Example of Mounting a ZFS Zpool as an NFS Filesystem

Objectives: To create a zpool on a Linux Mint 19 server that can be mounted on and shared by a client machine on the same network, using the Network File System. Subsequently, to mirror that zpool on the server so that a degree of redundancy is achieved on that machine.

Pre-Requisites: Completion of Chapter W22. In particular, the following printed book, and Examples and sub-sections of Chapter W22, are essential references for what is done here-

Appendix A, Section W26.4.7, ZFS Installation, Example W26.8a also applicable to Linux Mint 19

Example W22.3, Mirroring of Physical User Data Hard Disks

Example W22.5, Creating a ZFS Pool on a USB thumbdrive

Section W22.3.2 ZFS Storage Poolsand the **zpool** Command, items 11 and 12.

Section W22.3, ZFS File System Commands and the **zfs** Command , sub-section W22.3.3, items 6 and 7.

It is assumed that you have installed ZFS on your system, per the Appendix A instructions.

On our Linux Mint 19 system, the following is also true in the code below-

The logical device names of the USB thumbdrives we have on the system are /dev/sdc and /dev/sdd

The IP address of the server on the LAN is 192.168.0.25

The IP address of the client on the LAN is 192.168.0.31

The same user, with superuser privilege, was on both systems.

Background and Discussion:

Why would you want to share user data, particularly ZFS data sets, using the Network File System (NFS)? Traditionally NFS has the great utility of allowing users on multiple “client” systems to easily and securely share diverse file system types and their data that are contained on a “server” machine. We extend this model by showing you how to achieve the same sharing in a simple manner using the **zfs sharenfs** sub-command. Sharing file systems in this way gives you the expeditious advantage of applying all of the ZFS functionality to the shared directories and files, and in particular, having ZFS mirroring redundancy on the server-side files. Here we show a simple method of sharing a zpool on the server with a client machine on the same Intranet that mounts that zpool. Note that the client machine does not support or have ZFS installed! Users on the client machine can still retrieve and add files from and to the remotely mounted zpool.

Procedures: Do the following steps, in the order presented, to fulfill the requirements of this example. We give brief explanations of what each of the procedural steps accomplish.

*On the server-*

1. Create a zpool named “rasp”, that will contain the file system we want to share with the client. This zpool’s vdev is /*dev/*sdc -

$ **sudo zpool create -f rasp /dev/sdc**

Please note here that we are creating the ZFS zpool on a vdev that is NOT the system disk!

2. ZFS automatically mounts the file system for the zpool at /rasp. Change the current working directory to /rasp, and create a file in that file system-

$ **cd /rasp**

$ **sudo touch test**

3. Install the nfs-kerenel-server service with the following command, which on our Linux Mint 19 system was not done by default.

$ **sudo apt-get install -y nfs-kernel-server**

4. The critical step. Use the **zfs sharenfs** command to establish the source zpool that will be shared with the client.

$ **sudo zfs set sharenfs="rw=192.168.0.0/24,async" rasp**

5. Make the zpool rasp a mirror, with /dev/sdc and /dev/sdd as the two vdevs in the mirror.

$ **sudo zpool attach -f rasp /dev/sdc /dev/sdd**

6. Change the ownership and group of the file “test”, and the directory “rasp” to the ordinary user bob. This allows bob to put files into the directory /rasp and access them.

$ **sudo chown bob test**

$ **sudo chgrp bob test**

$ **cd ..**

$ **sudo chown bob rasp**

$ **sudo chgrp bob rasp**

*On the client-*

7. If your client machine is running Linux Mint 19, then you need to do three things. First, use the apt package manager to download the nfs-common package. Second, create a directory where the server file system will be mounted. And finally, use the **mount** command to mount the server file system.

$ **sudo apt-get install nfs-common**

$ **sudo mkdir /mnt/rasp**

$ **sudo mount -t nfs 192.168.0.25:/rasp /mnt/rasp**

7Rpi. For a Raspberry Pi 3B running Raspbian Linux for example, the only things that have to be done on the client machine is create a directory mount point, and then mount the zpool file system from the server at IP address 192.168.0.25:/rasp at a location in the Raspbian file system /mnt/rasp

$ **sudo mkdir /*mnt/*rasp**

$ **sudo mount -t nfs 192.168.0.25:/rasp /mnt/rasp**

8. You also want to make the following addition to your /etc/fstab file on the client so that the shared zpool auto-mounts on boot-

192.168.0.25:/rasp /mnt/rasp nfs auto 0 0

8Rpi. A caveat here for Rpi users, on our Raspbian system, we also had to use the following command to edit the raspi-config file, and have the boot proceed by waiting for the network to come up first-

$ **sudo raspi-config**

It is now possible to add files to the server directory /rasp, and they will appear in /mnt/rasp on the client. In addition, every file added to /rasp on the server is mirrored onto both USB thumbdrives.

Going Further:

We present two possible extensions to the above Procedures. First, a contingency that would be encountered over the long term deployment of the model we present here, and then a way of increasing system security using zfs datasets.

*Contingency-* What do you do if the server, the client, or both, need to be rebooted? The easiest and most straight forward way of maintaining the NFS sharing of the zpool created above would be to use the **zpool export** and **zpool import** commands on the server to take the pool offline, reboot, and then bring the pool back on line. The following commands achieve this on the example server from the above Procedures.

$ **sudo zpool export -f rasp**

$ **sudo reboot**

After rebooting-

$ **sudo zpool import rasp**

On the client, nothing needs to be done after rebooting if you have modified your systems /*etc/fstab* file as shown above in step 8. of the Procedures.

*Increasing System Security-* It is possible, and also very effective, to not only share an entire zpool, as we have done above, between a server and possibly multiple clients, but to share selected datasets within a zpool. The datasets, implemented on the server using the **zfs create** command, are basically individual filesystems, with all of the attributes that you want them to have controlled using the sub-commands and options of the **zfs** command. This combination of ZFS and NFS is a valuable security feature. The following general syntax for achieving this dataset sharing is as follows-

On server-

**zfs set sharenfs=on zpool\_name/name\_of\_dataset**

On client(s)-

**mount -t nfs IP\_of\_server:/zpool\_name/name\_of\_dataset /path\_on\_client**

Modifying access permissions using the standard Linux model ensures dataset security. Additionally, the advantages of applying NFSv4 ACL’s to both directories and files on both server and client (if those systems support them), allow an administrator to exercise a much finer grained control over access permissions than the traditional Linux permissions. We show some extended examples of applying NFSv4 ACL’s in Chapter W26.