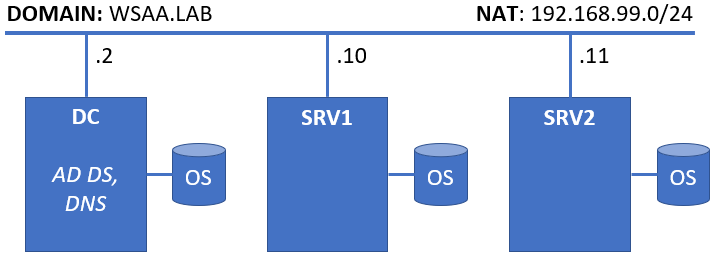
# Practice M8: Automated Configuration Management

For the purpose of this lab and the course, we will consider that we are working in a pure Windows environment either on-premise or in the cloud and using **Hyper-V** as a virtualization solution. All tasks can be achieved under different configuration (another host OS or virtualization solution) with the appropriate adjustments

The expected lab infrastructure consists of three machines. Windows Server 2019 Standard is enough, Desktop experience or core (managed remotely) – it is up to you

## Part 1: PowerShell DSC

The overall picture should look like:



### Explore PowerShell Modules

Log on to the **DC** with account that has administrative privileges

Open a **PowerShell** console

Execute the following command to get a list of where modules are stored:

**$env:PSModulePath -Split ";"**

Navigate to the system store:

**cd C:\Windows\system32\WindowsPowerShell\v1.0\Modules**

And check what modules exit there with:

**dir**

Navigate to:

**cd PSDesiredStateConfiguration\DSCResources**

And check what is there with

**dir**

Enter one of the folders. For example, **MSFT\_ServiceResource** and check what is there

**cd MSFT\_ServiceResource**

Open it with **PowerShell ISE**

**ise MSFT\_ServiceResource.psm1**

Explore the code

Close **PowerShell ISE** and return to the root folder

Check the help for **Get-DSCResource**

**Get-Help Get-DSCResource**

You may need to update the help files

Use it to list the available resources

**Get-DSCResource**

Now, let’s ask for a particular resource:

**Get-DSCResource Service**

Get all properties with:

**Get-DSCResource Service | Select \***

Check what are the required parameters and their values

**Get-DSCResource Service | Select -ExpandProperty Properties**

Now, let’s check how a resource is used

**Get-DSCResource Service -Syntax**

### Simple Localhost Configuration

Open **PowerShell ISE**

Create a new file and save it under **C:\DSC\Demo1.ps1**

Enter the following:

Configuration Demo1

{

Import-DscResource -ModuleName PsDesiredStateConfiguration

Node 'localhost'

{

File HelloWorld

{

DestinationPath = "C:\Temp\HelloWorld.txt"

Ensure = "Present"

Contents = "Hello World from DSC!"

}

}

}

Save it and execute it. You can execute it either by

* Executing in a **PowerShell** session the following:

**. C:\DSC\Demo1.ps1**

**Demo1**

* Or while in **PowerShell ISE** hit **F5** for the whole script or **F8** for the selection only

Let’s go with the first approach (don’t forget to open the session with administrative privileges)

To apply the configuration the **PowerShell DSC** way, you must execute first

**Start-DscConfiguration -Path C:\DSC\Demo1 -Verbose -Wait**

After the successful completion, we can test the configuration with

**Test-DscConfiguration -Path C:\DSC\Demo1**

Check what is inside the file with

**Get-Content -Path C:\Temp\HelloWorld.txt**

We can remove the file with

**Remove-Item -Path C:\Temp\HelloWorld.txt**

And test the configuration again

**Test-DscConfiguration -Path C:\DSC\Demo1**

### Simple Configuration

Create a new file and save it under **C:\DSC\Demo2.ps1**

Enter the following:

Configuration Demo2

{

Import-DscResource -ModuleName PsDesiredStateConfiguration

Node SRV1

{

File ReadMe

{

Ensure = "Present"

Contents = "This file was created with DSC."

Type = "File"

DestinationPath = "C:\Readme.txt"

}

}

}

Save it and execute it

**. C:\DSC\Demo2.ps1**

**Demo2**

Check what artefacts are created

Let’s test it, execute

**Test-DscConfiguration -Path .\Demo2**

It appears that the target machine is not in the desired state

Repeat the test but this time ask for more details

**Test-DscConfiguration -Path .\Demo2 -Verbose**

Apply the configuration with

**Start-DscConfiguration -Wait -Verbose -Path .\Demo2**

Let’s expand the configuration to cover the other node as well

Duplicate the **Node** block and adjust the name

Now, change the file to look like:

Configuration Demo2

{

Import-DscResource -ModuleName PsDesiredStateConfiguration

Node SRV1

{

File ReadMe

{

Ensure = "Present"

Contents = "This file was created with DSC."

Type = "File"

DestinationPath = "C:\Readme.txt"

}

}

Node SRV2

{

File ReadMe

{

Ensure = "Present"

Contents = "This file was created with DSC."

Type = "File"

DestinationPath = "C:\Readme.txt"

}

}

}

Save it. Execute it (both source it and generate the MOFs). Test it again with:

**Test-DscConfiguration -Path .\Demo2 -Verbose**

It appears that the first node is okay, but second it is not

Let’s apply it on both nodes

**Start-DscConfiguration -Wait -Verbose -Path .\Demo2**

Or instead, just on node **SRV2**

**Start-DscConfiguration -Wait -Verbose -Path .\Demo2 -ComputerName SRV2**

Let’s go with the first command

Pay attention to the **Skip Set** statement for **SRV1** versus the **Start Set** for **SRV2**

### Configuration Parameters

Copy the **C:\DSC\Demo2.ps1** file to **C:\DSC\Demo2P.ps1**

Open the new file for editing and change the content to this

Configuration Demo2P

{

Param ($server = 'SRV1')

Import-DscResource -ModuleName PsDesiredStateConfiguration

Node $server

{

File ReadMe

{

Ensure = "Present"

Contents = "This file was created with DSC."

Type = "File"

DestinationPath = "C:\Readme.txt"

}

}

}

Save and execute (source) the file

Create configuration only for **SRV2** with

**Demo2P -server SRV2**

If we execute it without parameters, it will build configuration for the default value – **SRV1**

### Separate Configuration and Data

Create a new file named **Demo3.psd1** and store it in **C:\DSC**

Enter the following:

@{

AllNodes = @(

@{ NodeName = "SRV1" },

@{ NodeName = "SRV2" }

)

}

Save it

Create a copy of the **Demo2.ps1** file and save it under **Demo3.ps1** name

Change it to match this one

Configuration Demo3

{

Import-DscResource -ModuleName PsDesiredStateConfiguration

ForEach($Node in $AllNodes)

{

Node $Node.NodeName

{

File ReadMe

{

Ensure = "Present"

Contents = "This file was created with DSC."

Type = "File"

DestinationPath = "C:\Readme.txt"

}

File Version

{

Ensure = "Present"

Contents = "2.0"

Type = "File"

DestinationPath = "C:\Version.txt"

}

}

}

}

Save it and execute (source) it

Generate the artefacts with

**Demo3 -ConfigurationData .\Demo3.psd1**

Test it first

**Test-DscConfiguration -Path .\Demo3 -Verbose**

And then apply it on both nodes

**Start-DscConfiguration -Wait -Verbose -Path .\Demo3**

### Configuration for Web Servers and Custom Start Page

Create new file named **Demo4.psd1** and store it in **C:\DSC**

Enter the following:

@{

AllNodes = @(

@{ NodeName = "SRV1" ; WindowsFeatures = @("Web-Server", "Web-Mgmt-Tools") },

@{ NodeName = "SRV2" ; WindowsFeatures = @("Web-Server") }

)

}

Save it

Create a copy of the **Demo3.ps1** file and save it under **Demo4.ps1** name

Change it to match this one

Configuration Demo4

{

Import-DscResource -ModuleName PsDesiredStateConfiguration

ForEach($Node in $AllNodes)

{

Node $Node.NodeName

{

ForEach($Feature in $Node.WindowsFeatures)

{

WindowsFeature $Feature

{

Name = $Feature

Ensure = "Present"

}

}

}

}

}

Save it and execute (source) it

Generate the artefacts

**Demo4 -ConfigurationData .\Demo4.psd1**

Let’s apply it on both nodes

**Start-DscConfiguration -Wait -Verbose -Path .\Demo4**

### Configuration Drift

Remove one of the components on one of the servers

Being on the **DC**, execute this:

**Invoke-Command -ComputerName SRV1 -ScriptBlock { Remove-WindowsFeature Web-Mgmt-Tools }**

Now, test **Demo4** again with

**Test-DscConfiguration -Path .\Demo4**

As it seems now the **SRV1** is not in the desired state

Let’s first check what are the default settings of the **Local Configuration Manager** on node **SRV1**

First create a session with

**$C = New-CimSession -ComputerName SRV1**

Then check the default settings

**Get-DscLocalConfigurationManager -CimSession $C**

As we can see, the default option of **ConfigurationMode** is **ApplyAndMonitor**

In order to force the machine to take self-correction actions, we must set the parameter to **ApplyAndAutoCheck**

Create a new file **C:\DSC\Demo5.psd1** with the following content:

@{

AllNodes = @(

@{ NodeName = "SRV1" },

@{ NodeName = "SRV2" }

)

}

Create a new file **C:\DSC\Demo5.ps1** with the following content:

[DSCLocalConfigurationManager()]

configuration Demo5

{

ForEach($Node in $AllNodes)

{

Node $Node.NodeName

{

Settings

{

ConfigurationMode = 'ApplyAndAutoCorrect'

ConfigurationModeFrequencyMins = 15

RefreshFrequencyMins = 30

}

}

}

}

Source the configuration

**. .\Demo5.ps1**

Create configuration files for both nodes

**Demo5 -ConfigurationData .\Demo5.psd1**

Apply the configuration with

**Set-DscLocalConfigurationManager -Path "C:\DSC\Demo5\"**

We can see that the changes were applied with

**Get-DscLocalConfigurationManager -CimSession $C**

After 15 minutes the node (**SRV1**) will alter its configuration to match the desired state

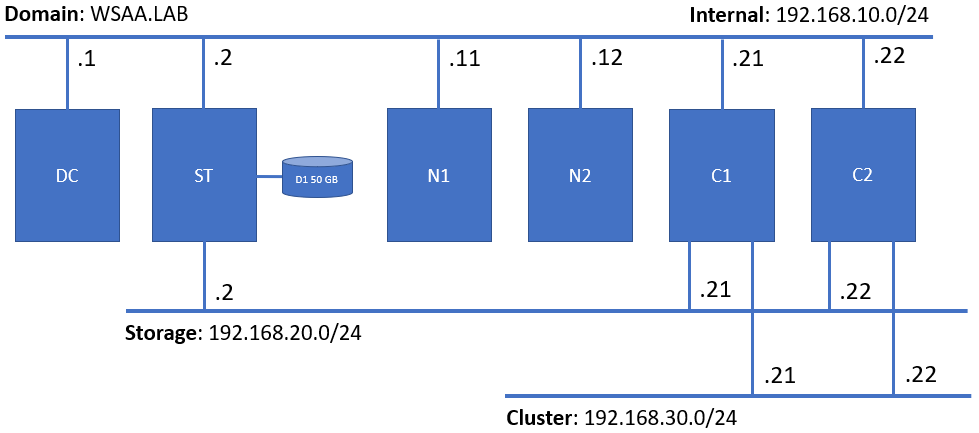
*Please note that both 15 and 30 minutes are set by default and are the lowest possible values for how often the configuration is checked and how often the pull server is contacted respectively*

## Part 2: Exam Preparation

On the actual practice exam, you will be presented with a challenge (set of tasks) like the one that follows

### Infrastructure

You will work in the following infrastructure:



### Proof

The result of your work will be checked in an automated fashion, so please be sure to follow **strictly** the stated names, paths, etc., otherwise you may not get points for some of tasks you consider as done

### Tasks checklist

#### Infrastructure [5 pts]

Considering the setup, and the tasks you are expected to fulfill, do some preliminary work:

* (T101 / 3 pts) Install **File Server** role on **C1** and **C2**
* (T102 / 2 pts) Install the following additional features from the **RSAT Tools** node – **NLB** and **Failover Cluster** including any related PowerShell modules on **DC**

#### Storage [13 pts]

Deploy a simple storage solution utilizing storage pools and **iSCSI technologies**:

* (T201 / 3 pts) Create a volume that occupies the entire spare disk (D1), format it with **NTFS** with allocation unit of **64KB** and assign it a drive letter **P**
* (T202 / 6 pts) Convert the **ST** machine to an **iSCSI target server** by installing and configuring the appropriate components
* (T203 / 2 pts) Create a new **dynamically** expanding iSCSI virtual disk **1 GB** in size and store it on drive P
* (T204 / 2 pts) Create a new **dynamically** expanding iSCSI virtual disk **15 GB** in size and store it on drive P

#### Network Load Balancing [12 pts]

Create and configure a two-node **NLB cluster** by completing the following tasks:

* (T301 / 3 pts) Prepare both **N1** and **N2** to become a part of an NLB cluster
* (T302 / 3 pts) Create a two-node **NLB cluster** named **NLB.WSAA.LAB** with IP address **192.168.10.44**
* (T303 / 1 pts) Configure the cluster mode to be **multicast**
* (T304 / 1 pts) Configure the rule for port **80/tcp** and affinity set to **none**
* (T305 / 2 pts) Create a DNS **A** record named **nlb** pointing to the IP address of the NLB cluster
* (T306 / 2 pts) Create a DNS **CNAME** record named **web** pointing to the FQDN of the **nlb** record

#### Failover Cluster [18 pts]

Create and configure **Windows Server Failover Cluster** that meets the following requirements:

* (T401 / 3 pts) Configure the initiator part on both **C1** and **C2** nodes
* (T402 / 4 pts) Initialize, format, and attach (where applicable) storage for quorum and data to both nodes
* (T403 / 2 pts) Install the required **failover cluster** role components
* (T404 / 3 pts) Create the cluster **SOFS.WSAA.LAB** with IP address set to **192.168.10.33**
* (T405 / 2 pts) Ensure that the storage is appropriately recognized and that there is a **cluster shared volume**
* (T406 / 2 pts) Configure a **Scale-Out File Server** cluster role named **SOFSRole**
* (T407 / 2 pts) Create a share named **SOFSShare** for the **SOFSRole**

#### Desired State Configuration [12 pts]

Create a configuration script using the **PowerShell DSC** techniques that meet the following requirements:

* (T501 / 1 pts) The script should be named **Exam.ps1** and must be stored in **C:\DSC** on the domain controller
* (T502 / 2 pts) Creates a **share** named **Exam** on the **domain controller** and pointing to folder **C:\Exam**
* (T503 / 2 pts) Creates **two** **index.html** files, **personalized** for every node of the NLB cluster, and **store them on the share**
* (T504 / 2 pts) Installs **IIS** on the two NLB nodes
* (T505 / 2 pts) Distributes **index.html** files from the share to the root folder of the default site on each node
* (T506 / 3 pts) Executes **successfully**