Contents

1 Introduction

1.1 A potted history of ZFS

Development of ZFS was started in 2001 and has been available on Solaris and OpenSolaris since 2005.

Since then there have been a number of ZFS implementations for FOSS Unix flavours - FreeBSD released a port in 2008 and it has been available as an on-and-off user-space filesystem for Linux since around 2006, and gaining fully-supported kernel-module-based filesystem since 2008.

Apple made good progress at integrating ZFS as their main filesystem for OS-X and came very close to releasing it before backing away from it in 2009 for undisclosed reasons.

FreeBSD has had the ability to boot from ZFS mirrored zpools for a few years and (although I haven't confirmed this) recent versions of Ubuntu are reported to have this ability as well.

Unfortunately a few different incompatibilities between the Solaris, IllumOS, FreeBSD and OpenZFS implementations have crept in with different extensions having the potential to make your filesystems unsupportable on other platforms, but for the most part they are quite interchangeable when these features are managed appropriately.

As of early 2019, FreeBSD has announced that they will migrate to using the OpenZFS/ZFSOnLinux port for their implementation.

Much much more detail on this is in the ZFS Wikipedia article at https://en.wikipedia.org/wiki/ZFS

1.2 ZFS Concepts

Before ZFS there were a number of different ad-hoc schemes for managing filesystems but most relied on maintaining the separation between the filesystem layer and provision of mirrored or RAID block devices.

Legacy filesystems could make use of the redundancy offered by the layer below them, but because they weren't pooled, space could not be shared between them as necessary. This space sharing issue has been mitigated somewhat by LVM as it allows sectioning disk space into volume groups and creation of logical volumes, however the filesystem atop these logical volumes are generally not capable of sharing space between themselves without administrator intervention.

ZFS instead collapses management of the filesystem with the disks in the pool and their logical layout into a single layer. ZFS employs a copy-on-write model and a meta-data model that means that creating a "dataset" (ZFS' term for a "filesystem") is cheap and gives the ability to create snapshots, clones and so on into the bargain.

It is possible to manage space use effectively with the 'quota' and 'reservation' parameters to provide similar hard partitioning of resources as required, but mostly these are not used. In addition, ZFS supports the creation and management of thin-provisioned block devices known as "zvols" which can house other binary data (such as filesystems) whilst still benefiting from the full use of snapshots and clones as for datasets.

Finally, the key feature for Solaris' maintenance process is the use of Boot Environments - this makes heavy reliance on ZFS' snapshot and cloning ability to create an offline copy of the OS that can be patched whilst the running copy of the OS is unaffected. Once complete, the administrator can opt to activate and boot into the new version with a quick fail-back if required.

This is so useful a feature, FreeBSD has also adopted this model for performing OS updates and many of the tools used there are very similar as for Solaris/IllumOS.

1.3 Why not BTRFS?

As a Solaris administrator since 1998, I've been using ZFS for long enough to be familiar with many of it's quirks and failure modes. The ZFS model makes slightly more sense to me and the tools are (IMHO) better integrated than is the case for BTRFS (at the time I was investigating it - this may have changed since).

Unfortunately I've had personal history where BTRFS has trashed all my data and I was never able to pin down the absolute cause for that at the time (again, later development may have fixed the issue) but I'm now a little gun-shy.

I also would like to retain the ability to migrate data between Linux and FreeBSD whilst keeping the door open to Solaris/IllumOS in the future.

Many of the concepts discussed here can probably be adopted with BTRFS however.

2 Project Goals

For this project we're wanting to achieve the following:

- stock Ubuntu installation
- encrypted-at-rest data
- simple backup rotation

- guard against total disk failure with redundancy
- guard against bad sectors with selective block-level redundancy
- automated snapshot management
- integration with Dropbox client
- multi-TB install with option to grow
- ability to access data without main system if required

3 Design

This design will use both drive bays in the laptop with the 500GB SDD used as the boot/OS disk and the 2TB HDD as the main data disk containing the ZFS data pool.

Backup rotation will be achieved by making the 2TB and two 4TB disks into a 3-way mirror initially, then presenting each of the 4TB disks to the system at regular intervals (alternating fortnights) to be synced with the internal 2TB drive.

All disks will use LUKS partition-level encryption with pass phrases. These will be the same across all of the disks although this is not a strict requirement.

The ZFS filesystems will use zfs-auto-snapshot to generate daily and weekly pool-wide snapshots, with a monthly retention period.

To provide support for Dropbox, a sparse 1TB zvol will be created and formatted with ext4.

Where possible, the main OS will be configured to boot without the zpool available, so that if there are issues with it, this will not prevent normal OS function.

4 Bill of Materials

- a copy of ubuntu-18.04.2-desktop-amd64.iso on DVD
- a 2012 MSI 8GB i7 something, with two 2.5in SATA bays (or similar)
- an external USB DVD/RW drive
- 1 x 500GB 2.5in SATA SDD (“sda”)
- 1 x 2TB Samsung 2.5in SATA HDD (“sdb”)
- 2 x 4TB Toshiba external USB3 HDD (“sdc”, “sdd”)
- good pass phrase for your internal disk
- (optional) a good pass phrase for your external disks

5 Method

5.1 Boot the laptop

- remove all drives except internal drives
- attach external DVD/RW drive
- boot from DVD

5.2 Install Ubuntu 18.04.2 (suggested)

- Welcome
  - Select “install ubuntu”
- Keyboard layout
  - Choose your keyboard layout as “English (Australian)”
- Wireless
  - Select “I don’t want to connect to a wi-fi network right now”
- Updates and other software
  - Select “Minimal installation”
  - Uncheck “Download updates while installing Ubuntu”
  - Uncheck “Install third-party software”

- Installation type
  - Select “Erase disk and install Ubuntu”
  - Check “Encrypt the new Ubuntu installation for security”
  - Check “Use LVM with the new Ubuntu installation”
- Choose a security key
  - Enter your internal disk pass phrase
  - (optional) Check “Overwrite empty disk space”
- Erase disk and install Ubuntu
  - Select the 500GB SDD (`sda`) to install to
- Confirm you wish to write changes to disk
- Where are you?
  - Choose your appropriate location for timezone (Melbourne)
- Who are you?
  - Create an appropriate username/hostname/password combo
  - Select “Required my password to log in”
- Install
- Installation Complete
  - Restart

5.3 Prepare OS

- provide internal disk pass-phrase for `sda5_crypt`
- log in to OS
- locate Terminal and start it
- update the package repository and install zfs-utilslinux and zfs-auto-snapshot:

```
macaque# apt-get update
Hit:1 http://au.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://au.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:3 http://au.archive.ubuntu.com/ubuntu bionic-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu bionic-security InRelease
Reading package lists... Done
macaque# apt-get install zfsutils-linux zfs-auto-snapshot
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libnvpair1linux libuutil1linux libzfs2linux libzpool2linux zfs-zed
Suggested packages:
  nfs-kernel-server samba-common-bin zfs-initramfs | zfs-dracut
The following NEW packages will be installed:
  libnvpair1linux libuutil1linux libzfs2linux libzpool2linux zfs-auto-snapshot \
zfs-zed zfsutils-linux
0 to upgrade, 7 to newly install, 0 to remove and 399 not to upgrade.
Need to get 1,176 kB of archives.
After this operation, 4,301 kB of additional disk space will be used.
Do you want to continue? [Y/n]
:
:
Setting up zfs-auto-snapshot (1.2.4-1) ...
Processing triggers for libc-bin (2.27-3ubuntu1) ...
```

5.4 Prepare internal 2TB drive

- locate the internal 2TB SDD (`sdb`)

```
macaque# ls -al /dev/disk/*/*1R8174_WCC1TT95* | cut -c40-
/dev/disk/by-id/ata-ST2000LM007-1R8174_WCC1TT95 -> ../../sdb
```

- use parted to lay out the partition on `sdb`. We also name the partition to contain the serial number of the drive for positive identification later:

```
macaque# parted /dev/disk/by-id/ata-ST2000LM007-1R8174_WCC1TT95
GNU Parted 3.2
Using /dev/sdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) mklabel gpt
Warning: The existing disk label on /dev/sdb will be destroyed and all data on \
this disk will be lost. Do you want to continue?
Yes/No? yes
(parted) unit s
(parted) mkpart 1R8174_WCC1TT95-part1 2048s 100%
(parted) print
Model: ATA ST2000LM007-1R81 (scsi)
Disk /dev/sdb: 3907029168s
Sector size (logical/physical): 512B/4096B
Partition Table: gpt
Disk Flags:

Number  Start  End          Size         File system  Name                   Flags
 1      2048s  3907028991s  3907026944s               1R8174_WCC1TT95-part1

(parted) quit
Information: You may need to update /etc/fstab.
```

- confirm that the disk partition for `sdb` is properly referenced by id and by the partition label:

```
macaque# ls -al /dev/disk/*/*1R8174_WCC1TT95* | cut -c40-
/dev/disk/by-id/ata-ST2000LM007-1R8174_WCC1TT95 -> ../../sdb
/dev/disk/by-id/ata-ST2000LM007-1R8174_WCC1TT95-part1 -> ../../sdb1
/dev/disk/by-partlabel/1R8174_WCC1TT95-part1 -> ../../sdb1
```

5.5 Prepare external 4TB drives

- attach the external 4TB USB3 disk (`sdc`) if it isn't already
- locate the external 4TB HDD (`sdc`)

```
macaque# ls -al /dev/disk/*/*20190505012566F* | cut -c40-
/dev/disk/by-id/usb-TOSHIBA_External_USB_3.0_20190505012566F-0:0 -> ../../sdc
```

- use parted to modify the partition layout on the 4TB drive (`sdc`):

```
macaque# parted /dev/disk/by-id/usb-TOSHIBA_External_USB_3.0_20190505012566F-0:0
GNU Parted 3.2
Using /dev/sdc
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) mklabel gpt
Warning: The existing disk label on /dev/sdc will be destroyed and all data on \
this disk will be lost. Do you want to continue?
Yes/No? yes
(parted) mkpart 20190505012566F-part1 2048s 3907028991s
(parted) print
Model: TOSHIBA External USB 3.0 (scsi)
Disk /dev/sdc: 4001GB
Sector size (logical/physical): 512B/4096B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name                    Flags
 1      1049kB  2000GB  2000GB               20190505012566F-part1
```

```
(parted) quit
Information: You may need to update /etc/fstab.
```

- confirm that the disk partition for `sdc` is properly referenced by id and by the partition label:

```
macaque# ls -al /dev/disk/*/*20190505012566F* | cut -c40-
/dev/disk/by-id/usb-TOSHIBA_External_USB_3.0_20190505012566F-0:0 -> ../../sdc
/dev/disk/by-id/usb-TOSHIBA_External_USB_3.0_20190505012566F-0:0-part1 -> ../../sdc1
/dev/disk/by-partlabel/20190505012566F-part1 -> ../../sdc1
```

- attach the remaining 4TB USB3 disk (`sdd`) if it isn't already
- locate the external 4TB HDD (`sdd`)

```
macaque# ls -al /dev/disk/*/*20190505012715F* | cut -c40-
/dev/disk/by-id/usb-TOSHIBA_External_USB_3.0_20190505012715F-0:0 -> ../../sdd
```

- use parted to modify the partition layout on the 4TB drive (`sdd`):

```
macaque# parted /dev/disk/by-id/usb-TOSHIBA_External_USB_3.0_20190505012715F-0:0
GNU Parted 3.2
Using /dev/sdd
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) mklabel gpt
Warning: The existing disk label on /dev/sdd will be destroyed and all data on \
this disk will be lost. Do you want to continue?
Yes/No? yes
(parted) unit s
(parted) mkpart 20190505012715F-part1 2048s 3907028991s
(parted) print
Model: TOSHIBA External USB 3.0 (scsi)
Disk /dev/sdd: 7814037164s
Sector size (logical/physical): 512B/4096B
Partition Table: gpt
Disk Flags:

Number  Start  End          Size         File system  Name                  Flags
 1      2048s  3907028991s  3907026944s               20190505012715F-part1

(parted) quit
Information: You may need to update /etc/fstab.
```

- confirm that the disk partition for `sdd` is properly referenced by id and by the partition label:

```
macaque# ls -al /dev/disk/*/*20190505012715F* | cut -c40-
/dev/disk/by-id/usb-TOSHIBA_External_USB_3.0_20190505012715F-0:0 -> ../../sdd
/dev/disk/by-id/usb-TOSHIBA_External_USB_3.0_20190505012715F-0:0-part1 -> ../../sdd1
/dev/disk/by-partlabel/20190505012715F-part1 -> ../../sdd1
```

5.6 create LUKS devices

- use cryptsetup to use the first partition for the LUKS device for 2TB SDD (`sdb`):

```
macaque# cryptsetup -v --force-password luksFormat \
  /dev/disk/by-partlabel/1R8174_WCC1TT95-part1

WARNING!
========
This will overwrite data on /dev/disk/by-partlabel/1R8174_WCC1TT95-part1 \
irrevocably.

Are you sure? (Type uppercase yes): YES
Enter passphrase for /dev/disk/by-partlabel/1R8174_WCC1TT95-part1:
Verify passphrase:
```

```
Command successful.
```

- get the UUID for the LUKS device

```
macaque# cryptsetup luksUUID /dev/disk/by-partlabel/1R8174_WCC1TT95-part1
dc99110f-a1af-49e8-a70a-fe7e2b632187
```

- open the LUKS device:

```
macaque# cryptsetup -v luksOpen /dev/disk/by-partlabel/1R8174_WCC1TT95-part1 \
  luks-dc99110f-a1af-49e8-a70a-fe7e2b632187
Enter passphrase for /dev/disk/by-partlabel/1R8174_WCC1TT95-part1:
Key slot 0 unlocked.
Command successful.
```

- do the same with the first partition for the LUKS device on each of the 4TB HDD disks (`sdc`, `sdd`)
- once complete, confirm the identity of each disk for reference:

```
macaque# cryptsetup luksUUID /dev/disk/by-partlabel/1R8174_WCC1TT95-part1
dc99110f-a1af-49e8-a70a-fe7e2b632187
macaque# cryptsetup luksUUID /dev/disk/by-partlabel/20190505012566F-part1
3088b20e-b521-43e9-8b2a-7206b8b76e82
macaque# cryptsetup luksUUID /dev/disk/by-partlabel/20190505012715F-part1
54cc0838-bf1a-4791-b3c4-44c7e543d172
```

- edit /etc/crypttab and add the following:

```
luks-dc99110f-a1af-49e8-a70a-fe7e2b632187 UUID=dc99110f-a1af-49e8-a70a-fe7e2b632187 none luks
luks-3088b20e-b521-43e9-8b2a-7206b8b76e82 UUID=3088b20e-b521-43e9-8b2a-7206b8b76e82 none luks
luks-54cc0838-bf1a-4791-b3c4-44c7e543d172 UUID=54cc0838-bf1a-4791-b3c4-44c7e543d172 none luks
```

- reboot the host normally. as it comes up, you should see:
- internal disk pass phrase challenge
- external disk pass phrase challenge (if different from internal)
- login normally
- once logged in, confirm that LUKS devices are present by their name

```
macaque# ls -1 /dev/disk/by-id/dm-name-luks-\*
/dev/disk/by-id/dm-name-luks-3088b20e-b521-43e9-8b2a-7206b8b76e82
/dev/disk/by-id/dm-name-luks-54cc0838-bf1a-4791-b3c4-44c7e543d172
/dev/disk/by-id/dm-name-luks-dc99110f-a1af-49e8-a70a-fe7e2b632187
```

5.7 create the zpool

- using the order of the devices in /etc/crypttab as a guide, create the zpool:

```
macaque# zpool create macaque mirror \
  dm-name-luks-3088b20e-b521-43e9-8b2a-7206b8b76e82 \
  dm-name-luks-54cc0838-bf1a-4791-b3c4-44c7e543d172 \
  dm-name-luks-dc99110f-a1af-49e8-a70a-fe7e2b632187
```

- confirm the status of the new pool

```
macaque# zpool status macaque
  pool: macaque
 state: ONLINE
  scan: none requested
config:

        NAME                                            STATE     READ WRITE CKSUM
        macaque                                         ONLINE       0     0     0
          mirror-0                                      ONLINE       0     0     0
            dm-name-luks-3088b20e-b521-43e9-8b2a-7206b8b76e82  ONLINE       0     0     0
```

```
            dm-name-luks-54cc0838-bf1a-4791-b3c4-44c7e543d172  ONLINE       0     0     0
            dm-name-luks-dc99110f-a1af-49e8-a70a-fe7e2b632187  ONLINE       0     0     0

errors: No known data errors
```

- edit `/etc/default/zfs` (wrapped here to fit - don't do this yourself):

```
#ZPOOL_IMPORT_PATH="/dev/disk/by-vdev:/dev/disk/by-id"
ZPOOL_IMPORT_PATH="\
  /dev/disk/by-id/dm-name-luks-3088b20e-b521-43e9-8b2a-7206b8b76e82\
  :/dev/disk/by-id/dm-name-luks-54cc0838-bf1a-4791-b3c4-44c7e543d172\
  :/dev/disk/by-id/dm-name-luks-dc99110f-a1af-49e8-a70a-fe7e2b632187"
```

5.8 confirm configuration

- reboot the host normally, as it comes up, you should see
- internal disk pass phrase challenge
- external disk pass phrase challenge (if different from internal)
- login normally
- once logged in, confirm that the zpool is imported successfully as previously

5.9 configure for internal disks only

- re-edit /etc/crypttab, comment out entries the external drives - this prevents your laptop waiting 3min for these to become available

```
luks-dc99110f-a1af-49e8-a70a-fe7e2b632187 UUID=dc99110f-a1af-49e8-a70a-fe7e2b632187 none luks
# luks-3088b20e-b521-43e9-8b2a-7206b8b76e82 UUID=3088b20e-b521-43e9-8b2a-7206b8b76e82 none luks
# luks-54cc0838-bf1a-4791-b3c4-44c7e543d172 UUID=54cc0838-bf1a-4791-b3c4-44c7e543d172 none luks
```

- offline the relevant vdevs from the pool and their LUKS device:

```
macaque# zpool offline macaque dm-name-luks-54cc0838-bf1a-4791-b3c4-44c7e543d172
macaque# cryptsetup -v luksClose luks-54cc0838-bf1a-4791-b3c4-44c7e543d172
Command successful.
macaque# zpool offline macaque dm-name-luks-dc99110f-a1af-49e8-a70a-fe7e2b632187
macaque# cryptsetup -v luksClose luks-dc99110f-a1af-49e8-a70a-fe7e2b632187
Command successful.
```

- you may now physically remove the external 4TB HDDs
- confirm that the zpool is still functional, but in a degraded state:

```
macaque# zpool status macaque
  pool: macaque
 state: DEGRADED
status: One or more devices has been taken offline by the administrator.
        Sufficient replicas exist for the pool to continue functioning in a
        degraded state.
action: Online the device using 'zpool online' or replace the device with
        'zpool replace'.
  scan: none requested
config:
        NAME                                                STATE     READ WRITE CKSUM
        macaque                                             DEGRADED     0     0     0
          mirror-0                                          DEGRADED     0     0     0
            dm-name-luks-3088b20e-b521-43e9-8b2a-7206b8b76e82  ONLINE       0     0     0
            dm-name-luks-54cc0838-bf1a-4791-b3c4-44c7e543d172  OFFLINE      0     0     0
            dm-name-luks-dc99110f-a1af-49e8-a70a-fe7e2b632187  OFFLINE      0     0     0
```

```
errors: No known data errors
```

- reboot the host normally, as it comes up, you should see
- internal disk pass phrase challenge
- no external disk pass phrase challenge (if it is different from internal)
- login normally
- once logged in, confirm that the zpool is imported, albeit with degraded performance as before

6 Attach/detach process

For the following examples, disk `20190505012715F` has been attached as `sdc`

Also, the git repo contains sample `import.sh` and `export.sh` scripts which may be useful to automate this process, however you will need to customise `data/config` for your devices

6.1 The attach process

- physically re-attach one of the external USB3 4TB HDDs
- udev will detect the device and modify /dev/disk/* as appropriate
- gnome will prompt for the passphrase, or you can manually supply it from the command-line:

```
macaque# cryptsetup luksUUID /dev/disk/by-partlabel/20190505012715F-part1
4e51b754-0618-43e7-a1ea-def5ffe440a0
macaque# cryptsetup luksOpen /dev/disk/by-partlabel/20190505012715F-part1 \
  luks-4e51b754-0618-43e7-a1ea-def5ffe440a0
Enter passphrase for /dev/disk/by-partlabel/20190505012715F-part1:
```

- once LUKS has the partition open successfully, attach the disk to the zpool:

```
macaque# zpool online macaque dm-name-luks-4e51b754-0618-43e7-a1ea-def5ffe440a0
```

- once the zpool vdev is made online, the zpool will immediately start resilvering

6.2 Monitor the zpool resync process

Depending on the differences between disks and the speed of the I/O path for your laptop, it make take quite some time for a disk to come back into sync with the rest of the pool. This process is known as "resilvering" in ZFS parlance.

- once-off status can be had with:

```
    macaque# zpool status
  pool: macaque
 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 Fri Aug 16 19:11:44 2019
        13.5G scanned out of 1.14T at 11.2M/s, 29h17m to go
        13.5G resilvered, 1.15% done
config:

        NAME                                              STATE     READ WRITE CKSUM
        macaque                                           DEGRADED     0     0     0
          mirror-0                                        DEGRADED     0     0     0
            dm-name-luks-609aa987-db4f-49cf-ade3-5aa437bedccd  ONLINE       0     0     0
            dm-name-luks-412821db-f812-4fdb-883a-3f6c567446ce  ONLINE       0     0     0  (resilvering)
            dm-name-luks-4e51b754-0618-43e7-a1ea-def5ffe440a0  OFFLINE      0     0     0
```

```
errors: No known data errors
```

- continuous performance stats at a pool level (10sec intervals):

```
macaque# zpool iostat 10
              capacity     operations     bandwidth
pool        alloc   free   read  write   read  write
----------  -----  -----  -----  -----  -----  -----
macaque     1.35T   477G      1      2   141K   171K
macaque     1.35T   477G      2     15  25.6K   264K
macaque     1.35T   477G      3      0  25.6K      0
macaque     1.35T   477G      2     12  19.2K   233K
macaque     1.35T   477G      2      0  20.0K      0
:
:
```

- continuous per-vdev performance (10sec intervals):

```
macaque# zpool iostat -v 10
                                                      capacity     operations     bandwidth
pool                                                alloc   free   read  write   read  write
--------------------------------------------------  -----  -----  -----  -----  -----  -----
macaque                                             1.35T   477G      1      2   141K   171K
  mirror                                            1.35T   477G      1      2   141K   171K
    dm-name-luks-609aa987-db4f-49cf-ade3-5aa437bedccd     -      -      1      0   133K  22.6K
    dm-name-luks-412821db-f812-4fdb-883a-3f6c567446ce     -      -      0      0      0      0
    dm-name-luks-4e51b754-0618-43e7-a1ea-def5ffe440a0     -      -      0      1  7.35K   148K
--------------------------------------------------  -----  -----  -----  -----  -----  -----
:
:
```

- during this time, your system will be resilvering the zpool in the background - you can use your laptop normally
- it is generally a good idea to let the resilvering complete before shutting down or rebooting

6.3 Exercise the export process

- use cryptsetup to confirm the LUKS device we're removing:

```
macaque# cryptsetup luksUUID /dev/disk/by-partlabel/20190505012715F-part1
4e51b754-0618-43e7-a1ea-def5ffe440a0
```

- confirm the state of the zpool:

```
macaque# zpool status
:
:
        NAME                                                STATE     READ WRITE CKSUM
        macaque                                             DEGRADED     0     0     0
          mirror-0                                          DEGRADED     0     0     0
            dm-name-luks-609aa987-db4f-49cf-ade3-5aa437bedccd  ONLINE       0     0     0
            dm-name-luks-412821db-f812-4fdb-883a-3f6c567446ce  OFFLINE      0     0     0
            dm-name-luks-4e51b754-0618-43e7-a1ea-def5ffe440a0  ONLINE       0     0     0

errors: No known data errors
```

- use zpool offline to remove the LUKS device from the zpool:

```
macaque# zpool offline macaque dm-name-luks-4e51b754-0618-43e7-a1ea-def5ffe440a0
```

- zpool status

```
macaque# zpool status
:
:
 NAME                                                STATE     READ WRITE CKSUM
```

```
        macaque                                                 DEGRADED     0     0     0
          mirror-0                                              DEGRADED     0     0     0
            dm-name-luks-609aa987-db4f-49cf-ade3-5aa437bedccd   ONLINE       0     0     0
            dm-name-luks-412821db-f812-4fdb-883a-3f6c567446ce   OFFLINE      0     0     0
            dm-name-luks-4e51b754-0618-43e7-a1ea-def5ffe440a0   OFFLINE      0     0     0
```

- close the LUKS device:

  ```
  macaque# cryptsetup luksClose /dev/disk/by-id/dm-name-luks-4e51b754-0618-43e7-a1ea-def5ffe440a0
  ```

- you may now physically remove the external disk

7 Recovering from bad detaches

This model of disk management does have a weakness in that it requires proper removal of the disk from the zpool and proper closing of the LUKS crypt devices or the zpool will get into a state that it may require a little bit of work to recover.

Typically these are not serious, however they may prompt the zpool to perform a whole-pool resilver which may take an extended length of time to complete.

Here are some common scenarios where this may occur and their expected mechanism for recovery.

7.1 Inadvertent shutdown of laptop whilst connected

- zpool status will list the now-missing device(s) as "UNAVAIL" and present their zpool vdev ID, plus some notes as what these were known as last import (wrapped for space):

  ```
  macaque# zpool status
    pool: macaque
   state: DEGRADED
  status: One or more devices could not be used because the label is missing or
          invalid.  Sufficient replicas exist for the pool to continue
          functioning in a degraded state.
  action: Replace the device using 'zpool replace'.
     see: http://zfsonlinux.org/msg/ZFS-8000-4J
    scan: resilvered 1.10M in 0h0m with 0 errors on Sun Sep  1 15:54:12 2019
  config:

          NAME                                                  STATE     READ WRITE CKSUM
          macaque                                               DEGRADED     0     0     0
            mirror-0                                            DEGRADED     0     0     0
              dm-name-luks-609aa987-db4f-49cf-ade3-5aa437bedccd  ONLINE       0     0     0
              dm-name-luks-412821db-f812-4fdb-883a-3f6c567446ce  OFFLINE      0     0     0
              12957871109559519687                              UNAVAIL      0     0     0 \
                was /dev/disk/by-id/dm-name-luks-4e51b754-0618-43e7-a1ea-def5ffe440a0

  errors: No known data errors
  ```

- physically re-attach the missing disk (if necessary)
- udev will detect the device and modify /dev/disk/* as appropriate
- gnome will prompt for the passphrase, or you can manually supply it from the command-line:

  ```
  macaque# cryptsetup luksUUID /dev/disk/by-partlabel/20190505012715F-part1
  4e51b754-0618-43e7-a1ea-def5ffe440a0
  macaque# cryptsetup luksOpen /dev/disk/by-partlabel/20190505012715F-part1 \
    luks-4e51b754-0618-43e7-a1ea-def5ffe440a0
  Enter passphrase for /dev/disk/by-partlabel/20190505012715F-part1:
  ```

- once LUKS has the partition open successfully, clear the error and allow the zpool to resilver

  ```
  macaque# zpool clear macaque
  ```

- confirm the status of the zpool:

```
macaque# zpool status
  pool: macaque
 state: DEGRADED
status: One or more devices has been taken offline by the administrator.
        Sufficient replicas exist for the pool to continue functioning in a
        degraded state.
action: Online the device using 'zpool online' or replace the device with
        'zpool replace'.
  scan: resilvered 4.45M in 0h0m with 0 errors on Sun Sep  1 17:07:40 2019
config:

        NAME                                                STATE     READ WRITE CKSUM
        macaque                                             DEGRADED     0     0     0
          mirror-0                                          DEGRADED     0     0     0
            dm-name-luks-609aa987-db4f-49cf-ade3-5aa437bedccd  ONLINE       0     0     0
            dm-name-luks-412821db-f812-4fdb-883a-3f6c567446ce  OFFLINE      0     0     0
            dm-name-luks-4e51b754-0618-43e7-a1ea-def5ffe440a0  ONLINE       0     0     0

errors: No known data errors
```

7.2 Inadvertent physical removal of device whilst open

This is potentially more of an issue for the pool as there may be data in flight that wasn't committed, however the recovery process is very similar to that above.

- zpool status will list the now-missing device(s) as "FAULTED" and the status of the vdev

```
macaque# zpool status
  pool: macaque
 state: DEGRADED
status: One or more devices are faulted in response to persistent errors.
        Sufficient replicas exist for the pool to continue functioning in a
        degraded state.
action: Replace the faulted device, or use 'zpool clear' to mark the device
        repaired.
  scan: resilvered 4.45M in 0h0m with 0 errors on Sun Sep  1 17:07:40 2019
config:

        NAME                                                STATE     READ WRITE CKSUM
        macaque                                             DEGRADED     0     0     0
          mirror-0                                          DEGRADED     0     0     0
            dm-name-luks-609aa987-db4f-49cf-ade3-5aa437bedccd  ONLINE       0     0     0
            dm-name-luks-412821db-f812-4fdb-883a-3f6c567446ce  OFFLINE      0     0     0
            dm-name-luks-4e51b754-0618-43e7-a1ea-def5ffe440a0  FAULTED      0   198     0  \
          too many errors

errors: No known data errors
```

- if the host has rebooted in the interim, then the output will look similar with the missing disk marked as "UNAVAIL"
- if the disk is not actually failed, but is just ready to be re-attached and synced, follow the process as per the previous section

7.3 replacing a failed disk

XXXXXX

8 zfs dataset creation

- create macaque/home/mjch filesystem

```
macaque# zfs create -p macaque/home/mjch
macaque# chown -R mjch:mjch /macaque/home/mjch
```

9 zfs snapshot creation

XXXXXX

9.1 manually creating/destroying snapshots

9.2 zfs-auto-snapshot management

There's not much to do here - installing it is all that's required.

By default zfs-auto-snapshot will use crontab entries to create snapshots at 15min, hourly, daily, weekly and monthly intervals:

/etc/cron.d/zfs-auto-snapshot "frequent" snapshots, occurring every 15min
/etc/cron.hourly/zfs-auto-snapshot hourly snapshots
/etc/cron.daily/zfs-auto-snapshot daily snapshots
/etc/cron.weekly/zfs-auto-snapshot weekly snapshots
/etc/cron.monthly/zfs-auto-snapshot monthly snapshots

XXXXXX

9.3 zfs clones

XXXXXX

9.4 zfs clone promotion

XXXXXX

10 Dropbox client setup

For their own arbitrary reasons, Dropbox have decided to only support ext4 on Linux, so we need to create an ext4 filesystem inside a ZFS zvol, as below. All of these examples use my home directory and username - yours will likely be different.

- create a dataset to house the zvol, but make it unmountable - it's just a container

```
macaque# zfs create -o canmount=off macaque/home/mjch/zvol
```

- create the zvol as a sparse 1TB volume

```
macaque# zfs create -V 1T -s macaque/home/mjch/zvol/dropbox
```

- format the zvol device

```
macaque# mkfs -t ext4 /dev/zvol/macaque/home/mjch/zvol/dropbox
mke2fs 1.44.1 (24-Mar-2018)
Discarding device blocks: done
Creating filesystem with 268435456 4k blocks and 67108864 inodes
Filesystem UUID: f20d5da2-d394-41db-b2a8-2564e121c755
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
        4096000, 7962624, 11239424, 20480000, 23887872, 71663616, 78675968,
        102400000, 214990848
```

```
Allocating group tables: done
Writing inode tables: done
Creating journal (262144 blocks): done
Writing superblocks and filesystem accounting information: done
```

- we'll mount this at /media/mjch/dropbox

```
macaque# mkdir /media/mjch/dropbox
macaque# chown mjch:mjch /media/mjch/dropbox
```

- (optional) add something similar to the following to your /etc/fstab - caution: this may allow ANY user to mount this volume:

```
/dev/zvol/macaque/home/mjch/zvol/dropbox /media/mjch/dropbox ext4 noauto,nofail,user 0 0
```

- once mounted, make sure permissions are set correctly

```
macaque# mkdir /media/mjch/dropbox/Dropbox
macaque# chown -R mjch:mjch /media/mjch/dropbox/
```

- install the Dropbox client and hook it up to your account - caution: the Dropbox client will start syncing immediately
- navigate through the Dropbox client preferences to the "Sync" panel and change the folder location to /media/mjch/dropbox/Dropbox

XXXXXX

11 operational tasks

11.1 reading zpool disk in another host safely

- attach the external disk
- Ubuntu should automatically detect the LUKS partition and prompt for the passphrase
- use zdb to determine where the disk may have come from

XXXXXX

- import the zpool by name, making sure to assert the 'readonly' parameter:

```
nirgal# zpool import -o readonly=on macaque
nirgal#
```

- confirm the status of the zpool:

```
nirgal# zpool status macaque
  pool: macaque
 state: DEGRADED
status: One or more devices could not be used because the label is missing or
        invalid.  Sufficient replicas exist for the pool to continue
        functioning in a degraded state.
action: Replace the device using 'zpool replace'.
   see: http://zfsonlinux.org/msg/ZFS-8000-4J
  scan: none requested
config:

        NAME                                                STATE     READ WRITE CKSUM
        macaque                                             DEGRADED     0     0     0
          mirror-0                                          DEGRADED     0     0     0
            8752437891776700926                             UNAVAIL      0     0     0  \
              was /dev/disk/by-id/dm-name-luks-3088b20e-b521-43e9-8b2a-7206b8b76e82
            dm-name-luks-54cc0838-bf1a-4791-b3c4-44c7e543d172  ONLINE       0     0     0
            13794551325672317156                            UNAVAIL      0     0     0  \
              was /dev/disk/by-id/dm-name-luks-dc99110f-a1af-49e8-a70a-fe7e2b632187
```

```
errors: No known data errors
nirgal#
```

- when done, export the pool

```
nirgal# zpool export macaque
nirgal#
```

- close the LUKS device

```
nirgal# cryptsetup luksClose luks-54cc0838-bf1a-4791-b3c4-44c7e543d172
nirgal#
```

11.2 adding/removing disks from the pool

XXXXXX

11.3 increasing the size of the pool

XXXXXX

11.4 making and restoring from an offline backup

The design described here is more of a "nearline" model - there is no real "making a backup" process discussed. ZFS does support this model at a zpool level using `zfs send` and `zfs receive`.

In addition, because this process produces a file stream it is possible to manage that as one would for a more traditional backup, such as a archive, for example.

Unfortunately, in order to get at the content of the zfs stream, it needs to be "received" into a zpool that is configured for it - there is no mechanism to to get random access to individual files as one might with a tar or zip archive or a traditional UFS dump.

The benefit however, is that these zfs streams understand all of the ZFS data models, so the backup will contain datasets, snapshots, zvols and so on.

XXXXXX

[comment]: [//] vim: tabstop=8 softtabstop=0 expandtab shiftwidth=4 smarttab