Automating VDI

Who am I?

- Chris Hildebrandt
- Sr. Global VDI Engineer
- vExpert
- VMware EUC Champion
- Kansas City VMUG Leader
- Twitter @childebrandt42
- Blog https://childebrandt42. com/
- GitHub https://github.com/childebrandt42/

Why Automate Daily Tasks

- Fast
- Reliable
- Reusable
- Enhance your logging and reporting
- **If you can automate yourself out of a job; You can get a better job!**

Who in here has ever automated any VDI processes?

Define the problem you are trying to solve!

Challenges

- 2 people take 2 months to update 50 pools 6 times a year.
  - Each pool takes 2 to 3 hours update.
- Hand checked Check list means missed steps.
- Average Ticket Count: 24 tickets per update cycle
  - MTTR: 2 - 3 hours. (includes Pool Refresh)
- Time Wasted (Average): **360 hours!!**

Step 1: Create A Plan

- What needs to happen? Why? Use your Task List!
- Does it have to be *exactly* that way?
- Does it really?
- What tools do you already have?
- What tools do you need?
- THINK OUTSIDE THE BOX

Plan A bad idea's


```
#Set AutoLog on and auto kick off step3
$RegPath = "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon"
$RegROPath = "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce"
Set-ItemProperty $RegPath "AutoAdminLogon" -Value "1" -type String
Set-ItemProperty $RegPath "DefaultUsername" -Value "$VDadminUsername" -type String
Set-ItemProperty $RegPath "DefaultPassword" -Value "$VDIadminPassword" -type String
Set-ItemProperty $RegROPath "(Default)" -Value "$ShutdownScript" -type String
```

Step 2: Scrap Plan A, Create Plan B

- Look at what you are trying to do?
- Look at why you are trying to do it that way.
- Does it need to be that way?
- Does it really?
- Are you sure it needs to be that way? Why?
- DON'T BE AFRAID OF CHANGE! Embrace it!

Questions to Ponder:

- How to update the Windows?
- How to update 3rd Party applications?
- How to cut down the time to update images?
- How to cut down on the Post Refresh Incidents
- Ways to cut down human error.

My answer to the Questions!

My Answers:

- Use SCCM for Windows updates
- Use Software Center for 3rd Party Applications
- Using SCCM, Software Center and Adaptiva for complete image builds.
- Automate the process including creation of the Service Now Incidents and Changes.

Update Process

- Start with Powered On Image
- Make API call to install Microsoft and 3rd Party application Updates
- Reboot the VM
- Run SEP prep tool
- Run Optimization Tool
- Run Defrag
- Update Custom Reg Keys
- Disk Cleanup
- Disable SCCM Services
- Update vCenter Notes
- Power Down VM
- Take Snapshot
- Power on the Master
- Recompose Pool
- Turn back on SCCM services.

Snippet from Cleanup Script

```
#Run Disk Cleanup to remove temp files, empty recycle bin and remove other unneeded files
Start-Process -Filepath "c:\windows\system32\cleanmgr" -argumentlist '/sagerun:1'
# Enable Defrag Service and Defrag C drive, stop Defrag Service.
Set-Service -Name "defragsvc" -StartupType Automatic
start-Service -Name "defragsvc"
Start-sleep -Seconds 90
Optimize-Volume -DriveLetter C -Analyze -Defrag
stop-Service -Name "defragsvc"
Set-Service -Name "defragsvc" -StartupType disabled
# Clear all event logs
wevtutil el | Foreach-Object {wevtutil cl "$_"}
# Pre-complile .NET framework Assemblies
Start-Process -Filepath "C:\Windows\Microsoft.NET\Framework\v4.0.30319\ngen.exe" -argumentlist 'update','/force'
New-ItemProperty -Name verbosestatus -Path 'HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\System' -PropertyType REG_DWORD
# Check to see if SEP is installed and prep for Recompose
$CheckSEP = get-wmiobject win32_Service | Where {$_.Name -eq "SepMasterService"} | select Name
if($CheckSEP.Name -eq 'SepMasterService')
{…
# Run VMware optimizer
if($RunOptimizer -eq '1')
{
start /wait $OptimizerLocation -t $OptimizerTemplate -r $OptimizerTemplateLocation
start /wait $OptimizerLocation -o recommended -t $OptimizerTemplate -v > $OptimizerLogFile 2>&1
start /wait $OptimizerLocation -t $OptimizerTemplate -r $OptimizerTemplateLocation
}
stop-Service -Name "wuauserv"
Set-Service -Name "wuauserv" -StartupType disabled
ipconfig /release
ipconfig /flushdns
shutdown -s
```

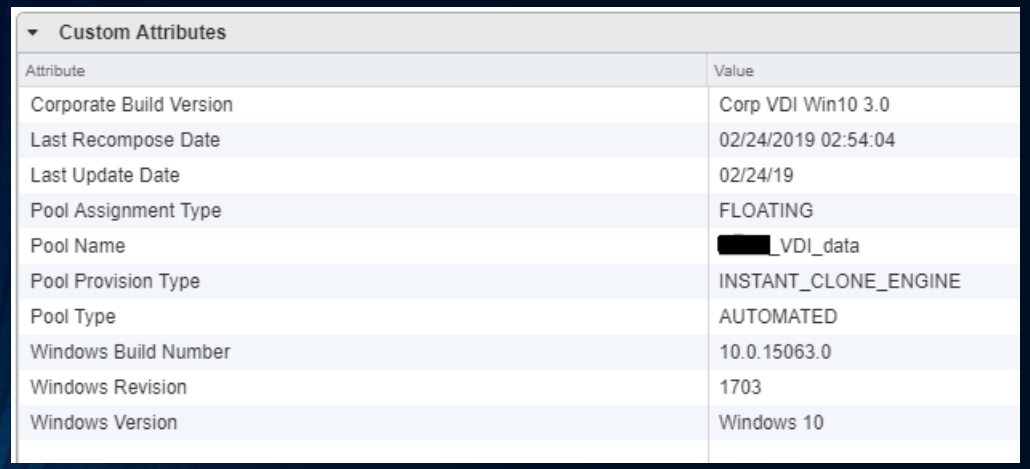
Snippet from Guts of main Script

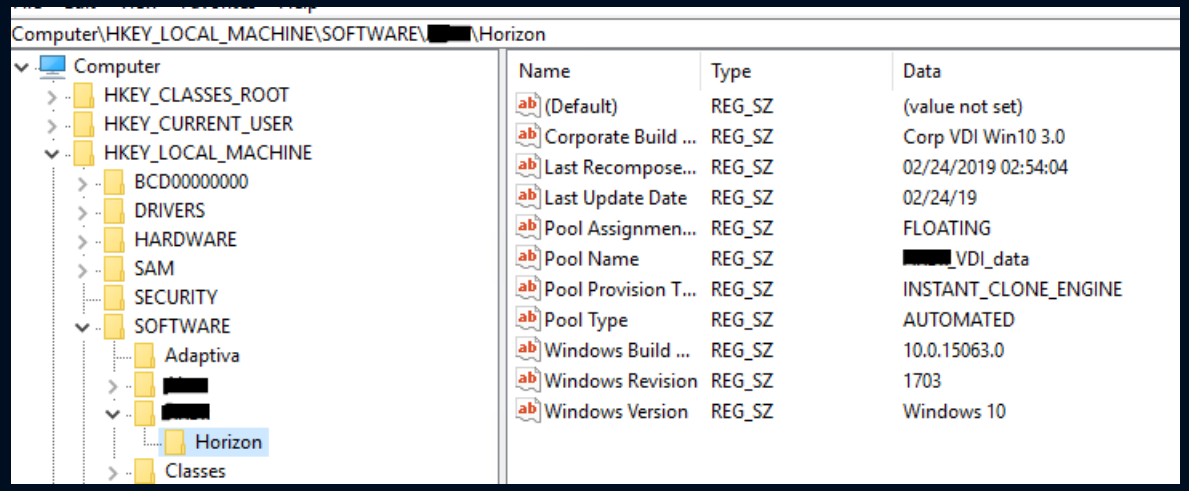
```
#Start the Install Updates Script on Master VM
Write-Host "About to kick off the Install Updates Script on VDI Master $VMLine"
Invoke-Command -ComputerName $VMLine -filepath "$ScriptLocation\$InstallUpdatesScript"
Write-Host "Finished running the Install Updates Script on VDI Master $VMLine"
#Restart the Master VM
Write-Host "Restarting VDI Master VM $VMLine"
Restart-Computer -ComputerName $VMLine -Force -Wait
#ReStart the Debug Logging
$ErrorActionPreference="SilentlyContinue"
Stop-Transcript | out-null
$ErrorActionPreference = "Continue"
Start-Transcript -Force -Path $LogLocation\$ScriptDate\WeeklyVDIUpdates.txt -Append
Write-Host "Start Debug Logs"
#Start the Shutdown Scriopt on Master VM
Write-Host "About to kick off the Shutdown Script on VDI Master $VMLine"
Invoke-Command -ComputerName $VMLine -filepath "$ScriptLocation\$ShutdownScript"
Write-Host "Finished running the Shutdown Script on VDI Master $VMLine"
#Check to VM powerstate and wait for VM to power off before continuing down the script
$VMPower = get-vm $VMLine
Write-Host "Waiting for VDI Master Image $VMLine to Shutdown"
Do {
    Start-Sleep 30
    $VMPower = get-vm $VMLine
    Write-Host $VMPower.PowerState
    } While ($VMpower.PowerState -eq "PoweredOn")
#Create SNAPshots for each of the VMs
get-vm $VMLine | Where-Object {$_.powerstate -eq "poweredoff"} | New-Snapshot -Name $ScriptDate -Description "Automated Install of Updates for $ScriptDate" -RunAsync
Write-Host "Created Snapshot for $VMLine Named $ScriptDate Details Below"
set-vm $VMLine -Notes "Last Recomposed: $ScriptDate" -Confirm:$false
Write-Host "End of Snapshot Config for $VMline"
#Set VM nots for each of the Master VMs
set-vm $VMLine -Notes "Last Recomposed: $ScriptDate" -Confirm:$false
Write-Host "Set VM Nots for Last Recompose Date: $ScriptDate"
get-vm $VMLine | Select-Object Name,Notes
#Remove old SNAPshots from each of the VMs
Write-Host "Removing old Snapshots listed below"
get-vm $VMLine | Where-Object {$_.powerstate -eq "poweredoff"} | Get-Snapshot | Where-Object {$_.Created -lt (Get-Date).AddDays($SnapDays)}
get-vm $VMLine | Where-Object {$_.powerstate -eq "poweredoff"} | Get-Snapshot | Where-Object {$_.Created -lt (Get-Date).AddDays($SnapDays)} | Remove-Snapshot -Confirm:$false
Write-Host "Removed old Snapshots from above"
#PowerOn each of the master VMs
get-vm $VMLine | Where-Object {$_.powerstate -eq "poweredoff"} | Start-VM
Start-Sleep 240
write-host Get-vm $vmline.name " Is in the powerstate of " Get-vm $vmline.PowerState
#Start the SCCM Service on Master VM
Invoke-Command -ComputerName $VMLine -ScriptBlock ${Function:Start-VDIservices} -ArgumentList CCMexec
Write-Host "Start SCCM Service and current status is below"
```

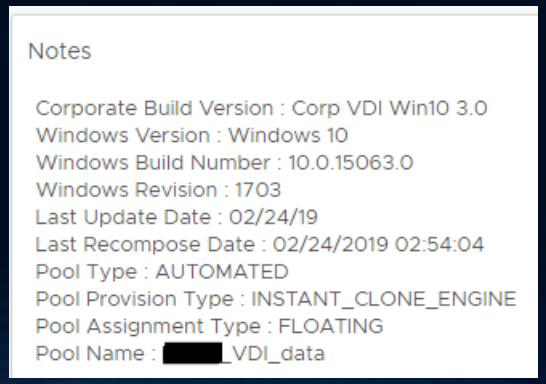
Snippet from Refresh Portion of Script

```
#Connect to Connection servers
foreach ($HVServer in $HVServers)
{
    Connect-HVServer $HVServer -Credential $VMwareSVCCreds
    Write-Host "Connected to $HVServer"
    #Get Pool Names
    $poolsupdate = Get-HVPool
    foreach($poolID in $poolsupdate)
    {
        $HVPoolIDs = $poolID | get-hvpoolspec | ConvertFrom-Json
        $HVPoolIDText = $HVPoolIDs.base.name
        # Set recompose delay based on if pool is Test or Prod
        if($HVPoolIDText -like $TestPoolNC)
            {
                $RecomposeDate = (Get-Date).AddHours($RCTimeDelay1)
                Write-host "Pool Named"$HVPoolIDs.base.name"is a Test pool and will be Recoposed or Push Image at $RecomposeDate"
            }else {$RecomposeDate = (Get-Date).AddHours($RCTimeDelay2)}
            Write-host "Pool Named"$HVPoolIDs.base.name"is a Production pool and will be Recoposed or Push Image at $RecomposeDate"
        #Does Image Push to Instant Clone Pool
        if($HVPoolIDs.AutomatedDesktopSpec.provisioningType -like 'INSTANT_CLONE_ENGINE')
            {
                Start-HVPool -SchedulePushImage -Pool $HVPoolIDs.base.name -LogoffSetting WAIT_FOR_LOGOFF -ParentVM $HVPoolIDs.AutomatedDesktopSpec.virtualCenterProvisioningSettings.VirtualCenterProvisioningData.paren
                Write-Host "Recomposing the Pool"$HvpoolIDs.Base.name"with ParentVM"$HvpoolIDs.AutomatedDesktopSpec.virtualCenterProvisioningSettings.VirtualCenterProvisioningData.parentVm"with pool type of"$HVPoolIDs
            }
        #Does Recompose to Linked Clone Pool
        elseif($HVPoolIDs.AutomatedDesktopSpec.provisioningType -like 'VIEW_COMPOSER')
            {
                Start-HVPool -Recompose -Pool $HVPoolIDs.base.name -LogoffSetting WAIT_FOR_LOGOFF -ParentVM $HVPoolIDs.AutomatedDesktopSpec.virtualCenterProvisioningSettings.VirtualCenterProvisioningData.parentVm -Sna
                Write-Host "Recomposing the Pool"$HvpoolIDs.Base.name"with ParentVM"$HvpoolIDs.AutomatedDesktopSpec.virtualCenterProvisioningSettings.VirtualCenterProvisioningData.parentVm"with pool type of"$HVPoolIDs
            }
       }
    #Disconnect from Connection Server
    Disconnect-HVServer $hvserver -Confirm:$false
    Write-Host "Disconnected from $hvServer"
}
```

Custom Notes Updates



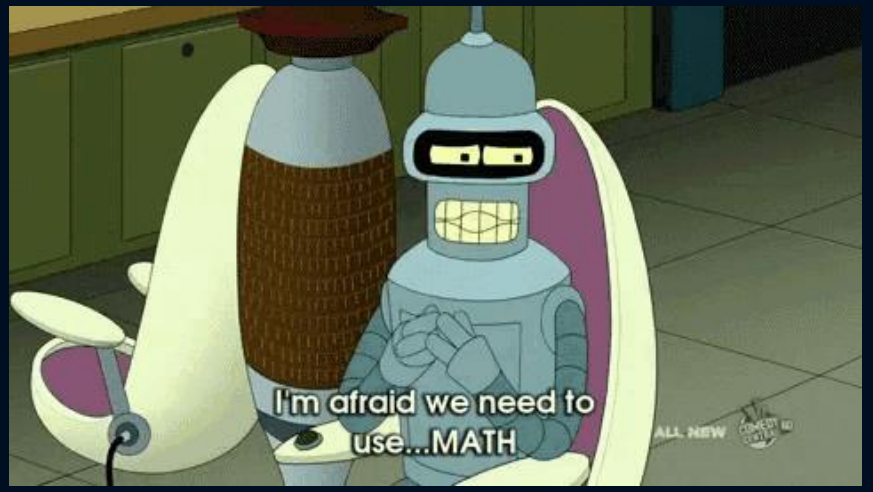




Savings

- Automating the process == No Inconsistency and No Missed Updates
- Log files and automated Service Now tickets provide full audit trail for each Master Image.
- From time savings we have went from 50 pools to now down to 26.
- Moved from 6 cycles a year to 52, That's over a 800% increase of refresh cycles.
- Pool Refresh now a scheduled task!
- Average Runtime: 20 to 30 Min/Pool
- Down from 2 FTE and 2.5 Hours/Pool!
- No More Check Lists!
- New Normal: .5 ticket per refresh cycle, DOWN FROM 24!!
- **Gave me back my Sanity!**

Math Slide x1:



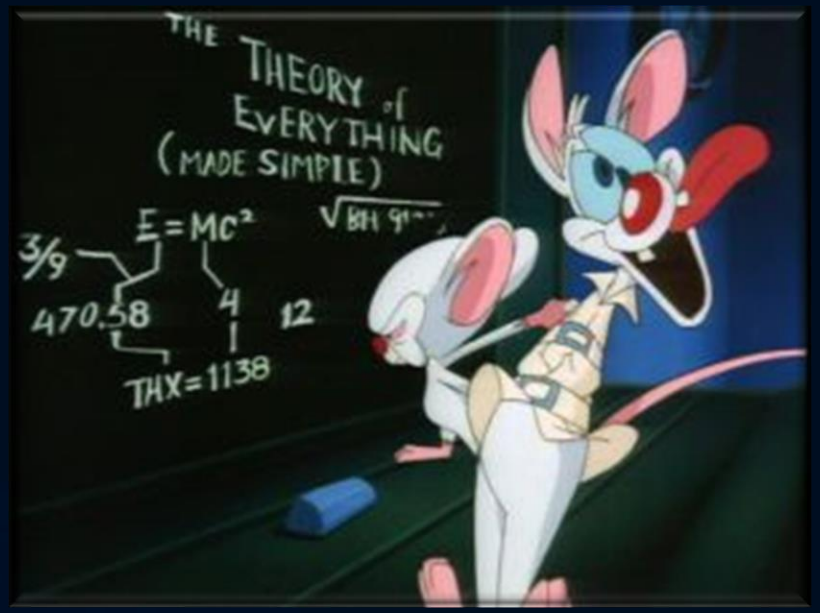
- OLD WAY:
- (50 Pools @ 2.5 hours: 125 Hours +
  24 Tickets @ 2.5 hours MTTR: 60 Unplanned Hours)
  x 6 Updates a year: **1110 Hours/year**
- New Way:
- (50 Pools @ .5 hours: 25 Hours +
- 1 Ticket @ 2.5 hours MTTR: 2.5 Unplanned Hours)
  x 6 Updates a year: **165 Hours/year**
- Net Savings From Automation: **945 Hours**
- ~111 Business Days
- ~ 22 Weeks

Math Slide x 2:

- What if we went from 6 updates a year to doing updates once a week!

- Old Way
  - (50 Pools @ 2.5 hours: 125 Hours +
    24 Tickets @ 2.5 hours MTTR: 60 Unplanned Hours)
    x 52 Updates a year: **9620 Hours/year**

- New Way
  - (50 Pools @ .5 hours: 25 Hours +
  - 1 Ticket @ 2.5 hours MTTR: 2.5 Unplanned Hours)
    x 52 Updates a year: **3250 Hours/year**
  - Net Savings From Automation: **6370 Hours**
  - ~796 Business Days
  - ~ 159 Weeks



What its Saved us:

- Now that we can run a scheduled task average weekly hours spent on pool tasks are:
- 26 Tickets a year @ 2.5 hours MTTR: 65 unplanned hours a year

- True Net Savings From Automation:

9555 Hours a year saved!

That is 5 FTE's hours for a whole Year!

Other Ways to Automate

- vCheck Horizon View
- https://github.com/vCheckReport/vCheck-HorizonView

- AsBuiltReport for Horizon
- https://github.com/AsBuiltReport/AsBuiltReport.VMware.Horizon

- Automating Creation AppStacks
- https://bit.ly/2RW7DHq

- Automation of Image builds
  - packer.io

- Cleanup Failed VDI desktops
  - https://github.com/Magneet/Get_Horizon_view_bad_vdi

- Building Link or Instant Clone pools. It's a one line script!
  - New-HVPool (https://bit.ly/2zXIpl2)

Questions, Comments Concerns?
Visit My Blog!

- Twitter @childebrandt42
- Blog https://childebrandt42.com/
- GitHub https://github.com/childebrandt42/