iSCSI

You may have heard of iSCSI in the context of corporate SANs perhaps supplying storage to a VMWare cluster of hosts, or some other heavy-weight application. These systems would generally involve running iSCSI over specialist 10Gbe (or more?) hardware and cabling and allow the operators to divorce the supply of storage from the hosts that use it. If that's your interest then this guide is probably not for you.

This guide is for people wanting to make use of iSCSI to share a DISK on a network between two machines. There are a number of reasons why you might want to do this instead of resorting to the probably more familiar file sharing mechanisms such as Samba and NFS but I can immediately think of one: You want the flexibility of a disk that's available on the network, without all the hassle of dealing with the edge cases that the big two file share protocols (CIFS/NFS) introduce. Since it's a SCSI-command (block level) protocol here are some common problems that will simply disappear with iSCSI:

- UID/GID mismatch issues between client/server
- File perms mismatch resulting in 'unsupported operation' type errors
- The typical 're-export' issues when exporting an imported network file system as something else (capability mismatch)
- Problems with 'special' files: pipes, unix domain sockets

Since you have a block-level protocol there is really nothing to configure. All you have to decide on is a Linux file system.

Nomenclature

The machine hosting the disk, is called an **iSCSI server** The disk itself, a shared resource on said machine, is a **target**. The client to access the server over a network, is an **Initiator**.

Target Software

For the target software, we will use:

- The Linux kernel
- Kernel module target_core_mod
- Kernel module iscsi_target_mod
- targetcli-fb utility

The modules will be automatically loaded by Slackware-current, so nothing to do there.

I installed targetcli-fb with slkpg -s sbo targetcli-fb. This little utility controls the configuration of the kernel modules above.

Initiator Software

For the initiator we will use:

- The Linux kernel.
- module scsi_transport_iscsi
- module iscsi_tcp
- module libiscsi
- open-iscsi

The kernel modules are also loaded automatically by Slackware-current. open-iscsi SlackBuild on SBO has been recently fixed, so should install with slpkg -s sbo open-isci.

Target Configuration

First, get the initiator ID from the initiator (client):

```
# cat /etc/iscsi/initiatorname.iscsi
```

Mine looks like this:

```
InitiatorName=iqn.2005-03.org.open-iscsi:91871cf5e261
```

It will be important later.

Next we need to configure our target (that's the 'server' remember?). In order to do this we use something called target-cli, so get targetcli-fb from Slackbuilds and install it:

```
slpkg -s sbo targetcli-fb
```

Targetcli requires a directory to store its data, make sure it exists:

```
# mkdir /etc/target
```

targetcli is a bit like a command-line directory browser except you use it to change values in a hierarchical store. That same hierarchy also contains the commands you invoke to make changes, however. You've probably not seen anything like it before and it's best just described by example.

Start by launching targetcli

```
# targetcli
targetcli shell version 2.1.51
Copyright 2011-2013 by Datera, Inc and others.
For help on commands, type 'help'.

/>
```

Assuming you have a spare (unused) disk, you can now create a block device, e.g.:

```
/> backstores/block create name=myblockdevice dev=/dev/sdb
Created block storage object block using /dev/sdb.
```

Next move to the iscsi target creation step. Just type create and allow the target to be given a generated name, it's good enough for our purposes. You can give your own name if you want but I saw little point for my setup.

```
/> iscsi/
/iscsi> create
created target iqn.2003-01.org.linux-iscsi.iscsi-
server.x8664:sn.7f83425fbbe8
Created TPG 1.
Global pref auto_add_default_portal=true
Created default portal listening on all IPs (0.0.0.0), port 3260
```

Next create LUN for the created target. Type the name of the target/tpg1 to move to the configuration area:

```
/iscsi> iqn.2003-01.org.linux-iscsi.iscsi-server.x8664:sn.7f83425fbbe8/tpg1
```

Then configure it:

```
/iscsi/iqn.20....425fbbe8/tpg1> luns/ create /backstores/block/myblockdevice
```

Then add an ACL by moving into the acl area and creating an ACL which matches the name of the initiator name that you got from the client:

```
/iscsi/iqn.20....425fbbe8/tpg1> acls/
/iscsi/iqn.20....be8/tpg1/acls> create iqn.2005-03.org.open-
iscsi:91871cf5e261
```

Simply exit to save your changes:

```
/> exit
Global pref auto_save_on_exit=true
Last 10 configs saved in /etc/target/backup.
Configuration saved to /etc/target/saveconfig.json
#
```

Service Setup

TO BE CONTINUED....

Sources

- Originally written by User bifferos

howtos, iSCSI