

# Disruptive Storage Workshop

## Hands-On ZFS

Mark Miller

<http://www.pinedalab.org/disruptive-storage-workshop/>



JOHNS HOPKINS  
BLOOMBERG SCHOOL  
of PUBLIC HEALTH

# Schedule

- Setting up our VM (15 minutes)
- Installing ZFS (15 minutes)
- ZFS Configuration and Administration (60 minutes)



# Setting up our VMs



# Prerequisites

- ssh client - ssh/Putty/Mobaxterm
- VirtualBox
- Centos VM from either Globus endpoint: `jhpce#dsw` or <http://www.pinedalab.org/img/Centos6.6-Base.ova>
- 20GB of free disk space
- Fairly fast 64 bit CPU
- 4 GB of RAM



# Setup VirtualBox and Import VM

- Create "Host Only Network" in Virtual Box
  - Navigate to "Virtual Box" -> "Preferences" -> "Network" -> "Host Only Network" tab
  - If there is no Adapter ("vboxnet0" on MacOS) create one by clicking on the green "+"
  - Double click on "vboxnet0" and make sure that the IP address is "192.168.56.1" and netmask is 255.255.255.0
- In VirtualBox, Go to "File" -> "Import Appliance".
  - Navigate to location of "Centos6.6-Base.ova" file, and import with defaults
  - \*\* NOTE: Do not start the VM



# Create VM clone for ZFS

- Right-click on “Centos 6.6 Base” VM and select “Clone”
- Name clone “ZFS1”
- Make sure you check “Reinitialize the MAC Address”
- Select “Full Clone”
- Once the ZFS1 VM is ready, start the VM.



# ZFS1 setup

- Login on console: root/password
- cd /software
- Run “./demo-setup zfs1”
- Reboot
- Notes about console:
  - To escape, hit either Left-⌘ or Right-CTRL
  - Page Up - <Shift>-<fn>-<up-arrow>
  - Page Down - <Shift>-<fn>-<down-arrow>



# Login via ssh

May find it easier to use a friendlier shell interface than the console. You can login to the VM by bringing up an ssh client (from a Terminal on Macs, or with Putty or MobaXterm on a PC), and then connecting to 192.168.56.11 and logging in as root:

```
$ ssh root@192.168.56.11
```





# Installing ZFS



# Install ZFS software

All commands that we will be doing for the lab (including the ones below) are in the file /software/STEPS.zfs

```
# yum localinstall --nogpgcheck \  
  https://download.fedoraproject.org/pub/epel/6/x86\_64/epel-release-6-8.noarch.rpm  
  
# yum localinstall --nogpgcheck \  
  http://archive.zfsonlinux.org/epel/zfs-release.el6.noarch.rpm  
  
# yum install kernel-devel-2.6.32-504.el6.x86_64 kernel-headers-2.6.32-504.el6.x86_64  
# yum install glibc-devel-2.12-1.149.el6.x86_64 glibc-headers-2.12-1.149.el6.x86_64  
  
# yum install -y zfs-0.6.3
```

If you do not have network access, these RPMs are included in the VM

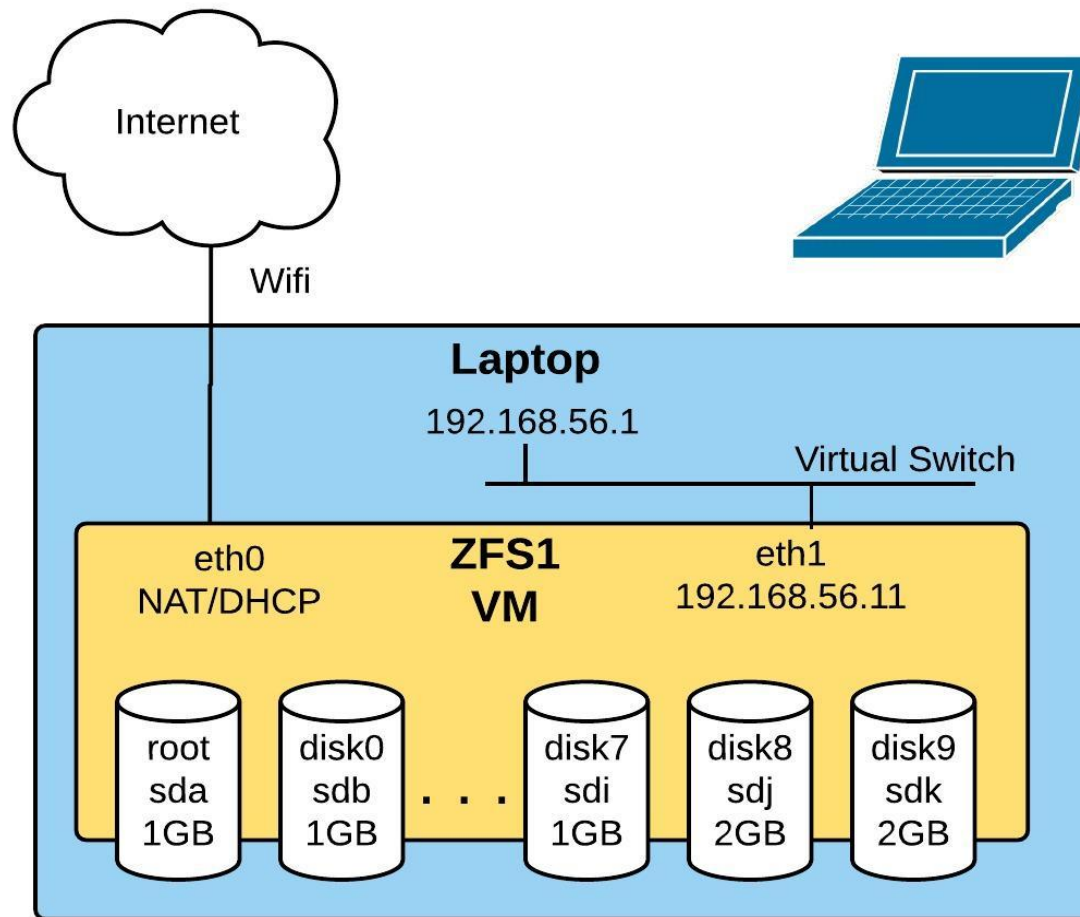
```
# cd /software/zfs/prereqs  
# yum --disablerepo=* install *  
# cd /software/zfs  
# yum --disablerepo=* install *
```

DKMS - Dynamic Kernel Module Support - Kernel modules get compiled for each kernel version

ZFS is now installed and running...



# Layout of our Virtualbox VM



- Added 8 x 1GB Virtual Drives and 2 x 2GB Virtual Drives (Dynamically Allocated)
- Added second Network Interface of type "Host Only"



# Notes about VM

- Standard Centos 6.6 image
- Enabled Centos 6.6 Vault access for yum.
- Created /software directory for lab scripts and staged packages
- chkconfig iptables off, selinux disabled
- Modified /etc/grub to show boot messages (removed “quiet” and “rhgb”)
- set the root password set to “passwd”
- installed "parted", "rsync", and "man" packages via “yum install”
- Added /etc/hosts entries



# Differences between VM and real world

- No card/driver issues to deal with
  - No cabling
  - No faulty hardware
  - Much faster reboots
  - Not really worried about I/O performance
  - Provisioning work already done
- 
- \*\* Please make sure you are not logged into VMs and production systems \*\*



# Last bits of setup

Create handy disk aliases with `vdev_id`. The “`vdev_id`” is a

udev helper (`/lib/udev/rules.d/69-vdev.rules`) that comes

bundled with `zfs`.

```
# cp /software/vdev_id.conf /etc/zfs
```

```
# udevadm trigger (or reboot)
```

```
# ls -al /dev/disk/by-vdev
```

```
# fdisk -l /dev/disk/by-vdev/* | grep /dev/disk/by-vdev
```



# Last last bit of setup

Need to Label disks:

```
# for DISK in /dev/disk/by-vdev/disk*  
do  
    parted $DISK mklabel gpt  
done
```



# ZFS Configuration and Administration





# RAID Levels for pools

## ZFS Raid Levels:

- striped (RAID 0) - this is the default if no pool type is given
- mirror (RAID 1)
- striped-mirrored (RAID10)
- raidz (RAID5 - uses XOR for parity)
- raidz2 (RAID6 - uses XOR for one parity and Reed-Solomon code for the other parity)
- raidz3



# Our First ZPOOL - mirror

To create our first ZFS pool, and ZFS filesystem:

```
# zpool create pool1 mirror disk0 disk1
```

```
# zpool list  
# zpool status  
# zfs list  
# df -h
```

```
# zfs create pool1/zfs1
```

```
# zpool list  
# zpool status  
# zfs list  
# df -h
```

```
# cp -R /etc /pool1/zfs1
```

```
? Why do we create zfs filesystems  
   zpools are permanent, where zfs filesystems are  
flexible
```



# Our Second ZPOOL - RAIDZ

To create our Second ZFS pool, and ZFS filesystem:

```
# zpool create pool2 raidz disk2 disk3 disk4 disk5  
# zfs create pool2/alice  
# zfs create pool2/bob
```

```
# zpool list  
# zpool status  
# zfs list  
# df -h
```

To Note that "zpool list" on a "raidz" pool will show raw space without taking into account parity.



# Destroying a ZPOOL/ZFS

As with unmounting a standard filesystem, a ZFS filesystem must not be in use for it to be deleted.

```
# zfs list  
# df -h
```

```
# cd /pool2/alice  
# zfs destroy pool2/alice  
# cd /  
# zfs destroy pool2/alice
```

```
# zfs list  
# df -h
```

```
# zpool destroy pool2
```

```
# zpool list  
# zpool status  
# zfs list  
# df -h
```



# Growing a ZPOOL by adding stripes

You can add an additional similar VDEVs

```
# zpool add -n pool1 mirror disk2 disk3
# zpool add pool1 mirror disk2 disk3
# zpool status
# zpool list
# zfs list
# df -h
```

We've just created a RAID10! The same structure could have been done at the beginning with:

```
zpool create pool1 mirror disk0 disk1 mirror disk2 disk3
```

Additional VDEVs that get added to a zpool will be treated as additional stripes. New writes will favor VDEVs with lower usage, so this may affect performance.



# Growing a zpool with bigger disks

Replacing 1GB disks with 2GB disks.

```
# zpool set autoexpand=on pool1
# zpool status; zpool list -v; df -h

# zpool replace pool1 disk2 disk8
# zpool status; zpool list -v; df -h

# zpool replace pool1 disk3 disk9
# zpool status; zpool list -v; df -h

# zpool replace pool1 disk8 disk2
cannot replace disk8 with disk2: device is
too small
```

<http://blog.ociru.net/2013/09/25/let-your-zfs-extend>



# Adding a spare to a ZPOOL

Spares are used to replace a failed drive in a zpool

```
# zpool add pool1 spare disk4  
# zpool status  
# zpool list  
# zfs list
```

**NOTE** - Spares do not automatically kick in in the event of a disk failure on ZoL.



# Adding an SLOG device to a ZPOOL to be used as a ZIL

Separate Intent Logs (SLOGs) are used to speed up **writes** to the ZFS filesystem. They are not a write cache, but enable metadata to be written quickly so that response can be given. ZILs are never read from unless a recovery is needed. Typically these are very fast SSD disks.

```
# zpool add pool1 log disk5
# zpool status
# zpool list
# zfs list
```

Log devices should be mirrored.

```
# zpool remove pool1 disk5
# zpool add pool1 log mirror disk5 disk6
```

<http://www.zfsbuild.com/2010/06/03/howto-zfs-add-log-drives-zil/>

<https://forums.freenas.org/index.php?threads/some-insights-into-slog-zil-with-zfs-on-freenas.13633/>





# Adding a cache device (L2ARC) to a ZPOOL

Cache devices are used to speed up **reads** to the ZFS filesystem. Typically these are very fast SSD disks.

```
# zpool add pool1 cache disk7  
# zpool status  
# zpool list  
# zfs list
```

<https://blogs.oracle.com/brendan/entry/test>

<http://queue.acm.org/detail.cfm?id=1809426>



# Removing devices from a ZPOOL

Only hot spares, cache (L2ARC), and log (zil) devices can be removed from a pool.

```
# zpool status
# zpool remove pool1 disk7
# zpool status
# zpool remove pool1 disk6
# man zpool
# zpool remove pool1 mirror-2
# zpool status
```

<https://blogs.oracle.com/brendan/entry/test>

<http://queue.acm.org/detail.cfm?id=1809426>



# Scrubbing a ZPOOL

Each zpool should be scrubbed on a regular basis to identify potential disk errors.

```
# zpool status  
# zpool scrub pool1  
# zpool status
```

## Resilvering

Scrubbing and resilvering are very similar operations. The difference is that resilvering only examines data that ZFS knows to be out of date (for example, when replacing an existing device), whereas scrubbing examines all data to discover silent errors due to hardware faults or disk failure.



# Zpool import and export

Typically used when moving pools between dual-connected servers

```
# zpool export pool1
```

Note - just like unmount, this will not work if the pool or ZFS filesystem is in use

```
# zpool import  
# zpool import pool1
```

To import a zpool that has become compromised

```
# zpool import -F pool1
```

To forcibly import a zpool that will not import

```
# zpool import -f pool1
```



# Controlling what gets imported at boot

By default, when you reboot, any pools that are imported will get imported upon reboot. This is controlled by the `/etc/zfs/zpool.cache` file.

```
# strings /etc/zfs/zpool.cache | grep pool
# mv /etc/zfs/zpool.cache /etc/zfs/zpool.cache.old
# reboot
```

To rebuild the `zpool.cache` files, run

```
# zpool set cachefile=/etc/zfs/zpool.cache pool1
```

This can be run against any pool.



# Setting attributes - ZPOOL

```
# zpool get all pool1
```

Notable attributes that can be set on the pool:

- autoexpand
- ashift
- readonly

Some settable attributes can be set in real time.

Some settable attributes can be set when the pool is imported.  
One attribute can only be set at create time - ashift

```
# zpool set autoexpand=on pool1  
# zpool import -o readonly=on pool1  
# zpool create -o ashift=12 pool1
```



# ZPOOL Setting - ashift

The `ashift` setting is used to set the sector size for I/O alignment. `ashift=9` for older disks with 512b sectors and `ashift=12` for newer drives with 4k sectors. Improves write performance, but causes more disk space overhead.

This can only be set at create time:

```
# zpool create -o ashift=12 pool1
```

<http://louwrentius.com/zfs-performance-and-capacity-impact-of-ashift9-on-4k-sector-drives.html>  
<http://lists.freebsd.org/pipermail/freebsd-fs/2013-May/017337.html>



# Setting attributes - ZFS

```
# zfs get all pool1/zfs1
```

Notable attributes that can be set on the pool:

- mountpoint
- quota
- reservation
- compression
- readonly
- sharenfs

Most attributes can be set in real time or at "zfs create" time

```
# zpool create pool2 raidz disk2 disk3 disk6 disk7
# zfs create pool2/chuck
# zfs create pool2/dave

# zfs set mountpoint=/chuck pool2/chuck
# df -h
# zfs create -o mountpoint=/eric pool2/eric
```





# ZFS attributes - Quotas/Reservations

- Quotas are used on a ZFS filesystem to limit the size of a filesystem.

```
# zfs set quota=500M pool2/chuck  
# df -h  
# zfs set quota=none pool2/chuck  
# df -h
```

- Reservations are used on a ZFS filesystem to reserve a portion of the ZPOOL for the ZFS filesystem.

```
# zfs set reservation=1G pool2/dave  
# df -h
```

- Reservations and quotas can be combined.

```
# zfs set quota=1G pool2/dave  
# df -h
```



# Order of setting reservation and quota

Quota must always be larger than (or equal to) reservation

- To increase, set quota first, then reservation
- To decrease, set reservation, then quota

```
# zfs set quota=1G pool2/chuck  
# zfs set reservation=1G pool2/chuck  
# df -h  
# zfs set reservation=800M pool2/chuck  
# zfs set quota=800M pool2/chuck
```



# ZFS attributes - Compression

Compression can be enabled on ZFS filesystems on the fly.

```
# df -h /chuck
# dd if=/dev/zero of=/chuck/zeros01 bs=1M count=100
# df -h /chuck      (shows 100 MBs used)

# zfs set compression=on pool2/chuck
# dd if=/dev/zero of=/chuck/zeros02 bs=1M count=100
# df -h /chuck      (shows ~101 MB used)
# ls -la /chuck
# du -sh /chuck/*
# du -sh --apparent-size /chuck/*

# dd if=/dev/urandom of=/chuck/random01 bs=1M count=100
# df -h              (shows ~201 MB used)

# zfs set compression=off pool2/chuck
# dd if=/dev/zero of=/chuck/zeros03 bs=1M count=100
# df -h              (shows ~301 MB used)
```

If compression is set, any data previously written will remain uncompressed - only new data will be compressed.



# ZFS Snapshots

## Creating Snapshots

```
# zfs snap pool2/chuck@snap1
# zfs list -t snapshot
# zfs list -t all
```

```
# dd if=/dev/zero of=/chuck/zeros04 bs=1M count=100
# ls -al /chuck
# ls -la /chuck/.zfs/
# ls -la /chuck/.zfs/snapshot
# zfs snap pool2/chuck@snap2
# ls -la /chuck/.zfs/snapshot
```

## Hidden Snapshots?

```
# ls -al /chuck
# zfs set snapdir=visible pool2/chuck
# ls -al /chuck
```



# Benefits of using ZFS Snapshots

## Easy File Recovery

```
# cd /chuck
# ls -al
# df -h . ; zfs list -t snapshot
# rm random01
# ls -al
# df -h . ; zfs list -t snapshot
# cp .zfs/snapshot/snap2/random01 .
```

## But snapshots take up space

```
# rm zeros3
# df -h . ; zfs list -t snapshot
# zfs destroy pool2/chuck@snap2
# df -h . ; zfs list -t snapshot
```



# ZFS Snapshot Rollback

An entire ZFS filesystem can be resorted back to a snapshot view with the "zfs rollback" command.

```
# touch /chuck/file1
# touch /chuck/file2
# zfs snap pool2/chuck@snap3
# touch /chuck/file3
# rm /chuck/file1
# ls -al /chuck (shows file1 file is gone)
# zfs rollback pool2/chuck@snap3
# ls -al /chuck
```

... shows file1 is back, but file3 is gone because we rolled back the filesystem to what it looked like at when "snap3" was taken



# ZFS Send/Receive

ZFS send and ZFS recv are used to send a copy of a ZFS snapshot from 1 pool to another

```
# zfs list -t all
# zfs send pool2/chuck@snap1 | zfs recv pool1/zfs1-backup
# zfs list -t all
# ls -al /pool1/zfs1-backup
```



# ZFS Send & Receive (remote)

ZFS Send & Receive can also be used to send snapshots over the network to another ZFS server. We cannot do this now, as we only have 1 server, but will do it tomorrow

```
[root@zfs1 zfs]# zfs list -t all
[root@zfs1 zfs]# zfs send pool2/chuck@snap1 | ssh host2 zfs
recv pool2/zfs1-backup
```

```
[root@host2 zfs]# zfs list -t all
```

NAME	USED	AVAIL	REFER	MOUNTPOINT
pool1	100M	1.86G	31K	/pool1
pool1/zfs1	100M	1.86G	100M	/pool1/zfs1
pool1/zfs1@snap1	0	-	100M	-
pool2	100M	2.80G	41.9K	/pool2
pool2/zfs1-backup	100M	2.80G	100M	/pool2/zfs1-backup
pool2/zfs1-backup@snap1	0	-	100M	-

```
[root@host2 zfs]# ls -al /pool2/zfs1-backup
```

total		102363						
drwxr-xr-x	2	root	root	3	Sep	8	16:41	.
drwxr-xr-x	3	root	root	3	Sep	9	11:24	..
-rw-r--r--	1	root	root	104857600	Sep	8	16:36	file2





# ZFS Tuning (condensed to 1 slide...)

General principle is that not much tuning is needed. ZFS experts have already done a lot of tuning.

Realtime changes can be made to files in `/sys/module/zfs/parameters`

```
# cat /sys/module/zfs/parameters/zfs_arc_max  
67719876608  
# echo 42949672960 > /sys/module/zfs/parameters/zfs_arc_max
```

Permanent changes can be set in `/etc/modprobe.d/zfs.conf`

<https://wiki.freebsd.org/ZFSTuningGuide>

[http://www.solarisinternals.com/wiki/index.php/ZFS\\_Evil\\_Tuning\\_Guide](http://www.solarisinternals.com/wiki/index.php/ZFS_Evil_Tuning_Guide)

<http://www.princeton.edu/~unix/Solaris/troubleshoot/zfs.html>

[http://docs.oracle.com/cd/E26502\\_01/html/E29022/chapterzfs-1.html#scrolltoc](http://docs.oracle.com/cd/E26502_01/html/E29022/chapterzfs-1.html#scrolltoc)



# Zpool iostat

The "zpool iostat" command lets you look at disks statistics for your zpool.

```
# zpool iostat
```

```
# zpool iostat -v
```



# ZED

ZED is the ZFS Event Daemon, and can be used to provide notification on events that occur within ZFS

- Need to enable email in `/etc/zfs/zed.d/zed.rc`
- No service is provided by default, so need to run "zed" to start.
- Triggered from "zpool events"
- Can be configured to auto-engage Hot Spares



# Troubleshooting ZFS



# Failure Scenario 1 - Disk Failure

In our VM, no easy way to “yank a disk”.

```
# zpool status  
# init 0
```

In VirtualBox, remove "Disk 1", then start up the VM. Log back into your VM.

```
# zpool replace pool1 disk0 disk4
```

This will cause the vdev to rebuild on the hot spare.

Edit `/etc/zfs/vdev_id.conf` and change "disk0" config to use the path to disk5.

Run "udevadm trigger"

```
# zpool online pool1 disk0  
# zpool replace pool1 disk0
```



# Failure Scenario 2

## Disk corruption - The importance of scrubs and checksums

```
# dd if=/dev/urandom of=/dev/disk/by-vdev/disk0 bs=1M count=10
# zpool status
# find /pool1 -type f -exec cat {} > /dev/null \;
(generate traffic to slow down scrub)
# zpool status

# dd if=/dev/urandom of=/pool1/zfs1/random4 bs=1M count=1000 &
# zpool scrub pool1
# zpool status
# zpool clear pool1
# zpool status
```



# Failure Scenario 3

## Oops - zpool destroyed

```
# zpool destroy pool2
(oops)
# zpool import
(oh no!)
# zpool import -D
(yay)
# zpool import -D pool2
(almost there)
# zpool import -D -f pool2
# zpool status ; df -h
```

<http://docs.oracle.com/cd/E19253-01/819-5461/6n7ht6r0o/index.html>



# Failure Sceanrio 4

## Oops - zfs destroy

May be able to use `zdb`, `zpool history` and `zpool import -N -o readonly=on -f -R /pool -F -T <transaction_id> <pool>`

Could not get this to work on the VMs.

<https://forums.freenas.org/index.php?threads/all-data-gone-accidental-zfs-destroy.18240/>  
<http://lists.freebsd.org/pipermail/freebsd-hackers/2013-July/043125.html>





# Upgrading ZFS

Upgrading ZFS can be done with the following commands:

```
# yum update zfs  
# reboot  
# zpool upgrade <pool>
```

Backups should be done prior to performing an upgrade.



# RWE 1 - Destroying unimportable zpool

```
# zpool import
  pool: pool1
    id: 14828013078418513725
  state: FAULTED
  status: The pool metadata is corrupted.
  action: The pool cannot be imported due to
damaged devices or data.
  see: http://zfsonlinux.org/msg/
ZFS-8000-72
  config:
```

pool1	FAULTED	corrupted data
mirror-1	ONLINE	
disk6	ONLINE	
disk7	ONLINE	
logs		
disk3	ONLINE	

```
pool: pool1
  id: 9521125143068629779
  state: ONLINE
  action: The pool can be imported using its
name or numeric identifier.
  config:
```

pool1	ONLINE
mirror-0	ONLINE
disk8	ONLINE
disk9	ONLINE

```
# zpool import -f 14828013078418513725
cannot import 'pool1': I/O error
  Destroy and re-create the pool from
  a backup source.
```

```
# zpool labelclear -f /dev/disk/by-vdev/disk3-part1
# zpool labelclear -f /dev/disk/by-vdev/disk6-part1
# zpool labelclear -f /dev/disk/by-vdev/disk7-part1
# zpool import
  pool: pool1
    id: 9521125143068629779
  state: ONLINE
  action: The pool can be imported using its name or
numeric identifier.
  config:
```

pool1	ONLINE
mirror-0	ONLINE
disk8	ONLINE
disk9	ONLINE

```
# zpool import pool1
# zpool list
NAME  SIZE  ALLOC  FREE  CAP  DEDUP  HEALTH  ALTROOT
pool1 1.98G 101M  1.89G   4%  1.00x  ONLINE  -
# zpool create pool2 raidz disk2 disk3 disk4 disk5
invalid vdev specification
use '-f' to override the following errors:
/dev/disk/by-vdev/disk3 does not contain an EFI labe
but it may contain partition
information in the MBR.
# zpool create -f pool2 raidz disk2 disk3 disk4 disk
```



# RWE 2 - Firmware issue

```
Jun 29 11:08:34 dcl01-oss01 kernel: mpt2sas4: log_info(0x31080000): originator(PL), code(0x08), sub_code(0x0000)
Jun 29 11:08:34 dcl01-oss01 kernel: sd 11:0:33:0: [sdfn] Result: hostbyte=DID_SOFT_ERROR driverbyte=DRIVER_OK
Jun 29 11:08:34 dcl01-oss01 kernel: sd 11:0:33:0: [sdfn] CDB: Read(10): 28 00 1c c2 c1 a0 00 00 40 00
Jun 29 11:08:34 dcl01-oss01 kernel: end_request: I/O error, dev sdfn, sector 482525600
Jun 29 11:08:34 dcl01-oss01 kernel: mpt2sas4: log_info(0x31080000): originator(PL), code(0x08), sub_code(0x0000)
Jun 29 11:08:34 dcl01-oss01 kernel: sd 11:0:33:0: [sdfn] Result: hostbyte=DID_OK driverbyte=DRIVER_SENSE
Jun 29 11:08:34 dcl01-oss01 kernel: sd 11:0:33:0: [sdfn] Sense Key : Aborted Command [current]
Jun 29 11:08:34 dcl01-oss01 kernel: sd 11:0:33:0: [sdfn] Add. Sense: Information unit iuCRC error detected
Jun 29 11:08:34 dcl01-oss01 kernel: sd 11:0:33:0: [sdfn] CDB: Read(10): 28 00 1c c2 c1 58 00 00 20 00
Jun 29 11:08:34 dcl01-oss01 kernel: end_request: I/O error, dev sdfn, sector 482525528
Jun 29 11:08:34 dcl01-oss01 kernel: sd 11:0:80:0: [sdhg] Result: hostbyte=DID_SOFT_ERROR driverbyte=DRIVER_OK
Jun 29 11:08:34 dcl01-oss01 kernel: sd 11:0:80:0: [sdhg] CDB: Read(10): 28 00 1c c2 c1 c0 00 00 20 00
Jun 29 11:08:34 dcl01-oss01 kernel: end_request: I/O error, dev sdhg, sector 482525632
Jun 29 11:08:34 dcl01-oss01 kernel: sd 11:0:80:0: [sdhg] Result: hostbyte=DID_OK driverbyte=DRIVER_SENSE
Jun 29 11:08:34 dcl01-oss01 kernel: sd 11:0:80:0: [sdhg] Sense Key : Aborted Command [current]
Jun 29 11:08:34 dcl01-oss01 kernel: sd 11:0:80:0: [sdhg] Add. Sense: Information unit iuCRC error detected
Jun 29 11:08:34 dcl01-oss01 kernel: sd 11:0:80:0: [sdhg] CDB: Read(10): 28 00 1c c2 c1 58 00 00 40 00
Jun 29 11:08:34 dcl01-oss01 kernel: end_request: I/O error, dev sdhg, sector 482525528
Jun 29 11:11:15 dcl01-oss01 kernel: sd 9:0:70:0: [sddl] Result: hostbyte=DID_OK driverbyte=DRIVER_SENSE
Jun 29 11:11:15 dcl01-oss01 kernel: sd 9:0:70:0: [sddl] Sense Key : Aborted Command [current]
Jun 29 11:11:15 dcl01-oss01 kernel: sd 9:0:70:0: [sddl] Add. Sense: Information unit iuCRC error detected
Jun 29 11:11:15 dcl01-oss01 kernel: sd 9:0:70:0: [sddl] CDB: Read(10): 28 00 19 fd 1d 50 00 00 a8 00
Jun 29 11:11:15 dcl01-oss01 kernel: end_request: I/O error, dev sddl, sector 436018512
```

<https://forums.servethehome.com/index.php?threads/lsi-sas2-and-sas3-hba-cards-optimal-firmware-versions-features.5829/>

<https://www.centos.org/forums/viewtopic.php?f=49&t=50032>

Downgraded LSI Controller Firmware from Version 20 to version 19.



# Firmware issue (cont)

The downgrade resolved the issue, but the next "zpool scrub" found a lot of parity errors:

```
[dcl01-oss01 /home/mmi/mmll116]$ sudo zpool status
[sudo] password for mmll116:
  pool: dcl01-pool01
  state: ONLINE
status: One or more devices has experienced an unrecoverable error. An
       attempt was made to correct the error. Applications are unaffected.
action: Determine if the device needs to be replaced, and clear the errors
       using 'zpool clear' or replace the device with 'zpool replace'.
       see: http://zfsonlinux.org/msg/ZFS-8000-9P
       scan: scrub repaired 15.4M in 120h24m with 0 errors on Fri Sep 11 21:42:00 2015
config:
```

NAME	STATE	READ	WRITE	CKSUM
dcl01-pool01	ONLINE	0	0	0
raidz2-0	ONLINE	0	0	0
j01-r00	ONLINE	0	0	0
<b>j01-r01</b>	<b>ONLINE</b>	<b>25</b>	<b>0</b>	<b>7</b>
j02-r00	ONLINE	0	0	58
j02-r01	ONLINE	0	0	0
j03-r00	ONLINE	0	0	0
j03-r01	ONLINE	0	0	8
j04-r00	ONLINE	0	0	34
j04-r01	ONLINE	0	0	18
j05-r00	ONLINE	0	0	15
j05-r01	ONLINE	0	0	33
raidz2-1	ONLINE	0	0	0
j01-r02	ONLINE	0	0	0
j01-r03	ONLINE	0	0	0
j02-r02	ONLINE	0	0	0
j02-r03	ONLINE	0	0	0
j03-r02	ONLINE	0	0	0
j03-r03	ONLINE	0	0	0

...

Also note number of errors on j01-r01. We opted to replace j01-r01. We then initiated another scrub...



# Firmware issue (cont)

The second scrub found another failed disk, which we replaced. During the resilver, the replacement disk failed:

```
# zpool status
pool: dcl01-pool01
state: DEGRADED
  scan: resilvered 604G in 14h53m with 0 errors on Sat Sep 19 17:45:33 2015
config:
```

NAME	STATE	READ	WRITE	CKSUM	
dcl01-pool01	DEGRADED	0	0	0	
raidz2-0	DEGRADED	0	0	0	
j01-r00	ONLINE	0	0	0	
j01-r01	ONLINE	0	0	0	
replacing-2	UNAVAIL	0	0	0	insufficient replicas
old	UNAVAIL	52	0	0	
j02-r00	UNAVAIL	3	503	0	corrupted data
j02-r01	ONLINE	0	0	0	
j03-r00	ONLINE	0	0	0	
j03-r01	ONLINE	0	0	0	
j04-r00	ONLINE	0	0	0	
j04-r01	ONLINE	0	0	0	
j05-r00	ONLINE	0	0	0	
j05-r01	ONLINE	0	0	0	



# Firmware issue (cont)

We ended up using our standard process to replace the drive:

```
# zpool status | head -24
pool: dcl01-pool101
state: DEGRADED
status: One or more devices is currently being resilvered.  The pool will
        continue to function, possibly in a degraded state.
action: Wait for the resilver to complete.
scan: resilver in progress since Mon Sep 21 15:20:49 2015
      2.96G scanned out of 171T at 379M/s, 131h46m to go
      11.4M resilvered, 0.00% done
config:
```

NAME	STATE	READ	WRITE	CKSUM	
dcl01-pool101	DEGRADED	0	0	0	
raidz2-0	DEGRADED	0	0	0	
j01-r00	ONLINE	0	0	0	
j01-r01	ONLINE	0	0	0	
replacing-2	UNAVAIL	0	0	0	insufficient replicas
old	UNAVAIL	52	0	0	
old	UNAVAIL	3	503	0	
j02-r00	ONLINE	0	0	0	(resilvering)
j02-r01	ONLINE	0	0	0	
j03-r00	ONLINE	0	0	0	
j03-r01	ONLINE	0	0	0	
j04-r00	ONLINE	0	0	0	
j04-r01	ONLINE	0	0	0	



# Todos

- X Install packages with no network
- X Graphic layout of VM
- X NO - makes VM larger - Add several 100MB zero and random files in /software
- X Additional RAID levels
- X Add a disk to a pool
- X Upgrade to larger disks
- X Add a spare
- X Add a log device (ZIL)
- X Scrubbing
- X Snapshots
- X import/export
- X Remove a disk from a pool (Can't)
- X zpool clear
- x zpool history
- X SLOG ARC ARC2CACHE ZIL
- X ZIL is contained on SLOG the same thing
- x zdb
- x /etc/zfs/zpool.cache
- X Options:
  - Mountpoint
  - Compression
  - Quota/Reservation
  - NO sharenfs
- Error Recovery:
  - X Lose disk (zpool replace)
  - X /etc/zfs/zpool.cache
- Add message in demo-setup
- - Change picture – rott drive is 8GB



# Links

<http://www.zfsbuild.com/2010/05/26/zfs-raid-levels/>  
[https://downloads.hpdd.intel.com/public/lustre/lustre-2.7.0/el6.6/server/RPMS/x86\\_64/](https://downloads.hpdd.intel.com/public/lustre/lustre-2.7.0/el6.6/server/RPMS/x86_64/)

<http://zfsonlinux.org/lustre-configure-single.html>  
<https://wiki.hpdd.intel.com/display/PUB/Rebuilding+the+Lustre-client+rpms+for+a+new+kernel>  
<http://www.zfsbuild.com/2010/05/26/zfs-raid-levels/>

<https://forums.freebsd.org/threads/suggested-zil-and-l2arc-sizes.37123/>

<https://pthree.org/2013/04/19/zfs-administration-appendix-a-visualizing-the-zfs-intent-log/>

<http://zfsonlinux.org/epel.html>

<http://zfsonlinux.org/lustre-configure-single.html>

[https://downloads.hpdd.intel.com/public/lustre/lustre-2.7.0/el6.6/server/RPMS/x86\\_64/](https://downloads.hpdd.intel.com/public/lustre/lustre-2.7.0/el6.6/server/RPMS/x86_64/)

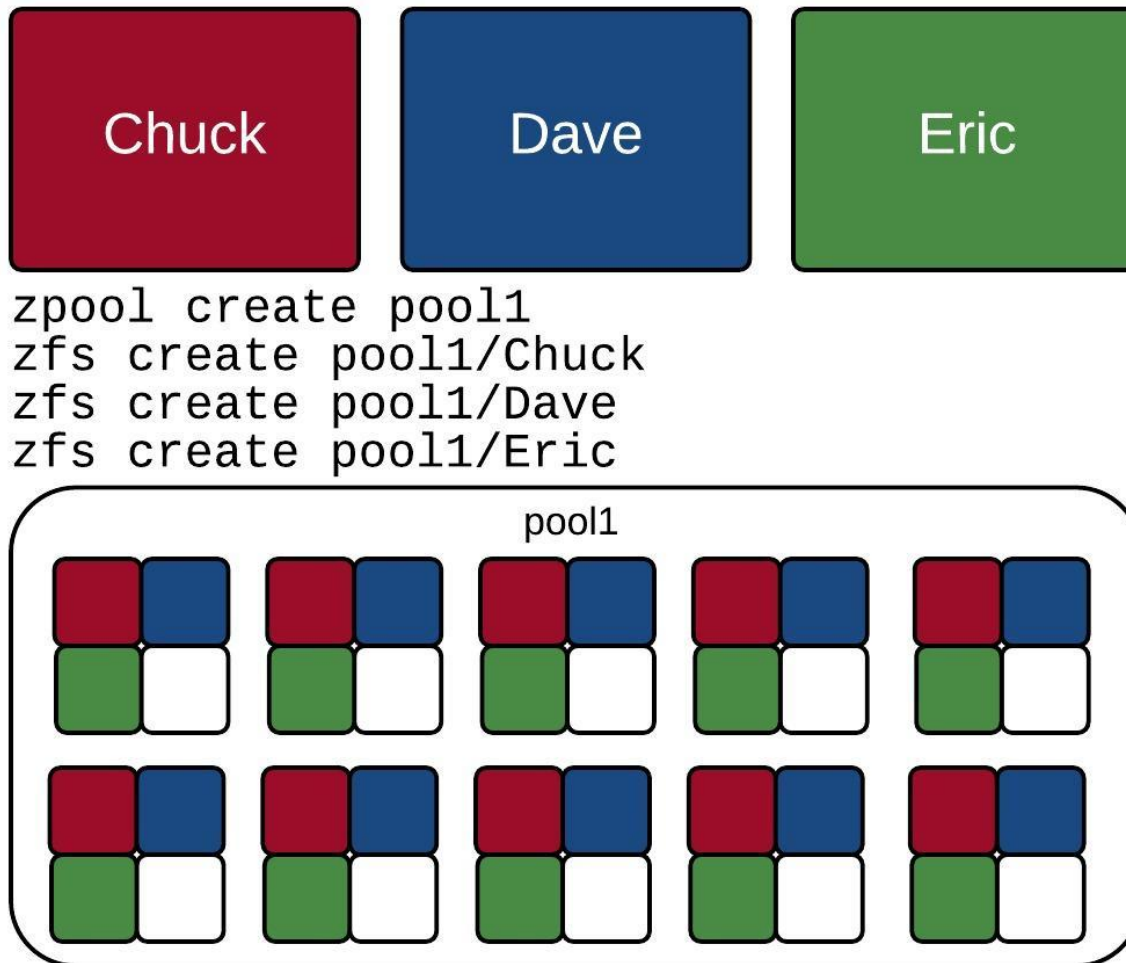
<https://forums.freenas.org/index.php?threads/some-insights-into-slog-zil-with-zfs-on-freenas.13633/>

<http://cuddletech.com/blog/pivot/entry.php?id=1013>





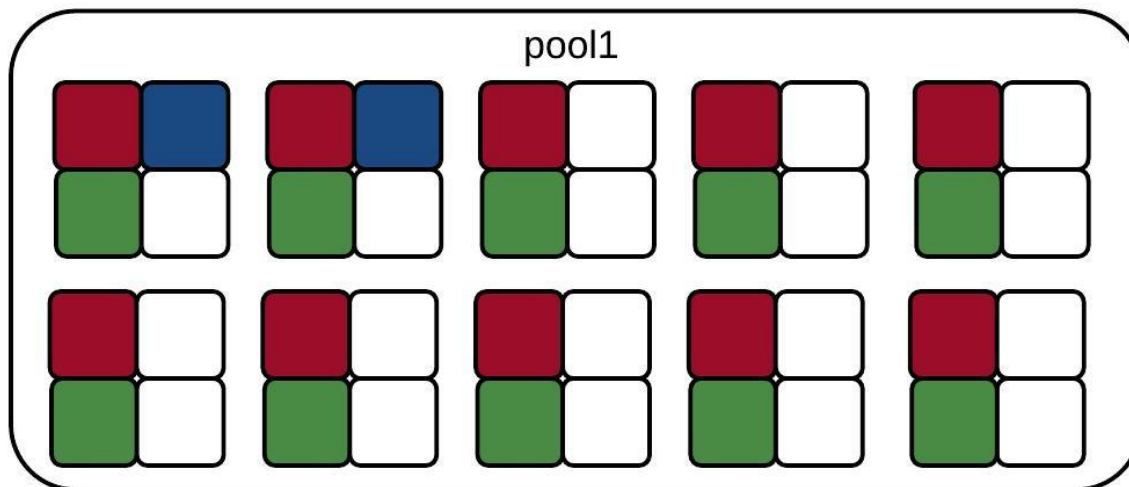
# Quotas and Reservations



# Quotas and Reservations



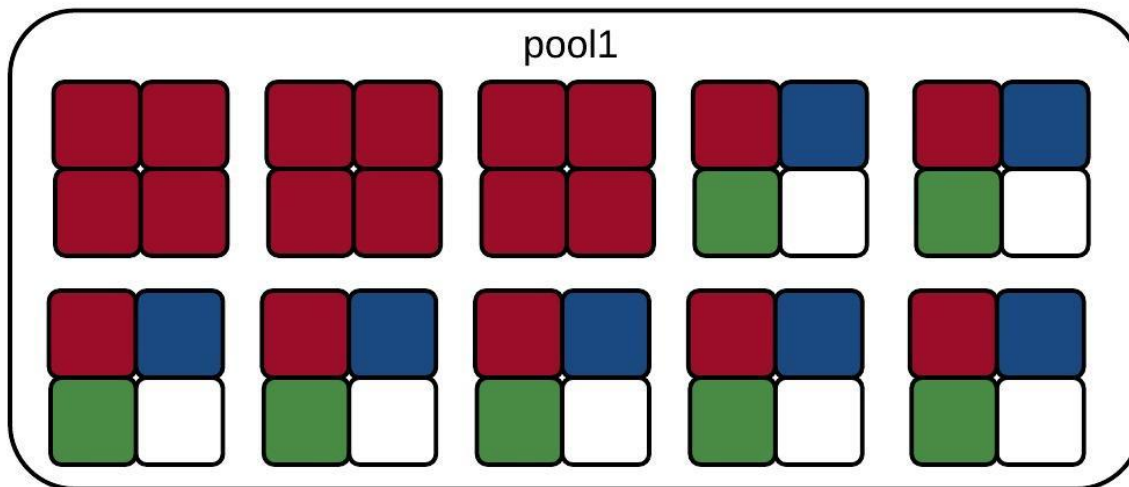
zfs set quota=2 pool1/Dave



# Quotas and Reservations



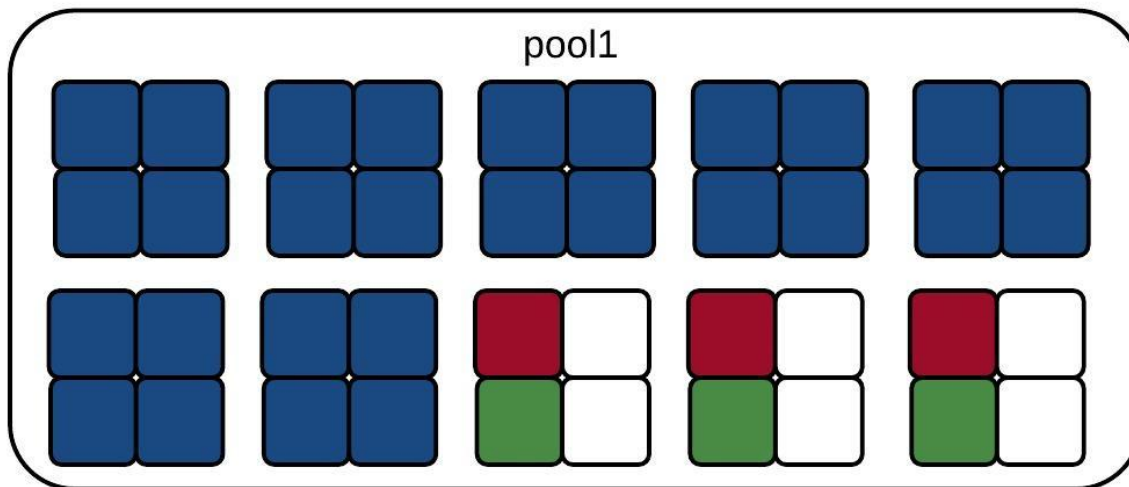
```
zfs set reservation=3 pool1/Chuck
```



# Quotas and Reservaitons



```
zfs set quota=6 pool1/Dave  
zfs set reservation=6 pool1/Dave
```



# Thanks for attending! Questions?

