ZFS For Newbies

Dan Langille
FreeBSD Fridays: 14 Aug 2020
online

`@dlangille`
`https://dan.langille.org/`

Disclaimer

- This is ZFS for newbies
    - grossly simplified
        - stuff omitted
            - options skipped
                - because newbies….

What?

- a short history of the origins
- an overview of how ZFS works
- replacing a failed drive
- why you don’t want a RAID card
- scalability
- data integrity (detection of file corruption)
- why you’ll love snapshots
- sending of filesystems to remote servers
- creating a mirror
- how to create a ZFS array with multiple drives which can lose up to 3 drives without loss of data.
- mounting datasets anywhere in other datasets
- using zfs to save your current install before upgrading it
- simple recommendations for ZFS arrays
- why single drive ZFS is better than no ZFS
- no, you don’t need ECC
- quotas
- monitoring ZFS

Origins

- 2001 - Started at Sun Microsystems
- 2005 - released as part of OpenSolaris
- 2008 - released as part of FreeBSD
- 2010 - OpenSolaris stopped, Illumos forked
- 2013 - First stable release of ZFS On Linux
- 2013 - OpenZFS umbrella project
- 2016 - Ubuntu includes ZFS by default

Stuff you can look up

- ZFS is a 128-bit file system
- 2^48: number of entries in any individual directory
- 16 exbibytes (2^64 bytes): maximum size of a single file
- 256 quadrillion zebibytes (2^128 bytes): maximum size of any zpool
- 2^64: number of zpools in a system
- 2^64: number of file systems in a zpool

Gross simplification

- the next few slides are overly simplified

zpool

- Group your drives together: pool -> **zpool**
- **zpool create** - operates on drives (vdevs - virtual devices)

zpool variations

- create a mirror, stripe, or raidz
- mirror from 2..N drives
- create a raidz[1..3] from 4+ drives
- stripe 1+ drives

file systems

- **zfs create** - operates on a zpool, creates filesystems
- filesystems can contain filesystems - hierarchy with inherited properties
- e.g. `zroot/users/dan/projects/foo`
- mounted at `/usr/home/dan/projects/foo`
- Based on pathname, you don’t always know zfs name

pooling your drives

- no more:
    - out of space on **`/var/db`**
    - loads of free space on **`/usr`**’

zpool

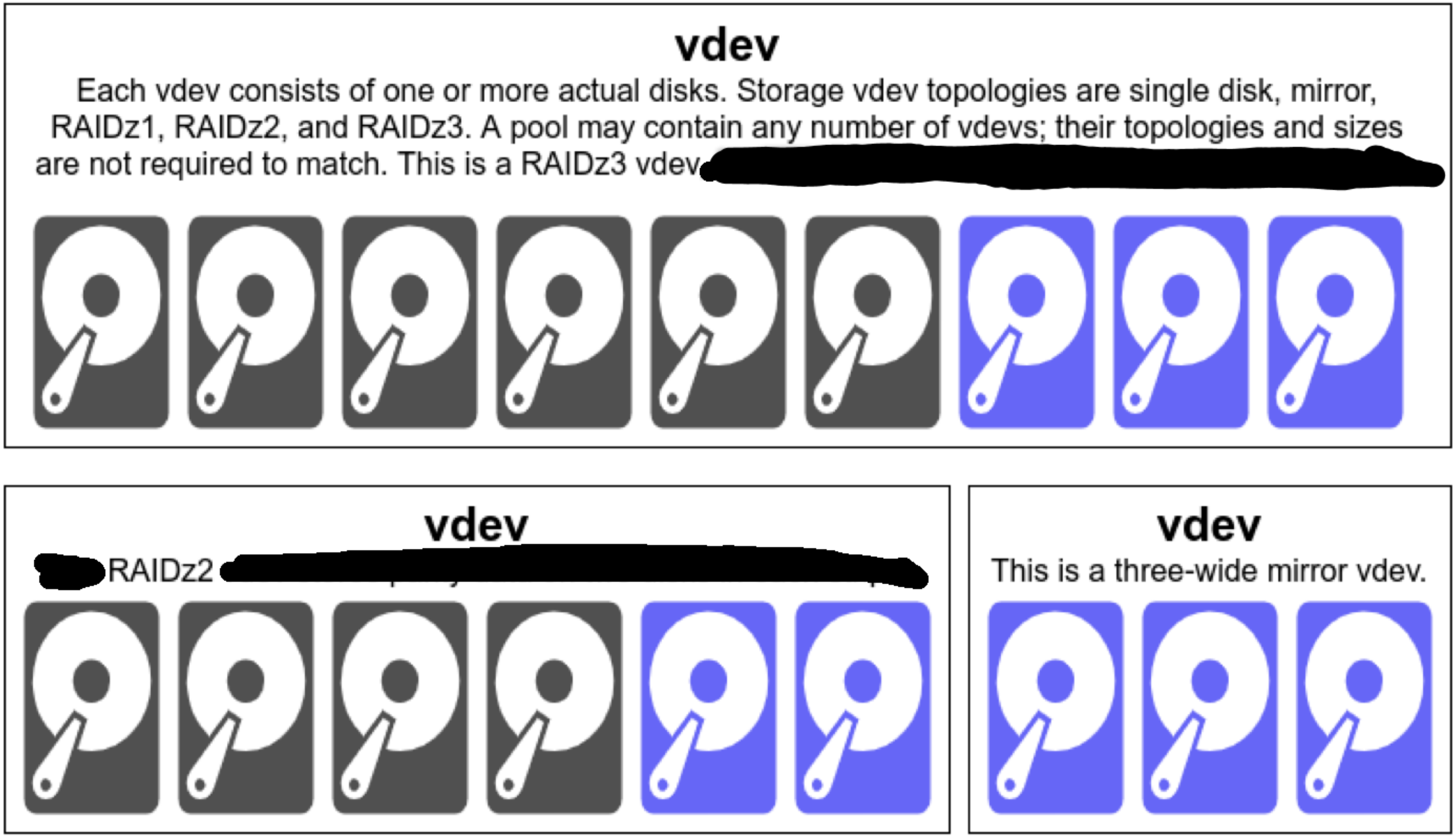
```
$ zpool list
NAME     SIZE  ALLOC   FREE    FRAG    CAP  DEDUP  HEALTH  ALTROOT
zroot  17.9G  8.54G  9.34G     47%    47%  1.00x  ONLINE  -
```

JBOD

zpool

The highest level of the ZFS storage hierarchy is the zpool. A zpool consists of one or more vdevs. Data is distributed across the vdevs. There is no fault tolerance at the pool level—only within individual vdevs.

The blue drives indicate how many drives can be lost without losing data.



vdev

Each vdev consists of one or more actual disks. Storage vdev topologies are single disk, mirror, RAIDz1, RAIDz2, and RAIDz3. A pool may contain any number of vdevs; their topologies and sizes are not required to match. This is a RAIDz3 vdev[redacted]

vdev

[redacted] RAIDz2 [redacted]

vdev

This is a three-wide mirror vdev.

Blue does not indicate parity drives or specific drives which can be lost.

filesystems

```
$ zfs list
NAME                      USED  AVAIL  REFER  MOUNTPOINT
zroot                    8.54G  8.78G    19K  none
zroot/ROOT               8.45G  8.78G    19K  none
zroot/ROOT/11.1-RELEASE     1K  8.78G  4.14G  legacy
zroot/ROOT/default       8.45G  8.78G  6.18G  legacy
zroot/tmp                 120K  8.78G   120K  /tmp
zroot/usr                4.33M  8.78G    19K  /usr
zroot/usr/home           4.28M  8.78G  4.26M  /usr/home
zroot/usr/ports            19K  8.78G    19K  /usr/ports
zroot/usr/src              19K  8.78G    19K  /usr/src
zroot/var                76.0M  8.78G    19K  /var
zroot/var/audit            19K  8.78G    19K  /var/audit
zroot/var/crash            19K  8.78G    19K  /var/crash
zroot/var/log            75.9M  8.78G  75.9M  /var/log
zroot/var/mail             34K  8.78G    34K  /var/mail
zroot/var/tmp              82K  8.78G    82K  /var/tmp
$
```

vdev?

- What’s a vdev?
  - a single disk
  - a mirror: two or more disks
  - a **`raidz`**: group of drives in a **`raidz`**

Terms used here

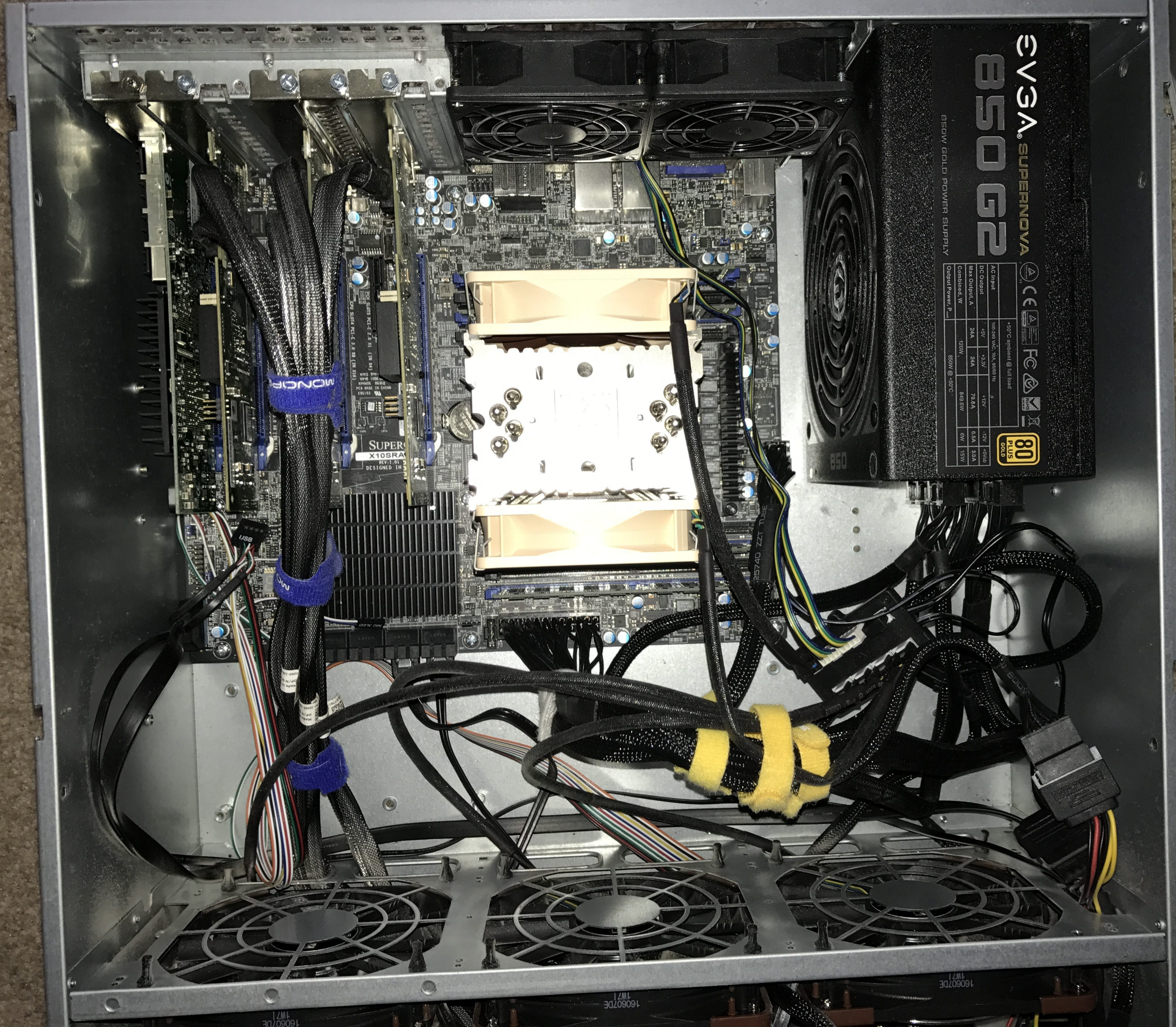
- filesystem ~== dataset

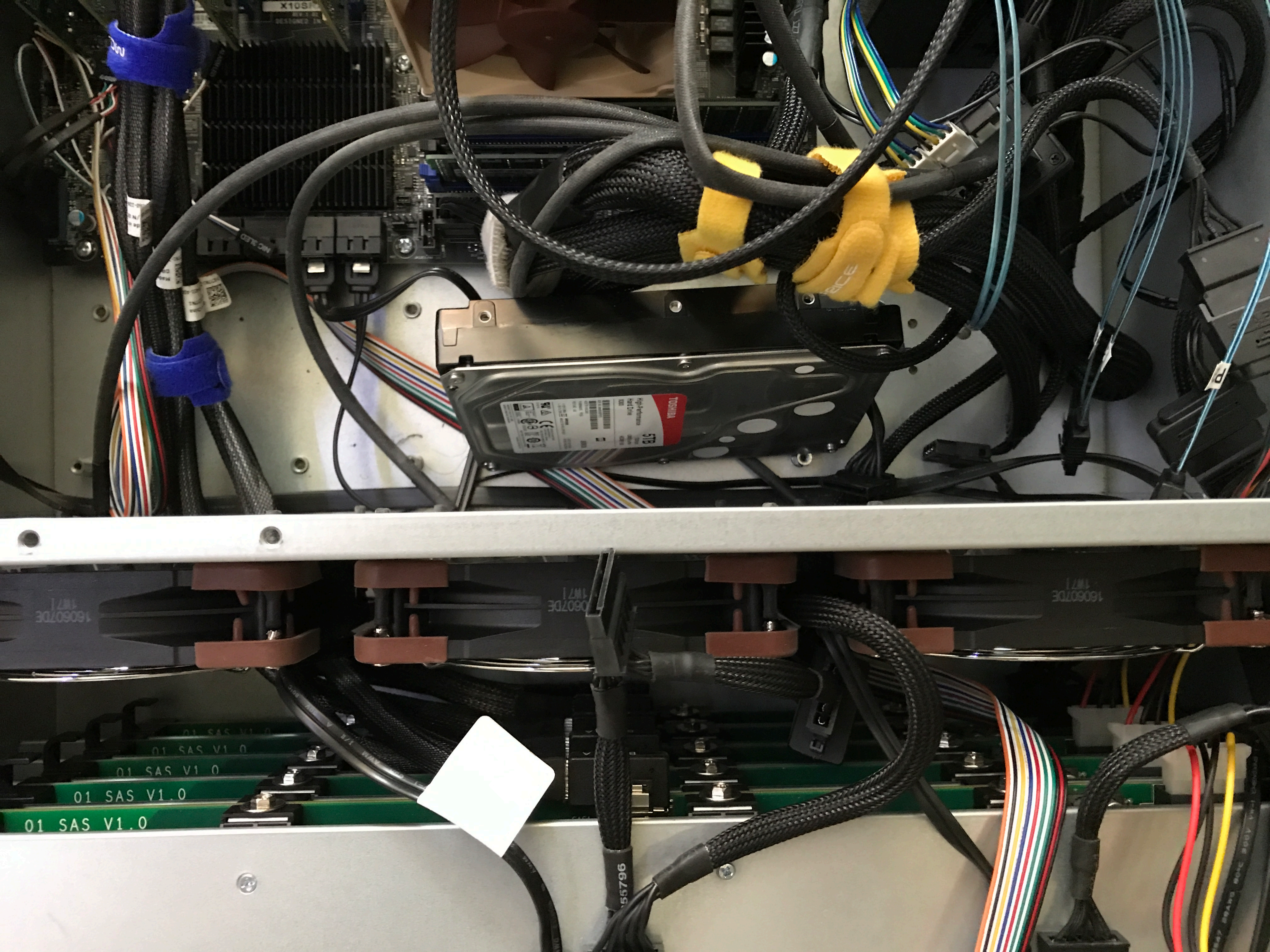
interesting properties

- `compression=lz4`
- `atime=off`
- `exec=no`
- `reservation=10G`
- `quota=5G`

Replacing a failed drive

1. identify the drive
2. add the new drive to the system
3. `zpool replace zroot gpt/disk6 gpt/disk_Z2T4KSTZ6`
4. remove failing drive


EVGA
850 G2
SUPERNOVA
850W GOLD POWER SUPPLY
80 PLUS GOLD
SUPERMICRO
X10SRA


01 SAS V1.0
01 SAS V1.0
01 SAS V1.0
01 SAS V1.0
01 SAS V1.0
TOSHIBA
5TB

Just say NO! to RAID cards

- RAID hides stuff
- The RAID card will try try try to fix it then say, it’s dead
- ZFS loves your drives
- ZFS will try to fix it, and if it fails, will look elsewhere
- Use HBA, not RAID cards

Scalability

- Need more space
- UPGRADE ALL THE DRIVES!
- add a new vdev
- add more disk banks

Data Integrity

- ZFS loves metadata
- hierarchical checksumming of all data and metadata
- ZFS loves checksums & hates errors
- ZFS will tell you about errors
- ZFS will look for errors and correct them if it can

enable scrubs

- there is no fsck on zfs

```
$ grep zfs /etc/periodic.conf
daily_scrub_zfs_enable="YES"
daily_scrub_zfs_default_threshold="7"
```

Mirrors

- two or more drives with duplicate content
- Create 2+ mirrors, stripe over all of them

raidz[1-3]

- four or more drives (min 4 drives for raidz1)
- parity data
- raidzN == can loose any N drives and still be operational
- avoiding lost data is great
- staying operational is also great

simple configurations

- to get you started

disk preparation

```
gpart create -s gpt da0
gpart add -t freebsd-zfs -a 4K -l S3PTNF0JA705A da0
```

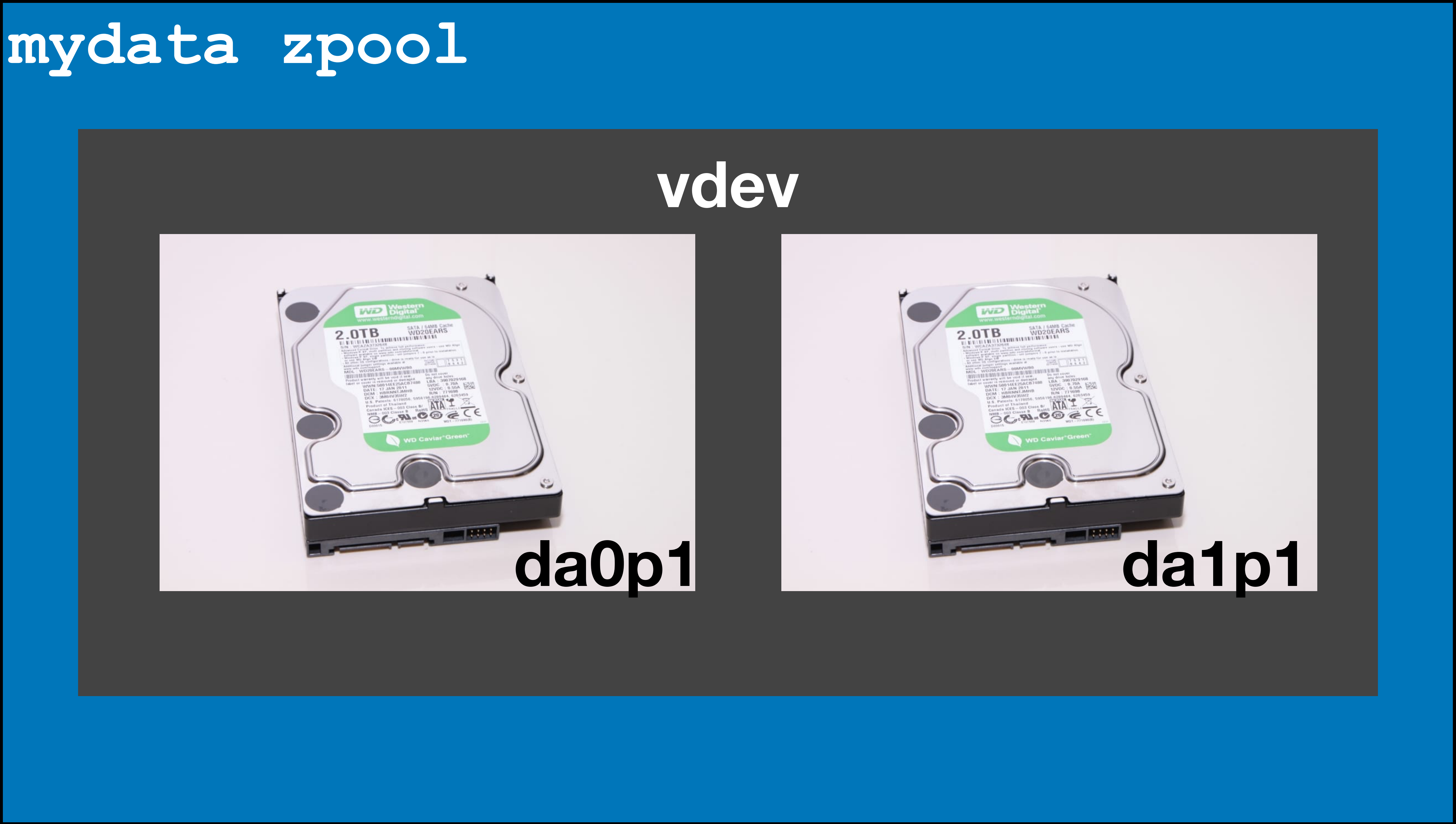
```
$ gpart show da0
=>        40  468862048  da0  GPT  (224G)
          40  468862048    1  freebsd-zfs  (224G)
```

standard partitions

```
root@mfsbsd:~ # gpart show
=>         40  488397088  ada0  GPT  (233G)
           40       1024     1  freebsd-boot  (512K)
         1064        984        - free -  (492K)
         2048   41943040     2  freebsd-swap  (20G)
     41945088  446451712     3  freebsd-zfs  (213G)
    488396800        328        - free -  (164K)
```

- For FreeBSD boot drives
- partition sizes vary

mirror

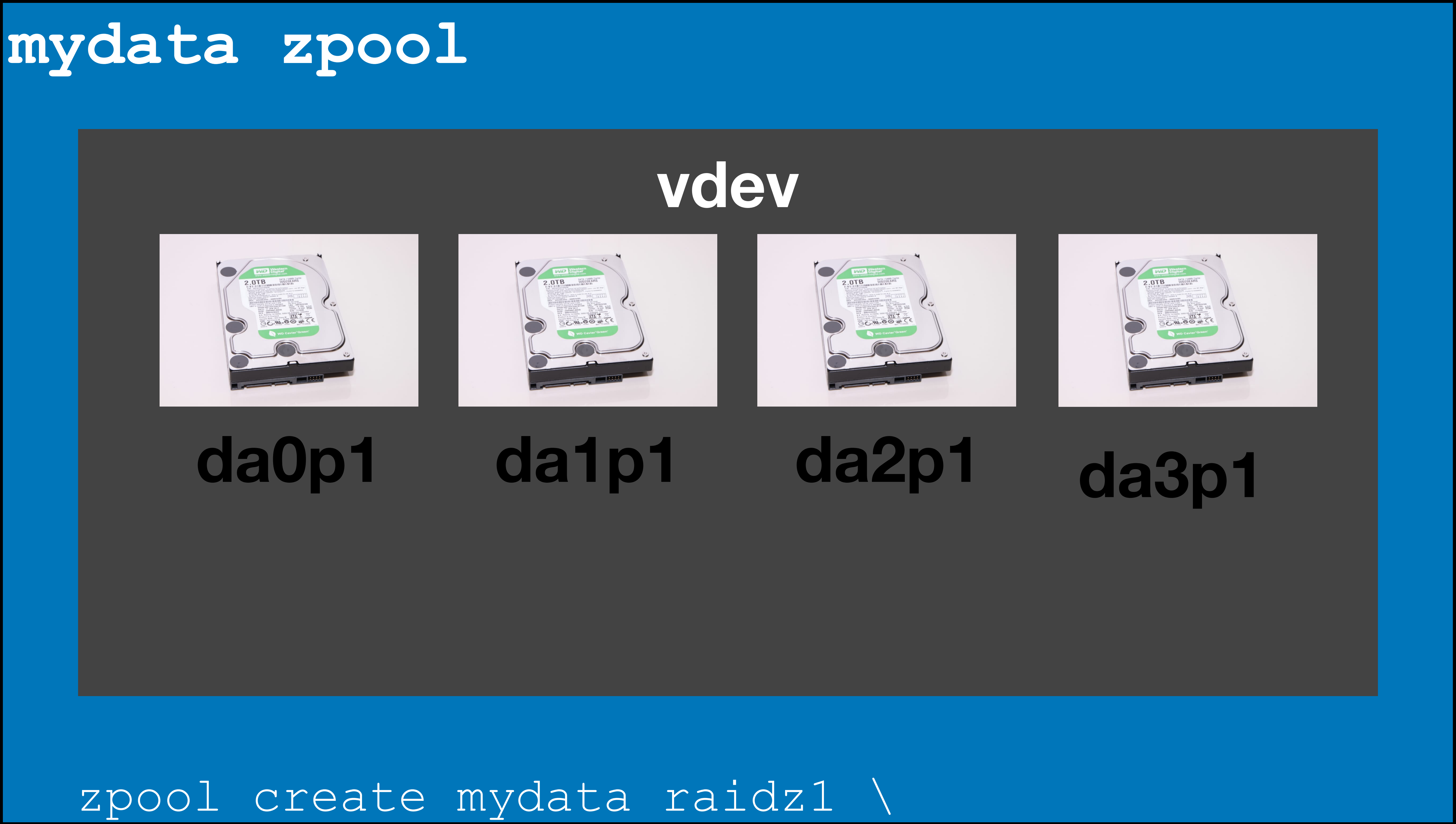


```
zpool create mydata mirror da0p1 da1p1
```

zpool status

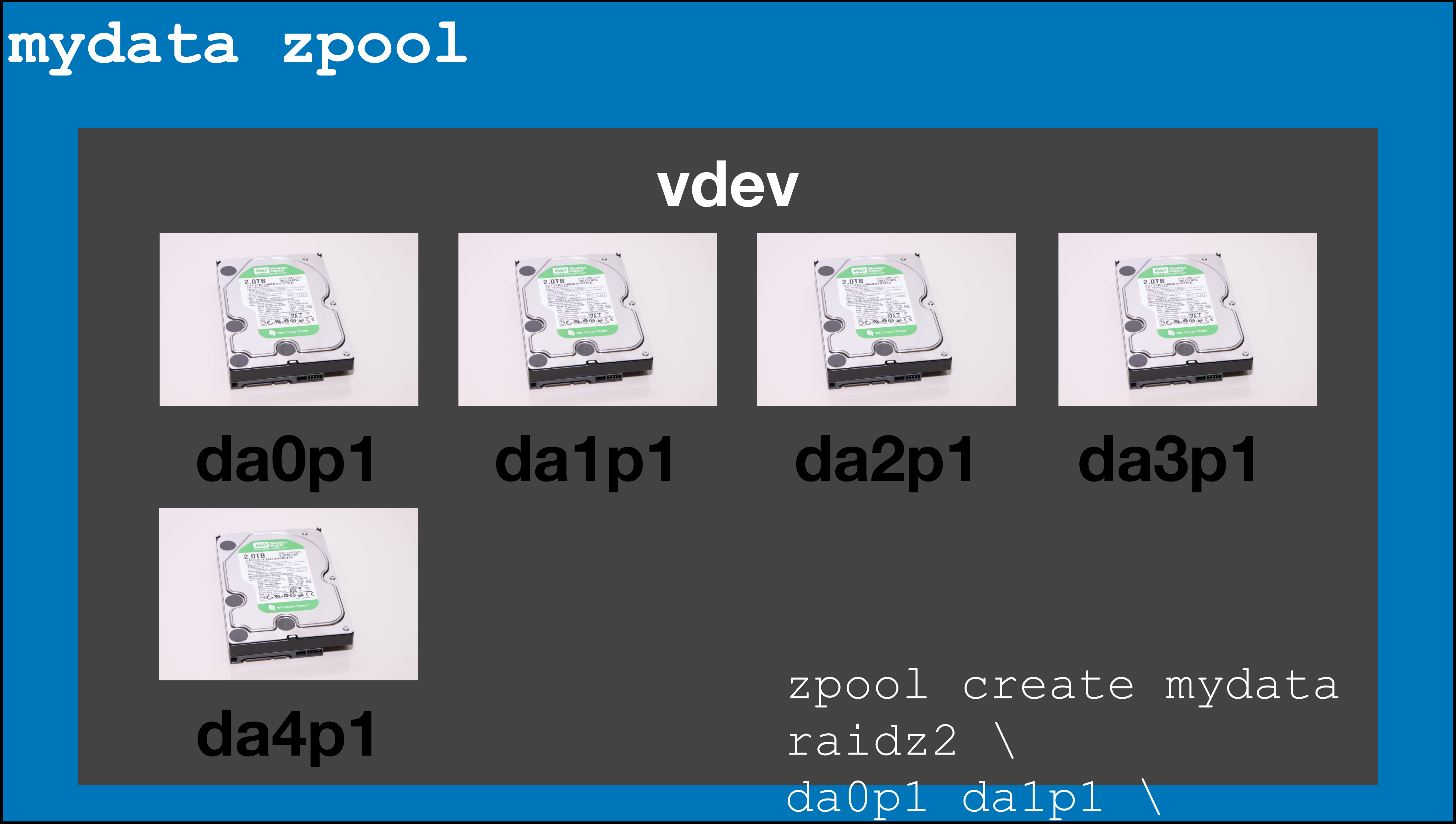
```
$ zpool status mydata
  pool: data
 state: ONLINE
  scan: scrub repaired 0 in 0 days 00:07:03
with 0 errors on Tue Aug 13 03:54:42 2019
config:

	NAME          STATE     READ WRITE CKSUM
	nvd           ONLINE       0     0     0
	  mirror-0    ONLINE       0     0     0
	    da0p1     ONLINE       0     0     0
	    da1p1     ONLINE       0     0     0

errors: No known data errors
```

raidz1



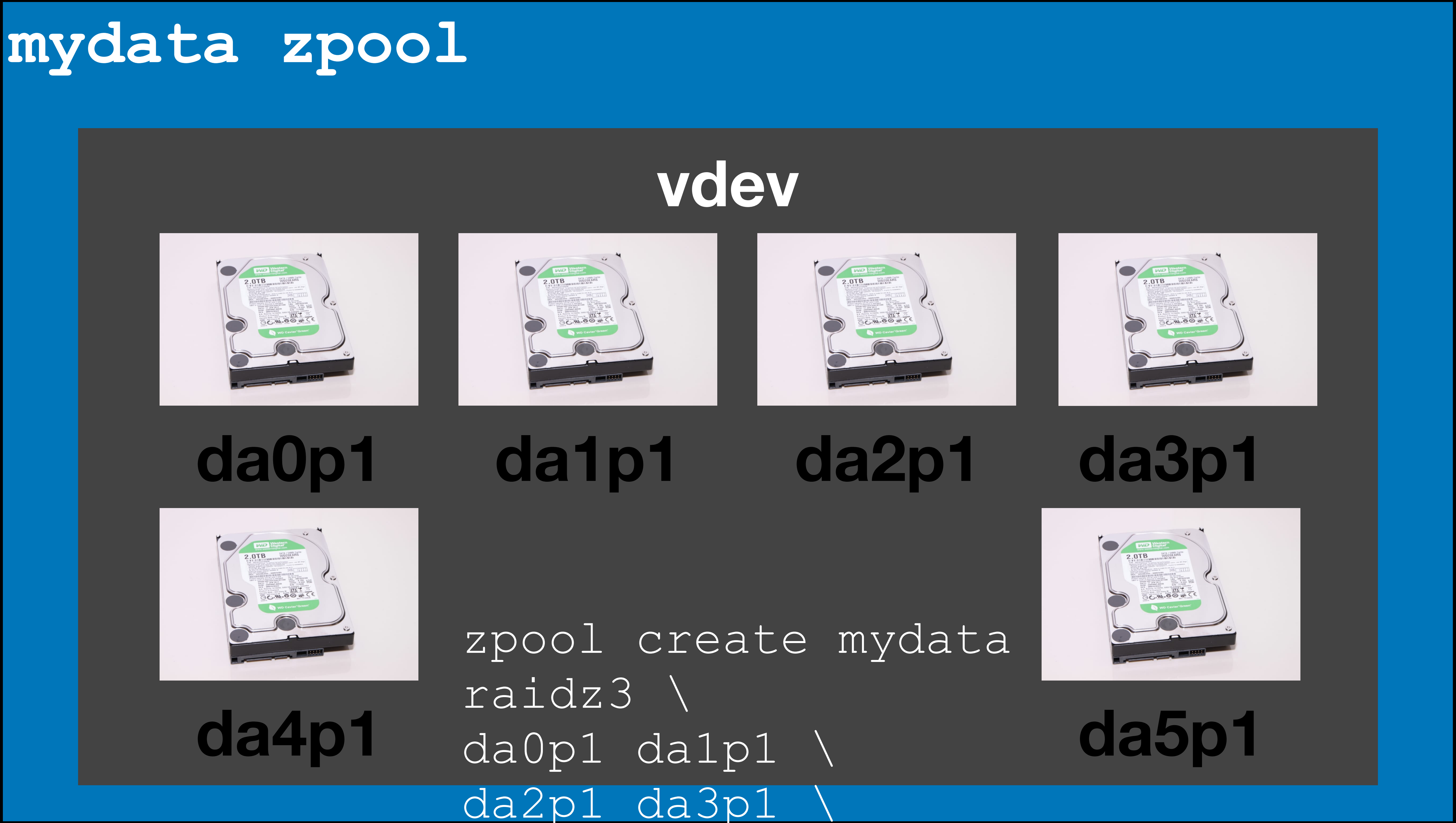
```
zpool create mydata raidz1 \
da0p1 da1p1 \
da2p1 da3p1
```

raidz2



```
zpool create mydata
raidz2 \
da0p1 da1p1 \
da2p1 da3p1 \
da4p1
```

raidz3

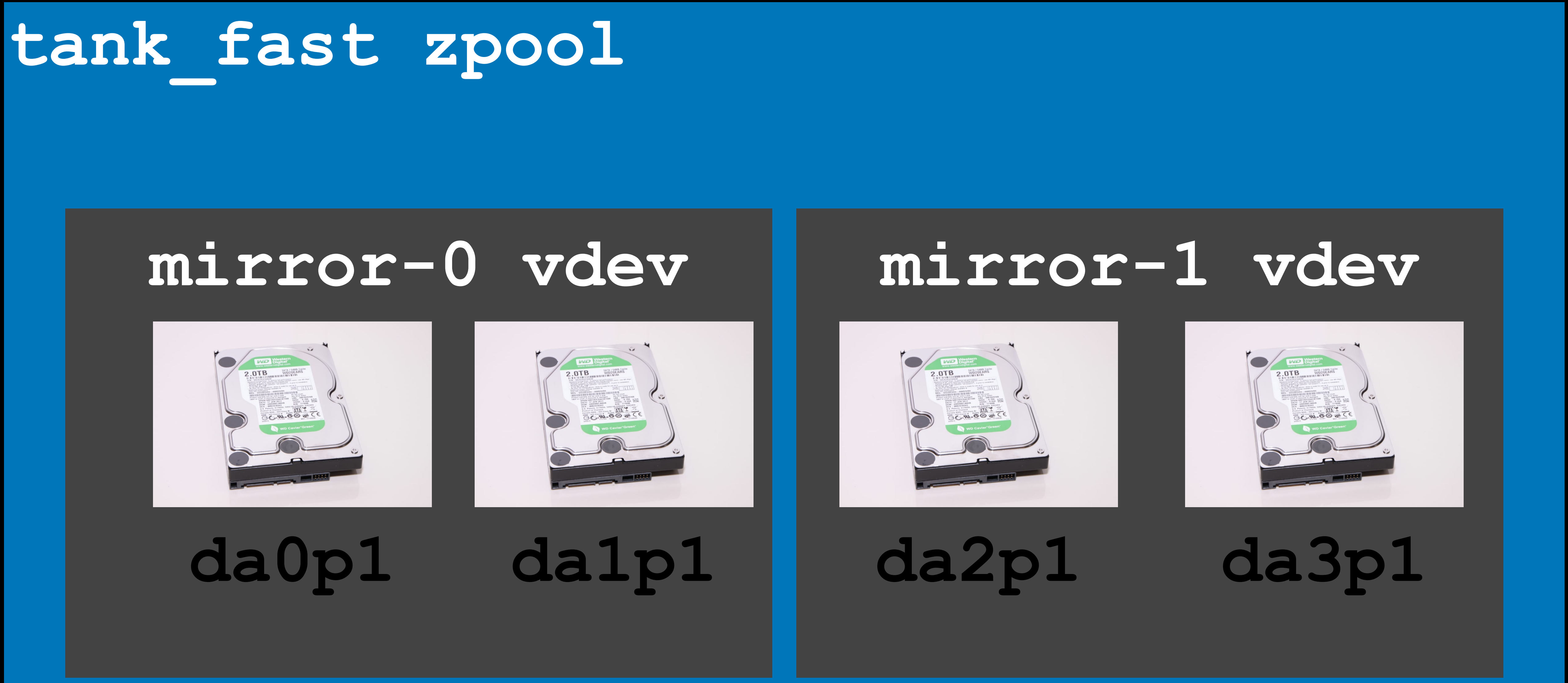


```
zpool create mydata
raidz3 \
da0p1 da1p1 \
da2p1 da3p1 \
da4p1 da5p1
```

zpool status

```
$ zpool status system
  pool: system
 state: ONLINE
  scan: scrub repaired 0 in 0 days 03:01:47 with 0 
errors on Tue Aug 13 06:50:10 2019
config:

	NAME                     STATE     READ WRITE CKSUM
	system                   ONLINE       0     0     0
	  raidz2-0               ONLINE       0     0     0
	    da3p3                ONLINE       0     0     0
	    da1p3                ONLINE       0     0     0
	    da6p3                ONLINE       0     0     0
	    gpt/57NGK1Z9F57D     ONLINE       0     0     0
	    da2p3                ONLINE       0     0     0
	    da5p3                ONLINE       0     0     0

errors: No known data errors
```

raid10



```
zpool create tank_fast \
mirror da0p1 da1p1 \
mirror da2p1 da3p1
```

zpool status

```
$ zpool status tank_fast
  pool: tank_fast
 state: ONLINE
  scan: scrub repaired 0 in 0 days 00:09:10 with 0 
errors on Mon Aug 12 03:14:48 2019
config:

  NAME          STATE     READ WRITE CKSUM
  tank_fast     ONLINE       0     0     0
    mirror-0    ONLINE       0     0     0
      da0p1     ONLINE       0     0     0
      da1p1     ONLINE       0     0     0
    mirror-1    ONLINE       0     0     0
      da2p1     ONLINE       0     0     0
      da3p1     ONLINE       0     0     0

errors: No known data errors
```

so what?

mounting in mounts

- Bunch of slow disks for the main system
- Fast SSD for special use
- create `zpool` on SSD
- mount them in `/var/db/postgres`

```
# zfs list zroot data01/pg02/postgres
NAME                      USED  AVAIL  REFER  MOUNTPOINT
data01/pg02/postgres      450G   641G   271G  /var/db/postgres
zroot                    33.1G  37.1G    88K  /zroot
```

beadm / bectl

- manage BE - boot environments
- save your current BE
- upgrade it
- reboot
- All OK? Great!
- Not OK, reboot & choose BE via bootloader

see also nextboot

- specify an alternate kernel for the next reboot
- Great for trying things out
- automatically reverts to its previous configuration

Quotas

- property on a dataset
- limit on space used
- includes descendants
- includes snapshots
- see also:
  - **`reservation`** - includes descendents, such as snapshots and clones
  - **`refreservation`** - EXCLUDES descendents

Monitoring ZFS

- **`scrub`**
- Nagios monitoring of **`scrub`**
- **`zpool`** status
- **`quota`**
- **`zpool`** capacity

semi-myth busting

single drive ZFS

- single drive ZFS > no ZFS at all

ECC RAM not required

- ZFS without ECC > no ZFS at all

High-end hardware

- Most of my drives are consumer grade drives
- HBA are about $100 off ebay
- Yes, I have some SuperMicro chassises
- Look at FreeNAS community for suggestions

LOADS OF RAM!

- I have ZFS systems running with 1GB of RAM
- runs with 250M free
- That’s the Digital Ocean droplet used in previous examples

Myths end here

Things to do

Snapshots

- read-only
- immutable : cannot be modified
- therefore: FANTASTIC for backups - by that I mean files are in a consistent state
- snapshots on the **same host** are not backups

Sending snapshots

- share your snapshots
- send them to another host
- send them to another data center
- snapshots on another host ARE backups

Use snapshots for clones

- snapshot your database at rest, then clone it
- snapshot a dev environment
- set of files

zfs create all the things!

- got photos? `zfs create`
- got a project? `zfs create`
- Instead of mkdir, think `zfs create`

Other tips

- OS on a ZFS mirror, data on rest
- OS on something else, say UFS, data on rest
- don't boot from HBA

Tips from @Savagedlight

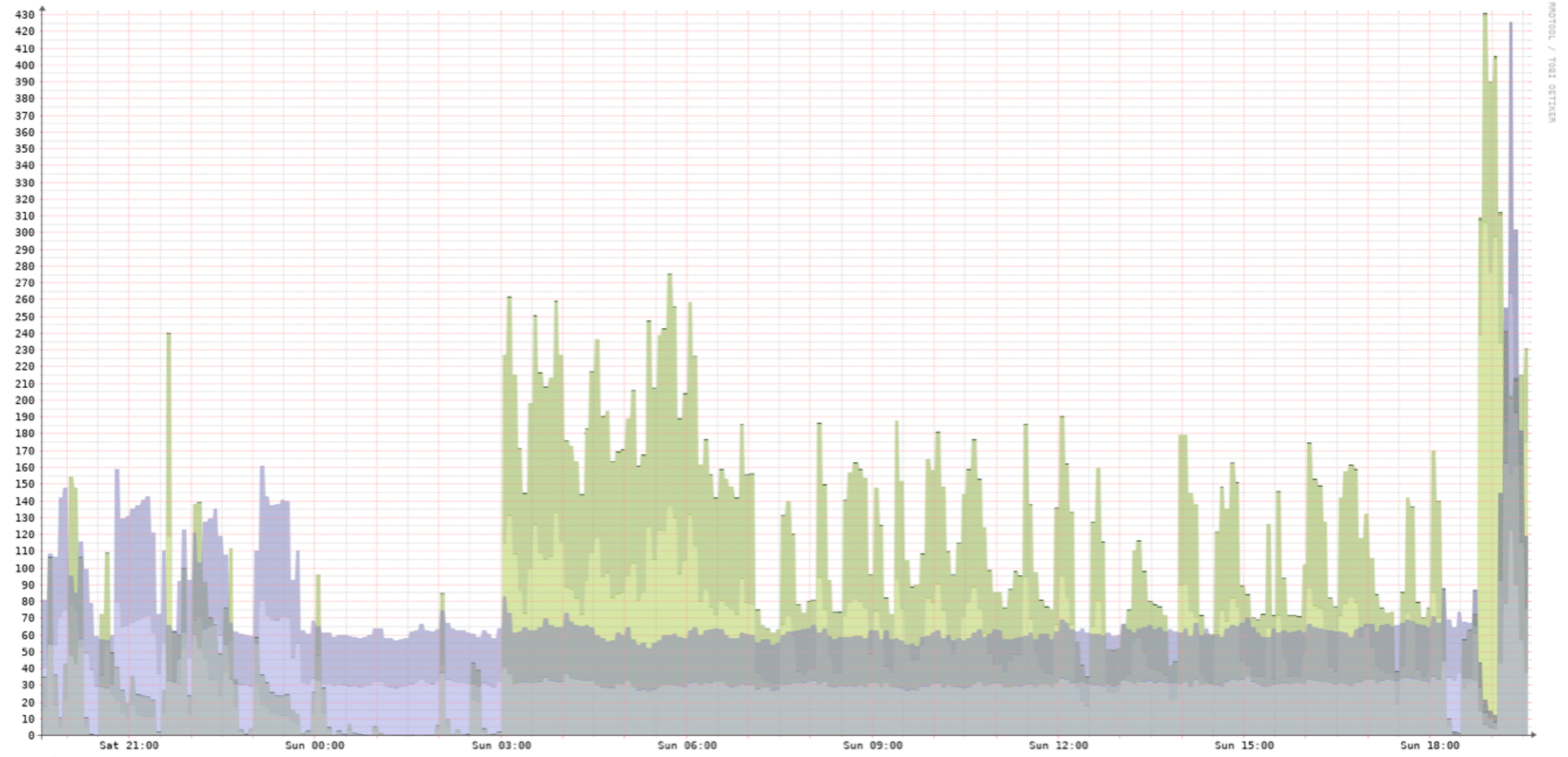
- Tell your BIOS to ignore the HBA. (fewer drives to scan, faster boot)
- You can safely partition the SSD's used in the OS mirror pool so that they can be used for l2arc/cache of the data pool. (Also log device)
- Lots of large files on a dataset? **`recordsize=1m`**

What we covered

- lots of amazing stuff, see original slide

From 2019-07-20 15:35 To 2019-07-21 15:35 Update

Hide Legend | Show Previous | Show RRD Command



```
Operations/sec          Now       Avg      Max
□ ada0      In      174.86     55.36   306.28
□           Out      37.11     35.37   122.34
□ ada1      In       55.12     49.79   138.25
□           Out      38.48     35.53   141.98
□ ada2      In     982.80m    52.75m     2.26
□           Out      43.53      2.01   161.52
□ ada3      In        0.00     8.90m  822.06m
□           Out       0.00  243.06u    20.08m
□ pass0     In      67.72m    71.76m   89.36m
□           Out       0.00      0.00     0.00
□ pass1     In      67.77m    73.99m  138.29m
□           Out       0.00      0.00     0.00
□ pass2     In        0.00      0.00     0.00
□           Out       0.00      0.00     0.00
□ pass3     In        0.00      0.00     0.00
□           Out       0.00      0.00     0.00

□ Total     In        0.00    839.91    3.45k    9.10MB
□           Out       0.00    581.28    3.41k    6.30MB
□           Agg       0.00     1.42k    5.03k   15.40MB
```

Disk activity during ‘zfs replace’ on a mirror