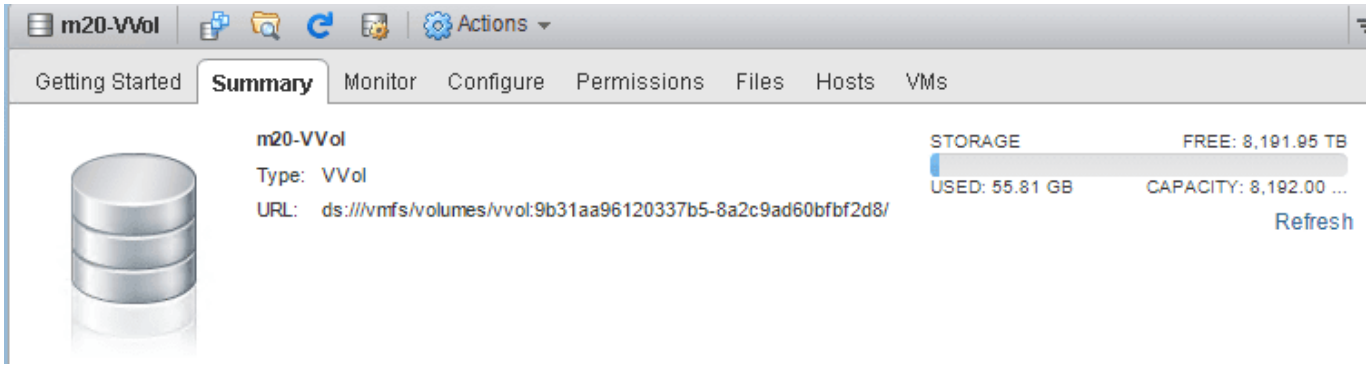
VMware Capacity Reporting Part IV: VVol Capacity Reporting



Storage capacity reporting seems like a pretty straight forward topic. How much storage am I using? But when you introduce the concept of multiple levels of thin provisioning AND data reduction into it, all usage is not equal (does it compress well? does it dedupe well? is it zeroes?).

This multi-part series will break it down in the following sections:

1. VMFS and thin virtual disks
2. VMFS and thick virtual disks
3. Thoughts on VMFS Capacity Reporting
4. VVols and capacity reporting
5. VVols and UNMAP

Let's talk about the ins and outs of these in detail, then of course finish it up with why VVols makes this so much better.

NOTE: Examples in this are given from a FlashArray perspective. So mileage may vary depending on the type of array you have. The VMFS and above layer though are the same for all. This is the benefit of VMFS–it abstracts the physical layer. This is also the downside, as I will describe in these posts.

Okay we agree VMFS is great, but has some glaring downsides. we've all spent time working around for years. Virtual Volumes is the remedy for all of this, in my opinion.

VVols have been **designed specifically** to remove the trade-offs and negatives that come with the traditional mechanism for ESXi storage.

Tradeoffs/choices like VMFS or RDM? Thin or thick virtual disk? Etc. Etc. Etc.

I wrote a detailed synopsis of generically why a VVol is better than an RDMs or a VMDKs here:

Comparing VVols to VMDKs and RDMs

and compared thin VVols vs thin VMDKs here:

Do thin VVols perform better than thin VMDKs?

But let's talk about VVol space reporting specifically.

There are three major parts to capacity reporting:

1. VVol Datastore
2. Data VVol
3. Data VVol Snapshots

But first, let me remind: VVols allow the array to manage reporting as it wishes. So specifics WILL vary from vendor to vendor. So if you are using a different vendor, please of course consult their documentation.

VVol Datastore

What, what, WHAT?! I thought we got rid of datastores? Nah. VMware doesn't want to break your scripts…

new-vm -name CodyVM -datastore Mydatastore

…doesn't break.

VMware certainly doesn't want to change every menu and reference in vCenter. Plus, you still likely want a way to limit how much can be provisioned and a way to choose particular storage in some way. Datastores do this.

What VVols gets rid of is VMFS.

A VVol datastore is **not** a file system. It is **not** a LUN. It is **neither** a volume, a device, **nor** anything physical.

It **is** a capacity limit. It is a capacity limit that **represents** an array.

See this post for more information:

What is a VVol Datastore?

How is VVol datastore usage is reported is entirely up to the storage vendor. Because there is not file system, no LBAs associated with the datastore, no direct files to manage, it is basically a sum of whatever the array reports. It only means, once it is full (used = total) you can no longer provision VVols on that datastore.

The total number is fairly simple to understand. This shows how much can be **used** by those with access to that VVol datastore.

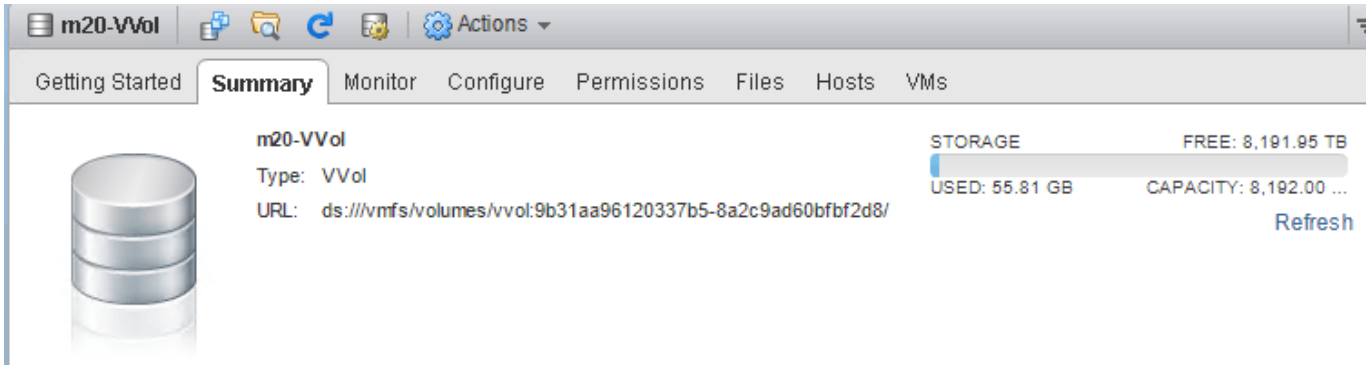
On the FlashArray we default the VVol datastore size to 8 PB, but it can be made to whatever. VMware doesn't really have a practical limit on this number.

The vSphere maximums guide limits VVol datastore size to simply 2^64. Whatever that means. Presumably that is bytes or kilobytes. Which would mean either 16 exabytes or 16 zettabytes. IDC said that

the all humans in total created 16 zettabytes in 2017, so humans will need a new VVol datastore every year or so.

Anyways…

What used **reflects** is the question.



In VMFS, used was the sum of the files. VMDK sizes plus snapshot sizes and then whatever files are also on the file system. ISOs etc.

This is no different with VVol datastores. It is a sum of all of the objects in it. VM config files, VMDKs, snapshots. How those specific object capacities are calculated by your VASA provider is where the VVol magic happens.

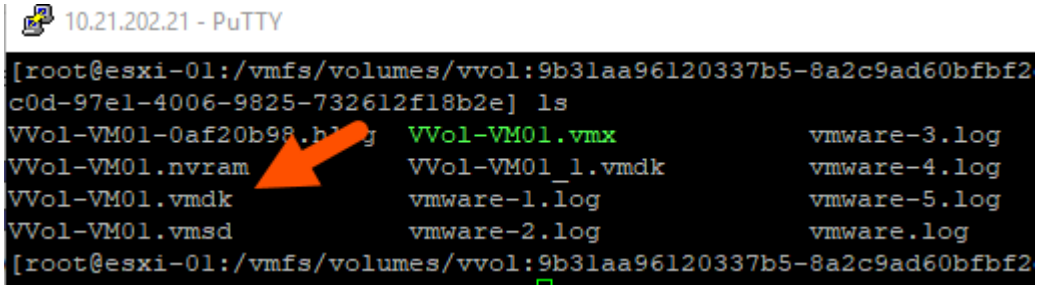
Data VVol

So how is a data VVol capacity reported?

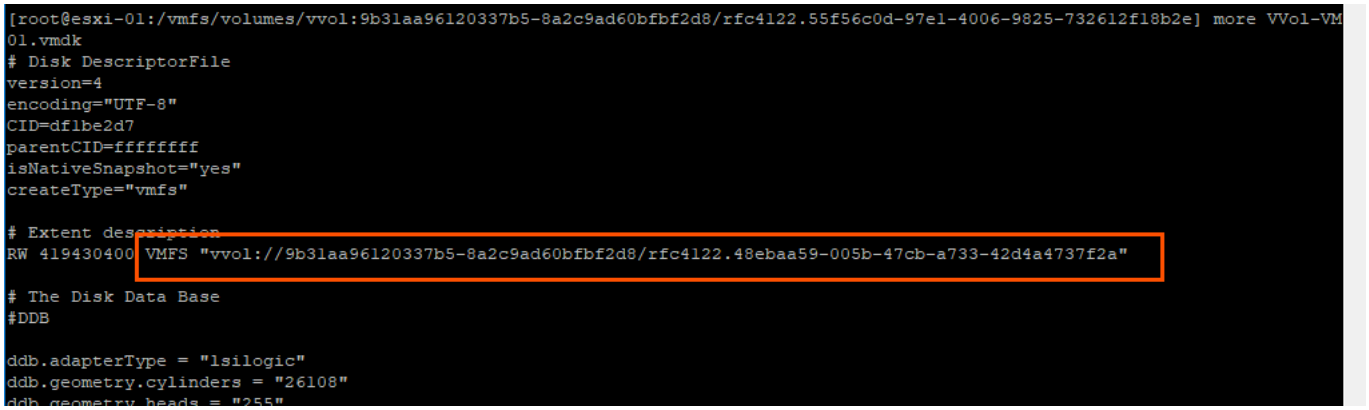
To answer that, let's remind ourselves what a data VVol is. Every time you add a virtual disk to a VM, a vmdk file is created. For a "regular" VMFS/NFS-based VMDK, this is a VMDK and a VMDK-flat file. The VMDK is the descriptor file and the flat file is the actual virtual disk.

For an RDM, there is a VMDK descriptor and then a PRDM pointer file.

VVols, do not change this concept. When you add a virtual disk, you get a VMDK descriptor file–this also acts as a pointer file to the VVol itself.



If you look at the VMDK descriptor, you can see it points to the VVol UUID:



By the way, you can also get this via PowerCLI:
[crayon-6515b2eb3afb6858309248/]

```
PS C:\Users\Administrator> $vm = get-vm VVol-VM01
PS C:\Users\Administrator> $disk = $vm | Get-hardDisk
PS C:\Users\Administrator> $disk[0].ExtensionData.Backing.BackingObjectId
rfc4122.48ebaa59-005b-47cb-a733-42d4a4737f2a
PS C:\Users\Administrator>
```

In short, VMDKs always exist. This allows for new storage constructs to be introduced (e.g. VVols) without breaking everything else (e.g. backup).

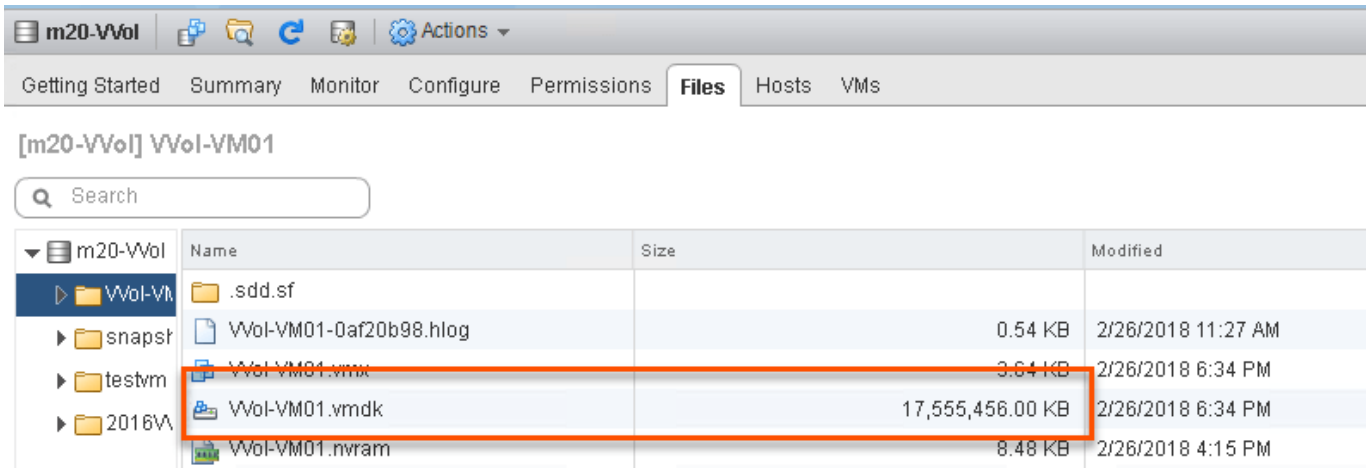
A VMDK has two capacity-related numbers:

- Its provisioned size. This is how much capacity is provides to the VM. So when you create a 50 GB virtual disk, the 50 GB is its provisioned size. This is always the case with any type of virtual disk (thin, thick, RDM, VVol etc).
- Its allocated size. In a thin virtual disk on VMFS this is how much the guest has written. In a thick virtual disk on VMFS,= this is always equal to provisioned size. Allocated size is reflected in VMFS as the size of the VMDK file itself.

The allocated size is what consumes space from a datastore. If a 50 GB thin virtual disk has 10 GB written to it (aka allocated, aka the current file size of the thin virtual disk), the VMFS reports 10 GB used. If a 50 GB thick virtual disk has 10 GB written to it, VMFS still reports 50 GB as used, no matter how much has been written to it.

What about data VVols?

A VMDK still exists on a VVol datastore, and if you look at it, it reports a size:



It reports 17 GB. Where does this number come from?

The nice thing about this, is that it is not the size of that file. The size of that VMDK file is a less than a KB in reality. What it reflects is what we report for the data VVol that it points to. This can be ANY NUMBER the storage array feel like it reports. So we could include data reduction if we want. We could make it thick by reporting the provisioned size. We could always make it 9. This is the flexibility of VVol capacity reporting.

When we were designing VVols, we thought about this. Our default thought was to report the unique space of that data volume, in other words, how much, after global dedupe and compression that volume was using on the array. Or in other words again, how much space would be returned if you were to delete that volume.

Yay! That makes VMware dedupe aware. But we ended up NOT doing that. Well not entirely. Why you might ask? Well:

- It doesn't tell you much about the actual capacity footprint of the VM. It just tells you how well it reduces. Not how much someone wrote. So if I Storage vMotion it to another FlashArray (which has a different data set) it might not reduce as well, so it could be much larger. If I remove it to a non-data reducing array, it could be FAR larger. So it makes a Storage vMotion a question mark.
- If the guest issues UNMAP to the VVol, how do I know if it worked? If the UNMAP simply reclaimed dedupe space, the final footprint of the VM may not change at all.
- If the VM is cloned on the same array, all of the space is no longer unique to either VM (it all becomes deduped) and then both fall to zero. If someone deletes that cloned VM (which might be on VMFS or VVols on that array) the source capacity will immediately climb back up. So deleting data elsewhere could, through no fault of your own, cause your usage to shoot up! See this post for more information on that phenomena.

Because of this, reporting data reduction for that metric is not a good idea for VVols. The concept of GLOBAL data reduction on an array makes it too unpredictable to use as an accounting number for an individual volume (VVol).

So no. That won't work.

Instead we went with host written space. Very similar to a thin virtual disk. With on major exception. If you write contiguous zeroes, it will not be reported in this number. So you are not penalized if an application zeroes out a virtual disk.

If you want to know how we calculate this "host-written" metric on the FlashArray, see this blog post. We

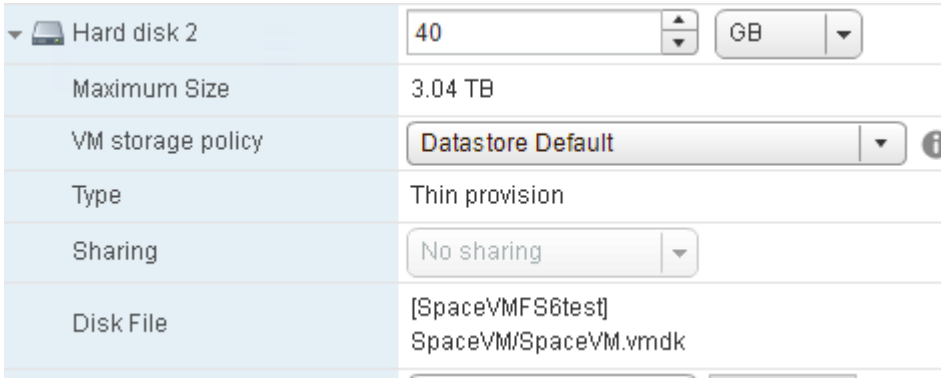
use a metric the “thin_provisioning” (which admittedly is somewhat of a misleading name).

[Detecting what FlashArray VMFS Volumes Have Dead Space]

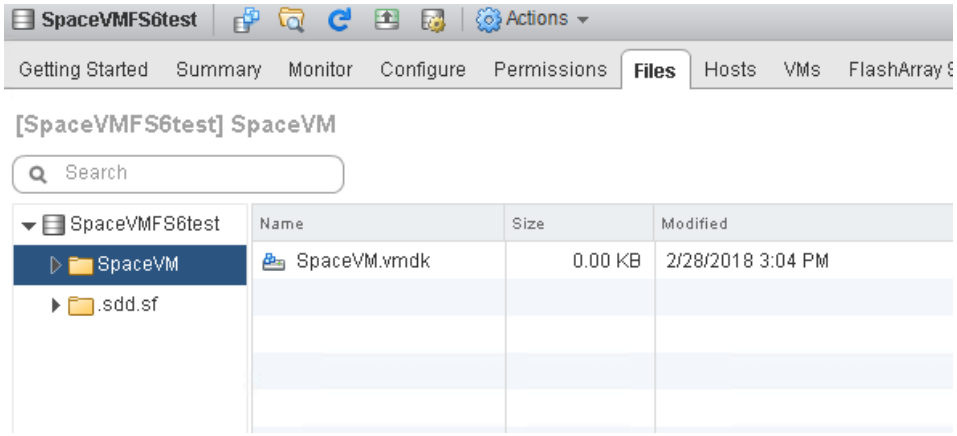
Let me give you an example.

Starting with VMFS.

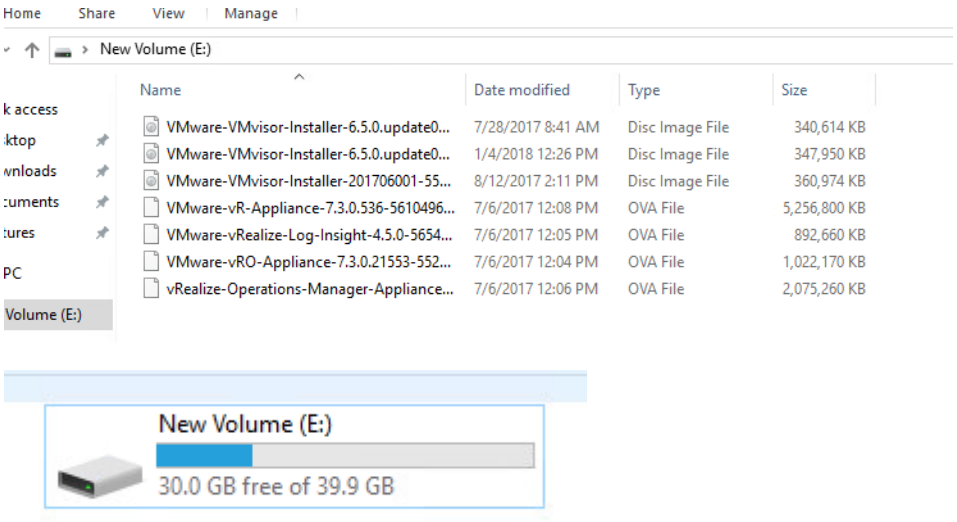
I create a 40 GB thin virtual disk on VMFS:



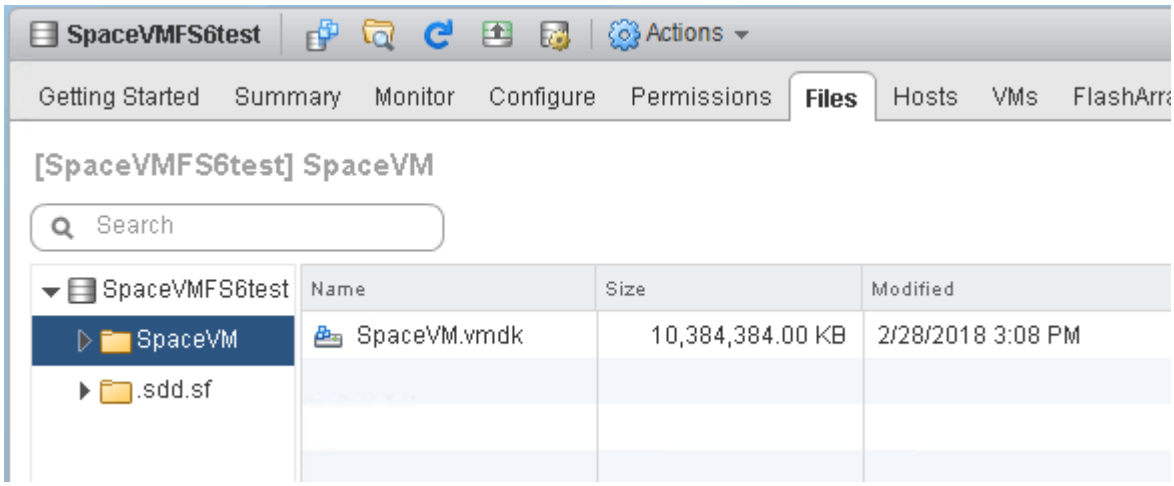
If I look at the VMDK in the VMFS, it is using 0 GB.



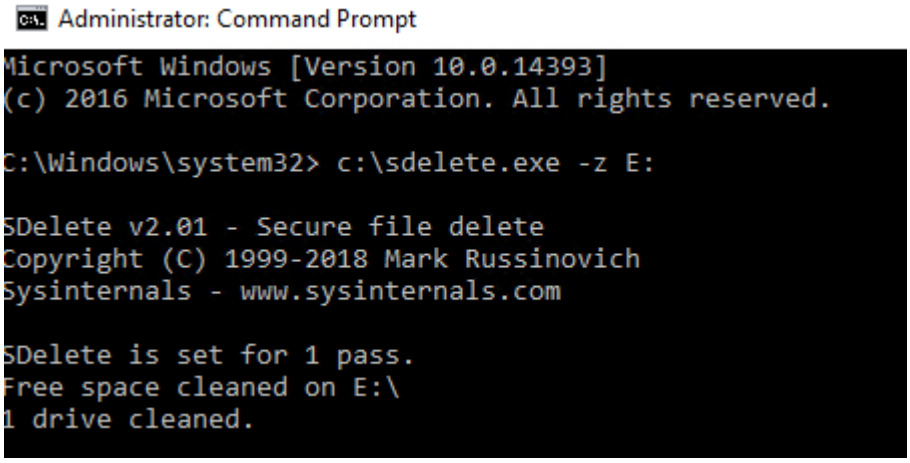
Now, let’s write some data. I will copy 10 GB of ISOs to it.



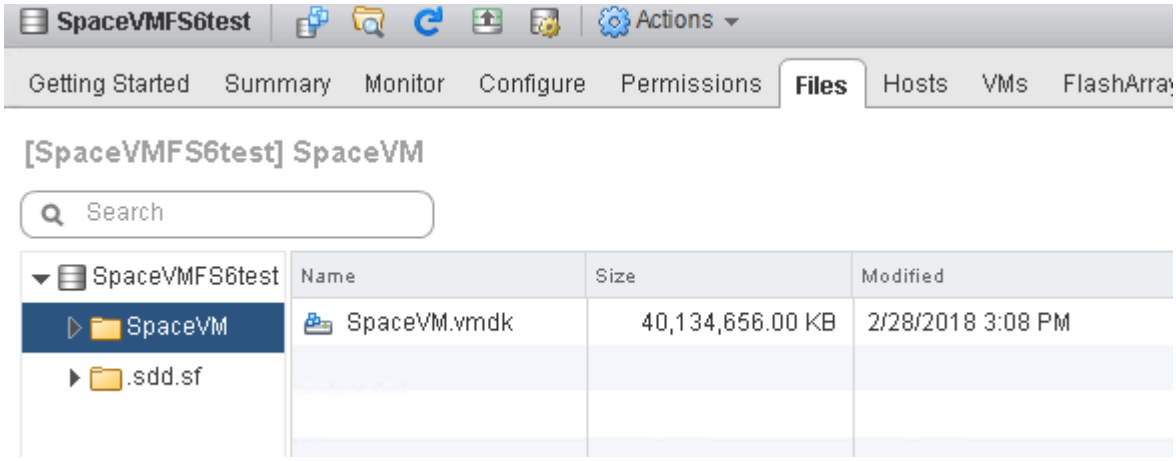
Let’s look at the VMDK on the VMFS:



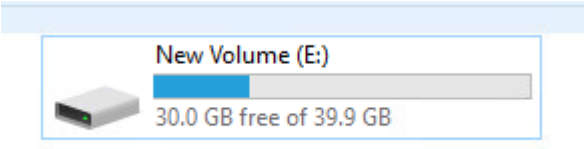
10 GB. But what happens if we write a bunch of zeroes? I’ll use sDelete for that.



The VMDK bloats to 40 GB:

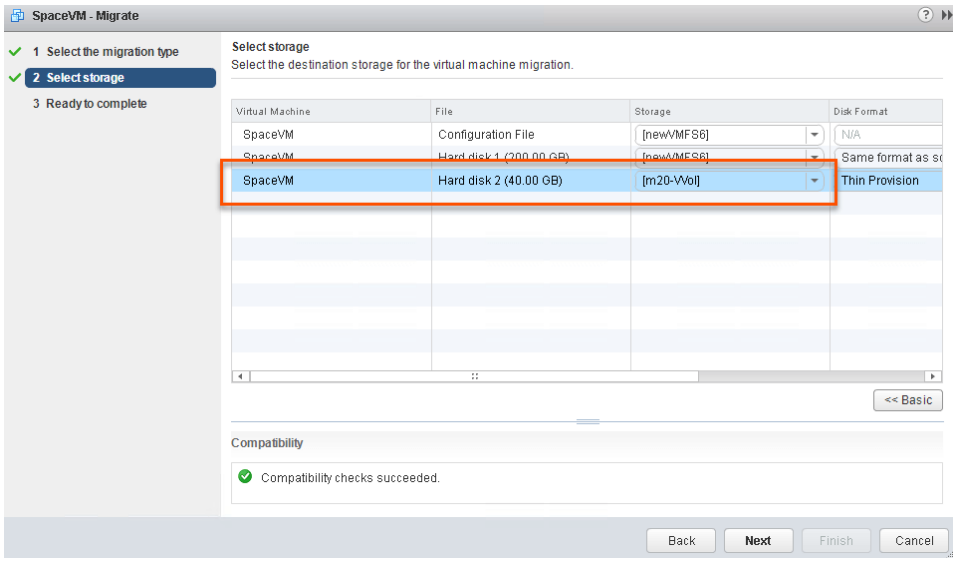


This is because VMFS doesn’t discriminate on what is written. Writing zeroes allocates the thin virtual disk file, just like “normal” I/O. So even though the space isn’t used in the VM…

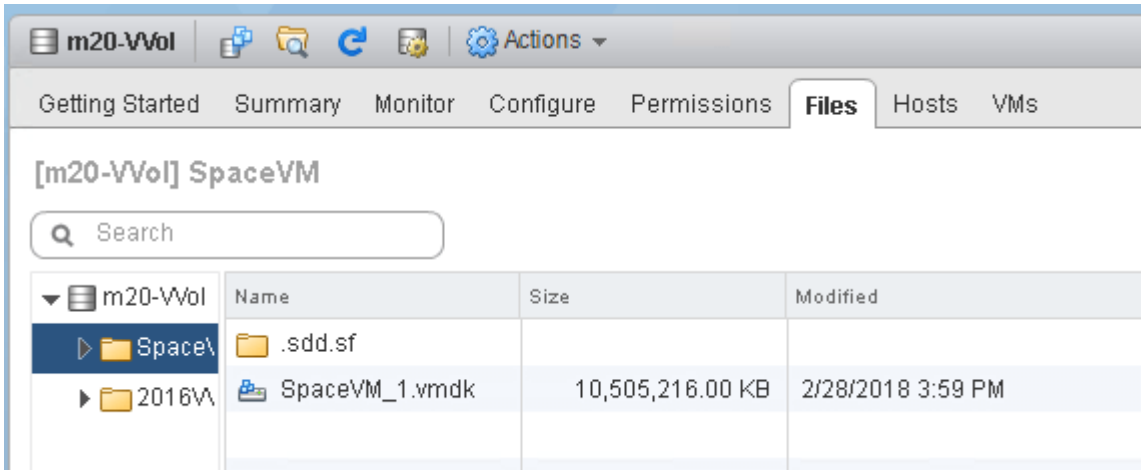


...it is used on the VMFS. Of course a zero-removing array like the FlashArray will remove the zeroes, but the reporting is off in the VMware layer. This issue is discussed in part 1 of this series.

Next, I Storage vMotion this VMDK from VMFS to a VVol:



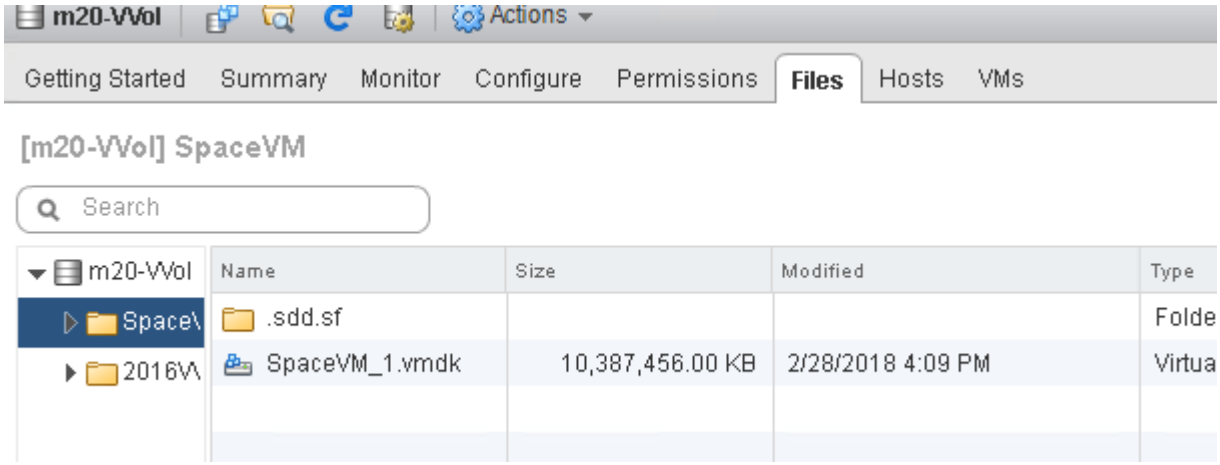
After the Storage vMotion, the virtual disk is now a VVol. If we query the VVol datastore and look at the VMDK pointer there, we see it is 10 GB. The zeroes aren't reported.



But you might say, well that is because ESXi removed the zeroes during the Storage vMotion. And you'd be right! That does happen. So to prove this out, let's write zeroes again.

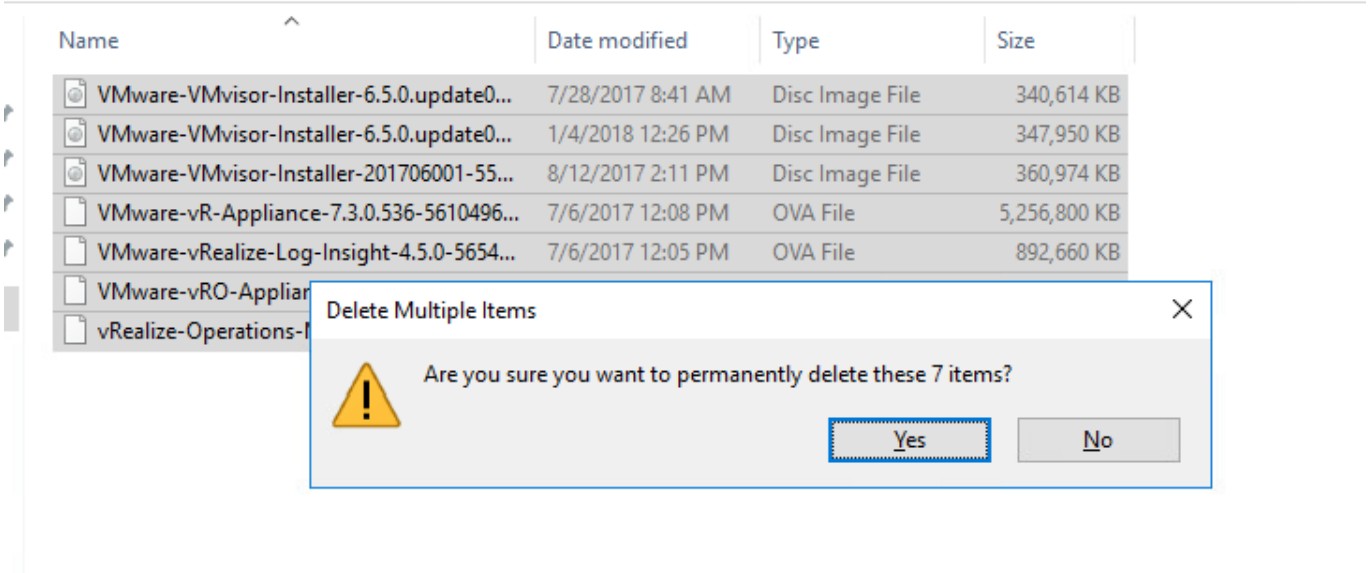
```
C:\Windows\system32> c:\sdelete.exe -z E:

SDelete v2.01 - Secure file delete
Copyright (C) 1999-2018 Mark Russinovich
Sysinternals - www.sysinternals.com

SDelete is set for 1 pass.
Free space cleaned on E:\
1 drive cleaned.

C:\Windows\system32>
```

After writing 30 GB of zeroes, the VVol is still only reported as using 10 GB:

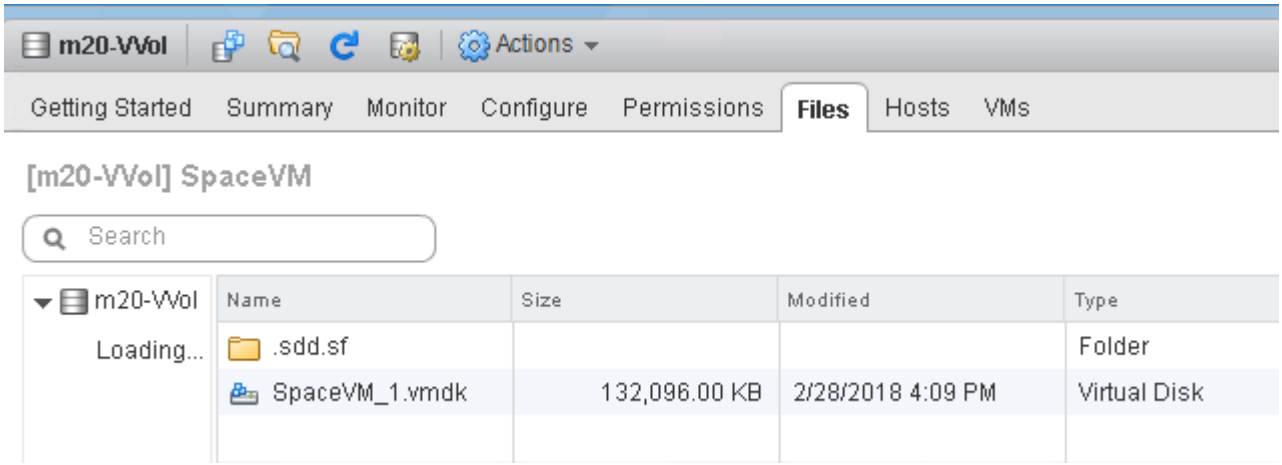


If we Storage vMotion it back the thin VMDK will not bloat again, because ESXi doesn't copy the zeroes, so it will stay 10 GB. Having it behave this way allows for you still to take advantage of data reduction (the array still globally reduces), but still gives you consistent reporting in the VMware layer. The VMDKs report what the guest has written. With the added bonus of not being bloated by zeroing. Plus, you get an accurate idea of how big a VMDK will be if you move it to a different FlashArray, or different array entirely, or to VMFS.

If I delete the 10 GB of files:



NTFS will issue UNMAP and the VMDK will be reported back down to the original size, which is about 130 MB (which is the metadata of NTFS).



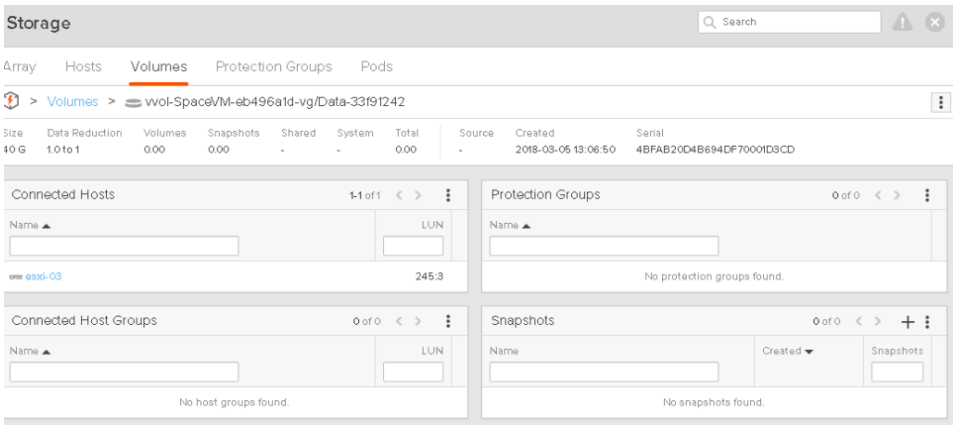
This is a great benefit of VVols–direct automatic UNMAP. There is no file shrinking or translation required. The guest issues UNMAP directly to the array, so all of those old limitations of in-guest UNMAP disappear. It just works.

Data VVol Snapshots

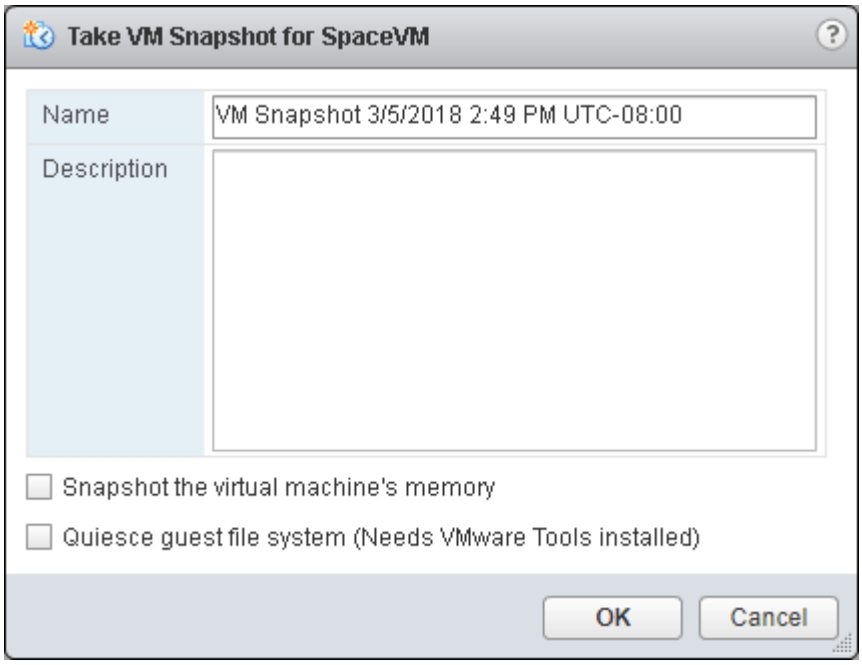
There is also space to innovate in VVol snapshot reporting.

There are some important changes with snapshots when it comes to VVols. I will be posting in more detail on VVol snapshots soon, but have not gotten to it yet, so for reference see Cormac's great post on them here.

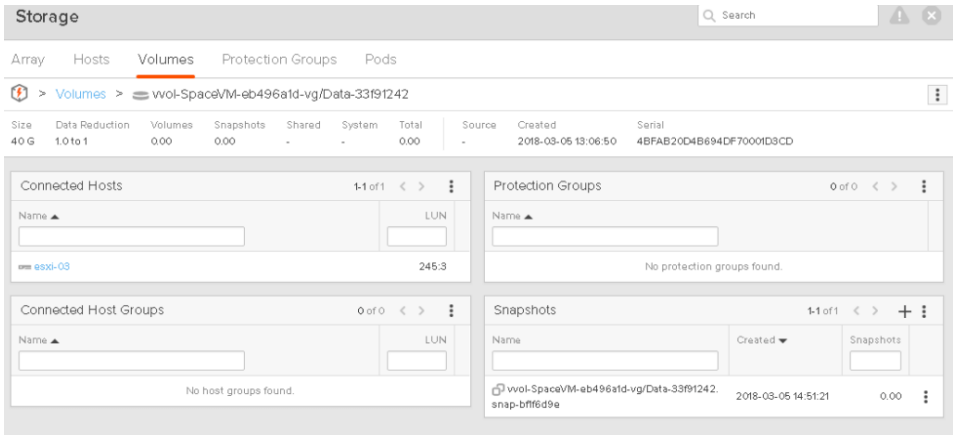
The long and short of it, is that the whole redirect-on-write mechanism for VMware snapshots is gone and the snapshot is offloaded to the array.



So when you click "Take Snapshot"



An array based snapshot is created.



This means you get the benefit of your arrays snapshots. So from a FlashArray perspective you get:

- Instant snapshot creation
- No performance penalty
- Instant restore
- Instant deletion
- Small capacity footprint as the data they point to is globally deduped and compressed with everything else on the array
- Can be copied to physical servers or whatever, because they are just regular array snapshots of a block volume with a file system on them.

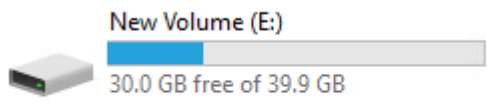
More mobile and more useful. Back to capacity reporting.

So we looked at how we should report capacity for these. We decided against doing the same thing for snapshots as we did with data VVols.

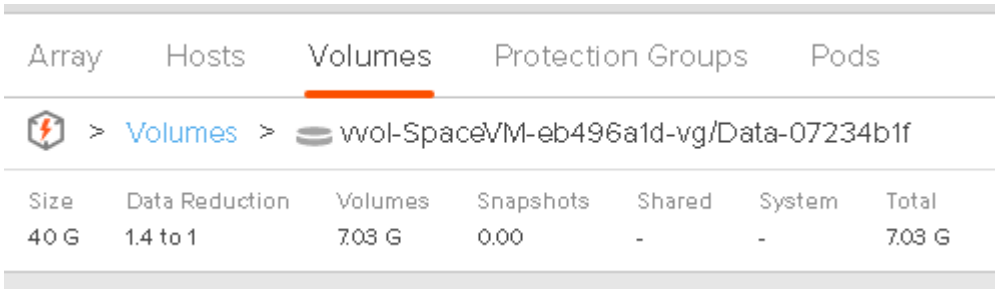
So in the case of snapshots, we report and track their capacity in the VVol datastore with a number that includes data reduction.

Let’s take this example.

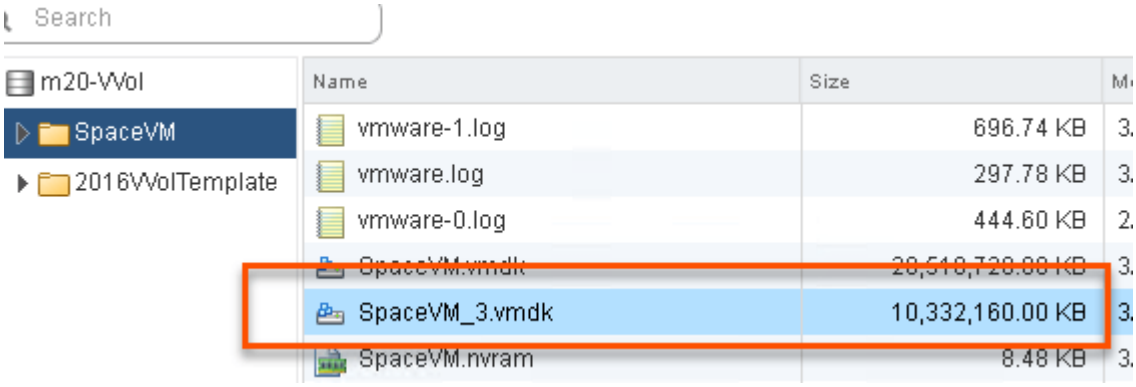
I have 10 GB used on my VVol:



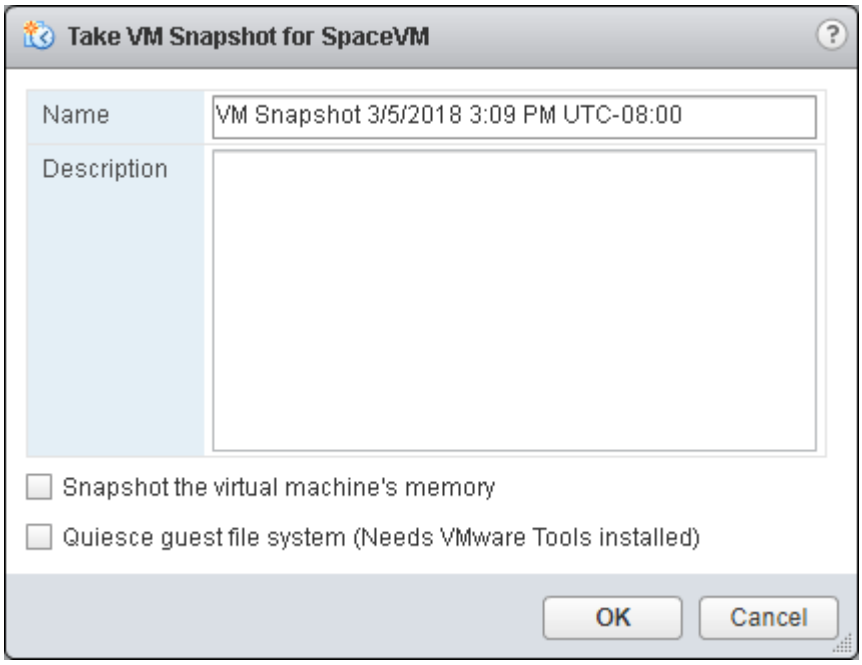
My array has reduced this to 7 GB:



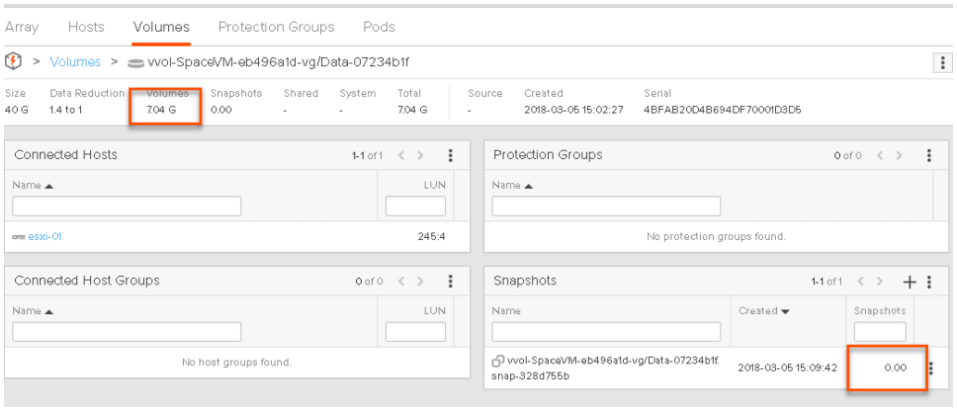
Though as explained above, the VVol VMDK reports 10 GB:



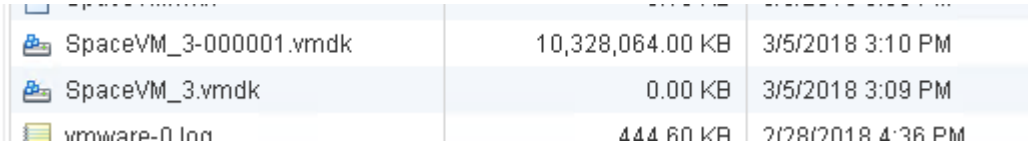
Now, I take a snapshot:



If we look at the array, the volume still reports 7 GB as used and the snapshot is reporting 0 GB. This is because nothing has really changed to the volume, so the snapshot doesn't really have any differences from the source, so it is not really storing any unique space.



This is also reflected in the VMDK sizes for the data VVol and its snapshot. Snapshots are also represented by VMDKs:

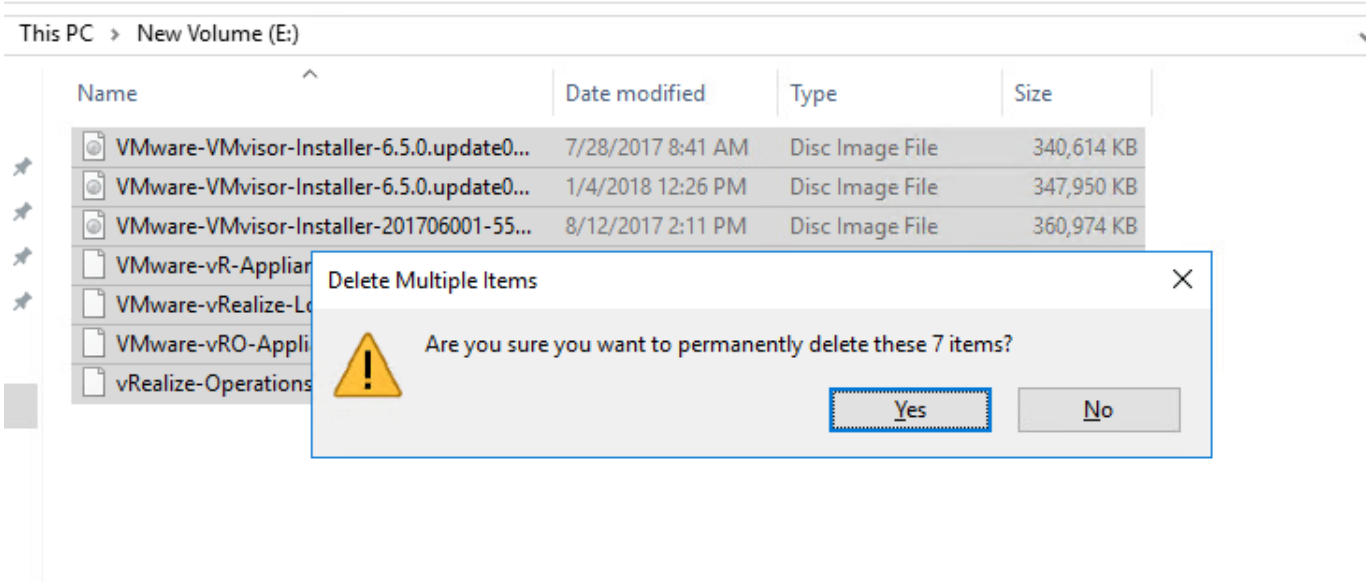


You might say, hey why does 00001 VMDK (which is the snapshot) list the 10 GB not, the original VMDK. This is because when you take the snapshot, VMware switches which VVol the VMDK points to. You are ALWAYS running off of the data VVol. So which ever point-in-time you are running off of is the one that points to the data VVol. So the reporting will change for each individual VMDK depending on snapshot creation, deletion, and restores, but overall it will be the same, because it is just moving between the snapshots.

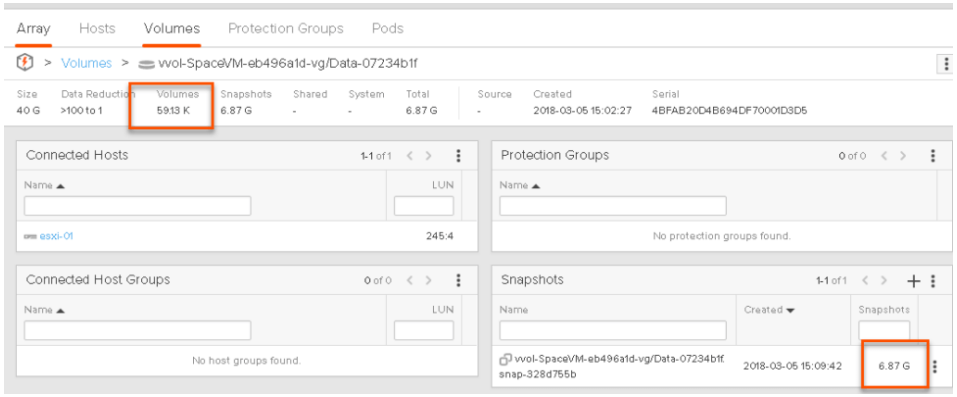
Like I said, I will write a more in-depth post on this to explain it further.

So the VMDK pointing to the data VVol is 10 GB and the VMDK pointing to the snapshot is 0 GB.

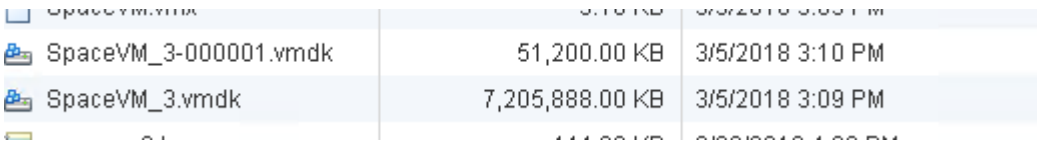
Let's now delete the data in the NTFS. This will free up the space on the data VVol with UNMAP.



But the snapshot will now be uniquely referencing that space. So after the delete and subsequent automatic UNMAP from Windows, the data VVol (the volume) is reporting essentially no space used, a few KB, and the snapshot now preserves the 7 GB of ISOs I deleted.



This is shown in the VVol datastore too:



There are potential downsides to this, and they basically are the same as the downside listed above for the data VVols. But after quite a lot of thought and talking to a variety of customers, we decided that the preference was to allow capacity use reporting in VVols to enjoy data reduction benefits of snapshots. But keep data VVol space reporting pretty similar to thin virtual disks.

The cool thing about this is that since it isn't a file system, it can be easily changed. Or it could be an option–do you want to include data reduction reporting in your snapshots? In your data VVols? Both

neither? Etc. Something we could look into in the future.

The last part of this series I will dive a bit more into UNMAP and VVols.