**Get a listing of all VMs on all hosts.**

 Get-VM

**Get a listing of all VMs on esx35-02.vitoolkit.local.** Get-VMHost esx35-02.vitoolkit.local| Get-VM

**Get a listing of all VMs that reside in the “student-resourcepool-01” resource pool.** Get-ResourcePool student-resourcepool-01 | Get-VM

**Create a new VM on esx35-02.vitoolkit.local. Give it 1 gigabyte of RAM and 10 megabytes of disk space. Call it StudentVm1.** Get-VMHost esx35-02.vitoolkit.local | New-VM -Name "StudentVm1" ` -DiskMB 10 -MemoryMB 1024

**Create 10 new VMs on esx35-02.vitoolkit.local. Give them all 1 gigabyte of RAM and 5 megabytes of disk space. Name the VMs student1 through student10.**Foreach ($i in 1..10) {Get-VMHost esx35-02.vitoolkit.local | New-VM -Name "student$i” ` -DiskMB 5 -MemoryMB 1024}

**Reconfigure each VM to use 256 megabytes of RAM rather than its current memory setting.** Get-VM | Set-VM -MemoryMB 256 –confirm:$false

**Reconfigure each VM to have 2 virtual CPUs rather than 1.** Get-VM | Set-VM –NumCpu 2 –confirm:$false

**Power on all VMs of form student-vm-\***

Get-VM student-vm-\* | Where { $\_.PowerState –eq “PoweredOff”} | Start-VM

**Get a listing of all VMs that have connected CD-ROM drives.**

Get-VM | where {($\_.CDDrives)[0].ConnectionState.Connected}

**Disconnect any connected CD-ROM drives.** Get-VM | Get-CDDrive | Set-CDDrive -Connected $false ` –confirm:$false

**Get a listing of all VMs along with the sizes of their hard drives.** Get-VM | select Name, @{ Name="DatastoreCapacityKB"; Expression = { $\_.Harddisks | foreach { $\_.CapacityKb }} }

**Get a listing of all VMs along with the datastores on which they reside.**Get-Vm | select Name, @{ Name = "Datastore"; Expression = {$\_ | Get-Datastore} }

**Get a listing of all available templates.** Get-Template

**Get a listing of all available customization specifications.** Get-OSCustomizationSpec

**Create a clone of the template, using the available customization specification. Deploy the template to the nas\_system\_datastore2 datastore.** $template = Get-Template $spec = Get-OSCustomizationSpec New-VM -Name "vmFromTemplate" -VMHost esx35-02.vitoolkit.local ` -Template $template -OSCustomizationSpec $spec ` -Datastore nas\_system\_datastore2

**Create a new VM on esx35-02.vitoolkit.local, attaching it to the nas\_student\_datastore1 datastore. Give it 10 megabytes of disk.** Get-VMHost esx35-02.vitoolkit.local| ` New-VM -Name "vmOnDatastore" -Datastore nas\_student\_datastore1 ` -diskmb 10

**Create a new VM on esx35-02.vitoolkit.local, attaching it to the Internal network. Give it 10 megabytes of disk.** Get-VMHost esx35-02.vitoolkit.local| ` New-VM -Name "vmOnNetwork" -NetworkName Internal ` -diskmb 10

**List all network interfaces on the ESX host esx35-01.vitoolkit.local.** Get-VMHost 'esx35-01.vitoolkit.local' | Get-VMHostNetwork | foreach { $\_.PhysicalNic }

**List all network interfaces defined on all ESX hosts.** Get-VMHost | foreach { Write-Host "`nNics at host $\_ :" –ForegroundColor Yellow $\_ | Get-VMHostNetwork | foreach { $\_.PhysicalNic } }

**List available virtual switches on all hosts.** Get-VMHost | Get-VirtualSwitch

**List available portgroups on all hosts.** Get-VMHost | Get-VirtualSwitch | Get-VirtualPortGroup

**Find all network interfaces that are not currently used in a virtual switch on esx35-03.vitoolkit.local.** $vmhost = Get-VMHost esx35-03.vitoolkit.local foreach ($pnic in $hostNetwork.PhysicalNic) { $switchesUsingPnic = Get-VirtualSwitch $vmHost | where { $\_.Nic -ne $null -and $\_.Nic -contains $pnic.DeviceName } if ($switchesUsingPnic -eq $null) { $pnic } }

**On esx35-03.vitoolkit.local, create a new virtual switch called “New Switch”. Assign 64 ports to the switch and assign two interfaces to the switch.** $pnic = (Get-VMHostNetwork -VMHost ` 'esx35-03.vitoolkit.local').PhysicalNic[1] New-VirtualSwitch -VMHost 'esx35-03.vitoolkit.local' ` -Nic $pnic.DeviceName -NumPorts 64 -Name 'New Switch'

**Create a new portgroup named “New Group”. Create it on New Switch and give it a VLAN ID of 100.** $switch = ` Get-VirtualSwitch -VMHost esx35-03.vitoolkit.local -Name 'New Switch' New-VirtualPortGroup ` -Name 'New Group' -VirtualSwitch $switch -VLanId 100

**Create a new VM with a single network adapter attached to the New Group portgroup.** New-VM -Name MyVM -VMHost esx35-03.vitoolkit.local ` -NetworkName 'New Group' -DiskMB 8

**Move the VM you created to the External network.** (Get-VM MyVM).NetworkAdapters[0] | ` Set-NetworkAdapter -NetworkName 'External' –confirm:$false

**Move student-vm-01 virtual machine to esx35-03.vitoolkit.local esx host.**$vm01 = Get-VMHost esx35-02.vitoolkit.local | Get-Vm "student-vm-01" $vm01 | Get-NetworkAdapter | Set-NetworkAdapter ` -Connectedfalse -Confirm:$false $vm01 | Move-Vm -Destination esx35-03.vitoolkit.local

**Move student-vm-01 virtual machine back to esx35-02.vitoolkit.local esx host.**

$vm01 = Get-VMHost esx35-03.vitoolkit.local | Get-Vm "student-vm-01" $vm01 | Move-Vm -Destination esx35-02.vitoolkit.local

**Get a listing of all VMs and the datastores where they reside.**

Get-VM | Select Name, { $\_ | Get-Datastore }

**Move student-vm-01, student-vm-02 and student-vm-03 from nas\_student\_datastore1 to nas\_system\_datastore1**

Get-VM student-vm-01,student-vm-02,student-vm-03 | Move-Vm -datastore nas\_system\_datastore1

**Move all powered off VMs from esx35-03.vitoolkit.local to esx35-02.vitoolkit.local.**

Get-vmhost esx35-03.vitoolkit.local | get-vm | where-object {$\_.Powerstate -eq "PoweredOff"} | move-vm -destination esx35-02.vitoolkit.local

**Enable VMotion esx35-01.vitoolkit.local. Create VMKernels on the vswitch0 switch. The VMKernel IP for esx35-01 should be 192.168.0.101. The netmask should be 255.255.255.0.**

$vs = Get-VMHost esx35-01.vitoolkit.local | Get-VirtualSwitch –Name vswitch0 Get-VMHost esx35-01.vitoolkit.local | New-VMHostNetworkAdapter -PortGroup VMKernel -VirtualSwitch $vs ` -IP 192.168.0.101 -SubnetMask 255.255.255.0 –VmotionEnabled:$true

**Enable VMotion at the host level for esx35-01.vitoolkit.local.**

get-vmhost esx35-01.vitoolkit.local | set-vmhostadvancedconfiguration ` -Name Migrate.Enabled -Value 1

**Ensure that VMotion is enabled.**

# When you run these commands you should see one line of output each.

Get-VMHost esx35-01.vitoolkit.local | Get-VirtualPortGroup | Where { $\_.Port } | Where { $\_.Port[0].Type -eq "host" }

**List all datastores along with their capacity and free space.**

Get-vmhost | get-datastore

**An NFS datastore has been created at 192.168.0.4. Add this datastore to esx35-02.vitoolkit.local and esx35-03.vitoolkit.local Call the datastore DS\_NFS.**

new-datastore -nfs -vmhost esx35-03.vitoolkit.local ` -name NFS -path /mnt/shared/nfs1/student\_nfs -nfshost 192.168.0.4

**An iSCSI datastore has been created at 192.168.0.4 port 3260. Add this datastore to esx35- 02.vitoolkit.local and esx35-03.vitoolkit.local. Call it DS\_iSCSI.** #

Get lunPath and create new storage $lunpath = Get-ScsiLun -VmHost (Get-VMHost esx35-02.vitoolkit.local) | where {$\_.CanonicalName.StartsWith('vmhba32:0:7')} | Get-ScsiLunPath New-Datastore -Vmfs -VMHost $h -Path $lunpath.LunPath -Name DS\_iSCSI

**Rescan storage on all hosts.**Get-VMHost | Get-VMHostStorage -RescanAllHba

**List all SCSI LUNs defined on all hosts as well as their multipath policies.**

Get-VMHost | Get-ScsiLun

**List all SCSI LUN paths.**

Get-VMHost | Get-ScsiLun | Get-ScsiLunPath

**Display hard disk sizes for all available VMs.**

Get-VM | Get-HardDisk

**# Or!**

Get-VM | Select Name, @{Name = "CapacityKB"; Expression = {$\_ | Get-HardDisk | foreach {$\_.CapacityKB}}}

**Add a second hard disk to student-vm-03. Allocate 10Mb to the hard disk.**

Get-VM student-vm-03 | New-HardDisk -CapacityKB 10000

**Create a new cluster named “New Cluster” with HA and DRS disabled.**

New-Cluster "New Cluster" -Location StudentLab

**Move all ESX hosts into New Cluster.**

Move-VMHost ESX\* -Destination "New Cluster"

**Enable HA on New Cluster. When enabling HA, set admission control on the cluster.**

Set-Cluster "New Cluster" –HAAdmissionControlEnabled $true ` –confirm:$false

**Enable DRS on New Cluster. When enabling DRS, set the automation level to partially automated.**

Set-Cluster "New Cluster" -DrsAutomationLevel PartiallyAutomated ` -confirm:$false

**Change the DRS automation level of all VMs currently on esx35-01 to fully automated.**

Get-VM -Location "ESX35-01\*" | Set-VM -DrsAutomationLevel FullyAutomated

**Change the HA restart priority of all VMs currently on esx35-02 to High.**

Get-VM -Location "ESX35-02\*" | Set-VM -HARestartPriority High

**Try to move all hosts out of New Cluster. (will not work with Running VM's!)**

Move-VMHost ESX\* -Destination "StudentLab"

**Suspend all running VMs on all hosts.**

Get-VM | Where { $\_.PowerState –eq “PoweredOn” } | Suspend-VM –confirm:$false

**Put all hosts into maintenance mode.**

Get-VMHost ESX\* | Set-VMHost -State Maintenance

**Move all hosts out of New Cluster.**

Move-VMHost ESX\* -Destination "StudentLab"

**Take all hosts out of maintenance mode.**

Get-VMHost ESX\* | Set-VMHost -State Connected

**Resume all suspended VMs.**

Get-VM | Where { $\_.PowerState –eq “Suspended” } | Start-VM

**Create new snapshots of all VMs.**

Get-VM student-vm\*| New-Snapshot -Name Snapshot1

**Find VM snapshots “Snapshot1” and display VM Name and Snapshot Name**

Get-VM | Get-Snapshot -Name Snapshot1 | Select VM, Name

**Start VMs, as memory state Snapshot only available for Powered On VMs**

Get-VM student-vm\* | Start-VM **#**

**To save memory state when creating snapshots use the –Memory option. Revert all VMs to the state they were in before taking the snapshot.**

Get-VM student-vm\* | foreach {$\_ | Set-VM -Snapshot (Get-Snapshot -VM $\_ -Name Snapshot2) -Confirm:$false }

**Get a listing of all snapshots, including the time when they were created.**

Get-VM | Get-Snapshot | Select VM, Name, Created

**Remove all snapshots older than 1 week.**

Get-VM | Get-Snapshot | where {$\_.Created -le (Get-Date).AddDays(-7)} | Remove-Snapshot –confirm:$false

**Get all processes running in win2k3-student{n}. \*Where n is your student number**

$hostCred = Get-Credential

$guestCred = Get-Credential $vmName = “win2k3-student{n}”

Get-VM $vmName | Invoke-VMScript –HostCredential $hostCred ` -GuestCredential $guestCred Get-Process

# Note: The remaining solutions assume these variables are defined!

**List all services running inside win2k3-student{n}.**

Get-VM $vmName | Invoke-VMScript –HostCredential $hostCred ` -GuestCredential $guestCred Get-Service

**Restart the DHCP service within win2k3-student{n}.**

Get-VM $vmName | Invoke-VMScript –HostCredential $hostCred ` -GuestCredential $guestCred “Get-Service Dhcp | Restart-Service ` -force”

**List the names of all available eventlogs available within win2k3-student{n}.**

Get-VM $vmName | Invoke-VMScript –HostCredential $hostCred ` -GuestCredential $guestCred “Get-EventLog –list”

**Retrieve all security events from win2k3-student{n}.**

Get-VM $vmName | Invoke-VMScript –HostCredential $hostCred ` -GuestCredential $guestCred “Get-EventLog Security”

**Retrieve all application events from win2k3-student{n} of type Information.**

Get-VM $vmName | Invoke-VMScript –HostCredential $hostCred ` -GuestCredential $guestCred ` “Get-EventLog Application | Where { `$\_.EntryType –eq ` ‘Information’ }”

**Determine the version of VMware Tools installed on a VM.**

$vm = Get-VM Win2k3-01.vitoolkit.local $vmView = $vm | Get-View $vmView.Config.Tools.ToolsVersion

**Change the amount of memory assigned to a VM.**

$vm = Get-VM Win2k3-01.vitoolkit.local $vmView = $vm | Get-View $spec = New-Object VMware.Vim.VirtualMachineConfigSpec $spec.MemoryMB = 2048 $vmView.ReconfigVM($spec)

**Set per-VM CPU and memory reservations.**

$vm = Get-VM Win2k3-01.vitoolkit.local $vmView = $vm | Get-View $cpuAllocationInfo = New-Object VMware.Vim.ResourceAllocationInfo $cpuAllocationInfo.shares = “High” $cpuAllocationInfo.reservation = 2000 $memAllocationInfo = New-Object VMware.Vim.ResourceAllocationInfo $memAllocationInfo.shares = “High” $memAllocationInfo.reservation = 2kb $configSpec = New-Object VMware.Vim.VirtualMachineConfigSpec $configSpec.cpuAllocation = $cpuAllocationInfo $configSpec.memoryAllocation = $memAllocationInfo $vmView.ReconfigVM($configSpec)

**Create a virtual switch with custom teaming properties.**

$networkView = Get-VMHost | Get-VMHostNetwork | Get-View $switch = $networkView.NetworkInfo.Vswitch | Where { $\_.Name –eq “New Switch” } $spec = $switch.spec $spec.policy.nicTeaming.policy = "loadbalance\_ip" $view.UpdateVirtualSwitch($switch.Name, $spec)

**Manage licenses.**

# The ScheduledTaskManager is in the Service Instance.

$si = get-view ServiceInstance $scheduledTaskManager = Get-View $si.Content.ScheduledTaskManager

# We need to identify the VM and the host where it will be powered on.

$vmView = Get-VM PowerOnTest | Get-View $esxView = Get-VMHost esx35-01.vitoolkit.local | Get-View

# Now we construct the task argument.

$arg = New-Object VMware.Vim.MethodActionArgument $arg.Value = $esxview.MoRef $action = New-Object VMware.Vim.MethodAction $action.Argument = $arg $action.Name = "PowerOnVM\_Task" $scheduler = new-object VMware.Vim.OnceTaskScheduler $scheduler.runat = (get-date).addminutes(5) $task = New-Object VMware.Vim.ScheduledTaskSpec $task.Action = $action $task.Description = "Start a VM with a scheduled task." $task.Enabled = $true $task.Name = "Power On Virtual Machine" $task.Scheduler = $scheduler $scheduledTaskManager.CreateScheduledTask($vmView.MoRef, $task)

**Create functions to make automation easy.**

# Part 1.

# Define our reusable function.

function Set-VMCpuReservation { param ($vm, $shares, $reservation) $vmView = $vm | Get-View $allocationInfo = New-Object VMware.Vim.ResourceAllocationInfo $allocationInfo.shares = “High” $allocationInfo.reservation = 2000 $configSpec = New-Object VMware.Vim.VirtualMachineConfigSpec $configSpec.cpuAllocation = $allocationInfo $vmView.ReconfigVM($configSpec) }