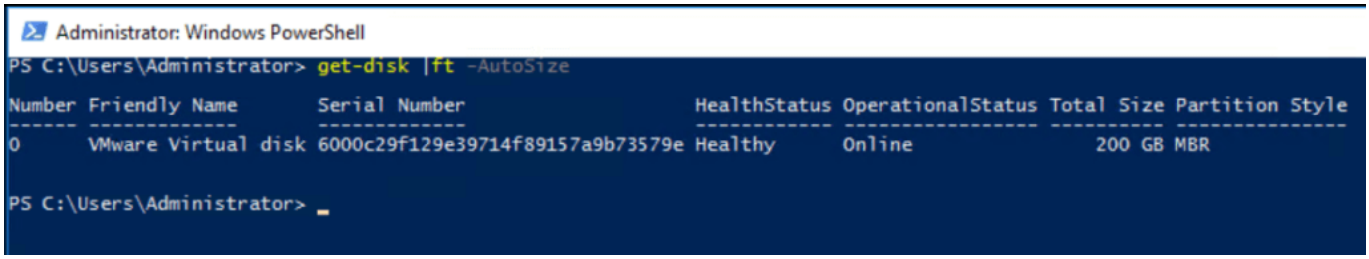
PowerCLI and VVols Part IV: Correlating a Windows NTFS to a VMDK



My last post in this series was about getting a VVol UUID and figuring out what volume on a FlashArray it is. But what about the step before that? If I have a guest OS file system how do I even figure out what VMDK it is? Here, will will discuss this as part of our PowerCLI and VVols series.

PowerCli and VVols 101

There is a basic option, which can potentially be used, which is correlating the bus ID and the unit ID of the device in the guest and matching it to what VMware displays for the virtual disks.

But that always felt to me as somewhat inexact. What if you accidentally look at the wrong VM object and then do something to a volume you do not mean to? Or the opposite?

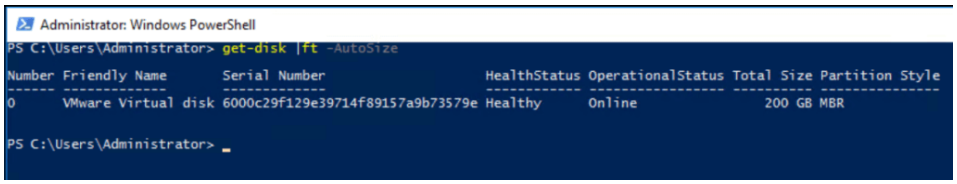
Not ideal. Luckily there is a more exact approach. I will focus this particular post on Windows. I will look at Linux in an upcoming one.

First off, well if this is VVols, wont the volume in Windows just show me my FlashArray (or whatever vendor/model) volume serial number?

No–this is the same idea as with VMFS-based (or NFS, or VSAN-based) virtual disks. VMware virtualizes the VPD information of the volume so you can Storage vMotion it (etc.) between different storage without the guest thinking it sees different storage.

If we look at my Windows VM (which currently has one disk) we will see that the following is its serial number:

NOTE: get-disk only works on Windows 2012 and later, you have to use WMI for earlier versions...



If this field is blank, you likely have the VM advanced setting disk.EnableUUID set to false or unset. This needs to be set to either TRUE.

This setting is the key for this to work. Here is a VMware KB that talks about it:

https://kb.vmware.com/s/article/50121797

Is there a down side to setting this? Well I am still looking into it, but the main thing I know is that if this is set on a VM and that VM is running, when it is cloned or Storage vMotioned, the process will not be offloaded via XCOPY to the array. See this blog post here:

VAAI XCOPY not being used with Powered-On Windows VM

So the serial is 6000C29f129e39714f89157a9b73579e.

If I want to do this from the drive letter, it is fairly easy too. So like tell me the drive letter and I will tell you the serial number. This is a combination of get-partition and get-disk.

So the below will return to me the disk that hosts drive E:
[crayon-6515c03e85e41431895163/]

```
PS C:\Users\Administrator> Get-Partition -DriveLetter E |Get-disk |ft -AutoSize

Number Friendly Name       Serial Number                    HealthStatus OperationalStatus Total Size Partition Style
------ -------------       -------------                    ------------ ----------------- ---------- ---------------
1      VMware Virtual disk 6000c29b43f450e9cacf969ed913c131 Healthy      Online                 40 GB MBR

PS C:\Users\Administrator> Get-Partition -DriveLetter C |Get-disk |ft -AutoSize

Number Friendly Name       Serial Number                    HealthStatus OperationalStatus Total Size Partition Style
------ -------------       -------------                    ------------ ----------------- ---------- ---------------
0      VMware Virtual disk 6000c29f129e39714f89157a9b73579e Healthy      Online                200 GB MBR
```

Let's now look at the virtual disk from a VMware perspective:

```
PS C:\Users\Administrator> $vmdisk = get-vm WindowsVVol | get-harddisk
PS C:\Users\Administrator> $vmdisk.ExtensionData.Backing.Uuid.Replace("-","")
6000C29f129e39714f89157a9b73579e
PS C:\Users\Administrator>
```

Also, 6000C29f129e39714f89157a9b73579e! So perfect, we can make an exact correlation.

So in windows, run get-disk then grab the property SerialNumber and then in vCenter, use get-harddisk against the VM, iterate through the disks and find the one with the matching UUID under extensiondata.backing.uuid.

Fairly simple! And by they way, this process also works for VMFS-based virtual disks too.

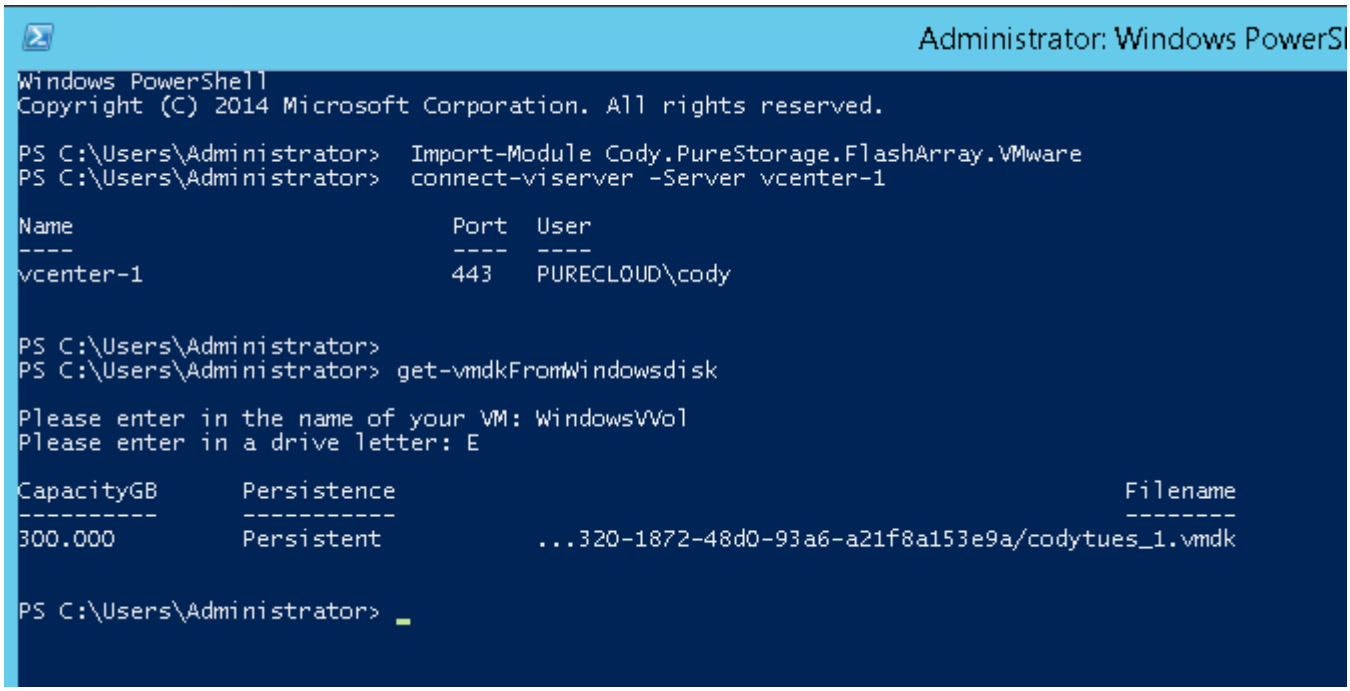
I have added a cmdlet in my Pure Storage/VMware PowerShell module that does this for you though.

get-vmdkFromWindowsdisk

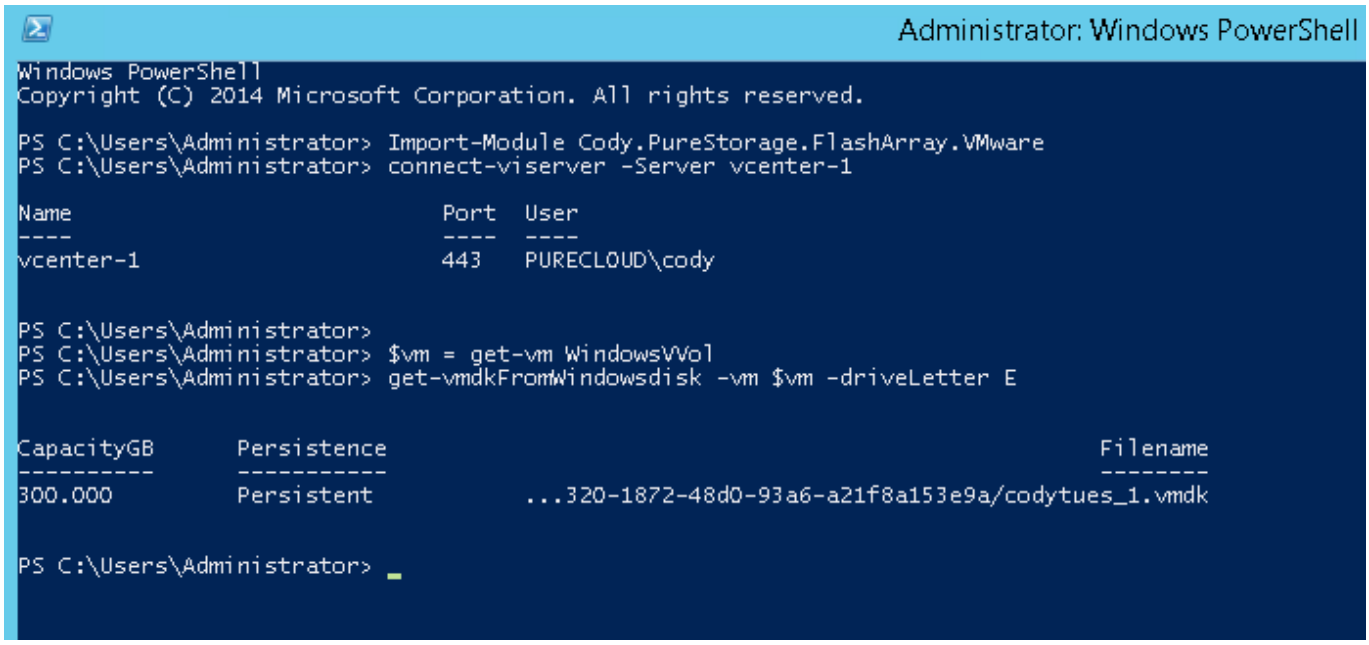
It takes in a VM, a Drive Letter (C, E, whatever) and returns the resulting matching virtual disk (what you would find by running get-harddisk).

This can be run interactively, or by entering parameters. It also supports pipeline input, so you can pass the VM in via pipline. Some examples:

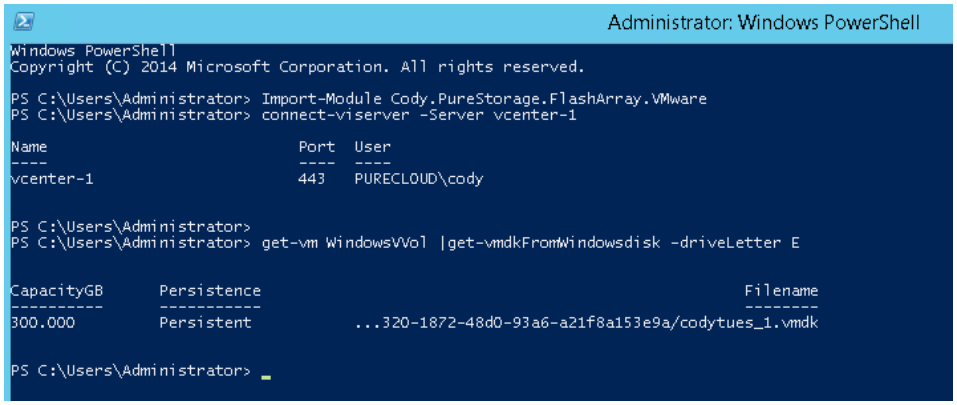
Interactively:



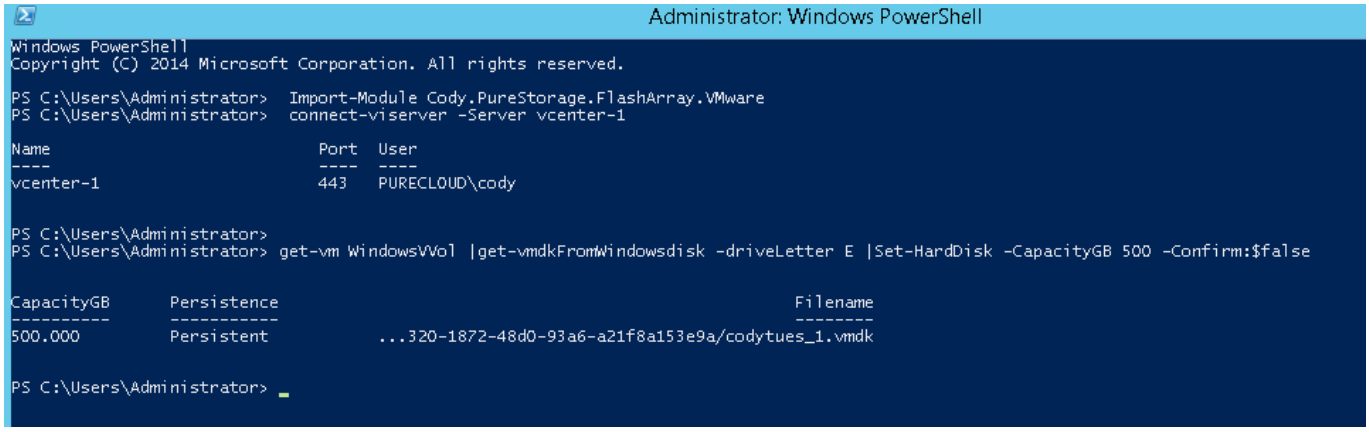
Via parameters:



Via pipeline (the VM object can be piped in):



Returning the resulting disk to another cmdlet. In this case taking the disk and expanding it with set-harddisk. Expands it to 500 GB.



This has been added to my PowerShell module found here.

Note that this will work on any storage platform–not just Pure. It also does not require VVols–it will work with VMFS.

It does have a few requirements:

- The Guest OS must be Windows 2012 or later
- VMware tools must be installed and online
- The Disk.EnableUUID on the VM must be either unset, or set to TRUE.