Linux ZFS

Du möchtest dich gerne für unsere Hilfe erkenntlich zeigen . Gerne. Wir bedanken uns bei dir für deine Spende!

Spenden

Hauseigenes Apt-Repo: https://apt.iteas.at

GITLAB Enterprise:

ZFS (ursprünglich Zettabyte File System) wird oft als Dateisystem angesehen, was im Grunde genommen ein Missverständnis darstellt. ZFS kann ein Dateisystem sein, aber beherrscht auch noch einiges mehr. Es vereint die Funktionalität eines Logical Volume Managers und eines Software-RAID mit einem Copy-on-Write-Dateisystem (COW). Das heißt, dass es (aufgrund seiner Kenntnisse der Festplattenbelegung) effizienter als jedes Hardware-RAID arbeitet, Daten-Integrität per Transaktionen ähnlich wie bei relationalen Datenbanken sichert und im Falle von Daten-Redundanz (Mehrfachspeicherung) sogar selbständig Daten repariert.

- Systemgrundlage: Proxmox 3.4 / 4.X / 5.0
- ZFSutils: 0.6.4.3 / 0.6.5-pve6~jessie
- Kernel: 3.10.0-10-pve / 4.2.3-2-pve / 4.10.17-4-pve

Alles was unter diese Version liegt, ist entweder durch Bugs, oder fehlende Features wie z.B. das ZFS die Geräte per ID anspricht, nicht brauchbar.

Option Ashift

Advanced Format (AF) is a new disk format which natively uses a 4,096 byte instead of 512 byte sector size. To maintain compatibility with legacy systems AF disks emulate a sector size of 512

bytes. By default, ZFS will automatically detect the sector size of the drive. This combination will result in poorly aligned disk access which will greatly degrade the pool performance.

Therefore the ability to set the ashift property has been added to the zpool command. This allows users to explicitly assign the sector size when devices are first added to a pool (typically at pool creation time or adding a vdev to the pool). The ashift values range from 9 to 16 with the default value 0 meaning that zfs should auto-detect the sector size. This value is actually a bit shift value, so an ashift value for 512 bytes is 9 (29 = 512) while the ashift value for 4,096 bytes is 12 (212 = 4,096). To force the pool to use 4,096 byte sectors at pool creation time.

Sämtliche neue Festplatten im Enterpriseberreich wie z.B. die WD Red PRO haben diese Sectorengröße. Man sieht das auch schön mit den Smarttools:

```
=== START OF INFORMATION SECTION ===
Model Family:     Western Digital Red Pro
Device Model:     WDC WD2001FFSX-68JNUN0
Serial Number:    WD-WMC5C0D688XW
LU WWN Device Id: 5 0014ee 003f548a4
Firmware Version: 81.00A81
User Capacity:    2,000,398,934,016 bytes [2.00 TB]
Sector Sizes:     512 bytes logical, 4096 bytes physical
Rotation Rate:    7200 rpm
Device is:        In smartctl database [for details use: -P show]
ATA Version is:   ATA8-ACS (minor revision not indicated)
SATA Version is:  SATA 3.0, 6.0 Gb/s (current: 6.0 Gb/s)
Local Time is:    Mon Jan  4 23:23:43 2016 CET
SMART support is: Available - device has SMART capability.
SMART support is: Enabled
```

Memorylimit setzen

Genaue Info zu den aktuellen Werten bekommt man mit:

```
arcstat
```

oder

```
arc_summary
```

Wenn man möchte kann man ZFS ein Memorylimit setzten.

```
nano  /etc/modprobe.d/zfs.conf
options zfs zfs_arc_max=10737418240
```

oder 8GB

```
8589934592
```

Danach noch die initram updaten und rebooten.

```
update-initramfs -u
```

Anlegen eines neuen Pools im Raid10 und hinzufügen zweier weiteren Festplatten

Wir gehen hier von einer Mindeskonfiguration von 4 Platten aus. Erstellen im nächsten Befehl 2 Raid1 und legen dann hierdrüber noch ne 0. Also Raid10.

```
zpool create -f -o ashift=12 <pool-name> mirror <device1> <device2> mirror
<device3> <device4>
```

Der Befehl würde dann so aussehen:

```
zpool create -f -o ashift=12 v-machines mirror ata-
WDC_WD2001FFSX-68JNUN0_WD-WMC5C0D0KRWP ata-WDC_WD20EARX-00ZUDB0_WD-
WCC1H0343538 mirror ata-WDC_WD2001FFSX-68JNUN0_WD-WMC5C0D688XW ata-
WDC_WD2001FFSX-68JNUN0_WD-WMC5C0D63WM0
```

Wichtig ist immer die ID's zu benutzen. Würde man z.B. sda oder sdb nehmen, wird es vorkommen das sich die Reihenfolge beim booten irgendwann mal ändert, und dann fehlen Platten. Insbesondere wenn man mehrere SAS/SATA Controller zur Verfügung hat. Die IDs kann man hier auslesen: Als Beispiel SDB

```
ls  /dev/disk/by-id/ -l | grep sdb
lrwxrwxrwx 1 root root  9 May 23 11:50 ata-WDC_WD2001FFSX-68JNUN0_WD-
WMC5C0D688XW -> ../../sdb
lrwxrwxrwx 1 root root 10 May 23 11:50 ata-WDC_WD2001FFSX-68JNUN0_WD-
WMC5C0D688XW-part1 -> ../../sdb1
lrwxrwxrwx 1 root root 10 May 23 11:50 ata-WDC_WD2001FFSX-68JNUN0_WD-
WMC5C0D688XW-part9 -> ../../sdb9
lrwxrwxrwx 1 root root  9 May 23 11:50 scsi-SATA_WDC_WD2001FFSX-_WD-
WMC5C0D688XW -> ../../sdb
lrwxrwxrwx 1 root root 10 May 23 11:50 scsi-SATA_WDC_WD2001FFSX-_WD-
WMC5C0D688XW-part1 -> ../../sdb1
lrwxrwxrwx 1 root root 10 May 23 11:50 scsi-SATA_WDC_WD2001FFSX-_WD-
WMC5C0D688XW-part9 -> ../../sdb9
lrwxrwxrwx 1 root root  9 May 23 11:50 wwn-0x50014ee003f548a4 -> ../../sdb
lrwxrwxrwx 1 root root 10 May 23 11:50 wwn-0x50014ee003f548a4-part1 ->
../../sdb1
lrwxrwxrwx 1 root root 10 May 23 11:50 wwn-0x50014ee003f548a4-part9 ->
../../sdb9
```

Wenn man eine Festplatte einsteckt, werden auch im Syslog alle wichtigen Infos angezeigt. Um dann noch zwei weitere Platten zu unserem 4er Gespann hinzuzufügen bedient man diesen Befehl:

```
zpool add -o ashift=12 v-machines mirror ata-WDC_WD20EARX-00ZUDB0_WD-
```

```
WCC1H0381420 ata-WDC_WD20EURS-63S48Y0_WD-WMAZA9381012
```

Nun hat man 6 Platten im Raid10:

```
zfs status
```

```
pool: v-machines
 state: ONLINE
  scan: resilvered 1.09T in 4h45m with 0 errors on Sat May 23 02:48:52 2015
config:

    NAME                                            STATE     READ WRITE
CKSUM
    v-machines                                      ONLINE       0     0
0
      mirror-0                                      ONLINE       0     0
0
        ata-WDC_WD2001FFSX-68JNUN0_WD-WMC5C0D0KRWP  ONLINE       0     0
0
        ata-WDC_WD20EARX-00ZUDB0_WD-WCC1H0343538    ONLINE       0     0
0
      mirror-1                                      ONLINE       0     0
0
        ata-WDC_WD2001FFSX-68JNUN0_WD-WMC5C0D688XW  ONLINE       0     0
0
        ata-WDC_WD2001FFSX-68JNUN0_WD-WMC5C0D63WM0  ONLINE       0     0
0
      mirror-2                                      ONLINE       0     0
0
        ata-WDC_WD20EARX-00ZUDB0_WD-WCC1H0381420    ONLINE       0     0
0
        ata-WDC_WD20EURS-63S48Y0_WD-WMAZA9381012    ONLINE       0     0
0

errors: No known data errors
```

Die Daten werden bei einer Erweiterung nicht gesynct. Somit ist der Erweitungsprozess binnen Minuten abgeschlossen. Unabhängig von der Speichergröße. Daten werden erst dann auf die neuen Platten verteilt wenn diese das erste mal geschrieben werden.

Wichtig ist auch zu erwähnen das die neu hinzugefügten Platten zuerst bevorzugt auf den Füllstand der anderen Platten gebracht werden, erst dann werden wieder immer alle HDD's gleichzeitg verwendet.

Anlegen eines neuen Pools im RaidZ10 (entspricht Raid50)

Annahme ist hier ein Gerät mit 2 SDDs für das OS im Raid1 (läuft bereits). Nun bauen wir 10 Stück

SATA Fesplatten mit einer Enterprise SSD für für Cache und Log dazu. Man das ganze nicht auf einmal erstellen, also zuerst die hälfte der Disks + Log und Cache, danach den Rest:

```
zpool create -f -o ashift=12 stytank01 raidz1 ata-WDC_WD2001FFSX-68JNUN0_WD-WMC5C0D1LZD5 ata-WDC_WD2001FFSX-68JNUN0_WD-WMC5C0D3404L ata-WDC_WD2001FFSX-68JNUN0_WD-WMC5C0D43PSX ata-WDC_WD2001FFSX-68JNUN0_WD-WMC5C0D4JVEY ata-WDC_WD2001FFSX-68JNUN0_WD-WMC5C0D6U08D log ata-SAMSUNG_MZ7KM240HAGR-00005_S2HRNX0HB00643-part1 cache ata-SAMSUNG_MZ7KM240HAGR-00005_S2HRNX0HB00643-part2

zpool add  -o ashift=12 stytank01 raidz1  ata-WDC_WD2002FFSX-68PF8N0_WD-WMC6N0EAUML4 ata-WDC_WD2002FFSX-68PF8N0_WD-WMC6N0E6U9RS ata-WDC_WD2001FFSX-68JNUN0_WD-WMC5C0DA95YF ata-WDC_WD2001FFSX-68JNUN0_WD-WMC5C0D9CNRT ata-WDC_WD2001FFSX-68JNUN0_WD-WMC5C0D78NCJ
```

Das ganze sieht dann so aus:

```
zpool status
  pool: rpool
 state: ONLINE
  scan: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        rpool       ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            sdg2    ONLINE       0     0     0
            sdi2    ONLINE       0     0     0

errors: No known data errors

  pool: stytank01
 state: ONLINE
  scan: none requested
config:

        NAME                                                STATE
READ WRITE CKSUM
        stytank01                                           ONLINE
0     0     0
          raidz1-0                                          ONLINE
0     0     0
            ata-WDC_WD2001FFSX-68JNUN0_WD-WMC5C0D1LZD5      ONLINE
0     0     0
            ata-WDC_WD2001FFSX-68JNUN0_WD-WMC5C0D3404L      ONLINE
0     0     0
            ata-WDC_WD2001FFSX-68JNUN0_WD-WMC5C0D43PSX      ONLINE
0     0     0
            ata-WDC_WD2001FFSX-68JNUN0_WD-WMC5C0D4JVEY      ONLINE
0     0     0
            ata-WDC_WD2001FFSX-68JNUN0_WD-WMC5C0D6U08D      ONLINE
```

```
0     0     0
          raidz1-2                                            ONLINE
0     0     0
            ata-WDC_WD2002FFSX-68PF8N0_WD-WMC6N0EAUML4         ONLINE
0     0     0
            ata-WDC_WD2002FFSX-68PF8N0_WD-WMC6N0E6U9RS         ONLINE
0     0     0
            ata-WDC_WD2001FFSX-68JNUN0_WD-WMC5C0DA95YF         ONLINE
0     0     0
            ata-WDC_WD2001FFSX-68JNUN0_WD-WMC5C0D9CNRT         ONLINE
0     0     0
            ata-WDC_WD2001FFSX-68JNUN0_WD-WMC5C0D78NCJ         ONLINE
0     0     0
        logs
          ata-SAMSUNG_MZ7KM240HAGR-00005_S2HRNX0HB00643-part1  ONLINE
0     0     0
        cache
          ata-SAMSUNG_MZ7KM240HAGR-00005_S2HRNX0HB00643-part2  ONLINE
0     0     0

errors: No known data errors
```

Nachdem dies ein Rpool ist, danach nicht vergessen Grub auf allen hinzugefügten Platten zu installieren.

Korrupte Daten

Wenn kaputte Daten vorliegen aus was für Gründen auch immer und ZFS diese nicht automatisch reparieren kann und Festplatte doch noch in Ordnung ist, kann man versuchen die Platte mit sich selbst zu ersetze, oder was noch besser ist, die Platte Offline schalten vom Verbund entfernen und wieder hinzuzufügen. In meinem Fall hier war es eine defekte WD Red Pro Festplatte, die nach zwei Wochen Betrieb herumspinnte und komplett zusammenviel.

Hier wird die Platte mit sich selbst ersetzt.

```
zpool replace v-machines ata-WDC_WD20EARX-00ZUDB0_WD-WCC1H0381420  ata-
WDC_WD20EARX-00ZUDB0_WD-WCC1H0381420
```

Geht das dann trotz force nicht, kann man die Platte vom Verbund entfernen und neu hinzufügen:

```
zpool offline v-machnies ata-WDC_WD20EARX-00ZUDB0_WD-WCC1H0381420
zpool detach v-machines ata-WDC_WD20EARX-00ZUDB0_WD-WCC1H0381420
zpool attach -o ashift=12 v-machines ata-WDC_WD20EURS-63S48Y0_WD-
WMAZA9381012 ata-WDC_WD20EARX-00ZUDB0_WD-WCC1H0381420
```

Beim Hinzufügen müssen natürlich beide Platten angegeben werden. Zuerst die vorhandene und dann welche hinzugefügt werden soll. Danach wird der Resilverprozess gestartet.

Festplatte ersetzen

Um eine defekte Fesplatte zu ersetzen geht man folgender Maßen vor. Um event. Probleme zu vermeiden kann man die Platte vorher auch partitionslos machen.

```
zpool replace <poolname> <alte ID> <neue ID>
```

Danach startet unverzüglich der Resilverprozess. Es wird natürlich nur der vorher genutzte Speicherplatz der alten Platte gesynct.

Festplatte im Rpool ersetzen

Im Rpool ist etwas mehr zu tun, da Linux nicht von ZFS booten kann. Wir gehen hier davon aus wir die Platte im gleichen Festplattenslot tauschen, was bei einen Rpool auf z.B. SSD nicht ungewöhnlich ist. Zuerst übernehmen wir mal den Partiontable einer anderen Platte im Pool. Hier ein RaidZ-1.

SDB ist die neue Platte und war gleichzeitig auch die defekte. SDA ist eine andere Platte im Rpool.

```
sgdisk -R /dev/sdb /dev/sda
```

Da wir hier auch die UUID mit kopieren muss diese mit dem nächsten Befehl neu gesetzt werden.

```
sgdisk -G /dev/sdb
```

Nun den Resilverprozess anstoßen.

```
zpool replace -f rpool sdb2 sdb2
Make sure to wait until resilver is done before rebooting.
```

Da wir ja den gleichen Festplattenslot verwenden ersetzen wir die Platte mit sich selbst. Zum Besseren Verständnis kann man natürlich immer mit den DiskID's arbeiten. **/dev/disk/by-id/** Es es wirklich die gleiche Platte, also nicht nur der gleiche Anschluss, genügt ein

```
zpool online rpool sdb2
```

```
root@pvetest:/home# zpool status
  pool: rpool
 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 Sat Nov 28 22:46:22 2015
    192M scanned out of 1.48G at 32.0M/s, 0h0m to go
    61.0M resilvered, 12.65% done
config:

    NAME             STATE     READ WRITE CKSUM
    rpool            DEGRADED     0     0     0
```

```
      raidz1-0       DEGRADED     0     0     0
        sda2         ONLINE       0     0     0
        replacing-1  UNAVAIL      0     0     0
          old        UNAVAIL      0     0     0
          sdb2       ONLINE       0     0     0  (resilvering)
        sdc2         ONLINE       0     0     0

errors: No known data errors
```

Nun noch Grub auf die neue Platte schreiben:

```
grub-install /dev/sdb
Installing for i386-pc platform.
Installation finished. No error reported.
```

Nach dem Resilvern sieht das ganze wieder schön aus:

```
pool: rpool
 state: ONLINE
  scan: resilvered 486M in 0h0m with 0 errors on Sat Nov 28 22:46:52 2015
config:

    NAME        STATE     READ WRITE CKSUM
    rpool       ONLINE       0     0     0
      raidz1-0  ONLINE       0     0     0
        sda2    ONLINE       0     0     0
        sdb2    ONLINE       0     0     0
        sdc2    ONLINE       0     0     0

errors: No known data errors
root@pvetest:~# reboot
Write failed: Broken pipe
```

Die wichtigsten ZFS-Befehle auf einen Blick

ZFS Commando	**Beschreibung**
zpool status	Zeigt eine Übersicht aller Pools und den Status dieser
zpool list	Zeigt alle Zpools in Kurzform an
zfs list	Zeigt alle ZFS-Pools und Datasets inkl. Einhängepunkte an
zpool iostat -v	Zeigt alle Festplattenaktiviäten genau an
zfs set compression=lz4 <pool-name>	Aktiviert die Komprimierung des Dateisystem, empfohlen gut 7mal so schnell wie ohne.

zpool upgrade <poolname>	Hebt den geannten Pool auf eine neue ZFSversion.
zpool upgrade -a	Hebt alle Pools auf die neu installierte ZFSversion.
zfs create v-machines/home	Legt ein neues Dataset namens „home“ im Pool „v-machnes“ an.
zfs get all v-machines/home	Zeigt sämtliche Infos und Attribute dieses Datasets an.
zfs set acltype=posixacl v-machines/home	Schaltet ACLs im Dataset „home“ an.
zfs set acltype=off v-machines/home	Schaltet diese ACLs aus.
zfs get mounted	Zeigt alle ZFS Mountpoints und den Status dieser an.
zfs mount v-machines/home	Hängt das Dataset „home“ ein.
zfs unmount v-machines/home	Hängt das Dataset „home“ aus.
mount /dev/zvol/v-machines/home/vm-<Nummer>-disk-<Nummer> /<mountpoint>	Hängt eine VMdisk vom Zpool „v-machines“ am gewünschten Mountpoint ein.
zpool import -d /dev/disk/by-id/ -a	ersetzt SDX durch die ID der Festplatte
zpool import v-machines neuepoolname	importiert einen bestehenden Pool mit einem anderen Namen
zfs list -t snapshot	zeigt alle Snapshots an
zpool set listsnapshots=on rpool	zeigt bei „zfs list“ auch snapshots an
zfs list -r -t snapshot -o name,creation rpool	Snapshots mit Datum anzeigen
zfs get volsize v-machines/HDD-vmdata-KVM/vm-113-disk-1 NAME PROPERTY VALUE SOURCE v-machines/HDD-vmdata-KVM/vm-113-disk-1 volsize 36G local	zeigt den maximal möglichen Speicher eines Volumes an
zfs set volsize=32g v-machines/HDD-vmdata-KVM/vm-113-disk-1	Verkleinert/Vergrößert das zvol auf 32GB (Blockdevice)
zfs create -V 5gb tank/vol	legt ein neues zvol mit einer maximalen Größe von 5G an (Blockdevice)
zfs set quota=50g tank/backupfolder	setzt die Quota eines normalen Datesets auf 50g
zfs rename -p rpool/test rpool/server123	Verschiebt eine Dataset
zfs list -o space	Speicherauslastung inkl. wie viel für Snapshots verbraucht wird
mount -t zfs -o ro v-machines/home@rep_home_2017-07-05_00:36:48 /mnt/zfsmountsnap	Snapshot mounten
zfs set primarycache=metadata pool/dataset	Optimal für Backups, hier wird der Ramverbrauch erheblich reduziert
apt install zfs-auto-snapshot	Infolink

ZFS Snapshots und deren Verwaltung

Snapshots eigenes sich hervorragend für die Datensicherung oder auch wenn man was testet oder auch nur Updates fährt. Man kann sofort wieder zurück. Um nun ein Snapshot von unserem Rootfilesystem zu erstellen geht von wie folgt vor:

```
zfs snapshot rpool/ROOT/pve-1@Freitag
```

Man generiert Snapshots immer von einem Dataset. Der ZFSpool kann ja mehreren Datasets bestehen. Wie bei unserer Testmaschine hier:

```
NAME                       USED  AVAIL  REFER  MOUNTPOINT
backup                    56.2G   618G  56.2G  /backup
rpool                     61.8G   836G   128K  /rpool
rpool/ROOT                1.04G   836G   128K  /rpool/ROOT
rpool/ROOT/pve-1          1.04G   836G  1.04G  /
rpool/ROOT/pve-1@Freitag  3.71M      -  1.04G  -
rpool/home                56.5G   836G  56.5G  /rpool/home
rpool/home@bla1            107K      -  54.6G  -
rpool/home@Freitag        10.7K      -  56.5G  -
rpool/swap                4.25G   841G  85.2K  -
```

Hier sieht man der Rpool aus den Datasets pve-1,swap und home besteht. Die Einträge mit dem „@“ sind snapshots. Um nun ein Snapshot zurück zu spielen z.B. am Dataset home bedient man sich diesem Befehle:

```
zfs rollback rpool/home@Freitag
```

Man darf sich hier vom Pool nicht verwirren lassen. Auch direkt am Rootpool kann man problemlos ein Snapshot zurückrollen.

```
zfs rollback rpool/ROOT/pve-1@Freitag
```

Sind neuere Snapshots vorhanden muss man die vorher löschen. Man kann das aber auch gleich im obigen Befehle mit der Option „r“ forcen und includen. Natürlich kommt es Rootfilesystem auf die Änderungen an. Wenn man z.B. den Teil vernichtet der für die Snapshots zuständig ist, geht das dann natürlich nicht mehr. Deswegen sollte man ich immer den Rpool nur für das nackte System verwenden und eigene Pools mit eigenen Datasets generieren um solchen Dingen zu entgehen. Wann Snapshots erstellt wurden kann mit folgenden Befehl gut sehen: Hier mit dem Rpool.

```
zfs list -r -t snapshot -o name,creation rpool

NAME                       CREATION
rpool/ROOT/pve-1@Freitag  Fri Dec  4 23:19 2015
rpool/home@bla1           Sat Dec  5 20:11 2015
```

Snapshots Remote abspeichern

Um nun auch wirkliche Backups zu generieren muss man die Snapshots natürlich außerhalb des Servers ablegen. In unserem ersten Beispiel senden wir einen Snapshot auf eine Backupfestplatte in unserem Server.

```
 zpool list backup
NAME     SIZE  ALLOC   FREE  EXPANDSZ   FRAG    CAP  DEDUP  HEALTH  ALTROOT
backup   696G  56.2G   640G         -     4%     8%  1.00x  ONLINE  -

zfs list backup
NAME     USED  AVAIL  REFER  MOUNTPOINT
backup  56.2G   618G  56.2G  /backup
```

Unkomprimiert:

```
zfs send -v rpool/home@bla1  > /backup/home@bla1.snap
```

Mit Komprimierung:

```
zfs send -v rpool/home@bla1 | pigz |  > /backup/home@bla1.gz
```

Mit Komprimierung und SSL Verschlüsselung:

```
zfs send -v rpool/home@bla1 | pigz | openssl enc -aes-256-cbc -a -salt >
/backup/home@bla1.gz.ssl
```

Senden eines Snapshots über SSH auf ein ZFS Filesystem auf einem anderen Host

```
zfs send -v rpool/home@bla1 | ssh otherhost "/usr/sbin/zfs receive
otherpool/home@bla1"
```

Im nächsten beispiel befinden wir uns auf dem Server node2.tux.local. Am node1.tux.local ist eine 8TB Festplatte mit einem Pool eingebunden für externe Backups. Um nun die Festplatte nicht umstecken zu müssen, kann das ganze einfach in SSH pipen.

```
zfs send rpool/ROOT/pve-1@halbjahresbackup | ssh node1.tux.local zfs receive
backup/pve-1-node1
```

Snapshots inkrementell abspeichern

Die beste Methode für Backups ist wohl diese. Es werden hier immer nur die Änderungen gespeichert und man kann auch auf jede einzelne Datei zugreifen. In diesem Fall möchten wir wieder unser home-Dataset auf einer anderen Festplatte abspeichern. Zuerst müssen wir das Dataset auf der Zielfestplatte erstellen. Danach machen wir unser erstes Snapshot und überspielen das auf diese Platte. ACHTUNG: Feature wie Posixacls werden nicht mit aktiviert. Diese mit „zfs set …“ einfach vorher oder nachher nach Wunsch aktivieren.

```
zfs snapshot rpool/home@Montag
zfs send -v rpool/home@Montag | zfs receive  backup/home
```

Nun machen wir auf unseren lokalen Homeverzeichnis Änderungen. Am Dienstag erstellen wir wieder ein Snapshot, und spielen das wieder auf die Festplatte:

```
zfs snapshot rpool/home@Dienstag
zfs send -v -i rpool/home@Montag rpool/home@Dienstag | zfs receive
backup/home
```

So kann man das Spiel immer weiter treiben. Wichtig ist das man darauf achtet immer zwei Snapshots zu behalten. Sonst kann natürlich kein Vergleich angestellt werden. „backup/home“ sollte auch nicht gemountet sein. Es kann sonst zu Problemen kommen. Lässt man die Snapshots am Backup bestehen, hat man auch gleich eine Versionierung. Also vorher immer aushängen:

```
zfs umount backup/home
```

Möchte man nun das Backup zurück spielen, geht man gleicher Maßen vor, nur umgekehrt:

```
zfs send -v backup/home@Dienstag | zfs receive  rpool/home -F
```

Im nächsten beispiel befinden wir uns auf dem Server node2.tux.local. Am node1.tux.local ist eine 8TB Festplatte mit einem Pool eingebunden für externe Backups. Um nun die Festplatte nicht umstecken zu müssen, kann das ganze einfach in SSH pipen. Natürlich vorher neues Snapshot zum Abgleich erstellen.

```
zfs send rpool/ROOT/pve-1@halbjahresbackup
rpool/ROOT/pve-1@halbjahresbackup1 | ssh node1.tux.local zfs receive
backup/pve-1-node1
```

Danach kann man auf beiden seiten das alte Snapshot löschen. Muss man natürlich nicht, wenn man mal weiter zurück möchte.

```
zfs destroy rpool/ROOT/pve-1@halbjahresbackup
zfs destroy backup/pve-1-node1@halbjahresbackup
```

Snapshot kann nicht gesendet werden

Sollte das Snapshot nicht gesendet werden können z.b. hier ein riesiges Homedataset:

```
zfs send -v -i v-machines/home@halbjahresbackup v-
machines/home@halbjahresbackup1 | zfs receive backup/home
send from @halbjahresbackup to v-machines/home@halbjahresbackup1 estimated
size is 116G
total estimated size is 116G
TIME        SENT   SNAPSHOT
cannot receive incremental stream: destination backup/home has been modified
```

```
since most recent snapshot
warning: cannot send 'v-machines/home@halbjahresbackup1': Broken pipe
```

So kann sich die Änderungen mit diff ausgeben lassen.

```
zfs diff backup/home@halbjahresbackup backup/home
```

Danach sollten Änderungen angezeigt werden. Wird hier nichts angezeigt, ist der Inhalt ok. Dann aber sein das Features am Ziel aktiviert hat, das wäre auch schon eine Änderung. Man hat nun zwei Möglichkeit. Man kann mit der Option „-F“ das ganze forcen. Dann wird aber auch am Ziel auf der receive seite automatisiert den rollback durchführt, sorgt aber auch dafür dass snapshots die auf der Quelle nicht mehr vorhanden sind am Ziel gelöscht werden. Ich habe mich hier für die zweite Variante entschieden wo einfach am Ziel ein Rollbackup des Snapshots durchgeführt habe:

```
zfs rollback backup/home@halbjahresbackup
```

Danach funktioniert auch das Senden des Snapshots so wie oben angeführt.

Snapshot von einem anderen Ort zurückspielen

Um ein Snapshot zurückspielen zu können muss man zuerst vorhandene lokale Snapshots löschen:

```
zfs destroy rpool/home@bla1
zfs destroy rpool/home@Freitag
```

Um nun das Snapshot von unserem zweiten Beispiel von /backup wieder zurück zu spielen geht man folgender maßen vor.

```
pigz -d -c /backup/home@bla1.gz | zfs receive -F rpool/home
```

Snapshot mounten und Daten rausholen

Sehr praktisch. Um ein Snapshot zu mounten kann den normalen Mountbefehl benutzen. Z.B.

```
mount -t zfs v-machines/home@rep_home_2017-07-05_00:36:48 /mnt/zfsmountsnap
```

Um das ganze automatisch und elegant mit ZFS zu gestalten machten den Ordner `.zfs` in Datasets sichtbar. Z.B.

```
zfs set snapdir=visible  v-machines/home
```

Danach wird das gewünschte Snapshot automatisch beim Zugriff auf `.zfs/snapshot/rep_home_2017-07-16_00:01:25` gemountet.

Snapshot klonen

Es ist oft sehr praktisch ein Filesystem zu klonen um Dinge auszutesten.

```
zfs clone tank/test/productA@today tank/test/productAbeta
```

```
zfs list -r tank/test
NAME                        USED  AVAIL  REFER  MOUNTPOINT
tank/test                   104M  66.2G    23K  /tank/test
tank/test/productA          104M  66.2G   104M  /tank/test/productA
tank/test/productA@today       0      -   104M  -
tank/test/productAbeta         0  66.2G   104M  /tank/test/productAbeta
zfs promote tank/test/productAbeta

zfs list -r tank/test
NAME                           USED  AVAIL  REFER  MOUNTPOINT
tank/test                      104M  66.2G    24K  /tank/test
tank/test/productA                0  66.2G   104M  /tank/test/productA
tank/test/productAbeta         104M  66.2G   104M  /tank/test/productAbeta
tank/test/productAbeta@today      0      -   104M  -
```

Mit komplettem ZFSpool umziehen

Mit einem kompletten Pool um zu ziehen ist super einfach, in Gegesatz zu nem Rsync oder ähnlichem. Zuerst macht man rekursiv ein Snapshot, danach kopiert man dem Pool auf die neuen/anderen Platten.

```
zfs snapshot -r oldpool@migration
zfs send -v -R -p oldpool@migration  | zfs receive -F myfreshpool
```

Autoexpand

Ist „autoexpand“ aktiviert vergrößert sich der Festplattenspeicher beim Tausch einer kleineren durch eine größere Platte automatisch. Beispiel: Man hat ein Raid1 mit 2x500GB Festplatten. Man tauscht die platten nacheinander aus (replace/resilvern). Ist die zweite Platte getauscht ist der Mehrspeicher sofort verfügbar.

Um für alle Pools abzugragen ob die Funktion aktiv ist gibt man folgendes Befehl ein:

```
zpool get autoexpand

NAME        PROPERTY    VALUE   SOURCE
rpool       autoexpand  off     default
v-machines  autoexpand  on      local
```

Eingeschaltet wurde „autoexpand“ auf „v-machines“ mit folgendem Befehl:

```
zpool set autoexpand=on v-machines
```

Autoexpand auf einem Rootpool

Das ganze ist ein wenig komplizierter da man die GPT Bootpartion beachten muss. Zuerst erstellt auf der neuen getauschten Disk eine GPT Partition:

```
sgdisk -Z /dev/sdf # löscht nur gpt und mbr struktur
sgdisk -Z -o /dev/sdf # löscht auch Partitionen
sgdisk -a1 -n1:34:2047 -t1:EF02 -n9:-8M:0 -t9:BF07 -n2:2048:0 -t2:BF01 -c
2:zfs /dev/sdf
zpool replace rpool 10714300945297318711 sdf2
grub-install /dev/sdf
```

Das natürlich mit jeder Platte wiederholen.

that basically means:

1: BIOS Boot partition (⇐1M)

2: ZFS partition (all free space)

9: reserved space (8MB)

Umwandeln eines Rpool Singledisk in einen Mirror inkl. Autoexpand

Annahme ist hier ein Rpool mit einer Samsung EVO750. Da die Disk nicht Enterprise ist und das Wearoutlevel schon bei 90% ist, fügen wir eine Samsung SM863a als Mirror hinzu. Dann können wir beim Ausfall der EVO bequem eine weitere SM863a hinzufügen. Der zeitiger Status ist:

```
zpool status
  pool: rpool
 state: ONLINE
  scan: scrub repaired 0B in 0h3m with 0 errors on Sun Oct 14 00:27:05 2018
config:

        NAME        STATE     READ WRITE CKSUM
        rpool       ONLINE       0     0     0
          sda2      ONLINE       0     0     0
```

Rpool's könne hier nur mit den Alias „SDX“ umgehen. Alle anderen „nicht Rootpools“ mit `/dev/disk/by-id` Wir haben nun unsere neue SSD in's laufende System gehängt. Diese scheint mit `sdb` auf. Wir konfigurieren diese DISK mit GPT schauen das wie oben beschrieben auch Autoexpand am Pool aktiviert ist und partitionieren diese richtig:

```
sgdisk -a1 -n1:34:2047 -t1:EF02 -n9:-8M:0 -t9:BF07 -n2:2048:0 -t2:BF01 -c
2:zfs /dev/sdb
Setting name!
```

```
partNum is 1
REALLY setting name!
The operation has completed successfully.
```

Mit `partx -s /dev/sdb` sehen wir das die Disk nun richtig konfiguriert wurde:

```
partx -s /dev/sdb
NR     START       END   SECTORS   SIZE NAME UUID
 1        34      2047      2014  1007K      abdb3664-8f24-4f9c-
b69f-48041a12dd2a
 2      2048 468845709 468843662 223,6G zfs  54a8aa0b-3e13-4e05-
b8f6-3bcc1a5452d5
 9 468845710 468862094     16385     8M      8dfaf865-
cv4f-4931-86ba-9a9a274d3ead
```

Nun können wir die neue Disk zu unserer alten dazu hängen:

```
zpool attach  -o ashift=12 -f  rpool  /dev/sda2 /dev/sdb2
Make sure to wait until resilver is done before rebooting.
```

Nach erfolgreichen resilvern, sieht unser Pool nun so aus:

```
zpool status
  pool: rpool
 state: ONLINE
  scan: resilvered 31,9G in 0h2m with 0 errors on Mon Nov  5 17:02:53 2018
config:

        NAME        STATE     READ WRITE CKSUM
        rpool       ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            sda2    ONLINE       0     0     0
            sdb2    ONLINE       0     0     0

errors: No known data errors
```

Danach noch Grub installieren und fertig.

```
grub-install /dev/sdb
```

Nachdem wir dann später mal die EVO getauscht haben, wird der Pool automatisch auf die 240GB vergrößert.

Autoreplace

Autoreplace ersetzt automatische eine defekte Platte aus einem Zpool. Hierfür ist aber ein eingener Hotsparepool erforderlich.

Proxmox spezifisches

Unter Proxmox wird LVM defaultmäßing mitinstalliert, man im Installer wählen kann ob man ZFS oder EXT4 mit LVM haben möchte. Unter ZFS kann LVM einen Filter erstellen damit der HOST die LVMs von den Gästen nicht sieht. Ist sauberer.

```
nano /etc/lvm/lvm.conf
filter = [ "r|/dev/zd*|" ]
```

Auf jeden sollte auch eine Überwachung der Smart-Werte konfiguriert werden.

Proxmox Rescue ZFS

Sollte die Maschine, aus was für einen Grund auch immer nicht mehr ins System hoch booten, und auch keine Busybox zur Verfügung stehen, kann man sich mit einem auf USB installierten PVE helfen, oder auch ein anderes ZFS fähiges OS verwenden. Solaris funktioniert nicht, da Solaris eine viel zu alte Version von ZFS verwendet, und somit nicht kompatibel ist. PCBSD wurde nicht getestet, sollte aber auch funktionieren.

Hat man mit seinem PVE Stick gebootet werden sämtliche Zpools automatisch eingebunden und gemounted. Möchte man aber auf dem Rpool Dinge ändern, muss man den Mountpoint anders setzten. Da sich der Rpool ja auf „/“ mounten möchte und das natürlich nicht geht da dieser Slot schon von unserem Sticksystem besetzt ist. Um nun trotzdem auch auf diesen Teil Zugriff zu bekommen ändern wir einfach kurzfristig den Mountpoint.

```
zfs set mountpoint=/mnt rpool/ROOT/pve-1
zfs mount rpool/ROOT/pve-1
```

Wir führen unsere Änderungen durch, und hängen aus und switchen zurück.

```
zfs umount rpool/ROOT/pve-1
zfs set mountpoint=/ rpool/ROOT/pve-1
```

Natürlich bevor wir das alles erledigen möge man sich vorher erkundigen ob noch etwaige Zusatzdatasets auf Root zeigen. Diese müssen vorher ausgehängt werden.

Kompletten RPOOL mit Proxmox recovern

https://darkdevil.osit.cc/index.php/s/rWkEAEytAsmLc1r

sharenfs

Nutzt man ZFS als Dateisystem ist es klug die „sharenfs“ Funktion von ZFS direkt statt dem System Export zu verwenden. Da hier die zeitliche Abfolge beim Systemstart immer optimal ist. Um eine Freigabe zu erstellen inkl. eines Datasets zu erstellen bedient man sich folgendem Befehl:

```
zfs create testpool/testnfs -o
sharenfs="rw=@hostname1.local,rw=@192.168.1.3,no_root_squash,no_subtree_chec
k,async"
```

Für IPV6 können als Source nur mehr FQDN verwendet werden.

Bei einem bestehenden Dataset:

```
zfs set
sharenfs="rw=@hostname1.local,rw=@192.168.1.3,no_root_squash,no_subtree_chec
k,async" testpool/testnfs
```

Für eine einfache Freigabe:

```
zfs set sharenfs=on testpool/testnfs
```

Um eine Freigabe zu beenden:

```
zfs set sharenfs=off testpool/testnfs
```

Das Dataset löschen, löscht natürlich auch die Freigabe. Um zu sehen welche Freigaben nun aktiv sind gibt es mehrere Möglichkeiten. Am Host selbst:

```
cat /etc/dfs/sharetab
```

```
zfs get sharenfs # kann auch mit weiteren Optionen kombiniert werden
```

Von einem anderen Host:

```
showmount  -e hostname.local
```

Swap

Swap direkt auf ZFS erstellen. Empfohlen, genug RAM, oder Swap auf einem nicht ZFS-Filesystem.

```
zfs create -V 8G -b $(getconf PAGESIZE) -o compression=zle -o
logbias=throughput -o sync=always -o primarycache=metadata -o
secondarycache=none -o com.sun:auto-snapshot=false v-machines/swap
```

Links

- ProxmoxWIKI ZFS
- ZFS auf Ubuntu
- Corrupted device label in a replicated configuration
- Archlinux mit ZFS
- Oracle Reslivering
- ZFS Tuning

Notiz

```
zpool replace  -o ashift=9 zfs_raid <virtual device> /dev/sdd1
zdb --help
```