Advanced DSC Configurations

Jeff Hicks
Author/Teacher

@jeffhicks | https://jdhitsolutions.com/blog

Leverage PowerShell

A DSC configuration is a PowerShell command type

Defined in a PowerShell script file

You can leverage your PowerShell scripting skills

Avoid hard-coding values

Create a single MOF per server

```
Configuration MyServers {

 Param(
  [string[]]$Computername,
  [switch]$CreateReports
)

Node $Computername {
   ...
 }
```

Leverage PowerShell

The same configuration will be created for all computers in $Computername
You might use –CreateReports on some servers

```
Node $Computername {
 if ($CreateReports) {
  File Reports {
    DestinationPath = “C:\Reports"
    Ensure          = "Present"
    Type            = "Directory"
  }

} #if create reports
```

Leverage PowerShell

Use PowerShell scripting logic

```
PS C:\> help myservers

NAME
    MyServers

SYNTAX
    MyServers [[-InstanceName] <string>] [[-DependsOn] <string[]>] [[-PsDscRunAsCredential] <pscredential>]
    [[-OutputPath] <string>] [[-ConfigurationData] <hashtable>] [[-Computername] <string[]>] [-CreateReports]
    [<CommonParameters>]


ALIASES
    None


REMARKS
    None
```

Parameters displayed in help

```
Node $Computername {
 Import-Csv .\features.csv |
   ForEach-Object {
    WindowsFeature ($_.Feature).Replace("-", "") {
     Ensure = $_.ensure
     Name = $_.Feature
     IncludeAllSubFeature = ([int]$_.Includeall) -as [bool]
    }
}
```

Leverage PowerShell

Import data using PowerShell
Use your scripting experience
There are better ways of using external data

Using Configuration Data

Store node customizations in a configuration data file

Pass the path to the file

- ConfigurationData parameter
- Use Import-PowerShellDataFile to test

The configuration consumes the configuration data

One configuration can create multiple unique MOFs based on data

```
@{
   AllNodes = @(@{})
```

Configuration Data

AllNodes is an array of hashtables

```
@{
   AllNodes = @(
     @{
         NodeName = "*“
         MaxLogSize = 1GB
      }
   )
```

Configuration Data

AllNodes is an array of hashtables
You can define an entry to apply all nodes

```
@{
  AllNodes = @(
    @{
       NodeName = "*"
       MaxLogSize = 1GB
    },
     @{
       NodeName = "SRV1"
       AddFeatures = "NLB"
  )
```

Configuration Data

AllNodes is an array of hashtables
You can define an entry to apply all nodes
Define a node entry for each server

```
@{
   AllNodes = @(
     @{..},
      @{..}
   )
   NonNodeData = @{
     Domain = "Company“
 }
}
```

Configuration Data

You can also define non-node data as a hashtable

I tend to use static configuration data stored in .psd1 files, but you can create them anyway you want, even using PowerShell

```
Configuration CompanyServer {
   Import-DscResource -ModuleName PSDesiredStateConfiguration -ModuleVersion 1.1
   Import-DscResource -ModuleName ComputerManagementDSC -ModuleVersion 8.5.0

   Node $AllNodes.where({$true}).NodeName {
       LocalConfigurationManager {
           ...
       }
```

Using Configuration Data

Dynamically define nodes in your configuration

```
Configuration CompanyServer {
   Import-DscResource -ModuleName PSDesiredStateConfiguration -ModuleVersion 1.1
   Import-DscResource -ModuleName ComputerManagementDSC -ModuleVersion 8.5.0

   Node $AllNodes.where({$true}).NodeName {
       LocalConfigurationManager {
           ...
       }
       WindowsEventLog Security {
           LogName            = "Security"
           MaximumSizeInBytes = $node.MaxSecurityLog
           IsEnabled          = $True
       }
```

Using Configuration Data

Dynamically define nodes in your configuration
Use values from configuration data

Credentials

Some resources may require a credential

- Local user account

Or run configuration not as SYSTEM

- PSDscRunAsCredential

Credential will be stored in the MOF

```
AllNodes     = @(
        @{
            NodeName                        = "*"
            PSDscAllowPlainTextPassword = $true
            PSDscAllowDomainUser        = $true
        },
```

Use PlainText Passwords

Use configuration data
For testing or development

```
$Secure = ConvertTo-SecureString -String $configurationdata.nonNodeData.TestPassword
-AsPlainText -Force
$HelpDeskCredential = New-Object -TypeName Pscredential -ArgumentList HelpDesk, $secure
```

Use Plain Text Passwords

The DSC resource may still need a PSCredential object

```
User HelpDesk {
    UserName                = "HelpDesk"
    Ensure                  = "Present"
    Description             = "HelpDesk User Account"
    PasswordNeverExpires    = $True
    Password                = $HelpDeskCredential
}
```

Use Plain Text Passwords

The DSC resource may still need a PSCredential object

```
instance of MSFT_Credential as $MSFT_Credential1ref
{
 Password = "P@ssw0rd";
 UserName = "HelpDesk";

};
```

Use Plain Text Passwords

WSMan still encrypts network traffic
Protect your configuration scripts and MOFs

Secure Credentials

Encrypt credentials with certificate

- Document Encryption
- Install certificate following your procedures

Store public key on authoring desktop

- Export server certificate
- Install locally

Use certificate thumbprint in the configuration

- Store in configuration data

```
Invoke-Command -scriptblock {
    Get-Childitem Cert:\LocalMachine\my |
    Where-Object { $_.EnhancedKeyUsageList.FriendlyName -contains "Document Encryption"
-AND $_.notAfter -gt (Get-Date) }
} -computer SRV1,SRV2
```

Find Certificates

You may have other ways to retrieve certificate details

```
   PSParentPath: Microsoft.PowerShell.Security\Certificate::LocalMachine\my

Thumbprint                                Subject                                 PSComputerName
----------                                -------                                 --------------
9F24C541113C1940E22B4436720E3E0496AD60DA  CN=SRV1.Company.Pri                      SRV1
5940D7352AB397BFB2F37856AA062BB471B43E5E  CN=Virtual Engine Test Lab DSC Clie...  SRV1
5940D7352AB397BFB2F37856AA062BB471B43E5E  CN=Virtual Engine Test Lab DSC Clie...  SRV2
4822355C555B1A1FA86705FC08F327304D4CA351  CN=SRV2.Company.Pri                      SRV2
```

Find Certificates

Need to use the thumbprint

```
 @{
    NodeName        = "SRV2"
    Role            = "Dev"
    CertificateFile = "C:\certs\srv2.cer"
    Thumbprint      = "4822355C555B1A1FA86705FC08F327304D4CA351"
}
```

Using Certificates

Specify the exported certificate
Add the thumbprint

```
LocalConfigurationManager {
    RebootNodeIfNeeded = $True
    ConfigurationMode  = "ApplyAndAutoCorrect"
    ActionAfterReboot  = "ContinueConfiguration"
    RefreshMode        = "Push"
    CertificateID      = $node.thumbprint
}
```

Using Certificates

Configure the LCM to use the same thumbprint
Or hard-code the value per node
Set the LocalConfigurationManager before deploying the configuration

```
Configuration CompanyServerSecure {

    Param([pscredential]$HelpDeskCredential = "HelpDesk")
...
```

Using Credentials

Prompt for a credential
I'm using a default for expediency

```
instance of MSFT_Credential as $MSFT_Credential1ref
{
Password = "-----BEGIN CMS-----
\nMIIB5AYJKoZIhvcNAQcDoIIB1TCCAdECAQAxggGMMIIBiAIBADBwMFkxEDAOBgNVBAoTB0NvbXBh\nbnkxEDA
OBgNVBAgTB0FyaXpvbmExEDAOBgNVBAcTB1Bob2VuaXgxCzAJBgNVBAYTAlVTMRQwEgYD\nVQQDEwtDb21wYW55
LlByaQITHQAAAAute9pvvSnVlgAAAAAACzANBgkqhkiG9w0BAQcwAASCAQBw\nH7VC+DUHXcqp1Y5iyWTTs7mCu
X7dcUC28tPwGOvrVLXzPcI2H9xmiA8Ys2VJl1Y9lQZCVV4NS2AC\nGmm77qS9wZJAYVVn0e3l57C4wzYlKnJ+x/
S8tzrJ5vIPf3IGJntsHlRjRoDf3NliIjtA87lNwjtc\n7F59B2CIPFbl6iUcZQL8w0CoQW+NesPYVe8tdjdxHAW
Lw/YChZ202L/CW6cTk5hsH5j3TaeA5RVs\neJhEbPv7FOAQv0vtQ3Tls4o4LzIzEiI009ytjdVQqPxV4BXuXtAN
VRJOW79wYOPva2ESNeMHE27f\ntWscgYEqukWWOi0F1Z5rhjT477RGHuVZlGDiMDwGCSqGSIb3DQEHATAdBglgh
kgBZQMEASoEEEZ1\nTLkf6AgA1s93TBew4k2AEHH6wG5glXqfyAqEY01fOBA=\n-----END CMS-----";
 UserName = "HelpDesk";
};
```

Using Credentials

The MOF stores an encrypted credential
Can only be decrypted with the server's private key

DSC and Certificates

How you install, export, and manage certificates is up to you.

You could automate thumbprint and certificate process

- Temporarily store exported certs?

I'll show you where to use them, the how is up to you

- You could pass thumbprints as parameters
- Dynamically add to configuration data

Demo

Using Advanced Configurations