Installation Guide

General notes

- If you get a WinRM error, execute `vagrant plugin install vagrant-reload` followed by `vagrant reload WIN-XXX`. Replace `WIN-XXX` with the servername.
    - If you continue getting the error, open the VM in VirtualBox.
    - Log in using `VANLIEFFERINGE\Administrator` as username and `Admin2019` as password.
    - Browse to `C:\vagrant\provisioning\XXX`. Replace `XXX` with your servername.
    - Right-click the first script ande select `Edit`.
    - Press the green arrow at the top to run the script inside the VM.
    - Repeat this process for the other scripts.
- All accounts use `Admin2019` as password.
- Installation of the `WIN-DC1` forest and the `WIN-SQL-SCCM` general SCCM installation can take a while.

Install WIN-DC1

1. Execute `vagrant up WIN-DC1` in your vagrant directory.
2. When the scripts are done, open the VM in VirtualBox.
3. Select `Other User`. Use `VANLIEFFERINGE\Administrator` as username and `Admin2019` as password.
4. Open the Start Menu and search for `Server Manager`. Open the application.
5. Select `Manage > Add roles and features`.
6. Click `Next` Until the wizard asks for roles. Select the `Remote Access` role.
7. Continue. When the wizard asks for the wanted role services, select `Direct Access and VPN` and `Routing`.
8. Follow the wizard to install the role.
9. Close the Server Manager.
10. Restart the server.
11. Log in again using `VANLIEFFERINGE\Administrator` as username and `Admin2019` as password.
12. Open the Server Manager.
13. In the top right menu, select `Tools > Routing And Remote Access`.
14. A new window opens.
15. Right-click on the server and select `Configure and Enable Routing and Remote Access`.
16. Select `Network Adress Translation (NAT)`.
17. Select the `WAN` interface.
18. Finish the wizard.
19. Open the Start Menu and search for `Firewall`. Open `Windows Defender Firewall`.
20. Select `Allow an app or feature through the firewall`.
21. Make sure `Routing and Remote Access` is ticked.
22. This server is now mostly ready. To install DHCP, complete the `WIN-DC2` installation first.

DHCP

1. Make sure `WIN-DC2` is running and in the domain.
2. Open `WIN-DC1` in VirtualBox.
3. Use `VANLIEFFERINGE\Administrator` as username and `Admin2019` as password.

4. Open the file explorer.
5. Go to `C:\vagrant\provisioning\WIN-DC1\`.
6. Right-click on `6_dhcp.ps1`. Select `Run with Powershell`.
7. Wait for the script to install and configure DHCP.
8. This server is now ready.

Install WIN-DC2

1. Execute `vagrant up WIN-DC2` in your vagrant directory.
2. No additional steps are needed for this server. This server is now ready.

- If adding to domain fails, set the domain manually
- Reboot
- After reboot, promote server to DC in existing domain

3. Go back to the `WIN-DC1` installation guide to configure DHCP.

Install WIN-SQL-SCCM

1. Execute `vagrant up WIN-SQL-SCCM` in your vagrant directory.
2. While the scripts are running, open the VM in VirtualBox.
3. Use `VANLIEFFERINGE\Administrator` as username and `Admin2019` as password.
4. A popup will appear asking for MDT integration.
5. Follow the wizard.
6. After the wizard is done, the scripts will continue to run automatically.
7. When the scripts are done, open the VM in VirtualBox again.
8. Use `VANLIEFFERINGE\Administrator` as username and `Admin2019` as password.
9. Open the Start Menu and search for `System Center Configuration Manager`. Open the console.
10. Follow the Task sequence documentation to create a task sequence in order to deploy a client.
11. This server is now ready.

Install WIN-EXC-SHP

1. Execute `vagrant up WIN-EXC-SHP` in your vagrant directory.
2. When the scripts are done, open the VM in VirtualBox.
3. Use `VANLIEFFERINGE\Administrator` as username and `Admin2019` as password.
4. This server is now ready.

Deploy a Windows 10 client

1. Refer to the Task sequence to deploy a client.

Task sequence to deploy Windows 10 clients

General note

- If the `DeploymentShare` folder isn't available on the network, share it manually by browsing to `C:\`, right-clicking on the `DeploymentShare` folder, going into properties and sharing it with Administrators.

Creating a boot image

1. In the SCCM console, go to `Software Library > Overview > Operating images > Boot images`.
2. Open the context menu by pressing RMB on `boot images`.
3. Select `Create Boot image using MDT`.
4. Enter `\\WIN-SQL-SCCM\DeploymentShare\Boot` as path.
5. Click `Next`.
6. Enter `Windows10x64` as boot name.
7. Click `Next`. Select `X64` as platform. Leave all the other options on default.
8. Click on `Finish`.
9. Open the context menu on the newly created `Windows10x64` boot image by using RMB.
10. Select `Properties`.
11. Go to the `Data source` tab and check `Deploy this boot image from PXE enabled Distribution point`.
12. Apply and exit out of the properties menu.
13. Open the context menu on the newly created `Windows10x64` boot image by using RMB.
14. Select `Distribute content`. Enter `WIN-SQL-SCCM` as distribution point.
15. Click on `Finish`.
16. The boot image is now ready.

Creating a OS image

1. In the SCCM console, go to `Software Library > Overview > Operating Systems > Operating systems images`.
2. Open the context menu by pressing RMB on `Operating system images`.
3. Select `Add operating system image`.
4. Enter `\\WIN-SQL-SCCM\DeploymentShare\Operating Systems\Win10Consumers1809\sources\install.wim` as path.
5. Press `Next`.
6. Enter `Windows 10` as name and `1809` as version.
7. Press `Finish`.
8. Open the context menu on the newly created `Windows 10` image by using RMB.
9. Select `Distribute content`.
10. Press `Next` in the wizard.
11. Click `Add distribution point`.
12. Enter `WIN-SQL-SCCM.vanliefferinge.periode1` as distribution point.
13. Press `Next`.
14. Press `Finish`.
15. The OS image is now ready.

Creating a Task Sequence

1. In the SCCM console, go to `Software Library > Overview > Operating Systems > Task sequences`.
2. Open the context menu by pressing RMB on `Task sequences`.
3. Select `Create MDT task sequence`.
4. Leave the template on `Client task sequence`.
5. Enter `Windows 10` as task sequence name.
6. Enter the following on the `details` window:

```
Join a domain:
Domain: vanliefferinge.periode1
Account: VANLIEFFERINGE\\Administrator

Windows Settings:
User name: Admin
Organization name: VANLIEFFERINGE

Administrator Account:
Enable password: Admin2019
```

7. Click `Next`.
8. Leave `capture settings` on default and click `Next`.
9. On the `boot image` window, select the boot image you created earlier: `Windows10x64`.
10. Under `MDT package`, select `Create a new MDT package`.
11. Browse to `\\WIN-SQL-SCCM\DeploymentShare\Packages\MDT`.
12. Enter `MDT` as name.
13. Click `Next`.
14. On the `OS image` window, select `browse for existing..`.
15. Select the `Windows 10 1809 en-US` image.
16. Click `Next`.
17. Select `Windows 10`.
18. Leave deployment method on `No user interaction`.
19. Under `client package`, select `browse for existing...`.
20. Select `Microsoft Corporation Configuration Manager Client Package`.
21. Click `Next`.
22. Under `USMT package`, select `browse for existing...`.
23. Select `Microsoft Corporation User State Migration Tool for Windows 10`.
24. Click `Next`.
25. Under `Settings package`, select `Create a new Settings package`.
26. Browse to `\\WIN-SQ-SCCM\DeploymentShare\Settings`.
27. Enter `Windows 10 Settings` as name.
28. Click `Next`.
29. Under the `SysPrep` menu, leave everyting on default and press `Next`.
30. Confirm your settings.
31. The task sequence is now successfully created.

32. In the SCCM console, go to `Software Library > Overview > Application Management > Packages`.
33. Open the context menu on `MDT` using RMB.
34. Select `Distribute content`.
35. Press `Next` in the wizard.
36. Click `Add distribution point`.
37. Enter `WIN-SQL-SCCM.vanliefferinge.periode1` as distribution point.
38. Press `Next`.
39. Press `Finish`.
40. Repeat steps `33` to `40` for the following packages:
    - `User State Migration Tool (USMT)`
    - `Windows 10 Settings`
41. The task sequence is now complete.

Creating a Adobe Reader Application

1. In SCCM console, go to `Software Library > Overview > Application Management > Applications`. Select `Create application`.
2. Select `MSI` and browse to `C:\SetupMedia\AcroRdrDC1500720033_en_US`.
3. On the `General Information` page, enter the following line into the `Installation program` field:

```
msiexec /i "AcroRdrDC1500720033_en_US.msi" /q
```

4. Make sure `Install behavior` is set to `Install for user`.
5. Click `Next`, and `Next` again.
6. Close the wizard.
7. Open the Adobe Reader application properties by clicking RMB on to the newly created Adobe Reader application.
8. Check `Allow this application to be installed from the install application task sequence action without being deployed`.
9. Open the Adobe Reader context menu by clicking RMB on to the newly created Adobe Reader application.
10. Select `Distribute content`.
11. Adobe Reader is now ready for deployment.

Add applications into task sequence

1. In SCCM console, go to `Software Library > Overview > Operating Systems > Task sequences`.
2. Open the context menu by pressing RMB on `Windows 10`.
3. Select `Edit`.
4. Browse to the `Post install` section.
5. Press `Apply network settings`.
6. Enter the following in the `domain OU` section:
   `LDAP://CN=Computers,DC=vanliefferinge,DC=periode1`
7. Go to the `State restore` section.

8. Select `Install Application`.
9. Check `Install the following applications` and add Adobe Reader.
10. Add an extra step before the `Install software` step by pressing the `Add` button.
11. Select `general > Restart Computer` and press `Apply`.
12. Open Windows Explorer and browse to `C:\DeploymentShare\Settings`.
13. Open `CustomSettings.ini` using Notepad or a similar program.
14. Copy the following settings into the file:

```
[Settings]
Priority=Default
Properties=MyCustomProperty

[Default]
OSInstall=Y
OSDComputerName=Client01
SkipAppsOnUpgrade=YES
SkipComputerName=YES
SkipDomainMembership=YES
SkipUserData=YES
UserDataLocation=Auto
SkipLocaleSelection=YES
SkipTaskSequence=NO
MachineObjectOU=CN=Computers,DC=vanliefferinge,dc=periode1
DeploymentType=NEWCOMPUTER
SkipTimeZone=YES
SkipApplications=NO
SkipBitLocker=YES
SkipSummary=YES
SkipBDDWelcome=YES
SkipCapture=YES
DoCapture=NO
SkipFinalSummary=NO
TimeZone105
TimeZoneName=Romance Standard Time
JoinDomain=VANLIEFFERINGE
DomainAdmin=Administrator
DomainAdminDomain=VANLIEFFERINGE
DomainAdminPassword=Admin2019
SkipAdminPassword=YES
SkipProductKey=YES
```

15. Save and close the file.
16. In the SCCM console, go to `Software Library > Overview > Application Management > Packages`.
17. Open the context menu on the `MDT` package using RMB.
18. Check `Copy the content in this package to a package share on distribution points`.
19. Close the window with `Ok`.
20. Open the context menu on the `MDT` package using RMB.
21. Select `Update distribution points`.

22. Repeat steps `17` to `21` for the following packages:
    - `User State Migration Tool (USMT)`
    - `Windows 10 Settings`
23. In the SCCM console, go to `Software Library > Overview > Operating Systems > Task Sequences`.
24. Open the context menu on `Windows 10`.
25. Select `Deploy`.
26. In the `Collection` section, press `Browse`.
27. Select `All unknown computers`.
28. Press `Next`.
29. Change the `make available to the following` option to `Only media and PXE`.
30. Leave the rest of the wizard on default. Continue by pressing `Next` and `Finish`.
31. The task sequence is now complete.

Setting up a VirtualBox Client

1. In VirtualBox, create a new VM using `New`.
2. Enter `Client01` as name and `Windows 10 (64 bit)` as version.
3. Click `Next`.
4. Leave RAM settings on default.
5. Click `Next`.
6. Check `Create a virtual hard disk now`.
7. Click `Create`.
8. Check `VHD - Virtual Hard Drive`.
9. Click `Next`.
10. Check `Dynamically allocated`.
11. Click `Next`.
12. Leave size on 50GB.
13. Click `Create`.
14. Open the settings of the newly created VM.
15. Under `Network`, make sure only 1 internal adapter is enabled. Use Intel PRO/1000 T Server (82543GC) as the adapter type.
16. Select a LAN interface. To create new interfaces, refer to the VirtualBox documentation.
17. Under `System`, have the following boot order:
    - Hard Disk: checked
    - Network: checked
    - Optical: unchecked
    - Floppy: unchecked
18. Close the settings.
19. Launch the newly created VM using `Launch`.
20. Press `F12` when prompted.
21. Click `Next`.
22. Select `Windows 10`.
23. The installation will now continue to run automatically.
24. When prompted with a login screen, enter `VANLIEFFERINGE\Administrator` as username and `Admin2019` as password.
25. The client is now complete.

Provisioning Scripts

WIN-DC1

```
# Script 1 for WIN-DC1

Write-Host "Setting up adapters..."

# Rename adapters to WAN and LAN for clarity
$adaptercount = (Get-NetAdapter | Measure-Object).count
if ($adaptercount -eq 1) {
        (Get-NetAdapter -Name "Ethernet" 2> $null) | Rename-NetAdapter -
NewName "WAN"
    }

elseif ($adaptercount -eq 2) {
    (Get-NetAdapter -Name "Ethernet" 2> $null) | Rename-NetAdapter -
NewName "WAN"
    (Get-NetAdapter -Name "Ethernet 2" 2> $null) | Rename-NetAdapter -
NewName "LAN"
}

# Clear potential static settings and default gateway
Set-NetIPInterface -InterfaceAlias "LAN" -Dhcp Enabled 2> $null
Get-NetIPAddress -InterfaceAlias "LAN" | Remove-NetRoute -
Confirm:$false 2> $null

$dns1 = '192.168.100.10'
$dns2 = '192.168.100.20'

# Set the new IP settings + DNS

Write-Host "Setting up new IP settings..."

New-NetIPAddress -InterfaceAlias "LAN" -IPAddress $dns1 -PrefixLength 24 -
DefaultGateway $dns1 -AddressFamily IPv4 > $null

Set-DnsClientServerAddress -InterfaceAlias "LAN" -ServerAddresses($dns1,$dns2)
Set-DnsClientServerAddress -InterfaceAlias "WAN" -ResetServerAddresses

# Enable domain sharing services
Write-Host "Enabling specific sharing services..."

Set-Service -Name "FDResPub" -StartupType "Automatic"
Start-Service -DisplayName "Function Discovery Resource Publication"

Set-Service -Name "Dnscache" -StartupType "Automatic" 2> $null
Start-Service -DisplayName "DNS Client"
```

```
Set-Service -Name "SSDPSRV" -StartupType "Automatic"
Start-Service -DisplayName "SSDP Discovery"

Set-Service -Name "upnphost" -StartupType "Automatic"
Start-Service -DisplayName "UPnP Device Host"

# Allow services through firewall
# Set action to allow
Get-NetFirewallRule -DisplayGroup "Network Discovery" | Set-NetFirewallRule -
Action Allow
# Enabling the rule
Get-NetFirewallRule -DisplayGroup "Network Discovery" | Enable-NetFirewallRule

Get-NetFirewallRule -DisplayGroup "File and Printer Sharing" | Set-
NetFirewallRule -Action Allow
Get-NetFirewallRule -DisplayGroup "File and Printer Sharing" | Enable-
NetFirewallRule

# Turn on network discovery
netsh advfirewall firewall set rule group="Network Discovery" new enable=Yes

Write-Host "Complete..."
```

--

```
# Script 2 for WIN-DC1

Write-Host "Starting DC 1 configuration..."

# set timezone
Set-Culture -CultureInfo 'eng-BE'
Set-Timezone -Name "Romance Standard Time"

# add a local admin user
Write-Host "Adding local Admin..."
$password = "Admin2019" | ConvertTo-SecureString -AsPlainText -Force
Set-LocalUser -Name Administrator -AccountNeverExpires -Password $password -
PasswordNeverExpires:$true -UserMayChangePassword:$true

#install AD DS features
Write-Host "Installing AD features..."
Install-WindowsFeature AD-Domain-Services
Install-WindowsFeature RSAT-AD-AdminCenter
Install-WindowsFeature RSAT-ADDS-Tools

Write-Host "Adding a forest..."

# add a forest
```

```
Import-Module ADDSDeployment
Install-ADDSForest -InstallDns -CreateDnsDelegation:$False -ForestMode 7 -
DomainMode 7 -DomainName "vanliefferinge.periode1" -
SafeModeAdministratorPassword $password -NoRebootOnCompletion -Force

# try to find the domain every 10 seconds until it is installed
#Write-Host "Waiting for domain..."

#while ($true) {
#    try {
#        Get-ADDomain | Out-Null
#        break
#    } catch {
#        Write-Host "Still waiting..."
#        Start-Sleep -Seconds 10
#    }
#}

#Import-Module ActiveDirectory

#Write-Host "Disable unused accounts..."
#$enabledaccounts = @("vagrant","Administrator")
#Get-ADUser -Filter {Enabled -eq $true} | Where-Object {$enabledaccounts -
notcontains $_.Name} | Disable-ADAccount

#Set-ADAccountPassword -Identity "CN=Administrator,CN=users,Get-
ADDomain.DistinguishedName" -Reset -NewPassword $password
#Set-ADUser -Identity "CN=Administrator,CN=users,Get-
ADDomain.DistinguishedName" -PasswordNeverExpires $true

#Write-Host "Add vagrant account..."
#Add-ADGroupMember -Identity 'Domain Admins' -
Members "CN=vagrant,CN=users,Get-ADDomain.DistinguishedName"
#Add-ADGroupMember -Identity 'Enterprise Admins' -
Members "CN=vagrant,CN=users,Get-ADDomain.DistinguishedNameh"

#add a OU and account
#New-ADOrganizationalUnit -Name "IT" -Description "IT OU"
#New-ADGroup -Name "IT" -DisplayName "IT" -
Path "OU=IT,DC=vanliefferinge,DC=periode1" -GroupCategory Security -
GroupScope Global
#New-AdUser -Name "Jens" -Surname "Van Liefferinge" -SamAccountName "JensVL" -
Department "IT" -Description "JensVL Account" -DisplayName "JensVL" -
GivenName "Jens" -State "Brussels" -City "Brussels" -PostalCode "1000" -
EmailAddress "jensvl@vanliefferinge.periode1" -Office "D1" -EmployeeID 10 -
HomePhone "0499999999" -Initials "JVL" -
Path "OU=IT,DC=vanliefferinge,DC=periode1" -AccountPassword $password
```

```
#configure user and managers into group
#Add-ADGroupMember -Identity "CN=IT,OU=IT,DC=vanliefferinge,DC=periode1" -
Members "CN=Jens,OU=IT,DC=vanliefferinge,DC=periode1"
#Set-ADGroup -Identity "CN=IT,OU=IT,DC=vanliefferinge,DC=periode1" -
ManagedBy "CN=Jens,OU=IT,DC=vanliefferinge,DC=periode1"
#Set-ADOrganizationalUnit -Identity "OU=IT,DC=vanliefferinge,DC=periode1" -
ManagedBy "CN=Jens,OU=IT,DC=vanliefferinge,DC=periode1"

#enable the account
#Enable-ADAccount -Identity "CN=Jens,OU=IT,DC=vanliefferinge,DC=periode1"

Write-Host "Complete..."
```

```
# Extra script because server needs to reboot

Write-Host "Waiting for domain..."

while ($true) {
    try {
        Get-ADDomain | Out-Null
        break
    } catch {
        Write-Host "Still waiting..."
        Start-Sleep -Seconds 10
    }
}

Import-Module ActiveDirectory

Write-Host "Disabling unused accounts..."

$enabledaccounts = @("vagrant","Administrator")
$password = "Admin2019" | ConvertTo-SecureString -AsPlainText -Force
$domad = Get-ADDomain
$domname = $domad.DistinguishedName
$path = "CN=Users,$domname"

Get-ADUser -Filter {Enabled -eq $true} | Where-Object {$enabledaccounts -
notcontains $_.Name} | Disable-ADAccount

Set-ADAccountPassword -Identity "CN=Administrator,$path" -Reset -
NewPassword $password
Set-ADUser -Identity "CN=Administrator,$path" -PasswordNeverExpires $true

Write-Host "Adding vagrant account..."
Add-ADGroupMember -Identity 'Domain Admins' -Members "CN=vagrant,$path"
Add-ADGroupMember -Identity 'Enterprise Admins' -Members "CN=vagrant,$path"
```

```
#add a OU and account
Write-Host "Setting up Org Units..."

New-ADOrganizationalUnit -Name "IT" -Description "IT OU"
New-ADGroup -Name "IT" -DisplayName "IT" -
Path "OU=IT,DC=vanliefferinge,DC=periode1" -GroupCategory Security -
GroupScope Global
New-AdUser -Name "Jens" -Surname "Van Liefferinge" -SamAccountName "JensVL" -
Department "IT" -Description "JensVL Account" -DisplayName "JensVL" -
GivenName "Jens" -State "Brussels" -City "Brussels" -PostalCode "1000" -
EmailAddress "jensvl@vanliefferinge.periode1" -Office "D1" -EmployeeID 10 -
HomePhone "0499999999" -Initials "JVL" -
Path "OU=IT,DC=vanliefferinge,DC=periode1" -AccountPassword $password

#configure user and managers into group
Add-ADGroupMember -Identity "CN=IT,OU=IT,DC=vanliefferinge,DC=periode1" -
Members "CN=Jens,OU=IT,DC=vanliefferinge,DC=periode1"
Set-ADGroup -Identity "CN=IT,OU=IT,DC=vanliefferinge,DC=periode1" -
ManagedBy "CN=Jens,OU=IT,DC=vanliefferinge,DC=periode1"
Set-ADOrganizationalUnit -Identity "OU=IT,DC=vanliefferinge,DC=periode1" -
ManagedBy "CN=Jens,OU=IT,DC=vanliefferinge,DC=periode1"

#enable the account
Enable-ADAccount -Identity "CN=Jens,OU=IT,DC=vanliefferinge,DC=periode1"
```

```
# Script 4 for WIN-DC1

$dns1 = '192.168.100.10'
$dns2 = '192.168.100.20'

Write-Host "Fixing DNS settings after reboot..."

Set-DnsClientServerAddress -InterfaceAlias "LAN" -ServerAddresses($dns1,$dns2)
Set-DnsClientServerAddress -InterfaceAlias "WAN" -ResetServerAddresses

Write-Host "Complete..."
```

```
# Script 5 for WIN-DC1

$zone_exists=(Get-DnsServerZone -Name "vanliefferinge.periode1")
if (!$zone_exists) {
    Write-Host 'Adding primary DNS zone and enabling replication...'
    Add-DnsServerPrimaryZone -Name "vanliefferinge.periode1" -
ReplicationScope "Domain" -DynamicUpdate "Secure"
    Set-DnsServerPrimaryZone -Name "vanliefferinge.periode1" -
SecureSecondaries "TransferToZoneNameServer"
```

```
}

Write-Host 'Adding A records...'
Add-DnsServerResourceRecordA -Name "WINSQLSCCM" -
ZoneName "vanliefferinge.periode1" -IPv4Address "192.168.100.30"
Add-DnsServerResourceRecordA -Name "WINEXCSHP" -
ZoneName "vanliefferinge.periode1" -IPv4Address "192.168.100.40"

Write-Host 'Adding MX and CNAME records..'
Add-DnsServerResourceRecordMX -Name "WINEXCSHP" -
ZoneName "vanliefferinge.periode1" -
MailExchange "mail.vanliefferinge.periode1" -Preference 100
Add-DnsServerResourceRecordCName -Name "owa" -
ZoneName "vanliefferinge.periode1" -
HostNameAlias "mail.vanliefferinge.periode1"

Write-Host "Complete..."
```

```
# Script 6 for WIN-DC1

Write-Host "Starting DHCP configuration..."

Install-WindowsFeature DHCP -IncludeManagementTools

Add-DhcpServerInDC -DnsName "WIN-DC1.vanliefferinge.periode1" -
IpAddress 192.168.100.10

# add a scope and settings
Write-Host "Adding scope..."
Add-DhcpServerV4Scope -Name 'DHCP Scope' -StartRange 192.168.100.150 -
EndRange 192.168.100.200 -SubnetMask 255.255.255.0 -State Active

Write-Host "Configuring scope..."
Set-DhcpServerv4OptionValue -ScopeId 192.168.100.150 -OptionId 066 -
Value "WIN-SQL-SCCM.vanliefferinge.periode1"
Set-DhcpServerv4OptionValue -ScopeId 192.168.100.150 -OptionId 067 -
Value "\smsboot\x64\wdsnbp.com"

Set-DhcpServerv4Scope -ScopeId 192.168.100.150 -LeaseDuration 7.00:00:00
Set-DhcpServerV4OptionValue -ScopeId 192.168.100.150 -
DnsDomain "vanliefferinge.periode1" -DnsServer 192.168.100.10,192.168.100.20 -
Router 192.168.100.10

Restart-service -Name dhcpserver

# verify settings
Get-DhcpServerv4Scope
```

```
Get-DhcpServerInDC

Write-Host "Enabling specific sharing services..."

Set-Service -Name "FDResPub" -StartupType "Automatic"
Start-Service -DisplayName "Function Discovery Resource Publication"

Set-Service -Name "Dnscache" -StartupType "Automatic" 2> $null
Start-Service -DisplayName "DNS Client"

Set-Service -Name "SSDPSRV" -StartupType "Automatic"
Start-Service -DisplayName "SSDP Discovery"

Set-Service -Name "upnphost" -StartupType "Automatic"
Start-Service -DisplayName "UPnP Device Host"

# Allow services through firewall
# Set action to allow
Write-Host "Allow services trough firewall..."
Get-NetFirewallRule -DisplayGroup "Network Discovery" | Set-NetFirewallRule -
Action Allow
# Enabling the rule
Get-NetFirewallRule -DisplayGroup "Network Discovery" | Enable-NetFirewallRule

Get-NetFirewallRule -DisplayGroup "File and Printer Sharing" | Set-
NetFirewallRule -Action Allow
Get-NetFirewallRule -DisplayGroup "File and Printer Sharing" | Enable-
NetFirewallRule

# Turn on network discovery
netsh advfirewall firewall set rule group="Network Discovery"? new enable=Yes

Write-Host "Complete..."
```

WIN-DC2

```
# Script 1 for WIN-DC2

Write-Host "Setting up adapters..."

# Rename adapters to WAN and LAN for clarity
$adaptercount = (Get-NetAdapter | Measure-Object).count
if ($adaptercount -eq 1) {
        (Get-NetAdapter -Name "Ethernet" 2> $null) | Rename-NetAdapter -
NewName "WAN"
    }

elseif ($adaptercount -eq 2) {
    (Get-NetAdapter -Name "Ethernet" 2> $null) | Rename-NetAdapter -
NewName "WAN"
    (Get-NetAdapter -Name "Ethernet 2" 2> $null) | Rename-NetAdapter -
NewName "LAN"
}

# Clear potential static settings and default gateway
Set-NetIPInterface -InterfaceAlias "LAN" -Dhcp Enabled 2> $null
Get-NetIPAddress -InterfaceAlias "LAN" | Remove-NetRoute -
Confirm:$false 2> $null

$dns1 = '192.168.100.10'
$dns2 = '192.168.100.20'

# Set the new IP settings + DNS

Write-Host "Setting up new IP settings..."

New-NetIPAddress -InterfaceAlias "LAN" -IPAddress $dns2 -PrefixLength 24 -
DefaultGateway $dns1 -AddressFamily IPv4 > $null

Set-DnsClientServerAddress -InterfaceAlias "LAN" -ServerAddresses($dns1,$dns2)
Set-DnsClientServerAddress -InterfaceAlias "WAN" -ResetServerAddresses

# Enable domain sharing services
Write-Host "Enabling specific sharing services..."

Set-Service -Name "FDResPub" -StartupType "Automatic"
Start-Service -DisplayName "Function Discovery Resource Publication"

Set-Service -Name "Dnscache" -StartupType "Automatic" 2> $null
Start-Service -DisplayName "DNS Client"

Set-Service -Name "SSDPSRV" -StartupType "Automatic"
Start-Service -DisplayName "SSDP Discovery"
```

```
Set-Service -Name "upnphost" -StartupType "Automatic"
Start-Service -DisplayName "UPnP Device Host"

# Allow services through firewall
# Set action to allow
Get-NetFirewallRule -DisplayGroup "Network Discovery" | Set-NetFirewallRule -
Action Allow
# Enabling the rule
Get-NetFirewallRule -DisplayGroup "Network Discovery" | Enable-NetFirewallRule

Get-NetFirewallRule -DisplayGroup "File and Printer Sharing" | Set-
NetFirewallRule -Action Allow
Get-NetFirewallRule -DisplayGroup "File and Printer Sharing" | Enable-
NetFirewallRule

# Turn on network discovery
netsh advfirewall firewall set rule group="Network discovery" new enable=Yes

Write-Host "Complete..."
```

```
# Script 2 for WIN-DC2

Write-Host "Starting DC 2 configuration..."

. "c:\vagrant\provisioning\forest.ps1"


# set timezone
Set-Culture -CultureInfo 'eng-BE'
Set-Timezone -Name "Romance Standard Time"

# add a local admin user
Write-Host "Adding local Admin..."
$password = "Admin2019" | ConvertTo-SecureString -AsPlainText -Force
$localpw = "Admin1234" | ConvertTo-SecureString -AsPlainText -Force
Set-LocalUser -Name Administrator -AccountNeverExpires -Password $localpw -
PasswordNeverExpires:$true -UserMayChangePassword:$true

# add ADDS features
Write-Host "Installing AD features..."
Install-WindowsFeature AD-Domain-Services
Install-WindowsFeature RSAT-AD-AdminCenter
Install-WindowsFeature RSAT-ADDS-Tools

# Add to existing forest
Write-Host " Adding to forest..."
```

```
Import-Module ADDSDeployment
$credentials = New-
Object System.Management.Automation.PSCredential("VANLIEFFERINGE\Administrator
",$password)
install-ADDSDomainController -DomainName "vanliefferinge.periode1" -
ReplicationSourceDC "WIN-DC1.vanliefferinge.periode1" -
credential $credentials -InstallDns -createDNSDelegation:$false -
NoRebootOnCompletion -SafeModeAdministratorPassword $password -Force

# above gives error

# trying with function

$domain = 'vanliefferinge.periode1'
$dc1 = 'WIN-DC1'

#forest $domain $password $dc1

Write-Host " Complete..."
```

```
# Script 3 for WIN-DC2

$dns1 = '192.168.100.10'
$dns2 = '192.168.100.20'

Write-Host "Fixing DNS settings after reboot..."

Set-DnsClientServerAddress -InterfaceAlias "LAN" -ServerAddresses($dns1,$dns2)
Set-DnsClientServerAddress -InterfaceAlias "WAN" -ResetServerAddresses

Write-Host "Complete..."
```

```
function forest() {
    param(
        [string]$domain,
        [string]$password,
        [string]$dc1
    )

    $secpw = ConvertTo-SecureString $password -AsPlainText -Force
    Write-Host 'Installing into forest...'
    Import-Module ADDSDeployment
```

```
    $credentials = New-
Object System.Management.Automation.PSCredential("VANLIEFFERINGE\Administrator
",$secpw)
    Install-ADDSDomainController -DomainName $domain -
ReplicationSourceDC "$dc1.$domain" -credential $credentials -InstallDns -
createDNSDelegation:$false -NoRebootOnCompletion -
SafeModeAdministratorPassword $secpw -Force
}
```

WIN-SQL-SCCM

```
# Script 1 for WIN-SQL-SCCM

Write-Host "Setting up adapters..."

# Rename adapters to WAN and LAN for clarity
$adaptercount = (Get-NetAdapter | Measure-Object).count
if ($adaptercount -eq 1) {
        (Get-NetAdapter -Name "Ethernet" 2> $null) | Rename-NetAdapter -
NewName "WAN"
    }

elseif ($adaptercount -eq 2) {
    (Get-NetAdapter -Name "Ethernet" 2> $null) | Rename-NetAdapter -
NewName "WAN"
    (Get-NetAdapter -Name "Ethernet 2" 2> $null) | Rename-NetAdapter -
NewName "LAN"
}

# Clear potential static settings and default gateway
Set-NetIPInterface -InterfaceAlias "LAN" -Dhcp Enabled 2> $null
Get-NetIPAddress -InterfaceAlias "LAN" | Remove-NetRoute -
Confirm:$false 2> $null

$dns1 = '192.168.100.10'
$dns2 = '192.168.100.20'
$sqlip = '192.168.100.30'

# Set the new IP settings + DNS

Write-Host "Setting up new IP settings..."

New-NetIPAddress -InterfaceAlias "LAN" -IPAddress $sqlip -PrefixLength 24 -
DefaultGateway $dns1 -AddressFamily IPv4 > $null

Set-DnsClientServerAddress -InterfaceAlias "LAN" -ServerAddresses($dns1,$dns2)
Set-DnsClientServerAddress -InterfaceAlias "WAN" -ResetServerAddresses

# Enable domain sharing services
Write-Host "Enabling specific sharing services..."

Set-Service -Name "FDResPub" -StartupType "Automatic"
Start-Service -DisplayName "Function Discovery Resource Publication"

Set-Service -Name "Dnscache" -StartupType "Automatic" 2> $null
Start-Service -DisplayName "DNS Client"

Set-Service -Name "SSDPSRV" -StartupType "Automatic"
```

```
Start-Service -DisplayName "SSDP Discovery"

Set-Service -Name "upnphost" -StartupType "Automatic"
Start-Service -DisplayName "UPnP Device Host"

# Allow services through firewall
# Set action to allow
Get-NetFirewallRule -DisplayGroup "Network Discovery" | Set-NetFirewallRule -
Action Allow
# Enabling the rule
Get-NetFirewallRule -DisplayGroup "Network Discovery" | Enable-NetFirewallRule

Get-NetFirewallRule -DisplayGroup "File and Printer Sharing" | Set-
NetFirewallRule -Action Allow
Get-NetFirewallRule -DisplayGroup "File and Printer Sharing" | Enable-
NetFirewallRule

# Turn on network discovery
netsh advfirewall firewall set rule group="Network Discovery" new enable=Yes

Write-Host "Complete..."
```

```
# Script 2 for WIN-SQL-SCCM

Write-Host "Joining domain..."

$password = "Admin2019" | ConvertTo-SecureString -AsPlainText -Force

$credentials = New-
Object System.Management.Automation.PsCredential("VANLIEFFERINGE\Administrator
",$password)
Add-computer -DomainName "vanliefferinge.periode1" -
DomainCredential $credentials -Verbose

Write-Host "Complete..."
```

```
# Script 1 for WIN-SQL-SCCM

Write-Host "Installing SQL Server..."

# Download SQL Server
$downloadpath = 'C:\SetupMedia'

Write-Host "Verifying downloadpath..."
if($downloadpath.EndsWith("\")){
    $computerName.Remove($computerName.LastIndexOf("\"))
```

```
}

if(!(Test-Path $downloadpath)){
    mkdir $downloadpath
}

Write-Host 'Downloading SQL installer...'
(New-
Object System.Net.WebClient).DownloadFile("https://go.microsoft.com/fwlink/?li
nkid=853016", "$downloadpath\sqlinstaller.exe")

Write-Host 'Starting SQL installer...'
Start-Process -FilePath "$downloadpath\sqlinstaller.exe" -
ArgumentList "/action=download /quiet /enu /MediaPath=$downloadpath" -Wait -
WindowStyle hidden

Write-Host 'Extracting SQL Server files...'
Start-Process -FilePath $downloadpath\SQLServer2017-DEV-x64-ENU.exe -
WorkingDirectory $downloadpath /q -wait

# Install SQL Server
$sqlusername = 'VANLIEFFERINGE\Administrator'
$sqlpassword = 'Admin2019'

Write-Host "Installing SQL Server..."
$secure_pw = ConvertTo-SecureString $sqlpassword -AsPlainText -Force
$sccm_pw = [System.Runtime.Interopservices.Marshal]::PtrToStringAuto([System.R
untime.InteropServices.Marshal]::SecureStringToBSTR($secure_pw))
set-location "$downloadpath\SQLServer2017-DEV-x64-ENU"
.\SETUP.exe  `
/Q  `
/ACTION=Install `
/IACCEPTSQLSERVERLICENSETERMS `
/FEATURES="SQLENGINE,FULLTEXT" `
/INSTANCENAME="MSSQLSERVER" `
/INSTANCEID="MSSQLSERVER" `
/INSTANCEDIR="C:\Program Files\Microsoft SQL Server" `
/SQLCOLLATION=SQL_Latin1_General_CP1_CI_AS `
/SQLSVCACCOUNT="$sqlusername" `
/SQLSVCPASSWORD="$sccm_pw" `
/SQLTELSVCACCT="NT Service\SQLTELEMETRY" `
/SQLTELSVCSTARTUPTYPE="Automatic" `
/AGTSVCACCOUNT="$sqlusername" `
/AGTSVCPASSWORD="$sccm_pw" `
/AGTSVCSTARTUPTYPE="Automatic" `
/FTSVCACCOUNT="$sqlusername" `
/FTSVCPASSWORD="$sccm_pw" `
/SQLSYSADMINACCOUNTS="$sqlusername" `
```

```
/SAPWD="$sccm_pw" `
/SECURITYMODE="SQL" `
/INSTALLSHAREDDIR="C:\Program Files\Microsoft SQL Server" `
/INSTALLSHAREDWOWDIR="C:\Program Files (x86)\Microsoft SQL Server" `
/TCPENABLED="1" `
/NPENABLED="1"

# Add local users to admin
Write-Host 'Adding local users to administrator group...'
Add-LocalGroupMember -Group "Administrators" -
Member "VANLIEFFERINGE\Administrator"
Add-LocalGroupMember -Group "Administrators" -Member "VANLIEFFERINGE\WIN-SQL-
SCCM$"

# Misc sql settings like firewall etc
Write-Host "Importing SQL module..."
Import-Module -
name 'C:\Program Files (x86)\Microsoft SQL Server\140\Tools\PowerShell\Modules
\SQLPS'

Write-Host "Restarting SQL Instance..."
Restart-Service -Force "MSSQLSERVER" > $null

Write-Host "Setting firewall rule..."
New-NetFirewallRule -DisplayName "Allow inbound sqlserver" -
Direction Inbound -LocalPort 1443 -Protocol TCP -Action Allow > $null

# Download SSMS
Write-Host 'Downloading SSMS installer...'
(New-
Object System.Net.WebClient).DownloadFile("https://aka.ms/ssmsfullsetup", "$do
wnloadpath\SSMS-Setup-ENU.exe")

Write-Host "Installing SSMS..."
Start-Process -Filepath "$downloadpath\SSMS-Setup-ENU.exe" -ArgumentList -
silent -Wait

# Enable TCP/IP
$wmi = New-
Object ('Microsoft.SqlServer.Management.Smo.Wmi.ManagedComputer').

$Tcp = $wmi.GetSmoObject("ManagedComputer[@Name='WIN-SQL-
SCCM']/ ServerInstance[@Name='MSSQLSERVER']/ServerProtocol[@Name='Tcp']")
$Tcp.IsEnabled = $true
$Tcp.Alter()

#Set up scripts to run on next reboot
Write-Host 'Copying scripts...'
```

```
$scripts = "C:\scripts"

# Check paths again
if(!(Test-Path $scripts)){
    mkdir $scripts
}

Copy-Item -r "C:\vagrant\provisioning\WIN-SQL-SCCM\" "$scripts" -Force

# Set up auto logon
Write-Host 'Configuring auto logon as domain admin...'
$secure = ConvertTo-SecureString $sqlpassword -AsPlainText -Force
$credentials = New-
Object System.Management.Automation.PSCredential ($sqlusername, $secure)

$pw = $credentials.GetNetworkCredential().Password

Set-ItemProperty -
Path "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" -
Name DefaultUserName -Value $sqlusername
Set-ItemProperty -
Path "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" -
Name DefaultPassword -Value $pw
Set-ItemProperty -
Path "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" -
Name AutoAdminLogon -Value 1
Set-ItemProperty -
Path "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" -
Name ForceAutoLogon -Value 1

# load in next script
Set-ItemProperty -
Path 'HKLM:\Software\Microsoft\Windows\CurrentVersion\RunOnce' -
Name ResumeScript -
Value "C:\Windows\system32\WindowsPowerShell\v1.0\Powershell.exe -
executionpolicy bypass -file $scripts\WIN-SQL-SCCM\4_admin_perms.ps1"
```

```
# Script 4 for WIN-SQL-SCCM

. "c:\vagrant\provisioning\sql.ps1"

Write-Host "Setting up admin permissions..."

[System.Reflection.Assembly]::LoadWithPartialName("Microsoft.SqlServer.Smo") |
 Out-Null
[System.Reflection.Assembly]::LoadWithPartialName("Microsoft.SqlServer.SqlEnum
") | Out-Null
```

```
[System.Reflection.Assembly]::LoadWithPartialName("Microsoft.SqlServer.Connect
ionInfo") | Out-Null

$connection = New-
Object Microsoft.SqlServer.Management.Common.ServerConnection
$server = New-Object Microsoft.SqlServer.Management.Smo.Server $connection

# Allow domain admin to create db

$sqlusername = 'VANLIEFFERINGE\Administrator'

set_sql_perm $server $sqlusername "securityadmin"
set_sql_perm $server $sqlusername "sysadmin"
set_sql_perm $server $sqlusername "dbcreator"

# Set min and max server memory
set_sql_mem $server (get_sql_max_mem)

# load in next script
Set-ItemProperty -
Path 'HKLM:\Software\Microsoft\Windows\CurrentVersion\RunOnce' -
Name ResumeScript -
Value "C:\Windows\system32\WindowsPowerShell\v1.0\Powershell.exe -
executionpolicy bypass -file C:\scripts\WIN-SQL-SCCM\5_sccm_install.ps1"

Restart-Computer
```

```
# Script 5 for WIN-SQL-SCCM

Write-Host "Starting SCCM installation..."

$downloadpath = 'C:\SetupMedia'

# Verify downloadpath
Write-Host 'Verifying downloadpath...'
if($downloadpath.EndsWith("\")){
    $computerName.Remove($computerName.LastIndexOf("\"))
}
if(!(Test-Path $downloadpath)){
    mkdir $downloadpath
}

# Check for elevated shell
Write-Host "Checking for elevation..."
if (-
NOT ([Security.Principal.WindowsPrincipal] [Security.Principal.WindowsIdentity
```

```
]::GetCurrent()).IsInRole([Security.Principal.WindowsBuiltInRole] "Administrat
or")) {
    Write-
Host "You need to run this script from an elevated PowerShell prompt..."
    Break
}

# Extend AD schema using separate script
Write-Host 'Loading script...'
Invoke-Command -FilePath "C:\scripts\WIN-SQL-SCCM\prep_ad_for_sccm.ps1" -
ComputerName "WIN-DC1"

Write-Host 'Extending AD schema on WIN-DC1...'
Copy-Item "C:\scripts\WIN-SQL-SCCM\ExtendADschema" -Destination "C:\" -Recurse
Start-Process "C:\ExtendADschema\extadsch.exe" -wait

# Downloading prereqs
Write-Host 'Downloading ADK installer...'
(New-
Object System.Net.WebClient).DownloadFile("https://go.microsoft.com/fwlink/?li
nkid=2086042", "$downloadpath\adksetup.exe")

Write-Host 'Downloading WinPE addon installer...'
(New-
Object System.Net.WebClient).DownloadFile("https://go.microsoft.com/fwlink/?li
nkid=2087112", "$downloadpath\adkwinpesetup.exe")

Write-Host 'Downloading MDT installer...'
(New-
Object System.Net.WebClient).DownloadFile("https://download.microsoft.com/down
load/3/3/9/339BE62D-B4B8-4956-B58D-
73C4685FC492/MicrosoftDeploymentToolkit_x64.msi", "$downloadpath\MicrosoftDepl
oymentToolkit_X64.msi")

Write-Host 'Downloading Adobe Reader installer...'
(New-
Object System.Net.WebClient).DownloadFile("ftp://ftp.adobe.com/pub/adobe/reade
r/win/AcrobatDC/1500720033/AcroRdrDC1500720033_en_US.msi", "$downloadpath\Acro
RdrDC1500720033_en_US.msi")

if (!(Test-Path -path "C:\scripts\WIN-SQL-SCCM\win10\Windows10_1809.iso")) {
  Write-Host "Downloading win10 iso..."
  (New-
Object System.Net.WebClient).DownloadFile("https://archive.org/download/Win10C
onsumerEditionsv1809x86x64/en_windows_10_consumer_editions_version_1809_update
d_dec_2018_x64_dvd_d7d23ac9.iso", "C:\scripts\WIN-SQL-
SCCM\win10\Windows10_1809.iso")
} else {
```

```
  Write-Host "win10 iso present..."
}


# Install prereqs

# Validate installers are present
if (!(Test-Path -path "$downloadpath\adksetup.exe")) {
    Write-Host 'ADK installer missing...'
    Break
}
if (!(Test-Path -path "$downloadpath\adkwinpesetup.exe")) {
    Write-Host 'WinPE installer missing...'
    Break
}
if (!(Test-Path -path "$downloadpath\MicrosoftDeploymentToolkit_X64.msi")) {
    Write-Host 'MDT installer missing...'
    Break
}

# Install ADK
Write-Host "Installing ADK..."
Start-Process -FilePath "$downloadpath\adksetup.exe" -
ArgumentList "/Features OptionId.DeploymentTools OptionId.ImagingAndConfigurat
ionDesigner OptionId.ICDConfigurationDesigner OptionId.UserStateMigrationTool
/norestart /quiet /ceip off" -NoNewWindow -Wait

if ($?) {
  Write-Host "Installation ADK completed..."
} else {
  Write-Host "Installation ADK failed..."
}

# Install WinPE
Write-Host "Installing WinPE..."
Start-Process -FilePath "$downloadpath\adkwinpesetup.exe" -
ArgumentList "/Features OptionId.WindowsPreinstallationEnvironment /norestart
/quiet /ceip off" -NoNewWindow -Wait

if ($?) {
  Write-Host "Installation WinPE completed..."
} else {
  Write-Host "Installation WinPE failed..."
}

# Install MDT
Write-Host "Installing MDT..."
Start-Process "$downloadpath\MicrosoftDeploymentToolkit_X64.msi" /quiet -wait
```

```
if ($?) {
  Write-Host "Installation Windows MDT completed..."
} else {
  Write-Host "Installation Windows MDT failed..."
}

# Install WDS
Write-Host 'Installing WDS...'
Import-Module ServerManager
Install-WindowsFeature -Name WDS -IncludeManagementTools

# Install IIS, BITS and RDS extras
Write-Host "Installing IIS, BITS and RDC..."
Install-WindowsFeature Web-Static-Content,Web-Default-Doc,Web-Dir-
Browsing,Web-Http-Errors,Web-Http-Redirect,Web-Net-Ext,Web-ISAPI-Ext,Web-Http-
Logging,Web-Log-Libraries,Web-Request-Monitor,Web-Http-Tracing,Web-Windows-
Auth,Web-Filtering,Web-Stat-Compression,Web-Mgmt-Tools,Web-Mgmt-Compat,Web-
Metabase,Web-WMI,BITS,RDC

# Install and configure WSUS for SQL
Write-Host "Installing WSUS..."
Install-WindowsFeature -Name UpdateServices-DB, UpdateServices-Services -
IncludeManagementTools
New-Item -Path "C:\" -ItemType Directory -Name "WSUS"

# Link WSUS to SQL
Write-Host "Configuring WSUS for SQL Server"
Set-Location -Path "C:\Program Files\Update Services\Tools"
.\wsusutil.exe postinstall SQL_INSTANCE_NAME="WIN-SQL-
SCCM" CONTENT_DIR=C:\WSUS

# Configure firewall to allow traffic
Enable-NetFirewallRule -
DisplayGroup "Windows Management Instrumentation (WMI)" -confirm:$false
Enable-NetFirewallRule -DisplayName "File and Printer Sharing (NB-Name-In)" -
confirm:$false
Enable-NetFirewallRule -DisplayName "File and Printer Sharing (NB-Session-
In)" -confirm:$false
Enable-NetFirewallRule -DisplayName "File and Printer Sharing (SMB-In)" -
confirm:$false

netsh advfirewall firewall add rule name="SCCM Management Point" dir=in action
=allow profile=domain localport="7080,7443,10123" protocol=TCP

New-NetFirewallRule -Group SCCM -DisplayName "SCCM - File Share - TCP - 445" -
Direction Inbound -Protocol TCP -LocalPort 445 -Action Allow -
Profile Domain | Out-Null
```

```
New-NetFirewallRule -Group SCCM -DisplayName "SCCM - File Share - UDP - 137-
138" -Direction Inbound -Protocol UDP -LocalPort "137-138" -Action Allow -
Profile Domain | Out-Null
New-NetFirewallRule -Group SCCM -DisplayName "SCCM - RPC - TCP - 135" -
Direction Inbound -Protocol TCP -LocalPort 135 -Action Allow -
Profile Domain | Out-Null
New-NetFirewallRule -Group SCCM -DisplayName "SCCM - NetBIOS - TCP - 139" -
Direction Inbound -Protocol TCP -LocalPort 139 -Action Allow -
Profile Domain | Out-Null
New-NetFirewallRule -Group SCCM -
DisplayName "SCCM - Dynamic Ports - TCP - 49154-49157" -Direction Inbound -
Protocol TCP -LocalPort "49154-49157" -Action Allow -Profile Domain | Out-Null
New-NetFirewallRule -Group SCCM -DisplayName "SCCM - UDP - 5355" -
Direction Inbound -Protocol UDP -LocalPort "5355" -Action Allow -
Profile Domain | Out-Null

Get-Service RemoteRegistry | Set-Service -StartupType Automatic -
PassThru | Start-Service

# Install SCCM

# Download installer
Write-Host 'Downloading SCCM installer...'
(New-
Object System.Net.WebClient).DownloadFile("http://download.microsoft.com/downl
oad/1/B/C/1BCADBD7-47F6-40BB-8B1F-
0B2D9B51B289/SC_Configmgr_SCEP_1902.exe", "$downloadpath\SC_Configmgr_SCEP_190
2.exe")

Write-Host 'Extracting SCCM files...'
Move-
Item "$downloadpath\SC_Configmgr_SCEP_1902.exe" "$downloadpath\SC_Configmgr_SC
EP_1902.zip"
Expand-Archive -Path "$downloadpath\SC_Configmgr_SCEP_1902.zip" -
DestinationPath "$downloadpath\SC_Configmgr_SCEP_1902"

Write-Host 'Downloading prereqs...'
Start-
Process "$downloadpath\SC_Configmgr_SCEP_1902\SMSSETUP\BIN\x64\setupdl.exe" -
ArgumentList "$downloadpath\prereqs" -wait

Write-Host 'Running prereq check...'
Start-
Process "$downloadpath\SC_Configmgr_SCEP_1902\SMSSETUP\BIN\x64\prereqchk.exe"
-ArgumentList "/NOUI /PRI /SQL WIN-SQL-SCCM.vanliefferinge.periode1 /SDK WIN-
SQL-SCCM.vanliefferinge.periode1 /MP WIN-SQL-
SCCM.vanliefferinge.periode1 /DP WIN-SQL-SCCM.vanliefferinge.periode1" -wait
```

```
Write-Host 'Installing SCCM...'
Start-
Process "$downloadpath\SC_Configmgr_SCEP_1902\SMSSETUP\BIN\x64\setup.exe" -
ArgumentList "/script C:\scripts\WIN-SQL-SCCM\sccm_install_config.ini" -wait

# Integrate MDT with SCCM
Write-Host "Integrating MDT with SCCM..."

$MDT = "C:\Program Files\Microsoft Deployment Toolkit"
$SCCM = "C:\Program Files (x86)\Microsoft Configuration Manager\AdminConsole"
$MOF = "$SCCM\Bin\Microsoft.BDD.CM12Actions.mof"

Copy-
Item "$MDT\Bin\Microsoft.BDD.CM12Actions.dll" "$SCCM\Bin\Microsoft.BDD.CM12Act
ions.dll"
Copy-
Item "$MDT\Bin\Microsoft.BDD.Workbench.dll" "$SCCM\Bin\Microsoft.BDD.Workbench
.dll"
Copy-
Item "$MDT\Bin\Microsoft.BDD.ConfigManager.dll" "$SCCM\Bin\Microsoft.BDD.Confi
gManager.dll"
Copy-
Item "$MDT\Bin\Microsoft.BDD.CM12Wizards.dll" "$SCCM\Bin\Microsoft.BDD.CM12Wiz
ards.dll"
Copy-
Item "$MDT\Bin\Microsoft.BDD.PSSnapIn.dll" "$SCCM\Bin\Microsoft.BDD.PSSnapIn.d
ll"
Copy-Item "$MDT\Bin\Microsoft.BDD.Core.dll" "$SCCM\Bin\Microsoft.BDD.Core.dll"
Copy-Item "$MDT\SCCM\Microsoft.BDD.CM12Actions.mof" $MOF
Copy-Item "$MDT\Templates\CM12Extensions\*" "$SCCM\XmlStorage\Extensions\" -
Force -Recurse
(Get-
Content $MOF).Replace('%SMSSERVER%', "Van Liefferinge Site").Replace('%SMSSITE
CODE%', "JVL") | Set-Content $MOF
Get-Content $MOF
& "C:\Windows\System32\wbem\mofcomp.exe" "$SCCM\Bin\Microsoft.BDD.CM12Actions.
mof"

# Configure SCCM

# import SCCM commands
Set-Location -Path "$SCCM\bin"
Import-Module .\ConfigurationManager.psd1
New-PSDrive -Name "JVL" -PsProvider "AdminUI.PS.Provider\CMSite" -Root "WIN-
SQL-SCCM.vanliefferinge.periode1" -Description "Site drive for JVL"

Start-Sleep -s 30
Set-Location -Path JVL:
```

```
# Create boundary group
Write-Host "Creating boundaries and boundary groups..."
New-CMBoundary -Type ADSite -DisplayName "Active Directory Site" -
Value "Default-First-Site-Name"
New-CMBoundaryGroup -Name "ADsite"
Set-CMBoundaryGroup -Name "ADsite" -AddSiteSystemServerName "WIN-SQL-
SCCM.vanliefferinge.periode1" -DefaultSiteCode "JVL"
Add-CMBoundaryToGroup -BoundaryGroupName "ADSite" -
BoundaryName "Active Directory Site"
Write-Host "boundaries creation complete..."

### Configure client settings and network access account
Set-CMClientSettingComputerAgent -BrandingTitle "JVL" -DefaultSetting

Write-Host "Configuring Network Access account..."
$password = ConvertTo-SecureString 'Admin2019' -AsPlainText -Force
New-CMAccount -UserName "VANLIEFFERINGE\Administrator" -Password $password -
SiteCode "JVL"
Set-CMSoftwareDistributionComponent -SiteCode "JVL" -
AddNetworkAccessAccountName "VANLIEFFERINGE\Administrator"

# Turn on discovery method to scan for new devices etc
Write-Host 'Turning on discovery methods...'
Set-CMDiscoveryMethod -ActiveDirectoryForestDiscovery -SiteCode "JVL" -
Enabled $true
Set-CMDiscoveryMethod -NetworkDiscovery -SiteCode "JVL" -Enabled $true -
NetworkDiscoveryType ToplogyAndClient
Set-CMDiscoveryMethod -ActiveDirectorySystemDiscovery -SiteCode "JVL" -
Enabled $true -ActiveDirectoryContainer "LDAP://DC=vanliefferinge,DC=periode1"
Set-CMDiscoveryMethod -ActiveDirectoryUserDiscovery -SiteCode "JVL" -
Enabled $true -ActiveDirectoryContainer "LDAP://DC=vanliefferinge,DC=periode1"

$scope = New-CMADGroupDiscoveryScope -
LDAPlocation "LDAP://DC=vanliefferinge,DC=periode1" -Name "ADdiscoveryScope" -
RecursiveSearch $true
Set-CMDiscoveryMethod -ActiveDirectoryGroupDiscovery -SiteCode "JVL" -
Enabled $true -AddGroupDiscoveryScope $scope

# Configure PXE boot
Write-Host "Configuring PXE boot..."
Set-CMDistributionPoint -SiteSystemServerName "WIN-SQL-
SCCM.vanliefferinge.periode1" -enablePXE $true -AllowPxeResponse $true -
EnableUnknownComputerSupport $true -RespondToAllNetwork
Write-Host "PXE boot configured..."


# Create win10 ref image
Write-Host "Copying win10 iso..."
```

```
Copy-Item "C:\scripts\WIN-SQL-SCCM\win10" -Destination "C:\" -Recurse -Verbose

Mount-DiskImage -ImagePath "C:\scripts\WIN-SQL-SCCM\win10\Windows10_1809.iso"
Write-Host "Copy and mount of iso complete..."
Write-Host "Creating deployment share..."

# Import MDT commamds
Import-
Module "C:\Program Files\Microsoft Deployment Toolkit\Bin\MicrosoftDeploymentT
oolkit.psd1"
New-Item -type "Directory" -Path "C:\DeploymentShare"

# Create share
([wmiclass]"win32_share").Create("C:\DeploymentShare","DeploymentShare",0)
New-PSDrive -Name "DS001" -
PSProvider "MicrosoftDeploymentToolkit\MDTProvider" -
Root "C:\DeploymentShare" -Description "Deployment Share for Windows 10" -
Verbose

# 10.3) Import win10 img into share
Import-MDTOperatingSystem -SourcePath "D:\" -Path "DS001:\Operating Systems" -
DestinationFolder "Win10Consumers1809" -verbose

# add software update role
Set-Location JVL:
Write-Host "Adding software update point..."
Add-CMSoftwareUpdatePoint -SiteCode "JVL" -SiteSystemServerName "WIN-SQL-
SCCM.vanliefferinge.periode1" -ClientConnectionType "Intranet"

# link wsus and sccm
Set-CMSoftwareUpdatePointComponent -
SynchronizeAction "SynchronizeFromMicrosoftUpdate" -
ReportingEvent "DoNotCreateWsusReportingEvents" -
RemoveUpdateClassification "Service Packs","Upgrades","Update Rollups","Tools"
,"Driver sets","Applications","Drivers","Feature Packs","Definition Updates","
Updates" -AddUpdateClassification "Updates", "Driver sets" -SiteCode "JVL" -
verbose

# only set english updates
Write-Host "Enabling ENG for updates..."

$WSUSserver = Get-WSUSserver
$WSUSconfig = $WSUSserver.GetConfiguration()
$WSUSconfig.AllUpdateLanguagesEnabled = $false
$WSUSconfig.SetEnabledUpdateLanguages("en")
$WSUSconfig.Save()

#Sync software updates with update servers
```

```powershell
$beforeSync = Get-Date

Restart-Service -Name sms_executive
Sync-CMSoftwareUpdate -FullSync $true

Start-Sleep -Seconds 300
Write-Host "Starting sync of updates..."
$syncStatus = Get-CMSoftwareUpdateSyncStatus
Sync-CMSoftwareUpdate -FullSync $true

$maxSeconds = (60*70)
$endWait = $beforeSync.AddSeconds($maxSeconds)

Write-Host "Waiting on sync..."
while ($now -lt $endWait) {
    $now = Get-Date
    $syncStatus = Get-CMSoftwareUpdateSyncStatus
    if ($null -eq $syncStatus.LastSuccessfulSyncTime -
or $syncStatus.LastSuccessfulSyncTime -lt $beforeSync) {
        continue
    } else {
        Write-Host "Sync complete..."
        break
    }
}

# add win10
Set-CMSoftwareUpdatePointComponent -SiteCode "JVL" -
AddProduct "Windows 10, version 1809 and later, Upgrade & Servicing Drivers"
```

```powershell
# Extra script for WIN-SQL-SCCM

Write-Host "Preparing AD Schema..."

# Add server to AD
Write-Host "Adding WIN-SQL-SCCM to AD..."
New-ADComputer -Name "WIN-SQL-SCCM"

Write-Host 'Connecting to ADSIedit...'
$ADSIconnection = [ADSI]"LDAP://localhost:389/cn=System,dc=vanliefferinge,dc=p
eriode1"

Write-Host 'Creating container...'
$SysManContainer = $ADSIconnection.Create("container", "cn=System Management")
$SysManContainer.SetInfo()

# Set perms
```

```
Write-Host 'Setting permissions for container...'
$SystemManagementCN = [ADSI]("LDAP://localhost:389/cn=System Management,cn=Sys
tem,dc=vanliefferinge,dc=periode1")
$SCCMserver = get-adcomputer "WIN-SQL-SCCM"
$SID = [System.Security.Principal.SecurityIdentifier] $SCCMserver.SID
$ServerIdentity = [System.Security.Principal.IdentityReference] $SID

$perms = [System.DirectoryServices.ActiveDirectoryRights] "GenericAll"
$allow = [System.Security.AccessControl.AccessControlType] "Allow"
$inheritall = [System.DirectoryServices.ActiveDirectorySecurityInheritance] "A
ll"

# Apply perms
$permrule = New-Object System.DirectoryServices.ActiveDirectoryAccessRule `
$ServerIdentity,$perms,$allow,$inheritall

$SystemManagementCN.psbase.ObjectSecurity.AddAccessRule($permrule)
$SystemManagementCN.psbase.commitchanges()

Write-Host "AD prep complete..."
```

```
# useful functions

# set database perms for a user - add to role
function set_sql_perm() {
    param(
        [Microsoft.SQLServer.Management.Smo.Server]$server,
        [string]$sqlusernameFc,
        [string]$role
    )
    $login = New-
Object Microsoft.SqlServer.Management.Smo.Login $server, "$sqlusernameFc"
    $login.LoginType = [Microsoft.SqlServer.Management.Smo.LoginType]::Windows
User
    $login.AddToRole($role)
}

# set as much memory as possible to allow sccm to install faster

# helper function to set_sql_mem
function get_sql_max_mem() {
    $memtotal = 8192
    $min_os_mem = 2048
    if ($memtotal -le $min_os_mem) {
        Return $null
    }
    if ($memtotal -ge 8192) {
```

```
        $sql_mem = $memtotal - 2048
    } else {
        $sql_mem = $memtotal * 0.8
    }
    return [int]$sql_mem
}

# actual set function
function set_sql_mem() {
    param(
        [Microsoft.SQLServer.Management.Smo.Server]$serverFc,
        [int]$maxmem = $null,
        [int]$minmem = 0
    )
    if ($minmem -eq 0) {
        $minmem = $maxmem
    }
    if ($serverFc.status) {
        $serverFc.Configuration.MaxServerMemory.ConfigValue = $maxmem
        $serverFc.Configuration.MinServerMemory.ConfigValue = $minmem
        $serverFc.Configuration.Alter()
    }
}
```

WIN-EXC-SHP

```
# Script 1 for WIN-EXC-SHP

Write-Host "Setting up adapters..."

# Rename adapters to WAN and LAN for clarity
$adaptercount = (Get-NetAdapter | Measure-Object).count
if ($adaptercount -eq 1) {
        (Get-NetAdapter -Name "Ethernet" 2> $null) | Rename-NetAdapter -
NewName "WAN"
    }

elseif ($adaptercount -eq 2) {
    (Get-NetAdapter -Name "Ethernet" 2> $null) | Rename-NetAdapter -
NewName "WAN"
    (Get-NetAdapter -Name "Ethernet 2" 2> $null) | Rename-NetAdapter -
NewName "LAN"
}

# Clear potential static settings and default gateway
Set-NetIPInterface -InterfaceAlias "LAN" -Dhcp Enabled 2> $null
Get-NetIPAddress -InterfaceAlias "LAN" | Remove-NetRoute -
Confirm:$false 2> $null

$dns1 = '192.168.100.10'
$dns2 = '192.168.100.20'
$shpip = '192.168.100.40'

# Set the new IP settings + DNS

Write-Host "Setting up new IP settings..."

New-NetIPAddress -InterfaceAlias "LAN" -IPAddress $shpip -PrefixLength 24 -
DefaultGateway $dns1 -AddressFamily IPv4 > $null

Set-DnsClientServerAddress -InterfaceAlias "LAN" -ServerAddresses($dns1,$dns2)
Set-DnsClientServerAddress -InterfaceAlias "WAN" -ResetServerAddresses

# Enable domain sharing services
Write-Host "Enabling specific sharing services..."

Set-Service -Name "FDResPub" -StartupType "Automatic"
Start-Service -DisplayName "Function Discovery Resource Publication"

Set-Service -Name "Dnscache" -StartupType "Automatic" 2> $null
Start-Service -DisplayName "DNS Client"

Set-Service -Name "SSDPSRV" -StartupType "Automatic"
```

```
Start-Service -DisplayName "SSDP Discovery"

Set-Service -Name "upnphost" -StartupType "Automatic"
Start-Service -DisplayName "UPnP Device Host"

# Allow services through firewall
# Set action to allow
Get-NetFirewallRule -DisplayGroup "Network Discovery" | Set-NetFirewallRule -
Action Allow
# Enabling the rule
Get-NetFirewallRule -DisplayGroup "Network Discovery" | Enable-NetFirewallRule

Get-NetFirewallRule -DisplayGroup "File and Printer Sharing" | Set-
NetFirewallRule -Action Allow
Get-NetFirewallRule -DisplayGroup "File and Printer Sharing" | Enable-
NetFirewallRule

# Turn on network discovery
netsh advfirewall firewall set rule group= "Network Discovery" new enable=Yes

Write-Host "Complete..."
```

```
# Script 2 for WIN-EXC-SHP

Write-Host "Joining domain..."

$password = "Admin2019" | ConvertTo-SecureString -AsPlainText -Force

$credentials = New-
Object System.Management.Automation.PsCredential("VANLIEFFERINGE\Administrator
",$password)
Add-computer -DomainName "vanliefferinge.periode1" -
DomainCredential $credentials -Verbose

Write-Host "Complete..."
```

```
# Script 3 for WIN-EXC-SHP

# install prereq features
Write-Host "Installing Exchange prerequisites..."

Install-WindowsFeature Server-Media-Foundation
Install-WindowsFeature RSAT-ADDS
Install-WindowsFeature RSAT-Clustering-CmdInterface, NET-Framework-45-
Features, RPC-over-HTTP-proxy, RSAT-Clustering, RSAT-Clustering-
CmdInterface, RSAT-Clustering-Mgmt, RSAT-Clustering-PowerShell, Web-Mgmt-
```

```
Console, WAS-Process-Model, Web-Asp-Net45, Web-Basic-Auth, Web-Client-
Auth, Web-Digest-Auth, Web-Dir-Browsing, Web-Dyn-Compression, Web-Http-
Errors, Web-Http-Logging, Web-Http-Redirect, Web-Http-Tracing, Web-ISAPI-
Ext, Web-ISAPI-Filter, Web-Lgcy-Mgmt-Console, Web-Metabase, Web-Mgmt-
Console, Web-Mgmt-Service, Web-Net-Ext45, Web-Request-Monitor, Web-
Server, Web-Stat-Compression, Web-Static-Content, Web-Windows-Auth, Web-
WMI, Windows-Identity-Foundation, RSAT-ADDS

Set-ExecutionPolicy Bypass -Scope Process -Force; Invoke-Expression ((New-
Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps
1'))

Write-Host "Install complete..."

# autoconfirm chocolatey
choco feature enable -n=allowGlobalConfirmation

# install additional prereqs
Write-Host 'Installing .NET ...'
choco install dotnet4.7.2 -y

Write-Host 'Installing Visual C++ redistributables...'
choco install vcredist2013 -y

Write-Host 'Installing UCMA...'
choco install ucma4 -y

# copy scripts for easy access
Write-Host 'Copying scripts...'
$scripts = "C:\scripts"

# check paths
if(!(Test-Path $scripts)){
    mkdir $scripts
}

Copy-Item -r "c:\vagrant\provisioning\WIN-EXC-SHP\" "$scripts" -Force

# Set up auto logon
Write-Host 'Configuring auto logon as domain admin...'

$password='Admin2019'
$user='VANLIEFFERINGE\Administrator'

$secure = ConvertTo-SecureString $password -AsPlainText -Force
$credentials = New-
Object System.Management.Automation.PSCredential ($user, $secure)
```

```
$pw = $credentials.GetNetworkCredential().Password

Set-ItemProperty -
Path "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" -
Name DefaultUserName -Value $user
Set-ItemProperty -
Path "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" -
Name DefaultPassword -Value $pw
Set-ItemProperty -
Path "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" -
Name AutoAdminLogon -Value 1
Set-ItemProperty -
Path "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" -
Name ForceAutoLogon -Value 1

Set-ItemProperty -
Path 'HKLM:\Software\Microsoft\Windows\CurrentVersion\RunOnce' -
Name ResumeScript -
Value "C:\Windows\system32\WindowsPowerShell\v1.0\Powershell.exe -
executionpolicy bypass -file $scripts\WIN-EXC-SHP\4_exc_prep_ad.ps1"
```

```
# Script 4 for WIN-EXC-SHP

Write-Host "Preparing AD and schema..."

$iso = "c:\scripts\WIN-EXC-SHP\ExchangeServer2016-x64-cu14.iso"

#check for iso, if doesnt exist, download
Write-Host "checking or downloading iso..."
if (!(Test-Path("$iso"))) {
    (New-
Object System.Net.WebClient).DownloadFile("https://download.microsoft.com/down
load/f/4/e/f4e4b3a0-925b-4eff-8cc7-8b5932d75b49/ExchangeServer2016-x64-
cu14.iso","$iso")
}
# mount exchange iso
$isodrive = (Get-DiskImage -ImagePath $iso | Get-Volume)
if (!($isodrive)) {
    Write-Host 'Mounting iso...'
    Mount-DiskImage -ImagePath $iso
} else {
    Write-Host 'Already mounted...'
}

# set location on iso
$isodrive = (Get-DiskImage -ImagePath $iso | Get-Volume).DriveLetter
Set-Location "${isodrive}:"
```

```
# run preparation
Write-Host 'Preparing Schema...'
Invoke-
Expression "& .\setup.exe /PrepareSchema /IAcceptExchangeServerLicenseTerms"

Write-Host 'Preparing AD...'
Invoke-
Expression "& .\Setup.exe /PrepareAD /OrganizationName:'JVL' /IAcceptExchangeS
erverLicenseTerms"

Write-Host 'Preparing domains...'
Invoke-
Expression "& .\Setup.exe /IAcceptExchangeServerLicenseTerms /PrepareAllDomain
s"

# load next script
Set-ItemProperty -
Path 'HKLM:\Software\Microsoft\Windows\CurrentVersion\RunOnce' -
Name ResumeScript -
Value "C:\Windows\system32\WindowsPowerShell\v1.0\Powershell.exe -
executionpolicy bypass -file c:\scripts\WIN-EXC-SHP\5_exc_mail.ps1"

Restart-Computer
```

```
# Script 5 for WIN-EXC-SHP

# set up mail service
Write-Host "Installing Exchange server mail..."

$iso = "c:\scripts\WIN-EXC-SHP\ExchangeServer2016-x64-cu14.iso"

#check for iso, if doesnt exist, download
Write-Host "checking or downloading iso..."
if (!(Test-Path("$iso"))) {
    (New-
Object System.Net.WebClient).DownloadFile("https://download.microsoft.com/down
load/f/4/e/f4e4b3a0-925b-4eff-8cc7-8b5932d75b49/ExchangeServer2016-x64-
cu14.iso","$iso")
}
# mount exchange iso
$isodrive = (Get-DiskImage -ImagePath $iso | Get-Volume)
if (!($isodrive)) {
    Write-Host 'Mounting iso...'
    Mount-DiskImage -ImagePath $iso
} else {
    Write-Host 'Already mounted...'
}
```

```
$isodrive = (Get-DiskImage -ImagePath $iso | Get-Volume).DriveLetter
Set-Location "${isodrive}:"

Invoke-
Expression "& .\Setup.exe /mode:Install /role:Mailbox /OrganizationName:'JVL'
/IAcceptExchangeServerLicenseTerms"

Write-Host "Setting up mail..."
Add-PSSnapin Microsoft.Exchange.Management.PowerShell.SnapIn
$users = Get-ADUser -filter {userAccountControl -eq 512} -properties *
$users | ForEach-Object (enable-mailbox -Identity $_.Name -Database (get-
mailboxdatabase).name)

New-SendConnector -Name 'E-mail SMTP' -AddressSpaces * -Internet -
SourceTransportServer "WIN-EXC-SHP.vanliefferinge.periode1"
Install-WindowsFeature ADLDS

Write-Host "Mail setup complete..."

# load next script
Set-ItemProperty -
Path 'HKLM:\Software\Microsoft\Windows\CurrentVersion\RunOnce' -
Name ResumeScript -
Value "C:\Windows\system32\WindowsPowerShell\v1.0\Powershell.exe -
executionpolicy bypass -file c:\scripts\WIN-EXC-SHP\6_shp_prereq.ps1"

Restart-Computer
```

```
# Script 6 for WIN-EXC-SHP

$iso = "c:\scripts\WIN-EXC-SHP\officeserver.img"

Write-Host "Installing Sharepoint prereqs..."

Write-Host "checking or downloading iso..."
if (!(Test-Path("$iso"))) {
    (New-
Object System.Net.WebClient).DownloadFile("https://download.microsoft.com/down
load/0/0/4/004EE264-7043-45BF-99E3-3F74ECAE13E5/officeserver.img","$iso")
}
# mount exchange iso
$isodrive = (Get-DiskImage -ImagePath $iso | Get-Volume)
if (!($isodrive)) {
    Write-Host 'Mounting iso...'
    Mount-DiskImage -ImagePath $iso
```

```
} else {
    Write-Host 'Already mounted...'
}

# set location on iso
$isodrive = (Get-DiskImage -ImagePath $iso | Get-Volume).DriveLetter
Set-Location "${isodrive}:"

# install prereqs
Write-Host 'Installing prereqs...'
Start-Process prerequisiteinstaller.exe /unattended -wait
Import-Module Servermanager
Install-WindowsFeature Net-Framework-Core, NET-HTTP-Activation, NET-Non-HTTP-
Activ, NET-WCF-HTTP-Activation45, Web-Common-Http, Web-Static-Content, Web-
Default-Doc, Web-Dir-Browsing, Web-Http-Errors, Web-App-Dev, Web-Asp-Net, Web-
ISAPI-Ext, Web-ISAPI-Filter, Web-Health, Web-Http-Logging, Web-Log-
Libraries, Web-Request-Monitor, Web-Http-Tracing, Web-Security, Web-Basic-
Auth, Web-Filtering, Web-Digest-Auth, Web-Performance, Web-Stat-
Compression, Web-Dyn-Compression, Web-Mgmt-Tools, Web-Mgmt-Console, Web-Mgmt-
Compat, Web-Metabase, Web-Lgcy-Scripting, Windows-Identity-Foundation, Server-
Media-Foundation, Xps-Viewer, BITS-IIS-Ext, WinRM-IIS-Ext, Web-Scripting-
Tools, Web-WMI, Web-IP-Security, Web-url-Auth, Web-Cert-Auth, Web-Client-Auth
Install-WindowsFeature Web-Server -IncludeAllSubFeature -
IncludeManagementTools
Install-WindowsFeature Was -IncludeAllSubFeature

#load next script
Set-ItemProperty -
Path 'HKLM:\Software\Microsoft\Windows\CurrentVersion\RunOnce' -
Name ResumeScript -
Value "C:\Windows\system32\WindowsPowerShell\v1.0\Powershell.exe -
executionpolicy bypass -file c:\scripts\WIN-EXC-SHP\7_shp_setup.ps1"

Restart-Computer
```

```
# Script 7 for WIN-EXC-SHP

Write-Host "Installing Sharepoint..."

#set location on iso
$iso = "c:\scripts\WIN-EXC-SHP\officeserver.img"

Write-Host "Installing Sharepoint prereqs..."

Write-Host "checking or downloading iso..."
if (!(Test-Path("$iso"))) {
```

```
    (New-
Object System.Net.WebClient).DownloadFile("https://download.microsoft.com/down
load/0/0/4/004EE264-7043-45BF-99E3-3F74ECAE13E5/officeserver.img","$iso")
}
# mount exchange iso
$isodrive = (Get-DiskImage -ImagePath $iso | Get-Volume)
if (!($isodrive)) {
    Write-Host 'Mounting iso...'
    Mount-DiskImage -ImagePath $iso
} else {
    Write-Host 'Already mounted...'
}


$isodrive = (Get-DiskImage -ImagePath $iso | Get-Volume).DriveLetter
Set-Location "${isodrive}:"

#run install
Start-Process ".\setup.exe" -ArgumentList "/config `"c:\scripts\WIN-EXC-
SHP\shp-install-config.xml`"" -WindowStyle Minimized -wait | Out-Null
Start-Process ".\setup.exe" -ArgumentList "/config `"c:\scripts\WIN-EXC-
SHP\shp-install-config.xml`"" -WindowStyle Minimized -wait | Out-Null

Add-PSSnapIn Microsoft.SharePoint.PowerShell

Write-Host "install complete..."

# load next script
& "c:\scripts\WIN-EXC-SHP\8_shp_farm.ps1"
```

```
# Script 8 for WIN-EXC-SHP

Write-Host "Configuring sharepoint farm..."

Add-PsSnapin Microsoft.SharePoint.PowerShell | Out-Null
Start-SPAssignment -Global | Out-Null

$password = ConvertTo-SecureString "Admin2019" -AsPlainText -Force
$credentials = New-
Object System.Management.Automation.PSCredential ("VANLIEFFERINGE\Administrato
r",$password)

Write-Host 'Creating database...'
New-SPConfigurationDatabase -DatabaseServer "WIN-SQL-SCCM" -
DatabaseName "SHP_conf" -AdministrationContentDatabaseName "SP2016_conf" -
Passphrase $password -FarmCredentials $credentials -
localserverrole "SingleServerFarm"
```

```
# install features
Write-Host "Installing features..."
Install-SPHelpCollection -All
Initialize-SPResourceSecurity
Install-SPService
Install-SPFeature -AllExistingFeatures
Install-SPApplicationContent
New-SPCentralAdministration -Port 8080 -WindowsAuthProvider "NTLM"
Write-Host "install complete..."

# load next script
& "c:\scripts\WIN-EXC-SHP\9_shp_webapp.ps1"
```

```
# Script 9 for WIN-EXC-SHP

Write-Host "Configuring webapp..."

Add-PsSnapin "Microsoft.SharePoint.PowerShell" -EA 0

#vars
$AppPoolAccount = "VANLIEFFERINGE\Administrator"
$ApplicationPoolName ="SharePoint - 8081"
$ContentDatabase = "SharePoint_ContentDB"
$DatabaseServer = "WIN-SQL-SCCM"
$Url = "http://WIN-EXC-SHP:8081/"
$Name = "WIN-EXC-SHP - Documents"
$Description = "SharePoint Site"
$SiteCollectionTemplate = 'STS#0'

# set up webapp
Write-Host 'Creating New-SPWebApplication...'
New-SPWebApplication -ApplicationPool $ApplicationPoolName -
ApplicationPoolAccount (Get-SPManagedAccount $AppPoolAccount) -
Name $Description -AuthenticationProvider (New-SPAuthenticationProvider -
UseWindowsIntegratedAuthentication) -DatabaseName $ContentDatabase -
DatabaseServer $DatabaseServer -URL $Url

Write-Host 'Creating New-SPSite...'
New-SPSite -Url $Url -Name $Name -Description $Description -
OwnerAlias $AppPoolAccount -Template $SiteCollectionTemplate


$w = Get-SPWebApplication $Url
$w.Properties["portalsuperuseraccount"] = $AppPoolAccount
$w.Properties["portalsuperreaderaccount"] = $AppPoolAccount
$w.Update()
```

```
Write-Host "install complete..."

iisreset /restart
```

WIN-CLT

```
# Script 1 for WIN-CLT

Write-Host "Joining domain..."

$password = "Admin2019" | ConvertTo-SecureString -AsPlainText -Force

$credentials = New-
Object System.Management.Automation.PsCredential("VANLIEFFERINGE\JensVL",$pass
word)
Add-computer -DomainName "vanliefferinge.periode1" -
DomainCredential $credentials -Verbose

Write-Host "Complete..."
```

Vagrant files

```
# vagrant_hosts.yml
#
# List of hosts to be created by Vagrant. This file controls the Vagrant
# settings, specifically host name and network settings. You should at least
# have a `name:`.  Other optional settings that can be specified:
#
# * `box`: choose another base box instead of the default one specified in
#          Vagrantfile. A box name in the form `USER/BOX` (e.g.
#          `bertvv/centos72`) is fetched from Atlas.
# * `box_url`: Download the box from the specified URL instead of from Atlas.
# * `ip`: by default, an IP will be assigned by DHCP. If you want a fixed
#         addres, specify it.
# * `netmask`: by default, the network mask is `255.255.255.0`. If you want
#              another one, it should be specified.
# * `mac`: The MAC address to be assigned to the NIC. Several notations are
#          accepted, including "Linux-style" (`00:11:22:33:44:55`) and
#          "Windows-style" (`00-11-22-33-44-55`). The separator characters can
#          be omitted altogether (`001122334455`).
# * `intnet`: If set to `true`, the network interface will be attached to an
#             internal network rather than a host-only adapter.
# * `auto_config`: If set to `false`, Vagrant will not attempt to configure
#                  the network interface.
# * `synced_folders`: A list of dicts that specify synced folders. `src` and
#   `dest` are mandatory, `options:` are optional. For the possible options,
#   see the Vagrant documentation[1]. Keys of options should be prefixed with
#   a colon, e.g. `:owner:`.
#
# To enable *provisioning*, add these hosts to site.yml and assign some roles.
#
# [1] http://docs.vagrantup.com/v2/synced-folders/basic_usage.html
---

- name: WIN-DC1
  box: gusztavvargadr/windows-server
  version: '1809.0.1910-standard'
  ip: 192.168.100.10
  lan_prefix: 24
  intnet: true
  gui: false
  cpus: 2
  memory: 2048
  primary_dns: 192.168.100.10
  secondary_dns: 192.168.100.20
  default_gateway: 192.168.100.10
  ip_win_sql_sccm: 192.168.100.30
  ip_win_exc_shp: 192.168.100.40
  domain: 'vanliefferinge.periode1'
```

```
  netbios: 'VANLIEFFERINGE'
  admin_pw: 'Admin2019'
  debug_output: 'yes'
  forwarded_ports:
    - guest: '3389'
      host: '3390'

- name: WIN-DC2
  box: gusztavvargadr/windows-server
  version: '1809.0.1910-standard'
  ip: 192.168.100.20
  lan_prefix: 24
  intnet: true
  gui: false
  cpus: 2
  memory: 2048
  hostname_dc1: WIN-DC1
  primary_dns: 192.168.100.10
  secondary_dns: 192.168.100.20
  default_gateway: 192.168.100.10
  domain: 'vanliefferinge.periode1'
  netbios: 'VANLIEFFERINGE'
  admin_pw: 'Admin2019'
  debug_output: 'yes'
  forwarded_ports:
    - guest: '3389'
      host: '3391'

- name: WIN-SQL-SCCM
  box: gusztavvargadr/windows-server
  ip: 192.168.100.30
  lan_prefix: 24
  intnet: true
  gui: false
  cpus: 2
  memory: 8192
  primary_dns: 192.168.100.10
  secondary_dns: 192.168.100.20
  default_gateway: 192.168.100.10
  domain: 'vanliefferinge.periode1'
  domain_user: 'VANLIEFFERINGE\Administrator'
  domain_pw: 'Admin2019'
  sqlusername: 'VANLIEFFERINGE\Administrator'
  sqlpassword: 'Admin2019'
  debug_output: 'yes'
  downloadpath: 'C:\SetupMedia'
  forwarded_ports:
    - guest: '3389'
```

```
      host: '3393'

- name: WIN-EXC-SHP
  box: gusztavvargadr/windows-server
  version: '1607.0.1909-standard'
  ip: 192.168.100.40
  lan_prefix: 24
  intnet: true
  gui: false
  cpus: 2
  memory: 8192
  primary_dns: 192.168.100.10
  secondary_dns: 192.168.100.20
  default_gateway: 192.168.100.10
  domain: 'vanliefferinge.periode1'
  domain_user: 'VANLIEFFERINGE\Administrator'
  domain_pw: 'Admin2019'
  netbios: 'VANLIEFFERINGE'
  debug_output: 'yes'
  forwarded_ports:
    - guest: '3389'
      host: '3394'

- name: WIN-CLT1
  box: gusztavvargadr/windows-10
  intnet: true
  gui: true
  cpus: 2
  memory: 4096
  domain: 'vanliefferinge.periode1'
  domain_user: 'VANLIEFFERINGE\JensVL'
  domain_pw: 'Admin2019'
  debug_output: 'yes'
  forwarded_ports:
    - guest: '3389'
      host: '3395'
```

```
# vagrantfile
# -*- mode: ruby -*-
# vi: ft=ruby :

require 'rbconfig'
require 'yaml'

# Set your default base box here
DEFAULT_BASE_BOX = 'bertvv/centos72'
```

```
#
# No changes needed below this point
#

VAGRANTFILE_API_VERSION = '2'
PROJECT_NAME = '/' + File.basename(Dir.getwd)

hosts = YAML.load_file('vagrant-hosts.yml')

# {{{ Helper functions

def is_windows
  RbConfig::CONFIG['host_os'] =~ /mswin|mingw|cygwin/
end

# Set options for the network interface configuration. All values are
# optional, and can include:
# - ip (default = DHCP)
# - netmask (default value = 255.255.255.0
# - mac
# - auto_config (if false, Vagrant will not configure this network interface
# - intnet (if true, an internal network adapter will be created instead of a
#   host-only adapter)
def network_options(host)
  options = {}

  if host.has_key?('ip')
    options[:ip] = host['ip']
    options[:netmask] = host['netmask'] ||= '255.255.255.0'
  else
    options[:type] = 'dhcp'
  end

  if host.has_key?('mac')
    options[:mac] = host['mac'].gsub(/[-:]/, '')
  end
  if host.has_key?('auto_config')
    options[:auto_config] = host['auto_config']
  end
  if host.has_key?('intnet') && host['intnet']
    options[:virtualbox__intnet] = true
  end

  options
end

def custom_synced_folders(vm, host)
  if host.has_key?('synced_folders')
```

```
    folders = host['synced_folders']

    folders.each do |folder|
      vm.synced_folder folder['src'], folder['dest'], folder['options']
    end
  end
end

# Adds forwarded ports to your Vagrant machine
#
# example:
#  forwarded_ports:
#    - guest: 88
#      host: 8080
def forwarded_ports(vm, host)
  if host.has_key?('forwarded_ports')
    ports = host['forwarded_ports']

    ports.each do |port|
      vm.network "forwarded_port", guest: port['guest'], host: port['host']
    end
  end
end


# }}}

Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
  config.ssh.insert_key = false
  hosts.each do |host|
    config.vm.define host['name'] do |node|
      node.vm.box = host['box'] ||= DEFAULT_BASE_BOX
      node.vm.box_version = host['version']
      if(host.key? 'box_url')
        node.vm.box_url = host['box_url']
      end

      node.vm.hostname = host['name']
      node.vm.network :private_network, network_options(host)
      custom_synced_folders(node.vm, host)
      forwarded_ports(node.vm, host)

      node.vm.provider :virtualbox do |vb|
        vb.gui    = host['gui']
        vb.cpus   = host['cpus'] if host.key? 'cpus'
        vb.memory = host['memory'] if host.key? 'memory'

        # WARNING: if the name of the current directory is the same as the
        # host name, this will fail.
```

```
        vb.customize ['modifyvm', :id, '--groups', PROJECT_NAME]
        if host['name'] == "WIN-CLT"
          vb.customize [
            'modifyvm', :id,
            '--boot1', 'net',
            '--boot2', 'disk',
            '--boot3', 'none',
            '--boot4', 'none'
          ]
        end
      end

      # use the plaintext WinRM transport and force it to use basic authentica
tion.
      # This is needed because the default negotiate transport stops working
      #    after the domain controller is installed.
      #    see https://groups.google.com/forum/#!topic/vagrant-up/sZantuCM0q4
      config.winrm.transport       = :plaintext
      config.winrm.basic_auth_only = true
      config.winrm.timeout         = 3600 # 60 minutes
      config.vm.boot_timeout       = 3600 # 60 minutes

      if host['name'] == "WIN-DC1"
        provision_files="provisioning/WIN-DC1/"

        node.vm.provision 'shell',
          privileged: true,
          path: provision_files + "/1_adapter.ps1",
          args: []
        node.vm.provision 'shell',
          privileged: true,
          path: provision_files + "/2_dc_install.ps1",
          args: []
        node.vm.provision 'shell', reboot: true
        node.vm.provision 'shell',
          privileged: true,
          path: provision_files + "/3_dc_conf.ps1",
          args: []
        node.vm.provision 'shell',
          privileged: true,
          path: provision_files + "/4_dns_fix.ps1",
          args: []
        node.vm.provision 'shell',
          privileged: true,
          path: provision_files + "/5_dns_config.ps1",
          args: []
      elsif host['name'] == "WIN-DC2"
        provision_files="provisioning/WIN-DC2/"
```

```
    node.vm.provision 'shell',
      privileged: true,
      path: provision_files + "/1_adapter.ps1",
      args: []
    node.vm.provision 'shell',
      privileged: true,
      path: provision_files + "/2_dc_install.ps1",
      args: []
    node.vm.provision 'shell', reboot: true
    node.vm.provision 'shell',
      privileged: true,
      path: provision_files + "/3_dns_fix.ps1",
      args: []
  elsif host['name'] == "WIN-CLT1"
    node.vm.provision 'shell',
      privileged: true,
      path: "provisioning/1_client.ps1",
      args: []
    node.vm.provision 'shell', reboot: true
  elsif host['name'] == "WIN-EXC-SHP"
    provision_files="provisioning/WIN-EXC-SHP/"

    node.vm.provision 'shell',
      privileged: true,
      path: provision_files + "/1_adapter.ps1",
      args: []
    node.vm.provision 'shell',
      privileged: true,
      path: provision_files + "/2_join_domain.ps1",
      args: []
    node.vm.provision 'shell',
      privileged: true,
      path: provision_files + "/3_exc_prereq.ps1",
      args: []
    node.vm.provision 'shell', reboot: true
  elsif host['name'] == "WIN-SQL-SCCM"
    provision_files="provisioning/WIN-SQL-SCCM/"

    node.vm.provision 'shell',
      privileged: true,
      path: provision_files + "/1_adapter.ps1",
      args: []
    node.vm.provision 'shell',
      privileged: true,
      path: provision_files + "/2_join_domain.ps1",
      args: []
    node.vm.provision 'shell', reboot: true
```

```
        node.vm.provision 'shell',
          privileged: true,
          path: provision_files + "/3_sql_install.ps1",
          args: []
        node.vm.provision 'shell', reboot: true
      end
    end
  end
end
```

Resources

WIN-DC1 / DC2

- http://www.rebeladmin.com/2018/10/step-step-guide-install-active-directory-windows-server-2019-powershell-guide/
- http://pc-addicts.com/join-windows-10-to-domain-server-2016/
- https://www.faqforge.com/windows/configure-dhcp-powershell/
- http://www.sambrentnall.co.uk/GatewayOnHyperVCore/
- https://www.server-world.info/en/note?os=Windows_Server_2019&p=dns&f=1
- https://www.manageengine.com/products/active-directory-audit/kb/how-to/how-to-check-if-domain-controllers-are-in-sync-with-each-other.html
- https://www.itprotoday.com/windows-78/how-can-i-quickly-determine-whether-domain-controller-dc-available-specific-domain

WIN-EXC-SHP

- https://www.starwindsoftware.com/blog/installing-exchange-server-2016-on-windows-server-2016
- https://www.prajwaldesai.com/step-by-step-guide-to-install-exchange-server-2016/
- https://www.starwindsoftware.com/blog/managing-exchange-server-2016-using-powershell
- https://blogit.create.pt/miguelisidoro/2018/07/28/how-to-install-a-sharepoint-2016-farm-using-powershell-and-autospinstaller-part-1/
- https://blogit.create.pt/miguelisidoro/2018/07/28/how-to-install-a-sharepoint-2016-farm-using-powershell-and-autospinstaller-part-2/
- http://blog.kuppens-switsers.net/sharepoint/sharepoint-2016-farm-configuration-using-powershell/
- https://github.com/brianlala/AutoSPSourceBuilder
- http://www.luisevalencia.com/2016/09/25/installing-sharepoint-server-2016-with-powershell/
- https://docs.microsoft.com/en-us/powershell/sharepoint/?view=sharepoint-ps
- https://cann0nf0dder.wordpress.com/2016/08/30/building-sharepoint-2016-development-environment-part-8-installing-sql-2016-ready-for-sharepoint-2016/
- https://www.itprotoday.com/email-and-calendaring/how-install-microsoft-exchange-server-2016-windows-server-2016-powershell
- https://www.kerneldatarecovery.com/blog/fix-exchange-server-is-in-inconsistent-state-error/

WIN-SQL-SCCM

- https://sqlplayer.net/2019/01/unattended-installation-of-sql-server/
- https://hinchley.net/articles/install-and-configure-microsoft-deployment-toolkit-2013-using-powershell/
- https://hinchley.net/articles/install-microsoft-sccm-2012-r2-sp1/
- https://ss64.com/nt/powercfg.html
- https://octopus.com/blog/automate-sql-server-install
- https://www.youtube.com/watch?v=4zwQsQEtrwY
- https://www.prajwaldesai.com/install-sccm-distribution-point-using-powershell-script/
- https://www.youtube.com/watch?v=dBARKz-3mf4
- http://www.checkyourlogs.net/?p=26481

- https://www.youtube.com/watch?v=lpsUbaboULc
- http://damcuvelier.over-blog.net/2017/06/sccm-2016-silent-auto-install-of-sccm2016-mdt2013u2.html
- https://www.youtube.com/watch?v=Mht4mPFRwk8
- https://www.youtube.com/watch?v=-Tia-awekGQ
- https://docs.microsoft.com/en-us/powershell/sccm/overview?view=sccm-ps
- https://www.youtube.com/watch?v=wFgSkXs2r84
- https://douwevanderuit.wordpress.com/2014/02/10/howto-install-a-primary-site-server-and-all-prereqs-with-powershell/
- https://www.youtube.com/watch?v=RAFFT9GquSw