PowerShell Desired State Configuration for Linux

By: Kolby Allen

August 19, 2015

LINUXCON North America

Outline

ISOutsource

- What is PowerShell and Desired State Configuration?
- Why would I use PowerShell for Linux?
- Review of Desired State Configuration
  - Server Setup
  - Client Setup
- Examples
  - Webserver Configuration
  - End to End Provisioning (Hyper-V to Webserver)
- Q/A

What is PowerShell?

ISOutsource

- PowerShell – task automation and management tool
  - Allow for both local and remote management
  - Scripting language
  - V1 shipped in 2006 as optional
  - V2 shipped with Win7 and Server 2008
  - V5 currently in preview (unless your on Win10)

Reboot Local Machine:

```
Restart-Computer
```

Reboot Remote Machine:

```
Restart-Computer -ComputerName Server01
```

Reboot Remote Machines (multiple):

```
Restart-Computer -ComputerName Server01, Server02, localhost
```

What is Desired State Configuration?

ISOutsource

- PowerShell Desired State Configuration was released to allow for deployment and management of configuration data
- Follows DMTF management standards and WS-Management Protocol

- Applications:
  - Server role/feature management
  - Setting/Changing registry items
  - Process/service management
  - User/Group Management
  - Software Deployment and setup
  - Remote script execution
  - Managing configuration drift
  - Actual configuration reporting

Why use DSC?

ISOutsource

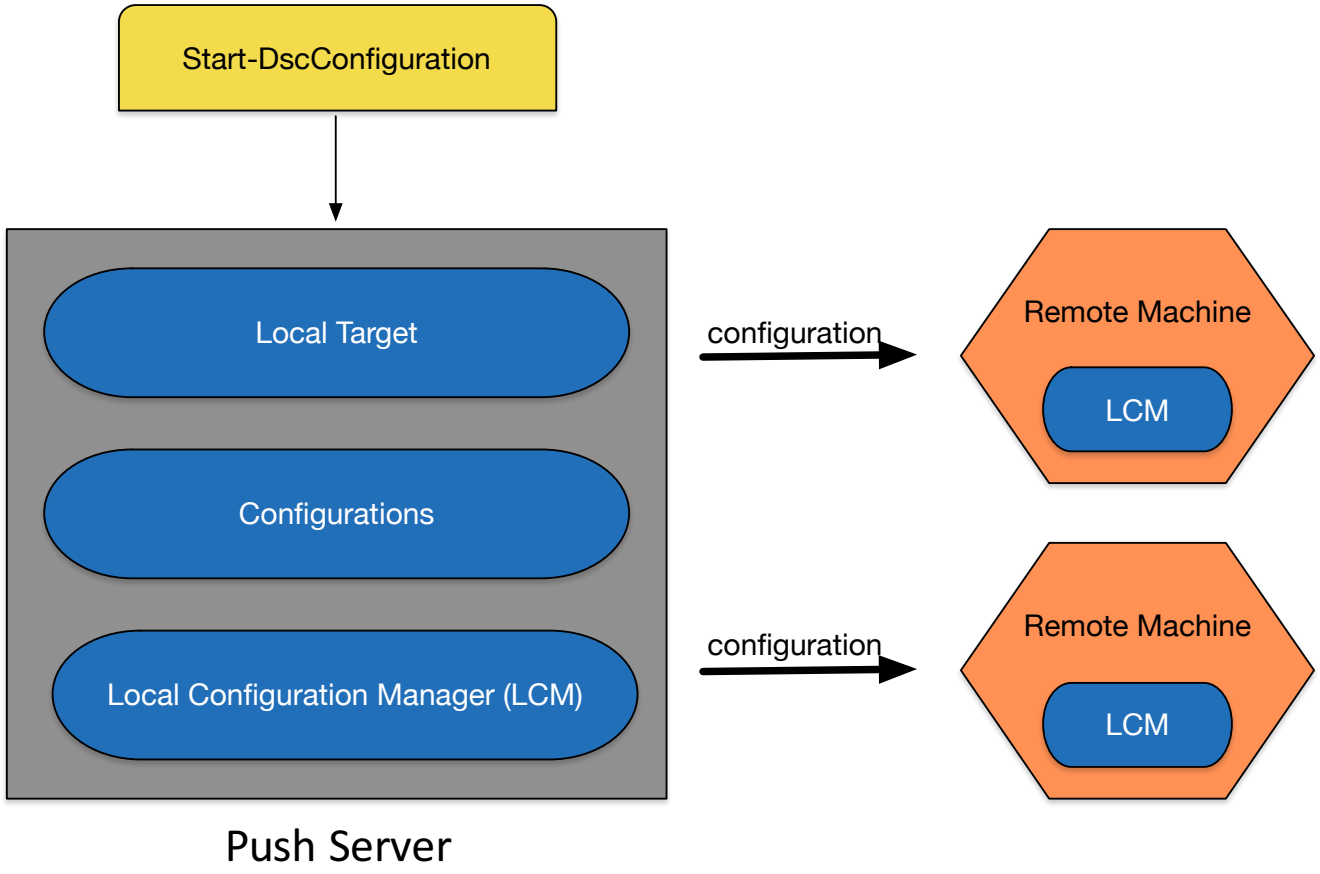
WHY NOT ONE OF THESE?

Why use DSC?

ISOutsource

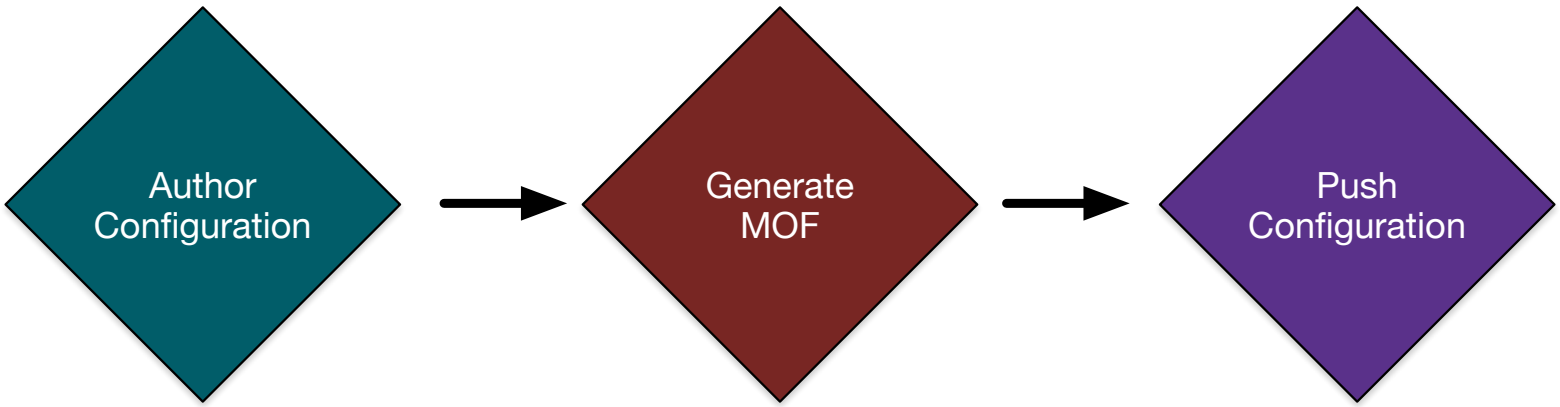
Because of this…

Review of DSC – Server Setup - PUSH



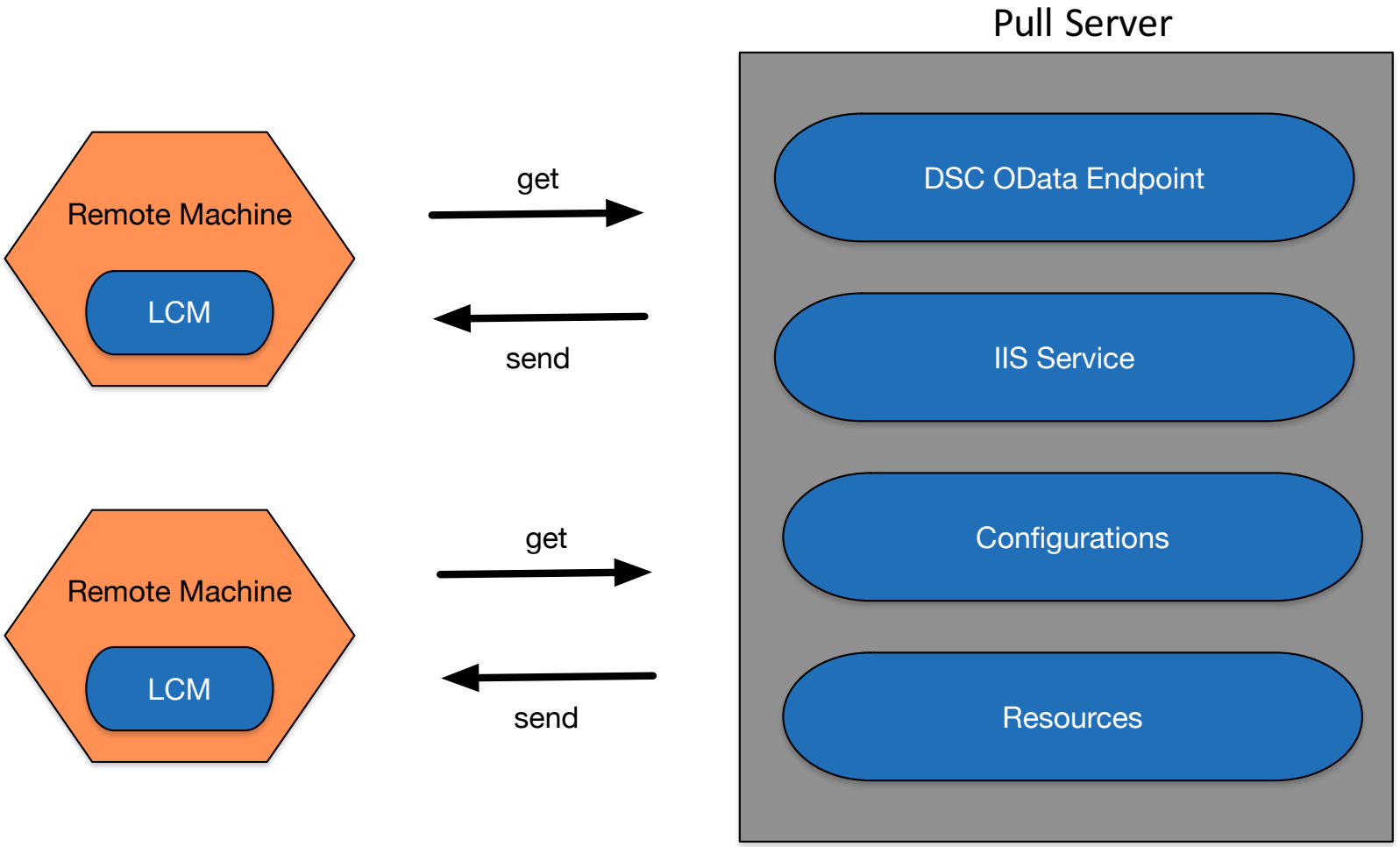
http://blogs.msdn.com/b/powershell/archive/2013/11/26/push-and-pull-configuration-modes.aspx

Review of DSC – Server Setup - PUSH



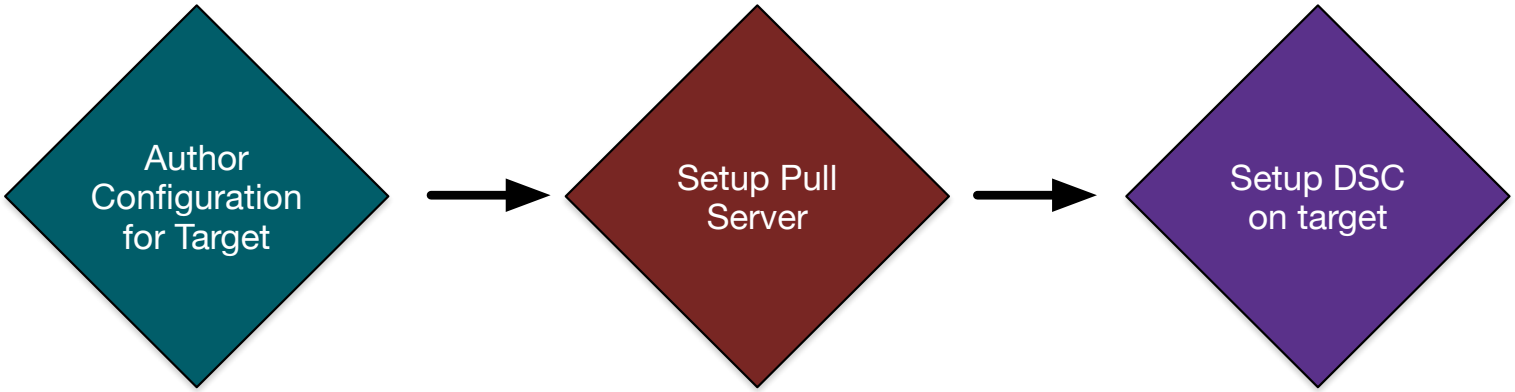
http://blogs.msdn.com/b/powershell/archive/2013/11/26/push-and-pull-configuration-modes.aspx

Review of DSC – Server Setup - PULL



http://blogs.msdn.com/b/powershell/archive/2013/11/26/push-and-pull-configuration-modes.aspx

Review of DSC – Server Setup - PULL



http://blogs.msdn.com/b/powershell/archive/2013/11/26/push-and-pull-configuration-modes.aspx

Support Operating Systems

- CentOS 5, 6, & 7
- Debian GNU/Linux 6 & 7
- Oracle Linux 5, 6, & 7
- RHEL 5, 6, & 7
- SUSE 10, 11, & 12
- Ubuntu 12.04 LTS & 14.04 LTS

Required Packages:

- glibc
- python
- omniserver - Open Management Infrastructure
- openssl
- ctypes
- libcurl

DSC for Linux Binaries

https://github.com/MSFTOSSMgmt/WPSDSCLinux

Review of DSC – Client Setup – LINUX Resources

DSC Resources – building blocks to DSC Configurations
Linux Resources

- nxArchive – allows un packing of tar, zip
- nxEnviroment – manage system variables
- nxFile – manages files and directory state
- nxFileLine – manages lines within Linux config
- nxScript – run script blocks on target nodes
- nxUser – manages linux users
- nxGroup – manages linux groups
- nxService – manages linux services (System-V, upstart, systemd)
- nxPackage – managing packages on the system
- nxAuthorizedKeys – manage ssh key's for specific user

https://technet.microsoft.com/en-us/library/mt126209.aspx
Currently in beta but growing.

Review of DSC – Client Setup – LINUX Resources - nxArchive

ISOutsource

```
nxArchive <string> #ResourceName {
    SourcePath = <string>
    DestinationPath = <string>
    [ Checksum = <string> { ctime | mtime | md5 }  ]
    [ Force = <bool> ]
    [ DependsOn = <string[]> ]
    [ Ensure = <string> { Absent | Present }  ]
}
```

For comparison

$true or $false

```
nxArchive SyncWebDir {
   SourcePath = “/usr/release/staging/website.tar”
   DestinationPath = “/usr/local/apache2/htdocs/”
   Force = $false
   DependsOn = "[nxFile]SyncArchiveFromWeb"
}
```

Review of DSC – Client Setup – LINUX Resources - nxEnviroment

ISOutsource

```
nxEnvironment <string> #ResourceName {
    Name = <string>
    [ Value = <string>
    [ Ensure = <string> { Absent | Present }  ]
    [ Path = <bool> }
    [ DependsOn = <string[]> ]

}
```

```
nxEnvironment EnvironmentExample {
    Ensure = “Present”
    Name = “TestEnvironmentVariable”
    Value = “TestValue”
}
```

ISOutsource

```
nxFile <string> #ResourceName {
    DestinationPath = <string>
    [ SourcePath = <string> ]
    [ Ensure = <string> { Absent | Present }  ]
    [ Type = <string> { directory | file | link } ]
    [ Contents = <string> ]
    [ Checksum = <string> { ctime | mtime | md5 }  ]
    [ Recurse = <bool> ]
    [ Force = <bool> ]
    [ Links = <string> { follow | manage } ]
    [ Group = <string> ]
    [ Mode = <string> ]
    [ Owner = <string> ]
    [ DependsOn = <string[]> ]

}
```

ISOutsource

From HTTP/HTTPS/FTP

```
nxFile resolvConf {
    SourcePath = "http://10.185.85.11/conf/resolv.conf"
    DestinationPath = "/etc/resolv.conf"
    Mode = "644"
    Type = "file"

}
```

From Windows Local Path

```
$OFS = "`n"
$Contents = Get-Content C:\temp\resolv.conf

nxFile resolvConf {
    DestinationPath = "/etc/resolv.conf"
    Mode = "644"
    Type = "file"
    Contents = "$Contents"
}
```

Review of DSC – Client Setup – LINUX Resources - nxFileLine

ISOutsource

```
nxFileLine <string> #ResourceName {
    FilePath = <string>
    ContainsLine = <string>
    [ DoesNotContainPattern = <string> ]
    [ DependsOn = <string[]> ]

}
```

```
nxFileLine DoNotRequireTTY {
   FilePath = “/etc/sudoers”
   ContainsLine = 'Defaults:monuser !requiretty'
   DoesNotContainPattern = "Defaults:monuser[ ]+requiretty"
}
```

```
nxUser <string> #ResourceName {
    UserName = <string>
    [ Ensure = <string> { Absent | Present }  ]
    [ FullName = <string> ]
    [ Description = <string> ]
    [ Password = <string> ]
    [ Disabled = <bool> ]
    [ PasswordChangeRequired = <bool> ]
    [ HomeDirectory = <string> ]
    [ Mode = <string> ]
    [ GroupID = <string> ]
    [ DependsOn = <string[]> ]

}
```

Password: The hash of the users password in the appropriate form for the Linux computer.

Review of DSC – Client Setup – LINUX Resources -nxGroups

ISOutsource

```
nxGroup <string> #ResourceName {
    GroupName = <string>
    [ Ensure = <string> { Absent | Present }  ]
    [ Members = <string[]> ]
    [ MebersToInclude = <string[]>]
    [ MembersToExclude = <string[]> ]
    [ DependsOn = <string[]> ]

}
```

Review of DSC – Client Setup – LINUX Resources – nxUsers/nxGroups

ISOutsource

```
nxUser UserExample{
   UserName = "monuser"
   Description = "Monitoring user"
   Password  =
'$6$fZAne/Qc$MZejMrOxDK0ogv9SLiBP5J5qZFBvXLnDu8HY1Oy7ycX.Y3C7mGPUfeQy3A82ev
3zIabhDQnj2ayeuGn02CqE/0'
   Ensure = "Present"
   HomeDirectory = "/home/monuser"
}

nxGroup GroupExample{
   GroupName = "DBusers"
   Ensure = "Present"
   MembersToInclude = "monuser"
   DependsOn = "[nxUser]UserExample"
}
```

Review of DSC – Client Setup – LINUX Resources - nxPackage

ISOutsource

```
nxPackage <string> #ResourceName {
    Name = <string>
    [ Ensure = <string> { Absent | Present }  ]
    [ PackageManager = <string> { Yum | Apt | Zypper } ]
    [ PackageGroup = <bool>]
    [ Arguments = <string> ]
    [ ReturnCode = <uint32> ]
    [ LogPath = <string> ]
    [ DependsOn = <string[]> ]

}
```

```
nxPackage httpd {
    Name = "httpd"
    Ensure = "Present"
    PackageManager = "Yum"
}
```

Review of DSC – Client Setup – LINUX Resources - nxScript

ISOutsource

```
nxScript <string> #ResourceName {
    GetScript = <string>
    SetScript = <string>
    TestScript = <string>
    [ User = <string> ]
    { Group = <string> ]
    [ DependsOn = <string[]> ]

}
```

GetScript: Provides the script that runs when Get-DscConfiguration is called
SetScript: The script to run
TestScript: Provides the script for Start-DscConfiguration
exit 0: SetScript will not run
exit 1: SetScript runs

Review of DSC – Client Setup – LINUX Resources - nxScript

ISOutsource

```
nxScript KeepDirEmpty{

    GetScript = @"
#!/bin/bash
ls /tmp/mydir/ | wc -l
"@

    SetScript = @"
#!/bin/bash
rm -rf /tmp/mydir/*
"@

    TestScript = @'
#!/bin/bash
filecount=`ls /tmp/mydir | wc -l`
if [ $filecount -gt 0 ]
then
    exit 1
else
    exit 0
fi
'@
}
```

Review of DSC – Client Setup – LINUX Resources - nxService

ISOutsource

```
nxService <string> #ResourceName {
    Name = <string>
    [ Controller = <string> { init | upstart | system }  ]
    [ Enambled = <bool> ]
    [ State = <string> { Running | Stopped } ]
    [ DependsOn = <string[]> ]

}
```

```
nxService ApacheService {
     Name = "httpd"
     State = "running"
     Enabled = $true
     Controller = "systemd"
}
```

Review of DSC – Client Setup – LINUX Resources - nxAuthorizedKeys

ISOutsource

```
nxAuthorizedKeys <string> #ResourceName {
    KeyComment = <string>
    [ Ensure = <string> { Absent | Present }  ]
    [ Username = <string> ]
    [ Key = <string> ]
    [ DependsOn = <string[]> ]

}
```

```
nxSshAuthorizedKeys myKey{
   KeyComment = "myKey"
   Ensure = "Present"
   Key = 'ssh-rsa
AAAAB3NzaC1yc2EAAAABJQAAAQEA0b+0xSd07QXRifm3FXj7Pn/DblA6QI5VAkDm6OivFzj3U6q
GD1VJ6AAxWPCyMl/qhtpRtxZJDu/TxD8AyZNgc8aN2CljN1hOMbBRvH2q5QPf/nCnnJRaGsrxIq
ZjyZdYo9ZEEzjZUuMDM5HI1LA9B99k/K6PK2Bc1NLivpu7nbtVG2tLOQs+GefsnHuetsRMwo/+c
3LtwYm9M0XfkGjYVCLO4CoFuSQpvX6AB3TedUy6NZ0iuxC0kRGg1rIQTwSRcw+McLhslF0drs33
fw6tYdzlLBnnzimShMuiDWiT37WqCRovRGYrGCaEFGTG2e0CN8Co8nryXkyWc6NSDNpMzw==
rsa-key-20150401'
   UserName = "monuser"
}
```

DSC Examples – DSC Configuration

```
1  configuration Name{
2      node("Node1", "Node2", "Node3"){
3
4          WindowsFeature FriendlyName{
5              Ensure = "Present"
6              Name = "Feature Name"
7          }
8
9          File FriendlyName{
10             Ensure = "Present"
11             SourcePath = $SourcePath
12             DestinationPath = $DestinationPath
13             Type = "Directory"
14             DependsOn = "[WindowsFeature]FriendlyName"
15         }
16     }
17 }
18
19
```

DSC Examples – DSC Configuration – Windows WebServer

ISOutsource

```
configuration FourthCoffee {
    Import-DscResource -Module xWebAdministration

    # Install the IIS role
    WindowsFeature IIS {
        Ensure          = "Present"
        Name            = "Web-Server"
    }

    # Copy the website content
    File WebContent  {
        Ensure          = "Present"
        SourcePath      = "C:\Program Files\WindowsPowerShell\Modules\xWebAdministration\BakeryWebsite"
        DestinationPath = "C:\inetpub\FourthCoffee"
        Recurse         = $true
        Type            = "Directory"
        DependsOn       = "[WindowsFeature]IIS"
    }

    # Create a new website
    xWebsite BakeryWebSite   {
        Ensure          = "Present"
        Name            = "FourthCoffee"
        State           = "Started"
        PhysicalPath    = "C:\inetpub\FourthCoffee"
        DependsOn       = "[File]WebContent"
    }
}

```

DEMO

DSC Examples – End to End Provisioning

ISOutsource

- Most corporate environments use Hyper-V or VMWare
- Hyper-V has DSC Resources (form MS)
- VMWare Powershell tools (requires custom DSC Resource)
- Just requires that you have built a VM with all pre-reqs
  - Hyper-V and PS DSC for Linux support the same operating services
- Once VM is deployed you can use DSC push server to do initial setup

DSC Examples – End to End Provisioning – Hyper-V

ISOutsource

```
Configuration HyperV_VM {

    param (
        [Parameter(Mandatory)]
        [string]$VMName,

        [Parameter(Mandatory)]
        [string]$baseVhdPath,

        [Parameter(Mandatory)]
        [string]$ParentPath,

        [Parameter(Mandatory)]
        [string]$VMSwitchName
    )

    Import-DscResource -module xHyper-V

    xVMSwitch switch {
        Name = $VMSwitchName
        Ensure = 'Present'
        Type = 'Internal'
    }

    xVHD DiffVHD {
        Ensure = 'Present'
        Name = $VMName
        Path = $baseVhdPath
        ParentPath = $ParentPath
        Generation = 'vhdx'
    }

    xVMHyperV CreateVM {
        Name = $VMName
        SwitchName = $VMSwitchName
        VhdPath = Join-Path -Path $baseVhdPath -ChildPath "$VMName.vhdx"
        ProcessorCount = 1
        MaximumMemory = 2GB
        MinimumMemory = 512MB
        RestartIfNeeded = 'True'
        DependsOn = '[xVHD]DiffVHD', '[xVMSwitch]switch'
        State = 'Running'
        Generation = 'vhdx'
    }
}
```

http://mikefrobbins.com/2015/01/22/creating-hyper-v-vms-with-desired-state-configuration/

Review of DSC – Conclusion

ISOutsource

- DSC Provides a new set of tools for machine configuration
- Allows integration with current Windows SysAdmin toolset
- Hyper-V and VMWare end to end provision capabilities
- A new tool to manage servers

THANK YOU

@kolbyallen

https://github.com/allenk1/Linux-DSC-Demo