Migrate your datacenter without downtime

A customer case

Introduction

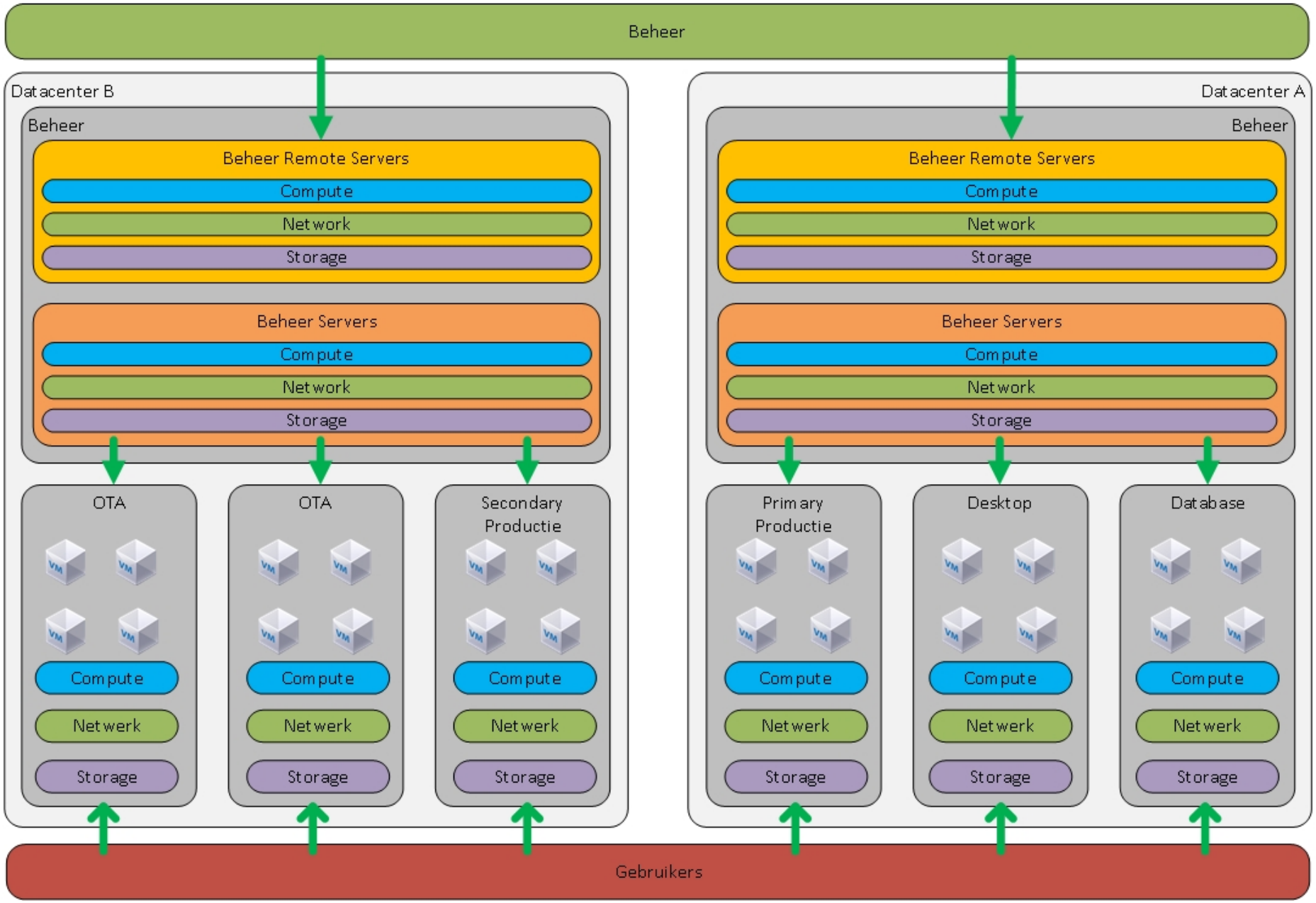
- Michael Wilmsen
- Self employed Architect/Trainer
  - Hyper Converged
- VCDX #210
- VMware vExpert 2011 – 2018
  - vSAN 2016-2017
- Nutanix Technology Champion 2015-2018
- Virtual-Hike.com
- @WilmsenIT

Assignment

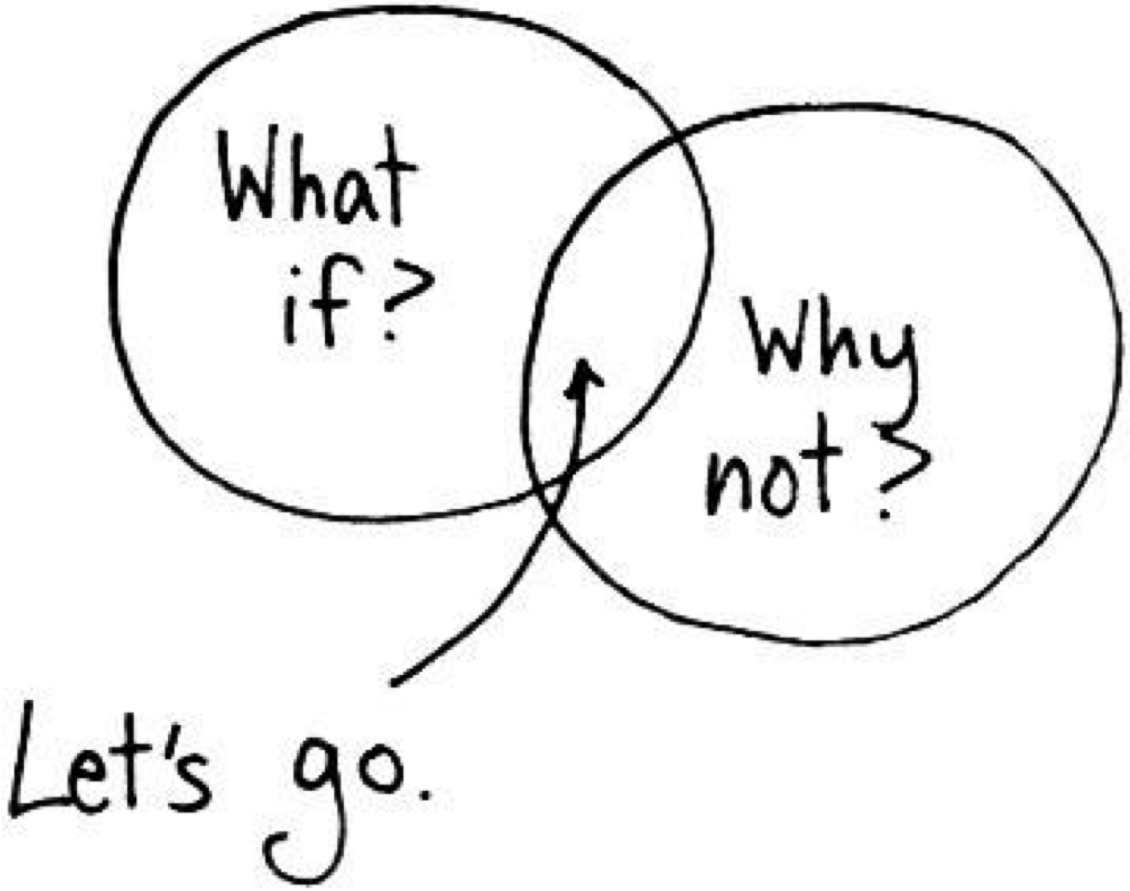
Migrate a datacenter with 500 virtual machines to a new datacenter

Customer: University

- 1 Campus
- 2 Buildings
- Each building has 1 datacenter
- 1 Building will be renovated in 2018
- New datacenter 50 km distance
- 500 virtual machines


Beheer
Datacenter B
Beheer
Beheer Remote Servers
Compute
Network
Storage
Beheer Servers
Compute
Network
Storage
OTA
OTA
Secondary Productie
Compute
Netwerk
Storage
Datacenter A
Beheer
Beheer Remote Servers
Compute
Network
Storage
Beheer Servers
Compute
Network
Storage
Primary Productie
Desktop
Database
Compute
Netwerk
Storage
Gebruikers

What if……?



vMotion requirement

- A RTT (round-trip time) latency of 150 milliseconds or less, between hosts.
- Long distances vMotion requires Enterprise Plus license.
- You must place the traffic related to virtual machine files transfer to the destination host on the provisioning TCP/IP stack.

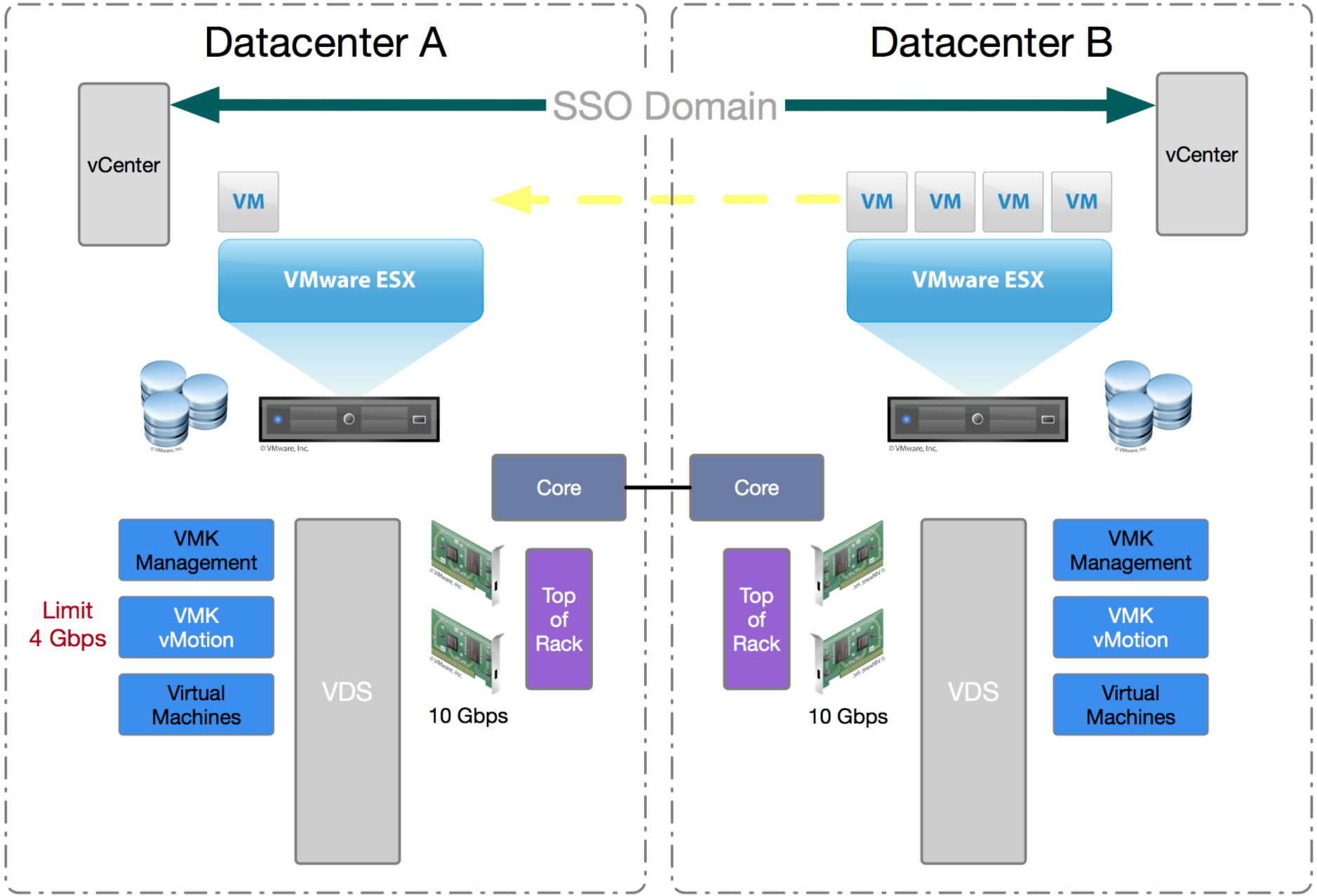
Cross vCenter migration

- The source and destination vCenter Server instances and ESXi hosts must be running version 6.0 or later.
- Enterprise Plus license
- vSphere Web Client
  - vCenter Server instances must be in Enhanced Linked Mode
  - Same vCenter Single Sign-On domain
- Time-synchronized
- vSphere APIs/SDK
  - vCenter Server instances may exist in separate vSphere Single Sign-On domains. Additional parameters are required.

Source: kb2106952

Challenges

- Different vCenter servers
- Different Portgroups names
- vMotion bandwidth limit 4Gbps
- Migration only outside business hours (7u-19u)


Datacenter A
Datacenter B
SSO Domain
vCenter
vCenter
VM
VM
VM
VM
VM
VMware ESX
VMware ESX
Core
Core
VMK Management
VMK vMotion
Virtual Machines
Limit 4 Gbps
VDS
Top of Rack
10 Gbps
Top of Rack
10 Gbps
VDS
VMK Management
VMK vMotion
Virtual Machines

Move-VM requirements

- Active Connection to both Source and Destination vCenters
- Source vCenter
  - VM
  - VM’s Network Adapter/s
- Destination vCenter
  - Destination ESXi Host
  - Destination Datastore
  - Destination PortGroup/s

Script overview

1. ConnectVC (Source + Destination)
2. Get Virtual Machine variables
3. Test destination FreeSpace
4. Move Virtual Machine
5. Check registration

```
Param (
  [Parameter(Mandatory=$true)] [string]$VMList = $null,
  [Parameter(Mandatory=$true)] [string]$SourceCluster = $null,
  [Parameter(Mandatory=$true)] [string]$DestinationCluster = $null,
  [Parameter(Mandatory=$true)] [string]$SourceVC = $null,
  [Parameter(Mandatory=$true)] [string]$DestinationVC = $null,
  [Parameter(Mandatory=$true)] [string]$Username = $null,
  [string]$Password,
  [bool]$Dryrun = $false,
  [bool]$SendWhatsApp = $false
)
```

```
#Define log function
Function LogWrite
{
    Param ([string]$logstring)

    #Add logtime to entry
    $LogTime = Get-Date -Format "MM-dd-yyyy_hh-mm-ss"
    $logstring = $LogTime + " : " + $logstring

    #Write logstring
    Add-content $LogFile -value $logstring
    Write-Host $logstring
}
```

```
Function SendWhatsApp
{
  Param ([string] $message)

  if ( $SendWhatsApp ) {
    $LogTime = Get-Date -Format "MM-dd-yyyy_hh-mm-ss"
    $message = $logtime + " : " + $message

    $instanceId = "2"
    $clientId = "[email]"
    $clientSecret = "[secret]"

    foreach ( $number in $WhatsAppNumbers )
    {
       $jsonObj = @{'group_admin'=$number;
                    'group_name'=$WhatsAppGroup;
                    'message'=$message;}

      Try {
        $res = Invoke-WebRequest -Uri "http://api.whatsmate.net/v2/whatsapp/group/message/$instanceId" `
                                 -Method Post  `
                                 -Headers @{"X-WM-CLIENT-ID"=$clientId; "X-WM-CLIENT-SECRET"=$clientSecret;} `
                                 -Body (ConvertTo-Json $jsonObj)

        LogWrite "WhatsMate Status Code: "  $res.StatusCode
        LogWrite $res.Content

      }
    }
  }
}
```

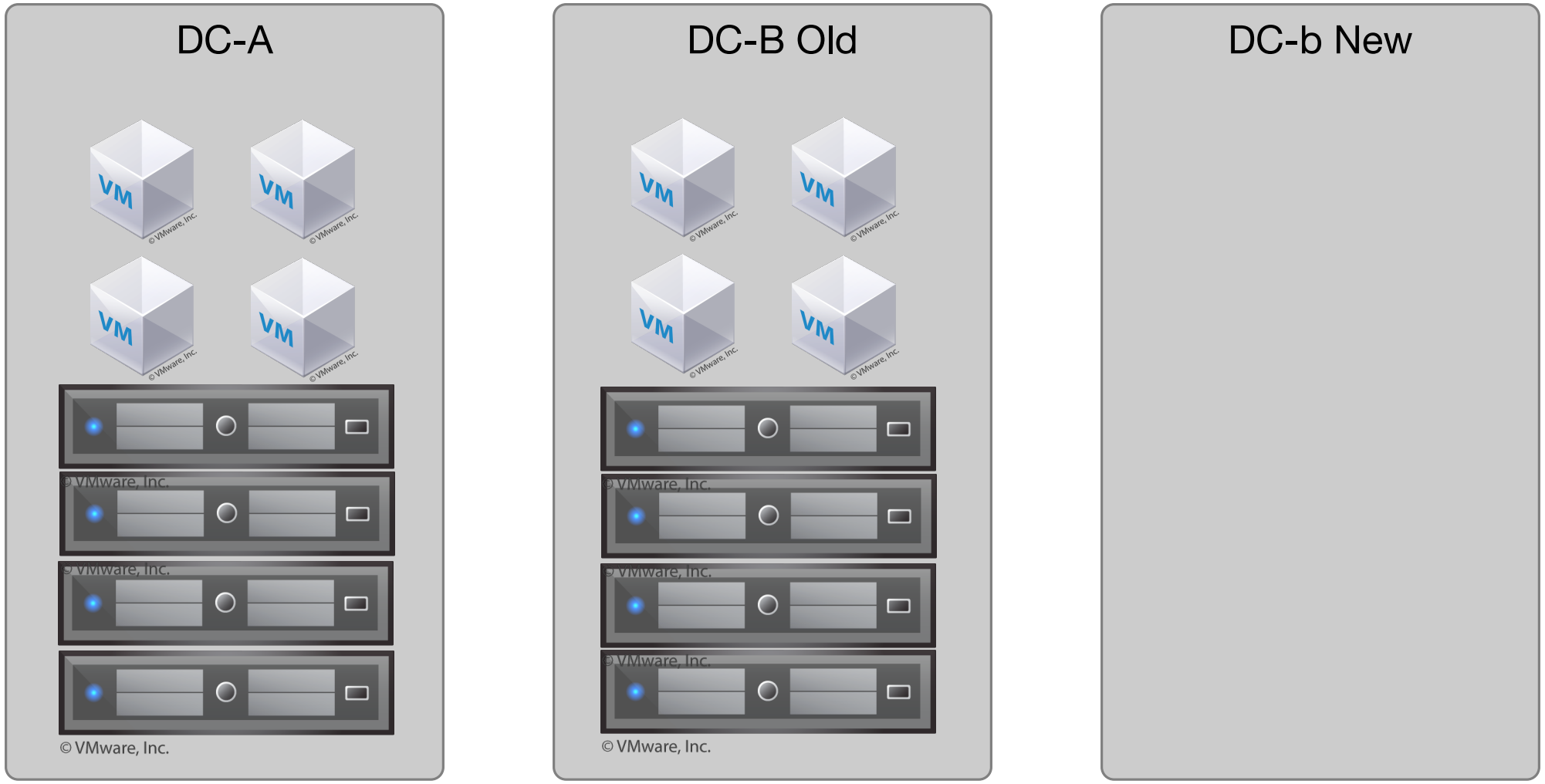
```
#Get VM variables
$vm = get-vm $_
$vmip = $vm  | Select @{N="IP Address";E={@($_.guest.IPAddress[0])}}
$vmip = $vmip."ip address"
$VMHDDSize = Get-VmSize($vm)
$VMHDDSize = [Math]::Round(($VMHDDSize / 1GB),2)
$vmfolder = $vm.folder
$vmfolder = $vmfolder.name
$NetworkAdapter = Get-NetworkAdapter -VM $vm -Server $SourceVC
$SourceVMPortGroup = Get-NetworkAdapter -vm $vm | Get-VDPortgroup
$DestinationVMPortgroup = $SourceVMPortGroup.name -replace $SourceCluster,$DestinationCluster
$switchname = $DestinationCluster
$destinationDatastore = Get-DatastoreCluster -Name $DestinationCluster -Server $DestinationVC |
  Get-Datastore | Sort-Object -Property FreeSpaceGB -Descending | Select-Object -First 1
$destinationDatastoreFreeSpace = $destinationDatastore | Select Name,@{N="FreeSpace";E={$_.ExtensionData.Summary.FreeSpace}}
$destinationDatastoreFreeSpace = [Math]::Round(($destinationDatastoreFreeSpace."FreeSpace" / 1GB),2)
$DestinationPortgroup = Get-VDPortgroup -VDSwitch $switchname -Name $DestinationVMPortgroup -Server $destVCConn
$DestinationHost = Get-Cluster -Name $DestinationCluster -Server $DestinationVC | Get-VMHost -State Connected | Select-Object -First 1
```

```
if ( -NOT $Dryrun) {
  Try {
    $Result = Move-VM -VM $vm `
                      -Destination $DestinationHost `
                      -Datastore $DestinationDatastore `
                      -NetworkAdapter $NetworkAdapter `
                      -PortGroup $DestinationPortgroup `
                      -ErrorAction Stop
    }
  Catch {
    $ErrorMessage = $_.Exception.Message
    LogWrite "ERROR: Migration of $vm failed!!!"
    Logwrite "ERROR: Migration Status Code:  $ErrorMessage"
    SendWhatsApp "ERROR: Migration of $vm failed!!!"
    $MigError = $true
  }
}
```

```
$MigError = $false
#Test if VM is running on destination cluster
if ( -NOT $MigError -AND -NOT $Dryrun ) {
  LogWrite "Check $vm is registered in $DestinationVC"
  try {
    $CheckVM = get-vm -name $vm -server $DestinationVC -ErrorAction Stop

    if ( $CheckVM ) {
      Logwrite "$vm registered in $DestinationVC"
    }
    else {
      Logwrite "ERROR: $vm not found in $DestinationVC"
    }
  }
  catch {
    $ErrorMessage = $_.Exception.Message
    Logwrite "ERROR: $vm not found in $DestinationVC"
    Logwrite "ERROR: $ErrorMessage"
    SendWhatsApp "ERROR move: $vm not found in $DestinationVC"
  }
}
```

Move in batches



Question?

Download

virtual-hike.com

VMUG® USERCON