iSCSI

Introduction

This page summarises configuring iSCSI on Debian 6 and 7. Thanks to HowtoForge's excellent *Using iSCSI On Debian Squeeze (Initiator And Target)* fro m which most of the information was learned.

Overview

iSCSI allows a server to provide a virtual block device over a network to a client. The virtual block device can then be treated like a real block device – for example it can be partitioned and file systems created in the partitions.

In iSCSI terminology the server is a "target" and the client is an "initiator". On this page they are called server/target and client/initiator.

Links

- HowtoForge's "Using iSCSI On Debian Squeeze (Initiator And Target)": http://www.howtoforge.com/using-iscsi-on-debian-squeeze-initiator-and-target
- Open-iSCSI: http://www.open-iscsi.org/ including a README: http://www.open-iscsi.org/docs/README
- iSCSI Enterprise Target: http://iscsitarget.sourceforge.net/

Planning

You will need:

- A user name and password (for the iSCSI configuration so a free choice. There may be a limit of 16 characters on the password).
- A server/target computer:
    - root access.
    - The IP address. If there is more than one, the one that will be used by the client/initiator to access it.
    - A local block device to be made available to the initiator (client) via iSCSI. May be a file, a HDD (whole device or partition), an LVM volume or a RAID device.
- A client/initiator computer
    - root access.

Setting up the server/target

Installation

```
aptitude -y install iscsitarget iscsitarget-dkms
```

Configuration

Optionally backup the configuration files that will be changed: /etc/default/iscsitarget and /etc/iet/ietd.conf.

```
sed -i 's/ISCSITARGET_ENABLE=false/ISCSITARGET_ENABLE=true/' /etc/default/iscsitarget
```

The next step sets up to serve a single LVM volume, /dev/vg0/lv0. Values that need to be changed are red. The user and password values are need when configuring the client/initiator. Values that are arbitrary strings (so could be changed) are in blue.

```
user=someone
password=secret
local_device=/dev/vg0/lv0

oIFS=$IFS; array=($(hostname --long)); IFS=$oIFS
for ((i=${#array[*]};i>0;i--)); do backwards_fqdn+=.${array[i-1]}; done

( echo "Target iqn.$(date +%Y-%m)$backwards_fqdn:storage.lun0"
    echo "    IncomingUser $user $password"
    echo "    OutgoingUser"
    echo "    Lun 0 Path=$local_device,Type=fileio"
    echo "    Alias LUN0"
) > /etc/iet/ietd.conf
```

It can be useful to know the Target value just created when configuring the client/initiator. It can be displayed with

```
head -1 /etc/iet/ietd.conf
```

Further devices can be added by editing /etc/iet/ietd.conf, replicating and modifying the first stanza.

Setting up the client/initiator

Installation

```
aptitude -y install open-iscsi
```

Configuration

Optionally backup the configuration file that will be changed: /etc/iscsi/iscsid.conf.

```
sed -i 's/node.startup = manual/node.startup = automatic/' /etc/iscsi/iscsid.conf
```

In the next step, the iSCSI daemon is used to generate an initial configuration. Values that need to be changed are red. Starting the daemon will generate error messages because there's no configuration yet.

```
target_ip=192.168.10.27
/etc/init.d/open-iscsi restart
iscsiadm -m discovery -t st -p $target_ip
```

This should create a sub-directory of /etc/iscsi/nodes/ with the same name as the Target created when configuring the server/target.

Within that sub-directory there should be a further sub-directory with name beginning with the server/target's IP address.

Note: if the server/target has two IP address (accessible by the client/initiator?) there will be two such sub-sub-directories. It may be possible to configure a client/initiator to work this way but initial explorations did not identify how to do so. In this case, delete the sub-sub-directory for the IP address you do not want to use.

In the next step, the user name and password are added to the configuration.

Change to the new /etc/iscsi/nodes/<target>/<IP address ...> directory. In the commands below, the sed command should be on a single line.

```
user=someone
password=secret
sed -i "s/^node.session.auth.authmethod = None$/node.session.auth.authmethod = CHAP\nnode.session.auth.username =
$user\nnode.session.auth.password = $password/" default
```

Testing

```
/etc/init.d/open-iscsi restart
```

The output should include:

```
Login to [iface: default, target: <target>, portal: <ip address>,<port>]: successful
```

and a new /dev/sd[a-z]+ device file should have appeared.

Using the iSCSI-provided block device

The new /dev/sd[a-z]+ block device can be configured as desired.

If it is configured with file system(s) that should be mounted at boot there are two solutions dpending on whether the file systems should be fscked ...

/etc/fstab (fsck not possible)

/etc/fstab is used in the usual way with some special considerations:

- LABEL or UUID must be used in case the /dev/sd[a-z]+ name assigned by udev changes from boot to boot.
- The options must include _netdev. This ensures that mounting is deferred until the networking daemons (including open-iscsi) are running.
- The sixth field (fs_passno) must be set to 0. This disables fsck when fstab is processed, necessary because the devices are not created until later, when the open-iscsi boot script runs.

/etc/fstab-iscsi (fsck possible)

Create /etc/fstab-iscsi, based on this sample. The UUID can be found using the blkid command while the iSCSI-backed device is present:

```
# This is the configuration file for /etc/init.d/mountscsi.sh

# This file follows the same format as /etc/fstab.
# First column: it is strongly recommended that UIDs or LABELs are used.
# dump column: values may be omitted; if they are present they are ignored.
# pass column: 0 disables fsck; any other value (conventionally 1) enables it.

# <special>                               <mount point>  <type>  <options>  <dump>  <pass>
UUID=ff17c31e-eaff-4b49-b5f9-39ec81892e70 /mnt/hd/iSCSI  jfs     defaults           1
```

Install `/etc/init.d/mountscsi.sh` by creating the file with read and execute permission for root and writeable only by root (sorry about the formatting; apparently I can't drive Confluence WIKI) ....

```
#! /bin/bash
### BEGIN INIT INFO
# Provides: mountiscsi
# Required-Start: open-iscsi
# Required-Stop:
# Default-Start: S
# Default-Stop:
# Short-Description: Mounts file systems based on iSCSI-backed devices
# Description: Mounts file systems based on iSCSI-backed devices.
# As far as practicable, does with /etc/fstab-iscsi what
# mountall.sh does with /etc/fstab.
# In this version only ext* and JFS file systems are
# supported with fsck before mount.
### END INIT INFO

# Author: Charles Atkinson <c@charlesmatkinson.org>

. /lib/init/vars.sh
. /lib/lsb/init-functions
regex='^#|^$'
shopt -s extglob


do_start () {
 log_begin_msg "Mounting iscsi-backed filesystems"$'\n'

 my_fstab=/etc/fstab-iscsi
 [[ -r $my_fstab ]] || { log_failure_msg "Cannot read $my_fstab" exit 1; }

 # For each line of my fstab
 mount_fail=0
 line_n=0
 while read -r special mountpoint type options dump pass spurious
 do
 ((line_n++))
 [[ $special =~ $regex ]] && continue
 [[ $pass = '' ]] && pass=$dump
 if [[ $spurious != '' ]]; then
 log_warning_msg "Spurious data on line $line_n of $my_fstab: $spurious"
 continue
 fi
```

```
# Normalise the special value
if [[ $special =~ ^LABEL= || $special =~ ^UUID= ]]; then
buf=$(findfs "$special" 2>&1)
if (($?>0)); then
log_warning_msg "Cannot find the file system corresponding to '$special'"
continue
fi
special_norm=$buf
elif [[ $special =~ ^/dev/ ]]; then
buf=$(readlink --canonicalize -- "$special" 2>&1)
if (($?>0)); then
log_warning_msg "Problems running readlink for $special: $buf"
continue
fi
special_norm=$buf
else
log_warning_msg "Invalid special on $my_fstab, line $line_n: $special"
continue
fi
#echo "DEBUG: special:$special, mountpoint:$mountpoint,type:$type,options:$options, pass:$pass" >&2

# Check for existing mounts
mounted_flag=0
while read -r pm_fs_spec pm_mountpoint _
do
if [[ $pm_fs_spec = $special_norm ]]; then
mounted_flag=1
log_warning_msg "$special_norm is already mounted on $pm_mountpoint"
fi
done < <(cat /proc/mounts)

# Run file system check if possible and indicated
if ((mounted_flag==0)); then
fsck_flag=0
case $type in
ext* )
tune2fs_out=$(tune2fs -l "$special_norm" 2>&1)
if (($?>0)); then
log_warning_msg "Can't list $special_norm file system parameters: $tune2fs_out"
continue
fi

buf=${tune2fs_out##*Filesystem state:*( )}
filesystem_state=${buf%%$'\n'*}
[[ $filesystem_state != clean ]] && fsck_flag=1
#echo "DEBUG: filesystem_state:$filesystem_state" >&2

buf=${tune2fs_out##*Maximum mount count:*( )}
max_mount_count=${buf%%$'\n'*}

buf=${tune2fs_out##*Mount count:*( )}
mount_count=${buf%%$'\n'*}
((mount_count>max_mount_count)) && fsck_flag=1
#echo "DEBUG: max_mount_count:$max_mount_count, mount_count:$mount_count" >&2

buf=${tune2fs_out##*Check interval:*( )}
check_interval=${buf%% *}
#echo "DEBUG: check_interval:$check_interval" >&2
if ((check_interval>0)); then
buf=${tune2fs_out##*Last checked:*( )}
last_checked=${buf%%$'\n'*}
#echo "DEBUG: last_checked:$last_checked" >&2
last_checked_secs=$(date "--date=$last_checked" +%s)
now=$(date +%s)
#echo "DEBUG: last_checked_secs:$last_checked_secs, now:$now" >&2
(((now-last_checked_secs)>check_interval)) && fsck_flag=1
fi
;;
jfs )
jfs_tune_out=$(jfs_tune -l "$special_norm")
if (($?>0)); then
```

```
 log_warning_msg "Can't list file system parameters: $jfs_tune_out"
 continue
 fi

 buf=${jfs_tune_out##*JFS state:*([[:space:]])}
 jfs_state=${buf%%$'\n'*}
 #echo "DEBUG: jfs_state:$jfs_state" >&2
 [[ $jfs_state != clean ]] && fsck_flag=1
 ;;
 esac

 if ((fsck_flag==1)); then
 # fsck option -p is OK for fsck.ext[234] and for jfs_fsck`
 fsck -Tp "$special_norm"
 fi
 fi

 # Mount
 mount -t $type -o $options $special $mountpoint || mount_fail=1

 done < <(cat "$my_fstab")

 if ((mount_fail==0)); then
 exit 0
 else
 exit 1
 fi
}


case "$1" in
 start)
 do_start
 ;;
 restart|reload|force-reload)
 echo "Error: argument '$1' not supported" >&2
 exit 3
 ;;
 stop|"")
 # No-op
 ;;
 *)
 echo "Usage: mountiscsi.sh start" >&2
 exit 3
 ;;
esac
```

... and creating the required symlinks:

```
update-rc.d mountiscsi.sh defaults
```

That will generate two "do not match LSB Default" warnings which can be ignored.

Normal operations

In normal operations the client/initiator should be shut down before the server/target. Doing otherwise will result in a delayed shutdown by the client /initiator.

Issue investigation

How to identify which /dev/sd[a-z]+ are iSCSI devices

The easiest way is to list /dev/disk/by-path/:

```
ls -l /dev/disk/by-path/ | grep 'ip-.*iqn\.'
```

If lshw is installed, more information is available by

```
lshw -class disk -class storage
```

hdparm doesn't work on iSCSI devices. When smartctl was tried there was a server/target kernel abort task for iSCSI target.

Error messages

iscsiadm: initiator reported error (19 - encountered non-retryable iSCSI login failure)

As the messages suggest, an authentication failure. Check user name and password consistency between server/target and client/initiator.