The Cuddletech Veritas Volume Manager Series

Exploring Layered Volumes in VxVM 3.X

Ben Rockwood
CDLTK
Cuddletech

benr@cuddletech.com

VxVM 3.x adds the powerful capability to create "layered volumes", the ability to create a volume inside a volume. Sadly, practical coverage of creating such volumes has been sparse. In this tutorial we discuse the topic at length and walk thru several examples of creating layered volumes from start to finish with a focus on RAID1+0.

Introduction

A greatly powerful, and often requested, function of Veritas Volume Manager if often the least well understood. This being the usage of Layered Volumes. Working with RAID0+1 (Stripping plus Mirroring), RAID0 (Stripping), and RAID5 are well understood concepts for most. But when it comes to the popular alternative of RAID1+0 (Mirroring plus Stripping) we find a knowledge rift due to poor documentation on the part of Veritas. Well, in this document we're going to fix that. But first, what is RAID1+0?

In a "typical" RAID0+1 volume configuration we take several disks and then create a stripe across thoughs disks (the RAID0 part). Then once complete we do this again on a separate set of disks, and then attach that new stripe to the first creating a mirror (the +1 part). We then have a RAID0+1 volume thats ready to have a filesystem put on it. The point of interest with this setup is that we're actually mirroring a complete stripe (and therefore ALL the disks in that stripe) to another stripe (and therefore ALL of it's disks). The problem here is that if for some reason we need to re-sync the volume we'd need to re-sync a full stripe to a full stripe (very timely) which is a nearly tragic proposition if your talking about 50G+. A far more efficient setup would be to mirror each disk to each disk... in other words, to mirror a bunch of disks on a one-to-one basis, and then build a stripe on top of these mirrors. In this case if we need to re-sync due to a disk failure we can simply sync the failed disk to its mirror, instead of the full stripe. This is the power of RAID1+0; the difference between mirroring the stripes (0+1) and stripping the mirrors (1+0).

If the terms seem to confuse you, try this for size:

```
RAID0 Striping (VxVM says: stripe)
RAID1 Mirroring (VxVM says: mirror)
RAID0+1 Striping plus Mirroring (VxVM says: mirror-stripe)
 Think this: Striped disks, then mirror the stripes
RAID1+0 Mirroring plus Striping (VxVM says: stripe-mirror)
 (Veritas Marketing Dept says: StripePro
 Think this: Mirrored disks, then stripe on top of the mirrors
Concat+Mirror Concatenation plus Mirroring (VxVM says: mirror)
  Same as RAID1
Mirror+Concat Mirroring plus Concatenation (VxVM says: concat-mirror)
  (Veritas Marketing Dept says: ConcatPro)
  Think this: Concatenation on top of mirrored disks.
```

The difference between the different types of combinations of concatenation, striping and mirroring are really just a matter of what is done first. Do you mirror subdisk and then stripe on top of thoughs mirrors or do you stripe several subdisks and then mirror the full stripe? Sometimes thinking "Stripe Then Mirror" is more clear than "Striping And Mirroring". The terms can be confusing, and I guaranty you'll fight with someone about it eventually, but luckily with the nifty terms StripePro and ConcatPro, not only is there no argument of 1s and 0s, but it sounds techy too! When working with layered volumes just remember that, at its heart, its just the ability to embed volumes inside volumes.

Clear? Lets look the actual process of building these RAID1+0 volumes.

How Veritas Volume Manager Makes it Possible

When we create a RAID0+1 volume, mirroring a stripe of subdisks against another stripe of subdisks, we represent each side of the mirror with a plex. The volume contains plexes which contain the subdisks. So if we look at where the data flows int the volume we see that (roughly) the data is written to the volume, which is passed to each of the plexes, which parses it among the subdisks. We don't think much about it, after working with VxVM for awhile, but isn't that kinda limiting? What if you could double the length of that chain? What if a volume itself could be assigned as a subdisk, so that we'd have a new data chain like this: the data is written to the volume, which is passed to each of the plexes, which parses it among the "subvolumes", which pass it to each of its plexes, which parses it among its subdisks?! That'd be nice. Well, that's what Layered Volumes are. We add a set of new objects to VxVM: the subvolume (sv/v2), the subplex(p2), and the sub-subdisk(s2). Because of Layered Volumes we can get down to the level needed to mirror individual disks against each other. Here's a quick overview of how we can use Layered Volumes to get us to the place we wanna be to work with RAID1+0:

```
1) Create your subdisks
2) Create a plex for each subdisk and attach the subdisk to the plex
3) Create a volume  which contains two plexes (one plex mirroring the other)
4) Tell VxVM that the volume you just created is REALLY suppose to be a
 SubVolume (which is treated similarly to a subdisk)
5) Repeat this process till all the subdisks you are gonna use are
 mirrored.
6) Create a new plex, and attach each of the subvolumes to it.
7) Create a volume and assign the plex to it.
8) Done! Start it.
```

Lets clarify on that a bit more, and do a rough sketch of that vxprint would show us, by basically repeating the previous which some new names:

```
 v testvol
 pl testvolplex
So here (v,pl) we have a normal volume and normal plex.
 sv testvol-sub1
This is our subvolume (treated as subdisk) which contains a "normal" volume,
plexes and subdisks.
 v2 layter1
 p2 layer1-plex1
 s2 disk01-01
 p2 layer1-plex2
 s2 disk02-01
So here is what looks like a normal volume, except there are "2"s on each
of the object names (v becomes v2, pl becomes p2, sd becomes s2) to represent
the fact that they are part of a subvolume.
 sv testvol-sub2
This is our second subvolume
 v2 layer2
 p2 layer2-plex1
 s2 disk03-01
 p2 layer2-plex2
 s2 disk04-01
Just like above.
 sv   testvol-sub3
Etc,etc. Just keep going all you like.
```

See how that fits? Each of the p2's obviously are concats. Because we're mirroring one subdisk against another subdisk (not multiples) we don't/can't stripe. The striping is done at the plex (not subplex) level. So if we looked at the above example again with RAID type in mind, we'd see something like this:

```
 v
 pl stripe
 sv
 v2
 p2 concat
 s2
 p2 concat
 s2
 sv
 v2
 ...
```

And just in case those examples above are still hard to understand, just think of it like this (I'll use the wrong, but more familiar object names):

```
 v testvol
 pl testplex (stripe)
 sv testvol-sub1:
  v layer1
  pl layer1-plex1 (concat)
  sd disk01-01
```

```
  pl layer1-plex2 (concat)
  sd disk02-01
 sv testvol-sub2:
  v  layer2
  .....
```

At this point lets point out that you can do more than just RAID1+0 with VxVM, you can also do a hybrid "concat+0". "Concat+0" is just like RAID1+0, but instead of striping the primary plex (as seen above) you would concatenate it. Frankly, the only reason you might end up doing a contact over a stripe would be to compensate for a major design flaw (or lack of resources). In a stripe each column (and hence subvolume in our case) must be identical, concats don't have this problem, so you can "mix-n-match" disks into the volume you'd like. I'll demonstrate this later. Further, know that Veritas has a groovy marketing name for RAID1+0 and CONCAT+0, they are respectively: Stripe-Pro and Concat-Pro. Snazzy huh? You'll see this more in the walk throughs later.

The concepts should be starting to be clear now. This gives you the background to look at the building of an actual volume. So lets move onto actually working with layered volumes.

Working with Layered Volumes

As is often true with Veritas Volume Manager, you can work with layered volumes in two ways: vxmake or vxassist. Using vxmake you can build each object one-by-one until you have a finished volume. Alternately, you can use vxassist in one of several ways, to build the layered volume for you. We will address both methods bellow, but let me say that it is NOT recommended that you build layered volumes via vxmake, there are just too may places you can make a mistake that you won't see until its too late. I will step through the vxmake process only to give you a better idea of how everything works together, and I do recommend that you try it if you have a system that is designated as TEST ONLY, and data isn't important (at all). Vxassist is far easier and safer to work with, as you'll see. Layered volumes aren't difficult at all with the use of the tool. We'll look at vxmake first.

Method 1: VxMake

For the test I'm going to use four 9G drives. I've already added them to a disk group (oradg) and I'm first going start by creating subdisks for each of the disks:

```
root@sun5:/root# vxmake -g oradg sd oradg02-01 oradg02,0,17678493
root@sun5:/root# vxmake -g oradg sd oradg03-01 oradg03,0,17678493
root@sun5:/root# vxmake -g oradg sd oradg04-01 oradg04,0,17678493
root@sun5:/root# vxmake -g oradg sd oradg05-01 oradg05,0,17678493
```

Now we'll create some plexes that the subdisk can be apart of, remember that each subdisk will be the only one in the plex (see above):

```
root@sun5:/root# vxmake -g oradg plex layer1-01 sd=oradg02-01
root@sun5:/root# vxmake -g oradg plex layer1-02 sd=oradg03-01
root@sun5:/root# vxmake -g oradg plex layer2-01 sd=oradg04-01
root@sun5:/root# vxmake -g oradg plex layer2-02 sd=oradg05-01
```

Now we'll create two volumes. We're looking to create a layered volume with two columns, which means the primary plex of the layered vol should only have 2 subdisks, each of thoughs are actually volumes (the layering part), so lets create the volumes:

```
root@sun5:/root# vxmake -g oradg -U fsgen vol layer1 plex=layer1-01,layer1-02
root@sun5:/root# vxmake -g oradg -U fsgen vol layer2 plex=layer2-01,layer2-02
```

Great! Now lets look at what we've got so far (vxprint -hrt):

```
v  layer1       fsgen        DISABLED EMPTY    17678493 ROUND     -
pl layer1-01    layer1       DISABLED EMPTY    17678493 CONCAT    -        RW
sd oradg02-01   layer1-01    oradg02  0        17678493 0         c1t11d0  ENA
pl layer1-02    layer1       DISABLED EMPTY    17678493 CONCAT    -        RW
sd oradg03-01   layer1-02    oradg03  0        17678493 0         c1t10d0  ENA

v  layer2       fsgen        DISABLED EMPTY    17678493 ROUND     -
pl layer2-01    layer2       DISABLED EMPTY    17678493 CONCAT    -        RW
sd oradg04-01   layer2-01    oradg04  0        17678493 0         c1t5d0   ENA
pl layer2-02    layer2       DISABLED EMPTY    17678493 CONCAT    -        RW
sd oradg05-01   layer2-02    oradg05  0        17678493 0         c1t8d0   ENA
```

So thoughs are going to be the "legs", if you will. Now, PAY ATTENTION! YOU MUST START THE VOLUMES BEFORE CONTINUING!!! If you don't start the volumes you'll keep building and building with everything looking fine, until you try to start the finished layered vol and find that you can't because the subvols aren't started, and you can't start subvols! So at this point lets:

```
root@sun5:/root# vxvol -g oradg start layer1
root@sun5:/root# vxvol -g oradg start layer2
```

Note that both this syncs take the normal amount of time (along time!). Make sure you have a book and some coffee for the syncing. Once their started you'll see this (vxprint -hrt):

```
v  layer1       fsgen        ENABLED  ACTIVE   17678493 ROUND     -
pl layer1-01    layer1       ENABLED  ACTIVE   17678493 CONCAT    -        RW
sd oradg02-01   layer1-01    oradg02  0        17678493 0         c1t11d0  ENA
pl layer1-02    layer1       ENABLED  ACTIVE   17678493 CONCAT    -        RW
sd oradg03-01   layer1-02    oradg03  0        17678493 0         c1t10d0  ENA

v  layer2       fsgen        ENABLED  ACTIVE   17678493 ROUND     -
pl layer2-01    layer2       ENABLED  ACTIVE   17678493 CONCAT    -        RW
sd oradg04-01   layer2-01    oradg04  0        17678493 0         c1t5d0   ENA
pl layer2-02    layer2       ENABLED  ACTIVE   17678493 CONCAT    -        RW
sd oradg05-01   layer2-02    oradg05  0        17678493 0         c1t8d0   ENA
```

Now here's the magic part, this is the key! Use vxedit to tell Veritas Volume Manager that the volumes you just started are actually subdisks!:

```
root@sun5:/root# vxedit -g oradg set layered=on layer1
root@sun5:/root# vxedit -g oradg set layered=on layer2
```

Now the kool part, use the volumes (now subvolumes) to create subdisks!:

```
root@sun5:/root# vxmake -g oradg sd test-subvol1 layer1,0,17678493
```

```
root@sun5:/root# vxmake -g oradg sd test-subvol2 layer2,0,17678493
```

And then create a plex for the subdisks:

```
root@sun5:/root#  vxmake -g oradg plex plex1 sd=test-subvol1,test-subvol2
```

Notice that I didn't specify a layout in the "vxmake plex" statement, so this volumes is going to be a CONCAT-PRO volume. To create a STRIPE-PRO volume I could have used this:

```
vxmake -g oradg plex plex1 layout=stripe ncolumns=2 stwidth=128 sd=test-subvol1,test-
subvol2
```

Now, we just need to create the volume object to wrap it all up:

```
root@sun5:/root#  vxmake -g oradg -U fsgen vol cat-pro-vol plex=plex1
```

Presto! We're done! We just need to take a look at it, and then start the volume. Here's what it looks like now (vxprint -hrt):

```
v  cat-pro-vol  fsgen        DISABLED EMPTY    35356986 ROUND     -
pl plex1        cat-pro-vol  DISABLED EMPTY    35356986 CONCAT    -        RW
sv test-subvol1 plex1        layer1   1        17678493 0         2/2      ENA
v2 layer1       fsgen        ENABLED  ACTIVE   17678493 ROUND     -
p2 layer1-01    layer1       ENABLED  ACTIVE   17678493 CONCAT    -        RW
s2 oradg02-01   layer1-01    oradg02  0        17678493 0         c1t11d0  ENA
p2 layer1-02    layer1       ENABLED  ACTIVE   17678493 CONCAT    -        RW
s2 oradg03-01   layer1-02    oradg03  0        17678493 0         c1t10d0  ENA
sv test-subvol2 plex1        layer2   1        17678493 0         2/2      ENA
v2 layer2       fsgen        ENABLED  ACTIVE   17678493 ROUND     -
p2 layer2-01    layer2       ENABLED  ACTIVE   17678493 CONCAT    -        RW
s2 oradg04-01   layer2-01    oradg04  0        17678493 0         c1t5d0   ENA
p2 layer2-02    layer2       ENABLED  ACTIVE   17678493 CONCAT    -        RW
s2 oradg05-01   layer2-02    oradg05  0        17678493 0         c1t8d0   ENA
```

Now we just need to start it and then take a final peek:

```
root@sun5:/root# vxvol -g oradg start cat-pro-vol
 (LONG time spend syncing)
 (vxprint -hrt:)
v  cat-pro-vol  fsgen        ENABLED  ACTIVE   35356986 ROUND     -
pl plex1        cat-pro-vol  ENABLED  ACTIVE   35356986 CONCAT    -        RW
sv test-subvol1 plex1        layer1   1        17678493 0         2/2      ENA
v2 layer1       fsgen        ENABLED  ACTIVE   17678493 ROUND     -
p2 layer1-01    layer1       ENABLED  ACTIVE   17678493 CONCAT    -        RW
s2 oradg02-01   layer1-01    oradg02  0        17678493 0         c1t11d0  ENA
p2 layer1-02    layer1       ENABLED  ACTIVE   17678493 CONCAT    -        RW
s2 oradg03-01   layer1-02    oradg03  0        17678493 0         c1t10d0  ENA
sv test-subvol2 plex1        layer2   1        17678493 0         2/2      ENA
v2 layer2       fsgen        ENABLED  ACTIVE   17678493 ROUND     -
p2 layer2-01    layer2       ENABLED  ACTIVE   17678493 CONCAT    -        RW
s2 oradg04-01   layer2-01    oradg04  0        17678493 0         c1t5d0   ENA
p2 layer2-02    layer2       ENABLED  ACTIVE   17678493 CONCAT    -        RW
s2 oradg05-01   layer2-02    oradg05  0        17678493 0         c1t8d0   ENA
```

Bingo! Done! Just create your file system and your ready to go! You can see from the step through that really your are just creating a bunch of volumes, telling Veritas Volume Manager to just think of the volumes as subdisk objects (via vxedit), and then building another volume with the subvolumes as subdisks. The only trick along the way is to make sure to start the subvolumes before you you convert them via vxedit and build the main volume.

Now, that process can be fraught with mistakes, so lets look at the far better and far easier way of doing things no that you've got a better idea of whats going on.

Method 2: VxAssist

Rather than mess with vxmake and all the objects on a very personal basis you can employ vxassist to do all the dirty work. If you have any amount of experience with vxassist you'll know that the more information you can supply to vxassist the better the end product will be.

I'm going to use vxassist to build a stripe-pro volume from four disks and I want the volume to be 1G in size:

```
root@sun7:/usr/sbin# vxassist -g testdg make stripeprovol 1g  layout=stripe-mirror \
   testdg01 testdg02 testdg03 testdg04
```

This command took along time to return due to syncing, but when it came back I had this to show for it (vxprint -hrt):

```
v  stripeprovol -            ENABLED  ACTIVE   2097152  fsgen     stripeprovol-
03 SELECT
pl stripeprovol-03 stripeprovol ENABLED ACTIVE 2097152  STRIPE    2/128    RW
sv stripeprovol-S01 stripeprovol-03 stripeprovol-L01 1 1048576 0/0 2/2     ENA
v2 stripeprovol-L01 -        ENABLED  ACTIVE   1048576  fsgen     -        SELECT
p2 stripeprovol-P01 stripeprovol-L01 ENABLED ACTIVE 1048576 CONCAT -       RW
s2 testdg02-02  stripeprovol-P01 testdg02 0    1048576  0         c1t3d0   ENA
p2 stripeprovol-P02 stripeprovol-L01 ENABLED ACTIVE 1048576 CONCAT -       RW
s2 testdg04-02  stripeprovol-P02 testdg04 0    1048576  0         c1t3d2   ENA
sv stripeprovol-S02 stripeprovol-03 stripeprovol-L02 1 1048576 1/0 2/2     ENA
v2 stripeprovol-L02 -        ENABLED  ACTIVE   1048576  fsgen     -        SELECT
p2 stripeprovol-P03 stripeprovol-L02 ENABLED ACTIVE 1048576 CONCAT -       RW
s2 testdg03-02  stripeprovol-P03 testdg03 0    1048576  0         c1t3d1   ENA
p2 stripeprovol-P04 stripeprovol-L02 ENABLED ACTIVE 1048576 CONCAT -       RW
s2 testdg01-02  stripeprovol-P04 testdg01 0    1048576  0         c1t3d30  ENA
```

Pretty kool, huh? Quick, efficient, and poorly named; everything you love about vxassist. I can then go a bit further and explore my sizing options to see how much I can grow my new volume if I need to:

```
root@sun7:/usr/sbin# vxassist -g testdg maxgrow stripeprovol
Volume stripeprovol can be extended by 282050560 to 284147712 (138744Mb)
```

See? Just like a normal volume. Now comes the beauty part. When you look at that seemingly unmanageable mess of objects above does it really make you want to tear it apart and work on it like you might other "normal" volumes? Probably not. And you'd be wise to feel that way, there are just too many places to get confused or make a mistake when real data is involved. What if you could get back to a more normal point of view? Luckily you can, check this out:

root@sun7:/usr/sbin# vxassist -g testdg convert stripeprovol layout=mirror-stripe

Pretty easy line huh? We're simply telling Veritas Volume Manager that we want it convert (actually, its a relayout) our "stripe-mirror" to a "mirror-stripe". The beauty of this is that there is NO CONVERSION TIME! Because the process is a simple change in the way that VxVM looks at the disks rather than the way that it positions the data there is no conversion time. Lets look at what our earlier built volume looks like now that we've converted it (vxprint -hrt):

```
v  stripeprovol -            ENABLED  ACTIVE   2097152  fsgen     -        SE-
LECT
pl stripeprovol-01 stripeprovol ENABLED ACTIVE 2097152  STRIPE    2/128    RW
sd testdg02-01  stripeprovol-01 testdg02 0     1048576  0/0       c1t3d0   ENA
sd testdg03-01  stripeprovol-01 testdg03 0     1048576  1/0       c1t3d1   ENA
pl stripeprovol-02 stripeprovol ENABLED ACTIVE 2097152  STRIPE    2/128    RW
sd testdg04-01  stripeprovol-02 testdg04 0     1048576  0/0       c1t3d2   ENA
sd testdg01-01  stripeprovol-02 testdg01 0     1048576  1/0       c1t3d30  ENA
```

Look alittle less intimidating? It should. Using vxassist we can convert back and forth seamlessly between stripe-mirror and mirror-stripe. The same can be done with concat-mirror and mirror-concat. The only catch is that you can't use "vxassist convert" to convert from stripe-mirror to concat-mirror (or any other concat to stripe combination), for that you'd need to do a full-blown vxrelayout.

I should also note that the vxassist conversion will work on any properly formed volume. This means that you can build a volume by hand (vxmake) as a RAID0+1 and then convert it to the RAID1+0. Or you could create a volume using vxassist as RAID1+0 and then convert it, work on it (tear apart the plexes, whatever) and then when finished just convert it back! This make everything really simple to work with.

And just to round this out, lets add take one last look at things by adding a Dirty Region Log to our volume:

```
root@sun7:/usr/sbin# vxassist -g testdg addlog stripeprovol

(vxprint -hrt)
v  stripeprovol -            ENABLED  ACTIVE   2097152  fsgen     -        SELECT
pl stripeprovol-01 stripeprovol ENABLED ACTIVE 2097152  STRIPE    2/128    RW
sd testdg02-01  stripeprovol-01 testdg02 0     1048576  0/0       c1t3d0   ENA
sd testdg03-01  stripeprovol-01 testdg03 0     1048576  1/0       c1t3d1   ENA
pl stripeprovol-02 stripeprovol ENABLED ACTIVE 2097152  STRIPE    2/128    RW
sd testdg04-01  stripeprovol-02 testdg04 0     1048576  0/0       c1t3d2   ENA
sd testdg01-01  stripeprovol-02 testdg01 0     1048576  1/0       c1t3d30  ENA
pl stripeprovol-03 stripeprovol ENABLED ACTIVE LOGONLY  CONCAT    -        RW
sd testdg02-02  stripeprovol-03 testdg02 1048576 33     LOG       c1t3d0   ENA
```

Now, let's convert out newly logged volume back into a stripe-mirror (StripePro):

```
root@sun7:/usr/sbin# vxassist -g testdg convert stripeprovol layout=stripe-mirror

(vxprint -hrt):
v  stripeprovol -            ENABLED  ACTIVE   2097152  fsgen     stripeprovol-
04 SELECT
pl stripeprovol-04 stripeprovol ENABLED ACTIVE 2097152  STRIPE    2/128    RW
sv stripeprovol-S01 stripeprovol-04 stripeprovol-L01 1 1048576 0/0 3/3     ENA
v2 stripeprovol-L01 -        ENABLED  ACTIVE   1048576  fsgen     -        SELECT
p2 stripeprovol-P01 stripeprovol-L01 ENABLED ACTIVE LOGONLY CONCAT -       RW
```

```
s2 testdg06-01  stripeprovol-P01 testdg06 0    33       LOG       c1t3d3   ENA
p2 stripeprovol-P02 stripeprovol-L01 ENABLED ACTIVE 1048576 CONCAT -       RW
s2 testdg02-03  stripeprovol-P02 testdg02 0    1048576  0         c1t3d0   ENA
p2 stripeprovol-P03 stripeprovol-L01 ENABLED ACTIVE 1048576 CONCAT -       RW
s2 testdg04-02  stripeprovol-P03 testdg04 0    1048576  0         c1t3d2   ENA
sv stripeprovol-S02 stripeprovol-04 stripeprovol-L02 1 1048576 1/0 3/3     ENA
v2 stripeprovol-L02 -        ENABLED  ACTIVE   1048576  fsgen     -        SELECT
p2 stripeprovol-P04 stripeprovol-L02 ENABLED ACTIVE LOGONLY CONCAT -       RW
s2 testdg06-02  stripeprovol-P04 testdg06 33   33       LOG       c1t3d3   ENA
p2 stripeprovol-P05 stripeprovol-L02 ENABLED ACTIVE 1048576 CONCAT -       RW
s2 testdg03-02  stripeprovol-P05 testdg03 0    1048576  0         c1t3d1   ENA
p2 stripeprovol-P06 stripeprovol-L02 ENABLED ACTIVE 1048576 CONCAT -       RW
s2 testdg01-02  stripeprovol-P06 testdg01 0    1048576  0         c1t3d30  ENA
```

Interesting, huh? Now lets convert back just for fun:

```
root@sun7:/usr/sbin# vxassist -g testdg convert stripeprovol layout=mirror-stripe

(vxprint -hrt):
v  stripeprovol -            ENABLED  ACTIVE   2097152  fsgen     -        SELECT
pl stripeprovol-01 stripeprovol ENABLED ACTIVE 2097152  STRIPE    2/128    RW
sd testdg02-04  stripeprovol-01 testdg02 0     1048576  0/0       c1t3d0   ENA
sd testdg03-01  stripeprovol-01 testdg03 0     1048576  1/0       c1t3d1   ENA
pl stripeprovol-02 stripeprovol ENABLED ACTIVE 2097152  STRIPE    2/128    RW
sd testdg04-01  stripeprovol-02 testdg04 0     1048576  0/0       c1t3d2   ENA
sd testdg01-01  stripeprovol-02 testdg01 0     1048576  1/0       c1t3d30  ENA
pl stripeprovol-04 stripeprovol ENABLED ACTIVE LOGONLY  CONCAT    -        RW
sd testdg06-01  stripeprovol-04 testdg06 0     33       LOG       c1t3d3   ENA
```

Notice, how the log (the same log) gets seeing in both subvolumes. This goes against all sense, but there it is. This is one of thoughs questions to save for the Veritas Engineers while waiting for phone support. But you can see how easily everything works together.

The Wrap Up

I hope that this tutorial has shown you both the power and the ease of implementing layered volumes under Veritas Volume Manager. This tutorial very well could have been merely a lesson in the usage of the "vxassist convert" tool, but hopefully by looking at the structure of the volumes and going through a vxmake of a volume you can get a solid grasp of the concepts behind volumes.

A precaution is in order (yes another one), earlier I had said that vxassist made conversion of properly formed volumes very easy. Notice closely that I said "properly formed volumes". Several cases have popped up on various mailing lists over the last couple months from people who had significant trouble with conversion to and from layered volumes because, for one reason or another, something got changed so that the volume was no longer proper and the volume was stuck in one state or another. Take, for example, a situation where you have a StripePro volume and a disk fails and you replace it. In this example you might end up replacing the disk and then accidental making one column longer than another; the volume is still in tact, and technically everything should still jive, but you eventually decide to convert back to mirror-stripe and the operation fails due to inconstancy. This is an example of things that can and do happen. Great care should be taken when dealing with layered volumes. Let me further

supplement this by saying that many admins don't consider layered vols ready for prime time. Even in the latest Release Notes for VxVM 3.1.1 there are still bugs and inconsistencies that make it too unreliable. Here are two two "Veritas Incident Numbers" found on pages 19 and 20 of the VXVM 3.1.1 Release Notes:

```
45668 Due to the current implementation to handle the resize of layered
 volumes, it is recommended not to grow or shrink layered volumes
 (stripe-mirror, concat-mirror, and so on) while resynchronization
 is going on.
 Internally, the Volume Manager converts the layout of layered volumes
 and updates the configuration database before it shrinks or grows their
 sizes.  This causes any ongoing operation, such as the resynchronization,
 to fail.
 [snip]
 Although this release supports layered volumes, creating volumes
 with mixed layout types is not recommended.  For example, adding a
 mirror use the vxassist mirror command specifying "layout=mirror-stripe
 nmirror=1" to an existing volume with stripe-mirror layout is not
 recommended. (Se Sun BugID 4339626)
none While doing relayout on a mirrored volume the vxassist command
 keeps the volume as mirrored even if the layout attribute is specified as
 stripe or nomirror.  For example, see the following commands:
  # vxassist make vol 1024 layout=mirror-stripe ncol=3
  # vxassist relayout vol layout=stripe ncol=2
 The volume vol is converted to a 2 column volume, but it is
 still mirrored even if the layout attribute is specified as stripe
 and nomirror.
```

Both the above incidents might seem like no-brainers, but these are real problems that any of us can run into. As a general rule, any time you make changes to the volume you should consider converting back to RAID0+1, you'd be better off stuck in 0+1 than in 1+0. As with anything in Veritas Volume Manager (Veritas products in general really) you need to be very careful and look at things from every possible angle before making changes. When real data is on the line the amount of time it takes to think and rethink things through is nothing compared to the time it'll take to restore if you make a mistake.

```
For More Information See:
 - The Veritas Volume Manager Documentation (Included with Product)
 - The Veritas Volume Manager Support Site (www.veritas.com, click support)

For Help contact:
 - The Veritas-Vx Mailing List
 (http://mailman.eng.auburn.edu/mailman/listinfo/veritas-vx)

Special Thanx to:
 - Doug Hughes (doug@Eng.Auburn.EDU), one of the nets greatest Vx gurus,
 for giving the tutorial a final look over and sharing his thoughts.
```

Warning

Disclaimer: This documentation is provided without warranty of any kind. If you blast your data due to something found in this tutorial its not my or Cuddletech's fault. All methods found here should be tested in a sandbox environment where you can experiment and become comfortable BEFORE being used on real data or in production environments. Veritas Corp had nothing to do creation of this doc and has no connections of any kind to Cuddletech.com.