Table of Contents

HDD/SSD Validation

HDD validation (in this case) basically involves 5 stages.

1. A SMART short test. This is a test that looks at certain aspects of the electrical and mechanical performance of the HDD. It is not a thorough test of the HDD. The tests take somewhere in the region of 2-5 minutes to complete.
2. A SMART conveyance test. This is a test performed on HDDs to check if they have survived transit without any damage. (I don’t know how they differ from the short or long test if someone wants to give me the information Fester will try to add it.)
3. A SMART long test. Think of this as an extended version of the short test. It is a much more thorough test of the HDD and will include a surface scan of the drive. This test will take many hours to complete depending on the capacity of the HDD.
4. A Badblocks test. This is a test where every physical location on the HDD has a write/read test performed on it. The test consists of a block of data that gets written to every physical location on the HDD in sequence. Every physical location on the HDD is then read back also in sequence and each time at each location the value is checked to see if it is correct. This is one pass. The whole process if repeated with a different block of data, this is the second pass. The badblocks test uses 4 patterns by default. This test will take a very long time, usually between 24 hours to a few days depending on the capacity of the drive.
5. The SMART long test is repeated.

SMART Tests: A Short Introduction

SMART stands for Self-Monitoring Analysis and Reporting Technology.

A SMART test is a test a HDD or SDD can perform by itself on itself. These tests, often referred to as “self tests” are carried out by the HDD’s/SDD’s onboard firmware, not a separate piece of software running on the server as we have already seen.

The results of these tests are stored in the drives onboard non-volatile memory so they can be retrieved and utilised by simply interrogating the drive in the correct way.

However, to be able to use the SMART capabilities built into the drives we need a program or an OS that is capable of communicating with the built in SMART functions of the drive.

With such a program or OS present we can simply issue commands to invoke the firmware to initiate a SMART test and/or interrogate a SMART drive to obtain the results of that test (very convenient).

Only one SMART test can be performed per drive. So you cannot run the short test and the long test on the same drive simultaneously. Also the current SMART test must complete before another can be run on the same drive. If a SMART test is running on a drive and you start another then the current test is stopped and abandoned in favour of the newly requested test.

Fortunately, you can run SMART tests in parallel on different drives. So you could have any number of

drives all performing the short test at the same time, or the long test or a mix if you wish (i.e. some performing the short test and some performing the long test).

HDD/SSD Validation Via the FreeNAS OS Route

There is more than one way to carry out the HDD/SDD validation tests on the server.

A program specially written for this purpose could be used in a bootable form and run on the server.

However, the easiest way to conduct the HDD/SDD validation tests is to install the FreeNAS OS on the server. It has everything we need. This is not a proper installation of the OS, but just a test installation so we can conduct the SMART and badblocks tests needed.

There are a number of ways the FreeNAS OS can be installed, for example from a CD/DVD, or across a network using PXE. Fester favours a USB stick.

Create the installer USB stick, install FreeNAS, and enable the SSH service, as described on the linked pages. When you first log in to the web GUI, you'll likely see the Initial Wizard; you can just exit out of this at this time.

Conducting The HDD/SDD Validation Tests (Finally!) In FreeNAS Via An SSH Console

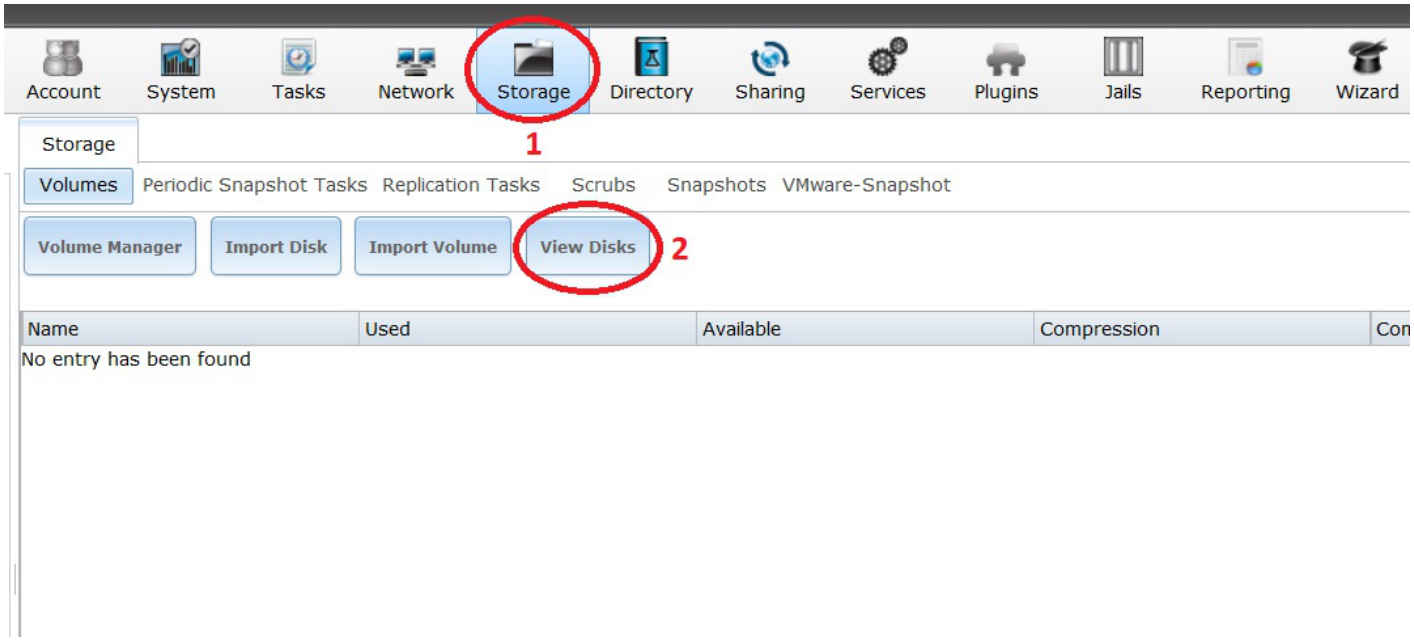
FreeNAS comes with certain software tools and capabilities built into it that will make the task of HDD/SDD validation much easier. This is why we needed to install it before conducting the tests.

The SSH console provides no tools for validation purposes, but does provide the means by which we can flexibly interact with the built in tools in FreeNAS to accomplish the validation tests. This is why we needed to set this up before carrying out the tests.

SMART Short Tests Via The SSH Console

Open up the FreeNAS web GUI in your browser and log in.

Go to the “Storage” page (1) and click the “View Disks” button (2).

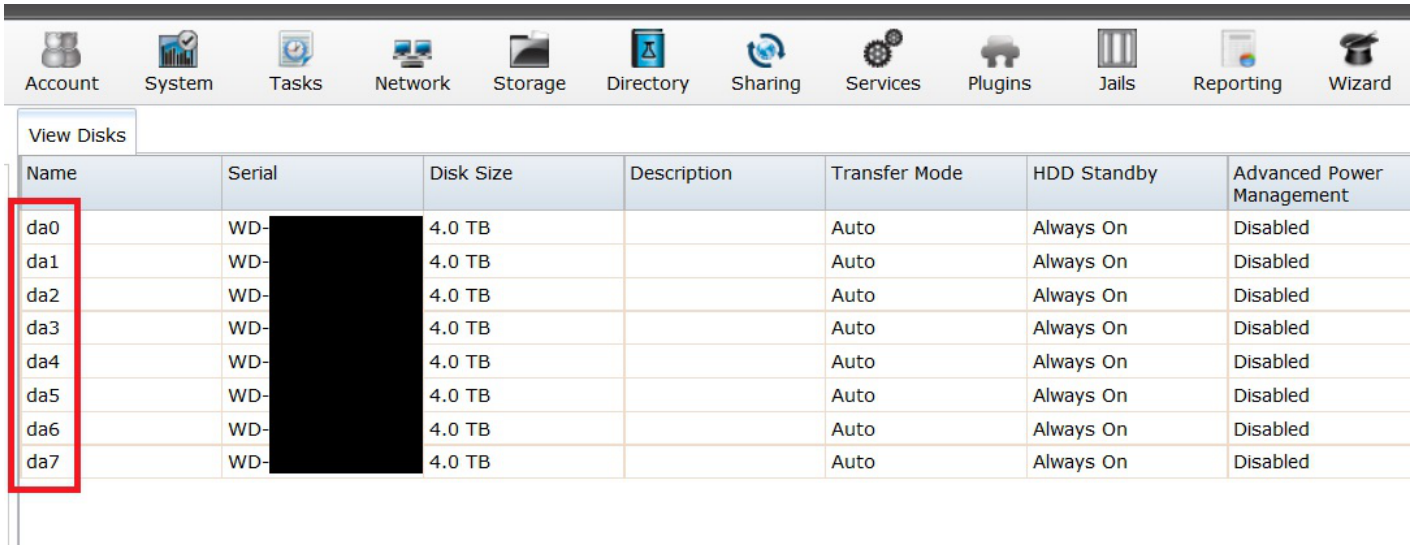


This should bring up a list of the storage HDDs (i.e. for data, not the OS) that are currently in your system.

Make a list of the names of each drive (shown in a red box in the screen shot) these will be needed soon.

(On Fester's system this would be **da0 – da7**, giving a total of 8 HDDs.)

Incidentally, the name FreeNAS gives the OS HDD is **ada0**. If you have two OS drives (i.e. a mirrored configuration) this would be **ada0** and **ada1** respectively.



Start an SSH session and log in.

Where possible when entering commands it is easier and more accurate to use copy and paste. You can copy the text out of this document in the usual way (i.e. highlight the text, right click with the mouse and from the pop up menu select "Copy") and then paste it into the PuTTY SSH console by simply right clicking with the mouse anywhere in the console window (the copied text will appear at the command prompt).

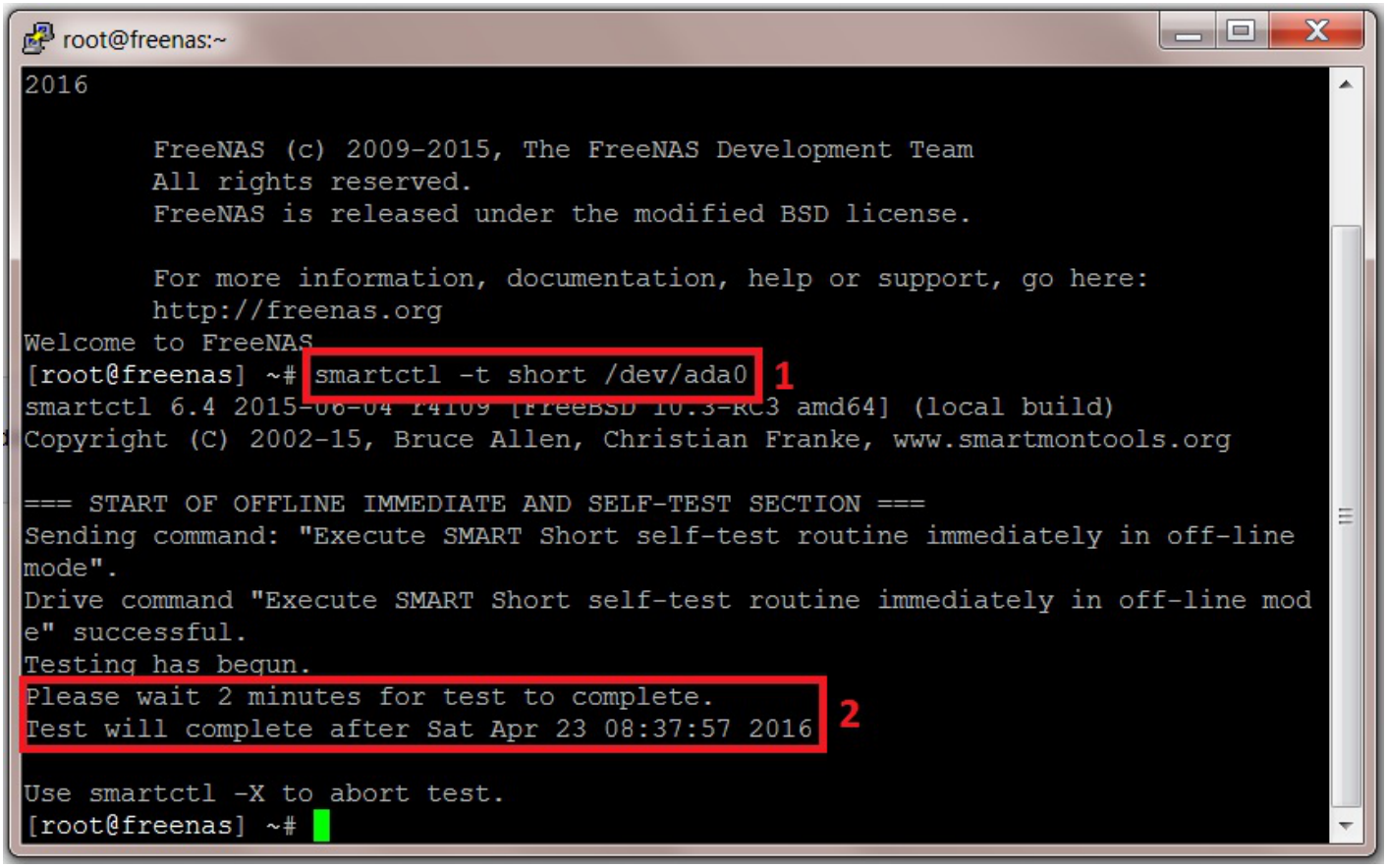
If you do it manually then commands entered at the prompt must be exactly as shown including all the spaces or they tend not to work.

Let us start by running the SMART short test on the OS drive labelled ada0 (in Fester's case this is the SSD drive).

At the command prompt type:

```
smartctl -t short /dev/ada0
```

You should get the following screen, the entered command is shown in the first red box (1) and the duration and completion time are shown in the second (2).



(Do not worry about the fact that you cannot see any results or the test running. This is completely correct. The results are obtained later by entering another command at the command prompt after all the tests are completed.)

We now need to repeat this process for each drive in the system. We do not need to wait for this drive to complete its test before starting another on a different drive.

So at the command prompt enter the command to start the SMART short test for the next drive in your system (in Fester's case this is da0, the first storage drive).

```
smartctl -t short /dev/da0
```

Then do the same operation for the next drive, and the next, until all the drives are running the short

SMART test. In Fester's case this would be:

```
smartctl -t short /dev/da1

smartctl -t short /dev/da2

smartctl -t short /dev/da3

smartctl -t short /dev/da4

smartctl -t short /dev/da5

smartctl -t short /dev/da6

smartctl -t short /dev/da7
```

Make a note of the time when the last drive will complete the test and then go and get a cup of tea (or in Fester's case training Ferrets to make cheese cake).

When you are certain the last short test, on the last HDD has completed (you will know because you noted the completion time on the last test) then it is time to start the conveyance tests.

SMART Conveyance Tests Via The SSH Console

If you have exited the SSH session then start another and login.

Run the SMART conveyance test on the OS drive labelled ada0 (in Fester's case this is the SSD drive).

At the command prompt type in:

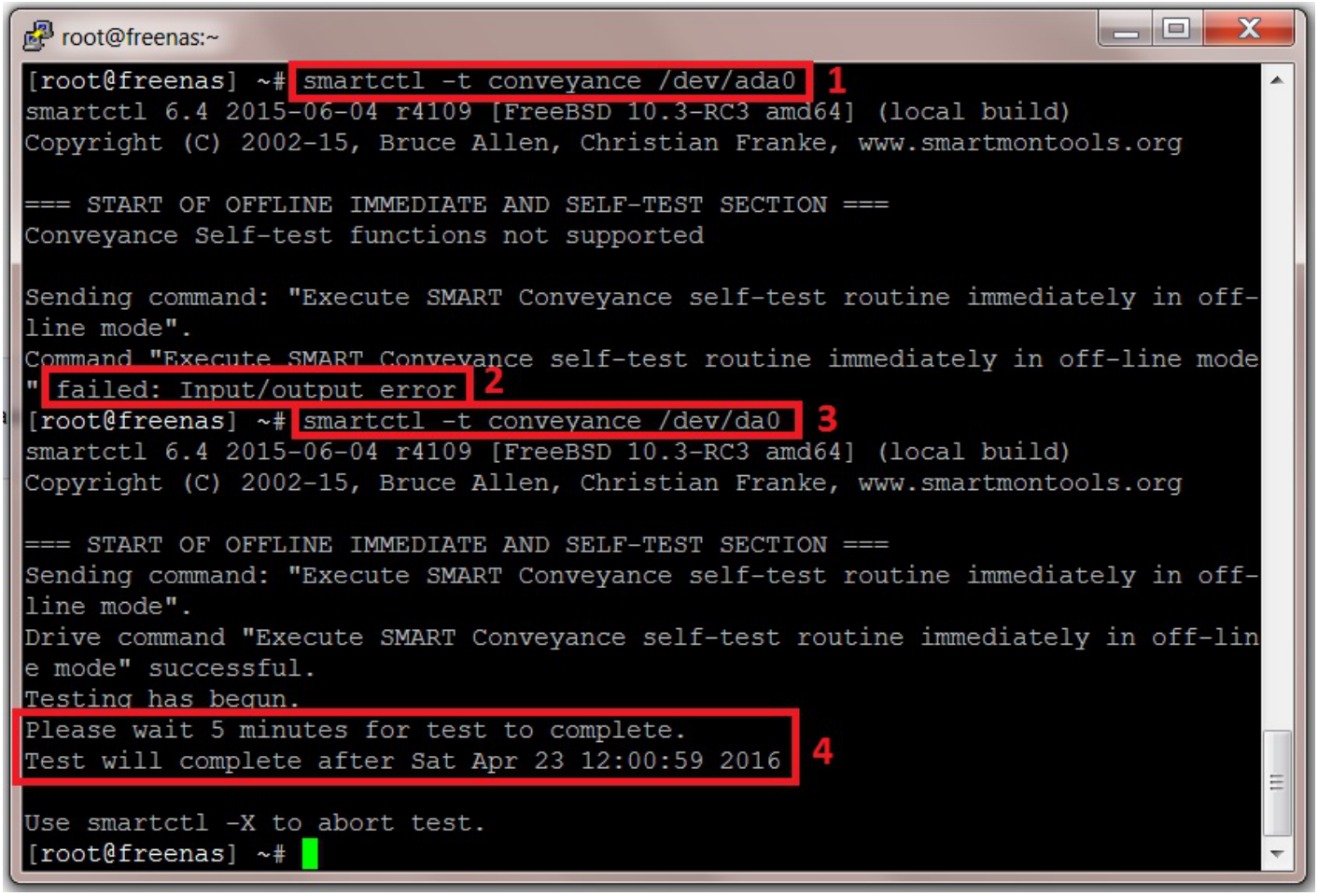
```
smartctl –t conveyance /dev/ada0
```

You should get the following screen, the entered command is shown in the first red box (1). However, the conveyance test failed on this drive due to an input/output error shown in the second red box (2) (some drives don't support conveyance tests, if yours does this just move on to the SMART long test).

So I ran the test on the next drive in the system with the following command:

```
smartctl –t conveyance /dev/da0
```

This is shown in the third red box (3) and now we see how it normally looks when the command is successful. The duration and completion time are shown in the fourth red box (4).



We now need to repeat this process for each drive in the system. We do not need to wait for this drive to complete its test before starting another on a different drive.

So at the command prompt enter the command to start the SMART conveyance test for the next drive in your system (in Fester's case this is da1, the second storage drive).

```
smartctl -t conveyance /dev/da1
```

Then do the same operation for the next drive, and the next, until all the drives are running the SMART conveyance test. In Fester's case this would be:

```
smartctl -t conveyance /dev/da2

smartctl -t conveyance /dev/da3

smartctl -t conveyance /dev/da4

smartctl -t conveyance /dev/da5

smartctl -t conveyance /dev/da6

smartctl -t conveyance /dev/da7
```

Make a note of the time when the last drive will complete the test and then go and get a cup of tea (or in

Fester’s case cleaning cheese cake off the walls, bloody ferrets!).

When you are certain the last conveyance test, on the last HDD has completed (you will know because you noted the completion time on the last test) then it is time to start the long tests.

SMART Long Tests Via The SSH Console

If you have exited the SSH session then start another and login.

Run the SMART long test on the OS drive labelled ada0.

(Fester did not run this test on ada0 because the drive is an SSD drive. A surface scan on an SSD drive is pointless. The reasons why are beyond the scope of this guide and relate to the way in which SSDs handle a bad memory location using the built in hardware manager and over-provisioned memory).

At the command prompt type in:

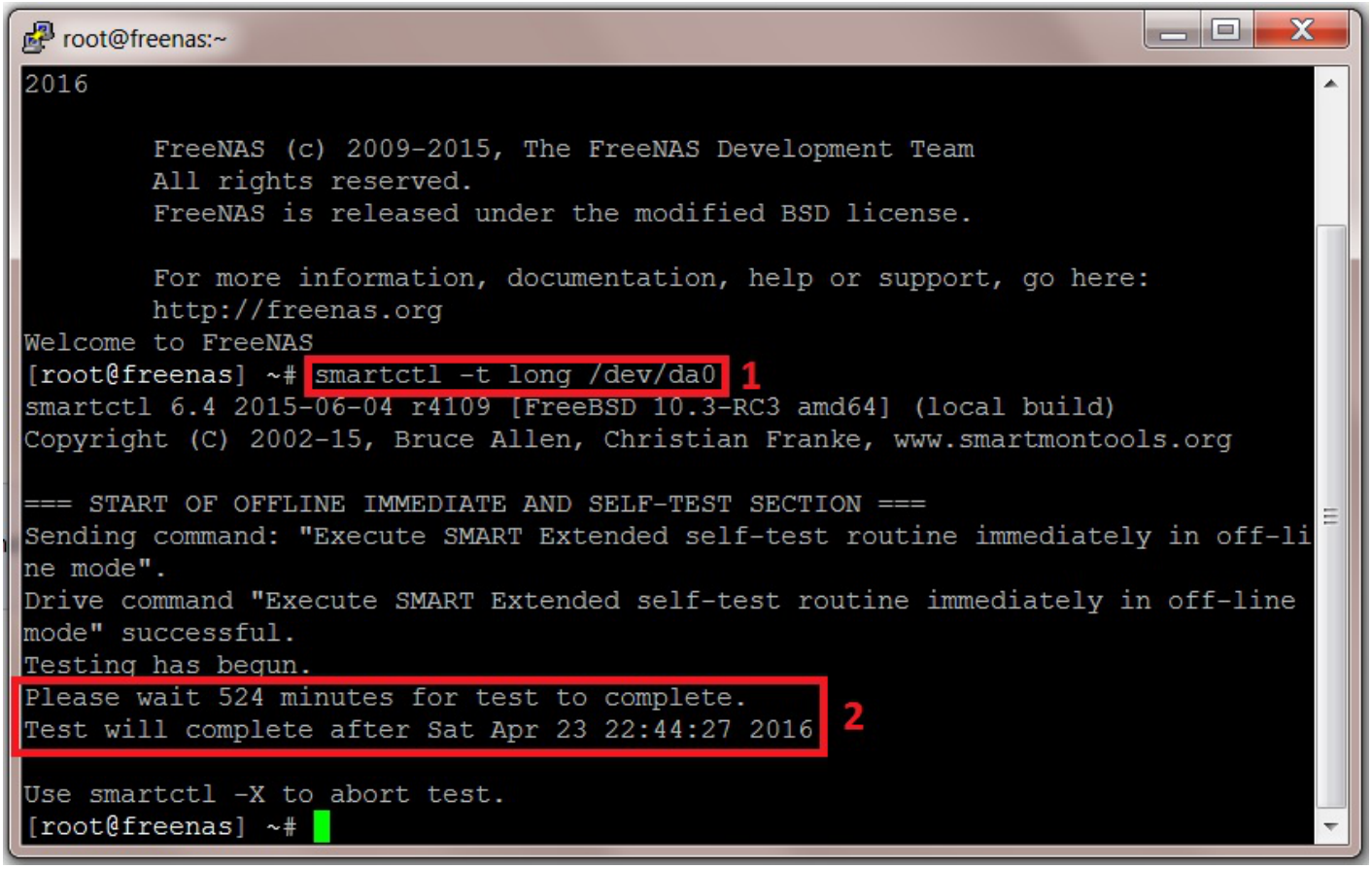
```
smartctl –t long /dev/ada0
```

I can’t show you a screen shot of this on ada0 for reasons I have already explained. So let us go on to the next drive in the system and run the SMART long test on that.

At the command prompt type in:

```
smartctl –t long /dev/da0
```

You should get the following screen, the entered command is shown in the first red box (1) and the duration and completion time are shown in the second (2).



We now need to repeat this process for each drive in the system. We do not need to wait for this drive to complete its test before starting another on a different drive.

So at the command prompt enter the command to start the SMART long test for the next drive in your system (in Fester's case this is da1, the second storage drive).

```
smartctl -t long /dev/da1
```

Then do the same operation for the next drive, and the next, until all the drives are running the SMART long test. In Fester's case this would be:

```
smartctl -t long /dev/da2

smartctl -t long /dev/da3

smartctl -t long /dev/da4

smartctl -t long /dev/da5

smartctl -t long /dev/da6

smartctl -t long /dev/da7
```

Make a note of the time when the last drive will complete the test and then go and get several cups of tea (this one takes a while, most likely several hours).

When you are certain the last long test, on the last HDD has completed (you will know because you noted the completion time on the last test) then it is time to start the badblocks tests.

Badblocks Tests Via The SSH Console

The Badblocks test differs from the SMART tests in important ways.

Unlike the SMART test it is not a self-test. If is done using a piece of software built into the FreeNAS OS (it's actually part of FreeBSD which FreeNAS is built on).

This means if we end the SSH session we also terminate the Badblocks test. Due to the long period of time these tests take to complete it becomes seriously inconvenient to keep an SSH session open that long.

Another problem that occurs is when we start the Badblocks program we can no longer input commands into the SSH command prompt until Badblocks completes its test. Therefore, we cannot run Badblocks tests in parallel on different drives (unlike the SMART tests that can run concurrently).

This means we would need to run one Badblocks test at a time on each drive consecutively (i.e. run Badblocks on one drive and wait for that to complete. Then run it on the next drive and wait for that to complete, until all the drives had been tested).

Considering that this test can take anything from 24 hours to 2 – 4 days depending on the capacity of the drive, then the Badblocks test on an 8 drive system would take an inordinate amount of time (assuming 1 drive takes 3 days, an 8 drive system would take 3 x 8 = 24 days!).

So when conducting these tests we will use **tmux** which is a session multiplexer built into FreeNAS. A session multiplexer is a console that is capable of running more than one session at the same time. This means we can now run multiple instances of Badblocks in parallel on different drives (this reduces the 24 days to just 3 days).

Also when we close the SSH console, tmux on the FreeNAS system is kept open. It only closes properly when we formerly exit the tmux session. This means we do not need to keep the SSH console open for 3 days on the client computer (very convenient).

However, there are some caveats to be aware of when using tmux.

FreeNAS Storage Volumes And A Known Problem With badblocks

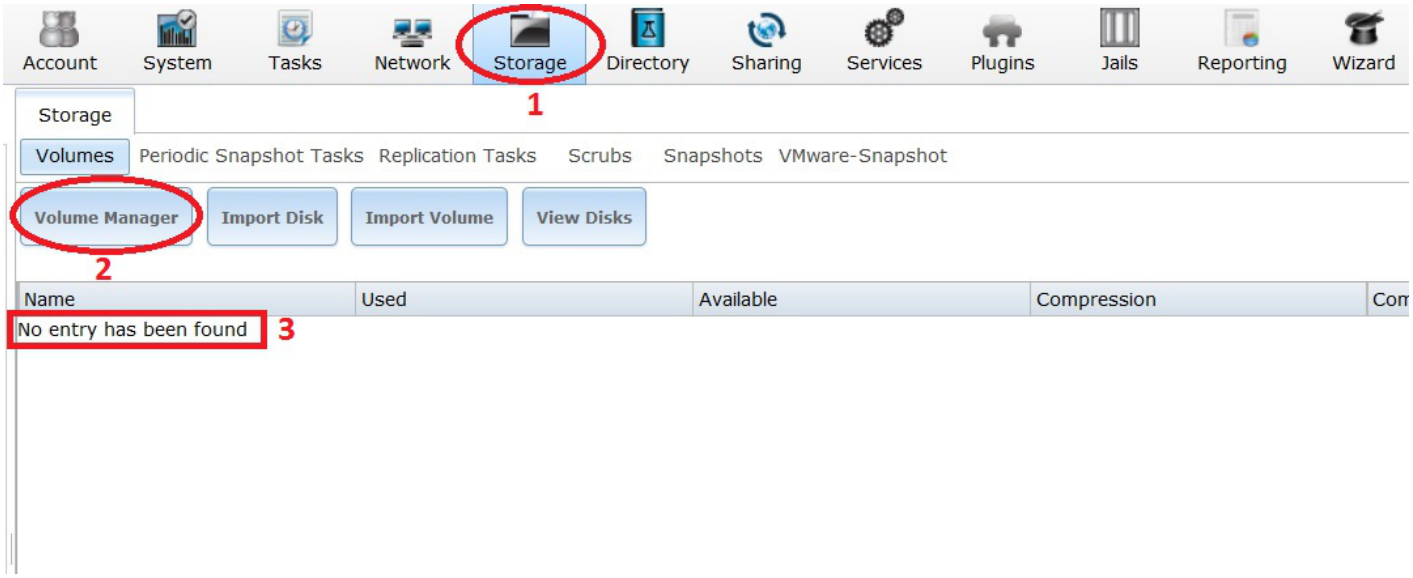
If you have a volume (this is a Zpool) created on the server using the "Volume Manager" in the FreeNAS web GUI, then it is essential to detach the volume before commencing any Badblocks tests.

This is because the FreeNAS OS does a series of small short writes to the volume (Fester does not know the how or why of this, if someone wants to provide some information I will try to include it in the guide so everyone can benefit).

This activity will mess up the Badblocks tests!

This is how to check if your system has a volume.

- Go to the “Storage” page (1).
- Click on the “Volume Manager” button (2).
- If you see text that states “No entry has been found” (3) then your system has no volume and you are good to go with the Badblocks tests.



However, if your system has a volume then you must detach it before continuing.

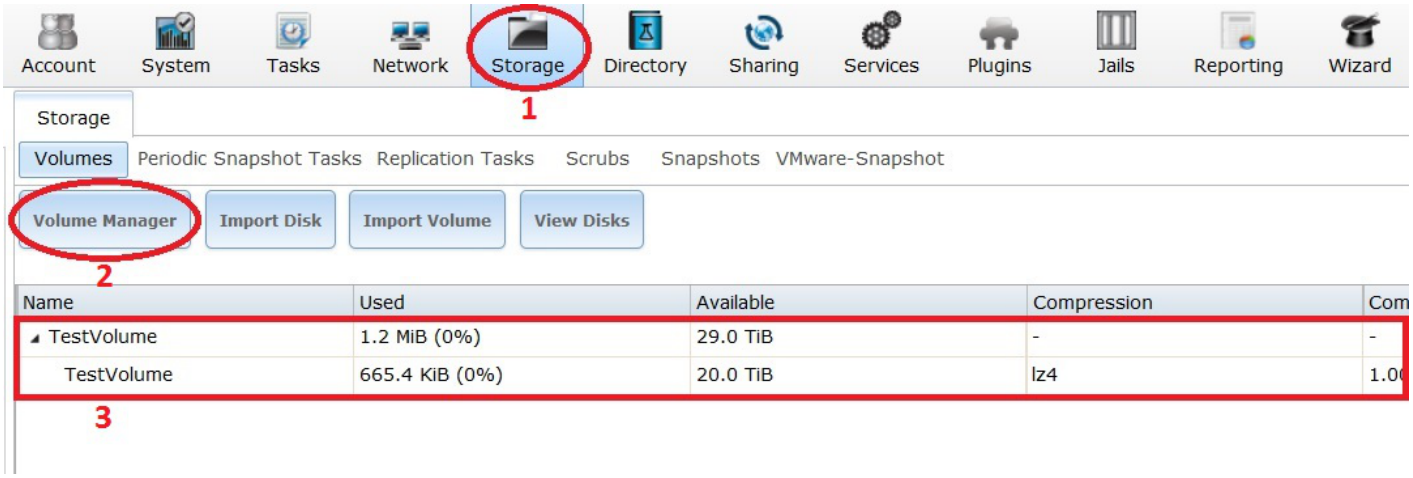
Detaching The Storage Volume Before A Badblocks Test (DESTRUCTIVE Method)

This is the DESTRUCTIVE method of how to detach a volume.

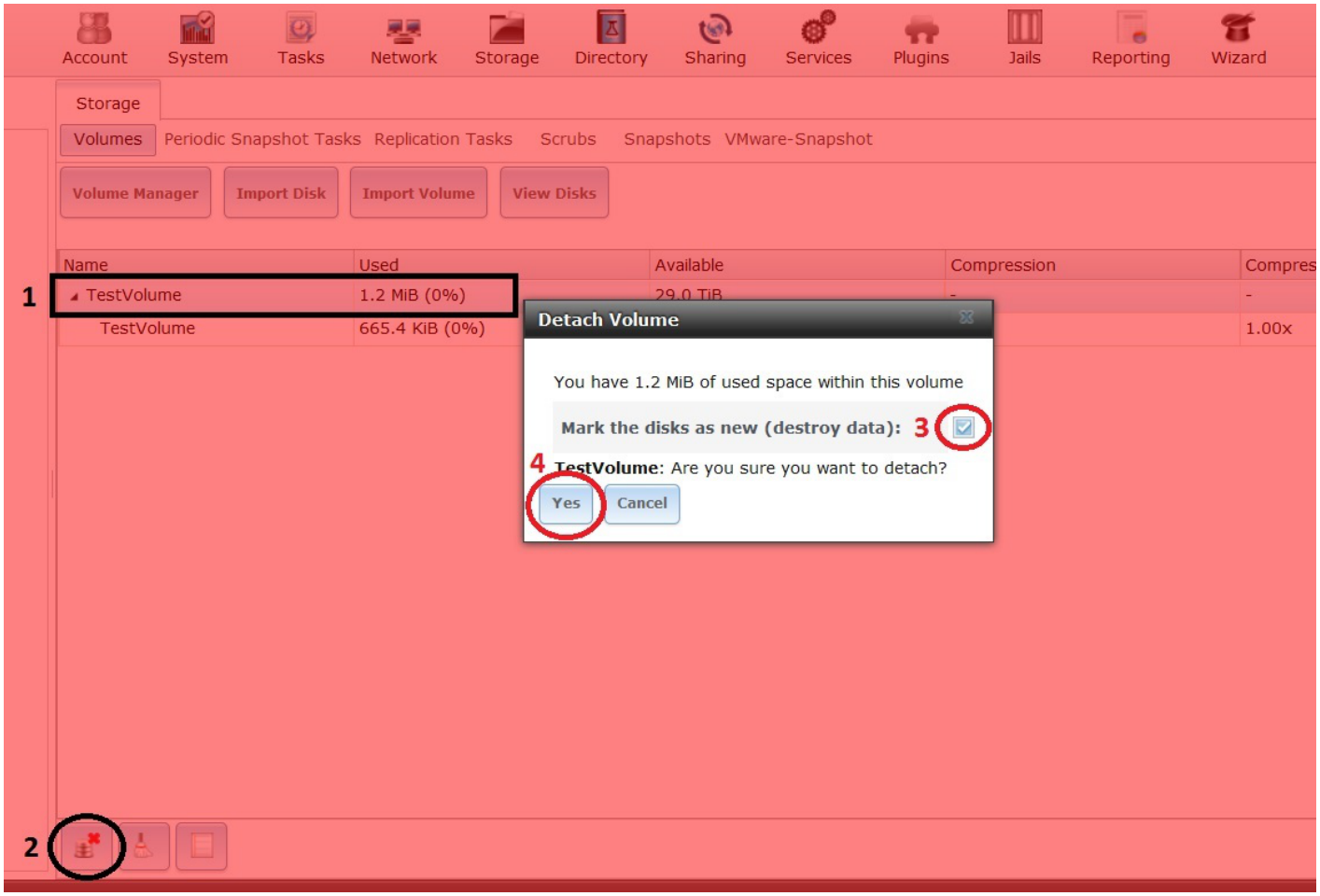
This means that any and all data on the storage disks will be destroyed forever.

(Which according to the latest scientific research is apparently a long time!)

- Go to the “Storage” page (1).
- Click on the “Volume Manager” button (2).
- If you see entries similar to the screen shot (Fester’s volume is called “TestVolume”) (3) then your system has a volume which you must detach before you are good to go with the Badblocks tests.

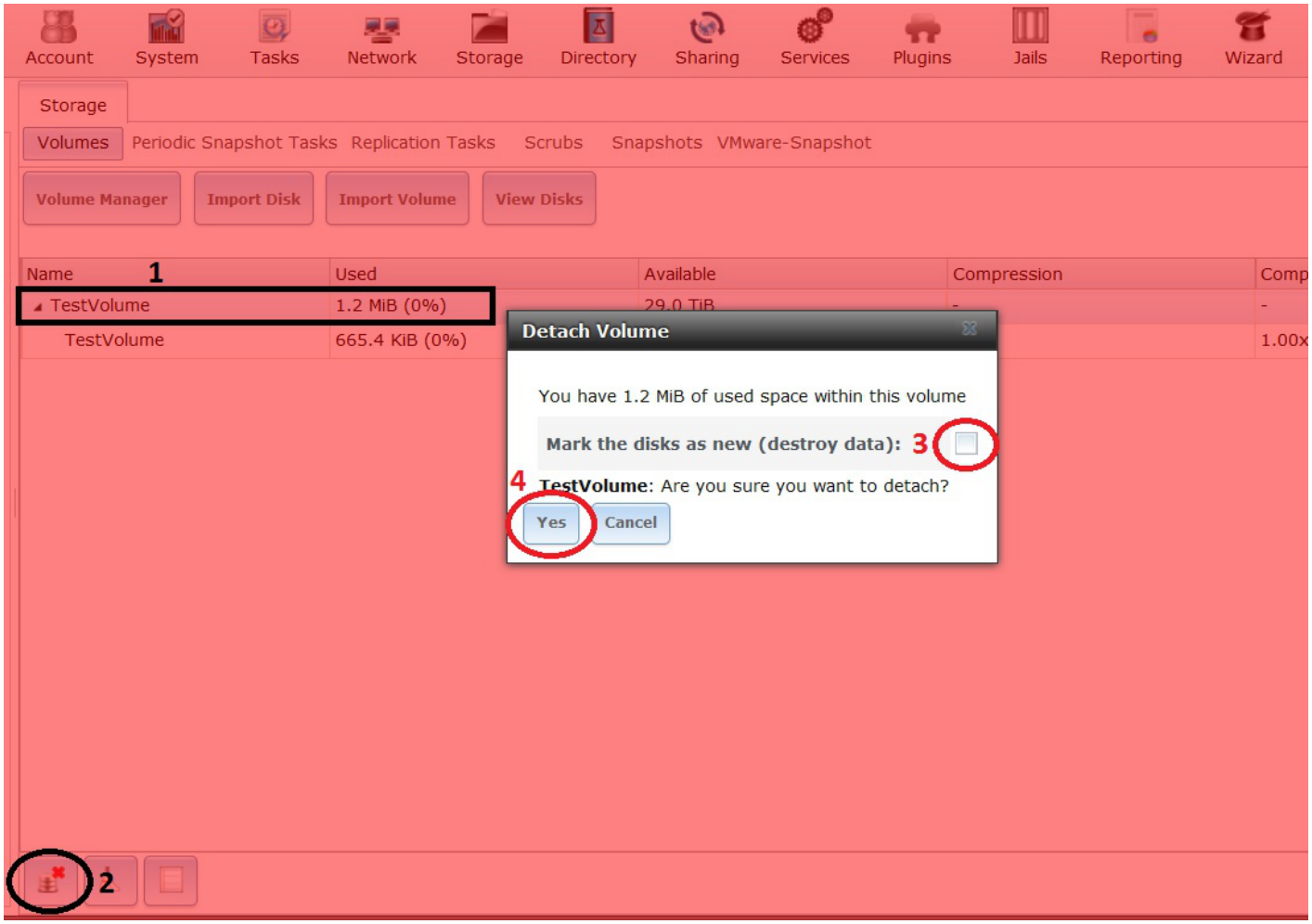


- To detach the volume select it by clicking on it (it will turn blue when this is done) (1).
- Now click the “Detach Volume” button (2).
- Then tick the “Mark the disks as new (destroy data):” tick box (3)
- (THIS WILL DESTROY ANY AND ALL DATA YOU MAY HAVE ON THE STORAGE DRIVES, DON’T DO THIS IF YOU HAVE DATA YOU WISH TO KEEP).
- Now click the “Yes” button (4).



Detaching The Storage Volume Before A Badblocks Test (Non-Destructive Method)

- To detach the volume select it by clicking on it (it will turn blue when this is done) (1).
- Now click the “Detach Volume” button (2).
- DO NOT TICK the “Mark the disks as new (destroy data):” tick box (3)
- Now click the “OK” button (4).

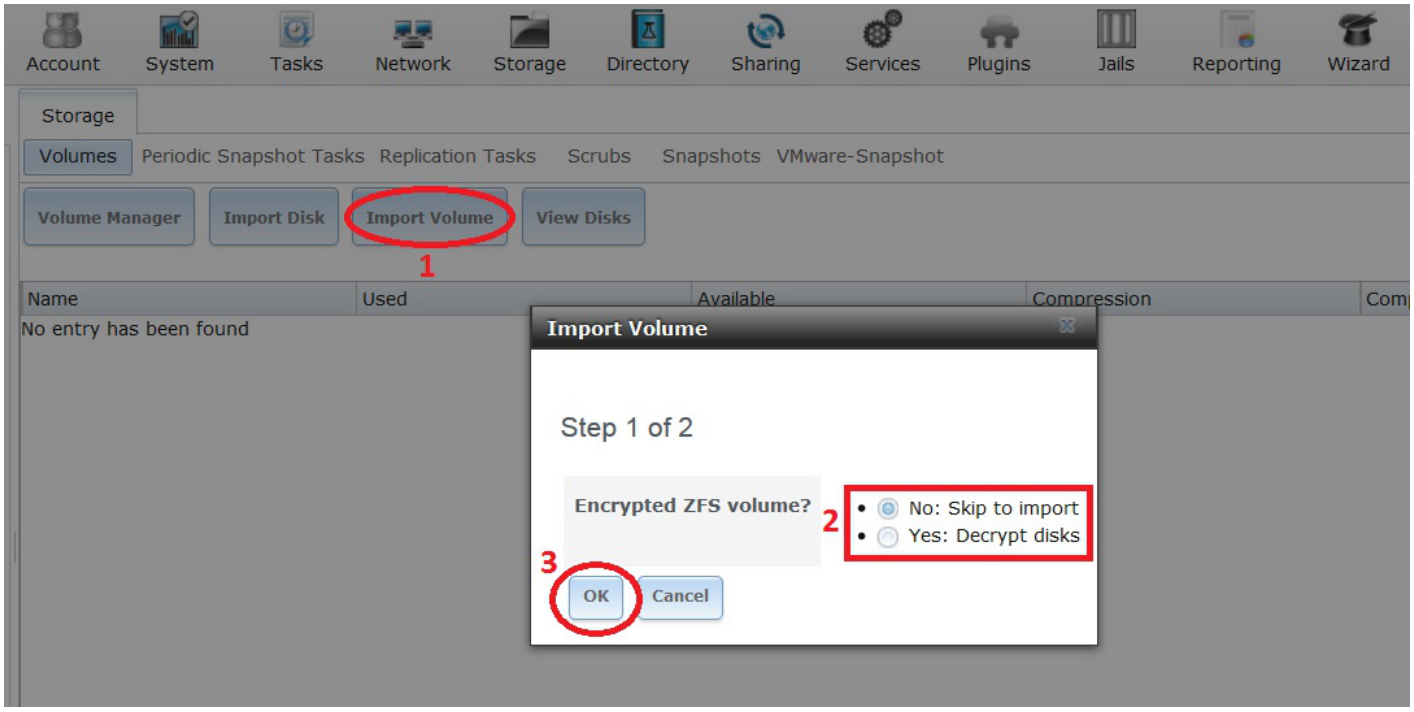


When you have carried out the **non-destructive** version of the Badblocks test (more on this in a moment) you will then need to reattach the volume.

Importing An Unencrypted Volume After A Non-Destructive Badblocks Test

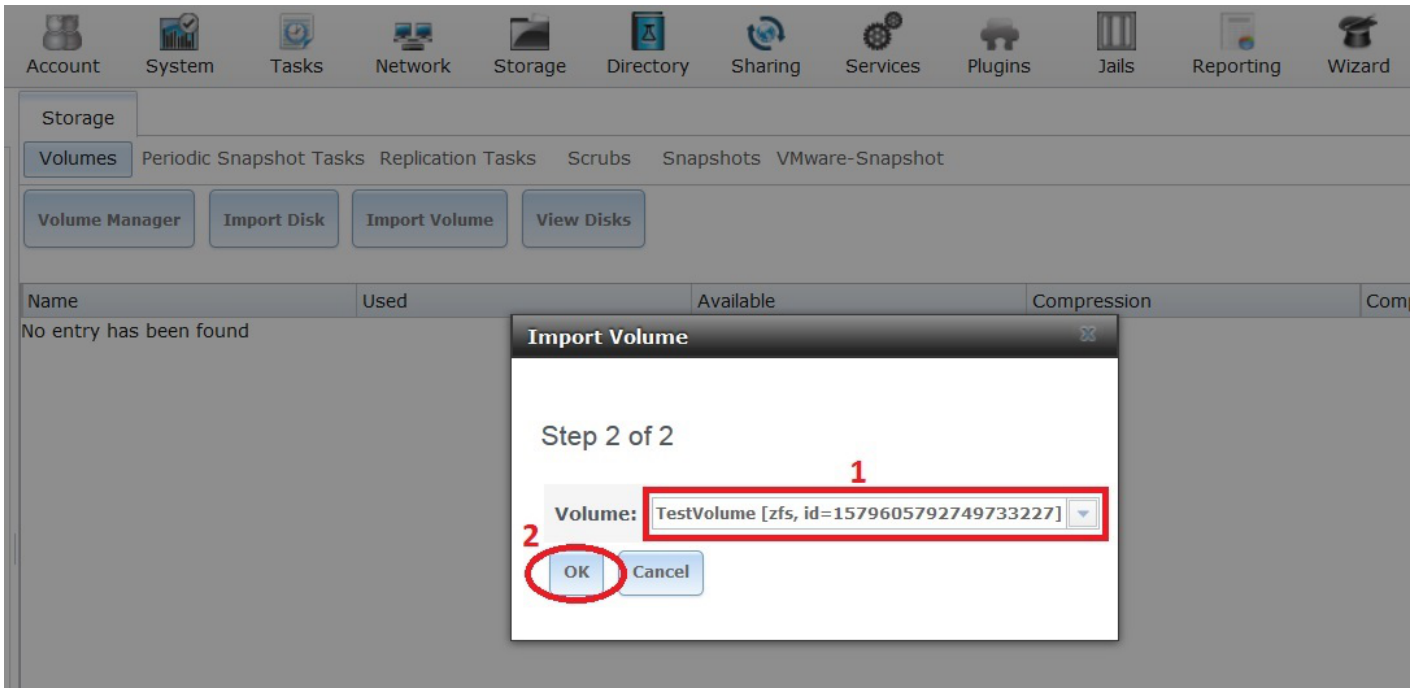
This is how to reattach a non-encrypted volume in the FreeNAS web GUI.

- Assuming you have selected the “Storage” page click on the “Import Volume” button (1).
- If the volume is not encrypted then click the “No: Skip to import” radio button (2).
- Now click the “OK” button (3).



This will take you to a second screen and step 2 of a 2 part process.

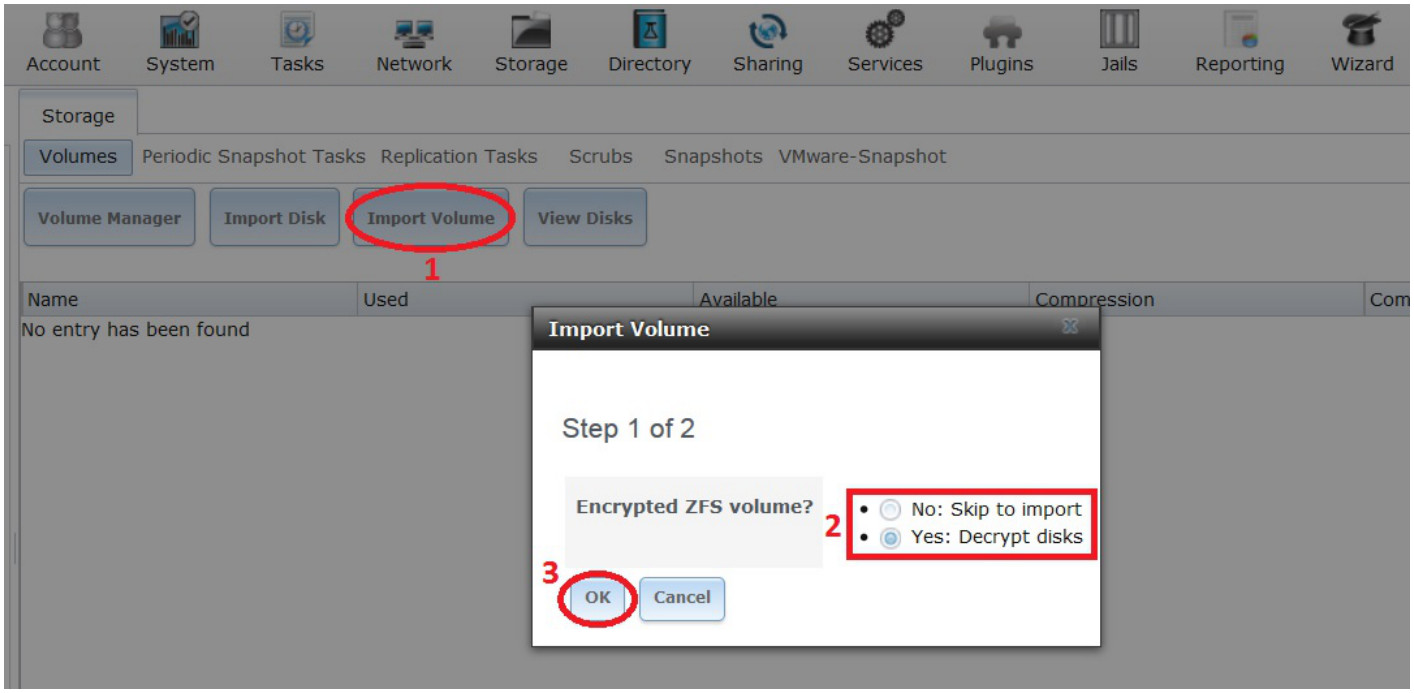
- In the “Volume:” drop down selection box (1) you should see your previously detached volume.
- With the correct volume selected click the “OK” button (2) and the volume should be imported momentarily.



Importing An Encrypted Volume After A Non-Destructive Badblocks Test

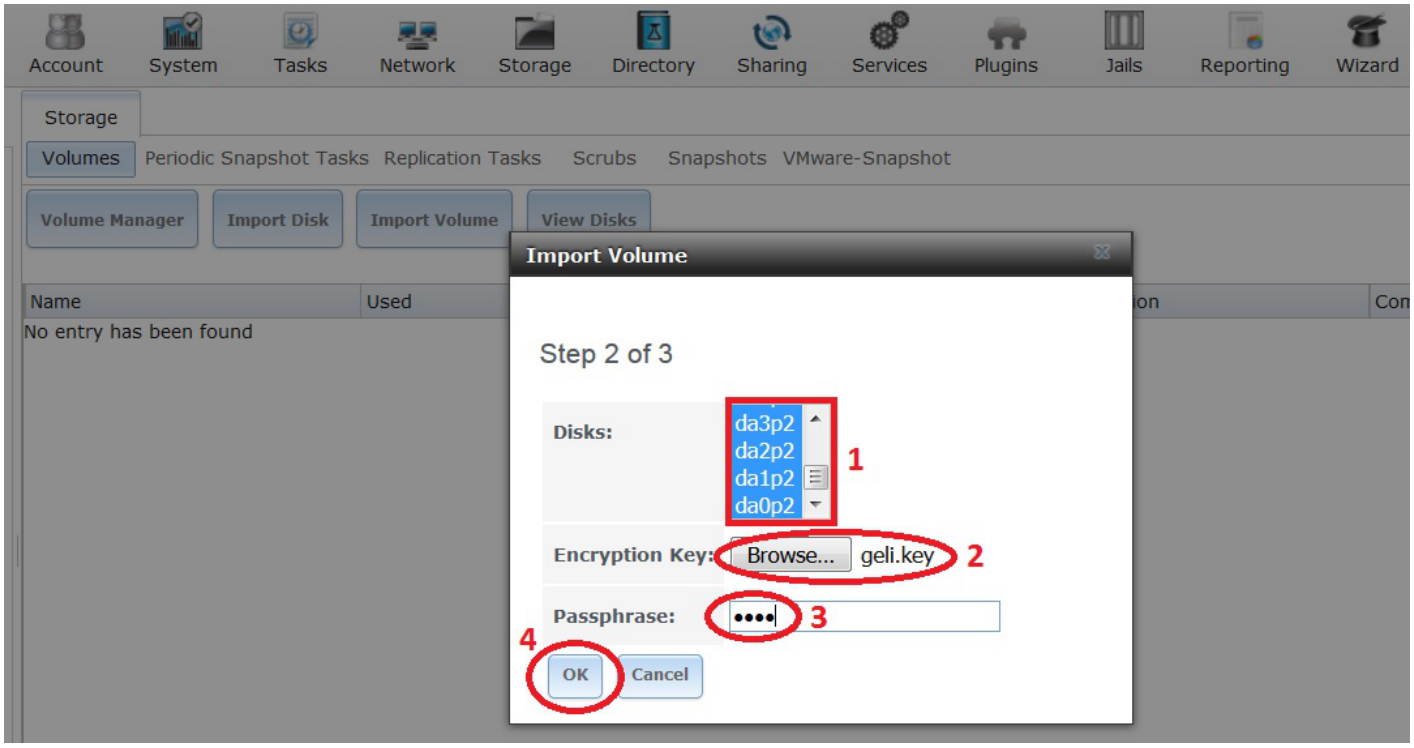
This is how to reattach an encrypted volume in the FreeNAS web GUI.

- Assuming you have selected the “Storage” page click on the “Import Volume” button (1).
- If the volume is encrypted then click the “Yes : Decrypt disks” radio button (2).
- Now click the “OK” button (3).



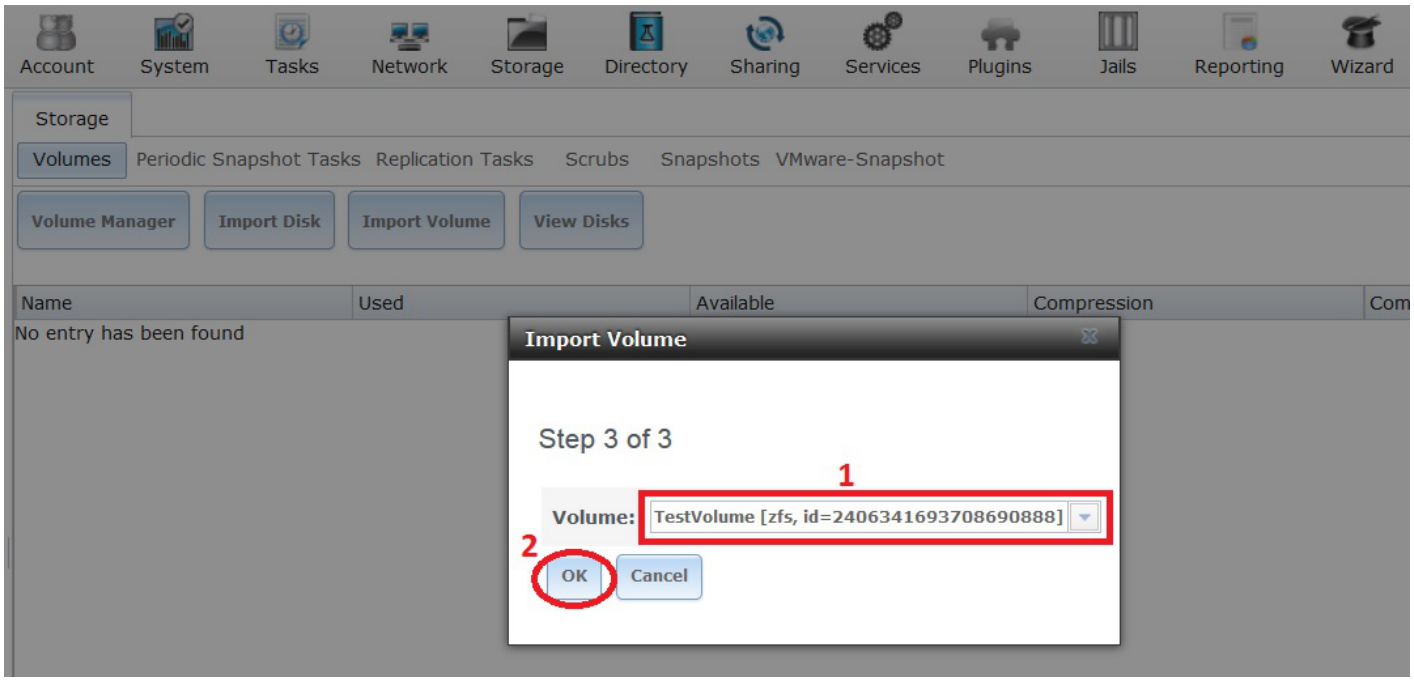
This will take you to a second screen and step 2 of a 3 part process.

- Select the disks that form the volume from the “Disks:” window (1) (in Fester’s case this was all of them).
- Now click the “Browse” button (2) and a window will pop up that allows you to load in your previously saved geli key (when creating encrypted volumes always make sure you save a recovery key).
- Navigate to the location of your key and load it into the FreeNAS GUI. If all goes well you should see it next to the “Browse” button (Fester’s shows up as geli.key) (2).
- Now type in the passphrase (which is a password you created when you made the encrypted volume), in the text box next to “Passphrase:” (3) (Fester very imaginatively used **test** here).
- Now click the “OK” button (4).



The third and final screen will now appear.

- In the “Volume:” drop down selection box (1) you should see your previously detached volume.
- With the correct volume selected click the “OK” button (2) and the volume should be imported momentarily.



Destructive Badblocks Test Using tmux

Start an SSH session and log in.

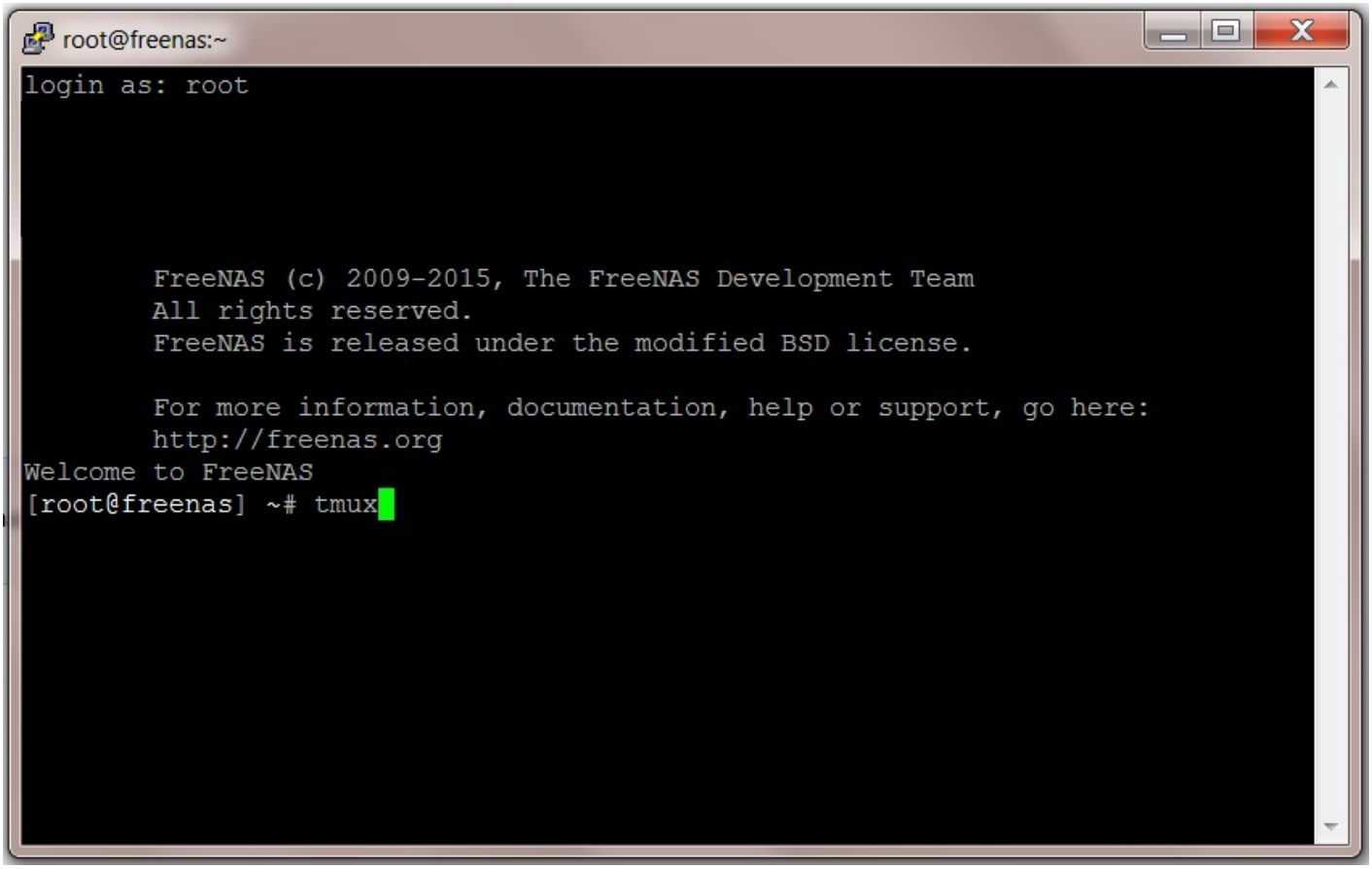
Before starting tmux we need to enable the kernel geometry debug flags, so type in this command at the command prompt.

```
sysctl kern.geom.debugflags=0x10
```

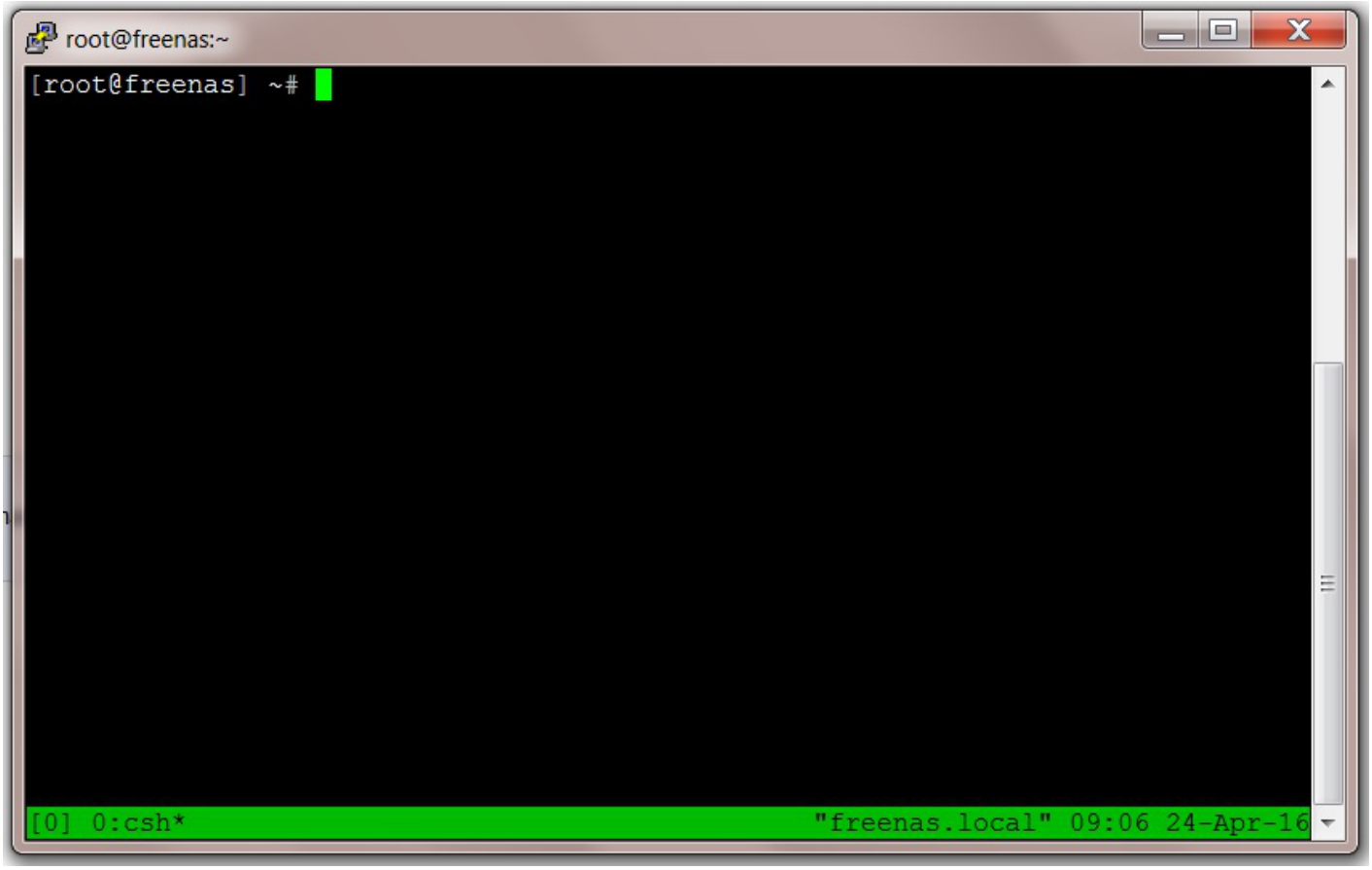
(When all the Badblocks tests are done the kernel geometry debug flags must be returned to their normal state. Thankfully no additional command is necessary, just reboot the server as this setting is not persistent and cannot survive the reboot.)

Now type the following command at the command prompt.

```
tmux
```



You should see a screen something like this. Notice the green band at the bottom of the screen, this is a tmux session.



I will not be running the Badblocks test on ada0 (the SSD drive) there is no point as already explained and this is a destructive test (the FreeNAS OS is on this drive!).

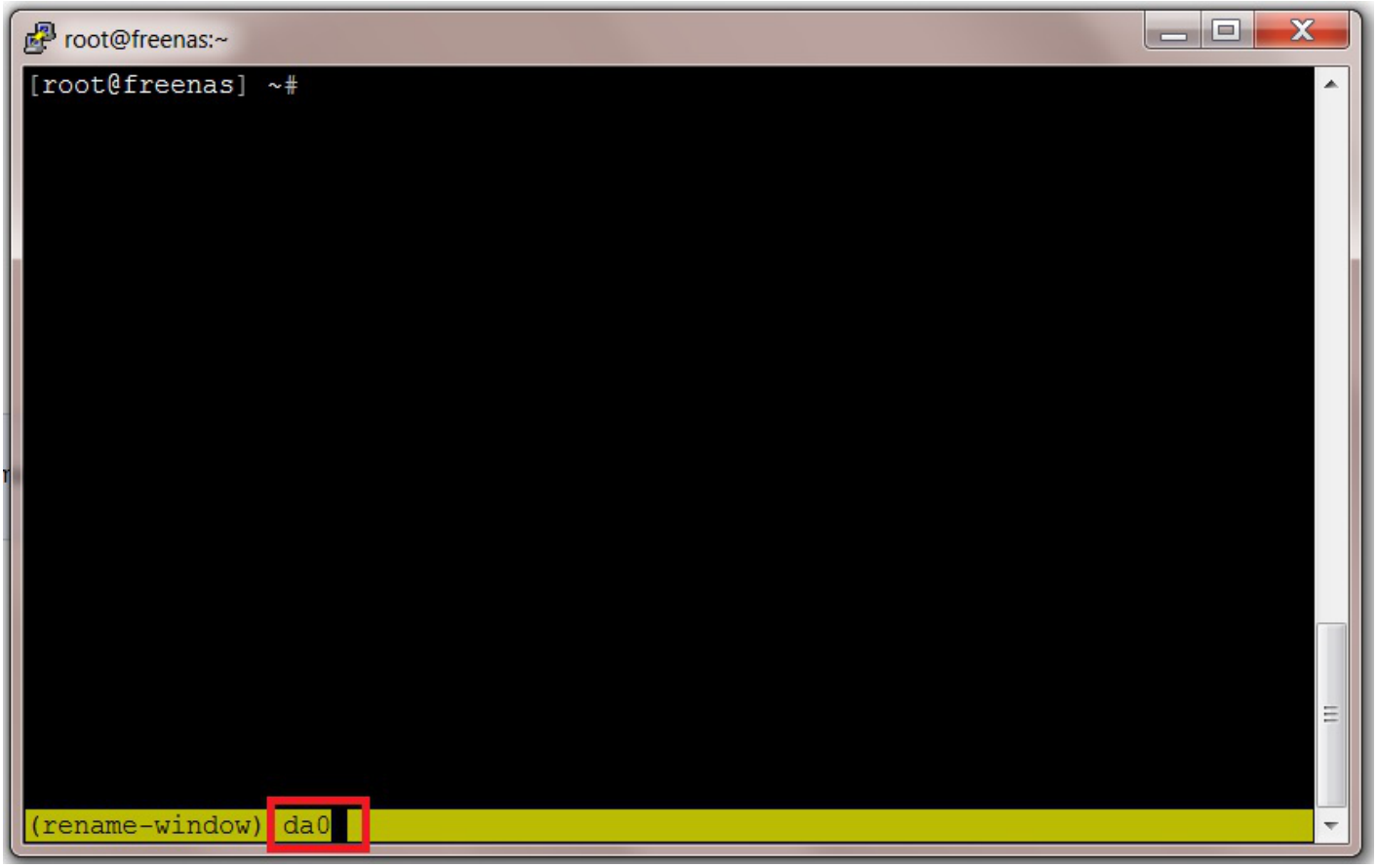
This leaves the 8 data storage drives to check.

This means I will need 8 sessions opened in tmux (open the number of sessions that suits your requirements).

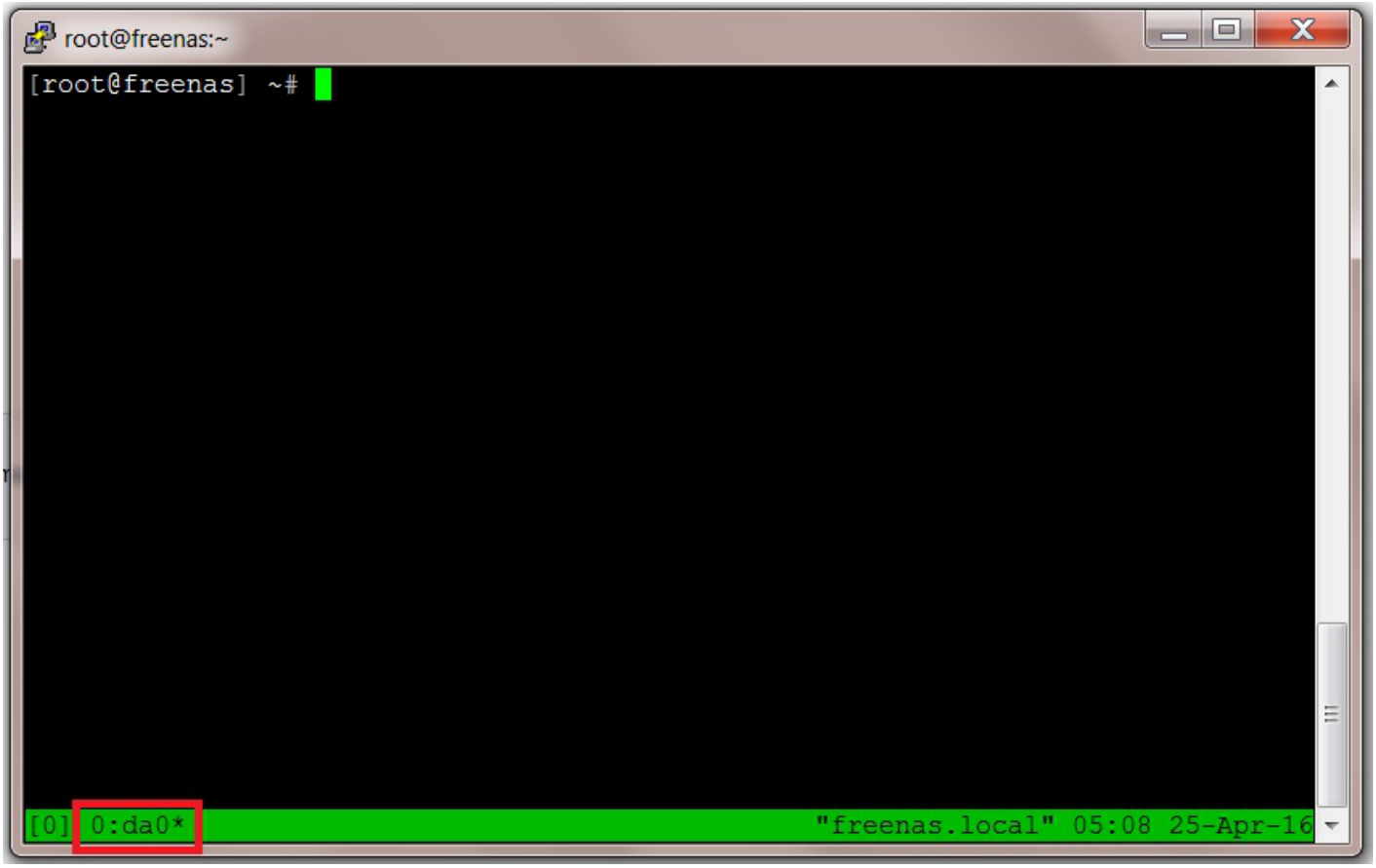
Let us start by renaming the current session in tmux to something more meaningful than “csh”.

In the tmux window press the “Ctrl” and “b” keys together, release them and then press the “,” key.

The bar at the bottom of the window should turn yellow and you can now delete the “csh” text and rename it (Fester called his “da0” after the drive that will be tested).



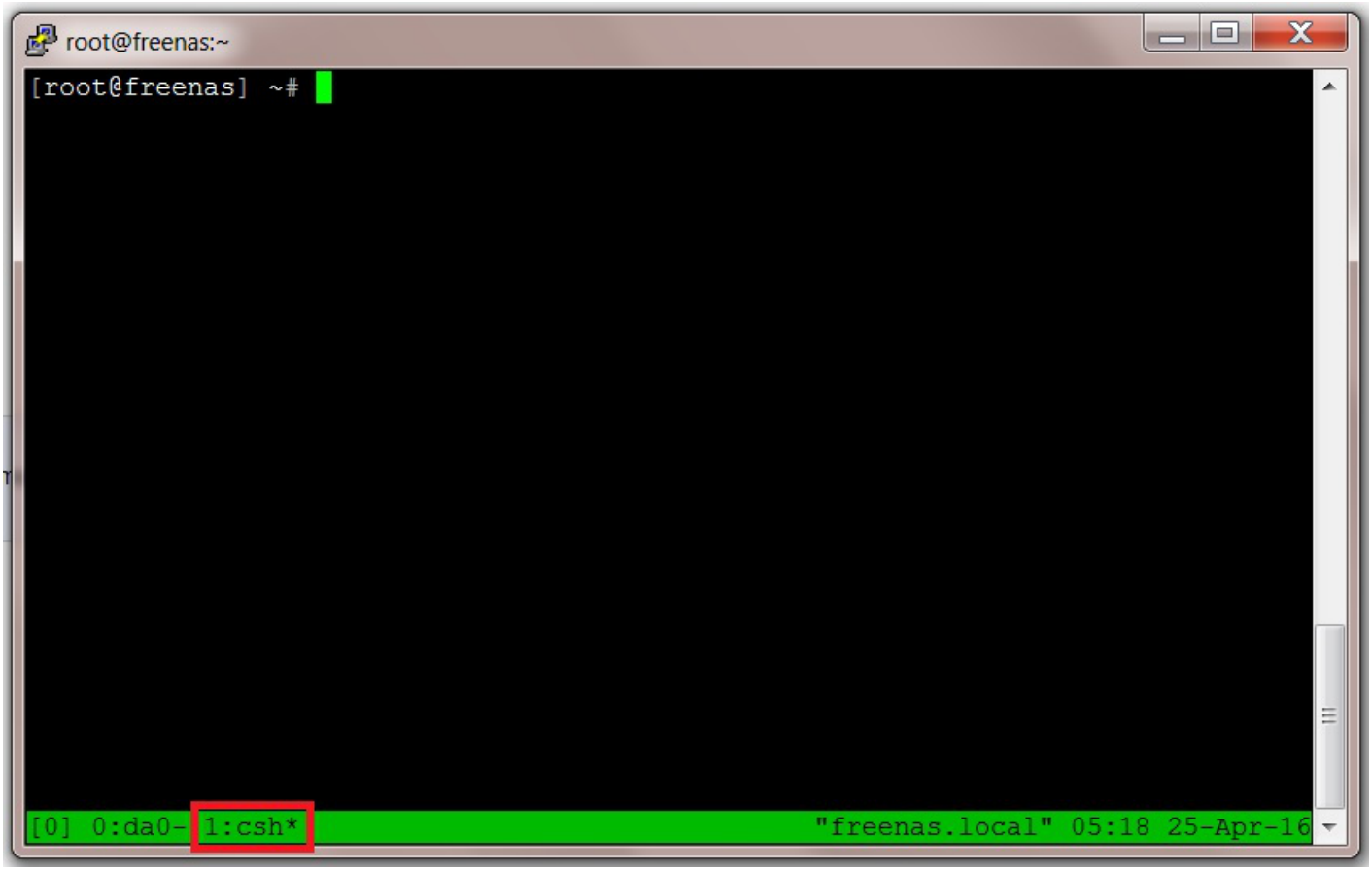
When you have typed in the new name press the “Return/Enter” key, the bar should now resort back to its original green colour and the session should be renamed.



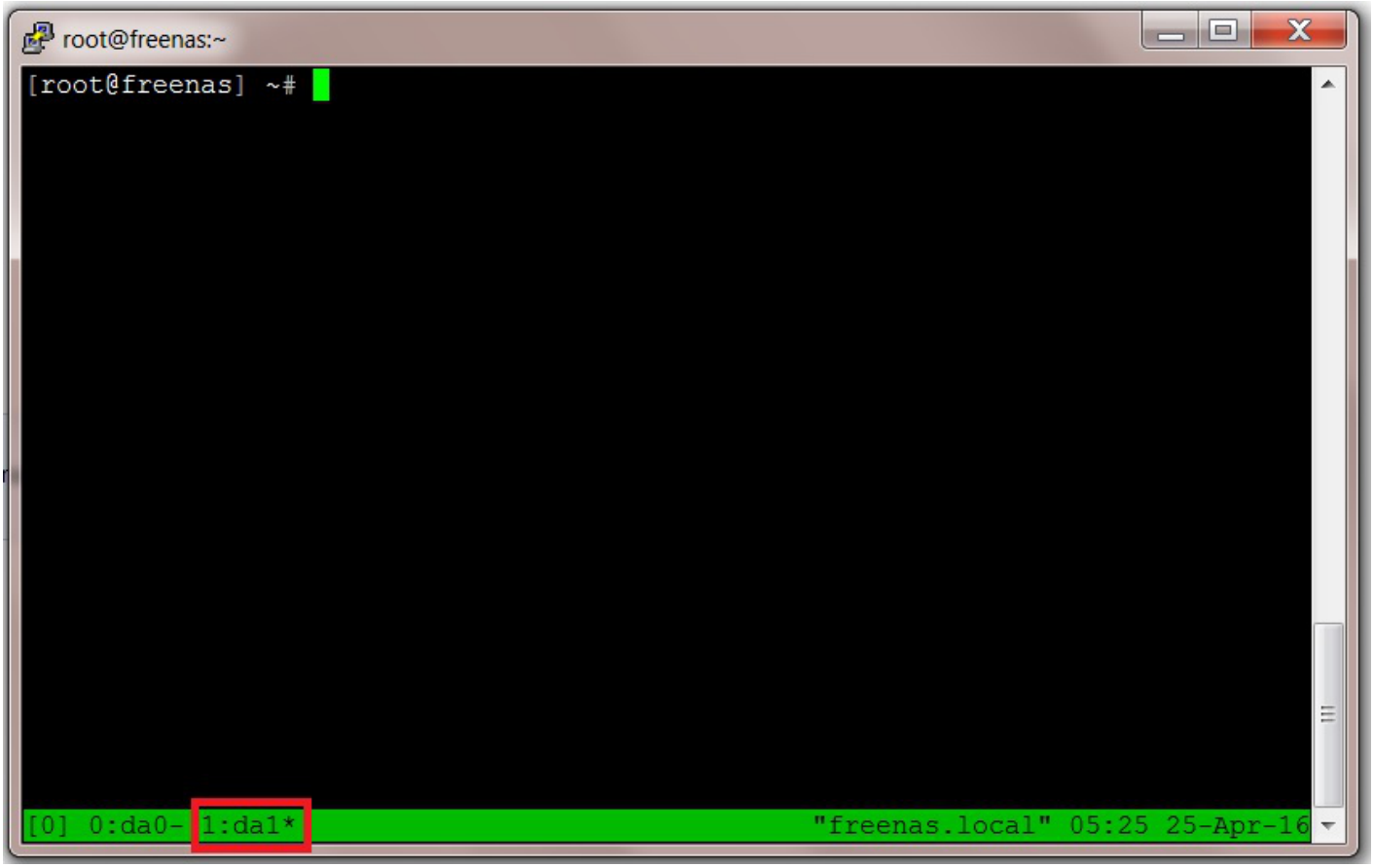
At this point we need to create an additional session and rename it for the next drive to be tested.

To create a new session press the “Ctrl” and “b” keys together, release them and then press the “c” key.

You should get something like this where “1:csh*” is the newly created session. Incidentally the asterisk just denotes the currently selected session.



Let us rename this session by pressing the “Ctrl” and “b” keys together, releasing them and then pressing the “,” key. Type in the new name and press the “Return/Enter” key (just as we did before, I called this one “da1” after the next drive to be tested).

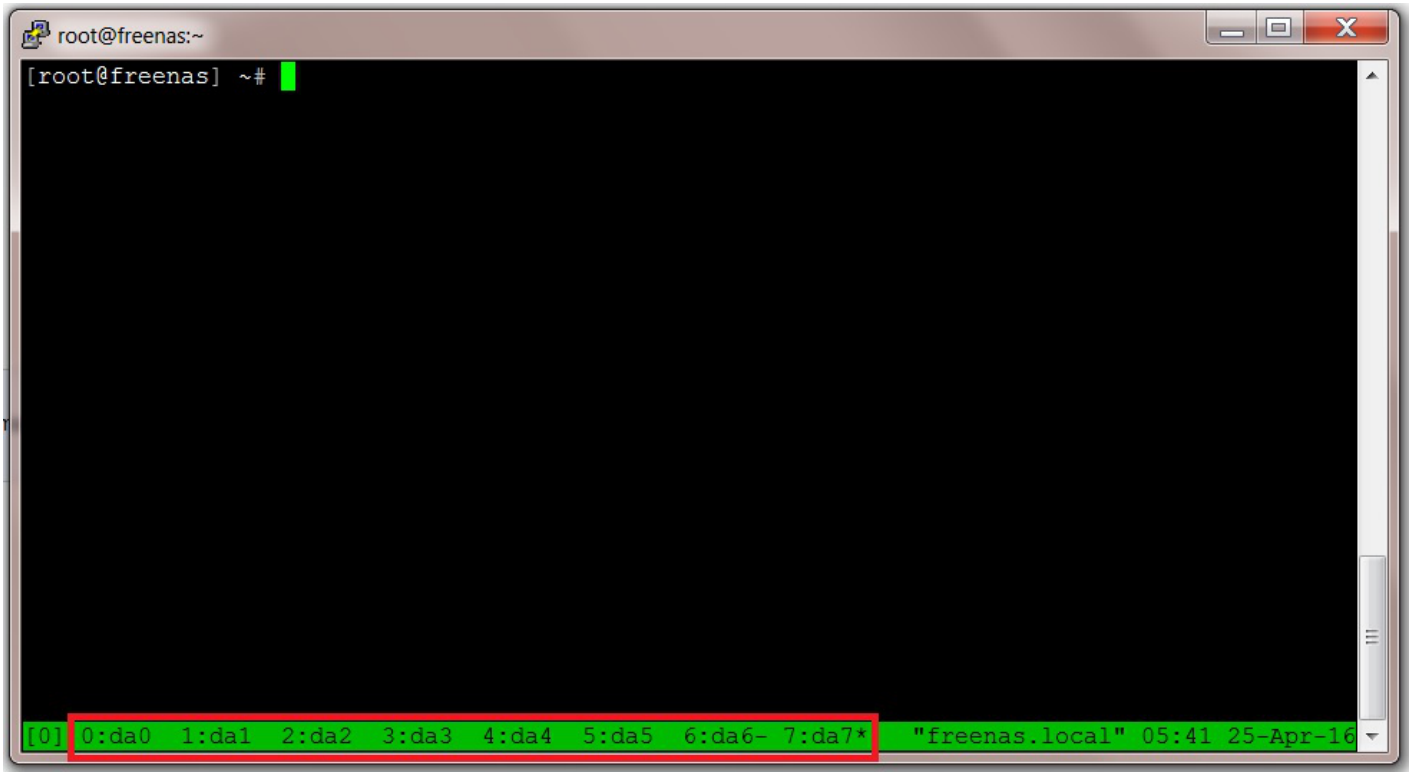


Navigation between the different sessions is achieved by pressing the “Ctrl” and “b” keys together, releasing them and then pressing the “n” key. This will take you to the next session along.

Alternatively you can also press the “Ctrl” and “b” keys together, release them and then press the “p” key. This will take you to the previous session.

By using the next and previous navigational keystroke combinations you can navigate through the different sessions, the asterisk signifying which session you are currently viewing.

Using the key combinations already explained let us create the remaining sessions needed and rename each one.



Now we can run the Badblocks tests from within tmux.

Navigate to the first session (i.e. “0:da0”) and type in the following command at the prompt.

```
badblocks -ws /dev/da0
```

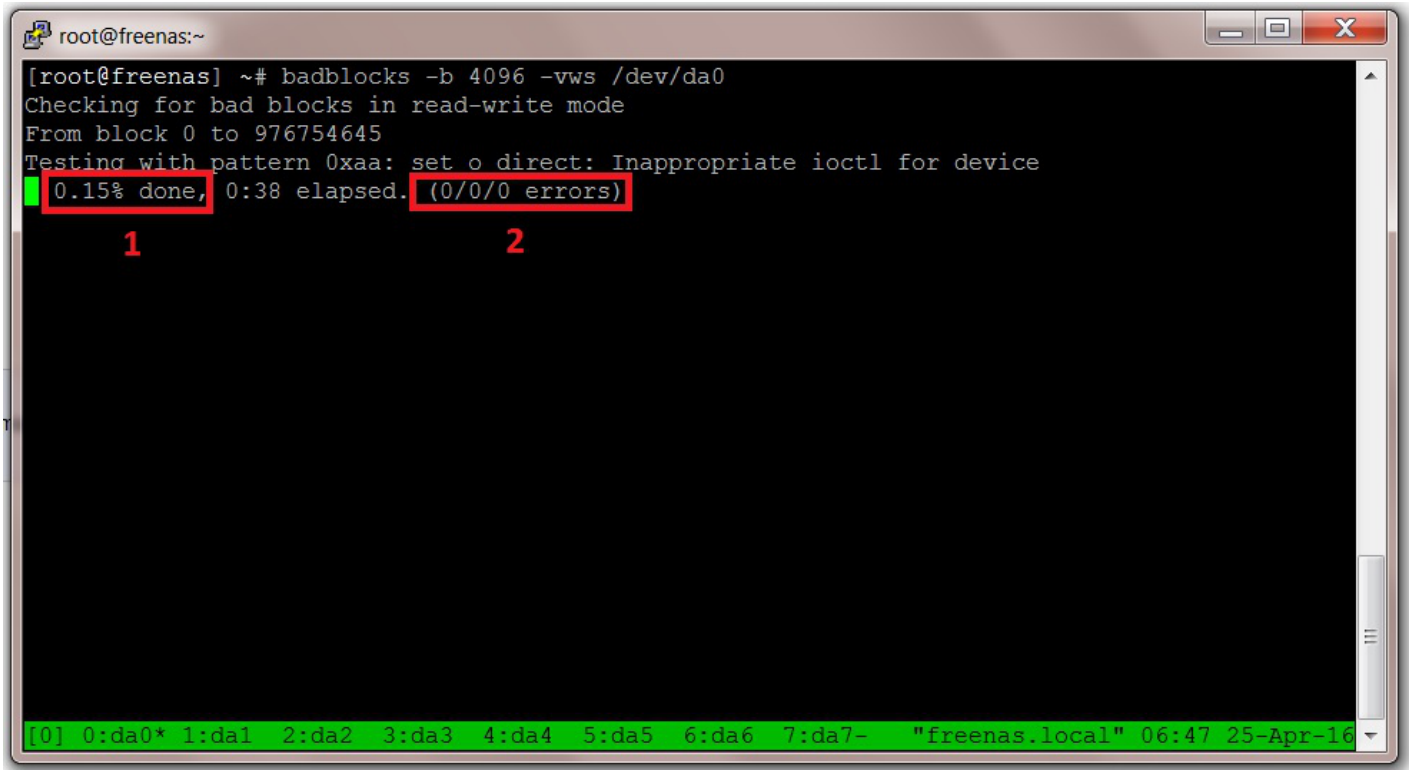
Fester uses a slightly different command to improve the efficiency of the tests with the WD40EFRX drives. These drives have a sector size of 4096 bytes (even though they report 512 bytes, naughty Western Digital). I also like a more verbose output from these tests so the command includes the –v switch. I include it here for informational purposes only. In addition to improving efficiency, the “-b 4096” below is required with larger disks (*i.e.*, larger than about 4 TB).

```
badblocks –b 4096 –vws /dev/da0
```

If the command executes properly you should see something like this.

You will see from the screen the completion progress expressed as a percentage (1) and any errors that have occurred expressed like this “(0/0/0 errors)” (2).

There should be zero errors throughout the test. If you get even one error then you should return the disk for testing.



Now navigate to the next session (in Fester's case that is "1:da1") and type this at the command prompt.

```
badblocks -ws /dev/da1
```

(Or Fester's variation if it suits you better, but remember to change the drive name from "da0" to "da1".)

Repeat this process of changing session and running the Badblocks command for every drive in your system that you want to test. In Fester's case this means running these commands while changing sessions each time.

```
badblocks -ws /dev/da2

badblocks -ws /dev/da3

badblocks -ws /dev/da4

badblocks -ws /dev/da5

badblocks -ws /dev/da6

badblocks -ws /dev/da7
```

Non-Destructive Badblocks Test Using tmux

Stopping A Badblocks Test In tmux

Resuming A Session In tmux

Getting Your Test Results

Making Sense of SMART Data