ukcloud

Pure commitment.

EXPORTING AND IMPORTING VAPPS AND VAPP TEMPLATES

version 2.0

CONTENTS

OVERVIEW

Our cloud platform was designed and implemented so that it is identical in each data centre, with few differences between the Assured OFFICAL platform and the Elevated OFFICIAL platform. This makes it easy for customers and partners to move workloads between data centres, between security zones and between compatible cloud providers with very little reconfiguration. So, for example, a customer or partner may choose to implement the initial phase of their solution in the Assured OFFICIAL platform, and then move it to the Elevated OFFICIAL platform shortly before it becomes a live service with production data.

The UKCloud portal and vCloud Director provide an easy-to-use graphical user interface (GUI) that lets organisations manually export their workloads and related configuration, and then manually import them into the destination platform. Alternatively, organisations with a large or complex implementation may prefer to automate this process using the UKCloud secure API.

This Blueprint describes how customers can adapt and develop scripts to facilitate the automated migration of workloads on our cloud platform. Note that this guidance is provided without warranty.

GETTING STARTED

Prerequisites

The examples provided in this blueprint make use of the following tools and packages:

- Microsoft PowerShell 4.0: http://www.microsoft.com/powershell
- VMware PowerCLI 5.5: http://www.vmware.com/go/powercli
- VMware OVF Tool 3.5: http://www.vmware.com/go/ovftool
- (Optional) PowerGUI Script Editor: http://powergui.org

You need to install these before using any of the script examples provided.

Preparing your credentials

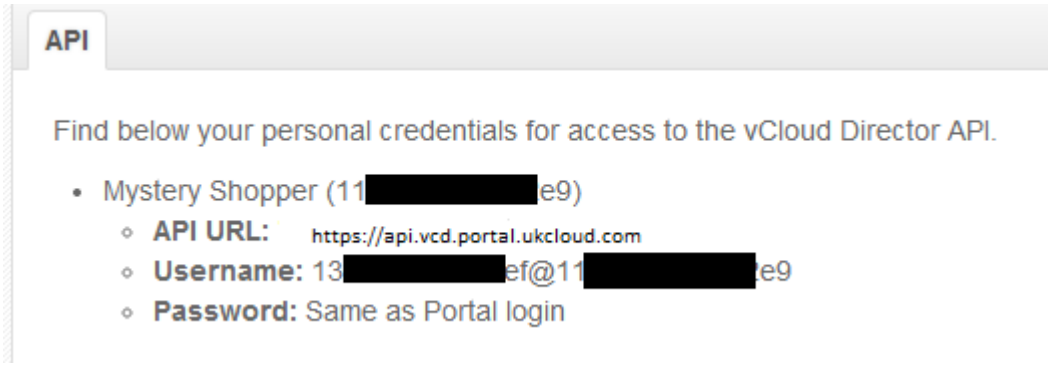
To ensure your scripts can authenticate without manual intervention, you first need to encrypt your credentials, and store them in a file for later use.

You will need to obtain your API credentials from the UKCloud portal at https://portal.UKCloudcloud.com/user/api

See **Error! Reference source not found.**. This page will display your username in the format <username>@<organisation> and your API URL.

Your password is the same as the one you use to log into the UKCloud portal and the vCloud Director GUI.

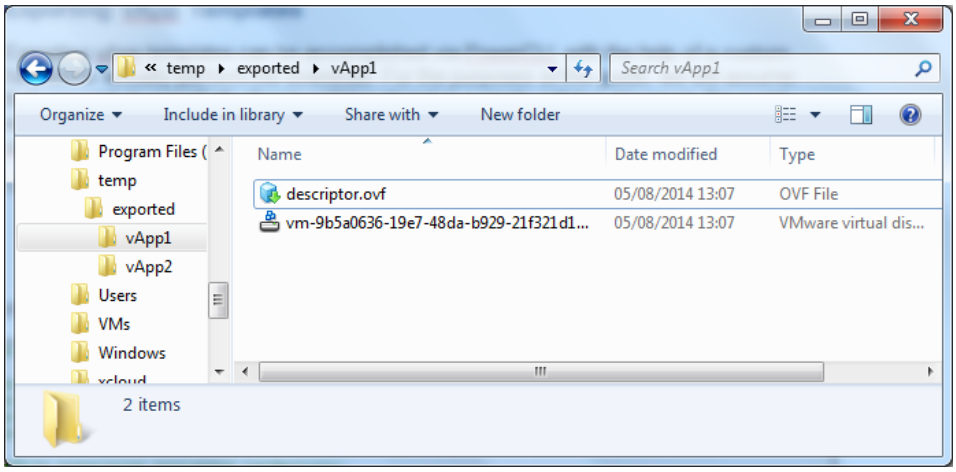
Additional scripts in this blueprint will assume that you have used the script shown on page 9 to store your credentials. This behaviour can be overridden by modifying the variables at the top of each script.



EXPORTING VAPP TEMPLATES

You can export vApp templates using PowerCLI with the help of a custom function to perform the download. This guide assumes you wish to export all vApp templates stored in a single catalogue. The script example available on page 9 can be easily modified so that you can work with multiple catalogues or named vApp templates only.

Your vApp templates will be stored as .OVF and .VMDK files in subfolders of the folder you specify for $ovfpath:



Adjust the variables at the top of the script to suit your environment before you start.

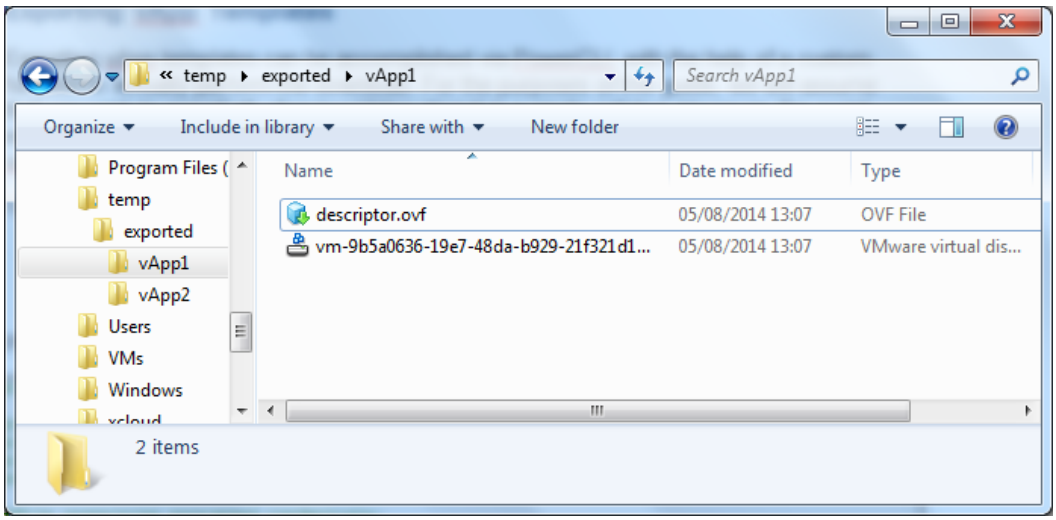
EXPORTING VAPPS

You can export vApps with PowerCLI using the same custom function as for exporting vApp templates.

This blueprint assumes you wish to download all vApps in a given VDC. The script example on page 10 can be easily modified to work in other ways.

A vApp cannot be exported while it is running. If the script encounters a running vApp, it will power it down, export it, and then power it back on again.

Your vApp templates will be stored as .OVF and .VMDK files in subfolders of the folder you specify for $ovfpath:



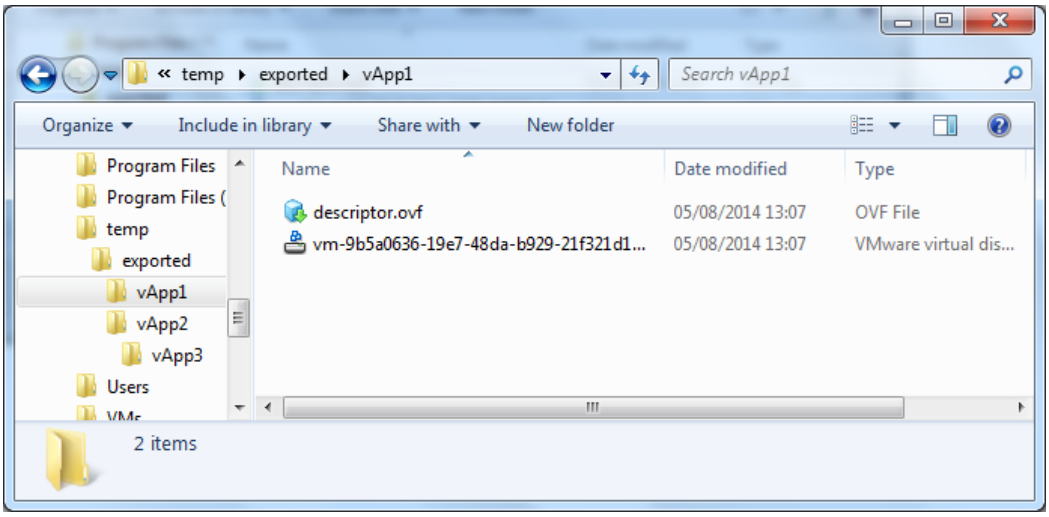
Adjust the variables at the top of the script to suit your environment before you start.

IMPORTING VAPP TEMPLATES

You can import vApp templates using PowerCLI. There is no need for a custom function — all functionality is built in to the cmdlets. This guide assumes you wish to import OVF files and associated VMDK files stored in the folder structure created by the export scripts above.

This script, available on page 12, will iterate through all subfolders of a given folder name and import all the OVF files found into a single catalogue. You need to set the catalogue up before you run the script. Each vApp template will be given the name of its parent folder.

For example, for the following folder structure:



- C:\Temp\Exported\vApp1\descriptor.ovf will be imported to the catalogue as ‘vApp1’.
- C:\Temp\Exported\VApp2\vApp3\descriptor.ovf will not be imported — the script will search only one level deep.
- C:\Temp\Exported\descriptor.ovf will not be imported — the script will work only on OVFs contained in subfolders.

Adjust the variables at the top of the script to suit your environment before you start.

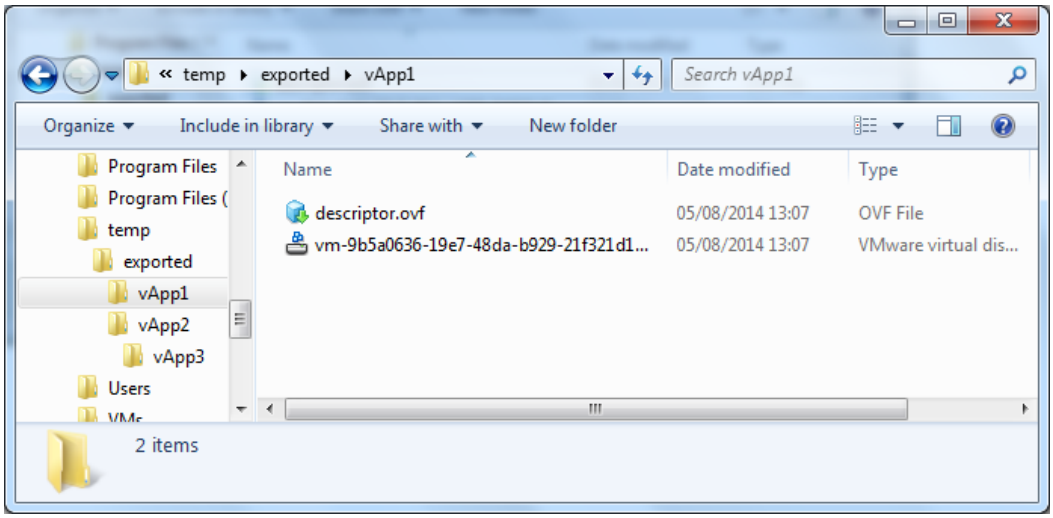
IMPORTING VAPPS

To import your OVF files directly to a vApp (without first creating a vApp template), you'll need to use OVF Tool from VMware, as PowerCLI doesn't offer this functionality.

You'll still need to use a PowerShell script to perform the required logic and call out to OVF Tool when required.

This blueprint assumes you wish to import OVF files and associated VMDK files stored in the folder structure created by the export scripts above. The script, available from page 13, will iterate through all subfolders of a given folder and import all of the OVF files found to vApps in a specified VDC. The vApps will not be powered on after import, as they will usually need further configuration first (for example, connecting to networks).

For example, for the following folder structure:



- C:\Temp\Exported\vApp1\descriptor.ovf will be imported to the catalogue as ‘vApp1’.
- C:\Temp\Exported\VApp2\vApp3\descriptor.ovf will not be imported — the script will search only one level deep.
- C:\Temp\Exported\descriptor.ovf will not be imported — the script will work only on OVFs contained in subfolders.

  Adjust the variables at the top of the script to suit your environment before you start.

THE SCRIPTS

Script for encrypting and storing credentials

```
# Script to encrypt and store credentials

$user = Read-Host "Enter your API username as displayed in the UKCloud portal (user@org):"
$pass = Read-Host -AsSecureString "Enter your password:"
$file = Read-Host "Enter the filename for your encrypted credentials:"

#Write out the credentials file - note this will overwrite the file if it exists
$user | Out-File -FilePath $file
$pass | ConvertFrom-SecureString | Out-File -FilePath $file -Append
```

Script for exporting vApp templates

Adjust the variables at the top of the script to suit your environment before you start and amend the highlighted script section with your API URL.

```
# Script to export all vApp templates in a Catalog to OVF

#-------------------------------------------------------------------------------------
# VARIABLE DEFINITION SECTION
# Please adjust the variables below to suit your environment
#-------------------------------------------------------------------------------------

#File containing encrypted credentials:
$credsfile = "c:\temp\filename.txt"

#Name of the Catalog you wish to export from
$catname = "My-Catalog"

#Path to store your exported vApp Templates (as OVF) - REQUIRES trailing backslash (\)
$ovfpath = "C:\temp\exported\"

#-------------------------------------------------------------------------------------
# AUTHENTICATION SECTION
#-------------------------------------------------------------------------------------
#Retrieve stored credentials
$creds = Get-Content $credsfile
$user = $creds[0]
$pass = $creds[1] | ConvertTo-SecureString

#Split out the username and organisation name into separate variables
$org = ($user -split '@')[1]
$user = ($user -split '@')[0]

#Create PSCredential object to use for authentiation with vCloud Director
$creds = new-object -typename System.Management.Automation.PSCredential -argumentlist $user,$pass

#Connect to vCloud API
Connect-CIServer -server api.vcd.portal.UKCloudcloud.com -Org $org -Credential $creds

#-------------------------------------------------------------------------------------
# FUNCTION DEFINITION SECTION
#-------------------------------------------------------------------------------------
Function Export-OVF {
    <#
        .DESCRIPTION
            Export specific vApp or vAppTemplate to OVF
        .EXAMPLE
            PS C:\> Get-CIVAppTemplate testing | Export-OVF
    #>
    Param (
        $targetDir,
        [Parameter(Mandatory=$True, Position=1, ValueFromPipeline=$true)]
        [PSObject[]]$InputObject
    )
```

```
    Process {
        $InputObject | %{
            if(!($_ | Get-CIView | %{ $_.Link } | where {$_.Rel -eq "download:default"})) {
                Write-Host 'Enabling' $_.name 'for download at' (Get-Date)
                                try {
                                        $headers = @{"Accept"="application/*+xml;version=5.1"}
                                        $headers += @{"x-vcloud-
authorization"=$global:DefaultCIServers[0].sessionid}
                                        $response = invoke-restmethod -uri ($_.ExtensionData.Href +
"/action/enableDownload") -Method POST -Headers $headers -ErrorAction Stop
                    #$_ | Get-CIView | %{ $_.EnableDownload() }
                } catch {
                    Write-Host 'Error enabling' $_.name 'for download'
                                        write-error $error[0]
                }
            }

                    while (!($_ | Get-CIView | %{ $_.Link } | where {$_.Rel -eq "download:default"})) {
                            Start-sleep -s 60
                    }
            $href = $_ | Get-CIView | %{ $_.Link } | where {$_.Rel -eq "download:default"} | %{ $_.Href }
            if(!$targetDir) { $targetDir = $ .name }

            if($href) {
                Write-Host 'Downloading' $_.name 'at' (Get-Date)
                $webClient = New-Object system.net.webclient
                $webClient.Headers.Add('x-vcloud-authorization',$global:DefaultCIServers[0].sessionid)

                if(!(Test-Path $targetDir)) { New-Item -type Directory $targetDir  | out-null }
                $targetDir = (Get-Item $targetDir).FullName

                $baseHref = $href -replace "descriptor.ovf",""
                $webClient.DownloadFile("$href","$($targetDir)\descriptor.ovf")
                [xml]$xmlDescriptor = Get-Content "$($targetDir)\descriptor.ovf"

                $xmlDescriptor.Envelope.References.File | %{ $_.href } | %{
                    try {
                        $webClient.DownloadFile("$($baseHref+$ )","$($targetDir)\$($ )")
                    } catch {
                        write-host -fore red "Error downloading OVF disk named $($_)"
                    }
                }
            } else {
                write-host -fore red "Ovf Download Link Not Found"
            }

        }

        #$inputObject | Get-CIView | %{ $ .DisableDownload() }
    }
}

#--------------------------------------------------------------------------------
# MAIN SCRIPT SECTION
#--------------------------------------------------------------------------------
#Retrieve Catalog object
$catalog = Get-Catalog -Name $catname

#Loop through each vApp template in the catalog, and export to OVF (this might take a while!)
Get-CIVAppTemplate -Catalog $catalog | %{Export-OVF -InputObject $_ -targetDir ($ovfpath + $_.name)}
```

Script for exporting vApps

Adjust the variables at the top of the script to suit your environment before you start and amend the highlighted script section with your API URL.

```
# Script to export all vApps in a VDC to OVF

#--------------------------------------------------------------------------------
```

```
# VARIABLE DEFINITION SECTION
# Please adjust the variables below to suit your environment
#-------------------------------------------------------------------------------

#File containing encrypted credentials:
$credsfile = "c:\temp\filename.txt"

#Name of the VDC you wish to export from
$vdcname = "ACME Inc - (IL0-TRIAL-BASIC)"

#Path to store your exported vApp Templates (as OVF) - REQUIRES trailing backslash (\)
$ovfpath = "C:\temp\exported\"

#-------------------------------------------------------------------------------
# AUTHENTICATION SECTION
#-------------------------------------------------------------------------------
#Retrieve stored credentials
$creds = Get-Content $credsfile
$user = $creds[0]
$pass = $creds[1] | ConvertTo-SecureString

#Split out the username and organisation name into separate variables
$org = ($user -split '@')[1]
$user = ($user -split '@')[0]

#Create PSCredential object to use for authentiation with vCloud Director
$creds = new-object -typename System.Management.Automation.PSCredential -argumentlist $user,$pass

#Connect to vCloud API
Connect-CIServer -server api.vcd.portal.UKCloudcloud.com -Org $org -Credential $creds

#-------------------------------------------------------------------------------
# FUNCTION DEFINITION SECTION
#-------------------------------------------------------------------------------
Function Export-OVF {
    <#
        .DESCRIPTION
            Export specific vApp or vAppTemplate to OVF
        .EXAMPLE
            PS C:\> Get-CIVAppTemplate testing | Export-OVF
    #>
    Param (
        $targetDir,
        [Parameter(Mandatory=$True, Position=1, ValueFromPipeline=$true)]
        [PSObject[]]$InputObject
    )
    Process {
        $InputObject | %{
            if(!($_ | Get-CIView | %{ $_.Link } | where {$_.Rel -eq "download:default"})) {
                Write-Host 'Enabling' $_.name 'for download at' (Get-Date)
                                try {
                                        $headers = @{"Accept"="application/*+xml;version=5.1"}
                                        $headers += @{"x-vcloud-
authorization"=$global:DefaultCIServers[0].sessionid}
                                        $response = invoke-restmethod -uri ($_.ExtensionData.Href +
"/action/enableDownload") -Method POST -Headers $headers -ErrorAction Stop
                    #$_ | Get-CIView | %{ $_.EnableDownload() }
                } catch {
                    Write-Host 'Error enabling' $_.name 'for download'
                                        write-error $error[0]
                }
            }

                        while (!($_ | Get-CIView | %{ $_.Link } | where {$_.Rel -eq "download:default"})) {
                                Start-sleep -s 60
                        }
            $href = $_ | Get-CIView | %{ $_.Link } | where {$_.Rel -eq "download:default"} | %{ $_.Href }
            if(!$targetDir) { $targetDir = $_.name }

            if($href) {
                Write-Host 'Downloading' $_.name 'at' (Get-Date)
                $webClient = New-Object system.net.webclient
                $webClient.Headers.Add('x-vcloud-authorization',$global:DefaultCIServers[0].sessionid)
```

```
                if(!(Test-Path $targetDir)) { New-Item -type Directory $targetDir  | out-null }
                $targetDir = (Get-Item $targetDir).FullName

                $baseHref = $href -replace "descriptor.ovf",""
                $webClient.DownloadFile("$href","$($targetDir)\descriptor.ovf")
                [xml]$xmlDescriptor = Get-Content "$($targetDir)\descriptor.ovf"

                $xmlDescriptor.Envelope.References.File | %{ $_.href } | %{
                    try {
                        $webClient.DownloadFile("$($baseHref+$_)","$($targetDir)\$($_)")
                    } catch {
                        write-host -fore red "Error downloading OVF disk named $($_)"
                    }
                }
            } else {
                write-host -fore red "Ovf Download Link Not Found"
            }

        }

        #$inputObject | Get-CIView | %{ $ .DisableDownload() }
    }
}

#--------------------------------------------------------------------------------
# MAIN SCRIPT SECTION
#--------------------------------------------------------------------------------
#Retrieve OrgVDC Object
$vdc = Get-OrgVdc -Name $vdcname

#Loop through each vApp in the VDC, and export to OVF (this might take a while!)
Get-CIVApp -OrgVdc $vdc | %{
        $waspoweredon = $false

        #Check to see if the vApp was running - if so, power it off
        if ($ .Status -eq 'PoweredOn') {
                $waspoweredon = $true
                Stop-CIVApp -VApp $_ -Confirm:$false
        }

        #Call our custom function to export to OVF
        Export-OVF -InputObject $_ -targetDir ($ovfpath + $_.name)

        #Power on the vApp if it was previously running
        if ($waspoweredon) {Start-CIVApp -VApp $_}
}
```

Script for importing vApp templates

Adjust the variables at the top of the script to suit your environment before you start and amend the highlighted script section with your API URL.

```
# Script to import all OVFs in a folder structure to a pre-existing Catalog

#--------------------------------------------------------------------------------
# VARIABLE DEFINITION SECTION
# Please adjust the variables below to suit your environment
#--------------------------------------------------------------------------------

#File containing encrypted credentials:
$credsfile = "c:\temp\filename.txt"

#Name of the Catalog you wish to import to
$catalogname = "imported"

#Name of the VDC you wish to import to
$vdcname = "ACME Inc - Mystery Shopper (IL0-TRIAL-BASIC)"

#Path to the parent folder containing subfolders with your OVF files - REQUIRES trailing backslash (\)
```

```
$ovfpath = "C:\temp\exported\"

#-------------------------------------------------------------------------------
# AUTHENTICATION SECTION
#-------------------------------------------------------------------------------
#Retrieve stored credentials
$creds = Get-Content $credsfile
$user = $creds[0]
$pass = $creds[1] | ConvertTo-SecureString

#Split out the username and organisation name into separate variables
$org = ($user -split '@')[1]
$user = ($user -split '@')[0]

#Create PSCredential object to use for authentiation with vCloud Director
$creds = new-object -typename System.Management.Automation.PSCredential -argumentlist $user,$pass

#Connect to vCloud API
Connect-CIServer -server api.vcd.portal.UKCloudcloud.com -Org $org -Credential $creds

#-------------------------------------------------------------------------------
# MAIN SCRIPT SECTION
#-------------------------------------------------------------------------------
#Retrieve VDC Object
$vdc = Get-OrgVdc $vdcname

#Retrieve Catalog Object
try {
        $catalog = Get-Catalog -Name $catalogname -ErrorAction Stop
        } catch {
        Write-Host 'Error fetching Catalog - does it exist?'
}

#Loop through each subfolder in the path provided and import each OVF to the catalog (this might take a
while!)
$ovfpath | Get-ChildItem -Directory | %{
        $ovf = $_ | Get-ChildItem -filter '*.ovf'
        Import-CIVAppTemplate -SourcePath ($ovf.FullName) -Name $_.name -orgVdc $vdc -Catalog $catalog
}
```

Script for importing vApps

Adjust the variables at the top of the script to suit your environment before you start and amend the highlighted script section with your API URL.

```
# Script to import all OVFs in a folder structure to a specified VDC

#-------------------------------------------------------------------------------
# VARIABLE DEFINITION SECTION
# Please adjust the variables below to suit your environment
#-------------------------------------------------------------------------------

#File containing encrypted credentials:
$credsfile = "c:\temp\filename.txt"

#Name of the VDC you wish to import to
$vdcname = "ACME Inc - Mystery Shopper (IL0-TRIAL-BASIC)"

#Path to the parent folder containing subfolders with your OVF files - REQUIRES trailing backslash (\)
$ovfpath = "C:\temp\exported\"

#Path to OVFTOOL
$ovftool = "C:\Program Files\VMware\VMware OVF Tool\ovftool.exe"

#-------------------------------------------------------------------------------
# AUTHENTICATION SECTION
#-------------------------------------------------------------------------------
#Retrieve stored credentials
$creds = Get-Content $credsfile
$user = $creds[0]
```

```
$pass = $creds[1] | ConvertTo-SecureString

#Split out the username and organisation name into separate variables
$org = ($user -split '@')[1]
$user = ($user -split '@')[0]

#Create PSCredential object to use for authentiation with vCloud Director
$creds = new-object -typename System.Management.Automation.PSCredential -argumentlist $user,$pass

#Connect to vCloud API
Connect-CIServer -server api.vcd.portal.UKCloudcloud.com -Org $org -Credential $creds

#--------------------------------------------------------------------------------
# MAIN SCRIPT SECTION
#--------------------------------------------------------------------------------
#Loop through each subfolder in the path provided and import each OVF to the VDC (this might take a
while!)
$ovfpath | Get-ChildItem -Directory | %{
        $ovf = $_ | Get-ChildItem -filter '*.ovf'
        $dest = "vcloud://" + $user + ":" + ($creds.GetNetworkCredential().Password |
%{[System.Web.HttpUtility]::UrlEncode($_)})+ "@api.vcd.portal.UKCloudcloud.com:443?org=" + $org + "&vdc="
+ $vdcname + "&vapp=" + $ .name
        & $ovftool $($ovf.FullName) $($dest)
}
```

MORE INFORMATION

Unfortunately, UKCloud Support cannot help you with troubleshooting or modifying any of the scripts provided in this document.

We recommend you test these scripts in a non-production environment first, and that you ensure you are comfortable with the actions each script is taking before you run them in a production environment.

If you need further advice or guidance, contact your Account Director. UKCloud has a talented team of cloud architects and many partners who may be able to assist you.

For further information about UKCloud and how we can help you, please email us at info@ukcloud.com.

ABOUT UKCLOUD

UKCloud is dedicated to the UK Public Sector. We provide assured, agile and value-based true public cloud that enable our customers to deliver enhanced performance through technology.

- We're focused on cloud. Delivering a true cloud platform that is scalable, flexible, assured and cost-effective.
- We're open. You are never locked in. Using industry standards and open source software we enable flexibility and choice across multiple cloud solutions.
- Dedicated to the UK Public Sector. Our business is designed specifically to serve and understand the needs of public sector organisations.
- We develop communities. We bring together communities of users that are able to share datasets, reuse code, test ideas and solve problems.
- Customer engagement. We will only be successful if our customers are successful. We embody this in the promise: Easy to adopt. Easy to use. Easy to leave.

Additional information about UKCloud can be found at www.ukcloud.com or by following us on Twitter at @ukcloudltd.

UKCloud. The power behind public sector technology.

UKCloud Ltd

A8 Cody Technology Park
Ively Road, Farnborough
Hampshire, GU14 0LX

T 01252 303300
E info@ukcloud.com

www.ukcloud.com

 @ukcloudltd
 ukcloudltd
 ukcloud-ltd

UKC-GEN-136 • 08/2017 • version 2