```
## DataCenter_Duplication_between_2_vCenters.ps1
## build 1.26
## Eric Krejci
## ekrejci.wordpress.com
## Twitter - @ekrejci
## on error stop the script
$erroractionpreference = "Stop"
$Username = "DOMAIN\USERNAME"
$Password = read-host "Enter Password" -assecurestring

### declare your source vcenter

$sourceserver = "source.vcenter.fqdn"

### declare your destination vcenter

$destserver = "destination.vcenter.fqdn"

### connect to the Source vCenter
Connect-VIServer -Server $sourceserver -user $Username -Password
([Runtime.InteropServices.Marshal]::PtrToStringAuto([Runtime.InteropService
s.Marshal]::SecureStringToBSTR($password)))


### connect to the Destination vCenter
Connect-VIServer -Server $destserver -user $Username -Password
([Runtime.InteropServices.Marshal]::PtrToStringAuto([Runtime.InteropService
s.Marshal]::SecureStringToBSTR($password)))


### retreive the authorizationManager from both vCenters

[Reflection.Assembly]::LoadWithPartialName("vmware.vim") | Out-Null

$svcRef = new-object VMware.Vim.ManagedObjectReference
$svcRef.Type = "ServiceInstance"
$svcRef.Value = "ServiceInstance"

$serviceInstanceSrc = get-view $svcRef -Server $sourceserver
$serviceInstanceDst = get-view $svcRef -Server $destserver

$authMgrSrc = get-view $serviceInstanceSrc.Content.authorizationManager -
Server $sourceserver
$authMgrDst = get-view $serviceInstanceDst.Content.authorizationManager -
Server $destserver

### create array with roles from the Source vCenter

$listofrolessrc =@()


foreach($role in $authMgrSrc.RoleList){
  $row = "" | Select RoleName, Label, RoleId
  $row.RoleName = $role.Name
  $row.Label = $role.Info.Label
  $row.RoleId = $role.RoleId

  $listofrolessrc += $row
  }
```

```
### create array with roles from the Destination vCenter

$listofrolesdst =@()


foreach($role in $authMgrDst.RoleList){
  $row = "" | Select RoleName, Label, RoleId
  $row.RoleName = $role.Name
  $row.Label = $role.Info.Label
  $row.RoleId = $role.RoleId

  $listofrolesdst += $row
  }

### Functions Start

### Copy permission Function

function Copy-Permission {
  Param( $SourceID,  ## This is the ID of the Source Object
         $DestID)        ## This is ID of the Destination Object

    ## Now let's copy the permissions from the current object to the new
one

    $entityFrom = get-view $SourceID -Server $sourceserver
    $entityTo = get-view $DestID -Server $destserver
    $inherited = $FALSE

    $permissions = $authMgrSrc.RetrieveEntityPermissions($entityFrom.MoRef,
$inherited)
    foreach($perm in $permissions){

    foreach ($res in $listofrolessrc) {

        if ($res.RoleID -eq $perm.RoleID) {

        $rolelabel = $res.Label
        $rolename = $res.RoleName

        }

        }

        foreach ($res in $listofrolesdst) {

        if (($res.Label -like $rolelabel)) {

        $newroleid = $res.RoleID

        }

        }

      $newperm = new-object VMware.Vim.Permission
      $newperm.Group = $perm.Group
      $newperm.Principal =  $perm.Principal
      $newperm.Propagate = $perm.Propagate
      $newperm.RoleId = $newroleid
```

```
          $authMgrDst.SetEntityPermissions($entityTo.MoRef,$newperm)
    }

}


### Copy Resource Pool Function

function Copy-ResourcePool {
  Param( $FromResourcePool,  ## This is the name of the top-level pool to
copy
         $ToLocation)        ## This is the destination location

    ## let's check if the Resource Pool already exists.
      $NewResourcePool=Get-ResourcePool -Location $ToLocation -Server
$destserver -Name $FromResourcePool.Name -NoRecursion -ErrorAction
SilentlyContinue

  if ($NewResourcePool -eq $Null) {
    "Copying " + $FromResourcePool.Name + " to " + $ToLocation.Name
    $NewResourcePool = New-ResourcePool -Location $ToLocation -Name
$FromResourcePool.Name -Server $destserver
    }
  else {
    $NewResourcePool=@($NewResourcePool)[0]  ## We want to make sure we get
only one pool
    "Updating existing pool " + $FromResourcePool.Name + " in " +
$ToLocation.Name
    }

  ## If the CPU Shares is Custom, we set each setting, otherwise we just
set the shares
  if ($FromResourcePool.CpuSharesLevel -eq "Custom" ) {
    Set-ResourcePool -ResourcePool $NewResourcePool `
      -CpuSharesLevel $FromResourcePool.CpuSharesLevel `
      -CpuExpandableReservation $FromResourcePool.CpuExpandableReservation
`
      -CpuLimitMhz $FromResourcePool.CpuLimitMhz `
      -CpuReservationMhz $FromResourcePool.CpuReservationMhz `
      -NumCpuShares $FromResourcePool.NumCpuShares | Out-Null
    }
  else {
    Set-ResourcePool -ResourcePool $NewResourcePool `
      -CpuSharesLevel $FromResourcePool.CpuSharesLevel | Out-Null
    }

  ## If the Mem Shares is Custom, we set each setting, otherwise we just
set the shares
  if ($FromResourcePool.MemSharesLevel -eq "Custom" ) {
    Set-ResourcePool -ResourcePool $NewResourcePool `
      -MemExpandableReservation $FromResourcePool.MemExpandableReservation
`
      -MemLimitMB $FromResourcePool.MemLimitMB `
      -MemReservationMB $FromResourcePool.MemReservationMB `
      -NumMemShares $FromResourcePool.NumMemShares | Out-Null
    }
  else {
    Set-ResourcePool -ResourcePool $NewResourcePool `
      -MemSharesLevel $FromResourcePool.MemSharesLevel | Out-Null
    }
    ## Now let's copy the permissions au the current resource pool to the
new one

    Copy-Permission -SourceID $FromResourcePool.ID -DestID
$NewResourcePool.ID


  ## Now we copy the resource pools under me.
  Get-ResourcePool -Location $FromResourcePool -Server $sourceserver -
NoRecursion | Foreach-Object {
    Copy-ResourcePool -FromResourcePool $_ -ToLocation $NewResourcePool
    }
  }

### Copy VMs folder Function

function Copy-Folder {
  Param( $FromFolder,  ## This is the name of the top-level Folder to copy
         $ToLocation)        ## This is the destination location


    ## let's check if the folder already exists.
      $NewFolder=Get-folder -Location $ToLocation -Server $destserver -Name
$FromFolder.Name -NoRecursion -ErrorAction SilentlyContinue

  if ($NewFolder -eq $Null) {
    "Cloning " + $FromFolder.Name + " to " + $ToLocation.Name
    $NewFolder = New-folder -Location $ToLocation -Name $FromFolder.Name
    }
  else {
    $NewFolder=@($NewFolder)[0]  ## We want to make sure we get only one
folder
    "Updating existing Folder " + $FromFolder.Name + " in " +
$ToLocation.Name
    }

    ## Now let's copy the permissions from the current folder to the new
one

    if ($newfolder.Parent.ParentFolder.type -ne "Datacenter" ) {

    Copy-Permission -SourceID $FromFolder.ID -DestID $NewFolder.ID

    }

  ## Now we copy the folders under me.
  Get-folder -Location $FromFolder -Server $sourceserver -NoRecursion |
Foreach-Object {
    Copy-Folder -FromFolder $_ -ToLocation $NewFolder
    }
  }

### enf of the Functions



### enter the name of the Source DataCenter

$SourceDCName= read-host "Source DataCenter Name"

$SourceDC = Get-Datacenter -Name $SourceDCName -Server $sourceserver
```

```
### Create the Destination DataCenter. if it already exists ...

$DestDC = New-Datacenter -Name $SourceDC.Name -Location (Get-Folder -Name
"Datacenters" -server $destserver)

### set the permission for the new DC

Copy-Permission -SourceID $SourceDC.Id -DestID $DestDC.Id

### list the clusters in the source vCenter and duplicate them in the
destination one.

$SourceClusters = Get-Cluster -Location $SourceDC -Server $sourceserver

### now for each cluster clone it to the destination DataCenter

foreach ($SourceCluster in $SourceClusters) {

### create the cluster

$DestCluster = New-Cluster -Name $SourceCluster.Name -Location $DestDC -
Server $destserver -VMSwapfilePolicy $SourceCluster.VMSwapfilePolicy

### set HA if enabled

if ($SourceCluster.HAEnabled -like "true") {

    ## had to check because the value must be between 0 and 4 for the
cmdlet set-cluster
    if (($SourceCluster.HAFailoverLevel -le 1) -or
($SourceCluster.HAFailoverLevel -ge 4)) {
    Set-Cluster -Cluster $DestCluster -HAEnabled:$true -HAFailoverLevel 1 -
HAAdmissionControlEnabled $SourceCluster.HAAdmissionControlEnabled -
HAIsolationResponse $SourceCluster.HAIsolationResponse -HARestartPriority
$SourceCluster.HARestartPriority -Server $destserver -Confirm:$false | Out-
Null
    }else {
    Set-Cluster -Cluster $DestCluster -HAEnabled:$true -HAFailoverLevel
$SourceCluster.HAFailoverLevel -HAAdmissionControlEnabled
$SourceCluster.HAAdmissionControlEnabled -HAIsolationResponse
$SourceCluster.HAIsolationResponse -HARestartPriority
$SourceCluster.HARestartPriority -Server $destserver -Confirm:$false | Out-
Null
    }
} else {

"No HA for the cluster " + $DestCluster.name
}

### set DRS if enabled

if ($SourceCluster.DrsEnabled -like "true") {

Set-Cluster -Cluster $DestCluster -DRSEnabled:$true -DrsAutomationLevel
$SourceCluster.DrsAutomationLevel -Server $destserver -Confirm:$false |
Out-Null

### if DRS is enabled, then we start to replicate the Resource Pools of the
SourceCluster
Get-ResourcePool -Location $SourceCluster -NoRecursion | Get-Resourcepool -
NoRecursion | Foreach-Object {
  Copy-ResourcePool -FromResourcepool $_ -ToLocation (Get-ResourcePool -
Location $DestCluster -NoRecursion)
  }


} else {

"No DRS for the cluster " + $DestCluster.name
}



### set the permission for the new Cluster

Copy-Permission -SourceID $SourceCluster.Id -DestID $DestCluster.Id


}

## Let's copy the VM folder from the source DataCenter


Get-folder -Location $SourceDC -NoRecursion -Type "VM" | Foreach-Object {
  Copy-Folder -FromFolder $_ -ToLocation ($DestDC)
  }


## Let's copy the HostAndCluster folder from the source DataCenter


Get-folder -Location $SourceDC -NoRecursion -Type "HostAndCluster" |
Foreach-Object {
  Copy-Folder -FromFolder $_ -ToLocation ($DestDC)
  }

## Let's copy the Datastore folder from the source DataCenter


Get-folder -Location $SourceDC -NoRecursion -Type "Datastore" | Foreach-
Object {
  Copy-Folder -FromFolder $_ -ToLocation ($DestDC)
  }

## Let's copy the Network folder from the source DataCenter


Get-folder -Location $SourceDC -NoRecursion -Type "Network" | Foreach-
Object {
  Copy-Folder -FromFolder $_ -ToLocation ($DestDC)
  }


"Finish"

Disconnect-VIServer -Server $sourceserver  -Confirm:$false

Disconnect-VIServer -Server $destserver  -Confirm:$false
```