Table of Contents

Building a FreeNAS replication target on an Odroid HC2

THIS IS A VERY ROUGH WORK IN PROGRESS. DON'T RELY ON THIS FOR ANYTHING OTHER THAN ENTERTAINMENT. IT WILL EAT YOUR DATA, KICK YOUR DOG, AND POSSIBLY BURN YOUR HOUSE DOWN

The Odroid HC2 is a single-board computer that comes equipped with a SATA interface and a heatsink designed to mount a hard drive. It's designed to serve as a simple home NAS, and a version of OpenMediaVault is available specifically for the HC2. However, the board runs Linux, and there's also an OS image of Ubuntu 18.04 available. Since Ubuntu supports ZFS pretty well, it should be trivial to set up one of these to serve as a replication target for a FreeNAS system. Or so I thought.

Unfortunately, the Ubuntu 18.04 image uses a 32-bit kernel, and the available ZFS packages aren't compatible with a 32-bit kernel. This means I'll need to compile the ZFS pieces myself, and they're not known to be stable with a 32-bit kernel in any event. Updates to come.

Create a non-privileged user with sudo capability

```
adduser fred
usermod -aG sudo fred
```

Rebuild the kernel

In order to build ZFS, you must have the headers for the running kernel installed on the system. Since there does not appear to be a linux-headers package available matching the kernel version installed on the Odroid, this means you'll need to rebuild the kernel. To do this, run the following commands (taken from the Hardkernel wiki):

```
sudo apt update && sudo apt upgrade && apt dist-upgrade && apt autoremove
sudo apt install git gcc g++ build-essential bc libssl-dev
git clone --depth 1 https://github.com/hardkernel/linux -b odroidxu4-4.14.y
cd linux
make odroidxu4_defconfig
make -j8
sudo make modules_install
sudo cp -f arch/arm/boot/zImage /media/boot
sudo cp -f arch/arm/boot/dts/exynos5422-odroidxu3.dtb /media/boot
sudo cp -f arch/arm/boot/dts/exynos5422-odroidxu4.dtb /media/boot
sudo cp -f arch/arm/boot/dts/exynos5422-odroidxu3-lite.dtb /media/boot
sudo cp .config /boot/config-`make kernelrelease`
```

```
sudo update-initramfs -c -k `make kernelrelease`
sudo mkimage -A arm -O linux -T ramdisk -C none -a 0 -e 0 -n uInitrd -d
/boot/initrd.img-`make kernelrelease` /boot/uInitrd-`make kernelrelease`
sudo cp /boot/uInitrd-`make kernelrelease` /media/boot/uInitrd
sync
```

Then reboot your system to start using the new kernel.

Build ZFS on Linux

These instructions are taken from the ZFSonLinux Wiki. First, install the necessary dependencies:

```
sudo apt install autoconf libtool gawk alien fakeroot zlib1g-dev uuid-dev
libattr1-dev libblkid-dev libselinux-dev libudev-dev parted lsscsi ksh libssl-
dev libelf-dev
```

Then download, build, and install the ZFS code:

```
git clone https://github.com/zfsonlinux/zfs
cd zfs
git checkout master
sh autogen.sh
./configure
make -s -j$(nproc)
sudo make install
```

Then load the ZFS modules:

```
sudo modprobe zfs
```

Create the pool

Create your pool. Make sure to set `ashift=12`.

```
zpool create -o ashift=12 dozer /dev/disk/by-id/ata-
WDC_WD80EMAZ-00M9AA0_VAGA2PLD
```

Enable encryption, and create an encrypted dataset

The idea of this system is to be a standalone storage “brick”, which could be left at a remote location where you might not fully trust the network operator. ZFS on Linux supports dataset encryption for this

purpose, and material for this section is drawn from this blog post. You'll first need to enable that feature on your pool:

```
zpool set feature@encryption=enabled dozer
```

Then, create the encrypted dataset:

```
zfs create -o encryption=on -o keylocation=prompt -o keyformat=passphrase
dozer/encrypted
```

The system will prompt you for a passphrase, which you'll need whenever you mount that dataset. Minimum length is eight characters.

Create a replication user

For the sake of security, it would be best if replication to this device ran as a user other than root. First, create a user in the FreeNAS web GUI called `zfsuser`. Note the numeric userid for that user.

Then, on the Odroid, as root, run

```
adduser zfsuser -u userid -s /bin/false
```

where “userid” is the numeric user ID noted on the FreeNAS box.

Now allow that user to make changes on the encrypted dataset:

```
zfs allow -ldu zfsuser
create,destroy,diff,mount,readonly,receive,release,send,userprop dozer/backup
```

Install Zerotier

Zerotier will create an encrypted virtual network connection between your Odroid and your FreeNAS box. It's installed by default on FreeNAS, but you'll need to install it on the Odroid. Run these commands:

```
sudo apt install curl
curl https://install.zerotier.com | sudo bash
```