2019

FreeNAS + QuadstorVTL

A PROOF OF PRACTICE

DAVID SOBEL

Table of Contents

Introduction

This paper will detail the process of setting up and utilizing a Network Attached Storage (NAS) backup system supplemented by magnetic tape backups. As the intention here is to provide a proof of practice, all processes are being run on virtual machines (VMs) to remove the limitation of cost and setup time. This paper details the setup process and execution of a NAS backup system with an additional layer of backup storage in the form of tape backups. Some benefits of tape include a strong cost to storage ratio and easy backup and recovery processes due to the linear nature of tape backups (Hope, 2018). The example setup in this paper does not use real tape drives, but the benefits of combining FreeNAS with tape-like functionality will become apparent.

Setup Processes

At its most basic, the virtual NAS and tape library setup requires two things: a VM running FreeNAS, an open source and free operating system for running a NAS system (iXsystems, 2019), a VM running Debian 7 or 8 which will act as the tape library, and data that needs to be backed up. We will go through the setup for each system individually, then demonstrate it in action.

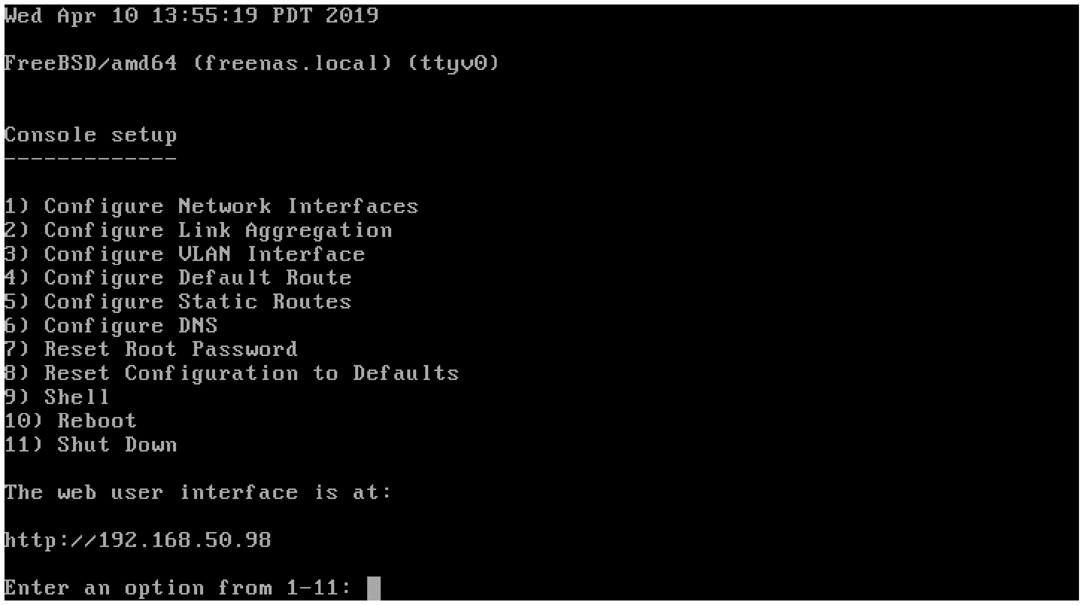
FreeNAS Setup

FreeNAS is a distribution of FreeBSD made by iXSystems, designed specifically around file sharing, among other services (iXsystems, 2019), that can easily be installed and run in a virtual machine (iXsystems, 2015). This section will detail the installation process, the baseline setup for allowing multiple users to have access to their own backup directory, and adding QuadstorVTL as an iSCSI target for FreeNAS.

The latest version of FreeNAS as of writing is 11.2-U3, which is the version used in this test setup. The installation process is as follows for VirtualBox:

1. Set up a FreeBSD preconfigured VM (64-bit as FreeNAS is suggested to run with 8GB of RAM (iXsystems, 2019)) with a drive of any size
2. Add an additional SATA or SCSI drive in the virtual machine settings of any size
3. Set the VM to bridged mode (or any private network you have set up)

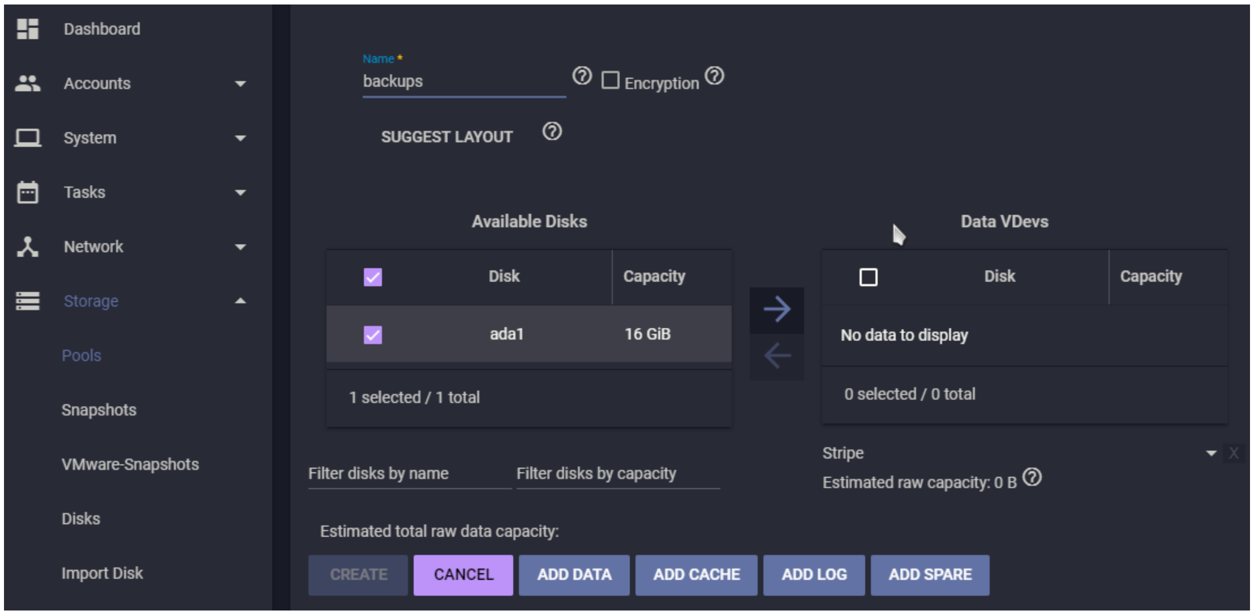
4. Boot the FreeNAS disk image and run the text-based installation. After the installation completes and the VM is rebooted, you will be greeted by a menu with 11 options:



```
Wed Apr 10 13:55:19 PDT 2019

FreeBSD/amd64 (freenas.local) (ttyv0)


Console setup
-------------

1) Configure Network Interfaces
2) Configure Link Aggregation
3) Configure VLAN Interface
4) Configure Default Route
5) Configure Static Routes
6) Configure DNS
7) Reset Root Password
8) Reset Configuration to Defaults
9) Shell
10) Reboot
11) Shut Down

The web user interface is at:

http://192.168.50.98

Enter an option from 1-11:
```

Figure 1 FreeNAS Post-Install

For the moment, if you go to the machine's IP address, you can access the web GUI, which is where most backup management will take place in the test setup. The first place you should go after logging in with the root account is the "Storage" tab in the sidebar then the "Pools" subsection. The reason for adding the second disk in the setup process is because the boot drive (ada0 in our case) cannot be used in a pool due to being the drive FreeNAS runs on. In here you should see the option to add ada1 to a pool:



Name * backups ☐ Encryption

Available Disks

✓	Disk	Capacity
✓	ada1	16 GiB

1 selected / 1 total

Data VDevs

☐	Disk	Capacity
	No data to display	

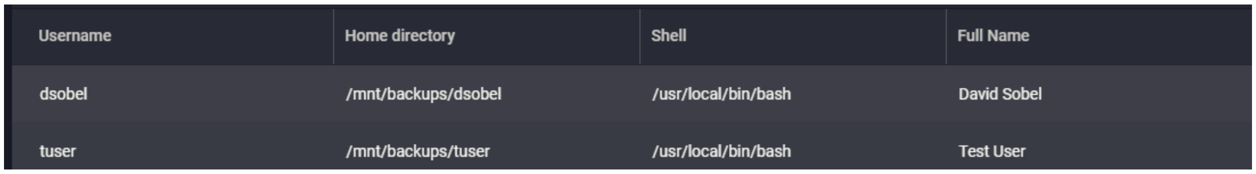
0 selected / 0 total

Filter disks by name
Filter disks by capacity
Stripe
Estimated raw capacity: 0 B
Estimated total raw data capacity:

Figure 2 Creating a storage pool

In our example the pool has been called backups and only contains the ad1 disk. Clicking the right arrow adds the drive to the list of drives to be pooled, thus allowing you to click create.

Once you have created the pool, create a couple users in the "Accounts" tab in the sidebar such as the ones in the example here:

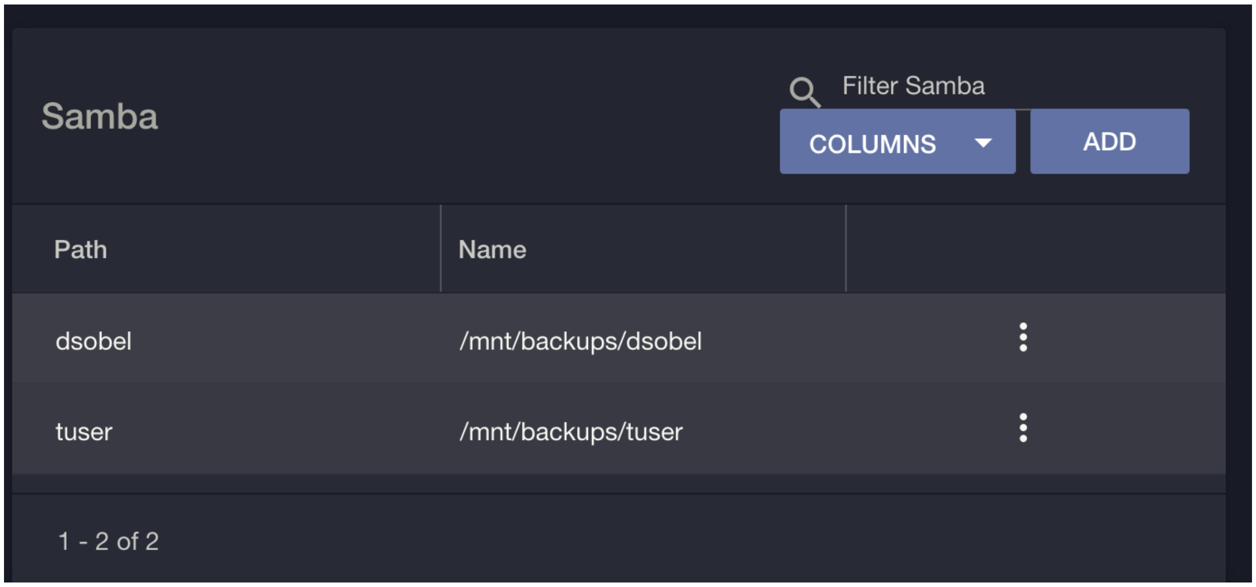


Username	Home directory	Shell	Full Name
dsobel	/mnt/backups/dsobel	/usr/local/bin/bash	David Sobel
tuser	/mnt/backups/tuser	/usr/local/bin/bash	Test User

Figure 3 Newly created users dsobel and tuser

You can see that the home directories have been set to ones with respective names. Since a goal of this setup was for individual users to be able to back up their files to a designated directory, the permissions have been set to 755, or full access for the user and read for everyone else.

With users set up with their own home directories, we can set up network shares to allow each user to access their own files. The example setup uses Windows SMB type shares. To enable SMB, go to the "Services" tab, then enable SMB in the list, making sure to set values for "NetBIOS Name" and "NetBIOS Alias" to make it easier to access the shares. Next, go to the "Sharing" tab, go to "Windows (SMB) Shares," and click add. The paths you set will follow the general format of /mnt/poolname/*, which is demonstrated in figure 4.

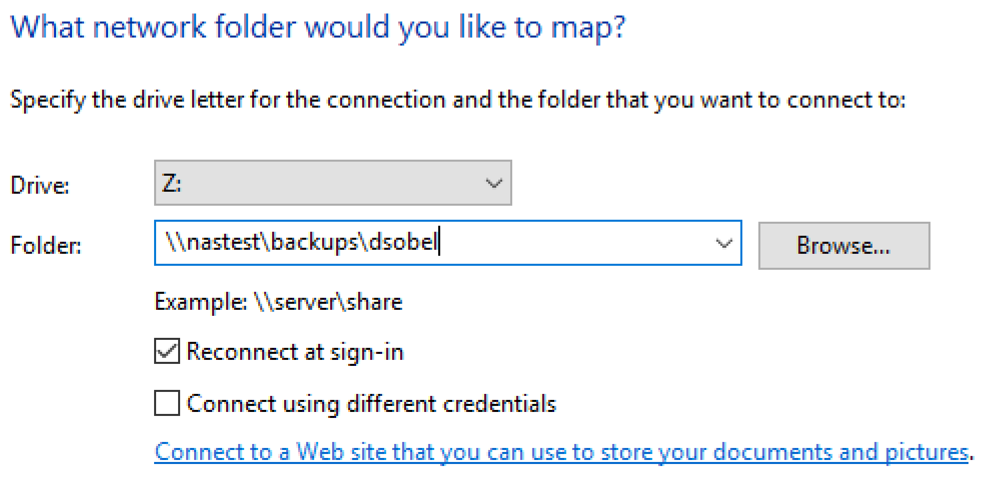


Path	Name
dsobel	/mnt/backups/dsobel
tuser	/mnt/backups/tuser

1 - 2 of 2

Figure 4 SMB shares for both users

There are now two SMB shares set up, those being backup1 and backup2, both designated to the path to a user's home / backup directory on FreeNAS; what these allow for is a single Windows user to have a network drive that is specific to their corresponding user on FreeNAS. As an example, here is how the mapping works for user dsobel (note that the share location, based on the previous figure, should be "\\nastest\dsobel", but figure 5 comes from an earlier test):


What network folder would you like to map?

Specify the drive letter for the connection and the folder that you want to connect to:

Drive: Z:

Folder: \\nastest\backups\dsobel

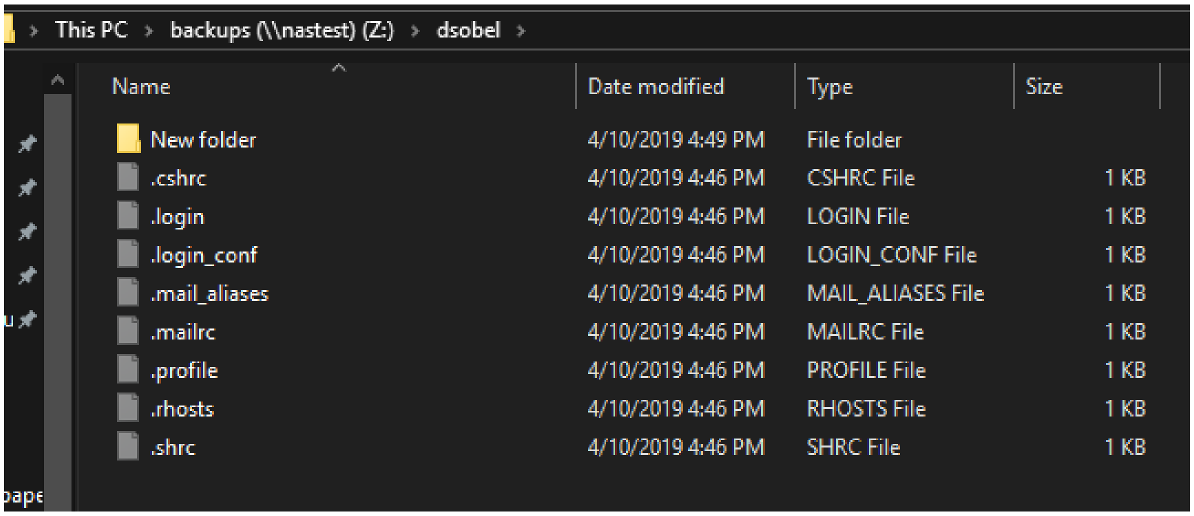
Example: \\server\share

[✓] Reconnect at sign-in

[ ] Connect using different credentials

Connect to a Web site that you can use to store your documents and pictures.

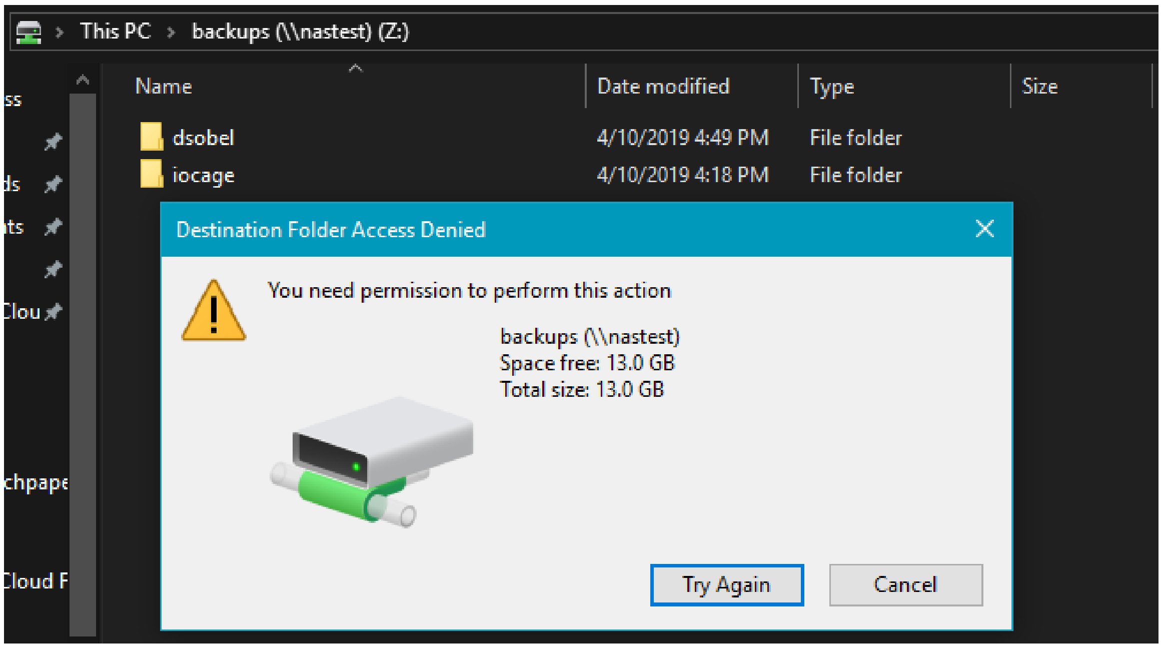
Figure 5 Adding dsobel's SMB share to Windows


This PC > backups (\\nastest) (Z:) > dsobel >

Name	Date modified	Type	Size
New folder	4/10/2019 4:49 PM	File folder	
.cshrc	4/10/2019 4:46 PM	CSHRC File	1 KB
.login	4/10/2019 4:46 PM	LOGIN File	1 KB
.login_conf	4/10/2019 4:46 PM	LOGIN_CONF File	1 KB
.mail_aliases	4/10/2019 4:46 PM	MAIL_ALIASES File	1 KB
.mailrc	4/10/2019 4:46 PM	MAILRC File	1 KB
.profile	4/10/2019 4:46 PM	PROFILE File	1 KB
.rhosts	4/10/2019 4:46 PM	RHOSTS File	1 KB
.shrc	4/10/2019 4:46 PM	SHRC File	1 KB

Figure 6 Proper write permissions were set

We can see that dsobel was allowed to make a new folder in the designated backup directory, as expected. However, to verify proper access control, here is the message that pops up when dsobel attempts to delete another folder outside of the dsobel directory:



Destination Folder Access Denied

You need permission to perform this action

backups (\\nastest)
Space free: 13.0 GB
Total size: 13.0 GB

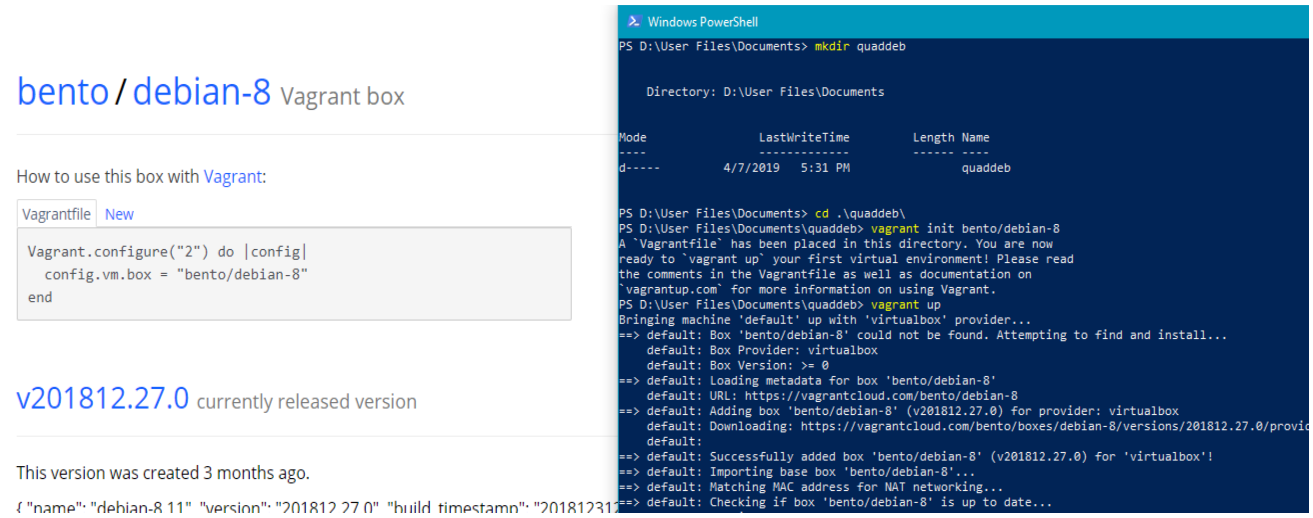
Figure 7 Further demonstration of write permissions

Having set up a storage pool, added users with home / backup directories in the root of the pool with the denoted permissions, and verified the ability to map the created SMB shares, we can move on to setting up the virtual tape library to which storage in FreeNAS will be backed up.

QuadstorVTL

With FreeNAS set up, we can move on to QuadstorVTL (virtual tape library), a relatively simple utility for emulating a tape environment that runs on top of Debian 7 or 8 and various others (we will use Debian 8 for demonstrating the setup due to 7 being now unusable) as a Linux system service and provides an online interface for adding and removing virtual tape drives and cartridges.

The simplest way to quickly bring up a Debian 8 VM is through Vagrant, which provides a command line interface for deploying VMs with access to a massive repository of premade boxes for projects like this one. Note that the easiest (and free) virtualization program to use with Vagrant is Oracle Virtualbox. Figures 8 and 9 are screenshots of the initial setup with Vagrant and a few of its installation steps:



bento / debian-8 Vagrant box

How to use this box with Vagrant:

Vagrantfile New

```
Vagrant.configure("2") do |config|
  config.vm.box = "bento/debian-8"
end
```

v201812.27.0 currently released version

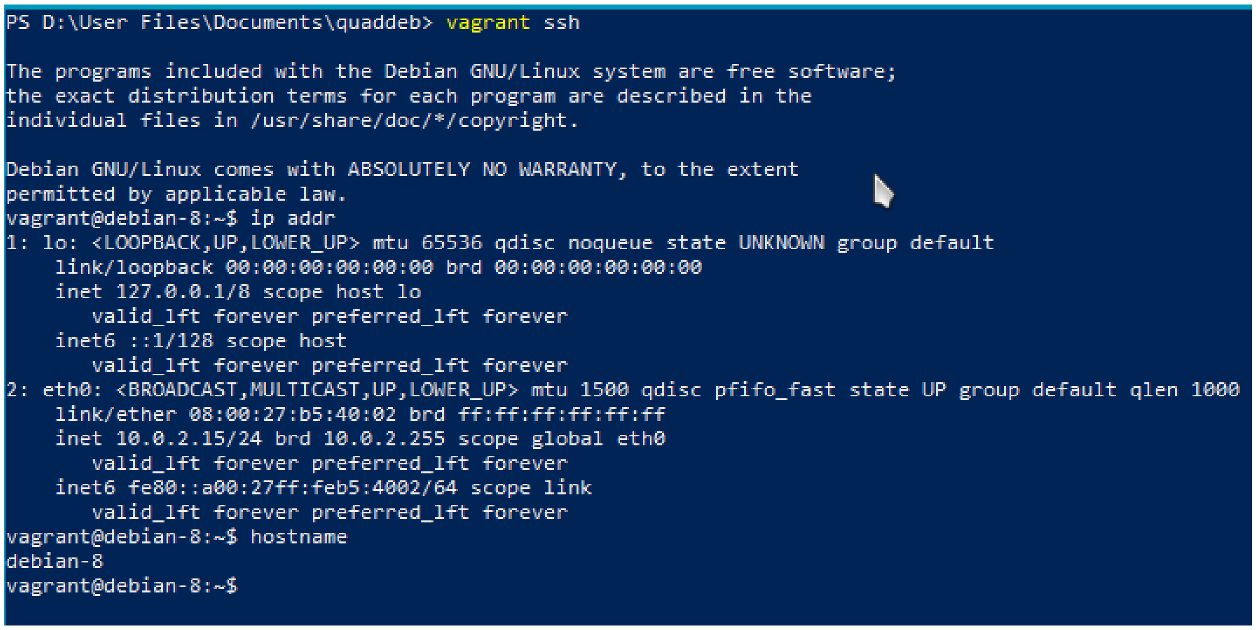
This version was created 3 months ago.

{ "name": "debian-8.11", "version": "201812.27.0", "build_timestamp": "20181231

Windows PowerShell

```
PS D:\User Files\Documents> mkdir quaddeb


    Directory: D:\User Files\Documents


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----         4/7/2019   5:31 PM                quaddeb


PS D:\User Files\Documents> cd .\quaddeb\
PS D:\User Files\Documents\quaddeb> vagrant init bento/debian-8
A `Vagrantfile` has been placed in this directory. You are now
ready to `vagrant up` your first virtual environment! Please read
the comments in the Vagrantfile as well as documentation on
`vagrantup.com` for more information on using Vagrant.
PS D:\User Files\Documents\quaddeb> vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Box 'bento/debian-8' could not be found. Attempting to find and install...
    default: Box Provider: virtualbox
    default: Box Version: >= 0
==> default: Loading metadata for box 'bento/debian-8'
    default: URL: https://vagrantcloud.com/bento/debian-8
==> default: Adding box 'bento/debian-8' (v201812.27.0) for provider: virtualbox
    default: Downloading: https://vagrantcloud.com/bento/boxes/debian-8/versions/201812.27.0/provid
    default:
==> default: Successfully added box 'bento/debian-8' (v201812.27.0) for 'virtualbox'!
==> default: Importing base box 'bento/debian-8'...
==> default: Matching MAC address for NAT networking...
==> default: Checking if box 'bento/debian-8' is up to date...
```

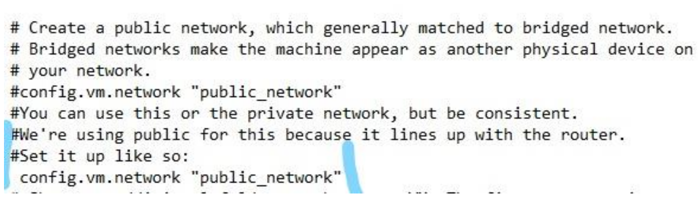
Figure 8 Vagrant initiation of Debian 8 Box from user bento



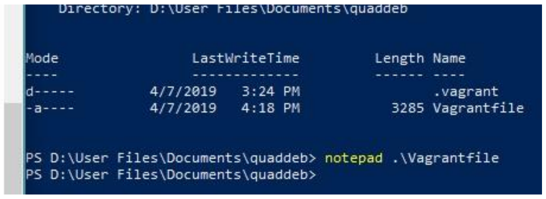
```
PS D:\User Files\Documents\quaddeb> vagrant ssh

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
vagrant@debian-8:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:b5:40:02 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:feb5:4002/64 scope link
       valid_lft forever preferred_lft forever
vagrant@debian-8:~$ hostname
debian-8
vagrant@debian-8:~$
```

Figure 9 SSH into Debian 8 machine with hostname verification

After running the command “vagrant up” in the folder we set up for the Debian vagrant box it will run through relevant software and driver installations. At the end you can run “vagrant ssh” to get into the machine. As shown in the screenshot, the Debian box is up and running. The next steps will be setting up the VM with an accessible IP address (by default it runs as NAT, which will make it harder for connecting the two VMs). In the next screenshot you can see the line that needs to be changed for this to work. Run “vagrant halt” and wait for it to shut down before changing anything.



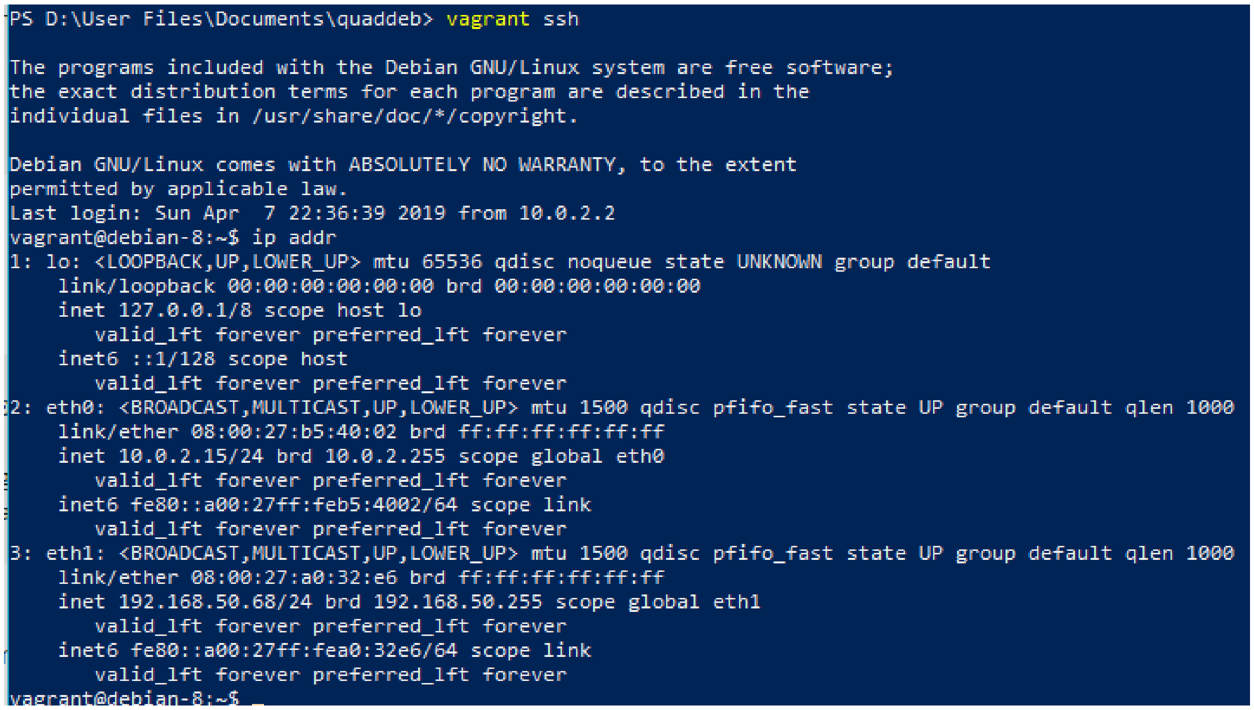
```
# Create a public network, which generally matched to bridged network.
# Bridged networks make the machine appear as another physical device on
# your network.
#config.vm.network "public_network"
#You can use this or the private network, but be consistent.
#We're using public for this because it lines up with the router.
#Set it up like so:
 config.vm.network "public_network"
```



```
    Directory: D:\User Files\Documents\quaddeb

Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----         4/7/2019   3:24 PM                .vagrant
-a----         4/7/2019   4:18 PM           3285 Vagrantfile

PS D:\User Files\Documents\quaddeb> notepad .\Vagrantfile
PS D:\User Files\Documents\quaddeb>
```

Figure 10 Vagrantfile bridged configuration

If you uncomment “config.vm.network ‘public_network,’” in the Vagrantfile (in same directory where you ran vagrant init, as seen in the screenshot), then you can rely on your router to provide an open IP address.



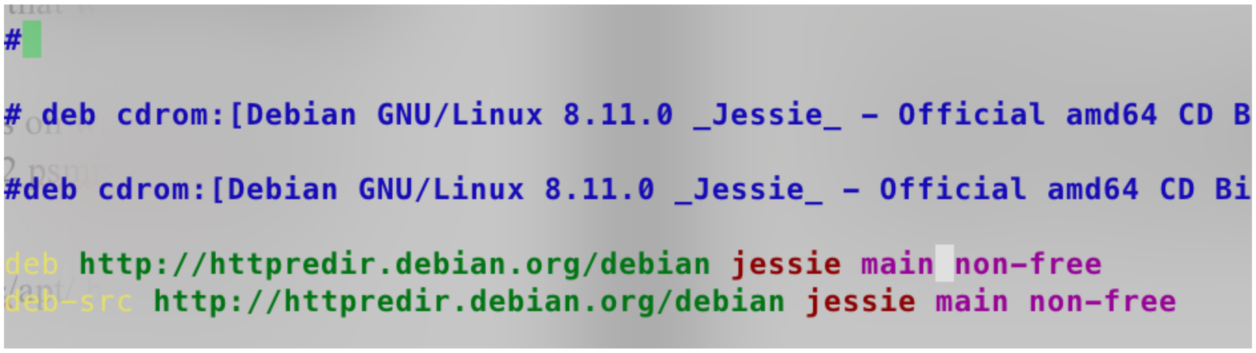
```
PS D:\User Files\Documents\quaddeb> vagrant ssh

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Apr  7 22:36:39 2019 from 10.0.2.2
vagrant@debian-8:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:b5:40:02 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:feb5:4002/64 scope link
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:a0:32:e6 brd ff:ff:ff:ff:ff:ff
    inet 192.168.50.68/24 brd 192.168.50.255 scope global eth1
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fea0:32e6/64 scope link
       valid_lft forever preferred_lft forever
vagrant@debian-8:~$ _
```

Figure 11 Verifying IP address assignment

After editing the Vagrantfile, you may see connection abortion and reset messages; just wait for Vagrant to run through its paces and it will eventually be ready for use. After running “vagrant ssh” again and then “ip addr” you should see an IP matching your router’s subnetting (.50 in this case) like in the black boxed area. With this working, we will run through setting up QuadstorVTL, which has guidance on its own website at quadstor.com that we will be following here.

The first step is to install the main packages QuadstorVTL relies on with the following command: “apt-get install uuid-runtime build-essential sg3-utils apache2 psmisc linux-headers-`uname -r`” (Quadstor, 2019)

After these are installed, you will want to edit sources.list located in /etc/apt/ by adding “non-free” to the first two sources (Quadstor, 2019) as seen in figure 12.

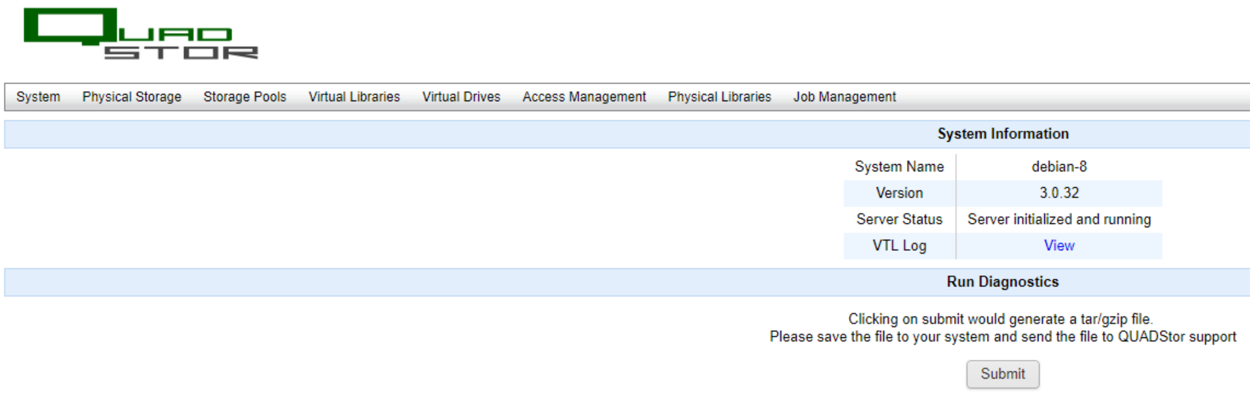


```
#

# deb cdrom:[Debian GNU/Linux 8.11.0 _Jessie_ - Official amd64 CD B

#deb cdrom:[Debian GNU/Linux 8.11.0 _Jessie_ - Official amd64 CD Bi

deb http://httpredir.debian.org/debian jessie main non-free
deb-src http://httpredir.debian.org/debian jessie main non-free
```

Figure 12 Final version of sources.list file

Now that the sources are updated, you should run "sudo apt-get update" then "sudo apt-get install firmware-qlogic" to install firmware-qlogic. The next command is ensuring Apache is set up for cgi, which can be done by running "sudo a2enmod cgi." (Quadstor, 2019).

The last step on the command line is installing QuadstorVTL, which can be downloaded using the command "wget https://www.quadstor.com/vtldownloads/quadstor-vtl-ext-3.0.32-debian-x86_64.deb" followed by "sudo dpkg -i quadstor-vtl-ext-3.0.32-debian-x86_64.deb." After this, you need to run "sudo systemctl enable quadstorvtl" (it should start on boot) then reboot. After this, run ip addr to confirm your IP address and go to it in your web browser. If it worked out, you will be greeted with the QuadstorVTL web interface along with multiple options.



System Information

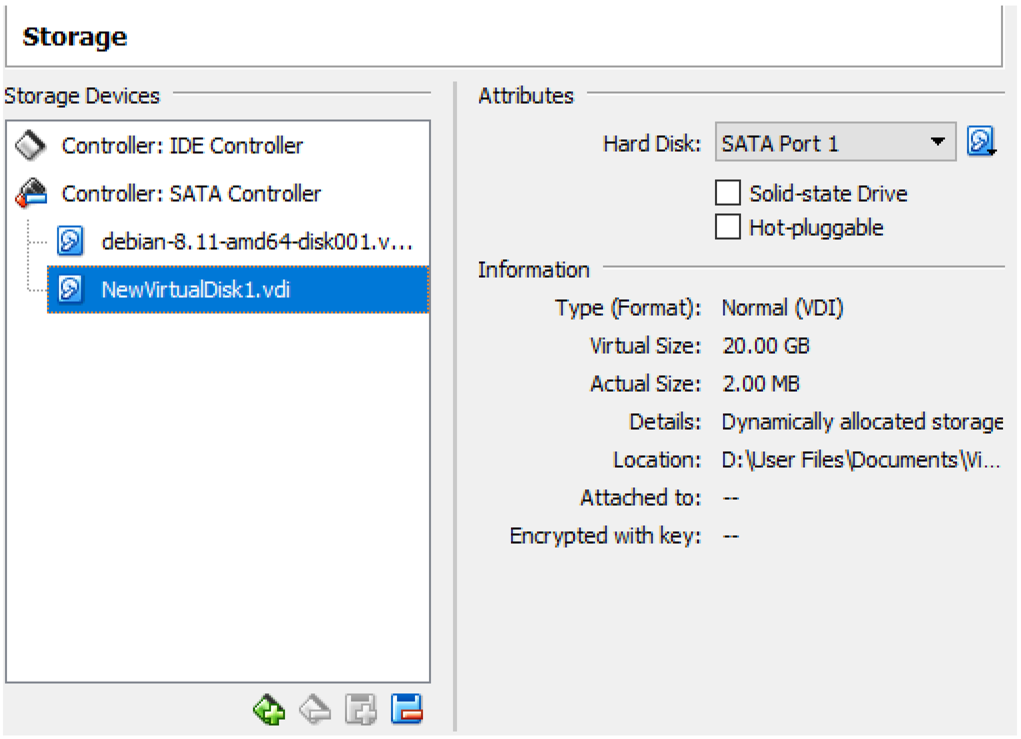
System Name	debian-8
Version	3.0.32
Server Status	Server initialized and running
VTL Log	View

Run Diagnostics

Clicking on submit would generate a tar/gzip file.
Please save the file to your system and send the file to QUADStor support

Figure 13 Verifying access to QuadstorVTL web interface

In testing we set up both a virtual tape library with virtual tape drives (called vDrives and vCartridges here) inside of it and a standalone virtual tape drive, but for the demonstration we will focus on standalone virtual drives. Before continuing on, however, shut down your VM and add a virtual hard drive of any size to your VM (this can be done through VirtualBox's interface and you can choose either SATA or SCSI) which can be turned into a storage pool in the "Storage Pool" tab.

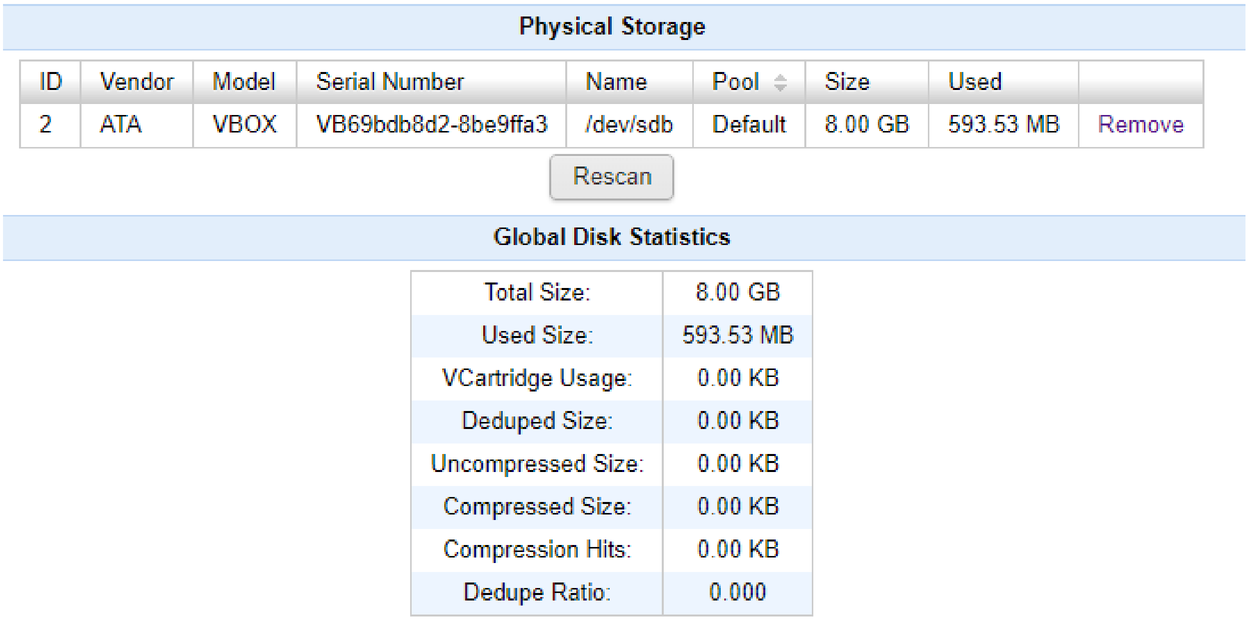

Storage
Storage Devices
Controller: IDE Controller
Controller: SATA Controller
debian-8.11-amd64-disk001.v...
NewVirtualDisk1.vdi
Attributes
Information

Hard Disk: SATA Port 1
Solid-state Drive
Hot-pluggable

Type (Format):	Normal (VDI)
Virtual Size:	20.00 GB
Actual Size:	2.00 MB
Details:	Dynamically allocated storage
Location:	D:\User Files\Documents\Vi...
Attached to:	--
Encrypted with key:	--

Figure 14 Adding a new virtual drive

To use the new hard disk, go to the “Physical Storage” tab and create a storage pool with the new drive by clicking “Add.” After about 30 seconds or so, click rescan at this screen again to make sure it was successfully added to the pool. This should be the final result:

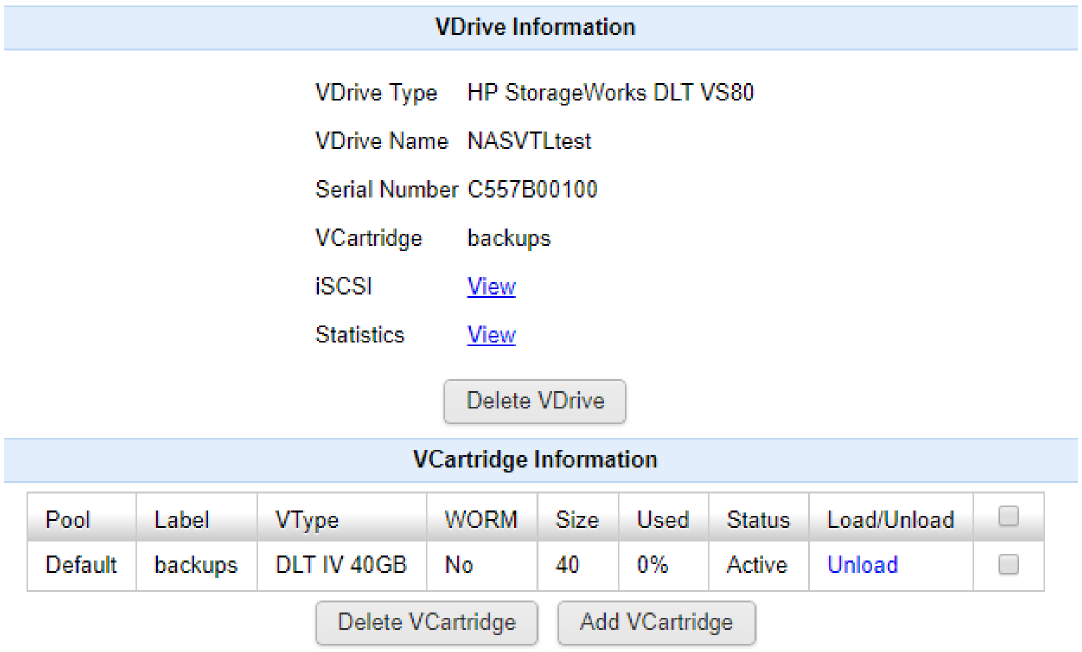


ID	Vendor	Model	Serial Number	Name	Pool	Size	Used	
2	ATA	VBOX	VB69bdb8d2-8be9ffa3	/dev/sdb	Default	8.00 GB	593.53 MB	Remove

Total Size:	8.00 GB
Used Size:	593.53 MB
VCartridge Usage:	0.00 KB
Deduped Size:	0.00 KB
Uncompressed Size:	0.00 KB
Compressed Size:	0.00 KB
Compression Hits:	0.00 KB
Dedupe Ratio:	0.000

Figure 15 Available “physical” storage to pool

Now you can move onto the “Virtual Drives” tab. In this tab you can add a new virtual tape drive by clicking on “Add VDrive.” Although the size of the storage pool we made was significantly larger than the VM drive we added, for testing purposes it makes no difference if the sizes match. On the next screen you can keep the default settings (smaller drives) enabled and name it something indicative of its use; you will name the VCartridge as well. At the end of this process the tab should show up like this:


VDrive Information

VDrive Type	HP StorageWorks DLT VS80
VDrive Name	NASVTLtest
Serial Number	C557B00100
VCartridge	backups
iSCSI	View
Statistics	View

VCartridge Information

Pool	Label	VType	WORM	Size	Used	Status	Load/Unload	☐
Default	backups	DLT IV 40GB	No	40	0%	Active	Unload	☐

Figure 16 Fully set up VDrive

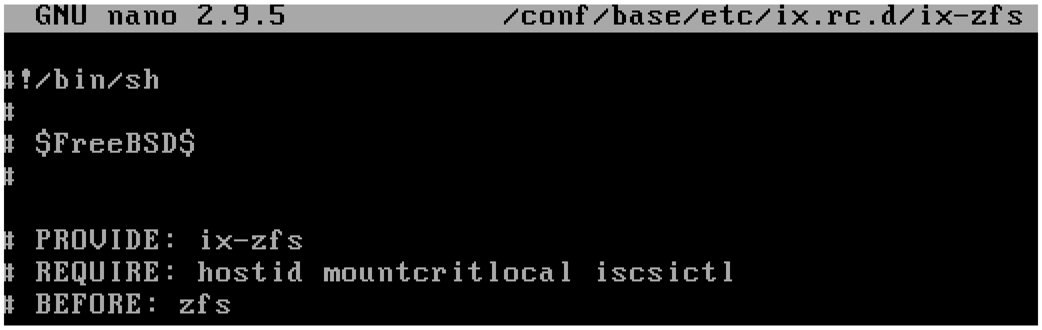
Note that WORM (write once read many), which means data can only be read after it is written (FUJITSU), is disabled for the VCartridge in this case because the goal of this NAS VTL setup is to treat the tape as a constantly overwritten backup device. With a virtual tape drive set up, the baseline setup for QuadstorVTL is complete. If you click on “View” next to iSCSI you can see the IQN address of the tape drive; in our case the address is iqn.2006-06.com.quadstor.vdrive.NASVTL, which we will use in the next section.

Setting up FreeNAS as an iSCSI Initiator

Normally, FreeNAS is set up to allow the creation of iSCSI targets, but there is no part of the web GUI which supports using it as an iSCSI Initiator. A four step solution for this which has been employed comes from the iXsystems community forum in a post by user thewizard231 on 6/18/2018. The rundown of the post is as follows, with some additional information that came out of testing the system:

1. Go to the terminal interface in the FreeNAS VM window (or go to the “Shell” tab in the web GUI) and choose option 9, shell.
2. Set the FreeNAS system to require iscsictl by editing line 7 of the ix-zfs file in nano/vi (see figure 17). According to the post, ix-zfs determines the general mount point for

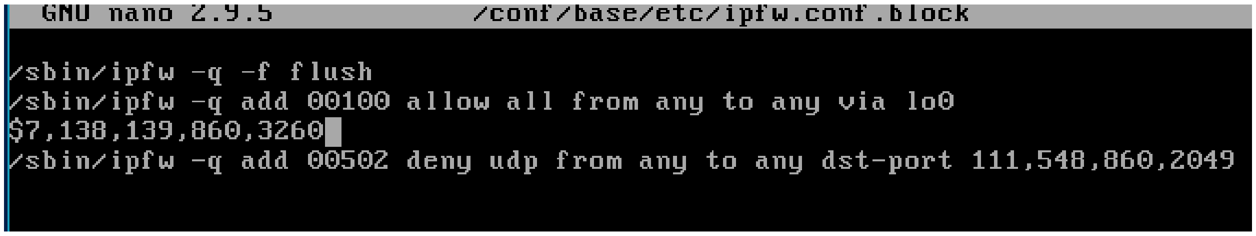
volumes on boot; by requiring iscsictl to run before zfs is up and running, iSCSI devices can be seen by FreeNAS (thewizard231, 2018).



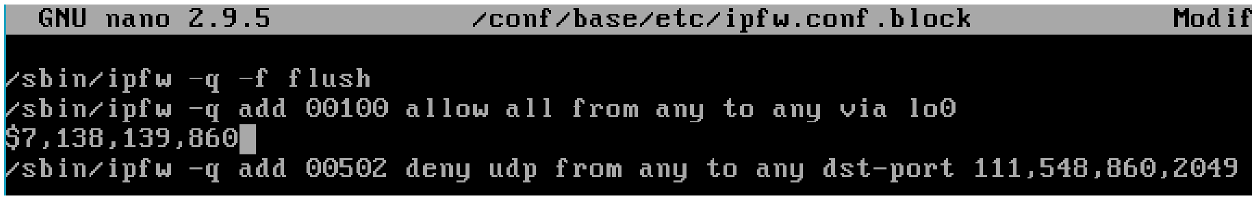
```
  GNU nano 2.9.5            /conf/base/etc/ix.rc.d/ix-zfs

#!/bin/sh
#
# $FreeBSD$
#


# PROVIDE: ix-zfs
# REQUIRE: hostid mountcritlocal iscsictl
# BEFORE: zfs
```

Figure 17 Added iscsictl to ix-zfs

3. Remove the block on port 3260 (iSCSI) by deleting it from line 4 of ipfw.conf.block (See figure 18+19). If this is not done, then FreeNAS will hang on boot as it will try to start iscsid infinitely (thewizard231, 2018).



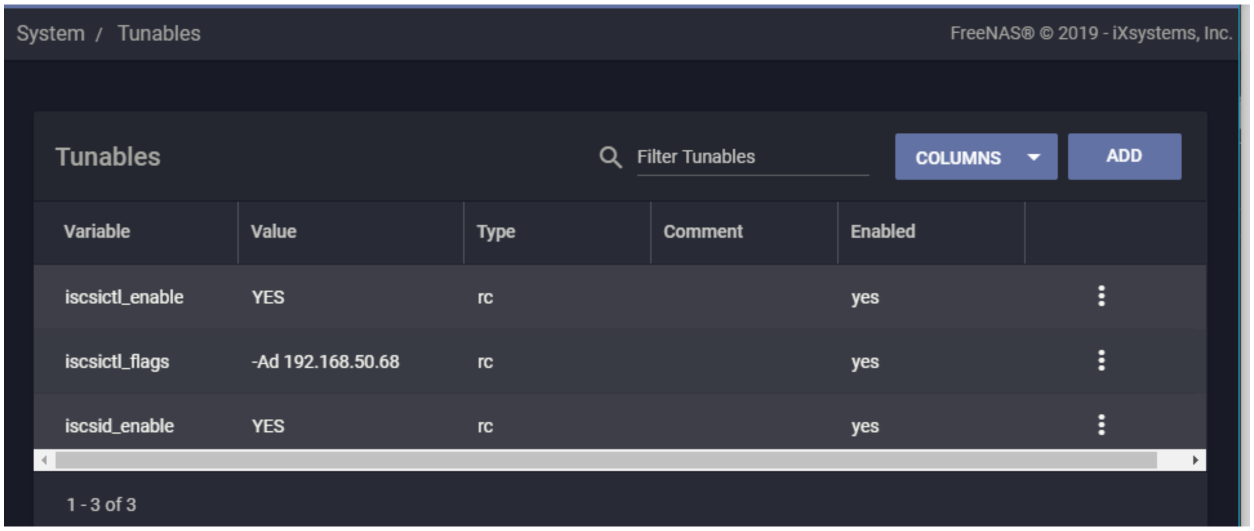
```
  GNU nano 2.9.5            /conf/base/etc/ipfw.conf.block

/sbin/ipfw -q -f flush
/sbin/ipfw -q add 00100 allow all from any to any via lo0
$7,138,139,860,3260
/sbin/ipfw -q add 00502 deny udp from any to any dst-port 111,548,860,2049
```



```
  GNU nano 2.9.5            /conf/base/etc/ipfw.conf.block           Modif

/sbin/ipfw -q -f flush
/sbin/ipfw -q add 00100 allow all from any to any via lo0
$7,138,139,860
/sbin/ipfw -q add 00502 deny udp from any to any dst-port 111,548,860,2049
```

Figures 18+19 Before and after removing deny on port 3260

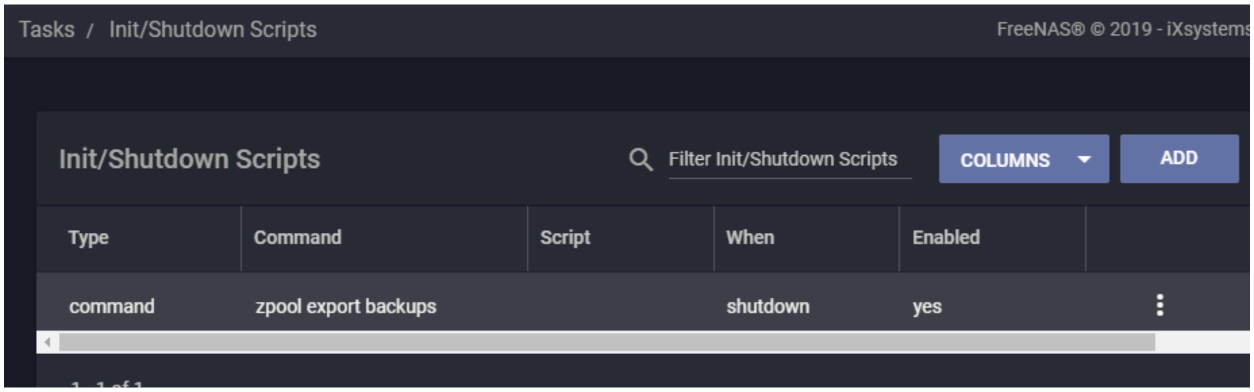
4. Add three rc.conf configurations in “Tunables” under the “System” tab in the web GUI: iscsid_enable, iscsictl_enable, and iscsictl_flags with the values in the screenshot, ensuring all commands are type rc.conf (thewizard231, 2018). These are what will allow the system to start iscsid and iscsictl on boot.



Variable	Value	Type	Comment	Enabled
iscsictl_enable	YES	rc		yes
iscsictl_flags	-Ad 192.168.50.68	rc		yes
iscsid_enable	YES	rc		yes

Figure 20 Newly created rc.conf tunables for full iSCSI functionality on boot

5. Add the command zpool export <poolname> to the shutdown scripts in "Init/Shutdown Scripts" under the "Tasks" tab to ensure the system does not hang on shutdown after all disks are synced (thewizard231, 2018). The FreeNAS version referenced in the post is 11.1 U5, so this may have been fixed in the version used in this example but should be included to ensure everything works. The section should look like this now:

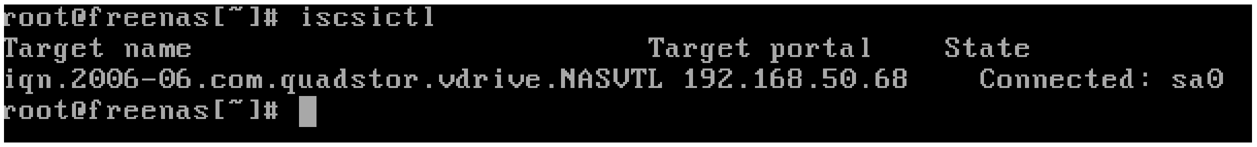

Tasks / Init/Shutdown Scripts
FreeNAS® © 2019 - iXsystems
Init/Shutdown Scripts
Filter Init/Shutdown Scripts
COLUMNS
ADD

Type	Command	Script	When	Enabled
command	zpool export backups		shutdown	yes

Figure 21 Added "zpool export backups" to shutdown to avoid hanging

Adding and testing the QuadstorVTL iSCSI Target

1. To add the iSCSI target(s) from QuadstorVTL to FreeNAS, run "iscsictl -Ad <VTL IP>," in our case 192.168.50.68
2. Reboot by entering the command "reboot" while still in the shell.
3. Run iscsictl through the shell again on reboot and it should show all connected iSCSI targets:

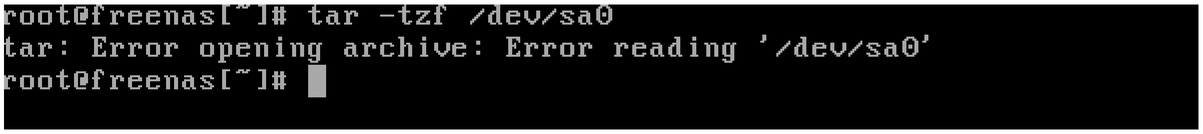


```
root@freenas[~]# iscsictl
Target name                                Target portal    State
iqn.2006-06.com.quadstor.vdrive.NASVTL 192.168.50.68    Connected: sa0
root@freenas[~]#
```

Figure 22 iSCSI device from QuadstorVTL connected

We can see that the NASVTL drive we made earlier is now mounted as /dev/sa0

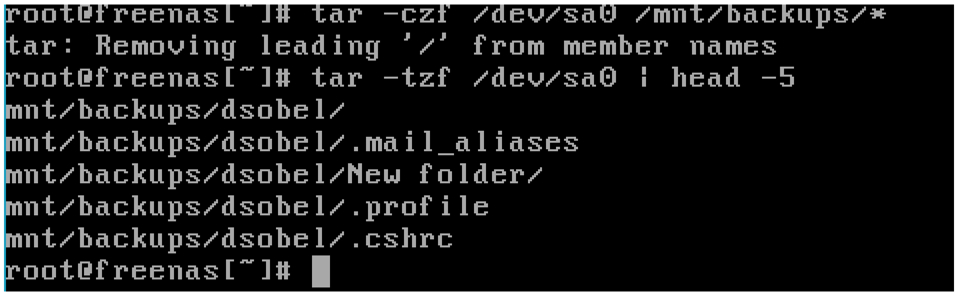
4. To list the data being stored on the tape drive we can run “tar -tzf /dev/sa0” (nixCraft, 2009). It will be empty at first because nothing has been written yet.



```
root@freenas[~]# tar -tzf /dev/sa0
tar: Error opening archive: Error reading '/dev/sa0'
root@freenas[~]#
```

Figure 23 Demonstration of reading tape, error due to being empty

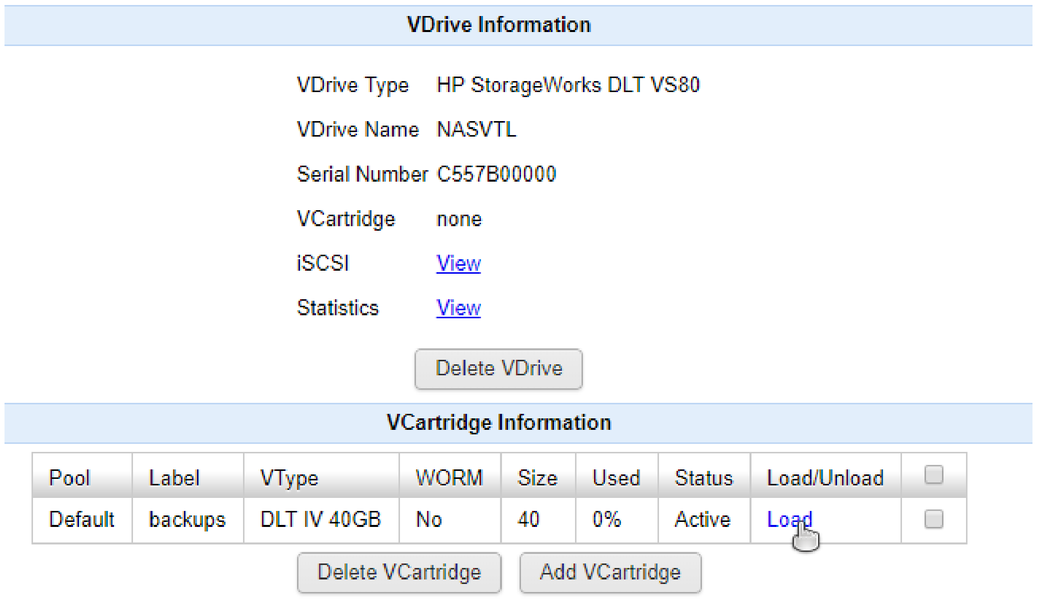
5. To back up data to the tape drive, we run “tar -czf /dev/sa0 <directory(ies)-to-backup>” (nixCraft, 2009). In this example we want to back up the contents of all the home directories within the backups storage pool. The first five lines of the backup contents are shown in the following screenshot:



```
root@freenas[~]# tar -czf /dev/sa0 /mnt/backups/*
tar: Removing leading '/' from member names
root@freenas[~]# tar -tzf /dev/sa0 | head -5
mnt/backups/dsobel/
mnt/backups/dsobel/.mail_aliases
mnt/backups/dsobel/New folder/
mnt/backups/dsobel/.profile
mnt/backups/dsobel/.cshrc
root@freenas[~]#
```

Figure 24 Demonstration of copying files to tape

6. Note that whenever QuadstorVTL is rebooted, you will need to go into the VDrive then click “Load” to connect the VCartridges for FreeNAS to see and access them. You will need to do this on every boot of the Debian 8 machine (ideally boot this one first).



VDrive Type	HP StorageWorks DLT VS80
VDrive Name	NASVTL
Serial Number	C557B00000
VCartridge	none
iSCSI	View
Statistics	View

Pool	Label	VType	WORM	Size	Used	Status	Load/Unload	☐
Default	backups	DLT IV 40GB	No	40	0%	Active	Load	☐

Figure 25 Demonstrating the need for actively loading VCartridges

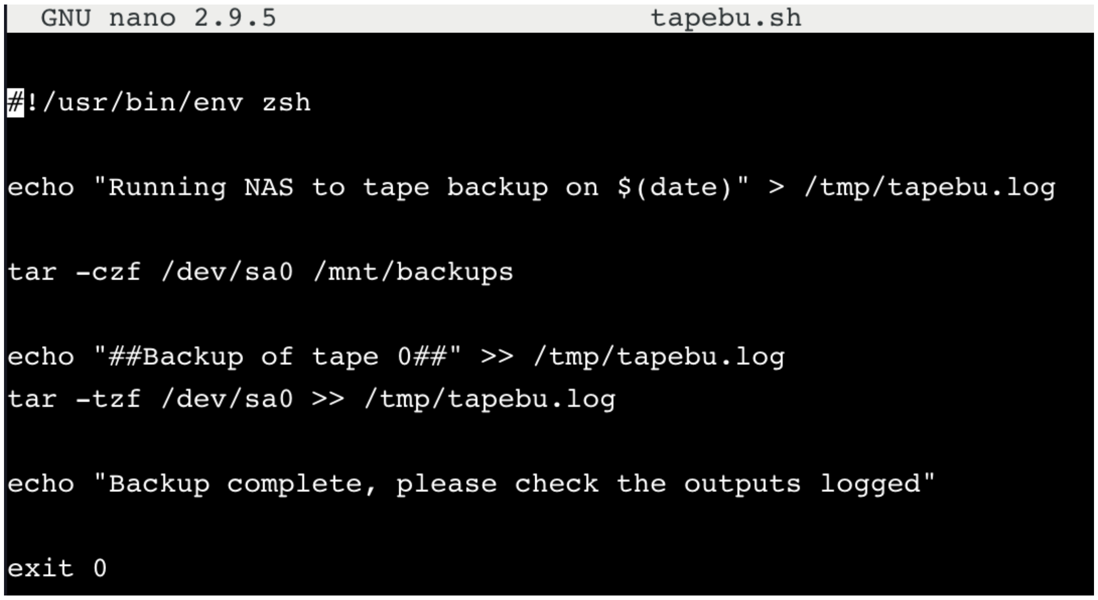
7. On a functional level, the NAS to tape backup system is finished.

Post-Setup Automation and Usage

While the system is ready for use, a few more steps should be taken to ensure that the process of backing up user data, at minimum, is automated. As such, a script for backing up FreeNAS data to the VTL tape storage has been created and added as a cronjob that runs every Sunday at midnight, with recovery remaining a manual job.

Backup/Restore Scripts

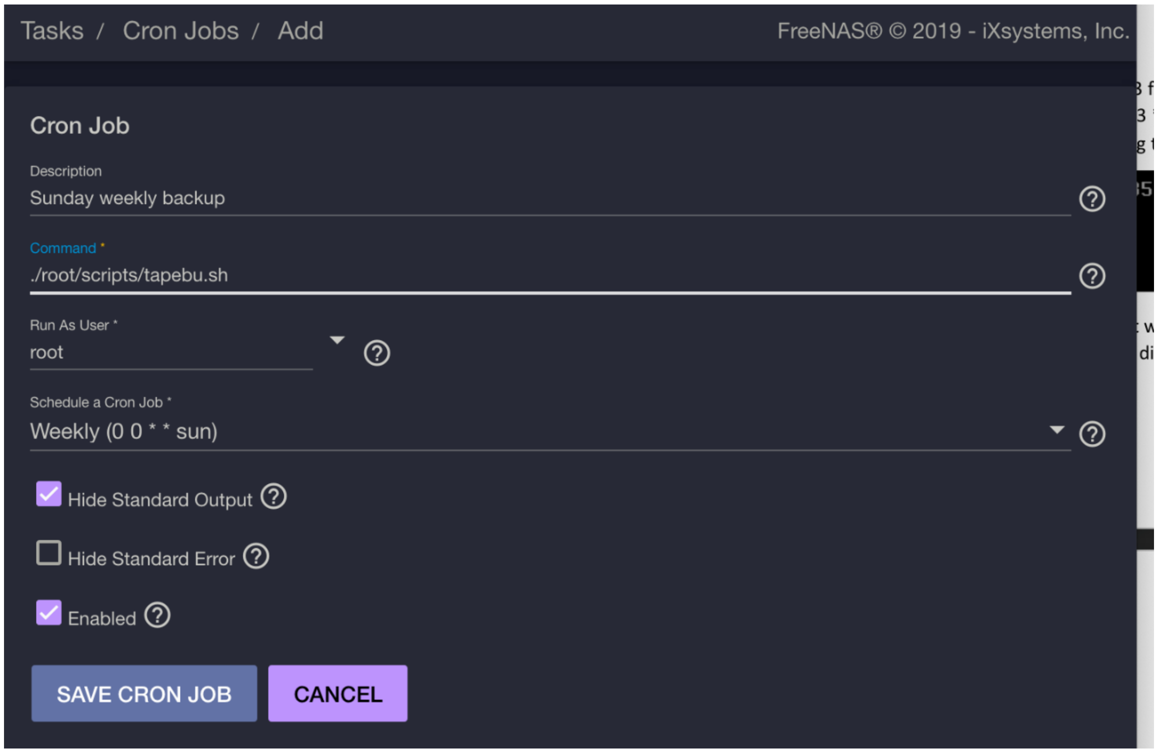
The script used for backing up data from FreeNAS to QuadstorVTL in this case was called tapebu.sh and put in a newly made directory called “scripts” located at /scripts. The main function of the script is denoting the tape backup date in a log file “/tmp/tapebu.log,” backing up data to designated drive(s), and logging which data was backed up.



```
  GNU nano 2.9.5                    tapebu.sh

#!/usr/bin/env zsh

echo "Running NAS to tape backup on $(date)" > /tmp/tapebu.log

tar -czf /dev/sa0 /mnt/backups

echo "##Backup of tape 0##" >> /tmp/tapebu.log
tar -tzf /dev/sa0 >> /tmp/tapebu.log

echo "Backup complete, please check the outputs logged"

exit 0
```

Figure 26 Backup script to run at midnight every Sunday

To automate this script to run every Sunday at midnight, you can go to "Cron Jobs" under the "Tasks" tab in the FreeNAS web GUI and add the job there as shown in figure 27:



Tasks / Cron Jobs / Add FreeNAS® © 2019 - iXsystems, Inc.

Cron Job

Description: Sunday weekly backup

Command *: ./root/scripts/tapebu.sh

Run As User *: root

Schedule a Cron Job *: Weekly (0 0 * * sun)

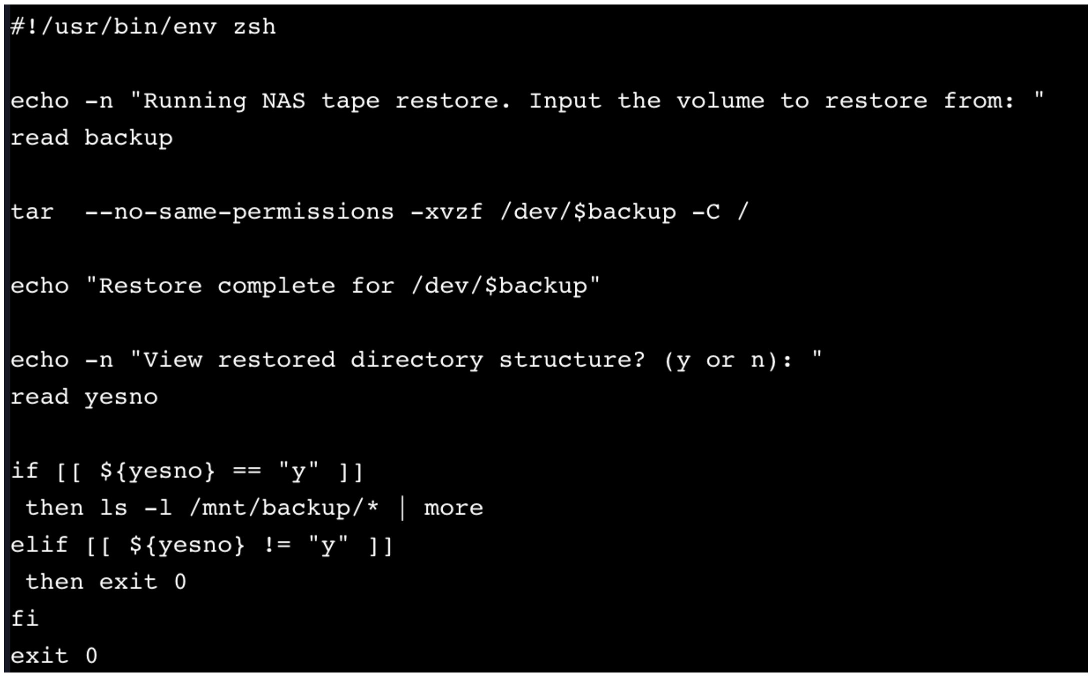
[✓] Hide Standard Output

[ ] Hide Standard Error

[✓] Enabled

Figure 27 Adding backup Cron Job

The other script is taperes.sh, a simple script which prompts the user to select which drive to restore from and could be used to easily restore the directories backed up with tapebu.sh. The script is shown below in figure 28:



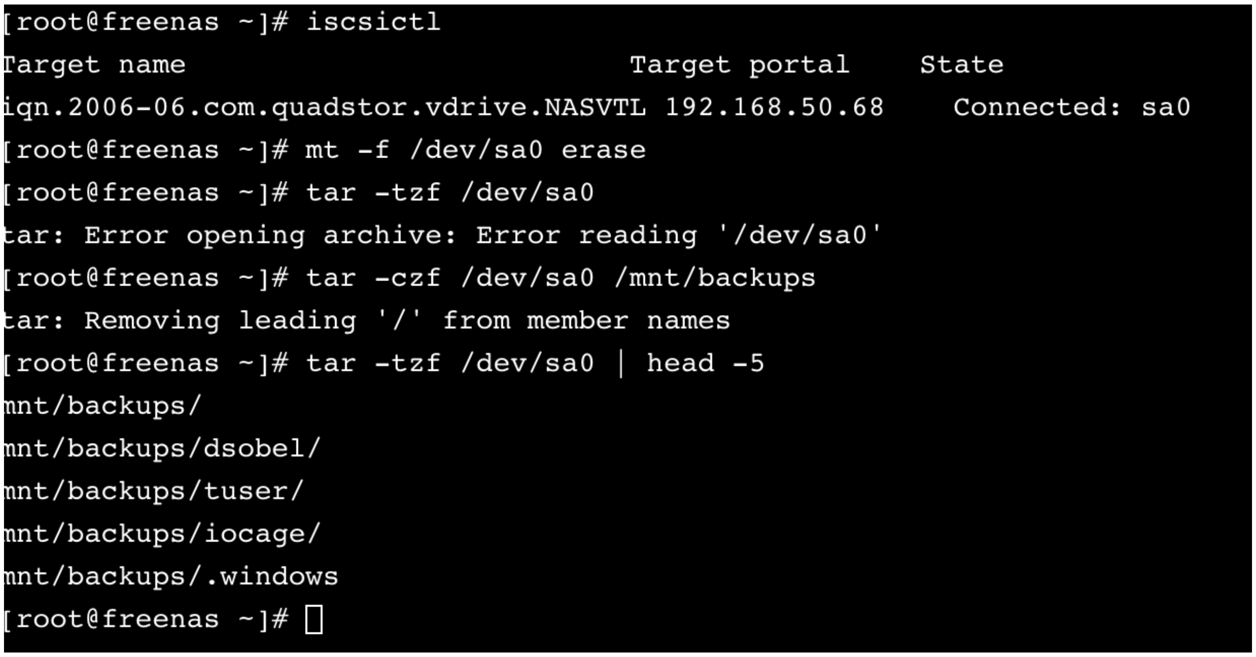
```
#!/usr/bin/env zsh

echo -n "Running NAS tape restore. Input the volume to restore from: "
read backup

tar  --no-same-permissions -xvzf /dev/$backup -C /

echo "Restore complete for /dev/$backup"

echo -n "View restored directory structure? (y or n): "
read yesno

if [[ ${yesno} == "y" ]]
 then ls -l /mnt/backup/* | more
elif [[ ${yesno} != "y" ]]
 then exit 0
fi
exit 0
```

Figure 28 Restore script

We can see that this script runs tar -xvzf, the x being what denotes “extracting” the contents (nixCraft, 2009) of the tape backup to a designated directory, in this case the / directory as the structures of the backed up data are /mnt/backups/*. In addition, the flag –no-same-permissions is used because the restore process would otherwise produce errors regarding setting directories to 755 (this flag was gathered from the manual pages for tar); in testing, folder and file ownership remained the same before and after restoring; note that the backup script, at a small scale like this, can be edited to account for additional drives as needed. With these two scripts available, the backup and restore situation is functional and is ready for use.

Additional Items

A couple additional commands should be considered for handling connected tape drives, specifically for erasing and rewinding. The command format for erasing a tape drive is “mt -f /dev/<tapedrive> erase” (Red Hat, 2013), which is tested in figure 29.



```
[root@freenas ~]# iscsictl
Target name                             Target portal     State
iqn.2006-06.com.quadstor.vdrive.NASVTL 192.168.50.68     Connected: sa0
[root@freenas ~]# mt -f /dev/sa0 erase
[root@freenas ~]# tar -tzf /dev/sa0
tar: Error opening archive: Error reading '/dev/sa0'
[root@freenas ~]# tar -czf /dev/sa0 /mnt/backups
tar: Removing leading '/' from member names
[root@freenas ~]# tar -tzf /dev/sa0 | head -5
mnt/backups/
mnt/backups/dsobel/
mnt/backups/tuser/
mnt/backups/iocage/
mnt/backups/.windows
[root@freenas ~]# 
```

Figure 29 Erasing a tape drive and verifying

For this example, the amount of data stored on the tape drive is small enough to not require it, but the command for rewinding the tape if necessary is "mt -f /dev/<tapedrive> rewind" (Red Hat, 2019). More commands can of course be found in the manual pages for mt.

The last additional item is backing up FreeNAS configurations, which should be restored before attempting a restore from the tape backup(s). To back up the FreeNAS configuration, go to "General" in the "System" tab and click "Save config" to receive a download of the configuration (Protocase Incorporated, 2015). Note that this contains information such as passwords and as such should be stored appropriately.

Conclusions

The benefit of setting up FreeNAS with a tape backup system is two-fold. First, users can back up their data to a storage pool on FreeNAS, allowing for the convenience of backing up and restoring from hard disk drives. Second, with data on FreeNAS being backed up to tape (or tape-like drives) on a weekly basis, there will always be a relatively recent backup available in the case that FreeNAS suffers data loss which can be easily restored due to the sequential nature of the format. The main goal of this project was a small proof of practice for a system like this, which was met; this system may see further use in a home setup.

Sources

FUJITSU. (n.d.). WORM (Write Once Read Many) Function Data tampering prevention and original content retention. Retrieved April 10, 2019, from https://www.fujitsu.com/global/products/computing/storage/tape/eternus-lt/feature/STRSYS_b13.html

Hope, M. (2018, May 17). Evaluating Your Storage Options: Tape Or Cloud. Retrieved April 10, 2019, from https://www.infogoto.com/evaluating-your-storage-options-tape-or-cloud/

IXsystems, Inc. (n.d.). Features - FreeNAS - Open Source Storage Operating System. Retrieved April 10, 2019, from https://freenas.org/about/features/

IXsystems. (2015, May 12). Yes, You Can Virtualize FreeNAS. Retrieved April 10, 2019, from https://www.ixsystems.com/blog/yes-you-can-virtualize-freenas/

NixCraft. (2009, June 03). Linux Tape Backup With mt And tar Command Howto. Retrieved April 10, 2019, from https://www.cyberciti.biz/faq/linux-tape-backup-with-mt-and-tar-command-howto/

Protocase Incorporated. (2015, May 11). How to back up configurations in FreeNAS in the event of boot failure. Retrieved April 10, 2019, from https://www.45drives.com/wiki/index.php?title=How_to_back_up_configurations_in_FreeNAS_in_the_event_of_boot_failure

QUADStor Systems. (n.d.). Installation/Upgrading on RHEL/CentOS, SLES 12 SP2 and Debian 7/8 | QUADStor Systems. Retrieved April 10, 2019, from https://www.quadstor.com/vtlsupport/145-installation-on-rhel-centos-sles-debian.html

Red Hat, Inc. (2013, October 24). Useful Linux Tape Backup commands. Retrieved April 10, 2019, from https://access.redhat.com/solutions/68115

Thewizard231. (2018, June 19). Freenas as an iscsi initiator. Retrieved April 10, 2019, from https://www.ixsystems.com/community/threads/freenas-as-an-iscsi-initiator.22098/#post-462100