How PowerCLI Makes vSphere Configuration Management Easy

Luc Dekens
Kyle Ruddy

Luc Dekens

PowerCLI Mastermind

Writer @ lucd.info
GitHub @ github.com/lucdekens
Twitter @lucd22
MS MVP Cloud\Datacenter Management

Kyle Ruddy

Senior Architect, Technical Marketing

Writer @ kmruddy.com
GitHub @ github.com/kmruddy
Podcast @ vBrownBag.com
Twitter @kmruddy
MS MVP Cloud\Datacenter Management

Agenda

- What is Configuration Management?
- What is PowerShell?
- What is PowerCLI?
- Desired State Configuration (DSC) Introduction
- DSC Resources for VMware
- DSC Resource Testing

Configuration Management

“Configuration management (CM) is a systems engineering process for establishing and maintaining consistency of a product's performance, functional, and physical attributes with its requirements, design, and operational information throughout its life.”

https://en.wikipedia.org/wiki/Configuration_management

Provides:

- Declarative and idempotent (repeatable) deployment
- Unified configuration
- Conformance

Configuration files separate intent:

- **Formerly:** How I want to do it
- **Now:** What I want it to be

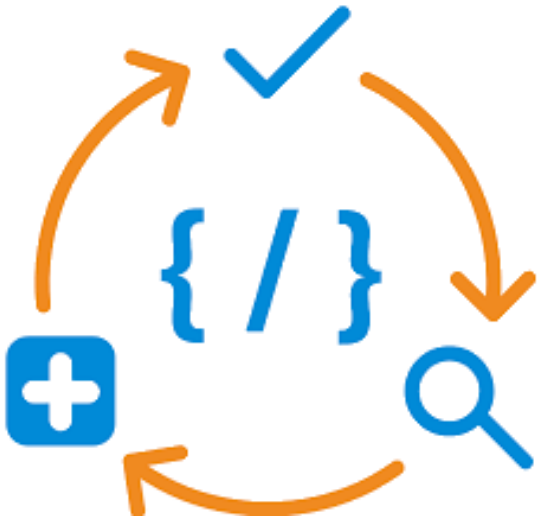
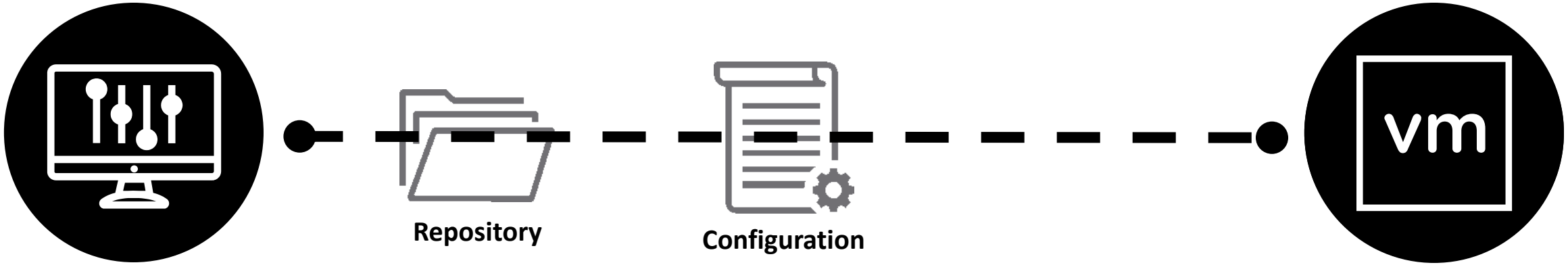


Image Courtesy of: https://www.chef.io/configuration-management/

Configuration Management Breakdown

What's really going on…



	Ansible	Chef	Puppet	SaltStack	PowerShell DSC
Script Language	Python	Ruby	Custom Ruby DSL	Python	PowerShell
Infrastructure	Controller applies config via SSH	Chef Workstation push config to Chef Server, then to nodes	Puppet Master syncs config to nodes	Salt Master push config to Minion	Node pulls config from Pull Server
Central Point of Control	Ansible Tower	Chef Server	Puppet Master	Salt Master	Pull Server or Azure Automation
Script Terminology	Playbook / Roles	Recipes / Cookbooks	Manifests / Modules	State / Formula	Configurations / Resources
Task Execution Order	Sequential	Sequential	Non-Sequential	Sequential	Sequential

Partial Source: https://www.digitalocean.com/community/tutorials/an-introduction-to-configuration-management

Ansible Example – VMware

Configure Host NTP Settings

```
- name: Set NTP servers for all ESXi Host in given Cluster
  vmware_host_ntp:
    hostname: '{{ vcenter_hostname }}'
    username: '{{ vcenter_username }}'
    password: '{{ vcenter_password }}'
    cluster_name: '{{ cluster_name }}'
    state: present
    ntp_servers:
        - 0.pool.ntp.org
        - 1.pool.ntp.org
  delegate_to: localhost
```

Ansible Example - URI

VCSA SSH Service Management

```yaml
- hosts: localhost
  become: no

  tasks:
    - name: vcenter login
      uri:
        url: https://cloudvc.student.lab/rest/com/vmware/cis/session
        force_basic_auth: yes
        method: POST
        user: administrator@vsphere.local
        password: P@ssw0rd
        status_code: 200
        validate_certs: no
      register: login

    - name: disable ssh
      uri:
        url: https://cloudvc.student.lab/rest/appliance/access/ssh
        force_basic_auth: yes
        method: PUT
        body_format: json
        body: "{{ lookup('file','sshoff.json') }}"
        validate_certs: no
        headers:
          Cookie: "{{login.set_cookie}}"
```

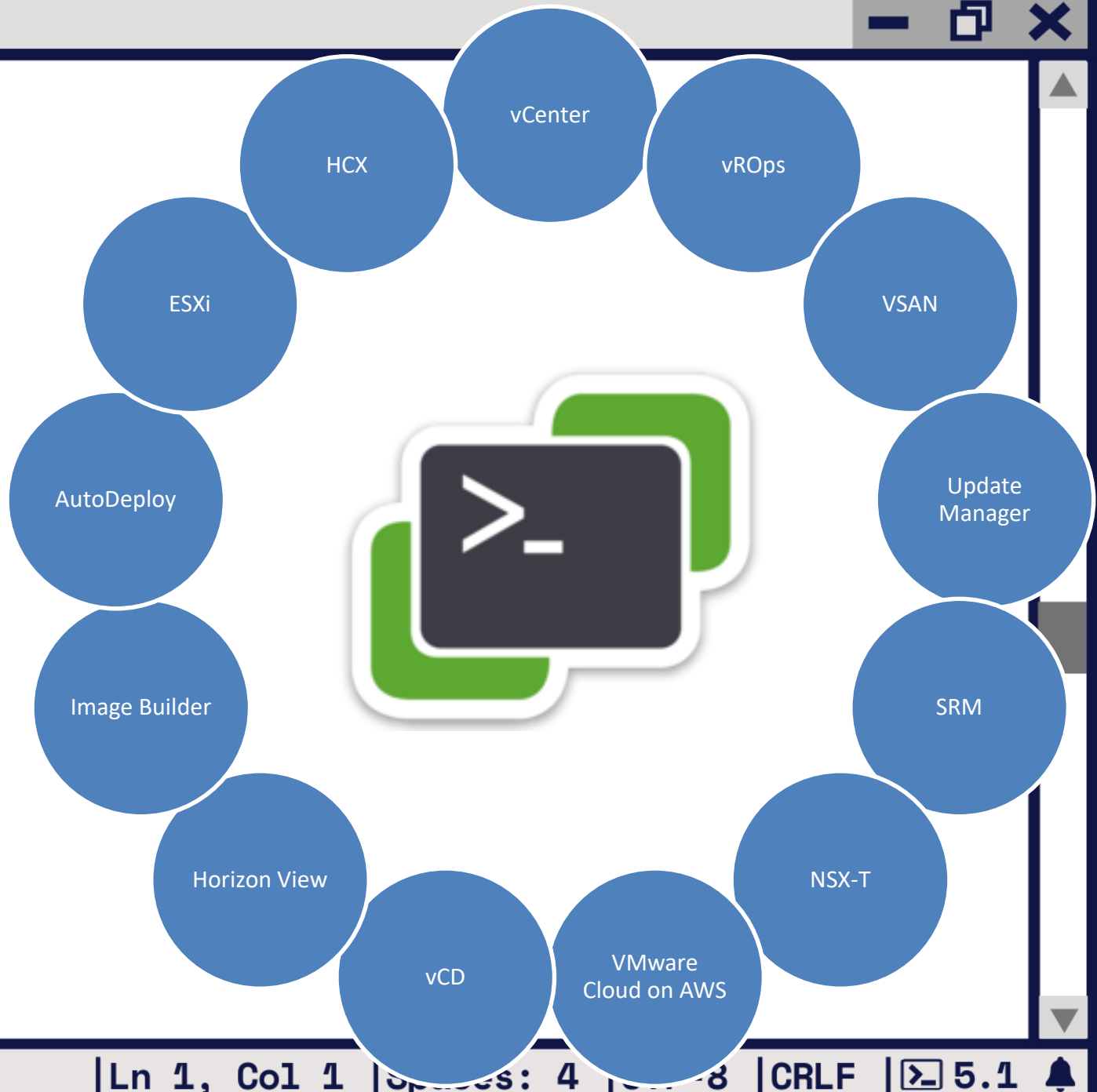
What is PowerShell?

- A simple and straight-forward path to automation
  - Already installed on all modern Windows Operating Systems
  - Integrated and rich help system
- Modular and Object-oriented
  - The best of a programming language melded with a scripting language
  - True portability of code via modules (and snap-ins)
- PowerShell Core 6.0 available additionally for Linux and MacOS
- Open Sourced: https://github.com/PowerShell/PowerShell

What is PowerCLI?

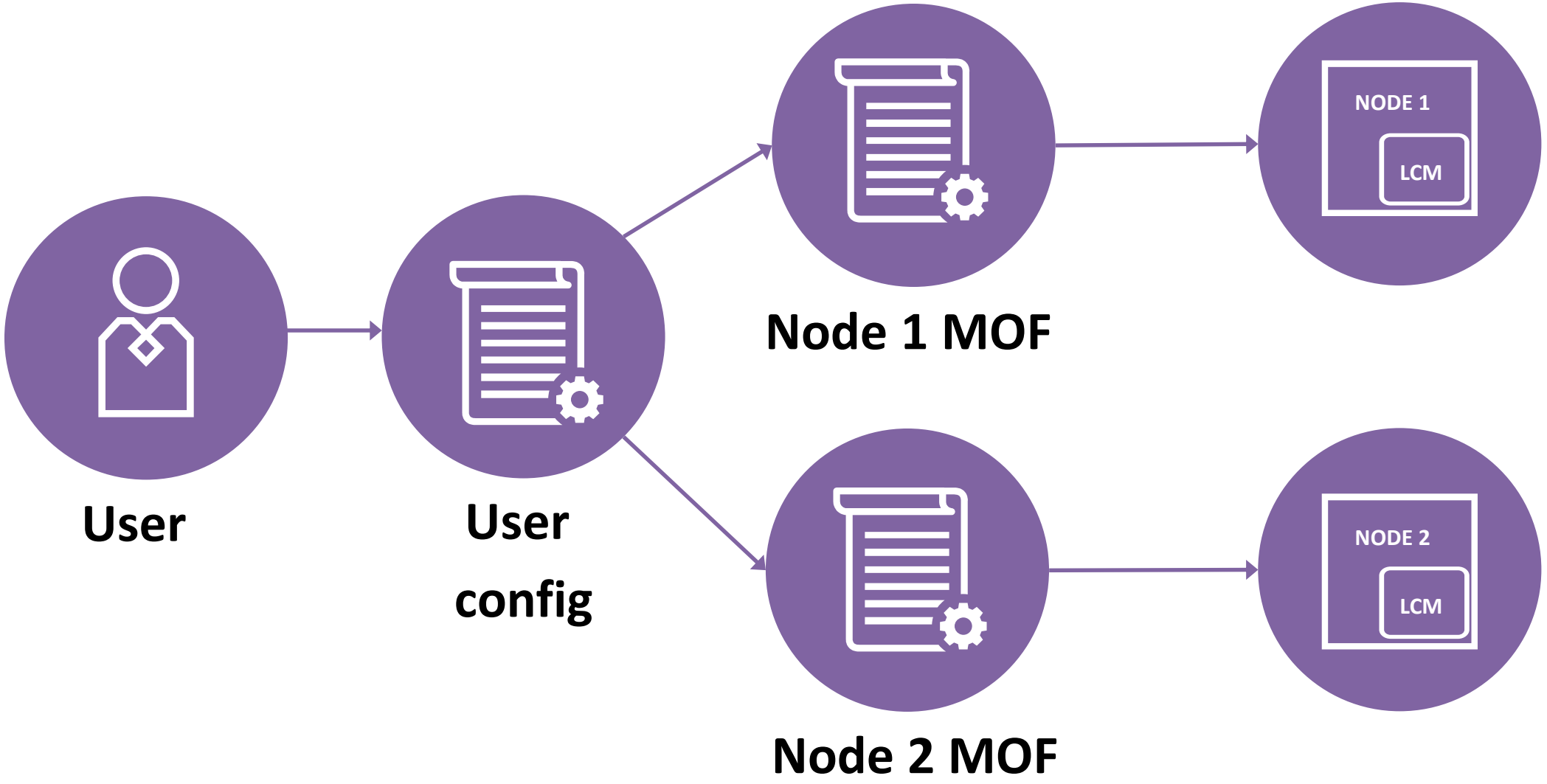
What is PowerCLI?

- VMware's command-line and scripting tool built on Windows PowerShell
- Features more than 700 cmdlets for managing and automating vSphere, vCloud, and Horizon environments
- One of the most robust and complete PowerShell deployments in the world



PowerShell Desired State Configuration

How DSC works



```
1  Configuration MyDscConfiguration {
2
3      Node @('localhost', 'server01') {
4          WindowsFeature MyFeatureInstance {
5              Ensure = 'Present'
6              Name = 'RSAT'
7          }
8
9          WindowsFeature MyFeatureInstance {
10             Ensure = 'Present'
11             Name = 'BitLocker'
12         }
13     }
14 }
15
16  MyDscConfiguration
```

DSC Resources for VMware

DSC Resources for VMware

- Introduced December of 2018: VMware.vSphereDSC
- Allows declarative language-based management against vCenter and ESXi Hosts
- Open Sourced: https://github.com/vmware/dscr-for-vmware
  - Contributions Welcome!

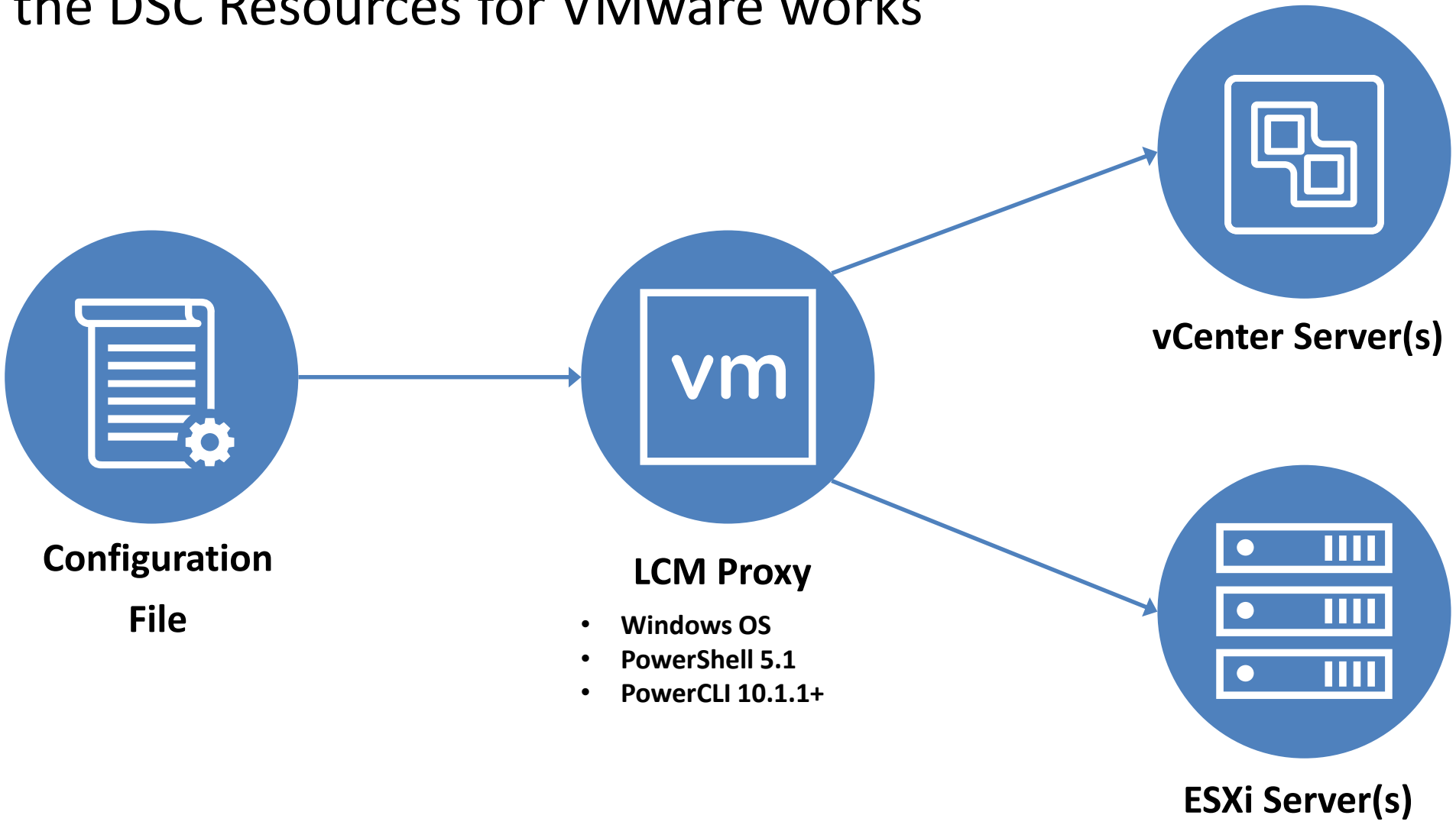
Dependencies:

- PowerShell 5.1
- PowerCLI 10.1.1 or newer

Requirements:

- Able to run PowerShell as Administrator
- PowerCLI and vSphere DSC installed for all users' access

How the DSC Resources for VMware works



```powershell
Configuration VMHostNtpSettings_Config {
    Import-DscResource -ModuleName VMware.vSphereDSC

    Node localhost {
        $Password = $Password | ConvertTo-SecureString -AsPlainText -Force
        $Credential = New-Object System.Management.Automation.PSCredential($User, $Password)

        VMHostNtpSettings vmHostNtpSettings {
            Name = $Name
            Server = $Server
            Credential = $Credential
            NtpServer = @("0.pool.ntp.org", "1.pool.ntp.org")
            NtpServicePolicy = "automatic"
        }
    }
}

VMHostNtpSettings_Config -ConfigurationData $script:configurationData
```

Currently Available

- vCenter Settings
  - Statistics
  - Logging Level
  - TaskMaxAge
  - EventMaxAge
  - Motd
  - Issue
- ESXi Host Settings
  - NTP
  - DNS
  - SATP Claim Rules
  - TPS
  - Motd
  - Issue
  - Syslog
  - Services
- PowerCLI Settings
  - Configuration

Work In Progress

- Standard Switch Resources
  - Security
  - Shaping
  - Teaming
- Cluster Resources
  - HA Settings
  - DRS Settings
- VMHost Resources
  - Advanced Settings
  - PCI Passthrough
  - Power Policies

Demo Time!

vSphere - vcsa01.corp.local - Sun

Not secure | https://vcsa01.corp.local/ui/#?extensionId=vsphere.core.folder.summary&objectId=urn:vmomi:Folder:group-d1:d9e1adc7-5304-4168-8419-230ac07c98d2&navigator=vsphere.core.viTree.hostsAndClustersView

vSphere Client | Menu | Search | Administrator@VSPHERE.LOCAL

vcsa01.corp.local

ACTIONS

Summary | Monitor | Configure | Permissions | Datacenters | Hosts & Clusters | VMs | Datastores | Networks | Linked vCe

- vcsa01.corp.local
  - Demo

Virtual Machines: 0
Hosts: 4

CPU — Free: 41.35 GHz
Used: 249 MHz — Capacity: 41.6 GHz

Memory — Free: 57.73 GB
Used: 6.27 GB — Capacity: 64 GB

Storage — Free: 0 B
Used: 0 B — Capacity: 0 B

Custom Attributes

Attribute	Value

Tags

Assigned Tag	Category	Description

Recent Tasks | Alarms

Task Name	Target	Status	Initiator	Queued For	Start Time	Completion Time	Server

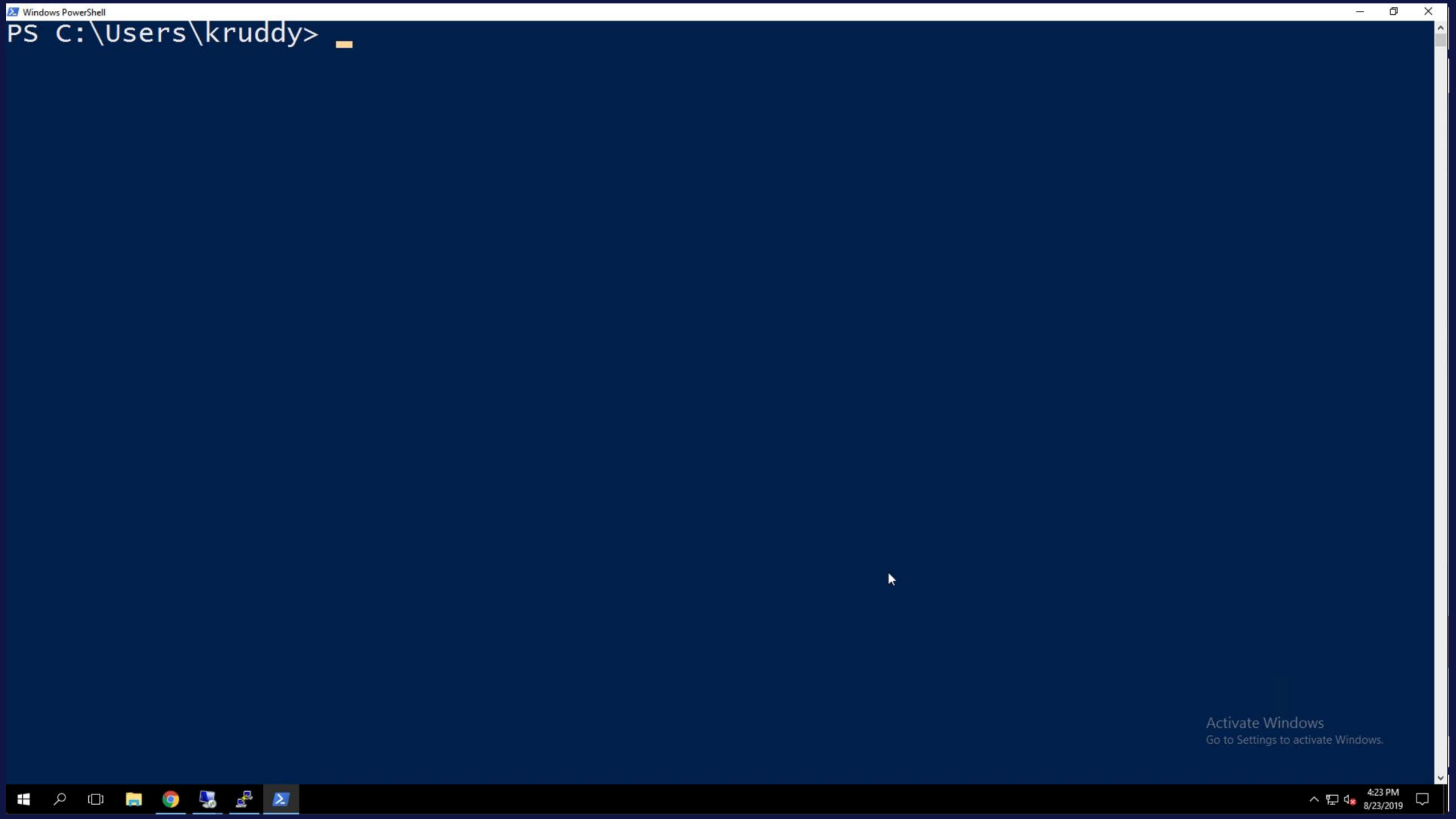
Activate Windows
Go to Settings to activate Windows.

All

More Tasks

6:28 PM
8/21/2019

What About Integrations?


Windows PowerShell
PS C:\Users\kruddy>
Activate Windows
Go to Settings to activate Windows.
4:23 PM
8/23/2019

DSC Resource Testing

DSC and Pester

- Why?
  - To validate new code is working
    - Unit Test
  - To validate it is behaving with other code
    - Integration Test
- How?
  - Pester
    - PS Standard
    - Intuitive (somewhat)

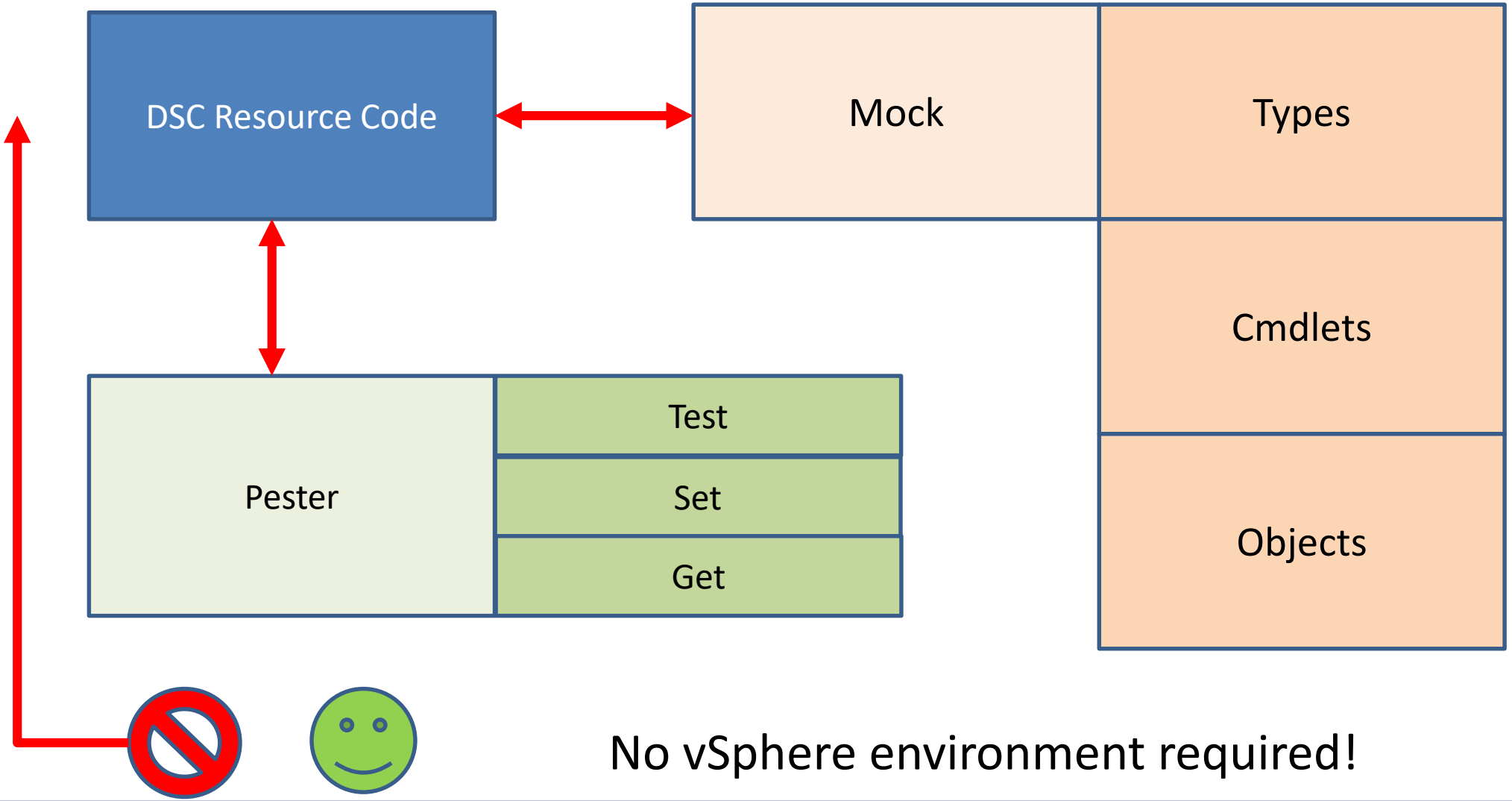
Unit Testing

- Piece of code
  - One DSC resource
  - One DSC resource function
  - Helpers
- Mocking
- Feed with all possibilities
- Verify result
  - Expected vs Reality

Integration Testing

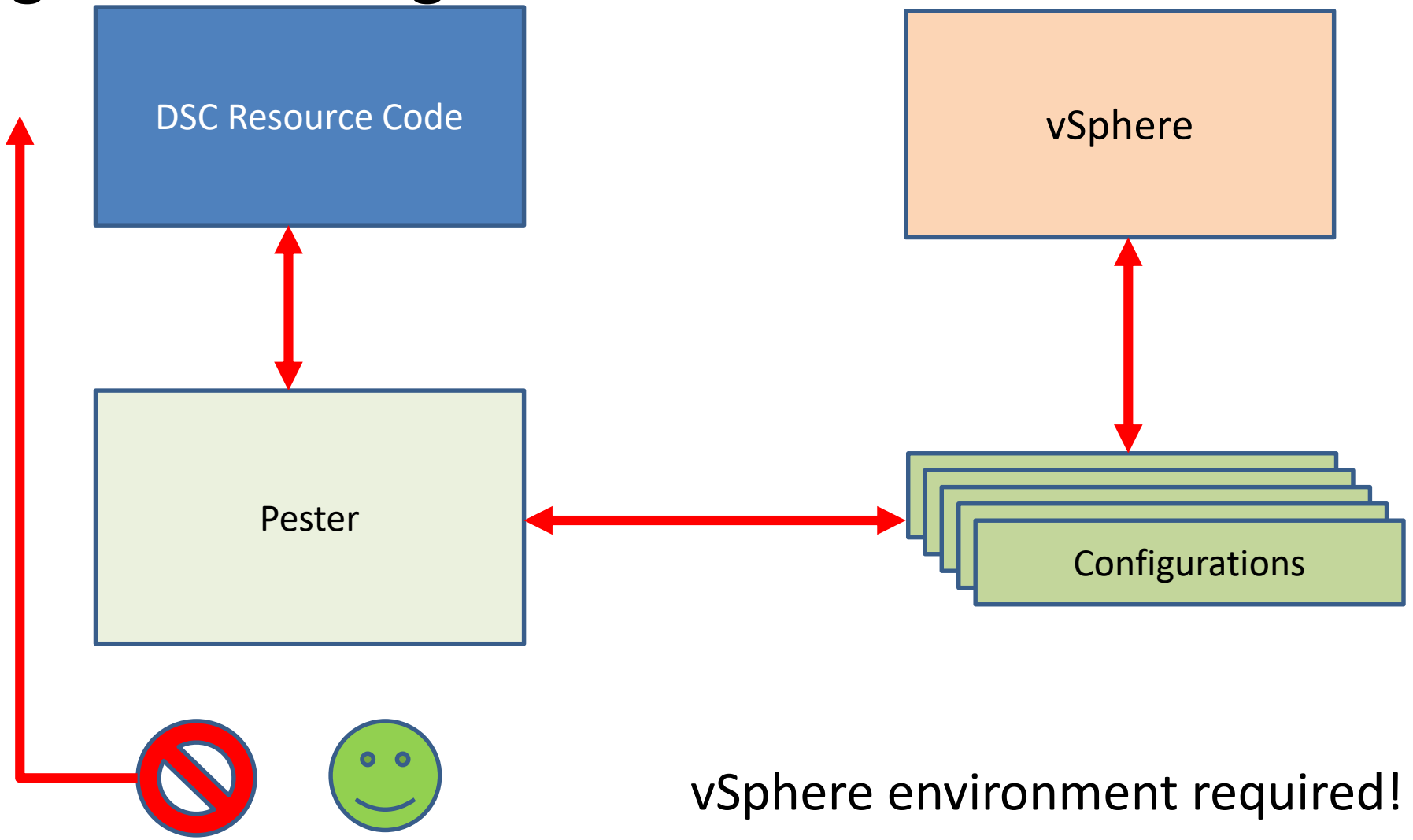
- How does it behave with others?
  - One DSC resource
  - Composite DSC resource
- Verify combinations
  - Expected vs Reality
- Environment
  - Set up
  - Part of pipeline
  - Cattle!

Unit Testing – How It Is Done

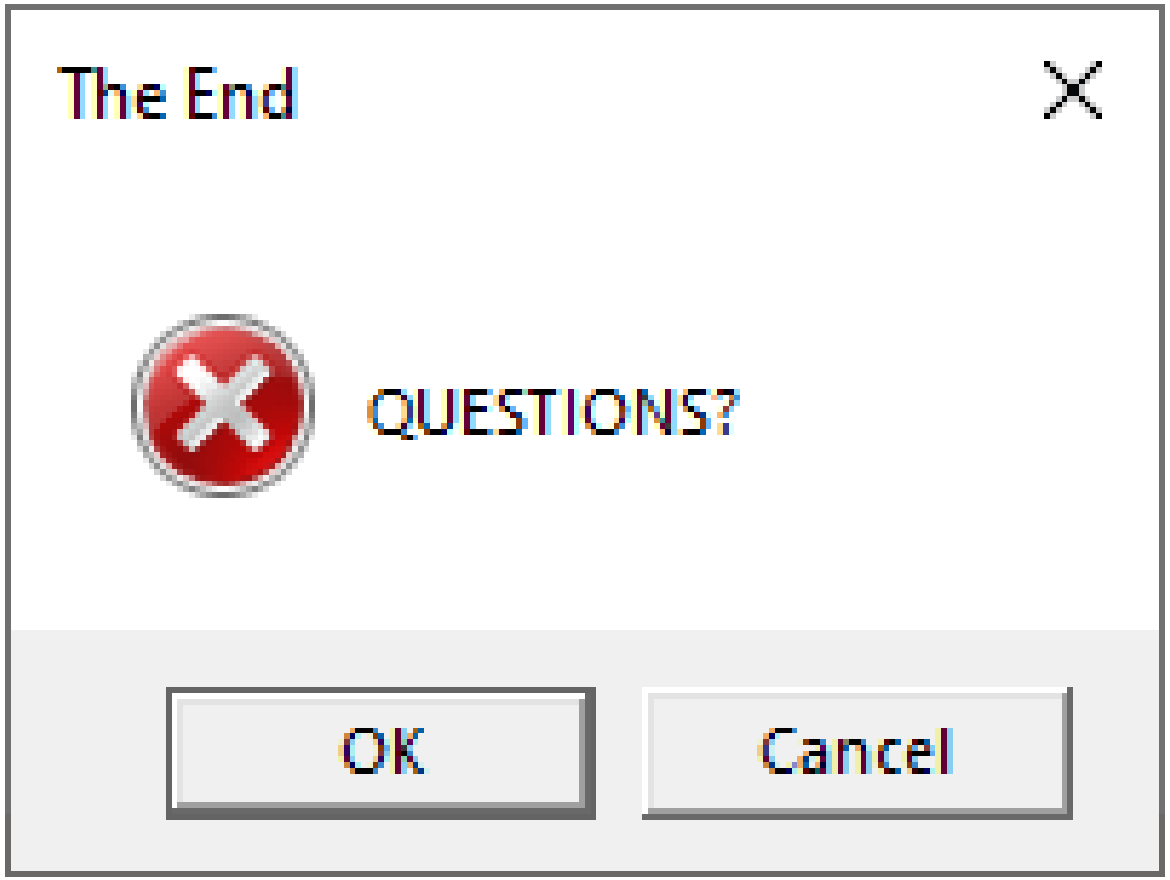


No vSphere environment required!

Integration Testing – How It Is Done



vSphere environment required!


The End
QUESTIONS?
OK
Cancel

t to rate the ses

https://my.eventraft.com/psconfasi

Opening Ceremony

Fri, 20 Sep, 08:45 AM - 09:00 AM

Beginner

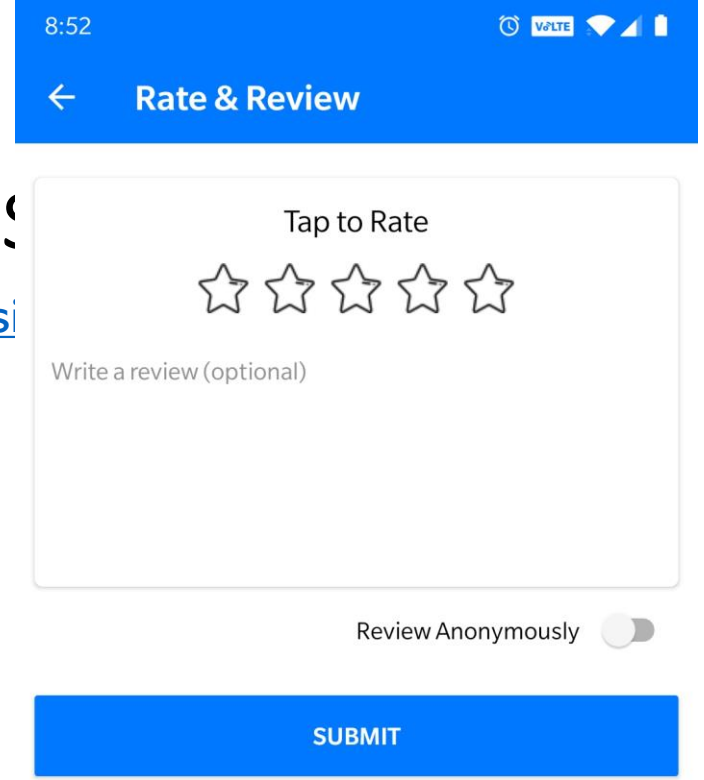
Common

0 Rating

Rate now >

Description

PowerShell Conference Asia 2019 Opening ceremony!



PSConf.Asia

Mock Types

- Generated CS file

```
namespace VMware.Vim
{
    namespace VMware.VimAutomation.ViCore.Types.V1
    {
        public interface VIContainer : InventoryItem
        {
        }
```

Mock VMware.VimAutomation.Core

```
Add-Type -Path "$($env:PSModulePath)/VMware.VimAutomation.Core/PowerCLITypes.cs"


function New-Folder {
    [CmdletBinding(DefaultParameterSetName = "__AllParameterSets")]
    param(
        [Parameter(Mandatory = $true, ParameterSetName = "__AllParameterSets", ValueFromPipeline = $false)]
        [string]
        $Name,

        [Parameter(Mandatory = $false, ParameterSetName = "__AllParameterSets", ValueFromPipeline = $true)]
        [VMware.VimAutomation.ViCore.Types.V1.Inventory.VIContainer]
        $Location,
```

Mock Objects

```
$script:datacenterHostFolder = [VMware.VimAutomation.ViCore.Impl.V1.Inventory.FolderImpl] @{
    Id = $script:constants.DatacenterHostFolderId
    Name = $script:constants.DatacenterHostFolderName
    Parent = [VMware.VimAutomation.ViCore.Impl.V1.Inventory.DatacenterImpl] @{
        Name = $script:constants.DatacenterName
    }
}
```

Actual Unit tests 1

```
using module VMware.vSphereDSC

$script:moduleName = 'VMware.vSphereDSC'

InModuleScope -ModuleName $script:moduleName {
    try {
        $unitTestsFolder = Join-Path (Join-Path (Get-Module VMware.vSphereDSC -
ListAvailable).ModuleBase 'Tests') 'Unit'
        $modulePath = $env:PSModulePath
        $resourceName = 'Folder'

        . "$unitTestsFolder\TestHelpers\TestUtils.ps1"

        # Calls the function to Import the mocked VMware.VimAutomation.Core module before all tests.
        Invoke-TestSetup

        . "$unitTestsFolder\TestHelpers\Mocks\MockData.ps1"
        . "$unitTestsFolder\TestHelpers\Mocks\FolderMocks.ps1"
```

Actual Unit tests 2

```
Describe 'Folder\Test' -Tag 'Test’ {
 BeforeAll {
   # Arrange
   New-MocksForFolder
 }

 Context 'Invoking with Ensure Present and non existing Folder’ {
  BeforeAll {
   # Arrange
    $resourceProperties = New-MocksWhenEnsurePresentAndNonExistingFolder
    $resource = New-Object -TypeName $resourceName -Property $resourceProperties
  }

  It 'Should return $false when the Folder does not exist’ {
   # Act
   $result = $resource.Test()

   # Assert
   $result | Should -Be $false
  }
 }
```

Configurations

```
Configuration Folder_WhenAddingFolderWithEmptyLocation_Config {
    Import-DscResource -ModuleName VMware.vSphereDSC

    Node $AllNodes.NodeName {
        Datacenter $script:datacenterWithEmptyLocationResourceName {
            Server = $Server
            Credential = $script:viServerCredential
            Name = $script:datacenterName
            Location = $script:datacenterEmptyLocation
            Ensure = 'Present'
        }

        Folder $script:folderWithEmptyLocationResourceName {
            Server = $Server
            Credential = $script:viServerCredential
            Name = $script:folderName
            Location = $script:folderWithEmptyLocation
            DatacenterName = $script:datacenterName
            DatacenterLocation = $script:datacenterEmptyLocation
            Ensure = 'Present'
            FolderType = $script:folderType
            DependsOn = $script:datacenterWithEmptyLocationResourceId
        }
    }
}
```

Integration 1

```
$script:dscResourceName = 'Folder'
$script:moduleFolderPath = (Get-Module VMware.vSphereDSC -ListAvailable).ModuleBase
$script:integrationTestsFolderPath = Join-Path (Join-Path $moduleFolderPath 'Tests') 'Integration'
$script:configurationFile = `
  "$script:integrationTestsFolderPath\Configurations\$($script:dscResourceName)\$($script:dscResourceName)_Config.ps1"

$script:configWhenAddingFolderWithEmptyLocation = "$($script:dscResourceName)_WhenAddingFolderWithEmptyLocation_Config"
```

Integration 2

```
Describe "$($script:dscResourceName)_Integration" {
    Context "When using configuration $($script:configWhenAddingFolderWithEmptyLocation)" {
        BeforeAll {
            # Arrange
            $startDscConfigurationParameters = @{
                Path = $script:mofFileWhenAddingFolderWithEmptyLocationPath
#
            }

            # Act
            Start-DscConfiguration @startDscConfigurationParameters
        }

        It 'Should compile and apply the MOF without throwing' {
            # Arrange
            $startDscConfigurationParameters = @{
                Path = $script:mofFileWhenAddingFolderWithEmptyLocationPath
                ComputerName = 'localhost’
#
            }

            # Act && Assert
            { Start-DscConfiguration @startDscConfigurationParameters } | Should -Not -Throw
        }
    }
}
```