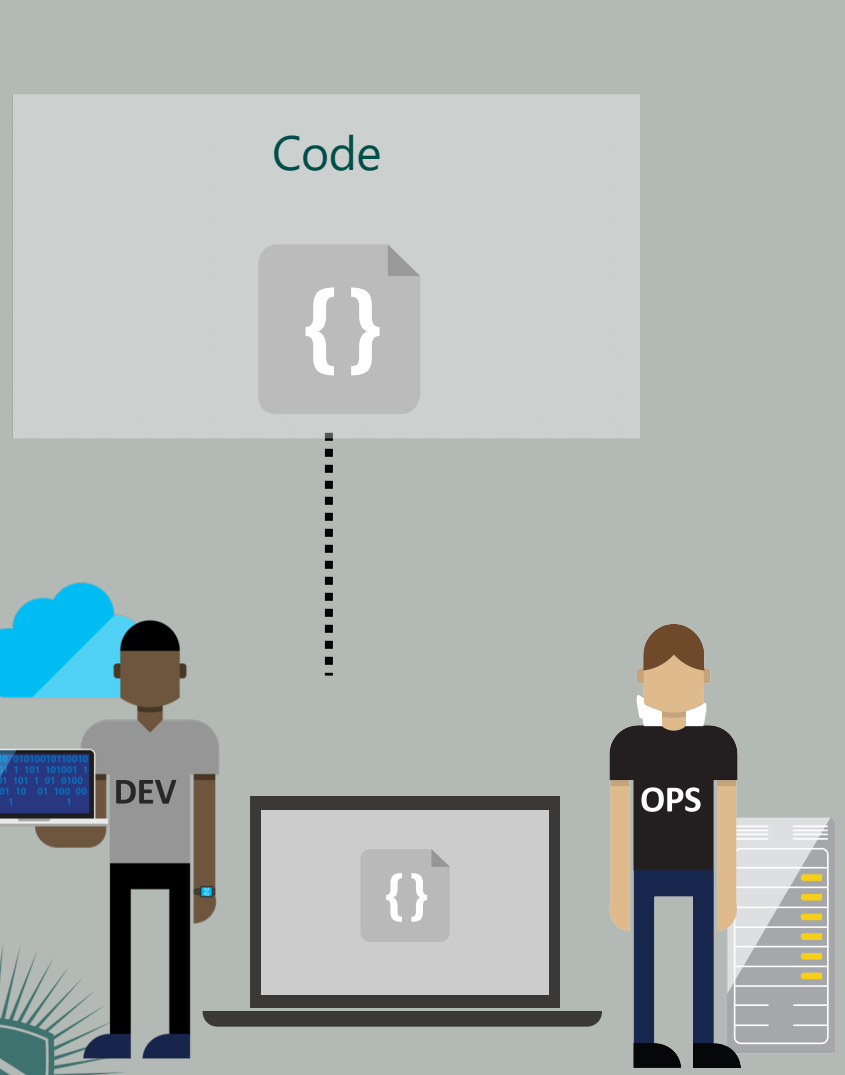


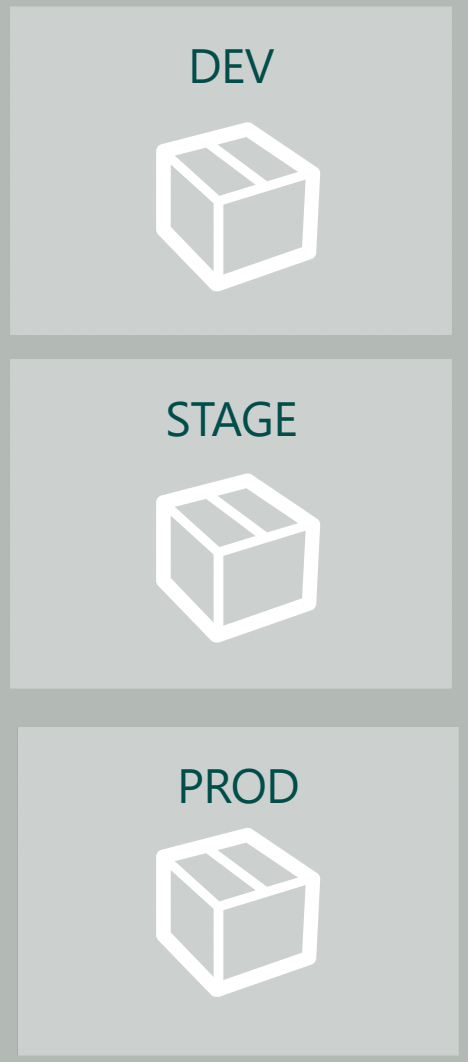


1	People
2	Process
3	Tools

@MarkWarneke

Infrastructure as Code





Value

- Optimized Resources
- Accelerate Delivery

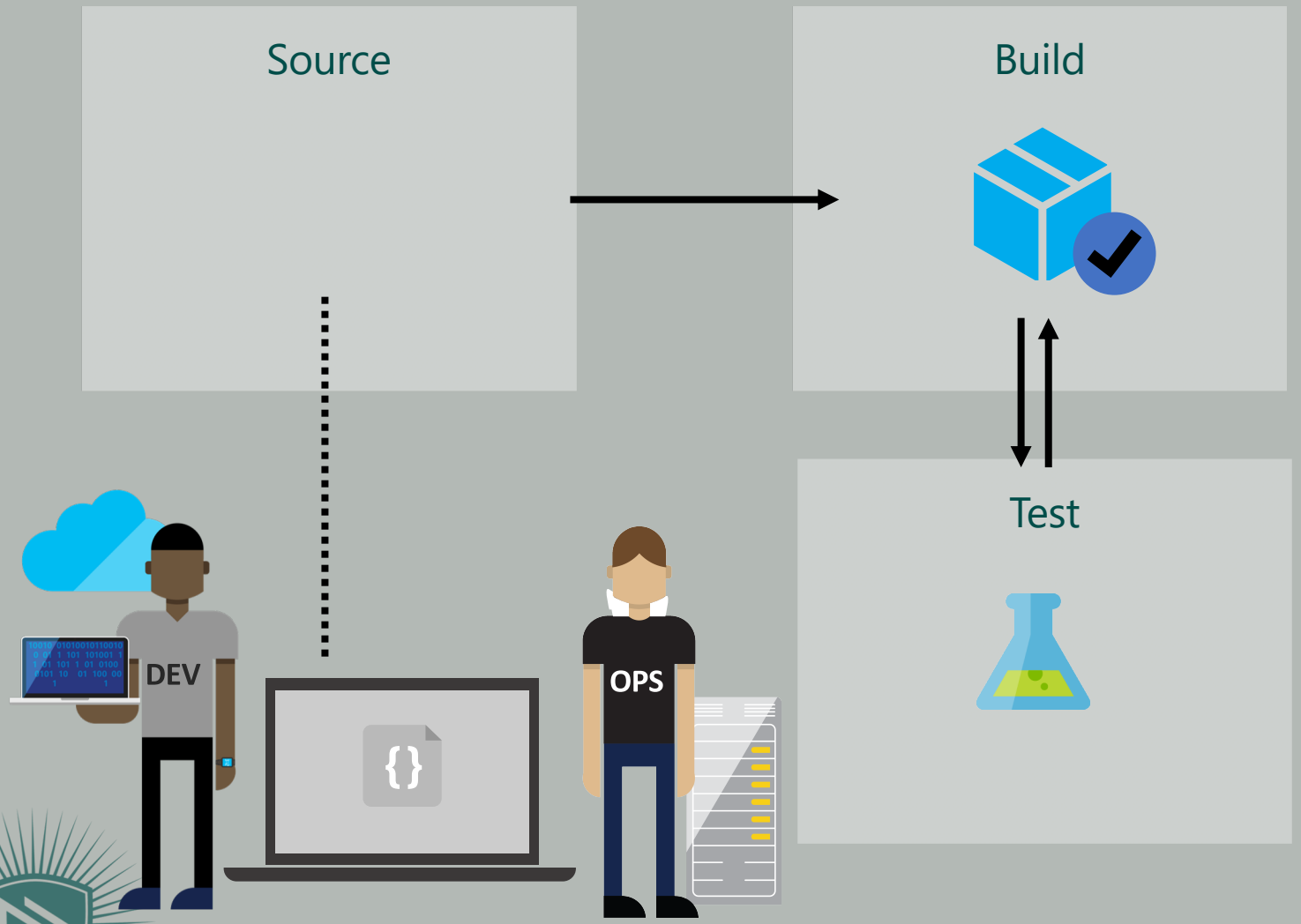
Measure

- Deployment Rate
- Mean Time To Release

PSCONF.EU

@MarkWarneke

Continuous Integration



Value

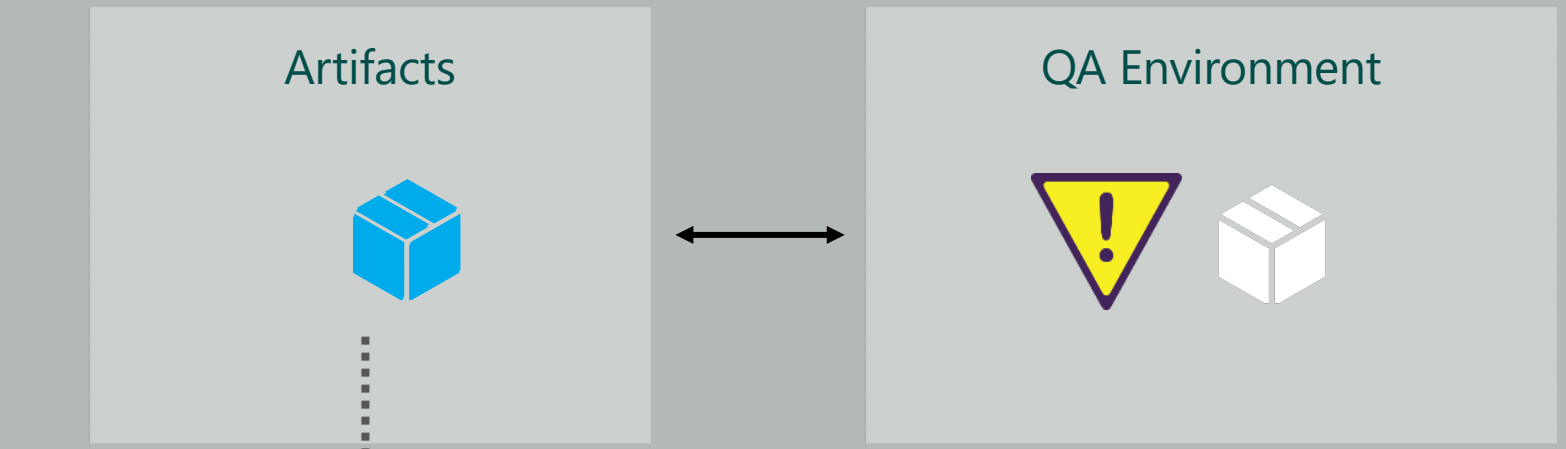
- Accelerate Delivery
- Repeatability
- Optimized Resources

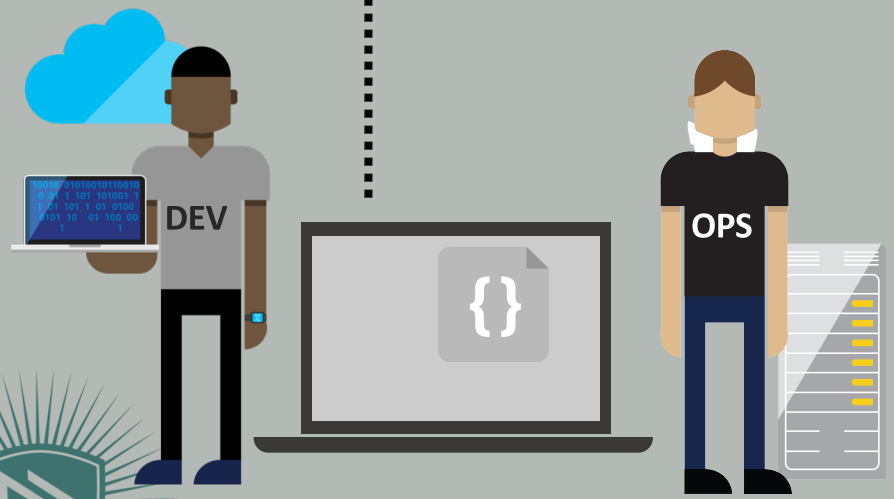
Measure

- More frequent releases
- Mean Time To Release
- Mean Time To Deploy

@MarkWarneke

Continuous Deployment





Value

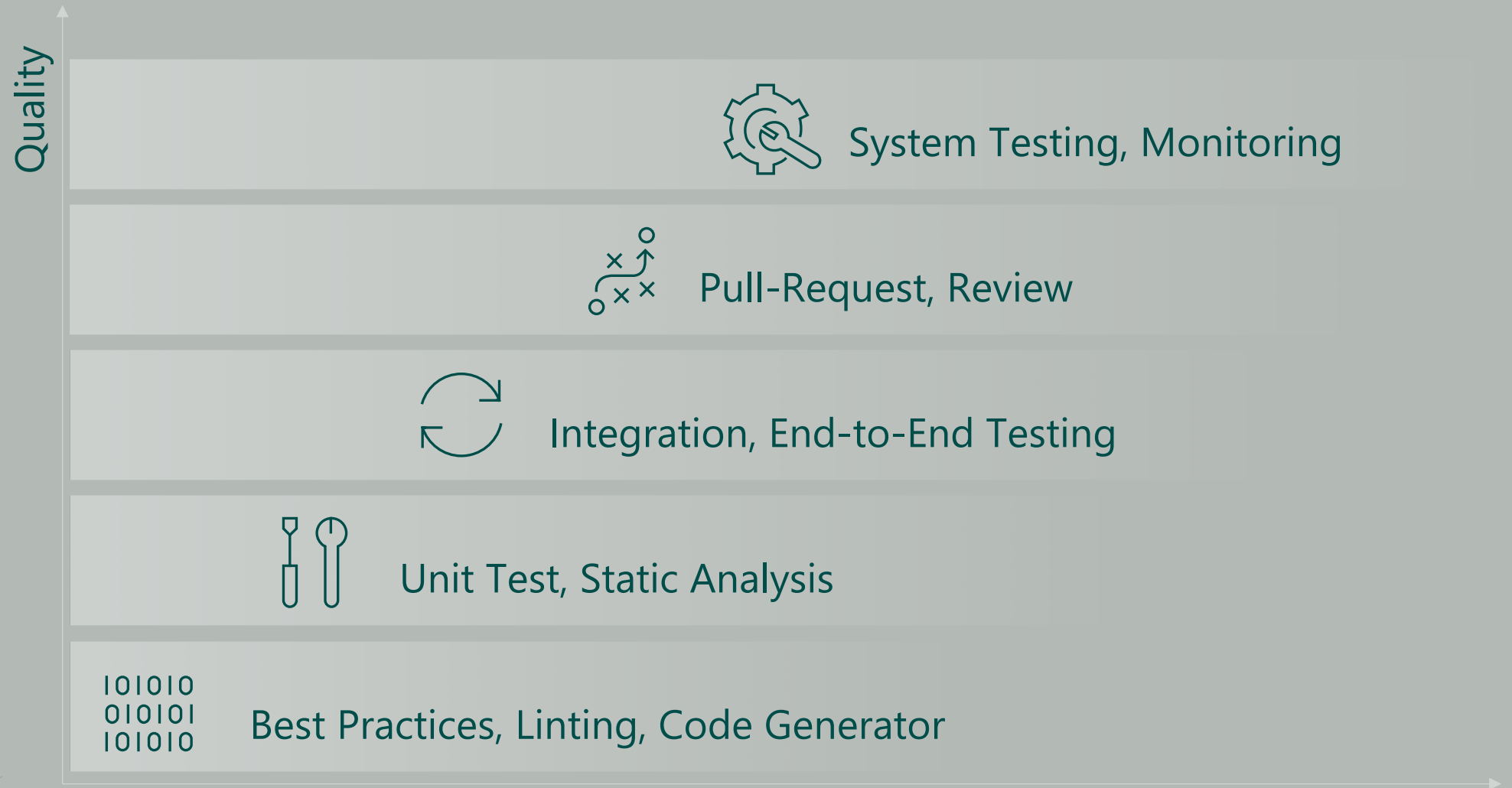
- Optimized Resources
- Accelerate Delivery

Measure

- Deployment Rate
- Mean Time To Release
- Availability

@MarkWarneke

Quality & Maturity



PSCONF.EU

@MarkWarneke

https://twitter.com/iamdevloper/status/397664295875805184?lang=en

@MarkWarneke

DEMO

- aka.ms/az.new
- aka.ms/az.new/resources

@MarkWarneke

DEMO

VSCode Extensions & Liniting

```
Describe "Get-Name function parameter validation" -Tags Unit {

    It "Should return name by convention" {

        $Company = 'PSConf'
        $Environment = 'TEST'

        $ComponentName = 'psconf-test'

        $name = @{
            Company $Company
            Environment $Environment
        }

        $AutomationAccountName = Get-xAz.AutomationAccountName @Name

        $AutomationAccountName | should BeLike "$Company*"
        $AutomationAccountName | should BeLike "*$Environment*"
        $AutomationAccountName | should BeLike "*-*" -Because 'Name should use delimter'
        $AutomationAccountName | should Not BeLike "*-" -Because 'Name should not end in delimter'
        $AutomationAccountName | should BeExactly $ComponentName
    }
}
```

```powershell
# Ensure valid json -> azuredeploy.json

param (
    $Path = 'azuredeploy.json'
)

Describe "..." {

  try {
      $text = Get-Content $Path -Raw -ErrorAction Stop
      $json = ConvertFrom-Json $text -ErrorAction Stop
  }
  catch {
      $JsonException = $_
  }

  it "should throw no expection" {
      $JsonException | Should -BeNullOrEmpty
  }
}
```

```powershell
# Validate presence of mandatory attributes
$TestCases = @(
  @{
    Expected = "parameters"
  },
  @{
    Expected = "variables"
  },
  @{
    Expected = "resources"
  },
  @{
    Expected = "outputs"
  }
)

it "should have <Expected>" -TestCases $TestCases {
  param(
      $Expected
  )
  $json.PSObject.Members.Name | Should -Contain $Expected
}
```

```powershell
# Ensure metadata (description) is present
context "parameters tests" {
    $parameters = $json.parameters | Get-Member -MemberType NoteProperty

    foreach ($parameter in $parameters) {
        $ParameterName = $($parameter.Name)
        it "$ParameterName should have metadata" {
            $json.parameters.$ParameterName.metadata | Should -Not -BeNullOrEmpty
        }
    }
}
```

```powershell
# Ensure the order of properties within azuredeploy.json
context "resources structure test" {
  foreach ($resource in $json.resources) {
      it "should follow comment > type > apiVersion > name > properties" {
          "$resource" | Should -BeLike "*comments*type*apiVersion*name*properties*"
      }
  }
}
```

DEMO

WhatIf

```
Describe "New-xAz.NetVnetComponent WhatIf" -Tags Unit {

    InModuleScope $script:ModuleName {
        It "should return valid parameter" {

            $Location = 'westeurope'
            $VnetName = 'Net-01-WE'
            $TestResourceGroupName = 'Dev-Vnet'

            Mock Get-AzResourceGroup { return @{ ResourceGroupName = 'Dev-Vnet' } }

            $InputObject = @{
               #...
            }
            $OutputObject = New-xAz.NetVnetComponent @InputObject -WhatIf

            $OutputObject.VirtualNetwork | Should Be $VnetName
            $OutputObject.ResourceGroupName | Should Be $TestResourceGroupName
        }
    }
}
```

DEMO

Integration

```powershell
Describe "New-xAz.KvltComponent integration tests" -Tags Build {
    # ...
    try {
        $OutputObject = New-xAz.KvltComponent @InputObject -ErrorAction Stop
        Write-Verbose -Message ("Got outputobject {0}" -f $OutputObject)

        $Vault = Get-AzKeyVault $ComponentName
    }
    catch {
        $Exception = $_
        Write-Verbose -Message $Exception.Exception
    }

    it "should not throw an exception" {
        $Exception | Should -BeNullOrEmpty
    }

    it "should have firewall" {
        $Vault.NetworkAcls.IpAddressRanges.Count | Should -BeGreaterOrEqual 1
    }

    it "should have access policies" {
        $vault.AccessPolicies.Count | Should -BeGreaterOrEqual 1
    }
}
```

```
Connect-AzAccount

Describe "how to clean up" {
    $TestResourceGroupName = "AUTOTEST-$(Get-Date -Format FileDateTime)"
    $Location = 'WestEurope'

    BeforeEach {
        Write-Host "Create Test ResourceGroup $TestResourceGroupName..." -ForeGroundColor Blue
        $null = New-AzResourceGroup -Name $TestResourceGroupName -Location $Location
        Write-Host "Test started." -ForeGroundColor Blue
    }

    it "should cleanup" { "A" | Should Be "A" }

    AfterEach {
        Write-Host "Remove ResourceGroup $TestResourceGroupName..." -ForeGroundColor Blue
        Get-AzResourceGroup -Name $TestResourceGroupName -ErrorAction SilentlyContinue |
             Remove-AzResourceGroup -Force
        Write-Host "Test completed." -ForeGroundColor Blue
    }
}
```

```powershell
# Pass configuration file to test
param(
    [string] $ConfigurationFile = 'config.test.json'
)

$script:ModuleName = 'xAz.La' # log analytics module

# Get the config values from file
$Path = Join-Path $PSScriptRoot $ConfigurationFile
$script:config = (Get-Content -Raw $Path) | ConvertFrom-Json

# Test Case
Describe "..." {

    $InputObject = @{
        Environment = $config.Environment
        # other parameter
    }
    $OutputObject = New-xAz.LaComponent @InputObject -Verbose

    it "..." {
        $OutputObject | Should # ...
    }
}
```

DEMO

Azure DevOps

Azure DevOps YAML

https://docs.microsoft.com/en-us/azure/devops/pipelines/yaml-schema?view=azure-devops&tabs=schema

```yaml
# Trigger pipeline, can provide wildecard
# and file path
trigger:
- master

# Variables for pipeline e.g. module name
variables:
  azureSubscription: "Mark"
  feed.name: "xAz"
  organization: "az-new"
  module.name: "xAz.Cosmos"

# one or more jobs (job per agent)
jobs:
    # could be self hosted agent
  - job: Build_PS_Win2016
    pool:
      vmImage: vs2017-win2016


    steps:
    - checkout: self
      persistCredentials: true

   - task: AzurePowerShell@4
   # ...
   # more tasks with configuration
```

```yaml
- task: AzurePowerShell@4                          # Version 4 in preview for az module
  inputs:
    azureSubscription: $(azureSubscription)       # Scope could be handled (RG or Subscription)
    scriptType: "FilePath"
    scriptPath: $(Build.SourcesDirectory)\$(Module.Name)\psake.ps1
    scriptArguments: -TaskList Test -Verbose
    azurePowerShellVersion: "latestVersion"
    errorActionPreference: "continue"             # Result will handle termination

- task: PublishTestResults@2
  inputs:
    testRunner: 'NUnit'
    testResultsFiles: '**/TestResults.module.xml'
    testRunTitle: 'PS_Win2016_Module'
    failTaskOnFailedTests: true                   # Leave empty if test result should not terminate
  displayName: 'Publish Module Test Results'
  condition: in(variables['Agent.JobStatus'], 'Succeeded', 'SucceededWithIssues', 'Failed')
```

```
Task Test -Depends Init, PrepareTest {

    $Folder = "Unit" # "Module", "Integration" ...

    foreach ($module in $ModuleBase) {
        # Execute tests
        $moduleRoot = Join-Path -Path $ProjectRoot -ChildPath $module
        $testScriptsPath = Join-Path -Path $moduleRoot -ChildPath 'test' | Join-Path -ChildPath $Folder
        $testResultsFile = Join-Path -Path $ProjectRoot -ChildPath "TestResults.$Folder.xml"

        $pester = @{
            Script       = $testScriptsPath
            OutputFormat = 'NUnitXml'       # Expected Format of Test Result Publisher
            OutputFile   = $testResultsFile # location on build agent path -> For Result Publish
            PassThru     = $true
            ExcludeTag   = 'Incomplete, Unit'
        }
        $null = Invoke-Pester @pester
    }
}
```

xAz.New
Overview
Boards
Repos
Pipelines
Builds
Releases
Library
Task groups
Deployment groups
Test Plans
Artifacts
Compliance
Project settings

#20190212.9: fix interactive prompt

Release All logs

Triggered feb 12 at 6:15 pm 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

Bug Link Test run Column Options

Filter by test or run name Test file Owner Outcome: **Aborted (+1)**

Hooray! There are no test failures.

Change the test outcome filter to view tests relevant to you.

Summary

- What is infrastructure as code
- DevOps foundation
- Script Analyzer & Help Checker
- Static Analysis of configuration file
- Unit Test of IaC deployment
- Unit Test of pipeline
- Integration / System Test with Pester

Questions?

Use the conference app to vote for this session:
https://my.eventraft.com/psconfeu

@MarkWarneke

Maturity Model

Practice	Build management and continuous integration	Environments and deployment	Release management and compliance	Testing	Data management	Configuration management
Level 3 - Optimizing: Focus on process improvement	Teams regularly meet to discuss integration problems and resolve them with automation, faster feedback, and better visibility.	All environments managed effectively. Provisioning fully automated. Virtualization used if applicable.	Operations and delivery teams regularly collaborate to manage risks and reduce cycle time.	Production rollbacks rare. Defects found and fixed immediately.	Release to release feedback loop of database performance and deployment process.	Regular validation that CM policy supports effective collaboration, rapid development, and auditable change management processes.
Level 2 - Quantitatively managed: Process measured and controlled	Build metrics gathered, made visible, and acted on. Builds are not left broken.	Orchestrated deployments managed. Release and rollback processes tested.	Environment and application health monitored and proactively managed. Cycle time monitored.	Quality metrics and trends tracked. Non functional requirements defined and measured.	Database upgrades and rollbacks tested with every deployment. Database performance monitored and optimized.	Developers check in to mainline at least once a day. Branching only used for releases.
Level 1 - Consistent: Automated processes applied across whole application lifecycle	Automated build and test cycle every time a change is committed. Dependencies managed. Re-use of scripts and tools.	Fully automated, self-service push-button process for deploying software. Same process to deploy to every environment.	Change management and approvals processes defined and enforced. Regulatory and compliance conditions met.	Automated unit and acceptance tests, the latter written with testers. Testing part of development process.	Database changes performed automatically as part of deployment process.	Libraries and dependencies managed. Version control usage policies determined by change management process.
Level 0 – Repeatable: Process documented and partly automated	Regular automated build and testing. Any build can be re-created from source control using automated process.	Automated deployment to some environments. Creation of new environments is cheap. All configuration externalized / versioned	Painful and infrequent, but reliable, releases. Limited traceability from requirements to release.	Automated tests written as part of story development.	Changes to databases done with automated scripts versioned with application.	Version control in use for everything required to recreate software: source code, configuration, build and deploy scripts, data migrations.
Level -1 – Regressive: processes unrepeatable, poorly controlled, and reactive	Manual processes for building software. No management of artifacts and reports.	Manual process for deploying software. Environment-specific binaries. Environments provisioned manually.	Infrequent and unreliable releases.	Manual testing after development.	Data migrations unversioned and performed manually.	Version control either not used, or check-ins happen infrequently.

[Jez Humble, David Farley: Continuous Delivery Reliable]

@MarkWarneke

Testing, Testing, 1…2…3: Using Pester for Infrastructure Validation by Brandon Olin

- https://www.youtube.com/watch?v=6bPByJX5euc

Infrastructure validation using Pester, by Irwin Strachan

- https://www.youtube.com/watch?v=Qfi_H7IZyHg

Microsoft AzureCAT
Customer Advisory Team

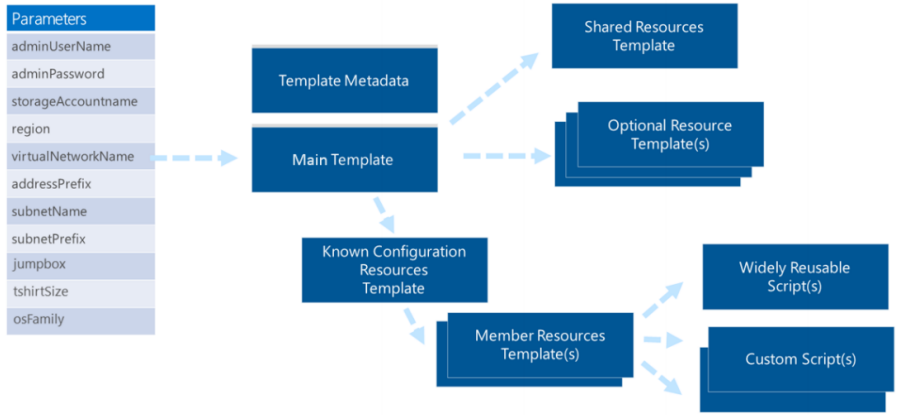
World Class ARM Templates - Considerations and Proven Practices

Marc Mercuri, Principal Program Manager,
Ulrich Homann, Distinguished Architect
George Moore, Principal Program Manager Lead

Reviewers – Silvano Coriani, Rafael Godinho, Paige Lu, Rama Ramani, Jeremiah Talker, Arsen Vladimirskiy, Tim Wieman , Geert Baeke

June 30, 2015

Executive summary



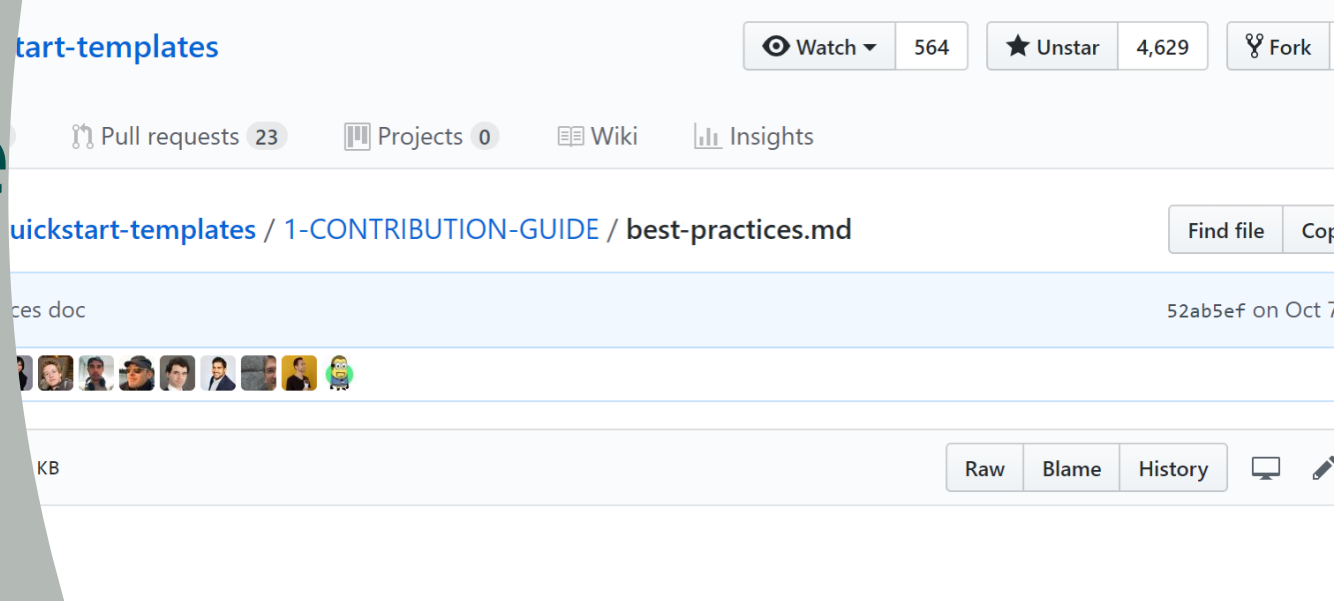
http://download.microsoft.com/download/8/E/1/8E1DBEFA-CECE-4DC9-A813-93520A5D7CFE/World%20Class%20ARM%20Templates%20-%20Considerations%20and%20Proven%20Practices.pdf

PSCONF.EU

@MarkWarneke

Best Practices Guide

- https://github.com/Azure/azure-quickstart-templates
- https://github.com/Azure/azure-quickstart-templates/blob/master/1-CONTRIBUTION-GUIDE/best-practices.md



Best Practices Guide

- https://blogs.msdn.microsoft.com/mvpawardprogram/2018/05/01/azure-resource-manager/

Best Practices For Using Azure Resource Manager Templates

Rate this article ★★★★★

MVP Award Program May 1, 2018

Share 19 | 0 | 0 | 1

Editor's note: The following post was written by Visual Studio and Development Technologies MVP Peter Groenewegen Microsoft Azure MVP Pascal Naber as part of our Technical Tuesday series. Mia Chang of the Technical Committee served as the Technical Reviewer of this piece.

This article focuses on best practices regarding the automated deployment of resources to Azure. We have mplemented Continuous Deployment (CD) pipelines including the provisioning of Azure resources for many ustomers, and we would like to share our experience so you can benefit from it. These practices will help you eate more reliable, testable, reusable, and maintainable templates.

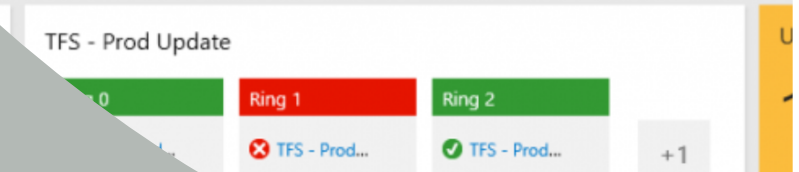
tomate deployments to Azure

Resource Manager templates (ARM templates) are the preferred way of automating the deployment of es to Azure Resource Manager (AzureRM). ARM templates are JavaScript Object Notation (JSON) files. urces that you want to deploy are declaratively described within JSON. An ARM template is nt, which means it can be executed as many times as you wish, and the result will be the same every takes care of the execution and identifies the changes that need to be executed.

oning infrastructure, we apply the same best practices as with deploying applications. This is also tructure as Code. Applying CD enables you to develop your infrastructure in a repeatable and you can reuse your ARM templates over multiple teams by applying these practices. This dashboard to monitor the quality of the infrastructure provisioning.

ates in a CD pipeline, our preferred method uses Visual Studio Team Services (VSTS). The y a VSTS task: "Azure Resource Group Deployment".

onitor all your builds and releases, and this will give you a quick overview of the he quality of your templates. It is very useful to show your team and other



Videos

- https://app.pluralsight.com/library/courses/microsoft-azure-resource-manager-mastering/table-of-contents
- https://channel9.msdn.com/Events/Build/2015/3-618
- https://channel9.msdn.com/Events/Build/2015/2-659
- https://channel9.msdn.com/Events/Ignite/2015/BRK4453

Blogs

Azure Security Audits with Pester:
https://samcogan.com/azure-security-audits-with-pester/

Infrastructure as Code Maturity Model:
https://medium.com/@GaryStafford/infrastructure-as-code-maturity-model-9206b21d5dad

An introduction to infrastructure testing with PowerShell Pester
https://4sysops.com/archives/an-introduction-to-infrastructure-testing-with-powershell-pester/

about_Speaker

Mark Warneke

Consultant

Microsoft

http://aka.ms/mark/

github.com/MarkWarneke

@MarkWarneke

PSCONF.EU