# Server Build Out Procedures V2 – High Detail description

## Use Chef to generate a cookbook

On the lab box with HyperV and ChefDK installed

From d:\Chef using ChefDK, run the following commands. (Substitute ‘ServerX’ for [ServerName])

* chef generate cookbook [ServerName]
* chef generate template [ServerName] server-info.txt
* chef generate file [ServerName] Scripts

## Customize the cookbook

Be sure to run as administrator when editing default/existing cookbook files

One simple way to do this is to Run Visual Studio Code as Administrator and within it open the folder D:\chef\ServerX

Template

Erb contents:

fqdn: <%= node['fqdn'] %>  
hostname: <%= node['hostname'] %>  
platform: <%= node['platform'] %> - <%= node['platform\_version'] %>  
cpu count: <%= node['cpu']['total'] %>

This is a good example of using Attributes when built, this file is in the C:\Temp folder of the VM and will read something like this:

fqdn: Server8.sea.corp.expecn.com  
hostname: Server8  
platform: windows - 6.3.9600  
cpu count: 2

File

Copy PowerShell scripts to D:\chef\[ServerName]\files\default.

* VMScript1.ps1
* VMScript2.ps1
* MyVMCommands.psm1

These files are located in this repository. The first two are called from the new VM to customize it during the install/config process. The third file is a PS Module and contains functions to be called during normal operations of the VM in the lab. These will examined more closely in a bit.

Kitchen.yml

---

driver:

name: hyperv

parent\_vhd\_folder: D:\HyperVResources\VMs\BaseBox

parent\_vhd\_name: BaseBox.vhdx

vm\_switch: ExternalSwitch

memory\_startup\_bytes: 2GB

provisioner:

name: chef\_zero

transport:

password: H0rnyBunny

platforms:

- name: windows-2012r2

suites:

- name: default

run\_list:

- recipe[[ServerName]::default]

attributes:

Recipes Default.rb

#

# Cookbook:: Server9

# Recipe:: default

#

# Copyright:: 2017, The Authors, All Rights Reserved.

# Create new directories in the new machine

directory 'C:\temp'

directory 'C:\Scripts'

directory 'C:\Program Files\WindowsPowerShell\Modules\MyVmCommands'

# Copy the Template

template 'C:\temp\server-info.txt' do

source 'server-info.txt.erb'

end

# Copy the files

cookbook\_file 'C:\Program Files\WindowsPowerShell\Modules\MyVmCommands\MyVmCommands.psm1' do

source 'MyVmCommands.psm1'

end

cookbook\_file 'C:\Scripts\VMScript1.ps1' do

source 'VMScript1.ps1'

end

cookbook\_file 'C:\Scripts\VMScript2.ps1' do

source 'VMScript2.ps1'

end

## Converge to build the VM

Copy HostScript1 and 2 to root of new Server D:\Chef\[Servername] and edit

from d:\chef\serverXX\ kitchen converge

## Post Converge Scripts/Configuration/Chef-Client

Host Script 1

* Must be edited
* Rename the VM

$VMNname = 'ServerX\_2012R2'

# Rename the VM

Rename-VM -Name default-windows-2012r2 -NewName $VMNname

VM Script 1

* Must be edited
* Rename the Computer
* Rename the included NIC (ExternalNIC)
* Stop the VM OS

# VMScript1

# Settings

$ServerName = 'ServerX'

# Rename the computer

Rename-Computer -NewName $ServerName

# Rename the existing (External) NIC

Get-NetAdapter | Rename-NetAdapter -NewName ExternalNIC

# Shutdown VM so a Host script can add the next NIC

Stop-Computer

Host Script 2

* Must be edited
* Adds a second NIC (InternalSwitch)
* Starts the VM

# HostScript2

$VMNname = 'ServerX\_2012R2'

# Add VM Network Adapter

Add-VMNetworkAdapter -VMName $VMNname -SwitchName InternalSwitch

# Start VM

Get-VM -Name $VMNname | Start-VM

VM Script 2

* Must be editied
* Rename Internal NIC, Statically address it
* Configure to run internal NiC only using PS function in module that should be there
* Configure RDP
* Join the Domain

# Settings

$IP = '192.168.0.1XX'

# Rename new Internal NIC.

Get-NetAdapter | Where-Object Name -ne 'ExternalNIC' | Rename-NetAdapter -NewName InternalNIC

# Set InternalNIC static IP address

Get-NetAdapter | Where-Object Name -eq 'InternalNIC' | New-NetIPAddress -PrefixLength 24 -IPAddress $IP

# Set DNS on InternalNIC

# Get-NetAdapter | Where-Object Name -eq 'InternalNIC' | Set-DnsClientServerAddress -ServerAddresses ('192.168.0.101')

Set-InternalDNS '192.168.0. 1XX,192.168.0. 1XX

# Disable External NIC

Set-MyVmNetwork -Toggle InternalOnly

# Enable Remote Desktop

Enable-RemoteDesktop

# Set the firewall for PS Remoting

Get-NetFirewallPortFilter | ?{$\_.LocalPort -eq 5985 } | Get-NetFirewallRule | ?{ $\_.Direction –eq "Inbound" -and $\_.Profile -eq "Public" -and $\_.Action –eq "Allow"} | Set-NetFirewallRule -RemoteAddress "Any"

Modules

MyVmCommands.psm1

<#

.Synopsis

Toggles Internal/External NICs

.EXAMPLE

Set-MyVmNetwork -Toggle InternalOnly

.EXAMPLE

Set-MyVmNetwork -Toggle ExternalOnly

#>

Function Set-MyVmNetwork

{

[CmdletBinding()]

Param

(

# Param1 help description

[Parameter(Mandatory=$true)]

[ValidateSet("InternalOnly", "ExternalOnly")]

$Toggle,

[switch]

$Quiet

)

switch ($Toggle)

{

'InternalOnly'

{

Enable-NetAdapter -Name InternalNIC -Confirm:$false

Disable-NetAdapter -Name ExternalNIC -Confirm:$false

# $FWStatus = Set-NetFirewallProfile -Enabled False -Name Public -PassThru

}

'ExternalOnly'

{

# $FWStatus = Set-NetFirewallProfile -Enabled True -Name Public -PassThru

Enable-NetAdapter -Name ExternalNIC -Confirm:$false

Disable-NetAdapter -Name InternalNIC -Confirm:$false

}

}

# "Public Firewall " + ($FWStatus).Enabled

If ($Quiet -eq $false) {Get-NetAdapter | select name, status}

}

Function Set-InternalDNS

{

[CmdletBinding()]

Param

(

[Parameter(Mandatory=$true)]

$DNSIPAdresses

)

Get-NetAdapter | Where-Object Name -eq 'InternalNIC' | Set-DnsClientServerAddress -ServerAddresses ($DNSIPAdresses)

}

Function Enable-RemoteDesktop

{

# Enable Remote Desktop

set-ItemProperty -Path 'HKLM:\System\CurrentControlSet\Control\Terminal Server'-name "fDenyTSConnections" -Value 0

# Allow incoming RDP on firewall

Enable-NetFirewallRule -DisplayGroup "Remote Desktop"

# Enable secure RDP authentication

set-ItemProperty -Path 'HKLM:\System\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp' -name "UserAuthentication" -Value 1

}

## Automation Goals:

* Parameterize Scripts so that can be called with ComputerName, IP Address etc

Try moving some of the functionality of VMScript 1 & 2 into a Module

Tools.psm1

## Chef Bootstrap, Role and Cookbooks configuration

### Bootstrap ChefDK client

Run from ChefDK server, Server9

Does this have the right credentials when I have not joined the new server to the domain?

knife bootstrap windows winrm 192.168.0.1XX --winrm-user coatelab\administrator --winrm-password 'xxxxxxxxx' --node-name serverX.coatelab.com --run-list 'role[web]' --msi-url <http://server8.coatelab.com/chef-client-12.18.31-1-x64.msi>

Role: Web

chef-client::default (Sched Task go every 5 min)  
chef-client::delete\_validation

learn\_chef\_iis::default

Default Recipe

powershell\_script 'Install IIS' do

code 'Add-WindowsFeature Web-Server'

guard\_interpreter :powershell\_script

not\_if '(Get-WindowsFeature -Name Web-Server).Installed'

end

service 'w3svc' do

action [:enable, :start]

end

directory 'c:\inetpub\wwwroot' do

rights :read, 'IIS\_IUSRS'

recursive true

action :create

end

template 'c:\inetpub\wwwroot\Default.htm' do # ~FC033

source 'Default.htm.erb'

end

cookbook\_file 'C:\Scripts\NewScript.ps1' do

source 'NewScript.ps1'

end

Daves-Test-Cookbook

Default Recipe

registry\_key 'HKEY\_LOCAL\_MACHINE\Software\Microsoft\ServerManager' do

values [{

:name => 'DoNotOpenServerManagerAtLogon',

:type => :dword,

:data => 1

}]

action :create

end

ad-join

Library Cookbook

JoinCoatelabDomain

Default Recipe

include\_recipe 'ad-join::default'

#The parameter in the first line is ignored if the server is not to be renamed

domain\_join node['hostname'] do

domain 'coatelab.com'

domain\_user 'coateds'

domain\_password 'H0rnyBunny'

ou nil

action :join

end

test\_powershell

Default Recipe

#

# Cookbook:: test\_powershell

# Recipe:: default

#

# Copyright:: 2017, The Authors, All Rights Reserved.

# This is the Cheffy way to make it indempotent

powershell\_script 'make\_test\_file' do

guard\_interpreter :powershell\_script

code 'New-Item -Type File -Path c:/scripts/cheftestfile.txt'

not\_if 'test-path c:/scripts/cheftestfile.txt'

end

# This is the PowerShell way to make it indempotent

# powershell\_script 'make\_test\_file' do

# code <<-EOH

# $File = 'c:/scripts/cheftestfile.txt'

# If (!(Test-Path $File)) {New-Item -Type File -Path $File}

# EOH

# end

# This works

powershell\_script 'set\_dns' do

code <<-EOH

Set-InternalDNS '192.168.0.110,192.168.0.101'

EOH

end

powershell\_script 'check\_wins' do

guard\_interpreter :powershell\_script

code 'Set-Content -Path c:/scripts/cheftestfile.txt -Value 1'

only\_if 'get-service WINS'

end

From ChefDK PS prompt (As Administrator)

In D:\chef

Edit variables at top of CreateNewServer.ps1

.\CreateNewServer.ps1 (execute)

(Currently recipes\default.rb is built manually)

Cd to d:\chef\[servername]

Kitchen converge

Run HostScript1, VMScript1, HostScript2,VMScript2

Bootstrap from server9