A Quick Guide to iSCSI on Linux

Cuddletech TekRef Series

Ben Rockwood, Cuddletech <benr@cuddletech.com>

Revision History

Revision v1.0	April 11th 2004	benr
	Initial Document	
Revision v1.1	April 12th 2004	benr
	Clarifications and Cisco Ref added	
Revision v1.2	April 23th 2004	benr
	Fixes, Typos, and Various Additions	
Revision v1.3	June 7th 2004	benr
	Added Solaris procedure	
Revision v1.4	Aug 6th 2004	benr
	Added sidebar for IET, ardis to be removed soon	

A quick and gentle overview of iSCSI on Linux. Setting up both an iSCSI initiator and iSCSI target on a Linux system are covered with a quick overview of essential iSCSI terms and ideas.

Table of Contents

Introduction

More and more it appears that iSCSI isn't going away, but might just be here to stay for awhile. In a nutshell, iSCSI is simply the pairing of the "best" of both NAS (using NFS or

CIFSs over IP Ethernet networks) and SAN (Fibre Channel Fabrics) technologies. What you ultimately get is a protocol that allows you to use SCSI commands like Fibre Channels FCP, yet does it over an IP network instead of a costly Fibre Channel Fabric. Instead of buying an expensive Brocade or McData switch and costly Fibre Channel HBAs from companies like JNI, Adaptec and Emulex, you can use any IP switch (NetGear, 3Com, Extreme, Foundry, etc) and normal Ethernet cards. Because SCSI is CPU intensive during high I/O loads iSCSI Host Bus Adapters (HBA) have arrived which act just like a FC HBA except that it uses Ethernet instead of FC Fabric; the idea being that the SCSI requests are offloaded from your primary CPU onto the ASIC on your iSCSI HBA.

SCSI uses a client-server architecture. A "client" (ie: your system) is an initiator, it initiates requests. A "server" (ie: your storage device) is a target, it has something you want and answers requests from the initiator(s). Initiators come in two varieties: software and hardware. A software initiator is just a driver that handles all requests and pairs the network interfaces driver and the SCSI drivers together to make it all work. Using a software initiator any system with an ethernet card can act as an iSCSI initiator. A hardware initiator is an iSCSI HBA, which is basically just an ethernet card with a SCSI ASIC onboard to offload all the work from the system CPU. Because there is no cost involved in using a software initiator we'll look at that. Adaptec is currently selling iSCSI HBA's which have Linux drivers available if you choose to go that route.

There are three very interesting projects in the Linux community regarding iSCSI at the moment. The first is a project from the University of New Hampshire's Inter-op Labs (legendary in Fibre Channel circles). Their iSCSI implementation provides both software initiator and target code for use with a Linux kernel. While it's an interesting project, they admit that their target code is really only around for testing their initiator code, and documentation and tools for the UNH iSCSI code is confusing. Even when you get UNH iSCSI running it's unclear what it's doing and how. So we look at 2 other projects which complement each other: The Linux iSCSI Target Implementation from Ardis Technologies and the Linux-iSCSI Project (software initiator). We will look at these 2 projects for testing and playing with iSCSI.

The Basics

Before we actually build anything, you need to understand a couple things. Lets look at how iSCSI addresses are formed and how iSCSI is used, some general terms, and overall methodology.

It's important to understand that iSCSI, unlike NAS, makes block devices available via the network. Effectively, you can mount block devices (disks) across an IP network to your local system and then use them like any other block device. Generally when you first use an iSCSI device your going to need to partition it, label it, and create a filesystem on it. Unlike NAS your kernel will be able to read and write to your new iSCSI device just as if it were a local hard disk and therefore you can use any filesystem you like (EXT2/3, JFS, XFS, ReiserFS, etc). Also, because it is being handled as a block device only one (1) system can use the iSCSI device at a time! (This changes if you use a global filesystem, or read-only filesystem.) So just like Fibre Channel your not making 4 machines access 1 filesystem, but instead your allocating 4 chunks of disk on one large device and making it avalible via an iSCSI target so that 4 initiators can access it.

All devices in an iSCSI enviroment will have addresses. Think of addresses in iSCSI as being the iSCSI equivelent to a Fibre Channel WWN. Every address must be unique. Initiators will have addresses, and targets will have addresses. When you define a target you can specify the address yourself. When you use an initiator the address is typically defined for you. For instance, on my test setup used for this tutorial I have the following addresses:

```
On Nexus, Linux System:
iqn.1987-05.com.cisco:01.54197b8f9a8 -> Linux Initiator Address
iqn.1997-06.com.homestead:storage.disk1.nexus -> iSCSI Target Disk on Nexus
iqn.1997-06.com.homestead:storage.disk2.drew -> iSCSI Target Disk for Drew
iqn.1997-06.com.homestead:storage.raid.stripe0 -> iSCSI Target RAID

On Blade, Solaris System:
iqn.1987-05.com.cisco:01.f554845210af -> Solaris Initiator Address
```

iSCSI uses the following form for addresses, as found documented in IETF iSCSI Protocol Draft 20.

```
iSCSI Address Form:
   The following are examples of iSCSI qualified names that might be
   generated by "EXAMPLE Storage Arrays, Inc."

                     Naming     String defined by
        Type  Date    Auth      "example.com" naming authority
       +--++-----+ +---------+ +--------------------------------+
       |  ||     | |         | |                                |

       iqn.2001-04.com.example:storage:diskarrays-sn-a8675309
       iqn.2001-04.com.example
       iqn.2001-04.com.example:storage.tape1.sys1.xyz
       iqn.2001-04.com.example:storage.disk2.sys1.xyz
```

In the above examples, you can see the form. There are two different address types, iqn (iSCSI Qualified Name) and eui (IEEE EUI-64 format). The date field is the date of the first full month that your naming authorities domain name was registered. The Naming Auth is the naming authority (domain name) for this target, reversed. Following the naming authority is a colon, after which you can put anything you want to help you better organize your resourced. In the example above (#2) this field is left off completely which is legal. In examples 1 and 3 above you can see different naming conventions being used to clarify the target resources type. But in both cases the date and naming authority are the same. Cuddletech was registered on Jan 30th, 1999, so if I shared a 20G target I might use the target address: "iqn.1999-02.com.cuddletech:public.dump.20g-Seagate.jfs". So far I have not seen an implementation that actually used the naming authority for resolution, so it doesn't really matter what the naming authority is. Most Cisco targets and initiators will get the naming authority of cisco.com, and thats not a problem. You'll almost always be asked for the IP address of the iSCSI target, so feel free to name your targets anything you want so long as it follows the convension above.

Some other terms you may want to know include "portals" and "discovery". An iSCSI portal is a targets IP and TCP port number pair. Typically, if the port number is not specified it is defaulted to 3260. Discovery, or auto-discovery, is the proccess of an initiator asking a target portal for a list of it's targets and thus making those available to the initator for use. Most of the time using discovery on a target portal is the best way to get connected, and you'll see this done when we setup an initiator later. You can, however, alternately specify specifically the portal and target you wish to use. You can also use iSNS (the Internet Storage Name Service) for discovery instead of contacting a specific portal. iSNS is effectively like DNS for network storage devices. Finally, one kool thing you can do, especially if your using iSCSI as a method to extend your FC-SAN, is to allow multipathing for fault-tolerance. In such as case, a storage device would be avalible as a target on 2 or more target portals (or specified as such in your iSNS directory) and if one server died, you could continue to access the device thru the other portals. But doing such is beyond the scope of this tutorial.

Setting up a Target

Using the Linux iSCSI Target Implemention from Ardis Technologies [http://www.ardistech.com/iscsi/] we can create iSCSI targets on any Linux system which can be accessed from the iSCSI initiator we'll set up in the next section. A system that will make targets available only needs some partition to be available for export, and it does not need to be a SCSI disk.

Changes to the Ardis Target Implementation

Ming Zhang was nice enough to inform me that he and several others have forked the Ardis codebase and are adding significant improvements to the target implementation. The new projected is called the iSCSI Enterprise Target [http://sourceforge.net/projects/iscsitarget/] project.

Ming and his team are doing some really great work and they deserve the respect and appreciation of the whole community. Future revisions of this document will be based solely on the new implementation.

The following are the steps to setup an iSCSI target.

Procedure 1. Setting up an ISCSI Target

1. Download the iSCSI Target code from: http://www.ardistech.com/iscsi/

   ## Warning

   I had problems with the 20040211 release, when loading the modules I'd get unresolved symbols. Using the January released worked fine.

2. Download and configure (make menuconfig) a Linux 2.4.22 kernel. Patch the kernel with the diff in the target code (ie: kernel.2.4.22.diff).

```
[benr@nexus linux-2.4.22]$ patch -p0 <
        ../linux-iscsi-target-20040116/kernel.2.4.22.diff
patching file include/linux/mm.h
patching file include/linux/pagemap.h
patching file kernel/ksyms.c
patching file mm/filemap.c
[benr@nexus linux-2.4.22]$
```

3. There are no iSCSI options you need to select except to have the normal SCSI driver built. Just build and install your kernel in the usual way. Then boot it.

4. Next, build the target module and tools source. You'll need to define the environmental variable KERNELSRC to point to the location of the kernel tree you patched and installed.

```
[benr@nexus linux-iscsi-target-20040116]$ make
set $KERNELSRC!
[benr@nexus linux-iscsi-target-20040116]$ export KERNELSRC=../linux-2.4.22
[benr@nexus linux-iscsi-target-20040116]$ make
 ( ... Removed for clarity ... )
[benr@nexus linux-iscsi-target-20040116]$ make install
 ( ... Removed for clarity ... )
```

   This will install the iSCSI target module (iscsi_trgt_mod.o) into /lib/modules and the target daemon (iscsi_trgtd) into /usr/sbin.

5. Copy the iscsid.config file to /etc, and edit it to meet your needs.

```
[benr@nexus /etc]$ cat iscsid.config
## iSCSId Target Configuration File
User benr
```

```
Target iqn.1997-06.com.homestead:storage.disk1.nexus
        User
        # Lun definition
        # (right now only block devices are possible)
        Lun 0 /dev/sdb
        # Alias name for this target
        Alias Test


Target iqn.1997-06.com.homestead:storage.disk2.drew
        User
        # Lun definition
        # (right now only block devices are possible)
        Lun 0 /dev/sdc
        # Alias name for this target
        Alias NTFS
```

6. Copy the iscsid.rc to /etc/init.d/iscsid and make it executable.

7. Now, you should be able to start the iSCSI Targets.

```
[root@nexus /root]# /etc/init.d/iscsid start
Starting iSCSI target.
[root@nexus /root]#
```

As defined in the above procedure, according to my iscsid.conf, I'm offering 2 iSCSI targets. I can use these target names later when I connect to them with an initiator.

Keep an eye on the syslog when you start the iSCSI Target Daemon (tail -f /var/log/messages) to watch for startup messages and any warnings.

Setting up an Initiator

The Linux-iSCSI Proejct [http://linux-iscsi.sourceforge.net/] provides a driver, daemon, and tools for their iSCSI software initiator. This project is the Open Source version of the Cisco iSCSI driver intended for use with the Cisco SN 5428-2 Storage Router, but can be used with any iSCSI target. This driver is also available in a closed form for Solaris and HP-UX from Cisco.

The following are the steps to set up a software initiator.

Procedure 2. Setting up an iSCSI Initiator

1. Download the appropriate tarball from http://sourceforge.net/project/showfiles.php?group_id=26396. Make sure that the version you download corresponds with the kernel you are using, otherwise you'll have all types of strange problems. If you are also running an iSCSI target on this system you'll want to use the 2.4 Production Source (3.4.2 as of this writing).

2. Untar the source tarball and edit the Makefile. Change KERNEL_CONFIG and TOPDIR to match the location of your kernel source tree if your kernel tree isn't in the "usual place" (/usr/src/linux).

3. Compile the source with "make" and then as root install with "make install". This will install all the tools, daemons, init script, sample config file, and modules into place.

```
[root@nexus linux-iscsi-3.4.2]# make install

Note: using kernel source from /lib/modules/2.4.22/build containing
kernel version 2.4.22

Note: using kernel config from /lib/modules/2.4.22/build/.config

Installing iSCSI driver for Linux 2.4.22

The initialization script has been installed as /etc/rc.d/init.d/iscsi.

iSCSI has been set up to run automatically when you reboot.

InitiatorName iqn.1987-05.com.cisco:01.9c871ac985a was generated and
written to /etc/initiatorname.iscsi.

Make sure you check and edit the /etc/iscsi.conf file!
```

4. Edit your /etc/iscsi.conf. If you aren't sure how to configure your initiator you can specify any username and password (assuming your not using authentication on your target) and use the DiscoveryAddress keyword to define an address to discover. If you created a target on this machine, specify it's address to discover.

```
[root@nexus /etc]# cat iscsi.conf
Username=benr
Password=fakepass

DiscoveryAddress=10.10.1.100
  Username=benr
  Password=somepass
```

5. Now start the initiator using the RC script. Watch syslog (tail -f /var/log/messages) while you do this.

```
[root@nexus /]# /etc/init.d/iscsi start
Starting iSCSI: iscsi iscsid fsck/mount
[root@nexus /]#
```

My syslog output is below. You may see errors and warnings during the startup, I have removed warnings and dates from the output below for clarity.

```
[root@nexus /]# tail -f /var/log/messages

iscsid[30112]: version 3.4.2 (16-Feb-2004)
iscsid[30112]: INBP boot check returned this_is_inbp_boot = 0
kernel: iSCSI: bus 0 target 1 = iqn.1997-06.com.homestead:storage.disk2.drew
kernel: iSCSI: bus 0 target 1 portal 0 = address 10.10.1.100 port 3260 group 1
kernel: iSCSI: bus 0 target 1 trying to establish session df2dc000 to portal 0,
        address 10.10.1.100 port 3260 group 1
kernel: iSCSI: bus 0 target 1 established session df2dc000 #1, portal 0,
        address 10.10.1.100 port 3260 group 1
kernel: iSCSI: bus 0 target 0 = iqn.1997-06.com.homestead:storage.disk1.nexus
kernel: iSCSI: bus 0 target 0 portal 0 = address 10.10.1.100 port 3260 group 1
kernel: iSCSI: bus 0 target 0 trying to establish session ca4ee000 to portal 0,
        address 10.10.1.100 port 3260 group 1
kernel: iSCSI: bus 0 target 0 established session ca4ee000 #1, portal 0,
        address 10.10.1.100 port 3260 group 1
kernel: scsi singledevice 1 0 0 0
kernel:   Vendor: LINUX     Model: ISCSI             Rev: 0
kernel:   Type:   Direct-Access                      ANSI SCSI revision: 03
kernel: Attached scsi disk sdh at scsi1, channel 0, id 0, lun 0
kernel: SCSI device sdh: 35566480 512-byte hdwr sectors (18210 MB)
kernel:  sdh: unknown partition table
kernel: scsi singledevice 1 0 1 0
kernel:   Vendor: LINUX     Model: ISCSI             Rev: 0
kernel:   Type:   Direct-Access                      ANSI SCSI revision: 03
kernel: Attached scsi disk sdi at scsi1, channel 0, id 1, lun 0
kernel: SCSI device sdi: 35566480 512-byte hdwr sectors (18210 MB)
kernel:  sdi: sdi1
```

6. If everything goes properly, your initiator should have auto-discovered the targets from your target daemon. You can use the command "iscsi-ls" to see which targets your connected to.

```
[root@nexus /]# iscsi-ls
*************************************************************************
        Cisco iSCSI Driver Version ... 3.4.2 (16-Feb-2004 )
*************************************************************************
TARGET NAME             : iqn.1997-06.com.homestead:storage.disk1.nexus
TARGET ALIAS            :
HOST NO                 : 1
BUS NO                  : 0
TARGET ID               : 0
TARGET ADDRESS          : 10.10.1.100:3260
SESSION STATUS          : ESTABLISHED AT Mon Apr 12 15:13:32 2004
NO. OF PORTALS          : 1
```

```
PORTAL ADDRESS 1        : 10.10.1.100:3260,1
SESSION ID              : ISID 00023d000001 TSID 100
*************************************************************************
TARGET NAME             : iqn.1997-06.com.homestead:storage.disk2.drew
TARGET ALIAS            :
HOST NO                 : 1
BUS NO                  : 0
TARGET ID               : 1
TARGET ADDRESS          : 10.10.1.100:3260
SESSION STATUS          : ESTABLISHED AT Mon Apr 12 15:13:32 2004
NO. OF PORTALS          : 1
PORTAL ADDRESS 1        : 10.10.1.100:3260,1
SESSION ID              : ISID 00023d000001 TSID 100
*************************************************************************
[root@nexus /]#
```

You'll notice in the above output that these 2 targets are using the names I defined in the Target daemon config file.

7. You can now partition and create a filesystem on the target in the usual ways. Use the syslog output to figure out the mapping from iSCSI target to Linux device name.

```
[root@nexus /]# fdisk /dev/sdh
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-17366, default 1):
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-17366, default 17366):
Using default value 17366

Command (m for help): p

Disk /dev/sdh: 64 heads, 32 sectors, 17366 cylinders
Units = cylinders of 2048 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/sdh1             1     17366  17782768   83  Linux

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: If you have created or modified any DOS 6.x
partitions, please see the fdisk manual page for additional
information.
Syncing disks.
[root@nexus /]# mkfs -t jfs /dev/sdh1
mkfs.jfs version 1.1.0, 20-Nov-2002
Warning!  All data on device /dev/sdh1 will be lost!

Continue? (Y/N) y
```

```
   \

Format completed successfully.

17782768 kilobytes total disk space.
[root@nexus /]# mount -t jfs /dev/sdh1 /iscsi
[root@nexus /]# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/hda1             4.8G  4.3G  286M  94% /
/dev/hda5              13G   11G  2.0G  85% /home
/dev/sdh1              17G  2.3M   16G   1% /iscsi
[root@nexus /]#
```

This should get your target mounted and ready to use!

A word about initiator device paths on Linux

There are 2 paths avalible for use, the standard Linux block device (ie: `/dev/sdx`, `/dev/sdx1`, etc) and the iSCSI device path (the `/dev/iscsi` tree). The devices in the iSCSI tree are just symlinks back to the standard Linux paths, however the iSCSI paths provide more clarity which is useful for troubleshooting. Specifically, because the iSCSI commands shown above do not report the Linux block device name, they do however show the bus number, target id, and lun. An iSCSI device path looks like this:

```
[root@nexus /]# ls -l /dev/iscsi/bus0/target0/lun0
total 0
lrwxrwxrwx    1 root     root            8 Apr 13 12:25 disk -> /dev/sdh
lrwxrwxrwx    1 root     root            8 Apr 13 12:25 generic -> /dev/sg7
lrwxrwxrwx    1 root     root            9 Apr 13 12:25 part1 -> /dev/sdh1
 ...
```

In this tutorial I have utilized the Linux block device name instead of the full iSCSI device path primarily due to formatting concerns (iSCSI paths run off the page). I *do not* recommend that you use the Linux block device directly for any reason. Failuring to utilize the iSCSI paths will only create confusion and impair troubleshooting.

Notes for using a Linux Initiator with NetApp Filer Targets

If you expect to use the initiator to connect to a NetApp Filer at some point you will need to add some extra parameters to your `/etc/iscsi.conf`. Here is the configuration file I use for connecting to a NetApp 840 Filer:

```
Username=benr
Password=nopassneeded

### NetApp 940 iSCSI Portal
DiscoveryAddress=10.10.2.240

### These lines are required for NetApp Filers
### If you do not add them you will not beable
###  to communicate with the filer!
Continuous=no
HeaderDigest=never
DataDigest=never
ImmediateData=yes
```

If the four parameters listed here are not present you'll get all sorts of strange timeouts and warning, and you'll even connect but never be able to access the LUNs. For further information about iSCSI on NetApp Filers, head over to now.netapp.com.

Accessing iSCSI Targets from non-Linux Hosts

You can obtain obtain iSCSI software initiators for a variety of different operating systems. You can get the a Windows initiator from Microsoft and other initiators (HP-UX, Solaris, etc) from Cisco. The Cisco initiator uses the exact same syntax as the Linux client. For Windows clients just have the client software try to auto-discover your system hosting the targets and go from there.

If you do plan to use targets on Solaris initiators, take note of disk settings (heads, disks, cylinders) on the Linux host before you setup the Solaris system. On Solaris you'll need to install the Cisco package, edit your `/etc/iscsi.conf` just like you did on your Linux initiator (the syntax is identical) and then start the initiator (**/etc/init.d/iscsi start**). Once the targets are avalible to your Solaris system you'll need to create the device paths (**devfsadm**) and then repartition and label the device (the **format** command). When you select the iSCSI target in the format tool you'll be told that it can't be recognized as a Solaris disk and will want you to configure it, and auto-layout won't work. You'll need to take the values that your linux system saw and use those. Most of the values can be defaulted. You can find the values you need in the `/proc` filesystem on your linux initiator, but it's probly easier to just look at the partition map on the device using **fdisk** and then use those values. At that point you can actually partition slices on the disk and label it for use.

Here is a quick look at adding an initiator on Solaris:

Procedure 3. Setting up a Solaris Initiator

1. Start by installing the Cisco iSCSI package for Solaris

```
# gzcat solaris-iscsi-3.3.5.tar.Z | tar xfv -
        (Lines removed)
# pkgadd -d .

The following packages are available:
  1  CSCOiscsi     Cisco iSCSI device driver
                   (sparc) 3.3.5

Select package(s) you wish to process (or 'all' to process
all packages). (default: all) [?,??,q]: 1
        (Lines removed)
```

2. Now check the initiator name so that you can configure your targets as needed.

```
# cat /etc/initiatorname.iscsi
InitiatorName=iqn.1987-05.com.cisco:01.7fcfb3a9ad78
```

3. After setting up your targets, start up the iSCSI initiator and see whats avalible.

```
# cat /etc/initiatorname.iscsi
InitiatorName=iqn.1987-05.com.cisco:01.7fcfb3a9ad78
# /etc/init.d/iscsi start
iSCSI is starting.
#
# iscsi-ls
********************************************************************************
TARGET NAME iqn.1992-08.com.netapp:sn.33582139
TARGET ID 0:
  ADDRESS = 10.10.2.240:3260, 1
  STATUS  = Connected 10.10.2.244:38784<->10.10.2.240:3260  6/6/2004 22:19:50
  SESSION = ISID 00023d000001  TSID 351  PID 6440
********************************************************************************
```

4. If you take a look at **format** you won't see our iSCSI block device because it hasn't been added to the devfs yet. Use **devfsadm** to create the devices. Note: I use the -vC options to **devfsadm** to show me what it does (-v: verbose) and to clean any old or unneeded entries from the tree (-C: Cleanup).

```
# format
Searching for disks...done


AVAILABLE DISK SELECTIONS:
       0. c0t0d0 <SUN18G cyl 7506 alt 2 hd 19 sec 248>
          /sbus@1f,0/SUNW,fas@e,8800000/sd@0,0
Specify disk (enter its number): ^D
```

```
# devfsadm -vC
devfsadm[6456]: verbose: mknod /devices/iscsipseudo/iscsi@0/sd@0,0:a 0l/3l/60640
devfsadm[6456]: verbose: symlink /dev/dsk/c1t0d0s0 -> ../../devices/iscsipseudo/iscsi@
devfsadm[6456]: verbose: mknod /devices/iscsipseudo/iscsi@0/sd@0,0:b 0l/3l/60640
devfsadm[6456]: verbose: symlink /dev/dsk/c1t0d0s1 -> ../../devices/iscsipseudo/iscsi@
devfsadm[6456]: verbose: mknod /devices/iscsipseudo/iscsi@0/sd@0,0:c 0l/3l/60640
devfsadm[6456]: verbose: symlink /dev/dsk/c1t0d0s2 -> ../../devices/iscsipseudo/iscsi@
devfsadm[6456]: verbose: mknod /devices/iscsipseudo/iscsi@0/sd@0,0:d 0l/3l/60640
devfsadm[6456]: verbose: symlink /dev/dsk/c1t0d0s3 -> ../../devices/iscsipseudo/iscsi@
devfsadm[6456]: verbose: mknod /devices/iscsipseudo/iscsi@0/sd@0,0:e 0l/3l/60640
devfsadm[6456]: verbose: symlink /dev/dsk/c1t0d0s4 -> ../../devices/iscsipseudo/iscsi@
devfsadm[6456]: verbose: mknod /devices/iscsipseudo/iscsi@0/sd@0,0:f 0l/3l/60640
devfsadm[6456]: verbose: symlink /dev/dsk/c1t0d0s5 -> ../../devices/iscsipseudo/iscsi@
devfsadm[6456]: verbose: mknod /devices/iscsipseudo/iscsi@0/sd@0,0:g 0l/3l/60640
devfsadm[6456]: verbose: symlink /dev/dsk/c1t0d0s6 -> ../../devices/iscsipseudo/iscsi@
devfsadm[6456]: verbose: mknod /devices/iscsipseudo/iscsi@0/sd@0,0:h 0l/3l/60640
devfsadm[6456]: verbose: symlink /dev/dsk/c1t0d0s7 -> ../../devices/iscsipseudo/iscsi@
devfsadm[6456]: verbose: mknod /devices/iscsipseudo/iscsi@0/sd@0,0:a,raw 0l/3l/20640
devfsadm[6456]: verbose: symlink /dev/rdsk/c1t0d0s0 -> ../../devices/iscsipseudo/iscsi
devfsadm[6456]: verbose: mknod /devices/iscsipseudo/iscsi@0/sd@0,0:b,raw 0l/3l/20640
devfsadm[6456]: verbose: symlink /dev/rdsk/c1t0d0s1 -> ../../devices/iscsipseudo/iscsi
devfsadm[6456]: verbose: mknod /devices/iscsipseudo/iscsi@0/sd@0,0:c,raw 0l/3l/20640
devfsadm[6456]: verbose: symlink /dev/rdsk/c1t0d0s2 -> ../../devices/iscsipseudo/iscsi
devfsadm[6456]: verbose: mknod /devices/iscsipseudo/iscsi@0/sd@0,0:d,raw 0l/3l/20640
devfsadm[6456]: verbose: symlink /dev/rdsk/c1t0d0s3 -> ../../devices/iscsipseudo/iscsi
devfsadm[6456]: verbose: mknod /devices/iscsipseudo/iscsi@0/sd@0,0:e,raw 0l/3l/20640
devfsadm[6456]: verbose: symlink /dev/rdsk/c1t0d0s4 -> ../../devices/iscsipseudo/iscsi
devfsadm[6456]: verbose: mknod /devices/iscsipseudo/iscsi@0/sd@0,0:f,raw 0l/3l/20640
devfsadm[6456]: verbose: symlink /dev/rdsk/c1t0d0s5 -> ../../devices/iscsipseudo/iscsi
devfsadm[6456]: verbose: mknod /devices/iscsipseudo/iscsi@0/sd@0,0:g,raw 0l/3l/20640
devfsadm[6456]: verbose: symlink /dev/rdsk/c1t0d0s6 -> ../../devices/iscsipseudo/iscsi
devfsadm[6456]: verbose: mknod /devices/iscsipseudo/iscsi@0/sd@0,0:h,raw 0l/3l/20640
devfsadm[6456]: verbose: symlink /dev/rdsk/c1t0d0s7 -> ../../devices/iscsipseudo/iscsi
```

5. Now the device exists. Use **format** to label and if needbe partition the device.

```
# format
Searching for disks...done

c1t0d0: configured with capacity of 19.99GB


AVAILABLE DISK SELECTIONS:
       0. c0t0d0 <SUN18G cyl 7506 alt 2 hd 19 sec 248>
          /sbus@1f,0/SUNW,fas@e,8800000/sd@0,0
       1. c1t0d0 <NETAPP-LUN-0.2 cyl 5118 alt 2 hd 16 sec 512>
          /iscsipseudo/iscsi@0/sd@0,0
Specify disk (enter its number): 1
selecting c1t0d0
[disk formatted]
Disk not labeled.  Label it now? y
        (blah blah blah)
format> quit
```

6. If everything looks good so far, create your filesystem(s) and mount it.

```
# newfs /dev/rdsk/c1t0d0s2
newfs: construct a new file system /dev/rdsk/c1t0d0s2: (y/n)? y
/dev/rdsk/c1t0d0s2:     41926656 sectors in 5118 cylinders of 16 tracks, 512 sectors
        20472.0MB in 394 cyl groups (13 c/g, 52.00MB/g, 6336 i/g)
super-block backups (for fsck -F ufs -o b=#) at:
 32, 107040, 214048, 321056, 428064, 535072, 642080, 749088, 856096, 963104,
Initializing cylinder groups:
.......
super-block backups for last 10 cylinder groups at:
 40894496, 41001504, 41108512, 41215520, 41322528, 41429536, 41536544,
 41643552, 41750560, 41857568,
# mount /dev/dsk/c1t0d0s2 /iscsi/
# df -h
Filesystem             size   used  avail capacity  Mounted on
/dev/dsk/c0t0d0s0       16G   2.4G    13G    16%    /
/proc                    0K     0K     0K     0%    /proc
mnttab                   0K     0K     0K     0%    /etc/mnttab
fd                       0K     0K     0K     0%    /dev/fd
swap                   487M    40K   487M     1%    /var/run
swap                   487M    40K   487M     1%    /tmp
/dev/dsk/c1t0d0s2       20G     9K    19G     1%    /iscsi
#
```

7. Lastly, if you want the iSCSI share automatically mounted at boot when the iscsi init script runs, update your /etc/vfstab like you usually would, but in the "Mount at Boot" column instead of "yes" or "no", place the identifier "iscsi" to note that it's an iSCSI share. My line looks like this:

```
/dev/dsk/c1t0d0s2 /dev/rdsk/c1t0d0s2 /iscsi     ufs     1       iscsi   -
```

8. Your done! Read and write stuff to the share to make sure everything is working properly and possibly even pull out a benchmark to see what thruput your getting. I personally recommend IOzone [http://www.iozone.org/].

Using Volumes as iSCSI Targets

The target implementation used earlier can be used only on block devices at the present. This means that we can create block devices and use those, allowing us to use Linux RAID, LVM, Vinum, EVMS or any other volume manager in order to create more efficient storage solutions. In the following procedure we'll use the Linux RAID tools to create a 4 disk RAID0 (striped) volume and then make it avalible as an iSCSI target.

Warning

I am personally not a fan of the Linux RAID tools. I use them here simply because it's quick and easy, and so that this procedure is about iSCSI and not an LVM or EVMS tutorial. I highly suggest that if you plan to seriuosly use volumes for iSCSI that you consider using LVM, Vinum, or EVMS.

Procedure 4. Setting up RAID Targets

1. Ensure that Linux RAID (and LVM if you want it) are compiled into your kernel, and that the **mkraid** tool is installed.

2. Create a `raidtab` for your volume, such as the following.

```
[root@nexus /etc]# cat raidtab
raiddev /dev/md0
        raid-level      0
        nr-raid-disks   4
        persistent-superblock 1
        chunk-size     8

        device          /dev/sdd1
        raid-disk       0
        device          /dev/sde1
        raid-disk       1
        device          /dev/sdf1
        raid-disk       2
        device          /dev/sdg1
        raid-disk       3
[root@nexus /etc]#
```

3. Create the volume.

```
[root@nexus /etc]# mkraid /dev/md0
handling MD device /dev/md0
analyzing super-block
disk 0: /dev/sdd1, 17782768kB, raid superblock at 17782656kB
disk 1: /dev/sde1, 17782768kB, raid superblock at 17782656kB
disk 2: /dev/sdf1, 17782768kB, raid superblock at 17782656kB
disk 3: /dev/sdg1, 17782768kB, raid superblock at 17782656kB
[root@nexus /etc]#
```

4. Add the metadevice to your `/etc/iscsid.conf`.

```
# 4 Disk Linux RAID Stripe
Target iqn.1997-06.com.homestead:storage.raid.stripe0
        User
        Lun 0 /dev/md0
        Alias RAID0
```

5. If iSCSI is in use, stop both the initiator and the target daemons now.

```
[root@nexus /etc]# /etc/init.d/iscsi stop
Stopping iSCSI: sync umount sync iscsid iscsi
[root@nexus /etc]# /etc/init.d/iscsid  stop
Stopping iSCSI target.
```

6. Start the target daemon and then the initiator. Watch syslog while you do this to make sure your new target is auto-discovered.

```
[root@nexus /etc]# /etc/init.d/iscsid start
Starting iSCSI target.
[root@nexus /etc]# /etc/init.d/iscsi start
Starting iSCSI: iscsi iscsid fsck/mount
```

In syslog (`/var/log/messages`) we see a new target auto-discovered.

```
kernel: scsi singledevice 1 0 2 0
kernel:   Vendor: LINUX     Model: ISCSI             Rev: 0
kernel:   Type:   Direct-Access                      ANSI SCSI revision: 03
kernel: Attached scsi disk sdj at scsi1, channel 0, id 2, lun 0
kernel: SCSI device sdj: 142261248 512-byte hdwr sectors (72838 MB)
kernel:  sdj: unknown partition table
```

7. Now partition, label and create a filesystem on the new target.

```
[root@nexus /etc]# fdisk /dev/sdj
        ( Removed for Clarity )
Command (m for help): p

Disk /dev/sdj: 64 heads, 32 sectors, 69463 cylinders
Units = cylinders of 2048 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-69463, default 1):
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-69463, default 69463):
```

```
Using default value 69463

Command (m for help): p

Disk /dev/sdj: 64 heads, 32 sectors, 69463 cylinders
Units = cylinders of 2048 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/sdj1             1     69463  71130096   83  Linux

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.

        ( Removed for Clarity )
Syncing disks.
[root@nexus /etc]# mkfs -t jfs /dev/sdj1
mkfs.jfs version 1.1.0, 20-Nov-2002
Warning!  All data on device /dev/sdj1 will be lost!

Continue? (Y/N) y
   \

Format completed successfully.

71130096 kilobytes total disk space.
```

8. Now mount it.

```
[root@nexus /]# mkdir /iscsi_raid
[root@nexus /]# mount -t jfs /dev/sdj1 /iscsi_raid
[root@nexus /]# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/hda1             4.8G  4.3G  286M  94% /
/dev/hda5              13G   11G  2.0G  85% /home
/dev/sdj1              68G  8.7M   67G   1% /iscsi_raid
```

9. Done! Put some data on the new device and watch all the lights blink while you do writes, run some benchmarks, or just watch iostat data while you copy data onto the device.

```
[root@nexus benr]# iostat -kt
Linux 2.4.22 (nexus.homestead.com)      04/13/2004

        ( Removed for Clarity )

avg-cpu:  %user   %nice    %sys %iowait   %idle
           7.00    0.00   50.00    0.00   43.00

Device:            tps    kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
dev3-0          205.00      8488.00      2248.00       8488       2248
dev8-0            0.00         0.00         0.00          0          0
```

```
    dev8-1          0.00         0.00         0.00          0          0
    dev8-2          0.00         0.00         0.00          0          0
    dev8-3        122.00         0.00      3604.00          0       3604
    dev8-4        120.00         0.00      3596.00          0       3596
    dev8-5        121.00         0.00      3604.00          0       3604
    dev8-6        122.00         0.00      3600.00          0       3600
    dev8-7          0.00         0.00         0.00          0          0
    dev8-8          0.00         0.00         0.00          0          0
    dev8-9        126.00         0.00     14404.00          0      14404
```

Cisco iSCSI Initiator Command Reference

As mentioned earlier, the Linux-iSCSI Initiator implementation is simply the Open Source version of the Cisco closed source driver avilable for several platforms. Here is a quick overview of it's diffrent tools.

Cisco iSCSI Initiator Tools

iscsi-iname A iSCSI initiator address generator. Using `/dev/random` and the system clock it creates a (hopefully) unique address that can be used for initiator setup. This tool is used for generating the initial iSCSI initiator address stored in `/etc/initiatorname.iscsi`

```
[root@nexus /]# iscsi-iname
iqn.1987-05.com.cisco:01.8494d06dc35d
[root@nexus /]#
```

Warning

Some iSCSI devices implement access control based on the initiators iSCSI address. *Do not* use use the address supplied by iscsi-iname. Your true initator address is stored in the file `/etc/initiatorname.iscsi`. The address in this file will be returned to the portal as your valid address, and therefore should always be used. The address in that file can in fact be changed but it must be unique. Using the default is recommended.

iscsi-ls Display details about avalible targets. With no arguments you can view all the avilible targets. Using -l you can view LUN information for each target. Using -c you can view the target configuration information (timeouts, burst lengths, etc). Both -b and -t provide ways to query a specific target either by the local busId or by the target ID (address) itself.

```
[root@nexus /]# iscsi-ls
***********************************************************
        Cisco iSCSI Driver Version ... 3.4.2 (16-Feb-2004 )
***********************************************************
TARGET NAME             : iqn.1997-06.com.homestead:stora..
TARGET ALIAS            :
HOST NO                 : 1
BUS NO                  : 0
TARGET ID               : 0
TARGET ADDRESS          : 10.10.1.100:3260
SESSION STATUS          : ESTABLISHED AT Tue Apr 13 12:25:..
NO. OF PORTALS          : 1
PORTAL ADDRESS 1        : 10.10.1.100:3260,1
SESSION ID              : ISID 00023d000001 TSID 100
************************************************************
```

iscsi-device

Display tab delimited output detailing information reguarding an iSCSI block device, including target address, target IP and port, LUN number, etc. This is just a diffrent way of getting the same info you can get form **iscsi-ls**.

```
[root@nexus /]# df -h
Filesystem           Size  Used Avail Use% Mounted on
/dev/hda1            4.8G  4.3G  285M  94% /
/dev/hda5             13G   11G  2.0G  85% /home
/dev/sdj1             68G  391M   67G   1% /iscsi_raid
[root@nexus /]# iscsi-device /dev/sdj
/dev/sdj: 0   2   0        10.10.1.100   3260
        iqn.1997-06.com.homestead:storage.raid.stripe0
```

iscsi-mountall

The iSCSI equivelent to mountall. It attempts to mount all the file systems listed in the `/etc/fstab.iscsi` file, which uses the same syntax as the standard system fstab.

```
[root@nexus /]# df -h
Filesystem           Size  Used Avail Use% Mounted on
/dev/hda1            4.8G  4.3G  285M  94% /
/dev/hda5             13G   11G  2.0G  85% /home
[root@nexus /]# cat /etc/fstab.iscsi
# /etc/fstab.iscsi file for filesystems built on iscsi devices.
#iSCSI LUN      Mount Point     FS      Options  Dump Pass
#---------      -----------     ---     -------  ---- ----

/dev/sdj1       /iscsi_raid     jfs     defaults    0    0
[root@nexus /]# iscsi-mountall
fsck.jfs version 1.1.0, 20-Nov-2002
```

```
The current device is:  /dev/sdj1
Block size in bytes:  4096
File system size in blocks:  17782524
Phase 0 - Replay Journal Log
File system is clean.

[root@nexus /]# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/hda1             4.8G  4.3G  285M  94% /
/dev/hda5              13G   11G  2.0G  85% /home
/dev/sdj1              68G  391M   67G   1% /iscsi_raid
[root@nexus /]#
```

iscsi-umountall The converse of **iscsi-mountall**. It unmounts all filesystems specified in `/etc/fstab.iscsi`.

Resources and Further Reading

- Linux iSCSI Target Implementation [http://www.ardistech.com/iscsi/]: A Open Source Linux Target implementation by Ardis Technologies.
- Linux iSCSI Project [http://linux-iscsi.sourceforge.net/]: An Open Source driver and daemon for iSCSI on Linux
- UNH-iSCSI [http://unh-iscsi.sourceforge.net/]: An Open Source iSCSI target and initiator project by the University of New Hampshire.
- Cisco iSCSI Drivers [http://www.cisco.com/pcgi-bin/tablebuild.pl/sn5420-scsi]: Official drivers from Cisco. Get the Solaris initiator here. You must have a customer login to download.
- Microsoft iSCSI Home [http://www.microsoft.com/windows/storage/iscsi.mspx]: Micro$osft resources for using iSCSI and iSNS.
- IETF IP Storage Charter [http://www.ietf.org/html.charters/ips-charter.html]: The IETF overview of where IP Storage is going and whats in the works. Find all the IETF Drafts here.
- IP Storage Network Trends [http://www.iscsistorage.com/index.html]: Your general run of the mill marketing site full of crap, ads, and maybe 1 or 2 useful tidbits.
- Linux iSNS Project [http://sourceforge.net/projects/linuxisns]: An Open Source iSNS Server and Client for Linux (and Windows)

- A Technical Overview of iSNS [http://www.san-ip.com/iSCSI/techoverviewart.htm]: A decent read on the topic.